Integrating Hedvig with Rancher-managed Kubernetes clusters

Document version 1.0

Table of contents

Prerequisites

General

- Hedvig Storage Cluster (version 3.0 or later)
- Rancher Setup (version 2.0 or later)
- One or more Kubernetes clusters (version 1.7 or later)
- Hedvig Dynamic Provisioner installer package

Network

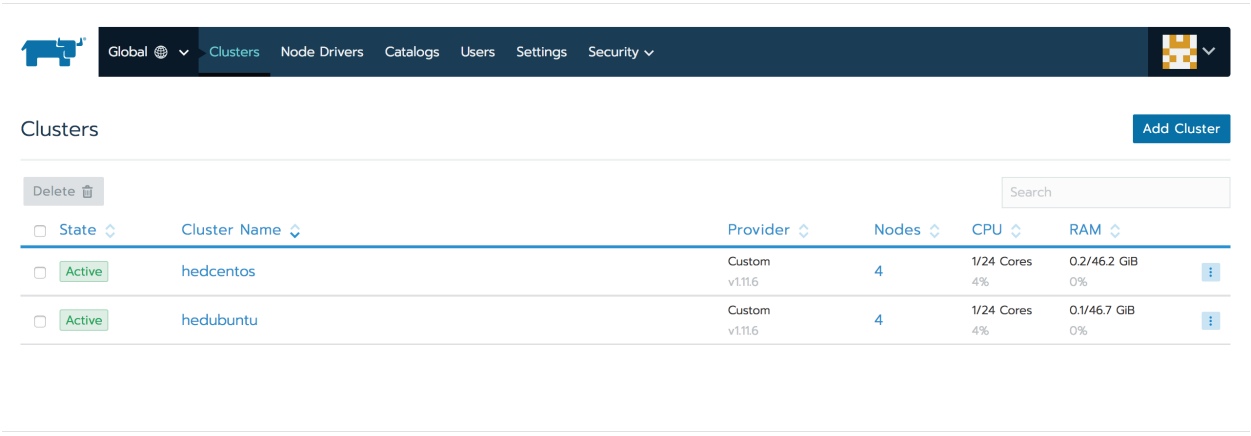
- If a firewall is enabled on the Kubernetes worker nodes, for the list of ports that need to be unblocked, see *Appendix A: List of Ports.*
- Ensure that all Kubernetes worker nodes and the Hedvig Storage Cluster Nodes can communicate with each other.

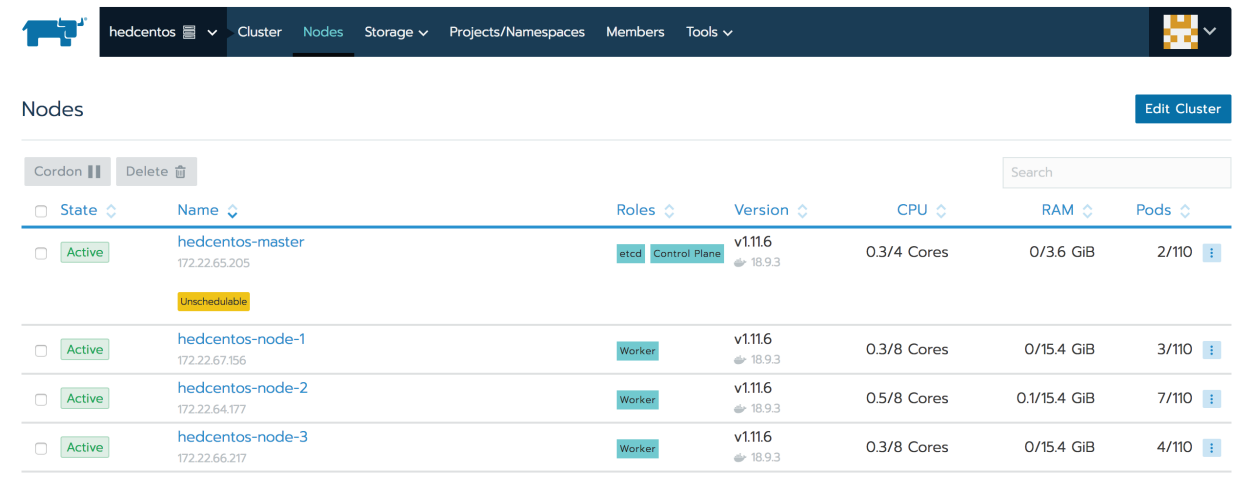
Setup for Rancher

This document assumes that Rancher has already been set up to manage at least one Kubernetes cluster.

The steps outlined in the following sections should be executed for each Kubernetes cluster managed by Rancher to enable Hedvig persistent storage for stateful applications in the Kubernetes cluster(s).

For illustration, this document will describe how to integrate Hedvig with the Kubernetes cluster `hedcentos` in the following Rancher setup.





Setup instructions for Hedvig Storage Proxies

Setting up caches on all Kubernetes worker nodes

In the Rancher setup used for reference, execute the following steps on the Kubernetes worker nodes: `hedcentos-node-1`, `hedcentos-node-2`, and `hedcentos-node-3`.

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

```
[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
```

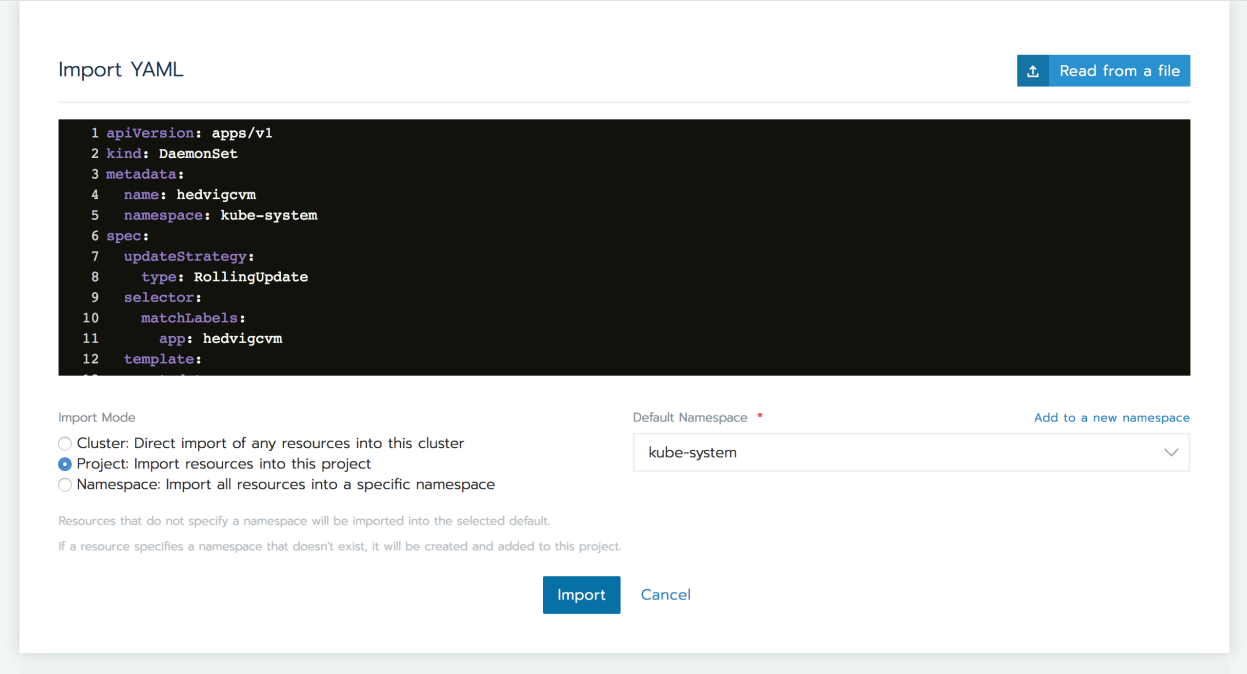
3. Verify that `/hedvig/cache` is present by running the following command.

```
df -kh
```

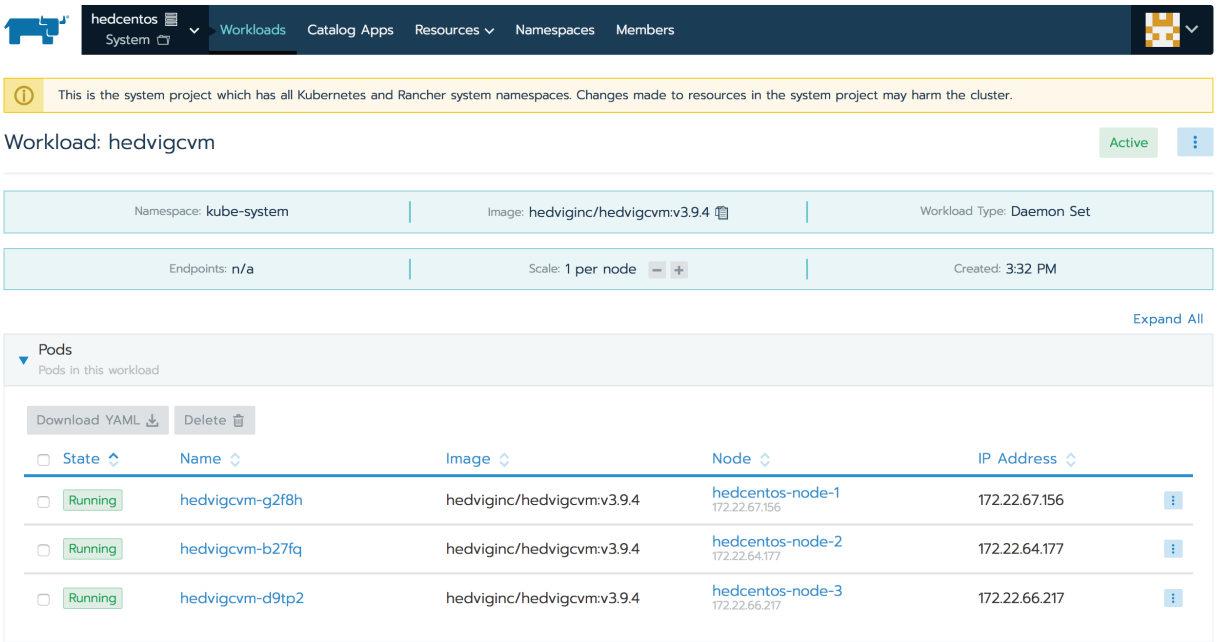
Installing the DaemonSet on all Kubernetes worker nodes

1. Locate the DaemonSet manifest template `manifests/ds/daemonset.yml.tmpl` in the Hedvig Dynamic Provisioner installer package.
2. Copy this file into `daemonset.yml` and update the values for the following fields.
   - `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 cluster (for example, in the reference setup, this is set to `hedcentos`).
3. Update the image name in `daemonset.yaml` to `hedviginc/hedvigcvm:<tag>` and set the `<tag>` to the software version installed on the Hedvig Storage Cluster. A complete list of available versions can be found here.

4. Deploy the DaemonSet by importing the `daemonset.yaml` file into the `kube-system` namespace of the Kubernetes cluster using the Rancher UI.



5. Verify that the DaemonSet has been successfully deployed by looking for a System workload with the name `hedvigcvm` in the Kubernetes cluster. There should be exactly one `hedvigcvm` Pod on each Kubernetes worker node in the Kubernetes cluster.



Setup instructions for the Hedvig Dynamic Provisioner

1. Create a ServiceAccount.

   Create a ServiceAccount for the Hedvig Dynamic Provisioner by importing the `hedvig-serviceaccounts.yaml` (from the installer package) into the `kube-system` namespace.

   Verify that the ServiceAccount has been successfully created by launching the `kubectl` console.

2. Create a ClusterRole.

   Create a ClusterRole for the Hedvig Dynamic Provisioner by importing the `hedvig-clusterroles-k8s.yaml` (from the installer package) into the `kube-system` namespace.

   Verify that the ClusterRole has been successfully created by launching the `kubectl` console.

3. Create a ClusterRoleBinding.

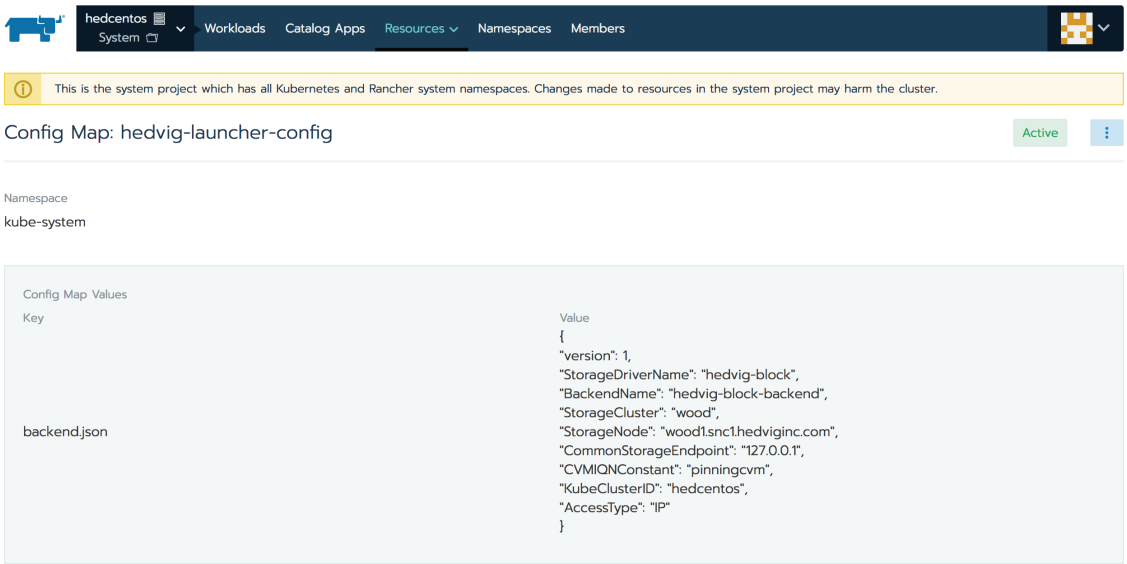
   Create a ClusterRoleBinding for the Hedvig Dynamic Provisioner by importing the `hedvig-clusterrolebindings-k8s.yaml` (from the installer package) into the `kube-system` namespace. Verify that the ClusterRoleBinding has been successfully created by launching the `kubectl` console.

4. Create a ConfigMap.

   Update the following configuration values in the `setup/backend.json` file to point to your Hedvig Storage Cluster:

   - `StorageCluster` — name of the Hedvig Storage Cluster
   - `StorageNode` — hostname/IP address of one of the Hedvig Storage Cluster Nodes
   - `KubeClusterID` — unique id for the Kubernetes cluster (for example, in the reference setup, this is set to `hedcentos`)
   - `AccessType` — set to `"IP"`

   Create a ConfigMap with the name `hedvig-launcher-config` in the `kube-system` namespace in the Kubernetes cluster with `backend.json` as the key and the contents of that file as the value.



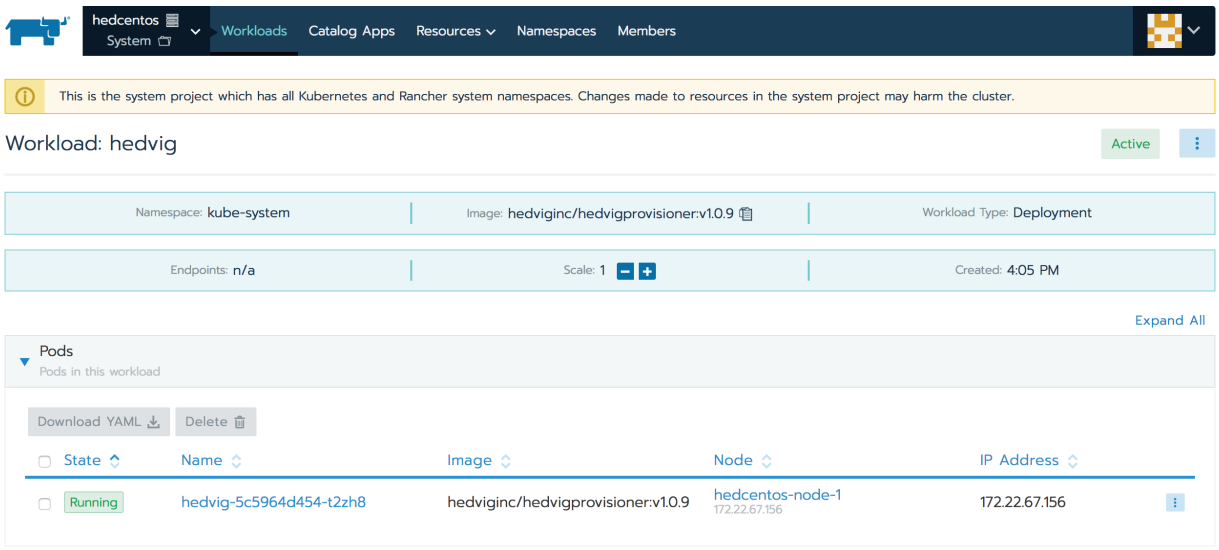
5. Deploy the Hedvig Dynamic Provisioner.

   Update the image name in `hedvig-deployment.yaml` to `hedviginc/hedvigprovisioner:<tag>` and set the `<tag>` to the most recently released version of the Hedvig Dynamic Provisioner.

   A complete list of available versions can be found here.

   Deploy the Hedvig Dynamic Provisioner by importing the `hedvig-deployment.yaml` file into the `kube-system` namespace of the Kubernetes cluster.

   Verify that the Hedvig Dynamic Provisioner has been successfully deployed by looking for a System workload with name `hedvig` in the Kubernetes cluster.



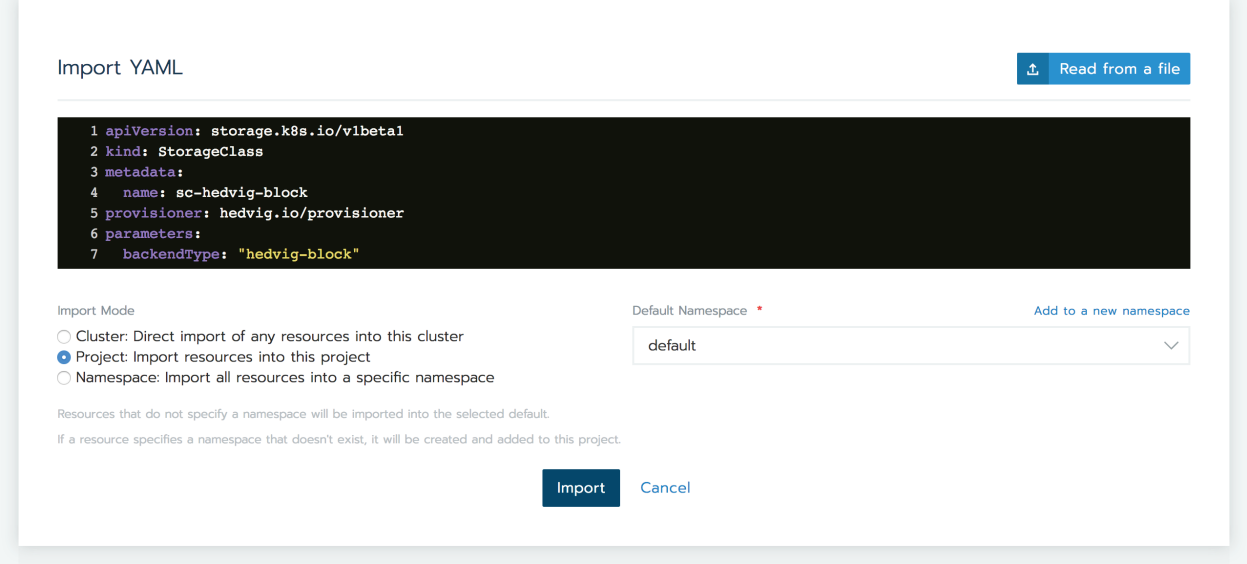
Storage Classes and Hedvig Virtual Disk Attributes

Before provisioning persistent volumes, at least one storage class must be created.

A storage class can be created by providing a unique name for the storage class and specifying the provisioner to be used for the storage class.

Creating a Storage Class for Persistent Volumes (PVs)

The following manifest creates a default storage class for Hedvig by importing the storage class manifest file.

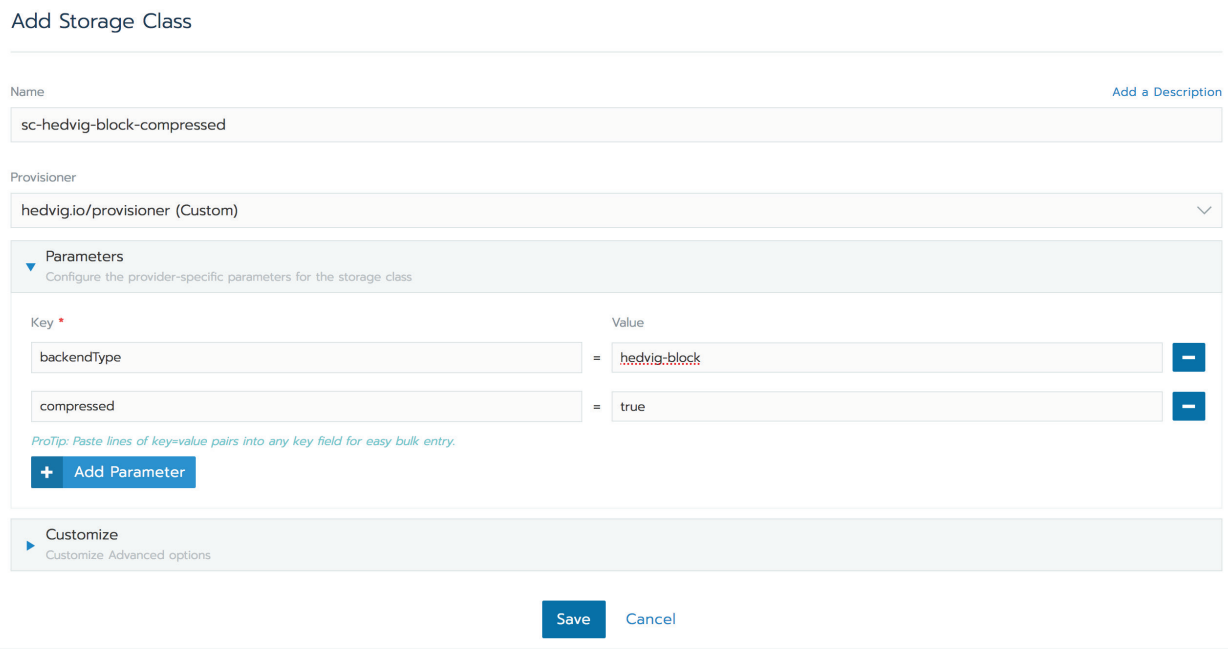


Note: The `backendType` should match exactly the storage driver name mentioned in `backend.json`.

Customizing Storage Classes with Hedvig Virtual Disk attributes

Storage classes can be customized by providing Hedvig Virtual Disk attributes as parameters.

The following manifest creates a storage class with compression enabled for persistent volumes.



The following table lists all of the Hedvig Virtual Disk attribute keys and their possible values.

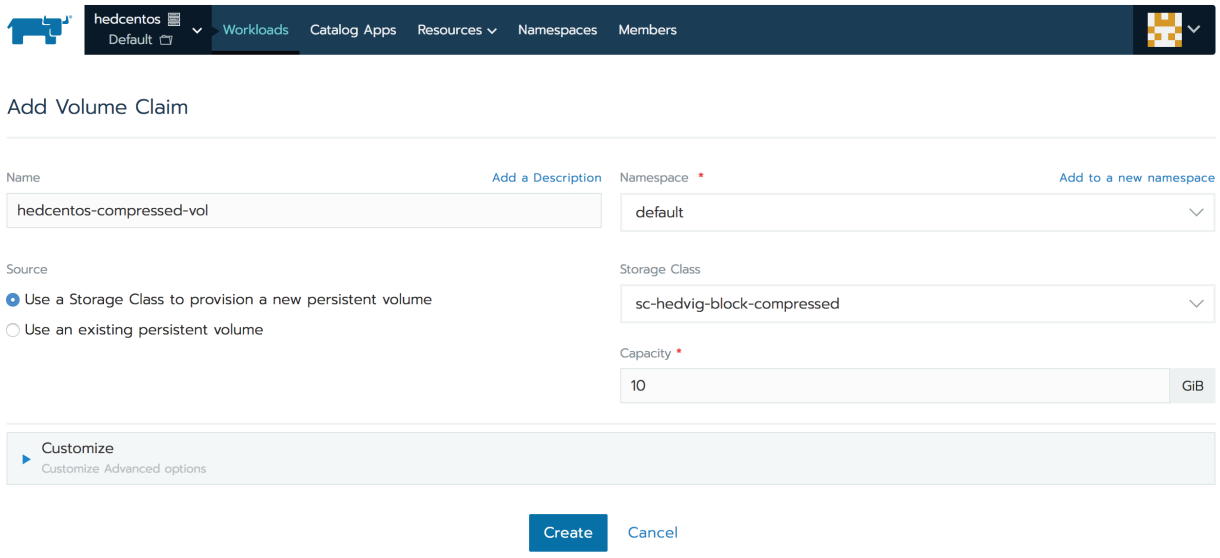
Table 1: Hedvig Virtual Disk attribute keys

key	values	default value	notes
dedupEnable	true/false	false	
compressed	true/false	false	
cacheEnable	true/false	false	
rf	1 to 6	3	
rp	Agnostic/RackAware/ DataCenterAware	Agnostic	
dcNames	comma-separated list of data center names		Applies only to an rp (replication policy) of DataCenterAware
diskResidence	flash/hdd	hdd	In an all-flash cluster, diskResidence should always be set to flash
encryptionEnable	true/false	false	
blockSize	512/4096	4096	
description	any string		

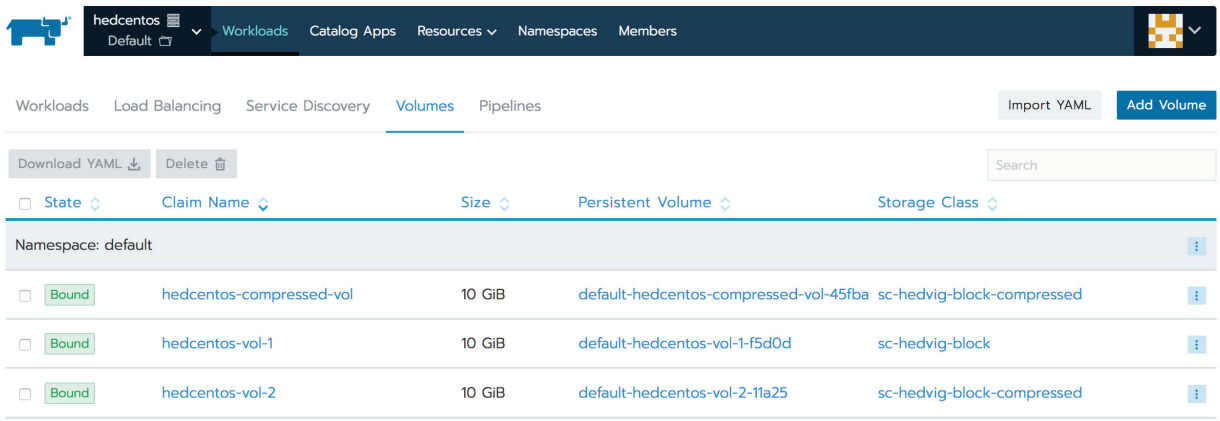
Provision Persistent Volumes

A persistent volume is dynamically provisioned on Hedvig by creating a persistent volume claim with one of the storage classes created in the previous section.

The following manifest creates a persistent volume claim for a persistent volume of size 10Gi.



When the persistent volume claim is successfully created, it should be bound to a persistent volume that is created dynamically, as shown below.



Appendix A: List of Ports

Table 2: List of Ports

protocol	port range	description
TCP	50022	ssh
TCP	3260	iscsi
TCP	50000 - 50008	thrift