Tolerância a Falhas e Distribuição MongoDB

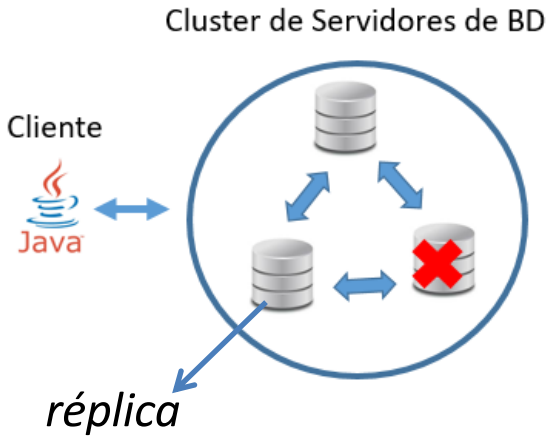
Pedro Ramos, Catarina Ferreira da Silva
ISCTE-IUL, 2020/2021

Armazenamento de Dados em Ambientes distribuídos (ADAD)

Mestrado em Engenharia de Telecomunicações e Informática (METI)

Tolerância a falhas - Mongo DB

A tolerância a falhas é assegurada através da replicação de dados de uma forma controlada

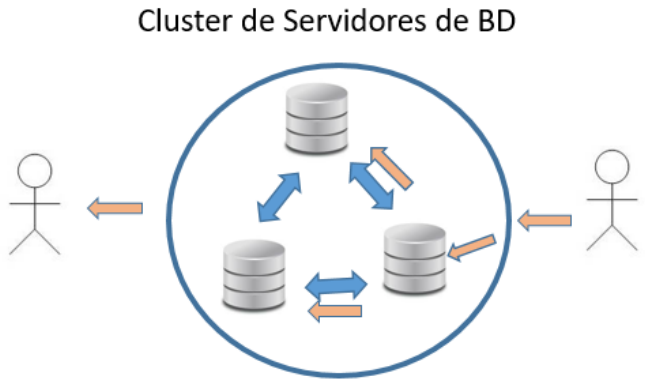


- O cliente liga-se a um conjunto (*cluster*) de bases de dados que se sincronizam entre si de modo a garantir que a informação é tendencialmente a mesma em todas as bases de dados
- Se um servidor deixar de estar acessível os outros continuam a comunicar com o cliente

Quando o servidor volta a estar acessível tem que existir um mecanismo automático que o "actualize" (sincronize) com os dados entretanto modificados ou inseridos (caso contrário, não existiria uma tolerância a falhas mas apenas "disponibilidade")

Tolerância a falhas - Mongo DB

Fluxo de dados num sistema com bases de dados replicadas



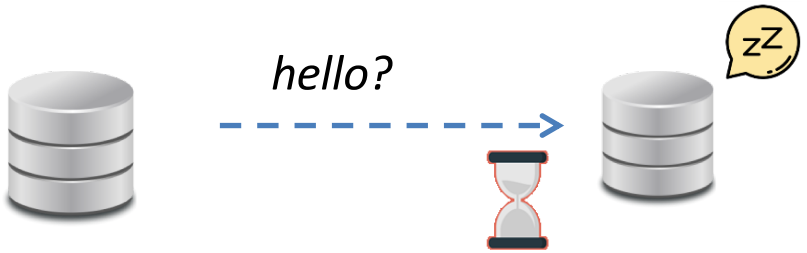
1) Uma aplicação cliente introduz dados (seta alaranjada) num cluster sem indicar explicitamente qual o servidor destino

2) O sistema escolhe o servidor e este, depois de receber a informação, replica-a pelos restantes nós (servidores)

- Quando outro cliente efectua uma consulta, o cluster escolhe qual o nó/servidor que processa o pedido
- Se um nó estiver ausente, o cluster procura outro nó para satisfazer o pedido
- Cada servidor pode estar num computador diferente

Tolerância a falhas - Mongo DB

Outros factores a ter em conta neste tipo de arquitectura



(i) definição do tempo de "silêncio" de um servidor a partir do qual se considera que ele está incontactável

Heartbeats

- De dois em dois segundos cada servidor envia um **ping** (*heartbeat*) aos parceiros para actualizar a informação (não apenas para saber se algum desapareceu, mas para saber se ocorre um processo de eleição, se ocorreu alguma troca, etc.)

- Se passados 10 segundos (valor por omissão que pode ser alterado) não obtém resposta o servidor decide que é necessário um processo de eleição

Tolerância a falhas - Mongo DB

Outros factores a ter em conta neste tipo de arquitectura

- Outro parâmetro que pode ser alterado (2 segundos por omissão): **tempo** limite **para** uma recém-eleita réplica primária **sincronizar** com as restantes réplicas que podem ter dados mais actualizados

- Um tempo mais elevado pode fazer perder menos informação, mas sobrecarrega o sistema. Durante esse período a nova réplica não aceita escrita e leituras de clientes

Tolerância a falhas - Mongo DB

Réplicas em MongoDB

i) O número de réplicas tem de ser ímpar
ii) Apenas é possível escrever em uma das réplicas

- Todas as réplicas têm a mesma informação depois de terminada a sincronização (existem excepções, por exemplo, no caso de serem designados “árbitros”)

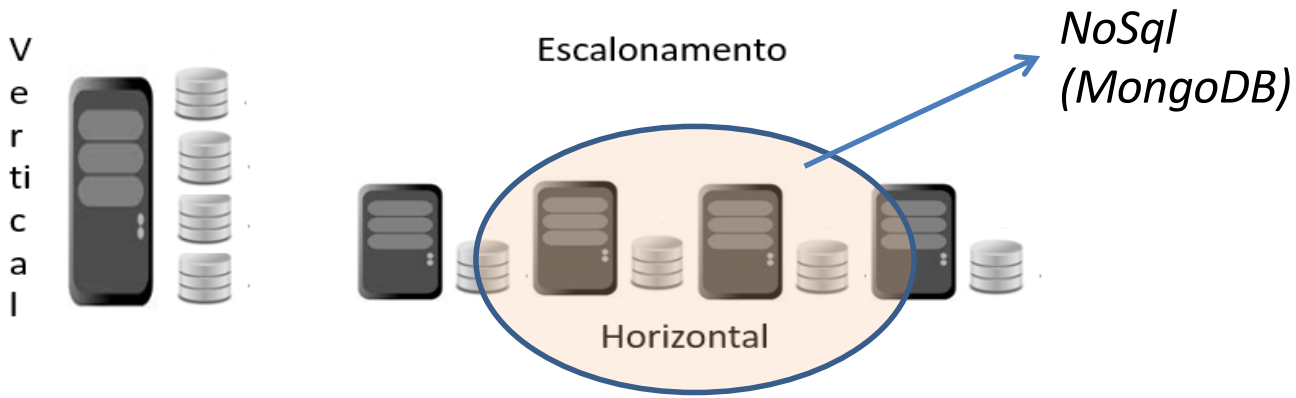
- A réplica onde as **operações de escrita** são permitidas é denominada **Primária**, e as restantes são denominadas **Secundárias**

- Apenas é possível ler dados das Secundárias, caso seja dada permissão explícita (por omissão a permissão não é dada)

- Um cluster com 5 nós é tolerante a 2 falhas (dois servidores desligados) e um com 3 nós é apenas tolerante a uma falha

Armazenamento Distribuído - Mongo DB

Duas estratégias para lidar com escalonamento



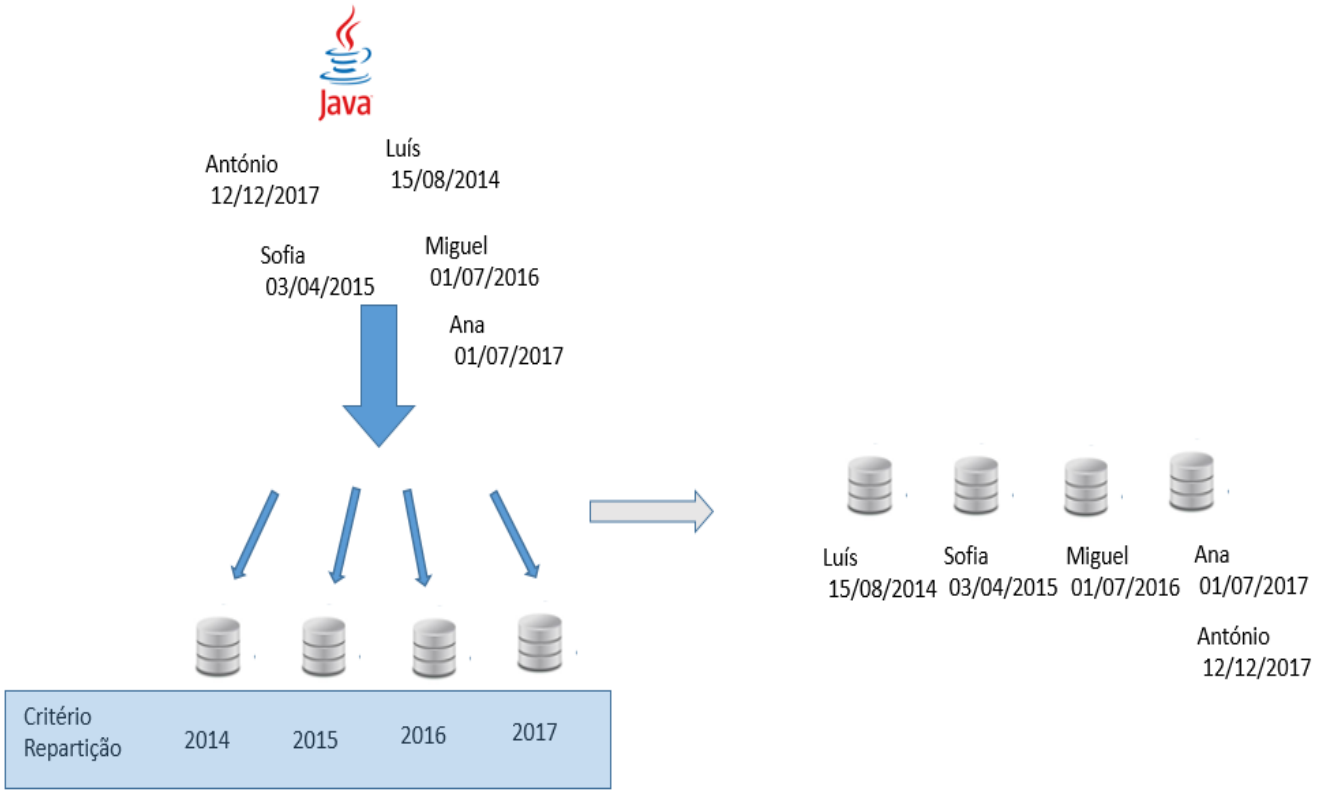
A dispersão da informação por várias bases de dados dificulta a sua manutenção, quer em termos de operações de escrita, quer nas consultas de dados

A perda de eficiência (por oposição à solução de escalonamento vertical) deve-se a

(i) Antes de aceder aos dados, ter de "perguntar" em que bases de dados eles estão armazenados
(ii) "Custo físico" do transporte da informação
(iii) Manutenção do registo de localização da informação

Armazenamento Distribuído - Mongo DB

Exemplo de escalonamento horizontal

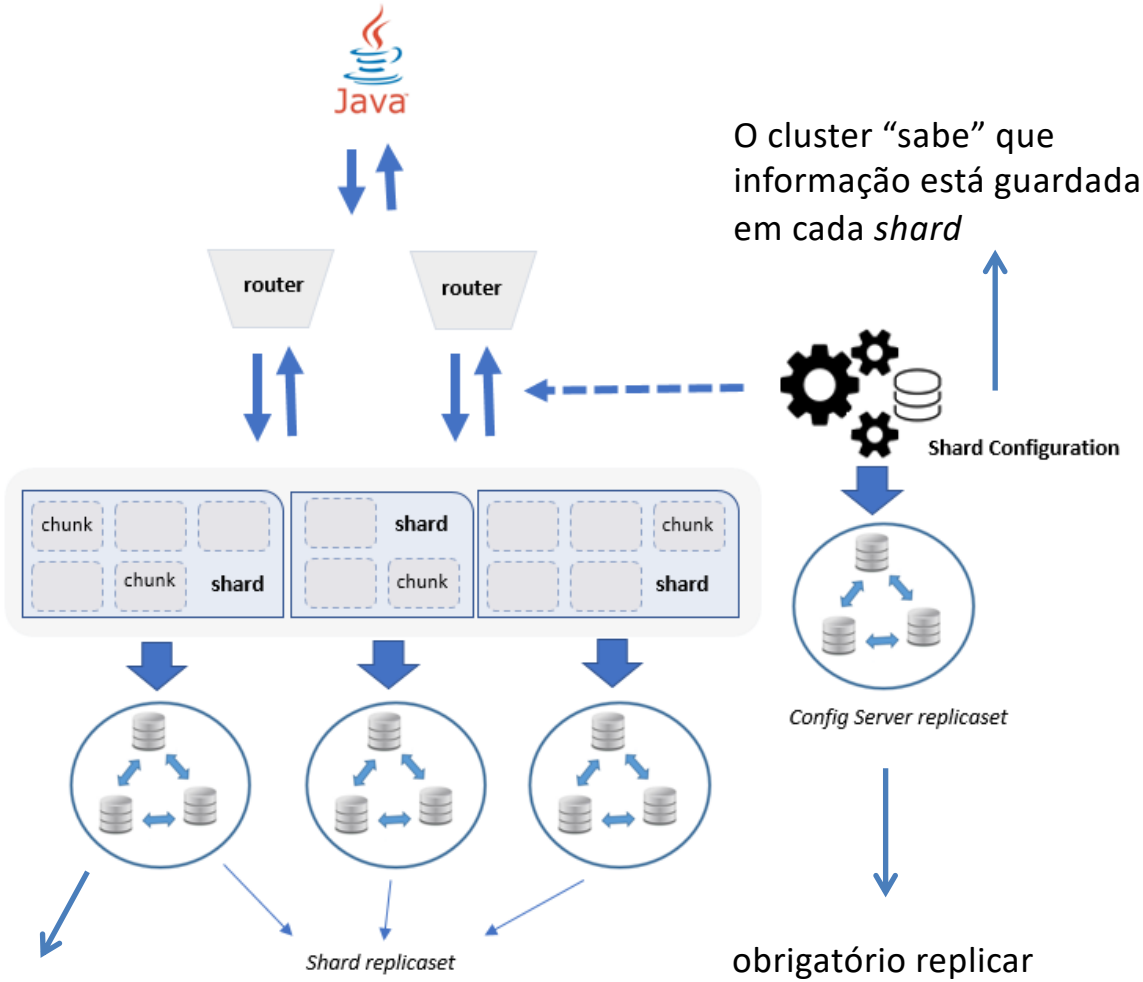


Armazenamento Distribuído - Mongo DB

Arquitectura de distribuição de dados em MongoDB

shard = bd
Chunk = fragmento da bd

Os *shard* podem ser definidos pelo utilizador na configuração inicial da arquitectura, mas os *chunks* são geridos automaticamente pelo MongoDB (sem redundância)

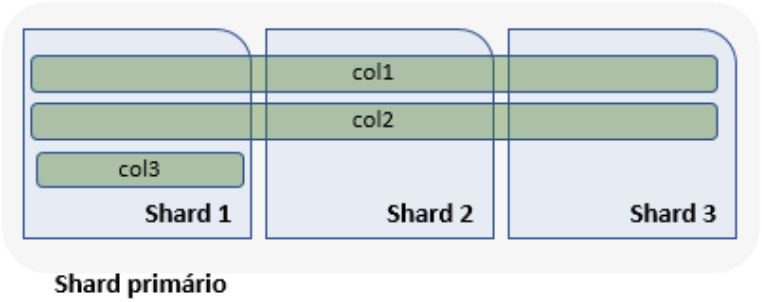


O cluster "sabe" que informação está guardada em cada *shard*

obrigatório replicar

recomendado replicar

Armazenamento Distribuído - Mongo DB



- Não é necessário particionar todas as colecções. Apenas são particionadas (*sharded*) as colecções que explicitamente sejam indicadas na definição da arquitectura

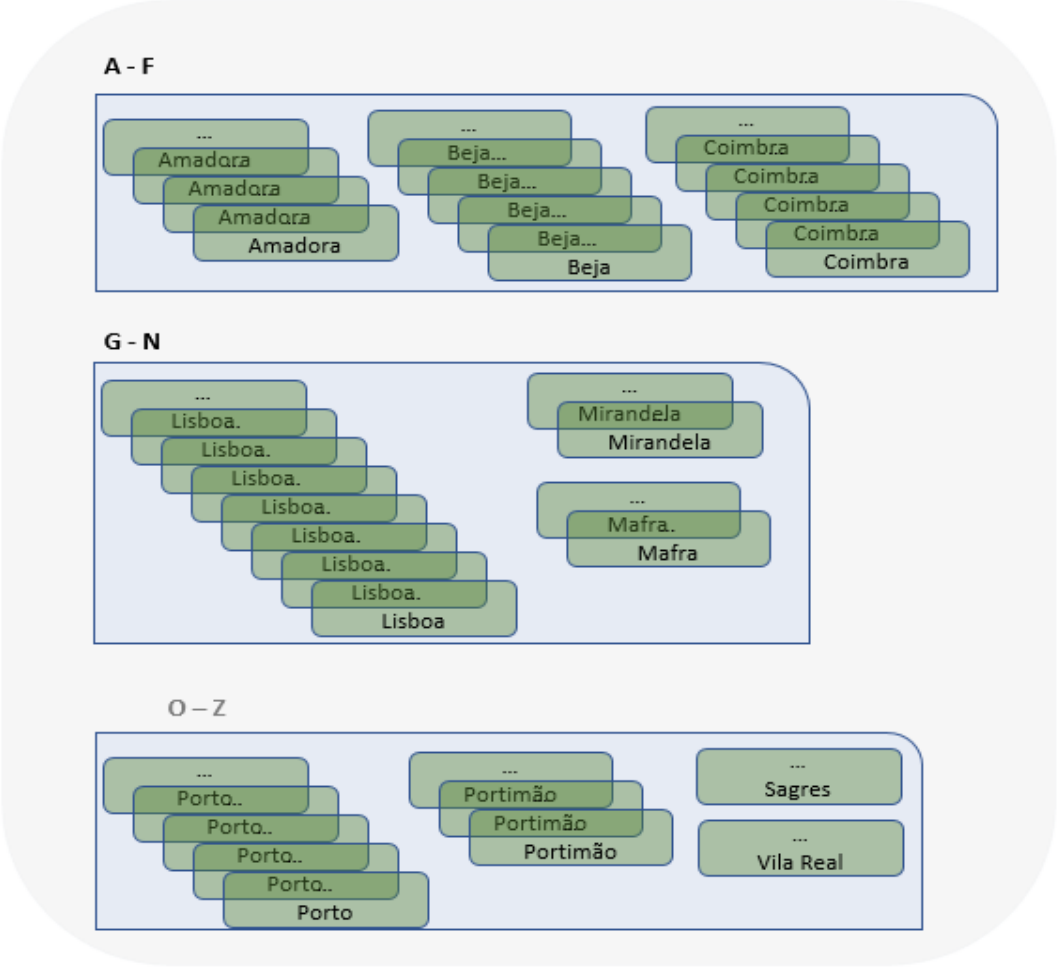
- As restantes colecções (col 3) são todas armazenadas integralmente num único *shard* (primário)

Armazenamento Distribuído - Mongo DB

Estratégia de particionamento
Ranged Sharding

- Agrupar as colecções de forma a guardar no mesmo *shards* aquelas que têm uma chave com valores próximos

- A cada *chunk* é atribuído um intervalo de valores para a chave, e no *chunk* são armazenados os documentos cujo valor da chave esteja no intervalo do *chunk*

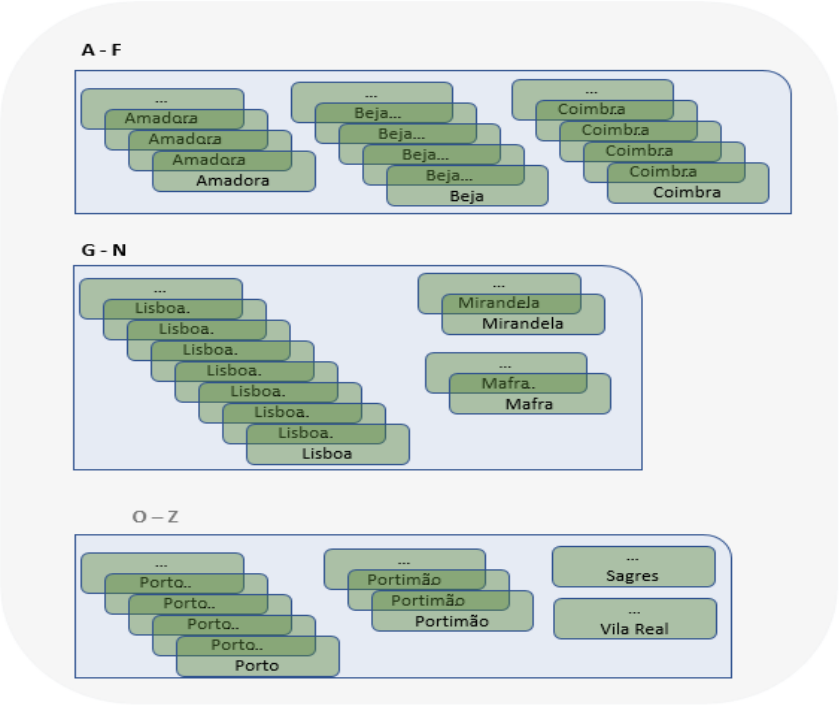


- Esta estratégia serve para facilitar as consultas que vão pesquisar documentos com valores semelhantes para a chave

Armazenamento Distribuído - Mongo DB

Ainda sobre a estratégia *Ranged Sharding*

- O MongoDB fornece um processo de balanceamento que corre de forma transparente para os utilizadores, e que está encarregue de assegurar que **o número de *chunks* em cada *shard* é sensivelmente o mesmo**



- Quando o número de *chunks* dum um dado *shard* atinge um valor demasiado elevado (em termos relativos) o processo de balanceamento (*Balancer*) automaticamente move *chunks* para outro *shard*, qualquer que seja a estratégia de particionamento utilizada

Armazenamento Distribuído - Mongo DB

Estratégia de particionamento *Hashed Sharding*

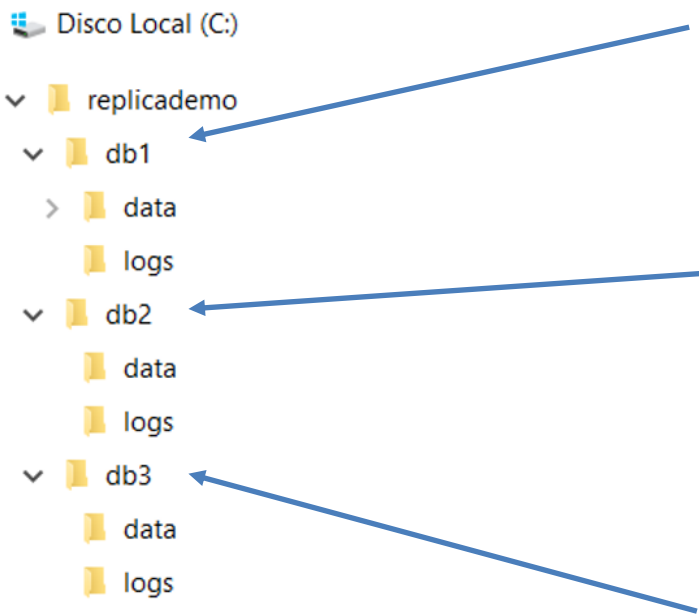
- Os documentos não são indexados com base na chave, mas no resultado da aplicação de uma função *hash* à chave
- Continua a ser necessário escolher uma chave, mas como o índice é sobre o resultado da função e não sobre o valor, os registos são distribuídos de uma forma mais equitativa
  - A aplicação da função a dois valores muito semelhantes retorna valores diferentes

```
> hex_md5('joão1')
ae25aa963af2953c26ed478b6fe7e0f8
> hex_md5('joão2')
bdd8f6558d7cd55fbda5011999d242b1
>
```

- Penaliza as consultas a documentos com valores semelhantes para a chave
- Os routers vão ter de consultar todos os shards para efectuar interrogações
- É a estratégia adequada para chaves cujo valor vai sendo incrementado

Criar cluster de réplicas – Guia Rápido (etapa 1)

1 - Criar estrutura de pastas:



Criar ficheiro *db1.conf* colocar em *c:\replicademo\db1*

```
dbpath=c:\replicademo\db1\data
logpath=c:\replicademo\db1\logs\mongo.log
port=27017
replSet=replicademo
```

Criar ficheiro *db2.conf* colocar em *c:\replicademo\db1*

```
dbpath=c:\replicademo\db2\data
logpath=c:\replicademo\db2\logs\mongo.log
port=25017
replSet=replicademo
```

Criar ficheiro *db3.conf* colocar em *c:\replicademo\db1*

```
dbpath=c:\replicademo\db3\data
logpath=c:\replicademo\db3\logs\mongo.log
port=23017
replSet=replicademo
```

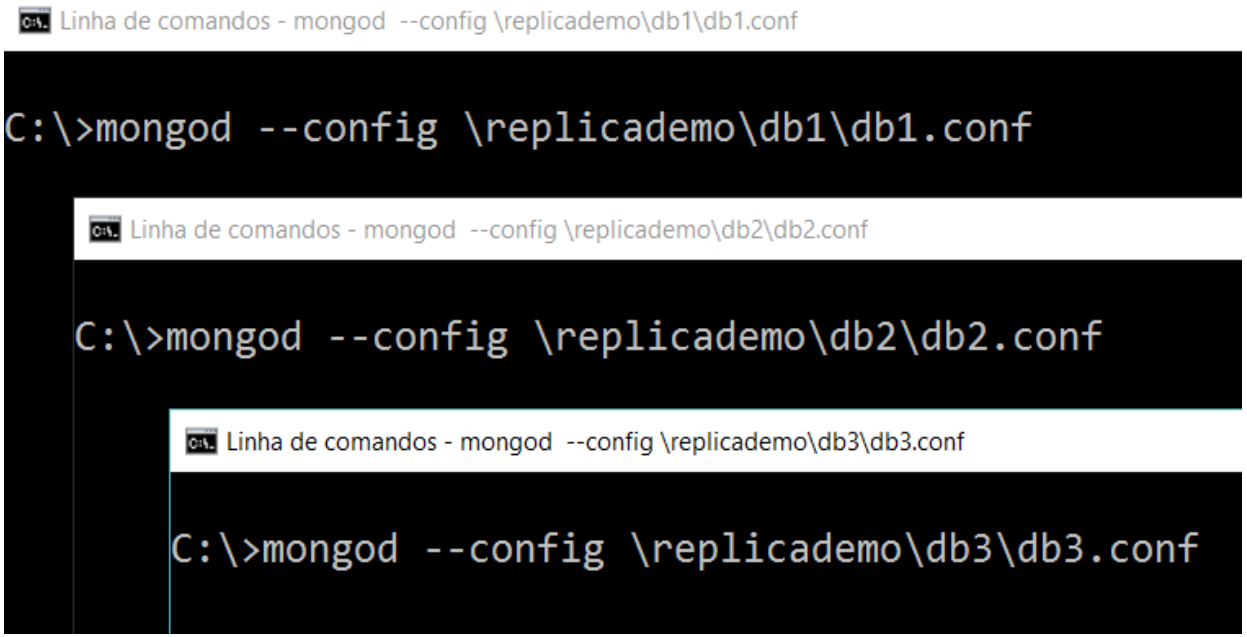
Criar cluster de réplicas– Guia Rápido (etapa 2)

2 - Abrir três janelas de comando, em cada uma delas chamar o servidor:

```
C:\>mongod --config \replicademo\db1\db1.conf

C:\>mongod --config \replicademo\db2\db2.conf

C:\>mongod --config \replicademo\db3\db3.conf
```



Criar cluster de réplicas – Guia Rápido (etapas 3 e 4)

3 - Abrir uma janelas de comando, entrar como cliente no servidor que vai ser primário:

```
mongo --port 27017
```

Iniciar Cluster:

```
rs.initiate()
```

```
> rs.initiate()
{
        "info2" : "no configuration specified. Using a default configuration for the set",
        "me" : "localhost:27017",
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1577029956, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1577029956, 1)
}
```

4- Adicionar os restantes dois (na linha de comando do cliente primário) :

```
rs.add("localhost:25017")
rs.add("localhost:23017")
```

```
replicademo:PRIMARY>
replicademo:PRIMARY>
replicademo:PRIMARY> rs.add("localhost:25017")
{
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1577030053, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1577030053, 1)
}
replicademo:PRIMARY> rs.status()
{
        "set" : "replicademo",
        "date" : ISODate("2019-12-22T15:55:25.483Z"),
```

Criar cluster de réplicas – Guia Rápido (etapa 5)

5 -Testar o Cluster

Abrir o 1º cliente secundário em nova linha de comando:

```
mongo --port 25017
```

```
C:\Users\pnram>mongo --port 25017
MongoDB shell version v4.2.2
connecting to: mongodb://127.0.0.1:25017/?compre
Implicit session: session { "id" : UUID("887b194
MongoDB server version: 4.2.2
```

Abrir o 2º cliente secundário em nova linha de comando:

```
mongo --port 23017
```

```
C:\Users\pnram>mongo --port 23017
MongoDB shell version v4.2.2
connecting to: mongodb://127.0.0.1:23017/?compr
Implicit session: session { "id" : UUID("2abdae
```

Ir à janela do cliente primário e inserir um documento:

```
use db_demo
db.collection_demo.insert({"nome":"ana"})
```

```
replicademo:PRIMARY> use db_demo
switched to db db_demo
replicademo:PRIMARY> db.collection_demo.insert({"nome":"ana"})
WriteResult({ "nInserted" : 1 })
replicademo:PRIMARY>
```

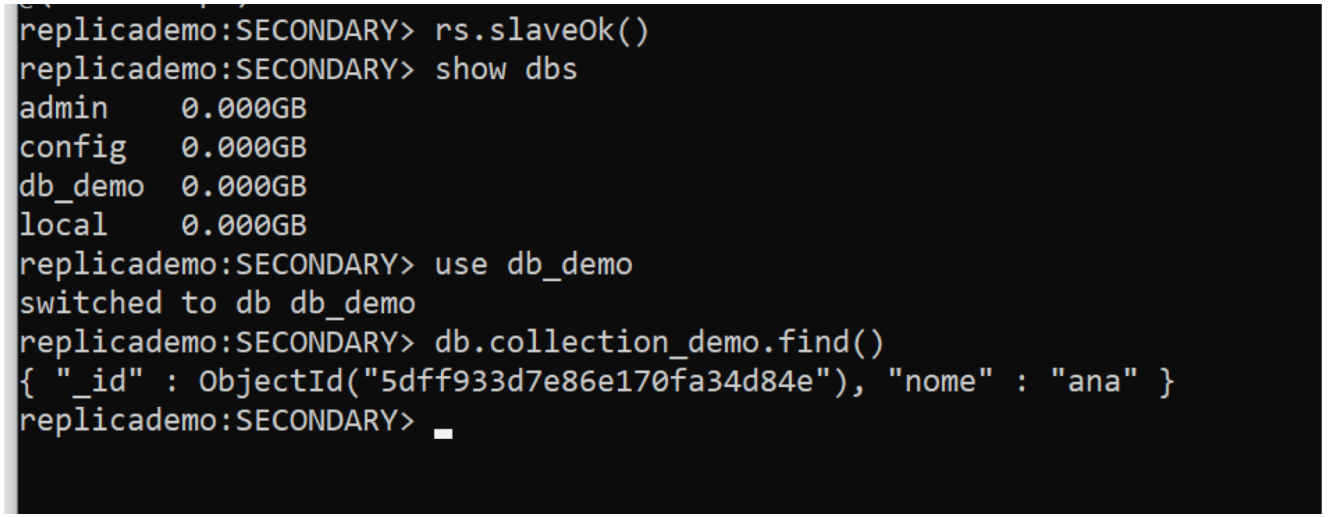
Criar cluster de réplicas – Guia Rápido (etapa 6)

Ir à janela do cliente secundário :

`rs.slaveOk()` → **Obrigatório para que secundário possa servir para leitura**

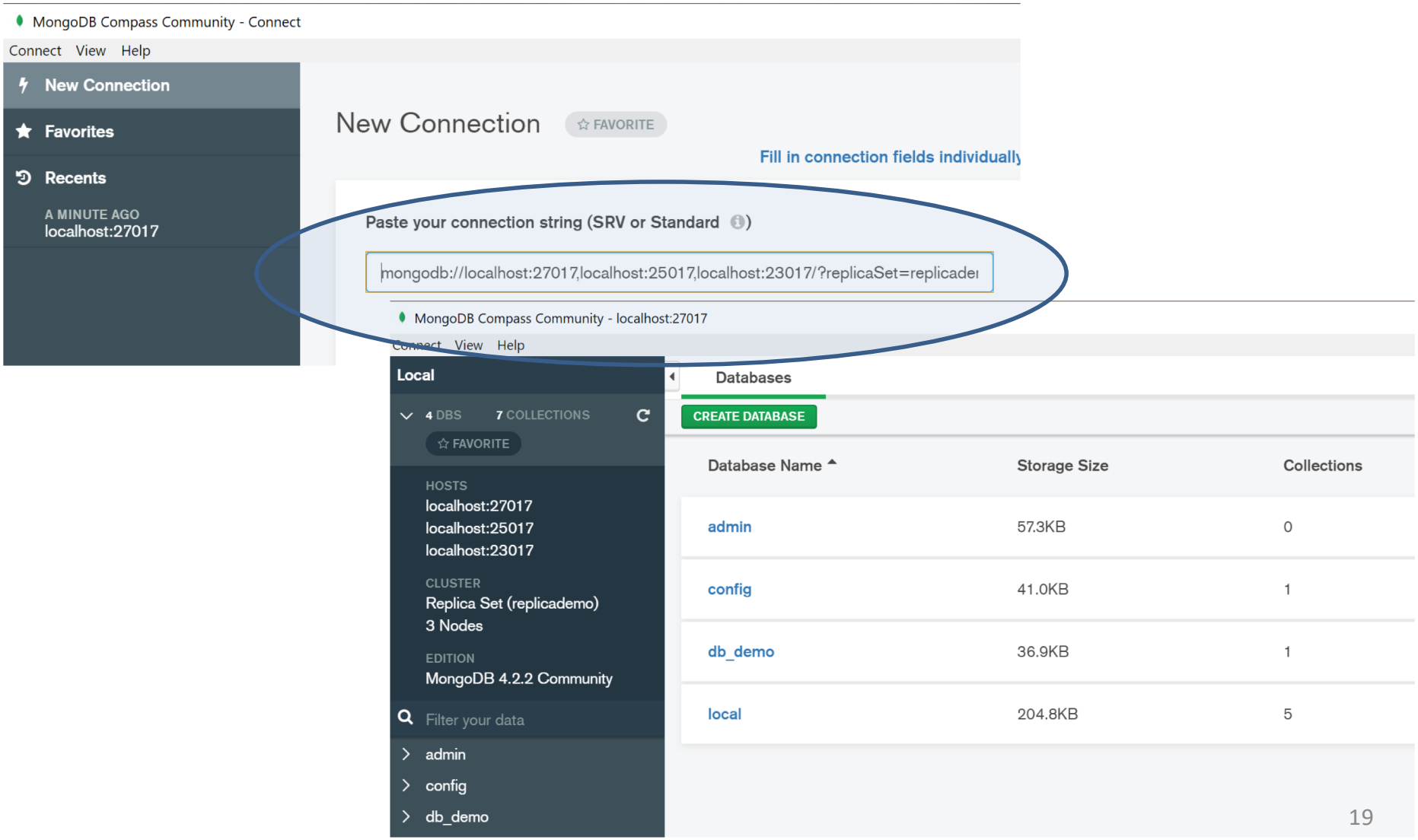
```
use db_demo

db.collection_demo.find()
```



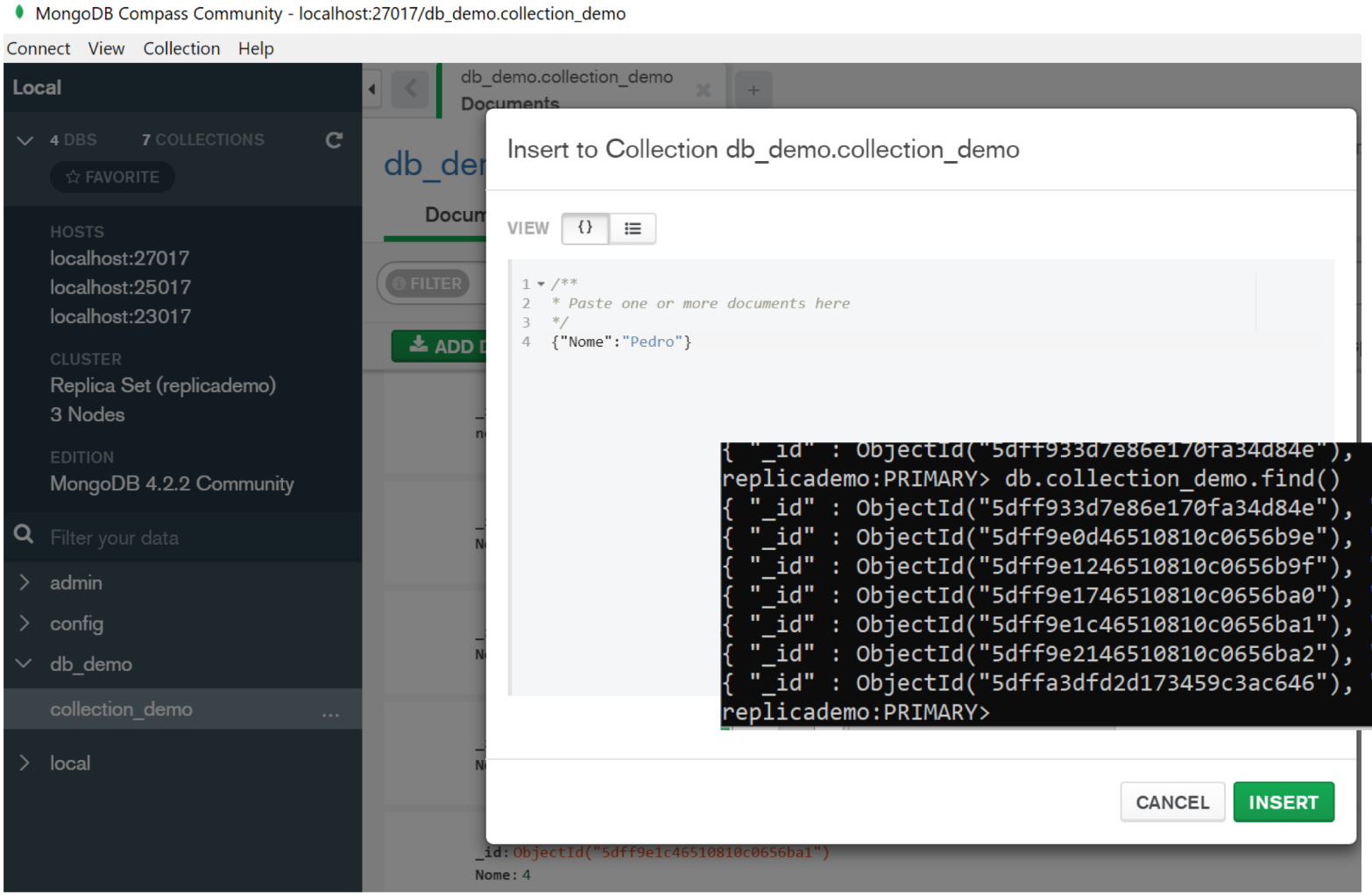
MongoDB Compass

Para aceder ao Cluster através da interface gráfica MongoDB Compass



MongoDB Compass

Como inserir um documento:



Testar na linha de comandos:

```
{ "_id" : ObjectId("5dff933d7e86e170fa34d84e"), "nome" : "ana" }
replicademo:PRIMARY> db.collection_demo.find()
{ "_id" : ObjectId("5dff933d7e86e170fa34d84e"), "nome" : "ana" }
{ "_id" : ObjectId("5dff9e0d46510810c0656b9e"), "Nome" : 1 }
{ "_id" : ObjectId("5dff9e1246510810c0656b9f"), "Nome" : 2 }
{ "_id" : ObjectId("5dff9e1746510810c0656ba0"), "Nome" : 3 }
{ "_id" : ObjectId("5dff9e1c46510810c0656ba1"), "Nome" : 4 }
{ "_id" : ObjectId("5dff9e2146510810c0656ba2"), "Nome" : 5 }
{ "_id" : ObjectId("5dffa3dfd2d173459c3ac646"), "Nome" : "Pedro" }
replicademo:PRIMARY>
```

Inseriu …