Creating Services by Exposing Ports

In this lesson, we will explore how to create Kubernetes Services by exposing ports.

WE'LL COVER THE FOLLOWING

Creating ReplicaSets

Before we dive into services, we should create a ReplicaSet similar to the one we used in the previous chapter. It'll provide the Pods we can use to demonstrate how Services work.

Let's take a quick look at the ReplicaSet definition.

```
cat svc/go-demo-2-rs.yml
```

The only significant difference is the `db` container definition. It is as follows.

```
...
- name: db
  image: mongo:3.3
  command: ["mongod"]
  args: ["--rest", "--httpinterface"]
  ports:
  - containerPort: 28017
    protocol: TCP
...
```

the REST interface. We also defined the `containerPort`. Those additions are needed so that we can test that the database is accessible through the Service.

Let’s create the ReplicaSet.

```
kubectl create -f svc/go-demo-2-rs.yml
kubectl get -f svc/go-demo-2-rs.yml
```

We created the ReplicaSet and retrieved its state from Kubernetes. The **output** is as follows.

```
NAME      DESIRED CURRENT READY AGE
go-demo-2 2       2       2     1m
```

You might need to wait until both replicas are up-and-running. If, in your case, the `READY` column does not yet have the value `2`, please wait for a while and `get` the state again. We can proceed after both replicas are running.

Exposing a Resource

We can use the `kubectl expose` command to expose a resource as a new Kubernetes Service. That resource can be a Deployment, another Service, a ReplicaSet, a ReplicationController, or a Pod. We’ll expose the ReplicaSet since it is already running in the cluster.

```
kubectl expose rs go-demo-2 \
    --name=go-demo-2-svc \
    --target-port=28017 \
    --type=NodePort
```

- **Line 1:** We specified that we want to expose a ReplicaSet (`rs`).
- **Line 2:** The name of the new Service should be `go-demo-2-svc`.
- **Line 3:** The port that should be exposed is `28017` (the port MongoDB interface is listening to).
- **Line 4:** we specified that the type of the Service should be `NodePort`.

As a result, the target port will be exposed on every node of the cluster to the outside world, and it will be routed to one of the Pods controlled by the ReplicaSet.

Other Types of Services

There are other Service types we could have used to establish communication:

ClusterIP

`ClusterIP` (the default type) exposes the port only inside the cluster. Such a port would not be accessible from anywhere outside. `ClusterIP` is useful when we want to enable communication between Pods and still prevent any external access.

> If `NodePort` is used, `ClusterIP` will be created automatically.

LoadBalancer

The `LoadBalancer` type is only useful when combined with cloud provider's load balancer.

ExternalName

`ExternalName` maps a service to an external address (e.g., `kubernetes.io`).

In this chapter, we'll focus on `NodePort` and `ClusterIP` types. `LoadBalancer` will have to wait until we move our cluster to one of the cloud providers and `ExternalName` has a very limited usage.

In the next lesson, we will go through the sequential breakdown of the process of Service creation.