

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

```
saran@FingerGripPC:~$ kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
my-deploy-6fc494dbf6-4wh4f         0/1     ErrImagePull  0           57s
saran@FingerGripPC:~$ kubectl delete all --all
pod "my-deploy-6fc494dbf6-4wh4f" deleted
service "kubernetes" deleted
service "my-service" deleted
deployment.apps "my-deploy" deleted
saran@FingerGripPC:~$ kubectl apply -f deployment.yml
deployment.apps/my-deploy created
service/my-service created
saran@FingerGripPC:~$ kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
my-deploy-6fc494dbf6-grn4n         1/1     Running    0            5s
saran@FingerGripPC:~$ kubectl get deploy
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
my-deploy 1/1      1             1           12s
saran@FingerGripPC:~$ minikube service my-service
|-----|
| NAMESPACE | NAME      | TARGET PORT | URL                               |
|-----|
| default   | my-service | 8000         | http://192.168.49.2:30007       |
|-----|
* Starting tunnel for service my-service.
|-----|
| NAMESPACE | NAME      | TARGET PORT | URL                               |
|-----|
| default   | my-service |             | http://127.0.0.1:43609         |
|-----|
🔗 Opening service default/my-service in default browser...
🔗 http://127.0.0.1:43609
! Because you are using a Docker driver on linux, the terminal needs to be open to run it.
^C 🛑 Stopping tunnel for service my-service.
saran@FingerGripPC:~$ curl http://192.168.49.2:30007/maven-web-app
saran@FingerGripPC:~$ curl http://192.168.49.2:30007/maven-web-app/
<html>
<body>
<h2>Hello World!</h2>
</body>
</html>
saran@FingerGripPC:~$
```