**K8s Notes**

**Minikube:- Single node cluster setup**

Kubetcl get nodes: list all the available nodes

Kubectl cluster-info

Kubectl run hello-minikube

Kubectl run nginx --image=nginx

Kubectl get pods :-lists all available pods

Kubectl describe pod nginx [pod-name] :gives desc about pods

Kubectl get pods -o wide : describes IP address and which node pod is running

**YAML File in Kubernetes :-**

apiVersion:<version configuration>

kind:<resource type> [Pod,Replicaset,Deployment,Service] The starting letter should be capital CASE SENSITIVE

metadata:

name:

label:  
spec:

Container: [list or array of container]

- Name:

Image:

Create a pod by running this command: Kubectl create -f pod-def.yml or Kubectl apply -f pod-def.yml both are for same purpose

**REPLICA SET**

Kubectl create -f repclia-def.yml

Kubectl get replicaset

Kubectl scale –-replicas=3 replica-def.yml

Kubectl replace -f replica-def.yml

Kubectl delete replicaset myapp-replicaset

Kubectl describe replicaset myapp-replicaset

Kubectl edit replicaset myapp-replicaset

Kubectl scale replicaset myapp-replicaset –replicase=2 [scale down]

**YAML file**

**apiVersion:app/v1**

**kind: ReplicaSet**

**Spec:**

replicas:6

selector:

matchLabels:

**DEPLOYMENTS:-**

**(Deployment = ReplicaSet + Rolling Updates + Rollbacks)**

Kubectl create -f deployment-de.yml

Kubectl get all – gets all deployment,replicaset ,and pods

Kubectl describe deployment myapp-deployment

Syntx for yaml file: (Almost same as replicaset)

apiVersion: app/v1

kind: Deployment

metadata:

labels:

app:

type:

spec:

replicas:

selector:

matchlabels:

type: frontend

template:

metadata:

labels:

**ROLLOUT:**

Kubectl rollout status deployment/myapp-deployment

Kubectl rollout history deployments/myapp-deployment

Kubectl rollout undo deployment/myapp-deployment

Kubectl set image deployment myapp-deployment nginx=nginx:1.18

**Kubernetes Services:**

**** ClusterIP (default)Exposes the Service on a cluster-internal IP. This means the Service is only reachable within the cluster.

 NodePort  
Exposes the Service on each Node’s IP at a static port. Makes the Service accessible outside the cluster by requesting <NodeIP>:<NodePort>.

 LoadBalancer  
Creates an external load balancer (if supported by the cloud provider) that forwards traffic to the Service.

apiVersion:v1

kind: Service

metadate:

spec:

type: NodePort ,ClusterIP,LoadBalancer

ports:

* targetPort: 80

port: 80

nodePort:

**Example 1: ClusterIP Service**

Suppose you have a simple web app deployed with Pods labeled app=web.

**Pod definition snippet:**

yaml

CopyEdit

apiVersion: v1

kind: Pod

metadata:

name: web-pod-1

labels:

app: web

spec:

containers:

- name: web-container

image: nginx

**Service definition:**

yaml

CopyEdit

apiVersion: v1

kind: Service

metadata:

name: web-service

spec:

selector:

app: web

ports:

- protocol: TCP

port: 80 # Port exposed by the service

targetPort: 80 # Port on Pod container

type: ClusterIP # Default, accessible only within cluster

* This Service will forward requests sent to its cluster IP on port 80 to one of the Pods labeled app=web.
* Other Pods inside the cluster can access web-service on port 80.

**Example 2: NodePort Service**

To expose this web app to external clients, use a **NodePort** Service.

yaml

CopyEdit

apiVersion: v1

kind: Service

metadata:

name: web-nodeport-service

spec:

type: NodePort

selector:

app: web

ports:

- port: 80

targetPort: 80

nodePort: 30007 # fixed external port on each Node

* The app is accessible externally via any Node IP address at port 30007.
* For example, if your node's IP is 192.168.1.10, access the app at http://192.168.1.10:30007.

**Example 3: LoadBalancer Service (Cloud)**

On a cloud provider (like AWS, GCP, Azure), you can create a LoadBalancer Service:

yaml

CopyEdit

apiVersion: v1

kind: Service

metadata:

name: web-lb-service

spec:

selector:

app: web

ports:

- port: 80

targetPort: 80

type: LoadBalancer

* The cloud provider will provision an external load balancer.
* You get a public IP or DNS, and the traffic is routed to the Service, which balances it across the Pods.

**ConfigMap:**

-It is used to store non sensitive information such as command line arguments,environment variables,configuration management in key value pairs

Syntax:

apiVersion: v1

kind: ConfigMap

metadata:

name: example-config

data:

APP\_MODE: "production"

LOG\_LEVEL: "info"

WELCOME\_MSG: "Welcome to Kubernetes!"

**Kubernetes Secrets**:

A **Secret** stores sensitive information, such as passwords, OAuth tokens, SSH keys, and other confidential data.

apiVersion:

kind: Secret

metadata:

name: db-secret

type: opaque

data:

username: YWRtaW4= # base64 encoded "admin"

password: MWYyZDFlMmU2N2Rm # base64 encoded "1f2d1e2e67df

**Kubernetes Namespace:**

It is a virtual cluster in a real cluster

 Isolate environments (e.g., dev, test, prod)

 Avoid name conflicts (same pod name in different namespaces is allowed)

 Apply resource limits and access controls separately

kubectl apply -f app.yaml -n dev

**ENTER INTO A POD:**  
kubectl exec -it podname – sh or --/bin/bash