GLUSTER CONTAINER STORAGE

STORAGE FOR CONTAINERS, IN CONTAINERS

Kaushal

OVERVIEW

- What is GCS?
- GCS in detail
- Try GCS!

Speaker notes

I'll be calling Gluster Container Storage GCS through the rest of the presentation. In the K8S world, GCS already stands for something else, so this is a temporary term and we're looking for a new name.

GLUSTER CONTAINER STORAGE

GLUSTER CONTAINER STORAGE

- Provides persistent storage for your containers
  - RWX & RWO volumes
  - *Kubernetes only for now*
- Uses industry standard container interfaces
- Hyperconverged deployment

Speaker notes

New effort from the Gluster community to provide persistent storage for containers.
Provides shared storage and exclusive storage volumes.
Implements CSI. Allows GCS to provide storage for different containers orchestrators. But initially targeting K8S.
GCS is deployed on the same nodes that run you containers.
In K8S GCS is deployed as pods using K8S native mechanisms.
Not normal app containers but privileged.

GLUSTER CONTAINER STORAGE

- Brings together
  - GlusterFS
  - GlusterD2
  - Gluster CSI Driver
  - Anthill
  - Gluster-prometheus
  - Gluster-mixins

Speaker notes

GCS brings together many different Gluster related projects.
GlusterFS - which everyone knows about already
GlusterD2 - the newer management framework for Gluster, being specifically made to work for modern usecases like containers.
CSI Driver - Which implements the CSI spec for GlusterFS
Anthill - A K8S operator which manages Gluster
Gluster-prometheus & Gluster-mixins improve gluster monitoring.

WHY IS GCS GOOD FOR YOU?

- Simplified stack
  - *Tuned for container workloads*
- Opinionated defaults
- Automated management
- Better scale
  - 2000 RWX/5000 RWO

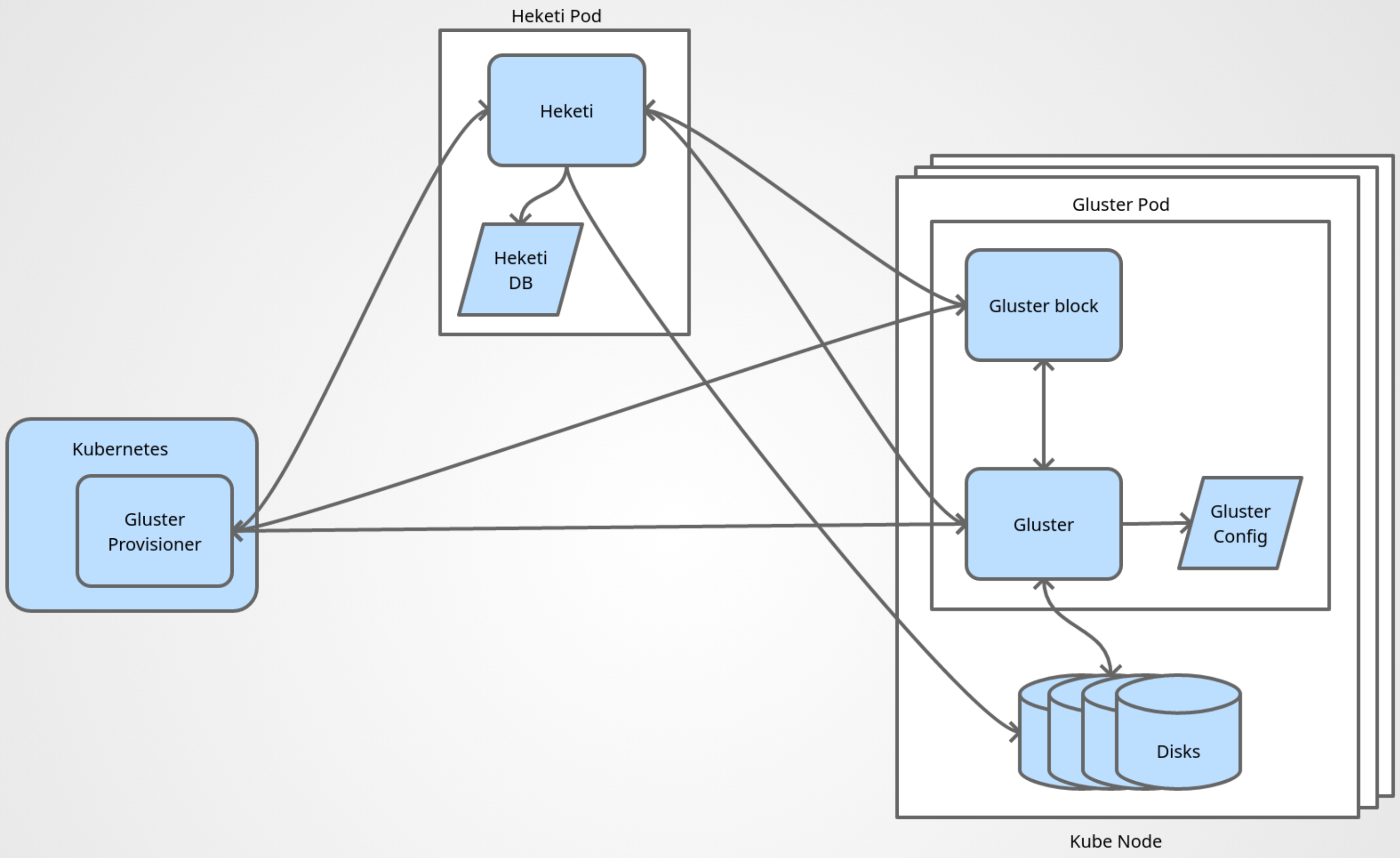
Speaker notes

GCS simplifies the overall.
Simplified container deployment.
Simpler workflow.
Simpler GlusterFS stack too.
All tuned to ensure they work well in the container ecosystem.
GCS makes opinionated decisions about default options to ensure that the standard deployment works for most users.
All operations are automated - deployment, day-1/day-2 operations, upgrades.

CURRENT STACK

Speaker notes

GCS is the new upcoming container story for Gluster.
But we already do storage for containers.
So let's take a look at what exists.



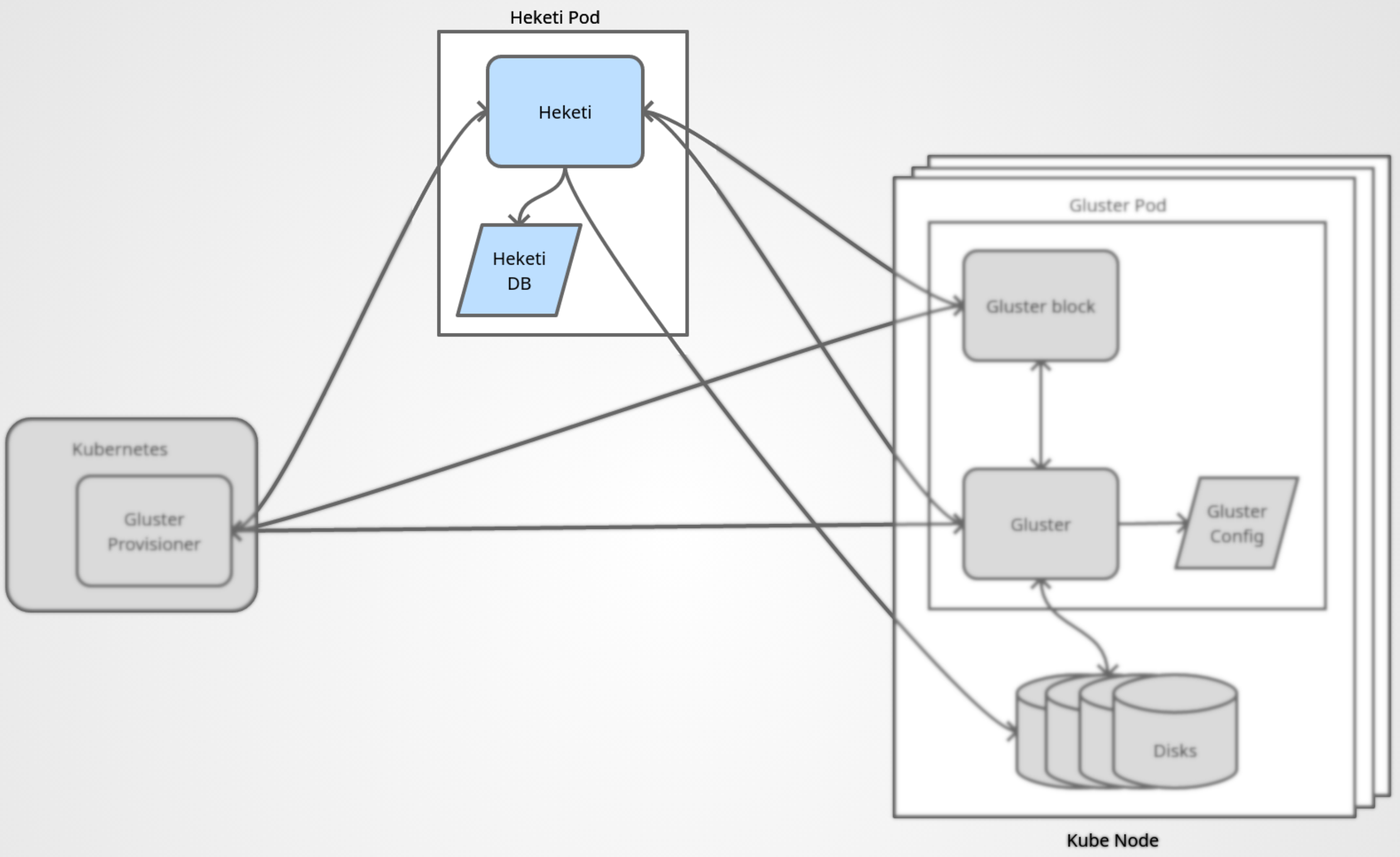
HEKETI BASED STACK

Speaker notes

Current Gluster container stack revolves around Heketi. This stack is currently supported by Red Hat as OCS.
After deployment the stack looks like this. I'll go quickly through the stack a in a couple of moments.

This stack looks simple but it not really
It has a relatively complex deployment which is handled by Ansible (gdeploy or openshift-ansible).

And there are other issues that with the architecture that make it quite complex to manage.

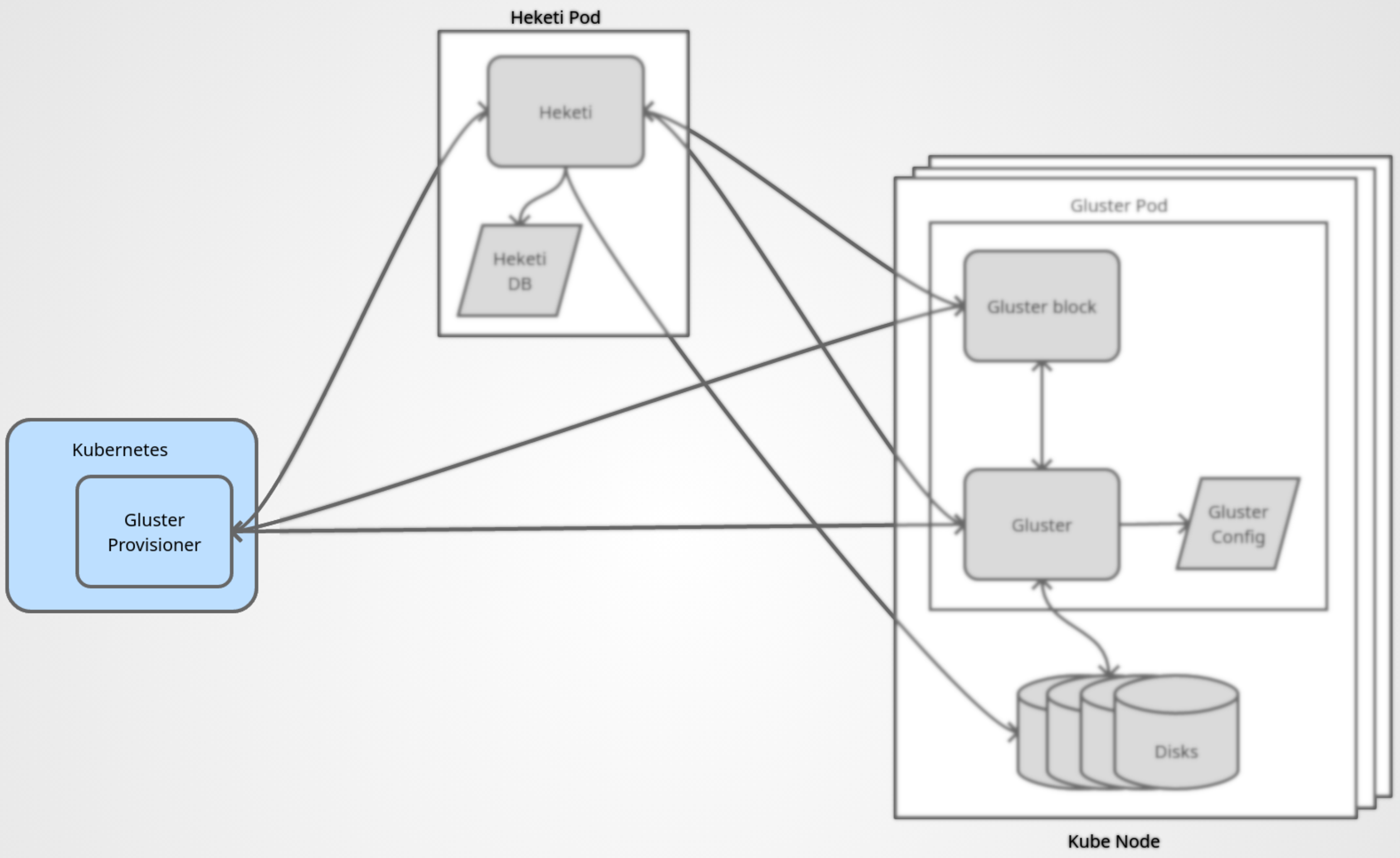

Heketi Pod
Heketi
Heketi DB
Gluster Pod
Gluster block
Gluster
Gluster Config
Disks
Kube Node
Kubernetes
Gluster Provisioner

HEKETI

Speaker notes

Let's start with Heketi.
Heketi provides an API for automated Gluster cluster and volume management.
It forms a Gluster pool. Manages disks and provisions them for bricks. Creates volumes from provisioned bricks.

One thing to note is that Heketi maintains it's own state of the cluster, separate from Gluster. This is the heketi DB.
Fun fact, heketi DB itself is kept on a Gluster volume from the Gluster pool Heketi manages. A few tricks are performed during deployment to achieve this.

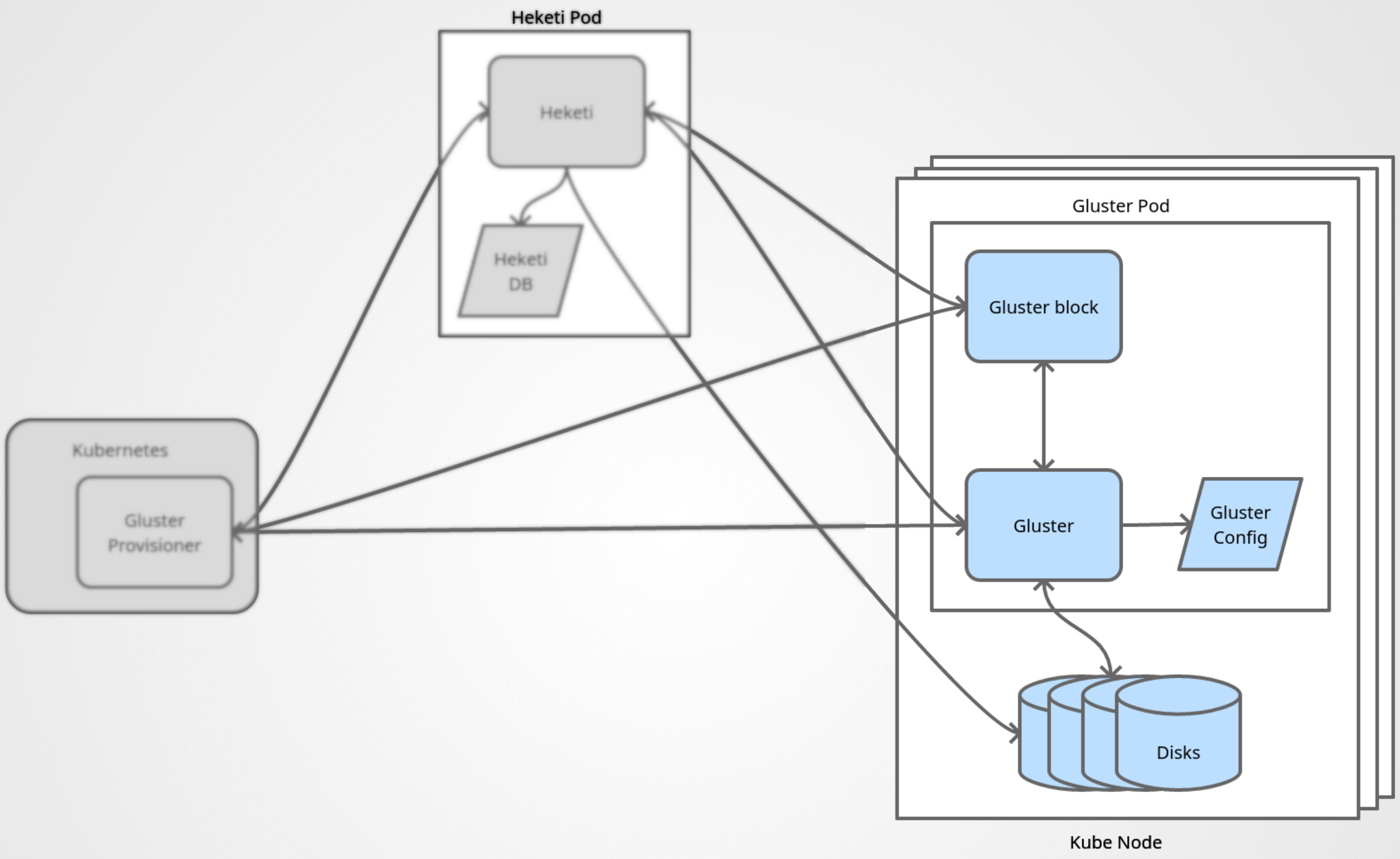


IN TREE GLUSTER PROVISIONER

Speaker notes

There is a Gluster provisioner in the K8S tree, that listens for Gluster PVCs and talks with heketi to provision the requested volume. Then it also mounts the created gluster volume where required.

While this works, it is not the most flexible solution.
The provisioner is K8S only.
And to make changes to it, we need to go through the K8S development process, and then wait for the next K8S release for it to become available. Granted the next release is not that far away. But we're tied to the K8S release cycle.

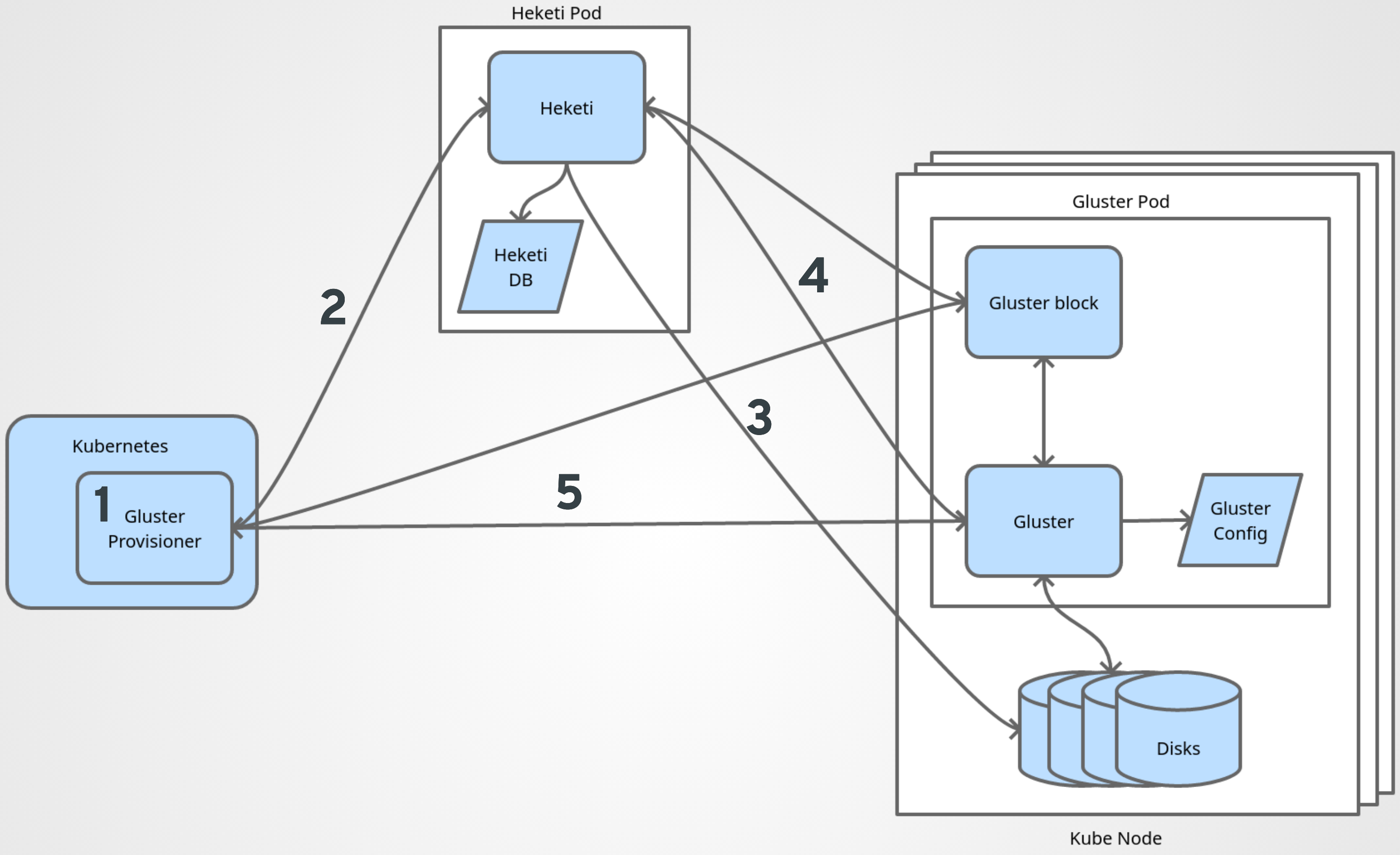


GLUSTER & GLUSTER BLOCK

Speaker notes

The 3rd part is the Gluster pod. It runs on each K8S node that has storage that needs to be exported.
The pod contains GlusterFS, and the Gluster-block daemon.
GlusterFS volume provide RWX PVs.
Gluster-block provides RWO volumes.

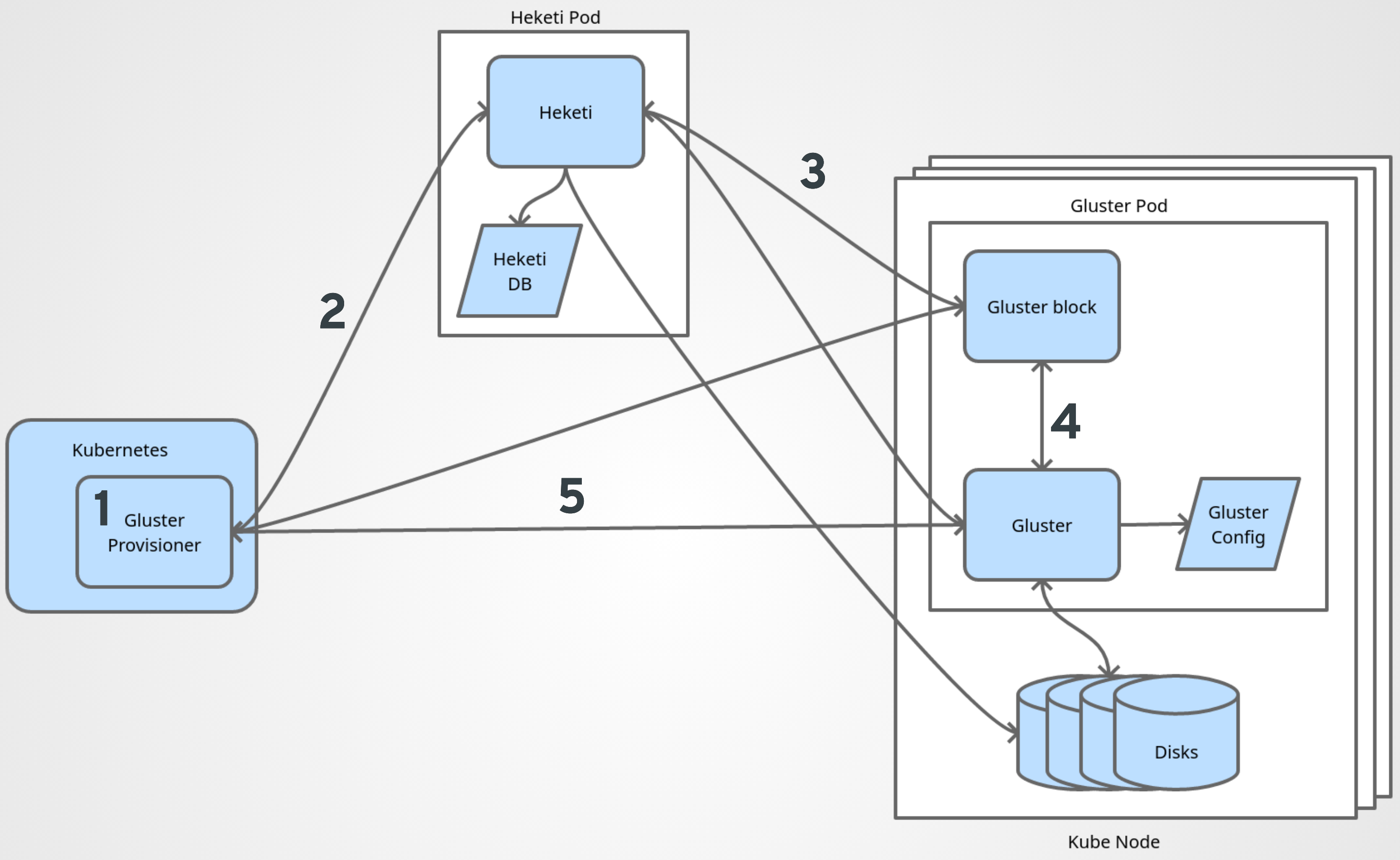
Note here that the Gluster cluster formed by the gluster pods have their own cluster configuration. This is seperate from Heketi.



RWX PERSISTENT VOLUME

Speaker notes

Let's see how a PVC request is processed. For a RWX volume.
1. Request reaches the in-tree Gluster provisioner
2. Gluster-provisioner then reaches out to heketi with a volume create request.
3. Heketi figures out the disks to be used for the volume. Then prepares the disks by `exec`ing into the gluster pods on the required nodes.
4. Once the disks are prepared, heketi then creates the volume. Again by execing into the gluster pods and running gluster volume create.

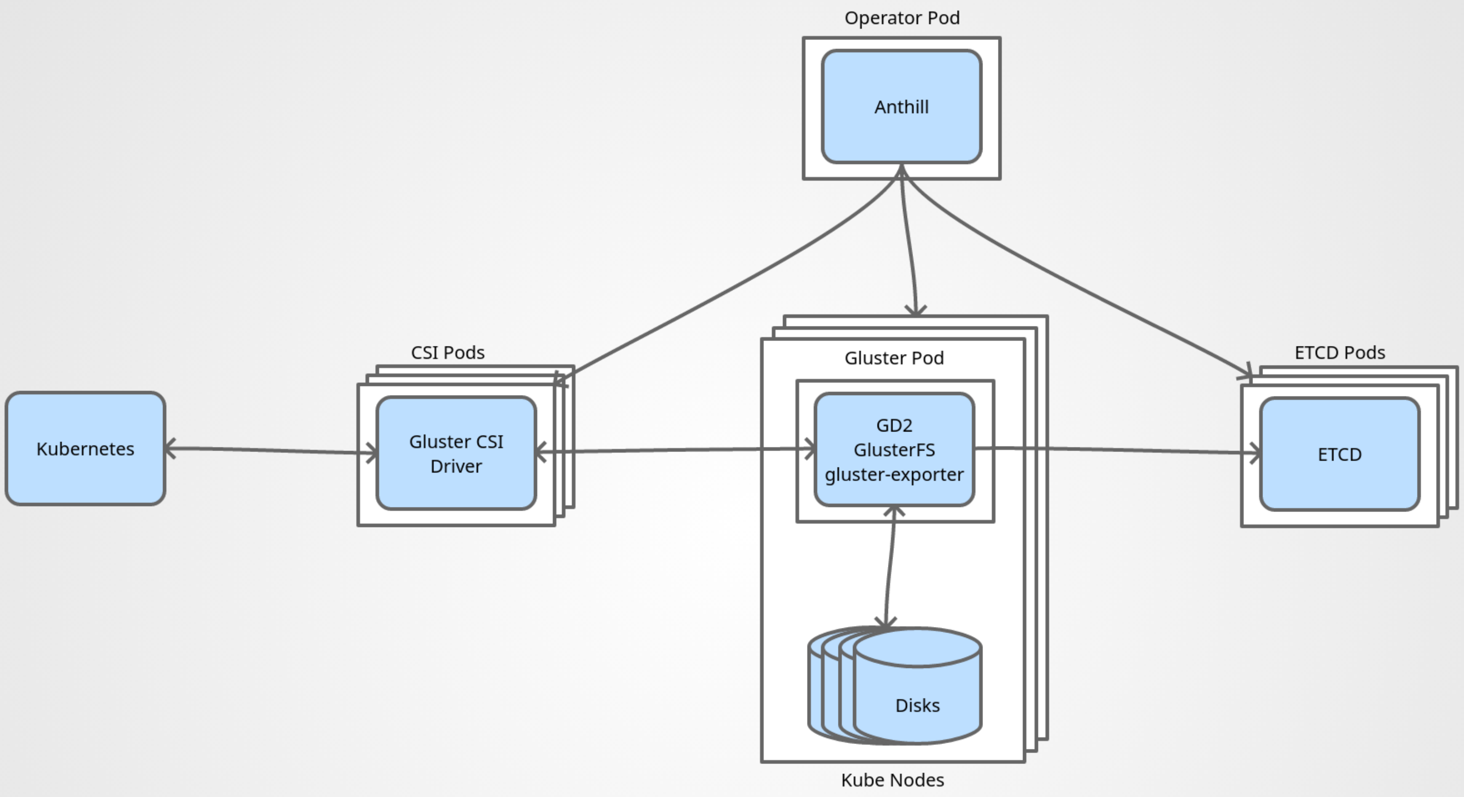


RWO PERSISTENT VOLUME

Speaker notes

RWO volumes happen nearly the same. But Heketi reaches out to Gluster-block to create and export a block device. Gluster-block then creates the block device file on a previously provisioned gluster-volume. This volume is created during deployment.

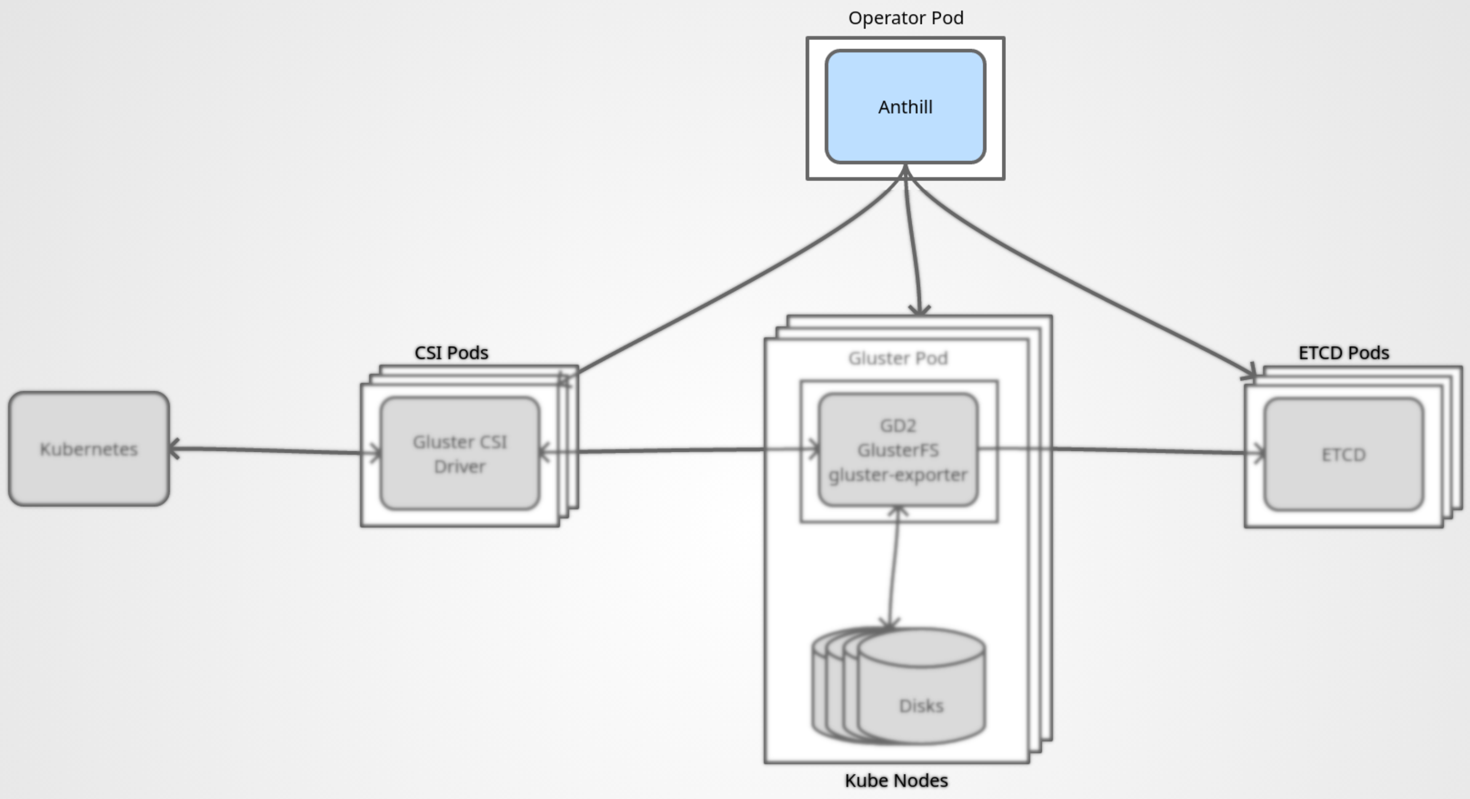
GCS - THE DETAILS


Operator Pod
Anthill
CSI Pods
Gluster CSI Driver
Kubernetes
Gluster Pod
GD2
GlusterFS
gluster-exporter
Disks
Kube Nodes
ETCD Pods
ETCD

GCS STACK

Speaker notes

Initially does look like it has more parts than Heketi. But it is actually a simpler stack, that actually does more.

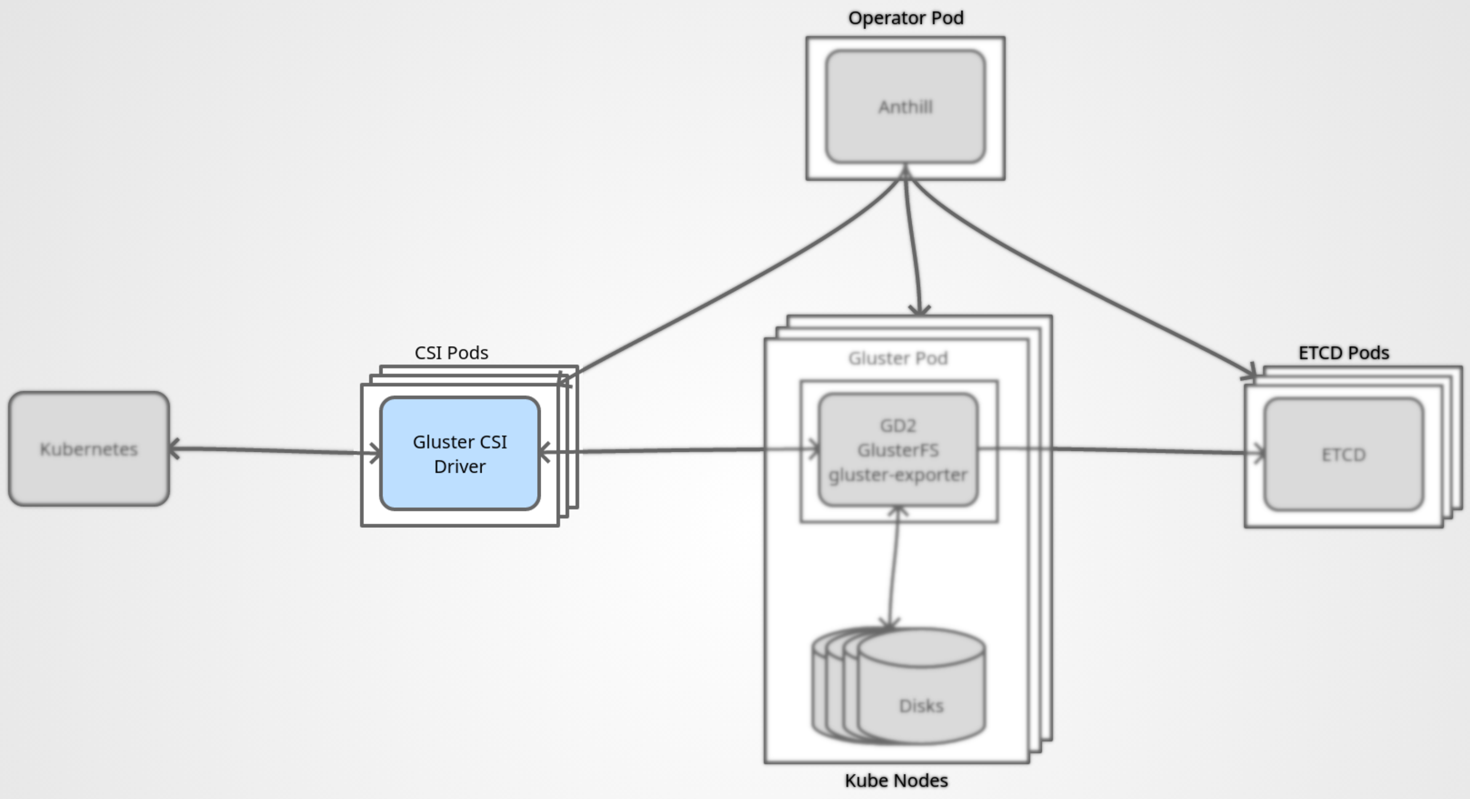


ANTHILL OPERATOR

Speaker notes

Let's start with Anthill.

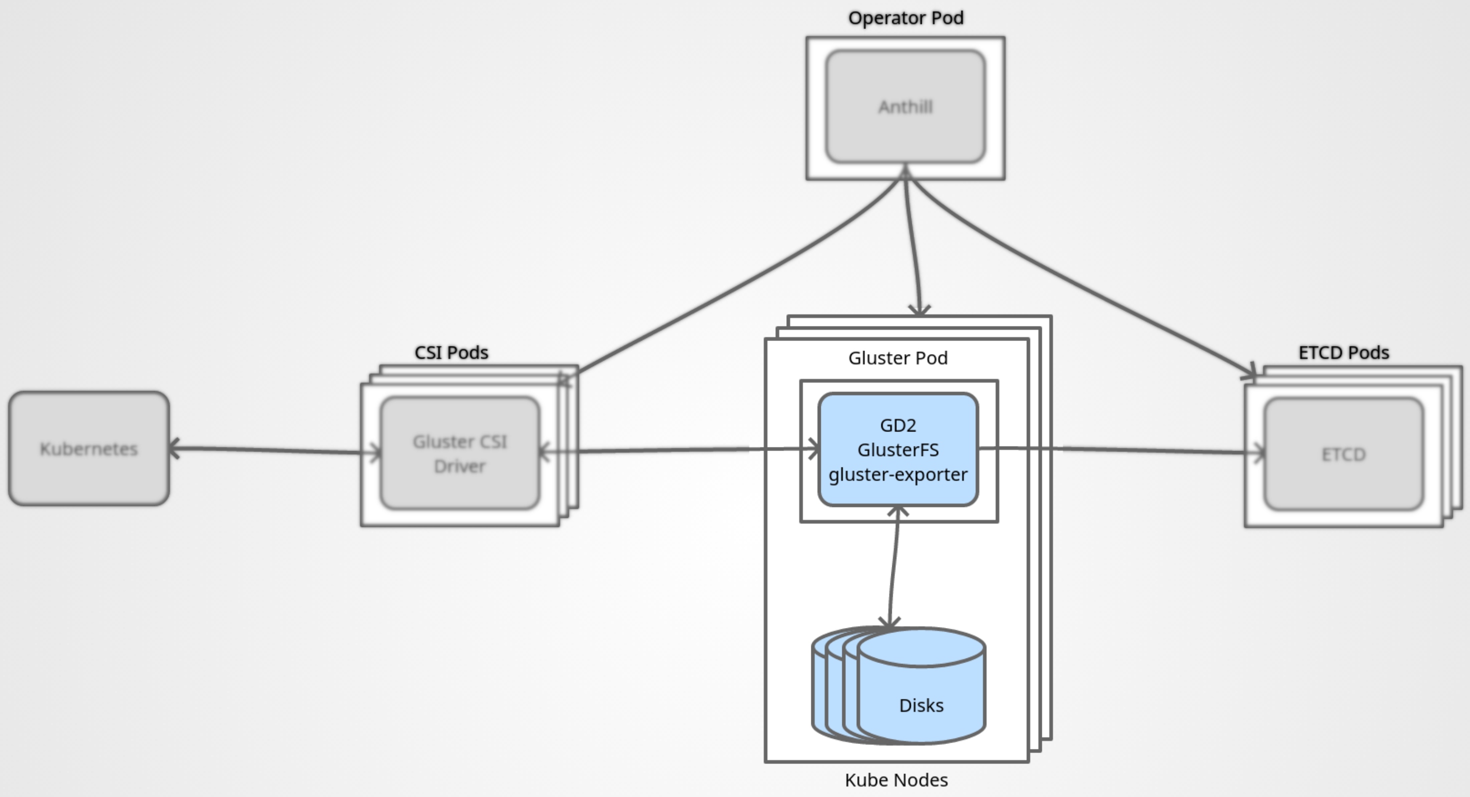
Anthill is a kubernetes operator. It takes care of deploying and managing the rest of the GCS stack. All of which is automated.



GLUSTER CSI DRIVER

Speaker notes

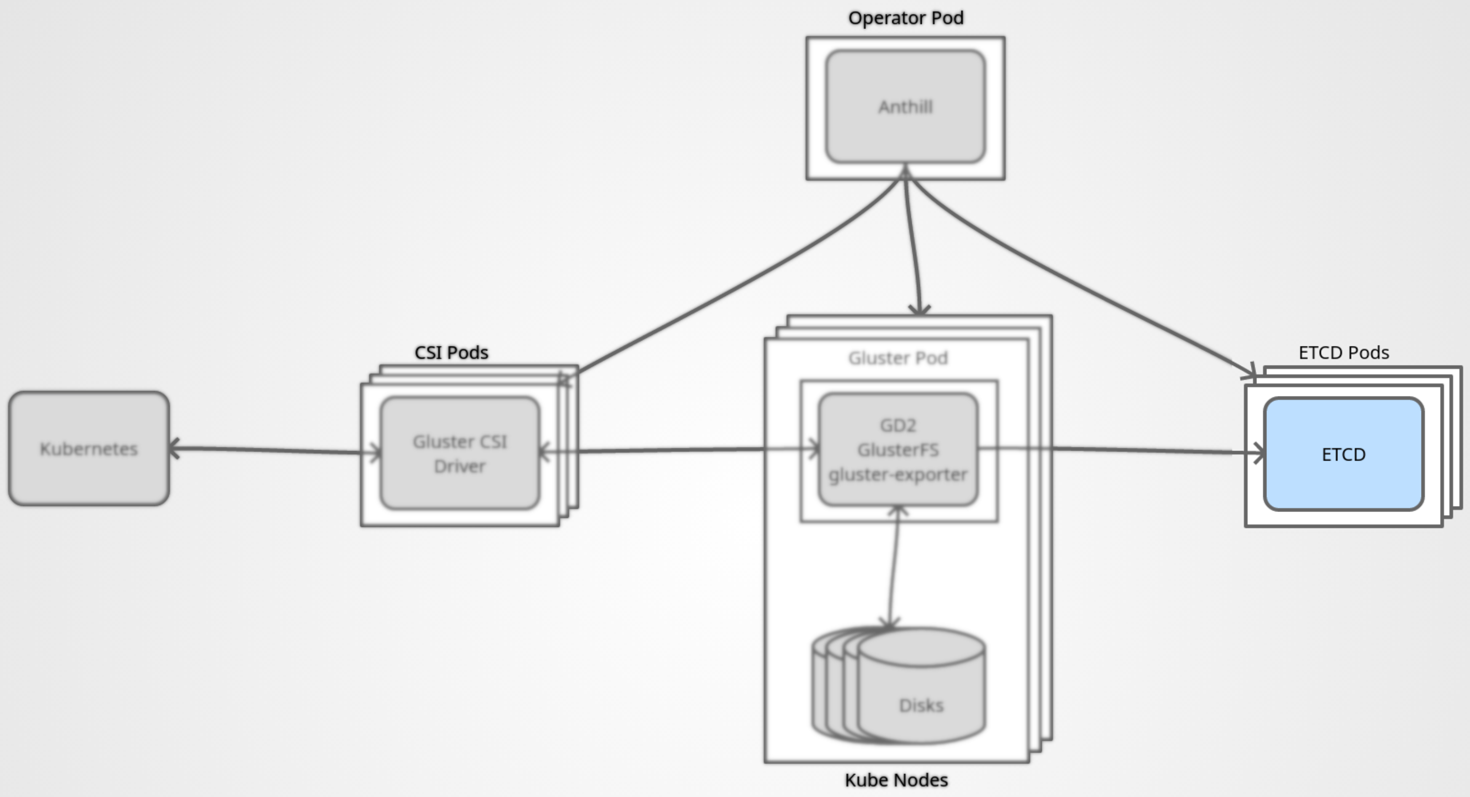
CSI driver implements the Container Storage Interface. A standardized interface that is being adopted across the container orchestrators. CSI driver works with GlusterD2 to satisfy PV requests from K8S.



GLUSTERD2 & GLUSTERFS

Speaker notes

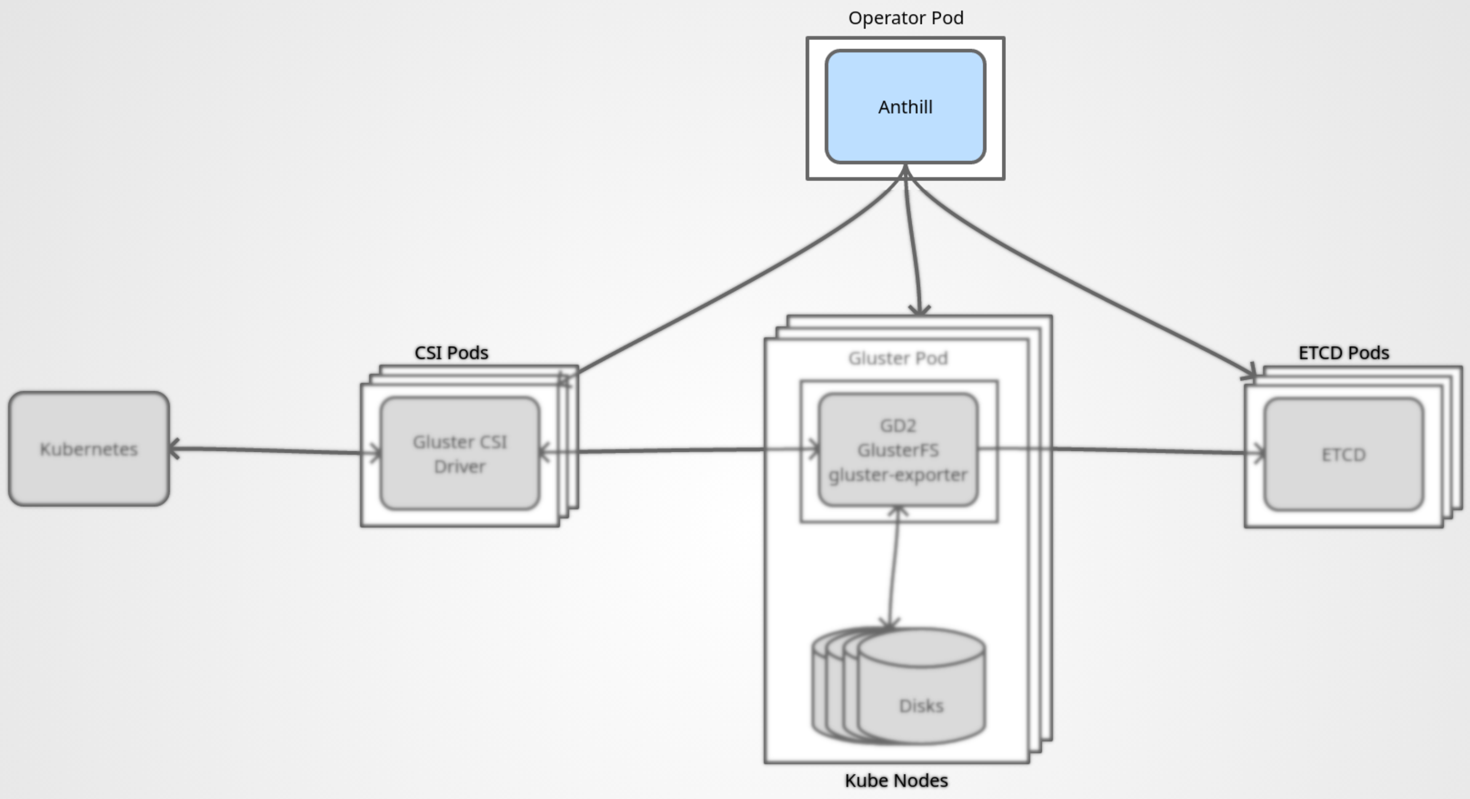
The Gluster pod in GCS contains GlusterFS along with GD2. It also contains the gluster prometheus exporter.


Operator Pod
Anthill
CSI Pods
Kubernetes
Gluster CSI Driver
Gluster Pod
GD2
GlusterFS
gluster-exporter
Disks
Kube Nodes
ETCD Pods
ETCD

ETCD

Speaker notes

Anthill will deploy an etcd cluster as well. This etcd cluster is used by GD2 as it config store. This is the only location of cluster config in GCS.



ANTHILL OPERATOR

ANTHILL OPERATOR

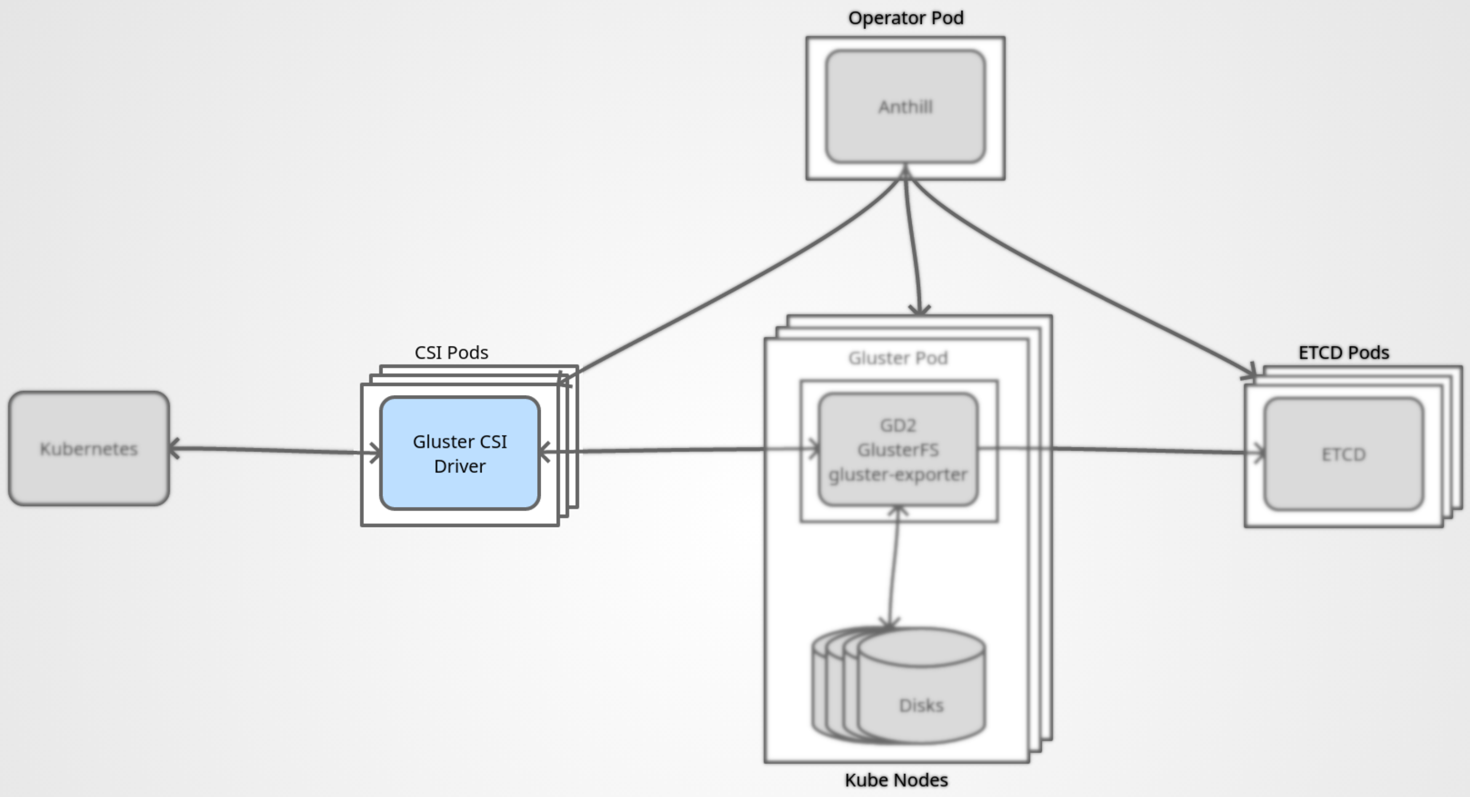
- Kubernetes Operator
  - Custom GCS resource definitions
- Automates GCS
  - deployment
  - upgrades
  - day-to-day management

Speaker notes

Anthill is a K8S operator which defines custom GCS resource types. Once Anthill is fully developed, all users would need to do is to deploy Anthill and create a manifest for the needed CRD.

ANTHILL STATUS

- Very early stage
  - CRDs defined
- github.com/gluster/anthill



GLUSTER CSI DRIVER

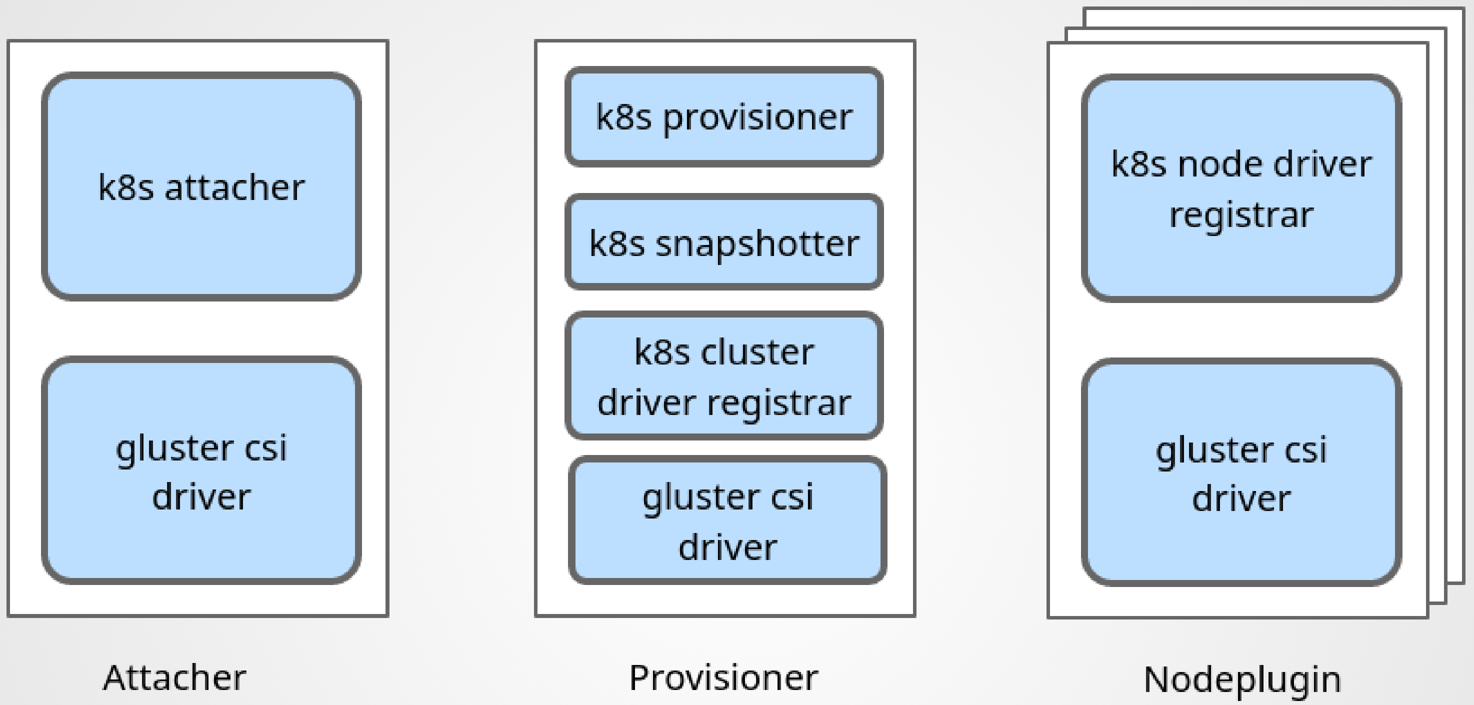
GLUSTER CSI DRIVER

- Translates K8S CSI requests to GD2 ReST requests
- Manages mounting of Gluster volumes
- Three different types of pods
  - nodeplugin
  - provisioner
  - attacher

Speaker notes

CSI driver as mentioned implements the CSI spec.
CSI is a container standard spec that defines how container orchestrators can request for persistent volumes from storage providers.

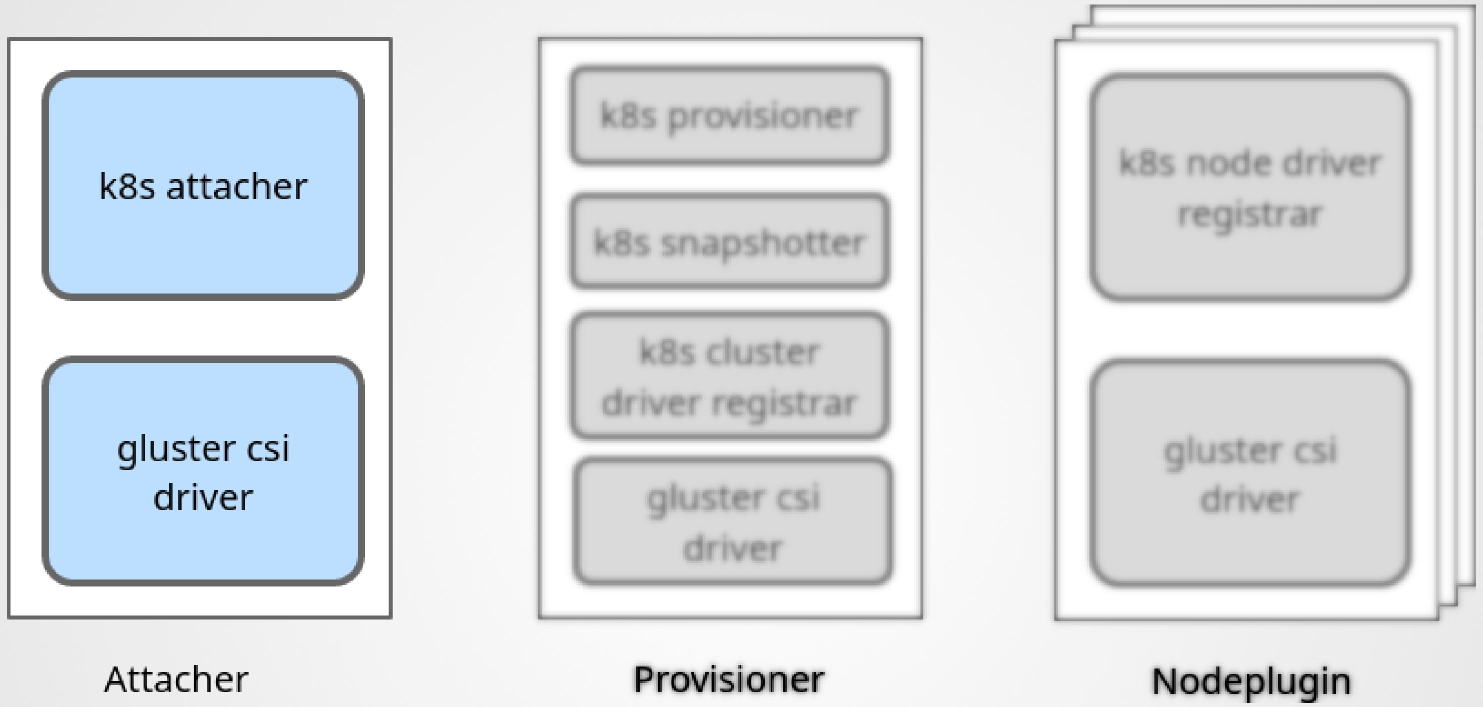
The gluster-csi-driver translates the K8S CSI requests to GD2 rest requests.
It is deployed in 3 different types of pods, but all essentially have the same gluster-csi-driver application.



CSI DRIVER PODS

Speaker notes

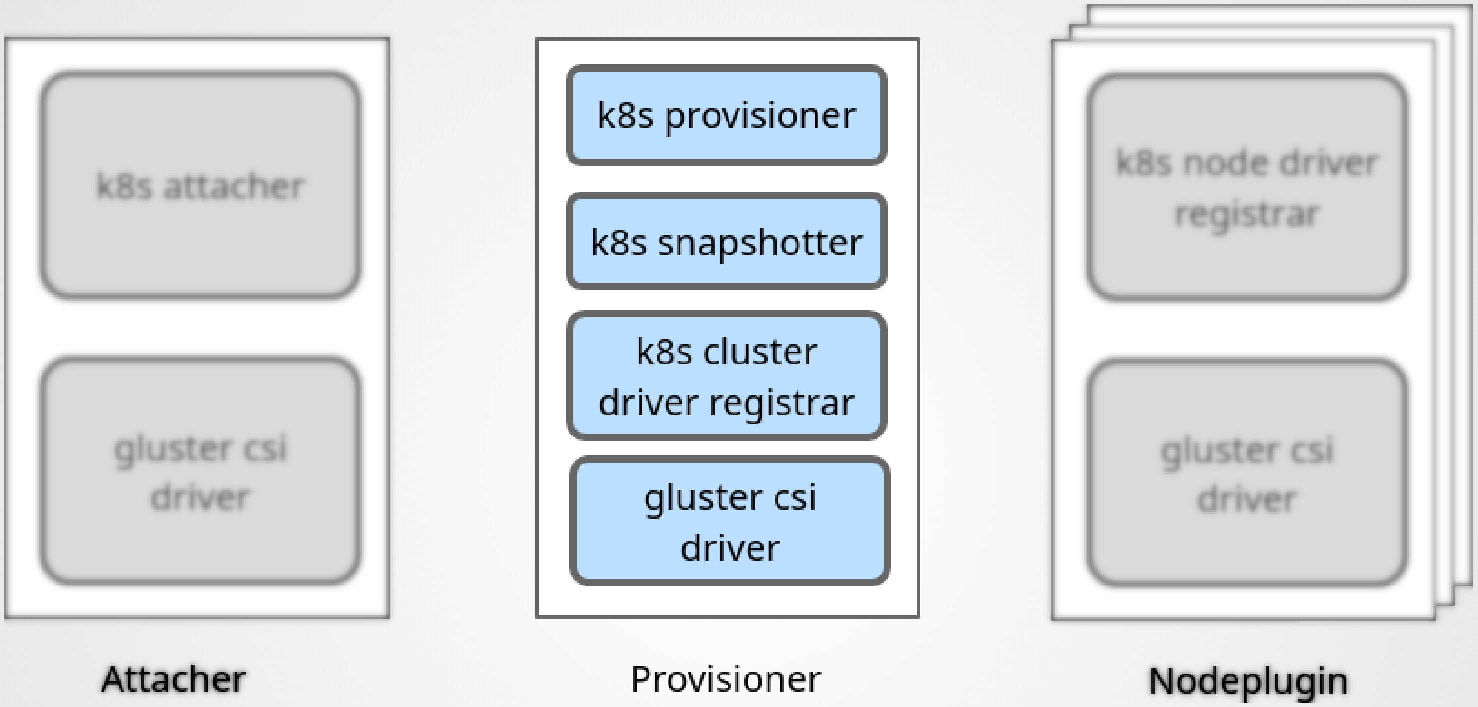
Let's take a look at the 3 types of CSI driver pods.



CSI ATTACHER

Speaker notes

Attacher is basically a no-op pod for GCS, it just gets deployed because the K8S CSI implementation expects it.
Work is ongoing in the K8S community to remove this need.

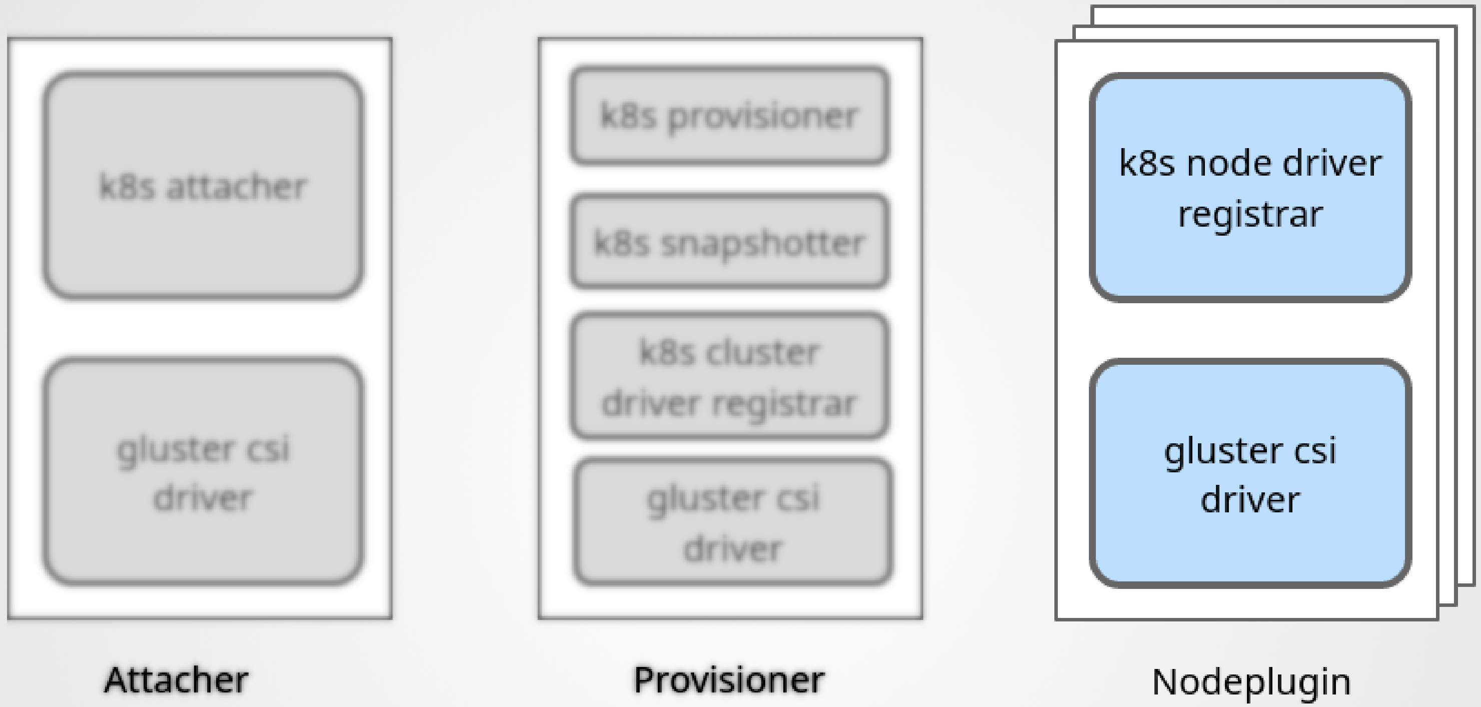


CSI PROVISIONER

Speaker notes

The provisioner pod primarily listens for the PVC create/delete requests from K8S and translates them to volume create/delete requests to GD2.

Made up of 4 pods,
- a k8s provisioner sidecar
- a k8s snapshotter sidecar
- a k8s cluster driver registrar



CSI NODEPLUGIN

Speaker notes

The Nodeplugin pod runs on each K8S node and handles the mounting of the create Gluster PVs.

The node driver-registrar sidecar registers the pod as a CSI driver on the K8S node. This ensures that the mount requets for the gluster-csi PV type reach the gluster nodeplugin, which then does the mount.

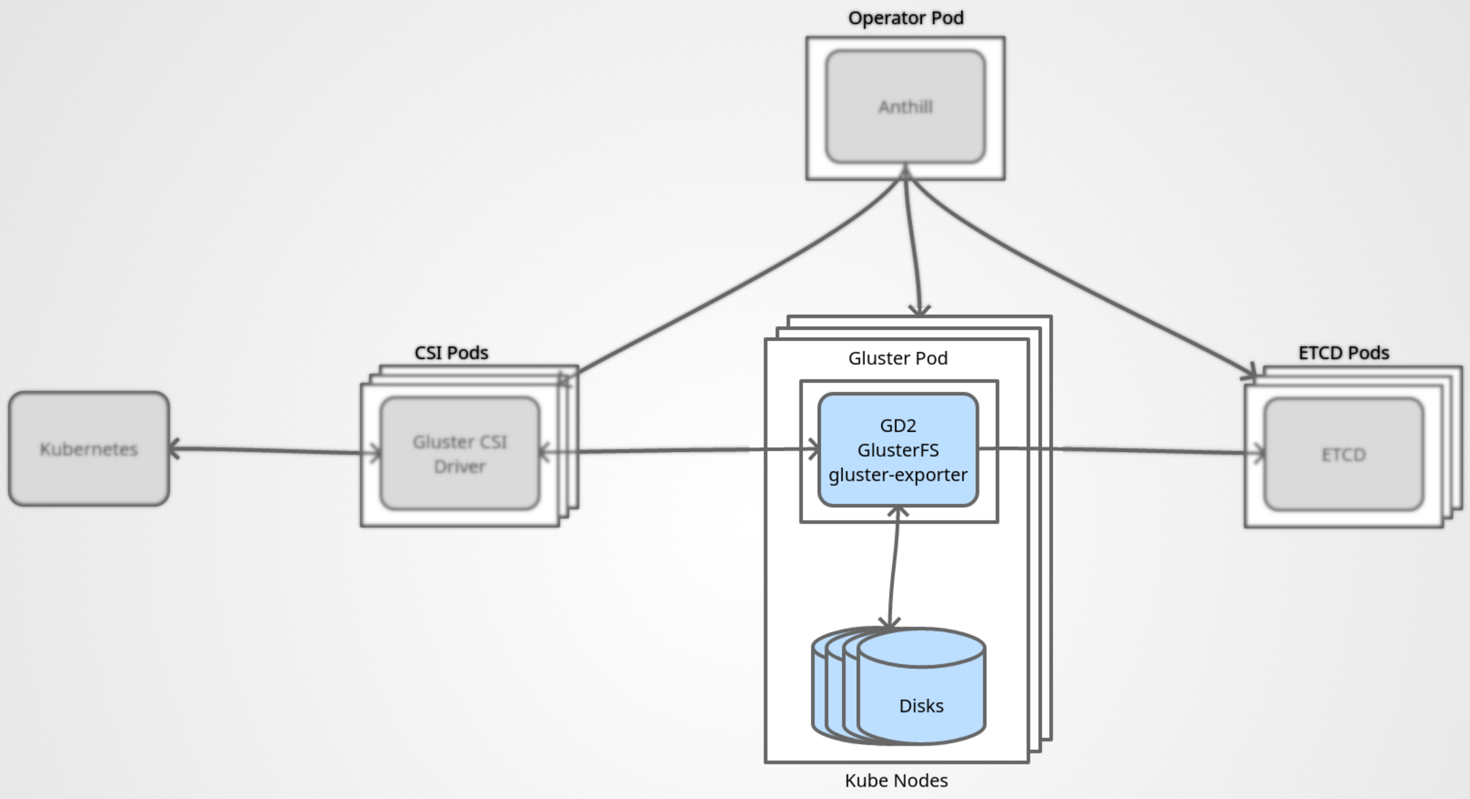
The nodeplugin pod also contains GlutsterFS client bits which is needed to perform the mounts.

GLUSTER CSI DRIVER STATUS

- Implements CSI v1.0.0 spec
  - Update to v1.0.1 underway
- Supports RWX PVs at the moment
- Support for RWO volumes under development
- github.com/gluster/gluster-csi-driver

Speaker notes

Right now the 1.0.0 spec of CSI has been implemented. 1.0.1 is underway, but I believe that the change has been already merged.
The driver supports only RWX PV types at the moment. RWO work is underway.


Operator Pod
Anthill
CSI Pods
Kubernetes
Gluster CSI Driver
Gluster Pod
GD2
GlusterFS
gluster-exporter
Disks
Kube Nodes
ETCD Pods
ETCD

GLUSTERD2

GLUSTERD2

- Automated Gluster volume management
- Converges multiple different tools
  - *heketi*
  - *gluster-block*

Speaker notes

GD2 is the newer management framework for GlusterFS. The provides automated and smart volume management operations.
GD2 can figure out the layout of a volume, provision the bricks and create the volume all by itself.
It converges functionality of multiple different tools like heketi and gluster-block into one.

GLUSTERD2

- ReST API
- Reslient orchestration
- Single source of information

Speaker notes

GD2 provides a ReST API that can be programmed against. This allows the CSI driver to exist.

Instead of cluster information being duplicated between Heketi and Gluster, GD2 is the single source of cluster information in GCS. GD2 connects to the etcd cluster that is deployed with GCS to store the cluster information.

GD2 also implements a reslient orchestration engine that ensures that operations happen reliably, and if operations fail proper cleanup happens.

GLUSTERD2

- Prometheus monitoring
- OpenCensus based tracing

Speaker notes

It also includes support for monitoring and tracing using industry standard APIs.

GLUSTERD2

- RWX Volumes
  - Automatic provisioning of bricks
  - Volume templates
- RWO Volumes
  - loopback based block exports
  - under development

Speaker notes

GD2 provides both RWX volumes and RWO volumes from one place.

For the RWX volume, GD2 does automatic provisioning as mentioned before. In addition, GD2 has support for volume templates. This allows us to create simplified and tuned volume stacks for container workloads.

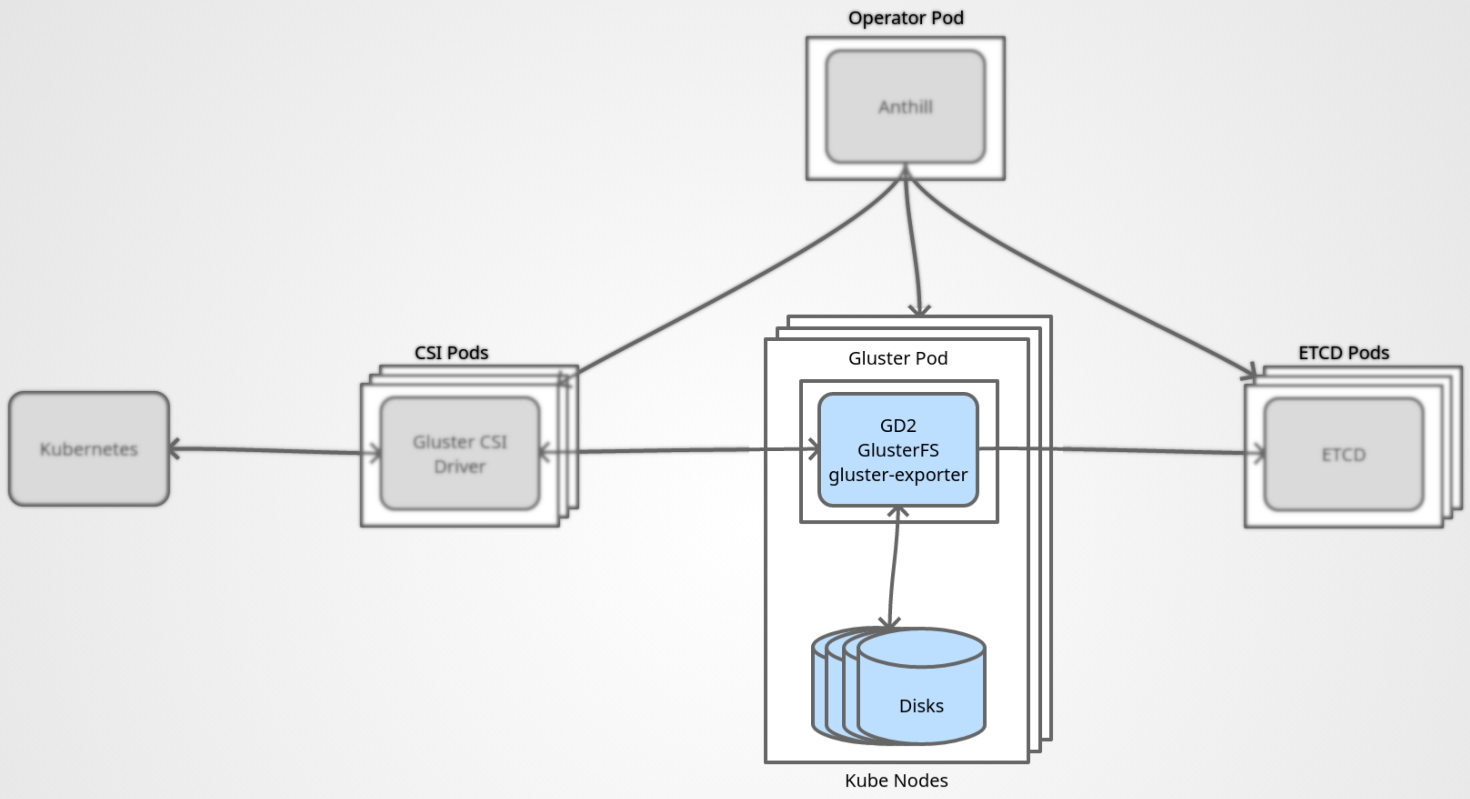
Work is in progress for implementing RWO volumes. Focus is on using loopback devices to provide block devices.

GLUSTERD2 STATUS

- Scales to ~1000 RWX volumes
  - ~100s of parallel operations
- Hitting limits of LVM scaling
  - optimization work in progress
- Hitting limits of etcd
  - tuned etcd config
  - optimization of etcd usage
- github.com/gluster/glusterd2

Speaker notes

Currently GD2 can scale to over a 1000 RWX volumes and several hundreds of parallel operations without problems.
But we hit limits of LVM and etcd, and we're working on optimizing how we use lvm operations and etcd.


Operator Pod
Anthill
CSI Pods
Gluster CSI Driver
Kubernetes
Gluster Pod
GD2
GlusterFS
gluster-exporter
Disks
Kube Nodes
ETCD Pods
ETCD

GLUSTERFS

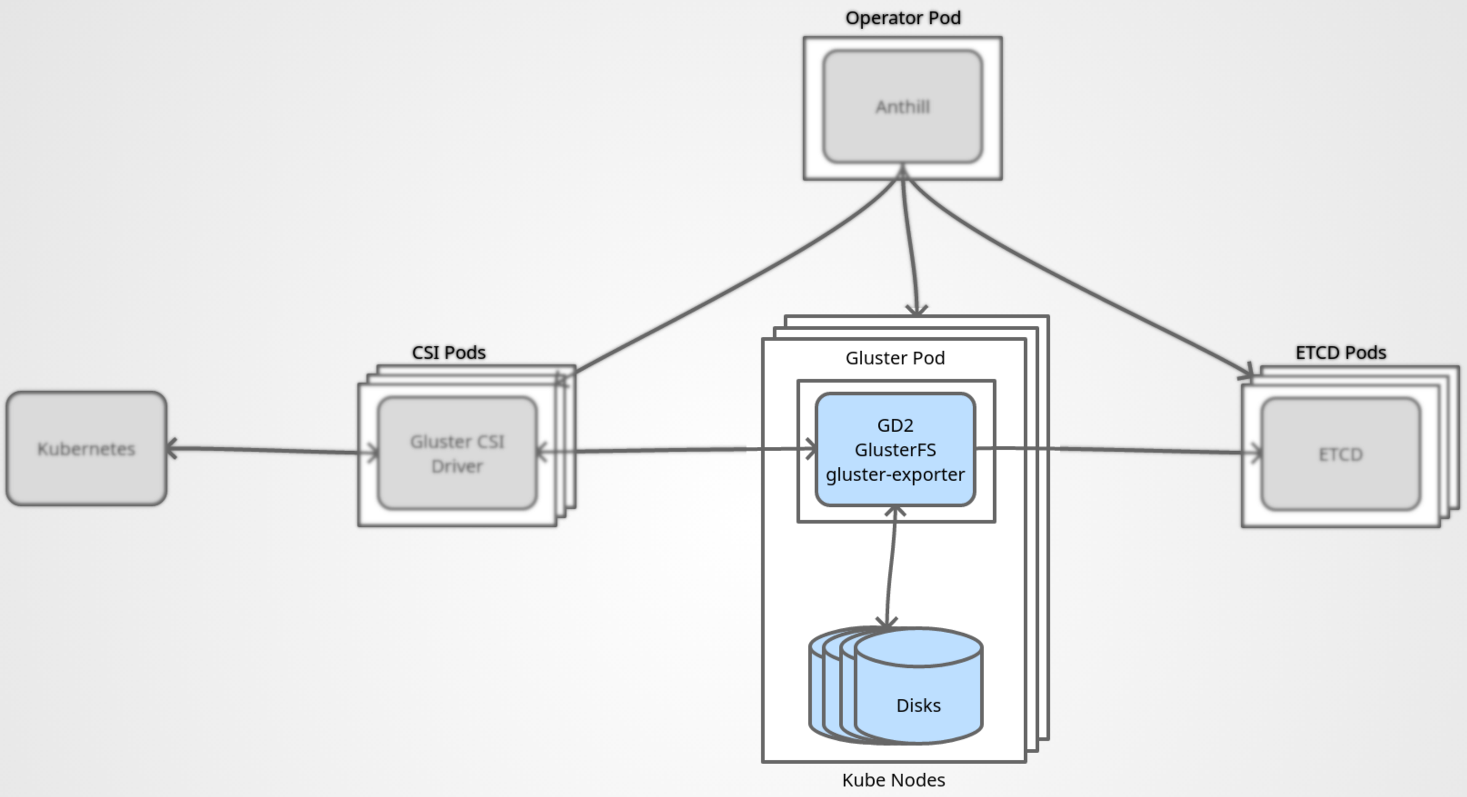
GLUSTERFS STATUS

- Optimize resource usage
  - memory
  - threads
- Improvements to multiplexing
  - self-heal multiplex
- Fencing for RWO volumes
- github.com/gluster/glusterfs

Speaker notes

The work in GlusterFS is mainly about improving stability, performance and optimizing resource usage.
This is a requirement for us to safely scale to 1000s of active volumes.

Brick-multiplexing is of special concern as it is crucial for the scale we want to acheive. Work is in progress to multiplex the shd. This is needed as the RWX volumes are replicate, and at 1000's of volumes we'd have 3x1000s of SHD processes if not multiplexed.


Operator Pod
Anthill
CSI Pods
Kubernetes
Gluster CSI Driver
Gluster Pod
GD2
GlusterFS
gluster-exporter
Disks
Kube Nodes
ETCD Pods
ETCD

MONITORING

MONITORING

- gluster-prometheus
- gluster-mixins

Speaker notes

Monitoring in GCS is handled in two different projects, Gluster-prometheus and Gluster-mixins.

To help users test this out, the current GCS delpoyment can also deploy a prometheus server. Though in real world workloads, the gluster-exporter would be configured to work with K8S cluster's prometheus server.

GLUSTER-PROMETHEUS

- gluster-exporter
  - Runs along with GD2
- Available metrics
  - Peer metrics
  - Volume metrics
  - Brick metrics
  - Process metrics
- github.com/gluster/gluster-prometheus

Speaker notes

Gluster-prometheus project provides a gluster-exporter binary, which exports gluster metrics to prometheus. The gluster-exporter runs along with GD2 and GlusterFS in each Gluster pod.

Different types of metrics are exported, including
- Peer metrics - Peer system utilization metrics
- Volume metrics - volume status, size, usage etc.
- Brick metrics - Brick utilization, status, size etcd.

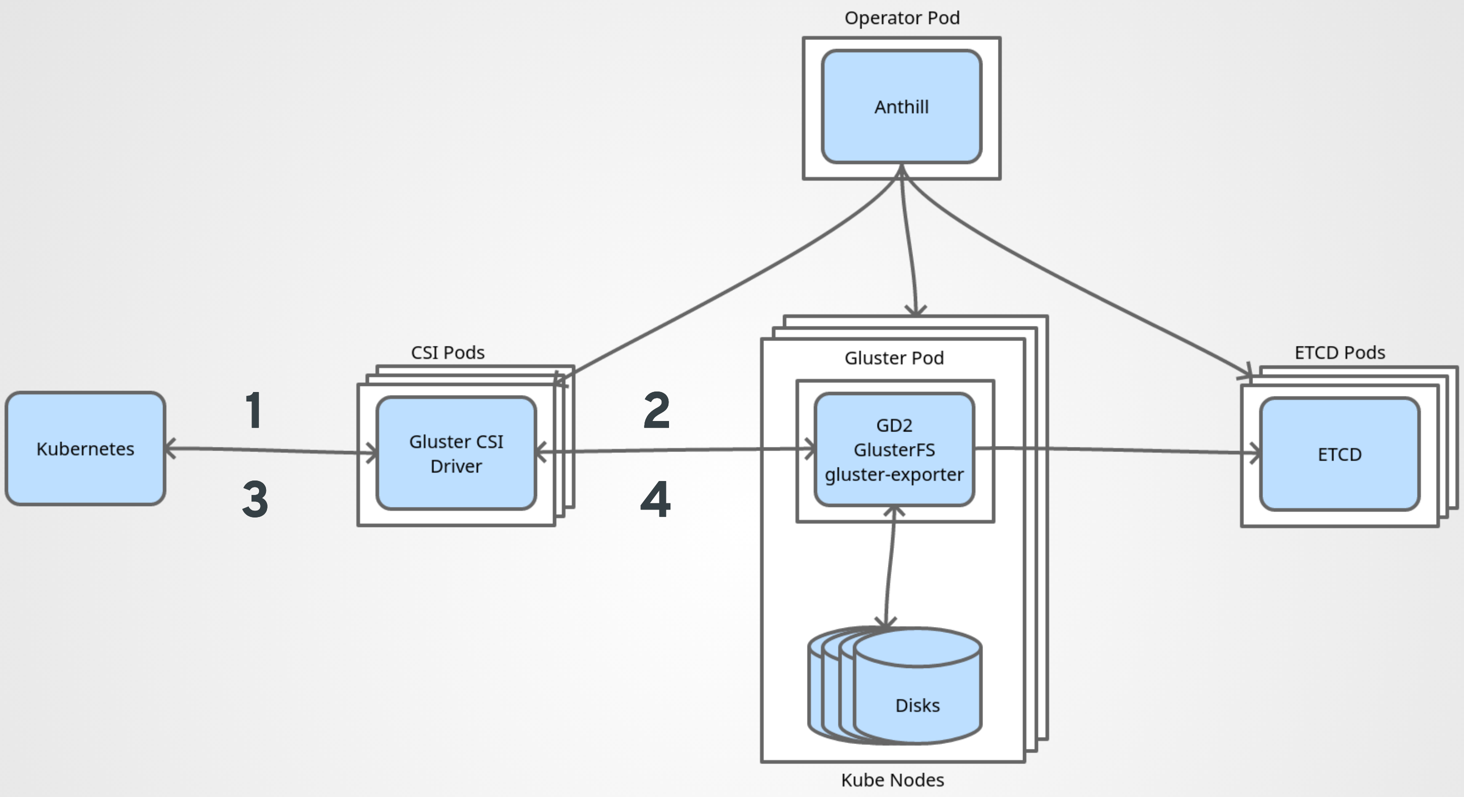
GLUSTER-MIXINS

- Prometheus configuration
  - rules
  - alerts
- Grafana dashboard configuration
- github.com/gluster/gluster-mixins

Speaker notes

Gluster-mixins provides configuration files for Prometheus and Grafana. These help better monitor the GCS deployment.
The prometheus configs include metric aggregation rules and alert rules.
For grafana a dashboard config is provided that helps visualize the GCS cluster.

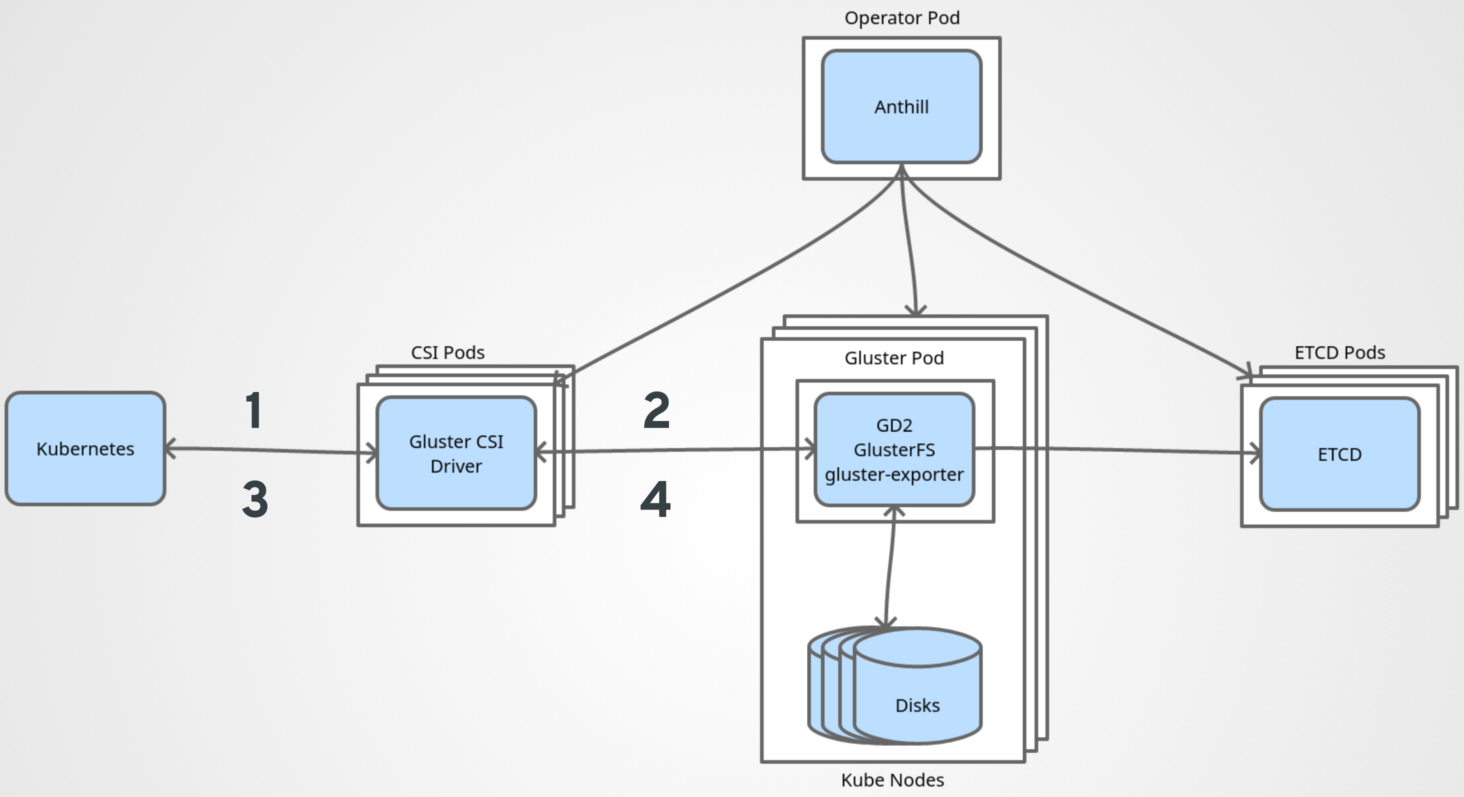
The deployment process for gluster-mixins is still a WIP. In short a temporary pod comes up, pushes the configuration to the prometheus server and goes back down.



RWX PERSISTENT VOLUME

Speaker notes

Now that we know the stack, let's see how it works together.
The GCS flow of operations is straightforward.
1. K8S raises a PVC request, which is recieved by the CSI provisioner.
2. The CSI provisioner reaches out the GD2 with the request to create the required volume. Once the volume is created it returns back to K8S with the PV.
3. When needed K8S sends a mount/bind request for the created PV, which is picked up by the CSI nodeplugin.
4. The CSI Nodeplugin then performs the gluster volume mount where requested.



RWO PERSISTENT VOLUME

Speaker notes

The flow is the same for both RWX and RWO volumes.

TRY OUT GCS

Speaker notes

Now let's see how you can try out GCS. Keep in mind that GCS is still under heavy development and you'll be trying out pre-stable versions of GCS. The currently released version of GCS is 0.5.

GCS/DEPLOY

- github.com/gluster/gcs
- Ansible based deployment
  - *deploy-gcs.yml*
  - Performs full deployment
  - Will shrink as Anthill matures
- More information in README

Speaker notes

At the moment we're based on Ansible. We have a playbook which deploys the full GCS stack. Anthill isn't feature complete as yet, but as Anthill matures, more responsibility will be handed over to Anthill. The Ansible deployer will reduce to just deploying Anthill later on.
Information on how to prepare the inventory files for deployment is provided in the README.

GCS/DEPLOY

- Vagrant based test clusters
  - *vagrant-libvirt only for now*
- Deploys K8S and GCS
  - Kubespray
  - *deploy-k8s.yml*
- More infomation in README

Speaker notes

A vagrantfile is available that deploys a 3-node K8S cluster with GCS on your system. It's vagrant-libvirt only for now.
It uses Kubespray, which provides playbooks to deploy K8S.

THANKS!

#GCS_gluster_for_containers

github.com/gluster/GCS

waffle.io/gluster/gcs

Speaker notes

Please tweet/share about GCS using this hashtag and include @gluster, for some gluster swag.
For more information visit the GCS repository and the GCS waffle project board.