Bancos de Dados Não-Relacionais
CET 0620 - Módulo 03d

Professor Franklin Amorim

MongoDB

Arquitetura

Replica Set

Replica Set

Um Replica Set é um grupo de instâncias do MongoDB que mantêm o mesmo conjunto de dados.

Ele fornece redundância e alta disponibilidade, garantindo que, se um nó falhar, os dados permanecerão acessíveis a partir de outros nós.

Replica Set é o principal mecanismo para implementar a tolerância a falhas nas implantações de MongoDB.

Replica Set

Principais características:

Nó primário: apenas um nó no serve como nó primário. Ele recebe todas as operações de gravação e propaga as alterações para os outros nós do conjunto.

Nós secundários: os nós restantes são nós secundários. Eles replicam dados do nó primário e estão disponíveis para assumir o controle do nó primário em caso de falha.

Failover automático: se o nó primário ficar indisponível (devido a uma falha ou manutenção), o conjunto de réplicas acionará automaticamente um processo de failover. Um dos nós secundários é então eleito como o novo primário, garantindo a disponibilidade contínua do banco de dados.

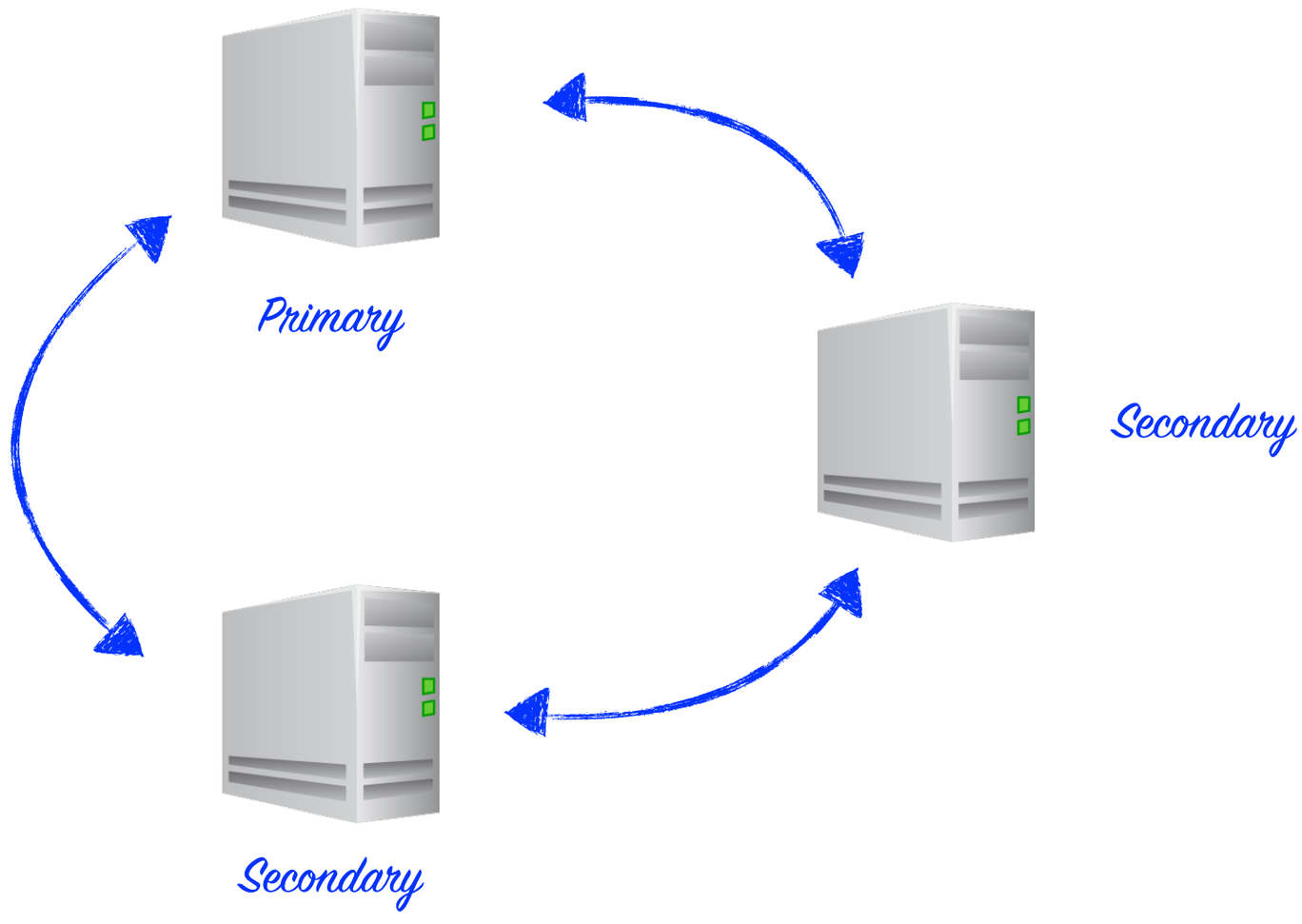
Replicação de dados: o MongoDB usa replicação para sincronizar dados entre nós. O nó primário registra todas as operações de gravação em seu oplog (log de operações), que os nós secundários então replicam.

Escalabilidade de leitura: os nós secundários também podem servir operações de leitura, melhorando a escalabilidade de leitura. Isso permite distribuir a carga de leitura entre vários nós e melhorar o desempenho geral do sistema.

Árbitros: árbitros são nós opcionais no conjunto de réplicas que participam do processo de eleição, mas não armazenam dados. Eles ajudam a garantir que haja sempre um número ímpar de membros votantes para evitar cenários de split brain durante as eleições.

Replica Set

Replica Set



Replica Set

Criado um Replica Set:

```
docker network create my-mongo-cluster

docker run --name mongo_rs1 -d --net my-mongo-cluster -p 27021:27021
mongo --port 27021 --replSet replica

docker run --name mongo_rs2 -d --net my-mongo-cluster -p 27022:27022
mongo --port 27022 --replSet replica


docker ps -f name=mongo_rs
```

Replica Set

Criado um Replica Set:

```
docker exec -it mongo_rs1 mongosh --port 27021

> rsconf = {
        _id: "replica",
        members: [
                  {
                   _id: 0,
                   host: "mongo_rs1:27021"
                  }
                 ]
       };

> rs.initiate( rsconf );

> rs.status().members
```

Replica Set

Criado um Replica Set:

```
> rs.add("mongo_rs2:27022")

> rs.status().members
```

Replica Set

Simulando a falha em um nó:

```
> exit;

docker stop mongo_rs2

docker exec -it mongo_rs1 mongosh --port 27021

> rs.status().members
```

Who is the primary? Why?

Replica Set

Falha em um nó

Primary

Replica Set

Falha em um nó

Replica Set

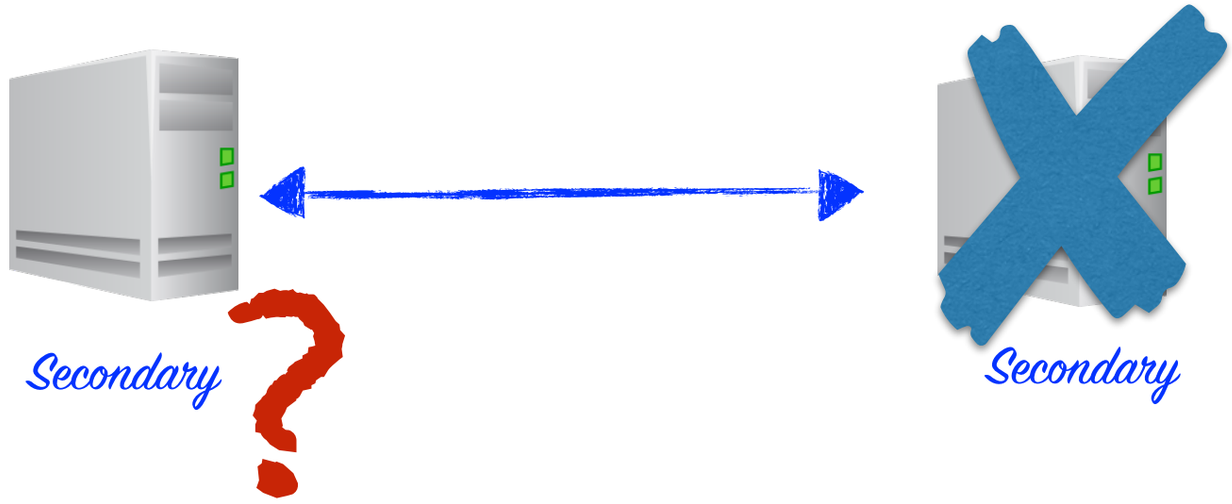
Falha em um nó

Replica Set

Falha em um nó

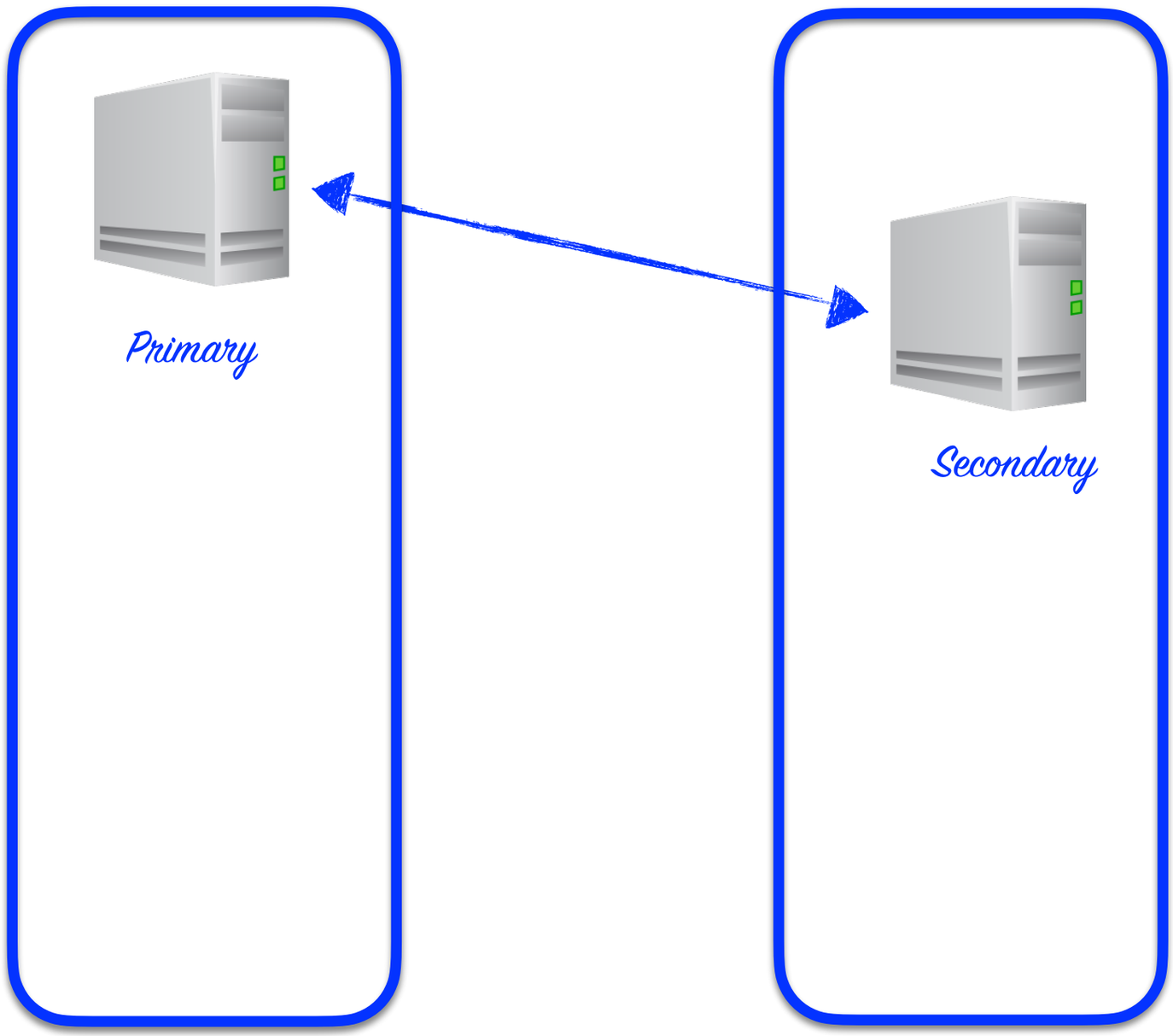
Replica Set

Falha em um nó



Replica Set

Split Brain

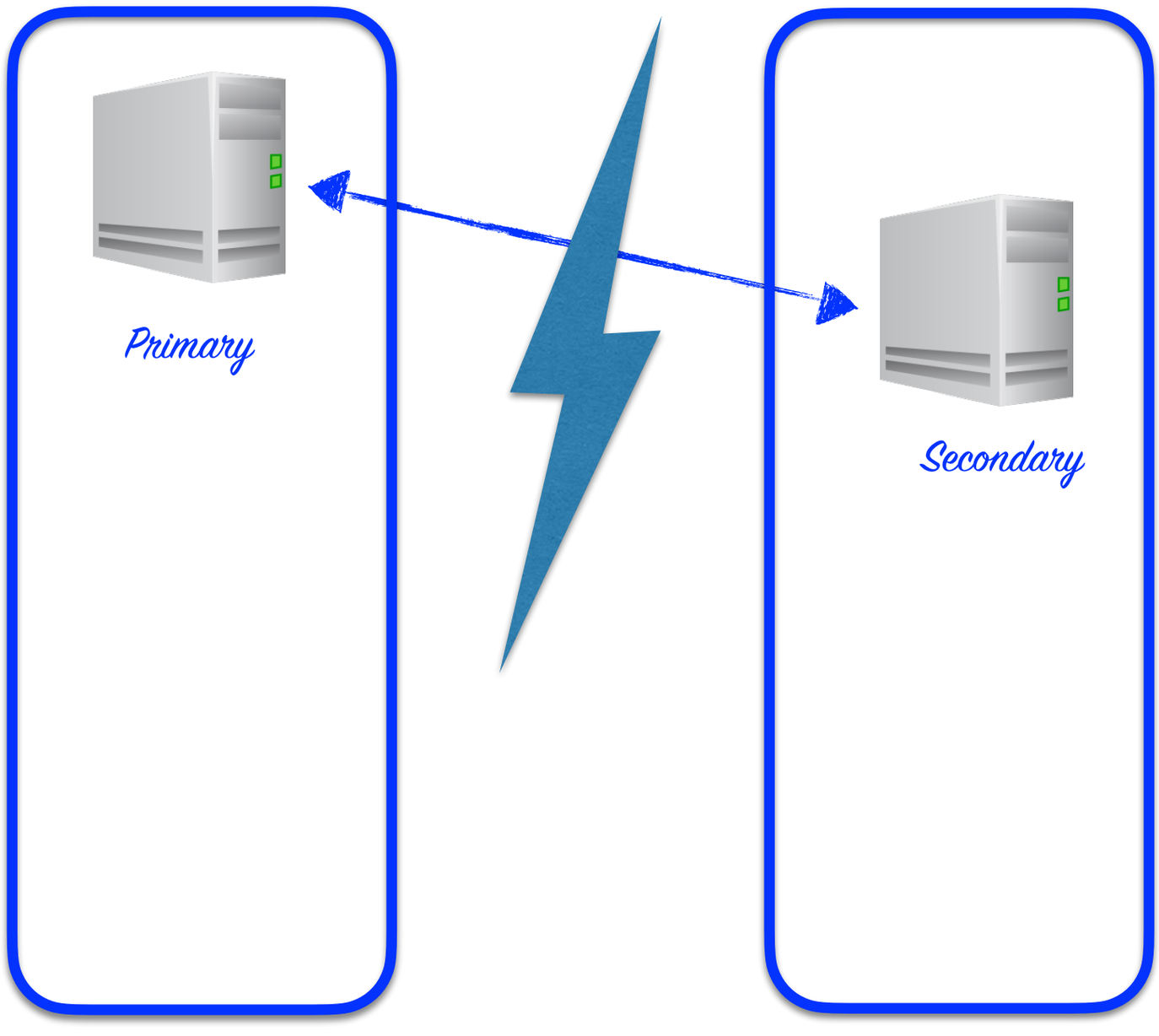


Países Baixos

Brasil

Replica Set

Split Brain

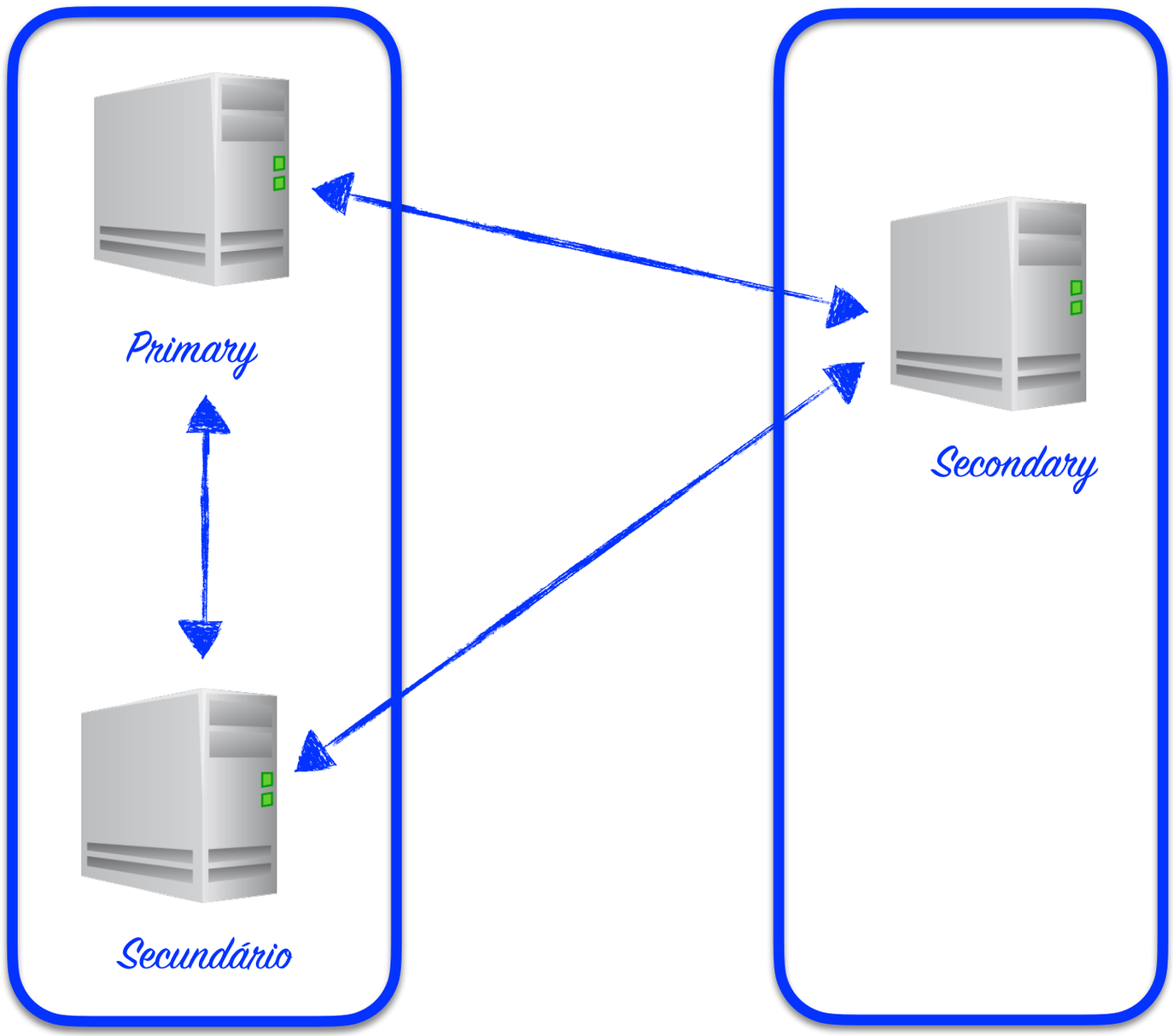


Países Baixos

Brasil

Replica Set

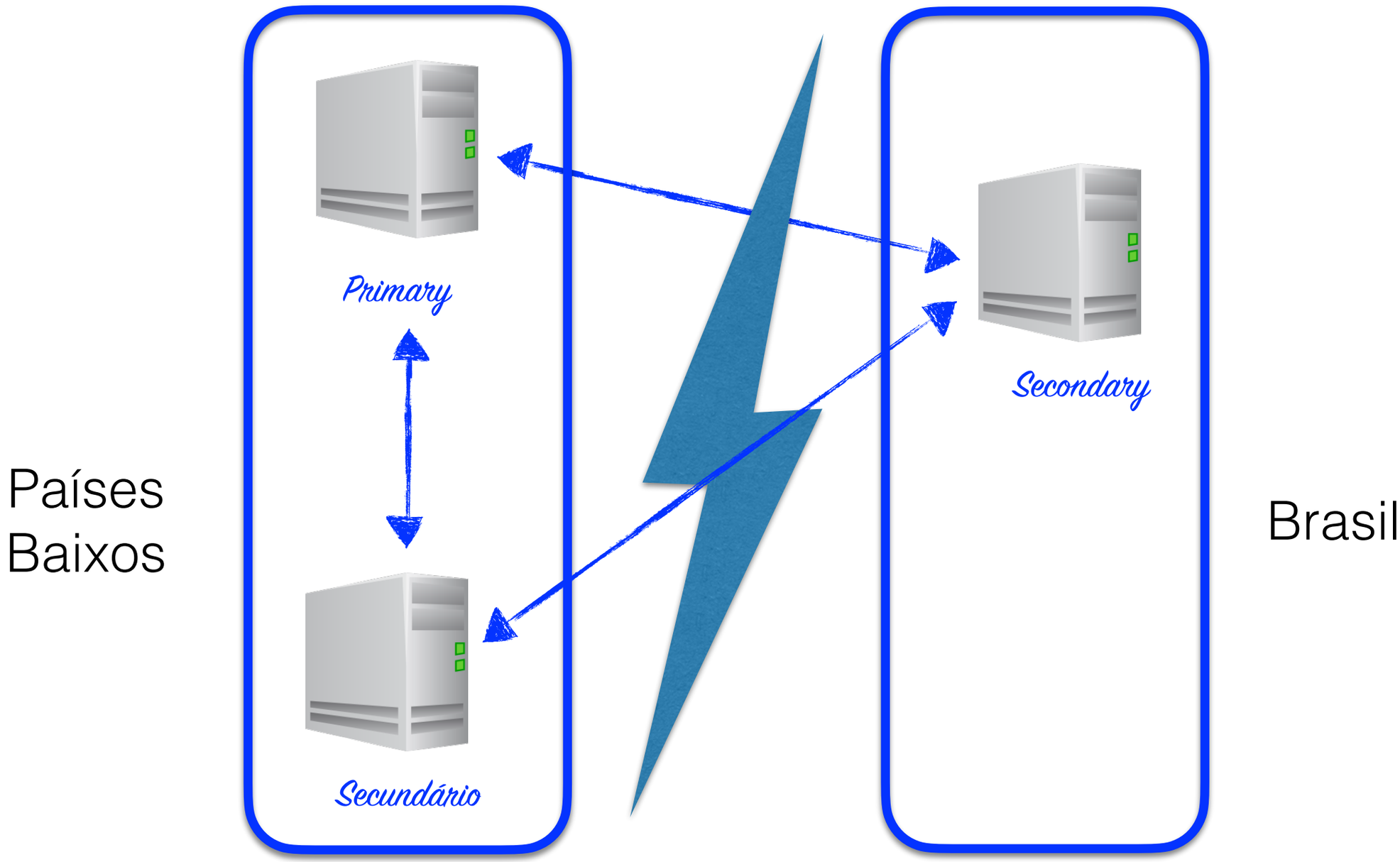
Split Brain



Países Baixos

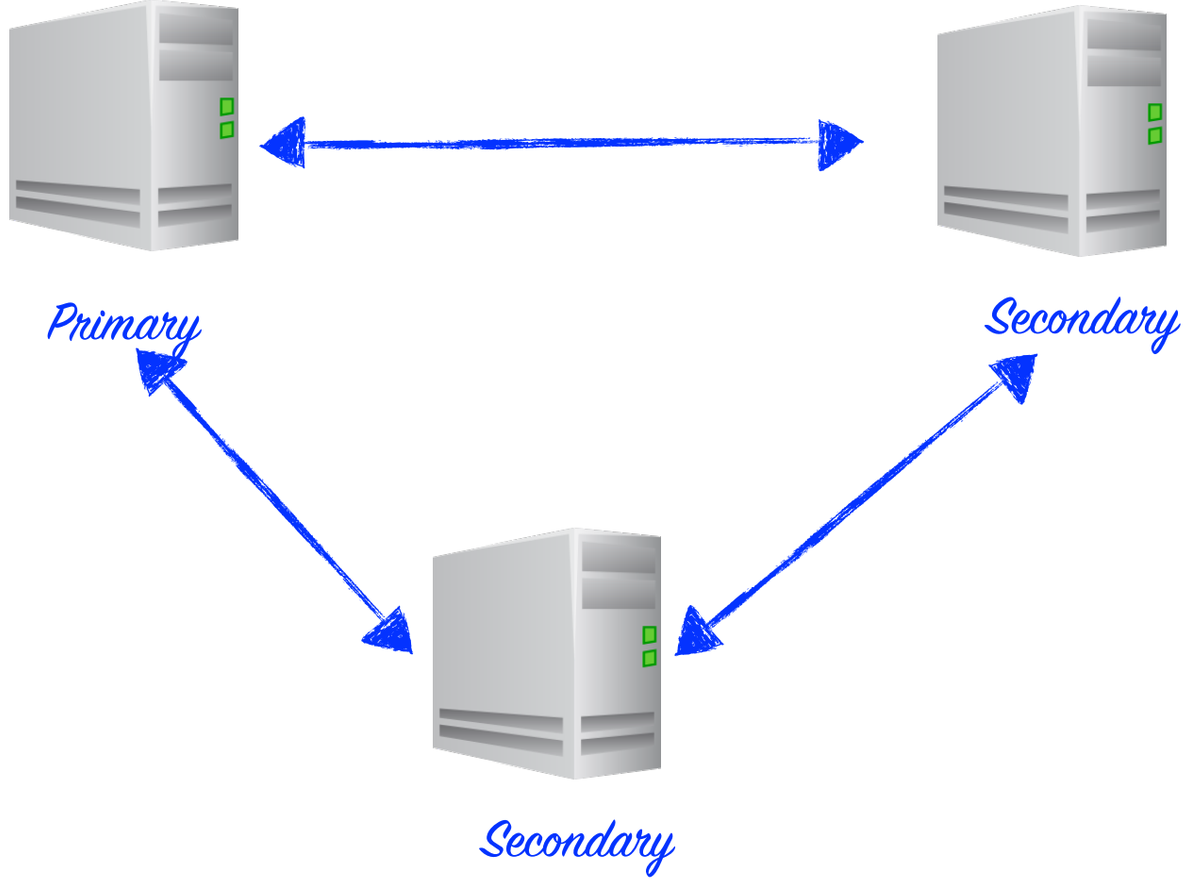
Brasil

Replica Set


Split Brain
Primary
Secundário
Secondary
Países Baixos
Brasil

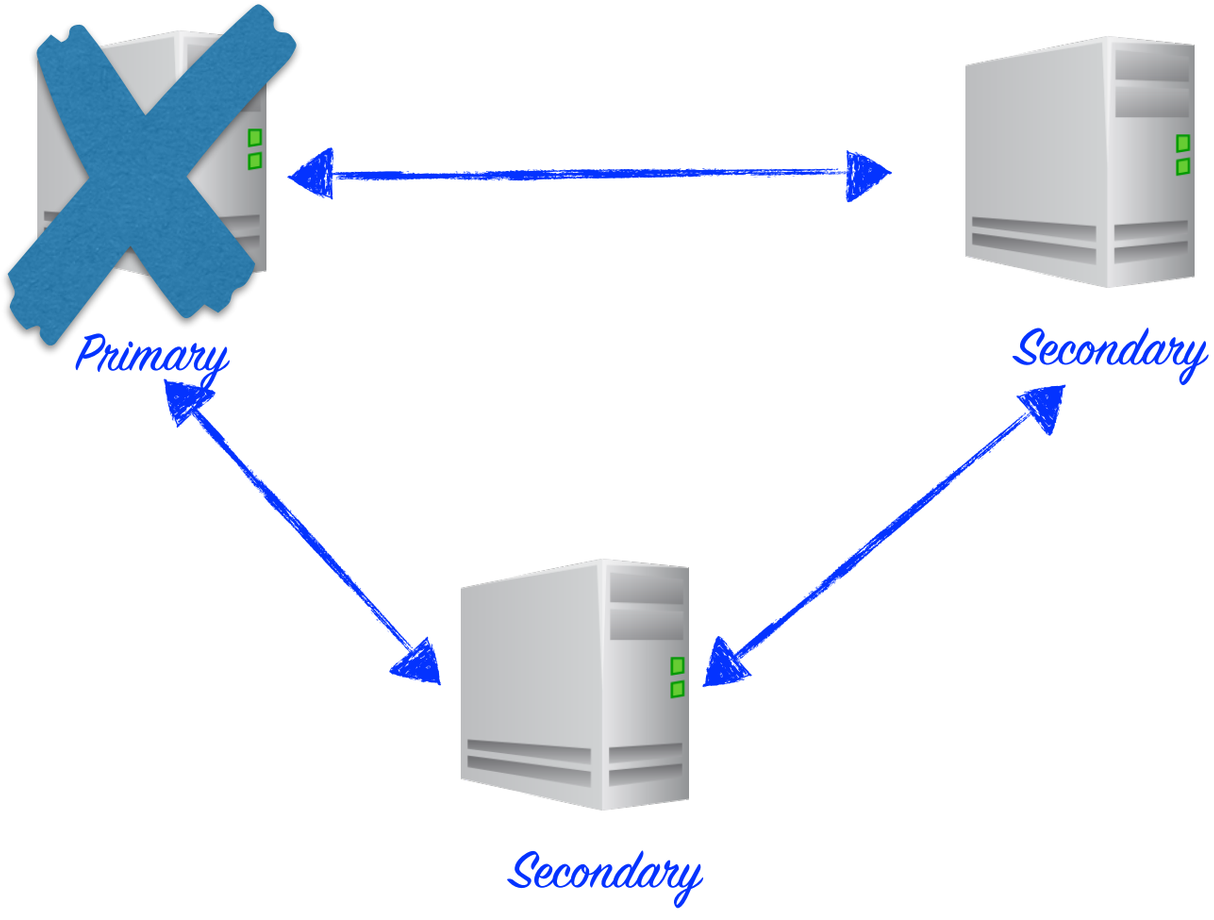
Replica Set

Elegendo um novo primário



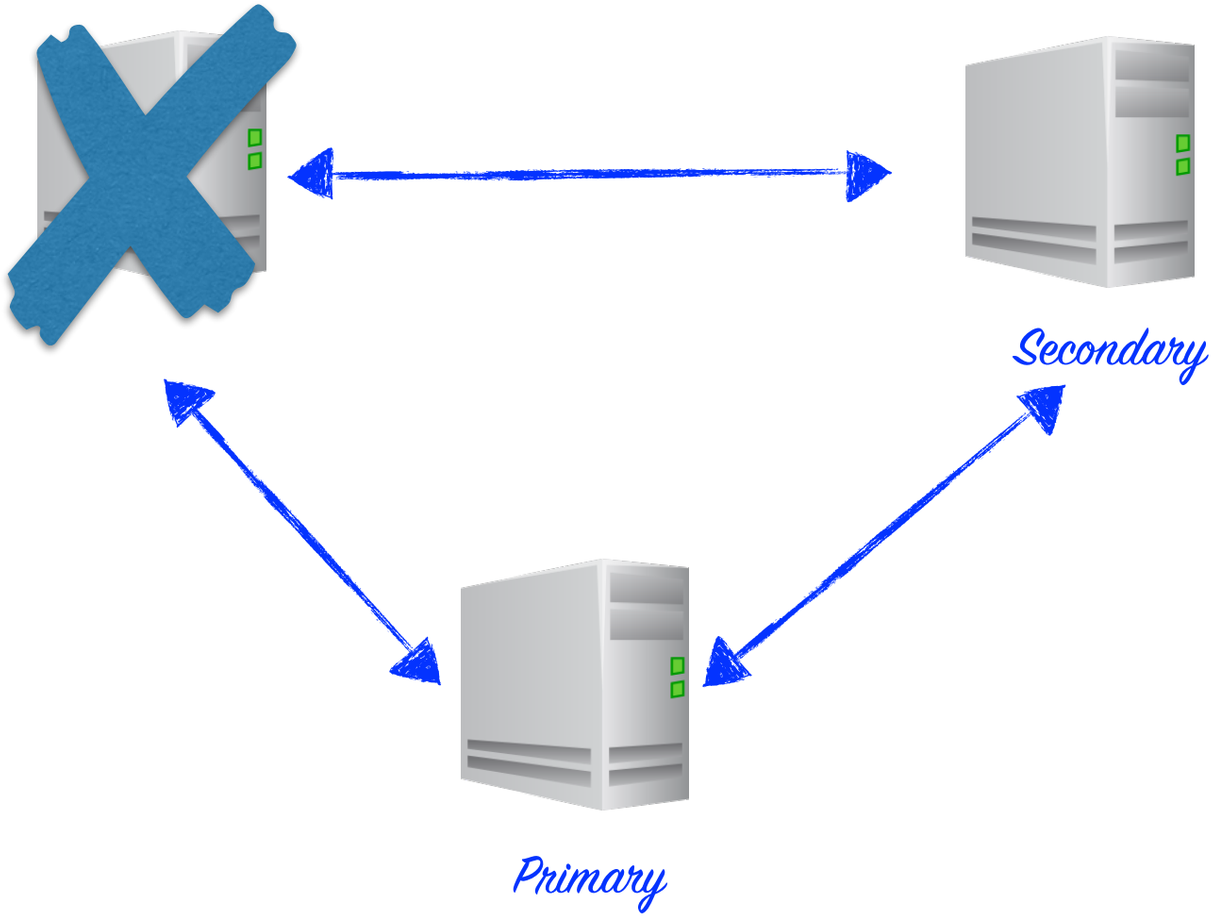
Replica Set

Elegendo um novo primário



Replica Set

Elegendo um novo primário



Replica Set

Inicie o mongo_rs2 novamente

```
> exit;

docker start mongo_rs2
```

Replica Set

Adicionando um terceiro nó

```
docker run --name mongo_rs3 -d --net my-mongo-cluster
-p 27023:27023 mongo --port 27023 --replSet replica

docker exec -it mongo rs1 mongosh --port 27021

> rs.add("mongo_rs3:27023")

> rs.status().members
```

Replica Set

Agora, pare um nó. O que acontece?

Inicie-o de novo.

Replica Set

Vamos carregar os dados da coleção "movies" para o cluster usando o MongoDB Compass

MongoDB Compass

Compass

New connection +

Saved connections

accounting-de-prod
Nov 17, 2023, 12:21 PM

pom-nl-prod-fca
Nov 29, 2023, 4:22 PM

store-nl-prod
Nov 21, 2023, 9:43 AM

Recents

localhost:27017
Mar 10, 2024, 8:45 AM

core.3hasb.mongodb....
Mar 13, 2024, 5:35 PM

localhost:27017
Mar 14, 2024, 1:55 PM

localhost:27017
Mar 12, 2024, 4:52 PM

salesforce-bridge.jfqc...
Mar 14, 2024, 9:23 AM

New Connection

Connect to a MongoDB deployment

atualize a porta para 27021

FAVORITE

URI

Edit Connection String

mongodb://localhost:27021/

> Advanced Connection Options

Save

Save & Connect

Connect

New to Compass and don't have a cluster?

If you don't already have a cluster, you can create one for free using MongoDB Atlas

CREATE FREE CLUSTER

MongoDB Compass

Compass

New connection +

Saved connections

- accounting-de-prod
  Nov 17, 2023, 12:21 PM
- pom-nl-prod-fca
  Nov 29, 2023, 4:22 PM
- store-nl-prod
  Nov 21, 2023, 9:43 AM

Recents

- localhost:27017
  Mar 10, 2024, 8:45 AM
- core.3hasb.mongodb....
  Mar 13, 2024, 5:35 PM
- localhost:27017
  Mar 14, 2024, 1:55 PM
- localhost:27017
  Mar 12, 2024, 4:52 PM
- salesforce-bridge.jfqc...
  Mar 14, 2024, 9:23 AM

New Connection

Connect to a MongoDB deployment

FAVORITE

URI

Edit Connection String

mongodb://localhost:27021/

> Advanced Connection Options

Save

Save & Connect

Connect

New to Compass and don't have a cluster?

If you don't already have a cluster, you can create one for free using MongoDB Atlas

CREATE FREE CLUSTER

Clique para abrir as opções avançadas

MongoDB Compass

Compass

New connection +

Saved connections

- accounting-de-prod
  Nov 17, 2023, 12:21 PM
- pom-nl-prod-fca
  Nov 29, 2023, 4:22 PM
- store-nl-prod
  Nov 21, 2023, 9:43 AM

Recents

- localhost:27017
  Mar 10, 2024, 8:45 AM
- core.3hasb.mongodb....
  Mar 13, 2024, 5:35 PM
- localhost:27017
  Mar 14, 2024, 1:55 PM
- localhost:27017
  Mar 12, 2024, 4:52 PM
- salesforce-bridge.jfqc...
  Mar 14, 2024, 9:23 AM

Advanced Connection Options

General | Authentication | TLS/SSL | Proxy/SSH | In-Use Encryption | Advanced

Connection String Scheme

mongodb | mongodb+srv

Standard Connection String Format. The standard format of the MongoDB connection URI is used to connect to a MongoDB deployment: standalone, replica set, or a sharded cluster.

Host

localhost:27021 +

Direct Connection

Specifies whether to force dispatch all operations to the specified host.

marque esta opção

Save | Save & Connect | Connect

MongoDB Compass - localhost:27021/My Queries

localhost:27021

My Queries

Performance

Databases

Search

admin

config

local

My Queries

No saved queries yet.

Start saving your aggregations and find queries, you'll see them here.

Not sure where to start? Visit our Docs →

Siga os mesmos procedimentos usados
no módulo 04 para carregar a collection movies

>_MONGOSH

Replica Set

Se você tentar ler do secundário, receberá um erro:

```
docker exec -it mongo_rs3 mongosh --port 27023

replica [direct: secondary] test> use aula

replica [direct: secondary] aula> db.movies.find({}, {title:1,  id:0})

MongoServerError: not primary and secondaryOk=false - consider using
db.getMongo().setReadPref() or readPreference in the connection string
```

Replica Set

Para ler de um secundário você precisa definir a preferência de leitura no shell:

```
replica [direct: secondary] aula> db.getMongo().setReadPref('secondary')

replica [direct: secondary] aula> db.movies.find({}, {title:1,  _id:0})

[
  { title: 'The Poor Little Rich Girl' },
  { title: 'The Gold Rush' },
  { title: 'Greed' },
  (...)
]
```

Replica Set

Oplog

O MongoDB usa uma coleção especial chamada oplog para replicar as informações entre o primário e o secundário.

O oplog (abreviação de log de operação) é uma coleção especial no MongoDB que registra todas as alterações no nó primário.

Replica Set

Oplog

Vamos reconectar no primário e inserir um documento na coleção `aluno`:

```
replica [direct: secondary] test> exit;

docker exec -it mongo_rs1 mongosh --port 27021

replica [direct: primary] teste> use aula;

replica [direct: primary] aula> db.aluno.insert({nome: "joao"});
```

Replica Set

Oplog

Vamos mudar para o database `local` para consultar o oplog:

```
replica [direct: primary] aula> db.aluno.insert({nome: "joao"});

replica [direct: primary] aula> use local

replica [direct: primary] local> db.oplog.rs.find({ns: "aula.aluno"})
[
  {
    lsid: {
      id: UUID('0622525e-9511-4407-9e69-dbfec7bc104e'),
      uid: Binary.createFromBase64('47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=', 0)
    },
    txnNumber: Long('1'),
    op: 'i',
    ns: 'aula.aluno',
    ui: UUID('d891ec87-a7cd-45df-9ae8-62dd5d2625e5'),
    o: { _id: ObjectId('66c9b78bb0f809c307c76a8c'), nome: 'joao' },
    o2: { _id: ObjectId('66c9b78bb0f809c307c76a8c') },
    stmtId: 0,
    ts: Timestamp({ t: 1724495755, i: 2 }),
    t: Long('1'),
    v: Long('2'),
    wall: ISODate('2024-08-24T10:35:55.793Z'),
    prevOpTime: { ts: Timestamp({ t: 0, i: 0 }), t: Long('-1') }
  }
]
```

Replica Set

Podemos também forçar um "failover" do nó primário

```
docker exec -it mongo_rs1 mongosh --port 27021
```

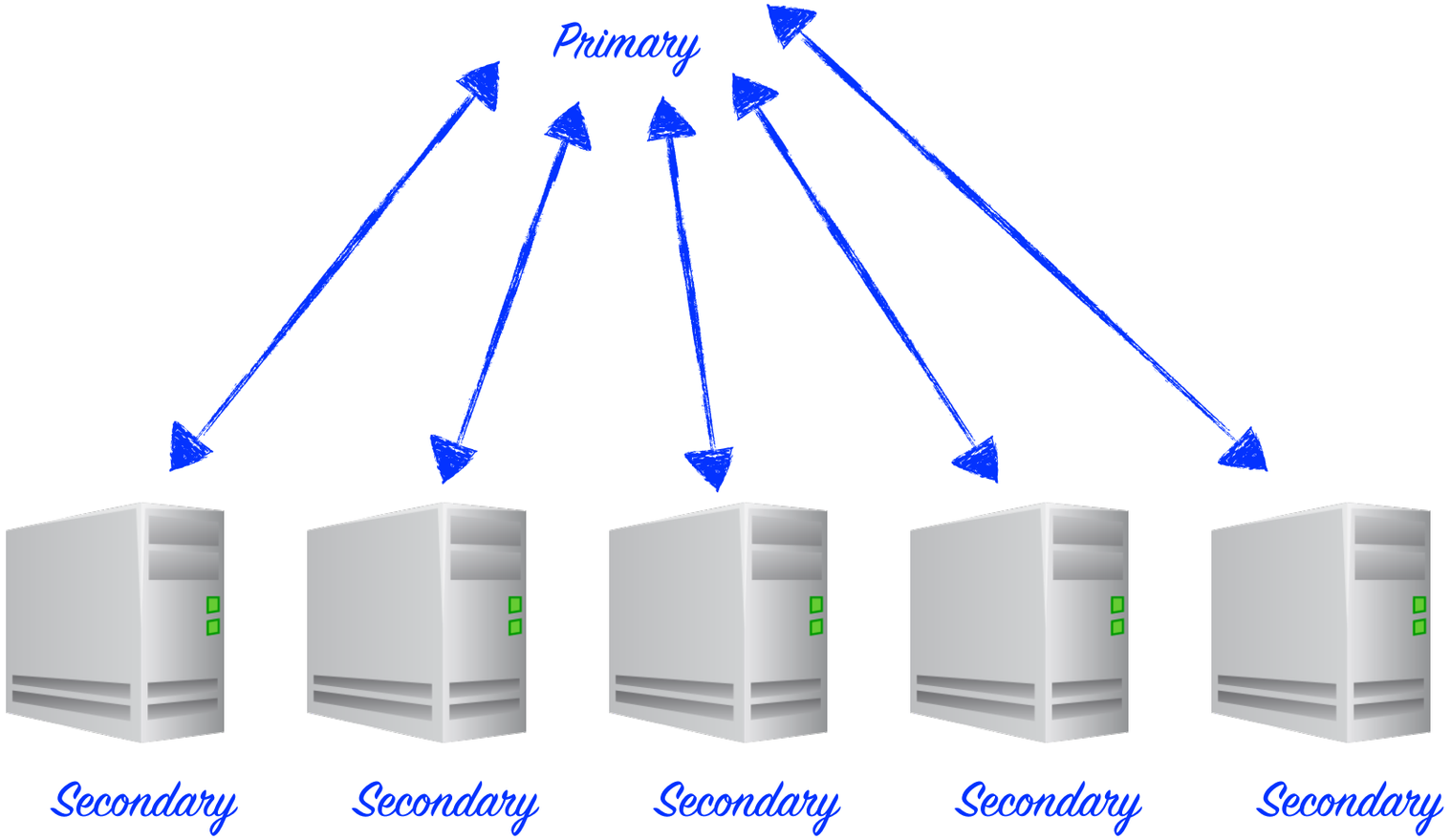
```
> rs.status().members
```

```
> rs.stepDown()
```

Replica Set

Escalando leituras

Leitura e escrita



Apenas leitura