Agenda

- Google & Kubernetes
- Kubernetes Volume Subsystem
- Container Storage Interface (CSI)
- Untapped Opportunities
- Q&A
Google & Kubernetes
“Google is living a few years in the future and sends the rest of us messages,”

-- Doug Cutting, Hadoop founder, 2013
Humble Beginnings
Humble Beginnings

Google File System

Figure 1 GFS Architecture
Cattle

Not Pets
Kubernetes
Storage Layer
What do these words mean and how do they fit together?

Persistent Volume Claims  Driver  Persistent Volumes
Remote  File  Flex  Block  CSI  Stateless
Storage Classes  Ephemeral  Local  Out-of-tree
Dynamic Provisioning  In-tree  Volume  Object
Stateful  Plugin
Kubernetes Principle

Portability

Workload
Kubernetes: Workload Portability

Kubernetes Goal

- Abstract away cluster details
- Decouple apps from infrastructure

To enable users to

- Write once, run anywhere (workload portability!)
- Avoid vendor lock-in
Kubernetes: Workload Portability

Kubernetes Cluster

Node 1
- Kernel/OS
- Hardware

Node 2
- Kernel/OS
- Hardware

Node 3
- Kernel/OS
- Hardware
Kubernetes: Workload Portability
Kubernetes: Workload Portability

EC2 Instance 1
Kernel/OS
Hardware

EC2 Instance 2
Kernel/OS
Hardware

EC2 Instance 3
Kernel/OS
Hardware

App 1
App 2
App 3
App 4

Kubernetes Cluster
Kubernetes: Workload Portability

- App 1
- App 2
- App 3
- App 4

Kubernetes Cluster

Bare Metal 1
- Kernel/OS
- Hardware

Bare Metal 2
- Kernel/OS
- Hardware

Bare Metal 3
- Kernel/OS
- Hardware
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
spec:
  replicas: 2
  template:
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v3
Kubernetes: Workload Portability

App 1

Node 1

Kernel

Frontend Pod Replica 1

Hardware

App 2

Node 2

Kernel

Frontend Pod Replica 2

Hardware

App 3

Node 3

Kernel/OS

Hardware

App 4
Problem with Containers and State

What about stateful apps?

Pod and ReplicaSet abstract compute and memory.

1. Containers are ephemeral: no way to persist state
   ○ Container termination/crashes result in loss of data
   ○ Can’t run stateful applications
2. Containers can’t share data between each other.
Challenges with Abstracting Storage

So many different types of storage

- **Object Stores**
  - AWS S3, GCE GCS, etc.
- **SQL Databases**
  - MySQL, SQL Server, Postgres, etc.
- **NoSQL Databases**
  - MongoDB, ElasticSearch, etc.
- **Pub Sub Systems**
  - Apache Kafka, Google Cloud Pub/Sub, AWS SNS, etc.
- **Time series databases**
  - InfluxDB, Graphite, etc.
- **File Storage**
  - NFS, SMB, etc.
- **Block Storage**
  - GCE PD, AWS EBS, iSCSI, Fibre Channel, etc.
- **File on Block Storage**
- **And more!**

What do we focus on?
What do we focus on?

In scope:

- **File Storage**
  - NFS, SMB, etc.

- **Block Storage**
  - GCE PD, AWS EBS, iSCSI, Fibre Channel, etc.

- **File on Block Storage**

Out of scope:

- **Object Stores**
  - AWS S3, GCE GCS, etc.

- **SQL Databases**
  - MySQL, SQL Server, Postgres, etc.

- **NoSQL Databases**
  - MongoDB, ElasticSearch, etc.

- **Pub Sub Systems**
  - Apache Kafka, Google Cloud Pub/Sub, AWS SNS, etc.

- **Time series databases**
  - InfluxDB, Graphite, etc.

- **etc.**
What do we focus on?

In scope:
- File Storage
  - NFS, SMB, etc.
- Block Storage
  - GCE PD, AWS EBS, iSCSI, Fibre Channel
- File on Block Storage

Data Path
Standardized (Posix, SCSI)

Out of scope:
- Object Stores
  - AWS S3, GCE GCS, etc.
- SQL Databases
  - MySQL, SQL Server, Postgres, etc.
- NoSQL Databases
  - MongoDB, ElasticSearch, etc.
- Pub Sub Systems
  - Apache Kafka, Google Cloud Pub/Sub, AWS SNS, etc.
- Time series databases
  - InfluxDB, Graphite, etc.
- etc.
Kubernetes Volume Plugins

A way to reference **block device** or **mounted filesystem** (possibly with some data in it)

Accessible by all containers in pod

Volume plugins specify

- How volume is setup in pod
- Medium that backs it

Lifetime of volume is same as the pod or longer
Kubernetes Volume Plugins

Kubernetes has many volume plugins

Remote Storage
- GCE Persistent Disk
- AWS Elastic Block Store
- Azure File Storage
- Azure Data Disk
- Dell EMC ScaleIO
- iSCSI
- Flocker
- NFS
- vSphere
- GlusterFS
- Ceph File and RBD
- Cinder
- Quobyte Volume
- FibreChannel
- VMware Photon PD

Ephemeral Storage
- EmptyDir
- Expose Kubernetes API
  - Secret
  - ConfigMap
  - DownwardAPI

Local
- Host path
- Local Persistent Volume (Beta)

Out-of-Tree
- Flex (exec a binary)
- CSI (Beta)
- Other
Ephemeral Storage

Temp scratch file space from host machine

Data exists only for lifecycle of pod.

Can only be referenced “in-line” in pod definition not via PV/PVC.

Volume Plugin: EmptyDir
Ephemeral Storage

Temp scratch file space from host machine

Data exists only for lifecycle of pod.

Can only be referenced “in-line” in pod definition not via PV/PVC.

Volume Plugin: EmptyDir

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
spec:
  containers:
  - image: k8s.gcr.io/container1
    name: container1
    volumeMounts:
      - mountPath: /shared
        name: shared-scratch-space
  - image: k8s.gcr.io/container2
    name: container2
    volumeMounts:
      - mountPath: /shared
        name: shared-scratch-space
  volumes:
  - name: shared-scratch-space
    emptyDir: {}
```
Ephemeral Storage

Built on top of EmptyDir:

- Secret Volume
- ConfigMap Volume
- DownwardAPI Volume

Populate Kubernetes API as files into an EmptyDir
Kubernetes Principle

Meet the user where they are
Ephemeral Storage

Built on top of EmptyDir:

- Secret Volume
- ConfigMap Volume
- DownwardAPI Volume

Populate Kubernetes API as files in to an EmptyDir
Remote Storage

Data persists beyond lifecycle of any pod

Referenced in pod either in-line or via PV/PVC

Examples:

- GCE Persistent Disk
- AWS Elastic Block Store
- Azure Data Disk
- iSCSI
- NFS
- GlusterFS
- Cinder
- Ceph File and RBD
- And more!
Remote Storage

Kubernetes will automatically:

- Attach volume to node
- Mount volume to pod

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sleepypod
spec:
  volumes:
  - name: data
gcePersistentDisk:
    pdName: panda-disk
    fsType: ext4
  containers:
  - name: sleepycontainer
    image: gcr.io/google_containers/busybox
    command:
      - sleep
      - "6000"
  volumeMounts:
  - name: data
    mountPath: /data
    readOnly: false
```
Remote Storage

Kubernetes will automatically:

- Attach volume to node
- Mount volume to pod

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sleepypod
spec:
  volumes:
    - name: data
      gcePersistentDisk:
        pdName: panda-disk
        fsType: ext4
  containers:
    - name: sleepycontainer
      image: gcr.io/google_containers/busybox
      command:
        - sleep
        - "6000"
      volumeMounts:
        - name: data
          mountPath: /data
          readOnly: false
```
Kubernetes Principle

- Workload
- Portability
Remote Storage

Pod yaml is no longer portable across clusters!!

```
apiVersion: v1
google_containers/busybox

command:
- sleep
- "6000"

volumeMounts:
- name: data
  mountPath: /data
  readOnly: false
```
Persistent Volumes & Persistent Volume Claims

PersistentVolume and PersistentVolumeClaim Abstraction

Decouples storage implementation from storage consumption
PersistentVolume

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: myPV1
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 100Gi
  persistentVolumeReclaimPolicy: Retain
  gcePersistentDisk:
    fsType: ext4
    pdName: panda-disk

apiVersion: v1
kind: PersistentVolume
metadata:
  name: myPV2
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 100Gi
  persistentVolumeReclaimPolicy: Retain
  gcePersistentDisk:
    fsType: ext4
    pdName: panda-disk2
```
PersistentVolumeClaim

apiVersion: v1
type: PersistentVolumeClaim

metadata:
  name: mypvc
  namespace: testns

spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
PV to PVC Binding

$ kubectl create -f pv.yaml
persistentvolume "pv1" created
persistentvolume "pv2" created

$ kubectl get pv
<table>
<thead>
<tr>
<th>NAME</th>
<th>CAPACITY</th>
<th>ACCESSMODES</th>
<th>STATUS</th>
<th>CLAIM</th>
<th>REASON</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>pv1</td>
<td>10Gi</td>
<td>RWO</td>
<td>Available</td>
<td></td>
<td></td>
<td>1m</td>
</tr>
<tr>
<td>pv2</td>
<td>100Gi</td>
<td>RWO</td>
<td>Available</td>
<td></td>
<td></td>
<td>1m</td>
</tr>
</tbody>
</table>

$ kubectl create -f pvc.yaml
persistentvolumeclaim "mypvc" created

$ kubectl get pv
<table>
<thead>
<tr>
<th>NAME</th>
<th>CAPACITY</th>
<th>ACCESSMODES</th>
<th>STATUS</th>
<th>CLAIM</th>
<th>REASON</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>pv1</td>
<td>10Gi</td>
<td>RWO</td>
<td>Available</td>
<td></td>
<td></td>
<td>3m</td>
</tr>
<tr>
<td>pv2</td>
<td>100Gi</td>
<td>RWO</td>
<td>Bound</td>
<td>testns/mypvc</td>
<td></td>
<td>3m</td>
</tr>
</tbody>
</table>
Remote Storage

Volume referenced via PVC

Pod YAML is portable across clusters again!!

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sleepypod
spec:
  volumes:
  - name: data
    gcePersistentDisk:
      pdName: panda-disk
      fsType: ext4
  containers:
  - name: sleepycontainer
    image: gcr.io/google_containers/busybox
    command:
      - sleep
      - "6000"
    volumeMounts:
    - name: data
      mountPath: /data
      readOnly: false
```

```yaml
volumes:
  - name: data
    persistentVolumeClaim:
      claimName: mypvc
```
Dynamic Provisioning

Cluster admin pre-provisioning PVs is painful and wasteful.

Dynamic provisioning creates new volumes on-demand (when requested by user).

Eliminates need for cluster administrators to pre-provision storage.
Dynamic Provisioning

Dynamic provisioning “enabled” by creating StorageClass.

StorageClass defines the parameters used during creation.

StorageClass parameters opaque to Kubernetes so storage providers can expose any number of custom parameters for the cluster admin to use.

```yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: slow
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-standard
--
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: fast
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-ssd
```
Dynamic Provisioning

Users consume storage the same way: PVC

“Selecting” a storage class in PVC triggers dynamic provisioning

```yaml
apiVersion: v1
group: storage.k8s.io
kind: PersistentVolumeClaim
metadata:
  name: mypvc
  namespace: testns
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
storageClass: fast
```
Dynamic Provisioning

$ kubectl create -f storage_class.yaml
storageclass "fast" created

$ kubectl create -f pvc.yaml
persistentvolumeclaim "mypvc" created

$ kubectl get pvc --all-namespaces

<table>
<thead>
<tr>
<th>NAMESPACE</th>
<th>NAME</th>
<th>STATUS</th>
<th>VOLUME</th>
<th>CAPACITY</th>
<th>ACCESSMODES</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>testns</td>
<td>mypvc</td>
<td>Bound</td>
<td>pvc-331d7407-fe18-11e6-b7cd-42010a8000cd</td>
<td>100Gi</td>
<td>RWO</td>
<td>6s</td>
</tr>
</tbody>
</table>

$ kubectl get pv pvc-331d7407-fe18-11e6-b7cd-42010a8000cd

<table>
<thead>
<tr>
<th>NAME</th>
<th>CAPACITY</th>
<th>ACCESSMODES</th>
<th>RECLAIMPOLICY</th>
<th>STATUS</th>
<th>CLAIM</th>
<th>REASON</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>pvc-331d7407-fe18-11e6-b7cd-42010a8000cd</td>
<td>100Gi</td>
<td>RWO</td>
<td>Delete</td>
<td>Bound</td>
<td>testns/mypvc</td>
<td>13m</td>
<td></td>
</tr>
</tbody>
</table>
Dynamic Provisioning

Volume referenced via PVC

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sleepypod
spec:
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: mypvc
  containers:
  - name: sleepycontainer
    image: gcr.io/google_containers/busybox
    command:
      - sleep
      - "6000"
    volumeMounts:
    - name: data
      mountPath: /data
      readOnly: false
```
Hostpath Volumes

Expose a directory on the host machine to pod

What happens if your pod is moved to a different node?

Don't use hostpath (unless you know what you are doing)!!
Local Persistent Volumes

Expose a local block or file as a PersistentVolume

Reduced durability

Useful for building distributed storage systems

Useful for high performance caching

Kubernetes takes care of data gravity

Referenced via PV/”PVC so workload portability is maintained
In-Tre e Volume Plugins

Kubern etes “In-tree” Volume Plugins are awesome =)

Powerful abstraction for file and block storage

Automate provisioning, attaching, mounting, and more!

Storage portability via PV/PVC/StorageClass objects
In-Tree Volume Plugins

Kubernetes “In-tree” Volume Plugins are painful =( 

- Painful for Kubernetes Developers
  - Testing and maintaining external code
  - Bugs in volume plugins affect critical Kubernetes components
  - Volume plugins get full privileges of kubernetes components (kubelet and kube-controller-manager)

- Painful for Storage Vendors
  - Dependent on Kubernetes releases
  - Source code forced to be open source
Out-of-Tree Volume Plugins

Container Storage Interface (CSI) - Beta in v1.10; Targeting GA in v1.13
- Follows in the steps of CRI and CNI
- Collaboration with other cluster orchestration systems
- CSI makes Kubernetes volume layer truly extensible
- Plugins may be containerized

Flex Volumes
- Legacy attempt at out-of-tree
- Exec based
- Deployment difficult
- Doesn't support clusters with no master access
Untapped Opportunities
Application Portability

**Legacy Software**
- Ecommerce site
- Catalog, ERP

**Local Execution**
- Warehouse
- Branch
- Factory

**Jurisdictional / PII**
- Europe
- US
- Secure records

**Augmented Services**
- On-Prem
- Cloud
- Cloud Storage
- Cloud ML
- Big Query

**Edge / IoT**

**Cloud bursting**
Snapshot Portability
Unified Observability
Uniform Management
“The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.”

- Mark Weiser, The Computer for the 21st Century
Questions?

Get Involved!

- Container Storage Interface Community
  - [github.com/container-storage-interface/community](https://github.com/container-storage-interface/community)
  - Meeting every week, Wednesdays at 9 AM (PT)
  - [container-storage-interface-community@googlegroups.com](mailto:container-storage-interface-community@googlegroups.com)

- Kubernetes Storage Special-Interest-Group (SIG)
  - [github.com/kubernetes/community/tree/master/sig-storage](https://github.com/kubernetes/community/tree/master/sig-storage)
  - Meeting every 2 weeks, Thursdays at 9 AM (PST)
  - [kubernetes-sig-storage@googlegroups.com](mailto:kubernetes-sig-storage@googlegroups.com)