Skills required

AWS app runner

Amazon ECS

Elastic Load balancing

Auto scaling

Amazon cloudwatch

Amazon ECR

Amazon EKS

AWS Fargate

AWS CloudFormation

AWS lambda

Terraform

Github workflow

MSQl

AWS RDS

NoSQl

CI/CD

Docker

Kubernetes

Kubernetes Highlight

Demo webapp project

Node -v

Npm create-react-app testapp – Y

This will create an folder named test app

Cd testapp

Npm start

Docker build and run check etc

Docker ps

Docker build -t abd/jef .

Docker images

Docker login

Docker push phil/web:02

Minikube status

Kubectl create deployment my-webapp –image=(name of image from dockerhub)

Kubectl get deployment

Kubectl get pods

Kubectl get deployments

Minikube dashboard

Kubectl logs (name of deployment)

Kubectl describe pods

Whenever a pod is created an ip is assigned to it

Kubectl expose deployment my-webapp –type=LoadBalancer –port=3000

Kubectl get services

Minikube service my-webapp

Will get a url -> copy and paste that url in browser

Kubectl set image deployment my-webapp

Till the new version didi not came Kubernetes kept the last version live

Kubectl rollout status deployment my-web -> will tell you the current state of kubernetes

Like facing ImagePullBAckOff issue -> It means image not getting pull

Kubectl rollout undo deployment my-webapp

Self Healing in K8

Kubectl create deployment node-app –image= philipp(name of image on dockerhub)

Kubectl get deployment

Kubectl get pods

Kubectl get services

Kubectl expose deployment node-app—type=Loadbalancer –port=3000

Kubectl scale deploymentnode-app –replicas=4 -> will scale the when it is live -> will increase the number of instances. Here there are 4 instances

Template -> information about pod

Mongo-config.yml

apiVersion: v1

kind: ConfigMap

metadata:

name: mongo-config

data:

MONGO\_HOST : “ service-mongodb”

MONGO\_PORT: 27017

Use -> now go to file like node-app.yml

apiVersion: app/v1

kind: Deployment

metadata:

name: node-app

spec:

replicas:1

env:

* name: MONGO\_HOST

valueFrom:

configMapKeyRef:

name: mongo-config

key: MONGO\_HOST

* name: MONGO\_PORT

valueFrom:

configMapKeyRef:

name: mongo-config

key: MONGO\_PORT

kubectl apply -f mongo-config.yml

configmap/mongo-config created

Storage and volume

lets say you have a container with node.js as pod and mongodb as another to store dataIf .mongodb crashes. Volume is attached to it pod.

Can Store volume in cloud base so it will always be there in cloud.

Persistent volume -> lets name -> host-pv.yml

Apiversion : v1

Kind: PrersistentVolume

Metadata:

Name: host-pv

Spec:

Capacity:

Storage: 1Gi

volumeMode: Filesystem

storageClassName: standard -> kubectl get sc

accessModes:

* ReadwriteMany

Hostpath:

Path:/data/

Type: DirectoryOrCreate

Pvc

Apiversion : v1

Kind: PrersistentVolumeClaim

Metadata:

Name: host-pvc

Spec:

Capacity:

volumeName: host-pvc

volumeMode: Filesystem

storageClassName: standard -> kubectl get sc

accessModes:

* ReadwriteMany

Resources:

Requests:

Storage:1Gi

How to call this file in main file

Spec:

Containers:

* Name: mongo-app
* Image:mongo:latest

volumeMounts:

* mountPath: /data/db
* Name: mongo-vol

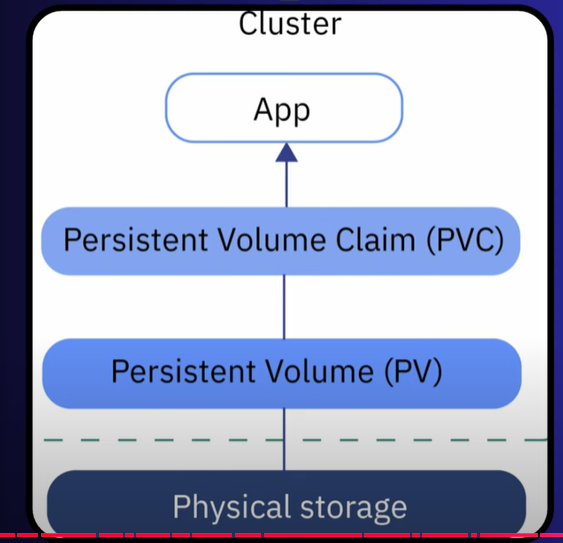
Volumes:

Name: mongo-vol

persistentVolumeClaim:

claimName: host-pvc

Ephemral volume -> volume that exists till the point pod exists



Kubectl get pv

Kubectl apply -f host-pvc.yml

Emptydir -> Emptydir volume, the volume is created when the pod is assigned to a node.

hostPath -> A hostPath volume mounts a file or directory from the host node’s filesystem into your pod. -> not secure -> but can be used when using internal directories

to access a service in Kubernetes local -> minikube service service-node-file-app

minikube service (name of service)

kubectl delete -f node-file-demo.yml

Completed