Introduction

As you may already know, GlusterFS provides several methods for storage access from clients. However, only the native FUSE GlusterFS client has built-in failover and high availability features.

A single NFS server would be handling all requests, distributing them to the different bricks as needed. This would create a bottleneck and a single point of failure (the NFS server itself), and both are not desirable in a scalable storage environment. There are ways around this using CTDB.

Load Balancing and High availability using CTDB + DNS round robin

We will be taking advantage of two different features to provide a highly available, scalable NFS and CIFS service. First, we will use DNS round robin to have each client use one of the Gluster servers for their mounts. Then, CTDB will provide virtual IPs and failover mechanisms to ensure that, in the case of a server failure, failover is transparent to clients. Let's assume the following Gluster server setup with four nodes:

- gluster1, with IP 192.168.122.100
- gluster2, with IP 192.168.122.101
- gluster3, with IP 192.168.122.102
- gluster4, with IP 192.168.122.103

We will now assign four virtual IPs for the servers:

- 192.168.122.200
- 192.168.122.201
- 192.168.122.202
- 192.168.122.203

And then, define DNS entries for two load balanced services, called glusternfs and glustercifs. These services will be defined in the DNS zone with the following fragment:

```
; zone file fragment
glusternfs         1         IN A          192.168.122.200
glusternfs         1         IN A          192.168.122.201
glusternfs         1         IN A          192.168.122.202
glusternfs         1         IN A          192.168.122.203
glustercifs         1         IN A          192.168.122.200
glustercifs         1         IN A          192.168.122.201
glustercifs         1         IN A          192.168.122.202
glustercifs          1          IN A          192.168.122.203
```

You will notice that we are using the virtual IPs here. When combined with CTDB IP failover, this allows us to have both load balancing and high availability.

We are setting a low TTL for the records so, if a virtual IP is down while a client is trying to mount, the client can retry using a different one.

With this architecture, load is properly balanced among the NFS / CIFS servers, provided that we have a similar access pattern from all clients. So now, let's move on to the actual CTDB implementation.

CTDB configuration

> **NOTE:** If you try this with GlusterFS 3.3.x, you **must disable SELinux** manually, editing /etc/sysconfig/selinux.

We will be creating a CTDB configuration for the architecture illustrated above. We start with a single volume called **vol1**, configured as distributed+replicated (2 replicas), and we are going to export it using NFS and CIFS. Keep in mind that this is not a good idea for a production environment, as the caching mechanisms of NFS and CIFS do not seem to behave well with each other when there are concurrent access to the same file from both protocols. This is not a RHS 2.0 limitation.

First, mount the GlusterFS volume locally on each RHS server:

```
# mount -t glusterfs gluster1:/vol1 /mnt/glustervol
```

Each server will use its local hostname to mount. Of course, you will need to add this mount to /etc/fstab if you want it to persist across reboots.

Now, we will create the Samba base configuration. Edit /etc/samba/smb.conf and add the following lines to the [global] section:

```
     clustering = yes
     idmap backend = tdb2
     private dir = /mnt/glustervol/lock
```

Then, add the CIFS export section, at the end of the document:

```
[glustertest]
     comment = For testing a Gluster volume exported through CIFS
     path = /mnt/glustervol
     read only = no
     guest ok = yes
     valid users = jpena
```

The CIFS export is only allowing user jpena to access. Depending on your configuration, you will want to use Active Directory, and then you will need to add the required information to the [global] section. In this example, we will use a simpler security model.

The first three lines instruct Samba to use CTDB for clustering. We are defining a private directory (/mnt/glustervol/lock), where the CTDB lock file will be stored. This must be a shared file system, and using GlusterFS for that seems to be the most natural choice 🙂. From one of the GlusterFS servers, we will create this directory and edit a few files there:

/mnt/glustervol/lock/ctdb

```
CTDB_RECOVERY_LOCK=/mnt/glustervol/lock/lockfile
#CIFS only
CTDB_PUBLIC_ADDRESSES=/etc/ctdb/public_addresses
CTDB_MANAGES_SAMBA=yes #CIFS only
CTDB_NODES=/etc/ctdb/nodes
```

/mnt/glustervol/lock/nodes

This file will contain the IP addresses of the RHS servers.

```
192.168.122.100
192.168.122.101
192.168.122.102
192.168.122.103
```

/etc/ctdb/public_addresses

This file will be created on each node, containing the virtual IP addresses. Since it is defined locally, we can create different groups of servers, each sharing a pool of virtual IP addresses, if needed. In this case, all nodes will have the same contents.

```
192.168.122.200/24 eth0
192.168.122.201/24 eth0
```

```
192.168.122.202/24 eth0
192.168.122.203/24 eth0
```

The first two files could also be created locally on each node, but it's easier just to have them shared. We will know create some symbolic links:

```
# ln -s /mnt/glustervol/lock/ctdb /etc/sysconfig/ctdb
# ln -s /mnt/glustervol/lock/nodes /etc/ctdb/nodes
```

We are now done with the configuration. Since the ctdb service starts the samba service on its own, we need to disable the samba service startup:

```
# chkconfig smbd off
# chkconfig ctdb on
# service ctdb start
```

Testing and troubleshooting

Once the ctdb service is started, it will take a while until the cluster is formed and ready for client access. You can monitor the status using the following commands:

```
# ctdb status
```

It will show the status for all cluster nodes. If a node is marked as OK, it is working fine. Otherwise, you will have to wait for a while

```
# ctdb ip
```

It will show which cluster node is using one of the virtual IP addresses. Beware: **ifconfig** will not show the virtual IP, but **ip a** will.

The CTDB log file is located at /var/log/log.ctdb.

Once the CTDB service is ready, we can test the configuration. For this use case, we will first add user jpena to the nodes

```
# adduser <username>
```

and to the samba configuration on all nodes

```
# smbpasswd -a <username>
```

Then, it is just a matter of mounting the exported file systems using NFS and CIFS from a client:

```
# sudo mount -t cifs -o user=<username> //glustercifs/glustervol /mnt/cifs
# sudo mount glusternfs:/vol1 /mnt/nfs
```

It is now possible to test failover. Find the node serving NFS/CIFS to your client, and turn it off. In some cases, it will take some time for the failover to happen, and you can check the process with the previously mentioned commands.