

# CSI Driver for Dell EMC PowerMax

Version 1.2

## Product Guide

REV 01

April 2020

Copyright © 2019-2020 Dell Inc. or its subsidiaries. All rights reserved.

Dell believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED “AS-IS.” DELL MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. USE, COPYING, AND DISTRIBUTION OF ANY DELL SOFTWARE DESCRIBED IN THIS PUBLICATION REQUIRES AN APPLICABLE SOFTWARE LICENSE.

Dell Technologies, Dell, EMC, Dell EMC and other trademarks are trademarks of Dell Inc. or its subsidiaries. Other trademarks may be the property of their respective owners. Published in the USA.

Dell EMC  
Hopkinton, Massachusetts 01748-9103  
1-508-435-1000 In North America 1-866-464-7381  
[www.DellEMC.com](http://www.DellEMC.com)

# CONTENTS

<b>Chapter 1</b>	<b>Introduction</b>	<b>5</b>
	Product overview.....	6
	CSI Driver components.....	6
	Controller Plug-in.....	6
	Node Plug-in.....	6
	Features of the CSI Driver for Dell EMC PowerMax .....	6
<b>Chapter 2</b>	<b>Installation</b>	<b>9</b>
	Installation overview.....	10
	Prerequisites.....	10
	Fiber channel requirements.....	11
	iSCSI requirements.....	11
	Enable Kubernetes feature gates (Kubernetes only).....	11
	Configure Docker service (Kubernetes only).....	13
	Install the Helm 3 package manager for helm-based installation.....	13
	Linux multipathing requirements.....	13
	Install the CSI Driver for Dell EMC PowerMax using Helm.....	13
	Install the CSI Driver for Dell EMC PowerMax using Operator .....	15
	Update the CSI Driver for Dell EMC PowerMax .....	16
	CSI Driver usage.....	16
	Controller Plug-in query commands.....	17
	Node plug-in query command.....	17
	Certificate validation for Unisphere REST API calls.....	17
	Snapshot Support.....	18
<b>Chapter 3</b>	<b>Test the CSI Driver for Dell EMC PowerMax</b>	<b>21</b>
	Test the CSI Driver for Dell EMC PowerMax.....	22
	Volume Test.....	22
	Snapshot Test.....	22



# CHAPTER 1

## Introduction

This chapter includes the following topics:

- [Product overview](#) ..... 6
- [CSI Driver components](#)..... 6
- [Features of the CSI Driver for Dell EMC PowerMax](#) ..... 6

## Product overview

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

The CSI Driver for Dell EMC PowerMax and Kubernetes communicate using the Container Storage Interface protocol. The CSI Driver for Dell EMC PowerMax conforms to CSI specification version 1.1. It is compatible with Kubernetes version 1.14 and Openshift 4.2 (with the Red Hat Enterprise Linux 7.6 worker nodes). The CSI driver uses Unisphere REST APIs for PowerMax version 9.1 to manage the PowerMax arrays.

## CSI Driver components

This topic describes the components of the CSI Driver for Dell EMC PowerMax.

The CSI Driver for Dell EMC PowerMax has two components:

- Controller plug-in
- Node plug-in

### Controller Plug-in

The Controller plug-in is deployed in a `StatefulSet` in the Kubernetes cluster with maximum number of replicas set to 1. There is one pod for the Controller plug-in that gets scheduled on any node which is not necessarily the master.

This pod contains the CSI Driver for Dell EMC PowerMax container and a few side-car containers like the *provisioner*, *attacher*, and *external-snapshotter* that the Kubernetes community provides.

The Controller plug-in primarily deals with provisioning activities like creating volumes, deleting volumes, attaching the volume to a node, and detaching the volume from a node. Additionally, it deals with creating snapshots, deleting snapshots and creating a volume from snapshot. The CSI Driver for Dell EMC PowerMax automates the creation and deletion of Storage Groups (SGs) and Masking Views when required.

### Node Plug-in

The Node plug-in is deployed in a *DaemonSet* in the Kubernetes cluster. The Node plug-in deploys the pod containing the driver container on all nodes in the cluster (where the scheduler can schedule the pod).

The Node plug-in performs tasks like identifying, publishing, and unpublishing a volume to the node.


The Node plug-in identifies the Fiber Channel Host Bus Adapters (HBAs) and the iSCSI Qualified Names (IQN) present on the node and creates *Hosts* using these initiators on the PowerMax array. On a single node, the same plug-in supports both iSCSI and Fiber Channel connectivity for different PowerMax arrays. The Controller plug-in uses these hosts to create masking views for the nodes.

## Features of the CSI Driver for Dell EMC PowerMax

The CSI Driver for Dell EMC PowerMax has the following features:

- Supports CSI 1.1
- Supports Kubernetes version 1.14

- Supports OpenShift 4.2 with Red Hat Enterprise Linux CoreOS on master nodes and Red Hat Enterprise Linux 7.6 on worker nodes
- Requires Unisphere for PowerMax 9.1
- Supports Fibre Channel
- Supports Red Hat Enterprise Linux 7.6 host operating system
- Supports PowerMax Enginuity versions 5978.479.479, 5978.444.444 and 5978.221.221
- Supports Linux native multipathing
- Persistent Volume (PV) capabilities:
  - Create
  - Delete
  - Create from Snapshot
- Dynamic and Static PV provisioning
- Volume mount as ext4 or xfs file system on the worker node
- Volume prefix for easier LUN identification in Unisphere
- Helm 3 charts installer
- Dell EMC Storage CSI Operator deployment
- Snapshot Capabilities - create, delete (Kubernetes Only, not supported on Openshift)
- Access modes:
  - SINGLE\_NODE\_WRITER
  - SINGLE\_NODE\_READER\_ONLY

 **Note:** Volume Snapshots is an Alpha feature in Kubernetes. It is recommended for use only in short-lived testing clusters, as features in the Alpha stage have an increased risk of bugs and a lack of long-term support. See [Kubernetes documentation](#) for more information about feature stages.





# CHAPTER 2

## Installation

This chapter includes the following topics:

• <a href="#">Installation overview</a> .....	10
• <a href="#">Prerequisites</a> .....	10
• <a href="#">Install the CSI Driver for Dell EMC PowerMax using Helm</a> .....	13
• <a href="#">Install the CSI Driver for Dell EMC PowerMax using Operator</a> .....	15
• <a href="#">Update the CSI Driver for Dell EMC PowerMax</a> .....	16
• <a href="#">CSI Driver usage</a> .....	16
• <a href="#">Certificate validation for Unisphere REST API calls</a> .....	17
• <a href="#">Snapshot Support</a> .....	18

## Installation overview

This topic gives an overview of the CSI Driver for Dell EMC PowerMax installation.

The CSI Driver for Dell EMC PowerMax can either be deployed in Kubernetes platforms using Helm version 3 charts or the Dell EMC Storage CSI Operator. The CSI Driver for Dell EMC PowerMax 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 PowerMax. The shell script installs the CSI Driver container image along with the required Kubernetes sidecar containers.

**Note:** In the previous releases of the driver, the driver was installed using Helm version 2. Starting from v1.2.0, the driver installation charts and test charts have been upgraded to be compatible with Helm 3. Since Helm 3 is not backward compatible with Helm 2, it means that rolling upgrade of driver from v1.0.0/v1.1.0 to v1.2.0 is not supported. So, any existing installations of the driver has to be uninstalled and then the users may choose to use the Helm 3 based installer or the Dell Storage CSI Operator to install v1.2.0 of the driver (See, [Update the CSI Driver for Dell EMC PowerMax](#) on page 16).

If using a Helm installation, the controller section of the Helm chart installs the following components in a StatefulSet in the *powermax* namespace:

- CSI Driver for Dell EMC PowerMax
- Kubernetes Provisioner that provisions the persistent volumes
- Kubernetes Attacher that attaches the volumes to the containers

The node section of the Helm chart installs the following component in a DaemonSet in the *powermax* namespace:

- CSI Driver for Dell EMC PowerMax
- Kubernetes Registrar that handles the driver registration
- Kubernetes Snapshotter that creates and deletes snapshots

## Prerequisites

This topic lists the prerequisites to install the CSI Driver for Dell EMC PowerMax.

Before you install the CSI Driver for Dell EMC PowerMax, you must complete the following task:

- Install Kubernetes.  
The CSI Driver for Dell EMC PowerMax works with Kubernetes version 1.14.  
OR
- Install Openshift.  
The CSI Driver for Dell EMC PowerMax works with OpenShift 4.2 with Red Hat Enterprise Linux 7.6 worker nodes.
- Enable Kubernetes feature gates (Kubernetes only)
- Install the Helm 3 package manager for helm-based installation  
OR
- Install Dell EMC Storage CSI Operator for operator-based installation
- Configure Docker service (Kubernetes only)
- Fiber channel requirements
- iSCSI requirements

- Linux multipathing requirements

## Fiber channel requirements

CSI Driver for Dell EMC PowerMax supports fiber channel communication. Ensure that the following requirements are met before you install the CSI Driver:

- Zoning of the Host Bus Adapters (HBAs) to the fiber channel port director must be completed.
- Ensure that the HBA WWNs (initiators) appear on the list of initiators that are logged into the array.
- If number of volumes that are published in a particular node is high, configure the maximum number of LUNs for an HBA. See the appropriate HBA document to configure maximum number of LUNs.

## iSCSI requirements

The CSI Driver for Dell EMC PowerMax supports iSCSI connectivity. These requirements are applicable for the nodes that use iSCSI initiator to connect to the PowerMax arrays.

Set up the iSCSI initiators as follows:


- Ensure that the iSCSI initiators are available on both Master and Worker nodes.
- Kubernetes nodes should have access (network connectivity) to an iSCSI director on the Dell EMC PowerMax array that has IP interfaces. Manually create IP routes for each node that connects to the Dell EMC PowerMax.
- All Kubernetes nodes must have the *iscsi-initiator-utils* package installed.
- Ensure that the iSCSI initiators on the nodes are not a part of any existing Host (Initiator Group) on the Dell EMC PowerMax.
- The CSI Driver needs the port group names containing the required iSCSI director ports. These Port Groups must be set up on each Dell EMC PowerMax array. All the port groups names supplied to the driver must exist on each Dell EMC PowerMax with the same name.

For information about configuring iSCSI, see Dell EMC PowerMax documentation on [Dell EMC Support](#).

## Enable Kubernetes feature gates (Kubernetes only)

Enable the Kubernetes feature gates before installing the CSI Driver for Dell EMC PowerMax.

### About this task

 **Note:** Enable other feature gates for different Kubernetes versions and distributions. The feature gates that are described in this section are applicable to Kubernetes 1.14.

The [Feature Gates section](#) of the Kubernetes documentation lists the Kubernetes feature gates. Enable the following Kubernetes feature gates:

- KubeletPluginsWatcher
- CSINodeInfo
- CSIDriverRegistry
- BlockVolume
- CSIBlockVolume
- VolumeSnapshotDataSource

## Procedure

1. On each master and node of Kubernetes, edit `/var/lib/kubelet/config.yaml` to add the following lines at the end to set feature-gate settings for the kubelets:

```
KubeletPluginsWatcher: true
CSINodeInfo: true
CSIDriverRegistry: true
BlockVolume: true
CSIBlockVolume: true
VolumeSnapshotDataSource: true
```

2. On the master, set the feature gate settings of the `kube-apiserver.yaml` file as follows:

```
/etc/kubernetes/manifests/kube-apiserver.yaml - --feature-
gates=KubeletPluginsWatcher=true,CSINodeInfo=true,CSIDriverRegistry=true
,BlockVolume=true,CSIBlockVolume=true
```

3. On the master, set the feature gate settings of the `kube-controller-manager.yaml` file as follows:


```
/etc/kubernetes/manifests/kube-controller-manager.yaml - --feature-
gates=KubeletPluginsWatcher=true,CSINodeInfo=true,CSIDriverRegistry=true
,BlockVolume=true,CSIBlockVolume=true
```

4. On the master, set the feature gate settings of the `kube-scheduler.yaml` file as follows:

```
/etc/kubernetes/manifests/kube-scheduler.yaml - --feature-
gates=KubeletPluginsWatcher=true,CSINodeInfo=true,CSIDriverRegistry=true
,BlockVolume=true,CSIBlockVolume=true
```

5. On each node, edit the variable `KUBELET_KUBECONFIG_ARGS` of `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` file as follows:

```
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrapkubeconfig=/etc/
kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/
kubelet.conf --allow-privileged=true --feature-
gates=KubeletPluginsWatcher=true,CSINodeInfo=true,CSIDriverRegistry=true
,BlockVolume=true,CSIBlockVolume=true"
```

 **Note:** The location of the `10-kubeadm.conf` file depends on the Kubernetes version and the installation process.

6. Restart the kublet with `systemctl daemon-reload` and `systemctl restart kubelet` on all nodes.

## Configure Docker service (Kubernetes only)

Configure the mount propagation in Docker on all Kubernetes nodes before installing the CSI Driver for Dell EMC PowerMax. The recommended docker version is 18.06.

### Procedure

1. Edit the service section of `/etc/systemd/system/multi-user.target.wants/docker.service` file to add the following lines:

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

**Note:** The location of the `docker.service` file depends on the Kubernetes version and the installation process.

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

## Install the Helm 3 package manager for helm-based installation

### About this task

The CSI Driver for Dell EMC PowerMax version 1.2 supports Helm 3 only. Helm 3 has an easier install than previous versions and poses less security risks because no Tiller installation or special privileges are required. To install Helm 3, follow the instructions here: [Install Helm 3](#).

## Linux multipathing requirements

CSI Driver for Dell EMC PowerMax supports Linux multipathing. Configure Linux multipathing before installing the CSI Driver.

Set up Linux multipathing as follows:

- All the nodes must have *Device Mapper Multipathing* package installed.
- **Note:** This package is installed by default and creates a multipath configuration file. This file is located in `/etc/multipath.conf`.
- Enable multipathing using `mpathconf --enable --with_multipathd y`
- Enable `user_friendly_names` and `find_multipaths` in the `multipath.conf` file.
- Ensure that the `multipath` command for `multipath.conf` is available on all Kubernetes nodes.

## Install the CSI Driver for Dell EMC PowerMax using Helm

Install the CSI Driver for Dell EMC PowerMax using this procedure.


### Before you begin

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

- You have the downloaded files ready for this procedure as described in the [Prerequisites](#) section..

- You have the driver's Helm chart from the source repository at <https://github.com/dell/csi-powermax>, ready for this procedure.
- The top-level helm directory contains the *install.powermax* and *uninstall.powermax* shell scripts. The scripts perform certain preinstallation and postinstallation operations (like creating Custom Resource Definitions), which cannot be performed in the Helm chart.
- You have the Kubernetes secret with your Unisphere username and password. You can use the *secret.yaml* file to create the secret with the following values to match the default installation parameters:


- Name: `powermax-creds`
- Namespace: `powermax`

 **Note:** For more information about creating a Kubernetes secret, see [Kubernetes documentation: Overview of Secrets](#).

- If using iSCSI, initiators are available on all nodes, including the master and worker nodes.
- The Kubernetes feature gates are enabled.
- The mount propagations are configured in Docker.
- The nonsecure registries are defined in Docker, for CSI drivers that are hosted in a nonsecure location.

### Procedure

1. Run `git clone https://github.com/dell/csi-powermax.git` to clone the git repository to the master node of the Kubernetes cluster.
2. Run `cd csi-powermax/helm && cp csi-powermax/values.yaml ./myvalues.yaml` to change the directory to the top-level helm directory.
3. Run `vi myvalues.yaml` to edit the *myvalues.yaml* file, and configure the Unisphere endpoint.
4. Copy the *csi-powermax/values.yaml* to the *myvalues.yaml* in the helm directory and provide values for the following parameters:
  - `unisphere`: This value must be the IP address or the hostname. It must include the port number as well.
  - `clusterPrefix`: This parameter holds a prefix that is used during the creation of various masking-related entities on the array. These masking-related entities include Storage Groups, Masking Views, Hosts, and Volume Identifiers. The value that you specify here must be unique. Ensure that no other CSI PowerMax driver is managing the same arrays that are configured with the same prefix. The max length for this prefix is three characters.
  - `portGroups`: This parameter holds a list of comma-separated Port group names. Any port group that is specified here, must be present on all the arrays that the driver manages.
 

 **Note:** The `portGroups` parameter is required for iSCSI initiators only.
  - `arrayWhitelist`: This parameter holds a list of comma-separated array IDs. If this parameter remains empty, the driver manages all the arrays that are managed by the Unisphere instance that is configured for the driver. Specify the IDs of the arrays that you want to manage, using the driver.
  - `driver`: This parameter must specify the location of the docker image for the driver container. This value specifies the image location and tag for the driver image, and usually remains unchanged.
  - `symmetrixID`: This parameter must specify the Dell EMC PowerMax arrays that the driver manages. This value is used to create a default Storage class.

- `storageResourcePool`: This parameter must mention one of the SRPs on the PowerMax array that the `symmetrixID` specifies. This value is used to create the default Storage class.
  - `serviceLevel`: This parameter must mention one of the Service Levels on the PowerMax array. This value is used to create the default Storage class.
  - `skipCertificateValidation`: This parameter should be set to *false* if you want to do client side TLS verification of Unisphere for PowerMax SSL certificates. It is set to *true* by default.
  - `transportProtocol`: This parameter indicates a preferred transport protocol for the Kubernetes cluster which helps the driver choose between FC and iSCSI when a node has both FC and iSCSI connectivity to a particular PowerMax array.
5. Create a secret file for the Unisphere credentials by editing the `secret.yaml`. 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`
  6. Run `kubectl create namespace powermax` to create the *PowerMax* namespace.
  7. Run `kubectl create -f secret.yaml` to create the secret.
  8. Run `sh install.powermax` to install the driver.

This script also runs the `verify.kubernetes` script that is present in the same directory. You are prompted to enter the credentials for each of the Kubernetes nodes. The `verify.kubernetes` script needs the credentials to check if the kubelet is configured with the appropriate feature gates on each of the Kubernetes nodes.

## Results

The CSI Driver for Dell EMC PowerMax is installed. You can check for the pods that are deployed by running the following command:

```
kubectl get pods -n powermax
```

Test the installation of your driver using the procedure found in [Test the CSI Driver for Dell EMC PowerMax](#) section.

# Install the CSI Driver for Dell EMC PowerMax using Operator

Learn how to install the CSI Driver for Dell EMC PowerMax using Operator.

## About this task

From version 1.2.0, CSI Driver for Dell EMC Powermax can also be installed using the new Dell EMC Storage Operator. The Dell EMC Storage CSI Operator is a Kubernetes Operator which can be used to install and manage the CSI Drivers 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](#). 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](#).

Instructions on how to deploy the CSI Driver for Dell EMC PowerMax using the operator is available on [GitHub](#). 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 and the parameters for installation and configuration will be slightly different from the ones specified using 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 the CSI Driver for Dell EMC PowerMax

Procedure to update the CSI Driver for Dell EMC PowerMax.

### About this task

Users can update the driver using Helm:

### Procedure

1. Uninstall the driver using `uninstall.powermax` under `csi-powermax/helm` directory.
2. Get the latest code for CSI Driver for Dell EMC PowerMax Version 1.2 from [GitHub](#).
3. Prepare `myvalues.yaml` file.
4. Execute `./install.powermax` to upgrade the driver.
5. List the pods with the following command (to verify the status):

```
kubectl get pods -n powermax
```

**Note:** In version 1.2 of the driver, only Helm 3 is supported. Hence, it is not possible to perform a rolling upgrade using the `upgrade.powermax` script from a driver previously installed with Helm 2.

## CSI Driver usage

Once you install the plug-in, it creates a default storage class using parameters from `myvalues.yaml`. You can also create your own storage class by specifying parameters which decide how storage gets provisioned on the Dell EMC PowerMax array. The storage classes have two mandatory parameters and two optional parameters as follows:

Mandatory parameters:

- SYMID – Symmetrix ID of the Dell EMC PowerMax
- SRP – Storage Resource Pool name

Optional parameters:

- ServiceLevel – Service Level for the volume. If not specified, the driver takes **Optimized** service level as default. As a best practice, it is suggested to use **Optimized** service level or use only metals (Diamond, Platinum, Gold) for all storage classes. Avoid using **Optimized** service level for some storage classes and Service Level like **Gold** for some storage class.
- Application Prefix – Used to group volumes belonging to the same application.

You can create Persistent Volumes (PV) and PersistentVolumeClaims (PVC) using these storage classes. These PVC names can be used in the pod manifests where you can specify which containers need these volumes and where they have to be mounted. The creation of PVCs and pods is outside the scope of this document. See the Kubernetes documentation about creating PVCs and pods.

Starting v1.2 release of the CSI Driver for Dell EMC PowerMax, a snapshot class is also created by the installer (both Helm and Operator based installation methods). This snapshot class does not



have any parameters. Also, there is no need to create any more snapshot class apart from the one created during the installation.

You can create VolumeSnapshots using this snapshot class. These VolumeSnapshot names can further be used as sources to create Persistent Volume Claims (PVC).

## Controller Plug-in query commands

This topic lists the commands to view the details of StatefulSet and check logs for the Controller Plug-in.

### Procedure

1. Run the following command to query the details of the StatefulSet:

```
kubectl get statefulset -n powermax
kubectl describe statefulset powermax-controller -n powermax
```

2. Run the following command to check the logs for the Controller plug-in:

```
kubectl logs powermax-controller-0 driver -n powermax
```

Similarly, logs for provisioner, snapshotter, and attacher sidecars can be obtained by specifying the container names.

## Node plug-in query command

This topic lists the commands to view the details of DaemonSet.

### Procedure

1. Run the following command to get the details of the DaemonSet:

```
kubectl get daemonset -n powermax
kubectl describe daemonset powermax-node -n powermax
```

2. Use the following sample command to check the logs for the Node plug-in:

```
kubectl logs -n powermax <node plugin pod name> driver
```

## Certificate validation for Unisphere REST API calls

This topic provides details about setting up the certificate validation for the CSI Driver for Dell EMC PowerMax.

### Before you begin

As part of the CSI driver installation, the CSI driver requires a secret with the name *powermax-certs* present in the namespace *powermax*. This secret contains the X509 certificates of the CA which signed the Unisphere SSL certificate in PEM format. This secret is mounted as a volume in the driver container. In the earlier releases, if the install script did not find the secret, it created an empty secret with the same name. From 1.2.0 release, the secret volume has been made optional. The install script no longer attempts to create an empty secret.

### About this task

The CSI driver exposes an install parameter `skipCertificateValidation` which determines if the driver performs client-side verification of the Unisphere certificates. The `skipCertificateValidation` parameter is set to *true* by default, and the driver does not verify the Unisphere certificates.

If the `skipCertificateValidation` parameter is set to *false* and a previous installation attempt created an empty secret, then this secret must be deleted and re-created using the CA certs.

If the Unisphere certificate is self-signed or if you are using an embedded Unisphere, then perform the following steps:

#### Procedure

1. To fetch the certificate, run `openssl s_client -showcerts -connect <Unisphere IP> :8443 </dev/null 2>/dev/null | openssl x509 -outform PEM > ca_cert.pem`

The IP address varies for each user.

2. To create the secret, run `kubectl create secret generic powermax-certs --from-file=ca_cert.pem -n powermax`

## Snapshot Support

Learn about volume snapshots and the supported functions.

### About this task

Support for volume snapshots added in the 1.2.0 release, to enable customers to create and delete volume snapshots and create volumes from these snapshots. Volume snapshots are a point in time copy of the source volume. Kubernetes does not support any type of grouping mechanism for volumes. So, these snapshots are taken for each individual volume and they should not be confused with the Storage Group level snapshots on the PowerMax array.

**Note:** Unisphere for PowerMax only supports manipulation of snapshots at a storage group level.

#### Procedure

1. To create a VolumeSnapshot, you have to use the VolumeSnapshotClass, which is created as part of the installation, to create a VolumeSnapshot object. Here is a sample manifest:

VolumeSnapshot Example:

```
apiVersion: snapshot.storage.k8s.io/v1alpha1
kind: VolumeSnapshot
metadata:
  name: snap-pvol-0
  namespace: test
spec:
  snapshotClassName: powermax-snapclass
  source:
    kind: PersistentVolumeClaim
    name: pvol-0
```

Once the snapshot is successfully created by the driver, a VolumeSnapshotContent object is automatically created. The status of VolumeSnapshot is updated with the creationTime and the readyToUse flag set to true. This means that the VolumeSnapshot is available to be used for any future operations. Here is an example of a VolumeSnapshot object:

**VolumeSnapshot Completed:**

```

apiVersion: snapshot.storage.k8s.io/v1alpha1
kind: VolumeSnapshot
metadata:
  creationTimestamp: "2020-03-20T12:23:36Z"
  finalizers:
    - snapshot.storage.kubernetes.io/volumesnapshot-protection
  generation: 3
  name: snap-pvol-0
  namespace: test
  resourceVersion: "19644628"
  selfLink: /apis/snapshot.storage.k8s.io/v1alpha1/namespaces/test/volumesnapshots/snap-pvol-0-pool1-0
  uid: 9fa5fea9-6aa5-11ea-a026-005056809fcd
spec:
  snapshotClassName: powermax-snapclass
  snapshotContentName: snapcontent-9fa5fea9-6aa5-11ea-a026-005056809fcd
  source:
    apiGroup: null
    kind: PersistentVolumeClaim
    name: pvol-0
status:
  creationTime: "2020-03-20T12:26:21Z"
  readyToUse: true
  restoreSize: null

```

2. To create a volume from the VolumeSnapshot, you must use a PVC with a source of VolumeSnapshot:

**PVC from Snapshot:**

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: restored-pvol-0
  namespace: test
spec:
  storageClassName: powermax
  dataSource:
    name: snap-pvol-0
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 8Gi

```

when the volume is created, it should be in the Bound state:

**bound pvc:**

NAME	STORAGECLASS	STATUS	VOLUME	CAPACITY	ACCESS
restored-pvol-0-pool1-0		Bound	pmax-6f5af23c6a	9Gi	
RWO	powermax	102s			

Support for *VolumeSnapshots* is in Alpha stage for Kubernetes v1.14. This means that it may not be suitable for use in production environments. There are a few limitations in the

*external-snapshotter* sidecar provided by the Kubernetes community which can have impact on snapshot related operations with the CSI driver. These include:

- a. Create snapshot requests are not retried by the external-snapshotter in the event of an error. This can happen if the driver is overloaded with many snapshot requests. If there is an initial error in creating a snapshot the volumesnapshot object will be left unbound with no corresponding volumesnapshotcontent object. To retry the operation you must delete and recreate the volumesnapshot object manually.
- b. If you encounter one of these errors, a finalizer `snapshot.storage.kubernetes.io/pvc-protection` may be left on the Persistent Volume Claim (PVC) for the source volume from which the snapshot was being created. This will prevent the deletion of this PVC and the PVC object has to be patched to remove the finalizer to enable deletion of the PVC.
- c. Due to restrictions around the maximum number of parallel RESTAPI operations which are supported by Unisphere for PowerMax, the driver throttles incoming requests to create snapshots. If there are significant number of volume snapshots being created in parallel, you may see context deadline exceeded messages because the external snapshotter timed out waiting for the response. Because of the restriction mentioned in step a, this operation will not be retried. To avoid this, the recommendation is to not take more than a few snapshots concurrently.

# CHAPTER 3

## Test the CSI Driver for Dell EMC PowerMax

This chapter includes the following topics:

- [Test the CSI Driver for Dell EMC PowerMax](#)..... 22

## Test the CSI Driver for Dell EMC PowerMax

This topic provides information about testing the CSI Driver for Dell EMC PowerMax. The tests are validated using bash as the default shell.

To run the test for CSI Driver for Dell EMC PowerMax, install Helm 3.

**Note:** Helm 3 is not supported officially in Openshift 4.2.

The *csi-powermax* repository includes examples of how you can use the CSI Driver for Dell EMC PowerMax. These examples automate the creation of pods using the default storage classes that were created during installation. The shell scripts are used to automate the installation and uninstallation of helm charts for the creation of pods with different number of volumes. To test the installation of the CSI driver, perform the following procedures:

### Volume Test

Procedure to perform a volume test.

#### About this task

**Note:** Helm tests are designed assuming users are using the default storageclass names (powermax and powermax-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.

#### Procedure

1. Create a namespace with the name *test*.
2. Run the `cd csi-powermax/test/helm` command to go to the *csi-powermax/test/helm* directory, which contains the *starttest.sh* and the *2vols* directories.
3. Run the *starttest.sh* script and provide it a test name. The following is a sample command that can be used to run the 2vols test:

```
./starttest.sh -t 2vols -n test
```

This script installs a helm chart that creates a pod with a container, creates two PVCs, and mounts them into the created container. You can now log in to the newly created container and check the mounts.

4. Run the `./stoptest.sh -t 2vols` script to stop the test.

This script deletes the pods and the PVCs created during the test and uninstalls the helm chart.

### Snapshot Test

Procedure to perform snapshot test.

#### About this task

**Note:** This test is designed assuming users are using the snapshot class name `powermax-snapclass` which is created by the Helm based installer. If you have an operator based deployment, the name of the snapshot class will differ. You must update the snapshot class name in the file *snap1.yaml* present in the `test/helm` folder based on your method of deployment. To get a list of volume snapshot classes, run the command - `kubectl get volumesnapshotclass`

## Procedure

1. Create a namespace with the name *test*.
2. Run the `cd csi-powermax/test/helm` command to go to the `csi-powermax/test/helm` directory, which contains the *snaprestoretest.sh*.
3. Run the *snaprestoretest.sh* script.
4. The following command can be used to run the *snaprestoretest.sh*:

```
bash snaprestoretest.sh
```

This script installs a helm chart that creates a pod with a container, creates two PVCs, and mounts them into the created container. Then it writes some data to one of the PVCs. After that, it creates a snapshot on that PVC and uses it as a datasource to create a new PVC. It mounts the newly created PVC to the container created earlier and then list the contents of the source and the target PVCs. Finally, it cleans up all the resources created as part of the test.

