

CSI Driver for Dell EMC PowerMax

Version 1.0

Product Guide

302-005-904

Rev 01

July 2019

Copyright © 2019 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

CONTENTS

Chapter 1	Introduction	5
	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
Chapter 2	Installation	9
	Installation overview.....	10
	Prerequisites.....	10
	Set up the iSCSI Initiator.....	10
	Enable Kubernetes feature gates.....	11
	Configure Docker service.....	12
	Install the Helm and Tiller package manager.....	12
	Install the CSI Driver for Dell EMC PowerMax.....	13
	CSI Driver usage.....	15
	Controller Plug-in query commands.....	16
	Node plug-in query command.....	16
	Certificate validation for Unisphere REST API calls.....	16
Chapter 3	Test the CSI Driver for Dell EMC PowerMax	19
	Test the CSI Driver for Dell EMC PowerMax.....	20

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 v1.0. It is compatible with Kubernetes versions 1.13.1, 1.13.2, and 1.13.3. with the Red Hat Enterprise Linux 7.6 host operating system. Use Unisphere for PowerMax 9.0 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* and *attacher*, 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. The CSI Driver for Dell EMC PowerMax automates the creation and deletion of Storage Groups (SGs) and Masking Views that are required for these tasks.

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 communicates with the Kubelet to perform tasks like identifying, publishing, and unpublishing a volume to the node where the plug-in is running.

The Node plug-in identifies all the iSCSI Qualified Names (IQN) present on the node and creates a *Host* using these initiators on the PowerMax array. The Controller Plug-in uses the hosts to create Masking Views for nodes.

Features of the CSI Driver for Dell EMC PowerMax

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

- Supports CSI 1.0
- Supports Kubernetes version 1.13.1, 1.13.2, and 1.13.3
- Supports Unisphere for PowerMax 9.0

- Supports Red Hat Enterprise Linux 7.6 host operating system
- Supports PowerMax - 5978.221.221 (ELM SR)
- Persistent Volume (PV) capabilities:
 - Create
 - Delete
- 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 charts installer
- Access modes:
 - SINGLE_NODE_WRITER
 - SINGLE_NODE_READER_ONLY

CHAPTER 2

Installation

This chapter includes the following topics:

• Installation overview	10
• Prerequisites	10
• Install the CSI Driver for Dell EMC PowerMax	13
• CSI Driver usage	15
• Certificate validation for Unisphere REST API calls	16

Installation overview

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

The CSI Driver for Dell EMC PowerMax is deployed in the Kubernetes platform using Helm charts. 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.

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

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 versions 1.13.1, 1.13.2, and 1.13.3.
- [Set up the iSCSI Initiator](#)
- [Enable Kubernetes feature gates](#)
- [Configure Docker service](#)
- [Install the Helm and Tiller package manager](#)

Set up the iSCSI Initiator

The CSI Driver for Dell EMC PowerMax v1.0 supports iSCSI connectivity.

Set up the iSCSI initiators as follows:

- Make sure that the iSCSI initiators are available on both Master and Minions 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.
- Make sure 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

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

About this task

Note: You may need to enable other feature gates for different Kubernetes versions and distributions. The feature gates that are described in this section are applicable for with Kubernetes 1.13.1, 1.13.2, and 1.13.3.

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

- KubeletPluginsWatcher
- CSINodeInfo
- CSIDriverRegistry
- BlockVolume
- CSIBlockVolume

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:

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

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

```
/etc/kubernetes/manifests/kube-apiserver.yaml - --
featuregates=VolumeSnapshotDataSource=true,KubeletPluginsWatch
er=true,CSIN
odeInfo=true,CSIDriverRegistry=true,BlockVolume=true,CSIBlockV
olume=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 - --
featuregates=VolumeSnapshotDataSource=true,KubeletPluginsWatch
er=true,CSIN
odeInfo=true,CSIDriverRegistry=true,BlockVolume=true,CSIBlockV
olume=true
```

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

```
/etc/kubernetes/manifests/kube-scheduler.yaml - --
featuregates=VolumeSnapshotDataSource=true,KubeletPluginsWatch
er=true,CSIN
odeInfo=true,CSIDriverRegistry=true,BlockVolume=true,CSIBlockV
olume=true
```

- 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 --
allowprivileged=true --
featuregates=VolumeSnapshotDataSource=true,KubeletPluginsWatch
er=true,CSIN
odeInfo=true,CSIDriverRegistry=true,BlockVolume=true,CSIBlockV
olume=true"
```

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

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

Configure Docker service

This topic gives the procedure to configure docker service. 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

- 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
```

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

Install the Helm and Tiller package manager

Install the Helm and Tiller package manager on the master node before you install the CSI Driver for Dell EMC PowerMax. The recommended Helm and Tiller package version is 2.13.1.

Procedure

- Run `curl https://raw.githubusercontent.com/helm/helm/master/scripts/get > get_helm.sh`.

2. Run `chmod 700 get_helm.sh`.
3. Run `./get_helm.sh`.
4. Run `helm init`.
5. Run `helm version` to test the helm installation.
6. Set up a service account for Tiller:
 - a. Create a yaml file named *rbac-config.yaml* and add the following information to the file:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: tiller
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: tiller
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: tiller
  namespace: kube-system
```

- b. Run `kubectl create -f rbac-config.yaml` to create the service account.
7. Run `helm init --upgrade --service-account tiller` to apply the service account to Tiller.


Install the CSI Driver for Dell EMC PowerMax

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.
- You have the 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](#).

- The iSCSI initiators are available on all nodes, including the master and minion 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 and copy the values file for driver configuration.
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 are managed by the driver.
 - `arrayWhitelist`: This parameter holds a list of comma-separated array IDs. If this parameter remains empty, the driver manages all the arrays that were 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 can remain unchanged in most cases.
 - `symmetrixID`: This parameter must specify the Dell EMC PowerMax arrays that are being managed by the driver. This value is used to create a default Storage class.
 - `storageResourcePool`: This parameter must mention one of the SRPs on the PowerMax array that is specified by the symmetrixID. 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.
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 optionally using base64 encoding as described in the following example:

- `echo -n "myusername" | base64`
 - `echo -n "mypassword" | base64`
6. Run `kubect1 create namespace powermax` to create the *PowerMax* namespace.
 7. Run `kubect1 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 will be 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:

```
kubect1 get pods -n powermax
```

You can also test the installation of your driver.

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.

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 daemonset powermax-node -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 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. If the install script does not find the secret, it creates an empty secret with the same name.

About this task

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

If the `powerMaxInsecure` is set to *false*, then the secret *powermax-certs* must contain the CA certificate for Unisphere. If this secret is an *empty* secret, then the validation of the certificate fails, and the driver fails to start.

If the `powerMaxInsecure` parameter is set to *false* and a previous installation attempt created the 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> </dev/null 2>/dev/null | openssl x509 -outform PEM > ca_cert.pem`.
2. To create the secret, run `kubectl create secret generic powermax-certs --from-file=ca_cert.pem -n powermax`.

CHAPTER 3

Test the CSI Driver for Dell EMC PowerMax

This chapter includes the following topics:

- [Test the CSI Driver for Dell EMC PowerMax](#).....20

Test the CSI Driver for Dell EMC PowerMax

This topic provides information about testing the CSI Driver for Dell EMC PowerMax.

About this task

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 procedure:

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 script 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.