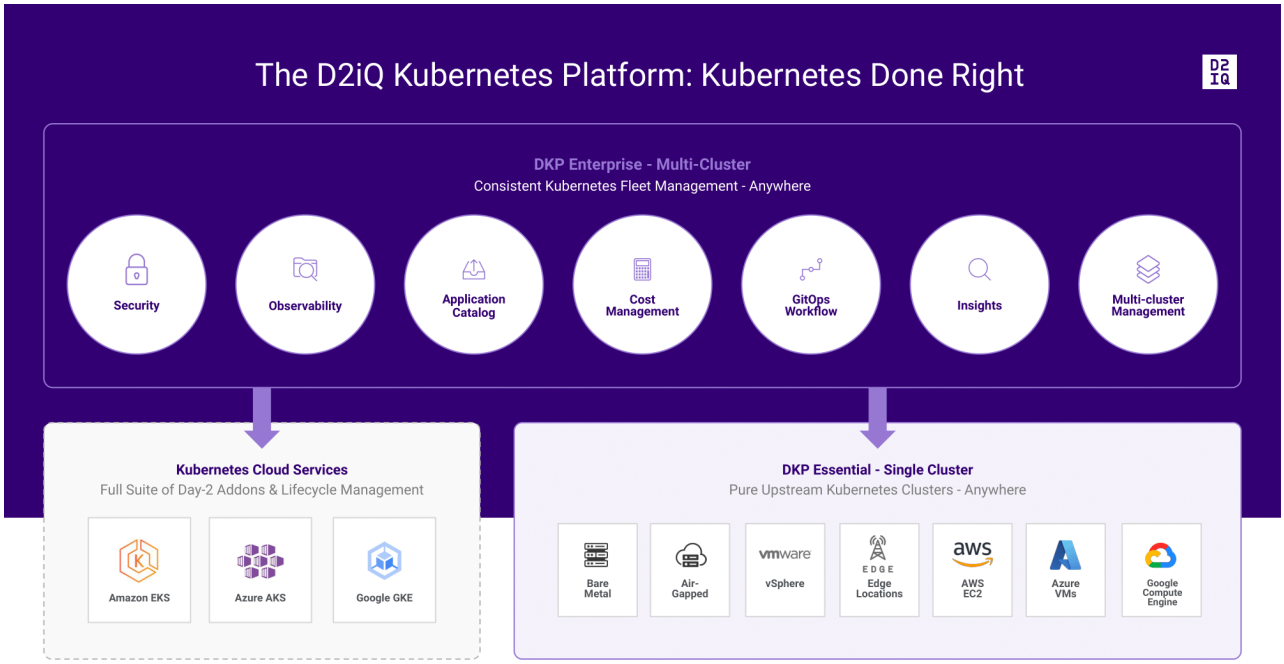
Introducing the D2iQ Kubernetes Platform

D2iQ Kubernetes Platform (DKP)

Exported on 07/21/2023

Table of Contents

As the leading independent platform for Kubernetes in production, the D2iQ Kubernetes Platform (DKP) provides a holistic approach and a complete set of enterprise-grade technologies, services, training, and support to build and run applications in production at scale. Built around the open-source Cluster API, the new version of DKP becomes the single, centralized point of control for an organization's application infrastructure, empowering organizations to more easily deploy, manage, and scale Kubernetes workloads in Day 2 production environments.



Features	Benefits
Container Orchestration	Leverage an industry standard distribution of open-source Kubernetes for cluster and container management.
Application Management and Deployment	Deploy applications and services within Kubernetes clusters with Helm[1].
Observability	Gain deep insight into your Kubernetes clusters and applications with open source metrics leveraging Telegraf, Prometheus, and Grafana[2].

1 https://helm.sh/
2 https://grafana.com/docs/

Features	Benefits
Cluster API	Automate cluster lifecycle management using Cluster API[3] to simplify the provisioning, upgrading, and operating multiple Kubernetes clusters across a wide range of distributions and virtual and physical environments.
Cluster Autoscaling	Save operational costs by automatically scaling down(see page 171) capacity when it's not needed and adding capability when there is greater demand, with CAPI enabled autoscaling groups.
Logging	Collect and analyze logs and metrics(see page 560) to ensure optimal performance and troubleshooting with Grafana, Loki, and Fluentbit.
Cloud-Native Scale Testing	Extensive integration and workload testing at massive scale with a wide range of workloads to ensure real-world preparedness.
Networking and Routing	Easily automate and expose application endpoints(see page 595) with Calico,Traefik, and CoreDNS.
Fine-Grained Cluster	Upgrades(see page 780) Reduce operational overhead with non-disruptive patching or parallel worker node upgrades.
Backup and Recovery	Ensure business continuity and disaster recovery(see page 552) with Velero[4].
Declarative Automated Installer With Day 2 Platform Services	Accelerate time-to-production on any infrastructure with consistency and reliability with the required platform services(see page 421) needed for Day 2 production.
Operate in Air-gapped Environments(see page 155)	Leverage declarative APIs to optimize cluster resources for cost, resilience, and performance.
End-to-End Support	Enterprise-grade support[5] and services for both Kubernetes and its supporting platform services. the entire platform including all components.

3 https://cluster-api.sigs.k8s.io/
4 https://velero.io/docs/main/
5 https://support.d2iq.com/hc/en-us

1 Quick Start Guides

Use the following guides to get started in your infrastructure.

1.1 AWS Quick Start

This page provides instructions for getting started with DKP, to get your Kubernetes cluster up and running with basic configuration requirements on an Amazon Web Services (AWS) public cloud instances. If you want to customize your AWS environment, see Install AWS Advanced(see page 111).

1.1.1 Prerequisites

Before starting the DKP installation, verify that you have:

- An x86_64-based Linux or macOS machine with a supported version of the operating system.
- The `dkp` binary for Linux, or macOS.
- Docker[6] version 18.09.2 or later.
- kubectl[7] for interacting with the running cluster.
- A valid AWS account with credentials configured[8].
- Resource requirements(see page 0)

1.1.2 Configure AWS prerequisites

Follow these steps:

1. Follow the steps in IAM Policy Configuration(see page 140).
2. Export the AWS region where you want to deploy the cluster:

   ```
   export AWS_REGION=us-west-2
   ```

3. Export the AWS Profile with the credentials that you want to use to create the Kubernetes cluster:

   ```
   export AWS_PROFILE=<profile>
   ```

4. Name your cluster.
   Give your cluster a unique name suitable for your environment. In AWS, it is critical that the name be unique as no two clusters in the same AWS account can have the same name.
   Set the environment variable to be used throughout this documentation:

6 https://docs.docker.com/get-docker/
7 https://kubernetes.io/docs/tasks/tools/#kubectl
8 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-profiles.htm

```
export CLUSTER_NAME=aws-example
```

The cluster name may only contain the following characters: `a-z`, `0-9`, `.`, and `-`. Cluster creation will fail if the name has capital letters. See Kubernetes[9] for more naming information.

1.1.2.1 Create a new AWS Kubernetes cluster

By default, the control-plane Nodes will be created in 3 different zones. However, the default worker Nodes will reside in a single Availability Zone. You may create additional node pools in other Availability Zones with the `dkp create nodepool` command.

Ensure your AWS credentials are up to date. If you are using Static Credentials, use the following command to refresh the credentials. Otherwise, proceed to step 1 below:

```
dkp update bootstrap credentials aws
```

If you use these instructions to create a cluster on AWS using the DKP default settings without any edits to configuration files or additional flags, your cluster is deployed on an Ubuntu 20.04 operating system image(see page 78) with 3 control plane nodes, and 4 worker nodes.

The default AWS image is not recommended for use in production. We suggest using DKP Image Builder to create a custom AMI(see page 324) to take advantage of enhanced cluster operations, and to explore the advanced AWS installation(see page 113) topics for more options.

1. Create a Kubernetes cluster:
   NOTE: To increase Docker Hub's rate limit[10] use your Docker Hub credentials when creating the cluster, by setting the following flag `--registry-mirror-url=https://registry-1.docker.io --registry-mirror-username= --registry-mirror-password=` on the `dkp create cluster` command.

   ```
   dkp create cluster aws \
   --cluster-name=${CLUSTER_NAME} \
   --additional-tags=owner=$(whoami) \
   --with-aws-bootstrap-credentials=true \
   --self-managed
   ```

2. Verify your output is similar to the following:

9 https://kubernetes.io/docs/concepts/overview/working-with-objects/names/
10 https://docs.docker.com/docker-hub/download-rate-limit/

```
 ✓ Creating a bootstrap cluster
 ✓ Initializing new CAPI components
Generating cluster resources
cluster.cluster.x-k8s.io/aws-example created
awscluster.infrastructure.cluster.x-k8s.io/aws-example created
kubeadmcontrolplane.controlplane.cluster.x-k8s.io/aws-example-control-plane
created
awsmachinetemplate.infrastructure.cluster.x-k8s.io/aws-example-control-plane
created
secret/aws-example-etcd-encryption-config created
machinedeployment.cluster.x-k8s.io/aws-example-md-0 created
awsmachinetemplate.infrastructure.cluster.x-k8s.io/aws-example-md-0 created
kubeadmconfigtemplate.bootstrap.cluster.x-k8s.io/aws-example-md-0 created
clusterresourceset.addons.cluster.x-k8s.io/calico-cni-installation-aws-example
created
configmap/calico-cni-installation-aws-example created
configmap/tigera-operator-aws-example created
clusterresourceset.addons.cluster.x-k8s.io/aws-ebs-csi-aws-example created
configmap/aws-ebs-csi-aws-example created
clusterresourceset.addons.cluster.x-k8s.io/cluster-autoscaler-aws-example
created
configmap/cluster-autoscaler-aws-example created
clusterresourceset.addons.cluster.x-k8s.io/node-feature-discovery-aws-example
created
configmap/node-feature-discovery-aws-example created
clusterresourceset.addons.cluster.x-k8s.io/nvidia-feature-discovery-aws-example
created
configmap/nvidia-feature-discovery-aws-example created
 ✓ Waiting for cluster infrastructure to be ready
 ✓ Waiting for cluster control-planes to be ready
 ✓ Waiting for machines to be ready
 ✓ Initializing new CAPI components
 ✓ Moving cluster resources
You can now view resources in the moved cluster by using the --kubeconfig flag
with kubectl. For example: kubectl --kubeconfig=/aws-example.conf get nodes
 ✓ Deleting bootstrap cluster

Cluster default/aws-example kubeconfig was written to to the filesystem.
You can now view resources in the new cluster by using the --kubeconfig flag
with kubectl.
For example: kubectl --kubeconfig=aws-example.conf get nodes
```

As part of the underlying processing, the DKP CLI:

- creates a bootstrap cluster
- creates a workload cluster
- moves CAPI controllers from the bootstrap cluster to the workload cluster, making it self-managed
- deletes the bootstrap cluster

To understand how this process works step by step, you can follow the workflow in Install AWS Advanced(see page 113).

1.1.3 Explore the new Kubernetes Cluster

The kubeconfig file is written to your local directory and you can now explore the cluster.

1. List the Nodes with the command:

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf get nodes
```

2. Verify the output is similar to:

```
NAME                                         STATUS   ROLES
AGE   VERSION
ip-10-0-108-63.us-west-2.compute.internal    Ready    <none>
59m   v1.22.8
ip-10-0-115-181.us-west-2.compute.internal   Ready    <none>
59m   v1.22.8
ip-10-0-118-159.us-west-2.compute.internal   Ready    <none>
59m   v1.22.8
ip-10-0-122-136.us-west-2.compute.internal   Ready    control-plane,master
60m   v1.22.8
ip-10-0-122-6.us-west-2.compute.internal     Ready    <none>
59m   v1.22.8
ip-10-0-154-239.us-west-2.compute.internal   Ready    control-plane,master
59m   v1.22.8
ip-10-0-199-233.us-west-2.compute.internal   Ready    control-plane,master
57m   v1.22.8
```

3. List the Pods with the command:

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf get pods -A
```

4. Verify the output is similar to:

```
NAMESPACE                           NAME
READY   STATUS    RESTARTS   AGE
calico-system                       calico-typha-665d976df-rf7jg
1/1     Running   0          60m
capa-system                         capa-controller-manager-697b7df888-vhcbj
2/2     Running   0          57m
capi-kubeadm-bootstrap-system       capi-kubeadm-bootstrap-controller-
manager-67d8fc9688-5p65s           1/1     Running   0          57m
capi-kubeadm-control-plane-system   capi-kubeadm-control-plane-controller-
manager-846ff8b565-jqmhd       1/1     Running   0          57m
capi-system                         capi-controller-manager-865fddc84c-9g7bb
1/1     Running   0          57m
cappp-system                        cappp-controller-manager-7859fbbb7f-xjh6k
1/1     Running   0          56m
...
```

1.1.4 Install and Log in to the DKP UI

You can now proceed to installing the DKP UI(see page 351) and applications. After installation, you will be able to log in(see page 385) to the DKP UI to explore it.

1.1.5 Delete the Kubernetes Cluster and Cleanup your Environment

If you no longer need the cluster and want to delete it, you can do so using the DKP CLI.

1. Delete the provisioned Kubernetes cluster:

```
dkp delete cluster \
--cluster-name=${CLUSTER_NAME} \
--with-aws-bootstrap-credentials=true \
--kubeconfig=${CLUSTER_NAME}.conf \
--self-managed
```

2. Verify the output is similar to:

```
 ✓ Creating a bootstrap cluster
 ✓ Initializing new CAPI components
 ✓ Moving cluster resources
You can now view resources in the moved cluster by using the --kubeconfig flag
with kubectl. For example: kubectl --kubeconfig=aws-example-bootstrap.conf get
nodes
 ✓ Waiting for cluster infrastructure to be ready
 ✓ Waiting for cluster control-planes to be ready
 ✓ Waiting for machines to be ready
 ✓ Deleting Services with type LoadBalancer for Cluster default/aws-example
 ✓ Deleting ClusterResourceSets for Cluster default/aws-example
  ✓ Deleting cluster resources
 ✓ Waiting for cluster to be fully deleted
Deleted default/aws-example cluster
 ✓ Deleting bootstrap cluster
```

Similar to `create cluster`, use the flag `--self-managed` with the `delete cluster` command:

- Creates a bootstrap cluster.
- Moves the CAPI controllers from the workload cluster back to the bootstrap cluster.
- Deletes the workload cluster.
- Deletes the bootstrap cluster.

To understand how this process works step by step, you can follow the workflow in Delete Cluster(see page 134).

1.2 Azure Quick Start

Get started by installing a cluster with default configuration settings on Azure

This Quick Start guide provides simplified instructions for using DKP to get your Kubernetes cluster up and running with minimal configuration requirements on an Azure public cloud instance. To customize your Azure installation, refer to Azure Advanced(see page 173) installation.

1.2.1 Prerequisites

Before starting the DKP installation, verify that you have:

- An x86_64-based Linux or macOS machine with a supported version of the operating system.
- The `dkp` binary on this machine.
- Docker[11] version 18.09.2 or later.
- kubectl[12] for interacting with the running cluster.
- Azure CLI[13].
- A valid Azure account with credentials configured[14].
- Resource requirements(see page 0)

1.2.2 Configure Azure Prerequisites

Follow these steps:

1. Log in to Azure:

```
az login
```


```
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "a1234567-b132-1234-1a11-1234a5678b90",
    "id": "b1234567-abcd-11a1-a0a0-1234a5678b90",
    "isDefault": true,
    "managedByTenants": [],
    "name": "Mesosphere Developer Subscription",
    "state": "Enabled",
    "tenantId": "a1234567-b132-1234-1a11-1234a5678b90",
    "user": {
      "name": "user@azuremesosphere.onmicrosoft.com",
      "type": "user"
    }
  }
]
```


11 https://docs.docker.com/get-docker/
12 https://kubernetes.io/docs/tasks/tools/#kubectl
13 https://docs.microsoft.com/en-us/cli/azure/install-azure-cli
14 https://github.com/kubernetes-sigs/cluster-api-provider-azure/blob/master/docs/book/src/topics/getting-started.md#prerequisites

2. Create an Azure Service Principal (SP) by running the following command:
Note: If you have more than one Azure account, run this command to ensure you are pointing to the correct Azure subscription ID: $(az account show --query id -o tsv)

If an SP with the name exists, this command will rotate the password.

```
az ad sp create-for-rbac --role contributor --name "$(whoami)-konvoy" --scopes=/
subscriptions/$(az account show --query id -o tsv)
```

```
{
  "appId": "7654321a-1a23-567b-b789-0987b6543a21",
  "displayName": "azure-cli-2021-03-09-23-17-06",
  "password": "Z79yVstq_E.R0R7RUUck718vEHSuyhAB0C",
  "tenant": "a1234567-b132-1234-1a11-1234a5678b90"
}
```

3. Set the required environment variables:

```
export AZURE_SUBSCRIPTION_ID="<id>"          # b1234567-abcd-11a1-a0a0-1234a5678b90
export AZURE_TENANT_ID="<tenant>"            # a1234567-b132-1234-1a11-1234a5678b90
export AZURE_CLIENT_ID="<appId>"             # 7654321a-1a23-567b-b789-0987b6543a21
export AZURE_CLIENT_SECRET="<password>"      # Z79yVstq_E.R0R7RUUck718vEHSuyhAB0C
```

4. Base64 encode the same environment variables:

```
export AZURE_SUBSCRIPTION_ID_B64="$(echo -n "${AZURE_SUBSCRIPTION_ID}" | base64 | tr
-d '\n')"
export AZURE_TENANT_ID_B64="$(echo -n "${AZURE_TENANT_ID}" | base64 | tr -d '\n')"
export AZURE_CLIENT_ID_B64="$(echo -n "${AZURE_CLIENT_ID}" | base64 | tr -d '\n')"
export AZURE_CLIENT_SECRET_B64="$(echo -n "${AZURE_CLIENT_SECRET}" | base64 | tr -d
'\n')"
```

1.2.3 Create a new Azure Kubernetes cluster

If you use these instructions to create a cluster on Azure using the DKP default settings without any edits to configuration files or additional flags, your cluster will be deployed on an Ubuntu 20.04 operating system image(see page 78) with 3 control plane nodes, and 4 worker nodes.

NOTE: The default Azure image is not recommended for use in production. We suggest using Konvoy Image Builder to create a custom image[15] to take advantage of enhanced cluster operations, and to explore the advanced Azure installation(see page 173) topics for more options.

15 https://d2iq.atlassian.net/wiki/spaces/DENT/pages/89327100/%282.3%29+KIB+with+Azure

1. Give your cluster a name suitable for your environment:

```
export CLUSTER_NAME=azure-example
```

2. Create a Kubernetes cluster:

NOTE: To increase Docker Hub's rate limit[16] use your Docker Hub credentials when creating the cluster, by setting the following flag `--registry-mirror-url=https://registry-1.docker.io --registry-mirror-username=<username> --registry-mirror-password=<password>` on the `dkp create cluster` command.

```
dkp create cluster azure \
--cluster-name=${CLUSTER_NAME} \
--additional-tags=owner=$(whoami) \
--self-managed
```

You will see output similar to the following:

```
Generating cluster resources
cluster.cluster.x-k8s.io/azure-example created
azurecluster.infrastructure.cluster.x-k8s.io/azure-example created
kubeadmcontrolplane.controlplane.cluster.x-k8s.io/azure-example-control-plane created
azuremachinetemplate.infrastructure.cluster.x-k8s.io/azure-example-control-plane
created
secret/azure-example-etcd-encryption-config created
machinedeployment.cluster.x-k8s.io/azure-example-md-0 created
azuremachinetemplate.infrastructure.cluster.x-k8s.io/azure-example-md-0 created
kubeadmconfigtemplate.bootstrap.cluster.x-k8s.io/azure-example-md-0 created
clusterresourceset.addons.cluster.x-k8s.io/calico-cni-installation-azure-example
created
configmap/calico-cni-installation-azure-example created
configmap/tigera-operator-azure-example created
clusterresourceset.addons.cluster.x-k8s.io/azure-disk-csi-azure-example created
configmap/azure-disk-csi-azure-example created
clusterresourceset.addons.cluster.x-k8s.io/cluster-autoscaler-azure-example created
configmap/cluster-autoscaler-azure-example created
clusterresourceset.addons.cluster.x-k8s.io/node-feature-discovery-azure-example
created
configmap/node-feature-discovery-azure-example created
clusterresourceset.addons.cluster.x-k8s.io/nvidia-feature-discovery-azure-example
created
configmap/nvidia-feature-discovery-azure-example created
```

As part of the underlying processing, the DKP CLI:

- creates a bootstrap cluster

16 https://docs.docker.com/docker-hub/download-rate-limit/

- creates a workload cluster
- moves CAPI controllers from the bootstrap cluster to the workload cluster, making it self-managed
- deletes the bootstrap cluster

1.2.4 Explore the new Kubernetes cluster

The kubeconfig file is written to your local directory and you can now explore the cluster.

1. List the Nodes with the command:

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf get nodes
```

You will see output similar to:

```
NAME                                 STATUS   ROLES                  AGE
VERSION
azure-example-control-plane-84htt    Ready    control-plane,master   8m11s
v1.22.7
azure-example-control-plane-r8srg    Ready    control-plane,master   4m17s
v1.22.7
azure-example-control-plane-wrdql    Ready    control-plane,master   6m15s
v1.22.7
azure-example-md-0-9crp9             Ready    <none>                 6m47s
v1.22.7
azure-example-md-0-dvx5d             Ready    <none>                 6m42s
v1.22.7
azure-example-md-0-gc9mx             Ready    <none>                 5m27s
v1.22.7
azure-example-md-0-tkqf7             Ready    <none>                 4m48s
v1.22.7
```

2. List the Pods with the command:

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf get pods -A
```

You will see output similar to:

```
NAMESPACE                           NAME
READY   STATUS    RESTARTS   AGE
calico-system                       calico-typha-665d976df-rf7jg
1/1     Running   0          60m
capa-system                         capa-controller-manager-697b7df888-vhcbj
2/2     Running   0          57m
capi-kubeadm-bootstrap-system       capi-kubeadm-bootstrap-controller-
manager-67d8fc9688-5p65s           1/1     Running   0          57m
capi-kubeadm-control-plane-system   capi-kubeadm-control-plane-controller-
manager-846ff8b565-jqmhd       1/1     Running   0          57m
capi-system                         capi-controller-manager-865fddc84c-9g7bb
1/1     Running   0          57m
```

```
cappp-system                        cappp-controller-manager-7859fbbb7f-xjh6k
1/1     Running   0          56m
...
```

1.2.5 Install and Log in to the DKP UI

You can now proceed to installing the DKP UI(see page 351) and applications. After installation, you will be able to log in(see page 385) to the DKP UI to explore it.

1.2.6 Delete the Kubernetes Cluster and Cleanup your Environment

Follow these steps:

1. Delete the provisioned Kubernetes cluster and wait a few minutes:

```
dkp delete cluster \
--cluster-name=${CLUSTER_NAME} \
--kubeconfig=${CLUSTER_NAME}.conf \
--self-managed
```

```
✓ Deleting Services with type LoadBalancer for Cluster default/azure-example
✓ Deleting ClusterResourceSets for Cluster default/azure-example
✓ Deleting cluster resources
✓ Waiting for cluster to be fully deleted
Deleted default/azure-example cluster
```

1.3 GCP Quick Start

This Quick Start guide provides simplified instructions for using DKP to get your Kubernetes cluster up and running with minimal configuration requirements on Google Cloud Platform (GCP).

1.3.1 Prerequisites

Before beginning a DKP installation, verify that you have:

- An x86_64-based Linux or macOS machine with a supported version of the operating system.
- The `dkp` binary on this machine.
- Docker[17] version 18.09.2 or later.
- kubectl[18] for interacting with the running cluster.

17 https://docs.docker.com/get-docker/
18 https://kubernetes.io/docs/tasks/tools/#kubectl

- Install the GCP `gcloud` CLI by following the GCP install documentation[19].[20]

1.3.2 GCP prerequisites

- If you are creating the bootstrap cluster on a non-GCP instance or one that does not have the required `editor` role:
  - (option 1) Create a service account using the following `gcloud` commands:

```
export GCP_PROJECT=<your GCP project ID>
export SERVICE_ACCOUNT_USER=<some new service account user>
export GOOGLE_APPLICATION_CREDENTIALS="$HOME/.gcloud/credentials.json"

gcloud iam service-accounts create "$SERVICE_ACCOUNT_USER" --
project=$GCP_PROJECT
gcloud projects add-iam-policy-binding $GCP_PROJECT --
member="serviceAccount:
$SERVICE_ACCOUNT_USER@$GCP_PROJECT.iam.gserviceaccount.com" --role=roles/
editor
gcloud iam service-accounts keys create $GOOGLE_APPLICATION_CREDENTIALS --
iam-account="$SERVICE_ACCOUNT_USER@$GCP_PROJECT.iam.gserviceaccount.com"
```

  - (option 2) Retrieve the credentials for an existing service account using the following `gcloud` commands:

```
export GCP_PROJECT=<your GCP project ID>
export SERVICE_ACCOUNT_USER=<existing service account user>
export GOOGLE_APPLICATION_CREDENTIALS="$HOME/.gcloud/credentials.json"

gcloud iam service-accounts keys create $GOOGLE_APPLICATION_CREDENTIALS --
iam-account="$SERVICE_ACCOUNT_USER@$GCP_PROJECT.iam.gserviceaccount.com"
```

  - Export the static credentials that will be used to create the cluster:

```
export GCP_B64ENCODED_CREDENTIALS=$(base64 < "$
{GOOGLE_APPLICATION_CREDENTIALS}" | tr -d '\n')
```

- Export the GCP location where you want to deploy the cluster, the default is set to `us-west1` :

```
export GCP_REGION=us-west1
```

19 https://cloud.google.com/sdk/docs/install
20 https://cloud.google.com/sdk/docs/install.

1.3.3 Create a new GCP Kubernetes cluster

If you use these instructions to create a cluster on GCP using the DKP default settings without any edits to configuration files or additional flags, your cluster will be deployed with 3 control plane nodes, and 4 worker nodes in the default `us-west1` region.

Follow these steps:

1. Create an image using Konvoy Image Builder (KIB)(see page 328) and export the image name:

   ```
   export IMAGE_NAME=projects/$GCP_PROJECT/global/images/<image-name>
   ```

2. Give your cluster a name suitable for your environment:

   ```
   export CLUSTER_NAME=gcp-example
   ```

3. Create a Kubernetes cluster:
   NOTE: To increase Docker Hub's rate limit[21] use your Docker Hub credentials when creating the cluster, by setting the following flag `--registry-mirror-url=https://registry-1.docker.io --registry-mirror-username= --registry-mirror-password=` on the `dkp create cluster` command.

   ```
   dkp create cluster gcp \
   --cluster-name=${CLUSTER_NAME} \
   --additional-tags=owner=$(whoami) \
   --with-gcp-bootstrap-credentials=true \
   --project=$GCP_PROJECT \
   --image=$IMAGE_NAME \
   --self-managed
   ```

   You should see output similar to this example:

   ```
   Generating cluster resources
   cluster.cluster.x-k8s.io/gcp-example created
   gcpcluster.infrastructure.cluster.x-k8s.io/gcp-example created
   kubeadmcontrolplane.controlplane.cluster.x-k8s.io/gcp-example-control-plane
   created
   gcpmachinetemplate.infrastructure.cluster.x-k8s.io/gcp-example-control-plane
   created
   secret/gcp-example-etcd-encryption-config created
   machinedeployment.cluster.x-k8s.io/gcp-example-md-0 created
   gcpmachinetemplate.infrastructure.cluster.x-k8s.io/gcp-example-md-0 created
   kubeadmconfigtemplate.bootstrap.cluster.x-k8s.io/gcp-example-md-0 created
   clusterresourceset.addons.cluster.x-k8s.io/calico-cni-installation-gcp-example
   created
   ```

21 https://docs.docker.com/docker-hub/download-rate-limit/

```
configmap/calico-cni-installation-gcp-example created
configmap/tigera-operator-gcp-example created
clusterresourceset.addons.cluster.x-k8s.io/gcp-persistent-disk-gcp-example
created
configmap/gcp-persistent-disk-csi-gcp-example created
clusterresourceset.addons.cluster.x-k8s.io/cluster-autoscaler-gcp-example
created
configmap/cluster-autoscaler-gcp-example created
clusterresourceset.addons.cluster.x-k8s.io/node-feature-discovery-gcp-example
created
configmap/node-feature-discovery-gcp-example created
clusterresourceset.addons.cluster.x-k8s.io/nvidia-feature-discovery-gcp-example
created
configmap/nvidia-feature-discovery-gcp-example created
```

As part of the underlying processing, the DKP CLI:

- creates a bootstrap cluster
- creates a workload cluster
- moves CAPI controllers from the bootstrap cluster to the workload cluster, making it self-managed
- deletes the bootstrap cluster

1.3.4 Explore the new Kubernetes cluster

The `kubeconfig` file is written to your local directory and you can now explore the cluster.

Follow these steps:

1. List the Nodes:

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf get nodes
```

You should see output similar to this example:

```
NAME                              STATUS   ROLES                  AGE
VERSION
gcp-example-control-plane-9z77w   Ready    control-plane,master   4m44s
v1.23.12
gcp-example-control-plane-rtj9h   Ready    control-plane,master   104s
v1.23.12
gcp-example-control-plane-zbf9w   Ready    control-plane,master   3m23s
v1.23.12
gcp-example-md-0-88c46            Ready    <none>                 3m28s   v1.23
.12
gcp-example-md-0-fp8s7            Ready    <none>                 3m28s   v1.23
.12
gcp-example-md-0-qvnx7            Ready    <none>                 3m28s   v1.23
.12
gcp-example-md-0-wjdrg            Ready    <none>                 3m27s   v1.23
.12
```

2. List the Pods with the command:

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf get pods -A
```

You should see output similar to this example:

```
NAMESPACE                           NAME
READY   STATUS    RESTARTS       AGE
calico-system                       calico-kube-controllers-577c696df9-v2nzv
1/1     Running   0              5m23s
calico-system                       calico-node-4x5rk
1/1     Running   0              4m22s
calico-system                       calico-node-cxsgc
1/1     Running   0              4m23s
calico-system                       calico-node-dvlnm
1/1     Running   0              4m23s
calico-system                       calico-node-h6nlt
1/1     Running   0              4m23s
calico-system                       calico-node-jmkwq
1/1     Running   0              5m23s
calico-system                       calico-node-tnf54
1/1     Running   0              4m18s
calico-system                       calico-node-v6bwq
1/1     Running   0              2m39s
calico-system                       calico-typha-6d8c94bfdf-dkfvq
1/1     Running   0              5m23s
calico-system                       calico-typha-6d8c94bfdf-fdfn2
1/1     Running   0              3m43s
calico-system                       calico-typha-6d8c94bfdf-kjgzj
1/1     Running   0              3m43s
capa-system                         capa-controller-manager-6468bc488-w7nj9
1/1     Running   0              67s
capg-system                         capg-controller-manager-5fb47f869b-6jgms
1/1     Running   0              53s
capi-kubeadm-bootstrap-system       capi-kubeadm-bootstrap-controller-
manager-65ffc94457-7cjdn      1/1     Running   0              74s
capi-kubeadm-control-plane-system   capi-kubeadm-control-plane-controller-
manager-bc7b688d4-vv8wg   1/1     Running   0              72s
capi-system                         capi-controller-manager-dbfc7b49-dzvw8
1/1     Running   0              77s
cappp-system                        cappp-controller-manager-8444d67568-rmms2
1/1     Running   0              59s
capv-system                         capv-controller-manager-58b8ccf868-rbscn
1/1     Running   0              56s
capz-system                         capz-controller-manager-6467f986d8-dnvj4
1/1     Running   0              62s
cert-manager                        cert-manager-6888d6b69b-7b7m9
1/1     Running   0              91s
cert-manager                        cert-manager-cainjector-76f7798c9-gnp8f
1/1     Running   0              91s
```

```
cert-manager                   cert-manager-webhook-7d4b5d8484-gn5dr
1/1     Running   0            91s
gce-pd-csi-driver              csi-gce-pd-controller-5bd587fbfb-lrx29
5/5     Running   0            5m40s
gce-pd-csi-driver              csi-gce-pd-node-4cgd8
2/2     Running   0            4m22s
gce-pd-csi-driver              csi-gce-pd-node-5qsfk
2/2     Running   0            4m23s
gce-pd-csi-driver              csi-gce-pd-node-5w4bq
2/2     Running   0            4m18s
gce-pd-csi-driver              csi-gce-pd-node-fbdbw
2/2     Running   0            4m23s
gce-pd-csi-driver              csi-gce-pd-node-h82lx
2/2     Running   0            4m23s
gce-pd-csi-driver              csi-gce-pd-node-jzq58
2/2     Running   0            5m39s
gce-pd-csi-driver              csi-gce-pd-node-k6bz9
2/2     Running   0            2m39s
kube-system                    cluster-autoscaler-7f695dc48f-v5kvh
1/1     Running   0            5m40s
kube-system                    coredns-64897985d-hbkqd
1/1     Running   0            5m38s
kube-system                    coredns-64897985d-m8g5j
1/1     Running   0            5m38s
kube-system                    etcd-gcp-example-control-plane-9z77w
1/1     Running   0            5m32s
kube-system                    etcd-gcp-example-control-plane-rtj9h
1/1     Running   0            2m37s
kube-system                    etcd-gcp-example-control-plane-zbf9w
1/1     Running   0            4m17s
kube-system                    kube-apiserver-gcp-example-control-
plane-9z77w                1/1     Running   0            5m32s
kube-system                    kube-apiserver-gcp-example-control-plane-
rtj9h                  1/1     Running   0            2m38s
kube-system                    kube-apiserver-gcp-example-control-plane-
zbf9w                  1/1     Running   0            4m17s
kube-system                    kube-controller-manager-gcp-example-
control-plane-9z77w        1/1     Running   0            5m33s
kube-system                    kube-controller-manager-gcp-example-
control-plane-rtj9h        1/1     Running   0            2m37s
kube-system                    kube-controller-manager-gcp-example-
control-plane-zbf9w        1/1     Running   0            4m17s
kube-system                    kube-proxy-bskz2
1/1     Running   0            4m18s
kube-system                    kube-proxy-gdkn5
1/1     Running   0            4m23s
kube-system                    kube-proxy-knvb9
1/1     Running   0            4m22s
kube-system                    kube-proxy-tcj7r
1/1     Running   0            4m23s
kube-system                    kube-proxy-thdpl
1/1     Running   0            5m38s
```

```
kube-system                         kube-proxy-txxmb
1/1     Running   0             4m23s
kube-system                         kube-proxy-vq6kv
1/1     Running   0             2m39s
kube-system                         kube-scheduler-gcp-example-control-
plane-9z77w                 1/1     Running   0             5m33s
kube-system                         kube-scheduler-gcp-example-control-plane-
rtj9h                   1/1     Running   0             2m37s
kube-system                         kube-scheduler-gcp-example-control-plane-
zbf9w                   1/1     Running   0             4m17s
node-feature-discovery              node-feature-discovery-master-7d5985467-
lh7dc                    1/1     Running   0             5m40s
node-feature-discovery              node-feature-discovery-worker-5qtvg
1/1     Running   0             3m40s
node-feature-discovery              node-feature-discovery-worker-66rwx
1/1     Running   0             3m40s
node-feature-discovery              node-feature-discovery-worker-7h92d
1/1     Running   0             3m35s
node-feature-discovery              node-feature-discovery-worker-b4666
1/1     Running   0             3m40s
tigera-operator                     tigera-operator-5f9bdc5c59-j9tnr
1/1     Running   0             5m38s
```

1.3.5 Install and Log in to the DKP UI

You can now proceed to installing the DKP UI(see page 351) and applications. After installation, you will be able to log in(see page 385) to the DKP UI to explore it.

1.3.6 Delete the Kubernetes cluster and clean up your environment

Delete the provisioned Kubernetes cluster and wait a few minutes:

```
dkp delete cluster \
--cluster-name=${CLUSTER_NAME} \
--with-gcp-bootstrap-credentials=true \
--kubeconfig=${CLUSTER_NAME}.conf \
--self-managed
```

You should see output similar to this example:

```
 ✓ Creating a bootstrap cluster
 ✓ Initializing new CAPI components
 ✓ Moving cluster resources
You can now view resources in the moved cluster by using the --kubeconfig flag with
kubectl. For example: kubectl --kubeconfig=gcp-example-bootstrap.conf get nodes
 ✓ Waiting for cluster infrastructure to be ready
 ✓ Waiting for cluster control-planes to be ready
```

```
 ✓ Waiting for machines to be ready
 ✓ Deleting Services with type LoadBalancer for Cluster default/gcp-example
 ✓ Deleting ClusterResourceSets for Cluster default/gcp-example
 ✓ Deleting cluster resources
 ✓ Waiting for cluster to be fully deleted
Deleted default/gcp-example cluster
 ✓ Deleting bootstrap cluster
```

1.4 On Premises Quick Start

Configure a cluster for On Premises operation

To create a cluster for your on premises infrastructure, follow the procedure that describes creating a cluster on a pre-provisioned infrastructure(see page 222) using SSH access.

Cluster API (CAPI) has a concept of an Infrastructure Provider[22], a set of controllers tasked with managing provider-specific resources for cluster infrastructure and machines via an API. In an on premises environment, there may not be a programmatic API to create and delete infrastructure. Other teams or departments may be responsible for pre-provisioning this infrastructure. In these cases, uses its own pre-provisioned infrastructure provider that relies on having SSH access to the machines and executes the bootstrap steps through SSH.

To use the pre-provisioned infrastructure provider, use Konvoy Image Builder(see page 322) to ensure that the correct versions of kubeadm, kubelet, and containerd are installed with their relevant configurations to enable to run on your infrastructure. Then use the `dkp` CLI to create a DKP cluster on your infrastructure.

Completing this procedure produces a Kubernetes cluster, including a Container Networking Interface (CNI)[23] and a Local Persistence Volume Static Provisioner[24], that is ready for workload deployment.

Before moving to a production environment, you may want to add applications for logging and monitoring, storage, security, and other functions. You can use the Application Manager(see page 387) to select and deploy applications, or deploy your own.

To get started, follow the procedure that describes creating a DKP cluster on a pre-provisioned infrastructure(see page 222).

1.5 vSphere Quick Start

ENTERPRISE

This guide provides instructions for getting started with DKP to get your Kubernetes cluster up and running with basic configuration requirements in a vSphere environment. If you want to customize your vSphere environment, see vSphere Advanced Install(see page 0).

1.5.1 DKP Prerequisites

Before using DKP to create a vSphere cluster, verify that you have:

22 https://cluster-api.sigs.k8s.io/reference/glossary.html#infrastructure-provider
23 https://docs.projectcalico.org/
24 https://github.com/kubernetes-sigs/sig-storage-local-static-provisioner

- An x86_64-based Linux® or macOS® machine.
- The `dkp` binaries and Konvoy Image Builder (KIB)(see page 322) image bundle for Linux or macOS.
- Docker®[25] version 18.09.2 or later installed. You must have Docker installed on the host where the DKP Konvoy CLI runs. For example, if you are installing Konvoy on your laptop, ensure the laptop has a supported version of Docker.

On macOS, Docker runs in a virtual machine. Configure this virtual machine with at least 8GB of memory.

- kubectl[26] 1.21.6 for interacting with the running cluster, installed on the host where the DKP Konvoy command line interface (CLI) runs.
- A valid VMware vSphere account with credentials configured(see page 255).
- Resource requirements(see page 96)

1.5.2 Configure vSphere Prerequisites

Before installing, verify that your VMware vSphere Client environment[27] meets the following basic requirements:

- Access to a bastion VM or other network connected host.
  - You must be able to reach the vSphere API endpoint from where the Konvoy command line interface (CLI) runs.
- vSphere account with credentials configured - this account must have Administrator privileges(see page 255).
- A RedHat® subscription with user name and password for downloading DVD ISOs.
- For air-gapped environments, a bastion VM host template(see page 274) with access to a configu.red Docker registry.
- Valid vSphere values for the following:
  - vCenter API server URL.
  - Datacenter name.
  - vCenter Cluster name that contains ESXi hosts[28] for your cluster's nodes.
  - Datastore name for the shared storage resource to be used for the VMs in the cluster.
    - Use of PersistentVolumes in your cluster depends on Cloud Native Storage (CNS), available in vSphere v6.7.x with Update 3 and later versions. CNS depends on this shared Datastore's configuration.
  - Datastore URL from the datastore record for the shared datastore you want your cluster to use.
    - You need this URL value to ensure that the correct Datastore is used when DKP creates VMs for your cluster in vSphere.
  - Folder name.
  - Base template name, such as base-rhel-8, or base-rhel-7.
  - Name of a Virtual Network that has DHCP enabled for both air-gapped and non air-gapped environments.
  - Resource Pools - at least one resource pool needed, with every host in the pool having access to shared storage, such as VSAN.
    - Each host in the resource pool needs access to shared storage, such as NFS or VSAN, to make use of MachineDeployments and high-availability control planes.

The next step is:

---

25 https://docs.docker.com/get-docker/
26 https://kubernetes.io/docs/tasks/tools/#kubectl
27 https://docs.vmware.com/en/VMware-vSphere/6.7/com.vmware.vsphere.vm_admin.doc/GUID-55238059-912E-411F-A0E9-A7A536972A91.html
28 https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.esxi.install.doc/GUID-B2F01BF5-078A-4C7E-B505-5DFFED0B8C38.html

- For non air-gapped environments, create a base OS image(see page 256)
- For air-gapped environments, create and prepare a bastion VM(see page 274)

1.5.2.1 Create directory for KIB and DKP CLI

This command creates directories for working with images created in KIB as well as a directory for running the commands for DKP:

```
mkdir kib && mkdir dkp
```

1.5.2.2 Get the needed D2iQ Software

Download and decompress KIB by running the following command:

```
cd kib
wget https://github.com/mesosphere/konvoy-image-builder/releases/download/v1.19.14/
konvoy-image-bundle-v1.19.14_linux_amd64.tar.gz
tar -xvf konvoy-image-bundle-v1.19.14_linux_amd64.tar.gz
```

Use this link to Download and decompress DKP(see page 83) and copy or move the dkp binary into the `dkp` subdirectory you created above..

1.5.2.3 Create a folder and resource pool in vCenter for DKP cluster

To create a folder in vCenter, follow the creation steps below:

1. Right click on the datacenter
2. Select New Folder
3. Select host and cluster folder
4. Name folder "D2IQ"

In order to create a Resource Pool, follow the steps below:

1. Right click on the vCenter cluster you plan to use for DKP
2. Select New Resource Pool
3. Adjust values if you need to restrict resources for DKP

1.5.2.4 Build template using KIB

Run the following command replacing `rhel-84.yaml` with `rhel-79.yaml` if necessary to create the compatible image:

```
cd ..
cd kib/images/ova
vi rhel-84.yaml
```

For troubleshooting building an image, see the following solutions guide[29].

1.5.2.5 Adjust the packer file for your vSphere cluster

- DKP recommends to install with the packages provided by the operating system package managers. Use the version that corresponds to the major version of your operating system.
- Passwords should be generated not static and a minimum of 20 characters long.

```
---
download_images: true
build_name: "vsphere-rhel-84"
packer_builder_type: "vsphere"
guestinfo_datasource_slug: "https://raw.githubusercontent.com/vmware/cloud-init-
vmware-guestinfo"
guestinfo_datasource_ref: "v1.4.0"
guestinfo_datasource_script: "{{guestinfo_datasource_slug}}/
{{guestinfo_datasource_ref}}/install.sh"
packer:
  vcenter_server: "10.0.1.52"
  vsphere_username: "administrator@vsphere.local"
  vsphere_password: "Password"
  cluster: "cluster1"
  datacenter: "dc1"
  datastore: "vsanDatastore"
  folder: "d2iq"
  insecure_connection: "true"
  network: "VM Network"
  resource_pool: "D2IQ"
  template: "rhel-boot-8.4"
  vsphere_guest_os_type: "rhel8_64Guest"
  guest_os_type: "rhel8-64"
  # goss params
  distribution: "RHEL"
  distribution_version: "8.4"
```

1.5.2.6 Create overrides for docker credentials

Navigate to the directory that contains your image and then create the override files:

```
cd ..
cd ..
```

To override the docker credentials, run the overrides file command below:

29 https://support.d2iq.com/hc/en-us/articles/7315058530196-Common-issues-encountered-when-building-a-compliant-RHEL-template-to-be-used-with-the-vSphere-provider-in-DKP-2-2-X

```
vi overrides.yaml
image_registries_with_auth:
- host: "registry-1.docker.io"
  username: "<dockerhub-user>"
  password: "<dockerhub-password>
  auth: ""
  identityToken: ""
```

1.5.2.7 Build VM template using KIB

Run the following command, to build your template:

```
./konvoy-image build images/ova/rhel-84.yaml --overrides overrides.yaml
```

1.5.2.8 Create DKP cluster on vSphere

Export your vSphere Environment Variables

Copy the set of “exports” below to a text document in a notepad so that you can modify them and copy/paste into the CLI terminal. It is recommended to save this information for later reference.

```
export VSPHERE_SERVER="<vcenter-server-ip-address>"
export VSPHERE_PASSWORD='<password>'
export VSPHERE_USERNAME="<administrator@vsphere.local>"
export export CLUSTER_NAME=dkp
openssl s_client -connect <vcenter_ip.:443 < /dev/null 2>/dev/null | openssl x509
-fingerprint -noout -in /dev/stdin
```

Build the Bootstrap Cluster

```
cd ..
cd dkp
./dkp create bootstrap --with-aws-bootstrap-credentials=false
```

Create the DKP cluster deployment YAML

If you are not using self-signed certificates, remove the last line of the command `--tls-thumb-print=` before running the command below:

```
dkp create cluster vsphere --cluster-name="dkp" --network="VM Network" --control-
plane-endpoint-host="<vip_for_api" --virtual-ip-interface="eth0" --data-center="<dc1>
" --data-store="vsanDatastore" --folder="${VSPHERE_FOLDER}" --server="<vsphere_server
_ip>" --ssh-public-key-file=/root/.ssh/id_rsa.pub --resource-pool="DKP" --vm-
```

```
template=konvoy-ova-vsphere-rhel-84-1.23.12-1649344885 --tls-thumb-print="tls-
thumbprint" --dry-run -o yaml > ${DKP_CLUSTER_NAME}.yaml
```

1.5.3 Create a new vSphere Kubernetes cluster

Create/deploy a cluster using the command below:

```
kubectl create -f ${DKP_CLUSTER_NAME}.yaml
```

If you wish to watch the cluster build, run the command below:

```
dkp describe cluster -c ${DKP_CLUSTER_NAME}
```

DKP deploys MetalLB on vSphere. This is an advanced step as it will require the IP addresses managed that needed to be managed by the MetalLB and therefore not necessary for Quick Start. However, know that it is an option and see the Configure MetalLB(see page 272) for vSphere documentation if necessary for your configuration.

1.5.3.1 Pivot the Cluster Controllers and Create CAPI Controllers on Cluster

```
./dkp create capi-components --kubeconfig ${DKP_CLUSTER_NAME}.conf
```

1.5.3.2 Once created, move the configuration to the new cluster using the command below:

```
./dkp move --to-kubeconfig ${DKP_CLUSTER_NAME}.conf
```

You now have a Self-Managing Kubernetes Cluster deployed on vSphere.

Adjust the Storage class to only allow PVs on a specfic VMware datastore

```
kubectl delete sc vsphere-raw-block-sc --kubeconfig ${DKP_CLUSTER_NAME}.conf
```

Create a storage class yaml with the URL of the VMware datastore you want to use

```
vi sc.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
```

```
  name: vsphere-raw-block-sc
provisioner: csi.vsphere.vmware.com
allowVolumeExpansion: true
parameters:
  datastoreurl: "ds:///vmfs/volumes/vsan:5238a205736fdb1f-c71f7ec7a0353662/"
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
```

Apply the Storage class YAML to create a new default SC

```
kubectl apply -f sc.yaml --kubeconfig ${DKP_CLUSTER_NAME}.conf
```

1.5.3.3 Kommander Deployment

Deploy Kommander to the DKP Cluster

```
./dkp install kommander --kubeconfig ${DKP_CLUSTER_NAME}.conf
```

If you would like to watch the Helm Releases Deploy, run the following command:

```
watch kubectl get hr -A --kubeconfig ${DKP_CLUSTER_NAME}.conf
```

1.5.4 Explore the new Kubernetes cluster

The kubeconfig file is written to your local directory and you can now explore the cluster.

List the Nodes with the command:

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf get nodes
```

1.5.5 Log in to the DKP UI

You can now log in to the DKP UI(see page 385) to explore.

1.5.6 Delete the Kubernetes Cluster and Cleanup your Environment

Follow these steps:

Delete the provisioned Kubernetes cluster and wait a few minutes:

```
dkp delete cluster \
--cluster-name=${CLUSTER_NAME} \
```

```
--kubeconfig=${CLUSTER_NAME}.conf \
--self-managed
```

1.6 DKP Enterprise Quick Start

This page provides generic instructions for getting started with DKP, to get your Kubernetes cluster up and running with basic configuration requirements. Each Infrastructure Provider will have different steps. If your provider is listed in the Table of Contents(see page 37) with a Quick Start, please begin there instead. For AKS and EKS, please follow the steps below first before beginning a Quick Start for either AKS or EKS.

If your provider or environment is different than AKS or EKS, please see that provider's Quick Start(see page 37) page instead.

1.6.1 Prerequisites

Before starting the DKP installation, verify that you have:

- An x86_64-based Linux or macOS machine with a supported version of the operating system.
- The `dkp` binary for Linux, or macOS.
- Docker[30] version 18.09.2 or later.
- kubectl[31] for interacting with the running cluster.
- A valid infrastructure provider account with credentials configured(see page 103).
- Resource requirements(see page 96)

1.6.2 Create a new Kubernetes cluster

Create/deploy a cluster using the command below:

```
kubectl create -f ${DKP_CLUSTER_NAME}.yaml
```

If you wish to watch the cluster build, run the command below:

```
dkp describe cluster -c ${DKP_CLUSTER_NAME}
```

1.6.2.1 Pivot the Cluster Controllers and Create CAPI Controllers on Cluster

```
./dkp create capi-components --kubeconfig ${DKP_CLUSTER_NAME}.conf
```

30 https://docs.docker.com/get-docker/
31 https://kubernetes.io/docs/tasks/tools/#kubectl

1.6.2.2 Once created, move the configuration to the new cluster using the command below:

```
./dkp move --to-kubeconfig ${DKP_CLUSTER_NAME}.conf
```

You now have a Self-Managing Kubernetes Cluster deployed. Proceed to the Quick Start Guide of the Infrastructure Provider of your choice below:

- AKS Quick Start(see page 61)
- EKS Quick Start(see page 68)

1.6.3 AKS Quick Start

ENTERPRISE

Get started by installing a cluster with the default configuration settings on AKS.

This guide provides instructions for getting started with DKP to get your Kubernetes cluster up and running with basic configuration requirements on an Azure Kubernetes Service (AKS) public cloud instance. If you want to customize your AKS environment, see Install AKS Advanced(see page 103).

1.6.3.1 Prerequisites

Before starting the DKP installation, verify that you have:

- An x86_64-based Linux or macOS machine with a supported version of the operating system.
- A Self-managed Azure cluster(see page 187).
- The `dkp` binary for Linux, or macOS.
- Docker[32] version 18.09.2 or later.
- kubectl[33] for interacting with the running cluster.
- Azure CLI[34].
- A valid Azure account used to sign in with the Azure CLI[35].
- Resource requirements(see page 0)

1.6.3.2 Configure AKS Prerequisites

Follow these steps:

1. Log in to Azure:

   ```
   az login
   ```

32 https://docs.docker.com/get-docker/
33 https://kubernetes.io/docs/tasks/tools/#kubectl
34 https://docs.microsoft.com/en-us/cli/azure/install-azure-cli
35 https://docs.microsoft.com/en-us/cli/azure/authenticate-azure-cli?view=azure-cli-latest

```
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "a1234567-b132-1234-1a11-1234a5678b90",
    "id": "b1234567-abcd-11a1-a0a0-1234a5678b90",
    "isDefault": true,
    "managedByTenants": [],
    "name": "Mesosphere Developer Subscription",
    "state": "Enabled",
    "tenantId": "a1234567-b132-1234-1a11-1234a5678b90",
    "user": {
      "name": "user@azuremesosphere.onmicrosoft.com",
      "type": "user"
    }
  }
]
```

2. Create an Azure Service Principal (SP) by running the following command:
   NOTE: If an SP with the name exists, this command will rotate the password.

```
az ad sp create-for-rbac --role contributor --name "$(whoami)-konvoy" --
scopes=/subscriptions/$(az account show --query id -o tsv)
```

```
{
  "appId": "7654321a-1a23-567b-b789-0987b6543a21",
  "displayName": "azure-cli-2021-03-09-23-17-06",
  "password": "Z79yVstq_E.R0R7RUUck718vEHSuyhAB0C",
  "tenant": "a1234567-b132-1234-1a11-1234a5678b90"
}
```

3. Set the required environment variables:

```
export AZURE_SUBSCRIPTION_ID="<id>"       # b1234567-abcd-11a1-
a0a0-1234a5678b90
export AZURE_TENANT_ID="<tenant>"         # a1234567-b132-1234-1a11-1234a5678b9
0
export AZURE_CLIENT_ID="<appId>"          # 7654321a-1a23-567b-
b789-0987b6543a21
export AZURE_CLIENT_SECRET="<password>"   # Z79yVstq_E.R0R7RUUck718vEHSuyhAB0C
```

4. Base64 encode the same environment variables:

```
export AZURE_SUBSCRIPTION_ID_B64="$(echo -n "${AZURE_SUBSCRIPTION_ID}" | base64
| tr -d '\n')"
export AZURE_TENANT_ID_B64="$(echo -n "${AZURE_TENANT_ID}" | base64 | tr -d
'\n')"
```

```
export AZURE_CLIENT_ID_B64="$(echo -n "${AZURE_CLIENT_ID}" | base64 | tr -d
'\n')"
export AZURE_CLIENT_SECRET_B64="$(echo -n "${AZURE_CLIENT_SECRET}" | base64 |
tr -d '\n')"
```

5. Check to see what version of Kubernetes is available in your region. When deploying with AKS, you need to declare the version of Kubernetes you wish to use by running the following command, substituting `<your-location>` for the Azure region you're deploying to:

```
az aks get-versions -o table --location <your-location>
```

6. Set the version of Kubernetes you've chosen:
NOTE: Using Kubernetes v1.23.x is recommended. The version listed in the command is an example.

```
export KUBERNETES_VERSION=1.23.5
```

Name your cluster

Give your cluster a unique name suitable for your environment. In AKS it is critical that the name is unique as no two clusters in the same AKS account can have the same name.

To increase Docker Hub's rate limit use your Docker Hub credentials when creating the cluster, by setting the following flag: `--registry-mirror-url=https://registry-1.docker.io --registry-mirror-username=<username> --registry-mirror-password=<password>` on `dkp create cluster`.

1. Set the environment variable to be used throughout this documentation:

```
export CLUSTER_NAME=aks-example
```

2. Then, modify `dkp create cluster aks` to include the Kubernetes version:

```
dkp create cluster aks --cluster-name=${CLUSTER_NAME} --additional-tags=owner=$
(whoami) --kubernetes-version=${KUBERNETES_VERSION}
```

Create a new AKS Kubernetes cluster

1. Create a Kubernetes cluster:

```
dkp create cluster aks --cluster-name=${CLUSTER_NAME} --additional-tags=owner=$
(whoami)
```

```
Generating cluster resources
cluster.cluster.x-k8s.io/aks-example created
azuremanagedcontrolplane.infrastructure.cluster.x-k8s.io/aks-example created
azuremanagedcluster.infrastructure.cluster.x-k8s.io/aks-example created
machinepool.cluster.x-k8s.io/aks-example created
azuremanagedmachinepool.infrastructure.cluster.x-k8s.io/cp4n2bm created
machinepool.cluster.x-k8s.io/aks-example-md-0 created
azuremanagedmachinepool.infrastructure.cluster.x-k8s.io/mpn6l25 created
clusterresourceset.addons.cluster.x-k8s.io/cluster-autoscaler-aks-example
created
configmap/cluster-autoscaler-aks-example created
clusterresourceset.addons.cluster.x-k8s.io/node-feature-discovery-aks-example
created
configmap/node-feature-discovery-aks-example created
clusterresourceset.addons.cluster.x-k8s.io/nvidia-feature-discovery-aks-example
created
configmap/nvidia-feature-discovery-aks-example created
```

2. (Optional) Specify an authorized key file to have SSH access to the machines.
   The file must contain exactly one entry, as described in this manual[36].
   You can use the `.pub` file that complements your private ssh key. For example, use the public key that complements your RSA private key:

   ```
   --ssh-public-key-file=${HOME}/.ssh/id_rsa.pub
   ```

   The default username for SSH access is `konvoy`. For example, use your own username:

   ```
   --ssh-username=$(whoami)
   ```

3. Wait for the cluster control-plane to be ready:

   ```
   kubectl wait --for=condition=ControlPlaneReady "clusters/${CLUSTER_NAME}" --
   timeout=20m
   ```

   ```
   cluster.cluster.x-k8s.io/aks-example condition met
   ```

⚠ Pivoting is not supported in AKS.

1.6.3.3 Explore the new Kubernetes Cluster

Follow these steps:

36 https://man7.org/linux/man-pages/man8/sshd.8.html#AUTHORIZED_KEYS_FILE_FORMAT

1. Fetch the kubeconfig file:

```
dkp get kubeconfig -c ${CLUSTER_NAME} > ${CLUSTER_NAME}.conf
```

2. List the Nodes with the command:

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf get nodes
```

```
NAME                                STATUS   ROLES   AGE   VERSION
aks-cp4n2bm-57672902-vmss000000   Ready    agent   28m   v1.22.6
aks-cp4n2bm-57672902-vmss000001   Ready    agent   29m   v1.22.6
aks-cp4n2bm-57672902-vmss000002   Ready    agent   29m   v1.22.6
aks-mpn6l25-57672902-vmss000000   Ready    agent   29m   v1.22.6
aks-mpn6l25-57672902-vmss000001   Ready    agent   29m   v1.22.6
aks-mpn6l25-57672902-vmss000002   Ready    agent   29m   v1.22.6
aks-mpn6l25-57672902-vmss000003   Ready    agent   29m   v1.22.6
```

NOTE: It may take a couple of minutes for the Status to move to `Ready` while `calico-node` pods are being deployed.

3. List the Pods with the command:

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf get pods -A
```

```
NAMESPACE               NAME                                                READY
STATUS     RESTARTS   AGE
calico-system           calico-kube-controllers-78f65cd5dd-5m6t2            1/1
Running    0          28m
calico-system           calico-node-27h8r                                   1/1
Running    0          28m
calico-system           calico-node-cn4zw                                   1/1
Running    0          28m
calico-system           calico-node-fgsqx                                   1/1
Running    0          28m
calico-system           calico-node-htr4f                                   1/1
Running    0          28m
calico-system           calico-node-l7skw                                   1/1
Running    0          28m
calico-system           calico-node-mn67v                                   1/1
Running    0          28m
calico-system           calico-node-z626n                                   1/1
Running    0          28m
calico-system           calico-typha-b6c9c78f4-hcnmd                        1/1
Running    0          28m
```

```
calico-system            calico-typha-b6c9c78f4-pz52w                    1/1
Running    0          28m
calico-system            calico-typha-b6c9c78f4-xknwt                    1/1
Running    0          28m
kube-system              azure-ip-masq-agent-9hxsr                       1/1
Running    0          30m
kube-system              azure-ip-masq-agent-bh5m6                       1/1
Running    0          31m
kube-system              azure-ip-masq-agent-c6s4v                       1/1
Running    0          31m
kube-system              azure-ip-masq-agent-gg77k                       1/1
Running    0          30m
kube-system              azure-ip-masq-agent-k5sl8                       1/1
Running    0          31m
kube-system              azure-ip-masq-agent-mmpsp                       1/1
Running    0          31m
kube-system              azure-ip-masq-agent-z4n24                       1/1
Running    0          31m
kube-system              cloud-node-manager-42shm                        1/1
Running    0          31m
kube-system              cloud-node-manager-b9scr                        1/1
Running    0          30m
kube-system              cloud-node-manager-ccmwl                        1/1
Running    0          31m
kube-system              cloud-node-manager-csrml                        1/1
Running    0          31m
kube-system              cloud-node-manager-gkv6x                        1/1
Running    0          31m
kube-system              cloud-node-manager-ttxz7                        1/1
Running    0          30m
kube-system              cloud-node-manager-twlh8                        1/1
Running    0          31m
kube-system              cluster-autoscaler-68c759fbf6-cnkkp             0/1
Init:0/1   0          29m
kube-system              coredns-845757d86-brpzs                         1/1
Running    0          33m
kube-system              coredns-845757d86-nqmlc                         1/1
Running    0          31m
kube-system              coredns-autoscaler-7d56cd888-8bc28              1/1
Running    0          33m
kube-system              csi-azuredisk-node-4bl85                        3/3
Running    0          30m
kube-system              csi-azuredisk-node-8dw5n                        3/3
Running    0          31m
kube-system              csi-azuredisk-node-bg2kb                        3/3
Running    0          31m
kube-system              csi-azuredisk-node-fr9bm                        3/3
Running    0          31m
kube-system              csi-azuredisk-node-nm4k9                        3/3
Running    0          31m
kube-system              csi-azuredisk-node-twvcv                        3/3
Running    0          31m
```

```
kube-system                 csi-azuredisk-node-wgds6                          3/3
Running    0          30m
kube-system                 csi-azurefile-node-5xv28                          3/3
Running    0          31m
kube-system                 csi-azurefile-node-9nl7n                          3/3
Running    0          31m
kube-system                 csi-azurefile-node-c6mn9                          3/3
Running    0          31m
kube-system                 csi-azurefile-node-q69zr                          3/3
Running    0          31m
kube-system                 csi-azurefile-node-q894n                          3/3
Running    0          31m
kube-system                 csi-azurefile-node-v2rmj                          3/3
Running    0          30m
kube-system                 csi-azurefile-node-wkgck                          3/3
Running    0          30m
kube-system                 kube-proxy-5kd77                                  1/1
Running    0          31m
kube-system                 kube-proxy-96jfn                                  1/1
Running    0          30m
kube-system                 kube-proxy-96pj6                                  1/1
Running    0          30m
kube-system                 kube-proxy-b8vzs                                  1/1
Running    0          31m
kube-system                 kube-proxy-fqnw4                                  1/1
Running    0          31m
kube-system                 kube-proxy-rvpp8                                  1/1
Running    0          31m
kube-system                 kube-proxy-sfqnm                                  1/1
Running    0          31m
kube-system                 metrics-server-6576d9ccf8-kfm5q                   1/1
Running    0          33m
kube-system                 tunnelfront-78777b4fd6-g84wp                      1/1
Running    0          27m
node-feature-discovery   node-feature-discovery-master-84c67dcbb6-vgxfm   1/1
Running    0          29m
node-feature-discovery   node-feature-discovery-worker-2htgg               1/1
Running    0          29m
node-feature-discovery   node-feature-discovery-worker-5cpnt               1/1
Running    0          29m
node-feature-discovery   node-feature-discovery-worker-6cjvb               1/1
Running    0          29m
node-feature-discovery   node-feature-discovery-worker-jdmkj               1/1
Running    0          29m
node-feature-discovery   node-feature-discovery-worker-ms749               1/1
Running    0          29m
node-feature-discovery   node-feature-discovery-worker-p2z55               1/1
Running    0          29m
node-feature-discovery   node-feature-discovery-worker-wnwfx               1/1
Running    0          29m
tigera-operator             tigera-operator-74d785cb58-vbr4d                  1/1
Running    0          33m
```

1.6.3.4 Install and Log in to the DKP UI

You can now proceed to installing the DKP UI(see page 351) and applications. After installation, you will be able to log in(see page 385) to the DKP UI to explore it.

1.6.3.5 Delete the Kubernetes Cluster and Cleanup your Environment

Follow these steps:

1. Delete the provisioned Kubernetes cluster and wait a few minutes:

```
dkp delete cluster --cluster-name=${CLUSTER_NAME}
```

```
✓ Deleting Services with type LoadBalancer for Cluster default/aks-example
✓ Deleting ClusterResourceSets for Cluster default/aks-example
✓ Deleting cluster resources
✓ Waiting for cluster to be fully deleted
Deleted default/aks-example cluster
```

1.6.4 EKS Quick Start

ENTERPRISE

This guide provides instructions for getting started with DKP to get your Kubernetes cluster up and running with basic configuration requirements on an Elastic Kubernetes Service (EKS) public cloud instance. If you want to customize your EKS environment, see EKS Advanced Install(see page 197).

1.6.4.1 DKP Prerequisites

Before you begin using DKP, you must have:

- An x86_64-based Linux or macOS machine.
- The `dkp` binary for Linux, or macOS.
- Docker[37] version 18.09.2 or later installed.
- kubectl[38] for interacting with the running cluster.
- Resource requirements(see page 0)

On macOS, Docker runs in a virtual machine. Configure this virtual machine with at least 8GB of memory.

37 https://docs.docker.com/get-docker/
38 https://kubernetes.io/docs/tasks/tools/install-kubectl/

1.6.4.2 AWS prerequisites

Before you begin using DKP with AWS, you must have:

- A valid AWS account with credentials configured(see page 140).
- Installation of aws-iam-authenticator[39]. This binary is used to access your cluster using kubectl. Amazon EKS uses IAM to provide authentication to your Kubernetes cluster.
- Create an IAM policy configuration(see page 202).
- Export the AWS region where you want to deploy the cluster:

```
export AWS_REGION=us-west-2
```

- Export the AWS profile with the credentials you want to use to create the Kubernetes cluster:

```
export AWS_PROFILE=<profile>
```

See the AWS site for more information about AWS credentials.[40]

EKS for DKP 2.3.x is compatible with Kubernetes 1.22.x because EKS uses its own Kubernetes release cycle.

1.6.4.3 Configure EKS Prerequisites

Follow these steps:

1. Follow the steps in EKS Cluster IAM Policy and Roles Configuration(see page 0).
2. Export the AWS region where you want to deploy the EKS cluster:

```
export AWS_REGION=us-west-2
```

3. Export the AWS Profile with the credentials that you want to use to create the EKS Kubernetes cluster:

```
export AWS_PROFILE=<profile>
```

1.6.4.4 Name your cluster

Give your cluster a unique name suitable for your environment. In EKS it is critical that the name is unique as no two clusters in the same EKS account can have the same name.

Set the environment variable to be used throughout this documentation:

39 https://docs.aws.amazon.com/eks/latest/userguide/install-aws-iam-authenticator.html
40 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-profiles.html

```
export CLUSTER_NAME=eks-example
```

Follow these steps:

1. (Optional) To get a list of names in use in your EKS account, use the `aws` CLI tool. For example:

   ```
   aws ec2 describe-vpcs --filter "Name=tag-key,Values=kubernetes.io/cluster" --
   query "Vpcs[*].Tags[?Key=='kubernetes.io/cluster'].Value | sort(@[*][0])"
   ```

   ```
       "alex-eks-cluster-afe98",
       "sam-aws-cluster-8if9q"
   ```

2. (Optional) If you want to create a cluster name that matches the example above, use this command. This creates a unique name every time you run it, so use the command with forethought.

   ```
   export CLUSTER_NAME=eks-example-$(LC_CTYPE=C tr -dc 'a-z0-9' </dev/urandom |
   fold -w 5 | head -n1)
   echo $CLUSTER_NAME
   ```

   ```
   eks-example-i05l6
   ```

1.6.4.5 Create a new EKS Kubernetes cluster

Follow these steps:

1. Make sure your AWS credentials are up to date. If you are using User Profiles, you must refresh the credentials using the command in Step 1. Otherwise, proceed to Step 2.

   ```
   dkp update bootstrap credentials aws
   ```

2. Create a Kubernetes cluster:

   ```
   dkp create cluster eks --cluster-name=${CLUSTER_NAME} --additional-tags=owner=$
   (whoami)
   ```

   ```
   Generating cluster resources
   cluster.cluster.x-k8s.io/eks-example created
   ```

```
awsmanagedcontrolplane.controlplane.cluster.x-k8s.io/eks-example-control-plane
created
machinedeployment.cluster.x-k8s.io/eks-example-md-0 created
awsmachinetemplate.infrastructure.cluster.x-k8s.io/eks-example-md-0 created
eksconfigtemplate.bootstrap.cluster.x-k8s.io/eks-example-md-0 created
clusterresourceset.addons.cluster.x-k8s.io/calico-cni-installation-eks-example
created
configmap/calico-cni-installation-eks-example created
configmap/tigera-operator-eks-example created
clusterresourceset.addons.cluster.x-k8s.io/cluster-autoscaler-eks-example
created
configmap/cluster-autoscaler-eks-example created
clusterresourceset.addons.cluster.x-k8s.io/node-feature-discovery-eks-example
created
configmap/node-feature-discovery-eks-example created
clusterresourceset.addons.cluster.x-k8s.io/nvidia-feature-discovery-eks-example
created
configmap/nvidia-feature-discovery-eks-example created
```

3. (Optional) Specify an authorized key file to have SSH access to the machines.
   The file must contain exactly one entry, as described in this manual[41].
   You can use the `.pub` file that complements your private ssh key. For example, use the public key that complements your RSA private key:

   ```
   --ssh-public-key-file=${HOME}/.ssh/id_rsa.pub
   ```

   The default username for SSH access is `konvoy`. For example, use your own username:

   ```
   --ssh-username=$(whoami)
   ```

4. Wait for the cluster control-plane to be ready:

   ```
   kubectl wait --for=condition=ControlPlaneReady "clusters/${CLUSTER_NAME}" --
   timeout=20m
   ```

   ```
   cluster.cluster.x-k8s.io/eks-example condition met
   ```

1.6.4.6 Explore the new Kubernetes cluster

Follow these steps:

1. Fetch the kubeconfig file:

41 https://man7.org/linux/man-pages/man8/sshd.8.html#AUTHORIZED_KEYS_FILE_FORMAT

```
dkp get kubeconfig -c ${CLUSTER_NAME} > ${CLUSTER_NAME}.conf
```

2. List the Nodes with the command:

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf get nodes
```

```
NAME                                          STATUS   ROLES    AGE   VERSION
ip-10-0-122-211.us-west-2.compute.internal    Ready    <none>   32s   v1.21.5-
eks-9017834
ip-10-0-127-74.us-west-2.compute.internal     Ready    <none>   42s   v1.21.5-
eks-9017834
ip-10-0-71-155.us-west-2.compute.internal     Ready    <none>   46s   v1.21.5-
eks-9017834
ip-10-0-93-47.us-west-2.compute.internal      Ready    <none>   51s   v1.21.5-
eks-9017834
```

NOTE: It may take a couple of minutes for the Status to move to `Ready` while `calico-node` pods are being deployed.

3. List the Pods with the command:

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf get pods -A
```

```
NAMESPACE              NAME                                       READY
STATUS     RESTARTS   AGE
calico-system          calico-kube-controllers-69845d4df5-sc9vq   1/1
Running    0          44s
calico-system          calico-node-5lppw                          1/1
Running    0          44s
calico-system          calico-node-dwbfj                          1/1
Running    0          44s
calico-system          calico-node-q6tg6                          1/1
Running    0          44s
calico-system          calico-node-rbm7c                          1/1
Running    0          44s
calico-system          calico-typha-68c68c96d-tcrxn               1/1
Running    0          35s
calico-system          calico-typha-68c68c96d-xhrjv               1/1
Running    0          44s
kube-system            aws-node-25bnt                             1/1
Running    0          80s
kube-system            aws-node-dr4b7                             1/1
Running    0          89s
kube-system            aws-node-mmn87                             1/1
Running    0          70s
```

```
kube-system                 aws-node-z6cdb                                    1/1
Running     0          84s
kube-system                 cluster-autoscaler-68c759fbf6-zszxr               0/1
Init:0/1    0          9m50s
kube-system                 coredns-85d5b4454c-n54rq                          1/1
Running     0          12m
kube-system                 coredns-85d5b4454c-xzd9w                          1/1
Running     0          12m
kube-system                 kube-proxy-4bhzp                                  1/1
Running     0          84s
kube-system                 kube-proxy-5hkv9                                  1/1
Running     0          80s
kube-system                 kube-proxy-g82d7                                  1/1
Running     0          70s
kube-system                 kube-proxy-h2jv5                                  1/1
Running     0          89s
node-feature-discovery      node-feature-discovery-master-84c67dcbb6-s6874    1/1
Running     0          9m50s
node-feature-discovery      node-feature-discovery-worker-677hh               1/1
Running     0          69s
node-feature-discovery      node-feature-discovery-worker-fvjwz               1/1
Running     0          49s
node-feature-discovery      node-feature-discovery-worker-xcgvt               1/1
Running     0          64s
node-feature-discovery      node-feature-discovery-worker-zctnz               1/1
Running     0          60s
tigera-operator             tigera-operator-d499f5c8f-b56xn                   1/1
Running     1          9m47s
```

1.6.4.7 Log in to the DKP UI

You can now proceed to installing the DKP UI(see page 351) and applications. After installation, you will be able to log in(see page 385) to the DKP UI to explore it.

1.6.4.8 Delete the Kubernetes cluster and cleanup your environment

1. Delete the provisioned Kubernetes cluster and wait a few minutes:

```
dkp delete cluster --cluster-name=${CLUSTER_NAME}
```

```
✓ Deleting Services with type LoadBalancer for Cluster default/eks-example
✓ Deleting ClusterResourceSets for Cluster default/eks-example
✓ Deleting cluster resources
✓ Waiting for cluster to be fully deleted
Deleted default/eks-example cluster
```

2. Delete the `kind` Kubernetes cluster:

```
dkp delete bootstrap --kubeconfig $HOME/.kube/config
```

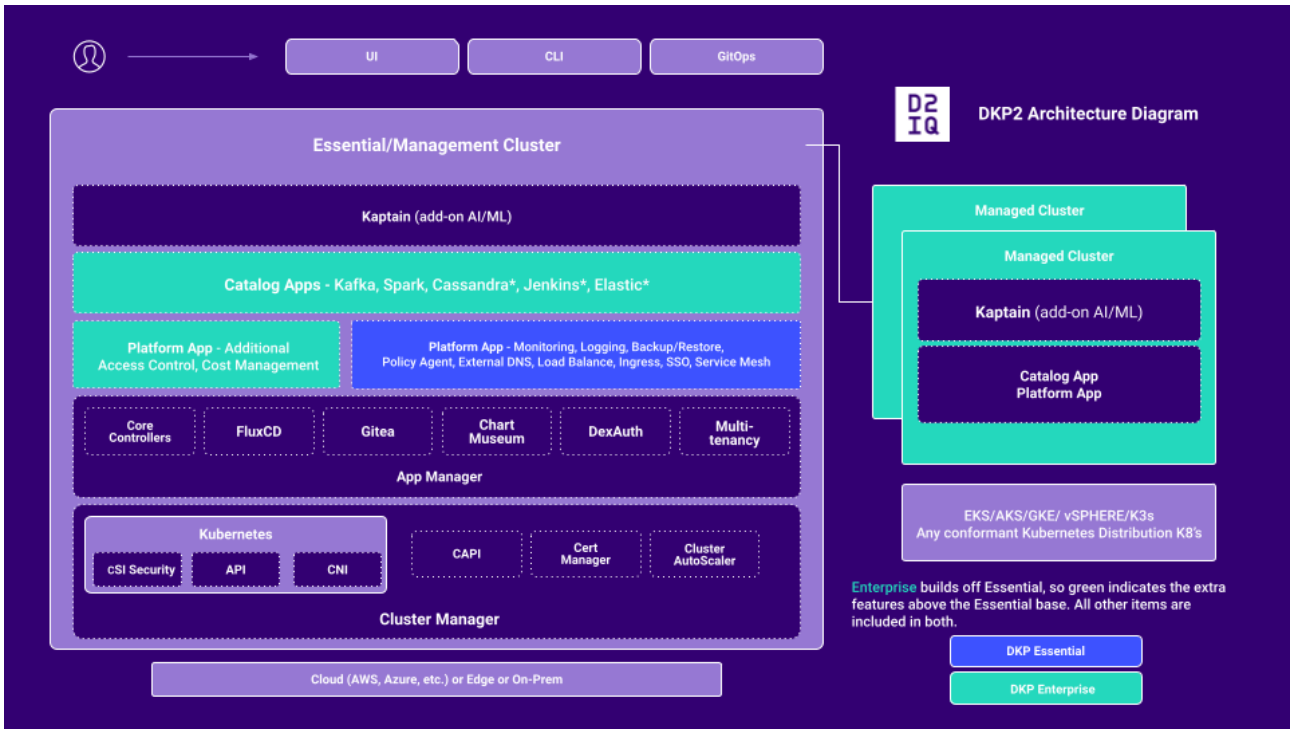
```
✓ Deleting bootstrap cluster
```

2 Architecture

2.1 Learn the key concepts and architectural components of a DKP cluster

Kubernetes provides the foundation of a DKP cluster. Because of this fundamental relationship, you should be familiar with a few key concepts and terms. The topics in this section provide a brief overview of the native Kubernetes architecture, a simplified view of the DKP architecture - both Essential and Enterprise versions. Also, it shows the operational workflow for a DKP cluster.

The following diagram provides a simplified architectural overview to help you visualize the flow of the key components:



DKP Architectural overview

2.2 Components for the Kubernetes control plane

The native Kubernetes cluster consists of **components** that provide the cluster's **control plane** and **worker nodes** that run users' containers and maintain the runtime environment.

DKP supplements the native Kubernetes cluster by providing a predefined and pre-configured set of applications. Because this predefined set of applications provides critical features for managing a Kubernetes cluster in a production environment, the default set is identified as DKP **platform services**.

See Platform applications(see page 387) for the full set of DKP platform services.

2.3 Related information

For information on related topics or procedures, refer to the following Kubernetes[42] documentation.

2.4 DKP Ports

2.4.1 Understand the configured ports for DKP deployment

This section describes ports used by the different Kubernetes components in your DKP cluster.

2.4.1.1 Control-plane nodes

Port	DKP Component	Notes
22	Ansible	ssh
179	calico-node	BGP
1338	Containerd	metrics
2379	etcd	client
2380	etcd	peer
6443	kube-apiserver	
9091	calico-node	felix metrics
9092	calico-node	bird metrics
9099	calico-node	felix liveliness
9100	prometheus node-exporter	metrics
10248	kubelet	health
10249	kube-proxy	metrics

42 https://kubernetes.io/docs/concepts/overview/components/

Port	DKP Component	Notes
10250	kubelet	
10256	kube-proxy	health
10257	kube-controller-manager	secure port
10259	kube-scheduler	secure port
30000-32767	Kubernetes NodePorts	

2.4.1.2 Worker nodes

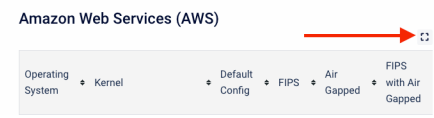
Port	DKP Component	Notes
22	Ansible	ssh
179	calico-node	BGP
1338	Containerd	metrics
5473	calico-typha	syncserver
9091	calico-node	felix metrics
9092	calico-node	bird metrics
9093	calico-typha	metrics
9099	calico-node	felix liveliness
9100	prometheus node-exporter	metrics
9400	NVIDIA GPU DCGM	metrics
10248	kubelet	health
10249	kube-proxy	metrics

Port	DKP Component	Notes
10250	kubelet	
10256	kube-proxy	health
30000-32767	Kubernetes NodePorts	

2.5 Supported Infrastructure Operating Systems

DKP supports the following base Operating Systems.

If the full table is not showing below, select the box icon in the upper-right corner to expand the view:



2.5.1 Amazon Web Services (AWS)

Operating System	Kernel	Default Config	FIPS	Air Gapped	FIPS with Air Gapped	GPU Support	Konvoy Image Builder
CentOS 7.9[43]	3.10.0-1160.el7.x86_64	Yes	Yes	Yes	Yes	Yes	Yes
RHEL 7.9[44]	3.10.0-1160.66.1.el7.x86_64	Yes	Yes	Yes	Yes	Yes	Yes
RHEL 8.2[45]	4.18.0-193.el8.x86_64	Yes	Yes	Yes	Yes	Yes	Yes

43 https://wiki.centos.org/action/show/Manuals/ReleaseNotes/CentOS7.2003
44 https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/7.9_release_notes/index
45 https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/8.2_release_notes/index

Operating System	Kernel	Default Config	FIPS	Air Gapped	FIPS with Air Gapped	GPU Support	Konvoy Image Builder
RHEL 8.4[46]	4.18.0-305.12.1.el8_4.x86_64	Yes	Yes	Yes	Yes	Yes	Yes
Ubuntu 18.04 (Bionic Beaver)[47]	5.4.0-1080-aws	Yes					Yes
Ubuntu 20.04 (Focal Fossa)[48]	5.13.0-1031-aws	Yes				Yes	Yes
SLES 15 SP3[49]	5.3.18-150300.59.63-default	Yes				Yes	Yes
Oracle Linux RHCK 7.9[50]	3.10.0-1160.71.1.0.1.el7.x86_64	Yes	Yes				Yes

2.5.2 Azure

Operating System	Kernel	Default Config	FIPS	Air Gapped	FIPS with Air Gapped	GPU Support	Konvoy Image Builder
Ubuntu 20.04 (Focal Fossa)[51]	5.13.0-1025-azure	Yes					

46 https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/8.4_release_notes/index
47 https://wiki.ubuntu.com/BionicBeaver/ReleaseNotes
48 https://wiki.ubuntu.com/FocalFossa/ReleaseNotes
49 https://documentation.suse.com/en-us/sles/15-SP3/
50 https://docs.oracle.com/en/operating-systems/oracle-linux/7/relnotes7.9/
51 https://wiki.ubuntu.com/FocalFossa/ReleaseNotes

2.5.3 GCP

Operating System	Kernel	Default Config	FIPS	Air Gapped	FIPS with Air Gapped	GPU Support	Konvoy Image Builder
CentOS 7.9[52]	3.10.0-1160.62.1.el7.x86_64	Yes					Yes
Ubuntu 18.04[53]	5.4.0-1072-gcp	Yes					Yes
Ubuntu 20.04[54]	5.13.0-1024-gcp	Yes					Yes

2.5.4 Pre-Provisioned/On Premises

Operating System	Kernel	Default Config	FIPS	Air Gapped	FIPS with Air Gapped	GPU Support	Konvoy Image Builder
CentOS 7.9[55]	3.10.0-1160.el7.x86_64	Yes	Yes	Yes	Yes	Yes	Yes
RHEL 7.9[56]	3.10.0-1160.66.1.el7.x86_64	Yes	Yes	Yes		Yes	Yes
RHEL 8.4[57]	4.18.0-305.12.1.el8_4.x86_64	Yes	Yes	Yes		Yes	Yes
Flatcar 3033.2.4[58]	5.10.84	Yes					Yes

52 https://wiki.centos.org/action/show/Manuals/ReleaseNotes/CentOS7.2003
53 https://wiki.ubuntu.com/BionicBeaver/ReleaseNotes
54 https://wiki.ubuntu.com/FocalFossa/ReleaseNotes
55 https://wiki.centos.org/action/show/Manuals/ReleaseNotes/CentOS7.2003
56 https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/7.9_release_notes/index
57 https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/8.4_release_notes/index
58 https://www.flatcar.org/releases/#stable-release

Operating System	Kernel	Default Config	FIPS	Air Gapped	FIPS with Air Gapped	GPU Support	Konvoy Image Builder
Ubuntu 20.04	5.13.0-1031-aws	Yes				Yes	Yes
SLES 15 SP3[59]	5.3.18-150300.59.63-default	Yes				Yes	Yes
Oracle Linux RHCK 7.9[60]	3.10.0-1160.71.1.0.1.el7.x86_64	Yes					Yes

2.5.5 vSphere

Operating System	Kernel	Default Config	FIPS	Air Gapped	FIPS with Air Gapped	GPU Support	Konvoy Image Builder
RHEL 7.9[61]	3.10.0-1160.el7.x86_64	Yes	Yes	Yes			Yes
RHEL 8.4[62]	4.18.0-305.el8.x86_64	Yes	Yes	Yes			Yes

2.5.6 EKS

Operating System	Kernel	Default Config	FIPS	Air Gapped	FIPS with Air Gapped	GPU Support	Konvoy Image Builder
Amazon Linux 2 v7.9[63]	5.4.204-113.362.amzn2.x86_64	Yes					

59 https://documentation.suse.com/en-us/sles/15-SP3/
60 https://docs.oracle.com/en/operating-systems/oracle-linux/7/relnotes7.9/
61 https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/7.9_release_notes/index
62 https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/8.4_release_notes/index
63 https://docs.aws.amazon.com/AL2/latest/relnotes/relnotes-al2.html

2.5.7 AKS

Operating System	Kernel	Default Config	FIPS	Air Gapped	FIPS with Air Gapped	GPU Support	Konvoy Image Builder
Ubuntu 18.04 (Bionic Beaver)[64]	5.4.0-1085-azure	Yes					

64 https://wiki.ubuntu.com/BionicBeaver/ReleaseNotes

3 Download DKP

To download a new version of DKP, you have 2 options:

- Download from the support website(see page 83)
- Download from the AWS Marketplace(see page 83)

3.1 Download from the support website

Download DKP

3.2 Download from the AWS Marketplace

Follow the instructions on AWS console to download the container image.

After downloading the image, run the following command to copy the binaries onto your host.

```
docker run -it --rm -u $(id -u):$(id -g) -v $(pwd):/dkp $CONTAINER_IMAGES
```

You will then see the following output:

```
dkp binary is placed in the local directory, to run:
./dkp --help
```

You will now see the `dkp` binary in your working directory. Follow the DKP installation instructions(see page 366) using these binaries, and then add your license(see page 91) to DKP. If you have problems downloading or installing DKP, contact your sales representative or sales@d2iq.com[65].

This Docker image includes code from the MinIO Project (“MinIO”), which is © 2015-2021 MinIO, Inc. MinIO is made available subject to the terms and conditions of the GNU Affero General Public License 3.0. The complete source code for the versions of MinIO packaged with DKP/Kommander/Konvoy 2.3.2 are available at these URLs:

3.2.1 https://github.com/minio/minio/tree/RELEASE.2022-01-08T03-11-54Z

3.2.2 https://github.com/minio/minio/tree/RELEASE.2021-02-14T04-01-33Z

65 mailto:sales@d2iq.com

4 Licenses

This section contains overviews of the feature sets for DKP Essential(see page 89) and DKP Enterprise(see page 87) licenses, and how to acquire and use your license.

All DKP Enterprise features are marked with the Enterprise logo:

ENTERPRISE

The type of license you purchase determines how much of DKP you can utilize. In general, DKP Essential helps you maintain your single-cluster environment, a single Management cluster, in other words. DKP Enterprise helps you maintain a multi-cluster environment, with a Management Cluster and one or more Attached or Managed Clusters. The following table shows the differences between the two types of licenses:

Feature	DKP Essential	DKP Enterprise
Applications		
Workspace Platform Applications	✅	✅
Project Platform Applications(see page 481)		✅
Catalog Applications (Kafka, Zookeeper, Spark)		✅
Custom Applications		✅
Kaptain[66]	✅	✅
DKP Insights[67]	✅	✅
Cluster Management		
Single Cluster	✅	✅
Multiple Clusters		✅
Attaching Cluster		✅
Provisioning an additional cluster		✅

66 https://d2iq.atlassian.net/wiki/spaces/DKAP
67 https://d2iq.atlassian.net/wiki/spaces/DINS

Feature	DKP Essential	DKP Enterprise
Upgrades(see page 780)	✅	✅
GitOps		
Continuous Deployment (FluxCD)		✅
UX		
DKP CLI(see page 687)	✅	✅
DKP UI	✅	✅
Workspaces		✅
Projects		✅
Logging		
Workspace Level Logging	✅	✅
Multi-tenant Logging		✅
Monitoring	✅	✅
Backup & Restore(see page 552)	✅	✅
GPU	✅	✅

Feature	DKP Essential	DKP Enterprise
Security		
Authentication using Dex	✅	✅
Policy control using Gatekeeper	✅	✅
Cluster Provisioning		
DKP on AWS(see page 37)	✅	✅
DKP on Azure(see page 41)	✅	✅
DKP on GCP(see page 46)	✅	✅
DKP on vSphere(see page 53)	✅	✅
Pre-provisioned(see page 222)	✅	✅
On-Prem(see page 53)	✅	✅
EKS Provisioning(see page 68)		✅
AKS Provisioning(see page 61)		✅
FIPS Compliance	✅	✅
Konvoy Image Builder(see page 322)	✅	✅
Air-Gapped Deployments	✅	✅

4.1 Purchase a License

Licenses for both DKP Enterprise and DKP Essential are sold in units of cores. To learn more about both, and to obtain a valid license:

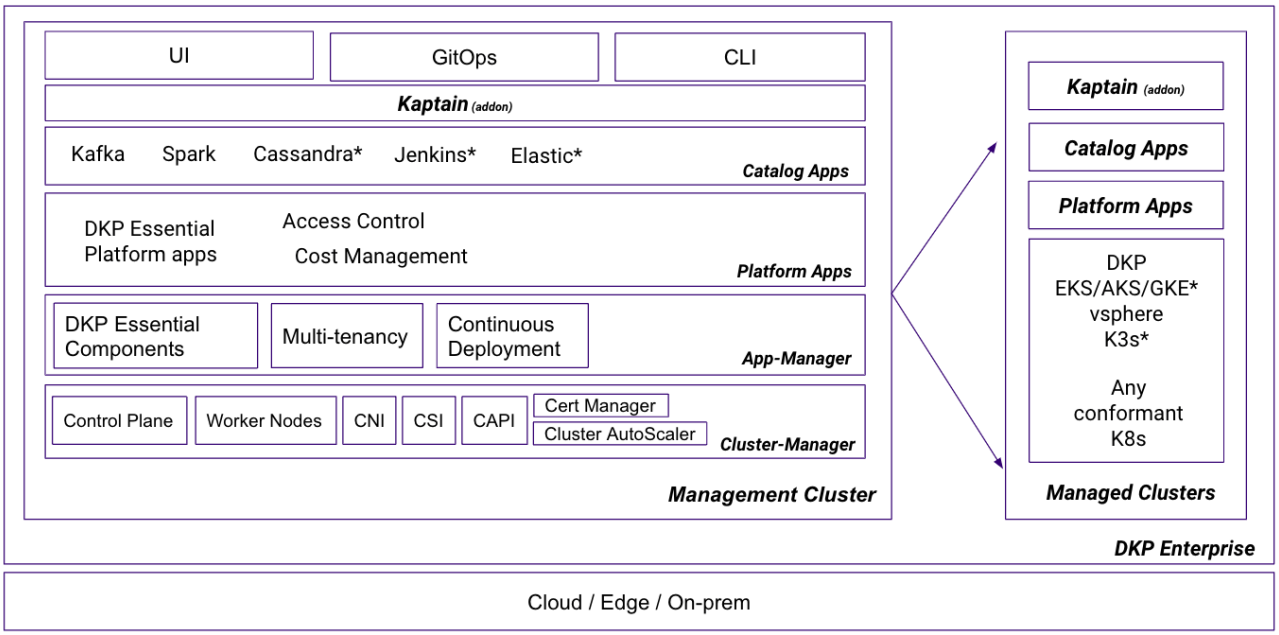
- Contact your sales representative at sales@d2iq.com[68] to purchase one or more licenses.
- Purchase a license via the AWS Marketplace[69]. After purchase, your license information (ARN) is accessible in the AWS License Manager console.

After purchase, download the binary files(see page 83).

4.2 DKP Enterprise

ENTERPRISE

Features and capabilities that are specific to DKP Enterprise are marked with this badge at the top of each page. These items are NOT available to DKP Essential users.



DKP Enterprise is a multi-cluster lifecycle management Kubernetes solution centered around a management cluster that manages multiple attached or managed Kubernetes clusters via a centralized management dashboard. The management dashboard gives you a single point of observability and control throughout all of your attached or managed clusters. The DKP Enterprise license gives the user access to the entire Konvoy cluster environment, the DKP UI dashboard that deploys platform and catalog applications, and multi-cluster management, and comprehensive compatibility with our full range of infrastructure deployment options.

4.2.1 Compatible infrastructure

DKP Enterprise operates across D2iQ's entire range of cloud, on-premises, edge, and air-gapped infrastructures and has support for various OSs, including immutable OSs. See Supported Operating Systems(see page 78) for a full list of compatible infrastructure.

For the basics on standing up a DKP Enterprise cluster in one of the listed environments of your choice, see Advanced Configuration(see page 103).

68 mailto:sales@d2iq.com
69 https://aws.amazon.com/marketplace/

4.2.2 Platform applications

Applications can be deployed in any DKP managed clusters, giving you complete flexibility to operate across cloud, on-premises, edge, and air-gapped scenarios. DKP Enterprise users can use the DKP UI to customize which platform applications to deploy to the cluster in a given workspace.

4.2.3 Catalog Applications

Quickly and easily deploy applications and complex data services from a centralized service catalog to specific or multiple clusters, with governance. Fast data pipelines can be provisioned automatically from the following catalog of DKP Applications:

- **Kafka:** Primarily used to build real-time streaming data pipelines and applications that adapt to the data streams. It combines messaging, storage, and stream processing to allow storage and analysis of both historical and real-time data.
- **Spark:** An industry standard analytics engine for big data processing and machine learning. Spark enables you to process data for both batch and streaming workloads.
- **Zookeeper:** A centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services.

For instructions on how to deploy catalog applications, see Workspace Catalog Applications(see page 441)

4.2.4 Cluster manager

Konvoy is the Kubernetes installer component of DKP Enterprise that uses industry standard tools to produce a certified Kubernetes cluster. These industry standard tools create a cluster management system that includes:

- **Control Plane:** Manages the worker nodes and pods in the cluster.
- **Worker Nodes:** Used to run containerized applications and handle networking to ensure that traffic between applications across the cluster and from outside of the cluster can be properly facilitated.
- **Container Networking Interface (CNI):** Calico's open source networking and network security solution for containers, virtual machines, and native host-based workloads.
- **Container Storage Interface (CSI):** A common abstraction to container orchestrators for interacting with storage subsystems of various types.
- **Kubernetes Cluster API (CAPI):** Cluster API uses Kubernetes-style APIs and patterns to automate cluster lifecycle management for platform operators. For more on how CAPI is integrated into DKP Enterprise, see Understanding CAPI Concepts and Terms(see page 95)
- **Cert Manager:** A Kubernetes addon to automate the management and issuance of TLS certificates from various issuing sources.
- **Cluster Autoscaler:** A component that automatically adjusts the size of a Kubernetes cluster so that all pods have a place to run and there are no unneeded nodes.

4.2.5 Built-in GitOps

DKP Enterprise comes bundled with GitOps, an operating model for Kubernetes and other cloud native technologies, providing a set of best practices that unify Git deployment, management and monitoring for containerized clusters and applications. GitOps works by using Git as a single source of truth for declarative infrastructure and applications. With GitOps, the use of software agents can alert on any divergence between Git with what is running in a cluster, and if there's a difference, Kubernetes reconcilers automatically update or rollback the cluster depending on the case.

4.2.6 DKP Enterprise multi-cluster UI

Bundled with DKP Enterprise is a multi-cluster management view of DKP UI that can be used in lieu of the bundled CLI. From the UI you can:

- **Connect to an infrastructure provider:** DKP Enterprise supports on-premises and cloud infrastructure providers such as AWS and Azure for your Konvoy clusters. To automate their provisioning, DKP requires authentication keys to your preferred infrastructure provider entered on the Add Infrastructure Provider form.
- **Setup identity providers:** DKP Enterprise supports GitHub, LDAP, SAML, and standard OIDC identity providers such as Google. These identity management providers support the login and authentication process for your Kubernetes cluster. See Identity Providers(see page 400) for more information.
- **Configure access control:** Role-based authorization control (RBAC) is central to DKP Enterprise and controls access to resources on all connected clusters. The resources are similar to Kubernetes RBAC. You add an identity provider group, and in that group add cluster roles and cluster role bindings for those roles.
- **Deploy applications:** The DKP Enterprise UI allows you to customize your workspace application deployments via the Applications page within the UI.
- **Create a project:** Create projects within a workspace and deploy project-scoped applications. Projects enable teams to deploy configurations and services to their clusters consistently. After configuring roles, ConfigMaps, secrets, and applications for a project, DKP distributes this configuration to each project namespace. For more information concerning projects, see Projects(see page 459).
- **Add a license:** To add a license via the UI, see Add a License(see page 91)
- **Kubernetes cost monitoring and management**: The kubecost platform application provides real-time cost visibility and insights for external cloud services such as AWS, helping you continuously reduce your cloud costs.

For more information concerning the global and workspace-level UI, see Workspaces(see page 432)

4.3 DKP Essential

DKP Essential is a self-managed single cluster Kubernetes solution that gives you a feature-rich, easy-to-deploy, and easy-to-manage entry-level cloud container platform. The DKP Essential license gives the user access to the entire Konvoy cluster environment, and to the Kommander platform application manager.

Features and capabilities that are specific to DKP Enterprise are marked with this badge at the top of each page. These items are NOT available to DKP Essential users.

4.3.1 Compatible Infrastructure

DKP Essential operates across a range of cloud, on-premise, edge, and air-gapped infrastructures and has support for various OSs, including immutable OSs. See Supported Operating Systems(see page 78) for a full list of compatible infrastructure.

For the basics on standing up a DKP Essential cluster in one of the listed environments of your choice, see Advanced Configuration(see page 103).

4.3.2 Platform Applications

When creating a cluster, the application manager deploys certain platform applications on the newly created cluster. DKP Essential users can use the Kommander UI to customize which platform applications to deploy to the cluster in a given workspace. For a list of available platform applications that are included with DKP Essential, see Workspace Platform Applications(see page 430).

NOTE: The platform application `kubecost` is not included with DKP Essential, but is included with DKP Enterprise(see page 87).

4.3.3 Cluster manager

Konvoy is the Kubernetes installer component of DKP Essential that uses industry standard tools to produce a certified Kubernetes cluster. These industry standard tools create a cluster management system that includes:

- **Control Plane:** Manages the worker nodes and pods in the cluster.
- **Worker Nodes:** Used to run containerized applications and handle networking to ensure that traffic between applications across the cluster and from outside of the cluster can be properly facilitated.
- **Container Networking Interface (CNI):** Calico's open source networking and network security solution for containers, virtual machines, and native host-based workloads.
- **Container Storage Interface (CSI):** A common abstraction to container orchestrators for interacting with storage subsystems of various types.
- **Kubernetes Cluster API (CAPI):** Cluster API uses Kubernetes-style APIs and patterns to automate cluster lifecycle management for platform operators. For more on how CAPI is integrated into DKP Essential, see Understanding CAPI Concepts and Terms(see page 95)
- **Cert Manager:** A Kubernetes addon to automate the management and issuance of TLS certificates from various issuing sources.
- **Cluster Autoscaler:** A component that automatically adjusts the size of a Kubernetes cluster so that all pods have a place to run and there are no unneeded nodes.

4.3.4 Built-in GitOps

DKP Essential comes bundled with GitOps, an operating model for Kubernetes and other cloud native technologies, providing a set of best practices that unify Git deployment, management and monitoring for containerized clusters and applications. GitOps works by using Git as a single source of truth for declarative infrastructure and applications. With GitOps, the use of software agents can alert on any divergence between Git with what is running in a cluster, and if there's a difference, Kubernetes reconcilers automatically update or rollback the cluster depending on the case.

4.3.5 DKP Essential Single Cluster UI

Bundled with DKP Essential is a single cluster management view of DKP UI that can be used in lieu of the bundled CLI. From the UI you can:

- **Setup identity providers:** DKP Essential supports GitHub, LDAP, SAML, and standard OIDC identity providers such as Google. These identity management providers support the login and authentication process for your Kubernetes cluster. See Identity Providers(see page 400) for more information.
- **Deploy applications:** The DKP Essential UI allows you to customize your workspace application deployments via the Applications page within the UI.
- **Add a license:** To add a license via the UI, see Add a license to Kommander(see page 91)

For more information concerning the global and workspace-level UI, see Workspaces(see page 432)

4.4 Add a DKP license

4.4.1 Prerequisites

For licenses that you purchase from the AWS Marketplace[70], an AWS administrator must attach the AWS managed policy[71] `AWSLicenseManagerConsumptionPolicy` to the `control-plane.cluster-api-provider-aws.sigs.k8s.io` role created when configuring AWS IAM policies for Konvoy[72].

Without this policy attached to the role, DKP cannot verify the license information provided in the steps below.

4.4.2 Obtain a License Token or AWS License ARN

For licenses purchased directly from D2iQ, obtain the license token via the customer support portal. Input this token in the last step below.

For licenses purchased via the AWS marketplace, find the license in the “Granted Licenses” table of AWS License Manager inside the AWS console. If necessary, modify the table view preferences to show “License ARN.” Copy this value for use in the last step below.

4.4.3 Enter License Information

⚠ An administrator must add licenses to DKP.

From the DKP UI, complete the following steps:

1. Select **Global** in the workspace header drop-down.
2. In the sidebar menu, select **Administration** > **Licensing**.
3. Select **+ Add License** to enter the Add License form.
4. On the Add License form page, select D2iQ or AWS Marketplace depending on where you acquired your license.
5. Paste your license token or AWS ARN in the text box and select **Save**.

If there is an error submitting a license acquired directly from D2iQ, you can add the license directly through `kubectl`.

4.4.4 Enter a DKP License via kubectl

You can add a license acquired from D2iQ directly using `kubectl`.

1. Create a secret, replacing `MY_LICENSE` in the below command with your D2iQ-provided DKP license:

70 https://aws.amazon.com/marketplace/
71 https://docs.aws.amazon.com/license-manager/latest/userguide/security-iam-awsmanpol.html#security-iam-AWSLicenseManagerConsumptionPolicy
72 https://d2iq.atlassian.net/wiki/spaces/DENT/pages/9897837/Configure+AWS+IAM+Policies#Configure-IAM-Prerequisites-before-starting-a-cluster

```
kubectl create secret generic my-license-secret --from-literal=jwt=MY_LICENSE
-n kommander
kubectl label secret my-license-secret kommanderType=license -n kommander
```

2. Create a license object:

```
cat <<EOF | kubectl apply -f -
apiVersion: kommander.mesosphere.io/v1beta1
kind: License
metadata:
  name: my-license
  namespace: kommander
spec:
  licenseRef:
    name: my-license-secret
EOF
```

Return to the license page in the UI to see your valid license display.

4.5 Remove a DKP license

4.5.1 Remove a license

If your license information has changed, you may need to remove an existing license from DKP to add a new one. Only DKP administrators have the ability to remove licenses.

Original license information can still be obtained from D2iQ or the AWS License Manager console even after removing from DKP.

In the DKP UI, do the following:

1. Select **Global** in the workspace header drop-down.
2. In the sidebar menu, select **Administration** > **Licensing**.
3. Your existing licenses will be listed. Click **Remove License** on the license you would like to remove, and follow the prompts.

4.5.1.1 Manually remove a license using kubectl

To remove a license from DKP using `kubectl`, you have to delete the `Secret` and `License` objects. In this example, the secret is named “my-license-secret”.

1. Validate that the secret exists in the `kommander` namespace:

```
kubectl describe secret -n kommander my-license-secret
```

Expected output:

```
Name:         my-license-secret
Namespace:    kommander
Labels:       kommanderType=license
Annotations:  <none>

Type:         Opaque

Data
====
jwt:  455 bytes
```

2. Delete the secret from the `kommander` namespace:

```
kubectl delete secret -n kommander my-license-secret
```

Expected output:

```
secret "my-license-secret" deleted
```

3. We do the same with the `License` object. Validate that it exists in the `kommander` namespace:

```
kubectl describe license -n kommander my-license
```

Expected output:

```
Name:         my-license
Namespace:    kommander
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"kommander.mesosphere.io/v1beta1","kind":"License
","metadata":{"annotations":{},"name":"my-license", "namespace":"kommand...
API Version:  kommander.mesosphere.io/v1beta1
Kind:         License
Metadata:
  Creation Timestamp:  2020-03-25T14:57:31Z
  Generate Name:       license-
  Generation:          1
  Resource Version:    17895
  Self Link:           /apis/kommander.mesosphere.io/v1beta1/namespaces/
kommander/licenses/my-license
  UID:                 35ee9254-4094-40eb-a2d8-4687c5d212d9
Spec:
  License Ref:
    Name:  my-license-secret
Status:
  Cluster Capacity:  500
  Customer Id:       mesosphere-developer
```

```
  End Date:           2020-10-02T14:00:09Z
  License Id:         mesosphere-developer
  Start Date:         2019-10-02T14:00:09Z
  Valid:              true
  Version:            1.0
Events:
  Type    Reason                Age                   From               Message
  ----    ------                ----                  ----               -------
  Normal  LicenseUpdateSuccess  7m7s (x2 over 7m7s)  LicenseSignature  License
updated successfully
```

4. Delete the license from the `kommander` namespace:

   ```
   kubectl delete license -n kommander my-license
   ```

   Expected output:

   ```
   license.kommander.mesosphere.io "my-license" deleted
   ```

You have now successfully removed a license.

5 Get Started with DKP

When installing DKP for a project, line-of-business, or enterprise, the first step is to determine the infrastructure on which you want to deploy.

For example, you can:

- Install on a public cloud infrastructure, such as Amazon Web Services (AWS) or Azure.
- Install on an internal network, on-premises environment, with a physical or virtual infrastructure.
- Install on an air-gapped edge environment.

⚠ The infrastructure you select then determines the specific requirements for a successful installation. (see Advanced Infrastructure Provider Configurations(see page 103))

DKP is a tool for provisioning Kubernetes clusters with a suite of pre-selected [Cloud Native Computing Foundation CNCF[73] and community-contributed tools. By combining a native Kubernetes cluster as its foundation with a default set of cluster extensions, DKP provides a complete out-of-the-box solution for organizations that want to deploy production-ready Kubernetes.

As an example, this Installation guide provides simplified instructions to get your DKP cluster up and running with minimal configuration requirements on an Amazon Web Services (AWS) public cloud instance. For information about installing on a different platform, see Advanced Infrastructure Provider Configurations(see page 103).

Below is a list of topics to help you get started:

5.1 CAPI Concepts and Terms

To get the most from this major version upgrade, there are a few CAPI concepts to be familiar with, starting with a few important terms and their relationships to each other. You can find a deeper discussion of the architecture in the ClusterAPI Book[74].

CAPI makes use of a bootstrap cluster for provisioning and starting clusters. A bootstrap cluster handles the following:

1. Generating the cluster certificates, if they are not otherwise specified.
2. Initializing the control plane, and managing the creation of other nodes until it is complete.
3. Joining control plane and worker nodes to the cluster.
4. Installing and configuring networking plugin (Calico CNI), CSI volume provisioners, cluster-autoscaler and other core Kubernetes components.

BootstrapData is machine or node role-specific data, such as cloud initialization data, used to bootstrap a “machine” onto a node.

For customers using Kommander for multi-cluster management, a **management cluster** that manages the lifecycle of workload clusters. As the management cluster, DKP Kommander works with bootstrap providers, infrastructure providers, and maintains cluster resources such as bootstrap configurations and templates. If you are working with

---

73 https://www.cncf.io/
74 https://cluster-api.sigs.k8s.io/user/concepts.html

only one cluster, Kommander will provide you with addon (platform application) management for that cluster, but not others.

A **workload cluster** is a Kubernetes cluster whose lifecycle is managed by a management cluster, and provides the platform on which you deploy and execute your workloads.

As part of a collection of **Custom Resource Definitions** or **CRDs** that extend the Kubernetes API, these additional concepts are important for understanding the upgrade.

A **ClusterResourceSet** Kubernetes cluster created by CAPI is functionally minimal in nature. Crucial components like CSI and CNI are not a part of the default cluster spec. A ClusterResourceSet is a CRD that can be used to group and deploy core cluster components post-Kubernetes cluster install.

When you create a bootstrap cluster. You can find all the components in the default namespace and we move them to the workload cluster during the process of making the cluster self-managed.

A **machine** is a declarative spec for a platform or infrastructure component that hosts a Kubernetes node such as a bare metal server or a VM. CAPI uses provider-specific controllers to provision and install new hosts which then register as nodes. When you update a machine spec other than for certain values, such as annotations, status, and labels, the controller deletes the host and creates a new one that conforms to the new spec. This is called machine immutability. If you delete a machine, the controller deletes both the infrastructure and the node. Provider-specific information is not portable between providers.

Within CAPI, you use declarative **MachineDeployments** to handle changes to machines by replacing them in much the same way that a core Kubernetes Deployment replaces Pods. MachineDeployments reconcile changes to machine specs by rolling out changes to two **MachineSets** (similar to a ReplicaSet), both the old and the newly-updated.

A **MachineHealthCheck** identifies unhealthy node conditions, and initiates remediation for nodes owned by a MachineSet.

5.2 Resource Requirements

To ensure a successful DKP install, there are certain requirements to be met. These resource requirements can be slightly different for different Infrastructure Providers.

The general resource requirements are listed after specific providers information.

5.2.1 Infrastructure Provider Specific

For Specific Infrastructure Providers additional requirements apply. An example would be that DKP on Azure defaults to deploying a `Standard_D4s_v3` virtual machine with an 128 GiB volume for the OS and an 80GiB volume for etcd storage, which meets the above requirements. For additional information regarding your Infrastructure Provider, please see the related links below:

Amazon (AWS(see page 114), AWS Air-gapped(see page 155) or EKS(see page 199))

Microsoft Azure (Azure(see page 173) or AKS(see page 103))

Google (GCP(see page 300))

vSphere(see page 253)

Pre-provisioned(see page 223)

5.2.2 General Resource Requirements

In order to Install DKP(see page 97) with the correct amount of resources, please see the list below before beginning installation.

5.2.3 Control plane nodes

You must have at least three control plane nodes. Each control plane node should have at least:

- 4 cores
- 16 GiB memory
- Approximately 80 GiB of free space for the volume used for /var/lib/kubelet and /var/lib/containerd.
- Disk usage must be below 85% on the root volume.

5.2.4 Worker nodes

You must have at least four worker nodes. The specific number of worker nodes required for your environment can vary depending on the cluster workload and size of the nodes. Each worker node should have at least:

- 8 cores
- 32 GiB memory
- Around 80 GiB of free space for the volume used for /var/lib/kubelet and /var/lib/containerd.
- Disk usage must be below 85% on the root volume.

5.3 Install Overview

5.3.1 Before you begin

Before you create a DKP image and deploy the initial DKP cluster, the operator's machine is required to be either an OSX or Linux based machine of a supported version.

For DKP and Konvoy Image Builder to run, the operator machine requires:

- Docker[75] in latest version
- CLI tooling of the cloud provider being used to deploy DKP commands:
  - aws-cli[76]
  - googlecloud-cli[77]
  - azure[78]
- kubectl[79] in latest version
- Konvoy Image Builder(see page 322)

DKP expects permissions on the cloud provider being used. See Advanced Konvoy Configuration(see page 103) for more information on your provider permissions needed. Also see the Supported Operating Systems(see page 78) page for further setup requirements.

---

75 https://www.docker.com/
76 https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html
77 https://cloud.google.com/sdk/docs/install
78 https://learn.microsoft.com/en-us/cli/azure/install-azure-cli
79 https://kubernetes.io/docs/tasks/tools/#kubectl

5.3.1.1 Installation:

Follow the steps below to install the basic DKP package before you can being any custom configuration based on your environment.

1. Install required packages. In most cases, you can install the required software using your preferred package manager. For example, on a macOS computer, you can use Homebrew[80] to install `kubectl` and the `aws` command-line utility by running the following command:

   ```
   brew install kubernetes-cli awscli
   ```

2. Check the Kubernetes client version. Many important Kubernetes functions **do not work** if your client is outdated. You can verify that the version of `kubectl` you have installed is supported by running the following command:

   ```
   kubectl version --short=true
   ```

3. To download DKP, see the Download(see page 83) topic for information. You will need to download and extract the DKP binary package tarball.
4. Verify you have valid **cloud provider security credentials** to deploy the cluster on that platform.
   NOTE: This step regarding provider security credentials is not required if you are installing DKP on an on-premises environment. For information about installing in an on-premises environment, see Install on-premises(see page 222).
5. Deploy with all of the default settings depending on which infrastructure you have.
6. Proceed to the Advanced Konvoy Configuration(see page 103) section of the documentation for further steps on creating a cluster with the Konvoy component of DKP on your infrastructure provider listed in that section.
7. Now that you have a basic DKP cluster installed through Konvoy, you configure the Kommander(see page 351) component.
8. Last, continue with install configuration through sections of documentation under the Kommander(see page 351) component of DKP.

5.3.1.2 Next Step:

Deploy Cluster in Konvoy(see page 99)

Now that you have a basic DKP cluster installed and ready to use, you might want to test operations by deploying a simple, sample application, customizing the cluster configuration, or checking the status of cluster components.

For more details, see the following topics:

- Deploy a sample application(see page 387)
- Platform application deployment(see page 482)
- Troubleshooting(see page 632)

80 https://docs.brew.sh/Installation

5.4 Deploy Cluster in Konvoy

You can use a single command line entry to create a Kubernetes cluster on any of the infrastructures supported by DKP. Within your environment, each cluster that you create with the `dkp create cluster` command requires a globally-unique cluster name that you specify as a flag.

The basic DKP deploy command structure is:

```
dkp create cluster <provider> --cluster-name=clustername --self-managed --flag1=value
--flag2=value ... --flagn=value
```

For a complete list of supported providers, enter the command:

```
dkp create cluster --help
```

The default value for the `--self-managed` flag is false, so you must specify it to enable cluster creation from a single command.

When you execute it, this command:

- Creates a bootstrap cluster, if one is not present
- Deploys CAPI controllers on the bootstrap cluster
- Waits for the cluster to be created, moves the CAPI controllers, and deletes the bootstrap cluster

5.4.1 Infrastructure-specific flags

Additional flags are available to enable needed features on supported cluster providers, and for on-premises and pre-provisioned clusters. You can view additional provider-specific flags and their descriptions with one of the following commands:

```
dkp create cluster <provider> --help
```

5.5 Kommander Installation

5.5.1 Prerequisites

Before you begin using the Kommander UI, you must:

- Install the DKP binary[81] before executing any `dkp` commands. Ensure you have the version of the CLI that matches the DKP version you want to install.
- Configure a Konvoy cluster using the Advanced Konvoy Configuration(see page 103) for instructions infrastructure provider specific instructions on building a cluster based on environment.

81 https://d2iq.atlassian.net/wiki/spaces/DENT/pages/29917289/Download+DKP

- Review the Management Cluster Application Requirements(see page 101) and Workspace Platform Application Defaults and Resource Requirements(see page 418) to ensure that your cluster has sufficient resources.

5.5.1.1 Default StorageClass

To ensure the Git repository that Kommander ships with deploys successfully, the cluster where Kommander is installed must have a default `StorageClass` configured. Run the following command:

```
kubectl get sc
```

The output should look similar to this. Note the `(default)` after the name:

```
NAME               PROVISIONER       RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION   AGE
ebs-sc (default)   ebs.csi.aws.com   Delete          WaitForFirstConsumer   false
41s
```

If the `StorageClass` is not set as default, add the following annotation to the `StorageClass` manifest:

```
annotations:
  storageclass.kubernetes.io/is-default-class: "true"
```

More information on setting a StorageClass as default can be found at Changing the default storage class in k8s docs[82].

5.5.2 Install Kommander on Konvoy

To customize a Kommander installation, see the Kommander Install Configuration(see page 351) for more details.

Before running the commands below, ensure that your `kubectl` configuration **references the cluster on which you want to install Kommander**, otherwise it will install on the bootstrap cluster. You can do this by setting the `KUBECONFIG` environment variable to the appropriate kubeconfig file's location[83].

NOTE: An alternative to initializing the KUBECONFIG environment variable as stated earlier is to use the `--kubeconfig=cluster_name.conf` flag. This ensures that Kommander is installed on the workload cluster.

Begin with this command:

```
dkp install kommander
```

82 https://kubernetes.io/docs/tasks/administer-cluster/change-default-storage-class
83 https://kubernetes.io/docs/tasks/access-application-cluster/configure-access-multiple-clusters/

5.5.3 Verify Installation

Once the Konvoy cluster is built and Kommander has been installed, you will want to verify your installation of Kommander[84] in that section of documentation.

For other environments, see the section of documentation called Advanced Kommander Configuration(see page 351).

5.5.3.1 Next Step

Log in to the UI with Kommander(see page 385)

5.6 Management Cluster Application Requirements

5.6.1 Management cluster application minimum resources and storage requirements

This section only details requirements for management cluster-specific applications. For the list of all platform applications, see Platform Application Configuration Requirements(see page 421).

This table describes the workspace platform applications specific to the management cluster, minimum resource requirements, and minimum persistent storage requirements.

Common Name	Application Name	Minimum Resources Suggested	Minimum Persistent Storage Required
Grafana	centralized-grafana	cpu: 200m memory: 100Mi	
Kubecost	centralized-kubecost	cpu: 1200m memory: 4151Mi	# of PVs: 1 PV sizes: 32Gi
Chartmuseum	chartmuseum		# of PVs: 1 PV sizes: 2Gi
Dex	dex	cpu: 100m memory: 50Mi	
Dex Authenticator	dex-k8s-authenticator	cpu: 100m memory: 128Mi	
Gitea	gitea	cpu: 500m memory: 512Mi	# of PVs: 2 PV sizes: 10Gi

84 https://d2iq.atlassian.net/wiki/spaces/DENT/pages/129663206

Common Name	Application Name	Minimum Resources Suggested	Minimum Persistent Storage Required
Karma	karma		
Flux	kommander-flux	cpu: 5000m memory: 5Gi	
Kubefed	kubefed	cpu: 300m memory: 192Mi	
Thanos	thanos		
Traefik ForwardAuth	traefik-forward-auth-mgmt	cpu: 100m memory: 128Mi	

6 Advanced Konvoy Configuration

When configuring DKP, the first step is to determine your infrastructure

When installing DKP for a project, line-of-business, or enterprise, the first step is to determine the infrastructure on which you want to deploy.

The infrastructure you select then determines the specific requirements for a successful installation. If you have decided to uninstall DKP, refer to the same infrastructure documentation you have selected for specific steps to take down your cluster.

The different types of infrastructures supported in DKP are listed in this section.

Section Contents

6.1 AKS Infrastructure

ENTERPRISE

When installing DKP on Azure Kubernetes Service (**AKS**) infrastructure, you can choose from multiple configuration types. The different types of AKS configuration types supported in DKP are covered in this section.

6.1.1 AKS Prerequisites

ENTERPRISE

Before you begin using DKP with AKS, you must have:

- An x86_64-based Linux or macOS machine.
- A Self-managed Azure Cluster(see page 187)
- kubectl[85] for interacting with the running cluster.
- Azure CLI[86].
- A valid Azure account with credentials configured(see page 173).

85 https://kubernetes.io/docs/tasks/tools/#kubectl
86 https://docs.microsoft.com/en-us/cli/azure/install-azure-cli

NOTE: Ensure that the `KUBECONFIG` environment variable is set to the self-managed cluster by running `export KUBECONFIG={SELF_MANAGED_AZURE_CLUSTER}.conf`

6.1.2 Create a New AKS Cluster

ENTERPRISE

6.1.2.1 **DKP to create a new AKS cluster**

Ensure that the `KUBECONFIG` environment variable is set to the self-managed cluster by running `export KUBECONFIG={SELF_MANAGED_AZURE_CLUSTER}.conf`

6.1.2.2 Create a new AKS Kubernetes cluster

1. Set the environment variable to a name for this cluster.

   ```
   export CLUSTER_NAME=aks-example
   ```

   See Get Started with AKS(see page 61) for information on naming your cluster.
2. Check to see what version of Kubernetes is available in your region. When deploying with AKS, you need to declare the version of Kubernetes you wish to use by running the following command, substituting `<your-location>` for the Azure region you're deploying to:

   ```
   az aks get-versions -o table --location <your-location>
   ```
3. Set the version of Kubernetes you've chosen:
   NOTE: Using Kubernetes v1.23.x is recommended. The version listed in the command is an example.

   ```
   export KUBERNETES_VERSION=1.23.5
   ```
4. Create the cluster:

   ```
   dkp create cluster aks --cluster-name=${CLUSTER_NAME} --additional-tags=owner=$
   (whoami) --kubernetes-version=${KUBERNETES_VERSION}
   ```

   ```
   Generating cluster resources
   cluster.cluster.x-k8s.io/aks-example created
   azuremanagedcontrolplane.infrastructure.cluster.x-k8s.io/aks-example created
   ```

```
azuremanagedcluster.infrastructure.cluster.x-k8s.io/aks-example created
machinepool.cluster.x-k8s.io/aks-example created
azuremanagedmachinepool.infrastructure.cluster.x-k8s.io/cp6dsz8 created
machinepool.cluster.x-k8s.io/aks-example-md-0 created
azuremanagedmachinepool.infrastructure.cluster.x-k8s.io/mp6gglj created
clusterresourceset.addons.cluster.x-k8s.io/cluster-autoscaler-aks-example
created
configmap/cluster-autoscaler-aks-example created
clusterresourceset.addons.cluster.x-k8s.io/node-feature-discovery-aks-example
created
configmap/node-feature-discovery-aks-example created
clusterresourceset.addons.cluster.x-k8s.io/nvidia-feature-discovery-aks-example
created
configmap/nvidia-feature-discovery-aks-example created
```

Inspecting or editing the cluster objects

Use your favorite editor.

NOTE: Editing the cluster objects requires some understanding of Cluster API. Edits can prevent the cluster from deploying successfully.

The objects are Custom Resources[87] defined by Cluster API components, and they belong in three different categories:

- Cluster
  A *Cluster* object has references to the infrastructure-specific and control plane objects.
- Control Plane
- Node Pool
  A Node Pool is a collection of machines with identical properties. For example, a cluster might have one Node Pool with large memory capacity, another Node Pool with GPU support. Each Node Pool is described by three objects: The MachinePool references an object that describes the configuration of Kubernetes components (for example, kubelet) deployed on each node pool machine, and an infrastructure-specific object that describes the properties of all node pool machines. Here, it references a *KubeadmConfigTemplate*.

For in-depth documentation about the objects, read Concepts[88] in the Cluster API Book.

1. Wait for the cluster control-plane to be ready:

   ```
   kubectl wait --for=condition=ControlPlaneReady "clusters/${CLUSTER_NAME}" --
   timeout=20m
   ```

   ```
   cluster.cluster.x-k8s.io/aks-example condition met
   ```

   The `READY` status will become `True` after the cluster control-plane becomes ready.

87 https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/
88 https://cluster-api.sigs.k8s.io/user/concepts.html

2. Once the objects are created on the API server, the Cluster API controllers reconcile them. They create infrastructure and machines. As they progress, they update the Status of each object. DKP provides a command to describe the current status of the cluster:

```
dkp describe cluster -c ${CLUSTER_NAME}
```

```
NAME                                                          READY  SEVERITY
REASON  SINCE  MESSAGE
Cluster/aks-example                                           True
48m
├──ClusterInfrastructure - AzureManagedCluster/aks-example
└──ControlPlane - AzureManagedControlPlane/aks-example
```

3. As they progress, the controllers also create Events. List the Events using this command:

```
kubectl get events | grep ${CLUSTER_NAME}
```

For brevity, the example uses `grep`. It is also possible to use separate commands to get Events for specific objects. For example, `kubectl get events --field-selector involvedObject.kind="AKSCluster"` and `kubectl get events --field-selector involvedObject.kind="AKSMachine"`.

```
48m         Normal    SuccessfulSetNodeRefs               machinepool/aks-
example-md-0                  [{Kind: Namespace: Name:aks-mp6gglj-41174201-
vmss000000 UID:e3c30389-660d-46f5-b9d7-219f80b5674d APIVersion:
ResourceVersion: FieldPath:} {Kind: Namespace: Name:aks-mp6gglj-41174201-
vmss000001 UID:300d71a0-f3a7-4c29-9ff1-1995ffb9cfd3 APIVersion:
ResourceVersion: FieldPath:} {Kind: Namespace: Name:aks-mp6gglj-41174201-
vmss000002 UID:8eae2b39-a415-425d-8417-d915a0b2fa52 APIVersion:
ResourceVersion: FieldPath:} {Kind: Namespace: Name:aks-mp6gglj-41174201-
vmss000003 UID:3e860b88-f1a4-44d1-b674-a54fad599a9d APIVersion:
ResourceVersion: FieldPath:}]
6m4s        Normal    AzureManagedControlPlane available
azuremanagedcontrolplane/aks-example          successfully reconciled
48m         Normal    SuccessfulSetNodeRefs               machinepool/aks-
example                       [{Kind: Namespace: Name:aks-mp6gglj-41174201-
vmss000000 UID:e3c30389-660d-46f5-b9d7-219f80b5674d APIVersion:
ResourceVersion: FieldPath:} {Kind: Namespace: Name:aks-mp6gglj-41174201-
vmss000001 UID:300d71a0-f3a7-4c29-9ff1-1995ffb9cfd3 APIVersion:
ResourceVersion: FieldPath:} {Kind: Namespace: Name:aks-mp6gglj-41174201-
vmss000002 UID:8eae2b39-a415-425d-8417-d915a0b2fa52 APIVersion:
ResourceVersion: FieldPath:}]
```

6.1.2.3 Known Limitations

NOTE: Be aware of these limitations in the current release of DKP.

- The DKP version used to create a workload cluster must match the DKP version used to create a workload cluster.
- DKP supports deploying one workload cluster.
- DKP generates a single nodepool is deployed by default, but you can add additional nodepools.
- DKP does not validate edits to cluster objects.

When complete, you can explore the new cluster(see page 107).

6.1.3 Explore New AKS Cluster

ENTERPRISE

6.1.3.1 Learn to interact with your AKS Kubernetes cluster

This guide explains how to use the command line to interact with your newly deployed Kubernetes cluster.

Before you start, make sure you have created a workload cluster, as described in Create a New Cluster(see page 104).

6.1.3.2 Explore the new AKS cluster

1. Get a kubeconfig file for the workload cluster:
   When the workload cluster is created, the cluster lifecycle services generate a kubeconfig file for the workload cluster, and write it to a *Secret*. The kubeconfig file is scoped to the cluster administrator. Get the kubeconfig from the *Secret*, and write it to a file, using this command:

   ```
   dkp get kubeconfig -c ${CLUSTER_NAME} > ${CLUSTER_NAME}.conf
   ```

2. List the Nodes using this command:

   ```
   kubectl --kubeconfig=${CLUSTER_NAME}.conf get nodes
   ```

   ```
   NAME                                STATUS   ROLES   AGE   VERSION
   aks-cp6dsz8-41174201-vmss000000   Ready    agent   56m   v1.24.6
   aks-cp6dsz8-41174201-vmss000001   Ready    agent   55m   v1.24.6
   aks-cp6dsz8-41174201-vmss000002   Ready    agent   56m   v1.24.6
   aks-mp6gglj-41174201-vmss000000   Ready    agent   55m   v1.24.6
   aks-mp6gglj-41174201-vmss000001   Ready    agent   55m   v1.24.6
   ```

```
aks-mp6gglj-41174201-vmss000002   Ready    agent   55m   v1.24.6
aks-mp6gglj-41174201-vmss000003   Ready    agent   56m   v1.24.6
```

NOTE: It may take a few minutes for the Status to move to `Ready` while the Pod network is deployed. The Nodes' Status should change to Ready soon after the `calico-node` DaemonSet Pods are Ready.

3. List the Pods using this command:

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf get --all-namespaces pods
```

```
NAMESPACE                 NAME                                       READY
STATUS     RESTARTS   AGE
calico-system             calico-kube-controllers-5dcd4b47b5-tgslm   1/1
Running    0          3m58s
calico-system             calico-node-46dj9                          1/1
Running    0          3m58s
calico-system             calico-node-crdgc                          1/1
Running    0          3m58s
calico-system             calico-node-m7s7x                          1/1
Running    0          3m58s
calico-system             calico-node-qfkqc                          1/1
Running    0          3m57s
calico-system             calico-node-sfqfm                          1/1
Running    0          3m57s
calico-system             calico-node-sn67x                          1/1
Running    0          3m53s
calico-system             calico-node-w2pvt                          1/1
Running    0          3m58s
calico-system             calico-typha-6f7f59969c-5z4t5              1/1
Running    0          3m51s
calico-system             calico-typha-6f7f59969c-ddzqb              1/1
Running    0          3m58s
calico-system             calico-typha-6f7f59969c-rr4lj              1/1
Running    0          3m51s
kube-system               azure-ip-masq-agent-4f4v6                  1/1
Running    0          4m11s
kube-system               azure-ip-masq-agent-5xfh2                  1/1
Running    0          4m11s
kube-system               azure-ip-masq-agent-9hlk8                  1/1
Running    0          4m8s
kube-system               azure-ip-masq-agent-9vsgg                  1/1
Running    0          4m16s
kube-system               azure-ip-masq-agent-b9wjj                  1/1
Running    0          3m57s
```

```
kube-system              azure-ip-masq-agent-kpjtl                        1/1
Running    0          3m53s
kube-system              azure-ip-masq-agent-vr7hd                        1/1
Running    0          3m57s
kube-system              cluster-autoscaler-b4789f4bf-qkfk2               0/1
Init:0/1   0          3m28s
kube-system              coredns-845757d86-9jf8b                          1/1
Running    0          5m29s
kube-system              coredns-845757d86-h4xfs                          1/1
Running    0          4m
kube-system              coredns-autoscaler-5f85dc856b-xjb5z              1/1
Running    0          5m23s
kube-system              csi-azuredisk-node-4n4fx                         3/3
Running    0          3m53s
kube-system              csi-azuredisk-node-8pnjj                         3/3
Running    0          3m57s
kube-system              csi-azuredisk-node-sbt6r                         3/3
Running    0          3m57s
kube-system              csi-azuredisk-node-v25wc                         3/3
Running    0          4m16s
kube-system              csi-azuredisk-node-vfbxg                         3/3
Running    0          4m11s
kube-system              csi-azuredisk-node-w5ff5                         3/3
Running    0          4m11s
kube-system              csi-azuredisk-node-zzgqx                         3/3
Running    0          4m8s
kube-system              csi-azurefile-node-2rpcc                         3/3
Running    0          3m57s
kube-system              csi-azurefile-node-4gqkf                         3/3
Running    0          4m11s
kube-system              csi-azurefile-node-f6k8m                         3/3
Running    0          4m16s
kube-system              csi-azurefile-node-k72xq                         3/3
Running    0          4m8s
kube-system              csi-azurefile-node-vx7r4                         3/3
Running    0          3m53s
kube-system              csi-azurefile-node-zc8kr                         3/3
Running    0          4m11s
kube-system              csi-azurefile-node-zkl6b                         3/3
Running    0          3m57s
kube-system              kube-proxy-4fpb6                                 1/1
Running    0          3m53s
kube-system              kube-proxy-6qfbf                                 1/1
Running    0          4m16s
kube-system              kube-proxy-6wnt2                                 1/1
Running    0          4m8s
kube-system              kube-proxy-cspd5                                 1/1
Running    0          3m57s
kube-system              kube-proxy-nsgq6                                 1/1
Running    0          4m11s
kube-system              kube-proxy-qz2st                                 1/1
Running    0          4m11s
```

```
kube-system                kube-proxy-zvh9k                                  1/1
Running    0          3m57s
kube-system                metrics-server-6bc97b47f7-ltkkj                   1/1
Running    0          5m28s
kube-system                tunnelfront-77d68f78bf-t78ck                      1/1
Running    0          5m23s
node-feature-discovery     node-feature-discovery-master-65dc499cd-fxwb5     1/1
Running    0          3m28s
node-feature-discovery     node-feature-discovery-worker-277xc               1/1
Running    0          3m28s
node-feature-discovery     node-feature-discovery-worker-4dq5k               1/1
Running    0          3m28s
node-feature-discovery     node-feature-discovery-worker-57nb8               1/1
Running    0          3m28s
node-feature-discovery     node-feature-discovery-worker-b4lkl               1/1
Running    0          3m28s
node-feature-discovery     node-feature-discovery-worker-kslst               1/1
Running    0          3m28s
node-feature-discovery     node-feature-discovery-worker-ppjtm               1/1
Running    0          3m28s
node-feature-discovery     node-feature-discovery-worker-x5bgf               1/1
Running    0          3m28s
tigera-operator            tigera-operator-74c4d9cf84-k7css                  1/1
Running    0          5m25s
```

When ready, you can delete the cluster(see page 110).

6.1.4 Delete AKS Cluster

ENTERPRISE

Ensure that the `KUBECONFIG` environment variable is set to the self-managed cluster by running `export KUBECONFIG={SELF_MANAGED_AZURE_CLUSTER}.conf`

6.1.4.1 Delete the AKS cluster and clean up your environment

6.1.4.2 Delete the workload cluster

1. Delete the Kubernetes cluster and wait a few minutes:
   Before deleting the cluster, DKP deletes all Services of type LoadBalancer on the cluster. Deleting the Service deletes the Azure LoadBalancer that backs it. To skip this step, use the flag `--delete-kubernetes-resources=false`.

NOTE: Do not skip this step if the Azure Network is managed by DKP. When DKP deletes cluster, it deletes the Network.

```
dkp delete cluster --cluster-name=${CLUSTER_NAME}
```

```
 ✓ Deleting Services with type LoadBalancer for Cluster default/aks-example
 ✓ Deleting ClusterResourceSets for Cluster default/aks-example
 ✓ Deleting cluster resources
 ✓ Waiting for cluster to be fully deleted
Deleted default/aks-example cluster
```

6.1.4.3 Known Limitations

NOTE: Be aware of these limitations in the current release of DKP.

- The DKP version used to create the workload cluster must match the DKP version used to delete the workload cluster.

6.2 AWS Infrastructure

6.2.1 Configuration types

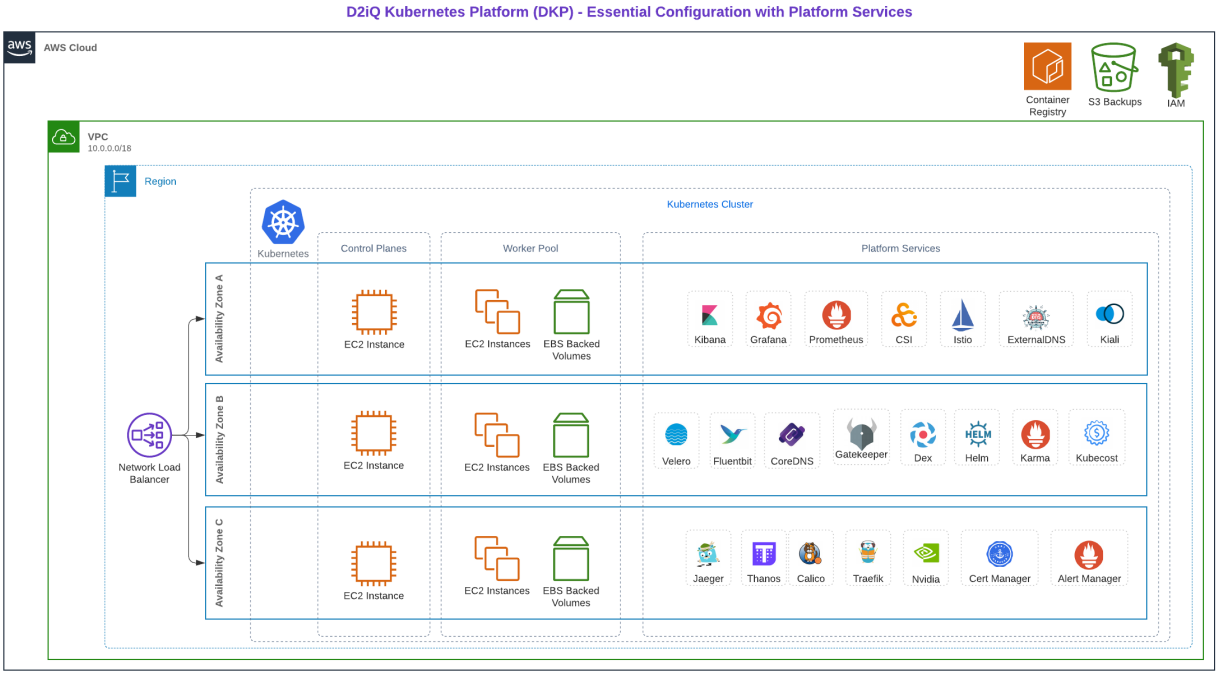
When installing DKP on AWS infrastructure, you can choose from multiple configuration types. The different types of AWS configuration types supported in DKP are listed below.

- Advanced AWS Install(see page 113)
- Minimal Permissions and Role to Create Clusters(see page 140)
- Cluster IAM Policies and Roles(see page 145)
- Multiple AWS Accounts(see page 152)
- Install AWS Air-Gapped(see page 155)
- GPUs in an AWS environment(see page 165)
- Manage AWS Node Pools(see page 166)

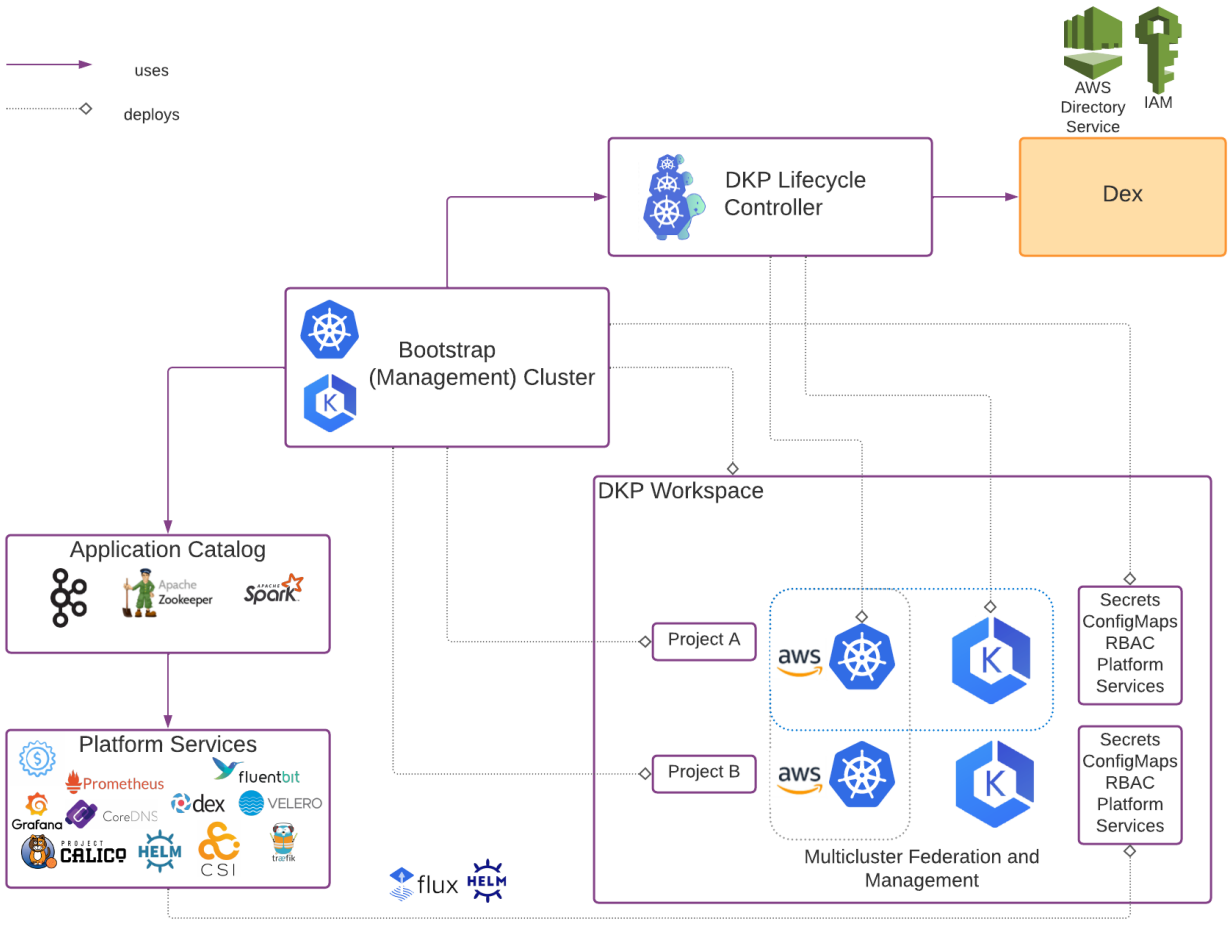
6.2.2 AWS Diagrams

The following diagrams show the two different ways you can implement DKP Essential and DKP Enterprise on AWS.

This diagram shows the granular detail of a single Kubernetes cluster running in AWS Cloud:



This diagram shows a higher-level view of DKP Enterprise, and assumes a multi-cluster environment, where each cluster might look like the single cluster example above:



6.2.3 AWS pricing considerations

Deploying AWS services can incur costs to your organization, depending on how and what you deploy. For more information, see the AWS Pricing Calculator[89].

6.2.4 AWS service limits

When using DKP on AWS, you need to be aware of the possibility of errors due to AWS service limits. For more information, see the AWS Service Limits[90].

6.2.5 Advanced AWS Install

This section provides AWS advanced configuration information to use with DKP.

- AWS Install Prerequisites(see page 114)
- Bootstrap AWS(see page 115)
- Create a New AWS Cluster(see page 117)
- Explore New AWS Cluster(see page 126)

89 https://calculator.aws/#/
90 https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/

6.2.5.1 AWS Install Prerequisites

Konvoy prerequisites

Before you begin using Konvoy, you must have:

- An x86_64-based Linux or macOS machine.
- The `dkp` binary for Linux, or macOS.
- Docker[91] version 18.09.2 or later installed.
- kubectl[92] for interacting with the running cluster.
- A valid AWS account with credentials configured[93].

NOTE: On macOS, Docker runs in a virtual machine. Configure this virtual machine with at least 8GB of memory.

Control plane nodes

You should have at least three control plane nodes. Each control plane node should have at least:

- 4 cores
- 16 GiB memory
- Approximately 80 GiB of free space for the volume used for /var/lib/kubelet and /var/lib/containerd.
- Disk usage must be below 85% on the root volume.

DKP on AWS defaults to deploying an `m5.xlarge` instance with an 80GiB root volume for control plane nodes, which meets the above requirements.

Worker nodes

You should have at least four worker nodes. The specific number of worker nodes required for your environment can vary depending on the cluster workload and size of the nodes. Each worker node should have at least:

- 8 cores
- 32 GiB memory
- Around 80 GiB of free space for the volume used for /var/lib/kubelet and /var/lib/containerd.
- Disk usage must be below 85% on the root volume.

DKP on AWS defaults to deploying a `m5.2xlarge` instance with an 80GiB root volume for worker nodes, which meets the above requirements.

---

91 https://docs.docker.com/get-docker/
92 https://kubernetes.io/docs/tasks/tools/#kubectl
93 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-profiles.html

If you use these instructions to create a cluster on AWS using the DKP default settings without any edits to configuration files or additional flags, your cluster is deployed on an Ubuntu 20.04 operating system image(see page 78) with 3 control plane nodes, and 4 worker nodes which match the requirements above.

NOTE: Using these default images work, but due to missing optimizations, the created cluster will have certain limits. We suggest using Konvoy Image Builder to create a custom AMI(see page 324) to take advantage of enhanced cluster operations.

AWS prerequisites

Before you begin using Konvoy with AWS, you must:

- Create an IAM policy configuration(see page 140).
- Understand IAM Policies and Roles(see page 145).
- Export the AWS region where you want to deploy the cluster:

```
export AWS_REGION=us-west-2
```

- Export the AWS profile with the credentials you want to use to create the Kubernetes cluster:

```
export AWS_PROFILE=<profile>
```

6.2.5.2 Bootstrap AWS

To create Kubernetes clusters, Konvoy uses Cluster API[94] (CAPI) controllers. These controllers run on a Kubernetes cluster. To get started, you need a *bootstrap* cluster. By default, Konvoy creates a bootstrap cluster for you in a Docker container using the Kubernetes-in-Docker (KIND[95]) tool.

Prerequisites

Before you begin, you must:

- Complete the steps in Prerequisites(see page 114).
- Ensure the `dkp` binary can be found in your $PATH.

Bootstrap Cluster Lifecycle Services

1. If an HTTP proxy is required for the bootstrap cluster, set the local `http_proxy`, `https_proxy`, and `no_proxy` environment variables. They are copied into the bootstrap cluster.
2. Create a bootstrap cluster:

94 https://cluster-api.sigs.k8s.io/
95 https://github.com/kubernetes-sigs/kind

```
dkp create bootstrap --kubeconfig $HOME/.kube/config
```

```
✓ Creating a bootstrap cluster
✓ Initializing new CAPI components
```

Konvoy creates a bootstrap cluster using KIND[96] as a library. Konvoy then deploys the following Cluster API[97] providers on the cluster:

- Core Provider[98]
- AWS Infrastructure Provider[99]
- Kubeadm Bootstrap Provider[100]
- Kubeadm ControlPlane Provider[101]

Konvoy waits until the controller-manager and webhook deployments of these providers are ready. List these deployments using this command:

```
kubectl get --all-namespaces deployments -l=clusterctl.cluster.x-k8s.io
```

```
NAMESPACE                           NAME
READY   UP-TO-DATE   AVAILABLE   AGE
capa-system                         capa-controller-manager
1/1     1            1           2m8s
capi-kubeadm-bootstrap-system       capi-kubeadm-bootstrap-controller-manager
1/1     1            1           2m10s
capi-kubeadm-control-plane-system   capi-kubeadm-control-plane-controller-
manager   1/1     1            1           2m10s
capi-system                         capi-controller-manager
1/1     1            1           2m11s
cappp-system                        cappp-controller-manager
1/1     1            1           2m6s
capv-system                         capv-controller-manager
1/1     1            1           2m5s
capz-system                         capz-controller-manager
1/1     1            1           2m7s
cert-manager                        cert-manager
1/1     1            1           2m21s
cert-manager                        cert-manager-cainjector
1/1     1            1           2m21s
cert-manager                        cert-manager-webhook
1/1     1            1           2m21s
```

96 https://github.com/kubernetes-sigs/kind
97 https://cluster-api.sigs.k8s.io/
98 https://github.com/kubernetes-sigs/cluster-api/tree/v0.3.20/
99 https://github.com/kubernetes-sigs/cluster-api-provider-aws
100 https://github.com/kubernetes-sigs/cluster-api/tree/v0.3.20/bootstrap/kubeadm
101 https://github.com/kubernetes-sigs/cluster-api/tree/v0.3.20/controlplane/kubeadm

6.2.5.3 Create a New AWS Cluster

Prerequisites

Before you begin, make sure you have created a Bootstrap(see page 115) cluster.

Name your cluster

Follow these steps:

1. Give your cluster a unique name suitable for your environment.
   In AWS it is critical that the name is unique, as no two clusters in the same AWS account can have the same name.
2. Set the environment variable:

```
export CLUSTER_NAME=aws-example
```

NOTE: The cluster name may only contain the following characters: `a-z`, `0-9`, `.`, and `-`. Cluster creation will fail if the name has capital letters. See Kubernetes[102] for more naming information.

Tips and Tricks

Below are a few ways to customize your setup. If you prefer to do a basic setup, skip Tips and Tricks and proceed to Create a New AWS Cluster(see page 118) section.

1. To get a list of names used in your AWS account, use the `aws` CLI[103]. After downloading, use the following command:

1.
```
aws ec2 describe-vpcs --filter "Name=tag-key,Values=kubernetes.io/cluster" --
query "Vpcs[*].Tags[?Key=='kubernetes.io/cluster'].Value | sort(@[*][0])"
```

```
    "alex-aws-cluster-afe98",
    "sam-aws-cluster-8if9q"
```

2. (Optional) To create a cluster name that is unique, use the following command:

102 https://kubernetes.io/docs/concepts/overview/working-with-objects/names/
103 https://aws.amazon.com/cli/

```
export CLUSTER_NAME=aws-example-$(LC_CTYPE=C tr -dc 'a-z0-9' </dev/urandom |
fold -w 5 | head -n1)
echo $CLUSTER_NAME
```

```
aws-example-pf4a3
```

This will create a unique name every time you run it, so use it with forethought.

3. (Optional) To use a custom AMI when creating your cluster, you must create that AMI using KIB(see page 324) first. Then perform the export and name the custom AMI. Then set the environment variable for the AMI you choose for use in the second command `dkp create cluster` :

```
export AWS_AMI_ID=ami-<ami-id-here>
```

After export, run the following command:

```
dkp create cluster aws --cluster-name=${CLUSTER_NAME} \
--ami=${AWS_AMI_ID} \
--dry-run \
--output=yaml \
> ${CLUSTER_NAME}.yaml
```

Create a new AWS Kubernetes cluster

By default, the control-plane Nodes will be created in 3 different zones. However, the default worker Nodes will reside in a single Availability Zone. You may create additional node pools in other Availability Zones with the `dkp create nodepool` command.

1. Ensure your AWS credentials are up to date. If you are using Static Credentials, use the following command to refresh the credentials. Otherwise, proceed to step 2:

   ```
   dkp update bootstrap credentials aws
   ```

2. Generate the Kubernetes cluster objects:
   NOTE: To increase Dockerhub's rate limit[104] use your Dockerhub credentials when creating the cluster, by setting the following flag `--registry-mirror-url=https://registry-1.docker.io --registry-mirror-username= --registry-mirror-password=` on the `dkp create cluster command` .

104 https://docs.docker.com/docker-hub/download-rate-limit/

```
dkp create cluster aws --cluster-name=${CLUSTER_NAME} \
--dry-run \
--output=yaml \
> ${CLUSTER_NAME}.yaml
```

```
Generating cluster resources
```

3. (Optional) To configure the Control Plane and Worker nodes to use an HTTP proxy:

```
export CONTROL_PLANE_HTTP_PROXY=http://example.org:8080
export CONTROL_PLANE_HTTPS_PROXY=http://example.org:8080
export
CONTROL_PLANE_NO_PROXY=
"example.org,example.com,example.net,localhost,127.0.0.1,10.96.0.0/12,192.168.0
.0/16,kubernetes,kubernetes.default,kubernetes.default.svc,kubernetes.default.s
vc.cluster,kubernetes.default.svc.cluster.local,.svc,.svc.cluster,.svc.cluster.
local,169.254.169.254,.elb.amazonaws.com"

export WORKER_HTTP_PROXY=http://example.org:8080
export WORKER_HTTPS_PROXY=http://example.org:8080
export
WORKER_NO_PROXY=
"example.org,example.com,example.net,localhost,127.0.0.1,10.96.0.0/12,192.168.0
.0/16,kubernetes,kubernetes.default,kubernetes.default.svc,kubernetes.default.s
vc.cluster,kubernetes.default.svc.cluster.local,.svc,.svc.cluster,.svc.cluster.
local,169.254.169.254,.elb.amazonaws.com"
```

   - Replace `example.org,example.com,example.net` with you internal addresses
   - `localhost` and `127.0.0.1` addresses should not use the proxy
   - `10.96.0.0/12` is the default Kubernetes service subnet
   - `192.168.0.0/16` is the default Kubernetes pod subnet
   - `kubernetes,kubernetes.default,kubernetes.default.svc,kubernetes.default.svc.cluster,kubernetes.default.svc.cluster.local` is the internal Kubernetes kube-apiserver service
   - `.svc,.svc.cluster,.svc.cluster.local` is the internal Kubernetes services
   - `169.254.169.254` is the AWS metadata server
   - `.elb.amazonaws.com` is for the worker nodes to allow them to communicate directly to the kube-apiserver ELB
4. (Optional) Create a Kubernetes cluster with HTTP proxy configured. This step assumes you did not already create a cluster in the previous steps:

NOTE: To increase Dockerhub's rate limit[105] use your Dockerhub credentials when creating the cluster, by setting flags `--registry-mirror-url=https://registry-1.docker.io --registry-mirror-username= --registry-mirror-password=` then running `dkp create cluster`

```
dkp create cluster aws --cluster-name=${CLUSTER_NAME} \
--control-plane-http-proxy="${CONTROL_PLANE_HTTP_PROXY}" \
--control-plane-https-proxy="${CONTROL_PLANE_HTTPS_PROXY}" \
--control-plane-no-proxy="${CONTROL_PLANE_NO_PROXY}" \
--worker-http-proxy="${WORKER_HTTP_PROXY}" \
--worker-https-proxy="${WORKER_HTTPS_PROXY}" \
--worker-no-proxy="${WORKER_NO_PROXY}" \
--dry-run \
--output=yaml \
> ${CLUSTER_NAME}.yaml
```

5. Inspect or edit the cluster objects:

   NOTE: Familiarize yourself with Cluster API before editing the cluster objects as edits can prevent the cluster from deploying successfully.
   The objects are Custom Resources[106] defined by Cluster API components, and they belong in three different categories:
   a. Cluster
      A *Cluster* object has references to the infrastructure-specific and control plane objects. Because this is an AWS cluster, there is an *AWSCluster* object that describes the infrastructure-specific cluster properties. Here, this means the AWS region, the VPC ID, subnet IDs, and security group rules required by the Pod network implementation.
   b. Control Plane
      A *KubeadmControlPlane* object describes the control plane, which is the group of machines that run the Kubernetes control plane components, which include the etcd distributed database, the API server, the core controllers, and the scheduler. The object describes the configuration for these components. The object also has a reference to an infrastructure-specific object that describes the properties of all control plane machines. Here, it references an *AWSMachineTemplate* object, which describes the instance type, the type of disk used, and the size of the disk, among other properties.
   c. Node Pool
      A Node Pool is a collection of machines with identical properties. For example, a cluster might have one Node Pool with large memory capacity, another Node Pool with GPU support. Each Node Pool is described by three objects: The MachinePool references an object that describes the configuration of Kubernetes components (for example, kubelet) deployed on each node pool machine, and an infrastructure-specific object that describes the properties of all node pool machines. Here, it references a *KubeadmConfigTemplate*, and an *AWSMachineTemplate* object, which describes the instance type, the type of disk used, the size of the disk, among other properties.
   For in-depth documentation about the objects, read Concepts[107] in the Cluster API Book.
6. Modify Control Plane Audit logs settings using the information contained in the page Configuring the Control Plane(see page 343).
7. Create the cluster from the objects.

105 https://docs.docker.com/docker-hub/download-rate-limit/
106 https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/
107 https://cluster-api.sigs.k8s.io/user/concepts.html

```
kubectl create -f ${CLUSTER_NAME}.yaml
```

```
cluster.cluster.x-k8s.io/aws-example created
awscluster.infrastructure.cluster.x-k8s.io/aws-example created
kubeadmcontrolplane.controlplane.cluster.x-k8s.io/aws-example-control-plane
created
awsmachinetemplate.infrastructure.cluster.x-k8s.io/aws-example-control-plane
created
secret/aws-example-etcd-encryption-config created
machinedeployment.cluster.x-k8s.io/aws-example-md-0 created
awsmachinetemplate.infrastructure.cluster.x-k8s.io/aws-example-md-0 created
kubeadmconfigtemplate.bootstrap.cluster.x-k8s.io/aws-example-md-0 created
clusterresourceset.addons.cluster.x-k8s.io/calico-cni-installation-aws-example
created
configmap/calico-cni-installation-aws-example created
configmap/tigera-operator-aws-example created
clusterresourceset.addons.cluster.x-k8s.io/aws-ebs-csi-aws-example created
configmap/aws-ebs-csi-aws-example created
clusterresourceset.addons.cluster.x-k8s.io/cluster-autoscaler-aws-example
created
configmap/cluster-autoscaler-aws-example created
clusterresourceset.addons.cluster.x-k8s.io/node-feature-discovery-aws-example
created
configmap/node-feature-discovery-aws-example created
clusterresourceset.addons.cluster.x-k8s.io/nvidia-feature-discovery-aws-example
created
configmap/nvidia-feature-discovery-aws-example created
```

8. Wait for the cluster control-plane to be ready:

```
kubectl wait --for=condition=ControlPlaneReady "clusters/${CLUSTER_NAME}" --
timeout=20m
```

```
cluster.cluster.x-k8s.io/aws-example condition met
```

   The `READY` status becomes `True` after the cluster control-plane becomes ready in one of the following steps.
9. After the objects are created on the API server, the Cluster API controllers reconcile them. They create infrastructure and machines. As they progress, they update the Status of each object. Konvoy provides a command to describe the current status of the cluster:

```
dkp describe cluster -c ${CLUSTER_NAME}
```

```
NAME                                                          READY
SEVERITY  REASON  SINCE  MESSAGE
Cluster/aws-example                                          True
60s
├──ClusterInfrastructure - AWSCluster/aws-example             True
5m23s
├──ControlPlane - KubeadmControlPlane/aws-example-control-plane  True
60s
│  ├──Machine/aws-example-control-plane-55jh4                 True
4m59s
│  ├──Machine/aws-example-control-plane-6sn97                 True
2m49s
│  └──Machine/aws-example-control-plane-nx9v5                 True
66s
└──Workers
   └──MachineDeployment/aws-example-md-0                      True
117s
     ├──Machine/aws-example-md-0-cb9c9bbf7-hcl8z              True
3m1s
     ├──Machine/aws-example-md-0-cb9c9bbf7-rtdqw              True
3m2s
     ├──Machine/aws-example-md-0-cb9c9bbf7-t894m              True
3m1s
     └──Machine/aws-example-md-0-cb9c9bbf7-td29r              True
3m1s
```

10. As they progress, the controllers also create Events. List the Events using this command:

```
kubectl get events | grep ${CLUSTER_NAME}
```

For brevity, the example uses `grep`. It is also possible to use separate commands to get Events for specific objects. For example, `kubectl get events --field-selector involvedObject.kind="AWSCluster"` and `kubectl get events --field-selector involvedObject.kind="AWSMachine"`.

```
7m26s       Normal    SuccessfulSetNodeRef                          machine/
aws-example-control-plane-2wb9q      ip-10-0-182-218.us-west-2.compute.internal
11m         Normal    SuccessfulCreate
awsmachine/aws-example-control-plane-vcjkr   Created new control-plane instance
with id "i-0dde024e80ae3de7a"
11m         Normal    SuccessfulAttachControlPlaneELB
awsmachine/aws-example-control-plane-vcjkr   Control plane instance
"i-0dde024e80ae3de7a" is registered with load balancer
```

```
7m25s       Normal    SuccessfulDeleteEncryptedBootstrapDataSecrets
awsmachine/aws-example-control-plane-vcjkr   AWS Secret entries containing
userdata deleted
7m6s        Normal    FailedDescribeInstances
awsmachinepool/aws-example-mp-0              No Auto Scaling Groups with aws-
example-mp-0 found
7m3s        Warning   FailedLaunchTemplateReconcile
awsmachinepool/aws-example-mp-0              Failed to reconcile launch
template: ValidationError: AutoScalingGroup name not found - AutoScalingGroup
aws-example-mp-0 not found
7m1s        Warning   FailedLaunchTemplateReconcile
awsmachinepool/aws-example-mp-0              Failed to reconcile launch
template: ValidationError: AutoScalingGroup name not found - AutoScalingGroup
aws-example-mp-0 not found
6m59s       Warning   FailedLaunchTemplateReconcile
awsmachinepool/aws-example-mp-0              Failed to reconcile launch
template: ValidationError: AutoScalingGroup name not found - AutoScalingGroup
aws-example-mp-0 not found
6m57s       Warning   FailedLaunchTemplateReconcile
awsmachinepool/aws-example-mp-0              Failed to reconcile launch
template: ValidationError: AutoScalingGroup name not found - AutoScalingGroup
aws-example-mp-0 not found
6m55s       Warning   FailedLaunchTemplateReconcile
awsmachinepool/aws-example-mp-0              Failed to reconcile launch
template: ValidationError: AutoScalingGroup name not found - AutoScalingGroup
aws-example-mp-0 not found
6m53s       Warning   FailedLaunchTemplateReconcile
awsmachinepool/aws-example-mp-0              Failed to reconcile launch
template: ValidationError: AutoScalingGroup name not found - AutoScalingGroup
aws-example-mp-0 not found
6m51s       Warning   FailedLaunchTemplateReconcile
awsmachinepool/aws-example-mp-0              Failed to reconcile launch
template: ValidationError: AutoScalingGroup name not found - AutoScalingGroup
aws-example-mp-0 not found
6m49s       Warning   FailedLaunchTemplateReconcile
awsmachinepool/aws-example-mp-0              Failed to reconcile launch
template: ValidationError: AutoScalingGroup name not found - AutoScalingGroup
aws-example-mp-0 not found
6m47s       Warning   FailedLaunchTemplateReconcile
awsmachinepool/aws-example-mp-0              Failed to reconcile launch
template: ValidationError: AutoScalingGroup name not found - AutoScalingGroup
aws-example-mp-0 not found
74s         Warning   FailedLaunchTemplateReconcile
awsmachinepool/aws-example-mp-0              (combined from similar events):
Failed to reconcile launch template: ValidationError: AutoScalingGroup name not
found - AutoScalingGroup aws-example-mp-0 not found
16m         Normal    SuccessfulCreateVPC
awscluster/aws-example                       Created new managed VPC
"vpc-032fff0fe06a85035"
16m         Normal    SuccessfulSetVPCAttributes
awscluster/aws-example                       Set managed VPC attributes for
"vpc-032fff0fe06a85035"
```

```
16m         Normal    SuccessfulCreateSubnet
awscluster/aws-example                      Created new managed Subnet
"subnet-0677a4fbd7d170dfe"
16m         Normal    SuccessfulModifySubnetAttributes
awscluster/aws-example                      Modified managed Subnet
"subnet-0677a4fbd7d170dfe" attributes
16m         Normal    SuccessfulCreateSubnet
awscluster/aws-example                      Created new managed Subnet
"subnet-04fc9deb4fa9f8333"
16m         Normal    SuccessfulCreateSubnet
awscluster/aws-example                      Created new managed Subnet
"subnet-0a266c15dd211ce6c"
16m         Normal    SuccessfulModifySubnetAttributes
awscluster/aws-example                      Modified managed Subnet
"subnet-0a266c15dd211ce6c" attributes
16m         Normal    SuccessfulCreateSubnet
awscluster/aws-example                      Created new managed Subnet
"subnet-06269d5b52d50840f"
16m         Normal    SuccessfulCreateSubnet
awscluster/aws-example                      Created new managed Subnet
"subnet-0fc41ffef7dceface"
16m         Normal    SuccessfulModifySubnetAttributes
awscluster/aws-example                      Modified managed Subnet
"subnet-0fc41ffef7dceface" attributes
16m         Normal    SuccessfulCreateSubnet
awscluster/aws-example                      Created new managed Subnet
"subnet-0725068cca16ad9f9"
16m         Normal    SuccessfulCreateInternetGateway
awscluster/aws-example                      Created new managed Internet
Gateway "igw-07cd7ad3e6c7c1ca7"
16m         Normal    SuccessfulAttachInternetGateway
awscluster/aws-example                      Internet Gateway
"igw-07cd7ad3e6c7c1ca7" attached to VPC "vpc-032fff0fe06a85035"
16m         Normal    SuccessfulCreateNATGateway
awscluster/aws-example                      Created new NAT Gateway
"nat-0a0cf17d29150cf9a"
16m         Normal    SuccessfulCreateNATGateway
awscluster/aws-example                      Created new NAT Gateway
"nat-065e5e383e6f23320"
16m         Normal    SuccessfulCreateNATGateway
awscluster/aws-example                      Created new NAT Gateway
"nat-01c4a6fed2a31ed4c"
13m         Normal    SuccessfulCreateRouteTable
awscluster/aws-example                      Created managed RouteTable
"rtb-09f4e2eecb7462d22"
13m         Normal    SuccessfulCreateRoute
awscluster/aws-example                      Created route {
13m         Normal    SuccessfulAssociateRouteTable
awscluster/aws-example                      Associated managed RouteTable
"rtb-09f4e2eecb7462d22" with subnet "subnet-0677a4fbd7d170dfe"
```

```
13m         Normal    SuccessfulCreateRouteTable
awscluster/aws-example                      Created managed RouteTable
"rtb-0007b98b36f37d1e4"
13m         Normal    SuccessfulCreateRoute
awscluster/aws-example                      Created route {
13m         Normal    SuccessfulAssociateRouteTable
awscluster/aws-example                      Associated managed RouteTable
"rtb-0007b98b36f37d1e4" with subnet "subnet-04fc9deb4fa9f8333"
13m         Normal    SuccessfulCreateRouteTable
awscluster/aws-example                      Created managed RouteTable
"rtb-079a1d7d3667c2525"
13m         Normal    SuccessfulCreateRoute
awscluster/aws-example                      Created route {
13m         Normal    SuccessfulAssociateRouteTable
awscluster/aws-example                      Associated managed RouteTable
"rtb-079a1d7d3667c2525" with subnet "subnet-0a266c15dd211ce6c"
13m         Normal    SuccessfulCreateRouteTable
awscluster/aws-example                      Created managed RouteTable
"rtb-0e5ebc8ec29848a17"
13m         Normal    SuccessfulCreateRoute
awscluster/aws-example                      Created route {
13m         Normal    SuccessfulAssociateRouteTable
awscluster/aws-example                      Associated managed RouteTable
"rtb-0e5ebc8ec29848a17" with subnet "subnet-06269d5b52d50840f"
13m         Normal    SuccessfulCreateRouteTable
awscluster/aws-example                      Created managed RouteTable
"rtb-087f0c400675c4bce"
13m         Normal    SuccessfulCreateRoute
awscluster/aws-example                      Created route {
13m         Normal    SuccessfulAssociateRouteTable
awscluster/aws-example                      Associated managed RouteTable
"rtb-087f0c400675c4bce" with subnet "subnet-0fc41ffef7dceface"
13m         Normal    SuccessfulCreateRouteTable
awscluster/aws-example                      Created managed RouteTable
"rtb-05a05080bbb3cead9"
13m         Normal    SuccessfulCreateRoute
awscluster/aws-example                      Created route {
13m         Normal    SuccessfulAssociateRouteTable
awscluster/aws-example                      Associated managed RouteTable
"rtb-05a05080bbb3cead9" with subnet "subnet-0725068cca16ad9f9"
13m         Normal    SuccessfulCreateSecurityGroup
awscluster/aws-example                      Created managed SecurityGroup
"sg-0379bf77211472854" for Role "bastion"
13m         Normal    SuccessfulCreateSecurityGroup
awscluster/aws-example                      Created managed SecurityGroup
"sg-0a4e0635f68a2f57d" for Role "apiserver-lb"
13m         Normal    SuccessfulCreateSecurityGroup
awscluster/aws-example                      Created managed SecurityGroup
"sg-022da9dfc21ef3d5e" for Role "lb"
13m         Normal    SuccessfulCreateSecurityGroup
awscluster/aws-example                      Created managed SecurityGroup
"sg-00db2e847c0b49d6e" for Role "controlplane"
```

```
13m         Normal    SuccessfulCreateSecurityGroup
awscluster/aws-example                      Created managed SecurityGroup
"sg-01fe3426404f94708" for Role "node"
13m         Normal    SuccessfulAuthorizeSecurityGroupIngressRules
awscluster/aws-example                      Authorized security group ingress
rules [protocol=tcp/range=[22-22]/description=SSH] for SecurityGroup
"sg-0379bf77211472854"
13m         Normal    SuccessfulAuthorizeSecurityGroupIngressRules
awscluster/aws-example                      Authorized security group ingress
rules [protocol=tcp/range=[6443-6443]/description=Kubernetes API] for
 SecurityGroup "sg-0a4e0635f68a2f57d"
13m         Normal    SuccessfulAuthorizeSecurityGroupIngressRules
awscluster/aws-example                      Authorized security group ingress
rules [protocol=tcp/range=[5473-5473]/description=typha (calico) protocol=tcp/
range=[179-179]/description=bgp (calico) protocol=4/range=[-1-65535]/
description=IP-in-IP (calico) protocol=tcp/range=[22-22]/description=SSH
protocol=tcp/range=[6443-6443]/description=Kubernetes API protocol=tcp/range=[2
379-2379]/description=etcd protocol=tcp/range=[2380-2380]/description=etcd
peer] for SecurityGroup "sg-00db2e847c0b49d6e"
13m         Normal    SuccessfulAuthorizeSecurityGroupIngressRules
awscluster/aws-example                      Authorized security group ingress
rules [protocol=tcp/range=[5473-5473]/description=typha (calico) protocol=tcp/
range=[179-179]/description=bgp (calico) protocol=4/range=[-1-65535]/
description=IP-in-IP (calico) protocol=tcp/range=[22-22]/description=SSH
protocol=tcp/range=[30000-32767]/description=Node Port Services protocol=tcp/
range=[10250-10250]/description=Kubelet API] for SecurityGroup
"sg-01fe3426404f94708"
```

Known Limitations

NOTE: Be aware of these limitations in the current release of Konvoy.

- The Konvoy version used to create a bootstrap cluster must match the Konvoy version used to create a workload cluster.
- Konvoy supports deploying one workload cluster.
- Konvoy generates a set of objects for one Node Pool.
- Konvoy does not validate edits to cluster objects.

6.2.5.4 Explore New AWS Cluster

This guide explains how to use the command line to interact with your newly deployed Kubernetes cluster.

Before you start, make sure you have created a workload cluster, as described in Create a New Cluster(see page 126).

Explore the new Kubernetes cluster

1. Get a kubeconfig file for the workload cluster:
   When the workload cluster is created, the cluster lifecycle services generate a kubeconfig file for the workload cluster, and write it to a *Secret*. The kubeconfig file is scoped to the cluster administrator.

Get the kubeconfig from the *Secret*, and write it to a file, using this command:

```
dkp get kubeconfig -c ${CLUSTER_NAME} > ${CLUSTER_NAME}.conf
```

2. List the Nodes using this command:

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf get nodes
```

```
NAME                                          STATUS   ROLES
AGE     VERSION
ip-10-0-100-85.us-west-2.compute.internal     Ready    <none>
8m13s   v1.24.6
ip-10-0-106-183.us-west-2.compute.internal    Ready    control-plane,master
6m22s   v1.24.6
ip-10-0-158-104.us-west-2.compute.internal    Ready    control-plane,master
8m58s   v1.24.6
ip-10-0-203-138.us-west-2.compute.internal    Ready    control-plane,master
7m52s   v1.24.6
ip-10-0-70-169.us-west-2.compute.internal     Ready    <none>
8m14s   v1.24.6
ip-10-0-77-176.us-west-2.compute.internal     Ready    <none>
8m11s   v1.24.6
ip-10-0-96-61.us-west-2.compute.internal      Ready    <none>
8m12s   v1.24.6
```

NOTE: It may take a few minutes for the Status to move to `Ready` while the Pod network is deployed. The Nodes' Status should change to Ready soon after the `calico-node` DaemonSet Pods are Ready.

3. List the Pods using this command:

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf get --all-namespaces pods
```

```
NAMESPACE                NAME
READY   STATUS     RESTARTS        AGE
calico-system            calico-kube-controllers-57fbd7bd59-xlnfq
1/1     Running    0               9m41s
calico-system            calico-node-7lfmj
1/1     Running    0               9m2s
calico-system            calico-node-8jm52
1/1     Running    0               9m41s
calico-system            calico-node-cgrwx
1/1     Running    0               9m23s
```

```
calico-system            calico-node-g96f6
1/1     Running    0             9m22s
calico-system            calico-node-kvwgq
1/1     Running    0             9m21s
calico-system            calico-node-nbjr6
1/1     Running    0             7m32s
calico-system            calico-node-w5sb5
1/1     Running    0             9m24s
calico-system            calico-typha-8699c7db75-645pm
1/1     Running    0             9m22s
calico-system            calico-typha-8699c7db75-7czwp
1/1     Running    0             9m12s
calico-system            calico-typha-8699c7db75-xk6wn
1/1     Running    0             9m41s
kube-system              cluster-autoscaler-68c759fbf6-2bclx
0/1     Init:0/1   0             10m
kube-system              coredns-78fcd69978-882jg
1/1     Running    0             10m
kube-system              coredns-78fcd69978-ntp5f
1/1     Running    0             10m
kube-system              ebs-csi-controller-77574b5cf5-84z7l
6/6     Running    0             10m
kube-system              ebs-csi-controller-77574b5cf5-tsr5q
6/6     Running    0             10m
kube-system              ebs-csi-node-2j9pd
3/3     Running    0             9m23s
kube-system              ebs-csi-node-6t4sr
3/3     Running    0             9m2s
kube-system              ebs-csi-node-8qkpr
3/3     Running    0             10m
kube-system              ebs-csi-node-kbq57
3/3     Running    0             9m24s
kube-system              ebs-csi-node-qf9mh
3/3     Running    0             9m22s
kube-system              ebs-csi-node-v84dd
3/3     Running    0             9m21s
kube-system              ebs-csi-node-zqrtp
3/3     Running    0             7m32s
kube-system              etcd-ip-10-0-106-183.us-west-2.compute.internal
1/1     Running    0             7m31s
kube-system              etcd-ip-10-0-158-104.us-west-2.compute.internal
1/1     Running    0             10m
kube-system              etcd-ip-10-0-203-138.us-west-2.compute.internal
1/1     Running    0             9m1s
kube-system              kube-apiserver-ip-10-0-106-183.us-west-2.compute.inter
nal            1/1     Running    0             7m31s
kube-system              kube-apiserver-ip-10-0-158-104.us-west-2.compute.inter
nal            1/1     Running    0             10m
kube-system              kube-apiserver-ip-10-0-203-138.us-west-2.compute.inter
nal            1/1     Running    0             9m1s
kube-system              kube-controller-manager-ip-10-0-106-183.us-west-2.comp
ute.internal   1/1     Running    0             7m31s
```

```
kube-system                 kube-controller-manager-ip-10-0-158-104.us-west-2.comp
ute.internal   1/1     Running    1 (8m51s ago)   10m
kube-system                 kube-controller-manager-ip-10-0-203-138.us-west-2.comp
ute.internal   1/1     Running    0               9m2s
kube-system                 kube-proxy-4j9s5
1/1     Running    0               9m23s
kube-system                 kube-proxy-64svf
1/1     Running    0               9m2s
kube-system                 kube-proxy-9xghm
1/1     Running    0               7m32s
kube-system                 kube-proxy-cwbqm
1/1     Running    0               9m24s
kube-system                 kube-proxy-p6thh
1/1     Running    0               9m22s
kube-system                 kube-proxy-pgs47
1/1     Running    0               9m21s
kube-system                 kube-proxy-zrplb
1/1     Running    0               10m
kube-system                 kube-scheduler-ip-10-0-106-183.us-west-2.compute.inter
nal            1/1     Running    0               7m31s
kube-system                 kube-scheduler-ip-10-0-158-104.us-west-2.compute.inter
nal            1/1     Running    1 (8m51s ago)   10m
kube-system                 kube-scheduler-ip-10-0-203-138.us-west-2.compute.inter
nal            1/1     Running    0               9m2s
kube-system                 snapshot-controller-545b6bf98-k2fbv
1/1     Running    0               10m
kube-system                 snapshot-controller-545b6bf98-nf2m8
1/1     Running    0               10m
node-feature-discovery   node-feature-discovery-master-84c67dcbb6-gqfq8
1/1     Running    0               10m
node-feature-discovery   node-feature-discovery-worker-46b9r
1/1     Running    0               8m12s
node-feature-discovery   node-feature-discovery-worker-g6dw7
1/1     Running    0               8m12s
node-feature-discovery   node-feature-discovery-worker-jzkmt
1/1     Running    0               8m12s
node-feature-discovery   node-feature-discovery-worker-vmsbm
1/1     Running    0               8m12s
tigera-operator          tigera-operator-d499f5c8f-glx25
1/1     Running    1 (8m51s ago)   10m
```

6.2.5.5 Make the New AWS Cluster Self-Managed

Konvoy deploys all cluster lifecycle services to a bootstrap cluster, which then deploys a workload cluster. When the workload cluster is ready, move the cluster lifecycle services to the workload cluster, which makes the workload cluster self-managed. This section describes how to make a workload cluster self-managed.

Before starting, ensure you create a workload cluster as described in Create a New Cluster(see page 117).

Make the new Kubernetes cluster manage itself

1. Deploy cluster lifecycle services on the workload cluster:

By default, `create bootstrap controllers` configures the Cluster API controllers to use the AWS credentials from your environment. We recommend you use the `--with-aws-bootstrap-credentials=false` flag to configure the Cluster API controllers of your self-managed AWS cluster to use AWS IAM Instance Profiles, instead of the AWS credentials from your environment.

```
dkp create capi-components --with-aws-bootstrap-credentials=false --kubeconfig
${CLUSTER_NAME}.conf
```

```
✓ Initializing new CAPI components
```

2. Move the Cluster API objects from the bootstrap to the workload cluster:
The cluster lifecycle services on the workload cluster are ready, but the workload cluster configuration is on the bootstrap cluster. The `move` command moves the configuration, which takes the form of Cluster API Custom Resource objects, from the bootstrap to the workload cluster. This process is also called a Pivot[108].

```
dkp move capi-resources --to-kubeconfig ${CLUSTER_NAME}.conf
```

```
 ✓ Moving cluster resources
You can now view resources in the moved cluster by using the --kubeconfig flag
with kubectl. For example: kubectl --kubeconfig=aws-example.conf get nodes
```

NOTE: To ensure only one set of cluster lifecycle services manages the workload cluster, Konvoy first pauses reconciliation of the objects on the bootstrap cluster, then creates the objects on the workload cluster. As Konvoy copies the objects, the cluster lifecycle services on the workload cluster reconcile the objects. The workload cluster becomes self-managed after Konvoy creates all the objects. If it fails, the `move` command can be safely retried.

3. Wait for the cluster control-plane to be ready:

```
kubectl --kubeconfig ${CLUSTER_NAME}.conf wait --for=condition=ControlPlaneRead
y "clusters/${CLUSTER_NAME}" --timeout=20m
```

```
cluster.cluster.x-k8s.io/aws-example condition met
```

4. Use the cluster lifecycle services on the workload cluster to check the workload cluster status:

108 https://cluster-api.sigs.k8s.io/reference/glossary.html?highlight=pivot#pivot

NOTE: After moving the cluster lifecycle services to the workload cluster, remember to use Konvoy with the workload cluster kubeconfig.

```
dkp describe cluster --kubeconfig ${CLUSTER_NAME}.conf -c ${CLUSTER_NAME}
```

```
NAME                                                              READY
SEVERITY  REASON  SINCE  MESSAGE
Cluster/aws-example                                              True
109s
├──ClusterInfrastructure - AWSCluster/aws-example                 True
112s
├──ControlPlane - KubeadmControlPlane/aws-example-control-plane  True
109s
│  ├──Machine/aws-example-control-plane-55jh4                     True
111s
│  ├──Machine/aws-example-control-plane-6sn97                     True
111s
│  └──Machine/aws-example-control-plane-nx9v5                     True
110s
└──Workers
   └──MachineDeployment/aws-example-md-0                          True
114s
     ├──Machine/aws-example-md-0-cb9c9bbf7-hcl8z                  True
111s
     ├──Machine/aws-example-md-0-cb9c9bbf7-rtdqw                  True
111s
     ├──Machine/aws-example-md-0-cb9c9bbf7-t894m                  True
111s
     └──Machine/aws-example-md-0-cb9c9bbf7-td29r                  True
111s
```

5. Remove the bootstrap cluster, as the workload cluster is now self-managed:

```
dkp delete bootstrap
```

```
✓ Deleting bootstrap cluster
```

Known Limitations

NOTE: Be aware of these limitations in the current release of Konvoy.

- Before making a workload cluster self-managed, be sure that its control plane nodes have sufficient permissions for running Cluster API controllers. See IAM Policy Configuration(see page 145).
- Konvoy supports moving only one set of cluster objects from the bootstrap cluster to the workload cluster, or vice-versa.
- Konvoy only supports moving all namespaces in the cluster; Konvoy does not support migration of individual namespaces.

6.2.5.6 AWS Certificate Renewal

During cluster creation, Kubernetes establishes a Public Key Infrastructure (PKI) for generating the TLS certificates needed for securing cluster communication for various components such as `etcd`, `kubernetes-apiserver` and `kube-proxy`. The certificates created by these components have a default expiration of one year and are renewed when an administrator updates the cluster.

Kubernetes provides a facility to renew all certificates automatically during control plane updates. For administrators who need long-running clusters or clusters that are not upgraded often, `dkp` provides automated certificate renewal, without a cluster upgrade.

Prerequisites

- This feature requires Python 3.5 or greater to be installed on all control plane hosts.
- Complete the Bootstrap Cluster Lifecycle(see page 115) topic.

Create a cluster with automated certificate renewal

To enable the automated certificate renewal, create a Konvoy cluster using the `certificate-renew-interval` flag:

```
dkp create cluster aws --certificate-renew-interval=60 --cluster-name=long-running
```

The `certificate-renew-interval` is the number of days after which Kubernetes-managed PKI certificates will be renewed. For example, an `certificate-renew-interval` value of 60 means the certificates will be renewed every 60 days.

Technical details

The following manifests are modified on the control plane hosts, and are located at `/etc/kubernetes/manifests`. Modifications to these files requires SUDO access.

```
kube-controller-manager.yaml
kube-apiserver.yaml
kube-scheduler.yaml
kube-proxy.yaml
```

The following annotation indicates the time each component was reset:

```
metadata:
  annotations:
    konvoy.d2iq.io/restartedAt: $(date +%s)
```

This only occurs when the PKI certificates are older than the interval given at cluster creation time. This is activated by a `systemd` `timer` called `renew-certs.timer` that triggers an associated `systemd` service called `renew-certs.service` that runs on all of the control plane hosts.

Debugging

To debug the automatic certificate renewal feature, a cluster administrator can look at several different components to see if the certificates were renewed. For example, an administrator might start with a look at the control plane pod definition to check the last reset time. To determine if a scheduler pod was properly reset, run the command:

```
kubectl get pod -n kube-system kube-scheduler-ip-10-0-xx-xx.us-west-2.compute.interna
l -o yaml
```

The output of the command will be similar to the following:

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    konvoy.d2iq.io/restartedAt: "1626124940.735733"
```

Administrators who want more details on the execution of the `systemd` service can use `ssh` to connect to the control plane hosts, and then use the `systemctl` and `journalctl` commands that follow to help diagnose potential issues.

To check the status of the timers, when they last ran, and when they are scheduled to run next, use the command:

```
systemctl list-timers
```

To check the status of the `renew-certs` service, use the command:

```
systemctl status renew-certs
```

To get the logs of the last run of the service, use the command:

```
journalctl logs -u renew-certs
```

6.2.5.7 Configure Infrastructure in UI

This page refers to working inside the AWS Console in order to add Infrastructure Provider.

Create Infrastructure Provider

To configure an AWS Provider with a User Role in the AWS Console (UI), perform the following steps:

1. From the top menu bar, select your target workspace.
2. Select **Infrastructure Providers** in the **Administration** section of the sidebar menu.
3. Select the **Add Infrastructure Provider** button.
4. Select the **Amazon Web Services (AWS)** option.
5. Ensure **Role** is selected as the **Authentication Method**.
6. Enter a name for your infrastructure provider. Select a name that matches the AWS user.
7. Enter the **Role ARN**.
8. You can add an **External ID** if you share the Role with a 3rd party. External IDs secure your environment from accidentally used roles. Read more about External IDs[109].
9. Select **Save** to save your provider.

Fill out the Add Infrastructure Provider form

To fill out the Add Infrastructure Provider form in the AWS Console using Static Credentials, perform the following steps:

1. In Kommander, select the Workspace associated with the credentials you are adding.
2. Navigate to **Administration > Infrastructure Providers** and click the **Add Infrastructure Provider** button.
3. Select the Amazon Web Services (AWS) option.
4. Ensure **Static** is selected as the Authentication Method.
5. Enter a name for your infrastructure provider for later reference. Consider choosing a name that matches the AWS user.
6. Fill out the access and secret keys using the keys generated above.
7. Select **Save** to save your provider.

6.2.5.8 Delete an AWS Cluster

Delete your AWS cluster using the procedure on this page.

> **NOTE:** A self-managed workload cluster cannot delete itself. If your workload cluster is self-managed, you must create a bootstrap cluster and move the cluster lifecycle services to the bootstrap cluster before deleting the workload cluster.

If you did not make your workload cluster self-managed, as described in Make New Cluster Self-Managed(see page 129), see Delete the workload cluster(see page 136).

Follow these steps to delete an AWS Cluster

1. Create a bootstrap cluster:

109 https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create_for-user_externalid.html

The bootstrap cluster will host the Cluster API controllers that reconcile the cluster objects marked for deletion:
NOTE: To avoid using the wrong kubeconfig, the following steps use explicit kubeconfig paths and contexts.

```
dkp create bootstrap --kubeconfig $HOME/.kube/config --with-aws-bootstrap-
credentials=true
```

```
 ✓ Creating a bootstrap cluster
 ✓ Initializing new CAPI components
```

2. Move the Cluster API objects from the workload to the bootstrap cluster: The cluster lifecycle services on the bootstrap cluster are ready, but the workload cluster configuration is on the workload cluster. The `move` command moves the configuration, which takes the form of Cluster API Custom Resource objects, from the workload to the bootstrap cluster. This process is also called a Pivot[110].

```
dkp move capi-resources \
    --from-kubeconfig ${CLUSTER_NAME}.conf \
    --from-context ${CLUSTER_NAME}-admin@${CLUSTER_NAME} \
    --to-kubeconfig $HOME/.kube/config \
    --to-context kind-konvoy-capi-bootstrapper
```

```
✓ Moving cluster resources
You can now view resources in the moved cluster by using the --kubeconfig flag
with kubectl. For example: kubectl --kubeconfig $HOME/.kube/config get nodes
```

3. Use the cluster lifecycle services on the workload cluster to check the workload cluster status:

```
dkp describe cluster --kubeconfig $HOME/.kube/config -c ${CLUSTER_NAME}
```

```
NAME                                                          READY  SEVERITY
REASON  SINCE  MESSAGE
Cluster/aws-example                                           True
91s
 ├──ClusterInfrastructure - AWSCluster/aws-example             True
103s
 ├──ControlPlane - KubeadmControlPlane/aws-example-control-plane  True
91s
```

110 https://cluster-api.sigs.k8s.io/reference/glossary.html?highlight=pivot#pivot

```
│ ├──Machine/aws-example-control-plane-55jh4                    True
102s
│ ├──Machine/aws-example-control-plane-6sn97                    True
102s
│ └──Machine/aws-example-control-plane-nx9v5                    True
102s
└──Workers
  └──MachineDeployment/aws-example-md-0                         True
108s
    ├──Machine/aws-example-md-0-cb9c9bbf7-hcl8z                 True
102s
    ├──Machine/aws-example-md-0-cb9c9bbf7-rtdqw                 True
102s
    ├──Machine/aws-example-md-0-cb9c9bbf7-td29r                 True
102s
    └──Machine/aws-example-md-0-cb9c9bbf7-w64kg                 True
102s
```

NOTE: After moving the cluster lifecycle services to the workload cluster, remember to use dkp with the workload cluster kubeconfig.
Use dkp with the bootstrap cluster to delete the workload cluster.

4. Wait for the cluster control-plane to be ready:

```
kubectl --kubeconfig $HOME/.kube/config wait --for=condition=controlplaneready
"clusters/${CLUSTER_NAME}" --timeout=20m
```

```
cluster.cluster.x-k8s.io/aws-example condition met
```

Persistent Volumes (PVs) are not deleted automatically by design in order to preserve your data. However, they take up storage space if not deleted. You must delete PVs manually.

Delete the workload cluster

1. Make sure your AWS credentials are up to date. Refresh the credentials using this command:

```
dkp update bootstrap credentials aws --kubeconfig $HOME/.kube/config
```

2. Delete the Kubernetes cluster and wait a few minutes:
Before deleting the cluster, dkp deletes all Services of type LoadBalancer on the cluster. Each Service is backed by an AWS Classic ELB. Deleting the Service deletes the ELB that backs it. To skip this step, use the flag `--delete-kubernetes-resources=false`.

NOTE: Do not skip this step if the VPC is managed by DKP. When DKP deletes the cluster, it deletes the VPC. If

the VPC has any AWS Classic ELBs, AWS does not allow the VPC to be deleted, and DKP cannot delete the cluster.

```
dkp delete cluster --cluster-name=${CLUSTER_NAME} --kubeconfig $HOME/.kube/
config
```

```
✓ Deleting Services with type LoadBalancer for Cluster default/azure-example
✓ Deleting ClusterResourceSets for Cluster default/azure-example
✓ Deleting cluster resources
✓ Waiting for cluster to be fully deleted
Deleted default/azure-example cluster
```

After the workload cluster is deleted, delete the bootstrap cluster.

Delete the bootstrap cluster

After you have moved the workload resources back to a bootstrap cluster and deleted the workload cluster, you no longer need the bootstrap cluster. You can safely delete the bootstrap cluster with this command:

```
dkp delete bootstrap --kubeconfig $HOME/.kube/config
```

```
✓ Deleting bootstrap cluster
```

Known Limitations

NOTE: Be aware of these limitations in the current release of Konvoy.

- The Konvoy version used to create the workload cluster must match the Konvoy version used to delete the workload cluster.

6.2.5.9 Replace an AWS Node

Prerequisites

Before you begin, you must:

- Create a workload cluster(see page 117).
- Make the new cluster self-managed(see page 129).

Replace a worker node

In certain situations, you may want to delete a worker node and have Cluster API[111] replace it with a newly provisioned machine.

1. Identify the name of the node to delete.
   List the nodes:

   ```
   kubectl --kubeconfig ${CLUSTER_NAME}.conf get nodes
   ```

   The output from this command resembles the following:

   ```
   NAME                                          STATUS   ROLES
   AGE   VERSION
   ip-10-0-100-85.us-west-2.compute.internal     Ready    <none>
   16m   v1.24.6
   ip-10-0-106-183.us-west-2.compute.internal    Ready    control-plane,master
   15m   v1.24.6
   ip-10-0-158-104.us-west-2.compute.internal    Ready    control-plane,master
   17m   v1.24.6
   ip-10-0-203-138.us-west-2.compute.internal    Ready    control-plane,master
   16m   v1.24.6
   ip-10-0-70-169.us-west-2.compute.internal     Ready    <none>
   16m   v1.24.6
   ip-10-0-77-176.us-west-2.compute.internal     Ready    <none>
   16m   v1.24.6
   ip-10-0-96-61.us-west-2.compute.internal      Ready    <none>
   16m   v1.24.6
   ```

2. Export a variable with the node name to use in the next steps:
   This example uses the name `ip-10-0-100-85.us-west-2.compute.internal`.

   ```
   export NAME_NODE_TO_DELETE="<ip-10-0-100-85.us-west-2.compute.internal>"
   ```

3. Delete the Machine resource

   ```
   export NAME_MACHINE_TO_DELETE=$(kubectl --kubeconfig ${CLUSTER_NAME}.conf get
   machine -ojsonpath="{.items[?
   (@.status.nodeRef.name==\"$NAME_NODE_TO_DELETE\")].metadata.name}")
   kubectl --kubeconfig ${CLUSTER_NAME}.conf delete machine
   "$NAME_MACHINE_TO_DELETE"
   ```

111 https://cluster-api.sigs.k8s.io/

```
machine.cluster.x-k8s.io "aws-example-1-md-0-cb9c9bbf7-t894m" deleted
```

The command will not return immediately. It will return once the Machine resource has been deleted. A few minutes after the Machine resource is deleted, the corresponding Node resource is also deleted.

4. Observe that the Machine resource is being replaced using this command:

```
kubectl --kubeconfig ${CLUSTER_NAME}.conf get machinedeployment
```

```
NAME                CLUSTER       REPLICAS   READY   UPDATED   UNAVAILABLE
PHASE       AGE     VERSION
aws-example-md-0    aws-example   4          3       4         1
ScalingUp   7m53s   v1.24.6
```

In this example, there are two replicas, but only 1 is ready. One replica is unavailable, and the `ScalingUp` phase means a new Machine is being created.

5. Identify the replacement Machine using this command:

```
export NAME_NEW_MACHINE=$(kubectl --kubeconfig ${CLUSTER_NAME}.conf get
machines \
    -l=cluster.x-k8s.io/deployment-name=${CLUSTER_NAME}-md-0 \
    -ojsonpath='{.items[?(@.status.phase=="Running")].metadata.name}{"\n"}')
echo "$NAME_NEW_MACHINE"
```

```
aws-example-md-0-cb9c9bbf7-hcl8z aws-example-md-0-cb9c9bbf7-rtdqw aws-example-
md-0-cb9c9bbf7-td29r aws-example-md-0-cb9c9bbf7-w64kg
```

If the output is empty, the new Machine has probably exited the `Provisioning` phase and entered the `Running` phase.

6. Identify the replacement Node using this command:

```
kubectl --kubeconfig ${CLUSTER_NAME}.conf get nodes
```

```
NAME                                          STATUS   ROLES
AGE   VERSION
ip-10-0-106-183.us-west-2.compute.internal   Ready    control-plane,master
20m   v1.24.6
```

```
ip-10-0-158-104.us-west-2.compute.internal   Ready    control-plane,master
23m   v1.24.6
ip-10-0-203-138.us-west-2.compute.internal   Ready    control-plane,master
22m   v1.24.6
ip-10-0-70-169.us-west-2.compute.internal    Ready    <none>
22m   v1.24.6
ip-10-0-77-176.us-west-2.compute.internal    Ready    <none>
22m   v1.24.6
ip-10-0-86-58.us-west-2.compute.internal     Ready    <none>
57s   v1.24.6
ip-10-0-96-61.us-west-2.compute.internal     Ready    <none>
22m   v1.24.6
```

If the output is empty, the Node resource is not yet available, or does not yet have the expected annotation. Wait a few minutes, then repeat the command.

6.2.6 Minimal Permissions and Role to Create Clusters

Configure IAM Prerequisites before starting a cluster

This section guides you in creating and using a minimally-scoped policy to create DKP clusters on an AWS account.

6.2.6.1 Prerequisites

Before applying the IAM Policies, verify the following:

- You have a valid AWS account with credentials configured[112] that can manage CloudFormation Stacks, IAM Policies, IAM Roles, and IAM Instance Profiles.
- The AWS CLI utility[113] is installed.

6.2.6.2 Minimal Permissions

The following is an AWSCloudformation stack that creates:

- A policy named `dkp-bootstrapper-policy` that enumerates the minimal permissions for a user that can create dkp aws clusters.
- A role named `dkp-bootstrapper-role` that uses the `dkp-bootstrapper-policy` with a trust policy to allow IAM users and ec2 instances from `MYAWSACCOUNTID` to use the role via STS.
- An instance profile `DKPBootstrapInstanceProfile` that wraps the `dkp-bootstrapper-role` to be used by ec2 instances.

Create Resources in Cloudformation Stack

To create the resources in the cloudformation stack:

1. Copy the following contents into a file:

112 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-profiles.html
113 https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  AWSIAMInstanceProfileDKPBootstrapper:
    Properties:
      InstanceProfileName: DKPBootstrapInstanceProfile
      Roles:
      - Ref: DKPBootstrapRole
    Type: AWS::IAM::InstanceProfile
  AWSIAMManagedPolicyDKPBootstrapper:
    Properties:
      Description: Minimal policy to create dkp clusters in AWS
      ManagedPolicyName: dkp-bootstrapper-policy
      PolicyDocument:
        Statement:
        - Action:
          - ec2:AllocateAddress
          - ec2:AssociateRouteTable
          - ec2:AttachInternetGateway
          - ec2:AuthorizeSecurityGroupIngress
          - ec2:CreateInternetGateway
          - ec2:CreateNatGateway
          - ec2:CreateRoute
          - ec2:CreateRouteTable
          - ec2:CreateSecurityGroup
          - ec2:CreateSubnet
          - ec2:CreateTags
          - ec2:CreateVpc
          - ec2:ModifyVpcAttribute
          - ec2:DeleteInternetGateway
          - ec2:DeleteNatGateway
          - ec2:DeleteRouteTable
          - ec2:DeleteSecurityGroup
          - ec2:DeleteSubnet
          - ec2:DeleteTags
          - ec2:DeleteVpc
          - ec2:DescribeAccountAttributes
          - ec2:DescribeAddresses
          - ec2:DescribeAvailabilityZones
          - ec2:DescribeInstances
          - ec2:DescribeInternetGateways
          - ec2:DescribeImages
          - ec2:DescribeNatGateways
          - ec2:DescribeNetworkInterfaces
          - ec2:DescribeNetworkInterfaceAttribute
          - ec2:DescribeRouteTables
          - ec2:DescribeSecurityGroups
          - ec2:DescribeSubnets
          - ec2:DescribeVpcs
          - ec2:DescribeVpcAttribute
          - ec2:DescribeVolumes
          - ec2:DetachInternetGateway
```

```
      - ec2:DisassociateRouteTable
      - ec2:DisassociateAddress
      - ec2:ModifyInstanceAttribute
      - ec2:ModifyNetworkInterfaceAttribute
      - ec2:ModifySubnetAttribute
      - ec2:ReleaseAddress
      - ec2:RevokeSecurityGroupIngress
      - ec2:RunInstances
      - ec2:TerminateInstances
      - tag:GetResources
      - elasticloadbalancing:AddTags
      - elasticloadbalancing:CreateLoadBalancer
      - elasticloadbalancing:ConfigureHealthCheck
      - elasticloadbalancing:DeleteLoadBalancer
      - elasticloadbalancing:DescribeLoadBalancers
      - elasticloadbalancing:DescribeLoadBalancerAttributes
      - elasticloadbalancing:ApplySecurityGroupsToLoadBalancer
      - elasticloadbalancing:DescribeTags
      - elasticloadbalancing:ModifyLoadBalancerAttributes
      - elasticloadbalancing:RegisterInstancesWithLoadBalancer
      - elasticloadbalancing:DeregisterInstancesFromLoadBalancer
      - elasticloadbalancing:RemoveTags
      - autoscaling:DescribeAutoScalingGroups
      - autoscaling:DescribeInstanceRefreshes
      - ec2:CreateLaunchTemplate
      - ec2:CreateLaunchTemplateVersion
      - ec2:DescribeLaunchTemplates
      - ec2:DescribeLaunchTemplateVersions
      - ec2:DeleteLaunchTemplate
      - ec2:DeleteLaunchTemplateVersions
      - ec2:DescribeKeyPairs
      Effect: Allow
      Resource:
      - '*'
    - Action:
      - autoscaling:CreateAutoScalingGroup
      - autoscaling:UpdateAutoScalingGroup
      - autoscaling:CreateOrUpdateTags
      - autoscaling:StartInstanceRefresh
      - autoscaling:DeleteAutoScalingGroup
      - autoscaling:DeleteTags
      Effect: Allow
      Resource:
      - arn:*:autoscaling:*:*:autoScalingGroup:*:autoScalingGroupName/*
    - Action:
      - iam:CreateServiceLinkedRole
      Condition:
        StringLike:
          iam:AWSServiceName: autoscaling.amazonaws.com
      Effect: Allow
      Resource:
```

```
          - arn:*:iam::*:role/aws-service-role/autoscaling.amazonaws.com/
AWSServiceRoleForAutoScaling
        - Action:
          - iam:CreateServiceLinkedRole
          Condition:
            StringLike:
              iam:AWSServiceName: elasticloadbalancing.amazonaws.com
          Effect: Allow
          Resource:
          - arn:*:iam::*:role/aws-service-role/
elasticloadbalancing.amazonaws.com/AWSServiceRoleForElasticLoadBalancing
        - Action:
          - iam:CreateServiceLinkedRole
          Condition:
            StringLike:
              iam:AWSServiceName: spot.amazonaws.com
          Effect: Allow
          Resource:
          - arn:*:iam::*:role/aws-service-role/spot.amazonaws.com/
AWSServiceRoleForEC2Spot
        - Action:
          - iam:PassRole
          Effect: Allow
          Resource:
          - arn:*:iam::*:role/*.cluster-api-provider-aws.sigs.k8s.io
        - Action:
          - secretsmanager:CreateSecret
          - secretsmanager:DeleteSecret
          - secretsmanager:TagResource
          Effect: Allow
          Resource:
          - arn:*:secretsmanager:*:*:secret:aws.cluster.x-k8s.io/*
        Version: 2012-10-17
      Roles:
      - Ref: DKPBootstrapRole
    Type: AWS::IAM::ManagedPolicy
  DKPBootstrapRole:
    Properties:
      AssumeRolePolicyDocument:
        Statement:
        - Action:
          - sts:AssumeRole
          Effect: Allow
          Principal:
            Service:
            - ec2.amazonaws.com
        - Action:
            - sts:AssumeRole
          Effect: Allow
          Principal:
                AWS: arn:aws:iam::MYAWSACCOUNT:root
        Version: 2012-10-17
```

```
      RoleName: dkp-bootstrapper-role
    Type: AWS::IAM::Role
```

2. Replace the following with the correct values:
   a. `MYFILENAME.yaml` - give your file a meaningful name.
   b. `MYSTACKNAME` - give your cloudformation stack a meaningful name.
   c. `MYAWSACCOUNT` - replace with an AWS Account ID number such as: `111122223333`
3. Run the following command to create the stack :

```
aws cloudformation create-stack --template-body=file://MYFILENAME.yaml --stack-
name=MYSTACKNAME --capabilities CAPABILITY_NAMED_IAM
```

6.2.6.3 Leverage the Role

Temporary User Access Keys via STS

The created `dkp-bootstrapper-role` can be assumed by IAM users for temporary credentials via STS by running the command below:

```
aws sts assume-role --role-arn arn:aws:iam::MYAWSACCOUNT:role/dkp-bootstrapper-role
--role-session-name EXAMPLE
```

Which returns something similar to this:

```
{
    "Credentials": {
        "AccessKeyId": "ASIA6RTF53ZH5B52EVM5",
        "SecretAccessKey": "BSssyvSsdfJY74jubsadfdsafdsaH7x1L+8Vk/",
        "SessionToken": "IQoJb3JpZ2z5cyChb9PtJvP0S6KAi",
        "Expiration": "2022-07-14T20:19:13+00:00"
    },
    "AssumedRoleUser": {
        "AssumedRoleId": "ASIA6RTF53ZH5B52EVM5:test",
        "Arn": "arn:aws:sts::MYAWSACCOUNTID:assumed-role/dkp-bootstrapper-role/test"
    }
}
```

And then `export` the following environment variables with the results:

```
export AWS_ACCESS_KEY_ID=(.Credentials.AccessKeyId)

export AWS_SECRET_ACCESS_KEY=(.Credentials.SecretAccessKey)
```

```
export AWS_SESSION_TOKEN=(.Credentials.SessionToken)
```

These credentials are short lived and would need to be updated in the bootstrap cluster

Use EC2 Instance Profiles

The created `dkp-bootstrapper-role` can be assumed by an ec2 instance a user would run `dkp create cluster` commands from. To do this, specify the IAM Instance Profile `DKPBootstrapInstanceProfile` on creation.

Use Access Keys

AWS administrators can attach the `dkp-bootstrapper-policy` to an existing IAM user[114] and authenticate with Access Keys[115] on the work station they would run `dkp create cluster` commands from by exporting the following environment variables with the appropriate values for the IAM user.

```
export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
export AWS_DEFAULT_REGION=us-west-2
```

In regards to Access Keys usage, a system administrator should always consider AWS's Best practices[116].

6.2.7 Cluster IAM Policies and Roles

This guides a DKP user in creating IAM Policies and Instance Profiles used by the cluster's control plane and worker nodes using the provided AWS CloudFormation Stack.

6.2.7.1 Prerequisites

Before applying the IAM Policies, verify the following:

- You have a valid AWS account with credentials configured[117] that can manage CloudFormation Stacks, IAM Policies, IAM Roles, and IAM Instance Profiles.
- You will need to have the AWS CLI utility installed[118].

6.2.7.2 IAM Artifacts

Below is information about the following areas of setup. After reading the information for each of these areas, you will find the CloudFormation Stack that creates:

- IAM Policies

114 https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_manage-attach-detach.html
115 https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html
116 https://docs.aws.amazon.com/general/latest/gr/aws-access-keys-best-practices.html
117 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-profiles.html
118 https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html

- IAM Roles
- IAM Instance Profiles

Policies

1. `AWSIAMManagedPolicyCloudProviderControlPlane` enumerates the Actions required by the workload cluster control plane machines. It is attached to the `AWSIAMRoleControlPlane` Role.
2. `AWSIAMManagedPolicyCloudProviderNodes` enumerates the Actions required by the workload cluster worker machines. It is attached to the `AWSIAMRoleNodes` Role.
3. `AWSIAMManagedPolicyControllers` enumerates the Actions required by the workload cluster worker machines. It is attached to the `AWSIAMRoleControlPlane` Role.

Roles

1. `AWSIAMRoleControlPlane` is the Role associated with the `AWSIAMInstanceProfileControlPlane` Instance Profile.
2. `AWSIAMRoleNodes` is the Role associated with the `AWSIAMInstanceProfileNodes` Instance Profile.

For more information on learning how to grant cluster access to IAM users and roles, see the official AWS Documentation[119].

Instance Profiles

1. `AWSIAMInstanceProfileControlPlane`, assigned to workload cluster control plane machines.

> If the name is changed from the default, used below, it must be passed to `dkp create cluster` wit h the `--control-plane-iam-instance-profile` flag.

2. `AWSIAMInstanceProfileNodes`, assigned to workload cluster worker machines.

> If the name is changed from the default, used below, it must be passed to `dkp create cluster` wit h the `--worker-iam-instance-profile` flag.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  AWSIAMInstanceProfileControlPlane:
    Properties:
      InstanceProfileName: control-plane.cluster-api-provider-aws.sigs.k8s.io
```

119 https://docs.aws.amazon.com/eks/latest/userguide/add-user-role.html

```
    Roles:
    - Ref: AWSIAMRoleControlPlane
  Type: AWS::IAM::InstanceProfile
AWSIAMInstanceProfileNodes:
  Properties:
    InstanceProfileName: nodes.cluster-api-provider-aws.sigs.k8s.io
    Roles:
    - Ref: AWSIAMRoleNodes
  Type: AWS::IAM::InstanceProfile
AWSIAMManagedPolicyCloudProviderControlPlane:
  Properties:
    Description: For the Kubernetes Cloud Provider AWS Control Plane
    ManagedPolicyName: control-plane.cluster-api-provider-aws.sigs.k8s.io
    PolicyDocument:
      Statement:
      - Action:
        - autoscaling:DescribeAutoScalingGroups
        - autoscaling:DescribeLaunchConfigurations
        - autoscaling:DescribeTags
        - ec2:DescribeInstances
        - ec2:DescribeImages
        - ec2:DescribeRegions
        - ec2:DescribeRouteTables
        - ec2:DescribeSecurityGroups
        - ec2:DescribeSubnets
        - ec2:DescribeVolumes
        - ec2:CreateSecurityGroup
        - ec2:CreateTags
        - ec2:CreateVolume
        - ec2:ModifyInstanceAttribute
        - ec2:ModifyVolume
        - ec2:AttachVolume
        - ec2:AuthorizeSecurityGroupIngress
        - ec2:CreateRoute
        - ec2:DeleteRoute
        - ec2:DeleteSecurityGroup
        - ec2:DeleteVolume
        - ec2:DetachVolume
        - ec2:RevokeSecurityGroupIngress
        - ec2:DescribeVpcs
        - elasticloadbalancing:AddTags
        - elasticloadbalancing:AttachLoadBalancerToSubnets
        - elasticloadbalancing:ApplySecurityGroupsToLoadBalancer
        - elasticloadbalancing:CreateLoadBalancer
        - elasticloadbalancing:CreateLoadBalancerPolicy
        - elasticloadbalancing:CreateLoadBalancerListeners
        - elasticloadbalancing:ConfigureHealthCheck
        - elasticloadbalancing:DeleteLoadBalancer
        - elasticloadbalancing:DeleteLoadBalancerListeners
        - elasticloadbalancing:DescribeLoadBalancers
        - elasticloadbalancing:DescribeLoadBalancerAttributes
        - elasticloadbalancing:DetachLoadBalancerFromSubnets
```

```
        - elasticloadbalancing:DeregisterInstancesFromLoadBalancer
        - elasticloadbalancing:ModifyLoadBalancerAttributes
        - elasticloadbalancing:RegisterInstancesWithLoadBalancer
        - elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer
        - elasticloadbalancing:AddTags
        - elasticloadbalancing:CreateListener
        - elasticloadbalancing:CreateTargetGroup
        - elasticloadbalancing:DeleteListener
        - elasticloadbalancing:DeleteTargetGroup
        - elasticloadbalancing:DescribeListeners
        - elasticloadbalancing:DescribeLoadBalancerPolicies
        - elasticloadbalancing:DescribeTargetGroups
        - elasticloadbalancing:DescribeTargetHealth
        - elasticloadbalancing:ModifyListener
        - elasticloadbalancing:ModifyTargetGroup
        - elasticloadbalancing:RegisterTargets
        - elasticloadbalancing:SetLoadBalancerPoliciesOfListener
        - iam:CreateServiceLinkedRole
        - kms:DescribeKey
        Effect: Allow
        Resource:
        - '*'
      Version: 2012-10-17
    Roles:
    - Ref: AWSIAMRoleControlPlane
  Type: AWS::IAM::ManagedPolicy
AWSIAMManagedPolicyCloudProviderNodes:
  Properties:
    Description: For the Kubernetes Cloud Provider AWS nodes
    ManagedPolicyName: nodes.cluster-api-provider-aws.sigs.k8s.io
    PolicyDocument:
      Statement:
      - Action:
        - ec2:DescribeInstances
        - ec2:DescribeRegions
        - ecr:GetAuthorizationToken
        - ecr:BatchCheckLayerAvailability
        - ecr:GetDownloadUrlForLayer
        - ecr:GetRepositoryPolicy
        - ecr:DescribeRepositories
        - ecr:ListImages
        - ecr:BatchGetImage
        Effect: Allow
        Resource:
        - '*'
      - Action:
        - secretsmanager:DeleteSecret
        - secretsmanager:GetSecretValue
        Effect: Allow
        Resource:
        - arn:*:secretsmanager:*:*:secret:aws.cluster.x-k8s.io/*
      - Action:
```

```
        - ssm:UpdateInstanceInformation
        - ssmmessages:CreateControlChannel
        - ssmmessages:CreateDataChannel
        - ssmmessages:OpenControlChannel
        - ssmmessages:OpenDataChannel
        - s3:GetEncryptionConfiguration
        Effect: Allow
        Resource:
        - '*'
      Version: 2012-10-17
    Roles:
    - Ref: AWSIAMRoleControlPlane
    - Ref: AWSIAMRoleNodes
  Type: AWS::IAM::ManagedPolicy
AWSIAMManagedPolicyControllers:
  Properties:
    Description: For the Kubernetes Cluster API Provider AWS Controllers
    ManagedPolicyName: controllers.cluster-api-provider-aws.sigs.k8s.io
    PolicyDocument:
      Statement:
      - Action:
        - ec2:AllocateAddress
        - ec2:AssociateRouteTable
        - ec2:AttachInternetGateway
        - ec2:AuthorizeSecurityGroupIngress
        - ec2:CreateInternetGateway
        - ec2:CreateNatGateway
        - ec2:CreateRoute
        - ec2:CreateRouteTable
        - ec2:CreateSecurityGroup
        - ec2:CreateSubnet
        - ec2:CreateTags
        - ec2:CreateVpc
        - ec2:ModifyVpcAttribute
        - ec2:DeleteInternetGateway
        - ec2:DeleteNatGateway
        - ec2:DeleteRouteTable
        - ec2:DeleteSecurityGroup
        - ec2:DeleteSubnet
        - ec2:DeleteTags
        - ec2:DeleteVpc
        - ec2:DescribeAccountAttributes
        - ec2:DescribeAddresses
        - ec2:DescribeAvailabilityZones
        - ec2:DescribeInstances
        - ec2:DescribeInternetGateways
        - ec2:DescribeImages
        - ec2:DescribeNatGateways
        - ec2:DescribeNetworkInterfaces
        - ec2:DescribeNetworkInterfaceAttribute
        - ec2:DescribeRouteTables
        - ec2:DescribeSecurityGroups
```

```
    - ec2:DescribeSubnets
    - ec2:DescribeVpcs
    - ec2:DescribeVpcAttribute
    - ec2:DescribeVolumes
    - ec2:DetachInternetGateway
    - ec2:DisassociateRouteTable
    - ec2:DisassociateAddress
    - ec2:ModifyInstanceAttribute
    - ec2:ModifyNetworkInterfaceAttribute
    - ec2:ModifySubnetAttribute
    - ec2:ReleaseAddress
    - ec2:RevokeSecurityGroupIngress
    - ec2:RunInstances
    - ec2:TerminateInstances
    - tag:GetResources
    - elasticloadbalancing:AddTags
    - elasticloadbalancing:CreateLoadBalancer
    - elasticloadbalancing:ConfigureHealthCheck
    - elasticloadbalancing:DeleteLoadBalancer
    - elasticloadbalancing:DescribeLoadBalancers
    - elasticloadbalancing:DescribeLoadBalancerAttributes
    - elasticloadbalancing:ApplySecurityGroupsToLoadBalancer
    - elasticloadbalancing:DescribeTags
    - elasticloadbalancing:ModifyLoadBalancerAttributes
    - elasticloadbalancing:RegisterInstancesWithLoadBalancer
    - elasticloadbalancing:DeregisterInstancesFromLoadBalancer
    - elasticloadbalancing:RemoveTags
    - autoscaling:DescribeAutoScalingGroups
    - autoscaling:DescribeInstanceRefreshes
    - ec2:CreateLaunchTemplate
    - ec2:CreateLaunchTemplateVersion
    - ec2:DescribeLaunchTemplates
    - ec2:DescribeLaunchTemplateVersions
    - ec2:DeleteLaunchTemplate
    - ec2:DeleteLaunchTemplateVersions
    Effect: Allow
    Resource:
    - '*'
  - Action:
    - autoscaling:CreateAutoScalingGroup
    - autoscaling:UpdateAutoScalingGroup
    - autoscaling:CreateOrUpdateTags
    - autoscaling:StartInstanceRefresh
    - autoscaling:DeleteAutoScalingGroup
    - autoscaling:DeleteTags
    Effect: Allow
    Resource:
    - arn:*:autoscaling:*:*:autoScalingGroup:*:autoScalingGroupName/*
  - Action:
    - iam:CreateServiceLinkedRole
    Condition:
      StringLike:
```

```
              iam:AWSServiceName: autoscaling.amazonaws.com
          Effect: Allow
          Resource:
          - arn:*:iam::*:role/aws-service-role/autoscaling.amazonaws.com/
AWSServiceRoleForAutoScaling
        - Action:
          - iam:CreateServiceLinkedRole
          Condition:
            StringLike:
              iam:AWSServiceName: elasticloadbalancing.amazonaws.com
          Effect: Allow
          Resource:
          - arn:*:iam::*:role/aws-service-role/elasticloadbalancing.amazonaws.com/
AWSServiceRoleForElasticLoadBalancing
        - Action:
          - iam:CreateServiceLinkedRole
          Condition:
            StringLike:
              iam:AWSServiceName: spot.amazonaws.com
          Effect: Allow
          Resource:
          - arn:*:iam::*:role/aws-service-role/spot.amazonaws.com/
AWSServiceRoleForEC2Spot
        - Action:
          - iam:PassRole
          Effect: Allow
          Resource:
          - arn:*:iam::*:role/*.cluster-api-provider-aws.sigs.k8s.io
        - Action:
          - secretsmanager:CreateSecret
          - secretsmanager:DeleteSecret
          - secretsmanager:TagResource
          Effect: Allow
          Resource:
          - arn:*:secretsmanager:*:*:secret:aws.cluster.x-k8s.io/*
        Version: 2012-10-17
      Roles:
      - Ref: AWSIAMRoleControlPlane
    Type: AWS::IAM::ManagedPolicy
  AWSIAMRoleControlPlane:
    Properties:
      AssumeRolePolicyDocument:
        Statement:
        - Action:
          - sts:AssumeRole
          Effect: Allow
          Principal:
            Service:
            - ec2.amazonaws.com
        Version: 2012-10-17
      RoleName: control-plane.cluster-api-provider-aws.sigs.k8s.io
    Type: AWS::IAM::Role
```

```
  AWSIAMRoleNodes:
    Properties:
      AssumeRolePolicyDocument:
        Statement:
        - Action:
          - sts:AssumeRole
          Effect: Allow
          Principal:
            Service:
            - ec2.amazonaws.com
        Version: 2012-10-17
      RoleName: nodes.cluster-api-provider-aws.sigs.k8s.io
    Type: AWS::IAM::Role
```

To create the resources in the cloudformation stack copy the contents above into a file and run the following command:

```
 aws cloudformation create-stack --template-body=file://MYFILENAME.yaml --stack-
name=MYSTACKNAME --capabilities  CAPABILITY_NAMED_IAM
```

replacing `MYFILENAME.yaml` and `MYSTACKNAME` with the intended values.

6.2.8 Multiple AWS Accounts

Leverage Multiple AWS Accounts for Kubernetes Cluster Deployments

6.2.8.1 Objective

You can leverage multiple AWS accounts in your organization to meet specific business purposes, reflect your organizational structure, or implement a multi-tenancy strategy. Specific scenarios include:

- Implementing isolation between environment tiers such as development, testing, acceptance, and production.
- Implementing separation of concerns between management clusters, and workload clusters.
- Reducing the impact of security events and incidents.

For additional benefits of using multiple AWS accounts, refer to the following white paper[120].

This document describes how to leverage the D2iQ Kubernetes Platform (DKP) to deploy a management cluster, and multiple workload clusters, leveraging multiple AWS accounts.

6.2.8.2 Assumptions

This guide assumes you have some understanding of Cluster API concepts and basic DKP provisioning workflows on AWS.

Cluster API Concepts - cluster API concepts[121]

120 https://docs.aws.amazon.com/whitepapers/latest/organizing-your-aws-environment/benefits-of-using-multiple-aws-accounts.html
121 https://cluster-api.sigs.k8s.io/user/concepts.html

Getting Started with DKP on AWS - AWS Quick Start(see page 37)

6.2.8.3 Glossary

- **Management cluster** - The cluster that runs in AWS and is used to create target clusters in different AWS accounts.
- **Target account** - The account where the target cluster is created.
- **Source account** - The AWS account where the CAPA controllers for the management cluster runs.

6.2.8.4 Prerequisites

Before you begin deploying DKP on AWS, you must:

Configure the Prerequisites(see page 114)

6.2.8.5 Deploy DKP on AWS

1. Deploy a management cluster in your AWS source account.
   AWS: create Kubernetes AWS cluster(see page 117)
2. Configure a trusted relationship between source and target accounts and create a management cluster:

Step 1:

DKP leverages the Cluster API provider for AWS (CAPA) to provision Kubernetes clusters in a declarative way. Customers declare the desired state of the cluster through a cluster configuration YAML file which is generated using:

(AWS)

```
dkp create cluster aws --cluster-name=${CLUSTER_NAME} \
--dry-run \
--output=yaml \
> ${CLUSTER_NAME}.yaml
```

Step 2:

Configure a trust relationship between the source and target accounts.

Follow all the prerequisite steps in both the source and target accounts

1. Create all policies and roles in management and workload accounts a. The prerequisite IAM policies for DKP are documented here: Configure AWS IAM policies(see page 145).
2. Establish a trust relationship in workload account for the management account.
   a. Go to your target (workload) account b. Search for the role control-plane.cluster-api-provider-aws.sigs.k8s.io c. Navigate to the Trust Relationship tab and select Edit Trust Relationship d. Add the following relationship:

   ```
   {
     "Effect": "Allow",
     "Principal": {
   ```

```
      "AWS": "arn:aws:iam::${mgmt-aws-account}:role/control-plane.cluster-api-
provider-aws.sigs.k8s.io"
    },
    "Action": "sts:AssumeRole"
}
```

3. Give permission to role in the source (management cluster) account to call the `sts:AssumeRole API` a. Log in to the source AWS account and attach the following inline policy to control-plane.cluster-api-provider-aws.sigs.k8s.io role:

```
{
  "Version": "2012-10-17",
  "Statement": [
     {
       "Effect": "Allow",
       "Action": "sts:AssumeRole",
       "Resource": [
"arn:aws:iam::${workload-aws-account}:role/control-plane.cluster-api-provider-
aws.sigs.k8s.io"
       ]
     }
  ]
}
```

4. Modify the management cluster configuration file and update the AWSCluster object with following details:

```
apiVersion: infrastructure.cluster.x-k8s.io/v1alpha3
kind: AWSCluster
metadata:
spec:
  identityRef:
     kind: AWSClusterRoleIdentity
     name: cross-account-role
…
…
---
apiVersion: infrastructure.cluster.x-k8s.io/v1alpha3
kind: AWSClusterRoleIdentity
metadata:
  name: cross-account-role
spec:
  allowedNamespaces: {}
  roleARN: "arn:aws:iam::${workload-aws-account}:role/control-plane.cluster-
api-provider-aws.sigs.k8s.io"
  sourceIdentityRef:
    kind: AWSClusterControllerIdentity
    name: default
```

After performing the above steps, your Management cluster will be configured to create new managed clusters in the target AWS workload account.

6.2.9 Install AWS Air-Gapped

6.2.9.1 Create a Kubernetes cluster in a private subnet with no access to the Internet (air-gapped)

This section provides the instructions to create a Kubernetes cluster in an existing VPC with the nodes and the kube-apiserver ELB in a private subnet with no access to the Internet.

6.2.9.2 AWS Air-gapped Prerequisites

Air-gapped installation prerequisites

Prerequisites

Before you begin, you must have:

- Linux machine (bastion) that has access to the existing VPC.
- The `dkp` binary on the bastion.
- Docker[122] version 18.09.2 or later installed on the bastion.
- kubectl[123] for interacting with the running cluster on the bastion.
- Valid AWS account with credentials configured[124].
- An existing docker registry.
- Ability to download artifacts from the internet and then copy those onto your bootstrap machine.
- An AWS Air-Gapped Machine Image(see page 326)

Air-Gapped Registry Prerequisites

JFrog Artifactory
If you use **Jfrog Artifactory** or **Jfrog Container Registry**, you must update to a new version of the software. Any build newer than version **7.11** will work, as we have confirmed that older versions are not compatible.

Nexus Registry
If you use **Nexus Registry**, there is currently an issue that prevents usage with DKP 2.X and OCI Images. Support for OCI Images was added here in this publicly available Jira ticket:

https://issues.sonatype.org/browse/NEXUS-21087

A new issue was filed here determining that OCI image support is currently broken:

---

122 https://docs.docker.com/get-docker/
123 https://kubernetes.io/docs/tasks/tools/#kubectl
124 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-profiles.html

https://issues.sonatype.org/plugins/servlet/mobile#issue/NEXUS-27494

You can track this Jira link for status on a resolution for this issue.

Harbor Registry
Any newer version than **Harbor Registry v2.1.1**-5f52168e will support OCI images.

Configure AWS prerequisites

1. Follow the steps in IAM Policy Configuration(see page 140).
2. Export the AWS region where you want to deploy the cluster:

```
export AWS_REGION=us-west-2
```

3. Export the AWS profile with the credentials you want to use to create the Kubernetes cluster:

```
export AWS_PROFILE=<profile>
```

6.2.9.3 AWS Air-gapped Seed Docker Registry

Seed your docker registry

Before creating a Kubernetes cluster you must have the required images in a local docker registry. This registry must be accessible from both the bastion machine and the AWS EC2 instances that will be created for the Kubernetes cluster.

1. Download the images bundle.

```
curl -o konvoy-image-bundle.tar -O https://downloads.d2iq.com/dkp/v2.3.3/
konvoy_image_bundle_v2.3.3_linux_amd64.tar
```

2. Place the bundle in a location where you can load and push the images to your private docker registry.
3. Set an environment variable with your registry address.

```
export DOCKER_REGISTRY_ADDRESS=<registry-address>:<registry-port>
export DOCKER_REGISTRY_USERNAME=<username>
export DOCKER_REGISTRY_PASSWORD=<password>
```

4. Run the following command to load the air-gapped image bundle into your private Docker registry.

```
dkp push image-bundle --image-bundle konvoy-image-bundle.tar.gz --to-registry
$DOCKER_REGISTRY_ADDRESS --to-registry-username $DOCKER_REGISTRY_USERNAME --to-
registry-password $DOCKER_REGISTRY_PASSWORD
```

It may take a while to push all the images to your image registry, depending on the performance of the network between the machine you are running the script on and the Docker registry.

Then, begin by Creating the Bootstrap cluster(see page 157).

This Docker image includes code from the MinIO Project ("MinIO"), which is © 2015-2021 MinIO, Inc. MinIO is made available subject to the terms and conditions of the GNU Affero General Public License 3.0. The complete source code for the versions of MinIO packaged with DKP/Kommander/Konvoy 2.3.2 are available at these URLs:

https://github.com/minio/minio/tree/RELEASE.2022-01-08T03-11-54Z

https://github.com/minio/minio/tree/RELEASE.2021-02-14T04-01-33Z

6.2.9.4 AWS Air-gapped Bootstrap

Bootstrap a kind cluster and CAPI controllers

Konvoy deploys all cluster lifecycle services to a bootstrap cluster, which deploys a workload cluster. When the workload cluster is ready, move the cluster lifecycle services to the workload cluster, after which the workload cluster manages its own lifecycle.

1. Download the bootstrap docker image on a machine that has access to this artifact.

   ```
   curl -O https://downloads.d2iq.com/dkp/v2.3.3/konvoy-bootstrap_v2.3.3.tar
   ```

2. Load the bootstrap docker image on your bastion machine.

   ```
   docker load -i konvoy-bootstrap_v2.3.3.tar
   ```

3. Create a bootstrap cluster:

   ```
   dkp create bootstrap --kubeconfig $HOME/.kube/config
   ```

4. (Optional) Refresh the credentials used by the AWS provider at any time, using the command:

   ```
   dkp update bootstrap credentials aws
   ```

Then, you can create a cluster(see page 157).

6.2.9.5 Create a New AWS Air-gapped Cluster

Create a new AWS Kubernetes cluster in an Air-gapped environment.

Create a new AWS Kubernetes cluster in an existing infrastructure

When you use existing infrastructure, DKP does *not* create, modify, or delete the following AWS resources:

- Internet Gateways
- NAT Gateways
- Routing tables
- Subnets
- VPC

- VPC Endpoints (for subnets without NAT Gateways)

An AWS subnet has Network ACLs that can control traffic in and out of the subnet. DKP does not modify the Network ACLs of an existing subnet. DKP uses Security Groups to control traffic. If a Network ACL denies traffic that is allowed by DKP-managed Security Groups, the cluster may not work correctly.

1. Set the environment variable to the name you assigned this cluster:

```
export CLUSTER_NAME=aws-example
```

See Get Started with AWS(see page 37) for information on naming your cluster.

NOTE: The cluster name may only contain the following characters: `a-z`, `0-9`, `.`, and `-`. Cluster creation will fail if the name has capital letters. See Kubernetes[125] for more naming information.

2. Export variables for the existing infrastructure details:

```
export AWS_VPC_ID=<vpc-...>
export AWS_SUBNET_IDS=<subnet-...,subnet-...,subnet-...>
export AWS_ADDITIONAL_SECURITY_GROUPS=<sg-...,sg-...>
export AWS_AMI_ID=<ami-...>
```

   - `AWS_VPC_ID`: the VPC ID where the cluster will be created. The VPC requires the `ec2`, `elasticloadbalancing`, `secretsmanager` and `autoscaling` VPC endpoints to be already present.
   - `AWS_SUBNET_IDS`: a comma-separated list of one or more private Subnet IDs with each one in a different Availability Zone. The cluster control-plane and worker nodes will automatically be spread across these Subnets.
   - `AWS_ADDITIONAL_SECURITY_GROUPS`: a comma-seperated list of one or more Security Groups IDs to use in addition to the ones automatically created by CAPA[126].
   - `AWS_AMI_ID`: the AMI ID to use for control-plane and worker nodes. The AMI must be created by the konvoy-image-builder(see page 322).

IMPORTANT: You must tag the subnets as described below to allow for Kubernetes to create ELBs for services of type `LoadBalancer` in those subnets. If the subnets are not tagged, they will not receive an ELB and the following error displays: `Error syncing load balancer, failed to ensure load balancer; could not find any suitable subnets for creating the ELB.`. The tags should be set as follows, where `<CLUSTER_NAME>` corresponds to the name set in `CLUSTER_NAME` environment variable:

```
kubernetes.io/cluster = <CLUSTER_NAME>
kubernetes.io/cluster/<CLUSTER_NAME> = owned
kubernetes.io/role/internal-elb = 1
```

125 https://kubernetes.io/docs/concepts/overview/working-with-objects/names/
126 https://github.com/kubernetes-sigs/cluster-api-provider-aws

3. Configure your cluster to use an existing Docker registry as a mirror when attempting to pull images:
IMPORTANT: The AMI must be created by the konvoy-image-builder[127] project in order to use the registry mirror feature.

```
export DOCKER_REGISTRY_URL=<https/http>://<registry-address>:<registry-port>
export DOCKER_REGISTRY_CA=<path to the CA on the bastion>
export DOCKER_REGISTRY_USERNAME=<username>
export DOCKER_REGISTRY_PASSWORD=<password>
```

   - `DOCKER_REGISTRY_URL` : the address of an existing Docker registry accessible in the VPC that the new cluster nodes will be configured to use a mirror registry when pulling images.
   - `DOCKER_REGISTRY_CA` : (optional) the path on the bastion machine to the Docker registry CA. `dkp` will configure the cluster nodes to trust this CA. This value is only needed if the registry is using a self-signed certificate and the AMIs are not already configured to trust this CA.
   - `DOCKER_REGISTRY_USERNAME` : optional, set to a user that has pull access to this registry.
   - `DOCKER_REGISTRY_PASSWORD` : optional if username is not set.
4. Create a Kubernetes cluster:

```
dkp create cluster aws --cluster-name=${CLUSTER_NAME} \
--vpc-id=${AWS_VPC_ID} \
--ami=${AWS_AMI_ID} \
--subnet-ids=${AWS_SUBNET_IDS} \
--internal-load-balancer=true \
--additional-security-group-ids=${AWS_ADDITIONAL_SECURITY_GROUPS} \
--registry-mirror-url=${DOCKER_REGISTRY_URL} \
--registry-mirror-cacert=${DOCKER_REGISTRY_CA} \
--registry-mirror-username=${DOCKER_REGISTRY_USERNAME} \
--registry-mirror-password=${DOCKER_REGISTRY_PASSWORD}
```

5. (Optional) The Control Plane and Worker nodes can be configured to use an HTTP proxy:

```
export CONTROL_PLANE_HTTP_PROXY=http://example.org:8080
export CONTROL_PLANE_HTTPS_PROXY=http://example.org:8080
export
CONTROL_PLANE_NO_PROXY=
"example.org,example.com,example.net,localhost,127.0.0.1,10.96.0.0/12,192.168.0
.0/16,kubernetes,kubernetes.default,kubernetes.default.svc,kubernetes.default.s
vc.cluster,kubernetes.default.svc.cluster.local,.svc,.svc.cluster,.svc.cluster.
local,169.254.169.254,.elb.amazonaws.com"

export WORKER_HTTP_PROXY=http://example.org:8080
export WORKER_HTTPS_PROXY=http://example.org:8080
export
WORKER_NO_PROXY=
"example.org,example.com,example.net,localhost,127.0.0.1,10.96.0.0/12,192.168.0
.0/16,kubernetes,kubernetes.default,kubernetes.default.svc,kubernetes.default.s
```

127 https://github.com/mesosphere/konvoy-image-builder

```
vc.cluster,kubernetes.default.svc.cluster.local,.svc,.svc.cluster,.svc.cluster.
local,169.254.169.254,.elb.amazonaws.com"
```

- Replace `example.org,example.com,example.net` with you internal addresses
- `localhost` and `127.0.0.1` addresses should not use the proxy
- `10.96.0.0/12` is the default Kubernetes service subnet
- `192.168.0.0/16` is the default Kubernetes pod subnet
- `kubernetes,kubernetes.default,kubernetes.default.svc,kubernetes.defau lt.svc.cluster,kubernetes.default.svc.cluster.local` is the internal Kubernetes kube-apiserver service
- `.svc,.svc.cluster,.svc.cluster.local` are the internal Kubernetes services
- `169.254.169.254` is the AWS metadata server
- `.elb.amazonaws.com` is for the worker nodes to allow them to communicate directly to the kube-apiserver ELB

6. (Optional) Create a Kubernetes cluster with HTTP proxy configured. This step assumes you did not already create a cluster in the previous steps:

```
dkp create cluster aws --cluster-name=${CLUSTER_NAME} \
--vpc-id=${AWS_VPC_ID} \
--ami=${AWS_AMI_ID} \
--subnet-ids=${AWS_SUBNET_IDS} \
--internal-load-balancer=true \
--additional-security-group-ids=${AWS_ADDITIONAL_SECURITY_GROUPS} \
--registry-mirror-url=${DOCKER_REGISTRY_URL} \
--registry-mirror-cacert=${DOCKER_REGISTRY_CA} \
--registry-mirror-username=${DOCKER_REGISTRY_USERNAME} \
--registry-mirror-password=${DOCKER_REGISTRY_PASSWORD}
--control-plane-http-proxy="${CONTROL_PLANE_HTTP_PROXY}" \
--control-plane-https-proxy="${CONTROL_PLANE_HTTPS_PROXY}" \
--control-plane-no-proxy="${CONTROL_PLANE_NO_PROXY}" \
--worker-http-proxy="${WORKER_HTTP_PROXY}" \
--worker-https-proxy="${WORKER_HTTPS_PROXY}" \
--worker-no-proxy="${WORKER_NO_PROXY}"
```

7. Inspect the created cluster resources:

```
kubectl get clusters,kubeadmcontrolplanes,machinedeployments
```

8. Wait for the cluster control-plane to be ready:

```
kubectl wait --for=condition=ControlPlaneReady "clusters/${CLUSTER_NAME}" --
timeout=60m
```

Then, Explore the New Cluster(see page 161).

6.2.9.6 Explore New AWS Air-gapped Cluster

Explore the new Kubernetes cluster

Follow these steps:

1. Fetch the kubeconfig file with the command:

   ```
   dkp get kubeconfig -c ${CLUSTER_NAME} > ${CLUSTER_NAME}.conf
   ```

2. List the Nodes with the command:

   ```
   kubectl --kubeconfig=${CLUSTER_NAME}.conf get nodes
   ```

   Note: wait for the Status to move to `Ready` while `calico-node` pods are being deployed.

3. List the Pods with the command:

   ```
   kubectl --kubeconfig=${CLUSTER_NAME}.conf get pods -A
   ```

When you're ready, delete your cluster and clean up your environment(see page 161).

6.2.9.7 Delete AWS Air-gapped Cluster

Delete the Kubernetes cluster and cleanup your environment

Follow these steps:

1. Delete the provisioned Kubernetes cluster and wait a few minutes:

   ```
   dkp delete cluster --cluster-name=${CLUSTER_NAME}
   ```

2. Delete the `kind` Kubernetes cluster:

   ```
   dkp delete bootstrap
   ```

6.2.9.8 Make Air-gapped AWS Cluster Self-Managed

Konvoy deploys all cluster lifecycle services to a bootstrap cluster, which then deploys a workload cluster. When the workload cluster is ready, move the cluster lifecycle services to the workload cluster, which makes the workload cluster self-managed. This section describes how to make a workload cluster self-managed.

Before starting, ensure you create a workload cluster as described in Create a New Cluster(see page 157).

Make the new Kubernetes cluster manage itself

1. Deploy cluster lifecycle services on the workload cluster:
   By default, `create bootstrap controllers` configures the Cluster API controllers to use the AWS credentials from your environment. We recommend you use the `--with-aws-bootstrap-credentials=false` flag to configure the Cluster API controllers of your self-managed AWS cluster to use AWS IAM Instance Profiles, instead of the AWS credentials from your environment.

   ```
   dkp create capi-components --with-aws-bootstrap-credentials=false --kubeconfig
   ${CLUSTER_NAME}.conf
   ```

   ```
   ✓ Initializing new CAPI components
   ```

2. Move the Cluster API objects from the bootstrap to the workload cluster:
   The cluster lifecycle services on the workload cluster are ready, but the workload cluster configuration is on the bootstrap cluster. The `move` command moves the configuration, which takes the form of Cluster API Custom Resource objects, from the bootstrap to the workload cluster. This process is also called a Pivot[128].

   ```
   dkp move capi-resources --to-kubeconfig ${CLUSTER_NAME}.conf
   ```

   ```
    ✓ Moving cluster resources
   You can now view resources in the moved cluster by using the --kubeconfig flag
   with kubectl. For example: kubectl --kubeconfig=aws-example.conf get nodes
   ```

   NOTE: To ensure only one set of cluster lifecycle services manages the workload cluster, Konvoy first pauses reconciliation of the objects on the bootstrap cluster, then creates the objects on the workload cluster. As Konvoy copies the objects, the cluster lifecycle services on the workload cluster reconcile the objects. The workload cluster becomes self-managed after Konvoy creates all the objects. If it fails, the `move` command can be safely retried.
3. Wait for the cluster control-plane to be ready:

   ```
   kubectl --kubeconfig ${CLUSTER_NAME}.conf wait --for=condition=ControlPlaneRead
   y "clusters/${CLUSTER_NAME}" --timeout=20m
   ```

   ```
   cluster.cluster.x-k8s.io/aws-example condition met
   ```

128 https://cluster-api.sigs.k8s.io/reference/glossary.html?highlight=pivot#pivot

4. Use the cluster lifecycle services on the workload cluster to check the workload cluster status:

   NOTE: After moving the cluster lifecycle services to the workload cluster, remember to use Konvoy with the workload cluster kubeconfig.

```
dkp describe cluster --kubeconfig ${CLUSTER_NAME}.conf -c ${CLUSTER_NAME}
```

```
NAME                                                          READY
SEVERITY  REASON  SINCE  MESSAGE
Cluster/aws-example                                          True
109s
├──ClusterInfrastructure - AWSCluster/aws-example             True
112s
├──ControlPlane - KubeadmControlPlane/aws-example-control-plane  True
109s
│  ├──Machine/aws-example-control-plane-55jh4                 True
111s
│  ├──Machine/aws-example-control-plane-6sn97                 True
111s
│  └──Machine/aws-example-control-plane-nx9v5                 True
110s
└──Workers
   └──MachineDeployment/aws-example-md-0                      True
114s
     ├──Machine/aws-example-md-0-cb9c9bbf7-hcl8z              True
111s
     ├──Machine/aws-example-md-0-cb9c9bbf7-rtdqw              True
111s
     ├──Machine/aws-example-md-0-cb9c9bbf7-t894m              True
111s
     └──Machine/aws-example-md-0-cb9c9bbf7-td29r              True
111s
```

5. Remove the bootstrap cluster, as the workload cluster is now self-managed:

```
dkp delete bootstrap
```

```
✓ Deleting bootstrap cluster
```

Install and Log in to the DKP UI

You can now proceed to installing the DKP UI[129] and applications. After installation, you will be able to log in(see page 385) to the DKP UI to explore it.

DKP Install Configuration

You can configure[130] the Kommander component of DKP during the initial installation, and also post-installation using the Kommander CLI.

The following explains how to run DKP on top of an air-gapped DKP cluster(see page 157) installation with catalog applications.

Depending on your configuration, there are three different ways you can install DKP to an air-gapped environment.
Ensure you follow the correct procedure for your configuration type, and ignore the other two sections that do not pertain to your environment:

- Install air-gapped Kommander with DKP Catalog Applications[131]
- Install air-gapped Kommander with DKP Insights[132]
- Install air-gapped Kommander with DKP Insights and DKP Catalog Applications[133]

Known Limitations

NOTE: Be aware of these limitations in the current release of Konvoy.

- Before making a workload cluster self-managed, be sure that its control plane nodes have sufficient permissions for running Cluster API controllers. See IAM Policy Configuration[134].
- Konvoy supports moving only one set of cluster objects from the bootstrap cluster to the workload cluster, or vice-versa.
- Konvoy only supports moving all namespaces in the cluster; Konvoy does not support migration of individual namespaces.

129 https://d2iq.atlassian.net/wiki/spaces/DENT/pages/29892161/Kommander+Customizations
130 https://d2iq.atlassian.net/wiki/spaces/DENT/pages/29892161/Kommander+Customizations
131 https://d2iq.atlassian.net/wiki/spaces/DENT/pages/29924336/Install+in+an+Air-gapped+Environment+with+Catalog+Applications
132 https://d2iq.atlassian.net/wiki/spaces/DENT/pages/29924336/Install+in+an+Air-gapped+Environment+with+Catalog+Applications
133 https://d2iq.atlassian.net/wiki/spaces/DENT/pages/29924336/Install+in+an+Air-gapped+Environment+with+Catalog+Applications
134 https://d2iq.atlassian.net/wiki/spaces/DENT/pages/29559226/Cluster+IAM+Policies%2C+Roles%2C+and+Artifacts

6.2.10 GPUs in an AWS environment

6.2.10.1 Understanding GPUs

Before working with GPUs, ensure you are familiar with the following:

- Install GPU support on supported distributions on AWS
- GPU node labeling specifications(see page 166)

Kommander also accesses GPU[135] resources.

GPUs in an air-gapped on-premises environment are not supported currently.

6.2.10.2 Install GPU support on supported distributions on AWS

Using the Konvoy Image Builder(see page 322), you can build an image that has support to use Nvidia GPU hardware to support GPU workloads. DKP supported Nvidia driver[136] versions are 470.x and 460.x.

1. In your `overrides/nvidia.yaml` file add the following to enable GPU builds. You can also access and use the overrides repo.[137]

   ```
   gpu:
     type: nvidia
   ```

2. Build your image using the following Konvoy image builder commands:

   ```
   konvoy-image build --region us-west-2 --source-ami=ami-12345abcdef images/ami/
   centos-7.yaml --overrides overrides/nvidia.yaml
   ```

   By default, your image builds in the `us-west-2` region. To specify another region, set the `--region` flag:

   ```
   konvoy-image build --region us-east-1 --overrides override-source-ami.yaml
   images/ami/<Your OS>.yaml
   ```

NOTE: Ensure that an AMI file is available for your OS selection.

1. When the command is complete the `ami` id is printed and written to `./manifest.json`.

   To use the built `ami` with Konvoy, specify it with the `--ami` flag when calling cluster create.

135 https://d2iq.atlassian.net/wiki/spaces/DENT/pages/10420227/GPUs#Kommander-GPU-Overview
136 https://www.nvidia.com/Download/Find.aspx
137 https://github.com/mesosphere/konvoy-image-builder/tree/main/overrides

```
dkp create cluster aws --cluster-name=$(whoami)-aws-cluster --region us-west-2
 --ami <ami>
```

More helpful information can be found in the Nvidia Device Plug-in[138] for Kubernetes instructions and the Installation Guide of Supported Platforms[139].

See also: Nvidia documentation[140]

6.2.10.3 Configure Konvoy Automatic GPU Node Labels

When using GPU nodes, it is important they have the proper label identifying them as Nvidia GPU nodes. Node feature discovery (NFD), by default labels PCI hardware as:

```
"feature.node.kubernetes.io/pci-<device label>.present": "true"
```

where `<device label>` is by default as defined in this topic[141]:

```
< class > _ < vendor >
```

However, due to the wide variety in devices and their assigned PCI classes, you may find that the labels assigned to your GPU nodes do not always properly identify them as containing an Nvidia GPU.

If the default detection does not work, you can manually change the configmap “nvidia-feature-discovery.yaml” before creating the cluster and change the lines:

```
          nodeSelector:
            feature.node.kubernetes.io/pci-< class > _ < vendor>.present: "true"
```

where `class` is any 4 digit number starting with `03xy` and the vendor for Nvidia is `10de`. If this is already deployed, you can always change the `daemonset` and change the `nodeSelector` field so that it will deploy to the right nodes.

6.2.11 Manage AWS Node Pools

Node pools are part of a cluster and managed as a group, and you can use a node pool to manage a group of machines using the same common properties. When Konvoy creates a new default cluster, there is one node pool for the worker nodes and all nodes in that new node pool have the same configuration. You can create additional node pools for more specialized hardware or configuration. For example, if you want to tune your memory usage on a cluster where you need maximum memory for some machines and minimal memory on other machines, you would create a new node pool with those specific resource needs.

---

138 https://github.com/NVIDIA/k8s-device-plugin/blob/master/README.md
139 https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/install-guide.html
140 https://nvidia.custhelp.com/app/answers/detail/a_id/131/kw/driver%20installation%20docs/related/1
141 https://kubernetes-sigs.github.io/node-feature-discovery/v0.7/get-started/features.html#pci

DKP implements node pools using Cluster API MachineDeployments[142]. For more information on node pools, see these sections:

- Create AWS Node Pools(see page 167)
- List AWS Node Pools(see page 168)
- Scale AWS Node Pools(see page 168)
- Delete AWS Node Pools(see page 170)
- AWS Cluster Autoscaler(see page 171)

6.2.11.1 Create AWS Node Pools

Creating a node pool is useful when you need to run workloads that require machines with specific resources, such as a GPU, additional memory, or specialized network or storage hardware.

Prepare the environment

1. Set the environment variable to the name you assigned this cluster.

   ```
   export CLUSTER_NAME=aws-example
   ```

   See Get Started with AWS(see page 37) for information on naming your cluster.
2. If your workload cluster is self-managed, as described in Make the New Cluster Self-Managed(see page 129), configure `kubectl` to use the kubeconfig for the cluster.

   ```
   export KUBECONFIG=${CLUSTER_NAME}.conf
   ```

3. Define your node pool name.

   ```
   export NODEPOOL_NAME=example
   ```

Create an AWS node pool

Availability zones (AZs) are isolated locations within data center regions from which public cloud services originate and operate. Because all the nodes in a node pool are deployed in a single Availability Zone, you may wish to create additional node pools is to ensure your cluster has nodes deployed in multiple Availability Zones.

By default, the first Availability Zone in the region will be used for the Nodes in the node pool, set `--availability-zone` to create the Nodes in a different Availability Zone.

Create a new AWS node pool with 3 replicas using this command:

```
dkp create nodepool aws ${NODEPOOL_NAME} \
```

142 https://cluster-api.sigs.k8s.io/developer/architecture/controllers/machine-deployment.html

```
    --cluster-name=${CLUSTER_NAME} \
    --replicas=3
```

This example uses default values for brevity. Use flags to define custom instance types, AMIs, and other properties.

Advanced users can use a combination of the `--dry-run` and `--output=yaml` flags to get a complete set of node pool objects to modify locally or store in version control.

6.2.11.2 List AWS Node Pools

List node pools for a cluster

Use this command to list the node pools of a given cluster. This returns specific properties of each node pool so that you can see the name of the MachineDeployments.

To list all node pools for a managed cluster, run:

```
dkp get nodepools --cluster-name=${CLUSTER_NAME} --kubeconfig=${CLUSTER_NAME}.conf
```

The expected output is similar to the following example, indicating the desired size of the node pool, the number of replicas ready in the node pool, and the Kubernetes version those nodes are running:

```
NODEPOOL                        DESIRED             READY
KUBERNETES VERSION
example                         3                   3                   v1.24.6
aws-example-md-0                4                   4                   v1.24.6
```

6.2.11.3 Scale AWS Node Pools

Scale node pools in a cluster

While you can run Cluster Autoscaler(see page 171), you can also manually scale your node pools up or down when you need more finite control over your environment. For example, if you require 10 machines to run a process, you can manually set the scaling to run those 10 machines only. However, if also using the Cluster Autoscaler, you must stay within your minimum and maximum bounds.

Scaling Up Node Pools

To scale up a node pool in a cluster, run:

```
dkp scale nodepools ${NODEPOOL_NAME} --replicas=5 --cluster-name=${CLUSTER_NAME}
```

Your output should be similar to this example, indicating the scaling is in progress:

```
✓ Scaling node pool example to 5 replicas
```

After a few minutes, you can list the node pools to:

```
dkp get nodepools --cluster-name=${CLUSTER_NAME} --kubeconfig=${CLUSTER_NAME}.conf
```

Your output should be similar to this example, with the number of DESIRED and READY replicas increased to 5:

```
NODEPOOL                        DESIRED             READY
KUBERNETES VERSION
example                         5                   5                   v1.24.6
aws-example-md-0                4                   4                   v1.24.6
```

Scaling Down Node Pools

To scale down a node pool, run:

```
dkp scale nodepools ${NODEPOOL_NAME} --replicas=4 --cluster-name=${CLUSTER_NAME}
```

```
✓ Scaling node pool example to 4 replicas
```

After a few minutes, you can list the node pools using this command:

```
dkp get nodepools --cluster-name=${CLUSTER_NAME} --kubeconfig=${CLUSTER_NAME}.conf
```

Your output should be similar to this example, with the number of DESIRED and READY replicas decreased to 4:

```
NODEPOOL                        DESIRED             READY
KUBERNETES VERSION
example                         4                   4                   v1.24.6
aws-example-md-0                4                   4                   v1.24.6
```

In a default cluster, the nodes to delete are selected at random. This behavior is controlled by CAPI's delete policy[143]. However, when using the DKP CLI to scale down a node pool, it is also possible to specify the Kubernetes Nodes you want to delete.

To do this, set the flag `--nodes-to-delete` with a list of nodes as below. This adds an annotation `cluster.x-k8s.io/delete-machine=yes` to the matching Machine object that contains `status.NodeRef` with the node names from `--nodes-to-delete`.

```
dkp scale nodepools ${NODEPOOL_NAME} --replicas=3 --nodes-to-delete=<> --cluster-
name=${CLUSTER_NAME}
```

143 https://github.com/kubernetes-sigs/cluster-api/blob/v0.4.0/api/v1alpha4/machineset_types.go#L85-L105

```
✓ Scaling node pool example to 3 replicas
```

Scaling Node Pools When Using Cluster Autoscaler

If you configured the cluster autoscaler(see page 171) for the `demo-cluster-md-0` node pool, the value of `--replicas` must be within the minimum and maximum bounds.

For example, assuming you have these annotations:

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf annotate machinedeployment ${NODEPOOL_NAME}
cluster.x-k8s.io/cluster-api-autoscaler-node-group-min-size=2
kubectl --kubeconfig=${CLUSTER_NAME}.conf annotate machinedeployment ${NODEPOOL_NAME}
cluster.x-k8s.io/cluster-api-autoscaler-node-group-max-size=6
```

Try to scale the node pool to 7 replicas with the command:

```
dkp scale nodepools ${NODEPOOL_NAME} --replicas=7 -c demo-cluster
```

Which results in an error similar to:

```
 ✗ Scaling node pool example to 7 replicas
failed to scale nodepool: scaling MachineDeployment is forbidden: desired replicas 7
 is greater than the configured max size annotation cluster.x-k8s.io/cluster-api-
autoscaler-node-group-max-size: 6
```

Similarly, scaling down to a number of replicas less than the configured `min-size` also returns an error.

6.2.11.4 Delete AWS Node Pools

Delete node pools in a cluster

Deleting a node pool deletes the Kubernetes nodes and the underlying infrastructure. All nodes will be drained prior to deletion and the pods running on those nodes will be rescheduled.

To delete a node pool from a managed cluster, run:

```
dkp delete nodepool ${NODEPOOL_NAME} --cluster-name=${CLUSTER_NAME}
```

Here, `example` is the node pool to be deleted.

The expected output will be similar to the following example, indicating the node pool is being deleted:

```
✓ Deleting default/example nodepool resources
```

Deleting an invalid node pool results in output similar to this example:

```
dkp delete nodepool ${CLUSTER_NAME}-md-invalid --cluster-name=${CLUSTER_NAME}
```

```
MachineDeployments or MachinePools.infrastructure.cluster.x-k8s.io "no
MachineDeployments or MachinePools found for cluster aws-example" not found
```

6.2.11.5 AWS Cluster Autoscaler

Configure autoscaler for node pools

Cluster Autoscaler[144] provides the ability to automatically scale-up or scale-down the number of worker nodes in a cluster, based on the number of pending pods to be scheduled. Running the Cluster Autoscaler is optional.

Unlike Horizontal-Pod Autoscaler[145], Cluster Autoscaler does not depend on any Metrics server and does not need Prometheus or any other metrics source.

The Cluster Autoscaler looks at the following annotations on a MachineDeployment to determine its scale-up and scale-down ranges:

```
cluster.x-k8s.io/cluster-api-autoscaler-node-group-min-size
cluster.x-k8s.io/cluster-api-autoscaler-node-group-max-size
```

The full list of command line arguments to the Cluster Autoscaler controller is on the Kubernetes public GitHub repository[146].

For more information about how Cluster Autoscaler works, see these documents:

- What is Cluster Autoscaler[147]
- How does scale-up work[148]
- How does scale-down work[149]
- CAPI Provider for Cluster Autoscaler[150]

Cluster Autoscaler Prerequisites

Before you begin, you must have:

- A Bootstrap Cluster Lifecycle(see page 115).
- Created a new Kubernetes Cluster(see page 117).
- A Self-Managed Cluster(see page 129).

144 https://github.com/kubernetes/autoscaler/tree/master/cluster-autoscaler/cloudprovider/clusterapi
145 https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/FAQ.md#how-fast-is-hpa-when-combined-with-ca
146 https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/FAQ.md#what-are-the-parameters-to-ca
147 https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/FAQ.md#what-is-cluster-autoscaler
148 https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/FAQ.md#how-does-scale-up-work
149 https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/FAQ.md#how-does-scale-down-work
150 https://github.com/kubernetes/autoscaler/tree/master/cluster-autoscaler/cloudprovider/clusterapi

Run Cluster Autoscaler

The Cluster Autoscaler controller runs on the workload cluster. Upon creation of the workload cluster, this controller does not have all the objects required to function correctly until after a `dkp move` is issued from the bootstrap cluster.

Run the following steps to enable Cluster Autoscaler:

1. Ensure the Cluster Autoscaler controller is up and running (no restarts and no errors in the logs)

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf logs deployments/cluster-autoscaler
cluster-autoscaler -n kube-system -f
```

2. Enable Cluster Autoscaler by setting the min & max ranges

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf annotate machinedeployment $
{NODEPOOL_NAME} cluster.x-k8s.io/cluster-api-autoscaler-node-group-min-size=2
kubectl --kubeconfig=${CLUSTER_NAME}.conf annotate machinedeployment $
{NODEPOOL_NAME} cluster.x-k8s.io/cluster-api-autoscaler-node-group-max-size=6
```

3. The Cluster Autoscaler logs will show that the worker nodes are associated with node-groups and that pending pods are being watched.
4. To demonstrate that it is working properly, create a large deployment which will trigger pending pods (For this example we used AWS m5.2xlarge worker nodes. If you have larger worker-nodes, you should scale up the number of replicas accordingly).

```
cat <<EOF | kubectl --kubeconfig=${CLUSTER_NAME}.conf apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: busybox-deployment
  labels:
    app: busybox
spec:
  replicas: 600
  selector:
    matchLabels:
      app: busybox
  template:
    metadata:
      labels:
        app: busybox
    spec:
      containers:
      - name: busybox
        image: busybox:latest
        command:
          - sleep
          - "3600"
        imagePullPolicy: IfNotPresent
```

```
      restartPolicy: Always
EOF
```

5. Cluster Autoscaler will scale up the number of Worker Nodes until there are no pending pods.
6. Scale down the number of replicas for `busybox-deployment`.

```
kubectl --kubeconfig ${CLUSTER_NAME}.conf scale --replicas=30 deployment/
busybox-deployment
```

7. Cluster Autoscaler starts to scale down the number of Worker Nodes after the default timeout of 10 minutes.

6.3 Azure Infrastructure

This section describes how you create a DKP cluster in Azure.

6.3.1 Azure Prerequisites

Prepare your machine and environment to run DKP

6.3.1.1 DKP Prerequisites

Before you begin using DKP you must have:

- An x86_64-based Linux or macOS machine.
- The `dkp` binary for Linux, or macOS.
- Docker[151] version 18.09.2 or later installed.
- kubectl[152] for interacting with the running cluster.
- Azure CLI[153].
- A valid Azure account with credentials configured[154].

On macOS, Docker runs in a virtual machine. Configure this virtual machine with at least 8GB of memory.

151 https://docs.docker.com/get-docker/
152 https://kubernetes.io/docs/tasks/tools/#kubectl
153 https://docs.microsoft.com/en-us/cli/azure/install-azure-cli
154 https://github.com/kubernetes-sigs/cluster-api-provider-azure/blob/master/docs/book/src/topics/getting-started.md#prerequisites

Control plane nodes

You must have at least three control plane nodes. Each control plane node should have at least:

- 4 cores
- 16 GiB memory
- Approximately 80 GiB of free space for the volume used for /var/lib/kubelet and /var/lib/containerd.
- Disk usage must be below 85% on the root volume.

DKP on Azure defaults to deploying a `Standard_D4s_v3` virtual machine with an 128 GiB volume for the OS and an 80GiB volume for etcd storage, which meets the above requirements.

Worker nodes

You must have at least four worker nodes. The specific number of worker nodes required for your environment can vary depending on the cluster workload and size of the nodes. Each worker node should have at least:

- 8 cores
- 32 GiB memory
- Around 80 GiB of free space for the volume used for /var/lib/kubelet and /var/lib/containerd.
- Disk usage must be below 85% on the root volume.

DKP on Azure defaults to deploying a `Standard_D8s_v3` virtual machine with an 80 GiB volume for the OS, which meets the above requirements.

If you use these instructions to create a cluster on Azure using the DKP default settings without any edits to configuration files or additional flags, your cluster is deployed on an Ubuntu 20.04 operating system image(see page 78) with 3 control plane nodes, and 4 worker nodes which match the requirements above.

6.3.1.2 Azure Prerequisites

Before you begin using Konvoy with Azure, you must:

1. Log in to Azure:

```
az login
```


```
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "a1234567-b132-1234-1a11-1234a5678b90",
    "id": "b1234567-abcd-11a1-a0a0-1234a5678b90",
    "isDefault": true,
    "managedByTenants": [],
    "name": "Mesosphere Developer Subscription",
    "state": "Enabled",
    "tenantId": "a1234567-b132-1234-1a11-1234a5678b90",
    "user": {
      "name": "user@azuremesosphere.onmicrosoft.com",
      "type": "user"
    }
```

```
  }
]
```

2. Create an Azure Service Principal (SP) by running the following command:

If an SP with the name exists, this command will rotate the password.

```
az ad sp create-for-rbac --role contributor --name "$(whoami)-konvoy" --scopes=/
subscriptions/$(az account show --query id -o tsv)
```

```
{
  "appId": "7654321a-1a23-567b-b789-0987b6543a21",
  "displayName": "azure-cli-2021-03-09-23-17-06",
  "password": "Z79yVstq_E.R0R7RUUck718vEHSuyhAB0C",
  "tenant": "a1234567-b132-1234-1a11-1234a5678b90"
}
```

3. Set the required environment variables:

```
export AZURE_SUBSCRIPTION_ID="<id>"       # b1234567-abcd-11a1-a0a0-1234a5678b90
export AZURE_TENANT_ID="<tenant>"         # a1234567-b132-1234-1a11-1234a5678b90
export AZURE_CLIENT_ID="<appId>"          # 7654321a-1a23-567b-b789-0987b6543a21
export AZURE_CLIENT_SECRET="<password>"   # Z79yVstq_E.R0R7RUUck718vEHSuyhAB0C
```

4. Base64 encode the same environment variables:

```
export AZURE_SUBSCRIPTION_ID_B64="$(echo -n "${AZURE_SUBSCRIPTION_ID}" | base64 | tr
-d '\n')"
export AZURE_TENANT_ID_B64="$(echo -n "${AZURE_TENANT_ID}" | base64 | tr -d '\n')"
export AZURE_CLIENT_ID_B64="$(echo -n "${AZURE_CLIENT_ID}" | base64 | tr -d '\n')"
export AZURE_CLIENT_SECRET_B64="$(echo -n "${AZURE_CLIENT_SECRET}" | base64 | tr -d
'\n')"
```

When you completed, move on to the Bootstrap(see page 175) section.

6.3.2 Azure Bootstrap

6.3.2.1 Prepare to deploy Kubernetes clusters

To create Kubernetes clusters, Konvoy uses Cluster API[155] (CAPI) controllers. These controllers run on a Kubernetes cluster. To get started, you need a *bootstrap* cluster. By default, Konvoy creates a bootstrap cluster for you in a Docker container using the Kubernetes-in-Docker (KIND[156]) tool.

155 https://cluster-api.sigs.k8s.io/
156 https://github.com/kubernetes-sigs/kind

6.3.2.2 Prerequisites

Before you begin, you must:

- Complete the steps in Prerequisites(see page 173).
- Ensure the `dkp` binary can be found in your $PATH.

6.3.2.3 Bootstrap Cluster Lifecycle Services

1. If an HTTP proxy is required for the bootstrap cluster, set the local `http_proxy`, `https_proxy`, and `no_proxy` environment variables. They are copied into the bootstrap cluster.
2. Create a bootstrap cluster:

```
dkp create bootstrap --kubeconfig $HOME/.kube/config
```

```
✓ Creating a bootstrap cluster
✓ Initializing new CAPI components
```

Konvoy creates a bootstrap cluster using KIND[157] as a library. Konvoy then deploys the following Cluster API[158] providers on the cluster:

- Core Provider[159]
- Azure Infrastructure Provider[160]
- kubeadm Bootstrap Provider[161]
- kubeadm ControlPlane Provider[162]

Konvoy waits until the controller-manager and webhook deployments of these providers are ready. List these deployments using this command:

```
kubectl get --all-namespaces deployments -l=clusterctl.cluster.x-k8s.io
```

```
NAMESPACE                           NAME
READY   UP-TO-DATE   AVAILABLE   AGE
capa-system                         capa-controller-manager
1/1     1            1           69s
capi-kubeadm-bootstrap-system       capi-kubeadm-bootstrap-controller-manager
1/1     1            1           71s
```

157 https://github.com/kubernetes-sigs/kind
158 https://cluster-api.sigs.k8s.io/
159 https://github.com/kubernetes-sigs/cluster-api/tree/v0.3.20/
160 https://github.com/kubernetes-sigs/cluster-api-provider-azure
161 https://github.com/kubernetes-sigs/cluster-api/tree/v0.3.20/bootstrap/kubeadm
162 https://github.com/kubernetes-sigs/cluster-api/tree/v0.3.20/controlplane/kubeadm

```
capi-kubeadm-control-plane-system   capi-kubeadm-control-plane-controller-
manager   1/1     1            1           70s
capi-system                         capi-controller-manager
1/1     1            1           73s
cappp-system                        cappp-controller-manager
1/1     1            1           66s
capv-system                         capv-controller-manager
1/1     1            1           65s
capz-system                         capz-controller-manager
1/1     1            1           67s
cert-manager                        cert-manager
1/1     1            1           16m
cert-manager                        cert-manager-cainjector
1/1     1            1           16m
cert-manager                        cert-manager-webhook
1/1     1            1           16m
```

6.3.2.4 (Optional) Create identity secret for Azure

If your bootstrap cluster resides on a Virtual machine inside Azure, create an identity secret that uses the cappz-controller:

```
export AZURE_CLUSTER_IDENTITY_SECRET_NAME="cluster-identity-secret"
export CLUSTER_IDENTITY_NAME="cluster-identity"
export AZURE_CLUSTER_IDENTITY_SECRET_NAMESPACE="default"

kubectl create secret generic "${AZURE_CLUSTER_IDENTITY_SECRET_NAME}" --from-
literal=clientSecret="${AZURE_CLIENT_SECRET}"
```

When complete, move on to the Create a New Cluster(see page 177) section.

6.3.3 Create a new Azure Cluster

6.3.3.1 Prerequisites

- Before you begin, make sure you have created a Bootstrap(see page 175) cluster.

6.3.3.2 Name your cluster

1. Give your cluster a unique name suitable for your environment.
2. Set the environment variable:

```
export CLUSTER_NAME=azure-example
```

6.3.3.3 Tips and Tricks

Below are a few ways to customize your setup. If you prefer to do a basic setup, skip Tips and Tricks and proceed to Create a New Azure Cluster(see page 0) section.

1. (Optional) To create a cluster name that is unique, use the following command:

```
export CLUSTER_NAME=azure-example-$(LC_CTYPE=C tr -dc 'a-z0-9' </dev/urandom | fold
-w 5 | head -n1)
echo $CLUSTER_NAME
```

```
azure-example-pf4a3
```

This creates a unique name every time you run it, so use it with forethought.

2. (Optional) To use a custom Azure Image when creating your cluster, you must create that Azure Image using KIB(see page 177) first.

```
dkp create cluster azure --cluster-name=${CLUSTER_NAME} \
--dry-run \
--output=yaml \
> ${CLUSTER_NAME}.yaml
```

6.3.3.4 Create a new Azure Kubernetes cluster

Availability zones (AZs) are isolated locations within data center regions from which public cloud services originate and operate. Because all the nodes in a node pool are deployed in a single Availability Zone, you may wish to create additional node pools to ensure your cluster has nodes deployed in multiple Availability Zones.

By default, the control-plane Nodes will be created in 3 different zones. However, the default worker Nodes will reside in a single Availability Zone. You may create additional node pools in other Availability Zones with the `dkp create nodepool` command.

1. Generate the Kubernetes cluster objects:

```
dkp create cluster azure --cluster-name=${CLUSTER_NAME} \
--dry-run \
--output=yaml \
> ${CLUSTER_NAME}.yaml
```

```
Generating cluster resources
```

2. (Optional) To configure the Control Plane and Worker nodes to use an HTTP proxy:

```
export CONTROL_PLANE_HTTP_PROXY=http://example.org:8080
export CONTROL_PLANE_HTTPS_PROXY=http://example.org:8080
export
CONTROL_PLANE_NO_PROXY=
"example.org,example.com,example.net,localhost,127.0.0.1,10.96.0.0/12,192.168.0
.0/16,kubernetes,kubernetes.default,kubernetes.default.svc,kubernetes.default.s
vc.cluster,kubernetes.default.svc.cluster.local,.svc,.svc.cluster,.svc.cluster.
local,169.254.169.254,.cloudapp.azure.com"

export WORKER_HTTP_PROXY=http://example.org:8080
export WORKER_HTTPS_PROXY=http://example.org:8080
export
WORKER_NO_PROXY=
"example.org,example.com,example.net,localhost,127.0.0.1,10.96.0.0/12,192.168.0
.0/16,kubernetes,kubernetes.default,kubernetes.default.svc,kubernetes.default.s
vc.cluster,kubernetes.default.svc.cluster.local,.svc,.svc.cluster,.svc.cluster.
local,169.254.169.254,.cloudapp.azure.com"
```

- Replace `example.org,example.com,example.net` with your internal addresses
- `localhost` and `127.0.0.1` addresses should not use the proxy
- `10.96.0.0/12` is the default Kubernetes service subnet
- `192.168.0.0/16` is the default Kubernetes pod subnet
- `kubernetes,kubernetes.default,kubernetes.default.svc,kubernetes.default.svc.cluster,kubernetes.default.svc.cluster.local` is the internal Kubernetes kube-apiserver service
- `.svc,.svc.cluster,.svc.cluster.local` is the internal Kubernetes services
- `169.254.169.254` is the Azure metadata server
- `.cloudapp.azure.com` is for the worker nodes to allow them to communicate directly to the kube-apiserver load balancer

3. (Optional) Create a Kubernetes cluster with HTTP proxy configured. This step assumes you did not already create a cluster in the previous steps:

```
dkp create cluster azure --cluster-name=${CLUSTER_NAME} \
--control-plane-http-proxy="${CONTROL_PLANE_HTTP_PROXY}" \
--control-plane-https-proxy="${CONTROL_PLANE_HTTPS_PROXY}" \
--control-plane-no-proxy="${CONTROL_PLANE_NO_PROXY}" \
--worker-http-proxy="${WORKER_HTTP_PROXY}" \
--worker-https-proxy="${WORKER_HTTPS_PROXY}" \
--worker-no-proxy="${WORKER_NO_PROXY}" \
--dry-run \
--output=yaml \
> ${CLUSTER_NAME}.yaml
```

4. Inspect or edit the cluster objects:
NOTE: Familiarize yourself with Cluster API before editing the cluster objects as edits can prevent the cluster from deploying successfully.

The objects are Custom Resources[163] defined by Cluster API components, and they belong in three different categories:

a. Cluster
   A *Cluster* object has references to the infrastructure-specific and control plane objects. Because this is an Azure cluster, there is an *AzureCluster* object that describes the infrastructure-specific cluster properties. Here, this means the Azure region, the VPC ID, subnet IDs, and security group rules required by the Pod network implementation.
b. Control Plane
   A *KubeadmControlPlane* object describes the control plane, which is the group of machines that run the Kubernetes control plane components, which include the etcd distributed database, the API server, the core controllers, and the scheduler. The object describes the configuration for these components. The object also has a reference to an infrastructure-specific object that describes the properties of all control plane machines. Here, it references an *AzureMachineTemplate* object, which describes the instance type, the type of disk used, and the size of the disk, among other properties.
c. Node Pool
   A Node Pool is a collection of machines with identical properties. For example, a cluster might have one Node Pool with large memory capacity, another Node Pool with GPU support. Each Node Pool is described by three objects: The MachinePool references an object that describes the configuration of Kubernetes components (for example, kubelet) deployed on each node pool machine, and an infrastructure-specific object that describes the properties of all node pool machines. Here, it references a *KubeadmConfigTemplate*, and an *AzureMachineTemplate* object, which describes the instance type, the type of disk used, the size of the disk, among other properties.

For in-depth documentation about the objects, read Concepts[164] in the Cluster API Book.

5. Modify Control Plane Audit logs settings using the information contained in the page Configuring the Control Plane(see page 343).
6. Create the cluster from the objects.

```
kubectl create -f ${CLUSTER_NAME}.yaml
```

```
cluster.cluster.x-k8s.io/azure-example created
azurecluster.infrastructure.cluster.x-k8s.io/azure-example created
kubeadmcontrolplane.controlplane.cluster.x-k8s.io/azure-example-control-plane
created
azuremachinetemplate.infrastructure.cluster.x-k8s.io/azure-example-control-
plane created
secret/azure-example-etcd-encryption-config created
machinedeployment.cluster.x-k8s.io/azure-example-md-0 created
azuremachinetemplate.infrastructure.cluster.x-k8s.io/azure-example-md-0 created
kubeadmconfigtemplate.bootstrap.cluster.x-k8s.io/azure-example-md-0 created
clusterresourceset.addons.cluster.x-k8s.io/calico-cni-installation-azure-
example created
configmap/calico-cni-installation-azure-example created
configmap/tigera-operator-azure-example created
clusterresourceset.addons.cluster.x-k8s.io/azure-disk-csi-azure-example created
configmap/azure-disk-csi-azure-example created
```

163 https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/
164 https://cluster-api.sigs.k8s.io/user/concepts.html

```
clusterresourceset.addons.cluster.x-k8s.io/cluster-autoscaler-azure-example
created
configmap/cluster-autoscaler-azure-example created
clusterresourceset.addons.cluster.x-k8s.io/node-feature-discovery-azure-example
created
configmap/node-feature-discovery-azure-example created
clusterresourceset.addons.cluster.x-k8s.io/nvidia-feature-discovery-azure-
example created
configmap/nvidia-feature-discovery-azure-example created
```

7. Wait for the cluster control-plane to be ready:

```
kubectl wait --for=condition=ControlPlaneReady "clusters/${CLUSTER_NAME}" --
timeout=20m
```

```
cluster.cluster.x-k8s.io/azure-example condition met
```

8. After the objects are created on the API server, the Cluster API controllers reconcile them. They create infrastructure and machines. As they progress, they update the Status of each object. Konvoy provides a command to describe the current status of the cluster:

```
dkp describe cluster -c ${CLUSTER_NAME}
```

```
NAME                                                              READY
SEVERITY  REASON  SINCE  MESSAGE
Cluster/azure-example                                             True
3m4s
├──ClusterInfrastructure - AzureCluster/azure-example              True
8m26s
├──ControlPlane - KubeadmControlPlane/azure-example-control-plane  True
3m4s
│  ├──Machine/azure-example-control-plane-l8j9r                    True
3m9s
│  ├──Machine/azure-example-control-plane-slprd                    True
7m17s
│  └──Machine/azure-example-control-plane-xhxxg                    True
5m9s
└──Workers
   └──MachineDeployment/azure-example-md-0                         True
4m31s
     ├──Machine/azure-example-md-0-d67567c8b-2674r                 True
5m19s
     ├──Machine/azure-example-md-0-d67567c8b-mbmhk                 True
5m17s
     ├──Machine/azure-example-md-0-d67567c8b-pzg8k                 True
5m17s
```

```
    └──Machine/azure-example-md-0-d67567c8b-z8km9                    True
5m17s
```

9. As they progress, the controllers also create Events. List the Events using this command:

```
kubectl get events | grep ${CLUSTER_NAME}
```

For brevity, the example uses `grep`. It is also possible to use separate commands to get Events for specific objects. For example, `kubectl get events --field-selector involvedObject.kind="AzureCluster"` and `kubectl get events --field-selector involvedObject.kind="AzureMachine"`.

```
15m         Normal    AzureClusterObjectNotFound                  azurecluster
AzureCluster object default/azure-example not found
15m         Normal    AzureManagedControlPlaneObjectNotFound
azuremanagedcontrolplane                                AzureManagedControlPlane
object default/azure-example not found
15m         Normal    AzureClusterObjectNotFound                  azurecluster
AzureCluster.infrastructure.cluster.x-k8s.io "azure-example" not found
8m22s       Normal    SuccessfulSetNodeRef                        machine/
azure-example-control-plane-bmc9b          azure-example-control-plane-fdvnm
10m         Normal    Machine controller dependency not yet met   azuremachine/
azure-example-control-plane-fdvnm     Machine Controller has not yet set
OwnerRef
12m         Normal    SuccessfulSetNodeRef                        machine/
azure-example-control-plane-msftd          azure-example-control-plane-z9q45
10m         Normal    SuccessfulSetNodeRef                        machine/
azure-example-control-plane-nrvff          azure-example-control-plane-vmqwx
12m         Normal    Machine controller dependency not yet met   azuremachine/
azure-example-control-plane-vmqwx     Machine Controller has not yet set
OwnerRef
14m         Normal    Machine controller dependency not yet met   azuremachine/
azure-example-control-plane-z9q45     Machine Controller has not yet set
OwnerRef
14m         Warning   VMIdentityNone
azuremachinetemplate/azure-example-control-plane   You are using Service
Principal authentication for Cloud Provider Azure which is less secure than
Managed Identity. Your Service Principal credentials will be written to a file
on the disk of each VM in order to be accessible by Cloud Provider. To learn
more, see https://capz.sigs.k8s.io/topics/identities-use-cases.html#azure-host-
identity
12m         Warning   ControlPlaneUnhealthy
kubeadmcontrolplane/azure-example-control-plane    Waiting for control plane to
pass preflight checks to continue reconciliation: [machine azure-example-
control-plane-msftd does not have APIServerPodHealthy condition, machine azure-
example-control-plane-msftd does not have ControllerManagerPodHealthy
condition, machine azure-example-control-plane-msftd does not have
SchedulerPodHealthy condition, machine azure-example-control-plane-msftd does
```

```
not have EtcdPodHealthy condition, machine azure-example-control-plane-msftd
does not have EtcdMemberHealthy condition]
11m         Warning   ControlPlaneUnhealthy
kubeadmcontrolplane/azure-example-control-plane    Waiting for control plane to
pass preflight checks to continue reconciliation: [machine azure-example-
control-plane-nrvff does not have APIServerPodHealthy condition, machine azure-
example-control-plane-nrvff does not have ControllerManagerPodHealthy
condition, machine azure-example-control-plane-nrvff does not have
SchedulerPodHealthy condition, machine azure-example-control-plane-nrvff does
not have EtcdPodHealthy condition, machine azure-example-control-plane-nrvff
does not have EtcdMemberHealthy condition]
9m52s       Normal    SuccessfulSetNodeRef                          machine/
azure-example-md-0-84bd8b5f5b-b8cnq        azure-example-md-0-bsc82
9m53s       Normal    SuccessfulSetNodeRef                          machine/
azure-example-md-0-84bd8b5f5b-j8ldg        azure-example-md-0-mjcbn
9m52s       Normal    SuccessfulSetNodeRef                          machine/
azure-example-md-0-84bd8b5f5b-lx89f        azure-example-md-0-pmq8f
10m         Normal    SuccessfulSetNodeRef                          machine/
azure-example-md-0-84bd8b5f5b-pcv7q        azure-example-md-0-vzprf
15m         Normal    SuccessfulCreate                              machineset/
azure-example-md-0-84bd8b5f5b           Created machine "azure-example-
md-0-84bd8b5f5b-j8ldg"
15m         Normal    SuccessfulCreate                              machineset/
azure-example-md-0-84bd8b5f5b           Created machine "azure-example-
md-0-84bd8b5f5b-lx89f"
15m         Normal    SuccessfulCreate                              machineset/
azure-example-md-0-84bd8b5f5b           Created machine "azure-example-
md-0-84bd8b5f5b-pcv7q"
15m         Normal    SuccessfulCreate                              machineset/
azure-example-md-0-84bd8b5f5b           Created machine "azure-example-
md-0-84bd8b5f5b-b8cnq"
15m         Normal    Machine controller dependency not yet met   azuremachine/
azure-example-md-0-bsc82              Machine Controller has not yet set
OwnerRef
15m         Normal    Machine controller dependency not yet met   azuremachine/
azure-example-md-0-mjcbn              Machine Controller has not yet set
OwnerRef
15m         Normal    Machine controller dependency not yet met   azuremachine/
azure-example-md-0-pmq8f              Machine Controller has not yet set
OwnerRef
```

6.3.3.5 Known Limitations

NOTE: Be aware of these limitations in the current release of Konvoy.

- The Konvoy version used to create a bootstrap cluster must match the Konvoy version used to create a workload cluster.
- Konvoy supports deploying one workload cluster.
- Konvoy generates a set of objects for one Node Pool.
- Konvoy does not validate edits to cluster objects.

Next, you can Explore the New Cluster(see page 184).

6.3.4 Explore new Azure Cluster

6.3.4.1 Learn to interact with your Kubernetes cluster

This guide explains how to use the command line to interact with your newly deployed Kubernetes cluster.

Before you start, make sure you have created a workload cluster, as described in Create a New Cluster(see page 177).

6.3.4.2 Explore the new Kubernetes cluster

1. Get a kubeconfig file for the workload cluster:
   When the workload cluster is created, the cluster lifecycle services generate a kubeconfig file for the workload cluster, and write it to a *Secret*. The kubeconfig file is scoped to the cluster administrator.
   Get the kubeconfig from the *Secret*, and write it to a file, using this command:

   ```
   dkp get kubeconfig -c ${CLUSTER_NAME} > ${CLUSTER_NAME}.conf
   ```

2. List the Nodes using this command:

   ```
   kubectl --kubeconfig=${CLUSTER_NAME}.conf get nodes
   ```

   ```
   NAME                                STATUS   ROLES                  AGE
   VERSION
   azure-example-control-plane-7ffnl   Ready    control-plane,master   6m18s
   v1.24.6
   azure-example-control-plane-l4bv8   Ready    control-plane,master   14m
   v1.24.6
   azure-example-control-plane-n4g4l   Ready    control-plane,master   18m
   v1.24.6
   azure-example-md-0-mpctb            Ready    <none>                 15m     v1.2
   4.6
   azure-example-md-0-qglp9            Ready    <none>                 15m     v1.2
   4.6
   azure-example-md-0-sgrd6            Ready    <none>                 16m     v1.2
   4.6
   azure-example-md-0-wzbkl            Ready    <none>                 16m     v1.2
   4.6
   ```

NOTE: It may take a few minutes for the Status to move to `Ready` while the Pod network is deployed. The Nodes' Status should change to Ready soon after the `calico-node` DaemonSet Pods are Ready.

3. List the Pods using this command:

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf get --all-namespaces pods
```

```
NAMESPACE                           NAME
READY   STATUS    RESTARTS      AGE
calico-system                       calico-kube-controllers-57fbd7bd59-v4tss
1/1     Running   0             19m
calico-system                       calico-node-59llv
1/1     Running   0             17m
calico-system                       calico-node-7t7wj
1/1     Running   0             16m
calico-system                       calico-node-pf8q8
1/1     Running   0             17m
calico-system                       calico-node-sh2b7
1/1     Running   0             8m17s
calico-system                       calico-node-tmxl5
1/1     Running   0             19m
calico-system                       calico-node-vt5fh
1/1     Running   0             18m
calico-system                       calico-node-whfs8
1/1     Running   0             18m
calico-system                       calico-typha-797c9666d5-5w99r
1/1     Running   0             19m
calico-system                       calico-typha-797c9666d5-hj6mj
1/1     Running   0             18m
calico-system                       calico-typha-797c9666d5-s7rc6
1/1     Running   0             17m
capa-system                         capa-controller-manager-74fffb5676-ch6xd
1/1     Running   0             11m
capi-kubeadm-bootstrap-system       capi-kubeadm-bootstrap-controller-
manager-867759cc67-vg4lh       1/1     Running   0             15m
capi-kubeadm-control-plane-system   capi-kubeadm-control-plane-controller-
manager-5df55579c4-pc8x9   1/1     Running   1 (11m ago)   15m
capi-system                         capi-controller-manager-79cc58bf5f-xsp9t
1/1     Running   0             15m
cappp-system                        cappp-controller-manager-85b5c77497-8ss8r
1/1     Running   0             14m
capv-system                         capv-controller-manager-7bf4d8b66-6x2mx
1/1     Running   0             14m
capz-system                         capz-controller-manager-5d4c6468bf-wfhcc
1/1     Running   0             14m
capz-system                         capz-nmi-2cbrg
1/1     Running   0             14m
capz-system                         capz-nmi-8dllm
1/1     Running   0             14m
capz-system                         capz-nmi-95dfk
1/1     Running   0             14m
```

```
capz-system                         capz-nmi-rtnd4
1/1     Running   0             14m
cert-manager                        cert-manager-848f547974-gjc5p
1/1     Running   1 (10m ago)   15m
cert-manager                        cert-manager-cainjector-54f4cc6b5-rnh4f
1/1     Running   0             15m
cert-manager                        cert-manager-webhook-7c9588c76-rn2sd
1/1     Running   0             15m
kube-system                         cluster-autoscaler-68c759fbf6-6vg5r
1/1     Running   1 (11m ago)   20m
kube-system                         coredns-78fcd69978-6gx44
1/1     Running   0             20m
kube-system                         coredns-78fcd69978-gr5q7
1/1     Running   0             20m
kube-system                         csi-azuredisk-controller-c8fb44c8b-jhmfz
6/6     Running   5 (11m ago)   20m
kube-system                         csi-azuredisk-controller-c8fb44c8b-lpbbs
6/6     Running   0             20m
kube-system                         csi-azuredisk-node-2g7vw
3/3     Running   0             8m17s
kube-system                         csi-azuredisk-node-6rdqc
3/3     Running   0             18m
kube-system                         csi-azuredisk-node-99c6q
3/3     Running   0             17m
kube-system                         csi-azuredisk-node-9b4ms
3/3     Running   0             17m
kube-system                         csi-azuredisk-node-mz5pr
3/3     Running   0             18m
kube-system                         csi-azuredisk-node-r2t99
3/3     Running   0             16m
kube-system                         csi-azuredisk-node-t7gfs
3/3     Running   0             20m
kube-system                         etcd-azure-example-control-plane-7ffnl
1/1     Running   0             8m15s
kube-system                         etcd-azure-example-control-plane-l4bv8
1/1     Running   0             16m
kube-system                         etcd-azure-example-control-plane-n4g4l
1/1     Running   0             19m
kube-system                         kube-apiserver-azure-example-control-plane-7ffnl
1/1     Running   0             8m16s
kube-system                         kube-apiserver-azure-example-control-plane-l4bv8
1/1     Running   0             16m
kube-system                         kube-apiserver-azure-example-control-plane-n4g4l
1/1     Running   0             19m
kube-system                         kube-controller-manager-azure-example-control-
plane-7ffnl         1/1     Running   0             8m17s
kube-system                         kube-controller-manager-azure-example-control-
plane-l4bv8         1/1     Running   0             16m
kube-system                         kube-controller-manager-azure-example-control-
plane-n4g4l         1/1     Running   1 (17m ago)   19m
kube-system                         kube-proxy-82zdl
1/1     Running   0             8m17s
```

```
kube-system                         kube-proxy-fd9f9
1/1     Running   0             18m
kube-system                         kube-proxy-l6lgc
1/1     Running   0             17m
kube-system                         kube-proxy-lzswh
1/1     Running   0             16m
kube-system                         kube-proxy-ndfmt
1/1     Running   0             20m
kube-system                         kube-proxy-nxlp9
1/1     Running   0             18m
kube-system                         kube-proxy-v9sxp
1/1     Running   0             17m
kube-system                         kube-scheduler-azure-example-control-plane-7ffnl
1/1     Running   0             8m16s
kube-system                         kube-scheduler-azure-example-control-plane-l4bv8
1/1     Running   0             16m
kube-system                         kube-scheduler-azure-example-control-plane-n4g4l
1/1     Running   1 (17m ago)   19m
node-feature-discovery              node-feature-discovery-master-84c67dcbb6-d2gm7
1/1     Running   0             20m
node-feature-discovery              node-feature-discovery-worker-drgf6
1/1     Running   0             17m
node-feature-discovery              node-feature-discovery-worker-hcz6k
1/1     Running   0             17m
node-feature-discovery              node-feature-discovery-worker-pgbcd
1/1     Running   0             16m
node-feature-discovery              node-feature-discovery-worker-vhj96
1/1     Running   0             16m
tigera-operator                     tigera-operator-d499f5c8f-jnj8b
1/1     Running   1 (18m ago)   19m
```

6.3.5 Azure Make new Cluster Self-Managed

Konvoy deploys all cluster lifecycle services to a bootstrap cluster, which then deploys a workload cluster. When the workload cluster is ready, move the cluster lifecycle services to the workload cluster, which makes the workload cluster self-managed. This section describes how to make a workload cluster self-managed.

Before starting, ensure you create a workload cluster as described in Create a New Cluster(see page 177).

6.3.5.1 Make the new Kubernetes cluster manage itself

1. Deploy cluster lifecycle services on the workload cluster:

```
dkp create capi-components --kubeconfig ${CLUSTER_NAME}.conf
```

```
✓ Initializing new CAPI components
```

2. Move the Cluster API objects from the bootstrap to the workload cluster:
The cluster lifecycle services on the workload cluster are ready, but the workload cluster configuration is on the bootstrap cluster. The `move` command moves the configuration, which takes the form of Cluster API Custom Resource objects, from the bootstrap to the workload cluster. This process is also called a Pivot[165].

```
dkp move capi-resources --to-kubeconfig ${CLUSTER_NAME}.conf
```

```
✓ Moving cluster resources
You can now view resources in the moved cluster by using the --kubeconfig flag
with kubectl. For example: kubectl --kubeconfig=azure-example.conf get nodes
```

NOTE: To ensure only one set of cluster lifecycle services manages the workload cluster, Konvoy first pauses reconciliation of the objects on the bootstrap cluster, then creates the objects on the workload cluster. As Konvoy copies the objects, the cluster lifecycle services on the workload cluster reconcile the objects. The workload cluster becomes self-managed after Konvoy creates all the objects. If it fails, the `move` command can be safely retried.

3. Wait for the cluster control-plane to be ready:

```
kubectl --kubeconfig ${CLUSTER_NAME}.conf wait --for=condition=ControlPlaneReady
"clusters/${CLUSTER_NAME}" --timeout=20m
```

```
cluster.cluster.x-k8s.io/azure-example condition met
```

4. Use the cluster lifecycle services on the workload cluster to check the workload cluster status:

NOTE: After moving the cluster lifecycle services to the workload cluster, remember to use Konvoy with the workload cluster kubeconfig.

```
dkp describe cluster --kubeconfig ${CLUSTER_NAME}.conf -c ${CLUSTER_NAME}
```

165 https://cluster-api.sigs.k8s.io/reference/glossary.html?highlight=pivot#pivot

```
NAME                                                                READY  SEVERITY
REASON  SINCE  MESSAGE
Cluster/azure-example                                               True
55s
├──ClusterInfrastructure - AzureCluster/azure-example               True
67s
├──ControlPlane - KubeadmControlPlane/azure-example-control-plane   True
55s
│ ├──Machine/azure-example-control-plane-67f47                      True
58s
│ ├──Machine/azure-example-control-plane-7pllh                      True
65s
│ └──Machine/azure-example-control-plane-jtfgv                      True
65s
└──Workers
  └──MachineDeployment/azure-example-md-0                           True
67s
    ├──Machine/azure-example-md-0-f9cb9c79b-6nsb9                   True
59s
    ├──Machine/azure-example-md-0-f9cb9c79b-jxwl6                   True
58s
    ├──Machine/azure-example-md-0-f9cb9c79b-ktg7z                   True
59s
    └──Machine/azure-example-md-0-f9cb9c79b-nxcm2                   True
66s
```

5. Remove the bootstrap cluster, as the workload cluster is now self-managed:

```
dkp delete bootstrap --kubeconfig $HOME/.kube/config
```

```
✓ Deleting bootstrap cluster
```

6.3.5.2 Known Limitations

NOTE: Be aware of these limitations in the current release of Konvoy.

- Konvoy supports moving only one set of cluster objects from the bootstrap cluster to the workload cluster, or vice-versa.
- Konvoy only supports moving all namespaces in the cluster; Konvoy does not support migration of individual namespaces.

6.3.6 Azure Certificate Renewal

6.3.6.1 Configure Automated Renewal for Managed Kubernetes PKI Certficates

During cluster creation, Kubernetes establishes a Public Key Infrastructure (PKI) for generating the TLS certificates needed for securing cluster communication for various components such as `etcd`, `kubernetes-apiserver` and `kube-proxy`. The certificates created by these components have a default expiration of one year and are renewed when an administrator updates the cluster.

Kubernetes provides a facility to renew all certificates automatically during control plane updates. For administrators who need long-running clusters or clusters that are not upgraded often, `dkp` provides automated certificate renewal, without a cluster upgrade.

6.3.6.2 Requirements

This feature requires Python 3.5 or greater to be installed on all control plane hosts.

6.3.6.3 Prerequisites

Prerequisite:

- Complete the bootstrap Cluster Lifecycle(see page 175) topic.

Create a cluster with automated certificate renewal

To enable the automated certificate renewal, create a Konvoy cluster using the `certificate-renew-interval` flag:

```
dkp create cluster azure --certificate-renew-interval=60 --cluster-name=long-running
```

The `certificate-renew-interval` is the number of days after which Kubernetes-managed PKI certificates will be renewed. For example, a `certificate-renew-interval` value of 60 means the certificates will be renewed every 60 days.

Technical details

The following manifests are modified on the control plane hosts, and are located at `/etc/kubernetes/manifests`. Modifications to these files requires SUDO access.

```
kube-controller-manager.yaml
kube-apiserver.yaml
kube-scheduler.yaml
kube-proxy.yaml
```

The following annotation indicates the time each component was reset:

```
metadata:
  annotations:
    konvoy.d2iq.io/restartedAt: $(date +%s)
```

This only occurs when the PKI certificates are older than the interval given at cluster creation time. This is activated by a `systemd` `timer` called `renew-certs.timer` that triggers an associated `systemd` service called `renew-certs.service` that runs on all of the control plane hosts.

Debugging

To debug the automatic certificate renewal feature, a cluster administrator can look at several different components to see if the certificates were renewed. For example, an administrator might start with a look at the control plane pod definition to check the last reset time. To determine if a scheduler pod was properly reset, run the command:

```
kubectl get pod -n kube-system kube-scheduler-ip-10-0-xx-xx.us-west-2.compute.interna
l -o yaml
```

The output of the command will be similar to the following:

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    konvoy.d2iq.io/restartedAt: "1626124940.735733"
```

Administrators who want more details on the execution of the `systemd` service can use `ssh` to connect to the control plane hosts, and then use the `systemctl` and `journalctl` commands that follow to help diagnose potential issues.

To check the status of the timers, when they last ran, and when they are scheduled to run next, use the command:

```
systemctl list-timers
```

To check the status of the `renew-certs` service, use the command:

```
systemctl status renew-certs
```

To get the logs of the last run of the service, use the command:

```
journalctl logs -u renew-certs
```

6.3.7 Azure Replace a Node

6.3.7.1 Replace a worker node

6.3.7.2 Prerequisites

Before you begin, you must:

- Create a workload cluster(see page 177).
- Make the new cluster self-managed(see page 187).

6.3.7.3 Replace a worker node

In certain situations, you may want to delete a worker node and have Cluster API[166] replace it with a newly provisioned machine.

1. Identify the name of the node to delete.
   List the nodes:

   ```
   kubectl --kubeconfig ${CLUSTER_NAME}.conf get nodes
   ```

   The output from this command resembles the following:

   ```
   NAME                                STATUS   ROLES                  AGE
   VERSION
   azure-example-control-plane-ckwm4   Ready    control-plane,master   35m
   v1.24.6
   azure-example-control-plane-d4fdf   Ready    control-plane,master   31m
   v1.24.6
   azure-example-control-plane-qrvm9   Ready    control-plane,master   33m
   v1.24.6
   azure-example-md-0-4w7gq            Ready    <none>                 33m   v1.24
   .6
   azure-example-md-0-6gb9k            Ready    <none>                 33m   v1.24
   .6
   azure-example-md-0-p2n8c            Ready    <none>                 11m   v1.24
   .6
   azure-example-md-0-s5zbh            Ready    <none>                 33m   v1.24
   .6
   ```

2. Export a variable with the node name to use in the next steps:
   This example uses the name `azure-example-control-plane-ckwm4`.

166 https://cluster-api.sigs.k8s.io/

```
export NAME_NODE_TO_DELETE="<azure-example-control-plane-ckwm4>"
```

3. Delete the Machine resource

```
export NAME_MACHINE_TO_DELETE=$(kubectl --kubeconfig ${CLUSTER_NAME}.conf get
machine -ojsonpath="{.items[?
(@.status.nodeRef.name==\"$NAME_NODE_TO_DELETE\")].metadata.name}")
kubectl --kubeconfig ${CLUSTER_NAME}.conf delete machine
"$NAME_MACHINE_TO_DELETE"
```

```
machine.cluster.x-k8s.io "azure-example-control-plane-slprd" deleted
```

The command will not return immediately. It will return once the Machine resource has been deleted. A few minutes after the Machine resource is deleted, the corresponding Node resource is also deleted.

4. Observe that the Machine resource is being replaced using this command:

```
kubectl --kubeconfig ${CLUSTER_NAME}.conf get machinedeployment
```

```
NAME                  CLUSTER         REPLICAS   READY   UPDATED   UNAVAILABLE
PHASE       AGE     VERSION
azure-example-md-0    azure-example   4          3       4         1
ScalingUp   7m30s   v1.24.6
long-running-md-0     long-running    4          4       4         0
Running     7m28s   v1.24.6
```

In this example, there are two replicas, but only 1 is ready. One replica is unavailable, and the `ScalingUp` phase means a new Machine is being created.

5. Identify the replacement Machine using this command:

```
export NAME_NEW_MACHINE=$(kubectl --kubeconfig ${CLUSTER_NAME}.conf get
machines \
    -l=cluster.x-k8s.io/deployment-name=${CLUSTER_NAME}-md-0 \
    -ojsonpath='{.items[?(@.status.phase=="Running")].metadata.name}{"\n"}')
echo $NAME_NEW_MACHINE
```

```
azure-example-md-0-d67567c8b-2674r azure-example-md-0-d67567c8b-n276j azure-
example-md-0-d67567c8b-pzg8k azure-example-md-0-d67567c8b-z8km9
```

If the output is empty, the new Machine has probably exited the `Provisioning` phase and entered the `Running` phase.

6. Identify the replacement Node using this command:

```
kubectl --kubeconfig ${CLUSTER_NAME}.conf get nodes
```

```
NAME                                STATUS   ROLES                  AGE
VERSION
azure-example-control-plane-d4fdf   Ready    control-plane,master   43m
v1.24.6
azure-example-control-plane-qrvm9   Ready    control-plane,master   45m
v1.24.6
azure-example-control-plane-tz56m   Ready    control-plane,master   8m22s
v1.24.6
azure-example-md-0-4w7gq            Ready    <none>                 45m     v1.2
4.6
azure-example-md-0-6gb9k            Ready    <none>                 45m     v1.2
4.6
azure-example-md-0-p2n8c            Ready    <none>                 22m     v1.2
4.6
azure-example-md-0-s5zbh            Ready    <none>                 45m     v1.2
4.6
```

If the output is empty, the Node resource is not yet available, or does not yet have the expected annotation. Wait a few minutes, then repeat the command.

6.3.8 Azure Delete Cluster

6.3.8.1 Delete the Kubernetes cluster and clean up your environment

6.3.8.2 Prepare to delete a self-managed workload cluster

NOTE: A self-managed workload cluster cannot delete itself. If your workload cluster is self-managed, you must create a bootstrap cluster and move the cluster lifecycle services to the bootstrap cluster before deleting the workload cluster.

If you did not make your workload cluster self-managed, as described in Make New Cluster Self-Managed(see page 187), proceed to Delete the workload cluster(see page 196) section below.

1. Create a bootstrap cluster:
   The bootstrap cluster will host the Cluster API controllers that reconcile the cluster objects marked for deletion:

NOTE: To avoid using the wrong kubeconfig, the following steps use explicit kubeconfig paths and contexts.

```
dkp create bootstrap --kubeconfig $HOME/.kube/config
```

```
✓ Creating a bootstrap cluster
✓ Initializing new CAPI components
```

2. Move the Cluster API objects from the workload to the bootstrap cluster: The cluster lifecycle services on the bootstrap cluster are ready, but the workload cluster configuration is on the workload cluster. The `move` command moves the configuration, which takes the form of Cluster API Custom Resource objects, from the workload to the bootstrap cluster. This process is also called a Pivot[167].

```
dkp move capi-resources \
    --from-kubeconfig ${CLUSTER_NAME}.conf \
    --from-context ${CLUSTER_NAME}-admin@${CLUSTER_NAME} \
    --to-kubeconfig $HOME/.kube/config \
    --to-context kind-konvoy-capi-bootstrapper
```

```
✓ Moving cluster resources
```

3. Use the cluster lifecycle services on the workload cluster to check the workload cluster status:

```
dkp describe cluster --kubeconfig $HOME/.kube/config -c ${CLUSTER_NAME}
```

```
NAME                                                          READY  SEVERITY
REASON  SINCE  MESSAGE
Cluster/azure-example                                         True
15s
├──ClusterInfrastructure - AzureCluster/azure-example          True
29s
├──ControlPlane - KubeadmControlPlane/azure-example-control-plane  True
15s
│  ├──Machine/azure-example-control-plane-gvj5d                 True
22s
│  ├──Machine/azure-example-control-plane-l8j9r                 True
23s
│  └──Machine/azure-example-control-plane-xhxxg                 True
23s
└──Workers
```

167 https://cluster-api.sigs.k8s.io/reference/glossary.html?highlight=pivot#pivot

```
   └──MachineDeployment/azure-example-md-0                          True
35s
     ├──Machine/azure-example-md-0-d67567c8b-2674r                  True
24s
     ├──Machine/azure-example-md-0-d67567c8b-n276j                  True
25s
     ├──Machine/azure-example-md-0-d67567c8b-pzg8k                  True
23s
     └──Machine/azure-example-md-0-d67567c8b-z8km9                  True
24s
```

NOTE: After moving the cluster lifecycle services to the workload cluster, remember to use dkp with the workload cluster kubeconfig.

Use dkp with the bootstrap cluster to delete the workload cluster.

1. Wait for the cluster control-plane to be ready:

```
kubectl --kubeconfig $HOME/.kube/config wait --for=condition=controlplaneready
"clusters/${CLUSTER_NAME}" --timeout=60m
```

```
cluster.cluster.x-k8s.io/azure-example condition met
```

Persistent Volumes (PVs) are not deleted automatically by design in order to preserve your data. However, they take up storage space if not deleted. You must delete PVs manually.

6.3.8.3 Delete the workload cluster

1. Make sure your Azure credentials are up to date. Refresh the credentials using this command:

```
dkp update bootstrap credentials azure --kubeconfig $HOME/.kube/config
```

2. Delete the Kubernetes cluster and wait a few minutes:
   Before deleting the cluster, dkp deletes all Services of type LoadBalancer on the cluster. To skip this step, use the flag `--delete-kubernetes-resources=false`.

```
dkp delete cluster --cluster-name=${CLUSTER_NAME} --kubeconfig $HOME/.kube/
config
```

```
✓ Deleting Services with type LoadBalancer for Cluster default/azure-example
✓ Deleting ClusterResourceSets for Cluster default/azure-example
✓ Deleting cluster resources
✓ Waiting for cluster to be fully deleted
Deleted default/azure-example cluster
```

After the workload cluster is deleted, delete the bootstrap cluster.

6.3.8.4 Delete the bootstrap cluster

```
dkp delete bootstrap --kubeconfig $HOME/.kube/config
```

```
✓ Deleting bootstrap cluster
```

6.3.8.5 Known Limitations

NOTE: Be aware of these limitations in the current release of Konvoy.

- The Konvoy version used to create the workload cluster must match the Konvoy version used to delete the workload cluster.

6.4 EKS Infrastructure

ENTERPRISE

When installing DKP on your EKS infrastructure, you can choose from multiple configuration types. The steps for creating and accessing your cluster are listed below:

6.4.1 EKS Introduction

ENTERPRISE

DKP brings value to EKS customers by providing all components needed for a production-ready Kubernetes environment. DKP 2.3 provides the capability to provision EKS clusters using the DKP UI, In addition to above, DKP 2.3 also provides the ability to upgrade your EKS clusters using the DKP platform, making it possible to manage the complete lifecycle of EKS clusters from a centralized platform.

DKP adds value to Amazon EKS through:

- **Time to Value** in hours/days to get to production, instead of weeks/months, or even failure. Particularly in complex environments like air-gapped, customers tried various options and even after spending millions did not see success, or saw Day 2 later than they could have. We delivered results in hours/days.
- **Less Risk**
    - **Cloud-Native Expertise** eliminates the lack of skills issue. Our industry-leading expertise closes skill gaps on the customer side, avoids costly mistakes, transfers skills, and improves project success rates while shortening timelines.
    - **Simplicity** mitigates operational complexity. We focus on a great user experience and automate the hard parts of cloud-native operations to get customers to Day 2 faster and meet all Day 2 operational challenges. This frees up time on the customer side to build what differentiates them, instead of reinventing the wheel for Kubernetes operations.
    - **Military-Grade Security** alleviates security concerns. The D2iQ Kubernetes Platform can be configured to meet NSA Kubernetes security hardening guidelines. D2iQ Kubernetes Platform and supported add-on components are security scanned and secure out of the box. Encryption of data-at-rest, FIPS compliance, and fully supported air-gapped deployments round out D2iQ offerings.
- **Lower TCO** with operational insights and a simpler platform that curates needed capabilities from Amazon EKS and the open source community that reduces the time and cost of consulting engagements as well as ongoing support costs.
- **Enterprise-grade Kubernetes** - comes with a curated list of Day 2 applications necessary for running Kubernetes in production.
- **One platform for all** - Single platform to manage multiple clusters on any infrastructure cloud, on-premise, and edge.
- **D2IQ GitOps and EKS** - Delivering business value through applications is the primary goal of any Kubernetes cluster. While EKS provides the hosted framework that leads the market, delivering applications to your environment requires a mature and integrated approach. D2IQ DKP provides workspace and project level constructs to a Kubernetes cluster so that application teams have division of resources, security and cost optimization at the the project and namespace level.
    - Projects deliver applications via the built in GitOps via FluxCD - Just provide a Git repository and DKP does the rest
    - Through integration with Kubecost, DKP monitors utilization of project resources and proves real time reporting for performance and cost optimization
    - Security of projects is defined through DKP integration of customer authentication methods and is reinforced through several layers of application security
- **Cluster Lifecycle Management through CAPI -** Through the use of cluster API, DKP gives customers full lifecycle management of their EKS clusters with the ability to instantiate new EKS clusters through a unified API. This allows administrators to deploy new EKS clusters through code and deliver consistent cluster configurations.
- Time to application value is greatly reduced by minimizing the steps necessary to provision a cluster segment clusters through integrated permissions
- Secure and reliable cluster deployments
- Automatic day 2 operations of EKS clusters (Monitoring, Logging, Central Management, Security, Cost Optimization)
- Day 2 GitOps integration with every EKS cluster

6.4.2 EKS Prerequisites

ENTERPRISE

6.4.2.1 DKP Prerequisites

Before you begin using DKP, you must have:RIP

- An x86_64-based Linux or macOS machine.
- The `dkp` binary for Linux, or macOS.
- A Self-managed AWS Cluster(see page 129)
- kubectl[168] for interacting with the running cluster.

Ensure that the `KUBECONFIG` environment variable is set to the self-managed cluster by running `export KUBECONFIG={SELF_MANAGED_AWS_CLUSTER}.conf`.

6.4.2.2 AWS prerequisites

Before you begin using DKP with AWS, you must have:

- You have a valid AWS account with credentials configured[169] that can manage CloudFormation Stacks, IAM Policies, and IAM Roles.
- You will need to have the AWS CLI utility installed[170].
- Install aws-iam-authenticator[171]. This binary is used to access your cluster using kubectl.

Minimal User Permissions for Creating EKS Clusters:

The following is a cloudformation stack which adds a policy named `eks-bootstrapper` to manage EKS cluster to the `dkp-bootstrapper-role` created by the cloudformation stack in the Minimal Permissions and Role to Create Cluster(see page 140) section. Consult the Leveraging the Role (see page 0)section for an example of how to use this role and how a system administrator wants to expose using the permissions.

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  existingBootstrapperRole:
    Type: CommaDelimitedList
    Description: 'Name of existing minimal role you want to add to add EKS cluster
management permissions to'
    Default: dkp-bootstrapper-role
Resources:
  EKSMinimumPermissions:
```

168 https://kubernetes.io/docs/tasks/tools/install-kubectl/
169 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-profiles.html
170 https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html
171 https://docs.aws.amazon.com/eks/latest/userguide/install-aws-iam-authenticator.html

```
    Properties:
      Description: Minimal user policy to manage eks clusters
      ManagedPolicyName: eks-bootstrapper
      PolicyDocument:
        Statement:
          - Action:
              - 'ssm:GetParameter'
            Effect: Allow
            Resource:
              - 'arn:*:ssm:*:*:parameter/aws/service/eks/optimized-ami/*'
          - Action:
              - 'iam:CreateServiceLinkedRole'
            Condition:
              StringLike:
                'iam:AWSServiceName': eks.amazonaws.com
            Effect: Allow
            Resource:
              - >-
                arn:*:iam::*:role/aws-service-role/eks.amazonaws.com/
AWSServiceRoleForAmazonEKS
          - Action:
              - 'iam:CreateServiceLinkedRole'
            Condition:
              StringLike:
                'iam:AWSServiceName': eks-nodegroup.amazonaws.com
            Effect: Allow
            Resource:
              - >-
                arn:*:iam::*:role/aws-service-role/eks-nodegroup.amazonaws.com/
AWSServiceRoleForAmazonEKSNodegroup
          - Action:
              - 'iam:CreateServiceLinkedRole'
            Condition:
              StringLike:
                'iam:AWSServiceName': eks-fargate.amazonaws.com
            Effect: Allow
            Resource:
              - >-
                arn:aws:iam::*:role/aws-service-role/eks-fargate-pods.amazonaws.com/
AWSServiceRoleForAmazonEKSForFargate
          - Action:
              - 'iam:GetRole'
              - 'iam:ListAttachedRolePolicies'
            Effect: Allow
            Resource:
              - 'arn:*:iam::*:role/*'
          - Action:
              - 'iam:GetPolicy'
            Effect: Allow
            Resource:
              - 'arn:aws:iam::aws:policy/AmazonEKSClusterPolicy'
          - Action:
              - 'eks:DescribeCluster'
```

```
      - 'eks:ListClusters'
      - 'eks:CreateCluster'
      - 'eks:TagResource'
      - 'eks:UpdateClusterVersion'
      - 'eks:DeleteCluster'
      - 'eks:UpdateClusterConfig'
      - 'eks:UntagResource'
      - 'eks:UpdateNodegroupVersion'
      - 'eks:DescribeNodegroup'
      - 'eks:DeleteNodegroup'
      - 'eks:UpdateNodegroupConfig'
      - 'eks:CreateNodegroup'
      - 'eks:AssociateEncryptionConfig'
      - 'eks:ListIdentityProviderConfigs'
      - 'eks:AssociateIdentityProviderConfig'
      - 'eks:DescribeIdentityProviderConfig'
      - 'eks:DisassociateIdentityProviderConfig'
    Effect: Allow
    Resource:
      - 'arn:*:eks:*:*:cluster/*'
      - 'arn:*:eks:*:*:nodegroup/*/*/*'
  - Action:
      - 'ec2:AssociateVpcCidrBlock'
      - 'ec2:DisassociateVpcCidrBlock'
      - 'eks:ListAddons'
      - 'eks:CreateAddon'
      - 'eks:DescribeAddonVersions'
      - 'eks:DescribeAddon'
      - 'eks:DeleteAddon'
      - 'eks:UpdateAddon'
      - 'eks:TagResource'
      - 'eks:DescribeFargateProfile'
      - 'eks:CreateFargateProfile'
      - 'eks:DeleteFargateProfile'
    Effect: Allow
    Resource:
      - '*'
  - Action:
      - 'iam:PassRole'
    Condition:
      StringEquals:
        'iam:PassedToService': eks.amazonaws.com
    Effect: Allow
    Resource:
      - '*'
  - Action:
      - 'kms:CreateGrant'
      - 'kms:DescribeKey'
    Condition:
      'ForAnyValue:StringLike':
        'kms:ResourceAliases': alias/cluster-api-provider-aws-*
    Effect: Allow
    Resource:
```

```
            - '*'
        Version: 2012-10-17
      Roles: !Ref existingBootstrapperRole
    Type: 'AWS::IAM::ManagedPolicy'
```

If your role is not named `dkp-bootstrapper-role` change the parameter on line 6 of the file.

To create the resources in the cloudformation stack copy the contents above into a file and run the following command after replacing `MYFILENAME.yaml` and `MYSTACKNAME` with the intended values for your system:

```
aws cloudformation create-stack --template-body=file://MYFILENAME.yaml --stack-
name=MYSTACKNAME --capabilities CAPABILITY_NAMED_IAM
```

6.4.2.3 Access Cluster

- Use previously installed aws-iam-authenticator[172] to access your cluster using kubectl. Amazon EKS uses IAM to provide authentication to your Kubernetes cluster.
- Export the AWS region where you want to deploy the cluster:

  ```
  export AWS_REGION=us-west-2
  ```

- Export the AWS profile with the credentials you want to use to create the Kubernetes cluster:

  ```
  export AWS_PROFILE=<profile>
  ```

  See the AWS site for more information about AWS credentials.[173]

6.4.3 EKS Cluster IAM Policies and Roles

ENTERPRISE

This section guides a DKP user in creating IAM Policies and Instance Profiles that determines type of access to the cluster. The IAM Role is used by the cluster's control plane and worker nodes using the provided AWS CloudFormation Stack specific to EKS. This CloudFormation Stack has code for an additional role for EKS not needed in other AWS environments.

6.4.3.1 Prerequisites:

- The user you delegate from your role must have a minimum set of permissions, see User Roles and Instance Profiles(see page 140) page under AWS for details.
- Create the Cluster IAM Policies (see page 145)in your AWS account

172 https://docs.aws.amazon.com/eks/latest/userguide/install-aws-iam-authenticator.html
173 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-profiles.html

- You will need to have the AWS CLI utility installed[174].

6.4.3.2 EKS IAM Artifacts

Policies

- `controllers-eks.cluster-api-provider-aws.sigs.k8s.io` - enumerates the Actions required by the workload cluster to create and modify EKS clusters in the user's AWS Account. It is attached to the existing `control-plane.cluster-api-provider-aws.sigs.k8s.io` role
- `eks-nodes.cluster-api-provider-aws.sigs.k8s.io` - enumerates the Actions required by the EKS workload cluster's worker machines. It Is attached to the existing `nodes.cluster-api-provider-aws.sigs.k8s.io`

Roles

- `eks-controlplane.cluster-api-provider-aws.sigs.k8s.io` - is the Role associated with EKS cluster control planes

NOTE: `control-plane.cluster-api-provider-aws.sigs.k8s.io` and `nodes.cluster-api-provider-aws.sigs.k8s.io` roles were created by Cluster IAM Policies and Roles(see page 145) in AWS.

Below is a CloudFormation stack[175] that includes IAM policies and roles required to setup EKS Clusters:

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  existingControlPlaneRole:
    Type: CommaDelimitedList
    Description: 'Names of existing Control Plane Role you want to add to the newly
created EKS Managed Policy for AWS cluster API controllers'
    Default: control-plane.cluster-api-provider-aws.sigs.k8s.io
  existingNodeRole:
    Type: CommaDelimitedList
    Description: 'ARN of the Nodes Managed Policy to add to the role for nodes'
    Default: nodes.cluster-api-provider-aws.sigs.k8s.io
Resources:
  AWSIAMManagedPolicyControllersEKS:
    Properties:
      Description: For the Kubernetes Cluster API Provider AWS Controllers
      ManagedPolicyName: controllers-eks.cluster-api-provider-aws.sigs.k8s.io
```

174 https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html
175 https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html

```
      PolicyDocument:
        Statement:
          - Action:
              - 'ssm:GetParameter'
            Effect: Allow
            Resource:
              - 'arn:*:ssm:*:*:parameter/aws/service/eks/optimized-ami/*'
          - Action:
              - 'iam:CreateServiceLinkedRole'
            Condition:
              StringLike:
                'iam:AWSServiceName': eks.amazonaws.com
            Effect: Allow
            Resource:
              - >-
                arn:*:iam::*:role/aws-service-role/eks.amazonaws.com/
AWSServiceRoleForAmazonEKS
          - Action:
              - 'iam:CreateServiceLinkedRole'
            Condition:
              StringLike:
                'iam:AWSServiceName': eks-nodegroup.amazonaws.com
            Effect: Allow
            Resource:
              - >-
                arn:*:iam::*:role/aws-service-role/eks-nodegroup.amazonaws.com/
AWSServiceRoleForAmazonEKSNodegroup
          - Action:
              - 'iam:CreateServiceLinkedRole'
            Condition:
              StringLike:
                'iam:AWSServiceName': eks-fargate.amazonaws.com
            Effect: Allow
            Resource:
              - >-
                arn:aws:iam::*:role/aws-service-role/eks-fargate-pods.amazonaws.com/
AWSServiceRoleForAmazonEKSForFargate
          - Action:
              - 'iam:GetRole'
              - 'iam:ListAttachedRolePolicies'
            Effect: Allow
            Resource:
              - 'arn:*:iam::*:role/*'
          - Action:
              - 'iam:GetPolicy'
            Effect: Allow
            Resource:
              - 'arn:aws:iam::aws:policy/AmazonEKSClusterPolicy'
          - Action:
              - 'eks:DescribeCluster'
              - 'eks:ListClusters'
              - 'eks:CreateCluster'
              - 'eks:TagResource'
```

```
            - 'eks:UpdateClusterVersion'
            - 'eks:DeleteCluster'
            - 'eks:UpdateClusterConfig'
            - 'eks:UntagResource'
            - 'eks:UpdateNodegroupVersion'
            - 'eks:DescribeNodegroup'
            - 'eks:DeleteNodegroup'
            - 'eks:UpdateNodegroupConfig'
            - 'eks:CreateNodegroup'
            - 'eks:AssociateEncryptionConfig'
            - 'eks:ListIdentityProviderConfigs'
            - 'eks:AssociateIdentityProviderConfig'
            - 'eks:DescribeIdentityProviderConfig'
            - 'eks:DisassociateIdentityProviderConfig'
          Effect: Allow
          Resource:
            - 'arn:*:eks:*:*:cluster/*'
            - 'arn:*:eks:*:*:nodegroup/*/*/*'
        - Action:
            - 'ec2:AssociateVpcCidrBlock'
            - 'ec2:DisassociateVpcCidrBlock'
            - 'eks:ListAddons'
            - 'eks:CreateAddon'
            - 'eks:DescribeAddonVersions'
            - 'eks:DescribeAddon'
            - 'eks:DeleteAddon'
            - 'eks:UpdateAddon'
            - 'eks:TagResource'
            - 'eks:DescribeFargateProfile'
            - 'eks:CreateFargateProfile'
            - 'eks:DeleteFargateProfile'
          Effect: Allow
          Resource:
            - '*'
        - Action:
            - 'iam:PassRole'
          Condition:
            StringEquals:
              'iam:PassedToService': eks.amazonaws.com
          Effect: Allow
          Resource:
            - '*'
        - Action:
            - 'kms:CreateGrant'
            - 'kms:DescribeKey'
          Condition:
            'ForAnyValue:StringLike':
              'kms:ResourceAliases': alias/cluster-api-provider-aws-*
          Effect: Allow
          Resource:
            - '*'
      Version: 2012-10-17
    Roles: !Ref existingControlPlaneRole
```

```
    Type: 'AWS::IAM::ManagedPolicy'
  AWSIAMManagedEKSNodesPolicy:
    Properties:
      Description: Additional Policies to nodes role to work for EKS
      ManagedPolicyName: eks-nodes.cluster-api-provider-aws.sigs.k8s.io
      PolicyDocument:
        Statement:
          - Action:
              - "ec2:AssignPrivateIpAddresses"
              - "ec2:AttachNetworkInterface"
              - "ec2:CreateNetworkInterface"
              - "ec2:DeleteNetworkInterface"
              - "ec2:DescribeInstances"
              - "ec2:DescribeTags"
              - "ec2:DescribeNetworkInterfaces"
              - "ec2:DescribeInstanceTypes"
              - "ec2:DetachNetworkInterface"
              - "ec2:ModifyNetworkInterfaceAttribute"
              - "ec2:UnassignPrivateIpAddresses"
            Effect: Allow
            Resource:
              - '*'
          - Action:
              - "ec2:DescribeInstances"
              - "ec2:DescribeInstanceTypes"
              - "ec2:DescribeRouteTables"
              - "ec2:DescribeSecurityGroups"
              - "ec2:DescribeSubnets"
              - "ec2:DescribeVolumes"
              - "ec2:DescribeVolumesModifications"
              - "ec2:DescribeVpcs"
              - "eks:DescribeCluster"
            Effect: Allow
            Resource:
              - '*'
        Version: 2012-10-17
      Roles: !Ref existingNodeRole
    Type: 'AWS::IAM::ManagedPolicy'
  AWSIAMRoleEKSControlPlane:
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action:
              - 'sts:AssumeRole'
            Effect: Allow
            Principal:
              Service:
                - eks.amazonaws.com
        Version: 2012-10-17
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/AmazonEKSClusterPolicy'
      RoleName: eks-controlplane.cluster-api-provider-aws.sigs.k8s.io
    Type: 'AWS::IAM::Role'
```

To create the resources in the cloudformation stack copy the contents above into a file and run the following command after replacing `MYFILENAME.yaml` and `MYSTACKNAME` with the intended values:

```
 aws cloudformation create-stack --template-body=file://MYFILENAME.yaml --stack-
name=MYSTACKNAME --capabilities  CAPABILITY_NAMED_IAM
```

6.4.4 Create an EKS Cluster from the CLI

ENTERPRISE

Ensure that the `KUBECONFIG` environment variable is set to the self-managed cluster by running `export KUBECONFIG={SELF_MANAGED_AWS_CLUSTER}.conf`

6.4.4.1 Create a New EKS Kubernetes Cluster

By default, the control-plane Nodes will be created in 3 different Availability Zones. However, the default worker Nodes will reside in a single zone. You may create additional node pools in other Availability Zones with the `dkp create nodepool` command.

Follow these steps:

1. Set the environment variable to the name you assigned this cluster.

   ```
   export CLUSTER_NAME=eks-example
   ```

   See EKS Quick Start(see page 0) for information on naming your cluster.
2. Make sure your AWS credentials are up-to-date. Refresh the credentials command is only necessary if you are using Static Credentials(see page 0) (Access Keys) otherwise, if you are using role-based authentication on a bastion host, proceed to step 3:

   ```
   dkp update bootstrap credentials aws
   ```
3. Create the cluster:

   ```
   dkp create cluster eks --cluster-name=${CLUSTER_NAME} --additional-tags=owner=$
   (whoami)
   ```

   ```
   Generating cluster resources
   ```

```
cluster.cluster.x-k8s.io/eks-example created
awsmanagedcontrolplane.controlplane.cluster.x-k8s.io/eks-example-control-plane
 created
machinedeployment.cluster.x-k8s.io/eks-example-md-0 created
awsmachinetemplate.infrastructure.cluster.x-k8s.io/eks-example-md-0 created
eksconfigtemplate.bootstrap.cluster.x-k8s.io/eks-example-md-0 created
clusterresourceset.addons.cluster.x-k8s.io/calico-cni-installation-eks-example
 created
configmap/calico-cni-installation-eks-example created
configmap/tigera-operator-eks-example created
clusterresourceset.addons.cluster.x-k8s.io/cluster-autoscaler-eks-example
 created
configmap/cluster-autoscaler-eks-example created
clusterresourceset.addons.cluster.x-k8s.io/node-feature-discovery-eks-example
 created
configmap/node-feature-discovery-eks-example created
clusterresourceset.addons.cluster.x-k8s.io/nvidia-feature-discovery-eks-example
 created
configmap/nvidia-feature-discovery-eks-example created
```

4. Inspect or edit the cluster objects:
   Use your favorite editor.
   NOTE: Editing the cluster objects requires some understanding of Cluster API. Edits can prevent the cluster from deploying successfully.
   The objects are Custom Resources[176] defined by Cluster API components, and they belong in three different categories:
   a. **Cluster**
      A *Cluster* object has references to the infrastructure-specific and control plane objects. Because this is an AWS cluster, there is an *AWSCluster* object that describes the infrastructure-specific cluster properties. Here, this means the AWS region, the VPC ID, subnet IDs, and security group rules required by the Pod network implementation.
   b. **Control Plane**
      A *AWSManagedControlPlane* object describes the control plane, which is the group of machines that run the Kubernetes control plane components, which include the etcd distributed database, the API server, the core controllers, and the scheduler. The object describes the configuration for these components. The object also has a reference to an infrastructure-specific object that describes the properties of all control plane machines.
   c. **Node Pool**
      A Node Pool is a collection of machines with identical properties. For example, a cluster might have one Node Pool with large memory capacity, another Node Pool with GPU support. Each Node Pool is described by three objects: The MachinePool references an object that describes the configuration of Kubernetes components (for example, kubelet) deployed on each node pool machine, and an infrastructure-specific object that describes the properties of all node pool machines. Here, it references a *KubeadmConfigTemplate*, and an *AWSMachineTemplate* object, which describes the instance type, the type of disk used, the size of the disk, among other properties.
   For in-depth documentation about the objects, read Concepts[177] in the Cluster API Book.
5. Wait for the cluster control-plane to be ready:

176 https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/
177 https://cluster-api.sigs.k8s.io/user/concepts.html

```
kubectl wait --for=condition=ControlPlaneReady "clusters/${CLUSTER_NAME}" --
timeout=20m
```

```
cluster.cluster.x-k8s.io/eks-example condition met
```

The `READY` status will become `True` after the cluster control-plane becomes ready in one of the following steps.

6. Once the objects are created on the API server, the Cluster API controllers reconcile them. They create infrastructure and machines. As they progress, they update the Status of each object. Konvoy provides a command to describe the current status of the cluster:

```
dkp describe cluster -c ${CLUSTER_NAME}
```

```
NAME                                                          READY
SEVERITY  REASON  SINCE  MESSAGE
Cluster/eks-example                                           True
10m
├──ControlPlane - AWSManagedControlPlane/eks-example-control-plane  True
10m
└──Workers
   └──MachineDeployment/eks-example-md-0                       True
26s
     ├──Machine/eks-example-md-0-78fcd7c7b7-66ntt              True
84s
     ├──Machine/eks-example-md-0-78fcd7c7b7-b9qmc              True
84s
     ├──Machine/eks-example-md-0-78fcd7c7b7-v5vfq              True
84s
     └──Machine/eks-example-md-0-78fcd7c7b7-zl6m2              True
84s
```

7. As they progress, the controllers also create Events. List the Events using this command:

```
kubectl get events | grep ${CLUSTER_NAME}
```

For brevity, the example uses `grep`. It is also possible to use separate commands to get Events for specific objects. For example, `kubectl get events --field-selector involvedObject.kind="AWSCluster"` and `kubectl get events --field-selector involvedObject.kind="AWSMachine"`.

```
46m         Normal   SuccessfulCreateVPC
awsmanagedcontrolplane/eks-example-control-plane   Created new managed VPC
"vpc-05e775702092abf09"
46m         Normal   SuccessfulSetVPCAttributes
awsmanagedcontrolplane/eks-example-control-plane   Set managed VPC attributes
for "vpc-05e775702092abf09"
46m         Normal   SuccessfulCreateSubnet
awsmanagedcontrolplane/eks-example-control-plane   Created new managed Subnet
"subnet-0419dd3f2dfd95ff8"
46m         Normal   SuccessfulModifySubnetAttributes
awsmanagedcontrolplane/eks-example-control-plane   Modified managed Subnet
"subnet-0419dd3f2dfd95ff8" attributes
46m         Normal   SuccessfulCreateSubnet
awsmanagedcontrolplane/eks-example-control-plane   Created new managed Subnet
"subnet-0e724b128e3113e47"
46m         Normal   SuccessfulCreateSubnet
awsmanagedcontrolplane/eks-example-control-plane   Created new managed Subnet
"subnet-06b2b31ea6a8d3962"
46m         Normal   SuccessfulModifySubnetAttributes
awsmanagedcontrolplane/eks-example-control-plane   Modified managed Subnet
"subnet-06b2b31ea6a8d3962" attributes
46m         Normal   SuccessfulCreateSubnet
awsmanagedcontrolplane/eks-example-control-plane   Created new managed Subnet
"subnet-0626ce238be32bf98"
46m         Normal   SuccessfulCreateSubnet
awsmanagedcontrolplane/eks-example-control-plane   Created new managed Subnet
"subnet-0f53cf59f83177800"
46m         Normal   SuccessfulModifySubnetAttributes
awsmanagedcontrolplane/eks-example-control-plane   Modified managed Subnet
"subnet-0f53cf59f83177800" attributes
46m         Normal   SuccessfulCreateSubnet
awsmanagedcontrolplane/eks-example-control-plane   Created new managed Subnet
"subnet-0878478f6bbf153b2"
46m         Normal   SuccessfulCreateInternetGateway
awsmanagedcontrolplane/eks-example-control-plane   Created new managed Internet
Gateway "igw-09fb52653949d4579"
46m         Normal   SuccessfulAttachInternetGateway
awsmanagedcontrolplane/eks-example-control-plane   Internet Gateway
"igw-09fb52653949d4579" attached to VPC "vpc-05e775702092abf09"
46m         Normal   SuccessfulCreateNATGateway
awsmanagedcontrolplane/eks-example-control-plane   Created new NAT Gateway
"nat-06356aac28079952d"
46m         Normal   SuccessfulCreateNATGateway
awsmanagedcontrolplane/eks-example-control-plane   Created new NAT Gateway
"nat-0429d1cd9d956bf35"
46m         Normal   SuccessfulCreateNATGateway
awsmanagedcontrolplane/eks-example-control-plane   Created new NAT Gateway
"nat-059246bcc9d4e88e7"
46m         Normal   SuccessfulCreateRouteTable
awsmanagedcontrolplane/eks-example-control-plane   Created managed RouteTable
"rtb-01689c719c484fd3c"
```

```
46m         Normal   SuccessfulCreateRoute
awsmanagedcontrolplane/eks-example-control-plane   Created route {...
46m         Normal   SuccessfulAssociateRouteTable
awsmanagedcontrolplane/eks-example-control-plane   Associated managed
RouteTable "rtb-01689c719c484fd3c" with subnet "subnet-0419dd3f2dfd95ff8"
46m         Normal   SuccessfulCreateRouteTable
awsmanagedcontrolplane/eks-example-control-plane   Created managed RouteTable
"rtb-065af81b9752eeb69"
46m         Normal   SuccessfulCreateRoute
awsmanagedcontrolplane/eks-example-control-plane   Created route {...
46m         Normal   SuccessfulAssociateRouteTable
awsmanagedcontrolplane/eks-example-control-plane   Associated managed
RouteTable "rtb-065af81b9752eeb69" with subnet "subnet-0e724b128e3113e47"
46m         Normal   SuccessfulCreateRouteTable
awsmanagedcontrolplane/eks-example-control-plane   Created managed RouteTable
"rtb-03eeff810a89afc98"
46m         Normal   SuccessfulCreateRoute
awsmanagedcontrolplane/eks-example-control-plane   Created route {...
46m         Normal   SuccessfulAssociateRouteTable
awsmanagedcontrolplane/eks-example-control-plane   Associated managed
RouteTable "rtb-03eeff810a89afc98" with subnet "subnet-06b2b31ea6a8d3962"
46m         Normal   SuccessfulCreateRouteTable
awsmanagedcontrolplane/eks-example-control-plane   Created managed RouteTable
"rtb-0fab36f8751fdee73"
46m         Normal   SuccessfulCreateRoute
awsmanagedcontrolplane/eks-example-control-plane   Created route {...
46m         Normal   SuccessfulAssociateRouteTable
awsmanagedcontrolplane/eks-example-control-plane   Associated managed
RouteTable "rtb-0fab36f8751fdee73" with subnet "subnet-0626ce238be32bf98"
46m         Normal   SuccessfulCreateRouteTable
awsmanagedcontrolplane/eks-example-control-plane   Created managed RouteTable
"rtb-0e5c9c7bbc3740a0f"
46m         Normal   SuccessfulCreateRoute
awsmanagedcontrolplane/eks-example-control-plane   Created route {...
46m         Normal   SuccessfulAssociateRouteTable
awsmanagedcontrolplane/eks-example-control-plane   Associated managed
RouteTable "rtb-0e5c9c7bbc3740a0f" with subnet "subnet-0f53cf59f83177800"
46m         Normal   SuccessfulCreateRouteTable
awsmanagedcontrolplane/eks-example-control-plane   Created managed RouteTable
"rtb-0bf58eb5f73c387af"
46m         Normal   SuccessfulCreateRoute
awsmanagedcontrolplane/eks-example-control-plane   Created route {...
46m         Normal   SuccessfulAssociateRouteTable
awsmanagedcontrolplane/eks-example-control-plane   Associated managed
RouteTable "rtb-0bf58eb5f73c387af" with subnet "subnet-0878478f6bbf153b2"
46m         Normal   SuccessfulCreateSecurityGroup
awsmanagedcontrolplane/eks-example-control-plane   Created managed
SecurityGroup "sg-0b045c998a120a1b2" for Role "node-eks-additional"
46m         Normal   InitiatedCreateEKSControlPlane
awsmanagedcontrolplane/eks-example-control-plane   Initiated creation of a new
 EKS control plane default_eks-example-control-plane
```

```
37m         Normal   SuccessfulCreateEKSControlPlane
awsmanagedcontrolplane/eks-example-control-plane   Created new EKS control
plane default_eks-example-control-plane
37m         Normal   SucessfulCreateKubeconfig
awsmanagedcontrolplane/eks-example-control-plane   Created kubeconfig for
 cluster "eks-example"
37m         Normal   SucessfulCreateUserKubeconfig
awsmanagedcontrolplane/eks-example-control-plane   Created user kubeconfig for
 cluster "eks-example"
27m         Normal   SuccessfulCreate
awsmachine/eks-example-md-0-4t9nc                  Created new node instance
with id "i-0aecc1897c93df740"
26m         Normal   SuccessfulDeleteEncryptedBootstrapDataSecrets
awsmachine/eks-example-md-0-4t9nc                  AWS Secret entries
containing userdata deleted
26m         Normal   SuccessfulSetNodeRef                            machine/
eks-example-md-0-78fcd7c7b7-fn7x9          ip-10-0-88-24.us-west-2.compute.inte
rnal
26m         Normal   SuccessfulSetNodeRef                            machine/
eks-example-md-0-78fcd7c7b7-g64nv          ip-10-0-110-219.us-west-2.compute.in
ternal
26m         Normal   SuccessfulSetNodeRef                            machine/
eks-example-md-0-78fcd7c7b7-gwc5j          ip-10-0-101-161.us-west-2.compute.in
ternal
26m         Normal   SuccessfulSetNodeRef                            machine/
eks-example-md-0-78fcd7c7b7-j58s4          ip-10-0-127-49.us-west-2.compute.int
ernal
46m         Normal   SuccessfulCreate
machineset/eks-example-md-0-78fcd7c7b7             Created machine "eks-
example-md-0-78fcd7c7b7-fn7x9"
46m         Normal   SuccessfulCreate
machineset/eks-example-md-0-78fcd7c7b7             Created machine "eks-
example-md-0-78fcd7c7b7-g64nv"
46m         Normal   SuccessfulCreate
machineset/eks-example-md-0-78fcd7c7b7             Created machine "eks-
example-md-0-78fcd7c7b7-j58s4"
46m         Normal   SuccessfulCreate
machineset/eks-example-md-0-78fcd7c7b7             Created machine "eks-
example-md-0-78fcd7c7b7-gwc5j"
27m         Normal   SuccessfulCreate
awsmachine/eks-example-md-0-7whkv                  Created new node instance
with id "i-06dfc0466b8f26695"
26m         Normal   SuccessfulDeleteEncryptedBootstrapDataSecrets
awsmachine/eks-example-md-0-7whkv                  AWS Secret entries
containing userdata deleted
27m         Normal   SuccessfulCreate
awsmachine/eks-example-md-0-ttgzv                  Created new node instance
with id "i-0544fce0350fd41fb"
26m         Normal   SuccessfulDeleteEncryptedBootstrapDataSecrets
awsmachine/eks-example-md-0-ttgzv                  AWS Secret entries
containing userdata deleted
```

```
27m         Normal   SuccessfulCreate
awsmachine/eks-example-md-0-v2hrf                  Created new node instance
with id "i-0498906edde162e59"
26m         Normal   SuccessfulDeleteEncryptedBootstrapDataSecrets
awsmachine/eks-example-md-0-v2hrf                  AWS Secret entries
containing userdata deleted
46m         Normal   SuccessfulCreate
machinedeployment/eks-example-md-0                 Created MachineSet "eks-
example-md-0-78fcd7c7b7"
```

6.4.4.2 Known Limitations

Be aware of these limitations in the current release of Konvoy.

- The Konvoy version used to create a workload cluster must match the Konvoy version used to delete a workload cluster.
- EKS clusters cannot be Self-managed.
- Konvoy supports deploying one workload cluster.
- Konvoy generates a set of objects for one Node Pool.
- Konvoy does not validate edits to cluster objects.

6.4.5 Create an EKS Cluster from the UI

ENTERPRISE

DKP UI allows you to quickly and easily provision a Cluster from your browser.

6.4.5.1 Create an AWS Infrastructure Provider

Before you create a Cluster, you first need to create an AWS infrastructure provider to hold your AWS/EKS Credentials:

1. Get the AWS RoleARN(see page 140).

   ```
   aws iam get-role --role-name <role-name> --query 'Role.[RoleName, Arn]' --
   output text
   ```

2. Select **Infrastructure Providers** from the Dashboard menu.
3. Select **Add Infrastructure Provider**.
4. Choose a workspace. If you are already in a workspace, the provider is automatically created in that workspace.
5. Ensure you select **Amazon Web Services**.
6. Add a **Name** for your Infrastructure Provider and include the **Role ARN** from Step 1 above.
7. Select **Save**.

If you choose to, you can use static credentials. However, this method is not as secure so it is not recommended.

6.4.5.2 Provision an EKS Cluster

Follow these steps to provision the EKS cluster:

1. From the top menu bar, select your target workspace.
2. Select **Clusters > Add Cluster**.
   This begins the provisioning workflow.
3. Choose **Create Cluster**.
4. Enter the **Cluster Name**.
5. Select **EKS** from the **Choose Infrastructure** choices.
6. If available, choose a **Kubernetes Version**. Otherwise, the default Kubernetes version(see page 811) installs.
7. Select a data center **region** or specify a custom **region**.
8. Edit your worker **Node Pools** as necessary. You can choose the **Number of Nodes**, the **Machine Type**, and our **IAM Instance Profile**.
9. Add any additional **Labels** or **Infrastructure Provider Tags** as necessary.
10. Validate your inputs, and then select **Create**.

You are redirected to the **Clusters** page, where you see your **Cluster** in the **Provisioning** status. Hover over the status to view the details.

After about 15 minutes, your **Cluster** should be in the **Provisioned** status.

See AWS RoleARN[178] for more information from the AWS site.

6.4.5.3 Access EKS Cluster

How to access an attached (managed) cluster with Kommander credentials

6.4.5.4 Accessing your managed clusters using your Kommander administrator credentials

After the cluster is successfully attached, you can retrieve a custom kubeconfig file from the UI.

1. Select the Kommander username in the top right corner, and then select **Generate Token**.
2. Select the attached cluster name, and follow the instructions to assemble a kubeconfig for accessing its Kubernetes API. If Kommander prompts you to log in, use the credentials used to first login to the UI.

You can also retrieve a custom kubeconfig file by visiting the `/token` endpoint on the Kommander cluster domain (example URL: `https://your-server-name.your-region.elb.service.com/token/`. Selecting the attached cluster name displays the instructions to assemble a kubeconfig for accessing its Kubernetes API.

6.4.6 Grant Cluster Access

ENTERPRISE

178 https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html#identifiers-arns

6.4.6.1 How to Grant Cluster Access

You can access your cluster using AWS IAM roles in the dashboard. When you create an EKS cluster, the IAM entity is granted `system:masters` permissions in Kubernetes Role Based Access Control (RBAC) configuration.[179]

> More information about the configuration of the EKS control plane can be found on the EKS Cluster IAM Policies and Roles[180] page.

If the EKS cluster was created as a cluster using a self-managed AWS cluster that uses IAM Instance Profiles, you will need to modify the `IAMAuthenticatorConfig` field in the `AWSManagedControlPlane` API object to allow other IAM entities to access the EKS workload cluster. Follow the steps below:

1. Execute the following command with your `KUBECONFIG` configured to select the self-managed cluster previously used to create the workload EKS cluster. Ensure you substitute `${CLUSTER_NAME}` and `${CLUSTER_NAMESPACE}` with their corresponding values for your cluster.

   ```
   kubectl edit awsmanagedcontrolplane ${CLUSTER_NAME}-control-plane -n ${CLUSTER_NAMESPACE}
   ```

2. Edit the `IamAuthenticatorConfig` field with the IAM Role to the corresponding Kubernetes Role. In this example, the IAM role `arn:aws:iam::111122223333:role/PowerUser` is granted the cluster role `system:masters`. Note that this example uses example AWS resource ARNs[181], so these values should be substituted for real values in the corresponding AWS account.

   ```
   iamAuthenticatorConfig:
       mapRoles:
         - groups:
           - system:bootstrappers
           - system:nodes
           rolearn: arn:aws:iam::111122223333:role/my-node-role
           username: system:node:{{EC2PrivateDNSName}}
         - groups:
           - system:masters
           rolearn: arn:aws:iam::111122223333:role/PowerUser
           username: admin
   ```

For further instructions on changing or assigning `roles` or `clusterroles` to which you can map IAM users or roles, see Amazon Enabling IAM access to your cluster[182].

---

179 https://kubernetes.io/docs/reference/access-authn-authz/rbac/
180 https://d2iq.atlassian.net/wiki/spaces/DENT/pages/144182022/%282.4%29+EKS+Cluster+IAM+Policies+and+Roles
181 https://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html
182 https://docs.aws.amazon.com/eks/latest/userguide/add-user-role.html

6.4.7 Explore EKS Cluster

ENTERPRISE

This guide explains how to use the command line to interact with your newly deployed Kubernetes cluster.

Before you start, make sure you have created a workload cluster, as described in Create a New Cluster(see page 207).

6.4.7.1 Explore the new Kubernetes cluster

Follow these steps:

1. Get a kubeconfig file for the workload cluster:
   When the workload cluster is created, the cluster lifecycle services generate a kubeconfig file for the workload cluster, and write it to a *Secret*. The kubeconfig file is scoped to the cluster administrator. Get the kubeconfig from the *Secret*, and write it to a file, using this command:

   ```
   dkp get kubeconfig -c ${CLUSTER_NAME} > ${CLUSTER_NAME}.conf
   ```

2. List the Nodes using this command:

   ```
   kubectl --kubeconfig=${CLUSTER_NAME}.conf get nodes
   ```

   ```
   NAME                                          STATUS   ROLES    AGE   VERSION
   ip-10-0-122-211.us-west-2.compute.internal    Ready    <none>   32s   v1.21.5-
   eks-9017834
   ip-10-0-127-74.us-west-2.compute.internal     Ready    <none>   42s   v1.21.5-
   eks-9017834
   ip-10-0-71-155.us-west-2.compute.internal     Ready    <none>   46s   v1.21.5-
   eks-9017834
   ip-10-0-93-47.us-west-2.compute.internal      Ready    <none>   51s   v1.21.5-
   eks-9017834
   ```

   NOTE: It may take a few minutes for the Status to move to `Ready` while the Pod network is deployed. The Nodes' Status should change to Ready soon after the `calico-node` DaemonSet Pods are Ready.

3. List the Pods using this command:

   ```
   kubectl --kubeconfig=${CLUSTER_NAME}.conf get --all-namespaces pods
   ```

```
NAMESPACE                NAME                                                READY
STATUS     RESTARTS   AGE
calico-system            calico-kube-controllers-69845d4df5-sc9vq            1/1
Running    0          44s
calico-system            calico-node-5lppw                                   1/1
Running    0          44s
calico-system            calico-node-dwbfj                                   1/1
Running    0          44s
calico-system            calico-node-q6tg6                                   1/1
Running    0          44s
calico-system            calico-node-rbm7c                                   1/1
Running    0          44s
calico-system            calico-typha-68c68c96d-tcrxn                        1/1
Running    0          35s
calico-system            calico-typha-68c68c96d-xhrjv                        1/1
Running    0          44s
kube-system              aws-node-25bnt                                      1/1
Running    0          80s
kube-system              aws-node-dr4b7                                      1/1
Running    0          89s
kube-system              aws-node-mmn87                                      1/1
Running    0          70s
kube-system              aws-node-z6cdb                                      1/1
Running    0          84s
kube-system              cluster-autoscaler-68c759fbf6-zszxr                 0/1
Init:0/1   0          9m50s
kube-system              coredns-85d5b4454c-n54rq                            1/1
Running    0          12m
kube-system              coredns-85d5b4454c-xzd9w                            1/1
Running    0          12m
kube-system              kube-proxy-4bhzp                                    1/1
Running    0          84s
kube-system              kube-proxy-5hkv9                                    1/1
Running    0          80s
kube-system              kube-proxy-g82d7                                    1/1
Running    0          70s
kube-system              kube-proxy-h2jv5                                    1/1
Running    0          89s
node-feature-discovery   node-feature-discovery-master-84c67dcbb6-s6874      1/1
Running    0          9m50s
node-feature-discovery   node-feature-discovery-worker-677hh                 1/1
Running    0          69s
node-feature-discovery   node-feature-discovery-worker-fvjwz                 1/1
Running    0          49s
node-feature-discovery   node-feature-discovery-worker-xcgvt                 1/1
Running    0          64s
node-feature-discovery   node-feature-discovery-worker-zctnz                 1/1
Running    0          60s
tigera-operator          tigera-operator-d499f5c8f-b56xn                     1/1
Running    1          9m47s
```

6.4.8 Manage EKS Nodepools

ENTERPRISE

Ensure that the `KUBECONFIG` environment variable is set to the self-managed cluster by running `export KUBECONFIG={SELF_MANAGED_AWS_CLUSTER}.conf`

Node pools are part of a cluster and managed as a group, and can be used to manage a group of machines using common properties. New default clusters created by Konvoy contain one node pool of worker nodes that have the same configuration.

You can create additional node pools for specialized hardware or other configurations. For example, if you want to tune your memory usage on a cluster where you need maximum memory for some machines and minimal memory on others, you can create a new node pool with those specific resource needs.

NOTE: Konvoy implements node pools using Cluster API MachineDeployments[183].

6.4.8.1 Create a node pool

Availability zones (AZs) are isolated locations within data center regions from which public cloud services originate and operate. Because all the nodes in a node pool are deployed in a single Availability Zone, you may wish to create additional node pools is to ensure your cluster has nodes deployed in multiple Availability Zones.

By default, the first Availability Zone[184] in the region is used for the nodes in the node pool. To create the nodes in a different Availability Zone set the appropriate `--availability-zone`.

To create a new EKS node pool with 3 replicas, run:

```
dkp create nodepool eks ${NODEPOOL_NAME} \
    --cluster-name=${CLUSTER_NAME} \
    --replicas=3
```

```
machinedeployment.cluster.x-k8s.io/example created
awsmachinetemplate.infrastructure.cluster.x-k8s.io/example created
eksconfigtemplate.bootstrap.cluster.x-k8s.io/example created
 ✓ Creating default/example nodepool resources
```

Advanced users can use a combination of the `--dry-run` and `--output=yaml` flags to get a complete set of node pool objects to modify locally or store in version control.

183 https://cluster-api.sigs.k8s.io/developer/architecture/controllers/machine-deployment.html
184 https://aws.amazon.com/about-aws/global-infrastructure/regions_az/

6.4.8.2 Scaling Up Node Pools

To scale up a node pool in a cluster, run:

```
dkp scale nodepools ${NODEPOOL_NAME} --replicas=5 --cluster-name=${CLUSTER_NAME}
```

Your output should be similar to this example, indicating the scaling is in progress:

```
✓ Scaling node pool example to 5 replicas
```

After a few minutes, you can list the node pools to:

```
dkp get nodepools --cluster-name=${CLUSTER_NAME}
```

Your output should be similar to this example, with the number of DESIRED and READY replicas increased to 5:

```
NODEPOOL                        DESIRED             READY
KUBERNETES VERSION
example                         5                   5                   v1.23.6
eks-example-md-0                4                   4                   v1.23.6
```

6.4.8.3 Delete EKS Node Pools

Delete node pools in a cluster

Deleting a node pool deletes the Kubernetes nodes and the underlying infrastructure. All nodes are drained prior to deletion and the pods running on those nodes are rescheduled.

To delete a node pool from a managed cluster, run:

```
dkp delete nodepool ${NODEPOOL_NAME} --cluster-name=${CLUSTER_NAME}
```

Here, `example` is the node pool to be deleted.

The expected output will be similar to the following example, indicating the node pool is being deleted:

```
✓ Deleting default/example nodepool resources
```

Deleting an invalid node pool results in output similar to this example:

```
dkp delete nodepool ${CLUSTER_NAME}-md-invalid --cluster-name=${CLUSTER_NAME}
```

```
MachineDeployments or MachinePools.infrastructure.cluster.x-k8s.io "no
MachineDeployments or MachinePools found for cluster eks-example" not found
```

6.4.9 Delete EKS Cluster from CLI

ENTERPRISE

Ensure that the `KUBECONFIG` environment variable is set to the self-managed cluster by running `export KUBECONFIG={SELF_MANAGED_AWS_CLUSTER}.conf`

6.4.9.1 Delete the EKS cluster

Follow these steps:

1. Ensure your AWS credentials are up to date. If you are using user profiles, then refresh the credentials using the command below. Otherwise, proceed to step 2.

   ```
   dkp update bootstrap credentials aws
   ```

2. Delete the Kubernetes cluster and wait a few minutes:
   Before deleting the cluster, dkp deletes all Services of type LoadBalancer on the cluster. Each Service is backed by an AWS Classic ELB. Deleting the Service deletes the ELB that backs it. To skip this step, use the flag `--delete-kubernetes-resources=false`.
   NOTE: Do not skip this step if the VPC is managed by DKP. When DKP deletes the cluster, it deletes the VPC. If the VPC has any EKS Classic ELBs, EKS does not allow the VPC to be deleted, and DKP cannot delete the cluster.

   ```
   dkp delete cluster --cluster-name=${CLUSTER_NAME}
   ```

   ```
    ✓ Deleting Services with type LoadBalancer for Cluster default/eks-example
    ✓ Deleting ClusterResourceSets for Cluster default/eks-example
    ✓ Deleting cluster resources
    ✓ Waiting for cluster to be fully deleted
   Deleted default/eks-example cluster
   ```

6.4.9.2 Known Limitations

NOTE: Be aware of these limitations in the current release of DKP.

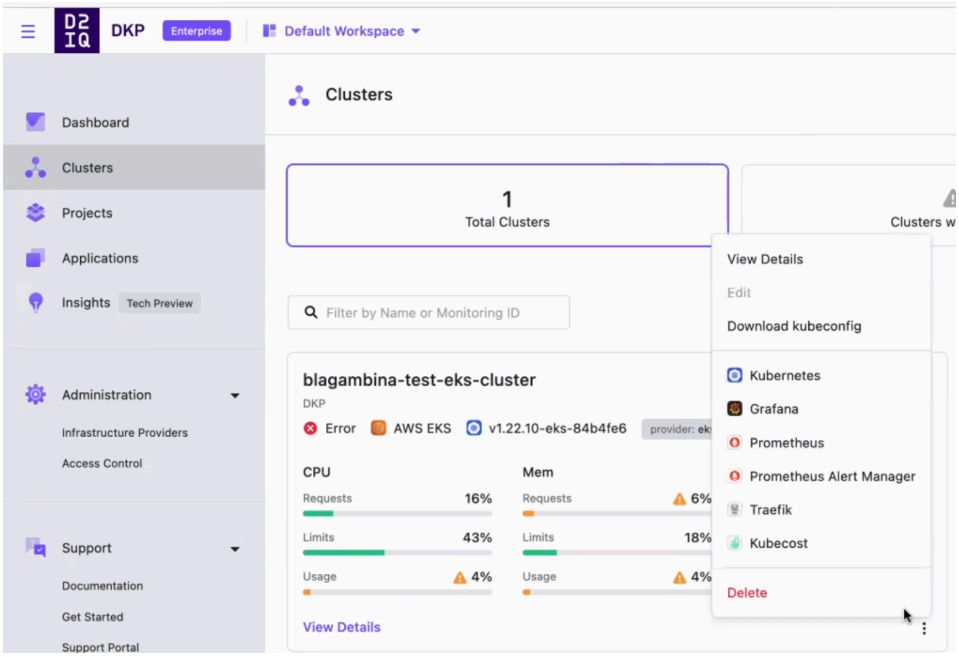
- The DKP version used to create the workload cluster must match the DKP version used to delete the workload cluster.

6.4.10 Delete EKS Cluster from UI

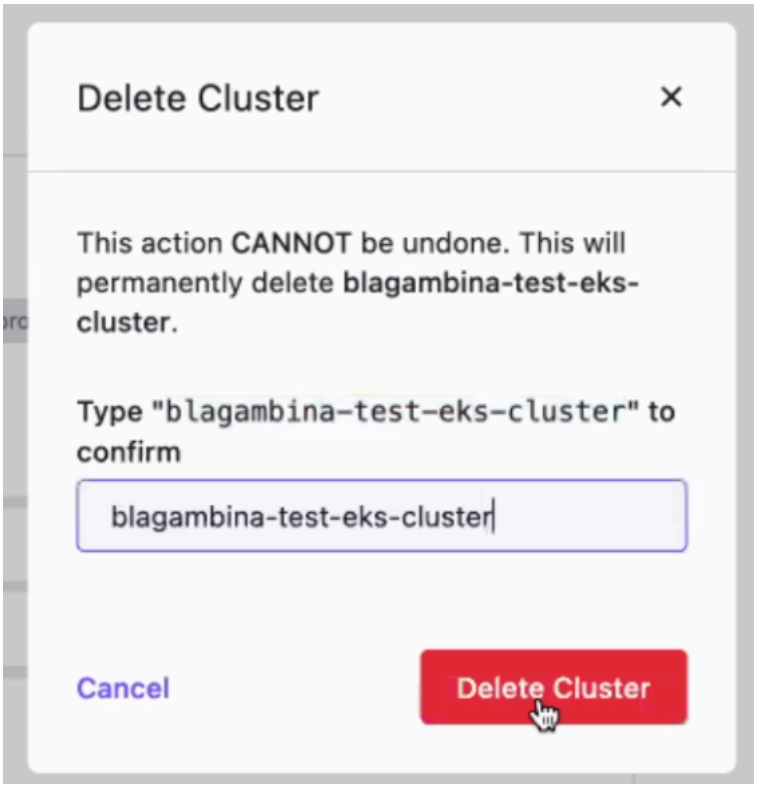
ENTERPRISE

In order to delete a cluster in the UI, you must first have created a cluster(see page 213) and have permissions to delete. Otherwise, an administrator privilege could delete a cluster as well.

1. Open the dashboard and select Clusters in the left menu.
2. Select the cluster you wish to delete and click the dotted icon in the bottom right corner.
3. Then select Delete in red.



4. When the next screen appears, copy the name of your cluster and paste it into the empty box.
5. Now execute the deletion using Delete Cluster button.



6. You will see the status as “Deleting” in the top left corner of the cluster you selected for deletion.

This removes the cluster from the DKP UI. For a generic overview of deleting clusters within the UI as well as troubleshooting, see the Disconnect or Delete Clusters(see page 551) instructions.

6.5 Pre-provisioned Infrastructure

6.5.1 Create a Kubernetes cluster on pre-provisioned infrastructure

The following procedure describes creating a DKP cluster on a pre-provisioned infrastructure using SSH.

Completing this procedure results in a Kubernetes cluster that includes a Container Networking Interface (CNI)[185] and a Local Persistence Volume Static Provisioner[186], and that is ready for workload deployment.

Before moving to a production environment, you may want to add applications for logging and monitoring, storage, security, and other functions. You can use DKP to select and deploy applications(see page 424), or deploy your own.

To get started, see these topics.

- Pre-provisioned Prerequisites(see page 223)
- Pre-provisioned Prerequisites Air-gapped(see page 224)
- Pre-provisioned Bootstrap(see page 229)
- Pre-provisioned Create Necessary Secrets and Overrides(see page 229)
- Pre-provisioned Define Infrastructure(see page 231)
- Pre-provisioned Define Control Plane Endpoint(see page 233)

185 https://docs.projectcalico.org/
186 https://github.com/kubernetes-sigs/sig-storage-local-static-provisioner

- Pre-provisioned Create new Cluster(see page 234)
- Pre-provisioned make Cluster Self-managed(see page 244)
- Pre-provisioned Configure MetalLB(see page 246)
- Pre-provisioned Create and Delete Node Pools(see page 248)
- Pre-provisioned Delete Cluster(see page 249)

6.5.2 Pre-provisioned Prerequisites

6.5.2.1 Fulfill the prerequisites for using a pre-provisioned infrastructure

Before you begin using DKP, you must have:

- An x86_64-based Linux or macOS machine.
- The `dkp` binary for Linux, or macOS.
- kubectl[187] for interacting with the running cluster.
- Pre-provisioned hosts with SSH access enabled.
- An unencrypted SSH private key, whose public key is configured on the above hosts.
- Resource requirements(see page 96)

When air-gapped, you must follow the steps described in the air-gapped prerequisites page(see page 224), otherwise begin creating the bootstrap cluster(see page 229).

6.5.2.2 Control plane machines

You should have at least three control plane machines.

Each control plane machine must have:

- 4 cores
- 16 GiB memory
- Approximately 80 GiB of free space for the volume used for `/var/lib/kubelet` and `/var/lib/containerd`.
- 15% free space on the root file system.
- Multiple ports open, as described in DKP Ports(see page 76).
- `firewalld` systemd service disabled. If it exists and is enabled, use the commands `systemctl stop firewalld` then `systemctl disable firewalld`, so that `firewalld` remains disabled after the machine restarts.

Swap is disabled. The `kubelet` does not have generally-available support for swap. Due to variable commands, refer to your operating system(see page 223) documentation.

6.5.2.3 Worker machines

You should have at least four worker machines. The specific number of worker machines required for your environment can vary depending on the cluster workload and size of the machines.

187 https://kubernetes.io/docs/tasks/tools/#kubectl

Each worker machine must have:

- 8 cores
- 32 GiB memory
- Around 80 GiB of free space for the volume used for `/var/lib/kubelet` and `/var/lib/containerd`.
- 15% free space on the root file system
- If you plan to use local volume provisioning to provide persistent volumes for your workloads, you must mount at least four volumes to the `/mnt/disks/` mount point on each machine. Each volume must have at least 55 GiB of capacity.
- Multiple ports open, as described in DKP Ports(see page 76).
- `firewalld` systemd service disabled. If it exists and is enabled, use the commands `systemctl stop firewalld` then `systemctl disable firewalld`, so that `firewalld` remains disabled after the machine restarts.

Swap is disabled. The `kubelet` does not have generally-available support for swap. Due to variable commands, refer to your operating system(see page 223) documentation.

6.5.3 Pre-provisioned Prerequisites Air-gapped

6.5.3.1 2.3 Fulfill the prerequisites for using a pre-provisioned infrastructure when Air-Gapped

The instructions below outline how to fulfill the prerequisites for using pre-provisioned infrastructure when using air-gapped.

6.5.3.2 Air-Gapped Registry Prerequisites

JFrog Artifactory

If you use **Jfrog Artifactory** or **Jfrog Container Registry**, you must update to a new version of the software. Any build newer than version **7.11** will work, as we have confirmed that older versions are not compatible.

Nexus Registry

If you use **Nexus Registry**, there is currently an issue that prevents usage with DKP 2.X and OCI Images. Support for OCI Images was added here in this publicly available Jira ticket:

https://issues.sonatype.org/browse/NEXUS-21087

A new issue was filed here determining that OCI image support is currently broken:

https://issues.sonatype.org/plugins/servlet/mobile#issue/NEXUS-27494

You can track this Jira link for status on a resolution for this issue.

Harbor Registry

Any newer version than **Harbor Registry v2.1.1**-5f52168e will support OCI images.

6.5.3.3 Download the bootstrap image

1. Download the bootstrap docker image on a machine that has access to this artifact:

```
curl -O https://downloads.d2iq.com/dkp/v2.3.3/konvoy-bootstrap_v2.3.3.tar
```

2. Load the bootstrap Docker image on your bastion machine:

```
docker load -i konvoy-bootstrap_v2.3.3.tar
```

6.5.3.4 Copy air-gapped artifacts onto cluster hosts

Using the Konvoy Image Builder(see page 322),you can copy the required artifacts onto your cluster hosts.

1. Create the directories where you will place the air-gapped bundles:

```
mkdir artifacts
mkdir artifacts/images
```

2. Define an environment variable for the Kubernetes version that corresponds with Konvoy release you are installing. You can find the correct Kubernetes version by checking the release notes for the release you are installing:

```
export VERSION=1.23.12
```

3. Set an environment variable for the image's OS you want to use. The OS packages bundles will contain the RPMs for Kubernetes and all of their dependencies required to install these packages without access to any external RPM repositories. The available options for your command are listed below followed by the command in which to replace them:
   - `centos_7_x86_64`
   - `centos_7_x86_64_fips`
   - `redhat_7_x86_64`
   - `redhat_7_x86_64_fips`
   - `redhat_8_x86_64`
   - `redhat_8_x86_64_fips`

```
export BUNDLE_OS=centos_7_x86_64
```

4. Download the OS packages bundle:

```
curl --output artifacts/"$VERSION"_"$BUNDLE_OS".tar.gz -O https://
downloads.d2iq.com/dkp/airgapped/os-packages/"$VERSION"_"$BUNDLE_OS".tar.gz
```

5. Download the Kubernetes images bundle. This bundle includes the necessary images for `kubeadm` to bootstrap a Kubernetes `Node`.
   The available options for each Kubernetes version are:
   - `<version>_images.tar.gz`

     ```
     curl --output artifacts/images/"$VERSION"_images.tar.gz -O https://
     downloads.d2iq.com/dkp/airgapped/kubernetes-
     images/"$VERSION"_images.tar.gz
     ```

   - `<version>_images_fips.tar.gz`

     ```
     curl --output artifacts/images/"$VERSION"_images_fips.tar.gz -O https://
     downloads.d2iq.com/dkp/airgapped/kubernetes-
     images/"$VERSION"_images_fips.tar.gz
     ```

6. Download the PIP packages using the command below. This bundle includes a few packages required by DKP to bootstrap machines.

   ```
   curl --output artifacts/pip-packages.tar.gz -O https://downloads.d2iq.com/dkp/
   airgapped/pip-packages/pip-packages.tar.gz
   ```

7. Download the Containerd `1.14.13` packages for the OS you plan to provision dkp on. The options for OS are listed below for replacement in the command before running:
   - `centos-7.9`
   - `ol-7.9`
   - `rhel-7.9`
   - `rhel-8.2`
   - `rhel-8.4`
   - `sles-15.3`
   - `ubuntu-18.04`
   - `ubuntu-20.04`

   ```
   export CONTAINERD_OS=centos-7.9
   ```

   ```
   curl --output artifacts/containerd-1.4.13-d2iq.1-"$CONTAINERD_OS"-x86_64.tar.gz
   --location https://packages.d2iq.com/dkp/containerd/containerd-1.4.13-d2iq.1-"$
   CONTAINERD_OS"-x86_64.tar.gz
   ```

   To get the fips builds append `_fips` after `-x86_64` in the url.

eTo get the fips build for `centos-7.9` the url would be `https://packages.d2iq.com/dkp/containerd/containerd-1.4.13-d2iq.1-centos-7.9-x86_64_fips.tar.gz`

The following OS's have containerd fips builds:

- `centos-7.9`
- `ol-7.9`
- `rhel-7.9`
- `rhel-8.2`
- `rhel-8.4`

8. Export the following environment variables, ensuring that all control plane and worker nodes are included:

```
export CONTROL_PLANE_1_ADDRESS="<control-plane-address-1>"
export CONTROL_PLANE_2_ADDRESS="<control-plane-address-2>"
export CONTROL_PLANE_3_ADDRESS="<control-plane-address-3>"
export WORKER_1_ADDRESS="<worker-address-1>"
export WORKER_2_ADDRESS="<worker-address-2>"
export WORKER_3_ADDRESS="<worker-address-3>"
export WORKER_4_ADDRESS="<worker-address-4>"
export SSH_USER="<ssh-user>"
export SSH_PRIVATE_KEY_FILE="<private key file>"
```

`SSH_PRIVATE_KEY_FILE` must be either the name of the SSH private key file in your working directory or an absolute path to the file in your user's home directory.

9. Generate an `inventory.yaml` to be used with `konvoy-image upload` in the next step:

```
cat <<EOF > inventory.yaml
all:
  vars:
    ansible_user: $SSH_USER
    ansible_port: 22
    ansible_ssh_private_key_file: $SSH_PRIVATE_KEY_FILE
  hosts:
    $CONTROL_PLANE_1_ADDRESS:
      ansible_host: $CONTROL_PLANE_1_ADDRESS
    $CONTROL_PLANE_2_ADDRESS:
      ansible_host: $CONTROL_PLANE_2_ADDRESS
    $CONTROL_PLANE_3_ADDRESS:
      ansible_host: $CONTROL_PLANE_3_ADDRESS
    $WORKER_1_ADDRESS:
      ansible_host: $WORKER_1_ADDRESS
    $WORKER_2_ADDRESS:
      ansible_host: $WORKER_2_ADDRESS
    $WORKER_3_ADDRESS:
      ansible_host: $WORKER_3_ADDRESS
    $WORKER_4_ADDRESS:
```

```
      ansible_host: $WORKER_4_ADDRESS
EOF
```

10. Upload the artifacts onto cluster hosts with the following command:

```
konvoy-image upload artifacts --container-images-dir=./artifacts/images/ \
              --os-packages-bundle=./artifacts/"$VERSION"_"$BUNDLE_OS".tar.gz \
              --pip-packages-bundle=./artifacts/pip-packages.tar.gz \
              --containerd-bundle=artifacts/containerd-1.4.13-d2iq.1-"$CONTAINE
RD_OS".tar.gz
```

Use the --overrides flag and reference either fips.yaml or offline-fips.yaml manifests located in the overrides directory[188] or see these pages in the documentation:

- FIPS Overrides(see page 331)
- Create FIPS 140 Images(see page 339)

Seed your docker registry

Before creating a Kubernetes cluster you must have the required images in a local docker registry. This registry must be accessible from both the bastion machine and the machines that will be created for the Kubernetes cluster.

1. Download the images bundle:

```
curl -o konvoy-image-bundle.tar -O downloads.d2iq.com/dkp/v2.3.3/
konvoy_image_bundle_v2.3.3_linux_amd64.tar
```

2. Place the bundle in a location where you can load and push the images to your private docker registry.
3. Set an environment variable with your registry address:

```
export DOCKER_REGISTRY_ADDRESS=<registry-address>:<registry-port>
```

4. Run the following command to load the air-gapped image bundle into your private Docker registry:

```
dkp push image-bundle --image-bundle konvoy-image-bundle.tar.gz --to-registry
$DOCKER_REGISTRY_ADDRESS
```

It may take a while to push all the images to your image registry, depending on the performance of the network between the machine you are running the script on and the Docker registry.

Then begin creating the bootstrap cluster(see page 229).

This Docker image includes code from the MinIO Project (“MinIO”), which is © 2015-2021 MinIO, Inc. MinIO is made available subject to the terms and conditions of the GNU Affero General Public License 3.0. The complete source code for the versions of MinIO packaged with DKP/Kommander/Konvoy 2.2.1 are available at these URLs: https://

188 https://github.com/mesosphere/konvoy-image-builder/tree/main/overrides

github.com/minio/minio/tree/RELEASE.2022-02-24T22-12-01Z https://github.com/minio/minio/tree/RELEASE.2021-02-14T04-01-33Z

For a full list of attributed 3rd party software, see D2IQ Legal[189].

6.5.4 Pre-provisioned Bootstrap

Bootstrap a kind cluster

A bootstrap cluster refers to a special type of local Kubernetes cluster used to bootstrap other clusters. The bootstrap cluster is required because the controllers that create other Kubernetes clusters require a Kubernetes cluster to run.

Konvoy deploys all cluster lifecycle services to a bootstrap cluster, which deploys a workload cluster. When the workload cluster is ready, move the cluster lifecycle services to the workload cluster. The workload cluster then manages its own lifecycle.

6.5.4.1 Bootstrap a kind cluster and CAPI controllers

Use the following command to create a bootstrap cluster:

```
dkp create bootstrap
```

```
✓ Creating a bootstrap cluster
✓ Initializing new CAPI components
```

When the bootstrap cluster is up, create the correct secrets and overrides(see page 229).

6.5.5 Pre-provisioned Create Necessary Secrets and Overrides

6.5.5.1 Create necessary secrets and overrides for pre-provisioned clusters

DKP requires SSH access to your infrastructure with superuser privileges. You must provide an unencrypted SSH private key to DKP.

Populate this key and create the required secret, on your bootstrap cluster using the following procedure.

6.5.5.2 Name your cluster

Give your cluster a unique name suitable for your environment.

Set the environment variable to be used throughout this procedure:

```
export CLUSTER_NAME=preprovisioned-example
```

189 https://d2iq.com/legal/3rd

6.5.5.3 Create a unique cluster name

(Optional) If you want to create a unique cluster name, use this command. This creates a unique name every time you run it, so use it carefully.

```
export CLUSTER_NAME=preprovisioned-example-$(LC_CTYPE=C tr -dc 'a-z0-9' </dev/urandom
| fold -w 5 | head -n1)
echo $CLUSTER_NAME
```

```
preprovisioned-example-pf4a3
```

6.5.5.4 Create a secret

Create a secret that contains the SSH key with these commands:

```
export SSH_PRIVATE_KEY_FILE=<path-to-ssh-private-key>
```

```
export SSH_PRIVATE_KEY_SECRET_NAME=${CLUSTER_NAME}-ssh-key
```

```
kubectl create secret generic ${SSH_PRIVATE_KEY_SECRET_NAME} --from-file=ssh-
privatekey=${SSH_PRIVATE_KEY_FILE}
kubectl label secret ${SSH_PRIVATE_KEY_SECRET_NAME} clusterctl.cluster.x-k8s.io/move=
```

```
secret/preprovisioned-example-ssh-key created
secret/preprovisioned-example-ssh-key labeled
```

6.5.5.5 Create overrides

If your pre-provisioned machines have override(see page 330) files, you must create a secret that includes all of the overrides you want to provide, in one file. For example, if you want to provide an override with Docker credentials and a different source for EPEL on a CentOS7 machine, you can create a file like this:

```
cat > overrides.yaml << EOF
image_registries_with_auth:
- host: "registry-1.docker.io"
  username: "<my-username>"
  password: "<my-password>"
  auth: ""
  identityToken: ""
```

```
epel_centos_7_rpm: https://my-rpm-repostory.org/epel/epel-release-latest-7.noarch.rpm
EOF
```

You can then create the related secret by running the following command:

```
kubectl create secret generic ${CLUSTER_NAME}-user-overrides --from-
file=overrides.yaml=overrides.yaml
kubectl label secret ${CLUSTER_NAME}-user-overrides clusterctl.cluster.x-k8s.io/move=
```

```
secret/preprovisioned-example-user-overrides-md-1 created
secret/preprovisioned-example-user-overrides-md-1 labeled
```

When using Oracle 7 OS, you may wish to to deploy the RHCK kernel instead of the default UEK kernel. To do so, add the following text to your overrides.yaml:

```
cat > overrides.yaml << EOF
---
oracle_kernel: RHCK
EOF
```

You can then create the related secret by running the following command:

```
kubectl create secret generic ${CLUSTER_NAME}-user-overrides --from-
file=overrides.yaml=overrides.yaml
kubectl label secret ${CLUSTER_NAME}-user-overrides clusterctl.cluster.x-k8s.io/move=
```

When this step is complete, define the infrastructure nodes and partitions(see page 231).

6.5.6 Pre-provisioned Define Infrastructure

Define the cluster hosts and infrastructure

Konvoy needs to know how to access your cluster hosts. This is done using inventory resources. For initial cluster creation, you must define a control-plane and at least one worker pool.

6.5.6.1 Define your infrastructure

1. Export the following environment variables, ensuring that all control plane and worker nodes are included:

```
export CONTROL_PLANE_1_ADDRESS="<control-plane-address-1>"
export CONTROL_PLANE_2_ADDRESS="<control-plane-address-2>"
export CONTROL_PLANE_3_ADDRESS="<control-plane-address-3>"
export WORKER_1_ADDRESS="<worker-address-1>"
export WORKER_2_ADDRESS="<worker-address-2>"
export WORKER_3_ADDRESS="<worker-address-3>"
```

```
export WORKER_4_ADDRESS="<worker-address-4>"
export SSH_USER="<ssh-user>"
export SSH_PRIVATE_KEY_SECRET_NAME="$CLUSTER_NAME-ssh-key"
```

2. Use the following template to help you define your infrastructure. The environment variables that you set in the previous step automatically replace the variable names when the file is created.

```
cat <<EOF > preprovisioned_inventory.yaml
---
apiVersion: infrastructure.cluster.konvoy.d2iq.io/v1alpha1
kind: PreprovisionedInventory
metadata:
  name: $CLUSTER_NAME-control-plane
  namespace: default
  labels:
    cluster.x-k8s.io/cluster-name: $CLUSTER_NAME
    clusterctl.cluster.x-k8s.io/move: ""
spec:
  hosts:
    # Create as many of these as needed to match your infrastructure
    # Note that the command line parameter --control-plane-replicas determines
how many control plane nodes will actually be used.
    #
    - address: $CONTROL_PLANE_1_ADDRESS
    - address: $CONTROL_PLANE_2_ADDRESS
    - address: $CONTROL_PLANE_3_ADDRESS
  sshConfig:
    port: 22
    # This is the username used to connect to your infrastructure. This user
must be root or
    # have the ability to use sudo without a password
    user: $SSH_USER
    privateKeyRef:
      # This is the name of the secret you created in the previous step. It
must exist in the same
      # namespace as this inventory object.
      name: $SSH_PRIVATE_KEY_SECRET_NAME
      namespace: default
---
apiVersion: infrastructure.cluster.konvoy.d2iq.io/v1alpha1
kind: PreprovisionedInventory
metadata:
  name: $CLUSTER_NAME-md-0
  namespace: default
  labels:
    cluster.x-k8s.io/cluster-name: $CLUSTER_NAME
    clusterctl.cluster.x-k8s.io/move: ""
spec:
  hosts:
    - address: $WORKER_1_ADDRESS
    - address: $WORKER_2_ADDRESS
```

```
    - address: $WORKER_3_ADDRESS
    - address: $WORKER_4_ADDRESS
  sshConfig:
    port: 22
    user: $SSH_USER
    privateKeyRef:
      name: $SSH_PRIVATE_KEY_SECRET_NAME
      namespace: default
EOF
```

3. Apply the new infrastructure file with the command:

```
envsubst < preprovisioned_inventory.yaml | kubectl apply -f -
```

```
preprovisionedinventory.infrastructure.cluster.konvoy.d2iq.io/preprovisioned-
example-control-plane created
preprovisionedinventory.infrastructure.cluster.konvoy.d2iq.io/preprovisioned-
example-md-0 created
```

After defining the infrastructure, define the control plane endpoint(see page 233).

6.5.7 Pre-provisioned Define Control Plane Endpoint

6.5.7.1 **Define the Control Plane Endpoint for your cluster**

A control plane should have three, five, or seven nodes, so it can remain available if one, two, or three nodes fail. A control plane with one node should not be used in production.

In addition, the control plane should have an endpoint that remains available if some nodes fail.

```
                       -------- cp1.example.com:6443
                       |
lb.example.com:6443 ---------- cp2.example.com:6443
                       |
                       -------- cp3.example.com:6443
```

In this example, the control plane endpoint host is `lb.example.com`, and the control plane endpoint port is `6443`. The control plane nodes are `cp1.example.com`, `cp2.example.com`, and `cp3.example.com`. The port of each API server is `6443`.

6.5.7.2 External load balancer

It is recommended that an external load balancer be the control plane endpoint. To distribute request load among the control plane machines, configure the load balancer to send requests to all the control plane machines. Configure the load balancer to send requests only to control plane machines that are responding to API requests.

6.5.7.3 Built-in virtual IP

If an external load balancer is not available, use the built-in virtual IP(see page 234). The virtual IP is *not* a load balancer; it does not distribute request load among the control plane machines. However, if the machine receiving requests does not respond to them, the virtual IP automatically moves to another machine.

6.5.7.4 Single-Node control plane

A control plane with one node can use its single node as the endpoint, so you will not require an external load balancer, or a the built-in virtual IP. However, do not use a single-node control plane configuration in production, as it is not possible to upgrade the control plane if you only have one node. This is due to the fact that CAPI does not perform in-situ upgrades, instead, it replaces machines. If you want to upgrade your cluster, use the default `--control-plane-replicas` setting of 3.

Modify Control Plane Audit logs settings using the information contained in the page Configuring the Control Plane(see page 343).

When the API server endpoints are defined, you can create the cluster(see page 234).

6.5.7.5 Known limitations

Be aware of these limitations in the current release of DKP.

The control plane endpoint port is also used as the API server port on each control plane machine. The default port is 6443. Before you create the cluster, ensure the port is available for use on each control plane machine.

6.5.8 Pre-provisioned Create new Cluster

6.5.8.1 Create a Kubernetes cluster using the infrastructure definition

Once you've defined the infrastructure(see page 231) and control plane endpoints(see page 233), you can proceed to creating the cluster by following these steps to create a new pre-provisioned cluster:

1. With the inventory, and the control plane endpoint defined, use the `dkp` binary to create a Konvoy cluster. The following command relies on the pre-provisioned cluster API infrastructure provider to initialize the Kubernetes control plane and worker nodes on the hosts defined in the inventory.
   NOTE: When specifying the `cluster-name`, you must use the same `cluster-name` as used when defining your inventory objects.
   NOTE: To increase Docker Hub's rate limit[190] use your Docker Hub credentials when creating the cluster, by setting the following flags `--registry-mirror-url=https://registry-1.docker.io --registry-mirror-username=<username> --registry-mirror-password=<password>` on the `dkp create cluster` command.

190 https://docs.docker.com/docker-hub/download-rate-limit/

```
dkp create cluster preprovisioned --cluster-name ${CLUSTER_NAME} --control-
plane-endpoint-host <control plane endpoint host> --control-plane-endpoint-port
<control plane endpoint port, if different than 6443>
```

```
Generating cluster resources
cluster.cluster.x-k8s.io/preprovisioned-example created
kubeadmcontrolplane.controlplane.cluster.x-k8s.io/preprovisioned-example-
control-plane created
preprovisionedcluster.infrastructure.cluster.konvoy.d2iq.io/preprovisioned-
example created
preprovisionedmachinetemplate.infrastructure.cluster.konvoy.d2iq.io/
preprovisioned-example-control-plane created
secret/preprovisioned-example-etcd-encryption-config created
machinedeployment.cluster.x-k8s.io/preprovisioned-example-md-0 created
preprovisionedmachinetemplate.infrastructure.cluster.konvoy.d2iq.io/
preprovisioned-example-md-0 created
kubeadmconfigtemplate.bootstrap.cluster.x-k8s.io/preprovisioned-example-md-0
 created
clusterresourceset.addons.cluster.x-k8s.io/calico-cni-installation-
preprovisioned-example created
configmap/calico-cni-installation-preprovisioned-example created
configmap/tigera-operator-preprovisioned-example created
clusterresourceset.addons.cluster.x-k8s.io/local-volume-provisioner-
preprovisioned-example created
configmap/local-volume-provisioner-preprovisioned-example created
clusterresourceset.addons.cluster.x-k8s.io/node-feature-discovery-
preprovisioned-example created
configmap/node-feature-discovery-preprovisioned-example created
clusterresourceset.addons.cluster.x-k8s.io/nvidia-feature-discovery-
preprovisioned-example created
configmap/nvidia-feature-discovery-preprovisioned-example created
clusterresourceset.addons.cluster.x-k8s.io/metallb-preprovisioned-example
created
configmap/metallb-installation-preprovisioned-example created
```

2. Use the wait command to monitor the cluster control-plane readiness:

```
kubectl wait --for=condition=ControlPlaneReady "clusters/${CLUSTER_NAME}" --
timeout=30m
```

```
cluster.cluster.x-k8s.io/preprovisioned-example condition met
```

(Optional) If you have overrides for your clusters(see page 229), you must specify the secret as part of the create cluster command.

NOTE: If these are not specified, the overrides for your nodes will not be applied.

```
dkp create cluster preprovisioned --cluster-name ${CLUSTER_NAME} --control-plane-
endpoint-host <control plane endpoint host> --control-plane-endpoint-port <control
plane endpoint port, if different than 6443> --override-secret-name=$CLUSTER_NAME-
user-overrides
```

NOTE: If your cluster is air-gapped or you have a local docker registry you must provide additional arguments when creating the cluster.

```
export DOCKER_REGISTRY_URL="<https/http>://<registry-address>:<registry-port>"
export DOCKER_REGISTRY_CA="<path to the CA on the bastion>"
export DOCKER_REGISTRY_USERNAME="<username>"
export DOCKER_REGISTRY_USERNAME="<password>"
```

- `DOCKER_REGISTRY_URL` : the address of an existing Docker registry accessible in the VPC that the new cluster nodes will be configured to use a mirror registry when pulling images.
- `DOCKER_REGISTRY_CA` : (optional) the path on the bastion machine to the Docker registry CA. Konvoy will configure the cluster nodes to trust this CA. This value is only needed if the registry is using a self-signed certificate and the AMIs are not already configured to trust this CA.
- `DOCKER_REGISTRY_USERNAME` : optional, set to a user that has pull access to this registry.
- `DOCKER_REGISTRY_PASSWORD` : optional if username is not set.

```
dkp create cluster preprovisioned --cluster-name ${CLUSTER_NAME} \
--control-plane-endpoint-host <control plane endpoint host> \
--control-plane-endpoint-port <control plane endpoint port, if different than 6443> \
--registry-mirror-url=${DOCKER_REGISTRY_URL} \
--registry-mirror-cacert=${DOCKER_REGISTRY_CA} \
--registry-mirror-username=${DOCKER_REGISTRY_USERNAME} \
--registry-mirror-password=${DOCKER_REGISTRY_PASSWORD}
```

4. Depending on the cluster size, it will take a few minutes to create. After the creation, use this command to get the Kubernetes kubeconfig for the new cluster and begin deploying workloads:

```
dkp get kubeconfig -c ${CLUSTER_NAME} > ${CLUSTER_NAME}.conf
```

6.5.8.2 Audit logs

To modify Control Plane Audit logs settings using the information contained in the page Configuring the Control Plane(see page 343).

6.5.8.3 Modify the Calico installation

Set the interface

Before exploring the new cluster, confirm your `calico` installation is correct. By default, Calico automatically detects the IP to use for each node using the `first-found` method[191]. This is not always appropriate for your particular nodes. In that case, you must modify Calico's configuration to use a different method. An alternative is to use the `interface` method by providing the interface ID to use. Follow the steps outlined in this section to modify Calico's configuration. In this example, all cluster nodes use `ens192` as the interface name.

Get the pods running on your cluster with this command:

```
kubectl get pods -A --kubeconfig ${CLUSTER_NAME}.conf
```

```
NAMESPACE                NAME
READY   STATUS            RESTARTS        AGE
calico-system            calico-kube-controllers-57fbd7bd59-vpn8b
1/1     Running           0               16m
calico-system            calico-node-5tbvl
1/1     Running           0               16m
calico-system            calico-node-nbdwd
1/1     Running           0               4m40s
calico-system            calico-node-twl6b
0/1     PodInitializing   0               9s
calico-system            calico-node-wktkh
1/1     Running           0               5m35s
calico-system            calico-typha-54f46b998d-52pt2
1/1     Running           0               16m
calico-system            calico-typha-54f46b998d-9tzb8
1/1     Running           0               4m31s
default                  cuda-vectoradd
0/1     Pending           0               0s
kube-system              coredns-78fcd69978-frwx4
1/1     Running           0               16m
kube-system              coredns-78fcd69978-kkf44
1/1     Running           0               16m
kube-system              etcd-ip-10-0-121-16.us-west-2.compute.internal
0/1     Running           0               8s
kube-system              etcd-ip-10-0-46-17.us-west-2.compute.internal
1/1     Running           1               16m
kube-system              etcd-ip-10-0-88-238.us-west-2.compute.internal
1/1     Running           1               5m35s
kube-system              kube-apiserver-ip-10-0-121-16.us-west-2.compute.internal
0/1     Running           6               7s
```

191 https://projectcalico.docs.tigera.io/reference/node/configuration#ip-autodetection-methods

```
kube-system                kube-apiserver-ip-10-0-46-17.us-west-2.compute.internal
1/1     Running            1               16m
kube-system                kube-apiserver-ip-10-0-88-238.us-west-2.compute.internal
1/1     Running            1               5m34s
kube-system                kube-controller-manager-ip-10-0-121-16.us-west-2.compute.int
ernal   0/1     Running           0               7s
kube-system                kube-controller-manager-ip-10-0-46-17.us-west-2.compute.inte
rnal    1/1     Running           1 (5m25s ago)   15m
kube-system                kube-controller-manager-ip-10-0-88-238.us-west-2.compute.int
ernal   1/1     Running           0               5m34s
kube-system                kube-proxy-gclmt
1/1     Running            0               16m
kube-system                kube-proxy-gptd4
1/1     Running            0               9s
kube-system                kube-proxy-mwkgl
1/1     Running            0               4m40s
kube-system                kube-proxy-zcqxd
1/1     Running            0               5m35s
kube-system                kube-scheduler-ip-10-0-121-16.us-west-2.compute.internal
0/1     Running            1               7s
kube-system                kube-scheduler-ip-10-0-46-17.us-west-2.compute.internal
1/1     Running            3 (5m25s ago)   16m
kube-system                kube-scheduler-ip-10-0-88-238.us-west-2.compute.internal
1/1     Running            1               5m34s
kube-system                local-volume-provisioner-2mv7z
1/1     Running            0               4m10s
kube-system                local-volume-provisioner-vdcrg
1/1     Running            0               4m53s
kube-system                local-volume-provisioner-wsjrt
1/1     Running            0               16m
node-feature-discovery     node-feature-discovery-master-84c67dcbb6-m78vr
1/1     Running            0               16m
node-feature-discovery     node-feature-discovery-worker-vpvpl
1/1     Running            0               4m10s
tigera-operator            tigera-operator-d499f5c8f-79dc4
1/1     Running            1 (5m24s ago)   16m
```

If a `calico-node` pod is not ready on your cluster, you must edit the `installation` file.

To edit the installation file, run the command:

```
kubectl edit installation default --kubeconfig ${CLUSTER_NAME}.conf
```

Change the value for `spec.calicoNetwork.nodeAddressAutodetectionV4` to `interface: ens192`, and save the file:

```
spec:
  calicoNetwork:
  ...
```

```
    nodeAddressAutodetectionV4:
      interface: ens192
```

After saving, you may need to delete the node feature discovery worker pod in the `node-feature-discovery` namespace if that pod has failed. After you delete it, Kubernetes replaces the pod as part of its normal reconciliation.

Change the encapsulation type

Calico can leverage different network encapsulation methods to route traffic for your workloads. Encapsulation is useful when running on top of an underlying network that is not aware of workload IPs. Common examples of this include:

- public cloud environments where you don't own the hardware
- AWS across VPC subnet boundaries
- environments where you cannot peer Calico over BGP to the underlay or easily configure static routes.

IPIP is the default encapsulation method.

To change the encapsulation, run the following command:

```
kubectl edit installation default --kubeconfig ${CLUSTER_NAME}.conf
```

Change the value for `spec.calicoNetwork.ipPools[0].encapsulation`

```
  spec:
  calicoNetwork:
    ipPools:
      - encapsulation: VXLAN
```

The supported values are "IPIPCrossSubnet", "IPIP", "VXLAN", "VXLANCrossSubnet", and "None".

VXLAN

VXLAN is a tunneling protocol that encapsulates layer 2 Ethernet frames in UDP packets, enabling you to create virtualized layer 2 subnets that span layer 3 networks. It has a slightly larger header than IP-in-IP which creates a slight reduction in performance over IP-in-IP.

IPIP

IP-in-IP is an IP tunneling protocol that encapsulates one IP packet in another IP packet. An outer packet header is added with the tunnel entrypoint and the tunnel exit point. The Calico implementation of this protocol uses BGP to determine the exit point making this protocol unusable on networks that don't pass BGP.

⚠ Be aware that switching encapsulation modes can cause disruption to in-progress connections. Plan accordingly.

For more information, see:

- Calico Overlay Networking[192]
- IP-in-IP RFC 2003[193]
- VXLAN RFC 7348[194]

6.5.8.4 Use the built-in Virtual IP

As explained in Define the Control Plane Endpoint(see page 233), we recommend using an external load balancer for the control plane endpoint, but provide a built-in virtual IP when an external load balancer is not available. The built-in virtual IP uses the kube-vip[195] project. To use the virtual IP, add these flags to the `create cluster` command:

Virtual IP Configuration	**Flag**
Network interface to use for Virtual IP. *Must exist on all control plane machines.*	`--virtual-ip-interface string`
IPv4 address. *Reserved for use by the cluster.*	`--control-plane-endpoint string`

Virtual IP example

```
dkp create cluster preprovisioned \
    --cluster-name ${CLUSTER_NAME} \
    --control-plane-endpoint-host 196.168.1.10 \
    --virtual-ip-interface eth1
```

Confirm that your Calico installation is correct(see page 237).

6.5.8.5 Provision on the Flatcar Linux OS

When provisioning onto the Flatcar Container Linux distribution, you must instruct the bootstrap cluster to make some changes related to the installation paths. To accomplish this, add the `--os-hint flatcar` flag to the above `create cluster` command.

Flatcar Linux example

```
dkp create cluster preprovisioned \
    --cluster-name ${CLUSTER_NAME} \
    --os-hint flatcar
```

Confirm that your Calico installation is correct(see page 237).

192 https://docs.projectcalico.org/networking/vxlan-ipip
193 https://datatracker.ietf.org/doc/html/rfc2003
194 https://datatracker.ietf.org/doc/html/rfc7348
195 https://kube-vip.chipzoller.dev/

For provisioning DKP on Flatcar, DKP configures cluster nodes to use Control Groups (cgroups) version 1[196]. In versions prior to Flatcar 3033.2.4, a restart is required in order to apply the changes to the kernel. For more information, refer to the Flatcar documentation[197].

6.5.8.6 Use an HTTP proxy

If you require HTTP proxy configurations, you can apply them during the `create` operation by adding the appropriate flags to the `create cluster` command:

Proxy configuration	Flag
HTTP proxy for control plane machines	`--control-plane-http-proxy string`
HTTPS proxy for control plane machines	`--control-plane-https-proxy string`
No Proxy list for control plane machines	`--control-plane-no-proxy strings`
HTTP proxy for worker machines	`--worker-http-proxy string`
HTTPS proxy for worker machines	`--worker-https-proxy string`
No Proxy list for worker machines	`--worker-no-proxy strings`

You must also add the same configuration as an override(see page 229). For more information, refer to this documentation(see page 337).

HTTP proxy example

To increase Docker Hub's rate limit[198] use your Docker Hub credentials when creating the cluster, by setting the following flag `--registry-mirror-url=https://registry-1.docker.io --registry-mirror-username=<username> --registry-mirror-password=<password>` on the `dkp create cluster` command.

196 https://www.kernel.org/doc/html/latest/admin-guide/cgroup-v1/cgroups.html#what-are-cgroups
197 https://www.flatcar.org/docs/latest/container-runtimes/switching-to-unified-cgroups/#starting-new-nodes-with-legacy-cgroups
198 https://docs.docker.com/docker-hub/download-rate-limit/

```
dkp create cluster preprovisioned \
    --cluster-name ${CLUSTER_NAME} \
    --control-plane-http-proxy http://proxy.example.com:8080 \
    --control-plane-https-proxy https://proxy.example.com:8080 \
    --control-plane-no-proxy
"127.0.0.1,10.96.0.0/12,192.168.0.0/16,kubernetes,kubernetes.default.svc,kubernetes.d
efault.svc.cluster,kubernetes.default.svc.cluster.local,.svc,.svc.cluster,.svc.cluste
r.local" \
    --worker-http-proxy http://proxy.example.com:8080 \
    --worker-https-proxy https://proxy.example.com:8080 \
    --worker-no-proxy
"127.0.0.1,10.96.0.0/12,192.168.0.0/16,kubernetes,kubernetes.default.svc,kubernetes.d
efault.svc.cluster,kubernetes.default.svc.cluster.local,.svc,.svc.cluster,.svc.cluste
r.local"
```

Confirm that your Calico installation is correct(see page 237).

Use an alternative mirror

To apply Docker registry configurations during the create operation, add the appropriate flags to the `create cluster` command:

Docker registry configuration	Flag
CA certificate chain to use while communicating with the registry mirror using TLS	`--registry-mirror-cacert file`
URL of a container registry to use as a mirror in the cluster	`--registry-mirror-url string`

This is useful when using an internal registry and when Internet access is not available (air-gapped installations).

When the cluster is up and running, you can deploy and test workloads.

Alternative mirror example

```
dkp create cluster preprovisioned \
    --cluster-name ${CLUSTER_NAME} \
    --registry-mirror-cacert /tmp/registry.pem \
    --registry-mirror-url https://registry.example.com
```

Confirm that your Calico installation is correct(see page 237).

6.5.8.7 Use alternate pod or service subnets

In Konvoy, the default pod subnet is 192.168.0.0/16, and the default service subnet is 10.96.0.0/12. If you wish to change the subnets you can do so with the following steps:

1. Generate the yaml manifests for the cluster using the `--dry-run` and `-o yaml` flags, along with the desired `dkp cluster create` command:

```
dkp create cluster preprovisioned --cluster-name ${CLUSTER_NAME} --control-
plane-endpoint-host <control plane endpoint host> --control-plane-endpoint-port
<control plane endpoint port, if different than 6443> --dry-run -o yaml >
cluster.yaml
```

2. To modify the service subnet, add or edit the `spec.clusterNetwork.services.cidrBlocks` field of the `Cluster` object:

```
kind: Cluster
spec:
  clusterNetwork:
    services:
      cidrBlocks:
      - 10.0.0.0/18
```

3. To modify the pod subnet, edit the `Cluster` and calico-cni `ConfigMap` resources:
   Cluster: Add or edit the `spec.clusterNetwork.pods.cidrBlocks` field:

```
kind: Cluster
spec:
  clusterNetwork:
    pods:
      cidrBlocks:
      - 172.16.0.0/16
```

   ConfigMap: Edit the `data."custom-resources.yaml".spec.calicoNetwork.ipPools.cidr` field with your desired pod subnet:

```
apiVersion: v1
data:
  custom-resources.yaml: |
    apiVersion: operator.tigera.io/v1
    kind: Installation
    metadata:
      name: default
    spec:
      # Configures Calico networking.
      calicoNetwork:
        # Note: The ipPools section cannot be modified post-install.
        ipPools:
        - blockSize: 26
          cidr: 172.16.0.0/16
kind: ConfigMap
```

```
metadata:
  name: calico-cni-<cluter-name>
```

When you provision the cluster, the configured pod and service subnets will be applied.

Confirm that your Calico installation is correct(see page 237).

When you complete this procedure, move on to Pre-provisioned make Cluster Self-managed(see page 244) to continue the process.

6.5.9 Pre-provisioned make Cluster Self-managed

Make the new Kubernetes cluster manage itself

Konvoy deploys all cluster lifecycle services to a bootstrap cluster, which then deploys a workload cluster. When the workload cluster is ready, move the cluster lifecycle services to the workload cluster, which is now self-managed. This guide describes how to make a workload cluster self-managed.

Before you start, make sure you have created a workload cluster, as described in Create the Cluster(see page 234).

6.5.9.1 Make the new Kubernetes cluster manage itself

If you have not already retrieved the kubeconfig after creating the cluster, use this command before proceeding: `dkp get kubeconfig -c ${CLUSTER_NAME} > ${CLUSTER_NAME}.conf`

1. Deploy cluster lifecycle services on the workload cluster:

```
dkp create capi-components --kubeconfig ${CLUSTER_NAME}.conf
```

```
✓ Initializing new CAPI components
```

2. Move the Cluster API objects from the bootstrap to the workload cluster:
The cluster lifecycle services on the workload cluster are ready, but the workload cluster configuration is on the bootstrap cluster. The `move` command moves the configuration, which takes the form of Cluster API Custom Resource objects, from the bootstrap to the workload cluster. This process is also called a Pivot[199].
First unset the kubeconfig and then move the CAPI:

```
unset KUBECONFIG
```

```
dkp move capi-resources --to-kubeconfig ${CLUSTER_NAME}.conf
```

199 https://cluster-api.sigs.k8s.io/reference/glossary.html?highlight=pivot#pivot

Output:

```
✓ Moving cluster resources
You can now view resources in the moved cluster by using the --kubeconfig flag
with kubectl. For example: kubectl --kubeconfig=preprovisioned-example.conf get
nodes
```

NOTE: To ensure only one set of cluster lifecycle services manages the workload cluster, Konvoy first pauses reconciliation of the objects on the bootstrap cluster, then creates the objects on the workload cluster. As Konvoy copies the objects, the cluster lifecycle services on the workload cluster reconcile the objects. The workload cluster becomes self-managed after Konvoy creates all the objects. If it fails, the `move` command can be safely retried.

3. Wait for the cluster control-plane to be ready:

```
kubectl --kubeconfig ${CLUSTER_NAME}.conf wait --for=condition=ControlPlaneRead
y "clusters/${CLUSTER_NAME}" --timeout=20m
```

```
cluster.cluster.x-k8s.io preprovisioned-example condition met
```

4. Use the cluster lifecycle services on the workload cluster to check the workload cluster status:
NOTE: After moving the cluster lifecycle services to the workload cluster, remember to use Konvoy with the workload cluster kubeconfig.

```
dkp describe cluster --kubeconfig ${CLUSTER_NAME}.conf -c ${CLUSTER_NAME}
```

```
NAME
READY  SEVERITY  REASON  SINCE  MESSAGE
Cluster/preprovisioned-example                                                 True
2m31s
├──ClusterInfrastructure - PreprovisionedCluster/preprovisioned-example
├──ControlPlane - KubeadmControlPlane/preprovisioned-example-control-plane
True                     2m31s
│ ├──Machine/preprovisioned-example-control-plane-6g6nr
True                     2m33s
│ ├──Machine/preprovisioned-example-control-plane-8lhcv
True                     2m33s
│ └──Machine/preprovisioned-example-control-plane-kk2kg
True                     2m33s
└──Workers
   └──MachineDeployment/preprovisioned-example-md-0
True                     2m34s
     └──Machine/preprovisioned-example-md-0-77f667cd9-tnctd
True                     2m33s
```

5. Remove the bootstrap cluster, as the workload cluster is now self-managed:

```
dkp delete bootstrap
```

```
✓ Deleting bootstrap cluster
```

Known limitations

Be aware of these limitations in the current release of Konvoy.

- DKP supports moving only one set of cluster objects from the bootstrap cluster to the workload cluster, or vice-versa.
- DKP only supports moving all namespaces in the cluster; DKP does not support migration of individual namespaces.

6.5.10 Pre-provisioned Configure MetalLB

6.5.10.1 Create a MetalLB configmap for your pre-provisioned infrastructure.

Choose one of the following two protocols you want to use to announce service IPs, either Layer 2 or BGP configurations.

6.5.10.2 Layer 2 configuration

Layer 2 mode is the simplest to configure: in many cases, you don't need any protocol-specific configuration, only IP addresses.

Layer 2 mode does not require the IPs to be bound to the network interfaces of your worker nodes. It works by responding to ARP requests on your local network directly, to give the machine's MAC address to clients.

For example, the following configuration gives MetalLB control over IPs from 192.168.1.240 to 192.168.1.250, and configures Layer 2 mode:

The following values are generic, enter your specific values into the fields where applicable.

```
cat << EOF > metallb-conf.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: metallb-system
  name: config
data:
  config: |
    address-pools:
    - name: default
      protocol: layer2
```

```
      addresses:
      - 192.168.1.240-192.168.1.250
EOF
```

Once complete, run the following `kubectl` command.

```
kubectl apply -f metallb-conf.yaml
```

6.5.10.3 BGP configuration

For a basic configuration featuring one BGP router and one IP address range, you need 4 pieces of information:

- The router IP address that MetalLB should connect to,
- The router's AS number,
- The AS number MetalLB should use,
- An IP address range expressed as a CIDR prefix.

As an example, if you want to give MetalLB the range 192.168.10.0/24 and AS number 64500, and connect it to a router at 10.0.0.1 with AS number 64501, your configuration will look like:

The following values are generic, enter your specific values into the fields where applicable.

```
cat << EOF > metallb-conf.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: metallb-system
  name: config
data:
  config: |
    peers:
    - peer-address: 10.0.0.1
      peer-asn: 64501
      my-asn: 64500
    address-pools:
    - name: default
      protocol: bgp
      addresses:
      - 192.168.10.0/24
EOF
```

Once complete, run the following `kubectl` command.

```
kubectl apply -f metallb-conf.yaml
```

6.5.11 Pre-provisioned Create and Delete Node Pools

Node pools are part of a cluster and managed as a group, and can be used to manage a group of machines using common properties. New default clusters created by Konvoy contain one node pool of worker nodes that have the same configuration.

You can create additional node pools for specialized hardware or other configurations. For example, if you want to tune your memory usage on a cluster where you need maximum memory for some machines and minimal memory on others, you could create a new node pool with those specific resource needs.

NOTE: Konvoy implements node pools using Cluster API MachineDeployments[200].

6.5.11.1 Create a node pool

Follow these steps:

1. Create an inventory object that has the same name as the node pool you're creating, and the details of the pre-provisioned machines that you want to add to it. For example, to create a node pool named `gpu-nodepool` an inventory named `gpu-nodepool` must be present in the same namespace:

```
apiVersion: infrastructure.cluster.konvoy.d2iq.io/v1alpha1
kind: PreprovisionedInventory
metadata:
  name: ${MY_NODEPOOL_NAME}
spec:
  hosts:
    - address: ${IP_OF_NODE}
  sshConfig:
    port: 22
    user: ${SSH_USERNAME}
    privateKeyRef:
      name: ${NAME_OF_SSH_SECRET}
      namespace: ${NAMESPACE_OF_SSH_SECRET}
```

2. (Optional) If your pre-provisioned machines have overrides(see page 330), you must create a secret that includes all of the overrides you want to provide in one file. Create an override secret using the instructions detailed on this page(see page 229).
3. Once the `PreprovisionedInventory` object and overrides are created, create a node pool:

```
dkp create nodepool preprovisioned -c ${MY_CLUSTER_NAME} ${MY_NODEPOOL_NAME} --
override-secret-name ${MY_OVERRIDE_SECRET}
```

200 https://cluster-api.sigs.k8s.io/developer/architecture/controllers/machine-deployment.html

6.5.11.2 Delete a node pool

Deleting a node pool deletes both the Kubernetes nodes and their underlying infrastructure. DKP drains all nodes prior to deletion and reschedules the pods running on those nodes.

To delete a node pool from a managed cluster, run the following command:

```
dkp delete nodepool ${NODEPOOL_NAME} --cluster-name=${CLUSTER_NAME}
```

The expected output is similar to the following example, indicating the `example` node pool is being deleted:

```
INFO[2021-07-28T17:14:26-07:00] Running nodepool delete command
Nodepool=example clusterName=d2iq-e2e-cluster-1 managementClusterKubeconfig=
namespace=default src="nodepool/delete.go:80"
```

Deleting an invalid node pool results in an output similar to this example command output:

```
dkp delete nodepool ${CLUSTER_NAME}-md-invalid --cluster-name=${CLUSTER_NAME}

INFO[2021-07-28T17:11:44-07:00] Running nodepool delete command
Nodepool=demo-cluster-md-invalid clusterName=d2iq-e2e-cluster-1
 managementClusterKubeconfig= namespace=default src="nodepool/delete.go:80"
Error: failed to get nodepool with name demo-cluster-md-invalid in namespace default
: failed to get nodepool with name demo-cluster-md-invalid in namespace default :
machinedeployments.cluster.x-k8s.io "demo-cluster-md-invalid" not found
```

6.5.12 Pre-provisioned Delete Cluster

NOTE: A self-managed workload cluster cannot delete itself. If your workload cluster is self-managed, you must create a bootstrap cluster and move the cluster lifecycle services to the bootstrap cluster before deleting the workload cluster.

If you did not make your workload cluster self-managed, as described in Make New Cluster Self-Managed(see page 244), see Delete the workload cluster(see page 249).

6.5.12.1 **Delete the Pre-provisioned cluster**

Follow these steps:

1. Create a bootstrap cluster:
   The bootstrap cluster will host the Cluster API controllers that reconcile the cluster objects marked for deletion:
   NOTE: To avoid using the wrong kubeconfig, the following steps use explicit kubeconfig paths and contexts.

```
dkp create bootstrap --kubeconfig $HOME/.kube/config
```

```
 ✓ Creating a bootstrap cluster
 ✓ Initializing new CAPI components
```

2. Move the Cluster API objects from the workload to the bootstrap cluster: The cluster lifecycle services on the bootstrap cluster are ready, but the workload cluster configuration is on the workload cluster. The `move` command moves the configuration, which takes the form of Cluster API Custom Resource objects, from the workload to the bootstrap cluster. This process is also called a Pivot[201].

```
dkp move capi-resources \
    --from-kubeconfig ${CLUSTER_NAME}.conf \
    --from-context ${CLUSTER_NAME}-admin@${CLUSTER_NAME} \
    --to-kubeconfig $HOME/.kube/config \
    --to-context kind-konvoy-capi-bootstrapper
```

```
✓ Moving cluster resources
You can now view resources in the moved cluster by using the --kubeconfig flag
with kubectl. For example: kubectl --kubeconfig $HOME/.kube/config get nodes
```

3. Use the cluster lifecycle services on the workload cluster to check the workload cluster status by running the following command:

```
dkp describe cluster --kubeconfig $HOME/.kube/config -c ${CLUSTER_NAME}
```

```
NAME
READY  SEVERITY  REASON  SINCE  MESSAGE
Cluster/preprovisioned-example                                              True
2m31s
├──ClusterInfrastructure - PreprovisionedCluster/preprovisioned-example
├──ControlPlane - KubeadmControlPlane/preprovisioned-example-control-plane
True                     2m31s
│ ├──Machine/preprovisioned-example-control-plane-6g6nr
True                     2m33s
│ ├──Machine/preprovisioned-example-control-plane-8lhcv
True                     2m33s
│ └──Machine/preprovisioned-example-control-plane-kk2kg
True                     2m33s
```

201 https://cluster-api.sigs.k8s.io/reference/glossary.html?highlight=pivot#pivot

```
└──Workers
   └──MachineDeployment/preprovisioned-example-md-0
True                     2m34s
     └──Machine/preprovisioned-example-md-0-77f667cd9-tnctd
True                     2m33s
```

NOTE: After moving the cluster lifecycle services to the workload cluster, remember to use dkp with the workload cluster kubeconfig.
Use dkp with the bootstrap cluster to delete the workload cluster.

4. Wait for the cluster control-plane to be ready. Run the command below and wait for the condition to be met:

```
kubectl --kubeconfig $HOME/.kube/config wait --for=condition=controlplaneready
"clusters/${CLUSTER_NAME}" --timeout=20m
```

```
cluster.cluster.x-k8s.io/preprovisioned-example condition met
```

Persistent Volumes (PVs) are not deleted automatically by design in order to preserve your data. However, they take up storage space if not deleted. You must delete PVs manually.

6.5.12.2 Delete the workload cluster

If you have a need to remove the Kubernetes cluster, such as for environment cleanup, use this command to delete the provisioned Kubernetes cluster:

```
dkp delete cluster --cluster-name=${CLUSTER_NAME}
```

```
✓ Deleting Services with type LoadBalancer for Cluster default/preprovisioned-
example
✓ Deleting ClusterResourceSets for Cluster default/preprovisioned-example
✓ Deleting cluster resources
✓ Waiting for cluster to be fully deleted
Deleted default/preprovisioned-example cluster
```

Delete the bootstrap cluster

After you have moved the workload resources back to a bootstrap cluster and deleted the workload cluster, you no longer need the bootstrap cluster. You can safely delete the bootstrap cluster with this command:

Delete the `kind` Kubernetes cluster:

```
dkp delete bootstrap --kubeconfig $HOME/.kube/config
```

```
✓ Deleting bootstrap cluster
```

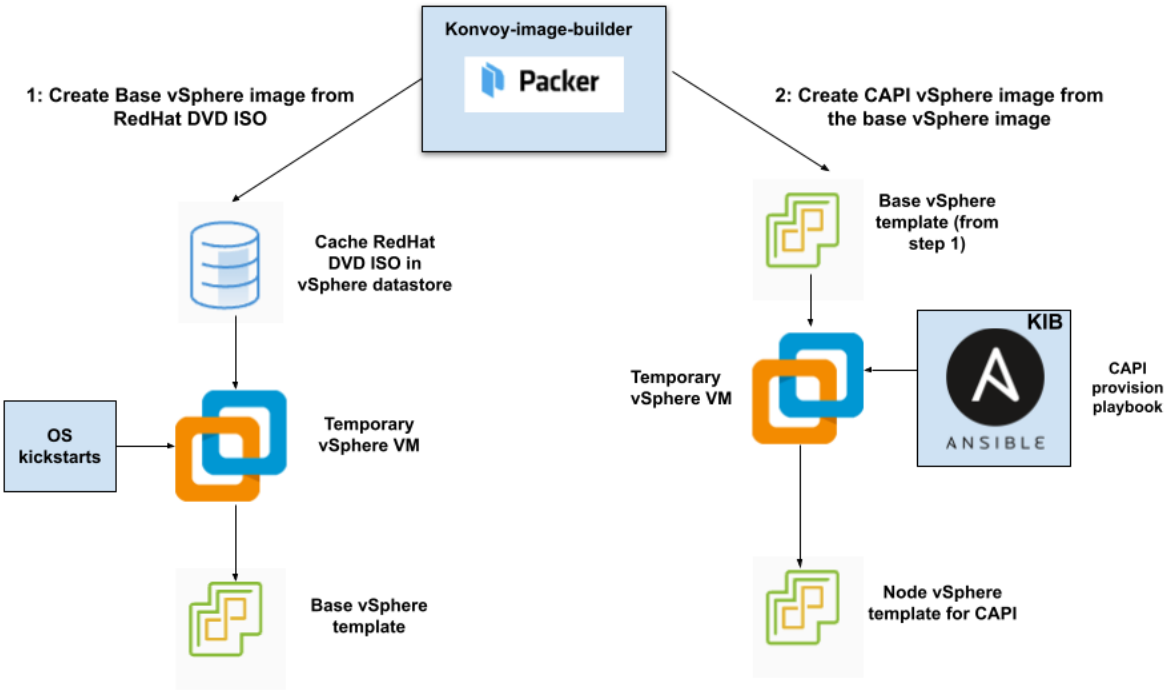
6.6 vSphere Infrastructure

6.6.1 Creating DKP clusters in a VMware vSphere environment

The overall process for configuring vSphere and DKP together includes the following steps:

1. Configure vSphere to provide the needed elements, described in the Prerequisites(see page 253).
2. Create a bastion VM host if you are using an air-gapped environment.
3. Create a base OS image.
4. Create a CAPI VM image that uses the base OS image and adds the needed Kubernetes cluster components.
5. Create a bootstrap cluster.
6. Create a new DKP cluster on vSphere.
7. Make the cluster self-managing.
8. Explore the cluster and perform other functions as needed.

This diagram illustrates the image creation process:



The workflow on the left shows the creation of a base OS image in the vCenter vSphere client using inputs from Packer. The workflow on the right shows how DKP uses that same base OS image to create CAPI-enabled VM images for your cluster.

After creating the base image, the DKP image builder uses it to create a CAPI-enabled vSphere template that includes the Kubernetes objects for the cluster. You can use that resulting template with the DKP `create cluster` command to create the VM nodes in your cluster directly on a vCenter server. From that point, you can use DKP to provision and manage your cluster.

To get started, fulfill the prerequisites(see page 253).

6.6.2 vSphere Prerequisites

6.6.2.1 Prepare your environment to run DKP with VMware vSphere

Fulfilling the prerequisites involves completing these areas:

- DKP prerequisites
- vSphere prerequisites

6.6.2.2 DKP Prerequisites

Before using DKP to create a vSphere cluster, verify that you have:

- An x86_64-based Linux® or macOS® machine.
- Download DKP binaries and Konvoy Image Builder (KIB)(see page 83) image bundle for Linux or macOS.
- Docker®[202] version 18.09.2 or later installed. You must have Docker installed on the host where the DKP Konvoy CLI runs. For example, if you are installing Konvoy on your laptop, ensure the laptop has a supported version of Docker.

On macOS, Docker runs in a virtual machine. Configure this virtual machine with at least 8GB of memory.

- kubectl[203] for interacting with the running cluster, installed on the host where the DKP Konvoy command line interface (CLI) runs.
- A valid VMware vSphere account with credentials configured.

6.6.2.3 VMware vSphere Prerequisites

Before installing, verify that your VMware vSphere Client environment[204] meets the following basic requirements:

- Access to a bastion VM, or other network connected host, running vSphere Client version v6.7.x with Update 3 or later version
    - You must be able to reach the vSphere API endpoint from where the Konvoy command line interface (CLI) runs.
- vSphere account with credentials configured - this account must have Administrator privileges.
- A RedHat® subscription with user name and password for downloading DVD ISOs

202 https://docs.docker.com/get-docker/
203 https://kubernetes.io/docs/tasks/tools/#kubectl
204 https://docs.vmware.com/en/VMware-vSphere/6.7/com.vmware.vsphere.vm_admin.doc/GUID-55238059-912E-411F-A0E9-A7A536972A91.html

- For air-gapped environments, a bastion VM host template(see page 274) with access to a configured Docker registry
- Valid vSphere values for the following:
    - vCenter API server URL
    - Datacenter name
    - Zone name that contains ESXi hosts[205] for your cluster's nodes
    - Datastore name for the shared storage resource to be used for the VMs in the cluster.
        - Use of PersistentVolumes in your cluster depends on Cloud Native Storage (CNS), available in vSphere v6.7.x with Update 3 and later versions. CNS depends on this shared Datastore's configuration.
    - Datastore URL from the datastore record for the shared datastore you want your cluster to use.
        - You need this URL value to ensure that the correct Datastore is used when DKP creates VMs for your cluster in vSphere.
    - Folder name
    - Base template name, such as base-rhel-8, or base-rhel-7
    - Name of a Virtual Network that has DHCP enabled for both air-gapped and non air-gapped environments
    - Resource Pools - at least one resource pool needed, with every host in the pool having access to shared storage, such as VSAN
        - Each host in the resource pool needs access to shared storage, such as NFS or VSAN, to make use of MachineDeployments and high-availability control planes.

6.6.2.4 Air-Gapped Registry Prerequisites

JFrog Artifactory

If you use **Jfrog Artifactory** or **Jfrog Container Registry**, you must update to a new version of the software. Any build newer than version **7.11** will work, as we have confirmed that older versions are not compatible.

Nexus Registry

If you use **Nexus Registry**, there is currently an issue that prevents usage with DKP 2.X and OCI Images. Support for OCI Images was added here in this publicly available Jira ticket:

https://issues.sonatype.org/browse/NEXUS-21087

A new issue was filed here determining that OCI image support is currently broken:

https://issues.sonatype.org/plugins/servlet/mobile#issue/NEXUS-27494

You can track this Jira link for status on a resolution for this issue.

Harbor Registry

Any newer version than **Harbor Registry v2.1.1**-5f52168e will support OCI images.

The next step is:

- for non air-gapped environments, create a base OS image(see page 275)
- for air-gapped environments, create and prepare a bastion VM(see page 274)

---

205 https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.esxi.install.doc/GUID-B2F01BF5-078A-4C7E-B505-5DFFED0B8C38.html

6.6.2.5 Minimum User Permissions

Create minimum required roles for provisioning and installing in vSphere

When a user needs permissions less than Admin, a role must be created. The process for configuring a vSphere role with the least permissions for provisioning nodes and installing includes the following steps:

1. Open a vSphere Client connection to the vCenter Server, described in the Prerequisites(see page 253).
2. Select Home > Administration > Roles > New.
3. Give the new role a name, then select these Privileges:
    - Datastore
        - Allocate space
    - Network
        - Assign network
    - Resource
        - Assign virtual machine to resource pool
    - Virtual machine
        - Change Configuration
        - Add new disk
        - Add or remove device
        - Advanced configuration
        - Change CPU count
        - Change Memory
        - Change Settings
        - Reload from path
    - Edit inventory
        - Create from existing
        - Remove
    - Interaction
        - Power off
        - Power on
    - Provisioning
        - Clone template
        - Deploy template
    - Session
        - ValidateSession
4. Add the permission at the highest level and set to propagate the permissions.

6.6.2.6 vSphere Storage Options

Explore storage options and considerations for using DKP with VMware vSphere

The vSphere Container Storage[206] plugin supports shared NFS, vNFS, and vSAN. You need to provision your storage options in vCenter prior to creating a CAPI image(see page 276) in DKP for use with vSphere.

DKP has integrated the CSI 2.x driver used in vSphere. When creating your DKP cluster, DKP uses whatever configuration you provide for the Datastore name. vSAN is not required. Using NFS can reduce the amount of tagging and permission granting required to configure your cluster.

---

206 https://docs.vmware.com/en/VMware-vSphere-Container-Storage-Plug-in/2.0/vmware-vsphere-csp-getting-started/GUID-74AF02D7-1562-48BD-A9FE-C81A53342AC3.html

6.6.3 Create a base OS image in the vSphere client

Creating a base OS image from DVD ISO files is a one-time process. Building a base OS image creates a base vSphere template in your vSphere environment. The base OS image is used by Konvoy Image Builder (see page 322)(KIB) to create a VM template to configure Kubernetes nodes by the DKP vSphere provider.

Depending on the cluster-api provider, a packer user configuration will vary. For vSphere, a password is required by default by packer[207]. This user should have administrator privileges. It is possible to configure a custom user and password when building the OS image, however, that requires the Konvoy Image Builder (KIB) configuration to be overridden(see page 330). DKP advises not to use static usernames and passwords for security reasons, but instead passwords should be generated and a minimum of 20 characters long.

While creating the base OS image, it is important to take into consideration the following elements:

1. Storage configuration: D2iQ recommends customizing disk partitions and not configuring a SWAP partition.
2. Network configuration: as KIB must download and install packages, activating the network is required.
3. Connect to Red Hat: if using RHEL, registering with Red Hat is required to configure software repositories and install software packages.
4. Software selection: D2iQ recommends choosing **Minimal Install**.
5. DKP recommends to install with the packages provided by the operating system package managers. Use the version that corresponds to the major version of your operating system.
6. DKP advises not to use static usernames and passwords for security reasons, but instead passwords should be generated and a minimum of 20 characters long.

The next step is to create a vSphere VM template(see page 256) that contains the CAPI and Kubernetes objects.

6.6.4 Create a vSphere VM Template

6.6.4.1 Prerequisites

You need to create a base OS image(see page 256) in vSphere before starting this procedure.

6.6.4.2 Create a vSphere template for your cluster from a base OS image

Using the base OS image created in a previous procedure, DKP creates the new vSphere template directly on the vCenter server.

1. Set the following vSphere environment variables on the bastion VM host:

   ```
   export VSPHERE_SERVER=your_vCenter_APIserver_URL
   export VSPHERE_USERNAME=your_vCenter_user_name
   export VSPHERE_PASSWORD=your_vCenter_password
   ```

2. Copy the base OS image file created in the vSphere Client to your desired location on the bastion VM host, and make a note of the path and file name.
3. Create an `image.yaml` file and add the following variables for vSphere. DKP uses this file and these variables as inputs in the next step.

---

207 https://github.com/mesosphere/konvoy-image-builder/blob/eecdd73ec9a1ca07c24962e2637bbab01a44d347/pkg/packer/manifests/vsphere/packer.json.tmpl#L15-L17

```
build_name: "vsphere-rhel-79"
packer_builder_type: "vsphere"
packer:
  cluster: "example_zone"
  datacenter: "example_datacenter"
  datastore: "example_datastore"
  folder: "example_folder"
  insecure_connection: "false"
  network: "example_network"
  resource_pool: "example_resource_pool"
  template: "example_base_OS_template_name"
  vsphere_guest_os_type: "example_rhel8_64Guest"
  guest_os_type: "example_rhel7-64"
  #goss params
  distribution: "example_RHEL"
  distribution_version: "example_7.9"
```

4. Create a vSphere VM template with the following command:

```
konvoy-image build path/to/image.yaml \
  --overrides /path/to/overrides/offline.yaml
```

The DKP image builder uses the values in `image.yaml` and the input base OS image to create a vSphere template that contains the required artifacts needed to create a Kubernetes cluster. Give the file a suitable name using this suggested naming convention: `creator-ova-vsphere-OS-ver-k8sver-unique_identifier`. As an example, the filename you create might resemble `dkp-ova-vsphere-rhel-84-1.21.6-1646938922`.
DKP creates the new vSphere template directly on the vCenter server.

Next, create a Kubernetes bootstrap cluster(see page 257) to enable creating your vSphere cluster and moving CAPI objects to it.

6.6.5 vSphere Bootstrap

6.6.5.1 Prepare to deploy Kubernetes clusters

To create Kubernetes clusters, Konvoy uses Cluster API[208] (CAPI) controllers which run on a Kubernetes cluster. To get started creating your vSphere cluster, you need a *bootstrap* cluster. By default, Konvoy creates a bootstrap cluster for you in a Docker container using the Kubernetes-in-Docker (KIND[209]) tool.

6.6.5.2 Prerequisites

Before you begin, you must:

208 https://cluster-api.sigs.k8s.io/
209 https://github.com/kubernetes-sigs/kind

- Ensure the `dkp` binary can be found in your $PATH.
- Complete the steps in Create a CAPI VM template(see page 256)

6.6.5.3 Bootstrap cluster lifecycle services

1. If an HTTP proxy is required for the bootstrap cluster, set the local `http_proxy`, `https_proxy`, and `no_proxy` environment variables. They are copied into the bootstrap cluster.
2. Create a bootstrap cluster:

   ```
   dkp create bootstrap --kubeconfig $HOME/.kube/config
   ```

   The output resembles this example:

   ```
   ✓ Creating a bootstrap cluster
   ✓ Initializing new CAPI components
   ```

   Konvoy creates a bootstrap cluster using KIND[210] as a library. Konvoy then deploys the following Cluster API[211] providers on the cluster:
   - Core Provider[212]
   - vSphere Infrastructure Provider[213]
   - Kubeadm Bootstrap Provider[214]
   - Kubeadm ControlPlane Provider[215]
3. Ensure that the CAPV controllers are present with the command:

   ```
   kubectl get pods -n capv-system
   ```

   The output resembles the following:

   ```
   NAME                                      READY   STATUS    RESTARTS   AGE
   capv-controller-manager-785c5978f-nnfns   1/1     Running   0          13h
   ```

4. Konvoy waits until the controller-manager and webhook deployments of these providers are ready. List these deployments using this command:

   ```
   kubectl get --all-namespaces deployments -l=clusterctl.cluster.x-k8s.i
   ```

210 https://github.com/kubernetes-sigs/kind
211 https://cluster-api.sigs.k8s.io/
212 https://github.com/kubernetes-sigs/cluster-api/tree/v0.3.20/
213 https://github.com/kubernetes-sigs/cluster-api-provider-vsphere
214 https://github.com/kubernetes-sigs/cluster-api/tree/v0.3.20/bootstrap/kubeadm
215 https://github.com/kubernetes-sigs/cluster-api/tree/v0.3.20/controlplane/kubeadm

```
NAMESPACE                           NAME
READY   UP-TO-DATE   AVAILABLE   AGE
capa-system                         capa-controller-manager
1/1     1            1           22m
capi-kubeadm-bootstrap-system       capi-kubeadm-bootstrap-controller-manager
1/1     1            1           22m
capi-kubeadm-control-plane-system   capi-kubeadm-control-plane-controller-
manager   1/1     1            1           22m
capi-system                         capi-controller-manager
1/1     1            1           22m
cappp-system                        cappp-controller-manager
1/1     1            1           22m
capv-system                         capv-controller-manager
1/1     1            1           22m
capz-system                         capz-controller-manager
1/1     1            1           22m
cert-manager                        cert-manager
1/1     1            1           22m
cert-manager                        cert-manager-cainjector
1/1     1            1           22m
cert-manager                        cert-manager-webhook
1/1     1            1           22m
```

When the bootstrap cluster is running, you are ready to Create a new vSphere cluster(see page 259).

6.6.6 Create new vSphere Cluster

6.6.6.1 Prerequisites

Before you begin, make sure you have created a vSphere Bootstrap(see page 257) cluster.

6.6.6.2 Name your cluster

1. Give your cluster a unique name suitable for your environment.
2. Set the CLUSTER_NAME environment variable with the command:

```
export CLUSTER_NAME=my-vsphere-cluster
```

6.6.6.3 Create a new vSphere Kubernetes cluster

Follow these steps:

1. Use the following command to set the environment variables for vSphere:

```
export VSPHERE_SERVER=example.vsphere.url
export VSPHERE_USERNAME=user@example.vsphere.url
export VSPHERE_PASSWORD=example_password
```

2. Ensure your vSphere credentials are up-to-date by refreshing the credentials with the command:

```
dkp update bootstrap credentials vsphere
```

3. Generate the Kubernetes cluster objects by copying and editing this command to include the correct values, including the VM template name you assigned in the previous procedure:

To increase Dockerhub's rate limit[216] use your Dockerhub credentials when creating the cluster, by setting the following flag `--registry-mirror-url=https://registry-1.docker.io --registry-mirror-username= --registry-mirror-password=` on the `dkp create cluster command`.

```
dkp create cluster vsphere \
  --cluster-name ${CLUSTER_NAME} \
  --network <NETWORK_NAME> \
  --control-plane-endpoint-host <xxx.yyy.zzz.000> \
  --data-center <DATACENTER_NAME> \
  --data-store <DATASTORE_NAME> \
  --folder <FOLDER_NAME> \
  --server <VCENTER_API_SERVER_UTR \
  --ssh-public-key-file <SSH_PUBLIC_KEY_FILE> \
  --resource-pool <RESOURE_POOL_NAME> \
  --virtual-ip-interface <ip_interface_name> \
  --vm-template <TEMPLATE_NAME>
```

4. (Optional) To configure the Control Plane and Worker nodes to use an HTTP proxy:

```
export CONTROL_PLANE_HTTP_PROXY=http://example.org:8080
export CONTROL_PLANE_HTTPS_PROXY=http://example.org:8080
export
CONTROL_PLANE_NO_PROXY=
"example.org,example.com,example.net,localhost,127.0.0.1,10.96.0.0/12,192.168.0.0/16,
kubernetes,kubernetes.default,kubernetes.default.svc,kubernetes.default.svc.cluster,k
ubernetes.default.svc.cluster.local,.svc,.svc.cluster,.svc.cluster.local,169.254.169.
254,.elb.amazonaws.com"

export WORKER_HTTP_PROXY=http://example.org:8080
```

216 https://docs.docker.com/docker-hub/download-rate-limit/

```
export WORKER_HTTPS_PROXY=http://example.org:8080
export
WORKER_NO_PROXY=
"example.org,example.com,example.net,localhost,127.0.0.1,10.96.0.0/12,192.168.0.0/16,
kubernetes,kubernetes.default,kubernetes.default.svc,kubernetes.default.svc.cluster,k
ubernetes.default.svc.cluster.local,.svc,.svc.cluster,.svc.cluster.local,169.254.169.
254,.elb.amazonaws.com"
```

- Replace `example.org,example.com,example.net` with you internal addresses
- `localhost` and `127.0.0.1` addresses should not use the proxy
- `10.96.0.0/12` is the default Kubernetes service subnet
- `192.168.0.0/16` is the default Kubernetes pod subnet
- `kubernetes,kubernetes.default,kubernetes.default.svc,kubernetes.default.svc.cluster,kubernetes.default.svc.cluster.local` is the internal Kubernetes kube-apiserver service
- `.svc,.svc.cluster,.svc.cluster.local` is the internal Kubernetes services
- `169.254.169.254` is the AWS metadata server
- `.elb.amazonaws.com` is for the worker nodes to allow them to communicate directly to the kube-apiserver ELB

5. (Optional) Create a Kubernetes cluster with HTTP proxy configured. This step assumes you did not already create a cluster in the previous steps:

To increase Dockerhub's rate limit[217] use your Dockerhub credentials when creating the cluster, by setting the following flag `--registry-mirror-url=https://registry-1.docker.io --registry-mirror-username= --registry-mirror-password=` on the `dkp create cluster command`.

```
dkp create cluster vsphere --cluster-name=${CLUSTER_NAME} \
--control-plane-http-proxy="${CONTROL_PLANE_HTTP_PROXY}" \
--control-plane-https-proxy="${CONTROL_PLANE_HTTPS_PROXY}" \
--control-plane-no-proxy="${CONTROL_PLANE_NO_PROXY}" \
--worker-http-proxy="${WORKER_HTTP_PROXY}" \
--worker-https-proxy="${WORKER_HTTPS_PROXY}" \
--worker-no-proxy="${WORKER_NO_PROXY}" \
--dry-run \
--output=yaml \
> ${CLUSTER_NAME}.yaml
```

217 https://docs.docker.com/docker-hub/download-rate-limit/

6. Inspect or edit the cluster objects:

> Familiarize yourself with Cluster API before editing the cluster objects as edits can prevent the cluster from deploying successfully.

The objects are Custom Resources[218] defined by Cluster API components, and they belong in three different categories:

1. Cluster
   A *Cluster* object has references to the infrastructure-specific and control plane objects. Because this is a vSphere cluster, there is an object that describes the infrastructure-specific cluster properties.
2. Control plane
   A *KubeadmControlPlane* object describes the control plane, which is the group of machines that run the Kubernetes control plane components, which include the etcd distributed database, the API server, the core controllers, and the scheduler. The object describes the configuration for these components. The object also has a reference to an infrastructure-specific object that describes the properties of all control plane machines. Here, it references an *vSphereMachineTemplate* object.
3. Node pool
   A node pool is a collection of machines with identical properties. For example, a cluster might have one node pool with large memory capacity, another node pool with GPU support. Each node pool is described by three objects: The MachinePool references an object that describes the configuration of Kubernetes components (for example, kubelet) deployed on each node pool machine, and an infrastructure-specific object that describes the properties of all node pool machines. Here, it references a *KubeadmConfigTemplate*, and a *vSphereMachineTemplate* object.

For in-depth documentation about the objects, read Concepts[219] in the Cluster API Book.

7. Modify control plane audit logs settings using the information contained in the page Configuring the Control Plane(see page 343).

8. Create the cluster from the objects.

```
kubectl apply -f ${CLUSTER_NAME}.yaml
```

```
cluster.cluster.x-k8s.io/vsphere-example created
cluster.infrastructure.cluster.x-k8s.io/vsphere-example created
kubeadmcontrolplane.controlplane.cluster.x-k8s.io/vsphere-example-control-plane
created
machinedeployment.cluster.x-k8s.io/vsphere-example-mp-0 created
kubeadmconfigtemplate.bootstrap.cluster.x-k8s.io/vsphere-example-mp-0 created
```

8. Use the `wait` command to monitor the cluster control-plane readiness:

218 https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/
219 https://cluster-api.sigs.k8s.io/user/concepts.html

```
kubectl wait --for=condition=ControlPlaneReady "clusters/${CLUSTER_NAME}" --
timeout=20m
```

```
cluster.cluster.x-k8s.io/${CLUSTER_NAME} condition met
```

The `READY` status becomes `True` after the cluster control-plane becomes Ready in one of the following steps.

After DKP creates the objects on the API server, the Cluster API controllers reconcile them, creating infrastructure and machines. As the controllers progress, they update the Status of each object.

9. Run the DKP describe command to monitor the current status of the cluster:

```
dkp describe cluster -c ${CLUSTER_NAME}
```

```
NAME                                                                READY  SEVERITY
REASON  SINCE  MESSAGE
Cluster/d2iq-e2e-cluster_name-1                                     True
13h
├──ClusterInfrastructure - VSphereCluster/d2iq-e2e-cluster_name-1    True
13h
├──ControlPlane - KubeadmControlPlane/d2iq-control-plane            True
13h
│  ├──Machine/d2iq--control-plane-7llgd                             True
13h
│  ├──Machine/d2iq--control-plane-vncbl                             True
13h
│  └──Machine/d2iq--control-plane-wbgrm                             True
13h
└──Workers
   └──MachineDeployment/d2iq--md-0                                  True
13h
   ├──Machine/d2iq--md-0-74c849dc8c-67rv4                           True
13h
   ├──Machine/d2iq--md-0-74c849dc8c-n2skc                           True
13h
   ├──Machine/d2iq--md-0-74c849dc8c-nkftv                           True
13h
   └──Machine/d2iq--md-0-74c849dc8c-sqklv                           True
13h
```

10. Check all machines has `NODE_NAME` assigned

```
kubectl get machines
```

The output appears similar to the following:

```
NAME                                        CLUSTER              NODENAME
PROVIDERID                                        PHASE     AGE   VERSION
d2iq-e2e-cluster-1-control-plane-7llgd     d2iq-e2e-cluster-1   d2iq-e2e-cluster-1-
control-plane-7llgd     vsphere://421638e2-e776-9af6-f683-5e105de5da5a   Running
13h   v1.22.8
d2iq-e2e-cluster-1-control-plane-vncbl     d2iq-e2e-cluster-1   d2iq-e2e-cluster-1-
control-plane-vncbl     vsphere://42168835-7fef-95c4-3652-ebcad3e10d36   Running
13h   v1.22.8
d2iq-e2e-cluster-1-control-plane-wbgrm     d2iq-e2e-cluster-1   d2iq-e2e-cluster-1-
control-plane-wbgrm     vsphere://421642df-afc4-b6c2-9e61-5b86e7c37eac   Running
13h   v1.22.8
d2iq-e2e-cluster-1-md-0-74c849dc8c-67rv4   d2iq-e2e-cluster-1   d2iq-e2e-cluster-1-
md-0-74c849dc8c-67rv4   vsphere://4216f467-8483-73cb-a8b6-8d6a4a71e4b4   Running
14h   v1.22.8
d2iq-e2e-cluster-1-md-0-74c849dc8c-n2skc   d2iq-e2e-cluster-1   d2iq-e2e-cluster-1-
md-0-74c849dc8c-n2skc   vsphere://42161cde-9904-4dd2-7a3e-cdfc7655f090   Running
14h   v1.22.8
d2iq-e2e-cluster-1-md-0-74c849dc8c-nkftv   d2iq-e2e-cluster-1   d2iq-e2e-cluster-1-
md-0-74c849dc8c-nkftv   vsphere://42163a0d-eb8d-b5a6-82d5-188e24817c00   Running
14h   v1.22.8
d2iq-e2e-cluster-1-md-0-74c849dc8c-sqklv   d2iq-e2e-cluster-1   d2iq-e2e-cluster-1-
md-0-74c849dc8c-sqklv   vsphere://42161dff-92a5-6da9-7ac1-e987e2c8fed2   Running
14h   v1.22.8
```

11. Verify that the kubeadm control plane is ready with the command

```
kubectl get kubeadmcontrolplane
```

The output appears similar to the following:

```
NAME                               CLUSTER              INITIALIZED   API SERVER
AVAILABLE   REPLICAS   READY   UPDATED   UNAVAILABLE   AGE   VERSION
d2iq-e2e-cluster-1-control-plane   d2iq-e2e-cluster-1   true          true
3          3       3         0             14h   v1.22.8
```

12. Describe the kubeadm control plane and check its status and events with the command:

```
kubectl describe kubeadmcontrolplane
```

13. As they progress, the controllers also create Events, which you can list using the command:

```
kubectl get events | grep ${CLUSTER_NAME}
```

For brevity, this example uses `grep`. You can also use separate commands to get Events for specific objects, such as `kubectl get events --field-selector involvedObject.kind="VSphereCluster"` and `kubectl get events --field-selector involvedObject.kind="VSphereMachine"`.

6.6.6.4 Known limitations

Be aware of these limitations in the current release of DKP Konvoy.

- The DKP Konvoy version used to create a bootstrap cluster must match the DKP Konvoy version used to create a workload cluster.
- DKP Konvoy supports deploying one workload cluster.
- DKP Konvoy generates a set of objects for one Node Pool.
- DKP Konvoy does not validate edits to cluster objects.

The optional next step is to make the vSphere cluster self-managing(see page 265). The step is optional because, as an example, if you are using an existing, self-managed cluster to create a managed cluster, you would not want the managed cluster to be self-managed.

6.6.7 Make vSphere Cluster Self-Managed

Konvoy deploys all cluster lifecycle services to a bootstrap cluster, which then deploys a workload cluster. When the workload cluster is ready, move the cluster lifecycle services to the workload cluster, which makes the workload cluster self-managed. This section describes how to make a workload cluster self-managed.

Before starting, ensure you create a workload cluster as described in Create a New Cluster(see page 259).

6.6.7.1 Make the new Kubernetes cluster manage itself

Follow these steps:

1. Deploy cluster lifecycle services on the workload cluster:

   ```
   dkp create capi-components --kubeconfig ${CLUSTER_NAME}.conf
   ```

```
✓ Initializing new CAPI components
```

2. Move the Cluster API objects from the bootstrap to the workload cluster:
   The cluster lifecycle services on the workload cluster are ready, but the workload cluster configuration is on the bootstrap cluster. The `move` command moves the configuration, which takes the form of Cluster API Custom Resource objects, from the bootstrap to the workload cluster. This process is also called a Pivot[220].

```
dkp move --to-kubeconfig ${CLUSTER_NAME}.conf
```

```
INFO[2021-08-11T12:09:36-07:00] Pivot operation complete.
src="move/move.go:154"
INFO[2021-08-11T12:09:36-07:00] You can now view resources in the moved cluster
by using the --kubeconfig flag with kubectl. For example: kubectl --
kubeconfig=/home/clusteradmin/.kube/config get nodes  src="move/move.go:155"
```

   NOTE: To ensure only one set of cluster lifecycle services manages the workload cluster, Konvoy first pauses reconciliation of the objects on the bootstrap cluster, then creates the objects on the workload cluster. As Konvoy copies the objects, the cluster lifecycle services on the workload cluster reconcile the objects. The workload cluster becomes self-managed after Konvoy creates all the objects. If it fails, the move command can be safely retried.
3. Wait for the cluster control-plane to be ready:

```
kubectl --kubeconfig ${CLUSTER_NAME}.conf wait --for=condition=ControlPlaneRead
y "clusters/${CLUSTER_NAME}" --timeout=20m
```

```
d2iq-e2e-cluster-1/vsphere-example condition met
```

4. Use the cluster lifecycle services on the workload cluster to check the workload cluster status:
   NOTE: After moving the cluster lifecycle services to the workload cluster, remember to use Konvoy with the workload cluster kubeconfig.

```
dkp describe cluster --kubeconfig ${CLUSTER_NAME}.conf -c ${CLUSTER_NAME}
```

```
NAME                                                           READY
SEVERITY  REASON  SINCE  MESSAGE
```

220 https://cluster-api.sigs.k8s.io/reference/glossary.html?highlight=pivot#pivot

```
Cluster/d2iq-e2e-cluster_name-1                                    True
13h
├──ClusterInfrastructure - VSphereCluster/d2iq-e2e-cluster_name-1    True
13h
├──ControlPlane - KubeadmControlPlane/d2iq-control-plane            True
13h
│  ├──Machine/d2iq--control-plane-7llgd                             True
13h
│  ├──Machine/d2iq--control-plane-vncbl                             True
13h
│  └──Machine/d2iq--control-plane-wbgrm                             True
13h
└──Workers
   └──MachineDeployment/d2iq--md-0                                  True
13h
   ├──Machine/d2iq--md-0-74c849dc8c-67rv4                           True
13h
   ├──Machine/d2iq--md-0-74c849dc8c-n2skc                           True
13h
   ├──Machine/d2iq--md-0-74c849dc8c-nkftv                           True
13h
   └──Machine/d2iq--md-0-74c849dc8c-sqklv                           True
13h
```

5. Remove the bootstrap cluster, if desired, as the workload cluster is now self-managed:

```
dkp delete bootstrap
```

```
INFO[2022-03-30T17:53:36-07:00] Deleting bootstrap cluster
src="bootstrap/bootstrap.go:182"
```

6.6.7.2 Known limitations

Be aware of these limitations in the current release of DKP Konvoy.

- Before making a workload cluster self-managed, be sure that its control plane nodes have sufficient permissions for running Cluster API controllers.
- DKP Konvoy supports moving only one set of cluster objects from the bootstrap cluster to the workload cluster, or vice-versa.
- DKP Konvoy only supports moving all namespaces in the cluster; DKP does not support migration of individual namespaces.

Next, you can explore the new cluster(see page 268) or explore the new air-gapped cluster(see page 281).

6.6.8 Explore a vSphere Cluster

This guide explains how to use the command line interface to interact with your newly-deployed Kubernetes cluster.

Before you start, make sure you have created a workload cluster(see page 259) and, if needed, that you have made the cluster self-managing(see page 265).

6.6.8.1 Get the kubeconfig file for the new Kubernetes cluster

1. Get a kubeconfig file for the workload cluster:
   When the workload cluster is created, the cluster lifecycle services generate a kubeconfig file for the workload cluster, and write it to a *Secret*. The kubeconfig file is scoped to the cluster Administrator. Get the kubeconfig from the *Secret*, and write it to a file using this command:

   ```
   dkp get kubeconfig -c ${CLUSTER_NAME} > ${CLUSTER_NAME}.conf
   ```

6.6.8.2 Create a StorageClass with a vSphere datastore

Follow these steps:

1. Access the Datastore tab in the vSphere client and select a datastore by name.
2. Copy the URL of that datastore from the information dialog that displays.
3. Return to the DKP CLI, and delete the existing `StorageClass` with the command:

   ```
   kubectl delete storageclass vsphere-raw-block-sc
   ```

4. Run the following command to create a new StorageClass, supplying the correct values for your environment:

   ```
   cat <<EOF > vsphere-raw-block-sc.yaml
   kind: StorageClass
   apiVersion: storage.k8s.io/v1
   metadata:
     annotations:
       storageclass.kubernetes.io/is-default-class: "true"
     name: vsphere-raw-block-sc
   provisioner: csi.vsphere.vmware.com
   parameters:
     datastoreurl: "<url>"
   volumeBindingMode: WaitForFirstConsumer
   EOF
   ```

6.6.8.3 Explore nodes and pods in the new cluster

1. List the nodes using this command:

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf get nodes
```

NOTE: It may take a few minutes for the Status to move to `Ready` while the Pod network is deployed. The Node's Status should change to Ready soon after the `calico-node` DaemonSet Pods are Ready.
The output resembles the following example:

```
NAME                                        STATUS   ROLES                  AGE
VERSION
d2iq-e2e-cluster-1-control-plane-7llgd      Ready    control-plane,master   20h
v1.24.6
d2iq-e2e-cluster-1-control-plane-vncbl      Ready    control-plane,master   19h
v1.24.6
d2iq-e2e-cluster-1-control-plane-wbgrm      Ready    control-plane,master   19h
v1.24.6
d2iq-e2e-cluster-1-md-0-74c849dc8c-67rv4    Ready    <none>                 19h
v1.24.6
d2iq-e2e-cluster-1-md-0-74c849dc8c-n2skc    Ready    <none>                 19h
v1.24.6
d2iq-e2e-cluster-1-md-0-74c849dc8c-nkftv    Ready    <none>                 19h
v1.24.6
d2iq-e2e-cluster-1-md-0-74c849dc8c-sqklv    Ready    <none>                 19h
v1.24.6
```

2. List the pods using this command:

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf get --all-namespaces pods
```

The output resembles the following example:

```
NAMESPACE              NAME
READY   STATUS     RESTARTS      AGE
calico-system          calico-kube-controllers-57fbd7bd59-qqd96
1/1     Running    0             20h
calico-system          calico-node-2m524
1/1     Running    3 (19h ago)   19h
calico-system          calico-node-bbhg5
1/1     Running    0             20h
calico-system          calico-node-cc5lf
1/1     Running    2 (19h ago)   19h
```

```
calico-system            calico-node-cwg7x
1/1     Running    1 (19h ago)   19h
calico-system            calico-node-d59hn
1/1     Running    1 (19h ago)   19h
calico-system            calico-node-qmmcz
1/1     Running    0             19h
calico-system            calico-node-wdqhx
1/1     Running    0             19h
calico-system            calico-typha-655489d8cc-b5jnt
1/1     Running    0             20h
calico-system            calico-typha-655489d8cc-q92x9
1/1     Running    0             19h
calico-system            calico-typha-655489d8cc-vjlkx
1/1     Running    0             19h
kube-system              cluster-autoscaler-68c759fbf6-7d2ck
0/1     Init:0/1   0             20h
kube-system              coredns-78fcd69978-qn4qt
1/1     Running    0             20h
kube-system              coredns-78fcd69978-wqpmg
1/1     Running    0             20h
kube-system              etcd-d2iq-e2e-cluster-1-control-plane-7llgd
1/1     Running    0             20h
kube-system              etcd-d2iq-e2e-cluster-1-control-plane-vncbl
1/1     Running    0             19h
kube-system              etcd-d2iq-e2e-cluster-1-control-plane-wbgrm
1/1     Running    0             19h
kube-system              kube-apiserver-d2iq-e2e-cluster-1-control-plane-7llgd
1/1     Running    0             20h
kube-system              kube-apiserver-d2iq-e2e-cluster-1-control-plane-vncbl
1/1     Running    0             19h
kube-system              kube-apiserver-d2iq-e2e-cluster-1-control-plane-wbgrm
1/1     Running    0             19h
kube-system              kube-controller-manager-d2iq-e2e-cluster-1-control-
plane-7llgd   1/1     Running    1 (19h ago)   20h
kube-system              kube-controller-manager-d2iq-e2e-cluster-1-control-
plane-vncbl   1/1     Running    0             19h
kube-system              kube-controller-manager-d2iq-e2e-cluster-1-control-
plane-wbgrm   1/1     Running    0             19h
kube-system              kube-proxy-cpscs
1/1     Running    0             19h
kube-system              kube-proxy-hhmxq
1/1     Running    0             19h
kube-system              kube-proxy-hxhnk
1/1     Running    0             19h
kube-system              kube-proxy-nsrbp
1/1     Running    0             19h
kube-system              kube-proxy-scxfg
1/1     Running    0             20h
kube-system              kube-proxy-tth4k
1/1     Running    0             19h
kube-system              kube-proxy-x2xfx
1/1     Running    0             19h
```

```
kube-system              kube-scheduler-d2iq-e2e-cluster-1-control-plane-7llgd
1/1     Running    1 (19h ago)   20h
kube-system              kube-scheduler-d2iq-e2e-cluster-1-control-plane-vncbl
1/1     Running    0             19h
kube-system              kube-scheduler-d2iq-e2e-cluster-1-control-plane-wbgrm
1/1     Running    0             19h
kube-system              kube-vip-d2iq-e2e-cluster-1-control-plane-7llgd
1/1     Running    1 (19h ago)   20h
kube-system              kube-vip-d2iq-e2e-cluster-1-control-plane-vncbl
1/1     Running    0             19h
kube-system              kube-vip-d2iq-e2e-cluster-1-control-plane-wbgrm
1/1     Running    0             19h
kube-system              vsphere-cloud-controller-manager-4zj7q
1/1     Running    0             19h
kube-system              vsphere-cloud-controller-manager-87tgm
1/1     Running    0             19h
kube-system              vsphere-cloud-controller-manager-xqmn4
1/1     Running    1 (19h ago)   20h
node-feature-discovery   node-feature-discovery-master-84c67dcbb6-txfw9
1/1     Running    0             20h
node-feature-discovery   node-feature-discovery-worker-8tg2l
1/1     Running    3 (19h ago)   19h
node-feature-discovery   node-feature-discovery-worker-c5f6q
1/1     Running    0             19h
node-feature-discovery   node-feature-discovery-worker-fjfkm
1/1     Running    0             19h
node-feature-discovery   node-feature-discovery-worker-x6tz8
1/1     Running    0             19h
tigera-operator          tigera-operator-d499f5c8f-r2srj
1/1     Running    1 (19h ago)   20h
vmware-system-csi        vsphere-csi-controller-7ffd6884cc-d7rql
7/7     Running    5 (19h ago)   20h
vmware-system-csi        vsphere-csi-controller-7ffd6884cc-k82cm
7/7     Running    2 (19h ago)   20h
vmware-system-csi        vsphere-csi-controller-7ffd6884cc-qttkp
7/7     Running    1 (19h ago)   20h
vmware-system-csi        vsphere-csi-node-678hw
3/3     Running    0             19h
vmware-system-csi        vsphere-csi-node-6tbsh
3/3     Running    0             19h
vmware-system-csi        vsphere-csi-node-9htwr
3/3     Running    5 (20h ago)   20h
vmware-system-csi        vsphere-csi-node-g8r6l
3/3     Running    0             19h
vmware-system-csi        vsphere-csi-node-ghmr6
3/3     Running    0             19h
vmware-system-csi        vsphere-csi-node-jhvgm
3/3     Running    0             19h
vmware-system-csi        vsphere-csi-node-rp77r
3/3     Running    0             19h
```

6.6.9 Configure MetalLB for a vSphere infrastructure

6.6.9.1 Create a MetalLB configmap for your vSphere infrastructure.

Choose one of the following two protocols you want to use to announce service IPs, either Layer 2 or BGP configurations.

6.6.9.2 Layer 2 configuration

Layer 2 mode is the simplest to configure: in many cases, you don't need any protocol-specific configuration, only IP addresses.

Layer 2 mode does not require the IPs to be bound to the network interfaces of your worker nodes. It works by responding to ARP requests on your local network directly, to give the machine's MAC address to clients.

For example, the following configuration gives MetalLB control over IPs from 192.168.1.240 to 192.168.1.250, and configures Layer 2 mode:

The following values are generic, enter your specific values into the fields where applicable.

```
cat << EOF > metallb-conf.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: metallb-system
  name: config
data:
  config: |
    address-pools:
    - name: default
      protocol: layer2
      addresses:
      - 192.168.1.240-192.168.1.250
EOF
```

When complete, run the following `kubectl` command.

```
kubectl apply -f metallb-conf.yaml
```

6.6.9.3 BGP configuration

For a basic configuration featuring one BGP router and one IP address range, you need four pieces of information:

- The router IP address that MetalLB should connect to,
- The router's AS number,
- The AS number MetalLB should use,
- An IP address range expressed as a CIDR prefix.

As an example, if you want to give MetalLB the range 192.168.10.0/24 and AS number 64500, and connect it to a router at 10.0.0.1 with AS number 64501, your configuration will look like:

The following values are generic, enter your specific values into the fields where applicable.

Extract the kubeconfig and deploy a config map for MetalLB using the following command:

```
dkp get kubeconfig -c ${DKP_CLUSTER_NAME} > ${DKP_CLUSTER_NAME}.conf
```

Deploy MetalLB Configuration with the command below:

```
cat << EOF > metallb-conf.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: metallb-system
  name: config
data:
  config: |
    peers:
    - peer-address: 10.0.0.1
      peer-asn: 64501
      my-asn: 64500
    address-pools:
    - name: default
      protocol: bgp
      addresses:
      - 192.168.10.0/24
EOF
```

When complete, run the following `kubectl` command:

```
kubectl apply -f metallb-conf.yaml
```

6.6.10 Install vSphere Air-Gapped

6.6.10.1 Create a Kubernetes vSphere cluster in a private network with no access to the Internet (air-gapped)

This section provides the instructions to create a Kubernetes cluster in a private, DHCP-enabled network with no access to the Internet. Complete the list of Prerequisites(see page 253) before you begin the procedures in this section.

6.6.10.2 Create and Prepare a Bastion VM Host

Create and Prepare a Bastion VM host for use with a vSphere air-gapped cluster

Prerequisites

Before you start this procedure, ensure that you complete the Prerequisites(see page 253).

Create a vSphere bastion VM host template

When creating an air-gapped vSphere cluster, the bastion VM hosts the installation DKP Konvoy bundles and images, and the Docker registry, needed to create and operate your vSphere cluster. The bastion VM must have access to the vSphere API Server (vCenter Server).

1. Create a bastion VM host template for the cluster nodes to use within the air-gapped network. This bastion VM host also needs access to a Docker registry in lieu of an Internet connection for pulling Docker images. The recommended template naming pattern is `../folder-name/dkp-e2e-bastion-template` or similar.
2. Find and record the bastion VM's IP or host name.
3. Download(see page 83) the following required DKP Konvoy binaries and installation bundles appropriate for your environment directly to the bastion host, or transfer them using your environment's approved methods:
    - dkp_v2.3.2_darwin_amd64.tar.gz
    - dkp_v2.3.2_linux_amd64.tar.gz
    - konvoy_image_bundle_v2.3.2_linux_amd64.tar.gz (air-gapped) - This bundle contains air-gapped images that you must push to a registry.)
    - dkp-bootstrap_v2.3.2.tar (air-gapped) - This bundle contains the KIND bootstrap image to load with the `docker load` command when you create the bootstrap cluster in a later step.
4. Use your credentials to SSH into the bastion VM host with the command:

    ```
    ssh -i </path/to/private_key> <USER>@<BASTION_IP>
    ```

5. Set the following vSphere environment variables on the bastion VM host:

    ```
    export VSPHERE_SERVER=your_vCenter_APIserver_URL
    export VSPHERE_USERNAME=your_vCenter_user_name
    export VSPHERE_PASSWORD=your_vCenter_password
    ```

6. Set the following environment variables to enable connection to an existing Docker registry:

⚠ **IMPORTANT:** You must create the VM template with the konvoy-image-builder[221] project to be able to use the registry mirror feature.

```
export DOCKER_REGISTRY_ADDRESS=<https/http>://<registry-address>:<registry-port>
export DOCKER_REGISTRY_CA=<path to the CA on the bastion host>
```

- `DOCKER_REGISTRY_ADDRESS` : the address of an existing Docker registry accessible in the vSphere Zone where the new cluster nodes will be configured, to use a mirror registry when pulling images.
- `DOCKER_REGISTRY_CA` : (optional) the path on the bastion host to the Docker registry CA. Konvoy configures the cluster nodes to trust this CA. This value is only needed if the registry is using a self-signed certificate and the VMs are not already configured to trust this CA.

When you complete this procedure, the next step is to create a bootstrap cluster(see page 277).

6.6.10.3 Create a Base OS VM Image

Create a Base OS VM Image in the VMware vSphere Client (air-gapped)

Create a base OS image in the vSphere client

Creating a base OS image from DVD ISO files is a one-time process. Building a base OS image creates a base vSphere template in your vSphere environment. You can use the base OS image template to create a Kubernetes node vSphere template for your cluster.

Depending on the cluster-api provider, a packer user configuration will vary. For vSphere, a user and password is required by default by packer[222]. This user should have administrator privileges. It is possible to configure a custom user and password when building the OS image, however, that requires the Konvoy Image Builder (KIB) configuration to be overridden(see page 330). DKP advises not to use static usernames and passwords for security reasons, but instead passwords should be generated and a minimum of 20 characters long.

While creating the base OS image, it is important to take into consideration the following elements:

1. Storage configuration: D2iQ recommends customizing disk partitions and not configuring a SWAP partition.
2. Network configuration: as KIB must download and install packages, activating the network is required.
3. Connect to Red Hat: if using RHEL, registering with Red Hat is required to configure software repositories and install software packages.
4. Software selection: D2iQ recommends choosing **Minimal Install**.
5. DKP recommends to install with the packages provided by the operating system package managers. Use the version that corresponds to the major version of your operating system.
6. DKP advises not to use static usernames and passwords for security reasons, but instead passwords should be generated and a minimum of 20 characters long.

221 https://github.com/mesosphere/konvoy-image-builder
222 https://github.com/mesosphere/konvoy-image-builder/blob/eecdd73ec9a1ca07c24962e2637bbab01a44d347/pkg/packer/manifests/vsphere/packer.json.tmpl#L15-L17

Refer to the vCenter and vSphere Client documentation for details.

The next step is to create a VM template(see page 276) that contains all of the configuration and artifacts needed to create a Kubernetes cluster.

6.6.10.4 Create a CAPI VM Image

Create a CAPI VM image using the DKP image builder

Prerequisites

Users will need to create a base OS image[223] in vSphere before starting this procedure.

Create a vSphere template for your cluster from a base OS image

Using the base OS image created in a previous procedure, DKP creates the new vSphere template directly on the vCenter server.

1. Set the following vSphere environment variables on the bastion VM host:

```
export VSPHERE_SERVER=your_vCenter_APIserver_URL
export VSPHERE_USERNAME=your_vCenter_user_name
export VSPHERE_PASSWORD=your_vCenter_password
```

2. Copy the base OS image file created in the vSphere Client to your desired location on the bastion VM host, and make a note of the path and file name.
3. Create an `image.yaml` file and add the following variables for vSphere. DKP uses this file and these variables as inputs in the next step.

```
build_name: "vsphere-rhel-79"
packer_builder_type: "vsphere"
packer:
  cluster: "example_zone"
  datacenter: "example_datacenter"
  datastore: "example_datastore"
  folder: "example_folder"
  insecure_connection: "false"
  network: "example_network"
  resource_pool: "example_resource_pool"
  template: "example_base_OS_template_name"
  vsphere_guest_os_type: "example_rhel8_64Guest"
  guest_os_type: "example_rhel7-64"
  #goss params
  distribution: "example_RHEL"
```

223 https://d2iq.atlassian.net/wiki/spaces/DKP/pages/10290580/Create+a+Base+OS+VM+Image#Create-a-Base-OS-VM-Image-in-the-VMware-vSphere-Client-(air-gapped)

```
  distribution_version: "example_7.9"
```

4. Create a vSphere VM template with the following command:

   ```
   konvoy-image build path/to/image.yaml \
     --overrides /path/to/overrides/offline.yaml
   ```

   The DKP image builder uses the values in `image.yaml` and the input base OS image to create a vSphere template that contains the required artifacts needed to create a Kubernetes cluster. Give the file a suitable name using this suggested naming convention: `creator-ova-vsphere-OS-ver-k8sver-unique_identifier`. As an example, the filename you create might resemble `dkp-ova-vsphere-rhel-84-1.21.6-1646938922`.
   DKP creates the new vSphere template directly on the vCenter server.

Next, create a Kubernetes bootstrap cluster[224] to enable creating your vSphere cluster and moving CAPI objects to it.

6.6.10.5 vSphere Air-gapped Bootstrap

Prerequisites

Before you perform this procedure, ensure that you have created a CAPI VM template(see page 256).

Bootstrap a kind cluster and CAPI controllers

DKP Konvoy deploys all cluster lifecycle services to a bootstrap cluster, which deploys a workload cluster. When the workload cluster is ready, move the cluster lifecycle services to the workload cluster, after which the workload cluster manages its own lifecycle.

1. Copy the image tar file to the machine where you want to run the bootstrap cluster.
2. Load the bootstrap Docker image. The image version should correspond to the version of Konvoy as returned by `dkp version`:

   ```
   docker load -i <path to mesosphere/konvoy-bootstrap image>
   ```

3. Create a bootstrap cluster with the command:

   ```
   dkp create bootstrap --kubeconfig $HOME/.kube/config
   ```

   The output resembles this example:

224 https://d2iq.atlassian.net/wiki/spaces/DKP/pages/10356022/vSphere+Air-gapped+Bootstrap#Bootstrap-a-kind-cluster-and-CAPI-controllers

```
✓ Creating a bootstrap cluster
✓ Initializing new CAPI components
```

4. Ensure that the CAPV controllers are present with the command:

```
kubectl get pods -n capv-system
```

The output resembles the following:

```
NAME                                       READY   STATUS    RESTARTS   AGE
capv-controller-manager-785c5978f-nnfns    1/1     Running   0          13h
```

5. Refresh the credentials used by the vSphere provider at any time, using the command:

```
dkp update bootstrap credentials vsphere
```

Next, you can create a new vSphere Kubernetes cluster(see page 278).

6.6.10.6 Create a new Air-gapped vSphere Cluster

Prerequisites

Before you begin, be sure that you have created a bootstrap cluster[225]

Create a new vSphere Kubernetes cluster

Use the following steps to create a new, air-gapped vSphere cluster.

1. Create a Kubernetes cluster by copying the following command and substituting the valid values for your environment:

```
./konvoy create cluster vsphere
  --cluster-name ${CLUSTER_NAME} \
  --network <NETWORK_NAME> \
  --control-plane-endpoint-host <CONTROL_PLANE_IP> \
  --data-center <DATACENTER_NAME> \
  --data-store <DATASTORE_NAME> \
  --folder <FOLDER_NAME> \
  --server <VCENTER_API_SERVER_URL> \
  --ssh-public-key-file </path/to/key.pub> \
```

225 https://d2iq.atlassian.net/wiki/spaces/DKP/pages/10356022/vSphere+Air-gapped+Bootstrap#Bootstrap-a-kind-cluster-and-CAPI-controllers

```
  --resource-pool <RESOURCE_POOL_NAME> \
  --vm-template konvoy-ova-vsphere-os-release-k8s_release-vsphere-timestamp \
  --virtual-ip-interface <ip_interface_name> \
  --extra-sans "127.0.0.1" \
  --registry-mirror-url ${DOCKER_REGISTRY_ADDRESS} \
  --registry-mirror-cacert /tmp/registry.pem
```

Review Create and Prepare a Bastion VM Host[226] for more about the ${DOCKER_REGISTRY_ADDRESS}.

2. Inspect the created cluster resources with the command:

```
kubectl get clusters,kubeadmcontrolplanes,machinedeployments
```

3. Use the `wait` command to monitor the cluster control-plane readiness:

```
kubectl wait --for=condition=ControlPlaneReady "clusters/${CLUSTER_NAME}" --
timeout=20m
```

```
cluster.cluster.x-k8s.io/${CLUSTER_NAME} condition met
```

The `READY` status becomes `True` after the cluster control-plane becomes Ready in one of the following steps.
After DKP creates the objects on the API server, the Cluster API controllers reconcile them, creating infrastructure and machines. As the controllers progress, they update the Status of each object.

4. Run the DKP `describe` command to monitor the current status of the cluster:

```
dkp describe cluster -c ${CLUSTER_NAME}
```

```
NAME                                                          READY
SEVERITY  REASON  SINCE  MESSAGE
Cluster/e2e-airgapped-1                                       True
13h
├──ClusterInfrastructure - VSphereCluster/e2e-airgapped-1      True
13h
├──ControlPlane - KubeadmControlPlane/e2e-airgapped-1-control-plane  True
13h
│  ├──Machine/e2e-airgapped-1-control-plane-7llgd              True
13h
```

226 https://d2iq.atlassian.net/wiki/spaces/DKP/pages/10552171/Create+and+Prepare+a+Bastion+VM+Host#Create-a-vSphere-Bastion-VM-Host-Template

```
│ ├──Machine/e2e-airgapped-1-control-plane-vncbl                    True
13h
│ └──Machine/e2e-airgapped-1-control-plane-wbgrm                    True
13h
└──Workers
    └──MachineDeployment/e2e-airgapped-1-md-0                       True
13h
    ├──Machine/e2e-airgapped-1-md-0-74c849dc8c-67rv4                True
13h
    ├──Machine/e2e-airgapped-1-md-0-74c849dc8c-n2skc                True
13h
    ├──Machine/e2e-airgapped-1-md-0-74c849dc8c-nkftv                True
13h
    └──Machine/e2e-airgapped-1-md-0-74c849dc8c-sqklv                True
13h
```

5. As they progress, the controllers also create Events, which you can list using the command:

   ```
   kubectl get events | grep ${CLUSTER_NAME}
   ```

   For brevity, this example uses `grep`. You can also use separate commands to get Events for specific objects, such as `kubectl get events --field-selector involvedObject.kind="VSphereCluster"` and `kubectl get events --field-selector involvedObject.kind="VSphereMachine"`.

Next, you can make the cluster self-managed[227], and then explore your new cluster[228].

Known limitations

NOTE: Be aware of these limitations in the current release of DKP Konvoy.

- The DKP Konvoy version used to create a bootstrap cluster must match the DKP Konvoy version used to create a workload cluster.
- DKP Konvoy supports deploying one workload cluster.
- DKP Konvoy generates a set of objects for one Node Pool.
- DKP Konvoy does not validate edits to cluster objects.

227 https://d2iq.atlassian.net/wiki/spaces/DKP/pages/10944515/Make+vSphere+Cluster+Self-Managed#Make-the-new-Kubernetes-cluster-manage-itself

228 https://d2iq.atlassian.net/wiki/spaces/DKP/pages/10552155/Explore+vSphere+Air-gapped+Cluster#Create-a-new-vSphere-Kubernetes-cluster

6.6.10.7 Explore vSphere Air-gapped Cluster

Get the kubeconfig file for the new Kubernetes cluster

1. Fetch the kubeconfig file with the command:

```
dkp get kubeconfig -c ${air-gapped_NAME} > ${air-gapped_NAME}.conf
```

Create a StorageClass with a vSphere datastore

Follow these steps:

1. Access the Datastore tab in the vSphere client and select a datastore by name.
2. Copy the URL of that datastore from the information dialog that displays.
3. Return to the DKP CLI, and delete the existing `StorageClass` with the command:

```
kubectl delete storageclass vsphere-raw-block-sc
```

4. Run the following command to create a new StorageClass, supplying the correct values for your environment:

```
cat <<EOF > vsphere-raw-block-sc.yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
  name: vsphere-raw-block-sc
provisioner: csi.vsphere.vmware.com
parameters:
  datastoreurl: "<url>"
volumeBindingMode: WaitForFirstConsumer
EOF
```

Explore nodes and pods in the new cluster

1. List the Nodes with this command:

```
kubectl --kubeconfig=${air-gapped_NAME}.conf get nodes
```

NOTE: It may take a few minutes for the Status to move to Ready while the Pod network is deployed. The Node's Status should change to Ready soon after the calico-node DaemonSet Pods are Ready.

The output resembles this example:

```
NAME                                          STATUS   ROLES                  AGE
VERSION
d2iq-e2e-air-gapped-1-control-plane-7llgd     Ready    control-plane,master   20h
v1.22.8
d2iq-e2e-air-gapped-1-control-plane-vncbl     Ready    control-plane,master   19h
v1.22.8
d2iq-e2e-air-gapped-1-control-plane-wbgrm     Ready    control-plane,master   19h
v1.22.8
d2iq-e2e-air-gapped-1-md-0-74c849dc8c-67rv4   Ready    <none>                 19h
v1.22.8
d2iq-e2e-air-gapped-1-md-0-74c849dc8c-n2skc   Ready    <none>                 19h
v1.22.8
d2iq-e2e-air-gapped-1-md-0-74c849dc8c-nkftv   Ready    <none>                 19h
v1.22.8
d2iq-e2e-air-gapped-1-md-0-74c849dc8c-sqklv   Ready    <none>                 19h
v1.22.8
```

2. List the pods with the command:

```
kubectl --kubeconfig=${air-gapped_NAME}.conf get pods -A
```

The output resembles the following example:

```
NAMESPACE                NAME
READY   STATUS     RESTARTS      AGE
calico-system            calico-kube-controllers-57fbd7bd59-qqd96
1/1     Running    0             20h
calico-system            calico-node-2m524
1/1     Running    3 (19h ago)   19h
calico-system            calico-node-bbhg5
1/1     Running    0             20h
calico-system            calico-node-cc5lf
1/1     Running    2 (19h ago)   19h
calico-system            calico-node-cwg7x
1/1     Running    1 (19h ago)   19h
calico-system            calico-node-d59hn
1/1     Running    1 (19h ago)   19h
calico-system            calico-node-qmmcz
1/1     Running    0             19h
```

```
calico-system            calico-node-wdqhx
1/1     Running    0             19h
calico-system            calico-typha-655489d8cc-b5jnt
1/1     Running    0             20h
calico-system            calico-typha-655489d8cc-q92x9
1/1     Running    0             19h
calico-system            calico-typha-655489d8cc-vjlkx
1/1     Running    0             19h
kube-system              cluster-autoscaler-68c759fbf6-7d2ck
0/1     Init:0/1   0             20h
kube-system              coredns-78fcd69978-qn4qt
1/1     Running    0             20h
kube-system              coredns-78fcd69978-wqpmg
1/1     Running    0             20h
kube-system              etcd-d2iq-e2e-air-gapped-1-control-plane-7llgd
1/1     Running    0             20h
kube-system              etcd-d2iq-e2e-air-gapped-1-control-plane-vncbl
1/1     Running    0             19h
kube-system              etcd-d2iq-e2e-air-gapped-1-control-plane-wbgrm
1/1     Running    0             19h
kube-system              kube-apiserver-d2iq-e2e-air-gapped-1-control-plane-7llgd
1/1     Running    0             20h
kube-system              kube-apiserver-d2iq-e2e-air-gapped-1-control-plane-vncbl
1/1     Running    0             19h
kube-system              kube-apiserver-d2iq-e2e-air-gapped-1-control-plane-wbgrm
1/1     Running    0             19h
kube-system              kube-controller-manager-d2iq-e2e-air-gapped-1-control-
plane-7llgd   1/1     Running    1 (19h ago)   20h
kube-system              kube-controller-manager-d2iq-e2e-air-gapped-1-control-plane-
vncbl   1/1     Running    0             19h
kube-system              kube-controller-manager-d2iq-e2e-air-gapped-1-control-plane-
wbgrm   1/1     Running    0             19h
kube-system              kube-proxy-cpscs
1/1     Running    0             19h
kube-system              kube-proxy-hhmxq
1/1     Running    0             19h
kube-system              kube-proxy-hxhnk
1/1     Running    0             19h
kube-system              kube-proxy-nsrbp
1/1     Running    0             19h
kube-system              kube-proxy-scxfg
1/1     Running    0             20h
kube-system              kube-proxy-tth4k
1/1     Running    0             19h
kube-system              kube-proxy-x2xfx
1/1     Running    0             19h
kube-system              kube-scheduler-d2iq-e2e-air-gapped-1-control-plane-7llgd
1/1     Running    1 (19h ago)   20h
kube-system              kube-scheduler-d2iq-e2e-air-gapped-1-control-plane-vncbl
1/1     Running    0             19h
kube-system              kube-scheduler-d2iq-e2e-air-gapped-1-control-plane-wbgrm
1/1     Running    0             19h
```

```
kube-system              kube-vip-d2iq-e2e-air-gapped-1-control-plane-7llgd
1/1     Running    1 (19h ago)   20h
kube-system              kube-vip-d2iq-e2e-air-gapped-1-control-plane-vncbl
1/1     Running    0             19h
kube-system              kube-vip-d2iq-e2e-air-gapped-1-control-plane-wbgrm
1/1     Running    0             19h
kube-system              vsphere-cloud-controller-manager-4zj7q
1/1     Running    0             19h
kube-system              vsphere-cloud-controller-manager-87tgm
1/1     Running    0             19h
kube-system              vsphere-cloud-controller-manager-xqmn4
1/1     Running    1 (19h ago)   20h
node-feature-discovery   node-feature-discovery-master-84c67dcbb6-txfw9
1/1     Running    0             20h
node-feature-discovery   node-feature-discovery-worker-8tg2l
1/1     Running    3 (19h ago)   19h
node-feature-discovery   node-feature-discovery-worker-c5f6q
1/1     Running    0             19h
node-feature-discovery   node-feature-discovery-worker-fjfkm
1/1     Running    0             19h
node-feature-discovery   node-feature-discovery-worker-x6tz8
1/1     Running    0             19h
tigera-operator          tigera-operator-d499f5c8f-r2srj
1/1     Running    1 (19h ago)   20h
vmware-system-csi        vsphere-csi-controller-7ffd6884cc-d7rql
7/7     Running    5 (19h ago)   20h
vmware-system-csi        vsphere-csi-controller-7ffd6884cc-k82cm
7/7     Running    2 (19h ago)   20h
vmware-system-csi        vsphere-csi-controller-7ffd6884cc-qttkp
7/7     Running    1 (19h ago)   20h
vmware-system-csi        vsphere-csi-node-678hw
3/3     Running    0             19h
vmware-system-csi        vsphere-csi-node-6tbsh
3/3     Running    0             19h
vmware-system-csi        vsphere-csi-node-9htwr
3/3     Running    5 (20h ago)   20h
vmware-system-csi        vsphere-csi-node-g8r6l
3/3     Running    0             19h
vmware-system-csi        vsphere-csi-node-ghmr6
3/3     Running    0             19h
vmware-system-csi        vsphere-csi-node-jhvgm
3/3     Running    0             19h
vmware-system-csi        vsphere-csi-node-rp77r
3/3     Running    0             19h
```

When you are ready, delete your cluster and clean up your environment[229].

229 https://d2iq.atlassian.net/wiki/spaces/DKP/pages/10552302/Delete+vSphere+Cluster#Prepare-to-delete-a-self-managed-workload-cluster

6.6.11 vSphere Certificate Renewal

6.6.11.1 Configure Automated Renewal for Managed Kubernetes PKI Certficates

6.6.11.2 Certificate Renewal

During cluster creation, Kubernetes establishes a Public Key Infrastructure (PKI) for generating the TLS certificates needed for securing cluster communication for various components such as `etcd`, `kubernetes-apiserver` and `kube-proxy`. The certificates created by these components have a default expiration of one year and are renewed when an administrator updates the cluster.

Kubernetes provides a facility to renew all certificates automatically during control plane updates. For administrators who need long-running clusters or clusters that are not upgraded often, `dkp` provides automated certificate renewal, without a cluster upgrade.

Requirements

This feature requires that you install Python 3.5 or higher version on all control plane hosts.

6.6.11.3 Prerequisites

- Complete the bootstrap Cluster Lifecycle[230] topic.

Create a cluster with automated certificate renewal

To enable the automated certificate renewal, create a Konvoy cluster using the `certificate-renew-interval` flag:

```
dkp create cluster vsphere --certificate-renew-interval=60 --cluster-name=long-
running
```

The `certificate-renew-interval` is the number of days after which Kubernetes-managed PKI certificates will be renewed. For example, an `certificate-renew-interval` value of 30 means the certificates are renewed every 30 days.

Technical details

The following manifests are modified on the control plane hosts, and are located at `/etc/kubernetes/manifests`. Modifications to these files requires SUDO access.

230 https://d2iq.atlassian.net/wiki/spaces/DKP/pages/10878997/vSphere+Bootstrap#Prepare-to-deploy-Kubernetes-clusters

```
kube-controller-manager.yaml
kube-apiserver.yaml
kube-scheduler.yaml
kube-proxy.yaml
```

The following annotation indicates the time each component was reset:

```
metadata:
  annotations:
    konvoy.d2iq.io/restartedAt: $(date +%s)
```

This only occurs when the PKI certificates are older than the interval given at cluster creation time. This is activated by a `systemd` `timer` called `renew-certs.timer` that triggers an associated `systemd` service called `renew-certs.service` that runs on all of the control plane hosts.

Debugging

To debug the automatic certificate renewal feature, a cluster administrator can look at several different components to see if the certificates were renewed. For example, an administrator might start with a look at the control plane pod definition to check the last reset time. To determine if a scheduler pod was properly reset, run the command:

```
kubectl get pod -n kube-system kube-scheduler-nodename -o yaml
```

The output of the command is similar to the following:

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    konvoy.d2iq.io/restartedAt: "1626124940.735733"
```

Administrators who want more details on the execution of the `systemd` service can use `ssh` to connect to the control plane hosts, and then use the `systemctl` and `journalctl` commands that follow to help diagnose potential issues.

Check timer status

To check the status of the timers, when they last ran, and when they are scheduled to run next, use the command:

```
systemctl list-timers
```

Check renew certs status

To check the status of the `renew-certs` service, use the command:

```
systemctl status renew-certs
```

Review logs

To get the logs of the last run of the service, use the command:

```
journalctl logs -u renew-certs
```

6.6.12 Delete vSphere Cluster

6.6.12.1 Prepare to delete a self-managed workload cluster

> A self-managed workload cluster cannot delete itself. If your workload cluster is self-managed, you must create a bootstrap cluster and move the cluster lifecycle services to the bootstrap cluster before deleting the workload cluster.

If you did not make your workload cluster self-managed, as described in Make New Cluster Self-Managed(see page 265), see Delete the workload cluster(see page 287).

1. Create a bootstrap cluster:
   The bootstrap cluster will host the Cluster API controllers that reconcile the cluster objects marked for deletion. To avoid using the wrong kubeconfig, the following steps use **explicit** kubeconfig paths and contexts.

   ```
   dkp create bootstrap --kubeconfig $HOME/.kube/config
   ```

   The output resembles this example:

   ```
   ✓ Creating a bootstrap cluster
   ✓ Initializing new CAPI components
   ```

2. Move the Cluster API objects from the workload to the bootstrap cluster: The cluster lifecycle services on the bootstrap cluster are ready, but the workload cluster configuration is on the workload cluster. The `move` command moves the configuration, which takes the form of Cluster API Custom Resource objects, from the workload to the bootstrap cluster. This process is also called a Pivot[231].

```
dkp move \
    --from-kubeconfig ${CLUSTER_NAME}.conf \
    --from-context konvoy-${CLUSTER_NAME}-admin@konvoy-${CLUSTER_NAME} \
    --to-kubeconfig $HOME/.kube/config \
    --to-context kind-konvoy-capi-bootstrapper
```

```
INFO[2021-06-09T11:47:11-07:00] Running pivot command
fromClusterKubeconfig=aws-example.conf fromClusterContext= src="move/
move.go:83" toClusterKubeconfig=/home/clusteradmin/.kube/config
toClusterContext=
INFO[2021-06-09T11:47:36-07:00] Pivot operation complete.
src="move/move.go:108"
INFO[2021-06-09T11:47:36-07:00] You can now view resources in the moved cluster
by using the --kubeconfig flag with kubectl. For example: kubectl --
kubeconfig=/home/clusteradmin/.kube/config get nodes  src="move/move.go:155"
```

3. Use the cluster lifecycle services on the workload cluster to check the workload cluster status:

```
dkp describe cluster --kubeconfig $HOME/.kube/config -c ${CLUSTER_NAME}
```

```
NAME                                                                READY
SEVERITY  REASON  SINCE  MESSAGE
Cluster/d2iq-e2e-cluster_name-1                                     True
13h
├──ClusterInfrastructure - VSphereCluster/d2iq-e2e-cluster_name-1    True
13h
├──ControlPlane - KubeadmControlPlane/d2iq-control-plane            True
13h
│  ├──Machine/d2iq--control-plane-7llgd                             True
13h
│  ├──Machine/d2iq--control-plane-vncbl                             True
13h
│  └──Machine/d2iq--control-plane-wbgrm                             True
13h
└──Workers
    └──MachineDeployment/d2iq--md-0                                 True
13h
    ├──Machine/d2iq--md-0-74c849dc8c-67rv4                          True
13h
```

231 https://cluster-api.sigs.k8s.io/reference/glossary.html?highlight=pivot#pivot

```
     ├──Machine/d2iq--md-0-74c849dc8c-n2skc                          True
13h
     ├──Machine/d2iq--md-0-74c849dc8c-nkftv                          True
13h
     └──Machine/d2iq--md-0-74c849dc8c-sqklv                          True
13h
```

After moving the cluster lifecycle services to the workload cluster, remember to use dkp with the workload cluster kubeconfig. Use DKP with the bootstrap cluster to delete the workload cluster.

4. Wait for the cluster control-plane to be ready:

```
kubectl --kubeconfig $HOME/.kube/config wait --for=condition=controlplaneready
"clusters/${CLUSTER_NAME}" --timeout=60m
```

The output should be similar to this example:

```
d2iq-e2e-cluster-1-control-plane/vsphere-example condition met
```

Persistent Volumes (PVs) are not deleted automatically by design in order to preserve your data. However, they take up storage space if not deleted. You must delete PVs manually. With Vsphere clusters, `dkp delete` doesn't delete the virtual disks backing the PVs for DKP add ons. Therefore internal VMware cluster runs out of storage eventually. Theses PVs are only visible if VSAN is installed which gives users a Container Native Storage tab.

6.6.12.2 Delete the workload cluster

1. Make sure your vSphere credentials are up-to-date. Refresh the credentials using this command:

```
dkp update bootstrap credentials vsphere --kubeconfig $HOME/.kube/config
```

2. Delete the Kubernetes cluster and wait a few minutes:
Before deleting the cluster, DKP deletes all Services of type LoadBalancer on the cluster.
To skip this step, use the flag `--delete-kubernetes-resources=false`.

```
dkp delete cluster --cluster-name=${CLUSTER_NAME} --kubeconfig $HOME/.kube/
config
```

```
INFO[2022-03-30T11:53:42-07:00] Running cluster delete command
clusterName=d2iq-e2e-cluster-1 managementClusterKubeconfig= namespace=default
 src="cluster/delete.go:95"
INFO[2022-03-30T11:53:42-07:00] Waiting for cluster to be fully deleted
src="cluster/delete.go:123"
INFO[2022-03-30T12:14:03-07:00] Deleted default/d2iq-e2e-cluster-1 cluster
src="cluster/delete.go:129"
```

After the workload cluster is deleted, delete the bootstrap cluster with the following command.

6.6.12.3 Delete the Bootstrap Cluster

```
dkp delete bootstrap --kubeconfig $HOME/.kube/config
```

```
INFO[2021-06-09T12:15:20-07:00] Deleting bootstrap cluster
src="bootstrap/bootstrap.go:182"
```

6.6.12.4 Known Limitations

Be aware of these limitations in the current release of DKP Konvoy.

- The DKP Konvoy version used to create the workload cluster must match the DKP Konvoy version used to delete the workload cluster.

6.6.13 Manage vSphere Node Pools

Node pools are part of a cluster and managed as a group, and you can use a node pool to manage a group of machines using the same common properties. When Konvoy creates a new default cluster, there is one node pool for the worker nodes, and all nodes in that new node pool have the same configuration.

You can create additional node pools for more specialized hardware or configuration. For example, if you want to tune your memory usage on a cluster where you need maximum memory for some machines and minimal memory on other machines, you would create a new node pool with those specific resource needs.

Konvoy implements node pools using Cluster API MachineDeployments[232].

6.6.13.1 Create vSphere Node Pools

Creating a node pool is useful when you need to run workloads that require machines with specific resources, such as a GPU, additional memory, or specialized network or storage hardware.

Prepare the environment

Follow these steps:

1. Set the environment variable to the name you assigned this cluster with the command:

   ```
   export CLUSTER_NAME=my-vsphere-cluster
   ```

232 https://cluster-api.sigs.k8s.io/developer/architecture/controllers/machine-deployment.html

2. If your workload cluster is self-managed, as described in Make the New Cluster Self-Managed(see page 265), configure `kubectl` to use the kubeconfig for the cluster:

   ```
   export KUBECONFIG=${CLUSTER_NAME}.conf
   ```

3. Define your node pool name:

   ```
   export NODEPOOL_NAME=example
   ```

Create a vSphere node pool

Create a new vSphere node pool with 3 replicas using this command:

```
dkp create nodepool vsphere ${NODEPOOL_NAME} \
  --cluster-name=${CLUSTER_NAME} \
  --network=example_network \
  --data-center=example_datacenter \
  --data-store=example_datastore \
  --folder=example_folder \
  --server=example_vsphere_api_server_url\
  --resource-pool=example_resource_pool \
  --vm-template=example_vm_template \
  --replicas=3
```

The output resembles this example:

```
machinedeployment.cluster.x-k8s.io/example created
vspheremachinetemplate.infrastructure.cluster.x-k8s.io/example created
kubeadmconfigtemplate.bootstrap.cluster.x-k8s.io/example created
 ✓ Creating default/example nodepool resources
```

This example uses default values for brevity. Advanced users can use a combination of the `--dry-run` and `--output=yaml` flags to get a complete set of node pool objects to modify locally or store in version control.

6.6.13.2 List vSphere Node Pools

Listing node pools

Use this command to list the node pools of a given cluster. This returns specific properties of each node pool so that you can see the name of the MachineDeployments.

To list all node pools for a managed cluster, run the command:

```
dkp get nodepools --cluster-name=${CLUSTER_NAME} --kubeconfig=${CLUSTER_NAME}.conf
```

The expected output is similar to the following example, indicating the desired size of the node pool, the number of replicas ready in the node pool, and the Kubernetes version those nodes are running:

```
NODEPOOL                    DESIRED             READY               KUBERNETES
VERSION
demo-cluster-md-0           4                   4                   v1.24.6
example                     3                   0                   v1.24.6
```

6.6.13.3 Scale vSphere Node Pools

Scaling node pools

While you can run Cluster Autoscaler(see page 295), you can also manually scale your node pools up or down when you need more finite control over your environment. For example, if you require 10 machines to run a process, you can manually set the scaling to run those 10 machines only. However, if also using the Cluster Autoscaler, you must stay within your minimum and maximum bounds.

Scaling up node pools

To scale up a node pool in a cluster, run the command that follows, replacing the value 5 with the actual number of replicas you need:

```
dkp scale nodepools ${NODEPOOL_NAME} --replicas=5 --cluster-name=${CLUSTER_NAME}
```

Your output should be similar to this example, indicating the scaling is in progress:

```
INFO[2021-07-26T08:54:35-07:00] Running scale nodepool command
clusterName=demo-cluster managementClusterKubeconfig= namespace=default src="nodepool
/scale.go:82"
INFO[2021-07-26T08:54:35-07:00] Nodepool example scaled to 5 replicas
clusterName=demo-cluster managementClusterKubeconfig= namespace=default src="nodepool
/scale.go:94"
```

After a few minutes, you can list the node pools with the command:

```
dkp get nodepools --cluster-name=${CLUSTER_NAME} --kubeconfig=${CLUSTER_NAME}.conf
```

Your output should be similar to this example, with the number of DESIRED and READY replicas increased to 5:

```
NODEPOOL                    DESIRED             READY               KUBERNETES
VERSION
example                     5                   5                   v1.24.6
demo-cluster-md-0           4                   4                   v1.24.6
```

Scaling down node pools

To scale down a node pool, run the command:

```
dkp scale nodepools ${NODEPOOL_NAME} --replicas=4 --cluster-name=${CLUSTER_NAME}
```

```
INFO[2021-07-26T08:54:35-07:00] Running scale nodepool command
clusterName=demo-cluster managementClusterKubeconfig= namespace=default src="nodepool
/scale.go:82"
INFO[2021-07-26T08:54:35-07:00] Nodepool example scaled to 4 replicas
clusterName=demo-cluster managementClusterKubeconfig= namespace=default src="nodepool
/scale.go:94"
```

In a default cluster, the nodes to delete are selected at random. This behavior is controller by CAPI's delete policy[233]. However, when using the Konvoy CLI to scale down a node pool, you can specify the Kubernetes Nodes you want to delete.

To do this, set the flag `--nodes-to-delete` with a list of nodes as shown in the next command. This adds an annotation `cluster.x-k8s.io/delete-machine=yes` to the matching Machine object that contains `status.NodeRef` with the node names from `--nodes-to-delete`.

```
dkp scale nodepools ${NODEPOOL_NAME} --replicas=3 --nodes-to-delete=<> --cluster-
name=${CLUSTER_NAME}
```

```
INFO[2021-07-26T08:54:35-07:00] Running scale nodepool command
clusterName=demo-cluster managementClusterKubeconfig= namespace=default src="nodepool
/scale.go:82"
```

233 https://github.com/kubernetes-sigs/cluster-api/blob/v0.4.0/api/v1alpha4/machineset_types.go#L85-L105

```
INFO[2021-07-26T08:54:35-07:00] Nodepool example scaled to 3 replicas
clusterName=demo-cluster managementClusterKubeconfig= namespace=default src="nodepool
/scale.go:94"
```

Scaling node pools when using cluster autoscaler

If you configured the cluster autoscaler[234] for the `demo-cluster-md-0` node pool, the value of `--replicas` must be within the minimum and maximum bounds.

For example, assuming you have the these annotations:

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf annotate machinedeployment ${NODEPOOL_NAME}
cluster.x-k8s.io/cluster-api-autoscaler-node-group-min-size=2
kubectl --kubeconfig=${CLUSTER_NAME}.conf annotate machinedeployment ${NODEPOOL_NAME}
cluster.x-k8s.io/cluster-api-autoscaler-node-group-max-size=6
```

Try to scale the node pool to 7 replicas with the command:

```
dkp scale nodepools ${NODEPOOL_NAME} --replicas=7 -c demo-cluster
```

This action results in an error similar to:

```
INFO[2021-07-26T09:46:37-07:00] Running scale nodepool command
clusterName=demo-cluster managementClusterKubeconfig= namespace=default src="nodepool
/scale.go:82"
Error: failed to scale nodepool: scaling MachineDeployment is forbidden: desired
replicas 7 is greater than the configured max size annotation cluster.x-k8s.io/
cluster-api-autoscaler-node-group-max-size: 6
```

Similarly, scaling down to a number of replicas less than the configured `min-size` also returns an error.

6.6.13.4 Delete vSphere Node Pools

Deleting a node pool deletes the Kubernetes nodes and the underlying infrastructure. DKP drains all nodes prior to deletion and reschedules the pods running on those nodes.

To delete a node pool from a managed cluster, run the command:

```
dkp delete nodepool ${NODEPOOL_NAME} --cluster-name=${CLUSTER_NAME}
```

234 https://docs.d2iq.com/dkp/konvoy/2.2/choose-infrastructure/vsphere/nodepools/cluster-autoscaler

Here, `example` is the node pool to be deleted.

The expected output is similar to the following example, indicating the node pool is being deleted:

```
INFO[2021-07-28T17:14:26-07:00] Running nodepool delete command
Nodepool=example clusterName=d2iq-e2e-cluster-1 managementClusterKubeconfig=
namespace=default src="nodepool/delete.go:80"
```

Deleting an invalid node pool results in output similar to this example command output:

```
dkp delete nodepool ${CLUSTER_NAME}-md-invalid --cluster-name=${CLUSTER_NAME}

INFO[2021-07-28T17:11:44-07:00] Running nodepool delete command
Nodepool=demo-cluster-md-invalid clusterName=d2iq-e2e-cluster-1
 managementClusterKubeconfig= namespace=default src="nodepool/delete.go:80"
Error: failed to get nodepool with name demo-cluster-md-invalid in namespace default
: failed to get nodepool with name demo-cluster-md-invalid in namespace default :
machinedeployments.cluster.x-k8s.io "demo-cluster-md-invalid" not found
```

6.6.13.5 vSphere Cluster Autoscaler

Cluster Autoscaler

Cluster Autoscaler[235] provides the ability to automatically scale-up or scale-down the number of worker nodes in a cluster, based on the number of pending pods to be scheduled. Running the Cluster Autoscaler is optional.

Unlike Horizontal-Pod Autoscaler[236], Cluster Autoscaler does not depend on any Metrics server and does not need Prometheus or any other metrics source.

The Cluster Autoscaler looks at the following annotations on a MachineDeployment to determine its scale-up and scale-down ranges:

```
cluster.x-k8s.io/cluster-api-autoscaler-node-group-min-size
cluster.x-k8s.io/cluster-api-autoscaler-node-group-max-size
```

The full list of command line arguments to the Cluster Autoscaler controller is found this topic[237].

For more information about how Cluster Autoscaler works, see these documents:

- What is Cluster Autoscaler[238]

235 https://github.com/kubernetes/autoscaler/tree/master/cluster-autoscaler/cloudprovider/clusterapi
236 https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/FAQ.md#how-fast-is-hpa-when-combined-with-ca
237 https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/FAQ.md#what-are-the-parameters-to-ca
238 https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/FAQ.md#what-is-cluster-autoscaler

- How does scale-up work[239]
- How does scale-down work[240]
- CAPI Provider for Cluster Autoscaler[241]

Cluster Autoscaler Prerequisites

Before you begin, you must have:

- A Bootstrap Cluster Lifecycle(see page 257).
- Created a new Kubernetes Cluster(see page 259).
- A Self-Managed Cluster(see page 265).

Run Cluster Autoscaler

The Cluster Autoscaler controller runs on the workload cluster. Upon creation of the workload cluster, this controller does not have all the objects required to function correctly until after a `dkp move` is issued from the bootstrap cluster.

Run the following steps to enable Cluster Autoscaler:

1. Ensure the Cluster Autoscaler controller is up and running (no restarts and no errors in the logs)

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf logs deployments/cluster-autoscaler
cluster-autoscaler -n kube-system -f
```

2. Enable Cluster Autoscaler by setting the min & max ranges

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf annotate machinedeployment $
{NODEPOOL_NAME} cluster.x-k8s.io/cluster-api-autoscaler-node-group-min-size=2
kubectl --kubeconfig=${CLUSTER_NAME}.conf annotate machinedeployment $
{NODEPOOL_NAME} cluster.x-k8s.io/cluster-api-autoscaler-node-group-max-size=6
```

3. The Cluster Autoscaler logs will show that the worker nodes are associated with node-groups and that pending pods are being watched.
4. To demonstrate that it is working properly, create a large deployment which will trigger pending pods.

```
cat <<EOF | kubectl --kubeconfig=${CLUSTER_NAME}.conf apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: busybox-deployment
  labels:
    app: busybox
spec:
  replicas: 600
```

239 https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/FAQ.md#how-does-scale-up-work
240 https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/FAQ.md#how-does-scale-down-work
241 https://github.com/kubernetes/autoscaler/tree/master/cluster-autoscaler/cloudprovider/clusterapi

```
  selector:
    matchLabels:
      app: busybox
  template:
    metadata:
      labels:
        app: busybox
    spec:
      containers:
      - name: busybox
        image: busybox:latest
        command:
          - sleep
          - "3600"
        imagePullPolicy: IfNotPresent
      restartPolicy: Always
EOF
```

Cluster Autoscaler scales up the number of Worker Nodes until there are no pending pods.

5. Scale down the number of replicas for `busybox-deployment` with the command:

```
kubectl --kubeconfig ${CLUSTER_NAME}.conf scale --replicas=30 deployment/
busybox-deployment
```

6. Cluster Autoscaler starts to scale down the number of Worker Nodes after the default timeout of 10 minutes.

6.6.14 Replace a vSphere Node

6.6.14.1 Prerequisites

Before you begin, you must:

- Create a workload cluster(see page 259) or create an air-gapped workload cluster(see page 278).
- Make the new cluster self-managed(see page 265).

6.6.14.2 Replace a worker node

In certain situations, you may want to delete a worker node and have Cluster API[242] replace it with a newly-provisioned machine.

1. Identify the name of the node to delete.
   List the nodes:

```
kubectl --kubeconfig ${CLUSTER_NAME}.conf get nodes
```

242 https://cluster-api.sigs.k8s.io/

The output from this command resembles the following:

```
NAME                                          STATUS   ROLES                  AGE
VERSION
d2iq-e2e-cluster-1-control-plane-7llgd        Ready    control-plane,master   20h
v1.24.6
d2iq-e2e-cluster-1-control-plane-vncbl        Ready    control-plane,master   20h
v1.24.6
d2iq-e2e-cluster-1-control-plane-wbgrm        Ready    control-plane,master   19h
v1.24.6
d2iq-e2e-cluster-1-md-0-74c849dc8c-67rv4      Ready    <none>                 20h
v1.24.6
d2iq-e2e-cluster-1-md-0-74c849dc8c-n2skc      Ready    <none>                 20h
v1.24.6
d2iq-e2e-cluster-1-md-0-74c849dc8c-nkftv      Ready    <none>                 20h
v1.24.6
d2iq-e2e-cluster-1-md-0-74c849dc8c-sqklv      Ready    <none>                 20h
v1.24.6
```

2. Export a variable with the node name to use in the next steps:
   This example uses the name `d2iq-e2e-cluster-1-md-0-74c849dc8c-67rv4`.

```
export NAME_NODE_TO_DELETE="d2iq-e2e-cluster-1-md-0-74c849dc8c-67rv4"
```

3. Delete the Machine resource with the command:

```
NAME_MACHINE_TO_DELETE=$(kubectl --kubeconfig ${CLUSTER_NAME}.conf get machine
-ojsonpath="{.items[?
(@.status.nodeRef.name==\"$NAME_NODE_TO_DELETE\")].metadata.name}")
kubectl --kubeconfig ${CLUSTER_NAME}.conf delete machine
"$NAME_MACHINE_TO_DELETE"
```

```
machine.cluster.x-k8s.io "d2iq-e2e-cluster-1-md-0-74c849dc8c-67rv4" deleted
```

   The command does not return immediately, but it does return after the Machine resource is deleted.
   A few minutes after the Machine resource is deleted, the corresponding Node resource is also deleted.
4. Observe the Machine resource replacement using this command:

```
kubectl --kubeconfig ${CLUSTER_NAME}.conf get machinedeployment
```

```
NAME                        CLUSTER              REPLICAS   READY   UPDATED
UNAVAILABLE   PHASE       AGE   VERSION
d2iq-e2e-cluster-1-md-0   d2iq-e2e-cluster-1   4          3       4         1
ScalingUp   20h   v1.24.6
```

In this example, there exist 4 replicas, but only 3 are ready. One replica is unavailable, and the `ScalingUp` phase means a new Machine is being created.

5. Identify the replacement Machine using this command:

```
export NAME_NEW_MACHINE=$(kubectl --kubeconfig ${CLUSTER_NAME}.conf get
machines \
    -l=cluster.x-k8s.io/deployment-name=${CLUSTER_NAME}-md-0 \
    -ojsonpath='{.items[?(@.status.phase=="Provisioning")].metadata.name}
{"\n"}')
echo "$NAME_NEW_MACHINE"
```

If the output is empty, the new Machine has probably exited the `Provisioning` phase and entered the `Running` phase.

6. Identify the replacement Node using this command:

```
kubectl --kubeconfig ${CLUSTER_NAME}.conf get nodes \
    -o=jsonpath="{.items[?(@.metadata.annotations.cluster\.x-k8s\.io/
machine==\"$NAME_NEW_MACHINE\")].metadata.name}"
```

The output should be similar to this example:

```
d2iq-e2e-cluster-1-md-0-74c849dc8c-rc528
```

If the output is empty, the Node resource is not yet available, or does not yet have the expected annotation. Wait a few minutes, then repeat the command.

6.7 GCP Infrastructure

The following procedure describes creating a DKP cluster on GCP.

6.7.1 GCP Prerequisites

6.7.1.1 Prerequisites

Before beginning a DKP installation, verify that you have:

- An x86_64-based Linux or macOS machine with a supported version of the operating system.
- The `dkp` binary on this machine.
- Docker[243] version 18.09.2 or later.
- kubectl[244] for interacting with the running cluster.
- Install the GCP `gcloud` CLI by following the GCP install documentation[245].[246]

6.7.1.2 Control plane nodes

You must have at least three control plane nodes. Each control plane node should have at least:

- 4 cores
- 16 GiB memory
- Approximately 80 GiB of free space for the volume used for /var/lib/kubelet and /var/lib/containerd.
- Disk usage must be below 85% on the root volume.

DKP on GCP defaults to deploying an `n2-standard-4` instance with an 80GiB root volume for control plane nodes, which meets the above requirements.

6.7.1.3 Worker nodes

You must have at least four worker nodes. The specific number of worker nodes required for your environment can vary depending on the cluster workload and size of the nodes. Each worker node should have at least:

- 8 cores
- 32 GiB memory
- Around 80 GiB of free space for the volume used for /var/lib/kubelet and /var/lib/containerd.
- Disk usage must be below 85% on the root volume.

DKP on GCP defaults to deploying a `n2-standard-8` instance with an 80GiB root volume for worker nodes, which meets the above requirements.

6.7.1.4 GCP Prerequisites

- If you are creating the bootstrap cluster on a non-GCP instance or one that does not have the required `editor` role:
    - (option 1) Create a service account using the following `gcloud` commands:

243 https://docs.docker.com/get-docker/
244 https://kubernetes.io/docs/tasks/tools/#kubectl
245 https://cloud.google.com/sdk/docs/install
246 https://cloud.google.com/sdk/docs/install.

```
export GCP_PROJECT=<your GCP project ID>
export SERVICE_ACCOUNT_USER=<some new service account user>
export GOOGLE_APPLICATION_CREDENTIALS="$HOME/.gcloud/credentials.json"

gcloud iam service-accounts create "$SERVICE_ACCOUNT_USER" --
project=$GCP_PROJECT
gcloud projects add-iam-policy-binding $GCP_PROJECT --
member="serviceAccount:
$SERVICE_ACCOUNT_USER@$GCP_PROJECT.iam.gserviceaccount.com" --role=roles/
editor
gcloud iam service-accounts keys create $GOOGLE_APPLICATION_CREDENTIALS --
iam-account="$SERVICE_ACCOUNT_USER@$GCP_PROJECT.iam.gserviceaccount.com"
```

- (option 2) Retrieve the credentials for an existing service account using the following `gcloud` commands:

```
export GCP_PROJECT=<your GCP project ID>
export SERVICE_ACCOUNT_USER=<existing service account user>
export GOOGLE_APPLICATION_CREDENTIALS="$HOME/.gcloud/credentials.json"

gcloud iam service-accounts keys create $GOOGLE_APPLICATION_CREDENTIALS --
iam-account="$SERVICE_ACCOUNT_USER@$GCP_PROJECT.iam.gserviceaccount.com"
```

- Export the static credentials that will be used to create the cluster:

```
export GCP_B64ENCODED_CREDENTIALS=$(base64 < "$
{GOOGLE_APPLICATION_CREDENTIALS}" | tr -d '\n')
```

6.7.2 Bootstrap GCP

To create Kubernetes clusters, DKP uses Cluster API[247] (CAPI) controllers. These controllers run on a Kubernetes cluster. To get started, you need a *bootstrap* cluster. By default, DKP creates a bootstrap cluster for you in a Docker container using the Kubernetes-in-Docker (KIND[248]) tool.

6.7.2.1 Prerequisites

Before you begin, you must:

- Complete the steps in Prerequisites(see page 300).
- Ensure the `dkp` binary can be found in your `$PATH`.

247 https://cluster-api.sigs.k8s.io/
248 https://github.com/kubernetes-sigs/kind

6.7.2.2 Bootstrap Cluster Lifecycle Services

1. If an HTTP proxy is required for the bootstrap cluster, set the local `http_proxy`, `https_proxy`, and `no_proxy` environment variables. They are copied into the bootstrap cluster.
2. Create a bootstrap cluster:

```
dkp create bootstrap --with-gcp-bootstrap-credentials=true
```

3. The output looks similar to:

```
✓ Creating a bootstrap cluster
✓ Initializing new CAPI components
```

4. DKP creates a bootstrap cluster using KIND[249] as a library. DKP then deploys the following Cluster API[250] providers on the cluster:
   - Core Provider[251]
   - GCP Infrastructure Provider[252]
   - Kubeadm Bootstrap Provider[253]
   - Kubeadm ControlPlane Provider[254]
5. DKP waits until the controller-manager and webhook deployments of these providers are ready. List these deployments using this command:

```
kubectl get --all-namespaces deployments -l=clusterctl.cluster.x-k8s.io
```

   You will then receive the following output:

```
NAMESPACE                           NAME
READY   UP-TO-DATE   AVAILABLE   AGE
capa-system                         capa-controller-manager
1/1     1            1           1h
capg-system                         capg-controller-manager
1/1     1            1           1h
capi-kubeadm-bootstrap-system       capi-kubeadm-bootstrap-controller-manager
1/1     1            1           1h
capi-kubeadm-control-plane-system   capi-kubeadm-control-plane-controller-
manager   1/1     1            1           1h
capi-system                         capi-controller-manager
1/1     1            1           1h
cappp-system                        cappp-controller-manager
1/1     1            1           1h
```

249 https://github.com/kubernetes-sigs/kind
250 https://cluster-api.sigs.k8s.io/
251 https://github.com/kubernetes-sigs/cluster-api/tree/v0.3.20/
252 https://github.com/kubernetes-sigs/cluster-api-provider-gcp
253 https://github.com/kubernetes-sigs/cluster-api/tree/v0.3.20/bootstrap/kubeadm
254 https://github.com/kubernetes-sigs/cluster-api/tree/v0.3.20/controlplane/kubeadm

```
capv-system                             capv-controller-manager
1/1     1            1           1h
capz-system                             capz-controller-manager
1/1     1            1           1h
cert-manager                            cert-manager
1/1     1            1           1h
cert-manager                            cert-manager-cainjector
1/1     1            1           1h
cert-manager                            cert-manager-webhook
1/1     1            1           1h
```

Once completed, move onto creating a new GCP cluster(see page 303).

6.7.3 Create a New GCP Cluster

6.7.3.1 Prerequisites

- Before you begin, ensure you create a Bootstrap(see page 301) cluster.

6.7.3.2 Name your Cluster

1. Give your cluster a unique name suitable for your environment.
   In GCP it is critical that the name is unique, as no two clusters in the same GCP account can have the same name.
2. Set the environment variable:

```
export CLUSTER_NAME=gcp-example
```

The cluster name may only contain the following characters: `a-z`, `0-9`, `.`, and `-`. Cluster creation will fail if the name has capital letters. See Kubernetes[255] for more naming information.

To increase Docker Hub's rate limit[256] use your Docker Hub credentials when creating the cluster, by setting the following flag `--registry-mirror-url=https://registry-1.docker.io --registry-mirror-username=<username> --registry-mirror-password=<password>` on the `dkp create cluster` command.

6.7.3.3 Create a New GCP Cluster

Availability zones (AZs) are isolated locations within data center regions from which public cloud services originate and operate. Because all the nodes in a node pool are deployed in a single Availability Zone, you may wish to create additional node pools is to ensure your cluster has nodes deployed in multiple Availability Zones.

---

255 https://kubernetes.io/docs/concepts/overview/working-with-objects/names/
256 https://docs.docker.com/docker-hub/download-rate-limit/

By default, the control-plane Nodes will be created in 3 different zones. However, the default worker Nodes will reside in a single zone. You may create additional node pools in other zones with the `dkp create nodepool` command. The default region for the availability zones is `us-west1`.

1. Create an image using Konvoy Image Builder (KIB)(see page 328) and then export the image name:

```
export IMAGE_NAME=projects/${GCP_PROJECT}/global/images/<image_name_from_kib>
```

2. (Optional) You can modify the Control Plane Audit logs settings using the information contained in the page Configuring the Control Plane(see page 303).
3. (Optional) Determine what VPC Network to use. All GCP accounts come with a preconfigured VPC Network named `default`, which will be used if you do not specify a different network.
   To use a different VPC network for your cluster, create one by following these instructions for Create and Manage VPC Networks[257]. Then specify the `--network <new_vpc_network_name>` option on the create cluster command below. Follow the link for more information on GCP Cloud Nat[258] and network flag.
4. Create a Kubernetes cluster. During create a new cluster command, the user can add flags. One important flag is `--network`. If the default network is not cloud nat and you would like it to be, then the `--network` flag needs to be set to true. This would enable a hidden or non-public IP address:

```
dkp create cluster gcp \
--cluster-name=${CLUSTER_NAME} \
--additional-tags=owner=$(whoami) \
--project=${GCP_PROJECT} \
--image=${IMAGE_NAME}
```

5. Wait for the cluster control-plane to be ready:

```
kubectl wait --for=condition=ControlPlaneReady "clusters/${CLUSTER_NAME}" --
timeout=20m
```

6. After the objects are created on the API server, the Cluster API controllers reconcile them. They create infrastructure and machines. As they progress, they update the Status of each object. Konvoy provides a command to describe the current status of the cluster:

```
dkp describe cluster -c ${CLUSTER_NAME}
```

```
NAME                                                                  READY
SEVERITY  REASON  SINCE  MESSAGE
```

257 https://cloud.google.com/vpc/docs/create-modify-vpc-networks#create-auto-network
258 https://github.com/kubernetes-sigs/cluster-api-provider-gcp/blob/3406adaae0f8f65a615844e55d856d306eddacc1/docs/book/src/topics/prerequisites.md#cloud-nat

```
Cluster/gcp-example                                                    True
52s
├──ClusterInfrastructure - GCPCluster/gcp-example
├──ControlPlane - KubeadmControlPlane/gcp-example-control-plane        True
52s
│  ├──Machine/gcp-example-control-plane-6fbzn                          True
2m32s
│  │  └──MachineInfrastructure - GCPMachine/gcp-example-control-plane-62g6s
│  ├──Machine/gcp-example-control-plane-jf6s2                          True
7m36s
│  │  └──MachineInfrastructure - GCPMachine/gcp-example-control-plane-bsr2z
│  └──Machine/gcp-example-control-plane-mnbfs                          True
54s
│     └──MachineInfrastructure - GCPMachine/gcp-example-control-plane-s8xsx
└──Workers
   └──MachineDeployment/gcp-example-md-0                               True
78s
      ├──Machine/gcp-example-md-0-68b86fddb8-8glsw                     True
2m49s
      │  └──MachineInfrastructure - GCPMachine/gcp-example-md-0-zls8d
      ├──Machine/gcp-example-md-0-68b86fddb8-bvbm7                     True
2m48s
      │  └──MachineInfrastructure - GCPMachine/gcp-example-md-0-5zcvc
      ├──Machine/gcp-example-md-0-68b86fddb8-k9499                     True
2m49s
      │  └──MachineInfrastructure - GCPMachine/gcp-example-md-0-k8h5p
      └──Machine/gcp-example-md-0-68b86fddb8-l6vfb                     True
2m49s
         └──MachineInfrastructure - GCPMachine/gcp-example-md-0-9h5vn
```

Once the cluster creation process completes, move on to exploring your new cluster.(see page 305)

6.7.4 Explore the GCP Cluster

This guide explains how to use the command line to interact with your newly deployed Kubernetes cluster.

Before you start, make sure you have created a workload cluster, as described in Create a New GCP Cluster(see page 303).

6.7.4.1 Explore the new Kubernetes cluster

1. Get a kubeconfig file for the workload cluster:
   When the workload cluster is created, the cluster lifecycle services generate a kubeconfig file for the workload cluster, and write it to a *Secret*. The kubeconfig file is scoped to the cluster administrator.

   ```
   dkp get kubeconfig -c ${CLUSTER_NAME} > ${CLUSTER_NAME}.conf
   ```

2. Verify the API server is up :

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf get nodes
```

NOTE: It may take a few minutes for the Status to move to Ready while the Pod network is deployed. The Nodes' Status should change to Ready soon after the calico-node DaemonSet Pods are Ready.
You should receive the following output:

```
NAME                                STATUS   ROLES                  AGE
VERSION
gcp-example-control-plane-9z77w   Ready    control-plane,master   4m44s
v1.24.6
gcp-example-control-plane-rtj9h   Ready    control-plane,master   104s
v1.24.6
gcp-example-control-plane-zbf9w   Ready    control-plane,master   3m23s
v1.24.6
gcp-example-md-0-88c46            Ready    <none>                 3m28s   v1.24
.6
gcp-example-md-0-fp8s7            Ready    <none>                 3m28s   v1.24
.6
gcp-example-md-0-qvnx7            Ready    <none>                 3m28s   v1.24
.6
gcp-example-md-0-wjdrg            Ready    <none>                 3m27s   v1.24
.6
```

3. List the Pods with the command:

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf get pods -A
```

4. View the following output:

```
NAMESPACE                           NAME
READY   STATUS    RESTARTS       AGE
calico-system                       calico-kube-controllers-577c696df9-v2nzv
1/1     Running   0              5m23s
calico-system                       calico-node-4x5rk
1/1     Running   0              4m22s
calico-system                       calico-node-cxsgc
1/1     Running   0              4m23s
calico-system                       calico-node-dvlnm
1/1     Running   0              4m23s
calico-system                       calico-node-h6nlt
1/1     Running   0              4m23s
calico-system                       calico-node-jmkwq
1/1     Running   0              5m23s
calico-system                       calico-node-tnf54
1/1     Running   0              4m18s
calico-system                       calico-node-v6bwq
1/1     Running   0              2m39s
calico-system                       calico-typha-6d8c94bfdf-dkfvq
1/1     Running   0              5m23s
```

```
calico-system                       calico-typha-6d8c94bfdf-fdfn2
1/1     Running   0             3m43s
calico-system                       calico-typha-6d8c94bfdf-kjgzj
1/1     Running   0             3m43s
capa-system                         capa-controller-manager-6468bc488-w7nj9
1/1     Running   0             67s
capg-system                         capg-controller-manager-5fb47f869b-6jgms
1/1     Running   0             53s
capi-kubeadm-bootstrap-system       capi-kubeadm-bootstrap-controller-
manager-65ffc94457-7cjdn      1/1     Running   0             74s
capi-kubeadm-control-plane-system   capi-kubeadm-control-plane-controller-
manager-bc7b688d4-vv8wg   1/1     Running   0             72s
capi-system                         capi-controller-manager-dbfc7b49-dzvw8
1/1     Running   0             77s
cappp-system                        cappp-controller-manager-8444d67568-rmms2
1/1     Running   0             59s
capv-system                         capv-controller-manager-58b8ccf868-rbscn
1/1     Running   0             56s
capz-system                         capz-controller-manager-6467f986d8-dnvj4
1/1     Running   0             62s
cert-manager                        cert-manager-6888d6b69b-7b7m9
1/1     Running   0             91s
cert-manager                        cert-manager-cainjector-76f7798c9-gnp8f
1/1     Running   0             91s
cert-manager                        cert-manager-webhook-7d4b5d8484-gn5dr
1/1     Running   0             91s
gce-pd-csi-driver                   csi-gce-pd-controller-5bd587fbfb-lrx29
5/5     Running   0             5m40s
gce-pd-csi-driver                   csi-gce-pd-node-4cgd8
2/2     Running   0             4m22s
gce-pd-csi-driver                   csi-gce-pd-node-5qsfk
2/2     Running   0             4m23s
gce-pd-csi-driver                   csi-gce-pd-node-5w4bq
2/2     Running   0             4m18s
gce-pd-csi-driver                   csi-gce-pd-node-fbdbw
2/2     Running   0             4m23s
gce-pd-csi-driver                   csi-gce-pd-node-h82lx
2/2     Running   0             4m23s
gce-pd-csi-driver                   csi-gce-pd-node-jzq58
2/2     Running   0             5m39s
gce-pd-csi-driver                   csi-gce-pd-node-k6bz9
2/2     Running   0             2m39s
kube-system                         cluster-autoscaler-7f695dc48f-v5kvh
1/1     Running   0             5m40s
kube-system                         coredns-64897985d-hbkqd
1/1     Running   0             5m38s
kube-system                         coredns-64897985d-m8g5j
1/1     Running   0             5m38s
kube-system                         etcd-gcp-example-control-plane-9z77w
1/1     Running   0             5m32s
kube-system                         etcd-gcp-example-control-plane-rtj9h
1/1     Running   0             2m37s
```

```
kube-system                         etcd-gcp-example-control-plane-zbf9w
1/1     Running   0             4m17s
kube-system                         kube-apiserver-gcp-example-control-
plane-9z77w                 1/1     Running   0             5m32s
kube-system                         kube-apiserver-gcp-example-control-plane-
rtj9h                   1/1     Running   0             2m38s
kube-system                         kube-apiserver-gcp-example-control-plane-
zbf9w                   1/1     Running   0             4m17s
kube-system                         kube-controller-manager-gcp-example-
control-plane-9z77w         1/1     Running   0             5m33s
kube-system                         kube-controller-manager-gcp-example-
control-plane-rtj9h         1/1     Running   0             2m37s
kube-system                         kube-controller-manager-gcp-example-
control-plane-zbf9w         1/1     Running   0             4m17s
kube-system                         kube-proxy-bskz2
1/1     Running   0             4m18s
kube-system                         kube-proxy-gdkn5
1/1     Running   0             4m23s
kube-system                         kube-proxy-knvb9
1/1     Running   0             4m22s
kube-system                         kube-proxy-tcj7r
1/1     Running   0             4m23s
kube-system                         kube-proxy-thdpl
1/1     Running   0             5m38s
kube-system                         kube-proxy-txxmb
1/1     Running   0             4m23s
kube-system                         kube-proxy-vq6kv
1/1     Running   0             2m39s
kube-system                         kube-scheduler-gcp-example-control-
plane-9z77w                 1/1     Running   0             5m33s
kube-system                         kube-scheduler-gcp-example-control-plane-
rtj9h                   1/1     Running   0             2m37s
kube-system                         kube-scheduler-gcp-example-control-plane-
zbf9w                   1/1     Running   0             4m17s
node-feature-discovery              node-feature-discovery-master-7d5985467-
lh7dc                    1/1     Running   0             5m40s
node-feature-discovery              node-feature-discovery-worker-5qtvg
1/1     Running   0             3m40s
node-feature-discovery              node-feature-discovery-worker-66rwx
1/1     Running   0             3m40s
node-feature-discovery              node-feature-discovery-worker-7h92d
1/1     Running   0             3m35s
node-feature-discovery              node-feature-discovery-worker-b4666
1/1     Running   0             3m40s
tigera-operator                     tigera-operator-5f9bdc5c59-j9tnr
1/1     Running   0             5m38s
```

When you're done exploring the cluster, move onto making your cluster self-managed.(see page 309)

6.7.5 Make the New GCP Cluster Self-Managed

DKP deploys all cluster lifecycle services to a bootstrap cluster, which then deploys a workload cluster. When the workload cluster is ready, move the cluster lifecycle services to the workload cluster, which makes the workload cluster self-managed. This section describes how to make a workload cluster self-managed.

Before starting, ensure you create a workload cluster as described in Create a New GCP Cluster(see page 303).

6.7.5.1 Make the New Kubernetes Cluster Manage Itself

Follow these steps:

1. Deploy cluster lifecycle services on the workload cluster:

```
dkp create capi-components --kubeconfig ${CLUSTER_NAME}.conf
```

```
✓ Initializing new CAPI components
```

2. Move the Cluster API objects from the bootstrap to the workload cluster:
The cluster lifecycle services on the workload cluster are ready, but the workload cluster configuration is on the bootstrap cluster. The `move` command moves the configuration, which takes the form of Cluster API Custom Resource objects, from the bootstrap to the workload cluster. This process is also called a Pivot[259].

```
dkp move capi-resources --to-kubeconfig ${CLUSTER_NAME}.conf
```

```
✓ Moving cluster resources
You can now view resources in the moved cluster by using the --kubeconfig flag
with kubectl. For example: kubectl --kubeconfig=gcp-example.conf get nodes
```

NOTE: To ensure only one set of cluster lifecycle services manages the workload cluster, DKP first pauses reconciliation of the objects on the bootstrap cluster, then creates the objects on the workload cluster. As DKP copies the objects, the cluster lifecycle services on the workload cluster reconcile the objects. The workload cluster becomes self-managed after DKP creates all the objects. If it fails, the `move` command can be safely retried.

3. Wait for the cluster control-plane to be ready:

```
kubectl --kubeconfig ${CLUSTER_NAME}.conf wait --for=condition=ControlPlaneRead
y "clusters/${CLUSTER_NAME}" --timeout=20m
```

259 https://cluster-api.sigs.k8s.io/reference/glossary.html?highlight=pivot#pivot

```
cluster.cluster.x-k8s.io/gcp-example condition met
```

4. Use the cluster lifecycle services on the workload cluster to check the workload cluster status:
NOTE: After moving the cluster lifecycle services to the workload cluster, remember to use DKP with the workload cluster kubeconfig.

```
dkp describe cluster --kubeconfig ${CLUSTER_NAME}.conf -c ${CLUSTER_NAME}
```

```
NAME                                                                    READY
SEVERITY  REASON  SINCE  MESSAGE
Cluster/gcp-example                                                     True
14s
├──ClusterInfrastructure - GCPCluster/gcp-example
├──ControlPlane - KubeadmControlPlane/gcp-example-control-plane          True
14s
│ ├──Machine/gcp-example-control-plane-6fbzn                             True
17s
│ │ └──MachineInfrastructure - GCPMachine/gcp-example-control-plane-62g6s
│ ├──Machine/gcp-example-control-plane-jf6s2                             True
17s
│ │ └──MachineInfrastructure - GCPMachine/gcp-example-control-plane-bsr2z
│ └──Machine/gcp-example-control-plane-mnbfs                             True
17s
│   └──MachineInfrastructure - GCPMachine/gcp-example-control-plane-s8xsx
└──Workers
   └──MachineDeployment/gcp-example-md-0                                 True
17s
    ├──Machine/gcp-example-md-0-68b86fddb8-8glsw                        True
17s
    │ └──MachineInfrastructure - GCPMachine/gcp-example-md-0-zls8d
    ├──Machine/gcp-example-md-0-68b86fddb8-bvbm7                        True
17s
    │ └──MachineInfrastructure - GCPMachine/gcp-example-md-0-5zcvc
    ├──Machine/gcp-example-md-0-68b86fddb8-k9499                        True
17s
    │ └──MachineInfrastructure - GCPMachine/gcp-example-md-0-k8h5p
    └──Machine/gcp-example-md-0-68b86fddb8-l6vfb                        True
17s
      └──MachineInfrastructure - GCPMachine/gcp-example-md-0-9h5vn
```

5. Remove the bootstrap cluster, as the workload cluster is now self-managed:

```
dkp delete bootstrap --kubeconfig $HOME/.kube/config
```

```
✓ Deleting bootstrap cluster
```

6.7.5.2 Known Limitations

Be aware of these limitations in the current release of DKP.

DKP only supports moving all namespaces in the cluster; DKP does not support migration of individual namespaces.

6.7.6 Manage GCP Node Pools

Node pools are part of a cluster and managed as a group, and you can use a node pool to manage a group of machines using the same common properties. When Konvoy creates a new default cluster, there is one node pool for the worker nodes and all nodes in that new node pool have the same configuration. You can create additional node pools for more specialized hardware or configuration. For example, if you want to tune your memory usage on a cluster where you need maximum memory for some machines and minimal memory on other machines, you would create a new node pool with those specific resource needs.

Konvoy implements node pools using Cluster API MachineDeployments[260].

6.7.6.1 Create GCP Node Pools

Creating a node pool is useful when you need to run workloads that require machines with specific resources, such as additional memory, or specialized network or storage hardware.

Prerequisites

Before you begin, make sure you have created a GCP cluster.(see page 303)

Prepare the environment

Follow these steps:

1. Set the environment variable to the name you assigned this cluster.

   ```
   export CLUSTER_NAME=gcp-example
   ```

2. If your workload cluster is self-managed, as described in Make the New Cluster Self-Managed(see page 309), configure `kubectl` to use the kubeconfig for the cluster.

   ```
   export KUBECONFIG=${CLUSTER_NAME}.conf
   ```

3. Define your node pool name.

260 https://cluster-api.sigs.k8s.io/developer/architecture/controllers/machine-deployment.html

```
export NODEPOOL_NAME=example
```

Create a GCP node pool

Availability zones (AZs) are isolated locations within data center regions from which public cloud services originate and operate. Because all the nodes in a node pool are deployed in a single Availability Zone, you may wish to create additional node pools is to ensure your cluster has nodes deployed in multiple Availability Zones.

Create a new AWS node pool with 3 replicas using this command:

Set the `--zone` flag to a zone[261] in the same same region as your cluster.

```
dkp create nodepool gcp ${NODEPOOL_NAME} \
    --cluster-name=${CLUSTER_NAME} \
    --image $IMAGE_NAME \
    --zone us-west1-b \
    --replicas=3
```

```
machinedeployment.cluster.x-k8s.io/example created
⋯ Creating default/example nodepool resources
gcpmachinetemplate.infrastructure.cluster.x-k8s.io/example created
kubeadmconfigtemplate.bootstrap.cluster.x-k8s.io/example created
 ✓ Creating default/example nodepool resources
```

This example uses default values for brevity. Use flags to define custom instance types, and other properties.

Advanced users can use a combination of the `--dry-run` and `--output=yaml` flags to get a complete set of node pool objects to modify locally or store in version control

6.7.6.2 List GCP Node Pools

List node pools for a cluster

Use this command to list the node pools of a given cluster. This returns specific properties of each node pool so that you can see the name of the MachineDeployments.

To list all node pools for a managed cluster, run:

```
dkp get nodepool --cluster-name=${CLUSTER_NAME}
```

The expected output is similar to the following example, indicating the desired size of the node pool, the number of replicas ready in the node pool, and the Kubernetes version those nodes are running:

```
NODEPOOL                        DESIRED              READY                KUBERNETES
VERSION
```

261 https://cloud.google.com/compute/docs/regions-zones

```
example                    3                3                v1.24.6
gcp-example-2-md-0         4                4                v1.24.6
```

6.7.6.3 Scale GCP Node Pools

While you can run Cluster Autoscaler[262], you can also manually scale your node pools up or down when you need more finite control over your environment. For example, if you require 10 machines to run a process, you can manually set the scaling to run those 10 machines only. However, if also using the Cluster Autoscaler, you must stay within your minimum and maximum bounds.

Scaling Up Node Pools

To scale up a node pool in a cluster, run:

```
dkp scale nodepool ${NODEPOOL_NAME} --replicas=5 --cluster-name=${CLUSTER_NAME}
```

Your output should be similar to this example, indicating the scaling is in progress:

```
✓ Scaling node pool example to 5 replicas
```

After a few minutes, you can list the node pools to:

```
dkp get nodepool --cluster-name=${CLUSTER_NAME}
```

Your output should be similar to this example, with the number of DESIRED and READY replicas increased to 5:

```
NODEPOOL                   DESIRED          READY
KUBERNETES VERSION
example                    5                5                v1.24.6
gcp-example-md-0           4                4                v1.24.6
```

Scaling Down Node Pools

To scale down a node pool, run:

```
dkp scale nodepool ${NODEPOOL_NAME} --replicas=4 --cluster-name=${CLUSTER_NAME}
```

```
✓ Scaling node pool example to 4 replicas
```

After a few minutes, you can list the node pools using this command:

262 https://github.com/kubernetes/autoscaler/tree/master/cluster-autoscaler/cloudprovider/clusterapi

```
dkp get nodepool --cluster-name=${CLUSTER_NAME}
```

Your output should be similar to this example, with the number of DESIRED and READY replicas decreased to 4:

```
NODEPOOL                          DESIRED              READY
KUBERNETES VERSION
example                           4                    4                    v1.22.8
gcp-example-md-0                  4                    4                    v1.22.8
```

In a default cluster, the nodes to delete are selected at random. This behavior is controlled by CAPI's delete policy[263]. However, when using the DKP CLI to scale down a node pool, it is also possible to specify the Kubernetes Nodes you want to delete.

To do this, set the flag `--nodes-to-delete` with a list of nodes as below. This adds an annotation `cluster.x-k8s.io/delete-machine=yes` to the matching Machine object that contains `status.NodeRef` with the node names from `--nodes-to-delete`.

```
dkp scale nodepool ${NODEPOOL_NAME} --replicas=3 --cluster-name=${CLUSTER_NAME}
```

```
✓ Scaling node pool example to 3 replicas
```

Scaling Node Pools When Using Cluster Autoscaler

If you configured the cluster autoscaler[264] for the `demo-cluster-md-0` node pool, the value of `--replicas` must be within the minimum and maximum bounds.

For example, assuming you have these annotations:

```
kubectl annotate machinedeployment ${NODEPOOL_NAME} cluster.x-k8s.io/cluster-api-
autoscaler-node-group-min-size=2
kubectl annotate machinedeployment ${NODEPOOL_NAME} cluster.x-k8s.io/cluster-api-
autoscaler-node-group-max-size=6
```

Try to scale the node pool to 7 replicas with the command:

```
dkp scale nodepool ${NODEPOOL_NAME} --replicas=7 --cluster-name=${CLUSTER_NAME}
```

Which results in an error similar to:

```
✗ Scaling node pool example to 7 replicas
```

263 https://github.com/kubernetes-sigs/cluster-api/blob/v0.4.0/api/v1alpha4/machineset_types.go#L85-L105
264 https://github.com/kubernetes/autoscaler/tree/master/cluster-autoscaler/cloudprovider/clusterapi

```
failed to scale nodepool: scaling MachineDeployment is forbidden: desired replicas 7
 is greater than the configured max size annotation cluster.x-k8s.io/cluster-api-
autoscaler-node-group-max-size: 6
```

Similarly, scaling down to a number of replicas less than the configured `min-size` also returns an error.

6.7.6.4 Delete GCP Node Pools

Delete node pools in a cluster

Deleting a node pool deletes the Kubernetes nodes and the underlying infrastructure. All nodes will be drained prior to deletion and the pods running on those nodes will be rescheduled.

To delete a node pool from a managed cluster, run:

```
dkp delete nodepool ${NODEPOOL_NAME} --cluster-name=${CLUSTER_NAME}
```

Here, `example` is the node pool to be deleted.

The expected output will be similar to the following example, indicating the node pool is being deleted:

```
✓ Deleting default/example nodepool resources
```

Deleting an invalid node pool results in output similar to this example:

```
dkp delete nodepool ${CLUSTER_NAME}-md-invalid --cluster-name=${CLUSTER_NAME}
```

```
MachineDeployments or MachinePools.infrastructure.cluster.x-k8s.io "no
MachineDeployments or MachinePools found for cluster gcp-example" not found
```

6.7.6.5 GCP Cluster Autoscaler

Configure autoscaler for node pools

Cluster Autoscaler[265] provides the ability to automatically scale-up or scale-down the number of worker nodes in a cluster, based on the number of pending pods to be scheduled. Running the Cluster Autoscaler is optional.

Unlike Horizontal-Pod Autoscaler[266], Cluster Autoscaler does not depend on any Metrics server and does not need Prometheus or any other metrics source.

The Cluster Autoscaler looks at the following annotations on a MachineDeployment to determine its scale-up and scale-down ranges:

265 https://github.com/kubernetes/autoscaler/tree/master/cluster-autoscaler/cloudprovider/clusterapi
266 https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/FAQ.md#how-fast-is-hpa-when-combined-with-ca

```
cluster.x-k8s.io/cluster-api-autoscaler-node-group-min-size
cluster.x-k8s.io/cluster-api-autoscaler-node-group-max-size
```

The full list of command line arguments to the Cluster Autoscaler controller is on the Kubernetes public GitHub repository[267].

For more information about how Cluster Autoscaler works, see these documents:

- What is Cluster Autoscaler[268]
- How does scale-up work[269]
- How does scale-down work[270]
- CAPI Provider for Cluster Autoscaler[271]

Cluster Autoscaler Prerequisites

Before you begin, you must have:

- A Bootstrap Cluster Lifecycle(see page 301).
- A New Kubernetes Cluster(see page 303).
- A Self-Managed Cluster(see page 309).

Run Cluster Autoscaler

The Cluster Autoscaler controller runs on the workload cluster. Upon creation of the workload cluster, this controller does not have all the objects required to function correctly until after a `dkp move` is issued from the bootstrap cluster.

Run the following steps to enable Cluster Autoscaler:

1. Ensure the Cluster Autoscaler controller is up and running (no restarts and no errors in the logs)

   ```
   kubectl --kubeconfig=${CLUSTER_NAME}.conf logs deployments/cluster-autoscaler
   cluster-autoscaler -n kube-system -f
   ```

2. Enable Cluster Autoscaler by setting the min & max ranges

   ```
   kubectl --kubeconfig=${CLUSTER_NAME}.conf annotate machinedeployment $
   {NODEPOOL_NAME} cluster.x-k8s.io/cluster-api-autoscaler-node-group-min-size=2
   kubectl --kubeconfig=${CLUSTER_NAME}.conf annotate machinedeployment $
   {NODEPOOL_NAME} cluster.x-k8s.io/cluster-api-autoscaler-node-group-max-size=6
   ```

3. The Cluster Autoscaler logs will show that the worker nodes are associated with node-groups and that pending pods are being watched.

---

267 https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/FAQ.md#what-are-the-parameters-to-ca
268 https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/FAQ.md#what-is-cluster-autoscaler
269 https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/FAQ.md#how-does-scale-up-work
270 https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/FAQ.md#how-does-scale-down-work
271 https://github.com/kubernetes/autoscaler/tree/master/cluster-autoscaler/cloudprovider/clusterapi

4. To demonstrate that it is working properly, create a large deployment which will trigger pending pods (For this example we used GCP n2-standard-8 worker nodes. If you have larger worker-nodes, you should scale up the number of replicas accordingly).

```
cat <<EOF | kubectl --kubeconfig=${CLUSTER_NAME}.conf apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: busybox-deployment
  labels:
    app: busybox
spec:
  replicas: 600
  selector:
    matchLabels:
      app: busybox
  template:
    metadata:
      labels:
        app: busybox
    spec:
      containers:
      - name: busybox
        image: busybox:latest
        command:
          - sleep
          - "3600"
        imagePullPolicy: IfNotPresent
      restartPolicy: Always
EOF
```

5. Cluster Autoscaler will scale up the number of Worker Nodes until there are no pending pods.
6. Scale down the number of replicas for `busybox-deployment`.

```
kubectl --kubeconfig ${CLUSTER_NAME}.conf scale --replicas=30 deployment/
busybox-deployment
```

7. Cluster Autoscaler starts to scale down the number of Worker Nodes after the default timeout of 10 minutes.

6.7.7 Delete a GCP Cluster

6.7.7.1 Prepare to delete a self-managed workload cluster

A self-managed workload cluster cannot delete itself. If your workload cluster is self-managed, you must create a bootstrap cluster and move the cluster lifecycle services to the bootstrap cluster before deleting the workload cluster.

If you did not make your workload cluster self-managed, as described in Make the New GCP Cluster Self-Managed(see page 309), proceed to the Delete the workload cluster(see page 0) section below.

1. Create a bootstrap cluster:
   The bootstrap cluster will host the Cluster API controllers that reconcile the cluster objects marked for deletion:
   NOTE: To avoid using the wrong `kubeconfig`, the following steps use explicit `kubeconfig` paths and contexts.

```
dkp create bootstrap --with-gcp-bootstrap-credentials=true --kubeconfig
$HOME/.kube/config
```

```
✓ Creating a bootstrap cluster
✓ Initializing new CAPI components
```

2. Move the Cluster API objects from the workload to the bootstrap cluster: The cluster lifecycle services on the bootstrap cluster are ready, but the workload cluster configuration is on the workload cluster. The `move` command moves the configuration, which takes the form of Cluster API Custom Resource objects, from the workload to the bootstrap cluster. This process is also called a Pivot[272].

```
dkp move capi-resources \
    --from-kubeconfig ${CLUSTER_NAME}.conf \
    --to-kubeconfig $HOME/.kube/config
```

3. Use the cluster lifecycle services on the workload cluster to check the workload cluster status:

```
dkp describe cluster --kubeconfig $HOME/.kube/config -c ${CLUSTER_NAME}
```

```
NAME                                                                    READY
SEVERITY  REASON  SINCE  MESSAGE
Cluster/gcp-example                                                     True
34s
├──ClusterInfrastructure - GCPCluster/gcp-example
├──ControlPlane - KubeadmControlPlane/gcp-example-control-plane          True
34s
│ ├──Machine/gcp-example-control-plane-6fbzn                             True
37s
│ │ └──MachineInfrastructure - GCPMachine/gcp-example-control-plane-62g6s
│ ├──Machine/gcp-example-control-plane-jf6s2                             True
37s
│ │ └──MachineInfrastructure - GCPMachine/gcp-example-control-plane-bsr2z
```

272 https://cluster-api.sigs.k8s.io/reference/glossary.html?highlight=pivot#pivot

```
│ └─Machine/gcp-example-control-plane-mnbfs                                  True
37s
│   └─MachineInfrastructure - GCPMachine/gcp-example-control-plane-s8xsx
└─Workers
  └─MachineDeployment/gcp-example-md-0                                       True
37s
    ├─Machine/gcp-example-md-0-68b86fddb8-8glsw                              True
37s
    │ └─MachineInfrastructure - GCPMachine/gcp-example-md-0-zls8d
    ├─Machine/gcp-example-md-0-68b86fddb8-bvbm7                              True
37s
    │ └─MachineInfrastructure - GCPMachine/gcp-example-md-0-5zcvc
    ├─Machine/gcp-example-md-0-68b86fddb8-k9499                              True
37s
    │ └─MachineInfrastructure - GCPMachine/gcp-example-md-0-k8h5p
    └─Machine/gcp-example-md-0-68b86fddb8-l6vfb                              True
37s
      └─MachineInfrastructure - GCPMachine/gcp-example-md-0-9h5vn
```

NOTE: After moving the cluster lifecycle services to the workload cluster, remember to use `dkp` with the workload cluster `kubeconfig`.

Use `dkp` with the bootstrap cluster to delete the workload cluster.

Persistent Volumes (PVs) are not deleted automatically by design in order to preserve your data. However, they take up storage space if not deleted. You must delete PVs manually.

6.7.7.2 Delete the workload cluster

1. Delete the Kubernetes cluster and wait a few minutes:
   Before deleting the cluster, DKP deletes all Services of type LoadBalancer on the cluster. To skip this step, use the flag `--delete-kubernetes-resources=false`.

```
dkp delete cluster --kubeconfig $HOME/.kube/config --cluster-name $
{CLUSTER_NAME}
```

```
✓ Deleting Services with type LoadBalancer for Cluster default/gcp-example
✓ Deleting ClusterResourceSets for Cluster default/gcp-example
✓ Deleting cluster resources
✓ Waiting for cluster to be fully deleted
Deleted default/gcp-example cluster
```

After the workload cluster is deleted, delete the bootstrap cluster.

6.7.7.3 Delete the bootstrap cluster

1. Delete the bootstrap cluster using `dkp delete`:

```
dkp delete bootstrap --kubeconfig $HOME/.kube/config
```

```
✓ Deleting bootstrap cluster
```

6.7.7.4 Known Limitations

Be aware of these limitations in the current release of DKP.

The DKP version used to create the workload cluster must match the DKP version used to delete the workload cluster.

6.8 Verify DKP installation

Check DKP components to verify the status of your cluster

In this section we show you how to verify a DKP installation.

6.8.1 Check the cluster infrastructure and nodes

DKP ships with some default diagnosis tools to check your cluster, such as the `describe` command. You can use those tools to validate your installation.

If you have not done so already, set the environment variable for your cluster name, substituting `dkp-example` with the name of your cluster:

```
export CLUSTER_NAME=dkp-example
```

Then, run this command to check the health of the cluster infrastructure:

```
dkp describe cluster --cluster-name=${CLUSTER_NAME}
```

A healthy cluster returns an output similar to:

```
NAME                                                              READY  SEVERITY
REASON  SINCE  MESSAGE
Cluster/dkp-example                                               True
121m
```

```
├──ClusterInfrastructure - AWSCluster/dkp-example                True
121m
├──ControlPlane - KubeadmControlPlane/dkp-example-control-plane  True
121m
│  ├──Machine/dkp-example-control-plane-h52t6                    True
121m
│  ├──Machine/dkp-example-control-plane-knrrh                    True
121m
│  └──Machine/dkp-example-control-plane-zmjjx                    True
121m
└──Workers
   └──MachineDeployment/dkp-example-md-0                         True
121m
     ├──Machine/dkp-example-md-0-88488cb74-2vxjq                 True
121m
     ├──Machine/dkp-example-md-0-88488cb74-84xsd                 True
121m
     ├──Machine/dkp-example-md-0-88488cb74-9xmc6                 True
121m
     └──Machine/dkp-example-md-0-88488cb74-mjf6s                 True
121m
```

Use this `kubectl` command to check to see if all cluster nodes are ready:

```
kubectl get nodes
```

The output should look similar to this, with all statuses set to `Ready`

```
NAME                                         STATUS   ROLES                  AGE
VERSION
ip-10-0-112-116.us-west-2.compute.internal   Ready    <none>                 135m
v1.21.6
ip-10-0-122-142.us-west-2.compute.internal   Ready    <none>                 135m
v1.21.6
ip-10-0-186-214.us-west-2.compute.internal   Ready    control-plane,master   133m
v1.21.6
ip-10-0-231-82.us-west-2.compute.internal    Ready    control-plane,master   135m
v1.21.6
ip-10-0-71-114.us-west-2.compute.internal    Ready    <none>                 135m
v1.21.6
ip-10-0-71-207.us-west-2.compute.internal    Ready    <none>                 135m
v1.21.6
ip-10-0-85-253.us-west-2.compute.internal    Ready    control-plane,master   137m
v1.21.6
```

To verify a successful installation, all of the previous commands should return as Ready or True.

6.8.2 Verify all pods are running

All pods installed by DKP should be in `Running` or `Completed` status if the install is successful.

```
kubectl get pods --all-namespaces
```

6.8.3 Troubleshooting

If any pod is not in `Running` or `Completed` status, you need to investigate further. If it appears that something has not deployed properly or fully, run a diagnostic bundle(see page 632):

```
dkp diagnose
```

This collects information from pods and infrastructure. For more information, see more about generating a support bundle and the different configurations that it supports(see page 632).

6.9 Konvoy Image Builder

Konvoy Image Builder (KIB) is a complete solution for building Cluster API compliant images.

This section describes how to use the KIB to create a Cluster API[273] compliant machine images. Machine images contain configuration information and software to create a specific, pre-configured, operating environment. For example, you can create an image of your current computer system settings and software. The machine image can then be replicated and distributed, creating your computer system for other users. The KIB uses variable overrides(see page 330) to specify base image and container images to use in your new machine image.

6.9.1 Prerequisites

Before you begin, you must ensure your versions of KIB and DKP are compatible:

- Download the supported version(see page 0) of the KIB[274] bundle from the KIB Version section of the chart below (prefixed with `konvoy-image-bundle`) for your Operating System.
- Create a working `Docker` setup.

6.9.2 Compatible versions

Along with the KIB Bundle, we publish a file containing checksums for the bundle files. The recommendation for using these checksums is to verify the integrity of the downloads.

273 https://cluster-api.sigs.k8s.io/
274 https://d2iq.atlassian.net/wiki/pages/resumedraft.action?draftId=89163373

DKP Version	KIB Version (Click for download)
v2.3.3	v1.19.14[275]
v2.3.2	v1.19.14[276]
v2.3.1	v1.19.11[277]
v2.3.0	v1.19.9[278]
v2.2.3	v1.17.4[279]
v2.2.2	v1.17.4[280]
v2.2.1	v1.13.2[281]
v2.2.0	v1.11.0[282]
v2.1.5	v1.5.2[283]
v2.1.4	v1.5.0[284]
v2.1.3	v1.5.0[285]
v2.1.2	v1.5.0[286]
v2.1.1	v1.5.0[287]
v2.1.0	v1.5.0[288]

275 https://github.com/mesosphere/konvoy-image-builder/releases/tag/v1.19.14
276 https://github.com/mesosphere/konvoy-image-builder/releases/tag/v1.19.14
277 https://github.com/mesosphere/konvoy-image-builder/releases/tag/v1.19.11
278 https://github.com/mesosphere/konvoy-image-builder/releases/tag/v1.19.9
279 https://github.com/mesosphere/konvoy-image-builder/releases/tag/v1.17.4
280 https://github.com/mesosphere/konvoy-image-builder/releases/tag/v1.17.4
281 https://github.com/mesosphere/konvoy-image-builder/releases/tag/v1.13.2
282 https://github.com/mesosphere/konvoy-image-builder/releases/tag/v1.11.0
283 https://github.com/mesosphere/konvoy-image-builder/releases/tag/v1.5.1
284 https://github.com/mesosphere/konvoy-image-builder/releases/tag/v1.5.1
285 https://github.com/mesosphere/konvoy-image-builder/releases/tag/v1.5.0
286 https://github.com/mesosphere/konvoy-image-builder/releases/tag/v1.5.0
287 https://github.com/mesosphere/konvoy-image-builder/releases/tag/v1.5.0
288 https://github.com/mesosphere/konvoy-image-builder/releases/tag/v1.5.0

KIB will run and print out the name of the created image for your infrastructure provider as shown on specific provider pages below. Use this name when creating a Kubernetes cluster.

- KIB with AWS(see page 324)
- KIB with GCP(see page 328)
- Use Override files with DKP(see page 330)

6.9.3 KIB with AWS

The following section describes how to use Konvoy Image Builder (KIB) with Amazon Web Services (AWS).

Section Contents

- Create a custom AMI(see page 324)
- AWS Air-gapped AMI(see page 326)

6.9.3.1 Create a custom AMI

Learn how to build a custom AMI for use with DKP

This procedure describes how to use the Konvoy Image Builder(see page 322) (KIB) to create a Cluster API[289] compliant Amazon Machine Image (AMI). AMI images contain configuration information and software to create a specific, pre-configured, operating environment. For example, you can create an AMI image of your current computer system settings and software. The AMI image can then be replicated and distributed, creating your computer system for other users. The KIB uses variable overrides(see page 330) to specify base image and container images to use in your new AMI.

Prerequisites

Before you begin, you must:

- Download the supported version(see page 811) of the KIB(see page 0) bundle (prefixed with `konvoy-image-bundle`) for your OS. Do not use the release prefixed with `konvoy-image-builder`.
- Create a working `Docker` setup.

Extract AMI Bundle

Extract the bundle and `cd` into the extracted `konvoy-image-bundle-$VERSION_$OS` folder. The bundled version of `konvoy-image` contains an embedded `docker` image that contains all the requirements for building.

289 https://cluster-api.sigs.k8s.io/

The `konvoy-image` binary and all supporting folders are also extracted. When run, `konvoy-image` bind mounts the current working directory (`${PWD}`) into the container to be used.

- Set environment variables for AWS access[290]. The following variables must be set using your credentials:

```
export AWS_ACCESS_KEY_ID
export AWS_SECRET_ACCESS_KEY
export AWS_DEFAULT_REGION
```

- Ensure you have an override file(see page 330) to configure specific attributes of your AMI file.

Build the image

Depending on which version of DKP you are running, steps and flags will be different.

Do the following:

Run the `konvoy-image` command to build and validate the image.

```
konvoy-image build images/ami/centos-7.yaml
```

By default it builds in the `us-west-2` region. to specify another region set the `--region` flag:

```
konvoy-image build --region us-east-1 images/ami/centos-7.yaml
```

When the command is complete the `ami` id is printed and written to `./manifest.json` .

Launch a DKP cluster with a custom AMI

To use the built `ami` with DKP, specify it with the `--ami` flag when calling cluster create.

```
dkp create cluster aws --cluster-name=$(whoami)-aws-cluster --ami ami-0123456789
```

Launch a DKP cluster with custom AMI lookup

By default `konvoy-image` will name the AMI in such a way that `dkp` can discover the latest AMI for a base OS and Kubernetes version. To create a cluster that will use the latest AMI, specify the `--ami-format` , `--ami-base-os` and `--ami-owner` flags:

290 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-envvars.html

```
dkp create cluster aws --cluster-name=$(whoami)-aws-cluster --ami-format "konvoy-ami-{{.BaseOS}}-?{{.K8sVersion}}-*" --ami-base-os centos-7 --ami-owner 123456789012
```

Related information

For information on related topics or procedures, refer to the following:

- Creating GPU enabled on-premises configurations(see page 165)

6.9.3.2 AWS Air-gapped AMI

Create an AMI using KIB for use in an air-gapped cluster

Using KIB(see page 322), you can build an AMI without requiring access to the internet by providing an additional `--override` flag. Depending on which version of DKP you are running, steps and flags will be different.

1. Create the directories where you will place the air-gapped bundles:

   ```
   mkdir artifacts
   mkdir artifacts/images
   ```

2. Define an environment variable for the Kubernetes version that corresponds with DKP release you are installing. You can find the correct Kubernetes version by checking the release notes for the release you are installing.

   ```
   export VERSION=1.23.12
   ```

3. Set an environment variable for the AMI's OS you are will be using. The OS packages bundles will contain the RPMs for `containerd`, Kubernetes and all of their dependencies required to install these packages without access to any external RPM repositories. The available options are:
   - `centos_7_x86_64`
   - `centos_7_x86_64_fips`
   - `redhat_7_x86_64`
   - `redhat_7_x86_64_fips`
   - `redhat_8_x86_64`
   - `redhat_8_x86_64_fips`

   ```
   export BUNDLE_OS=centos_7_x86_64
   ```

4. Download the OS packages bundle:

   ```
   curl --output artifacts/"$VERSION"_"$BUNDLE_OS".tar.gz -O https://downloads.d2iq.com/dkp/airgapped/os-packages/"$VERSION"_"$BUNDLE_OS".tar.gz
   ```

5. Download the Kubernetes images bundle. This bundle includes the necessary images for `kubeadm` to bootstrap a Kubernetes `Node`.
   The available options for each Kubernetes version are:
   - `<version>_images.tar.gz`
   - `<version>_images_fips.tar.gz`

   ```
   curl --output artifacts/images/"$VERSION"_images.tar.gz -O https://
   downloads.d2iq.com/dkp/airgapped/kubernetes-images/"$VERSION"_images.tar.gz
   ```

6. Download the PIP packages. This bundle includes a few packages required by DKP to bootstrap machines.

   ```
   curl --output artifacts/pip-packages.tar.gz -O https://downloads.d2iq.com/dkp/
   airgapped/pip-packages/pip-packages.tar.gz
   ```

7. Set a variable that indicates which OS-specific `containerd` bundle to download. The available options are:
   - `centos_7.9_x86_64`
   - `rhel-7.9-x86_64`
   - `rhel-8.2-x86_64`
   - `rhel-8.4-x86_64`
   - `ubuntu-20.04-x86_64`
   - `ubuntu-18.04-x86_64`
   - `ol-7.9-x86_64`
   - `sles-15.3-x86_64`

   ```
   export CONTAINERD_OS=centos-7.9-x86_64
   ```

8. Download the `containerd` bundle:

   ```
   curl --output artifacts/containerd-1.4.13-d2iq.1-"$CONTAINERD_OS".tar.gz --
   location https://packages.d2iq.com/dkp/containerd/containerd-1.4.13-
   d2iq.1-"$CONTAINERD_OS".tar.gz
   ```

9. Follow the instructions to build an AMI(see page 324) in the setting an additional `--overrides overrides/offline.yaml` flag.

Then, you can seed your docker registry(see page 156).

https://github.com/minio/minio/tree/RELEASE.2022-01-08T03-11-54Z

https://github.com/minio/minio/tree/RELEASE.2021-02-14T04-01-33Z

6.9.4 KIB with GCP

This procedure describes how to use the Konvoy Image Builder(see page 322) (KIB) to create a Cluster API[291] compliant GCP image. GCP images contain configuration information and software to create a specific, pre-configured, operating environment. For example, you can create a GCP image of your current computer system settings and software. The GCP image can then be replicated and distributed, creating your computer system for other users. The KIB uses variable overrides(see page 330) to specify base image and container images to use in your new GCP image.

Google Cloud Platform does not publish images. You must first build the image.

6.9.4.1 Prerequisites

Before you begin, you must:

- Download the supported version(see page 811) of the KIB(see page 0) bundle (prefixed with `konvoy-image-bundle`) for your OS. Do not use the release prefixed with `konvoy-image-builder`.
- Create a working `Docker` setup.

6.9.4.2 GCP Prerequisites

- If you are creating your image on either a non-GCP instance or one that does not have the required roles:
  - (option 1) Create a service account using the following `gcloud` commands:

```
export GCP_PROJECT=<your GCP project ID>
export SERVICE_ACCOUNT_USER=<some new service account user>
export GOOGLE_APPLICATION_CREDENTIALS="$HOME/.gcloud/credentials.json"

gcloud iam service-accounts create "${SERVICE_ACCOUNT_USER}" --project=$
{GCP_PROJECT}
gcloud projects add-iam-policy-binding ${GCP_PROJECT} --
member="serviceAccount:${SERVICE_ACCOUNT_USER}@$
{GCP_PROJECT}.iam.gserviceaccount.com" --role=roles/
compute.instanceAdmin.v1
gcloud projects add-iam-policy-binding ${GCP_PROJECT} --
member="serviceAccount:${SERVICE_ACCOUNT_USER}@$
{GCP_PROJECT}.iam.gserviceaccount.com" --role=roles/iam.serviceAccountUser
```

291 https://cluster-api.sigs.k8s.io/

```
gcloud iam service-accounts keys create ${GOOGLE_APPLICATION_CREDENTIALS}
--iam-account="${SERVICE_ACCOUNT_USER}@$
{GCP_PROJECT}.iam.gserviceaccount.com"
```

- (option 2) If you have already created a service account, retrieve the credentials for an existing service account using the following `gcloud` commands:

```
export GCP_PROJECT=<your GCP project ID>
export SERVICE_ACCOUNT_USER=<existing service account user>
export GOOGLE_APPLICATION_CREDENTIALS="$HOME/.gcloud/credentials.json"

gcloud projects add-iam-policy-binding ${GCP_PROJECT} --
member="serviceAccount:${SERVICE_ACCOUNT_USER}@$
{GCP_PROJECT}.iam.gserviceaccount.com" --role=roles/
compute.instanceAdmin.v1
gcloud projects add-iam-policy-binding ${GCP_PROJECT} --
member="serviceAccount:${SERVICE_ACCOUNT_USER}@$
{GCP_PROJECT}.iam.gserviceaccount.com" --role=roles/iam.serviceAccountUser
gcloud iam service-accounts keys create ${GOOGLE_APPLICATION_CREDENTIALS}
--iam-account="${SERVICE_ACCOUNT_USER}@$
{GCP_PROJECT}.iam.gserviceaccount.com"
```

6.9.4.3 Create a Network (optional)

Building an image requires a Network[292] with firewall rules that allow SSH access to the VM instance.

1. Set your GCP Project ID for your `gcp` account unless already set previously:

```
export GCP_PROJECT=<your GCP project ID>
```

2. Run the following to create a new network:

```
export NETWORK_NAME=kib-ssh-network
gcloud compute networks create "${NETWORK_NAME}" --project="${GCP_PROJECT}" --
subnet-mode=auto --mtu=1460 --bgp-routing-mode=regional
```

3. Create the firewall rule to allow Ingress access on port 22:

```
gcloud compute firewall-rules create "${NETWORK_NAME}-allow-ssh" --project="$
{GCP_PROJECT}" --network="projects/${GCP_PROJECT}/global/networks/$
{NETWORK_NAME}" --description="Allows TCP connections from any source to any
instance on the network using port 22." --direction=INGRESS --priority=65534 --
source-ranges=0.0.0.0/0 --action=ALLOW --rules=tcp:22
```

292 https://cloud.google.com/vpc/docs/vpc

Build the GCP Image

Follow these steps:

1. Run the `konvoy-image` command to build and validate the image:

```
./konvoy-image build gcp --project-id ${GCP_PROJECT} --network ${NETWORK_NAME}
images/gcp/ubuntu-2004.yaml
```

2. KIB will run and print out the name of the created image, you will use this name when creating a Kubernetes cluster. See sample output below:

```
...
==> ubuntu-2004-focal-v20220419: Deleting instance...
    ubuntu-2004-focal-v20220419: Instance has been deleted!
==> ubuntu-2004-focal-v20220419: Creating image...
==> ubuntu-2004-focal-v20220419: Deleting disk...
    ubuntu-2004-focal-v20220419: Disk has been deleted!
==> ubuntu-2004-focal-v20220419: Running post-processor: manifest
Build 'ubuntu-2004-focal-v20220419' finished after 7 minutes 46 seconds.

==> Wait completed after 7 minutes 46 seconds

==> Builds finished. The artifacts of successful builds are:
--> ubuntu-2004-focal-v20220419: A disk image was created: konvoy-ubuntu-2004-1-2
3-7-1658523168
--> ubuntu-2004-focal-v20220419: A disk image was created: konvoy-ubuntu-2004-1-2
3-7-1658523168
```

3. To find a list of images you have created in your account, run the following command:

```
gcloud compute images list --no-standard-images
```

With your KIB image now created, you can now move onto Bootstrap GCP(see page 301) and set up your Cluster API[293] (CAPI) controllers, or run GCP Quick Start(see page 46) to create a cluster with little customization.

6.9.5 Use Override files with DKP

6.9.5.1 Learn how to use override files with DKP

Konvoy Image Builder(see page 322) uses YAML `override` files to configure specific attributes. These files provide information to override default values for certain parts of your image file. `override` files can modify the version and parameters of the image description and the Ansible playbook.

There are 2 types of override files:

293 https://cluster-api.sigs.k8s.io/

- Default override(see page 331) files provided by DKP for you to use for specific purposes.
- Custom override files(see page 332) you create to use with DKP.

6.9.5.2 Default Override Files

Learn how to use override files with DKP

DKP comes with default override files:

- FIPS override file(see page 331)
- Nvidia GPU override file(see page 332)
- Offline override file(see page 332)
- Offline for FIPS override file(see page 331)
- Oracle Redhat Linux Kernel Override File(see page 332)

You can find these override files in the Konvoy Image Builder repo[294].

FIPS Override Files

You can find these override files in the Konvoy Image Builder repo[295].

```
---
k8s_image_registry: docker.io/mesosphere

fips:
  enabled: true

build_name_extra: -fips
kubernetes_build_metadata: fips.0
default_image_repo: hub.docker.io/mesosphere
kubernetes_rpm_repository_url: "https://packages.d2iq.com/konvoy/stable/linux/repos/
el/kubernetes-v{{ kubernetes_version }}-fips/x86_64"
docker_rpm_repository_url: "\
  https://containerd-fips.s3.us-east-2.amazonaws.com\
  /{{ ansible_distribution_major_version|int }}\
  /x86_64"
```

Offline FIPS Override File

```
# fips os-packages
os_packages_local_bundle_file: "{{ playbook_dir }}/../artifacts/
{{ kubernetes_version }}_{{ ansible_distribution|lower }}
_{{ ansible_distribution_major_version }}_x86_64_fips.tar.gz"
containerd_local_bundle_file: "{{ playbook_dir }}/../artifacts/
{{ containerd_tar_file }}"
pip_packages_local_bundle_file: "{{ playbook_dir }}/../artifacts/pip-packages.tar.gz"
```

294 https://github.com/mesosphere/konvoy-image-builder/tree/main/overrides
295 https://github.com/mesosphere/konvoy-image-builder/tree/main/overrides

```
images_local_bundle_dir: "{{ playbook_dir}}/../artifacts/images"
```

Nvidia GPU Override File

You can find these override files in the Konvoy Image Builder repo[296].

```
---
gpu:
  types:
    - nvidia

build_name_extra: "-nvidia"
```

Offline Override File

You can find these override files in the Konvoy Image Builder repo[297].

```
os_packages_local_bundle_file: "{{ playbook_dir }}/../artifacts/
{{ kubernetes_version }}_{{ ansible_distribution|lower }}
_{{ ansible_distribution_major_version }}_x86_64.tar.gz"
pip_packages_local_bundle_file: "{{ playbook_dir }}/../artifacts/pip-packages.tar.gz"
images_local_bundle_dir: "{{ playbook_dir}}/../artifacts/images"
```

Oracle Redhat Linux Kernel Override File

You can find these override files in the Konvoy Image Builder repo[298].

```
---
oracle_kernel: RHCK
```

6.9.5.3 Custom Override Files

You can specify customization of your KIB images through the use of override files, which are used to specify alternate package libraries, Docker image repos, and other customizations. Some example custom override files are described in the following topics:

Base Image Override Files

When using KIB to create an OS image that is compliant with DKP, the instructions on how to build and configure the image are included in the file located in `images/<builder-type>/<os-version.yaml`, as seen below:

```
./konvoy-image build images/<builder-type>/{os}-{version}.yaml
```

296 https://github.com/mesosphere/konvoy-image-builder/tree/main/overrides
297 https://github.com/mesosphere/konvoy-image-builder/tree/main/overrides
298 https://github.com/mesosphere/konvoy-image-builder/tree/main/overrides

In previous versions of KIB, only Azure and AWS providers were available. Therefore, specifying a vSphere was not necessary.

Although there are several parameters specified by default in the Packer templates for each provider, it is possible to override the default values.

One option is to execute KIB with specific flags to override the values of the source AMI (`--source-ami`), AMI region (`--ami-regions`), AWS EC2 instance type (`--aws-instance-type`), and so on. For a comprehensive list of these flags, please run:

```
./konvoy-image build --help
```

Another option is by creating a file with the parameters to be overridden and specify the `--overrides` flag as shown below:

```
./konvoy-image build images/<builder-type>/<os>-<version>.yaml --overrides
overrides.yaml
```

While CLI flags can be used in combination with override files, CLI flags take priority over any override files.

Example 1:

For example, when using the AWS Packer builder to override the above base image with another base image, create an override file and set the `source_ami` under the packer key. This overrides the image search and forces the use of the specified `source_ami` .

```
---
packer:
  source_ami: "ami-0123456789"
```

After creating the override file for our `source_ami` , we can pass our override file by using the `--overrides` flag when building our image. Replace infrastructure provider in the command below before running it (`aws` , `azure` , `vsphere`):

```
./konvoy-image build aws images/ami/centos-7.yaml --overrides override-source-
ami.yaml
```

Example 2:

To abide to security practices, a user could set their own username and password while creating the base OS image and override the default credentials in KIB. To do this, create a file with the following content:

```
---
packer:
  ssh_username: "<USERNAME>"
  ssh_password: "<PASSWORD>"
```

For a complete list of the variables that can be modified for each Packer builder, users can refer to:

- AWS Packer template[299]
- Azure Packer template[300]
- GCP Packer template[301]
- vSphere Packer template[302]

An AWS example would be the current base image description at `images/ami/centos-7.yaml` which is similar to the following:

```
---
# Example images/ami/centos-7.yaml
download_images: true
packer:
  ami_filter_name: "CentOS 7.9.2009 x86_64"
  ami_filter_owners: "125523088429"
  distribution: "CentOS"
  distribution_version: "7"
  source_ami: ""
  ssh_username: "centos"
  root_device_name: "/dev/sda1"
build_name: "centos-7"
packer_builder_type: "amazon"
python_path: ""
```

or

An Azure example would be the current base image description at `images/azure/centos-79.yaml` which is similar to the following:

```
---
# Example images/azure/centos-79.yaml
download_images: true
packer:
  distribution: "centos"  #offer
  distribution_version: "7_9-gen2"  #SKU
```

299 https://github.com/mesosphere/konvoy-image-builder/blob/v1.19.12/pkg/packer/manifests/aws/packer.json.tmpl#L2
300 https://github.com/mesosphere/konvoy-image-builder/blob/v1.19.12/pkg/packer/manifests/azure/packer.json.tmpl#L2
301 https://github.com/mesosphere/konvoy-image-builder/blob/v1.19.12/pkg/packer/manifests/gcp/packer.json.tmpl#L2
302 https://github.com/mesosphere/konvoy-image-builder/blob/v1.19.12/pkg/packer/manifests/vsphere/packer.json.tmpl#L2

```
  image_publisher: "openlogic"
  image_version: "latest"
  ssh_username: "centos"

build_name: "centos-7"
packer_builder_type: "azure"
python_path: ""
```

or

A vSphere example would be the current base image description at `images/vsphere/rhel-79.yaml` which is similar to the following:

```
---
download_images: true
build_name: "rhel-79"
packer_builder_type: "vsphere"
guestinfo_datasource_slug: "https://raw.githubusercontent.com/vmware/cloud-init-
vmware-guestinfo"
guestinfo_datasource_ref: "v1.4.0"
guestinfo_datasource_script: "{{guestinfo_datasource_slug}}/
{{guestinfo_datasource_ref}}/install.sh"
packer:
cluster: "zone1"
  datacenter: "dc1"
  datastore: "esxi-06-disk1"
  folder: "cluster-api"
  insecure_connection: "false"
  network: "Airgapped"
  resource_pool: "Users"
  template: "base-rhel-7"
  vsphere_guest_os_type: "rhel7_64Guest"
  guest_os_type: "rhel7-64"
  #goss params
  distribution: "RHEL"
  distribution_version: "7.9"
```

See Supported Operating Systems(see page 78) for details.

Container Image Override Files

Your machine image uses a container image. The Ansible playbooks pull a minimal set of container images[303] for use. You can add or delete additional images by specifying an `override` file for the `extra_images` variable. Konvoy requires several additional images be present. Create a new override file and specify the following `extra_images`:

```
# Example override-images.yaml
---
```

303 https://github.com/mesosphere/konvoy-image-builder/tree/main/ansible/roles/images

```
extra_images:
  - docker.io/mesosphere/cluster-api-controller:v1.1.3-d2iq.5
  - docker.io/mesosphere/cluster-api-preprovisioned-controller:v0.9.2
  - docker.io/mesosphere/kubeadm-bootstrap-controller:v1.1.3-d2iq.5
  - docker.io/mesosphere/kubeadm-control-plane-controller:v1.1.3-d2iq.5
  - gcr.io/cluster-api-provider-vsphere/release/manager:v1.2.0
  - gcr.io/k8s-staging-cluster-api/capd-manager:v1.1.3
  - k8s.gcr.io/cluster-api-aws/cluster-api-aws-controller:v1.4.1
  - mcr.microsoft.com/oss/azure/aad-pod-identity/nmi:v1.8.8
  - quay.io/jetstack/cert-manager-cainjector:v1.5.3
  - quay.io/jetstack/cert-manager-controller:v1.5.3
  - quay.io/jetstack/cert-manager-webhook:v1.5.3
  - us.gcr.io/k8s-artifacts-prod/cluster-api-azure/cluster-api-azure-
controller:v1.3.2
  - us.gcr.io/k8s-artifacts-prod/cluster-api-gcp/cluster-api-gcp-controller:v1.1.0
  - docker.io/mesosphere/konvoy-image-builder:v1.19.7
  - docker.io/plndr/kube-vip:v0.3.7
  - ghcr.io/kube-vip/kube-vip:v0.3.9
  - gcr.io/k8s-staging-sig-storage/snapshot-controller:v5.0.1
  - public.ecr.aws/ebs-csi-driver/aws-ebs-csi-driver:v1.8.0
  - public.ecr.aws/eks-distro/kubernetes-csi/external-attacher:v3.1.0-eks-1-18-13
  - public.ecr.aws/eks-distro/kubernetes-csi/external-provisioner:v2.1.1-eks-1-18-13
  - public.ecr.aws/eks-distro/kubernetes-csi/external-resizer:v1.1.0-eks-1-18-13
  - public.ecr.aws/eks-distro/kubernetes-csi/external-snapshotter/csi-snapshotter:v5.0
.1-eks-1-22-7
  - public.ecr.aws/eks-distro/kubernetes-csi/livenessprobe:v2.2.0-eks-1-18-13
  - public.ecr.aws/eks-distro/kubernetes-csi/node-driver-registrar:v2.1.0-eks-1-18-13
  - mcr.microsoft.com/oss/kubernetes-csi/azuredisk-csi:v1.19.0
  - mcr.microsoft.com/oss/kubernetes-csi/csi-attacher:v3.4.0
  - mcr.microsoft.com/oss/kubernetes-csi/csi-node-driver-registrar:v2.5.0
  - mcr.microsoft.com/oss/kubernetes-csi/csi-provisioner:v3.1.0
  - mcr.microsoft.com/oss/kubernetes-csi/csi-resizer:v1.4.0
  - mcr.microsoft.com/oss/kubernetes-csi/csi-snapshotter:v5.0.1
  - mcr.microsoft.com/oss/kubernetes-csi/livenessprobe:v2.6.0
  - quay.io/external_storage/local-volume-provisioner:v2.4.0
  - gcr.io/cloud-provider-vsphere/csi/release/driver:v2.5.2
  - gcr.io/cloud-provider-vsphere/csi/release/syncer:v2.5.2
  - k8s.gcr.io/sig-storage/csi-attacher:v3.4.0
  - k8s.gcr.io/sig-storage/csi-node-driver-registrar:v2.5.0
  - k8s.gcr.io/sig-storage/csi-provisioner:v3.1.0
  - k8s.gcr.io/sig-storage/csi-resizer:v1.4.0
  - k8s.gcr.io/sig-storage/csi-snapshotter:v5.0.1
  - k8s.gcr.io/sig-storage/livenessprobe:v2.6.0
  - k8s.gcr.io/cloud-provider-gcp/gcp-compute-persistent-disk-csi-driver:v1.7.2
  - k8s.gcr.io/sig-storage/csi-attacher:v3.4.0
  - k8s.gcr.io/sig-storage/csi-node-driver-registrar:v2.5.0
  - k8s.gcr.io/sig-storage/csi-provisioner:v3.1.0
  - k8s.gcr.io/sig-storage/csi-resizer:v1.4.0
  - k8s.gcr.io/sig-storage/csi-snapshotter:v4.0.1
  - gcr.io/cloud-provider-vsphere/cpi/release/manager:v1.23.1
  - docker.io/calico/cni:v3.23.2
  - docker.io/calico/kube-controllers:v3.23.2
  - docker.io/calico/node:v3.23.2
```

```
  - docker.io/calico/pod2daemon-flexvol:v3.23.2
  - docker.io/calico/typha:v3.23.2
  - quay.io/tigera/operator:v1.27.7
  - docker.io/bitnami/kubectl:1.23.12
  - us.gcr.io/k8s-artifacts-prod/autoscaling/cluster-autoscaler:v1.23.1
  - k8s.gcr.io/nfd/node-feature-discovery:v0.11.1-minimal
  - nvcr.io/nvidia/gpu-feature-discovery:v0.6.1
  - docker.io/library/busybox:1
  - docker.io/mesosphere/pause-busybox:3.2
  - docker.io/mesosphere/dkp-diagnostics-node-collector:v0.4.3
  - docker.io/mesosphere/kfips:v0.2.0
  - k8s.gcr.io/sig-storage/snapshot-validation-webhook:v3.0.2
  - quay.io/metallb/controller:v0.12.1
  - quay.io/metallb/speaker:v0.12.1
  - k8s.gcr.io/kube-proxy:v1.23.12
  - docker.io/mesosphere/kube-proxy:v1.23.12_fips.0
  - k8s.gcr.io/coredns/coredns:v1.8.4
```

HTTP Proxy Override Files

An HTTP proxy configuration can be used when creating your AMI image. The Ansible playbooks will create `systemd` drop-in files for `containerd` and `kubelet` to configure the `http_proxy`, `http_proxy`, and `no_proxy` environment variables for the service from the file `/etc/konvoy_http_proxy.conf`. To configure a proxy for use during image creation, create a new override file and specify the following:

```
# Example override-proxy.yaml
---
export http_proxy: http://example.org:8080
export https_proxy: http://example.org:8081
export no_proxy: example.org,example.com,example.net
```

These values are only used for the image creation. After the image is created, the Ansible playbooks remove the `/etc/konvoy_http_proxy.conf` file. The `dkp` command can be used to configure the `KubeadmConfigTemplate` object to create this file on bootup of the image with values supplied during the `dkp` invocation. This enables using different proxy settings for image creation and runtime.

Pre-Provisioned Override Files

Pre-provisioned environments require certain override files to work properly. Then, those override files must also have a secret secret that includes all of the overrides you wish to provide in one file. For example, if you wish to provide an override with Docker credentials and a different source for EPEL on a CentOS7 machine, you can create a file like this:

```
image_registries_with_auth:
- host: "registry-1.docker.io"
  username: "my-user"
  password: "my-password"
  auth: ""
```

```
  identityToken: ""

epel_centos_7_rpm: https://my-rpm-repostory.org/epel/epel-release-latest-7.noarch.rpm
```

6.10 FIPS 140-2 Compliance

Understand FIPS-140 Operating Mode and Requirements

Developed by a working group of government, industry operators, and vendors, the Federal Information Processing Standard (FIPS), FIPS-140 defines security requirements for cryptographic modules. The standard provides for a wide spectrum of data sensitivity, transaction values, and a diversity of application environment security situations. The standard specifies four security levels for each of eleven requirement areas. Each successive level offers increased security.

NIST introduced FIPS 140-2 validation, by accredited third party laboratories, as a formal, rigorous process to protect sensitive digitally-stored information not under Federal security classifications.

6.10.1 FIPS support in DKP

DKP supports provisioning a FIPS-enabled Kubernetes control plane. Core Kubernetes components are compiled using a version of Go, called goboring, which uses a FIPS-certified cryptographic module[304] for all cryptographic functions.

Before provisioning DKP, you will need to follow your OS vendor's instructions to ensure that your OS, or OS images, are prepared for operating in FIPS mode[305].

> You cannot apply FIPS-mode to an existing cluster, you must create a new cluster with FIPS enabled. Similarly, a FIPS-mode cluster must remain a FIPS-mode cluster; you cannot change the cluster's FIPS status after you create it.

6.10.2 Infrastructure requirements for FIPS-140-2 mode

To ensure proper operations in FIPS mode, be sure that your environment meets these requirements:

6.10.2.1 Supported operating systems

Supported Operating Systems for FIPS mode are Red Hat Enterprise Linux and CentOS. See the Supported Operating Systems(see page 78) for details on the tested and supported versions.

304 https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp3702.pdf

305 https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/security_guide/chap-federal_standards_and_regulations

6.10.3 Deploying a Cluster in FIPS mode

In order to create a cluster in FIPS mode, we must inform the bootstrap controllers of the appropriate image repository and version tags of the official D2iQ FIPS builds of kubernetes.

6.10.3.1 Supported FIPS builds

Component	Repository	Version
Kubernetes	docker.io/mesosphere[306]	v1.23.12+fips.0
etcd	docker.io/mesosphere[307]	v3.4.13+fips.0

When creating a cluster, use the following command line options:

- `--ami <fips enabled AMI created in the previous step>` (AWS only)
- `--kubernetes-version <version>+fips.<build>`
- `--etcd-version <version>+fips.<build>`
- `--kubernetes-image-repository docker.io/mesosphere`
- `--etcd-image-repository docker.io/mesosphere`

For example:

```
dkp create cluster aws --cluster-name myFipsCluster \
--ami=ami-03dcaa75d45aca36f \
--kubernetes-version=v1.23.12+fips.0 \
--kubernetes-image-repository=docker.io/mesosphere \
--etcd-image-repository=docker.io/mesosphere \
--etcd-version=3.4.13+fips.0
```

6.10.4 Create FIPS 140 Images

6.10.4.1 Use Konvoy Image Builder to create images with FIPS-compliant binaries

6.10.4.2 Create FIPS-140 images

KIB can produce images containing FIPS-140 compliant binaries. Use the `fips.yaml` override file(see page 330) provided with the image bundles.

306 https://hub.docker.com/layers/mesosphere/kube-apiserver/v1.23.12_fips.0/images/sha256-ca783b379499d926bee634942743e3cd50a64e275a5c89bc6bd9d83c43633697?context=explore

307 https://hub.docker.com/layers/kube-apiserver/mesosphere/kube-apiserver/v1.23.8_fips.0/images/sha256-f23f7c57020497582c26c399a993f37759fc14a1561568c6ff8a07033608ada2?context=explore

For example, this command produces a FIPS-compliant image on CentOS 8:

```
konvoy-image build --overrides overrides/fips.yaml images/ami/centos-8.yaml
```

Pre-provisioned infrastructure

If you are targeting a pre-provisioned infrastructure(see page 222), you can create a FIPS-compliant cluster by doing the following:

1. Create a bootstrap cluster(see page 229)
2. Create a secret on the bootstrap cluster with the contents from `fips.yaml` override file[308] and any other user overrides you wish to provide

```
kubectl create secret generic $CLUSTER_NAME-fips-overrides --from-
file=overrides.yaml=overrides.yaml
kubectl label secret $CLUSTER_NAME-fips-overrides clusterctl.cluster.x-k8s.io/move=
```

6.10.5 Validate FIPS in Cluster

You can use the FIPS validation tool to verify that specific components and services are FIPS-compliant by checking the signatures of the files against a signed signature file, and by checking that services are using the certified algorithms.

6.10.5.1 Download Signature Files

You need to download an appropriate, signed signature file before you run validation. Verify which version of DKP you are running to ensure you are downloading the manifest that is compliant with the DKP release number on your system. Use the links in the tables that follows to obtain a valid file:

DKP version 2.3

Operating System version	Kubernetes version	containerd version	Manifest URL
CentOS 7.9	v1.23.12	1.14.13	v1.23.12 CentOS 7.9 Manifest[309]
Oracle 7.9	v1.23.12	1.14.13	v1.23.12 OL 7.9 Manifest[310]
RHEL 7.9	v1.23.12	1.14.13	v1.23.12 EL 7.9 Manifest[311]

308 https://github.com/mesosphere/konvoy-image-builder/blob/main/overrides/fips.yaml
309 https://downloads.d2iq.com/dkp/fips/v2.3.3/manifest-centos-79.json.asc
310 https://downloads.d2iq.com/dkp/fips/v2.3.3/manifest-oracle-79.json.asc
311 https://downloads.d2iq.com/dkp/fips/v2.3.3/manifest-rhel-79.json.asc

Operating System version	Kubernetes version	containerd version	Manifest URL
RHEL 8.2	v1.23.12	1.14.13	v1.23.12 EL 8.2 Manifest[312]
RHEL 8.4	v1.23.12	1.14.13	v1.23.12 EL 8.4 Manifest[313]

6.10.5.2 Run FIPS validation

To verify the cluster is FIPS compliant, run the `dkp check cluster fips`. This command reads from the local manifest that was downloaded from the tables above in order to validate that specific components and services are FIPS-compliant. Run the command:

```
dkp check cluster fips --signature-file=manifest.asc --signature-configmap=signatures
--output-configmap=output
```

The full command usage and flags include:

```
dkp check cluster fips [flags]
```

Flags:

```
-h, --help                     help for fips
  -n, --namespace string         If present, the namespace scope for this CLI
request. (default "default")
  --output-configmap string      ConfigMap with fips signature data to verify.
[required]
  --signature-configmap string   ConfigMap with fips signature data to verify.
[required]
  --signature-file string        File containing fips signature data.
```

Validation command example

Upon successful completion, the command's output displays details about the deployment in JSON format. If validation fails, the command returns a non-zero status.

For example, to validate FIPS-mode operation with the signature file, `manifest-rhel8.json.asc`, you would run the following command:

312 https://downloads.d2iq.com/dkp/fips/v2.3.3/manifest-rhel-82.json.asc
313 https://downloads.d2iq.com/dkp/fips/v2.3.3/manifest-rhel-84.json.asc

```
dkp check cluster fips \
 --signature-file manifest-rhel8.json.asc \
 --signature-configmap prod-rhel8-fips-signatures \
 --output-configmap prod-rhel8-fips-validation
```

6.10.5.3 Run FIPS validation with existing ConfigMap

If you already have a signature ConfigMap, you can omit the `signature-file` flag, as in the following sample command:

```
dkp check cluster fips \
 --signature-configmap prod-rhel8-fips-signatures \
 --output-configmap prod-rhel8-fips-validation
```

In this case, the validation tool checks the cluster using the existing signature data and returns deployment details in JSON format.

6.10.6 FIPS 140 Mode Performance Impact

6.10.6.1 Understand the performance impact from operating your cluster in FIPS 140 mode

The Go language cryptographic module, Goboring, relies on CGO's foreign function interface to call C-language functions exposed by the cryptographic module. Each call into the C library starts with a base overhead of 200ns.

One benchmark[314] finds that the time to encrypt a single AES-128 block increased from 13ns to 209ns over the internal golang implementation. The preferred mode of D2iQ's FIPS module is `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`.

The aggregate impact on a stable control plane seems to be an increase of around 10% CPU utilization over default operation. Workloads that do not directly interact with the control plane are not affected.

6.11 Delete a DKP Cluster with One Command

You can use a single command line entry to delete a Kubernetes cluster on any of the platforms supported by DKP. Deleting a cluster means removing the cluster, all of its nodes, and all of the platform applications that were deployed on it as part of its creation. Use this command with extreme care, as it is not reversible!

You need to delete the attachment for any clusters attached in Kommander before running the `delete` command.

314 https://github.com/golang/go/issues/21525

Set the environment variable to be used throughout this documentation:

```
export CLUSTER_NAME=cluster-example
```

The basic DKP `delete` command structure is:

```
dkp delete cluster --cluster-name=${CLUSTER_NAME} --self-managed --kubeconfig=$
{CLUSTER_NAME}.conf
```

When you use the `--self-managed` flag, the prerequisite components and resources are moved from the self-managed cluster before deleting. When you omit this flag (the default value is false) the resources are assumed to be installed in a management cluster. The default value is false, or no flag.

This command performs the following actions:

- Creates a local bootstrap cluster
- Moves controllers to it
- Deletes the management cluster
- Deletes the local bootstrap cluster

6.12 Configuring the Control Plane

6.12.1 Prerequisites

Before you begin, make sure you have created a bootstrap cluster from the respective Infrastructure Providers(see page 103) section.

Users can make modifications to the `KubeadmControlplane` cluster-api object to configure different kubelet options. Please see the following guide if you wish to configure your control plane beyond the existing options that are available from flags.

6.12.1.1 Modifying Audit Logs

In order to modify control plane option, get the appropriate `cluster-api` objects that describe the cluster by running the following command:

The following example uses AWS, but can be used for `gcp`, `azure`, `preprovisioned`, and `vsphere` clusters.

```
dkp create cluster aws -c {MY_CLUSTER_NAME} -o yaml --dry-run >>
{MY_CLUSTER_NAME}.yaml
```

When you open `{MY_CLUSTER_NAME}.yaml` with your favorite text editor, look for the

`KubeadmControlPlane` object for your cluster. Below is an example object:

```
apiVersion: controlplane.cluster.x-k8s.io/v1beta1
kind: KubeadmControlPlane
metadata:
  name: my-cluster-control-plane
  namespace: default
spec:
  kubeadmConfigSpec:
    clusterConfiguration:
      apiServer:
        extraArgs:
          audit-log-maxage: "30"
          audit-log-maxbackup: "10"
          audit-log-maxsize: "100"
          audit-log-path: /var/log/audit/kube-apiserver-audit.log
          audit-policy-file: /etc/kubernetes/audit-policy/apiserver-audit-policy.yaml
          cloud-provider: aws
          encryption-provider-config: /etc/kubernetes/pki/encryption-config.yaml
        extraVolumes:
        - hostPath: /etc/kubernetes/audit-policy/
          mountPath: /etc/kubernetes/audit-policy/
          name: audit-policy
        - hostPath: /var/log/kubernetes/audit
          mountPath: /var/log/audit/
          name: audit-logs
      controllerManager:
        extraArgs:
          cloud-provider: aws
          configure-cloud-routes: "false"
      dns: {}
      etcd:
        local:
          imageTag: 3.4.13-0
      networking: {}
      scheduler: {}
    files:
    - content: |
        # Taken from https://github.com/kubernetes/kubernetes/blob/master/cluster/
gce/gci/configure-helper.sh
        # Recommended in Kubernetes docs
        apiVersion: audit.k8s.io/v1
        kind: Policy
        rules:
          # The following requests were manually identified as high-volume and low-
risk,
          # so drop them.
          - level: None
            users: ["system:kube-proxy"]
            verbs: ["watch"]
            resources:
```

```
      - group: "" # core
        resources: ["endpoints", "services", "services/status"]
  - level: None
    # Ingress controller reads 'configmaps/ingress-uid' through the unsecured
port.
    # TODO(#46983): Change this to the ingress controller service account.
    users: ["system:unsecured"]
    namespaces: ["kube-system"]
    verbs: ["get"]
    resources:
      - group: "" # core
        resources: ["configmaps"]
  - level: None
    users: ["kubelet"] # legacy kubelet identity
    verbs: ["get"]
    resources:
      - group: "" # core
        resources: ["nodes", "nodes/status"]
  - level: None
    userGroups: ["system:nodes"]
    verbs: ["get"]
    resources:
      - group: "" # core
        resources: ["nodes", "nodes/status"]
  - level: None
    users:
      - system:kube-controller-manager
      - system:kube-scheduler
      - system:serviceaccount:kube-system:endpoint-controller
    verbs: ["get", "update"]
    namespaces: ["kube-system"]
    resources:
      - group: "" # core
        resources: ["endpoints"]
  - level: None
    users: ["system:apiserver"]
    verbs: ["get"]
    resources:
      - group: "" # core
        resources: ["namespaces", "namespaces/status", "namespaces/finalize"]
  - level: None
    users: ["cluster-autoscaler"]
    verbs: ["get", "update"]
    namespaces: ["kube-system"]
    resources:
      - group: "" # core
        resources: ["configmaps", "endpoints"]
  # Don't log HPA fetching metrics.
  - level: None
    users:
      - system:kube-controller-manager
    verbs: ["get", "list"]
    resources:
```

```
            - group: "metrics.k8s.io"
          # Don't log these read-only URLs.
          - level: None
            nonResourceURLs:
              - /healthz*
              - /version
              - /swagger*
          # Don't log events requests.
          - level: None
            resources:
              - group: "" # core
                resources: ["events"]
          # node and pod status calls from nodes are high-volume and can be large,
don't log responses for expected updates from nodes
          - level: Request
            users: ["kubelet", "system:node-problem-detector",
"system:serviceaccount:kube-system:node-problem-detector"]
            verbs: ["update","patch"]
            resources:
              - group: "" # core
                resources: ["nodes/status", "pods/status"]
            omitStages:
              - "RequestReceived"
          - level: Request
            userGroups: ["system:nodes"]
            verbs: ["update","patch"]
            resources:
              - group: "" # core
                resources: ["nodes/status", "pods/status"]
            omitStages:
              - "RequestReceived"
          # deletecollection calls can be large, don't log responses for expected
namespace deletions
          - level: Request
            users: ["system:serviceaccount:kube-system:namespace-controller"]
            verbs: ["deletecollection"]
            omitStages:
              - "RequestReceived"
          # Secrets, ConfigMaps, and TokenReviews can contain sensitive & binary
data,
          # so only log at the Metadata level.
          - level: Metadata
            resources:
              - group: "" # core
                resources: ["secrets", "configmaps"]
              - group: authentication.k8s.io
                resources: ["tokenreviews"]
            omitStages:
              - "RequestReceived"
          # Get responses can be large; skip them.
          - level: Request
            verbs: ["get", "list", "watch"]
            resources:
```

```
            - group: "" # core
            - group: "admissionregistration.k8s.io"
            - group: "apiextensions.k8s.io"
            - group: "apiregistration.k8s.io"
            - group: "apps"
            - group: "authentication.k8s.io"
            - group: "authorization.k8s.io"
            - group: "autoscaling"
            - group: "batch"
            - group: "certificates.k8s.io"
            - group: "extensions"
            - group: "metrics.k8s.io"
            - group: "networking.k8s.io"
            - group: "node.k8s.io"
            - group: "policy"
            - group: "rbac.authorization.k8s.io"
            - group: "scheduling.k8s.io"
            - group: "settings.k8s.io"
            - group: "storage.k8s.io"
          omitStages:
            - "RequestReceived"
        # Default level for known APIs
        - level: RequestResponse
          resources:
            - group: "" # core
            - group: "admissionregistration.k8s.io"
            - group: "apiextensions.k8s.io"
            - group: "apiregistration.k8s.io"
            - group: "apps"
            - group: "authentication.k8s.io"
            - group: "authorization.k8s.io"
            - group: "autoscaling"
            - group: "batch"
            - group: "certificates.k8s.io"
            - group: "extensions"
            - group: "metrics.k8s.io"
            - group: "networking.k8s.io"
            - group: "node.k8s.io"
            - group: "policy"
            - group: "rbac.authorization.k8s.io"
            - group: "scheduling.k8s.io"
            - group: "settings.k8s.io"
            - group: "storage.k8s.io"
          omitStages:
            - "RequestReceived"
        # Default level for all other requests.
        - level: Metadata
          omitStages:
            - "RequestReceived"
    path: /etc/kubernetes/audit-policy/apiserver-audit-policy.yaml
    permissions: "0600"
  - content: |
      #!/bin/bash
```

```
        # CAPI does not expose an API to modify KubeProxyConfiguration
        # this is a workaround to use a script with preKubeadmCommand to modify the
kubeadm config files
        # https://github.com/kubernetes-sigs/cluster-api/issues/4512
        for i in $(ls /run/kubeadm/ | grep 'kubeadm.yaml\|kubeadm-join-config.yaml');
do
          cat <<EOF>> "/run/kubeadm//$i"
        ---
        kind: KubeProxyConfiguration
        apiVersion: kubeproxy.config.k8s.io/v1alpha1
        metricsBindAddress: "0.0.0.0:10249"
        EOF
        done
      path: /run/kubeadm/konvoy-set-kube-proxy-configuration.sh
      permissions: "0700"
    - content: |
        [metrics]
          address = "0.0.0.0:1338"
          grpc_histogram = false
      path: /etc/containerd/conf.d/konvoy-metrics.toml
      permissions: "0644"
    - content: |
        #!/bin/bash
        systemctl restart containerd

        SECONDS=0
        until crictl info
        do
          if (( SECONDS > 60 ))
          then
             echo "Containerd is not running. Giving up..."
             exit 1
          fi
          echo "Containerd is not running yet. Waiting..."
          sleep 5
        done
      path: /run/konvoy/restart-containerd-and-wait.sh
      permissions: "0700"
    - contentFrom:
        secret:
          key: value
          name: my-cluster-etcd-encryption-config
      owner: root:root
      path: /etc/kubernetes/pki/encryption-config.yaml
      permissions: "0640"
    format: cloud-config
    initConfiguration:
      localAPIEndpoint: {}
      nodeRegistration:
        kubeletExtraArgs:
          cloud-provider: aws
        name: '{{ ds.meta_data.local_hostname }}'
    joinConfiguration:
```

```
      discovery: {}
      nodeRegistration:
        kubeletExtraArgs:
          cloud-provider: aws
        name: '{{ ds.meta_data.local_hostname }}'
    preKubeadmCommands:
    - systemctl daemon-reload
    - /run/konvoy/restart-containerd-and-wait.sh
    - /run/kubeadm/konvoy-set-kube-proxy-configuration.sh
  machineTemplate:
    infrastructureRef:
      apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
      kind: AWSMachineTemplate
      name: my-cluster-control-plane
      namespace: default
    metadata: {}
  replicas: 3
  rolloutStrategy:
    rollingUpdate:
      maxSurge: 1
    type: RollingUpdate
  version: v1.23.12
```

Now a user can configure the fields below for the log backend. The log backend will write audit events to a file in JSON[315] format. You can configure the log audit backend using the `kube-apiserver` flags shown below:

```
audit-log-maxage
audit-log-maxbackup
audit-log-maxsize
audit-log-path
```

See upstream documentation[316]for more information.

After modifying the values appropriately, you can create the cluster by running the command below:

```
kubectl create -f {MY_CLUSTER_NAME}.yaml
```

Once the cluster is created, users can get the corresponding kubeconfig for the cluster by running the following command:

```
dkp get kubeconfig -c {MY_CLUSTER_NAME} >> {MY_CLUSTER_NAME}.conf
```

315 https://jsonlines.org/
316 https://kubernetes.io/docs/tasks/debug/debug-cluster/audit/#log-backend

6.12.1.2 Viewing the Audit Logs

Fluent Bit is disabled on the management cluster by default, to view the audit logs run the command below:

```
dkp diagnose --kubeconfig={MY_CLUSTER_NAME}.conf
```

A file similar to `support-bundle-2022-08-15T02_28_48.tar.gz` will be created. Untar the file using a command similar to the example below:

```
tar -xzf support-bundle-2022-08-15T02_28_48.tar.gz
```

Navigate to the node-diagnostics sub directory from the extracted file with a command like the one shown below:

```
cd support-bundle-2022-08-15T02_28_48/node-diagnostics
```

Finally, to find the audit logs run the following command:

```
$ find . -type f | grep audit.log
./ip-10-0-142-117.us-west-2.compute.internal/data/kube_apiserver_audit.log
./ip-10-0-148-139.us-west-2.compute.internal/data/kube_apiserver_audit.log
./ip-10-0-128-181.us-west-2.compute.internal/data/kube_apiserver_audit.log
```

See other related pages below:

Fluent bit(see page 579)

7 Advanced Kommander

After successfully building a cluster using the Konvoy component of DKP, you will select the type of Installation you would like for the Kommander component of DKP. There is a minimal Kommander installation shown below, but it is recommended to determine your actual advanced settings by following the steps in that section such as air-gapped.

7.1 Kommander Install Configuration

You can configure Kommander during the initial installation, and also post-installation using the Kommander CLI.

Review the Management cluster application requirements(see page 101) and Workspace platform application requirements(see page 430) to ensure that your cluster has sufficient resources.

7.1.1 Initializing a configuration file

To begin configuring Kommander, run the following command to initialize a default configuration file:

```
dkp install kommander --init > kommander.yaml
```

7.1.2 Configuring applications

After you have a default configuration file, you can then configure each `app` either inline **or** by referencing another YAML file. The configuration values for each `app` correspond to the Helm Chart values for the application.

After the initial deployment of Kommander, you can find the application Helm Charts by checking the `spec.chart.spec.sourceRef` field of the associated `HelmRelease`:

```
kubectl get helmreleases <application> -o yaml -n kommander
```

7.1.2.1 Inline configuration (using values)

In this example, you configure the `centralized-grafana` application with resource limits by defining the Helm Chart values in the Kommander configuration file.

```
apiVersion: config.kommander.mesosphere.io/v1alpha1
kind: Installation
apps:
  centralized-grafana:
    values: |
      grafana:
        resources:
          limits:
            cpu: 150m
            memory: 100Mi
          requests:
            cpu: 100m
            memory: 50Mi
...
```

7.1.2.2 Referencing another YAML file (using valuesFrom)

Alternatively, you could create another YAML file containing the configuration for `centralized-grafana` and reference that using `valuesFrom`. Point to this file by using either a relative path (from the configuration file location) or by using an absolute path.

```
cat > centralized-grafana.yaml <<EOF
grafana:
  resources:
    limits:
      cpu: 150m
      memory: 100Mi
    requests:
      cpu: 100m
      memory: 50Mi
EOF
```

```
apiVersion: config.kommander.mesosphere.io/v1alpha1
kind: Installation
apps:
  centralized-grafana:
    valuesFrom: centralized-grafana.yaml
...
```

7.1.3 Minimal Kommander installation

You can install Kommander with a bare minimum of applications on a small environment with smaller memory, storage, and CPU requirements for testing and demo purposes. Refer to the Install Kommander on a small environment(see page 381) documentation for more information.

7.1.4 Install with configuration file

Add the `--installer-config` flag to the `dkp install kommander` command to use a custom configuration file. To reconfigure applications, you can also run this command after the initial installation.

> An alternative to using the `--kubeconfig=<cluster-config>` flag is to initialize the KUBECONFIG environment variable. You can do this by running `export KUBECONFIG=<cluster-config>`. Setting your KUBECONFIG (either by flag or by environment variable) ensures that Kommander is installed on the workload cluster.

```
dkp install kommander --installer-config kommander.yaml --kubeconfig=<cluster-
kubeconfig>
```

When completed, you can verify your installation(see page 383).

7.1.5 Configure an Enterprise catalog

Configure an Enterprise catalog for DKP

DKP supports configuring default catalogs for clusters with Enterprise license.

7.1.5.1 Configure a default Enterprise catalog

To configure DKP to use a default catalog repository, create the following YAML file:

```
apiVersion: config.kommander.mesosphere.io/v1alpha1
kind: Installation
catalog:
  repositories:
    - name: dkp-catalog-applications
      labels:
        kommander.d2iq.io/project-default-catalog-repository: "true"
        kommander.d2iq.io/workspace-default-catalog-repository: "true"
        kommander.d2iq.io/gitapps-gitrepository-type: "dkp"
      gitRepositorySpec:
        url: https://github.com/mesosphere/dkp-catalog-applications
        ref:
          tag: v2.3.3
```

Use this configuration when installing or reconfiguring DKP by passing it to the `dkp install kommander` command:

```
dkp install kommander --installer-config <config_file>.yaml
```

- To ensure DKP is installed on the workload cluster, use the `--kubeconfig=cluster_name.conf` flag as an alternative to KUBECONFIG.
- When configuring the catalog repository post-upgrade, run `dkp install kommander --init > install.yaml` and update it accordingly with any custom configuration. This ensures you are using the proper default configuration values for the new DKP version.

The following section describes each label:

Label	Description
`kommander.d2iq.io/project-default-catalog-repository`	Indicates this acts as a Catalog Repository in all projects
`kommander.d2iq.io/workspace-default-catalog-repository`	Indicates this acts as a Catalog Repository in all workspaces
`kommander.d2iq.io/gitapps-gitrepository-type`	Indicates this Catalog Repository (and all its Applications) are certified to run on DKP

Air-gapped Catalog Configuration

When running in air-gapped environments, update the configuration by replacing `gitRepositorySpec` with the `path` field pointing to a local path of the DKP catalog applications git repository.

1. Download the DKP catalog application Git repository archive:

2. ```
   wget "https://downloads.d2iq.com/dkp/v2.3.3/dkp-catalog-applications-v2.3.3.tar.gz"
   ```

3. Update the configuration file with:

   ```
   apiVersion: config.kommander.mesosphere.io/v1alpha1
   kind: Installation
   catalog:
     repositories:
   ```
   ```

```
    - name: dkp-catalog-applications
      labels:
        kommander.d2iq.io/project-default-catalog-repository: "true"
        kommander.d2iq.io/workspace-default-catalog-repository: "true"
        kommander.d2iq.io/gitapps-gitrepository-type: "dkp"
      path: ./dkp-catalog-applications.tar.gz
```

4. Use this configuration when installing or reconfiguring DKP by passing it to the `dkp install kommander` command:

```
dkp install kommander --installer-config <config_file>.yaml
```

When configuring the catalog repository post-upgrade, run `dkp install kommander --init > install.yaml` and update it accordingly with any custom configuration. This ensures you are using the proper default configuration values for the new DKP version.

7.1.6 Configure a Custom Domain

Configure a custom domain during installation

Kommander supports configuring a custom domain name for accessing the DKP UI and other platform services. Additionally, you can provide a custom certificate for each domain, or one can be issued automatically by Let’s Encrypt, or other certificate authorities supporting the ACME protocol. Refer to Custom domains and certificates configuration(see page 547) for more information on use cases and alternatives.

This section provides instructions and examples on how to configure the DKP installation to add a customized domain and certificate on your **Essential cluster** or your **Management cluster**. If you want to customize the domain and certificate on any **Attached** or **Managed** cluster, refer to Configure a custom domain and certificate for your cluster(see page 548).

7.1.6.1 Prerequisite

- Review the DKP Install Configuration(see page 351) page to gain better understanding of how to initialize and edit a configuration file.

7.1.6.2 Configure a custom domain

To configure Kommander to use a custom domain, the domain name must be provided in an installation config file. For example, to use the domain `mycluster.example.com`, create the following file:

```
apiVersion: config.kommander.mesosphere.io/v1alpha1
kind: Installation
clusterHostname: <mycluster.example.com>
```

This configuration can be used when installing or reconfiguring Kommander by passing it to the `dkp install kommander` command:

```
dkp install kommander --installer-config <config_file.yaml>
```

To ensure Kommander is installed on the right cluster, use the `--kubeconfig=cluster_name.conf` flag as an alternative to KUBECONFIG.

After the command completes, obtain the cluster ingress IP address or hostname using the following command:

```
kubectl -n kommander get svc kommander-traefik -o go-template='{{with
index .status.loadBalancer.ingress 0}}{{or .hostname .ip}}{{end}}{{ "\n"}}'
```

Next, create a DNS record (for example, by using external-dns(see page 605)) for your custom hostname that resolves to the cluster ingress load balancer hostname or IP address. If the previous command returns a hostname, you should create a CNAME DNS entry that resolves to that hostname. If the cluster ingress is an IP address, create a DNS A record.

The domain must be resolvable from the client (your browser) and from the cluster. If you set up an `external-dns` service, it will take care of pointing the DNS record to the ingress of the cluster automatically. If you are manually creating a DNS record, you have to install Kommander first to obtain the load balancer address required for the DNS record. Refer to the examples below(see page 0) for more details on how and when to set up the DNS record.

7.1.6.3 Configure a custom certificate

If you want to use your own certificate for the configured domain, you need the following files (in PEM format):

- The certificate
- The certificate's private key
- The CA bundle (containing the root and intermediate certificates)

Specify the local file path to these files in the installation config file:

```
apiVersion: config.kommander.mesosphere.io/v1alpha1
kind: Installation

clusterHostname: <mycluster.example.com>
ingressCertificate:
  certificate: <certs/cert.pem>
  private_key: <certs/key.pem>
  ca: <certs/ca.pem>
```

7.1.6.4 Certificates that support ACME

You can configure the `cert-manager` to automatically issue a trusted certificate for the configured custom domain. The `cert-manager` also takes care of renewing the certificate before expiration.

The certificate must be supported by the *Automatic Certificate Management Environment* or ACME protocol.

Before you start, gather all relevant information (domain, email, keys, PEM file, others) from your certificate provider. In the following, we provide a few examples:

Let's Encrypt

What you need

- Your domain name
- Your email
- Basic understanding of how to initialize, configure and run a configuration file(see page 351)

Configure Let's Encrypt

This section provides information on how to set up a *Let's Encrypt* certificate for the cluster ingress. This allows most browsers to validate the certificate for the cluster when users try to log into the operations portal. DKP allows setting up *Let's Encrypt* in a few simple steps.

1. Open the `kommander.yaml` file:
    a. If you do not have the `kommander.yaml` file, initialize the configuration file(see page 351), so you can edit it in the following steps. **WARNING**: Initialize this file only ONCE, otherwise you will overwrite previous customizations.
    b. If you have initialized the configuration file already, open the `kommander.yaml` with the editor of your choice.
2. Provide the acquired domain name in the `clusterHostname` field, enable `acme`, and add an `email` to register with Let's encrypt.

```
apiVersion: config.kommander.mesosphere.io/v1alpha1
kind: Installation
clusterHostname: mycluster.example.com
acme:
  email: <your_email>
```

NOTE: Let's Encrypt uses this email to contact you about expiring certificates, and issues related to your account.

3. Create a DNS record and install Kommander:
    a. You can set up an external-dns service(see page 605). This way, the `external-dns` will take care of pointing the DNS record to the ingress of the cluster automatically.
    In this case: **FIRST**, set up the `external-dns` in the `kommander.yaml`. **THEN** use the configuration file to install Kommander(see page 0).
    b. Alternatively, create a DNS record manually, that maps your domain name or IP address to the cluster ingress.
    In this case: **FIRST**, use the configuration file to install Kommander(see page 0) and wait for the load balancer address to be provisioned. **THEN** manually create the DNS record pointing to the load balancer address.

ZeroSSL

What you need

- Your domain name
- Your email
- An access and a secret key provided by ZeroSSL
- Basic understanding of how to initialize, configure and run a configuration file(see page 351)

Configure ZeroSSL

This section provides information on how to set up a ZeroSSL certificate for the cluster ingress. This allows most browsers to validate the certificate for the cluster when users try to log into the operations portal.

1. Open the `kommander.yaml` file:
    a. If you do not have the `kommander.yaml` file, initialize the configuration file(see page 351), so you can edit it in the following steps. **WARNING**: Initialize this file only ONCE, otherwise you will overwrite previous customizations.
    b. If you have initialized the configuration file already, open the `kommander.yaml` with the editor of your choice.
2. Provide the acquired domain name in the `clusterHostname` field, enable `acme`, and add an `email` and `server` to register with ZeroSSL's.

```
apiVersion: config.kommander.mesosphere.io/v1alpha1
kind: Installation
clusterHostname: <mycluster.example.com>
acme:
  email: <email>
  server: https://acme.zerossl.com/v2/DV90
```

3. Create a DNS record:
    a. You can set up an external-dns service(see page 605). This way, the `external-dns` will take care of pointing the DNS record to the ingress of the cluster automatically.
    b. Alternatively, create a DNS record manually, that maps your domain name or IP address to the cluster ingress.
4. Use the configuration file to install Kommander(see page 0).

5. Set up External Account Bindings[317] by customizing the `Issuer` or `ClusterIssuer` details, as shown in the Customize Issuer Details(see page 0) section.

Other ACME issuers

1. Open the `kommander.yaml` file:
   a. If you do not have the `kommander.yaml` file, initialize the configuration file(see page 351), so you can edit it in the following steps. **WARNING**: Initialize this file only ONCE, otherwise you will overwrite previous customizations.
   b. If you have initialized the configuration file already, open the `kommander.yaml` with the editor of your choice.
2. You can use other issuers that support the ACME protocol by configuring the issuer's server in the installation configuration, for example:

```
acme:
  email: <your_email>
  server: <your_server>
[...]
```

3. Create a DNS record and install Kommander:
   a. You can set up an external-dns service. This way, the `external-dns` will take care of pointing the DNS record to the ingress of the cluster automatically.
   In this case: **FIRST**, set up the `external-dns` in the `kommander.yaml`. **THEN** use the configuration file to install Kommander(see page 0).
   b. Alternatively, create a DNS record manually, that maps your domain name or IP address to the cluster ingress.
   In this case: **FIRST**, use the configuration file to install Kommander(see page 0) and wait for the load balancer address to be provisioned. **THEN** manually create the DNS record pointing to the load balancer address.

⚠ Some certificate authorities require setting additional fields for the custom configuration to work. Use the `kubectl -n <namespace> patch` command to configure any other additional fields in the `Issuer` or `ClusterIssuer`. For an example, refer to the ZeroSSL configuration(see page 0).

Customize issuer details

By default, `dkp install kommander` sets up a working ACME solver using HTTP01 challenges. If further control over the certificate issuing is needed, you can modify the pre-configured `ClusterIssuer`. For example, you can use a DNS01 challenge:

```
cat <<EOF | kubectl apply -f -
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
```

317 https://cert-manager.io/docs/configuration/acme/#external-account-bindings

```
metadata:
  name: kommander-acme-issuer
spec:
  acme:
    email: <your_email>
    server: https://acme-v02.api.letsencrypt.org/directory
    privateKeySecretRef:
      name: kommander-acme-issuer-account
    solvers:
      - dns01:
          route53:
            region: us-east-1
            role: arn:aws:iam::YYYYYYYYYYYY:role/dns-manager
EOF
```

For more information on the available options, refer to the ACME section in the cert-manager documentation[318].

⚠ If you need to make changes in the configuration of your custom domain or certificate after you have installed DKP, modify the `ingress` in the `KommanderCluster` object as shown in the Custom domains and certificates configuration(see page 547) section.

7.1.7 Configure HTTP Proxy

Configure HTTP proxy for the Kommander cluster(s)

Kommander supports environments where access to the Internet is restricted, and must be made through an HTTP/HTTPS proxy.

In these environments, you must configure Kommander to use the HTTP/HTTPS proxy. In turn, Kommander configures all platform services to use the HTTP/HTTPS proxy.

Kommander follows a common convention for using an HTTP proxy server. The convention is based on three environment variables, and is supported by many, though not all, applications.

- `HTTP_PROXY` : the HTTP proxy server address
- `HTTPS_PROXY` : the HTTPS proxy server address
- `NO_PROXY` : a list of IPs and domain names that are not subject to proxy settings

7.1.7.1 Prerequisites

In the examples below:

1. The `curl` command-line tool is available on the host.
2. The proxy server address is `http://proxy.company.com:3128` .
3. The HTTP and HTTPS proxy server addresses use the `http` scheme.
4. The proxy server can reach `www.google.com` using HTTP or HTTPS.

318 https://cert-manager.io/docs/configuration/acme/

Verify the cluster nodes can access the Internet through the proxy server. On each cluster node, run:

```
curl --proxy http://proxy.company.com:3128 --head http://www.google.com
curl --proxy http://proxy.company.com:3128 --head https://www.google.com
```

If the proxy is working for HTTP and HTTPS, respectively, the `curl` command returns a `200 OK` HTTP response.

7.1.7.2 Enable Gatekeeper

Gatekeeper acts as a Kubernetes mutating webhook[319]. You can use this to mutate the Pod resources with `HTTP_PROXY`, `HTTPS_PROXY` and `NO_PROXY` environment variables.

1. Create (if necessary) or update the Kommander installation configuration file. If one does not already exist, then create it using the following commands:

   ```
   ./dkp install kommander --init > install.yaml
   ```

2. Append this `apps` section to the `install.yaml` file with the following values to enable Gatekeeper and configure it to add HTTP proxy settings to the pods.
   NOTE: Only pods created after applying this setting will be mutated. Also, this will only affect pods in the namespace with the `"gatekeeper.d2iq.com/mutate=pod-proxy"` label.

   ```
   apps:
     gatekeeper:
       values: |
         disableMutation: false
         mutations:
           enablePodProxy: true
           podProxySettings:
             noProxy:
   "127.0.0.1,192.168.0.0/16,10.0.0.0/16,10.96.0.0/12,169.254.169.254,169.254.0.0/
   24,localhost,kubernetes,kubernetes.default,kubernetes.default.svc,kubernetes.de
   fault.svc.cluster,kubernetes.default.svc.cluster.local,.svc,.svc.cluster,.svc.c
   luster.local,.svc.cluster.local.,kubecost-prometheus-server.kommander,ogging-
   operator-logging-fluentd.kommander.svc.cluster.local,elb.amazonaws.com"
             httpProxy: "http://proxy.company.com:3128"
             httpsProxy: "http://proxy.company.com:3128"
           excludeNamespacesFromProxy: []
           namespaceSelectorForProxy:
             "gatekeeper.d2iq.com/mutate": "pod-proxy"
   ```

3. Create the `kommander` and `kommander-flux` namespaces, or the namespace where Kommander will be installed. Label the namespaces to activate the Gatekeeper mutation on them:

319 https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/#mutatingadmissionwebhook

```
kubectl create namespace kommander
kubectl label namespace kommander gatekeeper.d2iq.com/mutate=pod-proxy

kubectl create namespace kommander-flux
kubectl label namespace kommander-flux gatekeeper.d2iq.com/mutate=pod-proxy
```

7.1.7.3 Create Gatekeeper ConfigMap in the kommander Namespace

To configure Gatekeeper so that these environment variables are mutated in the pods, create the following `gatekeeper-overrides` ConfigMap in the `kommander` Workspace you created in a previous step:

```
export NAMESPACE=kommander
```

```
cat << EOF | kubectl apply -f -
apiVersion: v1
kind: ConfigMap
metadata:
  name: gatekeeper-overrides
  namespace: ${NAMESPACE}
data:
  values.yaml: |
    ---
    # enable mutations
    disableMutation: false
    mutations:
      enablePodProxy: true
      podProxySettings:
        noProxy:
"127.0.0.1,192.168.0.0/16,10.0.0.0/16,10.96.0.0/12,169.254.169.254,169.254.0.0/24,loc
alhost,kubernetes,kubernetes.default,kubernetes.default.svc,kubernetes.default.svc.cl
uster,kubernetes.default.svc.cluster.local,.svc,.svc.cluster,.svc.cluster.local,.svc.
cluster.local.,kubecost-prometheus-server.kommander,ogging-operator-logging-
fluentd.kommander.svc.cluster.local,elb.amazonaws.com"
        httpProxy: "http://proxy.company.com:3128"
        httpsProxy: "http://proxy.company.com:3128"
      excludeNamespacesFromProxy: []
      namespaceSelectorForProxy:
        "gatekeeper.d2iq.com/mutate": "pod-proxy"
EOF
```

Set the `httpProxy` and `httpsProxy` environment variables to the address of the HTTP and HTTPS proxy servers, respectively. Set the `noProxy` environment variable to the addresses that should be accessed directly, not through the proxy.

Performing this step before installing Kommander allows the Flux components to respect the proxy configuration in this ConfigMap.

7.1.7.4 HTTP Proxy Configuration Considerations

To ensure that core components work correctly, always add these addresses to the `noProxy`:

- Loopback addresses (`127.0.0.1` and `localhost`)
- Kubernetes API Server addresses
- Kubernetes Pod IPs (for example, `192.168.0.0/16`). This comes from two places:
  - Calico pod CIDR - Defaults to `192.168.0.0/16`
  - The `podSubnet` is configured in CAPI objects and needs to match above Calico's - Defaults to `192.168.0.0/16` (same as above)
- Kubernetes Service addresses (for example, `10.96.0.0/12`, `kubernetes`, `kubernetes.default`, `kubernetes.default.svc`, `kubernetes.default.svc.cluster`, `kubernetes.default.svc.cluster.local`, `.svc`, `.svc.cluster`, `.svc.cluster.local`, `.svc.cluster.local.`)
- Auto-IP addresses `169.254.169.254,169.254.0.0/24`

In addition to the values above, the following settings are needed when installing on AWS:

- The default VPC CIDR range of `10.0.0.0/16`
- `kube-apiserver` internal/external ELB address

⚠
- The `NO_PROXY` variable contains the Kubernetes Services CIDR. This example uses the default CIDR, `10.96.0.0/12`. If your cluster's CIDR is different, update the value in the `NO_PROXY` field.
- Based on the order in which the Gatekeeper Deployment is Ready (in relation to other Deployments), not all the core services are guaranteed to be mutated with the proxy environment variables. Only the user deployed workloads are guaranteed to be mutated with the proxy environment variables. If you need a core service to be mutated with your proxy environment variables, you can restart the AppDeployment for that core service.

7.1.7.5 Install Kommander

Kommander installs with the DKP CLI. Install Kommander using the configuration files and ConfigMap from previous steps:

NOTE: To ensure Kommander is installed on the workload cluster, use the `--kubeconfig=cluster_name.conf` flag:

```
./dkp install kommander --installer-config ./install.yaml
```

7.1.7.6 Configure Workspace or Project

Configure the Workspace or Project in which you want to use the proxy. To have Gatekeeper mutate the manifests, create the `Workspace` (or `Project`) with the following label:

```
labels:
  gatekeeper.d2iq.com/mutate: "pod-proxy"
```

This can be done when creating the Workspace (or Project) from the UI OR by running the following command from the CLI once the namespace is created:

```
kubectl label namespace <NAMESPACE> "gatekeeper.d2iq.com/mutate=pod-proxy"
```

7.1.7.7 Configure HTTP Proxy in Attached Clusters

To ensure that Gatekeeper is deployed before everything else in the attached clusters that you want to configure with proxy configuration, you must manually create the exact Namespace of the Workspace in which the cluster is going to be attached, *before* attaching the cluster:

Execute the following command in the attached cluster before attaching it to the host cluster:

```
kubectl create namespace <NAMESPACE>
```

Then, to configure the pods in this namespace to use proxy configuration, you must label the Workspace with `gatekeeper.d2iq.com/mutate=pod-proxy` when creating it so that Gatekeeper deploys a `validatingwebhook` to mutate the pods with proxy configuration.

```
kubectl label namespace <NAMESPACE> "gatekeeper.d2iq.com/mutate=pod-proxy"
```

7.1.7.8 Create Gatekeeper ConfigMap in the Workspace Namespace

To configure Gatekeeper so that these environment variables are mutated in the pods, create the following `gatekeeper-overrides` ConfigMap in the Workspace Namespace:

```
export NAMESPACE=<NAMESPACE>
```

```
cat << EOF | kubectl apply -f -
apiVersion: v1
kind: ConfigMap
metadata:
  name: gatekeeper-overrides
```

```
  namespace: ${NAMESPACE}
data:
  values.yaml: |
    ---
    # enable mutations
    disableMutation: false
    mutations:
      enablePodProxy: true
      podProxySettings:
        noProxy:
"127.0.0.1,192.168.0.0/16,10.0.0.0/16,10.96.0.0/12,169.254.169.254,169.254.0.0/24,loc
alhost,kubernetes,kubernetes.default,kubernetes.default.svc,kubernetes.default.svc.cl
uster,kubernetes.default.svc.cluster.local,.svc,.svc.cluster,.svc.cluster.local,.svc.
cluster.local.,kubecost-prometheus-server.kommander,ogging-operator-logging-
fluentd.kommander.svc.cluster.local,elb.amazonaws.com"
        httpProxy: "http://proxy.company.com:3128"
        httpsProxy: "http://proxy.company.com:3128"
      excludeNamespacesFromProxy: []
      namespaceSelectorForProxy:
        "gatekeeper.d2iq.com/mutate": "pod-proxy"
EOF
```

Set the `httpProxy` and `httpsProxy` environment variables to the address of the HTTP and HTTPS proxy servers, respectively. Set the `noProxy` environment variable to the addresses that should be accessed directly, not through the proxy. The list of the recommended settings is in the section *HTTP Proxy Configuration Considerations* above.

7.1.7.9 Configure Your Applications

In a default installation with `gatekeeper` enabled, you can have proxy environment variables applied to all your pods automatically by adding the following label to your namespace:

```
"gatekeeper.d2iq.com/mutate": "pod-proxy"
```

No further manual changes are required.

7.1.7.10 Manually Configure Your Application

⚠ If Gatekeeper is not installed, and you need to use an HTTP proxy, you must manually configure your applications.

Some applications follow the convention of `HTTP_PROXY`, `HTTPS_PROXY`, and `NO_PROXY` environment variables.

In this example, the environment variables are set for a container in a Pod:

See Define Environment Variables for a Container[320] for more details.

Next Steps:

Now select your environment, and finish your Kommander Installation in one of the following:

- Install Kommander in Air-gapped(see page 360)
- Install Kommander in Nonair-gapped(see page 360)
- Install Kommander in Small Environment(see page 360)

7.2 Install Kommander in a Networked Environment

7.2.1 Prerequisites

Before you begin using DKP UI, you must:

- Install the DKP binary(see page 83) before executing any `dkp` commands. Ensure you have the version of the CLI that matches the DKP version you want to install.
- Configure a Konvoy cluster using the Advanced Infrastructure Providers(see page 103) instructions for a cluster based on environment.
- Review the Management cluster application requirements(see page 101) and Workspace platform application requirements(see page 430) to ensure that your cluster has sufficient resources.

7.2.1.1 Default StorageClass

To ensure the Git repository that Kommander ships with deploys successfully, the cluster where Kommander is installed must have a default `StorageClass` configured. Run the following command:

```
kubectl get sc
```

The output should look similar to this. Note the `(default)` after the name:

```
NAME               PROVISIONER       RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION   AGE
ebs-sc (default)   ebs.csi.aws.com   Delete          WaitForFirstConsumer   false
41s
```

If the `StorageClass` is not set as default, add the following annotation to the `StorageClass` manifest:

```
annotations:
```

320 https://kubernetes.io/docs/tasks/inject-data-application/define-environment-variable-container/#define-an-environment-variable-for-a-container

```
    storageclass.kubernetes.io/is-default-class: "true"
```

More information on setting a StorageClass as default can be found at Changing the default storage class in k8s docs[321].

7.2.2 Install Kommander on Konvoy

To customize a Kommander installation, see the configuration page(see page 351) for more details.

Before running the commands below, ensure that your `kubectl` configuration **references the cluster on which you want to install Kommander**, otherwise it will install on the bootstrap cluster. You can do this by setting the `KUBECONFIG` environment variable to the appropriate kubeconfig file's location[322].

> An alternative to initializing the KUBECONFIG environment variable as stated earlier is to use the `--kubeconfig=cluster_name.conf` flag. This ensures that Kommander is installed on the workload cluster.

Begin with this command:

```
dkp install kommander
```

7.2.3 Verify installation

After the CLI successfully installs the components, you must wait for all `HelmReleases` to deploy.

The Kommander installation is a multi-step process: Flux installs first, then the Git repository spins up permitting Flux to consume further `HelmReleases` from that repository.

After running the install command above, `HelmReleases` begin to appear on the cluster.

```
kubectl -n kommander wait --for condition=Released helmreleases --all --timeout 15m
```

This will wait for each of the helm charts to reach their `Released` condition, eventually resulting in something resembling this:

```
helmrelease.helm.toolkit.fluxcd.io/centralized-grafana condition met
helmrelease.helm.toolkit.fluxcd.io/dex condition met
helmrelease.helm.toolkit.fluxcd.io/dex-k8s-authenticator condition met
helmrelease.helm.toolkit.fluxcd.io/fluent-bit condition met
helmrelease.helm.toolkit.fluxcd.io/gitea condition met
helmrelease.helm.toolkit.fluxcd.io/grafana-logging condition met
helmrelease.helm.toolkit.fluxcd.io/grafana-loki condition met
```

321 https://kubernetes.io/docs/tasks/administer-cluster/change-default-storage-class
322 https://kubernetes.io/docs/tasks/access-application-cluster/configure-access-multiple-clusters/

```
helmrelease.helm.toolkit.fluxcd.io/karma condition met
helmrelease.helm.toolkit.fluxcd.io/kommander condition met
helmrelease.helm.toolkit.fluxcd.io/kommander-appmanagement condition met
helmrelease.helm.toolkit.fluxcd.io/kube-prometheus-stack condition met
helmrelease.helm.toolkit.fluxcd.io/kubecost condition met
helmrelease.helm.toolkit.fluxcd.io/kubecost-thanos-traefik condition met
helmrelease.helm.toolkit.fluxcd.io/kubefed condition met
helmrelease.helm.toolkit.fluxcd.io/kubernetes-dashboard condition met
helmrelease.helm.toolkit.fluxcd.io/kubetunnel condition met
helmrelease.helm.toolkit.fluxcd.io/logging-operator condition met
helmrelease.helm.toolkit.fluxcd.io/logging-operator-logging condition met
helmrelease.helm.toolkit.fluxcd.io/minio-operator condition met
helmrelease.helm.toolkit.fluxcd.io/prometheus-adapter condition met
helmrelease.helm.toolkit.fluxcd.io/prometheus-thanos-traefik condition met
helmrelease.helm.toolkit.fluxcd.io/reloader condition met
helmrelease.helm.toolkit.fluxcd.io/thanos condition met
helmrelease.helm.toolkit.fluxcd.io/traefik condition met
helmrelease.helm.toolkit.fluxcd.io/traefik-forward-auth-mgmt condition met
helmrelease.helm.toolkit.fluxcd.io/velero condition met
```

You can check the status of a `HelmRelease` with:

```
kubectl -n kommander get helmrelease <HELMRELEASE_NAME>
```

If you find any `HelmReleases` in a “broken” release state such as “exhausted” or “another rollback/release in progress”, you can trigger a reconciliation of the `HelmRelease` using the following commands:

```
kubectl -n kommander patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[{"op":
"replace", "path": "/spec/suspend", "value": true}]'
kubectl -n kommander patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[{"op":
"replace", "path": "/spec/suspend", "value": false}]'
```

7.2.3.1 Verify Installation

When completed, you can verify your installation(see page 383).

7.2.4 Access DKP UI

When all the `HelmReleases` are ready, use the following command to open the DKP UI in your browser:

```
dkp open dashboard
```

This command opens the URL of the Kommander web interface in your default browser, and prints the username and password in the CLI.

If you prefer not to open your browser, or are running the DKP CLI from a host that does not have a browser installed, run this command to retrieve the URL used for accessing the DKP UI:

```
kubectl -n kommander get svc kommander-traefik -o go-template='https://{{with
index .status.loadBalancer.ingress 0}}{{or .hostname .ip}}{{end}}/dkp/kommander/
dashboard{{ "\n"}}'
```

And, use the following command to access the username and password stored on the cluster:

```
kubectl -n kommander get secret dkp-credentials -o go-template='Username:
{{.data.username|base64decode}}{{ "\n"}}Password: {{.data.password|base64decode}}
{{ "\n"}}'
```

7.3 Install DKP in an Air-gapped Environment

How to run DKP using an Air-gapped installation

The following explains how to run DKP on top of an air-gapped DKP cluster(see page 157) installation.

7.3.1 Prerequisites

Before installing, ensure you have:

- A Docker registry containing all the necessary Docker installation images, including the Kommander images. The `kommander-image-bundle.tar` tarball has the required artifacts.
- A charts bundle file containing all Helm charts that Kommander installation needs.
- Connectivity with clusters attaching to the management cluster:
  - Both management and attached clusters must be able to connect to the Docker registry.
  - The management cluster must be able to connect to all attached cluster's API servers.
  - The management cluster must be able to connect to any load balancers created for platform services on the management cluster.
- A configuration file(see page 351) that you can adapt to your needs using the steps outlined in this topic. Make sure to create that file using the following command:

```
dkp install kommander --init --airgapped > install.yaml
```

- All the prerequisites covered in air-gapped DKP installation(see page 0).
- Sufficient resources on your cluster to run Kommander. Review the Management cluster application requirements(see page 101) and Workspace platform application requirements(see page 430) for application requirements.
- The image bundle files downloaded(see page 83).
- A load balancer to route external traffic which is provided by your cloud provider. For on-premises deployments, you must configure MetalLB. See Load Balancing(see page 604) topic for more details.

7.3.2 Kommander Charts Bundle

The charts bundle is a gzipped Tar archive containing Helm charts, which are required during Kommander installation. Create the charts bundle with the DKP CLI or downloaded along with the DKP CLI. Execute this command to create the charts bundle:

```
dkp create chart-bundle
```

Kommander creates `charts-bundle.tar.gz`. Optionally, specify the output using the `-o` parameter:

```
dkp create chart-bundle -o [name of the output file]
```

7.3.3 Kommander Internal Helm Repository

The Kommander charts bundle is pushed to Kommander's internal Helm repository. To inspect the contents:

```
dkp get charts
```

Individual charts can be removed using:

```
dkp delete chart [chartName] [chartVersion]
```

It is possible to push new charts as well:

```
dkp push chart [chartTarball]
```

Or push a new bundle:

```
dkp push chart-bundle [chartsTarball]
```

Check the built-in help text for each command for more information.

7.3.4 Load the Docker images into your Docker registry

1. See the `NOTICES.txt` file for 3rd party software attributions and place the `kommander-image-bundle-v2.3.3.tar` bundle within a location where you can load and push the images to your private Docker registry.
2. Run the following command to load the air-gapped image bundle into your private Docker registry:

```
dkp push image-bundle --image-bundle kommander-image-bundle-v2.3.3.tar --to-
registry <REGISTRY_URL>
```

It may take a while to push all the images to your image registry, depending on the performance of the network between the machine you are running the script on and the Docker registry.

7.3.5 Install on Konvoy

This docker image includes code from the MinIO Project ("MinIO"), which is © 2015-2021 MinIO, Inc. MinIO is made available subject to the terms and conditions of the GNU Affero General Public License 3.0[323].

Complete source code for MinIO is available here[324], here[325], and here[326]

1. Create the configuration file(see page 351) by running `dkp install kommander --init --airgapped > install.yaml` for the air-gapped deployment. Open the `install.yaml` file and review that it looks like the following:

```
apiVersion: config.kommander.mesosphere.io/v1alpha1
kind: Installation
airgapped:
  enabled: true
```

2. In the same file, if you are installing Kommander in an AWS VPC, set the Traefik annotation to create an internal facing ELB by setting the following:

```
apps:
  traefik:
    values: |
      service:
        annotations:
          service.beta.kubernetes.io/aws-load-balancer-internal: "true"
```

3. Download the Kommander application definitions:

```
wget "https://downloads.d2iq.com/dkp/v2.3.3/kommander-applications-
v2.3.3.tar.gz"
```

4. Download the Kommander charts bundle:

323 https://www.gnu.org/licenses/agpl-3.0.en.html
324 https://github.com/minio/minio/tree/RELEASE.2021-02-14T04-01-33Z
325 https://github.com/minio/minio/tree/RELEASE.2022-02-24T22-12-01Z
326 https://github.com/minio/minio/tree/RELEASE.2022-01-08T03-11-54Z

```
wget "https://downloads.d2iq.com/dkp/v2.3.3/dkp-kommander-charts-bundle-
v2.3.3.tar.gz" && tar -xvf dkp-kommander-charts-bundle-v2.3.3.tar.gz
```

5. To install Kommander in your air-gapped environment using the above configuration file, enter the following command:

```
dkp install kommander --installer-config ./install.yaml \
--kommander-applications-repository kommander-applications-v2.3.3.tar.gz \
--charts-bundle dkp-kommander-charts-bundle-v2.3.3.tar.gz
```

6. When completed, you can verify your installation(see page 383).

This Docker image includes code from the MinIO Project (“MinIO”), which is © 2015-2021 MinIO, Inc. MinIO is made available subject to the terms and conditions of the GNU Affero General Public License 3.0. The complete source code for the versions of MinIO packaged with DKP/Kommander/Konvoy 2.3.2 are available at these URLs:

7.3.5.1 https://github.com/minio/minio/tree/RELEASE.2022-01-08T03-11-54Z

7.3.5.2 https://github.com/minio/minio/tree/RELEASE.2021-02-14T04-01-33Z

7.3.6 Install DKP in an Air-gapped Environment with Catalog Applications

How to run DKP in an Air-gapped installation with catalog applications.

The following explains how to run DKP on top of an air-gapped DKP cluster(see page 157) installation with catalog applications.

Depending on your configuration, there are three different ways you can install DKP to an air-gapped environment.
Ensure you follow the correct procedure for your configuration type, and ignore the other two sections that do not pertain to your environment:

- Install air-gapped Kommander with DKP Catalog Applications (see page 373)
- Install air-gapped Kommander with DKP Insights(see page 375)
- Install air-gapped Kommander with DKP Insights and DKP Catalog Applications(see page 378)

7.3.6.1 Load the Docker images into your Docker Registry

Follow these steps:

1. Download the DKP image bundle file:

```
curl -LJ https://downloads.d2iq.com/dkp/v2.3.3/kommander-image-bundle-
v2.3.3.tar | tar -xv
```

2. Download the Konvoy Docker Images.

```
curl -o konvoy-image-bundle-v2.3.3.tar -O https://downloads.d2iq.com/dkp/
v2.3.3/konvoy_image_bundle_v2.3.3_linux_amd64.tar
```

3. Optionally download the DKP catalog applications image bundle file:

```
curl "https://downloads.d2iq.com/dkp/v2.3.3/dkp-catalog-applications-image-
bundle-v2.3.3.tar" -O && tar -xvf dkp-catalog-applications-image-bundle-v2.3.3.
tar
```

4. Optionally download the DKP insights image bundle file:

```
curl "https://downloads.d2iq.com/dkp/v2.3.3/dkp-insights-image-bundle-
v2.3.3.tar" -O
```

5. See the `NOTICES.txt` file for 3rd party software attributions and place the `kommander-image-bundle-v2.3.3.tar` and `dkp-catalog-applications-image-bundle-v2.3.3.tar` bundles within a location where you can load and push the images to your private Docker registry.
6. Run the following command to load the air-gapped image bundle into your private Docker registry:

```
dkp push image-bundle --image-bundle kommander-image-bundle-v2.3.3.tar --to-registry
<REGISTRY_URL>
dkp push image-bundle --image-bundle dkp-catalog-applications-image-bundle-v2.3.3.tar
--to-registry <REGISTRY_URL>
dkp push image-bundle --image-bundle dkp-insights-image-bundle-v2.3.3.tar --to-
registry <REGISTRY_URL>
```

7.3.6.2 Install air-gapped Kommander with the DKP Catalog Applications

Use this section to install DKP with Catalog Applications.

Prerequisites

To use the DKP Catalog Applications in an air-gapped environment, you need the following files (including downloading and pushing the `dkp-insights-image-bundle` file mentioned above):

1. Download the DKP catalog application definitions:

```
curl https://downloads.d2iq.com/dkp/v2.3.3/dkp-catalog-applications-
v2.3.3.tar.gz -O
```

2. Download the DKP catalog applications(see page 441) chart bundle:

```
curl -LJ https://downloads.d2iq.com/dkp/v2.3.3/dkp-catalog-applications-charts-
bundle-v2.3.3.tar.gz | tar -xvz
```

3. Download the Kommander charts bundle:

```
curl -LJ https://downloads.d2iq.com/dkp/v2.3.3/dkp-kommander-charts-bundle-
v2.3.3.tar.gz | tar -xvz
```

4. Download the Kommander application definitions:

```
curl -O https://downloads.d2iq.com/dkp/v2.3.3/kommander-applications-
v2.3.3.tar.gz
```

5. Download the Containerd `1.14.13` packages for the OS you plan to provision dkp on. The options for OS are listed below for replacement in the command before running:
   - `centos-7.9`
   - `ol-7.9`
   - `rhel-7.9`
   - `rhel-8.2`
   - `rhel-8.4`
   - `sles-15.3`
   - `ubuntu-18.04`
   - `ubuntu-20.04`

```
export CONTAINERD_OS=centos-7.9
```

```
curl --output artifacts/containerd-1.4.13-d2iq.1-"$CONTAINERD_OS"-x86_64.tar.gz
--location https://packages.d2iq.com/dkp/containerd/containerd-1.4.13-d2iq.1-"$
CONTAINERD_OS"-x86_64.tar.gz
```

   To get the fips builds append `_fips` after `-x86_64` in the url. To get the fips build for `centos-7.9` the url would be `https://packages.d2iq.com/dkp/containerd/containerd-1.4.13-d2iq.1-centos-7.9-x86_64_fips.tar.gz`

   The following OS's have containerd fips builds:
   - `centos-7.9`
   - `ol-7.9`
   - `rhel-7.9`
   - `rhel-8.2`
   - `rhel-8.4`

6. Set up the
7. Download the Kubernetes images:

```
curl -o kib/artifacts/images/"$VERSION"_images.tar.gz -O https://
downloads.d2iq.com/dkp/airgapped/kubernetes-images/"$VERSION"_images.tar.gz
```

Install Kommander

Follow these steps:

1. Create the configuration file[327] by running `dkp install kommander --init --airgapped > install.yaml` for the air-gapped deployment. Open the `install.yaml` file and review that it looks like the following:

```
apiVersion: config.kommander.mesosphere.io/v1alpha1
kind: Installation
airgapped:
  enabled: true
```

2. In the same file, if you are installing Kommander in an AWS VPC, set the Traefik annotation to create an internal facing ELB by setting the following:

```
apps:
  traefik:
    values: |
      service:
        annotations:
          service.beta.kubernetes.io/aws-load-balancer-internal: "true"
catalog:
  repositories:
    - name: dkp-catalog-applications
      labels:
        kommander.d2iq.io/project-default-catalog-repository: "true"
        kommander.d2iq.io/workspace-default-catalog-repository: "true"
        kommander.d2iq.io/gitapps-gitrepository-type: "dkp"
      path: ./dkp-catalog-applications.tar.gz
```

3. To install DKP in your air-gapped environment using the above configuration file, run the following command:

```
dkp install kommander --installer-config ./install.yaml \
--kommander-applications-repository kommander-applications-v2.3.3.tar.gz \
--charts-bundle dkp-kommander-charts-bundle-v2.3.3.tar.gz \
--charts-bundle dkp-catalog-applications-charts-bundle-v2.3.3.tar.gz
```

4. Verify your installation(see page 0).

7.3.6.3 Install air-gapped Kommander with DKP Insights

Use this section to install DKP with DKP Insights.

327 https://d2iq.atlassian.net/wiki/pages/createpage.action?spaceKey=DKP&title=Kommander+Install+Configuration

Prerequisites

If you are utilizing DKP Insights[328] in an air-gapped environment, there are additional files in order to use the DKP Insights engine:

1. Download the DKP Insights catalog:

```
curl "https://downloads.d2iq.com/dkp/v2.3.3/dkp-insights-v2.3.3.tar.gz" -O
```

2. Download the DKP Insights chart bundle:

```
curl "https://downloads.d2iq.com/dkp/v2.3.3/dkp-insights-charts-bundle-
v2.3.3.tar.gz" -O
```

3. Download the Kommander charts bundle:

```
curl "https://downloads.d2iq.com/dkp/v2.3.3/dkp-kommander-charts-bundle-
v2.3.3.tar.gz" -O && tar -xvf dkp-kommander-charts-bundle-v2.3.3.tar.gz
```

4. Download the Kommander application definitions:

```
curl -O https://downloads.d2iq.com/dkp/v2.3.3/kommander-applications-
v2.3.3.tar.gz
```

5. Download the Containerd `1.14.13` packages for the OS you plan to provision dkp on. The options for OS are listed below for replacement in the command before running:
   - `centos-7.9`
   - `ol-7.9`
   - `rhel-7.9`
   - `rhel-8.2`
   - `rhel-8.4`
   - `sles-15.3`
   - `ubuntu-18.04`
   - `ubuntu-20.04`

```
export CONTAINERD_OS=centos-7.9
```

```
curl --output artifacts/containerd-1.4.13-d2iq.1-"$CONTAINERD_OS"-x86_64.tar.gz
--location https://packages.d2iq.com/dkp/containerd/containerd-1.4.13-d2iq.1-"$
CONTAINERD_OS"-x86_64.tar.gz
```

328 https://d2iq.atlassian.net/wiki/spaces/DINS

To get the fips builds append `_fips` after `-x86_64` in the url. To get the fips build for `centos-7.9` the url would be `https://packages.d2iq.com/dkp/containerd/containerd-1.4.13-d2iq.1-centos-7.9-x86_64_fips.tar.gz`
The following OS's have containerd fips builds:

- `centos-7.9`
- `ol-7.9`
- `rhel-7.9`
- `rhel-8.2`
- `rhel-8.4`

6. Download the Kubernetes images:

```
curl -o kib/artifacts/images/"$VERSION"_images.tar.gz -O https://
downloads.d2iq.com/dkp/airgapped/kubernetes-images/"$VERSION"_images.tar.gz
```

Install Kommander

Follow these steps:

1. Create the configuration file(see page 351) by running `dkp install kommander --init --airgapped > install.yaml` for the air-gapped deployment. Open the `install.yaml` file and review that it looks like the following:

```
apiVersion: config.kommander.mesosphere.io/v1alpha1
kind: Installation
airgapped:
  enabled: true
```

2. In the same file, if you are installing Kommander in an AWS VPC, set the Traefik annotation to create an internal facing ELB by setting the following:

```
apps:
  traefik:
    values: |
      service:
        annotations:
          service.beta.kubernetes.io/aws-load-balancer-internal: "true"
dkp-insights-management:
  enabled: true
catalog:
  repositories:
    - name: insights-catalog-applications
      labels:
        kommander.d2iq.io/workspace-default-catalog-repository: "true"
        kommander.d2iq.io/gitapps-gitrepository-type: "dkp"
```

```
      path: ./dkp-insights-v2.3.3.tar.gz
```

3. Push the DKP Insights charts bundle:

```
dkp push chart-bundle dkp-insights-charts-bundle-v2.3.3.tar.gz
```

4. Install DKP with Insights enabled by running:

```
dkp install kommander --installer-config ./install.yaml \
--kommander-applications-repository kommander-applications-v2.3.3.tar.gz \
--charts-bundle dkp-kommander-charts-bundle-v2.3.3.tar.gz \
--charts-bundle dkp-insights-charts-bundle-v2.3.3.tar.gz
```

5. Verify your installation(see page 0).

7.3.6.4 Install air-gapped Kommander with DKP Insights and DKP Catalog Applications

Use this section to install DKP with DKP Insights and Catalog Applications.

Prerequisites

Follow these steps:

1. Download the DKP catalog application definitions:

```
curl https://downloads.d2iq.com/dkp/v2.3.3/dkp-catalog-applications-
v2.3.3.tar.gz -O
```

2. Download the DKP catalog applications(see page 441) chart bundle:

```
curl -LJ https://downloads.d2iq.com/dkp/v2.3.3/dkp-catalog-applications-charts-
bundle-v2.3.3.tar.gz | tar -xvz
```

3. Download the DKP Insights catalog:

```
curl "https://downloads.d2iq.com/dkp/v2.3.3/dkp-insights-v2.3.3.tar.gz" -O
```

4. Download the DKP Insights chart bundle:

```
curl "https://downloads.d2iq.com/dkp/v2.3.3/dkp-insights-charts-bundle-
v2.3.3.tar.gz" -O
```

5. Download the Kommander charts bundle:

```
curl "https://downloads.d2iq.com/dkp/v2.3.3/dkp-kommander-charts-bundle-
v2.3.3.tar.gz" -O && tar -xvf dkp-kommander-charts-bundle-v2.3.3.tar.gz
```

6. Download the Kommander application definitions:

```
curl -O https://downloads.d2iq.com/dkp/v2.3.3/kommander-applications-
v2.3.3.tar.gz
```

Install Kommander

Follow these steps:

1. Create the configuration file(see page 351) by running `dkp install kommander --init --airgapped > install.yaml` for the air-gapped deployment. Open the `install.yaml` file and review that it looks like the following:

```
apiVersion: config.kommander.mesosphere.io/v1alpha1
kind: Installation
airgapped:
  enabled: true
```

2. In the same file, if you are installing Kommander in an AWS VPC, set the Traefik annotation to create an internal facing ELB by setting the following:

```
apps:
  traefik:
    values: |
      service:
        annotations:
          service.beta.kubernetes.io/aws-load-balancer-internal: "true"
dkp-insights-management:
  enabled: true
catalog:
  repositories:
    - name: insights-catalog-applications
      labels:
        kommander.d2iq.io/workspace-default-catalog-repository: "true"
        kommander.d2iq.io/gitapps-gitrepository-type: "dkp"
      path: ./dkp-insights-v2.3.3.tar.gz
    - name: dkp-catalog-applications
      labels:
        kommander.d2iq.io/project-default-catalog-repository: "true"
        kommander.d2iq.io/workspace-default-catalog-repository: "true"
        kommander.d2iq.io/gitapps-gitrepository-type: "dkp"
      path: ./dkp-catalog-applications.tar.gz
```

3. Follow the steps on the Configure an Enterprise catalog(see page 353) page to configure the DKP catalog applications.
4. To install DKP in your air-gapped environment using the above configuration file, run the following command:

```
dkp install kommander --installer-config ./install.yaml \
```

```
--kommander-applications-repository kommander-applications-v2.3.3.tar.gz \
--charts-bundle dkp-kommander-charts-bundle-v2.3.3.tar.gz \
--charts-bundle dkp-catalog-applications-charts-bundle-v2.3.3.tar.gz \
--charts-bundle dkp-insights-charts-bundle-v2.3.3.tar.gz
```

5. Verify your installation(see page 0).

7.3.6.5 Useful DKP CLI Commands

Kommander Charts Bundle

The charts bundle is a gzipped tar archive containing Helm charts, which are required during Kommander installation. Create the charts bundle with the Kommander CLI or downloaded along with the DKP CLI. Execute this command to create the charts bundle:

```
dkp create chart-bundle
```

DKP creates `charts-bundle.tar.gz`. Optionally, specify the output using the `-o` parameter:

```
dkp create chart-bundle -o [name of the output file]
```

DKP Internal Helm Repository

The DKP charts bundle is pushed to DKP's internal Helm repository. To inspect the contents:

```
dkp get charts
```

Individual charts can be removed using:

```
dkp delete chart [chartName] [chartVersion]
```

It is possible to push new charts as well:

```
dkp push chart [chartTarball]
```

Or push a new bundle:

```
dkp push chart-bundle [chartsTarball]
```

Check the built-in help text for each command for more information.

This Docker image includes code from the MinIO Project ("MinIO"), which is © 2015-2021 MinIO, Inc. MinIO is made available subject to the terms and conditions of the GNU Affero General Public License 3.0. The complete source code for the versions of MinIO packaged with DKP/Kommander/Konvoy 2.3.2 are available at these URLs:

https://github.com/minio/minio/tree/RELEASE.2022-01-08T03-11-54Z

https://github.com/minio/minio/tree/RELEASE.2021-02-14T04-01-33Z

7.4 Install DKP on a small environment

You can install Kommander on a small environment with smaller memory, storage, and CPU requirements for testing and demo purposes. This topic describes methods for installing Kommander in these environments. Refer to the Kommander documentation(see page 387) for more information.

⚠ **Enterprise considerations:** D2iQ recommends performing testing and demo tasks in a single-cluster environment. The Enterprise license is designed for multi-cluster environments and fleet management, which require a minimum amount of resources(see page 381). Applying an Enterprise license key to the previous installation adds modifications to your environment that can exhaust a small environment's resources.

7.4.1 Prerequisites

Ensure you have done the following:

- You have acquired a DKP license.
- You have installed Konvoy.
- You have reviewed the prerequisite section pertaining to your air-gapped(see page 369), or networked(see page 366) environment.

7.4.2 Minimal Kommander installation

The YAML file that is used to install a minimal configuration of Kommander contains the bare minimum setup that allows you to deploy applications, and access the DKP UI. It does **NOT** include applications for cost monitoring, logging, alerting, object storage, etc.

In this YAML file you can find the lines that correspond to all platform applications which would be included in a normal Kommander setup. Applications that have `enabled` set to `false` are not taken into account during installation. If you want to test an additional application, you can enable it individually to be installed by setting `enabled` to `true` on the corresponding line in the YAML file.

For example, if you want to enable the logging stack, set `enabled` to `true` for `grafana-logging`, `grafana-loki`, `logging-operator` and `minio-operator`. Note that depending on the size of your cluster, enabling several platform applications could exhaust your cluster's resources.

⚠ Some applications depend on other applications to work properly. Refer to the dependencies documentation(see page 426) to find out which other applications you need to enable to test the target application.

1. Initialize your Kommander installation and name it `kommander_minimal.yaml`:

```
dkp install kommander --init --kubeconfig=<cluster-kubeconfig>.conf -oyaml >
kommander_minimal.yaml
```

2. Edit your `kommander_minimal.yaml` to match the following example:

```
apiVersion: config.kommander.mesosphere.io/v1alpha1
kind: Installation
apps:
  dex:
    enabled: true
  dex-k8s-authenticator:
    enabled: true
  dkp-insights-management:
    enabled: false
  fluent-bit:
    enabled: true
  gatekeeper:
    enabled: true
  gitea:
    enabled: true
  grafana-logging:
    enabled: false
  grafana-loki:
    enabled: false
  kommander:
    enabled: true
  kube-prometheus-stack:
    enabled: false
  kubefed:
    enabled: true
  kubernetes-dashboard:
    enabled: false
  kubetunnel:
    enabled: false
  logging-operator:
    enabled: false
  minio-operator:
    enabled: false
  prometheus-adapter:
    enabled: false
  reloader:
    enabled: true
  traefik:
    enabled: true
  traefik-forward-auth-mgmt:
    enabled: true
  velero:
```

```
    enabled: false
ageEncryptionSecretName: sops-age
clusterHostname: ""
```

3. Install Kommander on your cluster with the following command:

```
dkp install kommander --installer-config ./kommander_minimal.yaml --
kubeconfig=<cluster-kubeconfig>.conf
```

4. After running the install command above, HelmReleases begin to appear on the cluster.

```
kubectl -n kommander wait --for condition=Released helmreleases --all --timeout
15m
```

This will wait for each of the helm charts to reach their `Released` condition, eventually resulting in something resembling this:

```
helmrelease.helm.toolkit.fluxcd.io/cluster-observer-1234567890 condition met
helmrelease.helm.toolkit.fluxcd.io/dex condition met
helmrelease.helm.toolkit.fluxcd.io/dex-k8s-authenticator condition met
helmrelease.helm.toolkit.fluxcd.io/gitea condition met
helmrelease.helm.toolkit.fluxcd.io/karma-traefik-certs condition met
helmrelease.helm.toolkit.fluxcd.io/kommander condition met
helmrelease.helm.toolkit.fluxcd.io/kommander-appmanagement condition met
helmrelease.helm.toolkit.fluxcd.io/kube-oidc-proxy condition met
helmrelease.helm.toolkit.fluxcd.io/kubecost-traefik-certs condition met
helmrelease.helm.toolkit.fluxcd.io/kubefed condition met
helmrelease.helm.toolkit.fluxcd.io/prometheus-traefik-certs condition met
helmrelease.helm.toolkit.fluxcd.io/reloader condition met
helmrelease.helm.toolkit.fluxcd.io/traefik condition met
helmrelease.helm.toolkit.fluxcd.io/traefik-forward-auth-mgmt condition met
```

An alternative to using the `--kubeconfig=<cluster-config>` flag is to initialize the KUBECONFIG environment variable. You can do this by running `export KUBECONFIG=<cluster-config>`. Setting your KUBECONFIG (either by flag or by environment variable) ensures that Kommander is installed on the workload cluster.

When completed, you can verify your installation(see page 383). For more on exploring your cluster, view the install Kommander(see page 366) topic.

7.5 Kommander Verify Installation

Once the Konvoy cluster is built and Kommander has been installed, you will want to verify your installation of Kommander. After the CLI successfully installs the components, it will wait for all applications to be ready by default.

NOTE: If the Kommander installation fails or you wish to reconfigure applications, you can rerun the install command to retry the installation.

If you prefer the CLI to not wait for all applications to become ready, you can set the `--wait=false` flag.

If you choose not to wait via the DKP CLI, you can check the status of the installation using the following command:

```
kubectl -n kommander wait --for condition=Released helmreleases --all --timeout 15m
```

This will wait for each of the helm charts to reach their `Released` condition, eventually resulting in something resembling this:

```
helmrelease.helm.toolkit.fluxcd.io/centralized-grafana condition met
helmrelease.helm.toolkit.fluxcd.io/dex condition met
helmrelease.helm.toolkit.fluxcd.io/dex-k8s-authenticator condition met
helmrelease.helm.toolkit.fluxcd.io/fluent-bit condition met
helmrelease.helm.toolkit.fluxcd.io/gitea condition met
helmrelease.helm.toolkit.fluxcd.io/grafana-logging condition met
helmrelease.helm.toolkit.fluxcd.io/grafana-loki condition met
helmrelease.helm.toolkit.fluxcd.io/karma condition met
helmrelease.helm.toolkit.fluxcd.io/kommander condition met
helmrelease.helm.toolkit.fluxcd.io/kommander-appmanagement condition met
helmrelease.helm.toolkit.fluxcd.io/kube-prometheus-stack condition met
helmrelease.helm.toolkit.fluxcd.io/kubecost condition met
helmrelease.helm.toolkit.fluxcd.io/kubecost-thanos-traefik condition met
helmrelease.helm.toolkit.fluxcd.io/kubefed condition met
helmrelease.helm.toolkit.fluxcd.io/kubernetes-dashboard condition met
helmrelease.helm.toolkit.fluxcd.io/kubetunnel condition met
helmrelease.helm.toolkit.fluxcd.io/logging-operator condition met
helmrelease.helm.toolkit.fluxcd.io/logging-operator-logging condition met
helmrelease.helm.toolkit.fluxcd.io/prometheus-adapter condition met
helmrelease.helm.toolkit.fluxcd.io/prometheus-thanos-traefik condition met
helmrelease.helm.toolkit.fluxcd.io/reloader condition met
helmrelease.helm.toolkit.fluxcd.io/rook-ceph condition met
helmrelease.helm.toolkit.fluxcd.io/rook-ceph-cluster condition met
helmrelease.helm.toolkit.fluxcd.io/thanos condition met
helmrelease.helm.toolkit.fluxcd.io/traefik condition met
helmrelease.helm.toolkit.fluxcd.io/traefik-forward-auth-mgmt condition met
helmrelease.helm.toolkit.fluxcd.io/velero condition met
```

7.5.1 Failed HelmReleases

If any application fails to successfully deploy, you can check the status of a `HelmRelease` with:

```
kubectl -n kommander get helmrelease <HELMRELEASE_NAME>
```

If you find any `HelmReleases` in a "broken" release state such as "exhausted" or "another rollback/release in progress", you can trigger a reconciliation of the `HelmRelease` using the following commands:

```
kubectl -n kommander patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[{"op":
"replace", "path": "/spec/suspend", "value": true}]'
kubectl -n kommander patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[{"op":
"replace", "path": "/spec/suspend", "value": false}]'
```

NOTE: If the Kommander installation fails or you wish to reconfigure applications, you can rerun the install command to retry the installation.

7.6 Log in to DKP UI

7.6.1 Access the UI through Kommander

When all the `HelmReleases` are ready, open the DKP UI in your browser. This command opens the URL of the Kommander web interface in your default browser, and prints the username and password in the CLI.

```
dkp open dashboard
```

You can retrieve Kommander credentials by executing the following command to access your Username and Password stored on the cluster. You can retrieve it at any time using the command:

```
kubectl -n kommander get secret dkp-credentials -o go-template='Username:
{{.data.username|base64decode}}{{ "\n"}}Password: {{.data.password|base64decode}}
{{ "\n"}}'
```

When DKP is installed, a static user account and credentials are created that can be used to access the DKP Dashboard. You should only use these static credentials to access the DKP UI for configuring an external identity provider(see page 400). D2iQ recommends that you only use these static credentials as a backup to credentials configured using an external identity provider(see page 400).

You can perform the following operations on Identity Providers(see page 400):

- Create an Identity Provider
- Temporarily Disable an Identity Provider
- Create Groups

Always log in with your own identity from external identity providers that provide additional security features like Multi-Factor Authentication. If the need arises to change the password for these credentials, the following shell script can be used to update the secret that contains the static credentials:

```
#!/usr/bin/env bash
set -o errexit pipefail nounset
```

```
case "$OSTYPE" in darwin)
    new_password=$(head -c45 /dev/urandom | base64 --break=0)
    ;;
*)
    new_password=$(head -c45 /dev/urandom | base64 --wrap=0)
    ;;
esac

kubectl patch \
    --namespace kommander \
    secret/dkp-credentials \
    --type='json' \
    --patch="[ { \"op\" : \"replace\" , \"path\" : \"/data/password\" , \"value\" :
\"$new_password\" } ]"

printf 1>&2 "Password: %s\n" "$new_password"
```

8 Day 2 Operations

Use these sections to manage your DKP environment.

8.1 Deploy a Sample Application

8.1.1 Learn how to deploy a sample application on a DKP cluster

After you have a basic DKP cluster installed and ready to use, you might want to test operations by deploying a simple, sample application. This task is **optional** and is only intended to demonstrate the basic steps for deploying applications in a production environment. If you are configuring the DKP cluster for a production deployment, you can use this section to learn the deployment process. However, deploying applications on a production cluster typically involves more planning and custom configuration than covered in this example.

This tutorial shows how to deploy a simple application that connects to the `redis` service. The sample application used in this tutorial is a condensed form of the Kubernetes sample guestbook[329] application.

8.1.2 Before you begin

You must have a DKP cluster running(see page 103).

Before running the commands below, ensure that your `kubectl` configuration references the DKP cluster on which you want to install the application. You can do this by setting the `KUBECONFIG` environment variable to the appropriate kubeconfig file's location.

8.1.3 Deploy a sample application

1. Deploy the Redis pods and service by running the following commands:

329 https://kubernetes.io/docs/tutorials/stateless-application/guestbook/

```
kubectl apply -f https://k8s.io/examples/application/guestbook/redis-leader-deployment.yaml
kubectl apply -f https://k8s.io/examples/application/guestbook/redis-leader-service.yaml
```

2. Deploy Redis followers. The leader deployment created above is a single pod. Adding followers (or replicas) makes it highly available to meet greater traffic demands. You must then setup the guestbook application to communicate with the Redis followers to read the data. To do this, set up another service (the `redis-follower-service.yaml` below). Do this by running the following commands:

```
kubectl apply -f https://k8s.io/examples/application/guestbook/redis-follower-deployment.yaml
kubectl apply -f https://k8s.io/examples/application/guestbook/redis-follower-service.yaml
```

3. Deploy the web app frontend by running the following command:

```
kubectl apply -f https://k8s.io/examples/application/guestbook/frontend-deployment.yaml
```

4. Confirm that there are three frontend replicas running:

```
kubectl get pods -l app=guestbook -l tier=frontend
```

5. Apply the frontend Service:

```
kubectl apply -f https://k8s.io/examples/application/guestbook/frontend-service.yaml
```

6. Configure the frontend Service to use a cloud load balancer:

```
cat << EOF | kubectl apply -f -
apiVersion: v1
kind: Service
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  type: LoadBalancer
  ports:
  - port: 80
  selector:
    app: guestbook
    tier: frontend
EOF
```

7. View the frontend service via the LoadBalancer by running the following command to get the IP address for the frontend Service:

```
kubectl get service frontend
```

8. the external IP address, and load the page in your browser to view your guestbook.
   The service properties provide the name of the load balancer. You can connect to the application by accessing that load balancer address in your web browser. Because this sample deployment creates a **cloud load balancer**, you should keep in mind that creating the load balancer can take up to a few minutes. You also might experience a slight delay before it is running properly due to DNS propagation and synchronization.
9. Remove the sample application by running the following commands:

```
kubectl delete deployment -l app=redis
kubectl delete service -l app=redis
kubectl delete deployment frontend
kubectl delete service frontend
```

⚠ **WARNING:** This step is **required** because the sample deployment attaches a **cloud provider load balancer** to the DKP cluster. Therefore, you **must delete** the sample application before tearing down the cluster.

8.1.4 Related Information

- Example: Deploying PHP Guestbook application with Redis[330]

8.2 Operations

Manage your cluster and deployed applications using platform applications

After you deploy a DKP cluster and the platform applications you want to use, you are ready to begin managing cluster operations and their application workloads to optimize your organization's productivity.

In most cases, a production cluster requires additional advanced configuration tailored for your environment, ongoing maintenance, authentication and authorization, and other common activities. For example, it is important to monitor cluster activity and collect metrics to ensure application performance and response time, evaluate network traffic patterns, manage user access to services, and verify workload distribution and efficiency.

- Access Control(see page 390)
- Identity Providers(see page 400)
- Infrastructure Providers(see page 404)

330 https://kubernetes.io/docs/tutorials/stateless-application/guestbook/

8.2.1 Access Control

Centrally manage access across clusters

You can centrally define role-based authorization within DKP UI to control resource access on the management cluster and a set, or all, of the target clusters. These resources are similar to Kubernetes RBAC but with crucial differences, and they make it possible to define the roles and role bindings once, and have them federated to clusters within a given scope.

DKP UI has two conceptual groups of resources that are used to manage access control:

- Kommander Roles: control access to resources on the management cluster.
- Cluster Roles: control access to resources on all target clusters in scope.

Use these two groups of resources to manage access control within 3 levels of scope:

Context	Kommander Roles	Cluster Roles
Global	Create ClusterRoles on the management cluster.	Federates ClusterRoles on all target clusters across all workspaces.
Workspace	Create namespaced Roles on the management cluster in the workspace namespace.	Federates ClusterRoles on all target clusters in the workspace.
Project	Create namespaced Roles on the management cluster in the project namespace.	Federates namespaced Roles on all target clusters in the project in the project namespace.

The role bindings(see page 0) for each level and type create `RoleBindings` or `ClusterRoleBindings` on the clusters that apply to each category.

This approach gives you maximum flexibility over who has access to what resources, conveniently mapped to your existing identity providers' claims.

8.2.1.1 Special Limitation for Kommander Roles

In addition to granting a Kommander Role, you must also grant the appropriate DKP role to allow external users and groups into the UI. See RBAC - DKP UI Authorization(see page 393) for details about the built-in DKP roles. Here are examples of `ClusterRoleBindings` that grant an IDP group admin access to the Kommander routes:

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eng-kommander-dashboard
  labels:
    "workspaces.kommander.mesosphere.io/rbac": ""
```

```
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: dkp-kommander-admin
subjects:
  - apiGroup: rbac.authorization.k8s.io
    kind: Group
    name: oidc:engineering
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eng-dkp-routes
  labels:
    "workspaces.kommander.mesosphere.io/rbac": ""
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: dkp-admin
subjects:
  - apiGroup: rbac.authorization.k8s.io
    kind: Group
    name: oidc:engineering
```

8.2.1.2 Types of Access Control Objects

Kubernetes role-based access control can be controlled with three different object categories: Groups, Roles and Policies, as explained in more detail below.

Groups

You can map group and user claims made by your configured identity providers to Kommander groups by selecting Administration / Identity providers in the left sidebar in the global workspace level, and then selecting the **Groups** tab.

Roles

`ClusterRoles` are named collections of rules defining which verbs can be applied to which resources.

- Kommander Roles apply specifically to resources on the management cluster.
- Cluster Roles apply to target clusters within their scope at these levels:
    - Global level - this is all target clusters in all workspaces,
    - Workspace level - this is all target clusters in the workspace,
    - Project level - this is all target clusters that have been added to the project.

Propagating Workspace Roles to Projects

By default, users granted the Kommander Workspace Admin, Edit, or View roles will also be granted the equivalent Kommander Project Admin, Edit, or View role for any project created in the workspace. Other workspace roles are not automatically propagated to the equivalent role for a project in the workspace.

Each workspace has roles defined using `KommanderWorkspaceRole` resources. Automatic propagation is controlled using the annotation `"workspace.kommander.mesosphere.io/sync-to-project": "true"` on a `KommanderWorkspaceRole` resource. You can manage this only by using the CLI.

```
kubectl get kommanderworkspaceroles -n test-qznrn-6sz52
```

```
NAME                         DISPLAY NAME                        AGE
kommander-workspace-admin    Kommander Workspace Admin Role      2m18s
kommander-workspace-edit     Kommander Workspace Edit Role       2m18s
kommander-workspace-view     Kommander Workspace View Role       2m18s
```

To prevent propagation of the `kommander-workspace-view` role, remove this annotation from the `KommanderWorkspaceRole` resource.

```
kubectl annotate kommanderworkspacerole -n test-qznrn-6sz52 kommander-workspace-view
workspace.kommander.mesosphere.io/sync-to-project-
```

To enable propagation of the role, add this annotation to the relevant `KommanderWorkspaceRole` resource.

```
kubectl annotate kommanderworkspacerole -n test-qznrn-6sz52 kommander-workspace-view
workspace.kommander.mesosphere.io/sync-to-project=true
```

Special Limitation for Workspace > Project Role Inheritance

When granting users access to a workspace, you must manually grant access to the projects within that workspace. Each project is created with a set of admin/edit/view roles, and you can choose to add an additional `RoleBinding` to each group or user of the workspace for one of these project roles. Usually these are prefixed `kommander-project-(admin/edit/view)`. Here is an example `RoleBinding` that grants the Kommander Project Admin role access for the project namespace to the engineering group:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: workspace-admin-project1-admin
  namespace: my-project-namespace-xxxxx
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: kommander-project-admin-xxxxx
subjects:
  - apiGroup: rbac.authorization.k8s.io
    kind: Group
```

```
    name: oidc:engineering
```

Role Bindings

Kommander role bindings, cluster role bindings, and project role bindings bind a Kommander group to any number of roles. All groups defined in the **Groups** tab will be present at the global, workspace, or project level, and are ready for you to assign roles to them.

8.2.1.3 Related Information

- Project Role Bindings(see page 468)
- Workspace Role Bindings(see page 459)
- Kommander RBAC Tutorial(see page 393)
- RBAC - DKP UI Authorization(see page 393)
- Kubernetes RBAC Authorization[331]

8.2.1.4 Granting Access to Kubernetes and Kommander Resources

Grant access to Kommander and Kubernetes resources using RBAC

Granting Access to External Users

Users and groups from an external identity provider initially have no access to kubernetes resources. Privileges must be granted explicitly by interacting with the RBAC API. This section provides some basic examples for general usage. More information about the RBAC API can be found in the Kubernetes documentation[332].

The Basics

Kubernetes does not provide an identity database for standard users. Users and group membership must be provided by a trusted identity provider. In Kubernetes, RBAC policies are additive, which means that a subject (user, group, or service account) is denied access to a resource unless explicitly granted access by a cluster administrator. You can grant access by binding a subject to a role, which grants some level of access to one or more resources. Kubernetes ships with some default roles[333], which aid in creating broad access control policies.

For example, if you want to make `mary@example.com` a cluster administrator, bind her username to the `cluster-admin` default role as follows:

```
cat << EOF | kubectl apply -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: mary-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
```

331 https://kubernetes.io/docs/reference/access-authn-authz/rbac/
332 https://kubernetes.io/docs/reference/access-authn-authz/rbac/
333 https://kubernetes.io/docs/reference/access-authn-authz/rbac/#default-roles-and-role-bindings

```
  name: cluster-admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: mary@example.com
EOF
```

This user now has the highest level of access which can be achieved. Use the `cluster-admin` role and `system:masters` group sparingly.

Restricting a User to Namespace

A more common example would be to grant a user access to a specific namespace, by creating a RoleBinding (RoleBindings are namespaced scoped). For example, to make the user `bob@example.com` a *reader* of the `baz` namespace, bind the user to the `view` role:

```
cat << EOF | kubectl apply -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: bob-view
  namespace: baz
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: view
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: bob@example.com
EOF
```

The user can now perform non-destructive operations targeting resources in the `baz` namespace only.

Groups

If your external identity provider supports group claims, you can also bind groups to roles. To make the `devops` LDAP group administrators of the `production` namespace bind the group to the `admin` role:

```
cat << EOF | kubectl apply -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: devops-admin
  namespace: production
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
```

```
  name: admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: oidc:devops
EOF
```

One important distinction from adding users is that all external groups are prefixed with `oidc:`, so a group name becomes `oidc:devops`. This prevents collision with locally defined groups.

DKP UI Authorization

The DKP UI, and other HTTP applications protected by Kommander forward authentication, are also authorized by the Kubernetes RBAC API. In addition to Kubernetes API resources, it is possible to define rules which map to HTTP URIs and HTTP verbs. Kubernetes RBAC calls these `nonResourceURLs`, Kommander forward authentication uses these rules to grant or deny access to HTTP endpoints.

Default Roles

Roles have been created for granting access to the dashboard and select applications which expose an HTTP server through the ingress controller. The `cluster-admin` role is actually a system role that defines grants permission to all actions (verbs) on any resource; including non-resource URLs. The default dashboard user is bound to this role.

Granting user `admin` privileges on `/dkp/*` grants `admin` privileges to all sub-resources, even if bindings exist for sub-resources with less privileges.

Dashboard	Role	Path	access
*	cluster-admin	*	read, write, delete
kommander	dkp-view	/dkp/*	read
kommander	dkp-edit	/dkp/*	read, write
kommander	dkp-admin	/dkp/*	read, write, delete
kommander-dashboard	dkp-kommander-view	/dkp/kommander/dashboard/*	read
kommander-dashboard	dkp-kommander-edit	/dkp/kommander/dashboard/*	read, write
kommander-dashboard	dkp-kommander-admin	/dkp/kommander/dashboard/*	read, write, delete

Dashboard	Role	Path	access
alertmanager	dkp-kube-prometheus-stack-alertmanager-view	/dkp/alertmanager/*	read
alertmanager	dkp-kube-prometheus-stack-alertmanager-edit	/dkp/alertmanager/*	read, write
alertmanager	dkp-kube-prometheus-stack-alertmanager-admin	/dkp/alertmanager/*	read, write, delete
centralized-grafana	dkp-centralized-grafana-grafana-view	/dkp/kommander/monitoring/grafana/*	read
centralized-grafana	dkp-centralized-grafana-grafana-edit	/dkp/kommander/monitoring/grafana/*	read, write
centralized-grafana	dkp-centralized-grafana-grafana-admin	/dkp/kommander/monitoring/grafana/*	read, write, delete
centralized-kubecost	dkp-centralized-kubecost-view	/dkp/kommander/kubecost/*	read
centralized-kubecost	dkp-centralized-kubecost-edit	/dkp/kommander/kubecost/*	read, write
centralized-kubecost	dkp-centralized-kubecost-admin	/dkp/kommander/kubecost/*	read, write, delete
grafana	dkp-kube-prometheus-stack-grafana-view	/dkp/grafana/*	read
grafana	dkp-kube-prometheus-stack-grafana-edit	/dkp/grafana/*	read, write
grafana	dkp-kube-prometheus-stack-grafana-admin	/dkp/grafana/*	read, write, delete
grafana-logging	dkp-grafana-logging-view	/dkp/logging/grafana/*	read
grafana-logging	dkp-grafana-logging-edit	/dkp/logging/grafana/*	read, write

Dashboard	Role	Path	access
grafana-logging	dkp-grafana-logging-admin	/dkp/logging/grafana/*	read, write, delete
karma	dkp-karma-view	/dkp/kommander/monitoring/karma/*	read
karma	dkp-karma-edit	/dkp/kommander/monitoring/karma/*	read, write
karma	dkp-karma-admin	/dkp/kommander/monitoring/karma/*	read, write, delete
kubernetes-dashboard	dkp-kubernetes-dashboard-view	/dkp/kubernetes/*	read
kubernetes-dashboard	dkp-kubernetes-dashboard-edit	/dkp/kubernetes/*	read, write
kubernetes-dashboard	dkp-kubernetes-dashboard-admin	/dkp/kubernetes/*	read, write, delete
prometheus	dkp-kube-prometheus-stack-prometheus-view	/dkp/prometheus/*	read
prometheus	dkp-kube-prometheus-stack-prometheus-edit	/dkp/prometheus/*	read, write
prometheus	dkp-kube-prometheus-stack-prometheus-admin	/dkp/prometheus/*	read, write, edit
traefik	dkp-traefik-view	/dkp/traefik/*	read
traefik	dkp-traefik-edit	/dkp/traefik/*	read, edit
traefik	dkp-traefik-admin	/dkp/traefik/*	read, edit, delete
thanos	dkp-thanos-query-view	/dkp/kommander/monitoring/query/*	read
thanos	dkp-thanos-query-edit	/dkp/kommander/monitoring/query/*	read, write

Dashboard	Role	Path	access
thanos	dkp-thanos-query-admin	/dkp/kommander/monitoring/query/*	read, write, delete

This section provides a few examples of binding subjects to the default roles defined for the DKP UI endpoints.

Examples

User

To grant the user `mary@example.com` administrative access to all Kommander resources, bind the user to the `dkp-admin` role:

```
cat << EOF | kubectl apply -f -
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: dkp-admin-mary
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: dkp-admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: mary@example.com
EOF
```

If you inspect the role, you see what access is now granted:

```
kubectl describe clusterroles dkp-admin
```

```
Name:         dkp-admin
Labels:       app.kubernetes.io/instance=kommander
              app.kubernetes.io/managed-by=Helm
              app.kubernetes.io/version=v2.0.0
              helm.toolkit.fluxcd.io/name=kommander
              helm.toolkit.fluxcd.io/namespace=kommander
              rbac.authorization.k8s.io/aggregate-to-admin=true
Annotations:  meta.helm.sh/release-name: kommander
              meta.helm.sh/release-namespace: kommander
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
             [/dkp/*]           []              [delete]
             [/dkp]             []              [delete]
```

```
[/dkp/*]          []              [get]
[/dkp]            []              [get]
[/dkp/*]          []              [head]
[/dkp]            []              [head]
[/dkp/*]          []              [post]
[/dkp]            []              [post]
[/dkp/*]          []              [put]
[/dkp]            []              [put]
```

The user can now use the HTTP verbs HEAD, GET, DELETE, POST, and PUT when accessing any URL at or under `/dkp`. Provided the downstream application follows REST conventions, this effectively allows read, edit, and delete privileges.

NOTE: To allow users to access the DKP UI, ensure they have the appropriate `dkp-kommander-` role in addition to the Kommander roles granted in the DKP UI. For more information, see the Access Control section of the Kommander documentation(see page 390).

Group

In order to grant view access to the `/dkp/*` endpoints and edit access to the grafana logging endpoint to group `logging-ops`, create the following ClusterRoleBindings:

```
cat << EOF | kubectl apply -f -
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: dkp-view-logging-ops
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: dkp-view
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: oidc:logging-ops
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: dkp-logging-edit-logging-ops
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: dkp-logging-edit
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: oidc:logging-ops
EOF
```

Note: external groups must be prefixed by `oidc:`

Members of `logging-ops` are now able to `view` all resources under `/dkp` and edit all resources under `/dkp/logging/grafana`.

Accessing the Kubernetes Dashboard

The Kubernetes dashboard offloads authorization directly to the Kubernetes API server. Once authenticated, all users may access the dashboard at `/dkp/kubernetes/` without needing a `dkp` role. However, access to the underlying kubernetes resources exposed by the dashboard are protected by the cluster RBAC policy.

Further Reading

This page has provides some basic examples of operations which provide the building blocks of creating an access control policy. For more information about creating your own roles and advanced policies, we highly recommend reading the Kubernetes RBAC documentation[334].

8.2.2 Identity Providers

8.2.2.1 Grant access to users in your organization

DKP supports GitHub, LDAP(see page 401), SAML and standard OIDC identity providers such as Google. These identity management providers support the login and authentication process for DKP and your Kubernetes clusters. You can configure as many identity providers as you want and users can select from any method when logging in.

8.2.2.2 Prerequisites

To get started with DKP, you must:

- Log in to the DKP UI(see page 385)

Limit who has access

- The GitHub provider allows you to specify any of the organizations and teams are eligible for access.
- The LDAP(see page 401) provider allows you to configure search filters for either users or groups.
- The OIDC provider cannot limit users based on identity.
- The SAML provider allows users to log in using a single sign-on (SSO) profile.

Configure an identity provider via the DKP UI

1. From the drop down, select the **Global** workspace.
2. Select **Administration > Identity Providers**.
3. Select the **Identity Providers** tab.
4. Select **+ Add Identity Provider**.
5. Select an identity provider and complete the form field with the relevant details.
6. Select **Save** to create your Identity Provider.

334 https://kubernetes.io/docs/reference/access-authn-authz/rbac/

Temporarily disabling a provider

Select the three dot button on the Identity Providers table and select **Disable** from the drop-down menu. The provider option no longer appears on the login screen.

8.2.2.3 Groups

Access control groups are configured in the Groups tab of the Identity Providers page. See Access Control(see page 390) for an overview of groups in DKP.

8.2.2.4 External LDAP directory

How to connect your cluster to an external LDAP directory

This guide shows you how to configure your DKP cluster so that users can log in with the credentials stored in an external LDAP directory service.

Step 1: Add LDAP connector

Each LDAP directory is set up in its own specific manner, so these steps are important. The LDAP authentication mechanism can be added using the CLI or the UI in Kommander.

> The following example does not cover all possible configurations. Refer to the Dex LDAP connector reference documentation[335] for more details.

The example below configures a DKP cluster to connect to the Online LDAP Test Server[336] and for demonstration purposes, the configuration shown uses `insecureNoSSL: true`. In production, you should protect LDAP communication with a properly-configured transport layer security (TLS). When using TLS, the admin can add `insecureSkipVerify: true` to `spec.ldap` to skip server certificate verification, if needed.

Below are steps for the CLI followed by UI in the event you prefer to use it.

1. Create a YAML file (`ldap.yaml`) similar to the following:

```
apiVersion: v1
kind: Secret
metadata:
  name: ldap-password
  namespace: kommander
type: Opaque
stringData:
  password: password
---
apiVersion: dex.mesosphere.io/v1alpha1
kind: Connector
```

335 https://github.com/dexidp/dex/blob/v2.22.0/Documentation/connectors/ldap.md
336 https://www.forumsys.com/tutorials/integration-how-to/ldap/online-ldap-test-server/

```
metadata:
  name: ldap
  namespace: kommander
spec:
  enabled: true
  type: ldap
  displayName: LDAP Test
  ldap:
    host: ldap.forumsys.com:389
    insecureNoSSL: true
    bindDN: cn=read-only-admin,dc=example,dc=com
    bindSecretRef:
      name: ldap-password
    userSearch:
      baseDN: dc=example,dc=com
      filter: "(objectClass=inetOrgPerson)"
      username: uid
      idAttr: uid
      emailAttr: mail
    groupSearch:
      baseDN: dc=example,dc=com
      filter: "(objectClass=groupOfUniqueNames)"
      userMatchers:
      - userAttr: DN
        groupAttr: uniqueMember
      nameAttr: ou
```

NOTE: The value for the LDAP connector `name` parameter (here: `LDAP Test`) appears on one of the login buttons in the DKP UI. You should choose an expressive name.

2. Run the following command to deploy the LDAP connector.

```
kubectl apply -f ldap.yaml
```

UI - Deploy LDAP through Add Identity Provider screen within the UI.

Step 2: Log in

1. Visit `https://<YOUR-CLUSTER-HOST>/token` and initiate a login flow.
2. On the login page choose the `Log in with <ldap-name>` button.
3. Enter the LDAP credentials, and log in.

UI - While LDAP authentication has been enabled, additional access rights will need to be configured through the Add Identity Provider screen in the UI using the documentation for Granting Access to Kubernetes and Kommander Resources(see page 393).

Troubleshooting

It is likely that the Dex LDAP connector configuration is not quite right from the start. In that case, you need to be able to debug the problem and iterate on it. The Dex log output contains helpful error messages as indicated by the following examples.

Errors during Dex startup

If the Dex configuration fragment provided results in an invalid Dex config, Dex does not properly start up. In that case, reviewing the Dex logs will provide error details. Use the following command to retrieve the Dex logs:

```
kubectl logs -f dex-66675fcb7c-snxb8  -n kommander
```

You may see an error similar to the following:

```
error parse config file /etc/dex/cfg/config.yaml: error unmarshaling JSON: parse
connector config: illegal base64 data at input byte 0
```

Another reason for Dex not starting up correctly is that `https://<YOUR-CLUSTER-HOST>/token` throws a 5xx HTTP error response after timing out.

Errors upon login

Most problems with the Dex LDAP connector configuration become apparent only after a login attempt. A login failing from misconfiguration will result in an error page showing only `Internal Server Error` and `Login error`. You can then usually find the root cause by reading the Dex log, as shown in the following example:

```
kubectl logs -f dex-5d55b6b94b-9pm2d -n kommander
```

You can look for output similar to this example:

```
[...]
time="2019-07-29T13:03:57Z" level=error msg="Failed to login user: failed to connect:
LDAP Result Code 200 \"Network Error\": dial tcp: lookup freeipa.example.com on
10.255.0.10:53: no such host"
```

Here, the directory's DNS name was misconfigured, which should be easy to address.

A more difficult problem occurs when a login through Dex through LDAP fails because Dex was not able to find the specified user unambiguously in the directory. That could be the result of an invalid LDAP user search configuration. Here's an example error message from the Dex log:

```
time="2019-07-29T14:21:27Z" level=info msg="performing ldap search
cn=users,cn=compat,dc=demo1,dc=freeipa,dc=org sub (&(objectClass=posixAccount)
(uid=employee))"
```

```
time="2019-07-29T14:21:27Z" level=error msg="Failed to login user: ldap: filter
returned multiple (2) results: \"(&(objectClass=posixAccount)(uid=employee))\""
```

Solving problems like this requires you to review the directory structure carefully. (Directory structures can be very different between different LDAP setups.) Then you must carefully assemble a user search configuration matching the directory structure.

Notably, with some directories, it can be hard to distinguish between the cases “properly configured, and user not found” (login fails in an expected way) and “not properly configured, and therefore user not found” (login fails in an unexpected way).

Example for successful login

For comparison, here are some sample log lines issued by Dex after successful login:

```
time="2019-07-29T15:35:51Z" level=info msg="performing ldap search
cn=accounts,dc=demo1,dc=freeipa,dc=org sub (&(objectClass=posixAccount)
(uid=employee))"
time="2019-07-29T15:35:52Z" level=info msg="username \"employee\" mapped to entry
uid=employee,cn=users,cn=accounts,dc=demo1,dc=freeipa,dc=org"
time="2019-07-29T15:35:52Z" level=info msg="login successful: connector \"ldap\",
username=\"\", email=\"employee@demo1.freeipa.org\", groups=[]"
```

8.2.3 Infrastructure Providers

ENTERPRISE

Managing infrastructure providers used by Kommander

Infrastructure providers, like AWS, provide the infrastructure for your Konvoy clusters. To automate their provisioning, Kommander needs authentication keys for your preferred infrastructure provider. You may have many accounts for a single infrastructure provider.

To provision new clusters and manage them, Kommander needs infrastructure provider credentials. Currently, AWS is supported.

Infrastructure provider credentials are configured in each workspace. The name you assign must be unique across all namespaces in your cluster.

8.2.3.1 View and Modify Infrastructure Providers

1. From the top menu bar, select your target workspace.
2. Select **Infrastructure Providers** in the **Administration** section of the sidebar menu.

8.2.3.2 AWS

- Configure AWS Provider with Role Credentials(see page 405) (Recommended if using AWS)
- Configure AWS Provider with Static Credentials(see page 411)

8.2.3.3 Delete an infrastructure provider

Before deleting an infrastructure provider, Kommander verifies if any existing managed clusters were created using this provider. The infrastructure provider cannot be deleted until all clusters created with the infrastructure provider have been deleted. This ensures Kommander has access to your infrastructure provider to remove all resources created for a managed cluster.

8.2.3.4 Configure an AWS Provider with a User Role

ENTERPRISE

eConfigure your provider to add resources to your AWS account

We highly recommend using the role-based method as this is more secure.

The role authentication method can only be used if your management cluster is running in AWS.

For more flexible credential configuration, we offer a role-based authentication[337] method with an optional External ID for third party access.

Create a Role Manually

The role should grant permissions to create the following resources in the AWS account:

- EC2 Instances
- VPC
- Subnets
- Elastic Load Balancer (ELB)
- Internet Gateway
- NAT Gateway
- Elastic Block Storage (EBS) Volumes
- Security Groups
- Route Tables
- IAM Roles

The user you delegate from your role must have a minimum set of permissions. Below is the minimal IAM policy required:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:AllocateAddress",
        "ec2:AssociateRouteTable",
        "ec2:AttachInternetGateway",
```

337 https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html

```
"ec2:AuthorizeSecurityGroupIngress",
"ec2:CreateInternetGateway",
"ec2:CreateNatGateway",
"ec2:CreateRoute",
"ec2:CreateRouteTable",
"ec2:CreateSecurityGroup",
"ec2:CreateSubnet",
"ec2:CreateTags",
"ec2:CreateVpc",
"ec2:ModifyVpcAttribute",
"ec2:DeleteInternetGateway",
"ec2:DeleteNatGateway",
"ec2:DeleteRouteTable",
"ec2:DeleteSecurityGroup",
"ec2:DeleteSubnet",
"ec2:DeleteTags",
"ec2:DeleteVpc",
"ec2:DescribeAccountAttributes",
"ec2:DescribeAddresses",
"ec2:DescribeAvailabilityZones",
"ec2:DescribeInstances",
"ec2:DescribeInternetGateways",
"ec2:DescribeImages",
"ec2:DescribeNatGateways",
"ec2:DescribeNetworkInterfaces",
"ec2:DescribeNetworkInterfaceAttribute",
"ec2:DescribeRouteTables",
"ec2:DescribeSecurityGroups",
"ec2:DescribeSubnets",
"ec2:DescribeVpcs",
"ec2:DescribeVpcAttribute",
"ec2:DescribeVolumes",
"ec2:DetachInternetGateway",
"ec2:DisassociateRouteTable",
"ec2:DisassociateAddress",
"ec2:ModifyInstanceAttribute",
"ec2:ModifyNetworkInterfaceAttribute",
"ec2:ModifySubnetAttribute",
"ec2:ReleaseAddress",
"ec2:RevokeSecurityGroupIngress",
"ec2:RunInstances",
"ec2:TerminateInstances",
"tag:GetResources",
"elasticloadbalancing:AddTags",
"elasticloadbalancing:CreateLoadBalancer",
"elasticloadbalancing:ConfigureHealthCheck",
"elasticloadbalancing:DeleteLoadBalancer",
"elasticloadbalancing:DescribeLoadBalancers",
"elasticloadbalancing:DescribeLoadBalancerAttributes",
"elasticloadbalancing:ApplySecurityGroupsToLoadBalancer",
"elasticloadbalancing:DescribeTags",
"elasticloadbalancing:ModifyLoadBalancerAttributes",
"elasticloadbalancing:RegisterInstancesWithLoadBalancer",
```

```
        "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
        "elasticloadbalancing:RemoveTags",
        "autoscaling:DescribeAutoScalingGroups",
        "autoscaling:DescribeInstanceRefreshes",
        "ec2:CreateLaunchTemplate",
        "ec2:CreateLaunchTemplateVersion",
        "ec2:DescribeLaunchTemplates",
        "ec2:DescribeLaunchTemplateVersions",
        "ec2:DeleteLaunchTemplate",
        "ec2:DeleteLaunchTemplateVersions",
        "ec2:DescribeKeyPairs"
      ],
      "Resource": ["*"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:UpdateAutoScalingGroup",
        "autoscaling:CreateOrUpdateTags",
        "autoscaling:StartInstanceRefresh",
        "autoscaling:DeleteAutoScalingGroup",
        "autoscaling:DeleteTags"
      ],
      "Resource": [
        "arn:*:autoscaling:*:*:autoScalingGroup:*:autoScalingGroupName/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": ["iam:CreateServiceLinkedRole"],
      "Resource": [
        "arn:*:iam::*:role/aws-service-role/autoscaling.amazonaws.com/
AWSServiceRoleForAutoScaling"
      ],
      "Condition": {
        "StringLike": { "iam:AWSServiceName": "autoscaling.amazonaws.com" }
      }
    },
    {
      "Effect": "Allow",
      "Action": ["iam:CreateServiceLinkedRole"],
      "Resource": [
        "arn:*:iam::*:role/aws-service-role/elasticloadbalancing.amazonaws.com/
AWSServiceRoleForElasticLoadBalancing"
      ],
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "elasticloadbalancing.amazonaws.com"
        }
      }
    },
    {
```

```
      "Effect": "Allow",
      "Action": ["iam:CreateServiceLinkedRole"],
      "Resource": [
        "arn:*:iam::*:role/aws-service-role/spot.amazonaws.com/
AWSServiceRoleForEC2Spot"
      ],
      "Condition": {
        "StringLike": { "iam:AWSServiceName": "spot.amazonaws.com" }
      }
    },
    {
      "Effect": "Allow",
      "Action": ["iam:PassRole"],
      "Resource": ["arn:*:iam::*:role/*.cluster-api-provider-aws.sigs.k8s.io"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:CreateSecret",
        "secretsmanager:DeleteSecret",
        "secretsmanager:TagResource"
      ],
      "Resource": ["arn:*:secretsmanager:*:*:secret:aws.cluster.x-k8s.io/*"]
    },
    {
      "Effect": "Allow",
      "Action": ["ssm:GetParameter"],
      "Resource": ["arn:*:ssm:*:*:parameter/aws/service/eks/optimized-ami/*"]
    },
    {
      "Effect": "Allow",
      "Action": ["iam:CreateServiceLinkedRole"],
      "Resource": [
        "arn:*:iam::*:role/aws-service-role/eks.amazonaws.com/
AWSServiceRoleForAmazonEKS"
      ],
      "Condition": {
        "StringLike": { "iam:AWSServiceName": "eks.amazonaws.com" }
      }
    },
    {
      "Effect": "Allow",
      "Action": ["iam:CreateServiceLinkedRole"],
      "Resource": [
        "arn:*:iam::*:role/aws-service-role/eks-nodegroup.amazonaws.com/
AWSServiceRoleForAmazonEKSNodegroup"
      ],
      "Condition": {
        "StringLike": { "iam:AWSServiceName": "eks-nodegroup.amazonaws.com" }
      }
    },
    {
      "Effect": "Allow",
```

```
      "Action": ["iam:CreateServiceLinkedRole"],
      "Resource": [
        "arn:aws:iam::*:role/aws-service-role/eks-fargate-pods.amazonaws.com/
AWSServiceRoleForAmazonEKSForFargate"
      ],
      "Condition": {
        "StringLike": { "iam:AWSServiceName": "eks-fargate.amazonaws.com" }
      }
    },
    {
      "Effect": "Allow",
      "Action": ["iam:GetRole", "iam:ListAttachedRolePolicies"],
      "Resource": ["arn:*:iam::*:role/*"]
    },
    {
      "Effect": "Allow",
      "Action": ["iam:GetPolicy"],
      "Resource": ["arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "eks:DescribeCluster",
        "eks:ListClusters",
        "eks:CreateCluster",
        "eks:TagResource",
        "eks:UpdateClusterVersion",
        "eks:DeleteCluster",
        "eks:UpdateClusterConfig",
        "eks:UntagResource",
        "eks:UpdateNodegroupVersion",
        "eks:DescribeNodegroup",
        "eks:DeleteNodegroup",
        "eks:UpdateNodegroupConfig",
        "eks:CreateNodegroup",
        "eks:AssociateEncryptionConfig"
      ],
      "Resource": ["arn:*:eks:*:*:cluster/*", "arn:*:eks:*:*:nodegroup/*/*/*"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "eks:ListAddons",
        "eks:CreateAddon",
        "eks:DescribeAddonVersions",
        "eks:DescribeAddon",
        "eks:DeleteAddon",
        "eks:UpdateAddon",
        "eks:TagResource",
        "eks:DescribeFargateProfile",
        "eks:CreateFargateProfile",
        "eks:DeleteFargateProfile"
      ],
```

```
      "Resource": ["*"]
    },
    {
      "Effect": "Allow",
      "Action": ["iam:PassRole"],
      "Resource": ["*"],
      "Condition": {
        "StringEquals": { "iam:PassedToService": "eks.amazonaws.com" }
      }
    },
    {
      "Effect": "Allow",
      "Action": ["kms:CreateGrant", "kms:DescribeKey"],
      "Resource": ["*"],
      "Condition": {
        "ForAnyValue:StringLike": {
          "kms:ResourceAliases": "alias/cluster-api-provider-aws-*"
        }
      }
    }
  ]
}
```

Make sure to also add a correct trust relationship[338] to the created role. This example allows everyone within the same account to `AssumeRole` with the created role. `YOURACCOUNTRESTRICTION` must be replaced with the AWS Account ID you would like to `AssumeRole` from.

IMPORTANT: Never add a `*/` wildcard. This opens your account to the whole world.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com",
        "AWS": "arn:aws:iam::YOURACCOUNTRESTRICTION:root"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

To use the role created, attach the following policy to the role which is already attached to your Kommander cluster. Replace `YOURACCOUNTRESTRICTION` with the AWS Account ID where the role you like to `AssumeRole` is saved. Also, replace `THEROLEYOUCREATED` with the AWS Role name.

338 https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AssumeRoleKommander",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::YOURACCOUNTRESTRICTION:role/THEROLEYOUCREATED"
    }
  ]
}
```

Create Infrastructure Provider in the UI

1. From the top menu bar, select your target workspace.
2. Select **Infrastructure Providers** in the **Administration** section of the sidebar menu.
3. Select the **Add Infrastructure Provider** button.
4. Select the **Amazon Web Services (AWS)** option.
5. Ensure **Role** is selected as the **Authentication Method**.
6. Enter a name for your infrastructure provider. Select a name that matches the AWS user.
7. Enter the **Role ARN**.
8. You can add an **External ID** if you share the Role with a 3rd party. External IDs secure your environment from accidentally used roles. Read more about External IDs[339].
9. Select **Save** to save your provider.

8.2.3.5 AWS Static Credentials

ENTERPRISE

Configuring an AWS Infrastructure Provider with static credentials

Configure an AWS Infrastructure Provider with Static Credentials

When configuring an infrastructure provider with static credentials, you need an access id and secret key for a user with a set of minimum capabilities.

Create a new User via CLI commands

You will need to have the AWS CLI utility installed[340]. Create a new user via the AWS CLI commands below:

```
aws iam create-user --user-name Kommander
```

```
aws iam create-policy --policy-name kommander-policy --policy-document
'{"Version":"2012-10-17","Statement":[{"Effect":"Allow","Action":
["ec2:AllocateAddress","ec2:AssociateRouteTable","ec2:AttachInternetGateway","ec2:AuthorizeSecurityG
roupIngress","ec2:CreateInternetGateway","ec2:CreateNatGateway","ec2:CreateRoute","ec2:CreateRouteTa
ble","ec2:CreateSecurityGroup","ec2:CreateSubnet","ec2:CreateTags","ec2:CreateVpc","ec2:ModifyVpcAtt
ribute","ec2:DeleteInternetGateway","ec2:DeleteNatGateway","ec2:DeleteRouteTable","ec2:DeleteSecurit
```

339 https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create_for-user_externalid.html
340 https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html

yGroup","ec2:DeleteSubnet","ec2:DeleteTags","ec2:DeleteVpc","ec2:DescribeAccountAttributes","ec2:Des
cribeAddresses","ec2:DescribeAvailabilityZones","ec2:DescribeInstances","ec2:DescribeInternetGateway
s","ec2:DescribeImages","ec2:DescribeNatGateways","ec2:DescribeNetworkInterfaces","ec2:DescribeNetwo
rkInterfaceAttribute","ec2:DescribeRouteTables","ec2:DescribeSecurityGroups","ec2:DescribeSubnets","
ec2:DescribeVpcs","ec2:DescribeVpcAttribute","ec2:DescribeVolumes","ec2:DetachInternetGateway","ec2:
DisassociateRouteTable","ec2:DisassociateAddress","ec2:ModifyInstanceAttribute","ec2:ModifyNetworkIn
terfaceAttribute","ec2:ModifySubnetAttribute","ec2:ReleaseAddress","ec2:RevokeSecurityGroupIngress",
"ec2:RunInstances","ec2:TerminateInstances","tag:GetResources","elasticloadbalancing:AddTags","elast
icloadbalancing:CreateLoadBalancer","elasticloadbalancing:ConfigureHealthCheck","elasticloadbalancin
g:DeleteLoadBalancer","elasticloadbalancing:DescribeLoadBalancers","elasticloadbalancing:DescribeLoa
dBalancerAttributes","elasticloadbalancing:ApplySecurityGroupsToLoadBalancer","elasticloadbalancing:
DescribeTags","elasticloadbalancing:ModifyLoadBalancerAttributes","elasticloadbalancing:RegisterInst
ancesWithLoadBalancer","elasticloadbalancing:DeregisterInstancesFromLoadBalancer","elasticloadbalanc
ing:RemoveTags","autoscaling:DescribeAutoScalingGroups","autoscaling:DescribeInstanceRefreshes","ec2
:CreateLaunchTemplate","ec2:CreateLaunchTemplateVersion","ec2:DescribeLaunchTemplates","ec2:Describe
LaunchTemplateVersions","ec2:DeleteLaunchTemplate","ec2:DeleteLaunchTemplateVersions","ec2:DescribeK
eyPairs"],"Resource":["*"]},{"Effect":"Allow","Action":
["autoscaling:CreateAutoScalingGroup","autoscaling:UpdateAutoScalingGroup","autoscaling:CreateOrUpda
teTags","autoscaling:StartInstanceRefresh","autoscaling:DeleteAutoScalingGroup","autoscaling:DeleteT
ags"],"Resource":["arn:*:autoscaling:*:*:autoScalingGroup:*:autoScalingGroupName/*"]},
{"Effect":"Allow","Action":["iam:CreateServiceLinkedRole"],"Resource":["arn:*:iam::*:role/aws-
service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"],"Condition":{"StringLike":
{"iam:AWSServiceName":"autoscaling.amazonaws.com"}}},{"Effect":"Allow","Action":
["iam:CreateServiceLinkedRole"],"Resource":["arn:*:iam::*:role/aws-service-role/
elasticloadbalancing.amazonaws.com/AWSServiceRoleForElasticLoadBalancing"],"Condition":
{"StringLike":{"iam:AWSServiceName":"elasticloadbalancing.amazonaws.com"}}},
{"Effect":"Allow","Action":["iam:CreateServiceLinkedRole"],"Resource":["arn:*:iam::*:role/aws-
service-role/spot.amazonaws.com/AWSServiceRoleForEC2Spot"],"Condition":{"StringLike":
{"iam:AWSServiceName":"spot.amazonaws.com"}}},{"Effect":"Allow","Action":
["iam:PassRole"],"Resource":["arn:*:iam::*:role/*.cluster-api-provider-aws.sigs.k8s.io"]},
{"Effect":"Allow","Action":
["secretsmanager:CreateSecret","secretsmanager:DeleteSecret","secretsmanager:TagResource"],"Resource
":["arn:*:secretsmanager:*:*:secret:aws.cluster.x-k8s.io/*"]},{"Effect":"Allow","Action":
["ssm:GetParameter"],"Resource":["arn:*:ssm:*:*:parameter/aws/service/eks/optimized-ami/*"]},
{"Effect":"Allow","Action":["iam:CreateServiceLinkedRole"],"Resource":["arn:*:iam::*:role/aws-
service-role/eks.amazonaws.com/AWSServiceRoleForAmazonEKS"],"Condition":{"StringLike":
{"iam:AWSServiceName":"eks.amazonaws.com"}}},{"Effect":"Allow","Action":
["iam:CreateServiceLinkedRole"],"Resource":["arn:*:iam::*:role/aws-service-role/eks-
nodegroup.amazonaws.com/AWSServiceRoleForAmazonEKSNodegroup"],"Condition":{"StringLike":
{"iam:AWSServiceName":"eks-nodegroup.amazonaws.com"}}},{"Effect":"Allow","Action":
["iam:CreateServiceLinkedRole"],"Resource":["arn:aws:iam::*:role/aws-service-role/eks-fargate-
pods.amazonaws.com/AWSServiceRoleForAmazonEKSForFargate"],"Condition":{"StringLike":
{"iam:AWSServiceName":"eks-fargate.amazonaws.com"}}},{"Effect":"Allow","Action":
["iam:GetRole","iam:ListAttachedRolePolicies"],"Resource":["arn:*:iam::*:role/*"]},
{"Effect":"Allow","Action":["iam:GetPolicy"],"Resource":["arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy"]},{"Effect":"Allow","Action":
["eks:DescribeCluster","eks:ListClusters","eks:CreateCluster","eks:TagResource","eks:UpdateClusterVe
rsion","eks:DeleteCluster","eks:UpdateClusterConfig","eks:UntagResource","eks:UpdateNodegroupVersion
","eks:DescribeNodegroup","eks:DeleteNodegroup","eks:UpdateNodegroupConfig","eks:CreateNodegroup","e
ks:AssociateEncryptionConfig"],"Resource":["arn:*:eks:*:*:cluster/*","arn:*:eks:*:*:nodegroup/*/*/
*"]},{"Effect":"Allow","Action":
["eks:ListAddons","eks:CreateAddon","eks:DescribeAddonVersions","eks:DescribeAddon","eks:DeleteAddon
","eks:UpdateAddon","eks:TagResource","eks:DescribeFargateProfile","eks:CreateFargateProfile","eks:D
eleteFargateProfile"],"Resource":["*"]},{"Effect":"Allow","Action":["iam:PassRole"],"Resource":
["*"],"Condition":{"StringEquals":{"iam:PassedToService":"eks.amazonaws.com"}}},
{"Effect":"Allow","Action":["kms:CreateGrant","kms:DescribeKey"],"Resource":["*"],"Condition":
{"ForAnyValue:StringLike":{"kms:ResourceAliases":"alias/cluster-api-provider-aws-*"}}}]}'

```
aws iam attach-user-policy --user-name Kommander --policy-arn $(aws iam list-policies
--query 'Policies[?PolicyName==`kommander-policy`].Arn' | grep -o '".*"' | tr -d '"')
```

```
aws iam create-access-key --user-name Kommander
```

Using an existing user

You can use an existing AWS user with credentials configured[341]. The user must be authorized to create the following resources in the AWS account:

- EC2 Instances
- VPC
- Subnets
- Elastic Load Balancer (ELB)
- Internet Gateway
- NAT Gateway
- Elastic Block Storage (EBS) Volumes
- Security Groups
- Route Tables
- IAM Roles

Below is the minimal IAM policy required:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:AllocateAddress",
        "ec2:AssociateRouteTable",
        "ec2:AttachInternetGateway",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CreateInternetGateway",
        "ec2:CreateNatGateway",
        "ec2:CreateRoute",
        "ec2:CreateRouteTable",
        "ec2:CreateSecurityGroup",
        "ec2:CreateSubnet",
        "ec2:CreateTags",
        "ec2:CreateVpc",
        "ec2:ModifyVpcAttribute",
        "ec2:DeleteInternetGateway",
        "ec2:DeleteNatGateway",
        "ec2:DeleteRouteTable",
        "ec2:DeleteSecurityGroup",
        "ec2:DeleteSubnet",
        "ec2:DeleteTags",
        "ec2:DeleteVpc",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeAddresses",
        "ec2:DescribeAvailabilityZones",
```

341 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-profiles.html

```
      "ec2:DescribeInstances",
      "ec2:DescribeInternetGateways",
      "ec2:DescribeImages",
      "ec2:DescribeNatGateways",
      "ec2:DescribeNetworkInterfaces",
      "ec2:DescribeNetworkInterfaceAttribute",
      "ec2:DescribeRouteTables",
      "ec2:DescribeSecurityGroups",
      "ec2:DescribeSubnets",
      "ec2:DescribeVpcs",
      "ec2:DescribeVpcAttribute",
      "ec2:DescribeVolumes",
      "ec2:DetachInternetGateway",
      "ec2:DisassociateRouteTable",
      "ec2:DisassociateAddress",
      "ec2:ModifyInstanceAttribute",
      "ec2:ModifyNetworkInterfaceAttribute",
      "ec2:ModifySubnetAttribute",
      "ec2:ReleaseAddress",
      "ec2:RevokeSecurityGroupIngress",
      "ec2:RunInstances",
      "ec2:TerminateInstances",
      "tag:GetResources",
      "elasticloadbalancing:AddTags",
      "elasticloadbalancing:CreateLoadBalancer",
      "elasticloadbalancing:ConfigureHealthCheck",
      "elasticloadbalancing:DeleteLoadBalancer",
      "elasticloadbalancing:DescribeLoadBalancers",
      "elasticloadbalancing:DescribeLoadBalancerAttributes",
      "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer",
      "elasticloadbalancing:DescribeTags",
      "elasticloadbalancing:ModifyLoadBalancerAttributes",
      "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
      "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
      "elasticloadbalancing:RemoveTags",
      "autoscaling:DescribeAutoScalingGroups",
      "autoscaling:DescribeInstanceRefreshes",
      "ec2:CreateLaunchTemplate",
      "ec2:CreateLaunchTemplateVersion",
      "ec2:DescribeLaunchTemplates",
      "ec2:DescribeLaunchTemplateVersions",
      "ec2:DeleteLaunchTemplate",
      "ec2:DeleteLaunchTemplateVersions",
      "ec2:DescribeKeyPairs"
    ],
    "Resource": ["*"]
  },
  {
    "Effect": "Allow",
    "Action": [
      "autoscaling:CreateAutoScalingGroup",
      "autoscaling:UpdateAutoScalingGroup",
```

```
        "autoscaling:CreateOrUpdateTags",
        "autoscaling:StartInstanceRefresh",
        "autoscaling:DeleteAutoScalingGroup",
        "autoscaling:DeleteTags"
      ],
      "Resource": [
        "arn:*:autoscaling:*:*:autoScalingGroup:*:autoScalingGroupName/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": ["iam:CreateServiceLinkedRole"],
      "Resource": [
        "arn:*:iam::*:role/aws-service-role/autoscaling.amazonaws.com/
AWSServiceRoleForAutoScaling"
      ],
      "Condition": {
        "StringLike": { "iam:AWSServiceName": "autoscaling.amazonaws.com" }
      }
    },
    {
      "Effect": "Allow",
      "Action": ["iam:CreateServiceLinkedRole"],
      "Resource": [
        "arn:*:iam::*:role/aws-service-role/elasticloadbalancing.amazonaws.com/
AWSServiceRoleForElasticLoadBalancing"
      ],
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "elasticloadbalancing.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": ["iam:CreateServiceLinkedRole"],
      "Resource": [
        "arn:*:iam::*:role/aws-service-role/spot.amazonaws.com/
AWSServiceRoleForEC2Spot"
      ],
      "Condition": {
        "StringLike": { "iam:AWSServiceName": "spot.amazonaws.com" }
      }
    },
    {
      "Effect": "Allow",
      "Action": ["iam:PassRole"],
      "Resource": ["arn:*:iam::*:role/*.cluster-api-provider-aws.sigs.k8s.io"]
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "secretsmanager:CreateSecret",
        "secretsmanager:DeleteSecret",
        "secretsmanager:TagResource"
      ],
      "Resource": ["arn:*:secretsmanager:*:*:secret:aws.cluster.x-k8s.io/*"]
    },
    {
      "Effect": "Allow",
      "Action": ["ssm:GetParameter"],
      "Resource": ["arn:*:ssm:*:*:parameter/aws/service/eks/optimized-ami/*"]
    },
    {
      "Effect": "Allow",
      "Action": ["iam:CreateServiceLinkedRole"],
      "Resource": [
        "arn:*:iam::*:role/aws-service-role/eks.amazonaws.com/
AWSServiceRoleForAmazonEKS"
      ],
      "Condition": {
        "StringLike": { "iam:AWSServiceName": "eks.amazonaws.com" }
      }
    },
    {
      "Effect": "Allow",
      "Action": ["iam:CreateServiceLinkedRole"],
      "Resource": [
        "arn:*:iam::*:role/aws-service-role/eks-nodegroup.amazonaws.com/
AWSServiceRoleForAmazonEKSNodegroup"
      ],
      "Condition": {
        "StringLike": { "iam:AWSServiceName": "eks-nodegroup.amazonaws.com" }
      }
    },
    {
      "Effect": "Allow",
      "Action": ["iam:CreateServiceLinkedRole"],
      "Resource": [
        "arn:aws:iam::*:role/aws-service-role/eks-fargate-pods.amazonaws.com/
AWSServiceRoleForAmazonEKSForFargate"
      ],
      "Condition": {
        "StringLike": { "iam:AWSServiceName": "eks-fargate.amazonaws.com" }
      }
    },
    {
      "Effect": "Allow",
      "Action": ["iam:GetRole", "iam:ListAttachedRolePolicies"],
      "Resource": ["arn:*:iam::*:role/*"]
    },
    {
      "Effect": "Allow",
      "Action": ["iam:GetPolicy"],
```

```
      "Resource": ["arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "eks:DescribeCluster",
        "eks:ListClusters",
        "eks:CreateCluster",
        "eks:TagResource",
        "eks:UpdateClusterVersion",
        "eks:DeleteCluster",
        "eks:UpdateClusterConfig",
        "eks:UntagResource",
        "eks:UpdateNodegroupVersion",
        "eks:DescribeNodegroup",
        "eks:DeleteNodegroup",
        "eks:UpdateNodegroupConfig",
        "eks:CreateNodegroup",
        "eks:AssociateEncryptionConfig"
      ],
      "Resource": ["arn:*:eks:*:*:cluster/*", "arn:*:eks:*:*:nodegroup/*/*/*"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "eks:ListAddons",
        "eks:CreateAddon",
        "eks:DescribeAddonVersions",
        "eks:DescribeAddon",
        "eks:DeleteAddon",
        "eks:UpdateAddon",
        "eks:TagResource",
        "eks:DescribeFargateProfile",
        "eks:CreateFargateProfile",
        "eks:DeleteFargateProfile"
      ],
      "Resource": ["*"]
    },
    {
      "Effect": "Allow",
      "Action": ["iam:PassRole"],
      "Resource": ["*"],
      "Condition": {
        "StringEquals": { "iam:PassedToService": "eks.amazonaws.com" }
      }
    },
    {
      "Effect": "Allow",
      "Action": ["kms:CreateGrant", "kms:DescribeKey"],
      "Resource": ["*"],
      "Condition": {
        "ForAnyValue:StringLike": {
```

```
          "kms:ResourceAliases": "alias/cluster-api-provider-aws-*"
        }
      }
    }
  ]
}
```

Fill out the Add Infrastructure Provider form in the UI

1. In Kommander, select the Workspace associated with the credentials you are adding.
2. Navigate to **Administration > Infrastructure Providers** and click the **Add Infrastructure Provider** button.
3. Select the Amazon Web Services (AWS) option.
4. Ensure **Static** is selected as the Authentication Method.
5. Enter a name for your infrastructure provider for later reference. Consider choosing a name that matches the AWS user.
6. Fill out the access and secret keys using the keys generated above.
7. Select **Save** to save your provider.

8.3 Applications

This section includes information on the applications you can deploy in DKP.

- AppDeployment resources(see page 418)
- Platform Applications(see page 421)

8.3.1 AppDeployment resources

Use AppDeployments to deploy, and customize platform, DKP catalog, and custom applications in your environment

An `AppDeployment` is a Custom Resource[342] created by DKP with the purpose of deploying applications (platform, DKP catalog and custom applications) in the management cluster, managed clusters, or both. Customers of both Essential and Enterprise products use `AppDeployments`, regardless of their setup (networked, air-gapped, etc.), and their infrastructure provider.

When installing DKP, an `AppDeployment` resource is created for each enabled **Platform Application**. This `AppDeployment` resource references a `ClusterApp`, which then references the repository that contains a concrete declarative and preconfigured setup of an application, usually in the form of a `HelmRelease`. `ClusterApps` are cluster-scoped so that these platform applications are deployable to all workspaces or projects.

In the case of **DKP catalog** and **custom applications**, the `AppDeployment` references an `App` instead of a `ClusterApp`, which also references the repository containing the installation and deployment information. `Apps` are namespace-scoped and are meant to only be deployable to the workspace or project in which they have been created.

For example, this is the default `AppDeployment` for the Kube Prometheus Stack platform application:

342 https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/

```
apiVersion: apps.kommander.d2iq.io/v1alpha3
kind: AppDeployment
metadata:
  name: kube-prometheus-stack
  namespace: ${WORKSPACE_NAMESPACE}
spec:
  appRef:
    name: kube-prometheus-stack-34.9.3
    kind: ClusterApp
```

8.3.1.1 Customization

Prerequisites

Set the `WORKSPACE_NAMESPACE` environment variable to the name of the workspace's namespace where the cluster is attached:

```
export WORKSPACE_NAMESPACE=<your_workspace_namespace>
```

You are now able to copy the following commands without having to replace the placeholder with your workspace namespace every time you run a command.

Customize your application

If you want to customize an application, or change how a specific app is deployed, you can create a `ConfigMap` to change or add values to the information that is stored in the `HelmRelease`. Override the default configuration of an application by setting the `configOverrides` field on the `AppDeployment` to that `ConfigMap`. This overrides the configuration of the app for all clusters within the workspace.

For workspace applications, you can also enable and customize them on a per-cluster basis. Refer to the cluster-scoped configuration(see page 434) page for instructions on how to enable and customize an application per cluster in a given workspace.

This is an example of how to customize the `AppDeployment` of Kube Prometheus Stack:

1. Provide the name of a `ConfigMap` with the custom configuration in the `AppDeployment`:

```
cat <<EOF | kubectl apply -f -
apiVersion: apps.kommander.d2iq.io/v1alpha3
kind: AppDeployment
metadata:
  name: kube-prometheus-stack
  namespace: ${WORKSPACE_NAMESPACE}
spec:
  appRef:
    name: kube-prometheus-stack-34.9.3
```

```
    kind: ClusterApp
  configOverrides:
    name: kube-prometheus-stack-overrides-attached
EOF
```

2. Create the `ConfigMap` with the name provided in the previous step, which provides the custom configuration on top of the default configuration:

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: ${WORKSPACE_NAMESPACE}
  name: kube-prometheus-stack-overrides-attached
data:
  values.yaml: |
    prometheus:
      prometheusSpec:
        storageSpec:
          volumeClaimTemplate:
            spec:
              resources:
                requests:
                  storage: 150Gi
EOF
```

8.3.1.2 Print and review the current state of an `AppDeployment` resource

If you want to know how the `AppDeployment` resource is currently configured, use the commands below to print a table of the declared information. If the `AppDeployment` is configured for several clusters in a workspace, a column will display a list of the clusters.

Review all `AppDeployments` in a workspace

To review the state of the `AppDeployment` resource for a specific workspace, run the `get` command with the name of your workspace, as in this example:

```
dkp get appdeployments -w kommander-workspace
```

The output should contain a list of all your applications, here is an example:

```
NAME                          APP                              CLUSTERS
[...]
kube-oidc-proxy               kube-oidc-proxy-0.3.2            host-cluster
kube-prometheus-stack         kube-prometheus-stack-34.9.3     host-cluster
```

```
kubecost                        kubecost-0.25.3                 host-cluster
[...]
```

Review a specific `AppDeployment` of an application in a workspace

To review the state of a specific `AppDeployment` of an application, run the `get` command with the name of the application and your workspace, as in this example:

```
dkp get appdeployment kube-prometheus-stack -w kommander-workspace
```

The output should look similar to this:

```
NAME                  APP                             CLUSTERS
kube-prometheus-stack kube-prometheus-stack-34.9.3 host-cluster
```

8.3.1.3 Deployment scope

In a single-cluster environment with an **Essential license**, `AppDeployments` enable customizing any platform application.

In a multi-cluster environment with an **Enterprise license**, `AppDeployments` enable workspace-level(see page 430), project-level(see page 482), and per-cluster deployment and customization of workspace applications(see page 434).

8.3.1.4 More information

Refer to the CLI documentation(see page 687) for more information on how to `create`, or `get` an `AppDeployment`.

8.3.2 Platform Applications

How platform applications work

When attaching a cluster, DKP deploys certain platform applications on the newly attached cluster. Operators can use the DKP UI to customize which platform applications to deploy to the attached clusters in a given workspace.

Currently, the monitoring stack is deployed by default. The logging stack is not.

Review the workspace platform application resource requirements(see page 430) to ensure that the attached clusters have sufficient resources.

When deploying and upgrading applications, platform applications come as a bundle; they are tested as a single unit and you must deploy or upgrade them in a single process, for each workspace. This means all clusters in a workspace have the same set and versions of platform applications deployed.

8.3.2.1 Customize a workspace's applications

You can customize the applications that are deployed to a workspace's clusters using the DKP UI:

1. From the top menu bar, select your target workspace.
2. Select **Applications** from the sidebar menu to browse all applications that can be deployed or uninstalled.

To use the CLI to deploy or uninstall applications, see Application Deployment(see page 424)

⚠ There may be dependencies between the applications, which are listed in the Workspace Platform Application Dependencies(see page 426) documentation. Review them carefully prior to customizing to ensure that the applications are deployed successfully.

8.3.2.2 Upgrade Platform applications from the CLI

The DKP upgrade(see page 780) process deploys and upgrades Platform applications as a bundle for each cluster or workspace. For the Management Cluster or workspace, DKP upgrade handles all Platform applications; no other steps are necessary to upgrade the Platform application bundle. However, for managed or attached clusters or workspaces, you MUST manually upgrade the Platform applications bundle with the following command.

⚠ If you are upgrading your Platform applications as part of the DKP upgrade(see page 780), upgrade your Platform applications on any additional Workspaces before proceeding with the Konvoy upgrade. Some applications in the previous release are not compatible with the Kubernetes version(see page 811) of this release, and upgrading Kubernetes is part of the DKP Konvoy upgrade process.

Use this command to upgrade all platform applications in the given workspace and its projects to the same version as the platform applications running on the management cluster:

```
export WORKSPACE_NAME=<name-of-workspace>
```

```
dkp upgrade workspace ${WORKSPACE_NAME}
```

An output similar to this appears:

```
✓ Ensuring HelmReleases are upgraded on clusters in namespace "WORKSPACE_NAME"
...
```

If the upgrade fails or times out, retry the command with higher verbosity to get more information on the upgrade process:

```
dkp upgrade workspace ${WORKSPACE_NAME} -v 4
```

If you find any `HelmReleases` in a “broken” release state such as “exhausted” or “another rollback/release in progress”, you can trigger a reconciliation of the `HelmRelease` using the following commands:

```
kubectl -n <WORKSPACE_NAMESPACE> patch helmrelease <HELMRELEASE_NAME> --type='json'
 -p='[{"op": "replace", "path": "/spec/suspend", "value": true}]'
kubectl -n <WORKSPACE_NAMESPACE> patch helmrelease <HELMRELEASE_NAME> --type='json'
 -p='[{"op": "replace", "path": "/spec/suspend", "value": false}]'
```

8.3.2.3 Workspace platform applications

The following table provides a list of platform applications that are available in DKP. Some of them are deployed by default on attachment, others require manual installation(see page 424):

Common Name	APP NAME	APP ID	Deployed by default
Cert Manager	cert-manager-1.7.1	cert-manager	Yes
External DNS	external-dns-6.5.5	external-dns	No
Fluent Bit	fluent-bit-0.19.20	fluent-bit	No
Gatekeeper	gatekeeper-3.8.1	gatekeeper	Yes
Grafana	grafana-logging-6.28.0	grafana-logging	No
Loki	grafana-loki-0.48.4	grafana-loki	No
Istio	istio-1.14.1	istio	No
Jaeger	jaeger-2.32.2	jaeger	No
Kiali	kiali-1.52.0	kiali	No
Kube OIDC Proxy	kube-oidc-proxy-0.3.1	kube-oidc-proxy	Yes
Kube Prometheus Stack	kube-prometheus-stack-34.9.3	kube-prometheus-stack	Yes
Kubecost	kubecost-0.26.0	kubecost	Yes
Kubernetes Dashboard	kubernetes-dashboard-5.1.1	kubernetes-dashboard	Yes
Logging Operator	logging-operator-3.17.7	logging-operator	No

Common Name	APP NAME	APP ID	Deployed by default
Minio	minio-operator-4.4.25	minio-operator	No
Nvidia	nvidia-0.4.4	nvidia	No
Prometheus Adapter	prometheus-adapter-2.17.1	prometheus-adapter	Yes
Reloader	reloader-0.0.110	reloader	Yes
Traefik	traefik-10.9.1	traefik	Yes
Traefik ForwardAuth	traefik-forward-auth-0.3.6	traefik-forward-auth	Yes
Velero	velero-3.2.3	velero	No

- Currently, DKP only supports a single deployment of `cert-manager` per cluster. Because of this, `cert-manager` cannot be installed on any Konvoy managed clusters or clusters that come with cert-manager pre-installed.
- Only a single deployment of `traefik` per cluster is supported.
- DKP automatically manages the deployment of `traefik-forward-auth` and `kube-oidc-proxy` when clusters are attached to the workspace. These applications are not shown in the DKP UI.
- Applications are enabled in DKP and then deployed to attached clusters. To confirm that your enabled application has successfully deployed, you should verify via the CLI(see page 0).

8.3.2.4 Platform Application Deployment

Deploy platform applications to attached clusters using the CLI

This topic describes how to use the CLI to enable an application to deploy to attached clusters within a workspace.

See Workspace Platform Applications(see page 423) for a list of all applications and those that are enabled by default.

To use the DKP UI to enable applications, see Customize a workspace's applications(see page 0). To deploy an application to selected clusters within a workspace, refer to the cluster-scoped configuration(see page 434) section.

Prerequisites

Before you begin, you must have:

- A running cluster with Kommander installed(see page 83).
- An existing Kubernetes cluster attached to Kommander(see page 505).
- Determine the name of the workspace where you wish to perform the deployments. You can use the `dkp get workspaces` command to see the list of workspace names and their corresponding namespaces.
- Set the `WORKSPACE_NAMESPACE` environment variable to the name of the workspace's namespace where the cluster is attached:

```
export WORKSPACE_NAMESPACE=<workspace_namespace>
```

- Set the `WORKSPACE_NAME` environment variable to the name of the workspace where the cluster is attached:

```
export WORKSPACE_NAME=<workspace_name>
```

⚠ From the CLI, you can enable applications to deploy in the workspace. Verify that an application has successfully deployed via the CLI[343].

Create the `AppDeployment` to Enable the Application

If you have already created an AppDeployment and you only want to **edit the configuration** to customize your application, refer to the AppDeployment Customization(see page 418) instructions. If you want to enable or customize your application on **selected clusters within a workspace**, refer to the cluster-scoped configuration(see page 434) instructions.

Review the list of available applications(see page 423) that can be enabled to deploy to your cluster. Copy the app name of the application you want to deploy.

Enable a supported application to deploy to your existing attached or managed cluster with an AppDeployment resource(see page 418):

1. Run the following command and define the `--app` flag to specify which platform application will be enabled:

```
dkp create appdeployment istio --app istio-1.14.1 --workspace ${WORKSPACE_NAME}
```

- The `--app` flag must match the `APP NAME` from the list of available platform applications.
- Observe that the `dkp create` command must be run with the `WORKSPACE_NAME` instead of the `WORKSPACE_NAMESPACE` flag.

343 https://d2iq.atlassian.net/wiki/spaces/ES/pages/10387674/Deployment+of+Applications+in+Workspaces#Verify-applications

This instructs Kommander to create and deploy the `AppDeployment` to the `KommanderClusters` in the specified `WORKSPACE_NAME`.

Verify Applications

The applications are now enabled. Connect to the attached cluster and watch the `HelmReleases` to verify the deployment. In this example, we are checking if `istio` got deployed correctly:

```
kubectl get helmreleases istio -n ${WORKSPACE_NAMESPACE} -w
```

You should eventually see the `HelmRelease` marked as `Ready`:

```
NAMESPACE              NAME        READY   STATUS                              AGE
workspace-test-vjsfq   istio       True    Release reconciliation succeeded    7m3s
```

Some supported applications have dependencies on other applications. See Workspace Platform Application Dependencies(see page 426) for that table.

8.3.2.5 Platform Application Dependencies

Dependencies between workspace applications

Platform applications that are deployed to a workspace's attached clusters can depend on each other. It is important to note these dependencies when customizing the workspace platform applications to ensure that your applications are properly deployed to the clusters. For more information on how to customize workspace platform applications, see Workspace Platform Applications(see page 421).

When deploying or troubleshooting platform applications, it helps to understand how platform applications interact and may require other platform applications as dependencies.

If a platform application's dependency does not successfully deploy, the platform application requiring that dependency does not successfully deploy.

The following sections detail information about the workspace platform application.

Foundational Applications

Provides the foundation for all platform application capabilities and deployments on managed clusters. These applications must be enabled for any platform applications to work properly.

The foundational applications are comprised of the following platform application:

- cert-manager[344]: Automates TLS certificate management and issuance.
- reloader[345]: A controller that watches changes on ConfigMaps and Secrets, and automatically triggers updates on the dependent applications.

344 https://cert-manager.io/docs
345 https://github.com/stakater/Reloader

- traefik[346]: Provides an HTTP reverse proxy and load balancer. Requires cert-manager and reloader.

Platform Application	Dependencies
cert-manager	
reloader	
traefik	cert-manager, reloader

Logging

Collects logs over time from Kubernetes and applications deployed on managed clusters. Also provides the ability to visualize and query the aggregated logs.

- grafana-loki[347]: A horizontally-scalable, highly-available, multi-tenant log aggregation system inspired by Prometheus.
- grafana-logging[348]: Logging dashboard used to view logs aggregated to Grafana Loki.
- logging-operator[349]: Automates the deployment and configuration of a Kubernetes logging pipeline.
- minio-operator[350]: A Kubernetes-native high performance object store with an S3-compatible API that supports deploying MinIO Tenants onto private and public cloud infrastructures.
- fluent-bit[351]: Open source and multi-platform log processor tool which aims to be a generic Swiss knife for logs processing and distribution.

Platform Application	Dependencies
grafana-loki	minio-operator
grafana-logging	grafana-loki
logging-operator	
minio-operator	
fluent-bit	

Monitoring

Provides monitoring capabilities by collecting metrics, including cost metrics, for Kubernetes and applications deployed on managed clusters. Also provides visualization of metrics and evaluates rule expressions to trigger alerts when specific conditions are observed.

---

346 https://traefik.io/
347 https://grafana.com/oss/loki/
348 https://grafana.com/oss/grafana/
349 https://banzaicloud.com/docs/one-eye/logging-operator/
350 https://github.com/minio/operator/blob/master/README.md
351 https://docs.fluentbit.io/manual/

- kube-prometheus-stack[352]: A stack of applications that collect metrics and provide visualization and alerting capabilities.
  NOTE: Prometheus[353], Prometheus Alertmanager[354] and Grafana[355] are included in the bundled installation.
- prometheus-adapter[356]: Provides cluster metrics from Prometheus.
- kubecost[357]: provides real-time cost visibility and insights for teams using Kubernetes, helping you continuously reduce your cloud costs.
- kubernetes-dashboard[358]: A general purpose, web-based UI for Kubernetes clusters. It allows users to manage applications running in the cluster, troubleshoot them and manage the cluster itself.
- nvidia[359]: A suite of tools for managing and monitoring NVIDIA datacenter GPUs in cluster environments. Includes active health monitoring, comprehensive diagnostics, system alerts, and governance policies including power and clock management.

Platform Application	Dependencies
kube-prometheus-stack	traefik
prometheus-adapter	kube-prometheus-stack
kubecost	traefik
kubernetes-dashboard	traefik
nvidia	

Security

Allows management of security constraints and capabilities for the clusters and users.

- gatekeeper[360]: A policy Controller for Kubernetes.

Platform Application	Dependencies
gatekeeper	

Single Sign On (SSO)

Group of platform applications that allow enabling SSO on attached clusters. SSO is a centralized system for connecting attached clusters to the centralized authority on the management cluster.

---

352 https://github.com/prometheus-community/helm-charts/tree/main/charts/kube-prometheus-stack
353 https://prometheus.io/
354 https://prometheus.io/docs/alerting/latest/alertmanager
355 https://grafana.com/
356 https://github.com/DirectXMan12/k8s-prometheus-adapter
357 https://kubecost.com/
358 https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/
359 https://ngc.nvidia.com/catalog/containers/nvidia:k8s:dcgm-exporter
360 https://github.com/open-policy-agent/gatekeeper

- kube-oidc-proxy[361]: A reverse proxy server that authenticates users using OIDC to Kubernetes API servers where OIDC authentication is not available.
- traefik-forward-auth[362]: Installs a forward authentication application providing Google OAuth based authentication for Traefik.

Platform Application	Dependencies
kube-oidc-proxy	cert-manager, traefik
traefik-forward-auth	traefik

Backup

This platform application assists you with backing up and restoring your environment.

- velero[363]: An open source tool for safely backing up and restoring resources in a Kubernetes cluster, performing disaster recovery, and migrating resources and persistent volumes to another Kubernetes cluster.

Platform Application	Dependencies
velero	

Service Mesh

Allows deploying service mesh on clusters, enabling the management of microservices in cloud-native applications. Service mesh can provide a number of benefits, such as providing observability into communications, providing secure connections, or automating retries and backoff for failed requests.

- istio[364]: Addresses the challenges developers and operators face with a distributed or microservices architecture.
- kiali[365]: A management console for an Istio-based service mesh. It provides dashboards, observability, and lets you operate your mesh with robust configuration and validation capabilities.
- jaeger[366]: A distributed tracing system used for monitoring and troubleshooting microservices-based distributed systems.

Catalog Application	Dependencies
istio	kube-prometheus-stack
kiali	istio, jaeger

361 https://github.com/jetstack/kube-oidc-proxy
362 https://github.com/thomseddon/traefik-forward-auth
363 https://velero.io/
364 https://istio.io/latest/about/service-mesh/
365 https://kiali.io/
366 https://www.jaegertracing.io/

Catalog Application	Dependencies
jaeger	istio
knative	istio

Related information

- Kommander security architecture(see page 584)
- Traefik Ingress controller(see page 584)
- Logging and audits(see page 560)

8.3.2.6 Platform Application Configuration Requirements

Workspace Platform Application Descriptions and Resource Requirements

Workspace platform applications require more resources than solely deploying or attaching clusters into a workspace. Your cluster must have sufficient resources when deploying or attaching to ensure that the platform services are installed successfully.

The following table describes all the workspace platform applications that are available to the clusters in a workspace, minimum resource requirements, and whether they are enabled by default.

Name	Minimum Resources Suggested	Minimum Persistent Storage Required	Deployed by Default
cert-manager	cpu: 10m memory: 32Mi		Yes
fluentbit	cpu: 350m memory: 350Mi		No
gatekeeper	cpu: 300m memory: 768Mi		Yes
grafana-logging	cpu: 200m memory: 100Mi		No
grafana-loki		# of PVs: 8 PV sizes: 10Gi x 8 (total: 80Gi)	No
istio	cpu: 1270m memory: 4500Mi		No
jaeger			No

Name	Minimum Resources Suggested	Minimum Persistent Storage Required	Deployed by Default
kiali	cpu: 20m memory: 128Mi		No
knative	cpu: 610m memory: 400Mi		No
kube-prometheus-stack	cpu: 1210m memory: 4150Mi	# of PVs: 1 PV sizes: 100Gi	Yes
kube-oidc-proxy			Yes
kubecost	cpu: 700m memory: 1700Mi	# of PVs: 3 PV sizes: 2Gi, 32Gi, 32Gi (total: 66Gi)	Yes
kubernetes-dashboard	cpu: 250m memory: 300Mi		Yes
logging-operator	cpu: 350m * # of nodes + 600m memory: 228Mi + 350Mi * # of nodes	# of PVs: 1 PV sizes: 10Gi	No
minio-operator	cpu: 200m memory: 256Mi		No
nfs-server-provisioner		# of PVs: 1 PV sizes: 100Gi	No
nvidia	cpu: 100m memory: 128Mi		No
prometheus-adapter	cpu: 1000m memory: 1000Mi		Yes
reloader	cpu: 100m memory: 128Mi		Yes
traefik	cpu: 500m		Yes
traefik-forward-auth	cpu: 100m memory: 128Mi		Yes

Name	Minimum Resources Suggested	Minimum Persistent Storage Required	Deployed by Default
velero	cpu: 1000m memory: 1024Mi	PV sizes: 10Gi x 4 (total: 40Gi)	No

8.4 Workspaces

ENTERPRISE

Allow teams to manage their own clusters using workspaces

Workspaces give you the flexibility to represent your organization in a way that makes sense for your team and configuration. For example, you can create separate workspaces for each department, product, or business function. Each workspace manages their own clusters and role-based permissions, while retaining an overall organization-level view of all clusters in operation.

8.4.1 Global / Workspace UI

The UI is designed to be accessible for different roles at different levels:

- **Global**: At the top level, IT administrators manage all clusters across all workspaces.
- **Workspace**: DevOps administrators manage multiple clusters within a workspace.
- **Projects**: DevOps administrators or developers manage configuration and services across multiple clusters.

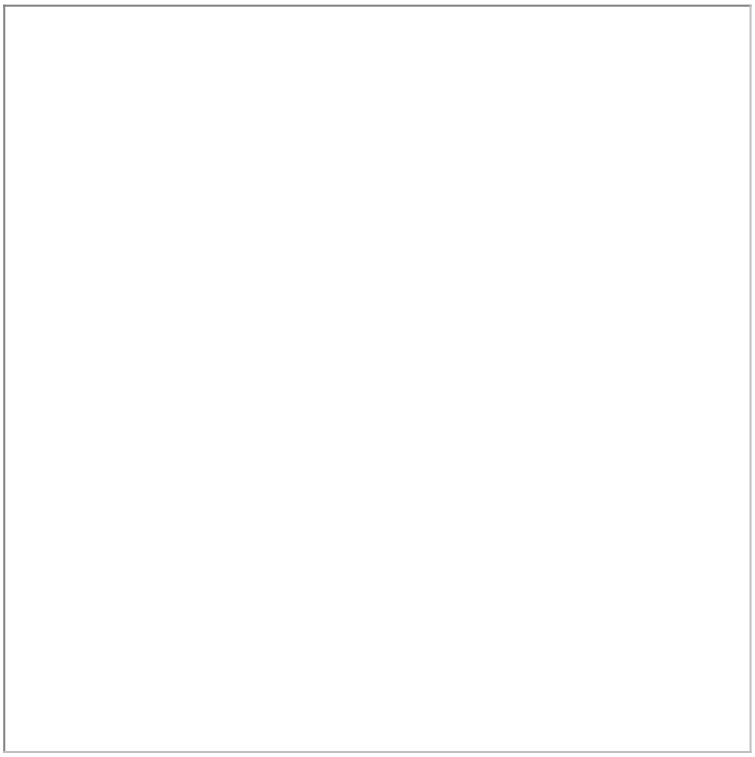
8.4.2 Default Workspace

To get started immediately, you can use the default workspace deployed in DKP. Your workspace delegation can be done at a later time.

8.4.3 Create a Workspace

In DKP you can create your own Workspaces. The following steps describe this procedure.

1. From the workspace selection dropdown in the top menu bar, select **Create Workspace**.



2. Type a name and description and select **Save**. The workspace is now accessible from the workspace selection dropdown.

8.4.4 Add, Edit, and Delete Workspace Annotations and Labels

When creating or editing a workspace, you can use the **Advanced Options** to add, edit, or delete annotations and labels to your workspace. Both the annotations and labels are applied to the workspace namespace.

1. From the top menu bar, select your target workspace.
2. Select the **Actions** dropdown button in the top-right, and select **Edit Workspace**.
3. Select the **Advanced** options.
4. Type in new **Key** and **Value** labels for your workspace or edit existing **Key** and **Value** labels.

Labels that are added to a workspace, are also applied to all of the clusters in the workspace.

8.4.5 Delete a Workspace

In DKP you can delete existing Workspaces. The following steps describe this procedure.

Workspaces can only be deleted if all the clusters in the workspace have been deleted or detached.

1. From the top menu bar, select **Global**.
2. From the sidebar menu, select **Workspaces**.
3. Select the three dot button to the right of the workspace you want to delete and then select **Delete**.
4. Confirm deleting the Workspace in the **Delete Workspace** dialog box.

The following procedures are supported for workspaces:

- Application Deployment using the CLI(see page 424)
- Platform Applications(see page 421)
- Platform Application Dependencies(see page 426)
- Workspace Platform Application Configuration Requirements(see page 430)

8.4.6 Workspace Applications

ENTERPRISE

This section documents the applications and application types that you can use with DKP.

Application types are:

- Catalog Applications(see page 441) are either pre-packaged applications from the D2iQ Application Catalog, or custom applications that you maintain for your teams or organization.
- Platform Applications(see page 421) are applications integrated into DKP.

8.4.6.1 Cluster-scoped Configuration for Existing AppDeployments

ENTERPRISE

Enable and customize an application on a per-cluster basis

This topic describes how to enable cluster-scoped configuration of applications for **existing** `AppDeployments`.

For more information on how to **create** `AppDeployments`, refer to the Application Deployment(see page 424) section.

When you enable an application for a workspace, you deploy this application to all clusters within that workspace. You can also choose to enable or customize an application on certain clusters within a workspace. This functionality allows you to use DKP in a multi-cluster scenario without restricting the management of multiple clusters from a single workspace.

Your DKP cluster comes bundled with a set of default application configurations. If you want to override the default configuration of your applications, you can define workspace `configOverrides` on top of the default workspace configuration. And if you want to further customize your workplace by enabling applications on a per-cluster basis or by defining per-cluster customizations, you can create and apply `clusterConfigOverrides`.

The cluster-scoped enablement and customization of applications is an **Enterprise-only** feature, which allows the configuration of all workspace Platform(see page 421), DKP catalog(see page 441) and Custom Applications(see page 451) via the **CLI** in your managed and attached clusters regardless of your environment setup (networked or air-gapped). This capability is not provided for project applications.

Prerequisites

- Any application you wish to enable or customize at a cluster level, first needs to be enabled at the workspace-level via an `AppDeployment` (platform application deployment(see page 424), workspace catalog application deployment(see page 447)).
- For custom configurations, you have created a ConfigMap(see page 0) with all the required `spec` fields for each customization you would like to add to an application in a cluster. You can apply a `ConfigMap` to

several clusters, or create a `ConfigMap` for each cluster, but the `ConfigMap` object must exist in the Management cluster.

- Determine the name of the workspace where you wish to perform the deployments. You can use the `dkp get workspaces` command to see the list of workspace names and their corresponding namespaces.
- Set the `WORKSPACE_NAMESPACE` environment variable to the name of the workspace's namespace where the cluster is attached:

```
export WORKSPACE_NAMESPACE=<workspace_namespace>
```

- Set the `WORKSPACE_NAME` environment variable to the name of the workspace where the cluster is attached:

```
export WORKSPACE_NAME=<workspace_name>
```

Enable or Disable an App on a Per-cluster basis and Define a Custom Configuration

If you want to:

- Enable/disable an application for certain clusters, or
- Provide a custom configuration for your enabled applications,

Edit the `AppDeployment` resource to include the new cluster-scoped configuration.

Enable an Application on a Per-cluster Basis **for the First Time**

When you enable an application on a workspace, it is deployed to all clusters in the workspace by default. If you would like to only deploy it to a subset of clusters when enabling it on a workspace for the first time, you can follow the steps below:

1. Create an `AppDeployment` for your application, selecting a subset of clusters within the workspace to enable it on. You can use the `dkp get clusters --workspace ${WORKSPACE_NAME}` command to see the list of clusters in the workspace.

```
dkp create appdeployment kube-prometheus-stack --app kube-prometheus-
stack-34.9.3 --workspace ${WORKSPACE_NAME} --clusters attached-
cluster1,attached-cluster2
```

2. **Optional:** Check the current status(see page 0) of the `AppDeployment` to see the names of the clusters where the application is currently enabled.

Enable or Disable an Application on a Per-cluster Basis **after it has been enabled at the workspace level**

You can enable or disable applications at any time. Once you have enabled the application at the workspace level(see page 0), the `spec.clusterSelector` field(see page 652) will be populated.

For clusters that are newly attached into the workspace, all applications enabled for the workspace are automatically enabled on and deployed to the new clusters.

If you want to see on which clusters your application is currently deployed, refer to the Print and Review the Current State of your AppDeployment(see page 0) documentation.

1. Edit the `AppDeployment` YAML by adding or removing the names of the clusters where you want to enable your application in the `clusterSelector` section:

```
cat <<EOF | kubectl apply -f -
apiVersion: apps.kommander.d2iq.io/v1alpha3
kind: AppDeployment
metadata:
  name: kube-prometheus-stack
  namespace: ${WORKSPACE_NAMESPACE}
spec:
  appRef:
    name: kube-prometheus-stack-34.9.3
    kind: ClusterApp
  clusterSelector:
    matchExpressions:
    - key: kommander.d2iq.io/cluster-name
      operator: In
      values:
      - attached-cluster1
      - attached-cluster3-new
EOF
```

Enable a Custom Configuration of an Application for a Cluster

You can customize the application for each cluster occurrence of said application. If you want to customize the application for a cluster that is not yet attached, refer to the instructions below(see page 437), so the application is deployed with the custom configuration on attachment.
To enable per-cluster customizations:

1. Reference the name of the `ConfigMap` to be applied per cluster in the `spec.clusterConfigOverrides` fields. In this example, you have three different customizations specified in three different `ConfigMaps` for three different clusters in one workspace:

```
cat <<EOF | kubectl apply -f -
apiVersion: apps.kommander.d2iq.io/v1alpha3
kind: AppDeployment
metadata:
  name: kube-prometheus-stack
  namespace: ${WORKSPACE_NAMESPACE}
spec:
  appRef:
```

```
    name: kube-prometheus-stack-34.9.3
    kind: ClusterApp
  clusterSelector:
    matchExpressions:
    - key: kommander.d2iq.io/cluster-name
      operator: In
      values:
      - attached-cluster1
      - attached-cluster2
      - attached-cluster3-new
  clusterConfigOverrides:
    - configMapName: kps-cluster1-overrides
      clusterSelector:
        matchExpressions:
        - key: kommander.d2iq.io/cluster-name
          operator: In
          values:
          - attached-cluster1
    - configMapName: kps-cluster2-overrides
      clusterSelector:
        matchExpressions:
        - key: kommander.d2iq.io/cluster-name
          operator: In
          values:
          - attached-cluster2
    - configMapName: kps-cluster3-overrides
      clusterSelector:
        matchExpressions:
        - key: kommander.d2iq.io/cluster-name
          operator: In
          values:
          - attached-cluster3-new
EOF
```

2. If you have not done so yet, create the ConfigMaps(see page 0) referenced in each `clusterConfigOverrides` entry.

- The changes are applied only if the YAML file has a valid syntax.
- Set up only one cluster override `ConfigMap` per cluster. If there are several `ConfigMaps` configured for a cluster, only one will be applied.
- Cluster override `ConfigMaps` must be created on the Management cluster.

Enable a Custom Configuration of an Application for a Cluster at Attachment
You can customize the application configuration for a cluster prior to its attachment, so that the application is deployed with this custom configuration on attachment. This is preferable, if you do not want to redeploy the application with an updated configuration after it has been initially installed, which may cause downtime.

To enable per-cluster customizations, follow these steps **before attaching the cluster**:

1. Set the `CLUSTER_NAME` environment variable to the cluster name that you will give your to-be-attached cluster.

```
export CLUSTER_NAME=<your_attached_cluster_name>
```

2. Reference the name of the `ConfigMap` you want to apply to this cluster in the `spec.clusterConfigOverrides` fields. **You do not need to update the `spec.clusterSelector` field**. In this example, you have the `kps-cluster1-overrides` customization specified for `attached-cluster-1` and a different customization (in `kps-your-attached-cluster-overrides` `ConfigMap`) for your to-be-attached cluster:

```
cat <<EOF | kubectl apply -f -
apiVersion: apps.kommander.d2iq.io/v1alpha3
kind: AppDeployment
metadata:
  name: kube-prometheus-stack
  namespace: ${WORKSPACE_NAMESPACE}
spec:
  appRef:
    name: kube-prometheus-stack-34.9.3
    kind: ClusterApp
  clusterSelector:
    matchExpressions:
    - key: kommander.d2iq.io/cluster-name
      operator: In
      values:
      - attached-cluster1
  clusterConfigOverrides:
    - configMapName: kps-cluster1-overrides
      clusterSelector:
        matchExpressions:
        - key: kommander.d2iq.io/cluster-name
          operator: In
          values:
          - attached-cluster1
    - configMapName: kps-your-attached-cluster-overrides
      clusterSelector:
        matchExpressions:
        - key: kommander.d2iq.io/cluster-name
          operator: In
          values:
          - ${CLUSTER_NAME}
EOF
```

3. If you have not done so yet, create the ConfigMap(see page 0) referenced for your to-be-attached cluster.

- The changes are applied only if the YAML file has a valid syntax.
- Cluster override `ConfigMaps` must be created on the Management cluster.

Disable a Custom Configuration of an Application for a Cluster

Enabled customizations are defined in a `ConfigMap`, which, in turn, is referenced in the `spec.clusterConfigOverrides` object of your `AppDeployment`.

1. Review your current configuration to establish what you want to remove:

   ```
   kubectl get appdeployment -n ${WORKSPACE_NAMESPACE} kube-prometheus-stack -o
   yaml
   ```

   The output looks similar to this:

   ```
   apiVersion: apps.kommander.d2iq.io/v1alpha3
   kind: AppDeployment
   metadata:
     name: kube-prometheus-stack
     namespace: ${WORKSPACE_NAMESPACE}
   spec:
     appRef:
       name: kube-prometheus-stack-34.9.3
       kind: ClusterApp
     configOverrides:
       name: kube-prometheus-stack-overrides-attached
     clusterSelector:
       matchExpressions:
       - key: kommander.d2iq.io/cluster-name
         operator: In
         values:
         - attached-cluster1
         - attached-cluster2
     clusterConfigOverrides:
       - configMapName: kps-cluster1-overrides
         clusterSelector:
           matchExpressions:
           - key: kommander.d2iq.io/cluster-name
             operator: In
             values:
             - attached-cluster1
       - configMapName: kps-cluster2-overrides
         clusterSelector:
           matchExpressions:
           - key: kommander.d2iq.io/cluster-name
             operator: In
             values:
   ```

```
          - attached-cluster2
```

Here you can see that `kube-prometheus-stack` has been enabled for the `attached-cluster1` and `attached-cluster2`. There is also a custom configuration for each of the clusters: `kps-cluster1-overrides` and `kps-cluster2-overrides`.

2. Edit the file by deleting the `configMapName` entry of the cluster for which you would like to delete the customization. This is located under the `clusterConfigOverrides`:

```
cat <<EOF | kubectl apply -f -
apiVersion: apps.kommander.d2iq.io/v1alpha3
kind: AppDeployment
metadata:
  name: kube-prometheus-stack
  namespace: ${WORKSPACE_NAMESPACE}
spec:
  appRef:
    kind: ClusterApp
    name: kube-prometheus-stack-34.9.3
  configOverrides:
    name: kube-prometheus-stack-ws-overrides
  clusterSelector:
    matchExpressions:
      - key: kommander.d2iq.io/cluster-name
        operator: In
        values:
        - attached-cluster1
  clusterConfigOverrides:
    - configMapName: kps-cluster1-overrides
      clusterSelector:
        matchExpressions:
        - key: kommander.d2iq.io/cluster-name
          operator: In
          values:
          - attached-cluster1
EOF
```

Compare steps one and two for a reference of how an entry should be deleted.

3. Before deleting a `ConfigMap` that contains your customization, **ensure you will NOT require it at a later time**. It is not possible to restore a deleted `ConfigMap`.
If you choose to delete it, run:

```
kubectl delete configmap <name_configmap> -n ${WORKSPACE_NAMESPACE}
```

It is **NOT** possible to delete a `ConfigMap` that is being actively used and referenced in the `configOverride` of any `AppDeployment`.

Verify the Applications and your Current Configuration

Refer to the Verify applications help(see page 0) to connect to the managed or attached cluster and check the status of the deployments.

If you want to know how the `AppDeployment` resource is currently configured, refer to the Print and review the state of your AppDeployment(see page 418) section.

8.4.6.2 Workspace Catalog Applications

ENTERPRISE

Catalog applications are any third-party or open source applications that appear in the Catalog.

D2iQ provides DKP Catalog Applications(see page 441) for use in your environment.

Workspace DKP Catalog Applications

ENTERPRISE

Catalog applications are applications provided by D2iQ for use in your environment.

Install the DKP catalog via the CLI

Follow these steps to install the DKP catalog from the CLI.

1. Refer to Install DKP in an Air-gapped Environment with Catalog Applications(see page 372) instructions, if you are running in air-gapped environment.
2. Set the `WORKSPACE_NAMESPACE` environment variable to the name of your workspace's namespace:

   ```
   export WORKSPACE_NAMESPACE=<workspace namespace>
   ```

3. Create the `GitRepository`:

   ```
   kubectl apply -f - <<EOF
   apiVersion: source.toolkit.fluxcd.io/v1beta1
   kind: GitRepository
   metadata:
     name: dkp-catalog-applications
     namespace: ${WORKSPACE_NAMESPACE}
     labels:
       kommander.d2iq.io/gitapps-gitrepository-type: catalog
       kommander.d2iq.io/gitrepository-type: catalog
   spec:
     interval: 1m0s
     ref:
       tag: v2.3.3
     timeout: 20s
     url: https://github.com/mesosphere/dkp-catalog-applications
   ```

```
EOF
```

4. Verify that you can see the DKP workspace catalog `Apps` available in the UI, and in the CLI, using `kubectl`:

```
kubectl get apps -n ${WORKSPACE_NAMESPACE}e
```

Workspace DKP catalog applications

Name	App ID	Supported Kubernetes version
kafka-operator-0.20.0	kafka-operator	1.21
kafka operator-0.20.2	kafka-operator	1.21 - 1.23
spark-operator-1.1.6	spark-operator	1.21
spark operator-1.1.17	spark-operator	1.21 - 1.23
zookeeper-operator-0.2.13	zookeeper-operator	1.21 - 1.23

Kafka in a Workspace

ENTERPRISE

Information about the Kafka Operator

Apache Kafka[367] is an open-source distributed event streaming platform used for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications. The Kafka Operator[368] is a Kubernetes operator to automate provisioning, management, autoscaling, and operations of Apache Kafka clusters deployed to Kubernetes. It works by watching custom resources, such as `KafkaClusters`, `KafkaUsers`, and `KafkaTopics`, to provision underlying Kubernetes resources (i.e `StatefulSets`) required for a production-ready Kafka Cluster.

Install
Follow these steps to install the Kafka operator in a workspace. This procedure results in a Kafka operator running in a workspace namespace, ready to manage and create Kafka clusters in any project namespaces. See the Kafka in a Project(see page 487) custom resource documentation for more information on creating Kafka clusters.

Only install the Kafka operator once per workspace.

367 https://kafka.apache.org/
368 https://banzaicloud.com/docs/supertubes/kafka-operator/

1. Follow the generic installation instructions for workspace catalog applications on the Application Deployment(see page 447) page.
2. Within the `AppDeployment`, update the `appRef` to specify the correct `kafka-operator` App. You can find the `appRef.name` by listing the available `Apps` in the workspace namespace:

```
kubectl get apps -n ${WORKSPACE_NAMESPACE}
```

Refer to the Kafka operator Helm Chart documentation[369] for details on custom configuration for the operator.

Uninstall via the CLI

Uninstalling the Kafka operator does not affect existing `KafkaCluster` deployments. After uninstalling the operator, you must manually remove any remaining Custom Resource Definitions (CRDs) from the operator.

1. Delete all of the deployed Kafka custom resources (see Deleting Kafka Custom Resources(see page 0)).
2. Uninstall a Kafka operator `AppDeployment`:

```
kubectl -n <workspace namespace> delete AppDeployment <name of AppDeployment>
```

3. Remove Kafka CRDs:
   NOTE: The CRDs are not finalized for deletion until you delete the associated custom resources.

```
kubectl delete crds kafkaclusters.kafka.banzaicloud.io
kafkausers.kafka.banzaicloud.io kafkatopics.kafka.banzaicloud.io
```

Resources

- Kafka Operator Documentation[370]
- Kafka Operator GitHub Repository[371]
- Sample Kafka Operator Custom Resources[372]
- Apache Kafka Documentation[373]

Spark Operator in a Workspace

ENTERPRISE

How to spin up your Spark Operator

The Kubernetes Operator for Apache Spark aims to make specifying and running Spark applications as easy and idiomatic as running other workloads on Kubernetes. It uses Kubernetes custom resources for specifying, running, and surfacing status of Spark applications. For a complete reference of the custom resource definitions, please refer to the API Definition. For details on its design, please refer to the design documentation. It requires Spark 2.3 and above that supports Kubernetes as a native scheduler backend.

369 https://github.com/banzaicloud/koperator/tree/master/charts/kafka-operator#configuration
370 https://banzaicloud.com/docs/supertubes/kafka-operator/
371 https://github.com/banzaicloud/koperator
372 https://github.com/banzaicloud/koperator/tree/master/config/samples
373 https://kafka.apache.org/documentation/

The default installation is basic, please provide your override configmap to enable desired Spark Operator features.

Install Spark Operator
You can find generic installation instructions for workspace catalog applications on the Application Deployment(see page 447) topic.

Only install the Spark operator once per workspace.

For details on custom configuration for the operator, refer to the Spark Operator Helm Chart documentation[374].

After you finish the installation, see Spark Operator in a Project(see page 491) custom resource documentation for more information about how to submit your Spark jobs.

Sample Override Configuration File

Ensure you configure the AppDeployment with the appropriate override configmap.

- Using UI

```
podLabels:
  owner: john
  team: operations
```

- Using CLI
  See Application Deployment(see page 447) for details.

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: ${WORKSPACE_NAMESPACE}
  name: spark-operator-overrides
data:
  values.yaml: |
    configInline:
      podLabels:
        owner: john
        team: operations
EOF
```

Uninstall via the CLI

374 https://github.com/mesosphere/spark-on-k8s-operator/blob/d2iq-master/charts/spark-operator-chart/README.md

Uninstalling the Spark Operator does not affect existing **SparkApplication** and **ScheduledSparkApplication** custom resources. You need to manually remove any leftover custom resources and CRDs from the operator. Please refer to deleting Spark Operator custom resources(see page 0).

Follow these steps:

1. Uninstall the Spark Operator `AppDeployment`:

```
kubectl -n <your workspace namespace> delete AppDeployment <name of
AppDeployment>
```

2. Remove the Spark Operator Service Account:

```
# <name of service account> is spark-operator-service-account if you didn't
override the RBAC resources.
kubectl -n <your workspace namespace> delete serviceaccounts <name of service
account>
```

3. Remove the Spark Operator CRDs:
NOTE: The CRDs are not finalized for deletion until you delete the associated custom resources.

```
kubectl delete crds scheduledsparkapplications.sparkoperator.k8s.io
sparkapplications.sparkoperator.k8s.io
```

Resources
Here are some resources to learn more about Spark Operator:

- Spark Operator Documentation[375]
- Spark Operator Quick Start Guide[376]
- Spark Operator User Guide[377]
- Spark Documentation[378]

Zookeeper Operator in a Workspace

ENTERPRISE

ZooKeeper[379] is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. The ZooKeeper operator[380] is a Kubernetes operator that handles the provisioning and management of ZooKeeper clusters. It works by watching custom resources, such as `ZookeeperClusters`, to provision the underlying Kubernetes resources (`StatefulSets`) required for a production-ready ZooKeeper Cluster.

375 https://github.com/mesosphere/spark-on-k8s-operator/blob/d2iq-master/README.md
376 https://github.com/mesosphere/spark-on-k8s-operator/blob/d2iq-master/docs/quick-start-guide.md
377 https://github.com/mesosphere/spark-on-k8s-operator/blob/d2iq-master/docs/user-guide.md
378 https://spark.apache.org/docs/latest/
379 https://zookeeper.apache.org/index.html
380 https://github.com/pravega/zookeeper-operator

Install Zookeeper

Follow these steps to install the ZooKeeper operator in a workspace. This procedure results in a ZooKeeper operator running in a workspace namespace, ready to manage and create ZooKeeper clusters in any project namespaces. See the ZooKeeper in a Project(see page 489) custom resource documentation for more information on creating ZooKeeper clusters.

Only install the ZooKeeper operator once per workspace.

1. Follow the generic installation instructions for workspace catalog applications on the Application Deployment(see page 447) page.
2. Within the `AppDeployment`, update the `appRef` to specify the correct `zookeeper-operator` App. You can find the `appRef.name` by listing the available `Apps` in the workspace namespace:

```
kubectl get apps -n ${WORKSPACE_NAMESPACE}
```

For details on custom configuration for the operator, please refer to the ZooKeeper operator Helm Chart documentation[381].

Uninstall via the CLI

Uninstalling the ZooKeeper operator will not directly affect any running `ZookeeperClusters`. By default, the operator waits for any `ZookeeperClusters` to be deleted before it will fully uninstall (you can set `hooks.delete: true` in the application configuration to disable this behavior). After uninstalling the operator, you need to manually clean up any leftover Custom Resource Definitions (CRDs).

1. Delete all `ZookeeperClusters`, see Deleting ZooKeeper Clusters[382].
2. Uninstall a ZooKeeper operator `AppDeployment`:

```
kubectl -n <workspace namespace> delete AppDeployment <name of AppDeployment>
```

3. Remove ZooKeeper CRDs:
   WARNING: Once you remove the CRDs, all deployed ZookeeperClusters will be deleted!

```
kubectl delete crds zookeeperclusters.zookeeper.pravega.io
```

Resources

- ZooKeeper Operator Documentation[383]
- ZooKeeper Cluster Helm Chart[384]
- ZooKeeper Documentation[385]

381 https://github.com/pravega/zookeeper-operator/tree/master/charts/zookeeper-operator#configuration
382 https://d2iq.atlassian.net/wiki/spaces/DKP/pages/10420421/Zookeeper+in+a+Project#Deleting-ZooKeeper-clusters
383 https://github.com/pravega/zookeeper-operator
384 https://github.com/pravega/zookeeper-operator/tree/master/charts/zookeeper
385 https://zookeeper.apache.org/documentation

DKP Kaptain in a Workspace

ENTERPRISE

Kaptain is a DKP catalog application that allows data scientists to implement machine learning models from Jupyter notebooks with KubeFlow. Kaptain packages together many key libraries and toolkits needed to deploy models at scale. To find out more about Kaptain, refer to the Kaptain documentation[386].

Install Kaptain
To install this DKP catalog application, refer to the Install Kaptain[387] documentation.

Uninstall
To uninstall this DKP catalog application, refer to the Uninstall Kaptain[388] documentation.

Deployment of Catalog Applications in Workspaces

ENTERPRISE

Deploy applications to attached clusters using the CLI

This topic describes how to use the CLI to deploy a workspace catalog application to attached clusters within a workspace. To deploy an application to selected clusters within a workspace, refer to the Cluster-scoped Configuration(see page 434) section of the documentation.

Prerequisites

Before you begin, you must have:

- A running cluster with Kommander installed(see page 351).
- An Attach an Existing Kubernetes Cluster(see page 505) section of the documentation completed.

Set the `WORKSPACE_NAMESPACE` environment variable to the name of the workspace's namespace the attached cluster exists in:

```
export WORKSPACE_NAMESPACE=<workspace_namespace>
```

After creating a GitRepository, use either the DKP UI or the CLI to enable your catalog applications.

⚠ From within a workspace, you can enable applications to deploy. Verify that an application has successfully deployed via the CLI[389].

Enable the Application using the DKP UI

Follow these steps to enable your catalog applications from the DKP UI:

1. From the top menu bar, select your target workspace.

386 https://d2iq.atlassian.net/wiki/spaces/DKAP
387 https://d2iq.atlassian.net/wiki/spaces/DKAP/pages/13271060/Install+Kaptain
388 https://d2iq.atlassian.net/wiki/spaces/DKAP/pages/50298944/Uninstall+Kaptain
389 https://d2iq.atlassian.net/wiki/spaces/DKP/pages/10387674/Deployment+of+Applications+in+Workspaces#Verify-applications

2. Select **Applications** from the sidebar menu to browse the available applications from your configured repositories.
3. Select the three dot button from the bottom-right corner of the desired application tile, and then select **Enable**.
4. If available, select a version from the drop-down menu. This drop-down menu will only be visible if there is more than one version.
5. (Optional) If you want to override the default configuration values, copy your customized values into the text editor under **Configure Service** or upload your yaml file that contains the values:

```
someField: someValue
```

6. Confirm the details are correct, and then select the **Enable** button.

For all applications, you must provide a display name and an ID which is automatically generated based on what you enter for the display name, unless or until you edit the ID directly. The ID must be compliant with Kubernetes DNS subdomain name validation rules[390].

Alternately, you can use the CLI to enable your catalog applications(see page 0).

Enable the Application using the CLI

See Workspace Catalog Applications(see page 441) for the list of available applications that you can deploy on the attached cluster.

1. Enable a supported application to deploy to your Attached Cluster(see page 505) with an `AppDeployment` resource.
2. Within the `AppDeployment`, define the `appRef` to specify which `App` to enable:

```
cat <<EOF | kubectl apply -f -
apiVersion: apps.kommander.d2iq.io/v1alpha3
kind: AppDeployment
metadata:
  name: spark-operator
  namespace: ${WORKSPACE_NAMESPACE}
spec:
  appRef:
    name: spark-operator-1.1.17
    kind: App
EOF
```

- The `appRef.name` must match the app `name` from the list of available catalog applications.
- Create the resource in the workspace you just created, which instructs Kommander to deploy the `AppDeployment` to the `KommanderCluster`s in the same workspace.

390 https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#dns-subdomain-names

Enable an Application with a Custom Configuration using the CLI

Follow these steps:

Provide the name of a `ConfigMap` in the `AppDeployment`, which provides custom configuration on top of the default configuration:

1.
```
cat <<EOF | kubectl apply -f -
apiVersion: apps.kommander.d2iq.io/v1alpha3
kind: AppDeployment
metadata:
  name: spark-operator
  namespace: ${WORKSPACE_NAMESPACE}
spec:
  appRef:
    name: spark-operator-1.1.17
    kind: App
  configOverrides:
    name: spark-operator-overrides
EOF
```

2. Create the `ConfigMap` with the name provided in the step above, with the custom configuration:

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: ${WORKSPACE_NAMESPACE}
  name: spark-operator-overrides
data:
  values.yaml: |
    configInline:
      uiService:
        enable: false
EOF
```

Kommander waits for the `ConfigMap` to be present before deploying the `AppDeployment` to the managed or attached clusters.

Verify Applications

The applications are now enabled. Connect to the attached cluster and check the `HelmReleases` to verify the deployment:

```
kubectl get helmreleases -n ${WORKSPACE_NAMESPACE}
```

The output looks similar to this:

```
NAMESPACE                NAME                READY   STATUS
AGE
workspace-test-vjsfq     spark-operator      True    Release reconciliation succeeded
7m3s
```

Workspace Catalog Application Upgrades

ENTERPRISE

Upgrade catalog applications using the CLI and UI

Upgrade Catalog Applications

Before upgrading your catalog applications, verify the current and supported versions of the application. Also, keep in mind the distinction between Platform applications and Catalog applications. Platform applications are deployed and upgraded as a set for each cluster or workspace. Catalog applications are deployed separately, so that you can deploy and upgrade them individually for each project.

If you are running on an **air-gapped environment with catalog applications**, ensure you have updated your catalog repository before upgrading. The catalog repository should contain the Docker registry containing the DKP catalog application(see page 441) images and a charts bundle file containing DKP catalog applications(see page 0) charts.

Catalog applications must be upgraded to the latest version BEFORE upgrading the Kubernetes version (or Konvoy version for managed Konvoy clusters) on attached clusters, due to the previous versions' incompatibility with Kubernetes 1.22.

Upgrade with UI
Follow these steps to upgrade an application from the DKP UI:

1. From the top menu bar, select your target workspace.
2. Select **Applications** from the sidebar menu.
3. Select the three dot button from the bottom-right corner of the desired application tile, and then select **Edit**.
4. Select the **Version** drop-down, and select a new version. This drop-down will only be available if there is a newer version to upgrade to.
5. Select **Save**.

Upgrade with CLI

```
dkp upgrade catalogapp <appdeployment-name> --workspace=my-workspace --to-
version=<version.number>
```

For example, the following command upgrades the Kafka Operator application, named `kafka-operator-abc`, in a workspace to version `0.20.2`:

```
dkp upgrade catalogapp kafka-operator-abc --workspace=my-workspace --to-version=0.20.2
```

Platform applications cannot be upgraded on a one-off basis, and must be upgraded in a single process for each workspace. If you attempt to upgrade a platform application with these commands, you receive an error and the application is not upgraded.

Custom Applications

ENTERPRISE

Custom applications are third-party applications you have added to the DKP Catalog.

Custom applications are any third-party applications that are not provided in the DKP Application Catalog. Custom applications can leverage applications from the DKP Catalog or be fully-customized. There is no expectation of support by D2iQ for a Custom application. Custom applications can be deployed on Konvoy clusters or on any D2iQ supported 3rd party Kubernetes distribution.

Create a Git Repository

ENTERPRISE

Create a Git Repository in the Workspace namespace

Use the CLI to create the GitRepository resource and add a new repository to your Workspace.

1. Refer to Air-gapped Environment(see page 369) setup instructions section, if you are running in air-gapped environment.
2. Set the `WORKSPACE_NAMESPACE` environment variable to the name of your workspace's namespace:

   ```
   export WORKSPACE_NAMESPACE=<workspace_namespace>
   ```

3. Adapt the URL of your Git repository:

   ```
   kubectl apply -f - <<EOF
   apiVersion: source.toolkit.fluxcd.io/v1beta1
   kind: GitRepository
   metadata:
     name: example-repo
     namespace: ${WORKSPACE_NAMESPACE}
     labels:
       kommander.d2iq.io/gitapps-gitrepository-type: dkp
       kommander.d2iq.io/gitrepository-type: catalog
   spec:
     interval: 1m0s
     ref:
       branch: <your-target-branch-name> # e.g., main
     timeout: 20s
     url: https://github.com/<example-org>/<example-repo>
   EOF
   ```

4. Ensure the status of the `GitRepository` signals a ready state:

```
kubectl get gitrepository example-repo -n ${WORKSPACE_NAMESPACE}
```

The repository commit also displays the ready state:

```
NAME         URL                                                     READY
STATUS                                                          AGE
example-repo https://github.com/example-org/example-repo             True
Fetched revision: master/6c54bd1722604bd03d25dcac7a31c44ff4e03c6a   11m
```

For more information on the GitRepository resource fields and how to make Flux aware of credentials required to access a private Git repository, see the Flux documentation[391].

Troubleshoot
To troubleshoot issues with adding the GitRepository, review the following logs:

```
kubectl -n kommander-flux logs -l app=source-controller
[...]
kubectl -n kommander-flux logs -l app=kustomize-controller
[...]
kubectl -n kommander-flux logs -l app=helm-controller
[...]
```

Related Information

- Flux[392]
- Flux docs[393]

Git Repository Structure

ENTERPRISE

Git repositories must be structured in a specific manner for defined applications to be processed by Kommander.

You must structure your git repository based on the following guidelines, for your applications to be processed properly by Kommander so that they can be deployed.

Git Repository Directory Structure
Use the following basic directory structure for your git repository:

```
├── helm-repositories
│   ├── <helm repository 1>
│   │   ├── kustomization.yaml
│   │   └── <helm repository name>.yaml
│   └── <helm repository 2>
│     ├── kustomization.yaml
```

391 https://fluxcd.io/docs/components/source/gitrepositories/
392 https://fluxcd.io/
393 https://fluxcd.io/docs

```
│   └── <helm repository name>.yaml
└── services
    ├── <app name>
    │   ├── <app version1> # semantic version of the app helm chart. e.g., 1.2.3
    │   │   ├── defaults
    │   │   │   ├── cm.yaml
    │   │   │   └── kustomization.yaml
    │   │   ├── <app name>.yaml
    │   │   └── kustomization.yaml
    │   ├── <app version2> # another semantic version of the app helm chart. e.g.,
2.3.4
    │   │   ├── defaults
    │   │   │   ├── cm.yaml
    │   │   │   └── kustomization.yaml
    │   │   ├── <app name>.yaml
    │   │   └── kustomization.yaml
    │   └── metadata.yaml
    └── <another app name>
    ...
```

- Define applications in the `services/` directory.
- You can define multiple versions of an application, under different directories nested under the `services/<app name>/` directory.
- Define application manifests, such as a HelmRelease[394], under each versioned directory `services/<app name>/<version>/` in `<app name>.yaml` which is listed in the `kustomization.yaml` Kubernetes Kustomization file. For more information, see the Kubernetes Kustomization docs[395].
- Define the default values ConfigMap for `HelmReleases` in the `services/<app name>/<version>/defaults` directory, accompanied by a `kustomization.yaml` Kubernetes Kustomization file pointing to the `ConfigMap` file.
- Define the `metadata.yaml` of each application under the `services/<app name>/` directory. For more information, see the Workspace Application Metadata docs(see page 454).

See the DKP Catalog repository[396] for an example of how to structure custom catalog Git repositories.

Helm Repositories

You must include the `HelmRepository` that is referenced in each `HelmRelease`'s `Chart` spec.

Each `services/<app name>/<version>/kustomization.yaml` must include the path of the YAML file that defines the `HelmRepository`. For example:

```
# services/<app name>/<version>/kustomization.yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
resources:
  - <app name>.yaml
```

394 https://fluxcd.io/docs/components/helm/helmreleases/
395 https://kubectl.docs.kubernetes.io/references/kustomize/kustomization/
396 https://github.com/mesosphere/dkp-catalog-applications

```
  - ../../../helm-repositories/<helm repository 1>
```

For more information, see the flux documentation about HelmRepositories[397].

Substitution variables
Some substitution variables[398] are provided.

- `${releaseName}`: For each App deployment, this variable is set to the `AppDeployment` name. Use this variable to prefix the names of any resources that are defined in the application directory in the Git repository so that multiple instances of the same application can be deployed. If you create resources without using the `releaseName` prefix (or suffix) in the name field, there can be conflicts if the same named resource is created in that same namespace.
- `${releaseNamespace}`: The namespace of the Workspace.
- `${workspaceNamespace}`: The namespace of the Workspace that the Workspace belongs to.

Related Information
- Flux[399]
- Flux docs[400]
- Getting started with Flux guide[401]

Workspace Application Metadata

ENTERPRISE

To display more information about custom applications in the UI, define a metadata.yaml file for each application in the git repository.

You can define how custom applications display in the DKP UI by defining a `metadata.yaml` file for each application in the git repository. You must define this file at `services/<application>/metadata.yaml` for it to process correctly.

You can define the following fields:

Field	Default	Description
displayName	falls back to App ID	Display name of the application for the UI.
description	“”	Short description, should be a sentence or two, displayed in the UI on the application card.

397 https://fluxcd.io/docs/components/source/helmrepositories/
398 https://fluxcd.io/docs/components/kustomize/kustomization/#variable-substitution
399 https://fluxcd.io/
400 https://fluxcd.io/docs
401 https://fluxcd.io/docs/get-started/

Field	Default	Description
category	general	1 or more categories for this application. Categories are used to group applications in the UI.
overview		Markdown overview used on the application detail page in the UI.
icon		Base64 encoded icon SVG file used for application logos in the UI.

None of these fields are required for the application to display in the UI.

Here is an example `metadata.yaml` file:

```
displayName: Prometheus Monitoring Stack
description: Stack of applications that collect metrics and provides visualization
and alerting capabilities. Includes Prometheus, Prometheus Alertmanager and Grafana.
category:
  - monitoring
overview: >
  # Overview
  A stack of applications that collects metrics and provides visualization and
alerting capabilities. Includes Prometheus, Prometheus Alertmanager and Grafana.

  ## Dashboards
  By deploying the Prometheus Monitoring Stack, the following platform applications
and their respective dashboards are deployed. After deployment to clusters in a
workspace, the dashboards are available to access from a respective cluster's detail
page.

  ### Prometheus

  A software application for event monitoring and alerting. It records real-time
metrics in a time series database built using a HTTP pull model, with flexible and
real-time alerting.

  - [Prometheus Documentation - Overview](https://prometheus.io/docs/introduction/
overview/)

  ### Prometheus Alertmanager
  A Prometheus component that enables you to configure and manage alerts sent by the
Prometheus server and to route them to notification, paging, and automation systems.

  - [Prometheus Alertmanager Documentation - Overview](https://prometheus.io/docs/
alerting/latest/alertmanager/)

  ### Grafana
```

```
  A monitoring dashboard from Grafana that can be used to visualize metrics collected
by Prometheus.

  - [Grafana Documentation](https://grafana.com/docs/)
icon:
```

```
PHN2ZyB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciIHZpZXdCb3g9IjAgMCAzMDAgMzAwIiBzd
HlsZT0iZW5hYmxlLWJhY2tncm91bmQ6bmV3IDAgMCAzMDAgMzAwIiB4bWw6c3BhY2U9InByZXNlcnZlIj48cG
F0aCBkPSJNMTUwIDUwQzk0LjggNTAgNTAgOTQuOCA1MCAxNTBzNDQuOCAxMDAgMTAwIDEwMCAxMDAtNDQuOCA
xMDAtMTAwUzIwNS4yIDUwIDE1MCA1MHptMCAxODcuMmMtMTUuNyAwLTI4LjUtMTAuNS0yOC41LTIzLjRoNTYu
OWMuMSAxMi45LTEyLjcgMjMuNC0yOC40IDIzLjR6bTQ3LTMxLjJoLTk0di0xN2g5NHYxN3ptLS4zLTI1LjloL
TkzLjRjLS4zLS40LS42LS43LS45LTEuMS05LjYtMTEuNy0xMS45LTE3LjgtMTQuMS0yNCAwLS4yIDExLjcgMi
40IDIwIDQuMyAwIDAgNC4zIDEgMTAuNSAyLjEtNi03LTkuNi0xNi05LjYtMjUuMSAwLTIwIDE1LjQtMzcuNiA
5LjgtNTEuNyA1LjQuNCAxMS4yIDExLjQgMTEuNiAyOC41IDUuNy03LjkgOC4xLTIyLjQgOC4xLTMxLjMgMC05
LjIgNi4xLTE5LjkgMTIuMS0yMC4yLTUuNCA4LjkgMS40IDE2LjUgNy40IDM1LjUgMi4zIDcuMSAyIDE5LjEgM
y43IDI2LjcuNi0xNS44IDMuMy0zOC44IDEzLjMtNDYuNy00LjQgMTAgLjcgMjIuNSA0LjEgMjguNSA1LjYgOS
43IDkgMTcuMSA5IDMxIDAgOS4zLTMuNCAxOC4xLTkuMyAyNy4yLjYtMS4yIDExLjItMi40IDExLjItMi40bDI
xLjQtNC4yYy4xIDAtMyAxMi44LTE0LjkgMjUuMXoiIHN0eWxlPSJmaWxsOiNmODQzMTEiLz48L3N2Zz4=
```


Enable a Custom Application from the Workspace Catalog

ENTERPRISE

Enable a Custom Application from the Workspace Catalog

After creating a GitRepository, you can either use the DKP UI or the CLI to enable your custom applications. To deploy an application to selected clusters within a workspace, refer to the cluster-scoped configuration(see page 434) section.

> From within a workspace, you can enable applications to deploy. Verify that an application has successfully deployed via the CLI(see page 0).

Enable the application using the DKP UI
Follow these steps:

1. From the top menu bar, select your target workspace.
2. Select **Applications** from the sidebar menu to browse the available applications from your configured repositories.
3. Select the three dot button from the bottom-right corner of the desired application tile, and then select **Enable**.
4. If available, select a version from the drop-down menu. This drop-down menu will only be visible if there is more than one version.
5. (Optional) If you want to override the default configuration values, copy your customized values into the text editor under **Configure Service** or upload your yaml file that contains the values:

   ```
   someField: someValue
   ```

6. Confirm the details are correct, and then select the **Enable** button.

For all applications, you must provide a display name and an ID which is automatically generated based on what you enter for the display name, unless or until you edit the ID directly. The ID must be compliant with Kubernetes DNS subdomain name validation rules[402].

Alternately, you can use the CLI to enable your custom applications(see page 456).

Prerequisites

- Determine the name of the workspace where you wish to perform the deployments. You can use the `dkp get workspaces` command to see the list of workspace names and their corresponding namespaces.
- Set the `WORKSPACE_NAMESPACE` environment variable to the name of the workspace's namespace where the cluster is attached:

```
export WORKSPACE_NAMESPACE=<workspace_namespace>
```

Enable the Application using the CLI
Follow these steps:

1. Get the list of available applications to enable using the following command:

```
kubectl get apps -n ${WORKSPACE_NAMESPACE}
```

2. Deploy one of the supported applications from the list with an `AppDeployment` resource.
3. Within the `AppDeployment`, define the `appRef` to specify which `App` will be enabled:

```
cat <<EOF | kubectl apply -f -
apiVersion: apps.kommander.d2iq.io/v1alpha3
kind: AppDeployment
metadata:
  name: my-custom-app
  namespace: ${WORKSPACE_NAMESPACE}
spec:
  appRef:
    name: custom-app-0.0.1
    kind: App
EOF
```

- The `appRef.name` must match the app `name` from the list of available catalog applications.
- Create the resource in the workspace you just created, which instructs Kommander to deploy the `AppDeployment` to the `KommanderCluster`s in the same workspace.

Enable an Application with a Custom Configuration using the CLI
Follow these steps:

402 https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#dns-subdomain-names

1. Provide the name of a `ConfigMap` in the `AppDeployment`, which provides custom configuration on top of the default configuration:

```
cat <<EOF | kubectl apply -f -
apiVersion: apps.kommander.d2iq.io/v1alpha3
kind: AppDeployment
metadata:
  name: my-custom-app
  namespace: ${WORKSPACE_NAMESPACE}
spec:
  appRef:
    name: custom-app-0.0.1
    kind: App
  configOverrides:
    name: my-custom-app-overrides
EOF
```

2. Create the `ConfigMap` with the name provided in the step above, with the custom configuration:

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: ${WORKSPACE_NAMESPACE}
  name: my-custom-app-overrides
data:
  values.yaml: |
    someField: someValue
EOF
```

Kommander waits for the `ConfigMap` to be present before deploying the `AppDeployment` to the managed or attached clusters in the Workspace.

Verify Applications
After completing the previous steps, your applications are enabled. Connect to the attached cluster and check the `HelmReleases` to verify the deployments:

```
kubectl get helmreleases -n ${WORKSPACE_NAMESPACE}
```

The output looks similar to this:

```
NAMESPACE               NAME            READY   STATUS
AGE
workspace-test-vjsfq    my-custom-app   True    Release reconciliation succeeded
7m3s
```

8.4.7 Workspace Role Bindings

ENTERPRISE

Workspace Role Bindings grant access to specified Workspace Roles for a specified group of people.

8.4.7.1 Configure Workspace Role Bindings

Before you can create a Workspace Role Binding, ensure you have created a Group. A Kommander Group can contain one or several Identity Provider users, groups or both.

You can assign a role to this Kommander Group:

1. From the top menu bar, select your target workspace.
2. Select **Access Control** in the **Administration** section of the sidebar menu.
3. Select the **Cluster Role Bindings** tab, and then select the **Add Roles** button next to the group you want.
4. Select the Role, or Roles, you want from the drop-down menu and select **Save**.

8.5 Projects

ENTERPRISE

Multi-cluster Configuration Management

Projects support the management of configMaps, continuous deployments, secrets, services, quotas, and role-based access control and multi-tenant logging by leveraging federated resources. When a Project is created, DKP creates a federated namespace that is propagated to the Kubernetes clusters associated with this Project.

Federation in this context means that a common configuration is pushed out from a central location (DKP) to all Kubernetes clusters, or a pre-defined subset group, under DKP management. This pre-defined subset group of Kubernetes clusters is called a Project.

Projects enable teams to deploy their configurations and services to clusters in a consistent way. Projects enable central IT or a business unit to share their Kubernetes clusters among several teams. Using Projects, DKP leverages Kubernetes Cluster Federation (KubeFed) to coordinate the configuration of multiple Kubernetes clusters.

Kommander allows a user to use labels to select, manually or dynamically, the Kubernetes clusters associated with a Project.

8.5.1 Project Namespace

Project Namespaces isolate configurations across clusters. Individual standard Kubernetes namespaces are automatically created on all clusters belonging to the project. When creating a new project, you can customize the Kubernetes namespace name that is created. It is the grouping of all of these individual standard Kubernetes namespaces that make up the concept of a Project Namespace. A Project Namespace is a Kommander specific concept.

8.5.2 Create a Project

When you create a Project, you must specify a Project Name, a Namespace Name (optional) and a way to allow Kommander to determine which Kubernetes clusters will be part of this project.

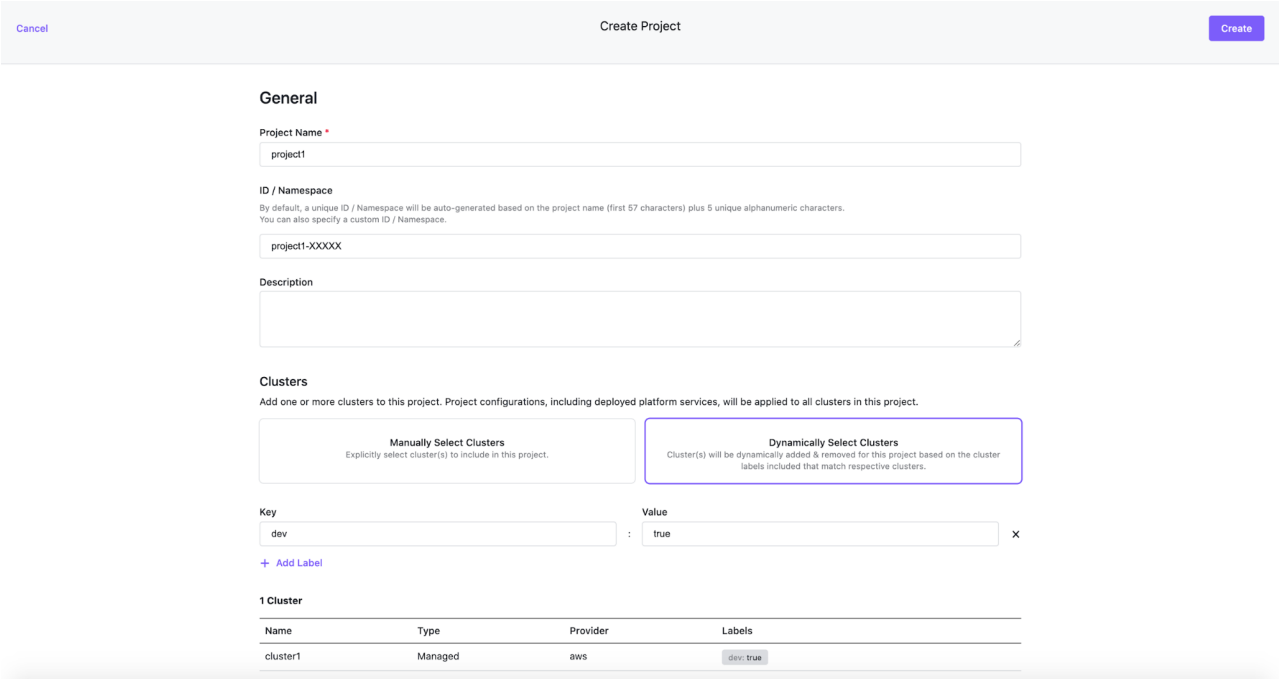
As mentioned previously, a Project Namespace corresponds to a Kubernetes Federated Namespace. By default, the name of the namespace is auto-generated based on the project name (first 57 characters) plus 5 unique alphanumeric characters. You can specify a namespace name, but you must ensure it does not conflict with any existing namespace on the target Kubernetes clusters, that will be a part of the Project.

To determine which Kubernetes clusters will be part of this project, you can either select manually existing clusters or define labels that Kommander will use to dynamically add clusters. The latter is recommended because it will allow you to deploy additional Kubernetes clusters later and to have them automatically associated with Projects based on their labels.

To create a Project, you can either use the DKP UI or create a Project object on the Kubernetes cluster where Kommander is running (using kubectl or the Kubernetes API). The latter allows you to configure Kommander resources in a declarative way. It is available for all kinds of Kommander resources.

8.5.2.1 Create a Project - UI Method

Here is an example of what it looks like to create a project using the DKP UI:



8.5.2.2 Create a Project - CLI Method

The following sample is a YAML Kubernetes object for creating a Kommander Project. This example does not work verbatim because it depends on a workspace name that has been previously created and does not exist by default in your cluster. Use this as an example format and fill in the workspace name and namespace name appropriately along with the proper labels.

```
apiVersion: workspaces.kommander.mesosphere.io/v1alpha1
kind: Project
metadata:
  name: My-Project-Name
  namespace: my-project-k8s-namespace-name
spec:
```

```
  workspaceRef:
    name: myworkspacename
  namespaceName: myworkspacename-di3tx
  placement:
    clusterSelector:
      matchLabels:
        cluster: prod
```

The following procedures are supported for projects:

- Manage Project ConfigMaps(see page 471)
- Manage Project Role Bindings(see page 468)
- Manage Project Quotas and Limit Ranges(see page 474)
- Manage Project Roles(see page 465)
- Manage Project Secrets(see page 473)

8.5.3 Project Deployments

ENTERPRISE

Use Project Deployments to manage GitOps based Continuous Deployments

You can configure Kommander Projects with GitOps-based Continuous Deployments for federation of your Applications to associated clusters of the project. This is backed by Flux, which enables software and applications to be continuously deployed (CD) using GitOps processes. GitOps enables the application to be deployed as per a manifest that is stored in a Git repository. This ensures that the application deployment can be automated, audited, and declaratively deployed to the infrastructure.

8.5.3.1 What is GitOps?

GitOps is a modern software deployment strategy. The configuration that describes how your application is deployed to a cluster are stored in a Git repository. The configuration is continuously synchronized from the Git repository to the cluster, ensuring that the specified state of the cluster always matches what is defined in the “GitOps” Git repository.

The benefits of using a GitOps deployment strategy are:

- Familiar, collaborative change and review process. Engineers are intimately familiar with Git-based workflows: branches, pull requests, code reviews, etc. GitOps leverages this experience to control the deployment of software and updates to catch issues early.
- Clear change log and audit trail. The Git commit log serves as an audit trail to answer the question: “who changed what, and when?” Having such information available, you can contact the right people when fixing or prioritizing a production incident to determine the why and correctly resolve the issue as quickly as possible. Additionally, Kommander’s CD component (Flux CD) maintains a separate audit trail in the form of Kubernetes Events, as changes to a Git repository don’t include exactly when those changes were deployed.
- Avoid configuration drift. The scope of manual changes made by operators expands over time. It soon becomes difficult to know which cluster configuration is critical and which is left over from temporary workarounds or live debugging. Over time, changing a project configuration or replicating a deployment to a new environment becomes a daunting task. GitOps supports simple, reproducible deployment to multiple different clusters by having a single source of truth for cluster and application configuration.

That said, there are some cases when live debugging is necessary in order to resolve an incident in the minimum amount of time. In such cases, pull-request-based workflow adds precious time to resolution for critical production outages. Kommander’s CD strategy supports this scenario by letting you disable the auto sync feature. After auto

sync is disabled, Flux will stop synchronizing the cluster state from the GitOps git repository. This lets you use `kubectl`, `helm`, or whichever tool you need to resolve the issue.

8.5.3.2 Continuous Deployment

ENTERPRISE

After installing Kommander and configuring your project and its clusters(see page 502), navigate to the **Continuous Deployment (CD)** tab under your Project. Here you create a GitOps source which is a source code management (SCM) repository hosting the application definition. D2iQ recommends that you create a secret first then create a GitOps source accessed by the secret.

Prerequisites

You must have Kommander installed(see page 351) to run this procedure.

Set up a secret for accessing GitOps

Create a secret that Kommander uses to deploy the contents of your GitOps repository:

> This dialog box creates a `types.kubefed.io/v1beta1, Kind=FederatedSecret` and this is not yet supported by DKP CLI. Use the GUI, as described above, to create a federated secret or create a `FederatedSecret` manifest and apply it to the project namespace. Learn more about FederatedSecrets(see page 473).

Kommander secrets (for CD) can be configured to support any of the following three authentication methods:

- HTTPS Authentication (described above)
- HTTPS self-signed certificates
- SSH Authentication

The following table describes the fields required for each authentication method:

HTTP Auth	HTTPS Auth (Self-signed)	SSH Auth
username	username	identity
password	password	identity.pub
	caFile	known_hosts

If you are using GitOps by using a GitHub repo as your source, you can create your secret with a personal access token[403]. Then, in the DKP UI, in your project, create a Secret, with a key:value pair of `password` : `<your-token-created-on-github>`. If you are using a GitHub personal access token, you do not need to have a key:value pair of `username` : `<your-github-username>`.

403 https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token

If you are using a secret with your GitHub username and your password, you will need one secret created in the DKP UI, with key:value pairs of `username`: `<your-github-username>` and `password`: `<your-github-password>`. **Note:** if you have multi-factor authentication turned on in your GitHub account, this will not work.

Using a token without a username is valid for GitHub, but other providers (such as GitLab) require both username and tokens.

If you are using a public GitHub repository, you do not need to use a secret.

Create GitOps Source

After the secret is created, you can view it in the `Secrets` tab. Configure the GitOps source accessed by the secret.

If using an SSH secret, the SCM repo URL needs to be an SSH address. It does not support SCP syntax. The URL format is `ssh://user@host:port/org/repository`.

It takes a few moments for the GitOps Source to be reconciled and the manifests from the SCM repository at the given path to be federated to attached clusters. After the sync is complete, manifests from GitOps source are created in attached clusters.

After a GitOps Source is created, there are various commands that can be executed from the CLI to check the various stages of syncing the manifests.

On the management cluster, check your `GitopsRepository` to ensure that the `CD` manifests have been created successfully.

```
kubectl describe gitopsrepositories.dispatch.d2iq.io -n<PROJECT_NAMESPACE> gitopsdemo
```

```
Name:         gitopsdemo
Namespace:    <PROJECT_NAMESPACE>
...
Events:
  Type    Reason               Age    From                         Message
  ----    ------               ----   ----                         -------
  Normal  ManifestSyncSuccess  1m7s  GitopsRepositoryController  manifests synced to
bootstrap repo
...
```

On the attached cluster, check for your `Kustomization` and `GitRepository` resources. The `status` field reflects the syncing of manifests:

```
kubectl get kustomizations.kustomize.toolkit.fluxcd.io -n<PROJECT_NAMESPACE>
<GITOPS_SOURCE_NAME> -oyaml
```

```
...
status:
  conditions:
  - reason: ReconciliationSucceeded
    status: "True"
    type: Ready
    ...
...
```

Similarly, with `GitRepository` resource:

```
kubectl get gitrepository.source.toolkit.fluxcd.io -n<PROJECT_NAMESPACE>
<GITOPS_SOURCE_NAME> -oyaml
```

```
...
status:
  conditions:
  - reason: GitOperationSucceed
    status: "True"
    type: Ready
    ...
...
```

If there are errors creating the manifests, those events are populated in the status field of the `GitopsRepository` resource on the management cluster, the `GitRepository` and `Kustomization` resources on the attached cluster(s), or both.

Suspend GitOps Source

There may be times when you need to suspend the auto-sync between the GitOps repository and the associated clusters. This *live debugging* may be necessary to resolve an incident in the minimum amount of time without the overhead of pull request based workflows.

To Suspend the GitOps Source from the DKP UI:

1. From the top menu bar, select your target workspace.
2. Select **Projects** from the sidebar menu.
3. Select your project from the list.
4. Select the **Continuous Deployment (CD)** tab.
5. Select the three dot button to the right of the desired GitOps Source.
6. Select **Suspend** to manually suspend the GitOps reconciliation.

This lets you use `kubectl`, `helm`, or another tool to resolve the issue. After the issue is resolved select **Resume** to sync the updated contents of the GitOps source to the associated clusters.

Similar to **Suspend/Resume**, you can use the **Delete** action to remove the GitOps source. Removing the GitOps source results in removal of all the manifests applied from the GitOps source.

You can have more than one GitOps Source in your Project to deploy manifests from various sources.

Kommander deployments are backed by FluxCD. Please refer to Flux Source Controller[404] and Kustomize controller[405] docs for advanced configuration and more examples.

8.5.3.3 Project Deployments Troubleshooting

ENTERPRISE

- Events related to federation are stored in respective `FederatedGitRepository`, `FederatedKustomization`, or both resources.
- View the events and logs for `deployments/Kommander-repository-controller` in Kommander namespace, if there are any unexpected errors.
- Enabling the Kommander repository controller for your project namespace causes a number of related Flux controller components[406] to deploy into the namespace. These are necessary for the proper operation of the repository controller and should not be removed.
- Ensure your GitOps repository does not contain any manifests that are cluster-scoped - for example, `Namespace`, `ClusterRole`, `ClusterRoleBinding`, etc. All of the manifests must be namespace-scoped.
- Ensure your GitOps repository does not contain any `HelmRelease` and `Kustomization` resources that are targeting a different namespace than the project namespace.

8.5.3.4 View Helm Releases

ENTERPRISE

View Helm Releases for Continuous Deployments

In addition to viewing the current GitOps Sources, you can also view the current Helm Releases that have been deployed.

1. From the top menu bar, select your target workspace.
2. Select **Projects** from the sidebar menu.
3. Select your project from the list.
4. Select the **Continuous Deployment (CD)** tab.
5. Select the **Helm Releases** button.

All of the current Helm Release charts are displayed with their Chart Version and the names of the clusters. In this example *daily* is the name of the current cluster being displayed.

8.5.4 Project Roles

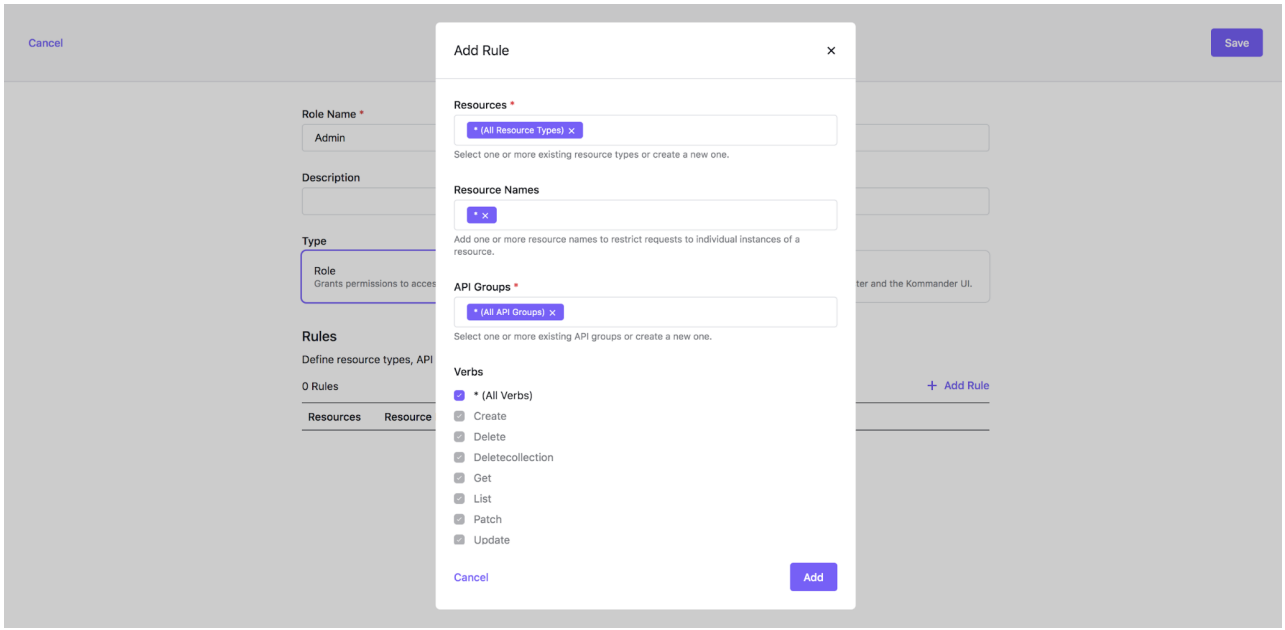
ENTERPRISE

Project Roles are used to define permissions at the namespace level.

404 https://fluxcd.io/docs/components/source/
405 https://fluxcd.io/docs/components/kustomize/
406 https://toolkit.fluxcd.io/components/

8.5.4.1 Configure Project Role - UI Method

In the example below, a Project Role is created with a single Rule. This Project Role corresponds to an admin role.



You can create a Project Role with several Rules.

8.5.4.2 Configure Project Role - CLI Method

The same Project Role can also be created using kubectl:

```
cat << EOF | kubectl create -f -
apiVersion: workspaces.kommander.mesosphere.io/v1alpha1
kind: ProjectRole
metadata:
  annotations:
    kommander.mesosphere.io/display-name: Admin
  generateName: admin-
  namespace: ${projectns}
spec:
  rules:
  - apiGroups:
    - '*'
    resources:
    - '*'
    verbs:
    - '*'
EOF
```

Ensure the `projectns` variable is set before executing the command.

You can set it using the following command (for a Kommander Project called `project1`, and after setting the `workspacens` as explained in the previous section):

```
projectns=$(kubectl -n ${workspacens} get projects.workspaces.kommander.mesosphere.io
-o jsonpath='{.items[?
(@.metadata.generateName=="project1-")].status.namespaceRef.name}')
```

When a Project Role is created, Kommander creates a Kubernetes `FederatedRole` on the Kubernetes cluster where Kommander is running:

```
kubectl -n ${projectns} get federatedroles.types.kubefed.io admin-dbfpj-l6s9g -o yaml
apiVersion: types.kubefed.io/v1beta1
kind: FederatedRole
metadata:
  creationTimestamp: "2020-06-04T11:54:26Z"
  finalizers:
  - kubefed.io/sync-controller
  generation: 1
  name: admin-dbfpj-l6s9g
  namespace: project1-5ljs9-lhvjl
  ownerReferences:
  - apiVersion: workspaces.kommander.mesosphere.io/v1alpha1
    blockOwnerDeletion: true
    controller: true
    kind: ProjectRole
    name: admin-dbfpj
    uid: e5f3b2ca-16bf-474d-8305-7be04c034793
  resourceVersion: "75680"
  selfLink: /apis/types.kubefed.io/v1beta1/namespaces/project1-5ljs9-lhvjl/
federatedroles/admin-dbfpj-l6s9g
  uid: 1e5a3d98-b223-4605-bba1-16276a3eb47c
spec:
  placement:
    clusterSelector: {}
  template:
    rules:
    - apiGroups:
      - '*'
      resourceNames:
      - '*'
      resources:
      - '*'
      verbs:
      - '*'
status:
  clusters:
  - name: konvoy-5nr5h
  conditions:
  - lastTransitionTime: "2020-06-04T11:54:26Z"
```

```
    lastUpdateTime: "2020-06-04T11:54:26Z"
    status: "True"
    type: Propagation
  observedGeneration: 1
```

Then, if you run the following command on a Kubernetes cluster associated with the Project, you see a Kubernetes Role object in the corresponding namespace:

```
kubectl -n ${projectns} get role admin-dbfpj-l6s9g -o yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  creationTimestamp: "2020-06-04T11:54:26Z"
  labels:
    kubefed.io/managed: "true"
  name: admin-dbfpj-l6s9g
  namespace: project1-5ljs9-lhvjl
  resourceVersion: "29218"
  selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/project1-5ljs9-lhvjl/roles/
admin-dbfpj-l6s9g
  uid: f05b998c-4649-4e73-bbfe-c12bc4c86a3c
rules:
- apiGroups:
  - '*'
  resourceNames:
  - '*'
  resources:
  - '*'
  verbs:
  - '*'
```

8.5.5 Project Role Bindings

ENTERPRISE

Project Role Bindings grant access to a specified Project Role for a specified group of people

8.5.5.1 Configure Project Role Bindings - UI Method

Before you can create a Project Role Binding, ensure you have created a Group. A Kommander Group can contain one or several Identity Provider users or groups.

You can assign a role to this Kommander Group:

1. From the Projects page, select your project.
2. Select the Role Bindings tab, then select Add Roles next to the group you want.
3. Select the Role, or Roles, you want from the drop-down menu, and then select Save.

8.5.5.2 Configure Project Role Bindings - CLI Method

A Project Policy can also be created using kubectl:

```
cat << EOF | kubectl create -f -
apiVersion: workspaces.kommander.mesosphere.io/v1alpha1
kind: VirtualGroupProjectRoleBinding
metadata:
  generateName: projectpolicy-
  namespace: ${projectns}
spec:
  projectRoleRef:
    name: ${projectrole}
  virtualGroupRef:
    name: ${virtualgroup}
EOF
```

Configure Project Role Bindings to Bind to WorkspaceRoles - CLI Method

You can also create a Project Policy to bind to a WorkspaceRole in certain instances. To list the WorkspaceRoles that you can bind to a Project, run the following command:

```
kubectl get workspaceroles -n ${workspacens} -o=jsonpath='{.items[?
(@.metadata.annotations.workspace\.kommander\.d2iq\.io\/project-default-workspace-
role-for=="${projectns}")].metadata.name}'
```

You can bind to any of the above WorkspaceRoles by setting `spec.workspaceRoleRef` in the project policy:

```
cat << EOF | kubectl create -f -
apiVersion: workspaces.kommander.mesosphere.io/v1alpha1
kind: VirtualGroupProjectRoleBinding
metadata:
  generateName: projectpolicy-
  namespace: ${projectns}
spec:
  workspaceRoleRef:
    name: ${workspacerole}
  virtualGroupRef:
    name: ${virtualgroup}
EOF
```

Note that you must specify either workspaceRoleRef or projectRoleRef to be validated by the admission webhook. Specifying both values is not valid and will cause an error.

Ensure the `projectns`, `workspacens`, `projectrole` (or `workspacerole`) and the `virtualgroup` variables are set before executing the command.

You can set them using the following commands (for a Kommander Group called `user1` and a Project Role called `admin`, and after setting the `projectns` as explained in the previous section):

```
virtualgroup=$(kubectl -n kommander get virtualgroup.kommander.mesosphere.io -o
jsonpath='{.items[?(@.metadata.generateName=="user1-")].metadata.name}')

projectns=$(kubectl -n ${workspacens} get projects.workspaces.kommander.mesosphere.io
-o jsonpath='{.items[?
(@.metadata.generateName=="project1-")].status.namespaceRef.name}')

projectrole=$(kubectl -n ${projectns} get
projectroles.workspaces.kommander.mesosphere.io -o jsonpath='{.items[?
(@.metadata.generateName=="admin-")].metadata.name}')
```

When a Project Role Binding is created, Kommander creates a Kubernetes `FederatedRoleBinding` on the Kubernetes cluster where Kommander is running:

```
kubectl -n ${projectns} get federatedrolebindings.types.kubefed.io projectpolicy-
gtct4-rdkwq -o yaml
apiVersion: types.kubefed.io/v1beta1
kind: FederatedRoleBinding
metadata:
  creationTimestamp: "2020-06-04T16:19:27Z"
  finalizers:
  - kubefed.io/sync-controller
  generation: 1
  name: projectpolicy-gtct4-rdkwq
  namespace: project1-5ljs9-lhvjl
  ownerReferences:
  - apiVersion: workspaces.kommander.mesosphere.io/v1alpha1
    blockOwnerDeletion: true
    controller: true
    kind: VirtualGroupProjectRoleBinding
    name: projectpolicy-gtct4
    uid: 19614de2-4593-433e-82fa-96dc9470e07a
  resourceVersion: "196270"
  selfLink: /apis/types.kubefed.io/v1beta1/namespaces/project1-5ljs9-lhvjl/
federatedrolebindings/projectpolicy-gtct4-rdkwq
  uid: beaffc29-edec-4258-9813-3a17ba27a2a6
spec:
  placement:
    clusterSelector: {}
  template:
    roleRef:
      apiGroup: rbac.authorization.k8s.io
      kind: Role
      name: admin-dbfpj-l6s9g
    subjects:
    - apiGroup: rbac.authorization.k8s.io
```

```
      kind: User
      name: user1@d2iq.lab
status:
  clusters:
  - name: konvoy-5nr5h
  conditions:
  - lastTransitionTime: "2020-06-04T16:19:27Z"
    lastUpdateTime: "2020-06-04T16:19:27Z"
    status: "True"
    type: Propagation
  observedGeneration: 1
```

Then, if you run the following command on a Kubernetes cluster associated with the Project, you'll see a Kubernetes RoleBinding Object, in the corresponding namespace:

```
kubectl -n ${projectns} get rolebinding projectpolicy-gtct4-rdkwq -o yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  creationTimestamp: "2020-06-04T16:19:27Z"
  labels:
    kubefed.io/managed: "true"
  name: projectpolicy-gtct4-rdkwq
  namespace: project1-5ljs9-lhvjl
  resourceVersion: "125392"
  selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/project1-5ljs9-lhvjl/
rolebindings/projectpolicy-gtct4-rdkwq
  uid: 2938398d-437b-4f3a-9cb9-c92e50139196
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: admin-dbfpj-l6s9g
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: user1@d2iq.lab
```

8.5.6 Project ConfigMaps

ENTERPRISE

Use ConfigMaps to automate ConfigMaps aretion on your clusters

Project ConfigMaps can be created to make sure Kubernetes ConfigMaps are automatically created on all Kubernetes clusters associated with the Project, in the corresponding namespace.

As reference, a ConfigMap is a key-value pair to store some type of non-confidential data like "name=bob" or "state=CA". For a full reference to the concept, consult the Kubernetes documentation on the topic of ConfigMaps[407].

407 https://kubernetes.io/docs/concepts/configuration/configmap/

8.5.6.1 Configuring Project ConfigMaps - UI Method

The below Project ConfigMap form can be navigated to by:

1. From the top menu bar, select your target workspace.
2. Select **Projects** from the sidebar menu.
3. Select your project from the list.
4. Select the **ConfigMaps** tab to browse the deployed ConfigMaps.
5. Select **+ Create ConfigMap** button.
6. Enter an ID, Description and Data for the ConfigMap, and select the **Create** button.

8.5.6.2 Configuring Project ConfigMaps - CLI Method

A Project ConfigMap is simply a Kubernetes FederatedConfigMap and can be created using kubectl with YAML:

```
cat << EOF | kubectl create -f -
apiVersion: types.kubefed.io/v1beta1
kind: FederatedConfigMap
metadata:
  generateName: cm1-
  namespace: ${projectns}
spec:
  placement:
    clusterSelector: {}
  template:
    data:
      key: value
EOF
```

Ensure the `projectns` variable is set before executing the command. This variable is the project namespace (the Kubernetes Namespace associated with the project) that was defined/created when the project itself was initially created.

```
projectns=$(kubectl -n ${workspacens} get projects.workspaces.kommander.mesosphere.io
-o jsonpath='{.items[?
(@.metadata.generateName=="project1-")].status.namespaceRef.name}')
```

Then, if you run the following command on a Kubernetes cluster associated with the Project, you'll see a Kubernetes ConfigMap Object, in the corresponding namespace:

```
kubectl -n ${projectns} get configmap cm1-8469c -o yaml
apiVersion: v1
data:
  key: value
kind: ConfigMap
metadata:
  creationTimestamp: "2020-06-04T16:37:10Z"
  labels:
```

```
    kubefed.io/managed: "true"
  name: cm1-8469c
  namespace: project1-5ljs9-lhvjl
  resourceVersion: "131844"
  selfLink: /api/v1/namespaces/project1-5ljs9-lhvjl/configmaps/cm1-8469c
  uid: d32acb98-3d57-421f-a677-016da5dab980
```

8.5.7 Project Secrets

ENTERPRISE

Project Secrets can be created to make sure a Kubernetes Secrets are automatically created on all the Kubernetes clusters associated with the Project, in the corresponding namespace.

8.5.7.1 Configure Project Secrets - UI Method

1. Select the workspace your project was created in from the workspace selection dropdown in the header.
2. In the sidebar menu, select **Projects**.
3. Select the project you want to configure from the table.
4. Select the **Secrets** tab, and then select the **Create Secret** button.
5. Complete the form and select the **Create** button.

8.5.7.2 Configure Project Secrets - CLI Method

A Project Secret is simply a Kubernetes FederatedConfigSecret and can also be created using kubectl:

```
cat << EOF | kubectl create -f -
apiVersion: types.kubefed.io/v1beta1
kind: FederatedSecret
metadata:
  generateName: secret1-
  namespace: ${projectns}
spec:
  placement:
    clusterSelector: {}
  template:
    data:
      key: dmFsdWU=
EOF
```

Ensure the `projectns` variable is set before executing the command.

```
projectns=$(kubectl -n ${workspacens} get projects.workspaces.kommander.mesosphere.io
-o jsonpath='{.items[?
(@.metadata.generateName=="project1-")].status.namespaceRef.name}')
```

The value of the key is base64 encoded.

If you run the following command on a Kubernetes cluster associated with the Project, you see a Kubernetes Secret Object, in the corresponding namespace:

```
kubectl -n ${projectns} get secret secret1-r9vk2 -o yaml
apiVersion: v1
data:
  key: dmFsdWU=
kind: Secret
metadata:
  creationTimestamp: "2020-06-04T16:51:59Z"
  labels:
    kubefed.io/managed: "true"
  name: secret1-r9vk2
  namespace: project1-5ljs9-lhvjl
  resourceVersion: "137215"
  selfLink: /api/v1/namespaces/project1-5ljs9-lhvjl/secrets/secret1-r9vk2
  uid: e5c6fc1d-93e7-47fe-ae1e-f418f8e35d72
type: Opaque
```

8.5.8 Project Quotas & Limit Ranges

ENTERPRISE

Project Quotas and Limit Ranges can be set up to limit the number of resources the Project team uses.

8.5.8.1 Creating Project Quotas & Limit Ranges- UI Method

Project Quotas and Limit Ranges can be set up to limit the number of resources the Project team uses. Quotas and Limit Ranges are applied to all project clusters.

1. Select the workspace your project was created in from the workspace selection dropdown in the header.
2. In the sidebar menu, select **Projects**.
3. Select the project you want to configure from the table.
4. Select the **Quotas & Limit Ranges** tab, and then select the **Edit** button.
   Kommander provides a set of default resources for which you can set Quotas. You can also define Quotas for custom resources. We recommend that you set Quotas for CPU and Memory. By using Limit Ranges, you can restrict the resource consumption of individual Pods, Containers, and Persistent Volume Claims in the project namespace. You can also constrain memory and CPU resources consumed by Pods and Containers, and storage resources consumed by Persistent Volume Claims.
5. To add a custom quota, scroll to the bottom of the form and select **Add Quota**.
6. When you are finished, select the **Save** button.

8.5.8.2 Create Project Quotas & Limit Ranges - CLI Method

All the Project Quotas are defined using a Kubernetes FederatedResourceQuota called `kommander` which you can also create/update using kubectl:

```
cat << EOF | kubectl apply -f -
apiVersion: types.kubefed.io/v1beta1
kind: FederatedResourceQuota
metadata:
  name: kommander
  namespace: ${projectns}
spec:
  placement:
    clusterSelector: {}
  template:
    spec:
      hard:
        limits.cpu: "10"
        limits.memory: 1024.000Mi
EOF
```

Ensure the `projectns` variable is set before executing the command.

```
projectns=$(kubectl -n ${workspacens} get projects.workspaces.kommander.mesosphere.io
-o jsonpath='{.items[?
(@.metadata.generateName=="project1-")].status.namespaceRef.name}')
```

Then, if you run the following command on a Kubernetes cluster associated with the Project, you'll see a Kubernetes Secret Object in the corresponding namespace:

```
kubectl -n ${projectns} get resourcequota kommander -o yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  creationTimestamp: "2020-06-05T08:04:37Z"
  labels:
    kubefed.io/managed: "true"
  name: kommander
  namespace: project1-5ljs9-lhvjl
  resourceVersion: "470822"
  selfLink: /api/v1/namespaces/project1-5ljs9-lhvjl/resourcequotas/kommander
  uid: 925b61b4-134b-4c45-915c-96a05b63d3c3
spec:
  hard:
    limits.cpu: "10"
    limits.memory: 1Gi
status:
  hard:
    limits.cpu: "10"
    limits.memory: 1Gi
  used:
    limits.cpu: "0"
    limits.memory: "0"
```

Similarly, Project Limit Ranges are defined using a FederatedLimitRange object with name `kommander` in the project namespace:

```
cat << EOF | kubectl apply -f -
apiVersion: types.kubefed.io/v1beta1
kind: FederatedLimitRange
metadata:
  name: kommander
  namespace: ${projectns}
spec:
  placement:
    clusterSelector: {}
  template:
    spec:
      limits:
      - type: "Pod"
        max:
          cpu: 500m
          memory: 50Gi
        min:
          cpu: 100m
          memory: 10Gi
      - type: "Container"
        max:
          cpu: 2
          memory: 100Mi
        min:
          cpu: 1
          memory: 10Mi
      - type: "PersistentVolumeClaim"
        max:
          storage: 3Gi
        min:
          storage: 1Gi
EOF
```

8.5.9 Project Network Policies

ENTERPRISE

Projects are created with a secure-by-default network policy and users needing more flexibility can edit or add more policies to tailor to their unique security needs.

Cluster networking is a critical and central part of Kubernetes that can also be quite challenging. All network communication within and between clusters depends on the presence of a Container Network Interface (CNI) plugin.

8.5.9.1 About Network Plugins

Network plugins ensure that Kubernetes networking requirements are met and surface features needed by network administrators, such as enforcing network policies. Common Network Plugins include Flannel, Calico, and Weave among many others. As an example, D2iQ® Konvoy® uses the Calico CNI plugin by default, but can support others.

Since pods are short-lived, the cluster needs a way to configure the network dynamically as pods are created and destroyed. Plugins provision and manage IP addresses to interfaces and let administrators manage IPs and their assignments to containers, in addition to connections to more than one host, when needed.

8.5.9.2 About Network Policies

By default, and to enable fluid communications within and between clusters, all traffic is authorized between nodes and pods. Most production environments require some kind of traffic flow control at the IP address or port level. An application-centric approach to this uses Network Policies. Pods are isolated by having a Network Policy that selects them, and the configuration of the policy limits the kind of traffic they can receive or send. Network policies do not conflict because they are additive. Pods selected by more than one policy are subject to the union of the policies' ingress and egress rules.

A Network Policy's rules define ingress and egress for network communications between pods and across namespaces. Successful traffic control using network policies is bi-directional. You have to configure both the egress policy on the source pod and the ingress policy on the destination pod to enable the traffic. If either end denies the traffic, it will not flow between the pods.

Since the Kubernetes default is to allow all traffic, it is a common practice to create a default "deny all traffic" rule, and then specifically open up some combination of the pods, ports, and applications as needed.

8.5.9.3 Creating Network Policies

You create network policies in three main parts:

- General information
- Ingress rules
- Egress rules

General information section

The fields in this part of the form allow you to create a name and description for this policy. Creating a detailed **Description** helps to keep policy functions understandable for additional use and maintenance.

This section also contains the **Pod Selector** fields for selecting pods using either Labels or Expressions. **Labels** added to pod declarations are a common means of identifying individual pods, or creating groups of pods, in a Kubernetes cluster. **Expressions** are similar to Labels, but allow you to define parameters that identify a range of pods.

The **Policy Types** selections help to define the type of Network Policy you are creating:

- **Default** - automatically includes ingress, and egress is set only if the network policy defines egress rules.
- **Ingress** - this policy applies to ingress traffic for the selected pods, to namespaces using the options you define below, or both.
- **Egress** - this policy applies to egress traffic for the selected pods, to namespaces using the options you define below, or both.

If the Default policy type is too rigid or does not offer what you need, you can select the Ingress or Egress type, or both, and explicitly define the policy with the options that follow. For example, if you do not want this policy to apply to ingress traffic, you would select only Egress, and then define the policy.

To deny all ingress traffic, select the **Ingress** option here and then leave the ingress rules empty.

To deny all egress traffic, select the **Egress** option here and then leave the egress rules empty.

Ingress rules section

Ingress rules use a combination of **Port** / **Protocol** and **Source** to define the incoming traffic allowed to some or all of the pods in this namespace.

The options under **Sources: From** enable you to define a source either by using the pod selector or by defining an IP block. When using the pod selector method, you can define the namespace, the pods within that namespace, or both.

Namespaces - Selecting a namespace in an ingress rule source permits the pods selected by the pod selector, in your selected namespaces, to receive incoming traffic that meets the other defined criteria. If you have not selected any pods, the rule permits traffic from all pods in the selected namespaces.

Pods - This option selects specific Pods which should be allowed as ingress sources or egress destinations. If you have not selected any namespaces in the namespace selector, this option selects all matching pods in the project namespace. Otherwise, this option selects all matching pods in the selected namespaces.

There also are options to select all namespaces, all pods, or both.

When defining ingress rules using the IP Block method, you define a CIDR and exception conditions. CIDR stands for Classless Inter-Domain Routing and is an IP standard for creating unique network and device identifiers. When grouped together so that they share an initial sequence of bits in their binary representation, the range of addresses creates a CIDR block. The block identity is in an IPv4-like notation including a dotted-decimal address, followed by a slash, then a colon and a number from 0 through 32, for example, 127.0.26.33:31.

Egress rules section

Egress rules use a similar combination of options to define the outgoing traffic from pods, ranges of pods, or namespaces in a Kommander Project. Port, Protocol, and Destination options for egress rules define the outgoing traffic. You can define your egress rules under **Destination: To**. Ensure the egress policy on the source pods, and the ingress policy on the destination pods, permit traffic in order for the pods to be able to communicate over the network.

8.5.9.4 Network Policy Examples

IMPORTANT: Before you begin each example, ensure you're on the Network Policy page for your project

To navigate to your project's Network Policy page:

1. From the top menu bar, select your target workspace.
2. Select **Projects** from the sidebar menu.
3. Select your project from the list.
4. Select the **Network Policy** tab.

Ingress: Permit access to API service pods from all namespaces

Suppose you need to create a network policy to permit incoming traffic to API service pods in a specific Kommander Project's namespace from any other pod in any namespace that has the label, `service.corp/users-api-`

`role: client`. For this example, API service pods are those pods created with the Label, `service.corp/users-api-role: api`.

You can limit the policy to just incoming traffic from select namespaces by adding an ingress rule with these characteristics:

- Use Port 8080 to receive incoming TCP traffic
- Refuse traffic from pods unless they are client pods that have a specific Label, such as `service.corp/users-api-role: client`. This example follows a common microservice architecture pattern, microservice-tier-role: access_mode

Configure the general information to access API service pods

1. Select the **+ Create Network Policy** button.
2. Type “microsvc-users-api-allow” in the **ID Name** field.
3. Type “Allow Users microservice clients to reach the APIs provided in this namespace” in the **Description** field.
4. Select **Add** under **Pod Selector** and then select **Match Label**.
5. Set the **Key** to “service.corp/users-api-role” and the **Value** to “API”.

Create an Ingress rule to access API service pods

1. Leave **Policy Types** set to Default.
2. Scroll down to **Ingress Rules** and select **+ Add an Ingress Rule**.
3. Select **+ Add Port**, and set the **Port** to 8080 and the **Protocol** to TCP.

Add Sources to access API service pods

1. Select **+ Add Source** and mark the **Select All Namespaces** check box.
2. Select **+ Add Pod Selector**.
3. Select **Match Label**.
4. Set the **Key** value to “service.corp/users-api-role” and set the **Value** to “client”.
5. Scroll up and select **Save**.

Ingress: Limit pods that access a database to this namespace

Suppose that while deploying an application in a project, you want to protect its database pods by permitting ingress only from API service pods in the current namespace, and prevent ingress from pods in any other namespace.

You can limit the database pods to just the incoming traffic from the current namespaces by adding an ingress rule with these characteristics:

- Use Port 3306 to receive incoming TCP traffic for pods that have the label, `tier: database`
- Refuse traffic from pods unless they have the label, `tier: api`

Configure the general information to access a database

1. Select the **+ Create Network Policy** button.
2. Type “database-access-api-only” in the **ID name** field.
3. Type “Allow MySQL access only from API pods in this namespace” in the **Description** field.
4. Select **Pod Selector** then select **Match Label**.

5. Set the **Key** to “tier” and the **Value** to “database”.

Create an Ingress rule to access a database

1. Select Default for the **Policy Types**.
2. Scroll down to the **Ingress** section.
3. Select **+ Add Ingress Rule**, then select **+ Add Port**.
4. Set the **Port** to “3306” and leave the **Protocol** set to “TCP”.

Add Sources to access a database

1. Select **+ Add Source**.
2. Select **+ Add Pod Selector**.
3. Select **Match Label** and set the **Key** to “tier” and the **Value** to “API”.
4. Scroll up and select **Save**.

Ingress: Disable but don’t delete ingress rules

Suppose that you want to disable ingress rules temporarily for testing or triaging purposes.

First, you need to create a network policy with one or more ingress rules. You could follow one of the preceding procedures, for example. Then, you need to edit the policy to match the following example:

Edit your Network Policy

1. In the table row belonging to your network policy, click the context menu at the right of the row and select **Edit**.

Disable Ingress rules

1. Update the **Policy Types** so that only **Egress** is selected. If you don’t want to deny *all* egress traffic, ensure that you add an egress rule that suits your preferred level of access. You can add an empty rule to allow all egress traffic.
2. Scroll up and select **Save**.

Egress: Deny all egress traffic from restricted pods

Suppose that you need to deny all egress traffic from a group of restricted pods. This is a simple egress rule and you can create it following this example and steps:

Configure the General Information to deny Egress

1. Select **+ Add Network Policy**.
2. Type “deny-restricted-egress” in the **ID name** field.
3. Type “Deny egress traffic from restricted pods” in the **Description** field.
4. Select **Pod selector** then select **Match Label.**
5. Set the **Key** to “access” and the **Value** to “restricted”.

Deny Egress traffic

1. Update the **Policy Types** so that only **Egress** is selected. Do not add any egress rules.
2. Scroll up and select **Save**.

8.5.10 Project Applications

ENTERPRISE

This section documents the applications and application types that you can utilize with DKP.

Application types are:

- Catalog Applications(see page 486) are either pre-packaged applications from the D2iQ Application Catalog, or custom applications that you maintain for your teams or organization.
  - DKP Applications(see page 487) are applications that are provided by D2iQ and added to the Catalog.
  - Custom Applications(see page 494) are applications you create and add to the Catalog.
- Platform Applications(see page 481) are applications integrated into Kommander.

When deploying and upgrading applications, platform applications come as a bundle; they are tested as a single unit and you must deploy or upgrade them in a single process, for each workspace. This means all clusters in a workspace have the same set and versions of platform applications deployed. Whereas catalog applications are individual, so you can deploy and upgrade them individually, for each project.

8.5.10.1 Project Platform Applications

ENTERPRISE

How project Platform applications work

The following table describes the list of applications that can be deployed to attached clusters within a project.

Review the Project Platform Application Requirements(see page 485) to ensure that the attached clusters in the project have sufficient resources.

From within a project, you can enable applications to deploy. Verify that an application has successfully deployed via the CLI(see page 424).

Enable applications in a project using the DKP UI

1. From the top menu bar, select your target workspace.
2. Select **Projects** from the sidebar menu.
3. Select your project from the list.
4. Select the **Applications** tab to browse the available applications.
5. Select the three dot button from the bottom-right corner of the desired application tile, and then select **Enable**.
6. To override the default configuration values, copy your values content into the text editor under **Configure Service** or just upload your yaml file that contains the values:

   ```
   someField: someValue
   ```

7. Select the **Enable** button.

To use the CLI to enable or disable applications, see Application Deployment(see page 424)

⚠ There may be dependencies between the applications, which are listed in Project Platform Application Dependencies(see page 484). Review them carefully prior to customizing to ensure that the applications are deployed successfully.

Platform Applications

NAME	APP ID	Deployed by default
project-grafana-logging-6.28.0	project-grafana-logging	False
project-grafana-loki-0.48.4	project-grafana-loki	False
project-logging-1.0.0	project-logging	False

Upgrade Platform Applications from the CLI

Platform Applications within a Project are automatically upgraded when the Workspace that a Project belongs to is upgraded. See Upgrade Kommander(see page 782) for more information on how to upgrade these applications.

Project Platform Application Deployment

ENTERPRISE

Deploy applications to attached clusters in a project using the CLI

This topic describes how to use the CLI to deploy an application to attached clusters within a project. To use the DKP UI to deploy applications, see Deploy Applications in a Project(see page 481).

See Project Platform Applications(see page 481) for a list of all applications and those that are enabled by default.

Prerequisites

Before you begin, you must have:

- A running cluster with Kommander installed(see page 97).
- An existing Kubernetes cluster attached to Kommander(see page 505).

Set the `WORKSPACE_NAME` environment variable to the name of the workspace where the cluster is attached:

```
export WORKSPACE_NAME=<workspace_name>
```

Set the `WORKSPACE_NAMESPACE` environment variable to the namespace of the above workspace:

```
export WORKSPACE_NAMESPACE=$(kubectl get namespace --
selector='workspaces.kommander.mesosphere.io/workspace-name=${WORKSPACE_NAME}' -o
jsonpath='{.items[0].metadata.name}')
```

Set the `PROJECT_NAME` environment variable to the name of the project in which the cluster is included:

```
export PROJECT_NAME=<project_name>
```

Set the `PROJECT_NAMESPACE` environment variable to the name of the above project's namespace:

```
export PROJECT_NAMESPACE=$(kubectl get project ${PROJECT_NAME} -n $
{WORKSPACE_NAMESPACE} -o jsonpath='{.status.namespaceRef.name}')
```

Deploy the Application

The list of available applications that can be deployed on the attached clusters in a project can be found in the project platform applications topic(see page 481).

1. Deploy one of the supported applications to your existing attached cluster(see page 505) with an `AppDeployment` resource. Provide the `appRef` and application version to specify which `App` is deployed:

   ```
   dkp create appdeployment project-grafana-logging --app project-grafana-
   logging-6.28.0 --workspace ${WORKSPACE_NAME} --project ${PROJECT_NAME}
   ```

2. Create the resource in the project you just created, which instructs Kommander to deploy the `AppDeployment` to the `KommanderClusters` in the same project.

- The `appRef.name` must match the app `name` from the list of available applications.
- Observe that the `dkp create` command must be run with both the `--workspace` and `--project` flags for project platform applications.

Deploy an Application with a Custom Configuration

Follow these steps:

1. Create the `AppDeployment` and provide the name of a `ConfigMap`, which provides custom configuration on top of the default configuration:

   ```
   dkp create appdeployment project-grafana-logging --app project-grafana-
   logging-6.28.0 --config-overrides project-grafana-logging-overrides --workspace
   ${WORKSPACE_NAME} --project ${PROJECT_NAME}
   ```

2. Create the `ConfigMap` with the name provided in the step above, with the custom configuration:

   ```
   cat <<EOF | kubectl apply -f -
   ```

```
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: ${PROJECT_NAMESPACE}
  name: project-grafana-logging-overrides
data:
  values.yaml: |
    datasources:
      datasources.yaml:
        apiVersion: 1
        datasources:
        - name: Loki
          type: loki
          url: "http://project-grafana-loki-loki-distributed-gateway"
          access: proxy
          isDefault: false
EOF
```

Kommander waits for the `ConfigMap` to be present before deploying the `AppDeployment` to the attached clusters.

Verify Applications

The applications are now enabled. Connect to the attached cluster and check the `HelmReleases` to verify the deployment:

```
kubectl get helmreleases -n ${PROJECT_NAMESPACE}
NAMESPACE               NAME                        READY   STATUS
AGE
project-test-vjsfq      project-grafana-logging     True    Release reconciliation
succeeded   7m3s
```

Some of the supported applications have dependencies on other applications. See Project Application Dependencies[408] for that table.

Project Platform Application Dependencies

ENTERPRISE

Dependencies between project platform applications

There are many dependencies between the applications that are deployed to a project's attached clusters. It is important to note these dependencies when customizing the platform applications to ensure that your services are properly deployed to the clusters. For more information on how to customize platform applications, see Platform Application Deployment(see page 482).

408 https://d2iq.atlassian.net/wiki/pages/createpage.action?spaceKey=DKP&title=Project+Platform+Application+Dependencies

Application Dependencies

When deploying or troubleshooting applications, it helps to understand how applications interact and may require other applications as dependencies.

If an application's dependency does not successfully deploy, the application requiring that dependency does not successfully deploy.

The following sections detail information about the platform applications.

Logging
Collects logs over time from Kubernetes pods deployed in the project namespace. Also provides the ability to visualize and query the aggregated logs.

- project-logging[409]: Defines resources for the Logging Operator which uses them to direct the project's logs to its respective Grafana Loki application.
- project-grafana-loki[410]: A horizontally-scalable, highly-available, multi-tenant log aggregation system inspired by Prometheus.
- project-grafana-logging[411]: Logging dashboard used to view logs aggregated to Grafana Loki.

⚠ The project logging applications depend on the workspace logging applications(see page 562) being deployed.

Application	Dependencies
project-logging	logging-operator (workspace)
project-grafana-loki	project-logging, grafana-loki (workspace), logging-operator (workspace)
project-grafana-logging	project-grafana-loki

Project Platform Application Configuration Requirements

ENTERPRISE

Project Platform Application Descriptions and Resource Requirements

Platform applications require more resources than solely deploying or attaching clusters into a project. Your cluster must have sufficient resources when deploying or attaching to ensure that the applications are installed successfully.

The following table describes all the platform applications that are available to the clusters in a project, minimum resource and persistent storage requirements, and whether they are enabled by default.

409 https://grafana.com/oss/grafana/
410 https://grafana.com/oss/loki/
411 https://grafana.com/oss/grafana/

Name	Minimum Resources Suggested	Minimum Persistent Storage Required	Deployed by Default
project-grafana-logging	cpu: 200m memory: 100Mi		No
project-grafana-loki		# of PVs: 3 PV sizes: 10Gi x 3 (total: 30Gi)	No
project-logging			No

8.5.10.2 Project Catalog Applications

ENTERPRISE

Catalog applications are any third-party or open source applications that appear in the Catalog. These can be DKP applications provided by D2iQ for use in your environment, or Custom Applications(see page 494) that can be used but are not supported by D2iQ.

Upgrade Project Catalog Applications

ENTERPRISE

Before upgrading your catalog applications, verify the current and supported versions of the application. Also, keep in mind the distinction between Platform applications and Catalog applications. Platform applications are deployed and upgraded as a set for each cluster or workspace. Catalog applications are deployed separately, so that you can deploy and upgrade them individually for each project.

Catalog applications must be upgraded to the latest version BEFORE upgrading the Kubernetes version (or Konvoy version for managed Konvoy clusters) on attached clusters, due to the previous versions' incompatibility with Kubernetes 1.22.

Upgrade with UI

Follow these steps to upgrade an application from the DKP UI:

1. From the top menu bar, select your target workspace.
2. From the side menu bar, select **Projects**.
3. Select your target project.
4. Select **Applications** from the project menu bar.
5. Select the three dot button from the bottom-right corner of the desired application tile, and then select **Edit**.
6. Select the **Version** drop-down, and select a new version. This drop-down will only be available if there is a newer version to upgrade to.
7. Select **Save**.

Upgrade with CLI

```
dkp upgrade catalogapp <appdeployment-name> --workspace=my-workspace --project=my-
project --to-version=<version.number>
```

Platform applications cannot be upgraded on a one-off basis, and must be upgraded in a single process for each workspace. If you attempt to upgrade a platform application with these commands, you receive an error and the application is not upgraded.

Project-level DKP Applications

ENTERPRISE

DKP applications are catalog applications provided by D2iQ for use in your environment.

Some DKP workspace catalog applications will provision `CustomResourceDefinitions`, which allow you to deploy Custom Resources to a Project. See your DKP workspace catalog application's documentation for instructions.

Kafka in a Project

ENTERPRISE

Deploying Kafka in a project

Getting started
To get started with creating and managing a Kafka Cluster in a project, you first need to deploy the Kafka operator(see page 442) and the ZooKeeper operator(see page 445) in the workspace where the project exists.

After deploying the Kafka operator, create Kafka Clusters by applying a `KafkaCluster` custom resource **on each attached cluster** in a project's namespace. Refer to the Kafka operator repository[412] for examples of the custom resources and their configurations.

Example Deployment

If you need to manage these custom resources across all clusters in a project, it is recommended you use Project Deployments(see page 461) which enables you to leverage GitOps to deploy the resources. Otherwise, you will need to create the custom resources manually in each cluster.

This example deployment walks you through first deploying a ZooKeeper cluster and then a Kafka cluster in a project namespace. The result of this procedure is a running ZooKeeper cluster and Kafka cluster ready for use in your project's namespace.

1. Set the `PROJECT_NAMESPACE` environment variable to the name of your project's namespace:

412 https://github.com/banzaicloud/koperator/tree/master/config/samples

```
export PROJECT_NAMESPACE=<project namespace>
```

2. Create a ZooKeeper Cluster custom resource in your project's namespace:

```
kubectl apply -f - <<EOF
apiVersion: zookeeper.pravega.io/v1beta1
kind: ZookeeperCluster
metadata:
  name: zookeeper
  namespace: ${PROJECT_NAMESPACE}
spec:
  replicas: 1
EOF
```

3. Check the status of your ZooKeeper cluster using `kubectl`:

```
kubectl -n ${PROJECT_NAMESPACE} get zookeeperclusters
```

4. Download the sample Kafka Cluster[413] file.
5. Update the following attribute `zkAddresses`, replacing `zookeeper-client.zookeeper:2181` with `zookeeper-client.<project namespace>:2181`. You can use `sed` to update the file:
   - MacOS sed

```
# If you are using sed that comes installed on macOS
sed -i '' "s/zookeeper-client.zookeeper:2181/zookeeper-client.$
{PROJECT_NAMESPACE}:2181/g" simplekafkacluster.yaml
```

   - GNU sed

```
# If you are using GNU sed
sed -i "s/zookeeper-client.zookeeper:2181/zookeeper-client.$
{PROJECT_NAMESPACE}:2181/g" simplekafkacluster.yaml
```

6. Verify the file contains the following line:

```
...
zkAddresses:
    - “zookeeper-client.<project namespace>:2181”
...
```

7. Apply the `KafkaCluster` to your project's namespace:

```
kubectl apply -n ${PROJECT_NAMESPACE} -f simplekafkacluster.yaml
```

413 https://raw.githubusercontent.com/banzaicloud/koperator/master/config/samples/simplekafkacluster.yaml

8. Check the status of your Kafka cluster using `kubectl`:

```
kubectl -n ${PROJECT_NAMESPACE} get kafkaclusters
```

With both the ZooKeeper cluster and Kafka cluster running in your project's namespace, refer to the Kafka Operator documentation[414] for information on how to test and verify they are working as expected in. When performing those steps, ensure you substitute: `zookeeper-client.<project namespace>:2181` anywhere that the zookeeper client address is mentioned.

Delete Kafka Custom Resources
Follow these steps to delete the Kafka custom resources.

1. View all Kafka resources in the cluster:

```
kubectl get kafkaclusters -A
kubectl get kafkausers -A
Kubectl get kafkatopics -A
```

2. Delete a `KafkaCluster` example:

```
kubectl -n ${PROJECT_NAMESPACE} delete kafkacluster <name of KafkaCluster>
```

Resources

- Kafka Operator Documentation[415]
- Kafka Operator GitHub Repository[416]
- Sample Kafka Operator Custom Resources[417]
- Apache Kafka Documentation[418]

Zookeeper in a Project

ENTERPRISE

Deploying ZooKeeper in a project

Getting started
To get started with creating ZooKeeper clusters in your project namespace, you first need to deploy the ZooKeeper operator(see page 445) in the workspace where the project exists.

After you deploy the ZooKeeper operator, you can create ZooKeeper Clusters by applying a `ZookeeperCluster` custom resource **on each attached cluster** in a project's namespace.

A Helm chart[419] exists in the ZooKeeper operator repository that can assist with deploying ZooKeeper clusters.

Example Deployment

414 https://banzaicloud.com/docs/supertubes/kafka-operator/test/
415 https://banzaicloud.com/docs/supertubes/kafka-operator/
416 https://github.com/banzaicloud/koperator
417 https://github.com/banzaicloud/koperator/tree/master/config/samples
418 https://kafka.apache.org/documentation/
419 https://github.com/pravega/zookeeper-operator/tree/master/charts/zookeeper

If you need to manage these custom resources across all clusters in a project, it is recommended you use Project Deployments(see page 489) which enables you to leverage GitOps to deploy the resources. Otherwise, you will need to create the resources manually in each cluster.

Follow these steps to deploy a ZooKeeper cluster in a project namespace. This procedure results in a running ZooKeeper cluster, ready for use in your project's namespace.

1. Set the `PROJECT_NAMESPACE` environment variable to the name of your project's namespace:

```
export PROJECT_NAMESPACE=<project namespace>
```

2. Create a ZooKeeper Cluster custom resource in your project namespace

```
kubectl apply -f - <<EOF
apiVersion: zookeeper.pravega.io/v1beta1
kind: ZookeeperCluster
metadata:
  name: zookeeper
  namespace: ${PROJECT_NAMESPACE}
spec:
  replicas: 1
EOF
```

3. Check the status of your ZooKeeper cluster using `kubectl`:

```
kubectl get zookeeperclusters -n ${PROJECT_NAMESPACE}
```

```
NAME        REPLICAS   READY REPLICAS    VERSION   DESIRED VERSION   INTERNAL
ENDPOINT    EXTERNAL ENDPOINT   AGE
zookeeper   1          1                0.2.13    0.2.13            10.100.200
.18:2181   N/A                94s
```

Delete ZooKeeper clusters
Follow these steps to delete the Zookeeper clusters.

1. View `ZookeeperClusters` in all namespaces:

```
kubectl get zookeeperclusters -A
```

2. Delete a specific `ZookeeperCluster`:

```
kubectl -n ${PROJECT_NAMESPACE} delete zookeepercluster <name of
zookeepercluster>
```

Resources

- ZooKeeper Operator Documentation[420]
- ZooKeeper Cluster Helm Chart[421]
- ZooKeeper Documentation[422]

Spark in a Project

Deploying Spark in a project

Getting started
To get started with creating and managing Spark workloads in a project, you first need to deploy the Spark Operator(see page 443) in the workspace where the project exists.

After deploying the Spark Operator, apply the Spark Operator specific custom resources. The Spark Operator works with the following kinds of custom resources:

- `SparkApplication`
- `ScheduledSparkApplication`

See Spark Operator API documentation[423] for more details.

Example Deployment

If you need to manage these custom resources and RBAC resources across all clusters in a project, it is recommended you use Project Deployments(see page 461) which enables you to leverage GitOps to deploy the resources. Otherwise, you will need to create the resources manually in each cluster.

This example deployment walks you through deploying a Spark application in a project namespace. The result of this procedure is a running Spark application ready for use in your project's namespace.

1. Create your Project(see page 459) if you don't already have one.
2. Set the `PROJECT_NAMESPACE` environment variable to the name of your project's namespace:

   ```
   export PROJECT_NAMESPACE=<project namespace>
   ```

3. Ensure the necessary RBAC resources referenced in your custom resources exist, otherwise the custom resources can fail. See the Spark Operator documentation[424] for details.
   - This is an example of commands for you to create the RBAC resources needed in your project namespace:

     ```
     kubectl apply -f - <<EOF
     apiVersion: v1
     kind: ServiceAccount
     metadata:
     ```

420 https://github.com/pravega/zookeeper-operator
421 https://github.com/pravega/zookeeper-operator/tree/master/charts/zookeeper
422 https://zookeeper.apache.org/documentation
423 https://github.com/mesosphere/spark-on-k8s-operator/blob/d2iq-master/docs/api-docs.md
424 https://github.com/mesosphere/spark-on-k8s-operator/blob/d2iq-master/docs/quick-start-guide.md#about-the-spark-job-namespace

```
  name: spark-service-account
  namespace: ${PROJECT_NAMESPACE}
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: ${PROJECT_NAMESPACE}
  name: spark-role
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["*"]
- apiGroups: [""]
  resources: ["services"]
  verbs: ["*"]
- apiGroups: [""]
  resources: ["configmaps"]
  verbs: ["*"]
- apiGroups: [""]
  resources: ["persistentvolumeclaims"]
  verbs: ["*"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: spark-role-binding
  namespace: ${PROJECT_NAMESPACE}
subjects:
- kind: ServiceAccount
  name: spark-service-account
  namespace: ${PROJECT_NAMESPACE}
roleRef:
  kind: Role
  name: spark-role
  apiGroup: rbac.authorization.k8s.io
EOF
```

4. Set the `SPARK_SERVICE_ACCOUNT` environment variable to one of the following:
   a. `${PROJECT_NAMESPACE}`, if you skipped the previous step to create RBAC resources.

```
# This service account is automatically created when you create a project
and has access to everything in the project namespace.
export SPARK_SERVICE_ACCOUNT=${PROJECT_NAMESPACE}
```

   b. Or set it to `spark-service-account`

```
export SPARK_SERVICE_ACCOUNT=spark-service-account
```

5. Apply the `SparkApplication` custom resource in your project namespace

```
kubectl apply -f - <<EOF
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: pyspark-pi
  namespace: ${PROJECT_NAMESPACE}
spec:
  type: Python
  pythonVersion: "3"
  mode: cluster
  image: "gcr.io/spark-operator/spark-py:v3.1.1"
  imagePullPolicy: Always
  mainApplicationFile: local:///opt/spark/examples/src/main/python/pi.py
  sparkVersion: "3.1.1"
  restartPolicy:
    type: OnFailure
    onFailureRetries: 3
    onFailureRetryInterval: 10
    onSubmissionFailureRetries: 5
    onSubmissionFailureRetryInterval: 20
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.1.1
    serviceAccount: ${SPARK_SERVICE_ACCOUNT}
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.1.1
EOF
```

Delete Spark custom resources
Follow these steps to delete the Spark custom resources:

1. View `SparkApplications` in all namespaces:

   ```
   kubectl get sparkapp -A
   ```

2. Delete a specific `SparkApplication`:

   ```
   kubectl -n ${PROJECT_NAMESPACE} delete sparkapp <name of sparkapplication>
   ```

Resources

- Spark Operator Documentation[425]
- Spark Operator API Documentation[426]
- Sample Spark Operator Custom Resources[427]
- Apache Spark Documentation[428]

Use Custom Resources with Workspace Catalog Applications

ENTERPRISE

Some workspace catalog applications provision some `CustomResourceDefinition`, which allow you to deploy Custom Resources. Refer to your workspace catalog application's documentation for instructions.

Custom Project Applications

ENTERPRISE

Custom applications are third-party applications you have added to the Kommander Catalog.

Custom applications are any third-party applications that are not provided in the DKP Application Catalog. Custom applications can leverage applications from the DKP Catalog or be fully-customized. There is no expectation of support by D2iQ for a Custom application. Custom applications can be deployed on Konvoy clusters or on any D2iQ supported 3rd party Kubernetes distribution.

Projects - Create a Git Repository

ENTERPRISE

Create a Git Repository in the Project namespace

Use the CLI to create the GitRepository resource and add a new repository to your Project.

1. Refer to air-gapped setup instructions(see page 372), if you are running in air-gapped environment.
2. Set the `PROJECT_NAMESPACE` environment variable to the name of your project's namespace:

```
export PROJECT_NAMESPACE=<project_namespace>
```

3. Adapt the URL of your Git repository.

```
kubectl apply -f - <<EOF
apiVersion: source.toolkit.fluxcd.io/v1beta1
kind: GitRepository
metadata:
  name: example-repo
  namespace: ${PROJECT_NAMESPACE}
spec:
```

425 https://github.com/pravega/zookeeper-operator
426 https://github.com/mesosphere/spark-on-k8s-operator/blob/d2iq-master/docs/api-docs.md
427 https://github.com/mesosphere/spark-on-k8s-operator/tree/d2iq-master/examples
428 https://spark.apache.org/docs/latest/

```
  interval: 1m0s
  ref:
    branch: <your-target-branch-name> # e.g., main
  timeout: 20s
  url: https://github.com/<example-org>/<example-repo>
EOF
```

4. Ensure the status of the `GitRepository` signals a ready state:

```
kubectl get gitrepository example-repo -n ${PROJECT_NAMESPACE}
```

The repository commit also displays the ready state:

```
NAME         URL                                                     READY
STATUS                                                               AGE
example-repo https://github.com/example-org/example-repo             True
Fetched revision: master/6c54bd1722604bd03d25dcac7a31c44ff4e03c6a   11m
```

For more information on the GitRepository resource fields and how to make Flux aware of credentials required to access a private Git repository, see the Flux documentation[429].

Troubleshoot
To troubleshoot issues with adding the GitRepository, review the following logs:

```
kubectl -n kommander-flux logs -l app=source-controller
[...]
kubectl -n kommander-flux logs -l app=kustomize-controller
[...]
kubectl -n kommander-flux logs -l app=helm-controller
[...]
```

Related Information

- Flux[430]
- Flux docs[431]

Projects - Git repository structure

ENTERPRISE

Git repositories must be structured in a specific manner for defined applications to be processed by Kommander.

You must structure your git repository based on the following guidelines, for your applications to be processed properly by Kommander so that they can be deployed.

Git Repository Directory Structure

---

429 https://fluxcd.io/docs/components/source/gitrepositories/
430 https://fluxcd.io/
431 https://fluxcd.io/docs

Use the following basic directory structure for your git repository:

```
├── helm-repositories
│   ├── <helm repository 1>
│   │   ├── kustomization.yaml
│   │   └── <helm repository name>.yaml
│   └── <helm repository 2>
│     ├── kustomization.yaml
│     └── <helm repository name>.yaml
└── services
    ├── <app name>
    │   ├── <app version1> # semantic version of the app helm chart. e.g., 1.2.3
    │   │   ├── defaults
    │   │   │   ├── cm.yaml
    │   │   │   └── kustomization.yaml
    │   │   ├── <app name>.yaml
    │   │   └── kustomization.yaml
    │   ├── <app version2> # another semantic version of the app helm chart. e.g.,
2.3.4
    │   │   ├── defaults
    │   │   │   ├── cm.yaml
    │   │   │   └── kustomization.yaml
    │   │   ├── <app name>.yaml
    │   │   └── kustomization.yaml
    │   └── metadata.yaml
    └── <another app name>
    ...
```

- Define applications in the `services/` directory.
- You can define multiple versions of an application, under different directories nested under the `services/<app name>/` directory.
- Define application manifests, such as a HelmRelease[432], under each versioned directory `services/<app name>/<version>/` in `<app name>.yaml` which is listed in the `kustomization.yaml` Kubernetes Kustomization file. For more information, see the Kubernetes Kustomization docs[433].
- Define the default values ConfigMap for `HelmReleases` in the `services/<app name>/<version>/defaults` directory, accompanied by a `kustomization.yaml` Kubernetes Kustomization file pointing to the `ConfigMap` file.
- Define the `metadata.yaml` of each application under the `services/<app name>/` directory. For more information, see the Application Metadata docs(see page 454).

See the DKP Catalog repository[434] for an example of how to structure custom catalog Git repositories.

Helm Repositories

You must include the `HelmRepository` that is referenced in each `HelmRelease`'s `Chart` spec.

---

432 https://fluxcd.io/docs/components/helm/helmreleases/
433 https://kubectl.docs.kubernetes.io/references/kustomize/kustomization/
434 https://github.com/mesosphere/dkp-catalog-applications

Each `services/<app name>/<version>/kustomization.yaml` must include the path of the YAML file that defines the `HelmRepository`. For example:

```
# services/<app name>/<version>/kustomization.yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
resources:
  - <app name>.yaml
  - ../../../helm-repositories/<helm repository 1>
```

For more information, see the flux documentation about HelmRepositories[435].

Substitution variables
Some substitution variables[436] are provided.

- `${releaseName}`: For each App deployment, this variable is set to the `AppDeployment` name. Use this variable to prefix the names of any resources that are defined in the application directory in the Git repository so that multiple instances of the same application can be deployed. If you create resources without using the `releaseName` prefix (or suffix) in the name field, there can be conflicts if the same named resource is created in that same namespace.
- `${releaseNamespace}`: The namespace of the Project.
- `${workspaceNamespace}`: The namespace of the Workspace that the Project belongs to.

Related Information
- Flux[437]
- Flux docs[438]
- Getting started with Flux guide[439]

Projects - Application Metadata

ENTERPRISE

To display more information about custom applications in the UI, define a metadata.yaml file for each application in the git repository.

You can define how custom applications display in the DKP UI by defining a `metadata.yaml` file for each application in your git repository. You must define this file at `services/<application>/metadata.yaml` for it to process correctly.

You can define the following fields:

435 https://fluxcd.io/docs/components/source/helmrepositories/
436 https://fluxcd.io/docs/components/kustomize/kustomization/#variable-substitution
437 https://fluxcd.io/
438 https://fluxcd.io/docs
439 https://fluxcd.io/docs/get-started/

Field	Default	Description
displayName	falls back to App ID	Display name of the application for the UI.
description	“”	Short description, should be a sentence or two, displayed in the UI on the application card.
category	general	1 or more categories for this application. Categories are used to group applications in the UI.
overview		Markdown overview used on the application detail page in the UI.
icon		Base64 encoded icon SVG file used for application logos in the UI.
scope	workspace	List of scopes, can be workspace and/or project currently.

None of these fields are required for the application to display in the UI.

Here is an example `metadata.yaml` file:

```
displayName: Prometheus Monitoring Stack
description: Stack of applications that collect metrics and provides visualization
and alerting capabilities. Includes Prometheus, Prometheus Alertmanager and Grafana.
category:
  - monitoring
overview: >
  # Overview
  A stack of applications that collects metrics and provides visualization and
alerting capabilities. Includes Prometheus, Prometheus Alertmanager and Grafana.

  ## Dashboards
  By deploying the Prometheus Monitoring Stack, the following platform applications
and their respective dashboards are deployed. After deployment to clusters in a
workspace, the dashboards are available to access from a respective cluster's detail
page.

  ### Prometheus

  A software application for event monitoring and alerting. It records real-time
metrics in a time series database built using a HTTP pull model, with flexible and
real-time alerting.
```

```
  - [Prometheus Documentation - Overview](https://prometheus.io/docs/introduction/
overview/)

  ### Prometheus Alertmanager
  A Prometheus component that enables you to configure and manage alerts sent by the
Prometheus server and to route them to notification, paging, and automation systems.

  - [Prometheus Alertmanager Documentation - Overview](https://prometheus.io/docs/
alerting/latest/alertmanager/)

  ### Grafana
  A monitoring dashboard from Grafana that can be used to visualize metrics collected
by Prometheus.

  - [Grafana Documentation](https://grafana.com/docs/)
icon:
PHN2ZyB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciIHZpZXdCb3g9IjAgMCAzMDAgMzAwIiBzd
HlsZT0iZW5hYmxlLWJhY2tncm91bmQ6bmV3IDAgMCAzMDAgMzAwIiB4bWw6c3BhY2U9InByZXNlcnZlIj48cG
F0aCBkPSJNMTUwIDUwQzk0LjggNTAgNTAgOTQuOCA1MCAxNTBzNDQuOCAxMDAgMTAwIDEwMCAxMDAtNDQuOCA
xMDAtMTAwUzIwNS4yIDUwIDE1MCA1MHptMCAxODcuMmMtMTUuNyAwLTI4LjUtMTAuNS0yOC41LTIzLjRoNTYu
OWMuMSAxMi45LTEyLjcgMjMuNC0yOC40IDIzLjR6bTQ3LTMxLjJoLTk0di0xN2g5NHYxN3ptLS4zLTI1LjloL
TkzLjRjLS4zLS40LS42LS43LS45LTEuMS05LjYtMTEuNy0xMS45LTE3LjgtMTQuMS0yNCAwLS4yIDExLjcgMi
40IDIwIDQuMyAwIDAgNC4zIDEgMTAuNSAyLjEtNi03LTkuNi0xNi05LjYtMjUuMSAwLTIwIDE1LjQtMzcuNiA
5LjgtNTEuNyA1LjQuNCAxMS4yIDExLjQgMTEuNiAyOC41IDUuNy03LjkgOC4xLTIyLjQgOC4xLTMxLjMgMC05
LjIgNi4xLTE5LjkgMTIuMS0yMC4yLTUuNCA4LjkgMS40IDE2LjUgNy40IDM1LjUgMi4zIDcuMSAyIDE5LjEgM
y43IDI2LjcuNi0xNS44IDMuMy0zOC44IDEzLjMtNDYuNy00LjQgMTAgLjcgMjIuNSA0LjEgMjguNSA1LjYgOS
43IDkgMTcuMSA5IDMxIDAgOS4zLTMuNCAxOC4xLTkuMyAyNSA2LjYtMS4yIDExLjItMi40IDExLjItMi40bDI
xLjQtNC4yYy4xIDAtMyAxMi44LTE0LjkgMjUuMXoiIHN0eWxlPSJmaWxsOiNmODQzMTEiLz48L3N2Zz4=
```

Enable a Custom Application from the Project Catalog

ENTERPRISE

Enable a Custom Application from the Project Catalog

After creating a GitRepository, you can either use the DKP UI or the CLI to enable your custom applications.

> ⚠ From within a project, you can enable applications to deploy. Verify that an application has successfully deployed via the CLI.

Enable the Application using the DKP UI

1. From the top menu bar, select your target workspace.
2. Select **Projects** from the sidebar menu.
3. Select your project from the list.
4. Select the **Applications** tab to browse the available applications from your configured repositories.
5. Select the three dot button from the bottom-right corner of the desired application tile, and then select **Enable**.
6. If available, select a version from the drop-down menu. This drop-down menu will only be visible if there is more than one version.
7. (Optional) If you want to override the default configuration values, copy your values content into the text editor under **Configure Service** or just upload your yaml file that contains the values:

```
someField: someValue
```

8. Confirm the details are correct, and then select the **Enable** button.

For all applications, you must provide a display name and an ID which is automatically generated based on what you enter for the display name, unless or until you edit the ID directly. The ID must be compliant with Kubernetes DNS subdomain name validation rules[440].

Alternately, you can use the CLI(see page 0) to enable your custom applications in the section below.

Enable the Application using the CLI
Follow these steps:

1. Set the `PROJECT_NAMESPACE` environment variable to the name of the project's namespace:

```
export PROJECT_NAMESPACE=<project_namespace>
```

2. Get the list of available applications to enable using the following command:

```
kubectl get apps -n ${PROJECT_NAMESPACE}
```

3. Enable one of the supported applications from the list with an `AppDeployment` resource.
4. Within the `AppDeployment` resource. Provide the `appRef` and application version to specify which `App` is deployed:

```
cat <<EOF | kubectl apply -f -
apiVersion: apps.kommander.d2iq.io/v1alpha3
kind: AppDeployment
metadata:
  name: my-custom-app
  namespace: ${PROJECT_NAMESPACE}
spec:
  appRef:
    name: custom-app-0.0.1
    kind: App
EOF
```

The `appRef.name` must match the app `name` from the list of available applications.

Enable an Application with a Custom Configuration using the CLI
Follow these steps:

1. Provide the name of a `ConfigMap` in the `AppDeployment`, which provides custom configuration on top of the default configuration:

440 https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#dns-subdomain-names

```
cat <<EOF | kubectl apply -f -
apiVersion: apps.kommander.d2iq.io/v1alpha3
kind: AppDeployment
metadata:
  name: my-custom-app
  namespace: ${PROJECT_NAMESPACE}
spec:
  appRef:
    name: custom-app-0.0.1
    kind: App
  configOverrides:
    name: my-custom-app-overrides
EOF
```

2. Create the `ConfigMap` with the name provided in the step above, with the custom configuration:

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: ${PROJECT_NAMESPACE}
  name: my-custom-app-overrides
data:
  values.yaml: |
    someField: someValue
EOF
```

Kommander waits for the `ConfigMap` to be present before enabling the `AppDeployment` to the attached clusters in the Project.

Verify Applications
After completing the previous steps, your applications are enabled. Connect to the attached cluster and check the `HelmReleases` to verify the deployments:

```
kubectl get helmreleases -n ${PROJECT_NAMESPACE}
```

The output looks similar to this:

```
NAMESPACE               NAME            READY   STATUS
AGE
project-test-vjsfq      my-custom-app   True    Release reconciliation succeeded
7m3s
```

Upgrade Custom Applications
You must maintain your custom applications manually. When upgrading DKP, ensure you validate for compatibility issues any custom applications you run against the current version of Kubernetes. We recommend upgrading to the latest compatible application versions as soon as possible.

8.5.10.3 Project AppDeployments

ENTERPRISE

An `AppDeployment` is a Custom Resource created by DKP with the purpose of deploying applications (platform, DKP catalog and custom applications). For more information about these Custom Resources and how to customize them, refer to the AppDeployments(see page 418) documentation.

8.6 Manage Clusters

View clusters created with Kommander or any connected Kubernetes cluster

Kommander allows you to monitor and manage very large numbers of clusters. Use the features described in this area to connect existing clusters, or to create new clusters whose lifecycle is managed by Konvoy. You can view clusters from the Clusters tab in the navigation pane on the left. You can see the details for a cluster by selecting the **View Details** link at the bottom of the cluster card or the cluster name in either the card or the table view.

8.6.1 Types

There are several types of clusters that display in the Clusters tab. The cluster type appears in the cluster card just under the cluster name.

The type values include:

- **Attached**: An Attached cluster is one that was not created with Kommander. You cannot manage an Attached cluster's lifecycle, but you can monitor it.
- **Managed**: A Managed cluster is a Konvoy cluster that was created with Kommander. You can use Kommander to manage a Managed cluster's lifecycle.
- **Management**: This is the Konvoy cluster that hosts Kommander.

8.6.2 Statuses

A cluster card's status line displays both the current status and the version of Kubernetes running in the cluster.

The status list includes these values:

Status	Description
Pending	This is the initial state when a cluster is created or connected.
Pending Setup	The cluster has networking restrictions that require additional setup, and is not yet connected or attached.
Loading Data	The cluster has been added to Kommander and we are fetching details about the cluster. This is the status before `Active`.

Status	Description
Active	The cluster is connected to API server.
Provisioning*	The cluster is being created on your cloud provider. This process may take some time.
Provisioned*	The cluster's infrastructure has been created and configured.
Joining	The cluster is being joined to the management cluster for federation.
Joined	The join process is done, and waiting for the first data from the cluster to arrive.
Deleting*	The cluster and its resources are being removed from your cloud provider. This process may take some time.
Error	There has been an error connecting to the cluster or retrieving data from the cluster.
Join Failed	This status can appear when kubefed does not have permission to create entities in the target cluster.
Unjoining	Kubefed is cleaning up after itself, removing all installed resources on the target cluster.
Unjoined	The cluster has been disconnected from the management cluster.
Unjoin Failed	The Unjoin from kubefed failed or there is some other error with deleting or disconnecting.
Unattached*	The cluster was created manually and the infrastructure has been created and configured. However, the cluster is not attached. Review the Manually attach a CLI-created cluster(see page 541) page to resolve this status.

*These statuses only appear on Managed clusters.

8.6.3 Resources

The Resources graphs on a cluster card show you a cluster's resource requests, limits, and usage. This allows a quick, visual scan of cluster health. Hover over each resource to get specific details for that specific cluster resource.

Resource	Description
CPU Requests	The requested portion of the total allocatable CPU resource for the cluster, measured in number of cores, such 0.5 cores.
CPU Limits	The portion of the total allocatable CPU resource to which the cluster is limited, measured in number of cores, such as 0.5 cores.
CPU Usage	The amount of the allocatable CPU resource being consumed. Cannot be higher than the configured CPU limit. Measured in number of cores, such as 0.5 cores)
Memory Requests	The requested portion of the total allocatable memory resource for the cluster, measured in bytes, such as 64 GiB.
Memory Limits	The portion of the allocatable memory resource to which the cluster is limited, measured in bytes, such as 64 GiB.
Memory Usage	The amount of the allocatable memory resource being consumed. Cannot be higher than the configured memory limit. Measured in bytes, such as 64 GiB.
Disk Requests	The requested portion of the allocatable ephemeral storage resource for the cluster, measured in bytes, such as 64 GiB.
Disk Limits	The portion of the allocatable ephemeral storage resource to which the cluster is limited, measured in bytes, such as 64 GiB.

For more detailed information, see the Kubernetes documentation[441] about resources.

441 https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

8.6.4 Platform applications

Platform applications, formerly called Addons, are services that the management cluster installs. You can visit a cluster's detail page to see which platform applications are enabled under the "Platform Applications" section.

Figure 1. Cluster detail page

Review the Workspace Platform Configuration Requirements(see page 430) to ensure that the attached clusters have sufficient resources. For more information on platform applications and how to customize them, see Workspace Platform Applications(see page 421).

8.6.5 Edit a cluster

8.6.5.1 Edit an attached cluster

For an attached cluster, you can edit labels assigned to that cluster.

8.6.6 Attach an Existing Kubernetes Cluster

ENTERPRISE

Attach an existing Kubernetes cluster using kubeconfig

8.6.6.1 Attach Kubernetes Cluster

You can attach an existing cluster directly to Kommander. At the time of attachment, certain namespaces are created on the cluster, and workspace platform applications are deployed automatically into the newly-created namespaces.

Review the Workspace Platform Application Configuration Requirements(see page 430) to ensure the attached cluster has sufficient resources. For more information on platform applications and customizing them, see Workspace Applications(see page 434).

If the cluster you want to attach was created using Amazon EKS or Google GKE, create a service account as described below. If you are attaching an Amazon EKS cluster to Kommander, detailed instructions are available(see page 533).

Create a kubeconfig file for your EKS cluster

⚠ If you already have a `kubeconfig` file, **SKIP this section**.

To get started, ensure you have kubectl[442] set up and configured with ClusterAdmin[443] for the cluster you want to connect to Kommander.

1. Create the necessary service account:

```
kubectl -n kube-system create serviceaccount kommander-cluster-admin
```

2. Create a token secret for the `serviceaccount`:

```
kubectl -n kube-system create  -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
  name: kommander-cluster-admin-sa-token
  annotations:
    kubernetes.io/service-account.name: kommander-cluster-admin
type: kubernetes.io/service-account-token
EOF
```

3. Verify that the `serviceaccount` token is ready by running this command:

```
kubectl -n kube-system get secret kommander-cluster-admin-sa-token -oyaml
```

   Verify that the `data.token` field is populated. The output should be similar to this:

```
apiVersion: v1
data:
  ca.crt: LS0tLS1CRUdJTiBDR...
  namespace: ZGVmYXVsdA==
  token: ZXlKaGJHY2lPaUpTVX...
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: kommander-cluster-admin
    kubernetes.io/service-account.uid: b62bc32e-b502-4654-921d-94a742e273a8
  creationTimestamp: "2022-08-19T13:36:42Z"
  name: kommander-cluster-admin-sa-token
  namespace: default
  resourceVersion: "8554"
  uid: 72c2a4f0-636d-4a70-9f1c-55a75f15e520
type: kubernetes.io/service-account-token
```

4. Configure the new service account for `cluster-admin` permissions:

442 https://kubernetes.io/docs/tasks/tools/#kubectl
443 https://kubernetes.io/docs/concepts/cluster-administration/

```
cat << EOF | kubectl apply -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: kommander-cluster-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: kommander-cluster-admin
  namespace: kube-system
EOF
```

5. Set up the following environment variables with the access data that is needed for producing a new kubeconfig file:

```
export USER_TOKEN_VALUE=$(kubectl -n kube-system get secret/kommander-cluster-
admin-sa-token -o=go-template='{{.data.token}}' | base64 --decode)
export CURRENT_CONTEXT=$(kubectl config current-context)
export CURRENT_CLUSTER=$(kubectl config view --raw -o=go-
template='{{range .contexts}}{{if eq .name "'''${CURRENT_CONTEXT}'''"}}
{{ index .context "cluster" }}{{end}}{{end}}')
export CLUSTER_CA=$(kubectl config view --raw -o=go-
template='{{range .clusters}}{{if eq .name "'''${CURRENT_CLUSTER}'''"}}"{{with
index .cluster "certificate-authority-data" }}{{.}}{{end}}"{{ end }}{{ end }}')
export CLUSTER_SERVER=$(kubectl config view --raw -o=go-
template='{{range .clusters}}{{if eq .name "'''${CURRENT_CLUSTER}'''"}}
{{ .cluster.server }}{{end}}{{ end }}')
```

6. Confirm these variables have been set correctly:

```
export -p | grep -E 'USER_TOKEN_VALUE|CURRENT_CONTEXT|CURRENT_CLUSTER|
CLUSTER_CA|CLUSTER_SERVER'
```

7. Generate a kubeconfig file that uses the environment variable values from the previous step:

```
cat << EOF > kommander-cluster-admin-config
apiVersion: v1
kind: Config
current-context: ${CURRENT_CONTEXT}
contexts:
- name: ${CURRENT_CONTEXT}
  context:
    cluster: ${CURRENT_CONTEXT}
    user: kommander-cluster-admin
    namespace: kube-system
clusters:
```

```
- name: ${CURRENT_CONTEXT}
  cluster:
    certificate-authority-data: ${CLUSTER_CA}
    server: ${CLUSTER_SERVER}
users:
- name: kommander-cluster-admin
  user:
    token: ${USER_TOKEN_VALUE}
EOF
```

8. This process produces a file in your current working directory called `kommander-cluster-admin-config`. The contents of this file are used in Kommander to attach the cluster.

   Before importing this configuration, verify the `kubeconfig` file can access the cluster:

   ```
   kubectl --kubeconfig $(pwd)/kommander-cluster-admin-config get all --all-namespaces
   ```

Finalize attaching your cluster from the UI

Using the **Add Cluster** option, you can attach an existing Kubernetes or Konvoy cluster directly to Kommander. This enables you to access the multi-cluster management and monitoring benefits that Kommander provides, while keeping your existing cluster on its current provider and infrastructure.

1. From the top menu bar, select your target workspace.
2. On the Dashboard page, select the **Add Cluster** option in the **Actions** dropdown menu at the top right.
3. Select **Attach Cluster**.
4. Select the **No additional networking restrictions** card. Alternatively, if you MUST use network restrictions, stop following the steps below, and see the instructions on the page Attach a Cluster with Network Restrictions(see page 512).
5. Upload the kubeconfig file you created in the previous section (or copy its contents) into the **Cluster Configuration** section.
6. The **Cluster Name** field automatically populates with the name of the cluster is in the kubeconfig. You can edit this field with the name you want for your cluster.
7. The **Context** select list is populated from the kubeconfig. Select the desired context with admin privileges from the **Context** select list.
8. Add labels to classify your cluster as needed.
9. Select **Create** to attach your cluster.

Platform applications extend the functionality of Kubernetes and provide ready-to-use logging and monitoring stacks by deploying platform applications when attaching a cluster to Kommander. For more information, refer to Workspace Applications(see page 434).

8.6.6.2 Requirements for Attaching an Existing Cluster

ENTERPRISE

Basic requirements

To attach an existing cluster in the UI, the Application Management cluster must be able to reach the services and the `api-server` of the target cluster.

> DKP does not support attachment of K3s clusters.

For attaching existing clusters without networking restrictions, the requirements depend on which DKP version you are using. Each version of DKP supports a specific range of Kubernetes versions(see page 0). You must ensure that the target cluster is running a compatible version.

For example, DKP 2.3 supports Kubernetes versions between 1.21.0 and 1.23.x. Any cluster you want to attach using must be running a Kubernetes version in that range.

Creating a default StorageClass

To deploy many of the services on the attached cluster, there must be a default `StorageClass` configured. Run the following command on the cluster you want to attach:

```
kubectl get sc
```

The output should look similar to this. Note the `(default)` after the name:

```
NAME               PROVISIONER       RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION   AGE
ebs-sc (default)   ebs.csi.aws.com   Delete          WaitForFirstConsumer   false
41s
```

If the `StorageClass` is not set as default, add the following annotation to the `StorageClass` manifest:

```
annotations:
  storageclass.kubernetes.io/is-default-class: "true"
```

Creating Projects and Workspaces

Before you attach clusters, you need to create one or more Workspaces, and we recommend that you also create Projects within your Workspaces. Workspaces(see page 432) give you a logical way to represent your teams and specific configurations. Projects(see page 459) let you define one or more clusters as a group to which Kommander pushes a common configuration. Grouping your existing clusters in Kommander projects and workspaces makes managing their platform services and resources easier and supports monitoring and logging.

> Do not attach a cluster in the "Management Cluster Workspace" workspace. This workspace is reserved for your Application Management cluster only.

Platform application requirements

In addition to the basic cluster requirements, the platform services you want DKP to manage on those clusters will have an impact on the total cluster requirements. The specific combinations of platform applications will make a difference in the requirements for the cluster nodes and their resources (CPU, memory, and storage).

See this table of platform applications(see page 421) that DKP provides by default.

Attaching existing AWS, EKS and GKE clusters

Attaching an existing AWS cluster requires that the cluster be fully configured and running(see page 117). You must create a separate service account when attaching existing AKS, EKS or Google GKE Kubernetes clusters. This is necessary because the kubeconfig files generated from those clusters are not usable out-of-the-box by Kommander. The kubeconfig files call CLI commands, such as `azure, aws` or `gcloud`, and use locally-obtained authentication tokens. Having a separate service account also allows you to keep access to the cluster specific and isolated to Kommander.

The suggested default cluster configuration includes a control plane pool containing three (3) m5.xlarge nodes and a worker pool containing four (4) m5.2xlarge nodes.

Consider the additional resource requirements for running the platform services you want DKP to manage, and ensure that your existing clusters comply.

To attach an existing EKS cluster, refer to the specific information in Attach Amazon EKS Cluster(see page 213).

To attach an existing GKE cluster, refer to the specific information in Attach GKE Cluster(see page 537).

Attach clusters with an existing cert-manager installation

If you are attaching clusters that already have cert-manager installed, the cert-manager `HelmRelease` provided by DKP will fail to deploy, due to the existing cert-manager installation. As long as the pre-existing cert-manager functions as expected, you can ignore this failure. It will have no impact on the operation of the cluster.

8.6.6.3 Generate a kubeconfig File

How to create a service account and generate a kubeconfig file for attaching an existing cluster

You should create a separate service account when attaching existing Amazon EKS, Azure AKS, or Google GKE Kubernetes clusters. This service account is needed because the kubeconfig files generated from those clusters are not usable out-of-the-box. They call CLI commands, such as `aws` or `gcloud`, and use locally-obtained authentication tokens. Having a separate service account also allows you to keep access to the cluster specifics and isolated to DKP Application Manager.

To get started, ensure you have kubectl[444] set up and configured with ClusterAdmin[445] for the cluster you want to connect.

1. Create the required service account using the command:

```
kubectl -n kube-system create serviceaccount kommander-cluster-admin
```

444 https://kubernetes.io/docs/tasks/tools/install-kubectl/
445 https://kubernetes.io/docs/concepts/cluster-administration/cluster-administration-overview/

2. Configure the new service account for `cluster-admin` permissions. You can and paste this example ensuring that you use the service account created previously.

```
cat << EOF | kubectl apply -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: kommander-cluster-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: kommander-cluster-admin
  namespace: kube-system
EOF
```

3. Set up the following environment variables with access data needed for producing a new kubeconfig file:

```
export USER_TOKEN_NAME=$(kubectl -n kube-system get serviceaccount kommander-
cluster-admin -o=jsonpath='{.secrets[0].name}')
export USER_TOKEN_VALUE=$(kubectl -n kube-system get secret/${USER_TOKEN_NAME}
-o=go-template='{{.data.token}}' | base64 --decode)
export CURRENT_CONTEXT=$(kubectl config current-context)
export CURRENT_CLUSTER=$(kubectl config view --raw -o=go-
template='{{range .contexts}}{{if eq .name "'''${CURRENT_CONTEXT}'''"}}
{{ index .context "cluster" }}{{end}}{{end}}')
export CLUSTER_CA=$(kubectl config view --raw -o=go-
template='{{range .clusters}}{{if eq .name "'''${CURRENT_CLUSTER}'''"}}"{{with
index .cluster "certificate-authority-data" }}{{.}}{{end}}"{{ end }}{{ end }}')
export CLUSTER_SERVER=$(kubectl config view --raw -o=go-
template='{{range .clusters}}{{if eq .name "'''${CURRENT_CLUSTER}'''"}}
{{ .cluster.server }}{{end}}{{ end }}')
```

4. Generate a kubeconfig file with these values:

```
cat << EOF > kommander-cluster-admin-config
apiVersion: v1
kind: Config
current-context: ${CURRENT_CONTEXT}
contexts:
- name: ${CURRENT_CONTEXT}
  context:
    cluster: ${CURRENT_CONTEXT}
    user: kommander-cluster-admin
    namespace: kube-system
clusters:
- name: ${CURRENT_CONTEXT}
```

```
  cluster:
    certificate-authority-data: ${CLUSTER_CA}
    server: ${CLUSTER_SERVER}
users:
- name: kommander-cluster-admin
  user:
    token: ${USER_TOKEN_VALUE}
EOF
```

This procedure produces a file in your current working directory called, `kommander-cluster-admin-config`. The contents of this file are used in DKP Application Manager to attach the cluster.

Before importing this configuration, you can verify that it is functional by running the following command:

```
kubectl --kubeconfig $(pwd)/kommander-cluster-admin-config get all --all-namespaces
```

Then, you can use this kubeconfig to:

- Attach a cluster with no additional networking restrictions(see page 512)
- Attach a cluster that has networking restrictions(see page 512)

If a cluster has limited resources to deploy all the federated platform services, it will fail to stay attached in the DKP UI. If this happens, check if there are any pods that are not getting the resources required.

8.6.6.4 Attach a Cluster with no Networking Restrictions

ENTERPRISE

How to attach an existing cluster that has no additional networking restrictions

Use this option when you want to attach a cluster that does not require additional access information.

1. From the top menu bar, select your target workspace.
2. On the Dashboard page, select the **Add Cluster** option in the **Actions** dropdown menu at the top right.
3. Select **Attach Cluster**.
4. Select the **No additional networking restrictions** card.
5. In the **Cluster Configuration** section, paste your kubeconfig file into the field, or select the **Upload kubeconfig File** button to specify the file.
6. The **Cluster Name** field will automatically populate with the name of the cluster is in the kubeconfig. You can edit this field with the name you want for your cluster.
7. The **Context** select list is populated from the kubeconfig. Select the desired context with admin privileges from the **Context** select list.
8. Add labels to classify your cluster as needed.
9. Select **Submit** to attach your cluster.

8.6.6.5 Attach a Cluster with Networking Restrictions

ENTERPRISE

How to attach an existing cluster that has additional networking restrictions

Use this option when you want to attach a cluster that is in a DMZ, behind a NAT gateway, behind a proxy server or a firewall, or that requires additional access information. This procedure gathers the information required to create a kubeconfig file for the network tunnel between Kommander and the cluster you want to attach.

If your cluster blocks public access, you may need to make the additional step of allowing certain authorized networks where Docker images are hosted for Konvoy to use your cluster, specifically `https://registry-1.docker.io/`

1. From the top menu bar, select your target workspace.
2. On the Dashboard page, select the **Add Cluster** option in the **Actions** dropdown menu at the top right.
3. Select **Attach Cluster**.
4. Select the **Cluster has networking restrictions** card to display the configuration page.
5. Enter the **Cluster Name** of the cluster you're attaching.
6. Create additional new Labels as needed.
7. Select the hostname that is the Ingress for the cluster from the **Load Balancer Hostname** dropdown menu. The hostname must match the Kommander Host cluster to which you are attaching your existing cluster with network restrictions.
8. Specify the **URL Path Prefix** for your Load Balancer Hostname. This URL path will serve as the prefix for the specific tunnel services you want to expose on the Kommander management cluster. If no value is specified, the value defaults to `/dkp/tunnel`.

   NOTE: Kommander uses Traefik 2 ingress, which requires explicit definition of strip prefix middleware as a Kubernetes API object, opposed to a simple annotation. Kommander provides default middleware that supports creating tunnels only on the `/dkp/tunnel URL` prefix. This is indicated by using the extra annotation, `traefik.ingress.kubernetes.io/router.middlewares: kommander-stripprefixes-kubetunnel@kubernetescrd` as shown in the code sample that follows. If you want to expose a tunnel on a different URL prefix, you must manage your own middleware configuration.
9. (Optional) Enter a value for the **Hostname** field.
10. If you have not attached this cluster before, you must create a new secret in the **Root CA Certificate** drop down menu. To do this in your Konvoy management cluster, view your base64 encoded Kubernetes secret values to copy and paste into the **Root CA Certificate** field:

    ```
    echo $(kubectl get secret -n cert-manager kommander-ca -o=go-
    template='{{index .data "tls.crt"}}')
    ```

    Otherwise, select from the list of available Secrets.
11. Add any **Extra Annotations** as needed.
12. Select the **Save & Generate kubeconfig** button to generate the kubeconfig file for the network tunnel.

After the above is complete, finish Attaching the Cluster(see page 533).

As an alternative procedure, you can follow these instructions to Use CLI to Add Managed Clusters to Kommander(see page 541).

For information on TunnelGateway, review the API documentation(see page 648).

Attach Cluster Using Tunnel

Using the CLI to attach a Kubernetes Cluster using a Tunnel

ENTERPRISE

Identify the management cluster endpoint

Obtain the hostname and CA certificate for the management cluster:

```
hostname=$(kubectl get service -n kommander kommander-traefik -o go-template='{{with
index .status.loadBalancer.ingress 0}}{{or .hostname .ip}}{{end}}')
b64ca_cert=$(kubectl get secret -n cert-manager kommander-ca -o=go-
template='{{index .data "tls.crt"}}')
```

Specify a workspace namespace

Obtain the desired workspace namespace on the management cluster for the tunnel gateway:

```
namespace=$(kubectl get workspace default-workspace -o jsonpath="{.status.namespaceRe
f.name}")
```

Alternatively, if you wish to create a new workspace instead of using an existing workspace:

```
workspace=sample
namespace=${workspace}

cat > workspace.yaml <<EOF
apiVersion: workspaces.kommander.mesosphere.io/v1alpha1
kind: Workspace
metadata:
  annotations:
    kommander.mesosphere.io/display-name: ${workspace}
  name: ${workspace}
spec:
  namespaceName: ${namespace}
EOF

kubectl apply -f workspace.yaml
```

You can verify the workspace exists using:

```
kubectl get workspace ${workspace}
```

Create a tunnel gateway

Create a tunnel gateway(see page 0) on the management cluster to listen for tunnel agents on remote clusters:

NOTE: Kommander uses Traefik 2 ingress, which requires explicit definition of strip prefix middleware as a Kubernetes API object, opposed to a simple annotation. Kommander provides default middleware that supports creating tunnels only on the `/dkp/tunnel URL` prefix. This is indicated by using the extra annotation, `traefik.ingress.kubernetes.io/router.middlewares: kommander-stripprefixes-`

`kubetunnel@kubernetescrd` as shown in the code sample that follows. If you want to expose a tunnel on a different URL prefix, you must manage your own middleware configuration.

```
cacert_secret=kubetunnel-ca
gateway=sample-gateway

cat > gateway.yaml <<EOF
apiVersion: v1
kind: Secret
metadata:
  namespace: ${namespace}
  name: ${cacert_secret}
data:
  ca.crt:
    ${b64ca_cert}
---
apiVersion: kubetunnel.d2iq.io/v1alpha1
kind: TunnelGateway
metadata:
  namespace: ${namespace}
  name: ${gateway}
spec:
  ingress:
    caSecretRef:
      namespace: ${namespace}
      name: ${cacert_secret}
    loadBalancer:
      hostname: ${hostname}
    urlPathPrefix: /dkp/tunnel
    extraAnnotations:
      kubernetes.io/ingress.class: kommander-traefik
      traefik.ingress.kubernetes.io/router.tls: "true"
      traefik.ingress.kubernetes.io/router.middlewares: kommander-stripprefixes-
kubetunnel@kubernetescrd
EOF

kubectl apply -f gateway.yaml
```

You can verify the gateway exists using the command:

```
kubectl get tunnelgateway -n ${namespace} ${gateway}
```

Connecting a remote cluster

Create a tunnel connector

Create a tunnel connector(see page 530) on the management cluster for the remote cluster:

```
connector=sample-connector
```

```
cat > connector.yaml <<EOF
apiVersion: kubetunnel.d2iq.io/v1alpha1
kind: TunnelConnector
metadata:
  namespace: ${namespace}
  name: ${connector}
spec:
  gatewayRef:
    name: ${gateway}
EOF

kubectl apply -f connector.yaml
```

You can verify the connector exists using:

```
kubectl get tunnelconnector -n ${namespace} ${connector}
```

Wait for the tunnel connector to reach `Listening` state and then export the agent manifest:

```
while [ "$(kubectl get tunnelconnector -n ${namespace} ${connector} -o jsonpath="{.st
atus.state}")" != "Listening" ]
do
  sleep 5
done

manifest=$(kubectl get tunnelconnector -n ${namespace} ${connector} -o
jsonpath="{.status.tunnelAgent.manifestsRef.name}")
while [ -z ${manifest} ]
do
  sleep 5
  manifest=$(kubectl get tunnelconnector -n ${namespace} ${connector} -o
jsonpath="{.status.tunnelAgent.manifestsRef.name}")
done

kubectl get secret -n ${namespace} ${manifest} -o jsonpath='{.data.manifests\.yaml}'
 | base64 -d > manifest.yaml
```

Create a network policy for the tunnel server
This step is optional, but improves security by restricting which remote hosts can connect to the tunnel.

Apply a network policy that restricts tunnel access to specific namespaces and IP blocks. The following example permits connections from pods running in the `kommander` namespace, from pods running in namespaces with a label `kubetunnel.d2iq.io/networkpolicy` matching the tunnel name and namespace, and to remote clusters with IP addresses in the ranges 192.0.2.0 to 192.0.2.255 and 203.0.113.0 to 203.0.113.255:

```
cat > net.yaml <<EOF
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
```

```
  namespace: ${namespace}
  name: ${connector}-deny
  labels:
    kubetunnel.d2iq.io/tunnel-connector: ${connector}
    kubetunnel.d2iq.io/networkpolicy-type: "tunnel-server"
spec:
  podSelector:
    matchLabels:
      kubetunnel.d2iq.io/tunnel-connector: ${connector}
  policyTypes:
  - Ingress
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  namespace: ${namespace}
  name: ${connector}-allow
  labels:
    kubetunnel.d2iq.io/tunnel-connector: ${connector}
    kubetunnel.d2iq.io/networkpolicy-type: "tunnel-server"
spec:
  podSelector:
    matchLabels:
      kubetunnel.d2iq.io/tunnel-connector: ${connector}
  policyTypes:
  - Ingress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: "kube-federation-system"
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: "kommander"
    - namespaceSelector:
        matchLabels:
          kubetunnel.d2iq.io/networkpolicy: ${connector}-${namespace}
    - ipBlock:
        cidr: 192.0.2.0/24
    - ipBlock:
        cidr: 203.0.113.0/24
EOF

kubectl apply -f net.yaml
```

Set up the managed cluster

the `manifest.yaml` file to the managed cluster and deploy the tunnel agent:

```
kubectl apply --context managed -f manifest.yaml
```

You can check the status of the created pods using:

```
kubectl get pods --context managed -n kubetunnel
```

After a short time, expect to see a `post-kubeconfig` pod that reaches `Completed` state and a `tunnel-agent` pod that stays in `Running` state.

```
NAME                            READY   STATUS      RESTARTS   AGE
post-kubeconfig-j2ghk           0/1     Completed   0          14m
tunnel-agent-f8d9f4cb4-thx8h    0/1     Running     0          14m
```

Add the managed cluster into Kommander
On the management cluster, wait for the tunnel to be connected by the tunnel agent:

```
while [ "$(kubectl get tunnelconnector -n ${namespace} ${connector} -o jsonpath="{.st
atus.state}")" != "Connected" ]
do
  sleep 5
done
```

Add the cluster into Kommander:

```
managed=private-cluster
display_name=${managed}

cat > kommander.yaml <<EOF
apiVersion: kommander.mesosphere.io/v1beta1
kind: KommanderCluster
metadata:
  namespace: ${namespace}
  name: ${managed}
  annotations:
    kommander.mesosphere.io/display-name: ${display_name}
spec:
  clusterTunnelConnectorRef:
    name: ${connector}
EOF

kubectl apply -f kommander.yaml
```

Wait for the managed cluster to join the Kommander cluster:

```
while [ "$(kubectl get kommandercluster -n ${namespace} ${managed} -o
jsonpath='{.status.phase}')" != "Joined" ]
do
  sleep 5
done
```

```
kubefed=$(kubectl get kommandercluster -n ${namespace} ${managed} -o
jsonpath="{.status.kubefedclusterRef.name}")
while [ -z "${kubefed}" ]
do
  sleep 5
  kubefed=$(kubectl get kommandercluster -n ${namespace} ${managed} -o
jsonpath="{.status.kubefedclusterRef.name}")
done

kubectl wait --for=condition=ready --timeout=60s kubefedcluster -n kube-federation-
system ${kubefed}

kubectl get kubefedcluster -n kube-federation-system ${kubefed}
```

Using a remote cluster

To access services running on the remote cluster from the management cluster, connect to the tunnel proxy.

You can use these three methods:

1. If the client program supports use of a kubeconfig file, use the managed cluster's kubeconfig.
2. If the client program supports SOCKS5 proxies, use the proxy directly.
3. Otherwise, deploy a proxy server on the management cluster.

Managed cluster service
These sections require a service to run on the managed cluster.

As an example, start the following service:

```
service_namespace=test
service_name=webserver
service_port=8888
service_endpoint=${service_name}.${service_namespace}.svc.cluster.local:$
{service_port}

cat > nginx.yaml <<EOF
apiVersion: v1
kind: Namespace
metadata:
  name: ${service_namespace}
---
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: ${service_namespace}
  name: nginx-deployment
  labels:
    app: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx-app
```

```
  template:
    metadata:
      labels:
        app: nginx-app
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  namespace: ${service_namespace}
  name: ${service_name}
spec:
  selector:
    app: nginx-app
  type: ClusterIP
  ports:
  - targetPort: 80
    port: ${service_port}
EOF

kubectl apply -f nginx.yaml

kubectl rollout status deploy -n ${service_namespace} nginx-deployment
```

On the managed cluster, a client Job can access this service using:

```
cat > curl.yaml <<EOF
apiVersion: batch/v1
kind: Job
metadata:
  name: curl
spec:
  template:
    spec:
      containers:
      - name: curl
        image: curlimages/curl:7.76.0
        command: ["curl", "--silent", "--show-error", "http://${service_endpoint}"]
      restartPolicy: Never
  backoffLimit: 4
EOF

kubectl apply -f curl.yaml

kubectl wait --for=condition=complete job curl
```

```
podname=$(kubectl get pods --selector=job-name=curl --field-
selector=status.phase=Succeeded -o jsonpath='{.items[0].metadata.name}')

kubectl logs ${podname}
```

The final command returns the default Nginx web page:

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Use of kubeconfig file

This is primarily useful for running `kubectl` commands on the management cluster to monitor the managed cluster.

On the management cluster, a `kubeconfig` file for the managed cluster configured to use the tunnel proxy is available as a Secret. The Secret's name can be identified using:

```
kubeconfig_secret=$(kubectl get tunnelconnector -n ${namespace} ${connector} -o
jsonpath='{.status.kubeconfigRef.name}')
```

After setting `service_namespace` and `service_name` to the managed service resource, on the management cluster run:

```
cat > get-service.yaml <<EOF
apiVersion: batch/v1
kind: Job
```

```
metadata:
  name: get-service
spec:
  template:
    spec:
      containers:
      - name: kubectl
        image: bitnami/kubectl:1.19
        command: ["kubectl", "get", "service", "-n", "${service_namespace}", "$
{service_name}"]
        env:
        - name: KUBECONFIG
          value: /tmp/kubeconfig/kubeconfig
        volumeMounts:
        - name: kubeconfig
          mountPath: /tmp/kubeconfig
      volumes:
      - name: kubeconfig
        secret:
          secretName: "${kubeconfig_secret}"
      restartPolicy: Never
  backoffLimit: 4
EOF

kubectl apply -n ${namespace} -f get-service.yaml

kubectl wait --for=condition=complete --timeout=5m job -n ${namespace} get-service

podname=$(kubectl get pods -n ${namespace} --selector=job-name=get-service --field-
selector=status.phase=Succeeded -o jsonpath='{.items[0].metadata.name}')

kubectl logs -n ${namespace} ${podname}
```

Direct use of SOCKS5 proxy
To use the SOCKS5 proxy directly, obtain the SOCKS5 proxy endpoint using:

```
proxy_service=$(kubectl get tunnelconnector -n ${namespace} ${connector} -o
jsonpath='{.status.tunnelServer.serviceRef.name}')

socks_proxy=$(kubectl get service -n ${namespace} "${proxy_service}" -o jsonpath='{.s
pec.clusterIP}{":"}{.spec.ports[?(@.name=="proxy")].port}')
```

Provide the value of `${socks_proxy}` as the SOCKS5 proxy to your client.

For example, since `curl` supports SOCKS5 proxies, the managed service started above can be accessed from the management cluster by adding the SOCKS5 proxy to the `curl` command. After setting `service_endpoint` to the service endpoint, on the management cluster run:

```
cat > curl.yaml <<EOF
apiVersion: batch/v1
```

```
kind: Job
metadata:
  name: curl
spec:
  template:
    spec:
      containers:
      - name: curl
        image: curlimages/curl:7.76.0
        command: ["curl", "--silent", "--show-error", "--socks5-hostname", "$
{socks_proxy}", "http://${service_endpoint}"]
      restartPolicy: Never
  backoffLimit: 4
EOF

kubectl apply -f curl.yaml

kubectl wait --for=condition=complete --timeout=5m job curl

podname=$(kubectl get pods --selector=job-name=curl --field-
selector=status.phase=Succeeded -o jsonpath='{.items[0].metadata.name}')

kubectl logs ${podname}
```

The final command returns the same output as for the job on the managed cluster, demonstrating that the job on the management cluster accessed the service running on the managed cluster.

Use of deployed proxy on management cluster
To deploy a proxy on the management cluster, obtain the SOCKS5 proxy endpoint using:

```
proxy_service=$(kubectl get tunnelconnector -n ${namespace} ${connector} -o
jsonpath='{.status.tunnelServer.serviceRef.name}')

socks_proxy=$(kubectl get service -n ${namespace} "${proxy_service}" -o jsonpath='{.s
pec.clusterIP}{":"}{.spec.ports[?(@.name=="proxy")].port}')
```

Provide the value of `${socks_proxy}` as the SOCKS5 proxy to a proxy deployed on the management cluster.

After setting `service_endpoint` to the service endpoint, on the management cluster run:

```
cat > nginx-proxy.yaml <<EOF
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: nginx-proxy-crt
spec:
  secretName: nginx-proxy-crt-secret
  dnsNames:
  - nginx-proxy-service.${namespace}.svc.cluster.local
  issuerRef:
    group: cert-manager.io
```

```
    kind: ClusterIssuer
    name: kubernetes-ca
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-proxy
  labels:
    app: nginx-proxy-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-proxy-app
  template:
    metadata:
      labels:
        app: nginx-proxy-app
    spec:
      containers:
      - name: nginx-proxy
        image: mesosphere/ghostunnel:v1.5.3-server-backend-proxy
        args:
        - "server"
        - "--listen=:443"
        - "--target=${service_endpoint}"
        - "--cert=/etc/certs/tls.crt"
        - "--key=/etc/certs/tls.key"
        - "--cacert=/etc/certs/ca.crt"
        - "--unsafe-target"
        - "--disable-authentication"
        env:
        - name: ALL_PROXY
          value: socks5://${socks_proxy}
        ports:
        - containerPort: 443
        volumeMounts:
        - name: certs
          mountPath: /etc/certs
      volumes:
      - name: certs
        secret:
          secretName: nginx-proxy-crt-secret
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-proxy-service
spec:
  selector:
    app: nginx-proxy-app
  type: ClusterIP
  ports:
```

```
  - targetPort: 443
    port: 8765
EOF

kubectl apply -n ${namespace} -f nginx-proxy.yaml

kubectl rollout status deploy -n ${namespace} nginx-proxy

proxy_port=$(kubectl get service -n ${namespace} nginx-proxy-service -o
jsonpath='{.spec.ports[0].port}')
```

Any client running on the management cluster can now access the service running on the managed cluster using the proxy service endpoint. Note in the following that the `curl` job runs in the same namespace as the proxy, to provide access to the CA certificate secret.

```
cat > curl.yaml <<EOF
apiVersion: batch/v1
kind: Job
metadata:
  name: curl
spec:
  template:
    spec:
      containers:
      - name: curl
        image: curlimages/curl:7.76.0
        command:
        - curl
        - --silent
        - --show-error
        - --cacert
        - /etc/certs/ca.crt
        - https://nginx-proxy-service.${namespace}.svc.cluster.local:${proxy_port}
        volumeMounts:
        - name: certs
          mountPath: /etc/certs
      volumes:
      - name: certs
        secret:
          secretName: nginx-proxy-crt-secret
      restartPolicy: Never
  backoffLimit: 4
EOF

kubectl apply -n ${namespace} -f curl.yaml

kubectl wait --for=condition=complete --timeout=5m job -n ${namespace} curl

podname=$(kubectl get pods -n ${namespace} --selector=job-name=curl --field-
selector=status.phase=Succeeded -o jsonpath='{.items[0].metadata.name}')

kubectl logs -n ${namespace} ${podname}
```

The final command returns the same output as the job on the managed cluster, demonstrating that the job on the management cluster accessed the service running on the managed cluster.

API documentation (v1alpha1)

API Documentation (v1alpha1)

> *This document is automatically generated from the API definition in the code.*

Page Contents

- TunnelAgentStatus(see page 529)
- TunnelConnector(see page 530)
- TunnelConnectorList(see page 530)
- TunnelConnectorSpec(see page 530)
- TunnelConnectorStatus(see page 531)
- TunnelServerStatus(see page 532)

TunnelGateway
Provides an endpoint for remote clusters to connect to the management cluster.

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[446]	false
spec		TunnelGatewaySpec(see page 529)	false

TunnelGatewayIngressSpec

Field	Description	Scheme	Required
loadBalancer	Ingress point for the load-balancer. Traffic intended for the service should be sent to these ingress points. If not specified, the controller will derive from the Ingress record status field.	corev1.LoadBalancerIngress	false
host	Restrict access to requests addressed to a specific host or domain using the `IngressRule` format. Defaults to allow all hosts.	string	false

446 https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.23/#objectmeta-v1-meta

Field	Description	Scheme	Required
urlPathPrefix	URL path prefix to prepend to all endpoints. For example, if this field is set to `/ops/portal/kt`, the ingresses created will have URL paths like `/ops/portal/kt/default/cluster1/tunnel-server` and `/ops/portal/kt/default/cluster1/kubeconfig`. Defaults to root path (`/`).	string	false
caSecretRef	A secret reference to the root CA required to verify the ingress endpoints. The secret should have type `Opaque` and contain the key `ca.crt`. If not specified, remote hosts will use their system root CA's to verify the endpoints.	corev1.ObjectReference	false
extraAnnotations	Extra annotations to set on the Ingress object.	map[string]string	false

TunnelGatewayList
Contains a list of `TunnelGateway`.

Field	Description	Scheme	Required
metadata		metav1.ListMeta[447]	false

447 https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.23/#objectmeta-v1-meta

Field	Description	Scheme	Required
items		[]TunnelGateway[448]	true

TunnelGatewaySpec
If no ingress is set, the services will only be accessible on `localhost`.

Field	Description	Scheme	Required
ingress	Expose services using an Ingress as specified in the `TunnelGatewayIngressSpec`.	TunnelGatewayIngressSpec(see page 527)	false

KubeconfigWebhookStatus
Status of the kubeconfig webhook.

Field	Description	Scheme	Required
deploymentRef	A reference to the deployment for the kubeconfig webhook.	corev1.LocalObjectReference	false
serviceRef	A reference to the service for the kubeconfig webhook.	corev1.LocalObjectReference	false
ingressRef	A reference to the ingress for the kubeconfig webhook.	corev1.LocalObjectReference	false

TunnelAgentStatus
Status of the tunnel agent.

448 https://docs.d2iq.com/dkp/kommander/2.2/clusters/tunnel-cli/api-reference/#tunnelgateway

Field	Description	Scheme	Required
manifestsRef	A reference to a secret holding YAML manifests for launching the tunnel agent on the target cluster. The secret is a generic typed secret with filenames as the keys. There might be multiple files in the secret.	corev1.LocalObjectReference	false

TunnelConnector
Describes the local endpoint for the tunnel. A remote cluster will connect to this endpoint to create a tunnel.

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[449]	false
spec		TunnelConnectorSpec(see page 530)	false
status		TunnelConnectorStatus(see page 531)	false

TunnelConnectorList
Contains a list of `TunnelConnector`.

Field	Description	Scheme	Required
metadata		metav1.ListMeta[450]	false
items		[]TunnelConnector(see page 530)	true

TunnelConnectorSpec

449 https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.23/#objectmeta-v1-meta
450 https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.23/#objectmeta-v1-meta

Field	Description	Scheme	Required
gatewayRef	A reference to the `TunnelGateway` object which describes how tunnel services will be exposed outside the current cluster.	corev1.LocalObjectReference	false
proxyPort	The port for the tunnel proxy.	int32	false

TunnelConnectorStatus

Field	Description	Scheme	Required
state	State of the tunnel connector: `Starting` - the initial state; `Listening` - the local tunnel server is waiting for the remote agent to connect; `Pending` - the remote agent has connected but the local proxy is not ready; `Connected` - the tunnel is configured and contact to the remote API server succeeded; `Disconnected` - the tunnel is configured but contact to the remote API server failed; `Failed` - an unexpected error occurred, such as not being able to parse the kubeconfig.	TunnelConnectorState	false
tunnelServer	Status of the tunnel server.	TunnelServerStatus(see page 532)	false
kubeconfigWebhook	Status of the kubeconfig webhook.	KubeconfigWebhookStatus(see page 529)	false

Field	Description	Scheme	Required
tunnelAgent	Status of the tunnel agent.	TunnelAgentStatus(see page 529)	false
serviceAccountRef	A reference to the service account that will be used for registration (of the tunnel agent) and authentication purpose.	corev1.LocalObjectReference	false
roleRef	A reference to the role that will be bound to the service account for authorization purpose.	corev1.LocalObjectReference	false
roleBindingRef	A reference to the rolebinding that will be created to bind the service account and the role.	corev1.LocalObjectReference	false
kubeconfigRef	A reference to the secret holding the KUBECONFIG that the clients can use to talk to the API server of the target cluster when it becomes available.	corev1.LocalObjectReference	false
gatewayObservedGeneration	The generation of the linked TunnelGateway object associated with this object. When the linked TunnelGateway object is updated, a controller will update this status field which will in turn trigger a reconciliation of this object.	int64	false

TunnelServerStatus
Status of the tunnel server.

Field	Description	Scheme	Required
deploymentRef	A reference to the deployment for the tunnel server.	corev1.LocalObjectReference	false
serviceRef	A reference to the service for the tunnel server.	corev1.LocalObjectReference	false
ingressRef	A reference to the ingress for the tunnel server.	corev1.LocalObjectReference	false

8.6.6.6 Finish attaching the existing cluster

ENTERPRISE

How to apply the kubeconfig file to create the network tunnel to attach an existing cluster

Though the required file is now generated, you still need to apply it to create the network tunnel and complete the attachment process.

1. Select the **Download Manifest** link to download the file you generated previously(see page 512).
2. Copy the `kubectl apply...` command from the user interface and paste into your terminal session, substituting the actual name of the file for the variable. Running this command starts the attachment process, which may take several minutes to complete. The Cluster details page will be displayed automatically when the cluster attachment process completes.
3. (Optional) Select the **Verify Connection to Cluster** button to send a request to Kommander to refresh the connection information. You can use this option to check to see if the connection is complete, though the Cluster Details page displays automatically when the connection is complete.

After the initial connection is made, and your cluster becomes viewable as attached in the DKP UI, the attachment, federated add-ons, and platform services will still need to be completed. This may take several additional minutes. If a cluster has limited resources to deploy all the federated platform services, the installation of the federated resources will fail and the cluster may become unreachable in the DKP UI. If this happens, check whether there are any pods that are not getting the resources required.

8.6.6.7 Attach Amazon EKS Cluster

ENTERPRISE

Attach an existing EKS cluster

You can attach existing Kubernetes clusters to the Management Cluster. After attaching the cluster, you can use the UI to examine and manage(see page 502) this cluster. The following procedure shows how to attach an existing Amazon Elastic Kubernetes Service (EKS) cluster.

Prerequisites

This procedure requires the following items and configurations:

- A fully configured and running Amazon EKS[451] cluster with administrative privileges.
- The current version DKP Enterprise is installed[452]on your cluster.
- Ensure you have installed `kubectl` in your Management cluster.
- Install aws-iam-authenticator[453]. This binary is used to access your cluster using kubectl.

This procedure assumes you have an existing and spun up Amazon EKS cluster(s) with administrative privileges. Refer to the Amazon EKS[454] for setup and configuration information.

Attach Amazon EKS Clusters

If you have an existing EKS cluster, you will use this section to attach your cluster. If you have a newly created DKP CLI cluster, you will use the Manually attach CLI-created cluster(see page 541) document.

Ensure you have access to your EKS clusters

1. Ensure you are connected to your EKS clusters. Enter the following commands for each of your clusters:

```
kubectl config get-contexts
kubectl config use-context <context for first eks cluster>
```

2. Confirm `kubectl` can access the EKS cluster:

```
kubectl get nodes
```

Create a kubeconfig file for your EKS cluster

If you already have a `kubeconfig` file, **SKIP this section**.

To get started, ensure you have kubectl[455] set up and configured with ClusterAdmin[456] for the cluster you want to connect to Kommander.

1. Create the necessary service account:

451 https://aws.amazon.com/eks/
452 https://d2iq.atlassian.net/wiki/spaces/DENT/pages/29892245
453 https://docs.aws.amazon.com/eks/latest/userguide/install-aws-iam-authenticator.html
454 https://aws.amazon.com/eks/
455 https://kubernetes.io/docs/tasks/tools/#kubectl
456 https://kubernetes.io/docs/concepts/cluster-administration/

```
kubectl -n kube-system create serviceaccount kommander-cluster-admin
```

2. Create a token secret for the `serviceaccount`:

```
kubectl -n kube-system create  -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
  name: kommander-cluster-admin-sa-token
  annotations:
    kubernetes.io/service-account.name: kommander-cluster-admin
type: kubernetes.io/service-account-token
EOF
```

3. Verify that the `serviceaccount` token is ready by running this command:

```
kubectl -n kube-system get secret kommander-cluster-admin-sa-token -oyaml
```

Verify that the `data.token` field is populated. The output should be similar to this:

```
apiVersion: v1
data:
  ca.crt: LS0tLS1CRUdJTiBDR...
  namespace: ZGVmYXVsdA==
  token: ZXlKaGJHY2lPaUpTVX...
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: kommander-cluster-admin
    kubernetes.io/service-account.uid: b62bc32e-b502-4654-921d-94a742e273a8
  creationTimestamp: "2022-08-19T13:36:42Z"
  name: kommander-cluster-admin-sa-token
  namespace: default
  resourceVersion: "8554"
  uid: 72c2a4f0-636d-4a70-9f1c-55a75f15e520
type: kubernetes.io/service-account-token
```

4. Configure the new service account for `cluster-admin` permissions:

```
cat << EOF | kubectl apply -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: kommander-cluster-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
```

```
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: kommander-cluster-admin
  namespace: kube-system
EOF
```

5. Set up the following environment variables with the access data that is needed for producing a new kubeconfig file:

```
export USER_TOKEN_VALUE=$(kubectl -n kube-system get secret/kommander-cluster-
admin-sa-token -o=go-template='{{.data.token}}' | base64 --decode)
export CURRENT_CONTEXT=$(kubectl config current-context)
export CURRENT_CLUSTER=$(kubectl config view --raw -o=go-
template='{{range .contexts}}{{if eq .name "'''${CURRENT_CONTEXT}'''"}}
{{ index .context "cluster" }}{{end}}{{end}}')
export CLUSTER_CA=$(kubectl config view --raw -o=go-
template='{{range .clusters}}{{if eq .name "'''${CURRENT_CLUSTER}'''"}}"{{with
index .cluster "certificate-authority-data" }}{{.}}{{end}}"{{ end }}{{ end }}')
export CLUSTER_SERVER=$(kubectl config view --raw -o=go-
template='{{range .clusters}}{{if eq .name "'''${CURRENT_CLUSTER}'''"}}
{{ .cluster.server }}{{end}}{{ end }}')
```

6. Confirm these variables have been set correctly:

```
export -p | grep -E 'USER_TOKEN_VALUE|CURRENT_CONTEXT|CURRENT_CLUSTER|
CLUSTER_CA|CLUSTER_SERVER'
```

7. Generate a kubeconfig file that uses the environment variable values from the previous step:

```
cat << EOF > kommander-cluster-admin-config
apiVersion: v1
kind: Config
current-context: ${CURRENT_CONTEXT}
contexts:
- name: ${CURRENT_CONTEXT}
  context:
    cluster: ${CURRENT_CONTEXT}
    user: kommander-cluster-admin
    namespace: kube-system
clusters:
- name: ${CURRENT_CONTEXT}
  cluster:
    certificate-authority-data: ${CLUSTER_CA}
    server: ${CLUSTER_SERVER}
users:
- name: kommander-cluster-admin
  user:
    token: ${USER_TOKEN_VALUE}
```

```
EOF
```

8. This process produces a file in your current working directory called `kommander-cluster-admin-config`. The contents of this file are used in Kommander to attach the cluster.

   Before importing this configuration, verify the `kubeconfig` file can access the cluster:

   ```
   kubectl --kubeconfig $(pwd)/kommander-cluster-admin-config get all --all-namespaces
   ```

Finalize attaching your cluster from the UI

Now that you have kubeconfig, go to the DKP UI and follow these steps below:

1. From the top menu bar, select your target workspace.
2. On the Dashboard page, select the **Add Cluster** option in the **Actions** dropdown menu at the top right.
3. Select **Attach Cluster**.
4. Select the **No additional networking restrictions** card. Alternatively, if you must use network restrictions, stop following the steps below, and see the instructions on the page Attach a cluster WITH network restrictions(see page 512).
5. Upload the kubeconfig file you created in the previous section (or copy its contents) into the **Cluster Configuration** section.
6. The **Cluster Name** field automatically populates with the name of the cluster in the kubeconfig. You can edit this field with the name you want for your cluster.
7. Add labels to classify your cluster as needed.
8. Select **Create** to attach your cluster.

If a cluster has limited resources to deploy all the federated platform services, it will fail to stay attached in the DKP UI. If this happens, ensure your system has sufficient resources for all pods.

Related information

For information on related topics or procedures, refer to the following:

- Configuring and Running Amazon EKS Clusters[457]
- Installing and Configuring Konvoy v2.0 or above(see page 197)
- Installing and Configuring Kommander v2.0 or above(see page 366)
- Working with Kommander Clusters(see page 502)

8.6.6.8 Attach GKE Cluster

ENTERPRISE

Attach an existing GKE cluster in DKP

You can attach existing Kubernetes clusters to the Management Cluster. After attaching the cluster, you can use the UI to examine and manage(see page 502) this cluster. The following procedure shows how to attach an existing **standard** GKE cluster.

457 https://aws.amazon.com/eks/

Before you begin

This procedure requires the following items and configurations:

- A fully configured and running GKE cluster[458] with a supported Kubernetes version(see page 0), and administrative privileges.
- The current version of DKP Enterprise installed (see page 366)on your cluster.
- Ensure you have installed `kubectl` in your Management cluster.

This procedure assumes you have an existing and spun up GKE cluster with administrator privileges.

Attach GKE Clusters

Ensure you have access to your GKE clusters

1. Ensure you are connected to your GKE clusters. Enter the following commands for each of your clusters:

```
kubectl config get-contexts
kubectl config use-context <context for first gcloud cluster>
```

2. Confirm `kubectl` can access the GKE cluster.

```
kubectl get nodes
```

Configure a kubeconfig file

If you already have a `kubeconfig` file, **SKIP this section**.

To get started, ensure you have kubectl[459] set up and configured with ClusterAdmin[460] for the cluster you want to connect to Kommander.

1. Create the necessary service account:

```
kubectl -n kube-system create serviceaccount kommander-cluster-admin
```

2. Create a token secret for the `serviceaccount`:

458 https://cloud.google.com/kubernetes-engine/docs/concepts/types-of-clusters
459 https://kubernetes.io/docs/tasks/tools/#kubectl
460 https://kubernetes.io/docs/concepts/cluster-administration/

```
kubectl -n kube-system create  -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
  name: kommander-cluster-admin-sa-token
  annotations:
    kubernetes.io/service-account.name: kommander-cluster-admin
type: kubernetes.io/service-account-token
EOF
```

3. Verify that the `serviceaccount` token is ready by running this command:

```
kubectl -n kube-system get secret kommander-cluster-admin-sa-token -oyaml
```

Verify that the `data.token` field is populated. The output should be similar to this:

```
apiVersion: v1
data:
  ca.crt: LS0tLS1CRUdJTiBDR...
  namespace: ZGVmYXVsdA==
  token: ZXlKaGJHY2lPaUpTVX...
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: kommander-cluster-admin
    kubernetes.io/service-account.uid: b62bc32e-b502-4654-921d-94a742e273a8
  creationTimestamp: "2022-08-19T13:36:42Z"
  name: kommander-cluster-admin-sa-token
  namespace: default
  resourceVersion: "8554"
  uid: 72c2a4f0-636d-4a70-9f1c-55a75f15e520
type: kubernetes.io/service-account-token
```

4. Configure the new service account for `cluster-admin` permissions:

```
cat << EOF | kubectl apply -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: kommander-cluster-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: kommander-cluster-admin
  namespace: kube-system
```

```
EOF
```

5. Set up the following environment variables with the access data that is needed for producing a new kubeconfig file:

```
export USER_TOKEN_VALUE=$(kubectl -n kube-system get secret/kommander-cluster-
admin-sa-token -o=go-template='{{.data.token}}' | base64 --decode)
export CURRENT_CONTEXT=$(kubectl config current-context)
export CURRENT_CLUSTER=$(kubectl config view --raw -o=go-
template='{{range .contexts}}{{if eq .name "'''${CURRENT_CONTEXT}'''"}}
{{ index .context "cluster" }}{{end}}{{end}}')
export CLUSTER_CA=$(kubectl config view --raw -o=go-
template='{{range .clusters}}{{if eq .name "'''${CURRENT_CLUSTER}'''"}}"{{with
index .cluster "certificate-authority-data" }}{{.}}{{end}}"{{ end }}{{ end }}')
export CLUSTER_SERVER=$(kubectl config view --raw -o=go-
template='{{range .clusters}}{{if eq .name "'''${CURRENT_CLUSTER}'''"}}
{{ .cluster.server }}{{end}}{{ end }}')
```

6. Confirm these variables have been set correctly:

```
export -p | grep -E 'USER_TOKEN_VALUE|CURRENT_CONTEXT|CURRENT_CLUSTER|
CLUSTER_CA|CLUSTER_SERVER'
```

7. Generate a kubeconfig file that uses the environment variable values from the previous step:

```
cat << EOF > kommander-cluster-admin-config
apiVersion: v1
kind: Config
current-context: ${CURRENT_CONTEXT}
contexts:
- name: ${CURRENT_CONTEXT}
  context:
    cluster: ${CURRENT_CONTEXT}
    user: kommander-cluster-admin
    namespace: kube-system
clusters:
- name: ${CURRENT_CONTEXT}
  cluster:
    certificate-authority-data: ${CLUSTER_CA}
    server: ${CLUSTER_SERVER}
users:
- name: kommander-cluster-admin
  user:
    token: ${USER_TOKEN_VALUE}
EOF
```

8. This process produces a file in your current working directory called `kommander-cluster-admin-config`. The contents of this file are used in Kommander to attach the cluster.

   Before importing this configuration, verify the `kubeconfig` file can access the cluster:

```
kubectl --kubeconfig $(pwd)/kommander-cluster-admin-config get all --all-
namespaces
```

Attach the cluster

Now that you have a kubeconfig file, go to the DKP UI and follow these steps:

1. Select your target workspace from the top menu bar.
2. Select **Add Cluster** in the **Actions** dropdown menu from the Dashboard page, located at the top-right.
3. Select **Attach Cluster**.
4. Select the **No additional networking restrictions** card.
   Alternatively, if you must use network restrictions, stop following the steps below, and see the instructions on the page Attach a cluster WITH network restrictions(see page 512).
5. Upload the kubeconfig file you created in the previous section (or copy its contents) into the **Cluster Configuration** section.
6. The **Cluster Name** field automatically populates with the name of the cluster in the kubeconfig. You can edit this field with the name you want for your cluster.
7. Add labels to classify your cluster as needed.
8. Select **Create** to attach your cluster.

If a cluster has limited resources to deploy all the federated platform services, it will fail to stay attached in the DKP UI. If this happens, ensure your system has sufficient resources for all pods.

Related information

For information on related topics or procedures, refer to the following:

- Installing and Configuring Kommander(see page 366)
- Working with Kommander Clusters(see page 502)

8.6.6.9 Manually attach CLI-created cluster

ENTERPRISE

Manually attach a cluster that was created with the CLI

When you create a cluster in a Workspace namespace using the DKP CLI, it does not attach automatically. To automatically attach a cluster, generate the cluster objects using the DKP CLI `--dry-run -o yaml` flags and create a cluster as stated in the Advanced Creation of CLI Clusters(see page 543) guide.

However, if you created the cluster using the CLI, you will still need to attach it. You will be able to see the new cluster in the UI while it is being provisioned. Once provisioning is completed, the status will change to Unattached.

Manually attach a cluster that was created with the CLI

From the CLI find out the `name` of the created `Cluster` so you can reference it later:

```
$ kubectl -n <workspace_namespace> get clusters
```

Attach the cluster by creating a `KommanderCluster`:

```
cat << EOF | kubectl apply -f -
apiVersion: kommander.mesosphere.io/v1beta1
kind: KommanderCluster
metadata:
  name: <cluster_name>
  namespace: <workspace_namespace>
spec:
  kubeconfigRef:
    name: <cluster_name>-kubeconfig
  clusterRef:
    capiCluster:
      name: <cluster_name>
EOF
```

8.6.6.10 Access a Managed Cluster

ENTERPRISE

How to access an attached (managed) cluster with credentials

Accessing your managed clusters using your UI administrator credentials

After the cluster is successfully attached, you can retrieve a custom kubeconfig file from the UI.

1. Select the Kommander username in the top right corner, and then select **Generate Token**.
2. Select the attached cluster name, and follow the instructions to assemble a kubeconfig for accessing its Kubernetes API. If Kommander prompts you to log in, use the credentials used to first login to the UI.

You can also retrieve a custom kubeconfig file by visiting the `/token` endpoint on the Kommander cluster domain (example URL: `https://your-server-name.your-region-2.elb.service.com/token/`. Selecting the attached cluster name displays the instructions to assemble a kubeconfig for accessing its Kubernetes API.

8.6.7 Creating DKP Clusters on AWS

A guide for creating DKP clusters on AWS

8.6.7.1 Before you begin

- Configure an AWS Infrastructure(see page 111), or add credentials using AWS role credentials(see page 405).

8.6.7.2 Simplified Cluster Creation on AWS

1. In the selected workspace Dashboard, select the **Add Cluster** option in the **Actions** dropdown menu at the top right.
2. On the Add Cluster page, select the **Create DKP Cluster**.

3. Provide some basic cluster details within the form:
    - **Workspace**: The workspace where this cluster belongs (if within the Global workspace).
    - **Kubernetes Version**: The initial version of Kubernetes to install on the cluster.
    - **Name**: A valid Kubernetes name for the cluster.
    - **Add Labels**: By default, your cluster has labels that reflect the infrastructure provider provisioning. For example, your AWS cluster may have a label for the data center region and `provider: aws`. Cluster labels are matched to the selectors created for Projects(see page 459). Changing a cluster label may add or remove the cluster from projects.
4. Select the pre-configured AWS Infrastructure(see page 111), or AWS role credentials(see page 405) to display the remaining options specific to AWS.
    - **Region**: Select a data center region or specify a custom region.
    - **Configure Node Pools**: Specify pools of nodes, their machine types, quantity, and the IAM instance profile.
        - **Machine Type**: Machine instance type.
        - **Quantity**: Number of nodes. The control plane must be an odd number.
        - **IAM instance profile**: Name of the IAM instance profile to assign to the machines.
    - **AMI Image Lookup**: You can also specify the base OS, lookup format, and owner ID of the AMI Image, or the AMI ID as part of each pool. Selecting 'Show Advanced' within the Configure Node Pools section will show these options.
        - **Base OS**: Base OS for Lookup search.
        - **Lookup Format**: Lookup Format string to generate AMI search name from.
        - **Owner ID**: Owner ID for AMI Lookup search.
        - **AMI ID**: AMI ID to use for all nodes.
    - **Add Infrastructure Provider Tags**: Specify tags applied on all resources created in your infrastructure for this cluster. Different infrastructure providers have varying restrictions on the usable tags. See the AWS Tags User Guide[461] for more information on using tags in AWS.
5. Select **Create** to begin provisioning the cluster. This step may take a few minutes, taking time for the cluster to be ready and fully deploy its components. The cluster automatically tries to join, and should resolve once it is fully-provisioned.

8.6.8 Advanced Creation of CLI Clusters

Create CLI clusters

⚠ This feature is for advanced users and users in unique environments only. We highly recommend using other documented methods to create clusters whenever possible.

8.6.8.1 Generate cluster objects

Depending on your infrastructure, DKP CLI can generate a set of cluster objects that can be customized for unusual use cases. Visit the Advanced Configuration documentation for AWS(see page 117) for an example on how to use the `--dry-run` and `--output` flags to create a set of cluster objects.

8.6.8.2 Use the upload YAML form

1. In the selected workspace Dashboard, select the **Add Cluster** option in the **Actions** dropdown menu at the top right.

461 https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Using_Tags.html

2. On the Add Cluster page, select **Upload YAML to Create a Cluster** and provide advanced cluster details:
   - **Workspace**: The workspace where this cluster belongs (if within the Global workspace).
   - **Cluster YAML**: Paste or upload your customized set of cluster objects into this field. Only valid YAML is accepted.
   - **Add Labels**: By default, your cluster has labels that reflect the infrastructure provider provisioning. For example, your AWS cluster may have a label for the data center region and `provider: aws`. Cluster labels are matched to the selectors created for Projects(see page 459). Changing a cluster label may add or remove the cluster from projects.
3. Click **Create** to begin provisioning the DKP CLI cluster. This step may take a few minutes, taking time for the cluster to be ready and fully deploy its components. The cluster automatically tries to join, and should resolve after it is fully-provisioned.

8.6.9 Management Cluster

A guide for the Management Cluster and the Management Cluster Workspace

When you install Kommander, the host cluster is attached in the **Management Cluster Workspace** as the **Management Cluster**, called **Management Cluster** in the Global workspace dashboard and **Kommander Host** inside the Management Cluster Workspace. This allows the Management Cluster to be included in Projects(see page 459) and enables the management of its Platform Applications(see page 481) from the Management Cluster Workspace.

Do not attach a cluster in the "Management Cluster Workspace" workspace. This workspace is reserved for your Kommander Management cluster only.

8.6.9.1 Editing

As an attached cluster, you can edit the Management Cluster to add or remove Labels, then you can use these labels to include the Management Cluster in Projects inside of the Management Cluster Workspace.

8.6.9.2 Disconnecting

The Management Cluster cannot be disconnected from the GUI like other attached clusters. Because of this, the Management Cluster Workspace cannot be deleted from the GUI as it will always have the Management Cluster inside itself.

8.6.10 Cluster Applications

Applications installed on your cluster

8.6.10.1 Application Dashboards

Applications, formerly called Platform Services and Addons, are installed by the management cluster. You can visit a cluster's detail page to see the application dashboards enabled from the deployed applications under the **Application Dashboards** section.

8.6.10.2 Applications

Under the **Applications** section of the cluster's detail page, you can view the workspace applications enabled for the cluster, grouped by category.

In this section, you can also view the current status of the enabled applications on the cluster, on each application card. Hovering on the status displays details about the status of the application.

Cluster applications can have one of the following statuses:

Status	Description
Enabled	The application is enabled, but the status on the cluster is not available.
Pending	The application is waiting to be deployed.
Deploying	The application is currently being deployed to the cluster.
Deployed	The application has successfully been deployed to the cluster.
Deploy Failed	The application failed to deploy to the cluster.

Review the Workspace Platform Application Configuration Requirements(see page 430) to ensure that the attached clusters have sufficient resources. For more information on applications and how to customize them, see Workspace Applications(see page 434).

8.6.10.3 Custom Cluster Application Dashboard Cards

Define custom application dashboard cards displayed on a cluster's detail page.

Custom Cluster Application Dashboard Cards

You can add custom application dashboard cards to the cluster detail page's Applications section by creating a `ConfigMap` on the cluster. The `ConfigMap` must have a `kommander.d2iq.io/application` label applied through the CLI, and must contain both `name` and `dashboardLink` data keys to be displayed. Upon creation of the `ConfigMap`, the DKP UI displays a card corresponding to the data provided in the `ConfigMap`. Custom application cards have a Kubernetes icon, and can link to a service running in the cluster, or use an absolute URL to link to any accessible URL.

ConfigMap example

```
apiVersion: v1
```

```
kind: ConfigMap
metadata:
  name: "my-app"
  namespace: "app-namespace"
  labels:
    "kommander.d2iq.io/application": "my-app"
data:
  name: "My Application"
  dashboardLink: "/path/to/app"
```

Key	Description	Required
`metadata.labels."kommander.d2iq.io/application"`	The application name (ID).	X
`data.name`	The display name that describes the application and displays on the custom application card in the user interface.	X
`data.dashboardLink`	The link to the application. This can be an absolute link, `https://www.d2iq.com` or a relative link, `/dkp/kommander/dashboard.` If you use a relative link, the link is built using the cluster's path as the base of the URL to the application.	X
`data.docsLink`	Link to documentation about the application. This is displayed on the application card, but omitted if not present.	
`data.category`	Category with which to group the custom application. If not provided, the application is grouped under the category, "None."	
`data.version`	A version string for the application. If not provided, "N/A" is displayed on the application card in the user interface.	

Use a command similar to this to create a new custom application `ConfigMap`:

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: ConfigMap
metadata:
  name: "my-app"
  namespace: "default"
  labels:
    "kommander.d2iq.io/application": "my-app"
data:
  name: "My Application"
  dashboardLink: "/path/to/app"
EOF
```

8.6.11 Custom domains and certificates configuration

Configure a custom domain and certificate for your Management or any Managed/Attached cluster

DKP supports configuring a custom domain name per cluster, so you can access the DKP UI and other platform services via that domain. Additionally, you can provide a custom certificate for the domain, or one can be issued automatically by Let's Encrypt (or other certificate authorities supporting the ACME protocol).

The configuration path is the same regardless of whether you are configuring a custom domain and certificate on the Management cluster, or a Workload (Managed or Attached) cluster. However, you can choose to set up a customized domain and certificate for the Management cluster during DKP installation.

- Customize a domain or certificate in the **Management Cluster**(see page 548), during installation(see page 355).
- Customize a domain or certificate in the **Management or a Managed/Attached Cluster**, after installation(see page 548).

8.6.11.1 Reasons for using a Custom DNS Domain

DKP supports the customization of domains to allow you to use your own domain or hostname for your services. For example, you can set up your DKP UI or any of your clusters to be accessible with your custom domain name instead of the domain provided by default.

8.6.11.2 Reasons for using a Custom Certificate

DKP's default CA identity supports the encryption of data exchange and traffic (between your client and your environment's server). To configure an additional security layer that validates your environment's server authenticity, DKP supports configuring a custom certificate issued by a trusted Certificate Authority either directly in a Secret or managed automatically using the ACME protocol (for example, Let's Encrypt).

Changing the default certificate for any of your clusters can be helpful. For example, you can adapt it to classify your DKP UI or any other type of service as trusted (when accessing a service via a browser).

Using Let's Encrypt or other ACME certificate authorities does not work in air-gapped scenarios, as these services require connection to the Internet for their setup. For air-gapped environments, you can either use self-signed certificates issued by the cluster (the default configuration), or a certificate created manually using a trusted Certificate Authority.

8.6.11.3 Configure a custom domain and certificate for your cluster

Configure a custom domain and certificate in the Management and any Managed or Attached Clusters

This section describes how to enable custom domains and certificates in your **Management and any Managed or Attached** clusters after the installation of DKP. If you want to set up a custom domain and certificate for your Management Cluster **during the installation**, refer to the Customize a domain and certificate during installation(see page 355) documentation.

To customize the domain or certificate for a specific cluster after the installation of DKP, adapt or create an API YAML to specify the custom domain and certification details.

Configure Custom Domains or Custom Certificates

⚠ Ensure your `dkp` configuration references the Management cluster of the environment where you want to customize the domain or certificate by setting the `KUBECONFIG=` environment variable, or using the `--kubeconfig` flag, in accordance with Kubernetes conventions[462].

To customize the domain or certificate of a cluster, alter the `spec` values of the `ingress` object in the `KommanderCluster` resource. Note that you can reference an issuer as an `issuerRef` **OR** a secret as a `certificateSecretRef`, as long as the object is created in the cluster where you want to customize the configuration.

In the Management cluster, both the `KommanderCluster` and `issuerRef` or `certificateSecretRef` objects are on the same cluster. In Managed and Attached clusters, the `KommanderCluster` object is stored on the Management cluster, and the `issuerRef` or `certificateSecretRef` object is on the Managed or Attached cluster.

Use the API YAML to customize the domain (via the `hostname` field), and the certificate (via the `issuerRef` or `certificateSecretRef` field).

You have two options to update and apply the `KommanderCluster` resource with the required ingress. Refer to the following examples:

1. One option is to use a certificate that is managed automatically and supported by cert-manager like ACME (if you use Let's Encrypt, refer to the example below(see page 0). For this, reference the **name of the** `Issuer` or `ClusterIssuer` that contains your ACME provider information in the `issuerRef` field, and enter the custom domain name in the `hostname` field of the target cluster:

```
cat <<EOF | kubectl -n <workspace_namespace> --kubeconfig
<management_cluster_kubeconfig> patch \
kommandercluster <cluster_name>  --type='merge' --patch-file=/dev/stdin
spec:
```

462 https://kubernetes.io/docs/tasks/access-application-cluster/configure-access-multiple-clusters/

```
  ingress:
    hostname: <cluster_hostname>
    issuerRef:
      name: <issuer_name>
      kind: ClusterIssuer # or Issuer depending on the issuer config
```

2. Another option is to use a manually created certificate that is **customized for your hostname**. To do so, create a TLS Secret holding the certificate[463] on the target cluster. Reference that secret in the `certificateSecretRef` field and the custom domain name in the `hostname` field of the target cluster:

```
kubectl create secret generic -n "${WORKSPACE_NAMESPACE}" domain-tls-certs \
  --from-file=ca.crt=$CERT_CA_PATH \
  --from-file=tls.crt=$CERT_PATH \
  --from-file=tls.key=$CERT_KEY_PATH \
  --type=kubernetes.io/tls
```

It is not possible to configure the namespace of the secret with a command. Ensure the secret is stored in the workspace namespace of the target cluster.

Example: Configure a custom certificate with Let's Encrypt

Let's Encrypt is one of the Certificate Authorities (CA) supported by cert-manager. To set up a Let's Encrypt certificate, create an `Issuer` or `ClusterIssuer` in the target cluster and then reference it in the `issuerRef` field of the `KommanderCluster` resource.

1. Create the Let's Encrypt ACME cert-manager issuer:

```
cat <<EOF | kubectl apply -f -
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: custom-acme-issuer
spec:
  acme:
    email: <your_email>
    server: https://acme-v02.api.letsencrypt.org/directory
    privateKeySecretRef:
      name: kommander-acme-issuer-account
    solvers:
      - dns01:
          route53:
            region: us-east-1
            role: arn:aws:iam::YYYYYYYYYYYY:role/dns-manager
```

463 https://kubernetes.io/docs/concepts/configuration/secret/#tls-secrets

```
EOF
```

2. Configure the Management cluster to use your `custom-domain.example.com` with a certificate issued by Let's Encrypt by referencing the created `ClusterIssuer`:

```
cat <<EOF | kubectl -n kommander --kubeconfig <management_cluster_kubeconfig>
patch \ kommandercluster host-cluster  --type='merge' --patch-file=/dev/stdin
spec:
  ingress:
    hostname: custom-domain.example.com
    issuerRef:
      name: custom-acme-issuer
      kind: ClusterIssuer
EOF
```

Verify and Troubleshoot Configuration Status

If you want to ensure the customization for a domain and certificate is completed, or if you want to obtain more information on the status of the customization, display the status information for the `KommanderCluster`.

Inspect the modified `KommanderCluster` object:

```
kubectl describe kommandercluster -n <workspace_name> <cluster_name>
```

If the ingress is still being provisioned, the output looks similar to this:

```
[...]
Conditions:
  Last Transition Time:  2022-06-24T07:48:31Z
  Message:               Ingress service object was not found in the cluster
  Reason:                IngressServiceNotFound
  Status:                False
  Type:                  IngressAddressReady
[...]
```

If the provisioning has been completed, the output looks similar to this:

```
[...]
Conditions:
  Last Transition Time:  2022-06-28T13:43:33Z
  Message:               Ingress service address has been provisioned
  Reason:                IngressServiceAddressFound
  Status:                True
  Type:                  IngressAddressReady
  Last Transition Time:  2022-06-28T13:42:24Z
  Message:               Certificate is up to date and has not expired
```

```
  Reason:                Ready
  Status:                True
  Type:                  IngressCertificateReady
[...]
```

The same command also prints the actual customized values for the `KommanderCluster.Status.Ingress`. Here is an example:

```
[...]
   ingress:
    address: 172.20.255.180
    caBundle: LS0tLS1CRUdJTiBD...<output has been shortened>...DQVRFLS0tLS0K
[...]
```

8.6.12 Disconnect or Delete Clusters

8.6.12.1 **Disconnect or delete a cluster**

8.6.12.2 Disconnect vs. delete

When you attach a cluster to Kommander that was not created with Kommander, you can later disconnect it. This does not alter the running state of the cluster, but simply removes it from the DKP UI. User workloads, platform services, and other Kubernetes resources are not cleaned up at detach.

⚠ After successfully detaching the cluster, manually disconnect the attached cluster's Flux installation from the management Git repository. Otherwise, changes to apps in the managed cluster's workspace will still be reflected on the cluster you just detached. Ensure your `dkp` configuration references the cluster, where you want to run the upgrade. You can do this by setting the `KUBECONFIG` environment variable to the appropriate kubeconfig file location[464]. An alternative to initializing the `KUBECONFIG` environment variable is to use the `-kubeconfig=cluster_name.conf` flag. Then, run `kubectl -n kommander-flux patch gitrepo management -p '{"spec":{"suspend":true}}' --type merge` to make the cluster's workloads not managed by Kommander, anymore.

If you created the managed clusters with Kommander, you cannot disconnect the cluster, but you can delete the cluster. This completely removes the cluster and all of its cloud assets.

⚠ If you delete the management (Konvoy) cluster, you can not use Kommander to delete any managed clusters created by Kommander. If you want to delete all clusters, ensure you delete any managed clusters before finally deleting the management cluster.

464 https://kubernetes.io/docs/tasks/access-application-cluster/configure-access-multiple-clusters/

Statuses

See Statuses(see page 502) for a list of possible states a cluster can have when it is getting disconnected or deleted.

8.6.12.3 Troubleshooting

I cannot detach an attached cluster that is “Pending”

Sometimes attaching a Kubernetes cluster causes that cluster to get stuck in the “Pending” state. This can happen because the wrong `kubeconfig` file is used or the cluster is just not reachable. In order to detach the cluster, so it does not show in the UI, follow these steps:

1. Determine the `KommanderCluster` resource backing the cluster you tried to attach. Enter the following command:

   ```
   kubectl -n WORKSPACE_NAMESPACE get kommandercluster
   ```

   Replace `WORKSPACE_NAMESPACE` with the actual current workspace name. You can find this name by going to `https://YOUR_CLUSTER_DOMAIN_OR_IP_ADDRESS/dkp/kommander/dashboard/workspaces` in your browser.
2. Delete the cluster. Enter the following command:

   ```
   kubectl -n WORKSPACE_NAMESPACE delete kommandercluster CLUSTER_NAME
   ```
3. If the resource does not go after a short time, remove its finalizers. Enter the following command:

   ```
   kubectl -n WORKSPACE_NAMESPACE patch kommandercluster CLUSTER_NAME --type json
   -p '[{"op":"remove", "path":"/metadata/finalizers"}]'
   ```

   This removes the cluster from the DKP UI.

8.7 Backup and Restore

For production clusters, regular maintenance should include routine backup operations to ensure data integrity and reduce the risk of data loss due to unexpected events. Backup operations should include the cluster state, application state, and the running configuration of both stateless and stateful applications in the cluster.

DKP stores all data as CRDs in the Kubernetes API and you can back up and restore it using the following procedure.

8.7.1 Velero

DKP provides Velero[465] by default, to support backup and restore operations for your Kubernetes clusters and persistent volumes.

465 https://velero.io/

8.7.1.1 MinIO

For on-premises deployments, DKP deploys Velero integrated with MinIO[466], operating inside the same cluster.

For production use-cases, D2iQ advises to provide an *external* storage class to use with MinIO[467].

You can customize(see page 0) your Velero instance in two ways: You can add the values below to the configuration file when installing(see page 351) DKP, or you can apply the configuration after the cluster is configured by running `kubectl apply -f`.

To specify an external storageClass for the **MinIO** instances, create a file called `velero-overrides.yaml` with the following content:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: velero-overrides
  namespace: kommander
data:
  values.yaml: |
     minio:
      persistence:
         storageClass: <external storage class name>
```

8.7.1.2 Amazon S3

You can also store your backups in **Amazon S3**. To do so, create a file called `velero-overrides.yaml` with the following content:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: velero-overrides
  namespace: kommander
data:
  values.yaml: |
    minioBackend: false
    configuration:
      backupStorageLocation:
        # `name:` must be empty
        bucket: <BUCKET_NAME>
        config:
          region: <AWS_REGION> # such as us-west-2
          s3ForcePathStyle: "false"
          insecureSkipTLSVerify: "false"
```

466 https://velero.io/docs/v1.0.0/get-started/
467 https://docs.min.io/

```
          s3Url: ""
          # profile should be set to the AWS profile name mentioned in the
secretContents below
          profile: default
    credentials:
      # With the proper IAM permissions with access to the S3 bucket,
      # you can attach the EC2 instances using the IAM Role, OR fill in
`existingSecret` OR `secretContents` below.
      #
      # Name of a pre-existing secret (if any) in the Velero namespace
      # that should be used to get IAM account credentials.
      existingSecret:
      # The key must be named `cloud`, and the value corresponds to the entire
content of your IAM credentials file.
      # For more information, consult the documentation for the velero plugin for AWS
at:
      # [AWS] https://github.com/vmware-tanzu/velero-plugin-for-aws/blob/main/
README.md
      secretContents:
        # cloud: |
        #   [default]
        #   aws_access_key_id=<REDACTED>
        #   aws_secret_access_key=<REDACTED>
```

8.7.1.3 Azure Blob

Prerequisites: Create Azure related assets such as storage account, blob containers, resource group, roles, service principals prior to continuing.

1. Prep your credentials-velero file with the values:

```
cat << EOF > ./credentials-velero
AZURE_SUBSCRIPTION_ID=${AZURE_SUBSCRIPTION_ID}
AZURE_TENANT_ID=${AZURE_TENANT_ID}
AZURE_CLIENT_ID=${AZURE_CLIENT_ID}
AZURE_CLIENT_SECRET=${AZURE_CLIENT_SECRET}
AZURE_RESOURCE_GROUP=${AZURE_RESOURCE_GROUP}
AZURE_CLOUD_NAME=AzurePublicCloud
EOF
```

2. Use the credentials-velero file to create the secret (in your case you named it as azure-bsl-credentials). Note that we used --from-env-file referencing the credentials-velero file.

```
kubectl create secret generic -n kommander azure-bsl-credentials --from-env-
file="credentials-velero" --kubeconfig=${CLUSTER_NAME}.conf
```

3. Create the backup storage location via Velero CLI

```
velero backup-location create -n kommander azure \
--provider azure \
--bucket $BLOB_CONTAINER \
--config
resourceGroup=$AZURE_BACKUP_RESOURCE_GROUP,storageAccount=$AZURE_STORAGE_ACCOUN
T_ID \
--credential=azure-bsl-credentials=azure --kubeconfig=${CLUSTER_NAME}.conf
```

4. Check to make sure the Azure backup location is created:

```
velero backup-location get -n kommander --kubeconfig=${CLUSTER_NAME}.conf
```

5. Configure DKP to load the plugins and to include the secret:
NOTE: This process has been tested to work with AWS v1.1.0 and Azure v1.1.2. Newer versions of these plugins can be used, but have not been tested by D2iQ.

```
:
velero:
    values: |
      minioBackend: false
      initContainers:
        - name: initialize-velero
          image: mesosphere/kubeaddons-addon-initializer:v0.5.5
          args: ["velero"]
          env:
            - name: "MINIO_INGRESS_NAMESPACE"
              value: kommander
            - name: "MINIO_INGRESS_SERVICE_NAME"
              value: kommander-traefik
            - name: "VELERO_NAMESPACE"
              value: kommander
            - name: "VELERO_MINIO_FALLBACK_SECRET_NAME"
              value: "velero-d2iq-credentials"
        - name: velero-plugin-for-aws
          image: velero/velero-plugin-for-aws:v1.1.0
          imagePullPolicy: IfNotPresent
          volumeMounts:
            - mountPath: /target
              name: plugins
        - name: velero-plugin-for-microsoft-azure
          image: velero/velero-plugin-for-microsoft-azure:v1.1.2
          imagePullPolicy: IfNotPresent
          volumeMounts:
            - mountPath: /target
              name: plugins
      credentials:
          extraSecretRef: azure-bsl-credentials
```

```
:
```

6. Check the Helm releases that the new Velero configuration applied/loaded:

```
kubectl get hr -n kommander --kubeconfig=${CLUSTER_NAME}.conf
```

7. Make sure that the Velero pod is running:

```
kubectl get pods -A --kubeconfig=${CLUSTER_NAME}.conf |grep velero
```

8. And connect to it via exec:

```
kubectl -n kommander exec -it velero-<hash> --kubeconfig=${CLUSTER_NAME}.conf
bash
```

9. Check the plugin and the env variables. The env variable should contain the contents of the `credentials-velero` file.

```
cd plugins
ls -l
env |grep AZURE
```

10. Create a test backup:

```
velero backup create azure-velero-testbackup -n kommander --kubeconfig=$
{CLUSTER_NAME}.conf --storage-location azure
```

8.7.2 Patch Velero

After you have created the `ConfigMap` with **MinIO**, **Amazon S3**, or **Azure Blob**, patch the Velero `AppDeployment` by adding the `configOverrides` value. This applies the `ConfigMap` to your instance after the cluster has been configured:

```
cat << EOF | kubectl -n kommander patch appdeployment velero --type="merge" --patch-
file=/dev/stdin
spec:
  configOverrides:
    name: velero-overrides
EOF
```

8.7.3 Install the Velero CLI

Although installing the Velero command-line interface is optional and independent of deploying a DKP cluster, having access to the command-line interface provides several benefits. For example, you can use the Velero

command-line interface to back up or restore a cluster on demand, or to modify certain settings without changing the Velero configuration.

- By default, DKP sets up Velero to use MinIO over TLS using a self-signed certificate.
- As a result, when using certain commands, you may be asked to use the `--insecure-skip-tls-verify` flag.

Again, the default setup is not suitable for production use-cases.

See the instructions to install the Velero command-line interface[468] for more information.

In DKP, the Velero platform application is installed in the `kommander` namespace, instead of `velero`. Thus, after installing the CLI, we recommend that you set the Velero CLI `namespace` config option so that subsequent Velero CLI invocations will use the correct namespace:

```
velero client config set namespace=kommander
```

8.7.4 Regular backup operations

For backing up production clusters, you should be familiar with the following basic administrative functions Velero provides:

- Set a backup schedule(see page 0)
- Run on-demand backups(see page 0)
- Restore from a backup archive(see page 558)

8.7.4.1 Set a backup schedule

By default, DKP configures a regular, automatic backup of the cluster's state in Velero. The default settings do the following:

- Create daily backups
- Save the data from all namespaces

These default settings take effect after the cluster is created. If you install DKP with the default platform services deployed, the initial backup starts after the cluster is successfully provisioned and ready for use.

Alternate backup schedules

The Velero CLI provides an easy way to create alternate backup schedules. For example, you can use a command similar to:

```
velero create schedule thrice-daily --schedule="@every 8h"
```

To change the default backup service settings:

1. Check the backup schedules currently configured for the cluster by running the following command:

468 https://velero.io/docs/v1.5/basic-install/#install-the-cli

```
velero get schedules
```

2. Delete the `velero-default` schedule by running the following command:

```
velero delete schedule velero-default
```

3. Replace the default schedule with your custom settings by running the following command:

```
velero create schedule velero-default --schedule="@every 24h"
```

You can also create backup schedules for specific namespaces. Creating a backup for a specific namespace can be useful for clusters running multiple apps operated by multiple teams. For example:

```
velero create schedule system-critical --include-namespaces=kube-system,kube-public,k
ommander --schedule="@every 24h"
```

The Velero command line interface provides many more options worth exploring. You can also find tutorials for disaster recovery[469] and cluster migration[470] on the Velero community site.

8.7.4.2 Back up on demand

In some cases, you might find it necessary create a backup outside of the regularly-scheduled interval. For example, if you are preparing to upgrade any components or modify your cluster configuration, you should perform a backup immediately before taking that action.

Create a backup by running the following command:

```
velero backup create BACKUP-NAME
```

8.7.5 Restore a cluster from backup

Before attempting to restore the cluster state using the Velero command-line interface, you should verify the following requirements:

- The backend storage, MinIO, is still operational.
- The Velero platform service in the cluster is still operational.
- The Velero platform service is set to a `restore-only-mode` to avoid having backups run while restoring.

To list the available backup archives for your cluster, run the following command:

```
velero backup get
```

469 https://velero.io/docs/v0.11.0/disaster-case
470 https://velero.io/docs/v0.11.0/migration-case

To set Velero to a `restore-only-mode`, run the following command:

```
velero server --restore-only=true
```

This mode is being deprecated and will be removed in Velero in Velero v2.0. Use read-only backup storage locations instead.

Finally, check your deployment to verify that the configuration change was applied correctly:

```
helm get values -n kommander velero
```

To restore cluster data on demand from a selected backup snapshot available in the cluster, run a command similar to the following:

```
velero restore create --from-backup BACKUP-NAME
```

8.7.5.1 Restore your DKP Management Cluster

When restoring a backup to the Management Cluster, you must adjust configuration to avoid restore errors.

Before restoring a backup, ensure the following `ResourceQuota` setup is not configured on your cluster:

When completed, the manually removed `ResourceQuota` named `one-kommandercluster-per-kommander-workspace` restores automatically.

Turn off the `Workspace` validation webhooks. Otherwise, you will not restore Workspaces with pre-configured namespaces. If the validation webhook named `kommander-validating` is present, it must be modified with the command:

```
kubectl patch validatingwebhookconfigurations kommander-validating \
  --type json \
  --patch '[
    {
      "op": "remove",
      "path": "/webhooks/0/rules/3/operations/0"
    }
  ]'
```

When the backup completes, re-add the removed `CREATE` webhook rule operation with:

```
kubectl patch validatingwebhookconfigurations kommander-validating \
```

```
  --type json \
  --patch '[
    {
      "op": "add",
      "path": "/webhooks/0/rules/3/operations/0",
      "value": "CREATE"
    }
  ]'
```

8.7.6 Backup service diagnostics

You can check whether the Velero service is currently running on your cluster through the Kubernetes dashboard (accessible through the DKP UI on the Management Cluster), or by running the following `kubectl` command:

```
kubectl get all -A | grep velero
```

If the Velero platform service application is currently running, you can generate diagnostic information about Velero backup and restore operations. For example, you can run the following commands to retrieve, back up, and restore information that you can use to assess the overall health of Velero in your cluster:

```
velero get schedules
velero get backups
velero get restores
velero get backup-locations
velero get snapshot-locations
```

8.8 Logging

D2iQ Kubernetes Platform (DKP) ships with a pre-configured logging stack that allows you to collect and visualize pod and admin log data at the Workspace level. The logging stack is also multi-tenant capable. Multi-tenancy is enabled at the Project level through role-based access control (RBAC).

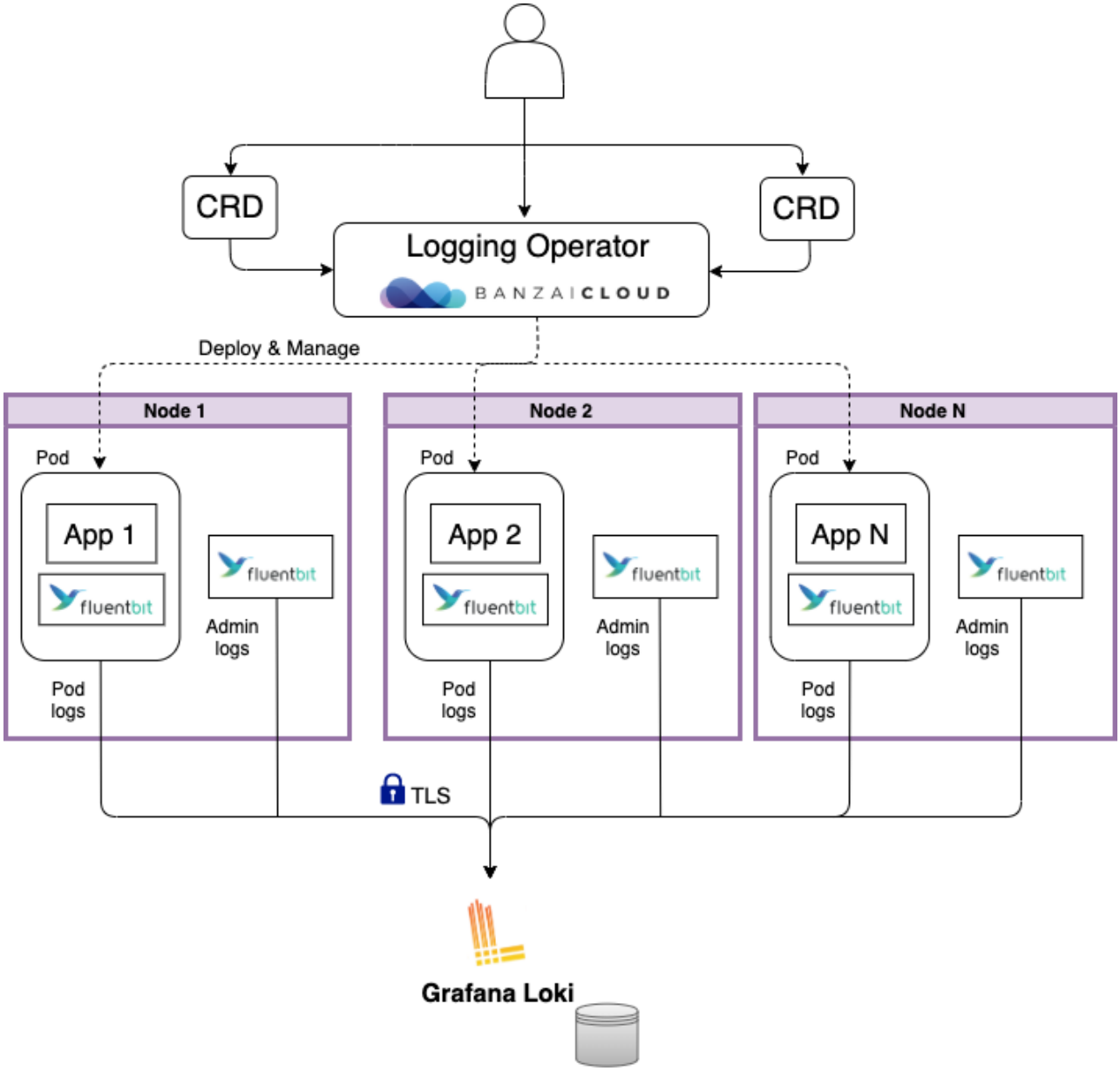
By default, logging is disabled on managed and attached clusters. You need to enable the logging stack applications explicitly on the workspace to make use of these capabilities.

The primary components of the logging stack include these platform services:

- BanzaiCloud Logging-operator
- Grafana and Grafana Loki
- Fluentbit and Fluentd

In addition to these platform services, logging relies on other software and system facilities, including the container runtime, the journald facility, and systemd configuration are used to collect logs and messages from all the machines in the cluster.

The following diagram illustrates how different components of the logging stack collect log data and provide information about clusters:



The DKP logging stack aggregates logs from applications and nodes running inside your cluster.

DKP uses the BanzaiCloud Logging-operator to manage the Fluentbit and Fluentd deployments that collect pod logs, using Kubernetes API extensions called custom resources. The custom resources allow users to declare logging configurations using `kubectl` commands. The Fluentbit instance deployed by Logging-operator gathers pod logs data and sends it to Fluentd, which forwards it to the appropriate Grafana Loki servers based on the configuration defined in custom resources.

Loki then indexes the log data by label and stores it for querying. Loki maintains log order integrity but does not index the log messages themselves, which improves its efficiency and lowers its footprint.

8.8.1 Admin-level logs

DKP also includes a Fluentbit instance to collect admin-level log information which is sent to the workspace Grafana Loki that's running on the cluster. The admin log information includes:

- Logs for host processes managed by systemd
- Kernel logs
- Kubernetes audit logs

This approach helps to isolate the more sensitive logs from Logging-operator, eliminating the possibility that users might gain inadvertent access to that data.

See Fluent Bit(see page 579) for more information about these logs.

⚠ On the Management cluster, the Fluentbit application is disabled by default. The amount of admin logs ingested to Loki requires additional disk space to be configured on the `grafana-loki-minio` Minio Tenant. Enabling admin logs may use around 2GB/day per node. See https://d2iq.atlassian.net/wiki/spaces/DENT/pages/89033391/2.3%2B2.3.0%2BRelease%2BNotes#Configuring-the-Grafana-Loki-Minio-Tenant(see page 0) for more details on how to configure the Minio Tenant.

8.8.2 Enable Workspace-level Logging

8.8.2.1 How to enable Workspace-level Logging for use with DKP.

Logging is disabled by default on managed and attached clusters. You will need to enable logging features explicitly at the Workspace level, if you want to capture and view log data.

⚠ You must perform these procedures to enable multi-tenant logging at the Project level as well.

8.8.2.2 Prerequisites

Before you begin, you must:

- Be a cluster administrator with permissions to configure cluster-level platform services.
- Set a default storage class(see page 508) on each attached cluster for successful Loki deployment.

8.8.2.3 Enable Workspace-level logging

The steps required to enable multi-tenant logging include:

1. Create the AppDeployments to enable logging.(see page 563)
2. Verify that the cluster's logging stack is installed.(see page 568)
3. View a cluster's log data.(see page 569)

To get started with logging, create the AppDeployments using the CLI. You can also use instructions in DKP UI to Enable Logging Applications(see page 563). When Workspace-level logging is fully-configured and operational, then you can configure Kommander to enable Multi-tenant Logging(see page 570), if needed.

8.8.2.4 Enable Logging Applications through the UI

How to enable the logging stack through the UI for Workspace-level logging

You can enable the Workspace logging stack to all attached clusters within the Workspace through the UI. If you prefer to enable the logging stack with `kubectl`, review how you create AppDeployments to Enable Workspace Logging(see page 563).

To enable workspace-level logging in DKP using the UI, follow these steps:

1. From the top menu bar, select your target workspace.
2. Select **Applications** from the sidebar menu.
3. Traefik and cert-manager are required to be deployed for the logging stack. Ensure that Traefik is enabled in the Workspace, and cert-manager is enabled or already deployed to the cluster.
4. Scroll to the **Logging** applications section.
5. Select the three dot button from the bottom-right corner of the card for MinIO, and click **Enable**. On the **Enable Workspace Platform Application** page, you can add a customized configuration for settings that best fit your organization. You can leave the configuration settings unchanged to enable with default settings.
6. Select **Enable** at the top right of the page.
7. Repeat the process for the Grafana Loki, Logging Operator, and Grafana Logging applications.
8. You can verify the cluster logging stack installation by waiting until the cards have a Deployed checkmark on the Cluster Application(see page 544) page, or you can verify the Cluster Logging Stack installation via the CLI(see page 568).
9. Then, you can view cluster log data(see page 569).

⚠ We do not recommend installing Fluent Bit, which is responsible for collecting admin logs, unless you have configured the Grafana Loki Minio Tenant with sufficient storage space. Enabling admin logs via Fluent Bit may use around 2GB/day per node. See https://d2iq.atlassian.net/wiki/spaces/DENT/pages/29923823/2.3.0+Release+Notes#Configuring-the-Grafana-Loki-Minio-Tenant(see page 0) for more details on how to configure the Minio Tenant.

Only for Attached GKE Clusters:
Create a ResourceQuota to enable log collection

At this point, log collection is not enabled for your attached GKE cluster. Create a ResourceQuota to enable log collection(see page 0).

8.8.2.5 Create AppDeployments to Enable Workspace Logging

How to create AppDeployments to enable Workspace-level logging

Workspace logging AppDeployments enable and deploy the logging stack to all attached clusters within the workspace. Use the DKP UI to enable the logging applications, or, alternately, use the CLI to create the AppDeployments.

To enable logging in DKP using the CLI, follow these steps on the **management** cluster:

1. Execute the following command to get the name and namespace of your workspace:

```
dkp get workspaces
```

And copy the values under the `NAME` and `NAMESPACE` columns for your workspace.

2. Export the `WORKSPACE_NAME` variable:

```
export WORKSPACE_NAME=<WORKSPACE_NAME>
```

3. Export the `WORKSPACE_NAMESPACE` variable:

```
export WORKSPACE_NAMESPACE=<WORKSPACE_NAMESPACE>
```

4. Ensure that Cert-Manager and Traefik are enabled in the workspace. If you want to find if the applications are enabled on the management cluster workspace, you can run:

```
dkp get appdeployments --workspace ${WORKSPACE_NAME}
```

5. You can confirm that the applications are deployed on the **managed** or **attached** cluster by running this `kubectl` command in that cluster. (Ensure you switch to the correct context or kubeconfig[471] of the attached cluster for the following `kubectl` command.)

```
kubectl get helmreleases -n ${WORKSPACE_NAMESPACE}
```

6. Copy these commands and execute them on the **management** cluster from a command line to create the Logging-operator, Grafana-loki, and Grafana-logging AppDeployments:

```
dkp create appdeployment logging-operator --app logging-operator-3.17.7 --
workspace ${WORKSPACE_NAME}
dkp create appdeployment minio-operator --app minio-operator-4.4.25 --workspace
${WORKSPACE_NAME}
dkp create appdeployment grafana-loki --app grafana-loki-0.48.4 --workspace $
{WORKSPACE_NAME}
dkp create appdeployment grafana-logging --app grafana-logging-6.28.0 --
workspace ${WORKSPACE_NAME}
```

Then, you can verify the cluster logging stack installation(see page 568).

To deploy the applications to selected clusters within the workspace, refer to the cluster-scoped configuration(see page 434) section.

471 https://kubernetes.io/docs/tasks/access-application-cluster/configure-access-multiple-clusters/

⚠ We do not recommend installing Fluent Bit, which is responsible for collecting admin logs, unless you have configured the Grafana Loki Minio Tenant with sufficient storage space. Enabling admin logs via Fluent Bit may use around 2GB/day per node. See https://d2iq.atlassian.net/wiki/spaces/DENT/pages/29923823/2.3.0+Release+Notes#Configuring-the-Grafana-Loki-Minio-Tenant(see page 0) for more details on how to configure the Minio Tenant.

To install Fluent Bit, create the AppDeployment:

```
dkp create appdeployment fluent-bit --app fluent-bit-0.19.21 --workspace $
{WORKSPACE_NAME}
```

Only for Attached GKE Clusters:
Create a ResourceQuota to enable log collection

At this point, log collection is not enabled for your attached GKE cluster. Create a ResourceQuota to enable log collection(see page 0).

8.8.2.6 Override ConfigMap to Restrict Logging to Specific Namespaces

ⓘ How to override the logging configMap to restrict logging to specific namespaces.

As a cluster administrator, you may have a need to limit, or restrict, logging activities only to certain namespaces. Kommander allows you to do this by creating an override configMap that modifies the logging configuration created in the Create AppDeployment for Workspace Logging(see page 563) procedure.

Prerequisites

- Implement each of the steps listed in Enable Workspace-level Logging(see page 562).
- Ensure that log data is available before you execute this procedure.

Create and use the override entries

To create and use the override configMap entries, follow these steps:

1. Execute the following command to get the namespace of your workspace:

   ```
   dkp get workspaces
   ```

   And copy the value under the `NAMESPACE` column for your workspace.
2. Set the `WORKSPACE_NAMESPACE` variable to the namespace copied in the previous step:

   ```
   export WORKSPACE_NAMESPACE=<WORKSPACE_NAMESPACE>
   ```
3. Identify one or more namespaces to which you want to restrict logging.
4. Create a file named `logging-operator-logging-overrides.yaml` and paste the following YAML code into it to create the overrides configMap:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: logging-operator-logging-overrides
  namespace: ${WORKSPACE_NAMESPACE}
data:
  values.yaml: |
    ---
    clusterFlows:
    - name: cluster-containers
      spec:
        globalOutputRefs:
        - loki
        match:
        - exclude:
            namespaces:
            - <your-namespace>
            - <your-other-namespace>
```

5. Add the relevant namespace values for `metadata.namespace` and the `clusterFlows[0].spec.match[0].exclude.namespaces` values at the end of the file, and save the file.
6. Use the following command to apply the YAML file:

```
kubectl apply -f logging-operator-logging-overrides.yaml
```

7. Edit the `logging-operator` AppDeployment to set the value of `spec.configOverrides.name` to `logging-operator-logging-overrides`.
(Refer to Deploy an application with a custom configuration(see page 0) for more information)

```
dkp edit appdeployment -n ${WORKSPACE_NAMESPACE} logging-operator
```

After your editing is complete, the AppDeployment resembles this example:

```
apiVersion: apps.kommander.d2iq.io/v1alpha3
kind: AppDeployment
metadata:
  name: logging-operator
  namespace: ${WORKSPACE_NAMESPACE}
spec:
  appRef:
    name: logging-operator-3.17.7
    kind: ClusterApp
  configOverrides:
    name: logging-operator-logging-overrides
```

8. Perform actions that generate log data, both in the specified namespaces *and* the namespaces you mean to exclude.

9. Verify that the log data contains only the data you expected to receive.

8.8.2.7 Override ConfigMap to Modify the Storage Retention Period in Workspace Grafana Loki

DKP configures Grafana Loki to retain log metadata and logs for 1 week, using the Compactor[472]. This retention policy can be customized.

> The minimum retention period is 24h.

To customize the retention policy using `configOverrides`, run these commands on the **management** cluster:

1. Execute the following command to get the namespace of your workspace:

   ```
   dkp get workspaces
   ```

   Copy the value under the `NAMESPACE` column for your workspace.

2. Set the `WORKSPACE_NAMESPACE` variable to the namespace copied in the previous step:

   ```
   export WORKSPACE_NAMESPACE=<WORKSPACE_NAMESPACE>
   ```

3. Create a `ConfigMap` with custom configuration values for Grafana Loki. Since the retention configuration is nested in a `config` string, you must copy the entire block. The following example sets the retention period to 360 hours (15 days). Refer to Grafana Loki's Retention Configuration documentation[473] for more information about this field.

   ```
   cat <<EOF | kubectl apply -f -
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: grafana-loki-custom-overrides
     namespace: ${WORKSPACE_NAMESPACE}
   data:
     values.yaml: |
       loki:
         structuredConfig:
           limits_config:
             retention_period: 360h
   EOF
   ```

4. Edit the `grafana-loki` AppDeployment to set the value of `spec.configOverrides.name` to `grafana-loki-custom-overrides` (Refer to Deploy a service with a custom configuration(see page 0) for more information).

472 https://grafana.com/docs/loki/latest/operations/storage/boltdb-shipper/#compactor
473 https://grafana.com/docs/loki/latest/operations/storage/retention/#retention-configuration

```
dkp edit appdeployment -n ${WORKSPACE_NAMESPACE} grafana-loki
```

After your editing is complete, the AppDeployment resembles this example:

```
apiVersion: apps.kommander.d2iq.io/v1alpha3
kind: AppDeployment
metadata:
  name: grafana-loki
  namespace: ${WORKSPACE_NAMESPACE}
spec:
  appRef:
    name: grafana-loki-0.48.4
    kind: ClusterApp
  configOverrides:
    name: grafana-loki-custom-overrides
```

8.8.2.8 Verify Cluster Logging Stack Installation

How to verify the cluster's logging stack installed successfully.

You must wait for the cluster's logging stack `HelmReleases` to deploy before attempting to configure or use the logging features.

Run the following commands on the **management** cluster:

1. Execute the following command to get the namespace of your workspace:

   ```
   dkp get workspaces
   ```

   And copy the value under the `NAMESPACE` column for your workspace.
2. Set the `WORKSPACE_NAMESPACE` variable to the namespace copied in the previous step:

   ```
   export WORKSPACE_NAMESPACE=<WORKSPACE_NAMESPACE>
   ```

   Ensure you switched to the correct context or kubeconfig[474] of the attached cluster for the following kubectl commands.
3. Check the deployment status using this command on the **attached** cluster:

   ```
   kubectl get helmreleases -n ${WORKSPACE_NAMESPACE}
   ```

NOTE: It may take some time for these changes to take effect, based on the duration configured for the Flux GitRepository reconciliation.

When the logging stack is successfully deployed, you will see output that includes the following `HelmReleases`:

474 https://kubernetes.io/docs/tasks/access-application-cluster/configure-access-multiple-clusters/

```
NAME                          READY   STATUS                             AGE
grafana-logging               True    Release reconciliation succeeded   15m
logging-operator              True    Release reconciliation succeeded   15m
logging-operator-logging      True    Release reconciliation succeeded   15m
minio-operator                True    Release reconciliation succeeded   15m
grafana-loki                  True    Release reconciliation succeeded   15m
```

Then, you can View Cluster Log Data(see page 569).

8.8.2.9 View Cluster Log Data

How to view the cluster's log data after enabling logging.

Though you enable logging at the Workspace level, viewing the log data is done at the cluster level, using the cluster's Grafana logging URL.

Run the following commands on the **management** cluster:

1. Execute the following command to get the namespace of your workspace:

   ```
   dkp get workspaces
   ```

   And copy the value under the `NAMESPACE` column for your workspace.

2. Set the `WORKSPACE_NAMESPACE` variable to the namespace copied in the previous step:

   ```
   export WORKSPACE_NAMESPACE=<WORKSPACE_NAMESPACE>
   ```

   Run the following commands on the **attached** cluster to access the Grafana UI:
   Ensure you switched to the correct context or kubeconfig[475] of the attached cluster for the following kubectl commands:

3. Get the Grafana URL:

   ```
   kubectl get ingress -n ${WORKSPACE_NAMESPACE} grafana-logging -o go-
   template='https://{{with index .status.loadBalancer.ingress 0}}
   {{or .hostname .ip}}{{end}}{{with index .spec.rules 0}}{{with index .http.paths
   0}}{{.path }}{{end}}{{end}}{{"\n"}}'
   ```

To view logs in Grafana:

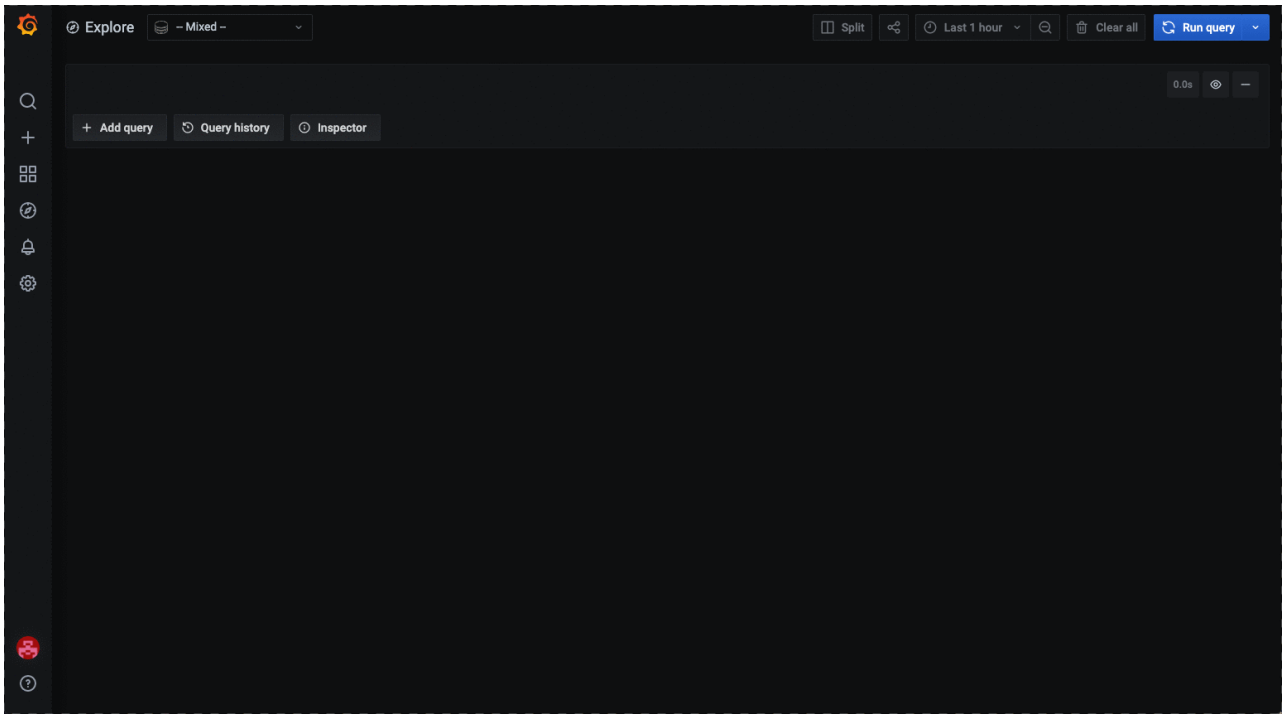
1. Go to the Explore tab:

   ```
   kubectl get ingress -n ${WORKSPACE_NAMESPACE} grafana-logging -o go-
   template='https://{{with index .status.loadBalancer.ingress 0}}
   {{or .hostname .ip}}{{end}}{{with index .spec.rules 0}}{{with index .http.paths
   0}}{{.path }}{{end}}{{end}}/explore{{"\n"}}'
   ```

475 https://kubernetes.io/docs/tasks/access-application-cluster/configure-access-multiple-clusters/

2. You may be prompted to log in using the SSO flow. See Authentication and Authorization(see page 584) for more information.
3. At the top of the page, change the datasource to `Loki`.

See the Grafana Loki documentation[476] for more on how to use the interface to view and query logs.



⚠ Cert-Manager and Traefik must be deployed in the attached cluster to be able to access the Grafana UI. These are deployed by default on the workspace.

8.8.3 Multi-Tenant Logging Overview

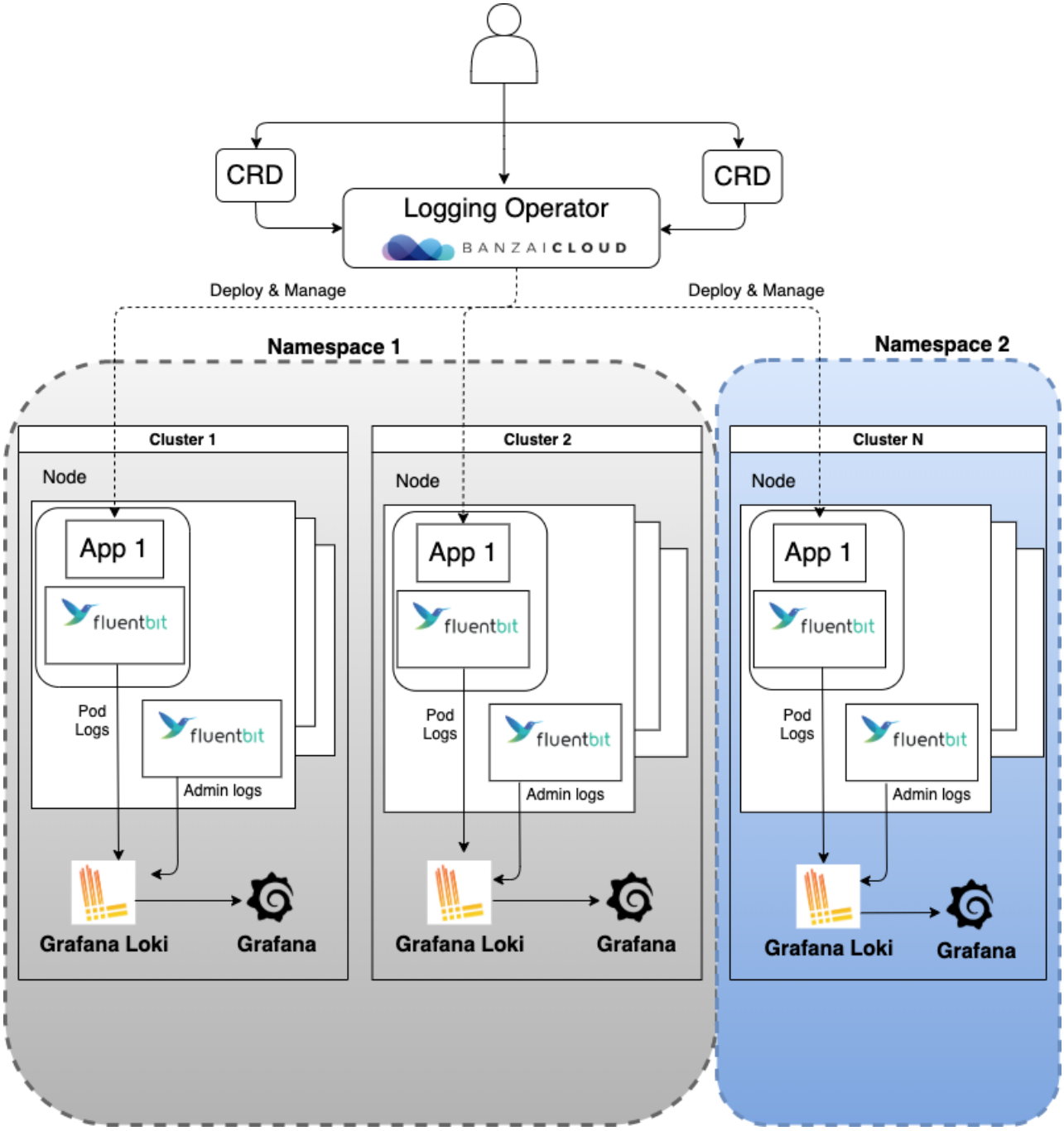
Multi-tenant logging in Kommander is built on the DKP Logging architecture. Multi-tenant logging uses the same components as the base logging architecture:

- BanzaiCloud logging-operator
- Grafana Loki
- Grafana

⚠ You must perform the Workspace-level Logging(see page 562) procedures as a prerequisite to enable multi-tenant logging at the Project level as well.

Access to log data is done at the namespace level, through the use of Projects within Kommander as shown in the diagram:

476 https://grafana.com/docs/grafana/v7.5/datasources/loki/



Each Project namespace has a logging-operator “Flow” that sends its pod logs to its own Loki server. A custom controller deploys corresponding Loki and Grafana servers in each namespace, and defines a logging-operator Flow in each namespace that forwards its pod logs to its respective Loki server. There is a corresponding Grafana server for visualizations for each namespace.

For the convenience of cluster Administrators, a cluster-scoped Loki/Grafana instance pair is deployed with a corresponding Logging-operator `ClusterFlow` that directs pod logs from all namespaces to the pair. A cluster Administrator can grant access either to none of the logs, or to all logs collected from all pods in a given

namespace. Assigning teams to specific namespaces enables the team members to see only the logs for the namespaces they own.

As with any endpoint, if an Ingress controller is in use in the environment, take care that the ingress rules do not supersede the RBAC permissions and thus prevent access to the logs.

Cluster Administrators will need to monitor and adjust resource usage to prevent operational difficulties or excessive use on a *per namespace* basis.

8.8.3.1 Enable Multi-tenant Logging

ENTERPRISE

Prerequisites

Before you begin, you must:

- Enable Workspace-level Logging(see page 562) before you can configure multi-tenant logging.
- Be a cluster administrator with permissions to configure cluster level platform services.

Multi-tenant logging enablement process

The steps required to enable multi-tenant logging include:

1. Create a Project.(see page 572)
2. Create the required Project-level AppDeployments.(see page 572)
3. Verify that the Project logging stack is installed.(see page 575)
4. View a Project's log data.(see page 576)

Get started with multi-tenant logging by Creating a Project(see page 572).

8.8.3.2 Create a Project for Logging

ENTERPRISE

To enable multi-tenant logging, you must first Create a Project(see page 459) and its namespace. Users assigned to this namespace will be able to access log data for only that namespace and not others.

Then, you can create project-level AppDeployments for use in multi-tenant logging(see page 572).

8.8.3.3 Create Project-level logging AppDeployments

ENTERPRISE

How to create Project-level AppDeployments for use in multi-tenant logging

You must create AppDeployments in the Project namespace to enable and deploy the logging stack to all clusters within a Project. You can use the CLI to do this, or use the DKP UI to enable the logging applications.

To create the AppDeployments needed for Project-level logging, follow these steps **on the management cluster**:

1. Determine the name and namespace of the workspace that your project is in. You can use the `dkp get workspaces` command to see the list of workspace names and their corresponding namespaces.

   ```
   dkp get workspaces
   ```

   Copy the values under the `NAME` and `NAMESPACE` columns for your workspace.
2. Export the `WORKSPACE_NAME` variable:

   ```
   export WORKSPACE_NAME=<WORKSPACE_NAME>
   ```
3. Export the `WORKSPACE_NAMESPACE` variable:

   ```
   export WORKSPACE_NAMESPACE=<WORKSPACE_NAMESPACE>
   ```
4. Execute the following command to get the namespace of your project:

   ```
   kubectl get projects -n ${WORKSPACE_NAMESPACE}
   ```

   Copy the value under the `NAME` column for your project. This may NOT be identical to the Display Name of the `Project`.
5. Export the `PROJECT_NAME` variable:

   ```
   export PROJECT_NAME=<PROJECT_NAME>
   ```
6. Copy these commands and execute them from a command line:

   ```
   dkp create appdeployment project-grafana-loki --app project-grafana-loki-0.48.3
    --workspace ${WORKSPACE_NAME} --project ${PROJECT_NAME}
   dkp create appdeployment project-grafana-logging --app project-grafana-
   logging-6.28.0 --workspace ${WORKSPACE_NAME} --project ${PROJECT_NAME}
   ```

Then, you can Verify the Project Logging Stack Installation (see page 575)for multi-tenant logging.

8.8.3.4 Override ConfigMap to Modify the Storage Retention Period in Project Grafana Loki

ENTERPRISE

DKP configures Project Grafana Loki to retain log metadata and logs for 1 week, using the Compactor[477]. This retention policy can be customized.

The minimum retention period is 24h.

477 https://grafana.com/docs/loki/latest/operations/storage/boltdb-shipper/#compactor

To customize the retention policy using `configOverrides`, run these commands on the **management** cluster:

1. Determine the namespace of the workspace that your project is in. You can use the `dkp get workspaces` command to see the list of workspace names and their corresponding namespaces.

   ```
   dkp get workspaces
   ```

   Copy the value under the `NAMESPACE` column for your workspace. This may NOT be identical to the Display Name of the `Workspace`.
2. Set the `WORKSPACE_NAMESPACE` variable to the namespace copied in the previous step:

   ```
   export WORKSPACE_NAMESPACE=<WORKSPACE_NAMESPACE>
   ```
3. Get the namespace of your project:

   ```
   kubectl get projects --namespace ${WORKSPACE_NAMESPACE}
   ```

   Copy the value under the `PROJECT NAMESPACE` column for your workspace. This may NOT be identical to the Display Name of the `Project`.
4. Set the `PROJECT_NAMESPACE` variable to the namespace copied in the previous step:

   ```
   export PROJECT_NAMESPACE=<PROJECT_NAMESPACE>
   ```
5. Create a `ConfigMap` with custom configuration values for Grafana Loki. Since the retention configuration is nested in a `config` string, you must copy the entire block. The following example sets the retention period under to 360 hours (15 days). Refer to Grafana Loki's Retention Configuration documentation[478] for more information about this field.

   ```
   cat <<EOF | kubectl apply -f -
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: project-grafana-loki-custom-overrides
     namespace: ${PROJECT_NAMESPACE}
   data:
     values.yaml: |
       loki:
         structuredConfig:
           limits_config:
             retention_period: 360h
   EOF
   ```

478 https://grafana.com/docs/loki/latest/operations/storage/retention/#retention-configuration

6. Run the following command on the **management** cluster to reference the `configOverrides` in the `project-grafana-loki` AppDeployment:

```
cat <<EOF | kubectl apply -f -
apiVersion: apps.kommander.d2iq.io/v1alpha3
kind: AppDeployment
metadata:
  name: project-grafana-loki
  namespace: ${PROJECT_NAMESPACE}
spec:
  appRef:
    name: project-grafana-loki-0.48.4
    kind: ClusterApp
  configOverrides:
    name: project-grafana-loki-custom-overrides
EOF
```

8.8.3.5 Verify Project Logging Stack Installation

ENTERPRISE

How to verify the project logging stack installation for multi-tenant logging

You must wait for the Project's logging stack `HelmReleases` to deploy before configuring or using the Project-level logging features, including multi-tenancy:

Run the following commands on the **management** cluster:

1. Determine the namespace of the workspace that your project is in. You can use the `dkp get workspaces` command to see the list of workspace names and their corresponding namespaces.

```
dkp get workspaces
```

   Copy the value under the `NAMESPACE` column for your workspace.
2. Export the `WORKSPACE_NAMESPACE` variable:

```
export WORKSPACE_NAMESPACE=<WORKSPACE_NAMESPACE>
```

3. Execute the following command to get the namespace of your project

```
kubectl get projects -n ${WORKSPACE_NAMESPACE}
```

   Copy the value under `PROJECT NAMESPACE` column for your project. This may NOT be identical to the Display Name of the `Project`.
4. Export the `PROJECT_NAMESPACE` variable:

```
export PROJECT_NAMESPACE=<PROJECT_NAMESPACE>
```

Run the following commands on the **managed** or **attached** cluster. For this, ensure you switch to the correct context or kubeconfig[479] of the cluster for the following `kubectl` commands:

5. Check the deployment status using this command on the attached cluster:

```
kubectl get helmreleases -n ${PROJECT_NAMESPACE}
```

NOTE: It may take some time for these changes to take effect, based on the duration configured for the Flux GitRepository reconciliation.

When successfully deployed, you will see output that includes the following `HelmReleases`:

```
NAMESPACE              NAME                        READY   STATUS
AGE
${PROJECT_NAMESPACE}   project-grafana-logging     True    Release
reconciliation succeeded   15m
${PROJECT_NAMESPACE}   project-grafana-loki        True    Release
reconciliation succeeded   11m
```

Then, you can View Project Log Data(see page 576) within multi-tenant logging.

8.8.3.6 View Project Log Data

ENTERPRISE

How to view project log data within multi-tenant logging

You can only view the log data for a Project to which you have been granted access.

Access Project Grafana's UI

Run the following commands on the **management** cluster:

1. Determine the namespace of the workspace that your project is in. You can use the `dkp get workspaces` command to see the list of workspace names and their corresponding namespaces.

```
dkp get workspaces
```

Copy the value under the `NAMESPACE` column for your workspace.

2. Export the `WORKSPACE_NAMESPACE` variable:

```
export WORKSPACE_NAMESPACE=<WORKSPACE_NAMESPACE>
```

3. Execute the following command to get the namespace of your project

479 https://kubernetes.io/docs/tasks/access-application-cluster/configure-access-multiple-clusters/

```
kubectl get projects -n ${WORKSPACE_NAMESPACE}
```

Copy the value under `PROJECT NAMESPACE` column for your project. This may NOT be identical to the Display Name of the `Project`.

4. Export the `PROJECT_NAMESPACE` variable:

```
export PROJECT_NAMESPACE=<PROJECT_NAMESPACE>
```

Run the following commands **on the attached cluster** to access the Project Grafana's UI:
Ensure you switched to the correct context or kubeconfig[480] of the attached cluster for the following kubectl commands:

5. Get the Grafana URL:

```
kubectl get ingress -n ${PROJECT_NAMESPACE} ${PROJECT_NAMESPACE}-project-
grafana-logging -o go-template='https://{{with
index .status.loadBalancer.ingress 0}}{{or .hostname .ip}}{{end}}{{with
index .spec.rules 0}}{{with index .http.paths 0}}{{.path }}{{end}}{{end}}
{{"\n"}}'
```

View Logs in Grafana

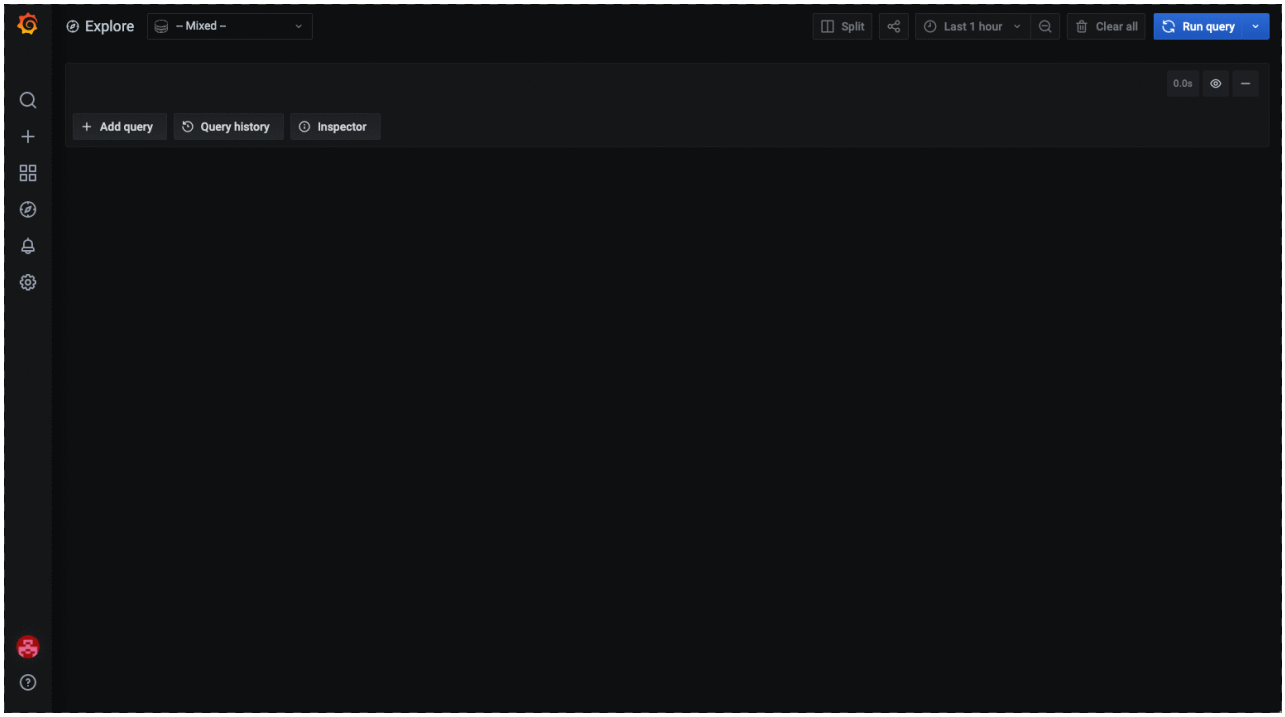
1. Go to the `Explore` tab:

```
kubectl get ingress -n ${PROJECT_NAMESPACE} ${PROJECT_NAMESPACE}-project-
grafana-logging -o go-template='https://{{with
index .status.loadBalancer.ingress 0}}{{or .hostname .ip}}{{end}}{{with
index .spec.rules 0}}{{with index .http.paths 0}}{{.path }}{{end}}{{end}}/
explore{{"\n"}}'
```

2. You may be prompted to log in using the SSO flow. See Authentication and Authorization(see page 584) for more information.
3. At the top of the page, change the data source to `Loki`.

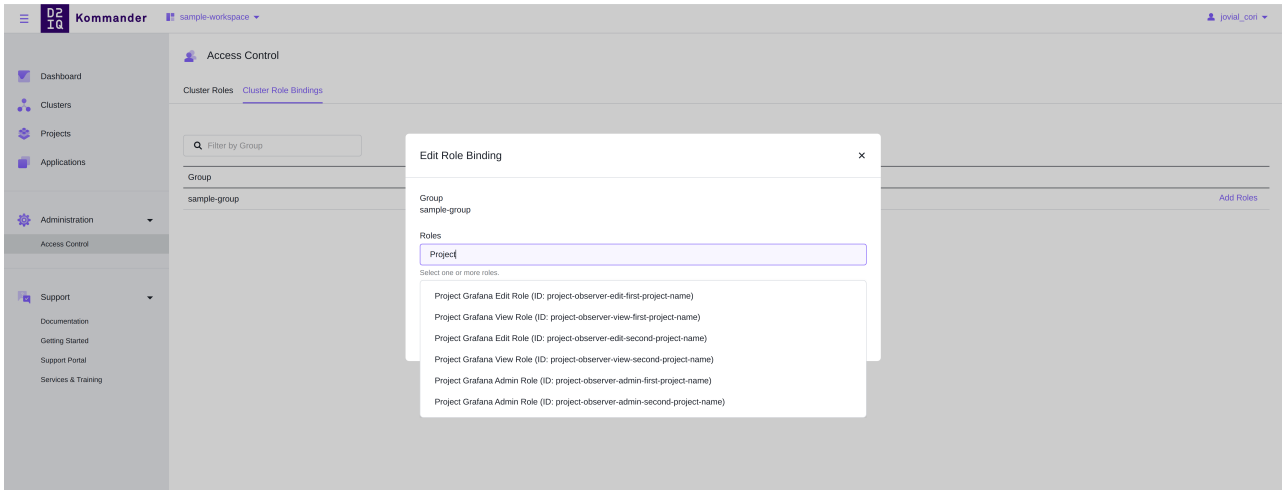
See the Grafana Loki documentation[481] for more on how to use the interface to view and query logs.

480 https://kubernetes.io/docs/tasks/access-application-cluster/configure-access-multiple-clusters/
481 https://grafana.com/docs/grafana/v7.5/datasources/loki/



> ⚠ Cert-Manager and Traefik must be deployed in the attached cluster to be able to access the Grafana UI. These are deployed by default on the workspace.

You can configure workspace policy(see page 560) to restrict access to the Project logging Grafana UI. Each Grafana instance in a Project has a unique URL at the cluster level. Consider creating a `WorkspaceRoleBinding` that maps to a `ClusterRoleBinding`, on attached cluster(s), for each Project level Grafana instance. For example, If you have a group named `sample-group` and two projects named `first-project` and `second-project` in `sample-workspace` workspace, then the Role Bindings look similar to the following:



Select the correct role bindings for each group for a project at the workspace level.

8.8.4 Fluent Bit

Fluent Bit[482] is the DKP choice of open-source log collection and forwarding tool.

⚠ On the Management cluster, Fluentbit is disabled by default. The amount of admin logs ingested to Loki requires additional disk space to be configured on the `grafana-loki-minio` Minio Tenant. Enabling admin logs may use around 2GB/day per node. See https://d2iq.atlassian.net/wiki/spaces/DENT/pages/29923823/2.3.0+Release+Notes#Configuring-the-Grafana-Loki-Minio-Tenant(see page 0) for more details on how to configure the Minio Tenant.

8.8.4.1 Audit Log Collection

Auditing[483] in Kubernetes provides a way to chronologically document the actions taken on a cluster. On Kommander, by default, audit logs are collected and stored for quick indexing. Viewing and accessing can be done via the Grafana logging UI.

To adjust the default Audit Policy log backend[484] configuration, you must modify the log retention settings by Configuring the Control Plane(see page 343) before creating the cluster. This needs to be done prior to creating the cluster since it cannot be edited after creation.

8.8.4.2 Collecting systemd logs from a non-default path

By default, Fluent Bit pods are configured to collect `systemd` logs from the `/var/log/journal/` path on cluster nodes.

If `systemd-journald` running as a part of the OS on the nodes uses a different path for writing logs, you will need to override configuration of the `fluent-bit` AppDeployment to make Fluent Bit collect `systemd` logs.

To configure the Fluent Bit AppDeployment to collect `systemd` logs from a non-default path, follow these steps (all `kubectl` and `dkp` invocations refer to the **management** cluster):

1. Execute the following command to get the namespace of the workspace in which you would like to configure Fluent Bit:

   ```
   dkp get workspaces
   ```

   And copy the value under the `NAMESPACE` column for your workspace.
2. Set the `WORKSPACE_NAMESPACE` variable to the namespace copied in the previous step:

   ```
   export WORKSPACE_NAMESPACE=<WORKSPACE_NAMESPACE>
   ```

482 https://fluentbit.io/
483 https://kubernetes.io/docs/tasks/debug-application-cluster/audit/
484 https://kubernetes.io/docs/tasks/debug/debug-cluster/audit/#log-backend

3. Identify the `systemd-journald` log data storage path on the nodes of the clusters in the workspace by using the OS documentation and examining the `systemd` configuration.
   Usually it will be either `/var/log/journal` (typically used when `systemd-journald` is configured to store logs permanently; in this case the default Fluent Bit configuration should work) or `/run/log/journal` (typically used when `systemd-journald` is configured to use a volatile storage).
4. Extract the default Helm values used by the Fluent Bit App:

```
kubectl get -n ${WORKSPACE_NAMESPACE} configmaps fluent-bit-0.19.21-d2iq-
defaults -o=jsonpath='{.data.values\.yaml}' > fluent-bit-values.yaml
```

5. Edit the resulting file `fluent-bit-values.yaml` by removing all sections except for `extraVolumes`, `extraVolumeMounts` and `config.inputs`. The result should look similarly to this:

```
extraVolumes:
# we create this to have a persistent tail-db directory an all nodes
# otherwise a restarted fluent-bit would rescrape all tails
- name: tail-db
  hostPath:
    path: /var/log/tail-db
    type: DirectoryOrCreate
# we create this to get rid of error messages that would appear on non control-
plane nodes
- name: kubernetes-audit
  hostPath:
    path: /var/log/kubernetes/audit
    type: DirectoryOrCreate
# needed for kmsg input plugin
- name: uptime
  hostPath:
    path: /proc/uptime
    type: File
- name: kmsg
  hostPath:
    path: /dev/kmsg
    type: CharDevice

extraVolumeMounts:
- name: tail-db
  mountPath: /tail-db
- name: kubernetes-audit
  mountPath: /var/log/kubernetes/audit
- name: uptime
  mountPath: /proc/uptime
- name: kmsg
  mountPath: /dev/kmsg

config:
```

```
  inputs: |
    # Collect audit logs, systemd logs, and kernel logs.
    # Pod logs are collected by the fluent-bit deployment managed by logging-
operator.
    [INPUT]
        Name tail
        Alias kubernetes_audit
        Path /var/log/kubernetes/audit/*.log
        Parser kubernetes-audit
        DB /tail-db/audit.db
        Tag audit.*
        Refresh_Interval 10
        Rotate_Wait 5
        Mem_Buf_Limit 135MB
        Buffer_Chunk_Size 5MB
        Buffer_Max_Size 20MB
        Skip_Long_Lines Off
    [INPUT]
        Name systemd
        Alias kubernetes_host
        DB /tail-db/journal.db
        Tag host.*
        Max_Entries 1000
        Read_From_Tail On
        Strip_Underscores On
    [INPUT]
        Name kmsg
        Alias kubernetes_host_kernel
        Tag kernel
```

6. Add the following item to the list under the `extraVolumes` key:

```
- name: kubernetes-host
  hostPath:
    path: <path to systemd logs on the node>
    type: Directory
```

7. Add the following item to the list under the `extraVolumeMounts` key:

```
- name: kubernetes-host
  mountPath: <path to systemd logs on the node>
```

   These items will make Kubernetes mount systemd logs into Fluent Bit pods.

8. Add the following line into the `[INPUT]` entry identified by `Name systemd` and `Alias kubernetes_host`.

```
Path <path to systemd logs on the node>
```

   This is needed to make Fluent Bit actually collect the mounted logs

9. Assuming that the path to systemd logs on the node is `/run/log/journal`, the result will look similarly to this:

```
extraVolumes:
# we create this to have a persistent tail-db directory an all nodes
# otherwise a restarted fluent-bit would rescrape all tails
- name: tail-db
  hostPath:
    path: /var/log/tail-db
    type: DirectoryOrCreate
# we create this to get rid of error messages that would appear on non control-
plane nodes
- name: kubernetes-audit
  hostPath:
    path: /var/log/kubernetes/audit
    type: DirectoryOrCreate
# needed for kmsg input plugin
- name: uptime
  hostPath:
    path: /proc/uptime
    type: File
- name: kmsg
  hostPath:
    path: /dev/kmsg
    type: CharDevice
- name: kubernetes-host
  hostPath:
    path: /run/log/journal
    type: Directory

extraVolumeMounts:
- name: tail-db
  mountPath: /tail-db
- name: kubernetes-audit
  mountPath: /var/log/kubernetes/audit
- name: uptime
  mountPath: /proc/uptime
- name: kmsg
  mountPath: /dev/kmsg
- name: kubernetes-host
  mountPath: /run/log/journal

config:
  inputs: |
    # Collect audit logs, systemd logs, and kernel logs.
    # Pod logs are collected by the fluent-bit deployment managed by logging-
operator.
    [INPUT]
        Name tail
        Alias kubernetes_audit
        Path /var/log/kubernetes/audit/*.log
```

```
        Parser kubernetes-audit
        DB /tail-db/audit.db
        Tag audit.*
        Refresh_Interval 10
        Rotate_Wait 5
        Mem_Buf_Limit 135MB
        Buffer_Chunk_Size 5MB
        Buffer_Max_Size 20MB
        Skip_Long_Lines Off
    [INPUT]
        Name systemd
        Alias kubernetes_host
        Path /run/log/journal
        DB /tail-db/journal.db
        Tag host.*
        Max_Entries 1000
        Read_From_Tail On
        Strip_Underscores On
    [INPUT]
        Name kmsg
        Alias kubernetes_host_kernel
        Tag kernel
```

10. Create a `ConfigMap` manifest with override values from `fluent-bit-values.yaml`:

```
cat <<EOF >fluent-bit-overrides.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: ${WORKSPACE_NAMESPACE}
  name: fluent-bit-overrides
data:
  values.yaml: |
$(cat fluent-bit-values.yaml | sed 's/^/    /g')
EOF
```

11. Create a `ConfigMap` from the manifest above:

```
kubectl apply -f fluent-bit-overrides.yaml
```

12. Edit the `fluent-bit` AppDeployment to set the value of `spec.configOverrides.name` to the name of the created `ConfigMap`. (You can use the steps in the procedure, Deploy an Application with a Custom Configuration(see page 447) as a guide.)

```
dkp edit appdeployment -n ${WORKSPACE_NAMESPACE} fluent-bit
```

After your editing is complete, the AppDeployment resembles this example:

```
apiVersion: apps.kommander.d2iq.io/v1alpha3
kind: AppDeployment
metadata:
  name: fluent-bit
  namespace: ${WORKSPACE_NAMESPACE}
spec:
  appRef:
    name: fluent-bit-0.19.21
    kind: ClusterApp
  configOverrides:
    name: fluent-bit-overrides
```

13. Log in into the Grafana logging UI of your workspace and verify that logs with a label `log_source=kubernetes_host` are now present in Loki.

8.8.4.3 Related information

For information on related topics or procedures, refer to the following:

- Logging(see page 560)

8.9 Security

Security architecture.

- Authentication and authorization architecture(see page 584)
- OpenID Connect (OIDC) Introduction(see page 585)
- SAML Connector(see page 589)
- Policy Controls(see page 591)

8.9.1 Authentication and authorization architecture

8.9.1.1 Details on distributed authentication and authorization between clusters

8.9.1.2 Authentication

DKP UI comes with a pre-configured authentication Dex[485] identity broker and provider.

> ⚠ Kubernetes, Kommander, and Dex do not store any user identities. The Kommander installation comes with default admin static credentials. These credentials should only be used to access the **DKP UI** for configuring an external identity provider. Currently, there is no way to update these credentials, so they should be treated as backup credentials and not used for normal access.

The DKP UI admin credentials are stored as a secret. They never leave the boundary of the UI cluster and are never shared to any other cluster.

485 https://github.com/dexidp/dex

The Dex service issues an OIDC ID token[486] on successful user authentication. Other platform services use the ID token as an authentication proof. User identity to the Kubernetes API server is provided by the `kube-oidc-proxy` platform service that reads the identity from an ID token. Web requests to DKP UI access are authenticated by the traefik forward auth[487] platform service.

The kube-oidc-proxy[488] service authenticates kubectl CLI requests using the Kubernetes API Server Go library. This library requires that if an **email_verified** claim is present, it must be set to **true**, even if the `insecureSkipEmailVerified: true` flag is configured in the Dex connector. Thus, ensure that the OIDC provider is configured to set the email_verified field to 'true'.

A user identity is shared across a UI cluster and all other attached clusters.

Attached clusters

A newly attached cluster has federated `kube-oidc-proxy`, `dex-k8s-authenticator`, and `traefik-forward-auth` platform applications . These platform applications are configured to accept UI cluster Dex issued ID tokens.

When the `traefik-forward-auth` is used as a Traefik Ingress authenticator[489], it checks if the user identity was issued by the Kommander cluster Dex service. An anonymous user is redirected to the UI cluster Dex service to authenticate and confirm their identity.

Never enter your own credentials on any of the attached clusters. On the UI cluster use the static admin credentials or an external identity provider (IDP).

8.9.1.3 Authorization

There is no centralized authorization component in Kommander. Each component and service makes its own authorization decisions based on user identity.

8.9.2 OpenID Connect (OIDC) Introduction

8.9.2.1 An introduction to OpenID Connect (OIDC) Authentication in Kubernetes

All Kubernetes clusters have two categories of users: service accounts and normal users. Kubernetes manages authentication for service accounts, but the cluster administrator, or a separate service, manages authentication for normal users.

Kommander configures the cluster to use OpenID Connect (OIDC), a popular and extensible user authentication method, and installs Dex, a popular, open-source software product that integrates your existing Identity Providers with Kubernetes.

To begin, set up an Identity Provider with Dex, then use OIDC as the Authentication method.

---

486 https://openid.net/specs/openid-connect-core-1_0.html#IDToken
487 https://github.com/mesosphere/traefik-forward-auth
488 https://github.com/jetstack/kube-oidc-proxy
489 https://docs.traefik.io/v2.4/providers/kubernetes-ingress/

8.9.2.2 Identity Provider

An Identity Provider (IdP) is a service that lets you manage identity information for users, including groups. A cluster created in Kommander uses Dex as its IdP. Dex, in turn, delegates to one or more external IdPs.

If you use already use one or more of the following IdPs, you can configure Dex to use them:

Name	Supports Refresh Tokens	Supports Groups Claim	Supports `preferred_username` Claim	Status	Notes
LDAP[490]	yes	yes	yes	stable	
GitHub[491]	yes	yes	yes	stable	
SAML 2.0[492]	no	yes	no	stable	
GitLab[493]	yes	yes	yes	beta	
OpenID Connect[494]	yes	yes	yes	beta	Includes Salesforce, Azure, etc.
Google[495]	yes	yes	yes	alpha	
LinkedIn[496]	yes	no	no	beta	
Microsoft[497]	yes	yes	no	beta	
AuthProxy[498]	no	no	no	alpha	Authentication proxies such as Apache2 mod_auth, etc.

490 https://github.com/dexidp/dex/blob/v2.22.0/Documentation/connectors/ldap.md
491 https://github.com/dexidp/dex/blob/v2.22.0/Documentation/connectors/github.md
492 https://github.com/dexidp/dex/blob/v2.22.0/Documentation/connectors/saml.md
493 https://github.com/dexidp/dex/blob/v2.22.0/Documentation/connectors/gitlab.md
494 https://github.com/dexidp/dex/blob/v2.22.0/Documentation/connectors/oidc.md
495 https://github.com/dexidp/dex/blob/v2.22.0/Documentation/connectors/google.md
496 https://github.com/dexidp/dex/blob/v2.22.0/Documentation/connectors/linkedin.md
497 https://github.com/dexidp/dex/blob/v2.22.0/Documentation/connectors/microsoft.md
498 https://github.com/dexidp/dex/blob/v2.22.0/Documentation/connectors/authproxy.md

Name	Supports Refresh Tokens	Supports Groups Claim	Supports `preferred_username` Claim	Status	Notes
Bitbucket Cloud[499]	yes	yes	no	alpha	
OpenShift[500]	no	yes	no	stable	

NOTE: These are the Identity Providers supported by Dex 2.22.0[501], the version used by DKP.

8.9.2.3 Add login connectors

Kommander uses Dex to provide OpenID Connect single sign-on (SSO) to the cluster. Dex can be configured to use multiple connectors, including GitHub, LDAP, and SAML 2.0. The Dex Connector documentation[502] describes how to configure different connectors. You can add the configuration as the values field in the Dex application. An example Dex configuration provided to the Kommander CLI's `install` command would look similar to this:

```
apiVersion: config.kommander.mesosphere.io/v1alpha1
kind: Installation
apps:
  dex:
    values: |
      config:
        connectors:
        - type: oidc
          id: google
          name: Google
          config:
            issuer: https://accounts.google.com/o/oauth2/v2/auth
            clientID: YOUR_CLIENT_ID
            clientSecret: YOUR_CLIENT_SECRET
            redirectURI: https://DKP_CLUSTER_DOMAIN/dex/callback
            scopes:
            - openid
            - profile
            - email
            insecureSkipEmailVerified: true
            insecureEnableGroups: true
            userIDKey: email
            userNameKey: email
[...]
```

499 https://github.com/dexidp/dex/blob/v2.22.0/Documentation/connectors/bitbucketcloud.md
500 https://github.com/dexidp/dex/blob/v2.22.0/Documentation/connectors/openshift.md
501 https://github.com/dexidp/dex/blob/v2.22.0/README.md
502 https://dexidp.io/docs/connectors/

8.9.2.4 Change the access token lifetime

By default, the client access token lifetime is 24 hours. After this time, the token expires and cannot be used to authenticate. See the Dex documentation[503] for more information on access token expiration and rotation settings.

Here is an example configuration for extending the token lifetime to 48 hours:

```
apiVersion: config.kommander.mesosphere.io/v1alpha1
kind: Installation
apps:
  dex:
    values: |
      config:
        expiry:
          idTokens: "48h"
[...]
```

8.9.2.5 Authentication

OpenID Connect is an extension of the OAuth2 authentication protocol[504]. As required by OAuth2, the client must be registered with Dex. Do this by passing the name of the application and a callback/redirect URI. These handle the processing of the OpenID token after the user authenticates successfully. After registration, Dex returns a `client_id` and a `secret`. Authentication requests use these between the client and Dex to identify the client.

Users access Kommander in two ways:

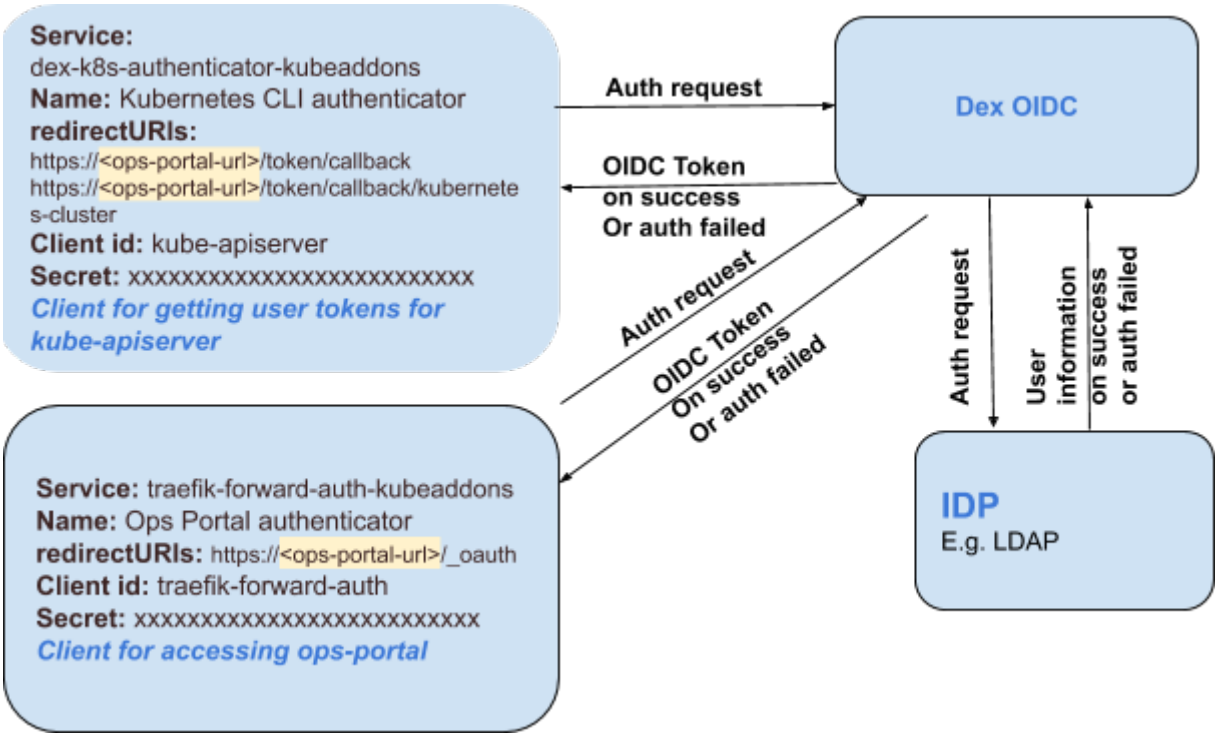
- To interact with Kubernetes API, usually through `kubectl`.
- To interact with the DKP UI, which has GUI dashboards for Prometheus, Grafana, etc.

In Kommander, Dex comes pre-configured with a client for these access use cases. The clients talk to Dex for authentication. Dex talks to the configured Identity Provider, or IdP, (for example LDAP, SAML, etc) to perform the actual task of authenticating the user.

If the user authenticates successfully, Dex pulls the user's information from the IdP and forms an OpenID token. The token contains this information and returns it to the respective client's callback URL. The client or end user uses this token for communicating with the DKP UI or Kubernetes API respectively.

This figure illustrates these components and their interaction at a high level:

503 https://dexidp.io/docs/id-tokens/#expiration-and-rotation-settings
504 https://oauth.net/2/



8.9.3 SAML Connector

8.9.3.1 Connect your Kommander cluster to an IdP using SAML

8.9.3.2 Connect Kommander to an IdP using SAML

This procedure configures your Kommander cluster to use SAML, to connect to an identity provider (IdP).

1. Install DKP(see page 97).
2. Configure the IdP
   Provide the issuer URL and the Assertion Consumer Service (ACS) or callback URL to your IdP. The issuer URL points to the authentication endpoint at the service provider (Dex), which issues a request towards the IdP via the user agent.
   The issuer URL follows this schema:

   ```
   https://<your-cluster-host>/dex
   ```

   The ACS URL points to the service provider (Dex) endpoint that receives SAML assertions issued by the IdP. The ACS or callback URL should look like this:

   ```
   https://<your-cluster-host>/dex/callback
   ```

   Depending on the IdP, you might be asked to provide the configuration in some form of an XML snippet. See the following example, making sure to replace `<your-cluster-host>` with your URL:

```
<?xml version="1.0" encoding="UTF-8"?>
<EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata" entityID="https:
//<your-cluster-host>/dex">
    <SPSSODescriptor AuthnRequestsSigned="false" WantAssertionsSigned="true"
 protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
        <NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:persistent</
NameIDFormat>
        <AssertionConsumerService index="0" isDefault="true" Binding="urn:oasis
:names:tc:SAML:2.0:bindings:HTTP-POST" Location="https://<your-cluster-host>/
dex/callback" />
    </SPSSODescriptor>
</EntityDescriptor>
```

3. Modify the `dex` configuration:

   For this step, get the following from your IdP:

   - single sign-on URL or SAML URL -> `ssoURL`
   - base64 encoded, PEM encoded CA certificate -> `caData`
   - username attribute name in SAML response -> `usernameAttr`
   - email attribute name in SAML response -> `emailAttr`

   From above you need:

   - issuer URL -> `entityIssuer`
   - callback URL -> `redirectURI`

   Ensure you base64 encode the contents of the PEM file. As an example, the prefix of the contents will result into this exact base64 prefix:

   ```
   LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tC[...]
   ```

   You can add the configuration as the values field in the `dex` application. An example `dex` configuration provided to the Kommander CLI's install command(see page 351) should look similar to:

   ```
   apiVersion: config.kommander.mesosphere.io/v1alpha1
   kind: Installation
   apps:
     dex:
       values: |
         config:
         connectors:
         - type: saml
             id: saml
             name: SAML
             config:
               ssoURL: < url for POST request >
               caData: < base64 PEM encoded CA for the IdP server >
               redirectURI: https://<your-cluster-host>/dex/callback
               entityIssuer: https://<your-cluster-host>/dex
               usernameAttr: < user attribute in saml response >
   ```

```
            emailAttr: < email attribute in saml response >
[...]
```

4. Modify the `traefik-foward-auth-mgmt` configuration and add a whitelist:
   This step is required to give access to a user to the DKP UI. For each user, you must give Access to Kubernetes resources(see page 390) and add an entry in the `whitelist` below.

```
apiVersion: config.kommander.mesosphere.io/v1alpha1
kind: Installation
apps:
...
  traefik-forward-auth-mgmt:
    values: |
      traefikForwardAuth:
        allowedUser:
          valueFrom:
            secretKeyRef: null
        whitelist:
        - < allowed email addresses >
```

5. Run `kommander install --installer-config kommander.yaml` to deploy modified `dex`.
6. Visit `https://<your-cluster-host>/dkp/kommander/dashboard` to login to the DKP UI.
7. Select `Launch Console` and follow the authentication steps to complete the procedure.

8.9.4 Policy Controls

8.9.4.1 Workload Policy Controls

8.9.4.2 Enforce Policies using Gatekeeper

Learn how to enforce policies using Gatekeeper

Gatekeeper[505] is the policy controller for Kubernetes, allowing organizations to enforce configurable policies using the Open Policy Agent[506], a policy engine for Cloud Native environments hosted by CNCF as a graduated-level project.

This tutorial describes how to use Gatekeeper to enforce policies by rejecting non-compliant resources. Specifically, this tutorial describes two constraints as a way to use Gatekeeper as an alternative to Pod Security Policies[507]:

- Prevent the running of privileged pods(see page 0)
- Prevent mounting host path volumes(see page 0)

505 https://github.com/open-policy-agent/gatekeeper
506 https://github.com/open-policy-agent/opa
507 https://kubernetes.io/docs/concepts/policy/pod-security-policy/

Before you begin

Before starting this tutorial, verify the following:

- You must have access to a Linux, macOS, or Windows computer with a supported operating system version.
- You must have a properly deployed and running cluster. For information about deploying Kubernetes with default settings on different types of infrastructures, see the Choose Infrastructure[508] topic.
- If you install Kommander with a custom configuration, make sure you enabled Gatekeeper.

Use Gatekeeper

Gatekeeper uses the OPA Constraint Framework[509] to describe and enforce policy. Before you can define a constraint, you must first define a `ConstraintTemplate`, which describes both the `Rego` (a powerful query language) that enforces the constraint and the schema of the constraint. The schema of the constraint allows an admin to fine-tune the behavior of a constraint, much like arguments to a function.

The Gatekeeper repository includes a library of policies[510] to replace Pod Security Policies which you will use in the following tutorials.

Prevent privileged pods

Define the ConstraintTemplate

Create the privileged pod policy constraint template `k8spspprivilegedcontainer` by running the following command:

```
kubectl apply -f https://raw.githubusercontent.com/open-policy-agent/gatekeeper-
library/master/library/pod-security-policy/privileged-containers/template.yaml
```

Define the Constraint

Constraints are then used to inform Gatekeeper that the admin wants to enforce a ConstraintTemplate, and how.

Create the privileged pod policy constraint `psp-privileged-container` by running the following command:

```
kubectl apply -f https://raw.githubusercontent.com/open-policy-agent/gatekeeper-
library/master/library/pod-security-policy/privileged-containers/samples/psp-
privileged-container/constraint.yaml
```

Test that the constraint is enforced

Create a privileged pod by running the following command:

```
kubectl apply -f https://raw.githubusercontent.com/open-policy-agent/gatekeeper-
library/master/library/pod-security-policy/privileged-containers/samples/psp-
privileged-container/example_disallowed.yaml
```

You should see the following output:

---

508 https://docs.d2iq.com/dkp/konvoy/2.2/choose-infrastructure/
509 https://github.com/open-policy-agent/frameworks/tree/master/constraint
510 https://github.com/open-policy-agent/gatekeeper-library/tree/master/library/pod-security-policy

```
Error from server ([denied by psp-privileged-container] Privileged container is not
allowed: nginx, securityContext: {"privileged": true}): error when creating "https://
raw.githubusercontent.com/open-policy-agent/gatekeeper-library/master/library/pod-
security-policy/privileged-containers/samples/psp-privileged-container/
example_disallowed.yaml": admission webhook "validation.gatekeeper.sh" denied the
request: [denied by psp-privileged-container] Privileged container is not allowed:
nginx, securityContext: {"privileged": true}
```

Prevent host path volumes

Define the ConstraintTemplate

Create the host path volume policy constraint template `k8spsphostfilesystem` by running the following command:

```
kubectl apply -f https://raw.githubusercontent.com/open-policy-agent/gatekeeper-
library/master/library/pod-security-policy/host-filesystem/template.yaml
```

Define the Constraint

Constraints are then used to inform Gatekeeper that the admin wants to enforce a ConstraintTemplate, and how.

Create the host path volume policy constraint `psp-host-filesystem` by running the following command to only allow `/foo` to be mounted as a host path volume:

```
cat <<EOF | kubectl apply -f -
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sPSPHostFilesystem
metadata:
  name: psp-host-filesystem
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    allowedHostPaths:
    - readOnly: true
      pathPrefix: "/foo"
EOF
```

Test that the constraint is enforced

Create a privileged pod by running the following command:

```
cat <<EOF | kubectl apply -f -
```

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-host-filesystem
  labels:
    app: nginx-host-filesystem
spec:
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
      readOnly: true
  volumes:
  - name: cache-volume
    hostPath:
      path: /tmp # directory location on host
EOF
```

You should see the following output:

```
Error from server ([denied by psp-host-filesystem] HostPath volume {"hostPath": {"pat
h": "/tmp", "type": ""}, "name": "cache-volume"} is not allowed, pod: nginx-host-
filesystem. Allowed path: [{"readOnly": true, "pathPrefix": "/foo"}]): error when
creating "STDIN": admission webhook "validation.gatekeeper.sh" denied the request:
[denied by psp-host-filesystem] HostPath volume {"hostPath": {"path": "/tmp", "type":
""}, "name": "cache-volume"} is not allowed, pod: nginx-host-filesystem. Allowed
path: [{"readOnly": true, "pathPrefix": "/foo"}]
```

Test that the constraint allows the allowed host paths
Create a pod that mounts an allowed host path by running the following command:

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: nginx-host-filesystem
  labels:
    app: nginx-host-filesystem
spec:
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
      readOnly: true
```

```
  volumes:
  - name: cache-volume
    hostPath:
      path: /foo # directory location on host
EOF
```

You should see the following output:

```
pod/nginx-host-filesystem created
```

8.10 Networking

8.10.1 Configure networking for Konvoy cluster

This section describes different networking components that come together to form a Konvoy networking stack. It assumes familiarity with Kubernetes networking.

8.10.2 Service

A Service[511] is an API resource that defines a logical set of pods and a policy by which to access them, and is an abstracted manner to expose applications as network services.

Kubernetes gives pods their own IP addresses and a single DNS name for a set of pods. Services are used as entrypoints to load-balance the traffic across the pods. A selector determines the set of Pods targeted by a Service.

For example, if you have a set of pods that each listen on TCP port `9191` and carry a label `app=MyKonvoyApp`, as configured in the following:

```
apiVersion: v1
kind: Service
metadata:
  name: my-konvoy-service
  namespace: default
spec:
  selector:
    app: MyKonvoyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9191
```

This specification creates a new `Service` object named `"my-konvoy-service"`, that targets TCP port `9191` on any pod with the `app=MyKonvoyApp` label.

511 https://kubernetes.io/docs/concepts/services-networking/service/#service-resource

Kubernetes assigns this Service an IP address. In particular, the `kube-proxy` implements a form of virtual IP for Services of type other than `ExternalName`.

- The name of a Service object must be a valid DNS label name.
- A Service is not a Platform Service.

8.10.3 Service Topology

Service Topology[512] is a mechanism in Kubernetes to route traffic based upon the Node topology of the cluster. For example, you can configure a Service to route the traffic to endpoints on specific nodes, or even based on the region or availability zone of the node's location.

To enable this new feature in your Kubernetes cluster, use the feature gates `--feature-gates="ServiceTopology=true,EndpointSlice=true"` flag. After enabling, you can control Service traffic routing by defining the `topologyKeys` field in the Service API object.

In the following example, a Service defines `topologyKeys` to be routed to endpoints only in the same zone:

```
apiVersion: v1
kind: Service
metadata:
  name: my-konvoy-service
  namespace: default
spec:
  selector:
    app: MyKonvoyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9191
  topologyKeys:
  - "topology.kubernetes.io/zone"
```

If the value of the `topologyKeys` field does not match any pattern, the traffic is rejected.

8.10.4 EndpointSlices

EndpointSlices[513] are an API resource that appears as a scalable and more manageable solution to network endpoints within a Kubernetes cluster. They allow for distributing network endpoints across multiple resources with a limit of 100 endpoints per EndpointSlice.

512 https://kubernetes.io/docs/concepts/services-networking/service-topology/#using-service-topology
513 https://kubernetes.io/docs/concepts/services-networking/endpoint-slices/#endpointslice-resource

An EndpointSlice contains references to a set of endpoints, and the control plane takes care of creating EndpointSlices for any Service that has a selector specified. These EndpointSlices include references to all the pods that match the Service selector.

Like Services, the name of a EndpointSlice object must be a valid DNS subdomain name.

In this example, here's a sample EndpointSlice resource for the example Kubernetes Service:

```
apiVersion: discovery.k8s.io/v1beta1
kind: EndpointSlice
metadata:
  name: konvoy-endpoint-slice
  namespace: default
  labels:
    kubernetes.io/service-name: my-konvoy-service
addressType: IPv4
ports:
- name: http
  protocol: TCP
  port: 80
endpoints:
- addresses:
  - "192.168.126.168"
  conditions:
    ready: true
  hostname: ip-10-0-135-39.us-west-2.compute.internal
  topology:
    kubernetes.io/hostname: ip-10-0-135-39.us-west-2.compute.internal
    topology.kubernetes.io/zone: us-west2-b
```

8.10.5 DNS for Services and Pods

Every new Service object in Kubernetes gets assigned a DNS name. The Kubernetes DNS component schedules a DNS name for the pods and services created on the cluster, and then the Kubelets are configured so containers can resolve these DNS names.

Considering previous examples, assume there is a Service named `my-konvoy-service` in the Kubernetes namespace `default`. A Pod running in namespace `default` can look up this service by performing a DNS query for `my-konvoy-service`. A Pod running in namespace `kommander` can look up this service by performing a DNS query for `my-konvoy-service.default`.

In general, a pod has the following DNS resolution:

```
pod-ip-address.namespace-name.pod-name.cluster-domain.example.
```

Similarly, a service has the following DNS resolution:

```
service-name.namespace-name.svc.cluster-domain.example.
```

You can find additional information about all the possible record types and layout here[514].

8.10.6 Ingress

Ingress[515] is an API resource that manages external access to the services in a cluster through HTTP or HTTPS. It offers name-based virtual hosting, SSL termination and load balancing when exposing HTTP/HTTPS routes from outside to services in the cluster.

The traffic policies are controlled by rules as part of the Ingress definition. Each rule defines the following details:

- An optional host to which apply the rules.
- A list of paths or routes which has an associated backend defined with a Service `name`, a port `name` and `number`.
- A backend is a combo of a Service and port names, or a custom resource backend defined as a CRD. Consequently HTTP/HTTPS requests to the Ingress that matches the host and path of the rule are sent to the listed backend.

An example of an Ingress specification is:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: konvoy-ingress
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /path
        pathType: Prefix
        backend:
          service:
            name: my-konvoy-service
            port:
              number: 80
```

In Kommander, you can expose services to the outside world using Ingress objects.

8.10.7 Ingress Controllers

In contrast with the controllers in the Kubernetes control plane, Ingress controllers are not started with a cluster so you need to choose the desired Ingress controller.

An Ingress controller has to be deployed in a cluster for the Ingress definitions to work.

Kubernetes as a project currently supports and maintains GCE and nginx controllers.

---

514 https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/
515 https://kubernetes.io/docs/concepts/services-networking/ingress/#what-is-ingress

These are four of the most known Ingress controllers[516]:

- HAProxy Ingress[517] is a highly customizable community-driven ingress controller for HAProxy.
- NGINX offers support and maintenance for the NGINX Ingress Controller[518] for Kubernetes.
- Traefik[519] is a fully featured Ingress controller (Let's Encrypt, secrets, http2, websocket), and has commercial support.
- Ambassador API Gateway[520] EXPERIMENTAL is an Envoy based Ingress controller with community and commercial support.

In Kommander, `Traefik` deploys by default as a well-suited Ingress controller.

8.10.8 Network Policies

NetworkPolicy is an API resource that controls the traffic flow at port level 3 or 4, or at the IP address level. It enables defining constraints on how a pod communicates with various network services such as `endpoints` and `services`.

A Pod can be restricted to talk to other network services through a selection of the following identifiers:

- Namespaces that have to access. There can be pods that are not allowed to talk to other namespaces.
- Other allowed IP blocks regardless of the node or IP address assigned to the targeted Pod.
- Other allowed Pods.

An example of a NetworkPolicy specification is:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: network-konvoy-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          app: MyKonvoyApp
```

516 https://kubernetes.io/docs/concepts/services-networking/ingress-controllers/#additional-controllers
517 https://haproxy-ingress.github.io/
518 https://www.nginx.com/products/nginx-ingress-controller
519 https://github.com/containous/traefik
520 https://www.getambassador.io/

```
    - podSelector:
        matchLabels:
          app: MyKonvoyApp
    ports:
    - protocol: TCP
      port: 6379
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
    ports:
    - protocol: TCP
      port: 5978
```

As shown in the example, when defining a pod or namespace based NetworkPolicy, you use a selector to specify what traffic is allowed to and from the Pod(s).

8.10.9 Adding entries to Pod /etc/hosts with HostAliases

The Pod API resource definition has a `HostAliases` field that allows adding entries to the Pod's container `/etc/hosts` file. This field overrides the hostname resolution when DNS and other options are not applicable.

For example, to resolve `foo.node.local`, `bar.node.local` to `127.0.0.1` and `foo.node.remote`, `bar.node.remote` to `10.1.2.3`, configure the `HostAliases` values as follows:

```
apiVersion: v1
kind: Pod
metadata:
  name: hostaliases-konvoy-pod
spec:
  restartPolicy: Never
  hostAliases:
  - ip: "127.0.0.1"
    hostnames:
    - "foo.node.local"
    - "bar.node.local"
  - ip: "10.1.2.3"
    hostnames:
    - "foo.node.remote"
    - "bar.node.remote"
  containers:
  - name: cat-hosts
    image: busybox
    command:
    - cat
    args:
    - "/etc/hosts"
```

8.10.10 Required Domains

8.10.10.1 This section describes the required domains for Kommander

You must have access to the following domains through the customer networking rules so that Kommander can download all required images:

- http://docker.io
- http://gcr.io
- http://k8s.gcr.io
- mcr.microsoft.com[521]
- nvcr.io
- http://quay.io
- http://us.gcr.io

In an air-gapped installation, these domains do not need to be accessible.

8.10.11 Configure Ingress for load balancing

8.10.11.1 Learn how to configure Ingress settings for load balancing (layer-7)

Ingress is the name used to describe an API object that manages external access to the services in a cluster. Typically, an Ingress exposes HTTP and HTTPS routes from outside the cluster to services running within the cluster.

The object is called an Ingress because it acts as a gateway for inbound traffic. The Ingress receives inbound requests and routes them according to the rules you defined for the **Ingress resource** as part of your cluster configuration.

This tutorial demonstrates how to expose an application running on the Konvoy cluster by configuring an Ingress for load balancing (layer-7).

8.10.11.2 Prerequisites

Before you begin, you must:

- Have access to a Linux, macOS, or Windows computer with a supported operating system version.
- Have a properly deployed and running cluster.

8.10.11.3 Expose a pod using an Ingress (L7)

1. Deploy two web application Pods on your Kubernetes cluster by running the following command:

521 http://mcr.microsoft.com

```
kubectl run --restart=Never --image hashicorp/http-echo --labels app=http-
echo-1 --port 80 http-echo-1 -- -listen=:80 --text="Hello from http-echo-1"
kubectl run --restart=Never --image hashicorp/http-echo --labels app=http-
echo-2 --port 80 http-echo-2 -- -listen=:80 --text="Hello from http-echo-2"
```

2. Expose the Pods with a service type of ClusterIP by running the following commands:

```
kubectl expose pod http-echo-1 --port 80 --target-port 80 --name "http-echo-1"
kubectl expose pod http-echo-2 --port 80 --target-port 80 --name "http-echo-2"
```

3. Create the Ingress to expose the application to the outside world by running the following command:

```
cat <<EOF | kubectl create -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: kommander-traefik
    traefik.ingress.kubernetes.io/router.tls: "true"
  generation: 7
  name: echo
  namespace: default
spec:
  rules:
  - http:
      paths:
      - backend:
          service:
            name: http-echo-1
            port:
              number: 80
        path: /echo1
        pathType: Prefix
  - http:
      paths:
      - backend:
          service:
            name: http-echo-2
            port:
              number: 80
        path: /echo2
        pathType: Prefix
EOF
```

The configuration settings in this example illustrates:

- setting the `kind` to `Ingress`.
- setting the `service.name` to be exposed as each `backend`.

4. Run the following command to get the URL of the load balancer created on AWS for the Traefik service:

```
kubectl get svc kommander-traefik -n kommander
```

This command displays the internal and external IP addresses for the exposed service. (Note that IP addresses and host names are for illustrative purposes. Always use the information from your own cluster)

```
NAME                 TYPE           CLUSTER-IP    EXTERNAL-IP
PORT(S)                                  AGE
kommander-traefik    LoadBalancer   10.0.24.215
abf2e5bda6ca811e982140acb7ee21b7-37522315.us-west-2.elb.amazonaws.com   80:3116
9/TCP,443:32297/TCP,8080:31923/TCP   4h22m
```

5. Validate that you can access the web application Pods by running the following commands: (Note that IP addresses and host names are for illustrative purposes. Always use the information from your own cluster)

```
curl -k https://abf2e5bda6ca811e982140acb7ee21b7-37522315.us-
west-2.elb.amazonaws.com/echo1
curl -k https://abf2e5bda6ca811e982140acb7ee21b7-37522315.us-
west-2.elb.amazonaws.com/echo2
```

8.10.12 Ingress

8.10.12.1 Traefik Ingress Controller

Kubernetes Ingress resources expose HTTP and HTTPS routes from outside the cluster to services within the cluster. In Kommander, the Traefik[522] Ingress controller is installed by default and provides access to the DKP UI.

An Ingress performs the following:

- Gives Services externally-reachable URLs
- Load balances traffic
- Terminates SSL/TLS sessions
- Offers name-based virtual hosting

An Ingress controller fulfills the Ingress with a load balancer.

A cluster can have multiple Ingress controllers. D2iQ recommends adding your own Ingress controllers for your applications. The Traefik Ingress controller that Kommander installs for access to the DKP UI can be replaced later if a different solution is a better fit. Using your own Ingress controller in parallel for your own business requirements ensures that you are not limited by any future changes in Kommander.

8.10.12.2 Traefik v2.4

Traefik is a modern HTTP reverse proxy and load balancer that deploys microservices with ease. Kommander currently installs Traefik v2.4 by default on every cluster. Traefik creates a service of type `LoadBalancer`. In the cloud, the cloud provider creates the appropriate load balancer. In an on-premises deployment, by default, it uses MetalLB.

522 https://landscape.cncf.io/card-mode?category=service-proxy&grouping=category&selected=traefik

Traefik listens to the Kubernetes API and automatically generates and updates the routes without any further configuration or intervention so that the Services selected by the Ingress resources are connected to the outside world. Further, Traefik supports a rich set of functionality such as Name-based routing, Path-based routing, Traffic splitting, etc.

Major features highlighted in the Traefik documentation:

- Continuously updates its configuration (No restarts!)
- Supports multiple load balancing algorithms
- Provides HTTPS to your microservices
- Circuit breakers, retry
- A clean web UI
- Websocket, HTTP/2, GRPC ready
- Provides metrics (Rest, Prometheus, Datadog, StatsD, InfluxDB)
- Keeps access logs (JSON, CLF)
- Exposes a Rest API
- Packaged as a single binary file (made with go) and available as a docker image

8.10.12.3 Related information

For information on related topics or procedures, refer to the following:

- List of Ingress controllers[523]
- Traefik v2.4 docs[524]
- Configure Ingress for load balancing(see page 601)
- Load Balancing(see page 604)

8.10.13 Load Balancing

In a Kubernetes cluster, depending on the flow of traffic direction, there are two kinds of load balancing:

- Load balancing for the traffic within a Kubernetes cluster
- Load balancing for the traffic coming from outside the cluster

8.10.13.1 Load balancing for internal traffic

Load balancing within a Kubernetes cluster is accessed through a `ClusterIP` service type. `ClusterIP` presents a single IP address to the client and load balances the traffic going to this IP to the backend servers. The actual load balancing happens using `iptables` or IPVS configuration. The `kube-proxy` Kubernetes component programs these. The `iptables` mode of operation uses DNAT[525] rules to distribute direct traffic to real servers, whereas IPVS[526] leverages in-kernel transport-layer load-balancing. Read a comparison between these two methods[527]. By default, `kube-proxy` runs in `iptables` mode. The kube-proxy configuration can be altered by updating the `kube-proxy` configmap in the `kube-system` namespace.

523 https://kubernetes.io/docs/concepts/services-networking/ingress-controllers/
524 https://doc.traefik.io/traefik/v2.4/
525 https://www.linuxtopia.org/Linux_Firewall_iptables/x4013.html
526 https://en.wikipedia.org/wiki/IP_Virtual_Server
527 https://www.projectcalico.org/comparing-kube-proxy-modes-iptables-or-ipvs/

8.10.13.2 Load balancing for external traffic

External traffic destined for the Kubernetes service requires a service of type `LoadBalancer`, through which external clients connect to your internal service. Under the hood, it uses a load balancer provided by the underlying infrastructure to direct the traffic.

In the Cloud

In cloud deployments, the load balancer is provided by the cloud provider. For example, in AWS, the service controller communicates with the AWS API to provision an ELB service which targets the cluster nodes.

On-premises

For an on-premises deployment, DKP ships with MetalLB[528], which provides load-balancing services. The environments that use MetalLB are pre-provisioned and vSpere infrastructures.

For more information on how to configure MetalB for these environments, refer to the following pages:

- Pre-provisioned Configure MetalLB(see page 246)
- Configure MetalLB for a vSphere infrastructure(see page 272)

8.10.14 External DNS

This section describes how to use `external-dns` to maintain your DNS records

You require a DNS record when setting up a custom domain for your cluster(see page 547). You can either create one manually, or set up the `external-dns` service to manage your DNS record automatically.

If you choose to use `external-dns` to maintain your DNS records, the `external-dns` will take care of pointing the DNS record to the ingress of the cluster automatically.

The following example shows how to configure `external-dns` to manage DNS records in AWS Route 53 automatically:

1. Open the `kommander.yaml` file:
    a. If you do not have a `kommander.yaml` file, initialize the configuration file(see page 351), so you can edit it in the following steps. **WARNING**: Initialize this file only ONCE, otherwise you will overwrite previous customizations.
    b. If you have initialized the configuration file already, open the `kommander.yaml` with the editor of your choice.
2. Adjust the `app` section of your `kommander.yaml` file to include these values:

```
apps:
  external-dns:
    enabled: true
    values: |
```

528 https://metallb.universe.tf/concepts/

```
      aws:
        credentials:
          secretKey: <secret-key>
          accessKey: <access-key>
        region: <provider-region>
        preferCNAME: true
      policy: upsert-only
      txtPrefix: local-
      domainFilters:
        - <example.com>
```

3. In the same `app` section, adjust the `traefik` section to include the following:

```
traefik:
    enabled: true
    values: |
      service:
        annotations:
          external-dns.alpha.kubernetes.io/hostname: <mycluster.example.com>
```

Refer to the external-dns documentation[529] for more information, as well as further instructions on how to configure `external-dns` to use other DNS providers like Google Cloud DNS, CloudFlare, or on-site providers.

8.10.15 Use Istio

EXPERIMENTAL

8.10.15.1 Learn how to integrate microservices managed by Istio into a DKP cluster

Istio helps you manage cloud-based deployments by providing an open-source service mesh to connect, secure, control, and observe microservices.

This topic describes how to expose an application running on the DKP cluster using the LoadBalancer (layer-4) service type.

Usage and installation of Istio on DKP is currently a self-service feature.

8.10.15.2 Before you begin

Before you begin, verify the following:

- You must have access to a Linux, macOS, or Windows computer with a supported operating system version.
- You must have a properly deployed and running cluster.

529 https://artifacthub.io/packages/helm/bitnami/external-dns

- Determine the name of the workspace where you wish to perform the deployment. You can use the `dkp get workspaces` command to see the list of workspace names and their corresponding namespaces.
- Set the `WORKSPACE_NAME` environment variable to the name of the workspace where the cluster is attached:

```
export WORKSPACE_NAME=<workspace_name>
```

8.10.15.3 Deploy Istio using DKP

Review the list of available applications[530] to obtain the current 'APP NAME' for the Istio application, as you will need this name later in this procedure.

Enable the deployment of Istio to your managed or attached cluster(see page 505) with an `AppDeployment` resource. Provide the application reference name, version, and the target workspace:

```
dkp create appdeployment istio --app istio-1.14.1 --workspace ${WORKSPACE_NAME}
```

- The `appRef.name` must match the app `name` from the list of available catalog applications.
- Create the resource in the workspace you just created, which instructs Kommander to deploy the `AppDeployment` to the `KommanderCluster` s in the same workspace.
- Observe that the `dkp create` command must be run with the `WORKSPACE_NAME` instead of the `WORKSPACE_NAMESPACE` flag.

8.10.15.4 Download the Istio command line utility

1. Store a local environment variable containing the current Istio version running in your cluster:

```
export KONVOY_ISTIO_VERSION="$(kubectl get clusteraddons istio -o go-
template='{{ .spec.chartReference.version }}')"
```

2. Pull a copy of the corresponding Istio command line to your system:

```
curl -L https://istio.io/downloadIstio | ISTIO_VERSION=${KONVOY_ISTIO_VERSION}
sh -
```

3. Change to the Istio directory and set your PATH environment variable by running the following commands:

530 https://docs.d2iq.com/dkp/kommander/2.2/workspaces/applications/platform-applications#workspace-platform-applications

```
cd istio*
export PATH=$PWD/bin:$PATH
```

4. You should now be able to run the following `istioctl` command and view the subsequent output:

```
istioctl version
```

```
client version: <your istio version here>
control plane version: <your istio version here>
data plane version: <your istio version here> (1 proxies)
```

8.10.15.5 Deploy a sample application on Istio

The Istio BookInfo sample application is composed of four separate microservices that demonstrate various Istio features.

1. Deploy the sample `bookinfo` application on the Kubernetes cluster by running the following commands:

   IMPORTANT: Ensure your `dkp` configuration references the cluster where you deployed Istio by setting the `KUBECONFIG` environment variable, or using the `--kubeconfig` flag, in accordance with Kubernetes conventions[531].

```
kubectl apply -f <(istioctl kube-inject -f samples/bookinfo/platform/kube/
bookinfo.yaml)
kubectl apply -f samples/bookinfo/networking/bookinfo-gateway.yaml
```

2. Get the URL of the load balancer created on AWS for this service by running the following command:

```
kubectl get svc istio-ingressgateway -n istio-system
```

   The command displays output similar to the following:

```
NAME                   TYPE           CLUSTER-IP    EXTERNAL-IP
PORT(S)
AGE
istio-ingressgateway   LoadBalancer   10.0.29.241
a682d13086ccf11e982140acb7ee21b7-2083182676.us-west-2.elb.amazonaws.com   15020:3
0380/TCP,80:31380/TCP,443:31390/TCP,31400:31400/TCP,15029:30756/TCP,15030:31420
/TCP,15031:31948/TCP,15032:32061/TCP,15443:31232/TCP   110s
```

3. Open a web browser and navigate to the external IP address for the load balancer to access the application.

531 https://kubernetes.io/docs/tasks/access-application-cluster/configure-access-multiple-clusters/

For example, the external IP address in the sample output is `a682d13086ccf11e982140acb7ee21b7-2083182676.us-west-2.elb.amazonaws.com`, enabling you to access the application using the following URL: `http://a682d13086ccf11e982140acb7ee21b7-2083182676.us-west-2.elb.amazonaws.com/productpage`

4. Follow the steps in the Istio BookInfo Application[532] documentation to understand the different Istio features.

8.11 GPUs

This section describes Nvidia GPU support on Kommander. DKP supported Nvidia driver[533] version is 470.x. GPUs, such as those made by AMD, are not currently supported. This document assumes familiarity with Kubernetes GPU support. More information about GPUs in AWS environment can be found in the Advanced AWS(see page 165) section.

8.11.1 Kommander GPU Overview

GPU support on Kommander uses the Nvidia container runtime[534]. With the Nvidia GPU turned on, Kommander configures the container runtime to run GPU containers, and installs all the necessary items to power up the Nvidia GPU devices.

The following components provide Nvidia GPU support on Kommander:

- `libnvidia-container` and `nvidia-container-runtime`: GPU Support in Kommander depends on the containerd runtime. `libnvidia-container` and `nvidia-container-runtime` fit between `containerd` and `runc`, simplifying the container runtime integration with the GPU.
- Nvidia Device Plugin[535]: Kommander makes use of Nvidia GPUs using this Kubernetes device plugin. It allows GPU enabled containers to run on Kubernetes, tracking the number of available GPUs on each node and their health.
- Nvidia Data Center GPU Manager[536]: Contains a Prometheus exporter that provides Nvidia GPU metrics.

Kommander runs these components as daemonsets, making them easier to manage and upgrade across all GPU nodes.

GPUs in an air-gapped pre-provisioned environment, and building GPU images while air-gapped is not supported currently.

For more information from Nvidia, see the Getting Started[537] page for Nvidia.

532 https://istio.io/docs/examples/bookinfo/
533 https://www.nvidia.com/Download/Find.aspx
534 https://github.com/NVIDIA/nvidia-container-runtime
535 https://github.com/NVIDIA/k8s-device-plugin
536 https://developer.nvidia.com/dcgm
537 https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/getting-started.html#chart-customization-options

8.11.2 Node Deployment

8.11.2.1 Prepare your node to use GPU platform services.

These instructions show how to prepare a node so Kommander can launch GPU services on the node.

8.11.2.2 Before you begin

This procedure requires the following items and configurations before you begin:

- Nodes must provide an Nvidia GPU.
- For AWS select a GPU instance type from the Accelerated Computing section of the AWS instance types[538].
- Nodes must run CentOS 7.

8.11.2.3 Use Konvoy 2 on AWS

To provision GPU nodes using Konvoy 2 on AWS:

1. Use `konvoy-image-builder` to create an Amazon AMI with the GPU override[539].

   ```
   konvoy-image build images/ami/centos-7.yaml --overrides overrides/nvidia.yaml
   ```

2. Begin the Konvoy Installation(see page 103) up to and including the `dkp create cluster aws` command.
3. Edit the `${CLUSTER_NAME}.yaml` file:
   - Update the `instanceType` of the worker node pool to an instance type that provides Nvidia GPUs, for example `p2.xlarge`.
   - Add an `ami.id` to reference the image generated by `konvoy-image-builder`. In this simplified example, we use AMI ID `ami-0d931a15fdf46f14f`, you should substitute the one from the konvoy-image-builder output.

   ```
   [...]
   ---
   apiVersion: infrastructure.cluster.x-k8s.io/v1alpha3
   kind: AWSMachineTemplate
   metadata:
     [...]
     name: <CLUSTER_NAME>-md-0
     [...]
   spec:
     template:
       spec:
   ```

538 https://aws.amazon.com/ec2/instance-types/
539 https://github.com/mesosphere/konvoy-image-builder/blob/main/docs/nvidia-gpu.md

```
      [...]
      instanceType: p2.xlarge
      [...]
      ami:
        id: ami-0d931a15fdf46f14f
```

4. Continue the Konvoy Installation(see page 103).

8.11.2.4 Manual Deployment

For clusters not covered in the previous procedure, run the following commands on each GPU node to configure the drivers:

CentOS 7

```
sudo yum update -y
sudo yum -y group install "Development Tools"
sudo yum -y install kernel-devel epel-release
sudo yum -y install dkms
sudo sed -i '/^GRUB_CMDLINE_LINUX=/s/"$/ module_name.blacklist=1
rd.driver.blacklist=nouveau modprobe.blacklist=nouveau"/' /etc/default/grub
sudo dracut --omit-drivers nouveau -f
sudo grub2-mkconfig -o /boot/grub2/grub.cfg
sudo reboot

lsmod | grep -i nouveau # ensure not loaded
sudo yum install -y tar bzip2 make automake gcc gcc-c++ pciutils elfutils-libelf-
devel libglvnd-devel iptables firewalld vim bind-utils wget
distribution=rhel7
ARCH=$( /bin/arch )
sudo yum-config-manager --add-repo http://developer.download.nvidia.com/compute/cuda/
repos/$distribution/${ARCH}/cuda-$distribution.repo
sudo yum install -y kernel-devel-$(uname -r) kernel-headers-$(uname -r)
sudo yum clean expire-cache
sudo yum install -y nvidia-driver-latest-dkms-3:460.73.01-1.el7.x86_64
```

8.11.2.5 Verification

Verify that the Nvidia driver is working by running:

```
nvidia-smi
```

When drivers are successfully installed the display will look like the following:

```
Fri Jun 11 09:05:31 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.73.01    Driver Version: 460.73.01    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
```

```
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000000:00:1E.0 Off |                    0 |
| N/A   35C    P0    73W / 149W |      0MiB / 11441MiB |     99%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

8.11.3 Kommander GPU configuration

8.11.3.1 Configure GPU for Kommander clusters

8.11.3.2 Prerequisites

Before you begin, you must:

- Ensure nodes provide an Nvidia GPU.
- For AWS, select a GPU instance type from the Accelerated Computing section of the AWS instance types[540].
- Run nodes on CentOS 7.
- Perform the Node Deployment(see page 610) procedure.

8.11.3.3 Enable Nvidia Platform Service on Kommander

To enable Nvidia GPU support when installing Kommander, perform the following steps:

1. Create an installation configuration file:

   ```
   dkp install kommander --init > install.yaml
   ```

2. Append the following to the apps section in the `install.yaml` file to enable Nvidia platform services.

   ```
   apps:
     nvidia:
       enabled: true
   ```

3. Install Kommander, using the configuration file you created:

540 https://aws.amazon.com/ec2/instance-types/

```
dkp install kommander --installer-config ./install.yaml
```

8.11.3.4 Disable Nvidia Platform Service on Kommander

1. Delete all GPU workloads on the GPU nodes where the Nvidia platform service needs to be upgraded.
2. Delete the existing Nvidia platform service.
3. Wait for all Nvidia-related resources in the `Terminating` state to be cleaned up. You can check pod status with:

```
kubectl get pods -A | grep nvidia
```

8.11.3.5 Upgrade Nvidia Platform Service on Kommander

Kommander can automatically upgrade the Nvidia GPU platform service. However, GPU workload must be drained before the Nvidia platform service can be upgraded.

To upgrade, follow the instructions to disable the service, and then the instructions to enable the service.

8.11.3.6 Nvidia GPU Monitoring

Kommander uses the NVIDIA Data Center GPU Manager[541] to export GPU metrics towards Prometheus. By default, Kommander has a Grafana dashboard called `NVIDIA DCGM Exporter Dashboard` to monitor GPU metrics. This GPU dashboard is shown in Kommander's Grafana UI.

8.11.3.7 Troubleshooting

1. Determine if all Nvidia pods are in `Running` state, as expected:

```
kubectl get pods -A | grep nvidia
```

2. Collect the logs for any problematic Nvidia pods, if there are any crashing, returning errors, or flapping. For example:

```
kubectl logs -n kube-system nvidia-nvidia-device-plugin-rpdwj
```

3. To recover from this problem, you must restart all Nvidia platform service pods that are running on the **SAME** host. In the example below, all Nvidia resources are restarted on the node `ip-10-0-101-65.us-west-2.compute.internal`:

```
$ kubectl get pod -A -o wide | grep nvidia
```

541 https://developer.nvidia.com/dcgm

```
kommander                              nvidia-nvidia-dcgm-exporter-s26r7
1/1     Running     0          51m     192.168.167.174   ip-10-0-101-65.us-
west-2.compute.internal    <none>           <none>
kommander                              nvidia-nvidia-dcgm-exporter-w7lf4
1/1     Running     0          51m     192.168.111.173   ip-10-0-75-212.us-
west-2.compute.internal    <none>           <none>
kube-system                            nvidia-nvidia-device-plugin-rpdwj
1/1     Running     0          51m     192.168.167.175   ip-10-0-101-65.us-
west-2.compute.internal    <none>           <none>
kube-system                            nvidia-nvidia-device-plugin-z7m2s
1/1     Running     0          51m     192.168.111.172   ip-10-0-75-212.us-
west-2.compute.internal    <none>           <none>
$ kubectl delete pod -n kommander nvidia-nvidia-dcgm-exporter-s26r7
pod "nvidia-nvidia-dcgm-exporter-s26r7" deleted
$ kubectl delete pod -n kube-system nvidia-nvidia-device-plugin-rpdwj
pod "nvidia-nvidia-device-plugin-rpdwj" deleted
```

4. To collect more debug information on the Nvidia platform service, run:

```
helm get all nvidia -n kommander
```

5. To validate metrics being produced and exported by the Nvidia DCGM exporter on a GPU node:

```
kubectl exec -n kommander nvidia-nvidia-dcgm-exporter-s26r7 --tty -- wget -nv
-O- localhost:9400/metrics
```

8.12 Monitoring and Alerts

Using DKP you can monitor the state of the cluster and the health and availability of the processes running on the cluster. By default, Kommander provides monitoring services using a pre-configured monitoring stack based on the Prometheus open-source project and its broader ecosystem.

The default DKP monitoring stack:

- Provides in-depth monitoring of Kubernetes components and platform services.
- Includes a default set of Grafana dashboards to visualize the status of the cluster and its platform services.
- Supports predefined critical error and warning alerts. These alerts notify immediately if there is a problem with cluster operations or availability.

By incorporating Prometheus, Kommander visualizes all the exposed metrics from your different nodes, Kubernetes objects, and platform service applications running in your cluster. The default monitoring stack also enables you to add metrics from any of your deployed applications, making those applications part of the overall Prometheus metrics stream.

- Recommendations(see page 615)
- Grafana Dashboards(see page 617)
- Cluster Metrics(see page 619)
- Configure alerts using AlertManager(see page 620)
- Centralized Monitoring(see page 624)
- Centralized Cost Monitoring(see page 628)
- Monitoring Applications Using Prometheus(see page 630)

- Set Storage Capacity for Prometheus(see page 632)

8.12.1 Recommendations

ENTERPRISE

Recommended settings for monitoring and collecting metrics for Kubernetes, platform services, and applications deployed on the cluster

D2iQ conducts routine performance testing of Kommander. The following table provides recommended settings, based on cluster size and increasing workloads, that maintain a healthy Prometheus monitoring deployment.

The resource settings reflect some settings but do not represent the exact structure to be used in the platform service configuration.

8.12.1.1 Prometheus

Cluster Size	Number of Pods	Number of Services	Resource settings

10	1k	250	`resources:`
			`  limits:`
			`  cpu: 500m`
			`  memory: 2192Mi`
			`  requests:`
			`  cpu: 100m`
			`  memory: 500Mi`
			`storage: 35Gi`

25	1k	250	`resources:`
			`  limits:`
			`  cpu: 2`
			`  memory: 6Gi`
			`  requests:`
			`  cpu: 1`
			`  memory: 3Gi`
			`storage: 60Gi`

50	1.5k	500	<pre>resources:
			  limits:
			    cpu: 7
			    memory:
			28Gi
			  requests:
			    cpu: 2
			    memory: 8Gi
			storage: 100Gi</pre>

100	3k	1k	<pre>resources:
			  limits:
			    cpu: 12
			    memory:
			50Gi
			  requests:
			    cpu: 10
			    memory:
			48Gi
			storage: 100Gi</pre>

200	10k	3k	<pre>resources:
			  limits:
			    cpu: 20
			    memory:
			80Gi
			  requests:
			    cpu: 15
			    memory:
			50Gi
			storage: 100Gi</pre>

300	15k	6k	<pre>resources:
			  limits:
			  cpu: 35
			  memory:
			150Gi
			  requests:
			  cpu: 25
			  memory:
			120Gi
			storage: 100Gi</pre>

8.12.2 Grafana Dashboards

With Grafana, you can query and view collected metrics in easy-to-read graphs. Kommander ships with a set of default dashboards including:

- Kubernetes Components: API Server, Nodes, Pods, Kubelet, Scheduler, StatefulSets and Persistent Volumes
- Kubernetes USE method: Cluster and Nodes
- Calico
- etcd
- Prometheus

Find the complete list of default enabled dashboards on GitHub[542].

To disable all of the default dashboards, follow these steps to define an overrides ConfigMap:

1. Create a file named `kube-prometheus-stack-overrides.yaml` and paste the following YAML code into it to create the overrides ConfigMap:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: kube-prometheus-stack-overrides
  namespace: <your-workspace-namespace>
data:
 values.yaml: |
   ---
   grafana:
     defaultDashboardsEnabled: false
```

2. Use the following command to apply the YAML file:

```
kubectl apply -f kube-prometheus-stack-overrides.yaml
```

542 https://github.com/prometheus-community/helm-charts/tree/main/charts/kube-prometheus-stack/templates/grafana/dashboards-1.14

3. Edit the `kube-prometheus-stack` AppDeployment to replace the `spec.configOverrides.name` value with `kube-prometheus-stack-overrides`. (You can use the steps in the procedure, Deploy an application with a custom configuration(see page 617) as a guide.) When your editing is complete, the AppDeployment will resemble this code sample:

```
apiVersion: apps.kommander.d2iq.io/v1alpha2
kind: AppDeployment
metadata:
  name: kube-prometheus-stack
  namespace: <your-workspace-namespace>
spec:
  appRef:
    name: kube-prometheus-stack-34.9.3
    kind: ClusterApp
  configOverrides:
    name: kube-prometheus-stack-overrides
```

To access the Grafana UI, browse to the landing page and then search for the Grafana dashboard, for example, `https://<CLUSTER_URL>/dkp/grafana`.

8.12.2.1 Add custom dashboards

In Kommander you can define your own custom dashboards. You can use a few methods to import dashboards[543] to Grafana.

One method is to use ConfigMaps to import dashboards[544]. Below are steps on how to create a ConfigMap with your dashboard definition.

For simplicity, this section assumes the desired dashboard definition is in `json` format:

```
{
    "annotations": {
        "list": []
    },
    "description": "etcd sample Grafana dashboard with Prometheus",
    "editable": true,
    "gnetId": null,
    "hideControls": false,
    "id": 6,
    "links": [],
    "refresh": false,
    ...
}
```

543 https://github.com/grafana/helm-charts/tree/main/charts/grafana#import-dashboards
544 https://github.com/grafana/helm-charts/tree/main/charts/grafana#sidecar-for-dashboards

After creating your custom dashboard json, insert it into a ConfigMap and save it as `etcd-custom-dashboard.yaml`:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: etcd-custom-dashboard
  labels:
    grafana_dashboard: "1"
data:
  etcd.json: |
    {
        "annotations": {
            "list": []
        },
        "description": "etcd sample Grafana dashboard with Prometheus",
        "editable": true,
        "gnetId": null,
        "hideControls": false,
        "id": 6,
        "links": [],
        "refresh": false,
        ...
    }
```

Apply the ConfigMap, which automatically gets imported to Grafana using the Grafana dashboard sidecar[545]:

```
kubectl apply -f etcd-custom-dashboard.yaml
```

8.12.3 Cluster Metrics

The `kube-prometheus-stack` is deployed by default on the management cluster and attached clusters. This stack deploys the following Prometheus components to expose metrics from nodes, Kubernetes units, and running apps:

- prometheus-operator: orchestrates various components in the monitoring pipeline.
- prometheus: collects metrics, saves them in a time series database, and serves queries.
- alertmanager: handles alerts sent by client applications such as the Prometheus server.
- node-exporter: deployed on each node to collect the machine hardware and OS metrics.
- kube-state-metrics: simple service that listens to the Kubernetes API server and generates metrics about the state of the objects.
- grafana: monitors and visualizes metrics.
- service monitors: collects internal Kubernetes components.

545 https://github.com/grafana/helm-charts/tree/main/charts/grafana#sidecar-for-dashboards

DKP has a listener on the `metrics.k8s.io/v1beta1/nodes` resource, which updates your backend store when that value changes. We then poll that backend store every 5 seconds, so the metrics are updated in realtime every 5-seconds without the need to refresh your view.

A detailed description of the exposed metrics can be found in the kube-state-metrics documentation on GitHub[546]. The `service-monitors` collect internal Kubernetes components but can also be extended to monitor customer apps as explained in the section Monitor Applications, on this page below.

8.12.4 Configure alerts using AlertManager

To keep your clusters and applications healthy and drive productivity forward, you need to stay informed of all events occurring in your cluster. DKP helps you to stay informed of these events by using the `alertmanager` of the `kube-prometheus-stack`.

Kommander is configured with pre-defined alerts to monitor four specific events. You receive alerts related to:

- State of your nodes
- System services managing the Kubernetes cluster
- Resource events from specific system services
- Prometheus expressions exceeding some pre-defined thresholds

Some examples of the alerts currently available are:

- CPUThrottlingHigh
- TargetDown
- KubeletNotReady
- KubeAPIDown
- CoreDNSDown
- KubeVersionMismatch

A complete list with all the pre-defined alerts can be found on GitHub[547].

8.12.4.1 Prerequisites

- Determine the name of the workspace where you wish to perform the actions. You can use the `dkp get workspaces` command to see the list of workspace names and their corresponding namespaces.
- Set the `WORKSPACE_NAMESPACE` environment variable to the name of the workspace's namespace where the cluster is attached:

```
export WORKSPACE_NAMESPACE=<workspace_namespace>
```

546 https://github.com/kubernetes/kube-state-metrics/tree/master/docs#exposed-metrics
547 https://github.com/prometheus-community/helm-charts/tree/main/charts/kube-prometheus-stack/templates/prometheus/rules-1.14

Use overrides configMaps to configure alert rules

You can enable or disable the default alert rules by providing the desired configuration in an overrides ConfigMap. For example, if you want to disable the default `node` alert rules, follow these steps to define an overrides ConfigMap:

1. Create a file named `kube-prometheus-stack-overrides.yaml` and paste the following YAML code into it to create the overrides ConfigMap:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: kube-prometheus-stack-overrides
  namespace: ${WORKSPACE_NAMESPACE}
data:
 values.yaml: |
   ---
   defaultRules:
     rules:
       node: false
```

2. Use the following command to apply the YAML file:

```
kubectl apply -f kube-prometheus-stack-overrides.yaml
```

3. Edit the `kube-prometheus-stack` AppDeployment to replace the `spec.configOverrides.name` value with `kube-prometheus-stack-overrides`. (You can use the steps in the procedure, Deploy an application with a custom configuration(see page 620) as a guide.)

```
dkp edit appdeployment -n ${WORKSPACE_NAMESPACE} kube-prometheus-stack
```

After your editing is complete, the AppDeployment resembles this example:

```
apiVersion: apps.kommander.d2iq.io/v1alpha2
kind: AppDeployment
metadata:
  name: kube-prometheus-stack
  namespace: ${WORKSPACE_NAMESPACE}
spec:
  appRef:
    name: kube-prometheus-stack-34.9.3
    kind: ClusterApp
  configOverrides:
    name: kube-prometheus-stack-overrides
```

4. To disable all rules, create an overrides ConfigMap with this YAML code:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: kube-prometheus-stack-overrides
  namespace: ${WORKSPACE_NAMESPACE}
data:
 values.yaml: |
   ---
   defaultRules:
     create: false
```

5. Alert rules for the Velero platform service are turned off by default. You can enable them with the following overrides ConfigMap. They should be enabled only if the `velero` platform service is enabled. If platform services are disabled disable the alert rules to avoid alert misfires.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: kube-prometheus-stack-overrides
  namespace: ${WORKSPACE_NAMESPACE}
data:
  values.yaml: |
    ---
    mesosphereResources:
      rules:
        velero: true
```

6. To create a custom alert rule named `my-rule-name`, create the overrides ConfigMap with this YAML code:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: kube-prometheus-stack-overrides
  namespace: ${WORKSPACE_NAMESPACE}
data:
  values.yaml: |
    ---
    additionalPrometheusRulesMap:
      my-rule-name:
        groups:
        - name: my_group
          rules:
          - record: my_record
            expr: 100 * my_record
```

After you set up your alerts, you can manage each alert using the Prometheus web console to mute or unmute firing alerts, and perform other operations. For more information about configuring `alertmanager`, see the Prometheus website[548].

To access the Prometheus Alertmanager UI, browse to the landing page and then search for the Prometheus Alertmanager dashboard, for example `https://<CLUSTER_URL>/dkp/alertmanager`.

Notify Prometheus Alerts in Slack

To hook up the Prometheus `alertmanager` notification system, you need to overwrite the existing configuration.

1. The following file, named `alertmanager.yaml`, configures `alertmanager` to use the Incoming Webhooks feature of Slack (`slack_api_url: https://hooks.slack.com/services/<HOOK_ID>`) to fire all the alerts to a specific channel `#MY-SLACK-CHANNEL-NAME`.

```
global:
  resolve_timeout: 5m
  slack_api_url: https://hooks.slack.com/services/<HOOK_ID>

route:
  group_by: ['alertname']
  group_wait: 2m
  group_interval: 5m
  repeat_interval: 1h

  # If an alert isn't caught by a route, send it to slack.
  receiver: slack_general
  routes:
    - match:
        alertname: Watchdog
      receiver: "null"

receivers:
  - name: "null"
  - name: slack_general
    slack_configs:
      - channel: '#MY-SLACK-CHANNEL-NAME'
        icon_url: https://avatars3.githubusercontent.com/u/3380462
        send_resolved: true
        color: '{{ if eq .Status "firing" }}danger{{ else }}good{{ end }}'
        title: '{{ template "slack.default.title" . }}'
        title_link: '{{ template "slack.default.titlelink" . }}'
        pretext: '{{ template "slack.default.pretext" . }}'
        text: '{{ template "slack.default.text" . }}'
        fallback: '{{ template "slack.default.fallback" . }}'
        icon_emoji: '{{ template "slack.default.iconemoji" . }}'
```

548 https://prometheus.io/docs/alerting/configuration/

```
templates:
  - '*.tmpl'
```

2. The following file, named `notification.tmpl`, is a template that defines a pretty format for the fired notifications:

```
{{ define "__titlelink" }}
{{ .ExternalURL }}/#/alerts?receiver={{ .Receiver }}
{{ end }}

{{ define "__title" }}
[{{ .Status | toUpper }}{{ if eq .Status "firing" }}:{{ .Alerts.Firing | len }}
{{ end }}] {{ .GroupLabels.SortedPairs.Values | join " " }}
{{ end }}

{{ define "__text" }}
{{ range .Alerts }}
{{ range .Labels.SortedPairs }}*{{ .Name }}*: `{{ .Value }}`
{{ end }} {{ range .Annotations.SortedPairs }}*{{ .Name }}*: {{ .Value }}
{{ end }} *source*: {{ .GeneratorURL }}
{{ end }}
{{ end }}

{{ define "slack.default.title" }}{{ template "__title" . }}{{ end }}
{{ define "slack.default.username" }}{{ template "__alertmanager" . }}{{ end }}
{{ define "slack.default.fallback" }}{{ template "slack.default.title" . }} |
{{ template "slack.default.titlelink" . }}{{ end }}
{{ define "slack.default.pretext" }}{{ end }}
{{ define "slack.default.titlelink" }}{{ template "__titlelink" . }}{{ end }}
{{ define "slack.default.iconemoji" }}{{ end }}
{{ define "slack.default.iconurl" }}{{ end }}
{{ define "slack.default.text" }}{{ template "__text" . }}{{ end }}
```

3. Finally, apply these changes to `alertmanager` as follows. Set `${WORKSPACE_NAMESPACE}` to the workspace namespace that `kube-prometheus-stack` is deployed in:

```
kubectl create secret generic -n ${WORKSPACE_NAMESPACE} \
  alertmanager-kube-prometheus-stack-alertmanager \
  --from-file=alertmanager.yaml \
  --from-file=notification.tmpl \
  --dry-run=client --save-config -o yaml | kubectl apply -f -
```

8.12.5 Centralized Monitoring

ENTERPRISE

Monitor clusters, created with Kommander, on any attached cluster

Kommander provides centralized monitoring, in a multi-cluster environment, using the monitoring stack running on any attached clusters. Centralized monitoring is provided by default in every Kommander cluster.

Attached clusters are distinguished by a monitoring ID. The monitoring ID corresponds to the kube-system namespace UID of the cluster. To find a cluster's monitoring ID, you can go to the Clusters tab on the DKP UI (in the relevant workspace), or go to the **Clusters** page in the **Global** workspace:

```
https://<CLUSTER_URL>/dkp/kommander/dashboard/clusters
```

Select the `View Details` link on the attached cluster card, and then select the **Configuration** tab, and find the monitoring ID under **Monitoring ID (clusterId)**.

You may also search or filter by monitoring IDs on the Clusters page, linked above.

You can also run this kubectl command, **using the correct cluster's context or kubeconfig**, to look up the cluster's kube-system namespace UID to determine which cluster the metrics and alerts correspond to:

```
kubectl get namespace kube-system -o jsonpath='{.metadata.uid}'
```

8.12.5.1 Centralized Metrics

The Kommander cluster collects and presents metrics from all attached clusters remotely using Thanos. You can visualize these metrics in Grafana using a set of provided dashboards.

The Thanos Query[549] component is installed on the Kommander cluster. Thanos Query queries the Prometheus instances on the attached clusters, using a Thanos sidecar running alongside each Prometheus container. Grafana is configured with Thanos Query as its datasource, and comes with pre-installed dashboards for a global view of all attached clusters. The `Thanos Query` dashboard is also installed, by default, to monitor the Thanos Query component.

NOTE: Metrics from clusters are read remotely from Kommander; they are not backed up. If a attached cluster goes down, Kommander no longer collects or presents its metrics, including past data.

You can access the centralized Grafana UI at:

```
https://<CLUSTER_URL>/dkp/kommander/monitoring/grafana
```

NOTE: This is a separate Grafana instance than the one installed on all attached clusters. It is dedicated specifically to components related to centralized monitoring.

Optionally, if you want to access the Thanos Query UI (essentially the Prometheus UI), the UI is accessible at:

```
https://<CLUSTER_URL>/dkp/kommander/monitoring/query
```

You can also check that the attached cluster's Thanos sidecars are successfully added to Thanos Query by going to:

549 https://thanos.io/v0.5/components/query/#query

```
https://<CLUSTER_URL>/dkp/kommander/monitoring/query/stores
```

The preferred method to view the metrics for a specific cluster is to go directly to that cluster's Grafana UI.

Adding custom dashboards

You can also define custom dashboards for centralized monitoring on Kommander. There are a few methods to import dashboards[550] to Grafana. For simplicity, assume the desired dashboard definition is in `json` format:

```
{
    "annotations":
    ...
    # Complete json file here
    ...
    "title": "Some Dashboard",
    "uid": "abcd1234",
    "version": 1
}
```

After creating your custom dashboard json, insert it into a ConfigMap and save it as `some-dashboard.yaml`:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: some-dashboard
  labels:
    grafana_dashboard_kommander: "1"
data:
  some_dashboard.json: |
    {
      "annotations":
      ...
      # Complete json file here
      ...
      "title": "Some Dashboard",
      "uid": "abcd1234",
      "version": 1
    }
```

Apply the ConfigMap, which will automatically get imported to Grafana via the Grafana dashboard sidecar:

```
kubectl apply -f some-dashboard.yaml
```

550 https://github.com/mesosphere/charts/tree/master/stable/grafana#import-dashboards

8.12.5.2 Centralized Alerts

A centralized view of alerts, from attached clusters, is provided using an alert dashboard called Karma[551]. Karma aggregates all alerts from the Alertmanagers running in the attached clusters, allowing you to visualize these alerts on one page. Using the Karma dashboard, you can get an overview of each alert and filter by alert type, cluster, and more.

Silencing alerts using the Karma UI is currently not supported.

You can access the Karma dashboard UI at:

```
https://<CLUSTER_URL>/dkp/kommander/monitoring/karma
```

When there are no attached clusters, the Karma UI displays an error message `Get https://placeholder.invalid/api/v2/status: dial tcp: lookup placeholder.invalid on 10.0.0.10:53: no such host`. This is expected, and the error disappears when clusters are connected.

Federating Prometheus Alerting Rules

You can define additional Prometheus alerting rules[552] on the Kommander cluster and federate them to all of the attached clusters by following these instructions. To use these instructions you must install the kubefedctl CLI[553].

1. Enable the PrometheusRule type for federation.

   ```
   kubefedctl enable PrometheusRules --kubefed-namespace kommander
   ```

2. Modify the existing alertmanager configuration.

   ```
   kubectl edit PrometheusRules/kube-prometheus-stack-alertmanager.rules -n
   kommander
   ```

3. Append a sample rule.

   ```
   - alert: MyFederatedAlert
     annotations:
       message: A custom alert that will always fire.
     expr: vector(1)
     labels:
       severity: warning
   ```

551 https://github.com/prymitive/karma
552 https://prometheus.io/docs/prometheus/latest/configuration/alerting_rules/
553 https://github.com/kubernetes-sigs/kubefed/blob/master/docs/installation.md#kubefedctl-cli

4. Federate the rules you just modified.

```
kubefedctl federate PrometheusRules kube-prometheus-stack-alertmanager.rules --
kubefed-namespace kommander -n kommander
```

5. Ensure that the clusters selection (`status.clusters`) is appropriately set for your desired federation strategy and check the propagation status[554].

```
kubectl get federatedprometheusrules kube-prometheus-stack-alertmanager.rules
-n kommander -oyaml
```

8.12.6 Centralized Cost Monitoring

ENTERPRISE

Monitoring costs of all attached clusters with Kubecost

Kubecost[555], running on Kommander, provides centralized cost monitoring for all attached clusters. This feature, installed by default in the management cluster, provides a centralized view of Kubernetes resources used on all attached clusters.

By default, up to 15 days of cost metrics are retained, with no backup to an external store.

8.12.6.1 Centralized Costs

Using Thanos, the management cluster collects cost metrics remotely from each attached cluster. Costs from the last day and the last 7 days are displayed for each cluster, workspace, and project in the respective DKP UI pages. Further cost analysis and details can be found in the centralized Kubecost UI running on Kommander, at:

```
https://<CLUSTER_URL>/dkp/kommander/kubecost/frontend/detail.html#&agg=cluster
```

For more information on cost allocation metrics and how to navigate this view in the Kubecost UI, see the Kubecost docs on Kubernetes Cost Allocation[556].

To identify the clusters in Kubecost, use the cluster's monitoring ID. The monitoring ID corresponds to the kube-system namespace UID of the cluster. To find the cluster's monitoring ID, select the **Clusters** tab on the DKP UI in the relevant workspace, or go to the **Clusters** page in the **Global** workspace:

```
https://<CLUSTER_URL>/dkp/kommander/dashboard/clusters
```

Select **View Details** on the attached cluster card. Select the **Configuration** tab, and find the monitoring ID under **Monitoring ID (clusterId)**.

554 https://github.com/kubernetes-sigs/kubefed/blob/master/docs/userguide.md#propagation-status
555 https://kubecost.com/
556 https://docs.kubecost.com/cost-allocation.html

You can also search or filter by monitoring ID on the **Clusters** page.

To look up a cluster's kube-system namespace UID directly using the CLI, run the following kubectl command, **using the cluster's context or kubeconfig**.

```
kubectl get namespace kube-system -o jsonpath='{.metadata.uid}'
```

Kubecost

Kubecost integrates directly with the Kubernetes API and cloud billing APIs to give you real-time visibility into your Kubernetes spend and cost allocation. By monitoring your Kubernetes spend across clusters, you can avoid overspend caused by uncaught bugs or oversights. With a cost monitoring solution in place you can realize the full potential and cost of these resources and avoid over-provisioning resources.

To customize pricing and out of cluster costs for AWS, you must apply these settings using the Kubecost UI running on each attached cluster. You can access the attached cluster's Kubecost Settings page at:

```
https://<MANAGED_CLUSTER_URL>/dkp/kubecost/frontend/settings.html
```

⚠ Make sure you access the cluster's Kubecost UI linked above, not the centralized Kubecost UI running on the Kommander management cluster.

AWS

For more accurate AWS Spot pricing, follow these steps[557] to configure a data feed for the AWS Spot instances.

To allocate out of cluster costs for AWS, visit this guide[558].

Grafana dashboards

A set of Grafana dashboards providing visualization of cost metrics is provided in the centralized Grafana UI:

```
https://<CLUSTER_URL>/dkp/kommander/monitoring/grafana
```

These dashboards give a global view of accumulated costs from all attached clusters. From the navigation in Grafana, you can find these dashboards by selecting those tagged with `cost`, `metrics`, and `utilization`.

8.12.6.2 Related Information

For information on related topics or procedures, refer to the following:

- Kubecost documentation[559]

557 https://docs.kubecost.com/getting-started#spot-nodes
558 https://docs.kubecost.com/aws-out-of-cluster.html
559 https://docs.kubecost.com/

8.12.7 Monitoring Applications Using Prometheus

Before attempting to monitor your own applications, you should be familiar with the Prometheus conventions for exposing metrics. In general, there are two key recommendations:

- You should expose metrics using an HTTP endpoint named `/metrics`.
- The metrics you expose must be in a format that Prometheus can consume.

By following these conventions, you ensure that your application metrics can be consumed by Prometheus itself or by any Prometheus-compatible tool that can retrieve metrics, using the Prometheus client endpoint.

The `kube-prometheus-stack` for Kubernetes provides easy monitoring definitions for Kubernetes services and deployment and management of Prometheus instances. It provides a Kubernetes resource called `ServiceMonitor`.

By default, the `kube-prometheus-stack` provides the following service monitors to collect internal Kubernetes components:

- kube-apiserver
- kube-scheduler
- kube-controller-manager
- etcd
- kube-dns/coredns
- kube-proxy

The operator is in charge of iterating over all of these `ServiceMonitor` objects and collecting the metrics from these defined components.

The following example illustrates how to retrieve application metrics. In this example, there are:

- Three instances of a simple app named `my-app`
- The sample app listens and exposes metrics on port 8080
- The app is assumed to already be running

To prepare for monitoring of the sample app, create a service that selects the pods that have `my-app` as the value defined for their app label setting.

The service object also specifies the port on which the metrics are exposed. The `ServiceMonitor` has a label selector to select services and their underlying endpoint objects. For example:

```
kind: Service
apiVersion: v1
metadata:
  name: my-app
  namespace: my-namespace
  labels:
    app: my-app
spec:
  selector:
    app: my-app
  ports:
  - name: metrics
```

```
    port: 8080
```

This service object is discovered by a `ServiceMonitor`, which defines the selector to match the labels with those defined in the service. The app label must have the value `my-app`.

In this example, in order for `kube-prometheus-stack` to discover this `ServiceMonitor`, add a specific label `prometheus.kommander.d2iq.io/select: "true"` in the `yaml`:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: my-app-service-monitor
  namespace: my-namespace
  labels:
    prometheus.kommander.d2iq.io/select: "true"
spec:
  selector:
    matchLabels:
      app: my-app
  endpoints:
  - port: metrics
```

In this example, you would modify the Prometheus settings to have the operator collect metrics from the service monitor by appending the following configuration to the overrides ConfigMap:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: kube-prometheus-stack-overrides
  namespace: ${WORKSPACE_NAMESPACE}
data:
  values.yaml: |
    ---
    prometheus:
      additionalServiceMonitors:
        - name: my-app-service-monitor
          selector:
            matchLabels:
              app: my-app
          namespaceSelector:
            matchNames:
              - my-namespace
          endpoints:
            - port: metrics
              interval: 30s
```

Official documentation about using a `ServiceMonitor` to monitor an app with the Prometheus-operator on Kubernetes can be found on this GitHub repository[560].

8.12.8 Set Storage Capacity for Prometheus

Follow the steps in this page to set a specific storage capacity for Prometheus.

When defining the requirements of a cluster, you can specify the capacity and resource requirements of Prometheus by modifying the settings in the overrides ConfigMap definition as shown below:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: kube-prometheus-stack-overrides
  namespace: ${WORKSPACE_NAMESPACE}
data:
  values.yaml: |
    ---
    prometheus:
      prometheusSpec:
        resources:
          limits:
            cpu: "4"
            memory: "8Gi"
          requests:
            cpu: "2"
            memory: "6Gi"
      storageSpec:
        volumeClaimTemplate:
          spec:
            resources:
              requests:
                storage: "100Gi"
```

8.13 DKP Troubleshooting

The following pages provide the steps to troubleshoot a Kubernetes cluster installation.

- Generate a Support Bundle(see page 632)
- Custom Collectors(see page 636)

8.13.1 Generate a Support Bundle

Follow these instructions to generate a support bundle with data collected for the last 48 hours of the life of the cluster.

560 https://github.com/coreos/prometheus-operator/blob/master/Documentation/user-guides/getting-started.md#related-resources

8.13.1.1 Prerequisites

Before generating a support bundle, verify that you have:

- An AMD64-based Linux or macOS machine with a supported version of the operating system.
- A running Kubernetes cluster.
- Access to the DKP CLI(see page 687).

8.13.1.2 Create a diagnostic bundle

`dkp diagnose` was developed by D2iQ and builds on the open source `troubleshoot.sh` project.

> The command `dkp diagnose` is based on version `0.13.16` of `troubleshoot.sh` with custom modifications. The D2iQ fork is open source and available from on this public GitHub repository[561].

`dkp diagnose` supports multiple support bundle collectors[562] and can be configured as a `SupportBundle` Kubernetes resource in a yaml file.

The following list is the minimum set of resources that is required to debug a cluster, but can be further customized.

The bundle uses the following collectors:

- clusterInfo[563] collects basic information about the cluster
- clusterResources[564] collects a subset of available resources in the cluster
- configMap[565] collects the values of Kubernetes ConfigMaps
- secrets[566] collects the values of Kubernetes ConfigMaps
- execCopyFromHost[567] runs a container on each node on the cluster and copies the created data
- allLogs[568] is capable of collecting logs from all containers on the cluster

8.13.1.3 Generate a Support Bundle

> The command `dkp diagnose` uses the same Kubernetes configuration as `kubectl`. `dkp diagnose` can also be pointed at a specific configuration by using the `--kubeconfig` parameter.

To generate the support bundle, perform the following steps:

1. Run the `dkp diagnose` command by running the default collectors configuration.

561 https://github.com/mesosphere/troubleshoot
562 https://troubleshoot.sh/docs/collect/all/
563 https://troubleshoot.sh/docs/collect/cluster-info/
564 https://troubleshoot.sh/docs/collect/cluster-resources/
565 https://troubleshoot.sh/docs/collect/configmap/
566 https://troubleshoot.sh/docs/collect/secret/
567 https://docs.d2iq.com/custom-collectors#execcopyfromhost_collector
568 https://docs.d2iq.com/custom-collectors#alllogs_collector

```
dkp diagnose
```

The output looks similar to this:

```
Collecting support bundle ...

support-bundle-2021-08-13T14_44_23.tar.gz
```

2. To view the bundle contents, extract the bundle (replacing `support-bundle-2021-08-13T14_44_23.tar.gz` with the location from the previous step):

```
tar -xzvf support-bundle-2021-08-13T14_44_23.tar.gz
```

3. A new directory named `support-bundle-<date-created>` is created. This directory contains the files specified:

```
ls support-bundle-2021-08-13T14_44_23
```

The output looks similar to this:

```
cluster-info  cluster-resources  configmaps  node-diagnostics  pod-logs
secrets  version.yaml
```

Collect information from a bootstrap cluster

In the case where your bootstrap cluster has not yet pivoted towards your Konvoy cluster, you can collect log information from that bootstrap cluster as well, and there are a preconfigured set of relevant collectors. Specify an additional bootstrap cluster kubeconfig using the `--bootstrap-kubeconfig` parameter to activate bootstrap cluster diagnostics. You will receive an additional support bundle named `bootstrap-support-bundle-<date created>`.

Note that the bootstrap cluster diagnostics are independent of the configuration of the “main” or Konvoy cluster diagnostics. We run a static collector set that collects the following bootstrap cluster information:

- ClusterInfo
- ClusterResources
- AllLogs
- ConfigMaps
- Secrets

1. Run the `dkp diagnose` command with bootstrap bundle configuration.

```
dkp diagnose bundle.yaml
```

Customizations

To print the default collectors configuration, run the following command:

```
dkp diagnose default-config > bundle.yaml
```

Edit the file to make appropriate modifications.

By default, `dkp diagnose` does not require that you supply a configuration. You can print the default bundle by running `dkp diagnose default-config`.

8.13.1.4 SSH fallback

In some cases the Kubernetes API is not available for the cluster. In those cases you can collect node level information using SSH access to the diagnosed nodes. Be aware that not all clusters have SSH access configured. If they do not then access using SSH fallback is not possible.

To get node level information from your cluster using SSH access, perform the following steps:

1. Enter the following command:

```
dkp diagnose ssh <path/to/ansible-inventory.yaml>
```

The `ansible-inventory.yaml` file specifies the nodes to access for data collection.

This collector does not use the full Ansible `inventory.yaml` format only a limited subset to describe the infrastructure.

Only the following attributes of the `ansible-inventory.yaml` are supported. All other group definitions are ignored.

- Support for `all` shared variables.
- Support for `hosts` key in `all` groups.
- Supported behavioral inventory is limited to:
  - `ansible_host`
  - `ansible_port`
  - `ansible_user`
  - `ansible_ssh_private_key_file`

  The following is an example `inventory.yaml` file:

```
all:
  vars:
    ansible_user: centos
```

```
  hosts:
    host-1:
      ansible_host: 192.168.10.1
    host-2:
      ansible_host: 192.168.10.22
      ansible_port: 2222
```

More information on these Ansible parameters can be found in the Ansible user guide[569].

All other group definitions in the `inventory.yaml` file are ignored.

Refer to the following example file:

```
all:
  vars:
    ansible_user: centos
  hosts:
    host-1:
      ansible_host: 192.168.10.1
    host-2:
      ansible_host: 192.168.10.22
      ansible_port: 2222
```

The fallback collector runs a bash script over SSH and copies the collected data. The format of the created bundle matches that of `dkp diagnose` collector generated bundles.

```
    node-diagnostics/<HOSTNAME_PORT>/data/
        - dmesg
        - ....
```

Redactors are supported and are in the same format as the main `dkp diagnose` command. Per node collection timeouts are supported using the `--timeout` parameter.

See also: dkp-cli(see page 687)

8.13.2 Custom Collectors

For creating diagnostic bundles, D2iQ is using a customized version of `troubleshoot.sh` integrated into `dkp-diagnose`.

569 https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html#connecting-to-hosts-behavioral-inventory-parameters

8.13.2.1 Customizations

To meet the specific needs of diagnosing DKP 2 clusters we have developed custom collectors and modified the behavior of upstream collectors. Go to our repository for more information on the details[570] of the changes.

`ExecCopyFromHost` collector

This is a new collector created specifically for gathering host level information from cluster nodes. The collector allows you to run a provided container image in a privileged mode, as a root user, with additional Linux capabilities and with the host filesystem mounted in the container.

You can collect host level information other than copying host level files. (This is already possible with the `CopyFromHost` collector.) Like the `CopyFromHost` collector, this collector runs as a Kubernetes `DaemonSet` executed on all nodes in the system. The data produced by the container are copied from a pre-defined directory into the diagnostics bundle under each node name. The name of the parent directory, in the diagnostics bundle, is determined by the name of the collector specified in its configuration.

The data written into the diagnostics bundle follows this format:

```
<collector-name> / <node-name> / data / (file1|file2|...)
```

The following is a sample configuration file:

```
spec:
  collectors:
    - execCopyFromHost:
        name: node-diagnostics
        image: mesosphere/dkp-diagnostics-node-collector:latest
        timeout: 30s
        command:
          - "/bin/bash"
          - "-c"
          - "/diagnostics/container.sh --hostroot /host --hostpath ${PATH} --
outputroot /output"
        workingDir: "/diagnostics"
        includeControlPlane: true
        privileged: true
        capabilities:
          - AUDIT_CONTROL
          - AUDIT_READ
          - BLOCK_SUSPEND
          - BPF
          - CHECKPOINT_RESTORE
          - DAC_READ_SEARCH
          - IPC_LOCK
          - IPC_OWNER
          - LEASE
```

570 https://github.com/mesosphere/troubleshoot/blob/v0.13-d2iq/README.d2iq.md

```
        - LINUX_IMMUTABLE
        - MAC_ADMIN
        - MAC_OVERRIDE
        - NET_ADMIN
        - NET_BROADCAST
        - PERFMON
        - SYS_ADMIN
        - SYS_BOOT
        - SYS_MODULE
        - SYS_NICE
        - SYS_PACCT
        - SYS_PTRACE
        - SYS_RAWIO
        - SYS_RESOURCE
        - SYS_TIME
        - SYS_TTY_CONFIG
        - SYSLOG
        - WAKE_ALARM
      extractArchive: true
```

The following is an example of the data produced by running this collector:

```
├── node-diagnostics
│   ├── troubleshoot-control-plane
│   │   └── data
│   │       ├── certs_expiration_kubeadm
│   │       ├── containerd_config.toml
│   │           ...
│   │       └── whoami_validate
│   └── troubleshoot-worker
│       └── data
│           ├── containerd_config.toml
│           ├── containers_crictl
│               ...
│           └── whoami_validate
```

In the event that an error occurs while collecting node diagnostics, the `node-diagnostics/<node>/pod-collector.json` file contains the serialized JSON representations of the running pod. This helps debug the reasons for the collection failure. The `node-diagnostics/<node>/pod-collector.log` file contains stdout from the collector container that runs the diagnostics script. In addition, the command may also produce certain `-error.txt` files. `file-copy-error.txt` and `pod-collector-files-copy-error.txt` are two file examples. These files contain error messages generated while trying to fetch log files from the collector.

When using this collector for node level information you must run additional docker containers and must have the following docker images:

- `mesosphere/pause-alpine:3.2`
- `mesosphere/dkp-diagnostics-node-collector:$(dkp-diagnose version)`

For more information on the configuration options see the `ExecCopyFromHost` in the `pkg/apis/troubleshoot/v1beta2/exec_copy_from_host.go` file.

`AllLogs` collector

This collector gathers pod logs from specified namespaces or from all namespaces if none are specified. You can collect logs of all the pods from all the namespaces. The pod logs are collected under the `allPodLogs` directory.

The data written into the diagnostics bundle follows this format:

```
<collector-name> / <namespace-name> / <pod-name> - (container1|container2|...)
```

The following is a sample configuration file to collect logs from all the pods from all the namespaces:

```
spec:
  collectors:
    - allLogs:
        namespaces:
          - "*"
```

The following is a sample configuration file to collect logs from all the pods from specific namespaces:

```
spec:
  collectors:
    - allLogs:
        namespaces:
          - default
          - dev
          - prod
```

The following is an example of the data produced by running this collector:

```
├── node-diagnostics
│   ├── troubleshoot-control-plane
│   │   └── data
│   │       ├── certs_expiration_kubeadm
│   │       ├── containerd_config.toml
                    ...
│   │       └── whoami_validate
│   └── troubleshoot-worker
│       └── data
│           ├── containerd_config.toml
│           ├── containers_crictl
                    ...
│           └── whoami_validate
```

In the event that an error occurs while collecting node diagnostics, the `node-diagnostics/<node>/pod-collector.json` file contains the serialized JSON representations of the running pod. This helps debug the reasons for the collection failure. The `node-diagnostics/<node>/pod-collector.log` file contains stdout from the collector container that runs the diagnostics script.

When using this collector for node level information you must run additional docker containers and must have the following docker images:

- `mesosphere/pause-alpine:3.2`
- `mesosphere/dkp-diagnostics-node-collector:$(dkp-diagnose version)`

For more information on the configuration options see the `ExecCopyFromHost` in the `pkg/apis/troubleshoot/v1beta2/exec_copy_from_host.go` file.

Support for collecting from all namespaces for `ConfigMap` and `Secret` collector

In the original collectors `namespace` there is a required parameter. This adds support for collecting from all namespaces by not setting the `namespace` (or setting it to `""`). Note: To collect all config maps / secrets an empty selector must be used (`selector: [""]`).

Support for optional support-bundle name prefix

When generating a support bundle, you need naming defaults to provide deterministic bundle identifiers. This feature is especially useful for our convenience extension of providing diagnostics for both, a bootstrap, Konvoy, or other K8s cluster. Using an empty prefix keeps the original naming convention.

`ClusterResources` collector

Another customization is added to collect custom resource definitions and all custom resources in the cluster.

8.14 Storage

An introduction to persistent storage in Kubernetes

This document describes the model used in Kubernetes for managing persistent, cluster-scoped storage for workloads requiring access to persistent data.

A workload on Kubernetes typically requires two types of storage:

- Ephemeral storage(see page 640)
- Persistent Volume(see page 641)

8.14.1 Ephemeral storage

Ephemeral storage, by its name, is ephemeral in the sense that it is cleaned up when the workload is deleted or the container crashes. For example, the following are examples of ephemeral storage provided by Kubernetes:

EmptyDir volume.	Managed by kubelet under /var/lib/kubelet.

Container logs.	Typically under /var/logs/containers.
Container image layers.	Managed by container runtime (e.g., under /var/lib/containerd).
Container writable layers.	Managed by container runtime (e.g., under /var/lib/containerd).

Ephemeral storage is automatically managed by Kubernetes, and typically does not require explicit settings. You may need to express the capacity requests for ephemeral storage so that `kubelet` can use that information to make sure it does not run out of ephemeral storage space on each node.

8.14.2 Persistent Volume

Persistent Volumes are storage resources that can be used by the cluster. Persistent Volumes are volume plug-ins that have lifecycle capabilities that are independent of any Kubernetes Pod or Deployment.

You may have stateful workloads requiring persistent storage whose lifecycle is longer than that of Pods or containers. For instance, a database server needs to recover database files after it crashes. For those cases, the workloads need to use PersistentVolumes (PV).

Persistent Volumes are resources that represent storage in the cluster that has been provisioned by an administrator or dynamically provisioned using Storage Classes. Unlike ephemeral storage, the lifecycle of a PersistentVolume is independent of that of the workload that uses it.

The Persistent Volume API objects capture the details of the implementation of the storage, be that NFS, iSCSI, or a cloud-provider-specific storage system. In order to use a Persistent Volume (PV), your application needs to invoke a Persistent Volume Claim (PVC).

8.14.2.1 Persistent Volume Claim

A PersistentVolumeClaim is a request for storage. For a workload that requires persistent volumes, the workload should use PersistentVolumeClaim (PVC) to express its request on persistent storage. A PersistentVolumeClaim can request specific size and Access Modes[571] (for example, they can be mounted once read/write or many times read-only).

Any workload can specify a PersistentVolumeClaim. For example, a Pod may need a volume that is at least 4Gi large or a volume mounted under `/data` in the container's filesystem. If there is a PersistentVolume (PV) that satisfies the specified requirements in the PersistentVolumeClaim (PVC), it will be bound to the PVC before the Pod starts.

8.14.2.2 Related Information

- Kubernetes Storage[572]
- Kubernetes persistent storage design document[573]

571 https://kubernetes.io/docs/concepts/storage/persistent-volumes/#access-modes
572 https://kubernetes.io/docs/concepts/storage/
573 https://github.com/kubernetes/community/blob/master/contributors/design-proposals/storage/persistent-storage.md

8.14.3 Provision a Static Local Volume

Learn how to provision a static local volume for a DKP cluster

The `localvolumeprovisioner` component uses the local volume static provisioner[574] to manage persistent volumes for pre-allocated disks. It does this by watching the `/mnt/disks` folder on each host and creating persistent volumes in the `localvolumeprovisioner` storage class for each disk that it discovers in this folder.

- Persistent volumes with a 'Filesystem' volume-mode are discovered if you mount them under `/mnt/disks`.
- Persistent volumes with a 'Block' volume-mode are discovered if you create a symbolic link to the block device in `/mnt/disks`.

8.14.3.1 Before you begin

Before starting this tutorial, verify the following:

- You have access to a Linux, macOS, or Windows computer with a supported operating system version.
- You have a provisioned `dkp` cluster that uses the `localvolumeprovisioner` platform application, but have not added any other Kommander applications to the cluster yet.

For this tutorial, you **do not deploy** using all the default settings as described in the Quick start guide(see page 37).

This distinction between provisioning and deployment is important because some applications depend on the storage class provided by the `localvolumeprovisioner` component and can fail to start if it is not configured yet.

8.14.3.2 Provision the cluster and a volume

1. Create a pre-provisioned cluster by following the steps outlined in the Pre-provisioned Infrastructure(see page 222) topic.
   As volumes are created/mounted on the nodes, the local volume provisioner detects each volume in the `/mnt/disks` directory and adds it as a persistent volume with the `localvolumeprovisioner` storage class.
2. Create at least one volume in `/mnt/disks` on each host.
   For example, mount a `tmpfs` volume:

   ```
   mkdir -p /mnt/disks/example-volume && mount -t tmpfs example-volume /mnt/disks/example-volume
   ```

3. Verify the persistent volume by running the following command:

   ```
   kubectl get pv
   ```

574 https://github.com/kubernetes-sigs/sig-storage-local-static-provisioner

The command displays output similar to the following:

```
NAME                CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS
CLAIM   STORAGECLASS             REASON   AGE
local-pv-4c7fc8ba   3986Mi     RWO            Delete           Available
localvolumeprovisioner            2s
```

4. Claim the persistent volume using a PersistentVolumeClaim, by running the following command:

```
cat <<EOF | kubectl create -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: example-claim
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
  storageClassName: localvolumeprovisioner
EOF
```

5. Reference the persistent volume claim in a pod by running the following command:

```
cat <<EOF | kubectl create -f -
apiVersion: v1
kind: Pod
metadata:
  name: pod-with-persistent-volume
spec:
  containers:
    - name: frontend
      image: nginx
      volumeMounts:
        - name: data
          mountPath: "/var/www/html"
  volumes:
    - name: data
      persistentVolumeClaim:
        claimName: example-claim
EOF
```

6. Verify the persistent volume claim by running the following command:

```
kubectl get pvc
```

The command displays output similar to the following:

```
NAME            STATUS   VOLUME              CAPACITY   ACCESS MODES
STORAGECLASS             AGE
example-claim   Bound    local-pv-4c7fc8ba   3986Mi     RWO
localvolumeprovisioner   78s
```

```
NAME                CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS
CLAIM                   STORAGECLASS            REASON   AGE
local-pv-4c7fc8ba   3986Mi     RWO            Delete           Bound
default/example-claim   localvolumeprovisioner            15m
```

Upon deletion of the persistent volume claim, the corresponding persistent volume resource uses the *Delete* reclaim policy, which removes all data on the volume.

8.14.4 Default Storage Providers in DKP

Default storage providers in DKP

When deploying DKP using a supported cloud provisioner (AWS), DKP automatically configures native storage drivers for the target platform. In addition, DKP deploys a default StorageClass[575] for dynamic persistent volume (PV)[576] creation. The table below lists the driver and default StorageClass for each supported cloud provisioner.

Cloud Provisioner	Driver	Default Storage Class
AWS	aws-ebs-csi-driver	awscsiprovisioner[577]
Azure	azuredisk-csi-driver	azurecsiprovisioner[578]
gCloud	gcp-filestore-csi-driver	gcpcsiprovisioner[579]

When a default StorageClass is specified, persistent volume claims (PVCs) can be created without needing to specify the storage class. For instance, to request a volume using the default provisioner, create a PVC with the following:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pv-claim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
```

575 https://kubernetes.io/docs/concepts/storage/storage-classes/
576 https://kubernetes.io/docs/concepts/storage/dynamic-provisioning/
577 https://github.com/kubernetes-sigs/aws-ebs-csi-driver
578 https://github.com/kubernetes-sigs/azuredisk-csi-driver
579 https://github.com/kubernetes-sigs/gcp-filestore-csi-driver

```
    requests:
      storage: 4Gi
```

To start the provisioning of a volume, launch a pod which references the PVC:

```
apiVersion: v1
kind: Pod
...
spec:
  containers:
...
    volumeMounts:
    - mountPath: /data
      name: persistent-storage
...
  volumes:
  - name: persistent-storage
    persistentVolumeClaim:
      claimName: my-pv-claim
```

8.14.4.1 Multiple Storage Classes

The default `StorageClass` provisioned with DKP is acceptable for most workloads and offers a good cost to performance ratio. If your workload has different requirements, you can create additional `StorageClass` types with specific configurations.

In some instances you can change the default `StorageClass`. Refer to this procedure:

- Changing the default storage class[580]

8.14.4.2 Driver Information

All default drivers implement the Container Storage Interface[581] (CSI). The CSI provides a common abstraction to container orchestrators for interacting with storage subsystems of various types. Each driver has specific configuration parameters which effect PV provisioning. This section details the default configuration for drivers used with DKP. This section also has links to driver documentation, if further customization is required.

NOTE: `StorageClass` parameters cannot be changed after creation. To use a different volume configuration, you must create a new `StorageClass`.

Amazon Elastic Block Store (EBS) CSI Driver

DKP EBS default `StorageClass`:

580 https://kubernetes.io/docs/tasks/administer-cluster/change-default-storage-class/
581 https://github.com/container-storage-interface/spec/blob/master/spec.md

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  annotations:
    storageclass.kubernetes.io/is-default-class: "true" # This tells kubernetes to
make this the default storage class
  name: ebs-sc
provisioner: ebs.csi.aws.com
reclaimPolicy: Delete  # volumes are automatically reclaimed when no longer in use
and PVCs are deleted
volumeBindingMode: WaitForFirstConsumer #  Physical volumes will not be created until
a pod is created that uses the PVC, required to use CSI's Topology feature
parameters:
  csi.storage.k8s.io/fstype: ext4
  type: gp3 # General Purpose SSD
```

DKP deploys with gp3 (general purpose SSDs) EBS volumes.

- Driver documentation: aws-ebs-csi-driver[582]
- Volume types and pricing: volume types[583]

Azure CSI Driver

DKP deploys with StandardSSD_LRS for Azure Virtual Disks.

- Driver documentation: azuredisk-csi-driver[584]
- Volume types and pricing: volume types[585]

GCP CSI Driver

This driver allows volumes backed by Google Cloud Filestore instances to be dynamically created and mounted by workloads.

- Driver documentation: gcp-filestore-csi-driver[586]
- Persistent volumes and dynamic provisioning: volume types[587]

8.14.4.3 On-Premises and other storage options

In an on-premises environment, accessible storage can be used for PV and PVCs. Using the Kubernetes CSI and third party drivers, you can use your local volumes and other storage devices in your data center.

8.14.4.4 Related Information

- Kubernetes Storage[588]

582 https://github.com/kubernetes-sigs/aws-ebs-csi-driver
583 https://aws.amazon.com/ebs/features/
584 https://github.com/kubernetes-sigs/azuredisk-csi-driver
585 https://azure.microsoft.com/en-us/pricing/details/storage/page-blobs/
586 https://github.com/kubernetes-sigs/gcp-filestore-csi-driver
587 https://cloud.google.com/kubernetes-engine/docs/concepts/persistent-volumes
588 https://kubernetes.io/docs/concepts/storage/

- Kubernetes Local Persistent Volumes[589]

589 https://kubernetes.io/blog/2019/04/04/kubernetes-1.14-local-persistent-volumes-ga/

9 CLI and API Tools

CLI commands and API tools.

9.1 API Documentation

This document is automatically generated from the API definition in the code.

9.1.1 apps.kommander.mesosphere.io/v1alpha2

9.1.1.1 App

App is the Schema for a Service or Application in Kommander.

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[590]	false
spec		GenericAppSpec(see page 651)	false
status		AppStatus(see page 650)	false

9.1.1.2 AppDeployment

AppDeployment is the Schema for a concrete installation of a Service or Application in Kommander.

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[591]	false
spec		AppDeploymentSpec(see page 649)	false

590 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta
591 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta

Field	Description	Scheme	Required
status		AppDeploymentStatus (see page 649)	false

9.1.1.3 AppDeploymentList

AppDeploymentList contains a list of AppDeployments.

Field	Description	Scheme	Required
items		[]AppDeployment(see page 648)	true
metadata		metav1.ListMeta[592]	false

9.1.1.4 AppDeploymentSpec

AppDeploymentSpec defines an instance of an Application.

Field	Description	Scheme	Required
appRef	AppRef provides reference to a ClusterApp or App object.	TypedLocalObjectReference(see page 652)	true
configOverrides	ConfigOverrides allows a user to define a ConfigMap that contains configuration overrides	corev1.LocalObjectReference	false

9.1.1.5 AppDeploymentStatus

AppDeploymentStatus defines the current state of the AppDeployment.

9.1.1.6 AppList

AppList contains a list of Apps.

592 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta

Field	Description	Scheme	Required
items		[]App(see page 648)	true
metadata		metav1.ListMeta[593]	false

9.1.1.7 AppStatus

AppStatus defines the current state of an App.

9.1.1.8 ClusterApp

ClusterApp is the Schema for a Service or Application in Kommander.

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[594]	false
spec		GenericAppSpec(see page 651)	false
status		ClusterAppStatus(see page 651)	false

9.1.1.9 ClusterAppList

ClusterAppList contains a list of ClusterApps.

Field	Description	Scheme	Required
items		[]ClusterApp(see page 650)	true
metadata		metav1.ListMeta[595]	false

593 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta
594 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta
595 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta

9.1.1.10 ClusterAppStatus

ClusterAppStatus defines the current state of an App.

9.1.1.11 CrossNamespaceGitRepositoryReference

CrossNamespaceGitRepositoryReference contains enough information to let you locate the typed referenced object at cluster level.

Field	Description	Scheme	Required
apiVersion	API version of the referent	string	false
kind	Kind of the referent	string	true
name	Name of the referent	string	true
namespace	Namespace of the referent, defaults to the Kustomization namespace	string	false

9.1.1.12 GenericAppSpec

GenericAppSpec defines the actual Application spec.

Field	Description	Scheme	Required
appId	AppID specifies the name of the application/ workload	string	true
gitRepositoryRef	GitRepository is reference to the Flux's GitRepository, which in turn describes Git repository where the service resides	CrossNamespaceGitRepositoryReference(see page 651)	true
version	Version depicts the version of the service in the semantic versioning format.	string	true

9.1.1.13 TypedLocalObjectReference

TypedLocalObjectReference contains enough information to let you locate the typed referenced object at cluster or local level.

Field	Description	Scheme	Required
apiVersion	API version of the referent	string	false
kind	Kind of the referent	string	true
name	Name of the referent	string	true

9.1.2 apps.kommander.mesosphere.io/v1alpha3

9.1.2.1 App

App is the Schema for a Service or Application in Kommander.

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[596]	false
spec		GenericAppSpec(see page 658)	false
status		AppStatus(see page 655)	false

9.1.2.2 AppDeployment

AppDeployment is the Schema for a concrete installation of a Service or Application in Kommander.

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[597]	false
spec		AppDeploymentSpec(see page 655)	false

596 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta
597 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta

Field	Description	Scheme	Required
status		AppDeploymentStatus (see page 654)	false

9.1.2.3 AppDeploymentClusterCondition

Field	Description	Scheme	Required
lastTransitionTime	LastTransitionTime is the last time the condition transitioned from one status to another.	metav1.Time	false
status	Status of the condition, one of True, False, Unknown	metav1.ConditionStatus	true
type	Type indicates type of condition in CamelCase or in foo.example.com/CamelCase[598].	AppDeploymentConditionType	true

9.1.2.4 AppDeploymentList

AppDeploymentList contains a list of AppDeployments.

Field	Description	Scheme	Required
items		[]AppDeployment(see page 652)	true
metadata		metav1.ListMeta[599]	false

9.1.2.5 AppDeploymentSpec

AppDeploymentSpec defines an instance of an Application.

598 http://foo.example.com/CamelCase
599 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta

Field	Description	Scheme	Required
appRef	AppRef provides reference to a ClusterApp or App object.	TypedLocalObjectReference(see page 658)	true
clusterConfigOverrides	ClusterConfigOverrides defines a list of config overrides configmaps and label selectors to select clusters on which to apply the config overrides.	[]ClusterConfigOverrides(see page 656)	false
clusterSelector	ClusterSelector defines the label selectors to select clusters on which to enable the AppDeployment.	metav1.LabelSelector[600]	false
configOverrides	ConfigOverrides allows a user to define a ConfigMap that contains configuration overrides at the workspace level.	corev1.LocalObjectReference	false

9.1.2.6 AppDeploymentStatus

AppDeploymentStatus defines the current state of the AppDeployment.

600 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#labelselector-v1-meta

Field	Description	Scheme	Required
clusters	Clusters tracks clusters that have the AppDeployment enabled along with a ref to the cluster config overrides configmap in the management cluster. Existence of cluster means that the cluster's default AppDeployment state has been processed. Nonexistence of cluster means that the cluster was newly attached and needs to be selected by default to enable the AppDeployment on it.	[] ClusterDeploymentStatus(see page 657)	false
observedGeneration	ObservedGeneration is the most recent generation observed for this AppDeployment. It corresponds to the AppDeployment's generation, which is updated on mutation by the API Server.	int64	false

9.1.2.7 AppList

AppList contains a list of Apps.

Field	Description	Scheme	Required
items		[]App(see page 652)	true
metadata		metav1.ListMeta[601]	false

9.1.2.8 AppStatus

AppStatus defines the current state of an App.

601 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta

9.1.2.9 ClusterApp

ClusterApp is the Schema for a Service or Application in Kommander.

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[602]	false
spec		GenericAppSpec(see page 658)	false
status		ClusterAppStatus(see page 656)	false

9.1.2.10 ClusterAppList

ClusterAppList contains a list of ClusterApps.

Field	Description	Scheme	Required
items		[]ClusterApp(see page 656)	true
metadata		metav1.ListMeta[603]	false

9.1.2.11 ClusterAppStatus

ClusterAppStatus defines the current state of an App.

9.1.2.12 ClusterConfigOverrides

ClusterConfigOverrides defines the cluster config overrides configmap and label selector to select clusters on which to apply the overrides configmap.

602 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta
603 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta

Field	Description	Scheme	Required
clusterSelector	ClusterSelector defines the label selectors to select clusters on which to deploy the configmap.	metav1.LabelSelector[604]	false
configMapName	ConfigMapName is the name of the cluster config overrides configmap.	string	true

9.1.2.13 ClusterDeploymentStatus

Field	Description	Scheme	Required
clusterConfigOverridesRef	ClusterConfigOverridesRef is set to reference the overrides ConfigMap in the management cluster	corev1.LocalObjectReference	false
conditions		[] AppDeploymentClusterCondition(see page 653)	false
name		string	true

9.1.2.14 CrossNamespaceGitRepositoryReference

CrossNamespaceGitRepositoryReference contains enough information to let you locate the typed referenced object at cluster level.

Field	Description	Scheme	Required
apiVersion	API version of the referent	string	false
kind	Kind of the referent	string	true
name	Name of the referent	string	true

604 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#labelselector-v1-meta

Field	Description	Scheme	Required
namespace	Namespace of the referent, defaults to the Kustomization namespace	string	false

9.1.2.15 GenericAppSpec

GenericAppSpec defines the actual Application spec.

Field	Description	Scheme	Required
appId	AppID specifies the name of the application/ workload	string	true
gitRepositoryRef	GitRepository is reference to the Flux's GitRepository, which in turn describes Git repository where the service resides	CrossNamespaceGitRepositoryReference(see page 657)	true
version	Version depicts the version of the service in the semantic versioning format.	string	true

9.1.2.16 TypedLocalObjectReference

TypedLocalObjectReference contains enough information to let you locate the typed referenced object at cluster or local level.

Field	Description	Scheme	Required
apiVersion	API version of the referent	string	false
kind	Kind of the referent	string	true
name	Name of the referent	string	true

9.1.3 dispatch.d2iq.io/v1alpha2

9.1.3.1 GitopsRepository

GitopsRepository represents an SCM repository that backs a gitops resource. A gitops resource can be a FluxCD HelmRelease or Kustomization resource.

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[605]	false
spec		GitopsRepositorySpec(see page 659)	true

9.1.3.2 GitopsRepositoryList

GitopsRepositoryList contains a list of Repository.

Field	Description	Scheme	Required
items		[]GitopsRepository(see page 659)	true
metadata		metav1.ListMeta[606]	false

9.1.3.3 GitopsRepositorySpec

GitopsRepositorySpec defines the desired state of GitopsRepository.

Field	Description	Scheme	Required
cloneUrl	URL to clone the repository from.	string	false

605 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta
606 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta

Field	Description	Scheme	Required
secret	Reference to the secret to use when interacting with the Git provider API. The secret should contain the following fields:\n username: the username (if password is not a token).\n password: the password or token (required).	string	false
template	Template of the gitops resource backing the repository.	GitopsRolloutTemplate (see page 660)	false

9.1.3.4 GitopsRolloutTemplate

Field	Description	Scheme	Required
path	Root path to fetch manifests from in the Git repository.	string	false
ref	The Git reference to checkout and monitor for changes, defaults to master branch. This field supersedes the “revision” field in v1alpha1 API and is a breaking change.	sourcectrlv1beta1.GitRepositoryRef	false
suspend	Whether to suspend periodic or webhook-notified sync.	bool	false

9.1.4 kommander.mesosphere.io/v1beta1

9.1.4.1 CAPIClusterReference

Field	Description	Scheme	Required
name		string	true

Field	Description	Scheme	Required
namespace		string	false

9.1.4.2 ClusterReference

ClusterReference holds a single reference to clusters provisioned via Kommander. Only one field is allowed to be set. Currently, only CAPI clusters are creatable, but this is left extensible for other provider types in the future.

Field	Description	Scheme	Required
capiCluster		CAPIClusterReference(see page 660)	false

9.1.4.3 IngressSpec

IngressSpec holds settings for the cluster's Kommander managed Ingress.

Field	Description	Scheme	Required
certificateSecretRef	CertificateSecretRef is a reference to a secret of type TLS that holds a TLS certificate, private key and CA certificate. The certificate must be valid for the Ingress hostname. The secret must be located in the workspace's namespace on the target cluster.	v1.LocalObjectReference	false
hostname	Hostname can be set to configure the cluster's Ingress with a custom domain name.	string	false

Field	Description	Scheme	Required
issuerRef	IssuerRef is a reference to an `Issuer` or `ClusterIssuer` on the target cluster to be used to create a `Certificate` for the cluster's Ingress. If the type is an Issuer, it must be located in the workspace's namespace on the cluster.	IssuerReference(see page 663)	false

9.1.4.4 IngressStatus

IngressSpec holds information about the cluster's Kommander managed Ingress.

Field	Description	Scheme	Required
address	Address is the main DNS name or IP address of the cluster's Ingress that should be used to access services on the cluster. If the field is empty, it means that the load balancer hasn't been populated yet.	string	false
caBundle	CABundle is a PEM encoded CA bundle which should be used to verify the TLS connection for the Ingress-served end-entity certificate.	[]byte	false

Field	Description	Scheme	Required
useSystemCA	UseSystemCA is set to true if the Ingress end-entity certificate was issued by a public CA and it is expected that the root CA cert is included in a OS CA bundle (which should be used for TLS verification in that case). If this field is `true` the `CABundle` field is empty. default: false	bool	false

9.1.4.5 IssuerReference

IssuerReference is a reference to an issuer with a given name, kind and group.

Field	Description	Scheme	Required
group	Group of the issuer being referred to.	string	false
kind	Kind of the issuer being referred to.	string	false
name	Name of the issuer being referred to.	string	true

9.1.4.6 KommanderCluster

KommanderCluster is the Schema for the kommander clusters API.

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[607]	false
spec		KommanderClusterSpec (see page 664)	false

607 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta

Field	Description	Scheme	Required
status		KommanderClusterStatus(see page 665)	false

9.1.4.7 KommanderClusterCondition

Field	Description	Scheme	Required
lastTransitionTime		metav1.Time	false
message		string	false
reason		string	false
status		metav1.ConditionStatus	true
type		KommanderClusterConditionType	true

9.1.4.8 KommanderClusterList

KommanderClusterList contains a list of Cluster.

Field	Description	Scheme	Required
items		[]KommanderCluster(see page 663)	true
metadata		metav1.ListMeta[608]	false

9.1.4.9 KommanderClusterSpec

KommanderClusterSpec defines the desired state of Cluster.

608 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta

Field	Description	Scheme	Required
clusterRef		ClusterReference(see page 661)	false
clusterTunnelConnectorRef	ClusterTunnelConnectorRef is a reference to TunnelConnector that should be used for connecting to cluster.	corev1.LocalObjectReference	false
kubeconfigRef		corev1.LocalObjectReference	false

9.1.4.10 KommanderClusterStatus

KommanderClusterStatus defines the observed state of Cluster.

Field	Description	Scheme	Required
clusterId	KubernetesClusterID is the stable cluster ID of the Kubernetes cluster that this Kommander cluster represents.	string	false
conditions		[] KommanderClusterCondition(see page 664)	false
dextfaclientRef	DexTFAClientRef holds a reference to a DexClient provisioned for Traefik Forward Auth running on managed cluster.	corev1.ObjectReference	false
kubefedclusterRef	KubefedClusterRef holds a reference to a kubefedcluster in the kubefed system namespace.	corev1.LocalObjectReference	false
kubernetesVersion	KubernetesVersion is the Kubernetes version of the cluster.	string	false

Field	Description	Scheme	Required
phase	Phase represents the current phase of cluster actuation. E.g. Pending, Provisioning, Provisioned, Deleting, Failed, etc.	KommanderClusterPhase	false
serviceEndpoints	ServiceEndpoints will be the addresses assigned to the Kubernetes exposed services	map[string]string	false
type	ClusterType represents the type of the cluster. E.g. EKS, GKE, etc.	KommanderClusterType	false

9.1.4.11 License

License is the Schema for the licenses API.

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[609]	false
spec		LicenseSpec(see page 668)	false
status		LicenseStatus(see page 668)	false

9.1.4.12 LicenseCondition

Field	Description	Scheme	Required
lastTransitionTime		metav1.Time	false
message		string	false
reason		string	false

609 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta

Field	Description	Scheme	Required
status		metav1.ConditionStatus	true
type		LicenseConditionType	true

9.1.4.13 LicenseExternalAWS

Field	Description	Scheme	Required
licenseArn	LicenseArn is a fully qualified AWS arn to a license for Kommander.	string	false

9.1.4.14 LicenseExternalReference

Field	Description	Scheme	Required
awsLicense	AWSLicense holds a single reference to a license in AWS's License Manager	LicenseExternalAWS(see page 667)	false
type	Type is the source of the external license referenced, i.e. AWS, GCP, Azure	LicenseExternalReferenceType	true

9.1.4.15 LicenseList

LicenseList contains a list of License.

Field	Description	Scheme	Required
items		[]License(see page 666)	true
metadata		metav1.ListMeta[610]	false

610 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta

9.1.4.16 LicenseSpec

LicenseSpec defines the desired state of License.

Field	Description	Scheme	Required
externalLicenseRef	ExternalLicenseRef holds a reference to an external license	LicenseExternalReference(see page 667)	false
licenseRef	LicenseReference holds a single reference to the secret holding the license JWT	corev1.LocalObjectReference	false

9.1.4.17 LicenseStatus

LicenseStatus defines the observed state of License.

Field	Description	Scheme	Required
clusterCapacity	Maximum number of clusters that the license allows.	int32	true
conditions	Conditions relevant to the license (currently used to track term breaches)	[]LicenseCondition(see page 666)	false
coreCapacity	Maximum number of cores that the license allows.	int32	true
customerId	The customer's ID. This is the customer name provided from Salesforce.	string	true
dkpLevel	The DKP license level.	string	true
endDate	End date of the licensing period.	metav1.Time	false

Field	Description	Scheme	Required
licenseId	The license's ID as provided from Salesforce.	string	true
productName	The product name the license is for.	string	true
productSKU	The product SKU the license is for.	string	true
startDate	Start date of the licensing period.	metav1.Time	false
valid	Indicates whether the license is valid, i.e. the secret containing the JWT exists and the JWT carries a valid D2iQ signature. This does NOT indicate whether the license has expired or terms have been breached.	bool	true
version	The license's version ID as provided when the license was created.	string	true

9.1.4.18 VirtualGroup

VirtualGroup is the Schema for the virtualgroups API.

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[611]	false
spec		VirtualGroupSpec(see page 671)	false

611 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta

9.1.4.19 VirtualGroupClusterRoleBinding

VirtualGroupClusterRoleBinding is the Schema for the virtualgroupclusterrolebindings API.

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[612]	false
spec		VirtualGroupClusterRoleBindingSpec(see page 670)	true

9.1.4.20 VirtualGroupClusterRoleBindingList

VirtualGroupClusterRoleBindingList contains a list of VirtualGroupClusterRoleBinding.

Field	Description	Scheme	Required
items		[] VirtualGroupClusterRoleBinding(see page 670)	true
metadata		metav1.ListMeta[613]	false

9.1.4.21 VirtualGroupClusterRoleBindingSpec

VirtualGroupClusterRoleBindingSpec defines the desired state of VirtualGroupClusterRoleBinding.

Field	Description	Scheme	Required
clusterRoleRef		corev1.LocalObjectReference	true
placement		PlacementSelector	false
virtualGroupRef		corev1.LocalObjectReference	true

612 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta
613 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta

9.1.4.22 VirtualGroupList

VirtualGroupList contains a list of VirtualGroup.

Field	Description	Scheme	Required
items		[]VirtualGroup(see page 669)	true
metadata		metav1.ListMeta[614]	false

9.1.4.23 VirtualGroupSpec

VirtualGroupSpec defines the desired state of VirtualGroup.

Field	Description	Scheme	Required
subjects		[]rbacv1.Subject	false

9.1.5 workspaces.kommander.mesosphere.io/v1alpha1

9.1.5.1 KommanderProjectRole

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[615]	false
spec		KommanderProjectRoleSpec(see page 672)	false
status		KommanderProjectRoleStatus(see page 672)	false

614 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta
615 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta

9.1.5.2 KommanderProjectRoleList

Field	Description	Scheme	Required
items		[]KommanderProjectRole (see page 671)	true
metadata		metav1.ListMeta[616]	false

9.1.5.3 KommanderProjectRoleSpec

Field	Description	Scheme	Required
projectObjectVerbs		[]string	false
rules		[]rbacv1.PolicyRule	false

9.1.5.4 KommanderProjectRoleStatus

Field	Description	Scheme	Required
roleInProjectRef		corev1.LocalObjectReference	false
roleInWorkspaceRef		corev1.LocalObjectReference	false

9.1.5.5 KommanderWorkspaceRole

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[617]	false

616 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta
617 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta

Field	Description	Scheme	Required
spec		KommanderWorkspaceRoleSpec(see page 673)	false
status		KommanderWorkspaceRoleStatus(see page 673)	false

9.1.5.6 KommanderWorkspaceRoleList

Field	Description	Scheme	Required
items		[] KommanderWorkspaceRole(see page 672)	true
metadata		metav1.ListMeta[618]	false

9.1.5.7 KommanderWorkspaceRoleSpec

Field	Description	Scheme	Required
rules		[]rbacv1.PolicyRule	false
workspaceObjectVerbs		[]string	false

9.1.5.8 KommanderWorkspaceRoleStatus

Field	Description	Scheme	Required
clusterRoleRef		corev1.LocalObjectReference	false
roleInWorkspaceRef		corev1.LocalObjectReference	false

618 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta

9.1.5.9 Project

Project is a logical top-level container for a set of Kommander resources.

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[619]	false
spec		ProjectSpec(see page 676)	false
status		ProjectStatus(see page 676)	false

9.1.5.10 ProjectCondition

Field	Description	Scheme	Required
lastTransitionTime	Last time the condition transitioned from one status to another.	metav1.Time	false
message	A human readable message indicating details about the transition.	string	false
reason	The reason for the condition's last transition.	string	false
status	Status of the condition, one of True, False, Unknown.	corev1.ConditionStatus	true
type	Type of project condition.	ProjectConditionType	true

9.1.5.11 ProjectList

ProjectList is a list of Project objects.

619 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta

Field	Description	Scheme	Required
items		[]Project(see page 674)	true
metadata		metav1.ListMeta[620]	false

9.1.5.12 ProjectRole

ProjectRole is the Schema for the workspaces ProjectRole API.

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[621]	false
spec		ProjectRoleSpec(see page 675)	false
status		ProjectRoleStatus(see page 676)	false

9.1.5.13 ProjectRoleList

ProjectRoleList contains a list of ProjectRole.

Field	Description	Scheme	Required
items		[]ProjectRole(see page 675)	true
metadata		metav1.ListMeta[622]	false

9.1.5.14 ProjectRoleSpec

Field	Description	Scheme	Required
rules		[]rbacv1.PolicyRule	false

620 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta
621 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta
622 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta

9.1.5.15 ProjectRoleStatus

Field	Description	Scheme	Required
federatedRoleRef		corev1.LocalObjectReference	false

9.1.5.16 ProjectSpec

ProjectSpec describes the attributes on a Project.

Field	Description	Scheme	Required
namespaceName	NamespaceName specifies the optional namespace name to use for the project. This field is immutable, only settable on create.	string	false
placement		v1beta1.PlacementSelector	false
workspaceRef		corev1.LocalObjectReference	false

9.1.5.17 ProjectStatus

ProjectStatus is information about the current status of a Project.

Field	Description	Scheme	Required
conditions	Represents the latest available observations of a project's current state.	[]ProjectCondition(see page 674)	false
namespaceRef		corev1.LocalObjectReference	false

9.1.5.18 VirtualGroupKommanderClusterRoleBinding

VirtualGroupKommanderClusterRoleBinding is the Schema for the VirtualGroupWorkspaceRoleBinding API.

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[623]	false
spec		VirtualGroupKommanderClusterRoleBindingSpec (see page 677)	true
status		VirtualGroupKommanderClusterRoleBindingStatus (see page 678)	false

9.1.5.19 VirtualGroupKommanderClusterRoleBindingList

VirtualGroupKommanderClusterRoleBindingList contains a list of VirtualGroupKommanderClusterRoleBinding.

Field	Description	Scheme	Required
items		[] VirtualGroupKommanderClusterRoleBinding (see page 677)	true
metadata		metav1.ListMeta[624]	false

9.1.5.20 VirtualGroupKommanderClusterRoleBindingSpec

Field	Description	Scheme	Required
clusterRoleRef		corev1.LocalObjectReference	true
virtualGroupRef		corev1.LocalObjectReference	true

623 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta
624 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta

9.1.5.21 VirtualGroupKommanderClusterRoleBindingStatus

Field	Description	Scheme	Required
clusterRoleBindingRef		corev1.LocalObjectReference	false

9.1.5.22 VirtualGroupKommanderProjectRoleBinding

VirtualGroupKommanderProjectRoleBinding is the Schema to be used in the API.

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[625]	false
spec		VirtualGroupKommanderProjectRoleBindingSpec (see page 679)	true
status		VirtualGroupKommanderProjectRoleBindingStatus (see page 679)	false

9.1.5.23 VirtualGroupKommanderProjectRoleBindingList

VirtualGroupKommanderProjectRoleBindingList contains a list of VirtualGroupKommanderProjectRoleBinding.

Field	Description	Scheme	Required
items		[] VirtualGroupKommanderProjectRoleBinding (see page 678)	true
metadata		metav1.ListMeta[626]	false

625 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta
626 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta

9.1.5.24 VirtualGroupKommanderProjectRoleBindingSpec

Field	Description	Scheme	Required
kommanderProjectRoleRef		corev1.LocalObjectReference	true
virtualGroupRef		corev1.LocalObjectReference	true

9.1.5.25 VirtualGroupKommanderProjectRoleBindingStatus

Field	Description	Scheme	Required
roleBindingInProjectRef		corev1.LocalObjectReference	false
roleBindingInWorkspaceRef		corev1.LocalObjectReference	false

9.1.5.26 VirtualGroupKommanderWorkspaceRoleBinding

VirtualGroupKommanderWorkspaceRoleBinding is the Schema to be used in the API.

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[627]	false
spec		VirtualGroupKommanderWorkspaceRoleBindingSpec(see page 680)	true
status		VirtualGroupKommanderWorkspaceRoleBindingStatus(see page 680)	false

627 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta

9.1.5.27 VirtualGroupKommanderWorkspaceRoleBindingList

VirtualGroupKommanderWorkspaceRoleBindingList contains a list of VirtualGroupKommanderWorkspaceRoleBinding.

Field	Description	Scheme	Required
items		[] VirtualGroupKommanderWorkspaceRoleBinding (see page 679)	true
metadata		metav1.ListMeta[628]	false

9.1.5.28 VirtualGroupKommanderWorkspaceRoleBindingSpec

Field	Description	Scheme	Required
kommanderWorkspaceRoleRef		corev1.LocalObjectReference	true
virtualGroupRef		corev1.LocalObjectReference	true

9.1.5.29 VirtualGroupKommanderWorkspaceRoleBindingStatus

Field	Description	Scheme	Required
clusterRoleBindingRef		corev1.LocalObjectReference	false
roleBindingInWorkspaceRef		corev1.LocalObjectReference	false

9.1.5.30 VirtualGroupProjectRoleBinding

VirtualGroupProjectRoleBinding is the Schema for the VirtualGroupProjectRoleBinding API.

628 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[629]	false
spec		VirtualGroupProjectRoleBindingSpec(see page 681)	true
status		VirtualGroupProjectRoleBindingStatus(see page 682)	false

9.1.5.31 VirtualGroupProjectRoleBindingList

VirtualGroupProjectRoleBindingList contains a list of VirtualGroupProjectRoleBinding.

Field	Description	Scheme	Required
items		[] VirtualGroupProjectRoleBinding(see page 680)	true
metadata		metav1.ListMeta[630]	false

9.1.5.32 VirtualGroupProjectRoleBindingSpec

Field	Description	Scheme	Required
projectRoleRef		corev1.LocalObjectReference	false
virtualGroupRef		corev1.LocalObjectReference	true

629 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta
630 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta

Field	Description	Scheme	Required
workspaceRoleRef	WorkspaceRoleRef maybe a LocalObjectReference but the WorkspaceRole is not created in project namespace but in Workspace namespace. “Local” in LocalObjectReference means “Local to project’s workspace” since there can only be one workspace the project is in.	corev1.LocalObjectReference	false

9.1.5.33 VirtualGroupProjectRoleBindingStatus

Field	Description	Scheme	Required
federatedRoleBindingRef		corev1.LocalObjectReference	false

9.1.5.34 VirtualGroupWorkspaceRoleBinding

VirtualGroupWorkspaceRoleBinding is the Schema for the VirtualGroupWorkspaceRoleBinding API.

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[631]	false
spec		VirtualGroupWorkspaceRoleBindingSpec(see page 683)	true
status		VirtualGroupWorkspaceRoleBindingStatus(see page 683)	false

631 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta

9.1.5.35 VirtualGroupWorkspaceRoleBindingList

VirtualGroupWorkspaceRoleBindingList contains a list of VirtualGroupWorkspaceRoleBinding.

Field	Description	Scheme	Required
items		[] VirtualGroupWorkspaceRoleBinding(see page 682)	true
metadata		metav1.ListMeta[632]	false

9.1.5.36 VirtualGroupWorkspaceRoleBindingSpec

Field	Description	Scheme	Required
placement		v1beta1.PlacementSelector	false
virtualGroupRef		corev1.LocalObjectReference	true
workspaceRoleRef		corev1.LocalObjectReference	true

9.1.5.37 VirtualGroupWorkspaceRoleBindingStatus

Field	Description	Scheme	Required
federatedClusterRoleBindingRef		corev1.LocalObjectReference	false

9.1.5.38 Workspace

Workspace is the Schema for the workspaces API.

632 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[633]	false
spec		WorkspaceSpec(see page 686)	false
status		WorkspaceStatus(see page 686)	false

9.1.5.39 WorkspaceCondition

Field	Description	Scheme	Required
lastTransitionTime	Last time the condition transitioned from one status to another.	metav1.Time	false
message	A human readable message indicating details about the transition.	string	false
reason	The reason for the condition's last transition.	string	false
status	Status of the condition, one of True, False, Unknown.	corev1.ConditionStatus	true
type	Type of workspace condition.	WorkspaceConditionType	true

9.1.5.40 WorkspaceList

WorkspaceList contains a list of Workspace.

633 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta

Field	Description	Scheme	Required
items		[]Workspace(see page 683)	true
metadata		metav1.ListMeta[634]	false

9.1.5.41 WorkspaceRole

WorkspaceRole is the Schema for the workspaces API.

Field	Description	Scheme	Required
metadata		metav1.ObjectMeta[635]	false
spec		WorkspaceRoleSpec(see page 686)	false
status		WorkspaceRoleStatus(see page 686)	false

9.1.5.42 WorkspaceRoleList

WorkspaceRoleList contains a list of WorkspaceRole.

Field	Description	Scheme	Required
items		[]WorkspaceRole(see page 685)	true
metadata		metav1.ListMeta[636]	false

634 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta
635 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#objectmeta-v1-meta
636 https://v1-21.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/#listmeta-v1-meta

9.1.5.43 WorkspaceRoleSpec

Field	Description	Scheme	Required
aggregationRule		rbacv1.AggregationRule	false
rules		[]rbacv1.PolicyRule	false

9.1.5.44 WorkspaceRoleStatus

Field	Description	Scheme	Required
federatedClusterRoleRef		corev1.LocalObjectReference	false

9.1.5.45 WorkspaceSpec

Field	Description	Scheme	Required
clusterLabels		map[string]string	false
namespaceName	NamespaceName specifies the optional namespace name to use for the workspace. This field is immutable, only settable on create.	string	false

9.1.5.46 WorkspaceStatus

Field	Description	Scheme	Required
conditions	Represents the latest available observations of a workspace's current state.	[]WorkspaceCondition(see page 684)	false

Field	Description	Scheme	Required
namespaceRef		corev1.LocalObjectReference	false

9.1.6 Enterprise badge

ENTERPRISE

9.2 CLI Commands

9.2.1 CLI Commands for DKP

The table of contents on the left lists the command groupings. Inside each section, you will find the individual commands for most actions.

9.2.2 dkp attach

Attach one of [cluster]

9.2.2.1 Options

```
      --config string            Config file to use (default "/root/.kommander/
config")
      --context string           The name of the kubeconfig context to use
  -h, --help                     help for attach
      --kubeconfig string        Path to the kubeconfig file to use for CLI requests.
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
```

9.2.2.2 Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

9.2.2.3 SEE ALSO

- dkp attach cluster(see page 688) - Attach a cluster

9.2.2.4 dkp attach cluster

Attach a cluster

```
dkp attach cluster -n NAME --attached-kubeconfig FILENAME [flags]
```

Options

```
      --attached-kubeconfig string   Path of the kubeconfig file of the cluster to be
attached
  -h, --help                         help for cluster
  -n, --name string                  Desired name of the attached cluster
  -w, --workspace string             Name of the workspace of the attached cluster
```

Options inherited from parent commands

```
      --config string            Config file to use (default "/root/.kommander/
config")
      --context string           The name of the kubeconfig context to use
      --kubeconfig string        Path to the kubeconfig file to use for CLI requests.
```

```
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
  -v, --verbose int              Output verbosity
```

SEE ALSO

- dkp attach(see page 687) - Attach one of [cluster]

9.2.3 dkp check

Check, one of [cluster]

9.2.3.1 Options

```
  -h, --help   help for check
```

9.2.3.2 Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

9.2.3.3 SEE ALSO

- dkp check cluster(see page 689) - Check a cluster, one of [fips]

9.2.3.4 dkp check cluster

Check a cluster, one of [fips]

Options

```
  -h, --help   help for cluster
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp check(see page 689) - Check, one of [cluster]
- dkp check cluster fips(see page 690) - Validate the components in your cluster are FIPS compliant

dkp check cluster fips

Validate the components in your cluster are FIPS compliant

Synopsis

The check cluster fips command is used to validate that specific components and services are FIPS compliant by checking the signatures of the files against a signed signature file, and checking that services are using the certified algorithms.

Examples:

```
With a signature file named "manifest-rhel-84.json.asc" run:

dkp check cluster fips \
    --signature-file manifest-rhel-84.json.asc \
    --signature-configmap prod-rhel-84-fips-signatures \
    --output-configmap prod-rhel-84-fips-validation

If you already have a signature ConfigMap, you can omit the signature-file flag:

dkp check cluster fips \
    --signature-configmap prod-rhel-84-fips-signatures \
    --output-configmap prod-rhel-84-fips-validation

The validation will be re-checked against the existing signature data.
```

```
dkp check cluster fips [flags]
```

Options

```
  -h, --help                          help for fips
      --kubeconfig string             Path to the kubeconfig file for the fips
cluster. If unspecified, default discovery rules apply.
  -n, --namespace string              If present, the namespace scope for this CLI
request. (default "default")
      --output-configmap string       ConfigMap with fips signature data to verify.
[required]
      --signature-configmap string    ConfigMap with fips signature data to verify.
[required]
      --signature-file string         File containing fips signature data.
      --timeout duration              The length of time to wait before giving up.
Zero means wait forever. (default 10m0s)
```

Options inherited from parent commands

```
-v, --verbose int   Output verbosity
```

SEE ALSO

- dkp check cluster(see page 689) - Check a cluster, one of [fips]

9.2.4 dkp completion

Generate the autocompletion script for the specified shell

9.2.4.1 Synopsis

Generate the autocompletion script for dkp for the specified shell. See each sub-command's help for details on how to use the generated script.

9.2.4.2 Options

```
-h, --help   help for completion
```

9.2.4.3 Options inherited from parent commands

```
-v, --verbose int   Output verbosity
```

9.2.4.4 SEE ALSO

- dkp completion bash(see page 692) - Generate the autocompletion script for bash
- dkp completion fish(see page 691) - Generate the autocompletion script for fish
- dkp completion powershell(see page 694) - Generate the autocompletion script for powershell
- dkp completion zsh(see page 693) - Generate the autocompletion script for zsh

9.2.4.5 dkp completion fish

Generate the autocompletion script for fish

Synopsis

Generate the autocompletion script for the fish shell.

To load completions in your current shell session:

```
dkp completion fish | source
```

To load completions for every new session, execute once:

```
dkp completion fish > ~/.config/fish/completions/dkp.fish
```

You will need to start a new shell for this setup to take effect.

```
dkp completion fish [flags]
```

Options

```
  -h, --help              help for fish
      --no-descriptions   disable completion descriptions
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp completion(see page 691) - Generate the autocompletion script for the specified shell

9.2.4.6 dkp completion bash

Generate the autocompletion script for bash

Synopsis

Generate the autocompletion script for the bash shell.

This script depends on the 'bash-completion' package. If it is not installed already, you can install it via your OS's package manager.

To load completions in your current shell session:

```
source <(dkp completion bash)
```

To load completions for every new session, execute once:

Linux:

```
dkp completion bash > /etc/bash_completion.d/dkp
```

macOS:

```
dkp completion bash > /usr/local/etc/bash_completion.d/dkp
```

You will need to start a new shell for this setup to take effect.

```
dkp completion bash
```

Options

```
  -h, --help              help for bash
      --no-descriptions   disable completion descriptions
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp completion(see page 691) - Generate the autocompletion script for the specified shell

9.2.4.7 dkp completion zsh

Generate the autocompletion script for zsh

Synopsis

Generate the autocompletion script for the zsh shell.

If shell completion is not already enabled in your environment you will need to enable it. You can execute the following once:

```
echo "autoload -U compinit; compinit" >> ~/.zshrc
```

To load completions for every new session, execute once:

Linux:

```
dkp completion zsh > "${fpath[1]}/_dkp"
```

macOS:

```
dkp completion zsh > /usr/local/share/zsh/site-functions/_dkp
```

You will need to start a new shell for this setup to take effect.

```
dkp completion zsh [flags]
```

Options

```
  -h, --help              help for zsh
      --no-descriptions   disable completion descriptions
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp completion(see page 691) - Generate the autocompletion script for the specified shell

9.2.4.8 dkp completion powershell

Generate the autocompletion script for powershell

Synopsis

Generate the autocompletion script for powershell.

To load completions in your current shell session:

```
dkp completion powershell | Out-String | Invoke-Expression
```

To load completions for every new session, add the output of the above command to your powershell profile.

```
dkp completion powershell [flags]
```

Options

```
  -h, --help              help for powershell
      --no-descriptions   disable completion descriptions
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp completion(see page 691) - Generate the autocompletion script for the specified shell

9.2.5 dkp config

Manage DKP Kommander's configuration

9.2.5.1 Options

```
      --config string            Config file to use (default "/root/.kommander/
config")
      --context string           The name of the kubeconfig context to use
  -h, --help                     help for config
      --kubeconfig string        Path to the kubeconfig file to use for CLI requests.
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
```

9.2.5.2 Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

9.2.5.3 SEE ALSO

- dkp config get(see page 696) - Retrieve DKP Kommander's configuration
- dkp config set(see page 697) - Modify DKP Kommander's configuration

9.2.5.4 dkp config get

Retrieve DKP Kommander's configuration

Options

```
  -h, --help   help for get
```

Options inherited from parent commands

```
      --config string           Config file to use (default "/root/.kommander/
config")
      --context string          The name of the kubeconfig context to use
      --kubeconfig string       Path to the kubeconfig file to use for CLI requests.
      --request-timeout string  The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
  -v, --verbose int             Output verbosity
```

SEE ALSO

- dkp config(see page 695) - Manage DKP Kommander's configuration
- dkp config get default-workspace(see page 696) - Displays the name of the configured default Workspace

dkp config get default-workspace

Displays the name of the configured default Workspace

```
dkp config get default-workspace [flags]
```

Options

```
  -h, --help   help for default-workspace
```

Options inherited from parent commands

```
      --config string           Config file to use (default "/root/.kommander/
config")
      --context string          The name of the kubeconfig context to use
      --kubeconfig string       Path to the kubeconfig file to use for CLI requests.
```

```
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
  -v, --verbose int              Output verbosity
```

SEE ALSO

- dkp config get(see page 696) - Retrieve DKP Kommander's configuration

9.2.5.5 dkp config set

Modify DKP Kommander's configuration

Options

```
  -h, --help   help for set
```

Options inherited from parent commands

```
      --config string            Config file to use (default "/root/.kommander/
config")
      --context string           The name of the kubeconfig context to use
      --kubeconfig string        Path to the kubeconfig file to use for CLI requests.
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
  -v, --verbose int              Output verbosity
```

SEE ALSO

- dkp config(see page 695) - Manage DKP Kommander's configuration
- dkp config set default-workspace(see page 697) - Set the name of the default Workspace to use in all commands when none is provided

dkp config set default-workspace

Set the name of the default Workspace to use in all commands when none is provided

```
dkp config set default-workspace [WORKSPACE] [flags]
```

Options

```
  -h, --help   help for default-workspace
```

Options inherited from parent commands

```
      --config string            Config file to use (default "/root/.kommander/
config")
      --context string           The name of the kubeconfig context to use
      --kubeconfig string        Path to the kubeconfig file to use for CLI requests.
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
  -v, --verbose int              Output verbosity
```

SEE ALSO

- dkp config set(see page 697) - Modify DKP Kommander's configuration

9.2.6 dkp create

Create one of [appdeployment, bootstrap, capi-components, chart-bundle, cluster, image-bundle, nodepool, workspace]

9.2.6.1 Options

```
  -h, --help   help for create
```

9.2.6.2 Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

9.2.6.3 SEE ALSO

- dkp create appdeployment(see page 699) - Create an AppDeployment
- dkp create bootstrap(see page 702) - Create bootstrap cluster
- dkp create capi-components(see page 700) - Create the CAPI components in the cluster
- dkp create chart-bundle(see page 703) - Create charts bundle based on a catalog applications git repository
- dkp create cluster(see page 704) - Create a Kubernetes cluster, one of [aks, aws, azure, eks, gcp, preprovisioned, vsphere]
- dkp create image-bundle(see page 701) - Create a tar.gz image bundle
- dkp create nodepool(see page 721) - Create a nodepool, one of [aks, aws, azure, eks, gcp, preprovisioned, vsphere]
- dkp create workspace(see page 699) - Create a Workspace

9.2.6.4 dkp create workspace

Create a Workspace

```
dkp create workspace WORKSPACE_NAME [flags]
```

Options

```
      --config string            Config file to use (default "/root/.kommander/
config")
      --context string           The name of the kubeconfig context to use
      --dry-run                  Export in YAML format to stdout
  -h, --help                     help for workspace
      --kubeconfig string        Path to the kubeconfig file to use for CLI requests.
  -n, --namespace string         Name of the Namespace to create for the workspace
  -o, --output string            Output format. One of: yaml|json (default "yaml")
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create(see page 698) - Create one of [appdeployment, bootstrap, capi-components, chart-bundle, cluster, image-bundle, nodepool, workspace]

9.2.6.5 dkp create appdeployment

Create an AppDeployment

Synopsis

Creates an AppDeployment in a workspace or project.

When [--clusters C1,C2..] is not specified, it targets all existing clusters in the specified workspace or project.

```
dkp create appdeployment APPDEPLOYMENT_NAME --app NAME [flags]
```

Options

```
      --add-cluster-config-overrides strings   Comma-separated list of mappings of
kommanderCluster name to cluster configuration override ConfigMap name to apply to
the targeting clusters. (e.g., cluster-1:override-1-cm,cluster-2:override-2-cm)(only
valid for dkp clusters version 2.3.x and up) (default [])
  -a, --app string                             Name of the App to deploy
      --clusters strings                       List of names of kommanderClusters to
select for the command. (e.g., cluster-1,cluster-2)(only valid for dkp clusters
version 2.3.x and up) (default [])
      --config string                          Config file to use (default "/
root/.kommander/config")
  -c, --config-overrides string                Name of the ConfigMap used to override
default configuration of the App
      --context string                         The name of the kubeconfig context to
use
      --dry-run                                Export in YAML format to stdout
  -h, --help                                   help for appdeployment
      --kubeconfig string                      Path to the kubeconfig file to use for
 CLI requests.
  -o, --output string                          Output format. One of: yaml|json
(default "yaml")
  -p, --project string                         Name of the project to create the
AppDeployment in. Requires workspace flag (workspace that the project belongs to).
      --request-timeout string                 The length of time to wait before
giving up on a single server request. Non-zero values should contain a corresponding
time unit (e.g. 1s, 2m, 3h). A value of zero means don't timeout requests.
  -w, --workspace string                       Name of the workspace to create the
AppDeployment in
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create(see page 698) - Create one of [appdeployment, bootstrap, capi-components, chart-bundle, cluster, image-bundle, nodepool, workspace]

9.2.6.6 dkp create capi-components

Create the CAPI components in the cluster

```
dkp create capi-components [flags]
```

Options

```
      --aws-service-endpoints string     Custom AWS service endpoints in a semi-colon
separated format: ${SigningRegion1}:${ServiceID1}=${URL},${ServiceID2}=${URL};$
{SigningRegion2}...
  -h, --help                             help for capi-components
      --http-proxy string                HTTP proxy for CAPI controllers
      --https-proxy string               HTTPS proxy for CAPI controllers
      --kubeconfig string                Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
      --no-proxy strings                 No Proxy list for CAPI controllers (default
 [])
      --timeout duration                 The length of time to wait before giving up.
Zero means wait forever. (default 10m0s)
      --wait                             If true, wait for operations to complete
before returning. (default true)
      --with-aws-bootstrap-credentials   Set true to use AWS bootstrap credentials
from your environment. When false, the instance profile of the EC2 instance where the
CAPA controller is scheduled on will be used instead.
      --with-gcp-bootstrap-credentials   Set true to use GCP bootstrap credentials
from your environment. When false, the service account of the VM instance where the
CAPG controller is scheduled on will be used instead.
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create(see page 698) - Create one of [appdeployment, bootstrap, capi-components, chart-bundle, cluster, image-bundle, nodepool, workspace]

9.2.6.7 dkp create image-bundle

Create a tar.gz image bundle

```
dkp create image-bundle [flags]
```

Options

```
  -h, --help                     help for image-bundle
      --images-file string       File containing list of images to create bundle
from, either as YAML configuration or a simple list of images
```

```
      --output-file string       Output file to write image bundle to (default
"images.tar.gz")
      --overwrite                Overwrite image bundle file if it already exists
      --platform platformSlice   platforms to download images (required format: <os>/
<arch>[/<variant>]) (default [linux/amd64])
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create(see page 698) - Create one of [appdeployment, bootstrap, capi-components, chart-bundle, cluster, image-bundle, nodepool, workspace]

9.2.6.8 dkp create bootstrap

Create bootstrap cluster

```
dkp create bootstrap [flags]
```

Options

```
      --aws-service-endpoints string     Custom AWS service endpoints in a semi-colon
separated format: ${SigningRegion1}:${ServiceID1}=${URL},${ServiceID2}=${URL};$
{SigningRegion2}...
  -h, --help                             help for bootstrap
      --http-proxy string                HTTP proxy for CAPI controllers
      --https-proxy string               HTTPS proxy for CAPI controllers
      --kind-cluster-image string        Kind node image for the bootstrap cluster (d
efault "mesosphere/konvoy-bootstrap:v2.3.3")
      --kubeconfig string                Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
      --no-proxy strings                 No Proxy list for CAPI controllers (default
 [])
      --timeout duration                 The length of time to wait before giving up.
Zero means wait forever. (default 10m0s)
      --wait                             If true, wait for operations to complete
before returning. (default true)
      --with-aws-bootstrap-credentials   Set true to use AWS bootstrap credentials
from your environment. When false, the instance profile of the EC2 instance where the
CAPA controller is scheduled on will be used instead.
      --with-gcp-bootstrap-credentials   Set true to use GCP bootstrap credentials
from your environment. When false, the service account of the VM instance where the
CAPG controller is scheduled on will be used instead.
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create(see page 698) - Create one of [appdeployment, bootstrap, capi-components, chart-bundle, cluster, image-bundle, nodepool, workspace]

9.2.6.9 dkp create chart-bundle

Create charts bundle based on a catalog applications git repository

```
dkp create chart-bundle [flags]
```

Options

```
      --catalog-repository string         Git repository containing catalog
application definitions
      --config string                     Config file to use (default "/
root/.kommander/config")
      --context string                    The name of the kubeconfig context to use
      --dry-run                           Export in YAML format to stdout
      --extra-charts strings              Extra charts to include in the bundle, in
the format <repo-url1>|<chart-name1>,<repo-url2>|<chart-name2>:<chart-version2>,...
(default [])
  -h, --help                              help for chart-bundle
      --kommander-charts-version string   Kommander helm charts version to download.
Default: download all available versions
      --kubeconfig string                 Path to the kubeconfig file to use for CLI
requests.
  -o, --output string                     File path to write charts bundle to
(default "charts-bundle.tar.gz")
      --request-timeout string            The length of time to wait before giving up
on a single server request. Non-zero values should contain a corresponding time unit
(e.g. 1s, 2m, 3h). A value of zero means don't timeout requests.
      --skip-charts strings               Charts to not to include in the bundle, in
the format <chart-name1>,<chart-name2>:<chart-version2>,... (default [])
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create(see page 698) - Create one of [appdeployment, bootstrap, capi-components, chart-bundle, cluster, image-bundle, nodepool, workspace]

9.2.6.10 dkp create cluster

Create a Kubernetes cluster, one of [aks, aws, azure, eks, gcp, preprovisioned, vsphere]

Options

```
  -h, --help   help for cluster
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create(see page 698) - Create one of [appdeployment, bootstrap, capi-components, chart-bundle, cluster, image-bundle, nodepool, workspace]
- dkp create cluster aks(see page 714) - Create a Konvoy cluster in AKS
- dkp create cluster aws(see page 706) - Create a Konvoy cluster in AWS
- dkp create cluster azure(see page 709) - Create a Konvoy cluster in Azure
- dkp create cluster eks(see page 704) - Create a Konvoy cluster in EKS
- dkp create cluster gcp(see page 719) - Create a Konvoy cluster in GCP
- dkp create cluster preprovisioned(see page 716) - Create a Konvoy cluster on pre-provisioned infrastructure
- dkp create cluster vsphere(see page 711) - Create a Konvoy cluster in vSphere

dkp create cluster eks

Create a Konvoy cluster in EKS

```
dkp create cluster eks [flags]
```

Options

```
      --additional-security-group-ids strings   A comma separated list of existing
security group IDs to use for machines in addition to those created automatically (de
fault [])
      --additional-tags stringToString          Tags to apply to the provisioned
infrastructure (default [])
```

```
      --allow-missing-template-keys             If true, ignore any errors in
templates when a field or map key is missing in the template. Only applies to golang
and jsonpath output formats. (default true)
      --aws-service-endpoints string            Custom AWS service endpoints in a
semi-colon separated format: ${SigningRegion1}:${ServiceID1}=${URL},${ServiceID2}=$
{URL};${SigningRegion2}...
  -c, --cluster-name name                       Name used to prefix the cluster and
all the created resources.
      --dry-run                                 Only print the objects that would be
created, without creating them.
      --etcd-image-repository string            The image repository to use for
 pulling the etcd image
      --etcd-version string                     The version of etcd to use.
Overriding kubeadm's default value as etcd v3.5.x is not recommended for production
use. This default value will removed in a future release once etcd is fixed. (default
 "3.4.13-0")
  -h, --help                                    help for eks
      --http-proxy string                       HTTP proxy for CAPI controllers
      --https-proxy string                      HTTPS proxy for CAPI controllers
      --kind-cluster-image string               Kind node image for the bootstrap
cluster (default "mesosphere/konvoy-bootstrap:v2.3.3")
      --kubeconfig string                       Path to the kubeconfig for the
management cluster. If unspecified, default discovery rules apply.
      --kubernetes-image-repository string      The image repository to use for
 pulling kubernetes images
      --kubernetes-version string               Kubernetes version (default "1.22.6")
  -n, --namespace string                        If present, the namespace scope for
this CLI request. (default "default")
      --no-proxy strings                        No Proxy list for CAPI controllers (d
efault [])
  -o, --output string                           Output format. One of: json|yaml|
name|go-template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|
jsonpath-file.
      --region string                           AWS region to deploy cluster to
(default "us-west-2")
      --show-managed-fields                     If true, keep the managedFields when
printing objects in JSON or YAML format.
      --subnet-ids strings                      A comma separated list of existing
subnet IDs to use for the kube-apiserver ELB and all control-plane and worker nodes (d
efault [])
      --template string                         Template string or path to template
file to use when -o=go-template, -o=go-template-file. The template format is golang
templates [http://golang.org/pkg/text/template/#pkg-overview].
      --timeout duration                        The length of time to wait before
giving up. Zero means wait forever. (default 10m0s)
      --vpc-id string                           Existing VPC ID to use for the
cluster
      --wait                                    If true, wait for operations to
complete before returning. (default true)
      --with-aws-bootstrap-credentials          Set true to use AWS bootstrap
credentials from your environment. When false, the instance profile of the EC2
instance where the CAPA controller is scheduled on will be used instead.
```

```
      --with-gcp-bootstrap-credentials          Set true to use GCP bootstrap
credentials from your environment. When false, the service account of the VM instance
where the CAPG controller is scheduled on will be used instead.
      --worker-availability-zone string         The AvailabilityZone in the region to
deploy the worker nodes to, if not set a random one will be selected (ex. us-west-2a)
      --worker-iam-instance-profile string      Name of the IAM instance profile to
assign to worker machines. (default "nodes.cluster-api-provider-aws.sigs.k8s.io")
      --worker-instance-type string             Worker machine instance type (default
 "m5.2xlarge")
      --worker-replicas int                     Number of workers (default 4)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create cluster(see page 704) - Create a Kubernetes cluster, one of [aks, aws, azure, eks, gcp, preprovisioned, vsphere]

dkp create cluster aws

Create a Konvoy cluster in AWS

```
dkp create cluster aws [flags]
```

Options

```
      --additional-security-group-ids strings       A comma separated list of
existing security group IDs to use for machines in addition to those created
automatically (default [])
      --additional-tags stringToString              Tags to apply to the provisioned
infrastructure (default [])
      --allow-missing-template-keys                 If true, ignore any errors in
templates when a field or map key is missing in the template. Only applies to golang
and jsonpath output formats. (default true)
      --ami string                                  AMI ID to use for machines
      --ami-base-os string                          Base OS for Lookup search (ex.
'centos-7', 'ubuntu-18.04', 'ubuntu-20.04') (default "ubuntu-20.04")
      --ami-format string                           Lookup Format string to generate
AMI search name from (default "capa-ami-{{.BaseOS}}-?{{.K8sVersion}}-*")
      --ami-owner string                            Owner ID for AMI Lookup search (d
efault "258751437250")
      --aws-service-endpoints string                Custom AWS service endpoints in a
semi-colon separated format: ${SigningRegion1}:${ServiceID1}=${URL},${ServiceID2}=$
{URL};${SigningRegion2}...
```

```
      --certificate-renew-interval int              The interval number of days
Kubernetes managed PKI certificates are renewed. For example, an Interval value of 30
 means the certificates will be refreshed every 30 days. A value of 0 disables the
feature. (default 0)
  -c, --cluster-name name                           Name used to prefix the cluster
and all the created resources.
      --control-plane-http-proxy string             HTTP proxy for control plane
machines
      --control-plane-https-proxy string            HTTPS proxy for control plane
machines
      --control-plane-iam-instance-profile string   Name of the IAM instance profile
to assign to control plane machines. (default "control-plane.cluster-api-provider-
aws.sigs.k8s.io")
      --control-plane-instance-type string          Control Plane machine instance
type (default "m5.xlarge")
      --control-plane-no-proxy strings              No Proxy list for control plane
machines (default [])
      --control-plane-replicas int                  Number of control plane replicas
(default 3)
      --dry-run                                     Only print the objects that would
be created, without creating them.
      --etcd-image-repository string                The image repository to use for
 pulling the etcd image
      --etcd-version string                         The version of etcd to use.
Overriding kubeadm's default value as etcd v3.5.x is not recommended for production
use. This default value will removed in a future release once etcd is fixed. (default
 "3.4.13-0")
      --extra-sans strings                          A comma separated list of
additional Subject Alternative Names for the API Server signing cert (default [])
  -h, --help                                        help for aws
      --http-proxy string                           HTTP proxy for CAPI controllers
      --https-proxy string                          HTTPS proxy for CAPI controllers
      --internal-load-balancer                      Make the control plane load
balancer internal, i.e., reachable only within the VPC.
      --kind-cluster-image string                   Kind node image for the bootstrap
cluster (default "mesosphere/konvoy-bootstrap:v2.3.3")
      --kubeconfig string                           Path to the kubeconfig for the
management cluster. If unspecified, default discovery rules apply.
      --kubernetes-image-repository string          The image repository to use for
 pulling kubernetes images
      --kubernetes-version string                   Kubernetes version (default
"1.23.12")
  -n, --namespace string                            If present, the namespace scope
for this CLI request. (default "default")
      --no-proxy strings                            No Proxy list for CAPI
controllers (default [])
  -o, --output string                               Output format. One of: json|yaml|
name|go-template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|
jsonpath-file.
      --region string                               AWS region to deploy cluster to
(default "us-west-2")
```

```
      --registry-mirror-cacert file                 CA certificate chain to use while
 communicating with the registry mirror using TLS
      --registry-mirror-password string             Password to authenticate to the
registry mirror with
      --registry-mirror-url string                  URL of a container registry to
use as a mirror in the cluster
      --registry-mirror-username string             Username to authenticate to the
registry mirror with
      --self-managed                                When set to true, the required
prerequisites are created before creating the cluster and the resulting cluster has
all necessary components deployed onto itself, so it can manage its own cluster
lifecycle. When set to false, a management cluster is used.(default false)
      --show-managed-fields                         If true, keep the managedFields
when printing objects in JSON or YAML format.
      --ssh-public-key-file string                  Path to the authorized SSH key
for the user
      --ssh-username string                         Name of the user to create on the
instance (default "konvoy")
      --subnet-ids strings                          A comma separated list of
existing subnet IDs to use for the kube-apiserver ELB and all control-plane and
worker nodes (default [])
      --template string                             Template string or path to
template file to use when -o=go-template, -o=go-template-file. The template format is
golang templates [http://golang.org/pkg/text/template/#pkg-overview].
      --timeout duration                            The length of time to wait before
giving up. Zero means wait forever. (default 10m0s)
      --vpc-id string                               Existing VPC ID to use for the
cluster
      --wait                                        If true, wait for operations to
complete before returning. (default true)
      --with-aws-bootstrap-credentials              Set true to use AWS bootstrap
credentials from your environment. When false, the instance profile of the EC2
instance where the CAPA controller is scheduled on will be used instead.
      --with-gcp-bootstrap-credentials              Set true to use GCP bootstrap
credentials from your environment. When false, the service account of the VM instance
where the CAPG controller is scheduled on will be used instead.
      --worker-availability-zone string             The AvailabilityZone in the
region to deploy the worker nodes to, if not set a random one will be selected (ex.
us-west-2a)
      --worker-http-proxy string                    HTTP proxy for nodes
      --worker-https-proxy string                   HTTPS proxy for nodes
      --worker-iam-instance-profile string          Name of the IAM instance profile
to assign to worker machines. (default "nodes.cluster-api-provider-aws.sigs.k8s.io")
      --worker-instance-type string                 Worker machine instance type
(default "m5.2xlarge")
      --worker-no-proxy strings                     No Proxy list for nodes (default
 [])
      --worker-replicas int                         Number of workers (default 4)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create cluster(see page 704) - Create a Kubernetes cluster, one of [aks, aws, azure, eks, gcp, preprovisioned, vsphere]

dkp create cluster azure

Create a Konvoy cluster in Azure

```
dkp create cluster azure [flags]
```

Options

```
      --additional-tags stringToString       Tags to apply to the provisioned
infrastructure (default [])
      --allow-missing-template-keys          If true, ignore any errors in templates
when a field or map key is missing in the template. Only applies to golang and
jsonpath output formats. (default true)
      --aws-service-endpoints string         Custom AWS service endpoints in a semi-
colon separated format: ${SigningRegion1}:${ServiceID1}=${URL},${ServiceID2}=${URL};$
{SigningRegion2}...
      --certificate-renew-interval int       The interval number of days Kubernetes
managed PKI certificates are renewed. For example, an Interval value of 30 means the
certificates will be refreshed every 30 days. A value of 0 disables the feature. (def
ault 0)
  -c, --cluster-name name                    Name used to prefix the cluster and all
the created resources.
      --control-plane-http-proxy string      HTTP proxy for control plane machines
      --control-plane-https-proxy string     HTTPS proxy for control plane machines
      --control-plane-machine-size string    Control Plane machine size (default
"Standard_D4s_v3")
      --control-plane-no-proxy strings       No Proxy list for control plane machines
(default [])
      --control-plane-replicas int           Number of control plane replicas (defaul
t 3)
      --dry-run                              Only print the objects that would be
created, without creating them.
      --etcd-image-repository string         The image repository to use for pulling
the etcd image
      --etcd-version string                  The version of etcd to use. Overriding
kubeadm's default value as etcd v3.5.x is not recommended for production use. This
```

```
default value will removed in a future release once etcd is fixed. (default
"3.4.13-0")
      --extra-sans strings                  A comma separated list of additional
Subject Alternative Names for the API Server signing cert (default [])
  -h, --help                                help for azure
      --http-proxy string                   HTTP proxy for CAPI controllers
      --https-proxy string                  HTTPS proxy for CAPI controllers
      --kind-cluster-image string           Kind node image for the bootstrap
cluster (default "mesosphere/konvoy-bootstrap:v2.3.3")
      --kubeconfig string                   Path to the kubeconfig for the
management cluster. If unspecified, default discovery rules apply.
      --kubernetes-image-repository string  The image repository to use for pulling
kubernetes images
      --kubernetes-version string           Kubernetes version (default "1.23.12")
      --location string                     Azure location to deploy cluster to
(default "westus")
  -n, --namespace string                    If present, the namespace scope for this
 CLI request. (default "default")
      --no-proxy strings                    No Proxy list for CAPI controllers (defa
ult [])
  -o, --output string                       Output format. One of: json|yaml|name|
go-template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|
jsonpath-file.
      --registry-mirror-cacert file         CA certificate chain to use while
 communicating with the registry mirror using TLS
      --registry-mirror-password string     Password to authenticate to the registry
mirror with
      --registry-mirror-url string          URL of a container registry to use as a
mirror in the cluster
      --registry-mirror-username string     Username to authenticate to the registry
mirror with
      --self-managed                        When set to true, the required
prerequisites are created before creating the cluster and the resulting cluster has
all necessary components deployed onto itself, so it can manage its own cluster
lifecycle. When set to false, a management cluster is used.(default false)
      --show-managed-fields                 If true, keep the managedFields when
printing objects in JSON or YAML format.
      --ssh-public-key-file string          Path to the authorized SSH key for the
user
      --ssh-username string                 Name of the user to create on the
instance (default "konvoy")
      --template string                     Template string or path to template file
to use when -o=go-template, -o=go-template-file. The template format is golang
templates [http://golang.org/pkg/text/template/#pkg-overview].
      --timeout duration                    The length of time to wait before giving
up. Zero means wait forever. (default 10m0s)
      --wait                                If true, wait for operations to complete
before returning. (default true)
      --with-aws-bootstrap-credentials      Set true to use AWS bootstrap
credentials from your environment. When false, the instance profile of the EC2
instance where the CAPA controller is scheduled on will be used instead.
```

```
      --with-gcp-bootstrap-credentials       Set true to use GCP bootstrap
credentials from your environment. When false, the service account of the VM instance
where the CAPG controller is scheduled on will be used instead.
      --worker-availability-zone string      The availability zone in the region to
deploy the worker nodes to, if not set a random one will be selected (ex. 1). Not all
locations, including the default 'westus', support setting this flag, see https://
docs.microsoft.com/en-us/azure/availability-zones/az-overview.
      --worker-http-proxy string             HTTP proxy for nodes
      --worker-https-proxy string            HTTPS proxy for nodes
      --worker-machine-size string           Worker machine size (default
"Standard_D8s_v3")
      --worker-no-proxy strings              No Proxy list for nodes (default [])
      --worker-replicas int                  Number of workers (default 4)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create cluster(see page 704) - Create a Kubernetes cluster, one of [aks, aws, azure, eks, gcp, preprovisioned, vsphere]

dkp create cluster vsphere

Create a Konvoy cluster in vSphere

```
dkp create cluster vsphere [flags]
```

Options

```
      --additional-tags stringToString       Tags to apply to the provisioned
infrastructure (default [])
      --allow-missing-template-keys          If true, ignore any errors in templates
when a field or map key is missing in the template. Only applies to golang and
jsonpath output formats. (default true)
      --aws-service-endpoints string         Custom AWS service endpoints in a semi-
colon separated format: ${SigningRegion1}:${ServiceID1}=${URL},${ServiceID2}=${URL};$
{SigningRegion2}...
      --certificate-renew-interval int       The interval number of days Kubernetes
managed PKI certificates are renewed. For example, an Interval value of 30 means the
certificates will be refreshed every 30 days. A value of 0 disables the feature. (def
ault 0)
  -c, --cluster-name name                    Name used to prefix the cluster and all
the created resources.
```

```
      --control-plane-cpus int                The number of virtual processors in a
control plane machine (default 4)
      --control-plane-disk-size int           The size of a control plane machine's
disk, in GB (default 80)
      --control-plane-endpoint-host string    The control plane endpoint address. To
use an external load balancer, set to its IP or hostname. To use the built-in virtual
IP, set to a static IPv4 address in the Layer 2 network of the control plane
machines. [Not for production use: To use a single-machine control plane, set to the
IP or hostname of the machine.]
      --control-plane-endpoint-port int       The control plane endpoint port. To use
an external load balancer, set to its listening port. (default 6443)
      --control-plane-http-proxy string       HTTP proxy for control plane machines
      --control-plane-https-proxy string      HTTPS proxy for control plane machines
      --control-plane-memory int              The size of a control plane machine's
memory, in GB (default 16)
      --control-plane-no-proxy strings        No Proxy list for control plane machines
(default [])
      --control-plane-replicas int            Number of control plane replicas (defaul
t 3)
      --data-center string                    The vSphere datacenter to deploy the
workload cluster on.
      --data-store string                     The vSphere datastore to deploy the
workload cluster on.
      --dry-run                               Only print the objects that would be
created, without creating them.
      --etcd-image-repository string          The image repository to use for pulling
the etcd image
      --etcd-version string                   The version of etcd to use. Overriding
kubeadm's default value as etcd v3.5.x is not recommended for production use. This
default value will removed in a future release once etcd is fixed. (default
"3.4.13-0")
      --extra-sans strings                    A comma separated list of additional
Subject Alternative Names for the API Server signing cert (default [])
      --folder string                         The vSphere folder for your VMs. Set to
"" to use the root vSphere folder.
  -h, --help                                  help for vsphere
      --http-proxy string                     HTTP proxy for CAPI controllers
      --https-proxy string                    HTTPS proxy for CAPI controllers
      --kind-cluster-image string             Kind node image for the bootstrap
cluster (default "mesosphere/konvoy-bootstrap:v2.3.3")
      --kubeconfig string                     Path to the kubeconfig for the
management cluster. If unspecified, default discovery rules apply.
      --kubernetes-image-repository string    The image repository to use for pulling
kubernetes images
      --kubernetes-version string             Kubernetes version (default "1.23.12")
  -n, --namespace string                      If present, the namespace scope for this
 CLI request. (default "default")
      --network string                        The vSphere network to deploy the
workload cluster on.
      --no-proxy strings                      No Proxy list for CAPI controllers (defa
ult [])
```

```
  -o, --output string                          Output format. One of: json|yaml|name|
go-template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|
jsonpath-file.
      --registry-mirror-cacert file            CA certificate chain to use while
 communicating with the registry mirror using TLS
      --registry-mirror-password string        Password to authenticate to the registry
mirror with
      --registry-mirror-url string             URL of a container registry to use as a
mirror in the cluster
      --registry-mirror-username string        Username to authenticate to the registry
mirror with
      --resource-pool string                   The vSphere resource pool for the
workload cluster's virtual machines.
      --self-managed                           When set to true, the required
prerequisites are created before creating the cluster and the resulting cluster has
all necessary components deployed onto itself, so it can manage its own cluster
lifecycle. When set to false, a management cluster is used.(default false)
      --server string                          The vCenter server IP or FQDN.
      --show-managed-fields                    If true, keep the managedFields when
printing objects in JSON or YAML format.
      --ssh-public-key-file string             Path to the authorized SSH key for the
user
      --ssh-username string                    Name of the user to create on the
instance (default "konvoy")
      --storage-policy string                  This is the vSphere storage policy. Set
it to "" if you don't want to use a storage policy.
      --template string                        Template string or path to template file
to use when -o=go-template, -o=go-template-file. The template format is golang
templates [http://golang.org/pkg/text/template/#pkg-overview].
      --timeout duration                       The length of time to wait before giving
up. Zero means wait forever. (default 10m0s)
      --tls-thumb-print string                 sha1 thumbprint of the vcenter
certificate: openssl x509 -sha1 -fingerprint -in ca.crt -noout
      --virtual-ip-interface string            The network interface, e.g, 'eth0' or
'ens5', to use for the built-in virtual IP control plane endpoint. This interface
 must be available on every control plane machine. If the value is empty, the flag
does nothing. If the value is not empty, the built-in virtual IP control plane
endpoint is created, using values from --control-plane-endpoint-host and --control-
plane-endpoint-port.
      --vm-template string                     The virtual machine template to use for
 the workload cluster's virtual machines.
      --wait                                   If true, wait for operations to complete
before returning. (default true)
      --with-aws-bootstrap-credentials         Set true to use AWS bootstrap
credentials from your environment. When false, the instance profile of the EC2
instance where the CAPA controller is scheduled on will be used instead.
      --with-gcp-bootstrap-credentials         Set true to use GCP bootstrap
credentials from your environment. When false, the service account of the VM instance
where the CAPG controller is scheduled on will be used instead.
      --worker-cpus int                        The number of virtual processors in a
worker machine (default 8)
```

```
      --worker-disk-size int                The size of a worker machine's disk, in
GB (default 80)
      --worker-http-proxy string            HTTP proxy for nodes
      --worker-https-proxy string           HTTPS proxy for nodes
      --worker-memory int                   The size of a worker machine's memory,
in GB (default 32)
      --worker-no-proxy strings             No Proxy list for nodes (default [])
      --worker-replicas int                 Number of workers (default 4)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create cluster(see page 704) - Create a Kubernetes cluster, one of [aks, aws, azure, eks, gcp, preprovisioned, vsphere]

dkp create cluster aks

Create a Konvoy cluster in AKS

```
dkp create cluster aks [flags]
```

Options

```
      --additional-tags stringToString      Tags to apply to the provisioned
infrastructure (default [])
      --allow-missing-template-keys         If true, ignore any errors in templates
when a field or map key is missing in the template. Only applies to golang and
jsonpath output formats. (default true)
      --aws-service-endpoints string        Custom AWS service endpoints in a semi-
colon separated format: ${SigningRegion1}:${ServiceID1}=${URL},${ServiceID2}=${URL};$
{SigningRegion2}...
      --certificate-renew-interval int      The interval number of days Kubernetes
managed PKI certificates are renewed. For example, an Interval value of 30 means the
certificates will be refreshed every 30 days. A value of 0 disables the feature. (def
ault 0)
  -c, --cluster-name name                   Name used to prefix the cluster and all
the created resources.
      --control-plane-http-proxy string     HTTP proxy for control plane machines
      --control-plane-https-proxy string    HTTPS proxy for control plane machines
      --control-plane-machine-size string   Control Plane machine size (default
"Standard_D4s_v3")
      --control-plane-no-proxy strings      No Proxy list for control plane machines
(default [])
```

```
      --control-plane-replicas int           Number of control plane replicas (defaul
t 3)
      --dry-run                              Only print the objects that would be
created, without creating them.
      --etcd-image-repository string         The image repository to use for pulling
the etcd image
      --etcd-version string                  The version of etcd to use. Overriding
kubeadm's default value as etcd v3.5.x is not recommended for production use. This
default value will removed in a future release once etcd is fixed. (default
"3.4.13-0")
      --extra-sans strings                   A comma separated list of additional
Subject Alternative Names for the API Server signing cert (default [])
  -h, --help                                 help for aks
      --http-proxy string                    HTTP proxy for CAPI controllers
      --https-proxy string                   HTTPS proxy for CAPI controllers
      --kind-cluster-image string            Kind node image for the bootstrap
cluster (default "mesosphere/konvoy-bootstrap:v2.3.3")
      --kubeconfig string                    Path to the kubeconfig for the
management cluster. If unspecified, default discovery rules apply.
      --kubernetes-image-repository string   The image repository to use for pulling
kubernetes images
      --kubernetes-version string            Kubernetes version. Run 'az aks get-
versions -o table --location <location>' to see available versions. See https://
docs.microsoft.com/en-us/azure/aks/supported-kubernetes-versions for more details.
Must be a patch version for v1.23.x.
      --location string                      Azure location to deploy cluster to
(default "westus")
  -n, --namespace string                     If present, the namespace scope for this
 CLI request. (default "default")
      --no-proxy strings                     No Proxy list for CAPI controllers (defa
ult [])
  -o, --output string                        Output format. One of: json|yaml|name|
go-template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|
jsonpath-file.
      --registry-mirror-cacert file          CA certificate chain to use while
 communicating with the registry mirror using TLS
      --registry-mirror-password string      Password to authenticate to the registry
mirror with
      --registry-mirror-url string           URL of a container registry to use as a
mirror in the cluster
      --registry-mirror-username string      Username to authenticate to the registry
mirror with
      --show-managed-fields                  If true, keep the managedFields when
printing objects in JSON or YAML format.
      --ssh-public-key-file string           Path to the authorized SSH key for the
user
      --ssh-username string                  Name of the user to create on the
instance (default "konvoy")
      --template string                      Template string or path to template file
to use when -o=go-template, -o=go-template-file. The template format is golang
templates [http://golang.org/pkg/text/template/#pkg-overview].
```

```
      --timeout duration                      The length of time to wait before giving
up. Zero means wait forever. (default 10m0s)
      --wait                                  If true, wait for operations to complete
before returning. (default true)
      --with-aws-bootstrap-credentials        Set true to use AWS bootstrap
credentials from your environment. When false, the instance profile of the EC2
instance where the CAPA controller is scheduled on will be used instead.
      --with-gcp-bootstrap-credentials        Set true to use GCP bootstrap
credentials from your environment. When false, the service account of the VM instance
where the CAPG controller is scheduled on will be used instead.
      --worker-http-proxy string              HTTP proxy for nodes
      --worker-https-proxy string             HTTPS proxy for nodes
      --worker-machine-size string           Worker machine size (default
"Standard_D8s_v3")
      --worker-no-proxy strings               No Proxy list for nodes (default [])
      --worker-replicas int                   Number of workers (default 4)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create cluster(see page 704) - Create a Kubernetes cluster, one of [aks, aws, azure, eks, gcp, preprovisioned, vsphere]

dkp create cluster preprovisioned

Create a Konvoy cluster on pre-provisioned infrastructure

```
dkp create cluster preprovisioned [flags]
```

Options

```
      --additional-tags stringToString          Tags to apply to the provisioned
infrastructure (default [])
      --allow-missing-template-keys             If true, ignore any errors in
templates when a field or map key is missing in the template. Only applies to golang
and jsonpath output formats. (default true)
      --aws-service-endpoints string            Custom AWS service endpoints in a
semi-colon separated format: ${SigningRegion1}:${ServiceID1}=${URL},${ServiceID2}=$
{URL};${SigningRegion2}...
      --certificate-renew-interval int          The interval number of days
Kubernetes managed PKI certificates are renewed. For example, an Interval value of 30
 means the certificates will be refreshed every 30 days. A value of 0 disables the
feature. (default 0)
```

```
  -c, --cluster-name name                         Name used to prefix the cluster and
all the created resources.
      --control-plane-endpoint-host string        The control plane endpoint address.
To use an external load balancer, set to its IP or hostname. To use the built-in
virtual IP, set to a static IPv4 address in the Layer 2 network of the control plane
machines. [Not for production use: To use a single-machine control plane, set to the
IP or hostname of the machine.]
      --control-plane-endpoint-port int           The control plane endpoint port. To
use an external load balancer, set to its listening port. (default 6443)
      --control-plane-http-proxy string           HTTP proxy for control plane machines
      --control-plane-https-proxy string          HTTPS proxy for control plane
machines
      --control-plane-no-proxy strings            No Proxy list for control plane
machines (default [])
      --control-plane-replicas int                Number of control plane replicas (def
ault 3)
      --dry-run                                   Only print the objects that would be
created, without creating them.
      --etcd-image-repository string              The image repository to use for
 pulling the etcd image
      --etcd-version string                       The version of etcd to use.
Overriding kubeadm's default value as etcd v3.5.x is not recommended for production
use. This default value will removed in a future release once etcd is fixed. (default
 "3.4.13-0")
      --extra-sans strings                        A comma separated list of additional
Subject Alternative Names for the API Server signing cert (default [])
  -h, --help                                      help for preprovisioned
      --http-proxy string                         HTTP proxy for CAPI controllers
      --https-proxy string                        HTTPS proxy for CAPI controllers
      --kind-cluster-image string                 Kind node image for the bootstrap
cluster (default "mesosphere/konvoy-bootstrap:v2.3.3")
      --kubeconfig string                         Path to the kubeconfig for the
management cluster. If unspecified, default discovery rules apply.
      --kubernetes-image-repository string        The image repository to use for
 pulling kubernetes images
      --kubernetes-version string                 Kubernetes version (default "1.23.12")
  -n, --namespace string                          If present, the namespace scope for
this CLI request. (default "default")
      --no-proxy strings                          No Proxy list for CAPI controllers (d
efault [])
      --os-hint flatcar                           A hint which will allow the installer
to generate appropriate configurations for a target OS. Presently, only the hint for
 flatcar is supported.
  -o, --output string                             Output format. One of: json|yaml|
name|go-template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|
jsonpath-file.
      --override-secret-name string               Name of the secret for any provided
overrides on a preprovisioned cluster.All overrides defined at provisioning should be
present in this secret.
      --pre-provisioned-inventory-file string     Path to PreprovisionedInventory
inventory file
```

```
      --registry-mirror-cacert file             CA certificate chain to use while
 communicating with the registry mirror using TLS
      --registry-mirror-password string         Password to authenticate to the
registry mirror with
      --registry-mirror-url string              URL of a container registry to use as
a mirror in the cluster
      --registry-mirror-username string         Username to authenticate to the
registry mirror with
      --self-managed                            When set to true, the required
prerequisites are created before creating the cluster and the resulting cluster has
all necessary components deployed onto itself, so it can manage its own cluster
lifecycle. When set to false, a management cluster is used.(default false)
      --show-managed-fields                     If true, keep the managedFields when
printing objects in JSON or YAML format.
      --ssh-private-key-file string             Path to the private SSH key file used
by Konvoy to access the pre-provisioned hosts
      --ssh-public-key-file string              Path to the authorized SSH key for
 the user
      --ssh-username string                     Name of the user to create on the
instance (default "konvoy")
      --template string                         Template string or path to template
file to use when -o=go-template, -o=go-template-file. The template format is golang
templates [http://golang.org/pkg/text/template/#pkg-overview].
      --timeout duration                        The length of time to wait before
giving up. Zero means wait forever. (default 10m0s)
      --virtual-ip-interface string             The network interface, e.g, 'eth0' or
'ens5', to use for the built-in virtual IP control plane endpoint. This interface
 must be available on every control plane machine. If the value is empty, the flag
does nothing. If the value is not empty, the built-in virtual IP control plane
endpoint is created, using values from --control-plane-endpoint-host and --control-
plane-endpoint-port.
      --wait                                    If true, wait for operations to
complete before returning. (default true)
      --with-aws-bootstrap-credentials          Set true to use AWS bootstrap
credentials from your environment. When false, the instance profile of the EC2
instance where the CAPA controller is scheduled on will be used instead.
      --with-gcp-bootstrap-credentials          Set true to use GCP bootstrap
credentials from your environment. When false, the service account of the VM instance
where the CAPG controller is scheduled on will be used instead.
      --worker-http-proxy string                HTTP proxy for nodes
      --worker-https-proxy string               HTTPS proxy for nodes
      --worker-no-proxy strings                 No Proxy list for nodes (default [])
      --worker-replicas int                     Number of workers (default 4)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create cluster(see page 704) - Create a Kubernetes cluster, one of [aks, aws, azure, eks, gcp, preprovisioned, vsphere]

dkp create cluster gcp

Create a Konvoy cluster in GCP

```
dkp create cluster gcp [flags]
```

Options

```
      --additional-tags stringToString              Tags to apply to the provisioned
infrastructure (default [])
      --allow-missing-template-keys                 If true, ignore any errors in
templates when a field or map key is missing in the template. Only applies to golang
and jsonpath output formats. (default true)
      --associate-public-ip-address                 Associate a public IP for all
machines. When set to false the specified network must have Cloud NAT configured to
provide internet access. (default true)
      --aws-service-endpoints string                Custom AWS service endpoints in
a semi-colon separated format: ${SigningRegion1}:${ServiceID1}=${URL},${ServiceID2}=$
{URL};${SigningRegion2}...
      --certificate-renew-interval int              The interval number of days
Kubernetes managed PKI certificates are renewed. For example, an Interval value of 30
 means the certificates will be refreshed every 30 days. A value of 0 disables the
feature. (default 0)
  -c, --cluster-name name                           Name used to prefix the cluster
and all the created resources.
      --control-plane-http-proxy string             HTTP proxy for control plane
machines
      --control-plane-https-proxy string            HTTPS proxy for control plane
machines
      --control-plane-instance-type string          Control Plane machine instance
type (default "n2-standard-4")
      --control-plane-no-proxy strings              No Proxy list for control plane
machines (default [])
      --control-plane-replicas int                  Number of control plane replicas
(default 3)
      --control-plane-service-account-email string   Control Plane Service Account
email address (default "default")
      --dry-run                                     Only print the objects that
would be created, without creating them.
      --etcd-image-repository string                The image repository to use for
 pulling the etcd image
      --etcd-version string                         The version of etcd to use.
Overriding kubeadm's default value as etcd v3.5.x is not recommended for production
```

```
use. This default value will removed in a future release once etcd is fixed. (default
 "3.4.13-0")
      --extra-sans strings                          A comma separated list of
additional Subject Alternative Names for the API Server signing cert (default [])
  -h, --help                                        help for gcp
      --http-proxy string                           HTTP proxy for CAPI controllers
      --https-proxy string                          HTTPS proxy for CAPI controllers
      --image string                                Full reference to an image to
use for all nodes (set either this or --image-family) (ex. 'projects/my-project/
global/images/konvoy-ubuntu-2004-1-99-99-1234567890')
      --image-family string                         Full reference to an image
family to use for all nodes (set either this or --image) (ex. 'projects/my-project/
global/images/family/konvoy-ubuntu-2004-{{.K8sVersion}}')
      --kind-cluster-image string                   Kind node image for the
bootstrap cluster (default "mesosphere/konvoy-bootstrap:v2.3.3")
      --kubeconfig string                           Path to the kubeconfig for the
management cluster. If unspecified, default discovery rules apply.
      --kubernetes-image-repository string          The image repository to use for
 pulling kubernetes images
      --kubernetes-version string                   Kubernetes version (default
"1.23.12")
  -n, --namespace string                            If present, the namespace scope
for this CLI request. (default "default")
      --network string                              The GCP network name to deploy
the cluster to (default "default")
      --no-proxy strings                            No Proxy list for CAPI
controllers (default [])
  -o, --output string                               Output format. One of: json|
yaml|name|go-template|go-template-file|template|templatefile|jsonpath|jsonpath-as-
json|jsonpath-file.
      --project string                              The GCP project name to deploy
the cluster to
      --region string                               GCP region to deploy cluster to
(default "us-west1")
      --registry-mirror-cacert file                 CA certificate chain to use
while communicating with the registry mirror using TLS
      --registry-mirror-password string             Password to authenticate to the
registry mirror with
      --registry-mirror-url string                  URL of a container registry to
use as a mirror in the cluster
      --registry-mirror-username string             Username to authenticate to the
registry mirror with
      --self-managed                                When set to true, the required
prerequisites are created before creating the cluster and the resulting cluster has
all necessary components deployed onto itself, so it can manage its own cluster
lifecycle. When set to false, a management cluster is used.(default false)
      --show-managed-fields                         If true, keep the managedFields
when printing objects in JSON or YAML format.
      --ssh-public-key-file string                  Path to the authorized SSH key
for the user
      --ssh-username string                         Name of the user to create on
the instance (default "konvoy")
```

```
      --template string                              Template string or path to
template file to use when -o=go-template, -o=go-template-file. The template format is
golang templates [http://golang.org/pkg/text/template/#pkg-overview].
      --timeout duration                             The length of time to wait
before giving up. Zero means wait forever. (default 10m0s)
      --wait                                         If true, wait for operations to
complete before returning. (default true)
      --with-aws-bootstrap-credentials               Set true to use AWS bootstrap
credentials from your environment. When false, the instance profile of the EC2
instance where the CAPA controller is scheduled on will be used instead.
      --with-gcp-bootstrap-credentials               Set true to use GCP bootstrap
credentials from your environment. When false, the service account of the VM instance
where the CAPG controller is scheduled on will be used instead.
      --worker-http-proxy string                     HTTP proxy for nodes
      --worker-https-proxy string                    HTTPS proxy for nodes
      --worker-instance-type string                  Worker machine instance type
(default "n2-standard-8")
      --worker-no-proxy strings                      No Proxy list for nodes (default
 [])
      --worker-replicas int                          Number of workers (default 4)
      --worker-service-account-email string          Worker machine Service Account
email address (default "default")
      --worker-zone string                           Zone in the region to deploy the
worker nodes to, if not set a random one will be selected (ex. us-west1-a)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create cluster(see page 704) - Create a Kubernetes cluster, one of [aks, aws, azure, eks, gcp, preprovisioned, vsphere]

9.2.6.11 dkp create nodepool

Create a nodepool, one of [aks, aws, azure, eks, gcp, preprovisioned, vsphere]

Options

```
  -h, --help   help for nodepool
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create(see page 698) - Create one of [appdeployment, bootstrap, capi-components, chart-bundle, cluster, image-bundle, nodepool, workspace]
- dkp create nodepool aks(see page 725) - Create a nodepool in AKS
- dkp create nodepool aws(see page 726) - Create a nodepool in AWS
- dkp create nodepool azure(see page 722) - Create a nodepool in Azure
- dkp create nodepool eks(see page 728) - Create a nodepool for EKS
- dkp create nodepool gcp(see page 729) - Create a nodepool in GCP
- dkp create nodepool preprovisioned(see page 731) - Create a nodepool for a Pre Provisioned Cluster.
- dkp create nodepool vsphere(see page 723) - Create a nodepool in vSphere

dkp create nodepool azure

Create a nodepool in Azure

```
dkp create nodepool azure name [flags]
```

Options

```
      --additional-tags stringToString    Tags to apply to the provisioned
infrastructure (default [])
      --allow-missing-template-keys       If true, ignore any errors in templates
when a field or map key is missing in the template. Only applies to golang and
jsonpath output formats. (default true)
      --availability-zone string          The availability zone in the region to
deploy the worker nodes to, if not set a random one will be selected (ex. 1). Not all
locations, including the default 'westus', support setting this flag, see https://
docs.microsoft.com/en-us/azure/availability-zones/az-overview.
  -c, --cluster-name name                 Name used to prefix the cluster and all the
created resources.
      --dry-run                           Only print the objects that would be
created, without creating them.
  -h, --help                              help for azure
      --http-proxy string                 HTTP proxy for nodes
      --https-proxy string                HTTPS proxy for nodes
      --kubeconfig string                 Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
      --kubernetes-version string         Kubernetes version (default "1.23.12")
      --machine-size string               Worker machine size (default
"Standard_D8s_v3")
  -n, --namespace string                  If present, the namespace scope for this
 CLI request. (default "default")
      --no-proxy strings                  No Proxy list for nodes (default [])
  -o, --output string                     Output format. One of: json|yaml|name|go-
template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|jsonpath-
file.
```

```
      --registry-mirror-cacert file       CA certificate chain to use while
 communicating with the registry mirror using TLS
      --registry-mirror-password string   Password to authenticate to the registry
mirror with
      --registry-mirror-url string        URL of a container registry to use as a
mirror in the cluster
      --registry-mirror-username string   Username to authenticate to the registry
mirror with
      --replicas int                      Number of replicas (default 1)
      --show-managed-fields               If true, keep the managedFields when
printing objects in JSON or YAML format.
      --ssh-public-key-file string        Path to the authorized SSH key for the user
      --ssh-username string               Name of the user to create on the instance
(default "konvoy")
      --template string                   Template string or path to template file to
use when -o=go-template, -o=go-template-file. The template format is golang templates
[http://golang.org/pkg/text/template/#pkg-overview].
      --timeout duration                  The length of time to wait before giving
up. Zero means wait forever. (default 30m0s)
      --wait                              If true, wait for operations to complete
before returning.
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create nodepool(see page 721) - Create a nodepool, one of [aks, aws, azure, eks, gcp, preprovisioned, vsphere]

dkp create nodepool vsphere

Create a nodepool in vSphere

```
dkp create nodepool vsphere name [flags]
```

Options

```
      --additional-tags stringToString    Tags to apply to the provisioned
infrastructure (default [])
      --allow-missing-template-keys       If true, ignore any errors in templates
when a field or map key is missing in the template. Only applies to golang and
jsonpath output formats. (default true)
  -c, --cluster-name name                 Name used to prefix the cluster and all the
created resources.
```

```
      --cpus int                          The number of virtual processors in a
worker machine (default 8)
      --data-center string                The vSphere datacenter to deploy the
workload cluster on.
      --data-store string                 The vSphere datastore to deploy the
workload cluster on.
      --disk-size int                     The size of a worker machine's disk, in GB
(default 80)
      --dry-run                           Only print the objects that would be
created, without creating them.
      --folder string                     The vSphere folder for your VMs. Set to ""
 to use the root vSphere folder.
  -h, --help                              help for vsphere
      --http-proxy string                 HTTP proxy for nodes
      --https-proxy string                HTTPS proxy for nodes
      --kubeconfig string                 Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
      --kubernetes-version string         Kubernetes version (default "1.23.12")
      --memory int                        The size of a worker machine's memory, in
GB (default 32)
  -n, --namespace string                  If present, the namespace scope for this
 CLI request. (default "default")
      --network string                    The vSphere network to deploy the workload
cluster on.
      --no-proxy strings                  No Proxy list for nodes (default [])
  -o, --output string                     Output format. One of: json|yaml|name|go-
template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|jsonpath-
file.
      --registry-mirror-cacert file       CA certificate chain to use while
 communicating with the registry mirror using TLS
      --registry-mirror-password string   Password to authenticate to the registry
mirror with
      --registry-mirror-url string        URL of a container registry to use as a
mirror in the cluster
      --registry-mirror-username string   Username to authenticate to the registry
mirror with
      --replicas int                      Number of replicas (default 1)
      --resource-pool string              The vSphere resource pool for the workload
cluster's virtual machines.
      --server string                     The vCenter server IP or FQDN.
      --show-managed-fields               If true, keep the managedFields when
printing objects in JSON or YAML format.
      --ssh-public-key-file string        Path to the authorized SSH key for the user
      --ssh-username string               Name of the user to create on the instance
(default "konvoy")
      --storage-policy string             This is the vSphere storage policy. Set it
to "" if you don't want to use a storage policy.
      --template string                   Template string or path to template file to
use when -o=go-template, -o=go-template-file. The template format is golang templates
[http://golang.org/pkg/text/template/#pkg-overview].
      --timeout duration                  The length of time to wait before giving
up. Zero means wait forever. (default 30m0s)
```

```
      --tls-thumb-print string            sha1 thumbprint of the vcenter certificate:
openssl x509 -sha1 -fingerprint -in ca.crt -noout
      --vm-template string               The virtual machine template to use for the
workload cluster's virtual machines.
      --wait                             If true, wait for operations to complete
before returning.
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create nodepool(see page 721) - Create a nodepool, one of [aks, aws, azure, eks, gcp, preprovisioned, vsphere]

dkp create nodepool aks

Create a nodepool in AKS

```
dkp create nodepool aks name [flags]
```

Options

```
      --additional-tags stringToString    Tags to apply to the provisioned
infrastructure (default [])
      --allow-missing-template-keys       If true, ignore any errors in templates
when a field or map key is missing in the template. Only applies to golang and
jsonpath output formats. (default true)
  -c, --cluster-name name                 Name used to prefix the cluster and all the
created resources.
      --dry-run                           Only print the objects that would be
created, without creating them.
  -h, --help                              help for aks
      --http-proxy string                 HTTP proxy for nodes
      --https-proxy string                HTTPS proxy for nodes
      --kubeconfig string                 Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
      --kubernetes-version string         Kubernetes version. Run 'az aks get-
versions -o table --location <location>' to see available versions. See https://
docs.microsoft.com/en-us/azure/aks/supported-kubernetes-versions for more details.
Must be a patch version for v1.23.x.
      --machine-size string               Worker machine size (default
"Standard_D8s_v3")
  -n, --namespace string                  If present, the namespace scope for this
 CLI request. (default "default")
```

```
      --no-proxy strings                  No Proxy list for nodes (default [])
  -o, --output string                     Output format. One of: json|yaml|name|go-
template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|jsonpath-
file.
      --registry-mirror-cacert file       CA certificate chain to use while
 communicating with the registry mirror using TLS
      --registry-mirror-password string   Password to authenticate to the registry
mirror with
      --registry-mirror-url string        URL of a container registry to use as a
mirror in the cluster
      --registry-mirror-username string   Username to authenticate to the registry
mirror with
      --replicas int                      Number of replicas (default 1)
      --show-managed-fields               If true, keep the managedFields when
printing objects in JSON or YAML format.
      --ssh-public-key-file string        Path to the authorized SSH key for the user
      --ssh-username string               Name of the user to create on the instance
(default "konvoy")
      --template string                   Template string or path to template file to
use when -o=go-template, -o=go-template-file. The template format is golang templates
[http://golang.org/pkg/text/template/#pkg-overview].
      --timeout duration                  The length of time to wait before giving
up. Zero means wait forever. (default 30m0s)
      --wait                              If true, wait for operations to complete
before returning.
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create nodepool(see page 721) - Create a nodepool, one of [aks, aws, azure, eks, gcp, preprovisioned, vsphere]

dkp create nodepool aws

Create a nodepool in AWS

```
dkp create nodepool aws [flags]
```

Options

```
      --additional-security-group-ids strings   A comma separated list of existing
security group IDs to use for machines in addition to those created automatically (de
fault [])
```

```
      --additional-tags stringToString          Tags to apply to the provisioned
infrastructure (default [])
      --allow-missing-template-keys             If true, ignore any errors in
templates when a field or map key is missing in the template. Only applies to golang
and jsonpath output formats. (default true)
      --ami string                              AMI ID to use for machines
      --ami-base-os string                      Base OS for Lookup search (ex.
'centos-7', 'ubuntu-18.04', 'ubuntu-20.04') (default "ubuntu-20.04")
      --ami-format string                       Lookup Format string to generate AMI
search name from (default "capa-ami-{{.BaseOS}}-?{{.K8sVersion}}-*")
      --ami-owner string                        Owner ID for AMI Lookup search (defau
lt "258751437250")
      --availability-zone string                The AvailabilityZone in the region to
deploy the worker nodes to, if not set a random one will be selected (ex. us-west-2a)
  -c, --cluster-name name                       Name used to prefix the cluster and
all the created resources.
      --dry-run                                 Only print the objects that would be
created, without creating them.
  -h, --help                                    help for aws
      --http-proxy string                       HTTP proxy for nodes
      --https-proxy string                      HTTPS proxy for nodes
      --iam-instance-profile string             Name of the IAM instance profile to
assign to worker machines. (default "nodes.cluster-api-provider-aws.sigs.k8s.io")
      --instance-type string                    Worker machine instance type (default
 "m5.2xlarge")
      --kubeconfig string                       Path to the kubeconfig for the
management cluster. If unspecified, default discovery rules apply.
      --kubernetes-version string               Kubernetes version (default "1.23.12")
  -n, --namespace string                        If present, the namespace scope for
this CLI request. (default "default")
      --no-proxy strings                        No Proxy list for nodes (default [])
  -o, --output string                           Output format. One of: json|yaml|
name|go-template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|
jsonpath-file.
      --registry-mirror-cacert file             CA certificate chain to use while
 communicating with the registry mirror using TLS
      --registry-mirror-password string         Password to authenticate to the
registry mirror with
      --registry-mirror-url string              URL of a container registry to use as
a mirror in the cluster
      --registry-mirror-username string         Username to authenticate to the
registry mirror with
      --replicas int                            Number of replicas (default 1)
      --show-managed-fields                     If true, keep the managedFields when
printing objects in JSON or YAML format.
      --ssh-public-key-file string              Path to the authorized SSH key for
 the user
      --ssh-username string                     Name of the user to create on the
instance (default "konvoy")
      --template string                         Template string or path to template
file to use when -o=go-template, -o=go-template-file. The template format is golang
templates [http://golang.org/pkg/text/template/#pkg-overview].
```

```
      --timeout duration                          The length of time to wait before
giving up. Zero means wait forever. (default 30m0s)
      --wait                                      If true, wait for operations to
complete before returning.
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create nodepool(see page 721) - Create a nodepool, one of [aks, aws, azure, eks, gcp, preprovisioned, vsphere]

dkp create nodepool eks

Create a nodepool for EKS

```
dkp create nodepool eks [flags]
```

Options

```
      --additional-security-group-ids strings   A comma separated list of existing
security group IDs to use for machines in addition to those created automatically (de
fault [])
      --additional-tags stringToString          Tags to apply to the provisioned
infrastructure (default [])
      --allow-missing-template-keys             If true, ignore any errors in
templates when a field or map key is missing in the template. Only applies to golang
and jsonpath output formats. (default true)
      --availability-zone string                The AvailabilityZone in the region to
deploy the worker nodes to, if not set a random one will be selected (ex. us-west-2a)
  -c, --cluster-name name                       Name used to prefix the cluster and
all the created resources.
      --dry-run                                 Only print the objects that would be
created, without creating them.
  -h, --help                                    help for eks
      --http-proxy string                       HTTP proxy for nodes
      --https-proxy string                      HTTPS proxy for nodes
      --iam-instance-profile string             Name of the IAM instance profile to
assign to worker machines. (default "nodes.cluster-api-provider-aws.sigs.k8s.io")
      --instance-type string                    Worker machine instance type (default
 "m5.2xlarge")
      --kubeconfig string                       Path to the kubeconfig for the
management cluster. If unspecified, default discovery rules apply.
      --kubernetes-version string               Kubernetes version (default "1.22.6")
```

```
  -n, --namespace string                          If present, the namespace scope for
this CLI request. (default "default")
      --no-proxy strings                          No Proxy list for nodes (default [])
  -o, --output string                             Output format. One of: json|yaml|
name|go-template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|
jsonpath-file.
      --registry-mirror-cacert file               CA certificate chain to use while
 communicating with the registry mirror using TLS
      --registry-mirror-password string           Password to authenticate to the
registry mirror with
      --registry-mirror-url string                URL of a container registry to use as
a mirror in the cluster
      --registry-mirror-username string           Username to authenticate to the
registry mirror with
      --replicas int                              Number of replicas (default 1)
      --show-managed-fields                       If true, keep the managedFields when
printing objects in JSON or YAML format.
      --ssh-public-key-file string                Path to the authorized SSH key for
 the user
      --ssh-username string                       Name of the user to create on the
instance (default "konvoy")
      --template string                           Template string or path to template
file to use when -o=go-template, -o=go-template-file. The template format is golang
templates [http://golang.org/pkg/text/template/#pkg-overview].
      --timeout duration                          The length of time to wait before
giving up. Zero means wait forever. (default 30m0s)
      --wait                                      If true, wait for operations to
complete before returning.
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create nodepool(see page 721) - Create a nodepool, one of [aks, aws, azure, eks, gcp, preprovisioned, vsphere]

dkp create nodepool gcp

Create a nodepool in GCP

```
dkp create nodepool gcp name [flags]
```

Options

```
      --additional-tags stringToString    Tags to apply to the provisioned
infrastructure (default [])
      --allow-missing-template-keys       If true, ignore any errors in templates
when a field or map key is missing in the template. Only applies to golang and
jsonpath output formats. (default true)
      --associate-public-ip-address       Associate a public IP for all machines.
When set to false the specified network must have Cloud NAT configured to provide
internet access. (default true)
  -c, --cluster-name name                 Name used to prefix the cluster and all the
created resources.
      --dry-run                           Only print the objects that would be
created, without creating them.
  -h, --help                              help for gcp
      --http-proxy string                 HTTP proxy for nodes
      --https-proxy string                HTTPS proxy for nodes
      --image string                      Full reference to an image to use for all
nodes (set either this or --image-family) (ex. 'projects/my-project/global/images/
konvoy-ubuntu-2004-1-99-99-1234567890')
      --image-family string               Full reference to an image family to use
for all nodes (set either this or --image) (ex. 'projects/my-project/global/images/
family/konvoy-ubuntu-2004-{{.K8sVersion}}')
      --instance-type string              Worker machine instance type (default "n2-
standard-8")
      --kubeconfig string                 Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
      --kubernetes-version string         Kubernetes version (default "1.23.12")
  -n, --namespace string                  If present, the namespace scope for this
 CLI request. (default "default")
      --no-proxy strings                  No Proxy list for nodes (default [])
  -o, --output string                     Output format. One of: json|yaml|name|go-
template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|jsonpath-
file.
      --registry-mirror-cacert file       CA certificate chain to use while
 communicating with the registry mirror using TLS
      --registry-mirror-password string   Password to authenticate to the registry
mirror with
      --registry-mirror-url string        URL of a container registry to use as a
mirror in the cluster
      --registry-mirror-username string   Username to authenticate to the registry
mirror with
      --replicas int                      Number of replicas (default 1)
      --service-account-email string      Worker machine Service Account email
address (default "default")
      --show-managed-fields               If true, keep the managedFields when
printing objects in JSON or YAML format.
      --ssh-public-key-file string        Path to the authorized SSH key for the user
      --ssh-username string               Name of the user to create on the instance
(default "konvoy")
```

```
      --template string                    Template string or path to template file to
use when -o=go-template, -o=go-template-file. The template format is golang templates
[http://golang.org/pkg/text/template/#pkg-overview].
      --timeout duration                   The length of time to wait before giving
up. Zero means wait forever. (default 30m0s)
      --wait                               If true, wait for operations to complete
before returning.
      --zone string                        Zone in the region to deploy the worker
nodes to, if not set a random one will be selected (ex. us-west1-a)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create nodepool(see page 721) - Create a nodepool, one of [aks, aws, azure, eks, gcp, preprovisioned, vsphere]

dkp create nodepool preprovisioned

Create a nodepool for a Pre Provisioned Cluster.

```
dkp create nodepool preprovisioned name [flags]
```

Options

```
      --additional-tags stringToString    Tags to apply to the provisioned
infrastructure (default [])
      --allow-missing-template-keys       If true, ignore any errors in templates
when a field or map key is missing in the template. Only applies to golang and
jsonpath output formats. (default true)
  -c, --cluster-name name                 Name used to prefix the cluster and all the
created resources.
      --dry-run                           Only print the objects that would be
created, without creating them.
  -h, --help                              help for preprovisioned
      --http-proxy string                 HTTP proxy for nodes
      --https-proxy string                HTTPS proxy for nodes
      --kubeconfig string                 Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
      --kubernetes-version string         Kubernetes version (default "1.23.12")
  -n, --namespace string                  If present, the namespace scope for this
 CLI request. (default "default")
      --no-proxy strings                  No Proxy list for nodes (default [])
```

```
      --os-hint flatcar                    A hint which will allow the installer to
generate appropriate configurations for a target OS. Presently, only the hint for
 flatcar is supported.
  -o, --output string                      Output format. One of: json|yaml|name|go-
template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|jsonpath-
file.
      --override-secret-name string        Name of the secret for any provided
overrides on a preprovisioned cluster.All overrides defined at provisioning should be
present in this secret.
      --registry-mirror-cacert file        CA certificate chain to use while
 communicating with the registry mirror using TLS
      --registry-mirror-password string    Password to authenticate to the registry
mirror with
      --registry-mirror-url string         URL of a container registry to use as a
mirror in the cluster
      --registry-mirror-username string    Username to authenticate to the registry
mirror with
      --replicas int                       Number of replicas (default 1)
      --show-managed-fields                If true, keep the managedFields when
printing objects in JSON or YAML format.
      --ssh-public-key-file string         Path to the authorized SSH key for the user
      --ssh-username string                Name of the user to create on the instance
(default "konvoy")
      --template string                    Template string or path to template file to
use when -o=go-template, -o=go-template-file. The template format is golang templates
[http://golang.org/pkg/text/template/#pkg-overview].
      --timeout duration                   The length of time to wait before giving
up. Zero means wait forever. (default 30m0s)
      --wait                               If true, wait for operations to complete
before returning.
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp create nodepool(see page 721) - Create a nodepool, one of [aks, aws, azure, eks, gcp, preprovisioned, vsphere]

9.2.7 dkp delete

Delete one of [bootstrap, capi-components, chart, cluster, nodepool]

9.2.7.1 Options

```
  -h, --help   help for delete
```

9.2.7.2 Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

9.2.7.3 SEE ALSO

- dkp delete bootstrap(see page 733) - Delete bootstrap cluster
- dkp delete capi-components(see page 733) - Delete the CAPI components from the cluster
- dkp delete chart(see page 735) - Delete a chart from the repository
- dkp delete cluster(see page 734) - Delete a Kubernetes cluster
- dkp delete nodepool(see page 736) - Delete a nodepool for a given cluster

9.2.7.4 dkp delete bootstrap

Delete bootstrap cluster

```
dkp delete bootstrap [flags]
```

Options

```
  -h, --help                      help for bootstrap
      --kind-cluster-name string   Kind cluster name for the bootstrap cluster (defau
lt "konvoy-capi-bootstrapper")
      --kubeconfig string          Path to the kubeconfig for the management cluster.
If unspecified, default discovery rules apply.
      --timeout duration           The length of time to wait before giving up. Zero
means wait forever. (default 5m0s)
      --wait                       If true, wait for operations to complete before
returning. (default true)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp delete(see page 732) - Delete one of [bootstrap, capi-components, chart, cluster, nodepool]

9.2.7.5 dkp delete capi-components

Delete the CAPI components from the cluster

```
dkp delete capi-components [flags]
```

Options

```
  -h, --help                help for capi-components
      --kubeconfig string   Path to the kubeconfig for the management cluster. If
unspecified, default discovery rules apply.
      --timeout duration    The length of time to wait before giving up. Zero means
wait forever. (default 5m0s)
      --wait                If true, wait for operations to complete before
returning. (default true)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp delete(see page 732) - Delete one of [bootstrap, capi-components, chart, cluster, nodepool]

9.2.7.6 dkp delete cluster

Delete a Kubernetes cluster

```
dkp delete cluster [flags]
```

Options

```
      --aws-service-endpoints string     Custom AWS service endpoints in a semi-colon
separated format: ${SigningRegion1}:${ServiceID1}=${URL},${ServiceID2}=${URL};$
{SigningRegion2}...
  -c, --cluster-name name                Name used to prefix the cluster and all the
created resources.
      --delete-kubernetes-resources      Delete Kubernetes resources on the cluster
before deleting that cluster (Services with type LoadBalancer) (default true)
  -h, --help                             help for cluster
      --http-proxy string                HTTP proxy for CAPI controllers
      --https-proxy string               HTTPS proxy for CAPI controllers
      --kind-cluster-image string        Kind node image for the bootstrap cluster (d
efault "mesosphere/konvoy-bootstrap:v2.3.3")
      --kubeconfig string                Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
```

```
  -n, --namespace string                If present, the namespace scope for this CLI
request. (default "default")
      --no-proxy strings                No Proxy list for CAPI controllers (default
 [])
      --self-managed                    When set to true, the required prerequisites
and resources are moved from the self managed cluster before deleting. When set to
false, the resources are assumed installed in a management cluster.(default false)
      --timeout duration                The length of time to wait before giving up.
Zero means wait forever. (default 15m0s)
      --wait                            If true, wait for operations to complete
before returning. (default true)
      --with-aws-bootstrap-credentials   Set true to use AWS bootstrap credentials
from your environment. When false, the instance profile of the EC2 instance where the
CAPA controller is scheduled on will be used instead.
      --with-gcp-bootstrap-credentials   Set true to use GCP bootstrap credentials
from your environment. When false, the service account of the VM instance where the
CAPG controller is scheduled on will be used instead.
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp delete(see page 732) - Delete one of [bootstrap, capi-components, chart, cluster, nodepool]

9.2.7.7 dkp delete chart

Delete a chart from the repository

```
dkp delete chart [chartName] [chartVersion] [flags]
```

Options

```
      --config string           Config file to use (default "/root/.kommander/
config")
      --context string          The name of the kubeconfig context to use
  -h, --help                    help for chart
      --kubeconfig string       Path to the kubeconfig file to use for CLI requests.
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp delete(see page 732) - Delete one of [bootstrap, capi-components, chart, cluster, nodepool]

9.2.7.8 dkp delete nodepool

Delete a nodepool for a given cluster

```
dkp delete nodepool name [flags]
```

Options

```
  -c, --cluster-name name   Name used to prefix the cluster and all the created
resources.
      --dry-run             Only print the objects that would be created, without
creating them.
  -h, --help                help for nodepool
      --kubeconfig string   Path to the kubeconfig for the management cluster. If
unspecified, default discovery rules apply.
  -n, --namespace string    If present, the namespace scope for this CLI request. (de
fault "default")
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp delete(see page 732) - Delete one of [bootstrap, capi-components, chart, cluster, nodepool]

9.2.8 dkp describe

Describe one of [cluster]

9.2.8.1 Options

```
  -h, --help   help for describe
```

9.2.8.2 Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

9.2.8.3 SEE ALSO

- dkp describe cluster(see page 737) - Describe a Kubernetes cluster status

9.2.8.4 dkp describe cluster

Describe a Kubernetes cluster status

```
dkp describe cluster [flags]
```

Options

```
  -c, --cluster-name name   Name used to prefix the cluster and all the created
resources.
  -h, --help                help for cluster
      --kubeconfig string   Path to the kubeconfig for the management cluster. If
unspecified, default discovery rules apply.
  -n, --namespace string    If present, the namespace scope for this CLI request. (de
fault "default")
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp describe(see page 736) - Describe one of [cluster]

9.2.9 dkp detach

Detach one of [cluster]

9.2.9.1 Options

```
      --config string            Config file to use (default "/root/.kommander/
config")
      --context string           The name of the kubeconfig context to use
  -h, --help                     help for detach
      --kubeconfig string        Path to the kubeconfig file to use for CLI requests.
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
```

9.2.9.2 Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

9.2.9.3 SEE ALSO

- dkp detach cluster(see page 738) - Detach a cluster

9.2.9.4 dkp detach cluster

Detach a cluster

```
dkp detach cluster CLUSTER_NAME [flags]
```

Options

```
  -h, --help               help for cluster
      --timeout duration   The length of time to wait before giving up on a detach,
defaults to wait forever (default 0s)
      --wait               If true, wait for resources to be gone before returning.
This waits for finalizers. (default true)
  -w, --workspace string   Name of the workspace of the attached cluster
```

Options inherited from parent commands

```
      --config string            Config file to use (default "/root/.kommander/
config")
      --context string           The name of the kubeconfig context to use
      --kubeconfig string        Path to the kubeconfig file to use for CLI requests.
```

```
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
  -v, --verbose int              Output verbosity
```

SEE ALSO

- dkp detach(see page 737) - Detach one of [cluster]

9.2.10 dkp diagnose

Generate a support bundle

9.2.10.1 Synopsis

A support bundle is an archive of files, output, metrics and state from a server that can be used to assist when troubleshooting a Kubernetes cluster.

```
dkp diagnose [flags]
```

9.2.10.2 Options

```
      --allow-insecure-connections     When set, do not verify TLS certs when
retrieving spec and reporting results
      --as string                      Username to impersonate for the operation.
User could be a regular user or a service account in a namespace.
      --as-group stringArray           Group to impersonate for the operation, this
 flag can be repeated to specify multiple groups. (default [])
      --as-uid string                  UID to impersonate for the operation.
      --bootstrap-kubeconfig string    Path to the kubeconfig file to use for
 requests towards an additional bootstrap cluster
      --cache-dir string               Default cache directory (default "/root/.kube/
cache")
      --certificate-authority string   Path to a cert file for the certificate
authority
      --client-certificate string      Path to a client certificate file for TLS
      --client-key string              Path to a client key file for TLS
      --cluster string                 The name of the kubeconfig cluster to use
      --collect-without-permissions    Always generate a support bundle, even if it
some require additional permissions (default true)
      --context string                 The name of the kubeconfig context to use
  -h, --help                           help for diagnose
      --insecure-skip-tls-verify       If true, the server's certificate will not be
checked for validity. This will make your HTTPS connections insecure
      --kubeconfig string              Path to the kubeconfig file to use for CLI
requests.
```

```
  -n, --namespace string               If present, the namespace scope for this CLI
request
      --redactors strings              Names of the additional redactors to use
(default [])
      --request-timeout string         The length of time to wait before giving up on
a single server request. Non-zero values should contain a corresponding time unit
(e.g. 1s, 2m, 3h). A value of zero means don't timeout requests.
  -s, --server string                  The address and port of the Kubernetes API
server
      --since string                   Force pod logs collectors to return logs newer
than a relative duration like 5s, 2m, or 3h.
      --since-time string              Force pod logs collectors to return logs after
a specific date (RFC3339)
      --tls-server-name string         Server name to use for server certificate
validation. If it is not provided, the hostname used to contact the server is used
      --token string                   Bearer token for authentication to the API
server
      --user string                    The name of the kubeconfig user to use
```

9.2.10.3 Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

9.2.10.4 SEE ALSO

- dkp diagnose default-config(see page 741) - Prints the default configuration of the diagnostics bundle collectors
- dkp diagnose ssh(see page 740) - Collect node-level diagnostics data over SSH

9.2.10.5 dkp diagnose ssh

Collect node-level diagnostics data over SSH

```
dkp diagnose ssh path/to/inventory-file.yaml [flags]
```

Options

```
  -h, --help                help for ssh
      --redactors strings   Names of the additional redactors to use (default [])
      --timeout duration    Timeout for collecting bundle per node (default 5m0s)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp diagnose(see page 739) - Generate a support bundle

9.2.10.6 dkp diagnose default-config

Prints the default configuration of the diagnostics bundle collectors

```
dkp diagnose default-config [flags]
```

Options

```
  -h, --help   help for default-config
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp diagnose(see page 739) - Generate a support bundle

9.2.11 dkp get

Get one of [appdeployments, chart, clusters, kubeconfig, nodepools, workspaces]

9.2.11.1 Options

```
  -h, --help   help for get
```

9.2.11.2 Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

9.2.11.3 SEE ALSO

- dkp get appdeployments(see page 742) - Get AppDeployments from a Workspace, Project, or all Workspaces and Projects
- dkp get chart(see page 744) - Obtain information about charts stored in the repository
- dkp get clusters(see page 745) - Get clusters from specified Workspace
- dkp get kubeconfig(see page 742) - Retrieve cluster kubeconfig and modify local kubeconfig file
- dkp get nodepools(see page 744) - Get nodepools for a given cluster
- dkp get workspaces(see page 743) - Get Workspaces

9.2.11.4 dkp get kubeconfig

Retrieve cluster kubeconfig and modify local kubeconfig file

```
dkp get kubeconfig [flags]
```

Options

```
      --cluster string      Kommander Cluster to get kubeconfig for
  -c, --cluster-name name   Name used to prefix the cluster and all the created
resources.
  -h, --help                help for kubeconfig
      --kubeconfig string   Path to the kubeconfig for the management cluster. If
unspecified, default discovery rules apply.
  -n, --namespace string    If present, the namespace scope for this CLI request. (de
fault "default")
  -w, --workspace string    Name of the workspace to show clusters from
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp get(see page 741) - Get one of [appdeployments, chart, clusters, kubeconfig, nodepools, workspaces]

9.2.11.5 dkp get appdeployments

Get AppDeployments from a Workspace, Project, or all Workspaces and Projects

Synopsis

Prints a table of the most important information about the specified resources. In case the AppDeployments are configured with cluster scoped specification, an optional CLUSTERS column (when printing in table format) will display enabled clusters for each AppDeployment.

```
dkp get appdeployments [APPDEPLOYMENT_NAME] [flags]
```

Options

```
  -A, --all-namespaces           If present, list the requested object(s) across all
namespaces.
      --config string            Config file to use (default "/root/.kommander/
config")
      --context string           The name of the kubeconfig context to use
  -h, --help                     help for appdeployments
      --kubeconfig string        Path to the kubeconfig file to use for CLI requests.
  -o, --output string            Output format. One of: table|yaml
  -p, --project string           Name of the project to show AppDeployments from.
Requires workspace flag (workspace that the project belongs to).
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
  -w, --workspace string         Name of the workspace to show AppDeployments from
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp get(see page 741) - Get one of [appdeployments, chart, clusters, kubeconfig, nodepools, workspaces]

9.2.11.6 dkp get workspaces

Get Workspaces

```
dkp get workspaces [flags]
```

Options

```
  -A, --all-namespaces           If present, list the requested object(s) across all
namespaces.
      --config string            Config file to use (default "/root/.kommander/
config")
      --context string           The name of the kubeconfig context to use
  -h, --help                     help for workspaces
      --kubeconfig string        Path to the kubeconfig file to use for CLI requests.
  -o, --output string            Output format. One of: table|yaml
```

```
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp get(see page 741) - Get one of [appdeployments, chart, clusters, kubeconfig, nodepools, workspaces]

9.2.11.7 dkp get nodepools

Get nodepools for a given cluster

```
dkp get nodepools [flags]
```

Options

```
  -c, --cluster-name name   Name used to prefix the cluster and all the created
resources.
  -h, --help                help for nodepools
      --kubeconfig string   Path to the kubeconfig for the management cluster. If
unspecified, default discovery rules apply.
  -n, --namespace string    If present, the namespace scope for this CLI request. (de
fault "default")
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp get(see page 741) - Get one of [appdeployments, chart, clusters, kubeconfig, nodepools, workspaces]

9.2.11.8 dkp get chart

Obtain information about charts stored in the repository

```
dkp get chart [chartName] [chartVersion] [flags]
```

Options

```
  -A, --all-namespaces           If present, list the requested object(s) across all
namespaces.
      --config string            Config file to use (default "/root/.kommander/
config")
      --context string           The name of the kubeconfig context to use
  -h, --help                     help for chart
      --kubeconfig string        Path to the kubeconfig file to use for CLI requests.
  -o, --output string            Output format. One of: table|yaml
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp get(see page 741) - Get one of [appdeployments, chart, clusters, kubeconfig, nodepools, workspaces]

9.2.11.9 dkp get clusters

Get clusters from specified Workspace

```
dkp get clusters [CLUSTER_NAME] [flags]
```

Options

```
  -A, --all-namespaces           If present, list the requested object(s) across all
namespaces.
      --config string            Config file to use (default "/root/.kommander/
config")
      --context string           The name of the kubeconfig context to use
  -h, --help                     help for clusters
      --kubeconfig string        Path to the kubeconfig file to use for CLI requests.
  -o, --output string            Output format. One of: table|yaml
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
  -w, --workspace string         Name of the workspace to show clusters from
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp get(see page 741) - Get one of [appdeployments, chart, clusters, kubeconfig, nodepools, workspaces]

9.2.12 dkp import

Import images from an image bundle into Containerd

9.2.12.1 Options

```
  -h, --help   help for import
```

9.2.12.2 Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

9.2.12.3 SEE ALSO

- dkp import image-bundle(see page 746) - Import images from an image bundle into Containerd

9.2.12.4 dkp import image-bundle

Import images from an image bundle into Containerd

```
dkp import image-bundle [flags]
```

Options

```
      --containerd-namespace string   Containerd namespace to import images into (def
ault "k8s.io")
  -h, --help                          help for image-bundle
      --image-bundle string           Tarball containing list of images to push
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp import(see page 746) - Import images from an image bundle into Containerd

9.2.13 dkp install

Install one of [kommander]

9.2.13.1 Options

```
      --config string            Config file to use (default "/root/.kommander/
config")
      --context string           The name of the kubeconfig context to use
  -h, --help                     help for install
      --kubeconfig string        Path to the kubeconfig file to use for CLI requests.
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
```

9.2.13.2 Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

9.2.13.3 SEE ALSO

- dkp install kommander(see page 747) - Install kommander

9.2.13.4 dkp install kommander

Install kommander

```
dkp install kommander [flags]
```

Options

```
      --airgapped                                  Enable airgapped mode.
      --bootstrap-repository string                git repository with bootstrap
definitions
      --charts-bundle stringArray                  Path to charts-bundle to upload to
chartmuseum, apart from parsing the kommander applications repository (default [])
      --disallow-charts-download                   make CLI rely solely on provided
chart bundles and do not try to download charts from the Internet
      --gitea-kommander-repository-name string     gitea kommander repository name
(default "kommander")
  -h, --help                                       help for kommander
      --init                                       Initialize default configuration
file, print it and exit without installing Kommander.
      --installer-config file                      Path to installation configuration
file
      --kommander-applications-repository string   git repository with application
definitions (default "v2.3.3")
  -o, --output string                              Output format for the
configuration file generated by --init. One of: yaml|json (default "yaml")
```

Options inherited from parent commands

```
      --config string            Config file to use (default "/root/.kommander/
config")
      --context string           The name of the kubeconfig context to use
      --kubeconfig string        Path to the kubeconfig file to use for CLI requests.
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
  -v, --verbose int              Output verbosity
```

SEE ALSO

- dkp install(see page 747) - Install one of [kommander]

9.2.14 dkp move

Move one of [capi-resources]

9.2.14.1 Synopsis

Command "move" is deprecated, use "dkp move capi-resources" instead Move one of [capi-resources]

```
dkp move [flags]
```

9.2.14.2 Options

```
      --from-context string    Context to be used within the from-cluster's
kubeconfig file. If empty, current context will be used. (DEPRECATED: use "dkp move
capi-resources" instead)
      --from-kubeconfig file   Path to the kubeconfig for pivot's source cluster. If
unspecified, default discovery rules apply. (DEPRECATED: use "dkp move capi-
resources" instead)
  -h, --help                   help for move
      --to-context string      Context to be used within the to-cluster's kubeconfig
file. If empty, current context will be used. (DEPRECATED: use "dkp move capi-
resources" instead)
      --to-kubeconfig file     Path to the kubeconfig for pivot's destination cluster
(DEPRECATED: use "dkp move capi-resources" instead)
```

9.2.14.3 Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

9.2.14.4 SEE ALSO

- dkp move capi-resources(see page 749) - Move controllers and objects from one cluster to the other

9.2.14.5 dkp move capi-resources

Move controllers and objects from one cluster to the other

```
dkp move capi-resources [flags]
```

Options

```
      --from-context string    Context to be used within the from-cluster's
kubeconfig file. If empty, current context will be used.
      --from-kubeconfig file   Path to the kubeconfig for pivot's source cluster. If
unspecified, default discovery rules apply.
  -h, --help                   help for capi-resources
      --to-context string      Context to be used within the to-cluster's kubeconfig
file. If empty, current context will be used.
      --to-kubeconfig file     Path to the kubeconfig for pivot's destination cluster
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp move(see page 748) - Move one of [capi-resources]

9.2.15 dkp open

Open one of [dashboard]

9.2.15.1 Options

```
      --config string            Config file to use (default "/root/.kommander/
config")
      --context string           The name of the kubeconfig context to use
  -h, --help                     help for open
      --kubeconfig string        Path to the kubeconfig file to use for CLI requests.
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
```

9.2.15.2 Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

9.2.15.3 SEE ALSO

- dkp open dashboard(see page 750) - Open DKP UI in your browser

9.2.15.4 dkp open dashboard

Open DKP UI in your browser

```
dkp open dashboard [flags]
```

Options

```
  -h, --help   help for dashboard
```

Options inherited from parent commands

```
      --config string            Config file to use (default "/root/.kommander/
config")
      --context string           The name of the kubeconfig context to use
      --kubeconfig string        Path to the kubeconfig file to use for CLI requests.
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
  -v, --verbose int              Output verbosity
```

SEE ALSO

- dkp open(see page 750) - Open one of [dashboard]

9.2.16 dkp push

Push one of [chart, chart-bundle, image-bundle]

9.2.16.1 Options

```
  -h, --help   help for push
```

9.2.16.2 Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

9.2.16.3 SEE ALSO

- dkp push chart(see page 753) - Upload charts to the repository
- dkp push chart-bundle(see page 751) - Upload chart bundles to the repository
- dkp push image-bundle(see page 752) - Push images from an image bundle into an existing image registry

9.2.16.4 dkp push chart-bundle

Upload chart bundles to the repository

```
dkp push chart-bundle [chartsTarball]... [flags]
```

Options

```
      --config string            Config file to use (default "/root/.kommander/
config")
      --context string           The name of the kubeconfig context to use
  -h, --help                     help for chart-bundle
      --kubeconfig string        Path to the kubeconfig file to use for CLI requests.
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp push(see page 751) - Push one of [chart, chart-bundle, image-bundle]

9.2.16.5 dkp push image-bundle

Push images from an image bundle into an existing image registry

```
dkp push image-bundle [flags]
```

Options

```
      --ecr-lifecycle-policy-file string       File containing ECR lifecycle policy
for newly created repositories (only applies if target registry is hosted on ECR,
ignored otherwise)
  -h, --help                                   help for image-bundle
      --image-bundle string                    Tarball containing list of images to
push
      --to-registry string                     Registry to push images to
      --to-registry-insecure-skip-tls-verify   Skip TLS verification of registry to
push images to (use for http registries)
      --to-registry-password string            Password to use to log in to
destination registry
      --to-registry-username string            Username to use to log in to
destination registry
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp push(see page 751) - Push one of [chart, chart-bundle, image-bundle]

9.2.16.6 dkp push chart

Upload charts to the repository

```
dkp push chart [chartTarball]... [flags]
```

Options

```
      --config string            Config file to use (default "/root/.kommander/
config")
      --context string           The name of the kubeconfig context to use
  -h, --help                     help for chart
      --kubeconfig string        Path to the kubeconfig file to use for CLI requests.
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp push(see page 751) - Push one of [chart, chart-bundle, image-bundle]

9.2.17 dkp scale

Scale one of [nodepool]

9.2.17.1 Options

```
  -h, --help   help for scale
```

9.2.17.2 Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

9.2.17.3 SEE ALSO

- dkp scale nodepool(see page 754) - Scale a nodepool of a given cluster to the number of replicas

9.2.17.4 dkp scale nodepool

Scale a nodepool of a given cluster to the number of replicas

```
dkp scale nodepool name [flags]
```

Options

```
  -c, --cluster-name name         Name used to prefix the cluster and all the created
resources.
  -h, --help                      help for nodepool
      --kubeconfig string         Path to the kubeconfig for the management cluster.
If unspecified, default discovery rules apply.
  -n, --namespace string          If present, the namespace scope for this CLI
request. (default "default")
      --nodes-to-delete strings   A list of node names to mark for deletion when
scaling down a node pool. If left empty, the nodes to delete will be selected at
random. (default [])
      --replicas int              The new desired number of replicas.
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp scale(see page 753) - Scale one of [nodepool]

9.2.18 dkp serve

Serve an image registry

9.2.18.1 Options

```
  -h, --help   help for serve
```

9.2.18.2 Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

9.2.18.3 SEE ALSO

- dkp serve image-bundle(see page 755) - Serve an image registry

9.2.18.4 dkp serve image-bundle

Serve an image registry

```
dkp serve image-bundle [flags]
```

Options

```
  -h, --help                          help for image-bundle
      --image-bundle string           Tarball containing list of images to push
      --listen-address string         Address to list on (default "localhost")
      --listen-port uint16            Port to listen on (0 means use any free port)
      --tls-cert-file string          TLS certificate file
      --tls-private-key-file string   TLS private key file
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp serve(see page 754) - Serve an image registry

9.2.19 dkp update

Update one of [bootstrap (cluster), controlplane, nodepool]

9.2.19.1 Options

```
-h, --help   help for update
```

9.2.19.2 Options inherited from parent commands

```
-v, --verbose int   Output verbosity
```

9.2.19.3 SEE ALSO

- dkp update bootstrap(see page 764) - Update bootstrap cluster
- dkp update controlplane(see page 760) - Update a Kubernetes cluster control plane, one of [aws, azure, eks, preprovisioned, vsphere]
- dkp update nodepool(see page 756) - Upate a Kubernetes cluster node pool, one of [aws, azure, eks, preprovisioned, vsphere]

9.2.19.4 dkp update nodepool

Upate a Kubernetes cluster node pool, one of [aws, azure, eks, preprovisioned, vsphere]

Options

```
-h, --help   help for nodepool
```

Options inherited from parent commands

```
-v, --verbose int   Output verbosity
```

SEE ALSO

- dkp update(see page 755) - Update one of [bootstrap (cluster), controlplane, nodepool]
- dkp update nodepool aws(see page 757) - Update a Konvoy cluster node pool in AWS
- dkp update nodepool azure(see page 756) - Update a Konvoy cluster node pool in Azure
- dkp update nodepool eks(see page 759) - Update a Konvoy cluster node pool in EKS
- dkp update nodepool preprovisioned(see page 758) - Update a Konvoy cluster node pool in Preprovisioned
- dkp update nodepool vsphere(see page 759) - Update a Konvoy cluster node pool in vSphere

dkp update nodepool azure

Update a Konvoy cluster node pool in Azure

```
dkp update nodepool azure [flags]
```

Options

```
  -c, --cluster-name name           Name used to prefix the cluster and all the
created resources.
  -h, --help                        help for azure
      --kubeconfig string           Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
      --kubernetes-version string   Kubernetes version
      --machine-size string         Worker machine size (ex. 'Standard_D2s_v3')
  -n, --namespace string            If present, the namespace scope for this CLI
request. (default "default")
      --use-context string          Use a specific context in a kubeconfig file.
      --wait                        If true, wait for operations to complete before
returning. (default true)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp update nodepool(see page 756) - Upate a Kubernetes cluster node pool, one of [aws, azure, eks, preprovisioned, vsphere]

dkp update nodepool aws

Update a Konvoy cluster node pool in AWS

```
dkp update nodepool aws [flags]
```

Options

```
      --ami string                  AMI id to use for node pool machines
  -c, --cluster-name name           Name used to prefix the cluster and all the
created resources.
  -h, --help                        help for aws
      --instance-type string        Instance type to use for node pool machines
      --kubeconfig string           Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
      --kubernetes-version string   Kubernetes version
```

```
  -n, --namespace string            If present, the namespace scope for this CLI
request. (default "default")
      --use-context string          Use a specific context in a kubeconfig file.
      --wait                        If true, wait for operations to complete before
returning. (default true)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp update nodepool(see page 756) - Upate a Kubernetes cluster node pool, one of [aws, azure, eks, preprovisioned, vsphere]

dkp update nodepool preprovisioned

Update a Konvoy cluster node pool in Preprovisioned

```
dkp update nodepool preprovisioned [flags]
```

Options

```
  -c, --cluster-name name           Name used to prefix the cluster and all the
created resources.
  -h, --help                        help for preprovisioned
      --kubeconfig string           Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
      --kubernetes-version string   Kubernetes version
  -n, --namespace string            If present, the namespace scope for this CLI
request. (default "default")
      --use-context string          Use a specific context in a kubeconfig file.
      --wait                        If true, wait for operations to complete before
returning. (default true)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp update nodepool(see page 756) - Upate a Kubernetes cluster node pool, one of [aws, azure, eks, preprovisioned, vsphere]

dkp update nodepool eks

Update a Konvoy cluster node pool in EKS

```
dkp update nodepool eks [flags]
```

Options

```
  -c, --cluster-name name           Name used to prefix the cluster and all the
created resources.
  -h, --help                        help for eks
      --instance-type string        Instance type to use for node pool machines
      --kubeconfig string           Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
      --kubernetes-version string   Kubernetes version
  -n, --namespace string            If present, the namespace scope for this CLI
request. (default "default")
      --use-context string          Use a specific context in a kubeconfig file.
      --wait                        If true, wait for operations to complete before
returning. (default true)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp update nodepool(see page 756) - Upate a Kubernetes cluster node pool, one of [aws, azure, eks, preprovisioned, vsphere]

dkp update nodepool vsphere

Update a Konvoy cluster node pool in vSphere

```
dkp update nodepool vsphere [flags]
```

Options

```
  -c, --cluster-name name           Name used to prefix the cluster and all the
created resources.
      --cpus int                    The number of virtual processors in a virtual
machine.
```

```
      --disk-size int                The size of a virtual machine's disk, in GB.
  -h, --help                         help for vsphere
      --kubeconfig string            Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
      --kubernetes-version string    Kubernetes version
      --memory int                   The size of a virtual machine's memory, in GB.
  -n, --namespace string             If present, the namespace scope for this CLI
request. (default "default")
      --use-context string           Use a specific context in a kubeconfig file.
      --vm-template string           The virtual machine template to use for a virtual
machine.
      --wait                         If true, wait for operations to complete before
returning. (default true)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp update nodepool(see page 756) - Upate a Kubernetes cluster node pool, one of [aws, azure, eks, preprovisioned, vsphere]

9.2.19.5 dkp update controlplane

Update a Kubernetes cluster control plane, one of [aws, azure, eks, preprovisioned, vsphere]

Options

```
  -h, --help   help for controlplane
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp update(see page 755) - Update one of [bootstrap (cluster), controlplane, nodepool]
- dkp update controlplane aws(see page 761) - Update a Konvoy cluster control plane in AWS
- dkp update controlplane azure(see page 762) - Update a Konvoy cluster control plane in Azure
- dkp update controlplane eks(see page 763) - Update a Konvoy cluster control plane in EKS
- dkp update controlplane preprovisioned(see page 761) - Update a Konvoy cluster control plane in Preprovisioned
- dkp update controlplane vsphere(see page 763) - Update a Konvoy cluster control plane in vSphere

dkp update controlplane preprovisioned

Update a Konvoy cluster control plane in Preprovisioned

```
dkp update controlplane preprovisioned [flags]
```

Options

```
  -c, --cluster-name name           Name used to prefix the cluster and all the
created resources.
  -h, --help                        help for preprovisioned
      --kubeconfig string           Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
      --kubernetes-version string   Kubernetes version
  -n, --namespace string            If present, the namespace scope for this CLI
request. (default "default")
      --use-context string          Use a specific context in a kubeconfig file.
      --wait                        If true, wait for operations to complete before
returning. (default true)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp update controlplane(see page 760) - Update a Kubernetes cluster control plane, one of [aws, azure, eks, preprovisioned, vsphere]

dkp update controlplane aws

Update a Konvoy cluster control plane in AWS

```
dkp update controlplane aws [flags]
```

Options

```
      --ami string                  AMI id to use for control plane machines
  -c, --cluster-name name           Name used to prefix the cluster and all the
created resources.
  -h, --help                        help for aws
      --instance-type string        Instance type to use for control plane machines
```

```
      --kubeconfig string           Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
      --kubernetes-version string   Kubernetes version
  -n, --namespace string            If present, the namespace scope for this CLI
request. (default "default")
      --use-context string          Use a specific context in a kubeconfig file.
      --wait                        If true, wait for operations to complete before
returning. (default true)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp update controlplane(see page 760) - Update a Kubernetes cluster control plane, one of [aws, azure, eks, preprovisioned, vsphere]

dkp update controlplane azure

Update a Konvoy cluster control plane in Azure

```
dkp update controlplane azure [flags]
```

Options

```
  -c, --cluster-name name           Name used to prefix the cluster and all the
created resources.
  -h, --help                        help for azure
      --kubeconfig string           Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
      --kubernetes-version string   Kubernetes version
      --machine-size string         Worker machine size (ex. 'Standard_D2s_v3')
  -n, --namespace string            If present, the namespace scope for this CLI
request. (default "default")
      --use-context string          Use a specific context in a kubeconfig file.
      --wait                        If true, wait for operations to complete before
returning. (default true)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp update controlplane(see page 760) - Update a Kubernetes cluster control plane, one of [aws, azure, eks, preprovisioned, vsphere]

dkp update controlplane eks

Update a Konvoy cluster control plane in EKS

```
dkp update controlplane eks [flags]
```

Options

```
  -c, --cluster-name name           Name used to prefix the cluster and all the
created resources.
  -h, --help                        help for eks
      --kubeconfig string           Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
      --kubernetes-version string   Kubernetes version
  -n, --namespace string            If present, the namespace scope for this CLI
request. (default "default")
      --use-context string          Use a specific context in a kubeconfig file.
      --wait                        If true, wait for operations to complete before
returning. (default true)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp update controlplane(see page 760) - Update a Kubernetes cluster control plane, one of [aws, azure, eks, preprovisioned, vsphere]

dkp update controlplane vsphere

Update a Konvoy cluster control plane in vSphere

```
dkp update controlplane vsphere [flags]
```

Options

```
  -c, --cluster-name name           Name used to prefix the cluster and all the
created resources.
      --cpus int                    The number of virtual processors in a virtual
machine.
      --disk-size int               The size of a virtual machine's disk, in GB.
  -h, --help                        help for vsphere
      --kubeconfig string           Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
      --kubernetes-version string   Kubernetes version
      --memory int                  The size of a virtual machine's memory, in GB.
  -n, --namespace string            If present, the namespace scope for this CLI
request. (default "default")
      --use-context string          Use a specific context in a kubeconfig file.
      --vm-template string          The virtual machine template to use for a virtual
machine.
      --wait                        If true, wait for operations to complete before
returning. (default true)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp update controlplane(see page 760) - Update a Kubernetes cluster control plane, one of [aws, azure, eks, preprovisioned, vsphere]

9.2.19.6 dkp update bootstrap

Update bootstrap cluster

Options

```
  -h, --help   help for bootstrap
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp update(see page 755) - Update one of [bootstrap (cluster), controlplane, nodepool]
- dkp update bootstrap credentials(see page 765) - Update credentials in the cluster

dkp update bootstrap credentials

Update credentials in the cluster

Options

```
  -h, --help   help for credentials
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp update bootstrap(see page 764) - Update bootstrap cluster
- dkp update bootstrap credentials aws(see page 765) - Update AWS credentials in the cluster and restart CAPA controllers
- dkp update bootstrap credentials azure(see page 766) - Update Azure credentials in the cluster and restart CAPZ controllers
- dkp update bootstrap credentials gcp(see page 766) - Update GCP credentials in the cluster and restart CAPG controllers
- dkp update bootstrap credentials vsphere(see page 767) - Update VSphere credentials in the cluster and restart CAPV controllers

dkp update bootstrap credentials aws

Update AWS credentials in the cluster and restart CAPA controllers

```
dkp update bootstrap credentials aws [flags]
```

Options

```
      --context string      The name of the kubeconfig context to use
  -h, --help                help for aws
      --kubeconfig string   Path to the kubeconfig for the management cluster. If
unspecified, default discovery rules apply.
      --print-only          Print the credentials and exit the function. Without
modifying cluster
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp update bootstrap credentials(see page 765) - Update credentials in the cluster

dkp update bootstrap credentials azure

Update Azure credentials in the cluster and restart CAPZ controllers

```
dkp update bootstrap credentials azure [flags]
```

Options

```
      --context string      The name of the kubeconfig context to use
  -h, --help                help for azure
      --kubeconfig string   Path to the kubeconfig for the management cluster. If
unspecified, default discovery rules apply.
      --print-only          Print the credentials and exit the function. Without
modifying cluster
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp update bootstrap credentials(see page 765) - Update credentials in the cluster

dkp update bootstrap credentials gcp

Update GCP credentials in the cluster and restart CAPG controllers

```
dkp update bootstrap credentials gcp [flags]
```

Options

```
      --context string      The name of the kubeconfig context to use
  -h, --help                help for gcp
      --kubeconfig string   Path to the kubeconfig for the management cluster. If
unspecified, default discovery rules apply.
      --print-only          Print the credentials and exit the function. Without
modifying cluster
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp update bootstrap credentials(see page 765) - Update credentials in the cluster

dkp update bootstrap credentials vsphere

Update VSphere credentials in the cluster and restart CAPV controllers

```
dkp update bootstrap credentials vsphere [flags]
```

Options

```
      --context string      The name of the kubeconfig context to use
  -h, --help                help for vsphere
      --kubeconfig string   Path to the kubeconfig for the management cluster. If
unspecified, default discovery rules apply.
      --print-only          Print the credentials and exit the function. Without
modifying cluster
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp update bootstrap credentials(see page 765) - Update credentials in the cluster

9.2.20 dkp upgrade

Upgrade one of [addons, capi-components, catalogapp, kommander, workspace]

9.2.20.1 Options

```
  -h, --help   help for upgrade
```

9.2.20.2 Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

9.2.20.3 SEE ALSO

- dkp upgrade addons(see page 771) - Upgrade the core Addons in a cluster, one of [aws, azure, eks, preprovisioned, vsphere]
- dkp upgrade capi-components(see page 769) - Upgrade the CAPI components in the cluster
- dkp upgrade catalogapp(see page 769) - Upgrade a Catalog Application to a newer version
- dkp upgrade kommander(see page 768) - Upgrade the Kommander version of the targeted cluster

- dkp upgrade workspace(see page 770) - Upgrade all platform applications in the given workspace and its projects to the same version as platform applications running on the management cluster

9.2.20.4 dkp upgrade kommander

Upgrade the Kommander version of the targeted cluster

Synopsis

Upgrades all Kommander components and platform applications running on the targeted cluster. No attached clusters and applications running on them are affected by this action.

```
dkp upgrade kommander [flags]
```

Options

```
      --charts-bundle stringArray                  Path to charts-bundle to upload to
chartmuseum, apart from parsing the kommander applications repository (default [])
      --config string                              Config file to use (default "/
root/.kommander/config")
      --context string                             The name of the kubeconfig context
to use
      --core-app-timeout duration                  Timeout to wait for upgrade of
each kommander core application (default 20m0s)
      --disable-appdeployments strings             List of AppDeployments to be
disabled during upgrade (default [fluent-bit])
      --disallow-charts-download                   make CLI rely solely on provided
chart bundles and do not try to download charts from the Internet
      --gitea-kommander-repository-name string     gitea kommander repository name
(default "kommander")
  -h, --help                                       help for kommander
      --kommander-applications-repository string   git repository with application
definitions (default "v2.3.3")
      --kommander-charts-version string            Kommander helm charts version to
download. Default: download all available versions
      --kubeconfig string                          Path to the kubeconfig file to use
for CLI requests.
      --platform-apps-timeout duration             Timeout to wait for upgrade of the
set of platform applications (default 30m0s)
      --request-timeout string                     The length of time to wait before
giving up on a single server request. Non-zero values should contain a corresponding
time unit (e.g. 1s, 2m, 3h). A value of zero means don't timeout requests.
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp upgrade(see page 767) - Upgrade one of [addons, capi-components, catalogapp, kommander, workspace]

9.2.20.5 dkp upgrade capi-components

Upgrade the CAPI components in the cluster

```
dkp upgrade capi-components [flags]
```

Options

```
  -h, --help                help for capi-components
      --kubeconfig string   Path to the kubeconfig for the management cluster. If
unspecified, default discovery rules apply.
      --timeout duration    The length of time to wait before giving up. Zero means
wait forever. (default 10m0s)
      --wait                If true, wait for operations to complete before
returning. (default true)
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp upgrade(see page 767) - Upgrade one of [addons, capi-components, catalogapp, kommander, workspace]

9.2.20.6 dkp upgrade catalogapp

Upgrade a Catalog Application to a newer version

```
dkp upgrade catalogapp CATALOGAPP_NAME --to-version VERSION [--workspace WORKSPACE |
--project PROJECT] [flags]
```

Options

```
      --config string                    Config file to use (default "/
root/.kommander/config")
      --context string                   The name of the kubeconfig context to use
```

```
      --core-app-timeout duration        Timeout to wait for upgrade of each
kommander core application (default 20m0s)
      --disable-appdeployments strings   List of AppDeployments to be disabled during
upgrade (default [fluent-bit])
  -h, --help                             help for catalogapp
      --kubeconfig string                Path to the kubeconfig file to use for CLI
requests.
      --platform-apps-timeout duration   Timeout to wait for upgrade of the set of
platform applications (default 30m0s)
      --project string                   Name of the Project to upgrade the Catalog
App in
      --request-timeout string           The length of time to wait before giving up
on a single server request. Non-zero values should contain a corresponding time unit
(e.g. 1s, 2m, 3h). A value of zero means don't timeout requests.
      --to-version string                Version the Catalog App should be upgraded
to
  -w, --workspace string                 Name of the Workspace to upgrade the Catalog
App in
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp upgrade(see page 767) - Upgrade one of [addons, capi-components, catalogapp, kommander, workspace]

9.2.20.7 dkp upgrade workspace

Upgrade all platform applications in the given workspace and its projects to the same version as platform applications running on the management cluster

```
dkp upgrade workspace WORKSPACE_NAME [--dry-run] [flags]
```

Options

```
      --config string                    Config file to use (default "/
root/.kommander/config")
      --context string                   The name of the kubeconfig context to use
      --core-app-timeout duration        Timeout to wait for upgrade of each
kommander core application (default 20m0s)
      --disable-appdeployments strings   List of AppDeployments to be disabled during
upgrade (default [fluent-bit])
      --dry-run                          Do not upgrade, just list the AppDeployments
that would be upgraded
  -h, --help                             help for workspace
```

```
      --kubeconfig string                Path to the kubeconfig file to use for CLI
requests.
      --platform-apps-timeout duration   Timeout to wait for upgrade of the set of
platform applications (default 30m0s)
      --request-timeout string           The length of time to wait before giving up
on a single server request. Non-zero values should contain a corresponding time unit
(e.g. 1s, 2m, 3h). A value of zero means don't timeout requests.
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp upgrade(see page 767) - Upgrade one of [addons, capi-components, catalogapp, kommander, workspace]

9.2.20.8 dkp upgrade addons

Upgrade the core Addons in a cluster, one of [aws, azure, eks, preprovisioned, vsphere]

Options

```
  -h, --help   help for addons
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp upgrade(see page 767) - Upgrade one of [addons, capi-components, catalogapp, kommander, workspace]
- dkp upgrade addons aws(see page 771) - Upgrade the core Addons in a AWS cluster
- dkp upgrade addons azure(see page 772) - Upgrade the core Addons in a Azure cluster
- dkp upgrade addons eks(see page 774) - Upgrade the core Addons in a EKS cluster
- dkp upgrade addons preprovisioned(see page 773) - Upgrade the core Addons in a Preprovisioned cluster
- dkp upgrade addons vsphere(see page 775) - Upgrade the core Addons in a vSphere cluster

dkp upgrade addons aws

Upgrade the core Addons in a AWS cluster

```
dkp upgrade addons aws [flags]
```

Options

```
      --allow-missing-template-keys   If true, ignore any errors in templates when a
field or map key is missing in the template. Only applies to golang and jsonpath
output formats. (default true)
  -c, --cluster-name name             Name used to prefix the cluster and all the
created resources.
      --dry-run                       Only print the objects that would be created,
without creating them.
  -h, --help                          help for aws
      --kubeconfig string             Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
  -n, --namespace string              If present, the namespace scope for this CLI
request. (default "default")
  -o, --output string                 Output format. One of: json|yaml|name|go-
template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|jsonpath-
file.
      --show-managed-fields           If true, keep the managedFields when printing
objects in JSON or YAML format.
      --template string               Template string or path to template file to use
when -o=go-template, -o=go-template-file. The template format is golang templates
[http://golang.org/pkg/text/template/#pkg-overview].
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp upgrade addons(see page 771) - Upgrade the core Addons in a cluster, one of [aws, azure, eks, preprovisioned, vsphere]

dkp upgrade addons azure

Upgrade the core Addons in a Azure cluster

```
dkp upgrade addons azure [flags]
```

Options

```
      --allow-missing-template-keys   If true, ignore any errors in templates when a
field or map key is missing in the template. Only applies to golang and jsonpath
output formats. (default true)
```

```
  -c, --cluster-name name             Name used to prefix the cluster and all the
created resources.
      --dry-run                       Only print the objects that would be created,
without creating them.
  -h, --help                          help for azure
      --kubeconfig string             Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
  -n, --namespace string              If present, the namespace scope for this CLI
request. (default "default")
  -o, --output string                 Output format. One of: json|yaml|name|go-
template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|jsonpath-
file.
      --show-managed-fields           If true, keep the managedFields when printing
objects in JSON or YAML format.
      --template string               Template string or path to template file to use
when -o=go-template, -o=go-template-file. The template format is golang templates
[http://golang.org/pkg/text/template/#pkg-overview].
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp upgrade addons(see page 771) - Upgrade the core Addons in a cluster, one of [aws, azure, eks, preprovisioned, vsphere]

dkp upgrade addons preprovisioned

Upgrade the core Addons in a Preprovisioned cluster

```
dkp upgrade addons preprovisioned [flags]
```

Options

```
      --allow-missing-template-keys   If true, ignore any errors in templates when a
field or map key is missing in the template. Only applies to golang and jsonpath
output formats. (default true)
  -c, --cluster-name name             Name used to prefix the cluster and all the
created resources.
      --dry-run                       Only print the objects that would be created,
without creating them.
  -h, --help                          help for preprovisioned
      --kubeconfig string             Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
```

```
  -n, --namespace string              If present, the namespace scope for this CLI
request. (default "default")
  -o, --output string                 Output format. One of: json|yaml|name|go-
template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|jsonpath-
file.
      --show-managed-fields           If true, keep the managedFields when printing
objects in JSON or YAML format.
      --template string               Template string or path to template file to use
when -o=go-template, -o=go-template-file. The template format is golang templates
[http://golang.org/pkg/text/template/#pkg-overview].
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp upgrade addons(see page 771) - Upgrade the core Addons in a cluster, one of [aws, azure, eks, preprovisioned, vsphere]

dkp upgrade addons eks

Upgrade the core Addons in a EKS cluster

```
dkp upgrade addons eks [flags]
```

Options

```
      --allow-missing-template-keys   If true, ignore any errors in templates when a
field or map key is missing in the template. Only applies to golang and jsonpath
output formats. (default true)
  -c, --cluster-name name             Name used to prefix the cluster and all the
created resources.
      --dry-run                       Only print the objects that would be created,
without creating them.
  -h, --help                          help for eks
      --kubeconfig string             Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
  -n, --namespace string              If present, the namespace scope for this CLI
request. (default "default")
  -o, --output string                 Output format. One of: json|yaml|name|go-
template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|jsonpath-
file.
      --show-managed-fields           If true, keep the managedFields when printing
objects in JSON or YAML format.
```

```
      --template string               Template string or path to template file to use
when -o=go-template, -o=go-template-file. The template format is golang templates
[http://golang.org/pkg/text/template/#pkg-overview].
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp upgrade addons(see page 771) - Upgrade the core Addons in a cluster, one of [aws, azure, eks, preprovisioned, vsphere]

dkp upgrade addons vsphere

Upgrade the core Addons in a vSphere cluster

```
dkp upgrade addons vsphere [flags]
```

Options

```
      --allow-missing-template-keys   If true, ignore any errors in templates when a
field or map key is missing in the template. Only applies to golang and jsonpath
output formats. (default true)
  -c, --cluster-name name             Name used to prefix the cluster and all the
created resources.
      --dry-run                       Only print the objects that would be created,
without creating them.
  -h, --help                          help for vsphere
      --kubeconfig string             Path to the kubeconfig for the management
cluster. If unspecified, default discovery rules apply.
  -n, --namespace string              If present, the namespace scope for this CLI
request. (default "default")
  -o, --output string                 Output format. One of: json|yaml|name|go-
template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|jsonpath-
file.
      --show-managed-fields           If true, keep the managedFields when printing
objects in JSON or YAML format.
      --template string               Template string or path to template file to use
when -o=go-template, -o=go-template-file. The template format is golang templates
[http://golang.org/pkg/text/template/#pkg-overview].
```

Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

SEE ALSO

- dkp upgrade addons(see page 771) - Upgrade the core Addons in a cluster, one of [aws, azure, eks, preprovisioned, vsphere]

9.2.21 dkp edit

Edit a resource on the server

9.2.21.1 Synopsis

Edit a resource from the default editor.

The edit command allows you to directly edit any API resource you can retrieve via the command-line tools. It will open the editor defined by your KUBE_EDITOR, or EDITOR environment variables, or fall back to 'vi' for Linux or 'notepad' for Windows. You can edit multiple objects, although changes are applied one at a time. The command accepts file names as well as command-line arguments, although the files you point to must be previously saved versions of resources.

```
dkp edit (RESOURCE/NAME | -f FILENAME)
```

9.2.21.2 Options

```
      --allow-missing-template-keys   If true, ignore any errors in templates when a
field or map key is missing in the template. Only applies to golang and jsonpath
output formats. (default true)
      --config string                 Config file to use (default "/root/.kommander/
config")
      --context string                The name of the kubeconfig context to use
      --field-manager string          Name of the manager used to track field
ownership. (default "kommander-cli")
  -f, --filename strings              Filename, directory, or URL to files to use to
edit the resource (default [])
  -h, --help                          help for edit
      --kubeconfig string             Path to the kubeconfig file to use for CLI
requests.
  -k, --kustomize string              Process the kustomization directory. This flag
can't be used together with -f or -R.
  -n, --namespace string              namespace of the resource (default "default")
```

```
  -o, --output string                 Output format. One of: (json, yaml, name, go-
template, go-template-file, template, templatefile, jsonpath, jsonpath-as-json,
jsonpath-file).
      --output-patch                  Output the patch if the resource is edited.
  -R, --recursive                     Process the directory used in -f, --filename
recursively. Useful when you want to manage related manifests organized within the
same directory.
      --request-timeout string        The length of time to wait before giving up on
a single server request. Non-zero values should contain a corresponding time unit
(e.g. 1s, 2m, 3h). A value of zero means don't timeout requests.
      --save-config                   If true, the configuration of current object
will be saved in its annotation. Otherwise, the annotation will be unchanged. This
flag is useful when you want to perform kubectl apply on this object in the future.
      --show-managed-fields           If true, keep the managedFields when printing
objects in JSON or YAML format.
      --subresource string            If specified, edit will operate on the
subresource of the requested object. Must be one of [status]. This flag is alpha and
may change in the future.
      --template string               Template string or path to template file to use
when -o=go-template, -o=go-template-file. The template format is golang templates
[http://golang.org/pkg/text/template/#pkg-overview].
      --validate string[="strict"]    Must be one of: strict (or true), warn, ignore
(or false).
                                            "true" or "strict" will use a schema to
validate the input and fail the request if invalid. It will perform server side
validation if ServerSideFieldValidation is enabled on the api-server, but will fall
back to less reliable client-side validation if not.
                                            "warn" will warn about unknown or
duplicate fields without blocking the request if server-side field validation is
enabled on the API server, and behave as "ignore" otherwise.
                                            "false" or "ignore" will not perform any
schema validation, silently dropping any unknown or duplicate fields. (default
"strict")
      --windows-line-endings          Defaults to the line ending native to your
platform.
```

9.2.21.3 Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

9.2.22 dkp version

Print version information

9.2.22.1 Synopsis

Print version information

```
dkp version [flags]
```

9.2.22.2 Options

```
  -h, --help            help for version
      --long            If true, print additional version information.
  -o, --output string   One of 'yaml' or 'json'.
```

9.2.22.3 Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

9.2.23 dkp cluster

Get cluster information

9.2.23.1 Options

```
      --config string            Config file to use (default "/root/.kommander/
config")
      --context string           The name of the kubeconfig context to use
  -h, --help                     help for cluster
      --kubeconfig string        Path to the kubeconfig file to use for CLI requests.
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
```

9.2.23.2 Options inherited from parent commands

```
  -v, --verbose int   Output verbosity
```

9.2.23.3 SEE ALSO

- dkp cluster type(see page 778) - Retrieve cluster type

9.2.23.4 dkp cluster type

Retrieve cluster type

```
dkp cluster type [flags]
```

Options

```
  -h, --help   help for type
```

Options inherited from parent commands

```
      --config string            Config file to use (default "/root/.kommander/
config")
      --context string           The name of the kubeconfig context to use
      --kubeconfig string        Path to the kubeconfig file to use for CLI requests.
      --request-timeout string   The length of time to wait before giving up on a
single server request. Non-zero values should contain a corresponding time unit (e.g.
1s, 2m, 3h). A value of zero means don't timeout requests.
  -v, --verbose int              Output verbosity
```

SEE ALSO

- dkp cluster(see page 778) - Get cluster information

10 Upgrade DKP

The DKP upgrade represents an important step of your environment's lifecycle, as it ensures that you are up-to-date with the latest features and can benefit from the most recent improvements, enhanced cluster management, and better performance. This section describes how to upgrade your air-gapped and non-air-gapped environment to the latest version of DKP compatible with the latest Kubernetes version.

10.1 Prerequisite

Check what version of DKP you have downloaded currently using cli command dkp version(see page 777).

10.1.1 Supported Upgrade Paths

	Upgrading from Release...															
...to Release		**2.1.0**	**2.1.1**	**2.1.2**	**2.1.3**	**2.1.4**	**2.2.0**	**2.2.1**	**2.2.2**	**2.2.3**	**2.3.0**	**2.3.1**	**2.3.2**	**2.3.3**	**2.4.0**	**2.4.1**	**2.5.0**
	2.2.0	Yes	Yes	No	No	No	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
	2.2.1	Yes	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
	2.2.2	Yes	Yes	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA	NA	NA	NA	NA
	2.2.3	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA	NA	NA	NA
	2.3.0	No	No	No	No	No	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA	NA	NA
	2.3.1	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA	NA
	2.3.2	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA
	2.3.3	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA
	2.4.0	No	No	No	No	No	No	No	No	No	Yes	Yes	Yes	Yes	NA	NA	NA

	2.4.1	No	No	No	No	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	NA	NA
	2.5.0	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes	Yes	NA
	2.5.1	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes

10.1.2 Understand the Upgrade Process

For this release, you perform the upgrade sequentially beginning with the DKP UI and then moving to upgrading clusters and CAPI components.

When upgrading DKP, the process is different depending on whether you run a stand-alone Management Cluster, or a multi-cluster environment that includes a combination of a Management cluster and managed or attached workspace clusters.

Start with your Management Cluster in the UI, and then, if more than one exists, proceed workspace by workspace until complete. You can then move to upgrading Konvoy, cluster by cluster.

The overall process for upgrading to the latest version of DKP is done on each Workspace or cluster, with the following processes:

For **Kommander**, on your Management Cluster:

1. **Upgrade DKP UI** (see page 782)and all Platform Applications.
   If you do not have any managed or attached clusters, skip to upgrading Konvoy on your Management Cluster.

On your Workspaces (which include Management Cluster and managed or attached clusters):

1. **Upgrade your Workspaces**(see page 770), which upgrades all Platform Applications on your managed or attached workspace clusters.
2. **Upgrade all DKP Catalog applications**(see page 450) deployed to Workspaces.
3. **Upgrade all DKP Catalog applications**(see page 481) deployed to Projects.
4. **Verify any Custom Catalog applications**(see page 798) and ensure they are compatible with the Kubernetes version included in the new release(see page 811).

For **Konvoy**, on your Management Cluster:

1. **Upgrade CAPI components**(see page 785) (Essential or Enterprise section). This upgrades the CAPI controllers, which only run on the Management Cluster.
2. **Upgrade the Core Addons**(see page 785) (Essential or Enterprise section). This upgrades multiple addons such as CSI, CNI, Cluster Autoscaler, and Node Feature Discovery.
3. **Upgrade the Kubernetes version**(see page 785) (Essential or Enterprise section). This upgrades your cluster's control plane and node pools.
   If you do not have any managed or attached clusters, you have finished the upgrade process and can start testing your environment. If you have managed or attached clusters, continue with the next section.

For **Konvoy**, on your Managed Clusters:

1. **Upgrade the Core Addons**(see page 0). This upgrades multiple addons such as CSI, CNI, Cluster Autoscaler, and Node Feature Discovery.

2. **Upgrade the Kubernetes version**(see page 0). This upgrades your cluster's control plane and node pools. We recommend you upgrade your Kubernetes version on any attached clusters.

10.2 Upgrade Kommander

This section describes how to upgrade your Kommander Management cluster and all Platform Applications to their supported versions in networked, air-gapped, and on-prem environments. To prevent compatibility issues, you must first upgrade Kommander on your Management Cluster before upgrading to DKP.

It is important you upgrade Kommander BEFORE upgrading the Kubernetes version (or Konvoy version for Managed Konvoy clusters) in attached clusters, due to the previous versions' incompatibility with 1.22.

Page Contents

10.2.1 Prerequisites

- **REQUIRED** Before upgrading, create an on-demand backup(see page 552) of your current configuration with Velero.
- Download(see page 83) and install the supported DKP CLI binary of this release(see page 811) on your computer.
- Ensure you are on DKP version 2.2 or higher and Kubernetes version 1.22 or higher.

If you are on DKP 2.1 or older, you must upgrade to DKP 2.2 before upgrading to 2.3.

- If you have attached clusters, ensure they are on Kubernetes versions 1.22 or higher.
- Review the Platform Application version updates(see page 815) that are part of this upgrade.
- For air-gapped environments **with** DKP Catalog Applications in a multi-cluster environment: Load the Docker images into your Docker registry(see page 0)
- For air-gapped environments **without** DKP Catalog Applications: Load the Docker images into your Docker registry(see page 0)
- For air-gapped environments only:

Download the Kommander application definitions:

```
wget "https://downloads.d2iq.com/dkp/v2.3.3/kommander-applications-v2.3.3.tar.gz"
```

Download the Kommander charts bundle:

```
wget "https://downloads.d2iq.com/dkp/v2.3.3/dkp-kommander-charts-bundle-
v2.3.3.tar.gz" -O - | tar -xzvf -
```

If you have DKP Catalog Applications:

- Download the DKP Catalog Application charts bundle:

```
wget "https://downloads.d2iq.com/dkp/v2.3.3/dkp-catalog-applications-charts-
bundle-v2.3.3.tar.gz" -O - | tar -xzvf -
```

- Download the DKP Catalog Application Git tarball:

```
wget "https://downloads.d2iq.com/dkp/v2.3.3/dkp-catalog-applications-
v2.3.3.tar.gz"
```

10.2.2 Upgrade Kommander

Before running the following command, ensure that your `dkp` configuration **references the Kommander Management cluster**, otherwise it attempts to run the upgrade on the bootstrap cluster. You can do this by setting the `KUBECONFIG` environment variable to the appropriate kubeconfig file's location[637], or by using the `--kubeconfig=cluster_name.conf` flag.

⚠

- If you have configured a **custom domain**, running the `upgrade command` could result in an inaccessibility of your services via your custom domain for a few minutes.
- The Fluentbit application, which is responsible for collecting admin level logs, is automatically disabled on upgrade to v2.3.3. To keep Fluentbit enabled on your clusters, you will need to pass in the new `--disable-appdeployment` flag set to `""`. You must ensure there is sufficient storage for these logs if you keep it enabled. See Fluent Bit(see page 579) for more information.

1. Use the DKP CLI to upgrade Kommander and all the Platform Applications in the Management Cluster:
   a. For **air-gapped** environments:

   ```
   dkp upgrade kommander --charts-bundle dkp-kommander-charts-bundle-v2.3.3.t
   ar.gz --kommander-applications-repository kommander-applications-v2.3.3.ta
   r.gz
   ```

   b. For **air-gapped** environments with **DKP Catalog Applications**:

   ```
   dkp upgrade kommander --charts-bundle dkp-kommander-charts-bundle-v2.3.3.t
   ar.gz --charts-bundle dkp-catalog-applications-charts-bundle-v2.3.3.tar.gz
   --kommander-applications-repository kommander-applications-v2.3.3.tar.gz
   ```

   After the upgrade, if you have DKP Catalog Applications deployed, update the GitRepository for `dkp-catalog-applications` int the Kommander Installer Configuration.

   ⚠⚠⚠ For the following section, ensure you modify the **most recent** `kommander.yaml` configuration file. It must be the file that reflects the current state of your environment. Reinstalling Kommander with an outdated `kommander.yaml` overwrites the list of platform applications that are currently running in your cluster. ⚠⚠⚠

637 https://kubernetes.io/docs/tasks/access-application-cluster/configure-access-multiple-clusters/

i. In the `kommander.yaml`, update the DKP Catalog Applications by setting the correct DKP version:

```
apiVersion: config.kommander.mesosphere.io/v1alpha1
kind: Installation
...
# The list of enabled/disabled apps here should reflect the current
state of the environment, including configuration overrides!
...
catalog:
  repositories:
    - name: dkp-catalog-applications
      labels:
        kommander.d2iq.io/project-default-catalog-repository: "true"
        kommander.d2iq.io/workspace-default-catalog-repository:
"true"
        kommander.d2iq.io/gitapps-gitrepository-type: "dkp"
      path: ./dkp-catalog-applications-v2.3.3.tar.gz # modify this
 version to match the DKP upgrade version
```

ii. Refresh the `kommander.yaml` to apply the updated tarball:

⚠️ Before running this command, ensure the `kommander.yaml` is the configuration file you are currently using for your environment. Otherwise, your previous configuration will be lost. ⚠️

```
dkp install kommander --installer-config kommander.yaml
```

c. For **non-air-gapped** environments:

```
dkp upgrade kommander
```

After the upgrade, if you have DKP Catalog Applications deployed, update the GitRepository for `dkp-catalog-applications`.

i. Update the GitRepository with the tag of your updated DKP version on the `kommander` workspace:

```
kubectl patch gitrepository -n kommander dkp-catalog-applications --
type merge --patch '{"spec":{"ref":{"tag":"v2.3.3"}}}'
```

ℹ️ This command updates the catalog application repositories for all workspaces.

ii. For any additional workspaces created outside the `kommander.yaml` configuration:

Set the `WORKSPACE_NAMESPACE` environment variable to the namespace of the workspace:

```
export WORKSPACE_NAMESPACE=<workspace namespace>
```

iii. Update the GitRepository for the additional workspace:

```
kubectl patch gitrepository -n ${WORKSPACE_NAMESPACE} dkp-catalog-
applications --type merge --patch '{"spec":{"ref":
{"tag":"v2.3.3"}}}'
```

2. For **air-gapped** deployments, an additional step is required to upgrade the Grafana Loki MinIO Tenant:

```
kubectl patch statefulset grafana-loki-minio-ss-0 -n kommander --type='json'
 -p='[{"op": "replace", "path": "/spec/template/spec/containers/0/image",
"value":"quay.io/minio/minio:RELEASE.2022-01-08T03-11-54Z"}]'
```

3. If the upgrade fails, run the following command to get more information on the upgrade process:

```
dkp upgrade kommander -v 6
```

If you find any `HelmReleases` in a “broken” release state such as “exhausted” or “another rollback/release in progress”, you can trigger a reconciliation of the `HelmRelease` using the following commands:

```
kubectl -n kommander patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[{"
op": "replace", "path": "/spec/suspend", "value": true}]'
kubectl -n kommander patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[{"
op": "replace", "path": "/spec/suspend", "value": false}]'
```

4. For **Enterprise customers**(see page 87) (multi-cluster environment): Upgrade your additional Workspaces(see page 422) on a per-Workspace basis to upgrade the Platform Applications on other clusters than the Management Cluster.
For **Essential customers**(see page 89) (single-cluster environment): Proceed with the Konvoy Upgrade(see page 785).

You can always go back to the DKP Upgrade overview(see page 0), to review the next steps depending on your environment and license type.

10.3 Upgrade Konvoy

10.3.1 Steps to upgrade Konvoy via CLI

This section describes how to upgrade your DKP clusters to the latest Konvoy version. To prevent compatibility issues, review and ensure you are following the correct upgrade process for customers with a single-cluster or multi-cluster experience.

- DKP Enterprise Upgrade(see page 786)
- DKP Essential Upgrade(see page 793)

10.3.2 DKP Enterprise Upgrade

ENTERPRISE

Upgrade your Konvoy environment within the DKP Enterprise license.

10.3.2.1 Prerequisites

- Create an on-demand backup(see page 552) of your current configuration with Velero.
- Follow the steps listed in the DKP upgrade overview(see page 0).
- Check what version of DKP you have downloaded currently using cli command dkp version(see page 777).
- Ensure that all platform applications in the management cluster have been upgraded to avoid compatibility issues with the Kubernetes version(see page 811) included in this release. This is done automatically when upgrading Kommander(see page 782), so ensure that you upgrade Kommander prior to upgrading Konvoy.
- For air-gapped: Download the required bundles either at our support site[638] or by using the CLI.
- For Azure, set the required environment variables(see page 41).
- For AWS and EKS(see page 140), set the required environment variables(see page 37).
- For vSphere, set the required environment variables(see page 259).

The following infrastructure environments are supported:

- Amazon Web Services (AWS)
- Microsoft Azure
- Pre-provisioned environments
- vSphere
- EKS

10.3.2.2 Overview

To upgrade Konvoy for DKP Enterprise:

1. Upgrade the Cluster API (CAPI) components
2. Upgrade the core addons
3. Upgrade the Kubernetes version

Run all three steps on the management cluster (Kommander cluster) first. Then, run the second and third steps on additional managed clusters (Konvoy clusters), one cluster at a time, using the KUBECONFIG for the management cluster and specifying the name of the managed cluster(s) to upgrade.

For a full list of DKP Enterprise features, see DKP Enterprise(see page 87).

- For pre-provisioned air-gapped environments, you must run `konvoy-image upload artifacts`.
- You must maintain your attached clusters manually. Review the documentation from your cloud provider for more information.

638 https://support.d2iq.com/hc/en-us

10.3.2.3 Upgrade the CAPI components

New versions of DKP come pre-bundled with newer versions of CAPI, newer versions of infrastructure providers, or new infrastructure providers. When using a new version of the DKP CLI, upgrade all of these components first.

If you are running on more than one management cluster (Kommander cluster), you must upgrade the CAPI components on each of these clusters.

⚠ Ensure your `dkp` configuration references the management cluster where you want to run the upgrade by setting the `KUBECONFIG` environment variable, or using the `--kubeconfig` flag, in accordance with Kubernetes conventions[639].

Run the following upgrade command for the CAPI components.

```
dkp upgrade capi-components
```

The output resembles the following:

```
✓ Upgrading CAPI components
✓ Waiting for CAPI components to be upgraded
✓ Initializing new CAPI components
✓ Deleting Outdated Global ClusterResourceSets
```

If the upgrade fails, review the prerequisites section and ensure that you've followed the steps in the DKP upgrade overview[640].

10.3.2.4 Upgrade the core addons

To install the core addons, DKP relies on the `ClusterResourceSet` Cluster API feature[641]. In the CAPI component upgrade, we deleted the previous set of outdated global `ClusterResourceSets` because prior to DKP 2.2 some addons were installed using a global configuration. In order to support individual cluster upgrades, DKP 2.2 now installs all addons with a unique set of `ClusterResourceSet` and corresponding referenced resources, all named using the cluster's name as a suffix. For example: `calico-cni-installation-my-aws-cluster`.

⚠ If you have modified any of the `clusterResourceSet` definitions, these changes will **not** be preserved when running the command `dkp upgrade addons`. You must use the `--dry-run -o yaml` options to save the new configuration to a file and remake the same changes upon each upgrade.

639 https://kubernetes.io/docs/tasks/access-application-cluster/configure-access-multiple-clusters/
640 https://d2iq.atlassian.net/wiki/pages/resumedraft.action?draftId=89033677
641 https://cluster-api.sigs.k8s.io/

Your cluster comes preconfigured with a few different core addons that provide functionality to your cluster upon creation. These include: CSI, CNI, Cluster Autoscaler, and Node Feature Discovery. New versions of DKP may come pre-bundled with newer versions of these addons. Perform the following steps to update these addons. If you have any additional managed clusters, you will need to upgrade the core addons and Kubernetes version for each one.

⚠ Ensure your `dkp` configuration references the management cluster where you want to run the upgrade by setting the `KUBECONFIG` environment variable, or using the `--kubeconfig` flag, in accordance with Kubernetes conventions[642].

Upgrade the core addons in a cluster using the `dkp upgrade addons` command specifying the cluster infrastructure (choose `aws`, `azure`, `vsphere`, `eks`, `preprovisioned`) and the name of the cluster.

Examples:

```
export CLUSTER_NAME=my-azure-cluster
dkp upgrade addons azure --cluster-name=${CLUSTER_NAME}
```

OR

```
export CLUSTER_NAME=my-aws-cluster
dkp upgrade addons aws --cluster-name=${CLUSTER_NAME}
```

The output for the AWS example should be similar to:

```
Generating addon resources
clusterresourceset.addons.cluster.x-k8s.io/calico-cni-installation-my-aws-cluster
upgraded
configmap/calico-cni-installation-my-aws-cluster upgraded
clusterresourceset.addons.cluster.x-k8s.io/tigera-operator-my-aws-cluster upgraded
configmap/tigera-operator-my-aws-cluster upgraded
clusterresourceset.addons.cluster.x-k8s.io/aws-ebs-csi-my-aws-cluster upgraded
configmap/aws-ebs-csi-my-aws-cluster upgraded
clusterresourceset.addons.cluster.x-k8s.io/cluster-autoscaler-my-aws-cluster upgraded
configmap/cluster-autoscaler-my-aws-cluster upgraded
clusterresourceset.addons.cluster.x-k8s.io/node-feature-discovery-my-aws-cluster
upgraded
configmap/node-feature-discovery-my-aws-cluster upgraded
clusterresourceset.addons.cluster.x-k8s.io/nvidia-feature-discovery-my-aws-cluster
upgraded
configmap/nvidia-feature-discovery-my-aws-cluster upgraded
```

See also

DKP upgrade addons(see page 771)

642 https://kubernetes.io/docs/tasks/access-application-cluster/configure-access-multiple-clusters/

Once complete, begin upgrading the Kubernetes version.

10.3.2.5 Upgrade the Kubernetes version

When upgrading the Kubernetes version of a cluster, first upgrade the control plane and then the node pools. If you have any additional managed or attached clusters, you will need to upgrade the core addons and Kubernetes version for each one.

1. Build a new image if applicable.
    - If an AMI was specified when initially creating a cluster for AWS, you must build a new one with Konvoy Image Builder(see page 324).
    - If an Azure Machine Image was specified for Azure, you must build a new one with Konvoy Image Builder[643].
    - If a vSphere template Image was specified for vSphere, you must build a new one with Konvoy Image Builder(see page 256).
2. Upgrade the Kubernetes version of the control plane. Each cloud provider has distinctive commands. Below is the AWS command example. Select the drop-down menu next to your provider for compliant CLI.

```
dkp update controlplane aws --cluster-name=${CLUSTER_NAME} --kubernetes-
version=v1.23.12
```

Azure

```
dkp update controlplane azure --cluster-name=${CLUSTER_NAME} --kubernetes-
version=v1.23.12 --compute-gallery-id <Azure Compute Gallery built by KIB for
 Kubernetes v1.23.12>
```

vSphere

```
dkp update controlplane vsphere --cluster-name=${CLUSTER_NAME} --kubernetes-
version=v1.23.12 --vm-template <vSphere template built by KIB for Kubernetes v1.23.12>
```

Pre-provisioned

```
dkp update controlplane preprovisioned --cluster-name=${CLUSTER_NAME} --kubernetes-
version=v1.23.12
```

The output should be similar to the below example, with the provider name corresponding to the one you entered in the command line above:

```
Updating control plane resource controlplane.cluster.x-k8s.io/v1beta1,
Kind=KubeadmControlPlane default/my-aws-cluster-control-plane
Waiting for control plane update to finish.
 ✓ Updating the control plane
```

643 https://d2iq.atlassian.net/wiki/spaces/DENT/pages/88411008/%282.3%29+Create+a+custom+Azure+Image

If upgrading a FIPS cluster, there is a bug in the upgrade of `kube-proxy` `DaemonSet` in that it doesn't get automatically upgraded. To correctly upgrade, run the command shown below:

```
kubectl set image -n kube-system daemonset.v1.apps/kube-proxy kube-proxy=docker.io/
mesosphere/kube-proxy:v1.23.12_fips.0
```

3. Upgrade the Kubernetes version of your node pools. Upgrading a node pool involves draining the existing nodes in the node pool and replacing them with new nodes. In order to ensure minimum downtime and maintain high availability of the critical application workloads during the upgrade process, we recommend deploying Pod Disruption Budget (Disruptions[644]) for your critical applications. For more information, refer to Update Cluster Nodepools[645] documentation.

- First, get a list of all node pools available in your cluster by running the following command:

```
dkp get nodepool --cluster-name ${CLUSTER_NAME}
```

- Select the nodepool you want to upgrade with the command below:

```
export NODEPOOL_NAME=my-nodepool
```

- Then update the selected nodepool using the command below. The first example command shows AWS language, so select the drop-down menu for your provider for the correct command. Execute the `update` command for each of the node pools listed in the previous command.

```
dkp update nodepool aws ${NODEPOOL_NAME} --cluster-name=${CLUSTER_NAME} --kubernetes-
version=v1.23.12
```

Azure

```
dkp update nodepool azure ${NODEPOOL_NAME} --cluster-name=${CLUSTER_NAME} --
kubernetes-version=v1.23.12 --compute-gallery-id <Azure Compute Gallery built by KIB
for Kubernetes v1.23.12>
```

vSphere

```
dkp update nodepool vsphere ${NODEPOOL_NAME} --cluster-name=${CLUSTER_NAME} --
kubernetes-version=v1.23.12 --vm-template <vSphere template built by KIB for
 Kubernetes v1.23.12>
```

Pre-provisioned

644 https://kubernetes.io/docs/concepts/workloads/pods/disruptions/
645 https://d2iq.atlassian.net/wiki/spaces/DENT/pages/113934685

```
dkp update nodepool preprovisioned ${NODEPOOL_NAME} --cluster-name=${CLUSTER_NAME} --
kubernetes-version=v1.23.12
```

The output should be similar to the following, with the name of the infrastructure provider shown accordingly:

```
Updating node pool resource cluster.x-k8s.io/v1beta1, Kind=MachineDeployment default/
my-aws-cluster-my-nodepool
Waiting for node pool update to finish.
 ✓ Updating the my-aws-cluster-my-nodepool node pool
```

Additional Considerations for upgrading a FIPS cluster:

If upgrading a FIPS cluster, to correctly upgrade the Kubernetes version, instead run the command shown below:

```
dkp update nodepool aws ${NODEPOOL_NAME} --cluster-name=${CLUSTER_NAME} --kubernetes-
version=v1.23.12+fips.0 --ami=<ami-with-fips-id> --etcd-version=v3.4.13+fips.0
```

NOTE: Some advanced options are available for various providers. An example would be regarding AMI instance type: `aws: --ami, --instance-type`. To see all the options for your particular provider, run this command `dkp update controlplane aws|vsphere|preprovisioned|azure|eks --help` for more advance options.

Repeat this step for each additional node pool.Once all nodepools have been updated, your upgrade is complete. For the overall process for upgrading to the latest version of DKP, refer back to DKP Upgrade(see page 0).

10.3.2.6 Upgrade Managed Clusters

If you have managed clusters, follow these steps to upgrade each cluster:

1. Using the kubeconfig of your management cluster, find your cluster name and be sure to copy the information for all of your clusters:

   ```
   kubectl get clusters -A
   ```

2. Set your cluster variable:

   ```
   export CLUSTER_NAME=<your-managed-cluster-name>
   ```

3. Set your cluster's workspace variable:

```
export CLUSTER_WORKSPACE=<your-workspace-namespace>
```

4. Then, upgrade the core addons (replacing `aws` with whatever infrastructure provider you would be using):

```
dkp upgrade addons aws --cluster-name=${CLUSTER_NAME} -n ${CLUSTER_WORKSPACE}
```

5. Check to see if you have any cluster resource sets that need to be cleaned up:

```
kubectl get clusterresourcesets -n ${CLUSTER_WORKSPACE}
```

6. Delete the ClusterResourceSet for nvidia-feature-discovery by running:

```
kubectl delete clusterresourceset nvidia-feature-discovery-my-aws-cluster -n $
{CLUSTER_WORKSPACE}
```

7. Delete ConfigMap ClusterResourceSet referred to by running the following command, ensure you use using `nvidia-feature-discover-${CLUSTER_NAME}`. If there is no related ConfigMap, then you can move on to the next step.

```
kubectl delete configmap nvidia-feature-discovery-my-aws-cluster -n $
{CLUSTER_WORKSPACE}
```

8. Get the kubeconfig for the managed cluster by running:

```
dkp get kubeconfig -c ${CLUSTER_NAME} -n ${CLUSTER_WORKSPACE} >> $
{CLUSTER_NAME}.conf
```

9. Delete the corresponding daemonset on the remote cluster by running. If there is no related daemonset, then you can move on to the next step.

```
kubectl --kubeconfig=${CLUSTER_NAME}.conf delete daemonset nvidia-feature-
discovery-gpu-feature-discovery -n node-feature-discovery
```

Upgrade Kubernetes Version on a Managed Cluster

After you complete the previous steps for all managed clusters and you update your core addons, begin upgrading the Kubernetes version.

You should first complete the upgrade of your Kommander Management Cluster before upgrading any managed clusters.

1. Use this command to start upgrading the Kubernetes version:

```
dkp update controlplane aws --cluster-name=${CLUSTER_NAME} --kubernetes-
version=v1.23.12 -n ${CLUSTER_WORKSPACE}
```

2. Get a list of all node pools available in your cluster by running the following command:

```
dkp get nodepools -c ${CLUSTER_NAME} -n ${CLUSTER_WORKSPACE}

export NODEPOOL_NAME=<my-nodepool>
```

3. Use this command to upgrade the node pools:

```
dkp update nodepool aws ${NODEPOOL_NAME} --cluster-name=${CLUSTER_NAME} --
kubernetes-version=v1.23.12 -n ${CLUSTER_WORKSPACE}
```

10.3.3 DKP Essential Upgrade

Upgrade your Konvoy environment within the DKP Essential License.

10.3.3.1 Prerequisites

- Create an on-demand backup(see page 552) of your current configuration with Velero.
- Follow the steps listed in the DKP upgrade overview(see page 0).
- Check what version of DKP you have downloaded currently using cli command dkp version(see page 777).
- Ensure that all platform applications in the management cluster have been upgraded to avoid compatibility issues with the Kubernetes version(see page 811) included in this release. This is done automatically when upgrading Kommander(see page 782), so ensure that you upgrade Kommander prior to upgrading Konvoy.
- For air-gapped: Download the required bundles either at our support site[646] or by using the CLI.
- For Azure, set the required environment variables(see page 41).
- For AWS, set the required environment variables(see page 37).
- For vSphere, set the required environment variables(see page 259).

The following infrastructure environments are supported:

- Amazon Web Services (AWS)
- Microsoft Azure
- Pre-provisioned environments
- vSphere

10.3.3.2 Overview

To upgrade Konvoy for DKP Essential:

1. Upgrade the Cluster API (CAPI) components
2. Upgrade the core addons
3. Upgrade the Kubernetes version

If you have more than one Essential cluster, repeat all of these steps for each Essential cluster (management cluster).

646 https://support.d2iq.com/hc/en-us

For pre-provisioned air-gapped environments, you must run `konvoy-image upload artifacts`.

For a full list of DKP Essential features, see DKP Essential(see page 798).

Upgrade the CAPI components

New versions of DKP come pre-bundled with newer versions of CAPI, newer versions of infrastructure providers or new infrastructure providers. When using a new version of the DKP CLI, upgrade all of these components first.

Ensure your `dkp` configuration references the management cluster where you want to run the upgrade by setting the `KUBECONFIG` environment variable, or using the `--kubeconfig` flag, in accordance with Kubernetes conventions[647].

Run the following upgrade command for the CAPI components.

```
dkp upgrade capi-components
```

The command should output something similar to the following:

```
✓ Upgrading CAPI components
✓ Waiting for CAPI components to be upgraded
✓ Initializing new CAPI components
✓ Deleting Outdated Global ClusterResourceSets
```

If the upgrade fails, review the prerequisites section and ensure that you've followed the steps in the DKP upgrade overview(see page 0).

10.3.3.3 Upgrade the core addons

To install the core addons, DKP relies on the `ClusterResourceSet` Cluster API feature[648]. In the CAPI component upgrade, we deleted the previous set of outdated global `ClusterResourceSets` because prior to DKP 2.2 some addons were installed using a global configuration. In order to support individual cluster upgrades, DKP 2.2 now installs all addons with a unique set of `ClusterResourceSet` and corresponding referenced resources, all named using the cluster's name as a suffix. For example: `calico-cni-installation-my-aws-cluster`.

If you modified any of the `clusterResourceSet` definitions, these changes are **not** preserved when running the command `dkp upgrade addons`. You can use the `--dry-run -o yaml` options to save the new configuration to a file and make the same changes again upon each of the upgrades.

647 https://kubernetes.io/docs/tasks/access-application-cluster/configure-access-multiple-clusters/
648 https://cluster-api.sigs.k8s.io/

Your cluster comes preconfigured with a few different core addons that provide functionality to your cluster upon creation. These include: CSI, CNI, Cluster Autoscaler, and Node Feature Discovery. New versions of DKP may come pre-bundled with newer versions of these addons. Perform the following steps to update these addons.

⚠ If you have more than one essential cluster, ensure your `dkp` configuration references the management cluster where you want to run the upgrade by setting the `KUBECONFIG` environment variable, or using the `--kubeconfig` flag, in accordance with Kubernetes conventions[649].

Upgrade the core addons in a cluster using the `dkp upgrade addons` command specifying the cluster infrastructure (choose [`aws` , `azure` , `vsphere` , `eks` , `preprovisioned`]) and the name of the cluster.

Examples:

```
export CLUSTER_NAME=my-azure-cluster
dkp upgrade addons azure --cluster-name=${CLUSTER_NAME}
```

OR

```
export CLUSTER_NAME=my-aws-cluster
dkp upgrade addons aws --cluster-name=${CLUSTER_NAME}
```

The output for the AWS example should be similar to:

```
Generating addon resources
clusterresourceset.addons.cluster.x-k8s.io/calico-cni-installation-my-aws-cluster
upgraded
configmap/calico-cni-installation-my-aws-cluster upgraded
clusterresourceset.addons.cluster.x-k8s.io/tigera-operator-my-aws-cluster upgraded
configmap/tigera-operator-my-aws-cluster upgraded
clusterresourceset.addons.cluster.x-k8s.io/aws-ebs-csi-my-aws-cluster upgraded
configmap/aws-ebs-csi-my-aws-cluster upgraded
clusterresourceset.addons.cluster.x-k8s.io/cluster-autoscaler-my-aws-cluster upgraded
configmap/cluster-autoscaler-my-aws-cluster upgraded
clusterresourceset.addons.cluster.x-k8s.io/node-feature-discovery-my-aws-cluster
upgraded
configmap/node-feature-discovery-my-aws-cluster upgraded
clusterresourceset.addons.cluster.x-k8s.io/nvidia-feature-discovery-my-aws-cluster
upgraded
configmap/nvidia-feature-discovery-my-aws-cluster upgraded
```

See also

DKP upgrade addons(see page 771)

Once complete, begin upgrading the Kubernetes version.

649 https://kubernetes.io/docs/tasks/access-application-cluster/configure-access-multiple-clusters/

10.3.3.4 Upgrade the Kubernetes version

When upgrading the Kubernetes version of a cluster, first upgrade the control plane and then the node pools.

If you have any additional managed or attached clusters, you will need to upgrade the core addons and Kubernetes version for each one.

1. Build a new image if applicable.
    - If an AMI was specified when initially creating a cluster for AWS, you must build a new one with Konvoy Image Builder(see page 324).
    - If an Azure Machine Image was specified for Azure, you must build a new one with Konvoy Image Builder[650].
    - If a vSphere template Image was specified for vSphere, you must build a new one with Konvoy Image Builder(see page 256).
2. Upgrade the Kubernetes version of the control plane. Each cloud provider has distinctive commands. Below is the AWS command example. Select the drop-down menu next to your provider for compliant CLI.

```
dkp update controlplane aws --cluster-name=${CLUSTER_NAME} --kubernetes-
version=v1.23.12
```

Azure

```
dkp update controlplane azure --cluster-name=${CLUSTER_NAME} --kubernetes-
version=v1.23.12 --compute-gallery-id <Azure Compute Gallery built by KIB for
 Kubernetes v1.23.12>
```

vSphere

```
dkp update controlplane vsphere --cluster-name=${CLUSTER_NAME} --kubernetes-
version=v1.23.12 --vm-template <vSphere template built by KIB for Kubernetes v1.23.12>
```

Pre-provisioned

```
dkp update controlplane preprovisioned --cluster-name=${CLUSTER_NAME} --kubernetes-
version=v1.23.12
```

The output should be similar to the below example, with the provider name corresponding to the CLI you executed from the choices above:

```
Updating control plane resource controlplane.cluster.x-k8s.io/v1beta1,
Kind=KubeadmControlPlane default/my-aws-cluster-control-plane
Waiting for control plane update to finish.
 ✓ Updating the control plane
```

650 https://d2iq.atlassian.net/wiki/spaces/DENT/pages/88411008/%282.3%29+Create+a+custom+Azure+Image

If upgrading a FIPS cluster, there is a bug in the upgrade of `kube-proxy` `DaemonSet` in that it doesn't get automatically upgraded. To correctly upgrade, run the command shown below:

```
kubectl set image -n kube-system daemonset.v1.apps/kube-proxy kube-proxy=docker.io/
mesosphere/kube-proxy:v1.23.12_fips.0
```

3. Upgrade the Kubernetes version of your node pools. Upgrading a node pool involves draining the existing nodes in the node pool and replacing them with new nodes. In order to ensure minimum downtime and maintain high availability of the critical application workloads during the upgrade process, we recommend deploying Pod Disruption Budget (Disruptions[651]) for your critical applications. For more information, refer to Update Cluster Nodepools[652] documentation.

- First, get a list of all node pools available in your cluster by running the following command:

```
dkp get nodepool --cluster-name ${CLUSTER_NAME}
```

- Select the nodepool you want to upgrade with the command below:

```
export NODEPOOL_NAME=my-nodepool
```

- Then update the selected nodepool using the command below. The first example command shows AWS language, so select the drop-down menu for your provider for the correct command. Execute the `update` command for each of the node pools listed in the previous command.

```
dkp update nodepool aws ${NODEPOOL_NAME} --cluster-name=${CLUSTER_NAME} --kubernetes-
version=v1.23.12
```

Azure

```
dkp update nodepool azure ${NODEPOOL_NAME} --cluster-name=${CLUSTER_NAME} --
kubernetes-version=v1.23.12 --compute-gallery-id <Azure Compute Gallery built by KIB
for Kubernetes v1.23.12>
```

vSphere

```
dkp update nodepool vsphere ${NODEPOOL_NAME} --cluster-name=${CLUSTER_NAME} --
kubernetes-version=v1.23.12 --vm-template <vSphere template built by KIB for
 Kubernetes v1.23.12>
```

Pre-provisioned

651 https://kubernetes.io/docs/concepts/workloads/pods/disruptions/
652 https://d2iq.atlassian.net/wiki/spaces/DENT/pages/113934685

```
dkp update nodepool preprovisioned ${NODEPOOL_NAME} --cluster-name=${CLUSTER_NAME} --
kubernetes-version=v1.23.12
```

The output should be similar to the following, with the name of the infrastructure provider shown accordingly:

```
Updating node pool resource cluster.x-k8s.io/v1beta1, Kind=MachineDeployment default/
my-aws-cluster-my-nodepool
Waiting for node pool update to finish.
 ✓ Updating the my-aws-cluster-my-nodepool node pool
```

Repeat this step for each additional node pool.

Additional Considerations for upgrading a FIPS cluster:

If upgrading a FIPS cluster, to correctly upgrade the Kubernetes version, instead run the command shown below:

```
dkp update nodepool aws ${NODEPOOL_NAME} --cluster-name=${CLUSTER_NAME} --kubernetes-
version=v1.24.6+fips.0 --ami=<ami-with-fips-id>
```

NOTE: Some advanced options are available for various providers. An example would be regarding AMI instance type: `aws: --ami, --instance-type`. To see all the options for your particular provider, run this command `dkp update controlplane aws|vsphere|preprovisioned|azure --help` for more advance options.

Once all nodepools have been updated, your upgrade is complete. For the overall process for upgrading to the latest version of DKP, refer back to DKP Upgrade[653] for more details.

For the overall process for upgrading to the latest version of DKP, refer back to DKP Upgrade(see page 0).

10.4 Upgrade Custom Applications

10.4.1 Verify the compatibility of Custom Applications with the current Kubernetes version

We recommend upgrading your Custom Applications to the latest compatible version as soon as possible. Since Custom Applications are not created, maintained or supported by D2iQ, you must upgrade them manually.

653 https://d2iq.atlassian.net/wiki/spaces/DENT/pages/29923608/Upgrade+DKP#Understand-the-upgrade-process

⚠ Ensure you validate any Custom Applications you run for compatibility issues against the Kubernetes version(see page 811) in the new release. If the Custom Application's version is not compatible with the Kubernetes version, do not continue with the Konvoy upgrade. Otherwise, your custom Applications may stop running.

Once you have ensured your Custom Applications are compatible with the current Kubernetes version, return to the DKP Upgrade overview(see page 0) documentation, to review the next steps depending on your environment and license type.

11 DKP Tutorials

Learn how to put DKP in action:

11.1 Authorize a group across clusters

ENTERPRISE

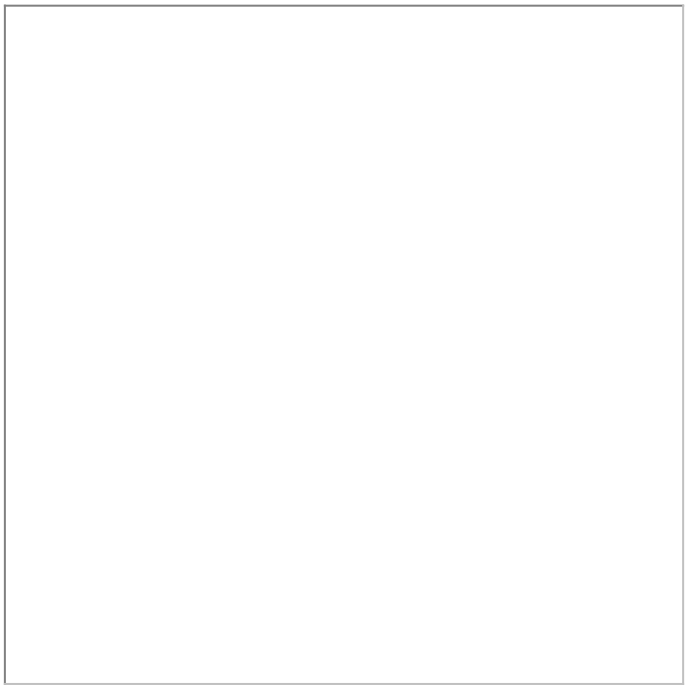
Install GitHub as an identity provider and grant access to all developers

11.1.1 Authorize all developers to have read access to your clusters

You want to ensure every developer in your GitHub organization has access to your Kubernetes clusters.

1. Set up GitHub as an identity provider. Start by creating a new OAuth Application in our GitHub Organization by filling out this form[654].
   Important: Use your cluster URL followed by `/dex/callback` as the Authorization callback URL.
2. After you create the application, you will be taken to a settings page. You will need the **Client ID** and **Client Secret** from this page for the DKP UI. Select the **Generate a new client secret** button if you do not already have a Client Secret for the application.
3. From the top menu bar in the DKP UI, select the **Global** workspace.
4. Select **Identity Providers** in the **Administration** section of the sidebar menu.
5. Select the **Identity Providers** tab, and then select the **+ Add Identity Provider** button.
6. Ensure GitHub is selected as the identity provider type, and copy the **Client ID** and **Client Secret** values into the form.
7. Select **Save** to create your Identity Provider.

654 https://github.com/settings/applications/new



D2iQ configured the identity provider to load all groups, so you need to map these groups to the Kubernetes groups.

11.1.1.1 Map the Identity Provider Groups to the Kubernetes Groups

Follow these steps:

1. Select the **Groups** tab, and then select the **Create Group** button.
2. Give your group a descriptive name and add the groups from your GitHub provider under **Identity Provider Groups**.
3. Click **Save** to create the group, which creates it on the management cluster and federated to all target clusters, and also describes the developers for your organization.

To enable this group, you need to first create a role which allows you to view every resource.

11.1.1.2 Create a “Read Everything” Role

Follow these steps:

1. Select **Access Control** in the **Administration** section of the sidebar menu.
2. Select the **Cluster Roles** tab, and then select the **+ Create Role** button.
3. Give the role a descriptive name, and ensure that **Cluster Role** is selected as the type.
4. For a read-only role, select **+ Add Rule**, then select **All Resource Types** in the **Resources** input, and select the **get**, **list**, and **watch** verbs.

Now you can assign the “Read Everything” role to the developers group.

11.1.1.3 Assign the Role to the Developers Group

Follow these steps:

1. Select the **Cluster Role Bindings** tab, and then select the **Add roles** button for your group.

2. Select “Read Everything” role from the **Roles** drop-down.

Lastly, follow the example in the Access Control documentation(see page 390) to grant users access to Kommander routes on your cluster.

When you check your attached clusters and login as a user from your matched groups, you can see every resource, but neither delete or edit them, as intended.

11.2 Continuous Delivery with GitOps

DKP enables software and applications to be continuously delivered (CD) using GitOps processes. GitOps enables you to deploy an application according to a manifest that is stored in a Git repository. This ensures that the application deployment can be automated, audited, and declaratively deployed to the infrastructure.

This section contains step-by-step tutorials for performing some common deployment-related tasks using DKP. All tutorials begin with a Prerequisites section that contains links to any steps that need to be taken first. This means you can visit any tutorial to get started.

11.2.1 Store secrets in GitOps repository using SealedSecrets

11.2.1.1 Securely managing secrets in a GitOps workflow using SealedSecrets

For security reasons, Kubernetes secrets are usually the only resource that cannot be managed with a GitOps workflow. Instead of managing secrets outside of GitOps and having to use a third-party tool like Vault, SealedSecrets provides a way to keep all the advantages of using a GitOps workflow while avoiding exposing secrets. SealedSecrets is composed of two main components:

- A CLI (Kubeseal) to encrypt secrets.
- A cluster-side controller to decrypt the sealed secrets into regular Kubernetes secrets. Only this controller can decrypt sealed secrets, not even the original author.

This tutorial describes how to install these two components, configure the controller, and add or remove sealed secrets.

11.2.1.2 Set up

These instructions are used as an example. For instructions on the latest release, see the release page[655]. For full documentation on SealedSecrets, see the GitHub repo[656].

11.2.1.3 Install Kubeseal CLI to encrypt your secrets

- On MacOS

```
brew install kubeseal
```

- On Linux

655 https://github.com/bitnami-labs/sealed-secrets/releases
656 https://github.com/bitnami-labs/sealed-secrets

```
wget https://github.com/bitnami-labs/sealed-secrets/releases/download/v0.18.1/
kubeseal-0.18.1-linux-amd64.tar.gz -O kubeseal
sudo install -m 755 kubeseal /usr/local/bin/kubeseal
```

11.2.1.4 Install the SealedSecrets controller on your cluster

This controller will be able to decrypt SealedSecrets and create Kubernetes secrets.

1. Create the controller:

   ```
   kubectl apply -f https://github.com/bitnami-labs/sealed-secrets/releases/
   download/v0.18.1/controller.yaml
   ```

2. Fetch the certificate that you will use to encrypt your secrets into sealed secrets:

   ```
   kubeseal --fetch-cert > mycert.pem
   ```

3. Commit `mycert.pem` to your git repo.

11.2.1.5 Add a secret

Secrets can be securely added to Git using sealed secrets:

1. Export the project namespace that you are using for your GitOps repository

   ```
   export PROJECT_NAMESPACE=<your-project-with-gitops>
   ```

2. Create a Kubernetes secret and pipe it into kubeseal[657] using the certificate `mycert.pem` that you fetched from the controller in the setup:

   ```
   echo '---' >> secrets.yaml
   kubectl create secret -n ${PROJECT_NAMESPACE} generic mysecret --dry-run=client
   -o yaml --from-literal=my-secret=value | \
       kubeseal --format yaml --cert mycert.pem >> secrets.yaml
   ```

3. Go to the end of `secrets.yaml` where you just added your new sealed secret. Remove any "creationTimestamp" fields from the YAML.
4. Apply the `secrets.yaml` file to your namespace. If you do not have permission, commit your changes to the repo and let FluxCD apply the changes for you.

   ```
   kubectl apply -f secrets.yaml
   ```

5. The sealed secret controller will then decrypt the sealed secret and generate a Kubernetes secret from it. Your secret got successfully created by running:

657 https://github.com/bitnami-labs/sealed-secrets#usage

```
kubectl get secret mysecret -n ${PROJECT_NAMESPACE} -o yaml
```

6. If your sealed secret got created successfully but did not generate the matching secret, look at the logs of the controller:

```
kubectl logs -l=name=sealed-secrets-controller -n kube-system
```

7. Commit `secrets.yaml` to your repo if you have not already done so in step 3.

11.2.1.6 Remove a secret

1. Following the same example from above in "Adding a secret", now remove the manifest for `mysecret` in `secrets.yaml` and commit those changes to the repo.
2. Delete the SealedSecret in the cluster:

```
kubectl delete SealedSecret -n ${PROJECT_NAMESPACE} mysecret
```

3. Delete the secret itself:

```
kubectl delete secret -n ${PROJECT_NAMESPACE} mysecret
```

11.2.1.7 Rotate the controller's sealing key

For added security, it is a good practice to rotate the key the controller uses to decrypt sealed secrets. By default, the controller generates a new key every 30 days. When this happens, you need to update the certificate you use to create sealed secrets by fetching the latest one:

```
kubeseal --fetch-cert > mycert.pem
```

Do not forget to commit it back to the repo!

In a disaster case, let's say your cluster gets destroyed, you would lose all your sealing keys, so you would not be able to recreate all the secrets from the sealed secrets in your GitOps repo. For this reason, you might want to back up the sealing keys. To do this every time a new sealing key is generated, run:

```
kubectl get secret -n kube-system -l sealedsecrets.bitnami.com/sealed-secrets-key -o
yaml > sealing-key
```

Then store `sealing-key` with the others in a safe location such as OneLogin Notes or Vault. To restore from a backup after a disaster, recreate all of the sealing keys with `kubectl apply -f sealing-key1 sealing-`

`key2 ...` before starting the controller. If the controller was already started, restart it: `kubectl delete pod -n kube-system -l name=sealed-secrets-controller`

To disable sealing key rotation For example, configure the controller's command in the pod template with `--key-renew-period=0`. See the following YAML file.

```
Pod Template:
  Labels:           name=sealed-secrets-controller
  Service Account:  sealed-secrets-controller
  Containers:
   sealed-secrets-controller:
    Image:      docker.io/bitnami/sealed-secrets-controller:v0.18.1
    Port:       8080/TCP
    Host Port:  0/TCP
    Command:    controller
      --key-renew-period=0
```

If required, edit the controller's manifest with:

```
kubectl edit deployment.apps/sealed-secrets-controller -n kube-system
```

11.2.2 Deploy a Sample App from DKP GitOps

Enterprise Gov Advanced

Use this procedure to deploy a sample `podinfo` application from DKP Enterprise GitOps.

11.2.2.1 Prerequisites

- Enterprise DKP installed(see page 97)
- Kommander installed(see page 366)
- Github account and personal access token[658]
- Add cluster to Kommander(see page 505)
- Setup Workspace and Projects(see page 432)

11.2.2.2 Deployment Steps

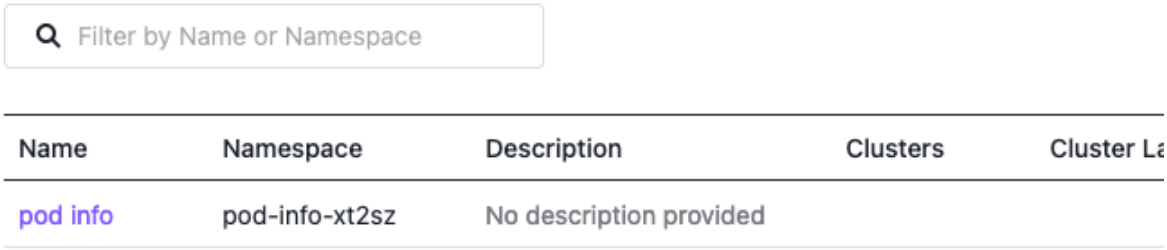
This procedure was run on an AWS cluster with DKP 2.2.2 installed.

Follow these steps:

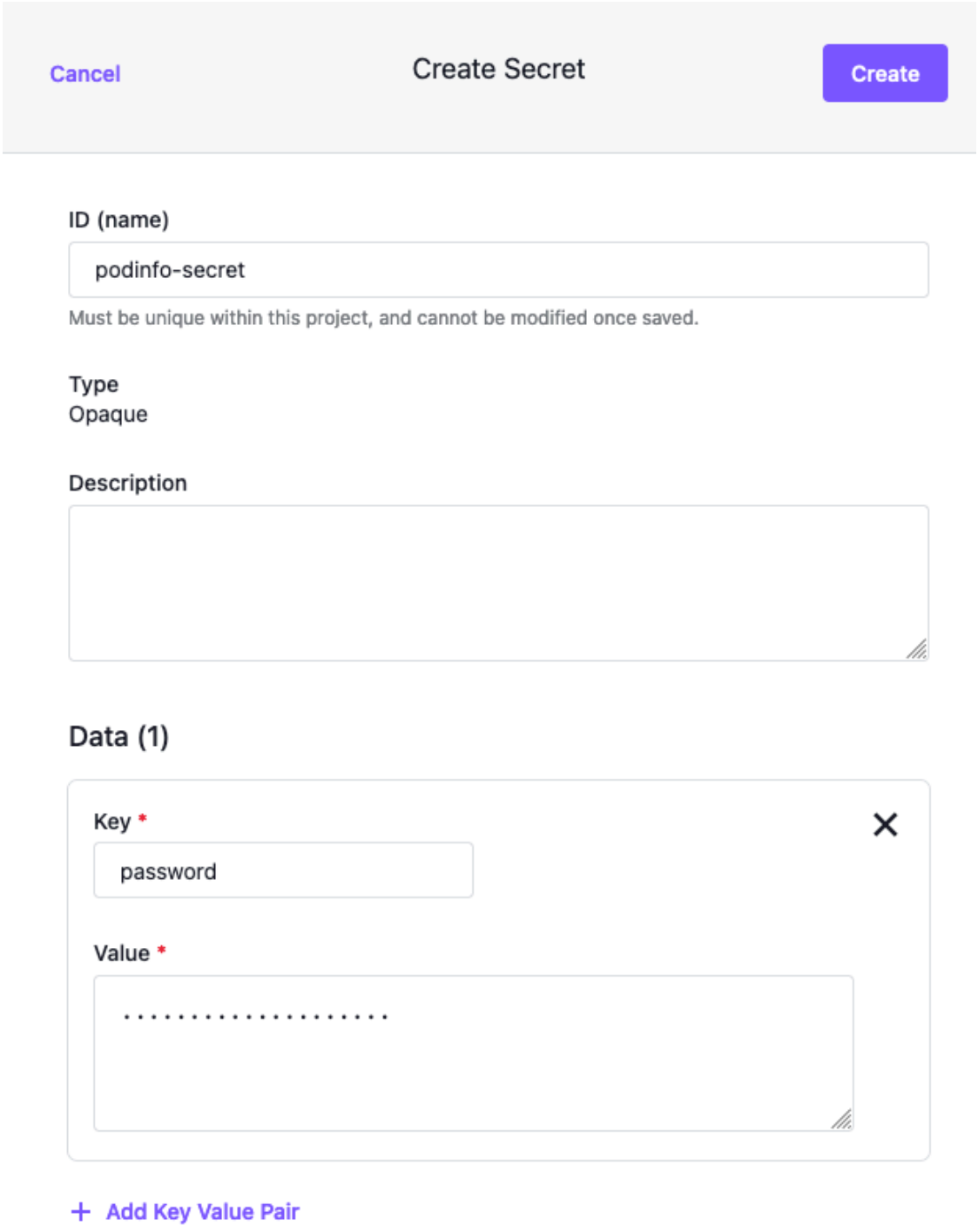
1. Ensure you are on the **Default Workspace** so that you can create a project.

658 https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token

2. Create a project(see page 459).
   In the working example we name the project **pod-info**. When you create a namespace, Kommander appends five alphanumeric characters. You can opt to select a target cluster for this project from one of the available attached clusters, and then this **(pod-info-xxxxx)** is the namespace used for deployments under the project, for example:



3. [Optional] Create a secret in order to pull from the repository, for private repositories.
   a. Select the Secrets tab and set up your secret according to the Continuous Deployment documentation(see page 462).
   b. Add a **key** and **value** pair for the GitHub personal access token and then select **Create**.

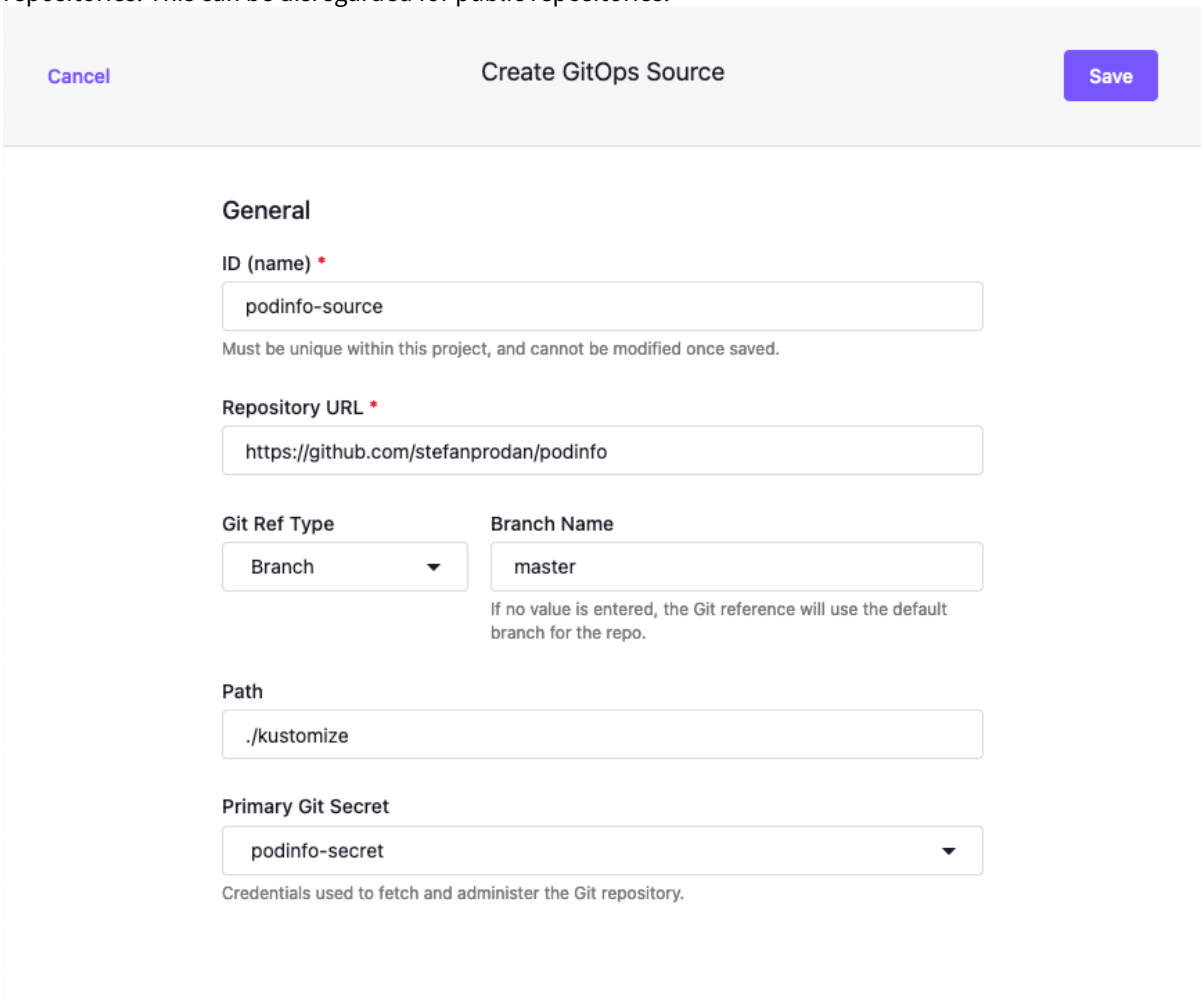


4. Verify that the secret `podinfo-secret` is created on the project namespace:

```
kubectl get secrets -n pod-info-xt2sz --kubeconfig=${CLUSTER_NAME}.conf
NAME                          TYPE                                  DATA   AGE
default-token-k685t           kubernetes.io/service-account-token   3      94m
```

```
pod-info-xt2sz-token-p9k5z   kubernetes.io/service-account-token   3      94m
podinfo-secret               Opaque                               1      1s
tls-root-ca                  Opaque                               1      93m
```

5. Select your project and then select the **CD** tab.
6. Create a GitOps Source, complete the required fields, and then **Save**.
   There are several configurable options such as selecting the **Git Ref Type** but in this example we use the master branch. **The Path** value should contain where the manifests are located. Additionally, the **Primary Git Secret** is the secret (**podinfo-secret**) that you created in the previous step, if you need to access private repositories. This can be disregarded for public repositories.



7. Verify the status of `gitrepository` creation with this command, (attached cluster) and if **READY** is marked as **True**:

```
kubectl get gitrepository -A --kubeconfig=${CLUSTER_NAME}.conf
NAMESPACE       NAME            URL
AGE READY   STATUS
kommander-flux   management     https://gitea-http.kommander.svc/kommander/
kommander.git   134m   True  stored artifact for revision 'main/
4fbee486076778c85e14f3196e49b8766e50e6ce'
```

```
pod-info-xt2sz   podinfo-source https://github.com/stefanprodan/podinfo
116m   True stored artifact for revision 'master/
b3b00fe35424a45d373bf4c7214178bc36fd7872'
```

8. Verify the `Kustomization` with this command below (attached cluster) and if **READY** is marked as **True**:

```
kubectl get kustomizations -n pod-info-xt2sz --kubeconfig=${CLUSTER_NAME}.conf
NAME                AGE READY   STATUS
originalpodinfo     10m True    Applied revision: master/
b3b00fe35424a45d373bf4c7214178bc36fd7872
podinfo-source      113m   True Applied revision: master/
b3b00fe35424a45d373bf4c7214178bc36fd7872
project             116m   True Applied revision: main/
4fbee486076778c85e14f3196e49b8766e50e6ce
project-tls-root-ca   117m   True   Applied revision: main/
4fbee486076778c85e14f3196e49b8766e50e6ce
```

Note the **port** so that you can use to verify if the app is deployed correctly:

```
kubectl get deployments,services -n pod-info-xt2sz --kubeconfig=$
{CLUSTER_NAME}.conf
NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/podinfo   2/2   2            2           118m

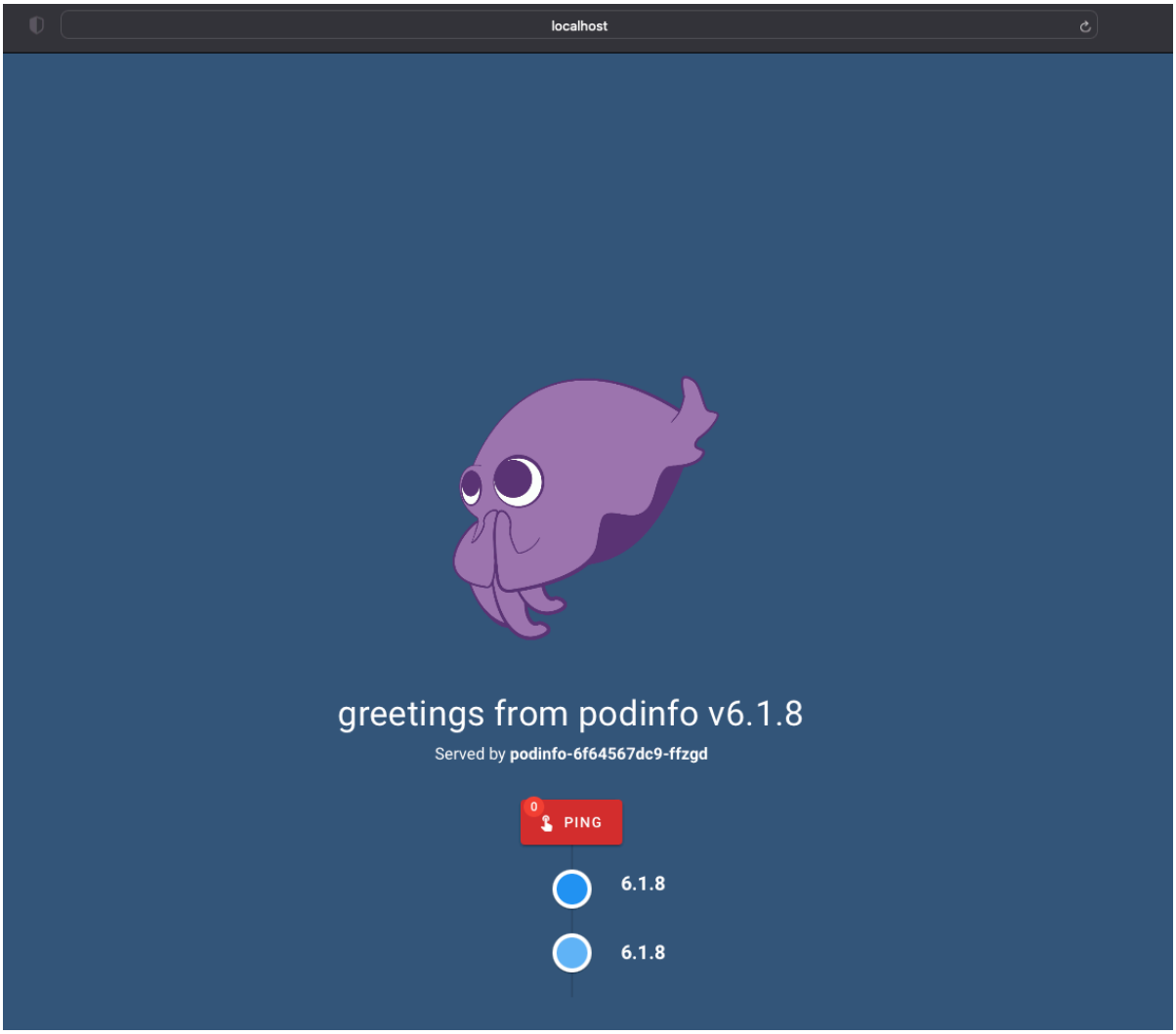
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)             AGE
service/podinfo   ClusterIP   10.99.239.120   <none>        9898/TCP,9999/TCP
118m
```

9. Port forward the **podinfo** service (port **9898**) to verify:

```
kubectl port-forward svc/podinfo -n pod-info-xt2sz 9898:9898 --kubeconfig=$
{CLUSTER_NAME}.conf

Forwarding from 127.0.0.1:9898 -> 9898
Forwarding from [::1]:9898 -> 9898
Handling connection for 9898
Handling connection for 9898
Handling connection for 9898
```

10. Open a browser and type in **localhost:9898**. A successful deployment of the podinfo app gives you this page:


localhost
greetings from podinfo v6.1.8
Served by podinfo-6f64567dc9-ffzgd
0
PING
6.1.8
6.1.8

12 Release Notes

View release-specific information for DKP

Download DKP

You must be a registered user and logged on to the support portal to download this product. New customers must contact their sales representative or sales@d2iq.com[659] before attempting to download or install DKP.

Release Notes for specific releases:

- Release Notes 2.3.0 (see page 811)
- Release Notes 2.3.1 (see page 824)
- Release Notes 2.3.2 (see page 836)
- Release Notes 2.3.3 (see page 852)

12.1 Release Notes 2.3.0

DKP® version 2.3 was released on August 16, 2022.

Download DKP

You must be a registered user and logged on to the support portal to download this product. New customers must contact their sales representative or sales@d2iq.com[660] before attempting to download or install DKP.

12.1.1 Release Summary

Welcome to D2iQ Kubernetes Platform (DKP) 2.3! This release provides fixes to reported issues, integrates changes from previous releases, and maintains compatibility and support for other packages used in DKP.

12.1.1.1 Supported Versions

Any DKP cluster you attach using DKP 2.3.0 must be running a Kubernetes version in the following ranges:

Kubernetes Support	Version
DKP Minimum	1.22.0
DKP Maximum	1.23.x
DKP Default	1.23.12

659 mailto:sales@d2iq.com
660 mailto:sales@d2iq.com

Kubernetes Support	Version
EKS Default	1.22.x
AKS Default	1.23.x
GKE Default	1.22.x-1.23.x

DKP 2.3 comes with support for Kubernetes 1.23, enabling you to benefit from the latest features and security fixes in upstream Kubernetes. This release comes with approximately 47 enhancements. To read more about major features in this release, visit https://kubernetes.io/blog/2021/12/07/kubernetes-1-23-release-announcement/.

12.1.2 Features and Enhancements

The following improvements are included in this release.

12.1.2.1 Support for Amazon EKS

ENTERPRISE

DKP 2.3 enables easier management of EKS clusters(see page 197). When utilizing AWS Elastic Kubernetes Service (EKS), you must add additional services for capabilities such as multi-cluster management, operational insights, Kubecost, Flux, Velero, Prometheus, Grafana, Cert-Manager, Calico, Gatekeeper, Dex, and artificial intelligence (AI) and machine learning (ML) based on Kubeflow.

DKP brings value to EKS customers by providing all components needed for a production-ready Kubernetes environment. DKP 2.3 provides the capability to provision EKS clusters using the DKP UI, In addition to above, DKP 2.3 also provides the ability to upgrade your EKS clusters using the DKP platform, making it possible to manage the comeplete lifecycle of EKS clusters from a centralized platform.

DKP adds value to Amazon EKS through features such as Time to Value, Cloud-Native Expertise, Military-Grade Security, and Lower TCO, among other features documented in the EKS insfrastructure section(see page 197).

Additionally, we now provide the ability to EKS Upgrade via CLI.

12.1.2.2 Support for GCP

Provision your Kubernetes clusters on the Google Cloud Platform(see page 300) with DKP 2.3. By making DKP as your choice of platform for multi-cluster management, you will now be able to centrally manage all your Kubernetes clusters in the top three public cloud providers (AWS, Azure, and GCP), making multi-cloud Kubernetes management easy with DKP.

12.1.2.3 Attach an existing GKE cluster to DKP

ENTERPRISE

You can attach an existing GKE(see page 537) (Google Kubernetes Engine) cluster to DKP. After attaching, you can use DKP to examine and manage(see page 502) the cluster.

12.1.2.4 Improved Documentation Site

DKP 2.3 comes with consolidated documentation, where we reorganized and updated the formerly separate Konvoy and Kommander documentation into the new D2iQ Help Center[661]. This provides you with improved search functionality and gives us the future ability to add multimedia content. Explore the new D2iQ Help Center at the same site as our previous documentation. You can still access all n-2 supported documentation at https://archive-docs.d2iq.com/.

The new organization highlights capabilities available to DKP Enterprise users only, such as EKS(see page 68) and AKS(see page 61), where you'll see the new Enterprise badge:

ENTERPRISE

Previous versions of the documentation remain available on our Archived Docs site[662].

12.1.2.5 Multiple Availability Zones

Availability zones (AZs) are isolated locations within data center regions where public cloud services originate and operate. DKP now supports multiple AZs. Because all the nodes in a node pool are deployed in a single AZ, you might want to create additional node pools, to ensure your cluster has nodes deployed in multiple AZs. By default, the control-plane Nodes are created in three different zones. However, the default worker Nodes reside in a single AZ. You can create additional node pools in other AZs with the `dkp create nodepool` command.

12.1.2.6 Custom Domains and Certificates for Workload (managed and attached) Clusters

ENTERPRISE

Configure a custom domain(see page 355) and certificate for your Managed or Attached cluster. DKP supports configuring a custom domain name per cluster, so you can access the DKP UI and other platform services via that domain. Additionally, you can provide a custom certificate for the domain, or one can be issued automatically by Let's Encrypt (or other certificate authorities supporting the ACME protocol).

12.1.2.7 Updated Image Bundle Names

We changed the Image Bundle extensions from `tar.gz` to `.tar`, as follows:

- The Kommander Image bundle(see page 0) is now `kommander-image-bundle-v2.3.0.tar`
- The DKP Catalog Image bundle(see page 0) is now `dkp-catalog-applications-image-bundle-v2.3.0.tar`
- The DKP Insights Catalog Application Image bundle is now `dkp-insights-image-bundle-v2.3.0.tar`

Since these files are no longer a compressed file format (.gz), they no longer require decompression.

661 http://docs.d2iq.com
662 https://archive-docs.d2iq.com/

12.1.2.8 Updated Custom Certificate Name

When you install or create a cluster with a custom domain, the Certificate Authority (CA) automatically creates a certificate. In DKP versions 2.2 and earlier, this certificate is called `kommander-traefik-acme`. In this version of DKP and later, the certificate is called `kommander-traefik-tls`.

If you have set up automation or customization around this certificate, ensure you update the certificate name in objects that reference it.

12.1.2.9 DKP Upgrades

From version 2.2 to the latest version 2.3, the following upgrades are available:

- Ability to upgrade all Platform apps in CLI (non air-gapped).
- Ability to upgrade all Platform apps in CLI (air-gapped).

For more information see https://d2iq.atlassian.net/wiki/spaces/DENT/pages/88411452/2.3%2BUpgrade%2BDKP#Supported-upgrade-paths(see page 0).

12.1.2.10 DKP Insights Alert Details

DKP Insights detects various kinds anomalies in the Kubernetes clusters and workloads and presents them as Insight Alerts in an Insights table. In this release, we enhance an insight alert with a details page. For more information, see DKP Insights Release Notes[663].

12.1.2.11 Support for Cluster-scoped Configuration and Deployments

ENTERPRISE

When you enable an application for a Workspace, you deploy that application to all clusters within the Workspace. You can also choose to enable or customize an application on certain clusters within a Workspace. This enhanced functionality allows you to use DKP in a multi-cluster scenario without restricting the management of your clusters from a single workspace.

The cluster-scoped enablement and customization of applications(see page 434) is an Enterprise only(see page 87) feature, which allows the configuration of all Workspace applications (Platform(see page 421), DKP Catalog(see page 441) and Custom applications(see page 451)) in your managed and attached clusters, regardless of your environment configuration (air-gapped or non-air-gapped).

12.1.2.12 Upgrade vSphere from the CLI

We provided the ability to upgrade Core Addons as well as Kubernetes version, via the CLI. Refer to these sections for more information:

- https://d2iq.atlassian.net/wiki/spaces/DENT/pages/89033677/2.3%2BDKP%2BEnterprise%2BUpgrade#Upgrade-the-core-addons(see page 0)
- https://d2iq.atlassian.net/wiki/spaces/DENT/pages/89033677/2.3%2BDKP%2BEnterprise%2BUpgrade#Upgrade-the-Kubernetes-version(see page 0)

663 https://d2iq.atlassian.net/wiki/spaces/DINS/pages/54984920/DKP+Insights+Release+Notes

12.1.2.13 Grafana Loki log retention policy

By default, Grafana Loki has a storage retention period of one week. If you want to keep log metadata and logs for a different period of time, override the ConfigMap to modify the storage retention period in Grafana Loki(see page 567).

12.1.3 2.3.0 components and applications

The following are component and application versions for DKP 2.3.0.

12.1.3.1 Components

Component Name	Version
Cluster API Core (CAPI)	1.1.3-d2iq.5
Cluster API AWS Infrastructure Provider (CAPA)	1.4.1
Cluster API Google Cloud Infrastructure Provider (CAPG)	1.1.0
Cluster API Pre-provisioned Infrastructure Provider (CAPPP)	0.9.2
Cluster API vSphere Infrastructure Provider (CAPV)	1.2.0
Cluster API Azure Infrastructure Provider (CAPZ)	1.3.2
Konvoy Image Builder	1.19.9[664]
containerd	1.4.13
etcd	3.4.13

12.1.3.2 Applications

Common Application Name	APP ID	Version	Component Versions
Centralized Grafana	centralized-grafana	34.9.3	• chart: 34.9.3 • prometheus-operator: 0.55.0

664 https://github.com/mesosphere/konvoy-image-builder/releases/tag/v1.19.8

Common Application Name	APP ID	Version	Component Versions
Centralized Kubecost	centralized-kubecost	0.26.0	• chart: 0.26.0 • kubecost: 1.95.0
Cert Manager	cert-manager	1.7.1	• chart: 1.7.1 • cert-manager: 1.7.1
Chartmuseum	chartmuseum	3.9.0	• chart: 3.9.0 • chartmuseum: 3.9.0
Dex	dex	2.9.18	• chart: 2.9.18 • dex: 2.31.0
Dex K8s Authenticator	dex-k8s-authenticator	1.2.13	• chart: 1.2.13 • dex-k8s-authenticator: 1.2.4
DKP Insights Management	dkp-insights-management	0.2.2	• chart: 0.2.2 • dkp-insights-management: 0.2.2
External DNS	external-dns	6.5.5	• chart: 6.5.5 • external-dns: 0.12.0
Fluent Bit	fluent-bit	0.19.21	• chart: 0.19.20 • fluent-bit: 1.9.3
Gatekeeper	gatekeeper	3.8.1	• chart: 3.8.1 • gatekeeper: 3.8.1
Gitea	gitea	5.0.9	• chart: 5.0.9 • gitea: 1.16.8
Grafana Logging	grafana-logging	6.28.0	• chart: 6.28.0 • grafana: 8.5.0
Grafana Loki	grafana-loki	0.48.4	• chart: 0.48.4 • loki: 2.5.0
Istio	istio	1.14.1	• chart: 1.14.1 • istio: 1.14.1
Jaeger	jaeger	2.32.2	• chart: 2.32.2 • jaeger: 1.34.1

Common Application Name	APP ID	Version	Component Versions
Karma	karma	2.0.1	• chart: 2.0.1 • karma: 0.70
Kiali	kiali	1.52.0	• chart: 1.52.0 • kiali: 1.52.0
Knative	knative	0.4.0	• chart: 0.4.0 • knative: 0.22.3
Kube OIDC Proxy	kube-oidc-proxy	0.3.1	• chart: 0.3.1 • kube-oidc-proxy: 0.3.0
Kube Prometheus Stack	kube-prometheus-stack	34.9.3	• chart: 34.9.3 • prometheus-operator: 0.55.0 • prometheus: 2.34.0 • prometheus-alertmanager: 0.24.0 • grafana: 8.5.0
Kubecost	kubecost	0.26.0	• chart: 0.26.0 • kubecost: 1.95.0
Kubefed	kubefed	0.9.2	• chart: 0.9.2 • kubefed: 0.9.2
Kubernetes Dashboard	kubernetes-dashboard	5.1.1	• chart: 5.1.1 • kubernetes-dashboard: 2.4.0
Kubetunnel	kubetunnel	0.0.13	• chart: 0.0.13 • kubetunnel: 0.0.13
Logging Operator	logging-operator	3.17.7	• chart: 3.17.7 • logging-operator: 3.17.7
MinIO Operator	minio-operator	4.4.25	• chart: 4.4.25 • minio-operator: 4.4.25
NFS Server Provisioner	nfs-server-provisioner	0.6.0	• chart: 0.6.0 • nfs-server-provisioner: 2.3.0
Nvidia	nvidia	0.4.4	• chart: 0.4.4 • nvidia-device-plugin: 0.1.4

Common Application Name	APP ID	Version	Component Versions
Grafana (project)	project-grafana-logging	6.28.0	• chart: 6.28.0 • grafana: 8.5.0
Grafana Loki (project)	project-grafana-loki	0.48.4	• chart: 0.48.4 • loki: 2.5.0
Prometheus Adapter	prometheus-adapter	2.17.1	• chart: 2.17.1 • prometheus-adapter: 0.9.1
Reloader	reloader	0.0.110	• chart: 0.0.110 • reloader: 0.0.110
Thanos	thanos	0.4.6	• chart: 0.4.6 • thanos: 0.17.1
Traefik	traefik	10.9.1	• chart: 10.9.1 • traefik: 2.5.6
Traefik ForwardAuth	traefik-forward-auth	0.3.8	• chart: 0.3.8 • traefik-forward-auth: 3.1.0
Velero	velero	3.2.3	• chart: 3.2.3 • velero: 1.5.2

12.1.4 Known issues and limitations

The following items are known issues with this release.

12.1.4.1 Use static credentials to provision an Azure cluster

Only static credentials can be used when provisioning an Azure cluster(see page 811).

12.1.4.2 When attaching GKE clusters, create a ResourceQuota to enable log collection

After you attach the GKE cluster, you can choose to deploy a stack of applications for workspace or project log collection. Once you have enabled this stack(see page 562), create a `ResourceQuota` which is required for the logging stack to function correctly. You will have to do this manually, because some DKP versions do not properly handle this by default.
Create the following resource to enable log collection:

1. Execute the following command to get the namespace of your workspace *on the management cluster*:

```
kubectl get workspaces
```

And copy the value under `WORKSPACE NAMESPACE` column for your workspace. This may NOT be identical to the Display Name of the `Workspace`.

2. Set the `WORKSPACE_NAMESPACE` environment variable to the name of the workspace's namespace:

```
export WORKSPACE_NAMESPACE=<gkeattached-cluster-namespace>
```

3. Run the following command *on your attached GKE cluster* to create the resource:

```
cat << EOF | kubectl apply -f -
apiVersion: v1
kind: ResourceQuota
metadata:
  name: fluent-bit-critical-pods
  namespace: ${WORKSPACE_NAMESPACE}
spec:
  hard:
    pods: "1G"
  scopeSelector:
    matchExpressions:
    - operator: In
      scopeName: PriorityClass
      values:
      - system-node-critical
EOF
```

After a few minutes, log collection is available in your GKE cluster.

This workflow only creates a `ResourceQuota` in the targeted workspace. Repeat these steps if you want to deploy the logging stack to additional workspaces with GKE clusters.

12.1.4.3 Resolve issues with failed HelmReleases

There is an existing issue with the Flux helm-controller[665] that can cause HelmReleases to get "stuck" with an error message such as *Helm upgrade failed: another operation (install/upgrade/rollback) is in progress*. This can happen when the helm-controller is restarted while a HelmRelease is upgrading, installing, and so on.

Workaround

To ensure the HelmRelease error was caused by the helm-controller restarting, first try to suspend/resume the HelmRelease:

665 https://github.com/fluxcd/helm-controller/issues/149

```
kubectl -n <namespace> patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[{"op":
"replace", "path": "/spec/suspend", "value": true}]'
kubectl -n <namespace> patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[{"op":
"replace", "path": "/spec/suspend", "value": false}]'
```

This might resolve the issue. If not, continue with the following steps:

You should see the HelmRelease attempting to reconcile, and then it either succeeds (with status: 'Release reconciliation succeeded') or it fails with the same error as before.

If the HelmRelease is still in the failed state, it is likely related to the helm-controller restarting. For example, if the 'reloader' HelmRelease is the one that is stuck.

To resolve the issue, follow these steps:

1. List secrets containing the affected HelmRelease name:

   ```
   kubectl get secrets -n ${NAMESPACE} | grep reloader
   ```

   ```
   kommander-reloader-reloader-token-9qd8b                          kubernetes.io/
   service-account-token   3      171m
   sh.helm.release.v1.kommander-reloader.v1                         helm.sh/
   release.v1                     1      171m
   sh.helm.release.v1.kommander-reloader.v2                         helm.sh/
   release.v1                     1      117m
   ```

   In this example, `sh.helm.release.v1.kommander-reloader.v2` is the most recent revision.

2. Find and delete the most recent revision secret. For example `sh.helm.release.v1.*.<revision>`

   ```
   kubectl delete secret -n <namespace> <most recent helm revision secret name>
   ```

3. Suspend and resume the HelmRelease to trigger a reconciliation:

   ```
   kubectl -n <namespace> patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[
   {"op": "replace", "path": "/spec/suspend", "value": true}]'
   kubectl -n <namespace> patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[
   {"op": "replace", "path": "/spec/suspend", "value": false}]'
   ```

You should see the HelmRelease is reconciled and eventually the upgrade and install succeeds.

12.1.4.4 Fluentbit disabled by default for DKP 2.3

Fluentbit is disabled by default in DKP 2.3 due to memory constraints. The amount of admin logs ingested to Loki requires additional disk space to be configured on the `grafana-loki-minio` Minio Tenant. Enabling admin

logs may use around 2GB/day per node. See Configuring-the-Grafana-Loki-Minio-Tenant[666] for more details on how to configure the Minio Tenant.

If Fluentbit is enabled on the management cluster and you would like it to continue to be deployed after the upgrade, you must pass in the `--disable-appdeployments {}` flag to the `dkp upgrade kommander` command. Otherwise, Fluentbit is automatically disabled upon upgrade.

12.1.4.5 Configure the Grafana Loki MinIO Tenant

Additional steps are required to change the default configuration of the MinIO Tenant that is deployed with Grafana Loki, `grafana-loki-minio`. Using config overrides is not supported.

By default, the `grafana-loki-minio` MinIO Tenant is configured with 2 pools with 4 servers each, 1 volume per server, for a total of 80GB.

The MinIO usable storage capacity is always less than the actual storage amount.

Use MinIO Erasure code calculator[667] to establish the appropriate configuration for your log storage requirement.

- You are only able to expand MinIO storage by adding more MinIO server pools with the correct configuration. Modifying existing server pools does not work as MinIO does not support reducing storage capacity. See this MinIO Operator documentation[668] for details.
- This impacts all your `AppDeployment` objects that reference the `grafana-loki` Kommander application definition.
- The changes introduced by the following procedure are wiped out upon Kommander install and upgrade.

In this example, we modify the `grafana-loki-minio` MinIO `Tenant` object in `kommander-workspace` (namespace: `kommander`)

1. Use this script to clone the management git repository from the Management cluster:

```
export KUBECONFIG=$KUBECONFIG

PASS=$(kubectl get secrets -nkommander admin-git-credentials -oyaml -o go-
template="{{.data.password | base64decode }}")
URL=https://gitea_admin:$PASS@$(kubectl -n kommander get ingress gitea -o
jsonpath='{.status.loadBalancer.ingress[0].hostname}'):443/dkp/kommander/git/
kommander/kommander

git clone -c http.sslVerify=false $URL repo
```

666 https://docs.d2iq.com/dkp/2.3/2-3-release-notes#Configuring-the-Grafana-Loki-Minio-Tenant
667 https://min.io/product/erasure-code-calculator
668 https://github.com/minio/operator/blob/7ae1610432ad3174150f4adaab1562c3ee522468/docs/expansion.md#adding-capacity-to-a-minio-tenant-

2. Modify `repo/services/grafana-loki/0.48.4/minio.yaml` by appending a new server pool to `.spec.pools` field, for example:

```
# the following will add a new server pool with 4 servers
# each server is attached with 1 PersistentVolume of 50G
- servers: 4
  volumesPerServer: 1
  volumeClaimTemplate:
    metadata:
      name: grafana-loki-minio
    spec:
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 50Gi
  resources:
    limits:
      cpu: 750m
      memory: 1Gi
    requests:
      cpu: 250m
      memory: 768Mi
  securityContext:
    runAsUser: 0
    runAsGroup: 0
    runAsNonRoot: false
    fsGroup: 0
```

3. Commit the changes to local clone of the git management repository when you are done editing:

```
git add services/grafana-loki/0.48.4/minio.yaml
git commit # finish the commit message editing in editor
```

4. Ensure that it is safe to apply the change, and then push the change to management git repository:

```
git push origin main
```

5. Set your `WORKSPACE_NAMESPACE` env variable:

```
# this is an example for kommander-workspace
export WORKSPACE_NAMESPACE=kommander
```

6. Verify that the `Tenant` is modified as expected, when the grafana-loki kustomizations reconcile:

```
# this prints the .status field of the tenant
```

```
kubectl get tenants -n kommander grafana-loki-minio -o jsonpath='{ .status }' |
jq
```

7. Verify that the new `StatefulSet` is `READY`:

```
kubectl get sts -n $WORKSPACE_NAMESPACE -l v1.min.io/tenant=grafana-loki-minio

NAME                       READY   AGE
grafana-loki-minio-ss-0    4/4     144m
grafana-loki-minio-ss-1    4/4     144m
grafana-loki-minio-ss-2    4/4     15m
```

8. Restart all the `StatefulSets` that back this `Tenant`:

```
kubectl -n $WORKSPACE_NAMESPACE rollout restart sts grafana-loki-minio-ss-0
statefulset.apps/grafana-loki-minio-ss-0 restarted
kubectl -n $WORKSPACE_NAMESPACE rollout restart sts grafana-loki-minio-ss-1
statefulset.apps/grafana-loki-minio-ss-1 restarted
kubectl -n $WORKSPACE_NAMESPACE rollout restart sts grafana-loki-minio-ss-2
statefulset.apps/grafana-loki-minio-ss-2 restarted
```

9. Verify that the MinIO Pods that back this `Tenant` are all online:

```
kubectl logs -n $WORKSPACE_NAMESPACE -l v1.min.io/tenant=grafana-loki-minio
...
Verifying if 1 bucket is consistent across drives...
Automatically configured API requests per node based on available memory on the
system: 424
All MinIO sub-systems initialized successfully
Waiting for all MinIO IAM sub-system to be initialized.. lock acquired
Status:         12 Online, 0 Offline.
API: http://minio.kommander.svc.cluster.local

Console: http://192.168.202.223:9090 http://127.0.0.1:9090

Documentation: https://docs.min.io
...
```

12.1.4.6 FIPS upgrade from 2.2.x to 2.3.0

If upgrading a FIPS cluster, there is a bug in the upgrade of `kube-proxy` `DaemonSet` in that it doesn't get automatically upgraded. After completing the cluster upgrade, run the following command to finish upgrading the kube-proxy `DaemonSet`:

```
kubectl set image -n kube-system daemonset.v1.apps/kube-proxy kube-proxy=docker.io/
mesosphere/kube-proxy:v1.23.7_fips.0
```

12.1.4.7 `Kube-oidc-proxy` not ready after upgrade

If you installed or attached a cluster in 2.1, `kube-oidc-proxy` is not available after upgrading to 2.3. This application is required to access the Kubernetes API (with `kubectl`) using SSO. For affected customers, there are issues with the authentication via `kubectl`.

To make the application available, run the following command on each cluster that was installed, created or attached in 2.1, and is now on DKP version 2.3.0. Replace `<namespace>` with each cluster's workspace namespace:

```
kubectl -n <namespace> patch appdeployment kube-oidc-proxy --type=json -p
'[{"op":"remove","path":"/spec/configOverrides"}]'
```

12.1.5 Additional resources

- For more information about working with native Kubernetes, see the Kubernetes documentation[669].
- For a full list of attributed 3rd party software, see http://d2iq.com/legal/3rd .

12.2 Release Notes 2.3.1

DKP® version 2.3.1 was released on October 13, 2022.

Download DKP

You must be a registered user and logged on to the support portal to download this product. New customers must contact their sales representative or sales@d2iq.com[670] before attempting to download or install DKP.

12.2.1 Release Summary

Welcome to D2iQ Kubernetes Platform (DKP) 2.3.1! This release provides fixes to reported issues, integrates changes from previous releases, and maintains compatibility and support for other packages used in DKP.

12.2.2 Fixes and Updates

The following updates and fixes are included in this release.

669 https://kubernetes.io/docs/home/
670 mailto:sales@d2iq.com

12.2.2.1 Kommander Install fails on Gatekeeper

Incident D2IQ-92981

When attempting to install Kommander on a cluster in FIPS mode, the install fails because the `gatekeeper-update-namespace-label` pod continuously crash loops. The following error message returns from the pod logs:

```
Error from server (InternalError): Internal error occurred: failed calling webhook
"check-ignore-label.gatekeeper.sh": failed to call webhook: Post "https://gatekeeper-
webhook-service.kommander.svc:443/v1/admitlabel?timeout=3s": remote error: tls:
protocol version not supported
```

This error occurred due to gatekeeper attempting to use a TLS 1.3 connection, however TLS 1.3 is not supported in FIPS 140-2 mode. The gatekeeper deployment was changed to always use TLS 1.2, correcting the issue.

12.2.2.2 KIB 1.19.9 AWS + Airgapped + FIPS fails to Install rpm Packages

Incident D2IQ-92900

Attempting to create a FIPS compliant machine image for RHEL 8.x fails, producing the following error:

```
rhel-8.4: FAILED - RETRYING: install kubectl rpm package (1 retries left).
    rhel-8.4: fatal: [default]:
FAILED! => {"attempts": 3, "changed": false, "failures": [], "msg": "Unknown Error
occured: Transaction test error:\n  package kubectl-1.23.12-0.x86_64 does not verify:
no digest\n", "rc": 1, "results": []}
rhel-8.4:rhel-8.4: PLAY RECAP
*********************************************************************
rhel-8.4: default: ok=39   changed=22   unreachable=0    failed=1    skipped=41
rescued=0  ignored=1 rhel-8.4:
```

This error occurred due to air-gapped FIPS RPM package bundles not being signed with FIPS compatible package signatures. This issue is now resolved.

12.2.2.3 Download Signature Files

You need to download an appropriate, signed signature file before you run FIPS validation. Verify which version of DKP you are running to ensure you are downloading the manifest that is compliant with the DKP release number on your system. You can use the FIPS validation tool to verify that specific components and services are FIPS-compliant by checking the signatures of the files against a signed signature file, and by checking that services are using the certified algorithms. Select the links in the **Manifest URL** column of the following table to obtain a valid file:

DKP version 2.3.1

Operating System version	Kubernetes version	containerd version	Manifest URL
CentOS 7.9	v1.23.12	1.14.13	v1.23.12 CentOS 7.9 Manifest[671]
Oracle 7.9	v1.23.12	1.14.13	v1.23.12 OL 7.9 Manifest[672]
RHEL 7.9	v1.23.12	1.14.13	v1.23.12 RHEL 7.9 Manifest[673]
RHEL 8.2	v1.23.12	1.14.13	v1.23.12 RHEL 8.2 Manifest[674]
RHEL 8.4	v1.23.12	1.14.13	v1.23.12 RHEL 8.4 Manifest[675]

12.2.2.4 Supported Versions

Any DKP cluster you attach using DKP 2.3.1 must be running a Kubernetes version in the following ranges:

Kubernetes Support	Version
DKP Minimum	1.22.0
DKP Maximum	1.23.x
DKP Default	1.23.12
EKS Default	1.22.x
AKS Default	1.23.x
GKE Default	1.22.x-1.23.x

DKP 2.3 comes with support for Kubernetes 1.23, enabling you to benefit from the latest features and security fixes in upstream Kubernetes. This release comes with approximately 47 enhancements. To read more about major features in this release, visit https://kubernetes.io/blog/2021/12/07/kubernetes-1-23-release-announcement/.

671 https://downloads.d2iq.com/dkp/fips/v2.3.1/manifest-centos-79.json.asc
672 https://downloads.d2iq.com/dkp/fips/v2.3.1/manifest-oracle-79.json
673 https://downloads.d2iq.com/dkp/fips/v2.3.1/manifest-rhel-79.json.asc
674 https://downloads.d2iq.com/dkp/fips/v2.3.1/manifest-rhel-82.json.asc
675 https://downloads.d2iq.com/dkp/fips/v2.3.1/manifest-rhel-84.json.asc

12.2.3 2.3.1 components and applications

The following are component and application versions for DKP 2.3.1.

12.2.3.1 Components

Component Name	Version
Cluster API Core (CAPI)	1.1.3-d2iq.5
Cluster API AWS Infrastructure Provider (CAPA)	1.4.1
Cluster API Google Cloud Infrastructure Provider (CAPG)	1.1.0
Cluster API Pre-provisioned Infrastructure Provider (CAPPP)	0.9.4
Cluster API vSphere Infrastructure Provider (CAPV)	1.2.0
Cluster API Azure Infrastructure Provider (CAPZ)	1.3.2
Konvoy Image Builder	1.19.11[676]
containerd	1.4.13
etcd	3.4.13

12.2.3.2 Applications

Common Application Name	APP ID	Version	Component Versions
Centralized Grafana	centralized-grafana	34.9.3	• chart: 34.9.3 • prometheus-operator: 0.55.0
Centralized Kubecost	centralized-kubecost	0.26.0	• chart: 0.26.0 • kubecost: 1.95.0
Cert Manager	cert-manager	1.7.1	• chart: 1.7.1 • cert-manager: 1.7.1

676 https://github.com/mesosphere/konvoy-image-builder/releases/tag/v1.19.11

Common Application Name	APP ID	Version	Component Versions
Chartmuseum	chartmuseum	3.9.0	• chart: 3.9.0 • chartmuseum: 3.9.0
Dex	dex	2.9.18	• chart: 2.9.18 • dex: 2.31.0
Dex K8s Authenticator	dex-k8s-authenticator	1.2.13	• chart: 1.2.13 • dex-k8s-authenticator: 1.2.4
DKP Insights Management	dkp-insights-management	0.2.2	• chart: 0.2.2 • dkp-insights-management: 0.2.2
External DNS	external-dns	6.5.5	• chart: 6.5.5 • external-dns: 0.12.0
Fluent Bit	fluent-bit	0.19.21	• chart: 0.19.20 • fluent-bit: 1.9.3
Gatekeeper	gatekeeper	3.8.2	• chart: 3.8.1 • gatekeeper: 3.8.1
Gitea	gitea	5.0.9	• chart: 5.0.9 • gitea: 1.16.8
Grafana Logging	grafana-logging	6.28.0	• chart: 6.28.0 • grafana: 8.5.0
Grafana Loki	grafana-loki	0.48.4	• chart: 0.48.4 • loki: 2.5.0
Istio	istio	1.14.1	• chart: 1.14.1 • istio: 1.14.1
Jaeger	jaeger	2.32.2	• chart: 2.32.2 • jaeger: 1.34.1
Karma	karma	2.0.1	• chart: 2.0.1 • karma: 0.70
Kiali	kiali	1.52.0	• chart: 1.52.0 • kiali: 1.52.0

Common Application Name	APP ID	Version	Component Versions
Knative	knative	0.4.0	• chart: 0.4.0 • knative: 0.22.3
Kube OIDC Proxy	kube-oidc-proxy	0.3.1	• chart: 0.3.1 • kube-oidc-proxy: 0.3.0
Kube Prometheus Stack	kube-prometheus-stack	34.9.3	• chart: 34.9.3 • prometheus-operator: 0.55.0 • prometheus: 2.34.0 • prometheus-alertmanager: 0.24.0 • grafana: 8.5.0
Kubecost	kubecost	0.26.0	• chart: 0.26.0 • kubecost: 1.95.0
Kubefed	kubefed	0.9.2	• chart: 0.9.2 • kubefed: 0.9.2
Kubernetes Dashboard	kubernetes-dashboard	5.1.1	• chart: 5.1.1 • kubernetes-dashboard: 2.4.0
Kubetunnel	kubetunnel	0.0.13	• chart: 0.0.13 • kubetunnel: 0.0.13
Logging Operator	logging-operator	3.17.7	• chart: 3.17.7 • logging-operator: 3.17.7
MinIO Operator	minio-operator	4.4.25	• chart: 4.4.25 • minio-operator: 4.4.25
NFS Server Provisioner	nfs-server-provisioner	0.6.0	• chart: 0.6.0 • nfs-server-provisioner: 2.3.0
Nvidia	nvidia	0.4.4	• chart: 0.4.4 • nvidia-device-plugin: 0.1.4
Grafana (project)	project-grafana-logging	6.28.0	• chart: 6.28.0 • grafana: 8.5.0
Grafana Loki (project)	project-grafana-loki	0.48.4	• chart: 0.48.4 • loki: 2.5.0

Common Application Name	APP ID	Version	Component Versions
Prometheus Adapter	prometheus-adapter	2.17.1	• chart: 2.17.1 • prometheus-adapter: 0.9.1
Reloader	reloader	0.0.110	• chart: 0.0.110 • reloader: 0.0.110
Thanos	thanos	0.4.6	• chart: 0.4.6 • thanos: 0.17.1
Traefik	traefik	10.9.1	• chart: 10.9.1 • traefik: 2.5.6
Traefik ForwardAuth	traefik-forward-auth	0.3.8	• chart: 0.3.8 • traefik-forward-auth: 3.1.0
Velero	velero	3.2.3	• chart: 3.2.3 • velero: 1.5.2

12.2.4 Known issues and limitations

The following items are known issues with this release.

12.2.4.1 Use static credentials to provision an Azure cluster

Only static credentials can be used when provisioning an Azure cluster(see page 824).

12.2.4.2 When attaching GKE clusters, create a ResourceQuota to enable log collection

After you attach the GKE cluster, you can choose to deploy a stack of applications for workspace or project log collection. Once you have enabled this stack(see page 824), create a `ResourceQuota` which is required for the logging stack to function correctly. You will have to do this manually, because some DKP versions do not properly handle this by default.
Create the following resource to enable log collection:

1. Execute the following command to get the namespace of your workspace *on the management cluster*:

   ```
   kubectl get workspaces
   ```

   And copy the value under `WORKSPACE NAMESPACE` column for your workspace. This may NOT be identical to the Display Name of the `Workspace`.
2. Set the `WORKSPACE_NAMESPACE` environment variable to the name of the workspace's namespace:

```
export WORKSPACE_NAMESPACE=<gkeattached-cluster-namespace>
```

3. Run the following command *on your attached GKE cluster* to create the resource:

```
cat << EOF | kubectl apply -f -
apiVersion: v1
kind: ResourceQuota
metadata:
  name: fluent-bit-critical-pods
  namespace: ${WORKSPACE_NAMESPACE}
spec:
  hard:
    pods: "1G"
  scopeSelector:
    matchExpressions:
    - operator: In
      scopeName: PriorityClass
      values:
      - system-node-critical
EOF
```

After a few minutes, log collection is available in your GKE cluster.

This workflow only creates a `ResourceQuota` in the targeted workspace. Repeat these steps if you want to deploy the logging stack to additional workspaces with GKE clusters.

12.2.4.3 Resolve issues with failed HelmReleases

There is an existing issue with the Flux helm-controller[677] that can cause HelmReleases to get "stuck" with an error message such as *Helm upgrade failed: another operation (install/upgrade/rollback) is in progress*. This can happen when the helm-controller is restarted while a HelmRelease is upgrading, installing, and so on.

Workaround

To ensure the HelmRelease error was caused by the helm-controller restarting, first try to suspend/resume the HelmRelease:

```
kubectl -n <namespace> patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[{"op":
"replace", "path": "/spec/suspend", "value": true}]'
kubectl -n <namespace> patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[{"op":
"replace", "path": "/spec/suspend", "value": false}]'
```

This might resolve the issue. If not, continue with the following steps:

You should see the HelmRelease attempting to reconcile, and then it either succeeds (with status: 'Release reconciliation succeeded') or it fails with the same error as before.

677 https://github.com/fluxcd/helm-controller/issues/149

If the HelmRelease is still in the failed state, it is likely related to the helm-controller restarting. For example, if the 'reloader' HelmRelease is the one that is stuck.

To resolve the issue, follow these steps:

1. List secrets containing the affected HelmRelease name:

```
kubectl get secrets -n ${NAMESPACE} | grep reloader
```

```
kommander-reloader-reloader-token-9qd8b                          kubernetes.io/
service-account-token   3      171m
sh.helm.release.v1.kommander-reloader.v1                         helm.sh/
release.v1                     1      171m
sh.helm.release.v1.kommander-reloader.v2                         helm.sh/
release.v1                     1      117m
```

   In this example, `sh.helm.release.v1.kommander-reloader.v2` is the most recent revision.

2. Find and delete the most recent revision secret. For example `sh.helm.release.v1.*.<revision>`

```
kubectl delete secret -n <namespace> <most recent helm revision secret name>
```

3. Suspend and resume the HelmRelease to trigger a reconciliation:

```
kubectl -n <namespace> patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[
{"op": "replace", "path": "/spec/suspend", "value": true}]'
kubectl -n <namespace> patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[
{"op": "replace", "path": "/spec/suspend", "value": false}]'
```

You should see the HelmRelease is reconciled and eventually the upgrade and install succeeds.

12.2.4.4 Fluentbit disabled by default for DKP 2.3

Fluentbit is disabled by default in DKP 2.3 due to memory constraints. The amount of admin logs ingested to Loki requires additional disk space to be configured on the `grafana-loki-minio` Minio Tenant. Enabling admin logs may use around 2GB/day per node. See Configuring-the-Grafana-Loki-Minio-Tenant[678] for more details on how to configure the Minio Tenant.

If Fluentbit is enabled on the management cluster and you would like it to continue to be deployed after the upgrade, you must pass in the `--disable-appdeployments {}` flag to the `dkp upgrade kommander` command. Otherwise, Fluentbit is automatically disabled upon upgrade.

678 https://docs.d2iq.com/dkp/2.3/2-3-release-notes#Configuring-the-Grafana-Loki-Minio-Tenant

12.2.4.5 Configure the Grafana Loki MinIO Tenant

Additional steps are required to change the default configuration of the MinIO Tenant that is deployed with Grafana Loki, `grafana-loki-minio`. Using config overrides is not supported.

By default, the `grafana-loki-minio` MinIO Tenant is configured with 2 pools with 4 servers each, 1 volume per server, for a total of 80GB.

The MinIO usable storage capacity is always less than the actual storage amount.

Use MinIO Erasure code calculator[679] to establish the appropriate configuration for your log storage requirement.

- You are only able to expand MinIO storage by adding more MinIO server pools with the correct configuration. Modifying existing server pools does not work as MinIO does not support reducing storage capacity. See this MinIO Operator documentation[680] for details.
- This impacts all your `AppDeployment` objects that reference the `grafana-loki` Kommander application definition.
- The changes introduced by the following procedure are wiped out upon Kommander install and upgrade.

In this example, we modify the `grafana-loki-minio` MinIO `Tenant` object in `kommander-workspace` (namespace: `kommander`)

1. Use this script to clone the management git repository from the Management cluster:

```
export KUBECONFIG=$KUBECONFIG

PASS=$(kubectl get secrets -nkommander admin-git-credentials -oyaml -o go-
template="{{.data.password | base64decode }}")
URL=https://gitea_admin:$PASS@$(kubectl -n kommander get ingress gitea -o
jsonpath='{.status.loadBalancer.ingress[0].hostname}'):443/dkp/kommander/git/
kommander/kommander

git clone -c http.sslVerify=false $URL repo
```

2. Modify `repo/services/grafana-loki/0.48.4/minio.yaml` by appending a new server pool to `.spec.pools` field, for example:

```
# the following will add a new server pool with 4 servers
# each server is attached with 1 PersistentVolume of 50G
- servers: 4
```

679 https://min.io/product/erasure-code-calculator
680 https://github.com/minio/operator/blob/7ae1610432ad3174150f4adaab1562c3ee522468/docs/expansion.md#adding-capacity-to-a-minio-tenant-

```
  volumesPerServer: 1
  volumeClaimTemplate:
    metadata:
      name: grafana-loki-minio
    spec:
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 50Gi
  resources:
    limits:
      cpu: 750m
      memory: 1Gi
    requests:
      cpu: 250m
      memory: 768Mi
  securityContext:
    runAsUser: 0
    runAsGroup: 0
    runAsNonRoot: false
    fsGroup: 0
```

3. Commit the changes to local clone of the git management repository when you are done editing:

```
git add services/grafana-loki/0.48.4/minio.yaml
git commit # finish the commit message editing in editor
```

4. Ensure that it is safe to apply the change, and then push the change to management git repository:

```
git push origin main
```

5. Set your `WORKSPACE_NAMESPACE` env variable:

```
# this is an example for kommander-workspace
export WORKSPACE_NAMESPACE=kommander
```

6. Verify that the `Tenant` is modified as expected, when the grafana-loki kustomizations reconcile:

```
# this prints the .status field of the tenant
kubectl get tenants -n kommander grafana-loki-minio -o jsonpath='{ .status }' |
jq
```

7. Verify that the new `StatefulSet` is `READY`:

```
kubectl get sts -n $WORKSPACE_NAMESPACE -l v1.min.io/tenant=grafana-loki-minio
```

```
NAME                          READY   AGE
grafana-loki-minio-ss-0   4/4     144m
grafana-loki-minio-ss-1   4/4     144m
grafana-loki-minio-ss-2   4/4     15m
```

8. Restart all the `StatefulSets` that back this `Tenant`:

```
kubectl -n $WORKSPACE_NAMESPACE rollout restart sts grafana-loki-minio-ss-0
statefulset.apps/grafana-loki-minio-ss-0 restarted
kubectl -n $WORKSPACE_NAMESPACE rollout restart sts grafana-loki-minio-ss-1
statefulset.apps/grafana-loki-minio-ss-1 restarted
kubectl -n $WORKSPACE_NAMESPACE rollout restart sts grafana-loki-minio-ss-2
statefulset.apps/grafana-loki-minio-ss-2 restarted
```

9. Verify that the MinIO Pods that back this `Tenant` are all online:

```
kubectl logs -n $WORKSPACE_NAMESPACE -l v1.min.io/tenant=grafana-loki-minio
...
Verifying if 1 bucket is consistent across drives...
Automatically configured API requests per node based on available memory on the
system: 424
All MinIO sub-systems initialized successfully
Waiting for all MinIO IAM sub-system to be initialized.. lock acquired
Status:         12 Online, 0 Offline.
API: http://minio.kommander.svc.cluster.local

Console: http://192.168.202.223:9090 http://127.0.0.1:9090

Documentation: https://docs.min.io
...
```

12.2.4.6 FIPS upgrade from 2.2.x to 2.3.0

If upgrading a FIPS cluster, there is a bug in the upgrade of `kube-proxy` `DaemonSet` in that it doesn't get automatically upgraded. After completing the cluster upgrade, run the following command to finish upgrading the kube-proxy `DaemonSet`:

```
kubectl set image -n kube-system daemonset.v1.apps/kube-proxy kube-proxy=docker.io/
mesosphere/kube-proxy:v1.23.12_fips.0
```

12.2.4.7 `Kube-oidc-proxy` not ready after upgrade

If you installed or attached a cluster in 2.1, `kube-oidc-proxy` is not available after upgrading to 2.3. This application is required to access the Kubernetes API (with `kubectl`) using SSO. For affected customers, there are issues with the authentication via `kubectl`.

To make the application available, run the following command on each cluster that was installed, created or attached in 2.1, and is now on DKP version 2.3.1. Replace `<namespace>` with each cluster's workspace namespace:

```
kubectl -n <namespace> patch appdeployment kube-oidc-proxy --type=json -p
'[{"op":"remove","path":"/spec/configOverrides"}]'
```

12.2.5 Additional resources

- For more information about working with native Kubernetes, see the Kubernetes documentation[681].
- For a full list of attributed 3rd party software, see http://d2iq.com/legal/3rd .

12.3 Release Notes 2.3.2

DKP® version 2.3.2 was released on February 14, 2023

Download DKP

You must be a registered user and logged on to the support portal to download this product. New customers must contact their sales representative or sales@d2iq.com[682] before attempting to download or install DKP.

12.3.1 Release Summary

Welcome to D2iQ Kubernetes Platform (DKP) 2.3.2! This release provides fixes to reported issues, integrates changes from previous releases, and maintains compatibility and support for other packages used in DKP.

12.3.2 DKP Fixes and Updates

The following issues are corrected or resolved in this release.

12.3.2.1 `Kube-oidc-proxy` is not Available After Upgrade

D2IQ-94629

681 https://kubernetes.io/docs/home/
682 mailto:sales@d2iq.com

If you installed or attached a cluster in 2.1, `kube-oidc-proxy` was not available after upgrading to 2.3.x. This prevented authentication via `kubectl` using SSO.

12.3.2.2 Failure to Upgrade Azure Clusters

D2IQ-95191

A bug in the Azure CSI driver caused problems during an upgrade by preventing volumes attached to cluster nodes from being detached, in some circumstances, so they were not available for the new nodes created by the upgrade process. This problem would prevent pods on the upgraded cluster from starting.

12.3.2.3 CAPA Provider for EKS not Working with -worker-iam-instance-profile

D2IQ-95493

When creating an EKS cluster and specifying a specific IAM instance profile to use on worker nodes, the nodes are created with the specified IAM instance profile. However, the CAPA controller currently does not honor this role, resulting in deployment failures. To workaround this issue, follow the instructions on the Grant Cluster Access(see page 214) page to adjust the roles the CAPA controller uses. In some instances, the IAM authenticator was missed as a prerequisite. For more information see the Amazon EKS documentation: Installing aws-iam-authenticator[683]

12.3.2.4 AKS Cluster Deployment hangs in a Pending State

D2IQ-94072

When deploying an AKS cluster using the DKP UI on an Azure hosted Management cluster, the AKS cluster deployed correctly, but was staying in the `Pending` state when viewed from the UI.

12.3.2.5 Improved Documentation for Changing Calico Encapsulation Type

D2IQ-94582

While deploying DKP 2.3 to a pre-provisioned Azure cluster, the documentation for changing the encapsulation type was not correct, which prevented correct configuration of the Calico Overlay network.

12.3.2.6 Kommander Installations Fail in Certain Scenarios

D2IQ-92981

When deploying DKP Applications to a centos79 pre-provisioned, air-gapped, and FIPS environment, deployment was failing due to the `gatekeeper-update-namespace-label` pod crashing and looping.

12.3.2.7 Corrected Upgrade Documentation to Specify Correct Kubernetes version

D2IQ-93793

This upgrade path to use when upgrading to DKP 2.3 failed due to an incorrectly configured flag.

683 https://docs.aws.amazon.com/eks/latest/userguide/install-aws-iam-authenticator.html

12.3.2.8 KIB 1.24.x Fails and Cannot Create Ubuntu Images

D2IQ-95429

Creating an Ubuntu 1804 or 2004 image for GCP with KIB 1.24.2 or 1.24.3 failed with an error.

12.3.3 DKP Insights Fixes and Updates

12.3.3.1 Incorrect CSI-related Polaris Messages for GCP and Azure

D2IQ-92665

DKP Insights was displaying some incorrect CSI-related critical Polaris messages for GCP and Azure clusters, but not AWS.

12.3.4 Download Signature Files

You need to download an appropriate, signed signature file before you run FIPS validation. Verify which version of DKP you are running to ensure you are downloading the manifest that is compliant with the DKP release number on your system. You can use the FIPS validation tool to verify that specific components and services are FIPS-compliant by checking the signatures of the files against a signed signature file, and by checking that services are using the certified algorithms. Select the links in the **Manifest URL** column of the following table to obtain a valid file:

12.3.4.1 DKP version 2.3.2

Operating System version	Kubernetes version	containerd version	Manifest URL
CentOS 7.9	v1.23.12	1.14.13	v1.23.12 CentOS 7.9 Manifest[684]
Oracle 7.9	v1.23.12	1.14.13	v1.23.12 OL 7.9 Manifest[685]
RHEL 7.9	v1.23.12	1.14.13	v1.23.12 RHEL 7.9 Manifest[686]
RHEL 8.2	v1.23.12	1.14.13	v1.23.12 RHEL 8.2 Manifest[687]
RHEL 8.4	v1.23.12	1.14.13	v1.23.12 RHEL 8.4 Manifest[688]

684 https://downloads.d2iq.com/dkp/fips/v2.3.2/manifest-centos-79.json.asc
685 https://downloads.d2iq.com/dkp/fips/v2.3.2/manifest-oracle-79.json
686 https://downloads.d2iq.com/dkp/fips/v2.3.2/manifest-rhel-79.json.asc
687 https://downloads.d2iq.com/dkp/fips/v2.3.2/manifest-rhel-82.json.asc
688 https://downloads.d2iq.com/dkp/fips/v2.3.2/manifest-rhel-84.json.asc

12.3.5 Supported Versions

Any DKP cluster you attach using DKP 2.3.2 must be running a Kubernetes version in the following ranges:

Kubernetes Support	Version
DKP Minimum	1.22.0
DKP Maximum	1.23.x
DKP Default	1.23.12
EKS Default	1.22.x
AKS Default	1.23.x
GKE Default	1.22.x-1.23.x

DKP 2.3 comes with support for Kubernetes 1.23, enabling you to benefit from the latest features and security fixes in upstream Kubernetes. This release comes with approximately 47 enhancements. To read more about major features in this release, visit https://kubernetes.io/blog/2021/12/07/kubernetes-1-23-release-announcement/.

12.3.6 2.3.2 Components and Applications

The following are component and application versions for DKP 2.3.2:

12.3.6.1 Components

Component Name	Version
Cluster API Core (CAPI)	1.1.3-d2iq.5
Cluster API AWS Infrastructure Provider (CAPA)	1.4.1
Cluster API Google Cloud Infrastructure Provider (CAPG)	1.1.0
Cluster API Pre-provisioned Infrastructure Provider (CAPPP)	0.9.5
Cluster API vSphere Infrastructure Provider (CAPV)	1.2.0
Cluster API Azure Infrastructure Provider (CAPZ)	1.3.2

Component Name	Version
Konvoy Image Builder	1.19.14[689]
containerd	1.4.13
etcd	3.4.13

12.3.6.2 Applications

Common Application Name	APP ID	Version	Component Versions	Helm Values	DKP Values
Centralized Grafana	centralized-grafana	34.9.3	• chart: 34.9.3[690] • prometheus-operator: 0.55.0 • grafana: 8.5.0	Link[691]	Link[692]
Centralized Kubecost	centralized-kubecost	0.27.0	• chart: 0.27.0[693] • kubecost: 1.96.0	Link[694]	Link[695]
Cert Manager	cert-manager	1.7.1	• chart: 1.7.1[696] • cert-manager: 1.7.1	Link[697]	Link[698]
Chartmuseum	chartmuseum	3.9.0	• chart: 3.9.0[699] • chartmuseum: 3.9.0	Link[700]	Link[701]

689 https://github.com/mesosphere/konvoy-image-builder/releases/tag/v1.19.14
690 https://artifacthub.io/packages/helm/mesosphere/kube-prometheus-stack/34.9.3
691 https://artifacthub.io/packages/helm/mesosphere/kube-prometheus-stack/34.9.3?modal=values
692 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/centralized-grafana/34.9.3/defaults/cm.yaml
693 https://artifacthub.io/packages/helm/mesosphere-stable/kubecost/0.27.0
694 https://artifacthub.io/packages/helm/mesosphere-stable/kubecost/0.27.0?modal=values
695 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/centralized-kubecost/0.27.0/defaults/cm.yaml
696 https://artifacthub.io/packages/helm/cert-manager/cert-manager/1.7.1
697 https://artifacthub.io/packages/helm/cert-manager/cert-manager/1.7.1?modal=values
698 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/cert-manager/1.7.1/defaults/cm.yaml
699 https://artifacthub.io/packages/helm/chartmuseum/chartmuseum/3.9.0
700 https://artifacthub.io/packages/helm/chartmuseum/chartmuseum/3.9.0?modal=values
701 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/chartmuseum/3.9.0/defaults/cm.yaml

Common Application Name	APP ID	Version	Component Versions	Helm Values	DKP Values
Dex	dex	2.9.19	• chart: 2.9.19[702] • dex: 2.31.0	Link[703]	Link[704]
Dex K8s Authenticator	dex-k8s-authenticator	1.2.14	• chart: 1.2.14[705] • dex-k8s-authenticator: 1.2.4	Link[706]	Link[707]
DKP Insights Management	dkp-insights-management	0.2.3	• chart: 0.2.3 • dkp-insights-management: 0.2.3	N/A	Link[708]
External DNS	external-dns	6.5.5	• chart: 6.5.5[709] • external-dns: 0.12.0	Link[710]	Link[711]
Fluent Bit	fluent-bit	0.19.24	• chart: 0.19.24[712] • fluent-bit: 1.8.15	Link[713]	Link[714]
Gatekeeper	gatekeeper	3.8.2	• chart: 3.8.1[715] • gatekeeper: 3.8.1	Link[716]	Link[717]

702 https://artifacthub.io/packages/helm/mesosphere-stable/dex/2.9.19
703 https://artifacthub.io/packages/helm/mesosphere-stable/dex/2.9.19?modal=values
704 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/dex/2.9.19/defaults/cm.yaml
705 https://artifacthub.io/packages/helm/mesosphere/dex-k8s-authenticator/1.2.14
706 https://artifacthub.io/packages/helm/mesosphere/dex-k8s-authenticator/1.2.14?modal=values
707 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/dex-k8s-authenticator/1.2.14/defaults/cm.yaml
708 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/dkp-insights-management/0.2.3/defaults/cm.yaml
709 https://artifacthub.io/packages/helm/bitnami/external-dns/6.5.5
710 https://artifacthub.io/packages/helm/bitnami/external-dns/6.5.5?modal=values
711 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/external-dns/6.5.5/defaults/cm.yaml
712 https://artifacthub.io/packages/helm/fluent/fluent-bit/0.19.24
713 https://artifacthub.io/packages/helm/fluent/fluent-bit/0.19.24?modal=values
714 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/fluent-bit/0.19.24/defaults/cm.yaml
715 https://artifacthub.io/packages/helm/gatekeeper/gatekeeper/3.8.1
716 https://artifacthub.io/packages/helm/gatekeeper/gatekeeper/3.8.1?modal=values
717 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/gatekeeper/3.8.2/defaults/cm.yaml

Common Application Name	APP ID	Version	Component Versions	Helm Values	DKP Values
Gitea	gitea	5.0.9	• chart: 5.0.9[718] • gitea: 1.16.8	Link[719]	Link[720]
Grafana Logging	grafana-logging	6.28.0	• chart: 6.28.0[721] • grafana: 8.5.0	Link[722]	Link[723]
Grafana Loki	grafana-loki	0.48.5	• chart: 0.48.4[724] • loki: 2.5.0	Link[725]	Link[726]
Istio	istio	1.14.1	• chart: 1.14.1[727] • istio: 1.14.1	Link[728]	Link[729]
Jaeger	jaeger	2.32.2	• chart: 2.32.2[730] • jaeger: 1.34.1	Link[731]	Link[732]
Karma	karma	2.0.1	• chart: 2.0.1[733] • karma: 0.70	Link[734]	Link[735]

718 https://artifacthub.io/packages/helm/gitea/gitea/5.0.9
719 https://artifacthub.io/packages/helm/gitea/gitea/5.0.9?modal=values
720 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/gitea/5.0.9/defaults/cm.yaml
721 https://artifacthub.io/packages/helm/grafana/grafana/6.28.0
722 https://artifacthub.io/packages/helm/grafana/grafana/6.28.0?modal=values
723 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/grafana-logging/6.28.0/defaults/cm.yaml
724 https://artifacthub.io/packages/helm/grafana/loki-distributed/0.48.4
725 https://artifacthub.io/packages/helm/grafana/loki-distributed/0.48.4?modal=values
726 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/grafana-loki/0.48.5/defaults/cm.yaml
727 https://artifacthub.io/packages/helm/mesosphere/istio/1.14.1
728 https://artifacthub.io/packages/helm/mesosphere/istio/1.14.1?modal=values
729 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/istio/1.14.1/defaults/cm.yaml
730 https://artifacthub.io/packages/helm/jaegertracing/jaeger-operator/2.32.2
731 https://artifacthub.io/packages/helm/jaegertracing/jaeger-operator/2.32.2?modal=values
732 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/jaeger/2.32.2/defaults/cm.yaml
733 https://artifacthub.io/packages/helm/mesosphere-stable/karma/2.0.1
734 https://artifacthub.io/packages/helm/mesosphere-stable/karma/2.0.1?modal=values
735 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/karma/2.0.1/defaults/cm.yaml

Common Application Name	APP ID	Version	Component Versions	Helm Values	DKP Values
Kiali	kiali	1.52.0	• chart: 1.52.0[736] • kiali: 1.52.0	Link[737]	Link[738]
Knative	knative	0.4.0	• chart: 0.4.0[739] • knative: 0.22.3	Link[740]	Link[741]
Flux	kommander-flux	0.31.4	• chart: N/A • flux: 0.31.4	N/A	N/A
Kube OIDC Proxy	kube-oidc-proxy	0.3.2	• chart: 0.3.1[742] • kube-oidc-proxy: 0.3.0	Link[743]	Link[744]
Kube Prometheus Stack	kube-prometheus-stack	34.9.3	• chart: 34.9.3[745] • prometheus-operator: 0.55.0 • grafana: 8.5.0 • prometheus: 2.34.0 • prometheus-alertmanager: 0.24.0	Link[746]	Link[747]
Kubecost	kubecost	0.27.0	• chart: 0.27.0[748] • kubecost: 1.96.0	Link[749]	Link[750]

736 https://artifacthub.io/packages/helm/kiali/kiali-operator/1.52.0
737 https://artifacthub.io/packages/helm/kiali/kiali-operator/1.52.0?modal=values
738 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/kiali/1.52.0/defaults/cm.yaml
739 https://artifacthub.io/packages/helm/mesosphere/knative/0.4.0
740 https://artifacthub.io/packages/helm/mesosphere/knative/0.4.0?modal=values
741 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/knative/0.4.0/defaults/cm.yaml
742 https://artifacthub.io/packages/helm/mesosphere/kube-oidc-proxy/0.3.1
743 https://artifacthub.io/packages/helm/mesosphere/kube-oidc-proxy/0.3.1?modal=values
744 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/kube-oidc-proxy/0.3.2/defaults/cm.yaml
745 https://artifacthub.io/packages/helm/mesosphere/kube-prometheus-stack/34.9.3
746 https://artifacthub.io/packages/helm/mesosphere/kube-prometheus-stack/34.9.3?modal=values
747 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/kube-prometheus-stack/34.9.3/defaults/cm.yaml
748 https://artifacthub.io/packages/helm/mesosphere-stable/kubecost/0.27.0
749 https://artifacthub.io/packages/helm/mesosphere-stable/kubecost/0.27.0?modal=values
750 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/kubecost/0.27.0/defaults/cm.yaml

Common Application Name	APP ID	Version	Component Versions	Helm Values	DKP Values
Kubefed	kubefed	0.9.2	• chart: 0.9.2[751] • kubefed: 0.9.2	Link[752]	Link[753]
Kubernetes Dashboard	kubernetes-dashboard	5.1.1	• chart: 5.1.1[754] • kubernetes-dashboard: 2.4.0	Link[755]	Link[756]
Kubetunnel	kubetunnel	0.0.13	• chart: 0.0.13 • kubetunnel: 0.0.13	N/A	Link[757]
Logging Operator	logging-operator	3.17.8	• chart: 3.17.7[758] • logging-operator: 3.17.7 • logging-operator-logging: 3.17.7	Link[759]	Link[760]
Metallb	metallb	0.12.3	• chart: 0.12.3[761] • metallb: 0.8.1	Link[762]	Link[763]
MinIO Operator	minio-operator	4.4.25	• chart: 4.4.25 • minio-operator: 4.4.25	Link(see page 836)	Link[764]

751 https://artifacthub.io/packages/helm/kubefed/kubefed/0.9.2
752 https://artifacthub.io/packages/helm/kubefed/kubefed/0.9.2?modal=values
753 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/kubefed/0.9.2/defaults/cm.yaml
754 https://artifacthub.io/packages/helm/k8s-dashboard/kubernetes-dashboard/5.1.1
755 https://artifacthub.io/packages/helm/k8s-dashboard/kubernetes-dashboard/5.1.1?modal=values
756 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/kubernetes-dashboard/5.1.1/defaults/cm.yaml
757 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/kubetunnel/0.0.13/defaults/cm.yaml
758 https://artifacthub.io/packages/helm/banzaicloud-stable/logging-operator/3.17.7
759 https://artifacthub.io/packages/helm/banzaicloud-stable/logging-operator/3.17.7?modal=values
760 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/logging-operator/3.17.8/defaults/cm.yaml
761 https://artifacthub.io/packages/helm/mesosphere-stable/metallb/0.12.3
762 https://artifacthub.io/packages/helm/mesosphere-stable/metallb/0.12.3?modal=values
763 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/metallb/0.12.3/defaults/cm.yaml
764 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/minio-operator/4.4.25/defaults/cm.yaml

Common Application Name	APP ID	Version	Component Versions	Helm Values	DKP Values
NFS Server Provisioner	nfs-server-provisioner	0.6.0	• chart: 0.6.0[765] • nfs-server-provisioner: 2.3.0	Link[766]	Link[767]
Nvidia	nvidia	0.4.4	• chart: 0.4.4[768] • nvidia-device-plugin: 0.2.0	Link[769]	Link[770]
Grafana (project)	project-grafana-logging	6.28.0	• chart: 6.28.0[771] • grafana: 8.5.0	Link[772]	Link[773]
Grafana Loki (project)	project-grafana-loki	0.48.5	• chart: 0.48.4[774] • loki: 2.5.0	Link[775]	Link[776]
Prometheus Adapter	prometheus-adapter	2.17.1	• chart: 2.17.1[777] • prometheus-adapter: 0.9.1	Link[778]	Link[779]

765 https://artifacthub.io/packages/helm/mesosphere/nfs-server-provisioner/0.6.0
766 https://artifacthub.io/packages/helm/mesosphere/nfs-server-provisioner/0.6.0?modal=values
767 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/nfs-server-provisioner/0.6.0/defaults/cm.yaml
768 https://artifacthub.io/packages/helm/mesosphere/nvidia/0.4.4
769 https://artifacthub.io/packages/helm/mesosphere/nvidia/0.4.4?modal=values
770 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/nvidia/0.4.4/defaults/cm.yaml
771 https://artifacthub.io/packages/helm/grafana/grafana/6.28.0
772 https://artifacthub.io/packages/helm/grafana/grafana/6.28.0?modal=values
773 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/project-grafana-logging/6.28.0/defaults/cm.yaml
774 https://artifacthub.io/packages/helm/grafana/loki-distributed/0.48.4
775 https://artifacthub.io/packages/helm/grafana/loki-distributed/0.48.4?modal=values
776 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/project-grafana-loki/0.48.5/defaults/cm.yaml
777 https://artifacthub.io/packages/helm/prometheus-community/prometheus-adapter/2.17.1
778 https://artifacthub.io/packages/helm/prometheus-community/prometheus-adapter/2.17.1?modal=values
779 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/prometheus-adapter/2.17.1/defaults/cm.yaml

Common Application Name	APP ID	Version	Component Versions	Helm Values	DKP Values
Reloader	reloader	0.0.110	• chart: 0.0.110[780] • reloader: 0.0.110	Link[781]	Link[782]
Thanos	thanos	0.4.7	• chart: 0.4.6[783] • thanos: 0.17.1	Link[784]	Link[785]
Traefik	traefik	10.9.3	• chart: 10.9.1[786] • traefik: 2.5.6	Link[787]	Link[788]
Traefik ForwardAuth	traefik-forward-auth	0.3.8	• chart: 0.3.8[789] • traefik-forward-auth: 3.1.0	Link[790]	Link[791]
Velero	velero	3.2.3	• chart: 3.2.3[792] • velero: 1.5.2	Link[793]	Link[794]

12.3.7 Known Issues and Limitations

The following items are known issues with this release.

12.3.7.1 Nvidia Feature Discovery Error

D2IQ-93676

780 https://github.com/stakater/Reloader/tree/v0.0.110/README.md#helm-charts
781 https://artifacthub.io/packages/helm/stakater/reloader/0.0.110?modal=values
782 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/reloader/0.0.110/defaults/cm.yaml
783 https://artifacthub.io/packages/helm/banzaicloud-stable/thanos/0.4.6
784 https://artifacthub.io/packages/helm/banzaicloud-stable/thanos/0.4.6?modal=values
785 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/thanos/0.4.7/defaults/cm.yaml
786 https://artifacthub.io/packages/helm/traefik/traefik/10.9.1
787 https://artifacthub.io/packages/helm/traefik/traefik/10.9.1?modal=values
788 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/traefik/10.9.3/defaults/cm.yaml
789 https://artifacthub.io/packages/helm/mesosphere/traefik-forward-auth/0.3.8
790 https://artifacthub.io/packages/helm/mesosphere/traefik-forward-auth/0.3.8?modal=values
791 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/traefik-forward-auth/0.3.8/defaults/cm.yaml
792 https://artifacthub.io/packages/helm/vmware-tanzu/velero/3.2.3
793 https://artifacthub.io/packages/helm/vmware-tanzu/velero/3.2.3?modal=values
794 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/velero/3.2.3/defaults/cm.yaml

When creating a new cluster to migrate Kaptain to version 2.1, after creating the cluster, the `nvidia-feature-discovery-gpu-feature-discovery` is in a `CrashLoopBackOff` state, with error.

Workaround

Follow these steps

1. Place the registry details in the override file, together with Nvidia.
2. Delete the current override and replaced it with the new override you just created in step 1.
3. Delete the machine to force preprovisioning.
4. Rename or delete the *.toml files from the import path directory set in `config.toml`
5. Restart containerd and GPU/Nvidia feature discovery.
6. Verify the Node now shows GPU resources.
7. Repeat these steps for all affected nodes.

12.3.7.2 Use static credentials to provision an Azure cluster

Only static credentials can be used when provisioning an Azure cluster(see page 177).

12.3.7.3 When attaching GKE clusters, create a ResourceQuota to enable log collection

After you attach the GKE cluster, you can choose to deploy a stack of applications for workspace or project log collection. Once you have enabled this stack(see page 836), create a `ResourceQuota` which is required for the logging stack to function correctly. You will have to do this manually, because some DKP versions do not properly handle this by default.
Create the following resource to enable log collection:

1. Execute the following command to get the namespace of your workspace *on the management cluster*:

   ```
   kubectl get workspaces
   ```

   And copy the value under `WORKSPACE NAMESPACE` column for your workspace. This may NOT be identical to the Display Name of the `Workspace`.
2. Set the `WORKSPACE_NAMESPACE` environment variable to the name of the workspace's namespace:

   ```
   export WORKSPACE_NAMESPACE=<gkeattached-cluster-namespace>
   ```
3. Run the following command *on your attached GKE cluster* to create the resource:

   ```
   cat << EOF | kubectl apply -f -
   apiVersion: v1
   kind: ResourceQuota
   metadata:
     name: fluent-bit-critical-pods
     namespace: ${WORKSPACE_NAMESPACE}
   spec:
     hard:
   ```

```
    pods: "1G"
  scopeSelector:
    matchExpressions:
    - operator: In
      scopeName: PriorityClass
      values:
      - system-node-critical
EOF
```

After a few minutes, log collection is available in your GKE cluster.

This workflow only creates a `ResourceQuota` in the targeted workspace. Repeat these steps if you want to deploy the logging stack to additional workspaces with GKE clusters.

12.3.7.4 Resolve issues with failed HelmReleases

There is an existing issue with the Flux helm-controller[795] that can cause HelmReleases to get "stuck" with an error message such as *Helm upgrade failed: another operation (install/upgrade/rollback) is in progress*. This can happen when the helm-controller is restarted while a HelmRelease is upgrading, installing, and so on.

Workaround

To ensure the HelmRelease error was caused by the helm-controller restarting, first try to suspend/resume the HelmRelease:

```
kubectl -n <namespace> patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[{"op":
"replace", "path": "/spec/suspend", "value": true}]'
kubectl -n <namespace> patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[{"op":
"replace", "path": "/spec/suspend", "value": false}]'
```

This might resolve the issue. If not, continue with the following steps:

You should see the HelmRelease attempting to reconcile, and then it either succeeds (with status: 'Release reconciliation succeeded') or it fails with the same error as before.

If the HelmRelease is still in the failed state, it is likely related to the helm-controller restarting. For example, if the 'reloader' HelmRelease is the one that is stuck.

To resolve the issue, follow these steps:

1. List secrets containing the affected HelmRelease name:

```
kubectl get secrets -n ${NAMESPACE} | grep reloader
```

795 https://github.com/fluxcd/helm-controller/issues/149

```
kommander-reloader-reloader-token-9qd8b                          kubernetes.io/
service-account-token   3      171m
sh.helm.release.v1.kommander-reloader.v1                         helm.sh/
release.v1                    1      171m
sh.helm.release.v1.kommander-reloader.v2                         helm.sh/
release.v1                    1      117m
```

In this example, `sh.helm.release.v1.kommander-reloader.v2` is the most recent revision.

2. Find and delete the most recent revision secret. For example `sh.helm.release.v1.*.<revision>`

```
kubectl delete secret -n <namespace> <most recent helm revision secret name>
```

3. Suspend and resume the HelmRelease to trigger a reconciliation:

```
kubectl -n <namespace> patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[
{"op": "replace", "path": "/spec/suspend", "value": true}]'
kubectl -n <namespace> patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[
{"op": "replace", "path": "/spec/suspend", "value": false}]'
```

You should see the HelmRelease is reconciled and eventually the upgrade and install succeeds.

12.3.7.5 Fluentbit disabled by default for DKP 2.3

Fluentbit is disabled by default in DKP 2.3 due to memory constraints. The amount of admin logs ingested to Loki requires additional disk space to be configured on the `grafana-loki-minio` Minio Tenant. Enabling admin logs may use around 2GB/day per node. See Configuring-the-Grafana-Loki-Minio-Tenant[796] for more details on how to configure the Minio Tenant.

If Fluentbit is enabled on the management cluster and you would like it to continue to be deployed after the upgrade, you must pass in the `--disable-appdeployments {}` flag to the `dkp upgrade kommander` command. Otherwise, Fluentbit is automatically disabled upon upgrade.

12.3.7.6 Configure the Grafana Loki MinIO Tenant

Additional steps are required to change the default configuration of the MinIO Tenant that is deployed with Grafana Loki, `grafana-loki-minio`. Using config overrides is not supported.

By default, the `grafana-loki-minio` MinIO Tenant is configured with 2 pools with 4 servers each, 1 volume per server, for a total of 80GB.

The MinIO usable storage capacity is always less than the actual storage amount.

Use MinIO Erasure code calculator[797] to establish the appropriate configuration for your log storage requirement.

796 https://docs.d2iq.com/dkp/2.3/2-3-release-notes#Configuring-the-Grafana-Loki-Minio-Tenant
797 https://min.io/product/erasure-code-calculator

⚠

- You are only able to expand MinIO storage by adding more MinIO server pools with the correct configuration. Modifying existing server pools does not work as MinIO does not support reducing storage capacity. See this MinIO Operator documentation[798] for details.
- This impacts all your `AppDeployment` objects that reference the `grafana-loki` Kommander application definition.
- The changes introduced by the following procedure are wiped out upon Kommander install and upgrade.

In this example, we modify the `grafana-loki-minio` MinIO `Tenant` object in `kommander-workspace` (namespace: `kommander`)

1. Use this script to clone the management git repository from the Management cluster:

```
export KUBECONFIG=$KUBECONFIG

PASS=$(kubectl get secrets -nkommander admin-git-credentials -oyaml -o go-
template="{{.data.password | base64decode }}")
URL=https://gitea_admin:$PASS@$(kubectl -n kommander get ingress gitea -o
jsonpath='{.status.loadBalancer.ingress[0].hostname}'):443/dkp/kommander/git/
kommander/kommander

git clone -c http.sslVerify=false $URL repo
```

2. Modify `repo/services/grafana-loki/0.48.4/minio.yaml` by appending a new server pool to `.spec.pools` field, for example:

```
# the following will add a new server pool with 4 servers
# each server is attached with 1 PersistentVolume of 50G
- servers: 4
  volumesPerServer: 1
  volumeClaimTemplate:
    metadata:
      name: grafana-loki-minio
    spec:
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 50Gi
  resources:
    limits:
      cpu: 750m
      memory: 1Gi
    requests:
```

798 https://github.com/minio/operator/blob/7ae1610432ad3174150f4adaab1562c3ee522468/docs/expansion.md#adding-capacity-to-a-minio-tenant-

```
      cpu: 250m
      memory: 768Mi
  securityContext:
    runAsUser: 0
    runAsGroup: 0
    runAsNonRoot: false
    fsGroup: 0
```

3. Commit the changes to local clone of the git management repository when you are done editing:

```
git add services/grafana-loki/0.48.4/minio.yaml
git commit # finish the commit message editing in editor
```

4. Ensure that it is safe to apply the change, and then push the change to management git repository:

```
git push origin main
```

5. Set your `WORKSPACE_NAMESPACE` env variable:

```
# this is an example for kommander-workspace
export WORKSPACE_NAMESPACE=kommander
```

6. Verify that the `Tenant` is modified as expected, when the grafana-loki kustomizations reconcile:

```
# this prints the .status field of the tenant
kubectl get tenants -n kommander grafana-loki-minio -o jsonpath='{ .status }' |
jq
```

7. Verify that the new `StatefulSet` is `READY`:

```
kubectl get sts -n $WORKSPACE_NAMESPACE -l v1.min.io/tenant=grafana-loki-minio

NAME                       READY   AGE
grafana-loki-minio-ss-0    4/4     144m
grafana-loki-minio-ss-1    4/4     144m
grafana-loki-minio-ss-2    4/4     15m
```

8. Restart all the `StatefulSets` that back this `Tenant`:

```
kubectl -n $WORKSPACE_NAMESPACE rollout restart sts grafana-loki-minio-ss-0
statefulset.apps/grafana-loki-minio-ss-0 restarted
kubectl -n $WORKSPACE_NAMESPACE rollout restart sts grafana-loki-minio-ss-1
statefulset.apps/grafana-loki-minio-ss-1 restarted
kubectl -n $WORKSPACE_NAMESPACE rollout restart sts grafana-loki-minio-ss-2
statefulset.apps/grafana-loki-minio-ss-2 restarted
```

9. Verify that the MinIO Pods that back this `Tenant` are all online:

```
kubectl logs -n $WORKSPACE_NAMESPACE -l v1.min.io/tenant=grafana-loki-minio
...
Verifying if 1 bucket is consistent across drives...
Automatically configured API requests per node based on available memory on the
system: 424
All MinIO sub-systems initialized successfully
Waiting for all MinIO IAM sub-system to be initialized.. lock acquired
Status:         12 Online, 0 Offline.
API: http://minio.kommander.svc.cluster.local

Console: http://192.168.202.223:9090 http://127.0.0.1:9090

Documentation: https://docs.min.io
...
```

12.3.7.7 FIPS upgrade from 2.2.x to 2.3.0

If upgrading a FIPS cluster, there is a bug in the upgrade of `kube-proxy` `DaemonSet` in that it does not get automatically upgraded. After completing the cluster upgrade, run the following command to finish upgrading the kube-proxy `DaemonSet`:

```
kubectl set image -n kube-system daemonset.v1.apps/kube-proxy kube-proxy=docker.io/
mesosphere/kube-proxy:v1.23.12_fips.0
```

12.3.8 Additional resources

- For more information about working with native Kubernetes, see the Kubernetes documentation[799].
- For a full list of attributed 3rd party software, see http://d2iq.com/legal/3rd.

12.4 Release Notes 2.3.3

DKP® version 2.3.3 was released on May 3, 2023

Download DKP

You must be a registered user and logged on to the support portal to download this product. New customers must contact their sales representative or sales@d2iq.com[800] before attempting to download or install DKP.

799 https://kubernetes.io/docs/home/
800 mailto:sales@d2iq.com

12.4.1 Release Summary

Welcome to D2iQ Kubernetes Platform (DKP) 2.3.3! This release provides fixes to reported issues, integrates changes from previous releases, and maintains compatibility and support for other packages used in DKP.

12.4.2 DKP Fixes and Updates

The following issues are corrected or resolved in this release.

12.4.2.1 Modifying Fluentd Pod Results in Failure to Deliver Logs

D2IQ-96943

When modifying fluentbit settings in the *logging-operator-logging-overrides* ConfigMap, the logging operator restarts the fluent-bit pods, as expected. However, afterwards fluentbit is not able to deliver logs to fluentd. This issue is resolved and fluentbit again delivers logs to fluentd.

12.4.2.2 Kommander Upgrade from 2.2.x to 2.3.2 Fails

D2IQ-96059

On premise upgrades from 2.2.x to 2.3.2 could fail with a segmentation fault. This issue is now resolved.

12.4.2.3 KIB 1.24.x failed to Create Ubuntu Images

D2IQ-95429

Attempting to create an Ubuntu 1804 or 2004 image with KIB 1.24.2 or 1.24.3 would fail with an image pull error. This release includes KIB 1.19.14, which corrects the error.

12.4.3 DKP Insights Fixes and Updates

12.4.4 Download Signature Files

You need to download an appropriate, signed signature file before you run FIPS validation. Verify which version of DKP you are running to ensure you are downloading the manifest that is compliant with the DKP release number on your system. You can use the FIPS validation tool to verify that specific components and services are FIPS-compliant by checking the signatures of the files against a signed signature file, and by checking that services are using the certified algorithms. Select the links in the **Manifest URL** column of the following table to obtain a valid file:

12.4.4.1 DKP version 2.3.3

Operating System version	Kubernetes version	containerd version	Manifest URL
CentOS 7.9	v1.23.12	1.14.13	v1.23.12 CentOS 7.9 Manifest[801]
Oracle 7.9	v1.23.12	1.14.13	v1.23.12 OL 7.9 Manifest[802]
RHEL 7.9	v1.23.12	1.14.13	v1.23.12 RHEL 7.9 Manifest[803]
RHEL 8.2	v1.23.12	1.14.13	v1.23.12 RHEL 8.2 Manifest[804]
RHEL 8.4	v1.23.12	1.14.13	v1.23.12 RHEL 8.4 Manifest[805]

12.4.5 Supported Versions

Any DKP cluster you attach using DKP 2.3.3 must be running a Kubernetes version in the following ranges:

Kubernetes Support	Version
DKP Minimum	1.22.0
DKP Maximum	1.23.x
DKP Default	1.23.12
EKS Default	1.22.x
AKS Default	1.23.x
GKE Default	1.22.x-1.23.x

DKP 2.3 comes with support for Kubernetes 1.23, enabling you to benefit from the latest features and security fixes in upstream Kubernetes. This release comes with approximately 47 enhancements. To read more about major features in this release, visit https://kubernetes.io/blog/2021/12/07/kubernetes-1-23-release-announcement/.

801 https://downloads.d2iq.com/dkp/fips/v2.3.3/manifest-centos-79.json.asc
802 https://downloads.d2iq.com/dkp/fips/v2.3.3/manifest-oracle-79.json.asc
803 https://downloads.d2iq.com/dkp/fips/v2.3.3/manifest-rhel-79.json.asc
804 https://downloads.d2iq.com/dkp/fips/v2.3.3/manifest-rhel-82.json.asc
805 https://downloads.d2iq.com/dkp/fips/v2.3.3/manifest-rhel-84.json.asc

12.4.6 2.3.3 Components and Applications

The following are component and application versions for DKP 2.3.3:

12.4.6.1 Components

Component Name	Version
Cluster API Core (CAPI)	1.1.3-d2iq.5
Cluster API AWS Infrastructure Provider (CAPA)	1.4.1
Cluster API Google Cloud Infrastructure Provider (CAPG)	1.1.0
Cluster API Pre-provisioned Infrastructure Provider (CAPPP)	0.9.5
Cluster API vSphere Infrastructure Provider (CAPV)	1.2.0
Cluster API Azure Infrastructure Provider (CAPZ)	1.3.2
Konvoy Image Builder	1.19.14[806]
containerd	1.4.13
etcd	3.4.13

12.4.6.2 Applications

Common Application Name	APP ID	Version	Component Versions	Helm Values	DKP Values
Centralized Grafana	centralized-grafana	34.9.3	• chart: 34.9.3[807] • prometheus-operator: 0.55.0 • grafana: 8.5.0	Link[808]	Link[809]

806 https://github.com/mesosphere/konvoy-image-builder/releases/tag/v1.19.14
807 https://artifacthub.io/packages/helm/mesosphere/kube-prometheus-stack/34.9.3
808 https://artifacthub.io/packages/helm/mesosphere/kube-prometheus-stack/34.9.3?modal=values
809 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/centralized-grafana/34.9.3/defaults/cm.yaml

Common Application Name	APP ID	Version	Component Versions	Helm Values	DKP Values
Centralized Kubecost	centralized-kubecost	0.27.0	• chart: 0.27.0[810] • kubecost: 1.96.0	Link[811]	Link[812]
Cert Manager	cert-manager	1.7.1	• chart: 1.7.1[813] • cert-manager: 1.7.1	Link[814]	Link[815]
Chartmuseum	chartmuseum	3.9.0	• chart: 3.9.0[816] • chartmuseum: 3.9.0	Link[817]	Link[818]
Dex	dex	2.9.19	• chart: 2.9.19[819] • dex: 2.31.0	Link[820]	Link[821]
Dex K8s Authenticator	dex-k8s-authenticator	1.2.14	• chart: 1.2.14[822] • dex-k8s-authenticator: 1.2.4	Link[823]	Link[824]
DKP Insights Management	dkp-insights-management	0.2.3	• chart: 0.2.3 • dkp-insights-management: 0.2.3	N/A	Link[825]

810 https://artifacthub.io/packages/helm/mesosphere-stable/kubecost/0.27.0
811 https://artifacthub.io/packages/helm/mesosphere-stable/kubecost/0.27.0?modal=values
812 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/centralized-kubecost/0.27.0/defaults/cm.yaml
813 https://artifacthub.io/packages/helm/cert-manager/cert-manager/1.7.1
814 https://artifacthub.io/packages/helm/cert-manager/cert-manager/1.7.1?modal=values
815 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/cert-manager/1.7.1/defaults/cm.yaml
816 https://artifacthub.io/packages/helm/chartmuseum/chartmuseum/3.9.0
817 https://artifacthub.io/packages/helm/chartmuseum/chartmuseum/3.9.0?modal=values
818 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/chartmuseum/3.9.0/defaults/cm.yaml
819 https://artifacthub.io/packages/helm/mesosphere-stable/dex/2.9.19
820 https://artifacthub.io/packages/helm/mesosphere-stable/dex/2.9.19?modal=values
821 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/dex/2.9.19/defaults/cm.yaml
822 https://artifacthub.io/packages/helm/mesosphere/dex-k8s-authenticator/1.2.14
823 https://artifacthub.io/packages/helm/mesosphere/dex-k8s-authenticator/1.2.14?modal=values
824 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/dex-k8s-authenticator/1.2.14/defaults/cm.yaml
825 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/dkp-insights-management/0.2.3/defaults/cm.yaml

Common Application Name	APP ID	Version	Component Versions	Helm Values	DKP Values
External DNS	external-dns	6.5.5	• chart: 6.5.5[826] • external-dns: 0.12.0	Link[827]	Link[828]
Fluent Bit	fluent-bit	0.19.24	• chart: 0.19.24[829] • fluent-bit: 1.8.15	Link[830]	Link[831]
Gatekeeper	gatekeeper	3.8.2	• chart: 3.8.1[832] • gatekeeper: 3.8.1	Link[833]	Link[834]
Gitea	gitea	5.0.9	• chart: 5.0.9[835] • gitea: 1.16.8	Link[836]	Link[837]
Grafana Logging	grafana-logging	6.28.0	• chart: 6.28.0[838] • grafana: 8.5.0	Link[839]	Link[840]
Grafana Loki	grafana-loki	0.48.5	• chart: 0.48.4[841] • loki: 2.5.0	Link[842]	Link[843]

826 https://artifacthub.io/packages/helm/bitnami/external-dns/6.5.5
827 https://artifacthub.io/packages/helm/bitnami/external-dns/6.5.5?modal=values
828 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/external-dns/6.5.5/defaults/cm.yaml
829 https://artifacthub.io/packages/helm/fluent/fluent-bit/0.19.24
830 https://artifacthub.io/packages/helm/fluent/fluent-bit/0.19.24?modal=values
831 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/fluent-bit/0.19.24/defaults/cm.yaml
832 https://artifacthub.io/packages/helm/gatekeeper/gatekeeper/3.8.1
833 https://artifacthub.io/packages/helm/gatekeeper/gatekeeper/3.8.1?modal=values
834 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/gatekeeper/3.8.2/defaults/cm.yaml
835 https://artifacthub.io/packages/helm/gitea/gitea/5.0.9
836 https://artifacthub.io/packages/helm/gitea/gitea/5.0.9?modal=values
837 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/gitea/5.0.9/defaults/cm.yaml
838 https://artifacthub.io/packages/helm/grafana/grafana/6.28.0
839 https://artifacthub.io/packages/helm/grafana/grafana/6.28.0?modal=values
840 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/grafana-logging/6.28.0/defaults/cm.yaml
841 https://artifacthub.io/packages/helm/grafana/loki-distributed/0.48.4
842 https://artifacthub.io/packages/helm/grafana/loki-distributed/0.48.4?modal=values
843 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/grafana-loki/0.48.5/defaults/cm.yaml

Common Application Name	APP ID	Version	Component Versions	Helm Values	DKP Values
Istio	istio	1.14.1	• chart: 1.14.1[844] • istio: 1.14.1	Link[845]	Link[846]
Jaeger	jaeger	2.32.2	• chart: 2.32.2[847] • jaeger: 1.34.1	Link[848]	Link[849]
Karma	karma	2.0.1	• chart: 2.0.1[850] • karma: 0.70	Link[851]	Link[852]
Kiali	kiali	1.52.0	• chart: 1.52.0[853] • kiali: 1.52.0	Link[854]	Link[855]
Knative	knative	0.4.0	• chart: 0.4.0[856] • knative: 0.22.3	Link[857]	Link[858]
Flux	kommander-flux	0.31.4	• chart: N/A • flux: 0.31.4	N/A	N/A

844 https://artifacthub.io/packages/helm/mesosphere/istio/1.14.1
845 https://artifacthub.io/packages/helm/mesosphere/istio/1.14.1?modal=values
846 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/istio/1.14.1/defaults/cm.yaml
847 https://artifacthub.io/packages/helm/jaegertracing/jaeger-operator/2.32.2
848 https://artifacthub.io/packages/helm/jaegertracing/jaeger-operator/2.32.2?modal=values
849 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/jaeger/2.32.2/defaults/cm.yaml
850 https://artifacthub.io/packages/helm/mesosphere-stable/karma/2.0.1
851 https://artifacthub.io/packages/helm/mesosphere-stable/karma/2.0.1?modal=values
852 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/karma/2.0.1/defaults/cm.yaml
853 https://artifacthub.io/packages/helm/kiali/kiali-operator/1.52.0
854 https://artifacthub.io/packages/helm/kiali/kiali-operator/1.52.0?modal=values
855 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/kiali/1.52.0/defaults/cm.yaml
856 https://artifacthub.io/packages/helm/mesosphere/knative/0.4.0
857 https://artifacthub.io/packages/helm/mesosphere/knative/0.4.0?modal=values
858 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/knative/0.4.0/defaults/cm.yaml

Common Application Name	APP ID	Version	Component Versions	Helm Values	DKP Values
Kube OIDC Proxy	kube-oidc-proxy	0.3.2	• chart: 0.3.1[859] • kube-oidc-proxy: 0.3.0	Link[860]	Link[861]
Kube Prometheus Stack	kube-prometheus-stack	34.9.3	• chart: 34.9.3[862] • prometheus-operator: 0.55.0 • grafana: 8.5.0 • prometheus: 2.34.0 • prometheus-alertmanager: 0.24.0	Link[863]	Link[864]
Kubecost	kubecost	0.27.0	• chart: 0.27.0[865] • kubecost: 1.96.0	Link[866]	Link[867]
Kubefed	kubefed	0.9.2	• chart: 0.9.2[868] • kubefed: 0.9.2	Link[869]	Link[870]
Kubernetes Dashboard	kubernetes-dashboard	5.1.1	• chart: 5.1.1[871] • kubernetes-dashboard: 2.4.0	Link[872]	Link[873]

859 https://artifacthub.io/packages/helm/mesosphere/kube-oidc-proxy/0.3.1
860 https://artifacthub.io/packages/helm/mesosphere/kube-oidc-proxy/0.3.1?modal=values
861 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/kube-oidc-proxy/0.3.2/defaults/cm.yaml
862 https://artifacthub.io/packages/helm/mesosphere/kube-prometheus-stack/34.9.3
863 https://artifacthub.io/packages/helm/mesosphere/kube-prometheus-stack/34.9.3?modal=values
864 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/kube-prometheus-stack/34.9.3/defaults/cm.yaml
865 https://artifacthub.io/packages/helm/mesosphere-stable/kubecost/0.27.0
866 https://artifacthub.io/packages/helm/mesosphere-stable/kubecost/0.27.0?modal=values
867 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/kubecost/0.27.0/defaults/cm.yaml
868 https://artifacthub.io/packages/helm/kubefed/kubefed/0.9.2
869 https://artifacthub.io/packages/helm/kubefed/kubefed/0.9.2?modal=values
870 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/kubefed/0.9.2/defaults/cm.yaml
871 https://artifacthub.io/packages/helm/k8s-dashboard/kubernetes-dashboard/5.1.1
872 https://artifacthub.io/packages/helm/k8s-dashboard/kubernetes-dashboard/5.1.1?modal=values
873 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/kubernetes-dashboard/5.1.1/defaults/cm.yaml

Common Application Name	APP ID	Version	Component Versions	Helm Values	DKP Values
Kubetunnel	kubetunnel	0.0.13	• chart: 0.0.13 • kubetunnel: 0.0.13	N/A	Link[874]
Logging Operator	logging-operator	3.17.8	• chart: 3.17.7[875] • logging-operator: 3.17.7 • logging-operator-logging: 3.17.7	Link[876]	Link[877]
Metallb	metallb	0.12.3	• chart: 0.12.3[878] • metallb: 0.8.1	Link[879]	Link[880]
MinIO Operator	minio-operator	4.4.25	• chart: 4.4.25 • minio-operator: 4.4.25	Link(see page 852)	Link[881]
NFS Server Provisioner	nfs-server-provisioner	0.6.0	• chart: 0.6.0[882] • nfs-server-provisioner: 2.3.0	Link[883]	Link[884]
Nvidia	nvidia	0.4.4	• chart: 0.4.4[885] • nvidia-device-plugin: 0.2.0	Link[886]	Link[887]

874 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/kubetunnel/0.0.13/defaults/cm.yaml
875 https://artifacthub.io/packages/helm/banzaicloud-stable/logging-operator/3.17.7
876 https://artifacthub.io/packages/helm/banzaicloud-stable/logging-operator/3.17.7?modal=values
877 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/logging-operator/3.17.8/defaults/cm.yaml
878 https://artifacthub.io/packages/helm/mesosphere-stable/metallb/0.12.3
879 https://artifacthub.io/packages/helm/mesosphere-stable/metallb/0.12.3?modal=values
880 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/metallb/0.12.3/defaults/cm.yaml
881 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/minio-operator/4.4.25/defaults/cm.yaml
882 https://artifacthub.io/packages/helm/mesosphere/nfs-server-provisioner/0.6.0
883 https://artifacthub.io/packages/helm/mesosphere/nfs-server-provisioner/0.6.0?modal=values
884 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/nfs-server-provisioner/0.6.0/defaults/cm.yaml
885 https://artifacthub.io/packages/helm/mesosphere/nvidia/0.4.4
886 https://artifacthub.io/packages/helm/mesosphere/nvidia/0.4.4?modal=values
887 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/nvidia/0.4.4/defaults/cm.yaml

Common Application Name	APP ID	Version	Component Versions	Helm Values	DKP Values
Grafana (project)	project-grafana-logging	6.28.0	• chart: 6.28.0[888] • grafana: 8.5.0	Link[889]	Link[890]
Grafana Loki (project)	project-grafana-loki	0.48.5	• chart: 0.48.4[891] • loki: 2.5.0	Link[892]	Link[893]
Prometheus Adapter	prometheus-adapter	2.17.1	• chart: 2.17.1[894] • prometheus-adapter: 0.9.1	Link[895]	Link[896]
Reloader	reloader	0.0.110	• chart: 0.0.110[897] • reloader: 0.0.110	Link[898]	Link[899]
Thanos	thanos	0.4.7	• chart: 0.4.6[900] • thanos: 0.17.1	Link[901]	Link[902]
Traefik	traefik	10.9.3	• chart: 10.9.1[903] • traefik: 2.5.6	Link[904]	Link[905]

888 https://artifacthub.io/packages/helm/grafana/grafana/6.28.0
889 https://artifacthub.io/packages/helm/grafana/grafana/6.28.0?modal=values
890 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/project-grafana-logging/6.28.0/defaults/cm.yaml
891 https://artifacthub.io/packages/helm/grafana/loki-distributed/0.48.4
892 https://artifacthub.io/packages/helm/grafana/loki-distributed/0.48.4?modal=values
893 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/project-grafana-loki/0.48.5/defaults/cm.yaml
894 https://artifacthub.io/packages/helm/prometheus-community/prometheus-adapter/2.17.1
895 https://artifacthub.io/packages/helm/prometheus-community/prometheus-adapter/2.17.1?modal=values
896 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/prometheus-adapter/2.17.1/defaults/cm.yaml
897 https://github.com/stakater/Reloader/tree/v0.0.110/README.md#helm-charts
898 https://artifacthub.io/packages/helm/stakater/reloader/0.0.110?modal=values
899 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/reloader/0.0.110/defaults/cm.yaml
900 https://artifacthub.io/packages/helm/banzaicloud-stable/thanos/0.4.6
901 https://artifacthub.io/packages/helm/banzaicloud-stable/thanos/0.4.6?modal=values
902 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/thanos/0.4.7/defaults/cm.yaml
903 https://artifacthub.io/packages/helm/traefik/traefik/10.9.1
904 https://artifacthub.io/packages/helm/traefik/traefik/10.9.1?modal=values
905 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/traefik/10.9.3/defaults/cm.yaml

Common Application Name	APP ID	Version	Component Versions	Helm Values	DKP Values
Traefik ForwardAuth	traefik-forward-auth	0.3.8	• chart: 0.3.8[906] • traefik-forward-auth: 3.1.0	Link[907]	Link[908]
Velero	velero	3.2.3	• chart: 3.2.3[909] • velero: 1.5.2	Link[910]	Link[911]

12.4.7 Known Issues and Limitations

The following items are known issues with this release.

12.4.7.1 Nvidia Feature Discovery Error

D2IQ-93676

When creating a new cluster to migrate Kaptain to version 2.1, after creating the cluster, the `nvidia-feature-discovery-gpu-feature-discovery` is in a `CrashLoopBackOff` state, with error.

Workaround

Follow these steps

1. Place the registry details in the override file, together with Nvidia.
2. Delete the current override and replaced it with the new override you just created in step 1.
3. Delete the machine to force preprovisioning.
4. Rename or delete the *.toml files from the import path directory set in `config.toml`
5. Restart containerd and GPU/Nvidia feature discovery.
6. Verify the Node now shows GPU resources.
7. Repeat these steps for all affected nodes.

12.4.7.2 Use static credentials to provision an Azure cluster

Only static credentials can be used when provisioning an Azure cluster(see page 177).

906 https://artifacthub.io/packages/helm/mesosphere/traefik-forward-auth/0.3.8
907 https://artifacthub.io/packages/helm/mesosphere/traefik-forward-auth/0.3.8?modal=values
908 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/traefik-forward-auth/0.3.8/defaults/cm.yaml
909 https://artifacthub.io/packages/helm/vmware-tanzu/velero/3.2.3
910 https://artifacthub.io/packages/helm/vmware-tanzu/velero/3.2.3?modal=values
911 https://raw.githubusercontent.com/mesosphere/kommander-applications/v2.3.2/services/velero/3.2.3/defaults/cm.yaml

12.4.7.3 When attaching GKE clusters, create a ResourceQuota to enable log collection

After you attach the GKE cluster, you can choose to deploy a stack of applications for workspace or project log collection. Once you have enabled this stack(see page 562), create a `ResourceQuota` which is required for the logging stack to function correctly. You will have to do this manually, because some DKP versions do not properly handle this by default.
Create the following resource to enable log collection:

1. Execute the following command to get the namespace of your workspace *on the management cluster*:

   ```
   kubectl get workspaces
   ```

   And copy the value under `WORKSPACE NAMESPACE` column for your workspace. This may NOT be identical to the Display Name of the `Workspace`.

2. Set the `WORKSPACE_NAMESPACE` environment variable to the name of the workspace's namespace:

   ```
   export WORKSPACE_NAMESPACE=<gkeattached-cluster-namespace>
   ```

3. Run the following command *on your attached GKE cluster* to create the resource:

   ```
   cat << EOF | kubectl apply -f -
   apiVersion: v1
   kind: ResourceQuota
   metadata:
     name: fluent-bit-critical-pods
     namespace: ${WORKSPACE_NAMESPACE}
   spec:
     hard:
       pods: "1G"
     scopeSelector:
       matchExpressions:
       - operator: In
         scopeName: PriorityClass
         values:
         - system-node-critical
   EOF
   ```

After a few minutes, log collection is available in your GKE cluster.

This workflow only creates a `ResourceQuota` in the targeted workspace. Repeat these steps if you want to deploy the logging stack to additional workspaces with GKE clusters.

12.4.7.4 Resolve issues with failed HelmReleases

There is an existing issue with the Flux helm-controller[912] that can cause HelmReleases to get "stuck" with an error message such as *Helm upgrade failed: another operation (install/upgrade/rollback) is in progress*. This can happen when the helm-controller is restarted while a HelmRelease is upgrading, installing, and so on.

Workaround

To ensure the HelmRelease error was caused by the helm-controller restarting, first try to suspend/resume the HelmRelease:

```
kubectl -n <namespace> patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[{"op":
"replace", "path": "/spec/suspend", "value": true}]'
kubectl -n <namespace> patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[{"op":
"replace", "path": "/spec/suspend", "value": false}]'
```

This might resolve the issue. If not, continue with the following steps:

You should see the HelmRelease attempting to reconcile, and then it either succeeds (with status: 'Release reconciliation succeeded') or it fails with the same error as before.

If the HelmRelease is still in the failed state, it is likely related to the helm-controller restarting. For example, if the 'reloader' HelmRelease is the one that is stuck.

To resolve the issue, follow these steps:

1. List secrets containing the affected HelmRelease name:

   ```
   kubectl get secrets -n ${NAMESPACE} | grep reloader
   ```

   ```
   kommander-reloader-reloader-token-9qd8b                          kubernetes.io/
   service-account-token   3      171m
   sh.helm.release.v1.kommander-reloader.v1                         helm.sh/
   release.v1                    1      171m
   sh.helm.release.v1.kommander-reloader.v2                         helm.sh/
   release.v1                    1      117m
   ```

   In this example, `sh.helm.release.v1.kommander-reloader.v2` is the most recent revision.

2. Find and delete the most recent revision secret. For example `sh.helm.release.v1.*.<revision>`

   ```
   kubectl delete secret -n <namespace> <most recent helm revision secret name>
   ```

3. Suspend and resume the HelmRelease to trigger a reconciliation:

912 https://github.com/fluxcd/helm-controller/issues/149

```
kubectl -n <namespace> patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[
{"op": "replace", "path": "/spec/suspend", "value": true}]'
kubectl -n <namespace> patch helmrelease <HELMRELEASE_NAME> --type='json' -p='[
{"op": "replace", "path": "/spec/suspend", "value": false}]'
```

You should see the HelmRelease is reconciled and eventually the upgrade and install succeeds.

12.4.7.5 Fluentbit disabled by default for DKP 2.3

Fluentbit is disabled by default in DKP 2.3 due to memory constraints. The amount of admin logs ingested to Loki requires additional disk space to be configured on the `grafana-loki-minio` Minio Tenant. Enabling admin logs may use around 2GB/day per node. See Configuring the Grafana Loki Minio Tenant(see page 865) for more details on how to configure the Minio Tenant.

If Fluentbit is enabled on the management cluster and you would like it to continue to be deployed after the upgrade, you must pass in the `--disable-appdeployments {}` flag to the `dkp upgrade kommander` command. Otherwise, Fluentbit is automatically disabled upon upgrade.

12.4.7.6 Configure the Grafana Loki MinIO Tenant

Additional steps are required to change the default configuration of the MinIO Tenant that is deployed with Grafana Loki, `grafana-loki-minio`. Using config overrides is not supported.

By default, the `grafana-loki-minio` MinIO Tenant is configured with 2 pools with 4 servers each, 1 volume per server, for a total of 80GB.

The MinIO usable storage capacity is always less than the actual storage amount.

Use MinIO Erasure code calculator[913] to establish the appropriate configuration for your log storage requirement.

- You are only able to expand MinIO storage by adding more MinIO server pools with the correct configuration. Modifying existing server pools does not work as MinIO does not support reducing storage capacity. See this MinIO Operator documentation[914] for details.
- This impacts all your `AppDeployment` objects that reference the `grafana-loki` Kommander application definition.
- The changes introduced by the following procedure are wiped out upon Kommander install and upgrade.

In this example, we modify the `grafana-loki-minio` MinIO `Tenant` object in `kommander-workspace` (namespace: `kommander`)

1. Use this script to clone the management git repository from the Management cluster:

913 https://min.io/product/erasure-code-calculator
914 https://github.com/minio/operator/blob/7ae1610432ad3174150f4adaab1562c3ee522468/docs/expansion.md#adding-capacity-to-a-minio-tenant-

```
export KUBECONFIG=$KUBECONFIG

PASS=$(kubectl get secrets -nkommander admin-git-credentials -oyaml -o go-
template="{{.data.password | base64decode }}")
URL=https://gitea_admin:$PASS@$(kubectl -n kommander get ingress gitea -o
jsonpath='{.status.loadBalancer.ingress[0].hostname}'):443/dkp/kommander/git/
kommander/kommander

git clone -c http.sslVerify=false $URL repo
```

2. Modify `repo/services/grafana-loki/0.48.4/minio.yaml` by appending a new server pool to `.spec.pools` field, for example:

```
# the following will add a new server pool with 4 servers
# each server is attached with 1 PersistentVolume of 50G
- servers: 4
  volumesPerServer: 1
  volumeClaimTemplate:
    metadata:
      name: grafana-loki-minio
    spec:
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 50Gi
  resources:
    limits:
      cpu: 750m
      memory: 1Gi
    requests:
      cpu: 250m
      memory: 768Mi
  securityContext:
    runAsUser: 0
    runAsGroup: 0
    runAsNonRoot: false
    fsGroup: 0
```

3. Commit the changes to local clone of the git management repository when you are done editing:

```
git add services/grafana-loki/0.48.4/minio.yaml
git commit # finish the commit message editing in editor
```

4. Ensure that it is safe to apply the change, and then push the change to management git repository:

```
git push origin main
```

5. Set your `WORKSPACE_NAMESPACE` env variable:

```
# this is an example for kommander-workspace
export WORKSPACE_NAMESPACE=kommander
```

6. Verify that the `Tenant` is modified as expected, when the grafana-loki kustomizations reconcile:

```
# this prints the .status field of the tenant
kubectl get tenants -n kommander grafana-loki-minio -o jsonpath='{ .status }' |
jq
```

7. Verify that the new `StatefulSet` is `READY`:

```
kubectl get sts -n $WORKSPACE_NAMESPACE -l v1.min.io/tenant=grafana-loki-minio

NAME                       READY   AGE
grafana-loki-minio-ss-0    4/4     144m
grafana-loki-minio-ss-1    4/4     144m
grafana-loki-minio-ss-2    4/4     15m
```

8. Restart all the `StatefulSets` that back this `Tenant`:

```
kubectl -n $WORKSPACE_NAMESPACE rollout restart sts grafana-loki-minio-ss-0
statefulset.apps/grafana-loki-minio-ss-0 restarted
kubectl -n $WORKSPACE_NAMESPACE rollout restart sts grafana-loki-minio-ss-1
statefulset.apps/grafana-loki-minio-ss-1 restarted
kubectl -n $WORKSPACE_NAMESPACE rollout restart sts grafana-loki-minio-ss-2
statefulset.apps/grafana-loki-minio-ss-2 restarted
```

9. Verify that the MinIO Pods that back this `Tenant` are all online:

```
kubectl logs -n $WORKSPACE_NAMESPACE -l v1.min.io/tenant=grafana-loki-minio
...
Verifying if 1 bucket is consistent across drives...
Automatically configured API requests per node based on available memory on the
system: 424
All MinIO sub-systems initialized successfully
Waiting for all MinIO IAM sub-system to be initialized.. lock acquired
Status:         12 Online, 0 Offline.
API: http://minio.kommander.svc.cluster.local

Console: http://192.168.202.223:9090 http://127.0.0.1:9090

Documentation: https://docs.min.io
...
```

12.4.7.7 FIPS upgrade from 2.2.x to 2.3

If upgrading a FIPS cluster, there is a bug in the upgrade of `kube-proxy` `DaemonSet` in that it does not get automatically upgraded. After completing the cluster upgrade, run the following command to finish upgrading the kube-proxy `DaemonSet`:

```
kubectl set image -n kube-system daemonset.v1.apps/kube-proxy kube-proxy=docker.io/
mesosphere/kube-proxy:v1.23.12_fips.0
```

12.4.8 Additional resources

- For more information about working with native Kubernetes, see the Kubernetes documentation[915].
- For a full list of attributed 3rd party software, see http://d2iq.com/legal/3rd.

915 https://kubernetes.io/docs/home/

13 Access documentation

The following sections describe how to access other versions of documentation:

13.1 Supported documentation

DKP 2.3 comes with consolidated documentation, where we reorganized and updated the formerly separate Konvoy and Kommander documentation into the new D2iQ Help Center[916]. You can still access all n-2 supported documentation at https://archive-docs.d2iq.com/.

13.2 Archived documentation

In accordance with our version support policy(see page 871), we regularly archive older, unsupported versions of our documentation. At this time, this includes documentation for:

- Konvoy 1.7 and below.
- Kommander 1.3 and below.

You can still access older versions of this documentation on our public GitHub repo[917].

916 http://docs.d2iq.com/
917 https://github.com/mesosphere/dcos-docs-site/tree/archive/pages/dkp/

14 Download Documentation

This PDF contains the entire DKP documentation space. The file name contains the version and date created.

(see page 870)

15 Legal Notices

D2iQ® and its licensors are the owners of all right, title, and interest in and to D2iQ software products, the documentation, all updates, upgrades, and derivative works thereto, and all intellectual property rights therein. D2iQ software products may additionally include third-party open source software.

See the D2iQ Legal page[918] for additional details.

15.1 Version support policy

D2iQ® supports N-2 of the latest `MAJOR.MINOR` version of DKP. For example, if the current GA version of DKP® is 2.3, then D2iQ supports all patch versions of DKP 2.2, and 2.1.

When the next version releases, support continues for 2.3, and 2.2. Support for DKP version 2.1.x expires. You should upgrade DKP with every new release to stay up-to-date with the latest features and bug fixes.

You can read more about our official support policy in D2iQ Support and Maintenance Terms[919].

15.1.1 Supported Kubernetes versions

Each DKP release supports a range of Kubernetes versions. Details for supported Kubernetes versions on DKP can be found in the Release Notes(see page 811).

15.1.2 Supported operating systems

Details for supported operating systems on DKP can be found in Supported Operating Systems(see page 78).

15.1.3 Features in patches

Occasionally, to make new features available at a faster rate, D2iQ releases features as part of a patch release. If the Release Notes indicate a feature you need and do not yet have, consider upgrading to the latest version to take full advantage of new features and functions.

15.1.4 Experimental status

“Experimental” means software, features, functionality, sample configurations, or other speculative content that is still under exploration, development, or testing by D2iQ. Experimental components carry no guarantee of eventual release as GA and therefore must not be used in Production Environments. Experimental components qualify for limited, Severity 4 support only and may be discontinued at any time, with or without notice.

Since Experimental components are not intended for Production Environment use, D2iQ cannot assume any responsibility for errors occurring during their use in Production. We can offer only these services in a commercially-reasonable manner, based on the availability of relevant subject matter experts (SMEs):

- General operational guidance for the Experimental component.
- Identifying and diagnosing of errors in configuration or implementation, if possible.
- Advice on preventing and recovering from failures and troubleshooting, as available.

---

918 https://d2iq.com/legal/3rd
919 https://d2iq.com/legal/support-terms

Support for Experimental components is provided on a Standard level, Severity 4 basis only.

This software is provided “as is” and without any express or implied warranties including, without limitation, the implied warranties of merchantability and fitness for a particular purpose.

15.1.4.1 Technical preview

We provide Technical Preview, or Tech Preview, features to showcase capabilities that might be added to future versions of the product. As they are not yet production-ready, the support terms are the same as defined for experimental features. Technical Preview features are not guaranteed to move forward, so could be removed from future versions of the product.

15.1.5 Support definitions

15.1.5.1 Secondary support

The following section describes D2iQ’s support for secondary applications, such as platform applications. All platform applications that D2iQ ships with DKP products are covered under secondary support:

Type	Scope Example	Support Offered
Configuration	• Guidance for base technology and DKP interoperability configuration questions and troubleshooting for different components of the DKP platform. • No support for base technology’s configuration that is unrelated to its integration with DKP. • No support for performance issues with the base technology that is unrelated to its integration with DKP.	Supported with severity 3 & 4 support terms

Type	Scope Example	Support Offered
Failure Assistance	• Assistance with installation, and upgrade failures of the service as captured in the supported DKP product upgrade pathway. • Assistance with service failures due to platform issues. For example: DKP Enterprise or DKP Essential • Support is limited to troubleshooting for root cause up to DKP product limit. Root causes that are identified to be beyond this limit will need to be pursued by the company who creates the platform application base technology. Note, if the RCA for the failure is due to a non-standard configuration or non-DKP use of the platform application, we will be unable to provide assistance beyond basic identification. • No assistance for base technology's failures that is unrelated to its integration with DKP.	Supported with all severities
Bug Fixes	• Bug fixes for service integration with DKP. • Upstream bug fixes to identified issues in the base technology of the platform application on a best effort basis. • No guarantees that our changes to upstream will be accepted. • No commitment to maintaining forks upstream.	RCA supported with all severities, Fix supported with severity 3 & 4 support terms

Type	Scope Example	Support Offered
Documentation errors	• Documentation fixes for life cycle management of services and integration with DKP. • Upstream documentation fixes to reported and identified issues in base technology of the platform application via a best effort basis. • No guarantees that our changes will be accepted.	Supported with severity 4 support terms

15.1.6 Standard level & severity definitions

To view our severity level and support terms refer to D2iQ Support and Maintenance Terms[920].

920 https://d2iq.com/legal/support-terms