

Mugesh S– 22CSR123 III CSE C DevOps Day 4

Task – Kubernetes, Namespace:

Kubernetes (K8s)

Kubernetes is an open source container orchestration engine for automating deployment, scaling, and management of containerized applications. The open source project is hosted by the Cloud Native Computing Foundation (CNCF).

It provides a scalable and resilient framework for automating the deployment, scaling, and management of applications across clusters of servers.

A SMALL HISTORY OF K8S:

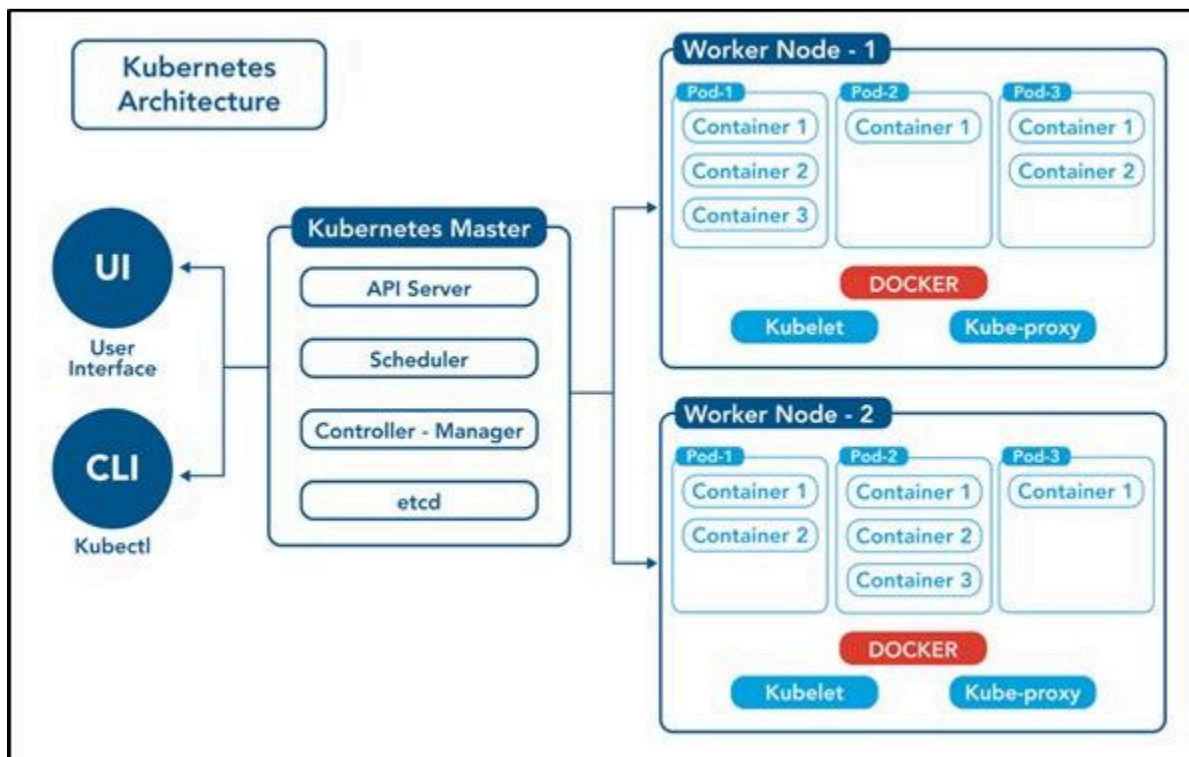
❑ In the early 2000s, Google started developing a system called Borg to manage their internal containerized applications.

❑ Borg enabled Google to run applications at scale, providing features such as automatic scaling, service discovery, and fault tolerance.

❑ In 2014, Google open-sourced a version of Borg called Kubernetes.

❑ Kubernetes was donated to the Cloud Native Computing Foundation (CNCF), a neutral home for open-source cloud-native projects, in July 2015.

- ❓ Kubernetes 1.8 added significant enhancements for storage, security, and networking. Key features included the stable release of the stateful sets API, expanded support for volume plugins, and improvements in security policies.
- ❓ Check URL: <https://kubernetes.io/releases/> for more release details.



Control Plane /Master Node

The control plane's components make global decisions about the cluster (for example, scheduling), as well as detecting and responding to cluster events (for example, starting up a new pod when a deployment's replicas field is unsatisfied).

Control plane components can be run on any machine in the cluster. Do not run user containers on this machine.

Node Components / Worker Nodes

Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.

1. Master Node: The master node is responsible for managing the cluster and coordinating the overall state of the system. It includes the following components:
 - a. API Server: The API server is the central control point for all interactions with the cluster. It exposes the Kubernetes API and handles requests from users and other components.
 - b. Scheduler: The scheduler is responsible for assigning workloads (pods) to individual worker nodes based on resource requirements, constraints, and other policies.
 - c. Controller Manager: The controller manager runs various controllers that monitor the cluster state and drive it towards the desired state. Examples include the replication controller, node controller, and service controller.
 - d. etcd: etcd is a distributed key-value store used by Kubernetes to store cluster state and configuration data.

1. Pod: The basic building block of Kubernetes. A pod represents a single instance of a running process within the cluster. It can encapsulate one or more containers that share the same network and storage resource

1. Create a pod using run command

```
$ kubectl run <pod-name> --image=<image-name> --port=<container-port>
```

```
$ kubectl run my-pod --image=nginx --port=80
```

2. View all the pods

(In default namespace)

```
$ kubectl get pods
```

(In All namespace)

```
$ kubectl get pods -A
```

For a specific namespace

```
$ kubectl get pods -n kube-system
```

For a specific type

```
$ kubectl get pods <pod-name>
$ kubectl get pods <pod-name> -o wide
$ kubectl get pods <pod-name> -o yaml
$ kubectl get pods <pod-name> -o json
```

3. Describe a pod (View Pod details)

```
$ kubectl describe pod <pod-name>
$ kubectl describe pod my-pod
```

4. View Logs of a pod

```
$ kubectl logs <pod-name>
$ kubectl logs my-pod
```

5. Execute any command inside Pod (Inside Pod OS)

```
$ kubectl exec <pod-name> -- <command>
kubectl exec -it my-pod
```

[4:34 PM, 3/20/2025] +91 90928 13114: Namespace (short name = ns):

namespace is a virtual cluster or logical partition within a cluster that provides a way to organize and isolate resources. It allows multiple teams or projects to share the same physical cluster while maintaining resource separation and access control.

[4:34 PM, 3/20/2025] +91 90928 13114: # To create a namespace:

```
$ kubectl create namespace <namespace-name>
```

```
$ kubectl create ns my-bank
```

To switch to a specific namespace: (make this as default type)

```
$ kubectl config set-context --current --namespace=<namespace-name>
```

To list all namespaces:

```
$ kubectl get namespaces
```

To get resources within a specific namespace:

```
$ kubectl get <resource-type> -n <namespace-name>
```

```
$ kubectl get deploy -n my-bank
```

```
$ kubectl get deploy --namespace my-bank
```

```
$ kubectl get all --namespace my-bank
```

To delete a namespace and all associated resources:

```
$ kubectl delete namespace <namespace-name>
```

```
$ kubectl delete ns my-bank
```

Deployment.yml

apiVersion: apps/v1

kind: Deployment

metadata:

name: my-deploy

labels:

name: my-deploy

spec:

replicas: 1

selector:

matchLabels:

apptype: web-backend

strategy:

type: RollingUpdate

template:

metadata:

labels:

apptype: web-backend

spec:

containers:

- name: maven-web-app

image: aswinprabusiva/webapp1:latest

ports:

- containerPort: 8000

apiVersion: v1

kind: Service

metadata:

- name: my-service

labels:

- app: my-service

spec:

- type: NodePort

ports:

- port: 8000

- targetPort: 8080

- nodePort: 30007


```
user@LAPTOP-SSQQTCS9: /mnt/c/Users/MUGESH.V$ minikube service list
NAME      TARGET PORT  URL
default   kube-dns     No node port
kube-system kube-dns     No node port

user@LAPTOP-SSQQTCS9: /mnt/c/Users/MUGESH.V$ kubectl get services --all-namespaces
NAMESPACE  NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
default    kube-dns      ClusterIP     10.96.0.1     <none>         443/TCP          22h
kube-system kube-dns      ClusterIP     10.96.0.1     <none>         53/UDP,53/TCP,9153/TCP 22h

user@LAPTOP-SSQQTCS9: /mnt/c/Users/MUGESH.V$ kubectl get pods
NAME      READY  STATUS   RESTARTS  AGE
my-replicaset-kxvmz  0/1    ImagePullBackOff  0          17h
my-replicaset-qblj6  0/1    ImagePullBackOff  0          17h
my-replicaset-zfc6m  0/1    ErrImagePull      0          17h
No resources found in default namespace.

user@LAPTOP-SSQQTCS9: /mnt/c/Users/MUGESH.V$ kubectl create deployment my-app --image=nginx
deployment.apps/my-app created
user@LAPTOP-SSQQTCS9: /mnt/c/Users/MUGESH.V$ kubectl expose deployment my-app --type=NodePort --port=80 --name=my-service
service/my-service exposed

user@LAPTOP-SSQQTCS9: /mnt/c/Users/MUGESH.V$ kubectl get services
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kube-dns      ClusterIP     10.96.0.1     <none>         443/TCP          22h
my-service    NodePort      10.109.212.58 <none>         80:30684/TCP     4s

user@LAPTOP-SSQQTCS9: /mnt/c/Users/MUGESH.V$ minikube service my-service
NAME      TARGET PORT  URL
default   my-service    80      http://192.168.49.2:30684

Starting tunnel for service my-service.
NAME      TARGET PORT  URL
default   my-service    80      http://127.0.0.1:42929

Opening service default/my-service in default browser...
http://127.0.0.1:42929
Because you are using a Docker driver on linux, the terminal needs to be open to run it.
```

```
user@LAPTOP-SSQQTCS9: /mnt/c/Users/MUGESH.V$ minikube start
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro

System information as of Fri Mar 21 03:37:36 UTC 2025

System load: 0.9          Processes: 47
Usage of /: 0.6% of 1006.85GB Users logged in: 0
Memory usage: 21%        IPv4 address for eth0: 172.19.223.2
Swap usage: 0%

This message is shown once a day. To disable it please create the
/home/user/.hushlogin file.
user@LAPTOP-SSQQTCS9: /mnt/c/Users/MUGESH.V$ minikube start
minikube v1.35.0 on Ubuntu 24.04 (amd64)
Using the docker driver based on existing profile
Starting "minikube" primary control-plane node in "minikube" cluster
Pulling base image v0.0.46 ...
Restarting existing docker container for "minikube" ...
Failing to connect to https://registry.k8s.io/ from inside the minikube container
To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
Verifying Kubernetes components...
* Using image gcr.io/k8s-minikube/storage-provisioner:v5
Enabled addons: default-storageclass, storage-provisioner
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
user@LAPTOP-SSQQTCS9: /mnt/c/Users/MUGESH.V$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

user@LAPTOP-SSQQTCS9: /mnt/c/Users/MUGESH.V$ kubectl get pod
NAME      READY  STATUS   RESTARTS  AGE
my-replicaset-kxvmz  0/1    Error    0          17h
my-replicaset-qblj6  0/1    ImagePullBackOff  0          16h
my-replicaset-zfc6m  0/1    ErrImagePull      0          17h

user@LAPTOP-SSQQTCS9: /mnt/c/Users/MUGESH.V$ sudo nano docker-compose.yml
[sudo] password for user:
```