07

Configurando volumes

Transcrição

Feita a configuração do objeto `StatefulSet`, ainda nos deparamos com a possibilidade de o `Pod` parar de funcionar e perdermos todas as informações cadastradas e armazenadas no banco de dados. Queremos garantir que isto não aconteça.

Vamos, então, configurar um volume para o mapeamento entre o container do MySQL e este volume externo que estaremos criando, tal como fizemos no `docker-compose.yaml`.

O arquivo `statefulset.yaml` ficará assim:

```
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: statefulset-mysql
spec:
  template:
    metadata:
      labels:
       name: mysql
    spec:
      containers:
        - name: container-mysql
          image: mysql:5.7.19
          ports:
            - containerPort: 3306
          env:
            - name: MYSQL_DATABASE
              value: "loja"
            - name: MYSQL_USER
              value: "root"
            - name: MYSQL_ALLOW_EMPTY_PASSWORD
              value: "1"
          volumeMounts:
            - name: volume-mysql
              mountPath: /var/lib/mysql
      volumes:
          - name: volume-mysql
```

Ou seja, criamos `volume-mysql`, e com `volumeMounts` estamos fazendo esta montagem em nosso container, no diretório `/var/lib/mysql`. Este volume pode estar sendo acessado por outros recursos em nosso *cluster*, portanto é preciso configurarmos permissões deste `Pod`, para o acesso de determinada quantidade de recursos.

Criaremos um novo arquivo YAML mais uma vez, que abriremos com "Ctrl + N" no Atom. Salvaremos ele no diretório "db" e o nomearemos de "permissoes.yaml".

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
```

```
    name: configuracao-mysql
  spec:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: 3Gi
```

Com isto, `accessModes` com `ReadWriteOnce` indica que apenas uma máquina, aquela que for receber nosso `Pod` do MySQL no *cluster*, terá permissão de acesso. E `requests` significa que estamos requisitando uma quantidade de armazenamento de `3Gi` , isto é, `3` gb do volume persistente.

Em `statefulset.yaml` , temos que `volume-mysql` será acessado por `Pod` com as devidas permissões. Acrescentaremos que o arquivo que configuramos com as permissões para acessar `volume-mysql` estão com nome `configuracao-mysql` :

```
volumes: // já existente anteriormente
  - name: volume-mysql // já existente anteriormente
    persistentVolumeClaim:
      claimName: configuracao-mysql
```

Recapitulando: fizemos o mapeamento de volumes, com `Pod` precisando de uma permissão de acesso de uma determinada quantidade de recursos no `volume-mysql` recém criado, indicada no arquivo `persistentVolumeClaim` em `permissoes.yaml` .

Somente a máquina que for receber este `Pod` do banco de dados é que terá a permissão de leitura e escrita. E, deste volume que configuramos, estamos requisitando um espaço de `3` gb.

Além disto, é preciso indicar que `volumes` será acessado pelo objeto `Pod` a partir da informação de qual arquivo contém tais permissões.

Legal! Falta estabelecermos a comunicação da nossa aplicação web com o banco de dados que acabamos de configurar. Como já foi dito, o `Pod` faz parte de um mundo instável, sendo necessário abstrairmos seu acesso em um objeto mais estável, de serviço. Vamos configurá-lo?