Configuring the Cisco HyperFlex CSI Integration for Kubernetes

Prerequisites

The following prerequisites must be met prior to configuring the Cisco HyperFlex CSI Integration.

On the HyperFlex cluster:

- Cisco HyperFlex cluster is installed and running HX 5.0(2a) or later.
- Configure an iSCSI network in HX Connect before installation. For more information about configuring an iSCSI network, see the Cisco HyperFlex Data Platform Administration Guide, Release 5.0.

On the Kubernetes cluster:

- Verify that all Kubernetes nodes have `2.0.874-5ubuntu2.10` version or later of the `open-iscsi` package installed ahead of proceeding with HXCSI. You can do so by running the following command: `$iscsid -version`.

  To install `open-iscsi version 2.0.874-5ubuntu2.10`, you can run the following command: `apt-get install -y open-iscsi=2.0.874-5ubuntu2.10`
- Verify that each Kubernetes node has either a dedicated interface on the HX iSCSI network or routable access to the HX iSCSI network.
- On each the Kubernetes node: Ensure that the file "`/etc/iscsi/initiatorname.iscsi`" exists. In this file, ensure that the value of "`InitiatorName`" is unique on EACH node e.g. `InitiatorName=iqn.1993-08.org.debian:01:6daab01fde1`

Note Sometimes, when VMs were cloned from a template, all nodes may have the same value of "InitiatorName" . This must be changed to ensure the uniqueness on each Kubernetes node.

- Ensure that all Kubernetes nodes contain the file "`/etc/iscsi/iscsid.conf`" with the following setting configured:

```
node.session.scan = manual

node.session.timeo.replacement_timeout = 120

node.conn[0].timeo.login_timeout = 120

node.conn[0].timeo.logout_timeout = 120

node.conn[0].timeo.noop_out_interval = 120

node.conn[0].timeo.noop_out_timeout = 120
```

  Restart the `iscsid` daemon after making this change.

- To ensure `iscsid` gets started on system reboot run the following command:

```
sudo systemctl enable iscsid
```

  The `iscsid` status should appear (as an example):

```
$ sudo systemctl status iscsid

iscsid.service - iSCSI initiator daemon (iscid)

Loaded: loaded (/lib/systemd/system/iscid.service; enabled; vendor preset: enabled
```

- Ensure that each Kubernetes primary (also known as the "master") host system contains the file "`/etc/kubernetes/manifests/kube-controller-manager.yaml`" which includes the following: `--disable-attach-detach-reconcile-sync=true`.
- Add the following text to the `-command` section of the file:
  `--disable-attach-detach-reconcile-sync=true`

Administrator Host

In this guide, the Administrator Host is used to refer to a Linux-based system that runs kubectl commands, etc. against the Kubernetes cluster. While this is typically a separate system (VM) that is not part of the Kubernetes cluster, you can use one of the Kubernetes nodes as the administrator host if you do not wish to install/manage a separate system (VM).

Installing the Cisco HyperFlex CSI Integration for Kubernetes

To install Cisco HyperFlex CSI Integration, complete the following procedures in the order presented:

Download the Cisco HyperFlex CSI Bundle

To download the Cisco HyperFlex CSI bundle (file) perform the following steps:

Step 1 Go to https://software.cisco.com

Step 2 Log in using your Cisco ID and credentials.

Step 3 In the **Download & Upgrade** section, click **Software Download**.

Step 4 In the **Select a Product** search field, type **`HyperFlex HX Data Platform`** and click **Enter**.

Step 5 Using the Release navigation pane on the left, select the HyperFlex Data Platform software version running on the cluster.

Cisco HyperFlex Data Platform Release 4.5(x) or later requires Cisco HyperFlex CSI integration.

Step 6 In the main navigation pane, locate and download the "Cisco HyperFlex Kubernetes Container Storage Interface (HX-CSI) bundle (tar.gz) file to your local machine.

Henceforth, the Cisco HyperFlex Kubernetes Container Storage Interface (HX-CSI) bundle (tar.gz) file shall be referred to as the "Cisco HyperFlex CSI bundle"

Step 7 On the administrator host, create a new directory called `hxcsi`.

Example:

```
administrator-host:~$ mkdir hxcsi
```

Step 8 Using secure copy (scp) or other preferred file transfer method, transfer (move or copy) the downloaded Cisco HyperFlex CSI bundle from your local machine to the "hxcsi" directory on the administrator host. The result should look like the following:

Example:

```
administrator-host:hxcsi$ ls

hxcsi-1.2.3a-659.tar.gz
```

What to do next

Open and Extract the Cisco HyperFlex CSI Bundle

Open and Extract the Cisco HyperFlex CSI Bundle

Perform the following steps to open the Cisco HyperFlex CSI bundle.:

Before you begin

Download the Cisco HyperFlex CSI bundle.

Use the `tar` command to unarchive the HyperFlex CSI bundle (.tar.gz file).

Example:

```
administrator-host:hxcsi$ tar -xf hxcsi-1.2.3a-659.tar.gz
```

Once completed, the following directory structure should exist:

- **examples (directory)** – includes some example YAML files for using the HXCSI integration
- **images (directory)** – includes HXCSI docker container image for the HXCSI integration. It also includes the base CSI images for the Provisioner, Attacher, Node-driver, Resizer and Snapshotter.

- **setup (directory)** – includes the setup script for deploying the HXCSI integration
- **support(directory)** – includes the script for collecting useful logs to help debugging.
- **`hxcsi-1.2.1.tgz` (file)**– this is the HELM chart package for this release of the HXCSI

Example

```
administrator-host:hxcsi$ ls -l
total 133196
-rw-r--r--  1 ubuntu ubuntu      6791 May 10 11:23 hxcsi-1.2.1.tgz
drwxr-xr-x  2 ubuntu ubuntu      4096 May 10 11:23 support
drwxr-xr-x  2 ubuntu ubuntu      4096 May 10 11:23 setup
drwxr-xr-x  2 ubuntu ubuntu      4096 May 10 11:23 images
drwxr-xr-x 11 ubuntu ubuntu      4096 May 10 11:23 examples
```

What to do next

Upload the Cisco HyperFlex CSI Container Image

Upload the Cisco HyperFlex CSI Container Image

The Cisco HyperFlex CSI integration components are deployed from a single container image provided in the “images” directory of the Cisco HyperFlex CSI bundle. The hxcsi container image leverages the other four base CSI images in the same directory. Before you can deploy the container image, move the container image to a location that is accessible to Docker running on the Kubernetes cluster worker nodes.

Manually Import the Cisco HyperFlex CSI Container Image Directly to each Kubernetes Worker Node

To add the Cisco HyperFlex CSI container image directly to each Kubernetes worker node, perform the following steps:

Before you begin

Open the Cisco HyperFlex CSI Bundle

Step 1 On the administrator host, copy the Cisco HyperFlex CSI container image (.tar) file, located in the “images” directory, to the /tmp directory on each Kubernetes worker node.

Example:

```
administrator-host:hxcsi$ scp ./images/hxcsi-1.2.3a-659.tar k8s-worker1:/tmp

administrator-host:hxcsi$ scp ./images/hxcsi-1.2.3a-659.tar k8s-worker2:/tmp

administrator-host:hxcsi$ scp ./images/hxcsi-1.2.3a-659.tar k8s-workerN:/tmp
```

Step 2 Copy the other base CSI container image files on each Kubernetes node:

```
csi-attacher-3.0.2-cisco1.tar, csi-node-driver-registrar-2.0.1-cisco1.tar, csi-resizer-1.0.1-cisco1.tar,
 csi-provisioner-2.0.4-cisco1.tar, csi-snapshotter-4.1.4-cisco1.tar,
snapshot-controller-4.1.1-cisco1.tar, snapshot-validation-webhook-4.1.1-cisco1.tar
```

Step 3 On each Kubernetes worker node, use the `docker load --input` command to load the Cisco HyperFlex CSI container image.

Example:

```
k8s-worker1:/tmp# docker load –input ./hxcsi-1.2.3a-659.tar
Loaded image: hxcsi:hxcsi-1.2.3a-659

k8s-worker2:/tmp# docker load –input ./hxcsi-1.2.3a-659.tar Loaded image: hxcsi:hxcsi-1.2.2a-626

k8s-workerN:/tmp# docker load –input ./hxcsi-1.2.3a-659.tar Loaded image: hxcsi:hxcsi-1.2.2a-626
```

Step 4 Docker load the other base CSI container image files on each Kubernetes node:

```
csi-attacher-3.2.1-cisco1.tar, csi-node-driver-registrar-2.2.0-cisco1.tar, csi-resizer-1.2.0-cisco1.tar, csi-provisioner-2.2.1-cisco1.tar,
csi-snapshotter-4.1.4-cisco1.tar, snapshot-controller-4.1.1-cisco1.tar,
snapshot-validation-webhook-4.1.1-cisco1.tar
```

What to do next

Install Cisco HyperFlex CSI.

Deployment HXCSI Using Helm Utility (Recommended)

In order to deploy the Cisco HyperFlex CSI integration using the Helm chart package, you must run the `helm` utility (version v3.5.2 or later). The following table describes the parameters that you can provide with the `helm` command.

Table 1: HELM Command Parameters

Parameter Name	Required or Optional	Description
`Hx.clientId string`	Optional	ClientID for the Tenant.
`Hx.iscsiUrl string`	Required	HyperFlex iSCSI cluster IP address of the eth-iscsi1:0 interface on the HyperFlex cluster.
`Hx.token string`	Required	Service Authentication Token. You must create the token out of band before invoking the helm install.
`Hx.url string`	Required	HyperFlex cluster management IP address of the eth0:mgmtip on the HyperFlex cluster. This IP is used for volume provisioning.
`hx.dockerRegistryName string`	Required	Docker registry name (e.g. mydockerhub.com/hx-docker)

Using service-token Utility

In order to deploy the Cisco HyperFlex CSI integration using the Helm chart package, you need to provide a service token. To do this, it is recommended that you run the `service-token` utility to generate the token

from the HyperFlex Data Platform. The following table describes the parameters that you can provide with the `service-token` command.

Table 2: Service Token Utility Parameters

Parameter Name	Required or Optional	Description
`clientId string`	Required	ClientID for the Tenant.
`mgmt-ip string`	Required	HyperFlex cluster management IP address. This is the IP of the eth0:mgmtip interface on the HyperFlex cluster.
`username string`	Required	user name to the HX cluster API (i.e., “admin”).
`password string`	Required	password to HX cluster API (if not specified in the command line, you will be prompted to enter it.)

As an example:

```
./setup/service-token -clientId myClient123 -mgmt-ip 10.2.17.13 -username admin
Password for [admin] at [10.2.17.13]:
Eyabc123457….

This token is to be passed to the helm command on AS-IS basis.

The token from the service-token command can be saved to an environment variable and then
pass that environment variable to the HELM command.

TOKEN=`./setup/service-token -clientId  myClient123 -mgmt-ip 10.2.17.13  -username admin `

helm install hxcsi hxcsi-1.2.1.tgz  --set hx.token=$TOKEN <OTHER parameters…>
```

Note Prior to running the following command, ensure that the IPs specified in the url and the iscsi-url parameters are reachable from the Kubernetes nodes.

The following example shows a Cisco HyperFlex CSI container image deployment that has been uploaded on the docker registry accessible from each Kubernetes node.

Example:

```
administrator-host:hxcsi$ helm install hxcsi hxcsi-1.2.1.tgz --set hx.url=10.2.17.13 \
--set hx.iscsiUrl=10.2.17.18 --set hx.clientId=myClientId \
--set hx.dockerRegistryName=mydockerhub.com/hx-docker --set hx.token=myToken123456
administrator-host:hxcsi$ helm status hxcsi
NAME: hxcsi
LAST DEPLOYED: Thu Oct 26 13:51:04 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
administrator-host:hxcsi$ kubectl get pods
NAME                                      READY    STATUS    RESTARTS     AGE
csi-attacher-hxcsi-0                       2/2     Running      0          9s
```

```
csi-attacher-hxcsi-1                          2/2     Running      0          9s
csi-nodeplugin-hxcsi-rm4h4                    2/2     Running      0          9s
csi-nodeplugin-hxcsi-w5fxt                    2/2     Running      0          9s
csi-provisioner-hxcsi-0                       2/2     Running      0          9s
csi-provisioner-hxcsi-1                       2/2     Running      0          9s
csi-resizer-hxcsi-0                           2/2     Running      0          9s
csi-resizer-hxcsi-1                           2/2     Running      0          9s
csi-snapshotter-hxcsi-0                       2/2     Running      0          9s
csi-snapshotter-hxcsi-1                       2/2     Running      0          9s
administrator-host:hxcsi$

administrator-host:hxcsi helm list
NAME   NAMESPACE  REVISION       UPDATED                  STATUS      CHART        APP VERSION
Hxcsi   default        1    2021-10-14 13:51:04 PDT.  deployed   hxcsi-1.2.1       1.2.2a
```

Example:

```
administrator-host:hxcsi$ kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
csi-attacher-hxcsi-0          2/2     Running   0          5d18h
csi-attacher-hxcsi-1          2/2     Running   0          5d18h
csi-nodeplugin-hxcsi-7qlw6    2/2     Running   0          5d18h
csi-nodeplugin-hxcsi-szrc4    2/2     Running   0          5d18h
csi-provisioner-hxcsi-0       2/2     Running   0          5d18h
csi-provisioner-hxcsi-1       2/2     Running   0          5d18h
csi-resizer-hxcsi-0           2/2     Running   0          5d18h
csi-resizer-hxcsi-1           2/2     Running   0          5d18h
csi-snapshotter-hxcsi-0       2/2     Running   0          5d18h
csi-snapshotter-hxcsi-1       2/2     Running   0          5d18h
```

What to do next

Create the Cisco HyperFlex CSI Storage Class.

Deploying HXCSI Using the hxcsi-setup Utility

In order to deploy the Cisco HyperFlex CSI integration, you must run the `hxcsi-setup` script. The `hxcsi-setup` script resides in the "setup" directory and automatically generates the necessary YAML files or Helm chart that then get applied (submitted) to the Kubernetes cluster to deploy the Cisco HyperFlex CSI components.

The following table describes parameters that you can provide with the `hxcsi-setup` command.

Table 3: hxcsi-setup Parameters

Parameter	Required or Optional	Description
`-clientId`	Optional	Client ID for the Tenant. **Note** You can create multiple Kubernetes clusters to request storage from the same HX cluster. The "clientId" parameter helps isolate the storage allocation for each of these clients/tenants.

Parameter	Required or Optional	Description
`-cluster-name`	Required	Provides a name to uniquely identify this specific Kubernetes cluster
`-helm-chart`	Optional	Generates helm chart for helm install (default will generate YAML files)
`-hx-csi-image string`	Required	Name and location of the Cisco HyperFlex CSI container image. This tells Kubernetes where the Cisco HyperFlex CSI container image should be pulled. [1]
`-iscsi-url string`	Required	HyperFlex iSCSI cluster IP address of the eth-iscsi1:0 interface on the HyperFlex cluster. For more information, see Prerequisites, on page 1.
`-output-dir string`	Optional	Output directory (default "./hxcsi-deploy/")
`-password string`	Required (if not entered initially, you will be prompted for it)	password to HX cluster API
`-token string`	Optional	Service Authentication Token. You can create the token out of band before invoking the hxcsi-setup.
`-url string`	Required	HyperFlex cluster management IP address. This IP is used for volume provisioning.
`-username string`	Required	user name to the HX cluster API (i.e., “admin”)
`-docker-registry`	Optional	Docker registry name (e.g. mydockerhub.com/hx-docker)

[1] If the Cisco HyperFlex CSI container image was imported directly into docker on each Kubernetes worker node, then the format for this parameter should be entered as **`<repository_name>:<tag>`**.

Before you begin

Upload the Cisco HyperFlex CSI Container Image and the base CSI container images.

On the administrator host, use the `hxcsi-setup` command in the "setup" directory to create the required Cisco HyperFlex CSI deployment files.

Note Prior to running the **hxcsi-setup** command, ensure that the IPs specified in the url and the iscsi-url parameters are reachable from the Kubernetes nodes.

Example:

When all images have been uploaded to a dockerhub: The following example shows when all images have been uploaded to a dockerhub that can be accessed from each of the Kubernetes nodes. The image name is `hxcsi` and the tag name is `hxcsi-1.2.3a-659`.

```
administrator-host:hxcsi$ ./setup/hxcsi-setup -cluster-name demo-hxcsi -clientId
 demo-client1 -hx-csi-image  hxcsi-1.2.3a-659 -iscsi-url 10.2.17.18 -url 10.2.17.13
-username admin -docker-registry dockerhub.cisco.com/hx-dev-docker

password for [admin] at [10.2.17.13]: *******
wrote config to hxcsi-deploy/hxcsi-config.yaml
wrote config to hxcsi-deploy/csi-attacher-hxcsi.yaml
wrote config to hxcsi-deploy/csi-nodeplugin-hxcsi.yaml
wrote config to hxcsi-deploy/csi-provisioner-hxcsi.yaml
wrote config to hxcsi-deploy/csi-attacher-rbac.yaml
wrote config to hxcsi-deploy/csi-nodeplugin-rbac.yaml
wrote config to hxcsi-deploy/csi-provisioner-rbac.yaml
wrote config to hxcsi-deploy/csi-resizer-rbac.yaml
wrote config to hxcsi-deploy/csi-resizer-hxcsi.yaml
```

Example:

When all images have been locally docker uploaded on each node: Thefollowing example shows a Cisco HyperFlex CSI container image deployment that has been uploaded on each node. The image name is `hxcsi` and the tag name is `hxcsi-1.2.3a-659`. This use case applies when there is no dockerhub used for pulling the images from a central location.

```
administrator-host:hxcsi$ ./setup/hxcsi-setup -cluster-name demo-hxcsi -clientId demo-client1
-hx-csi-image hxcsi:hxcsi-1.2.3a-659  -iscsi-url 10.2.17.18 -url 10.2.17.13 -username admin

password for [admin] at [10.2.17.13]: *******
wrote config to hxcsi-deploy/hxcsi-config.yaml
wrote config to hxcsi-deploy/csi-attacher-hxcsi.yaml
wrote config to hxcsi-deploy/csi-nodeplugin-hxcsi.yaml
wrote config to hxcsi-deploy/csi-provisioner-hxcsi.yaml
wrote config to hxcsi-deploy/csi-attacher-rbac.yaml
wrote config to hxcsi-deploy/csi-nodeplugin-rbac.yaml
wrote config to hxcsi-deploy/csi-provisioner-rbac.yaml
wrote config to hxcsi-deploy/csi-resizer-rbac.yaml
wrote config to hxcsi-deploy/csi-resizer-hxcsi.yaml
```

What to do next

Deploy the Cisco HyperFlex CSI Components

Deploying HXCSI Using Cisco HyperFlex CSI Components

After running the hxcsi-setup script and generating the Cisco HyperFlex CSI deployment files, a new “hxcsi-deploy” directory is created on the administrator host.

Before you begin

Create the Cisco HyperFlex CSI Deployment files.

Step 1 On the administrator host, use the `kubectl create -f` command to deploy the Cisco HyperFlex CSI components.

Example:

```
administrator-host:hxcsi$ kubectl create -f ./hxcsi-deploy/
service/csi-attacher-hxcsi created
statefulset.apps/csi-attacher-hxcsi created
serviceaccount/csi-attacher created
clusterrole.rbac.authorization.k8s.io/external-attacher-runner created
clusterrolebinding.rbac.authorization.k8s.io/csi-attacher-role created
daemonset.apps/csi-nodeplugin-hxcsi created
serviceaccount/csi-nodeplugin created
clusterrole.rbac.authorization.k8s.io/csi-nodeplugin created
clusterrolebinding.rbac.authorization.k8s.io/csi-nodeplugin created
service/csi-provisioner-hxcsi created
statefulset.apps/csi-provisioner-hxcsi created
serviceaccount/csi-provisioner created
clusterrole.rbac.authorization.k8s.io/external-provisioner-runner created
clusterrolebinding.rbac.authorization.k8s.io/csi-provisioner-role created
deployment.apps/csi-resizer-hxcsi created
serviceaccount/csi-resizer created
clusterrole.rbac.authorization.k8s.io/external-resizer-runner created
clusterrolebinding.rbac.authorization.k8s.io/csi-resizer-role created
role.rbac.authorization.k8s.io/external-resizer-cfg created
rolebinding.rbac.authorization.k8s.io/csi-resizer-role-cfg created
secret/hxcsitoken created
configmap/hxcsi-config created
```

Step 2 On the administrator host, use the `kubectl get pods` command to verify the HXCSI components have been deployed and have a status of `Running`.

Example:

```
administrator-host:hxcsi$ kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
csi-attacher-hxcsi-0          2/2     Running   0          5d18h
csi-attacher-hxcsi-1          2/2     Running   0          5d18h
csi-nodeplugin-hxcsi-7qlw6    2/2     Running   0          5d18h
csi-nodeplugin-hxcsi-szrc4    2/2     Running   0          5d18h
csi-provisioner-hxcsi-0       2/2     Running   0          5d18h
csi-provisioner-hxcsi-1       2/2     Running   0          5d18h
csi-resizer-hxcsi-0           2/2     Running   0          5d18h
csi-resizer-hxcsi-1           2/2     Running   0          5d18h
csi-snapshotter-hxcsi-0       2/2     Running   0          5d18h
csi-snapshotter-hxcsi-1       2/2     Running   0          5d18h
```

What to do next

Create Cisco HyperFlex CSI Storage Class.

HXCSI Sample Pods

The HXCSI package includes several examples to create Pods.

Table 4: HXCSI Package Samples

#	Directory Name	Description
1	`sample-hxcsi`	Pod running `nginx`. Basic example creates a PVC named ‘`hxpvclaim-default`’

#	Directory Name	Description
2	`sample-hxcsi-csi-clone`	Pod that creates a clone from the '`sample-hxcsi`', showing '`dataSource`' as '`hxpvclaim-default`'
3	`sample-hxcsi-ds`	Pod using named datastore '`test-ds`'
4	`sample-hxcsi-fs`	Pod using default datastore and file system type '`xfs`'
5	`sample-hxcsi-no-ds`	Pod using default datastore and default file system.
6	`sample-hxcsi-no-ds-clone`	Clone from the sample '`sample-hxcsi-no-ds`' where datastore name was specified.
7	`sample-resize-block`	Resize the block volume, uses attribute - `allowVolumeExpansion: true`
8	`sample-resize-fs`	Resize the default file-system '`ext4`', uses `allowVolumeExpansion: true`
9	`sample-resize-clone`	Clone from the `hxpvclaim-default-resize` '`sample-resize-fs`' example Pod.
10	`sample-se`	Creates a software-encrypted volume. A storage class with the attribute `datastoreEncryption: true` must be used for such volumes.
11	`sample-hxcsi-snapshot`	Creates a snapshot from volume (and volume from snapshot).
12	`sample-chap`	Enables CHAP protection.

Note To resize a volume:

1. Change the size of the volume in the PVC's yaml file.
2. Run the `kubectl apply -f <pvc.yaml>` command to apply the new size setting.

Create Cisco HyperFlex CSI Storage Class

Once the components are up and running, you must need to now create a Storage Class that allows developers to consume storage through the Cisco HyperFlex CSI integration.

Before you begin

Deploy the Cisco HyperFlex CSI Components

Step 1 On the administrator host, create a file named "`hxcsi-storage-class.yaml`" with the following contents:

Example:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
    name: csi-hxcsi-default
provisioner: csi-hxcsi
parameters:
    datastore: default-ds
    datastoreSize:"20000000000000”
```

You can specify the datastore name and size in the parameter section as above. You can optionally choose to make this the default Storage Class, which means that the Cisco HyperFlex CSI storage integration will be used by default for any Persistent Volume Claims that do not otherwise specify any other Storage Class to use. If you choose to make the Cisco HyperFlex CSI Storage Class the default storage class, then your "`hxcsi-storage-class.yaml`" file should contain the following contents:

Example:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
    name: csi-hxcsi-default
    annotations:
        storageclass.kubernetes.io/is-default-class: "true"
provisioner: csi-hxcsi
parameters:
```

Note If a datastore does not already exist, a new datastore will be created. If you do not specify a datastore name, a default datastore with the name "`iscsiDs`" is created.

Note Always create a datastore that is larger than the volumes to be created.

Step 2 On the administrator host, use the `kubectl create -f` command to create the Cisco HyperFlex CSI Storage Class.

Example:

```
root@administrator-host:hxcsi$ kubectl create -f ./hxcsi-storage-class.yaml

storageclass.storage.k8s.io/csi-hxcsi-default created
```

Sample Storage Class for Resize Volume

Example:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
    name: csi-hxcsi-default-resize
provisioner: csi-hxcsi
parameters:
          datastore: default-ds
          datastoreSize:"20000000000000"
allowVolumeExpansion: true
```

Note that resizing the volume only allows volumes that are provisioned against this storageclass to support resizing. To change the actual size of the volume, you will need to edit the PVC specifications to change to the new size, for example, edit the PVC YAML file, and run `kubectl apply -f <pvc-yaml>`

Sample Storage Class for File System

Example:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
    name: csi-hxcsi-default-fs
```

```
provisioner: csi-hxcsi
parameters:
    fsType: xfs
```

Note: Default file system is "`ext4`"

What to do next

Verify Cisco HyperFlex CSI Storage Class Creation.

Verifying Cisco HyperFlex CSI Storage Class Creation

To verify the storage class creation perform the following step:

Note If setting the Cisco HyperFlex CSI Storage Class as the default, verify that "(default)" is present next to the Storage Class name.

Before you begin

Create Cisco HyperFlex CSI Storage Class.

On the administrator host, use the `kubectl get sc` command to verify the Cisco HyperFlex CSI Storage Class was created.

Example:

```
root@administrator-host:hxcsi$ kubectl get sc
NAME    PROVISIONER  AGE
csi-hxcsi (default) csi-hxcsi  67s
```