Pod : Smallest unit of k8s

Service : permanent IP address

External service

Configs maps: for configuration

Secrets :store passwords,certificate

Volumes:

Data storage

Pods restarted data gone

Depolyments

-blueprint for my-app pods

-you create Depolyment

Datebase should be create using stateful sets

**To edit deployment:**

kubectl edit deployment <deploymentname>

**To gets logs**

kubectl logs <podname>

**To check pods**  
kubectl describe pod <podname>

**To get inside the container**

Kubectl exec –it <podname> -- bin/bash

**Delete pods**

kubectl delete deployment < name pod the deployment>

**kubenetes configuration file**

to apply the configuration

#k get replicaset

# k apply –f nginx-deployment.yaml

Create deployment file

# vim nginx-deployment.yaml file

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: nginx

spec:

replicas: 2

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.16

ports:

- containerPort: 80

# kubectl apply –f nginx-deployment.yaml

Deployment.apps/nginx-deployment created

**YAML configuration File in Kubernetes**

Kubernetes file contains three parts

1. Metadata

metadata:

name: nginx-deployment

labels:-

2.Specification

apiVersion: apps/v1

kind:Deployment/service

3.Status

Compares actual state and desired state

Kubernetes automatically updates the states

Etcd current holds state of kubernetes component

Use yaml validator for big files

Layers of Abstraction

1. Deployment

**Create internal service**

**Including in mongo.yaml**

**---**

**apiVersion: v1**

**kind: Service**

**metadata:**

**name: mongodb-service**

**spec:**

**selector:**

**app: mongodb**

**ports:**

**- protocol: TCP**

**port: 27017**

**targetPort: 27017**

# kubectl describe service <service name>

**Command to view all the components and filter service**

#kubectl get all | grep mongodb

**Creating mongo Express Deployment and service and Config Map**

Mongo express deployment and service file

**Workflow**

Step 1 : The request lands in the **external service** of **mongo-express.**

Step 2 : The **mongo-express** connects to **internal service of mongodb**

Step3 : From mongodb to mongodb pods

**Namespaces**

-Organize resources in namespaces

-virtual cluster inside a cluster

4 Namespaces per Default

Kube-system – it is not create or modify in kube-system

Kube-public – public accessible data.

Kube-Node-lease : holds the hear beats of nodes, nodes information

YAML file

apiVersion: V1

kind: Service

metadata:

name: myapp-external-service

spec:

selector:

app: myapp

type: LoadBalancer

ports:

* Protocol: TCP

port: 8080

targetPort: 8080

nodePort: 3510

**ingress YAML file**

**apiVersion: networking.k8s.io/v1beta1**

**kind: Ingress**

**metadata:**

**name: myapp-ingress**

**spec:**

**rules:**

* **host: myapp.com**
* **http:**
* **paths:**
* **-backend:**
* **serviceName: myapp-internal-service**
* **servicePort: 8080**

domain name should be map into entry point of the kubercluster node

**ingress controller use**

-evaluates all the rules

-manages redirections

-entrypoint to cluster

**Many ingress controllers**

-k8s Nginx ingress controller

**In cloud environment**

We don’t have to implement load balancer(aws,azure,gcp)

Proxy server s as a load balancer

Ingress Controller checks Ingress rules

**Install Ingress controller in microk8s**

**Helm**

Package manager for kubernetes

Helm Charts

* Bundle of YAML Files
* Create Your own Helm Charts with Helm
* Push them to helm repository
* Download and use existing ones

Commonly use deployments in Helm Charts

Database Apps

MongoDB

Elastic search

Mysql

Monitoring Apps( promethues)

Helm chart Structure

Directory structure:

Mychart/

Chart.yaml

Values.yaml

charts/

templates/

…

Helm is Templating engine

**Helm features**

**Values Injection into Template files**

**Helm Version 2 comes in two parts**

**1.Client (Helm Cli)**

**2.Server (Tiller)**

**In Helm 3 Tiller removed**

**Kubernetes Volumes**

1. **Persistent Volume**
2. **Persistent Volume Claim**
3. **Storage Class**

**Persistent Volumes**

Reads from the directory

apiVersion: v1

kind: PersistentVolume

Metadata:

Name: pv-name

Spec:

Capacity:

Storage: 5Gi

VolumeMode: Filesystem

accessModes:

* ReadWriteOnce

persistentVolumeReclaimPolicy:

storageClassName: slow

mountOptions:

* Hard
* nfsvers=4.0

nfs:

path: /dir/path/on/nfs/server

server: nfs-server-ip-address

Persistent Volumes are not in namespaces

K8s Admin sets up and maintains the cluster

**Use that PVC in pods configuration**

apiVersion: v1 kind: Persistent VolumesCalim

kind: Pod apiVersion: V1

metadata: metadata:

name: mypod name: pvc-name

spec: spec:

containers: storageClassName: manual

* name: myfrontend volumeMode: Filesystem

image: nginx accessModes:

volumeMounts: - ReadWriteOnce

* MountPath: “/var/www/html” resources:

name: mypod requests:

Volumes: storage: 10Gi

* name: mypod

persistentVolumeClaim:

claimName: pvc-name

Admin provisions storage resource creates **persistent Volumes**

User creates claim to PV . **persistent volume claim**

**Different Volumes types in Pod**

**awsElasticBlockstore**

**Elastic -app**

**secret**

configmaps

kind: Deployment

metadata:

name: elastic

spec:

selector:

matchLabels:

app: elastic

template:

metadata:

labels:

app:elastic

spec:

containers:

* image: elastic:latest

name: elastic-container

ports:

* containerPort: 9200

**volumeMounts:**

* **name: es-persistent-storage**

**mountpath: /var/lib/data**

* **name: es-secret-dir**

**mountPath: /var/lib/secret**

* **name: es-config-dir**

**mountPath : /var/lib/config**

volumes:

* **name: es-persistent-storage**

**persistentVolumeClaim:**

**claimName: es-pv-claim**

* **name: es-secret-dir**

**secret:**

**secretName: es-secret**

* **name: es-config-dir**

**configMap:**

**name: es-config-map**

**Storage Class**

**Creates persistant volumes dynamically**

**YAML file for StorageClass**

apiVersion: storage.k8s.io/v1

kind: StorageClass

metadata:

name: storage-class-name

provisioner: kubernetes.io/aws-ebs

parameters:

type: io1

iopsPerGB: “10”

fsType: ext4

StorageBackend is defines in the SC component

-via “provisioner” attribute

-each storage backend has own provisioner

-internal provisioner – “kubernetes.io”

- external Provisioner

**Storage Class**

1. Admins configure storage
2. Create Perssitent Volumes
3. K8s Users claim PV using PVC

K8s Statefulset

* It is used in stateful applications
* Stateless applications deployed using deployment
* Stateful applications deployed using statefulset

**Kubernetes service**

Each pod has its own IP address

Pods are ephemeral – are destroyed frequently

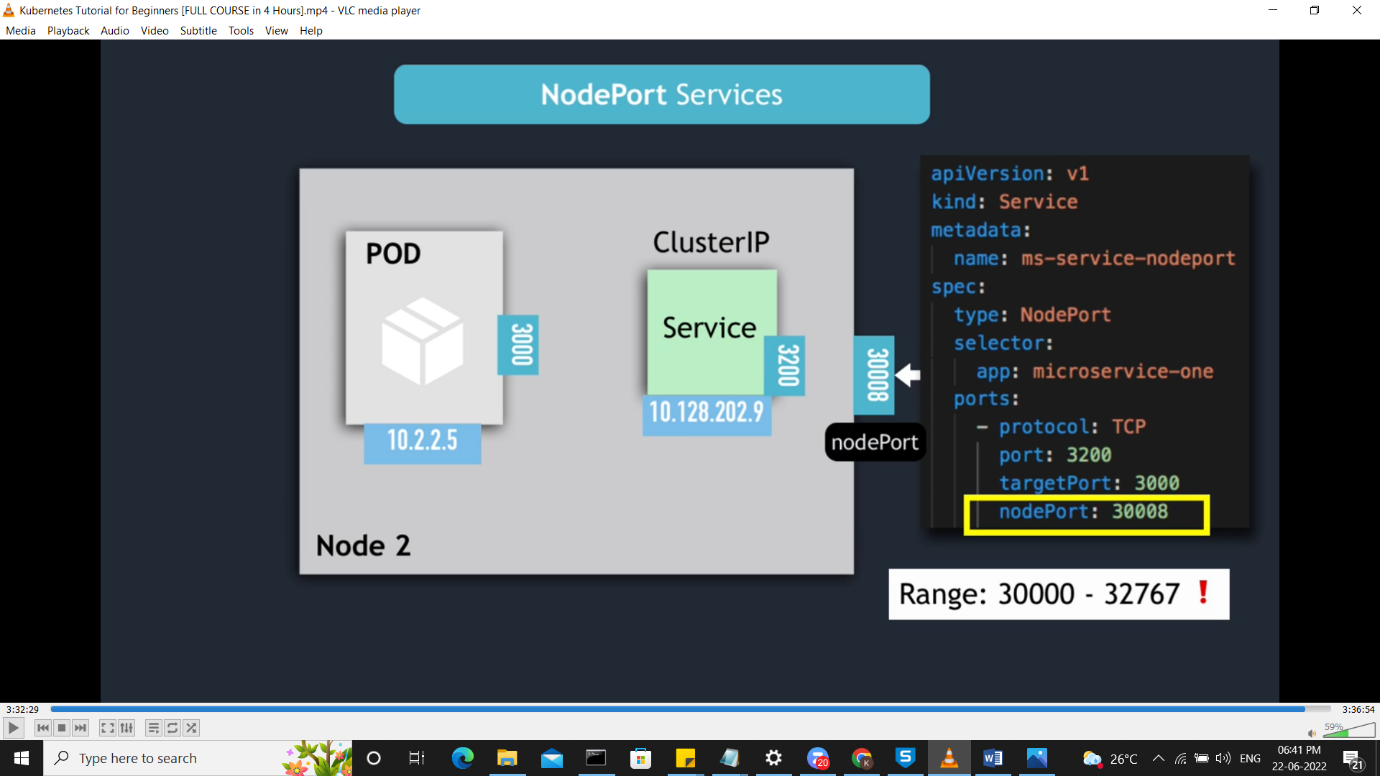
**Service:**

Stable IP address

Load balancing

Loose Coupling

ClusterIP Services



**Deployment strategies in kubernetes**

1.Recreate

2.Rolling update

3.Blue Green

4.Canary

**Recreate deployment**

Delete the pods and creates the pods

**Rollingupdate**

New version without downtime. Once new one pods comes up old pod gets destroyed.

Readiness check the application inside the pod is up or not

**Blue Green**

**EKS (Elastic kubernetes service)**

Managed Kubernetes service

AWS manages Master Nodes

Necessary apps Pre installed

Scaling and backups

You create Worker nodes

Required

eksctl create cluster \

> --name test-cluster \

> --version 1.19 \

> --region ap-south-1 \

> --nodegroup-name linux-nodes \

> --node-type t2.micro \

> --nodes 2

**Kuberentes deployment using K8s**

**Pods.yml**

apiVersion: v1

kind: Pod

metadata:

name: nodeapp

labels:

app: nodeapp

spec:

containers:

- name: nodeapp

image: kammana/nodeapp:tagVersion

ports:

- containerPort: 8080

**Interview questions answers**

**Labels: pod ,config**

**selectors : configuration**

**docker components**

**From,run.copy add etc**

**types of instances**

**generalpurpose**

**memory optimizing**

**computing**

**accelerator**

**storage**

**on master i forgot to take backup**