Red Hat Gluster Storage 3.1

Container-Native Storage for OpenShift Container Platform 3.3

Deploying Container-Native Storage for OpenShift Container Platform

Edition 1

Last Updated: 2017-11-17

Red Hat Gluster Storage 3.1 Container-Native Storage for OpenShift Container Platform 3.3

Deploying Container-Native Storage for OpenShift Container Platform
Edition 1

Divya Muntimadugu
Customer Content Services Red Hat
divya@redhat.com

Bhavana Mohan
Customer Content Services Red Hat
bmohanra@redhat.com

Legal Notice

Abstract

This guide describes the prerequisites and provides step-by-step instructions to deploy Container-Native Storage with OpenShift Platform.

Table of Contents

CHAPTER 1. INTRODUCTION TO CONTAINERIZED RED HAT GLUSTER STORAGE

With the Red Hat Gluster Storage 3.1 update 3 release, you can deploy Containerized Red Hat Gluster Storage in multiple scenarios. This guide provides step-by-step instructions to deploy Containerized Red Hat Gluster Storage in the following scenarios:

- Dedicated Storage Cluster - this solution addresses the use-case where the applications require both a persistent data store and a shared persistent file system for storing and sharing data across containerized applications.

  For information on creating OpenShift Container Platform cluster with persistent storage using Red Hat Gluster Storage, see https://access.redhat.com/documentation/en/openshift-enterprise/3.2/installation-and-configuration/chapter-15-configuring-persistent-storage#install-config-persistent-storage-persistent-storage-glusterfs

- Chapter 2, *Red Hat Gluster Storage Container Native with OpenShift Container Platform* - this solution addresses the use-case where applications require both shared file storage and the flexibility of a converged infrastructure with compute and storage instances being scheduled and run from the same set of hardware.

NOTE

OpenShift Enterprise 3.2 has a newer version and is renamed as OpenShift Container Platform 3.3. The instructions to deploy Red Hat Gluster Storage Container Native with OpenShift Platform and Containerized Red Hat Gluster Storage on OpenShift Enterprise cluster are the same.

CHAPTER 2. RED HAT GLUSTER STORAGE CONTAINER NATIVE WITH OPENSHIFT CONTAINER PLATFORM

This deployment delivers a hyper-converged solution, where the storage containers that host Red Hat Gluster Storage co-reside with the compute containers and serve out storage from the hosts that have local or direct attached storage to the compute containers. This solution integrates Red Hat Gluster Storage deployment and management with OpenShift services. As a result, persistent storage is delivered within an OpenShift pod that provides both compute and file storage.

Red Hat Gluster Storage Container Native with OpenShift Container Platform is built around three key technologies:

- OpenShift provides the platform as a service (PaaS) infrastructure based on Kubernetes container management. Basic OpenShift architecture is built around multiple master systems where each system contains a set of nodes.
- Red Hat Gluster Storage provides the containerized distributed storage based on Red Hat Gluster Storage 3.1.3 container. Each Red Hat Gluster Storage volume is composed of a collection of bricks, where each brick is the combination of a node and an export directory.
- Heketi provides the Red Hat Gluster Storage volume life cycle management. It creates the Red Hat Gluster Storage volumes dynamically and supports multiple Red Hat Gluster Storage clusters.

The following list provides the administrators a solution workflow. The administrators can:

- Create multiple persistent volumes (PV) and register these volumes with OpenShift.
- Developers then submit a persistent volume claim (PVC).
- A PV is identified and selected from a pool of available PVs and bound to the PVC.
- The OpenShift pod then uses the PV for persistent storage.

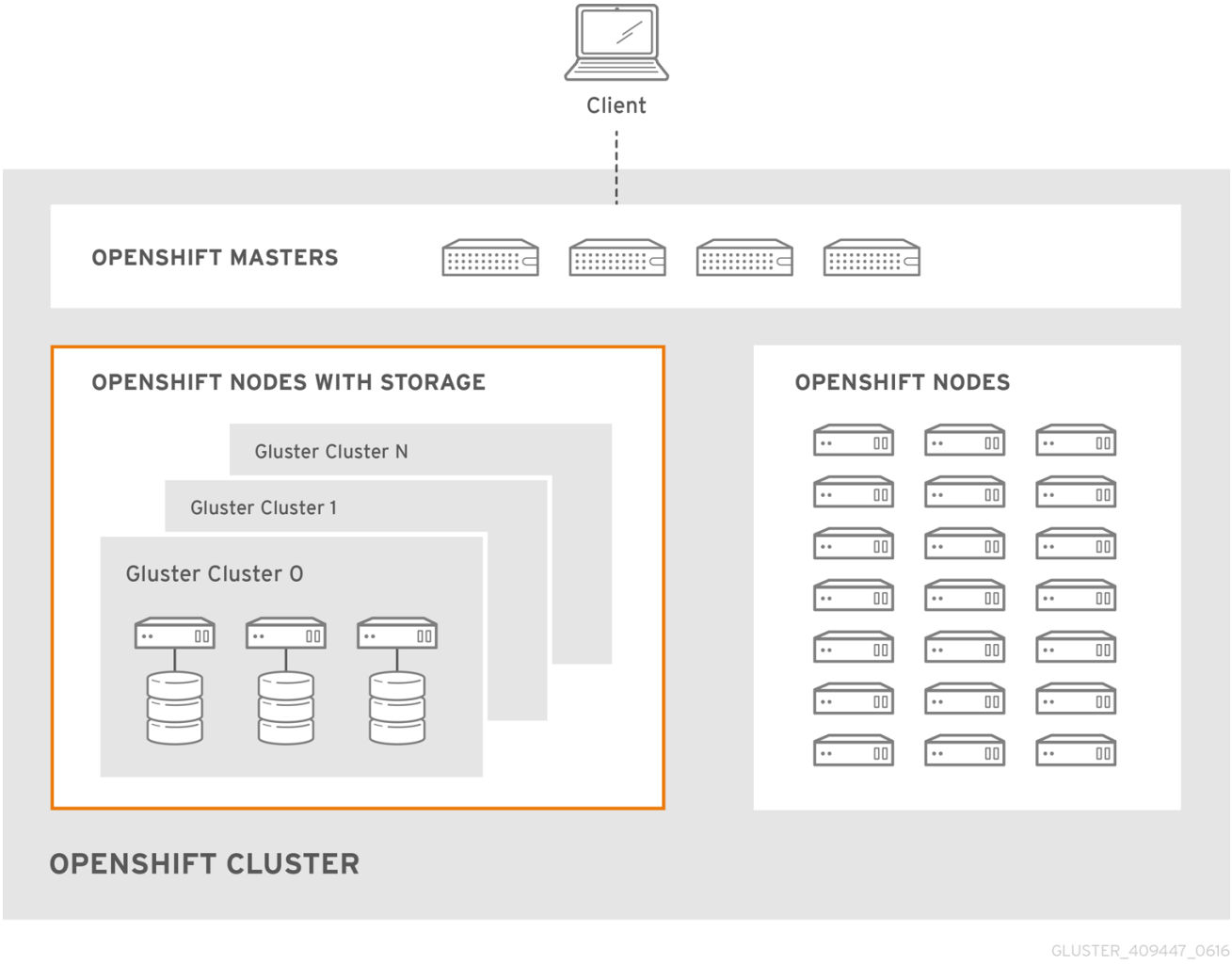


Figure 2.1. Architecture - Red Hat Gluster Storage Container Native with OpenShift Container Platform

CHAPTER 3. SUPPORT REQUIREMENTS

This chapter describes and lists the various prerequisites to set up Red Hat Gluster Storage Container Native with OpenShift Container Platform.

3.1. SUPPORTED VERSIONS

The following table lists the supported versions of OpenShift Container Platform with Red Hat Gluster Storage Server.

Table 3.1. Supported Versions

Red Hat Gluster Storage	OpenShift Container Platform
3.1.3	3.2 and later

3.2. ENVIRONMENT REQUIREMENTS

The requirements for Red Hat Enterprise Linux Atomic Host, Red Hat OpenShift Container Platform, Red Hat Enterprise Linux, and Red Hat Gluster Storage is described in this section. A Red Hat Gluster Storage Container Native with OpenShift Container Platform environment consists of Red Hat OpenShift Container Platform installed on Red Hat Enterprise Linux Atomic Host or Red Hat Enterprise Linux.

3.2.1. Installing Red Hat Gluster Storage Container Native with OpenShift Container Platform on Red Hat Enterprise Linux 7 based OpenShift Container Platform Cluster

This section describes the procedures to install Red Hat Gluster Storage Container Native with OpenShift Container Platform on Red Hat Enterprise Linux 7 based OpenShift Enterprise 3.2 or OpenShift Container Platform 3.3.

3.2.1.1. Setting up the Openshift Master as the Client

When deploying this solution on Atomic OpenShift it is necessary to deploy a client outside the cluster to install the heketi RPMs. It is optional, though an accepted best practice, to use the OpenShift Master as a client to execute the 'oc' commands across the cluster when installing OpenShift. Generally, this is setup as a non-scheduled node in the cluster. This is the default configuration when using the OpenShift installer. A user can also choose to install their client on their local machine to access the cluster remotely. For more information, see https://access.redhat.com/documentation/en/openshift-enterprise/version-3.2/cli-reference/#installing-the-cli.

Subscribe to the Red Hat Gluster Storage repository

This enables you to install the heketi client packages which are required to setup the client for Red Hat Gluster Storage Container Native with OpenShift Container Platform.

```
# subscription-manager repos --enable=rh-gluster-3-for-rhel-7-server-rpms
```

```
# yum install heketi-client heketi-templates
```

3.2.1.2. Setting up the Red Hat Enterprise Linux 7 Client for Installing Red Hat Gluster Storage Container Native with OpenShift Container Platform

To set up the Red Hat Enterprise Linux 7 client for installing Red Hat Gluster Storage Container Native with OpenShift Container Platform, perform the following steps:

Subscribe to the Red Hat Gluster Storage repository

This enables you to install the heketi client packages which are required to setup the client for Red Hat Gluster Storage Container Native with OpenShift Container Platform.

```
# subscription-manager repos --enable=rh-gluster-3-for-rhel-7-server-rpms
```

```
# yum install heketi-client heketi-templates
```

Subscribe to the OpenShift Container Platform 3.3 repository

If you are using OpenShift Container Platform 3.3, subscribe to 3.3 repository to enable you to install the Openshift client packages

```
# subscription-manager repos --enable=rhel-7-server-ose-3.3-rpms --
enable=rhel-7-server-rpms
```

```
# yum install atomic-openshift-clients
```

```
# yum install atomic-openshift
```

Subscribe to the OpenShift Enterprise 3.2 repository

If you are using OpenShift Enterprise 3.2, subscribe to 3.2 repository to enable you to install the Openshift client packages

```
# subscription-manager repos --enable=rhel-7-server-ose-3.2-rpms --
enable=rhel-7-server-rpms
```

```
# yum install atomic-openshift-clients
```

```
# yum install atomic-openshift
```

3.2.2. Installing Red Hat Gluster Storage Container Native with OpenShift Container Platform on Red Hat Enterprise Linux Atomic Host OpenShift Container Platform Cluster

Red Hat Enterprise Linux Atomic host does not support the installation of additional RPMs. Hence, an external client is required on Red Hat Enterprise Linux to install the required packages. To set up the client for Red Hat Enterprise Linux Atomic Host based installations, refer Section 3.2.1.2, “Setting up the Red Hat Enterprise Linux 7 Client for Installing Red Hat Gluster Storage Container Native with OpenShift Container Platform”

3.2.3. Red Hat OpenShift Container Platform Requirements

The following list provides the Red Hat OpenShift Container Platform requirements:

- The OpenShift cluster must be up and running.
- On each of the OpenShift nodes that will host the Red Hat Gluster Storage container, add the following rules to /etc/sysconfig/iptables in order to open the required ports:

```
-A OS_FIREWALL_ALLOW -p tcp -m state --state NEW -m tcp --dport
24007 -j ACCEPT
-A OS_FIREWALL_ALLOW -p tcp -m state --state NEW -m tcp --dport
24008 -j ACCEPT
-A OS_FIREWALL_ALLOW -p tcp -m state --state NEW -m tcp --dport 2222
-j ACCEPT
-A OS_FIREWALL_ALLOW -p tcp -m state --state NEW -m multiport --
dports 49152:49664 -j ACCEPT
```

  - Execute the following command to reload the iptables:

```
# systemctl reload iptables
```

- A cluster-admin user must be created. For more information, see Appendix A, *Cluster Administrator Setup*
- At least three OpenShift nodes must be created as the storage nodes with at least one raw device each.
- All OpenShift nodes on Red Hat Enterprise Linux systems must have glusterfs-client RPM installed.
- Ensure the nodes have the valid ports opened for communicating with Red Hat Gluster Storage. See the iptables configuration task in Step 1 of Section 4.1, "Preparing the Red Hat OpenShift Container Platform Cluster" for more information.

3.2.4. Red Hat Gluster Storage Requirements

The following list provides the details regarding the Red Hat Gluster Storage requirements:

- Installation of Heketi packages must have valid subscriptions to Red Hat Gluster Storage Server repositories.
- Red Hat Gluster Storage installations must adhere to the requirements outlined in the Red Hat Gluster Storage Installation Guide.
- The versions of Red Hat Enterprise OpenShift and Red Hat Gluster Storage integrated must be compatible, according to the information in Section 3.1, "Supported Versions" section.
- A fully-qualified domain name must be set for Red Hat Gluster Storage server node. Ensure that the correct DNS records exist, and that the fully-qualified domain name is resolvable via both forward and reverse DNS lookup.

3.2.5. Planning Guidelines

The following are the guidelines for setting up Red Hat Gluster Storage Container Native with OpenShift Container Platform.

- Ensure that the Trusted Storage Pool is not scaled beyond 100 volumes per 3 nodes per 32G of RAM.

- A trusted storage pool consists of a minimum of 3 nodes/peers.
- Distributed-Three-way replication is the only supported volume type.
- Each physical node that needs to host a Red Hat Gluster Storage peer:
  - will need a minimum of 32GB RAM.
  - is expected to have the same disk type.
  - by default the heketidb utilises 32 GB distributed replica volume.
- Red Hat Gluster Storage Container Native with OpenShift Container Platform supports up to 14 snapshots per volume.

CHAPTER 4. SETTING UP THE ENVIRONMENT

This chapter outlines the details for setting up the environment for Red Hat Gluster Storage Container Converged in OpenShift.

4.1. PREPARING THE RED HAT OPENSHIFT CONTAINER PLATFORM CLUSTER

Execute the following steps to prepare the Red Hat OpenShift Container Platform cluster:

1. On the client, execute the following command to login as the cluster admin user:

```
# oc login
```

For example:

```
# oc login

Authentication required for https://master.example.com:8443
(openshift)
Username: <cluster-admin-user>
Password: <password>
Login successful.

You have access to the following projects and can switch between
them with 'oc project <projectname>':

  * default (current)
  * management-infra
  * openshift
  * openshift-infra

Using project "default".
```

2. Execute the following command to create a project, which will contain all the containerized Red Hat Gluster Storage services:

```
# oc new-project <project name>
```

For example:

```
# oc new-project storage-project

Now using project "storage-project" on server
"https://master.example.com:8443"
```

3. Execute the following steps to set up the router:

NOTE

If a router already exists, proceed to Step 5.

1. Execute the following command on the client that is used to deploy Red Hat Gluster Storage Container Native with OpenShift Container Platform:

```
# oadm policy add-scc-to-user privileged -z router
# oadm policy add-scc-to-user privileged -z default
```

2. Execute the following command to deploy the router:

```
# oadm router storage-project-router --replicas=1
```

3. Edit the subdomain name in the config.yaml file located at /etc/origin/master/master-config.yaml .

   For example:

```
subdomain:  "cloudapps.mystorage.com”
```

4. Restart the master OpenShift services by executing the following command:

```
# systemctl restart atomic-openshift-master
```

NOTE

If the router setup fails, use the port forward method as described in Appendix B, Client Configuration using Port Forwarding.

For more information regarding router setup, see https://access.redhat.com/documentation/en/openshift-enterprise/3.2/single/installation-and-configuration/#install-config-install-deploy-router

4. After the router is running, the clients have to be setup to access the services in the OpenShift cluster. Execute the following steps to set up the DNS.

1. On the client, edit the /etc/dnsmasq.conf file and add the following line to the file:

```
address=/.cloudapps.mystorage.com/<Router_IP_Address>
```

   where, *Router_IP_Address* is the IP address of one of the nodes running the router.

NOTE

Ensure you do not edit the /etc/dnsmasq.conf file until the router has started.

2. Restart the **dnsmasq** service by executing the following command:

```
# systemctl restart dnsmasq
```

3. Edit /etc/resolv.conf and add the following line:

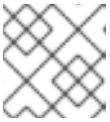
```
nameserver 127.0.0.1
```

-

For more information regarding setting up the DNS, see https://access.redhat.com/documentation/en/openshift-enterprise/version-3.2/installation-and-configuration/#prereq-dns.

5. After the project is created, execute the following command on the master node to deploy the privileged containers:

```
# oadm policy add-scc-to-user privileged -z default
```



NOTE

Red Hat Gluster Storage container can only run in privileged mode.

4.2. INSTALLING THE TEMPLATES

Execute the following steps to register the Red Hat Gluster Storage and Heketi templates with OpenShift:

1. Ensure you are in the newly created containerized Red Hat Gluster Storage project:

```
# oc project
Using project "storage-project" on server
"https://master.example.com:8443".
```

2. Execute the following command to install the templates:

```
# oc create -f /usr/share/heketi/templates
```

Example output:

```
template "deploy-heketi" created
template "glusterfs" created
template "heketi" created
```

3. Execute the following command to verify that the templates are installed:

```
# oc get templates
```

For example:

```
# oc get templates

NAME            DESCRIPTION
PARAMETERS    OBJECTS
deploy-heketi   Bootstrap Heketi installation            8 (7
blank)   3
glusterfs       GlusterFS container deployment template   1 (1
blank)   1
heketi          Heketi service deployment template        8 (7
blank)   3
```

4.3. DEPLOYING THE CONTAINERS

Execute the following commands to deploy the Red Hat Gluster Storage container on the nodes:

1. List out the hostnames of the nodes on which the Red Hat Gluster Storage container has to be deployed:

```
# oc get nodes
```

For example:

```
# oc get nodes

NAME                    STATUS                     AGE
node1.example.com       Ready                      12d
node2.example.com       Ready                      12d
node3.example.com       Ready                      12d
master.example.com      Ready,SchedulingDisabled   12d
```

2. Deploy a Red Hat Gluster Storage container on a node by executing the following command:

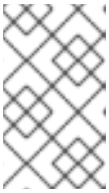
```
# oc process glusterfs -v GLUSTERFS_NODE=<node_hostname> | oc create
-f -
```

For example:

```
# oc process glusterfs -v GLUSTERFS_NODE=node1.example.com | oc
create -f -

deploymentconfig "glusterfs-dc-node1.example.com" created
```

Repeat the step of deploying the Red Hat Gluster Storage container on each node.



NOTE

This command deploys a single Red Hat Gluster Storage container on the node. This does not initialize the hardware or create trusted storage pools. That aspect will be taken care by Heketi which is explained in the further steps.

3. Execute the following command to deploy heketi:

```
# oc process deploy-heketi -v \
    HEKETI_KUBE_NAMESPACE=<Project name> \
    HEKETI_KUBE_APIHOST='<OpenShift master endpoint address>' \
    HEKETI_KUBE_INSECURE=y \
    HEKETI_KUBE_USER=<user name> \
    HEKETI_KUBE_PASSWORD=<password> | oc create -f -
```

For example:

```
# oc process deploy-heketi -v \
    HEKETI_KUBE_NAMESPACE=storage-project \
    HEKETI_KUBE_APIHOST='https://master.example.com:8443' \
```

```
    HEKETI_KUBE_INSECURE=y \
    HEKETI_KUBE_USER=test-admin \
    HEKETI_KUBE_PASSWORD=admin | oc create -f -

service "deploy-heketi" created
route "deploy-heketi" created
deploymentconfig "deploy-heketi" created
```

4. Execute the following command to verify that the containers are running:

```
# oc get pods
```

For example:

```
# oc get pods

NAME                                      READY     STATUS
RESTARTS   AGE
storage-project-router-1-lyxog            1/1       Running   0
1d
glusterfs-dc-node1.example.com-ghyta      1/1       Running   0
1m
glusterfs-dc-node2.example.com-gdfhr      1/1       Running   0
1m
glusterfs-dc-node3.example.com-nhtsa      1/1       Running   0
1m
deploy-heketi                             1/1       Running   0
1m
```

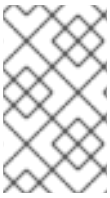
4.4. SETTING UP THE HEKETI SERVER

After deploying the containers and installing the templates, the system is now ready to load the Heketi topology file. Heketi provides a RESTful management interface which can be used to manage the lifecycle of Red Hat Gluster Storage volumes.

A sample, formatted topology file (topology-sample.json) is installed with the 'heketi-templates' package in the **/usr/share/heketi/** folder.

```
{
    "clusters": [
                {
                   "nodes": [
                                {
                         "node": {
                             "hostnames": {
                                 "manage": [
                                     "node1.example.com"
                                 ],
                                 "storage": [
                                     "192.168.121.168"
                                  ]
                              },
                              "zone": 1
                          },
```

```
                    "devices": [
                        "/dev/sdb",
                        "/dev/sdc",
                        "/dev/sdd",
                        "/dev/sde"
                    ]
            },...
```



NOTE

Edit the topology file based on the Red Hat Gluster Storage pod hostname under the **node.hostnames.manage** section and **node.hostnames.storage** section with the IP address. For simplicity, the file only sets up 4 nodes with 8 drives each.

Execute the following steps to set up the Heketi server:

1. Execute the following command to check if the bootstrap container is running:

```
# curl http://deploy-heketi-<project_name>.<sub-domain_name>/hello
```

For example:

```
# curl http://deploy-heketi-storage-
project.cloudapps.mystorage.com/hello

Hello from Heketi
```

2. Execute the following command to load the topology file:

```
# export  HEKETI_CLI_SERVER=http://deploy-heketi-<project_name>.
<sub_domain_name>
```

For example:

```
# export HEKETI_CLI_SERVER=http://deploy-heketi-storage-
project.cloudapps.mystorage.com
```

```
# heketi-cli topology load --json=topology.json
```

For example:

```
# heketi-cli topology load --json=topology.json

 Creating node node1.example.com ... ID:
95cefa174c7210bd53072073c9c041a3
        Adding device /dev/sdb ... OK
        Adding device /dev/sdc ... OK
        Adding device /dev/sdd ... OK
        Adding device /dev/sde ... OK
    Creating node node2.example.com ... ID:
f9920995e580f0fe56fa269d3f3f8428
        Adding device /dev/sdb ... OK
```

```
        Adding device /dev/sdc ... OK
        Adding device /dev/sdd ... OK
        Adding device /dev/sde ... OK
    Creating node node3.example.com ... ID:
73fe4aa89ba35c51de4a51ecbf52544d
        Adding device /dev/sdb ... OK
        Adding device /dev/sdc ... OK
        Adding device /dev/sdd ... OK
        Adding device /dev/sde ... OK
```

3. Execute the following command to verify that the topology is loaded:

```
# heketi-cli topology info
```

4. Execute the following command to create the Heketi storage volume which will store the database on a reliable Red Hat Gluster Storage volume:

```
# heketi-cli setup-openshift-heketi-storage
```

For example:

```
# heketi-cli setup-openshift-heketi-storage

Saving heketi-storage.json
```

NOTE

If the Trusted Storage Pool where the heketidbstorage volume is created is down, then the Heketi service will not work. Hence, you must ensure that the Trusted Storage Pool is up before running **`heketi-cli`**.

5. Execute the following command to create a job which will copy the database from deploy-heketi bootstrap container to the volume.

```
# oc create -f heketi-storage.json
```

For example:

```
# oc create -f heketi-storage.json

secret "heketi-storage-secret" created
endpoints "heketi-storage-endpoints" created
service "heketi-storage-endpoints" created
job "heketi-storage-copy-job" created
```

6. Execute the following command to verify that the job has finished successfully:

```
# oc get jobs
```

For example:

```
# oc get jobs
```

```
NAME                       DESIRED   SUCCESSFUL   AGE
heketi-storage-copy-job    1            1              2m
```

7. Execute the following command to deploy the Heketi service which will be used to create persistent volumes for OpenShift:

```
# oc process heketi -v \
    HEKETI_KUBE_NAMESPACE=<Project name> \
    HEKETI_KUBE_APIHOST='<OpenShift master endpoint address>' \
    HEKETI_KUBE_INSECURE=y \
    HEKETI_KUBE_USER=<user name> \
    HEKETI_KUBE_PASSWORD=<password> | oc create -f -
```

For example:

```
oc process heketi -v \
    HEKETI_KUBE_NAMESPACE=storage-project \
    HEKETI_KUBE_APIHOST='https://master.example.com:8443' \
    HEKETI_KUBE_INSECURE=y \
    HEKETI_KUBE_USER=test-admin \
    HEKETI_KUBE_PASSWORD=admin | oc create -f -


service "heketi" created
route "heketi" created
deploymentconfig "heketi" created
```

8. Execute the following command to let the client communicate with the container:

```
# export  HEKETI_CLI_SERVER=http://heketi-<project_name>.
<sub_domain_name>
```

For example:

```
# export  HEKETI_CLI_SERVER=http://heketi-storage-
project.cloudapps.mystorage.com
```

```
# heketi-cli topology info
```

9. Execute the following command to remove all the bootstrap containers which was used earlier to deploy Heketi

```
# oc delete all,job,template,secret --selector="deploy-heketi"
```

For example:

```
# oc delete all,job,template,secret --selector="deploy-heketi"

deploymentconfig "deploy-heketi" deleted
route "deploy-heketi" deleted
service "deploy-heketi" deleted
```

```
pod "deploy-heketi-1-4k1fh" deleted
job "heketi-storage-copy-job" deleted
template "deploy-heketi" deleted
```

CHAPTER 5. CREATING PERSISTENT VOLUMES

OpenShift Container Platform clusters can be provisioned with persistent storage using GlusterFS.

Persistent volumes (PVs) and persistent volume claims (PVCs) can share volumes across a single project. While the GlusterFS-specific information contained in a PV definition could also be defined directly in a pod definition, doing so does not create the volume as a distinct cluster resource, making the volume more susceptible to conflicts.

Binding PVs by Labels and Selectors

Labels are an OpenShift Container Platform feature that support user-defined tags (key-value pairs) as part of an object's specification. Their primary purpose is to enable the arbitrary grouping of objects by defining identical labels among them. These labels can then be targeted by selectors to match all objects with specified label values. It is this functionality we will take advantage of to enable our PVC to bind to our PV.

You can use labels to identify common attributes or characteristics shared among volumes. For example, you can define the gluster volume to have a custom attribute (key) named **storage-tier** with a value of **gold** assigned. A claim will be able to select a PV with **storage-tier=gold** to match this PV.

To enable persistent volume support in OpenShift and Kubernetes, few endpoints and a service must be created:

The sample glusterfs-endpoints.json file (sample-gluster-endpoint.json) and the sample glusterfs-service.json file (sample-gluster-service.json) are available at /usr/share/heketi/openshift/.

1. To specify the endpoints you want to create, update the **`sample-gluster-endpoint.json`** file with the endpoints to be created based on the environment. Each Red Hat Gluster Storage trusted storage pool requires its own endpoint with the IP of the nodes in the trusted storage pool.

```
{
  "kind": "Endpoints",
  "apiVersion": "v1",
  "metadata": {
    "name": "glusterfs-cluster"
  },
  "subsets": [
    {
      "addresses": [
        {
          "ip": "192.168.121.168"
        }
      ],
      "ports": [
        {
          "port": 1
        }
      ]
    },
    {
      "addresses": [
        {
          "ip": "192.168.121.172"
        }
      ],
```

```
      "ports": [
        {
          "port": 1
        }
      ]
    },
    {
      "addresses": [
        {
          "ip": "192.168.121.233"
        }
      ],
      "ports": [
        {
          "port": 1
        }
      ]
    }
  ]
}
```

name: is the name of the endpoint

ip: is the ip address of the Red Hat Gluster Storage nodes.

2. Execute the following command to create the endpoints:

```
# oc create -f <name_of_endpoint_file>
```

For example:

```
# oc create -f sample-gluster-endpoint.json
endpoints "glusterfs-cluster" created
```

3. To verify that the endpoints are created, execute the following command:

```
# oc get endpoints
```

For example:

```
# oc get endpoints
NAME                      ENDPOINTS
AGE
storage-project-router
192.168.121.233:80,192.168.121.233:443,192.168.121.233:1936   2d
glusterfs-cluster
192.168.121.168:1,192.168.121.172:1,192.168.121.233:1         3s
heketi                    10.1.1.3:8080
2m
heketi-storage-endpoints
192.168.121.168:1,192.168.121.172:1,192.168.121.233:1         3m
```

4. Execute the following command to create a gluster service:

```
# oc create -f <name_of_service_file>
```

For example:

```
# oc create -f sample-gluster-service.json
service "glusterfs-cluster" created
```

```
# cat sample-gluster-service.json

{
  "kind": "Service",
  "apiVersion": "v1",
  "metadata": {
    "name": "glusterfs-cluster"
  },
  "spec": {
    "ports": [
      {"port": 1}
    ]
  }
}
```

5. To verify that the service is created, execute the following command:

```
# oc get service
```

For example:

```
# oc get service
NAME                      CLUSTER-IP      EXTERNAL-IP   PORT(S)
AGE
storage-project-router    172.30.94.109   <none>
80/TCP,443/TCP,1936/TCP   2d
glusterfs-cluster         172.30.212.6    <none>        1/TCP
5s
heketi                    172.30.175.7    <none>        8080/TCP
2m
heketi-storage-endpoints  172.30.18.24    <none>        1/TCP
3m
```

NOTE

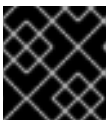
The endpoints and the services must be created for each project that requires a persistent storage.

6. Create a 100G persistent volume with Replica 3 from GlusterFS and output a persistent volume specification describing this volume to the file pv001.json:

```
$ heketi-cli volume create --size=100 --persistent-volume-
file=pv001.json
```

```
$ cat pv001.json
```

```
{
  "kind": "PersistentVolume",
  "apiVersion": "v1",
  "metadata": {
    "name": "glusterfs-f8c612ee",
    "creationTimestamp": null,
    "labels": {
      "storage-tier": "gold"
    }
  },
  "spec": {
    "capacity": {
      "storage": "12Gi"
    },
    "glusterfs": {
      "endpoints": "glusterfs-cluster",
      "path": "vol_f8c612eea57556197511f6b8c54b6070"
    },
    "accessModes": [
      "ReadWriteMany"
    ],
    "persistentVolumeReclaimPolicy": "Retain"
  },
  "status": {}
}
```



IMPORTANT

You must manually add the **Labels** information to the .json file.

Following is the example YAML file for reference:

```
apiVersion: v1
kind: PersistentVolume
metadata:
 name: pv-aplo-glusterfs1
 labels:
  storage-tier: gold
spec:
 capacity:
   storage: 12Gi
 accessModes:
   - ReadWriteMany
 persistentVolumeReclaimPolicy: Retain
 glusterfs:
   endpoints: TYPE END POINTS NAME HERE,
   path: vol_e6b77204ff54c779c042f570a71b1407
}
```

name: The name of the volume.

storage: The amount of storage allocated to this volume

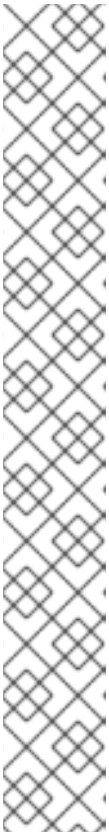
glusterfs: The volume type being used, in this case the glusterfs plug-in

endpoints: The endpoints name that defines the trusted storage pool created

path: The Red Hat Gluster Storage volume that will be accessed from the Trusted Storage Pool.

accessModes: accessModes are used as labels to match a PV and a PVC. They currently do not define any form of access control.

lables: Use labels to identify common attributes or characteristics shared among volumes. In this case, we have defined the gluster volume to have a custom attribute (key) named **storage-tier** with a value of **gold** assigned. A claim will be able to select a PV with **storage-tier=gold** to match this PV.



NOTE

- heketi-cli also accepts the endpoint name on the command line (--persistent-volume-endpoint="TYPE ENDPOINT HERE"). This can then be piped to **`oc create -f -`** to create the persistent volume immediately.
- Creation of more than 100 volumes per 3 nodes per cluster is not supported.
- If there are multiple Red Hat Gluster Storage trusted storage pools in your environment, you can check on which trusted storage pool the volume is created using the **`heketi-cli volume list`** command. This command lists the cluster name. You can then update the endpoint information in the **`pv001.json`** file accordingly.
- When creating a Heketi volume with only two nodes with the replica count set to the default value of three (replica 3), an error "No space" is displayed by Heketi as there is no space to create a replica set of three disks on three different nodes.
- If all the heketi-cli write operations (ex: volume create, cluster create..etc) fails and the read operations (ex: topology info, volume info ..etc) are successful, then the possibility is that the gluster volume is operating in read-only mode.

7. Edit the pv001.json file and enter the name of the endpoint in the endpoints section.

```
# oc create -f pv001.json
```

For example:

```
# oc create -f pv001.json
persistentvolume "glusterfs-4fc22ff9" created
```

8. To verify that the persistent volume is created, execute the following command:

```
# oc get pv
```

For example:

```
# oc get pv

NAME                CAPACITY   ACCESSMODES   STATUS      CLAIM
```

```
REASON    AGE
glusterfs-4fc22ff9   100Gi      RWX           Available
4s
```

9. Bind the persistent volume to the persistent volume claim by executing the following command:

```
# oc create -f pvc.json
```

For example:

```
# oc create -f pvc.json
persistentvolumeclaim"glusterfs-claim" created
```

```
# cat pvc.json

{
   "apiVersion": "v1",
   "kind": "PersistentVolumeClaim",
   "metadata": {
     "name":  "glusterfs-claim"
  },
   "spec": {
     "accessModes": [
       "ReadWriteMany"
    ],
     "resources": {
       "requests": {
         "storage ": "1Gi"
      },
       "selector": {
       "matchLabels": {
         "storage-tier": "gold"
       }
     }
    }
   }
}
```

10. To verify that the persistent volume and the persistent volume claim is bound, execute the following commands:

```
# oc get pv
# oc get pvc
```

For example:

```
# oc get pv

NAME                 CAPACITY   ACCESSMODES   STATUS    CLAIM
REASON    AGE
glusterfs-4fc22ff9   100Gi      RWX           Bound     storage-
project/glusterfs-claim            1m
```

```
# oc get pvc

NAME              STATUS    VOLUME               CAPACITY
ACCESSMODES   AGE
glusterfs-claim   Bound     glusterfs-4fc22ff9   100Gi      RWX
11s
```

11. The claim can now be used in the application:

    For example:

```
# cat app.yml

apiVersion: v1
kind: Pod
metadata:
  name: busybox
spec:
  containers:
    - image: busybox
      command:
        - sleep
        - "3600"
      name: busybox
      volumeMounts:
        - mountPath: /usr/share/busybox
          name: mypvc
  volumes:
    - name: mypvc
      persistentVolumeClaim:
        claimName: glusterfs-claim
```

```
# oc create -f app.yml
pod "busybox" created
```

    For more information about using the glusterfs claim in the application see, https://access.redhat.com/documentation/en/openshift-enterprise/version-3.2/installation-and-configuration/#complete-example-using-gusterfs-creating-the-pod.

12. To verify that the pod is created, execute the following command:

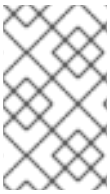
```
# oc get pods
```

13. To verify that the persistent volume is mounted inside the container, execute the following command:

```
# oc rsh busybox
```

```
/ $ df -h
Filesystem                Size      Used Available Use% Mounted on
/dev/mapper/docker-253:0-1310998-
81732b5fd87c197f627a24bcd2777f12eec4ee937cc2660656908b2fa6359129
                        100.0G     34.1M     99.9G   0% /
tmpfs                     1.5G         0      1.5G   0% /dev
```

```
tmpfs                     1.5G         0      1.5G   0%
/sys/fs/cgroup
192.168.121.168:vol_4fc22ff934e531dec3830cfbcad1eeae
                         99.9G     66.1M     99.9G   0%
/usr/share/busybox
tmpfs                     1.5G         0      1.5G   0%
/run/secrets
/dev/mapper/vg_vagrant-lv_root
                         37.7G      3.8G     32.0G  11%
/dev/termination-log
tmpfs                     1.5G     12.0K      1.5G   0%
/var/run/secrets/kubernetes.io/serviceaccount
```



NOTE

If you encounter a permission denied error on the mount point, then refer to section Gluster Volume Security at: https://access.redhat.com/documentation/en/openshift-enterprise/3.2/single/installation-and-configuration/#gluster-volume-security.

CHAPTER 6. OPERATIONS ON A RED HAT GLUSTER STORAGE POD IN AN OPENSHIFT ENVIRONMENT

This chapter lists out the various operations that can be performed on a Red Hat Gluster Storage pod (gluster pod):

1. To list the pods, execute the following command :

```
# oc get pods
```

For example:

```
# oc get pods
NAME                                                    READY
STATUS    RESTARTS   AGE
storage-project-router-1-v89qc                          1/1
Running   0          1d
glusterfs-dc-node1.example.com                          1/1
Running   0          1d
glusterfs-dc-node2.example.com                          1/1
Running   1          1d
glusterfs-dc-node3.example.com                          1/1
Running   0          1d
heketi-1-k1u14                                          1/1
Running   0          23m
rhel1                                                   1/1
Running   0          26s
```

Following are the gluster pods from the above example:

```
glusterfs-dc-node1.example.com
glusterfs-dc-node2.example.com
glusterfs-dc-node3.example.com
```

NOTE

The topology.json file will provide the details of the nodes in a given Trusted Storage Pool (TSP) . In the above example all the 3 Red Hat Gluster Storage nodes are from the same TSP.

2. To enter the gluster pod shell, execute the following command:

```
# oc rsh <gluster_pod_name>
```

For example:

```
# oc rsh glusterfs-dc-node1.example.com

sh-4.2#
```

3. To get the peer status, execute the following command:

```
# gluster peer status
```

For example:

```
# gluster peer status

Number of Peers: 2

Hostname: node2.example.com
Uuid: 9f3f84d2-ef8e-4d6e-aa2c-5e0370a99620
State: Peer in Cluster (Connected)
Other names:
node1.example.com

Hostname: node3.example.com
Uuid: 38621acd-eb76-4bd8-8162-9c2374affbbd
State: Peer in Cluster (Connected)
```

4. To list the gluster volumes on the Trusted Storage Pool, execute the following command:

```
# gluster volume info
```

For example:

```
Volume Name: heketidbstorage
Type: Distributed-Replicate
Volume ID: 2fa53b28-121d-4842-9d2f-dce1b0458fda
Status: Started
Number of Bricks: 2 x 3 = 6
Transport-type: tcp
Bricks:
Brick1:
192.168.121.172:/var/lib/heketi/mounts/vg_1be433737b71419dc9b395e221
255fb3/brick_c67fb97f74649d990c5743090e0c9176/brick
Brick2:
192.168.121.233:/var/lib/heketi/mounts/vg_0013ee200cdefaeb6dfedd28e5
0fd261/brick_6ebf1ee62a8e9e7a0f88e4551d4b2386/brick
Brick3:
192.168.121.168:/var/lib/heketi/mounts/vg_e4b32535c55c88f9190da7b7ef
d1fcab/brick_df5db97aa002d572a0fec6bcf2101aad/brick
Brick4:
192.168.121.233:/var/lib/heketi/mounts/vg_0013ee200cdefaeb6dfedd28e5
0fd261/brick_acc82e56236df912e9a1948f594415a7/brick
Brick5:
192.168.121.168:/var/lib/heketi/mounts/vg_e4b32535c55c88f9190da7b7ef
d1fcab/brick_65dceb1f749ec417533ddeae9535e8be/brick
Brick6:
192.168.121.172:/var/lib/heketi/mounts/vg_7ad961dbd24e16d62cabe10fd8
bf8909/brick_f258450fc6f025f99952a6edea203859/brick
Options Reconfigured:
performance.readdir-ahead: on

Volume Name: vol_9e86c0493f6b1be648c9deee1dc226a6
Type: Distributed-Replicate
Volume ID: 940177c3-d866-4e5e-9aa0-fc9be94fc0f4
```

```
Status: Started
Number of Bricks: 2 x 3 = 6
Transport-type: tcp
Bricks:
Brick1:
192.168.121.168:/var/lib/heketi/mounts/vg_3fa141bf2d09d30b899f2f260c
494376/brick_9fb4a5206bdd8ac70170d00f304f99a5/brick
Brick2:
192.168.121.172:/var/lib/heketi/mounts/vg_7ad961dbd24e16d62cabe10fd8
bf8909/brick_dae2422d518915241f74fd90b426a379/brick
Brick3:
192.168.121.233:/var/lib/heketi/mounts/vg_5c6428c439eb6686c5e4cee565
32bacf/brick_b3768ba8e80863724c9ec42446ea4812/brick
Brick4:
192.168.121.172:/var/lib/heketi/mounts/vg_7ad961dbd24e16d62cabe10fd8
bf8909/brick_0a13958525c6343c4a7951acec199da0/brick
Brick5:
192.168.121.168:/var/lib/heketi/mounts/vg_17fbc98d84df86756e7826326f
b33aa4/brick_af42af87ad87ab4f01e8ca153abbbee9/brick
Brick6:
192.168.121.233:/var/lib/heketi/mounts/vg_5c6428c439eb6686c5e4cee565
32bacf/brick_ef41e04ca648efaf04178e64d25dbdcb/brick
Options Reconfigured:
performance.readdir-ahead: on
```

5. To get the volume status, execute the following command:

```
# gluster volume status <volname>
```

For example:

```
# gluster volume status vol_9e86c0493f6b1be648c9deee1dc226a6

Status of volume: vol_9e86c0493f6b1be648c9deee1dc226a6
Gluster process                                 TCP Port  RDMA Port
Online  Pid
------------------------------------------------------------------
--------------
Brick 192.168.121.168:/var/lib/heketi/mounts/v
g_3fa141bf2d09d30b899f2f260c494376/brick_9f
b4a5206bdd8ac70170d00f304f99a5/brick        49154     0          Y
3462
Brick 192.168.121.172:/var/lib/heketi/mounts/v
g_7ad961dbd24e16d62cabe10fd8bf8909/brick_da
e2422d518915241f74fd90b426a379/brick        49154     0          Y
115939
Brick 192.168.121.233:/var/lib/heketi/mounts/v
g_5c6428c439eb6686c5e4cee56532bacf/brick_b3
768ba8e80863724c9ec42446ea4812/brick        49154     0          Y
116134
Brick 192.168.121.172:/var/lib/heketi/mounts/v
g_7ad961dbd24e16d62cabe10fd8bf8909/brick_0a
13958525c6343c4a7951acec199da0/brick        49155     0          Y
115958
Brick 192.168.121.168:/var/lib/heketi/mounts/v
```

```
g_17fbc98d84df86756e7826326fb33aa4/brick_af
42af87ad87ab4f01e8ca153abbbee9/brick        49155     0          Y
3481
Brick 192.168.121.233:/var/lib/heketi/mounts/v
g_5c6428c439eb6686c5e4cee56532bacf/brick_ef
41e04ca648efaf04178e64d25dbdcb/brick        49155     0          Y
116153
NFS Server on localhost                     2049      0          Y
116173
Self-heal Daemon on localhost               N/A       N/A        Y
116181
NFS Server on node1.example.com
2049      0          Y       3501
Self-heal Daemon on node1.example.com
N/A       N/A        Y       3509
NFS Server on 192.168.121.172                   2049      0
Y       115978
Self-heal Daemon on 192.168.121.172             N/A       N/A
Y       115986

Task Status of Volume vol_9e86c0493f6b1be648c9deee1dc226a6
------------------------------------------------------------------
--------------
There are no active volume tasks
```

6. To take the snapshot of the gluster volume, execute the following command:

```
# gluster snapshot create <snapname> <volname>
```

For example:

```
# gluster snapshot create snap1 vol_9e86c0493f6b1be648c9deee1dc226a6

snapshot create: success: Snap snap1_GMT-2016.07.29-13.05.46 created
successfully
```

7. To list the snapshots, execute the following command:

```
# gluster snapshot list
```

For example:

```
# gluster snapshot list

snap1_GMT-2016.07.29-13.05.46
snap2_GMT-2016.07.29-13.06.13
snap3_GMT-2016.07.29-13.06.18
snap4_GMT-2016.07.29-13.06.22
snap5_GMT-2016.07.29-13.06.26
```

8. To delete a snapshot, execute the following command:

```
# gluster snap delete <snapname>
```

-

For example:

```
# gluster snap delete snap1_GMT-2016.07.29-13.05.46

Deleting snap will erase all the information about the snap. Do you
still want to continue? (y/n) y
snapshot delete: snap1_GMT-2016.07.29-13.05.46: snap removed
successfully
```

For more information about managing snapshots, refer https://access.redhat.com/documentation/en-US/Red_Hat_Storage/3.1/html-single/Administration_Guide/index.html#chap-Managing_Snapshots.

CHAPTER 7. MANAGING CLUSTERS

Heketi allows administrators to add and remove storage capacity by managing either a single or multiple Red Hat Gluster Storage clusters.

7.1. INCREASING STORAGE CAPACITY

You can increase the storage capacity using any of the following ways:

- Adding devices
- Increasing cluster size
- Adding an entirely new cluster.

7.1.1. Adding New Devices

You can add more devices to existing nodes to increase storage capacity. When adding more devices, you must ensure to add devices as a set. For example, when expanding a distributed replicated volume with a replica count of replica 2, then one device should be added to at least two nodes. If using replica 3, then at least one device should be added to at least three nodes.

You can add a device either using CLI, or the API, or by updating the topology JSON file. The sections ahead describe using heketi CLI and updating topology JSON file. For information on adding new devices using API, see Heketi API https://github.com/heketi/heketi/wiki/API#device_add

7.1.1.1. Using Heketi CLI

Register the specified device. The following example command shows how to add a device **/dev/sde** to node **d6f2c22f2757bf67b1486d868dcb7794**:

```
# heketi-cli device add --name=/dev/sde --
node=d6f2c22f2757bf67b1486d868dcb7794
OUTPUT:
Device added successfully
```

7.1.1.2. Updating Topology File

You can add the new device to the node description in your topology JSON used to setup the cluster. Then rerun the command to load the topology.

Following is an example where a new **/dev/sde** drive added to the node:

In the file:

```
    {
                "node": {
                    "hostnames": {
                        "manage": [
                            "node4.example.com"
                        ],
                        "storage": [
                            "192.168.10.100"
                        ]
```

```
                    },
                    "zone": 1
                },
                "devices": [
                    "/dev/sdb",
                    "/dev/sdc",
                    "/dev/sdd",
                    "/dev/sde"
                ]
            }
```

Load the topology file:

```
# heketi-cli topology load --json=topology-sample.json
    Found node 192.168.10.100 on cluster d6f2c22f2757bf67b1486d868dcb7794
        Found device /dev/sdb
        Found device /dev/sdc
        Found device /dev/sdd
        Adding device /dev/sde ... OK
    Found node 192.168.10.101 on cluster d6f2c22f2757bf67b1486d868dcb7794
        Found device /dev/sdb
        Found device /dev/sdc
        Found device /dev/sdd
    Found node 192.168.10.102 on cluster d6f2c22f2757bf67b1486d868dcb7794
        Found device /dev/sdb
        Found device /dev/sdc
        Found device /dev/sdd
    Found node 192.168.10.103 on cluster d6f2c22f2757bf67b1486d868dcb7794
        Found device /dev/sdb
        Found device /dev/sdc
        Found device /dev/sdd
```

7.1.2. Increasing Cluster Size

Another way to add storage to Heketi, is to add new nodes to the cluster. Like adding devices, you can add a new node to an existing cluster by either using CLI or the API or by updating the topology JSON file. When you add a new node to the cluster, then you must register new devices to that node.

The sections ahead describe using heketi CLI and updating topology JSON file. For information on adding new devices using API, see Heketi API: https://github.com/heketi/heketi/wiki/API#node_add

7.1.2.1. Using Heketi CLI

Following shows an example of how to add new node in **zone 1** to **597fceb5d6c876b899e48f599b988f54** cluster using the CLI:

```
# heketi-cli node add --zone=1 --cluster=597fceb5d6c876b899e48f599b988f54
--management-host-name=node4.example.com --storage-host-
name=192.168.10.104

OUTPUT:
Node information:
Id: 095d5f26b56dc6c64564a9bc17338cbf
State: online
Cluster Id: 597fceb5d6c876b899e48f599b988f54
```

```
Zone: 1
Management Hostname node4.example.com
Storage Hostname 192.168.10.104
```

The following example command shows how to register **/dev/sdb** and **/dev/sdc** devices for **095d5f26b56dc6c64564a9bc17338cbf** node:

```
# heketi-cli device add --name=/dev/sdb --
node=095d5f26b56dc6c64564a9bc17338cbf
OUTPUT:
Device added successfully

# heketi-cli device add --name=/dev/sdc --
node=095d5f26b56dc6c64564a9bc17338cbf
OUTPUT:
Device added successfully
```

7.1.2.2. Updating Topology File

You can expand a cluster by adding a new node to your topology JSON file. When adding the new node you must add this node information **after** the existing ones so that the Heketi CLI identifies on which cluster this new node should be part of.

Following shows an example of how to add a new node and devices:

```
                {
                    "node": {
                        "hostnames": {
                            "manage": [
                                "node4.example.com"
                            ],
                            "storage": [
                                "192.168.10.104"
                            ]
                        },
                        "zone": 1
                    },
                    "devices": [
                        "/dev/sdb",
                        "/dev/sdc"
                     ]
                }
```

Load the topology file:

```
# heketi-cli topology load --json=topology-sample.json
    Found node 192.168.10.100 on cluster d6f2c22f2757bf67b1486d868dcb7794
        Found device /dev/sdb
        Found device /dev/sdc
        Found device /dev/sdd
        Found device /dev/sde
    Found node 192.168.10.101 on cluster d6f2c22f2757bf67b1486d868dcb7794
        Found device /dev/sdb
        Found device /dev/sdc
```

```
        Found device /dev/sdd
    Found node 192.168.10.102 on cluster d6f2c22f2757bf67b1486d868dcb7794
        Found device /dev/sdb
        Found device /dev/sdc
        Found device /dev/sdd
    Found node 192.168.10.103 on cluster d6f2c22f2757bf67b1486d868dcb7794
        Found device /dev/sdb
        Found device /dev/sdc
        Found device /dev/sdd
    Creating node node4.example.com ... ID:
ff3375aca6d98ed8a004787ab823e293
        Adding device /dev/sdb ... OK
        Adding device /dev/sdc ... OK
```

7.1.3. Adding a New Cluster

Storage capacity can also be increased by adding new clusters of GlusterFS. Just as before, there are three ways to add a new cluster to Heketi. One way is to use the API (https://github.com/heketi/heketi/wiki/API#clusters), another is to use *heketi-cli*, but the easiest way is to create another topology JSON file which defines the new nodes and devices which will compose the cluster.

7.1.3.1. Updating Topology file

You can add a new cluster to your topology JSON file which defines the new nodes and devices which will compose the cluster.

Following is an example showing how to add a new node and devices:

```
                {
                    "node": {
                        "hostnames": {
                            "manage": [
                                "node4.example.com"
                            ],
                            "storage": [
                                "192.168.10.104"
                            ]
                        },
                        "zone": 1
                    },
                    "devices": [
                        "/dev/sdb",
                        "/dev/sdc",
                        "/dev/sdd",
                     ]
                }
```

Load the topology file:

```
# heketi-cli topology load --json=topology-sample.json
    Found node 192.168.10.100 on cluster d6f2c22f2757bf67b1486d868dcb7794
        Found device /dev/sdb
        Found device /dev/sdc
        Found device /dev/sdd
```

```
        Found device /dev/sde
    Found node 192.168.10.101 on cluster d6f2c22f2757bf67b1486d868dcb7794
        Found device /dev/sdb
        Found device /dev/sdc
        Found device /dev/sdd
    Found node 192.168.10.102 on cluster d6f2c22f2757bf67b1486d868dcb7794
        Found device /dev/sdb
        Found device /dev/sdc
        Found device /dev/sdd
    Found node 192.168.10.103 on cluster d6f2c22f2757bf67b1486d868dcb7794
        Found device /dev/sdb
        Found device /dev/sdc
        Found device /dev/sdd
    Found node 192.168.10.104 on cluster d6f2c22f2757bf67b1486d868dcb7794
        Found device /dev/sdb
        Found device /dev/sdc
        Found device /dev/sdd
```

7.2. REDUCING STORAGE CAPACITY

Heketi also supports the reduction of storage capacity. You can reduce storage by deleting devices, nodes, and clusters. These requests can only be performed by using the Heketi CLI or from the command line API. For information on using command link API, see Heketi API https://github.com/heketi/heketi/wiki/API.

7.2.1. Deleting Devices

You can update the topology file by deleting the devices listed in the topolgy file.

For example, in the topology file, **/dev/sde** drive is deleted from the node:

In the file:

```
{
    "node": {
        "hostnames": {
            "manage": [
                "node4.example.com"
            ],
            "storage": [
                "192.168.10.100"
            ]
        },
        "zone": 1
    },
    "devices": [
        "/dev/sdb",
        "/dev/sdc",
        "/dev/sdd",
     ]
}
```

Load the topology file:

```
# heketi-cli topology load --json=topology-sample.json
    Found node 192.168.10.100 on cluster d6f2c22f2757bf67b1486d868dcb7794
        Found device /dev/sdb
        Found device /dev/sdc
        Found device /dev/sdd
    Found node 192.168.10.101 on cluster d6f2c22f2757bf67b1486d868dcb7794
        Found device /dev/sdb
        Found device /dev/sdc
        Found device /dev/sdd
    Found node 192.168.10.102 on cluster d6f2c22f2757bf67b1486d868dcb7794
        Found device /dev/sdb
        Found device /dev/sdc
        Found device /dev/sdd
    Found node 192.168.10.103 on cluster d6f2c22f2757bf67b1486d868dcb7794
        Found device /dev/sdb
        Found device /dev/sdc
        Found device /dev/sdd
```

7.2.2. Deleting Nodes

You can update the topology file by deleting the node listed in the file.

For example, in the topology file, delete the node and reload the topology file:

```
# heketi-cli topology load --json=topology-sample.json
    Found node 192.168.10.100 on cluster d6f2c22f2757bf67b1486d868dcb7794
        Found device /dev/sdb
        Found device /dev/sdc
        Found device /dev/sdd
    Found node 192.168.10.101 on cluster d6f2c22f2757bf67b1486d868dcb7794
        Found device /dev/sdb
        Found device /dev/sdc
        Found device /dev/sdd
    Found node 192.168.10.102 on cluster d6f2c22f2757bf67b1486d868dcb7794
        Found device /dev/sdb
        Found device /dev/sdc
        Found device /dev/sdd
```

In this example, node 192.168.10.103 on cluster d6f2c22f2757bf67b1486d868dcb7794 is deleted.

7.2.3. Deleting Clusters

You can update the topology file by deleting the cluster and reload the topology file.

CHAPTER 8. UNINSTALLING CONTAINERIZED RED HAT GLUSTER STORAGE

This chapter outlines the details for uninstalling containerized Red Hat Gluster Storage.

Perform the following steps for uninstalling:

1. **Cleanup Red Hat Gluster Storage using Heketi**

    1. Remove any containers using the persistent volume claim from Red Hat Gluster Storage.

    2. Remove the appropriate persistent volume claim and persistent volume:

    ```
    # oc delete pvc <pvc_name>
    # oc delete pv <pv_name>
    ```

2. **Remove all OpenShift objects**

    1. Delete all project specific pods, services, routes, and deployment configurations:

    ```
    # oc delete deploymentconfig glusterfs-dc-<IP-ADDR/Hostname>

    # oc delete deploymentconfig heketi

    # oc delete service heketi heketi-storage-endpoints

    # oc delete route heketi

    # oc delete endpoints heketi-storage-endpoints
    ```

    Wait until all the pods have been terminated.

    2. Check and delete the gluster service and endpoints from the projects that required a persistent storage:

    ```
    # oc get endpoints,service

    # oc delete endpoints <glusterfs-endpoint-name>

    # oc delete service <glusterfs-service-name>
    ```

3. **Cleanup the persistent directories**

    1. To cleanup the persistent directories execute the following command on each node as a root user:

    ```
    # rm -rf /var/lib/heketi \
       /etc/glusterfs \
       /var/lib/glusterd \
       /var/log/glusterfs
    ```

4. **Force cleanup the disks**

    1. Execute the following command to cleanup the disks:

```
# wipefs -a -f /dev/<disk-id>
```

APPENDIX A. CLUSTER ADMINISTRATOR SETUP

Authentication

Set up the authentication using **AllowAll Authentication** method.

AllowAll Authentication

Set up an authentication model which allows all passwords. Edit **`/etc/origin/master/master-config.yaml`** on the OpenShift master and change the value of **`DenyAllPasswordIdentityProvider`** to **`AllowAllPasswordIdentityProvider`**. Then restart the OpenShift master.

1. Now that the authentication model has been setup, login as a user, for example admin/admin:

```
# oc login openshift master e.g. https://1.1.1.1:8443  --
username=admin --password=admin
```

2. Grant the admin user account the **`cluster-admin`** role.

```
  # oadm policy add-cluster-role-to-user cluster-admin admin
```

For more information on authentication methods, see https://access.redhat.com/documentation/en/openshift-enterprise/3.2/single/installation-and-configuration/#identity-providers.

APPENDIX B. CLIENT CONFIGURATION USING PORT FORWARDING

If a router is not available, you may be able to set up port forwarding so that heketi-cli can communicate with the Heketi service. Execute the following commands for port forwarding:

1. Obtain the Heketi service pod name by running the following command:

```
# oc get pods
```

2. To forward the port on your local system to the pod, execute the following command on another terminal of your local system:

```
# oc port-forward <heketi pod name> 8080:8080
```

3. On the original terminal execute the following command to test the communication with the server:

```
# curl http://localhost:8080/hello
```

   This will forward the local port 8080 to the pod port 8080.

4. Setup the Heketi server environment variable by running the following command:

```
# export HEKETI_CLI_SERVER=http://localhost:8080
```

5. Get information from Heketi by running the following command:

```
# heketi-cli topology info
```

APPENDIX C. HEKETI CLI COMMANDS

This section provides a list of some of the useful heketi-cli commands:

- **heketi-cli topology info**

  This command retreives information about the current Topology.

- **heketi-cli cluster list**

  Lists the clusters managed by Heketi

  For example:

```
# heketi-cli cluster list
Clusters:
9460bbea6f6b1e4d833ae803816122c6
```

- **heketi-cli cluster info <cluster_id>**

  Retrieves the information about the cluster.

  For example:

```
# heketi-cli cluster info 9460bbea6f6b1e4d833ae803816122c6
Cluster id: 9460bbea6f6b1e4d833ae803816122c6
Nodes:
1030f9361cff8c6bfde7b9b079327c78
30f2ab4d971da572b03cfe33a1ba525f
f648e1ddc0b95f3069bd2e14c7e34475
Volumes:
142e0ec4a4c1d1cc082071329a0911c6
638d0dc6b1c85f5eaf13bd5c7ed2ee2a
```

- **heketi-cli node info <node_id>**

  Retrieves the information about the node.

  For example:

```
# heketi-cli node info 1030f9361cff8c6bfde7b9b079327c78
Node Id: 1030f9361cff8c6bfde7b9b079327c78
State: online
Cluster Id: 9460bbea6f6b1e4d833ae803816122c6
Zone: 1
Management Hostname: node1.example.com
Storage Hostname: 10.70.41.202
Devices:
Id:69214867a4d32251aaf1dcd77cb7f359   Name:/dev/vdg
State:online    Size (GiB):4999    Used (GiB):253     Free
(GiB):4746
Id:6cd437c304979ea004abc2c4da8bdaf4   Name:/dev/vde
State:online    Size (GiB):4999    Used (GiB):354     Free
(GiB):4645
Id:d2e9fcd9da04999ddab11cab651e18d2   Name:/dev/vdf
State:online    Size (GiB):4999    Used (GiB):831     Free
(GiB):4168
```

–

- **heketi-cli volume list**

  Lists the volumes managed by Heketi

  For example:

```
# heketi-cli volume list
Id:142e0ec4a4c1d1cc082071329a0911c6
Cluster:9460bbea6f6b1e4d833ae803816122c6    Name:heketidbstorage
Id:638d0dc6b1c85f5eaf13bd5c7ed2ee2a
Cluster:9460bbea6f6b1e4d833ae803816122c6    Name:scalevol-1
```

For more information, refer to the man page of the heketi-cli.

```
# heketi-cli --help
```

The command line program for Heketi.

Usage

- heketi-cli [flags]
- heketi-cli [command]

For example:

```
# export HEKETI_CLI_SERVER=http://localhost:8080
```

```
# heketi-cli volume list
```

The available commands are listed below:

- **cluster**

  Heketi cluster management

- **device**

  Heketi device management

- **setup-openshift-heketi-storage**

  Setup OpenShift/Kubernetes persistent storage for Heketi

- **node**

  Heketi Node Management

- **topology**

  Heketi Topology Management

- **volume**

  Heketi Volume Management

APPENDIX D. CLEANING UP THE HEKETI TOPOLOGY

1. Delete all the volumes by executing the following command:

```
heketi-cli volume delete <volume_id>
```

2. Delete all the devices by executing the following command:

```
# heketi-cli device delete <device_id>
```

3. Delete all the nodes by executing the following command:

```
 # heketi-cli node delete <node_id>
```

4. Delete all the clusters by executing the following command:

```
# heketi-cli cluster delete <cluster_id>
```

NOTE

- The IDs can be retrieved by executing the heketi-cli topology info command.
- The **heketidbstorage** volume cannot be deleted as it contains the heketi database.

APPENDIX E. KNOWN ISSUES

This chapter outlines a known issue at the time of release.

BZ#1356058

glusterd service is responsible for allocating the ports for the brick processes of the volumes. Currently for the newly created volumes, **glusterd** service does not reutilize the ports which were used for brick processes for the volumes which are now been deleted. Hence, having a number of active volumes versus number of open ports in the system ratio does not work.

Workaround: After all the open ports been consumed by the brick process, even in case of the older volumes been deleted/stopped, you must open the fresh ports for new volumes.