Configure GlusterFS for use as Shared Storage

Introduction

When working with GlusterFS as a shared storage provider for a Kubernetes cluster, the following components need to be installed and configured.

- The GlusterFS server on a cluster of at least 3 node (VMs).
- The GlusterFS client on the nodes that will be using the shared storage.
- A Heketi server used to administer the GlusterFS storage using the heketi-cli and a Kubernetes StorageClass.
- A Kubernetes StorageClass object that points to the Heketi server

The following approaches can be taken when installing these components.

- So called, "native" installation
- Containerized installation

This document only covers using Red Hat Enterprise Linux (RHEL)

References to vendor/product documentation are provided for more detail.

Install "native" GlusterFS server

This section describes the steps to installing the GlusterFS server directly on RHEL (not in a container).

If you are using the GlusterFS server installed in a Docker container, (obviously) this section must be skipped.

NOTE: The Gluster server topology and storage volumes do not need to be configured manually. In a later section the hekeit-cli is used to configure the server topology based on a YAML configuration file. The heketi-cli can then be used to define mountable storage volumes. Many sources you find on the Internet describe extensive manual steps to configure storage devices and volumes. You can ignore/skip all those configuration instructions. The Gluster server machines need only to have raw disk devices defined on them.

Installing GlusterFS server on RHEL rather than in a container may be a matter of preference and your level of comfort with containers vs native processes. This section is for those who are more comfortable with working native RHEL processes.

- A yum repository needs to be configured to get the GlusterFS, and Heketi RPMs. Here is a sample yum `gluster.repo` file that needs to be created in `/etc/yum.repos.d/`.

```
[Gluster_4.0]
name=Gluster 3.13
baseurl=http://mirror.centos.org/centos/7/storage/$basearch/gluster-4.0/
gpgcheck=0
enabled=1
```

NOTE: Go out to http://mirror.centos.org/ and walk down the `centos` and `storage` directory tree to find out the latest `gluster` release and update the `gluster.repo` `baseurl` accordingly.

- Disable SE-Linux enforcement (*TBD:* I'm not convinced this is necessary. Gluster doc does not indicate this is necessary. Need to test installing and running with `SELINUX=enforcing`.)
    - Edit `/etc/selinux/config` and set `SELINUX=disabled`
    - Reboot (`shutdown -r now`)
- Configure `dm_thin_pool` kernel module

```
> modprobe dm_thin_pool
```

- Configure a `dm_thin_pool` in a conf file in `/etc/modules-load.d/` to support setting it at machine reboot. (The files in `/etc/modules-load.d/` are used to configure the kernel when the machine boots.)

```
> echo dm_thin_pool >> /etc/modules-load.d/dm_thin_pool.conf
```

- Install glusterfs-server

```
> yum -y install glusterfs-server
```

- Enable and start the `glusterd` daemon.

```
> systemctl enable --now glusterd
```

- Configure passwordless ssh for root among the GlusterFS servers in the cluster. (This is a multi-way configuration. Each server needs to be able to ssh to the other servers in the cluster.) (*TBD* Add the detail of the commands.)

- Configure firewalld to open gluster server ports (*TBD* For now, stop,disable firewalld. See the Gluster Doc on ports that need to be open.)

Install "native" GlusterFS client

NOTE: Ansible playbook to do this: `icp_install_glusterfs_client.yml` with supporting file `dm_thin_pool.conf`.

See GlusterFS documentation for client installation: Accessing Data: Setting up GlusterFS Clients

Install Heketi on RHEL (aka "native" install)

This section describes the installation of Heketi on RHEL. Heketi and the Heketi client will be running directly on the VM rather than in Kubernetes pods.

Heketi is only used for administration of the GlusterFS storage. At run time the clients using the storage do not use Heketi. The clients use the GlusterFS client and work directly with the GlusterFS servers.

See Managing Volumes using Heketi for Red Hat documentation on the Heketi installation and configuration.

A reasonable place to install Heketi is on one of the GlusterFS servers.

NOTE: Heketi is not architected to run in a highly available configuration, i.e., a cluster. The Heketi server runs as a singleton. The `heketi.db` should be backed up regularly in order to recover from a catastrophic loss of the file system or machine where the Heketi server is

deployed.

- A yum repository needs to be configured to get the GlusterFS, and Heketi RPMs. Here is a sample yum `glusterfs.repo` file that needs to be created in `/etc/yum.repos.d/`.

```
[Gluster_4.0]
name=Gluster 3.13
baseurl=http://mirror.centos.org/centos/7/storage/$basearch/gluster-4.0/
gpgcheck=0
enabled=1
```

- It is assumed a yum repo has been configured that points to a GlusterFS repository.
- Install Heketi server and Heketi CLI

```
> yum -y install heketi
> yum -y install heketi-client
```

- Confirm that port 8080 is not already in use. (`netstat -an | grep 8080`) The heketi server uses port 8080 by default, but that can be changed in the `/etc/heketi/heketi.json` configuration file. (If you run the heketi server on an ICP master node, you will need to use a port other than 8080 since the ICP admin console process uses 8080.)

- Make sure the command line options on the `ExecStart` property in `/usr/lib/systemd/system/heketi.service` use double-dash (--) rather than a single dash (-). It is likely the only option will be `--config`.

- Set up passwordless `ssh` between the heketi server node and all of the gluster server nodes for the gluster cluster to be managed.

```
> ssh-keygen -b 4096 -t rsa -f /etc/heketi/heketi_key -N ""
> ssh-copy-id -i /etc/heketi/heketi_key.pub root@gluster##.xxx.yyy
```

where gluster##.xxx.yyy represents each of the VMs in your gluster server cluster.

- Change the owner and group of the heketi keys to heketi. (The heketi user got created as part of the heketi install.)

```
> chown heketi:heketi /etc/heketi/heketi_key*
```

- Modify the `/etc/heketi/heketi.json` file for your installation.
    - Things to check in particular:
    - port: something not already in use
    - use_auth: true
    - admin key
    - user key
    - executor: ssh
    - sshexec:
        - keyfile: `/etc/heketi/heketi_key`
        - user: root
        - port: 22
        - fstab: `/etc/fstab`
    - The kubeexec section can be ignored since sshexec is being used.
    - The heketi database is in the default location `/var/lib/heketi/heketi.db`.
    - The remainder of the options can be left at the defaults.

- Enable and start the heketi server.

```
> systemctl enable --now heketi
```

- Smoke test for heketi server:

```
> curl http://localhost:8080/hello
Hello from Heketi
```

In the above URL, you will need to use the port you configured for the Heketi server.

NOTE: This native install of Heketi has a single point of failure in the `heketi.db` being located on the node where it gets installed. *TBD:* Revisit this to create a `heketi-db` shared volume in GlusterFS after the initial installation of Heketi. Stop Heketi. Move the `heketi.db` file out of `/var/lib/heketi` to some temporary location. Mount the heketi-db shared volume on `/var/lib/heketi` and copy the existing `heketi.db` into the shared volume. Heketi can then be installed on the other master nodes with that same shared volume mounted on `/var/lib/heketi`. Only one heketi server an be running at any given time to avoid issues with multiple servers accessing the `heketi.db` file concurrently. Hence, running the Heketi server in a Kubernetes pod is a stronger approach.

Things that can go wrong with the Heketi install

unknown shorthand flag: 'c'

With Heketi 4.0.0, the service wouild fail to start. The error in `/var/log/messages` was:

```
Error: unknown shorthand flag: 'c' in -config=/etc/heketi/heketi.json
pvs-master01 heketi: unknown shorthand flag: 'c' in -config=/etc/heketi/heketi.json
```

There is nothing actually wrong with the heketi.json. (You can paste the content into a JSON validator to convince yourself.)

The problem is the -config option to the Heketi executable.

See https://bugzilla.redhat.com/show_bug.cgi?id=1439120

You need to edit the `/usr/lib/systemd/system/heketi.service` file and change the `-config` to `--config`.

Creating the GlusterFS topology using the native heketi-cli.

- Create a topology.json file that represents your GlusterFS server cluster.

NOTE: The GlusterFS documentation and other sources indicate the "manage" attribute for each hostname dictionary should be a fully qualified host name and the storage attribute for each host name should be an IP address. *TBD:* Whether that is actually a strict requirement has not been fully confirmed. Some deployments have shown that using IP addresses for both attributes also appears to work.

Here is a sample topology.json for a 3 server cluster.

```
{
  "clusters": [
    {
      "nodes": [
        {
          "node": {
            "hostnames": {
```

```
                "manage": [
                  "gluster01.yyy.zzz"
                ],
                "storage": [
                  "172.16.20.15"
                ]
              },
              "zone": 1
            },
            "devices": [
              "/dev/sdb",
              "/dev/sdc"
            ]
          },
          {
            "node": {
              "hostnames": {
                "manage": [
                  "gluster02.yyy.zzz"
                ],
                "storage": [
                  "172.16.20.16"
                ]
              },
              "zone": 1
            },
            "devices": [
              "/dev/sdb",
              "/dev/sdc"
            ]
          },
          {
            "node": {
              "hostnames": {
                "manage": [
                  "gluster03.yyy.zzz"
                ],
                "storage": [
                  "172.16.20.17"
                ]
              },
              "zone": 1
            },
            "devices": [
              "/dev/sdb",
              "/dev/sdc"
            ]
          }
        ]
      }
    ]
  }
```

If you tend to fat-finger JSON files, it is a good idea to run the `topology.json` content through

a JSON validator. (It is easy to incorrectly edit JSON syntax.) Search the Internet for a JSON validator to your liking.

To get general usage help with the heketi-cli, use `heketi-cli --help`.

To get more specific help use `heketi-cli command --help` where `command` is one of the heketi-cli commands. (The "commands" are not verbs but rather an object type or class, e.g., topology, cluster, volume, node and device.)

To get specific help for the "commands" associated with a given object type use `heketi-cli type command --help`, for example `heketi-cli topology load --help`.

When working with heketi-cli it is very convenient to export values for the following environment variables:

```
> export HEKETI_CLI_SERVER=http://localhost:8081
> export HEKETI_CLI_USER=admin
> export HEKETI_CLI_KEY=passw0rd
```

NOTE: When providing the Heketi server URL, be sure not to include a trailing slash on the URL. So for example `http://localhost:8081/` will lead to problems. The trailing slash causes an issue for a volume create operation, for example.

You may want to create a shell file named something like `configHeketiCLI.sh` and put the export statements there and then you can source the file. Be careful to protect the file as it has user and password in it.

The above example assumes the Heketi server is listening on port 8081 rather than the default port 8080. The port the Heketi server is using is defined in the heketi.json. In the above, the user and key are based on what is defined in the heketi.json that you configured before starting the Heketi server.

- Use the heketi-cli to load the topology.json file.

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

Obviously, the above command is assumed to have been run from the directory where the topology.json file is located.

Once the topology has successfully loaded you can use `heketi-cli topology info` to see

information about the topology.

At this point you are ready to create mountable volumes that can be used by the ICP master nodes for shared storage.

Things that can go wrong with Heketi CLI

Error: Unable to get topology information: Unknown user

This is caused by not having a --user argument or HEKETI_CLI_USER set.

Error: Unable to get topology information: signature is invalid

This error may occur when you run:

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

The above assumes the `topology.json` file is in the current directory.

The "signature" in question has nothing to do with the content of the `topology.json` file or some digital signature it might be missing.

When you run an heketi-cli command, and you have JWT authentication enabled (see your `heketi.json`), you need to provide a user and "key" (aka password or secret). One way to provide the key is to set the environment variable, HEKETI_CLI_KEY. You can also pass it in on the `heketi-cli` command line with the `--secret` option. (See `heketi-cli --help` usage info.)

If the password/secret you provide does not match up with the user and key in the heketi.json file in current use by the Heketi service, then you will get the "signature is invalid" error.

NOTE: If you change anything in the `heketi.json` file you need to restart the Heketi service (`systemctl restart heketi`) to pick up the changes.

Using heketi-cli to create persistent volumes for ICP master node shared

file systems

This section applies to an HA ICP deployment where there are 3 or 5 master nodes. If you are doing a simple sandbox installation with a single master node, this section can be skipped.

NOTE: The heketi-cli commands in this section assume that the following environment variables have been set appropriately:

- HEKETI_CLI_SERVER
- HEKETI_CLI_USER
- HEKETI_CLI_KEY
- Get the glusterfs cluster ID. The topology info will have the cluster ID.

```
heketi-cli topology info
```

NOTE: The heketi doc recommends not providing a name for the created persistent volumes. That is debatable. If you have a good naming convention that ensures uniqueness, then using a descriptive name seems like a good idea rather than using the default names that are generated strings which have no descriptive value.)

- Create a volume for the master audit log. In this example a 10GB volume is created.

```
> heketi-cli volume create --size=10 --clusters=042e3eb4b386b086c17d9d947e8ba885
10GiB volume created for use by master nodes for shared audit log (mounted at /var/
Name: vol_de785f066ce0e0ce86c39c5fb920682c
Size: 10
Volume Id: de785f066ce0e0ce86c39c5fb920682c
Cluster Id: 042e3eb4b386b086c17d9d947e8ba885
Mount: 172.16.20.17:vol_de785f066ce0e0ce86c39c5fb920682c
Mount Options: backup-volfile-servers=172.16.20.15,172.16.20.16
Durability Type: replicate
Distributed+Replica: 3
```

- Make note of the mountable volume host and name. That is what is used in the mount command and the entry in `/etc/fstab` on each of the master nodes. In the above example it is: `172.16.20.17:vol_de785f066ce0e0ce86c39c5fb920682c` You can replace the

IP address with an actual hostname if you have an entry in DNS or `/etc/hosts` on the master nodes for the GlusterFS servers.

- Mount the volume on each master node (Note the colon used to separate the backup servers rather than a comma.)

```
> mount -t glusterfs -o backup-volfile-servers=gluster01.xxx.yyy:gluster02.yyy.zzz
```

- Add a line to `/etc/fstab` (*TBD:* Does the backup-volfile-servers option work in fstab?)

```
gluster03.xxx.yyy:vol_de785f066ce0e0ce86c39c5fb920682c /var/lib/icp/audit glusterfs
```

- Create a volume for the master docker registry. In this example a 50GB volume is created.

```
> heketi-cli volume create --size=50 --clusters=042e3eb4b386b086c17d9d947e8ba885
Name: vol_ccdb21cfd1e83cf9c3299207f66fb705
Size: 50
Volume Id: ccdb21cfd1e83cf9c3299207f66fb705
Cluster Id: 042e3eb4b386b086c17d9d947e8ba885
Mount: 172.16.20.17:vol_ccdb21cfd1e83cf9c3299207f66fb705
Mount Options: backup-volfile-servers=172.16.20.15,172.16.20.16
Durability Type: replicate
Distributed+Replica: 3
```

- Mount the volume on each master node (Note the colon used to separate the backup servers rather than a comma.)

```
> mount -t glusterfs -o backup-volfile-servers=gluster01.xxx.yyy:gluster02.yyy.zzz
```

- Add a line to `/etc/fstab` (*TBD:* Does the backup-volfile-servers option work in fstab?)

```
gluster03.xxx.yyy:vol_ccdb21cfd1e83cf9c3299207f66fb705 /var/lib/registry glusterfs
```