How to Increase Your Storage Speed and Avail-ability with GlusterFS

Introduction and use cases

GlusterFS is a clustered file system designed to increase the speed, redundancy, and availability of network storage. When configured correctly with several machines, it can greatly decrease downtime due to maintenance and failures.

Gluster has a variety of use cases, with most configurations being small three server clusters. I’ve personally used Gluster for VM storage in Proxmox, and as a highly available SMB file server setup for Windows clients.

Configurations and requirements

For the purposes of this demonstration, I'll be using GlusterFS along with Proxmox. The two work very well with each other when set up in a cluster.

Before using GlusterFS, you'll need at least three physical servers, or three virtual machines, which I also recommend be on separate servers. This is the minimal configuration to set up high availability storage. Each server will have a minimum of two drives, one drive will be used for the OS, the other will be used for Gluster.

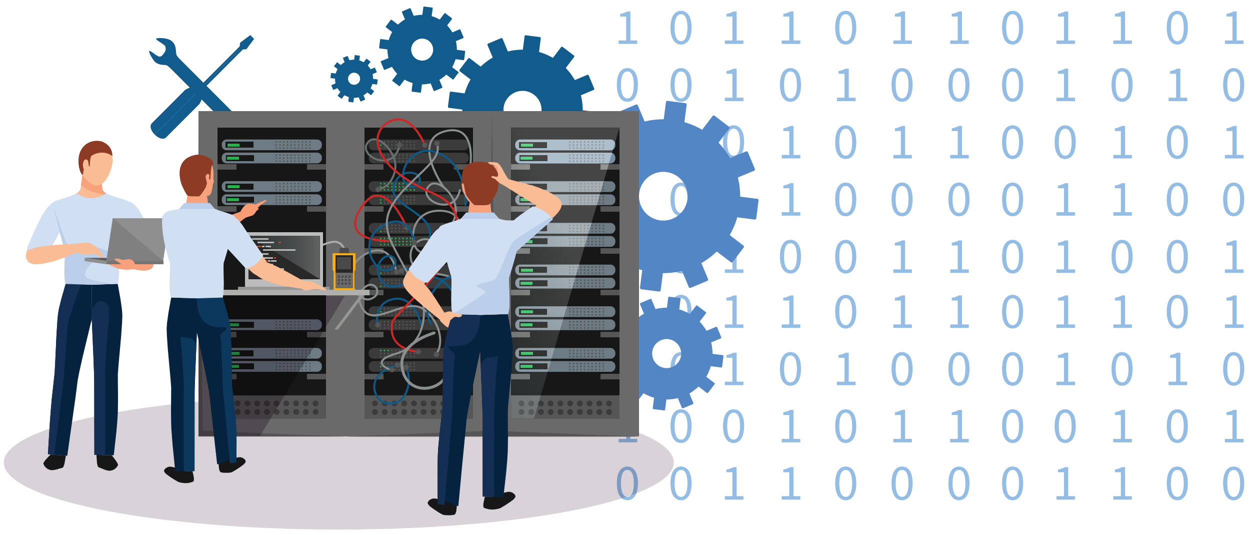
Gluster operates on a quorum based system in order to maintain consistency across the cluster. In a three server scenario, at least two of the three servers must be online in order to allow writes to the cluster. Two node clusters are possible, but not recommended. With two nodes, the cluster risks a scenario known as split-brain, where the data on the two nodes isn't the same. This type of inconsistency can cause major issues on production storage.

For demonstration purposes, I'll be using 3 CentOS 7 virtual machines on a single Proxmox server.

There are two ways we can go about high availability and redundancy, one of which saves more space than the other.

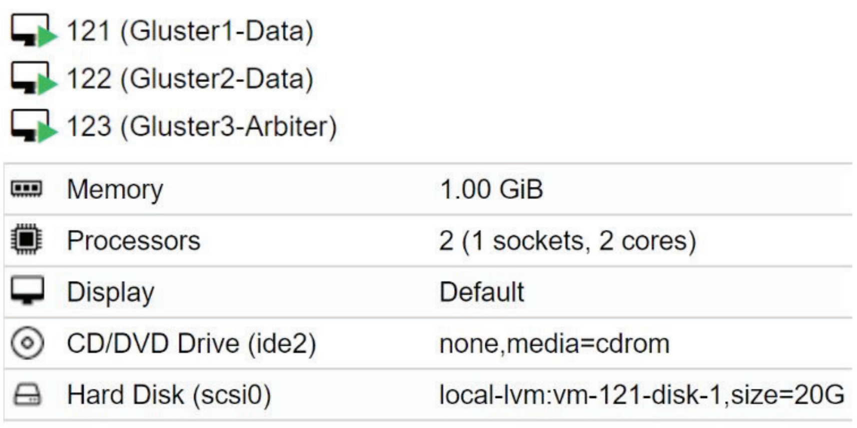
1. The first way is to set up gluster to simply replicate all the data across the three nodes. This configuration provides the highest availability of the data and maintains a three-node quorum, but also uses the most amount of space.

2. The second way is similar, but takes up ⅓ less space. This method involves making the third node in the cluster into what's called an arbiter node. The first two nodes will hold and replicate data. The third node will only hold the metadata of the data on the first two nodes. This way a three-node quorum is still maintained, but much less storage space is used.The only downside is that your data only exists on two nodes instead of three. In this demo I'll be using the latter configuration, as there are a few extra steps to configuring it correctly.



GlusterFS Configuration Tutorial

Start by setting up three physical servers or virtual machines with CentOS 7. In my case, I set up three virtual machines with 2 CPU cores, 1GB RAM, and 20GB OS disks. Through the guide, I'll specify what should be done on all nodes, or one specific node. All three machines should be on the same subnet/broadcast domain. After installing CentOS 7 on all three nodes, my IP configurations are as follows:


121 (Gluster1-Data)
122 (Gluster2-Data)
123 (Gluster3-Arbiter)

Memory	1.00 GiB
Processors	2 (1 sockets, 2 cores)
Display	Default
CD/DVD Drive (ide2)	none,media=cdrom
Hard Disk (scsi0)	local-lvm:vm-121-disk-1,size=20G

Gluster1: 10.255.255.21

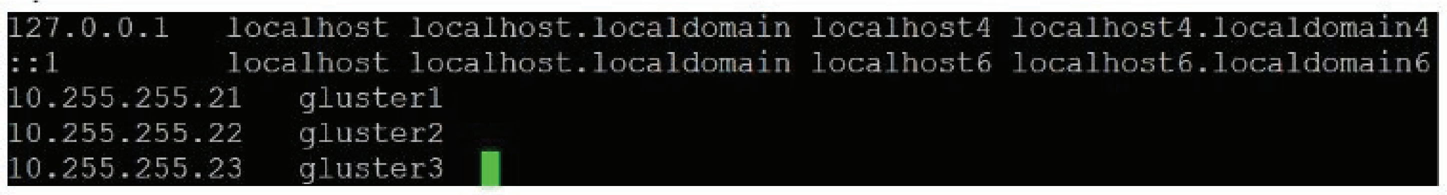
Gluster1: 10.255.255.22

Gluster1: 10.255.255.23

All Nodes:

The first thing we'll do after the install is edit the /etc/hosts file. We want to add the the hostname of each node along with their IPs into the file, this prevents Gluster from having issues in the case that a DNS server isn't reachable.

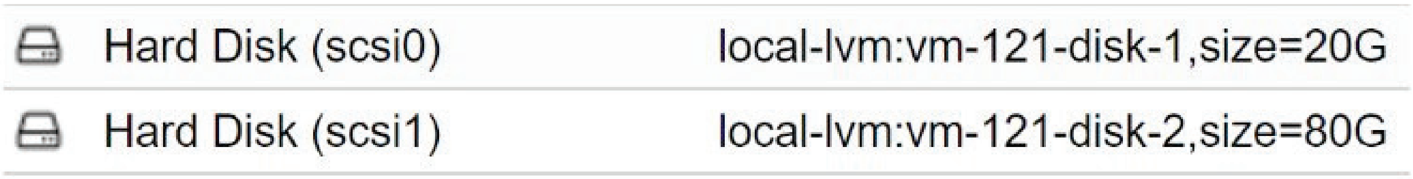
My hosts file on each node is as follows:


```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
10.255.255.21   gluster1
10.255.255.22   gluster2
10.255.255.23   gluster3
```

All Nodes:

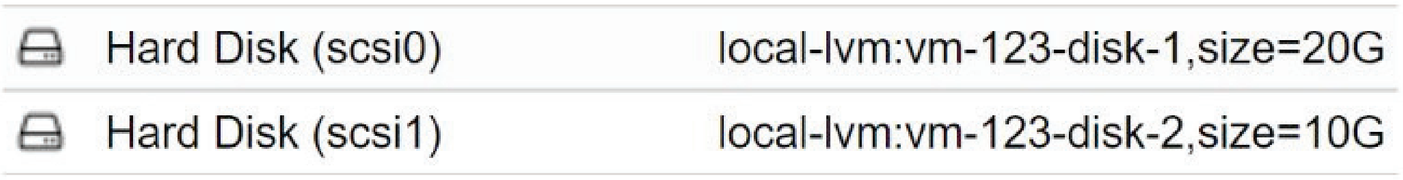
After configuring the hosts file, add the secondary disks to the hosts. I added an 80GB disk to Gluster1 and Gluster2, and a 10GB disk to Gluster3, which will be the arbiter node. If the Gluster nodes are VMs, the disks can simply be added live without shutting down

Gluster 1&2 Drive configuration:



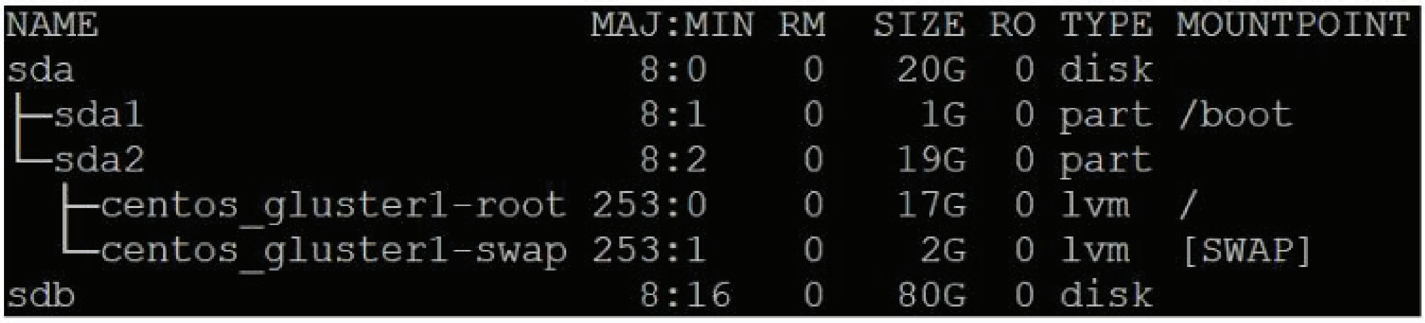
Hard Disk (scsi0)	local-lvm:vm-121-disk-1,size=20G
Hard Disk (scsi1)	local-lvm:vm-121-disk-2,size=80G

Gluster 3 Drive configuration:



Hard Disk (scsi0)	local-lvm:vm-123-disk-1,size=20G
Hard Disk (scsi1)	local-lvm:vm-123-disk-2,size=10G

After adding the disks, run lsblk to ensure they show up on each node:

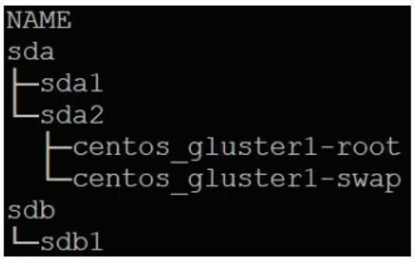


```
NAME                       MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                          8:0    0   20G  0 disk
├─sda1                       8:1    0    1G  0 part /boot
└─sda2                       8:2    0   19G  0 part
  ├─centos_gluster1-root 253:0    0   17G  0 lvm  /
  └─centos_gluster1-swap 253:1    0    2G  0 lvm  [SWAP]
sdb                          8:16   0   80G  0 disk
```

sda is the OS disk, sdb is the newly added storage disk. We'll want to format and mount the new storage disk for use. In the case of this demo, I'll be using xfs for the storage drive.

```
fdisk /dev/sdb
n
p
enter
enter
w
enter
mkfs.xfs /dev/sdb1
```

You should now see sdb1 when you run an lsblk:



```
NAME
sda
├─sda1
└─sda2
  ├─centos_gluster1-root
  └─centos_gluster1-swap
sdb
└─sdb1
```

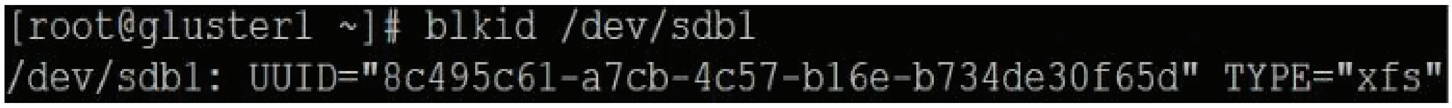
We'll now create a mount point and add the drive into /etc/fstab in order for it to mount on boot:

My mountpoint will be named brick1, I'll explain bricks in more detail after we mount the drive.

```
mkdir -p /data/brick1
```

After creating the mountpoint directory, we'll need to pull the UUID of the drive, you can do this with blkid

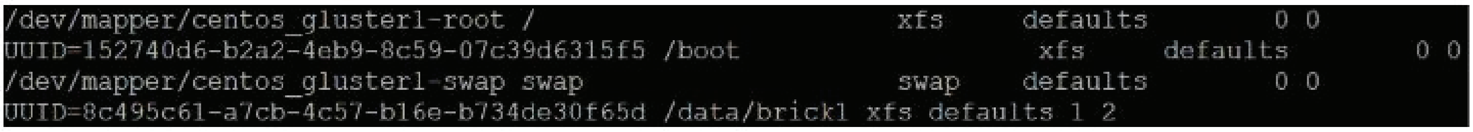
```
blkid /dev/sdb1
```



```
[root@gluster1 ~]# blkid /dev/sdb1
/dev/sdb1: UUID="8c495c61-a7cb-4c57-b16e-b734de30f65d" TYPE="xfs"
```

Copy down the long UUID string, then go into /etc/fstab and add a similar line:

```
UUID=<UUID without quotes> /data/brick1 xfs defaults 1 2
```



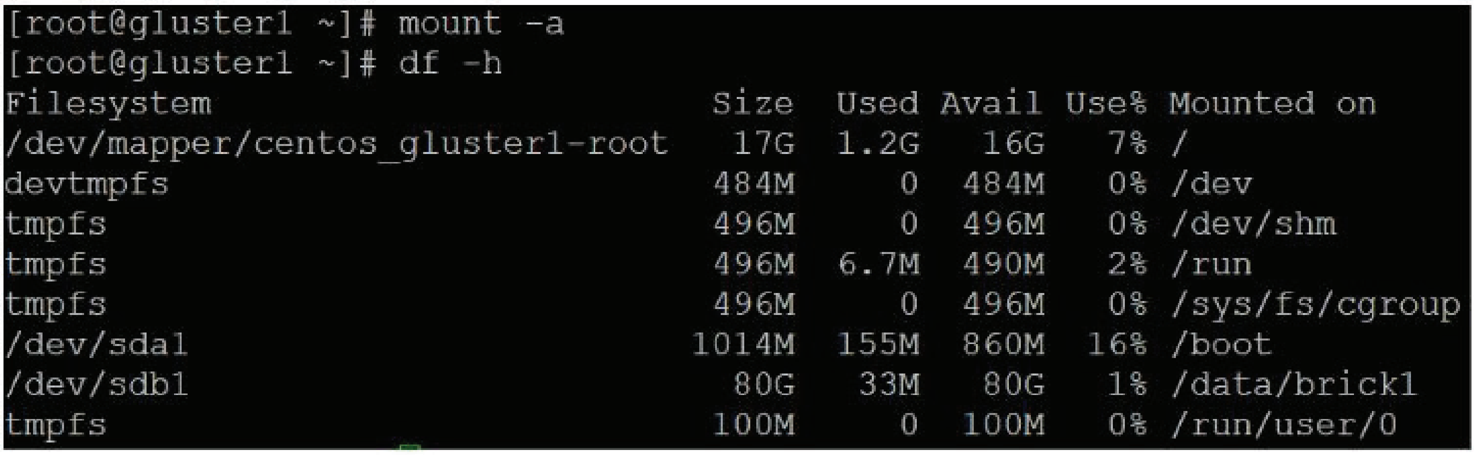
```
/dev/mapper/centos_gluster1-root /                       xfs     defaults        0 0
UUID=152740d6-b2a2-4eb9-8c59-07c39d6315f5 /boot                   xfs     defaults        0 0
/dev/mapper/centos_gluster1-swap swap                    swap    defaults        0 0
UUID=8c495c61-a7cb-4c57-b16e-b734de30f65d /data/brick1 xfs defaults 1 2
```

Save the file, then run mount -a

Then run df -h

You should now see /dev/sdb1 mounted on /data/brick1

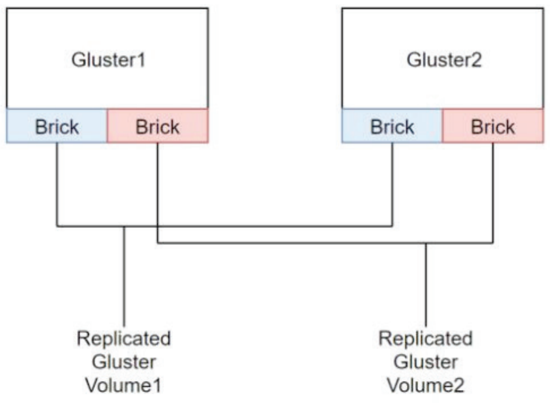
Make sure you format and mount the storage drives on each of the three nodes.



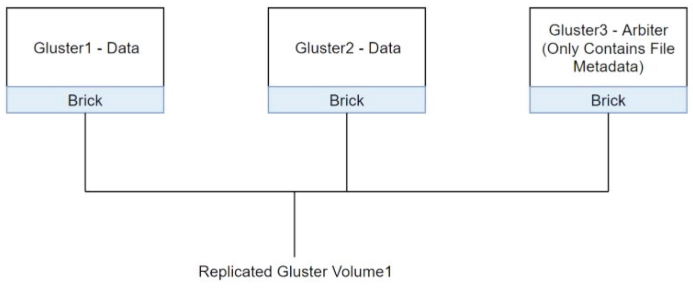
```
[root@gluster1 ~]# mount -a
[root@gluster1 ~]# df -h
Filesystem                        Size  Used Avail Use% Mounted on
/dev/mapper/centos_gluster1-root   17G  1.2G   16G   7% /
devtmpfs                          484M     0  484M   0% /dev
tmpfs                             496M     0  496M   0% /dev/shm
tmpfs                             496M  6.7M  490M   2% /run
tmpfs                             496M     0  496M   0% /sys/fs/cgroup
/dev/sda1                        1014M  155M  860M  16% /boot
/dev/sdb1                          80G   33M   80G   1% /data/brick1
tmpfs                             100M     0  100M   0% /run/user/0
```

Gluster volumes are made up of what what are called bricks. These bricks can be treated almost like virtual hard drives in a what we'd use for a RAID array.

This depiction gives an idea of what a two server cluster with two replicated Gluster volumes would look like:



This is what the Gluster volume we're creating will look closer to:


Gluster1 - Data
Brick
Gluster2 - Data
Brick
Gluster3 - Arbiter (Only Contains File Metadata)
Brick
Replicated Gluster Volume1

Now it's time to install and enable GlusterFS, run the following on all three nodes:

```
yum install centos-release-gluster

yum install glusterfs-server -y

systemctl enable glusterd
```

Gluster doesn't play well with selinux and the firewall, we'll disable the two for now. Since connecting to services such as NFS and Gluster doesn't require authentication, the cluster should be on a secure internal network in the first place.

Open up /etc/selinux/config with a text editor and change the following:

```
SELINUX=enforcing
```
to
```
SELINUX=disabled
```

Save and exit the file, then disable the firewall service:

```
systemctl disable firewalld.service
```

At this point, reboot your nodes in order for the SELINUX config change to take effect.

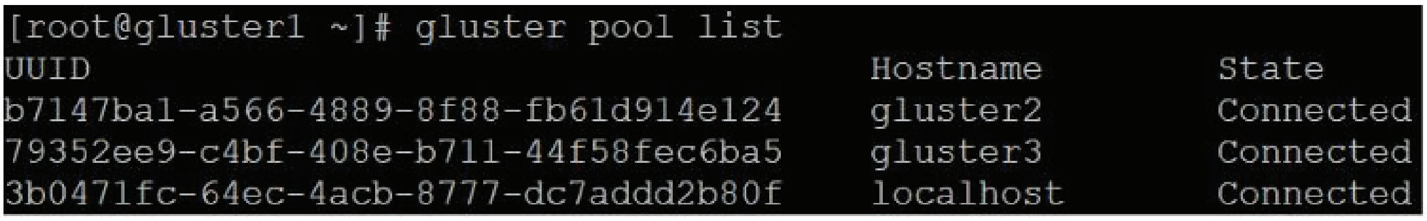
On Node Gluster1:

Now it's time to link all the Gluster nodes together, from the first node, run the following:

```
gluster peer probe gluster2

gluster peer probe gluster3
```

Remember to run the above commands using the hostnames of the other nodes, not the IPs.

Now let's check and see if the nodes have successfully connected to each other:



```
[root@gluster1 ~]# gluster pool list
UUID                                    Hostname        State
b7147ba1-a566-4889-8f88-fb61d914e124    gluster2        Connected
79352ee9-c4bf-408e-b711-44f58fec6ba5    gluster3        Connected
3b0471fc-64ec-4acb-8777-dc7addd2b80f    localhost       Connected
```

We can see that all of our nodes are communicating without issue.

Finally, we can create the replicated gluster volume, it's a long command, ensure there aren't any errors:

```
gluster volume create stor1 replica 3 arbiter 1 gluster1:/data/brick1/stor1
gluster2:/data/brick1/stor1 gluster3:/data/brick1/stor1
```

- "stor1" is the name of the replicated volume we're creating
- "replica 3 arbiter 1" specifies that we wish to create a three node cluster with a single arbiter node, the last node specified in the command will become the arbiter
- "gluster1:/data/brick1/stor1" creates the brick on the mountpoint we created earlier, I've named the bricks stor1 in order to reflect the name of the volume, but this isn't imperative.

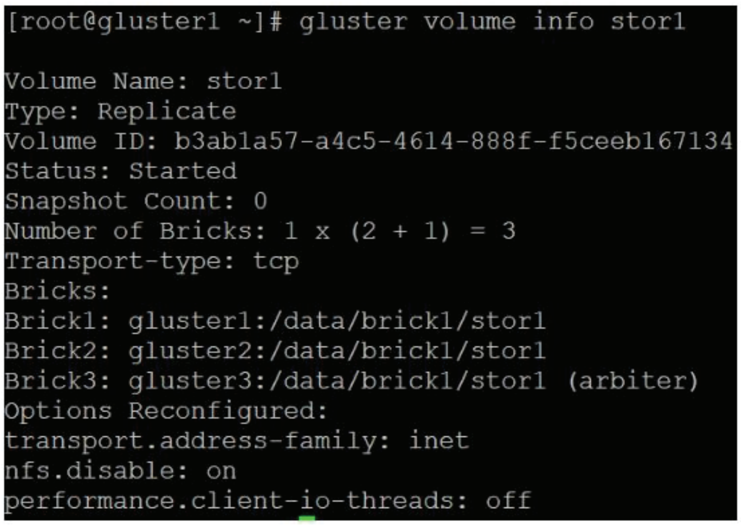
After running the command, you'll need to start the volume, do this from node 1:

```
gluster volume start stor1
```

Then check the status and information:

```
gluster volume start stor1
```

gluster volume info stor1:



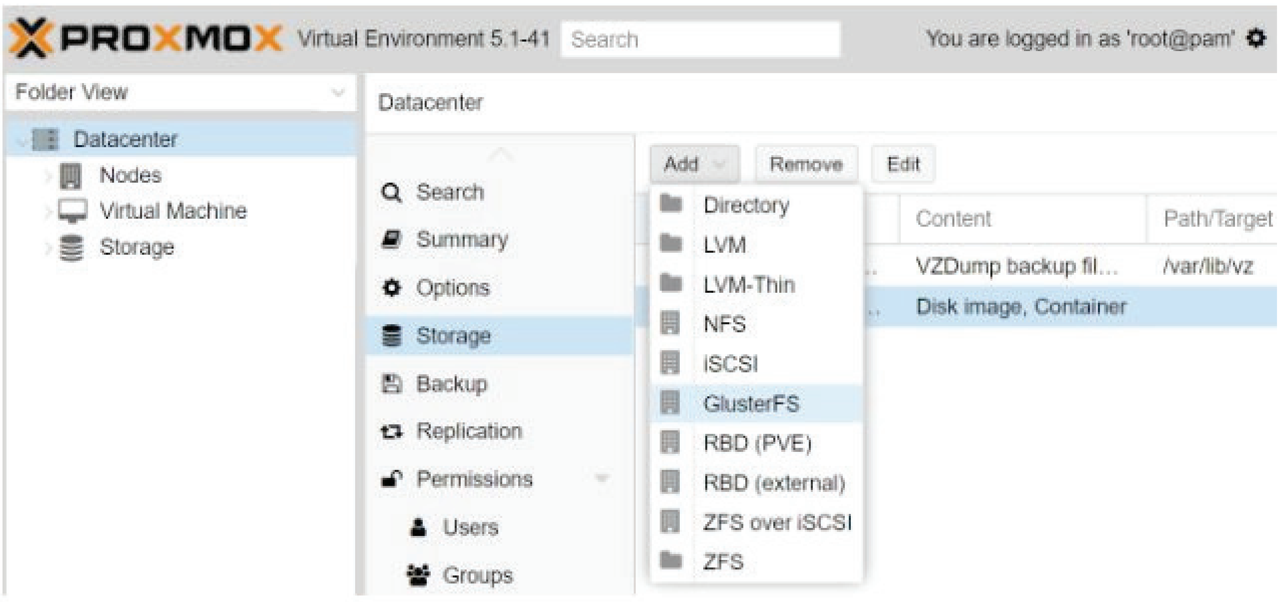
```
[root@gluster1 ~]# gluster volume info stor1

Volume Name: stor1
Type: Replicate
Volume ID: b3ab1a57-a4c5-4614-888f-f5ceeb167134
Status: Started
Snapshot Count: 0
Number of Bricks: 1 x (2 + 1) = 3
Transport-type: tcp
Bricks:
Brick1: gluster1:/data/brick1/stor1
Brick2: gluster2:/data/brick1/stor1
Brick3: gluster3:/data/brick1/stor1 (arbiter)
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
```

As you can see, the brick created on the third node is specified as an arbiter, and the other two nodes hold the actual data. At this point, your are ready to connect to GlusterFS from a client device.

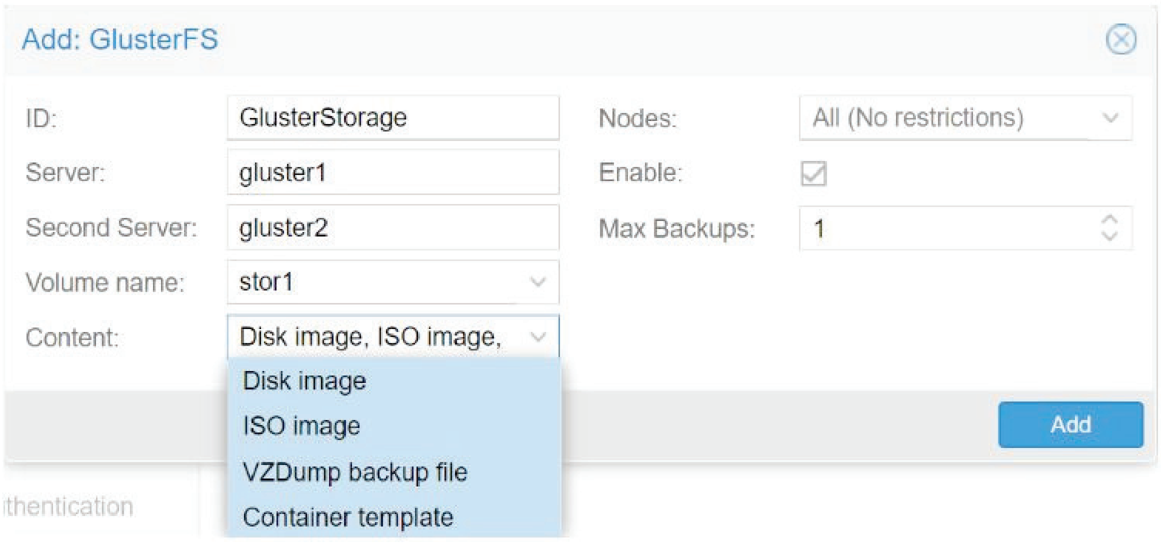
I'll demonstrate this connection from Proxmox, as the two naturally work very well together.

Ensure that the Gluster node hostnames are in your Proxmox /etc/hosts file or available through a DNS server prior to starting.

Start by logging into the Proxmox web gui, then go to Datacenter>Storage>Add>GlusterFS:



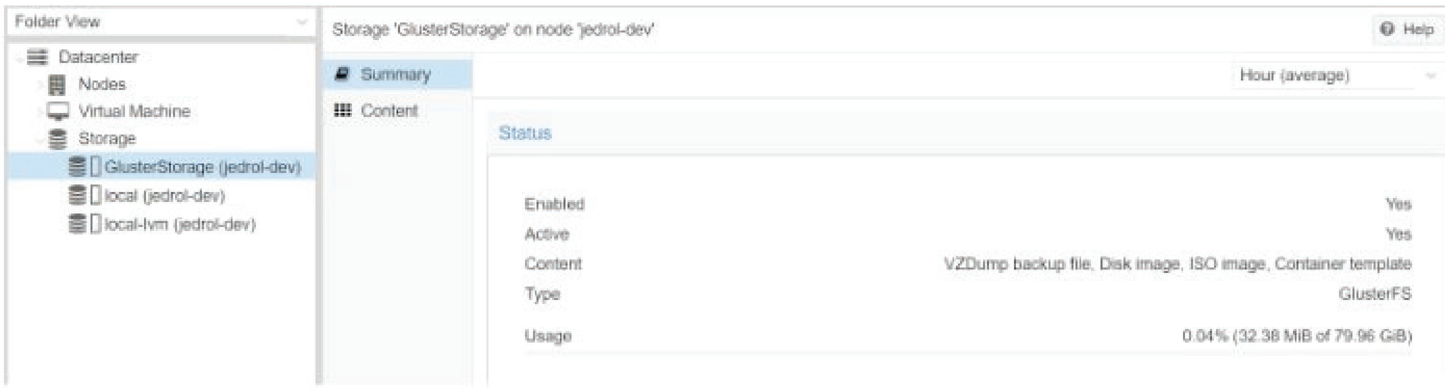
Then, input a storage ID, this can be any name, the first two node hostnames, and the name of the Gluster volume to be used. You can also specify what file types you wish to store on the Gluster volume:


Add: GlusterFS

ID:	GlusterStorage	Nodes:	All (No restrictions)
Server:	gluster1	Enable:	✓
Second Server:	gluster2	Max Backups:	1
Volume name:	stor1		
Content:	Disk image, ISO image,		

Don't worry about the fact that you weren't able to add the third node into the Proxmox menu, Gluster will automatically discover the rest of the nodes after it connects to one of them.

Click add, and Proxmox should automatically mount the new Gluster volume:



Storage 'GlusterStorage' on node 'jedrol-dev'

Enabled	Yes
Active	Yes
Content	VZDump backup file, Disk image, ISO image, Container template
Type	GlusterFS
Usage	0.04% (32.38 MiB of 79.96 GiB)

As we can see, we have a total of 80GB of space, which is now redundantly replicated. This newly added storage can now be used to store VM virtual hard disks, ISO images, backups, and more.

In a larger application scenarios, GlusterFS is able to increase the speed and resiliency of your storage network. Physical Gluster nodes combined with enterprise SSDs and 10/40G networking can make for an extremely high end storage cluster.

Similarly GigeNET uses enterprise SSDs with a 40 Gbe private network for our storage offerings.

Already have enough on your plate? Explore GigeNET's managed services.

Call us today or Chat with a specialist