### **Lab Exercise 2- Create Service in Kubernetes**

### **Objective:**

* Understand the syntax and structure of a Kubernetes Service definition file (YAML).
* Learn to create different types of Services: ClusterIP, NodePort, and LoadBalancer.
* Comprehend how Services operate independently of specific Pods.

### **Prerequisites**

* Kubernetes Cluster: Have a running Kubernetes cluster (locally using Minikube or kind, or a cloud-based service).
* kubectl: Install and configure kubectl to interact with your Kubernetes cluster.
* Basic Knowledge of YAML: Familiarity with YAML format will be helpful for understanding Kubernetes resource definitions.

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

**Step 1: Understand Kubernetes Services**

In Kubernetes, Services provide a way to expose applications running on Pods. They allow these applications to communicate internally and externally, providing a stable interface despite the dynamic nature of Pods.

**Service Types**

* ClusterIP: Exposes the Service on a cluster-internal IP. This is the default type and is accessible only within the cluster.
* NodePort: Exposes the Service on each Node's IP at a static port. This makes the Service accessible from outside the cluster using the Node's IP and the NodePort.
* LoadBalancer: Exposes the Service externally using a cloud provider's load balancer. It provides an external IP address that forwards traffic to the Service.

**Step 2: Create Services**

Even though Services usually route traffic to Pods, you can define and explore Services without explicitly having Pods, understanding their syntax and operational concepts.

**Example YAML Configurations**

1. ClusterIP Service

* Create a YAML file named clusterip-service.yaml with the following content:

apiVersion: v1 # Specifies the API version used.

kind: Service # The type of resource being defined; here, it's a Service.

metadata:

name: clusterip-service # The name of the Service.

spec:

selector: # Normally specifies the Pods the Service should target based on labels.

app: my-app # Since we're not creating Pods, this is just illustrative.

ports: # The list of ports that the Service will expose.

- protocol: TCP # The protocol used by the Service (typically TCP).

port: 80 # The port that will be exposed by the Service.

targetPort: 80 # The port on the Pods that the traffic will be directed to.

type: ClusterIP # The type of Service. ClusterIP exposes the Service only within the cluster.

**Explanation:**

* apiVersion: Defines the API version (v1) used for the Service resource.
* kind: Specifies that this resource is a Service.
* metadata: Contains metadata about the Service, including name.
  + name: The unique name for the Service.
* spec: Provides the specification for the Service.
  + selector: In a real scenario, this would define how the Service selects the Pods it routes traffic to, based on labels.
  + ports: Lists the ports that the Service will expose.
    - protocol: The protocol used (TCP in this case).
    - port: The port number exposed by the Service to the outside.
    - targetPort: The port on the Pod to which the traffic will be routed. Although we're not creating Pods, this field is required.
* type: The type of Service (ClusterIP), meaning it is accessible only within the cluster.

**Apply this YAML to create the ClusterIP Service:**

kubectl apply -f clusterip-service.yaml

**Verify the Service is running:**

kubectl get services

You should see the clusterip-service listed with a ClusterIP and an internal IP address.

**2. NodePort Service**

To expose the Service on a port on each Node in the cluster, modify the Service type to NodePort.

Create a YAML file named nodeport-service.yaml with the following content:

apiVersion: v1

kind: Service

metadata:

name: nodeport-service

spec:

selector:

app: my-app

ports:

- protocol: TCP

port: 80

targetPort: 80

nodePort: 30007 # A specific port in the range 30000-32767

type: NodePort

**Explanation:**

* The primary difference from the ClusterIP Service is the addition of nodePort, which specifies the static port on each Node.
* type: Set to NodePort, exposing the Service on a specific port across all Nodes.

**Apply this YAML to create the NodePort Service:**

kubectl apply -f nodeport-service.yaml

**Verify the Service:**

kubectl get services

You should see the nodeport-service listed with a NodePort and details about the port exposed.

**3. LoadBalancer Service**

If you're using a cloud provider that supports LoadBalancers, you can create a LoadBalancer Service to expose the Service externally.

**Create a YAML file named loadbalancer-service.yaml with the following content:**

apiVersion: v1

kind: Service

metadata:

name: loadbalancer-service

spec:

selector:

app: my-app

ports:

- protocol: TCP

port: 80

targetPort: 80

type: LoadBalancer

**Explanation:**

* type: Set to LoadBalancer, which requests the cloud provider to provision a load balancer for this Service.

**Apply this YAML to create the LoadBalancer Service:**

kubectl apply -f loadbalancer-service.yaml

**Verify the Service:**

kubectl get services

Once the LoadBalancer is provisioned, the external IP address will be listed under the EXTERNAL-IP column.

**Step 3: Cleaning Up Resources**

To delete the resources you created:

kubectl delete -f clusterip-service.yaml

kubectl delete -f nodeport-service.yaml

kubectl delete -f loadbalancer-service.yaml

This command deletes all the Services you defined.