Split-Brain Management in Red Hat Storage

Author: Wesley Duffee-Braun, Technical Account Manager
Technical Reviewer: Dustin Black, Senior Technical Account Manager
Editor: Chris Negus
11/20/2013

WHAT IS SPLIT-BRAIN

The term *split-brain* refers to a scenario when two (or more) clustered servers that provide a single service think that each is the correct owner of a shared resource. This setting can cause confusion and instability in a clustered environment.

As with any clustered software product, stability within Red Hat Storage environments depends on accurate and reliable communication between nodes. When the environment becomes unstable or when a Red Hat Storage server goes off-line, there is the possibility of a split-brain scenario upon recovery. In this scenario, it is important to heal the files affected or clients will not be able to operate correctly, and widespread filesystem corruption can occur.

Using this tech brief, you will learn how to recognize when a Red Hat Storage environment is in a split-brain mode and how to recover from that situation.

RECOGNIZING A SPLIT-BRAIN SCENARIO

There are three common ways to check if a Red Hat Storage deployment has gone into a split-brain mode. You can:

- Run filesystem commands on a Red Hat Storage client and see that it outputs error messages
- Check logs and see that they reflect the split-brain condition
- Run `gluster` commands to determine if you have a split-brain condition

Client Errors

When attempting to access a mounted volume that has become split-brain, clients may encounter errors similar to the following when accessing affected files or directories.

```
[user@system clientDirectory]$ ls -l
ls: cannot access application.log: Input/output error
total 4
drwxrwsr-x. 75 root      cad  4096 Mar 23 15:12 apps
-rw-------.  1 root      root 1359 Sep 28  2011 installDetails
??????????  ? ?         ?       ?            ? application.log
drwxr-x---. 10 root      cad   131 Feb 22 12:08 sw
```

This will often occur when one of the nodes in the split-brain is offline. At that point, waiting for the node to come back online is the best course. Once the node is back online, the self-heal process should activate and the files will become available.

If, however, the node comes back online and the errors above persist, the next step to is examine the logs as shown in the Client Logs section and proceed with a manual intervention.

Errors in Logs

There are errors indicating split-brain scenarios that appear in Red Hat Storage client and server logs.

Server Logs

There are errors and warnings that will be logged when a node goes offline. These are important to note as they are often a precursor to a split-brain scenario. Log messages differ depending on who is considered "online" and who is "offline." Below is an excerpt from the `/var/log/glusterfs/glustershd.log` "online" server (`192.168.200.101`)

```
I [client.c:2103:client_rpc_notify] 0-repvol1-client-5: disconnected from
192.168.200.102:24007. Client process will keep trying to connect to
glusterd until brick's port is available.

E [afr-common.c:3832:afr_notify] 0-repvol1-replicate-2: All subvolumes are
down. Going offline until atleast one of them comes back up.
```

And here is what appears in the "offline" server (`192.168.200.102`) at the same time:

```
E [socket.c:2158:socket_connect_finish] 0-repvol1-client-5: connection to
192.168.200.101:24007 failed (Connection timed out)
```

Client Logs

Below are warnings that the client will log when there is a split-brain condition on the volume. The client can usually continue writing to the volume, unless all the nodes are offline.

```
W [socket.c:522:__socket_rwv] 0-repvol1-client-1: readv on
192.168.200.102:49155 failed (Connection refused)

I [client.c:2103:client_rpc_notify] 0-repvol1-client-1: disconnected from
192.168.200.102:49155. Client process will keep trying to connect to
glusterd until brick's port is available.

W [socket.c:522:__socket_rwv] 0-repvol1-client-5: readv on
192.168.200.102:49156 failed (Connection refused)

I [client.c:2103:client_rpc_notify] 0-repvol1-client-5: disconnected from
192.168.200.102:49156. Client process will keep trying to connect to
glusterd until brick's port is available.
```

Server Commands

The command `gluster volume heal <VOLNAME> info` can be run to check for a split-brain condition.

```
# gluster volume heal repvol1 info
Gathering list of entries to be healed on volume repvol1 has been successful
Brick n1:/gluster/bricks/rep1
Number of entries: 2
/
/mydir
```

```
Brick n2:/gluster/bricks/rep1
Number of entries: 0
```

This indicates that there exist two entries on `n1:/gluster/bricks/rep1` that will need to be self-healed on other servers when they are available. In other words, the `n1:/gluster/bricks/rep1` brick contains data that other bricks do not have. The `n2:/gluster/bricks/rep1` brick does not have any files that need to be propagated to other bricks, thus its `Number of entries` is `0`. If there is not a split-brain condition for the volume, then the `Number of entries` value will be `0` for *all* bricks.

FIXING A SPLIT-BRAIN SCENARIO

`Red Hat Storage` servers (as of `Red Hat Storage 2.0`) will actively try to self-heal split-brain scenarios. However, that is not always possible and manual intervention can be required. The steps needing to heal the scenario sometimes can be as easy as bringing a server back online or can be complex, including deciding what version of a file is the “correct” version and propagating that file to other bricks.

Automatic Self Healing of Split-Brain

There are two ways of checking if an automatic self-heal has succeeded. The first is to run the same `gluster volume heal <VOLNAME> info` command. If all the results for `Number of entries` is `0`, where there had previously been entries listed, then the heal was successful.

The other way is to examine the logs for the volume bricks. Below is a snippet from the `/var/log/glustershd.log` on the previously “offline” node

```
I [afr-self-heald.c:1180:afr_dir_exclusive_crawl] 0-repvol1-replicate-0:
Another crawl is in progress for repvol1-client-1
I [afr-self-heald.c:1180:afr_dir_exclusive_crawl] 0-repvol1-replicate-0:
Another crawl is in progress for repvol1-client-1
```

and here is the same log on the “online” node

```
I [afr-self-heal-common.c:2843:afr_log_self_heal_completion_status] 0-
repvol1-replicate-2:  entry self heal  is successfully completed, on
<gfid:00000000-0000-0000-0000-000000000001>
I [afr-self-heal-common.c:2843:afr_log_self_heal_completion_status] 0-
repvol1-replicate-0:  entry self heal  is successfully completed, on
<gfid:00000000-0000-0000-0000-000000000001>

I [afr-self-heal-common.c:2843:afr_log_self_heal_completion_status] 0-
repvol1-replicate-2:  metadata self heal  is successfully completed, entry
self heal  is successfully completed, on <gfid:6e856d23-d864-46c7-b0d0-
0c57f4043caf>
I [afr-self-heal-common.c:2843:afr_log_self_heal_completion_status] 0-
repvol1-replicate-0:  metadata self heal  is successfully completed, entry
self heal  is successfully completed, on <gfid:6e856d23-d864-46c7-b0d0-
0c57f4043caf>
```

Manual Healing of Split-Brain Scenarios

There are situations where the automatic self-healing does not work. This is often indicated when the

`gluster volume heal <VOLNAME> info` returns the same file as needing a heal on multiple bricks, as in this example below:

```
# gluster volume heal repvol1 info
Gathering list of entries to be healed on volume repvol1 has been successful

Brick n1:/gluster/bricks/rep1
Number of entries: 2
/
/textFile

Brick n2:/gluster/bricks/rep1
Number of entries: 2
/
/textFile
```

In the example above, the `textFile` file is out of sync on both server `n1` and `n2`. When all gluster daemon communication is re-established, it is likely that `textFile` will have to be healed manually. Below are the server and client logs that indicate that the automatic self-heal has failed for `textFile`.

Server Logs Indicating Files that Cannot be Self-Healed

On `n1`, below is a snippet from `/var/log/glusterfs/glustershd.log`

```
E [afr-self-heal-common.c:1555:afr_sh_common_lookup_cbk] 0-repvol1-
replicate-0: Conflicting entries for <gfid:00000000-0000-0000-0000-
000000000001>/textFile

W [client-rpc-fops.c:1529:client3_3_inodelk_cbk] 0-repvol1-client-1: remote
operation failed: No such file or directory

W [client-rpc-fops.c:464:client3_3_open_cbk] 0-repvol1-client-1: remote
operation failed: No such file or directory. Path: <gfid:d0b84987-f6d4-44b6-
8902-792e01a35449> (d0b84987-f6d4-44b6-8902-792e01a35449)

E [afr-self-heal-data.c:1453:afr_sh_data_open_cbk] 0-repvol1-replicate-0:
open of <gfid:d0b84987-f6d4-44b6-8902-792e01a35449> failed on child repvol1-
client-1 (No such file or directory)
```

and here are the logs from `n2`

```
E [afr-self-heal-common.c:1555:afr_sh_common_lookup_cbk] 0-repvol1-
replicate-0: Conflicting entries for <gfid:00000000-0000-0000-0000-
000000000001>/textFile

W [client-rpc-fops.c:1529:client3_3_inodelk_cbk] 0-repvol1-client-0: remote
operation failed: No such file or directory

W [client-rpc-fops.c:464:client3_3_open_cbk] 0-repvol1-client-0: remote
operation failed: No such file or directory. Path: <gfid:fcbc2342-a649-44f6-
9a55-f3c6d7ae8477> (fcbc2342-a649-44f6-9a55-f3c6d7ae8477)

E [afr-self-heal-data.c:1453:afr_sh_data_open_cbk] 0-repvol1-replicate-0:
```

```
open of <gfid:fcbc2342-a649-44f6-9a55-f3c6d7ae8477> failed on child repvol1-
client-0 (No such file or directory)
```

We know that the `textFile` file exists, but the gluster daemons are having trouble opening the file. Because no one server is able to definitively say that their version of textFile is the correct version, we must fix this by hand. However, we will examine what is seen from the client side as well before the manual heal.

Client Logs Indicating Files that Cannot be Self-Healed

When looking at client logs, you may see errors similar to the following in a split-brain scenario that needs manual intervention for healing.

```
E [afr-self-heal-common.c:1555:afr_sh_common_lookup_cbk] 0-repvol1-replicate-
0: Conflicting entries for /textFile
E [afr-self-heal-common.c:2843:afr_log_self_heal_completion_status] 0-
repvol1-replicate-0:  entry self heal  failed, on /
```

The exact same entries are seen on all clients that are attempting to access the file `textFile`.

Manually Healing Files Via the Command Line

There are some commands you can try that will attempt to force the self-heal mechanism to work on files that were not healed automatically. These invoke the `gluster volume heal <VOLNAME>` command.

For example, to trigger self-heal on files on `test-volume` that had been previously marked as needing a self-heal:

```
# gluster volume heal test-volume
Heal operation on volume test-volume has been successful
```

To trigger self-heal on all the files of of `test-volume` (regardless if they had been marked as needing a self-heal previously).

```
# gluster volume heal test-volume full
Heal operation on volume test-volume has been successful
```

However, when those operations fail to resolve the issue, you must then fix each file with a manual process. You must examine, on each brick, the file in question. Then you must decide which file to keep and which file(s) to delete. Once you have decided which brick contains the good file and which brick(s) contain the file(s) to remove, continue with the steps below.

Get the `trusted.gfid` extended attribute from the affected files:

```
# getfattr -h -d -m trusted.gfid -e hex /bricks/replicate/path/to/file
trusted.gfid=0x42e989d0c2694d6fad4204b3071a878a
```

Note that you need to lookup the`trusted.gfid` for the file on each brick of the replica. It is possible that the `trusted.gfid` of one file is different on each brick.

Delete the wrong/old files from the bricks, keeping the good copies untouched.

```
# rm -f /bricks/replicate/path/to/file
```

Delete the link-file from the `.glusterfs` directory on the affected bricks, basing the filename on the `trusted.gfid` attribute:

```
# rm -f /bricks/replicate/.glusterfs/42/e9/42e989d0-c269-4d6f-ad42-04b3071a878a
```

Perform a targeted self-heal on the files that were affected by the split-brain:

- For single files do a stat from a client for the mounted volume:

  ```
  # stat /mnt/replicate-vol/path/to/file
  ```

- Or follow the steps concerning the `gluster volume heal test-volume full` command for a total volume heal.

Repeat the process as necessary for all files previously logged as needing a heal.

RECAP AND REVIEW

Split-brain can occur in any replica volume, and can be detrimental to an environment by limiting the availability of files to clients. Although the gluster daemons work to automatically self-heal, there are scenarios where files must be healed by hand. By identifying which file is the “correct” version, a system administrator can selectively remove the “bad” versions and continue.

For additional information, please visit the following Solutions and Administration Guide sections:

- **How to recover a file from a split-brain on a Red Hat Storage volume** (https://access.redhat.com/site/solutions/193843)
- **'ls -l' from glusterfs client gives Input/output error when one node of 2 node RHS is down** (https://access.redhat.com/site/solutions/350793)
- **Red Hat Storage Administration Guide: Triggering Self-Heal on Replicate** (https://access.redhat.com/site/documentation/en-US/Red_Hat_Storage/2.1/html-single/Administration_Guide/index.html#sect-User_Guide-Managing_Volumes-Self_heal)