Setup a scalable high availability GlusterFS network filesystem with Docker Swarm on Ubuntu Server 20.04 LTS

Link: https://florianmuller.com/setup-a-scalable-high-availability-glusterfs-network-filesystem-with-docker-swarm-on-ubuntu-server-20-04-lts

In this quick guide we are going to setup the scalable GlusterFS filesystem for a four node Docker Swarm cluster on Ubuntu 20.04 LTS. More precise: we replace an existing persistent NFS storage on the cluster with the new GlusterFS and make it available under the same old mount point as the old NFS. Therefore we can easily redeploy all of our containers from our docker-compose files, without changing anything. Lets get started!

Install GlusterFS

First we ssh into our master node of our docker swarm, in our case `pi-cluster-1`. There we update our repos and install GlusterFS together with its necessary dependencies:

```
sudo apt update && sudo apt install software-properties-common glusterfs-server -y
```

Now finish and start the GlusterFS service:

```
sudo systemctl start glusterd
sudo systemctl enable glusterd
```

Amazing! **Now repeat these install steps on ALL 3 remaining nodes!**

Prepare ufw firewall (if set):

If you are using a firewall on your ubuntu nodes, which I highly recommend, you need to open some ports to give GlusterFS a chance to communicate with the nodes. In our case we are using the built in ufw, **but we are NOT adding the standard ports itself, as glusterfs is generating new ports per bricks on a glusterfs volume, that can give you lots of problems later on.**

Instead we are allowing on every node explicitly the other nodes to be incoming traffic without restrictions, so you do not need to worry about future port changes by glusterfs. **Add the following rules ONLY on the cluster they are meant for. not on any others (replace with your cluster IP addresses):**

```
#cluster1 (192.168.0.1)
sudo ufw allow proto any from 192.168.0.2 to 192.168.0.1
sudo ufw allow proto any from 192.168.0.3 to 192.168.0.1
sudo ufw allow proto any from 192.168.0.4 to 192.168.0.1

#cluster2 (192.168.0.2)
sudo ufw allow proto any from 192.168.0.1 to 192.168.0.2
sudo ufw allow proto any from 192.168.0.3 to 192.168.0.2
sudo ufw allow proto any from 192.168.0.4 to 192.168.0.2

#cluster3 (192.168.0.3)
sudo ufw allow proto any from 192.168.0.1 to 192.168.0.3
sudo ufw allow proto any from 192.168.0.2 to 192.168.0.3
sudo ufw allow proto any from 192.168.0.4 to 192.168.0.3

#cluster4 (192.168.0.4)
sudo ufw allow proto any from 192.168.0.1 to 192.168.0.4
sudo ufw allow proto any from 192.168.0.2 to 192.168.0.4
sudo ufw allow proto any from 192.168.0.3 to 192.168.0.4
```

Repeat these firewall steps on ALL 3 remaining nodes, and use the rules applying to the node!

If, and only IF, you run into any issues (which you shouldn't) with glusterfs access or syncing or you get a split brain, than check out the specific ports per cluster as well, and see if you might need them. These are the glusterfs standard ports from their documentation:

```
#glusterfs
111 (portmapper)
24007/tcp
24008/tcp
49152 – 49155
```

Probe and connect the nodes to GlusterFS:

Now we're going to have Gluster probe all of the nodes. **This will be done from the master node.** I'm going to stick with my example of four nodes, which are these hostnames:
`pi-cluster-1`
`pi-cluster-2`
`pi-cluster-3`
`pi-cluster-4`
Before you issue the command, you'll need to change to the superuser with:

```
sudo -s
```

For some reason in my case the hostnames did not work and I had to use the local IP address of my nodes. So if you get a connection error from GlusterFS while probing, use the local IP address instead.

If you use hostnames probe like this (replace with YOUR nodes hostnames):

```
gluster peer probe pi-cluster-1; gluster peer probe pi-cluster-2; gluster peer probe pi-cluster-3; gluster peer probe pi-cluster-4;
```

When you want to use hostnames (recommended) and you get an error at first, try to lookup and set your hostnames in /etc/hosts on each cluster:

```
sudo nano /etc/hosts
```

Edit the host file like this, but remember to replace with your cluster nodes IP addresses:

```
127.0.0.1 localhost
192.168.0.1 pi-cluster-1
192.168.0.2 pi-cluster-2
192.168.0.3 pi-cluster-3
```

```
192.168.0.4 pi-cluster-4
```

If you still have issues probing, you can also use direct local IP addresses (not recommended, but works) probe like this (replace with YOUR nodes IP addresses):

```
gluster peer probe 192.168.0.1; gluster peer probe 192.168.0.2; gluster peer probe 192.168.0.3; gluster peer probe 192.168.0.4;
```

Great! Now you have successfully added the nodes to each other in Gluster. To verify your Gluster pool, type this on the Master node:

```
gluster pool list
```

If you get a list of your nodes you are good to go and should exit the superuser mode with:

```
exit
```

Create the GlusterFS Volume on your drives:

Now we create our GlusterFS volume mount point. I called mine dockerfiles, you can choose whatever you want here:

```
sudo mkdir -p /gluster/dockerfiles
```

You need to run and create this mount point on ALL nodes and machines!

Once done, you can create the volume across the cluster with this command (**run only once on the master**):

If you used hostnames before:

```
sudo gluster volume create staging-gfs replica 4 pi-cluster-1:/gluster/dockerfiles pi-cluster-2:/gluster/dockerfiles pi-cluster-3:/gluster/dockerfiles pi-cluster-4:/gluster/dockerfiles force
```

If you used IP addresses before:

```
sudo gluster volume create staging-gfs replica 4 192.168.0.1:/gluster/dockerfiles
192.168.0.2:/gluster/dockerfiles 192.168.0.3:/gluster/dockerfiles
192.168.0.4:/gluster/dockerfiles force
```

Startup the Gluster volume with (on the master node only):

```
sudo gluster volume start staging-gfs
```

The volume is now up and running, but we need to make sure the volume will mount on the same mount point as our old NFS storage. We also need to migrate our container data from the prior NFS storage to the new glusterFS volume first.

Migrate container data from NFS:

Create the directory where we will have our shared GlusterFS volume mounted to.
IMPORTANT if you migrate from NFS: This should be a temporary name first, as we copy back and forth from it. So for our case where we migrate we create a temporary mount point with:

```
#if not exists:
sudo mkdir -p /mnt/docker
sudo mkdir -p /mnt/tempdockerfiles
```

Next we mount our GlusterFS volume to it:

```
sudo mount.glusterfs localhost:/staging-gfs /mnt/tempdockerfiles
```

Perfect! Now we copy and migrate our docker container data over to GlusterFS.
IMPORTANT: If you have write sensitive data/services, you need to stop and remove first all our stacks and services. Get a list of what running on your stack with: `docker stack ls`

Now lets copy our existing NFS docker data from our actual docker used shared volume /mnt/docker to our temp GlusterFS mount point:

```
sudo cp -a /mnt/docker/. /mnt/tempdockerfiles/
```

This may take a while, but once finished you should have all your actuall container data under /mnt/tempdockerfiles as well. You can verify this with: `ls -l /mnt/tempdockerfiles`

Once you are sure all data has been copied, we unmount our NFS storage volume and our GlusterFS with. **Run the following commands on ALL FOUR NODES**:

```
sudo umount /mnt/docker/
sudo umount /mnt/tempdockerfiles/
```

Make sure they are unmounted with `ls -l /mnt/tempdockerfiles/` that should give you an empty result (0 files).

As we have copied our docker data to the GlusterFS volume we can also delete the temporary mount point now (**run on all nodes**):

```
rm -R /mnt/tempdockerfiles/
```

Now we mount the GlusterFS volume in the old mount point of our prior NFS storage (**run on all nodes**):

```
sudo mount.glusterfs localhost:/staging-gfs /mnt/docker
```

Verify with `ls -l /mnt/docker` and you should see all your old docker container data again. last but not least we need to set the permissions right on the volume and make the mount persistent over a reboot (or other circumstances):

Set permissions (**on all nodes**):

```
### replace ubuntu with your username
sudo chown -R ubuntu:ubuntu /mnt/docker/
sudo chown root:docker /mnt/docker/
```

Add mount point to /etc/fstab (**on all nodes**):

```
echo 'localhost:/staging-gfs /mnt/docker glusterfs defaults,_netdev,backupvolfile-
server=localhost 0 0' >> /etc/fstab
```

Important: If you had an old entry in /etc/fstab from auto mounting your NFS shared volume on boot to /mnt/docker, you need to remove that line from fstab now as well. (on all nodes)

Now reboot ALL NODES with:

```
sudo reboot now
```

After the reboot you can verify if the GlusterFS volume is mounted correctly on your nodes with:

```
df -h
```

Great! Thats it, you successfully installed GlusterFS on your Docker Swarm cluster and migrated all your container data from the NFS shared volume over the the Gluster volume!

Appendix: Complete uninstall glusterfs

If you at any point need to fully uninstall and remove glusterfs from all your nodes, you should to the following on all nodes:

```
#enter root
sudo -s

gluster volume stop staging-gfs
gluster volume delete staging-gfs

gluster peer detach 192.168.0.4
gluster peer detach 192.168.0.3
gluster peer detach 192.168.0.2
gluster peer detach 192.168.0.1

apt --purge remove glusterfs-server -y
apt autoremove -y

rm -rf /gluster/
rm -rf /var/log/glusterfs/
rm -rf /var/lib/glusterd/
rm -rf /run/gluster/
rm -rf /usr/lib/python3/dist-packages/gluster/



#Remove nano /etc/fstab boot mount entry for gluster:
nano /etc/fstab

#Remove all firewall rules as above:
ufw delete <position count of your rule in ufw status>
```

After that you should restart each node and you are back to a state where you could reinstall glusterfs fresh.

Published by Florian Müller in Development, Technology

Tags: docker, docker swarm, docker-compose, filesystem, gluster, glusterfs, ssh, terminal, ubuntu

---

Revision #1
Created 2024-07-08 02:34:38 UTC by Administrador
Updated 2024-07-08 02:36:30 UTC by Administrador