CSCI 5409 Cloud Computing
Fall, 2023
Instructor: Dr. Lu Yang

Kubernetes Workload & Production (2)
Oct 16, 2023

Housekeeping and Feedback

- Start recording
- Midterm in class next Monday, Oct 23.
  - 10 multiple choice, 4 short answer, and 2 long answer
  - It covers the contents up to this lecture.

Part 2 Deployments, jobs, and scaling

- Deployments
- Self-learning lab: Creating Google Kubernetes Engine Deployments
- Jobs
- Self-learning lab: Deploying Jobs on Google Kubernetes Engine
- Cluster Scaling
- Controlling Pod Placement
- Getting Software into Your Cluster
- Self-learning lab: Configuring Pod Autoscaling and Node Pools
- Summary

Part 2 Deployments, jobs, and scaling

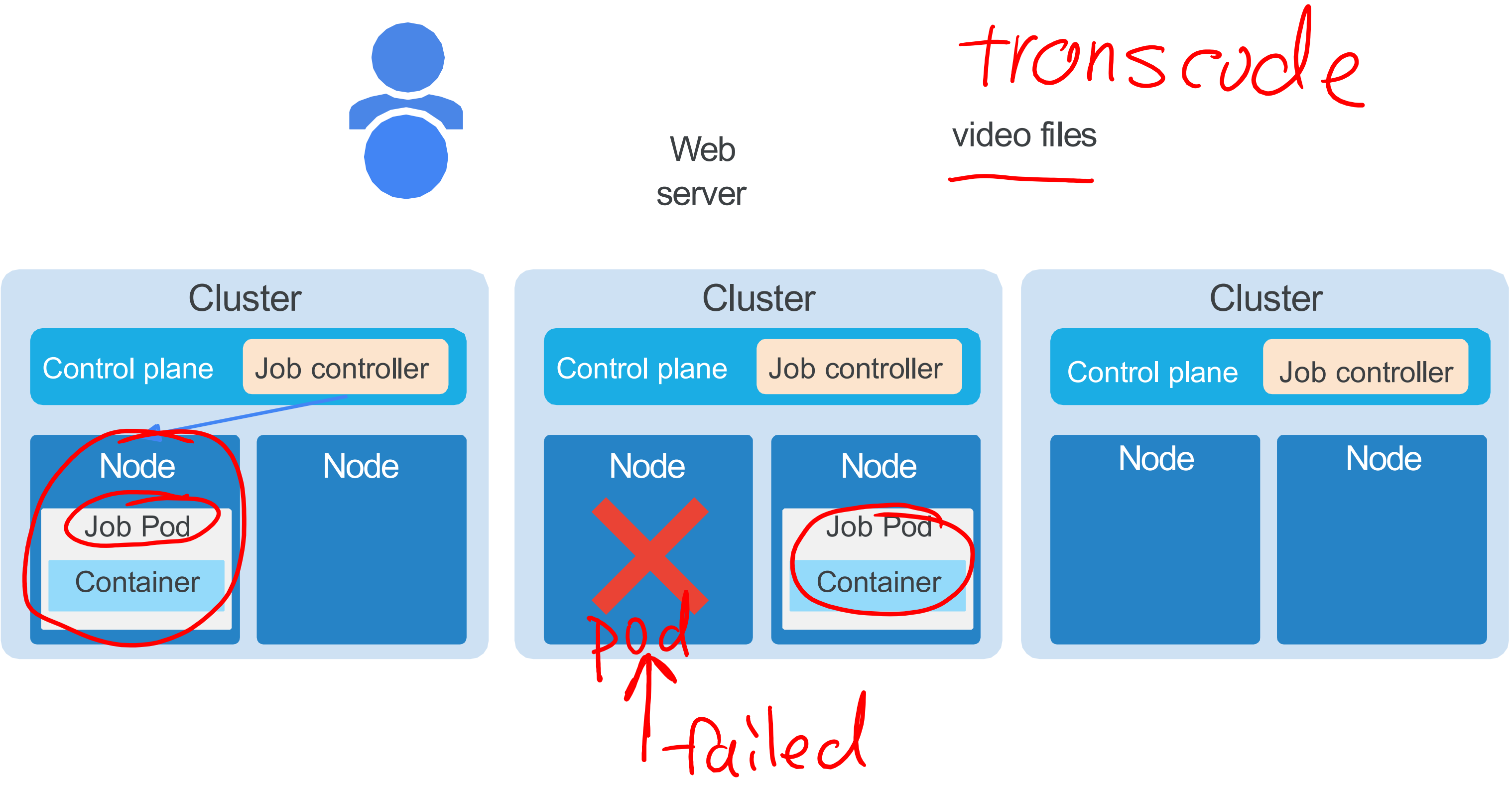
Lab

Creating Google Kubernetes Engine Deployments (https://www.youtube.com/watch?v=k4x4ce370LA)

Part 2 Deployments, jobs, and scaling

- Deployments
- Self-learning lab: Creating Google Kubernetes Engine Deployments
- Jobs
- Self-learning lab: Deploying Jobs on Google Kubernetes Engine
- Cluster Scaling
- Controlling Pod Placement
- Getting Software into Your Cluster
- Self-learning lab: Configuring Pod Autoscaling and Node Pools
- Summary

A scenario where Job provides the solution



Web server

transcode

video files

Cluster

Control plane

Job controller

Node

Job Pod

Container

Node

Cluster

Control plane

Job controller

Node

pod

failed

Node

Job Pod

Container

Cluster

Control plane

Job controller

Node

Node

Jobs definition

```
apiVersion: batch/v1
kind: Job
metadata:
  name: my-app-job
spec:
  completions: 3
  template:
     spec:
[…]
```

Part 2 Deployments, jobs, and scaling

Jobs

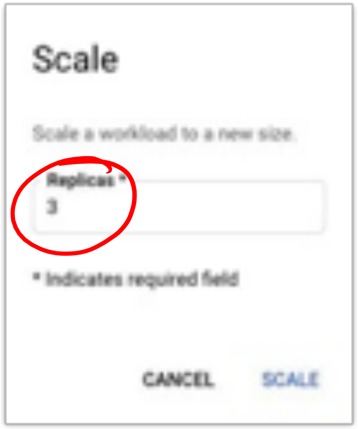
Inspecting a job

```
$ kubectl describe job [JOB_NAME]
```

```
$ kubectl get pod –l [job-name=my-app-job]
```

Scaling a job

```
$ kubectl scale job [JOB_NAME] --replicas [VALUE]
```



Scale

Scale a workload to a new size.

Replicas *: 3

* Indicates required field

Deleting a job

```
$ kubectl delete -f [JOB_FILE]
```

```
$ kubectl delete job [JOB_NAME]
```

Differences of jobs and deployments

The main difference between Deployments and Jobs is **how they handle a Pod that is terminated**. A Deployment is intended to be a "service", e.g. it should be up-and-running, so it will try to restart the Pods it manage, to match the desired number of replicas. While a Job is intended to execute and successfully terminate.

e.g. webserver, database server

e.g. database backup

In a Deployment, the default restartPolicy of your Pod is set Always. In a Job: Never. A job is not meant to restart your container once it would have completed. A deployment is not meant to complete.

https://stackoverflow.com/questions/68906801/what-is-difference-between-kubernetes-jobs-deployments

Part 2 Deployments, jobs, and scaling

- Deployments
- Self-learning lab: Creating Google Kubernetes Engine Deployments
- Jobs
- Self-learning lab: Deploying Jobs on Google Kubernetes Engine
- Cluster Scaling
- Controlling Pod Placement
- Getting Software into Your Cluster
- Self-learning lab: Configuring Pod Autoscaling and Node Pools
- Summary

Part 2 Deployments, jobs, and scaling

Lab

Deploying jobs on GKE (https://www.youtube.com/watch?v=pqGfzXHrYLk)

Part 2 Deployments, jobs, and scaling

- Deployments
- Self-learning lab: Creating Google Kubernetes Engine Deployments
- Jobs
- Self-learning lab: Deploying Jobs on Google Kubernetes Engine
- Cluster Scaling
- Controlling Pod Placement
- Getting Software into Your Cluster
- Self-learning lab: Configuring Pod Autoscaling and Node Pools

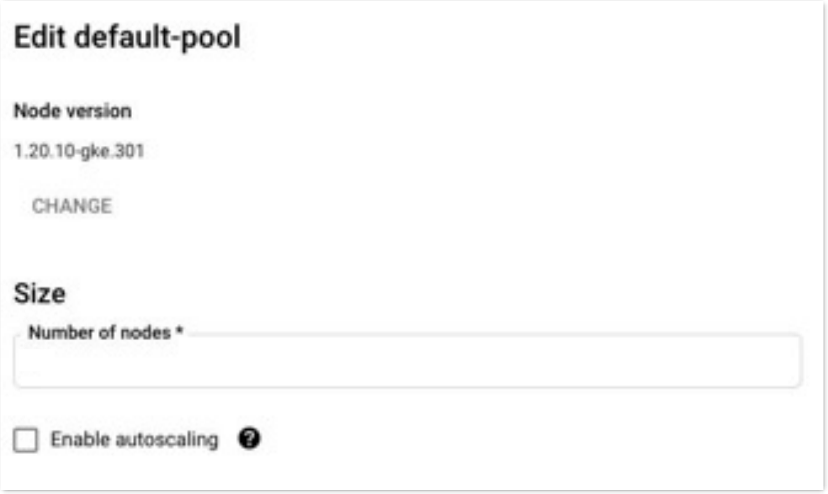
Part 2 Deployments, jobs, and scaling

Cluster scaling

Scaling down a cluster using the gcloud command

```
gcloud container clusters resize projectdemo --
node-pool \ default-pool \
--num_nodes 6
```

Scaling down a cluster from the cloud console



Edit default-pool

Node version

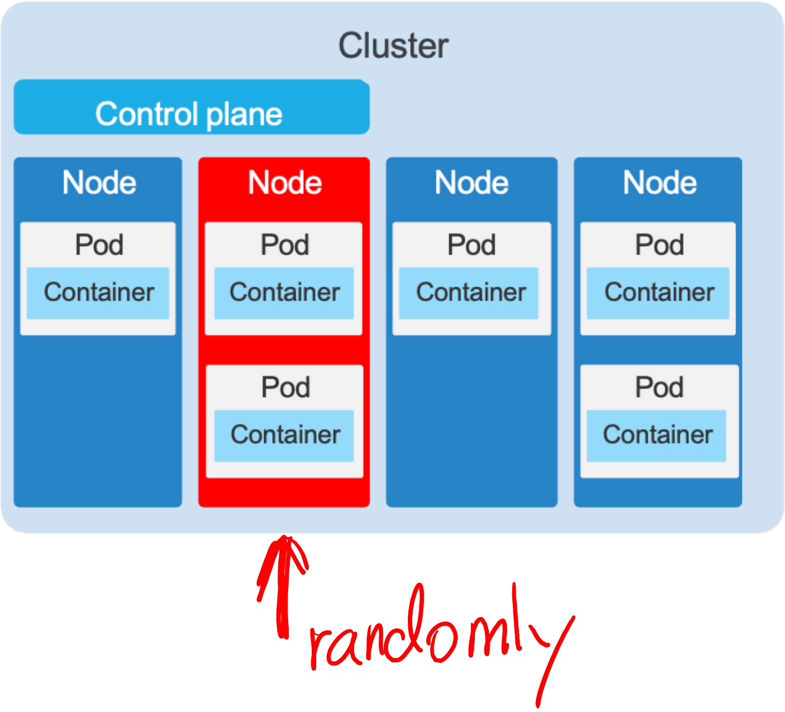
1.20.10-gke.301

Size

Number of nodes *

□ Enable autoscaling

Manual Cluster Scale down selects nodes randomly

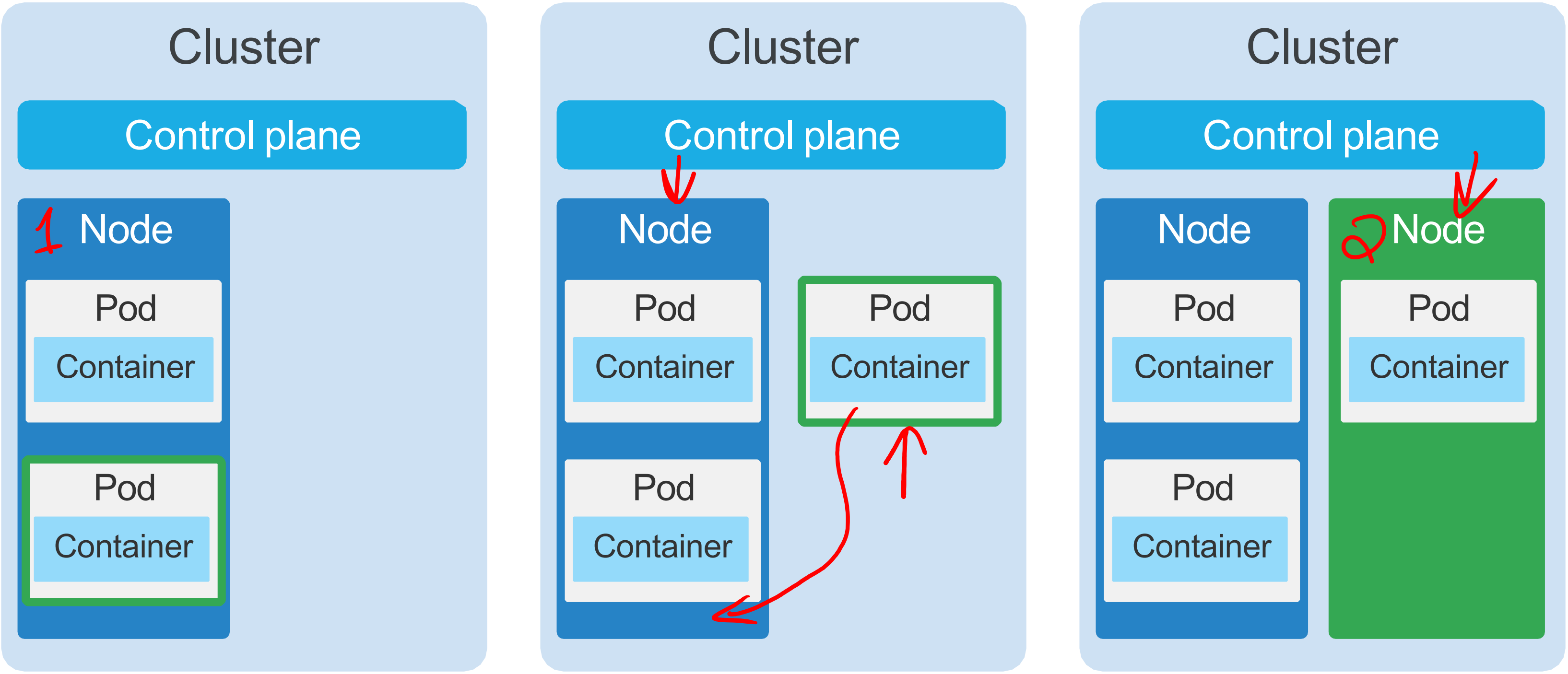


Part 2 Deployments, jobs, and scaling

Cluster scaling

Scale up a cluster with autoscaling

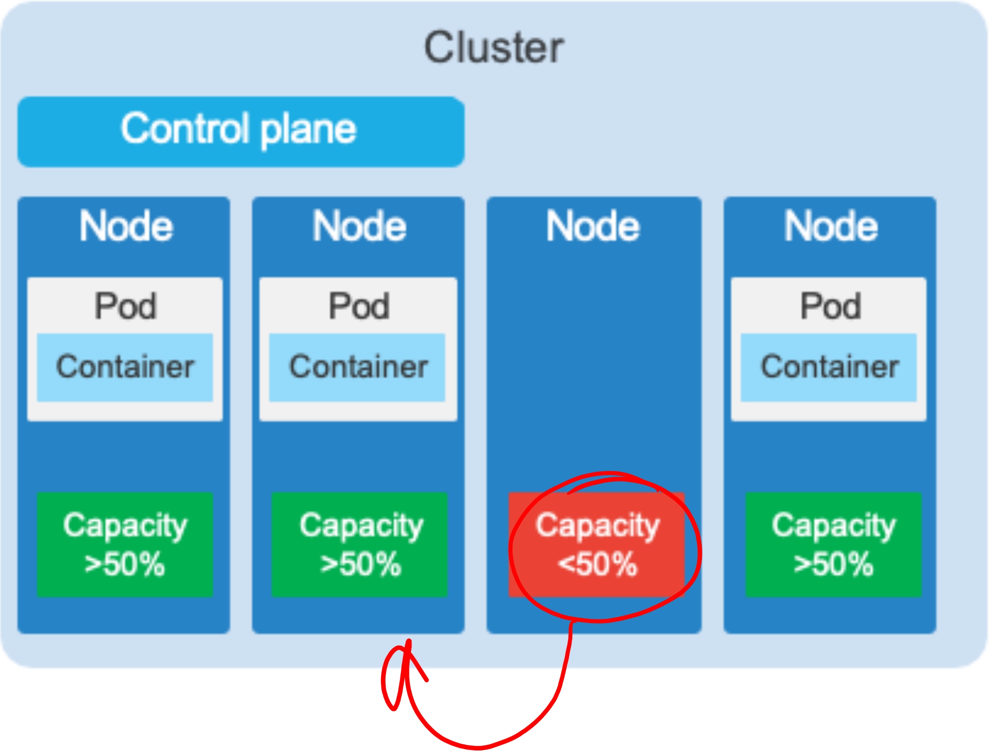
ECS⟷EKS



Part 2 Deployments, jobs, and scaling

Cluster scaling

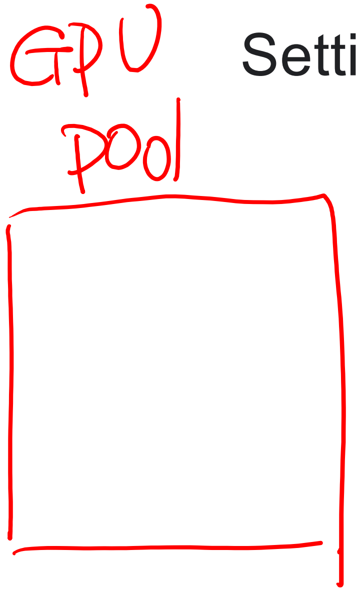
The Autoscaler removes nodes that remain below 50% utilization



Part 2 Deployments, jobs, and scaling

Cluster scaling

Setting a node pool size



Node pool = 0

Cluster size ≠ 0

MAX = 15,000 nodes x 110 Pods

Increase quota limits to avoid disruption

gcloud commands for autoscaling

Create a cluster with autoscaling enabled

```
gcloud container clusters create
[CLUSTER_NAME] --enable-autoscaling
--min-nodes 15 --max-nodes 50
[--zone COMPUTE_ZONE]
```

Enable autoscaling for an existing node pool

```
gcloud container clusters update
[CLUSTER_NAME] --enable-autoscaling \
--min-nodes 1 --max-nodes 10 --zone
[COMPUTE_ZONE] --node-pool [POOL_NAME]
```

Add a node pool with autoscaling enabled

```
gcloud container node-pools create
[POOL_NAME] --cluster [CLUSTER_NAME]
--enable-autoscaling --min-nodes 15
--max-nodes 50 [--zone COMPUTE_ZONE]
```

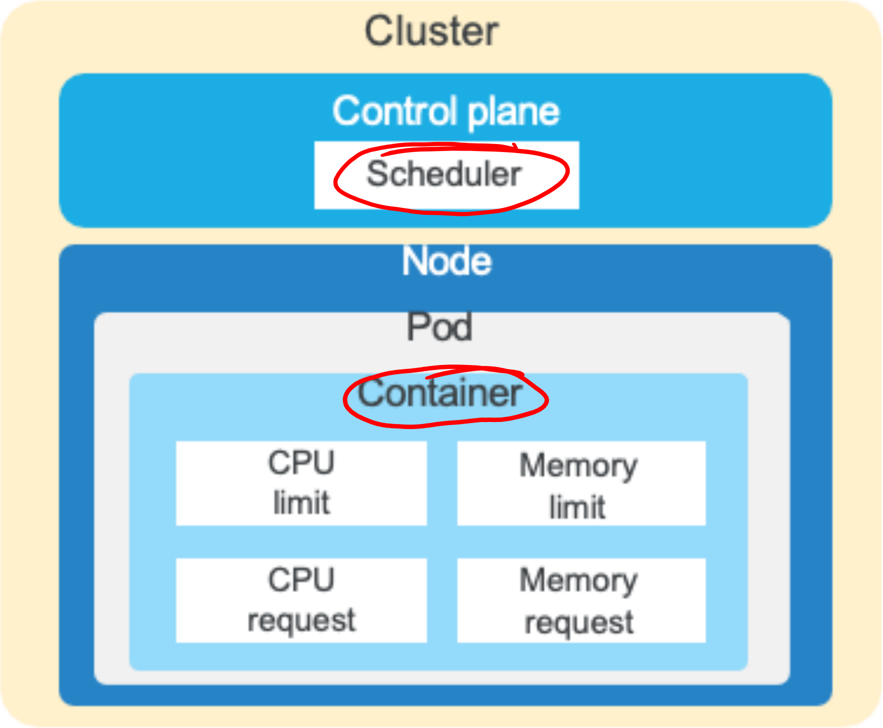
Disable autoscaling for an existing node pool

```
gcloud container clusters update
[CLUSTER_NAME] --no-enable-autoscaling \
--node-pool [POOL_NAME] [--zone
[COMPUTE_ZONE] --project [PROJECT_ID]]
```

Part 2 Deployments, jobs, and scaling

- Deployments
- Self-learning lab: Creating Google Kubernetes Engine Deployments
- Jobs
- Self-learning lab: Deploying Jobs on Google Kubernetes Engine
- Cluster Scaling
- Controlling Pod Placement
- Getting Software into Your Cluster
- Self-learning lab: Configuring Pod Autoscaling and Node Pools
- S

Controlled scheduling



Nodes must match all the labels present under the nodeSelector field

```
apiVersion: v1
kind: Pod
metadata:
  name: mysql
  labels:
    env: test

  containers:
  - name: mysql
    image: mysql
    imagePullPolicy: IfNotPresent
  nodeSelector:
    disktype: ssd
[…]
```

```
apiVersion: v1
kind: Node
metadata:
  name: node1
  labels:
    disktype: ssd
[…]
```

Nodes must match all the labels present under the nodeSelector field

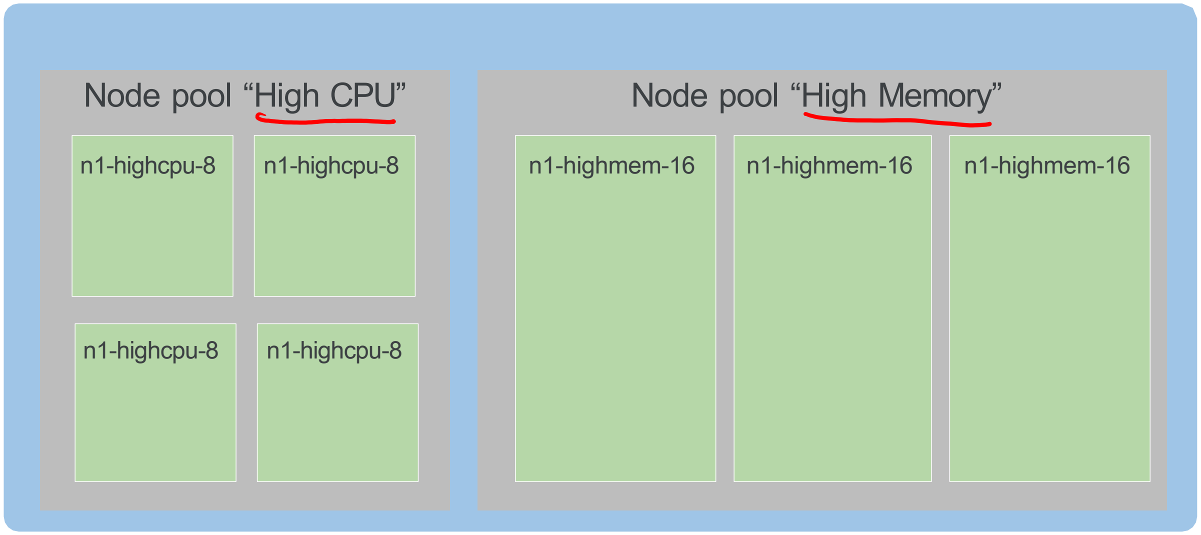
auto generated labels

```
apiVersion: v1
kind: Pod
metadata:
   name: mysql
   labels:
      env: test
spec:
   containers:
   - name: mysql
      image: mysql
      imagePullPolicy: IfNotPresent
   nodeSelector:
      cloud.google.com/gke-nodepool=ssd
[…]
```

Part 2 Deployments, jobs, and scaling

Controlling pod placement

Use node pools to manage different kinds of nodes



Node pool “High CPU”

n1-highcpu-8

n1-highcpu-8

n1-highcpu-8

n1-highcpu-8

Node pool “High Memory”

n1-highmem-16

n1-highmem-16

n1-highmem-16

Part 2 Deployments, jobs, and scaling

- Deployments
- Self-learning lab: Creating Google Kubernetes Engine Deployments
- Jobs
- Self-learning lab: Deploying Jobs on Google Kubernetes Engine
- Cluster Scaling
- Controlling Pod Placement
- Getting Software into Your Cluster
- Self-learning lab: Configuring Pod Autoscaling and Node Pools
- Summary

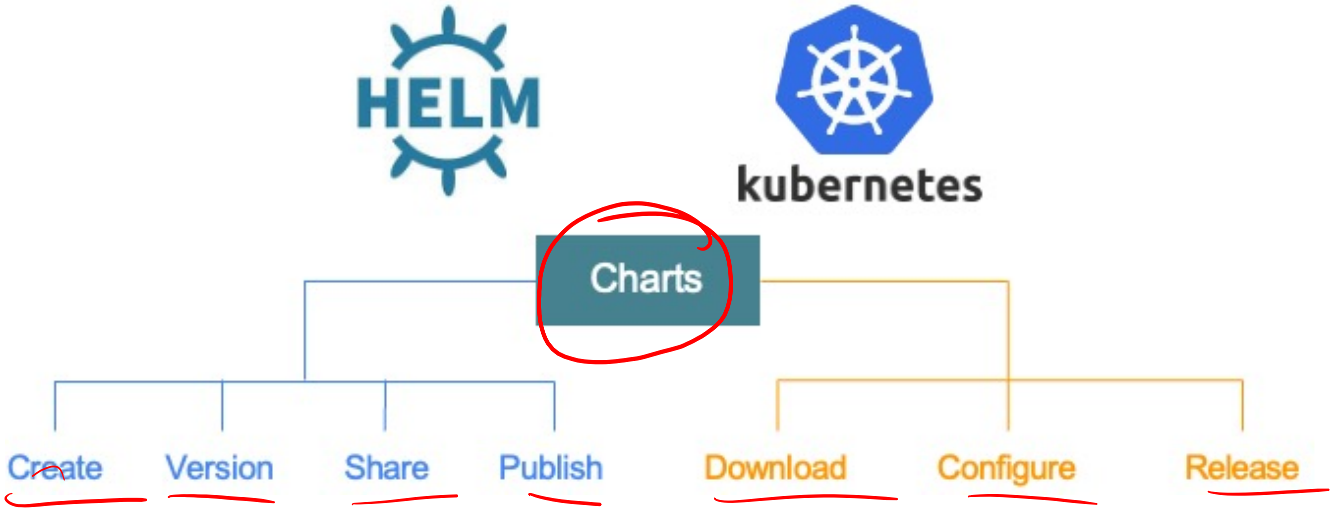
Part 2 Deployments, jobs, and scaling

Getting software into your cluster

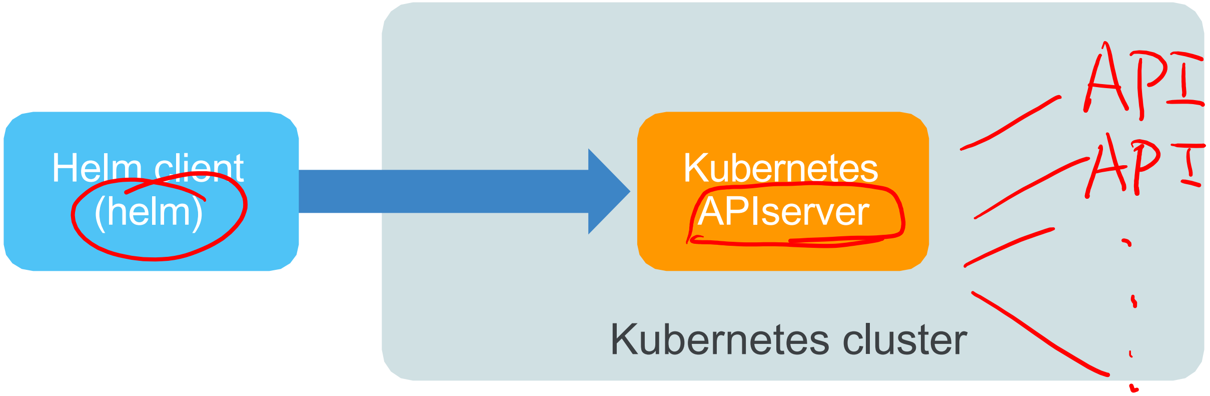
How to get software

- Build it yourself, and supply your own YAML.
- Use Helm to install software into your cluster.
- Use Google Cloud Marketplace to install both open-source and commercial software.

Organize Kubernetes objects in packages and deploy complex packages



Helm interacts directly with the Kubernetes APIserver



Part 2 Deployments, jobs, and scaling

- Deployments
- Self-learning lab: Creating Google Kubernetes Engine Deployments
- Jobs
- Self-learning lab: Deploying Jobs on Google Kubernetes Engine
- Cluster Scaling
- Controlling Pod Placement
- Getting Software into Your Cluster
- Self-learning lab: Configuring Pod Autoscaling and Node Pools
- Summary

Configuring Pod Autoscaling and Node Pools (https://www.youtube.com/watch?v=TDuBmjZqpPQ)

Part 2 Deployments, jobs, and scaling

- Deployments
- Self-learning lab: Creating Google Kubernetes Engine Deployments
- Jobs
- Self-learning lab: Deploying Jobs on Google Kubernetes Engine
- Cluster Scaling
- Controlling Pod Placement
- Getting Software into Your Cluster
- Self-learning lab: Configuring Pod Autoscaling and Node Pools
- Summary

Part 2 Deployments, jobs, and scaling

Summary

Summary

- Create and use Deployments.
- Create and run Jobs.
- Use Helm Charts.
- Scale clusters manually and automatically.

Part 3 Persistent data and storage

- Volumes
- Self-learning lab: Configuring Persistent Storage for Google Kubernetes Engine
- Summary

Part 3 Persistent data and storage

- Volumes
- Self-learning lab: Configuring Persistent Storage for Google Kubernetes Engine
- Summary

Part 3 Persistent data and storage

Volumes

Kubernetes offers storage abstraction options

Volumes

Are a directory which is accessible to all of the containers in a Pod.

Some Volumes are ephemeral.

Some Volumes are persistent.

Persistent Volumes

Manage durable storage in a cluster.

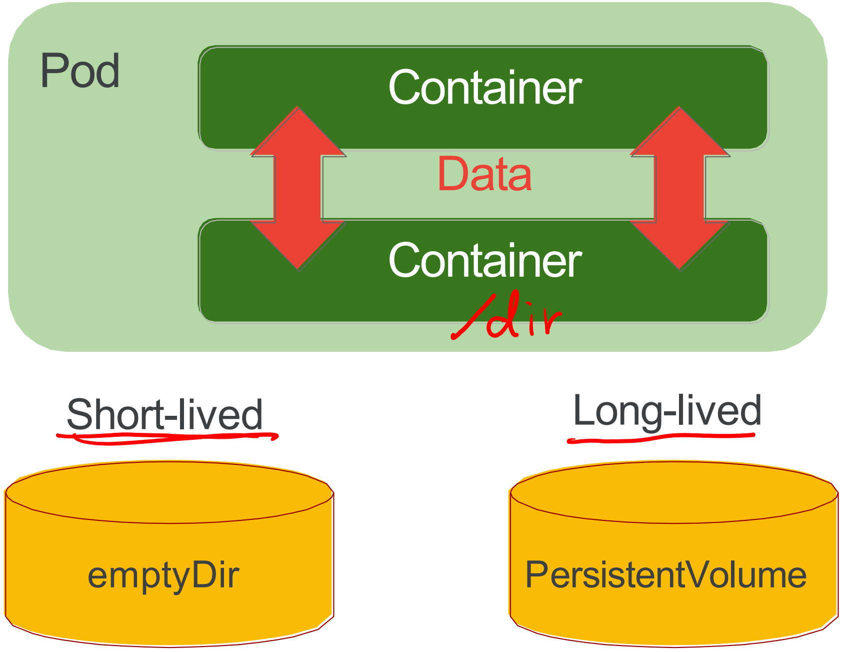
Are independent of the Pod's lifecycle.

Provisioned dynamically through PersistentVolumeClaims or explicitly created by a cluster admin.

Part 3 Persistent data and storage

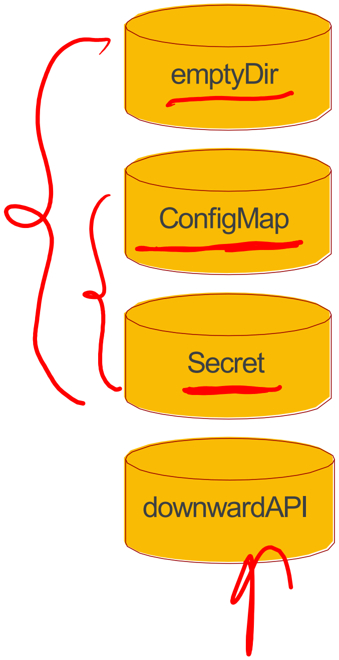
Volumes

Volumes allow containers within a Pod to share data



Volume specifications

Ephemeral volume types

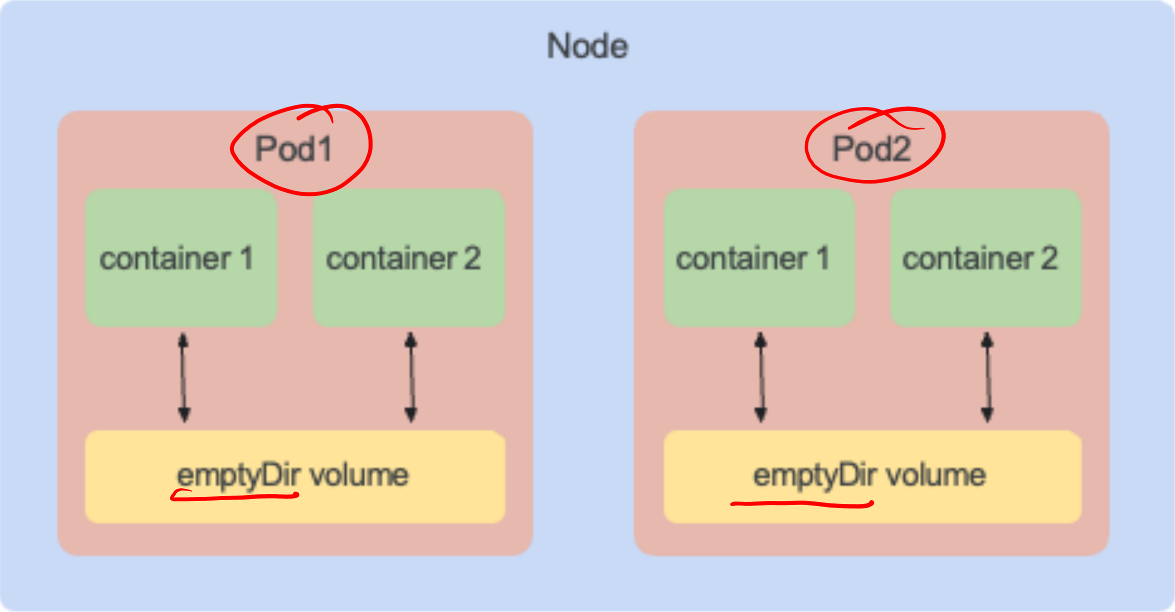


emptyDir	Ephemeral: shares Pod’s lifecycle.
ConfigMap	Object can be referenced in a volume.
Secret	Stores sensitive info, such as passwords.
downwardAPI	Makes data about Pods data available to containers.

Part 3 Persistent data and storage

Volumes

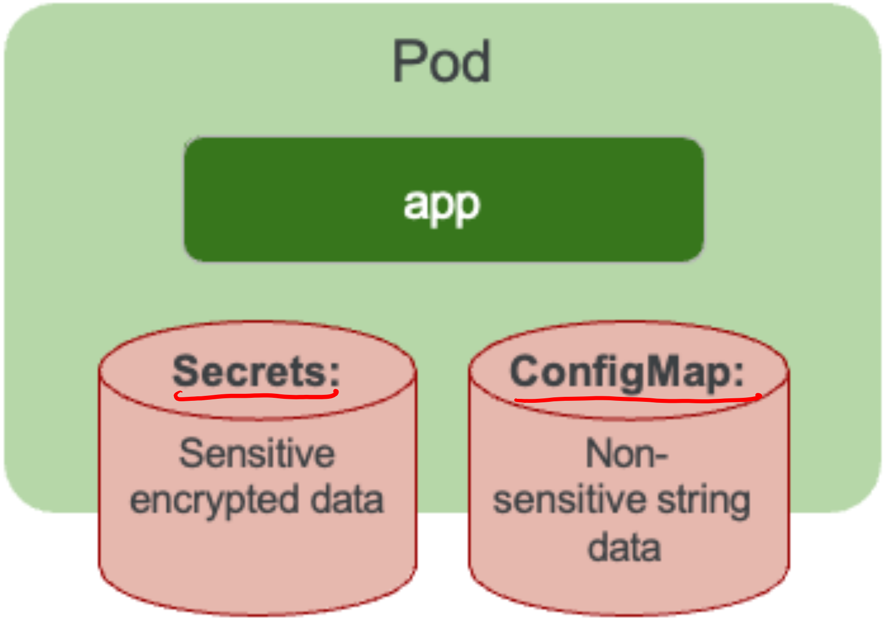
emptyDir volume:
created when a Pod is assigned to a node



Creating a Pod with an emptyDir volume

```
apiVersion: v1
kind: Pod
metadata:
  name: web
spec:
  containers:
  - name: web
    image: nginx
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  volumes:
  - name: cache-volume
    emptyDir: {}
```

Secret and ConfigMap Volumes are ephemeral



Lab: https://www.youtube.com/watch?v=BhC_-qqRdRk

PD

The benefits of PersistentVolumes

PV

Abstracts storage provisioning from storage consumption.

Promotes microservices architecture.

Allows cluster administrators to provision and maintain storage.

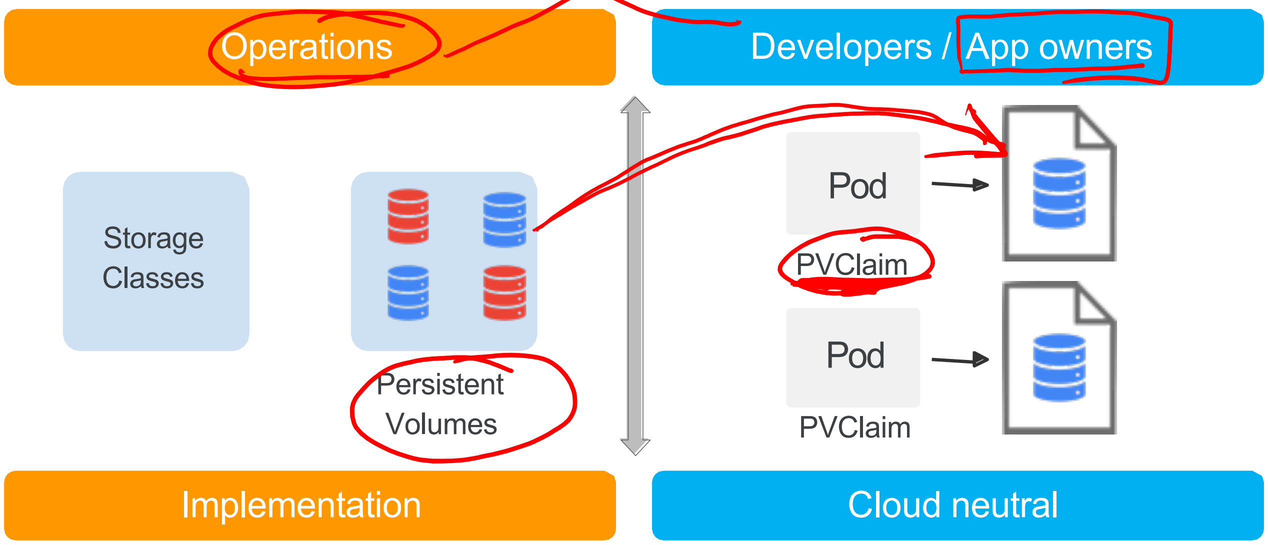
Developers can claim provisioned storage for app consumption.

Part 3 Persistent data and storage

Volumes

Consumer Admin

PersistentVolumeClaims and PersistentVolumes separate storage consumption from provisioning



Part 3 Persistent data and storage

Volumes

Creating a Compute Engine persistent disk using a gcloud command

PD

```
$ gcloud compute disks create
--size=100GB
--zone=us-central1-a  demo-disk
```

Part 3 Persistent data and storage

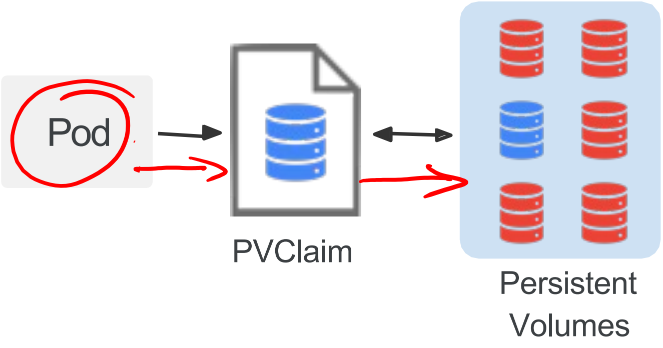
Volumes

PersistentVolumes abstraction has two components

PersistentVolume (PV)

- Independent of a Pod’s lifecycle.
- Managed by Kubernetes.
- Manually or dynamically provisioned.
- Persistent Disks are used by GKE as PersistentVolumes.

PersistentVolumeClaim (PVC)



Pod

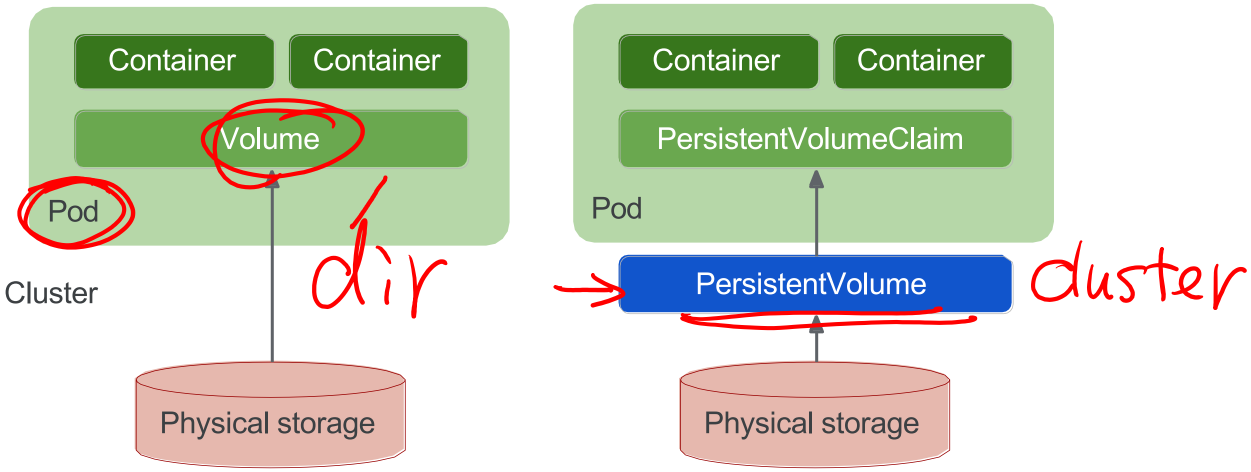
PVClaim

Persistent Volumes

Part 3 Persistent data and storage

Volumes

PersistentVolumes must be claimed



Part 3 Persistent data and storage

Volumes

The AccessModes you specify determine how this Volume can be read from or written to

```
apiVersion:   v1
kind:  PersistentVolume
metadata:
    name:  pd-volume
spec:
    storageClassName: "standard"
    capacity:
        storage:   100G
    accessModes:
     - ReadWriteOnce
    gcePersistentDisk:
        pdName:  demo-disk
        fsType:   ext4
```

node

```
apiVersion:   v1
kind:  PersistentVolume
metadata:
    name:  pd-volume
spec:
    storageClassName:  "standard"
    capacity:
        storage:   100G
    accessModes:
    -  ReadOnlyMany
    gcePersistentDisk:
        pdName:  demo-disk
        fsType:   ext4
```

- node

```
apiVersion:   v1
kind:  PersistentVolume
metadata:
    name:  pd-volume
spec:
    storageClassName:  "nfs"
    capacity:
        storage:   100G
    accessModes:
    -  ReadWriteMany
   nfs:
      path:  /tmp
      server:  172.17.0.2
```

- node

cluter admin

PV & GKE

AZ

Part 3 Persistent data and storage

Volumes

You can create a Persistent Volume from a YAML manifest

```
apiVersion:  v1
kind:  PersistentVolume
metadata:
   name:  pd-volume
spec:
   storageClassName:   "standard"
   capacity:
      storage:  100G
   accessModes:
   -  ReadWriteOnce:
   gcePersistentDisk:
      pdName:  demo-disk
      fsType:  ext4
```

Part 3 Persistent data and storage

Volumes

The modern, easier-to-manage way is to use the PersistentVolume abstraction

```yaml
apiVersion: v1
kind: Pod
metadata:
    name: demo-pod
spec:
    containers:
    - name: demo-container
      image: gcr.io/hello-app:1.0
      volumeMounts:
      - mountPath: /demo-pod
        name: pd-volume
    volumes:
    - name: pd-volume
      PersistentVolumeClaim:
         claimName: pd-volume-claim
```

Part 3 Persistent data and storage

Volumes

The PersistentVolume can be retained when the PersistentVolumeClaim is deleted

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
    name: pd-volume-claim
spec:
    storageClassName: “standard"
    accessModes:
    - ReadWriteOnce:
    resources:
        requests:
            storage: 100G
    persistentVolumeReclaimPolicy: Retain
```

Part 3 Persistent data and storage

- Volumes
- Self-learning lab: Configuring Persistent

Storage for Google Kubernetes Engine

- Summary

Part 3 Persistent data and storage

Lab

Configuring Persistent Storage
for Google Kubernetes Engine
(https://www.youtube.com/watch?v=MaN_deRwrhs)

Part 3 Persistent data and storage

- Volumes
- Self-learning lab: Configuring Persistent Storage for Google Kubernetes Engine
- Summary

Part 3 Persistent data and storage

Summary

Summary

Understand and work with Kubernetes storage abstractions.

Use ConfigMaps to decouple configuration from Pods.

Manage and store sensitive authorization and authentication data.