CSI Driver for Dell EMC Isilon

Version 1.0

Product Guide

REV 01

December 2019

DELLEMC

CONTENTS

Contents

CHAPTER 1

Introduction

This chapter includes the following topics:

Product overview

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

The CSI Driver for Dell EMC Isilon and Kubernetes communicate using the Container Storage Interface protocol. The CSI Driver for Dell EMC Isilon conforms to CSI specification v1.1. It is verified with Kubernetes versions 1.14.7 with the Red Hat Enterprise Linux 7.6 host operating system.

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.

Features of the CSI driver for Dell EMC Isilon

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

*   Supports CSI 1.1
*   Supports Kubernetes version 1.14.x
*   Supports Red Hat Enterprise Linux 7.6 host operating system
*   Supports Isilon OneFS 8.1 and 8.2
*   Persistent Volume (PV) capabilities:
    *   Create from scratch
    *   Create from snapshot
    *   Delete

- Dynamic PV provisioning
- Volume mount as NFS export
- HELM charts installer
- Access modes:
  - SINGLE_NODE_WRITER
  - MULTI_NODE_READER_ONLY
  - MULTI_NODE_MULTI_WRITER
- Snapshot capabilities:
  - Create
  - Delete

(i) 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.

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:

Procedure

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.

CHAPTER 2

Installation

This chapter includes the following topics:

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 repository includes Helm charts that use a shell script to deploy the CSI driver for Dell EMC Isilon. The shell script installs the container image along with the required 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 that provisions the persistent volumes
- Kubernetes Attacher that attaches the volumes to the node
- Kubernetes Snapshotter that provides snapshot 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 that 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
  The CSI driver for Dell EMC Isilon works with Kubernetes versions 1.14.x.
- Enable Kubernetes feature gates
- Configure Docker service
- Install the Helm and Tiller package manager

Enable Kubernetes feature gates

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

About this task

(i) 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 1.14.7.

The Feature Gates section of the Kubernetes documentation lists the Kubernetes feature gates. Enable the following Kubernetes feature gates:

- VolumeSnapshotDataSource
- KubeletPluginsWatcher
- CSINodeInfo
- CSIDriverRegistry

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/kuberlet/config.yaml
VolumeSnapshotDataSource: true
KubeletPluginsWatcher: 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,KubeletPluginsWatcher=true,CSIN
odeInfo=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,KubeletPluginsWatcher=true,CSIN
odeInfo=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,KubeletPluginsWatcher=true,CSIN
odeInfo=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=--bootstrapkubeconfig=/etc/
kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/
kubelet.conf --
allowprivileged=true --
feature-
gates=VolumeSnapshotDataSource=true,KubeletPluginsWatcher=true,CSIN
odeInfo=true,CSIDriverRegistry=true"
```

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

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

Configure Docker service

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 18.06.

Procedure

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

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

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 Isilon. The recommended Helm and Tiller package version is 2.14.0.

Procedure

1. 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 Isilon

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

Before you begin

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 GitHub, ready for this procedure.
- The top-level helm directory contains the *install.isilon* and *uninstall.isilon* 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 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`

  (i) **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 Kubernetes documentation: Overview of Secrets.

- 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-isilon.git` to clone the git repository to the master node of the Kubernetes cluster.

2. Run `cd csi-isilon/helm && cp csi-isilon/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 OneFS endpoint.

4. Copy the *csi-isilon/values.yaml* to the *myvalues.yaml* in the helm directory and provide values for the following parameters:

- `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.

- `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.

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

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

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

- `isiPath`: This parameter is the default base path for the volumes to be created, and it is used if a storage class does not have the `isiPath` parameter specified. Ensure that this path exists on Isilon.

- `enableDebug`: This parameter indicates whether debug level logs should be logged.

- `autoProbe`: This parameter enables auto probe.

- `noProbeOnStart`: This parameter specifies whether the controller/node should probe during initialization.

- `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.

- `isiPath`: This parameter is the base path for the volumes to be created. Ensure that this path 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.

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.
  (i) **Note:** This parameter must be consistent with isiPath in the storageclass that is used by the source volume.

5. 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`

6. Run `kubectl create namespace isilon` to create the *isilon* namespace.

7. Run `kubectl create -f secret.yaml` to create the secret.

8. Run `sh install.isilon` to install the driver.

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

   (i) 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.

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 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 array. The StorageClass parameters are as follows:

Parameter	Description
name	The name of the storage class to be defined.
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.
IsiPath	This parameter is the base path for the volumes to be created. Ensure that this path exists on Isilon.
AzServiceIP	Access Zone service IP.

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 used by the source volume.

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 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.

Node Plug-in query commands

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 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
```

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.

Before you begin

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.

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:

Procedure

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

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

Installation

CHAPTER 3

Test the CSI driver for Dell EMC Isilon

This chapter includes the following topics:

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:

Procedure

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.

Procedure

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*.

(i) **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.

Before you begin

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

About this task

To test the restore operation from a snapshot:

Procedure

1. Run the `snaprestoretest.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 *snaprestoretest.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
```

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

INDEX

Index