NFSs said to be dead live longer.

At this point it should be shown that it is also possible to set up a highly available NFS server. Together with GlusterFS, CTDB and bind9, a highly available NFS environment can be set up with the NFS kernel server. Together with DNS-round-robin, which is performed by Bind9, DNS load balancing can then also be used.

Although the main focus is on NFS, a short section nevertheless addresses the setup of GlusterFS.

Three systems are needed to set up the environment. Two for the servers and one system as a client for testing. Although it definitely makes sense to set up at least three servers for the gluster cluster in a production environment, the reason for this is the quorum, a cluster with two nodes is sufficient to test the function of the NFS server.

On www.gluster.org the current version is always provided. For almost all common distributions there are the packages. Here in the article Debian 12 with Gluster 10 and CTDB from the Samba packages of version 4.17 will be used. With the exception of the gluster packages all packages come directly from the Debian 12 repositories.

The NFS server setup describes how to set up both NFSv3 and NFSv4.

IP addresses of the system:

Hostname productive network	IP productive network	Hostname Heartbeat Network	IP Heartbeat Network
nfs01.example.net	192.168.56.201/24 Device: enp0s8	c01.gluster	192.168.57.201/24 Device: enp0s9
nfs02.example.net	192.168.56.202/24 Device: enp0s8	c02.gluster	192.168.57.202/24 Device: enp0s9

All communication between the gluster nodes takes place via the "heartbeat network". Subsequent access always goes via the production network.

GlusterFS setup

After setting up the repository, the package "glusterfs-server" is installed. For the gluster volume a separate partition is needed in any case which is formatted with "xfs". Xfs is the recommendation of gluster.
The partition is mounted in a separate mount point "/gluster". Then the subdirectory "/gluster/brick" is created in the directory. This will be the directory where the data of the node will be stored.
Before setup, the "glusterd" service must be enabled and started on both nodes. This is done with the command "systemctl enable --now glusterd".

After the partition is set up on both servers and the service is started, the two nodes can be made known to each other. Listing-01 shows the process:

----------Listing-01----------

```
root@nfs01:~# gluster peer probe c02
peer probe: success
```

```
root@nfs01:~# gluster peer status
Number of Peers: 1

Hostname: c02
Uuid: 5125ed73-6b0e-45ca-b0df-2b922b1d7e86
State: Peer in Cluster (Connected)
```

The mutual announcement of the nodes needs to be executed only on one node. As a result, both nodes now belong to one pool.

After setting up the pool, the gluster volume can now be created. Listing-02 shows the setup with the corresponding command:

----------Listing-02----------

```
root@nfs01:~# gluster volume create gv0 replica 2 c01:/gluster/brick
c02:/gluster/brick

 Replica 2 volumes are prone to split-brain. Use Arbiter or Replica 3
to avoid this. See: http://docs.gluster.org/en/latest/Administrator-
Guide/Split-brain-and-ways-to-deal-with-it/.
 Do you still want to continue?
 (y/n) y

volume create: gv0: success: please start the volume to access data
```

The note only says that a volume consisting of two nodes cannot quorum, it is not an error message here. For testing purposes, a volume can also consist of two nodes.

After creating the volume with the name "gv0" it is still necessary to start the volume with the command "gluster v start gv0". Both commands only need to be executed on one node.

Now the volume can be mounted and used.

!ATTENTION!
Under no circumstances should data be edited directly in the "/gluster/brick" directory. This can destroy the volume and result in data loss. Access should only ever take place via the mounted volume.

If the NFS server is to be set up on other systems, which always makes sense in a production environment, the volume can be mounted there with the "glustertfs-client".

The easiest way to start the volume is via a sysdtemd script. The volume is then mounted in the previously created mount point "/glustertf". Listing-03 shows the script:

----------Listing-03---------

```
[Unit]
Description = Data dir
After=network.target glusterfs-server.service
Requires=network-online
```target

```
[Mount]
What=c01:/gv0
 Where=/glusterfs
Type=glusterfs
Options=defaults,_netdev,negative-timeout=10,attribute-
timeout=30,fopen-keep-cache,direct-io-mode=enable,acl

[Install]
WantedBy=multi-user.target
-------------------------------
```

The script is stored in the "/etc/systemd/system" directory with the name "glustertf.mount". The name of the script corresponds to the path to the mount point. If the directory is not in the "root", the path must be separated by a dot in the filename.

The line:
`What=c01:/gv0`
Specifies the server from which the list of all gluster nodes is loaded. If a node fails, it will automatically connect to another node in the list.

The script is now activated and executed on both servers with the command "systemctl enable --now glusterfs.mount".

Now the gluster volume is mounted and can be used. Data written to the volume on one of the nodes is also visible on the other node.

Setting up CTDB
CTDB from the Samba packages is used for starting the NFS services and for failover if an NFS server in the cluster fails. CTDB assigns the NFS servers a dynamic IP address from the production network, via which the NFS server can be reached later. If one of the servers fails, CTDB swaps the address of the failed node to one of the still running nodes. This way, a client can immediately reconnect to the NFS server using the same IP address.

CTDB also has the task of file locking. Without file locking, two clients could open and write to the same file, at the same time, on different nodes. This would inevitably lead to data loss.

As mentioned at the beginning, CTDB is a part of the Samba environment, but can also be used for other services. Since no Samba server is to be set up here, only the "ctdb ethtool tdb-tools iptables" packages are required on all nodes.

Ethtool is used for dynamic assignment of CTDB's IP addresses.

After installation, the entire configuration of CTDB is managed in the "/etc/ctdb" directory. All following adjustments must always be performed on all nodes.

In order for CTDB to perform file locking, the path to a lock file in the gluster volume is required. This path with the file name is entered in the file "/etc/ctdb/ctdb.conf". The line from Listing-04 is entered in the "[cluster]" section for this purpose:
--------Listing-04---------
cluster lock = /glusterfs/lock.ctdb

In order for all nodes of the CTDB cluster to know each other, a file is required in which all IP addresses of the nodes are listed. These IP addresses are then used exclusively for communication between the CTDB nodes. No accesses from the production network will take place via these IP addresses. The IP addresses in the example are the IP addresses from the heartbeat network of the nodes. The IP addresses are stored on all CTDB nodes in the file "/etc/ctdb/nodes". Listing-05 shows the file:
--------Listing-05-------
```
192.168.57.201
192.168.57.202
```

The only thing missing are the dynamic IP addresses for the provision of the NFS server in the production network. The required IP addresses are entered in the file "/etc/ctdb/public_addresses" as in Listing-06:
--------Listing-06------
```
192.168.56.211/24 enp0s8
192.168.56.212/24 enp0s8
```

The CTDB service is automatically active on Debian systems, the service now only needs to be restarted on all nodes with the command "systemctl restart ctdb.service".

Notice:
The startup process can be followed on a second console with "journalctl -f".

After restarting the service, the congestion of the CTDB can be checked with the command "ctdb status". Only when all nodes there have reached the status "OK", as in Listing-07, can the setup be continued.
---------Listing-07------
```
root@nfs02:/etc/ctdb# ctdb status
Number of nodes:2
pnn:0 192.168.57.201 OK
pnn:1 192.168.57.202 OK (THIS NODE)
Generation:746838290
Size:2
hash:0 lmaster
hash:1 lmaster
Recovery mode:NORMAL (0)
```

The "ip a" command now shows that all nodes have received one of the dynamic IP addresses in the production network. If the service is stopped on one of the nodes, the dynamic IP address is

immediately transferred to another node. If the service is restarted, the node receives one of the IP addresses again.

In order to be able to use failover and load balancing via DNS, it is now still important that all dynamic IP addresses are entered under one name in the DNS. If the name is queried, all IP addresses are then always displayed. The client then always selects the first IP address from the list. Listing-07 shows the name resolution:

---------Listing-07--------

```
cluster01.example.net has address 192.168.56.212
cluster01.example.net has address 192.168.56.211

root@nfs02:/etc/ctdb# host cluster01
cluster01.example.net has address 192.168.56.211
cluster01.example.net has address 192.168.56.212
```

A new query, a few seconds later, shows the IP addresses in a different order. This is how DNS load balancing is then performed.

NFSv3 setup

Up to this point, it was only a matter of preparing for the NFS server. Only when all the previous steps have been performed can the setup of NFS begin.

NFSv3 should make the beginning. There is a problem with NFSv3, all ports used by the client to communicate with the NFS server are assigned dynamically by the portmapper when the service is started. If no adjustments are made at this point, each of the NFSv3 servers would almost always have different ports that the client accesses. As long as there is only one NFSv3 server in the network this is not a problem. However, for an NFSv3 server that is to be provided via a cluster, it is mandatory that all NFSv3 accounts always use the identical ports.

For the NFS server the packages "nfs-kernel-server" and "quota" are needed. The "quota" package must be installed even if no file system quotas are to be set up, because CTDB checks whether the service is available.

The entire configuration of the NFS server is done in the file "/etc/nfsd.conf". It is also specified there which service should use which port. This is the only way to run NFSv3 in the cluster.

Listing-08 shows all required changes in the file:

----------Listing-08---------

```
[nfsrahead]
[exports]
[exportfs]
[gssd]
[lockd]
port=32803
udp-port=32769
[exportd]
[mountd]
manage-gids=y
```

port=892
[nfsdcld]
[nfsdcltrack]
[nfsd]
[statd]
port=662
name=cluster01.example.net
[sm-notify]
[svcgssd]

It is important here that the configuration is identical on all nodes and that the fqdn of the cluster is entered in the variable "name" and not the name of each individual node.

The only thing missing is the definition of the port for the "quotad". The port must be specified even if quotas are not set up. CTDB searches for the ports. Listing-09 shows the changes in the file "/etc/default/quota".
---------Listing-09-------
```
RPCRQUOTADOPTS="-p 875"
```

After all settings have been made, the NFS share can now be entered in the file "/etc/exports".

Gasnz important is: Each share needs the parameter "fsid=" the parameter must be identical for each share on each NFS server. Where each share needs its own "fsid".

Without cluster operation, this parameter is only required for NFSv4. In cluster mode, the share is unique for NFSv3.

In the future not systemd should start the NFS server, but CTDB. Therefore it is necessary to deactivate the service "nfs-kernel-server" for systemd. The same applies to the "rpcbind.service" and "rpcbind.socket". Listing-10 shows the command to stop and disable the services at the same time :
---------Listing-10-------
```
systemctl disable --now nfs-kernel-server rpcbind.service
rpcbind.socket
```

Since CTDB always wants to activate the quotas, even if they should not be used at all, the line from Listing-11 must be added to the file "/etc/default/quota":
-------Listing-11--------
```
need_rquotad=1
```

Up to this point, the CTDB is running and the IP addresses are already distributed and also pivoted if one of the nodes fails, but the services are not yet started.

Unfortunately this is not all that needs to be done when no quotas are set up. There is still a test script that checks if the quota parameters in "/etc/fstab" are set. To disable this test, the file

"50.rquotad.check" must be moved in the directory "/etc/ctdb/nfs-checks.d/". Then the test is not executed.

For starting the services CTDB uses event scripts which have to be activated. Only then CTDB can start and stop the services. Listing-12 shows the activation of the two required scripts:
-------Listing-12-------
ctdb event script enable legacy 60.nfs
ctdb event script enable legacy 06.nfs

Subsequently, it makes sense to restart all nodes once, this is the only way to ensure that all services use static ports and that the services are started in the correct order. After the first restart, the services can always be restarted via CTDB. To do this, the CTDB is restarted via the Systemd.

The NFSv3 Client
Now only the test on a NFS client is missing. For this, the package "nfs-common" is required on a Debian client. Only when the package is installed can an NFS share be mounted. Listing-13 shows the mounting and the result of the mount process:
--------Listing-13-------
```
mkdir /nfs

mount -t nfs cluster01:/glusterfs /nfs

mount
cluster01:/glusterfs on /nfs type nfs
(rw,relatime,vers=3,rsize=262144,wsize=262144,namlen=255,hard,proto=t
cp,timeo=600,retrans=2,sec=sys,mountaddr=192.168.56.211,mountvers=3,m
ountport=892,mountproto=udp,local_lock=none,addr=192.168.56.211)
```

The result of the "mount" command shows that the connection to the node with the IP address 192.168.56.211 has been established. To test what happens in the event of a failure, the CTDB service can be stopped on the node that has this dynamic IP address. The other node then receives the IP address that has just been released.

After a short time, the client reconnects to the cluster, but this time to the other node. The client always reconnects to the same IP address, even if the node that was just switched off comes back online.

The time to reconnect when a node fails can be shortened using the mount options in Listing-14:
----------Listing-14---------
```
mount -t nfs cluster01:/glusterfs /nfs -o soft,timeo=20
```

The value of the parameter "timeo" is given in steps of 0.1 seconds. So in this case 2 seconds.

This concludes the setup of NFSv3. The next section will then cover the specifics and setup of NFSv4.

NFSv4 what is different?
Compared to NFSv3, NFSv4 is not only more performant, but also has the following advantages:

- Direct filelocking by the server without the lockd
- Faster access due to improved caching
- Stateless communication, thus simple management of accesses
- File operations can be delegated to the client
- UTF-8 in file names possible
- More than 16 groups possible in ACLs
- Kerberos can be used for authentication

The following points still need to be taken into account for the operation of NFSv4:
- Only one daemon, thus no more dynamic ports
- Only TCP port 2049 is still needed
- All shares are combined in a pseudo file system
- Idmapd is used to map groups and users

It is the change in functionality that makes it so interesting to use NFSv4 in cluster operation, because the portmapper is no longer needed and the ports can be disabled.
With a standalone NFSv4 server this is also feasible, you can even set up a firewall on the NFS server itself, which only releases port 2049. But: Here CTDB should be used to control NFS. Unfortunately, the CTDB server is currently only prepared for use with NFSv3, but with some manual work CTDB can also control an NFSv4 server. The needed steps to the goal are described in this section.

The starting point is again the running CTDB service. The NFS settings have not been changed yet. But the required packages are already installed.

NFSv4 setup
First, version 3 of the NFS server is disabled. To do this, the lines from Listing-15 are adapted in the "/etc/nfs.conf" file.
--------Listing-15--------
```
[nfsd]
vers3=n
[statd]
name=cluster01.example.net
```

This disables version 3. Now that the portmapper services are no longer required, it is also no longer necessary to specify the ports of the other services. Only the fqdn of the cluster must be entered additionally.
Since the CTDB is to start the NFS service later here as well, the service can be deactivated in Systemd again.

Setting up the pseudo file system

As mentioned at the beginning, NFSv4 does not provide the actual shares for the clients, but all shares are first mounted in a pseudo file system and then released from it. The client thus has no possibility of breaking out of the pseudo file system.

A separate directory is required for the pseudo file system, which then serves as a mount point for all shares. Within the directory, a separate directory is required for each individual share. Listing-16 shows the creation of the directories including the permissions that must be assigned:

-------Listing-16---------

```
mkdir -m 1777 /nfs4root
mkdir -m 1777 /nfs4root/glusterfs
```

All nodes of the future cluster need the identical folder structure. The permissions on the folders do not tell what permissions a client will have later on the actual file system. The access rights for the client are still assigned in the directory of the data.

Now the first share can be mounted with the command "mount --bind /glusterfs /nfs4root/glusterfs". If the command was successful, the line from Listing-17 can be entered in the "/etc/fstab" file:

--------Listing-17--------

```
/glusterfs /nfs4root/glusterfs none rw,bind 0 0
```

With NFSv4, the shares are also entered in the file "/etc/exports", except that here, in addition to the actual shares, there must also be a line for the pseudo file system. Listing-18 shows the shares set up here:

---------Listing-18------

```
/nfs4root
192.168.56.0/24(ro,sync,insecure,root_squash,no_subtree_check,fsid=0)
/nfs4root/glusterfs
192.168.56.0/24(rw,sync,insecure,no_root_squash,no_subtree_check,fsid
=1)
```

As with NFSv3, the "fsid" parameter is required for each share. The only important thing here is that the pseudo file system always has the "fsid=0".
This completes the setup of the NFS server, now CTDB for NFSv4 can be set up.

Setting up CTDB for NFSv4

In the first step, the line from Listing-11 is again added to the file "/etc/default/quote if no quota is to be used. A port does not have to be assigned at this point.

Now the situation with CTDB is that actually only NFSv3 is supported at the moment, because CTDB runs some tests that address and test the portmapper. The portmapper is also started on NFSv4 but does not accept any requests, so all tests fail and the CTDB service does not start. For this reason, the CTDB test scripts for NFS are moved. Listing-19 shows all commands necessary for this:

----------Listing-19----------

```
cd /etc/ctdb/nfs-checks.d/
mkdir not-needed
mv 00.portmapper.check 10.status.check 30.nlockmgr.check
40.mountd.check 50.rquotad.check not-needed/
```

In the same directory it is still necessary to change in the file "20.nfs.check". Listing-20 shows all changes:

--------Listing-20----------

```
version="4"
service_check_cmd="systemctl -q is-active nfs-kernel-server"
service_check_cmd="pidof -q nfsd"
```

Only now can the two event scripts "60.nfs" and "06.nfs" be activated on all nodes. A subsequent restart of the CTDB service now makes the NFSv4 servers available. A look into the file "proc/fs/nfsd/version" shows if really only NFSv4 is supported. The versions "2" and "3" are displayed there with a leading minus and are thus deactivated.

Notice.
The command "rpcinfo p" still shows all ports and the service is also running. Actually the services could be disabled with the command "systemctl disable rcpbind.socket rpcbind.service", but the nfs-kernel-server needs the services, although they are not used.

Testing on the client

Here again the test follows whether the client can connect and whether NFSv4 is really used. On the client the identical command is used as with NFSv3, only the result is now different. Listing-21 shows the command for mounting and the result:

-------Listing-21---------------

```
mount -t nfs cluster01:/glusterfs /nfs -o soft,timeo=20
cluster01:/glusterfs on /nfs type nfs4
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,soft,proto=tcp,timeo=20,retrans=2,sec=sys,clientaddr=192.168.56.203,local_lock=none,addr=192.168.56.212)
```

Although only "nfs" is specified here as the network file system, the client automatically uses NFSv4. The client automatically detects which NFS version is supported and selects the latest version.

Here, too, the failure of a node can be simulated by stopping the CTDB service on the node to which the client has connected. The client then automatically switches to the other node, which now has the IP address of the failed node. When the service is restarted, the client swings back to the original node.

Conclusion

NFS can also be used to set up a highly available file server with DNS load balancing. Together with GlusterFS and CTDB, files can be distributed in the network in a failsafe manner.