Hedvig CSI User Guide

Table of Contents

Introduction

The *Container Storage Interface* (CSI) is a standard for exposing arbitrary block and file storage systems to containerized workloads on Container Orchestration (CO) systems, such as Kubernetes.

Supported Container Orchestrators

- Kubernetes 1.13 – 1.19
- Red Hat OpenShift 4.1 – 4.6
- D2iQ Konvoy – For more information about setting up D2iQ Konvoy with Hedvig, see: https://docs.d2iq.com/ksphere/konvoy/partner-solutions/hedvig/

Supported Platforms

- CentOS/RHEL 7.x
- Ubuntu 16.04 LTS or above
- RHCOS

Compatible Hedvig CSI Drivers

Table 1: Hedvig CSI Driver compatibility matrix

Hedvig version	Hedvig CSI Driver version
Hedvig 4.0.x	2.0
Hedvig 4.1.x	2.0
Hedvig 4.2.x	2.0

Required Network Setup

- If a firewall is enabled on the Kubernetes/OpenShift nodes, unblock the ports in the following table.

Table 2: Ports to be unblocked if firewall enabled

Protocol	Port range	Description
TCP	50022	ssh
TCP	2049	nfs
TCP	33333	nfs (mountd)
TCP	3260	iscsi
TCP	50000 - 50008	thrift
TCP	15000	thrift
TCP	8000	thrift

- Make sure that the Kubernetes/OpenShift nodes and the Hedvig Storage Cluster Nodes can communicate with each other.

OpenShift Prerequisites

- Make sure that the following SCCs (security context constraints) exist:
  - `hostnetwork`
  - `hostpath`
  - `privileged`

- Make sure that the following capabilities are enabled on the aforementioned SCCs:
  - Allow Host Dir Volume Plugin: true
  - Allow Host Network: true
  - Allow Privileged Container: true

- Make sure that the aforementioned SCCs are added to the default and hedvig-csi service accounts in the namespace where the Hedvig components are installed. The following command can be used:

  ```
  oc adm policy add-scc-to-user <scc-name>
  ```

  ```
  system:serviceaccount:<namespace>:<service-account-name>
  ```

Host Prerequisites

The location for installed Hedvig services depends on whether you are using Kubernetes or OpenShift:

- For Kubernetes, Hedvig services are installed on Kubernetes worker nodes.
- For OpenShift, Hedvig services are installed (by default) on OpenShift compute nodes.

Prerequisites must be met on these specific nodes, that is, worker nodes for Kubernetes, compute nodes for OpenShift.

- *Hedvig Storage Proxy (Block) Prerequisites*
- *Hedvig Storage Proxy (NFS) Prerequisites*
- *Hedvig Storage Proxy Caching Prerequisites*

Hedvig Storage Proxy (Block) Prerequisites

1. Install iSCSI initiator utilities (`iscsi-initiator-utils/open-iscsi`).
2. Make sure that the following processes are enabled and running:
   - `rpcbind`
   - `iscsid`
3. Make sure that the `iscsi` kernel modules are loaded. To list these modules, run:

```
lsmod | grep iscsi

scsi_tcp               18333  2
libiscsi_tcp           25146  1 iscsi_tcp
libiscsi               57233  2 libiscsi_tcp,iscsi_tcp
scsi_transport_iscsi   99909  3 iscsi_tcp,libiscsi
```

Multipath Configuration

Multipath is **not recommended** for iSCSI persistent volumes with Hedvig. If multipath is enabled by default on Kubernetes/OpenShift compute nodes, a blacklist must be created for Hedvig volumes using the vendor id as shown below:

```
# cat /etc/multipath.conf
blacklist {
       device {
               vendor  "_HEDVIG_"
       }
}
```

Hedvig Storage Proxy (NFS) Prerequisites

1. Install NFS utilities (`nfs-utils/nfs-common`).
2. Make sure that the `rpcbind` process is running.
3. Make sure that the `messagebus/dbus` process is running.
4. Create the `/etc/dbus-1/system.d/org.ganesha.nfsd.conf` file with the following contents, and restart `messagebus/dbus`.

```
<?xml version="1.0" encoding="UTF-8"?> <!-- -*- XML -*- -->

<!DOCTYPE busconfig PUBLIC
"-//freedesktop//DTD D-BUS Bus Configuration 1.0//EN"
"http://www.freedesktop.org/standards/dbus/1.0/busconfig.dtd">
<busconfig>
  <!-- Only root can own the service -->
  <policy user="root">
    <allow own="org.ganesha.nfsd"/>
    <allow send_destination="org.ganesha.nfsd"/>
  </policy>
  <policy context="default">
    <deny own="org.ganesha.nfsd"/>

    <allow send_destination="org.ganesha.nfsd"
           send_interface="org.freedesktop.DBus.Introspectable"/>

    <allow send_destination="org.ganesha.nfsd"
           send_interface="org.ganesha.nfsd.CBSIM"/>

    <allow send_destination="org.ganesha.nfsd"
           send_interface="org.ganesha.nfsd.admin"/>

    <allow send_destination="org.ganesha.nfsd"
           send_interface="org.ganesha.nfsd.ExportMgr"/>
  </policy>
</busconfig>
```

Hedvig Storage Proxy Caching Prerequisites

For OpenShift installations, if you plan to run the Hedvig Storage Proxies on **infra nodes**, perform the following actions on OpenShift **infra nodes**, as well.

Setting up the metacache on tmpfs

The following steps describe the metacache setup on `tmpfs` mounted on host path **`/hedvig/cache`**. If you choose to change the host path, update the `metacache.service` file accordingly.

1. Create the `/etc/systemd/system/metacache.service` file.

```
[Unit]
Description=Setup Metacache
After=network.target tgt.service hedvigfsc.service

[Service]
Type=oneshot
ExecStart=/bin/bash -c "/bin/mount -t tmpfs -o size=4g tmpfs /hedvig/cache"
ExecStartPre=/bin/bash -c "/bin/mkdir -p /hedvig/cache"
RemainAfterExit=true
ExecStop=/bin/true
StandardOutput=journal

[Install]
WantedBy=multi-user.target
```

2. Run the following commands to set up the cache:

```
systemctl enable metacache
systemctl start metacache
```

3. Verify that **`/hedvig/cache`** is present by running:

```
df -kh
```

Hedvig Storage Proxy and CSI Driver Image Repositories

The following table provides the repository locations for the Hedvig Storage Proxy and the Hedvig CSI Driver.

Table 3: Hedvig Storage Proxy and CSI Driver Image Repositories

Hedvig Storage Proxy	*https://hub.docker.com/r/hedvig/hedvig-proxy/tags*
Hedvig CSI Driver	*https://hub.docker.com/r/hedvig/hedvig-csi-volume-driver/tags*

Download and Configure the Installer

Configure the installer

1. Download the Hedvig CSI Installer v2.0 from the Commvault Store:

   https://cloud.commvault.com/webconsole/softwarestore/#!/

2. Copy it to any machine where `kubectl` is installed.

3. Extract the downloaded package.

4. Locate the configuration file `hedvigconfig`.

5. Update the following parameters in `hedvigconfig`:

   - **NAMESPACE** – Kubernetes/OpenShift namespace where the components will be installed. (Make sure that this namespace already exists.)
   - **HEDVIG_CLUSTER_NAME** – Name of the Hedvig Storage Cluster
   - **HEDVIG_SEED_1**, **HEDVIG_SEED_2**, and **HEDVIG_SEED_3** – Hostnames of any three Hedvig Storage Cluster Nodes in the Hedvig Storage Cluster
   - **KUBE_CLUSTER_HEDVIG_ID** – Unique id for the Kubernetes/OpenShift/D2iQ cluster
   - **HEDVIG_BLOCK_IMAGE_TAG** – Hedvig Block proxy image tag corresponding to the software version installed on the Hedvig Storage Cluster.
   - **HEDVIG_NFS_IMAGE_TAG** – Hedvig NFS proxy image tag corresponding to the software version installed on the Hedvig Storage Cluster
   - **HEDVIG_CSI_IMAGE_TAG** – Hedvig CSI driver image tag corresponding to the platform/OS. Image tags without platform details are CentOS/RHEL based.
   - (optional) **CSI_SNAPSHOTTER_IMAGE_TAG** – Update this to **"v2.1.0"** if you use the `v1beta1` snapshot API or the snapshot scheduler.
   - (optional) **CSI_PROVISIONER_IMAGE_TAG**– Update this to **"v1.6.0"** if you use the `v1beta1` snapshot API or the snapshot scheduler.

6. If the metacache is configured on the worker/compute nodes, update the following parameter in `hedvigconfig`:

    - **HEDVIGCACHE_HOSTPATH** – Host path for metacache

Generate configuration files

After updating `hedvigconfig`, execute the following script to generate configuration files for Hedvig components:

```
# ./genconfig.sh
```

A successful execution will result in a `config/` directory with the generated configuration files.

Install the DaemonSet on Kubernetes Worker (or OpenShift Compute) Nodes

IMPORTANT: Recommended settings for the Hedvig Storage Proxy are 4 vCPUs, 8 GB RAM.

1. Locate the generated manifests for the DaemonSet:

   ```
   config/daemonset-block.yml
   config/daemonset-nfs.yml
   ```

2. Deploy the DaemonSet by running:

   ```
   kubectl create -f daemonset-block.yml
   kubectl create -f daemonset-nfs.yml
   ```

3. In the Hedvig WebUI, verify that the Hedvig Storage Proxies are up and running.

   In the Hedvig CLI, run:

   ```
   hedvig> lskubecontrollers -i <KUBE_CLUSTER_HEDVIG_ID> -t block
   hedvig> lskubecontrollers -i <KUBE_CLUSTER_HEDVIG_ID> -t nfs
   ```

   These commands should list all Kubernetes/OpenShift/D2iQ nodes.

Setup Instructions for the Hedvig CSI Driver

Create the CSI Driver Secret

Starting with v2.0, the Hedvig CSI controller authenticates with the Hedvig Storage Cluster on startup and processes requests only after successful authentication.

The CSI driver secret must have the name `hedvig-cluster-credentials` and must be created in the same namespace where the CSI driver will be installed.

To create the CSI driver secret, run the following command:

```
# kubectl create secret generic hedvig-cluster-credentials
  --from-literal=username='<username>'
  --from-literal=password='<password>' -n <namespace>
```

<username> and **<password>** should correspond to a valid user account on the Hedvig Storage Cluster.

Install the Hedvig CSI Driver

1. To install the Hedvig CSI Driver:

   ```
   ./install_hedvig.sh
   ```

2. See the sample templates for Storage Classes and applications here:

   ```
   manifests/
   ```

Verify the Installation

1. Verify that the Hedvig CSI Driver pods are in `Running` state.

   Run the following command to verify the installation:

   ```
   kubectl get pods -n <namespace>
   ```

   If the installation is successful, the output should look like this:

   ```
   kubectl get pods -n <namespace>

   NAME                                      READY    STATUS    RESTARTS    AGE
   hedvig-csi-controller-f69cf7f65-hxhlp     5/5      Running   0           68s
   hedvig-csi-node-cjd4q                     2/2      Running   0           68s
   hedvig-csi-node-f82km                     2/2      Running   0           68s
   hedvig-csi-node-nrclg                     2/2      Running   0           68s
   ```

2. From the Hedvig CLI, run the following command:

   ```
   lsallkubeclusters
   ```

   This command should list the `KubeClusterID`. This verifies that the Hedvig CSI Driver has successfully registered with the Hedvig Storage Cluster.

Hedvig CSI Driver Features and Capabilities

CSI Driver Name: io.hedvig.csi

Compatible CSI Version(s): v1.0

Dynamic Provisioning: Yes

Backend Types: **hedvig-block, hedvig-nfs**

Table 4: Hedvig CSI Driver features and capabilities

Backend Type	Access Modes	Snapshots and Clones	Volume Expansion	Raw Block Volume	Other Features/ Limitations
hedvig-block	ReadWriteOnce	Yes	Yes	Yes	Filesystems supported: • ext4 (default) • xfs
hedvig-nfs	ReadWriteOnce/ ReadWriteMany	No	Yes	No	• Only NFS v4.0 is supported • NFS locking is not supported

Storage Classes and Hedvig Virtual Disk Attributes

Creating a Default Storage Class for Persistent Volumes

The following manifests create a Storage Class for persistent volumes (PVs) backed by Hedvig virtual disks of type Block, depending on the replication policy of the cluster.

Note: Use `backendType` of `"hedvig-nfs"` in the Storage Class to create PVs backed by Hedvig virtual disks of type NFS.

Storage Class with Agnostic (Default) Replication Policy

If the cluster is set up with the `Agnostic` replication policy (the default), then the default storage class should be created using the following manifest:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-hedvig-block
provisioner: io.hedvig.csi
parameters:
  backendType: "hedvig-block"
  rp: "Agnostic"
```

Storage Class with RackAware Replication Policy

If the cluster is set up across multiple racks (`RackAware` replication policy), then the default Storage Class should be created using the following manifest:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
    name: sc-hedvig-block
provisioner: io.hedvig.csi
parameters:
    backendType: "hedvig-block"
    rp: "RackAware"
```

Storage Class with DataCenterAware Replication Policy

If the cluster is set up across multiple data centers (`DataCenterAware` replication policy), then the default Storage Class should be created using the following manifest:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
    name: sc-hedvig-block
provisioner: io.hedvig.csi
parameters:
    backendType: "hedvig-block"
    rp: "DataCenterAware"
    dcNames: "<comma-separated-list-of-datacenter-names>"
```

Storage Class with Agnostic/RackAware Replication Policy within a Data Center

If the cluster is set up across multiple data centers, then the Storage Class can also be created to replicate the data within a single data center using the following manifest:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-hedvig-block
provisioner: io.hedvig.csi
parameters:
  backendType: "hedvig-block"
  rp: "Agnostic"
  dcNames: "<datacenter-name>"
  rf: "<replication-factor>"
```

Customizing Storage Classes with Hedvig Virtual Disk Attributes

Storage Classes can be customized by specifying Hedvig virtual disk attributes in the `parameters` section. *Table 5: Hedvig Virtual Disk Attribute Keys* lists all possible attribute keys and their values.

Table 5: Hedvig Virtual Disk Attribute Keys

Key	Values	Default value	Notes
dedupEnable	true/false	false	Not supported for **hedvig-nfs** backend type
compressed	true/false	false	
cacheEnable	true/false	false	Not supported for **hedvig-nfs** backend type
rf (replication factor)	1 to 6	3	
rp (replication policy)	Agnostic/RackAware/ DataCenterAware	Agnostic	
dcNames	comma-separated list of data center names		Applicable only if **rp** (replication policy) is **DataCenterAware**
diskResidence	flash/hdd	hdd	In an all-flash cluster, **diskResidence** should always be set to **flash**.
encryptionEnable	true/false	false	
tenant	valid tenant name	Hedvig	

Key	Values	Default value	Notes
migrationEnable	true/false	false	Not supported for **hedvig-nfs** backend type Not supported if **dedupEnable** is true For more information, see *Data Migration for CSI Volumes.*
migrationLocations	comma-separated list of migration location names		Applicable only if **migrationEnable** is true For more information, see *Data Migration for CSI Volumes.*
schedulePolicy	snapshot schedule name		Not supported for **hedvig-nfs** backend type For more information, see *Scheduled Snapshots*.

Setting the Reclaim Policy

The default reclaim policy for dynamically created persistent volumes (PVs) is set to `Delete`. If the `PersistentVolumeClaim` is deleted, both the persistent volume bound to it and the corresponding Hedvig virtual disk are deleted.

To retain a persistent volume beyond the lifetime of its `PersistentVolumeClaim`, set the reclaim policy to `Retain` as shown in the following Storage Class.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-hedvig-block-retain
provisioner: io.hedvig.csi
parameters:
  backendType: "hedvig-block"
reclaimPolicy: "Retain"
```

Note: When the `PersistentVolume` is deleted, the corresponding Hedvig virtual disk will not be deleted. Deletion of the Hedvig virtual disk should be performed through the Hedvig WebUI.

Setting the File System Type

This procedure applies only to storage classes with a **hedvig-block** backend type.

By default, the `ext4` file system is installed on any persistent volume (PV) created with the **hedvig-block** backend type.

To use the `xfs` file system, set the file system type (`fsType`) as shown in the following storage class.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-hedvig-block-xfs
provisioner: io.hedvig.csi
parameters:
  backendType: "hedvig-block"
  fsType: "xfs"
```

Setting the Mount Options

Volume mount options can be specified in the storage class for both **hedvig-block** and **hedvig-nfs** backend types.

For a **hedvig-block** backend type, set the mount options as shown in the following storage class.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-hedvig-block-mntopt
provisioner: io.hedvig.csi
parameters:
  backendType: "hedvig-block"
  fsType: "ext4"
mountOptions:
  - noatime
  - nodiratime
  - max_batch_time=0
```

The mount options specified should be applicable to the file system selected.

For a **hedvig-nfs** backend type, set the mount options as shown in the following storage class.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-hedvig-nfs-mntopt
provisioner: io.hedvig.csi
parameters:
  backendType: "hedvig-nfs"
mountOptions:
  - actimeo=0
  - nolock
  - nordirplus
```

The mount options specified should be applicable to NFS.

Notes:

- Kubernetes does not validate the mount options specified. Therefore, if the mount options specified do not apply to the corresponding persistent volume (PV), the mount will fail.
- Certain mount options depend on the kernel settings of the host running the `kubelet` and can cause the mount to fail.

Volume Expansion

Volume expansion is supported for persistent volumes (PVs) with both **hedvig-block** and **hedvig-nfs** backend types.

- For a **hedvig-block** backend type, the controller server performs the volume expansion. If the volume mode specified is `Filesystem`, then the node server performs the filesystem expansion. Offline filesystem expansion is not supported.
- For a **hedvig-nfs** backend type, the controller server performs the volume expansion.

Volume expansion is described in the following sections:

Feature Gates

For Kubernetes versions 1.15 and below, the following feature gates must be enabled to perform volume expansion:

- `ExpandCSIVolumes`
- `ExpandInUsePersistentVolumes`.

Creating a Storage Class with Volume Expansion Enabled

Volume expansion is allowed only if it is permitted by the storage class corresponding to the persistent volume.

To enable volume expansion in the storage class, create a storage class with the highlighted attribute:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
    name: sc-hedvig-block
provisioner: io.hedvig.csi
parameters:
    backendType: "hedvig-block"
allowVolumeExpansion: true
```

Expanding Persistent Volumes

To expand a persistent volume, edit the corresponding persistent volume claim object, and update the storage size in the persistent volume claim spec.

To expand a persistent volume using the `kubectl` command, execute the following:

```
kubectl patch pvc <pvc-name>
   -p '{"spec":{"resources":{"requests":{"storage": "<new-size>"}}}}'
```

Raw Block Volume

Block volumes can be dynamically provisioned and consumed as block devices inside containers with raw block volume support. This is applicable only to a **hedvig-block** backend type.

Creating a Raw Block Volume

A raw block volume can be created by specifying the volume mode as Block in the corresponding persistent volume claim.

```
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Block
  resources:
    requests:
      storage: 10Gi
```

Consuming a Raw Block Volume

A raw block volume can be consumed as a block device inside the container by specifying a device path for the volume.

```
spec:
  volumes:
  - name: pv-block-raw
          persistentVolumeClaim:
              claimName: pvc-block-raw

  containers:
      - name: ctr-block-raw
    volumeDevices:
    - devicePath: "/dev/xvda"
              name: pv-block-raw
```

Snapshots and Clones

Snapshots and clones are supported only for persistent volumes (PVs) with a **hedvig-block** backend type.

Feature Gates

Clones can be created only with volume snapshot as the data source. On Kubernetes 1.16 (or lower) and OpenShift 4.3 (or lower), verify that the `VolumeSnapshotDataSource` feature gate is enabled on the API server.

Create a Default Snapshot Class

The following manifest creates a default Snapshot Class for the Hedvig CSI Driver, using the `v1alpha1` snapshot API.

```
apiVersion: snapshot.storage.k8s.io/v1alpha1
kind: VolumeSnapshotClass
metadata:
  name: hedvig-snapshot-class
snapshotter: io.hedvig.csi
```

The following manifest creates a default Snapshot Class for the Hedvig CSI Driver, using the `v1beta1` snapshot API.

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshotClass
metadata:
  name: hedvig-snapshot-class
driver: io.hedvig.csi
deletionPolicy: Delete
```

Create a Volume Snapshot

The following manifest creates a volume snapshot, using the `v1alpha1` snapshot API of the persistent volume (PV) bound to the `pvc-block` claim.

```
apiVersion: snapshot.storage.k8s.io/v1alpha1
kind: VolumeSnapshot
metadata:
  name: snapshot-pvc-block
spec:
  snapshotClassName: hedvig-snapshot-class
  source:
    name: pvc-block
    kind: PersistentVolumeClaim
```

The following manifest creates a volume snapshot, using the `v1beta1` snapshot API of the persistent volume (PV) bound to the `pvc-block` claim.

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshot
metadata:
  name: snapshot-pvc-block
spec:
  volumeSnapshotClassName: hedvig-snapshot-class
  source:
    persistentVolumeClaimName: pvc-nginx-block
```

Create a Default Storage Class for Clones

The following manifest creates a default Storage Class for cloning volume snapshots.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-hedvig-clone
provisioner: io.hedvig.csi
parameters:
  backendType: "hedvig-block"
```

Create a Clone from a Volume Snapshot

The following manifest creates a persistent volume claim (PVC), which results in the creation of a clone virtual disk on Hedvig from the given volume snapshot.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-clone
spec:
  storageClassName: sc-hedvig-clone
  dataSource:
    name: snapshot-pvc-block
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 25Gi
```

Note: The attributes associated with the clone virtual disk will be identical to those of the virtual disk corresponding to the volume snapshot.

Scheduled Snapshots

Scheduled snapshots enable users to schedule periodic snapshots of persistent volumes.

Snapshot API Compatibility

The change from the `v1alpha1` to the `v1beta1` snapshot API is not backward compatible. Therefore, scheduled snapshots will be supported only for the `v1beta1` snapshot API.

The `v1beta1` snapshot API is available only from Kubernetes 1.17 (and OpenShift 4.4) onward, and scheduled snapshots will be supported starting only with these releases.

Snapshots can be scheduled for persistent volumes by following these steps:

- *Create a Snapshot Schedule*
- *Create a Storage Class with the Snapshot Schedule*
- *Create a Snapshot Class*
- *Create a Persistent Volume Claim using the Storage Class*
- *Run Additional Helpful Commands As Needed*

Create a Snapshot Schedule

Kubernetes (and the CSI Spec) does not provide a native type for creating snapshot schedules. Snapshot schedules are implemented as a CRD (CustomResourceDefinition) and are created by the CSI controller server.

After the CSI driver has been deployed, you can verify the existence of the CRD by running the following command:

```
# kubectl get crd schedulepolicies.hedvig.io

NAME                          CREATED AT
schedulepolicies.hedvig.io    2020-05-13T20:58:54Z
```

Snapshot schedules are Cluster scoped. An interval schedule for snapshots can be created by specifying the periodicity of snapshots and the retention for each snapshot, for example:

```
apiVersion: hedvig.io/v1
kind: SchedulePolicy
metadata:
  name: minute-schedule
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: hedvig.io/provisioner
spec:
  interval:
    period: 1m
    retain: 2m
```

This schedule creates a new snapshot every minute and deletes the snapshot after two minutes. The values assigned to period and retain are duration strings.

Create a Storage Class with the Snapshot Schedule

After a snapshot schedule has been created, create a new storage class with this snapshot schedule.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-hedvig-minute-schedule
provisioner: io.hedvig.csi
parameters:
  backendType: "hedvig-block"
  schedulePolicy: "minute-schedule"
```

Create a Snapshot Class

The snapshot scheduler expects a snapshot class with name `snc-hedvig-block` to be present for creating snapshots of persistent volume claims. A default snapshot class can be created using the following configuration:

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshotClass
metadata:
  name: snc-hedvig-block
driver: io.hedvig.csi
deletionPolicy: Delete
```

Create a Persistent Volume Claim using the Storage Class

After the storage class has been created, create a persistent volume claim (PVC).

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-block
  annotations:
    volume.beta.kubernetes.io/storage-class: sc-hedvig-minute-schedule
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

Based on the associated snapshot schedule, you should now see snapshots created for this persistent volume claim every minute. To list the snapshots, run the following command:

```
kubectl get volumesnapshot
```

```
NAME                                          AGE
pvc-block-1589612240006-minute-schedule   113s
pvc-block-1589612300006-minute-schedule   53s
```

Use the snapshot AGE column to verify that the snapshots are deleted as per retention period.

Run Additional Helpful Commands as Needed

- **To list all schedule policies:**

  ```
  kubectl get schedulepolicies
  ```

- **To list all persistent volume claims for a given schedule policy:**

  ```
  kubectl get pvc -l "schedulepolicy.hedvig.io/<schedule-policy-name>"
  ```

- **To stop and start snapshots for a persistent volume claim:**

  When schedule policy is specified in a storage class, all persistent volume claims created using that storage class have snapshots enabled.

  To disable snapshots for a given persistent volume claim, run the following command:

  ```
  kubectl label pvc <pvc-name> snapshot-enabled=false --overwrite
  ```

  To re-enable snapshots, run the following command:

  ```
  kubectl label pvc <pvc-name> snapshot-enabled=true --overwrite
  ```

Data Migration for CSI Volumes

For data migration, the source and target Hedvig clusters should be running at least Hedvig 4.1.x.

Data migration can be configured for CSI volumes by following these steps:

- *Create a Migration Location*
- *Create a Snapshot Schedule (and Snapshot Class)*
- *Create a Storage Class with Migration Location and Snapshot Schedule*
- *Create a Persistent Volume Claim using the Storage Class*
- *Access the Migrated Persistent Volume on the Target Cluster*

Create a Migration Location

The migration location is implemented as a CRD (CustomResourceDefinition) and is cluster scoped. After the CSI driver has been deployed, you can verify the existence of the CRD by running the following command:

```
# kubectl get crd migrationlocations.hedvig.io

NAME                              CREATED AT
migrationlocations.hedvig.io      2020-05-17T01:15:16Z
```

A migration location can be created by specifying the name of the Hedvig cluster and the seeds, for example:

```
apiVersion: hedvig.io/v1
kind: MigrationLocation
metadata:
  name: ml-hedvig-a
spec:
  hedvig:
    name: "hedvig-a"
    seeds: ["hedvig-a.seed.1", "hedvig-a.seed.2", "hedvig-a.seed.3"]
```

Create a Snapshot Schedule (and Snapshot Class)

The data migration process is snapshot based. See the *Scheduled Snapshots* section to create a snapshot schedule for data migration.

When a persistent volume snapshot expires, data migration is triggered from the source cluster to one or more migration locations (target Hedvig clusters).

Create a Storage Class with Migration Location and Snapshot Schedule

After the migration location(s) and snapshot schedule have been created, create a new storage class by setting the following parameters:

- `migrationEnable` - Set to `"true"`
- `migrationLocations` - Comma-separated list of one or more migration location names
- `schedulePolicy` - Snapshot schedule name

Note: Migration is not currently supported on volumes with deduplication enabled.

Here is an example:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-hedvig-migration
provisioner: io.hedvig.csi
parameters:
  backendType: "hedvig-block"
  migrationEnable: "true"
  migrationLocations: "ml-hedvig-a,ml-hedvig-b"
  schedulePolicy: "migration-schedule"
```

Create a Persistent Volume Claim using the Storage Class

After the storage class has been created, create a PVC.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-migration
  annotations:
    volume.beta.kubernetes.io/storage-class: sc-hedvig-migration
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

Access the Migrated Persistent Volume on the Target Cluster

When the migration is successful, the target Hedvig cluster should have a virtual disk with the same name as that in the source Hedvig cluster corresponding to the persistent volume.

Here are the steps to mount the migrated persistent volume on the target Kubernetes cluster.

Register the Migrated Virtual Disk to the Kubernetes Cluster

On the Hedvig target cluster CLI, run the following command:

```
hedvig-target> addkubevolume -n <virtual-disk-name> -i <kube-cluster-id>
```

`<virtual-disk-name>` – Refers to the name of the migrated virtual disk

`<kube-cluster-id>` – Refers to the id of the target Kubernetes cluster

Create a PersistentVolume

Create a PersistentVolume corresponding to the migrated virtual disk. Here is an example:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: <virtual-disk-name>
spec:
  capacity:
    storage: <virtual-disk-size>
  accessModes:
    - ReadWriteOnce
csi:
    driver: io.hedvig.csi
    volumeAttributes:
      internalName: <virtual-disk-name>
      name: <virtual-disk-name>
      protocol: block
    volumeHandle: <virtual-disk-name>
    readOnly: false
    fsType: <filesystem-type>
  storageClassName: <storage-class-name>
  volumeMode: Filesystem
```

Create a PersistentVolumeClaim Corresponding to the PersistentVolume

Create a PersistentVolumeClaim corresponding to the PersistentVolume. Here is an example:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-migrated-volume
  annotations:
    volume.beta.kubernetes.io/storage-class: <storage-class-name>
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <virtual-disk-size>
  volumeName: <virtual-disk-name>
```

Snapshot the PersistentVolumeClaim

See the *Snapshots and Clones* section for the steps to snapshot a PersistentVolumeClaim.

Create a Clone from the Volume Snapshot

See the *Snapshots and Clones* section for the steps to clone from a volume snapshot.

When the clone operation is complete, a deep clone is created from the volume snapshot corresponding to the migrated volume.

The persistent volume claim can now be mounted on any pod.

Appendix A: Hedvig CSI Configuration for MicroK8s Clusters

In MicroK8s clusters, the default kubelet path is typically set to:

/var/snap/microk8s/common/`var/lib/kubelet`

instead of the default path of `/var/lib/kubelet` in traditional Kubernetes clusters.

After generating the configuration files for the CSI driver, update the CSI node server configuration (`config/hedvig-csidaemonset.yaml`) by prefixing the following paths with **/var/snap/microk8s/common**:

- `/var/lib/kubelet/plugins/io.hedvig.csi/csi.sock`
- `/var/lib/kubelet/plugins/io.hedvig.csi`
- `/var/lib/kubelet/plugins_registry/`
- `/var/lib/kubelet`

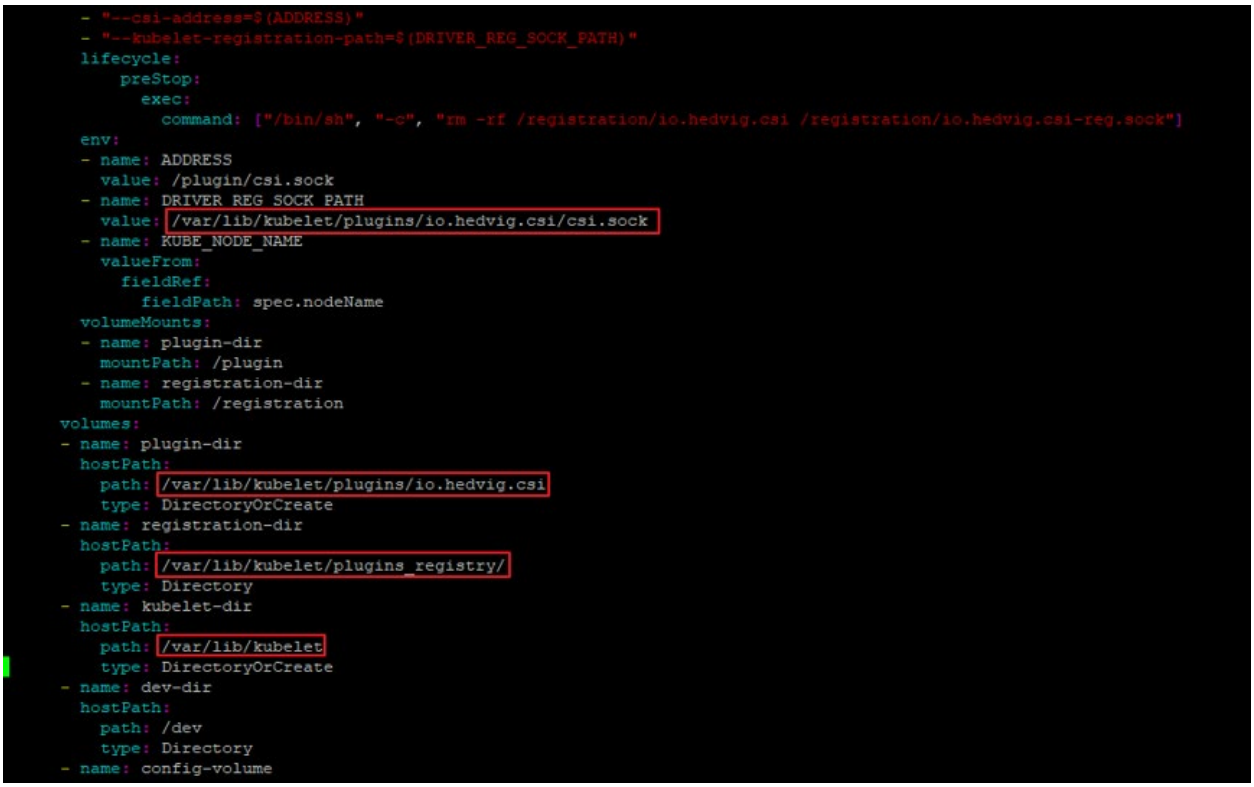

Figure 1: "Before" updating references in the CSI node server configuration

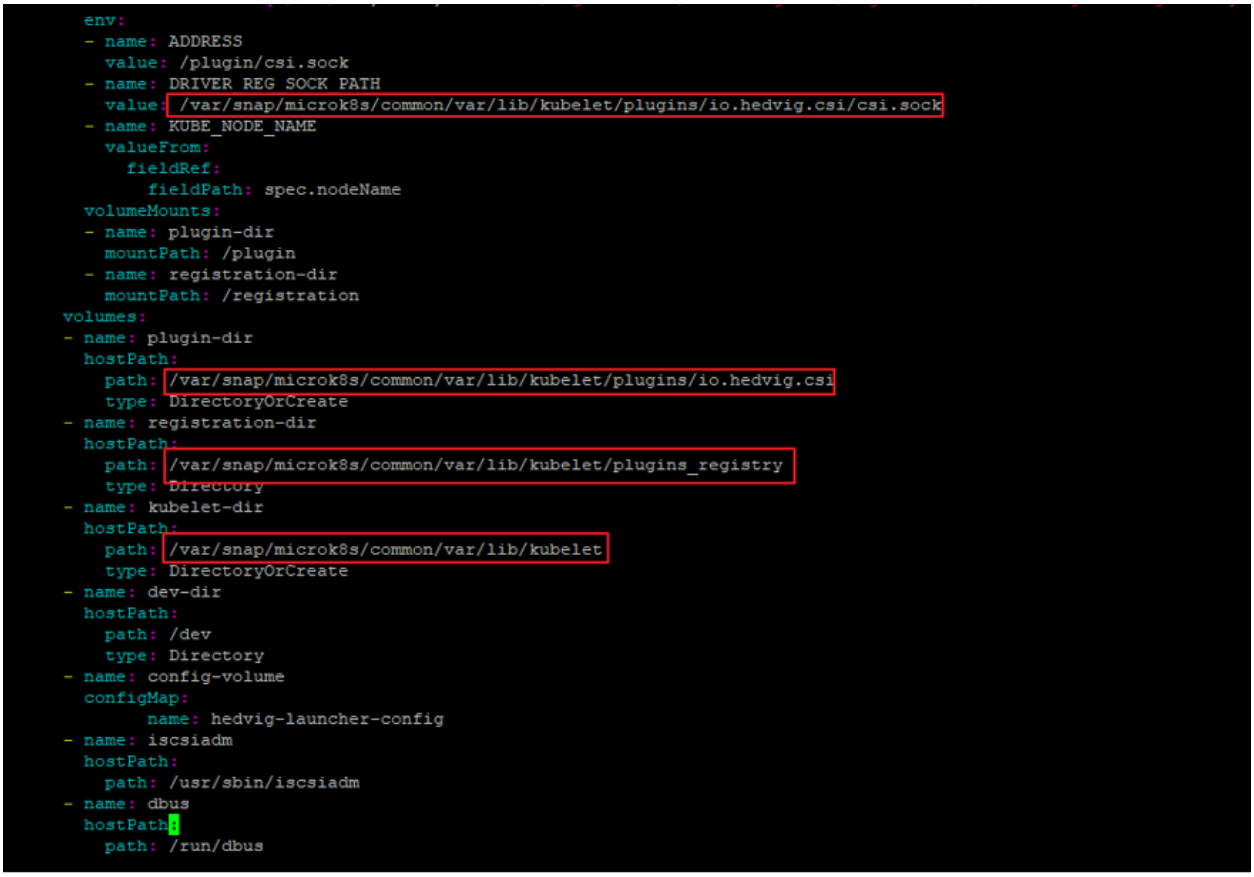

Figure 2: "After" updating references in the CSI node server configuration

After this update, you will be able to successfully install the CSI driver.

Appendix B: Hedvig Block Volumes with Rancher Kubernetes Clusters

In a Kubernetes cluster setup using Rancher, the iSCSI initiator is embedded in the kubelet, which is created using the rancher/hyperkube Docker image.

In most situations, the kubelet should be able to discover and create iSCSI sessions with the Hedvig iSCSI target (Hedvig proxy daemonset) for dynamically provisioned Hedvig Block volumes.

In some instances, the iSCSI initiator embedded in the kubelet might not be compatible with the Hedvig iSCSI target. In these instances, it is recommended that you follow the steps detailed in the Rancher documentation to reconfigure the kubelet.