10: Storage

- Kubernetes support lots of types of storage
  - block, file, object, external, cloud datasystem, etc.
- ALL TYPES of storage are called **VOLUME** in kubernetes.
- Storage providers uses a plugin to allow the storage resources to be surfaced as volumes in Kubernetes
- Moden plugins are based on the *Container Storage Interface (CSI)* open standard to providing clean storage interface for container orchestrators.
- The Kubernetes **persistent volume sybsystem** is a set of API objects to make it easy for apps to consume storage.
  - Persistent Volumes PV: map to external storage assets
  - Persistent Volume Claims PVC: like tickets that authorize Pods to use PVs
  - Storage Classes SC: Make PV and PVC runs automatic
- Example
  1. K8s cluster is running on AWS, AWS adm has created 25GB EBS volume called `ebs-vol`
  2. Kub adm creates a PV called `k8s-vol` that maps to `ebs-vol` using ```ebs.csi.aws.com CSI plugin
  - PV is a simply way of representing external storage asset on the K8s cluster
  3. Pod uses a PVC con claim access to the PV an using it.

Storage Providers

- Azure File
- AWS Elastic Block Store (EBS) Each provider needs a CSI plugin to expose their storage assets to Kubernetes.

Container Storage Interface CSI

Open source project that defines a standards-based interface so the storage can be uniform across multiple cont orchestrators.The day-to-day interaction with CSI will be referencing appropiate CSI plugin in the YAML manifest file.

Dynamic Provisioning wigh Storage Classes

- Storage Classes allow to define differenttiers of storage
- are resources in the storage.k8s.io/v1 API group
- StorageClass type (sc shortname in kubectl)

```
# defines a class of storage called fast-local
# based on AWS SSDs (io1) in Ireland (eu-west-1a)
```

```
kind: StorageClasee
apiVersion: storage.k8s.io/v1
metadata:
    name: fast-local
provisioner: ebs.csi.aws.com
parameters:
    type: io1
    iopsPerGB: "10"
    encrypted: true
allowedTopologies:
-   matchLabelExpressions:
    -   key: topology.ebs.csi.aws.com/zone
        values:
        -   eu-west-1a
```

- SC objects are immutable
- metadata.name should be meanigful
- parameters block is for plugin-specific values
- each class can onlu relato to a single *type* of storage on a single backend

SC Workflow

1. Have a storage backend (cloud or on premises)
2. Have a K8s cluster connected to the backend storage
3. Install and config the CSI storage plugin
4. Create one or more SC on K8s
5. Deploy Pods with PVCs that reference those SCs

- Pay clease attention to how
    - PodSpec refs PVC (by name)
    - PVC refs SC (by name)

```
# PVC
kinds: StorageClass                # StorageClass SC ################
apiVersion: storage.k8s.io/v1      ##
metadata:                          ##
    name: fast                     ## Referenced by the PVC
provisioner: pd.csi.storage.gke.io
parameters:                        ##
    type: pd-ssd                   ##################################
---
apiVersion: v1
kind: PersistentVolumeClaim        # Persistent Volume Claim ########
metadata:                          ##
    name: mypvc                    ## Referenced by the PodSpec
spec:                              ##
    accessModes:                   ##
    -   ReadWriteOnce              ##
    resources:                     ##
```

```
        requests:                ##
            storage: 50Gi        ##
    storageClassName: fast       ## Matches name of the StorageClass
---                              ####################################
apiVersion: v1
kind: Pod                        # Pod ##############################
metadata:                        ##
    name: mypod                  ##
spec:                            ##
    volumes:                     ##
    -   name: data               ##
        persistentVolumeClaim:   ##
            claimName: mypvc     # Matches PVC name ################
    containers: ...
    <SNIP>
```

Access Mode

- ReadWriteOnce RWO: defines a PV that can only be bound as RW by a **SINGLE PVC**
- ReadWriteMany RWM: defines a PV that can only be bound as RW by multiple PVCs. Only supported by file and object storage
- ReadOnlyMany ROM: defines a PV that can only be bound as R/O by multiple PVCs

Reclaim

ReclaimPoliciy: Tells K8s how to deal with a PV when its PVC is released

- Delete
    - Most Dangerous
    - Default for PVs created dynamically via SC
    - Deletes the PV and associated storage resource when PVC is released
- Retain
    - will keep the PV object on the cluster and the data associated
    - Other PVC are prevented from using it in the future

Summarizing Storage

- Lets dynamically create backend storage resources auto-mapped to PVs on K8s.
- You define SCs in YAML files and references a plugin storage provider
- Once deployed, SC watches the API for new PVC objects referencing its name
- When a PVCs matching appear, SC creates the asset on the backend to maps to the PV

Hands-on

Using an Existing StorageClass

- Using a regional Google Kubernetes Engine cluster wit CSI plugin installed

```
kubectl get sc premium-rwo
kubectl describe sc premium
```

- K8s platforms in general creates three SCs
    - default: for PVCs that don't explicitly specify an SC
    - In prod environments you should always specify an SC that meets app requirements
- PROVISIONER column shows two of the SCs using the CSI plugin
- RECLAIM POLICY is set to Delete: The volumes will be deleted when PVC is deleted
- VOLUMEBINDINGMODE: *immediate* will create the volume on the external storage system as soon as the PVC is created. Immediate is dangerous if have multiple datacenters or cloud regions. Setting WaitForFirstConsumer will delay creation until a Pod using the PVC is created.

```
kubectl get pv
kubectl get pvc
```

Creating a new StorageClass

Defines a class called sc-fast-repl

- Fast SSD storage (type pd-ssd)
- Replicated (replication-type: regional-pd)
- Create on-demand (volumeBindingMode: WaitForFirstConsumer)
- Keep data when the PVC is deleted (reclaimPolicy: Retain)

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-fast-repl
provisioner: pd.csi.storage.gke.io
parametres:
  type: pd-ssd
  replicatoin-type: regional-pd
volumeBindingMode: WaitForFirstConsumer
reclaimPolicy: Retain
```

```
kubectl apply -f sc-gke-fast-repl.yml
kuibectl get sc
```

1. Created sc-fast-repl SCSC controler started watching the API sv for new PVC
2. ap deployed created the pvc2 PVC that requestses a 20GB volume from the sc-fast-repl SC
3. SC controller notices the PVC dynamically created the backend volume and PV

Even though Pod and PVC deleted, kubeclt get pv whill show PV still exists because the class it was created using Retain reclaim policy. Manually delete the PV with a kbuectl delete pv command.