



# GlusterFS

*A Scale-out Software Defined Storage*

**Rajesh Joseph**  
**Poornima Gurusiddaiah**

# AGENDA

- ◆ Introduction
- ◆ GlusterFS Concepts
- ◆ Architecture
- ◆ Additional Features
- ◆ Installation
- ◆ Setup and Configuration
- ◆ Future Work
- ◆ Q&A



# GlusterFS

- ◆ Open-source general purpose scale-out distributed file system
- ◆ Aggregates storage exports over network interconnect to provide a single unified namespace
- ◆ Layered on disk file systems that support extended attributes

- ◆ No meta-data server
- ◆ Modular Architecture for Scale and Functionality
- ◆ Heterogeneous commodity hardware
- ◆ Scalable to petabytes & beyond



# GlusterFS Concepts



# Node

A node is server capable of hosting GlusterFS bricks

## ◆ Server

- ◆ Intel/AMD x86 64-bit processor
- ◆ Disk: 8GB minimum using direct-attached-storage, RAID, Amazon EBS, and FC/Infiniband/iSCSI SAN disk backends using SATA/SAS/FC disks
- ◆ Memory: 1GB minimum

## ◆ Logical Volume Manager

- ◆ LVM2 with thin provisioning

## ◆ Networking

- ◆ Gigabit Ethernet
- ◆ 10 Gigabit Ethernet
- ◆ InfiniBand (OFED 1.5.2 or later)

## ◆ Filesystem

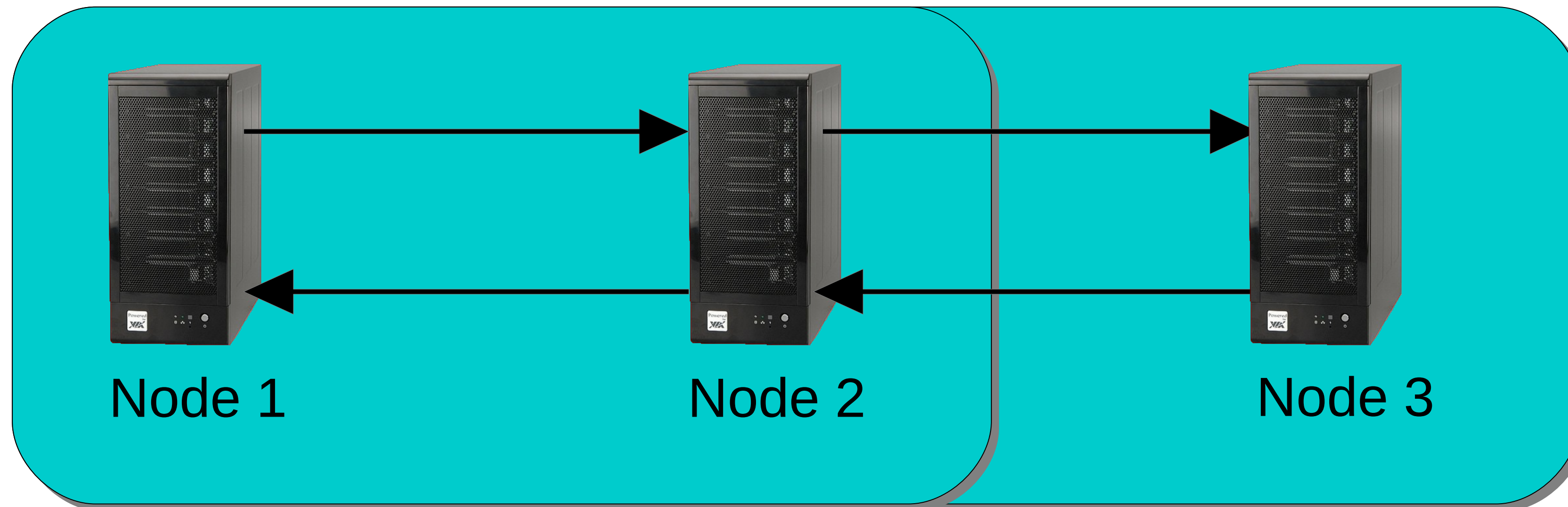
- ◆ POSIX w/ Extended Attributes (EXT4, XFS, BTRFS, ...)



# Trusted Storage Pool

A collection of storage servers (Node)

- ◆ Also known as Cluster
- ◆ Trusted Storage Pool is formed by invitation – “*probe*”
- ◆ Members can be dynamically added and removed from the pool
- ◆ Only nodes in a Trusted Storage Pool can participate in volume creation





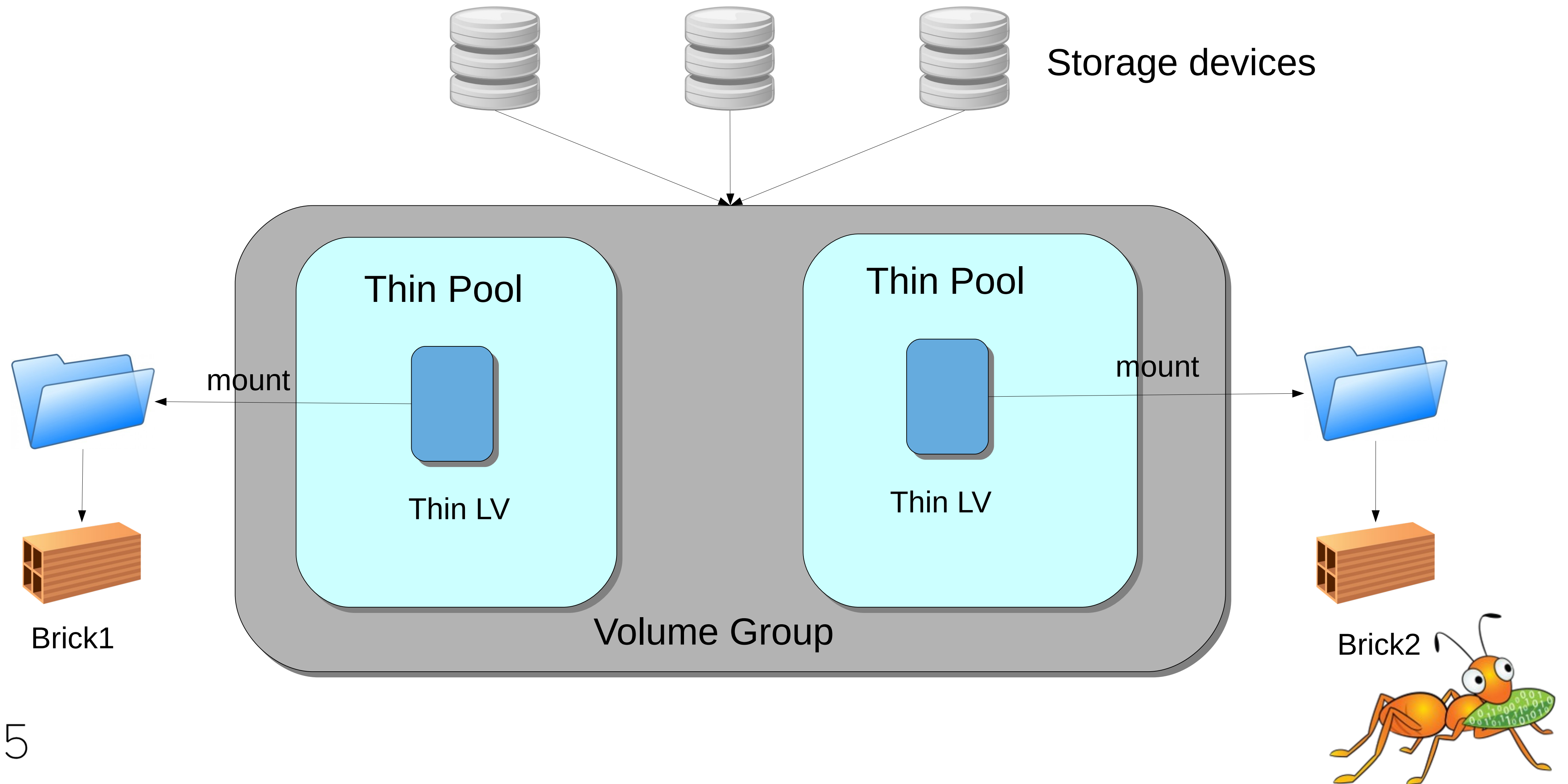
# Bricks

A unit of storage used as a capacity building block

- ♦ A brick is the combination of a node and an export directory – e.g. `hostname:/dir`
- ♦ Layered on posix compliant file-system (e.g. XFS, ext4)
- ♦ Each brick inherits limits of the underlying filesystem
- ♦ It is recommended to use an independent thinly provisioned LVM as brick
  - ♦ Thin provisioning is needed by Snapshot feature



# Bricks





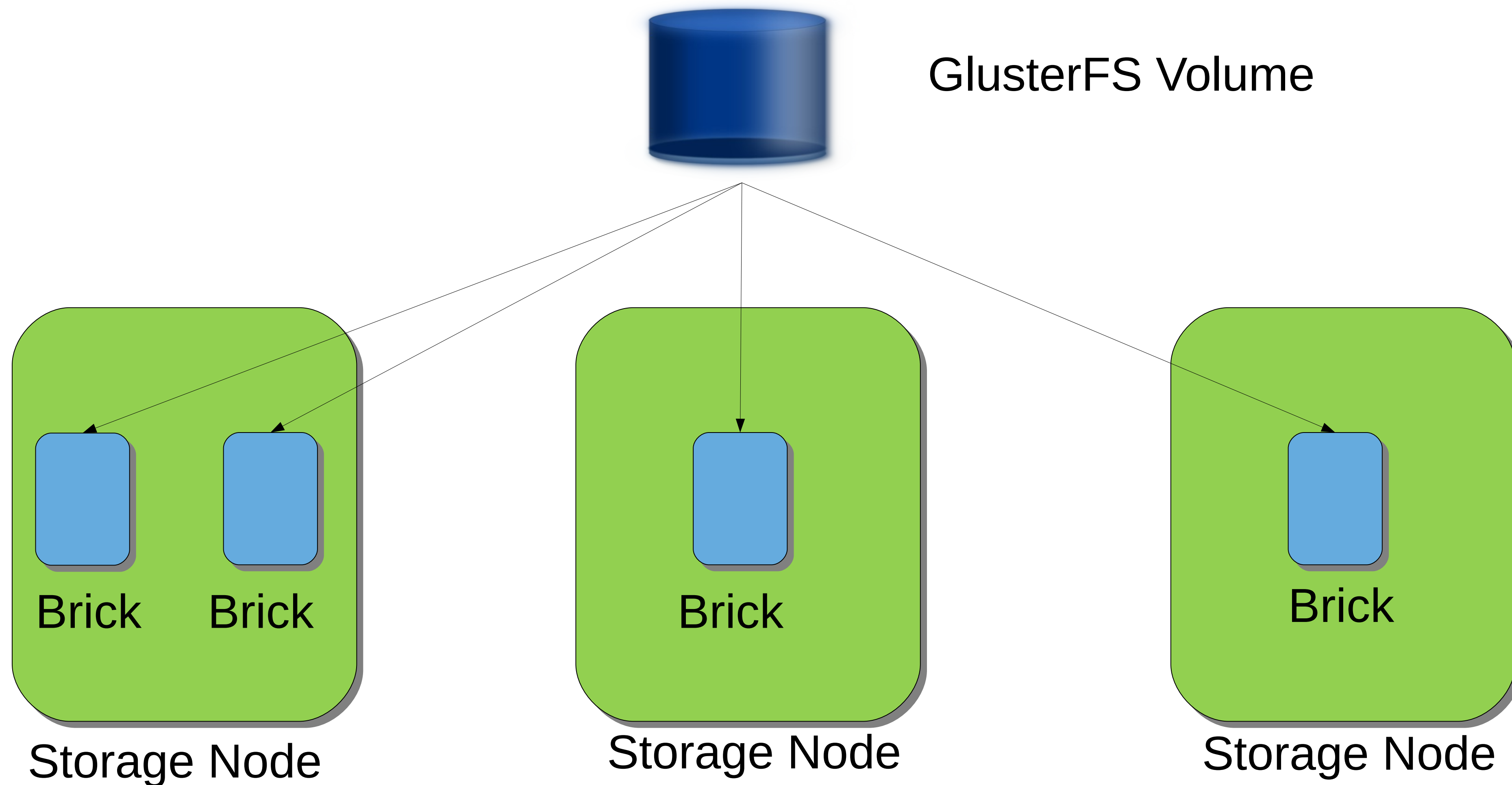
# GlusterFS Volume

A volume is a logical collection of one or more bricks

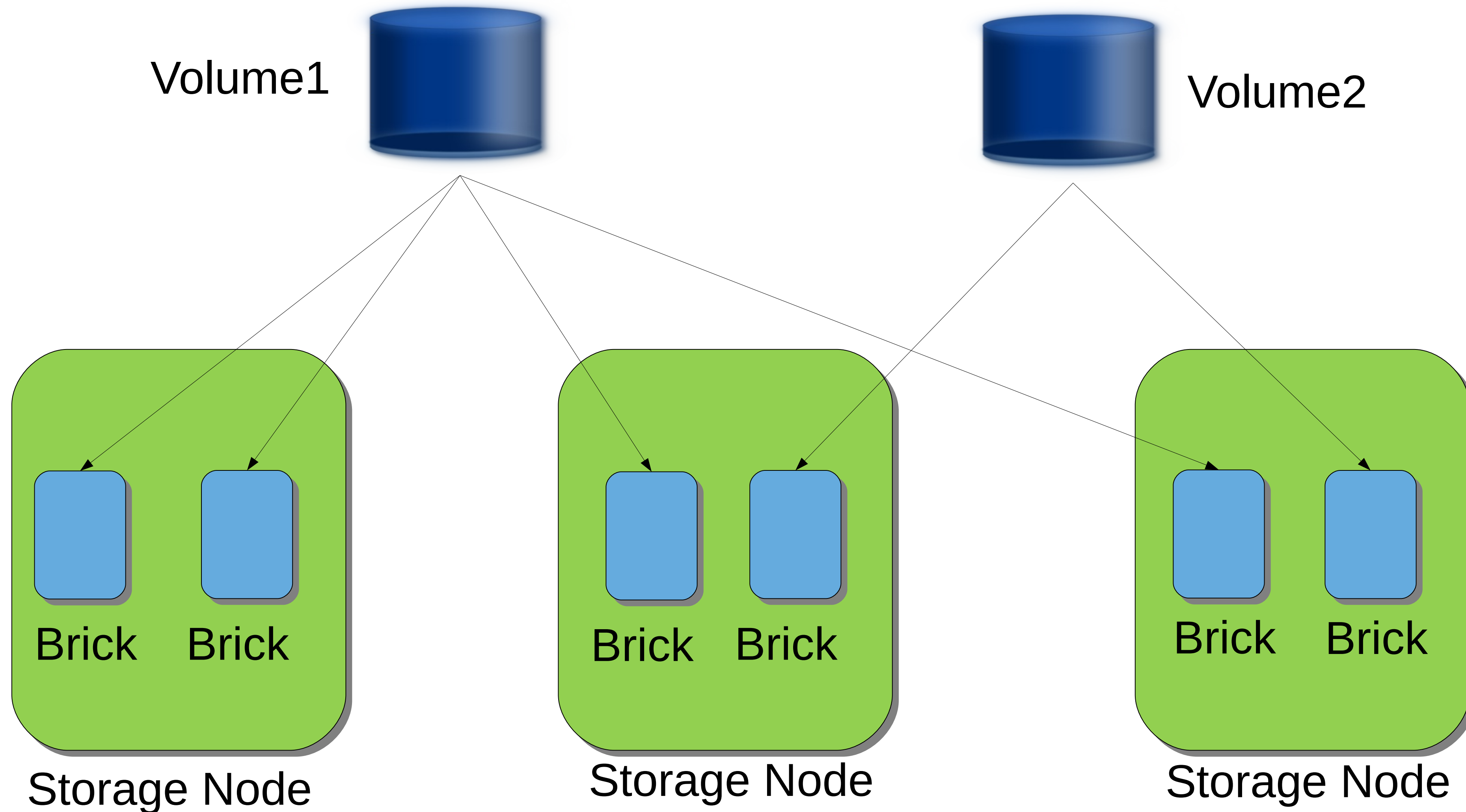
- ◆ Node hosting these bricks should be part of a single Trusted Storage Pool
- ◆ One or more volumes can be hosted on the same node



# GlusterFS Volume



# GlusterFS Volume



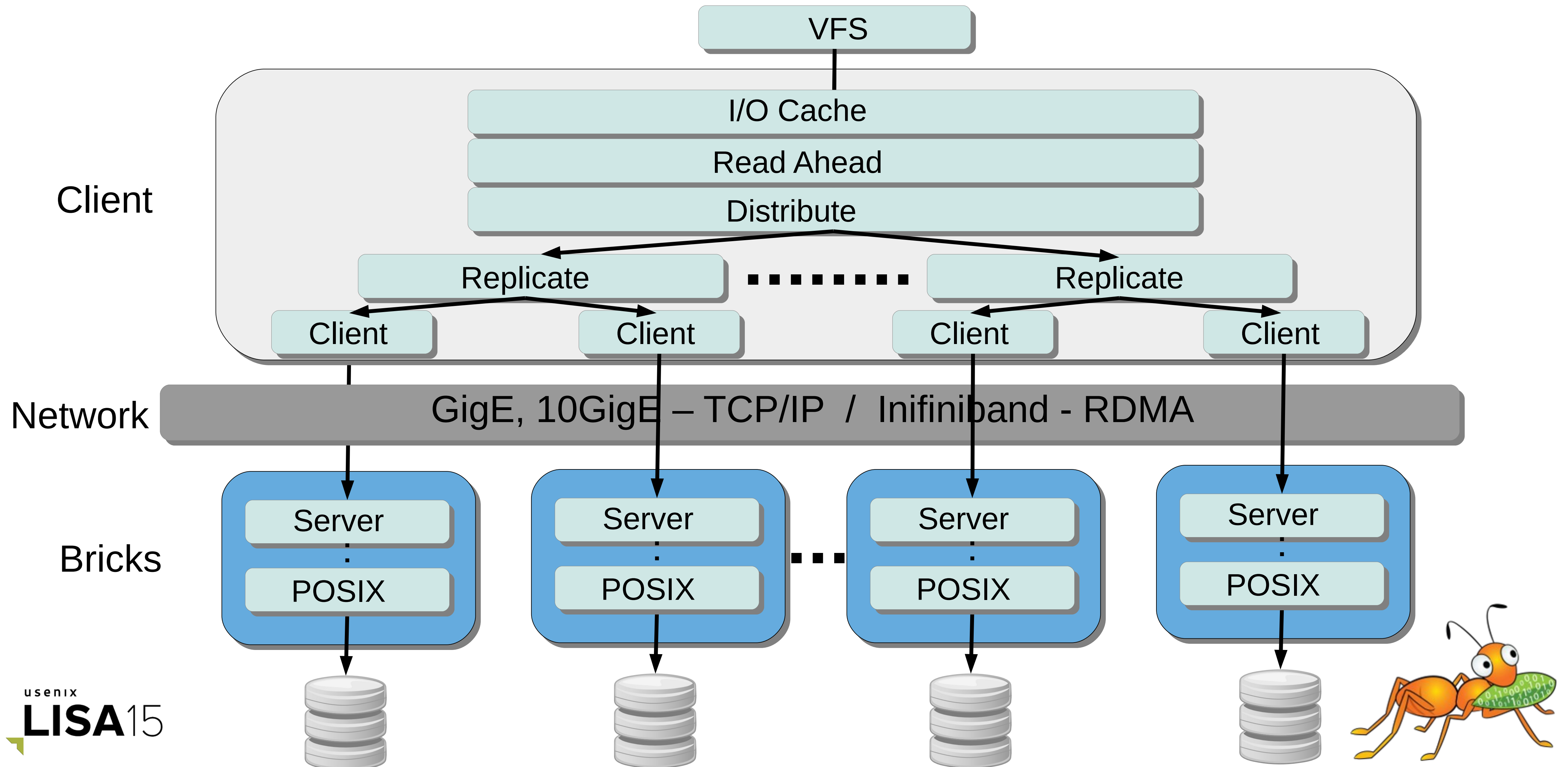
# Translators

Building block of GlusterFS process

- Translators can be stacked together for achieving desired functionality
- Translators are deployment agnostic – can be loaded in either the client or server stacks



# Translators



# GlusterFS Processes

## ◆ *glusterd*

- ◆ Management daemon
- ◆ One instance on each GlusterFS node

## ◆ *glusterfsd*

- ◆ GlusterFS brick daemon
- ◆ One process per brick on each node
- ◆ Managed by glusterd

## ◆ *gluster*

- ◆ Gluster console manager (CLI)

## ◆ *glusterfs*

- ◆ Node services
- ◆ One process for each service
  - ◆ NFS Server, Self heal, Quota, ...

## ◆ *mount.glusterfs*

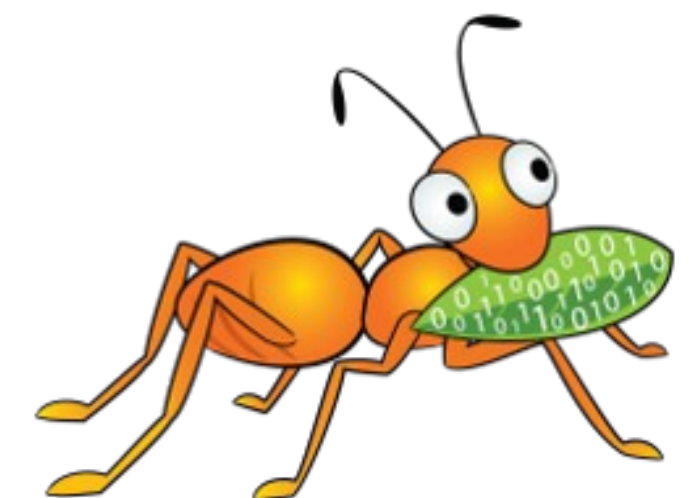
- ◆ FUSE native client mount extension

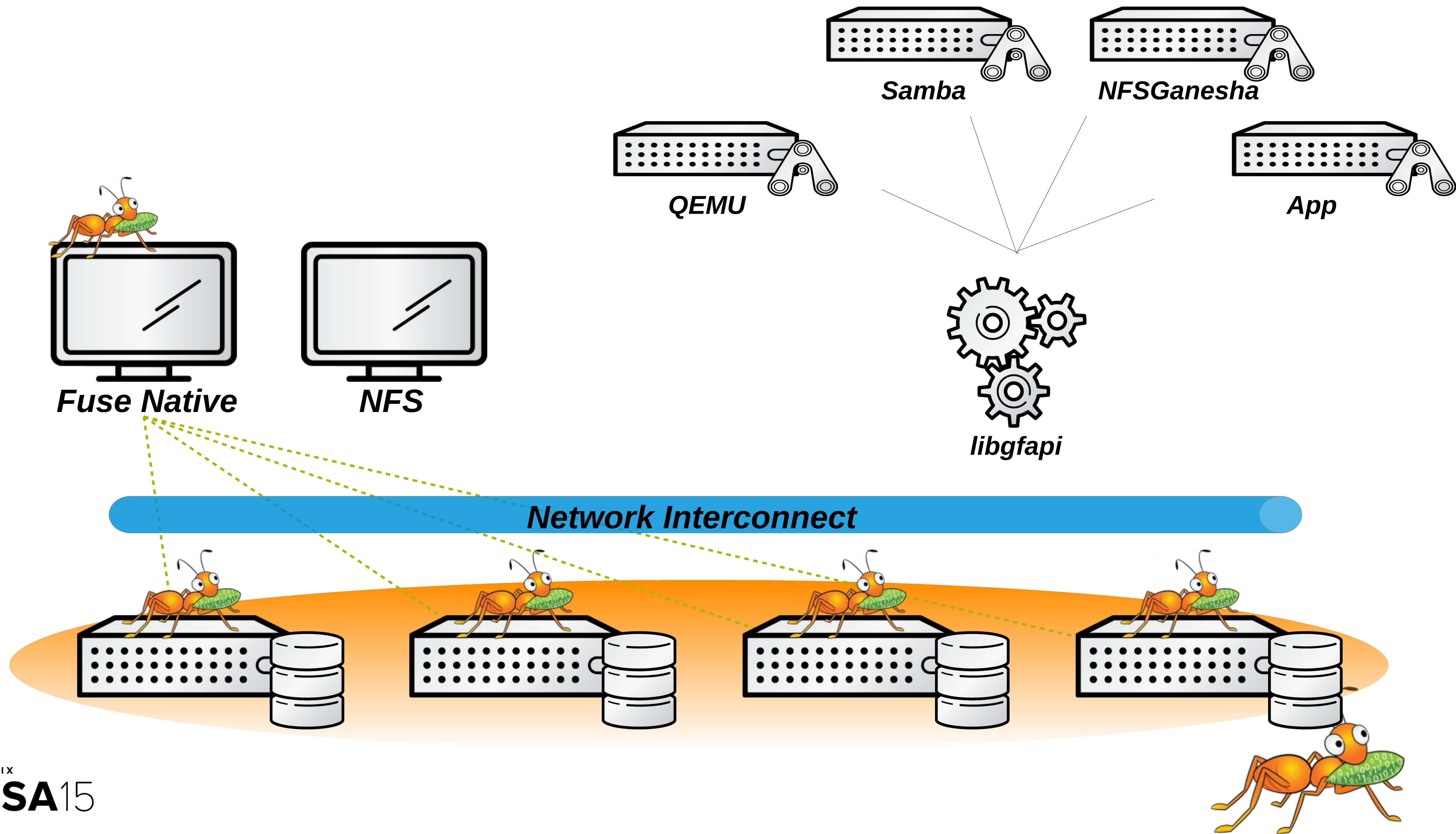




# Accessing Volume

- ◆ Volume can be mounted on local file-system
- ◆ Following protocols supported for accessing volume
  - ◆ GlusterFS Native client
    - ◆ Filesystem in Userspace (FUSE)
    - ◆ libgfapi flexible abstracted storage
      - ◆ Samba, NFS-Ganesha, QEMU
  - ◆ NFS
    - ◆ NFS Ganesha
    - ◆ Gluster NFSv3
- ◆ SMB / CIFS
  - ◆ Windows and Linux
- ◆ Gluster For OpenStack
  - ◆ Object-based access via OpenStack Swift





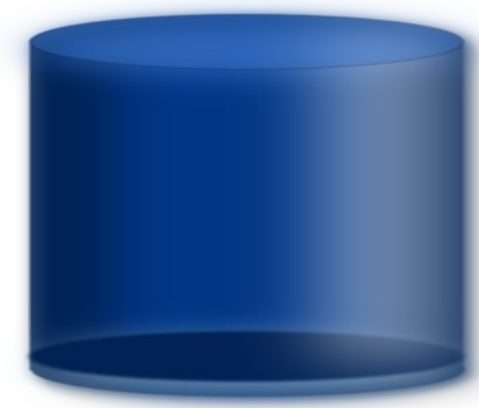
# GlusterFS Native Client (FUSE)

- ◆ FUSE kernel module allows the filesystem to be built and operated entirely in userspace
- ◆ Specify mount to any GlusterFS server
- ◆ Native Client fetches volfile from mount server, then communicates directly with all nodes to access data

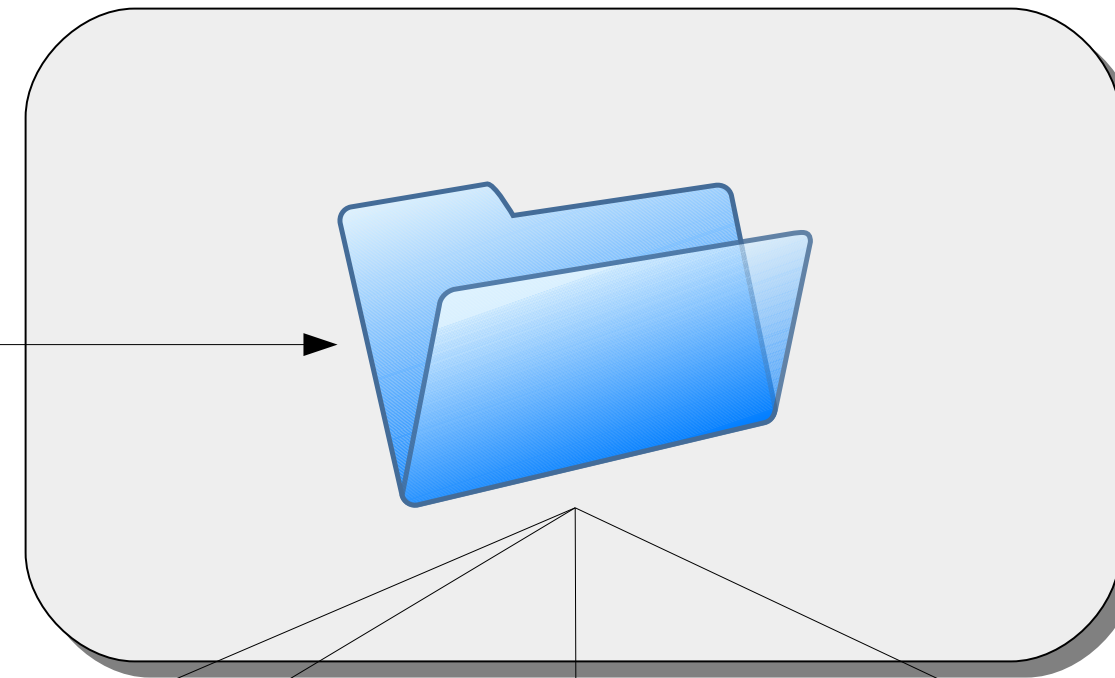


# Accessing Volume – Fuse client

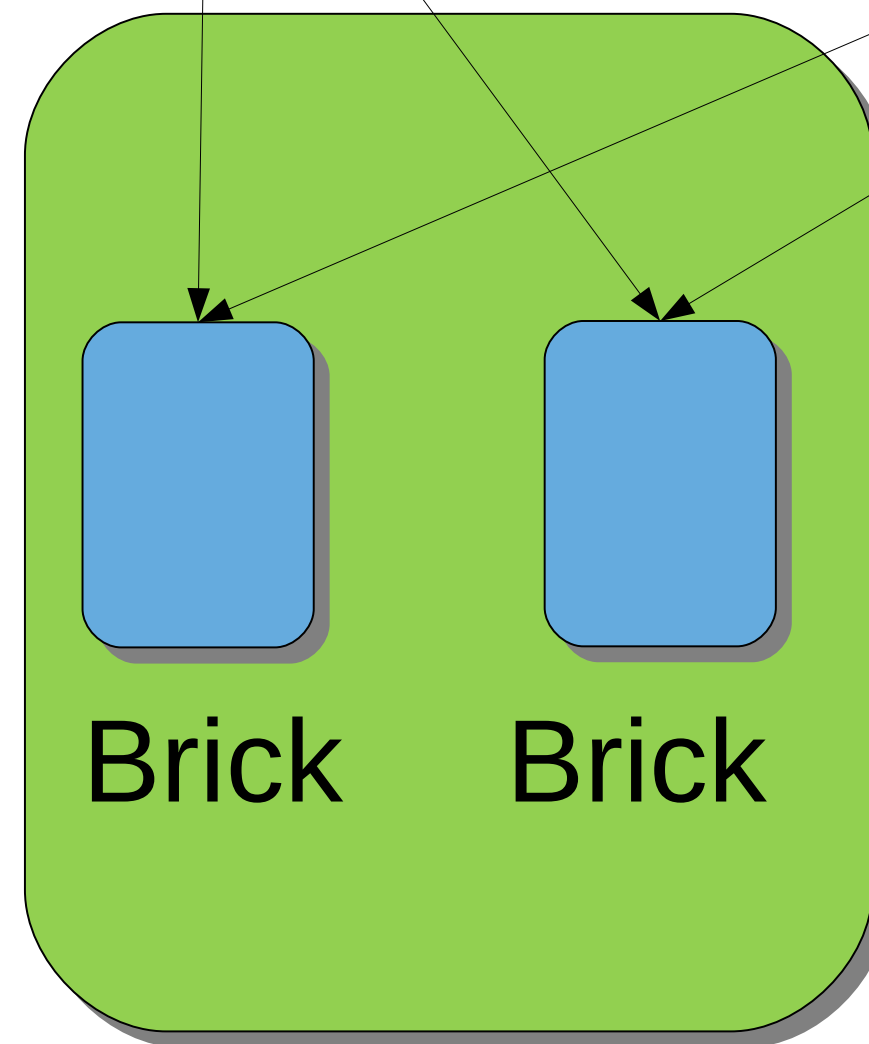
Gluster Volume



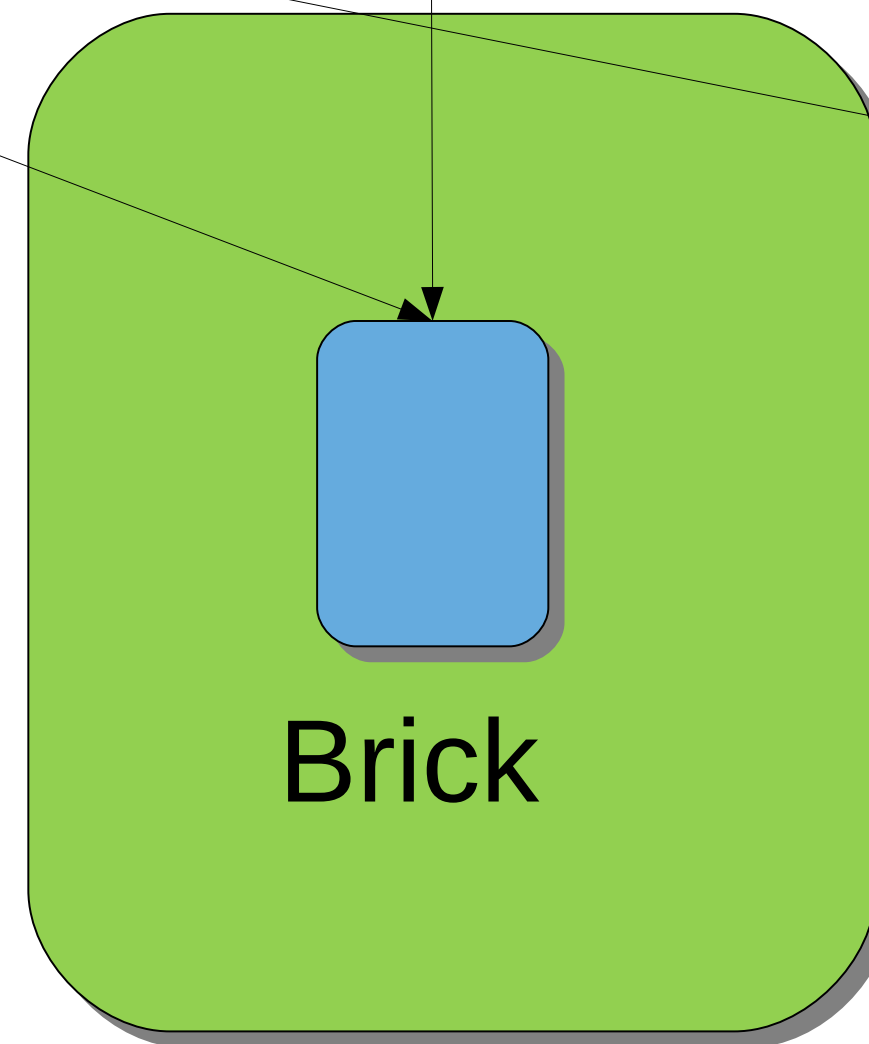
mount



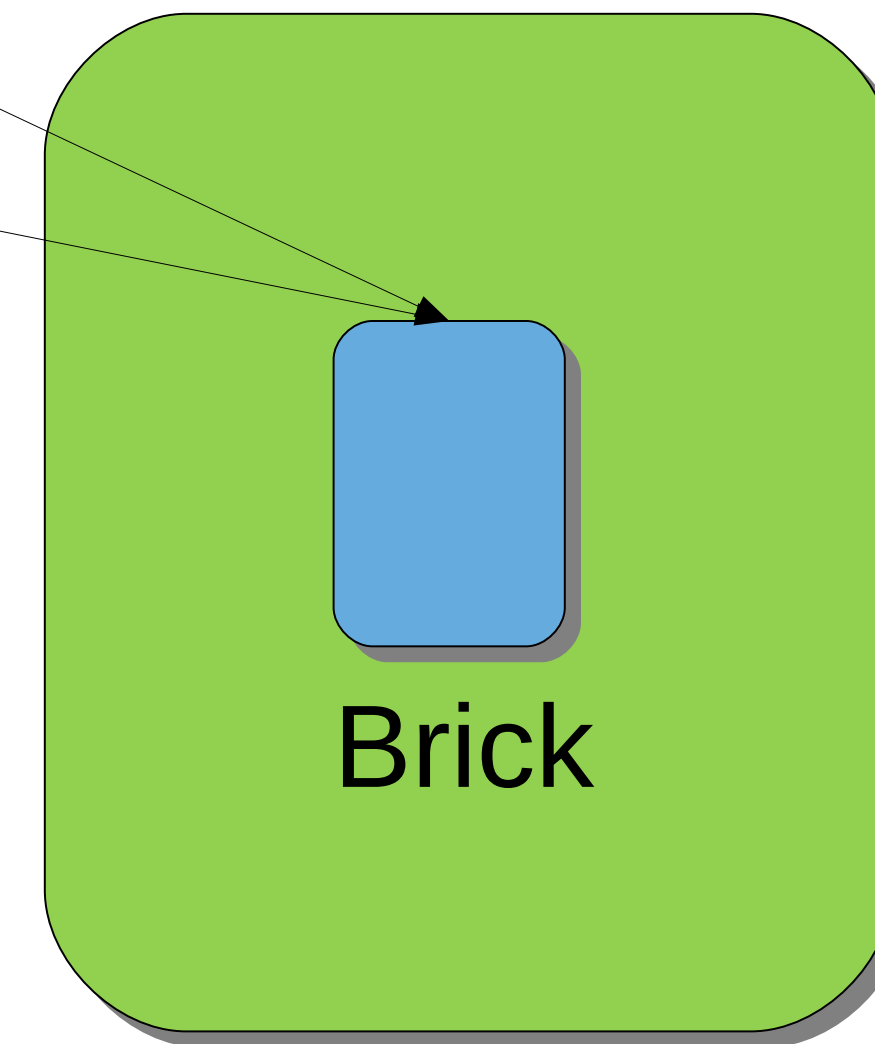
Client



Storage Node



Storage Node



Storage Node



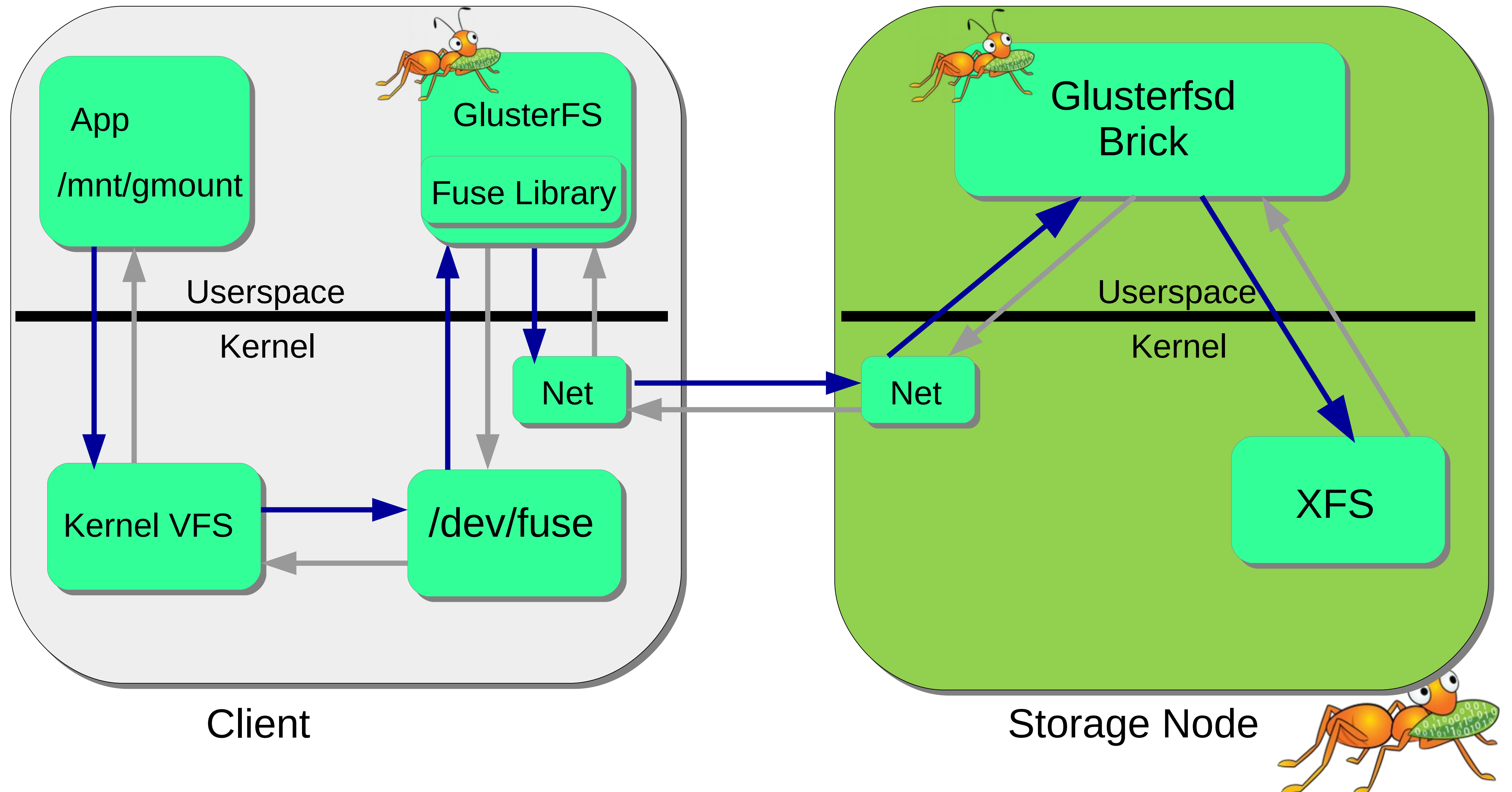
# libgfapi

- ◆ Userspace library for accessing GlusterFS volume
- ◆ File-system like API
- ◆ No FUSE, No copies, No context switches
- ◆ Samba, QEMU, NFS-Ganesha integrated with libgfapi



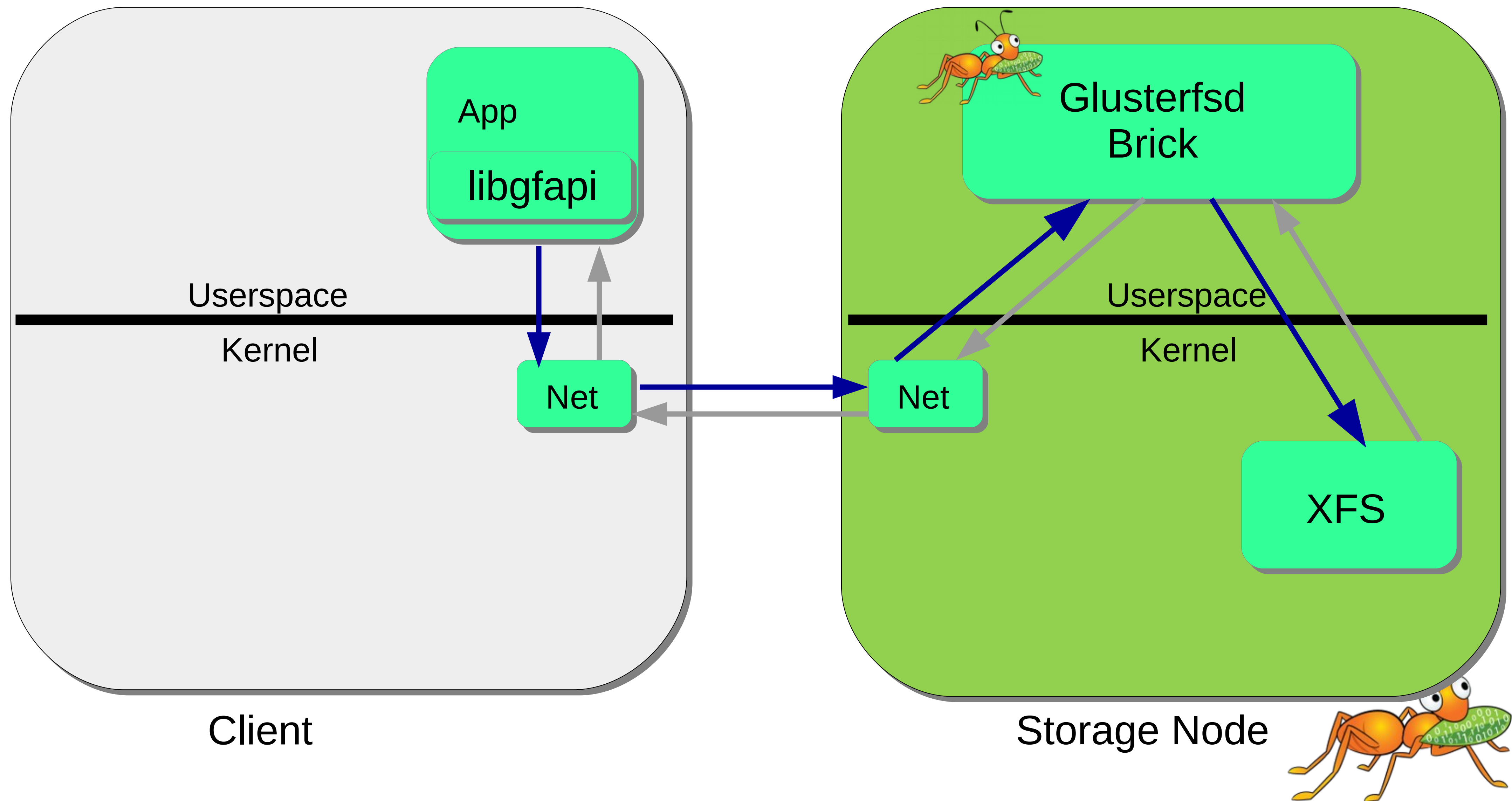


# Libgfapi - Advantage





# Libgfapi - Advantage

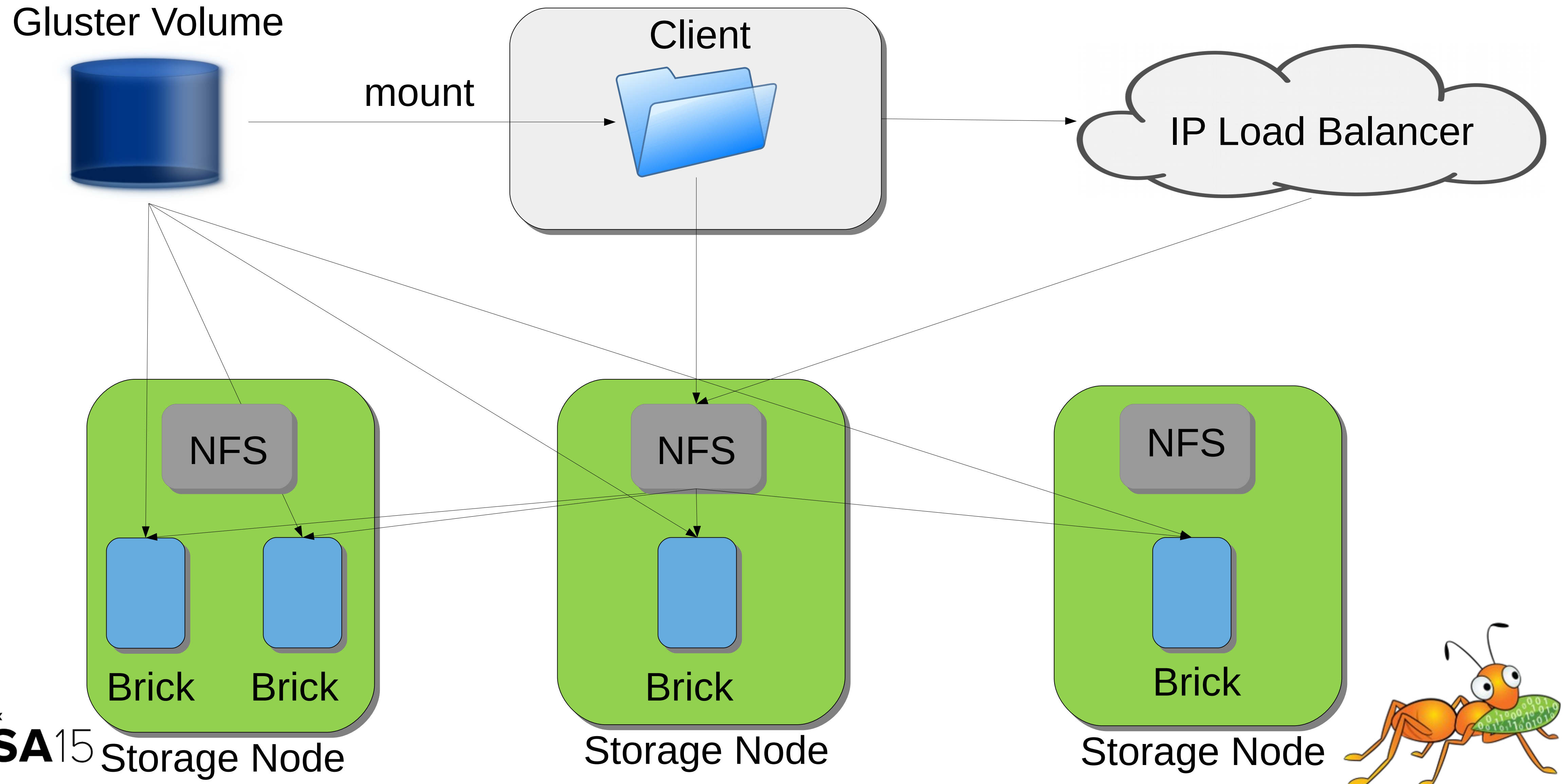


# Gluster NFS

- ◆ Supports NFS v3 protocol
- ◆ Mount to any server or use load-balancer
- ◆ GlusterFS NFS server uses Network Lock Manager (NLM) to synchronize locks across clients
- ◆ Load balancing should be managed externally



# Accessing Volume – Gluster NFS



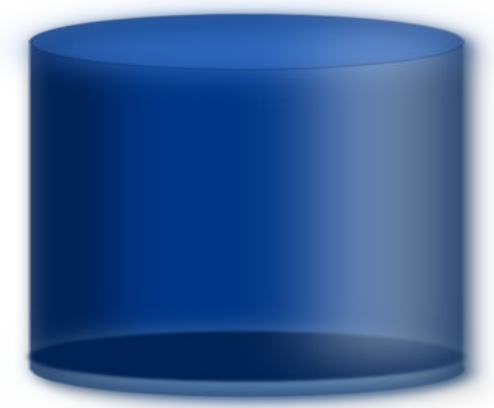
# SMB / CIFS

- ◆ Samba VFS plugin for GlusterFS
- ◆ Uses libgfapi
- ◆ Must be setup on each server you wish to connect
- ◆ CTDB is required for Samba clustering



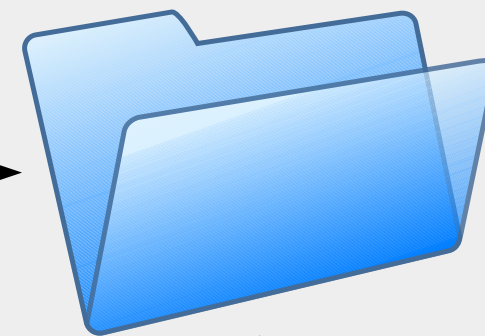
# Accessing Volume – SMB

Gluster Volume

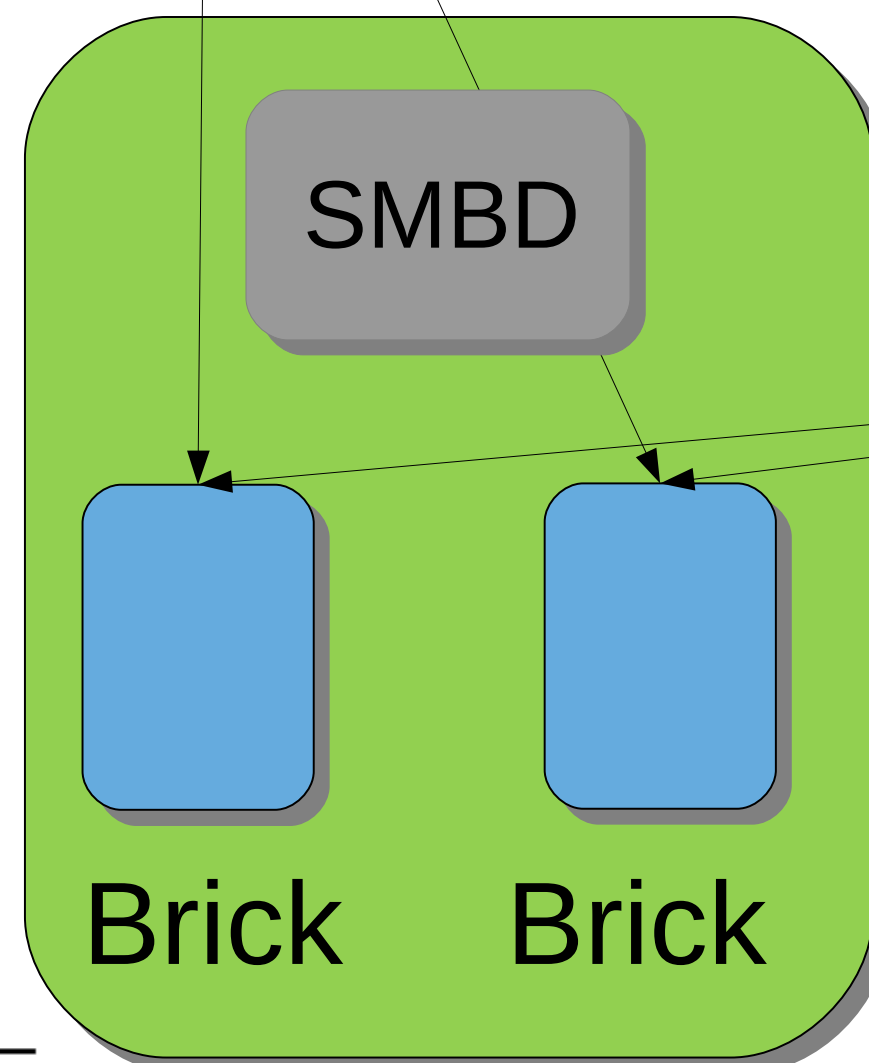


mount

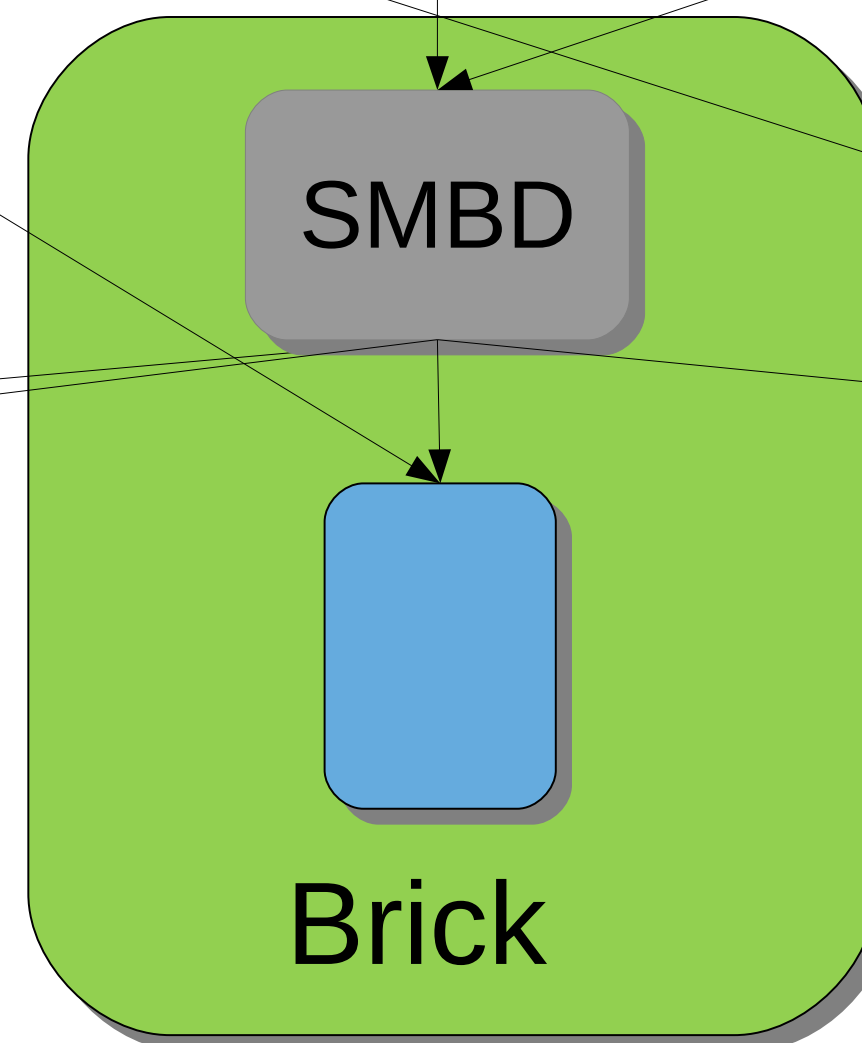
Client



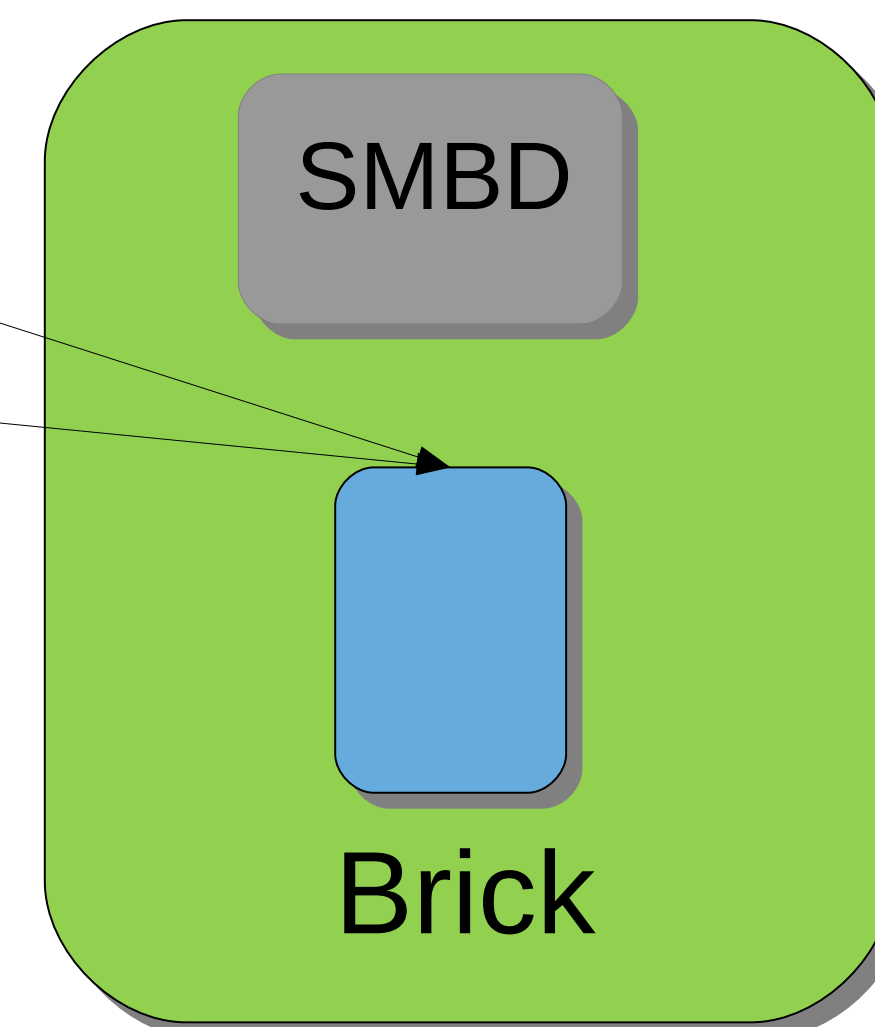
IP Load Balancer



Brick Brick



Brick



Brick





# Volume Types

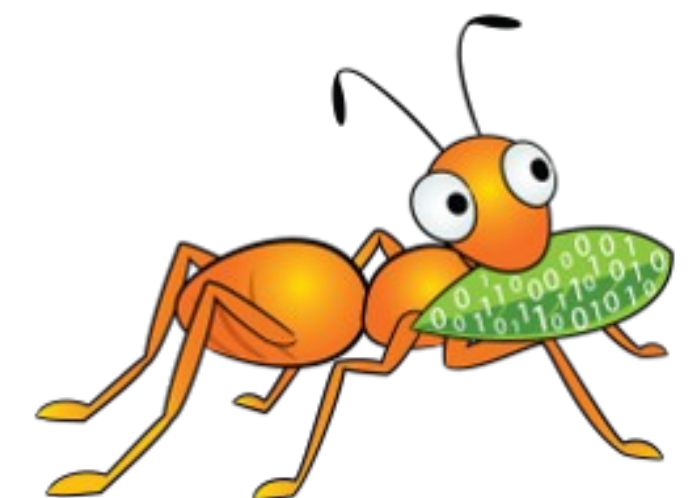
- ◆ Type of a volume is specified at the time of volume creation
- ◆ Volume type determines how and where data is placed

## ◆ Basic Volume Types

- ◆ Distributed
- ◆ Replicated
- ◆ Dispersed
- ◆ Striped
- ◆ Sharded

## ◆ Advance Volume Types

- ◆ Distributed Replicated
- ◆ Distributed Dispersed
- ◆ Distributed Striped
- ◆ Distributed Striped Replicated



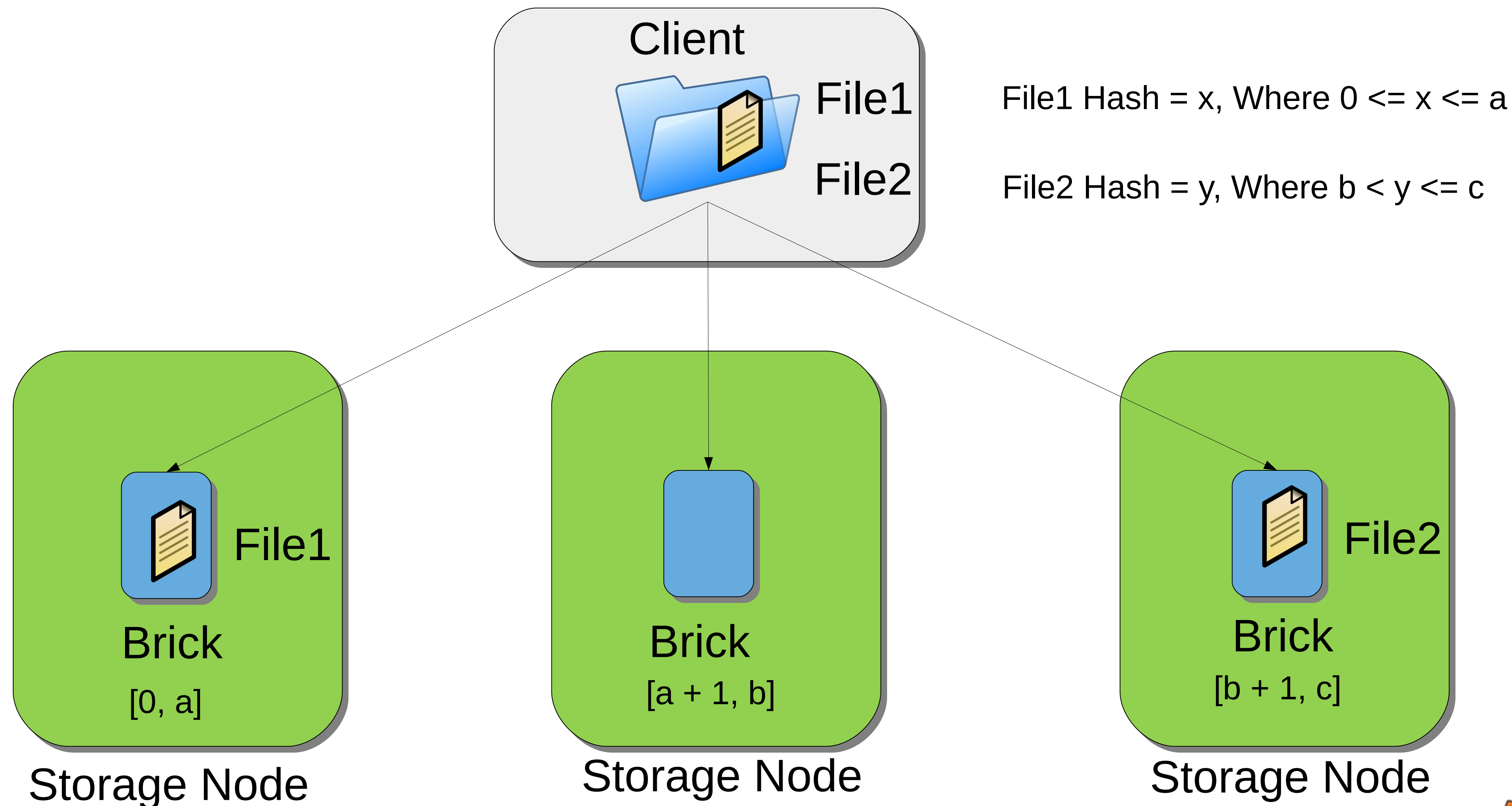


# Distributed Volume

- ◆ Distributes files across various bricks of the volume
- ◆ Similar to file-level RAID-0
- ◆ Scaling but no high availability
- ◆ Uses Davies-Meyer hash algorithm
- ◆ A 32-bit hash space is divided into N ranges for N bricks
- ◆ Removes the need for an external meta data server



# Distribute Volume

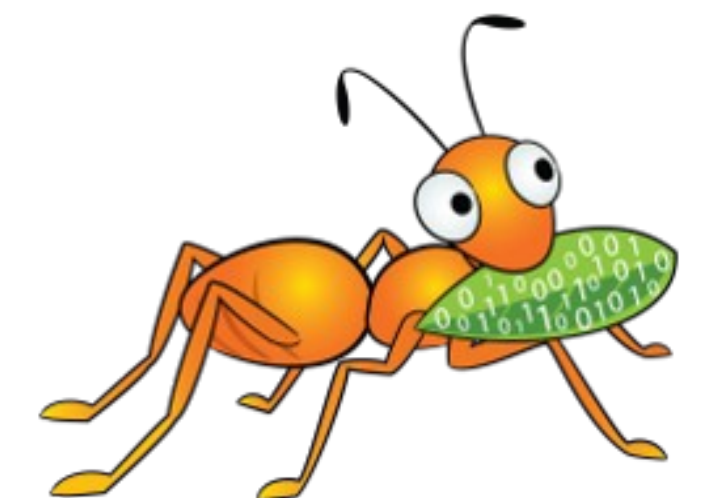
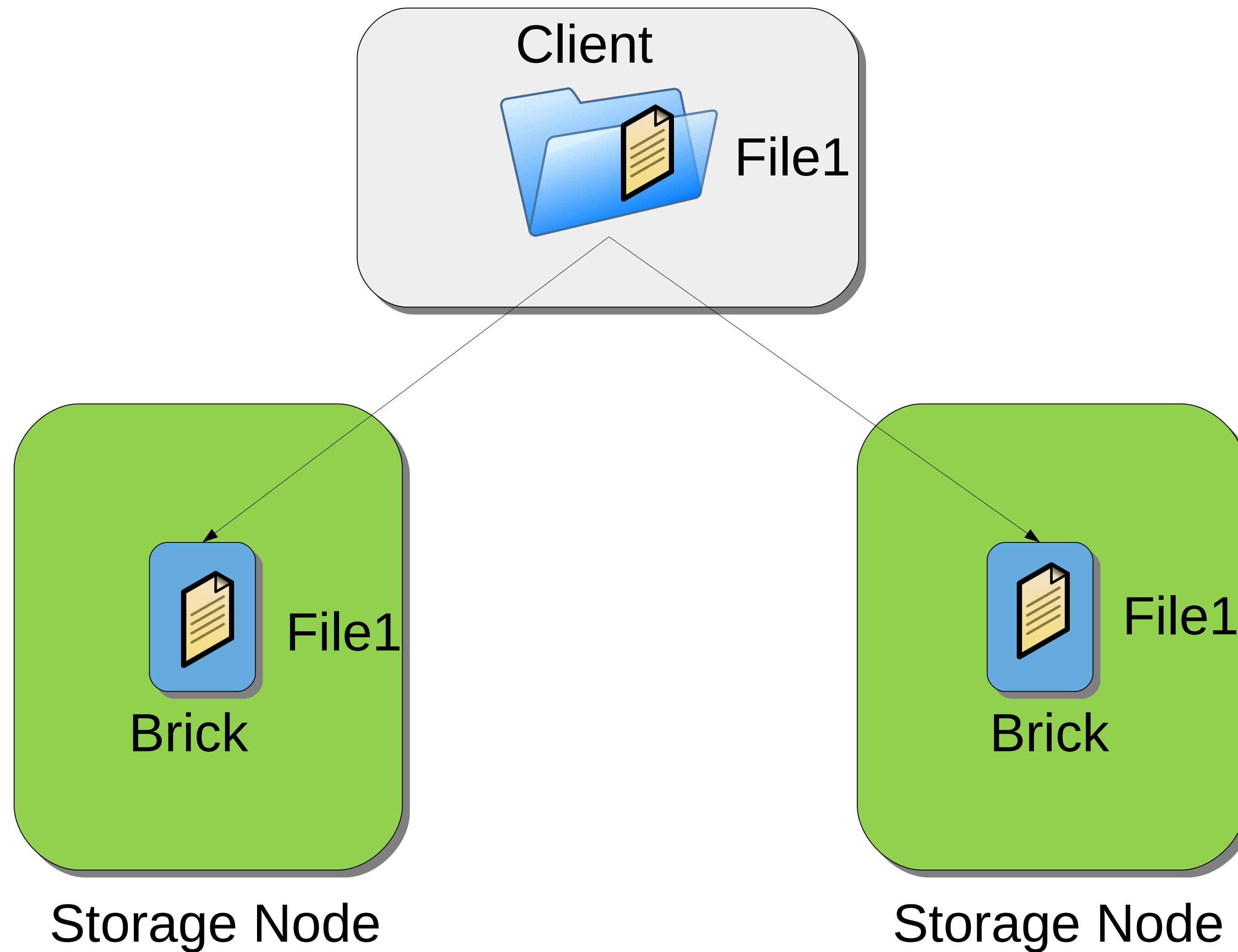


# Replicate Volume

- ◆ Synchronous replication of all directory and file updates
- ◆ Similar to file-level RAID-1
- ◆ High availability but no scaling
- ◆ Replica pairs are decided at the volume creation time
- ◆ Its recommended to host each brick of replica set on different node



# Replicate Volume

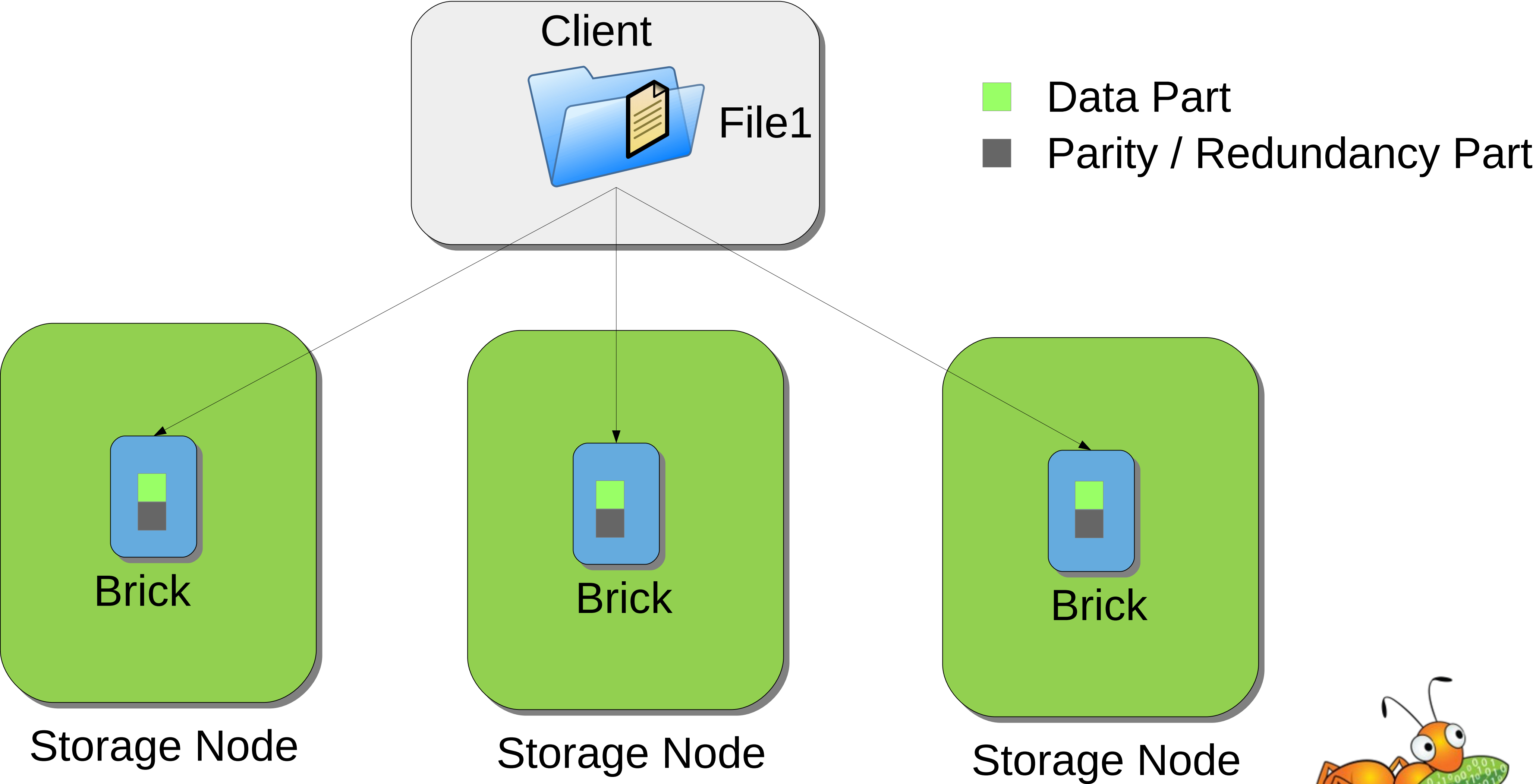


# Disperse Volume

- ◆ Erasure Coded volume
- ◆ High availability with less number of bricks
- ◆ Store  $m$  disk of data on  $k$  disk ( $k > m$ )
- ◆  $n$  ( $k-m$ ) redundant disks
- ◆ Following setup is the most tested configuration
  - ◆ 6 bricks with redundancy level 2 ( $4 + 2$ )
  - ◆ 11 bricks with redundancy level 3 ( $8 + 3$ )
  - ◆ 12 bricks with redundancy level 4 ( $8 + 4$ )



# Disperse Volume



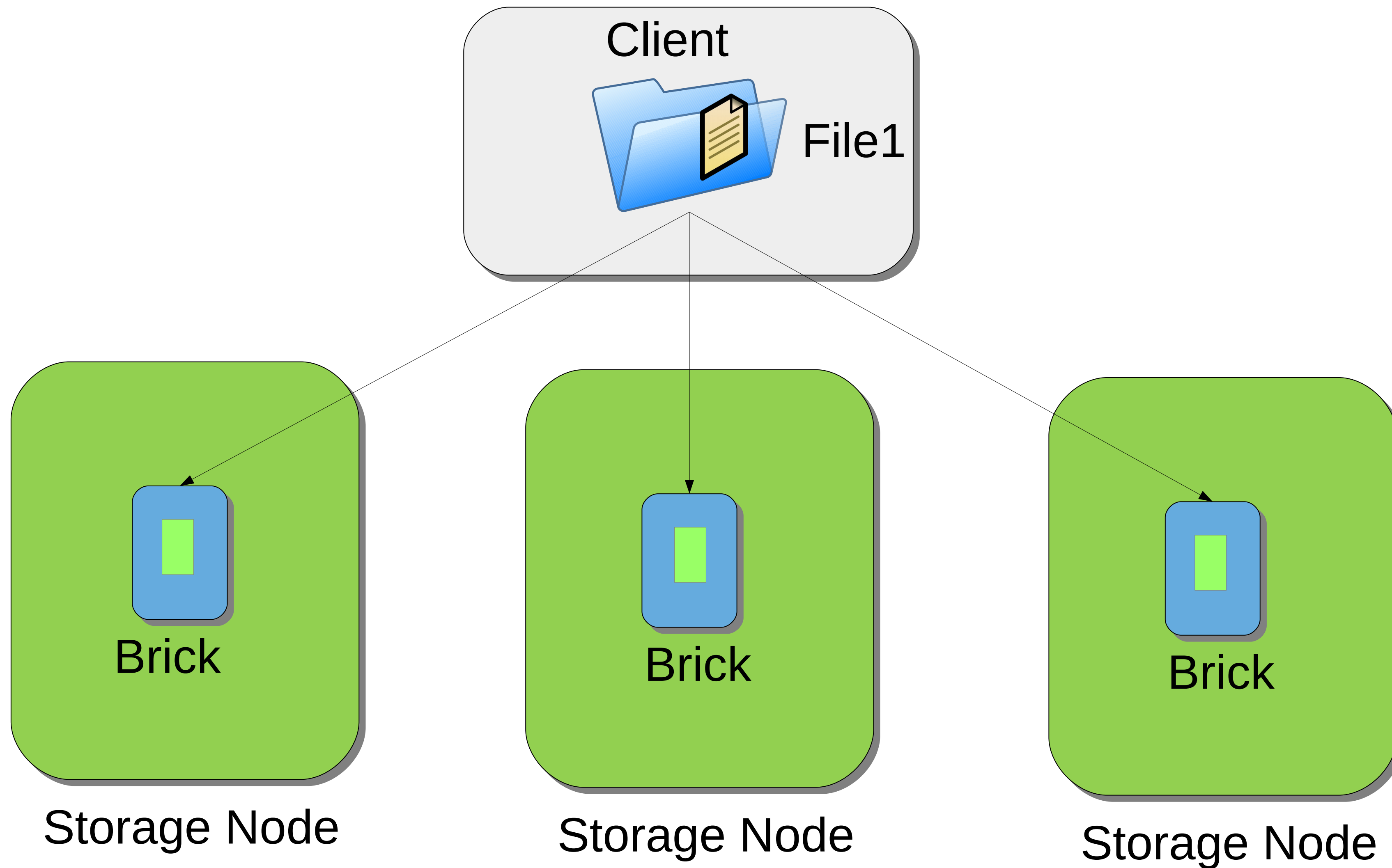


# Striped Volume

- ◆ Individual files split among bricks (sparse files)
- ◆ Similar to block-level RAID-0
- ◆ Recommended only when very large files greater than the size of the disks are present
- ◆ Chunks are files with holes – this helps in maintaining offset consistency



# Striped Volume

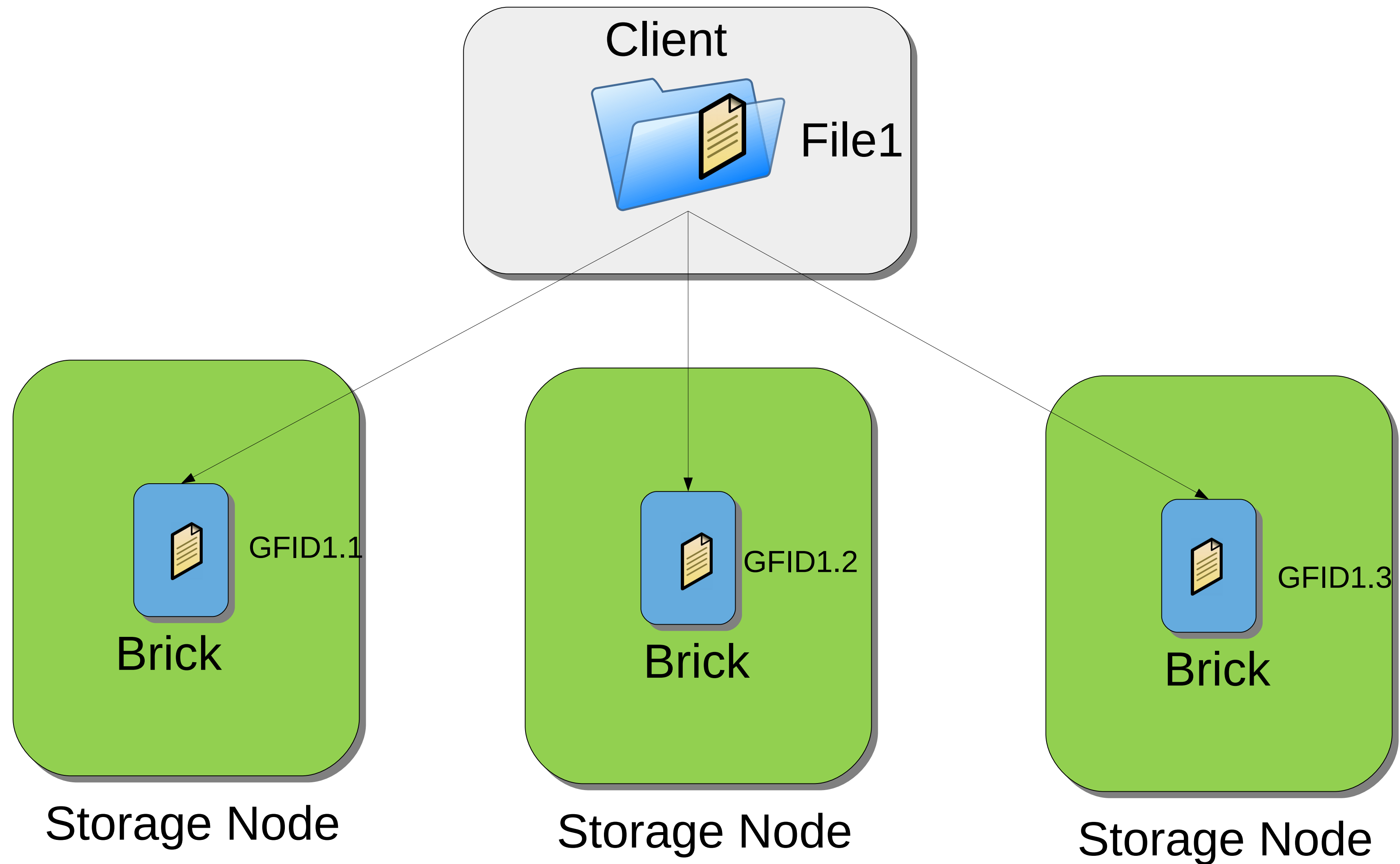


# Sharded (Stripe – 2.0) Volume

- ◆ Individual files split among bricks
  - ◆ Each part split is a different file in the backend
  - ◆ File name of each part derived from GFID (UUID) of actual file
- ◆ Unlike Stripe file rename will not have any impact on these parts.
- ◆ Unlike other volume types shard is a volume option which can be set on any volume
- ◆ Recommended only when very large files greater than the size of the disks are present



# Sharded Volume

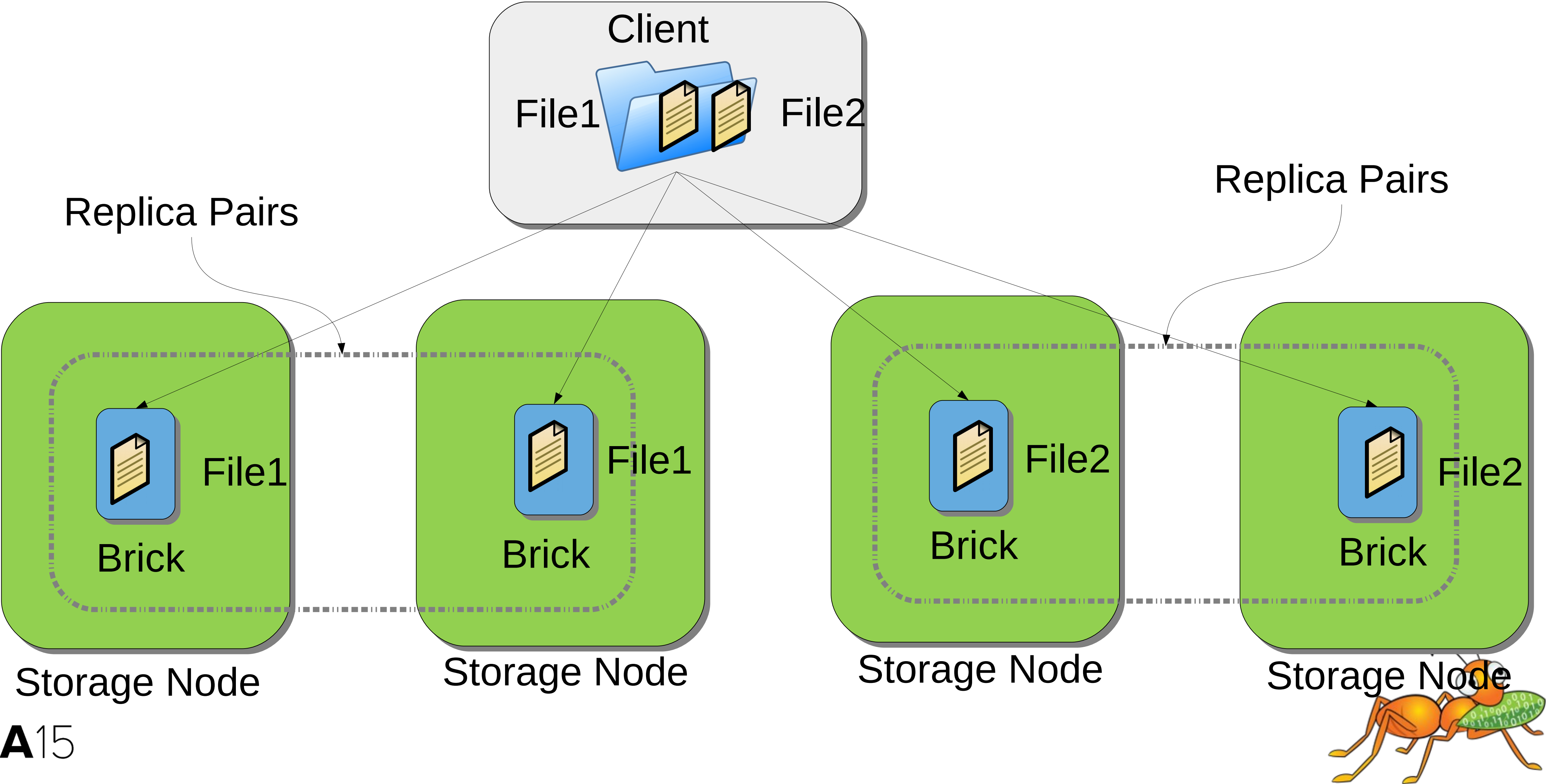


# Distribute Replicate Volume

- ◆ Distribute files across replicated bricks
- ◆ Number of bricks must be a multiple of the replica count
- ◆ Ordering of bricks in volume definition matters
- ◆ Scaling and high availability
- ◆ Most preferred model of deployment



# Distribute Replicate Volume





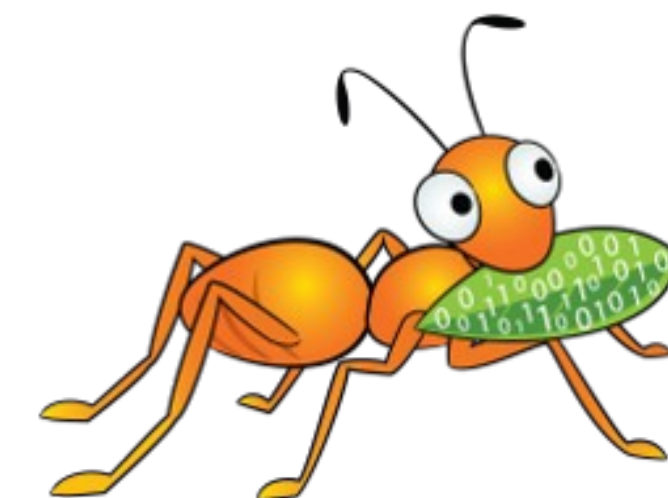
# Performance

- ◆ Different SLA than a local file-system
- ◆ Uneven performance distribution between nodes and clients
- ◆ Various performance options available (e.g., read-ahead, md-cache, etc)
  - ◆ Based on workload use various options
- ◆ Don't forget to attend detailed session on performance by Jeff Darcy

Topic : Evaluating Distributed File System Performance (GlusterFS and Ceph)  
When : Friday, November 13, 2015 - 9:00am-10:30am  
Detail : <https://www.usenix.org/conference/lisa15/conference-program/presentation/darcy>



# Additional Features



# Additional Features

- ◆ Geo Replication
- ◆ Volume Snapshot
- ◆ Quota
- ◆ Bit-rot
- ◆ Compression
- ◆ Encryption at rest
- ◆ Trash / Recycle-bin
- ◆ pNFS with NFS-Ganesha

- ◆ Data Tiering
- ◆ User Serviceable Snapshots
- ◆ Server side Quorum
- ◆ Client side Quorum
- ◆ And many more ...

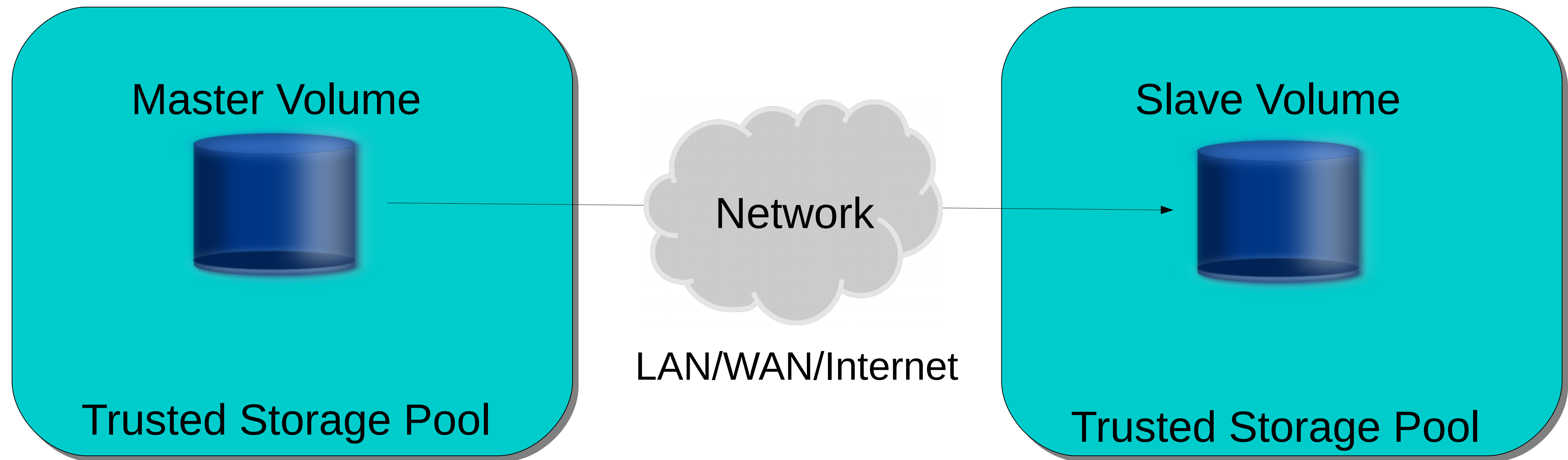


# Geo-Replication

- ◆ Mirrors data across geographically distributed trusted storage pools
- ◆ Provides back-ups of data for disaster recovery
- ◆ Master-Slave model
- ◆ Asynchronous replication
- ◆ Provides an incremental continuous replication service over
  - ◆ Local Area Networks (LANs)
  - ◆ Wide Area Network (WANs)
  - ◆ Internet



# Geo-Replication



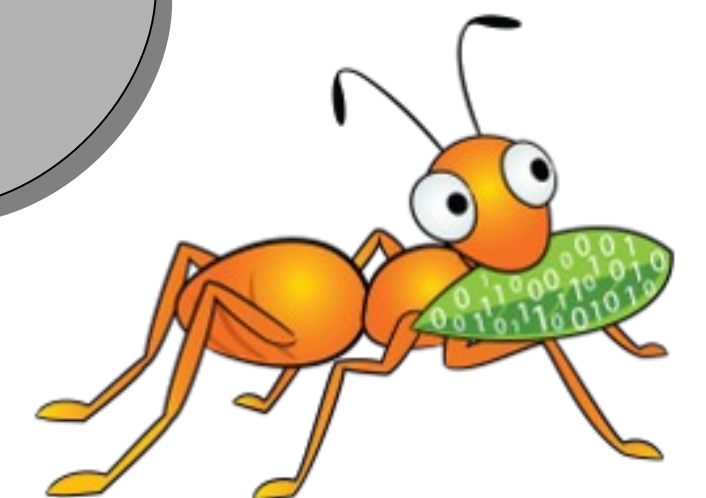
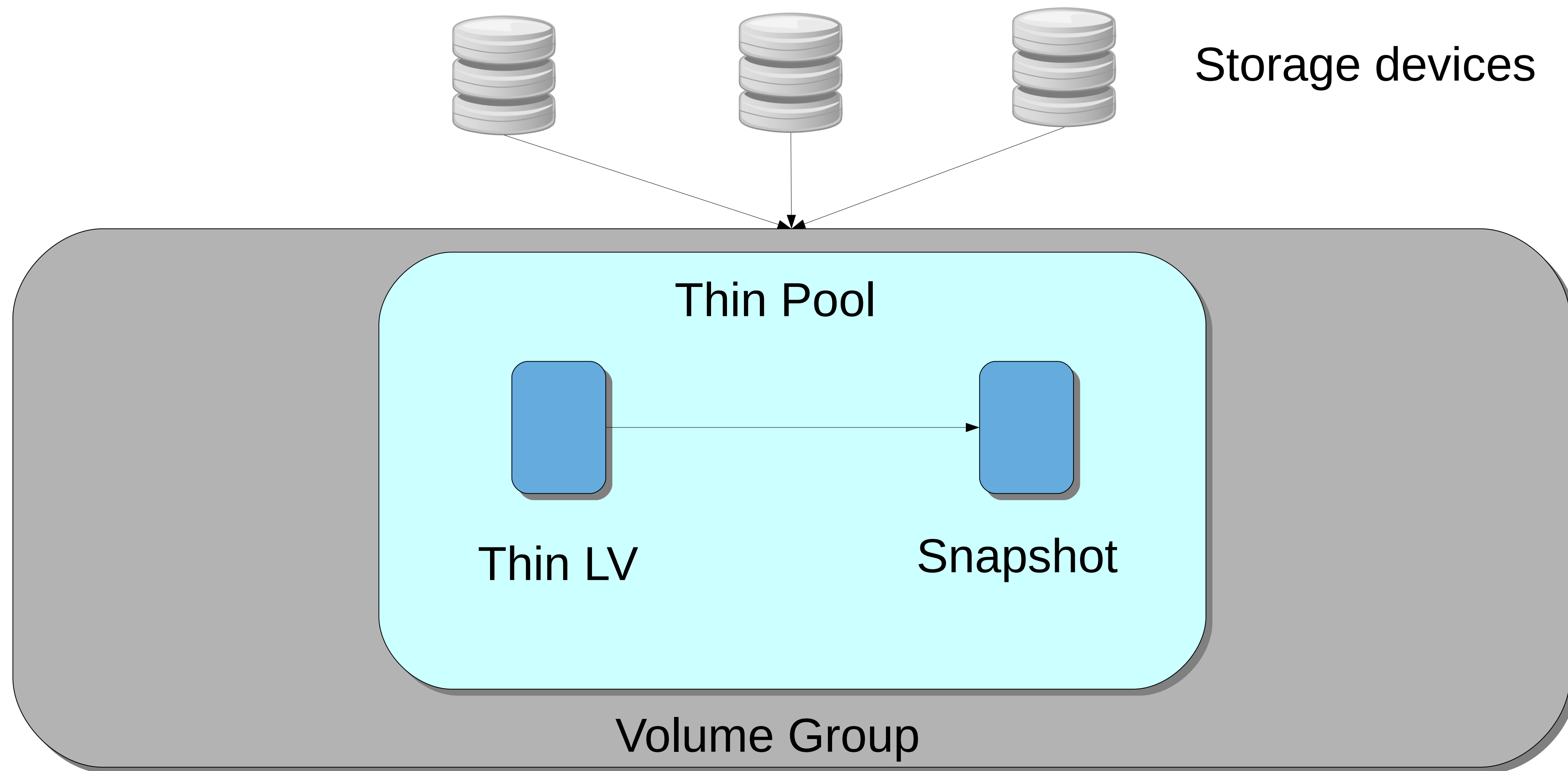
# GlusterFS Snapshot

- ◆ Volume level crash consistent snapshots
- ◆ LVM2 based
  - ◆ Operates only on thin-provisioned volumes
- ◆ Snapshots themselves are thin-provisioned snapshot volumes
- ◆ A GlusterFS volume snapshot consists of snapshots of the individual bricks making up the volume
- ◆ Snapshot of a GlusterFS volume is also a GlusterFS volume

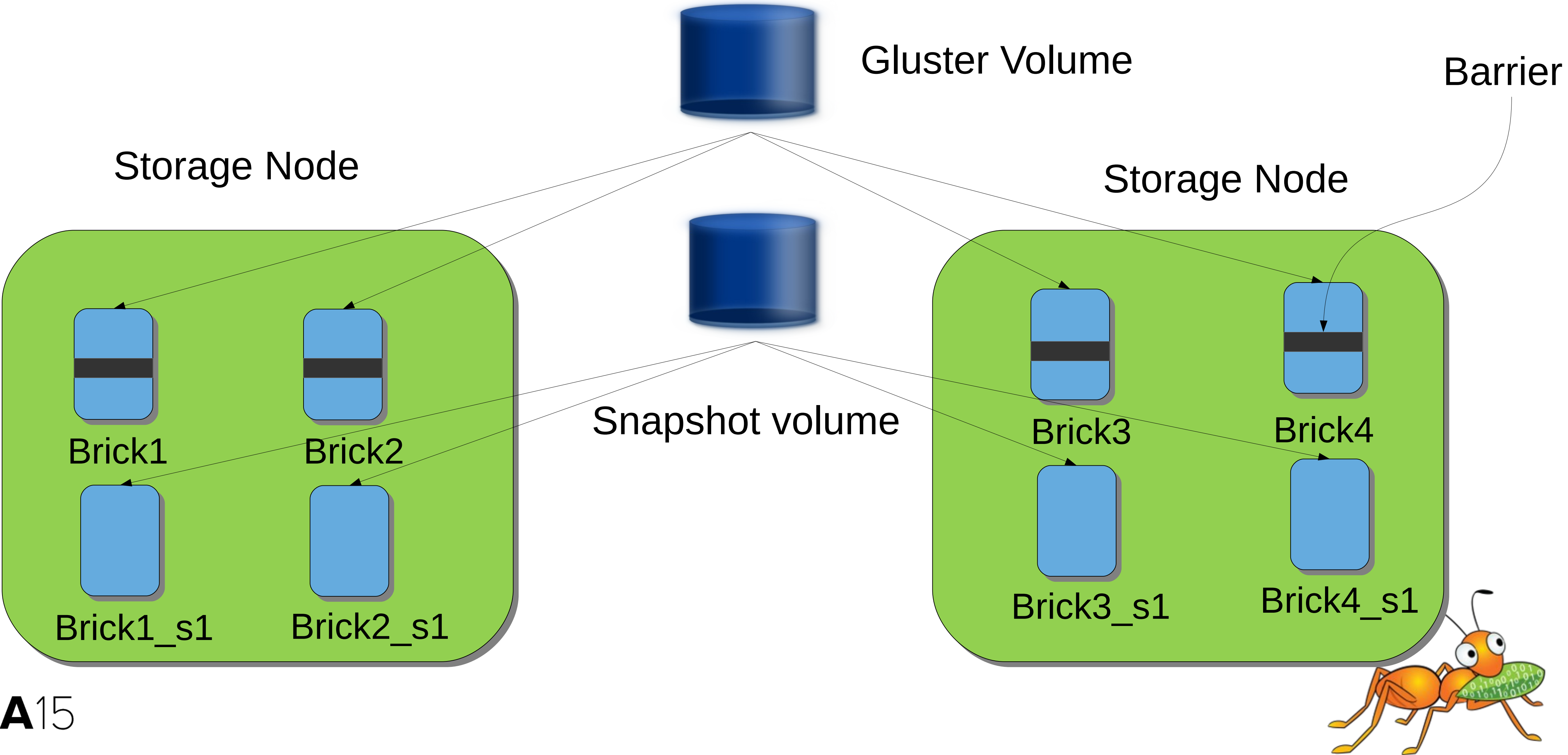




# Thinly Provisioned LVM2 Volume & Snapshot



# Gluster Volume Snapshot

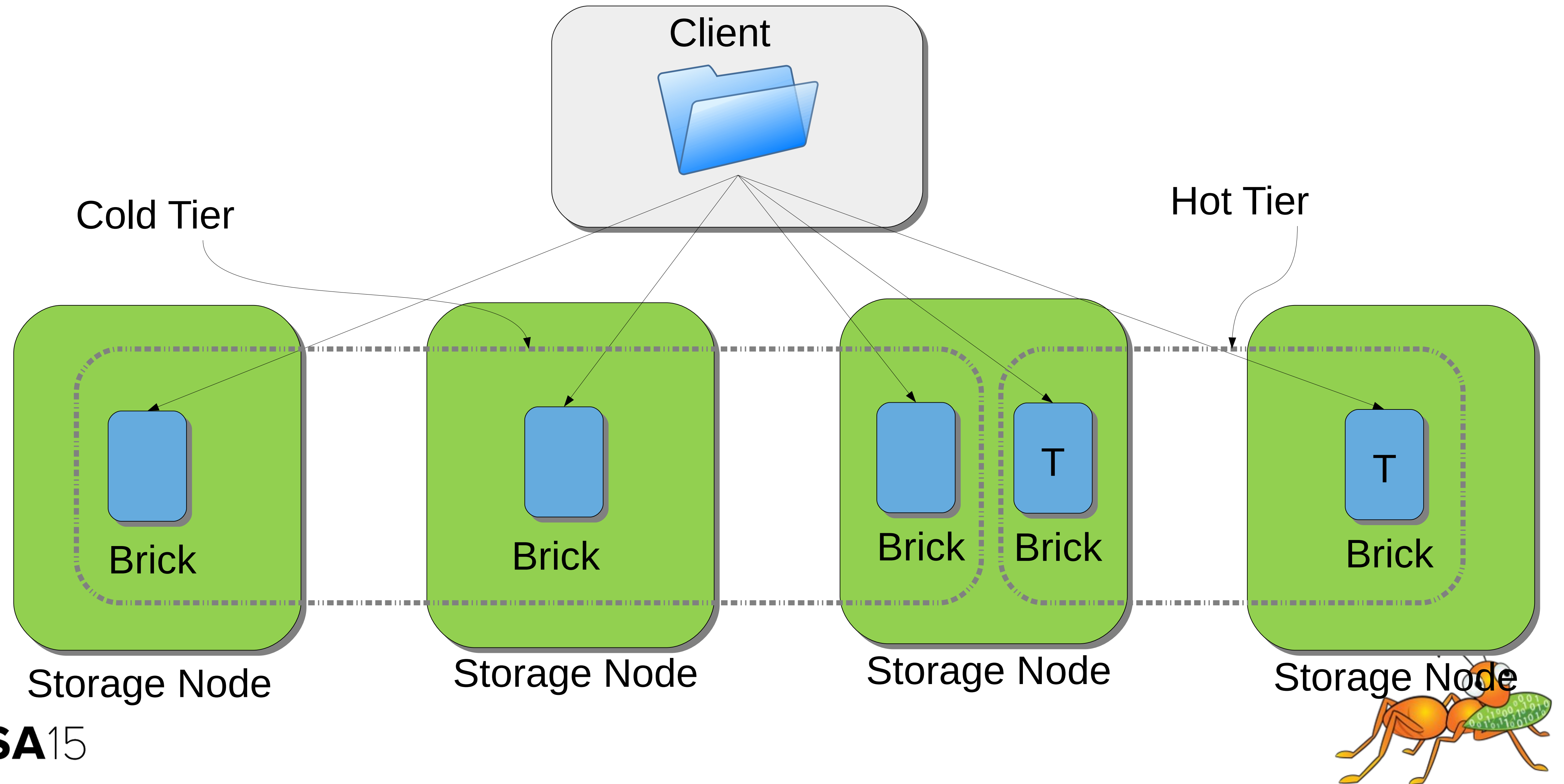


# Data Tiering

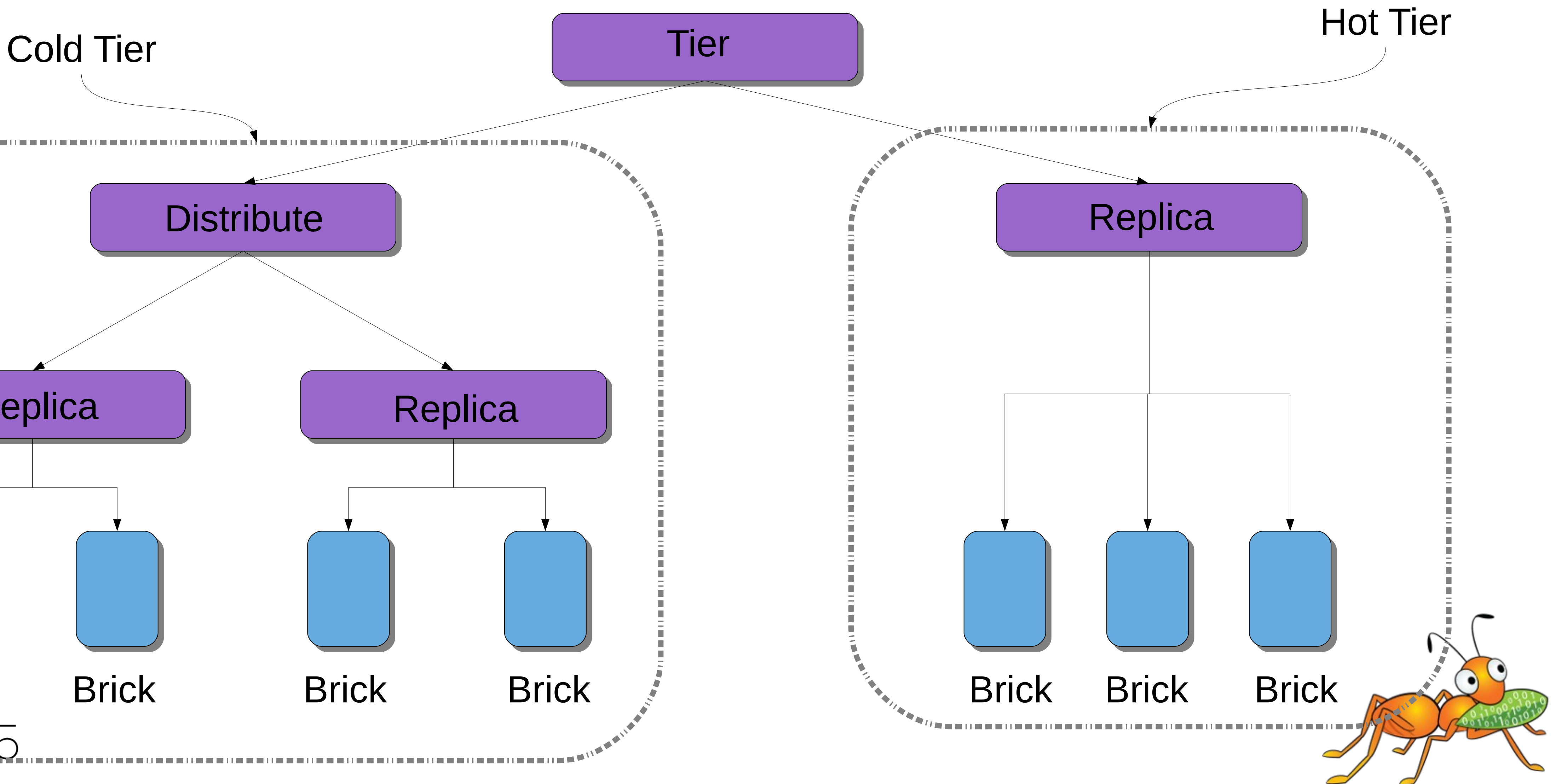
- ◆ Data classification and placement of data
  - ◆ Currently based on access rate
- ◆ Tiered volume can be created by attaching a tier
- ◆ Bricks in original volume are “cold” bricks
- ◆ Bricks in attached tier are “hot” bricks
- ◆ Hot bricks are normally hosted on faster disks e.g. SSDs
- ◆ Tier can be attached or detached using volume commands



# Tiered Volume



# Tiered Volume



# Future Work

- ◆ New Style Replication (NSR)
- ◆ Scalability improvements (1000+ Nodes)
- ◆ Multi-protocol support
- ◆ Multi-master Geo-Replication
- ◆ Caching Improvements
- ◆ Btrfs Support
- ◆ And many more...

