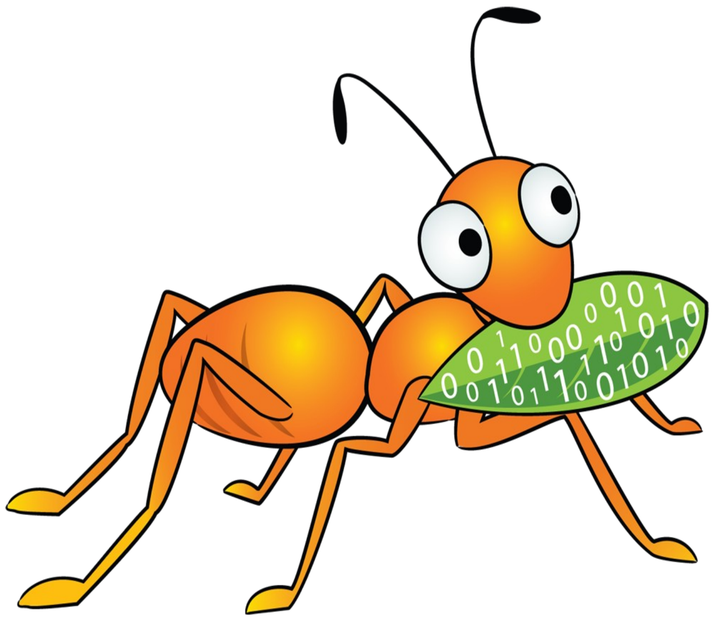
GlusterFS: Arbiter based replication

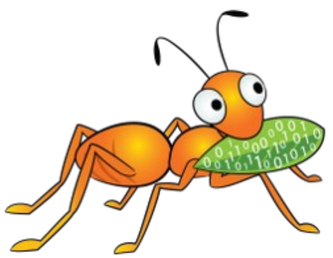
Without 3x storage cost + zero split-brains!



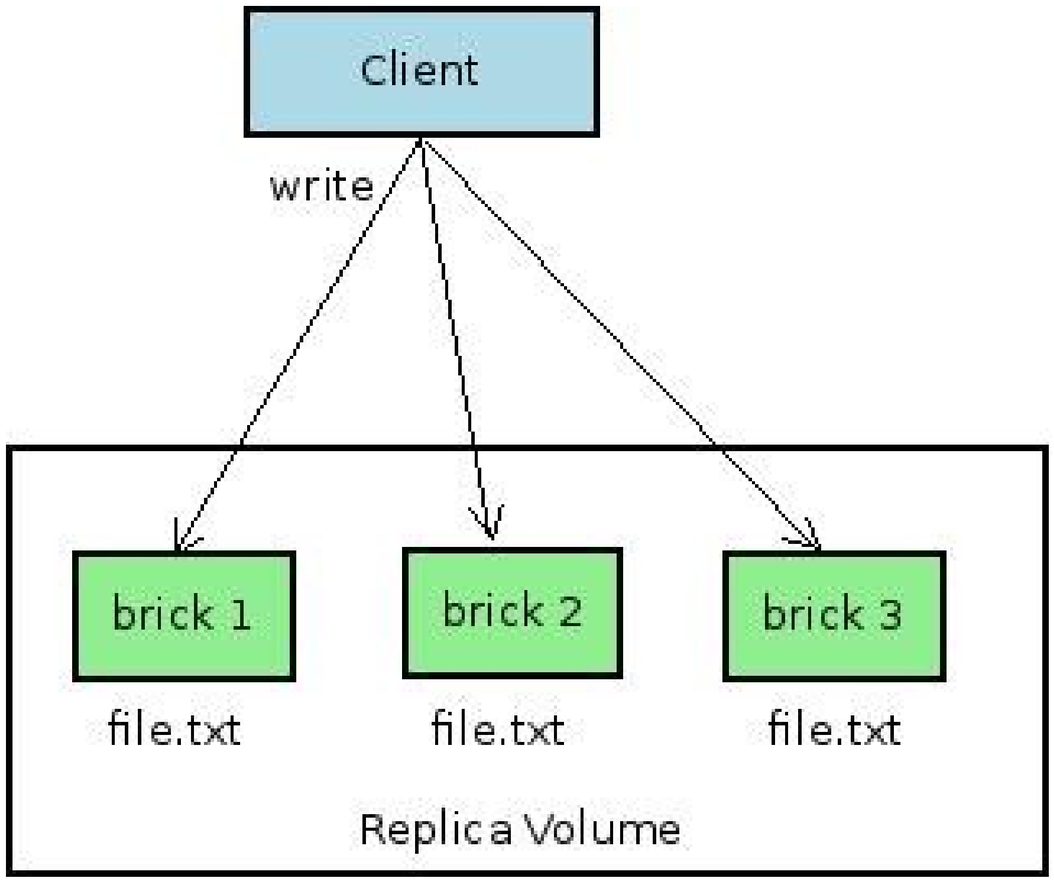
Ravishankar N.
Software Engineer, Red Hat
April 20th, VAULT- 2016

Agenda

- Introduction to replicate (AFR) volumes in gluster
- Split-brains in replica volumes
- Client-quorum to the rescue
- Arbiter volumes
- Volume creation
- How they work
- Brick sizing and placement strategies
- Monitoring and troubleshooting

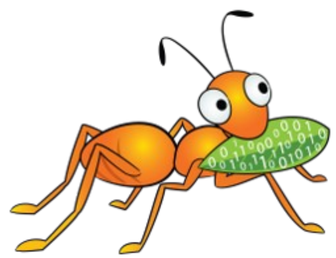


Introduction to replicate (AFR) volumes in gluster

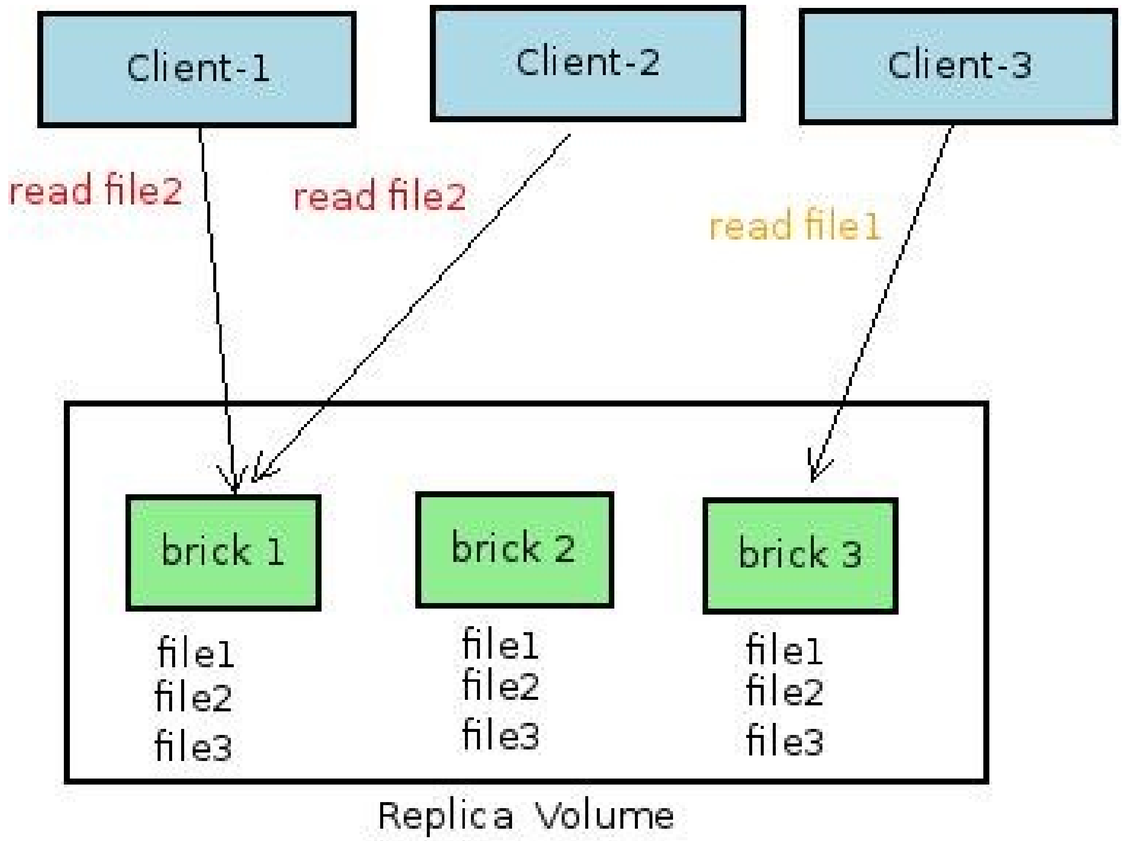


- Writes are synchronous- i.e. sent to all bricks of the replica.
- Follows a transaction model under locks – non-blocking locks that degrade to blocking.
- Uses extended attributes to mark failures:

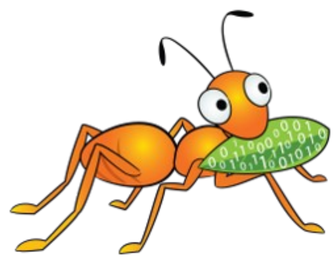
```
#getfattr -d -m . -e hex /brick1/file.txt
getfattr: Removing leading '/' from absolute path
names
# file: bricks/brick1/file.txt
trusted.afr.dirty=0x000000000000000000000000
trusted.afr.testvol-client-
2=0x000000020000000000000000
trusted.gfid=0xde0d499090a144ffa03841b7b317d052
```



(...cont'd) Introduction to replicate (AFR) volumes in gluster



- Reads - served from one brick (the 'read-subvolume') of the replica.
- The read-subvolume is a function of hash(file name) -but is also configurable via policies.



(...cont'd) Introduction to replicate (AFR) volumes in gluster

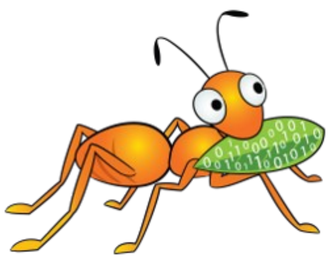
Self-heal

- GFIDs of files that need heal are stored inside `.glusterfs/indices` folder of the bricks.
- Self-heal daemon crawls this folder periodically:

  - For each GFID encountered, it fetches the xattrs of the file, finds out the source(s) and sink(s), does the heal.

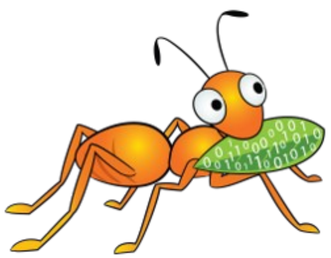
- Manual launching of heal – via gluster CLI:

```
#gluster volume heal <volname>
```



Split-brains in replica volumes

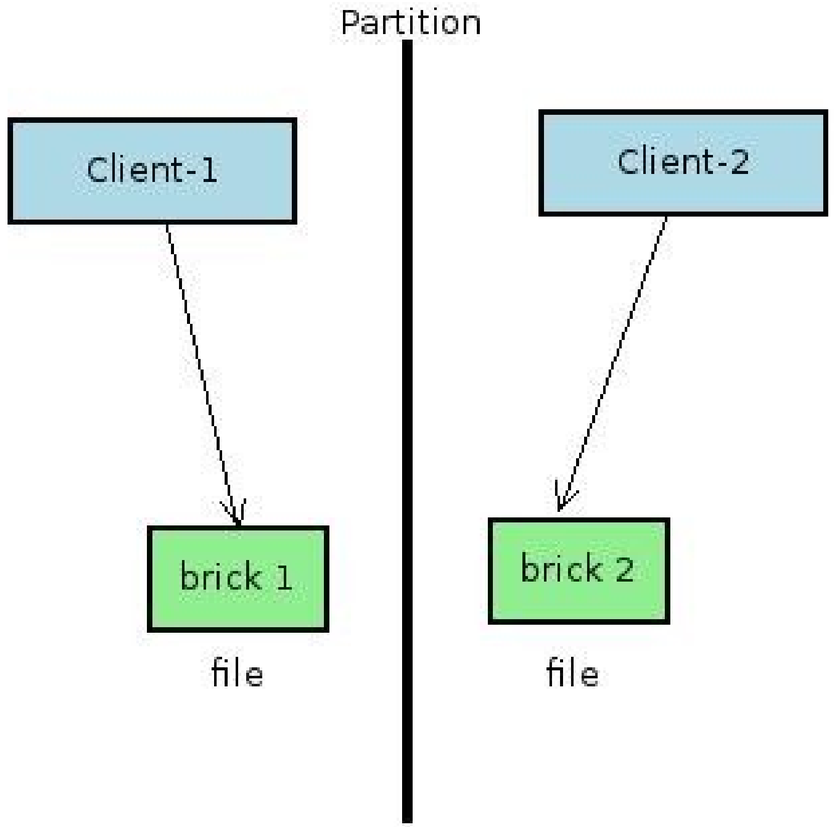
- What is split-brain?
  - Difference in file data/ metadata across the bricks of a replica.
  - Cannot identify which brick holds the good copy, even when all bricks are available.
  - Each brick accuses the other of needing heal.
  - All modification FOPs fail with Input/Output Error (EIO)

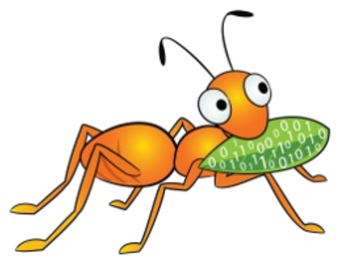


Split-brains in replica volumes

- How does it happen?
  - Split-brain in space

Two clients write to different replicas of the file leading to inconsistencies.

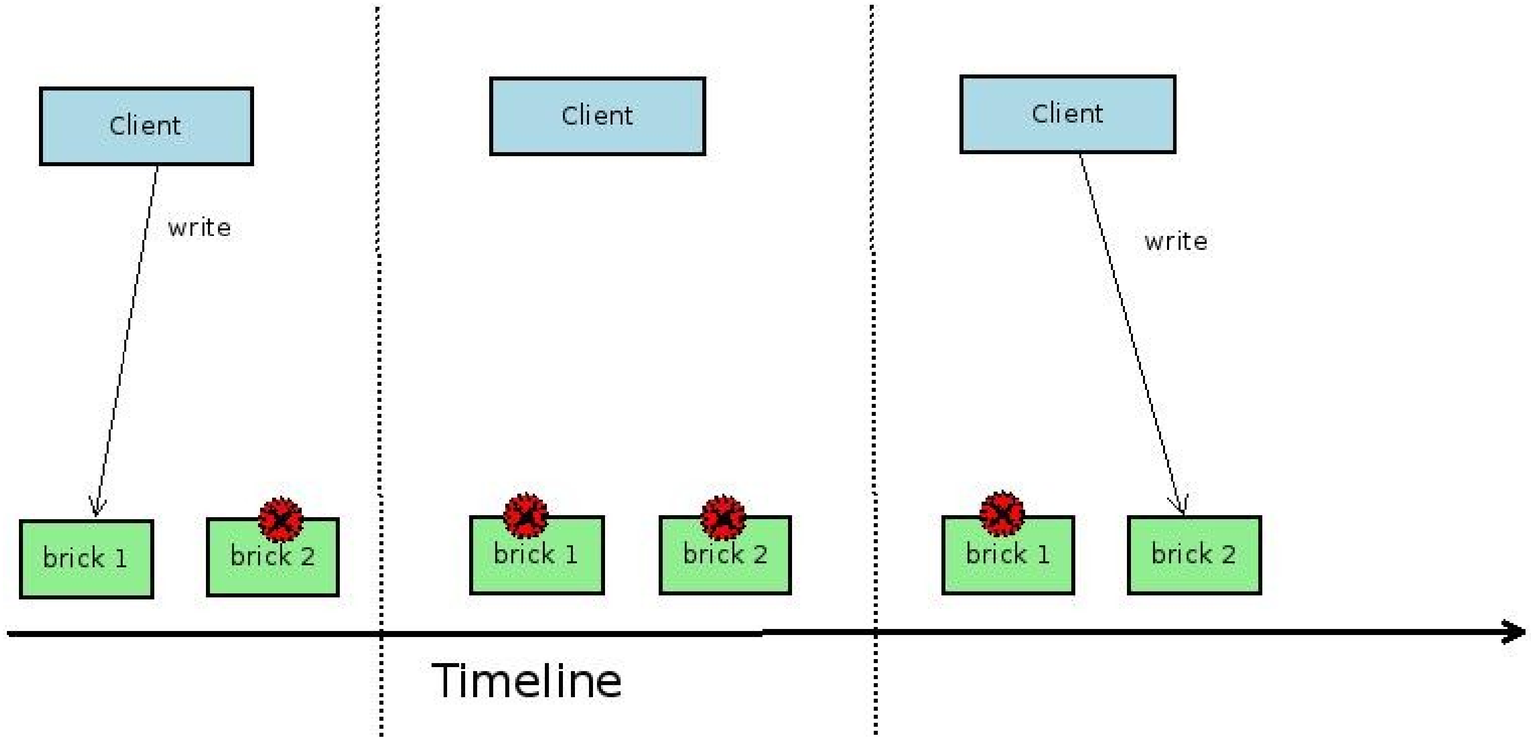


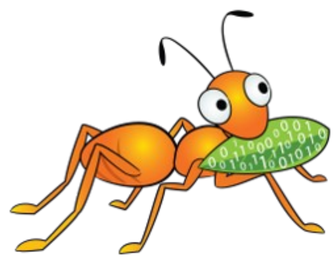


(...cont'd) Split-brains in replica volumes

- How does it happen?
  - Split-brain in time

The same client writes to different replicas of the file at different times before the healing begins.

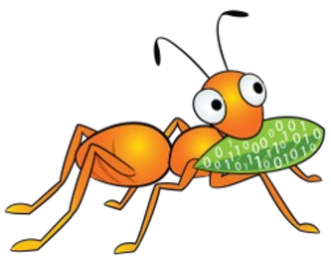




(...cont'd) Split-brains in replica volumes

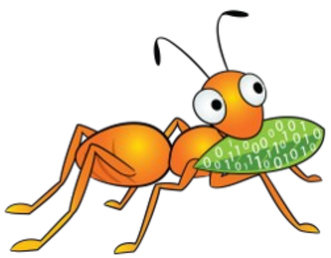
- How do you prevent it?
  - Theoretically speaking, use robust networks with zero {disconnects, packet losses, hardware issues.}

    But if wishes were horses...



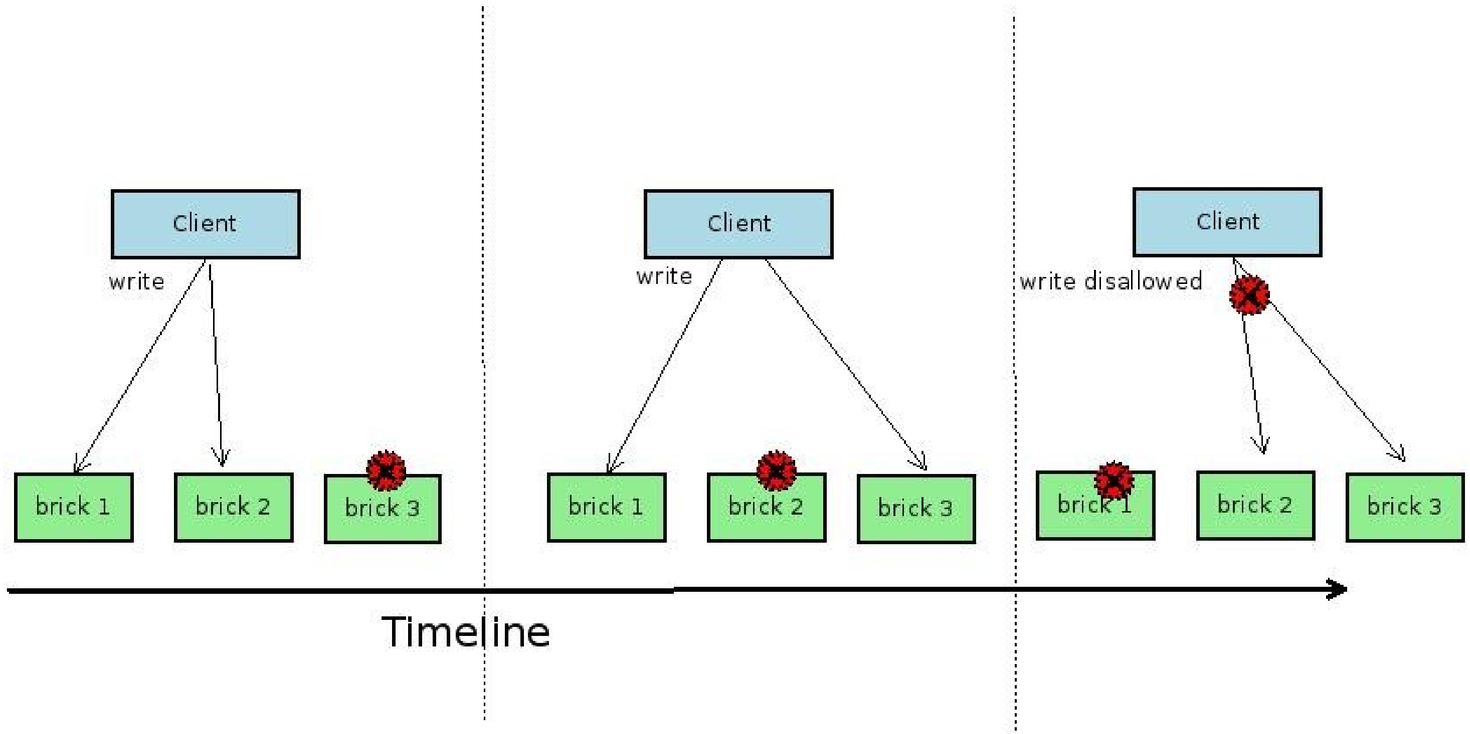
Client-quorum to the rescue

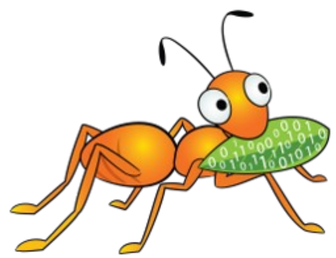
- quorum \ˈkwȯr-əm\ (noun): the smallest number of people who must be present at a meeting in order for decisions to be made.
- In glusterFS parlance: Minimum number of bricks that need to be up to allow modifications.
- What does this mean for replica-2?
  - The client need to be connected to both bricks at all times.
  - Fault tolerance = No down bricks. i.e. no high availability



(...cont'd) Client-quorum to the rescue

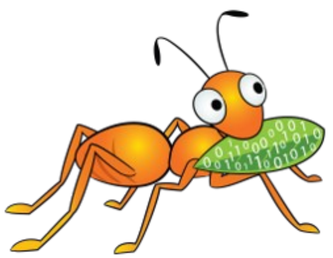
- How does replica-3 solve the problem?
  - Uses client quorum in 'auto' mode - i.e. 2 bricks need to be up to allow modifications.
  - In other words, fault tolerance = 1 down brick





(...cont'd) Client-quorum to the rescue

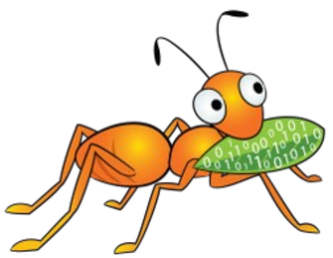
- Network split-brain – A corner case in replica-3 volumes.
  - 3 Clients simultaneously write to non-overlapping (offset + range):
    C1 succeeds on B1, B2
    C2 succeeds on B2, B3
    C3 succeeds on B1, B3



(...cont'd) Client-quorum to the rescue

- Key point to understand - to prevent split-brain, there will always be a case when you need to block FOPS, no matter how many replicas are used, if the only true copy is unavailable.

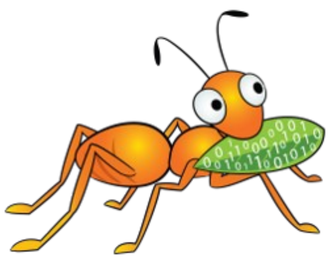
- What extra replicas give you is to increase in chance that more than one brick hosts the true copy (i.e. the copy has witnessed all writes without any failures)



Arbiter volumes

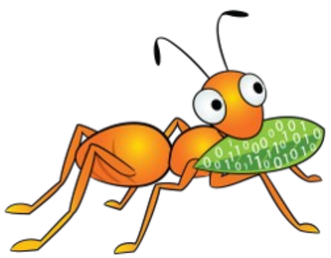
- A replica-3 volume where the 3rd brick only stores file metadata and the directory hierarchy.

```
0:root@vm1 ~$
0:root@vm1 ~$ ls -l /bricks/brick1/*
-rw-r--r--. 2 root root  813568 Apr 20 15:51 /bricks/brick1/file1
-rw-r--r--. 2 root root 2145792 Apr 20 15:51 /bricks/brick1/file2
-rw-r--r--. 2 root root 2797568 Apr 20 15:51 /bricks/brick1/file3
0:root@vm1 ~$
0:root@vm1 ~$
0:root@vm1 ~$ ls -l /bricks/brick2/*
-rw-r--r--. 2 root root  813568 Apr 20 15:51 /bricks/brick2/file1
-rw-r--r--. 2 root root 2145792 Apr 20 15:51 /bricks/brick2/file2
-rw-r--r--. 2 root root 2797568 Apr 20 15:51 /bricks/brick2/file3
0:root@vm1 ~$
0:root@vm1 ~$ ls -l /bricks/brick3/*
-rw-r--r--. 2 root root 0 Apr 20 15:51 /bricks/brick3/file1
-rw-r--r--. 2 root root 0 Apr 20 15:51 /bricks/brick3/file2
-rw-r--r--. 2 root root 0 Apr 20 15:51 /bricks/brick3/file3
0:root@vm1 ~$
```



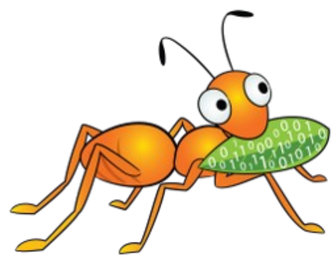
(...cont'd) Arbiter volumes

- Consumes much less than 3x space - But *how much* space? We'll see later.
- Provides the same level of consistency (not availability) as replica-3 volumes
  - i.e. it allows a fault tolerance of 1 down brick.
- Perfect sweet spot between 2 way and 3 way replica.



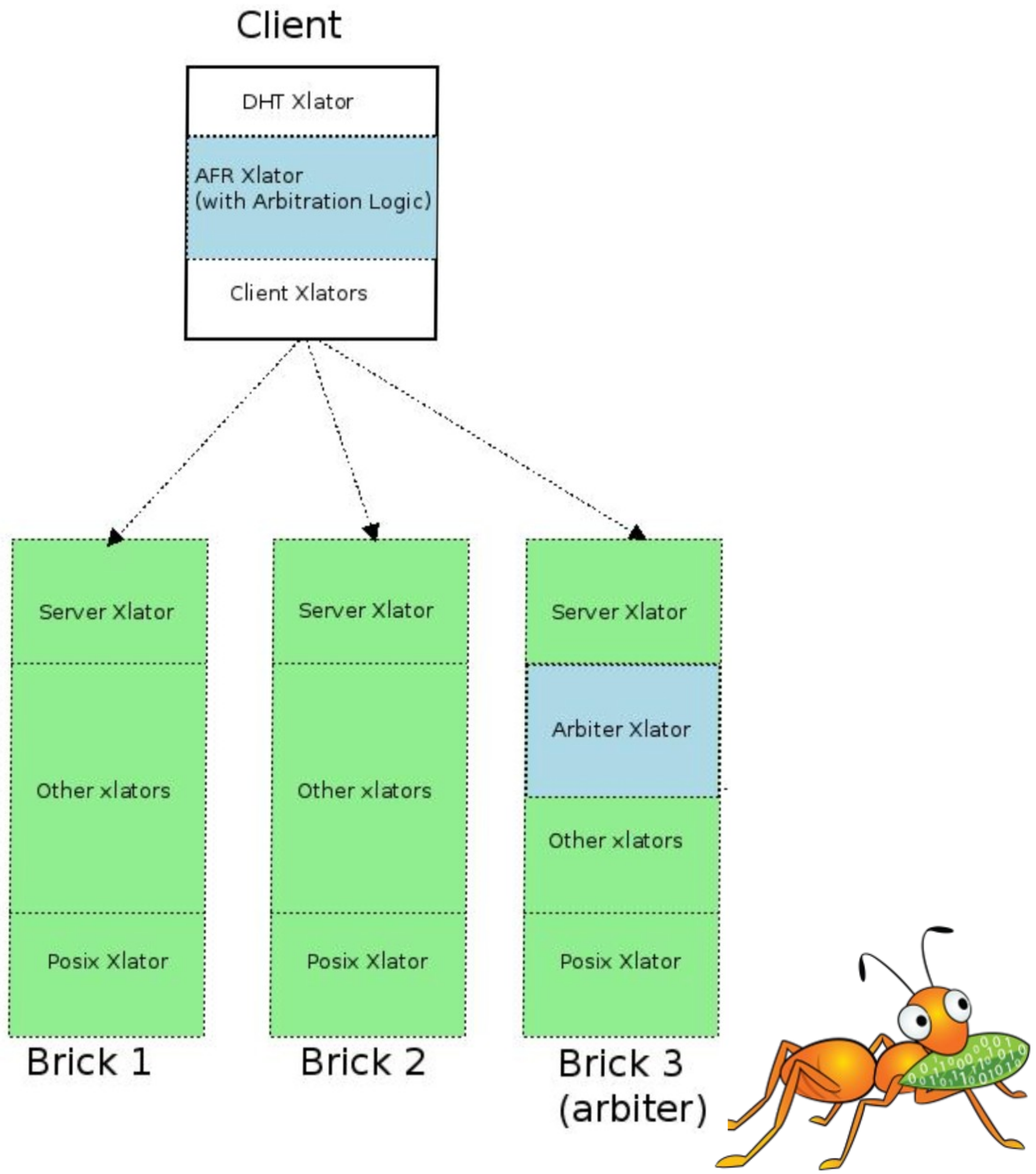
Creating Arbiter volumes

```
#gluster volume create testvol replica 3 arbiter 1 host1:brick1 host2:brick2 host3:brick3

#gluster volume info testvol

Volume Name: testvol

Type: Distributed-Replicate

Volume ID: ae6c4162-38c2-4368-ae5d-6bad141a4119

Status: Created

Number of Bricks: 2 x (2 + 1) = 6

Transport-type: tcp

Bricks:

Brick1: host1:/bricks/brick1

Brick2: host2:/bricks/brick2

Brick3: host3:/bricks/brick3 (arbiter)

Brick4: host4:/bricks/brick4

Brick5: host5:/bricks/brick5

Brick6: host6:/bricks/brick6 (arbiter)
```

Arbiter volumes- How do they work?

- The arbiter translator on (every 3rd) brick process.
- The arbitration logic in AFR in the client process.



Arbiter volumes- How do they work?

Arbitration logic:

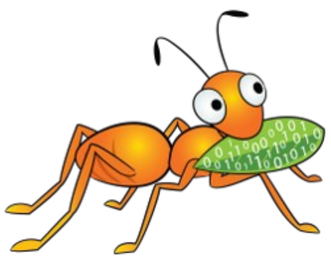
- Take full locks for writes.
- Decide go/no-go *before* sending the write to the bricks.
- Decide to return success/failure *after* we get the responses from the bricks.

Scenarios:

a)all 3 bricks (including arbiter) are up → allow writes

b)2 bricks are up, arbiter is down →allow writes

c)1 brick and arbiter are up → allow writes IFF the arbiter doesn't blame the up brick.



Arbiter volumes- How do they work?

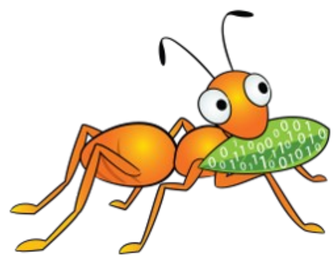
- Self-heal mechanism

  - Arbiter can serve as a source for metadata (but not data)

- What about performance?

  -Single writer - No penalty.

  -Multi writer - A little overhead due to full file locks.

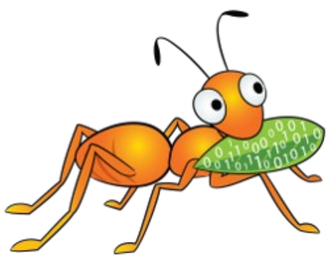


Brick sizing and placement

- What does each arbiter brick store?
  - Directory hierarchy + xattrs
  - The .glusterfs folder
- Safer side estimate is 4KB/file x no. of files

  Practical estimates by community users: 1KB/file x no. of files

  https://gist.github.com/pfactum/e8265ca07f7b19f30bb3
- Placement strategies
  - Low spec machine for holding arbiter bricks of all replicas.
  - Daisy chaining: Place arbiter bricks of different replicas in different nodes that host the data bricks.



Monitoring volumes

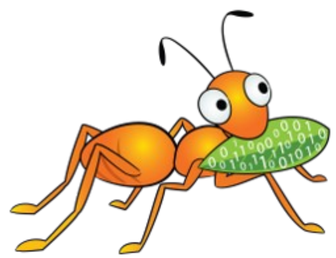
- ENOTCONN messages in client logs

  Need not necessarily mean a lost connection. Can be due to the arbitration logic to prevent split-brains.

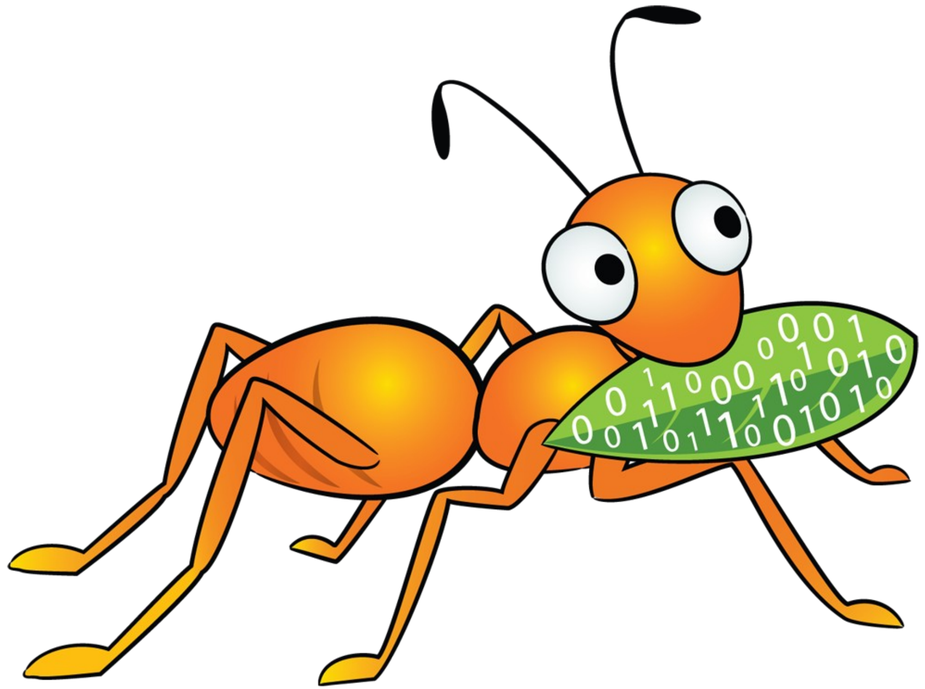
- `#gluster volume heal volname info` -> Monitor heals
- `#gluster volume heal volname info split-brain`

  -->Must always shows zero entries (Why? Because Arbiter volumes duh!)

  --> But it doesn't? Report a bug!



Questions/ comments?



Thank you!