redhat.

Container-Native Storage

Solving the Persistent Storage Challenge with GlusterFS

Michael Adam
Manager, Software Engineering

José A. Rivera
Senior Software Engineer

2017.09.11

WARNING

The following presentation may contain opinions, speculations, and bad jokes. These are entirely the fault of the presenters, and do not reflect the values of Red Hat or its subsidiaries.

INTRODUCTION

Michael Adam

Manager, CNS Team
Maintainer, gluster-kubernetes

- Continues upstream contributions to various projects
  - gluster-kubernetes
  - heketi
  - Kubernetes
  - Gluster, gluster-block
  - Samba
  - tinyproxy
- Samba Team member since 2007

José A. Rivera

Developer, CNS Team
Project Lead, gluster-kubernetes

- Has facilitated integration of GlusterFS with various other projects
  - Samba
  - Pacemaker
  - Kubernetes
- Member of the Kubernetes Storage SIG
- Samba Team member since 2014

OUTLINE

INTRODUCTION
WHAT IS THIS?

YOU ARE HERE

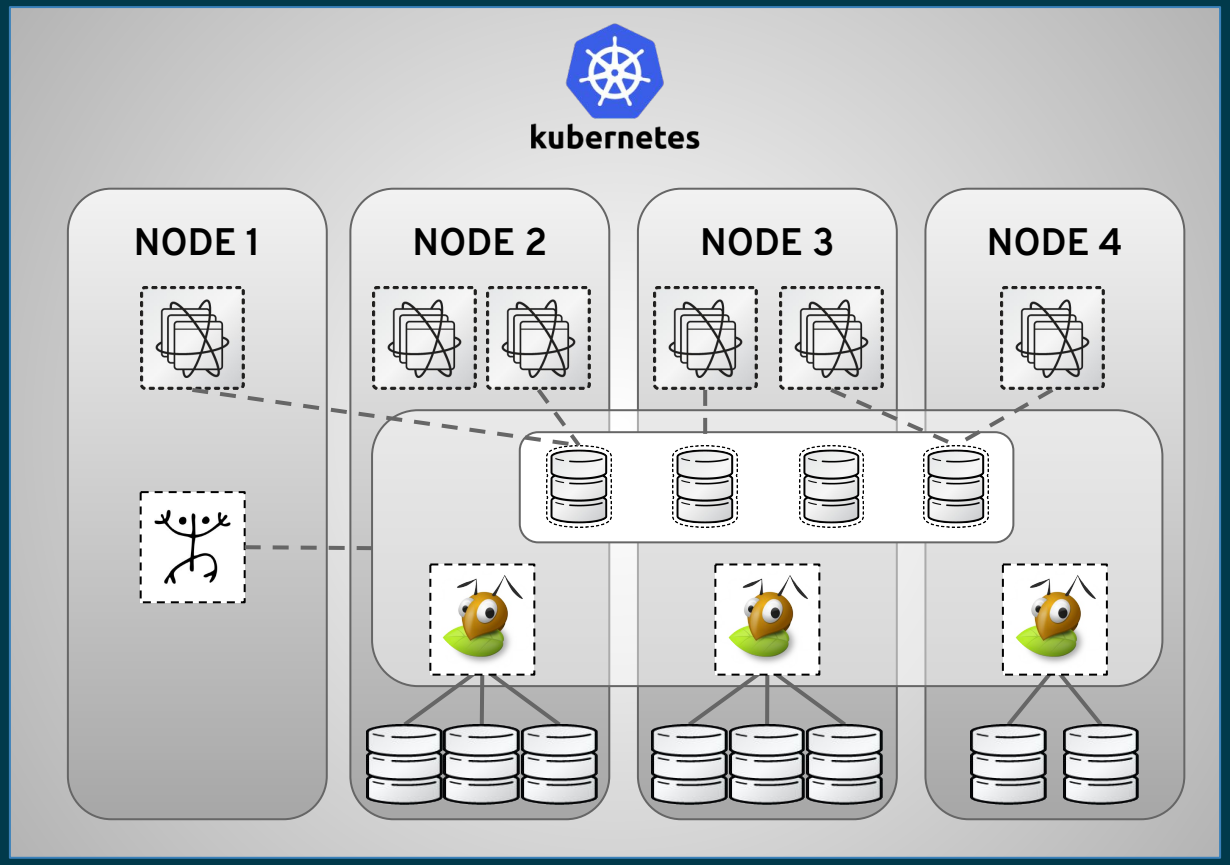
BACKGROUND
WHY DID WE DO THIS?

PUTTING IT ALL TOGETHER
HOW DID WE DO THIS?

ROADMAP
WHERE ARE WE GOING WITH THIS?

TL;DR:
CONTAINER-NATIVE STORAGE

Containerized persistent storage service for Kubernetes



BACKGROUND

CONTAINERS

Runtime isolation

Containers are a way to isolate processes in their own runtime environments on the same kernel with minimal overhead.

- Scaling at orders of magnitude higher than traditional VMs
- Faster deployment and startup
- Ideal for "microservices" application architecture
  - Run application components in isolation
  - Keep components "stateless"
  - Asynchronous communication (e.g. REST APIs)
  - Flexible maintenance and scaling of individual components

KUBERNETES

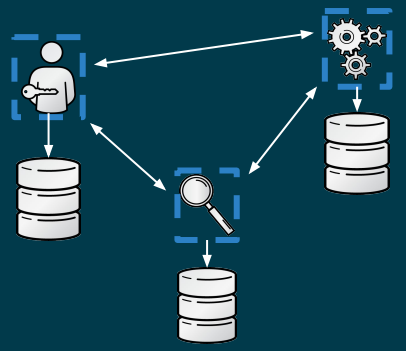
Container orchestration at scale

An open-source system for automating deployment, management, and scaling ("orchestration") of containerized applications.

- Clustered resource management
- Manages containers as "pods"
  - One or more containers and associated metadata
  - Used for scheduling, resource allocation, and network identity
- Calls programs "apps"
- Calls its users “developers”
- Supports multiple container runtimes
  - Docker, rkt, cri-o
- Abstracts away the underlying platform
  - On-premise (baremetal, VMs, ...) or in the cloud (AWS, GCE, Azure, ...)

CONTAINER STORAGE

Not completely stateless

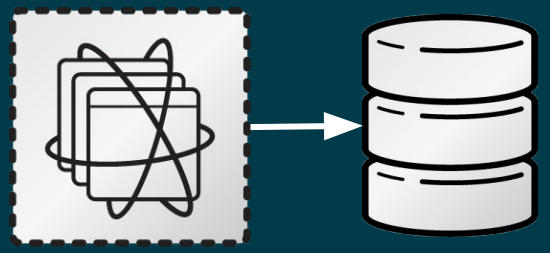


Containers are ephemeral in nature, using no actual storage on the host

- Destroying a container destroys its state
- All apps need some form of persistent storage
  - Content (e.g. websites)
  - Configuration
  - Databases
- Infrastructure services also need persistent storage
  - Container images
  - Clustered log data
- Storage must be able to follow apps

CONTAINER STORAGE

Accessible & flexible



To maintain flexibility for container scheduling, accessing storage over the network is the ideal solution

- Support multiple access protocols
  - iSCSI
  - NFS
  - SMB
- Support multiple kinds of raw storage
  - Cloud disks (e.g Amazon EBS, GCE Persistent Disk, AzureDisk)
  - SAN block devices
  - Local disks

CONTAINER STORAGE

So it's all great, right?

There was still no one solution that could provide:

- High availability
- Concurrent access
- Network resilience
- Ease of scale
- Abstraction of raw storage

We thought we had something better to offer...

GLUSTERFS

Software-defined, distributed filesystem

- Open-Source
- Software-defined storage
- Scale out / distributed file-volumes
- Highly available
- Easy to set up
- Easy to maintain
- Flexible
- Feature-rich

- https://gluster.org
- https://github.com/gluster/glusterfs

GLUSTERFS

Going further...

Why not containerize your storage solution?

- Monitor it as any other app
- Single control plane
- Integrate storage into Kubernetes nodes

PUTTING IT ALL TOGETHER

PERSISTENT VOLUMES

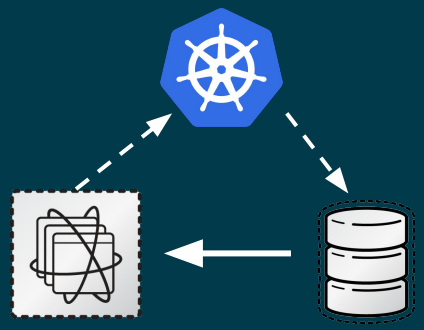
Abstracting storage

Kubernetes tries to abstract away storage from apps

- Storage volumes are called PersistentVolumes (PVs)
- Pods use PVs via PersistentVolumeClaims (PVCs)
  - PVCs are generic requests for an amount of storage with particular access modes
  - Kubernetes searches for a PV matching a given PVC
  - Matched PV is "bound" to the PVC
- Presents any storage as local within containers
- Allows for both static and dynamic creation of PVs

DYNAMIC PROVISIONING

Storage On-Demand



Kubernetes can create volumes for users upon request, without admin intervention.

- PVCs mention a particular StorageClass (SC)
  - Created by admin
  - Describes the storage type
  - Storage type is abstracted by SC
- SCs mention a Provisioner
  - Each type of storage has its own Provisioner
  - Creates a PV of appropriate size
  - PV is bound to incoming PVC

KUBERNETES + GLUSTERFS

Components

- In Kubernetes:
  - Gluster provisioner
  - GlusterFS mount plugin
- heketi: High-level REST interface for Gluster volume management
- GlusterFS
  - One or more Gluster clusters
  - Nodes running as containers inside Kubernetes
  - Can also run externally
- gk-deploy
  - Tool to deploy Gluster and heketi into an existing Kubernetes cluster

KUBERNETES + GLUSTERFS

Dynamic provisioning

- PVC created by user
  - GlusterFS provisioner is extracted from SC and invoked
  - Provisioner extracts details (type/size) about request from PVC
  - Provisioner tells heketi to create a corresponding Gluster volume
    - heketi looks for a cluster that can satisfy the request
    - If found, heketi creates a GlusterFS volume
    - heketi hands the volume back to the provisioner
  - Provisioner creates a PV with the GlusterFS volume details
- The PV is bound to the PVC

GLUSTERFS MOUNT PLUGIN

Kubernetes provides storage to containers by bind mounting directories into the containers

- GlusterFS mount plugin recognizes how to mount GlusterFS volumes onto the host
  - Ships with Kubernetes
- The Kubernetes HOST must be able to mount GlusterFS volumes
  - Must have mount.glusterfs installed

HEKETI

High-level service interface for managing GlusterFS volumes

- Hides nitty-gritty details from caller
  - Just takes Size and Durability type
- Recognizes commands like create, delete, expand, ...
- Can manage multiple Gluster clusters
- RESTful API and companion CLI tool ("heketi-cli")

https://github.com/heketi/heketi

CONTAINERIZATION

- heketi: A simple use case
  - Just a single usual application container
  - heketi DB needs to be persisted (currently in gluster volume...)
- Gluster: Slightly more complicated
  - Privileged
  - Consumes block devices of the nodes
  - Uses host network
  - Stores state directly on Kubernetes node

DEMOS

ROADMAP

How did we get here?

- On kube 1.4:
  - Dynamic provisioning
- On kube 1.5:
  - Improved scalability (# of volumes)
  - Openshift: support Gluster as registry backend
  - Day-2: remove disk

What's going on right now

- On kube 1.6 (will land this month):
  - Improved RWO support with gluster-block
  - Support logging (elastic search) on Gluster
  - Improved scalability (# volumes): brix mux
  - Day-2: remove node
  - Experimental S3-object access for apps

What will the future bring?

- On kube 1.7+
  - Integration into kubernetes primitives:
    - Snapshotting
    - Geo-replication
    - Volume expansion
    - ...
  - Support metrics (Cassandra / Prometheus) on Gluster
  - Fully support S3 object access
  - Reconcile heketi logic with Gluster proper (“gd2”)
    - ⇒ scalability, robustness, brown-field, ...
  - Use local block PVs when ready
  - StatefulSet for Gluster containers
  - Stretch cluster

QUESTIONS?

redhat.

THANKS!

https://github.com/gluster/gluster-kubernetes

Michael Adam <obnox@redhat.com>

 @obnoxXx  @obnoxXx

José A. Rivera <jarrpa@redhat.com>

 @jarrpa  @jarrpa