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

Deploying Container-Native Storage for OpenShift Container Platform
Edition 1

Divya Muntimadugu Bhavana Mohan

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

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-container-platform/3.4/paged/installation-and-configuration/chapter-21-configuring-persistent-storage#install-config-persistent-storage-persistent-storage-glusterfs

- Chapter 2, *Container-Native Storage for 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.

Chapter 2. Container-Native Storage for 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.

Container-Native Storage for 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.

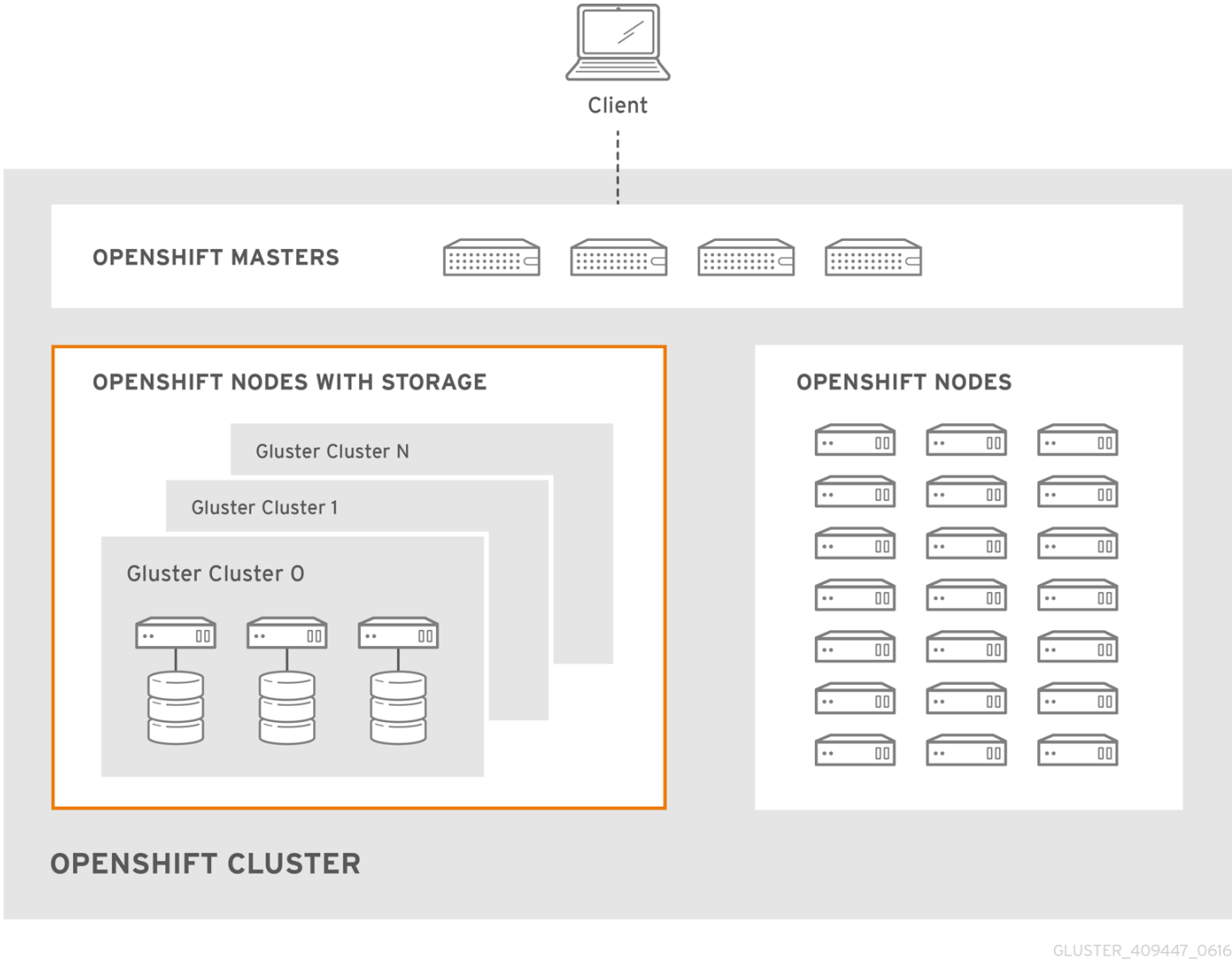


Figure 2.1. Architecture - Container-Native Storage for OpenShift Container Platform

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	3.4

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 Container Platform 3.4.

3.2.1.1. Setting up the Openshift Master as the Client

You can 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. You 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-container-platform/3.4/single/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 cns-deploy heketi-client
```

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 cns-deploy heketi-client
```

Subscribe to the OpenShift Container Platform 3.4 repository

If you are using OpenShift Container Platform 3.4, subscribe to 3.4 repository to enable you to install the Openshift client packages

```
# subscription-manager repos --enable=rhel-7-server-ose-3.4-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. For information on setting up OpenShift cluster, see https://access.redhat.com/documentation/en/openshift-container-platform/3.4/paged/installation-and-configuration.
- 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
```

- Execute the following command on each node to verify if the iptables are updated:

```
# iptables -L
```

- A cluster-admin user must be created. For more information, see Appendix B, *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.
- It is recommended to persist the logs for the Heketi container. For more information on persisting logs, refer https://access.redhat.com/documentation/en/openshift-container-platform/3.4/single/installation-and-configuration/#install-config-aggregate-logging.

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 2 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 master or 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. On the master or client, 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. After the project is created, execute the following command on the master node to enable the deployment of the privileged containers as Red Hat Gluster Storage container can only run in the privileged mode.

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

4. Execute the following steps on the master to set up the router:

Note

If a router already exists, proceed to Step 5.

a. Execute the following command to enable the deployment of the router:

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

b. Execute the following command to deploy the router:

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

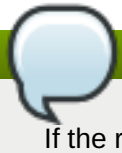
c. Edit the subdomain name in the config.yaml file located at /etc/origin/master/master-config.yaml .

For example:

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

d. 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 C, *Client Configuration using Port Forwarding* .

For more information regarding router setup, see https://access.redhat.com/documentation/en/openshift-container-platform/3.4/paged/installation-and-configuration/chapter-4-setting-up-a-router

5. Execute the following command to verify if the router is running:

```
# oc get dc <router_name>
```

For example:

```
# oc get dc storage-project-router
NAME                    REVISION   DESIRED   CURRENT   TRIGGERED BY
storage-project-router   1          1         1         config
```

Note

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

6. After the router is running, the client has to be setup to access the services in the OpenShift cluster.

Execute the following steps on the client to set up the DNS.

a. 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 the node where the router is running.

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

```
# systemctl restart dnsmasq
```

c. 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-container-platform/3.4/single/installation-and-configuration/#environment-requirements.

4.2. Deploying Container-Native Storage

The following section covers deployment of the Container-Native Storage pods using the **cns-deploy** tool. If you prefer to manually install Container-Native Storage, see Appendix A, *Manual Deployment*

1. You must first provide a topology file for heketi which describes the topology of the Red Hat Gluster Storage nodes and their attached storage devices. A sample, formatted topology file (topology-sample.json) is installed with the 'heketi-client' package in the /usr/share/heketi/ directory. .

```
{
    "clusters": [
                {
                   "nodes": [
                                {
                         "node": {
                             "hostnames": {
                                 "manage": [
                                     "node1.example.com"
                                 ],
                                 "storage": [
                                     "192.168.121.168"
                                  ]
                              },
                              "zone": 1
                          },
                          "devices": [
                              "/dev/sdb",
                              "/dev/sdc",
                              "/dev/sdd",
                              "/dev/sde"
                          ]
                },...
```

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 /usr/share/heketi/topology-sample.json file only sets up 4 nodes with 8 drives each.

Important

Heketi stores its database on a Red Hat Gluster Storage volume. In cases where the volume is down, the Heketi service does not respond due to the unavailability of the volume served by a disabled trusted storage pool. To resolve this issue, restart the trusted storage pool which contains the Heketi volume.

2. Execute the following command on the client to deploy the heketi and Red Hat Gluster Storage pods:

```
# cns-deploy -n <namespace> -g topology.json
```

For example:

```
# cns-deploy -n storage-project -g topology.json
Multiple CLI options detected. Please select a deployment option.
[O]penShift, [K]ubernetes? [O/o/K/k]: o
Using OpenShift CLI.
template "deploy-heketi" created
serviceaccount "heketi-service-account" created
template "heketi" created
template "glusterfs" created
node "192.168.121.168" labeled
node "192.168.121.169" labeled
node "192.168.121.170" labeled
daemonset "glusterfs" created
Waiting for GlusterFS pods to start ... OK
service "deploy-heketi" created
route "deploy-heketi" created
deploymentconfig "deploy-heketi" created
Waiting for deploy-heketi pod to start ... OK
  % Total    % Received % Xferd  Average Speed   Time    Time
Time  Current
                                 Dload  Upload   Total   Spent
Left  Speed
100    17  100    17    0     0   2283      0 --:--:-- --:--:-- --:--
:--  2428
Creating cluster ... ID: e9c3135bda886b0770f4fc7131f46061
    Creating node 192.168.121.168 ... ID:
994390ff41b0fc2116962454a860c356
        Adding device /dev/sdb ... OK
        Adding device /dev/sdc ... OK
        Adding device /dev/sdd ... OK
        Adding device /dev/sde ... OK
    Creating node 192.168.121.169 ... ID:
217e1740010b264624043011225ffc14
        Adding device /dev/sdb ... OK
        Adding device /dev/sdc ... OK
        Adding device /dev/sdd ... OK
        Adding device /dev/sde ... OK
```

```
    Creating node 192.168.121.170 ... ID:
f5ae3ecb0475729fbb70eba5842e52c0
        Adding device /dev/sdb ... OK
        Adding device /dev/sdc ... OK
        Adding device /dev/sdd ... OK
        Adding device /dev/sde ... OK
Saving heketi-storage.json
secret "heketi-storage-secret" created
endpoints "heketi-storage-endpoints" created
service "heketi-storage-endpoints" created
job "heketi-storage-copy-job" created
deploymentconfig "deploy-heketi" deleted
route "deploy-heketi" deleted
service "deploy-heketi" deleted
job "heketi-storage-copy-job" deleted
secret "heketi-storage-secret" deleted
service "heketi" created
route "heketi" created
deploymentconfig "heketi" created
Waiting for heketi pod to start ... OK
  % Total    % Received % Xferd  Average Speed   Time    Time
Time  Current
                                 Dload  Upload   Total   Spent
Left  Speed
100    17  100    17    0     0   2259      0 --:--:-- --:--:-- --:--
:--  2428
heketi is now running.
```

Note

For more information on the cns-deploy commands, refer to the man page of the cns-deploy.

```
# cns-deploy --help
```

3. 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
```

To verify if Heketi is loaded with the topology execute the following command:

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

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.

You can provision volumes either statically or dynamically. In static provisioning of volumes a persistent volume claim has to be created which the administrator uses to create a persistent volume. More details about static provisioning of volumes is provided in Section 5.1, "Static Provisioning of Volumes".

From Container Native Storage 3.4 release onwards dynamic provisioning of volumes is introduced. With dynamic provisioning no administrator intervention is required to create a persistent volume. The volume will be created dynamically and provisioned to the application containers. More details about dynamic provisioning of volumes is provided in Section 5.2, "Dynamic Provisioning of Volumes".

5.1. Static Provisioning of Volumes

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

The sample glusterfs endpoint file (sample-gluster-endpoints.yaml) and the sample glusterfs service file (sample-gluster-service.yaml) are available at **`/usr/share/heketi/templates/`** directory.

1. To specify the endpoints you want to create, update the **`sample-gluster-endpoints.yaml`** 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.

```
# cat sample-gluster-endpoints.yaml
apiVersion: v1
kind: Endpoints
metadata:
  name: glusterfs-cluster
subsets:
- addresses:
  - ip: 192.168.10.100
  ports:
  - port: 1
- addresses:
  - ip: 192.168.10.101
  ports:
  - port: 1
```

```
- addresses:
  - ip: 192.168.10.102
  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-endpoints.yaml
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.yaml
service "glusterfs-cluster" created
```

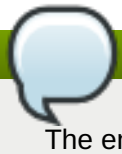
```
# cat sample-gluster-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: glusterfs-cluster
spec:
  ports:
  - port: 1
```

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
  },
  "spec": {
    "capacity": {
      "storage": "100Gi"
    },
    "glusterfs": {
      "endpoints": "TYPE ENDPOINT HERE",
      "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-storage-project-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 endpoint's section:

```
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": {}
}
```

8. Create a persistent volume by executing the following command:

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

For example:

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

9. 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
```

10. Create a persistent volume claim file. For example:

```
# cat pvc.json

{
   "apiVersion": "v1",
   "kind": "PersistentVolumeClaim",
   "metadata": {
     "name":  "glusterfs-claim"
  },
   "spec": {
     "accessModes": [
       "ReadWriteMany"
    ],
     "resources": {
       "requests": {
         "storage": "100Gi"
      },
       "selector": {
       "matchLabels": {
         "storage-tier": "gold"
      }
    }
   }
  }
}
```

11. 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
```

12. 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
```

13. 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-container-platform/3.4/single/installation-and-configuration/#install-config-storage-examples-gluster-example.

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

```
# oc get pods
```

15. 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/secretgit s/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-container-platform/3.4/single/installation-and-configuration/#gluster-volume-security.

5.2. Dynamic Provisioning of Volumes

Dynamic provisioning enables provisioning of Red hat Gluster Storage volume to a running application container without having to pre-create the volume. The volume will be created dynamically as the claim request comes in, and a volume of exactly the same size will be provisioned to the application containers.

Note

Dynamically provisioned Volumes are supported from Container Native Storage 3.4. If you have any statically provisioned volumes and require more information about managing it, then refer Section 5.1, “Static Provisioning of Volumes”

5.2.1. Configuring Dynamic Provisioning of Volumes

To configure dynamic provisioning of volumes, the administrator must define StorageClass objects that describe named "classes" of storage offered in a cluster. After creating a Storage Class, a secret for heketi authentication must be created before proceeding with the creation of persistent volume claim.

5.2.1.1. Registering a Storage Class

When configuring a StorageClass object for persistent volume provisioning, the administrator must describe the type of provisioner to use and the parameters that will be used by the provisioner when it provisions a PersistentVolume belonging to the class.

1. To create a storage class execute the following command:

```
# cat glusterfs-storageclass.yaml


apiVersion: storage.k8s.io/v1beta1
kind: StorageClass
metadata:
  name: gluster-container
provisioner: kubernetes.io/glusterfs
parameters:
  resturl: "http://127.0.0.1:8081"
  restuser: "admin"
  secretNamespace: "default"
  secretName: "heketi-secret"
```

   where,

   resturl: Gluster REST service/Heketi service url which provision gluster volumes on demand. The general format must be IPaddress:Port and this is a mandatory parameter for GlusterFS dynamic provisioner. If Heketi service is exposed as a routable service in openshift/kubernetes setup, this can have a format similar to http://heketi-storage-project.cloudapps.mystorage.com where the fqdn is a resolvable heketi service url.

   restuser : Gluster REST service/Heketi user who has access to create volumes in the trusted storage pool

   secretNamespace + secretName: Identification of Secret instance that contains the user password that is used when communicating with the Gluster REST service. These parameters are optional. Empty password will be used when both secretNamespace and secretName are omitted.

Note

When the persistent volumes are dynamically provisioned, the Gluster plugin automatically creates an endpoint and a headless service in the name gluster-dynamic-<claimname>. This dynamic endpoint and service will be deleted automatically when the persistent volume claim is deleted.

2. To register the storage class to Openshift, execute the following command:

```
# oc create -f glusterfs-storageclass.yaml
storageclass "gluster-container" created
```

3. To get the details of the storage class, execute the following command:

```
# oc describe storageclass gluster-container

Name: gluster-container
```

```
IsDefaultClass: No
Annotations: <none>
Provisioner: kubernetes.io/glusterfs
Parameters:
resturl=http://127.0.0.1:8081,restuser=admin,secretName=heketi-
secret,secretNamespace=default
No events.
```

5.2.1.2. Creating Secret for Heketi Authentication

To create a secret for Heketi authentication, execute the following commands:

1. Create an encoded value for the password by executing the following command:

   ```
   # echo -n "mypassword" | base64
   ```

   where “mypassword” is Heketi’s admin user password.

   For example:

   ```
   # echo -n "mypassword" | base64
   bXlwYXNzd29yZA==
   ```

2. Create a secret file. A sample secret file is provided below:

   ```
   # cat glusterfs-secret.yaml

   apiVersion: v1
   kind: Secret
   metadata:
     name: heketi-secret
     namespace: default
   data:
     # base64 encoded password. E.g.: echo -n "mypassword" | base64
     key: bXlwYXNzd29yZA==
   type: kubernetes.io/glusterfs
   ```

3. Register the secret on Openshift by executing the following command:

   ```
   # oc create -f glusterfs-secret.yaml
   secret "heketi-secret" created
   ```

5.2.1.3. Creating a Persistent Volume Claim

To create a persistent volume claim execute the following commands:

1. Create a Persistent Volume Claim file. A sample persistent volume claim is provided below:

   ```
   # cat glusterfs-pvc-claim1.yaml

   {
     "kind": "PersistentVolumeClaim",
     "apiVersion": "v1",
     "metadata": {
   ```

```
    "name": "claim1",
    "annotations": {
        "volume.beta.kubernetes.io/storage-class": "gluster-container"
    }
  },
  "spec": {
    "accessModes": [
      "ReadWriteOnce"
    ],
    "resources": {
      "requests": {
        "storage": "4Gi"
      }
    }
  }
}
```

2. Register the claim by executing the following command:

```
# oc create -f glusterfs-pvc-claim1.yaml
persistentvolumeclaim "claim1" created
```

3. To get the details of the claim, execute the following command:

```
# oc describe pvc <claim_name>
```

For example:

```
# oc describe pvc claim1

Name: claim1
Namespace: default
StorageClass: gluster-container
Status: Bound
Volume: pvc-54b88668-9da6-11e6-965e-54ee7551fd0c
Labels: <none>
Capacity: 4Gi
Access Modes: RWO
No events.
```

5.2.1.4. Verifying Claim Creation

To verify if the claim is created, execute the following commands:

1. To get the details of the persistent volume claim and persistent volume, execute the following command:

```
# oc get pv,pvc

NAME                                                CAPACITY   ACCESSMODES
RECLAIMPOLICY   STATUS     CLAIM                        REASON     AGE
pv/pvc-962aa6d1-bddb-11e6-be23-5254009fc65b   4Gi        RWO
Delete          Bound      storage-project/claim1              3m
```

```
NAME         STATUS    VOLUME
CAPACITY   ACCESSMODES   AGE
pvc/claim1   Bound     pvc-962aa6d1-bddb-11e6-be23-5254009fc65b   4Gi
RWO           4m
```

2. To validate if the endpoint and the services are created as part of claim creation, execute the following command:

```
# oc get endpoints,service

NAME                            ENDPOINTS
AGE
ep/storage-project-router
192.168.68.3:443,192.168.68.3:1936,192.168.68.3:80   28d
ep/gluster-dynamic-claim1
192.168.68.2:1,192.168.68.3:1,192.168.68.4:1         5m
ep/heketi                       10.130.0.21:8080
21d
ep/heketi-storage-endpoints
192.168.68.2:1,192.168.68.3:1,192.168.68.4:1         25d

NAME                             CLUSTER-IP       EXTERNAL-IP
PORT(S)                    AGE
svc/storage-project-router     172.30.166.64    <none>
80/TCP,443/TCP,1936/TCP   28d
svc/gluster-dynamic-claim1     172.30.52.17     <none>        1/TCP
5m
svc/heketi                       172.30.129.113   <none>
8080/TCP                   21d
svc/heketi-storage-endpoints   172.30.133.212   <none>        1/TCP
25d
```

5.2.1.5. Using the Claim in a Pod

Execute the following steps to use the claim in a pod.

1. To use the claim 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: claim1
```

```
# 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-container-platform/3.4/single/installation-and-configuration/#install-config-storage-examples-gluster-example.

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

```
# oc get pods

NAME                                  READY     STATUS
RESTARTS   AGE
storage-project-router-1-at7tf        1/1       Running         0
13d
busybox                               1/1       Running         0
8s
glusterfs-dc-192.168.68.2-1-hu28h     1/1       Running         0
7d
glusterfs-dc-192.168.68.3-1-ytnlg     1/1       Running         0
7d
glusterfs-dc-192.168.68.4-1-juqcq     1/1       Running         0
13d
heketi-1-9r47c                        1/1       Running         0
13d
```

3. 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-253:0-666733-
38050a1d2cdb41dc00d60f25a7a295f6e89d4c529302fb2b93d8faa5a3205fb9
                         10.0G     33.8M      9.9G   0% /
tmpfs                    23.5G         0     23.5G   0% /dev
tmpfs                    23.5G         0     23.5G   0%
/sys/fs/cgroup
/dev/mapper/rhgs-root
                         17.5G      3.6G     13.8G  21% /run/secrets
/dev/mapper/rhgs-root
                         17.5G      3.6G     13.8G  21%
/dev/termination-log
/dev/mapper/rhgs-root
                         17.5G      3.6G     13.8G  21%
/etc/resolv.conf
/dev/mapper/rhgs-root
                         17.5G      3.6G     13.8G  21%
/etc/hostname
```

```
/dev/mapper/rhgs-root
                        17.5G      3.6G     13.8G  21% /etc/hosts
shm                     64.0M         0     64.0M   0% /dev/shm
192.168.68.2:vol_5b05cf2e5404afe614f8afa698792bae
                         4.0G     32.6M      4.0G   1%
/usr/share/busybox
tmpfs                   23.5G     16.0K     23.5G   0%
/var/run/secrets/kubernetes.io/serviceaccount
tmpfs                   23.5G         0     23.5G   0% /proc/kcore
tmpfs                   23.5G         0     23.5G   0%
/proc/timer_stats
```

5.2.1.6. Deleting a Persistent Volume Claim

1. To delete a claim, execute the following command:

```
# oc delete pvc <claim-name>
```

For example:

```
# oc delete pvc claim1
persistentvolumeclaim "claim1" deleted
```

2. To verify if the claim is deleted, execute the following command:

```
# oc get pvc <claim-name>
```

For example:

```
# oc get pvc claim1
No resources found.
```

When the user deletes a persistent volume claim that is bound to a persistent volume created by dynamic provisioning, apart from deleting the persistent volume claim, Kubernetes will also delete the persistent volume, endpoints, service, and the actual volume. Execute the following commands if this has to be verified:

- To verify if the persistent volume is deleted, execute the following command:

```
# oc get pv <pv-name>
```

For example:

```
# oc get pv pvc-962aa6d1-bddb-11e6-be23-5254009fc65b
No resources found.
```

- To verify if the endpoints are deleted, execute the following command:

```
# oc get endpoints <endpointname>
```

For example:

```
# oc get endpoints gluster-dynamic-claim1
No resources found.
```

- To verify if the service is deleted, execute the following command:

```
# oc get service <servicename>
```

For example:

```
# oc get service gluster-dynamic-claim1
No resources found.
```

5.3. Volume Security

Volumes come with a UID/GID of 0 (root). For an application pod to write to the volume, it should also have a UID/GID of 0 (root). With the volume security feature the administrator can now create a volume with a unique GID and the application pod can write to the volume using this unique GID

Volume security for statically provisioned volumes

To create a statically provisioned volume with a GID, execute the following command:

```
$ heketi-cli volume create --size=100 --persistent-volume-file=pv001.json --
gid=590
```

In the above command, a 100G persistent volume with a GID of 590 is created and the output of the persistent volume specification describing this volume is added to the pv001.json file.

For more information about accessing the volume using this GID, refer https://access.redhat.com/documentation/en/openshift-container-platform/3.4/single/installation-and-configuration/#install-config-storage-examples-gluster-example.

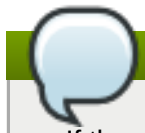
Volume security for dynamically provisioned volumes

Two new parameters, gidMin and gidMax, are introduced with dynamic provisioner. These values allows the administrator to configure the GID range for the volume in the storage class. To set up the GID values and provide volume security for dynamically provisioned volumes, execute the following commands:

1. Create a storage class file with the GID values. For example:

```
# cat glusterfs-storageclass.yaml

apiVersion: storage.k8s.io/v1beta1
kind: StorageClass
metadata:
  name:gluster-container
provisioner: kubernetes.io/glusterfs
parameters:
  resturl: "http://127.0.0.1:8081"
  restuser: "admin"
  secretNamespace: "default"
  secretName: "heketi-secret"
  gidMin: "2000"
  gidMax: "4000"
```



Note

If the gidMin and gidMax value are not provided, then the dynamic provisioned volumes will have the GID between 2000 and 2147483647.

2. Create a persistent volume claim. For more information see, Section 5.2.1.3, “Creating a Persistent Volume Claim”
3. Use the claim in the pod. Ensure that this pod is non-privileged. For more information see, Section 5.2.1.5, “Using the Claim in a Pod”
4. To verify if the GID is within the range specified, execute the following command:

```
# oc rsh busybox
```

```
$ id
```

For example:

```
$ id
uid=1000060000 gid=0(root) groups=0(root),2001
```

where, 2001 in the above output is the allocated GID for the persistent volume, which is within the range specified in the storage class. You can write to this volume with the allocated GID.

Note

When the persistent volume claim is deleted, the GID of the persistent volume is released from the pool.

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_1be433737b71419dc9b395e22125
5fb3/brick_c67fb97f74649d990c5743090e0c9176/brick
Brick2:
192.168.121.233:/var/lib/heketi/mounts/vg_0013ee200cdefaeb6dfedd28e50f
d261/brick_6ebf1ee62a8e9e7a0f88e4551d4b2386/brick
Brick3:
192.168.121.168:/var/lib/heketi/mounts/vg_e4b32535c55c88f9190da7b7efd1
fcab/brick_df5db97aa002d572a0fec6bcf2101aad/brick
Brick4:
192.168.121.233:/var/lib/heketi/mounts/vg_0013ee200cdefaeb6dfedd28e50f
d261/brick_acc82e56236df912e9a1948f594415a7/brick
Brick5:
192.168.121.168:/var/lib/heketi/mounts/vg_e4b32535c55c88f9190da7b7efd1
fcab/brick_65dceb1f749ec417533ddeae9535e8be/brick
Brick6:
192.168.121.172:/var/lib/heketi/mounts/vg_7ad961dbd24e16d62cabe10fd8bf
8909/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_3fa141bf2d09d30b899f2f260c49
4376/brick_9fb4a5206bdd8ac70170d00f304f99a5/brick
Brick2:
192.168.121.172:/var/lib/heketi/mounts/vg_7ad961dbd24e16d62cabe10fd8bf
8909/brick_dae2422d518915241f74fd90b426a379/brick
Brick3:
192.168.121.233:/var/lib/heketi/mounts/vg_5c6428c439eb6686c5e4cee56532
bacf/brick_b3768ba8e80863724c9ec42446ea4812/brick
Brick4:
192.168.121.172:/var/lib/heketi/mounts/vg_7ad961dbd24e16d62cabe10fd8bf
8909/brick_0a13958525c6343c4a7951acec199da0/brick
Brick5:
192.168.121.168:/var/lib/heketi/mounts/vg_17fbc98d84df86756e7826326fb3
3aa4/brick_af42af87ad87ab4f01e8ca153abbbee9/brick
Brick6:
192.168.121.233:/var/lib/heketi/mounts/vg_5c6428c439eb6686c5e4cee56532
bacf/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. Upgrading your Container-Native Storage Environment

This chapter describes the procedure to upgrade your environment from OpenShift 3.2 or 3.3 to OpenShift 3.4.

8.1. Pre-upgrade Task

Ensure you perform the following step before proceeding with the upgrade process.

1. Edit the glusterfs DeploymentConfig and Heketi DeploymentConfig to change the **`Strategy > type`** from **`Rolling`** to **`Recreate`** in the DeploymentConfig files.

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

For example:

```
apiVersion: v1
kind: DeploymentConfig
metadata:
  annotations:
  ...
  strategy:
    resources: {}
    rollingParams:
      intervalSeconds: 1
      maxSurge: 25%
      maxUnavailable: 25%
      timeoutSeconds: 600
      updatePeriodSeconds: 1
    type: Recreate
    ....
```

Important

Before upgrading your Container-Native Storage environment, you must ensure that you have upgraded and configured OpenShift Container Platform 3.4. For information on upgrading to OpenShift 3.4, see https://access.redhat.com/documentation/en/openshift-container-platform/3.4/single/installation-and-configuration/#upgrading-a-cluster

8.2. Upgrading your environment

1. Execute the following command to update the heketi packages:

```
# yum update heketi-templates -y
# yum update heketi-client -y
```

Note

When you run **`yum update heketi-templates -y`** command, the **`heketi-templates`** package is replaced with **`cns-deploy`**.

2. Execute the following command to delete the old gluster template:

```
# oc delete templates glusterfs
```

3. Execute the following command to register the new gluster template to OpenShift platform:

```
# oc create -f /usr/share/heketi/templates/glusterfs-template.yaml
```

4. Execute the following command to deploy the Red Hat Gluster Storage pods:

```
# oc process glusterfs | oc create -f -
daemonset "glusterfs" created
```

5. Execute the following command to delete the glusterfs Deploymentconfing for each gluster node:

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

For example,

```
# oc delete deploymentconfig  glusterfs-dc-node1.example.com
deploymentconfig "node1.example.com" deleted
```

6. Execute the following command to label each node that will run Red Hat Gluster Storage pods as **`glusterfs`**:

```
# oc label nodes <nodename> storagenode=glusterfs
```

For example,

```
# oc label nodes node1.example.com storagenode=glusterfs
node "node1.example.com" labeled
```

7. Execute the following command to delete the heketi template

```
# oc delete templates heketi
```

8. Execute the following command to install the heketi template:

```
# oc create -f /usr/share/heketi/templates/heketi-template.yaml
template "heketi" created
```

9. Execute the following command to create the heketi Service Account:

```
# oc create -f /usr/share/heketi/templates/heketi-service-account.yaml
serviceaccount "heketi-service-account" created
```

10. Execute the following command to grant the heketi Service Account the neccessary privileges:

```
# oc policy add-role-to-user edit system:serviceaccount:
<project_name>:heketi-service-account
```

For example,

```
# oc policy add-role-to-user edit system:serviceaccount:storage-
project:heketi-service-account
```

11. Execute the following command to delete the deployment configuration, service, and route for heketi:

```
# oc delete deploymentconfig heketi
# oc delete route heketi
# oc delete service heketi
```

12. Execute the following command to determine the mountable secret you should use for the heketi pods:

```
# oc describe sa heketi-service-account
Name:                   heketi-service-account
Namespace:              storage-project
Labels:                 none

Image pull secrets:     heketi-service-account-dockercfg-jmxyz

Mountable secrets:      heketi-service-account-dockercfg-jmxyz
                        heketi-service-account-token-fvhmb

Tokens:                 heketi-service-account-token-fvhmb
                        heketi-service-account-token-lzifj
```

You should select a secret that is mountable, in your project namespace, and of type **`kubernetes.io/service-account-token`** . You can use the following command to inspect a given secret:

```
# oc describe secret <secret name>
```

For example:

```
# oc describe secret heketi-service-account-token-fvhmb
Name:           heketi-service-account-token-fvhmb
Namespace:      storage-project
Labels:         none
Annotations:    kubernetes.io/service-account.name=heketi-service-
account
                kubernetes.io/service-account.uid=3afe5e84-c863-11e6-
8736-525400556335
Type:           kubernetes.io/service-account-token

Data
====
ca.crt:           1070 bytes
namespace:        4 bytes
```

```
service-ca.crt:  2186 bytes
token:
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY
2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJ
hcGxvIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZWNyZXQubmFtZSI6Imhla
2V0aS1zZXJ2aWNlLWFjY291bnQtdG9rZW4tZnZobWIiLCJrdWJlcm5ldGVzLmlvL3NlcnZ
pY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC5uYW1lIjoiaGVrZXRpLXNlcnZpY2UtYWNjb
3VudCIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50LnV
pZCI6IjNhZmU1ZTg0LWM4NjMtMTFlNi04NzM2LTUyNTQwMDU1NjMzNSIsInN1YiI6InN5c
3RlbTpzZXJ2aWNlYWNjb3VudDphcGxvOmhla2V0aS1zZXJ2aWNlLWFjY291bnQifQ.Oeso
jdalmQ9pAGR6CQ03vJizrVOGTrKYwvzVbEsms422YkJ1vFLI0idlA3Sxw3C_ZXamOEidYf
D7n5A8099qjSyfbEhSZDpaYDfpH-o5gnyjNFpuBkZwPGE9KAEmStOGMtNhO-
xAEtrD0UkHtDvy8JKnlrzMStP7NCfpGLL36X4qMx9exL3WGlWcth0kV0mGb3m-
NWxr2w7twZK8xP-N8sMbdyD4s-
N8naEBxBGAbNxufBBj6FXzzQSg5d1vIYls_ZgqbZV9Dn7fz9aB_GD9UfxI42EER8bzpoby
Nhxmu5GFfhX1TqE0kqfq1GR7P_Y6TuXrP2aYKPcR9ngZ56grLA
```

13. 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_SECRETNAME=<heketi-service-account secret> | 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_SECRETNAME=<secret from step 12 example> | oc create -
f -


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

14. 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-pj9ea     1/1       Running   0          1d
deploy-heketi-1-m7x8g   1/1       Running   0          1m
glusterfs-41lfl         1/1       Running   0          1m
glusterfs-dtyr4         1/1       Running   0          1m
glusterfs-ral2d         1/1       Running   0          1m
```

Chapter 9. Troubleshooting

This chapter describes the most common troubleshooting scenarios related to Container-Native Storage.

Viewing Log Files

Viewing Red Hat Gluster Storage Container Logs

Debugging information related to Red Hat Gluster Storage containers is stored on the host where the containers are started. Specifically, the logs and configuration files can be found at the following locations on the openshift nodes where the Red Hat Gluster Storage server containers run:

- /etc/glusterfs
- /var/lib/glusterd
- /var/log/glusterfs

Viewing Heketi Logs

Debugging information related to Heketi is stored locally in the container or in the persisted volume that is provided to Heketi container.

You can obtain logs for Heketi by running the **`docker logs`** ***`container-id`*** command on the openshift node where the container is being run.

Heketi command returns with no error or empty error like Error

Sometimes, running heketi-cli command returns with no error or empty error like **Error**. It is mostly due to heketi server not properly configured. You must first ping to validate that the Heketi server is available and later verify with a **curl** command and **/hello endpoint**.

Heketi reports an error while loading the topology file

Running heketi-cli reports : Error "Unable to open topology file" error while loading the topology file. This could be due to the use of old syntax of single hyphen (-) as prefix for json option. You must use the new syntax of double hyphens and reload the topology file.

cURL command to heketi server fails or does not respond

If the router or heketi is not configured properly, error messages from the heketi may not be clear. To troubleshoot, ping the heketi service using the endpoint and also using the IP address. If ping by the IP address succeeds and ping by the endpoint fails, it indicates a router configuration error.

After the router is setup properly, run a simple curl command like the following:

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

If heketi is configured correctly, a welcome message from heketi is displayed. If not, check the heketi configuration.

Heketi fails to start when Red Hat Gluster Storage volume is used to store heketi.db file

Sometimes Heketi fails to start when Red Hat Gluster Storage volume is used to store heketi.db and reports the following error:

```
[heketi] INFO 2016/06/23 08:33:47 Loaded kubernetes executor
[heketi] ERROR 2016/06/23 08:33:47
/src/github.com/heketi/heketi/apps/glusterfs/app.go:149: write
/var/lib/heketi/heketi.db: read-only file system
ERROR: Unable to start application
```

The read-only file system error as shown above could be seen while using a Red Hat Gluster Storage volume as backend. This could be when the quorum is lost for the Red Hat Gluster Storage volume. In a replica-3 volume, this would be seen if 2 of the 3 bricks are down. You must ensure the quorum is met for heketi gluster volume and it is able to write to heketi.db file again.

Even if you see a different error, it is a recommended practice to check if the Red Hat Gluster Storage volume serving heketi.db file is available or not. Access deny to heketi.db file is the most common reason for it to not start.

Chapter 10. 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**

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

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

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

2. **Remove all OpenShift objects**

    a. 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.

    b. 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**

    a. 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**

    a. Execute the following command to cleanup the disks:

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

Appendix A. Manual Deployment

The following section covers the steps required to manually deploy Container-Native Storage.

A.1. Installing the Templates

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

1. Use the newly created containerized Red Hat Gluster Storage project:

```
# oc project project_name
```

For example,

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

2. Execute the following commands to install the templates:

```
# oc create -f /usr/share/heketi/templates/deploy-heketi-template.yaml
template "deploy-heketi" created
```

```
# oc create -f /usr/share/heketi/templates/glusterfs-template.yaml
template "glusterfs" created
```

```
# oc create -f /usr/share/heketi/templates/heketi-service-account.yaml
serviceaccount  "heketi-service-account" created
```

```
# oc create -f /usr/share/heketi/templates/heketi-template.yaml
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              7 (6 blank)
3
glusterfs       GlusterFS DaemonSet template               0 (all set)
1
heketi          Heketi service deployment template         7 (6 blank)
3
```

4. Execute the following command to verify that the serviceaccount is created:

```
# oc get serviceaccount heketi-service-account
```

For example:

```
# oc get serviceaccount heketi-service-account
NAME                     SECRETS   AGE
heketi-service-account   2         7d
```

A.2. 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. Execute the following command to label all nodes that will run Red Hat Gluster Storage pods:

```
# oc label node <NODENAME> storagenode=glusterfs
```

For example:

```
# oc label nodes 192.168.90.3 storagenode=glusterfs
node "192.168.90.3" labeled
```

Repeat this command for every node that will be in the GlusterFS cluster.

Verify the label has set properly by running the following command:

```
# oc get nodes --show-labels
192.168.90.2   Ready                      12d
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.i
o/hostname=192.168.90.2,storagenode=glusterfs
192.168.90.3   Ready                      12d
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.i
o/hostname=192.168.90.3,storagenode=glusterfs
192.168.90.4   Ready                      12d
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.i
o/hostname=192.168.90.4,storagenode=glusterfs
192.168.90.5   Ready,SchedulingDisabled   12d
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.i
o/hostname=192.168.90.5
```

3. Execute the following command to deploy the Red Hat Gluster Storage pods:

```
# oc process glusterfs | oc create -f -
daemonset "glusterfs" created
```

Note

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.

4. Execute the following command to grant the heketi Service Account the neccessary privileges:

```
# oc policy add-role-to-user edit system:serviceaccount:
<project_name>:heketi-service-account
```

For example:

```
# oc policy add-role-to-user edit system:serviceaccount:storage-
project:heketi-service-account
```

5. Execute the following command to determine the mountable secret you should use for the heketi pods:

```
# oc describe sa heketi-service-account
Name:                   heketi-service-account
Namespace:              storage-project
Labels:                 none

Image pull secrets:     heketi-service-account-dockercfg-jmxyz

Mountable secrets:      heketi-service-account-dockercfg-jmxyz
                        heketi-service-account-token-fvhmb

Tokens:                 heketi-service-account-token-fvhmb
                        heketi-service-account-token-lzifj
```

You should select a secret that is mountable, in your project namespace, and of type **`kubernetes.io/service-account-token`** . You can use the following command to inspect a given secret:

```
# oc describe secret <secret name>
```

For example:

```
# oc describe secret heketi-service-account-token-fvhmb
Name:           heketi-service-account-token-fvhmb
Namespace:      storage-project
Labels:         none
Annotations:    kubernetes.io/service-account.name=heketi-service-
account
                kubernetes.io/service-account.uid=3afe5e84-c863-11e6-
```

```
8736-525400556335
Type:          kubernetes.io/service-account-token

Data
====
ca.crt:          1070 bytes
namespace:       4 bytes
service-ca.crt:  2186 bytes
token:
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY
2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJ
hcGxvIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZWNyZXQubmFtZSI6Imhla
2V0aS1zZXJ2aWNlLWFjY291bnQtdG9rZW4tZnZobWIiLCJrdWJlcm5ldGVzLmlvL3NlcnZ
pY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC5uYW1lIjoiaGVrZXRpLXNlcnZpY2UtYWNjb
3VudCIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50LnV
pZCI6IjNhZmU1ZTg0LWM4NjMtMTFlNi04NzM2LTUyNTQwMDU1NjMzNSIsInN1YiI6InN5c
3RlbTpzZXJ2aWNlYWNjb3VudDphcGxvOmhla2V0aS1zZXJ2aWNlLWFjY291bnQifQ.Oeso
jdalmQ9pAGR6CQ03vJizrVOGTrKYwvzVbEsms422YkJ1vFLI0idlA3Sxw3C_ZXamOEidYf
D7n5A8099qjSyfbEhSZDpaYDfpH-o5gnyjNFpuBkZwPGE9KAEmStOGMtNhO-
xAEtrD0UkHtDvy8JKnlrzMStP7NCfpGLL36X4qMx9exL3WGlWcth0kV0mGb3m-
NWxr2w7twZK8xP-N8sMbdyD4s-
N8naEBxBGAbNxufBBj6FXzzQSg5d1vIYls_ZgqbZV9Dn7fz9aB_GD9UfxI42EER8bzpoby
Nhxmu5GFfhX1TqE0kqfq1GR7P_Y6TuXrP2aYKPcR9ngZ56grLA
```

6. 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_SECRETNAME=<heketi-service-account secret> | 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_SECRETNAME=heketi-service-account-token-fvhmb | oc
create -f -

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

7. 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-pj9ea     1/1       Running   0          1d
deploy-heketi-1-m7x8g   1/1       Running   0          1m
```

```
glusterfs-41lfl         1/1       Running   0          1m
glusterfs-dtyr4         1/1       Running   0          1m
glusterfs-ral2d         1/1       Running   0          1m
```

A.3. 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/** directory.

```
{
    "clusters": [
                {
                   "nodes": [
                                {
                         "node": {
                             "hostnames": {
                                 "manage": [
                                     "node1.example.com"
                                 ],
                                 "storage": [
                                     "192.168.121.168"
                                  ]
                              },
                              "zone": 1
                          },
                          "devices": [
                              "/dev/sdb",
                              "/dev/sdc",
                              "/dev/sdd",
                              "/dev/sde"
                          ]
                },...
```

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 **/usr/share/heketi/topology-sample.json** file only sets up 4 nodes with 8 drives each.

Important

Heketi stores its database on a Red Hat Gluster Storage volume. Heketi service does not respond if the volume is down.

To resolve this issue, restart the gluster pods hosting the Heketi volume.

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 remove all resources used to bootstrap 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
```

8. 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_SECRETNAME=<heketi-service-account secret> | 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_SECRETNAME=<secret from step 4 example> | oc create -f
-


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

9. 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
```

Appendix B. 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-container-platform/3.4/single/installation-and-configuration/#identity-providers.

Appendix C. 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 D. 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 E. 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 F. Known Issues

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

BZ#1409848

The following two lines might be repeatedly logged in the rhgs-server-docker container/gluster container logs.

```
[MSGID: 106006] [glusterd-svc-mgmt.c:323:glusterd_svc_common_rpc_notify] 0-
management: nfs has disconnected from glusterd.
[socket.c:701:__socket_rwv] 0-nfs: readv on
/var/run/gluster/1ab7d02f7e575c09b793c68ec2a478a5.socket failed (Invalid
argument)
```

These logs are added as glusterd is unable to start the NFS service. There is no functional impact as NFS export is not supported in Containerized Red Hat Gluster Storage.