**Types of Kubernetes Services**

1. **ClusterIP (Default)** – Exposes the service internally within the cluster.
2. **NodePort** – Exposes the service externally on each node’s IP at a static port.
3. **LoadBalancer** – Exposes the service externally using a cloud provider’s load balancer.
4. **ExternalName** – Maps a Kubernetes service to an external DNS name.

**1. ClusterIP Service (Default)**

A **ClusterIP** service allows communication between different pods within the cluster. It is the default type of service in Kubernetes.

**Example**

Create a simple Nginx deployment and expose it using a ClusterIP service.

**Step 1: Create a Deployment**

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:latest

ports:

- containerPort: 80

**Apply the deployment:**

kubectl apply -f nginx-deployment.yaml

**Step 2: Create a ClusterIP Service**

apiVersion: v1

kind: Service

metadata:

name: nginx-clusterip

spec:

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

type: ClusterIP

**Apply the service:**

kubectl apply -f nginx-clusterip.yaml

**Explanation:**

* This service selects pods with the label app: nginx.
* It forwards traffic from port 80 on the service to port 80 on the container.
* The service is only accessible **within the cluster**.

**Check the Service**

kubectl get services

You will see output similar to:

NAME TYPE CLUSTER-IP PORT(S) AGE

nginx-clusterip ClusterIP 10.96.0.1 80/TCP 10s

To access the service inside the cluster:

kubectl exec -it <pod-name> -- curl http://nginx-clusterip

**2. NodePort Service**

A **NodePort** service makes a service accessible externally using each node's IP address and a static port.

**Example**

Modify the previous service to use NodePort:

apiVersion: v1

kind: Service

metadata:

name: nginx-nodeport

spec:

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

nodePort: 30007 # Optional, otherwise auto-allocated (30000-32767)

type: NodePort

Apply the service:

kubectl apply -f nginx-nodeport.yaml

**Explanation:**

* Exposes the service on **every node’s IP** at port 30007.
* Users can access the service using <Node-IP>:30007.

**Check the Service**

kubectl get services

Output:

pgsql

CopyEdit

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

nginx-nodeport NodePort 10.96.0.2 <none> 80:30007/TCP 5s

**Access the Service**

Find the Node's external IP:

kubectl get nodes -o wide

Then access it:

curl http://<Node-IP>:30007

If running on minikube open the tunnel:

minikube service <service-name> --url

This will provide an url, which can be used to access the service from outside the Cluster

**3. LoadBalancer Service**

A **LoadBalancer** service provisions a cloud provider’s external load balancer to route traffic.

**Example**

Modify the service type to LoadBalancer:

apiVersion: v1

kind: Service

metadata:

name: nginx-loadbalancer

spec:

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

type: LoadBalancer

Apply the service:

kubectl apply -f nginx-loadbalancer.yaml

**Explanation:**

* Provisions an external **Load Balancer** in cloud environments (AWS, GCP, Azure).
* External clients can access the service using the external IP assigned by the cloud provider.

**Check the External IP**

kubectl get services

Expected output:

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

nginx-loadbalancer LoadBalancer 10.96.0.3 pending 80:32345/TCP 10s

If running on minikube open the tunnel:

minikube tunnel

Expected output:

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

nginx-loadbalancer LoadBalancer 10.96.0.3 34.120.10.5 80:32345/TCP 10s

**Access the Service**

curl http://34.120.10.5

**4. ExternalName Service**

An **ExternalName** service maps a Kubernetes service to an external domain.

**Example**

apiVersion: v1

kind: Service

metadata:

name: external-service

spec:

type: ExternalName

externalName: example.com

Apply the service:

kubectl apply -f external-service.yaml

**Explanation:**

* Redirects traffic to example.com.
* It does not create an actual endpoint but acts as a DNS alias.

**Check the Service**

kubectl get services

Output:

NAME TYPE CLUSTER-IP EXTERNAL-NAME PORT(S) AGE

external-service ExternalName <none> example.com <none> 10s

**Access the External Service**

Inside a pod:

kubectl run curlpod --image=busybox --restart=Never -- curl external-service

**Summary of Service Types**

| **Service Type** | **Accessible From** | **Exposes External IP** | **Use Case** |
| --- | --- | --- | --- |
| ClusterIP (Default) | Internal Cluster Only | No | Internal microservices communication |
| NodePort | Any node’s IP & Port | Yes | External access without Load Balancer |
| LoadBalancer | External Clients | Yes | Cloud-managed Load Balancer |
| ExternalName | Internal Cluster Only | No | Redirecting services to external DNS |

**Useful kubectl Commands**

1. **Get all services**

kubectl get services

1. **Describe a service**

kubectl describe service <service-name>

1. **Delete a service**

kubectl delete service <service-name>

1. **Port forward a service**

kubectl port-forward svc/nginx-clusterip 8080:80

1. **Check logs of a pod**

kubectl logs <pod-name>

1. **Execute a command inside a pod**

kubectl exec -it <pod-name> -- /bin/sh