

# Lab Exercise 6- Create POD in Kubernetes

**Name : Raman Boora**

**SapID: 500109408**

**Roll no: R2142221160**

## Objective:

- Understand the basic structure and syntax of a Kubernetes Pod definition file (YAML).
- Learn to create, inspect, and delete a Pod in a Kubernetes cluster.

## Prerequisites

- Kubernetes Cluster: You need a running Kubernetes cluster. You can set up a local cluster using tools like Minikube or kind, or use a cloud-based Kubernetes service.
- kubectl: Install and configure kubectl to interact with your Kubernetes cluster.
- Basic Knowledge of YAML: Familiarity with YAML format will be helpful as Kubernetes resource definitions are written in YAML.

## Step-by-Step Guide

### Step 1: Create a YAML File for the Pod

We'll create a Pod configuration file named **pod-example.yaml**

```
apiVersion: v1
kind: Pod
metadata:
```

```
annotations:
labels:
  name: easy-drive-pod
name: easy-drive-pod
spec:
  containers:
  - image: booraraman/easy-drive-rentals:1
    name: easy-drive
    ports:
    - containerPort: 5600
```

## Explanation of the YAML File

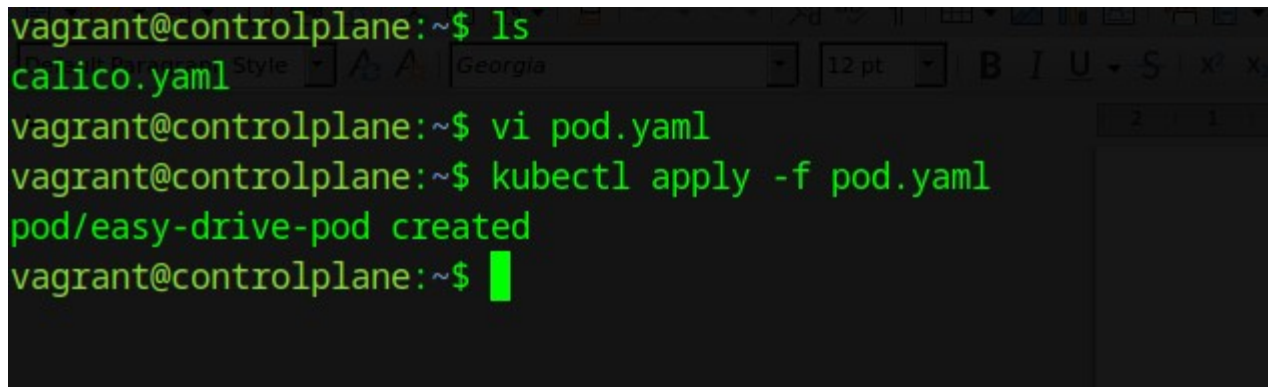
- **apiVersion:** Specifies the version of the Kubernetes API to use. For Pods, it's typically v1.
- **kind:** The type of object being created. Here it's a Pod.
- **metadata:** Provides metadata about the object, including name and labels. The name must be unique within the namespace, and labels help in identifying and organizing Pods.
- **spec:** Contains the specifications of the Pod, including:
  - o **containers:** Lists all containers that will run inside the Pod. Each container needs:
    - **name:** A unique name within the Pod.
    - **image:** The Docker image to use for the container.
    - **ports:** The ports that this container exposes.
    - **env:** Environment variables passed to the container.

## Step 2: Apply the YAML File to Create the Pod

Use the `kubectl apply` command to create the Pod based on the YAML configuration file.

```
kubectl apply -f pod.yaml
```

This command tells Kubernetes to create a Pod as specified in the `pod-example.yaml` file.

A terminal window with a dark background and green text. The prompt is 'vagrant@controlplane:~\$'. The user enters 'ls', and the output is 'calico.yaml'. The user then enters 'vi pod.yaml'. Next, the user enters 'kubectl apply -f pod.yaml', and the output is 'pod/easy-drive-pod created'. The prompt returns to 'vagrant@controlplane:~\$' with a green cursor. A text editor window is visible in the background.

```
vagrant@controlplane:~$ ls
calico.yaml
vagrant@controlplane:~$ vi pod.yaml
vagrant@controlplane:~$ kubectl apply -f pod.yaml
pod/easy-drive-pod created
vagrant@controlplane:~$
```

### Step 3: Verify the Pod Creation

To check the status of the Pod and ensure it's running, use:

```
kubectl get pods
```

This command lists all the Pods in the current namespace, showing their status, restart count, and other details.

```
vagrant@controlplane:~$ vi pod.yaml
vagrant@controlplane:~$ kubectl apply -f pod.yaml
pod/easy-drive-pod created
vagrant@controlplane:~$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
easy-drive-pod	0/1	ContainerCreating	0	26s

```
vagrant@controlplane:~$
```

You can get detailed information about the Pod using:

```
kubectl describe pod my-pod
```

This command provides detailed information about the Pod, including its events, container specifications, and resource usage.

```

vagrant@controlplane:~$ kubectl get pods
NAME          READY   STATUS             RESTARTS   AGE
easy-drive-pod 0/1     ContainerCreating   0          80s
vagrant@controlplane:~$ kubectl describe pods easy-drive-pod
Name:          easy-drive-pod
Namespace:     default
Priority:       0
Service Account: default
Node:          node02/10.0.0.12
Start Time:    Sun, 03 Nov 2024 09:26:43 +0000
Labels:        name=easy-drive-pod
Annotations:    <none>
Status:        Pending
IP:            <none>
IPs:           <none>
Containers:
  easy-drive:
    Container ID:
    Image:        booraraman/easy-drive-rentals:1
    Image ID:
    Port:         5600/TCP
    Host Port:    0/TCP
    State:        Waiting
      Reason:     ContainerCreating
    Ready:        False
    Restart Count: 0
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-kf7r5 (ro)

```

This command lists all the status, restart count, and other details of the pods in the cluster.

```

vagrant@controlplane:~$ kubectl describe pod easy-drive-pod
NAME          READY   STATUS             RESTARTS   AGE
easy-drive-pod 0/1     ContainerCreating   0          80s
vagrant@controlplane:~$

```

You can get detailed information about a specific pod by using the `kubectl describe pod my-pod` command.

## Step 4: Interact with the Pod

You can interact with the running Pod in various ways, such as accessing the logs or executing commands inside the container.

**View Logs:** To view the logs of the container in the Pod:

```
kubectl logs my-pod
```

```
vagrant@controlplane:~$ kubectl logs easy-drive-pod
[nodemon] 3.1.4
[nodemon] to restart at any time, enter 'rs'
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting 'node app.js'
(node:15) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
(Use 'node --trace-warnings ...' to show where the warning was created)
(node:15) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
Server running on port 6500
```

**Execute a Command: To run a command inside the container:**

```
kubectl exec -it my-pod -- /bin/bash
```

The `-it` flag opens an interactive terminal session inside the container, allowing you to run commands.

```
vagrant@controlplane:~$ kubectl exec -it easy-drive-pod -- /bin/bash
root@easy-drive-pod:/app# ls
Dockerfile app.js middleware models mongoo node_modules package-lock.json package.json public readme.md routes views
root@easy-drive-pod:/app#
```

## Step 5: Delete the Pod

To clean up and remove the Pod when you're done, use the following command:

```
kubectl delete pod my-pod
```

This command deletes the specified Pod from the cluster.

```
vagrant@controlplane:~$ kubectl delete pods easy-drive-pod  
pod "easy-drive-pod" deleted
```