

Lab Exercise 8 :- Create POD in Kubernetes

Name:- Vansh Bhatt

Sap ID:- 500125395

Batch:- DevOps B1

To:- Hitesh Sharma Sir

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

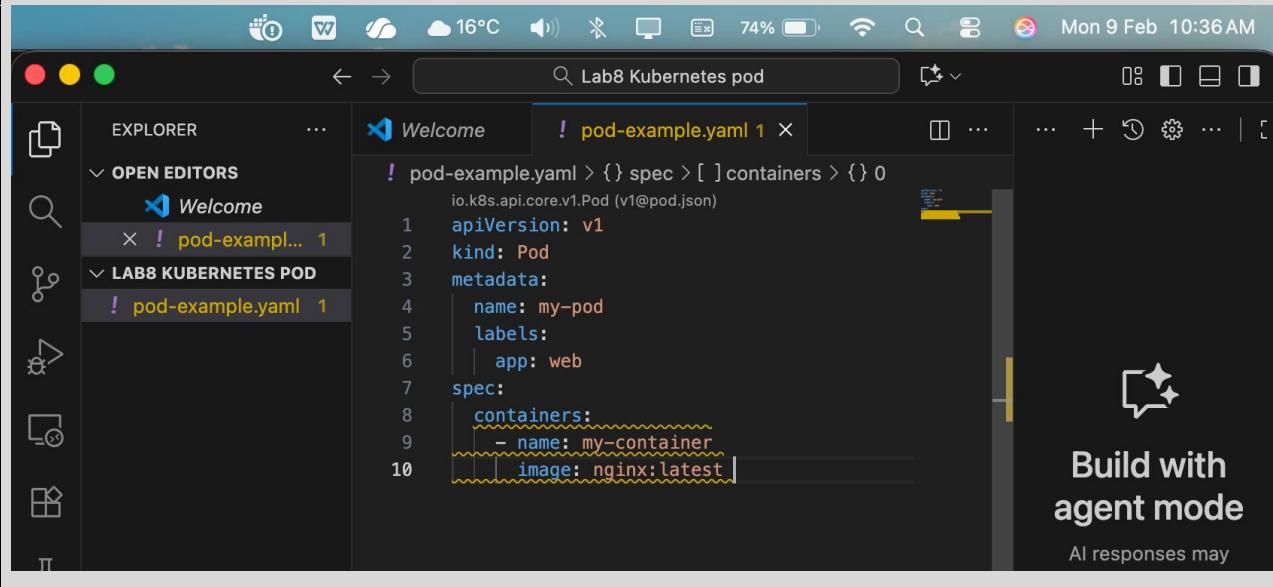
- Kubernetes Cluster: You need a running Kubernetes cluster. You can set up a local cluster using tools like Minikube or kind, or use a cloud-based Kubernetes service.
- kubectl: Install and configure kubectl to interact with your Kubernetes cluster.
- Basic Knowledge of YAML: Familiarity with YAML format 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:
  name: my-pod
  labels:
    app: web
spec:
  containers:
    - name: my-container
      image: nginx:latest
```



The screenshot shows a dark-themed code editor interface. The title bar reads "Lab8 Kubernetes pod". The left sidebar has sections for "EXPLORER", "OPEN EDITORS", and "LAB8 KUBERNETES POD". Under "OPEN EDITORS", there is a file named "pod-example.yaml" which is currently selected. The main editor area displays the YAML code provided above. The code is syntax-highlighted, with "apiVersion", "kind", "name", "labels", "spec", and "containers" keywords in blue, and "my-pod", "web", "my-container", and "nginx:latest" values in black. The number 1 is shown next to the file name in the sidebar and the line numbers 1 through 10 in the editor. A status bar at the bottom right says "Build with agent mode" and "AI responses may".

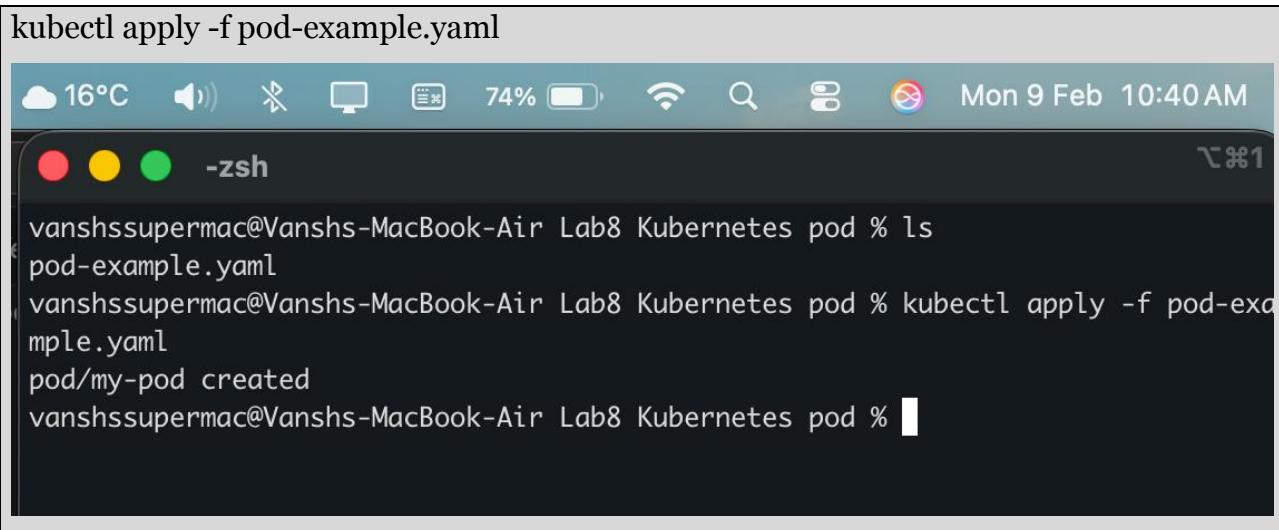
Explanation of the YAML File

- `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:
 - `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-example.yaml
```



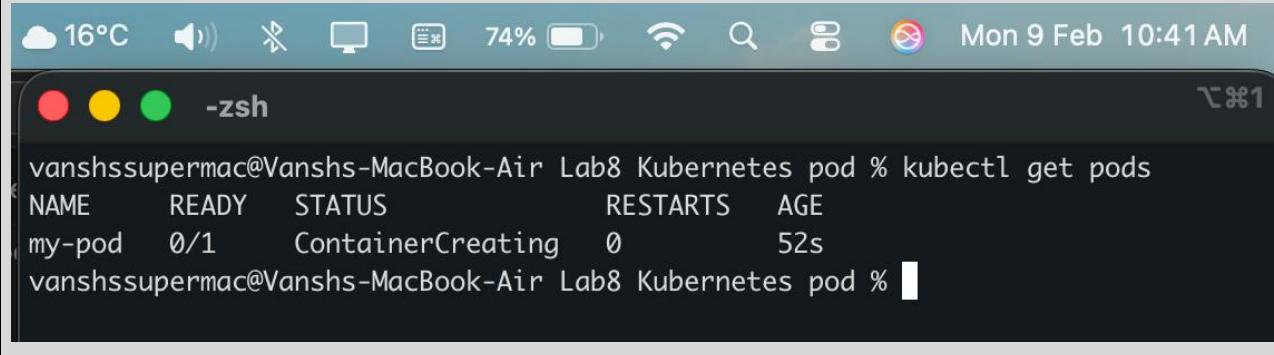
```
vanshssupermac@Vanshs-MacBook-Air Lab8 Kubernetes pod % ls
pod-example.yaml
vanshssupermac@Vanshs-MacBook-Air Lab8 Kubernetes pod % kubectl apply -f pod-exa
mple.yaml
pod/my-pod created
vanshssupermac@Vanshs-MacBook-Air Lab8 Kubernetes pod %
```

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

Step 3: Verify the Pod Creation

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

```
kubectl get pods
```



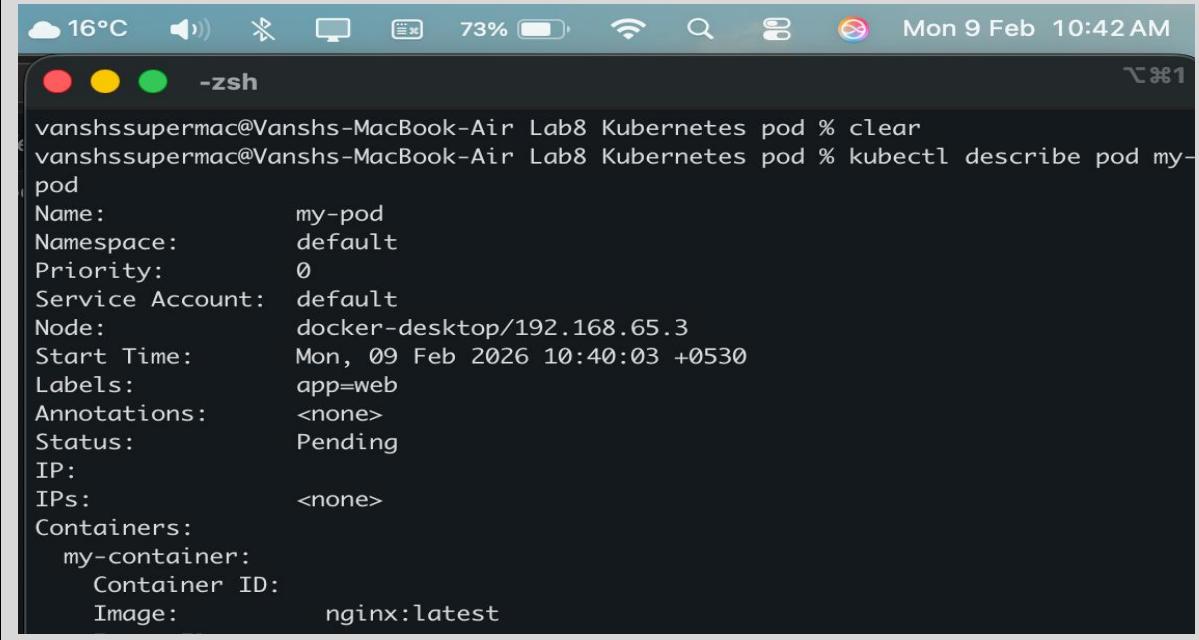
A screenshot of a macOS terminal window titled "-zsh". The window shows the command "kubectl get pods" being run. The output lists a single pod named "my-pod" with a status of "ContainerCreating". The terminal window has a dark background and includes a system menu bar at the top with icons for weather, battery, and date.

```
vanshssupermac@Vanshs-MacBook-Air Lab8 Kubernetes pod % kubectl get pods
NAME      READY   STATUS            RESTARTS   AGE
my-pod    0/1     ContainerCreating   0          52s
vanshssupermac@Vanshs-MacBook-Air Lab8 Kubernetes pod %
```

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

You can get detailed information about the Pod using:

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



A screenshot of a macOS terminal window titled "-zsh". The window shows the command "kubectl describe pod my-pod" being run. The output provides detailed information about the pod, including its name, namespace, service account, node, start time, labels, annotations, status, and containers. The terminal window has a dark background and includes a system menu bar at the top with icons for weather, battery, and date.

```
vanshssupermac@Vanshs-MacBook-Air Lab8 Kubernetes pod % clear
vanshssupermac@Vanshs-MacBook-Air Lab8 Kubernetes pod % kubectl describe pod my-pod
Name:           my-pod
Namespace:      default
Priority:       0
Service Account: default
Node:           docker-desktop/192.168.65.3
Start Time:     Mon, 09 Feb 2026 10:40:03 +0530
Labels:         app=web
Annotations:    <none>
Status:         Pending
IP:             <none>
IPs:            <none>
Containers:
  my-container:
    Container ID:
    Image:        nginx:latest
```

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

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.

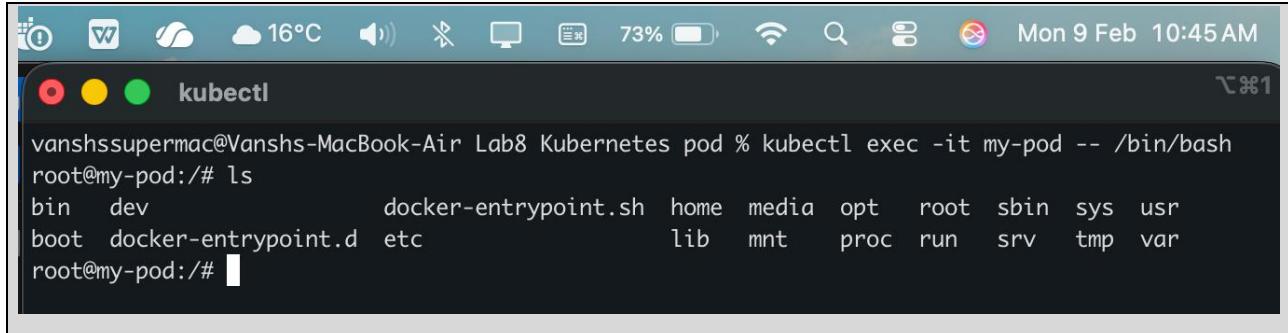
View Logs: To view the logs of the container in the Pod:

```
kubectl logs my-pod
```

```
vanshssupermac@Vanshs-MacBook-Air Lab8 Kubernetes pod % kubectl logs my-pod
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2026/02/09 05:12:44 [notice] 1#1: using the "epoll" event method
2026/02/09 05:12:44 [notice] 1#1: nginx/1.29.5
```

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

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

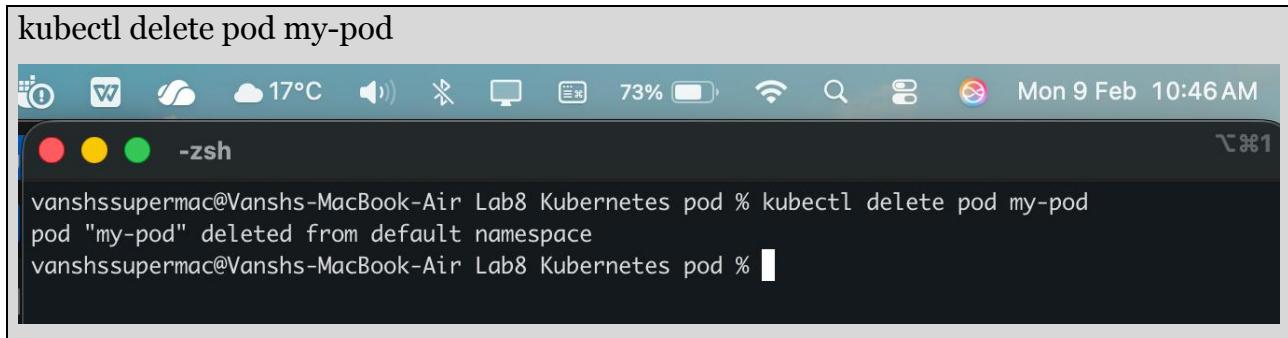


```
vanshssupermac@Vanshs-MacBook-Air Lab8 Kubernetes pod % kubectl exec -it my-pod -- /bin/bash
root@my-pod:/# ls
bin  dev  docker-entrypoint.sh  home  media  opt  root  sbin  sys  usr
boot  docker-entrypoint.d  etc      lib   mnt   proc  run  srv  tmp  var
root@my-pod:/#
```

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

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
vanshssupermac@Vanshs-MacBook-Air Lab8 Kubernetes pod % kubectl delete pod my-pod
pod "my-pod" deleted from default namespace
vanshssupermac@Vanshs-MacBook-Air Lab8 Kubernetes pod %
```

This command deletes the specified Pod from the cluster.

Thank You