

# CSI Driver for Dell EMC PowerMax

## Product Guide

1.4

## 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.

<b>Chapter 1: Introduction.....</b>	<b>5</b>
Product overview.....	5
CSI Driver components.....	5
Controller plug-in.....	5
Node plug-in.....	5
Features of the CSI Driver for Dell EMC PowerMax .....	6
<b>Chapter 2: Installation.....</b>	<b>7</b>
Installation overview.....	7
Prerequisites.....	7
Fibre Channel requirements.....	8
iSCSI requirements.....	8
Configure Docker service (Kubernetes only).....	8
Install the Helm 3 package manager for helm-based installation.....	9
Linux multipathing requirements.....	9
Ports in port group.....	9
Install the CSI Driver for Dell EMC PowerMax using Helm.....	9
Install the CSI Driver for Dell EMC PowerMax using Operator .....	11
Update the CSI Driver for Dell EMC PowerMax .....	11
Installation using Helm.....	11
Installation using dell-csi-operator.....	12
CSI Driver usage.....	12
Controller plug-in query commands.....	12
Node plug-in query command.....	13
Certificate validation for Unisphere REST API calls.....	13
Support for Volume Snapshots (beta).....	13
Installing the Volume Snapshot CRDs.....	14
Installing the Volume Snapshot Controller.....	14
Upgrade from v1alpha1 to v1beta1.....	14
Volume Snapshot Class.....	14
Creating Volume Snapshots.....	14
Creating PVCs with VolumeSnapshots as source.....	15
Creating PVCs with PVCs as source.....	15
iSCSI CHAP.....	15
CHAP support for PowerMax.....	16
Custom Driver Name (experimental feature).....	16
Install multiple drivers.....	16
Volume expansion.....	17
Raw block support.....	17
CSI PowerMax Reverse Proxy.....	18
Installation.....	18
Prerequisite.....	18
Using Helm installer.....	19
Using Dell CSI Operator.....	19

User-friendly hostnames.....	19
<b>Chapter 3: Test the CSI Driver for Dell EMC PowerMax.....</b>	<b>20</b>
Test the driver.....	20
Volume clone test.....	20
Volume test.....	20
Snapshot test.....	21

# Introduction

This chapter includes the following topics:

## Topics:

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

## 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 1.17, 1.18, and 1.19 and Openshift 4.3 and 4.4 (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 is 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*, *external-snapshotter*, and *resizer* that the Kubernetes community provides.

The Controller plug-in primarily deals with provisioning activities such as, creating volumes, deleting volumes, attaching the volume to a node, and detaching the volume from a node. Also, the plug-in 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 such as, identifying, publishing, and unpublishing a volume to the node.

The Node plug-in identifies the Fibre 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 Fibre 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 versions 1.17, 1.18, and 1.19
- Supports OpenShift 4.3 and 4.4 with Red Hat Enterprise Linux CoreOS and Red Hat Enterprise Linux 7.6
- Requires Unisphere for PowerMax 9.1
- Supports Fibre Channel
- Supports Red Hat Enterprise Linux 7.6/7.7/7.8 host operating system
- Supports CentOS 7.6/7.7/7.8 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
  - Create from Volume
  - Resize
- 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
- Supports Raw Block Volumes
- Supports (Online) Volume Expansion
- Access modes:
  - SINGLE\_NODE\_WRITER
  - SINGLE\_NODE\_READER\_ONLY

# Installation

This chapter includes the following topics:

## Topics:

- [Installation overview](#)
- [Prerequisites](#)
- [Install the CSI Driver for Dell EMC PowerMax using Helm](#)
- [Install the CSI Driver for Dell EMC PowerMax using Operator](#)
- [Update the CSI Driver for Dell EMC PowerMax](#)
- [CSI Driver usage](#)
- [Certificate validation for Unisphere REST API calls](#)
- [Support for Volume Snapshots \(beta\)](#)
- [iSCSI CHAP](#)
- [Custom Driver Name \(experimental feature\)](#)
- [Volume expansion](#)
- [Raw block support](#)
- [CSI PowerMax Reverse Proxy](#)
- [User-friendly hostnames](#)

## 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.

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 tasks:

- Install Kubernetes.

The CSI Driver for Dell EMC PowerMax works with Kubernetes versions 1.17, 1.18, and 1.19.

OR

- Install Openshift.

The CSI Driver for Dell EMC PowerMax works with OpenShift 4.3 and 4.4 with Red Hat Enterprise Linux CoreOS and Red Hat Enterprise Linux 7.6.

- The CSI Driver for Dell EMC PowerMax version 1.4 supports Helm 3 only
- OR

- Install Dell EMC Storage CSI Operator for operator-based installation
- Configure Docker service (Kubernetes only)
- Fibre Channel requirements
- iSCSI requirements
- Linux multipathing requirements
- TLS Secret for CSI PowerMax ReverseProxy (if enabled)

## Fibre Channel requirements

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

- Zoning of the Host Bus Adapters (HBAs) to the Fibre 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 will be published to nodes is high, then configure the maximum number of LUNs for your HBAs on each node. See the appropriate HBA document to configure the 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 array.
- 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](#).

## 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.

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

The CSI Driver for Dell EMC PowerMax version 1.4 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.

## Ports in port group

There are no restrictions around how many ports which can be present in the iSCSI port groups provided to the driver.


The same applies to Fibre Channel where there are no restrictions around the number of FA directors a host HBA can be zoned to. See the best practices for host connectivity to Dell EMC PowerMax to ensure that you have multiple paths to your data volumes.

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

Install the CSI Driver for Dell EMC PowerMax using these steps.

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 operation as described in the [Prerequisites](#) section.
- You have the Helm chart for the driver from the source repository at <https://github.com/dell/csi-powermax>, ready for this procedure.
- The directory `dell-csi-helm-installer` contains the new scripts `csi_install.sh` and `csi_uninstall.sh`.
- You have created a Kubernetes secret with the name `powermax-creds` using the username and password of your Unisphere for PowerMax server.
- If using iSCSI, iSCSI initiators are available on all nodes, including the master and worker nodes.
- 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 continuing.

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


```
git clone https://github.com/dell/csi-powermax.git
```
2. Create the namespace where the driver will be installed. This namespace is usually `powermax`, for example, 

```
kubectl create namespace powermax
```
3. Edit the `helm/secret.yaml` file and replace the values for the username and password parameters. These values can be obtained using base64 encoding as in the following example:
  - ```
echo -n "myusername" | base64
```
  - ```
echo -n "mypassword" | base64
```

Where `myusername` and `mypassword` are credentials for a user with StorageAdmin privileges on your Unisphere for PowerMax server.

4. If you are going to install the new CSI PowerMax ReverseProxy service, create a TLS secret with the name – `csireverseproxy-tls-secret` which holds a SSL certificate and the corresponding private key in the namespace where you are installing the driver.
5. Create the credentials secret `kubectl create -f secret.yaml`
6. Copy the default values.yaml file: `cd dell-csi-helm-installer && cp ../helm/csi-powermax/values.yaml ./my-powermax-settings.yaml`
7. Edit the newly created file and provide values for the following parameters `vi my-powermax-settings.yaml`
  - `unisphere`: This value is used to specify the URL of the Unisphere for PowerMax server. If using the CSI PowerMax Reverse Proxy, leave this value unchanged at `https://127.0.0.1:8443`.
  - `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 maximum 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.
  - `symmetrixID`: This parameter must specify a Dell EMC PowerMax array 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 PowerMax array.
  - `nodeNameTemplate`: This parameter should be used to specify a template which will be used by the driver to create Host/IG names on the PowerMax array. If you wish to use the default naming convention, then leave this value empty.
  - `csireverseproxy`: This section refers to configuration options for CSI PowerMax Reverse Proxy
    - `enabled`: This is a boolean parameter which indicates if CSI PowerMax Reverse Proxy is going to be configured and installed  
 **NOTE:** If not enabled, then there is no requirement to configure any of the following values.
    - `port`: This is used to specify the port number that is used by the NodePort service created by the CSI PowerMax Reverse Proxy installation
    - `primary`: This is a mandatory section.
      - `unisphere`: This must specify the URL of the Unisphere for PowerMax server
      - `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.
      - `certSecret`: The name of the secret in the same namespace containing the CA certificates of the Unisphere server
    - `backup`: This is an optional section and is used to specify the IP address of a Unisphere server which the CSI PowerMax Reverse Proxy can fall back to if the primary Unisphere is unreachable or unresponsive  
 **NOTE:** If you do not want to specify a backup Unisphere server, then remove the backup section from the file
    - `unisphere`: This must specify the IP address of the Unisphere for PowerMax server which manages the arrays being used by the CSI driver
    - `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.
    - `certSecret`: The name of the secret in the same namespace containing the CA certificates of the Unisphere server

8. Install the driver using the sh script. For example, if you are installing the driver in the namespace `powermax`, run the following command: `./csi-install.sh --namespace powermax --values ./my-powermax-settings.yaml`.

 **NOTE:** For detailed instructions on how to run the script, see 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 the iSCSI initiators have been configured on all nodes. You can also skip the verification step by specifying the `--skip-verify-node` option.

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


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](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 PowerMax 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 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

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

### Installation using Helm

A rolling upgrade of the driver from an older version to v1.4 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.

1. Delete any alpha VolumeSnapshot or VolumeSnapshotContent in the cluster.
2. Uninstall the driver using the `csi-uninstall.sh` script by running the command: `./csi-uninstall.sh --namespace <driver-namespace>`

Where `driver-namespace` is the namespace where driver is installed.

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

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

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

5. Install the driver using the new install script as described in the [Install section](#).

**NOTE:** If you are upgrading from a driver version which was installed using Helm2, ensure that you install Helm3 before installing the driver.

**NOTE:** Installation of the CSI PowerMax v1.4 driver is not supported on Kubernetes upstream clusters running version 1.16. You must upgrade your cluster to 1.17 or 1.18 before attempting to install the new version of the driver.

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

## Installation using dell-csi-operator

Follow the detailed instructions in the [dell-csi-operator documentation](#) to upgrade the driver from an older version to v1.4.

## CSI Driver usage

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

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 PersistentVolume (PV) and PersistentVolumeClaim (PVC) resources 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.

**NOTE:** To create additional storage classes and customize the installation, ensure you provide the provisioner value described in [Custom Driver Name](#).

Starting with the 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, only one snapshot class is required, that is, the one created during the installation.

You can create VolumeSnapshot using this snapshot class. These VolumeSnapshot names can further be used as sources to create PersistentVolumeClaim (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.

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.

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.

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 earlier releases, if the install script did not find the secret, it created an empty secret with the same name. From the 1.2.0 release, the secret volume has been made optional. The install script no longer attempts to create an empty secret.

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:

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

 **NOTE:** 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`.


## Support for Volume Snapshots (beta)

The 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 with version 1.4, the CSI PowerMax driver supports beta snapshots. Previous versions of the driver -1.2, 1.3, 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 (because it is based on upstream Kubernetes v1.16). When using the `dell-csi-operator` to install the CSI PowerMax driver on an OpenShift cluster running v4.3, the external-snapshotter sidecar will not be installed.

## 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`, a helper script is 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 has been split into two controllers, a common snapshot controller and a CSI external-snapshotter sidecar.

The common snapshot controller must be installed only once in the cluster irrespective of 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`, the manifests are available on [GitHub](#).

**NOTE:** The manifests available on the GitHub location install v2.0.1 of the [quay.io/k8scsi/snapshot-controller:v2.0.1](#). Dell EMC recommends using the v2.1.1 image of the [quay.io/k8scsi/csi-snapshotter:v2.1.1](#).

**NOTE:** The CSI external-snapshotter sidecar is still installed with the driver and does not involve any extra configuration.

## Upgrade from v1alpha1 to v1beta1

There is no upgrade path available from v1alpha1 to v1beta1 snapshot APIs.

If you have created any existing snapshots using the v1alpha1 APIs, you must delete them before upgrading to beta snapshots.

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 the external-snapshotter documentation on [GitHub](#).

## Volume Snapshot Class

During the installation of CSI PowerMax 1.4 driver, a Volume Snapshot Class is created using the new v1beta1 snapshot APIs.

This is the only Volume Snapshot Class you will need and there is no need to create any other Volume Snapshot Class. This is the manifest for the Volume Snapshot Class created during installation (using the default driver name):

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

## Creating Volume Snapshots

This is a sample manifest for creating a Volume Snapshot using the v1beta1 snapshot APIs:

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshot
metadata:
  name: pmax-snapshot-demo
  namespace: test
spec:
  volumeSnapshotClassName: powermax-snapclass
  source:
    persistentVolumeClaimName: pmax-pvc-demo
```

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

This 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 VolumeSnapshots as source

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

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pmax-restore-pvc-demo
  namespace: test
spec:
  storageClassName: powermax
  dataSource:
    name: pmax-snapshot-demo
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 8Gi
```

## Creating PVCs with PVCs as source

This is a sample manifest for creating a PVC with another PVC as a source:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pmax-clone-pvc-demo
  namespace: test
spec:
  storageClassName: powermax
  dataSource:
    name: pmax-pvc-demo
    kind: PersistentVolumeClaim
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 8Gi
```

## iSCSI CHAP

With version 1.3.0, support has been added for unidirectional Challenge Handshake Authentication Protocol (CHAP) for iSCSI.

To enable CHAP authentication:


1. Create secret `powermax-creds` with the key `chapsecret` set to the iSCSI CHAP secret. If the secret exists, delete and re-create the secret with this newly added key.
2. Set the parameter `enableCHAP` in `my-powermax-settings.yaml` to `true`.

The driver uses the provided `chapsecret` to configure the iSCSI node database on each node with iSCSI access.

When the driver is installed and all the node plug-ins have initialized successfully, the storage administrator must enable CHAP authentication using the following Solutions Enabler (SYMCLI) commands:

```
symaccess -sid <symid> -iscsi <host iqn> set chap -cred <host IQN> -secret <CHAP secret>
```

Where <host IQN> is the name of the iSCSI initiator of a host IQN, and <CHAP secret> is the chapsecret that is used at the time of the installation of the driver.

 **NOTE:** The host IQN is also used as the username when setting up the CHAP credentials.

## CHAP support for PowerMax

With unidirectional CHAP, the PowerMax array challenges the host initiator during the initial link negotiation process and expects to receive a valid credential and CHAP secret in response.

When challenged, the host initiator transmits a CHAP credential and CHAP secret to the storage array. The storage array looks for this credential and CHAP secret which stored in the host initiator initiator group. When a positive authentication occurs, the PowerMax array sends an acceptance message to the host. However, if the PowerMax array fails to find any record of the credential/secret pair, it sends a rejection message, and the link is closed.

## Custom Driver Name (experimental feature)

With version 1.3.0 of the driver, a custom name can be assigned to the driver at the time of installation. This enables installation of the CSI driver in a different namespace and installation of multiple CSI drivers for Dell EMC PowerMax in the same Kubernetes/OpenShift cluster.

To use this experimental feature, set the following values under `customDriverName` in `my-powermax-settings.yaml`.


- Value: Set this to the custom name of the driver.
- Enabled: Set this to true in case you want to enable this feature.

The driver helm chart installation uses the values above to:


- Configure the driver name which is used for communication with other Kubernetes components.
- Configure the provisioner value in the storage class template.
- Configure the snapshotter value in the snapshot class template.

If enabled, the driver name is in the following format: `<namespace>.<driver name>.dell EMC.com`

For example, if the driver name is set to `driver` and it is installed in the namespace `powermax`, then the name that is used for the driver (and the provisioner/snapshotter) is `powermax.driver.dell EMC.com`

 **NOTE:** If not enabled, the name is set to `csi-powermax.dell EMC.com` by default (without any namespace prefix).

## Install multiple drivers

 **NOTE:** This is an experimental feature and should be used with extreme caution after consulting with Dell EMC Support.

To install multiple CSI Drivers for Dell EMC PowerMax in a single Kubernetes cluster, you can take advantage of the custom driver name feature. There are a few important restrictions which should be strictly adhered to:

- Only one driver can be installed in a single namespace
- Different drivers should not connect to a single Unisphere server
- Different drivers should not be used to manage a single PowerMax array
- Storage class and snapshot class names must be unique across installations

To install multiple CSI drivers, follow these steps:

1. Create (or use) a new namespace.
2. Ensure that all the pre-requisites are met:
  - `powermax-creds` secret is created in this namespace
  - (Optional) `powermax-certs` secret is created in this namespace
3. Update `my-powermax-settings.yaml` with the required values.
4. Run the `csi-install.sh` script to install the driver.



# Volume expansion

Starting with v1.4, the CSI PowerMax driver supports expansion of Persistent Volumes (PVs). This expansion is done online, that is, when the PVC is attached to any node.

To use this feature, the storage class that is used to create the PVC must 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.


If you are creating more storage classes, ensure that this attribute is set to `true` to expand any PVs created using these new storage classes.

This is a sample manifest for a storage class which allows for Volume Expansion.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: powermax-expand-sc
  annotations:
    storageclass.beta.kubernetes.io/is-default-class: false
provisioner: csi-powermax.dellemc.com
reclaimPolicy: Delete
allowVolumeExpansion: true #Set this attribute to true if you plan to expand any PVCs
created using this storage class
parameters:
  SYMID: "00000000000001"
  SRP: "DEFAULT_SRP"
  ServiceLevel: "Bronze"
```

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 - `pmax-pvc-demo` of size 5Gi, then you can resize it to 10 Gi by updating the PVC.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pmax-pvc-demo
  namespace: test
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 10Gi #Updated size from 5Gi to 10Gi
  storageClassName: powermax-expand-sc
```

 **NOTE:** The Kubernetes Volume Expansion feature can only be used to increase the size of volume, it cannot be used to shrink a volume.

# Raw block support

The PowerMax driver version 1.4 adds support for Raw Block volumes.

Raw Block volumes are created using the `volumeDevices` list in the Pod template spec with each entry accessing a `volumeClaimTemplate` specifying a `volumeMode: Block`. An example configuration is outlined here:

```
kind: StatefulSet
apiVersion: apps/v1
metadata:
  name: powermaxtest
  namespace: {{ .Values.namespace }}
spec:
  ...
  spec:
    ...
    containers:
      - name: test
```

```

    ...
    volumeDevices:
      - devicePath: "/dev/data0"
        name: pvol0
  volumeClaimTemplates:
  - metadata:
    name: pvol0
    spec:
      accessModes:
      - ReadWriteOnce
      volumeMode: Block
      storageClassName: powermax
      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 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 the website: [Kubernetes](https://kubernetes.io)

## CSI PowerMax Reverse Proxy

To get the maximum performance out of the CSI driver for PowerMax and Unisphere forPowerMax REST APIs, starting with v1.4 of the driver, you can deploy the optional CSI PowerMax Reverse Proxy application.

CSI PowerMax Reverse Proxy is a (go) HTTPS server which acts as a reverse proxy for the Unisphere forPowerMax RESTAPI interface. Any RESTAPI request sent from the driver to the reverse proxy is forwarded to the Unisphere server and the response is routed back to the driver.

The Reverse Proxy helps regulate the maximum number of requests which can be sent to the Unisphere RESTAPI at a given time across all driver controller and node Pods. This helps with better queuing of CSI requests and performance of the CSI PowerMax driver.

Optionally you can specify an alternate (backup) Unisphere server and if the primary Unisphere server is not reachable or does not respond, the proxy will redirect the calls to this alternate Unisphere.

## Installation

CSI PowerMax Reverse Proxy is installed as a Kubernetes deployment in the same namespace as the driver.

It is also configured as a Kubernetes "NodePort" service. If the CSI PowerMax driver has been configured to use this service, then it will connect to the IP address and port exposed by the Kubernetes service instead of directly connecting to the Unisphere server.

## Prerequisite

CSI PowerMax Reverse Proxy is a HTTPS server and has to be configured with an SSL certificate and a private key.

The certificate and key are provided to the proxy via a Kubernetes TLS secret (in the same namespace). The SSL certificate must be a X.509 certificate encoded in PEM format. The certificates can be obtained via a Certificate Authority or can be self-signed and generated by a tool such as openssl.

Here is an example to generate a private key and use that to sign an SSL certificate using the openssl tool:

```

openssl genrsa -out tls.key 2048
openssl req -new -x509 -sha256 -key tls.key -out tls.crt -days 3650
kubectl create secret -n <namespace> tls revproxy-certs --cert=tls.crt --key=tls.key
kubectl create secret -n <namespace> tls csirevproxy-tls-secret --cert=tls.crt --
key=tls.key

```

## Using Helm installer

A new section, `csireverseproxy`, in the `my-powermax-settings.yaml` file can be used to deploy and configure the CSI PowerMax Reverse Proxy.

The new Helm chart is configured as a sub chart for the CSI PowerMax helm chart. If it is enabled (using the `enabled` parameter in the `csireverseproxy` section of the `my-powermax-settings.yaml` file), the install script automatically installs the CSI PowerMax Reverse Proxy and configures the CSI PowerMax driver to use this service.

## Using Dell CSI Operator

Starting with the v1.1.0 release of the Dell CSI Operator, a new Custom Resource Definition can be used to install CSI PowerMax Reverse Proxy.


This Custom Resource has to be created in the same namespace as the CSI PowerMax driver and it has to be created before the driver Custom Resource. To use the service, the driver Custom Resource manifest must be configured with the service name "powermax-reverseproxy". For complete installation instructions for the CSI PowerMax driver and the CSI PowerMax Reverse Proxy, see the Dell CSI Operator documentation.

## User-friendly hostnames

Users can set a value for the `nodeNameTemplate` in `my-powermax-settings.yaml` during the installation of the driver so that the driver can use this value to decide the name format of hosts to create or update in the PowerMax array for the nodes in a Kubernetes cluster. The `hostname` value in `nodeNameTemplate` should always be contained between two '%' characters. String prefixing first '%' and string suffixing second '%' is used as is before and after every node identifier.

Also, there is a new setting, `modifyHostName`, which could be set to `true` if you want the driver to rename the existing Hosts/IG for the host initiators on the PowerMax array. The new name uses the default naming convention (`csi-<ClusterPrefix>-<HostName>*`) or the `nodeNameTemplate` if it was specified.

For example, if `nodeNameTemplate` is `abc-%foo%-hostname` and `nodename` is `worker1`, then the host ID is created or updated as `abc-worker1-hostname`. This change will happen for all nodes in a cluster with the respective node name.

 **NOTE:** `nodeNameTemplate` can contain alphanumeric characters [a - z, A - Z, 0 - 9], '-' and '\_', other characters are not allowed.

# Test the CSI Driver for Dell EMC PowerMax

This chapter includes the following topics:


## Topics:

- [Test the driver](#)

## Test the driver

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 fully supported in Openshift 4.3.

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 these tests:

- [Volume clone test](#)
- [Volume test](#)
- [Snapshot test](#)

## Volume clone test

Procedure to perform volume clone test.

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 `volumeclonetest.sh` script.
3. Run the `volumeclonetest.sh` script using the following command:  


```
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 creates a file on one of the PVCs and calculates its checksum.
- After that, it uses that PVC as the datasource to create a new PVC and mounts it on the same container. It checks if the file that existed in the source PVC also exists in the new PVC, calculates its checksum and compares it to the checksum previously calculated.
- Finally, it cleans up all the resources that are created as part of the test.

## Volume test

Procedure to perform a volume test.

 **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, 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.

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.

**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`

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 lists the contents of the source and the target PVCs.
- Finally, it cleans up all the resources that are created as part of the test.