Kubernetes - Service

A service can be defined as a logical set of pods. It can be defined as an abstraction on the top of the pod which provides a single IP address and DNS name by which pods can be accessed. With Service, it is very easy to manage load balancing configuration. It helps pods to scale very easily.

A service is a REST object in Kubernetes whose definition can be posted to Kubernetes apiServer on the Kubernetes master to create a new instance.

Service without Selector

```
apiVersion: v1
kind: Service
metadata:
   name: Tutorial_point_service
spec:
   ports:
   - port: 8080
   targetPort: 31999
```

The above configuration will create a service with the name Tutorial_point_service.

Service Config File with Selector

```
apiVersion: v1
kind: Service
metadata:
   name: Tutorial_point_service
spec:
   selector:
      application: "My Application" ------------------> (Selector)
   ports:
   - port: 8080
   targetPort: 31999
```

In this example, we have a selector; so in order to transfer traffic, we need to create an endpoint manually.

```
apiVersion: v1
kind: Endpoints
metadata:
   name: Tutorial_point_service
subnets:
   address:
      "ip": "192.168.168.40" ------------------> (Selector)
   ports:
      - port: 8080
```

In the above code, we have created an endpoint which will route the traffic to the endpoint defined as “192.168.168.40:8080”.

Multi-Port Service Creation

```
apiVersion: v1
kind: Service
metadata:
   name: Tutorial_point_service
spec:
   selector:
      application: “My Application” ------------------> (Selector)
   ClusterIP: 10.3.0.12
   ports:
      -name: http
      protocol: TCP
      port: 80
      targetPort: 31999
   -name:https
      Protocol: TCP
      Port: 443
      targetPort: 31998
```

Types of Services

ClusterIP − This helps in restricting the service within the cluster. It exposes the service within the defined Kubernetes cluster.

```
spec:
   type: NodePort
   ports:
   - port: 8080
      nodePort: 31999
      name: NodeportService
```

NodePort − It will expose the service on a static port on the deployed node. A **ClusterIP** service, to which **NodePort** service will route, is automatically created. The service can be accessed from outside the cluster using the **NodeIP:nodePort**.

```
spec:
   ports:
   - port: 8080
      nodePort: 31999
      name: NodeportService
      clusterIP: 10.20.30.40
```

Load Balancer − It uses cloud providers’ load balancer. **NodePort** and **ClusterIP** services are created automatically to which the external load balancer will route.

A full service **yaml** file with service type as Node Port. Try to create one yourself.

```
apiVersion: v1
kind: Service
metadata:
   name: appname
   labels:
      k8s-app: appname
spec:
```

```
type: NodePort
ports:
- port: 8080
   nodePort: 31999
   name: omninginx
selector:
   k8s-app: appname
   component: nginx
   env: env_name
```