13

Mãos à obra: Criando o objeto Service

Como vimos nas etapas anteriores, os Pods estão em constante alteração, por isso que para acessar o Pod que irá conter o banco de dados, precisamos abstraí-lo em um objeto mais estável do Kubernetes que é o objeto **Service**. Crie um novo arquivo chamado **servico-banco.yaml**

Vamos colocar no início desse arquivo, a versão da API do Kubernetes responsável pela esquematização do objeto **Service** e também um nome de identificação para esse objeto. Aqui precisamos ter cuidado, nossa aplicação web irá procurar por esse objeto **Service** para acessar o banco. Devemos manter o mesmo nome do que foi configurado na aplicação.

```
apiVersion: v1
kind: Service
metadata:
  name: db
```

Nossa aplicação web irá procurar esse objeto **Service** para cadastrar os produtos no banco. Tudo isso está rodando no nosso Cluster, dessa forma, diferentemente do outro serviço que configuramos para acessarmos a aplicação web, nós podemos configurar esse serviço para que seja somente acessado internamente no Cluster. Não podemos nos esquecer que o MySQL utiliza a porta de comunicação **3306** e que temos que estabelecer o vínculo desse objeto **Service** com o **Pod** do banco de dados abstraído pelo objeto StatefulSet.

```
spec:
  type: ClusterIP
  ports:
    - port: 3306
  selector:
    name: mysql
```

Uma vez que estamos trabalhando com o objeto **StatefulSet** devemos estabelecer esse vínculo com o objeto **Service** bidirecionalmente, dessa forma, volte para o arquivo **statefulset.yaml** e logo abaixo da **primeira** chave spec coloque:

```
  serviceName: db
```

Feito isso, nossa configuração do banco está finalizada. Devemos agora fazer a implementação no Cluster, para isso vá até o local onde os arquivos estão salvos e vamos criar tais objetos no nosso Cluster. Vamos começar pelo objeto **StatefulSet** que está realizando a abstração do objeto Pod com o container do banco de dados, colocamos no terminal:

```
kubectl create -f statefulset.yaml
```

Na sequência, vamos passar o arquivo da configuração do objeto **Service** que irá abstrair o acesso a esse objeto **Pod**

```
kubectl create -f servico-banco.yaml
```

Por fim, iremos passar as configurações de requisições que configuramos no arquivo **permissoes.yaml**

```
kubectl create -f permissoes.yaml
```

Com isso, já temos todas as configurações necessárias, falta somente criarmos as tabelas no banco de dados. Para isso, temos que ver inicialmente qual é o nome do **Pod** que está abstraindo o container do banco de dados:

```
kubectl get pods
```

Dentre as opções listadas nós devemos ter uma informando **statefulset-mysql**, esse é o **Pod** que está relizando a abstração do container com o banco de dados. Para podermos acessar esse Pod e criarmos as tabelas digite:

```
kubectl exec -it [nome do Pod] sh
```

Uma vez que obtivemos o **shell** vamos acessar o MySQL com o usuário root:

```
mysql -u root
```

Toda a informação da nossa aplicação está rodando no banco **loja**, então devemos alterar para esse banco e criar as tabelas:

```
use loja
```

Por fim, iremos criar as tabelas com o comando:

```
create table produtos (id integer auto_increment primary key, nome varchar(255), preco decimal(
alter table produtos add column usado boolean default false;
alter table produtos add column descricao varchar(255);
create table categorias (id integer auto_increment primary key, nome varchar(255));
insert into categorias (nome) values ("Futebol"), ("Volei"), ("Tenis");
alter table produtos add column categoria_id integer;
update produtos set categoria_id = 1;
```

Agora que nós já criamos as tabelas, vamos testar nossa aplicação. Para isso, precisaremos do endereço IP que foi atribuído para o objeto **Service** que realiza o acesso a aplicação web, conseguimos obter isso no minikube com o comando:

```
minikube service servico-aplicacao --url
```

Tente acessar a aplicação e cadastrar alguns produtos. O que acontece se por exemplo você deletar um Pod, o Kubernetes conseguirá fazer esse gerenciamento?