Managing Kubernetes Clusters

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

This chapter contains the following topic:

- Setting up Kubernetes Dashboard, on page 1
- Configuring Node Pools, on page 1

Setting up Kubernetes Dashboard

Step 1  From the left pane, click Clusters.
Step 2  From the drop-down list displayed under the ACTIONS column, choose Download Token to get the Kubernetes configuration (Kubeconfig) file of the cluster that you want to access using the Kubernetes Dashboard.
Step 3  Use the Kubeconfig file from Step 2 to login to the Kubernetes Dashboard.

Configuring Node Pools

Node pools allow the creation of worker nodes with varying configurations. Nodes belonging to a single node pool have identical characteristics.

Cisco Container Platform supports node pools of multiple types, such as vSphere and nodepool. A single Cisco Container Platform deployment can have only one type of node pool.

In the Cisco Container Platform vSphere implementation, a node pool has the following properties:

- vcpus
- memory
- template
- labels
- taints
Labels and taints are optional parameters. All nodes that belong to a nodepool are tagged with labels and they are tainted. Taints are key-value pairs, which are associated with an effect.

The following table describes the available effects.

<table>
<thead>
<tr>
<th>Effect</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoSchedule</td>
<td>Ensures that the pods that do not contain this taint are not scheduled on the node.</td>
</tr>
<tr>
<td>PreferNoSchedule</td>
<td>Ensures that Kubernetes avoids scheduling pods that do not contain this taint on the node.</td>
</tr>
<tr>
<td>NoExecute</td>
<td>Ensures that a pod is removed from the node if it is already running on the node, and is not scheduled on the node if it is not yet running on the node.</td>
</tr>
</tbody>
</table>

Each cluster has a unique node pool. During cluster creation, each cluster is assigned a default node pool. Cisco Container Platform supports the ability for different master and worker configurations. To be able to support this, two new optional parameters, namely, master_node_pool and worker_node_pool are created as part of the cluster creation API. Upon cluster creation, the master node is created in the master-pool and the worker nodes are created in the default-pool.

**Note**

You can not delete a master-pool when there is only one master node.

---

**Customizing Master and Worker Configuration**

A cluster with different node configuration for master and worker can be created by specifying the attributes master_node_pool and worker_node_pool. In this case, you cannot specify a cluster level template, memory, and vcpus.

This section contains the following examples on customizing the master and worker node configuration:

- Specifying Attributes of master_node_pool and worker_node_pool, on page 2
- Cluster Create Request Example, on page 3
- Adding Custom Node Pool to an Existing Cluster, on page 3
- Deleting Node Pool, on page 4
- Editing Node Pool, on page 4

**Specifying Attributes of master_node_pool and worker_node_pool**

**Note**

Labels and taints are optional parameters and are valid for only a worker node pool.

```
"master_node_pool": { 
  "vcpus": 4, 
  "memory": 8196, 
```
"template": "ccp-tenant-image-1.10.1-a39424c.ova",
},
"worker_node_pool": {
"vcpus": 2,
"memory": 16384,
"template": "ccp-tenant-image-1.10.1-a39424c.ova",
"labels": "foo1=bar1,foo2=bar2"
	"taints": "key1=val1:NoSchedule,key2=val2:NoSchedule"
}
}

Cluster Create Request Example

curl -k -X POST
-<AUTH_TOKEN>
-Content-Type: application/json -d

{"deployer_type":"kubeadm",
"datastore":"hx1-data",
"name": "<NAME_OF_CLUSTER>",
"networks": [
  "VLAN 1154 = 10.10.96.0 - 22"
],
"workers": 2,
"master_vip": "10.10.96.143",
"ssh_user": "<SSH_USERNAME>",
"resource_pool": "hx1/Resources",
"description": "vip",
"deployer": {
  "provider": {
    "vsphere_datacenter": "Hyperflex",
    "vsphere_datastore": "hx1-data",
    "vsphere_client_config_uuid": "108b31fa-4b8c-4d9f-87f2-441b1b05921d",
    "vsphere_working_dir": "/Hyperflex/vm"
  },
  "provider_type": "vsphere"
},
"ssh_key": "<SSH_PUBLIC_KEY>",
"datacenter": "Hyperflex",
"cluster": "hx1",
"masters": 1,
"type": 1,
"kubernetes_version": "1.10.1",
"provider_client_config_uuid": "108b31fa-4b8c-4d9f-87f2-441b1b05921d",
"master_node_pool": {
  "vcpus": 4,
  "memory": 8196,
  "template": "ccp-tenant-image-1.10.1-a39424c.ova",
},
"worker_node_pool": {
  "vcpus": 2,
  "memory": 16384,
  "template": "ccp-tenant-image-1.10.1-a39424c.ova",
  "labels": "foo1=bar1,foo2=bar2",
  "taints": "key1=val1:NoSchedule,key2=val2:NoSchedule"
}
}

Adding Custom Node Pool to an Existing Cluster

curl -k -X POST
-<AUTH_TOKEN>
-Content-Type: application/json -d

{"template": "ccp-tenant-image-1.10.1-a39424c.ova",
,"worker_node_pool": {
"vcpus": 2,
"memory": 16384,
"template": "ccp-tenant-image-1.10.1-a39424c.ova",
"labels": "foo1=bar1,foo2=bar2",
"taints": "key1=val1:NoSchedule,key2=val2:NoSchedule"
}
}

https://ccp-api/2/clusters
Deleting Node Pool

curl -k -X DELETE \
-H "Content-Type: application/json" -d \ 
'{ "name": "small-pool-1", "vcpus": 1, "memory": 8196, "template": "ccp-tenant-image-1.10.1-a39424c.ova", "size": 2, "labels": "foo1=bar1,foo2=bar2", "taints": "key1=val1:NoSchedule,key2=val2:NoSchedule" }' \
https://ccp-api/2/clusters/fcd4a605-39a8-4e6e-9a0a-a1c7b1e?d9a7/nodepools

Editing Node Pool

curl -k -X PATCH \
-H "Content-Type: application/json" -d \ 
'{ "size": 20, "labels": "newFoo=newBar", "taints": "newKey1=newVal1:NoSchedule" }' \
https://ccp-api/2/clusters/fcd4a605-39a8-4e6e-9a0a-a1c7b1e?d9a7/nodepools

Suppose current NodePool is as follows:

"name": "small-pool-1", "vcpus": 1, "memory": 8196, "template": "ccp-tenant-image-1.10.1-a39424c.ova", "size": 2, "labels": "foo1=bar1,foo2=bar2", "taints": "key1=val1:NoSchedule,key2=val2:NoSchedule"

There are three cases in the editNodePool request:

- **Scale out operation:** When `sizeRequested` is greater than current NodePool size.

  Scale in operation: When `sizeRequested` is less than current nodePool size.

  In this case, the requested number of nodes are deleted. If labels and taints are present in the request, they are applied to remaining nodes. Previous labels and taints are removed.

- **When `sizeRequested` is equal to current nodePool size.**

  No change occurs if labels and taints are not provided. If labels and taints are provided, they are applied to the nodes after removing the previous labels and taints.