



# Troubleshooting Cisco Container Platform

---

This appendix describes the problems that may occur during the installation and operation of Cisco Container Platform and the possible ways of resolving these problems.

It contains the following topics:

- [Multi-master vSphere or Openstack Tenant Cluster Fails to Upgrade, on page 1](#)
- [Installation of Cisco Container Platform Fails , on page 6](#)
- [Unable to Upgrade Cisco Container Platform due to Network Misconfiguration , on page 6](#)
- [Unable to Deploy NGINX Ingress Controller Using Helm, on page 6](#)
- [Unable to Start NGINX Ingress Controller Pod, on page 7](#)
- [Unable to Power on Worker VMs after a Shutdown, on page 7](#)
- [Application Pods Crash When Using Contiv CNI in Tenant Clusters, on page 7](#)
- [How to Create Sosreports, on page 10](#)
- [Troubleshoot vSphere Operator in Cisco Container Platform, on page 10](#)
- [Troubleshoot Net Tinker in Cisco Container Platform, on page 11](#)
- [Unable to Delete EKS Clusters Properly, on page 12](#)
- [Unable to Manage Tenant Clusters due to a vCenter Password Update, on page 13](#)

## Multi-master vSphere or Openstack Tenant Cluster Fails to Upgrade

During the upgrade of a multi-master vSphere or Openstack tenant cluster, the upgrade fails because the etcd leader is reported missing.

The following error is displayed in the cloud-init logs of one of the master nodes:

```
Error: etcdserver: leader changed
```

### Recommended Solution

Before upgrading a multi-master tenant cluster, run the following scripts to ensure that the etcd leader is available during the upgrade process:

---

**Step 1** Run the `get-etcd-pod-name.sh` script on the control plane to get the name of the etcd pod.

```
get-etcd-pod-name.sh
```

```

#!/bin/bash

if [ -z "$1" ]; then
echo "usage ./get-etcd-pod-name.sh <tenant-cluster-name>"
  exit 1
fi

CLUSTER=$1

etcdNodes=$(kubectl get vsc $CLUSTER -o go-template='{{$array:= ""}}{{range $key, $value :=
.status.masterGroupStatus.nodes}}{{ $output := printf "%s%s" "etcd-" $key }}{{ $array = printf "%s
%s" $array $output }}{{ end }}{{$array}}')

echo "nodes in cluster $CLUSTER"
IFS=' ' read -ra PODS <<<"$etcdNodes"
for i in "${PODS[@]}"; do
  echo $i
done

if [ ${#PODS[@]} -eq 1 ]; then
  echo "cluster $CLUSTER has only 1 master node, no need of etcd migration to perform."
  exit 0
fi

echo "#####"
# the master node at index 2 is the last master node to upgrade
lastMasterNodeUID=$(kubectl get vsc $CLUSTER -o jsonpath={.status.masterGroupIndexUIDMap."2"})
lastMasterNodeName="etcd-"$CLUSTER"-master-gro-"$lastMasterNodeUID"
echo "migrate etcd-leader to etcd pod: $lastMasterNodeName"
echo "use the above etcd pod name as input for script to execute on tenant cluster"

```

Sample output of this script for a multi-master vSphere cluster:

**control-plane-output**

```

ccpuser@ccp800-master80bcc3ccdc:~/move-etcd-leader-20210402/control-plane$ ./get-etcd-pod-name.sh
vsc-multimaster-001
nodes in cluster vsc-multimaster-001
etcd-vsc-multimaster-001-master-gro-9a160a491e
etcd-vsc-multimaster-001-master-gro-9e3ee81be6
etcd-vsc-multimaster-001-master-gro-dd55508cf3
#####
migrate etcd-leader to etcd pod: etcd-vsc-multimaster-001-master-gro-9a160a491e
use the above etcd pod name as input for script to execute on tenant cluster

```

**Step 2** Copy the following script **move-etcd-leader.sh** and the job template **move-leader-job-ccp.yaml** to the same directory on the master node.

**move-etcd-leader.sh**

```

#!/bin/bash

set -eo pipefail

ETCD_POD=$1

if [ -z "$ETCD_POD" ]; then
  echo "usage ./move-etcd-leader.sh <ETCD_POD_NAME>"
  exit 1
fi

set -u

# info of new leader
ETCD_POD_IP=$(kubectl get po $ETCD_POD -n kube-system -o jsonpath={.status.podIP})

```

```

export NEWLEADERIP=$ETCD_POD_IP

# info of current/old leader
ETCD_PODS=$(kubectl get pods -n kube-system -l component=etcd,tier=control-plane -o
jsonpath="{.items[*]'.metadata.name'}")
IFS=' ' read -ra PODS <<<"$ETCD_PODS"
ETCD_LEADER_POD=""
for i in "${PODS[@]"; do
    STATUS=$(kubectl exec $i -n kube-system -- /bin/sh -c "etcdctl
--cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/peer.crt
--key=/etc/kubernetes/pki/etcd/peer.key --endpoints="https://127.0.0.1:2379" endpoint status")
    IFS="," read -r -a statusArray <<<"$STATUS"
    if [ "${statusArray[4]}" == " true" ]; then
        ETCD_LEADER_POD=$i
        break
    fi
done

if [ -z "$ETCD_LEADER_POD" ]; then
    echo "etcd leader pod not found, exiting"
    exit 1
fi

if [ "$ETCD_POD" == "$ETCD_LEADER_POD" ]; then
    echo "the chosen pod is already the leader, nothing to do"
    exit 0
fi

KUBENODE=$(kubectl get po $ETCD_LEADER_POD -n kube-system -o jsonpath={.spec.nodeName})
ETCDIMAGE=$(kubectl get pod $ETCD_POD -n kube-system -o jsonpath={.status.containerStatuses[0].image})

echo "future leader's IP address:" $NEWLEADERIP
echo "node chosen to run the job on:" $KUBENODE

ESCAPED_ETCDIMAGE=$(echo $ETCDIMAGE | sed 's/\/\//g')

DATE=$(date +"%s")

cat move-leader-job-ccp.yaml |
    sed 's/\$DATE/'$DATE'/g' |
    sed 's/\$NEWLEADERIP/'$NEWLEADERIP'/g' |
    sed 's/\$KUBENODE/'$KUBENODE'/g' |
    sed 's/\$ETCDIMAGE/'$ESCAPED_ETCDIMAGE'/g' | kubectl apply -f -

JOB_POD=$(kubectl get pods -n kube-system -l job-name=moveetcdleader$DATE -o
jsonpath={.items[0].metadata.name})
echo "the job's pod is kube-system/$JOB_POD"
echo "sleeping for 5 seconds to allow the pod to start"
for i in {1..5}; do
    sleep 1
    echo waited $i seconds
done
JOB_NAME=moveetcdleader$DATE

kubectl logs -f -n kube-system --pod-running-timeout=60s -l job-name=$JOB_NAME

EXIT_CODE=$(kubectl get pod -n kube-system -l job-name=$JOB_NAME -o
jsonpath="{.items[0].status.containerStatuses[0].state.terminated.exitCode}")
if [ "$EXIT_CODE" == "0" ]; then
    echo "job finished running without errors"
    echo "deleting job"
    kubectl delete job -n kube-system $JOB_NAME
fi

```

**move-leader-job-ccp.yaml**

```

apiVersion: batch/v1
kind: Job
metadata:
  name: moveetcdleader$DATE
  namespace: kube-system
spec:
  activeDeadlineSeconds: 90
  template:
    metadata:
    labels:
      jobtype: "forced-etcd-leader-migration"
    spec:
      containers:
        - name: etcdctl
          # replace with the etcd image present on the node
          image: $ETCDIMAGE
          command: ["/bin/sh", "-x", "-e", "-c"]
          args:
            - |
              ETCDCTL_OPTS="--cacert=/etc/kubernetes/pki/etcd/ca.crt
              --cert=/etc/kubernetes/pki/etcd/peer.crt --key=/etc/kubernetes/pki/etcd/peer.key
              --endpoints=https://127.0.0.1:2379\"
              # Using files instead of pipes because /bin/sh doesnt support -o pipefile
              # Get this nodes endpoint status
              nodeStatus=$(etcdctl $ETCDCTL_OPTS endpoint status)
              # etcdctl $ETCDCTL_OPTS endpoint status > nodeStatus 2>&1
              # Check if the IsLeader boolean is false
              # Array output includes leading spaces; do not remove
              nodeIsLeader=$(echo "$nodeStatus" | cut -d ',' -f5)
              if [ "$nodeIsLeader" = " false" ]; then
                # Not the leader, exit
                echo "Not the leader"
                exit 0
              fi
              # Get every node in the clusters endpoint status
              etcdctl $ETCDCTL_OPTS endpoint status --cluster > clusterStatus 2>&1
              # Print the clusterStatus
              echo "==== start etcd nodes list ===="
              while read -r line; do
                echo "$line"
              done < clusterStatus
              echo "==== end etcd nodes list ====="
              # Filter the list of status to find the first non-leader node
              while read -r line; do
                #IFS="," read -r -a statusArray <<< "$line"
                leaderIp=$(echo "$line" | cut -d ',' -f1)
                if [ "${leaderIp}" = "https://$NEWLEADERIP:2379" ]; then
                  # pull the node ID from the status for the node with this IP
                  # Read is used to remove leading spaces
                  #IFS=" " read -r transferID <<< "${statusArray[1]}"
                  transferID=$(echo "$line" | cut -d ',' -f2)
                  break
                fi
              done < clusterStatus
              if [ -z "${transferID}" ]; then
                # No other etcd nodes to transfer leadership too
                echo "No other members"
                exit 1
              fi
              # Transfer the leadership from this node to the node we found above
              etcdctl $ETCDCTL_OPTS move-leader $transferID
          volumeMounts:

```

```

        - mountPath: /etc/kubernetes/pki/etcd
          name: etcd-certs
hostNetwork: true
restartPolicy: Never
nodeSelector:
  kubernetes.io/hostname: "$KUBENODE"
tolerations:
  - effect: NoSchedule
    key: node.kubernetes.io/not-ready
    operator: Exists
  - effect: NoSchedule
    key: node-role.kubernetes.io/master
    operator: Exists
  - effect: NoSchedule
    key: node.kubernetes.io/unschedulable
    operator: Exists
  - effect: NoSchedule
    key: node.cloudprovider.kubernetes.io/uninitialized
    operator: Equal
    value: "true"
volumes:
  - hostPath:
      path: /etc/kubernetes/pki/etcd
      type: Directory
    name: etcd-certs

```

**Step 3** Run the **move-etcd-leader.sh** script on one of the master nodes of the tenant cluster.

**Note** The argument provided to this script is the output from the script executed in Step 1.

Sample output of the script executed on the chosen master node of the tenant cluster:

#### tenant-cluster-master-node

```

ccpuser@vsc-multimaster-001-master-gro-9a160a491e:~/move-etcd-leader-20210402/tenant-cluster$
./move-etcd-leader.sh etcd-vsc-multimaster-001-master-gro-9a160a491e
future leader's IP address: 10.10.96.45
node chosen to run the job on: vsc-multimaster-001-master-gro-9e3ee81be6
job.batch/moveetcdleader1620435247 created
the job's pod is kube-system/moveetcdleader1620435247-bppqs
sleeping for 5 seconds to allow the pod to start
waited 1 seconds
waited 2 seconds
waited 3 seconds
waited 4 seconds
waited 5 seconds
+ echo https://10.10.96.45:2379, 19a72824bde0e51, 3.4.10, 14 MB, false, false, 9, 15195, 15195,
+ leaderIp=https://10.10.96.45:2379
+ [ https://10.10.96.45:2379 = https://10.10.96.45:2379 ]
+ cut -d , -f2
+ echo https://10.10.96.45:2379, 19a72824bde0e51, 3.4.10, 14 MB, false, false, 9, 15195, 15195,
+ transferID= 19a72824bde0e51
+ break
+ [ -z 19a72824bde0e51 ]
+ etcdctl --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/peer.crt
--key=/etc/kubernetes/pki/etcd/peer.key --endpoints="https://127.0.0.1:2379" move-leader 19a72824bde0e51

Leadership transferred from 4a4da6f515bfe1ee to 19a72824bde0e51
job finished running without errors
deleting job
job.batch "moveetcdleader1620435247" deleted

```

After both these scripts are executed successfully, you can perform the upgrade of the tenant cluster as usual.

## Installation of Cisco Container Platform Fails

If installation of Cisco Container Platform fails, you can reattempt the installation.

### Recommended Solution

Reboot the installer VM and then access the installer UI again.

In case you want to update an OVA parameter on the installer node, for example, update the **CIDR for Kubernetes pod network** parameter, you can follow these steps:

1. From the right pane of the vSphere Web Client, navigate to the installer VM.
2. Right-click the installer VM and choose **Power off**.  
The installer VM is turned off.
3. Right-click the installer VM and choose **Edit Settings**.  
The **Edit Settings** dialog box appears.
4. Click the **vApp Options** tab, and then open and update the required property value.
5. Click **OK**.
6. From the right pane, right-click the installer VM and choose **Power on**.  
The installer VM is turned on. After the installer VM is turned on, the URL of the installer appears on the vCenter **Web console**.
7. Obtain the URL from the vCenter **Web console** and use a browser to access the installer UI to continue with the installation.

## Unable to Upgrade Cisco Container Platform due to Network Misconfiguration

When you enter a wrong IP address range for the Control Plane in the **Verify Network** screen of the **Upgrade** wizard, the following error message is appears:

```
Cannot patch address pool <uuid> with data: <some-data>
```

### Recommended Solution

You must go back to the **Verify Network** screen of the **Upgrade** wizard and configure the IP address range for the Control Plane again.

For more information, see [Upgrading Cisco Container Platform Control Plane](#).

## Unable to Deploy NGINX Ingress Controller Using Helm

When deploying the NGINX Ingress controller using Helm fails as RBAC is not configured in Helm, the following error message appears:

It seems the cluster it is running with Authorization enabled (like RBAC) and there is no permissions for the ingress controller. Please check the configuration

#### Recommended Solution

As Cisco Container Platform uses RBAC for authentication, Helm also needs to be configured to use RBAC.

Enable the RBAC parameter in Helm using the following command:

```
--set rbac.create=true
```

## Unable to Start NGINX Ingress Controller Pod

When kube-proxy is used, setting both the `controller.service.externalIPs` and `controller.hostNetwork` variables to `true` for the NGINX-Ingress chart results in an invalid configuration.

Both kube-proxy and NGINX uses port 80 for communication, causing a port conflict, and the NGINX Ingress controller pod is set to the `CrashLoopBackOff` state.

The following error message appears:

```
Port 80 is already in use. Please check the flag --http-port
```

#### Recommended Solution

Ensure that both the `controller.service.externalIPs` and `controller.hostNetwork` variables are not set to `true` at the same time.

## Unable to Power on Worker VMs after a Shutdown

Worker VMs may fail to power on after a shutdown and the following error message appears:

```
File system specific implementation of LookupAndOpen[file] failed.
```

#### Recommended Solution

Follow these steps to resolve the problem:

1. From the left pane, click the VM that you want to power on.
2. From the right pane, from the **Actions** drop-down list, choose **Edit Settings**.  
The **Edit Settings** window displays the multiple hard disks of the VM.
3. Except for the primary hard disk (Hard disk 1), click each hard disk, and then click the **Remove** icon.  
Ensure that the **Delete files from datastore** check box is not checked.
4. Click **OK**.

## Application Pods Crash When Using Contiv CNI in Tenant Clusters

When you use Contiv as the CNI for a tenant cluster, you need to ensure that the application pods that need HugePages must have the following section in the pod manifest. Otherwise, the pods may crash.

```
resources:
  limits:
    hugepages-2Mi: 512Mi
    memory: 512Mi
```

The preceding section in the pod manifest limits 512 MB in memory for HugePages for the pod. It allocates 256 HugePages, with each HugePage having 2MB size.

HugePages are allocated to the pods only if you have enabled HugePages on the host. Otherwise, the HugePage allocation in the pod manifest is ignored by Kubernetes. The following table shows the Cisco Container Platform CNIs that use HugePages.

Cisco Container Platform CNI	Use HugePages
Contiv	Yes
ACI	No
Calico	No

## Example of Allocating HugePages for Applications

**Step 1** Check the total and free HugePages on the worker nodes. Each HugePage is 2048 KB in size.

```
$ grep -i huge /proc/meminfo
AnonHugePages: 0 kB
ShmemHugePages: 0 kB
HugePages_Total: 1024
HugePages_Free: 972
HugePages_Rsvd: 0
HugePages_Surp: 0
Hugepagesize: 2048 kB

$ sudo sysctl -a | grep -i huge
vm.hugepages_treat_as_movable = 0
vm.hugetlb_shm_group = 0
vm.nr_hugepages = 1024
vm.nr_hugepages_mempolicy = 1024
vm.nr_overcommit_hugepages = 0
```

**Step 2** If the host has less HugePages, increase the HugePages allocation.

```
sudo su
echo 2048 > /proc/sys/vm/nr_hugepages

# Check the increased number of HugePages
cat /proc/sys/vm/nr_hugepages
grep -i huge /proc/meminfo
sudo sysctl -a | grep -i huge
```

**Note** You need to perform these steps on all the hosts.

**Step 3** Create the `bookinfo.yaml` file that allocates HugePages to the `reviews-v1` pod.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: reviews-v1
spec:
```



```

template:
  metadata:
    labels:
      app: reviews
      version: v1
  spec:
    containers:
      - name: reviews
        image: istio/examples-bookinfo-reviews-v1:1.5.0
        imagePullPolicy: IfNotPresent
        resources:
          limits:
            hugepages-2Mi: 512Mi
            memory: 512Mi
        ports:
          - containerPort: 9080
    
```

**Step 4** Deploy bookinfo.yaml and check usage of HugePages.

```

$ kubectl create -f istio- $\$$ ISTIO_VERSION/samples/bookinfo/kube/bookinfo.yaml
deployment.extensions "reviews-v1" created

$ kubectl get pods | grep reviews
reviews-v1-6f56455f68-t6p8s          1/1      Running    0          3m

# Check usage of HugePages by the pods
$ kubectl describe pod reviews-v1-6f56455f68-t6p8s | grep -i '^Name: \|Image: \|huge \|mem'
Name:          reviews-v1-6f56455f68-t6p8s
Image:         istio/examples-bookinfo-reviews-v1:1.5.0
hugepages-2Mi: 512Mi
memory:        512Mi
hugepages-2Mi: 512Mi
memory:        512Mi

# Check usage of HugePages on each host
$ grep -i huge /proc/meminfo
AnonHugePages:          0 kB
ShmemHugePages:        0 kB
HugePages_Total:       1024
HugePages_Free:        972
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:          2048 kB

$ sudo sysctl -a | grep -i huge
vm.hugepages_treat_as_movable = 0
vm.hugetlb_shm_group = 0
vm.nr_hugepages = 1024
vm.nr_hugepages_mempolicy = 1024
vm.nr_overcommit_hugepages = 0
    
```

**Step 5** Check the decrease of the HugePages\_Free field in the output when the reviews-v1 pod is using HugePages.

```
grep -i huge /proc/meminfo
```

## How to Create Sosreports

Sosreports are used by support engineers for troubleshooting customer support issues. They contain system log files, configuration details, and system information from your Cisco Container Platform environment.



### Note

- For Control Plane issues, you need to run the sosreport from the Control Plane master VM, if available.
- For tenant cluster issues, you need to run the sosreport from the Control Plane master VM and the tenant plane master VM.
- For network issues impacting pods on a particular worker, you need to run the sosreport from the impacted tenant worker node.

Follow these steps to create an sosreport:

**Step 1** ssh to the VM.

**Step 2** Run sosreport on the node of your choice.

```
sudo sosreport
```

The sosreport is created and saved in the following location:

```
/tmp/sosreport-xxxxxx.tar.xz
```

**Step 3** Validate the sosreport file using the following checksum:

```
xxxxxxxxxx
```

**Step 4** Securely transfer the sosreport file to your customer representative.

The file transfer method can vary depending on your deployment environment. For example, you can use Secure Copy (SCP) for Portable Operating System Interface systems (POSIX) and Windows Secure Copy (WinSCP) for windows clients. For more information, refer to [Uploading Files to Cisco Technical Assistance Center \(TAC\)](#).

## Troubleshoot vSphere Operator in Cisco Container Platform

Follow these steps to troubleshoot the vSphere operator in Cisco Container Platform:

**Step 1** Check the status of the pod, logs, and CRD of the vSphere operator.

```
$ kubectl get pods --all-namespaces | grep 'NAME\|vsphere-operator'
NAMESPACE   NAME                                     READY   STATUS    RESTARTS   AGE
default     kaas-ccp-vsphere-operator-788487bc68-h47dh  1/1     Running   2           156m

kubectl logs kaas-ccp-vsphere-operator-788487bc68-h47dh --all-containers=true

$ kubectl get crds | grep vsphereclusters
vsphereclusters.vsphere.ccp.cisco.com      2019-07-15T18:16:45Z
```

```
$ kubectl get vsphereclusters
NAME          AGE
vhosakot-vs   2h
```

- Step 2** Generate sosreport on the master node of the cluster running vSphere operator.  
See also, [How to Create Sosreports, on page 10](#).

```
cd <sosreport directory>/sos_commands/kubernetes/
cat vsphereclusters.vsphere.ccp.cisco.com
```

- Step 3** Check the status of VsphereCluster CR in the sosreport.

## Troubleshoot Net Tinker in Cisco Container Platform

Follow these steps to troubleshoot the Net Tinker operator in Cisco Container Platform:

- Step 1** Check the status of the pod, logs, and CRDs of the net tinker.

```
$ kubectl get pods --all-namespaces | grep 'NAME\|tinker'
NAMESPACE      NAME                                     READY   STATUS    RESTARTS   AGE
default        ccp-tinker-manager-85cf7fffd5-mnc24     2/2     Running   0           158m

kubectl logs ccp-tinker-manager-85cf7fffd5-mnc24 --all-containers=true

$ kubectl get crds | grep net.ccp.cisco.com
clusternetworks.net.ccp.cisco.com      2019-07-15T18:17:15Z
cnis.net.ccp.cisco.com                 2019-07-15T18:17:15Z
ipaddresses.net.ccp.cisco.com           2019-07-15T18:17:15Z
ipallocators.net.ccp.cisco.com         2019-07-15T18:17:15Z
metallbs.net.ccp.cisco.com             2019-07-15T18:17:15Z
netconfigs.net.ccp.cisco.com           2019-07-15T18:17:15Z
nginxingresses.net.ccp.cisco.com       2019-07-15T18:17:15Z

kubectl get clusternetworks,cnis,ipaddresses,ipallocators,metallbs,netconfigs,nginxingresses
```

- Step 2** Generate sosreport on the master node of the cluster running net tinker.  
See also, [How to Create Sosreports, on page 10](#).

```
cd <sosreport directory>/sos_commands/kubernetes/

# find . | grep
'clusternetworks\|cnis\|ipaddresses\|ipallocators\|metallbs\|netconfigs\|nginxingresses'
./cnis.net.ccp.cisco.com
./clusternetworks.net.ccp.cisco.com
./netconfigs.net.ccp.cisco.com
./ipaddresses.net.ccp.cisco.com
./ipallocators.net.ccp.cisco.com
./nginxingresses.net.ccp.cisco.com
./metallbs.net.ccp.cisco.com

find . | grep 'clusternetworks\|cnis\|ipaddresses\|ipallocators\|metallbs\|netconfigs\|nginxingresses'
| xargs cat
```

- Step 3** Check status of CRs of the net tinker in the sosreport.

# Unable to Delete EKS Clusters Properly

Orphaned AWS resources in your environment can cause errors when an EKS cluster does not get cleaned up properly. You can use the `ccpeksctl` utility library to delete the orphaned AWS resources.

The binary `ccpeksctl` is located in the root directory of the `ccp-eks-operator` pod.

## Recommended Solution

Ensure that you have access to the Cisco Container Platform control plane nodes.

Follow these steps to execute the binary:

1. Identify the EKS operator pod deployed in the Kubernetes cluster.

```
$ kubectl get pods
```

2. Access and execute the binary from the pod.

```
$ kubectl exec ccp-eks-operator-7fd7cf9646-gjw27 -- ./ccpeksctl -help
```

## Examples

- Display the documentation of the arguments used in the `ccpeksctl` utility.

```
$ ./ccpeksctl -help
usage ccpeksctl ARGS
  -help           Print this help
  -dryrun         Optional. Default value to false.
  -operation      Default value "list". Allowed values ["list","delete"].
  -uuid          Required. UUID value from the eks CR.
                  This value can be obtained from "Tags" section for
any of the AWS resources provisioned by Cisco Container Platform. Check the value for
key "ccp-cluster-id".
  -provider      Required. Kubernetes secret name stored containing aws
credentials.
  -region        Optional. Default value to "us-west-2" AWS region where the
resources are located.
```

- List the AWS resources with a given UUID.

```
$ ./ccpeksctl -operation list -provider eks-provider-key -region us-west-2 -uuid
<uuid-id-from-eks-cr>
```

Resource Type	Resource Status	Resource Name
eks	ACTIVE	eks-cluster-name
ec2	running	i-0ff7cc1e5404f9385
ec2	running	i-0ebe882d8e2b37bcc
subnet		subnet-023f5744de56fb436
subnet		subnet-05d5d3ab8f77f5084
subnet		subnet-0a50312f228b576ec
vpc	available	vpc-0e9c996a97f1ca69a
rtb		rtb-0b654dc7e1a86073a
rtb		rtb-09346eaf83b16094c
rtb		rtb-09de12ff95ec94db1
nat	available	nat-08e5c13eb71a5c6e4
nat	available	nat-03e3e7d563844f571
nat	available	nat-0b85fdb963b27494f
igw		igw-05cd989ac1b06f05a

```

sg                                sg-0dfb973f19c8b7bbd
eip                                eipalloc-0797eab6930c175b5    100.21.234.225
eip                                eipalloc-0077504a99fafd655    35.155.169.195
eip                                eipalloc-07218a8786b742551    35.161.22.10
cf                                CREATE_COMPLETE
eks-cluster-name-group1-networkconfig-iam-bb8d3aeb-dd41-4968-8620-50ab7865fd49

```

- Test your delete command to verify the AWS resources that will be deleted.

```

$ ./ccpeksctl -operation delete -dryrun -provider eks-provider-key -region us-west-2
-uuid <uuid-id-from-eks-cr>

```

- Delete the AWS resources tagged with UUID.

```

$ ./ccpeksctl -operation delete -provider eks-provider-key -region us-west-2 -uuid
<uuid-id-from-eks-cr>

```

## Unable to Manage Tenant Clusters due to a vCenter Password Update

If the password of the vCenter account used to install the control plane is changed, Cisco Container Platform cannot connect to the vCenter server. Cluster creation and cluster reboot failures may occur for existing control plane and tenant clusters.

### Recommended Solution

For v3 tenant clusters, you need to change the vCenter password in the Cisco Container Platform control plane, and the changes are updated in the `cloud-config` secret. However, for v2 tenant clusters, in addition to updating the vCenter password in the Cisco Container Platform control plane, you need to update the `cloud-config` secret that stores the vCenter password.

Follow these steps to resolve the problem:

**Step 1** Log in to Cisco Container Platform and go to **Infrastructure Providers**.

**Step 2** Click **Edit** under actions and change the vCenter password to reflect the password of the vCenter account.

**Step 3** Update the `cloud-config` secret that stores the vCenter password:

**Note** This step applies only for v2 tenant clusters.

- SSH to the master node of the control plane.
- Encode the new vCenter password.

```

$ echo -n "<vcenter-password>" | base64 -w 0

```

- Open the `cloud-config` secret in a text editor.

```

$ kubectl -n kube-system edit secret <secret-name>

```

Where, the `secret-name` is `/etc/kubernetes/pki/cloud-config`

- In the `cloud-config` secret, replace the vCenter password with the newly encoded password.
- Restart the kubelet to use the changed password.

```
$ sudo systemctl restart kubelet.service
```

---