

CSI Driver for Dell EMC Isilon

Product Guide

Version 1.2

Contents

1 Introduction.....	4
Product overview.....	4
Features of the CSI driver for Dell EMC Isilon	4
CSI Driver components.....	6
Controller Plug-in.....	6
Node Plug-in.....	6
2 Installation.....	7
Installation overview.....	7
Prerequisites.....	7
Enable Kubernetes feature gates.....	7
Configure Docker service.....	8
Install either Helm 3 or Helm 2 with Tiller package manager.....	9
Certificate validation for OneFS REST API calls.....	9
Install CSI driver for Dell EMC Isilon.....	10
Install the CSI driver for Dell EMC Isilon using Helm.....	10
Install CSI Isilon driver using Dell EMC Storage CSI Operator.....	13
Upgrade CSI driver for Dell EMC Isilon	14
Upgrade CSI driver for Dell EMC Isilon using Helm 2.....	14
Migrate from Helm 2 to Helm 3 and upgrade CSI driver for Dell EMC Isilon.....	15
Test deploying a simple pod with Isilon storage.....	15
CSI Driver Usage.....	16
Controller Plug-in query commands.....	16
Node Plug-in query commands.....	17
3 Test the CSI driver for Dell EMC Isilon.....	18
Test the CSI driver for Dell EMC Isilon.....	18
Test creating snapshots.....	18
Test restoring from a snapshot.....	19

Notes, cautions, and warnings

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

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

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

Introduction

This chapter includes the following topics:

Topics:

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

Product overview

The CSI driver for Dell EMC Isilon implements an interface between CSI enabled Container Orchestrator (CO) and Dell EMC Isilon storage system.

The CSI driver for Dell EMC Isilon conforms to CSI specification v1.1. This release of CSI driver for Dell EMC Isilon supports Kubernetes 1.14 and 1.16 and OpenShift 4.2 and 4.3.

To learn more about the CSI specification see: <https://github.com/container-storage-interface/spec/tree/release-1.1>.

Features of the CSI driver for Dell EMC Isilon

The CSI driver for Dell EMC Isilon has the following features:

- Persistent Volume (PV) capabilities:
 - Create
 - List
 - Delete
 - Mount
 - Unmount
- Supports mounting volumes as NFS export
- Supports snapshot capabilities:
 - Create snapshot
 - Create volume from snapshot
 - Delete
- Supports static and dynamic provisioning of volumes and volume expansion
- Supports the following access modes:
 - SINGLE_NODE_WRITER
 - MULTI_NODE_READER_ONLY
 - MULTI_NODE_MULTI_WRITER
- Conforms to CSI 1.1 specification
- Supports Kubernetes 1.14 and 1.16
- Supports OpenShift 4.2 and 4.3
- Supports Isilon OneFS 8.1, 8.2, and 9.0
- Supports Red Hat Enterprise Linux 7.6 and 7.7 as host operating system
- Supports HELM charts (Helm2, Helm3) installer
- Supports installation in OpenShift environments by using Dell CSI Operator

NOTE: Snapshot capabilities are not supported on OpenShift.

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

Use SmartQuotas to Limit Storage Consumption

Learn how CSI driver for Dell EMC Isilon handles capacity limiting using *SmartQuotas* feature.

About this task

To indicate whether to use the *SmartQuotas* feature, in the helm chart the user can specify the boolean value *enableQuota* in *myvalues.yaml*.

For example, when a user creates a *PersistentVolumeClaim* using the following *yaml* file:

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

The following is how CSI Isilon handles the *8Gi* capacity requested:

Steps

1. When *enableQuota* is *false*, the CSI Isilon ignores the *8Gi* request.
2. When *enableQuota* is *true*:
 - If *SmartQuotas* is not activated, the CSI Isilon ignores the *8Gi* request and continue.
 - If *SmartQuotas* is activated, the CSI Isilon attempts to create a *8Gi* quota on the volume.
 - If creating quota is successful, the call continues.
 - If creating quota fails, the call reports an error and the transaction is rolled back by removing the volume (directory) that was created. As a result, no PVC is created.

Use Volume Expansion in CSI driver for Dell EMC Isilon

Volume Expansion is implemented in CSI driver for Dell EMC Isilon as part of *ControllerExpandVolume* RPC call and supports (K8s 1.14 and K8s 1.16).

A resizer container is added to the Controller Pod for the volume expansion feature.

For custom storage classes to support volume expansion feature, they need to set **allowVolumeExpansion: true** field.

No additional feature gates need to be enabled to implement Volume Expansion. Static and dynamic volumes both are supported for volume expansion.

Volumes can be expanded when they are mounted to pod as well.

Expand PersistentVolumeClaim (PVC) capacity

To expand PVC capacity, edit the *storage* field of the volume. For example, the following shows how to edit the *storage* field of the *myclaim* volume:

kubectl edit pvc myclaim

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
  namespace: default
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
```

```
storage: 5Gi
storageClassName: isilon
```

 **NOTE:** Reducing current PVC size is not supported

CSI Driver components

This topic describes the components of the CSI driver for Dell EMC Isilon.

The CSI driver for Dell EMC Isilon has two components:

- Controller plug-in
- Node plug-in

Controller Plug-in

The Controller plug-in is deployed in a StatefulSet in the 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 Isilon container and a few sidecar containers like the *provisioner* and *attacher*, that the 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.

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.

Installation

This chapter includes the following topics:

Topics:

- [Installation overview](#)
- [Prerequisites](#)
- [Install CSI driver for Dell EMC Isilon](#)
- [Upgrade CSI driver for Dell EMC Isilon](#)
- [Test deploying a simple pod with Isilon storage](#)
- [CSI Driver Usage](#)

Installation overview

This topic gives an overview of the CSI driver for Dell EMC Isilon installation.

The CSI driver for Dell EMC Isilon is deployed in the Kubernetes platform using Helm charts. The Helm chart installs CSI driver for Dell EMC Isilon using a shell script (`helm/install.isilon`). This 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 *isilon* namespace:

- CSI driver for Dell EMC Isilon
- Kubernetes Provisioner, which provisions the persistent volumes
- Kubernetes Attacher, which attaches the volumes to the node
- Kubernetes Snapshotter, which provides snapshot support
- Kubernetes Resizer, which provides resize support

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

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

Prerequisites

This topic lists the prerequisites to install the CSI driver for Dell EMC Isilon.

Before you install the CSI driver for Dell EMC Isilon, you must complete the following task:

- Install Kubernetes
- [Enable Kubernetes feature gates](#) on page 7
- [Configure Docker service](#) on page 8
- [Install the Helm 2 and Tiller package manager](#) on page 9

After completing these prerequisites, you are able to [Install CSI driver for Dell EMC Isilon](#) on page 10.

Enable Kubernetes feature gates

Enable the Kubernetes feature gates before installing the CSI driver for Dell EMC Isilon.

About this task

NOTE: You might need to enable other feature gates for different Kubernetes versions and distributions. Some of the feature gates might already be enabled by default (the value already set to true). For these feature gates, leave them as they are.

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

- VolumeSnapshotDataSource
- CSINodeInfo
- CSIDriverRegistry

Steps

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
CSINodeInfo: true
CSIDriverRegistry: 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=VolumeSnapshotDataSource=true,CSINodeInfo=true,CSIDriverRegistry=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=VolumeSnapshotDataSource=true,CSINodeInfo=true,CSIDriverRegistry=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=VolumeSnapshotDataSource=true,CSINodeInfo=true,CSIDriverRegistry=true
```

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

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

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

6. Restart the kubelet 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 Isilon. The recommended docker version is 19.03.

Steps

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

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

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

Install either Helm 3 or Helm 2 with Tiller package manager

Install either Helm 3 or Helm 2 with the Tiller package manager on the master node before you install the CSI driver for Dell EMC Isilon.

- [Install Helm 3](#) on page 9
- [Install the Helm 2 and Tiller package manager](#) on page 9

Install Helm 3

Install Helm 3 on the master node before you install the CSI driver for Dell EMC Isilon.

Steps

Run the `curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 | bash` command to install Helm 3.

Install the Helm 2 and Tiller package manager

Install the Helm and Tiller package manager on the master node before you install the CSI driver for Dell EMC Isilon. The recommended Helm and Tiller package version is 2.16.3.

Steps

1. Run `curl https://raw.githubusercontent.com/helm/helm/v2.16.3/scripts/get > get_helm.sh`
2. Run `chmod 700 get_helm.sh`.
3. Run `./get_helm.sh --version v2.16.3`.
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.

Certificate validation for OneFS REST API calls

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

Prerequisites

As part of the CSI driver installation, the CSI driver requires a secret with the name *isilon-certs* present in the namespace *isilon*. This secret contains the X509 certificates of the CA which signed the OneFS 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 `isiInsecure` which determines if the driver performs client-side verification of the OneFS certificates. The `isiInsecure` parameter is set to true by default and the driver does not verify the OneFS certificates.

Steps

- If the `isiInsecure` is set to false, then the secret `isilon-certs` must contain the CA certificate for OneFS.
- If this secret is an empty secret, then the validation of the certificate fails, and the driver fails to start.
- If the `isiInsecure` 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 OneFS certificate is self-signed, then perform the following steps:

1. To fetch the certificate:

```
openssl s_client -showcerts -connect <OneFS IP> </dev/null 2>/dev/null | openssl x509 -outform PEM > ca_cert.pem
```

2. To create the secret:

```
kubectl create secret generic isilon-certs --from-file=ca_cert.pem -n isilon
```

Install CSI driver for Dell EMC Isilon

CSI driver for Dell EMC Isilon can be installed in one of two ways. Choose the method appropriate for your environment.

Install the CSI driver for Dell EMC Isilon with one of the following methods:

- [Install the CSI driver for Dell EMC Isilon using Helm](#) on page 10
- [Install the CSI driver for Dell EMC Isilon using the Operator](#) on page 13

Install the CSI driver for Dell EMC Isilon using Helm

Install the CSI driver for Dell EMC Isilon using this procedure.

Prerequisites

Ensure that you meet the following prerequisites before you install the CSI driver for Dell EMC Isilon:

- You have the downloaded files ready for this procedure.
- You have the Helm chart from the source repository at <https://github.com/dell/csi-isilon> ready for this procedure.
- The top-level helm directory contains the `install.isilon` and `uninstall.isilon` shell scripts.

The scripts perform certain pre-installation and post-installation operations, such as creating Custom Resource Definitions, which cannot be performed in the helm chart.

- You have the Kubernetes secret with your OneFS 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: **isilon-creds**
- Namespace: **isilon**



NOTE: The username must be from the authentication providers in the System zone of Isilon. The user must have enough privileges to perform the actions. The suggested privileges are as follows:

- **ISI_PRIV_LOGIN_PAPI**
- **ISI_PRIV_NFS**
- **ISI_PRIV_QUOTA**
- **ISI_PRIV_SNAPSHOT**
- **ISI_PRIV_IFS_RESTORE**
- **ISI_PRIV_NS_IFS_ACCESS**

For more information about creating a Kubernetes secret, see <https://kubernetes.io/docs/concepts/configuration/secret/>.

- 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.

Steps

1. Collect information from the Isilon system, such as IP address, username and password.
Make a note of the value for these parameters as they must be entered in the `secret.yaml` and `myvalues.yaml` file.
2. Copy the `csi-isilon/values.yaml` into a file in the same directory as the `install.isilon` named `myvalues.yaml`, to customize settings for installation.

isiIP

This parameter is the HTTPs endpoint of the Isilon OneFS API server.

isiPort

This parameter is the HTTPs port number of the Isilon OneFS API server. The default value is 8080.

isiInsecure

This parameter specifies whether the Isilon OneFS API server's certificate chain and host name should be verified.

isiAccessZone

This parameter is the name of the access zone a volume can be created in. This parameter is used if `accessZone` is not specified for `storageClass`.

volumeNamePrefix

This parameter is used for configuring the provisioner sidecar and is used to create PVC names by k8s. The value of this parameter must be in all lower case and preferably short.

controllerCount

This parameter is the number of `csi-isilon` controller nodes to deploy to the Kubernetes release.

enableDebug

This parameter indicates whether debug level logs should be logged.

verbose

This parameter Indicates what content of the OneFS REST API message should be logged in debug level logs.

enableQuota

This parameter indicates whether the provisioner should attempt to set (later unset) quota on a newly provisioned volume.

noProbeOnStart

This parameter specifies whether the controller/node should probe during initialization.

autoProbe

This parameter enables auto probe.

nfsV3

This parameter specifies whether to set the version to v3 when mounting an NFS export. If the value is *false*, the default version that is supported is used.

The following are the parameters for `StorageClass` values:

name

This parameter is the name of the storage class to be defined.

isDefault

This parameter specifies whether this storage class should be default.

reclaimPolicy

This parameter indicates what happens when a volume is removed from the Kubernetes API. Valid values are *Retain* and *Delete*.

accessZone

This parameter is the name of the access zone a volume can be created in. Ensure that this `accessZone` exists on Isilon.

AzServiceIP

This parameter is the Access Zone service IP. If the value is different from `isiIP`, specify in `values` and refer in `storageClass`.

rootClientEnabled

When a pod on a node mounts the PVC in `NodeStageVolume`, this parameter determines whether to add the Kubernetes node ip/fqdn to the Root clients field when true or Clients field when false of the NFS export.

The following is the parameter for `VolumeSnapshotClass` values:

IsiPath

This parameter is the base path for the source volume the snapshot is created from. Ensure that this path exists on Isilon.

NOTE: This parameter must be consistent with `isiPath` in the storageclass that is used by the source volume.

3. Create a secret file for the OneFS credentials by editing the `secret.yaml`. Replace the values for the username and password parameters. These values can be optioned using base64 encoding as described in the following example:

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

4. Run `kubectl create namespace isilon` to create the *isilon* namespace.
5. Run `kubectl create -f secret.yaml` to create the secret.
6. Run `sh install.isilon` to proceed with the installation..

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.

NOTE: The installation might fail if you are installing the CSI driver for the first time. It can be due to network speed, since the sidecar containers might still be creating or downloading the resources. Uninstall the driver and install it again, if it occurs.

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

```
NAME                READY   STATUS    RESTARTS   AGE
isilon-controller-0  5/5    Running   0           20s
isilon-node-97fph    2/2    Running   0           20s
CSIDrivers:
NAME      CREATED AT
isilon    2020-05-28T02:55:33Z
CSINodes:
NAME      CREATED AT
lglou233  2020-04-07T11:07:21Z
StorageClasses:
NAME      PROVISIONER          AGE
isilon (default)  csi-isilon.dellemc.com  20s
crd VolumeSnapshotClass was already created
installing volumesnapshotclass instance isilon-snapclass
volumesnapshotclass.snapshot.storage.k8s.io/isilon-snapclass created
```

7. Optional: Run `sh uninstall.isilon` to uninstall CSI driver for Dell EMC Isilon.

Results

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

```
kubectl get pods -n isilon
```

You will see the following:

- `isilon-controller-0` with 5/5 containers ready, and status displayed as Running.
- Agent pods with 2/2 containers and the status displayed as Running.

Finally, the script lists the created storageclasses such as, "isilon". Additional storage classes can be created for different combinations of file system types and Isilon storage pools. The script also creates volumesnapshotclass "isilon-snapclass".

You can also test the installation of your driver.

Install CSI Isilon driver using Dell EMC Storage CSI Operator

Install the CSI driver for Dell EMC Isilon using the Operator


CSI driver for Dell EMC Isilon can also be installed by using Dell EMC Storage Operator.

The Dell EMC Storage CSI Operator is a Kubernetes Operator¹ that can be used to install and manage the CSI Drivers that Dell EMC provides 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 the OpenShift Container Platform. Both these methods of installation use Operator Lifecycle Manager (OLM).

Instructions for either deploying the Operator directly or deploying the CSI driver for Dell EMC Isilon using the Operator can be found at:

<https://github.com/dell/dell-csi-operator>

These instructions include sample manifests that can be edited for the installation of the driver.

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

¹Kubernetes 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 uses custom controllers to manage them.

Listing CSI-Isilon drivers

About this task

User can query for csi-isilon driver using the following command:

```
kubectl get csiisilon --all-namespaces
```

Create a new CSI-Isilon driver

Steps

1. Create the Isilon namespace:

```
kubectl create namespace isilon
```
2. Create isilon-creds
 - a. Create a file called isilon-creds.yaml with the following content:

```
yaml
apiVersion: v1
kind: Secret
metadata:
  name: isilon-creds
  namespace: isilon
type: Opaque
data:
  # set username to the base64 encoded username
  username: <base64 username>
  # set password to the base64 encoded password
  password: <base64 password>
```

- b. Replace the values for the username and password parameters. These values can be optioned using base64 encoding as described in the following example:

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

- c. Run `kubectl create -f isilon-creds.yaml` command to create the secret.
3. Create a CR (Custom Resource) for Isilon using the sample files provided at `test/sample_files/CR/`.
 4. Run the following command to create Isilon custom resource:

```
kubectl create -f <input_sample_file.yaml>
```

NOTE: The above command will deploy the csi-isilon driver.

5. User can configure the following parameters in a CR:

Parameter	Description	Required	Default
Common parameters for node and controller			
CSI_ENDPOINT	Specifies the HTTP endpoint for Isilon	No	/var/run/csi/csi.sock
X_CSI_DEBUG	To enable debug mode	No	false
X_CSI_ISI_ENDPOINT	HTTPs endpoint of the Isilon OneFS API server	Yes	
X_CSI_ISI_INSECURE	Specifies whether SSL security needs to be enabled for communication between Isilon and CSI Driver	No	true
X_CSI_ISI_PATH	Base path for the volumes to be created	Yes	
X_CSI_ISI_AUTOPROBE	To enable auto probing for driver	No	true
X_CSI_ISILON_NO_PROBE_ON_START	Indicates whether the controller/node should probe during initialization	Yes	
Controller parameters			
X_CSI_MODE	Driver starting mode	No	controller
X_CSI_ISI_ACCESS_ZONE	Name of the access zone a volume can be created in	No	System
X_CSI_ISI_QUOTA_ENABLED	To enable SmartQuotas	Yes	
Node parameters			
X_CSI_ISILON_NFS_V3	Set the version to v3 when mounting an NFS export. If the value is "false", then the default version supported will be used	Yes	
X_CSI_MODE	Driver starting mode	No	node

Upgrade CSI driver for Dell EMC Isilon

Prior to v1.2, CSI driver for Dell EMC Isilon supported upgrades using Helm 2. Beginning with CSI driver for Dell EMC Isilon v1.2 users can upgrade the driver using either Helm 3 or Helm 2.

Use one of the following two approaches to upgrade the CSI driver for Dell EMC Isilon:

- [Upgrade CSI driver for Dell EMC Isilon using Helm 2](#) on page 14
- [Migrate from Helm 2 to Helm 3 and upgrade CSI driver for Dell EMC Isilon](#) on page 15

Upgrade CSI driver for Dell EMC Isilon using Helm 2

Steps

1. Get the latest code from <https://github.com/dell/csi-isilon> by executing the following command.
git clone -b <VERSION> https://github.com/dell/csi-isilon.git
where <VERSION> is the version of CSI driver for Dell EMC Isilon
2. Prepare myvalues.yaml.

3. Run `./install.isilon` to upgrade the driver.
4. List the pods with the following command to verify the status:
`kubect1 get pods -n isilon`

Migrate from Helm 2 to Helm 3 and upgrade CSI driver for Dell EMC Isilon

Steps

1. Get the latest code from <https://github.com/dell/csi-isilon> by executing the following command.
`git clone -b <VERSION> https://github.com/dell/csi-isilon.git`
 where <VERSION> is the version of CSI driver for Dell EMC Isilon
2. Uninstall the CSI driver for Dell EMC Isilon using the `uninstall.isilon` script under `csi-isilon/helm` using Helm 2.
3. Go to https://helm.sh/docs/topics/v2_v3_migration/ and follow the instructions to migrate from Helm 2 to Helm 3.
4. Once Helm 3 is ready, install the CSI driver for Dell EMC Isilon using `install.isilon` script under `csi-isilon/helm`.
5. List the pods with the following command to verify the status:
`kubect1 get pods -n isilon`

Test deploying a simple pod with Isilon storage

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

Steps

1. Create a volume.
 - a. Create a file `pvc.yaml` using the sample yaml file located at `test/sample_files`
 - b. Run the following command to create the test volume:
`kubect1 create -f $PWD/pvc.yaml`
 After executing the above command PVC will be created in the default namespace, and the user can see the pvc by executing `kubect1 get pvc`.
2. Verify system for the new volume.
3. Attach the volume to a Host.
 To attach the test volume to a host, create a new application(Pod) and use the PVC created above in the Pod. This scenario is explained using the Nginx application.
 - a. Create `nginx.yaml` using the sample yaml file located at `test/sample_files`.
 - b. Run the following command to mount the volume to kubernetes node:
`kubect1 create -f $PWD/nginx.yaml`
 After running the above command, a new nginx pod will be successfully created and started in the default namespace.
4. Verify Isilon system for host to be part of `clients/rootclients` field of export created for volume and used by nginx application.
5. Create a snapshot of the volume in the container using VolumeSnapshot objects defined in `snap.yaml`. The sample file for snapshot creation is located at `test/sample_files/`
`kubect1 create -f $PWD/snap.yaml`
 The `spec.source` section contains the volume that will be snapped in the default namespace. For example, if the volume to be snapped is `testvolclaim1`, then the created snapshot is named `testvolclaim1-snap1`.
6. Verify the Isilon system for new created snapshot.

NOTE:

- User can see the snapshots using `kubect1 get volumesnapshot`
- VolumeSnapshot class has a reference to a `snapshotClassName:isilon-snapclass`. The CSI Driver for Isilon installation creates this class as its default snapshot class.
- You can see its definition using `kubect1 get volumesnapshotclasses isilon-snapclass -o yaml`.

7. Create a volume from the snapshot.

The following will create a new volume from a given snapshot which is specified in spec dataSource field.

The sample file for volume creation from a snapshot, `volume_from_snap.yaml`, is located at `test/sample_files/`.

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

8. Verify the Isilon system for new created volume from snapshot.
9. Delete test snapshot.

```
kubectl get volumesnapshot
```

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

10. Delete the nginx application to unattach the volume from the host.

```
kubectl delete -f nginx.yaml
```

11. Delete the test volume.

```
kubectl get pvc
```

```
kubectl delete pvc testvolclaim1
```

```
kubectl get pvc
```

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 array. The `StorageClass` parameters are as follows:

Parameter	Description
<code>name</code>	The name of the storage class to be defined.
<code>reclaimPolicy</code>	This parameter indicates what happens when a volume is removed from the Kubernetes API. Valid values are <i>Retain</i> and <i>Delete</i> .
<code>AccessZone</code>	This parameter is the name of the access zone a volume can be created in. Ensure that this <code>AccessZone</code> exists on Isilon.
<code>IsiPath</code>	This parameter is the base path for the volumes to be created. Ensure that this path exists on Isilon.
<code>AzServiceIP</code>	Access Zone service IP.
<code>RootClientEnabled</code>	When a pod on a node mounts the PVC in <code>NodeStageVolume</code> , this parameter determines whether to add the Kubernetes node <code>ip/fqdn</code> to the Root clients field when true or Clients field when false of the NFS export.

`VolumeSnapshotClass` parameter:

- `IsiPath` – The base path for the source volume the snapshot is created from. Ensure that this path exists on Isilon. It should be consistent with `isiPath` in the `storageclass` that the source volume uses.

Controller Plug-in query commands

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

Steps

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

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


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

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

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

Set the log levels in the verbose field of `myvalues.yaml` to refine the granularity of the logs on RESTful calls to OneFS. Ensure the parameter `enableDebug` in `myvalues.yaml` is set to true to see these RESTful calls.

Log Level	Definition
0	Log full content of the HTTP request and response
1	Log without the HTTP response body
2	Log only first line of the HTTP request and response

Node Plug-in query commands

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

Steps

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

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

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

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

Test the CSI driver for Dell EMC Isilon

This chapter includes the following topics:

Topics:

- [Test the CSI driver for Dell EMC Isilon](#)
- [Test creating snapshots](#)
- [Test restoring from a snapshot](#)

Test the CSI driver for Dell EMC Isilon

This topic provides information about testing the CSI driver for Dell EMC Isilon.

About this task

The *csi-isilon* repository includes examples of how you can use the CSI driver for Dell EMC Isilon. 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:

Steps

1. Create a namespace with the name *test*.
2. Run the `cd csi-isilon/test/helm` command to go to the *csi-isilon/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 pod and the PVCs created during the test and uninstalls the helm chart.

Test creating snapshots

Test the workflow for snapshot creation.

Steps

1. Start the *2vols* container and leave it running.
2. Run the *snaptest.sh* shell script.

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

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

Results

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

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

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

Test restoring from a snapshot

Test the restore operation workflow to restore from a snapshot.

Prerequisites

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

About this task

To test the restore operation from a snapshot:

Steps

Run the *snapprestoretest.sh* shell script.

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

Results

An outline of this workflow is described below:

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

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: restorepvc
  namespace: test
spec:
  storageClassName: isilon
  dataSource:
    name: pvol0-snap1
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 8Gi
```

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