Managing Kubernetes Clusters

The Cisco Container Platform web interface allows you to manage Kubernetes clusters by using the **Kubernetes Dashboard**. Once you set up the **Kubernetes Dashboard**, you can deploy applications on the authorized Kubernetes clusters, and manage the application and the cluster itself.

This chapter contains the following topics:

Accessing Kubernetes Clusters

The steps to access the Kubernetes clusters differ based on the method used for cluster creation.

This section contains the following topics:

Accessing Kubernetes Clusters on vSphere

You can access clusters on vSphere using the Kubernetes dashboard and the Kubernetes default token.

Step 1 To download the Kubeconfig file that provides you access to the Kubernetes cluster, perform these steps on the Cisco Container Platform web interface:

a) In the left pane, click **Clusters**.

b) From the drop-down list displayed under the **ACTIONS** column, choose **Download Token** to get the `Kubeconfig` file of the vSphere cluster.

The `Kubeconfig` file is downloaded to your local system.

Step 2 To get the Kubernetes default token, perform these steps in the kubectl utility:

a) SSH into the master of the tenant cluster.

b) List the Kubernetes secrets in the kube-system namespace.

```
kubectl get secrets -n kube-system
```

c) Search for the secret that has the following format:

```
default-token-XXXXX
```

d) Get the default token in one of the following ways:

- `kubectl describe secret default-token-XXXXX -n kube-system`
- `kubectl get secret default-token-XXXXX -n kube-system -o jsonpath='{.data.token}' | base64 -d`

Step 3 To set up the Kubernetes dashboard access, perform these steps:

a) In the left pane of the Cisco Container Platform web interface, click **Clusters**.
b) From the drop-down list displayed under the **ACTIONS** column, choose **Kubernetes Dashboard**.
The Kubernetes Dashboard login screen is displayed.
c) Log in to the Kubernetes Dashboard in one of the following ways:

- Using the **TOKEN**: Click the **TOKEN** radio button and enter the token from Step 2-c.
- Using the **kubeconfig** file: Click the **Kubeconfig** radio button and select the Kubeconfig file from Step 1-b.

Accessing Kubernetes Clusters for On-prem AWS IAM Enabled Clusters

For an on-prem AWS cluster that has IAM enabled, you can access clusters using the Kubernetes dashboard and the Kubernetes default token.

Step 1 To download the `Kubeconfig` file that provides you access to the Kubernetes cluster, perform these steps on the Cisco Container Platform web interface:

a) In the left pane, click **Clusters**.
b) From the drop-down list displayed under the **ACTIONS** column, choose **Download Token** to get the `Kubeconfig` file of the on-prem AWS cluster.
The `Kubeconfig` file is downloaded to your local system.

Step 2 To get the Kubernetes default token, perform these steps in the kubectl utility:

a) List the Kubernetes secrets in the `kube-system` namespace.

```
kubectl get secrets -n kube-system
```

b) Search for the secret that has the following format:

```
default-token-XXXXX
```

c) Get the default token.

```
kubectl describe secret default-token-XXXXX -n kube-system
```

Step 3 To set up the Kubernetes dashboard access, perform these steps in the Kubernetes dashboard:

a) Click the **Token** radio button.
b) In the **Enter token** field, enter the Kubernetes default token from Step 2-c.

Accessing Kubernetes Clusters on AWS EKS

You can use the `Kubeconfig` file along with the kubectl command line tool to access the clusters on AWS EKS.

Step 1 To download the `Kubeconfig` file that provides you access to the AWS EKS cluster, perform these steps on the Cisco Container Platform web interface:

a) In the left pane, click **Clusters**.

b) From the drop-down list displayed under the **ACTIONS** column, choose **Download Token** to get the `Kubeconfig` file of the AWS EKS cluster.
The `Kubeconfig` file is downloaded to your local system.

Step 2 To set up the Kubernetes dashboard access, follow the steps provided on the Dashboard Tutorial page.

Accessing Kubernetes Clusters on AKS

You can use the `Kubeconfig` file along with the kubectl command line tool to access the clusters on AKS.

Step 1 To download the `Kubeconfig` file that provides you access to the AKS cluster, perform these steps on the Cisco Container Platform web interface:

a) In the left pane, click **Clusters** and then click the **Azure** tab.

b) For the cluster whose dashboard you would like to access, from the drop-down list displayed under the **ACTIONS** column, choose **Download Kubeconfig** to get the `kubeconfig` file of the AKS cluster.
The `kubeconfig` file is downloaded to your environment.

Step 2 To set up Kubernetes dashboard access, follow these steps:

a) Create the necessary cluster role binding.

```
kubectl --kubeconfig=[location_of_kubeconfig_downloaded_from_ccp_dashboard] create
clusterrolebinding kubernetes-dashboard --clusterrole=cluster-admin
--serviceaccount=kube-system:kubernetes-dashboard
```

b) Connect to the Kubernetes dashboard using the Azure CLI.

```
az aks browse --resource-group [name_of_your_resource_group] --name [name_of_your_aks_cluster]
```

The Kubernetes dashboard is displayed in a new browser tab.

For more information, see Dashboard Tutorial.

Note In the Dashboard Tutorial page, you must follow the steps in the RBAC enabled cluster section because currently, in Cisco Container Platform, RBAC is enabled by default for all clusters on AKS.

Accessing Kubernetes Clusters on GKE

Before you begin

- Install Google Cloud SDK on your computer. For more information, see Installing Google Cloud SDK.

  Note You can choose a suitable installation option depending on the OS installed on your computer.

- After the SDK is installed, add the `GOOGLE_CLOUD_SDK_INSTALLATION_DIR/bin` path to the environment variable `PATH`.
- Setup the GCP credentials.

  For more information, see Configuring cluster access for kubectl.

You can configure your computer to access clusters on GKE in one of the following ways:

Accessing Clusters on GKE using GKE Dashboard

We recommend that you use the GKE dashboard that is available on the GCP console to view, inspect, manage, and delete resources in your clusters.

For more information, see GKE Dashboard.

Note The open-source Kubernetes dashboard is deprecated for clusters on GKE.

Accessing Clusters on GKE using Kubeconfig and Kubectl

You can use the `Kubeconfig` file along with the kubectl command line tool to access the clusters on GKE.

Generate the `kubeconfig` file in one of the following ways:

- Use GCP to generate the `kubeconfig` file in your environment.

  When you create a cluster using GCP, an entry is automatically added to the `kubeconfig` in the `$HOME/.kube/config` file.

  For more information, see Setting up kubeconfig on your machine.

- Alternatively, use Cisco Container Platform to generate the `kubeconfig` file:
  1. **a.** In the left pane, click **Clusters**, and then click the **GKE** tab.
  2. **b.** To access the cluster, from the drop-down list displayed under the **ACTIONS** column, choose **Download Kubeconfig** to get the `kubeconfig` file of the cluster.

     The `kubeconfig` file is downloaded to your environment.
  3. **c.** Edit the `kubeconfig` file to replace `$GCLOUD_SDK_PATH` with the local path where google-cloud-sdk is installed.
  4. **d.** A separate `kubeconfig` file is generated to allow access to the GKE clusters.

Monitoring Health of Cluster Deployments

It is recommended to continuously monitor the health of your cluster deployment to improve the probability of early detection of failures and avoid any significant impact from a cluster failure.

Cisco Container Platform is deployed with Prometheus and Grafana, which are configured to start monitoring and logging services automatically when a Kubernetes cluster is created.

Prometheus is an open-source systems monitoring and alerting toolkit and Grafana is an open source metric analytics and visualization suite.

Prometheus collects the data from the cluster deployment, and Grafana provides a general purpose dashboard for displaying the collected data. Grafana offers a highly customizable and user-friendly dashboard for monitoring purposes.

Note A user with *Administrator* role can view all the cluster deployments, but a user with *User* role can view only those clusters for which the user has permission to view.

Step 1 Install the **Monitoring** add-on in the tenant cluster.
For more information, see Configuring Add-ons for Clusters on vSphere.

Step 2 In the left pane, click **Clusters**, and then click on the name of the tenant cluster.

Step 3 Click on the **ADD-ONS** tab.

Step 4 Under **Monitoring**, click on **DASHBOARD**.
The Grafana login page is displayed.

Step 5 Create an SSH connection to the tenant master node and follow these steps to access Grafana. Use values of `GRAFANA_USER` and `GRAFANA_PASSWORD` to login to Grafana.

```
   export GRAFANA_USER=$(kubectl get secret ccp-monitor-grafana -n ccp -o=jsonpath='{.data.admin-user}'
 | base64 --decode)

    export GRAFANA_PASSWORD=$(kubectl get secret ccp-monitor-grafana -n ccp
-o=jsonpath='{.data.admin-password}' | base64 --decode)

    echo $GRAFANA_USER
    echo $GRAFANA_PASSWORD
```

Note It is important to either change or retain the original login credentials since the secret that was used to initialize the Grafana login may be lost or changed with future upgrades.

Step 6 On the upper-left corner of the Grafana UI, click **Home**, and then click **Kubernetes cluster monitoring (via Prometheus)** to monitor the health of the tenant cluster.

Step 7 Add Prometheus as the data source and configure the Grafana dashboard to monitor the health of your cluster deployments.

Example of Monitoring Multiple Prometheus Instances

To monitor multiple Prometheus instances you must expose Prometheus as an Ingress resource so that you can access it from a Grafana instance that is running in a different cluster.

Note The following example is valid only if Harbor is not installed.

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/add-base-url: "true"
    nginx.ingress.kubernetes.io/rewrite-target: /$1
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
name: ccp-monitor-prometheus-server
namespace: ccp
spec:
rules:
- http:
    paths:
    - backend:
        serviceName: ccp-monitor-prometheus-server
        servicePort: 443
        path: /
```

After Promethus is accessible externally from the cluster, you can add it as a new datasource in Grafana.

Monitoring Logs from Cluster Deployments

The Elasticsearch, Fluentd, and Kibana (EFK) stack enables you to collect and monitor log data from containerized applications for troubleshooting or compliance purposes. These components are automatically installed when you install Cisco Container Platform.

Fluentd is an open source data collector. It works at the backend to collect and forward the log data to Elasticsearch.

Kibana is an open source analytics and visualization platform designed to work with Elasticsearch. It allows you to create rich visualizations and dashboards with the aggregated data.

Note A user with the *Administrator* role can view all logs, but a user with *User* role can view logs for only those clusters for which the user has permission to view.

This section contains the following topics:

Viewing EFK Logs Using Kibana (Tenant Cluster)

Before you begin

Ensure that you have installed the `kubectl` utility.

Step 1 Download the Kubeconfig file of the cluster whose logs you want to view, see Downloading Kubeconfig File.

Step 2 Copy the contents of the downloaded Kubeconfig file to:

- Your local host `~/.kube/config`

- A local file and export KUBECONFIG=<*Downloaded Kubeconfig file*>

Step 3 Perform one of the following steps to access Kibana from outside a cluster:

- Create a port forward using kubectl to access Kibana from outside a cluster.

  a. Determine the pod.

  ```
  kubectl -n ccp get pods
  ```

  Example

  ```
  ccp-efk-kibana-6d7c97575c-9qxbf
  ```

  b. Open a port forward.

  Example

  ```
  kubectl port-forward -n ccp
   ccp-efk-kibana-6d7c97575c-9qxbf 5601:5601
  ```

  c. Access the Kibana UI and view the data from the target tenant cluster using a web browser.

  ```
  http://localhost:5601/app/kibana
  ```

- Run a bash script to create a Kibana login and access Kibana through Ingress from outside a cluster.

  a. Create ingress resource yaml for kibana.

  ```
  cat > kibana_ingress.yaml <<EOF
  apiVersion: extensions/v1beta1
  kind: Ingress
  metadata:
    annotations:
      kubernetes.io/ingress.class: nginx
      nginx.ingress.kubernetes.io/add-base-url: "false"
      nginx.ingress.kubernetes.io/auth-realm: Kibana
      nginx.ingress.kubernetes.io/auth-secret: ccp-kibana-basic-auth
      nginx.ingress.kubernetes.io/auth-type: basic
      nginx.ingress.kubernetes.io/rewrite-target: /
    name: ccp-kibana
    namespace: ccp
  spec:
    rules:
    - http:
        paths:
        - path: /kibana/?(.*)
          backend:
            serviceName: ccp-efk-kibana
            servicePort: 5601
  EOF
  ```

  b. Create a bash script.

  ```
  cat > ccp_kibana_ingress_setup.sh << EOF

  #!/bin/bash

  # Kibana username
  KIBANA_USERNAME=<USER>
  KIBANA_PASSWORD=<PASSWORD>

  # Configure Kibana server.basePath param
  kubectl -n ccp get cm ccp-efk-kibana -o json |
  jq '.["data"]["kibana.yml"] += "server.basePath: \"/kibana\"\n"' |
  ```

```
kubectl replace -f -

# Generating password and creating secret
KIBANA_SECRET=`htpasswd -n -b $KIBANA_USERNAME $KIBANA_PASSWORD | head -n 1`
kubectl -n ccp create secret generic ccp-kibana-basic-auth --from-literal=auth=$KIBANA_SECRET


# Creating ingress
kubectl apply -f kibana_ingress.yaml

EOF
```

c. Run the script.

```
sudo apt-get install htpasswd
chmod 700 ccp_kibana_ingress_setup.sh
./ccp_kibana_ingress_setup.sh
```

d. Restart the Kibana pod.

```
kubectl -n ccp delete $(kubectl -n ccp get pods -o name | grep ccp-efk-kibana)
```

e. Access the Kibana UI and view the data using a web browser.

```
http://<INGRESS IP>/kibana
```

For more information on customizing the Kibana UI, see the latest Kibana documentation.

Viewing EFK Logs Using Kibana (Control Plane Cluster)

Before you begin

Ensure that you have installed the `kubectl` utility.

Step 1 Access the Kubernetes cluster master node using ssh.

```
ssh ccpuser@control plane master node
sudo cat /etc/kubernetes/admin.conf
```

Step 2 Copy the contents of the downloaded Kubeconfig file to:

- Your local host `~/.kube/config`
- A local file and export KUBECONFIG=<*Full path of the Kubeconfig local file*>

For more information on setting Kubeconfig, see Configure Access to Multiple Clusters.

Step 3 Perform one of the following steps to access Kibana from outside a cluster:

- Create a port forward using kubectl to access Kibana from outside a cluster.

  a. Determine the pod.

  ```
  kubectl get pods
  ```

  Example

  ```
  ccp-efk-kibana-6d7c97575c-9qxbf
  ```

b. Open a port forward.

Example

```
kubectl port-forward ccp-efk-kibana-6d7c97575c-9qxbf 5601:5601
```

c. Access the Kibana UI and view the data from the target tenant cluster using a web browser.

```
http://localhost:5601/app/kibana
```

- Run a bash script to create a Kibana login and access Kibana through Ingress from outside a cluster.

a. Create ingress resource yaml for kibana.

```
cat > kibana_ingress.yaml <<EOF
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/add-base-url: "false"
    nginx.ingress.kubernetes.io/auth-realm: Kibana
    nginx.ingress.kubernetes.io/auth-secret: ccp-kibana-basic-auth
    nginx.ingress.kubernetes.io/auth-type: basic
    nginx.ingress.kubernetes.io/rewrite-target: /
  name: ccp-kibana
  namespace: ccp
spec:
  rules:
  - http:
      paths:
      - path: /kibana/?(.*)
        backend:
          serviceName: ccp-efk-kibana
          servicePort: 5601
EOF
```

b. Create a bash script.

```
cat > ccp_kibana_ingress_setup.sh << EOF

#!/bin/bash

# Kibana username
KIBANA_USERNAME=<USER>
KIBANA_PASSWORD=<PASSWORD>

# Configure Kibana server.basePath param
kubectl -n ccp get cm ccp-efk-kibana -o json |
jq '.["data"]["kibana.yml"] += "server.basePath: \"/kibana\"\n"' |
kubectl replace -f -

# Generating password and creating secret
KIBANA_SECRET=`htpasswd -n -b $KIBANA_USERNAME $KIBANA_PASSWORD | head -n 1`
kubectl -n ccp create secret generic ccp-kibana-basic-auth --from-literal=auth=$KIBANA_SECRET


# Creating ingress
kubectl apply -f kibana_ingress.yaml

EOF
```

c. Run the script.

```
sudo apt-get install htpasswd
chmod 700 ccp_kibana_ingress_setup.sh
./ccp_kibana_ingress_setup.sh
```

d. Restart the Kibana pod.

```
kubectl -n ccp delete $(kubectl -n ccp get pods -o name | grep ccp-efk-kibana)
```

e. Access the Kibana UI and view the data using a web browser.

```
http://<INGRESS IP>/kibana
```

For more information on customizing the Kibana UI, see the latest Kibana documentation.

Forwarding Logs to External Elasticsearch Server

Step 1 SSH to the master node of the tenant cluster.

Step 2 Edit the `ccp-efk` helmchart custom resource.

a) Open the `ccp-efk` helmchart using the visual editor.

```
kubectl edit helmchart ccp-efk -n=ccp
```

b) In `ccp-efk` helmchart custom resource, add the following lines for the external Elasticsearch server to the `spec` section.

```
set:
  localLogForwarding.enabled: "False"
  localLogForwarding.elasticsearchHost: "<_IP address of Elasticsearch server_>"
  localLogForwarding.elasticsearchPort: "<_Port number of Elasticsearch server_>"
```

c) Save the `ccp-efk` helmchart.

Step 3 Verify the custom configurations for the external Elasticsearch server in the EFK helmchart.

```
helm get values ccp-efk --namespace ccp
```

Helm-operator and fluentd pod logs also show the IP and port of the external Elasticsearch server.

Renewing Kubernetes API Certificates

By default, the Kubernetes API certificates have a validity period of one year, after which you must manually renew these certificates.

As of Kubernetes 1.17, upgrading Kubernetes will automatically renew the certificates. If you perform regular upgrades, you should not need this procedure.

To manually renew the Kubernetes API certificates for a cluster:

Step 1 SSH to a cluster master node, and check for expired (or expiring) certificates using the following command:

```
sudo kubeadm alpha certs check-expiration
```

The command shows the expiration date and time of all the certificates in the node.

Step 2 On each of the master nodes of the cluster, back up the existing certificates and configuration files:

```
sudo cp -r /etc/kubernetes/pki "$HOME/previous-certs"
```

Step 3 On each of the master nodes of the cluster, renew the certificates:

a) Renew the certificates:

```
sudo kubeadm alpha certs renew all
```

b) Verify that the cluster has new certificates:

```
sudo kubeadm alpha certs check-expiration
```

c) Generate Kubernetes configuration files:

```
sudo su
 kubeadm alpha kubeconfig user --org system:masters --client-name kubernetes-admin >
/etc/kubernetes/admin.conf
 kubeadm alpha kubeconfig user --client-name system:kube-controller-manager >
/etc/kubernetes/controller-manager.conf
 kubeadm alpha kubeconfig user --client-name system:kube-scheduler >
/etc/kubernetes/scheduler.conf
 kubeadm alpha kubeconfig user --org system:nodes --client-name system:node:$(hostname) >
/etc/kubernetes/kubelet.conf
 exit
```

d) If there is a `/etc/kubernetes/node.conf`, update its `client-certificate-data` and `client-key-data` values to match those in `/etc/kubernetes/admin.conf`.

```
sudo vi /etc/kubernetes/node.conf
```

e) If you have a file `$HOME/.kube/config`, update its `client-certificate-data` and `client-key-data` values to match those in `/etc/kubernetes/admin.conf`.

```
cp $HOME/.kube/config $HOME/.kube/config.bak vi $HOME/.kube/config
```

f) Reboot the cluster master node.

```
sudo shutdown -r now
```

g) Verify that the master node is back in the cluster.

```
kubectl get nodes
```

Step 4 Repeat Steps 2 and 3 for each cluster master node.