LABORATÓRIO PRÁTICO: SHARDING NO MONGODB
DEMONSTRAÇÃO DE SHARDING NO MONGODB USANDO CONTAINERS

Requisitos:

- Docker e mongosh instalados.
- Recomendo que crie uma instância t3.small.
- Portas 27017-27020 liberadas no grupo de segurança da instância.
- Para não precisar de sudo nos comandos docker: sudo usermod -aG docker $USER. Depois, dê newgrp docker.

OBS: Em ambiente de produção, cada cluster precisa de vários nós “replicaset” para garantir disponibilidade. Mas aqui, para demonstração, é usado apenas um nó para cada função.

ARQUITETURA

1. Todos os containers vão falar entre si pela rede interna:

```
docker network create net-cluster
```

2. Conectado na instância via SSH, criaremos o servidor de configuração. Eles armazenam informações sobre como os dados são particionados e qual shard contém qual parte dos dados. Execute isso no terminal:

```
docker run --name mongo-config --net net-cluster -d mongo mongod --configsvr --replSet configserver --port 27018
```

É necessário informar a porta, nessa demonstração não iremos usar IP. A comunicação acontece através da network criada. Também passamos outros parâmetros, como id e tipo.

3. Após criar, devemos configurá-lo. Acesse com o comando:

```
docker exec -it mongo-config mongosh --port 27018
```

Esse comando irá abrir uma shell, nele você irá colocar e dar enter:

```
rs.initiate({
  _id: "configserver",
  configsvr: true,
  members: [
    { _id: 0, host: "mongo-config:27018" }
  ]
})
```

Se concluído com sucesso, digite exit para sair.

4. Criaremos também o primeiro e o segundo shard, cada shard armazena uma porção dos dados:

```
docker run --name node1 --net net-cluster -d mongo mongod --port 27019 --shardsvr --replSet shard1
```

```
docker run --name node2 --net net-cluster -d mongo mongod --port 27020 --shardsvr
--replSet shard2
```

5. Também é necessário configurá-los. Começando pelo primeiro shard:

```
docker exec -it node1 mongosh --port 27019
```

```
rs.initiate({
  _id: "shard1",
  version: 1,
  members: [
    { _id: 0, host : "node1:27019" },
  ]
})
```

Se concluído com sucesso, digite exit para sair.

6. Faça a mesma coisa para o segundo shard:

```
docker exec -it node2 mongosh --port 27020
```

```
rs.initiate({
   _id: "shard2",
   version: 1,
   members: [
     { _id: 0, host : "node2:27020" }
   ]
})
```

Se concluído com sucesso, digite exit para sair.

7. Criaremos o último servidor, o roteador mongo (mongos). Ele é o ponto de entrada entre o cliente e os shards, direciona consultas e gravações:

```
docker run -p 27017:27017 --name mongo-router --net net-cluster -d mongo mongos --port
27017 --configdb configserver/mongo-config:27018 --bind_ip_all
```

8. Agora, registre os shards no mongos. Mesmo processo, se conecte:

```
docker exec -it mongo-router mongosh
```

Adicione o primeiro shard:

```
sh.addShard("shard1/node1:27019")
```

Adicione o segundo shard:

```
sh.addShard("shard2/node2:27020")
```

9. Com a arquitetura montada, será montado uma base de dados e uma coleção, É nessa fase que você define a estratégia de fragmentação.

Crie o banco de dados e habilite o shard:

```
use exemploDB
sh.enableSharding("exemploDB")
```

Aqui você cria uma coleção e define o tipo de fragmentação hash, que divide uniformemente os dados:

```
sh.shardCollection("exemploDB.persons", { idade: "hashed" })
```

TESTANDO

10. Adicione dados e observe o id alfanumérico sendo criado:

```
db.persons.insert ( { nome : "Zeca" , idade : 11 })
```

Dê o próximo comando para vizualizar a distribuição dos dados. O documento foi armazenado em um dos shards:

```
db.persons.getShardDistribution()
```

Adicione mais alguns e dê o comando anterior novamente para observar a nova distribuição, observe que a ditribuição ficou equilibrada. Isso acontece justamente pelo uso do Hash:

```
db.persons.insertMany([
  { nome: "Harsh", idade: 20 },
  { nome: "Ana", idade: 22 },
  { nome: "Bruno", idade: 31 },
  { nome: "Carla", idade: 27 },
  { nome: "Diego", idade: 19 },
  { nome: "Elisa", idade: 45 },
  { nome: "Fernando", idade: 33 },
  { nome: "Gabriela", idade: 29 },
  { nome: "Hugo", idade: 41 }
])
```

11. Teste requisições também. Dê o comando:

```
db.persons.find().sort({ idade: 1 })
```

Nesse comando ele busca os documentos e os ordenam de forma crescente.