Basic of Stateful and Storage in Kubernetes

Kevin Setiawan Tanzil
Software Engineer @Alterra
1 August 2019

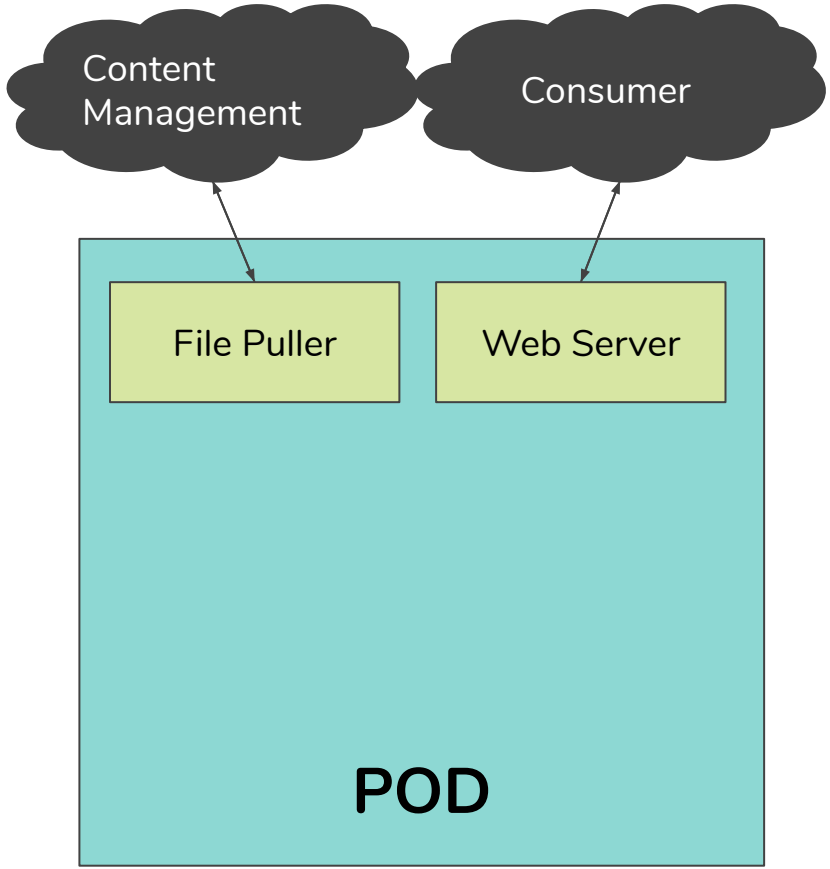
Agenda

How do these words mean and how they work together

Ephemeral	**Remote File**	**Out-of-Tree**
Stateful	**Block**	**Dynamic Provisioning**
Persistent Volume Claims	**CSI**	**In Tree**
Persistent Volume	**Stateless**	**Volume**
Storage Classes	**Local**	**Object**
Driver		**Plugin**

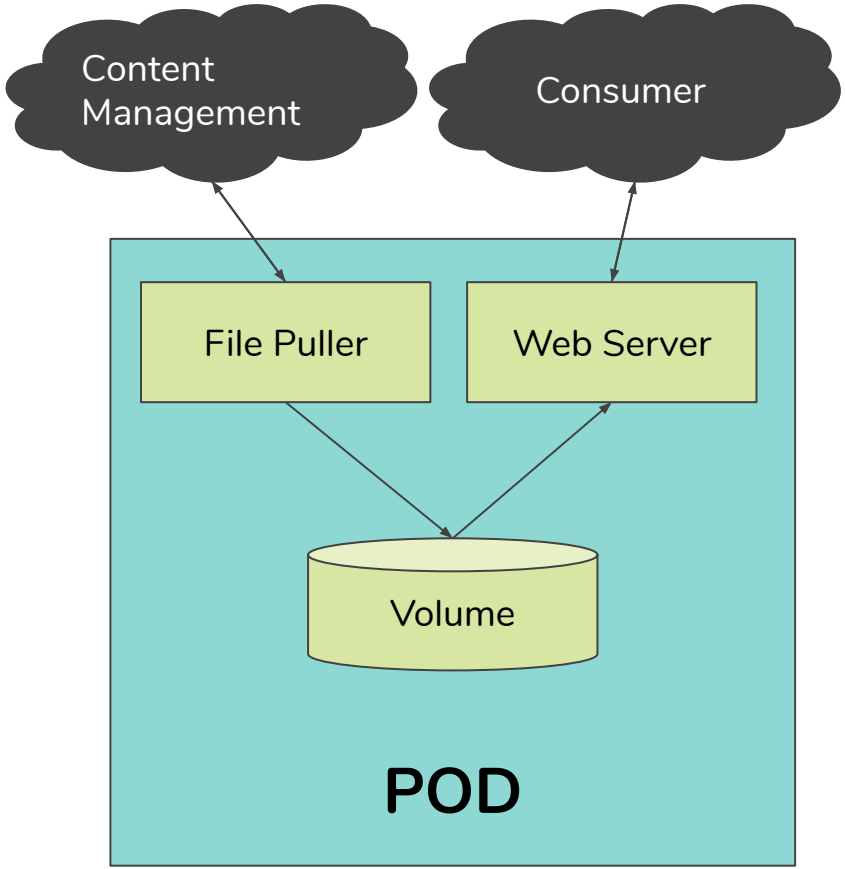
Container ?
What is the problem ?

1. **Container are ephemeral**
2. **Container can’t share data from each other**



The Solution is **Kubernetes Volume**

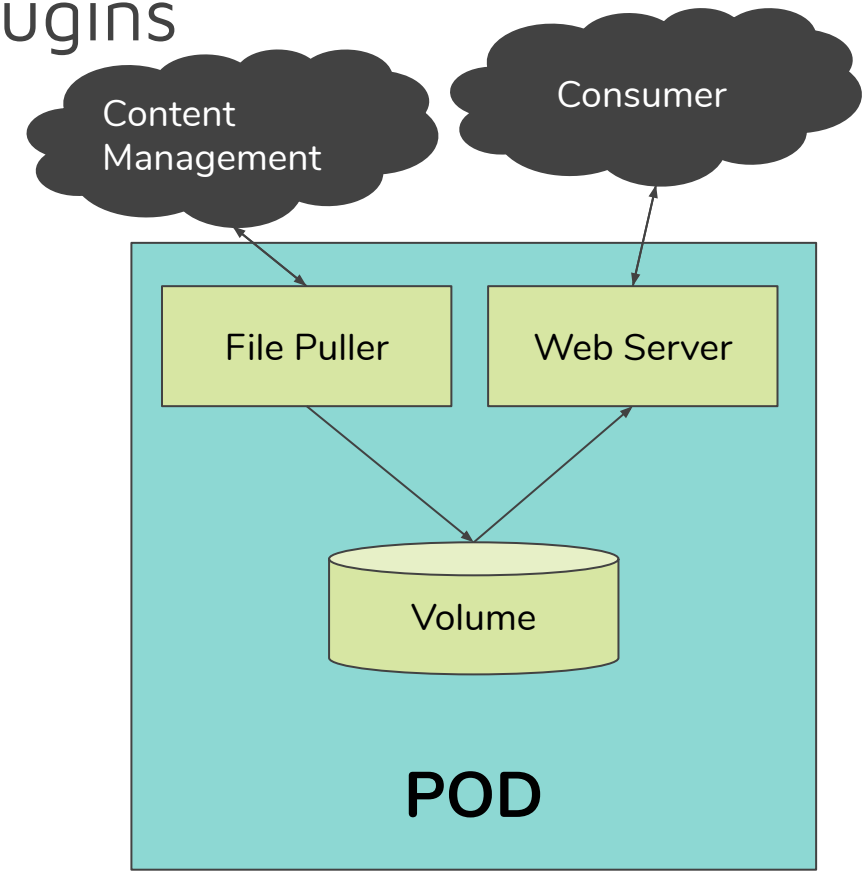
1. **Accessible by all container in pods**
2. **Lifetime of volume may same with pod or longer**



Kubernetes Volume Plugins

Kubernetes Volume Plugins
What is that ?

1. A way to reference **block device** or **mounted file system**
2. Volume plugins specify :
   a. How volume is set up inside a pod
   b. Medium that back it up

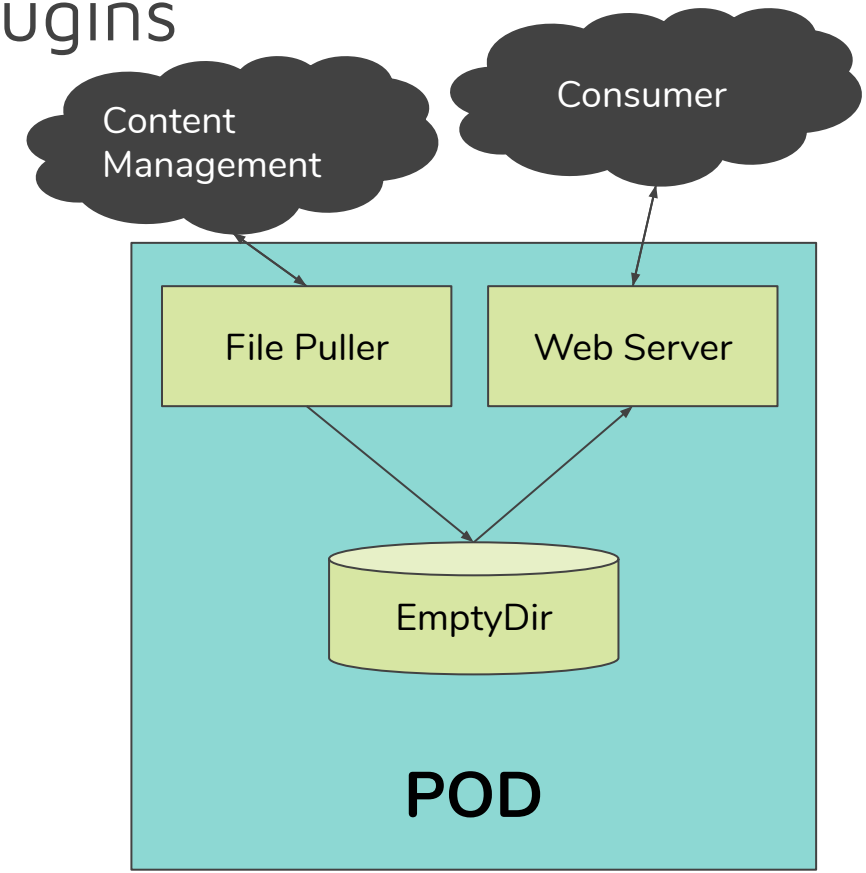


Kubernetes Volume Plugins
Currently Supported

1. **Remote Storage**
    a. GCE Persistent Disk
    b. AWS Elastic Block Store
    c. Azure File Storage
    d. NFS
    e. Ceph
    f. vSphere
    g. iSCSI
    h. Etc.
2. **Ephemeral Storage**
    a. EmptyDir
3. **Local Persistent Volume**
4. **Out-of-tree**
    a. CSI (Container Storage Interface)
    b. Flex
5. **Others**
    a. Host path

Kubernetes Volume Plugins
Ephemeral Storage

1. Basically is a temp scratch file system on host machine
2. Life is same with pod lifetime, pod destroy volume also destroy
3. Can only be reference in-line on pod cant via PV/PVC
4. Volume Plugins : EmptyDir



Kubernetes Volume Plugins
Ephemeral Storage

1. Basically is a temp scratch file system on host machine
2. Life is same with pod lifetime, pod destroy volume also destroy
3. Can only be reference in-line on pod cant via PV/PVC
4. Volume Plugins : EmptyDir

```
apiVersion: v1
kind: pod
metadata:
  name: test-pod
spec:
  containers:
  - image: example/container1
    name: container1
    volumeMounts:
    - mountPath: /shared
      name: shared-emptydir-space
  - image: example/container2
    name: container2
    volumeMounts:
    - mountPath: /shared
      name: shared-emptydir-space
  volumes:
  - name: shared-emptydir-space
    emptyDir: {}
```

Kubernetes Volume Plugins
Remote Storage

1. Lifetime data persists beyond of any pod lifecycle
2. Reference in-line or via PV/PVC
3. Example of Remote Storage :
    a. GCE Persistent Disk
    b. AWS Elastic Block Store
    c. Azure Data Disk
    d. Ceph
    e. And More....

Kubernetes Volume Plugins
Remote St

1. Kubernetes Will A
   a. Attach Volu
   b. Mount Volu


```
v1


pypod


ta-remote
stentDisk:
: panda-disk
: ext4
:
rc.io/google_container/busybox
eepycontainer


p
0"
unts:
- mountPath: /data
  name: data
  readOnly: false
```

Persistent Volumes & Persistent Volume Claims

Persistent Volume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mypv1
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 10Gi
  persistentVolumeReclaimPolicy: Retain
  gcePersistentDisk:
    fsType: ext4
    pdName: panda-disk-small
```

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mypv2
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 100Gi
  persistentVolumeReclaimPolicy: Retain
  gcePersistentDisk:
    fsType: ext4
    pdName: panda-disk-big
```

Persistent Volume Claims

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mypvc
  namespace: testns
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
```

PV to PVC Binding

```
kevinsetiawantanzil2@cloudshell:~/kubetalk-test (kubetalk)$ kubectl create -f small-pv.yaml
persistentvolume/mypv1 created
kevinsetiawantanzil2@cloudshell:~/kubetalk-test (kubetalk)$ kubectl create -f big-pv.yaml
persistentvolume/mypv2 created
kevinsetiawantanzil2@cloudshell:~/kubetalk-test (kubetalk)$ kubectl get pv
NAME    CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
mypv1   10Gi       RWO            Retain           Available           manual                  9s
mypv2   100Gi      RWO            Retain           Available           manual                  4s
kevinsetiawantanzil2@cloudshell:~/kubetalk-test (kubetalk)$ kubectl create -f mypvc.yaml
persistentvolumeclaim/mypvc created
kevinsetiawantanzil2@cloudshell:~/kubetalk-test (kubetalk)$ kubectl get pv
NAME    CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM           STORAGECLASS   REASON   AGE
mypv1   10Gi       RWO            Retain           Available                   manual                  26s
mypv2   100Gi      RWO            Retain           Bound       default/mypvc   manual                  21s
```

Kubernetes Volume Plugins
Remote Storage

1. Volume Reference via PVC
2. POD YAML is Portable across cluster

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

```
volume:
- name: data
  persistentVolumeClaim:
    claimName: mypvc
```

Dynamic Provisioning & Storage Classes

Dynamic Provisioning

1. Administrator admin pre-provisioning PVs is painful and wasteful
2. Create new volume on demand
3. Remove administrator for pre-provisioning PVs

Dynamic Provisioning
Storage Classes

1. Dynamic Provisioning Enabled by making StorageClass
2. StorageClass define parameter that needed
3. StorageClass parameter is opaque

```
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
Persistent Volume Claims

1. Select storageClass in PVC will trigger Dynamic Provisioning to create PV

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mypvc
  Namespace: testns
spec:
  accessModes:
  - ReadWriteOnce
  resource:
    request: 100Gi
storageClassName: fast
```

Dynamic Provisioning
Remote Storage

```
apiVersion: v1
kind: pod
metadata:
  name: sleepypod
volume:
- name: data
  persistentVolumeClaim:
    claimName: mypvc
  containers:
  - image: grc.io/google_container/busybox
    name: sleepycontainer
    command:
      - sleep
      - “6000”
    volumeMounts:
    - mountPath: /data
      name: data
      readOnly: false
```

Dynamic Provisioning
Default Storage Classes

1. Enable Dynamic Provisioning even StorageClass not spesific
2. Pre-install Storage Class

```
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
```

In-Tree Volume Plugins
What is this ?

1. A list of Supported Volume plugins by Kubernetes called In Tree Volume Plugins
2. In-Tree Volume plugins are awesome
    a. Can do dynamic provisioning
    b. Automate provisioning
    c. Automate mounting

Cons :

1. Kubernetes dev should maintain the volume plugins code
2. A little bug in volume plugin can make critical impact to kubernetes component
3. Storage Vendor should keep update with kubernetes release.
4. Vendor force to be open source

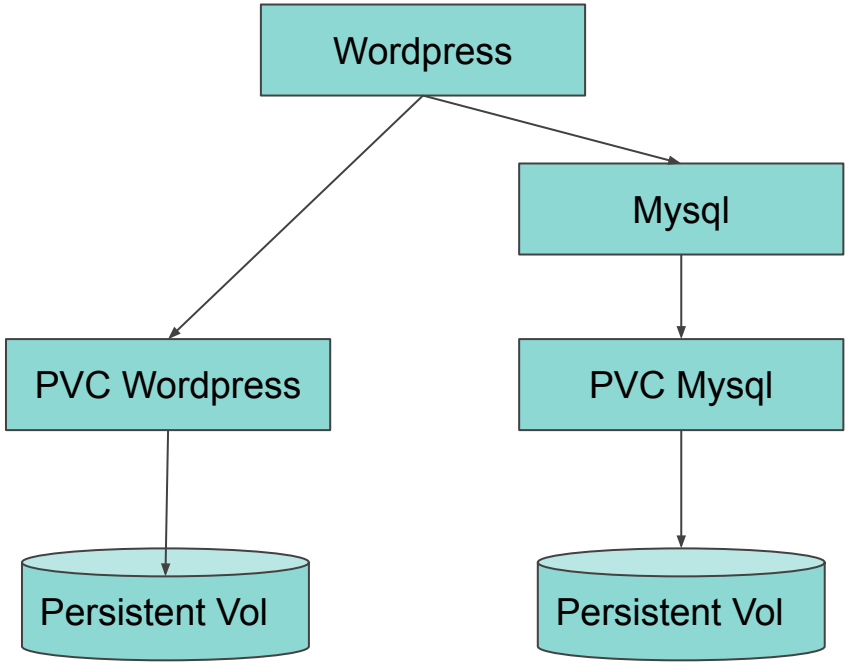
Out-of-Tree Volume Plugins

CSI (Container Storage Interface)

1. The idea is makes the Volume Plugins Truly extensible
2. Plugin may be containers
3. Storage Vendor can develope Driver base on CSI

DEMO

What we’re gonna to make
Apps Diagram



Thank you

Github : github.com/tanzilgr2288

Medium : medium.com/@kevinsetiawantanzil2