Cisco Container Platform 8.0.0 API Guide

First Published: Mar 15, 2021
Abstract
The Cisco Container Platform 8.0.0 API Guide gives information on Cisco Container Platform APIs and development features.

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB’s public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS” WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: www.cisco.com/go/trademarks. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1110R)

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

Cisco Container Platform 8.0.0 API Guide
© 2021 Cisco Systems, Inc. All rights reserved.
1 Overview ........................................................................................................................................5
2 Accessing Cisco Container Platform API Documentation ...............................................................5
3 Key Concepts ...................................................................................................................................5
  3.1 Provider Client Configuration .......................................................................................................5
  3.2 Cluster ..........................................................................................................................................5
  3.3 User Management and Authorization ............................................................................................6
    3.3.1 LDAP and Local Users ........................................................................................................... 6
  3.4 Subnets and Virtual IP Address Pools ............................................................................................6
4 Logging in to Cisco Container Platform ............................................................................................6
5 Managing Users ...................................................................................................................................7
  5.1.1 Configuring Windows AD Service Account for Authentication ............................................... 7
  5.1.2 Managing Windows AD Group Authorizations for Tenant Clusters ......................................... 9
6 Managing v3 Clusters on vSphere ......................................................................................................14
  6.1 Managing v3 vSphere Provider .......................................................................................................14
    6.1.1 Creating Providers for vSphere ................................................................................................14
    6.1.2 Retrieving List of Providers ....................................................................................................15
    6.1.3 Retrieving Specific Provider ..................................................................................................15
    6.1.4 Modifying vSphere Provider ................................................................................................16
    6.1.5 Deleting vSphere Provider ....................................................................................................17
  6.2 Administering v3 Clusters on vSphere ............................................................................................17
    6.2.1 Creating vSphere Cluster ........................................................................................................17
    6.2.2 Retrieving all Clusters .............................................................................................................19
    6.2.3 Retrieving Specific Clusters ....................................................................................................23
    6.2.4 Modifying vSphere Clusters ....................................................................................................26
    6.2.5 Deleting vSphere Clusters .......................................................................................................26
    6.2.6 Listing Add-ons ......................................................................................................................27
    6.2.7 Listing Catalog of Add-ons .....................................................................................................28
    6.2.8 Configuring Addons ............................................................................................................... 29
    6.2.9 Adding Addons with Overrides ..............................................................................................29
    6.2.10 Deleting Addons ....................................................................................................................30
    6.2.11 Adding Node Pools ............................................................................................................... 31
    6.2.12 Getting List of Node Pools ....................................................................................................32
    6.2.13 Modifying Node Pools ..........................................................................................................33
    6.2.14 Deleting Node Pools ..............................................................................................................34
    6.2.15 Downloading Tenant Cluster KUBECONFIG Environment File ............................................34
  6.3 Using ACI CNI Network Plugin ......................................................................................................34
    6.3.1 Creating ACI Profile ...............................................................................................................34
    6.3.2 Creating ACI-enabled vSphere Cluster ....................................................................................36
    6.3.3 Updating ACI Profile ..............................................................................................................36
    6.3.4 Deleting ACI Profile ...............................................................................................................36
7 Managing v2 Clusters on vSphere .....................................................................................................36
  7.1 Administering v2 Clusters on vSphere ............................................................................................37
    7.1.1 Creating vSphere Tenant Clusters ..........................................................................................37
    7.1.2 Deleting vSphere Tenant Clusters ..........................................................................................44
    7.1.3 Downloading Tenant Cluster KUBECONFIG Environment File .............................................45
    7.1.4 Obtaining TC Master and Ingress VIPs ...................................................................................47
8 Managing v3 Clusters on EKS ...........................................................................................................48
8.1 Managing v3 EKS Provider
8.1.1 Creating Providers for EKS
8.1.2 Retrieving List of Providers for EKS
8.1.3 Retrieving Specific Provider for EKS
8.1.4 Modifying Providers for EKS
8.1.5 Deleting Providers for EKS

8.2 Administering v3 Clusters on EKS
8.2.1 Creating EKS clusters
8.2.2 Retrieving all clusters
8.2.3 Retrieving Specific EKS Clusters
8.2.4 Modifying EKS clusters
8.2.5 Deleting EKS clusters
8.2.6 Downloading Tenant Cluster KUBECONFIG Environment File

9 Managing Clusters on Openstack
9.1 Managing v3 Openstack Provider
9.1.1 Creating Providers for Openstack
9.1.2 Retrieving List of Providers for Openstack
9.1.3 Retrieving Specific Provider for Openstack
9.1.4 Modifying Providers for Openstack
9.1.5 Deleting Providers for Openstack

9.2 Administering v3 Clusters on Openstack
9.2.1 Creating Openstack clusters
9.2.2 Retrieving all clusters
9.2.3 Retrieving Specific Openstack Clusters
9.2.4 Modifying Openstack clusters
9.2.5 Deleting Openstack clusters

10 Managing Clusters on AKS
10.1 Managing v3 AKS Provider
10.1.1 Creating Providers for AKS
10.1.2 Retrieving List of Providers for AKS
10.1.3 Retrieving Specific Provider for AKS
10.1.4 Modifying Providers for AKS
10.1.5 Deleting Providers for AKS

10.2 Administering v3 Clusters on AKS
10.2.1 Creating AKS clusters
10.2.2 Retrieving all clusters
10.2.3 Retrieving Specific AKS Clusters
10.2.4 Modifying AKS clusters
10.2.5 Deleting AKS clusters

11 Managing Clusters on GKE
11.1 Managing v3 GKE Provider
11.1.1 Creating Providers for GKE
11.1.2 Retrieving List of Providers for GKE
11.1.3 Retrieving Specific Provider for GKE
11.1.4 Modifying Providers for GKE
11.1.5 Deleting Providers for GKE

11.2 Administering v3 Clusters on GKE
11.2.1 Creating GKE clusters
11.2.2 Retrieving all clusters
11.2.3 Retrieving Specific GKE Clusters
11.2.4 Modifying GKE clusters
11.2.5 Deleting GKE clusters
1 Overview

Cisco Container Platform APIs provide REST APIs 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 using resources 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 Cisco Container Platform 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 Documentation

You can access the Cisco Container Platform API documentation using the following URL:

- For v2 Clusters:
  https://<CCP IP>/2/swaggerapi
- For v3 Clusters:
  https://<CCP IP>/v3/openapi/

Where, <CCP IP> is the IP address of the web user interface for the Cisco Container Platform control plane that you had used during the installation of Cisco Container Platform. In other words, <CCP IP> is the IP address of the Ingress Controller LoadBalancer.

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 for Cisco Container Platform to connect to the infrastructure provider is represented by the 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 Logging in to Cisco Container Platform

Cisco Container Platform uses an authentication token (auth-token) for authorizing users. You must pass the auth-token in all HTTP requests.

Procedure

1. Set the $CCP environment variable.

   Command

   ```sh
   export CCP=https://<Control Plane VIP>
   ```

   Example

   ```sh
   export CCP=https://10.20.30.40
   ```

2. Generate an authentication token (auth-token).

   Commands

   ```sh
   export TOKEN=$(curl -v -k -X POST \
   -H "Content-Type:application/x-www-form-urlencoded" \
   -d "username=<username>&password=<password>" \
   $CCP/v3/system/login/ 2> >(grep -i x-auth-token) | \
   grep -i x-auth-token | awk -F ":" '{print $2}' | tr -d \"\n\r\")
   ```

   Example
export TOKEN=$(curl -v -k -X POST \ 
-H "Content-Type:application/x-www-form-urlencoded" \ 
-d "username=<username>&password=<password>" \ 
$CCP/v3/system/login/ 2> >(grep -i x-auth-token) | \ 
grep -i x-auth-token | awk -F "":"(print $2)" | tr -d '\n')

Note: An Auth-token has an expiration time, after which it becomes invalid. In such cases, the server will reject it, and you need to generate a new token.

5 Managing Users

5.1.1 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 $CCP environment variable.

   Command
   
   ```bash
   export CCP=https://<Control Plane VIP>
   ```

   Example
   
   ```bash
   export CCP=https://10.20.30.40
   ```

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

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

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

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

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

   Example
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

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

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 $CCP/2/ldap/setup

Example

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

Response

{
  "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 $CCP/2/ldap/setup

Example

curl -k -b cookie.txt $CCP/2/ldap/setup

Response

{
  "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
}

5.1.2 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 $CCP environment variable.

Command

export CCP=https://<Control Plane VIP>

Example

export CCP=https://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>' $CCP/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>' $CCP/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 $CCP/2/ldap/groups
   ```

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

   **Response**
   ```
{
"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 $CCP/2/ldap/groups
   ```

   **Example**
   ```
curl -sk -b cookie.txt $CCP/2/ldap/groups
   ```
Response

[  
 {  
 "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 $CCP/2/clusters| jq -r ".[[]].name, .uuid"

Example

curl -sk -b cookie.txt $CCP/2/clusters| jq -r ".[[]].name, .uuid"

Response

tc1 aef65a35-c013-4d91-9edb-e2ef8359f95b
tc2 8dab31ef-3efa-4de6-9e0d-07e6ff68bc24
tc3 a523fce7-b71e-444a-9626-871e17fe1fcd
tc4 8ccaa3a1-8a11-4996-9224-5723b7ecfd

7. Export the selected tenant cluster.

Command

eexport TC=<Selected tenant cluster>

Example

eexport TC=8ccaa3a1-8a11-4996-9224-5723b7ecfd

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 $CCP/2/clusters/${TC}/authz
Example

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

Example

curl -sk -b cookie.txt $CCP/2/clusters/${TC}/authz

Response

{
  "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>" $CCP/2/system/login/

Example

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

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

Command

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

Example

curl -sk -b cookie_user.txt $CCP/2/clusters| jq -r '.[].name, .uuid'
13. Export the selected tenant cluster.

**Command**

```
export TC=<Selected tenant cluster>
```

**Example**

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


**Command**

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

**Example**

```
curl -sk -b cookie.txt $CCP/2/clusters/${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
```

**Response**

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
<th>ROLES</th>
<th>AGE</th>
<th>VERSION</th>
<th>EXTERNAL-IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>tc4-mc29ab3f9fd</td>
<td>Ready</td>
<td>master</td>
<td>1h</td>
<td>v1.9.2</td>
<td>10.20.30.250</td>
</tr>
<tr>
<td>Ubuntu 16.04.3 LTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tc4-w0d6e5b1836</td>
<td>Ready</td>
<td>&lt;none&gt;</td>
<td>1h</td>
<td>v1.9.2</td>
<td>10.20.30.151</td>
</tr>
<tr>
<td>Ubuntu 16.04.3 LTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tc4-w5ddf2d9f087</td>
<td>Ready</td>
<td>&lt;none&gt;</td>
<td>1h</td>
<td>v1.9.2</td>
<td>10.20.30.150</td>
</tr>
<tr>
<td>Ubuntu 16.04.3 LTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

17. Remove AD group access.

**Command**
#curl -sk -b cookie.txt -X DELETE $CCP/2/ldap/groups/<DN of Group>

**Example**

curl -sk -b cookie.txt -X DELETE $CCP/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 $CCP/2/clusters/${TC}/authz

**Example**

curl -sk -b cookie.txt $CCP/2/clusters/${TC}/authz
{
   "AuthList": []
}

6 Managing v3 Clusters on vSphere

Cisco Container Platform offers API support for v3 clusters to manage providers and clusters across the EKS, vSphere, AKS, and GKE environments.

6.1 Managing v3 vSphere Provider

6.1.1 Creating Providers for vSphere

**Procedure**

1. Log in to the Cisco Container Platform API on the control plane using the /v3/system/login/ endpoint and get an authentication token.

   For more information, see [Logging in to Cisco Container Platform](#).

2. Create a vSphere provider profile.

   **Command**

   ```
   curl -k -X POST -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d '{
   "type": "vsphere",
   "name": "name_of_vsphere_provider",
   "address": "vCenter_url",
   "username": "vCenter_username",
   "password": "vCenter_password",
   "port": vCenter_port,
   "insecure_skip_verify": true_or_false
   }' $CCP/v3/providers/
   ```

   **Example**
6.1.2 Retrieving List of Providers

Procedure

1. Log in to the Cisco Container Platform API on the control plane using the /v3/system/login/ endpoint and get an authentication token.

   For more information, see Logging in to Cisco Container Platform.

2. Retrieve the list of providers.

   Command
   ```bash
curl -k -X POST -H "x-auth-token: $TOKEN" -H "content-type: application/json" -d '{
   "type": "vsphere",
   "name": "aruna",
   "address": "hx3-vcenter.cpsg.ciscolabs.com",
   "username": "administrator@vsphere.local",
   "password": "password",
   "port": 443,
   "insecure_skip_verify": true
}' $CCP/v3/providers/
```

6.1.3 Retrieving Specific Provider

Procedure

1. Log in to the Cisco Container Platform API on the control plane using the /v3/system/login/ endpoint and get an authentication token.

   For more information, see Logging in to Cisco Container Platform.

2. Using the UUID of the provider, retrieve the specific provider.

   Command
   ```bash
curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/providers/
```

   Example
   ```bash
```

   Response
   ```json
{
   "id": "b54efda6-78c7-4418-9b89-955da6585984",
   "type": "vsphere",
   ...}
```
6.1.4 Modifying vSphere Provider

Procedure

1. Log in to the Cisco Container Platform API on the control plane using the /v3/system/login/ endpoint and get an authentication token.

   For more information, see Logging in to Cisco Container Platform.

2. Modify the parameters of the vSphere provider.

   Command

   ```
   curl -k -X PATCH -H "x-auth-token: $TOKEN" -d \
   '{
     "type": "vsphere",
     "name": "name_of_vsphere_provider",
     "address": "vCenter_url",
     "username": "vCenter_username",
     "password": "vCenter_password",
     "port": "vCenter_port",
     "insecure_skip_verify": true_or_false
   }' $CCP/v3/providers/your_provider_id/
   ```

   Example

   ```
   curl -k -X PATCH -H "x-auth-token: $TOKEN" -d \
   '{
     "type": "vsphere",
     "name": "vcenter-1",
     "address": "vcenter.domain.com",
     "username": "administrator@vsphere.local",
     "password": "password",
     "port": "443",
     "insecure_skip_verify": true
   }' https://10.20.30.40/v3/providers/b54efda6-78c7-4418-9b89-955da6585984/
   ```

   Response

   ```
   {
     "id": "b54efda6-78c7-4418-9b89-955da6585984",
     "type": "vsphere",
     "name": "vcenter-1",
     "address": "vcenter.domain.com",
     "port": 443,
     "username": "administrator@vsphere.local",
   }
   ```
6.1.5 Deleting vSphere Provider

Procedure

1. Log in to the Cisco Container Platform API on the control plane using the /v3/system/login/ endpoint and get an authentication token.

   For more information, see Logging in to Cisco Container Platform.

2. Using the UUID of the provider, delete the vSphere provider.

   Command

   ```
curl -k -X DELETE -H "x-auth-token: $TOKEN" $CCP/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/
   ```

6.2 Administering v3 Clusters on vSphere

6.2.1 Creating vSphere Cluster

Example

```bash
curl -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d '{
"type": "vsphere",
"provider": "cf900fac-9d65-4281-b1bb-9b415221cea3",

"name": "cluster-name",
"vsphere_infra": {
"folder": "folder",
"datacenter": "datacenter",
"datastore": "datastore",
"networks": ["network"],
"cluster": "cluster",
"resource_pool": "resource_pool"
},

"master_group": {
"name": "group1",
"size": 3,
"kubernetes_version": "1.2.3"
},

"network_plugin_profile": {
"details": {
"pod_cidr": "10.0.0.0/24"
}
}
'
Response

{
  "id": "6b0678b2-4d34-456d-b060-3106ee433c23",
  "type": "vsphere",
  "name": "cluster7",
  "provider": "cf900fac-9d65-4281-b1bb-9b415221cea3",
  "status": null,
  "kubernetes_version": null,
  "kubeconfig": null,
  "ip_allocation_method": "ccpnet",
  "master_vip": "1.2.3.4",
  "load_balancer_num": 3,
  "subnet_id": "5c2f63d5-5821-439f-acd5-fb8ddd559cac",
  "ntp_pools": [],
  "ntp_servers": [],
  "root_ca_registries": [],
  "self_signed_registries": {},
  "insecure_registries": [],
  "docker_http_proxy": null,
  "docker_https_proxy": null,
  "docker_bip": null,
  "vsphere_infra": {
    "datacenter": "foo",
    "datastore": "foo",
    "networks": [
      "foo"
    ],
    "cluster": "foo",
    "resource_pool": "ayyy",
    "folder": "yeet"
  },
  "master_group": {
    "name": "foo",
    "size": 3,
    "vcpus": 2,
    "memory_mb": 16384,
    "gpus": [],
    "ssh_user": "",
    "ssh_key": "",
    "nodes": [],
    "kubernetes_version": "1.2.3"
  },
  "node_groups": []
}
"network_plugin_profile": {
    "details": {
        "pod_cidr": "10.0.0.0/24"
    }
},
"ingress_as_lb": true,
"nginx_ingress_class": "",
"etcd_encrypted": false,
"skip_management": true,
"control_plane_migration": false,
"docker_no_proxy": [
    "foo",
    "bar"
],
"routable_cidr": null,
"image_prefix": null,
"aci_profile": null,
"description": "",
"cloud_provider": null

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

6.2.2 Retrieving all Clusters

Procedure

1. Log in to the Cisco Container Platform API on the control plane using the /v3/system/login/ endpoint and get an authentication token.

   For more information, see Logging in to Cisco Container Platform.

2. Retrieve all clusters.

   Command

   curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/clusters/

   Example


   Response

   [ 
    
    {"id": "35de61b9-5175-40d5-bea3-1b058fb22c45",
     "type": "vsphere",
     "name": "demo-cluster",}
"provider": "b54efda6-78c7-4418-9b89-955da6585984",
"status": "READY",
"spec": {
"name": "demo-cluster",
"type": "vsphere",
"kubernetes_version": "1.13.5",
"ip_allocation_method": "ccpnet",
"master_vip": "",
"load_balancer_num": 1,
"subnet_id": "ea042d99-9c69-43f8-ac44-ab0b9c843dcf",
"ntp_pools": [],
"ntp_servers": [],
"root_ca_registries": [],
"self_signed_registries": {},
"vsphere_infra": {
"cluster": "HX3",
"datacenter": "HX3",
"datastore": "hx3-data",
"folder": "",
"guestOS": "",
"hostSystem": "",
"networks": [
"VLAN 1161 - 10.10.100.0 - 22"
],
"resource_pool": ""
},
"master_group": {
"gpus": [],
"labels": null,
"name": "master-group",
"size": 1,
"taints": null,
"template": "ccp-tenant-image-1.18.14-8.0.ova",
"vcpus": 2,
"memory_mb": 16384,
"ssh_key": "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAHhxzv/zy/uhF567CqRlo71Z7Wo4Wk/3+H5APXv1cRM6",
"ssh_user": "ccpuser",
"nodes": [
{
"name": "demo-cluster-0-master-0",
"status": "ERROR",
"phase": "Running",
"private_ip": "10.10.100.109",
"public_ip": "10.10.100.109"
}]
},
"node_groups": [

"gpus": [],
"labels": null,
"name": "node-group",
"size": 1,
"taints": null,
"template": "ccp-tenant-image-1.18.14-8.0.0.ova",
"vcpus": 2,
"memory_mb": 16384,
"ssh_key": "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAInhzv/Zy/uHF567CqR1o71Z7Wo4Wk/3+",
"ssh_user": "ccpuser",
"nodes": [
  {
    "name": "demo-cluster-1-node-gr-0",
    "status": "READY",
    "phase": "Running",
    "private_ip": "10.10.100.108",
    "public_ip": "10.10.100.108"
  }
],
"network_plugin_profile": {
    "details": {
      "typhaReplicas": "1",
      "pod_cidr": "192.168.0.0/16",
      "ssh_user": "ccpuser"
    },
    "name": "calico"
},
"kubernetes_config_secret": "demo-cluster-kubeconfig",
"ingress_as_lb": true,
"nginx_ingress_class": "",
"etcd_encrypted": false,
"skip_management": null,
"docker_no_proxy": []
},
"kubeconfig": "...
"kubernetes_version": "1.13.5",
"kubernetes_config_secret": null,
"ip_allocation_method": "ccpnet",
"master_vip": "",
"load_balancer_num": 1,
"subnet_id": "ea042d99-9c69-43f8-ac44-ab0b9c843dcf",
"ntp_pools": [],
"ntp_servers": [],
"root_ca_registries": [],
"self_signed_registries": {},
"insecure_registries": [],
"docker_http_proxy": "",
"docker_https_proxy": "",
"vsphere_infra": {
    "datacenter": "HX3",
    "datastore": "hx3-data",
    "networks": [
        "VLAN 1161 - 10.10.100.0 - 22"
    ],
    "cluster": "HX3",
    "resource_pool": "",
    "folder": ""
},
"master_group": {
    "name": "master-group",
    "size": 1,
    "template": "ccp-tenant-image-1.18.14-8.0.0.ova",
    "vcpus": 2,
    "memory_mb": 16384,
    "gpus": [],
    "ssh_user": "ccpuser",
    "ssh_key": "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5A AAAINhzxv/Zy/uHF567CqR1o71Z7Wo4Wk/3+H5APXv1cRM6"
},
"nodes": [
    {
        "name": "demo-cluster-0-master-0",
        "status": "ERROR",
        "phase": "Running",
        "private_ip": "10.10.100.109",
        "public_ip": "10.10.100.109"
    }
],
"node_groups": [
    {
        "name": "node-group",
        "size": 1,
        "template": "ccp-tenant-image-1.18.14-8.0.0.ova",
        "vcpus": 2,
        "memory_mb": 16384,
        "gpus": [],
        "ssh_user": "ccpuser",
        "ssh_key": "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5A AAAINhzxv/Zy/uHF567CqR1o71Z7Wo4Wk/3+H5APXv1cRM6"
    },
    {
        "name": "demo-cluster-1-node-gr-0",
        "status": "READY",
        "phase": "Running",
        "private_ip": "10.10.100.108",
        "public_ip": "10.10.100.108"
6.2.3 Retrieving Specific Clusters

1. Log in to the Cisco Container Platform API on the control plane using the /v3/system/login/ endpoint and get an authentication token.

   For more information, see Logging in to Cisco Container Platform.

2. Using the UUID of the cluster, retrieve the details of the cluster.

   **Command**

   ```bash
   curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/clusters/<your_cluster_uuid>/
   ```

   **Example**

   ```bash
   ```

   **Response**

   ```json
   {
     "id": "35de61b9-5175-40d5-bea3-1b058fb22c45",
     "type": "vsphere",
     "name": "demo-cluster",
     "provider": "b54efda6-78c7-4418-9b89-955da6585984",
     "status": "READY",
     "spec": {
       "name": "demo-cluster",
       "type": "vsphere",
       "kubernetes_version": "1.13.5",
       "ip_allocation_method": "ccpnet",
       "master_vip": "",
       "network_plugin_profile": {
         "details": {
           "typhaReplicas": "1",
           "pod_cidr": "192.168.0.0/16",
           "ssh_user": "ccpuser"
         },
         "name": "calico",
         "ingress_as_lb": true,
         "nginx_ingress_class": "",
         "etcd_encrypted": false,
         "skip_management": false,
         "docker_no_proxy": [],
         "routable_cidr": null,
         "image_prefix": null,
         "aci_profile": null
       }
     }
   }
   ```
"load_balancer_num": 1,
"subnet_id": "ea042d99-9c69-43f8-ac44-ab0b9c843d8c",
"ntp_pools": [],
"ntp_servers": [],
"root_ca_registries": [],
"self_signed_registries": {},
"vsphere_infra": {
  "cluster": "HX3",
  "datacenter": "HX3",
  "datastore": "hx3-data",
  "folder": "",
  "guestOS": "",
  "hostSystem": "",
  "networks": [
    "VLAN 1161 - 10.10.100.0 - 22"
  ],
  "resource_pool": ""
},
"master_group": {
  "gpus": [],
  "labels": null,
  "name": "master-group",
  "size": 1,
  "taints": null,
  "template": "ccp-tenant-image-1.18.14-8.0.0.ova",
},
"kubernetes_version": "1.13.5",
"kubernetes_config_secret": null,
"ip_allocation_method": "ccpnet",
"master_vip": "",
"load_balancer_num": 1,
"subnet_id": "ea042d99-9c69-43f8-ac44-ab0b9c843d8c",
"ntp_pools": [],
"ntp_servers": [],
"root_ca_registries": [],
"self_signed_registries": {},
"insecure_registries": [],
"docker_http_proxy": "",
"docker_https_proxy": "",
"vsphere_infra": {
  "datacenter": "HX3",
  "datastore": "hx3-data",
  "networks": [
    "VLAN 1161 - 10.10.100.0 - 22"
  ],
  "cluster": "HX3",
  "resource_pool": "",
  "folder": ""
},
"master_group": {
  "name": "master-group",
  "size": 1,
  "template": "ccp-tenant-image-1.18.14-ubuntu18-8.0.0.ova"
"vcpus": 2,
"memory_mb": 16384,
"gpus": [],
"ssh_user": "ccpuser",
"ssh_key": "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINhxv/Zy/uHF567CqR1o71Z7Wo4Wk/3+H5APXv1cRM6",
"nodes": [
{
"name": "demo-cluster-0-master-0",
"status": "ERROR",
"phase": "Running",
"private_ip": "10.10.100.109",
"public_ip": "10.10.100.109"
}
],
"node_groups": [
{
"name": "node-group",
"size": 1,
"template": "ccp-tenant-image-1.18.14-ubuntu18-8.0.ova",
"vcpus": 2,
"memory_mb": 16384,
"gpus": [],
"ssh_user": "ccpuser",
"ssh_key": "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINhxv/Zy/uHF567CqR1o71Z7Wo4Wk/3+H5APXv1cRM6",
"nodes": [
{
"name": "demo-cluster-1-node-gr-0",
"status": "READY",
"phase": "Running",
"private_ip": "10.10.100.108",
"public_ip": "10.10.100.108"
}
]
},
"network_plugin_profile": {
"details": {
"typhaReplicas": "1",
"pod_cidr": "192.168.0.0/16",
"ssh_user": "ccpuser"
},
"name": "calico"
},
"ingress_as_lb": true,
"nginx_ingress_class": "",
"etcd_encrypted": false,
"skip_management": false,
"docker_no_proxy": [],
"routable_cidr": null,
6.2.4 Modifying vSphere Clusters

Example

curl -XPATCH -H "x-auth-token: $TOKEN" -d {"master_vip": "2.3.4.5"} $CCP/v3/clusters/cluster_uuid/

Response

{
   "id": "945625ce-4511-4d1f-9375-fa2fa43ffe23",
   "type": "vsphere",
   "provider": "cf900fac-9d65-4281-b1bb-9b415221cea3",
   "name": "cluster-name",
   "vsphere_infra": {
      "folder": "folder",
      "datacenter": "datacenter",
      "datastore": "datastore",
      "networks": ["network"],
      "cluster": "cluster",
      "resource_pool": "resource_pool"
   },
   "master_group": {
      "name": "group1",
      "size": 3,
      "kubernetes_version": "1.2.3"
   },
   "network_plugin_profile": {
      "details": {
         "pod_cidr": "10.0.0.0/24"
      }
   },
   "node_groups": [],
   "ip_allocation_method": "ccpnet",
   "master_vip": "2.3.4.5",
   "docker_no_proxy": ["host1", "host2"],
   "load_balancer_num": 3,
   "subnet_id": "5c2f63d5-5821-439f-acd5-fb8ddd559cac",
   "aci_profile_name": "optional-aci-name"
}

6.2.5 Deleting vSphere Clusters

1. Log in to the Cisco Container Platform API on the control plane using the /v3/system/login/ endpoint and get an authentication token.

   For more information, see Logging in to Cisco Container Platform.

2. Using the UUID of the cluster, delete the cluster.
Command

curl -k -X DELETE -H "x-auth-token: $TOKEN" $CCP/v3/cluster/cluster_uuid/

Example


6.2.6 Listing Add-ons

You can manage Helm charts using the add-ons API. Add-ons are installed on a tenant cluster. Follow these steps to list the add-ons available for a cluster.

Procedure

1. Log in to the Cisco Container Platform API on the control plane using the /v3/system/login/ endpoint and get an authentication token.

   For more information, see [Logging in to Cisco Container Platform](#).

2. Using the UUID of the cluster, list the add-ons available for a cluster.

   Command

   curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/clusters/<your_cluster_uuid>/addons/

   Example

   export CLUSTER=35de61b9-5175-40d5-bea3-1b058fb22c45

   Response

   {  
      "count": 2,
      "next": null,
      "previous": null,
      "results": [
         {  
            "name": "ccp-monitor",
            "namespace": "default",
            "overrides": "",
            "overrideFiles": [],
            "status": {},
            "url": "/opt/ccp/charts/ccp-monitor.tgz"
         },
         {  
            "name": "metrics",
            "namespace": "default",
            "overrides": "",
            "overrideFiles": [],
            "status": {}
         }
      ]
   }
6.2.7 Listing Catalog of Add-ons

You can list the built-in add-ons that you can install on a tenant cluster using the catalog.

**Procedure**

1. Log in to the Cisco Container Platform API on the control plane using the /v3/system/login/ endpoint and get an authentication token.

   For more information, see [Logging in to Cisco Container Platform](#).

2. List the built-in add-ons.

   **Command**

   ```
   curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/clusters/<your_cluster_uuid>/catalog/
   ```

   **Example**

   ```
   export CLUSTER=35de61b9-5175-40d5-bea3-1b058fb22c45
   ```

   **Response**

   ```
   {
   "_ccp-monitor": {
   "displayName": "Monitoring",
   "name": "ccp-monitor",
   "namespace": "ccp",
   "description": "Monitoring",
   "url": "/opt/ccp/charts/ccp-monitor.tgz"
   },
   "_ccp-efk": {
   "displayName": "Logging",
   "name": "ccp-efk",
   "namespace": "ccp",
   "description": "Logging",
   "url": "/opt/ccp/charts/ccp-efk.tgz"
   },
   "_ccp-kubernetes-dashboard": {
   "displayName": "Dashboard",
   "name": "kubernetes-dashboard",
   "namespace": "ccp",
   "description": "Dashboard",
   "url": "/opt/ccp/charts/kubernetes-dashboard.tgz",
   "overrideFiles": [
   
   ```
6.2.8 Configuring Addons

You can install the add-ons listed in the catalog on a tenant cluster.

Procedure

1. Log in to the Cisco Container Platform API on the control plane using the /v3/system/login/ endpoint and get an authentication token.

   For more information, see Logging in to Cisco Container Platform.

2. Configure the add-ons.

   Command

   ```
   curl -k -v -H "Content-Type:application/json" -H "x-auth-token: $TOKEN" $CCP/v3/clusters/$CLUSTER/addons/ -d '{"name":"addon_name", "url": "addn_url"}'
   ```

   For built-in add-ons, you can use the response from the add-on catalog listing command as the payload for an add-on creation. The payload from the catalog also includes the namespace into which the add-ons are installed.

   Example

   ```
   ```

   Response

   ```
   {
   "name": "ccp-monitor",
   "namespace": "ccp",
   "url": "/opt/ccp/charts/ccp-monitor.tgz"
   }
   ```

6.2.9 Adding Addons with Overrides

Procedure

1. Log in to the Cisco Container Platform API on the control plane using the /v3/system/login/ endpoint and get an authentication token.

   For more information, see Logging in to Cisco Container Platform.
2. Consider the following sample override:

```
prometheus:
  nodeExporter:
    enabled: false
```

This override translates to:

```
{"overrides": "prometheus:
  nodeExporter:
    enabled: false"}
```

```
curl -k -v -H "Content-Type:application/json" -H "x-auth-token: $TOKEN" $CCP/v3/clusters/$CLUSTER/addons/ -d '{
  "name": "ccp-monitor",
  "url": "_ccp-monitor",
  "namespace": "ccp",
  "overrides": "prometheus:
    nodeExporter:
      enabled: false"
}'
```

```
  "name": "ccp-monitor",
  "url": "_ccp-monitor",
  "namespace": "ccp",
  "overrides": "hx:
    url: 10.10.51.9
    token: eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbmlzdHJhdGlvbi5jb21wYW55IiwiZXhwIjoxNjA1MTk1NjU1fQ.7I4TbKZ8v0j7yL1dEJxP3Jt
  "}
```

```
curl -k -v \n- H "Content-Type:application/json" \n- H "x-auth-token: $TOKEN" \n$CCP/v3/clusters/$CLUSTER/addons/ \n-d \n' {
  "name": "ccp-monitor",
  "url": "/opt/ccp/charts/ccp-monitor.tgz",
  "namespace": "ccp",
  "overrides": "prometheus:
    server:
      persistentVolume:
        size: 16Gi
        extraArgs:
          storage.tsdb.size: 8Gi
          storage.tsdb.retention.size: 2Gi"
}'
```

### 6.2.10 Deleting Addons

**Procedure**

1. Log in to the Cisco Container Platform API on the control plane using the /v3/system/login/ endpoint and get an authentication token.
2. Delete the add-on.

**Command**

```
curl -k -v -X DELETE -H "x-auth-token: $TOKEN" $CCP/v3/clusters/$CLUSTER/addons/<addon-name>/
```

**Example**

```
```

**Response**

None

### 6.2.11 Adding Node Pools

**Procedure**

1. Log in to the Cisco Container Platform API on the control plane using the /v3/system/login/ endpoint and get an authentication token.

   For more information, see [Logging in to Cisco Container Platform](#).

2. Get the UUID of the cluster to which you want to add the node pool.

3. Create a request.json file with the following code:

   ```json
   {
     "name": "foo-node-pool",
     "size": 5,
     "vcpus": 2,
     "memory_mb": 16384,
     "gpus": [],
     "kubernetes_version": "1.16.3",
     "template": "ccp-tenant-image-1.18.14-ubuntu18-8.0.0.ova"
   }
   ```

4. Make a request to the API to create the node pool and include the authentication token header.

   **Command**

   ```
curl -H "content-type: application/json" --data @request.json $CCP/v3/clusters/<CLUSTER-UUID>/node-groups/
   ```

   **Example**

   ```
curl -H "content-type: application/json" --data @request.json $CCP/v3/clusters/2b011bdb-ceb7-486d-be02-c5bee1a42a95/node-groups/
   ```
Response

```
{
  "name": "foo-node-pool",
  "size": 5,
  "vcpus": 2,
  "memory_mb": 16384,
  "gpus": [],
  "nodes": [],
  "kubernetes_version": "1.16.3"
}
```

### 6.2.12 Getting List of Node Pools

**Procedure**

1. Log in to the Cisco Container Platform API on the control plane using the `/v3/system/login/` endpoint and get an authentication token. For more information, see [Logging in to Cisco Container Platform](#).

2. Get the UUID of the cluster that contains the node pools.

3. Get the list of node pools in the cluster.

   `/v3/<CLUSTER-UUID>/node-groups/` endpoint

4. Make a request to the API to list node pools in a cluster.

**Command**

```
curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/clusters/<CLUSTER-UUID>/node-groups/
```

**Example**

```
curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/clusters/08351e0d-42a4-4a4c-9458-2d907e6f75f3/node-groups/
```

**Response**

```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "name": "foo-node-pool",
      "size": 5,
      "vcpus": 2,
      "memory_mb": 16384,
      "gpus": [],
      "nodes": [],
      "kubernetes_version": "1.16.3"
    }
  ]
}
```
6.2.13 Modifying Node Pools

Procedure

1. Log in to the Cisco Container Platform API on the control plane using the /v3/system/login/ endpoint and get an authentication token.
   
   For more information, see Logging in to Cisco Container Platform.

2. Get the UUID of the cluster that contains the node pool that you want to modify.

3. Get the list of node pools in the cluster.
   
   /v3/<CLUSTER-UUID>/node-groups/ endpoint

4. Note down the name of node pool that you want to modify.

5. Create a request.json file with the necessary modifications.

   Example
   
   ```json
   {
     "name": "foo-node-pool",
     "size": <NEW-SIZE>,
     "vcpus": 2,
     "memory_mb": 16384,
     "gpus": [],
     "kubernetes_version": "1.16.3",
     "template": "ccp-tenant-image-1.18.14-ubuntu18-8.0.0.ova"
   }
   ```

   Note: You cannot modify the name of a node pool. Modifications to the Kubernetes_version and template fields will trigger an upgrade to the node pool. Modifications to vcpus, memory_mb, and gpus fields will not change the current node configurations in the node pool and will only take effect when the node pool is either upgraded or scaled.

6. Make a PATCH request to the API to modify the node pool that has the authentication token header.

   Command
   
   ```bash
   curl -XPATCH -H "content-type: application/json" --data @request.json $CCP/v3/clusters/<CLUSTER-UUID>/node-pools/<NAME>
   ```

   Example
   
   ```bash
   Curl -XPATCH -H "content-type: application/json"
   ```
6.2.14 Deleting Node Pools

1. Log in to the Cisco Container Platform API on the control plane using the /v3/system/login/ endpoint and get an authentication token.

   For more information, see Logging in to Cisco Container Platform.

2. Get the UUID of the cluster that contains the node pool that you want to modify.

3. Get the list of node pools in the cluster.

   /v3/<CLUSTER-UUID>/node-groups/

4. Note down the name of node pool that you want to delete.

5. Make a DELETE request to the API to delete the node pool that has the authentication token header.

   Command

   curl -XDELETE $CCP/v3/clusters/<CLUSTER-UUID>/node-groups/<NAME>/

   Example

   Curl -XDELETE $CCP/v3/clusters/2b011bdb-ceb7-486d-be02-c5bee1a42a95/node-groups/foo-node-pool/

   Response

   None

6.2.15 Downloading Tenant Cluster KUBECONFIG Environment File

The kubeconfig data is available in the kubeconfig key of the response when fetching a vSphere Cluster.

6.3 Using ACI CNI Network Plugin

6.3.1 Creating ACI Profile

Procedure
1. Log in to the Cisco Container Platform API on the control plane using the `/v3/system/login/` endpoint and get an authentication token.

   For more information, see [Logging in to Cisco Container Platform](#).

2. Create an ACI profile.

   **Example**

   ```
curl -d ' {
   "name": "example-aci-profile",
   "apic_username": "username",
   "apic_password": "password",
   "aci_tenant": "aci_tenant",
   "apic_hosts": "apic_hosts",
   "aci_vmm_domain_name": "aci_vmm_domain_name",
   "vrf_name": "vrf_name",
   "l3_outside_policy_name": "l3_outside_policy_name",
   "l3_outside_network_name": "l3_outside_network_name",
   "aaep_name": "aaep_name",
   "nameservers": ["nameservers"],
   "aci_infra_vlan_id": 1234,
   "node_vlan_start": 1,
   "node_vlan_end": 100,
   "multicast_range": "10.0.0.0/16",
   "service_subnet_start": "20.15.1.1/16",
   "pod_subnet_start": "10.2.0.0/16",
   "aci_profile_name": "aci_profile_name",
   "control_plane_contract_name": "control_plane_contract_name"
   ```

   **Response**

   ```
   {
   "id": "f0dcf8a3-0253-4a25-83a9-6b0b695e508c",
   "cluster_count": 0,
   "name": "example-aci-profile",
   "apic_hosts": "apic_hosts",
   "apic_username": "username",
   "aci_vmm_domain_name": "aci_vmm_domain_name",
   "aci_infra_vlan_id": 1234,
   "vrf_name": "vrf_name",
   "l3_outside_policy_name": "l3_outside_policy_name",
   "l3_outside_network_name": "l3_outside_network_name",
   "aaep_name": "aaep_name",
   "nameservers": ["nameservers"],
   "control_plane_contract_name": "control_plane_contract_name"
   }
   ```
6.3.2 Creating ACI-enabled vSphere Cluster

```bash
curl -d '{"type":"vsphere", "provider": "276ed502-1b95-4329-859e-12289d37953b", "name":"example-vsphere-cluster", "kubernetes_version":"1.12.7", "vsphere_infra":{"folder":"placeholder", "datacenter":"placeholder", "datastore":"placeholder", "networks":["placeholder"], "cluster":"placeholder", "resource_pool":"placeholder"}, "master_group":{"name":"placeholder", "size":1234}, "network_plugin_profile":{"details":{"pod_cidr":"10.0.0.0/24"}}, "node_groups":[]}, "ip_allocation_method":"ccpnet", "master_vip":"1.2.3.4","skip_management":true, "docker_no_proxy":[]}' $CCP/v3/clusters/
```

6.3.3 Updating ACI Profile

**Command**

```bash
curl -XPATCH -d '{"aaep_name":"new_aaep_name"}' $CCP/v3/aci-profiles/aadb0435-775d-445d-9bac-37dfcad1eb89/
```

**Note:** The cluster has to be PATCHed to pick up the new ACI details. This is by design.

**Example**

```bash
curl -s -XPATCH -d '{}' https://10.20.30.40/v3/clusters/d7dc05c7-78a6-4ff7-9657-1ac48ee09dcb/
```

6.3.4 Deleting ACI Profile

**Example**

```bash
```

7 Managing v2 Clusters on vSphere

**Note:** v2 clusters are currently being deprecated.

You can deploy v2 and v3 clusters in a vSphere environment.
7.1 Administering v2 Clusters on vSphere

7.1.1 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 SCCP environment variable.

   Command
   
   ```bash
   export CCP=https://<Control Plane VIP>
   ```

   Example
   
   ```bash
   export CCP=https://10.20.30.40
   ```

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

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

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

3. Get list of Provider Client Configurations.

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

   Example
   
   ```bash
   curl -sk -b cookie.txt -H "Content-Type: application/json" $CCP/2/providerclientconfigs/ | jq '.[].uuid'
   ```

   Response
   
   ```bash
   "fb53eae8-d973-4644-b13f-893949154a22"
   ```

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

   Command
   
   ```bash
   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 $CCP/2/providerclientconfigs/${PCC}/vsphere/datacenter | jq '.Datacenters[]'
```

Example

```
curl -sk -b cookie.txt $CCP/2/providerclientconfigs/${PCC}/vsphere/datacenter | jq '.Datacenters[]'
```

Response

"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 $CCP/2/providerclientconfigs/${PCC}/vsphere/datacenter/${DCC}/vm | jq '.VMs[]| select(.| startswith("ccp-tenant-image"))'| sort -u
```

Example

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

Response

"ccp-tenant-image-1.17.14-8.0.0.ova"
"ccp-tenant-image-1.18.14-8.0.0.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-1.18.14-8.0.0.ova
```

9. Get the list of networks.
Command

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

Example

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

Response

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

10. Configure the network that you want to use.

**Command**

```bash
export NETWORK=<From list of Networks>
```

**Example**

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

11. Get the list of clusters.

**Command**

```bash
curl -sk -b cookie.txt $CCP/2/providerclientconfigs/${PCC}/vsphere/datacenter/${DCC}/cluster| jq '.Clusters[]'
```

**Example**

```bash
curl -sk -b cookie.txt $CCP/2/providerclientconfigs/${PCC}/vsphere/datacenter/${DCC}/cluster| jq '.Clusters[]'
```

**Response**

```
"r9-hx2"
```

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

**Command**

```bash
export CLUSTER=<from list of clusters>
```

**Example**

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

13. Get the list of pools.

**Command**

```bash
```
curl -sk -b cookie.txt $CCP/2/providerclientconfigs/${PCC}/vsphere/datacenter/${DCC}/cluster/${CLUSTER}/pool | jq ".Pools[]"

Example

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

Response

"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 $CCP/2/providerclientconfigs/${PCC}/vsphere/datacenter/${DCC}/datastore | jq -r ".Datastores[],| select(.|startswith("SpringpathDS")|not)'
```

**Example**

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

Response

```
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

```bash
export NAME=<Name of cluster>
```

Example

```bash
export NAME=tc4
```

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

Command

```bash
export USER=<Username>
```

Example

```bash
export USER=ccpuser
```

19. Configure the ssh public key for remote access.

Command

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

Example

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

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

20. Get the list of subnets.

Command

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

Example

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

Response

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

21. Configure the subnet for the cluster.

Command

```bash
export SUBNET=<From the list of subnets>
```
Example

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

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

**Command**

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

**Example**

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

**Response**

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

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

**Command**

```bash
export VIP_POOL=<From the list of pools>
```

**Example**

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

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

```bash
#---------------------------------------------------
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": "${USER}";
EOF
```
"ssh_key": "${SSHKEY}",
"deployer_type": "kubeadm",
"kubernetes_version": "1.11.3",
"deployer": {
    "provider_type": "vsphere",
    "provider": {
        "vsphere_datacenter": "${DCC}",
        "vsphere_datastore": "${DATASTORE}",
        "vsphere_client_configUuid": "${PCC}"
    }
}
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 $CCP/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 $CCP/2/clusters | tee output.txt | jq '.name,.uuid,.state'
```

**Response**

```
“tc4”
“8ccaa3a1-8a11-4996-9224-5723b7ecf8dv
“READY”
```

27. Configure the tenant cluster UUID.

**Command**

```
export TC=<UUID of the selected tenant cluster>
```

**Example**

```
export TC=8ccaa3a1-8a11-4996-9224-5723b7ecf8dv
```


**Command**

```
curl -sk -b cookie.txt $CCP/2/clusters/$TC/env -o $TC.env
```

43
Example

```
curl -sk -b cookie.txt $CCP/2/clusters/${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
```

**Response**

```
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-generi
        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
```

### 7.1.2 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 $CCP environment variable.

   **Command**

   ```
   export CCP=https://<Control Plane VIP>
   ```

   **Example**

   ```
   export CCP=https://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>' $CCP/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>' $CCP/2/system/login/

3. List tenant clusters.

Command

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

Example

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

Response

tc1
ae65a35-c013-4d91-9edb-e2ef8359f95b
tc2
8dab31ef-3efa-4de6-9e0d-07e6ff68bc24
tc3
a523fceb7-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 $CCP/2/clusters/${TC}

Example

curl -sk -b cookie.txt -X DELETE $CCP/2/clusters/${TC}

7.1.3 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 $CCP environment variable.

   **Command**
   ```bash
   export CCP=https://<Control Plane VIP>
   ```

   **Example**
   ```bash
   export CCP=https://10.20.30.40
   ```

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

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

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

3. List tenant clusters.

   **Command**
   ```bash
   curl -sk -b cookie.txt $CCP/2/clusters | jq '.[].name, .uuid'
   ```

   **Example**
   ```bash
   curl -sk -b cookie.txt $CCP/2/clusters | jq '.[].name, .uuid'
   ```

   **Response**
   ```text
tc1
    aef65a35-c013-4d91-9edb-e2ef8359f9gb
    tc2
    8dab31ef-3efa-4de6-9e0d-07e6ff68bc24
    tc3
    a523fce7-b71e-444a-9626-871e17fe1fcd
    tc4
    8ccaa3a1-8a11-4996-9224-5723b7ecfddf
   ```

4. Export a tenant cluster.

   **Command**
   ```bash
   export TC=<selected cluster from list>
   ```

   **Example**
export TC=8ccaa3a1-8a11-4996-9224-5723b7ecfd6d

5. Download the KUBECONFIG environmental file.

   **Command**
   
   ```
curl -sk -b cookie.txt $CCP/2/clusters/${TC}/env -o ${TC}.env
   ```
   
   **Example**
   
   ```
curl -sk -b cookie.txt $CCP/2/clusters/${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
   ```

   **Response**
   
   ```
NAME              STATUS     ROLES   AGE   VERSION  EXTERNAL-IP  OS-IMAGE    KERNEL VERSION  CONTAINER-RUNTIME
---                --------    ------   ----  --------  -----------  ----------  ---------------  ----------------
tc4-mc29ab3f9fd  Ready      master  1h    v1.9.2   10.2.0.30.250 Ubuntu 16.04.3 LTS 4.4.0-104-generic docker://1.13.1
tc4-w0d6e5b1836  Ready      <none>  1h    v1.9.2   10.2.0.30.151 Ubuntu 16.04.3 LTS 4.4.0-104-generic docker://1.13.1
tc4-w5dfdd9f087  Ready      <none>  1h    v1.9.2   10.2.0.30.150 Ubuntu 16.04.3 LTS 4.4.0-104-generic docker://1.13.1
   ```

7.1.4 Obtaining TC Master and Ingress VIPs

For Master

```
curl -sk -X GET -b temp/cookie.txt $CCP/2/clusters/<clusternam e> | jq '.master_vip
``` 

For Ingress VIPs
8 Managing v3 Clusters on EKS

8.1 Managing v3 EKS Provider

8.1.1 Creating Providers for EKS

Procedure

1. Log in to Cisco Container Platform. For more information, see Logging in to Cisco Container Platform.

2. Create an EKS provider.

Command

```bash
curl -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d \n'{"type": "eks",
"name": "name_of_your_eks_cluster",
"role_arn": "your_aws_role_arn",
"access_key_id": "your_AWS_access_key_id",
"secret_access_key": "your_AWS_secret_access_key"
}' $CCP/v3/providers/
```

Example

```bash
curl -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d \n'{"type": "eks",
"name": "selvi-eks-provider",
"role_arn": "arn:aws:iam::123456789123:role/eksServiceRole",
"access_key_id": "ABCDEFGHIJKLMNOPQRSTUVWXYZ",
"secret_access_key": "THISISNOTAREALSECRETKEYBUTLOOKLIKEONE"
}' https://10.20.30.40/v3/providers/
```

8.1.2 Retrieving List of Providers for EKS

Command

```bash
curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/providers/
```

Example

```bash
```
8.1.3 Retrieving Specific Provider for EKS

**Command**
```
curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/providers/<provider_uuid>/
```

**Example**
```
```

**Response**
```
{
  "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"
}
```

8.1.4 Modifying Providers for EKS

**Example**
```
curl -X PATCH -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d \
  '{
    "access_key_id": "your_new_AWS_access_key_id",
    "secret_access_key": "your_new_AWS_secret_access_key"
  }' $CCP/v3/providers/<provider_uuid>/
```

**Response**
```
{
  "access_key_id": "your_new_AWS_secret_access_key",
  "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"
}
```

8.1.5 Deleting Providers for EKS

**Command**
```
curl -k -X DELETE -H "x-auth-token: $TOKEN" $CCP/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/
```
8.2 Administering v3 Clusters on EKS

8.2.1 Creating EKS clusters

Command

```
curl -H "content-type: application/json" -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>"]
  },
  "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>", "<optional_another_ssh_key>"]
}"
``` $CCP/v3/clusters/

Example

```
curl -H "content-type: application/json" -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 AAAAC3NzaC1lZDI1NTE5AAAAIHzSrKkW
hwED6aw95sjegF0dgcknotmyrealkey", "another_dummy_key"]
}
``` https://10.20.30.40/v3/clusters/

Response


```
{
  "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/KubernetesAdmin",
  "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.

### 8.2.2 Retrieving all clusters

**Command**

```
curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/clusters/
```

**Example**

```
```

**Response**

```
[
  {
    "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",
  }
]```
8.2.3 Retrieving Specific EKS Clusters

Command

curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/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/

Response

```json
{
    "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/KubernetesAdmin",
    "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"
}
```
8.2.4 Modifying EKS clusters

Command

```
curl -X PATCH -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d \
'{"worker_count": 2}
}' $CCP/v3/clusters/<cluster_uuid>/
```

Example

```
curl -X PATCH -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d \
'{"worker_count": 2}
}' https://10.20.99.190/v3/clusters/5a5f0db5-110c-4151-80e8-9b7889d30bc/
```

Response

```
{
"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/KubernetesAdmin",
"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"
}

8.2.5 Deleting EKS clusters

Command

```
curl -k -X DELETE -H "x-auth-token: $TOKEN" $CCP/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/
```

8.2.6 Downloading Tenant Cluster KUBECONFIG Environment File

The kubeconfig data is available in the kubeconfig key of the response when fetching an EKS Cluster.

9 Managing Clusters on Openstack

9.1 Managing v3 Openstack Provider

9.1.1 Creating Providers for Openstack

Procedure

1. Log in to Cisco Container Platform. For more information, see Logging in to Cisco Container Platform.
2. Create an Openstack provider.

Command

```
curl -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d \''{
  "username": "username",
  "insecure_skip_verify": true,
  "name": "provider name",
  "ca_cert": "cert text here",
  "tenant_name": "tenant name",
  "region": "region name",
  "domain_name": "domain name",
  "region": "region name",
  "image_id": "image_id",
  "image_name": "image name",
  "instance_type": "instance type"
}''
```
"auth_url": "your auth url",
"password": "password",
"type": "openstack"
}' $CCP/v3/providers/

Example

curl -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d ' {
  "username": "username",
  "insecure_skip_verify": true,
  "name": "demo-openstack-provider",
  "ca_cert": "cert text here",
  "tenant_name": "name",
  "region": "region",
  "domain_name": "default",
  "auth_url": "https://1.2.3.4:5000/v3",
  "password": "password",
  "type": "openstack"
}' https://10.20.30.40/v3/providers/

9.1.2 Retrieving List of Providers for Openstack

Command

curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/providers/

Example


Response

{
  "username": "username",
  "insecure_skip_verify": true,
  "name": "demo-openstack-provider",
  "tenant_name": "name",
  "region": "region",
  "domain_name": "default",
  "auth_url": "https://1.2.3.4:5000/v3",
  "type": "openstack"
}

9.1.3 Retrieving Specific Provider for Openstack

Command

curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/providers/<provider_uuid>/

Example

Response

{  
  "username": "username",  
  "insecure_skip_verify": true,  
  "name": "demo-openstack-provider",  
  "tenant_name": "name",  
  "region": "region",  
  "domain_name": "default",  
  "auth_url": "https://1.2.3.4:5000/v3",  
  "type": "openstack"
}

9.1.4 Modifying Providers for OpenStack

Example

curl -X PATCH -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d \  
'  
  "username": "foo"
'  
' $CCP/v3/providers/<provider_uuid>/

Response

{  
  "username": "foo",  
  "insecure_skip_verify": true,  
  "name": "demo-openstack-provider",  
  "tenant_name": "name",  
  "region": "region",  
  "domain_name": "default",  
  "auth_url": "https://1.2.3.4:5000/v3",  
  "type": "openstack"
}

9.1.5 Deleting Providers for OpenStack

Command

   curl -k -X DELETE -H "x-auth-token: $TOKEN" $CCP/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/

9.2 Administering v3 Clusters on Openstack

9.2.1 Creating Openstack clusters

Example
curl -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d \
'\n{
  "type": "openstack",
  "network_type": "tenant",
  "name": "ccp-tenant-cluster",
  "image": "3507aee3-6fb2-40bb-9067-54345c0217ac",
  "worker_count": 1,
  "vm_network_dns_servers": ["1.2.3.4"],
  "pod_cidr": "192.168.0.0/16",
  "ssh_key_name": "ccp-key-pair",
  "provider": "1d96b873-e40f-428b-819b-1f298c1effd3",
  "flavor": "20d96730-8aa4-49ca-8263-73d4ce62a33b",
  "kubernetes_version": "1.2.3",
  "vm_network_subnet": "77.0.0.0/24",
  "public_network_uuid": "f1a8371f-f922-40ce-869d-c544cc50fe55",
  "master_count": 3,
}' https://10.20.30.40/v3/clusters/

**Response**

```json
{
  "id": "945625ce-4511-4d1f-9375-fa2fa43ffe23",
  "type": "openstack",
  "name": "ccp-tenant-cluster",
  "provider": "1d96b873-e40f-428b-819b-1f298c1effd3",
  "status": "CREATING",
  "network_plugin": "calico",
  "network_type": "tenant",
  "public_network_uuid": "f1a8371f-f922-40ce-869d-c544cc50fe55",
  "vm_network_dns_servers": ["1.2.3.4"],
  "kubernetes_version": "1.13.5sdf",
  "pod_cidr": "192.168.0.0/16",
  "ssh_key_name": "ccp-key-pair",
  "master_count": 3,
  "flavor": "20d96730-8aa4-49ca-8263-73d4ce62a33b",
  "image": "3507aee3-6fb2-40bb-9067-54345c0217ac",
  "worker_count": 1,
  "vm_network_subnet": "77.0.0.0/24",
  "ntp_pools": [],
  "ntp_servers": [],
  "root_ca_registries": [],
  "self_signed_registries": [],
  "etcd_encrypted": false,
  "skip_management": false,
  "docker_no_proxy": [],
  "control_plane": true,
  "master_group": [],
  "worker_group": []
}
```
Note: The API returns the values immediately and the status is indicated as CREATING.

9.2.2 Retrieving all clusters

**Command**

```bash
curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/clusters/
```

**Example**

```bash
```

**Response**

```json
[
  {
    "id": "945625ce-4511-4d1f-9375-fa2fa43ffe23",
    "type": "openstack",
    "name": "ccp-tenant-cluster",
    "provider": "1d96b873-e40f-428b-819b-1f298c1effd3",
    "status": "CREATING",
    "network_plugin": "calico",
    "network_type": "tenant",
    "public_network_uuid": "f1a8371f-f922-40ce-869d-c544cc50fe55",
    "vm_network_dns_servers": [
      "1.2.3.4"
    ],
    "kubernetes_version": "1.13.5sdf",
    "pod_cidr": "192.168.0.0/16",
    "ssh_key_name": "ccp-key-pair",
    "master_count": 3,
    "flavor": "20d96730-8aa4-49ca-8263-73d4ce62a33b",
    "image": "3507aee3-6fb2-40bb-9067-54345c0217ac",
    "worker_count": 1,
    "vm_network_subnet": "77.0.0.0/24",
    "ntp_pools": [],
    "ntp_servers": [],
    "root_ca_registries": [],
    "self_signed_registries": {},
    "etcd_encrypted": false,
    "skip_management": false,
    "docker_no_proxy": [],
    "control_plane": true,
    "master_group": {},
    "worker_group": {}
  }
]
```
9.2.3 Retrieving Specific Openstack Clusters

**Command**

```bash
curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/clusters/<your_cluster_uuid>/
```

**Example**

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

**Response**

```json
{
   "id": "945625ce-4511-4d1f-9375-fa2fa43ffe23",
   "type": "openstack",
   "name": "ccp-tenant-cluster",
   "provider": "1d96b873-e40f-428b-819b-1f298c1effd3",
   "status": "CREATING",
   "network_plugin": "calico",
   "network_type": "tenant",
   "public_network_uuid": "f1a8371f-f922-40ce-869d-c544cc50fe55",
   "vm_network_dns_servers": [
      "1.2.3.4"
   ],
   "kubernetes_version": "1.13.5sdf",
   "pod_cidr": "192.168.0.0/16",
   "ssh_key_name": "ccp-key-pair",
   "master_count": 3,
   "flavor": "20d96730-8aa4-49ca-8263-73d4ce62a33b",
   "image": "3507aee3-6fb2-40bb-9067-54345c0217ac",
   "worker_count": 1,
   "vm_network_subnet": "77.0.0.0/24",
   "ntp_pools": [],
   "ntp_servers": [],
   "root_ca_registries": [],
   "self_signed_registries": {},
   "etcd_encrypted": false,
   "skip_management": false,
   "docker_no_proxy": [],
   "control_plane": true,
   "master_group": {},
   "worker_group": {}
}
```

9.2.4 Modifying Openstack clusters

**Command**

```bash
curl -k -X PATCH -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d \\
'{
```
"worker_count": 2
} $CCP/v3/clusters/<cluster_uuid>/

Example

curl -k -X PATCH -H "content-type: application/json" -H "x-aut h-token: $TOKEN" -d \
'{" "worker_count": 2 }
https://10.20.99.190/v3/clusters/5a5f0db5-110c-4151-80e8-9b 78889d30bc/

Response
{
  "id": "945625ce-4511-4d1f-9375-fa2fa43ffe23",
  "type": "openstack",
  "name": "ccp-tenant-cluster",
  "provider": "1d96b873-e40f-428b-819b-1f298c1effd3",
  "status": "CREATING",
  "network_plugin": "calico",
  "network_type": "tenant",
  "public_network_uuid": "f1a8371f-f922-40ce-869d-c544cc50fe 55",
  "vm_network_dns_servers": [ "1.2.3.4"
  ],
  "kubernetes_version": "1.13.5sdf",
  "pod_cidr": "192.168.0.0/16",
  "sshd_key_name": "ccp-key-pair",
  "master_count": 3,
  "flavor": "20d96730-8aa4-49ca-8263-73d4ce62a33b",
  "image": "3507aee3-6fb2-40bb-9067-54345c0217ac",
  "worker_count": 2,
  "vm_network_subnet": "77.0.0.0/24",
  "ntpd_pools": [],
  "ntp_servers": [],
  "self_signed_registries": [],
  "etcd_encrypted": false,
  "skip_management": false,
  "docker_no_proxy": [],
  "control_plane": true,
  "master_group": {},
  "worker_group": {}
}

9.2.5 Deleting Openstack clusters

Command

curl -k -X DELETE -H "x-auth-token: $TOKEN" $CCP/v3/clusters/cl uster_uuid/

Example
10 Managing Clusters on AKS

10.1 Managing v3 AKS Provider

10.1.1 Creating Providers for AKS

Procedure

1. Log in to Cisco Container Platform. For more information, see [Logging in to Cisco Container Platform](#).

2. Create an AKS provider.

   **Example**

   ```bash
curl -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d \\
   '{
   "type": "aks",
   "name": "example",
   "client_id": "client_id",
   "client_secret": "client_secret",
   "tenant_id": "tenant_id",
   "subscription_id": "subscription_id"
   }'
https://10.20.30.40/v3/providers/
```

   **Response**

   ```json
   {
   "id": "56de926b-daad-4382-b6e6-d0f67a2d13c8",
   "type": "aks",
   "name": "example",
   "app_name": "",
   "client_id": "client_id",
   "tenant_id": "tenant_id",
   "subscription_id": "subscription_id"
   }
   ```

10.1.2 Retrieving List of Providers for AKS

**Command**

```bash
curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/providers/
```

**Example**

```bash
```

**Response**
[{
  "id": "56de926b-daad-4382-b6e6-d0f67a2d13c8",
  "type": "aks",
  "name": "example",
  "app_name": "",
  "client_id": "client_id",
  "tenant_id": "tenant_id",
  "subscription_id": "subscription_id"
}]

10.1.3 Retrieving Specific Provider for AKS

**Command**

```bash
curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/providers/<provider_uuid>/
```

**Example**

```bash
```

**Response**

```json
{
  "id": "56de926b-daad-4382-b6e6-d0f67a2d13c8",
  "type": "aks",
  "name": "example",
  "app_name": "",
  "client_id": "client_id",
  "tenant_id": "tenant_id",
  "subscription_id": "subscription_id"
}
```

10.1.4 Modifying Providers for AKS

**Example**

```bash
```

**Response**

```json
{
  "id": "56de926b-daad-4382-b6e6-d0f67a2d13c8",
  "type": "aks",
  "name": "example",
  "app_name": "",
```
10.1.5 Deleting Providers for AKS

**Command**

```
curl -k -X DELETE -H "x-auth-token: $TOKEN" $CCP/v3/providers/ <provider_uid>/
```

**Example**

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

10.2 Administering v3 Clusters on AKS

10.2.1 Creating AKS clusters

**Example**

```
curl -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d "
{
  "type": "aks",
  "name": "cluster-name",
  "provider": "56de926b-daad-4382-b6e6-d0f67a2d13c8",
  "agent_pool_name": "name",
  "kubernetes_version": "1.2.3",
  "location": "location",
  "resource_group_name": "name",
  "worker_instance_type": "foo",
  "worker_count": 3
}
" $CCP/v3/clusters/
```

**Response**

```
{
  "id": "1846e180-3fbd-4388-a980-59f14a6eb0f6",
  "type": "aks",
  "name": "cluster-name",
  "provider": "56de926b-daad-4382-b6e6-d0f67a2d13c8",
  "status": "CREATING",
  "kubeconfig": null,
  "agent_pool_name": "name",
  "kubernetes_version": "1.2.3",
  "location": "location",
  "pod_cidr": "10.244.0.0/16",
  "resource_group_name": "name",
  "virtual_kubelet_enabled": false,
  "service_cidr": "10.0.0.0/16",
  "worker_instance_type": "foo",
  "worker_count": 3
}
```
"worker_count": 3,
"network_plugin": "kubenet",
"vnet_subnet_id": "",
"docker_bridge_cidr": null,
"dns_service_ip": null
}

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

### 10.2.2 Retrieving all clusters

**Command**

```bash
curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/clusters/
```

**Example**

```bash
```

**Response**

```
[
{
  "id": "1846e180-3fb3-4388-a980-59f14a6eb0f6",
  "type": "aks",
  "name": "cluster-name",
  "provider": "56de926b-daad-4382-b6e6-d0f67a2d13c8",
  "status": "CREATING",
  "kubeconfig": null,
  "agent_pool_name": "name",
  "kubernetes_version": "1.2.3",
  "location": "location",
  "pod_cidr": "10.244.0.0/16",
  "resource_group_name": "name",
  "virtual_kubelet_enabled": false,
  "service_cidr": "10.0.0.0/16",
  "worker_instance_type": "foo",
  "worker_count": 3,
  "network_plugin": "kubenet",
  "vnet_subnet_id": "",
  "docker_bridge_cidr": null,
  "dns_service_ip": null
}
]
```

### 10.2.3 Retrieving Specific AKS Clusters

**Command**

```bash
curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/clusters/<your_cluster_uuid>/
```

**Example**
curl -k -X GET -H "x-auth-token: $TOKEN" https://10.10.99.190/v3/clusters/5a5f0db5-110c-4151-80e8-9b7889d30bc/

Response

{
  "id": "1846e180-3fbfd4388-a980-59f14a6eb0f6",
  "type": "aks",
  "name": "cluster-name",
  "provider": "56de926b-daad-4382-b6e6-d0f67a2d13c8",
  "status": "CREATING",
  "kubeconfig": null,
  "agent_pool_name": "name",
  "kubernetes_version": "1.2.3",
  "location": "location",
  "pod_cidr": "10.244.0.0/16",
  "resource_group_name": "name",
  "virtual_kubelet_enabled": false,
  "service_cidr": "10.0.0.0/16",
  "worker_instance_type": "foo",
  "worker_count": 3,
  "network_plugin": "kubenet",
  "vnet_subnet_id": "",
  "docker_bridge_cidr": null,
  "dns_service_ip": null
}

10.2.4 Modifying AKS clusters

Command

curl -k -X PATCH -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d \
'\n  "worker_count": 2
}' $CCP/v3/clusters/<cluster_uuid>/

Example

curl -k -X PATCH -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d \
'\n  "worker_count": 2
}' https://10.20.99.190/v3/clusters/5a5f0db5-110c-4151-80e8-9b7889d30bc/

Response

{  
  "id": "1846e180-3fbfd4388-a980-59f14a6eb0f6",
  "type": "aks",
  "name": "cluster-name",
  "provider": "56de926b-daad-4382-b6e6-d0f67a2d13c8",
  "status": "CREATING",
  "kubeconfig": null,
}
"agent_pool_name": "name",
"kubernetes_version": "1.2.3",
"location": "location",
"pod_cidr": "10.244.0.0/16",
"resource_group_name": "name",
"virtual_kubelet_enabled": false,
"service_cidr": "10.0.0.0/16",
"worker_instance_type": "foo",
"worker_count": 2,
"network_plugin": "kubenet",
"vnet_subnet_id": "",
"docker_bridge_cidr": null,
"dns_service_ip": null
}

10.2.5 Deleting AKS clusters

Command

curl -k -X DELETE -H "x-auth-token: $TOKEN" $CCP/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/

11 Managing Clusters on GKE

11.1 Managing v3 GKE Provider

11.1.1 Creating Providers for GKE

Procedure

1. Log in to Cisco Container Platform. For more information, see Logging in to Cisco Container Platform.

2. Create an GKE provider.

Example

curl -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d \
'{
  "type":"gke",
  "name":"providername",
  "project_id": "redacted",
  "credentials": {
    "type": "service_account",
    "project_id": "redacted",
    "private_key_id": "redacted",
    "private_key": "redacted",
    "client_email": "redacted",
  }
}'}
"client_id": "redacted",
"auth_uri": "https://accounts.google.com/o/oauth2/auth",
"token_uri": "https://oauth2.googleapis.com/token",
"auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
"client_x509_cert_url": "redacted"
}' $CCP/v3/providers/

Response
{
"id": "eb13196c-2a72-457a-9c96-d2a38b94fec3",
"type": "gke",
"name": "providername",
"project_id": "redacted"
}

11.1.2 Retrieving List of Providers for GKE

Command

curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/providers/

Example


Response
[
{
"id": "eb13196c-2a72-457a-9c96-d2a38b94fec3",
"type": "gke",
"name": "providername",
"project_id": "redacted"
}
]

11.1.3 Retrieving Specific Provider for GKE

Command

curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/providers/<provider_uuid>/

Example


Response
11.4 Modifying Providers for GKE

Example

```bash
curl -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d \n'"project_id": "new_project_id"
' https://10.20.30.40/v3/providers/56de926b-daad-4382-b6e6-d0f67a2d13c8/
```

Response

```json
{
  "id": "eb13196c-2a72-457a-9c96-d2a38b94fec3",
  "type": "gke",
  "name": "providername",
  "project_id": "redacted"
}
```

11.5 Deleting Providers for GKE

Command

```bash
curl -k -X DELETE -H "x-auth-token: $TOKEN" $CCP/v3/providers/ <provider_uuid>/
```

Example

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

11.2 Administering v3 Clusters on GKE

11.2.1 Creating GKE clusters

Example

```bash
curl -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d \n'
"type": "gke",
"kubernetes_version": "latest",
"name": "cluster-name",
"provider": "283a325a-9665-439d-baf3-64c274bfc3d6",
"node_pools": [{
  "autoscaling": true,
  "autoscaling_min_nodes":2,
  "autoscaling_max_nodes":5,
```
"initial_node_count": 3,
"locations": ["us-west1"],
"image_type": "cos",
"preemptible": true,
"machine_type": "n1-standard-1",
"name": "nodepool1"
}
}
}' https://10.20.30.40/v3/clusters/

Response
{
"id": "87fec670-3794-4fc7-9e7e-ebc5367c47e4",
"type": "gke",
"name": "cluster-name",
"provider": "283a325a-9665-439d-baf3-64c274bcf3d6",
"status": "CREATING",
"kubeconfig": null,
"current_master_version": "",
"kubernetes_version": "latest",
"node_pools": [
{
"autoscaling": true,
"autoscaling_min_nodes": 2,
"autoscaling_max_nodes": 5,
"image_type": "cos",
"initial_node_count": 3,
"locations": [
"us-west1"
],
"machine_type": "n1-standard-1",
"name": "nodepool1",
"current_node_version": "",
"preemptible": true
}
],
"master_upgrade": false,
"worker_upgrade": false
}

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

11.2.2 Retrieving all clusters

Command

curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/v3/clusters/

Example


Response
11.2.3 Retrieving Specific GKE Clusters

Command

```
curl -k -X GET -H "x-auth-token: $TOKEN" $CCP/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/
```

Response

```
{
  "id": "87fec670-3794-4fc7-9e7e-ebc5367c47e4",
  "type": "gke",
  "name": "cluster-name",
  "provider": "283a325a-9665-439d-baf3-64c274bcf3d6",
  "status": "CREATING",
  "kubeconfig": null,
  "current_master_version": "",
  "kubernetes_version": "latest",
  "node_pools": [
    {
      "autoscaling": true,
      "autoscaling_min_nodes": 2,
      "autoscaling_max_nodes": 5,
      "image_type": "cos",
      "initial_node_count": 3,
      "locations": [
        "us-west1"
      ],
      "machine_type": "n1-standard-1",
      "name": "nodepool1",
      "current_node_version": "",
      "preemptible": true
    }
  ],
  "master_upgrade": false,
  "worker_upgrade": false
}
```
{  
  "autoscaling": true,  
  "autoscaling_min_nodes": 2,  
  "autoscaling_max_nodes": 5,  
  "image_type": "cos",  
  "initial_node_count": 3,  
  "locations": [  
    "us-west1"  
  ],  
  "machine_type": "n1-standard-1",  
  "name": "nodepool1",  
  "current_node_version": "",  
  "preemptible": true  
}

"master_upgrade": false,
"worker_upgrade": false

11.2.4 Modifying GKE clusters

Command

curl -X PATCH -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d "  
'  
{"kubernetes_version": "1.2.3"}  
' $CCP/v3/clusters/<cluster_uuid>/

Example

curl -X PATCH -H "content-type: application/json" -H "x-auth-token: $TOKEN" -d "  
'  
{"kubernetes_version": "1.2.3"}  
' https://10.20.99.190/v3/clusters/5a5f0db5-110c-4151-80e8-9b7889d30bc/

Response

[  
  {  
    "id": "87fec670-3794-4fc7-9e7e-ebc5367c47e4",  
    "type": "gke",  
    "name": "cluster-name",  
    "provider": "283a325a-9665-439d-baf3-64c274bcf3d6",  
    "status": "CREATING",  
    "kubeconfig": null,  
    "current_master_version": "",  
    "kubernetes_version": "latest",  
    "node_pools": [  
      {  
        "autoscaling": true,  
        "autoscaling_min_nodes": 2,  
        "autoscaling_max_nodes": 5,  
      }  
    ]  
  }  
]
"image_type": "cos",
"initial_node_count": 3,
"locations": [
  "us-west1"
],
"machine_type": "n1-standard-1",
"name": "nodepool1",
"current_node_version": "",
"preemptible": true
}
],
"master_upgrade": false,
"worker_upgrade": false
}
]

11.2.5 Deleting GKE clusters

Command

```bash
curl -k -X DELETE -H "x-auth-token: $TOKEN" $CCP/v3/clusters/cluster_uuid/
```

Example

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

For more information, see v3 openAPI documentation and v2 openAPI documentation.