

**Bootcamp IGTI****Trabalho Prático****Módulo 4****Performance e Otimização****Objetivos**

Exercitar os seguintes conceitos trabalhados no Módulo:

- ✓ Bancos de Dados Distribuídos.
- ✓ Técnicas de Distribuição de Dados.
- ✓ Recursos de segregação de workloads para melhorar performance.

**Enunciado**

Você foi contratado como analista de banco de dados por uma empresa que tem um sistema de banco de dados centralizado, com aplicações monolíticas, e que deseja reconstruir essa aplicação usando bancos de dados distribuídos. Primeiramente, você precisa fazer um laboratório para testar o SGBDD sugerido, o MongoDB, e verificar a sua aderência aos planos da empresa. Como não haverá investimento em infraestrutura neste momento, e você precisa ter agilidade e liberdade na construção dos ambientes, você decidiu usar o Docker para realizar as PoCs (*Proof of Concept* - Provas de Conceito).

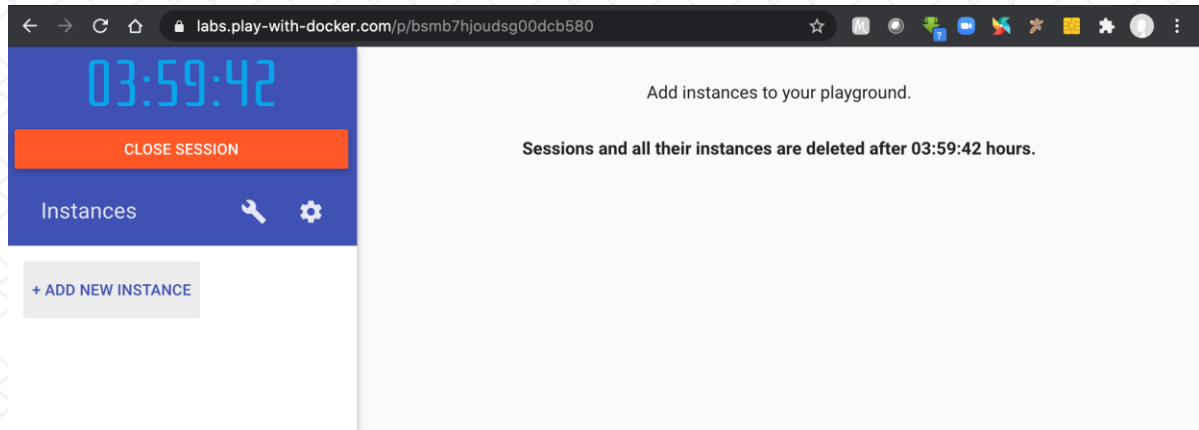
**Atividades**

O aluno deverá desempenhar as seguintes atividades:

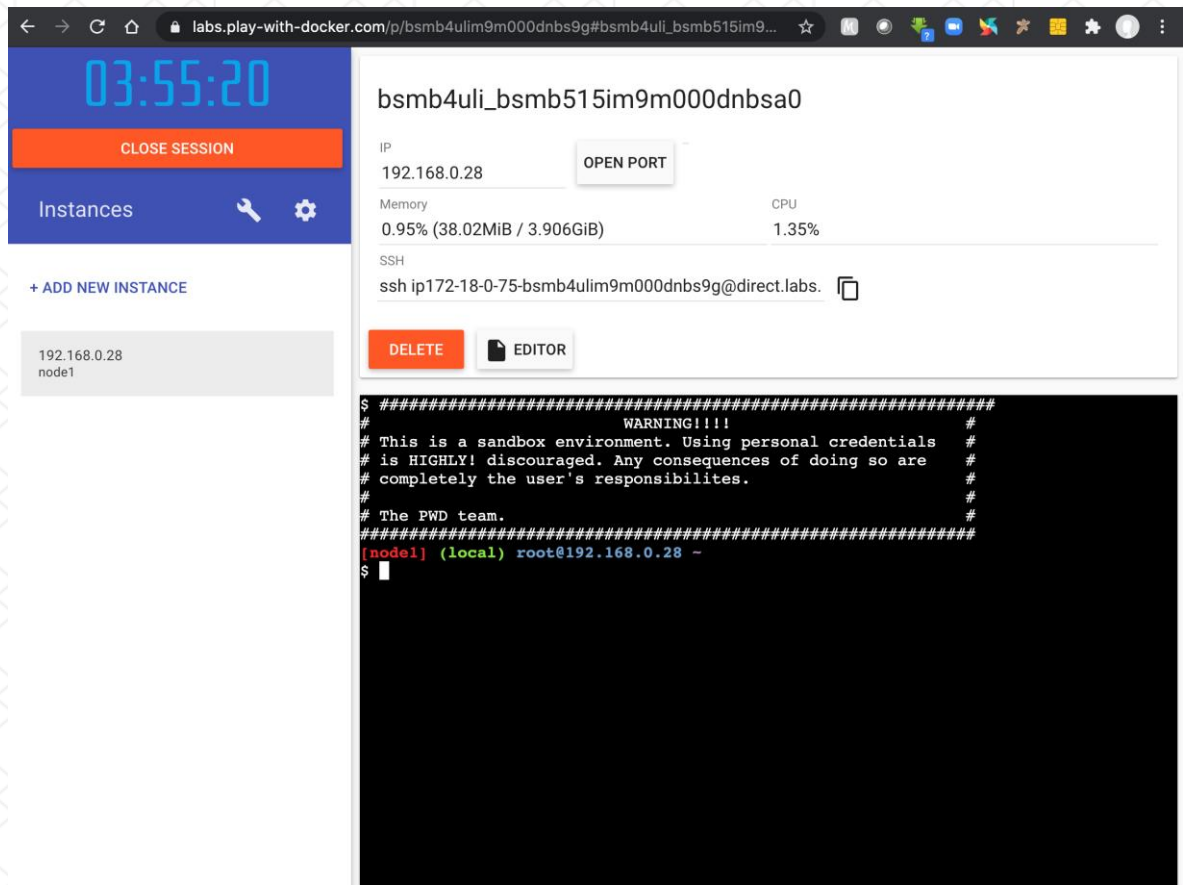
1. O primeiro passo é realizar o download do **Docker Desktop**, disponível em:  
<https://www.docker.com/products/docker-desktop>.

Obs.: aqueles que não desejarem instalar o Docker Desktop, ou que não possuam um equipamento com os requisitos necessários, podem utilizar o serviço gratuito de laboratório da Docker na nuvem, o **Play With Docker** (<https://labs.play-with-docker.com>).

O cadastro é gratuito e pode-se usar 4 horas seguidas de um Docker, por vez. Para instanciar um container Linux, basta clicar em **+ADD NEW INSTANCE**.



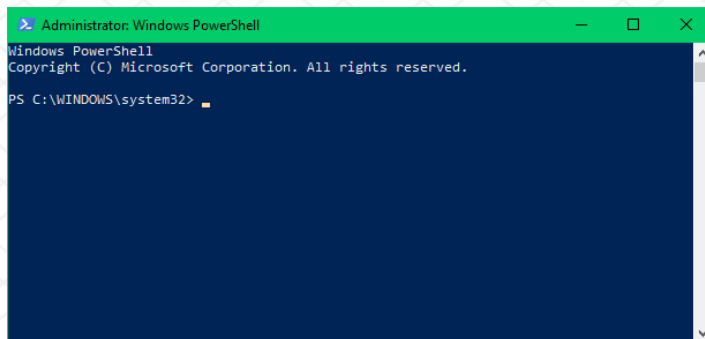
Feito isso, será aberta uma console com o shell habilitado, onde podem ser rodados os comandos a partir do item 5.



2. Para seguir com o Docker Desktop, faça a instalação com as opções default do setup.

**Obs.:** se o seu sistema operacional for o Windows 10 HOME, siga os passos de instalação do link <https://docs.docker.com/docker-for-windows/install-windows-home>.

3. Concluída a instalação, reinicie a máquina.
4. Após o restart, execute o **Windows Power Shell como administrador (no Windows)** ou a console de terminal de linha de comando preferida (no iOS / Linux):



5. Execute, no Power Shell, o comando ***docker images***, para verificar que, após a instalação, não há nenhuma imagem no repositório local do Docker;
6. Procure uma imagem com MongoDB no repositório on-line do Docker, executando o comando ***docker search mongodb*** no Power Shell;
7. Baixe para o repositório local a imagem oficial de nome ***mongo***, que contém sempre a última versão do MongoDB, usando o comando ***docker pull mongo***;
8. Confira que a imagem foi baixada para o repositório local com o comando ***docker images***;
9. Crie um container (servidor) de nome ***lab-btc01*** com essa imagem baixada, usando o comando ***docker run -d --name lab-btc01 mongo***;

**Obs.:** o parâmetro -d permite que o container seja executado em segundo plano.

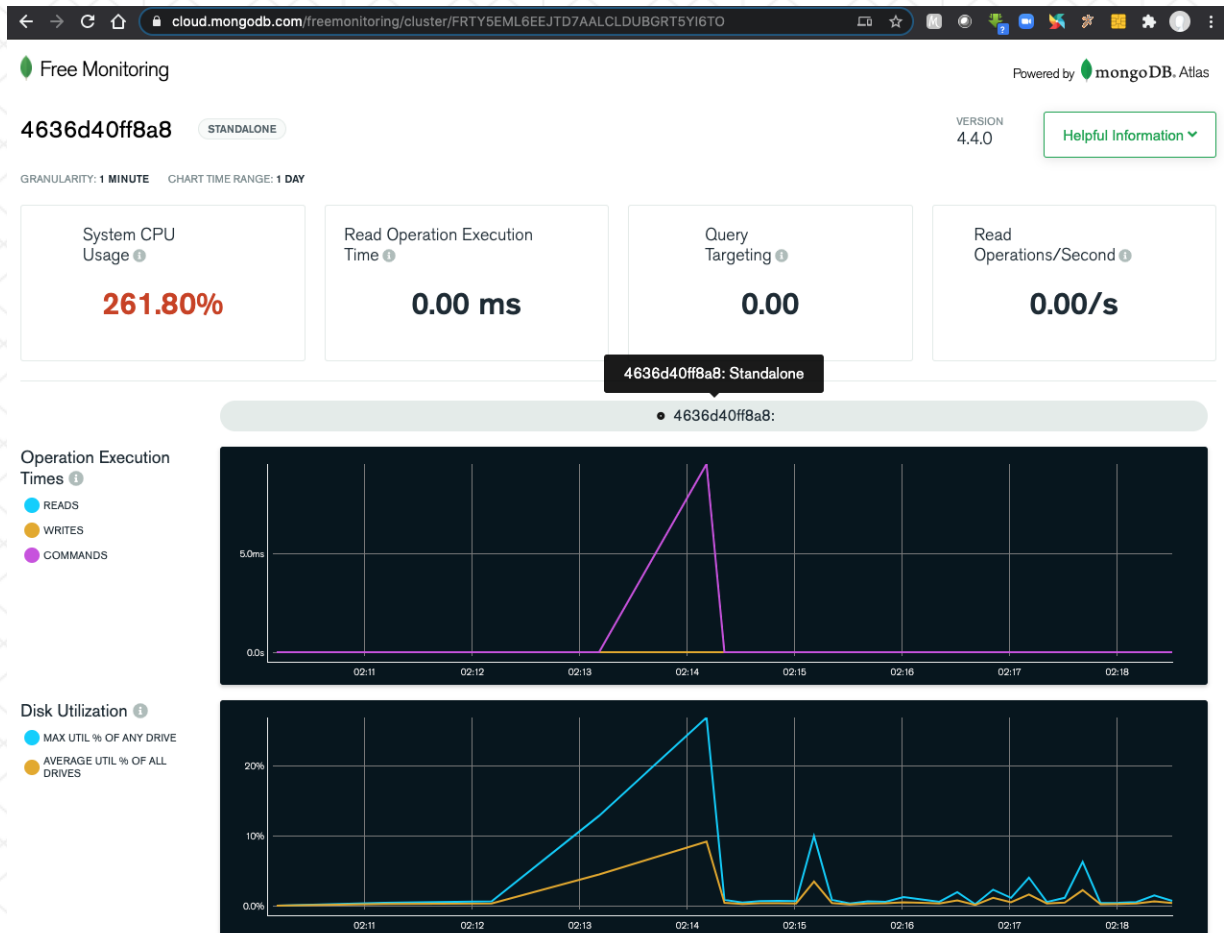
10. Verifique se o container foi criado com sucesso e se está sendo executado com o comando ***docker ps***;



11. Liste as informações do uso de recursos do container com o comando ***docker stats lab-btc01;***
12. Pare o container com o comando ***docker stop lab-btc01;***
13. Liste o status do container com o comando ***docker ps -a;***  
  
**Obs.:** o comando *docker ps*, sem o parâmetro *-a*, só lista os containers em execução.
14. Inicie novamente o container com o comando ***docker start lab-btc01;***
15. Verifique se container foi iniciado com sucesso;
16. Liste o IP associado ao container, usando o comando ***docker inspect --format '{{.NetworkSettings.IPAddress}}' lab-btc01;***
17. Sem entrar no container, liste os file systems que ele contém, usando o comando ***docker exec -it lab-btc01 df -kh;***
18. Entre no *bash* (terminal de linha de comando) do container, usando o comando ***docker exec -it lab-btc01 /bin/bash;***
19. No *bash*, liste o sistema operacional do container usando o comando ***cat /etc/issue;***
20. Ainda no *bash*, faça login no MongoDB do container, usando o comando ***mongo;***

**Obs.:** a instalação do MongoDB existente na imagem não está com o parâmetro de autenticação habilitado e, devido a isso, não há senha para logar no MongoDB quando se está conectado localmente. **Não é recomendado usar essa configuração para ambientes produtivos.**

21. Caso tenha interesse, habilite o serviço (SaaS – Software as a Service) **Free Monitoring**, executando, no MongoDB, o comando **db.enableFreeMonitoring()**. Com a URL que será informada após a execução do comando, é possível acessar o dashboard abaixo, via browser, onde consegue-se visualizar várias métricas acerca do ambiente.



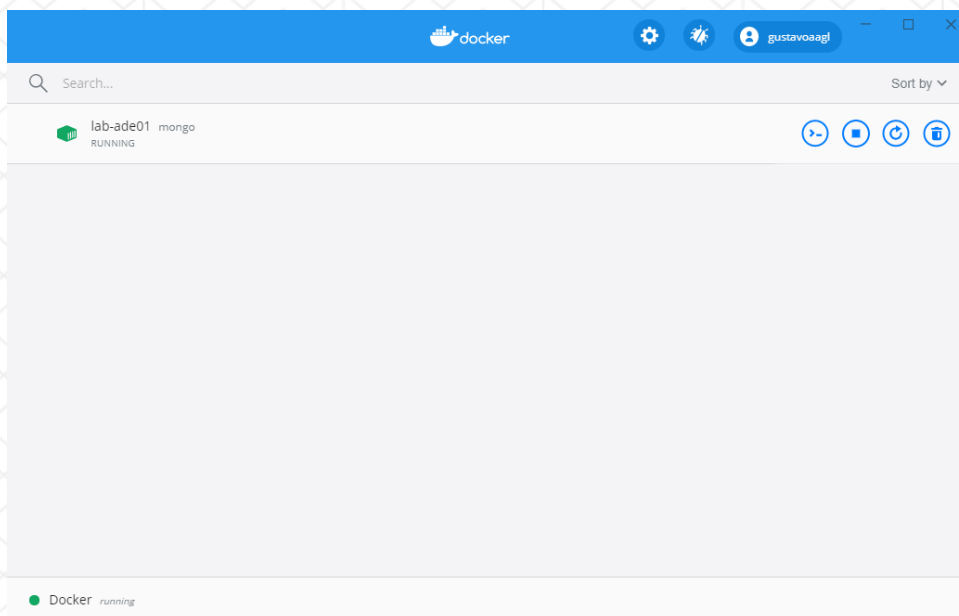
22. No MongoDB, liste os bancos de dados usando o comando **show dbs**, e a versão do SGBD com o comando **db.version()**;

23. Neste ponto da PoC, com um servidor stand alone disponibilizado, começaram os testes de carga e, em determinado momento, precisou-se escalar a camada de banco de dados MongoDB. Sem alterar a topologia da solução (de stand alone para cluster) e a quantidade de servidores, quais são as opções de escalabilidade que você pode aplicar neste cenário?

24. Saia do bash com o comando **exit**;

25. Repita os passos **9 ao 22** para criar 2 novos containers de nome **lab-btc02** e **lab-btc03**, usando a mesma imagem do container lab-btc01;
26. Neste ponto da PoC, com 3 servidores stand alone disponibilizados, quais são as opções de escalabilidade que você pode aplicar para a camada de banco de dados (que até o momento está concentrada no servidor **lab-btc01**) sem alterar a topologia da solução (de stand alone para cluster) e a quantidade de servidores?

**DICA:** na barra de tarefas, clicando com o botão direito sobre o ícone do *Docker* e escolhendo a opção **Dashboard**, é possível abrir a console a seguir, que permite gerenciar os containers existentes na máquina.



- 
27. Agora, vamos iniciar a construção do cluster **Replica Set MongoDB**. Inicialmente, vamos criar uma rede, na qual os nodes deste cluster ficarão. Use o comando **docker network ls** para exibir as redes locais criadas. Certifique-se de que **não existe** uma rede com o nome **net-cluster-mongo**;
28. Crie uma rede de nome **net-cluster-mongo** com o comando **docker network create net-cluster-mongo**;



29. Agora, vamos criar os containers, já iniciando o MongoDB com o parâmetro de Replica Set, e criando uma replica set de nome **lab-btc\_rs0**:

```
docker run -d -p 30001:27017 --name btc-node01 -h btc-node01 --net net-cluster-mongo mongo mongod --replSet lab-btc_rs0
```

```
docker run -d -p 30002:27017 --name btc-node02 -h btc-node02 --net net-cluster-mongo mongo mongod --replSet lab-btc_rs0
```

```
docker run -d -p 30003:27017 --name btc-node03 -h btc-node03 --net net-cluster-mongo mongo mongod --replSet lab-btc_rs0
```

**Obs.:**

- Parâmetro **-p 30001:27017**: expõe a porta 27017 do container, como a porta 30001 no host local;
- Parâmetro **-h**: informa o nome interno do host;
- Parâmetro **-replSet**: instancia o MongoDB já no modo de replicação.

30. Liste os containers que foram criados, assim como seus status, usando o comando **docker ps**;

31. Entre no terminal de linha de comando (bash) dos servidores usando os comandos abaixo (use 3 janelas);

```
docker exec -it btc-node01 /bin/bash  
docker exec -it btc-node02 /bin/bash  
docker exec -it btc-node03 /bin/bash
```

32. Em cada servidor, faça login no MongoDB com o comando **mongo**;

33. Certifique-se, usando o comando **rs.status()**, de que o replica set ainda não foi configurada. O resultado deve ser parecido como o mostrado abaixo:

```

~ — root@btc-node01: / — docker exec -it btc-node01 /bin/bash
> rs.status()
{
  "operationTime" : Timestamp(0, 0),
  "ok" : 0,
  "errmsg" : "no replset config has been received",
  "code" : 94,
  "codeName" : "NotYetInitialized",
  "$clusterTime" : {
    "clusterTime" : Timestamp(0, 0),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}
> █

```

34. Agora, vamos iniciar a configuração do replica set no MongoDB. Dentro do MongoDB, no servidor **btc-node01**, execute o comando abaixo:

```

config={
  _id: "lab-btc_rs0",
  members: [
    { _id: 0, host: "btc-node01:27017" },
    { _id: 1, host: "btc-node02:27017" }
  ]
}

```

35. Confira, executando **config** na linha de comando, que a variável contendo a configuração da replica set está ok. Neste momento, estamos deixando propositalmente de fora o terceiro node.

36. Execute o comando **rs.initiate(config)** para iniciar o replica set.

**Obs.:** pode-se usar o comando **rs.initiate()** para iniciar a replicação na instância em questão, utilizando as configurações padrões, uma vez que não é passada a variável com as configurações do replica set. Nesse caso, apenas a instância em questão (onde o comando é executado) é inserida no replica set.



37. Aguarde uns 30 segundos e execute o comando ***rs.status()*** para conferir o estado da replica set. Provavelmente o servidor ***btc-node01*** estará como ***“PRIMARY”*** e o servidor ***btc-node02*** estará como ***“SECONDARY”***.

38. Ainda no servidor ***btc-node01*** (primário), liste os bancos de dados existentes com o comando ***show dbs***. O resultado deve ser parecido como o mostrado abaixo.

```
[lab-btc_rs0:PRIMARY> show dbs
admin    0.000GB
config   0.000GB
local    0.000GB
[lab-btc_rs0:PRIMARY> _
```

39. Agora, crie um banco de dados de nome ***BDBTC*** e uma collection de nome ***databaseCriado***, usando os comandos abaixo:

***use BDBTC***

***db.databaseCriado.insert({"data\_criacao\_database":new Date(),  
"motivo\_solicitacao":"Laboratório de Bootcamp"});***

40. Liste os bancos criados e para confirmar a criação do banco BDBTC;

41. Para ver as collections criadas, execute o comando abaixo;

***use BDBTC***

***show collections***

42. Agora, liste os documentos existentes na collection criada inicialmente usando os comandos abaixo.

**Obs.:** caso já esteja no banco BDBTC (para conferir você pode executar o comando ***db***), a execução do comando ***use*** é opcional.

***use BDBTC***

***db.databaseCriado.find().pretty()***

43. Agora, no servidor **btc-node02** (secundário), faça login no MongoDB usando o comando **mongo** (caso ainda não tenha feito), e execute o comando **show dbs**. Qual o resultado? Por que isso acontece?

44. Ainda no servidor secundário, execute o comando **rs.slaveOk()** e dispare novamente o comando **show dbs** e os comandos abaixo. Por que a collection e os seus dados estão sendo exibidos neste servidor, uma vez que foram inseridos no MongoDB do servidor **btc-node01**?

**use BDBTC**

**show collections**

**db.databaseCriado.find().pretty()**

45. Agora, ainda no servidor secundário **btc-node02**, tente executar os comandos abaixo para criar uma nova collection e inserir um documento nela. Por que não é possível executar essa operação? Que tipo de replicação temos configurada no ambiente quanto à topologia?

**use BDBTC**

**db.Equipamentos.insert({"nome\_equipamento": 'MAC0001',  
"data\_insercao": new Date() });**

46. Agora saia do MongoDB do servidor secundário **btc-node02** com o comando **exit**;

47. Ainda no servidor **btc-node02**, logue no MongoDB, só que dessa vez usando a string de conexão de replica set, como mostrado abaixo:

**mongo --host "lab-btc\_rs0/btc-node01:27017,btc-node02:27017"**

48. Execute novamente os comandos do item 45 e o comando abaixo. Executou com sucesso e sem precisar executar antes o comando **rs.slaveOk()**? Por quê?

**show collections**

**db.Equipamentos.find().pretty()**

49. Neste ponto da PoC, foi detectado que há um problema de performance devido à falta de recursos para processar todas as operações de **escrita** nesse ambiente de banco de dados. Uma pessoa do time deu a ideia de acionar mais um servidor no replica set MongoDB usando o comando ***rs.add("btc-node03:27017")***. Essa ação irá resolver o problema? Em caso negativo, e considerando que não se pode alterar a configuração do replica set MongoDB, qual tipo de escalabilidade pode ser feita para tentar amenizar o problema? Explique sua resposta.

50. Execute o comando abaixo para incluir o terceiro servidor no ambiente e depois execute o comando ***rs.status()*** para verificar se ele foi adicionado com sucesso ao replica set do MongoDB;

```
rs.add("btc-node03:27017")
```

51. Identifique, com o comando ***rs.status()***, qual o servidor está atualmente como primário ("**PRIMARY**"). Saia do MongoDB e do bash do servidor e pare o container desse servidor com o comando ***docker stop NOME\_DO\_SERVIDOR***. Execute o comando ***docker ps*** para verificar que apenas 2 servidores estão em execução;

52. Agora, logue no MongoDB, a partir do shell de outro servidor, usando a string de conexão abaixo:

```
mongo --host "lab-btc_rs0/btc-node01:27017,btc-node02:27017,btc-node03:27017"
```

- Qual servidor assumiu como primário?
- Qual servidor está como secundário?
- Há algum servidor que não está disponível no replica set?

53. Inicie novamente o container que havia parado no passo 51, e verifique o status do replica set com o comando ***rs.status()***;

54. Passado algum tempo, foi identificado um novo problema de performance devido à falta de recursos, só que agora para processar as operações de **leitura** no banco de dados. Como foi informado que as aplicações utilizam a string de conexão abaixo, o



DBA sugeriu adicionar mais 2 réplicas secundárias no ambiente, de forma a escalar os recursos para as operações de leitura. Isso irá resolver o problema? Explique sua resposta.

```
mongo --host "lab-btc_rs0/btc-node01:27017,btc-node02:27017,btc-node03:27017  
readPreference:secondaryPreferred"
```

55. Se tratando de um ambiente crítico, que precisa de alta disponibilidade e de ser escalável tanto horizontalmente quanto verticalmente, seja para operações de leitura ou de escrita, o que você, como um profissional que cursou o Bootcamp Analista de Banco de Dados do IGTI, pode sugerir de melhoria?

## Respostas Finais

Os alunos deverão desenvolver a prática e, depois, responder às seguintes questões objetivas: