Kubernetes Service Objects and Namespaces

Service Objects

A service object in Kubernetes is an API that works on labels and selectors to route traffic. It sends traffic specifically to the selector labels defined in the manifest file.

There are three main types of Kubernetes service objects:

1. **LoadBalancer Service**
   - Used for **external access** to applications.
   - A manifest file is written to create a LoadBalancer service. Once applied, the LoadBalancer is provisioned in the cloud (e.g., AWS Console).
2. **ClusterIP Service**
   - Used for **internal communication** within the cluster.
   - Commonly used for front-end, back-end, and database communication.
   - Traffic is routed within the cluster via ClusterIP.
3. **NodePort Service**
   - Exposes specific pods to be accessed from a range of ports on the node.

LoadBalancer Service Example

To create a LoadBalancer service, write a YAML manifest file, apply it, and access the service for external traffic.

Architectures

1. **Monolithic Architecture**
   - Combines the entire application into a single JAR or WAR file with all binaries in one place.
2. **Microservices Architecture**
   - Divides the application into multiple services, each with its own binaries.
   - This results in multiple Docker images and services for independent deployment and scalability.

Namespaces

Namespaces are used to divide the cluster **virtually** and **logically** for better isolation and management, such as separating **Development**, **Testing**, and **Production** teams.

Default Kubernetes Namespaces

1. `default`
2. `kube-node-lease`
3. `kube-public`
4. `kube-system`

Commands to Manage Namespaces

Check existing namespaces:
bash
Copy code
```
kubectl get ns
```

-

Create a custom namespace:
bash
Copy code
```
kubectl create ns <namespace_name>
```

-

Namespaces are a powerful tool to isolate workloads and manage resources effectively within Kubernetes.

—---

Kubernetes Concepts: Service Objects, Load Balancers, and Namespaces

Service Object Overview

A **Service Object** in Kubernetes allows applications to communicate with each other and expose them to external or internal networks. It works based on **labels** and **selectors**.

- **Service Object Functionality**: Sends traffic to pods matching the defined selector labels.
- Types of Load Balancer Services:
  1. **LoadBalancer Service**: For external access, traffic is routed via a cloud provider's load balancer.
  2. **ClusterIP Service**: Provides internal communication within the cluster.
  3. **NodePort Service**: Exposes a pod on a specific port on all cluster nodes.

Service Manifest Example

LoadBalancer Service (External Access):

yaml
Copy code
```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-loadbalancer
spec:
  type: LoadBalancer
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
```

ClusterIP Service (Internal Communication):

yaml
Copy code
```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-clusterip
spec:
  type: ClusterIP
  selector:
    app: backend
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
```

NodePort Service (Specific Port Access):

yaml
Copy code
```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-nodeport
spec:
  type: NodePort
  selector:
    app: database
  ports:
    - protocol: TCP
      port: 3000
      targetPort: 3306
      nodePort: 32000
```

After applying these manifests, you can check the services and their external endpoints using:

bash
Copy code
```bash
kubectl get svc
```

ClusterIP Use Cases

- Used in **Monolithic Architecture** to route traffic to a single binary file (e.g., WAR or JAR).
- Used in **Microservices Architecture** to route traffic between multiple services and binaries (frontend, backend, database).

Namespaces

Namespaces logically divide the Kubernetes cluster to isolate resources for different teams (e.g., Development, Testing, Production).

- **Default Namespaces**:
  - `default`
  - `kube-node-lease`
  - `kube-public`
  - `kube-system`
- **Commands**:

View existing namespaces:
bash

Copy code

```
kubectl get ns
```

○

Create a new namespace:
bash
Copy code

```
kubectl create ns <namespace-name>
```

○

Example Namespace Manifest:

yaml
Copy code

```
apiVersion: v1
kind: Namespace
metadata:
  name: development
```

Namespaces help isolate resources and manage cluster access for teams effectively.