Red Hat Gluster Storage Advanced Features: Lab Guide

Red Hat Gluster Storage Advanced Features Information	
Technology/Product	Red Hat Gluster Storage
Difficulty	4
Time	90 minutes
Prerequisites	Fundamental understanding of gluster systems and processes, and familiarity with the command interface.

In this lab you will...

- Explore the GlusterFS data layout and extended attributes
- Understand the structure of the volfiles
- Administer quotas on volumes and subdirectories
- Induce a data split brain and observe the effects
- Administer quorum and understand its actions
- Configure asynchronous geo-replication
- Take snapshots, observe the operations, and restore data
- Configure and understand disperse volumes (erasure coding)

BEFORE YOU BEGIN...

Presumptions:

You already know how to...

- Configure a brick backend LVM structure and filesystem
- Peer gluster nodes into a trusted pool
- Create gluster volumes
- Mount gluster volumes to a filesystem client and access data

And you're comfortable with the gluster concepts of...

- Distributed volumes
- Replicated volumes
- Self-heal
- Rebalance

- Geo-replication

Know your lab:

- Most of the lab commands are executed on node n1. Where it is important that a command be executed on a different node, the node hostname will be highlighted in **green** to call out the difference in the lab instructions.
- Some important things to observe in the output of lab commands are highlighted in **yellow**.
- Root Password: redhat
- *Trusted Pool #1*
  - Nodes: n1 – n4
  - IPs: 10.11.12.101 – 104
  - Volume: `rep01` – Distributed-Replicated Volume 2x2
    - Brick Directories: `/rhgs/bricks/rep01`
    - Client Mountpoint: `/rhgs/client/rep01`
    - Pre-Populated Data: file001 – 100 (1MB each)
- *Trusted Pool #2*
  - Nodes: n5 – n6
  - IPs: 10.11.12.105 – 106
  - Volume: `srep01` – Distributed Volume
    - Brick Directories: `/rhgs/bricks/srep01`

Disclaimer:

This is alpha-level code we're working with in the lab. Here be dragons.

LAB INSTRUCTIONS

The Magic .glusterfs Directory

Take a look inside...

```
[root@n1 ~]# ls /rhgs/bricks/rep01/.glusterfs
```

The DHT (Distribute Hash Translator) assigns hash values to files on a per-directory basis. We can view the assigned value for a particular file in its metadata as the `trusted.gfid` value. Note that this metadata is only visible from the brick view of the file, not from the client mount.

```
[root@n1 ~]# getfattr -d -m . -e hex /rhgs/bricks/rep01/file002
getfattr: Removing leading '/' from absolute path names
# file: rhgs/bricks/rep01/file002
trusted.afr.dirty=0x000000000000000000000000
trusted.afr.rep01-client-0=0x000000000000000000000000
trusted.afr.rep01-client-1=0x000000000000000000000000
trusted.bit-rot.version=0x02000000000000000556cac8f0008d758
trusted.gfid=0x57ab4dd8afae41eda54f6ecba5985a6b
```

With the gfid from the extended attributes, we can locate the file within the brick's `.glusterfs` directory structure. Files are stored by their gfid values in a directory hierarchy two levels deep. The first level directory is named for the

first two characters of the gfid, and the second level directory for the second two characters of the gfid. The file is then named for the complete gfid value in UUID format (8-4-4-4-12).

```
[root@n1 ~]# ls /rhgs/bricks/rep01/.glusterfs/57/ab/57ab4dd8-afae-41ed-a54f-6ecba5985a6b
```

This gfid file represents a hard link to the file in its structured location within the filesystem. We can prove this simply with the `find` command. Note the matching inode numbers.

```
[root@n1 ~]# find /rhgs/bricks/rep01 -samefile /rhgs/bricks/rep01/\
  .glusterfs/57/ab/57ab4dd8-afae-41ed-a54f-6ecba5985a6b | xargs ls -li
136 -rw-r--r-- 2 root root 1048576 Jun  1 15:08 /rhgs/bricks/rep01/file002
136 -rw-r--r-- 2 root root 1048576 Jun  1 15:08
/rhgs/bricks/rep01/.glusterfs/57/ab/57ab4dd8-afae-41ed-a54f-6ecba5985a6b
```

Volfiles

Gluster builds functionality by layering translators into a stack. Each translator is a piece of code that adds incremental functionality or data transformation. The complete stack is stored as a volfile with a hierarchical structure. The volfiles for different functions of Gluster can be viewed in the `/var/lib/glusterd/vols/<volname>` directories.

```
[root@n1 ~]# ls /var/lib/glusterd/vols/rep01
```

The fuse volfile contains the stack to enable Gluster native client access to a volume. Note that we define four `client` subvolumes, each of which representing an individual brick. Then the `clients` are grouped in pairs as part of two defined `replicate` subvolumes. Then the `replicates` are grouped into one `dht` subvolume. This hierarchy represents the base functionality of a distribute-replicate 2x2 volume. The remaining subvolumes in the stack each add incremental functionality in the particular order they are defined. The last entry `rep01` represents the volume as it is presented to the client for consumption.

```
[root@n1 ~]# cat /var/lib/glusterd/vols/rep01/rep01.tcp-fuse.vol
volume rep01-client-0
    type protocol/client
    option send-gids true
    option transport-type tcp
    option remote-subvolume /rhgs/bricks/rep01
    option remote-host n1
    option ping-timeout 42
end-volume

volume rep01-client-1
    type protocol/client
    option send-gids true
    option transport-type tcp
    option remote-subvolume /rhgs/bricks/rep01
    option remote-host n2
    option ping-timeout 42
end-volume

volume rep01-client-2
    type protocol/client
```

```
    option send-gids true
    option transport-type tcp
    option remote-subvolume /rhgs/bricks/rep01
    option remote-host n3
    option ping-timeout 42
end-volume

volume rep01-client-3
    type protocol/client
    option send-gids true
    option transport-type tcp
    option remote-subvolume /rhgs/bricks/rep01
    option remote-host n4
    option ping-timeout 42
end-volume

volume rep01-replicate-0
    type cluster/replicate
    subvolumes rep01-client-0 rep01-client-1
end-volume

volume rep01-replicate-1
    type cluster/replicate
    subvolumes rep01-client-2 rep01-client-3
end-volume

volume rep01-dht
    type cluster/distribute
    subvolumes rep01-replicate-0 rep01-replicate-1
end-volume

volume rep01-write-behind
    type performance/write-behind
    subvolumes rep01-dht
end-volume

volume rep01-read-ahead
    type performance/read-ahead
    subvolumes rep01-write-behind
end-volume

volume rep01-readdir-ahead
    type performance/readdir-ahead
    subvolumes rep01-read-ahead
end-volume

volume rep01-io-cache
```

```
    type performance/io-cache
    subvolumes rep01-readdir-ahead
end-volume

volume rep01-quick-read
    type performance/quick-read
    subvolumes rep01-io-cache
end-volume

volume rep01-open-behind
    type performance/open-behind
    subvolumes rep01-quick-read
end-volume

volume rep01-md-cache
    type performance/md-cache
    subvolumes rep01-open-behind
end-volume

volume rep01
    type debug/io-stats
    option count-fop-hits off
    option latency-measurement off
    subvolumes rep01-md-cache
end-volume
```

The Gluster native client stores a copy of the volfile in memory, and it records in its log file each time the volfile changes.

```
[root@n1 ~]# grep -v '^\[' /var/log/glusterfs/rhgs-client-rep01.log
+------------------------------------------------------------------------------+
Final graph:
+------------------------------------------------------------------------------+
  1: volume rep01-client-0
  2:     type protocol/client
  3:     option ping-timeout 42
  4:     option remote-host n1
  5:     option remote-subvolume /rhgs/bricks/rep01
  6:     option transport-type socket
  7:     option username 960a3c99-d94e-4b15-9135-5f56aef9d63c
  8:     option password 439b2bab-0984-471a-8af2-242c05e2c2a5
  9:     option send-gids true
 10: end-volume
 …
```

Changing volume options may result in either a modification of an existing translator, or the insertion of a new translator into the stack.

```
[root@n1 ~]# gluster volume set rep01 performance.open-behind on
volume set: success
```

```
[root@n1 ~]# gluster volume set rep01 cluster.quorum-type auto
volume set: success
[root@n1 ~]# gluster volume info rep01
Volume Name: rep01
Type: Distributed-Replicate
Volume ID: 6ff17d21-035d-47e7-8bd1-d4a9e850be31
Status: Started
Number of Bricks: 2 x 2 = 4
Transport-type: tcp
Bricks:
Brick1: n1:/rhgs/bricks/rep01
Brick2: n2:/rhgs/bricks/rep01
Brick3: n3:/rhgs/bricks/rep01
Brick4: n4:/rhgs/bricks/rep01
Options Reconfigured:
cluster.quorum-type: auto
performance.open-behind: on
performance.readdir-ahead: on
geo-replication.indexing: on
geo-replication.ignore-pid-check: on
changelog.changelog: on
[root@n1 ~]# grep -B5 -A1 open-behind /var/lib/glusterd/vols/rep01/rep01.tcp-fuse.vol
volume rep01-quick-read
    type performance/quick-read
    subvolumes rep01-io-cache
end-volume

volume rep01-open-behind
    type performance/open-behind
    subvolumes rep01-quick-read
end-volume

volume rep01-md-cache
    type performance/md-cache
    subvolumes rep01-open-behind
end-volume
[root@n1 ~]# grep -B2 -A2 quorum-type /var/lib/glusterd/vols/rep01/rep01.tcp-fuse.vol
volume rep01-replicate-0
    type cluster/replicate
    option quorum-type auto
    subvolumes rep01-client-0 rep01-client-1
end-volume
--
volume rep01-replicate-1
    type cluster/replicate
    option quorum-type auto
    subvolumes rep01-client-2 rep01-client-3
```

```
end-volume
```

IMPORTANT: Reset the volume options before continuing with the lab.

```
[root@n1 ~]# gluster volume reset rep01 force
volume reset: success: reset volume successful
```

Quotas

Note: Ensure the native client mount is active before continuing.

```
[root@n1 ~]# mount /rhgs/client/rep01
```

```
[root@n2 ~]# mount /rhgs/client/rep01
```

Quotas are set at the directory level, but can be made effectively volume-level by setting the quota at the volume root. A *client-created* subdirectory must exist before a quota can be applied to it.

```
[root@n1 ~]# mkdir /rhgs/client/rep01/testdir
[root@n1 ~]# gluster volume quota rep01 enable
volume quota : success
[root@n1 ~]# gluster volume info rep01 | grep quota
features.inode-quota: on
features.quota: on
[root@n1 ~]# gluster volume quota rep01 limit-usage / 200MB
volume quota : success
[root@n1 ~]# gluster volume quota rep01 limit-usage /testdir 50MB
volume quota : success
[root@n1 ~]# gluster volume quota rep01 list
                  Path                   Hard-limit Soft-limit   Used  Available  Soft-
limit exceeded? Hard-limit exceeded?
-----------------------------------------------------------------------------------------
---------------------------------
/                                         200.0MB       80%     101.0MB  99.0MB
No                  No
/testdir                                   50.0MB       80%      0Bytes  50.0MB
No                  No
```

Note that by default the root quota limit is not reflected as a size limit of the client mount point. To enable this functionality, the `quota-deem-statfs` feature must turned on.

```
[root@n1 ~]# df -h /rhgs/client/rep01/
Filesystem      Size  Used Avail Use% Mounted on
n1:rep01         16G  167M   16G   2% /rhgs/client/rep01
[root@n1 ~]# gluster volume set rep01 features.quota-deem-statfs on
volume set: success
[root@n1 ~]# df -h /rhgs/client/rep01/
Filesystem      Size  Used Avail Use% Mounted on
n1:rep01        200M  101M   99M  51% /rhgs/client/rep01
```

The below commands from the quota_setup.sh script are for lab purposes only. Because we are testing with small quotas on a distributed system with a fast network interlink between hosts, we need to enforce strict accounting by reducing all timeouts to zero.

```
[root@n1 ~]# ~/quota_setup.sh
$ gluster volume set rep01 features.quota-timeout 0
volume set: success
$ gluster volume set rep01 features.hard-timeout 0
volume set: success
$ gluster volume set rep01 features.soft-timeout 0
volume set: success
```

Write enough data to our `testdir` to exceed the soft quota limit (default is 80%), and observe the effects. (The `quota1.sh` script has been placed in `/root` for your convenience.)

```
[root@n1 ~]# ~/quota1.sh
$ dd if=/dev/urandom of=/rhgs/client/rep01/testdir/testfile01 bs=1k count=1k
1024+0 records in
1024+0 records out
1048576 bytes (1.0 MB) copied, 0.199832 s, 5.2 MB/s
…
[root@n1 ~]# gluster volume quota rep01 list
                  Path                   Hard-limit Soft-limit   Used  Available  Soft-
limit exceeded? Hard-limit exceeded?
-----------------------------------------------------------------------------------------
---------------------------------
/                                         200.0MB       80%     142.0MB  58.0MB
No                   No
/testdir                                   50.0MB       80%      41.0MB   9.0MB
Yes                   No
```

Now we'll exceed the hard limit and prove that quota is indeed working as expected, and the client receives the appropriate EDQUOT error. The exception is recorded in both the brick and the client logs.

```
[root@n1 ~]# ~/quota2.sh
$ dd if=/dev/urandom of=/rhgs/client/rep01/testdir/testfile42 bs=1k count=1k
1024+0 records in
1024+0 records out
1048576 bytes (1.0 MB) copied, 0.17068 s, 6.1 MB/s
…
dd: failed to open ‘/rhgs/client/rep01/testdir/testfile51’: Disk quota exceeded
dd: failed to open ‘/rhgs/client/rep01/testdir/testfile52’: Disk quota exceeded
…
[root@n1 ~]# gluster volume quota rep01 list
                  Path                   Hard-limit Soft-limit   Used  Available  Soft-
limit exceeded? Hard-limit exceeded?
-----------------------------------------------------------------------------------------
---------------------------------
/                                         200.0MB       80%     151.0MB  49.0MB
No                   No
/testdir                                   50.0MB       80%      50.0MB  0Bytes
Yes                  Yes
[root@n1 ~]# grep exceeded /var/log/glusterfs/bricks/rhgs-bricks-rep01.log | tail -n1
```

```
[2015-06-02 19:15:56.829819] I [server-rpc-fops.c:1558:server_create_cbk] 0-rep01-server:
27585: CREATE /testdir/testfile69 (8180c1f2-b586-4f6c-9ad9-e83bf3d073c1/testfile69) ==>
(Disk quota exceeded)
[root@n1 ~]# grep exceeded /var/log/glusterfs/rhgs-client-rep01.log | tail -n1
[2015-06-02 19:15:56.838210] W [fuse-bridge.c:1970:fuse_create_cbk] 0-glusterfs-fuse:
472007: /testdir/testfile70 => -1 (Disk quota exceeded)
```

IMPORTANT: Reset the volume options before continuing with the lab.

```
[root@n1 ~]# gluster volume reset rep01 force
volume reset: success: reset volume successful
```

Split-Brain

In order to observe the complicated effects of a split-brain scenario, we will induce a network split between nodes n1 and n2, and then separately write to the same file on both nodes via the native client.

```
[root@n1 ~]# ~/split1.sh
Inducing split-brain with iptables…
$ iptables -F
$ iptables -A OUTPUT -d n2 -j DROP
Adding 1MB of random data to file002...
1024+0 records in
1024+0 records out
1048576 bytes (1.0 MB) copied, 0.147373 s, 7.1 MB/s
Generating md5sum for file002...
$ md5sum /rhgs/client/rep01/file002
5dc270754774cf733a97797a33dd9f82  /rhgs/client/rep01/file002
[root@n1 ~]# ls -lh /rhgs/client/rep01/file002
-rw-r--r-- 1 root root 2.1M Jun  2 15:52 /rhgs/client/rep01/file002
```

```
[root@n2 ~]# ~/split2.sh
Adding 2MB of random data to file002...
2048+0 records in
2048+0 records out
2097152 bytes (2.1 MB) copied, 0.281429 s, 7.5 MB/s
Generating md5sum for file002...
$ md5sum /rhgs/client/rep01/file002
7e0064e6eeb04014f694b536228ca7d8  /rhgs/client/rep01/file002
[root@n2 ~]# ls -lh /rhgs/client/rep01/file002
-rw-r--r-- 1 root root 3.1M Jun  2 15:53 /rhgs/client/rep01/file002
```

We can observe in the file extended attributes on both bricks that each copy of `file002` believes itself to be WISE (pending no changes, but accusing it's replica peer of pending changes). This is not a conflict that Gluster can automatically resolve because there is not enough information to determine which is the “good” copy of the data – and in fact *both* copies may be good. *Note from our review of the volfile above that the brick on host n1 is `rep01-client-0` and the brick on host n2 is `rep01-client-1`.*

```
[root@n1 ~]# getfattr -d -m . -e hex /rhgs/bricks/rep01/file002 |grep afr.rep01
getfattr: Removing leading '/' from absolute path names
```

```
trusted.afr.rep01-client-0=0x000000000000000000000000
trusted.afr.rep01-client-1=0x0000003a0000000000000000
```

```
[root@n2 ~]# getfattr -d -m . -e hex /rhgs/bricks/rep01/file002 |grep afr.rep01
getfattr: Removing leading '/' from absolute path names
trusted.afr.rep01-client-0=0x0000001e0000000000000000
trusted.afr.rep01-client-1=0x000000000000000000000000
```

Each trusted.afr extended attribute has a value which is 24 hexadecimal digits. The first 8 digits represent the changelog of data, the second 8 digits the changelog of metadata, and the last 8 digits the changelog of directory entries. Values other than 0 indicate that changes of that type are expected to be pending for the brick indicated.

```
0x 000003d7 00000001 00000110
        |      |       |
        |      |        \_ changelog of directory entries
        |       \_ changelog of metadata
         \ _ changelog of data
```

When the copies of the file disagree on the changelogs, each believing itself consistent but its peer pending changes, the file is in a split-brain state and cannot be automatically healed without conceding to ignore one copy's pending changes and accept the other's.

Resolving the network split, we can see that Gluster acknowledges the split-brain inconsistency in the file.

```
[root@n1 ~]# ~/split3.sh
Correcting network split with iptables...
$ iptables -F OUTPUT
Dropping caches due to BZ 1229226...
[root@n1 ~]# gluster volume heal rep01 info split-brain
Brick n1:/rhgs/bricks/rep01/
/file002
Number of entries in split-brain: 1

Brick n2:/rhgs/bricks/rep01/
/file002
Number of entries in split-brain: 1

Brick n3:/rhgs/bricks/rep01/
Number of entries in split-brain: 0

Brick n4:/rhgs/bricks/rep01/
Number of entries in split-brain: 0
```

When we attempt to read the split-brain file, Gluster returns EIO. This also triggers a message reported in the client log file.

```
[root@n1 ~]# cat /rhgs/client/rep01/file002 > /dev/null
cat: /rhgs/client/rep01/file002: Input/output error
```

```
[root@n1 ~]# grep split /var/log/glusterfs/rhgs-client-rep01.log
[2015-06-11 22:21:55.757038] W [MSGID: 108008] [afr-read-txn.c:241:afr_read_txn] 0-rep01-replicate-0: Unreadable subvolume -1 found with event generation 4. (Possible split-brain)
```

If after troubleshooting the split and analyzing the data we decide that the copy on n1 is the one we want to keep, we can resolve the split-brain manually on a per-file basis by clearing the appropriate `trusted.afr` flags. In this case, we will clear the flags on the `trusted.afr.rep01-client-0` attribute for `file002` on n2. Doing this clears n2's belief that the file on n1 is pending changes, and therefore allows the self-heal process to copy the file over from n1 to n2.

```
[root@n2 ~]# setfattr -n trusted.afr.rep01-client-0 -v 0x000000000000000000000000 \
 /rhgs/bricks/rep01/file002
[root@n2 ~]# getfattr -d -m . -e hex /rhgs/bricks/rep01/file002 |grep afr.rep01
getfattr: Removing leading '/' from absolute path names
trusted.afr.rep01-client-0=0x000000000000000000000000
trusted.afr.rep01-client-1=0x000000000000000000000000
```

The file will be healed when either it it access via the client, or when the self-heal daemon passes over it.

```
[root@n2 ~]# ls -lh /rhgs/bricks/rep01/file002
-rw-r--r-- 1 root root 3.1M Jun 11 18:20 /rhgs/client/rep01/file002
[root@n2 ~]# cat /rhgs/client/rep01/file002 > /dev/null
[root@n2 ~]# ls -lh /rhgs/bricks/rep01/file002
-rw-r--r-- 2 root root 2.1M Jun 11 18:51 /rhgs/bricks/rep01/file002
```

IMPORTANT: Reset the split files before continuing.

```
[root@n1 ~]# ~/split_reset.sh
Flushing firewall...
$ iptables -F
Deleting split files...
$ rm -f /rhgs/bricks/rep01/file002
$ rm -f /rhgs/bricks/rep01/.glusterfs/7e/fd/7efdf29f-9d42-4307-95d1-972c87dd8299
Healing...
Regenerating file...
$ dd if=/dev/urandom of=/rhgs/client/rep01/file002 bs=1k count=1k
1024+0 records in
1024+0 records out
1048576 bytes (1.0 MB) copied, 0.161904 s, 6.5 MB/s
```

Quorum – Server-Side

Server-side quorum is pool-aware only – it is not volume- or replica-aware. It attempts to enforce data consistency by killing glusterfsd brick processes on non-quorate nodes.

```
[root@n1 ~]# gluster volume set rep01 cluster.server-quorum-type server
volume set: success
```

We observe that enabling server-side quorum does not prevent the same split-brain scenario we induced previously.

```
[root@n1 ~]# ~/split1.sh
Inducing split-brain with iptables…
```

```
$ iptables -F
$ iptables -A OUTPUT -d n2 -j DROP
Adding 1MB of random data to file002...
1024+0 records in
1024+0 records out
1048576 bytes (1.0 MB) copied, 0.147373 s, 7.1 MB/s
Generating md5sum for file002...
$ md5sum /rhgs/client/rep01/file002
5dc270754774cf733a97797a33dd9f82  /rhgs/client/rep01/file002
[root@n1 ~]# ls -lh /rhgs/client/rep01/file002
-rw-r--r-- 1 root root 2.1M Jun  2 15:52 /rhgs/client/rep01/file002
```

```
[root@n2 ~]# ~/split2.sh
Adding 2MB of random data to file002...
2048+0 records in
2048+0 records out
2097152 bytes (2.1 MB) copied, 0.281429 s, 7.5 MB/s
Generating md5sum for file002...
$ md5sum /rhgs/client/rep01/file002
7e0064e6eeb04014f694b536228ca7d8  /rhgs/client/rep01/file002
[root@n2 ~]# ls -lh /rhgs/client/rep01/file002
-rw-r--r-- 1 root root 3.1M Jun  2 15:53 /rhgs/client/rep01/file002
```

In this situation, only one node out of four has lost contact, and so the default quorum minimum of 50% has not been surpassed.

IMPORTANT: Reset the split files before continuing.

```
[root@n1 ~]# ~/split_reset.sh
Flushing firewall...
$ iptables -F
Deleting split files...
$ rm -f /rhgs/bricks/rep01/file002
$ rm -f /rhgs/bricks/rep01/.glusterfs/7e/fd/7efdf29f-9d42-4307-95d1-972c87dd8299
Healing...
Regenerating file...
$ dd if=/dev/urandom of=/rhgs/client/rep01/file002 bs=1k count=1k
1024+0 records in
1024+0 records out
1048576 bytes (1.0 MB) copied, 0.161904 s, 6.5 MB/s
```

This time we induce a network split that will exceed the threshold and cause the quorum enforcement to kick in.

```
[root@n1 ~]# ~/split1a.sh
Inducing network split with iptables...
$ iptables -F
$ iptables -A OUTPUT -d n2 -j DROP
$ iptables -A OUTPUT -d n3 -j DROP
```

```
Sleeping for 1 minute...
$ sleep 60
Adding 1MB of random data to file002...
./split1a.sh: line 12: /rhgs/client/rep01/file002: No such file or directory

** You should have received a "No such file or directory" error above... **
```

We observe that the brick process on n1 has been killed.

```
[root@n1 ~]# ps -ef | grep glusterfsd | grep -v grep
```

The glusterd log file shows us that a loss of quorum was observed and acted on by killing the local bricks.

```
[root@n1 ~]# grep quorum /var/log/glusterfs/etc-glusterfs-glusterd.vol.log
…
[2015-06-08 15:51:57.558778] C [MSGID: 106002] [glusterd-server-
quorum.c:356:glusterd_do_volume_quorum_action] 0-management: Server quorum lost for
volume rep01. Stopping local bricks.
```

We also observe that the client can now only see the files available from the brick on n4, since n2 and n3 are out-of-contact and the local brick on n1 is dead.

```
[root@n1 ~]# ls /rhgs/client/rep01/
file000  file012  file017  file030  file040  file048  file061  file080  file090  file098
file001  file013  file018  file032  file042  file049  file073  file081  file093  testdir
file003  file014  file022  file033  file044  file050  file075  file085  file095
file006  file015  file025  file034  file046  file052  file076  file088  file096
file007  file016  file026  file036  file047  file053  file078  file089  file097
```

Flushing the iptables rules and waiting a few moments, we see quorum regained and the brick restarted.

```
[root@n1 ~]# ~/split3.sh
Correcting network split with iptables...
$ iptables -F OUTPUT
Dropping caches due to BZ 1229226...
[root@n1 ~]# ps -ef | grep glusterfsd | grep -v grep
root     15062     1  0 12:05 ?        00:00:00 /usr/sbin/glusterfsd -s n1 --volfile-id
rep01.n1.rhgs-bricks-rep01 -p /var/lib/glusterd/vols/rep01/run/n1-rhgs-bricks-rep01.pid
-S /var/run/gluster/720b9371db42f41c2c417722306fb4bf.socket --brick-name
/rhgs/bricks/rep01 -l /var/log/glusterfs/bricks/rhgs-bricks-rep01.log --xlator-option *-
posix.glusterd-uuid=e300ada3-5157-49a9-9100-e662c2a617df --brick-port 49152 --xlator-
option rep01-server.listen-port=49152
[root@n1 ~]# grep quorum /var/log/glusterfs/etc-glusterfs-glusterd.vol.log
…
[2015-06-08 16:05:02.430800] C [MSGID: 106003] [glusterd-server-
quorum.c:351:glusterd_do_volume_quorum_action] 0-management: Server quorum regained for
volume rep01. Starting local bricks.
```

IMPORTANT: Reset the volume options before continuing with the lab.

```
[root@n1 ~]# gluster volume reset rep01 force
volume reset: success: reset volume successful
```

Quorum – Client-Side

Client-side quorum is more intelligent, with an awareness of the volume layout and replica peers.

```
[root@n1 ~]# gluster volume set rep01 cluster.quorum-type auto
volume set: success
```

We observe that enabling client-side quorum allows the write from n1 to proceed during the split.

```
[root@n1 ~]# ~/split1.sh
Inducing network split with iptables...
$ iptables -F
$ iptables -A OUTPUT -d n2 -j DROP
Adding 1MB of random data to file002...
1024+0 records in
1024+0 records out
1048576 bytes (1.0 MB) copied, 0.163652 s, 6.4 MB/s
Generating md5sum for file002...
$ md5sum /rhgs/client/rep01/file002
01a3abda9270b4d15f49d32d4145f72c  /rhgs/client/rep01/file002
```

However, the write from n2 is blocked with the filesystem set to read-only.

```
[root@n2 ~]# ~/split2.sh
Adding 2MB of random data to file002...
dd: error writing ‘standard output’: Read-only file system
dd: closing output file ‘standard output’: Read-only file system
Generating md5sum for file002...
$ md5sum /rhgs/client/rep01/file002
8b5207d8cc2f6df5b2e29166bc27bd72  /rhgs/client/rep01/file002
```

In this case, because the brick on n1 is the first defined member of the replica pair of `n1:/rhgs/bricks/rep01` and `n2:/rhgs/bricks/rep01`, it receives an extra weight or vote in the quorum calculation. This default behavior allows for one member of a brick pair to remain in a read-write state even during a split.

Unlike server-side quorum where all access to a portion of the data is lost, client-side quorum allows for continued access that is degraded to read-only on one side of the split.

We observe that the logging is similar to server-side quorum, but is recorded in the client log files and only on node n2 where the quorum was enforced.

```
[root@n2 ~]# grep quorum /var/log/glusterfs/rhgs-client-rep01.log
…
[2015-06-08 16:35:06.112351] W [MSGID: 108001] [afr-common.c:3963:afr_notify] 0-rep01-
replicate-0: Client-quorum is not met
```

Flushing the iptables rules on n1 and waiting a few moments, we see quorum regained and the client on node n2 regains read-write access to the data..

```
[root@n1 ~]# ~/split3.sh
Correcting network split with iptables...
$ iptables -F OUTPUT
Dropping caches due to BZ 1229226...
```

```
[root@n2 ~]# grep quorum /var/log/glusterfs/rhgs-client-rep01.log
```

```
…
[2015-06-08 16:50:50.802194] I [MSGID: 108002] [afr-common.c:3959:afr_notify] 0-rep01-
replicate-0: Client-quorum is met
[root@n2 ~]# touch /rhgs/client/rep01/file002
```

Geo-Replication

Geo-replication provides asynchronous one-way replication to one or more remote Gluster volumes. The replication is negotiated, with one member of each synchronous replica set acting as an active sender and the others acting as passive failover members. Data is geo-replicated in parallel from all synchronous replica sets.

Note that initial configuration of security credentials and setup of the receiving slave volume has already been completed on your lab nodes.

```
[root@n1 ~]# gluster volume geo-replication rep01 n5::srep01 create push-pem
Creating geo-replication session between rep01 & n5::srep01 has been successful
[root@n1 ~]# gluster volume geo-replication rep01 status

MASTER NODE    MASTER VOL    MASTER BRICK          SLAVE USER    SLAVE
SLAVE NODE    STATUS     CRAWL STATUS    LAST_SYNCED
------------------------------------------------------------------------------------------
-----------------------------------------------
n1             rep01         /rhgs/bricks/rep01    root          ssh://n5::srep01    N/A
Created    N/A             N/A
n4             rep01         /rhgs/bricks/rep01    root          ssh://n5::srep01    N/A
Created    N/A             N/A
n3             rep01         /rhgs/bricks/rep01    root          ssh://n5::srep01    N/A
Created    N/A             N/A
n2             rep01         /rhgs/bricks/rep01    root          ssh://n5::srep01    N/A
Created    N/A             N/A
```

We start the geo-replication and immediately run a watch on the `status` command in order to observe the members in the `Initializing` state, and then progressing through `Hybrid Crawl` and `Changelog Crawl`.

```
[root@n1 ~]# gluster volume geo-replication rep01 n5::srep01 start &&
  watch -n .5 gluster volume geo-replication rep01 n5::srep01 status
Starting geo-replication session between rep01 & n5::srep01 has been successful

MASTER NODE    MASTER VOL    MASTER BRICK          SLAVE USER    SLAVE
SLAVE NODE    STATUS             CRAWL STATUS    LAST_SYNCED
------------------------------------------------------------------------------------------
-----------------------------------------------------
n1             rep01         /rhgs/bricks/rep01    root          ssh://n5::srep01    N/A
Initializing...    N/A             N/A
n2             rep01         /rhgs/bricks/rep01    root          ssh://n5::srep01    N/A
Initializing...    N/A             N/A
n4             rep01         /rhgs/bricks/rep01    root          ssh://n5::srep01    N/A
Initializing...    N/A             N/A
n3             rep01         /rhgs/bricks/rep01    root          ssh://n5::srep01    N/A
Initializing...    N/A             N/A
```

IMPORTANT: Stop geo-replication before continuing with the lab.

```
[root@n1 ~]# gluster volume geo-replication rep01 n5::srep01 stop
Stopping geo-replication session between rep01 & n5::srep01 has been successful
```

Snapshots

Snapshots provide point-in-time copies of Red Hat Gluster Storage volumes. Snaps are created as copy-on-write and take advantage of thin provisioning technology, allowing for near-instantaneous creation.

Volume snapshots rely on an orchestration of LVM snapshots across all bricks in a volume. As such, certain steps must be taken when creating bricks in order to enable snapshot functionality.

We explore the LVM configuration on our lab nodes to observe the minimum requirements. Note that a thin pool LV has been created as `rhgs_pool`, and then a thin LV `rhgs_lv` has been allocated a portion of the pool data (in this case, 100%). Because this is thin provisioning, the thin LV could be allocated any amount of space, even if it exceeds the thin pool space.

```
[root@n1 ~]# lvs
  LV        VG      Attr       LSize   Pool      Origin Data%  Meta%  Move Log Cpy%Sync
Convert
  root      rhel    -wi-ao----   4.40g
  swap      rhel    -wi-ao---- 512.00m
  rhgs_lv   rhgs_vg Vwi-aot---   8.00g rhgs_pool        0.81
  rhgs_pool rhgs_vg twi-aot---   8.00g                  0.81   0.02
```

A volume snapshot is created online and is non-disruptive to the clients.

```
[root@n1 ~]# gluster snapshot create snap01 rep01
snapshot create: success: Snap snap01_GMT-2015.06.09-19.08.34 created successfully
```

The new snapshot LVs are created for each brick and automatically mounted.

```
[root@n1 ~]# lvs
  LV                                VG      Attr       LSize   Pool      Origin  Data%
Meta%  Move Log Cpy%Sync Convert
  root                              rhel    -wi-ao----   4.40g
  swap                              rhel    -wi-ao---- 512.00m
  a9aa400404a8477b89823c51bbebeb91_0 rhgs_vg Vwi-aot---   8.00g rhgs_pool rhgs_lv 1.21
  rhgs_lv                           rhgs_vg Vwi-aot---   8.00g rhgs_pool         1.21
  rhgs_pool                         rhgs_vg twi-aot---   8.00g                  1.24
0.02
[root@n1 ~]# df -h | grep snaps
/dev/mapper/rhgs_vg-a9aa400404a8477b89823c51bbebeb91_0  8.0G  118M  7.9G   2%
/run/gluster/snaps/a9aa400404a8477b89823c51bbebeb91/brick1
```

Several administrative commands are available to give us information about the snapshots.

```
[root@n1 ~]# snap01=`gluster snapshot list rep01 | grep snap01`
[root@n1 ~]# gluster snapshot info $snap01
Snapshot                  : snap01_GMT-2015.06.11-23.06.26
Snap UUID                 : 86716030-c258-4643-9e22-53b646e62d6c
Created                   : 2015-06-11 23:06:26
Snap Volumes:

      Snap Volume Name          : a9aa400404a8477b89823c51bbebeb91
      Origin Volume name        : rep01
```

```
      Snaps taken for rep01      : 1
      Snaps available for rep01  : 255
      Status                     : Stopped

[root@n1 ~]# gluster snapshot status $snap01

Snap Name : snap01_GMT-2015.06.11-23.06.26
Snap UUID : 86716030-c258-4643-9e22-53b646e62d6c

      Brick Path        :
n1:/run/gluster/snaps/a9aa400404a8477b89823c51bbebeb91/brick1/rep01
      Volume Group      :   rhgs_vg
      Brick Running     :   No
      Brick PID         :   N/A
      Data Percentage   :   1.21
      LV Size           :   8.00g



      Brick Path        :
n2:/run/gluster/snaps/a9aa400404a8477b89823c51bbebeb91/brick2/rep01
      Volume Group      :   rhgs_vg
      Brick Running     :   No
      Brick PID         :   N/A
      Data Percentage   :   1.26
      LV Size           :   8.00g



      Brick Path        :
n3:/run/gluster/snaps/a9aa400404a8477b89823c51bbebeb91/brick3/rep01
      Volume Group      :   rhgs_vg
      Brick Running     :   No
      Brick PID         :   N/A
      Data Percentage   :   0.96
      LV Size           :   8.00g



      Brick Path        :
n4:/run/gluster/snaps/a9aa400404a8477b89823c51bbebeb91/brick4/rep01
      Volume Group      :   rhgs_vg
      Brick Running     :   No
      Brick PID         :   N/A
      Data Percentage   :   0.96
      LV Size           :   8.00g
```

Writing new files to the rep01 volume we can observe the copy-on-write functionality. Note that we have consumed additional *real* data blocks in the thin pool, and additional delta blocks in the thin LV. (You may need to wait a few moments before the changes appear in the lvs output.)

```
[root@n1 ~]# ~/snap1.sh
$ dd if=/dev/urandom of=/rhgs/client/rep01/newfile001 bs=1k count=1k
```

```
1024+0 records in
1024+0 records out
1048576 bytes (1.0 MB) copied, 0.14492 s, 7.2 MB/s
…
[root@n1 ~]# lvs
  LV                                   VG      Attr       LSize   Pool      Origin  Data%
Meta%  Move Log Cpy%Sync Convert
  root                                 rhel    -wi-ao----   4.40g
  swap                                 rhel    -wi-ao---- 512.00m
  a9aa400404a8477b89823c51bbebeb91_0 rhgs_vg Vwi-aot---   8.00g rhgs_pool rhgs_lv 1.21
  rhgs_lv                              rhgs_vg Vwi-aot---   8.00g rhgs_pool         1.52
  rhgs_pool                            rhgs_vg twi-aot---   8.00g                  1.60
0.02
```

We create a second snapshot to capture our new files.

```
[root@n1 ~]# gluster snapshot create snap02 rep01
snapshot create: success: Snap snap02_GMT-2015.06.09-19.46.17 created successfully
```

Deleting files from the volume, we see perhaps an unexpected effect – Data is not freed from the thin LV or pool.

```
[root@n1 ~]# ~/snap2.sh
$ rm -f /rhgs/client/rep01/newfile001
$ rm -f /rhgs/client/rep01/newfile002
$ rm -f /rhgs/client/rep01/newfile003
…
[root@n1 ~]# lvs
  LV                                   VG      Attr       LSize   Pool      Origin  Data%
Meta%  Move Log Cpy%Sync Convert
  root                                 rhel    -wi-ao----   4.40g
  swap                                 rhel    -wi-ao---- 512.00m
  6bc6d46eecca4123b1ff55a4afbbb07d_0 rhgs_vg Vwi-aot---   8.00g rhgs_pool rhgs_lv 1.52
  a9aa400404a8477b89823c51bbebeb91_0 rhgs_vg Vwi-aot---   8.00g rhgs_pool rhgs_lv 1.21
  rhgs_lv                              rhgs_vg Vwi-aot---   8.00g rhgs_pool         1.52
  rhgs_pool                            rhgs_vg twi-aot---   8.00g                  1.67
0.02
```

In order to reclaim the blocks, the `fstrim` command must be run. *Note that this would need to be run for all brick mount points in order to reclaim all space across the volume.*

```
[root@n1 ~]# fstrim -v /rhgs/bricks/rep01/
/rhgs/bricks/rep01/: 8 GiB (8520302592 bytes) trimmed
[root@n1 ~]# lvs
  LV                                   VG      Attr       LSize   Pool      Origin  Data%
Meta%  Move Log Cpy%Sync Convert
  root                                 rhel    -wi-ao----   4.40g
  swap                                 rhel    -wi-ao---- 512.00m
  6bc6d46eecca4123b1ff55a4afbbb07d_0 rhgs_vg Vwi-aot---   8.00g rhgs_pool rhgs_lv 1.52
  a9aa400404a8477b89823c51bbebeb91_0 rhgs_vg Vwi-aot---   8.00g rhgs_pool rhgs_lv 1.21
  rhgs_lv                              rhgs_vg Vwi-aot---   8.00g rhgs_pool         1.18
  rhgs_pool                            rhgs_vg twi-aot---   8.00g                  1.67
0.02
```

As an admin, we can mount the snapshot volume for read access to restore any needed data. Note the snap is mounted as read-only.

```
[root@n1 ~]# snap02=`gluster snapshot list rep01 | grep snap02`
[root@n1 ~]# gluster snapshot activate $snap02
Snapshot activate: snap02_GMT-2015.06.09-19.46.17: Snap activated successfully
[root@n1 ~]# mkdir /rhgs/client/snap02
[root@n1 ~]# mount -t glusterfs n1:/snaps/$snap02/rep01 /rhgs/client/snap02
[root@n1 ~]# mount | grep snap02
n1:/snaps/snap02_GMT-2015.06.09-19.46.17/rep01 on /rhgs/client/snap02 type fuse.glusterfs
(ro,relatime,user_id=0,group_id=0,default_permissions,allow_other,max_read=131072)
[root@n1 ~]# ls /rhgs/client/snap02/newfile001
/rhgs/client/snap02/newfile001
```

For a more convenient method to access snapshot data, we can enable the user-serviceable snapshot feature for the volume. Activated snapshots are available via a hidden .snaps directory on the client in each subdirectory of the snapshot volume.

```
[root@n1 ~]# gluster volume set rep01 features.uss enable
volume set: success
[root@n1 ~]# ls /rhgs/client/rep01/.snaps/$snap02/newfile001
/rhgs/client/rep01/.snaps/snap02_GMT-2015.06.09-19.46.17/newfile001
```

Disperse Volumes (Erasure Coding)

Erasure coding allows for data protection through dispersed redundancy across the bricks in a volume. It uses a configurable level of redundancy, relying on parity, similar to RAID 5/6, to rebuild data in the case of a loss.

Disperse volumes allow for a more efficient use of storage than replicate volumes, with potentially more redundancy. They also allow us to move data protection from a lower level in the storage stack up to the Gluster level, freeing us from proprietary RAID technology and allowing for further simplification of the commodity hardware stack.

When creating a disperse volume, we define the number of data (disperse) bricks and the number of parity (redundancy) bricks. The disperse count is the total number of data chunks that we will divide files into. The redundancy count is the number of the total data chunks we will treat as parity. Data can be reconstructed from (disperse count) – (redundancy count) bricks.

This type of volume does introduce additional overhead as parity calculations have to be made for reads and writes. It also breaks up individual file data, meaning that a whole file cannot be accessed via an offline brick.

Note that in our lab we will create 2 bricks on each of nodes n1 and n2 for our disperse volume. This allows us to create an optimal disperse volume, which cannot mathematically be accomplished with only 4 bricks. Because of this, we must append the `force` flag to the `volume create` command.

We create a 6/2 disperse volume. With this configuration, 6 bricks participate in the volume, and any 4 (6-2) of the bricks must be available in order to access the data. (The `mkecvol.sh` script has been provided for your convenience to avoid any typos in the long command.)

```
[root@n1 ~]# ~/mkecvol.sh
Creating ec01 volume...
$ gluster volume create ec01 disperse 6 redundancy 2 n1:/rhgs/bricks/ec01-1
n2:/rhgs/bricks/ec01-1 n3:/rhgs/bricks/ec01-1 n4:/rhgs/bricks/ec01-1
n1:/rhgs/bricks/ec01-2 n2:/rhgs/bricks/ec01-2 force
```

```
volume create: ec01: success: please start the volume to access data
[root@n1 ~]# gluster volume start ec01
volume start: ec01: success
[root@n1 ~]# gluster volume info ec01

Volume Name: ec01
Type: Disperse
Volume ID: f9f8d1d8-10d0-48cf-8292-a03860296b80
Status: Started
Number of Bricks: 1 x (4 + 2) = 6
Transport-type: tcp
Bricks:
Brick1: n1:/rhgs/bricks/ec01-1
Brick2: n2:/rhgs/bricks/ec01-1
Brick3: n3:/rhgs/bricks/ec01-1
Brick4: n4:/rhgs/bricks/ec01-1
Brick5: n1:/rhgs/bricks/ec01-2
Brick6: n2:/rhgs/bricks/ec01-2
Options Reconfigured:
performance.readdir-ahead: on
```

Examining the vol file for the ec01 volume, we can see better how the disperse volume is assembled in the translator stack.

```
[root@n1 ~]# cat /var/lib/glusterd/vols/ec01/ec01.tcp-fuse.vol
volume ec01-client-0
    type protocol/client
    option send-gids true
    option transport-type tcp
    option remote-subvolume /rhgs/bricks/ec01-1
    option remote-host n1
    option ping-timeout 42
end-volume

volume ec01-client-1
    type protocol/client
    option send-gids true
    option transport-type tcp
    option remote-subvolume /rhgs/bricks/ec01-1
    option remote-host n2
    option ping-timeout 42
end-volume

volume ec01-client-2
    type protocol/client
    option send-gids true
    option transport-type tcp
    option remote-subvolume /rhgs/bricks/ec01-1
    option remote-host n3
```

```
    option ping-timeout 42
end-volume

volume ec01-client-3
    type protocol/client
    option send-gids true
    option transport-type tcp
    option remote-subvolume /rhgs/bricks/ec01-1
    option remote-host n4
    option ping-timeout 42
end-volume

volume ec01-client-4
    type protocol/client
    option send-gids true
    option transport-type tcp
    option remote-subvolume /rhgs/bricks/ec01-2
    option remote-host n1
    option ping-timeout 42
end-volume

volume ec01-client-5
    type protocol/client
    option send-gids true
    option transport-type tcp
    option remote-subvolume /rhgs/bricks/ec01-2
    option remote-host n2
    option ping-timeout 42
end-volume

volume ec01-disperse-0
    type cluster/disperse
    option redundancy 2
    subvolumes ec01-client-0 ec01-client-1 ec01-client-2 ec01-client-3 ec01-client-4
ec01-client-5
end-volume

volume ec01-dht
    type cluster/distribute
    subvolumes ec01-disperse-0
end-volume

volume ec01-write-behind
    type performance/write-behind
    subvolumes ec01-dht
end-volume
```

```
volume ec01-read-ahead
    type performance/read-ahead
    subvolumes ec01-write-behind
end-volume

volume ec01-readdir-ahead
    type performance/readdir-ahead
    subvolumes ec01-read-ahead
end-volume

volume ec01-io-cache
    type performance/io-cache
    subvolumes ec01-readdir-ahead
end-volume

volume ec01-quick-read
    type performance/quick-read
    subvolumes ec01-io-cache
end-volume

volume ec01-open-behind
    type performance/open-behind
    subvolumes ec01-quick-read
end-volume

volume ec01-md-cache
    type performance/md-cache
    subvolumes ec01-open-behind
end-volume

volume ec01
    type debug/io-stats
    option count-fop-hits off
    option latency-measurement off
    subvolumes ec01-md-cache
end-volume
```

We mount ec01 volume via the native client, write some data, and observe the data placement.

```
[root@n1 ~]# ~/ec1.sh
Creating client mountpoint...
$ mkdir -p /rhgs/client/ec01
Mounting ec01 volume...
$ mount -t glusterfs n1:ec01 /rhgs/client/ec01
Writing plain-text data to file dirs.txt
[root@n1 ~]# file /rhgs/client/ec01/dirs.txt
/rhgs/client/ec01/dirs.txt: ASCII text
[root@n1 ~]# head /rhgs/client/ec01/dirs.txt
```

```
/
/boot
/boot/grub2
/boot/grub2/themes
/boot/grub2/themes/system
/boot/grub2/i386-pc
/boot/grub2/locale
/boot/grub2/fonts
/dev
[root@n1 ~]# file /rhgs/bricks/ec01-1/dirs.txt
/rhgs/bricks/ec01-1/dirs.txt: data
```

Removing access to two of the bricks, we see that the client can still read the data.

```
[root@n1 ~]# ~/ec2.sh
Blocking access to n3 and n4 with iptables...
$ iptables -F
$ iptables -A OUTPUT -d n3 -j DROP
$ iptables -A OUTPUT -d n4 -j DROP
[root@n1 ~]# head /rhgs/client/ec01/dirs.txt
/
/boot
/boot/grub2
/boot/grub2/themes
/boot/grub2/themes/system
/boot/grub2/i386-pc
/boot/grub2/locale
/boot/grub2/fonts
/dev
[root@n1 ~]# grep disperse /var/log/glusterfs/rhgs-client-ec01.log | tail -2
[2015-06-12 16:57:33.021348] W [ec-common.c:403:ec_child_select] 0-ec01-disperse-0:
Executing operation with some subvolumes unavailable (4)
[2015-06-12 16:57:33.025359] W [ec-common.c:121:ec_heal_report] 0-ec01-disperse-0: Heal
failed (error 107)
```

Writing additional data to the file, we observe the pending heal.

```
[root@n1 ~]# echo "new data" >> /rhgs/client/ec01/dirs.txt
[root@n1 ~]# gluster volume heal ec01 info
Brick n1:/rhgs/bricks/ec01-1/
/dirs.txt
Number of entries: 1

Brick n2:/rhgs/bricks/ec01-1/
/dirs.txt
Number of entries: 1

Brick n3:/rhgs/bricks/ec01-1
Status: Transport endpoint is not connected

Brick n4:/rhgs/bricks/ec01-1
```

```
Status: Transport endpoint is not connected

Brick n1:/rhgs/bricks/ec01-2/
/dirs.txt
Number of entries: 1

Brick n2:/rhgs/bricks/ec01-2/
/dirs.txt
Number of entries: 1
[root@n1 ~]# getfattr -d -m . -e hex /rhgs/bricks/ec01-1/dirs.txt
getfattr: Removing leading '/' from absolute path names
# file: rhgs/bricks/ec01-1/dirs.txt
trusted.bit-rot.version=0x020000000000000557afa920002b7eb
trusted.ec.config=0x0000080602000200
trusted.ec.dirty=0x00000000000000020000000000000002
trusted.ec.size=0x000000000006c369
trusted.ec.version=0x00000000000000330000000000000035
trusted.gfid=0x1ab0e229ec8548f8bc08dcb7c3874408
```

Restoring the network connections and touching the file, we observe that the heal is triggered and the file is no longer dirty.

```
[root@n1 ~]# ./ec3.sh
Flushing iptables rules...
$ iptables -F
Restarting gluster services due to BZ 1231334...
$ pkill glusterfs
$ pkill glusterfsd
$ mount -a
$ service glusterd restart
Redirecting to /bin/systemctl restart  glusterd.service
$ mount -t glusterfs n1:ec01 /rhgs/client/ec01
[root@n1 ~]# file /rhgs/client/ec01/dirs.txt
/rhgs/client/ec01/dirs.txt: ASCII text
[root@n1 ~]# getfattr -d -m . -e hex /rhgs/bricks/ec01-1/dirs.txt
getfattr: Removing leading '/' from absolute path names
# file: rhgs/bricks/ec01-1/dirs.txt
trusted.bit-rot.version=0x020000000000000557b1a64000b0287
trusted.ec.config=0x0000080602000200
trusted.ec.dirty=0x00000000000000000000000000000000
trusted.ec.size=0x000000000006c4dd
trusted.ec.version=0x00000000000000550000000000000055
trusted.gfid=0x60d0c42e23a44dd99076ddfbde525e8e
```