

CSI Driver for Dell EMC PowerFlex

Product Guide

1.2

Notes, cautions, and warnings

 **NOTE:** A NOTE indicates important information that helps you make better use of your product.

 **CAUTION:** A CAUTION indicates either potential damage to hardware or loss of data and tells you how to avoid the problem.

 **WARNING:** A WARNING indicates a potential for property damage, personal injury, or death.

Introduction

This chapter contains the following section:

Topics:

- [Product overview](#)


Product overview

The CSI Driver for Dell EMC PowerFlex is a plug-in that is installed into Kubernetes to provide persistent storage using Dell EMC PowerFlex storage system.

The CSI Driver for Dell EMC PowerFlex and Kubernetes communicate using the Container Storage Interface protocol. CSI Driver for Dell EMC PowerFlex conforms to CSI specification 1.1 and compatible with Kubernetes versions 1.17, 1.18 and 1.19, OpenShift 4.3 and 4.4 with Red Hat Enterprise Linux 7.6 worker nodes and Docker EE 3.1. The CSI Driver for Dell EMC PowerFlex is validated against the Kubernetes CSI Driver Tests version 3.1.0.

Features of CSI Driver for Dell EMC PowerFlex

The CSI Driver for Dell EMC PowerFlex supports the following features:

- Persistent volume (PV) capabilities:
 - Create
 - Delete
 - Create from Snapshot
 - Resize
 - Dynamic and static PV provisioning
 - Snapshot capabilities - create, delete, and list (Kubernetes Only, not supported on Openshift 4.3).
 - Volume mount as *ext4* or *xfs* file system on the worker node.
 - Supports the following access modes:
 - single-node-writer
 - single-node-reader-only
 - multi-node-reader-only
 - multi-node-multi-writer (block only)
 - Volume prefix for easy LUN identification
 - Supports HELM 3 charts installer
 - Supports Dell EMC Storage CSI Operator deployment
 - Supports Red Hat Enterprise Linux (RHEL) 7.6, 7.7, and 7.8 as host operating system.
 - Supports CentOS 7.6, 7.7, and 7.8 as host operating system.
 - Supports Dell EMC PowerFlex 3.0.x and 3.5 versions.
 - Supports CSI 1.1
 - Supports Kubernetes version 1.17, 1.18, and 1.19
 - Supports OpenShift 4.3 and 4.4 with Red Hat Enterprise Linux 7.6 worker nodes
 - Supports Docker EE 3.1
 - Supports Raw Block Volumes
 - Support (Online) Volume Expansion
 - Supports Topology
 - Compatible with PowerFlex 3.0/3.5 with medium and fine granularity storage pools
 - Supports mount options, specified in storageclass yaml under *mountOptions*.
-  **NOTE:** Before utilizing mount options, you must first be fully aware of the potential impact and understand your environment's requirements for the specified option.

Installing CSI Driver for Dell EMC PowerFlex

This chapter contains the following sections:

Topics:

- [Installation overview](#)
- [Prerequisites](#)

Installation overview

The CSI Driver for Dell EMC PowerFlex can be deployed in Kubernetes platforms using HELM version 3 charts or the Dell EMC Storage CSI Operator. The CSI Driver for Dell EMC PowerFlex can be deployed on Openshift platforms using the Dell EMC Storage CSI Operator. The CSI Driver repository includes HELM charts that use a shell script to deploy the CSI Driver for Dell EMC PowerFlex. The shell script installs the CSI Driver container image along with the required Kubernetes sidecar containers.

If using a Helm installation, the controller section of the Helm chart installs the following components in a *Stateful Set* in the namespace *vxflexos*:

- CSI Driver for Dell EMC PowerFlex
- Kubernetes Provisioner, which provisions the volumes
- Kubernetes Attacher, which attaches the volumes to the containers
- Kubernetes Snapshotter, which provides snapshot support
- Kubernetes Resizer, which resizes the volume

The node section of the Helm chart installs the following component in a *Daemon Set* in the namespace *vxflexos*:

- CSI Driver for Dell EMC PowerFlex
- Kubernetes Registrar, which handles the driver registration

Prerequisites

Before you install CSI Driver for Dell EMC PowerFlex, verify the requirements that are mentioned in this topic are installed and configured.


Requirements

- Install Kubernetes. The CSI Driver for Dell EMC PowerFlex works with Kubernetes version 1.17, 1.18 and 1.19
OR
- Install OpenShift. The CSI Driver for Dell EMC PowerFlex works with OpenShift 4.3 and 4.4 with Red Hat Enterprise Linux (RHEL) 7.6 worker nodes
- For HELM 3 based install (Kubernetes only): [Install Helm package manager](#).
- For Operator based install (Kubernetes and Openshift): [Install using Operator](#).
- A user must exist on the array with a role `>= FrontEndConfigure`
- Verify that zero padding is enabled on the PowerFlex storage pools that must be used. Use PowerFlex GUI in the PowerFlex CLI to check this setting. See Dell EMC PowerFlex documentation for more information to configure this setting.
- [Configure Docker service \(Kubernetes only\)](#)
- [Install VxFlex OS SDC](#)

Configure Docker service (Kubernetes only)

The mount propagation in Docker must be configured on all Kubernetes nodes before installing CSI Driver for Dell EMC PowerFlex.

Prerequisites

 **NOTE:** Configure docker service does not apply to Openshift based installations.

Steps

1. Edit the service section of `/etc/systemd/system/multi-user.target.wants/docker.service` file as follows:

```
docker.service
[Service]
...
MountFlags=shared
```

2. Restart the docker service with `systemctl daemon-reload` and `systemctl restart docker` on all the nodes.

Install PowerFlex Storage Data Client

Use the procedure in this topic to install PowerFlex Storage Data Client.

About this task

Install the PowerFlex Storage Data Client (SDC) on all Kubernetes nodes or with OpenShift on the RHEL worker nodes.

For detailed PowerFlex installation procedure, see the *Dell EMC PowerFlex Deployment Guide*. Install the PowerFlex SDC as follows:


Steps

1. Download the PowerFlex SDC from [Dell EMC Online support](#). The filename is `EMC-ScaleIO-sdc-*.rpm`, where `*` is the SDC name corresponding to the PowerFlex installation version.
2. Export the shell variable `MDM_IP` in a comma-separated list. This list contains the IP addresses of the MDMs.
`export MDM_IP=xx.xxx.xx.xx,xx.xxx.xx.xx`, where `xxx` represents the actual IP address in your environment.
3. Install the SDC using the following commands:
 - For Red Hat Enterprise Linux and Cent OS, run `rpm -iv ./EMC-ScaleIO-sdc-*.x86_64.rpm`, where `*` is the SDC name corresponding to the PowerFlex installation version.

Install CSI Driver for Dell EMC PowerFlex using HELM


Procedure to install CSI Driver for Dell EMC PowerFlex using HELM.

Prerequisites

 **NOTE:** The CSI Driver for Dell EMC PowerFlex version 1.1.5 and later support HELM 3 only. HELM 3 is easier to install than previous versions and poses less security risks, because no Tiller installation or special privileges are required. See, [Install HELM 3](#) for instructions to install HELM 3.

Ensure that you meet the following prerequisites before you install the CSI Driver for Dell EMC PowerFlex:

- You have installed the SDCs on the worker nodes.
- You have the driver's Helm chart from the source repository at <https://github.com/dell/csi-vxflexos>, ready for this procedure.
- The directory **dell-csi-helm-installer** contains the new scripts - `csi-install.sh` and `csi-uninstall.sh` scripts.
- You have created a Kubernetes secret with the name - `vxflexos-creds` using the username/password of your VxFlex instance.
- Mount propagation is configured for Docker on all nodes.

 **NOTE:** If your Kubernetes cluster does not already have the CRD for snapshot support, see [Installing the Volume Snapshot CRDs](#) before installing the driver.

Steps

1. Clone the git repository to the master node of the Kubernetes cluster:

```
git clone https://github.com/dell/csi-vxflexos
```

2. Create the namespace where the driver will be installed.

```
kubectl create namespace vxflexos
```

3. Edit the `helm/secret.yaml` and replace the values for the username and password parameters. These values can be obtained using base64 encoding as described in the following example:

```
echo -n "myusername" | base64
echo -n "mypassword" | base64
```


Where *myusername* and *mypassword* are credentials for a user with PowerFlex privileges.

4. Create the credentials secret:

```
kubectl create -f secret.yaml
```

5. Collect information from the PowerFlex SDC (Storage Data Client) by executing the `get_vxflexos_info.sh` script located in the top-level helm directory.

This script shows the *VxFlex OS system ID* and *MDM IP* addresses. Make a note of the value for these parameters as they must be entered in the *myvalues.yaml* file.


 **NOTE:** Your SDC might have multiple VxFlex OS systems registered. Ensure that you choose the correct values.

6. Copy the default values.yaml file

```
cd helm && cp .csi-vxflexos/values.yaml myvalues.yaml
```

7. Edit the newly created file and provide values for the following parameters

```
vi myvalues.yaml
```

- Set the *systemName* string variable to the VxFlex OS system name or system ID. This value was obtained by running the `get_vxflexos_info.sh` script in Step 1 of this procedure.
- Set the *restGateway* string variable to the URL of your system's REST API Gateway. You can obtain this value from the VxFlex OS administrator.
- Set the *storagePool* string variable to a default (already existing) storage pool name in your VxFlex OS system.
 **NOTE:** New storage pools can be created in VxFlex OS UI and CLI utilities.
- Set the *mdmIP* string variable to a comma separated list of MDM IP addresses.
- Set the *volumeNamePrefix* string variable so that volumes created by the driver have a default prefix. If one VxFlex OS system is servicing several different Kubernetes installations or users, these prefixes help you distinguish them.
- The *controllerCount* variable is used by advanced users to deploy multiple controller instances. The specified default value **1** is designed to work as expected. You can modify the value of this variable to set the desired number of CSI controller replicas.
- Set the *enablelistvolumesnapshot* variable **false** unless instructed otherwise, by Dell EMC support. It causes snapshots to be included in the CSI operation ListVolumes.
- The Helm charts create a Kubernetes *StorageClass* while deploying CSI Driver for Dell EMC VxFlex OS. The *StorageClass* section includes following variables:
 - The *name* string variable defines the name of the Kubernetes storage class that the Helm charts will create. For example, the *vxflexos* base name will be used to generate names such as *vxflexos* and *vxflexos-xfs*.

- The *isDefault* variable (valid values for this variable are `true` or `false`) will set the newly created storage class as default for Kubernetes.

NOTE:

- Set this value to **true** only if you expect VxFlex OS to be your principle storage provider, as it will be used in *PersistentVolumeClaims* where no *storageclass* is provided. After installation, you can add custom storage classes if desired.
- All strings must be contained within double quotes.

- The *reclaimPolicy* string variable defines whether the volumes will be retained or deleted when the assigned pod is destroyed. The valid values for this variable are `Retain` or `Delete`.

8. Install the driver using sh script.

For example, if you are installing the driver in the namespace `vxflexos`, then run the following command:

```
cd ../dell-csi-helm-installer && ./csi-install.sh --namespace vxflexos --values ../helm/myvalues.yaml
```

NOTE: For detailed instructions on how to run the script, refer to the readme document in the `dell-csi-helm-installer` folder.

This script also runs the `verify.sh` script that is present in the same directory. You will be prompted to enter the credentials for each of the Kubernetes nodes. The `verify.sh` script needs the credentials to check if SDC has been configured on all nodes. You can also skip the verification step by specifying the `--skip-verify-node` option.

Install using Operator

Starting from version 1.1.4, CSI Driver for Dell EMC VxFlex OS can also be installed using the new Dell EMC Storage CSI Operator.

The Dell EMC Storage CSI Operator is a Kubernetes Operator, which can be used to install and manage the CSI Drivers that are provided by Dell EMC for various storage platforms.

This operator is available as a community operator for upstream Kubernetes and can be deployed using [OperatorHub.io](https://operatorhub.io). It is also available as a community operator for OpenShift clusters and can be deployed using OpenShift Container Platform. Both these methods of installation use OLM (Operator Lifecycle Manager).

The operator can also be deployed directly by following the instructions available on [GitHub](https://github.com).

Instructions on how to deploy the CSI Driver for Dell EMC VxFlex OS using the operator is available on [GitHub](https://github.com). There are sample manifests provided which can be edited to do an easy installation of the driver.

NOTE: The deployment of the driver using the operator does not use any Helm charts. The installation and configuration parameters are slightly different from the ones that are specified by the Helm installer.

Kubernetes Operators make it easy to deploy and manage entire lifecycle of complex Kubernetes applications. Operators use Custom Resource Definitions (CRD) which represents the application and use custom controllers to manage them.

Update CSI Driver for Dell EMC PowerFlex

You can upgrade the CSI Driver for Dell EMC PowerFlex using Helm or `dell-csi-operator`.

Steps

1. Upgrade using Helm:

A direct upgrade of the driver from an older version to version 1.2 is not supported because of breaking changes in Kubernetes APIs in the migration from alpha snapshots to beta snapshots.

In order to upgrade the driver, use the following steps:

- Delete any *VolumeSnapshotClass* present in the cluster.
- Delete all the alpha snapshot CRDs from the cluster by running the following commands:

NOTE: Before deleting the CRDs, ensure that their version is *v1alpha1* by examining the output of the `kubectl get crd` command.

```
kubectl delete crd volumesnapshotclasses.snapshot.storage.k8s.io
```

```
kubectl delete crd volumesnapshotcontents.snapshot.storage.k8s.io
```

```
kubectl delete crd volumesnapshots.snapshot.storage.k8s.io
```

- c. Uninstall the driver using the `csi-uninstall.sh` script by running the command:
`./csi-uninstall.sh --namespace vxflexos`
- d. Install the driver using the install script as described in [Install CSI Driver for Dell EMC PowerFlex using HELM](#) on page 5.

NOTE:

- If you are upgrading from a driver version which was installed using Helm2, ensure that you install Helm3 before installing the driver.
- Installation of the CSI Driver for Dell EMC PowerFlex version 1.2 driver is not supported on Kubernetes upstream clusters running version 1.16. You must upgrade your cluster to 1.17, 1.18, or 1.19 before attempting to install the new version of the driver.

To update any installation parameter after the driver has been installed, change the `myvalues.yaml` file and run the install script with the option `--upgrade`, for example: `./csi-install.sh --namespace vxflexos --values ./myvalues.yaml --upgrade`

2. Upgrade using dell-csi-operator:

Follow the instructions in [Install using Operator](#) on page 7 to upgrade the driver from an older version to version 1.2.

Volume Snapshots

Support for Volume Snapshots (beta)

Volume Snapshots feature in Kubernetes has moved to beta in Kubernetes version 1.17. It was an alpha feature in earlier releases (1.13 onwards). The snapshot API version has changed from v1alpha1 to v1beta1 with this migration.

Starting from version 1.2, the CSI PowerFlex driver supports beta snapshots. Previous versions of the driver supported alpha snapshots.

In order to use Volume Snapshots, ensure the following components have been deployed to your cluster:

- Kubernetes Volume Snapshot CRDs
- Volume Snapshot Controller

NOTE: There is no support for Volume Snapshots on OpenShift 4.3 (as it is based on upstream Kubernetes v1.16). When using the `dell-csi-operator` to install the CSI PowerFlex driver on an OpenShift cluster running v4.3, the external-snapshotter sidecar is not installed.

Topics:

- [Installing the Volume Snapshot CRDs](#)
- [Installing the Volume Snapshot Controller](#)
- [Upgrade from v1alpha1 to v1beta1](#)
- [Volume Snapshot Class](#)
- [Create Volume Snapshot](#)
- [Creating PVCs with Volume Snapshots as Source](#)

Installing the Volume Snapshot CRDs

The Kubernetes Volume Snapshot CRDs can be obtained and installed from the external-snapshotter project on [Github](#).

Alternately, you can install the CRDs by supplying the option `--snapshot-crd` while installing the driver using the new `csi_install.sh` script. If you are installing the driver using the `dell-csi-operator`, there is a helper script provided to install the snapshot CRDs - `scripts/install_snap_crds.sh`.

Installing the Volume Snapshot Controller

Starting with the beta Volume Snapshots, the CSI external-snapshotter sidecar is split into two controllers:

- A common snapshot controller
- A CSI external-snapshotter sidecar

The common snapshot controller must be installed only once in the cluster irrespective the number of CSI drivers installed in the cluster. On OpenShift clusters 4.4 onwards, the common snapshot-controller is pre-installed. In the clusters where it is not present, it can be installed using `kubectl` and the manifests available on [GitHub](#).

NOTE:

- The manifests available on GitHub install v2.0.1 of the snapshotter image - quay.io/k8scsi/snapshot-controller:v2.0.1
- Dell EMC recommends using v2.1.1 image of the snapshot-controller - quay.io/k8scsi/csi-snapshotter:v2.1.1
- The CSI external-snapshotter sidecar is still installed along with the driver and does not involve any extra configuration.

Upgrade from v1alpha1 to v1beta1

All **v1alpha1** snapshot CRDs must be uninstalled from the cluster before installing the **v1beta1** snapshot CRDs and the common snapshot controller. For more information, see [external-snapshotter documentation](#).

Volume Snapshot Class

During the installation of CSI PowerFlex 1.2 driver, a Volume Snapshot Class is created using the new **v1beta1** snapshot APIs. This is the only Volume Snapshot Class required and there is no need to create any other Volume Snapshot Class.

Following is the manifest for the Volume Snapshot Class created during installation:

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshotClass
metadata:
  name: vxflexos-snapclass
driver: csi-vxflexos.dellemc.com
deletionPolicy: Delete
```

Create Volume Snapshot

The following is a sample manifest for creating a Volume Snapshot using the **v1beta1** snapshot APIs:

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshot
metadata:
  name: pvol0-snap1
  namespace: helmtest-vxflexos
spec:
  volumeSnapshotClassName: vxflexos-snapclass
  source:
    persistentVolumeClaimName: pvol0
```

Once the VolumeSnapshot has been successfully created by the CSI PowerFlex driver, a VolumeSnapshotContent object is automatically created. Once the status of the VolumeSnapshot object has the *readyToUse* field set to *true*, it is available for use.

Following is the relevant section of VolumeSnapshot object status:

```
status:
  boundVolumeSnapshotContentName: snapcontent-5a8334d2-eb40-4917-83a2-98f238c4bda1
  creationTime: "2020-07-16T08:42:12Z"
  readyToUse: true
```

Creating PVCs with Volume Snapshots as Source

The following is a sample manifest for creating a PVC with a VolumeSnapshot as a source:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: restorepvc
  namespace: helmtest-vxflexos
spec:
  storageClassName: vxflexos
```

```
dataSource:
  name: pvol0-snap1
  kind: VolumeSnapshot
  apiGroup: snapshot.storage.k8s.io
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 8Gi
```

Volume Expansion

Starting from Version 1.2, the CSI PowerFlex driver supports expansion of Persistent Volumes. This expansion is done online, that is, when PVC is attached to a node.

In order to use this feature, the storage class used to create the PVC needs to have the attribute *allowVolumeExpansion* set to *true*. The storage classes created during the installation (both using Helm or dell-csi-operator) have the *allowVolumeExpansion* set to *true* by default.

In case you are creating more storage classes, make sure that this attribute is set to *true* if you wish to expand any Persistent Volumes created using these new storage classes.

Following is a sample manifest for a storage class which allows for Volume Expansion:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: vxflexos-expand
  annotations:
provisioner: csi-vxflexos.dellemc.com
reclaimPolicy: Delete
allowVolumeExpansion: true
parameters:
  storagepool: pool1
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
- matchLabelExpressions:
  - key: csi-vxflexos.dellemc.com/sample
    values:
    - csi-vxflexos.dellemc.com
```

To resize a PVC, edit the existing PVC spec and set *spec.resources.requests.storage* to the intended size.

For example, if you have a PVC - pvol0 of size 8Gi, then you can resize it to 16 Gi by updating the PVC:

```
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 16Gi #update from 8Gi
  storageClassName: vxflexos
  volumeMode: Filesystem
  volumeName: k8s-0e50dada47
status:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 8Gi
  phase: Bound
```

NOTE: Kubernetes Volume Expansion feature cannot be used to shrink a volume and volumes cannot be expanded to a value that is not a multiple of 8. If attempted, the driver will round up. For example, if the above PVC was edited to have a size of 20 Gb, the size would actually be expanded to 24 Gb, the closest multiple of 8.

Raw Block Support

The PowerFlex driver version 1.2 adds support for Raw Block volumes, which are created using the *volumeDevices* list in the pod template spec with each entry accessing a *volumeClaimTemplate* specifying a *volumeMode: Block*.

Following is an example configuration of **Raw Block Outline**:

```
kind: StatefulSet
apiVersion: apps/v1
metadata:
  name: powerflextest
  namespace: helmtest-vxflexos
spec:
  ...
  spec:
    ...
    containers:
      - name: test
        ...
        volumeDevices:
          - devicePath: "/dev/data0"
            name: pvol0
    volumeClaimTemplates:
      - metadata:
          name: pvol0
        spec:
          accessModes:
            - ReadWriteOnce
          volumeMode: Block
          storageClassName: vxflexos
          resources:
            requests:
              storage: 8Gi
```

Allowable access modes are *ReadWriteOnce*, *ReadWriteMany*, and for block devices that have been previously initialized, *ReadOnlyMany*.

Raw Block volumes are presented as a block device to the pod by using a bind mount to a block device in the node's file system. The driver does not format or check the format of any file system on the block device. Raw Block volumes do support online Volume Expansion, but it is up to the application to manage reconfiguring the file system (if any) to the new size.

For additional information, see - <https://kubernetes.io/docs/concepts/storage/persistent-volumes/#raw-block-volume-support>.

Topology Support

The PowerFlex driver version 1.2 adds support for Topology which forces volumes to be placed on worker nodes that have connectivity to the backend storage. This covers use cases where:

- The PowerFlex SDC may not be installed or running on some nodes
- Users have chosen to restrict the nodes on which the CSI driver is deployed.

This Topology support does not include customer defined topology, users cannot create their own labels for nodes and storage classed and expect the labels to be honored by the driver.

Topics:

- [Topology Usage](#)

Topology Usage

In order to utilize the Topology feature, the storage classes are modified to specify the *volumeBindingMode* as *WaitForFirstConsumer* and to specify the desired topology labels within *allowedTopologies*. This ensures that pod scheduling takes advantage of the topology and be guaranteed that the node selected has access to provisioned volumes.

Storage Class Example with Topology Support:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    meta.helm.sh/release-name: vxflexos
    meta.helm.sh/release-namespace: vxflexos
    storageclass.beta.kubernetes.io/is-default-class: "true"
  creationTimestamp: "2020-05-27T13:24:55Z"
  labels:
    app.kubernetes.io/managed-by: Helm
  name: vxflexos
  resourceVersion: "170198"
  selfLink: /apis/storage.k8s.io/v1/storageclasses/vxflexos
  uid: abb094e6-2c25-42c1-b82e-bd80372e78b2
parameters:
  storagepool: pool1
provisioner: csi-vxflexos.dellemc.com
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
- matchLabelExpressions:
  - key: csi-vxflexos.dellemc.com/6c29fd07674c3356
    values:
      - csi-vxflexos.dellemc.com
```

For more details, see - <https://kubernetes-csi.github.io/docs/topology.html>.

NOTE: In the manifest file of the operator, topology can be enabled on openshift by specifying the system name or *systemid* in the allowed topologies field. *Volumebindingmode* is also set to *WaitForFirstConsumer* by default.

Testing VxFlex OS driver

This chapter contains the following sections:

Topics:

- [Test deploying a simple pod with PowerFlex storage](#)
- [Test creating snapshots](#)
- [Test restoring from a snapshot](#)

Test deploying a simple pod with PowerFlex storage

Test the deployment workflow of a simple pod on PowerFlex storage.

Prerequisites

In the source code, there is a directory that contains examples of how you can use the driver. To use these examples, you must create a `helmtest-vxflexos` namespace, using `kubectl create namespace helmtest-vxflexos`, before you can start testing. HELM 3 must be installed to perform the tests.

About this task

The `starttest.sh` script is located in the `csi-vxflexos/test/helm` directory. This script is used in the following procedure to deploy helm charts that test the deployment of a simple pod.

Steps

1. Navigate to the `test/helm` directory, which contains the `starttest.sh` and the `2vols` directories.

This directory contains a simple Helm chart that will deploy a pod that uses two PowerFlex volumes.

NOTE: Helm tests are designed assuming users are using the default `storageclass` names (`vxflexos` and `vxflexos-xfs`). If your `storageclass` names differ from the default values, such as when deploying with the Operator, please update the templates in `2vols` accordingly (located in `test/helm/2vols/templates` directory). You can use `kubectl get sc` to check for the `storageclass` names.

2. Run the `sh starttest.sh 2vols` command to deploy the pod.

You should see the following:

```
Normal      Pulled          38s           kubelet,
k8s113a-10-247-102-215.lss.emc.com Successfully pulled image "docker.io/
centos:latest"
Normal      Created          38s           kubelet,
k8s113a-10-247-102-215.lss.emc.com Created container
Normal      Started          38s           kubelet,
k8s113a-10-247-102-215.lss.emc.com Started container
/dev/scinib      8125880    36852    7653216    1% /data0
/dev/scinia      16766976   32944    16734032   1% /data1
/dev/scinib on /data0 type ext4 (rw,relatime,data=ordered)
/dev/scinia on /data1 type xfs (rw,relatime,attr2,inode64,noquota)
```

3. To stop the test, run `sh stoptest.sh 2vols`.

This script deletes the pods and the volumes depending on the retention setting you have configured.


Results

An outline of this workflow is described below:

1. The *2vols* helm chart contains two *PersistentVolumeClaim* definitions, one in *pvc0.yaml*, and the other in *pvc1.yaml*. They are referenced by the *test.yaml* which creates the pod. The contents of the *Pvc0.yaml* file are described below:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvol0
  namespace: helmtest-vxflexos
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 8Gi
  storageClassName: vxflexos
```


2. The *volumeMode: Filesystem* requires a mounted file system, and the *resources.requests.storage* of 8Gi requires an 8 GB file. In this case, the *storageClassName: vxflexos* directs the system to use one of the pre-defined storage classes created by the CSI Driver for Dell EMC PowerFlex installation process. This step yields a mounted *ext4* file system. You can see the storage class definitions in the PowerFlex installation helm chart files *storageclass.yaml* and *storageclass-xfs.yaml*.
3. If you compare *pvol0.yaml* and *pvol1.yaml*, you will find that the latter uses a different storage class; *vxflexos-xfs*. This class gives you an *xfs* file system.
4. To see the volumes you created, run `kubectl get persistentvolumeclaim -n helmtest-vxflexos` and `kubectl describe persistentvolumeclaim -n helmtest-vxflexos`.

 **NOTE:** For more information about Kubernetes objects like *StatefulSet* and *PersistentVolumeClaim* see [Kubernetes documentation: Concepts](#).

Test creating snapshots


Use the procedure in this topic to create snapshots.

Prerequisites

 **NOTE:** Test the workflow for snapshot creation. Snapshots are not enabled for Openshift.

Steps

1. Start the *2vols* container and leave it running.

 **NOTE:**

- Helm tests are designed assuming users are using the default *storageclass* names (*vxflexos* and *vxflexos-xfs*). If your *storageclass* names differ from the default values, such as when deploying with the Operator, update the templates in *2vols* accordingly (located in `test/helm/2vols/templates` directory). You can use `kubectl get sc` to check for the *storageclass* names.
- Helm tests are designed assuming users are using the default *snapshotclass* name. If your *snapshotclass* names differ from the default values, update *snap1.yaml* and *snap2.yaml* accordingly.

2. Run the `snaptest.sh` shell script.

This will create a snapshot of each of the volumes in the container using *VolumeSnapshot* objects defined in *snap1.yaml* and *snap2.yaml*. The following are the contents of *snap1.yaml*:

```
apiVersion: snapshot.storage.k8s.io/v1alpha1
kind: VolumeSnapshot
metadata:
  name: pvol0-snap1
  namespace: helmtest-vxflexos
spec:
  snapshotClassName: vxflexos-snapclass
  source:
    name: pvol0
    kind: PersistentVolumeClaim
```


Results

The `snaptest.sh` script will create a snapshot using the definitions in the `snap1.yaml` file. The `spec.source` section contains the volume that will be snapped. For example, if the volume to be snapped is `pvol0`, then the created snapshot is named `pvol0-snap1`.

NOTE: The `snaptest.sh` shell script creates the snapshots, describes them, and then deletes them. You can see your snapshots using `kubectl get volumesnapshot -n test`.

Notice that this `VolumeSnapshot` class has a reference to a `snapshotClassName: vxflexos-snapclass`. The CSI Driver for Dell EMC PowerFlex installation creates this class as its default snapshot class. You can see its definition in the installation directory file `volumesnapshotclass.yaml`.

Test restoring from a snapshot

Test the restore operation workflow to restore from a snapshot.

Prerequisites

Ensure that you have stopped any previous test instance before performing this procedure.

About this task

To test the restore operation from a snapshot:

Steps

Run the `snapprestoretest.sh` shell script.

This script deploys the `2vols` example, creates a snap of `pvol0`, and then updates the deployed helm chart from the updated directory `2vols+restore`. This then adds an additional volume that is created from the snapshot.

NOTE:

- Helm tests are designed assuming users are using the default `storageclass` names (`vxflexos` and `vxflexos-xfs`). If your `storageclass` names differ from the default values, such as when deploying with the Operator, update the templates for snap restore tests accordingly (located in `test/helm/2vols+restore/template` directory). You can use `kubectl get sc` to check for the `storageclass` names.
- Helm tests are designed assuming users are using the default `snapshotclass` name. If your `snapshotclass` names differ from the default values, update `snap1.yaml` and `snap2.yaml` accordingly.

Results

An outline of this workflow is described below:

- The snapshot is taken using `snap1.yaml`.
- `Helm` is called to upgrade the deployment with a new definition, which is found in the `2vols+restore` directory. The `csi-vxflexos/test/helm/2vols+restore/templates` directory contains the newly created `createFromSnap.yaml` file. The script then creates a `PersistentVolumeClaim`, which is a volume that is dynamically created from the snapshot. Then the helm deployment is upgraded to contain the newly created third volume. In other words, when the `snapprestoretest.sh` creates a new volume with data from the snapshot, the restore operation is tested. The contents of the `createFromSnap.yaml` are described below:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: restorepvc
  namespace: helmtest-vxflexos
spec:
  storageClassName: vxflexos
  dataSource:
    name: pvol0-snap1
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
  accessModes:
    - ReadWriteOnce
```

```
resources:
  requests:
    storage: 8Gi
```

 **NOTE:** The *spec.dataSource* clause, specifies a source *VolumeSnapshot* named *pvol0-snap1* which matches the snapshot's name in *snap1.yaml*.

Troubleshooting

This chapter contains the following section:

Topics:

- [Troubleshooting](#)

Troubleshooting

The following table lists the CSI Driver for Dell EMC VxFlex OS installation troubleshooting scenarios when installing on Kubernetes:

Table 1. Troubleshooting

Symptoms	Prevention, resolution, or workaround
The installation fails with the following error message: <code>Node xxx does not have the SDC installed</code>	Install the PowerFlex SDC on listed nodes. The SDC must be installed on all the nodes that needs to pull an image of the driver.
When you run the command <code>kubectl describe pods vxflexos-controller-0 -n vxflexos</code> , the system indicates that the driver image could not be loaded.	<ul style="list-style-type: none"> • If on Kubernetes, edit the <code>daemon.json</code> file found in the registry location and add <pre>{ "insecure-registries" : ["hostname.cloudapp.net:5000"] }</pre> • If on Openshift, run the command <code>oc edit image.config.openshift.io/cluster</code> and add registries to yaml file that is displayed when you run the command.
The <code>kubectl logs -n vxflexos vxflexos-controller-0 driver</code> logs shows that the driver is not authenticated.	Check the username, password, and the gateway IP address for the PowerFlex system.
The <code>kubectl logs vxflexos-controller-0 -n vxflexos driver</code> logs shows that the system ID is incorrect.	Use the <code>get_vxflexos_info.sh</code> to find the correct system ID. Add the system ID to <code>myvalues.yaml</code> script.
Defcontext mount option seems to be ignored, volumes still are not being labeled correctly.	Ensure SELinux is enabled on worker node, and ensure your container run time manager is properly configured to be utilized with SELinux.
Mount options that interact with SELinux are not working (like defcontext).	Check that your container orchestrator is properly configured to work with SELinux.