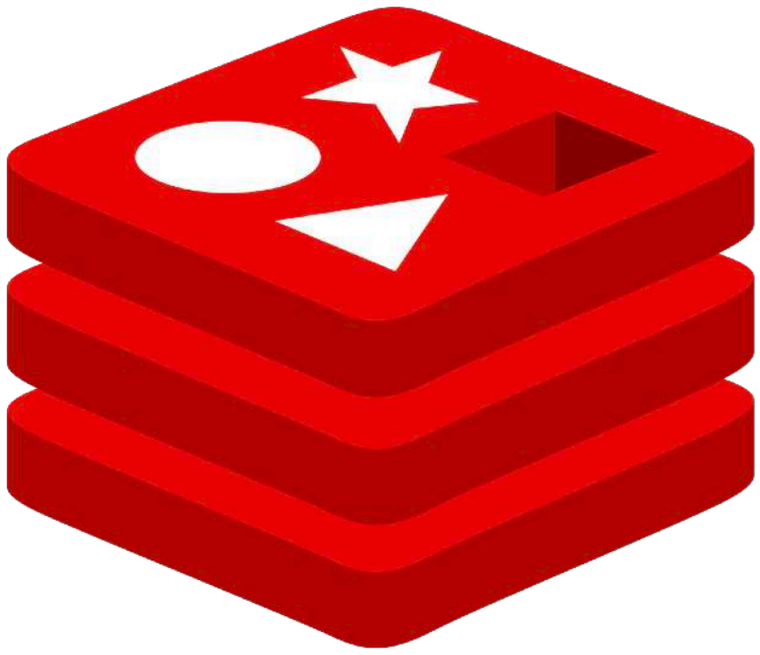


Banco de dados NoSQL - Chave/Valor com Redis

Cluster Redis

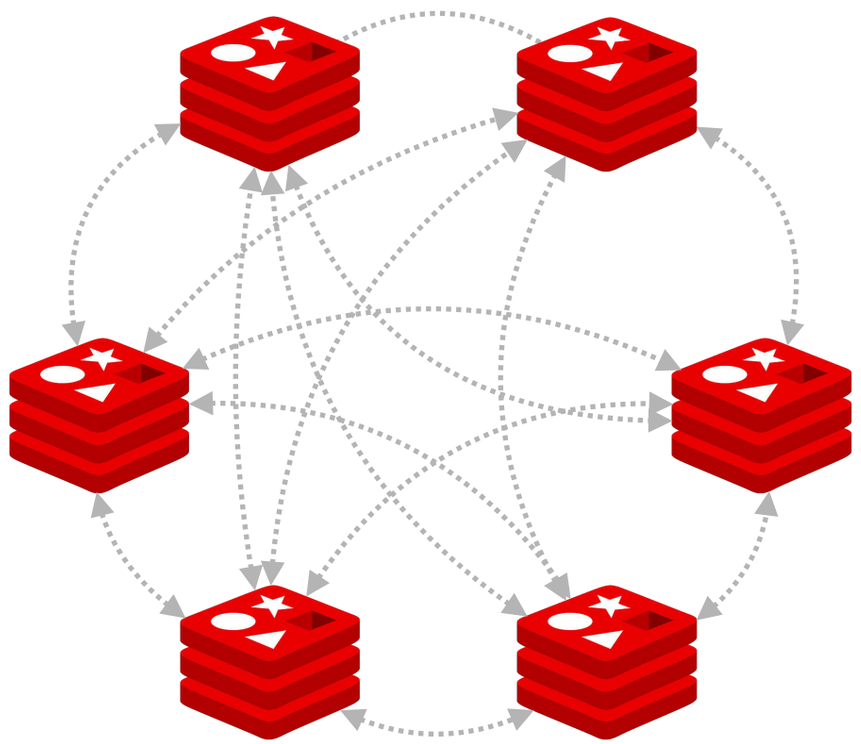
Prof. Gustavo Leitão

Cluster Redis

Como funciona?

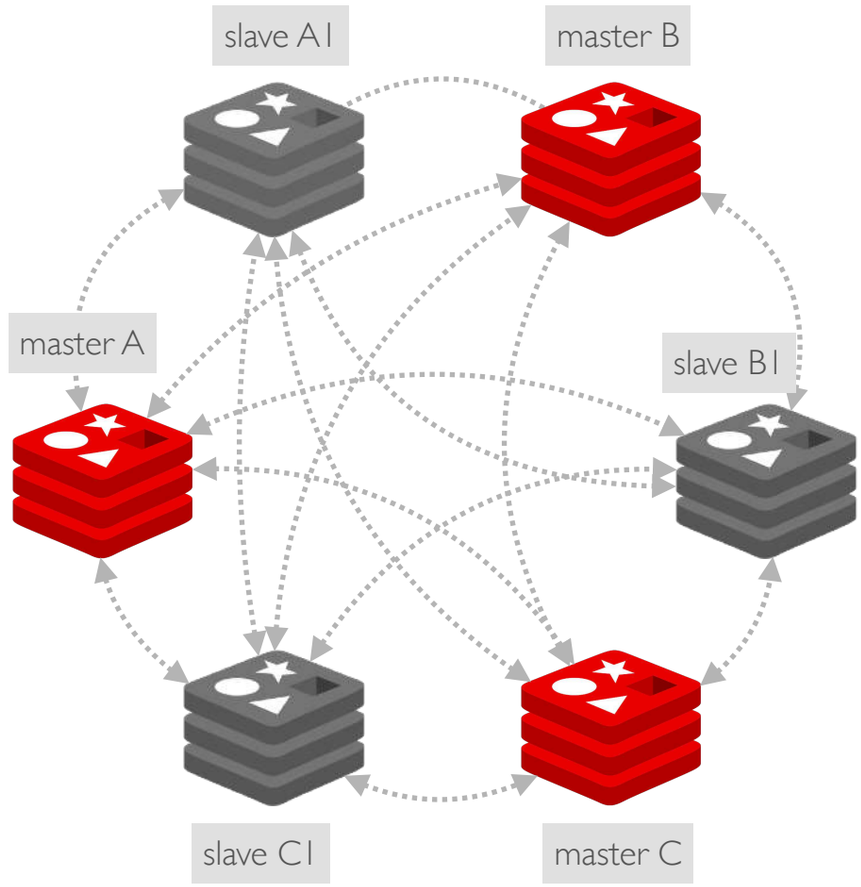
Cluster

- Divisão do trabalho entre um conjunto de máquinas.
- No Redis, os dados são automaticamente dividido (*sharded*) entre os nós do cluster.
- No Redis, o cluster permite também aumentar a disponibilidade. Ou seja, o cluster pode continuar operando mesmo com alguns nós indisponíveis.



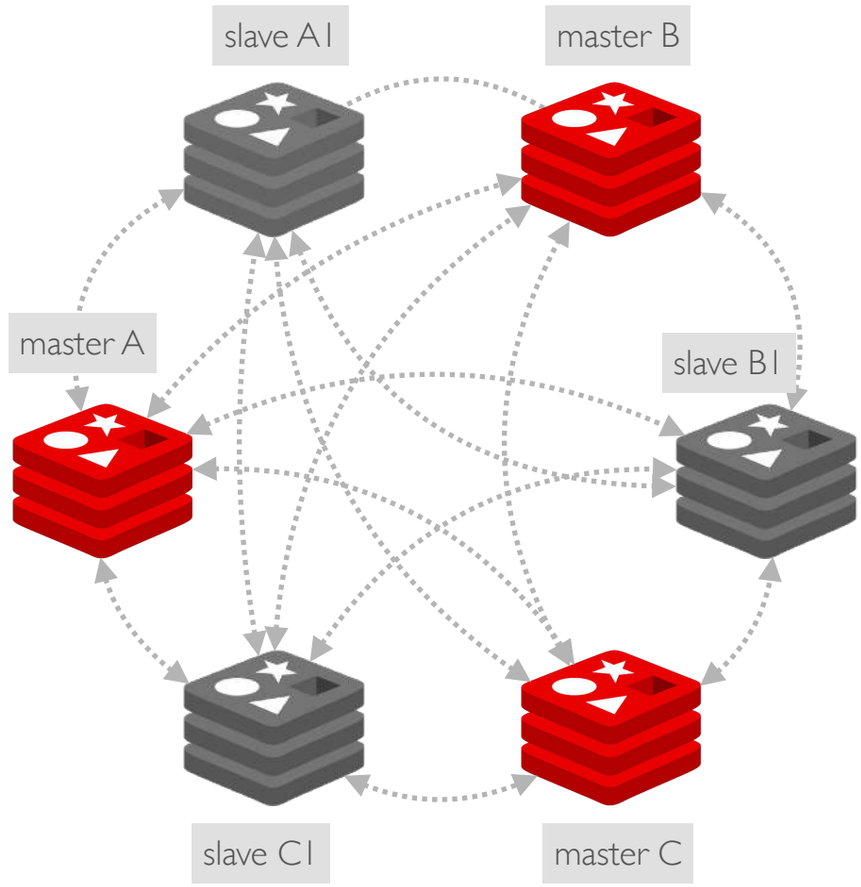
Cluster

- O redis define dois tipos de nós: *master* e *slave*
- Cada nó *slave* mantém uma cópia dos dados de um *master*.



Como funciona a divisão de dados?

- O Redis divide os dados em *slots* chamado de hash *slots*
- Cada dado é armazenado em um *hash slot*
- O Redis suporta até 16.384 *slots*
- Cada nó do Redis é responsável por um subconjunto de *slots*

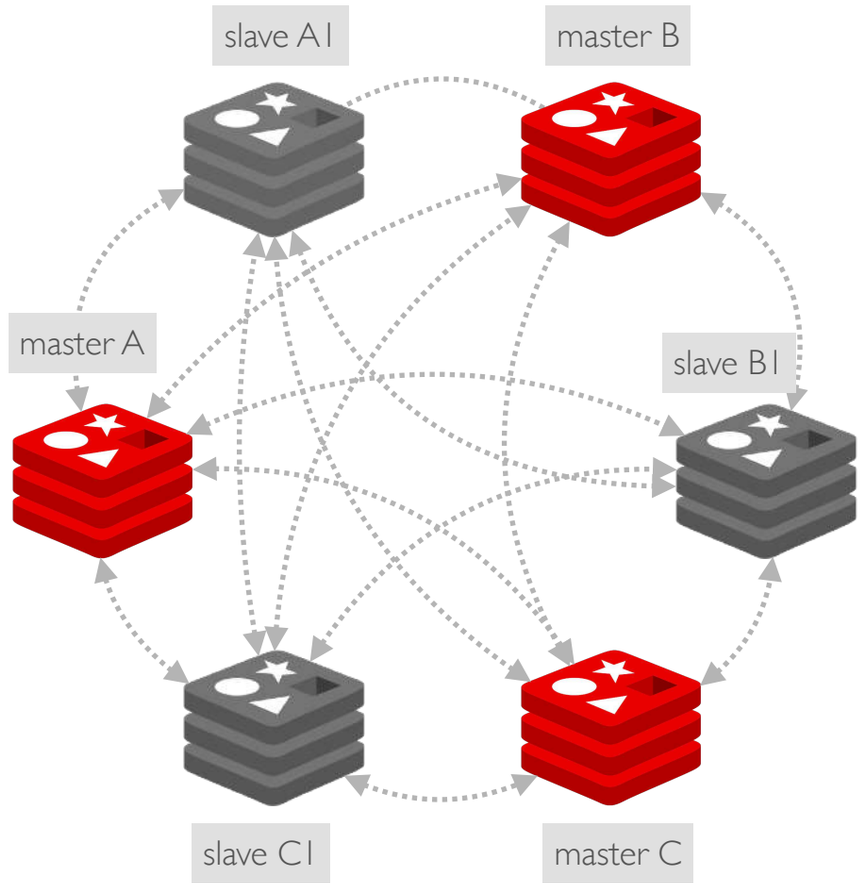


Como funciona a divisão de dados?

- Ex: Caso haja três nós master (A, B e C) cada nó ficará responsável por um conjunto de hash, por exemplo:

  A - 0 até 5500
  B - 5501 até 11000
  C - 11001 até 16383

- É possível incluir novos nós. Nesse caso o Redis redistribui os dados.



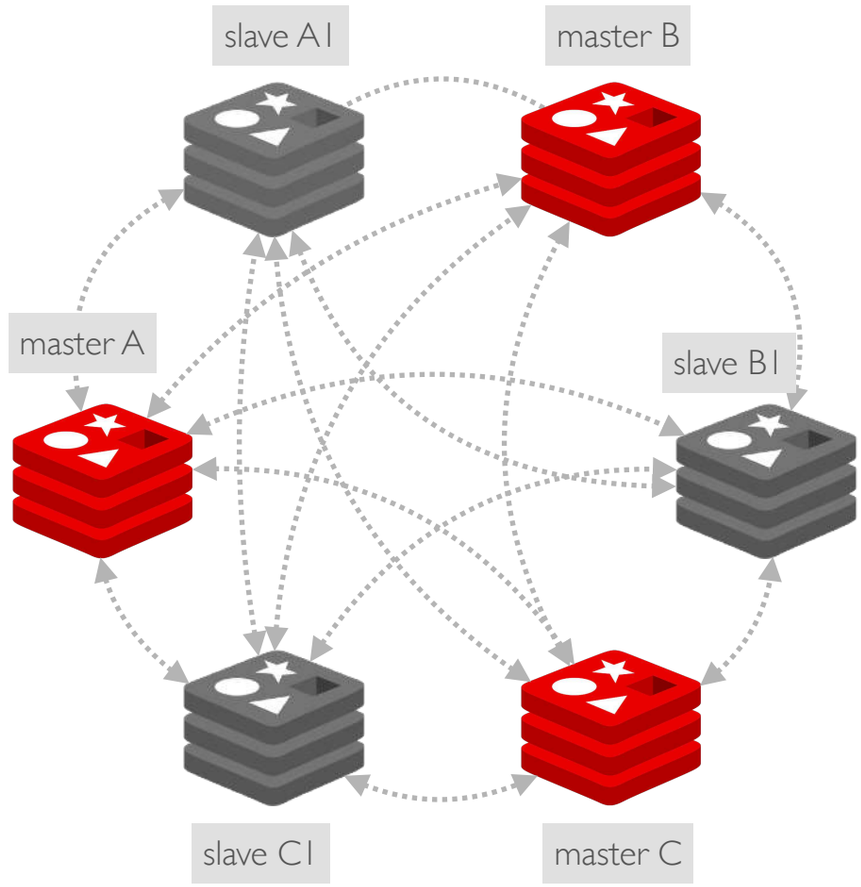
Como saber qual slot?

- Primeiro se computa o hash da chave (crc16) e em seguida faz a operação de módulo (resto da divisão) por 16384

CRC16(key) % 16383

Como funciona a divisão de dados?

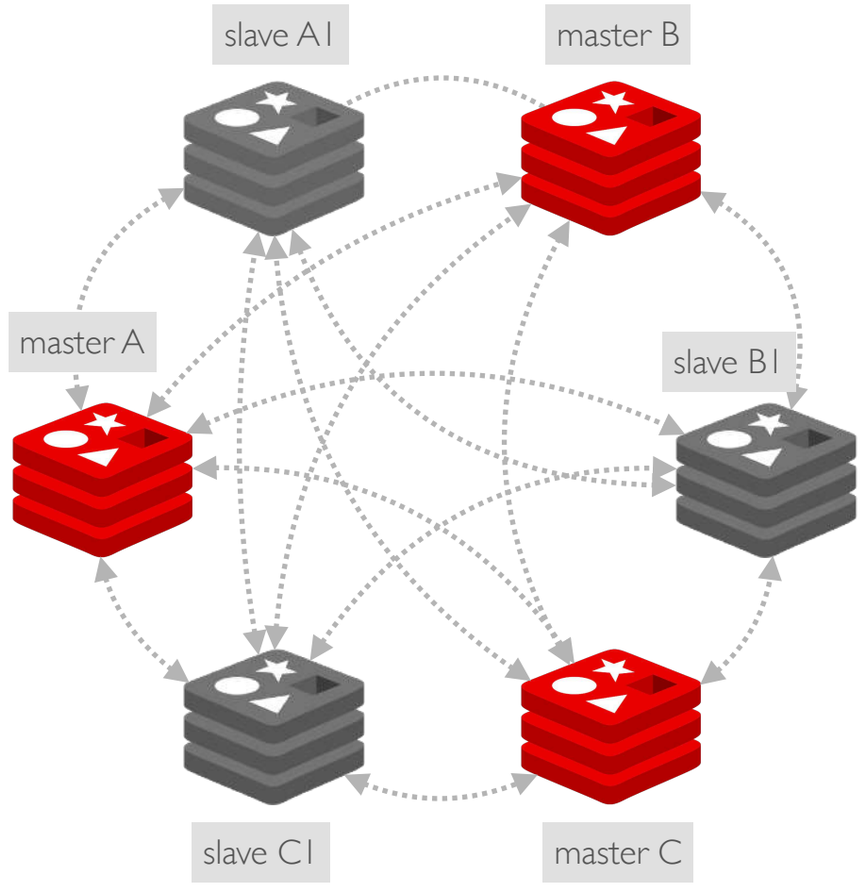
- Caso o nó A venha a cair, o seu *slave* será promovido a master mantendo o cluster operando normalmente.
- É possível incluir novos nós. Nesse caso o Redis redistribui os dados.
- Se o nó A e seu slave cairem, o cluster para de operar



redis

E a consistência dos dados?

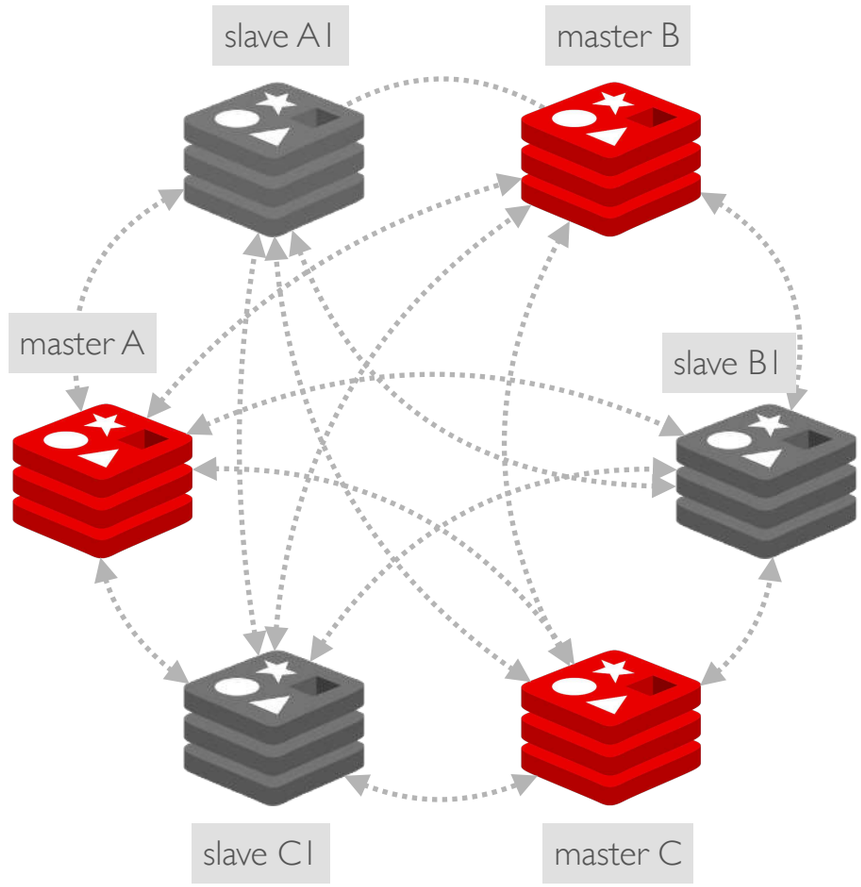
- Redis não garante consistência forte. Na prática isso significa que em algumas circunstâncias o Redis pode "perder" algumas escritas que foram retornado como sucesso ao cliente.
- Isso ocorre por que a replicação do dado é feita de forma assíncrona.
- Caso o Redis esperasse a replicação, as escritas teriam latência adicional e proporcionalmente maior a quantidade de réplicas.



E a consistência dos dados?

- Cenário 1 de falha

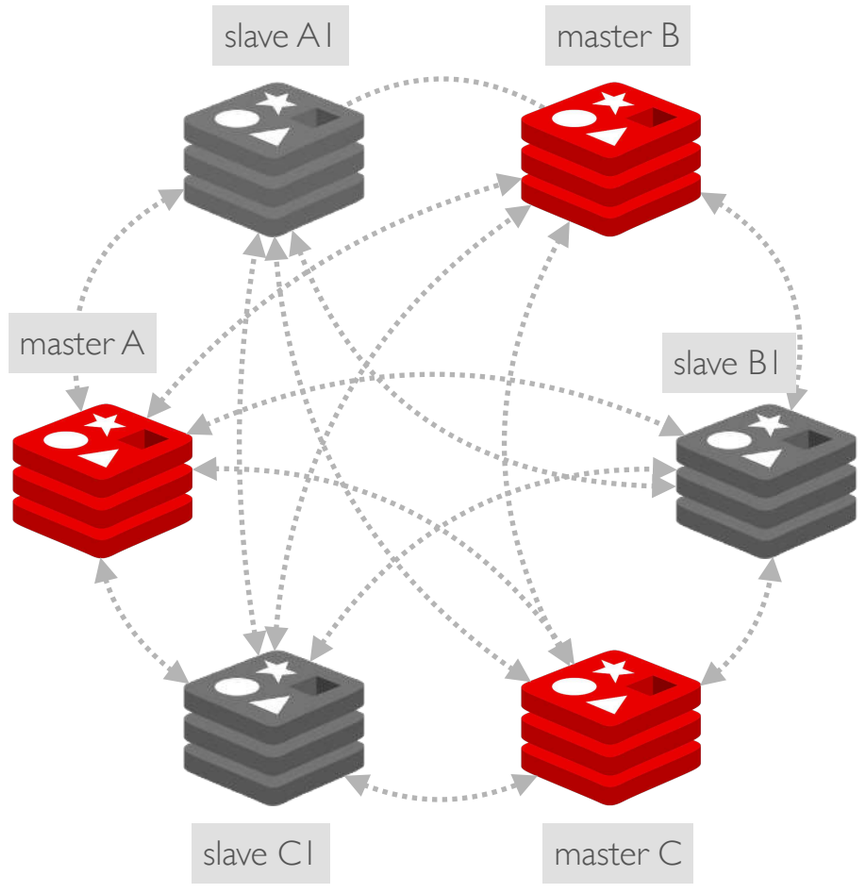
1. Cliente escreve no master B
2. B retorna sucesso
3. B falha (*crash*) antes de enviar o dado para seu slave (B1)
4. B1 é eleito master e o dado é perdido para sempre.



E a consistência dos dados?

- O Redis Cluster tem suporte para gravações síncronas quando absolutamente necessário, implementado por meio do comando WAIT, o que torna muito menos provável a perda de gravações.

No entanto, o Redis Cluster não implementa consistência forte mesmo quando a replicação síncrona é usada: sempre é possível em ambientes mais complexos, cenários de falha em que um escravo que não foi capaz de receber a gravação é eleito como mestre.



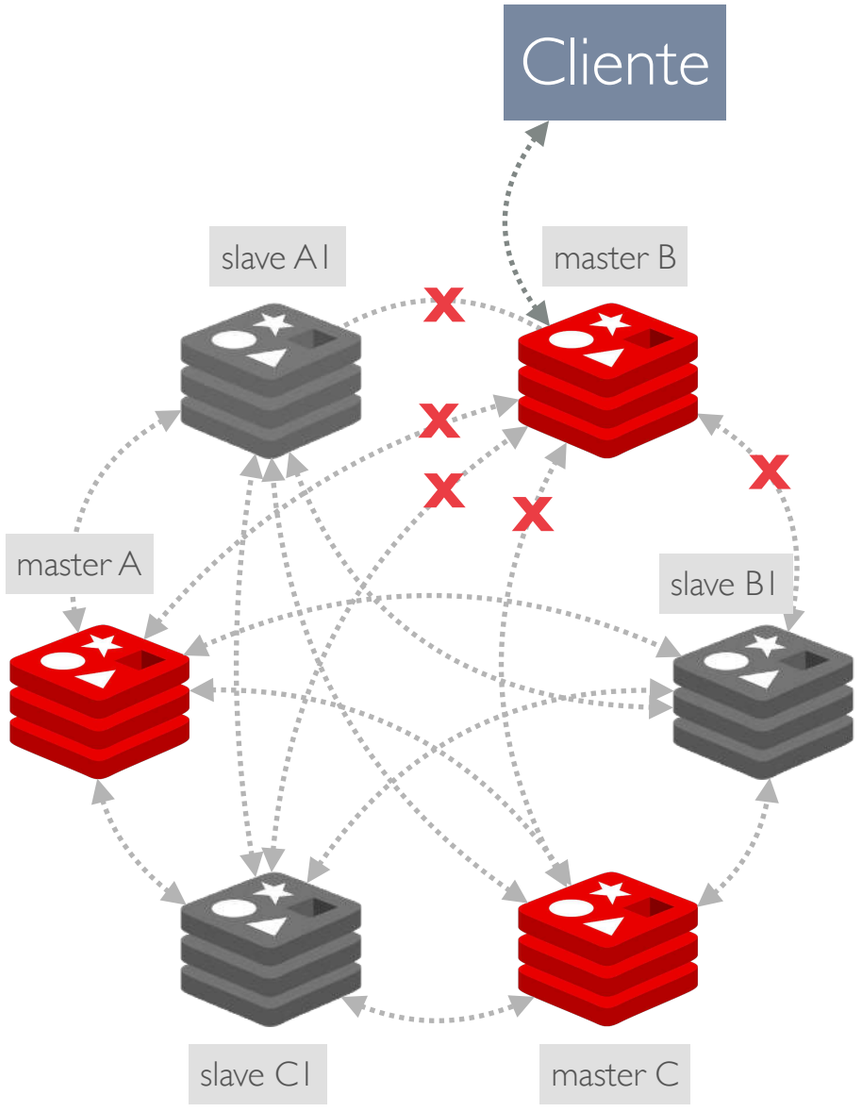
redis

E a consistência dos dados?

- Cenário 2 de falha - Partição da rede

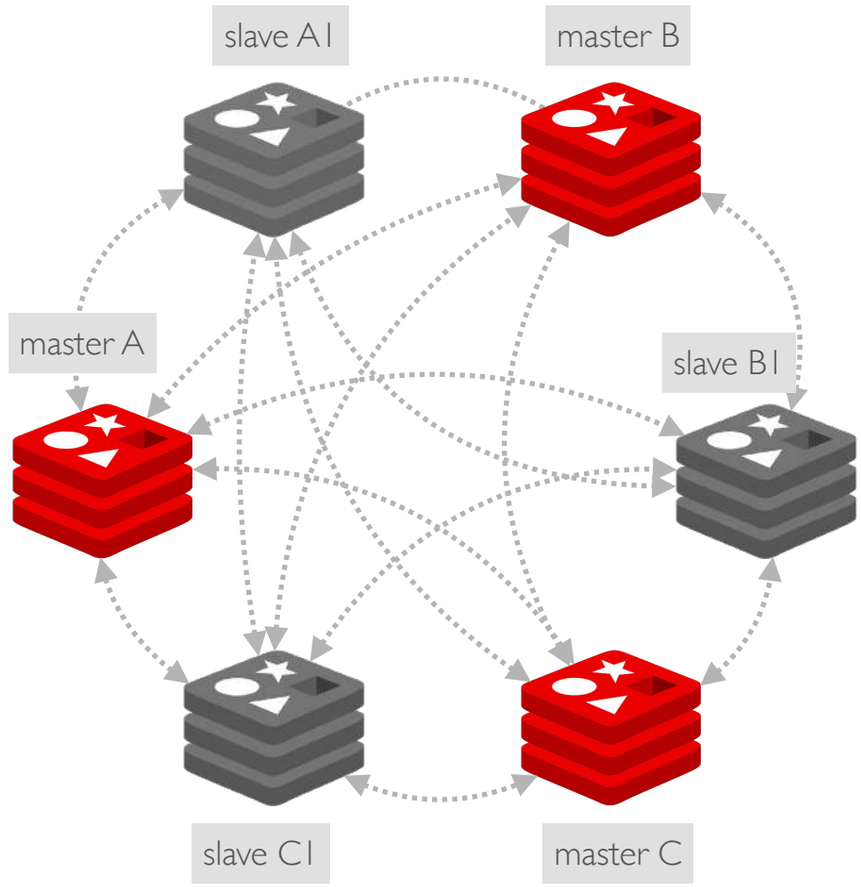
1. A rede é particionada ficando dois lados isolados
2. Em um lado fica o cliente e o master B
3. Em Outro lado fica A, A1, B1, C e C1
4. Cliente escreve em B que aceitará a escrita.

Se a partição durar tempo suficiente para que B1 seja promovido para master no lado majoritário da partição, as gravações do cliente enviadas para B serão perdidas.



E a consistência dos dados?

- Cenário 2 de falha - Partição da rede
- Há um tempo máximo que um master aceita escritas sem comunicação.
- Esse tempo é chamado de **node timeout**. O master é considerado em falha após decorrido este tempo sem comunicação e pode ser substituído por uma de suas réplicas (no lado majoritário). Após isso, novas escritas ao master em falha retornarão erro.



Alguns detalhes importantes

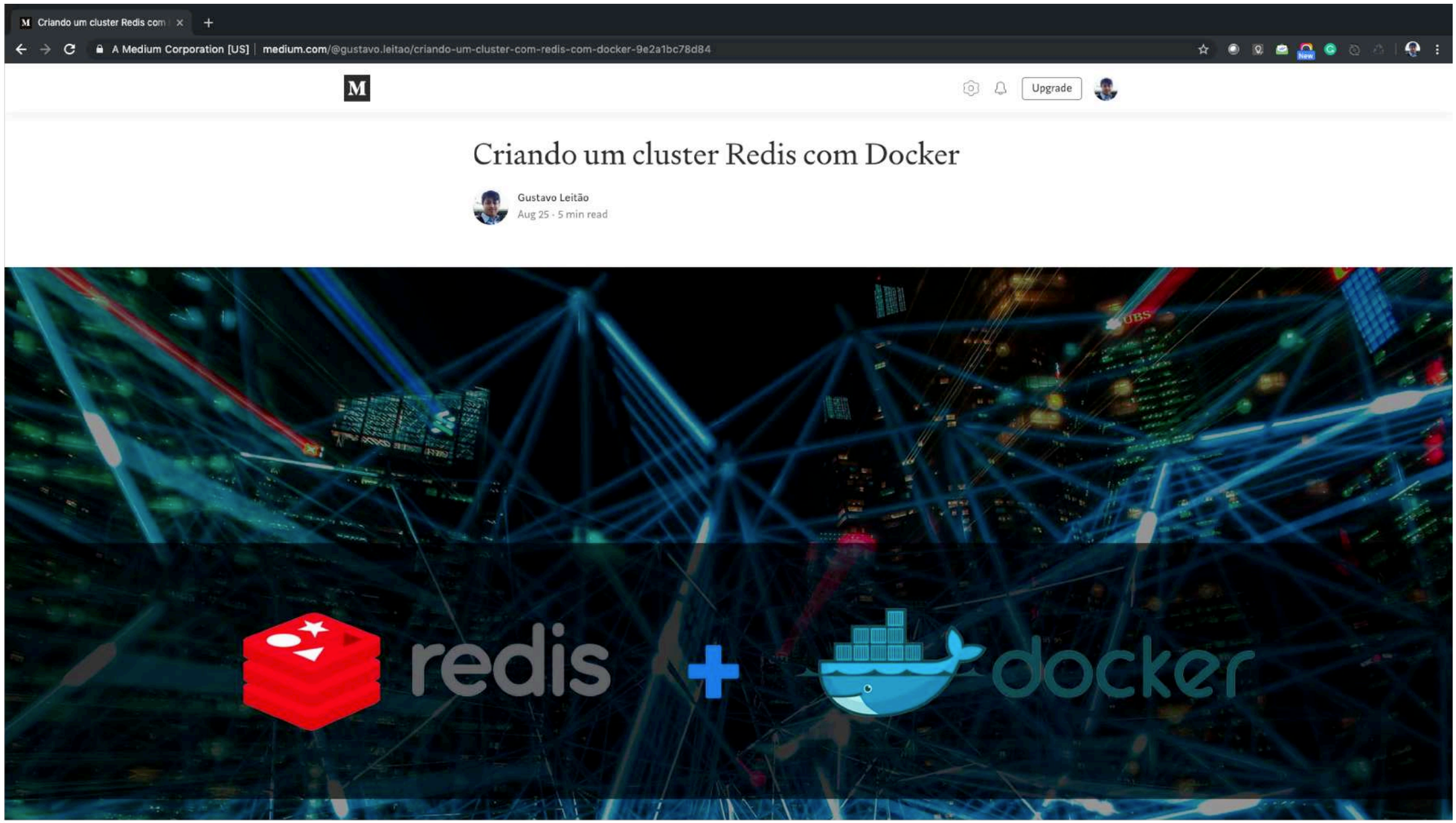
- Redis usa outra porta TCP para orquestração do cluster. A porta adicional será a porta de comunicação com os clientes somada de 10000. Ex. Se o Redis está rodando na porta 6379 a comunicação do cluster ocorrerá na porta 16379.

- O Redis Cluster suporta transações em múltiplas chaves desde que todas as chaves envolvidas pertençam ao mesmo *hash slot*. O usuário pode forçar várias chaves a fazer parte do mesmo *hash slot* através do conceito de *hash tags.*

- Para isso a chave deve possui o seguinte formado "this{foo}key". Apenas o que estiver dentro das chaves será utilizado para *hash*, garantindo que chaves diferentes com {foo} estejam no mesmo *slot*.

Cluster Redis

Vamos à prática?



https://medium.com/@gustavo.leitao/criando-um-cluster-com-redis-com-docker-9e2a1bc78d84

Exercício de Sala

- Crie um grupo de 2 colegas e crie um *cluster* em conjunto.
  - O *cluster* deve ter 3 nós master e 3 nós *slaves*
  - Crie um programa para acessar o *cluster* (Adapte algum criado em aula)
  - Faça alguns testes:
    - Desligue um nó master e veja como o Redis se comporta. O sistema continua funcional?
    - Desligue um *master* e seu *slave*. Como o sistema se comporta?
- Adicione um novo nó master e faça o *reshard*
- Adicione um novo nó slave
- Descomissione um nó master (remova seus slots antes).

https://github.com/luin/ioredis

Obrigado!

Prof. Gustavo Leitão