Hyper-converged, persistent storage for containers with GlusterFS

Mohamed Ashiq Liazudeen & José A. Rivera

0

Introductions & Agenda

Who are these guys and what are they talking about?

Introductions

Mohamed Ashiq Liazudeen
Associate Software Engineer, **Red Hat**

Ashiq is one of the maintainer of Gluster container and works on Hyper-Converged Gluster on Kubernetes and Openshift. Contributes to Heketi, Glusterfs, gluster-containers and gk-deploy.

José A. Rivera
Software Engineer, **Red Hat**

Part of Red Hat Storage, José works on integrating GlusterFS with container platforms and network file sharing protocols. Most recently, he helped develop a tool for deploying GlusterFS as a hyper-converged storage solution for Kubernetes and OpenShift.

José is a member of the Samba Team and the Kubernetes Storage SIG.

Agenda

1

Motivations

Why are we doing this?

Framing the Problem

Containers are ephemeral by nature, but many applications require storage that is persistent beyond the lifecycles of the application containers. However, robust storage solutions often require investments in proprietary hardware appliances and training.

We wanted to provide a persistent storage solution that:

- Requires minimal hardware investment
- Is as simple and transparent as possible to both administrators and users
- Is free and open source with a supportive community

Target Platforms

kubernetes

https://kubernetes.io/

OPENSHIFT

https://www.openshift.com/

Target Platforms

...because **Red Hat**

kubernetes

https://kubernetes.io/

OPENSHIFT

https://www.openshift.com/

Component Projects

GlusterFS

https://www.gluster.org/

heketi

https://github.com/heketi/heketi

GlusterFS

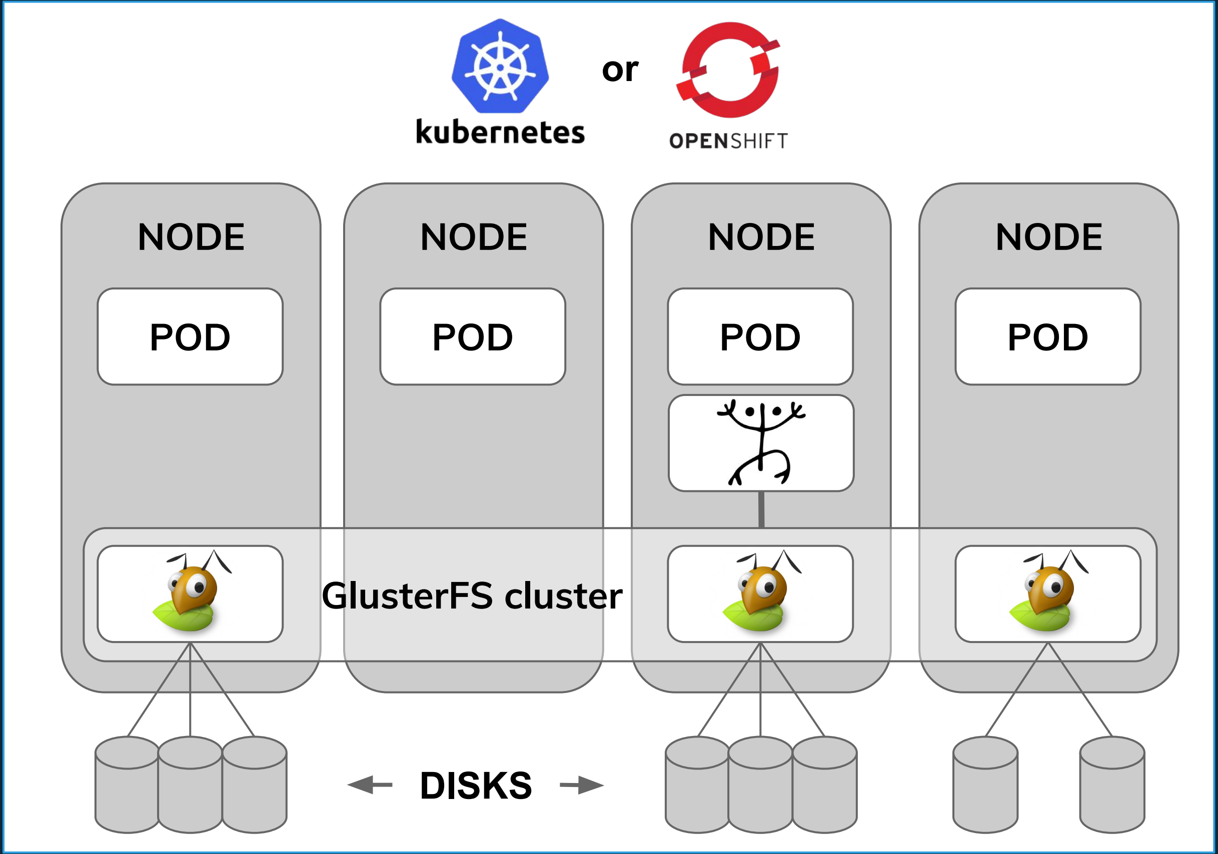
GlusterFS is a distributed, software-defined filesystem. Storage devices, called “bricks”, are selected on one or more nodes to form logical storage volumes across a Gluster cluster.

- Runs on commodity hardware (even a Raspberry Pi!)
- Scale-out design: easy to increase storage by simply adding more nodes
- Provides features like cross-node and cross-site replication, usage balancing, and iSCSI storage access

heketi

heketi is the RESTful volume management interface for GlusterFS.

- Allows for programmatic access to the most common GlusterFS volume management tasks
- Can manage multiple clusters from a single instance
- Lightweight, reliable, and simple


kubernetes
or
OPENSHIFT
NODE
NODE
NODE
NODE
POD
POD
POD
POD
GlusterFS cluster
DISKS

Gluing Pieces Together

You can find our work to bring together all these projects on GitHub, at **https://github.com/gluster/gluster-kubernetes**

- Documents how to put it all together
- Provides an easy-to-use deployment tool (gk-deploy)
- Has a quickstart guide for those who want to start playing with it right away

2

Creating Hyper-Convergence

Putting the pieces together.

Containerizing Gluster

Raw command for running a gluster container:

```
# docker run --name gluster -d -v
/etc/glusterfs:/etc/glusterfs:z -v
/var/lib/glusterd:/var/lib/glusterd:z -v
/var/log/glusterfs:/var/log/glusterfs:z -v
/sys/fs/cgroup:/sys/fs/cgroup:ro --net=host
--privileged=true -v /dev:/dev gluster/gluster-centos
```

Command for running GlusterFS pod in Kubernetes:

```
# kubectl create -f glusterfs-daemonset.yaml
```

Containerizing Gluster

The Gluster container required several pieces of configuration to get working:

- Containerized systemd
- Privileged container
- Startup script
- Bind mounts for persisting Gluster config
- Bind mount devices
- Access to the host network

Containerizing Gluster

We needed to make some changes to the standard way pods are usually deployed:

- **Containerized systemd:** Gluster has to run more than one process and needed someone to cleanup zombie processes.
- Running as a **privileged container** was required to run systemd and now needed to create logical volume from gluster container.
- We installed a **startup script** within the container:
  - Copy the Initial configuration
  - Gets the fstab maintained by heketi and mounts it on the pod for the brick process.

Containerizing Gluster

There were also some requirements from the fact that storage devices are bound to their nodes:

- We needed to create several **bind mounts for persisting Gluster config** on the host
  - /var/lib/glusterd – volume management files
  - /var/log/glusterfs – gluster log files
  - /etc/glusterfs – glusterd management files
- **Bind mount devices**: /dev has to be bind mounted to use the local storage disks.

Containerizing Gluster

Finally, **access to the host network** was also crucial:

- Gluster node IP needed to be constant
- Since the Gluster config is tied to the node, sharing the same network identity was needed
- Direct access gave better performance!

Containerizing heketi

Being a much smaller and newer application than Gluster, containerizing heketi proved easier:

◇ The image was easy to build

◇ We had an issue where we needed to authenticate through kube-api to access the Gluster pods. Using a service account and adding it to the deployment spec of heketi solved it!

◇ heketi stores configuration in a Bolt database, which must persist if the heketi pod goes down… wait, WE'RE persistent storage, let's put it in a Gluster volume! ;-D

3

Deployment & Usage

Look! It works!

Persistent Storage

The creation and usage of persistent storage in Kubernetes and OpenShift is enabled by the following:

- There are various **volume plugins** that allow Kubernetes/OpenShift to interface with different types of storage (e.g. cloud storage, network storage)
- There are two methods for creating persistent storage: **static provisioning** and **dynamic provisioning**. We'll be focusing on the latter
- A few resource types running in the cluster:
  - Persistent Volumes
  - Persistent Volume Claims
  - Storage Classes

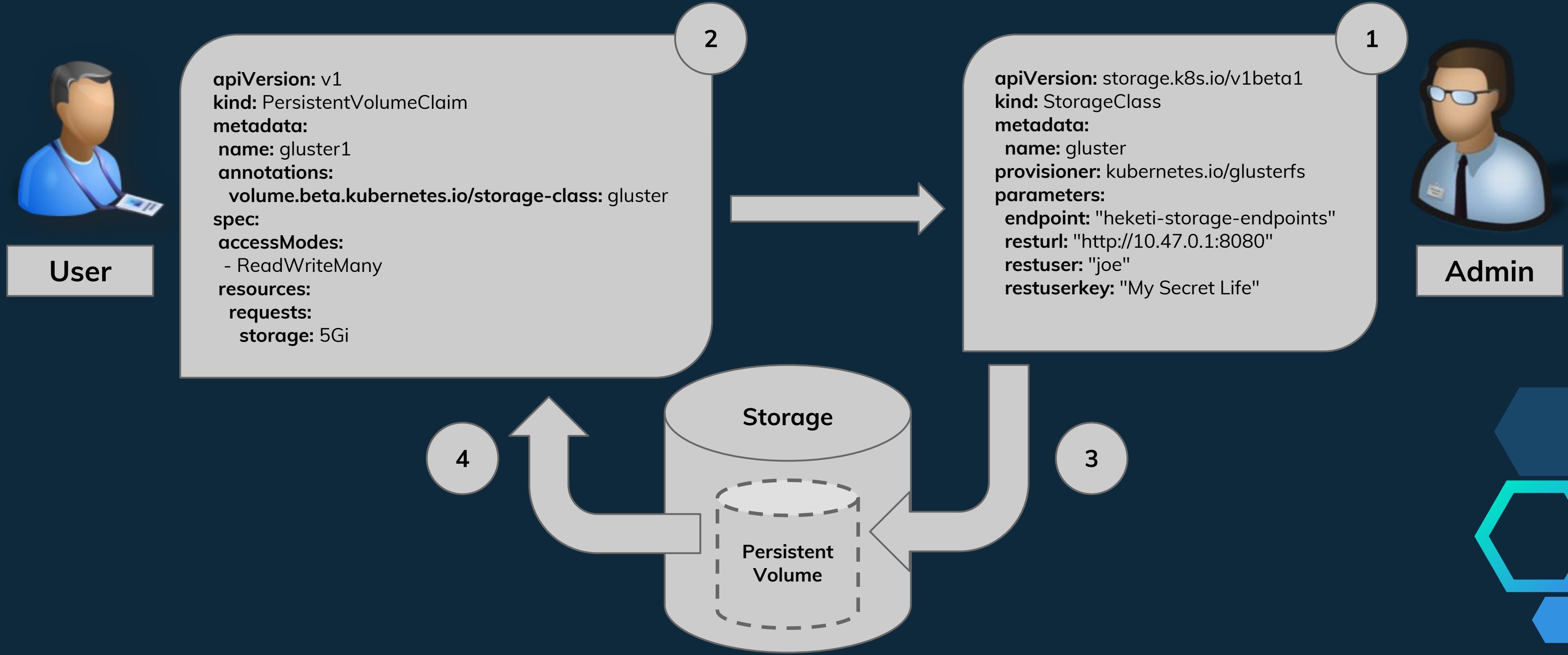
Dynamic Provisioning

The workflow for dynamically provisioning storage is as follows:

1. An administrator sets up some storage, then defines a **storage class** (SC) that describes the storage
2. A user creates a **persistent volume claim** (PVC) to request some storage of a given size, access type, and SC
3. A **persistent volume** (PV) is dynamically created of the requested size on some storage that matches the SC
4. The matched PV is then bound to the PVC and can be used by the user in pods.

The data in the PV persists beyond the lifecycle of the pod. When the PVC is deleted, the PV is released and dealt with as defined by the SC (e.g. is deleted)

Dynamic Provisioning



```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: gluster1
 annotations:
   volume.beta.kubernetes.io/storage-class: gluster
spec:
 accessModes:
  - ReadWriteMany
 resources:
   requests:
     storage: 5Gi
```

```
apiVersion: storage.k8s.io/v1beta1
kind: StorageClass
metadata:
  name: gluster
provisioner: kubernetes.io/glusterfs
parameters:
  endpoint: "heketi-storage-endpoints"
  resturl: "http://10.47.0.1:8080"
  restuser: "joe"
  restuserkey: "My Secret Life"
```

GlusterFS Storage

For hyper-converged GlusterFS storage, we also require a few additional resources:

- **Endpoints**: describes the list of IP addresses of the GlusterFS nodes, which can change over time
- **Service**: provides consistent access to the endpoints

And here are some configuration parameters within our SC:

```
resturl: “http://127.0.0.1:8081”
clusterid: “630372ccdc720a92c681fb928f27b53f”
restuser: “admin”
secretName: “heketi-secret”
gidMin: “40000”
gidMax: “50000”
```

Full Hyper-Convergence

Now, both GlusterFS and heketi run in containers on your Kubernetes or OpenShift cluster.

- Requires some additional administrative changes, but greatly reduces hardware costs
- Applications have native access to GlusterFS-backed storage via heketi
- The GlusterFS containers don't have to run on all nodes, they can be set to run only on nodes that can fulfill its storage needs
- Easy to scale out

DEMOS!

Sorry online readers. ;)

Thanks!

Find **Ashiq** at:

- **@MohamedAshiqrh** on GitHub
- mliyazud@redhat.com

Find **José** at:

- **@jarrpa** on GitHub and Twitter
- jarrpa@redhat.com

Projects:

- **GlusterFS** - https://www.gluster.org
- **heketi** - https://github.com/heketi/heketi
- https://github.com/gluster/gluster-kubernetes
- https://github.com/gluster/gluster-containers