Hitachi NAS CSI Driver for Kubernetes

V1.2.0

User Guide

This document describes how to use the Hitachi NAS (HNAS) CSI driver for Kubernetes, and explains the parameters used in the configuration files

MK-92HNAS091-03

September 2022

Table of Contents

Preface

About this document

This document describes how to use the Hitachi NAS (HNAS) CSI driver for Kubernetes, and explains the parameters used in the configuration files.

Document conventions

This document uses the following typographic convention:

Convention	Description
Bold	• Indicates text in a window, including window titles, menus, menu options, buttons, fields, and labels. Example: **Click OK**. • Indicates emphasized words in list items.
Italic	Indicates a document title or emphasized words in text.
`Monospace`	Indicates text that is displayed on screen or entered by the user. Example: `pairdisplay -g oradb`

Intended audience

This document is intended for system administrators, Hitachi Vantara representatives, and authorized service providers who install, configure, and run the Hitachi NAS Container Storage Interface (CSI) driver for Kubernetes.

Readers of this document should be familiar with the following:

- Containerized environments and their basic functions
- Hitachi NAS server platforms (Hitachi NAS platform and NAS module)

Accessing product downloads

Product software, drivers, and firmware downloads are available on Hitachi Vantara Support Connect: https://support.hitachivantara.com/.

Log in and select Product Downloads to access the most current downloads, including updates that may have been made after the release of the product.

Getting Help

Hitachi Vantara Support Connect is the destination for technical support of products and solutions sold by Hitachi Vantara. To contact technical support, log on to Hitachi Vantara Support Connect for contact information: https://support.hitachivantara.com/en_us/contact-us.html.

Hitachi Vantara Community is a global online community for customers, partners, independent software vendors, employees, and prospects. It is the destination to get answers, discover insights, and make connections. **Join the conversation today!** Go to community.hitachivantara.com, register, and complete your profile.

Chapter 1: Introduction

Overview

The Hitachi NAS Container Storage Interface (CSI) driver is a software component that contains libraries, settings, and commands that you can used to create and manage persistent storage for your containers. It enables the stateful applications to persist and maintain data after the life cycle of the container has ended. The Hitachi NAS CSI driver provides persistent volumes on Hitachi NAS server platforms (Hitachi NAS platform and NAS module), and is able to clone those volumes, and take snapshots of them.

The Hitachi NAS CSI driver supports both static and dynamically provisioned volumes.

As the driver relies on the ability for containers/pods to access HNAS NFS exports, it can only be used on Linux based systems.

All instructions in this document assume that Kubernetes is already setup and working. For more details about Kubernetes, refer to the following documentation:

https://kubernetes.io/docs/home/

Container Storage Interface (CSI)

The Container Storage Interface (CSI) is a standard for exposing arbitrary block and file storage systems to containerized workloads on Container Orchestration Systems (COs) like Kubernetes. Using CSI, third-party storage providers can write and deploy plugins exposing new storage systems in Kubernetes without ever having to touch the core Kubernetes code. For more details, refer to the following documentation:

https://kubernetes.io/docs/concepts/storage/volumes/#csi

Please note any prerequisites for allowing CSI drivers to work within your environment – Mount Propagation must be enabled on the Kubernetes cluster, otherwise it's not possible for the CSI drivers to perform mounts on behalf of pods/containers.

Chapter 2: System Requirements

Storage Platforms

The Hitachi NAS CSI driver requires access to the HNAS REST API, and can make use of versions 4, 7 or 8 – at least one of the API versions is supported on the following platforms, using the specified software versions or newer. The use of API keys for authentication was added to a later release. To use API keys instead of a username/password combination, make sure that you are using the minimum supported software version from the table below, and using at least version 7 of the API:

Hitachi storage platform	Software version	Software version with API key support
Hitachi NAS Platform 5000	13.3	13.7
Hitachi NAS Platform 4000	13.3	13.7
VSP Nx00 with NAS Module (N400/N600/N800)	83-06-01-x0/00	83-06-08-x0/00
VSP Gx00 with NAS Module (G400/G600/G800)	83-04-61-x0/00	83-05-36-x0/00

Port Requirements

All network communications are performed over TCP port 8444, using HTTPS, between the Kubernetes cluster and the HNAS system being used to host the storage volumes.

HNAS License Requirements

The Hitachi NAS CSI driver makes use of NFS and file cloning functionality within HNAS, and requires valid license keys for both **NFS** and **File Clone.**

Kubernetes Requirements

The Hitachi NAS CSI driver makes use of Kubernetes Volume Snapshot functionality that only became stable in Kubernetes v1.20. Consequently, the installation manifest file requires APIs present in Kubernetes v1.20 or newer. Previous versions of the Hitachi NAS CSI driver did not use the snapshot functionality, so could be installed on older versions of Kubernetes.

Note: Volume Snapshot functionality may need to be installed separately into the Kubernetes environment, depending on the distribution used – this process is not covered in this document.

Chapter 3: Installation and Configuration of the Driver

Kubernetes supports the hosting of volumes on an external NFS server, but Kubernetes has no ability to manage them. The Hitachi NAS CSI driver allows storage to be dynamically allocated on HNAS systems directly from Kubernetes, as well as consuming existing storage. It can manage the creation and deletion of NFS exports and virtual volumes and allows existing data already present on the HNAS to be cloned and presented to multiple different containers/pods. Volumes can also be snapshotted and the data re-used for new volumes or to take backups. If configured to do so, the driver can also delete the volume or snapshot contents when they are no longer needed.

Backup and restore of important data contained within container images can be an issue under normal circumstances. By providing NFS mounted remote storage, it is possible to use normal backup/restore tools to ensure that the important data associated with the containers can be properly backed up and restored should it be necessary.

The driver does not get involved with the normal operation of the containers, and NFS mounted exports behave in exactly the same way as if they were created and mounted manually. The driver makes it easier to create external storage on the HNAS system, without the need to directly interact with the HNAS management interfaces.

In addition to creating specific exports on the HNAS, the driver performs the NFS mount on behalf of Kubernetes. Specific mount options can be supplied, for example, which NFS version to use. NFSv3 is recommended when mounting HNAS volumes, as it is stateless, generally faster than NFSv4, and has less overhead.

The Hitachi NAS CSI driver package contains all required components and example yaml files that demonstrate how to create each resource type. The following table summarizes the yaml files supplied:

Configuration Files for the Kubernetes CSI environment	
File	**Details**
hnas-csi-k8s1.20.yaml	Contains information that installs the HNAS CSI driver into the Kubernetes CSI framework – this file can be use on Kubernetes version 1.20 and newer. This file should not be modified except for the registry IP address and port number. Refer to the *Install Hitachi NAS CSI Driver on Kubernetes* section in this topic
hnas-secret-sample.yaml	A sample file which contains information about each HNAS system and authentication information
hnas-sc-sample.yaml	A sample file which contains information about the Kubernetes StorageClass, which specifies where the data is stored on the HNAS system, and how it's mounted

hnas-pvc-sample.yaml	A sample file which contains information that will create a new dynamic PersistentVolume and associated PersistentVolumeClaim, on an HNAS system
hnas-pvc-clone-sample.yaml	A sample file which contains information that will create a dynamic PersistentVolume and associated PersistentVolumeClaim, where the new volume is pre-populated with data cloned from an existing HNAS PersistentVolume
hnas-pod-sample.yaml	A sample file which contains information about a Kubernetes Pod which will consume a PersistentVolume created by the HNAS CSI driver
hnas-snapclass-sample.yaml	A sample file which contains information used to create a VolumeSnapshotClass, which provides details on how snapshots are created of existing PersistentVolumes on an HNAS system
hnas-snapshot-sample.yaml	A sample file which contains information that can be used to create a VolumeSnapshot from an existing PersistentVolumeClaim
hnas-pvc-from-snapshot-sample.yaml	A sample file which contains information that will create a dynamic PersistentVolume and associated PersistentVolumeClaim, where the new volume is pre-populated with data from an existing HNAS VolumeSnapshot
hnas-static-pv-sample.yaml	A sample file which contains information that allows an existing HNAS NFS export to be used as a static PersistentVolume
hnas-static-pvc-sample.yaml	A sample file which contains information that will create a PersistentVolumeClaim on a static HNAS PersistentVolume

Install Hitachi NAS CSI Driver on Kubernetes

Because the Hitachi NAS CSI driver is not currently available from a public Docker registry, it is necessary to upload the image to a local registry so that it can be installed from there. This should be done via Docker.

1. Extract `hnas-csi-v1.2.0.tgz`:

```
# tar -xvf hnas-csi-v1.2.0.tgz
```

2. Load the Hitachi NAS CSI driver image into Docker:

```
# docker load < hnas-csi-driver-v1.2.0.tgz
```

3. Push the Hitachi NAS CSI driver image to your local registry:

```
# REGISTRY=<registry_ip>:<port>
# docker tag hitachi/hnas-csi-driver:1.2.0 ${REGISTRY}/hitachi/hnas-driver:1.2.0
# docker push ${REGISTRY}/hitachi/hnas-driver:1.2.0
```

4. Configure `hnas-csi-k8s1.20.yaml` to specify your local registry in the installation manifest:

```
# cd deploy/kubernetes
# sed -i.bak -e "s/__REGISTRY__/${REGISTRY}/" hnas-csi-k8s1.20.yaml
```

5a. Complete the following step for a **new install**:

```
# kubectl create -f hnas-csi-k8s1.20.yaml
```

5b. Complete the following step to **upgrade an existing install.** Note there should be no need to delete any existing configuration or volumes that are already in use:

```
# kubectl replace --force -f hnas-csi-k8s1.20.yaml
```

6. Verify whether the Hitachi NAS CSI driver is running – it may take a short while before the pods finish starting – the command output shown below is for a single node Kubernetes cluster, so only includes one instance of the `hnas-csi-node`:

```
# kubectl get pod -n kube-system | grep hnas
hnas-csi-controller-76c864f9f5-m5rf5   6/6     Running   0          62s
hnas-csi-node-qv22j                    3/3     Running   0          62s
```

Check Installed Components

Each component of the Hitachi NAS CSI driver is made up from multiple images. Check that the following images are present and in the running state. These should exist within the **kube-system** namespace:

hnas-csi-controller – single instance

The CSI controller is responsible for the creation, deletion and management of the volumes on the HNAS system. It also supplies the volume information to Kubernetes.

```
csi-provisioner     k8s.gcr.io/sig-storage/csi-provisioner:v2.2.2
external-attacher   k8s.gcr.io/sig-storage/csi-attacher:v3.4.0
csi-resizer         k8s.gcr.io/sig-storage/csi-resizer:v1.4.0
csi-snapshotter     k8s.gcr.io/sig-storage/csi-snapshotter:v4.2.1
liveness-probe      k8s.gcr.io/sig-storage/livenessprobe:v2.6.0
hnas-driver         <your registry>/hitachi/hnas-driver:1.2.0
```

hnas-csi-node – an instance on each Kubernetes worker node

Each CSI node is responsible for mounting and unmounting the volumes on the required pods/containers on the specific Kubernetes node they reside on. This is why there is a CSI node running on each Kubernetes node.

```
driver-registrar    k8s.gcr.io/sig-storage/csi-node-driver-registrar:v2.5.0
liveness-probe      k8s.gcr.io/sig-storage/livenessprobe:v2.6.0
hnas-driver         <your registry>/hitachi/hnas-driver:1.2.0
```

It is not possible to test if the driver is working until it has been configured.

Configuring the Hitachi NAS system

All communications between the Kubernetes cluster and the HNAS system is done using the HNAS REST API. To check if the API is enabled, use the HNAS `rest-server-status` command. If the REST server is not running, use the HNAS `rest-server-start` command to enable it.

Create a new API key with the HNAS `apikey-create` command – this is the preferred authentication method.

```
hnas:$ apikey-create csi-driver
Please make a note of this new API Key, as it is not possible to display the
full key again.
Only the prefix and description can be displayed in the future.
New key: xIAdbgTNVP.Nj2TOgxiOYgpTu2kjzEGS4QmIJIeLmF3aXKg6FhY9vC
```

Alternatively, a new user can be created, using the HNAS `user` command. The example below will create a new supervisor level user called `csi-driver` with a password of `abcd1234`:

```
hnas:$ user add csi-driver abcd1234 SUPERVISOR
```

Configuring the Hitachi NAS CSI Driver

Once the Hitachi NAS CSI driver is installed and running, the following tasks need to be performed before the HNAS can be used to host storage volumes:

Configure HNAS details – The address and credentials for each HNAS system to be used must be stored within Kubernetes **secret** files. Storing the details this way means they are not easily accessible by other processes. Refer to the *Secret Settings* in the *Configuration Files* section.

Configure one or more StorageClass resources – The details about the HNAS filesystem are configured in the StorageClass. A different StorageClass may be created depending on the underlying disk types, data retain policy, backup policy, etc. The StorageClass name should make it easy to distinguish between the different types. Refer to *StorageClass Settings* in the *Configuration Files* section.

Configure one or more VolumeSnapshotClass resources if you need to be able to take snapshots of the HNAS hosted volumes. Refer to the *VolumeSnapshotClass Settings* in the *Configuration Files* section.

Static Volumes

Static volumes refer to storage that already exists before being configured within Kubernetes. An existing HNAS NFS export can be made available for consumption within Kubernetes as a PersistentVolume.

For a static volume to make use of the other CSI driver features, cloning, snapshotting, etc, it must be associated with a StorageClass, as that is where the HNAS access details are stored. Static volumes to be used in this way **should not** be configured with NFS exports that export the root of an HNAS filesystem, as that may cause unexpected failures or unexpected results when attempting snapshots or clones. The StorageClass must share the same HNAS filesystem as the existing NFS export being used to create the static volume.

If none of the advanced functionality is needed for the static volume, then do not associate it with a StorageClass. The root of an HNAS filesystem may be exported in this case, and used as a static volume.

Dynamic Volumes

Dynamic volumes are created on demand by Kubernetes, and generally the PersistentVolume and PersistentVolumeClaim are created at the same time. The volume is created on the HNAS filesystem specified in the StorageClass that is used to create the new volume.

Dynamic volumes allow space to be allocated/deleted on a HNAS system directly from the Kubernetes environment, and do not require an administrator to manually create them.

Limitations

File Cloning and Snapshotting

The Hitachi NAS CSI driver uses HNAS file clones for both cloning and snapshotting of volumes, and is not able to clone/snapshot the following file types:

- Files that are not regular (excluding symlinks) i.e. sockets, FIFOs, block special devices, character devices
- Hard links, cross volume links

If these files are encountered within a volume being cloned or snapshotted, they will be ignored, and will not be reported as errors. For more details about this behavior, refer to the *`tree-clone-job-submit`* *man page* via the *HNAS Command Line Interface.*

This is a limitation of the HNAS file cloning functionality.

Virtual Volumes

The Hitachi NAS CSI driver makes use of HNAS tree delete functionality to remove volumes and snapshots. This functionality can be used to remove regular folders, and virtual volumes as a whole, but not folders within virtual volumes. Virtual volumes are also used by the Hitachi NAS CSI driver to enforce size restrictions/limits on dynamic volumes that it hosts, and it is not possible to create a virtual volume within an existing virtual volume. As a consequence, none of the root folder locations used for hosting volumes or snapshots should be within a virtual volume – this refers to the `folder-prefix` value.

For more details about the deletion behavior, refer to the *`tree-delete-job-submit` man page* via the *HNAS Command Line Interface.*

This is a limitation of the HNAS tree delete functionality.

Chapter 4: Using The Driver

This section describes how to work with volumes and snapshots managed by the Hitachi NAS CSI driver using the `kubectl` command.

Create a New Static Volume

An existing NFS export must already be present on an HNAS system before creating the static volume. The **name**, **volumeHandle**, **server** and **share** parameters must be specified in the yaml file - refer to *PersistentVolume Settings* in the *Configuration Files* section, and optionally the PersistentVolume can be associated with a StorageClass.

```
# kubectl create -f hnas-static-pv-sample.yaml
```

Once the PersistentVolume has been created within Kubernetes, a PersistentVolumeClaim needs to be made on the volume, before it can be consumed.

```
# kubectl create -f hnas-static-pvc-sample.yaml
```

Once the PersistentVolumeClaim has been bound, the static volume can be used by Kubernetes in the same way as any other PersistentVolumeClaim.

Create a New Dynamic Volume

CSI drivers perform this as a two stage process, which first involves the creation of a PersistentVolume, followed by a PersistentVolumeClaim on that volume. Most of the parameters that specify how the new volume is created are inherited from the StorageClass. The **name**, **size** and if the volume is to be shared are the parameters that must be specified in the yaml file - refer to *PersistentVolumeClaim Settings* in the *Configuration Files* section.

```
# kubectl create -f hnas-pvc-sample.yaml
```

Once a CSI driver has created a PersistentVolumeClaim, it can be used by Kubernetes in the same way as any other PersistentVolumeClaim.

Expand an Existing Dynamic Volume

If the StorageClass is created with the allowVolumeExpansion setting as true, then it is possible to expand the volume once it has been created. There is no need to delete and recreate the Pod for volume expansion. The following example expands the existing PersistentVolumeClaim, `pvc-sample-hnas`, to `5Gi`.

```
# kubectl patch pvc pvc-sample-hnas --patch
'{"spec":{"resources":{"requests":{"storage": "5Gi"}}}}'
```

Delete a Dynamic Volume

When the volume is deleted, the HNAS is instructed to delete the NFS export. If the PersistentVolume has a `reclaimPolicy` of Delete (this parameter is defined in the StorageClass but can be changed once the volume is created), the HNAS will be instructed to delete all files contained within the exported folder. This is done as a background process, using HNAS tree delete functionality.

The example below deletes the PersistentVolumeClaim called `pvc-sample-hnas`:

```
# kubectl delete pvc pvc-sample-hnas
```

Using a Volume

To create a new pod that makes use of a PersistentVolumeClaim as a volume, which has been created by the Hitachi NAS CSI driver, refer to the *Pod Settings* section in the *Configuration Files* section.

This process is identical to the way it would be done if the PersistentVolumeClaim were created manually.

Creating a Volume Snapshot

When a new snapshot is created, two objects are associated with it – a VolumeSnapshot and a VolumeSnapshotContent. The VolumeSnapshotClass specifies some of the snapshot parameters, but the yaml file specifies the snapshot name and existing volume to snapshot - refer to *VolumeSnapshost Settings* in the *Configuration Files* section.

```
# kubectl create -f hnas-snapshot-sample.yaml
```

Note that snapshots created by the CSI driver are not the same as an HNAS filesystem snapshot.

Deleting a Volume Snapshot

When the snapshot is deleted, the `deletionPolicy` setting from the VolumeSnapshotClass determines whether the snapshot contents are deleted or retained. If the deletion policy is set to Delete, the HNAS will be instructed to delete all files contained within the snapshot, otherwise the files are retained on the HNAS. The deletion is done as a background process, using HNAS tree delete functionality.

The example below deletes the VolumeSnapshot called `snapshot-hnas`:

```
# kubectl delete volumesnapshot snapshot-hnas
```

Deleting a snapshot does not affect any volumes that were created using the snapshot as a data source.

Creating a Volume from an Existing Source

CSI drivers allow new volumes to be created with a copy of the data from an existing volume or from an existing snapshot. The Hitachi NAS CSI driver supports creating a new volume as a clone of an existing volume, and also creating a new volume from a copy of the data contained within a snapshot of an existing volume. For details on both these options, refer to *PersistentVolumeClaim Settings* in the *Configuration Files* section.

The creation command is the same as creating a new, empty volume (see *Create a New Volum*e section) and can be used in exactly the same way as any other volume.

Chapter 5: Configuration Files

This section contains details about each of the configuration files that are used to allow the Hitachi NAS CSI driver to function within Kubernetes.

Secret Settings

The Secret file contains the HNAS REST API URL and credentials that are necessary for the Hitachi NAS CSI driver to work with your environment. The following sample provides information about the required and optional parameters. Refer to the *Troubleshooting* section for an example of how to verify that the REST API is functioning correctly.

Parameter references for hnas-secret-sample.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: hnas-system-1  #(1)
type: Opaque
stringData:
  apiurl: "https://172.16.1.1:8444"   #(2)
  apiversion: "7"  #(3)
data:
  apikey: 4oCTbiBlbmNvZGUtbWUK  #(4)
  username: 4oCTbiBlbmNvZGUtbWUK  #(5)
  password: 4oCTbiBlbmNvZGUtbWUK  #(6)
```

Legend:

(1) **name** – used to uniquely identify the HNAS system. If multiple HNAS systems are to be used, then create multiple secret files
(2) **apiurl** – HNAS REST API URL, including protocol and port number
(3) **apiversion** – HNAS REST API version to be used – versions 4, 7 and 8 are currently supported. This parameter is optional, and defaults to version 7 if omitted
(4) **apikey** – base64 encoded HNAS API Key – this parameter is optional, and can only be used with **apiversion** 7 or greater

If the **apikey** parameter is specified, then the following parameters are ignored:
(5) **username** – base64 encoded HNAS supervisor level username
(6) **password** – base64 encoded HNAS password

Example of base64 encoding a parameter:

```
# echo -n "encode-me" | base64
4oCTbiBlbmNvZGUtbWUK
```

The output from the echo command should be used within the secret file instead of the plain text version.

StorageClass Settings

The StorageClass file contains parameters related to the hosting filesystem, how it's to be mounted, and some details about how Kubernetes can manage it. The following sample provides information about the required parameters.

Parameter references for hnas-sc-sample.yaml

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: hnas-gold  #(1)
  annotations:
    kubernetes.io/description: High speed HNAS storage - Gold  #(2)
provisioner: hnas.csi.hitachi.com
reclaimPolicy: Delete  #(3)
volumeBindingMode: Immediate  #(4)
allowVolumeExpansion: true  #(5)
mountOptions:  #(6)
- "nfsvers=3"
parameters:
  filesystem: "fs1"  #(7)
  folderPrefix: "/csi"  #(8)
  accessConfig: "*(norootsquash)"  #(9)
  adminEmail: "email@example.com"  #(10)
  preferredAddress: "192.168.0.100"  #(11)
  csi.storage.k8s.io/provisioner-secret-namespace: "default"  #(12)
  csi.storage.k8s.io/provisioner-secret-name: "hnas-system-1"  #(13)
  csi.storage.k8s.io/controller-expand-secret-namespace: "default"  #(12)
  csi.storage.k8s.io/controller-expand-secret-name: "hnas-system-1"  #(13)
```

Legend:

(1) **name** – name of the new StorageClass
(2) **description** – a description to be associated with the StorageClass (optional)
(3) **reclaimPolicy** – Delete or Retain. If the StorageClass reclaim policy is Delete, all data associated with a volume will be deleted when the volume is deleted, but if the policy is set to Retain, it is the responsibility of the Administrator to manually delete the data
(4) **volumeBindingMode** – Immediate or WaitForFirstConsumer. When the mode is set to WaitForFirstConsumer, any new volumes made with the StorageClass are not actually created on the HNAS until the first time it is used, otherwise it will be created immediately
(5) **allowVolumeExpansion** – true or false. If true, the volume can be expanded once it has been created.
(6) **mountOptions** – these are the NFS client mount options – see below for more details
(7) **filesystem** – HNAS filesystem, where the new volumes will be created
(8) **folderPrefix** – Filesystem folder name, where all volumes will be created. This folder must not be within a virtual volume.
(9) **accessConfig** – NFS export mount options – refer to the *HNAS documentation* for a full list of options
(10) **adminEmail** – a comma separated list of email addresses, which will receive notifications when the storage is getting full – a warning message will be sent at 75% full and a severe message at 95% full.
(11) **preferredAddress** – optional EVS IP address to use for connecting to the HNAS server. If omitted, a reachable EVS IP address will be chosen for the NFS mount

(12) Namespace used to store the secrets file that contains the HNAS system details
(13) Name specified in the HNAS system secrets file – this associates the StorageClass with a specific HNAS system

mountOptions – more details

If the **mountOptions** are omitted, by default NFSv4 will be used to mount the HNAS NFS export – use the HNAS CLI command `nfs-max-supported-version` to check if NFSv4 is enabled of not, otherwise the mount will fail. If NFSv3 is required, then the `nfsvers=3` option should be specified.

For details on other mount options refer to the *Linux man page* for *mount*.

VolumeSnapshotClass Settings

The VolumeSnapshotClass file contains parameters related to the folder location to store the snapshots, and some details about how Kubernetes can manage them. The following sample provides information about the required parameters.

Parameter references for hnas-snapclass-sample.yaml

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: hnas-snapclass  #(1)
driver: hnas.csi.hitachi.com
deletionPolicy: Delete  #(2)
parameters:
  folderPrefix: "/csi-snapshots"  #(3)
  exportPrefix: "snapshots"  #(4)
  csi.storage.k8s.io/snapshotter-secret-namespace: "default"  #(5)
  csi.storage.k8s.io/snapshotter-secret-name: "hnas-system-1"  #(6)
```

Legend:

(1) **name** – name of the new VolumeSnapshotClass
(2) **deletionPolicy** – Delete or Retain. If the VolumeSnapshotClass deletion policy is Delete, all data associated with the snapshot will be deleted when the snapshot is deleted, but if the policy is set to Retain, it is the responsibility of the Administrator to manually delete the data
(3) **folderPrefix** – Filesystem folder name, where all snapshots will be created. Each snapshot will be stored on the same filesystem as it's associated volume. This folder must not be within a virtual volume, otherwise later deletion of the snapshot data by the CSI Driver will not possible.
(4) **exportPrefix** – Optional prefix to use for the NFS export that allows access to the snapshots. If omitted, all snapshots will be grouped under the same NFS export.
(5) Namespace used to store the secrets file that contains the HNAS system details
(6) Name specified in the HNAS system secrets file – this associates the VolumeSnapshotClass with a specific HNAS system

PersistentVolume Settings

The PersistentVolume file contains information related to an existing HNAS NFS export that is to be used as a static volume, rather than allowing the Hitachi NAS CSI driver to create a new dynamic one.

Parameter references for hnas-static-pv-sample.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: static-pv-hnas  #(1)
spec:
  capacity:
    storage: 123Gi  #(2)
  accessModes:
  - ReadWriteMany  #(3)
  persistentVolumeReclaimPolicy: Retain  #(4)
  mountOptions:  #(5)
  - "nfsvers=3"
  storageClassName: "hnas-gold"  #(6)
  csi:
    driver: hnas.csi.hitachi.com
    volumeAttributes:
      server: 172.27.146.72  #(7)
      share: 2_fs2  #(8)
    volumeHandle: dc9b1d2e-5ab4-11d1-9008-75728f512f9c  #(9)
```

Legend:

(1) **name** – meaningful name of the PersistentVolume, which will be used when creating a PersistentVolumeClaim.
(2) **capacity** – size that Kubernetes reports as being available for this volume. No check is performed as to whether this value matches the space available on the existing volume.
(3) **accessModes** – this specifies how the PersistentVolume can be accessed, and can be one of ReadWriteOnce, ReadOnlyMany, ReadWriteMany
(4) **persistentVolumeReclaimPolicy** – should always be set to Retain, otherwise the CSI driver may attempt to delete the existing data when the PV is deleted.
(5) **mountOptions** – these are the NFS client mount options – see the *StorageClass Settings* in the *Configuration Files* section for more details
(6) **storageClassName** – either the name of an existing StorageClass, which is being used to create dynamic volumes on the same HNAS filesystem, as this PersistentVolume is hosted, or an empty string. An empty string will not associate the PersistentVolume with any StorageClass. Omitting the storageClassName parameter completely will associate the PV with the default StorageClass, which is probably not the desired setting.
(7) **server** – EVS IP address to use for connecting to the HNAS server.
(8) **share** – the name of an existing NFS export hosted by the EVS referred to in the **server** parameter. This value **must not** start with a / character.
(9) **volumeHandle** – the volume handle needs to be the **Global identifier** parameter associated with the HNAS NFS export, and can be retrieved either via the HNAS CLI or via the REST API.

If a `storageClassName` is supplied, and the NFS export being used is not exporting the root of a filesystem, it should be possible to create clones and snapshots of the static volume, in the same way that they can be created for dynamic volumes.

To obtain the NFS export global identifier needed for the `volumeHandle` parameter, use the following HNAS CLI command, which is retrieving the **Global identifier** associated with the NFS export called **2_fs2**. The CLI must be in the correct EVS context:

```
hnas[EVS02]:$ nfs-export list -v 2_fs2 | grep Global
      Global identifier: dc9b1d2e-5ab4-11d1-9008-75728f512f9c
```

PersistentVolumeClaim Settings

The PersistentVolumeClaim file contains volume information that is used by the Hitachi NAS CSI driver to create new dynamic PersistentVolumes, or claim static PesistentVolumes. The dynamically created volumes can either be empty, or populated with data cloned from an existing PersistentVolume of the same class and size, or the contents of a snapshot.

The following sample provides information about the required parameters for a new, empty PersistentVolumeClaim.

Parameter references for hnas-pvc-sample.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sample-hnas  #(1)
spec:
  accessModes:
  - ReadWriteMany  #(2)
  resources:
    requests:
      storage: 1Gi  #(3)
  storageClassName: hnas-gold  #(4)
```

Legend:

(1) **name** – meaningful name, which will be used to access the PersistentVolumeClaim from a pod
(2) **accessModes** – this specifies how the PersistentVolumeClaim can be accessed, and can be one of ReadWriteOnce, ReadOnlyMany, ReadWriteMany
(3) **storage** – initial size to be allocated for the storage. If the StorageClass allows expansion, this can be changed later
(4) **storageClassName** – the name of the StorageClass used to create the PersistentVolumeClaim

The following file shows how to create a new PersistentVolumeClaim, which is created by cloning the contents of the volume called `pvc-sample-hnas`, as the data source. This creates a second, writable copy of the source data.

Parameter references for hnas-pvc-clone-sample.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-clone-hnas  #(1)
spec:
  accessModes:
  - ReadWriteMany  #(2)
  resources:
    requests:
      storage: 1Gi  #(3)
  storageClassName: hnas-gold  #(4)
  dataSource:
    name: pvc-sample-hnas  #(5)
    kind: PersistentVolumeClaim
    apiGroup: ""
```

Legend:

(1), (2), (3), (4) – same as definitions above
(5) **dataSource name** – the name of an existing PersistentVolumeClaim of the same StorageClass. Parameter (3) should be either the same size of the existing volume being used as the data source, or can be a larger value.

The following file shows how to create a new PersistentVolumeClaim, which is created by cloning the contents of the VolumeSnapshot called `snapshot-hnas`, as the data source. This creates a writable copy of the source snapshot data.

Parameter references for hnas-pvc-from-snapshot-sample.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snapshot  #(1)
spec:
  accessModes:
  - ReadWriteMany  #(2)
  resources:
    requests:
      storage: 1Gi  #(3)
  storageClassName: hnas-gold  #(4)
  dataSource:
    name: snapshot-hnas  #(5)
    kind: VolumeSnapshot
    apiGroup: "snapshot.storage.k8s.io"
```

Legend:

(1), (2), (3), (4) – same as definitions above
(5) **dataSource name** – the name of an existing VolumeSnapshot of the same StorageClass. Parameter (3) should be either the same size as the source of the snapshot, or can be a larger value.

The following file shows how to create a PersistentVolumeClaim for a statically created HNAS PersistentVolume. Creating the PersistentVolumeClaim is required before the volume can be consumed.

Parameter references for hnas-static-pvc-sample.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sample-hnas  #(1)
spec:
  accessModes:
  - ReadWriteMany  #(2)
  resources:
    requests:
      storage: 1Gi  #(3)
  storageClassName: ""  #(4)
  volumeName: static-pv-hnas  #(5)
```

Legend:

(1), (2) – same as definitions above
(3) **storage** – parameter is generally ignored for a static volume, as the size reported by Kubernetes is determined by the PersistentVolume. However, the parameter must be supplied, and valid, otherwise the creation will fail.
(4) **storageClassName** – must match the storageClassName used when the PersistentVolume was created – the example here shows how to specify a blank storage class.
(5) **volumeName** – name of the PersistentVolume the claim is being made against.

VolumeSnapshot Settings

The VolumeSnapshot file contains information that is used by the Hitachi NAS CSI driver to create a new VolumeSnapshot from an existing PersistentVolumeClaim. The following sample provides information about the required parameters for creating a new snapshot.

Parameter references for hnas-snapshot-sample.yaml

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: snapshot-hnas  #(1)
spec:
  volumeSnapshotClassName: hnas-snapclass  #(2)
  source:
    persistentVolumeClaimName: pvc-sample-hnas  #(3)
```

Legend:

(1) **name** – the name of the new snapshot
(2) **volumeSnapshotClassName** – the name of the VolumeSnapshotClass used when creating the snapshot
(3) **persistentVolumeClaimName** – name of the PersistentVolumeClaim that is to be snapshotted

Pod Settings

The Pod file contains container and volume information that is used by Kubernetes to create a Pod that uses the PersistentVolumeClaim as a mounted volume within the container. The following sample provides information about the required parameters.

Parameter references for hnas-pod-sample.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-sample  #(1)
spec:
  containers:
  - name: my-busybox
    image: busybox
    volumeMounts:
    - name: hnas-volume
      mountPath: "/data"  #(2)
    command: ["sleep", "1000000"]
    imagePullPolicy: IfNotPresent
  volumes:
  - name: hnas-volume
    persistentVolumeClaim:
      claimName: pvc-sample-hnas  #(3)
```

Legend:

(1) **name** – the name of the new Pod
(2) **mountPath** – a path within the pod where the HNAS volume will be mounted. The name should match the name within the volumes section, which references the PersistentVolumeClaim
(3) **claimName** – is the name of the PersistentVolumeClaim that is to be used with this pod

Chapter 6: Data Locations and Admin Tasks

One of the advantages of storing the volumes on an HNAS is that it's possible to access the volume data from other systems, outside of Kubernetes. This can be useful for pre-populating the volumes and backing up the data. It also allows data to be shared/manipulated by clients that are not run within a containerized environment.

Populating a Volume with Data

Once a PersistentVolumeClaim has been created for a dynamic volume, it's possible to populate it with data before it's mounted by a pod/container. Either access the NFS export associated with the PersistentVolumeClaim directly (provided the `accessConfig` options specified in the StorageClass allow it) – its name will correspond to the name of the PersistentVolumeClaim, and will be of the form:

```
/pvc-9161d39b-82f6-417e-b435-2a79fc0ca49c
```

It's also possible to access the folder hosting the PersistentVolumeClaim if a higher level mount point exists on the same filesystem. The StorageClass `filesystem` and `folderPrefix` values can be combined to determine the location of PersistentVolumeClaims. All PersistentVolumeClaims will be stored in individual directories under the directory specified by the `folderPrefix` value.

Access Volume Data for Backup

To access the data within a volume for backup purposes, use the same approach as shown in the *Populating a Volume with Data* section.

Access Volume Snapshot Data for Backup

Snapshots associated with a specific VolumeSnapshotClass are grouped together in a common location on each filesystem, specified by the `folderPrefix` value. When the first snapshot is created, an NFS export is created that exports the `folderPrefix` directory, which allows NFS access to the snapshots. The export is created to allow read-only access to the snapshots. By default, the export is named:

```
/csi-snapshots@<fs_label>
```

If the `exportPrefix` is specified in the VolumeSnapshotClass, the export will be named as below. This allows multiple snapshot classes to exist on the same filesystem.

```
/<exportPrefix>@<fs_label>
```

Do not delete the snapshot exports, otherwise the CSI driver will not be able to locate the snapshot data.

If the VolumeSnapshotClass `deletionPolicy` is set to Retain, then once the snapshot is no longer needed, the folder containing the snapshot data will need to be manually removed.

Deletion of the data can be done using the HNAS `tree-delete-job-submit` CLI command or from another client.

Manual Deletion of Volume Data

If a PersistentVolumeClaim has a Reclaim Policy of Retain, the data will not automatically get deleted when the PersistentVolumeClaim is deleted in Kubernetes. The NFS export associated with the PersistentVolumeClaim and Virtual Volume will also be retained on the HNAS and need to be removed using the HNAS management interface once they are no longer needed.

Deletion of the data can be done using the HNAS `tree-delete-job-submit` CLI command or from another client, accessing the data in the same way as shown in the *Populating a Volume with Data* section.

Update HNAS Credentials

The credentials used to access the HNAS systems management interface can be updated by editing the appropriate secret.yaml file and reapplying it to Kubernetes. The Hitachi NAS CSI driver will pick up the new credentials when it is next used.

```
# kubectl replace -f hnas-secret.yaml
```

Uninstall CSI Driver

To uninstall the Hitachi NAS CSI driver, perform the following steps below.

Procedure

1. Delete all Pods which are using the volumes created by the Hitachi NAS CSI driver.

2. Delete all PersistentVolumeClaims, PersistentVolumes, VolumeSnapshots, StorageClasses, VolumeSnapshotClasses and any Secrets related to the Hitachi NAS CSI driver.

3. Delete the Hitachi NAS CSI driver:

```
# kubectl delete -f hnas-csi-k8s1.20.yaml
```

Chapter 7: Troubleshooting

As a first step to troubleshooting any issues, ensure the HNAS REST API is enabled, as described in *Chapter 3*, and that the Kubernetes nodes are able to ping any IP addresses used in the configuration files.

Obtaining Logs

All the components that make up the Hitachi NAS CSI driver can have their log level changed to help with troubleshooting. The log levels are from 0 (minimal logging) to 9 (large amount of logging, including full HTTP request). Level 2 is recommended for normal operation, and Level 5 is recommended for troubleshooting.

The log level for each component is specified when they are originally created – so unless it's needed, they should not be changed.

If the driver does not appear to be working correctly, the first places to look are the console logs:

1. If the new volumes are not being created, then look at the logs associated with the `hnas-driver` container, hosted as part of the **CSI Controller** pod.

2. If there are issues mounting the volumes by newly created pods, look at the logs associated with the `hnas-driver` container, hosted as part of the **CSI Node** pod on the Kubernetes node hosting the new pod.

The following example shows how to get the logs from the `hnas-driver` container from one of the CSI Node drivers:

```
# kubectl get pods -n=kube-system
NAME                                   READY   STATUS    RESTARTS   AGE
coredns-66bff467f8-clgb5               1/1     Running   1          21d
coredns-66bff467f8-gtqj6               1/1     Running   1          21d
etcd-minikube                          1/1     Running   1          21d
hnas-csi-controller-67ccb6748f-j24pq   6/6     Running   0          20d
hnas-csi-node-4n59t                    3/3     Running   0          20d
kube-apiserver-minikube                1/1     Running   2          21d
kube-controller-manager-minikube       1/1     Running   2          21d
kube-proxy-hvvr8                       1/1     Running   1          21d
kube-scheduler-minikube                1/1     Running   1          21d
storage-provisioner                    1/1     Running   1          21d
# kubectl logs -n=kube-system hnas-csi-node-4n59t hnas-driver
```

Hitachi NAS CSI Driver cannot connect to the HNAS server

The connection parameters and credentials can be checked using a simple curl command, run on one of the Kubernetes nodes. Run the following command, and replace the `<username>`, `<password>` and `<admin_IP>` parameters with the ones to be used by the CSI driver. If the command works, it should return some information about the cluster nodes, and some additional debug info associated with the connection. If the credentials are wrong, a message should be returned with a `"401 Unauthorized"` error code. If the `<admin_IP>` address is wrong, the connection should either be refused, or timeout.

```
curl -v -k -H "X-Subsystem-User: <username>" -H "X-Subsystem-Password:
<password>" https://<admin_IP>:8444/v7/storage/nodes
```

If an API key is to be used instead of user/password combination, use the following curl command, replacing the `<api_key>` and `<admin_IP>` parameters as appropriate:

```
# curl -v -k -H "X-Api-Key: <api_key>" https://<admin_IP>:8444/v7/storage/nodes
```

For more details on troubleshooting connectivity issues to the REST API, refer to the *"Hitachi NAS File Storage, REST API Reference - MK-92HNAS088"*.

Kubernetes node unable to NFS mount the HNAS server

The most likely cause of a mount failure is an NFS version mismatch between the HNAS server and the Kubernetes node. By default, HNAS systems are not configured to support NFSv4, and if no mount options are specified in the StorageClass definition, the Kubernetes node will attempt to use NFSv4 to mount the HNAS export. There are two solutions to this issue - enable NFSv4 on the HNAS or specify NFSv3 as an option when creating the StorageClass. NFSv3 as a mount option should be the preferred solution, as enabling NFSv4 on an HNAS system that does not already have it enabled, could potentially cause other NFS clients to mount via NFSv4 when next mounted, which may have unintentional behavior changes.