GLUSTERD-2.0

The newer and better way to manage GlusterFS

Kaushal (kshlm/kshlmster)
GlusterD Maintainer

AGENDA

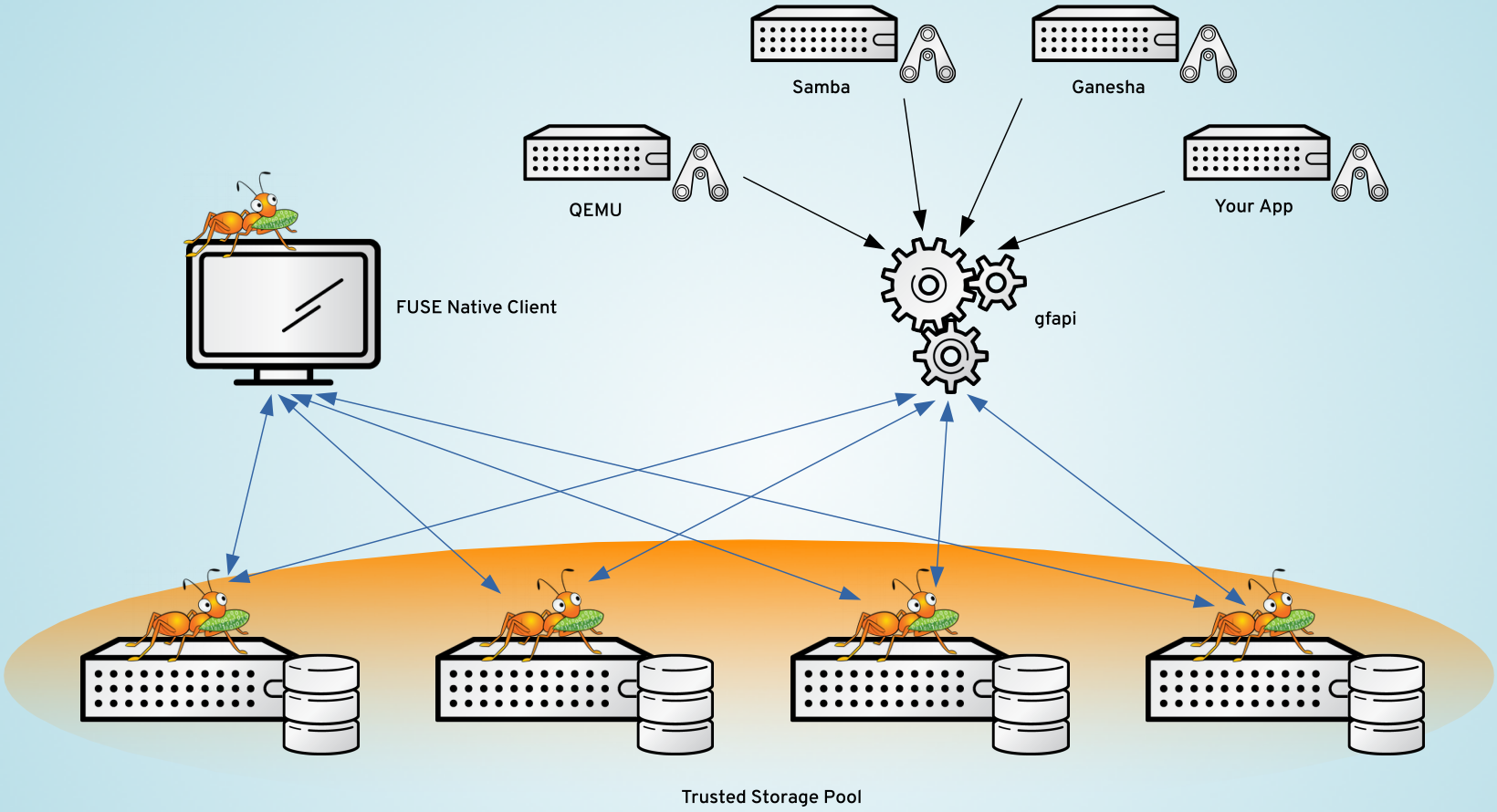
- Quick GlusterFS intro
- GlusterD & GlusterD-2.0
- Demo of GlusterD-2.0

WHAT IS GLUSTERFS?

- Distributed, scalable, network filesystem
  - No metadata server
  - Replication, erasure coding
- Posix compliant
- Flexible
  - Translators
  - Multiple access methods
- Commodity hardware

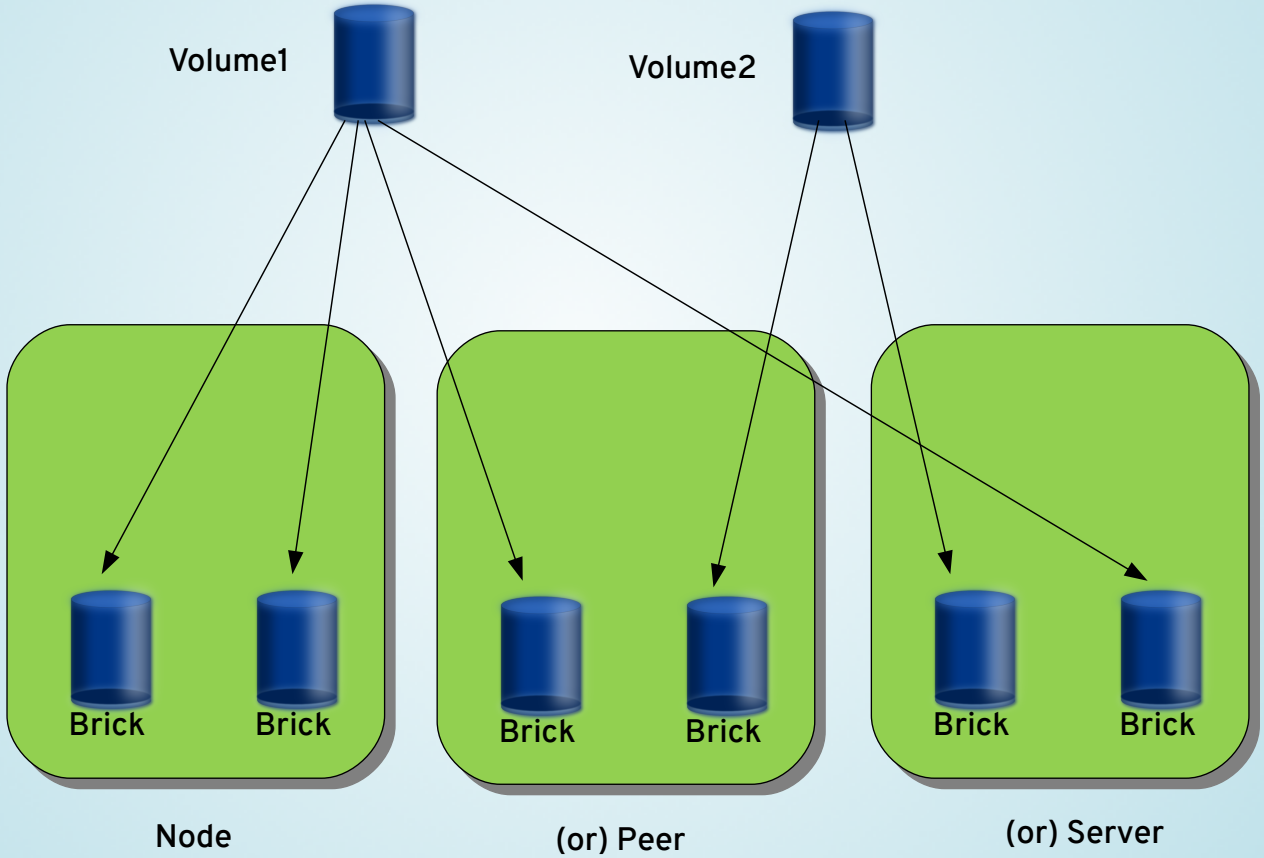
GLUSTERFS TERMS

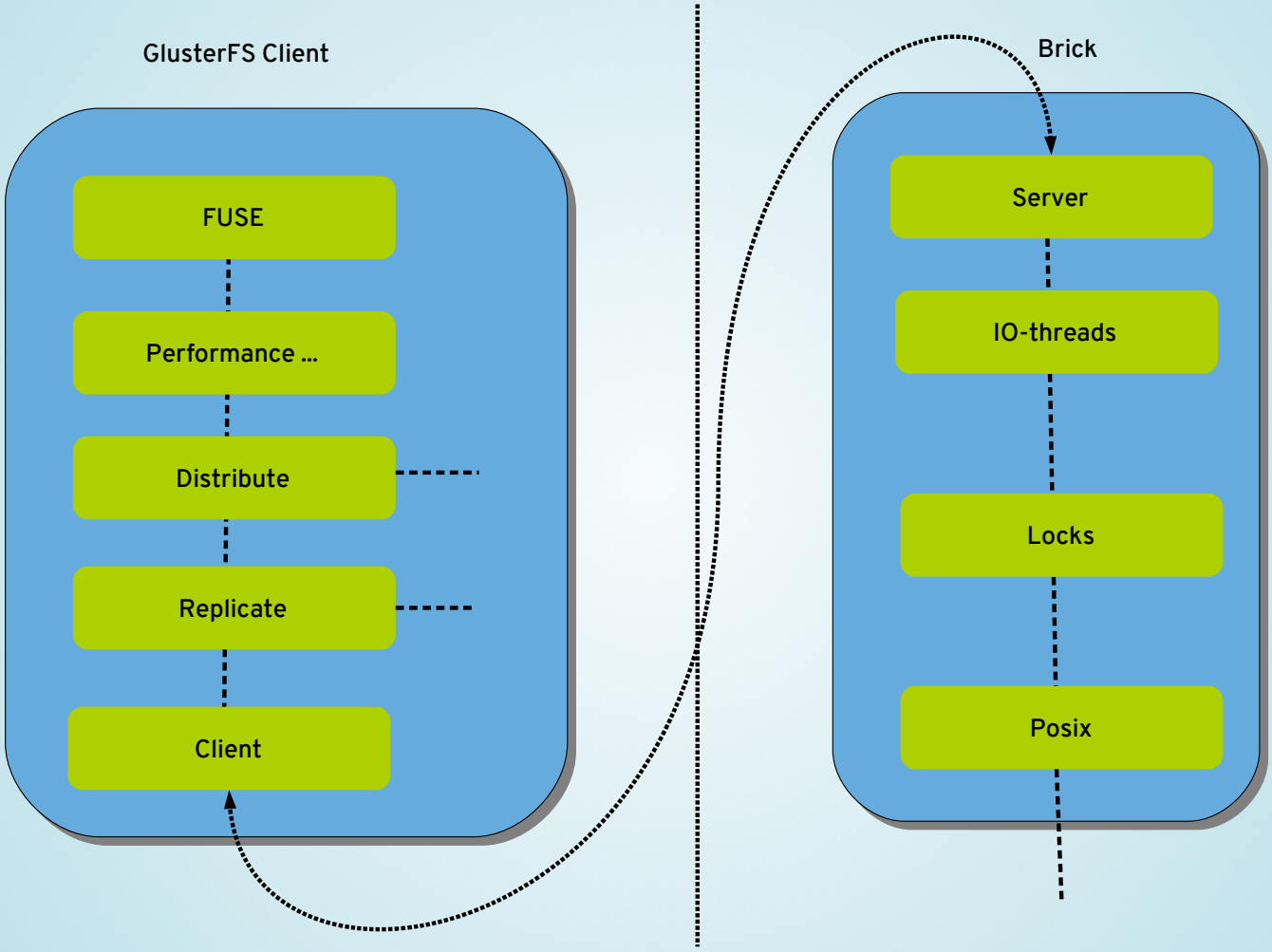
- Peer/Node/Server - A computer with the GlusterFS server packages installed
- Trusted Storage Pool - The GlusterFS cluster
- Brick - An empty directory on a server that can be exported
- Volume - A logical collection of bricks, that appears as a single export to clients
- Client - Any process that talks to bricks using the native protocol
- Translators - Modular bits of GlusterFS that implement the actual features


Samba
Ganesha
QEMU
Your App
FUSE Native Client
gfapi
Trusted Storage Pool

CREATING A GLUSTERFS VOLUME

- `gluster peer probe <hostname>`
- `gluster volume create <name> replica 2 <peername>:/path <peername>:/path ...`
- `gluster volume start <volumename>`
- `mount -t glusterfs <peername>:<volname> /<path to mountpoint>`


Volume1
Volume2
Brick
Brick
Brick
Brick
Brick
Brick
Node
(or) Peer
(or) Server


GlusterFS Client
FUSE
Performance ...
Distribute
Replicate
Client
Brick
Server
IO-threads
Locks
Posix

WHAT IS GLUSTERD?

- The distributed management daemon for GlusterFS
- Manages the TSP
- Manages the Volumes
- Gives clients volfiles
- Does other stuff as well

WHY DOES IT SUCK?

- Monolithic
- Complex
- Mesh network
- Equal peers

SO, GLUSTERD-2.0...

- Or GlusterD.next
- It's a new implementation of GD
- Solves all the problems
- Written in Go
- https://github.com/gluster/glusterd2
- Preview releases available

WHAT'S HAPPEN(ED/ING) WITH GD2?

- Building out the core of GD2
  - Transactions, plugins, basic commands ...
- Get this done first
  - Includes implementation and proper documentation
- Other GlusterFS features get implemented later

THE CENTRAL STORE

- Using etcd right now
- Automatic setup of etcd clusters
- etcd embedded within GD2
- Planning for automatic promotion/demotion etcd servers

THE TRANSACTION FRAMEWORK

- Runs actions across the cluster
- Flexible transaction framework
- Runs actions only on the required nodes
- List of "Steps"
  - A "Step" is
    - function that should be run
    - undo function that reverts changes done
    - a list of nodes to run the step on

THE DAEMON MANAGER

- Single framework for managing daemons
- Manages all daemons started by GD2
  - Bricks, SHD, QuotaD, SnapD etc.
- Describes a standard daemon interface
- Starts, stops, communicates with daemons
- Upcoming features
  - auto restart facility
  - dependencies

REST API

- Basic operations implemented
  - Peer add/remove/list
  - Volume add/remove/list/start/stop/info
- Need to revisit documentation
- Should possibly do a formal specification
  - Swagger/OpenAPI
- No auth yet

GRPC

- HTTP2 based RPC protocol
  - http://www.grpc.io
- Used for GD2 to GD2 communications
  - 2 services right now, peer and transaction
- Possibly for plugins
- TLS by default!

SUNRPC

- RPC protocol use by GlusterFS RPCs
- Uses XDR for data serialization
- Needed to communicate with GlusterFS bricks
- Clients communicate it to talk to GD2

STRUCTURED LOGGING

- Makes it easier to provide more context with logs
- Better machine parse-ability
  - `DEBU[0153] running step function reqid=e9dc9991-6f68-4da7-9d04-a9fa1a40fa00 stepfunc=testvol1.Unlock txnid=1e449f77-c5d5-4ea3-8bac-6b69257c9b06`
- Transaction framework uses it
  - Much easier to track transactions across cluster
- Improve formatting, different logging targets, msg-ids

STILL A LOT OF STUFF TODO

- Some stuff we have now will be rewritten
- Some more existing stuff aren't complete yet
- Stuff that hasn't had much/any work done yet
  - Plugins
  - Volgen
  - Events
  - Hooks
- Test everything
- Document everything

PLUGGABILITY

- Design the GD2 core to be pluggable
  - Allow external users to use a core framework without modifying source
  - Provide well documented interfaces users need to implement

PLUGGABILITY

- Pieces that require pluggability,
  - Xlators - to add new xlators into the graph, and to add new xlator options to volume set
  - Commands - to add new commands and extend existing ones
  - Daemons - to add new daemons to be managed by GD2
  - Events - for new features to add their own events to the event stream

PLUGGABILITY (ACTUAL PLUGINS)

- Two approaches
  - Go1.8 native plugin support
  - Sub-process plugin model
    - gRPC for communication and defining the plugin interface
  - Inspired by hashicorp/go-plugin

VOLGEN

- Volgen needs to be
  - Flexible - allow graph structures to be easily defined without changes to the GD2
  - Pluggable - allow new xlators to be inserted into the graph
  - *Composable*
- *Currently just a simple text template, which has values filled*
- A POC is in progress
  - https://github.com/kshlm/glusterd2-volgen

EVENTS AND HOOKS

- Will help in keeping GD2 pluggable and flexible
- Events
  - Stream of events happening on a GD2
  - 'volume-create', 'brick-start', 'brick-died' etc.
  - Maybe think about it being extended to the cluster
- Hooks
  - Basically the same as GD to a user
  - Will leverage events to provide hook points
  - Should avoid deadlock problems of current hooks.

QUESTIONS?

DEMO!

THANK YOU!