### **Lab Exercise 1.1- Create Multi-Container POD in Kubernetes**

### **Step-by-Step Guide**

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

Creating and accessing a multi-container pod in Kubernetes involves defining a pod specification that includes multiple containers within the same pod, deploying that pod, and then interacting with the containers inside it. Below is a step-by-step guide to achieving this.

**1. Create a Multi-Container Pod**

First, create a YAML file that defines the pod with multiple containers. Below is an example YAML file (multi-container-pod.yaml) that defines a pod with two containers:

apiVersion: v1

kind: Pod

metadata:

name: multi-container-pod

labels:

app: multi-container-app

spec:

containers:

- name: nginx-container

image: nginx:1.21

ports:

- containerPort: 80

- name: busybox-container

image: busybox

nginx-container: Runs an Nginx web server.

busybox-container: Runs a simple loop that outputs a message every hour.

**2. Deploy the Pod**

Use kubectl to create the pod from the YAML file:

kubectl apply -f multi-container-pod.yaml

This command will create the pod named multi-container-pod with the two containers specified.

**3. Verify the Pod is Running**

Check the status of the pod to ensure that it is running:

kubectl get pods

You should see an entry for multi-container-pod with a status of Running.

**4. Access the Containers in the Pod**

**a. Access the nginx-container**

To access the nginx-container, you can use kubectl exec to open a shell session inside the container:

kubectl exec -it multi-container-pod -c nginx-container -- /bin/bash

-it: Interactive terminal.

multi-container-pod: The name of the pod.

-c nginx-container: Specifies which container to access.

-- /bin/bash: Command to run inside the container.

This will give you a shell inside the nginx-container.

**b. Access the busybox-container**

Similarly, to access the busybox-container:

kubectl exec -it multi-container-pod -c busybox-container -- /bin/sh

This will open a shell in the busybox-container.

**5. Clean Up**

To delete the pod when you are done:

kubectl delete pod multi-container-pod

This command will remove the pod and all the containers within it.

**Summary**

* Multi-container pods are used to run tightly coupled applications that need to share resources such as storage and network.
* Accessing specific containers within a pod can be done using kubectl exec with the -c flag to specify the container name.
* Managing logs for each container is done via kubectl logs with the appropriate container name.
* This setup is typical in Kubernetes for scenarios where multiple containers need to work together within the same pod, sharing the same network namespace, and possibly sharing storage volumes.