Introduction to Kubernetes

Storage Primitives for Stateful Workloads

Saad Ali  Chris Duchesne
Google  {code}
@the_saad_ali  @ChrisDuchesne
Agenda

• Presentation
  – Quick intro to Kubernetes (and Pods and ReplicaSets)
  – What are Kubernetes Volumes
  – Introduce Kubernetes Primitives
• Demo
• What’s next for Kubernetes Storage?
• Q&A
• You try the demo!
What is Kubernetes

Greek for “Helmsman”; also the root of the word “Governor”

- Container orchestrator
- Supports multiple container runtimes (including Docker)
- Supports multiple cloud and bare-metal environments
- Inspired and informed by Google’s experiences
- **Open source**, written in **Go**

Manage applications, not machines
Separation of Concerns

- Application (Dev)
- Cluster
- Kernel/OS (System)
- Hardware
Workload Portability

Kubernetes Goals
• API and implementation 100% open
• Modular and replaceable
• Don’t force apps to know about concepts that are
  – Cloud Provider Specific
  – Kubernetes Specific

Enable Users To
• Write once, run anywhere
• Avoid vendor lock-in
• Avoid coupling app to infrastructure
Pods

Small group of containers & volumes

Tightly coupled

The atom of scheduling & placement

Shared namespace
  • share IP address & localhost
  • share IPC, etc.

Managed lifecycle
  • bound to a node, restart in place
  • can die, cannot be reborn with same ID

Example: data puller & web server

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
  - name: filepuller
    image: saadali/filepuller:v2
  - name: webserver
    image: saadali/webserver:v3
```

pod.yaml
ReplicaSets

A simple control loop

Runs out-of-process wrt API server

One job: ensure N copies of a pod

- grouped by a selector
- too few? start some
- too many? kill some

Layered on top of the public Pod API

Replicated pods are fungible

- No implied order or identity

```
apiVersion: extensions/v1beta1
kind: ReplicaSet
metadata:
  name: frontend
spec:
  replicas: 4
  template:
    spec:
      containers:
      - name: php-redis
        image: saadali/gb-frontend:v3
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        env:
        - name: GET_HOSTS_FROM
          value: dns
        ports:
        - containerPort: 80
```
Problem

• Files in containers are ephemeral
  – Container termination/crashes result in loss of data
  – Can’t run stateful applications
• Can’t share files between containers
Kubernetes Volumes

- Directory, possibly with some data in it
- Accessible by all containers in pod
- Volume Plugins Define
  - How directory is setup
  - Medium that backs it
  - Contents of the directory
- Lifetime same as the pod or longer
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**
- Empty dir (and tmpfs)
- Expose Kubernetes API
  - Secret
  - ConfigMap
  - DownwardAPI

**Local Storage (Alpha)**

**Out-of-Tree**
- Flex (exec a binary)
- CSI (Future)

**Other**
- Host path
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
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
apiVersion: v1
class: 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
```
apiVersion: v1
type: 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
```
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

pod.yaml
```
Persistent Volumes & Claims
PV/PVC Example

apiVersion: v1
kind: PersistentVolume
metadata:
  name: myPV1
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 10Gi
  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
PV/PVC Example

$ 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>
apiVersion: v1
kind: PersistentVolumeClaim
text:
metadata:
  name: mypvc
  namespace: testns
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
$ 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>
PV/PVC Example

Volume referenced via PVC

```yaml
apiVersion: v1
class: 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
```
Dynamic Provisioning & Storage Classes
Dynamic Provisioning and Storage Classes

- Allows storage to be created on-demand (when requested by user).
- Eliminates need for cluster administrators to pre-provision storage.
Dynamic Provisioning and Storage Classes

- Cluster/Storage admins “enable” dynamic provisioning 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 and Storage Classes

- Users consume storage the same way: PVC
- “Selecting” a storage class in PVC triggers dynamic provisioning

```yaml
apiVersion: v1
class: PersistentVolumeClaim
metadata:
  name: mypvc
  namespace: testns
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
    storageClassName: fast

pv2.yaml
```
Dynamic Provisioning and Storage Classes

- Users consume storage the same way: PVC
- “Selecting” a storage class in PVC triggers dynamic provisioning

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

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

$ kubectl get pvc --all-namespaces
NAMESPACE   NAME   STATUS    VOLUME                                     CAPACITY   ACCESSMODES   AGE
  testns      mypvc Bound     pvc-331d7407-fe18-11e6-b7cd-42010a8000cd 100Gi      RWO           6s

$ kubectl get pv pvc-331d7407-fe18-11e6-b7cd-42010a8000cd
NAME                                       CAPACITY   ACCESSMODES RECLAIMPOLICY   STATUS    CLAIM          REASON    AGE
  pvc-331d7407-fe18-11e6-b7cd-42010a8000cd 100Gi      RWO  Delete BetterBound  testns/mypvc 13m
```
Dynamic Provisioning and Storage Classes

Volume referenced via PVC

```yaml
apiVersion: v1
type: Pod
data:
  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
```

`pod.yaml`
**Dynamic Provisioning and Storage Classes**

- **Default Storage Classes**
  - Allow dynamic provisioning even when a StorageClass is not specified in PVC
- **Pre-installed Default Storage Classes:**
  - Amazon AWS - EBS volume
  - Google Cloud (GCE/GKE) - GCE PD
  - Openstack - Cinder Volume
- **Introduced as alpha in Kubernetes 1.2**
- **GA as of 1.6**

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: slow
  annotations:
    storageclass.beta.kubernetes.io/is-default-class: "true"
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
```
Helm

- Package manager for Kubernetes
- Simplifies deployment and management of Kubernetes applications.
Demo
What’s Next for Kubernetes Storage?

Kubernetes Storage is investing in

- **Container Storage Interface (CSI)**
  - Standardized Out-of-Tree File and Block Volume Plugins
- **Local Storage**
  - Making node local storage available as persistent volume
- **Capacity Isolation**
  - Setting up limits so that a single pod can’t consume all available node storage via overlay FS, logs, etc.
Questions?

Get Involved!

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

- Container Storage Interface Community
  - https://github.com/container-storage-interface/community
  - Meeting every 2 weeks, Wednesdays at 8 AM (PT)

- CNCF Storage Workgroup
  - https://github.com/cncf/wg-storage
  - Meeting 3rd Wednesday of every month at 8 AM (PT)

- Other Kubernetes SIGs
  - https://github.com/kubernetes/community/blob/master/sig-list.md

- Meetups
  - https://www.meetup.com/topics/kubernetes