Configuring HyperFlex FlexVolume Storage Integration for Cisco Container Platform

Support Matrix for HX FlexVolume Integration with OCP

The following table summarizes the Red Hat OpenShift Container Platform (OCP) software versions that are supported with each of the HX Data Platform software versions.

Table 1: Support Matrix for HX FlexVolume Integration with Red Hat OCP

HX Data Platform Version	Red Hat OCP Version 3.7	Red Hat OCP Version 3.9	Red Hat OCP Version 3.10	Red Hat OCP Version 3.11	Red Hat OCP Version 3.12	Red Hat OCP Version 3.13
3.0(1a) or later	Not Supported	Not Supported	—	—	—	—
3.5(1a) or later	—	Supported	Supported	—	—	—
3.5(2a) or later	—	Planned	Planned	—	—	—
4.0(1a) or later	—	—	Planned	Planned	—	—
4.1(1a) or later	—	—	Planned	Planned	TBD	TBD

Prerequisites

The following prerequisites must be met prior to configuring HyperFlex FlexVolume Storage Integration for Cisco Container Platform.

- Cisco HyperFlex cluster is installed and running 3.5(x).
- Cisco Container Platform Control Plane is installed and running 2.0 or later.

Creating Cisco Container Platform Tenant Cluster

During the CCP tenant cluster creation workflow within the CCP control plane, you must set the **HyperFlex Local Network** option in order to install and configure the HyperFlex FlexVolume Storage Integration for Kubernetes. Once the option is selected, CCP will automatically install both the HyperFlex FlexVolume Plugin and HyperFlex FlexVolume Provisioner as part of the tenant cluster deployment.

The following section is an abbreviated description of the CCP “Create Cluster” workflow which highlights the **HyperFlex Local Network** option to install the HyperFlex FlexVolume Storage Integration for Kubernetes. For more details on creating a new CCP tenant cluster, please refer to the CCP documentation.

Step 1 Log into the CCP control plane UI.

Step 2 On the **Clusters** page, click **New Cluster**.

Step 3 On the **Basic Information** page, enter the appropriate information and click **Next**.

Step 4 On the **Provider Settings** page, enter the appropriate information and ensure the **HyperFlex Local Network** option is set to `k8-priv-iscsvm-network` in order to tell CCP to install and configure the HyperFlex FlexVolume plug-in.

Step 5 Click **Next**.

Step 6 On the **Summary** page, view the cluster information and click **Submit**.

The CCP control plane will then deploy the requested CCP tenant cluster, including the installation and configuration of all required components of the HyperFlex FlexVolume Storage Integration for Kubernetes.

Managing HyperFlex FlexVolume Plug-in

Installing HyperFlex FlexVolume Plug-in

If the **HyperFlex Local Network** option is configured properly when deploying the CCP tenant cluster, the CCP control plane automatically installs and configures the HyperFlex FlexVolume plug-in on the CCP tenant cluster. No additional configuration is required.

Checking HyperFlex FlexVolume Plug-in Version

Important The following steps must be performed on one CCP tenant cluster node only.

Step 1 Log into one of the tenant Kubernetes cluster nodes using SSH. Use the username that corresponds to the SSH key provided during the **New Cluster** workflow when provisioning the CCP tenant cluster through CCP.

Step 2 Run the following command to change directories to the `/usr/libexec/kubernetes/kubelet-plugins/volume/exec/hyperflex-hxvolume/` directory.

```
cd /usr/libexec/kubernetes/kubelet-plugins/volume/exec/hyperflex~hxvolume/
```

Example:

```
ccpuser@tc1-mastercf1ff968f8:~$ cd
/usr/libexec/kubernetes/kubelet-plugins/volume/exec/hyperflex~hxvolume/$

ccpuser@tc1-mastercf1ff968f8:/usr/libexec/kubernetes/kubelet-plugins/volume/exec/hyperflex~hxvolume$
```

Step 3 Run the `hxvolume version` command as the root user (with `sudo`) to view the HyperFlex FlexVolume plug-in version.

```
sudo hxvolume version
```

Example:

```
ccpuser@tc1-mastercf1ff968f8:/usr/libexec/kubernetes/kubelet-plugins/volume/exec/hyperflex~hxvolume$
sudo ./hxvolume version

hxvolume version: 1.0.284.git.4022e8e

ccpuser@tc1-mastercf1ff968f8:/usr/libexec/kubernetes/kubelet-plugins/volume/exec/hyperflex~hxvolume
```

Upgrading HyperFlex FlexVolume Plug-in

Upgrading of the HyperFlex FlexVolume Plug-in is performed as part of the CCP tenant cluster upgrade process. Follow the steps that are provided in the CCP documentation to upgrade the CCP tenant cluster. This process includes upgrading the HyperFlex FlexVolume Plug-in to the latest available version.

Modifying Configuration for Kubernetes VMs

Complete the following task when existing HyperFlex clusters use IP addresses in the range, 169.154.1.0 to 169.154.1.24 for ESXi. After a Kubernetes cluster operation, such as scale up or upgrade, this procedure must be repeated on the ALL new VMs.

After an upgrade to HXDP release 3.5(2a), run the following command on each Kubernetes VM. This command will find the parameter "targetIp" in the `hxflexvolume.json` file, and replace the value from "169.254.254.1" to "169.254.254.1".

Note The `<ssh user>` must match the SSH user that was specified during cluster creation.

The `<private key file>` must correspond to the public key that was specified during cluster creation.

Managing HyperFlex FlexVolume Provisioner

Installing HyperFlex FlexVolume Provisioner

If the **HyperFlex Local Network** option is configured properly when deploying the CCP tenant cluster, the CCP control plane automatically installs and configures the HyperFlex FlexVolume Provisioner on the CCP tenant cluster. No additional configuration is required.

Checking HyperFlex FlexVolume Provisioner Version

Step 1 Run the `kubectl get pods -n kube-system` command to get the complete name of the deployed HyperFlex FlexVolume Provisioner pod.

```
kubectl get pods -n kube-system
```

Example:

```
ccpuser@admin-host:~$ kubectl get pods -n kube-system
NAME                                           READY     STATUS    RESTARTS   AGE
calico-node-6mc7b                              2/2       Running   0          7d
calico-node-tjks9                              2/2       Running   0          7d
calico-node-z4png                              2/2       Running   0          7d
calico-typha-7d48f84746-crrb2                  1/1       Running   0          7d
calico-typha-7d48f84746-vt6gm                  1/1       Running   0          7d
etcd-tc1-mastercf1ff968f8                      1/1       Running   0          7d
hx-provisioner-f98479996-k79v6                 1/1       Running   0          6d
kube-apiserver-tc1-mastercf1ff968f8            1/1       Running   0          7d
kube-controller-manager-tc1-mastercf1ff968f8   1/1       Running   0          7d
kube-dns-6c74cdd686-k877b                      3/3       Running   0          7d
kube-proxy-8s6j6                               1/1       Running   0          7d
kube-proxy-f2d2z                               1/1       Running   0          7d
kube-proxy-vfqjz                               1/1       Running   0          7d
kube-scheduler-tc1-mastercf1ff968f8            1/1       Running   0          7d
tiller-deploy-5c567bd778-7xr6d                 1/1       Running   0          7d
ccpuser@admin-host:~$
```

Step 2 Run the `kubectl describe pods...` command to get the complete details of the deployed HyperFlex FlexVolume Provisioner pod. Look for the `hx-provisioner` container image name which includes the version as a tag (that is, after the colon in the container name).

```
kubectl describe pods <pod_name> -n kube-system
```

Example:

```
ccpuser@admin-host:~$ kubectl describe pods hx-provisioner-f98479996-k79v6 -n kube-system
Name:           hx-provisioner-f98479996-k79v6
Namespace:      kube-system
```

```
Node:           tc1-worker87d761f2d0/172.0.13.116
Start Time:     Fri, 14 Sep 2018 21:23:41 -0400
Labels:         app=hx-provisioner
                pod-template-hash=954035552
Annotations:    cni.projectcalico.org/podIP=192.168.2.11/32
Status:         Running
IP:             192.168.2.11
Controlled By:  ReplicaSet/hx-provisioner-f98479996
Containers:
  hx-provisioner:
    Container ID:  docker://f5cc3d45480a7a706264b965cd71ee7af47680393101d507ce36826e4e4b384f
    Image:         hx-provisioner:0.10.274.git.365b059e
    Image ID:      docker://sha256:0184783ed8cd143b786ab77654a9a1ec693c6c005adb22a988f39e0538e1b822
    Port:          443/TCP
    Host Port:     0/TCP
    Args:
      -hxapi-url=$(HX_API_URL)
      -hxapi-token-file=/secrets/hxapi/token
      -hxapi-hxclusteruuid=$(HX_CLUSTERUUID)
    State:          Running


ccpuser@admin-host:~$
```

Upgrading HyperFlex FlexVolume Provisioner

Upgrading of the HyperFlex FlexVolume Provisioner is performed as part of the CCP tenant cluster upgrade. Follow the steps that are provided in CCP documentation to upgrade the CCP tenant cluster. This process includes upgrading the HyperFlex FlexVolume Provisioner to the latest available version.

Configuring Storage Classes

If the **HyperFlex Local Network** option was set properly when deploying the CCP tenant cluster, the CCP control plane automatically creates a StorageClass for HyperFlex on the CCP tenant cluster. By default, the HyperFlex StorageClass is not set as the default StorageClass in the CCP tenant cluster. In this case, by default, developers must explicitly specify HyperFlex as the StorageClass in Persistent Volume Claims to use the HyperFlex FlexVolume storage integration.

Use the `kubectl get sc` command to view the StorageClasses on the CCP tenant cluster.

```
ccpuser@admin-host:~$ kubectl get sc
NAME                 PROVISIONER                     AGE
hyperflex            hyperflex.io/hxvolume           4s
standard (default)   kubernetes.io/vsphere-volume   4s
ccpuser@admin-host:~$
```

Provisioning Persistent Volumes

Step 1 Run the following commands to create a Persistent Volume Claim YAML file that includes a user- defined **`persistent_volume_claim_name`** and **`size`**.

The `storageClassName: hyperflex` line is required in order to send the storage requests to the HyperFlex FlexVolume Provisioner.

```
vi <path>/<filename>.yaml
<insert the following>
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: <persistent_volume_claim_name>
spec:
  storageClassName: hyperflex
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <size>Gi
```

Example:

```
ccpuser@admin-host:~$ vi ~/hxkube/pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: message-board-pvc
spec:
  storageClassName: hyperflex
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
ccpuser@admin-host:~$
```

Step 2 Run the `kubectl create` command to submit the pvc.yaml file and create the Persistent Volume Claim object in the CCP tenant Kubernetes cluster. In parallel, as part of the operation, HyperFlex creates a Persistent Volume object to complement the Persistent Volume Claim object and bind the two together in Kubernetes.

```
kubectl create -f ~/hxkube/<pvc_name>.yaml
```

Example:

```
ccpuser@admin-host:~$ kubectl create -f ~/hxkube/pvc.yaml
persistentvolumeclaim/message-board-pvc created
ccpuser@admin-host:~$
```

Step 3 Check the status of the Persistent Volume Claim object with the `kubectl get pvc` command to make sure it was created successfully and is bound to a Persistent Volume object.

```
kubectl get pvc
```

Example:

```
ccpuser@admin-host:~$ kubectl get pvc
NAME                STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
message-board-pvc   Bound
```

Step 4 Deploy Kubernetes Pod using the `kubectl create` command while specifying the Persistent Volume Claim object in the Pod YAML file.

```
kubectl create -f <pod_yaml_file>
```

Example:

```
apiVersion: v1
kind: Pod
metadata:
    name: message-board
    labels:
        app: message-board
        name: message-board
    namespace: default
spec:
    containers:
    - name: message-board
      image: michzimm/message_board:version1
      volumeMounts:
      - name: demovolume1
        mountPath: /sqldb
      ports:
      - containerPort: 5000
    volumes:
    - name: demovolume1
      persistentVolumeClaim:
```

Example:

```
ccpuser@admin-host:~$ kubectl create -f ./message-board.yaml
pod/message-board created
ccpuser@admin-host:~$
```

Step 5 Check the status of the deployed Pod to ensure it is running.

```
kubectl get pods
```

Example:

```
ccpuser@admin-host:~$ kubectl get pods
NAME            READY     STATUS    RESTARTS   AGE
message-board   1/1       Running   0          35m
ccpuser@admin-host:~$
```