



Integrating Hedvig with Rancher-managed Kubernetes clusters

Document version 1.0

Table of contents

Prerequisites.....	3
General.....	3
Network.....	3
Setup for Rancher	4
Setup instructions for Hedvig Storage Proxies	5
Setting up caches on all Kubernetes worker nodes	5
Installing the DaemonSet on all Kubernetes worker nodes	5
Setup instructions for the Hedvig Dynamic Provisioner	7
Storage Classes and Hedvig Virtual Disk Attributes	10
Creating a Storage Class for Persistent Volumes (PVs)	10
Customizing Storage Classes with Hedvig Virtual Disk attributes	11
Provision Persistent Volumes.....	13
Appendix A: List of Ports	14

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

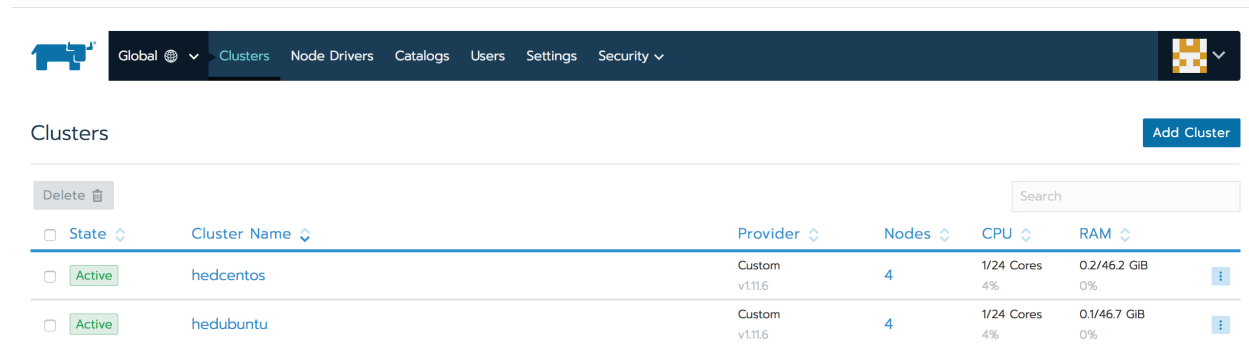
- If a firewall is enabled on the Kubernetes worker nodes, for the list of ports that need to be unblocked, see [Appendix A: List of Ports](#).
- Ensure that all Kubernetes worker nodes and the Hedvig Storage Cluster Nodes can communicate with each other.

Setup for Rancher

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

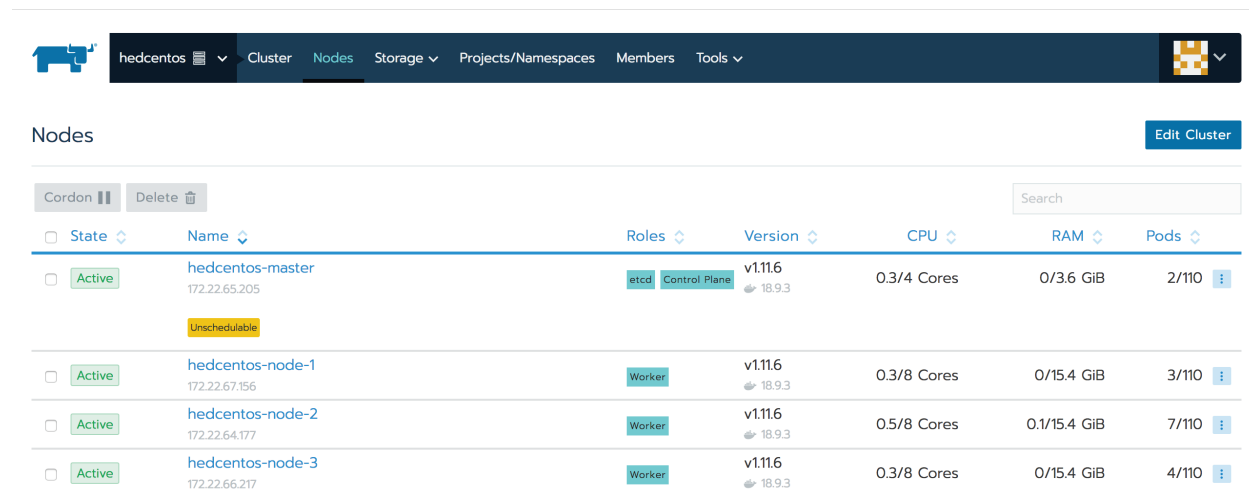
The steps outlined in the following sections should be executed for each Kubernetes cluster managed by Rancher to enable Hedvig persistent storage for stateful applications in the Kubernetes cluster(s).

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



The screenshot shows the Rancher interface with the 'Clusters' page selected. The navigation bar includes 'Global', 'Clusters', 'Node Drivers', 'Catalogs', 'Users', 'Settings', and 'Security'. The main content area shows a table of clusters with columns for State, Cluster Name, Provider, Nodes, CPU, and RAM. Two clusters are listed: 'hedcentos' and 'hedubuntu', both in an 'Active' state.

State	Cluster Name	Provider	Nodes	CPU	RAM
Active	hedcentos	Custom v1.11.6	4	1/24 Cores 4%	0.2/46.2 GiB 0%
Active	hedubuntu	Custom v1.11.6	4	1/24 Cores 4%	0.1/46.7 GiB 0%



The screenshot shows the Rancher interface with the 'Nodes' page selected for the 'hedcentos' cluster. The navigation bar includes 'hedcentos', 'Cluster', 'Nodes', 'Storage', 'Projects/Namespace', 'Members', and 'Tools'. The main content area shows a table of nodes with columns for State, Name, Roles, Version, CPU, RAM, and Pods. Four nodes are listed: 'hedcentos-master' (Control Plane) and three worker nodes ('hedcentos-node-1', 'hedcentos-node-2', 'hedcentos-node-3').

State	Name	Roles	Version	CPU	RAM	Pods
Active	hedcentos-master 172.22.65.205	etcd, Control Plane	v1.11.6 18.9.3	0.3/4 Cores	0/3.6 GiB	2/110
Active	hedcentos-node-1 172.22.67.156	Worker	v1.11.6 18.9.3	0.3/8 Cores	0/15.4 GiB	3/110
Active	hedcentos-node-2 172.22.64.177	Worker	v1.11.6 18.9.3	0.5/8 Cores	0.1/15.4 GiB	7/110
Active	hedcentos-node-3 172.22.66.217	Worker	v1.11.6 18.9.3	0.3/8 Cores	0/15.4 GiB	4/110

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.tpl` 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.yml` 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.

Import YAML Read from a file

```

1 apiVersion: apps/v1
2 kind: DaemonSet
3 metadata:
4   name: hedvigcvm
5   namespace: kube-system
6 spec:
7   updateStrategy:
8     type: RollingUpdate
9   selector:
10    matchLabels:
11     app: hedvigcvm
12   template:

```

Import Mode Default Namespace * Add to a new namespace

Cluster: Direct import of any resources into this cluster
 Project: Import resources into this project
 Namespace: Import all resources into a specific namespace

Resources that do not specify a namespace will be imported into the selected default.
 If a resource specifies a namespace that doesn't exist, it will be created and added to this project.

Import
Cancel

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.

This is the system project which has all Kubernetes and Rancher system namespaces. Changes made to resources in the system project may harm the cluster.

Workload: hedvigcvm Active

Namespace: kube-system	Image: hedviginc/hedvigcvm:v3.9.4	Workload Type: Daemon Set
Endpoints: n/a	Scale: 1 per node	Created: 3:32 PM

Expand All

▼ Pods
Pods in this workload

Download YAML Delete

State	Name	Image	Node	IP Address
Running	hedvigcvm-g2f8h	hedviginc/hedvigcvm:v3.9.4	hedcentos-node-1 172.22.67.156	172.22.67.156
Running	hedvigcvm-b27fq	hedviginc/hedvigcvm:v3.9.4	hedcentos-node-2 172.22.64.177	172.22.64.177
Running	hedvigcvm-d9tp2	hedviginc/hedvigcvm:v3.9.4	hedcentos-node-3 172.22.66.217	172.22.66.217

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.

Shell: hedcentos

Connected

```
> kubectl describe serviceaccount hedvig -n kube-system
Name:          hedvig
Namespace:    kube-system
Labels:        <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"v1","kind":"ServiceAccount","metadata":{"annotations":{"name":"hedvig","namespace":"kube-system"}}}

Image pull secrets: <none>
Mountable secrets:  hedvig-token-dq2xc
Tokens:             hedvig-token-dq2xc
Events:             <none>
>
```

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.

Shell: hedcentos

Connected

```
> kubectl describe clusterrole hedvig -n kube-system
Name:          hedvig
Namespace:    kube-system
Labels:        <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"rbac.authorization.k8s.io/v1","kind":"ClusterRole","metadata":{"annotations":{"name":"hedvig","namespace":""},"rules":[{"apiGroups":["...

PolicyRule:
  Resources                Non-Resource URLs  Resource Names      Verbs
-----
events                    []                  []                  [watch create update patch]
persistentvolumeclaims    []                  []                  [get list watch update patch]
persistentvolumeclaims/status  [patch]
persistentvolumes        []                  []                  [get list watch create delete patch]
secrets                   []                  []                  [get list watch create delete]
customresourcedefinitions.apiextensions.k8s.io  []                  []                  [create get]
storageclasses.storage.k8s.io  []                  []                  [get list watch]
volumesnapshotdatas.volumesnapshot.external-storage.k8s.io  []                  []                  [get list watch create update patch delete]
volumesnapshots.volumesnapshot.external-storage.k8s.io  []                  []                  [get list watch create update patch delete]
>
```

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.

```

> Shell: hedcentos Connected
# Run kubectl commands inside here
# e.g. kubectl get all
> kubectl describe clusterrolebinding hedvig -n kube-system
Name: hedvig
Labels: <none>
Annotations: kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"rbac.authorization.k8s.io/v1","kind":"ClusterRoleBinding","metadata":{"annotations":{"name":"hedvig","namespace":""},"roleRef":{"apiGr...
Role:
Kind: ClusterRole
Name: hedvig
Subjects:
  Kind      Name      Namespace
  ----      -
ServiceAccount hedvig kube-system
    
```

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.

The screenshot shows the Rancher interface for a ConfigMap. At the top, there's a navigation bar with 'hedcentos' selected. Below it, a warning message states: "This is the system project which has all Kubernetes and Rancher system namespaces. Changes made to resources in the system project may harm the cluster." The ConfigMap name is 'hedvig-launcher-config' and its status is 'Active'. The namespace is 'kube-system'. The 'Config Map Values' table is as follows:

Key	Value
backendjson	{ "version": 1, "StorageDriverName": "hedvig-block", "BackendName": "hedvig-block-backend", "StorageCluster": "wood", "StorageNode": "wood1.snc1.hedviginc.com", "CommonStorageEndpoint": "127.0.0.1", "CVMIOQNConstant": "pinningcvm", "KubeClusterID": "hedcentos", "AccessType": "IP" }

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.

The screenshot shows the Rancher management interface for a workload named 'hedvig'. At the top, there is a navigation bar with 'hedcentos' and 'System' selected. Below the navigation bar, a warning message states: 'This is the system project which has all Kubernetes and Rancher system namespaces. Changes made to resources in the system project may harm the cluster.' The workload details are as follows:

- Workload:** hedvig (Active)
- Namespace:** kube-system
- Image:** hedviginc/hedvigprovisioner:v1.0.9
- Workload Type:** Deployment
- Endpoints:** n/a
- Scale:** 1 (with minus and plus buttons)
- Created:** 4:05 PM

Under the 'Pods' section, there is a table of pods in this workload:

State	Name	Image	Node	IP Address
Running	hedvig-5c5964d454-t2zh8	hedviginc/hedvigprovisioner:v1.0.9	hedcentos-node-1 172.22.67.156	172.22.67.156

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.

Import YAML
Read from a file

```

1 apiVersion: storage.k8s.io/v1beta1
2 kind: StorageClass
3 metadata:
4   name: sc-hedvig-block
5 provisioner: hedvig.io/provisioner
6 parameters:
7   backendType: "hedvig-block"

```

Import Mode

Cluster: Direct import of any resources into this cluster

Project: Import resources into this project

Namespace: Import all resources into a specific namespace

Resources that do not specify a namespace will be imported into the selected default.

If a resource specifies a namespace that doesn't exist, it will be created and added to this project.

Default Namespace * Add to a new namespace

default
▼

Import
Cancel

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.

Add Storage Class

Name Add a Description

sc-hedvig-block-compressed

Provisioner

hedvig.io/provisioner (Custom) ▼

Parameters
Configure the provider-specific parameters for the storage class

Key *		Value	
backendType	=	hedvig-block	—
compressed	=	true	—

ProTip: Paste lines of key=value pairs into any key field for easy bulk entry.

+ Add Parameter

Customize
Customize Advanced options

Save Cancel

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.

The screenshot shows the 'Add Volume Claim' form in the Hedvig UI. The form includes the following fields and options:

- Name:** hedcentos-compressed-vol
- Namespace:** default
- Source:** Use a Storage Class to provision a new persistent volume; Use an existing persistent volume
- Storage Class:** sc-hedvig-block-compressed
- Capacity:** 10 GIB
- Buttons:** Create, Cancel

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

The screenshot shows the 'Volumes' page in the Hedvig UI. The table displays the following data:

State	Claim Name	Size	Persistent Volume	Storage Class
Bound	hedcentos-compressed-vol	10 GiB	default-hedcentos-compressed-vol-45fba	sc-hedvig-block-compressed
Bound	hedcentos-vol-1	10 GiB	default-hedcentos-vol-1-f5d0d	sc-hedvig-block
Bound	hedcentos-vol-2	10 GiB	default-hedcentos-vol-2-11a25	sc-hedvig-block-compressed

Appendix A: List of Ports

Table 2: List of Ports

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

Hedvig Inc. believes the information in this publication is accurate as of its publication date. The information is subject to change without notice. The information in this publication is provided as is. Hedvig Inc. makes no representations or warranties of any kind with respect to the information in this publication and specifically disclaims implied warranties of merchantability or fitness for a particular purpose. Use, copying, and distribution of any Hedvig Inc. software described in this publication requires an applicable software license. All trademarks are the property of their respective owners.

Software-defined AES-256, FIPS compliant encryption of data in flight and at rest.