What is service in Kubernetes and why it is needed ?

What is a Service in Kubernetes?

- A **Service** in Kubernetes provides a stable network endpoint for accessing **Pods**.
- **Pods** are ephemeral, with dynamically changing IP addresses.
- A **Service** ensures consistent access using a **single, unchanging IP address (ClusterIP)**.
- It also provides a **DNS name** for reliable communication.

Why Do We Need a Service in Kubernetes?

In Kubernetes, a **Pod** is the smallest deployable unit, but Pods are not reliable in terms of networking because:

1. **Pods are ephemeral** – They can be restarted, rescheduled, or scaled dynamically, leading to changing IP addresses.
2. **Pods cannot be directly exposed to external clients** – Kubernetes networking does not automatically provide a way for external clients or other Pods to discover them.
3. **Load balancing is needed** – If multiple replicas of a Pod are running, traffic needs to be distributed among them efficiently.

A **Service** solves these problems by:

- Providing a **fixed IP address and DNS name** for a group of Pods.
- Acting as a **load balancer** to distribute traffic among multiple Pod replicas.
- Enabling **communication between microservices** in a Kubernetes cluster.
- Facilitating **external access** (if needed) using different types of Services.

Types of Services in Kubernetes

1. **ClusterIP (Default)** – Exposes the Service only inside the cluster.
2. **NodePort** – Exposes the Service on a static port of each worker node.
3. **LoadBalancer** – Uses an external load balancer (e.g., AWS ELB, GCP LB).
4. **ExternalName** – Maps a Service to an external domain name.

Real-World Use Case of Kubernetes Service

Let's say you are running an **E-commerce Application** on Kubernetes. Your application has multiple microservices like:

1. **Frontend Service** (React, Angular, Vue.js)
2. **Backend Service** (Node.js, Spring Boot, Django, etc.)
3. **Database Service** (MySQL, PostgreSQL, MongoDB)

Each of these runs in separate **Pods**. Since Pods are dynamic, we need **Services** to ensure stable communication.

Use Case: Backend Service for an E-commerce App

Scenario

Your **frontend app** needs to call the **backend API** to get product details, but the backend Pods' IPs keep changing due to scaling and restarts.

A **Kubernetes Service** ensures that the frontend can always reach the backend without worrying about Pod changes.

]

Benefits of Using a Service Here

✅ **Stable Communication** – Frontend always reaches backend, even if Pods restart.
✅ **Load Balancing** – Service distributes requests among all backend Pods.
✅ **Scalability** – If backend scales up (e.g., from 3 to 5 Pods), the Service automatically includes new Pods.
✅ **Security** – The backend Service is `ClusterIP`, so it's accessible only within Kubernetes (not exposed publicly).

Summary

- Without a Service: Frontend would fail when backend Pods restart.
- With a Service: Frontend always connects, load balancing is automatic.
- Real-world case: **E-commerce, FinTech, SaaS apps where microservices interact.**

Steps to Run the Demo

1. Create a Deployment for Nginx

Save the following YAML as `nginx-deployment.yaml`:

```
yaml
```

```
---------
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:

```
sh
---------
kubectl apply -f nginx-deployment.yaml
```

2. Create a Service to Expose Nginx Pods

Save the following YAML as `nginx-service.yaml`:

```
yaml
---------
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
```

Apply the service:

```
sh
---------
kubectl apply -f nginx-service.yaml
```

3. Check the Service and Get the External IP

```
sh
---------
kubectl get services
```

If running on **Minikube**, expose the service manually:

```
sh
---------
minikube service nginx-service
```

4. Access the Nginx Service

- Open the **EXTERNAL-IP** in your browser (for cloud clusters).
- If using Minikube, the `minikube service` command will open it for you.

How does Kubernetes ensure high availability when using a Service?

A) By running multiple service replicas
B) By automatically creating multiple Pods
C) By load balancing traffic across multiple Pods
D) By using a persistent volume for service storage

✅ **Answer:** C) By load balancing traffic across multiple Pods

Which type of Kubernetes Service allows external traffic to reach cluster Pods through a port on each Node?

A) ClusterIP
B) NodePort
C) LoadBalancer
D) ExternalName

✅ **Answer:** B) NodePort

Which component of Kubernetes assigns an IP address to each Service?

A) Kubelet
B) Kube-proxy
C) Etcd
D) Scheduler

✅ **Answer:** B) Kube-proxy