

Lab Exercise 8- Create POD in Kubernetes

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
```

```
pod-example.yaml 1 X
lab > pod-example.yaml > {} spec > [ ] containers > {} 0 > image
io.k8s.api.core.v1.Pod (v1@pod.json)
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: my-pod
5   labels:
6     app: web
7 spec:
8   containers:
9     - name: my-container
10    image: nginx:latest
```

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
```

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
```

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
```

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

```
Name:           my-pod
Namespace:      default
Priority:       0
Service Account: default
Node:           docker-desktop/192.168.65.3
Start Time:     Mon, 09 Feb 2026 10:44:42 +0530
Labels:          app=web
Annotations:    <none>
Status:         Running
IP:             10.1.0.12
IPs:            IP: 10.1.0.12
Containers:
  my-container:
    Container ID:  docker://f6ea5b088f01001c9dc12ee7b91a4bb14a0e14df22b167bf717e6374b2fa0f06
    Image:          nginx:latest
    Image ID:      docker-pullable://nginx@sha256:341bf0f3ce6c5277d6002cf6e1fb8319fa4252add24ab6a0e262e0056d313208
    Port:          <none>
    Host Port:    <none>
    State:        Running
    Started:     Mon, 09 Feb 2026 10:44:54 +0530
    Ready:        True
    Restart Count: 0
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-cxzcc (ro)
Conditions:
  Type        Status
  PodReadyToStartContainers  True
  Initialized  True
  Ready        True
  ContainersReady  True
  PodScheduled  True
Volumes:
  kube-api-access-cxzcc:
    Type:          Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3600
    ConfigMapName:   kube-root-ca.crt
    Optional:       false
    DownwardAPI:    true
    QoS Class:     BestEffort
    Node-Selectors: <none>
    Tolerations:   node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                   node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type  Reason  Age   From            Message
  Normal Scheduled  25s  default-scheduler  Successfully assigned default/my-pod to docker-desktop
  Normal Pulling   24s  kubelet         Pulling image "nginx:latest"
  Normal Pulled   13s  kubelet         Successfully pulled image "nginx:latest" in 10.376s (10.376s including waiting). Image size: 61268012 bytes.
  Normal Created   13s  kubelet         Created container: my-container
  Normal Started   13s  kubelet         Started container my-container

```

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
```

```
/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 05:14:54 [notice] 1#1: using the "epoll" event method
2026/02/09 05:14:54 [notice] 1#1: nginx/1.29.5
2026/02/09 05:14:54 [notice] 1#1: built by gcc 14.2.0 (Debian 14.2.0-19)
2026/02/09 05:14:54 [notice] 1#1: OS: Linux 6.12.65-linuxkit
2026/02/09 05:14:54 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2026/02/09 05:14:54 [notice] 1#1: start worker processes
2026/02/09 05:14:54 [notice] 1#1: start worker process 28
2026/02/09 05:14:54 [notice] 1#1: start worker process 29
2026/02/09 05:14:54 [notice] 1#1: start worker process 30
2026/02/09 05:14:54 [notice] 1#1: start worker process 31
2026/02/09 05:14:54 [notice] 1#1: start worker process 32
2026/02/09 05:14:54 [notice] 1#1: start worker process 33
2026/02/09 05:14:54 [notice] 1#1: start worker process 34
2026/02/09 05:14:54 [notice] 1#1: start worker process 35
2026/02/09 05:14:54 [notice] 1#1: start worker process 36
2026/02/09 05:14:54 [notice] 1#1: start worker process 37
```

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.

```
root@my-pod:/# ls
bin boot dev docker-entrypoint.d docker-entrypoint.sh etc home lib media mnt opt proc root run sbin srv sys tmp usr var
root@my-pod:/# ls -a
. . . dockerenv bin boot dev docker-entrypoint.d docker-entrypoint.sh etc home lib media mnt opt proc root run sbin srv sys tmp usr var
root@my-pod:/# cat lab.txt
cat: lab.txt: No such file or directory
root@my-pod:/# touch lab.txt
root@my-pod:/# cat > lab.txt
This is a DevOps lab.
root@my-pod:/# cat lab.txt
This is a DevOps lab.
root@my-pod:/# ^C
root@my-pod:/# exit
exit
command terminated with exit code 130
```

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.

```
pod "my-pod" deleted from default namespace
```