Cisco Container Platform 4.0.0 API Guide

First Published: June 05, 2019

Cisco Systems, Inc.
www.cisco.com

Cisco has more than 200 offices worldwide.
Addresses, phone numbers, and fax numbers
are listed on the Cisco website at
www.cisco.com/go/offices.

Abstract
The Cisco Container Platform 4.0.0 API Guide gives information on Cisco Container Platform APIs and development features.

Cisco Container Platform 4.0.0 API Guide

Contents

1 Overview

Cisco Container Platform API provides REST API as a language-agnostic programmatic interface for applications to send requests to a Cisco Container Platform deployment.

An API conforms to the RESTful conventions and is defined by using resource and methods. A resource is a collection of information that is identified by a Uniform Resource Identifier (URI). For example, `providerclientconfig` is a resource that is used to represent configuration information to connect to an infrastructure provider such as vCenter. Methods are HTTP methods that are exposed for a resource. The commonly used HTTP methods are POST, GET, PATCH, PUT and DELETE.

2 Accessing Cisco Container Platform API

You can access the Cisco Container Platform APIs using the following URL:
`https://<CCP IP>/2/swaggerapi`

Where, `<CCP IP>` is the virtual IP address that you provided during the installation of Cisco Container Platform. It is the Ingress Controller LoadBalancer IP address.

3 Key Concepts

3.1 Provider Client Configuration

Cisco Container Platform connects to infrastructure providers such as vCenter to create and manage Virtual Machines that are used for Kubernetes Clusters. The configuration information to connect to the infrastructure provider is represented by a `providerclientconfig` resource.

3.2 Cluster

Cisco Container Platform automates the creation and lifecycle operations for Kubernetes Clusters. Each Kubernetes Cluster corresponds to a cluster resource type in Cisco Container Platform. It is identified by name for GET methods allowing you to poll the status of a Kubernetes cluster before its creation is complete. All other methods on a cluster object identify the cluster by its UUID in the URI.

3.3 User Management and Authorization

3.3.1 LDAP and Local Users

Cisco Container Platform supports Active Directory users and local users. Active directory configuration and authorization correspond to the ldap resource type in Cisco Container Platform. Local User management and authorizations correspond to the localusers resource type.

3.4 Subnets and Virtual IP Address Pools

Cisco Container Platform enables you to select an existing network, create a subnet in that network, and then create a Cisco Container Platform Virtual IP Address (VIP) pool within that subnet.
VIP pools are reserved ranges of IP addresses that are assigned as virtual IP addresses within the Cisco Container Platform clusters. Subnets correspond to network_service/subnets resource and VIP pools are a sub-resource of subnets of the type pools.

4 Examples of API Use Cases for vSphere Clusters

4.1 Creating vSphere Tenant Clusters

Before you Begin

Ensure that curl and jq are installed on your client machine.

Procedure

1. Export Cisco Container Platform Virtual IP to the MGMT_HOST environment variable.

   Command
   export MGMT_HOST=<Control Plane VIP>

   Example
   `export MGMT_HOST=10.20.30.40`

2. Obtain a cookie using the username and password for your Cisco Container Platform instance.

   Command
   curl -k -c cookie.txt -H "Content-Type:application/x-www-form-urlencoded" -d 'username=admin&password=<Password from the installer>' https://$MGMT_HOST/2/system/login/

   Example
   ```
   curl -k -c cookie.txt -H "Content-Type:application/x-www-form-urlencoded" -d 'username=admin&password=<Password from the installer>' https://$MGMT_HOST/2/system/login/
   ```

3. Get list of Provider Client Configurations.

   Command
   curl -sk -b cookie.txt -H "Content-Type: application/json" https://$MGMT_HOST/2/providerclientconfigs/ | jq '.[].uuid'

   Example
   ```
   curl -sk -b cookie.txt -H "Content-Type: application/json" https://$MGMT_HOST/2/providerclientconfigs/  | jq '.[].uuid'
   "fb53eae8-d973-4644-b13f-893949154a22"
   ```

4. Configure the provider client that you want to use.

   Command
   export PCC=<Selected Provider Client Configuration>

   Example
   ```
   export PCC=fb53eae8-d973-4644-b13f-893949154a22
   ```

5. Get the list of datacenters.

   Command
   curl -sk -b cookie.txt https://$MGMT_HOST/2/providerclientconfigs/${PCC}/vsphere/datacenter | jq '.Datacenters[]'

   Example
   ```
   curl -sk -b cookie.txt https://$MGMT_HOST/2/providerclientconfigs/${PCC}/vsphere/datacenter | jq  '.Datacenters[]'
   "RTP09"
   ```

6. Configure the datacenter that you want to use.

   Command
   export DCC=<from list of DataCenters>

   Example
   ```
   export DCC=RTP09
   ```

7. Get the list of tenant image VMs.

   Command
   curl -sk -b cookie.txt https://$MGMT_HOST/2/providerclientconfigs/${PCC}/vsphere/datacenter/${DCC}/vm | jq '.VMs[]| select(.| startswith("ccp-tenant-image"))'| sort -u

Example

```
curl -sk -b cookie.txt
https://$MGMT_HOST/2/providerclientconfigs/${PCC}/vsphere/datac
enter/${DCC}/vm | jq '.VMs[]| select(.| startswith("ccp-tenant-
image"))'| sort -u
```

```
"ccp-tenant-image-1.13.5-4.0.0.ova"
```

```
"ccp-tenant-image-1.12.7-4.0.0.ova"
```

8. Configure the name of the VM image that you want to use.

   Command
   export VM=<from list of VMs>

   Example

   ```
   export VM=ccp-tenant-image-1.12.3-3.1.0.ova
   ```

9. Get the list of networks.

   Command
   curl -sk -b cookie.txt
   https://$MGMT_HOST/2/providerclientconfigs/${PCC}/vsphere/datacenter/${DCC}/network| jq '.Networks[]'

   Example

   ```
   curl -sk -b cookie.txt
   https://$MGMT_HOST/2/providerclientconfigs/${PCC}/vsphere/datac
   enter/${DCC}/network| jq '.Networks[]'
   ```

   ```
   "r9-hx2-ccp"
   "Storage Controller Data Network"
   "k8-priv-iscsivm-network"
   ```

10. Configure the network that you want to use.

    Command
    export NETWORK=<From list of Networks>

    Example

    ```
    export NETWORK=r9-hx2-ccp
    ```

11. Get the list of clusters.

    Command
    curl -sk -b cookie.txt
    https://$MGMT_HOST/2/providerclientconfigs/${PCC}/vsphere/datacenter/${DCC}/cluster| jq '.Clusters[]'

    Example

    ```
    curl -sk -b cookie.txt
    https://$MGMT_HOST/2/providerclientconfigs/${PCC}/vsphere/datac
    enter/${DCC}/cluster| jq '.Clusters[]'
    ```

    ```
    "r9-hx2"
    ```

12. Configure the name of the cluster you want to use.

    Command
    export CLUSTER=<from list of clusters>

    Example

    ```
    export CLUSTER=r9-hx2
    ```

13. Get the list of pools.

    Command
    curl -sk -b cookie.txt https://$MGMT_HOST/2/providerclientconfigs/${PCC}/vsphere/datacenter/${DCC}/cluster/${CLUSTER}/pool| jq ".Pools[]"

    Example
    ```
    curl -sk -b cookie.txt
    https://$MGMT_HOST/2/providerclientconfigs/${PCC}/vsphere/datac
    enter/${DCC}/cluster/${CLUSTER}/pool| jq ".Pools[]"
    "Resources"
    "Resources/Infrastructure"
    ```

14. Configure the vSphere resource pool you want to use.

    Command
    export POOL=<from list of Pools>

    Example
    ```
    export POOL=Resources
    ```

15. Get the list of datastores.

    Command
    curl -sk -b cookie.txt https://$MGMT_HOST/2/providerclientconfigs/${PCC}/vsphere/datacenter/${DCC}/datastore | jq -r '.Datastores[]| select(.| startswith("SpringpathDS")|not)'

    Example
    ```
    curl -sk -b cookie.txt
    https://$MGMT_HOST/2/providerclientconfigs/${PCC}/vsphere/datac
    enter/${DCC}/datastore | jq -r '.Datastores[]| select(.|
    startswith("SpringpathDS")|not)'

    ds1
    ISOs
    Hxdump
    r9-hx2-datastore-1
    ```

16. Configure the datastore that you want to use.

    Command
    export DATASTORE=<from list of datastores>

    Example
    ```
    export DATASTORE=r9-hx2-datastore-1
    ```

17. Configure a name for the tenant cluster.

    Note: The cluster name must start with an alphanumeric character (a-z, A-Z, 0-9). It can contain a combination of hyphen (-) symbols and alphanumeric characters (a-z, A-Z, 0-9). The maximum length of the cluster name is 46 characters.

    Command
    export NAME=<Name of cluster>

    Example
    ```
    export NAME=tc4
    ```

18. Configure a username to remotely access cluster nodes with a given sshkey.

Command
export USER=<Username>

Example
```
export USER=ccpuser
```

19. Configure the ssh public key for remote access.

    Command
    export SSHKEY=<Selected ssh public key for remote access>

    Example
    ```
    export SSHKEY=`head -1 ~/.ssh/id_rsa.pub`
    ```

    Note: If there is no `public key file, please run ssh-keygen to create a key pair.`

20. Get the list of subnets.

    Command
    curl -sk -b cookie.txt -H "Content-Type: application/json" https://$MGMT_HOST/2/network_service/subnets/ | jq -r '.[0].uuid'

    Example
    ```
    curl -sk -b cookie.txt -H "Content-Type: application/json" https://10.20.30.40:32442/2/network_service/subnets/ | jq -r '.[0].uuid'
    ```

    ```
    "842e4baf-4877-4330-a3e3-4249983922a4"
    ```

21. Configure the subnet for the cluster.

    Command
    export SUBNET=<From the list of subnets>

    Example
    ```
    export SUBNET=842e4baf-4877-4330-a3e3-4249983922a4
    ```

22. Get the list of VIP pools in the subnet that you have chosen.

    Command
    curl -sk -b cookie.txt -H "Content-Type: application/json" https://$MGMT_HOST/2/network_service/subnets/${SUBNET}/pools| jq -r '.[0].uuid'

    Example
    ```
    curl -sk -b cookie.txt -H "Content-Type: application/json" https://10.20.30.40:32442/2/network_service/subnets/${SUBNET}/pools| jq -r '.[0].uuid'
    ```

    ```
    "fef830ce-dc92-46fe-8acb-01eaa539dc46"
    ```

23. Select the appropriate VIP pool if there are multiple options.

    Command
    export VIP_POOL=<From the list of pools>

    Example
    ```
    export VIP_POOL=fef830ce-dc92-46fe-8acb-01eaa539dc46
    ```

24. Copy and paste the following code to create a cluster json payload.

    ```
    #_____________________________________
    cat <<EOF > cluster_create.json
    {
    ```

```
  "provider_client_config_uuid": "${PCC}",
  "type": 1,
  "cluster": "${CLUSTER}",
  "name": "${NAME}",
  "description": "",
  "workers": 2,
  "masters": 1,
  "vcpus": 2,
  "memory": 8192,
  "datacenter": "${DCC}",
  "datastore": "${DATASTORE}",
  "networks": [
    "${NETWORK}"
  ],
   "ingress_vip_pool_id": "${SUBNET}",
   "load_balancer_ip_num": 1,
         "resource_pool": "${CLUSTER}/${POOL}",
      "template": "${VM}",
      "ssh_user": "${USER}",
      "ssh_key": "${SSHKEY}",
      "deployer_type": "kubeadm",
      "kubernetes_version": "1.11.3",
      "deployer": {
      "provider_type": "vsphere",
      "provider": {
        "vsphere_datacenter": "${DCC}",
        "vsphere_datastore": "${DATASTORE}",
        "vsphere_client_config_uuid": "${PCC}",
        "vsphere_working_dir": "\/${DCC}\/vm"
         }
      }
}
EOF

#————————————————————————————
```

25. Edit the `cluster_create.json` file to modify the number of workers, CPUs, memory, Kubernetes version, or description as needed.

26. Create a tenant cluster.

    Command
    curl -sk -X POST -b cookie.txt -H "Content-Type: application/json" -d @cluster_create.json https://$MGMT_HOST/2/clusters | tee output.txt | jq '.name,.uuid,.state'

    Example

    ```
    curl -sk -X POST -b cookie.txt -H "Content-Type:
    application/json" -d @cluster_create.json
    https://$MGMT_HOST/2/clusters | tee output.txt | jq
    '.name,.uuid,.state'

    "tc4"

    "8ccaa3a1-8a11-4996-9224-5723b7ecfdfd"

    "READY"
    ```

27. Configure the tenant cluster UUID.

    Command
    #export TC=<UUID of the selected tenant cluster>

Example

```
export TC=8ccaa3a1-8a11-4996-9224-5723b7ecfdfd
```

28. Download the KUBECONFIG environment file.

    Command

    curl -sk -b cookie.txt https://$MGMT_HOST/2/clusters/${TC}/env -o ${TC}.env

    Example

    ```
    curl -sk -b cookie.txt https://$MGMT_HOST/2/clusters/${TC}/env
    -o ${TC}.env
    ```

29. Export the config file to KUBECONFIG environment variable.

    Command

    export KUBECONFIG=./${TC}.env

    Example

    ```
    export KUBECONFIG=./${TC}.env
    ```

30. View nodes on a tenant cluster.

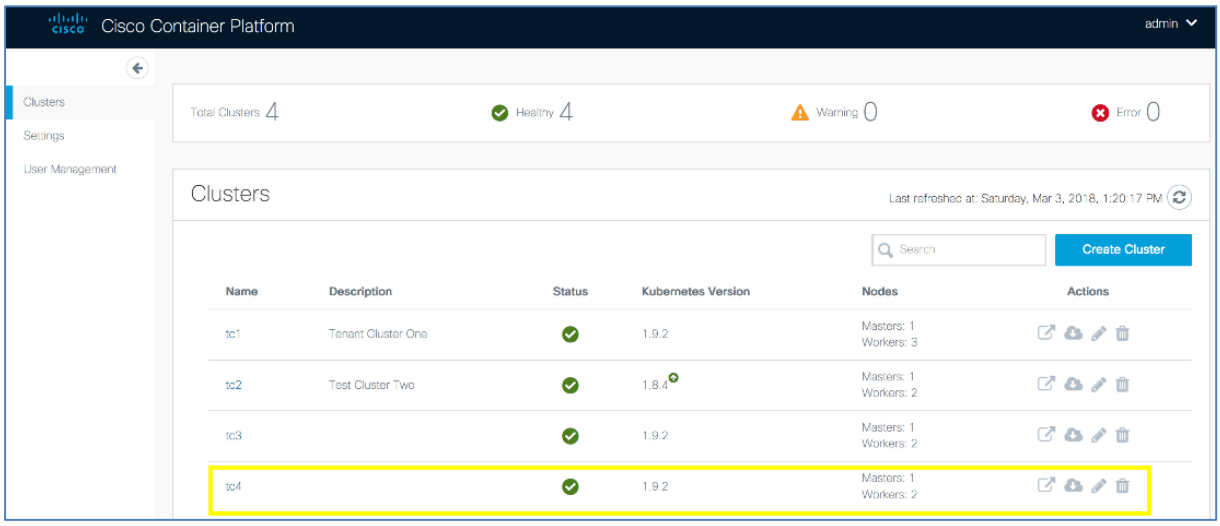
    Command

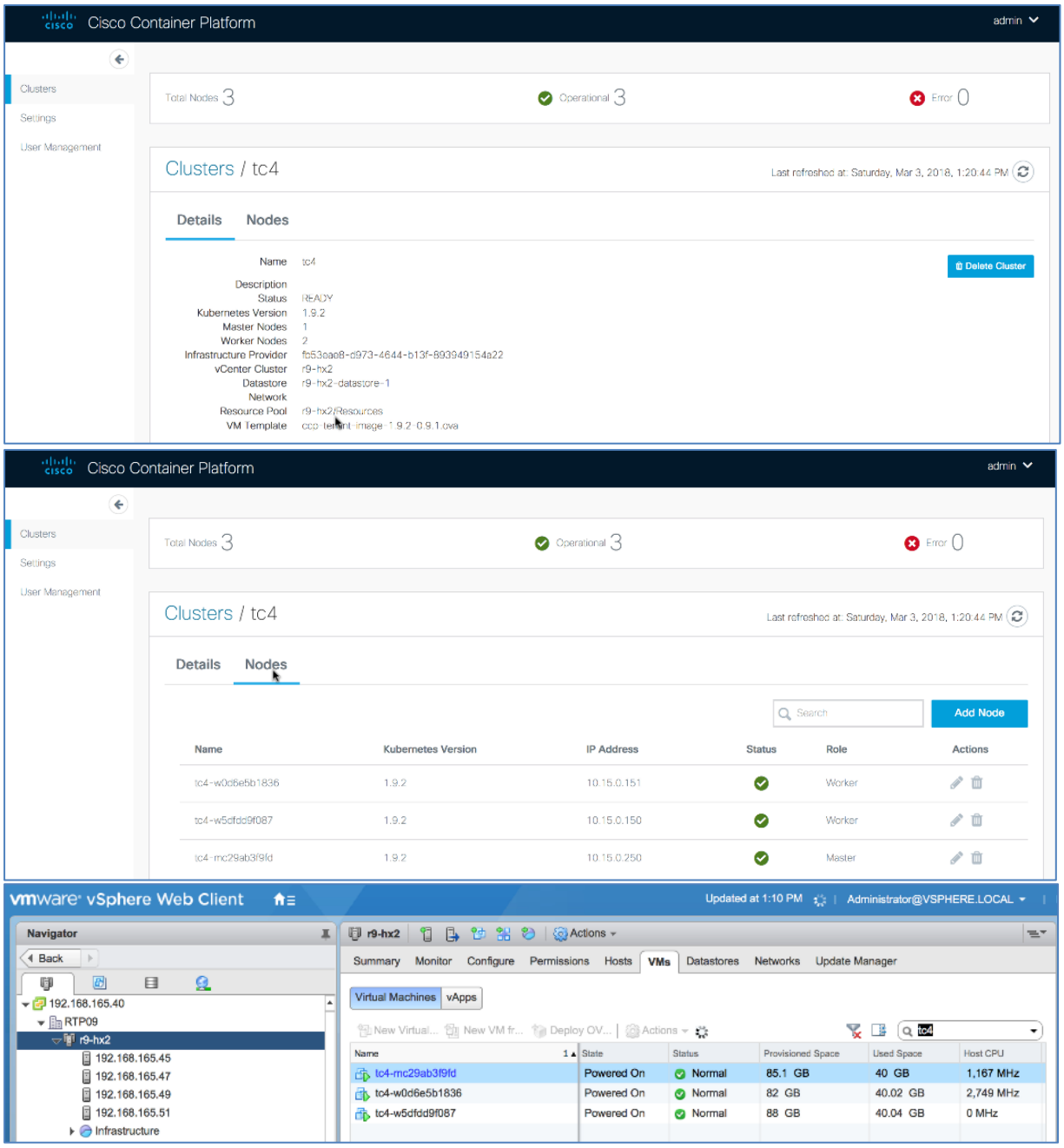
    kubectl get nodes -o wide

    Example

    ```
    kubectl get nodes -o wide
    ```

NAME	STATUS	ROLES	AGE	VERSION	EXTERNAL-IP	OS-IMAGE	KERNEL VERSION	CONTAINER RUNTIME
tc4-mc29ab3f9fd	Ready	master	3m	v1.9.2	10.15.0.250	Ubuntu 16.04.3 LTS	4.4.0-104-generic	Docker://1.13.1
tc4-w0d6e5b1836	Ready	<none>	2m	v1.9.2	10.15.0.151	Ubuntu 16.04.3 LTS	4.4.0-104-generic	Docker://1.13.1
Tc4-w5dfdd9f087	Ready	<none>	2m	v1.9.2	10.15.0.150	Ubuntu 16.04.3 LTS	4.4.0-104-generic	Docker://1.13.1





4.2 Deleting vSphere Tenant Clusters

Before you Begin

Ensure that curl and jq are installed on your client machine.

Procedure

1. Export Cisco Container Platform Virtual IP to the MGMT_HOST environment variable.

   Command

   export MGMT_HOST=<Control Plane VIP>

   Example

   ```
   export MGMT_HOST=10.20.30.40
   ```

2. Obtain a cookie using the username and password for your Cisco Container Platform instance.

   Command
   curl -k -c cookie.txt -H "Content-Type:application/x-www-form-urlencoded" -d 'username=admin&password=<Password from the installer>' https://$MGMT_HOST/2/system/login/

   Example
   ```
   curl -k -c cookie.txt -H "Content-Type:application/x-www-form-urlencoded" -d 'username=admin&password=<Password from the installer>' https://$MGMT_HOST/2/system/login/
   ```

3. List tenant clusters.

   Command
   curl -sk -b cookie.txt https://$MGMT_HOST/2/clusters| jq -r '.[].name, .uuid'

   Example
   ```
   curl -sk -b cookie.txt https://$MGMT_HOST/2/clusters| jq -r '.[].name, .uuid'
   tc1
   aef65a35-c013-4d91-9edb-e2ef8359f95b
   tc2
   8dab31ef-3efa-4de6-9e0d-07e6ff68bc24
   tc3
   a523fce7-b71e-444a-9626-871e17fe1fcd
   tc4
   8ccaa3a1-8a11-4996-9224-5723b7ecfdfd
   ```

4. Export the tenant cluster.

   Command
   export TC=<selected cluster from list>

   Example
   ```
   export TC=8ccaa3a1-8a11-4996-9224-5723b7ecfdfd
   ```

5. Delete the tenant cluster.

   Command
   curl -sk -b cookie.txt -X DELETE https://$MGMT_HOST/2/clusters/${TC}

   Example
   ```
   curl -sk -b cookie.txt -X DELETE https://$MGMT_HOST/2/clusters/${TC}
   ```

4.3 Configuring Windows AD Service Account for Authentication

Before you Begin

Ensure that curl and jq are installed on your client machine.

Procedure

1. Export Cisco Container Platform Virtual IP to the MGMT_HOST environment variable.

   Command
   export MGMT_HOST=<Control Plane VIP>

Example

```
export MGMT_HOST=10.20.30.40
```

2. Obtain a cookie using the username and password for your Cisco Container Platform instance.

   Command

   curl -k -c cookie.txt -H "Content-Type:application/x-www-form-urlencoded" -d 'username=admin&password=<Password from the installer>' https://$MGMT_HOST/2/system/login/

   Example

```
curl -k -c cookie.txt -H "Content-Type:application/x-www-form-urlencoded" -d 'username=admin&password=<Password from the installer>' https://$MGMT_HOST/2/system/login/
```

3. Query Windows AD server to verify the Service Account connection and members of the Cisco Container Platform accounts.

   Command

   ldapsearch -x -h <AD Server> -D "<Bind Distinguished Name>" -w '<Password>' -b "<Base Distinguished Name>" -s "<Scope>"

   Example

```
ldapsearch -x -h 192.0.2.1 -D "CN=Adam A. Arkanis,CN=Users,DC=r9-hx,DC=local" -w 'Password' -b "dc=r9-hx,dc=local" -s sub "(cn=CCP*)" member cn

# extended LDIF
#
# LDAPv3
# base <dc=r9-hx,dc=local> with scope subtree
# filter: (cn=CCP*)
# requesting: member cn
#

# CCPAdmins, Users, r9-hx.local
dn: CN=CCPAdmins,CN=Users,DC=r9-hx,DC=local
cn:  CCPAdmins
member: CN=Andrew A.  Andres,CN=Users,DC=r9-hx,DC=local
member: CN=Adam A.  Arkanis,CN=Users,DC=r9-hx,DC=local

# CCPDevOps, Users, r9-hx.local
dn: CN=CCPDevOps,CN=Users,DC=r9-hx,DC=local
cn: CCPDevOps
member: CN=Bob B. Bondurant,CN=Users,DC=r9-hx,DC=local
member: CN=Becky B. Bartholemew,CN=Users,DC=r9-hx,DC=local
```

4. Create json payload file for creating AD service account in Cisco Container Platform.

   Command

```
cat << EOF > ldap_serviceaccount.json
{
  "Server": " <AD Server>",
  "Port": 3268,
  "ServiceAccountDN": "<Bind Distinguished Name>",
  "ServiceAccountPassword": "<Password>",
```

```
  "StartTLS": false,
  "InsecureSkipVerify": true
}
EOF
```

Example

```
cat << EOF > ldap_serviceaccount.json
{
  "Server": ” 192.0.2.1”,
  "Port": 3268,
  "ServiceAccountDN": "CN=Adam A. Arkanis,CN=Users,DC=r9-
hx,DC=local",
  "ServiceAccountPassword": "Password",
  "StartTLS": false,
  "InsecureSkipVerify": true
}
EOF
```

5. Create the service account for Cisco Container Platform.

   Command

   curl -sk -b cookie.txt -X PUT -H "Content-Type: application/json" -d @ldap_serviceaccount.json https://$MGMT_HOST/2/ldap/setup

   Example

```
curl -sk -b cookie.txt -X PUT -H "Content-Type:
application/json" -d @ldap_serviceaccount.json
https://$MGMT_HOST/2/ldap/setup
{
    “Server”: ” 192.0.2.1”,
    “Port”: 3268,
    “BaseDN”: “DC=r9-hx,DC=local”,
    “ServiceAccountDN”: “CN=Adam A. Arkanis,CN=Users,DC=r9-
hx,DC=local”,
    “ServiceAccountPassword”: “”,
    “StartTLS”: false,
    “InsecureSkipVerify”: true
}
```

6. Confirm service account configuration.

   Command

   curl -k -b cookie.txt https://$MGMT_HOST/2/ldap/setup

   Example

```
curl -k -b cookie.txt https://$MGMT_HOST/2/ldap/setup
{
     “Server”: ” 192.0.2.1”,
     “Port”: 3268,
     “BaseDN”: “DC=r9-hx,DC=local”,
     “ServiceAccountDN”: “CN=Adam A. Arkanis,CN=Users,DC=r9-
hx,DC=local”,
     “ServiceAccountPassword”:  “”,
     “StartTLS”:  false,
     “InsecureSkipVerify”:  true
}
```

4.4 Managing Windows AD Group Authorizations for Tenant Clusters

Before you Begin

Ensure that curl and jq are installed on your client machine.

Procedure

1. Export Cisco Container Platform Virtual IP to the MGMT_HOST environment variable.

   Command
   export MGMT_HOST=<Control Plane VIP>

   Example
   ```
   export MGMT_HOST=10.20.30.40
   ```

2. Obtain a cookie using the username and password for your Cisco Container Platform instance.

   Command
   curl -k -c cookie.txt -H "Content-Type:application/x-www-form-urlencoded" -d 'username=admin&password=<Password from the installer>' https://$MGMT_HOST/2/system/login/

   Example
   ```
   curl -k -c cookie.txt -H "Content-Type:application/x-www-form-urlencoded" -d 'username=admin&password=<Password from the installer>' https://$MGMT_HOST/2/system/login/
   ```

3. Create json payload file for assigning an AD group to a SysAdmin or DevOps role.

   ```
   cat << EOF > ldap_devops_group.json
   {
     "LdapDN": "CN=CCPDevOps,CN=Users,DC=r9-hx,DC=local",
     "Role": "DevOps"
   }
   EOF
   ```

4. Create an LDAP group.
   An error message is displayed, if an LDAP group already exists and can continue with script.

   Command
   curl -sk -b cookie.txt -X POST -H "Content-Type: application/json" -d @ldap_devops_group.json https://$MGMT_HOST/2/ldap/groups

   Example
   ```
   curl -sk -b cookie.txt -X POST -H "Content-Type: application/json" -d @ldap_devops_group.json https://$MGMT_HOST/2/ldap/groups
   ```

   ```
   {
     "LdapDN": "CN=CCPDevOps,CN=Users,DC=r9-hx,DC=local",
     "Role": "DevOps"
   }
   ```

5. Get list of configured AD groups in Cisco Container Platform.

   Command
   curl -sk -b cookie.txt https://$MGMT_HOST/2/ldap/groups

Example

```
curl -sk -b cookie.txt https://$MGMT_HOST/2/ldap/groups

[
 {
  "LdapDN": "CN=CCPDevOps,CN=Users,DC=r9-hx,DC=local",
  "Role": "DevOps"
 }
]

#Return list of clusters to assign AD group to
```

6. Get list of clusters for which you want to assign an AD group.

   Command

   curl -sk -b cookie.txt https://$MGMT_HOST/2/clusters| jq -r '.[]|.name, .uuid'

   Example

```
curl -sk -b cookie.txt https://$MGMT_HOST/2/clusters| jq -r
'.[]|.name, .uuid'
tc1
aef65a35-c013-4d91-9edb-e2ef8359f95b
tc2
8dab31ef-3efa-4de6-9e0d-07e6ff68bc24
tc3
a523fce7-b71e-444a-9626-871e17fe1fcd
tc4
8ccaa3a1-8a11-4996-9224-5723b7ecfdfd
```

7. Export the selected tenant cluster.

   Command

   export TC=<Selected tenant cluster>

   Example

```
export TC=8ccaa3a1-8a11-4996-9224-5723b7ecfdfd
```

8. Create a json payload for assigning AD group to a tenant cluster.

```
cat << EOF > ldap_authz.json
{
  "name": "CN=CCPDevOps,CN=Users,DC=r9-hx,DC=local",
  "local": false
 }
EOF
```

9. Authorize group access to the selected tenant cluster.

   Command

   curl -sk -b cookie.txt -X POST -H "Content-Type: application/json" -d @ldap_authz.json https://$MGMT_HOST/2/clusters/${TC}/authz

   Example

```
curl -sk -b cookie.txt -X POST -H "Content-Type:
application/json" -d @ldap_authz.json
https://$MGMT_HOST/2/clusters/${TC}/authz

{
  "AuthID": "743e54da-037e-4386-99a7-a3da36e51936",
  "Name": "CN=CCPDevOps,CN=Users,DC=r9-hx,DC=local",
```

```
  "Local": false
}
```

10. Verify authorization of AD group to the tenant cluster.

    Command
    curl -sk -b cookie.txt https://$MGMT_HOST/2/clusters/${TC}/authz

    Example
    ```
    curl -sk -b cookie.txt
    https://$MGMT_HOST/2/clusters/${TC}/authz
    ```

    ```
    {
      "AuthList":[
        {
         "AuthID": "743e54da-037e-4386-99a7-a3da36e51936",
         "Name": "CN=CCPDevOps,CN=Users,DC=r9-hx,DC=local",
         "Local": false
        }
       ]
    }
    ```

11. Authenticate as a user from an AD DevOps group.

    Command
    curl -sk -c cookie_user.txt -H "Content-Type:application/x-www-form-urlencoded" -d "username=<AD User>&password=<Password>" https://$MGMT_HOST/2/system/login/

    Example
    ```
    curl -sk -c cookie_user.txt -H "Content-Type:application/x-www-form-urlencoded" -d "username=BobBB&password=Password"
    https://$MGMT_HOST/2/system/login/
    ```

12. Verify tenant cluster access list for an AD user.

    Command
    curl -sk -b cookie_user.txt https://$MGMT_HOST/2/clusters| jq -r '.[]|.name, .uuid'

    Example
    ```
    curl -sk -b cookie_user.txt https://$MGMT_HOST/2/clusters| jq -r '.[]|.name, .uuid'
    ```

    ```
    tc4
    ```

    ```
    8ccaa3a1-8a11-4996-9224-5723b7ecfdfd
    ```

13. Export the selected tenant cluster.

    Command
    export TC=<Selected tenant cluster>

    Example
    ```
    export TC=8ccaa3a1-8a11-4996-9224-5723b7ecfdfd
    ```

14. Download the KUBECONFIG environment file.

    Command
    curl -sk -b cookie.txt https://$MGMT_HOST/2/clusters/${TC}/env -o ${TC}.env

Example

```
curl -sk -b cookie.txt https://$MGMT_HOST/2/clusters/${TC}/env
-o ${TC}.env
```

15. Export the config file to KUBECONFIG environment variable.

    Command

    export KUBECONFIG=./${TC}.env

    Example

    ```
    export KUBECONFIG=./${TC}.env
    ```

16. View nodes on the tenant cluster.

    Command

    kubectl get nodes -o wide

    Example

    ```
    kubectl get nodes -o wide
    ```

    ```
    NAME              STATUS   ROLES   AGE   VERSION   EXTERNAL-IP  OS-IMAGE           KERNEL VERSION
    CONTAINER-RUNTIME

    tc4-mc29ab3f9fd   Ready    master  1h    v1.9.2    10.20.30.250  Ubuntu 16.04.3 LTS  4.4.0-104-generic
    docker://1.13.1

    tc4-w0d6e5b1836   Ready    <none>  1h    v1.9.2    10.20.30.151  Ubuntu 16.04.3 LTS  4.4.0-104-generic
    docker://1.13.1

    tc4-w5dfdd9f087   Ready    <none>  1h    v1.9.2    10.20.30.150  Ubuntu 16.04.3 LTS  4.4.0-104-generic
    docker://1.13.1
    ```

17. Remove AD group access.

    Command

    #curl -sk -b cookie.txt -X DELETE https://$MGMT_HOST/2/ldap/groups/<DN of Group>

    Example

    ```
    curl -sk -b cookie.txt -X DELETE
    https://$MGMT_HOST/2/ldap/groups/CN=CCPDevOps,CN=Users,DC=r9-
    hx,DC=local
    ```

18. Verify that authorization of AD group to tenant cluster is removed.

    Command

    curl -sk -b cookie.txt https://$MGMT_HOST/2/clusters/${TC}/authz

    Example

    ```
    curl -sk -b cookie.txt
    https://$MGMT_HOST/2/clusters/${TC}/authz
    ```

    ```
    {
      "AuthList": []
    }
    ```

4.5 Downloading Tenant Cluster KUBECONFIG Environment File

Before you Begin

Ensure that curl and jq are installed on your client machine.

Procedure

1. Export Cisco Container Platform Virtual IP to the MGMT_HOST environment variable.

Command
export MGMT_HOST=<Control Plane VIP>

Example
```
export MGMT_HOST=10.20.30.40
```

2. Obtain a cookie using the username and password for your Cisco Container Platform instance.

   Command
   curl -k -c cookie.txt -H "Content-Type:application/x-www-form-urlencoded" -d 'username=admin&password=<Password from the installer>' https://$MGMT_HOST/2/system/login/

   Example
   ```
   curl -k -c cookie.txt -H "Content-Type:application/x-www-form-urlencoded" -d 'username=admin&password=<Password from the installer>' https://$MGMT_HOST/2/system/login/
   ```

3. List tenant clusters.

   Command
   curl -sk -b cookie.txt https://$MGMT_HOST/2/clusters| jq -r '.[]|.name, .uuid

   Example
   ```
   curl -sk -b cookie.txt https://$MGMT_HOST/2/clusters| jq -r '.[]|.name, .uuid'
   tc1
   aef65a35-c013-4d91-9edb-e2ef8359f9gb
   tc2
   8dab31ef-3efa-4de6-9e0d-07e6ff68bc24
   tc3
   a523fce7-b71e-444a-9626-871e17fe1fcd
   tc4
   8ccaa3a1-8a11-4996-9224-5723b7ecfdfd
   ```

4. Export a tenant cluster.

   Command
   export TC=<selected cluster from list>

   Example
   ```
   export TC=8ccaa3a1-8a11-4996-9224-5723b7ecfdfd
   ```

5. Download the KUBECONFIG environmental file.

   Command
   curl -sk -b cookie.txt https://$MGMT_HOST/2/clusters/${TC}/env -o ${TC}.env

   Example
   ```
   curl -sk -b cookie.txt https://$MGMT_HOST/2/clusters/${TC}/env -o ${TC}.env
   ```

6. Export the config file to KUBECONFIG environment variable.

   Command
   export KUBECONFIG=./${TC}.env

   Example
   ```
   export KUBECONFIG=./${TC}.env
   ```

7. View nodes on the tenant cluster.
   Command
   kubectl get nodes -o wide

   Example
   ```
   kubectl get nodes -o wide
   ```

```
NAME             STATUS   ROLES    AGE   VERSION   EXTERNAL-IP   OS-IMAGE             KERNEL VERSION     CONTAINER-RUNTIME
tc4-mc29ab3f9fd   Ready    master   1h    v1.9.2    10.20.30.250  Ubuntu 16.04.3 LTS   4.4.0-104-generic  docker://1.13.1
tc4-w0d6e5b1836   Ready    <none>   1h    v1.9.2    10.20.30.151  Ubuntu 16.04.3 LTS   4.4.0-104-generic  docker://1.13.1
tc4-w5dfdd9f087   Ready    <none    1h    v1.9.2    10.20.30.150  Ubuntu 16.04.3 LTS   4.4.0-104-generic  docker://1.13.1
```

4.6 Obtaining TC Master and Ingress VIPs

FOR MASTER

```
`curl -sk -X GET -b temp/cookie.txt
https://$MGMT_HOST/2/clusters/<clustername> | jq  '.master_vip`
```

FOR INGRESS VIPS

```
`curl -sk -X GET -b temp/cookie.txt
https://$MGMT_HOST/2/clusters/<cluster> | jq  '.ingress_vips`
```

5 Examples of API Use Cases for AWS EKS Clusters

5.1 Logging in to Cisco Container Platform

Command
curl -c cookies.txt -k -X POST -d "username=admin&password=<your_password>" -H "Content-Type:application/x-www-form-urlencoded" "https://<ccp_url>/2/system/login"

Example

1. Log in to Cisco Container Platform.

   ```
   curl -c cookies.txt -k -X POST -d
   "username=admin&password=my_password" -H "Content-
   Type:application/x-www-form-urlencoded"
   "https://10.20.30.40/2/system/login"
   ```

2. Retrieve the token from the cookies.txt file created as a result of the above command and then store it in an environment variable like this:

   ```
   $ cat cookies.txt
   # Netscape HTTP Cookie File
   # https://curl.haxx.se/docs/http-cookies.html
   # This file was generated by libcurl! Edit at your own
   risk.

   10.20.30.40    FALSE     /     FALSE     0     CXAccessToken
        eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJBTExfQ0xVU1RFUl
   NfQVVUSCI6dHJ1ZSwiZXhwIjoxNTQ4NjM5MDMyLCJyb2xlIjoiQWRtaW5pc
   3RyYXRvciJ9.ypjTZFKKmfuBvRxodu-MLedIkQROVNqHdqXgKKdAv7M
   ```

3. Set your env variable using the token value obtained from Step 2.

```
export
TOKEN=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJBTExfQ0xVU1RF
UlNfQVVUSCI6dHJ1ZSwiZXhwIjoxNTQ4NjM5MDMyLCJyb2xlIjoiQWRtaW5
pc3RyYXRvciJ9.ypjTZFKKmfuBvRxodu-MLedIkQROVNqHdqXgKKdAv7M
```

5.2 Creating Providers for EKS

Command

```
curl -k -X POST -H "x-auth-token: $TOKEN" -d \
'{
        "type": "eks",
        "name":"name_of_your_eks_cluster",
        "role_arn": "you_aws_role_arn",
        "access_key_id": "your_AWS_access_key_id",
        "secret_access_key": "your_AWS_secret_access_key"
}' https:// <ccp-url>/v3/providers/
```

Example

```
curl -k -X POST -H "x-auth-token: $TOKEN" -d \
'{
            "type": "eks",
            "name":"selvi-eks-provider",
            "role_arn":
"arn:aws:iam::123456789123:role/eksServiceRole",
            "access_key_id": "ABCDEFGHIJKLMNOPQRST",
            "secret_access_key":
"THISISNOTAREALSECRETKEYBUTLOOKSLIKEONE"
}' https://10.20.30.40/v3/providers/
```

5.3 Retrieving List of Providers for EKS

Command

```
curl -k -X GET -H "X-Auth-Token":"$TOKEN" https://<ccp-url>/v3/providers
```

Example

```
curl -k -X GET -H "X-Auth-Token":"$TOKEN"
https://10.20.30.40/v3/providers
```

5.4 Retrieving Specific Provider for EKS

Command

```
curl -k -X GET -H "X-Auth-Token":"$TOKEN" https:// <ccp-
url>/v3/providers/<provider_uuid>/
```

Example

```
curl -k -X GET -H "X-Auth-Token":"$TOKEN" https://
10.20.30.40/v3/providers/17d7d949-cf95-4676-80a7-ae3d773dc3b0/

[
    {
        "access_key_id": "ABCDEFGHIJKLMNOPQRST",
        "id": "7edd7790-a776-4a91-91f3-0938483dbf78",
        "name": "selvi-eks-provider",
        "role_arn": "arn:aws:iam::12345678912:role/ccp-eks-
7edd7790-a776-4a91-91f3-0938483dbf78",
        "type": "eks"
```

```
        }
]
```

5.5 Modifying Providers for EKS

You cannot update the provider details once it is created. This includes parameters such as the Role_ARN, Type, Access_Key_ID, and Secret_Access_Key.

5.6 Deleting Providers for EKS

Command

curl -k -X DELETE -H "x-auth-token: $TOKEN" https:// <ccp-url>/v3/providers/<provider_uuid>/

Example

```
curl -k -X DELETE -H "x-auth-token: $TOKEN"
https://10.20.30.40/v3/providers/7edd7790-a776-4a91-91f3-
0938483dbf78/
```

5.7 Creating EKS clusters

Command

curl -k -X POST -H "x-auth-token: $TOKEN" -d \
'{
"provider":"provider_uuid",
"vpc_sizing": {
"subnet":"<your_desired_subnet>",
"public_subnets": ["<desired_pub_subnet1>", "<desired_pub_subnet2>", "
<desired_pub_subnet3>"],
"private_subnets": ["<desired_priv_subnet1>", "<desired_priv_subnet2>",
"<desired_priv_subnet3>"]
},
"region":"<aws_region_string>",
"type":"eks",
"ami": "<ami_id>",
"instance_type": "<amazon_instance_type>",
"worker_count": <number_of_workers_in_eks_cluster>,
"access_role_arn":"<arn_of_role_in_your_aws_account>",
"name": "<name_of_your_eks_cluster>",
"ssh_keys": ["<your_ssh_key_to_be_able_to_access_your_workers>",
"<optionally_another_ssh_key>"]
}' https://<ccp_url>/v3/clusters/

Example

```
curl -k -X POST -H "x-auth-token: $TOKEN" -d \
'{
    "provider":"17d7d949-cf95-4676-80a7-ae3d773dc3b0",
    "vpc_sizing": {
      "subnet":"10.20.0.0/16",
      "public_subnets": ["10.20.1.0/24", "10.20.2.0/24",
"10.20.3.0/24"],
      "private_subnets": ["10.20.4.0/24", "10.20.5.0/24",
"10.20.6.0/24"]
    },
```

```
    "region":"us-west-2",
    "type":"eks",
    "ami": "ami-09677889326e51ea1",
    "instance_type": "t2.small",
    "worker_count": 1,

"access_role_arn":"arn:aws:iam::123456789123:role/KubernetesAdmin
",
    "name": "selvi_eks_1",
    "ssh_keys": ["ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIHdSrKkWhwED6awk9sjegF0dgcKnotmyrealkey
selvik@SELVIK-M-C1DM", "another_dummy"]
}' https://10.20.30.40/v3/clusters/
```

```
{
   "id":"094c1544-58e5-46cf-8a3f-94de81f35574",
   "type":"eks",
   "name":" selvi_eks_1",
   "provider":"17d7d949-cf95-4676-80a7-ae3d773dc3b0",
   "region":"us-west-2",
   "status":"CREATING",
   "status_detail":null,
   "access_role_arn":"arn:aws:iam::123456789123:role/KubernetesAd
min",
   "kubeconfig":null,
   "vpc_sizing":{
      "subnet":"10.20.0.0/16",
      "public_subnets":[
         "10.20.1.0/24",
         "10.20.2.0/24",
         "10.20.3.0/24"
      ],
      "private_subnets":[
         "10.20.4.0/24",
         "10.20.5.0/24",
         "10.20.6.0/24"
      ]
   },
   "ami":"ami-09677889326e51ea1",
   "instance_type":"t2.small",
   "ssh_key_name":"",
   "worker_count":1,
   "vpc_id":null
}
```

Note: The API returns the values immediately and the status is indicated as *CREATING*.

5.8 Retrieving all EKS clusters

Command
curl -k -X GET -H "X-Auth-Token":"$TOKEN" https://<ccp_url>/v3/clusters

Example

```
curl -k -X GET -H "X-Auth-Token":"$TOKEN"
https://10.10.99.190/v3/clusters
```

```
 [
   {
      "id":"094c1544-58e5-46cf-8a3f-94de81f35574",
```

```
      "type":"eks",
      "name":"selvi_eks_1",
      "provider":"17d7d949-cf95-4676-80a7-ae3d773dc3b0",
      "region":"us-west-2",
      "status":"CREATING_MASTER",
      "status_detail":"",
      "access_role_arn":"arn:aws:iam::123456789123:role/Kubernete
sAdmin",
      "kubeconfig":null,
      "vpc_sizing":{
         "subnet":"10.20.0.0/16",
         "public_subnets":[
            "10.20.1.0/24",
            "10.20.2.0/24",
            "10.20.3.0/24"
         ],
         "private_subnets":[
            "10.20.4.0/24",
            "10.20.5.0/24",
            "10.20.6.0/24"
         ]
      },
      "ami":"ami-09677889326e51ea1",
      "instance_type":"t2.small",
      "ssh_key_name":"",
      "worker_count":1,
      "vpc_id":"vpc-thisis72e6cnotreal"
   }
]
```


5.9 Retrieving Specific EKS Clusters

Command

curl -k -X GET -H "X-Auth-Token":"$TOKEN"
https://<ccp_url>/v3/clusters/<your_cluster_uuid>/

Example

```
curl -k -X GET -H "X-Auth-Token":"$TOKEN"
https://10.10.99.190/v3/clusters/5a5f0db5-110c-4151-80e8-
9b78889d30bc/
```


```
 [
   {
      "id":"094c1544-58e5-46cf-8a3f-94de81f35574",
      "type":"eks",
      "name":"selvi_eks_1",
      "provider":"17d7d949-cf95-4676-80a7-ae3d773dc3b0",
      "region":"us-west-2",
      "status":"CREATING_MASTER",
      "status_detail":"",
      "access_role_arn":"arn:aws:iam::123456789123:role/Kubernete
sAdmin",
      "kubeconfig":null,
      "vpc_sizing":{
         "subnet":"10.20.0.0/16",
         "public_subnets":[
            "10.20.1.0/24",
            "10.20.2.0/24",
            "10.20.3.0/24"
```

```
            ],
            "private_subnets":[
               "10.20.4.0/24",
               "10.20.5.0/24",
               "10.20.6.0/24"
            ]
         },
         "ami":"ami-09677889326e51ea1",
         "instance_type":"t2.small",
         "ssh_key_name":"",
         "worker_count":1,
         "vpc_id":"vpc-thisis72e6cnotreal"
      }
]
```


5.10 Modifying EKS clusters

Command

```
curl -k -X PATCH -H "x-auth-token: $TOKEN" -d \
'{
   "worker_count": 2
}' https://<ccp_url>/v3/clusters/<cluster_uuid>/
```

Example

```
curl -k -X PATCH -H "x-auth-token: $TOKEN" -d \
'{
    "worker_count": 2
}' https://10.20.99.190/v3/clusters/5a5f0db5-110c-4151-80e8-
9b78889d30bc/
```


```
[
   {
      "id":"094c1544-58e5-46cf-8a3f-94de81f35574",
      "type":"eks",
      "name":"selvi_eks_1",
      "provider":"17d7d949-cf95-4676-80a7-ae3d773dc3b0",
      "region":"us-west-2",
      "status":"CREATING_MASTER",
      "status_detail":"",
      "access_role_arn":"arn:aws:iam::123456789123:role/Kubernete
sAdmin",
      "kubeconfig":null,
      "vpc_sizing":{
         "subnet":"10.20.0.0/16",
         "public_subnets":[
            "10.20.1.0/24",
            "10.20.2.0/24",
            "10.20.3.0/24"
         ],
         "private_subnets":[
            "10.20.4.0/24",
            "10.20.5.0/24",
            "10.20.6.0/24"
         ]
      },
      "ami":"ami-09677889326e51ea1",
      "instance_type":"t2.small",
      "ssh_key_name":"",
      "worker_count":1,
```

```
        "vpc_id":"vpc-thisis72e6cnotreal"
    }
]
```


5.11 Deleting EKS clusters

Command

curl -k -X DELETE -H "x-auth-token: $TOKEN"
https://<ccp_url>/v3/clusters/cluster_uuid/

Example

```
curl -k -X DELETE -H "x-auth-token: $TOKEN"
https://10.10.99.190/v3/clusters/5a5f0db5-110c-4151-80e8-
9b78889d30bc/
```

6 Cisco Container Platform API Reference

swagger-api.json Explore

Cisco Container Platform Control Plane API Documentation 4.0.0

[Base URL: https://Cisco Container Platform Control Plane IP/2/]
swagger-api.json

Schemes
HTTP

/v3 CCP v3 API

DELETE `/v3/{resource}` forwards v3 API requests to the v3 API service

GET `/v3/{resource}` forwards v3 API requests to the v3 API service

HEAD `/v3/{resource}` forwards v3 API requests to the v3 API service

PATCH `/v3/{resource}` forwards v3 API requests to the v3 API service

POST `/v3/{resource}` forwards v3 API requests to the v3 API service

PUT `/v3/{resource}` forwards v3 API requests to the v3 API service

2/aci_api accessing ACI api

POST `/2/aci_api/login` ACI login

2/aci_profiles List of ACI profile endpoints

GET `/2/aci_profiles` Get all ACI profiles

POST `/2/aci_profiles` Create an ACI profile with the given configuration

GET `/2/aci_profiles/{aciProfileName}` Get an ACI profile by name

DELETE `/2/aci_profiles/{aciProfileUUID}` Delete an ACI profile

PATCH `/2/aci_profiles/{aciProfileUUID}` Update an ACI profile

2/clusters List of cluster endpoints

GET `/2/clusters` Get all clusters

POST `/2/clusters` Create a cluster with the given specification

GET `/2/clusters/{clusterID}/authz` List authorizations for a cluster

POST `/2/clusters/{clusterID}/authz` Add authorization for a cluster

DELETE `/2/clusters/{clusterID}/authz/{authID}` Delete authorization for a cluster

GET `/2/clusters/{clusterName}` Get a cluster by name

DELETE `/2/clusters/{clusterUUID}` Delete a cluster

PATCH `/2/clusters/{clusterUUID}` Patch a cluster

PUT `/2/clusters/{clusterUUID}` Update a cluster

GET `/2/clusters/{clusterUUID}/dashboard` Get dashboard

GET `/2/clusters/{clusterUUID}/env` Get cluster environment

GET `/2/clusters/{clusterUUID}/helmcharts` Get HelmCharts object for a given cluster

POST `/2/clusters/{clusterUUID}/helmcharts` Create a helmChart for cluster with the given specification

DELETE `/2/clusters/{clusterUUID}/helmcharts/{HelmChartUUID}` Delete helm chart for cluster

POST `/2/clusters/{clusterUUID}/nodepools` Create a node pool for a cluster

DELETE `/2/clusters/{clusterUUID}/nodepools/{nodePoolID}` Delete a node pool from a cluster

PATCH `/2/clusters/{clusterUUID}/nodepools/{nodePoolID}` Update a node pool in a cluster

PATCH `/2/clusters/{clusterUUID}/upgrade` Upgrade a cluster

2/keyvalues List of endpoints for key values

GET `/2/keyvalues/{key}`

POST `/2/keyvalues/{key}`

2/ldap List of ldap endpoints

GET `/2/ldap/groups` Get CX LDAP Groups

POST `/2/ldap/groups` Create CX LDAP Group

PUT `/2/ldap/groups` Update a CX LDAP Group.

GET `/2/ldap/groups/authz` Get CX the cluster authorizations for a CX LDAP group

DELETE `/2/ldap/groups/{ldapDN}` Delete CX LDAP Group specified by LDAP DN

GET `/2/ldap/setup` Get LDAP parameters

PUT `/2/ldap/setup` Setup/update LDAP parameters

2/license List of licensing endpoints

DELETE `/2/license/{resource}` Refer to the smart licensing documentation

GET `/2/license/{resource}` Refer to the smart licensing documentation

DELETE `/2/license/{resource}/{agentID}` Refer to the smart licensing documentation

GET `/2/license/{resource}/{agentID}` Refer to the smart licensing documentation

POST `/2/license/{resource}/{agentID}` Refer to the smart licensing documentation

2/localusers

GET `/2/localusers` Get CX local users

POST `/2/localusers` Create CX local user

DELETE `/2/localusers/{username}` Delete a local user

PATCH `/2/localusers/{username}` Update a local user. Can provide either or both parameters.

PATCH `/2/localusers/{username}/password` Update

2/providerclientconfigs List of provider client config endpoints

GET	`/2/providerclientconfigs`	Get provider client configuration list
POST	`/2/providerclientconfigs`	Add provider client configuration
DELETE	`/2/providerclientconfigs/{clientconfigUUID}`	Delete provider client configuration
GET	`/2/providerclientconfigs/{clientconfigUUID}`	Get provider client configuration
PATCH	`/2/providerclientconfigs/{clientconfigUUID}`	Update provider client configuration
GET	`/2/providerclientconfigs/{clientconfigUUID}/clusters`	Get list of clusters who are using providerclientconfig
GET	`/2/providerclientconfigs/{clientconfigUUID}/vsphere/datacenter`	Gets the list of vSphere Data Centers.
GET	`/2/providerclientconfigs/{clientconfigUUID}/vsphere/datacenter/{datacenterName}/cluster`	Gets the list of vSphere Clusters in a datacenter.
GET	`/2/providerclientconfigs/{clientconfigUUID}/vsphere/datacenter/{datacenterName}/cluster/{clusterName}/gpu`	Gets the list of VSphere GPUs.
GET	`/2/providerclientconfigs/{clientconfigUUID}/vsphere/datacenter/{datacenterName}/cluster/{clusterName}/pool`	Gets the list of vSphere Pools.
GET	`/2/providerclientconfigs/{clientconfigUUID}/vsphere/datacenter/{datacenterName}/datastore`	Gets the list of vSphere Datastores.
GET	`/2/providerclientconfigs/{clientconfigUUID}/vsphere/datacenter/{datacenterName}/network`	Gets the list of vSphere Networks.
GET	`/2/providerclientconfigs/{clientconfigUUID}/vsphere/datacenter/{datacenterName}/vm`	Gets the list of vSphere Virtual Machines.

2/rbac

GET	`/2/rbac`	get the role of the current user

2/system List of system endpoints

GET	`/2/system/CorcHealth`	Get corc health

GET `/2/system/health` Returns the health of the system

GET `/2/system/livenessHealth` Returns a string representing the health of the system

POST `/2/system/login` Management server login

Models

api.ACILoginReply {

token*	string

}

api.ACILoginRequest {

apic_ips*	string
apic_password*	string
apic_username*	string

}

api.AddAuthorization {

Local*	boolean
Name*	string

}

api.AddAuthorizationReply {

AuthID*	string
Local*	boolean
Name*	string

}

api.CorcHealthReply {

}

api.CorcHealthRequest {

}

api.CreateLocalUserRequest {

```
    Disable*              boolean
    FirstName*            string
    LastName*             string
    Password*             string
    Role*                 string
    Token*                string
    UserName*             string
}
```

api.CreateLocalUserResponse {

```
}
```

api.CreateNodePoolReply {

```
    NodePool*
```

api.CreateNodePoolReply.NodePool {...}

```
}
```

api.CreateNodePoolReply.NodePool {

```
}
```

api.DeleteNodePoolReply {

```
}
```

api.GetVSphereClustersReply {

```
    Clusters*
                          [...]
}
```

api.GetVSphereDatacentersReply {

```
    Datacenters*
                          [...]
}
```

api.GetVSphereDatastoresReply {

```
    Datastores*
                          [...]
}
```

api.GetVSphereGpusReply {

```
    gpus*                       [...]
}
```

api.GetVSphereNetworksReply {

```
    Networks*                   [...]
}
```

api.GetVSpherePoolsReply {

```
    Pools*                      [...]
}
```

api.GetVSphereVMsReply {

```
    VMs*                        [...]
}
```

api.GpuHostIndex {

```
    gpu_type*           string
    hosts*                      [...]
}
```

api.HostGpuCount {

```
    count*              integer($int32)
    hostname*           string
}
```

api.LdapGroup {

```
    LdapDN*             string
    Role*               string
}
```

api.NodePoolRequest {

gpus*	[...]
labels*	string
memory*	integer($int64)
name*	string
node_ip_pool_uuid*	string
size*	integer($int32)
taints*	string
template*	string
vcpus*	integer($int32)

}

api.ResizeNodePoolRequest {

size*	integer($int32)

}

api.UpdateLocalUserPasswordRequest {

logged_in_user_password*	string
new_password*	string

}

api.UpdateLocalUserRequest {

Disable*	boolean
FirstName*	string
LastName*	string
Role*	string

}

ipam.IPInfo {

gateway*	string
id*	integer
ip*	string
mtu*	integer($int32)
nameservers*	[...]
netmask*	string
subnet	string
uuid*	string

}

ipam.LoadBalancerIPInfo {

IPInfo*	ipam.IPInfo {...}
never_release*	boolean

}

ipam.NodeIPInfo {

IPInfo*	ipam.IPInfo {...}
if_name*	string
type*	{...}

}

main.GetRoleResonse {

role*	string

}

types.ACIProfile {

aaep_name*	string
aci_allocator	types.ACIProfileAllocatorConfig {...}
aci_infra_vlan_id*	integer
aci_tenant*	string
aci_vmm_domain_name*	string
apic_hosts*	string
apic_password*	string
apic_username*	string
control_plane_contract_name*	string
l3_outside_network_name*	string
l3_outside_policy_name*	string
name*	string
nameservers*	[...]
uuid*	string
vrf_name*	string

}

types.ACIProfileAllocatorConfig {

multicast_range*	string
node_vlan_end*	integer
node_vlan_start*	integer
pod_subnet_start*	string
service_subnet_start*	string

}

types.Cluster {

Infra*	types.Cluster.Infra {...}
aci_profile_uuid*	string
aws_iam_enabled*	boolean
aws_iam_role_arn*	string
ccp_private_ssh_key*	string
ccp_public_ssh_key*	string
cluster_dashboard_url*	string
cluster_env_url*	string
deployer*	types.Kubeadm {...}
description*	string
etcd_encrypted*	boolean
harbor_admin_server_password*	string
harbor_registry_size*	string
helm_charts*	[...]
ingress_vip_pool_id*	string
ingress_vips*	[...]
is_adopt*	boolean
is_control_cluster*	boolean
is_harbor_enabled*	boolean
is_istio_enabled*	boolean
kubernetes_version*	string
labels*	[...]
load_balancer_ip_info_list*	[...]
load_balancer_ip_num*	integer($int32)
master_mac_addresses*	[...]
master_node_pool	types.Cluster.master_node_pool {...}
master_vip*	string
master_vip_addr_id*	string
masters*	integer($int32)
name*	string
network_plugin*	types.NetworkPluginProfile {...}
node_ip_pool_uuid	string
node_pools*	[...]
nodes*	[...]
ntp_pools*	[...]
ntp_servers*	[...]
provider_client_config_uuid*	string
registries_insecure*	[...]
registries_root_ca*	[...]
registries_self_signed*	[...]
secure_multitenancy_enabled*	boolean
ssh_key*	string
ssh_user*	string
state*	string
storage_class*	string
template*	string
tsig_key*	string
type*	{...}
uuid*	string
worker_node_pool	types.Cluster.worker_node_pool {...}
workers*	integer($int32)

}

types.Cluster.Infra {

}

types.Cluster.master_node_pool {

}

types.Cluster.node_pools {

}

types.Cluster.worker_node_pool {

}

types.GpuTypeCount {

```
count*                  integer($int32)
gpu_type*               string
```

}

types.HelmChart {

```
chart_url*              string
cluster_UUID*           string
helmchart_uuid*         string
name*                   string
options*                string
```

}

types.K8SNodeStatus {

```
LastTransitionTime*     string
NodeCondition*          string
NodeName*               string
NodeStatus*             string
```

}

types.K8SPodStatus {

```
LastTransitionTime*     string
PodCondition*           string
PodName*                string
PodStatus*              string
```

}

types.Kubeadm {

provider*	types.VsphereCloudProvider {...}
provider_type*	string

}

types.Label {

key*	string
value*	string

}

types.LdapSetup {

BaseDN*	string
InsecureSkipVerify*	boolean
Port*	integer
Server*	string
ServiceAccountDN*	string
ServiceAccountPassword*	string
StartTLS*	boolean

}

types.LoginStatus {

from_host*	string
last_fail*	string($date-time)
last_success*	string($date-time)
login_id*	string
proto*	string
status*	string
to_host*	string
total_fail*	integer($int32)

}

types.NetworkPluginProfile {

details*	string
name*	string
status*	string

}

types.Node {

cloud_init_data*	string
error_log*	string
ip_info*	[...]
is_master*	boolean
kubernetes_version*	string
mac_addresses*	[...]
name*	string
node_pool_id*	integer
node_pool_type*	string
private_ip*	string
public_ip*	string
state*	string
template*	string
uuid*	string

}

types.ProviderClientConfig {

config*	types.ProviderClientConfig.config {...}
name*	string
type*	{...}
uuid*	string

}

types.ProviderClientConfig.config {

}

types.SystemHealth {

CurrentNodes*	integer($int32)
ExpectedNodes*	integer($int32)
NodesStatus*	[...]
PodStatusList*	[...]
TotalSystemHealth*	string

}

types.VsphereClientConfig {

ip*	string
password	string
port*	integer
username*	string

}

types.VsphereCloudProvider {

client_config;omitempty*	**types.VsphereClientConfig** {...}
vsphere_client_config_uuid*	string
vsphere_datacenter*	string
vsphere_datastore*	string
vsphere_scsi_controller_type*	string
vsphere_working_dir*	string

}

ERROR {···}