

CSI Driver for Dell EMC Unity

Version 1.0

Product Guide

November 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	7
	Node Plug-in.....	7
Chapter 2	Install the CSI Driver for Dell EMC Unity	9
	Prerequisites.....	10
	Enable Kubernetes feature gates.....	10
	Configure Docker service.....	11
	Install the Helm and Tiller package manager.....	11
	Certificate validation for Unisphere REST API calls.....	12
	Install CSI Driver for Dell EMC Unity.....	13
Chapter 3	Test the CSI Driver for Dell EMC Unity	17
	Test deploying a simple pod with Dell EMC Unity storage.....	18

CHAPTER 1

Introduction

This chapter contains the following section:

- [Product overview](#) 6
- [CSI Driver components](#) 6

Product overview

The CSI Driver for Dell EMC Unity implements an interface between CSI enabled Container Orchestrator(CO) and Unity Storage Array. It allows you to dynamically provide Unity volumes and attach them to workloads.

The CSI Driver for Dell EMC Unity conforms to the CSI spec 1.1. This release of the CSI Driver for Dell EMC Unity supports Kubernetes 1.14. To learn more about the CSI specification see: <https://github.com/container-storage-interface/spec>.

Features of the CSI Driver for Dell EMC Unity

The CSI Driver for Dell EMC Unity driver supports the following features:

- Persistent volume (PV) capabilities:
 - Create
 - List
 - Delete
 - Mount
 - Unmount
- Supports mounting volume as file system
- Supports snapshot creation
- Supports static volumes and dynamic volumes
- Supports Bare Metal machine type
- Supports *SINGLE_NODE_WRITER* access mode
- Supports CentOS 7.6 as host operating system
- Supports Red Hat Enterprise Linux 7.5 and 7.6 as host operating system
- Supports Kubernetes version 1.14
- Supports Unity OE 5.0
- Supports FC Protocol

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.

CSI Driver components

This section describes the components of the CSI Driver for Dell EMC Unity. The CSI Driver for Dell EMC Unity 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 which gets scheduled on any node (not necessarily the master).

About this task

This pod contains the CSI Driver for Dell EMC Unity container along with a few side-car containers like *provisioner* and *attacher*. The Kubernetes community provides these side-car containers.

Similarly, logs for the provisioner and attacher side-cars can be obtained by specifying the container names.

The Controller plug-in primarily deals with provisioning activities like creating volumes, deleting volumes, attaching the volume to a node, detaching the volume from a node.

Perform the procedure described in this section to view the details of the 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 unity
$ kubectl describe statefulset unity-controller-0 -n unity
```

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

```
$ kubectl logs unity-controller-0 -c driver -n unity
```

Node Plug-in

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

About this task

The Node plug-in primarily communicates with Kubelet to carry out tasks like identifying the node, publishing a volume to the node, and unpublishing volume to the node where the plug-in is running.

The Node plug-in, as part of its startup identifies all the IQNs present on the node and creates a *Hosts* using these initiators on the Unity array. These *Hosts* are later used by the Controller Plug-in to export volumes to the node.

Perform the procedure described in this section to view the details of the DaemonSet and check logs for the Node Plug-in.

Procedure

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

```
$ kubectl get daemonset -n unity
$ kubectl describe daemonset unity-node -n unity
```

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

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


CHAPTER 2

Install the CSI Driver for Dell EMC Unity

This chapter contains the following sections:

- [Prerequisites](#).....10
- [Install CSI Driver for Dell EMC Unity](#)..... 13

Prerequisites

Before you install the CSI Driver for Dell EMC Unity, ensure that the requirements that are mentioned in this section are installed and configured.

Requirements

- This document assumes that Kubernetes has been installed using *kubeadm*. The CSI Driver for Dell EMC Unity works with Kubernetes version 1.14. The Red Hat Enterprise Linux 7.6 and 7.5, and CentOS 7.6 host operating systems are supported.
- Ensure that the Unity array being used is zoned with the nodes.
- CSI Driver for Dell EMC Unity supports only FC protocol. Ensure that native multipath has been installed on the nodes.

 **Note:** PowerPath is not supported.

- [Enable the Kubernetes feature gates](#)
- [Configure Docker service](#)
- [Install Helm and Tiller package manager](#)
- [Certificate validation for Unisphere REST API calls](#)

Enable Kubernetes feature gates

The Kubernetes feature gates must be enabled before installing the CSI Driver for Dell EMC Unity.

About this task

The [Feature Gates](#) section of Kubernetes home page lists the Kubernetes feature gates. The following Kubernetes feature gates must be enabled:

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

Procedure

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

```
/var/lib/kubelet/config.yaml
VolumeSnapshotDataSource: true
KubeletPluginsWatcher: true
CSINodeInfo: true
CSIDriverRegistry: true
BlockVolume: true
CSIBlockVolume: true
ExpandCSIVolumes: true
```

2. On the master, set the feature gate settings of the *kube-apiserver.yaml*, *kube-controllermanager.yaml*, and *kube-scheduler.yaml* files as follows:

```
/etc/kubernetes/manifests/kube-apiserver.yaml - --feature-
gates=VolumeSnapshotDataSource=true,KubeletPluginsWatcher=true,CSINodeIn
```

```
fo=true,CSIDriverRegistry=true,BlockVolume=true,CSIBlockVolume=true,ExpandCSIVolumes=true
```

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

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

3. On each node including the master node, edit the variable ***KUBELET_KUBECONFIG_ARGS*** of the `/usr/lib/systemd/system/kubelet.service.d/10-kubeadm.conf` file as follows:

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

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

Configure Docker service

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

Procedure

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

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

2. 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 Unity.

Procedure

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

2. Run the `chmod 700 get_helm.sh` command.
3. Run the `./get_helm.sh` command.
4. Run the `helm init` command.
5. Run the `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.

Certificate validation for Unisphere REST API calls

This topic provides details about setting up the certificate validation for the CSI driver for Dell EMC Unity.

Before you begin

As part of the CSI driver installation, the CSI driver requires a secret with the name `unity-certs` present in the namespace `unity`. If `unity` namespace is not already present, you must create the namespace using the command `kubectl create namespace unity`. 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 `unityInsecure` which determines if the driver performs client-side verification of the Unisphere certificates. The `unityInsecure` parameter is set to `true` by default, and the driver does not verify the Unisphere certificates.

If the `unityInsecure` is set to `false`, then the secret `unity-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 `unityInsecure` 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 the `openssl s_client -showcerts -connect <Unisphere IP:Port> </dev/null 2>/dev/null | openssl x509 -outform PEM > ca_cert.pem` command.
2. To create the secret, run the `kubectl create secret generic unity-certs --from-file=ca_cert.pem -n unity` command.

Install CSI Driver for Dell EMC Unity

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

Before you begin

- You must have the downloaded files, including the Helm chart from github.com/dell/csi-unity, using the following command:

```
/home/test# git clone https://github.com/dell/csi-unity
```

- In the top-level helm directory, there should be two shell scripts, `install.unity` and `uninstall.unity`. These scripts perform the preoperations and postoperations that cannot be performed in the helm chart; such as creating Custom Resource Definitions (CRDs), when needed.
- Create a storage pool (if it is not already created) and provide the `pool_id` in the `myvalues.yaml` file.

Procedure

1. Collect information from the Unity System, such as IP address, username, and password. Make a note of the value for these parameters as they must be entered in the `myvalues.yaml` file.
2. Copy the `csi-unity/values.yaml` into a file in the same directory as the `install.unity` named `myvalues.yaml`, to customize settings for installation.
3. Edit `myvalues.yaml` to set the following parameters for your installation:

Detailed information can be found in the `values.yaml` file at `helm/csi-unity/values.yaml` in this repository. The following table lists the primary configurable parameters of the Unity driver chart and their default values:

Parameter	Description	Required	Default
<code>systemName</code>	Name of the Unity system	false	unity
<code>restGateway</code>	REST API gateway HTTPS endpoint Unity system	true	
<code>storagePool</code>	Unity Storage Pool ID to use within the Kubernetes storage class	true	
<code>unityUsername</code>	Username for accessing unity system <base64 encoded string>	true	

Parameter	Description	Required	Default
unityPassword	Password for accessing unity system <base64 encoded string>	true	
volumeNamePrefix	String that is prefixed to any volumes that the driver creates	false	csivol
storageClass.name	Name of the storage class that will be defined	false	unity
storageClass.isDefault	If this storage class should be made the default storage class	false	true
storageClass.reclaimPolicy	What should happen when a volume is deleted	false	delete
Storage Class parameters: Following parameters are not present in values.yaml			
FsType	To set File system type. Possible values are ext3, ext4, or xfs.	false	ext4
volumeThinProvisioned	To set volume thinProvisioned	false	true
isVolumeDataReductionEnabled	To set volume data reduction	false	false
volumeTieringPolicy	To set volume tiering policy	false	0
hostIOLimitName	To set Unity host I/O limit	false	""
Snapshot Class parameters: Following parameters are not present in values.yaml			
snapshotRetentionDuration	To set snapshot retention duration. Format:"1:23:52:50" (Which is: days:hours:minutes:seconds)	false	""

4. Use the following commands to convert username/password to base64 encoded string:

- `echo -n 'admin' | base64`
- `echo -n 'password' | base64`

The following is an example of the edited *myvalues.yaml* file:

```
csiDebug: "true"
imagePullPolicy: Always
storagePool: pool_1
restGateway: "https://<IP of Unity System>"
unityUsername: <Base 64 Encoded String>
```

```

unityPassword: <Base 64 Encoded String>
systemName: unity
volumeNamePrefix: customcsivol
storageClass:
  name: mystorageclass
  isDefault: false
  reclaimPolicy: Retain
FsType: xfs
volumethinProvisioned: "true"
isVolumeDataReductionEnabled: "true"
volumetieringPolicy: "0"
hostIOLimitName: "<Value from Unity array>"
snapshotRetentionDuration: "1:0:0:0"
isSnapshotReadOnly: false
images:
  driver: <docker image>

```

5. Run the `sh install.unity` command to proceed with the installation.

A successful installation should emit messages that look similar to the following samples:

```

sh install.unity
Kubernetes version v1.14.2
Kubernetes master nodes: 10.247.97.151
Kubernetes minion nodes:
Verifying the feature gates.
NAME: unity
LAST DEPLOYED: Wed Aug 14 01:23:35 2019
NAMESPACE: unity
STATUS: DEPLOYED
[....]
NAME                READY   STATUS    RESTARTS   AGE
unity-controller-0   4/4     Running   0           21s
unity-node-kzv49     2/2     Running   0           21s
CSIDrivers:
No resources found.
CSINodeInfos:
No resources found.
StorageClasses:
NAME                PROVISIONER   AGE
unity (default)     csi-unity     21s

```

Results

At the end of the script, the `kubectl get pods -n unity` is called to *GET* the status of the pods and you will see the following:

- `unity-controller-0` with 4/4 containers ready, and status is displayed as `Running`.
- Agent pods with 2/2 containers and their statuses are displayed as `Running`.

Finally, the script lists the created *storageclasses* such as, *unity*. Other storage classes can be created for different combinations of file system types and Unity storage pools. The script also creates *volumesnapshotclasses* such as, *unity-snapclass* and other snapshots classes.

CHAPTER 3

Test the CSI Driver for Dell EMC Unity

This chapter contains the following sections:

- [Test deploying a simple pod with Dell EMC Unity storage](#).....18

Test deploying a simple pod with Dell EMC Unity storage

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

Before you begin

The host initiators must be zoned to Unity before you perform this procedure.

Procedure

1. Verify Unity system for Host.

After helm deployment, the *CSI Driver for Node* will create new host or hosts in the Unity system depending on the number of nodes in the kubernetes cluster. Verify the Unity system for new Hosts and Initiators.


2. To create a volume, create a *pvc.yaml* file with the following content:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: testvolclaim1
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
  storageClassName: unity
```

3. Execute the following command to create volume:

```
kubectl create -f $PWD/pvc.yaml
```

After executing this command, the PVC will be created in the default namespace, and the user can see the pvc by executing the `kubectl get pvc` command.

 **Note:** Verify the Unity system for the new volume.

4. Attach the volume to a host, create a new application (Pod) and use the created PVC in the Pod. This is explained using the Nginx application. Create the *nginx.yaml* with the following content:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pv-pod
spec:
  containers:
    - name: task-pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
        claimName: testvolclaim1
```

- Execute the following command to mount the volume to kubernetes node:

```
kubectl create -f $PWD/nginx.yaml
```

After executing the above command, new nginx pod will be successfully created and started in the default namespace.

Note: Verify the Unity system for volume to be attached to the Host where the nginx container is running.

- Create a snapshot of the volume in the container using VolumeSnapshot objects defined in the *snap.yaml* file. The following are the contents of snap.yaml file:

```
apiVersion: snapshot.storage.k8s.io/v1alpha1
kind: VolumeSnapshot
metadata:
  name: testvolclaim1-snap1
spec:
  snapshotClassName: unity-snapclass
  source:
    name: testvolclaim1
    kind: PersistentVolumeClaim
```

- Execute the following command to create snapshot:

```
kubectl create -f $PWD/snap.yaml
```

The spec.source section contains the volume that will be snapped in the default namespace. Verify the Unity system for new snapshot under the lun section.

Note:

- You can see the snapshots using the `kubectl get volumesnapshot` command.
- Note that this VolumeSnapshot class has a reference to a snapshotClassName:unity-snapclass. The CSI Driver for Unity installation creates this class as its default snapshot class.
- You can see the definition using the `kubectl get volumesnapshotclasses unity-snapclass -o yaml` command.

- Execute the following commands to delete the snapshot:

```
kubectl delete volumesnapshot testvolclaim1-snap1
```

- Delete the nginx application to unattach the volume from host, using the `kubectl delete -f nginx.yaml` command.

- To delete the volume, run the following commands:

- `kubectl delete pvc testvolclaim1`
- `kubectl get pvc`

