Cisco HyperFlex Systems Administration Guide for Kubernetes, Release 4.0

First Published: 2019-04-29

Last Modified: 2021-07-01

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

Communications, Services, Bias-free Language, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business impact you’re looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco Marketplace.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

Documentation Feedback

To provide feedback about Cisco technical documentation, use the feedback form available in the right pane of every online document.

Cisco Bug Search Tool

Cisco Bug Search Tool (BST) is a web-based tool that acts as a gateway to the Cisco bug tracking system that maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. BST provides you with detailed defect information about your products and software.

Bias-Free Language

The documentation set for this product strives to use bias-free language. For purposes of this documentation set, bias-free is defined as language that does not imply discrimination based on age, disability, gender, racial identity, ethnic identity, sexual orientation, socioeconomic status, and intersectionality. Exceptions may be present in the documentation due to language that is hardcoded in the user interfaces of the product software, language used based on standards documentation, or language that is used by a referenced third-party product.

CHAPTER 1

New and Changed

New and Changed Information

This table summarizes the new and changed features for the Cisco HyperFlex Systems Administration Guide for Kubernetes, Release 4.0(x) and where they are documented.

Feature	Description	Date/Release Added	Where Documented
Cisco HyperFlex CSI Interoperability Metrics	Added support for CCP, and Anthos Versions.	December, 16 2020	Cisco HyperFlex CSI Interoperability Metrics, on page 3
Cisco HyperFlex CSI Interoperability Metrics	Added support for Kubernetes Version 1.16, 1.17 and 1.18	November, 4 2020	Cisco HyperFlex CSI Interoperability Metrics, on page 3
Troubleshooting: Enable Kubernetes Operation Fails on Clusters Deployed By Cisco Intersight.	Added a new troubleshooting topic	April 8, 2020	Troubleshooting, on page 23
Cisco HyperFlex CSI interoperatility Metrics	Updated platform versions.	February 4, 2020	Cisco HyperFlex CSI Interoperability Metrics, on page 3
Troubleshooting	Added the Troubleshooting section.	September 12, 2019	Troubleshooting, on page 23
Cisco HyperFlex Systems Administration Guide for Kubernetes, Release 4.0(x)	This guide was introduced.	HX 4.0	N/A

CHAPTER 2

Cisco HyperFlex Kubernetes Support

Support Overview

There are two main components that are important to consider when determining support for a Kubernetes version or distribution on Cisco HyperFlex.

- Support for the Kubernetes version or distribution with Cisco HyperFlex.
- Support for the Cisco HyperFlex Container Storage Interface (CSI) storage integration with the specific Kubernetes version or distribution.

Note Kubernetes storage special interest group (K8 SIG community) is not supported on Stretch Clusters.

While in general, Cisco HyperFlex supports any version or distribution of Kubernetes, there is a specific sub-set of versions and distributions that have been tested and are recommended with the Cisco HyperFlex CSI storage integration for Kubernetes. Additionally while it is possible to run Kubernetes and container-based workloads on Cisco HyperFlex without using the HyperFlex CSI storage integration, we strongly recommend that you leverage the native capability when running any stateful Kubernetes-based applications and services that require persistent storage.

The Cisco HyperFlex Kubernetes CSI Integration allows Cisco HyperFlex to dynamically provide persistent storage to stateful Kubernetes workloads running on Cisco HyperFlex. The integration enables orchestration of the entire Persistent Volume object lifecycle to be offloaded and managed by Cisco HyperFlex, while being driven (initiated) by developers and users through standard Kubernetes Persistent Volume Claim objects. Developers and users get the benefit of leveraging Cisco HyperFlex for their Kubernetes persistent storage needs with zero additional administration overhead from their perspective.

Cisco HyperFlex CSI Interoperability Metrics

Cisco HyperFlex CSI and Kubernetes Platform Version and Distribution Interoperability:

Hyperflex Data Platform Version	CSI Spec Version	Kubernetes Version	Cisco Qualified CCP Version	Cisco Qualified Anthos Version
4.0(1a)	1.0	1.14	5.0, 5.1	1.1, 1.2, 1.3
4.0(2a)	1.0	1.15	5.1, 5.2	1.3
4.0(2b)	1.0	1.16	6.0, 7.0	1.4.1
4.0(2c)	1.0	1.17	6.0, 7.0	1.5.1

CHAPTER 3

Cisco HyperFlex Container Storage Interface (CSI) for Kubernetes

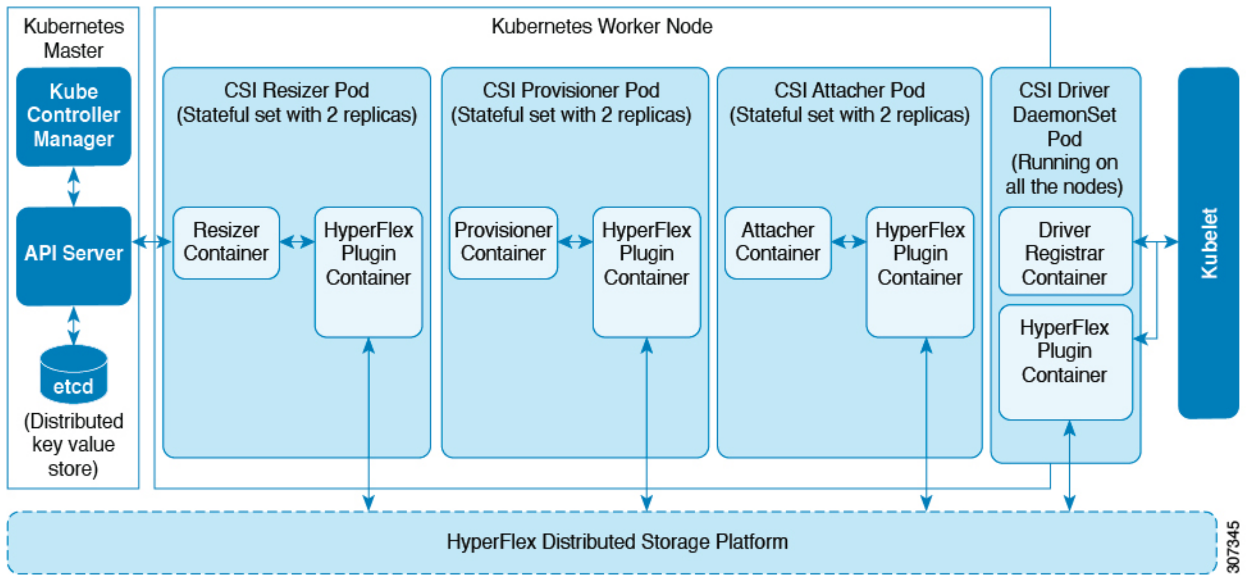
About Cisco Hyperflex Kubernetes CSI

Cisco HyperFlex Container Storage Interface (CSI) is an out-of-tree container-based Kubernetes storage integration; which is deployed and consumed through standard Kubernetes primitives such as Persistent Volume Claims and Storage Classes. Cisco HyperFlex CSI supports the following features:

- Dynamic creation and deletion of volumes
- Dynamic volume attach and detach

Cisco HyperFlex CSI Components

The Cisco HyperFlex CSI integration is deployed as containers on top of the target Kubernetes cluster. The following diagram shows the different components of the Cisco HyperFlex CSI deployment and how they interact with each other.


Kubernetes Master
Kube Controller Manager
API Server
etcd
(Distributed key value store)
Kubernetes Worker Node
CSI Resizer Pod (Stateful set with 2 replicas)
Resizer Container
HyperFlex Plugin Container
CSI Provisioner Pod (Stateful set with 2 replicas)
Provisioner Container
HyperFlex Plugin Container
CSI Attacher Pod (Stateful set with 2 replicas)
Attacher Container
HyperFlex Plugin Container
CSI Driver DaemonSet Pod (Running on all the nodes)
Driver Registrar Container
HyperFlex Plugin Container
Kubelet
HyperFlex Distributed Storage Platform
307345

Deployment includes three pods:

csi-attacher-hxcsi

- **Type:** StatefulSet
- **Number of Instances:** One per Kubernetes Cluster
- **Purpose:** Required by CSI, but not currently in used in the Cisco deployment.

csi-provisioner-hxcsi

- **Type:** StatefulSet
- **Number of Instances:** One per Kubernetes Cluster
- **Purpose:** Watches Kubernetes Persistent Volume Claim objects and triggers CreateVolume and DeleteVolume operations as part of CSI spec.

csi-nodeplugin-hxcsi

- **Type:** DaemonSet
- **Number of Instances:** One per Kubernetes Worker Node
- **Purpose:** Discovery and formatting of provisioned HyperFlex iSCSI LUNs on Kubernetes worker nodes. Implements NodeStage/NodeUnstage and NodePublish/NodeUnpublish Volume APIs as part of Kubernetes CSI spec.

CHAPTER 4

Configuring the Cisco HyperFlex CSI Integration for Kubernetes

Prerequisites

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

- Verify that you are using the Kubernetes Admin Guide for the installed version of HyperFlex. See the Cisco HyperFlex CSI Interoperability Metrics, on page 3
- This Cisco HyperFlex Systems Administration Guide applies only to HXDP Release 4.0(x).
- Kubernetes worker nodes have the Linux `iscsi-initiator-utils` package installed.
- If running RHEL7 / CentOS7 or later versions, ensure the `iscsi_tcp` kernel module is loaded into the kernel at boot on all Kubernetes worker nodes by running the following command:

```
echo iscsi_tcp >> /etc/modules-load.d/iscsi.conf
```

Administrator Host

In this chapter, the Administrator Host is simply a linux-based system that is used to administer run 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).

Enabling Kubernetes Integration in Cisco HyperFlex Connect

Perform the following steps to enable Kubernetes support in Cisco HyperFlex Connect:

Warning **HyperFlex clusters deployed using Cisco Intersight:** Before enabling Kubernetes, review and perform the required steps in the Troubleshooting chapter.

Before you begin

Enable Kubernetes Integration in HX Connect HX Release 4.0(x) during a maintenance window. Enabling the feature on a running cluster may impact storage IO operations.

Procedure

Step 1 Navigate to the Cisco HyperFlex cluster by using a supported web browser (for example, https://<hyperflex_cluster_management_IP_address).

Step 2 Log in to Cisco HyperFlex Connect using a VMware SSO account and password with administrative privileges (that is, administrator@vsphere.local).

Step 3 In the upper right-hand corner of Cisco HyperFlex Connect, click the **Settings** menu icon (represented by a Gear icon).

Step 4 From the drop-down list under **Integrations**, click **Kubernetes**.

Step 5 On the **Enable Persistent Volumes for Kubernetes** page, click **Enable All Nodes** to configure the Cisco HyperFlex cluster to support Persistent Volumes for Kubernetes. The default value for a new cluster is **Disabled**.

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:

Procedure

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 4.0 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-1.0.rel.4.0.410.git.04f91684.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.

Procedure

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

Example:

```
administrator-host:hxcsi$ tar -xf ./hxcsi-1.0.rel.4.0.410.git.04f91684.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
- **setup (directory)** – includes the setup script for deploying the HXCSI integration

Example

```
administrator-host:hxcsi$ ls -l
total 632308
drwxr-xr-x 4 root root      4096 Apr 22 18:19 examples
-rw-r--r-- 1 root root 647464960 Apr 23 19:21 hxcsi-1.0.rel.4.0.410.git.04f91684.tar
```

```
drwxr-xr-x 2 root root      4096 Apr 23 19:22 images
drwxr-xr-x 2 root root      4096 Apr 22 18:19 setup
```

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. Before the container image can be deployed, move the container image to a location that is accessible to Docker running on the Kubernetes cluster worker nodes.

There are two options for making the Cisco HyperFlex CSI container image available to the Kubernetes cluster nodes.

The first option involves transferring the Cisco HyperFlex CSI container image directly to each Kubernetes worker node and manually importing the container image into Docker. For detailed steps, see (Option 1) Manually Import the Cisco HyperFlex CSI Container Image Directly to each Kubernetes Worker Node, on page 10

The second option involves pushing the Cisco HyperFlex CSI container image to a private (local) image registry (repository) from which Docker running on each Kubernetes worker node can pull the image as necessary. For detailed steps, see (Option 2) Push the Cisco HyperFlex CSI Container Image to a Locally Available Docker Image Registry, on page 11.

(Option 1) 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

Procedure

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-1.0.rel.4.0.410.git.04f91684.tar k8s-worker1:/tmp

administrator-host:hxcsi$ scp ./images/hxcsi-1.0.rel.4.0.410.git.04f91684.tar k8s-worker2:/tmp

administrator-host:hxcsi$ scp ./images/hxcsi-1.0.rel.4.0.410.git.04f91684.tar k8s-workerN:/tmp
```

Step 2 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-1.0.rel.4.0.410.git.04f91684.tar

Loaded image: hxcsi-1.0.rel.4.0.410.git.04f91684

k8s-worker2:/tmp# docker load --input \                    >
./hxcsi-1.0.rel.4.0.410.git.04f91684.tar

Loaded image: hxcsi-1.0.rel.4.0.410.git.04f91684

k8s-workerN:/tmp# docker load --input \                    >
./hxcsi-1.0.rel.4.0.410.git.04f91684.tar

Loaded image: hxcsi-1.0.rel.4.0.410.git.04f91684
```

What to do next

Create the Cisco HyperFlex CSI Deployment (YAML) Files.

(Option 2) Push the Cisco HyperFlex CSI Container Image to a Locally Available Docker Image Registry

To push the Cisco HyperFlex CSI container image to a locally available container registry, perform the following steps:

Before you begin

Open the Cisco HyperFlex CSI bundle.

Procedure

Step 1 On the administrator host, use the `docker load --input` command to load the Cisco HyperFlex CSI container image from the “images” directory.

Example:

```
administrator-host:hxcsi$ docker load --input ./images/hxcsi-1.0.rel.4.0.410.git.04f91684.tar
762d8e1a6054: Loading layer [==================================================>]
91.39MB/91.39MB
e45cfbc98a50: Loading layer [==================================================>]
15.87kB/15.87kB
d60e01b37e74: Loading layer [==================================================>]
12.29kB/12.29kB
b57c79f4a9f3: Loading layer [==================================================>]
3.072kB/3.072kB
e2211c2d4feb: Loading layer [==================================================>]
19.95MB/19.95MB
a1fab2a5ef3e: Loading layer [==================================================>]
83.58MB/83.58MB
0d4ed16e2508: Loading layer [==================================================>]
1.64MB/1.64MB
41a9c1d6e07b: Loading layer [==================================================>]
```

```
25.92MB/25.92MB
15013e506922: Loading layer [==================================================>]
288.3kB/288.3kB
5a5e90f3a363: Loading layer [==================================================>]
161.8kB/161.8kB
d78b9b4a4846: Loading layer [==================================================>]
3.108MB/3.108MB
bceb0e273837: Loading layer [==================================================>]
413.3MB/413.3MB
Loaded image: hxcsi:1.0.rel.4.0.410.git.04f91684
```

Step 2 On the administrator host, run the docker images command to verify the Cisco HyperFlex CSI container image was successfully loaded.

Example:

```
administrator-host:hxcsi$ docker images
REPOSITORY   TAG                            IMAGE ID       CREATED         SIZE
hxcsi        1.0.rel.4.0.410.git.04f91684   419dd2559236   25 hours ago    628MB
```

Step 3 On the administrator host, use the `docker tag` command to create an additional tag representing the location of the private docker image repository.

Example:

Note In the following example, “k8s-repo” is the hostname of the server where the private docker image repository resides and the repository is running on port “5000”.

```
administrator-host:hxcsi$ docker tag hxcsi:1.0.rel.4.0.410.git.04f91684 \
k8s-repo:5000/hxcsi:1.0.rel.4.0.410.git.04f91684
```

Step 4 On the administrator host, verify the new tag is present using the `docker images` command.

Example:

```
administrator-host:hxcsi$ docker images
REPOSITORY            TAG                            IMAGE ID           CREATED
   SIZE
hxcsi                 1.0.rel.4.0.410.git.04f91684   419dd2559236       25 hours ago
   628MB
k8s-repo:5000/hxcsi   1.0.rel.4.0.410.git.04f91684   419dd2559236       25 hours ago
   628MB
```

Step 5 On the administrator host, use the `docker push` command to push the docker image to your desired locally available private docker image repository

Example:

```
administrator-host:hxcsi$ docker push k8s-repo:5000/hxcsi:1.0.rel.4.0.410.git.04f91684
The push refers to repository [k8s-repo:5000/hxcsi]
bceb0e273837: Pushed
d78b9b4a4846: Pushed
5a5e90f3a363: Pushed
15013e506922: Pushed
41a9c1d6e07b: Pushed
0d4ed16e2508: Pushed
a1fab2a5ef3e: Pushed
e2211c2d4feb: Pushed
b57c79f4a9f3: Pushed
d60e01b37e74: Pushed
e45cfbc98a50: Pushed
762d8e1a6054: Pushed
1.0.rel.4.0.410.git.04f91684: digest:
sha256:eb2db100139af40a3376b69b8c586fe912f38a261ff1420bd0860ef96775a278 size: 2838
```

Step 6 Once the Cisco HyperFlex CSI container image has been successfully pushed to the private docker image repository, you can now optionally delete the local docker image(s) on the administrator host using the `docker rmi` command.

Example:

```
administrator-host:hxcsi$ docker rmi k8s-repo:5000/hxcsi:1.0.rel.4.0.410.git.04f91684
Untagged: k8s-repo:5000/hxcsi:1.0.rel.4.0.410.git.04f91684
Untagged:
k8s-repo:5000/hxcsi@sha256:eb2db100139af40a3376b69b8c586fe912f38a261ff1420bd0860ef96775a278
Deleted: sha256:419dd2559236351af75b5de6d56885bb56d76b0cdbf85671076dfb0c19d2a284
Deleted: sha256:a8d324a4f852c6e002789347f3ddde02291af2d73cfbb1248795f2fe4274a338
Deleted: sha256:20dfa716f53c0995260f1ddaf1057e4db97dc1753bd223cf8b7794db41f3da4f
Deleted: sha256:ddebbafb4cdd12f033700ff0c182bef31a0e6df46b33895d422aab25196e40d1
Deleted: sha256:ad18c85d65210fe1e6daadf9e3bc506bbbda68b7392929088eb15a2156932c0f
Deleted: sha256:752ce3c6d3784571e219a9611bc8c70632247159e628e8440f17677e9ea02494
Deleted: sha256:51a1e93b695559102121fb43937600f959d29a79a175f0ff1c1831040132645d
Deleted: sha256:617d766bfbeff3e8a496f22887a20b92e878df65f0352c38b07159eebe1d4d22
Deleted: sha256:6c7c6f124a4f90e7ada71d26b278d6e81da42e6a4cffb109f7f6df952f411f0b
Deleted: sha256:e783d8ee44ce099d51cbe699f699a04e43c9af445d85d8576f0172ba92e4e16c
Deleted: sha256:cc7fae10c2d465c5e4b95167987eaa53ae01a13df6894493efc5b28b95c1bba2
Deleted: sha256:99fc3504db138523ca958c0c1887dd5e8b59f8104fbd6fd4eed485c3e25d2446
Deleted: sha256:762d8e1a60542b83df67c13ec0d75517e5104dee84d8aa7fe5401113f89854d9
```

What to do next

Generate the Cisco HyperFlex CSI Deployment (YAML) Files

Generate the Cisco HyperFlex CSI Deployment (YAML) Files

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 that then get applied (submitted) to the Kubernetes cluster to deploy the Cisco HyperFlex CSI components.

The following parameters must be provided with the `hxcsi-setup` command:

- `-cluster-name:` provide a name to uniquely identify this specific Kubernetes cluster
- `-hx-csi-image:` name and location of the Cisco HyperFlex CSI container image. This tells Kubernetes where the Cisco HyperFlex CSI container image should be pulled from.

Note 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 **`<image_name>:<tag>`**. If the Cisco HyperFlex CSI container image was pushed to a container image registry, then the format for this parameter should be entered as **`<registry_IP_or_hostname>/<image_name>:<tag>`**

- `-iscsi-url:` HyperFlex cluster management IP address
- `-url:` HyperFlex cluster management IP address
- `-username:` “admin”

Before you begin

Upload the Cisco HyperFlex CSI Container Image.

Procedure

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

Example:

The following example shows a Cisco HyperFlex CSI container image that has been pushed to a local container image repository with a hostname of “k8s-repo”, running on port “5000”. The image name is “hxcsi” and the tag name is “1.0.rel.4.0.410.git.04f91684”.

```
administrator-host:hxcsi$ ./setup/hxcsi-setup -cluster-name demo-hxcsi -hx-csi-image \
       k8s-repo:5000/hxcsi:1.0.rel.4.0.410.git.04f91684 -iscsi-url 10.2.17.13 -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
```

Note Users who chose to import the Cisco HyperFlex CSI container image directly to each Kubernetes worker node.

In versions of the Cisco HyperFlex CSI integration “hxcsi-setup” script prior to Cisco HyperFlex Data Platform 4.0.1b (fixed in 4.0.1b), it may be necessary to update the image pull policy (imagePullPolicy) values in some of the automatically generated YAML files. For more information, see Troubleshooting.

What to do next

Deploy the Cisco HyperFlex CSI Components

Deploy the Cisco HyperFlex CSI Components

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

```
root@administrator-host:hxcsi$ ls
examples  hxcsi-1.0.rel.4.0.410.git.04f91684.tar  hxcsi-deploy  images  setup
```

Before you begin

Create the Cisco HyperFlex CSI Deployment (YAML) Files.

Procedure

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
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:

Note There should be one instance of the "csi-attacher-hxcsi" pod, one instance of the "csi-provisioner-hxcsi" pod, and then instance of the "csi-nodeplugin-hxcsi" pod for each Kubernetes worker node. Meaning if you have a total of two Kubernetes worker nodes, you should see two instances of the "csi-nodeplugin-hxcsi" pod as shown in the following example:

```
administrator-host:hxcsi$ kubectl get pods

NAME                         READY   STATUS    RESTARTS   AGE
csi-attacher-hxcsi-0         2/2     Running   0          87s
csi-nodeplugin-hxcsi-9fgsf   2/2     Running   0          87s
csi-nodeplugin-hxcsi-qqvwj   2/2     Running   0          87s
csi-provisioner-hxcsi-0      2/2     Running   0          87s
```

What to do next

Create Cisco HyperFlex CSI Storage Class.

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

Procedure

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:
```

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:
```

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
```

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.

Procedure

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
```

CHAPTER 5

Deploying Stateful Applications with Cisco HyperFlex CSI

Prerequisites for Deploying Stateful Applications with Cisco HyperFlex CSI

The following prerequisites must be met prior to deploying stateful applications using the HyperFlex CSI storage integration.

- Cisco HyperFlex cluster is installed and running 4.0(x) and 4.5(1a) .
- Kubernetes support must be enabled in HyperFlex Connect.
- The Cisco HyperFlex CSI integration has been deployed.

Administrator Host

In this chapter, the Administrator Host is simply a linux-based system that is used to administer run 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).

Deploying Stateful Applications

To deploy stateful applications, perform the following procedures:

Creating a Persistent Volume Claim

A Persistent Volume Claim is a simply a request for storage by a user. Users specify their storage requirements, the size or capacity of the storage required, and other options. Depending on the associated Storage Class, the storage requirements are routed to the appropriate provisioner which knows how to provision the requested storage, and make it available to Kubernetes

Procedure

Step 1 On the administrator host, create a file named "message-board-pvc.yaml" with the following contents

Example:

```
administrator-host:hxcsi$ cat ./message-board-pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: message-board-pvc
spec:
  storageClassName: csi-hxcsi-default
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

Step 2 On the administrator host, use the `kubectl create -f` command to create the Persistent Volume Claim.

Example:

```
administrator-host:hxcsi$ kubectl create -f ./message-board-pvc.yaml

persistentvolumeclaim/message-board-pvc created
```

Step 3 On the administrator host, use the `kubectl get pvc` command to verify the Persistent Volume Claim was created and is successfully bound to a Persistent Volume.

Example:

```
administrator-host:hxcsi$ kubectl get pvc

NAME              STATUS VOLUME                                   CAPACITY ACCESS MODES
STORAGECLASS AGE
message-board-pvc BOUND  pvc-8069462e-662c-11e9-a163-005056a086d9 10Gi     RWO
csi-hxcsi-default  20s
```

Deploy Stateful Kubernetes Workload

Kubernetes workloads come in various forms, such as Pods and Deployments regardless of the type of Kubernetes workload, each can leverage persistent storage using the Cisco HyperFlex CSI integration and Persistent Volume Claims. The following shows the deployment of a sample open source application called Cisco Message Board that can be used to test the Cisco HyperFlex CSI integration. You can also test with your own applications following the same methodology and procedures.

Procedure

Step 1 On the administrator host, create the YAML file which defines the workload to be deployed.

Example:

The following shows the YAML file for the example Cisco Message Board application which will create both a Kubernetes Deployment and a Kubernetes Service which will allow for connecting to the deployed Cisco Message Board application through a NodePort.

Note That we are referencing the Persistent Volume Claim name in the "volumes" section of the Kubernetes Deployment definition. In this example, the Persistent Volume bound to the "message-board-pvc" Persistent Volume Claim will be mounted inside the "message_board:version1" container at the "/sqldb" location (path)

```
administrator-host:hxcsi$ cat ./message-board-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
    name: message-board
    labels:
        app: message-board
spec:
    replicas: 1
    selector:
        matchLabels:
            app: message-board
    template:
        metadata:
            labels:
                app: message-board
                name: message-board
        spec:
            volumes:
                - name: demovolume1
                  persistentVolumeClaim:
                    claimName: message-board-pvc
            containers:
            - name: message-board
              image: michzimm/message_board:version1
              ports:
              - containerPort: 5000
              volumeMounts:
                  - mountPath: "/sqldb"
                    name: demovolume1
---
apiVersion: v1
kind: Service
metadata:
  name: message-board
  labels:
    name: message-board
  namespace: default
spec:
  type: NodePort
  ports:
  - port: 5000
    nodePort: 30002
  selector:
    name: message-board
```

Step 2 On the administrator host, use the `kubectl create -f` command to create the Deployment and Service.

Example:

```
administrator-host:hxcsi$ kubectl create -f ./message-board-deployment.yaml
deployment.apps/message-board created
service/message-board created
```

Step 3 On the administrator host, use the `kubectl get pods` command to check the status of the deployed Pods.

Example:

```
administrator-host:hxcsi$ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
csi-attacher-hxcsi-0              2/2     Running   0          3h51m
csi-nodeplugin-hxcsi-9fgsf        2/2     Running   0          3h51m
csi-nodeplugin-hxcsi-qqvwj        2/2     Running   0          3h51m
csi-provisioner-hxcsi-0           2/2     Running   0          3h51m
message-board-6df65d6b59-49xhq    1/1     Running   0          95s
```

Step 4 On the administrator host, use the `kubectl get services` command to check the status of the deployed Service.

Example:

```
root@administrator-host:hxcsi$ kubectl get services
NAME                    TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
csi-attacher-hxcsi      ClusterIP   10.98.79.159     <none>        12346/TCP        3h53m
csi-provisioner-hxcsi   ClusterIP   10.99.73.185     <none>        12345/TCP        3h53m
kubernetes              ClusterIP   10.96.0.1        <none>        443/TCP          4h24m
message-board           NodePort    10.107.227.152   <none>        5000:30002/TCP   2m59s
```

For the sample Cisco Message Board application, the service is configured using “NodePort” and port “30002” meaning the application should be a up and running and accessible by pointing your web browser to any Kubernetes node IP address and port “30002”. For example: http://<k8s-worker1>:30002

CHAPTER 6

Troubleshooting

Troubleshooting

The following section highlights common issues seen when installing and using the HyperFlex CSI integration. The information provided includes symptoms to help diagnose the issue as well as a solution to resolve the issue.

HyperFlex CSI Node Plugin (“csi-nodeplugin…”) Pods Fail to Start

- **Symptom 1:** HyperFlex CSI components have been installed and the “csi-attacher…” and “csi-provisioner…” pods are running however the “csi-nodeplugin…” pods on each node fail to start.
- **Symptom 2:** Running the command “kubectl describe pod <csi-nodeplugin_pod_name>” shows a message containing the following error: “MountVolume.SetUp failed for volume “iscsi-dir” : hostPath type check failed: /etc/iscsi is not a directory”

Solution:

Ensure that the “iscsi-initiator-utils” package has been installed on each of the Kubernetes worker nodes. The HyperFlex CSI integration uses the software iSCSI initiator within the guest operating system to connect to the persistent volume storage objects via iSCSI. The “iscsi-intiator-utils” package is required for this operation. Perform the following steps:

1. Remove the deployed HyperFlex CSI components using the “kubectl delete -f ./hxcsi-deploy” command.
2. Install the “iscsi-initiator-utils” package on each Kubernetes worker node. Depending on the guest operation system, the command will vary. As an example, on Ubuntu the command would be “apt-get install iscsi-initiator-utils”.

3. Re-apply the HyperFlex CSI deployment YAML files to the Kubernetes cluster using the “kubectl create -f ./hxcsi-deploy” command

Stateful Applications Stuck in Container Creating Stage

- **Symptom 1:** After deploying a statful Kubernetes workload using HX CSI, the pods as part of that workload remain in the ContainerCreating stage indefinitely.
- **Symptom 2:** Running the command “kubectl describe pod <pod_name>” shows a message container the following error: “rpc error: code = Unkown desc = unable to find matching device for volume id”
- **Symptom 3:** Your Kubernetes nodes (VMs) are running RHEL7 or CentOS7 or later guest operating system.

Solution:

In versions of RHEL7 and CentOS7 (or later), changes to SELINUX cause the “iscsi_tcp” kernel module to be loaded when called, rather than at boot. This causes issues when using the HyperFlex CSI integration. Ensure the “iscsi_tcp” kernel module is loaded at boot.

1. On each Kubernetes worker node, run the following command “echo iscsi_tcp >> /etc/modules-load.d/iscsi.conf”.

ImagePullBackOff Status Errors when Deploying HX CSI Pods

- **Symptom 1:** Running the command “kubectl get pods [-n <namespace>]” shows that the HX CSI pods are showing a status of “ImagePullBackOff”.
- **Symptom 2:** Running the command “kubectl describe pod <csi-pod_name>” shows a message containing the following error: “Error: ErrImaePull” and “Back-off pulling image…”

Solution:

- **Solution 1:** Ensure the HX CSI container image name provided to the hxcsi-setup script is correct
- **Solution 2:** Ensure the HX CSI container image exists, either directly within docker on each Kubernetes worker node or on the local container image registry depending on which deployment option was chosen.
- **Solution 3:** Ensure the “imagePullPolicy” lines in the following YAML files generated by the hxcsi-setup script are set to “IfNotPresent”.
    - csi-attacher-hxcsi.yaml
    - csi-nodeplugin-hxcsi.yaml
    - csi-provisioner-hxcsi.yaml

Enable Kubernetes Operation Fails on Clusters Deployed By Cisco Intersight

- **Symptom:** The enable Kubernetes operation hangs on the “Volume Access” stage or does not complete when run on HyperFlex clusters that were initially deployed by Cisco Intersight.

Solution:

There are two solutions for this symptom; determined by whether you have run **`Enable Kubernetes`** or not.

The solution for users who have already run **`Enable Kubernetes`**:

- Solution 1 = Required
- Solution 2 = Required

The solution for users who have not run **`Enable Kubernetes`**:

- Solution 1 = Optional
- Solution 2 = Required

Solution 1:

Run the following command on all ESX hosts:

```
esxcfg-vmknic -d -p k8-priv-iscsi
esxcli network vswitch standard portgroup remove -p k8-priv-iscsi -v k8-iscsi
esxcli network vswitch standard portgroup remove -p k8-priv-iscsivm-network -v k8-iscsi
esxcli network vswitch standard remove -v k8-iscsi
```

Solution 2:

This solution is required for all users. Run the following commands on all HX Controller VMs:

```
sed -i -e "s/255.255.0.0/255.255.255.0/g"
/opt/springpath/storfs-mgmt/stMgr-1.0/conf/application.conf

sed -i -e "s/255.255.0.0/255.255.255.0/g" \
       -e "s/169.254.1./169.254.254./g"   \
       -e "s/except_vnic:$/except_vnic.device:/g"
/usr/share/springpath/storfs-misc/hx-scripts/iscsiVolumeAccessCheck.py

sed -i -e "s/255.255.0.0/255.255.255.0/g" \
       -e "s/169.254.1./169.254.254./g"   \
       -e "s/except_vnic:$/except_vnic.device:/g"
/usr/share/springpath/storfs-misc/hx-scripts/iscsiVolumeAccessEnable.py

restart hxSvcMgr
restart stMgr
```