

Pós-graduação

Engenharia de Dados / Big Data

Aluno: Charles de Araujo Melo

Infnet - Trabalho Infraestrutura MongoDB

1. Descreva com suas palavras 3 principais diferenças e 3 semelhanças entre bases de dados SQL e bases noSQL.

R. O grande diferencial dos bancos de dados NoSQL é que toda a informação é agrupada e guardada no mesmo registro e os dados podem ser armazenados em documentos, colunas, key values e gráficos. Já no SQL você precisa ter o relacionamento entre várias tabelas para ter a informação, informação esta disposta no modelo entidade e relacionamento.

As bases de dados SQL utilizam uma abordagem de escala vertical, o que significa que escalam adicionando mais poder ao servidor. As bases de dados NoSQL utilizam uma abordagem de escala horizontal, o que significa que são escaladas adicionando mais servidores.

As duas possuem as quatro principais operações realizadas em banco de dado, CRUD.

Create (criação), read (leitura), update (atualização) e delete (exclusão).

2. Em sua opinião as bases noSQL são melhores ou piores que as bases SQL? Em que elas se destacam? Justifique sua resposta.

R. Não existe nenhum melhor que o outro, cada um tem suas vantagens e desvantagens em cada cenário. O NoSQL é mais indicado para aqueles sistemas que tenham necessidades maiores de armazenamento e desempenho com dados não estruturados. Já o SQL é indicado quando necessitamos de maior consistência e transações retornando várias colunas.

3. Descreva com suas palavras um cenário onde você optaria por um modelo noSQL e um cenário onde você optaria por um modelo SQL.

R. Modelo SQL optaria em aplicações que requerem transações de várias linhas como sistemas de contabilidades ou sistemas que monitoram inventário ou rodam em

sistemas legados vão prosperar com a estrutura MySQL.

Modelo noSQL é uma boa escolha para negócios que têm crescimento rápido ou bases de dados sem definições claras de esquemas como costuma ser o caso de apps mobile, análises em tempo real e sistemas de gerenciamento de conteúdo.

4. Além do material usado em aula, faça uma pequena pesquisa e descreva com suas palavras quais são os tipos de bases noSQL existentes no mercado atual, com as principais características de uso cada um tipo. Cite as fontes da pesquisa

R. **MongoDB:** é, possivelmente, o SGBD NoSQL mais popular. Opera no **modelo de documentos** e é extremamente fácil de usar. Além de multiplataforma, o MongoDB é totalmente gratuito, o que ajuda bastante a aumentar a sua popularidade;

Cassandra: atualmente administrado pela Fundação Apache, o banco de dados Cassandra que opera no **formato de colunas**. Totalmente gratuito, essa tecnologia NoSQL é extremamente rápida e escalável, o que se torna uma ótima opção de mercado.

Neo4j: O Neo4j como o SGBD NoSQL no **modelo de grafos** mais popular do mercado, que se destaca pela velocidade de leitura e gravação dos dados de forma totalmente escalável, facilidade de aprendizado e uso, além de uma amigável modelagem de dados para o negócio. Ou seja, é uma tecnologia extremamente poderosa e amigável para a equipe de desenvolvimento.

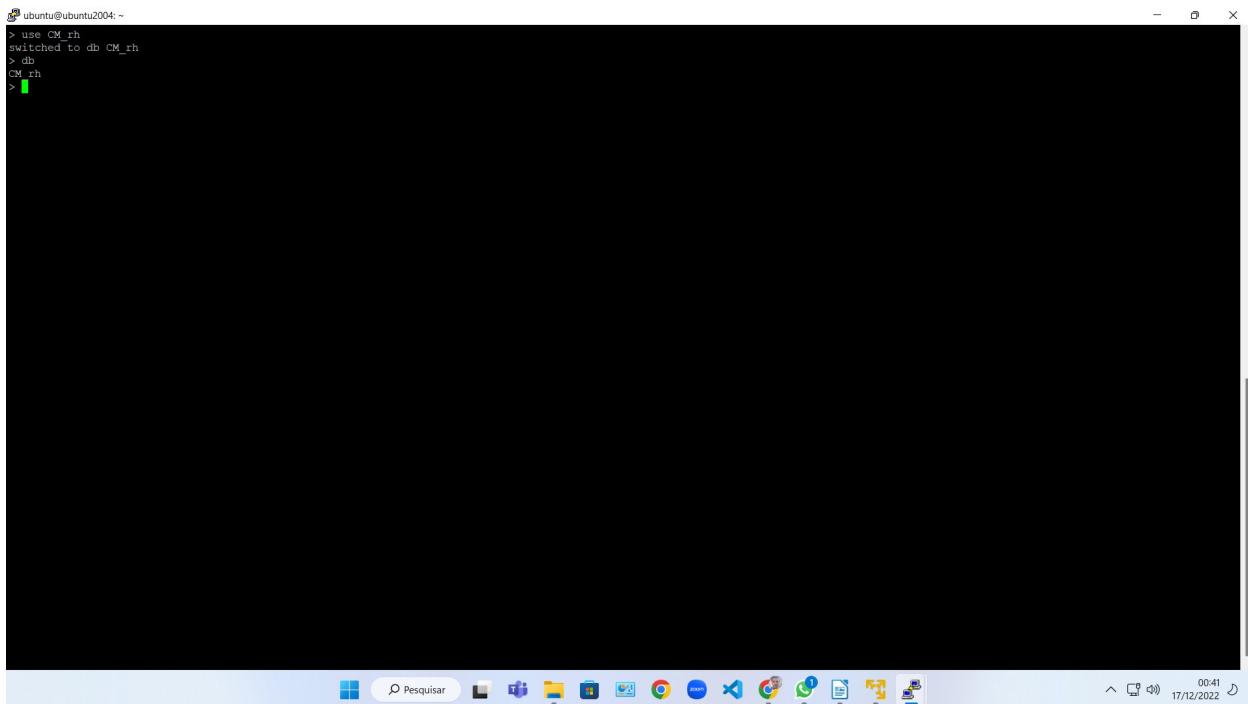
Redis: quando o assunto é banco de dados valores-chave, o Redis é o nome mais forte no quesito. Sua principal característica é a velocidade, muito útil para tratar um grande volume de dados em aplicações com muitas transações.

Fonte: <https://logap.com.br/blog/banco-de-dados-nosql-para-iniciantes/>

Parte 2 – Prática

1. Criar o database "&&_rh".

```
use CM_rh
```



```
ubuntu@ubuntu2004: ~
> use CM_rh
switched to db CM_rh
> db
CM_rh
>
```

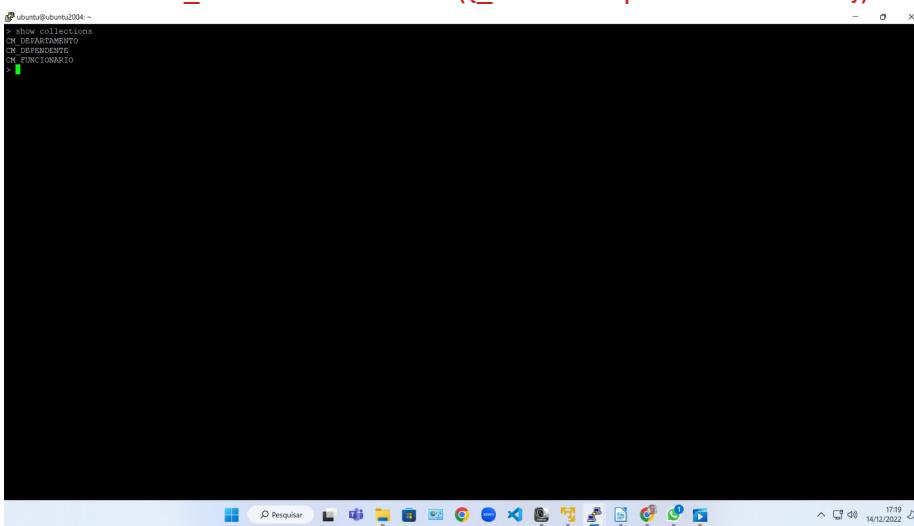
The screenshot shows a terminal window on an Ubuntu 20.04 desktop. The terminal command 'use CM_rh' has been run, switching the database to 'CM_rh'. The window title bar shows the terminal icon and the session name 'ubuntu@ubuntu2004: ~'. The desktop taskbar at the bottom includes icons for Pesquisar (Search), File Explorer, Task View, Google Chrome, Microsoft Edge, and others.

2. Criar as coleções: "&&_DEPARTAMENTO", "&&_FUNCIONARIO", "&&_DEPENDENTE".

```
db.CM_FUNCIONARIO.insert({_id:"01" nome:"Carolina", salario:"1000"})
```

```
db.CM_DEPARTAMENTO.insert({_id:"01" departamento:"Juridico"})
```

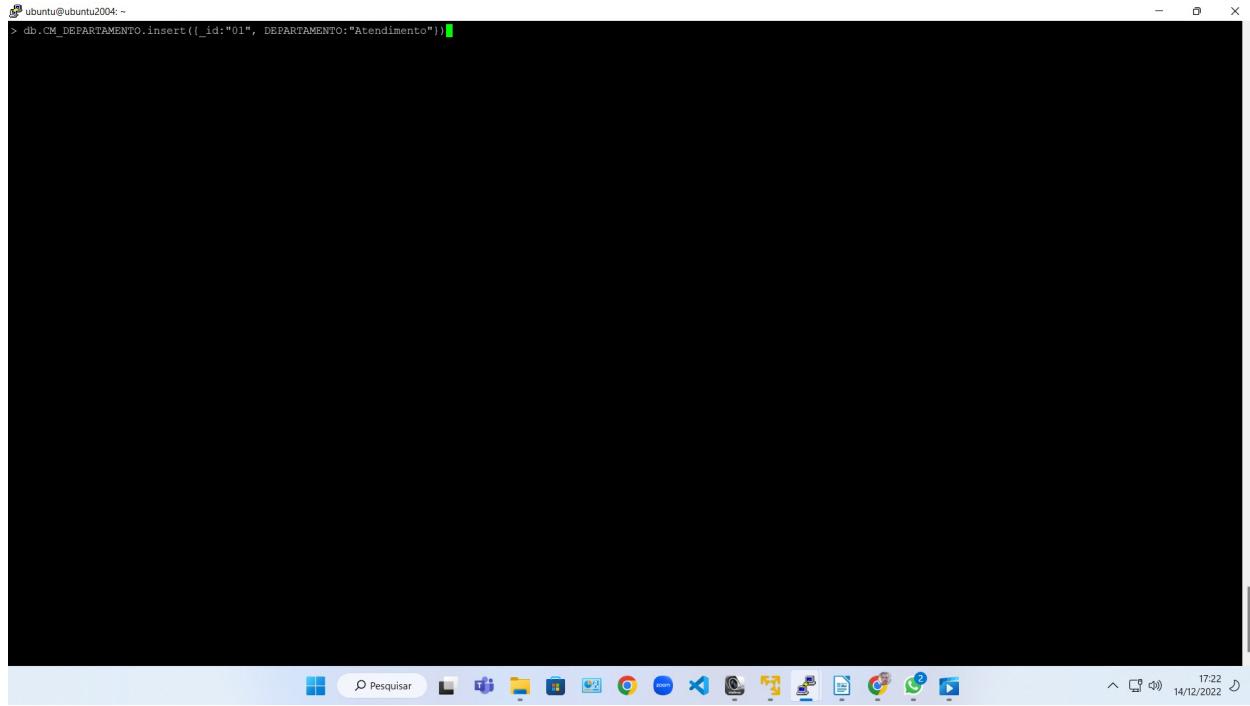
```
db.CM_DEPENDENTE.insert({_id:"01" dependente:"Melo"})
```



```
ubuntu@ubuntu2004: ~
> show collections
CM_DEPARTAMENTO
CM_DEPENDENTE
CM_FUNCIONARIO
>
```

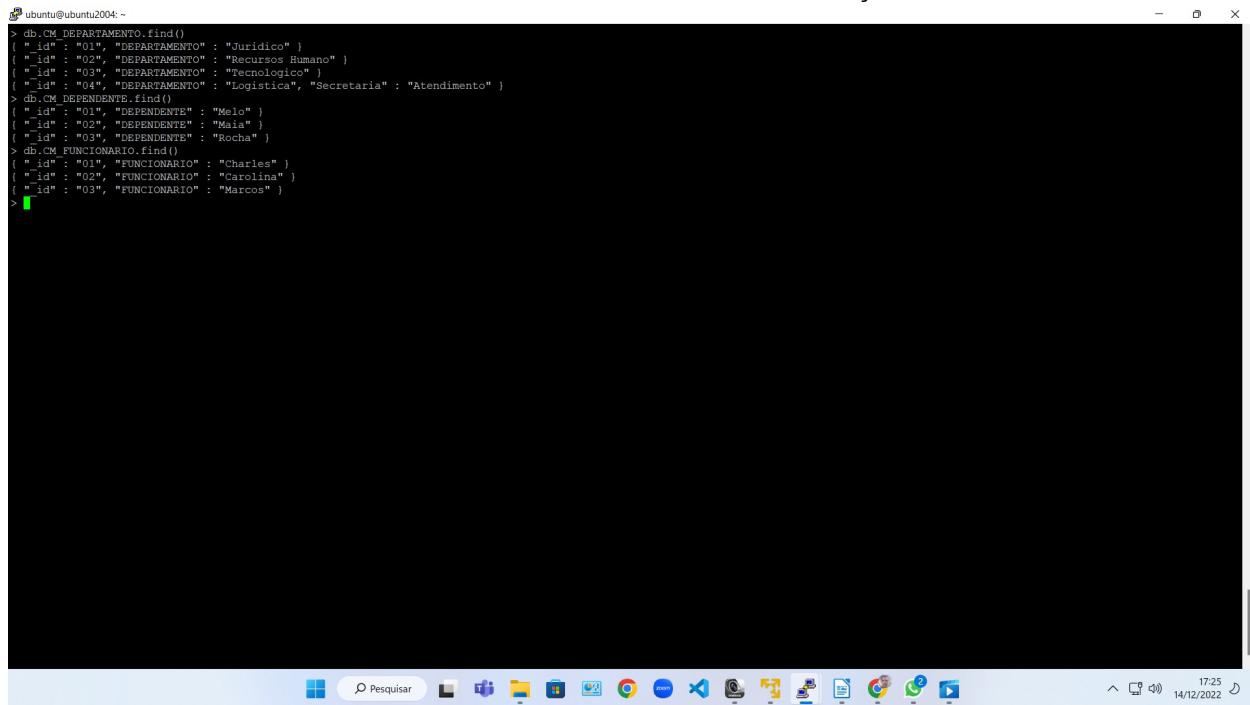
The screenshot shows a terminal window on an Ubuntu 20.04 desktop. The command 'show collections' has been run, displaying the three collections created: 'CM_DEPARTAMENTO', 'CM_DEPENDENTE', and 'CM_FUNCIONARIO'. The window title bar shows the terminal icon and the session name 'ubuntu@ubuntu2004: ~'. The desktop taskbar at the bottom includes icons for Pesquisar (Search), File Explorer, Task View, Google Chrome, Microsoft Edge, and others.

3. Inserir 3 documentos para cada coleção.



```
ubuntu@ubuntu2004:~\n> db.CM_DEPARTAMENTO.insert({_id:"01", DEPARTAMENTO:"Atendimento"})\n\n
```

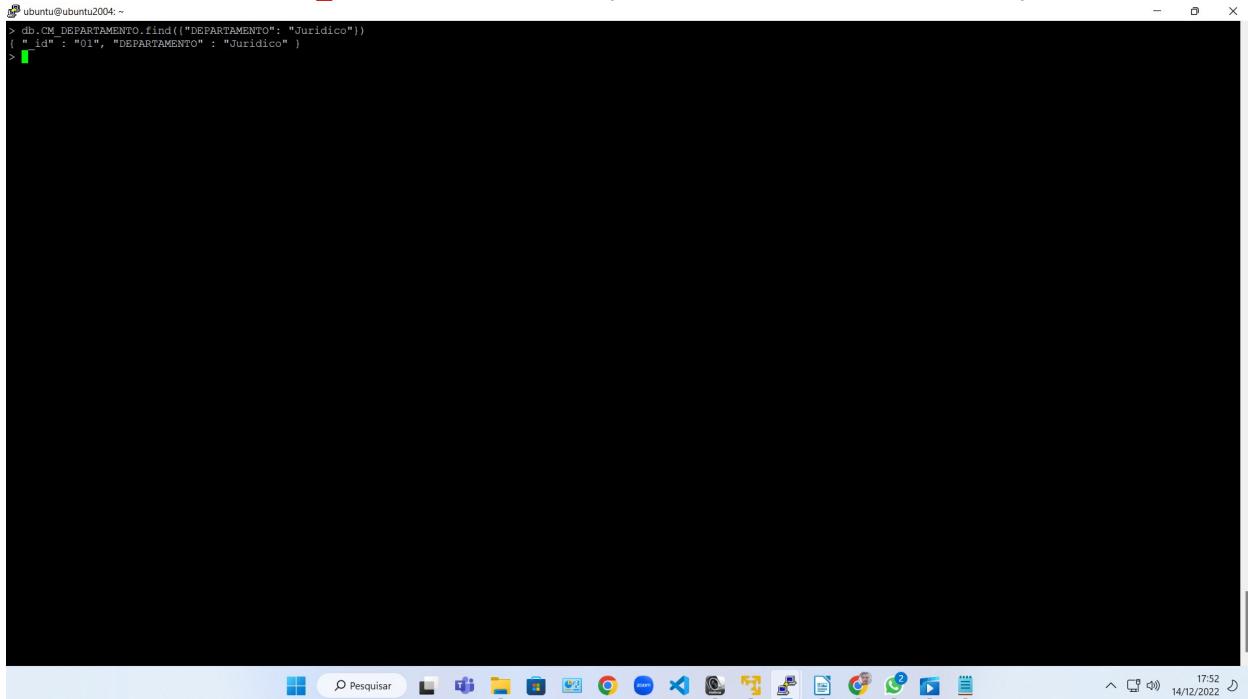
4. Mostrar o conteúdo de cada coleção.



```
ubuntu@ubuntu2004:~\n> db.CM_DEPARTAMENTO.find()\n{ "_id" : "01", "DEPARTAMENTO" : "Juridico" }\n{ "_id" : "02", "DEPARTAMENTO" : "Recursos Humano" }\n{ "_id" : "03", "DEPARTAMENTO" : "Tecnologico" }\n{ "_id" : "04", "DEPARTAMENTO" : "Logistica", "Secretaria" : "Atendimento" }\n> db.CM_DEPENDENTE.find()\n{ "_id" : "01", "DEPENDENTE" : "Melo" }\n{ "_id" : "02", "DEPENDENTE" : "Maia" }\n{ "_id" : "03", "DEPENDENTE" : "Rocha" }\n> db.CM_FUNCIONARIO.find()\n{ "_id" : "01", "FUNCIONARIO" : "Charles" }\n{ "_id" : "02", "FUNCIONARIO" : "Carolina" }\n{ "_id" : "03", "FUNCIONARIO" : "Marcos" }\n>
```

5. Para a coleção "&&_DEPARTAMENTO" fazer um filtro baseado numa descrição do departamento (igualdade).

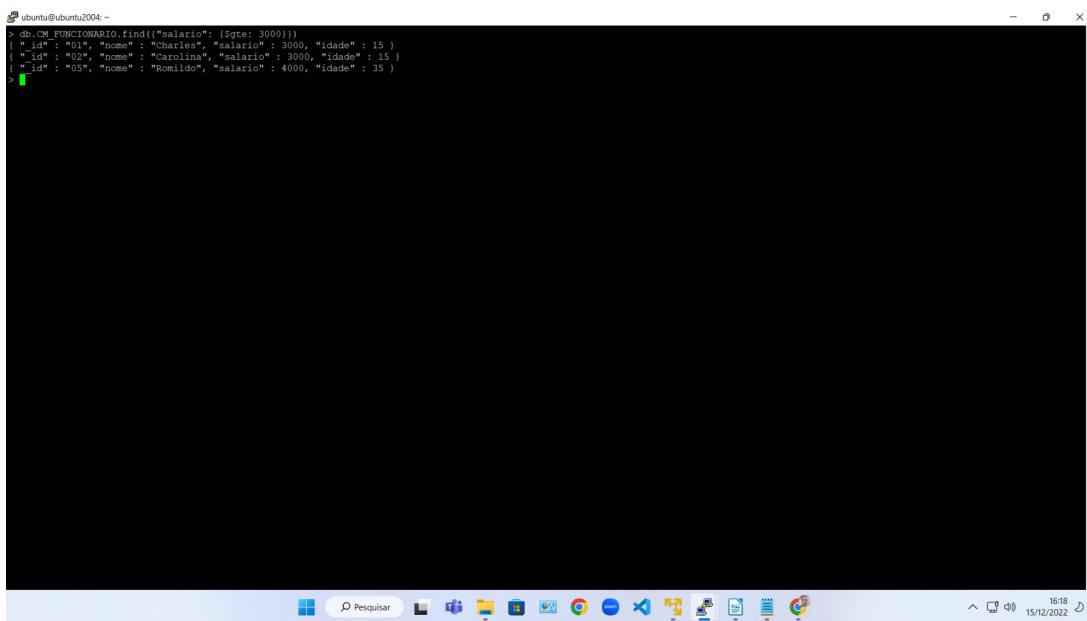
```
db.CM_DEPARTAMENTO.find({"DEPARTAMENTO": "Juridico"})
```



```
ubuntu@ubuntu2004:~> db.CM_DEPARTAMENTO.find({"DEPARTAMENTO": "Juridico"})
{ "id": "01", "DEPARTAMENTO" : "Juridico" }
>
```

6. Para a coleção "&&_FUNCIONARIO" mostrar os funcionários que recebem salário acima de R \$2000,00.

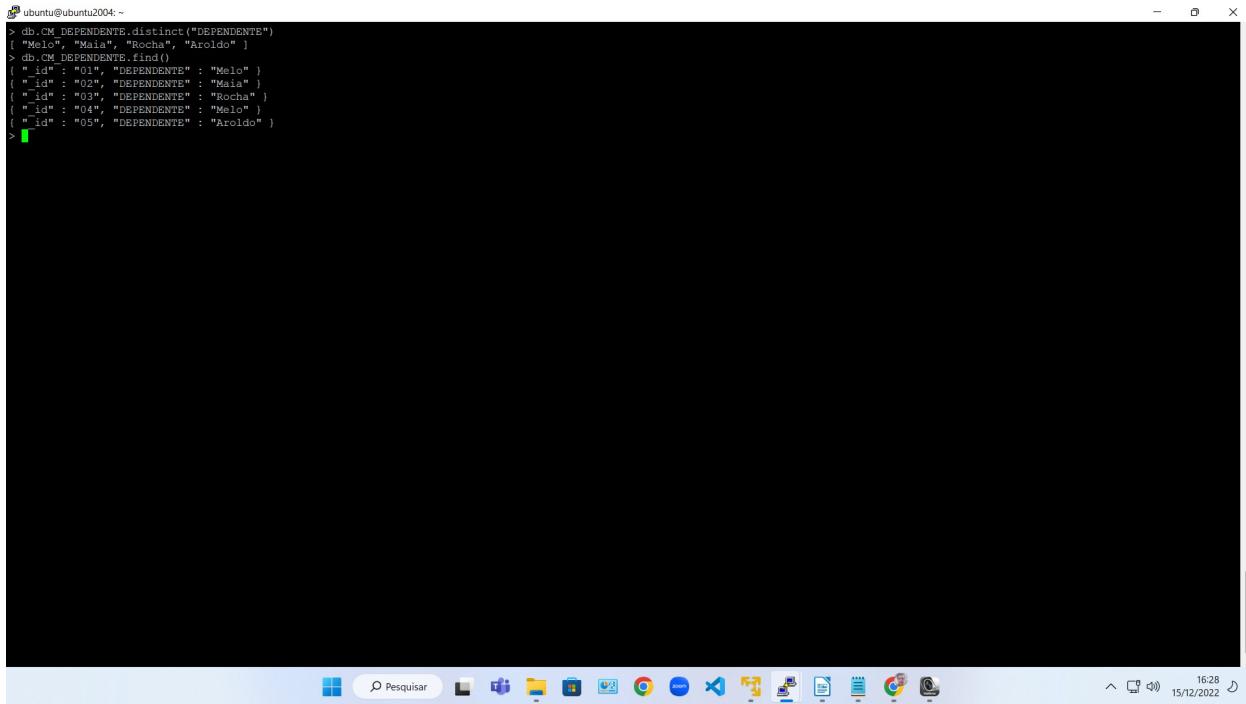
```
R.db.CM_FUNCIONARIO.find({"salario": {$gte: 2000}})
```



```
ubuntu@ubuntu2004:~> db.CM_FUNCIONARIO.find({"salario": {$gte: 3000}})
{ "id": "01", "nome": "Charles", "salario": 3000, "idade": 15 }
{ "id": "02", "nome": "Carolina", "salario": 3000, "idade": 15 }
{ "id": "05", "nome": "Romildo", "salario": 4000, "idade": 35 }
>
```

7. Para a coleção "&&_DEPENDENTE" executar o método distinct para o atributo nome.

```
db.CM_DEPENDENTE.distinct("DEPENDENTE")
```

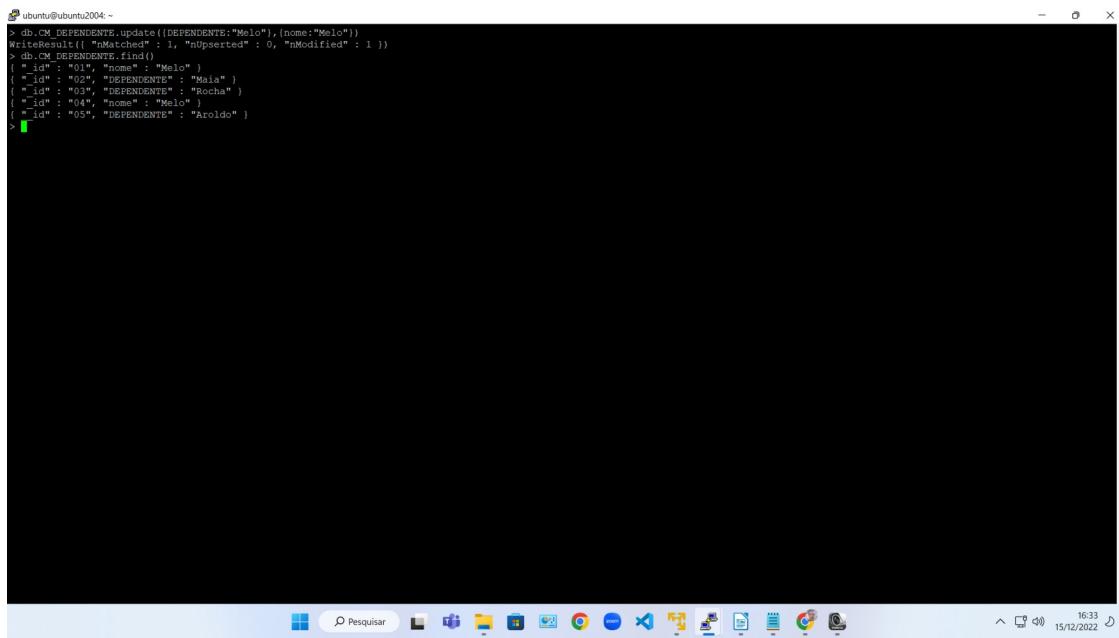


```
ubuntu@ubuntu2004: ~
> db.CM_DEPENDENTE.distinct("DEPENDENTE")
[{"nome": "Melo", "Dependente": "Rocinha", "Aroldo"}, {"nome": "Melo", "Dependente": "Maior", "Aroldo"}]
```

OBS: No caso coloquei o nome de DEPENDENTE

8. Executar um update para uma das coleções.

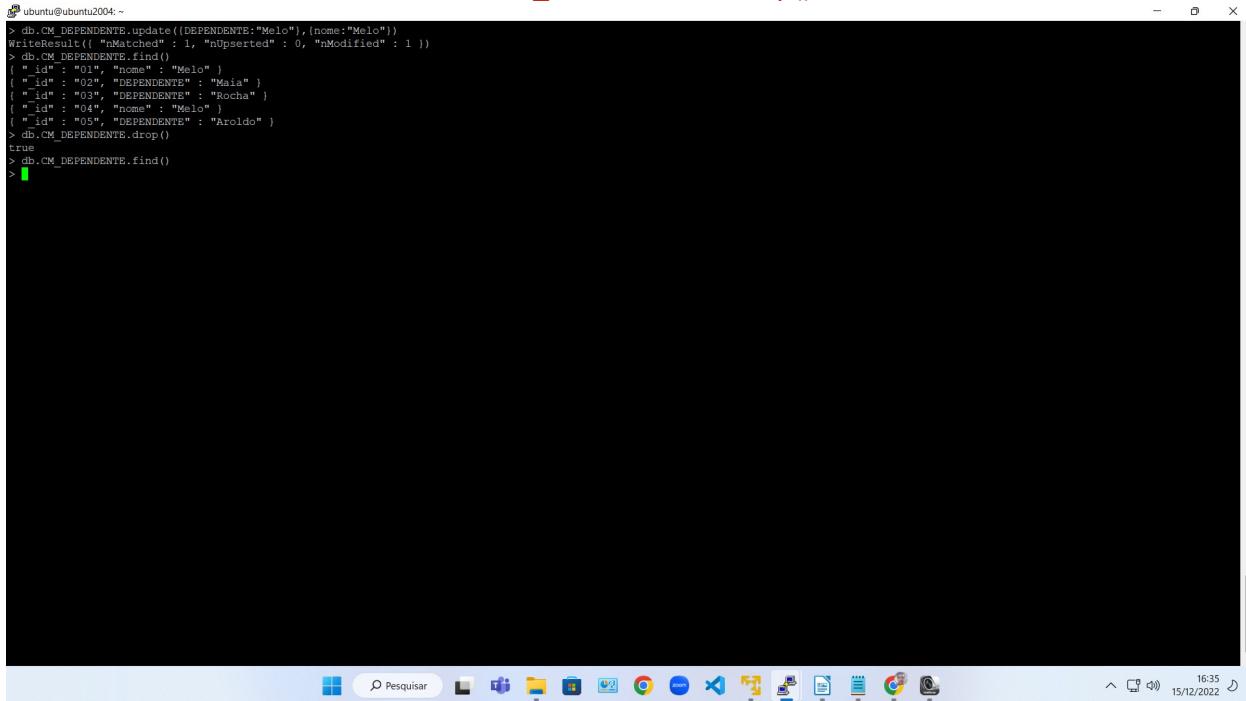
```
db.CM_DEPENDENTE.update({DEPENDENTE:"Melo"},{nome:"Melo"})
```



```
ubuntu@ubuntu2004: ~
> db.CM_DEPENDENTE.update({DEPENDENTE:"Melo"},{nome:"Melo"})
WriteResult({ "nMatched": 1, "nUpserted": 0, "nModified": 1 })
> db.CM_DEPENDENTE.find()
[{"_id": "01", "nome": "Melo"}, {"_id": "02", "DEPENDENTE": "Maior"}, {"_id": "03", "DEPENDENTE": "Rocinha"}, {"_id": "04", "nome": "Melo"}, {"_id": "05", "DEPENDENTE": "Aroldo"}]
```

9. Executar um delete (remove) para uma das coleções.

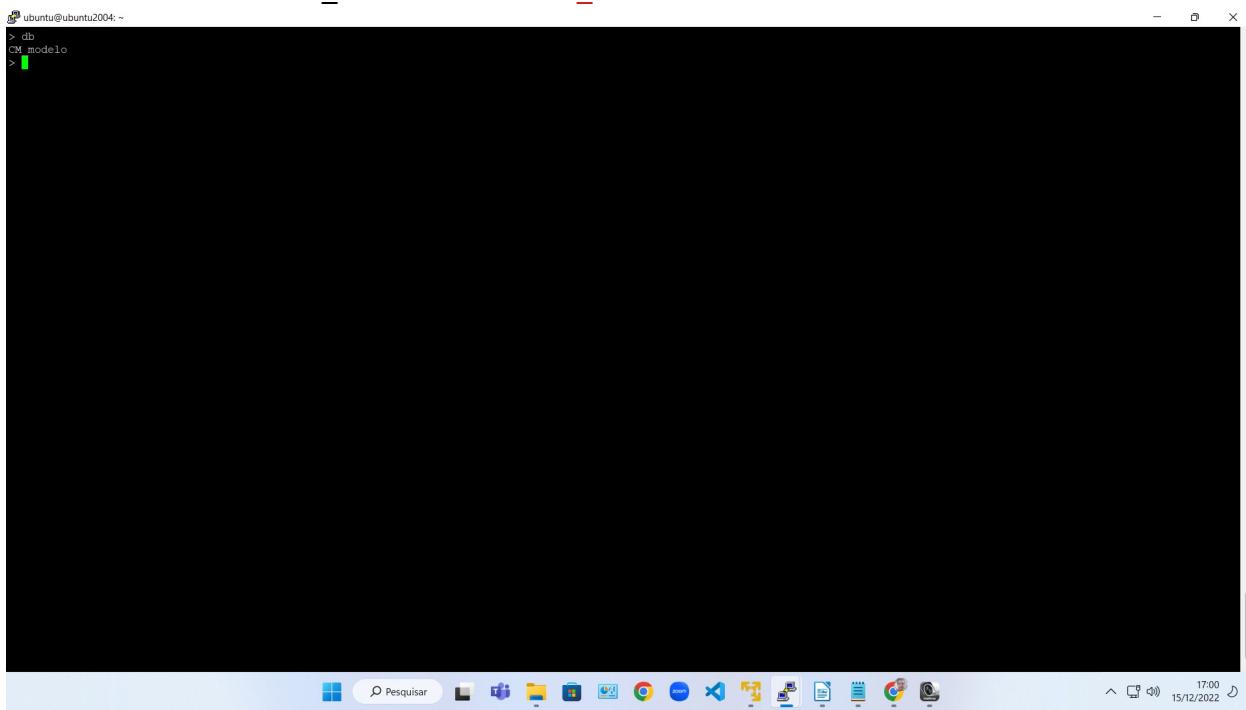
db.CM_DEPENDENTE.drop()



```
ubuntu@ubuntu2004:~\n> db.CM_DEPENDENTE.update({DEPENDENTE:"Melo"},{nome:"Melo"})\nWriteResult({ \"nMatched\" : 1, \"nUpserted\" : 0, \"nModified\" : 1 })\n> db.CM_DEPENDENTE.find()\n[ { _id : "01", nome : "Melo" },\n { _id : "02", DEPENDENTE : "Maia" },\n { _id : "03", DEPENDENTE : "Rocha" },\n { _id : "04", nome : "Melo" },\n { _id : "05", DEPENDENTE : "Aroldo" } ]\n> db.CM_DEPENDENTE.drop()\ntrue\n> db.CM_DEPENDENTE.find()\n
```

Modelagem

1. Criar o database "&&_modelo". use CM_modelo



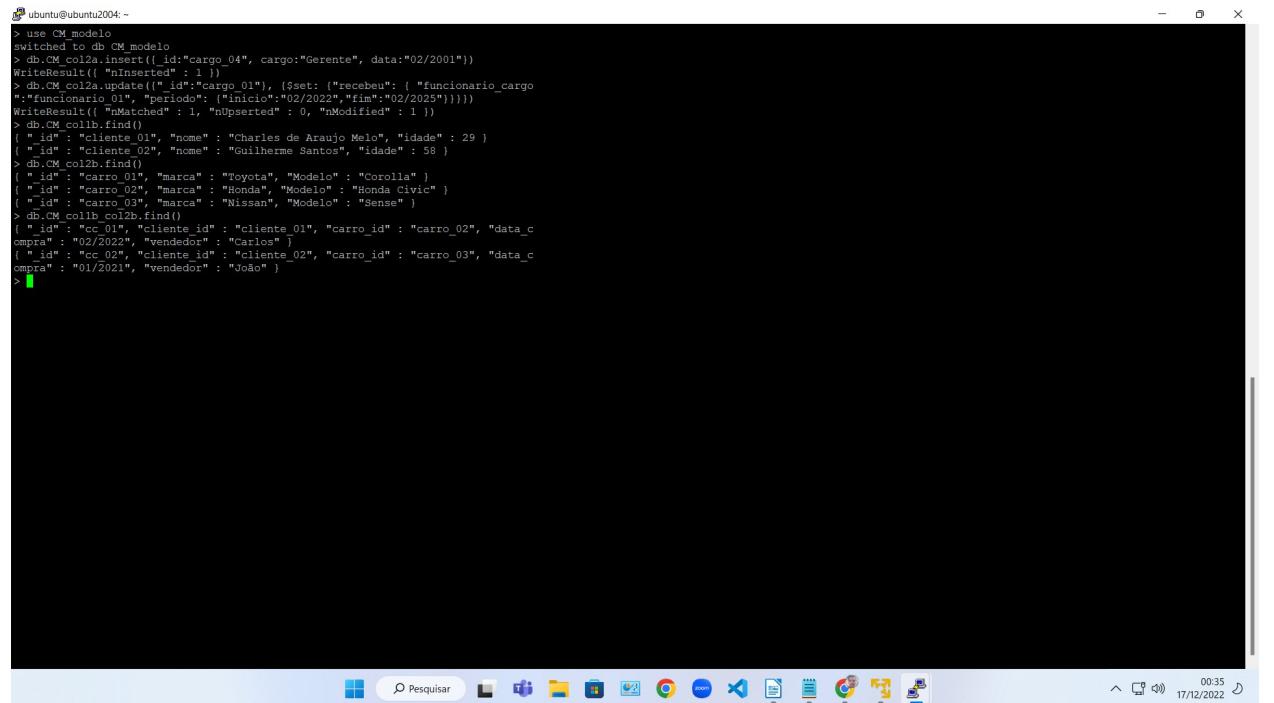
```
ubuntu@ubuntu2004:~\n> db\nCM.modelo\n> \n
```

2. Criar um pequeno modelo que envolva duas coleções com as cardinalidades 1-N. O nome das coleções deve ser "&&_col1a" e "&&_col2a"

A.Faça a inserção de 2 documentos na coleção (lado 1).

B.Faça a inserção de 4 documentos na coleção (lado N).

```
db.CM_col1a.insert({_id:"funcionario_01", nome:"Charles", sobrenome:"Melo"})
db.CM_col1a.insert({_id:"funcionario_02", nome:"Lucas", sobrenome:"Silva"})
db.CM_col2a.insert({_id:"cargo_01", cargo:"Secretario", data:"02/2022"})
db.CM_col2a.insert({_id:"cargo_02", cargo:"Help Desk", data:"01/2019"})
db.CM_col2a.insert({_id:"cargo_03", cargo:"Engenheiro", data:"02/2020"})
db.CM_col2a.insert({_id:"cargo_04", cargo:"Gerente", data:"02/2001"})
db.CM_col2a.update({"_id":"cargo_01"}, {$set: {"recebeu": 
{ "funcionario_cargo": "funcionario_01", "periodo": {"inicio": "02/2022", "fim": "02/2025"} }}})
```

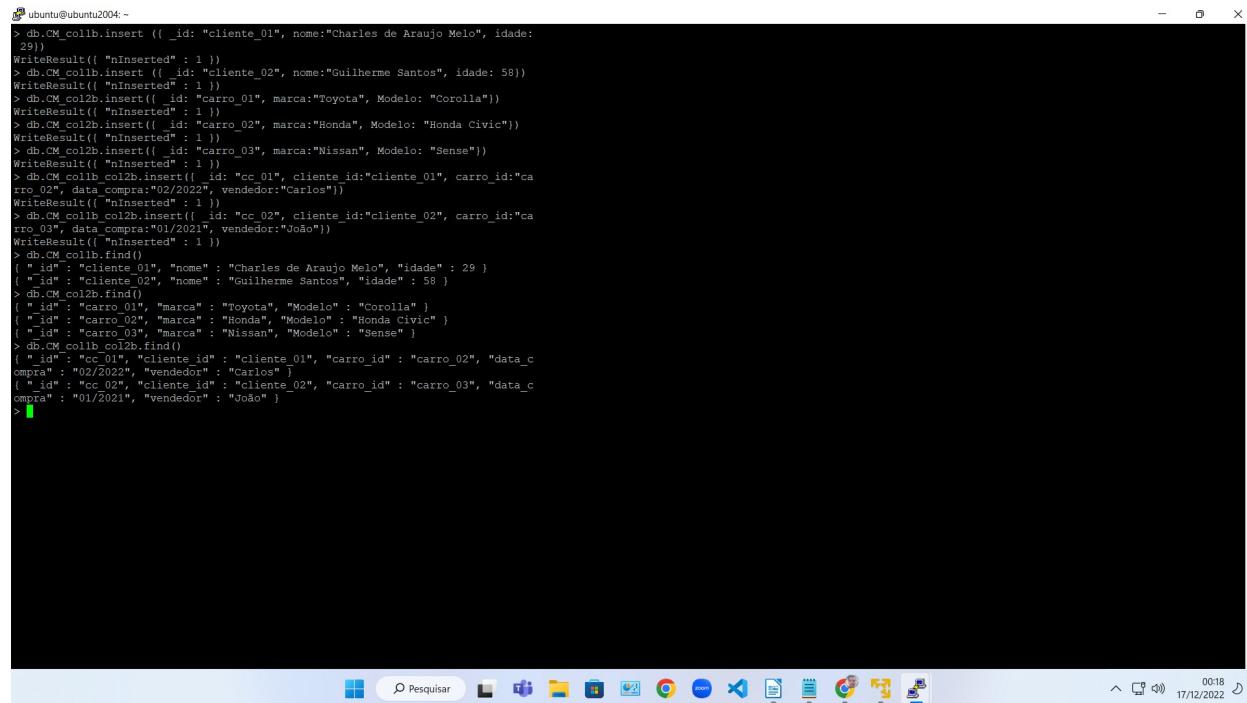


```
ubuntu@ubuntu2004: ~
> use CM modelo
switched to db CM modelo
> db.CM_col2a.insert({_id:"cargo_04", cargo:"Gerente", data:"02/2001"})
WriteResult({ "nInserted" : 1 })
> db.CM_col2a.update({"_id":"cargo_01"}, {$set: {"recebeu": { "funcionario_cargo": "funcionario_01", "periodo": {"inicio": "02/2022", "fim": "02/2025"} }}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.CM_col1b.find()
[{"_id": "cliente_01", "nome": "Charles de Araujo Melo", "idade": 29}
 {"_id": "cliente_02", "nome": "Guilherme Santos", "idade": 58}
> db.CM_col2b.find()
[{"_id": "carro_01", "marca": "Toyota", "Modelo": "Corolla"}
 {"_id": "carro_02", "marca": "Honda", "Modelo": "Honda Civic"}
 {"_id": "carro_03", "marca": "Nissan", "Modelo": "Sense"}
> db.CM_col1c.find()
[{"_id": "cc_01", "cliente_id": "cliente_01", "carro_id": "carro_02", "data_compra": "03/2022", "vendedor": "Carlo"}
 {"_id": "cc_02", "cliente_id": "cliente_02", "carro_id": "carro_03", "data_compra": "01/2021", "vendedor": "João"}]
```

3. Criar um pequeno modelo que envolva duas coleções com as cardinalidades N-N. O nome das coleções deve ser "&&_col1b" e "&&_col2b" Nesse caso você indicará se usará a referência ou se embarcará o documento no outro documento.

- A. Criar duas coleções de nome (se você embarcar) ou;
- B. Criar três coleções (se você usar referência);
- C. Caso você tenha escolhido a opção (3.1), criar as duas coleções e fazer um insert de 2 documentos em cada uma das coleções.
- D. Caso você tenha escolhido a opção (3.2), criar as três coleções e fazer um insert de 2 documentos em cada uma das coleções.

```
db.CM_col1b.insert({_id: "cliente_01", nome:"Charles de Araujo Melo", idade: 29})
db.CM_col1b.insert({_id: "cliente_02", nome:"Guilherme Santos", idade: 58})
db.CM_col2b.insert({_id: "carro_01", marca:"Toyota", Modelo: "Corolla"})
db.CM_col2b.insert({_id: "carro_02", marca:"Honda", Modelo: "Honda Civic"})
db.CM_col2b.insert({_id: "carro_03", marca:"Nissan", Modelo: "Sense"})
db.CM_col1b_col2b.insert({_id: "cc_01", cliente_id:"cliente_01", carro_id:"carro_02",
data_compra:"02/2022", vendedor:"Carlos"})
db.CM_col1b_col2b.insert({_id: "cc_02", cliente_id:"cliente_02", carro_id:"carro_03",
data_compra:"01/2021", vendedor:"João"})
```

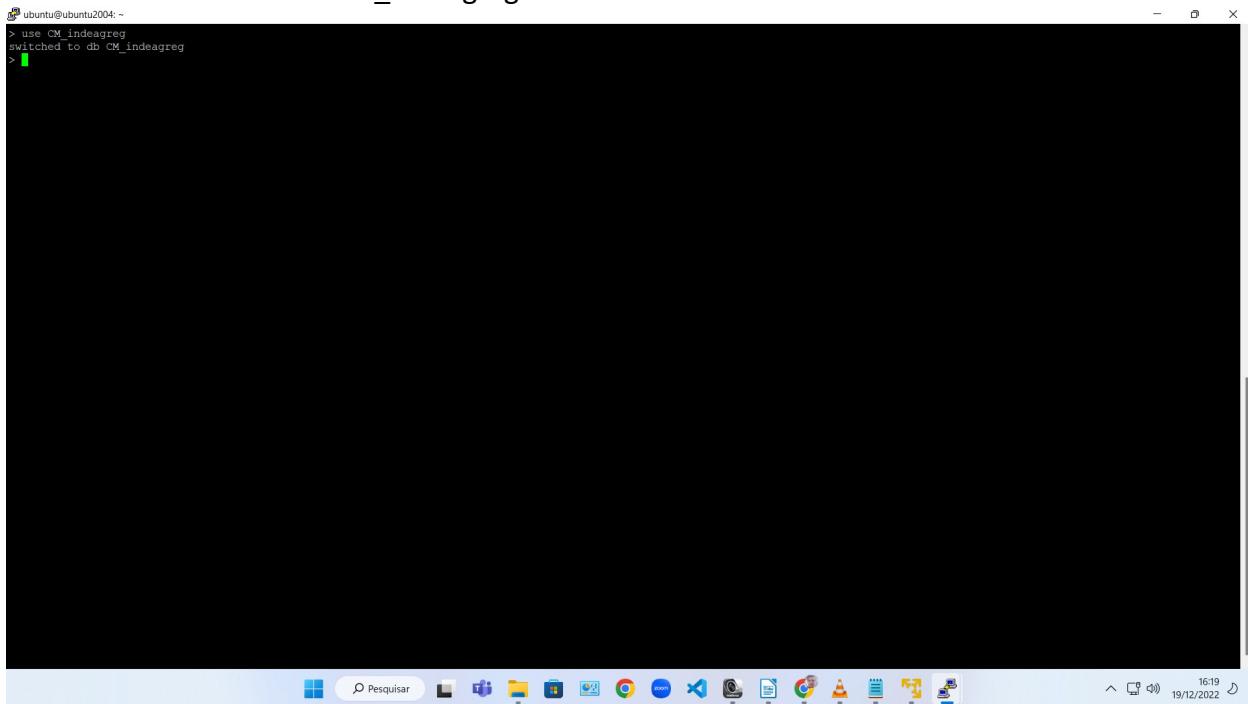


```
ubuntu@ubuntu2004:~>
> db.CM_col1b.insert({_id: "cliente_01", nome:"Charles de Araujo Melo", idade: 29})
WriteResult({ "nInserted": 1 })
> db.CM_col1b.insert({_id: "cliente_02", nome:"Guilherme Santos", idade: 58})
WriteResult({ "nInserted": 1 })
> db.CM_col2b.insert({_id: "carro_01", marca:"Toyota", Modelo: "Corolla"})
WriteResult({ "nInserted": 1 })
> db.CM_col2b.insert({_id: "carro_02", marca:"Honda", Modelo: "Honda Civic"})
WriteResult({ "nInserted": 1 })
> db.CM_col2b.insert({_id: "carro_03", marca:"Nissan", Modelo: "Sense"})
WriteResult({ "nInserted": 1 })
> db.CM_col1b_col2b.insert({_id: "cc_01", cliente_id:"cliente_01", carro_id:"carro_02",
data_compra:"02/2022", vendedor:"Carlos"})
WriteResult({ "nInserted": 1 })
> db.CM_col1b_col2b.insert({_id: "cc_02", cliente_id:"cliente_02", carro_id:"carro_03",
data_compra:"01/2021", vendedor:"João"})
WriteResult({ "nInserted": 1 })
> db.CM_col1b.find()
[{"_id": "cliente_01", "nome": "Charles de Araujo Melo", "idade": 29},
 {"_id": "cliente_02", "nome": "Guilherme Santos", "idade": 58}]
> db.CM_col2b.find()
[{"_id": "carro_01", "marca": "Toyota", "Modelo": "Corolla"},
 {"_id": "carro_02", "marca": "Honda", "Modelo": "Honda Civic"},
 {"_id": "carro_03", "marca": "Nissan", "Modelo": "Sense"}]
> db.CM_col1b_col2b.find()
[{"_id": "cc_01", "cliente_id": "cliente_01", "carro_id": "carro_02", "data_compra": "02/2022", "vendedor": "Carlos"},
 {"_id": "cc_02", "cliente_id": "cliente_02", "carro_id": "carro_03", "data_compra": "01/2021", "vendedor": "João"}]
> █
```

Índices

- 1.** Criar o database "&&_indeagreg".

Use CM_indeagreg



```
ubuntu@ubuntu2004: ~
> use CM_indeagreg
switched to db CM_indeagreg
> |
```

The screenshot shows a terminal window on a Linux desktop. The terminal prompt is 'ubuntu@ubuntu2004: ~'. The user runs the command 'use CM_indeagreg', which switches the database to 'CM_indeagreg'. The terminal then displays a single character '|'. The desktop environment includes a taskbar with various application icons like File Explorer, Google Chrome, and Microsoft Word, and a system tray showing the date and time as '19/12/2022 16:19'.

- 2.** Criar uma coleção de nome "&&_indexar1" que apresente pelo menos 7 atributos (tem que ter pelo menos um atributo array).

```
db.CM_indexar1.insert({
  nome: "Charles",
  sobrenome: "Melo",
  idade: 32,
  endereco: "Avenida General Guedes",
  esporte_praticado: ["Surf", "Futevolei", "Futebol"],
  relacionamento: "solteiro",
  possui_filhos: "nao"})
```

```

ubuntu@ubuntu2004:~ 
> show collections
CM_idexar1
> db.CM_idexar1.find().pretty()
{
    "_id" : ObjectId("63a0c3ebd8c38c1af52724b"),
    "name" : "Charles",
    "sobrenome" : "Melo",
    "idade" : 32,
    "endereco" : "Avenida General Guedes",
    "esporte_praticado" : [
        "Surf",
        "Futevolei",
        "Futebol"
    ],
    "relacionamento" : "solteiro",
    "possui_filhos" : "nao"
}
>

```

The terminal window shows the MongoDB shell running on an Ubuntu 20.04 system. It displays the creation of a collection named 'CM_idexar1' and a single document with fields like name, sobrenome, idade, endereco, esporte_praticado, relacionamento, and possui_filhos.

A. Criar um índice (não único e com ordenação ascendente).

```

var x = 10;
for(var i=0; i<x; i++) {
db.CM_idexar1.insert({a:i, b:i, c: i})
}
db.CM_idexar1.createIndex({a: 1, b: 10, c: 100})

```

```

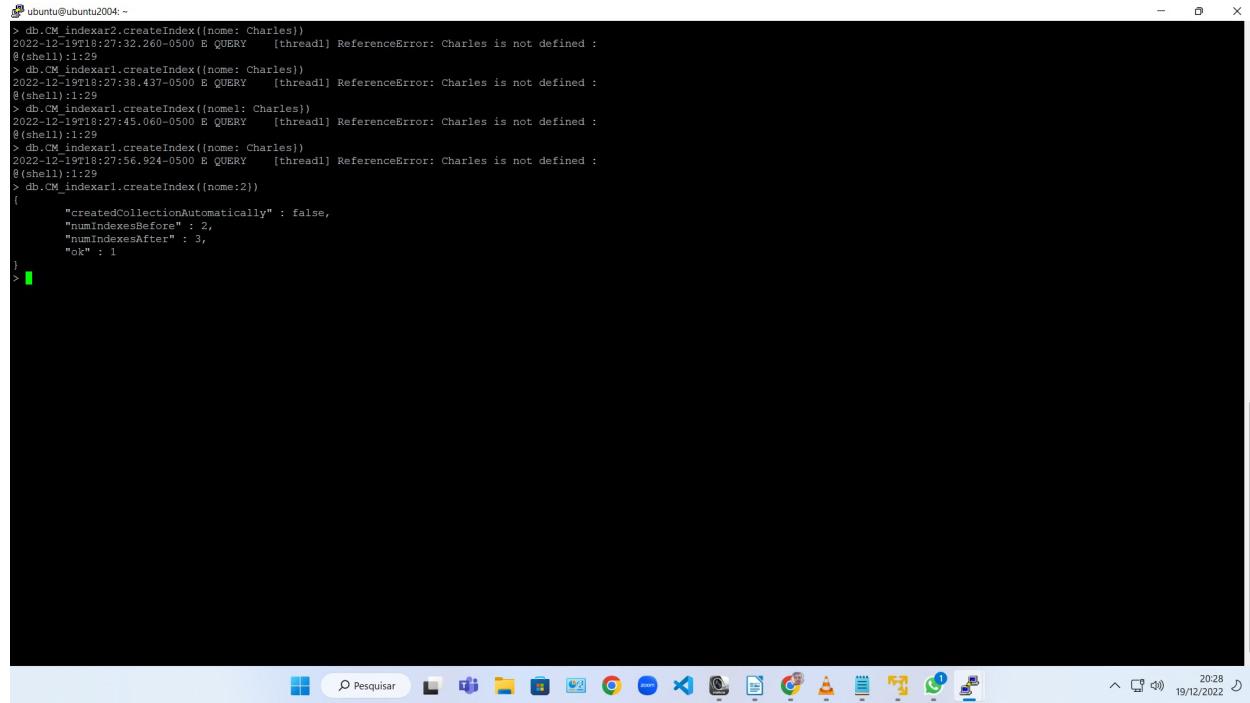
ubuntu@ubuntu2004:~ 
> db.CM_idexar1.find()
{
    "_id" : ObjectId("63a0dce4b7c35ea2872958d"),
    "a" : 0, "b" : 0, "c" : 0
}
{
    "_id" : ObjectId("63a0dce4b7c35ea2872958e"),
    "a" : 1, "b" : 1, "c" : 1
}
{
    "_id" : ObjectId("63a0dce4b7c35ea2872958f"),
    "a" : 2, "b" : 2, "c" : 2
}
{
    "_id" : ObjectId("63a0dce4b7c35ea28729590"),
    "a" : 3, "b" : 3, "c" : 3
}
{
    "_id" : ObjectId("63a0dce4b7c35ea28729591"),
    "a" : 4, "b" : 4, "c" : 4
}
{
    "_id" : ObjectId("63a0dce4b7c35ea28729592"),
    "a" : 5, "b" : 5, "c" : 5
}
{
    "_id" : ObjectId("63a0dce4b7c35ea28729593"),
    "a" : 6, "b" : 6, "c" : 6
}
{
    "_id" : ObjectId("63a0dce4b7c35ea28729594"),
    "a" : 7, "b" : 7, "c" : 7
}
{
    "_id" : ObjectId("63a0dce4b7c35ea28729595"),
    "a" : 8, "b" : 8, "c" : 8
}
{
    "_id" : ObjectId("63a0dce4b7c35ea28729596"),
    "a" : 9, "b" : 9, "c" : 9
}
> db.CM_idexar1.createIndex({a: 1, b: 10, c: 100})
{
    "createdCollectionAutomatically" : true,
    "numIndexesBefore" : 1,
    "numIndexesAfter" : 2,
    "ok" : 1
}
>

```

The terminal window shows the insertion of 10 documents into the 'CM_idexar1' collection and the creation of a new index on the fields a, b, and c. The index is created automatically, increasing the number of indexes from 1 to 2.

B. Criar um índice (único e com ordenação descendente).

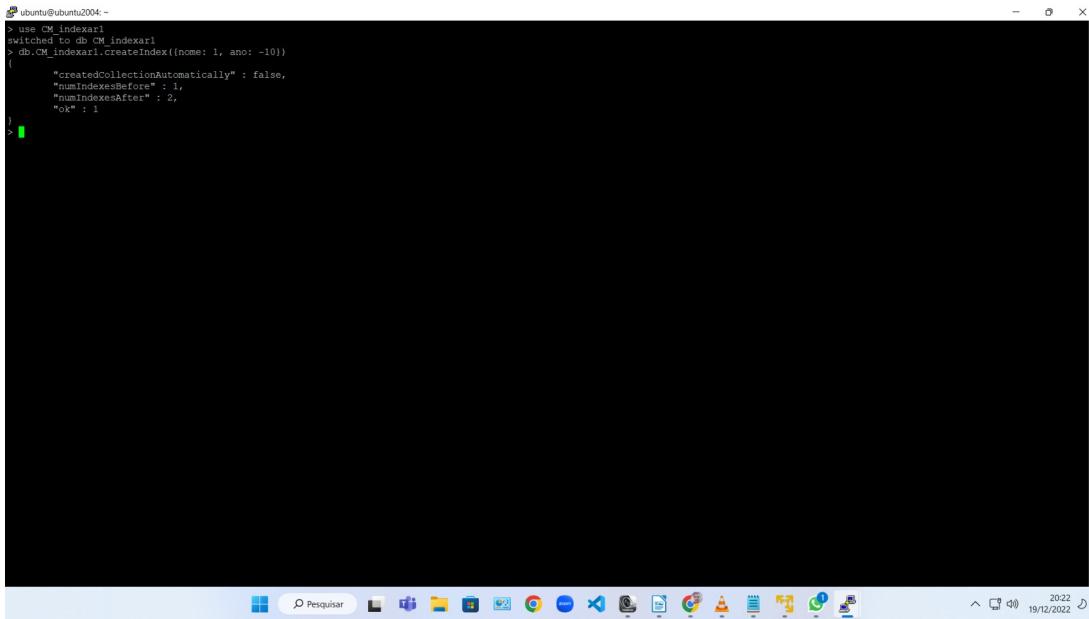
```
db.CM_indexar1.createIndex({nome: 1})
```



```
ubuntu@ubuntu2004:~> db.CM_indexar2.createIndex({nome: Charles})
2022-12-19T18:27:32.260+0500 E QUERY    [thread1] ReferenceError: Charles is not defined :
@shell:1:29
> db.CM_indexar1.createIndex({nome: Charles})
2022-12-19T18:27:38.437+0500 E QUERY    [thread1] ReferenceError: Charles is not defined :
@shell:1:29
> db.CM_indexar1.createIndex({nomel: Charles})
2022-12-19T18:27:45.060+0500 E QUERY    [thread1] ReferenceError: Charles is not defined :
@shell:1:29
> db.CM_indexar1.createIndex({nome: Charles})
2022-12-19T18:27:56.924+0500 E QUERY    [thread1] ReferenceError: Charles is not defined :
@shell:1:29
> db.CM_indexar1.createIndex({nome:2})
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 2,
  "numIndexesAfter" : 3,
  "ok" : 1
}
> 
```

C. Criar um índice (com dois atributos).

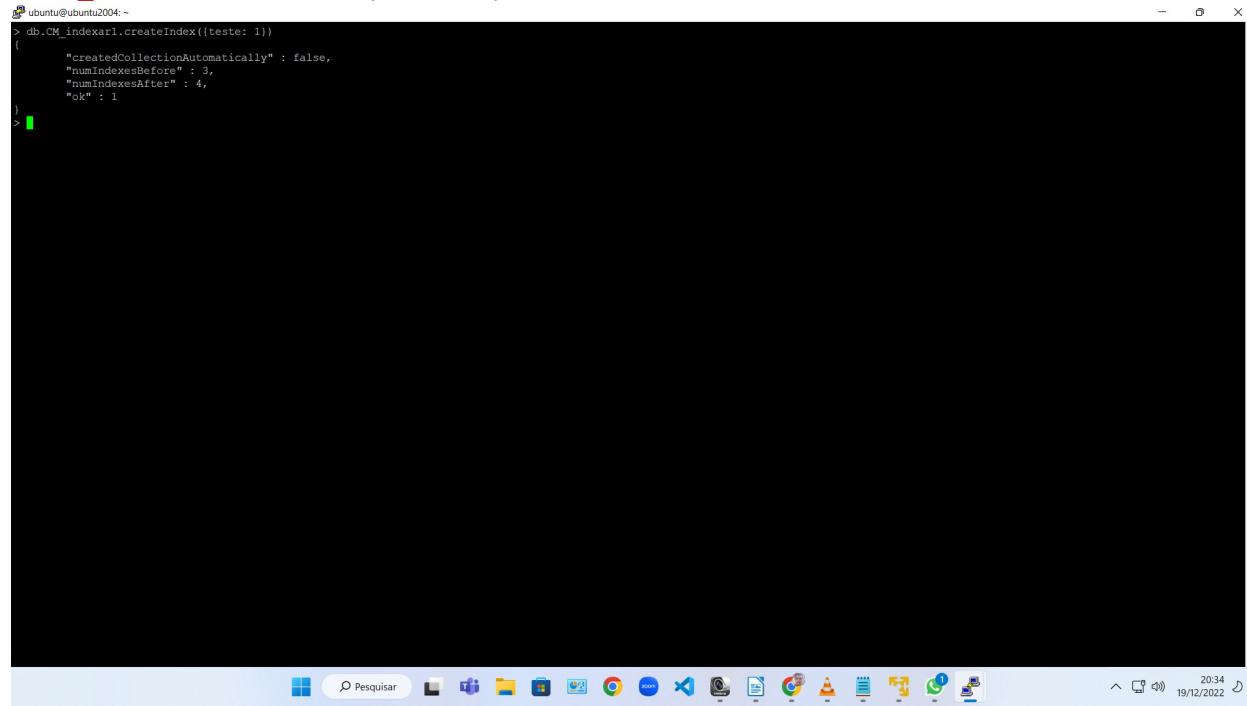
```
db.CM_indexar1.createIndex({nome: 1, ano: -10})
```



```
ubuntu@ubuntu2004:~> use CM_Indexar1
switched to db CM_Indexar1
> db.CM_indexar1.createIndex({nome: 1, ano: -10})
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "ok" : 1
}
> 
```

D. Criar um índice para um atributo array.

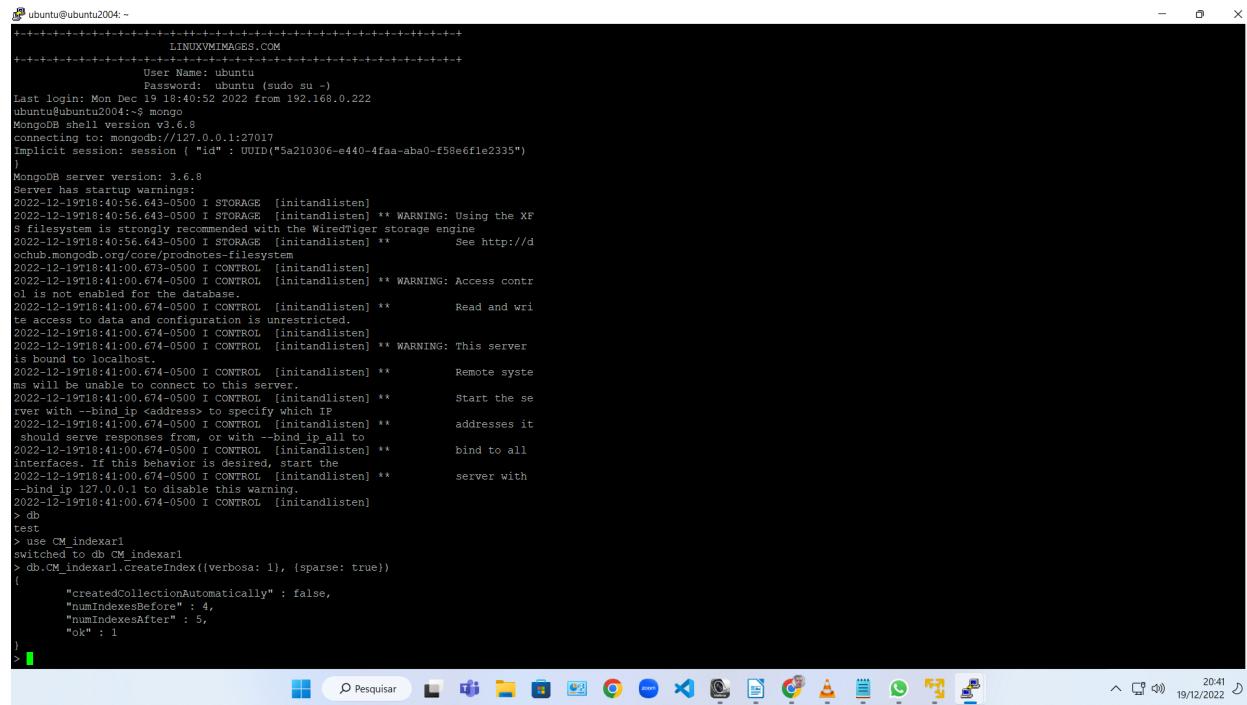
db.VJ_indexar1.createIndex({teste: 1})



```
ubuntu@ubuntu2004: ~
> db.VJ_indexar1.createIndex({teste: 1})
{
    "createdCollectionAutomatically" : false,
    "numIndexesBefore" : 3,
    "numIndexesAfter" : 4,
    "ok" : 1
}
>
```

E. Criar um índice esparso (com um atributo).

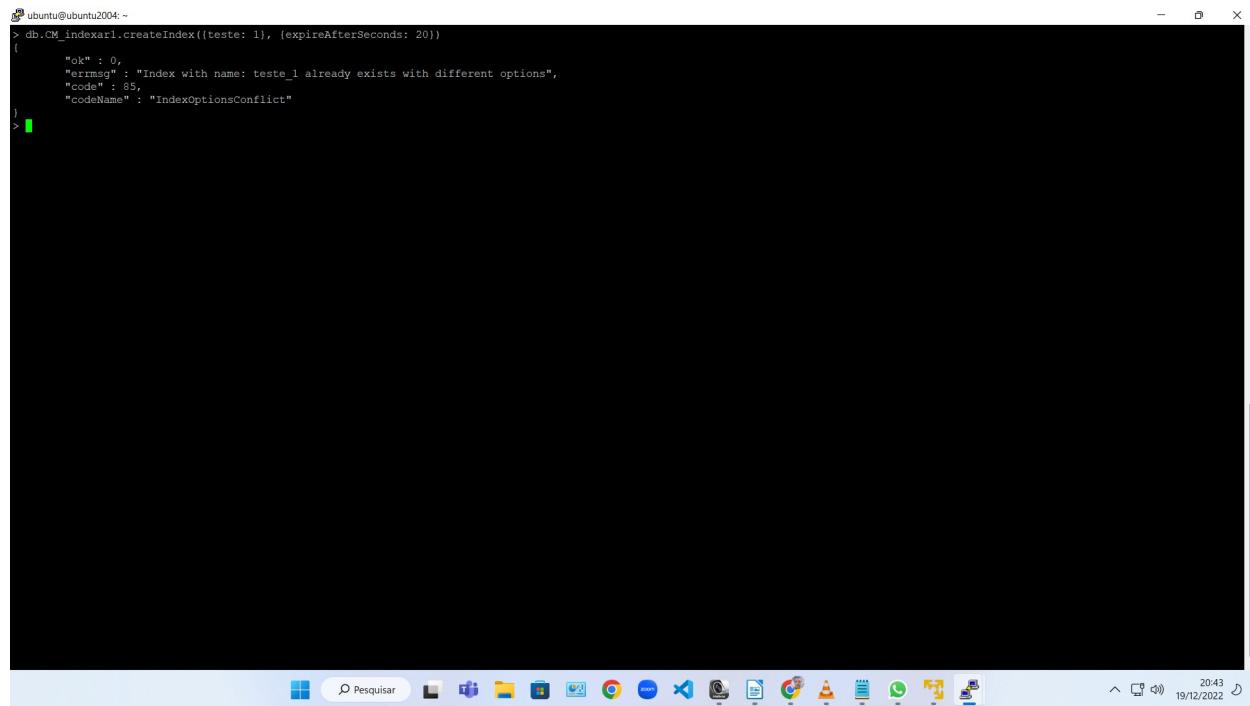
db.CM_indexar1.createIndex({teste: 1}, {sparse: true})



```
ubuntu@ubuntu2004: ~
LINUXVMIMAGES.COM
User Name: ubuntu
Password: ubuntu (sudo su -)
Last login: Mon Dec  5 18:40:52 2022 from 192.168.0.222
ubuntu@ubuntu2004: ~$ mongo
MongoDB shell version v3.6.8
connecting to: mongodb://127.0.0.1:27017
Implicit session: session [ "id" : UUID("5a210306-e440-4faa-aba0-f58e6f1e2335" )
]
MongoDB server version: 3.6.8
Server has startup warnings:
2022-12-19T18:40:56.643-0500 I STORAGE [initandlisten]
2022-12-19T18:40:56.643-0500 I STORAGE [initandlisten] ** WARNING: Using the XF
S filesystem is strongly recommended with the Wiredtiger storage engine
2022-12-19T18:40:56.643-0500 I STORAGE [initandlisten] ** See http://d
ocs.mongodb.com/v3.6/administration/storage-filesystems
2022-12-19T18:41:00.674-0500 I CONTROL [initandlisten]
2022-12-19T18:41:00.674-0500 I CONTROL [initandlisten] ** WARNING: Access contr
ol is not enabled for the database.
2022-12-19T18:41:00.674-0500 I CONTROL [initandlisten] ** Read and wri
te access to data and configuration is unrestricted.
2022-12-19T18:41:00.674-0500 I CONTROL [initandlisten]
2022-12-19T18:41:00.674-0500 I CONTROL [initandlisten] ** WARNING: This server
is bound to localhost.
2022-12-19T18:41:00.674-0500 I CONTROL [initandlisten] ** Remote syste
ms will be unable to connect to this server.
2022-12-19T18:41:00.674-0500 I CONTROL [initandlisten] ** Start the se
rver with --bind_ip <address> to specify which IP
2022-12-19T18:41:00.674-0500 I CONTROL [initandlisten] ** addresses it
should serve responses from, or with --bind_ip all to
2022-12-19T18:41:00.674-0500 I CONTROL [initandlisten] ** bind to all
interfaces. If this behavior is desired, start the
2022-12-19T18:41:00.674-0500 I CONTROL [initandlisten] ** server with
--bind_ip 127.0.0.1 to disable this warning.
2022-12-19T18:41:00.674-0500 I CONTROL [initandlisten]
> db
test
> use CM_indexar1
switched to db CM_indexar1
> db.CM_indexar1.createIndex({teste: 1}, {sparse: true})
{
    "createdCollectionAutomatically" : false,
    "numIndexesBefore" : 4,
    "numIndexesAfter" : 5,
    "ok" : 1
}
>
```

F. Criar um índice com tempo de vida - TTL (com um atributo e com expiração de 20 segundos).

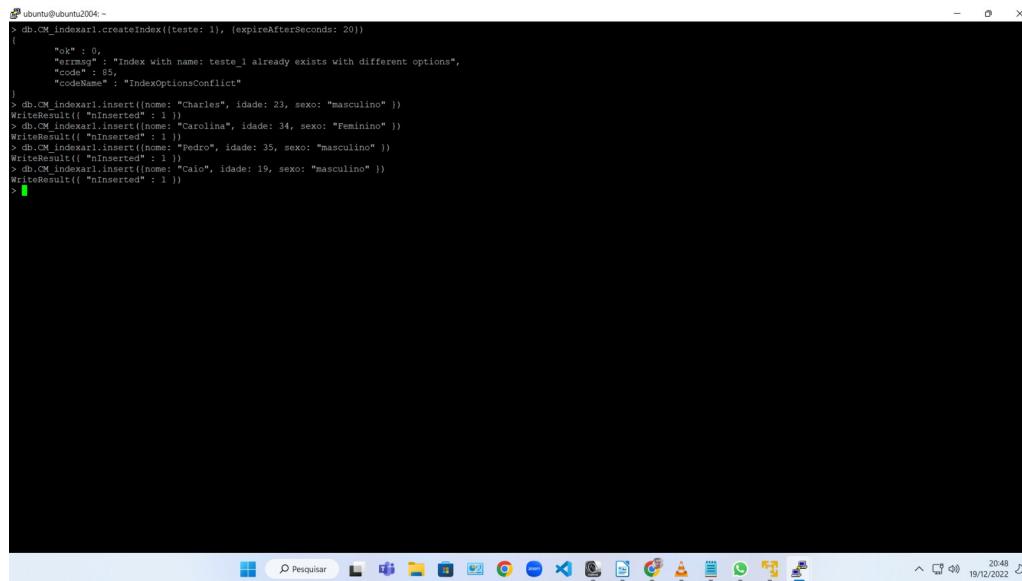
```
db.CM_indexar1.createIndex({teste: 1}, {expireAfterSeconds: 20})
```



```
ubuntu@ubuntu2004:~  
> db.CM_indexar1.createIndex({teste: 1}, {expireAfterSeconds: 20})  
{  
    "ok" : 0,  
    "errmsg" : "Index with name: teste_1 already exists with different options",  
    "code" : 85,  
    "codeName" : "IndexOptionsConflict"  
}  
>
```

G. Inserir 4 documentos nessa coleção.

```
db.CM_indexar1.insert({nome: "Charles", idade: 23, sexo: "masculino" })  
db.CM_indexar1.insert({nome: "Carolina", idade: 34, sexo: "Feminino" })  
db.CM_indexar1.insert({nome: "Pedro", idade: 35, sexo: "masculino" })  
db.CM_indexar1.insert({nome: "Caio", idade: 19, sexo: "masculino" })
```



```
ubuntu@ubuntu2004:~  
> db.CM_indexar1.createIndex({teste: 1}, {expireAfterSeconds: 20})  
{  
    "ok" : 0,  
    "errmsg" : "Index with name: teste_1 already exists with different options",  
    "code" : 85,  
    "codeName" : "IndexOptionsConflict"  
}  
> db.CM_indexar1.insert({nome: "Charles", idade: 23, sexo: "masculino" })  
writeResult({ "nInserted" : 1 })  
> db.CM_indexar1.insert({nome: "Carolina", idade: 34, sexo: "Feminino" })  
writeResult({ "nInserted" : 1 })  
> db.CM_indexar1.insert({nome: "Pedro", idade: 35, sexo: "masculino" })  
writeResult({ "nInserted" : 1 })  
> db.CM_indexar1.insert({nome: "Caio", idade: 19, sexo: "masculino" })  
writeResult({ "nInserted" : 1 })  
>
```

3. Criar uma segunda coleção de nome "&&_indexar2" (que apresente pelo menos 4 atributos string)

db.CM_indexar2.insert({ nome: "Charles", sobrenome: "Melo", endereco: "Avenida General Guedes", relacionamento: "solteiro"})

```
ubuntu@ubuntu2004:~  
> db  
CM_indexar2  
> db.CM_indexar2.insert({ nome: "Charles", sobrenome: "Melo", endereco: "Avenida General Guedes", relacionamento: "solteiro"})  
WriteResult({ "nInserted" : 1 })  
> |
```

A. Criar um índice textual para essa segunda coleção.

B. Inserir 4 documentos nessa coleção.

```
ubuntu@ubuntu2004:~  
CM_indexar2  
> db.CM_indexar2.insert({ nome: "Charles", sobrenome: "Melo", endereco: "Avenida General Guedes", relacionamento: "solteiro"})  
WriteResult({ "nInserted" : 1 })  
> db.CM_indexar2.insert({ words: "Cat tree obsidian." });  
WriteResult({ "nInserted" : 1 })  
> db.CM_indexar2.insert({ words: "Cat tree granite." });  
WriteResult({ "nInserted" : 1 })  
> db.CM_indexar2.insert({ words: "Cat shrub ruby." });  
WriteResult({ "nInserted" : 1 })  
> db.CM_indexar2.insert({ words: "Cat shrub obsidian." });  
WriteResult({ "nInserted" : 1 })  
> db.CM_indexar2.insert({ words: "Cat shrub granite." });  
WriteResult({ "nInserted" : 1 })  
> db.CM_indexar2.insert({ words: "Cat moss ruby." });  
WriteResult({ "nInserted" : 1 })  
> db.CM_indexar2.insert({ words: "Cat moss obsidian." });  
WriteResult({ "nInserted" : 1 })  
> db.CM_indexar2.insert({ words: "Cat moss granite." });  
WriteResult({ "nInserted" : 1 })  
> db.CM_indexar2.insert({ words: "dog tree ruby." });  
WriteResult({ "nInserted" : 1 })  
> db.CM_indexar2.insert({ words: "dog tree obsidian." });  
WriteResult({ "nInserted" : 1 })  
> db.CM_indexar2.insert({ words: "dog tree granite." });  
WriteResult({ "nInserted" : 1 })  
> db.CM_indexar2.insert({ words: "dog shrub ruby." });  
WriteResult({ "nInserted" : 1 })  
> db.CM_indexar2.find()  
{  
    "_id": ObjectId("63a0fdf6817292f289bebdd4"), "name": "Charles", "sobrenome": "Melo", "endereco": "Avenida General Guedes", "relacionamento": "solteiro"  
},  
{  
    "_id": ObjectId("63a0fdf6817292f289bebdd5"), "words": "Cat tree obsidian."  
},  
{  
    "_id": ObjectId("63a0fdf6817292f289bebdd6"), "words": "Cat tree granite."  
},  
{  
    "_id": ObjectId("63a0fdf6817292f289bebdd7"), "words": "Cat shrub ruby."  
},  
{  
    "_id": ObjectId("63a0fdf6817292f289bebdd8"), "words": "Cat shrub obsidian."  
},  
{  
    "_id": ObjectId("63a0fdf6817292f289bebdd9"), "words": "Cat shrub granite."  
},  
{  
    "_id": ObjectId("63a0fdf6817292f289bebdd0"), "words": "Cat moss ruby."  
},  
{  
    "_id": ObjectId("63a0fdf6817292f289bebdd1"), "words": "Cat moss obsidian."  
},  
{  
    "_id": ObjectId("63a0fdf6817292f289bebdd2"), "words": "Cat moss granite."  
},  
{  
    "_id": ObjectId("63a0fdf6817292f289bebdd3"), "words": "dog tree ruby."  
},  
{  
    "_id": ObjectId("63a0fdf6817292f289bebdd4"), "words": "dog tree obsidian."  
},  
{  
    "_id": ObjectId("63a0fdf6817292f289bebdd5"), "words": "dog tree granite."  
},  
{  
    "_id": ObjectId("63a0fdf6817292f289bebdd6"), "words": "dog shrub ruby."  
},  
{  
    "_id": ObjectId("63a0fdf6817292f289bebdd7"), "words": "dog shrub obsidian."  
},  
{  
    "_id": ObjectId("63a0fdf6817292f289bebdd8"), "words": "dog shrub granite."  
},  
{  
    "_id": ObjectId("63a0fdfda017292f289bebdd9"), "words": "text"  
}  
}  
db.CM_indexar2.ensureIndex({ words: "text" })  
> |
```

```
db.CM_indexar2.insert ({ words: "Cat tree obsidian." });
db.CM_indexar2.insert ({ words: "Cat tree granite." });
db.CM_indexar2.insert ({ words: "Cat shrub ruby." });
db.CM_indexar2.insert ({ words: "Cat shrub obsidian." });
```

4. Criar uma terceira coleção de nome "&&_Venda" (na qual terão os atributos: Cod_Venda, UF_Venda, Desc_Prod_Vendido, Valor_Venda).

A. Inserir 16 documentos (sendo 4 documentos para cada uma das UFs - serão 4 UFs diferentes). Preencher os 4 atributos citados.

```
db.CM_Venda.insert({
Cod_Venda: 1001,
UF_Venda: "Rio de Janeiro",
Desc_Prod_Vendido: "Televisao de 55 polegadas",
Valor_Venda: 1650.00})
```

```
db.CM_Venda.insert({
Cod_Venda: 1002,
UF_Venda: "Rio de Janeiro",
Desc_Prod_Vendido: "Geladeira inox",
Valor_Venda: 3000.00})
```

```
db.CM_Venda.insert({
Cod_Venda: 1003,
UF_Venda: "Rio de Janeiro",
Desc_Prod_Vendido: "xbox serie s",
Valor_Venda: 2300.00})
```

```
db.CM_Venda.insert({
Cod_Venda: 1004,
UF_Venda: "Rio de Janeiro",
Desc_Prod_Vendido: "Celular Sansung",
Valor_Venda: 900.00})
```

```
ubuntu@ubuntu2004:~  
> use CM_indexar2  
switched to db CM_indexar2  
> db.CM_Venda.insert({  
... Cod_Venda: 1001,  
... UF_Venda: "Rio de Janeiro",  
... Desc_Prod_Vendido: "Televisao de 55 polegadas",  
... Valor_Venda: 1650.00})  
WriteResult({ "nInserted" : 1 })  
> db.CM_Venda.insert({  
... Cod_Venda: 1002,  
... UF_Venda: "Rio de Janeiro",  
... Desc_Prod_Vendido: "Geladeira inox",  
... Valor_Venda: 3000.00})  
WriteResult({ "nInserted" : 1 })  
> db.CM_Venda.insert({  
... Cod_Venda: 1003,  
... UF_Venda: "Rio de Janeiro",  
... Desc_Prod_Vendido: "xbox serie s",  
... Valor_Venda: 2300.00})  
WriteResult({ "nInserted" : 1 })  
> db.CM_Venda.insert({  
... Cod_Venda: 1004,  
... UF_Venda: "Rio de Janeiro",  
... Desc_Prod_Vendido: "Celular Sansung",  
... Valor_Venda: 900.00})  
WriteResult({ "nInserted" : 1 })  
> █
```

```
db.CM_Venda.insert({  
Cod_Venda: 1001,  
UF_Venda: "Minas Gerais",  
Desc_Prod_Vendido: "Televisao de 55 polegadas",  
Valor_Venda: 1650.00})
```

```
db.CM_Venda.insert({  
Cod_Venda: 1002,  
UF_Venda: "Minas Gerais",  
Desc_Prod_Vendido: "Geladeira inox",  
Valor_Venda: 3000.00})
```

```
db.CM_Venda.insert({  
Cod_Venda: 1003,  
UF_Venda: "Minas Gerais",  
Desc_Prod_Vendido: "xbox serie s",  
Valor_Venda: 2300.00})
```

```
db.CM_Venda.insert({  
Cod_Venda: 1004,  
UF_Venda: "Minas Gerais",  
Desc_Prod_Vendido: "Celular Sansung",
```

```
Valor_Venda: 900.00})
```

```
db.CM_Venda.insert({  
Cod_Venda: 1001,  
UF_Venda: "Amapa",  
Desc_Prod_Vendido: "Televisao de 55 polegadas",  
Valor_Venda: 1650.00})
```

```
db.CM_Venda.insert({  
Cod_Venda: 1002,  
UF_Venda: "Amapa",  
Desc_Prod_Vendido: "Geladeira inox",  
Valor_Venda: 3000.00})
```

```
db.CM_Venda.insert({  
Cod_Venda: 1003,  
UF_Venda: "Amapa",  
Desc_Prod_Vendido: "xbox serie s",  
Valor_Venda: 2300.00})
```

```
db.CM_Venda.insert({  
Cod_Venda: 1004,  
UF_Venda: "Amapa",  
Desc_Prod_Vendido: "Celular Sansung",  
Valor_Venda: 900.00})
```

```
db.CM_Venda.insert({  
Cod_Venda: 1001,  
UF_Venda: "Brasilia",  
Desc_Prod_Vendido: "Televisao de 55 polegadas",  
Valor_Venda: 1650.00})
```

```
db.CM_Venda.insert({  
Cod_Venda: 1002,  
UF_Venda: "Brasilia",  
Desc_Prod_Vendido: "Geladeira inox",  
Valor_Venda: 3000.00})
```

```
db.CM_Venda.insert({  
    Cod_Venda: 1003,  
    UF_Venda: "Brasilia",  
    Desc_Prod_Vendido: "xbox serie s",  
    Valor_Venda: 2300.00})
```

```
db.CM_Venda.insert({  
    Cod_Venda: 1004,  
    UF_Venda: "Brasilia",  
    Desc_Prod_Vendido: "Celular Sansung",  
    Valor_Venda: 900.00})
```

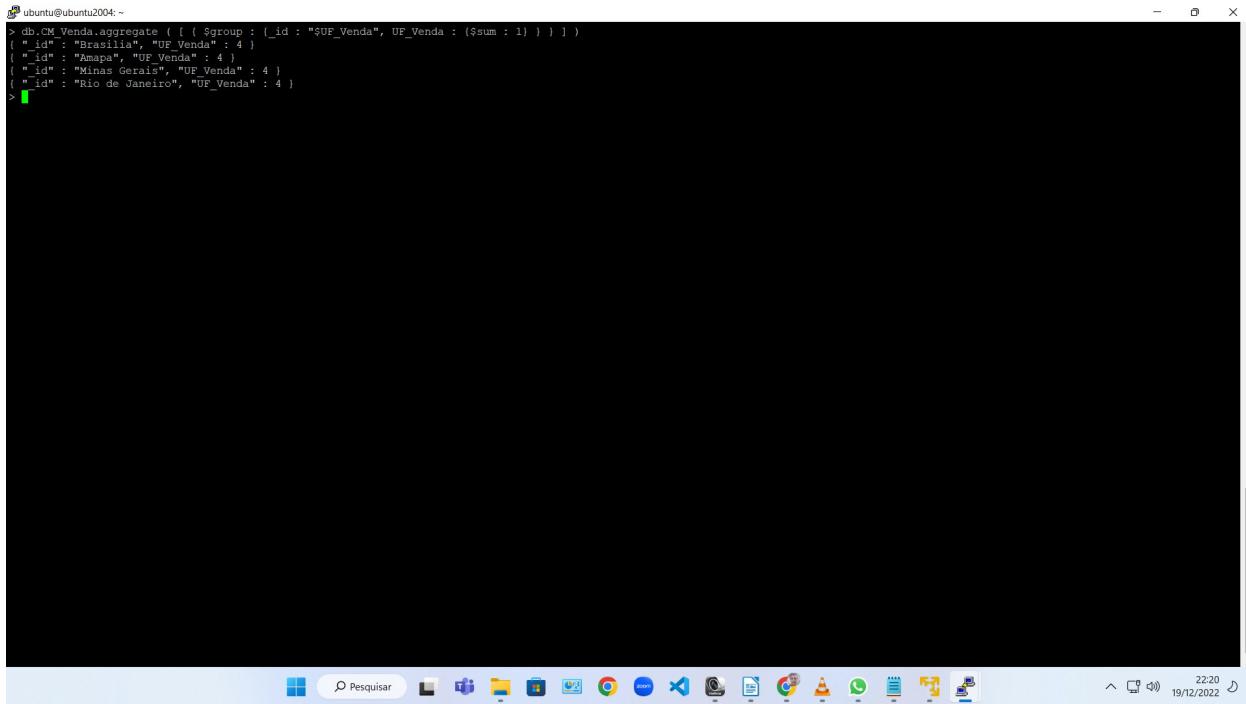
The screenshot shows a terminal window on a Windows operating system. The title bar says "ubuntu@ubuntu2004:~". The window contains the following MongoDB shell session:

```
... UF Venda: "Amapa",  
... Desc_Prod_Vendido: "Geladeira inox",  
... Valor_Venda: 3000.00)  
WriteResult({ "nInserted" : 1 })  
>  
> db.CM_Venda.insert({  
... Cod_Venda: 1003,  
... UF Venda: "Amapa",  
... Desc_Prod_Vendido: "xbox serie s",  
... Valor_Venda: 2300.00})  
WriteResult({ "nInserted" : 1 })  
>  
> db.CM_Venda.insert({  
... Cod_Venda: 1004,  
... UF Venda: "Amapa",  
... Desc_Prod_Vendido: "Celular Sansung",  
... Valor_Venda: 900.00})  
WriteResult({ "nInserted" : 1 })  
> db.CM_Venda.insert({  
... Cod_Venda: 1001,  
... UF Venda: "Brasilia",  
... Desc_Prod_Vendido: "Televisao de 55 polegadas",  
... Valor_Venda: 1650.00})  
WriteResult({ "nInserted" : 1 })  
Cod_Venda: 1001  
UF Venda: "Brasilia",  
Desc_Prod_Vendido: "Celular Sansung",  
Valor_Venda: 900.00)WriteResult({ "nInserted" : 1 })  
>  
> db.CM_Venda.insert({  
... Cod_Venda: 1002,  
... UF Venda: "Brasilia",  
... Desc_Prod_Vendido: "Geladeira inox",  
... Valor_Venda: 3000.00})  
WriteResult({ "nInserted" : 1 })  
>  
> db.CM_Venda.insert({  
... Cod_Venda: 1003,  
... UF Venda: "Brasilia",  
... Desc_Prod_Vendido: "xbox serie s",  
... Valor_Venda: 2300.00})  
WriteResult({ "nInserted" : 1 })  
>  
> db.CM_Venda.insert({  
... Cod_Venda: 1004,  
... UF Venda: "Brasilia",  
... Desc_Prod_Vendido: "Celular Sansung",  
... Valor_Venda: 900.00})  
WriteResult({ "nInserted" : 1 })
```

The terminal window has a dark background and light-colored text. The bottom of the window shows the Windows taskbar with various icons for apps like File Explorer, Google Chrome, and Microsoft Office.

B. Criar uma consulta que mostre o número de documentos por UF.

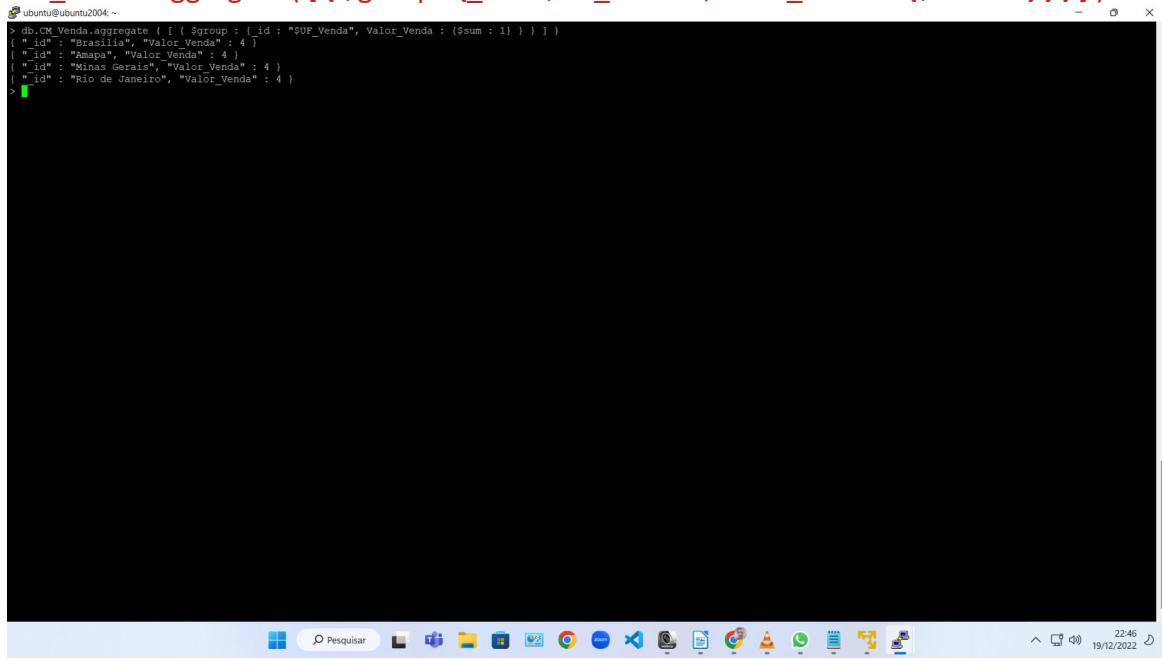
```
db.CM_Venda.aggregate ( [ { $group : { _id : "$UF_Venda", UF_Venda : { $sum : 1 } } } ] )
```



```
ubuntu@ubuntu2004:~> db.CM_Venda.aggregate ( [ { $group : { _id : "$UF_Venda", UF_Venda : { $sum : 1 } } } ] )
[ { "id" : "Brasilia", "UF_Venda" : 4 },
{ "id" : "Amapa", "UF_Venda" : 4 },
{ "id" : "Minas Gerais", "UF_Venda" : 4 },
{ "id" : "Rio de Janeiro", "UF_Venda" : 4 }]
```

C. Criar uma consulta que mostre o valor total das vendas por UF.

```
db.CM_Venda.aggregate ( [ { $group : { _id : "$UF_Venda", Valor_Venda : { $sum : 1 } } } ] )
```



```
ubuntu@ubuntu2004:~> db.CM_Venda.aggregate ( [ { $group : { _id : "$UF_Venda", Valor_Venda : { $sum : 1 } } } ] )
[ { "id" : "Brasilia", "Valor_Venda" : 4 },
{ "id" : "Amapa", "Valor_Venda" : 4 },
{ "id" : "Minas Gerais", "Valor_Venda" : 4 },
{ "id" : "Rio de Janeiro", "Valor_Venda" : 4 }]
```

D. Criar uma consulta que mostre o valor de médias das vendas por UF.

```
db.CM_Venda.aggregate([{$group:{_id : "$UF_Venda", media_venda : {$avg : "$Valor_Venda"}}}])
```

```
ubuntu@ubuntu2004:~\n> db.CM_Venda.aggregate ([ { $group : { _id : "$UF_Venda", Media : { $avg : "$Valor_Venda" } } } ] )\n{ \"_id\" : \"Brasilia\", \"Media\" : 1962.5 }\n{ \"_id\" : \"Amapa\", \"Media\" : 1962.5 }\n{ \"_id\" : \"Minas Gerais\", \"Media\" : 1962.5 }\n{ \"_id\" : \"Rio de Janeiro\", \"Media\" : 1962.5 }\n> [
```

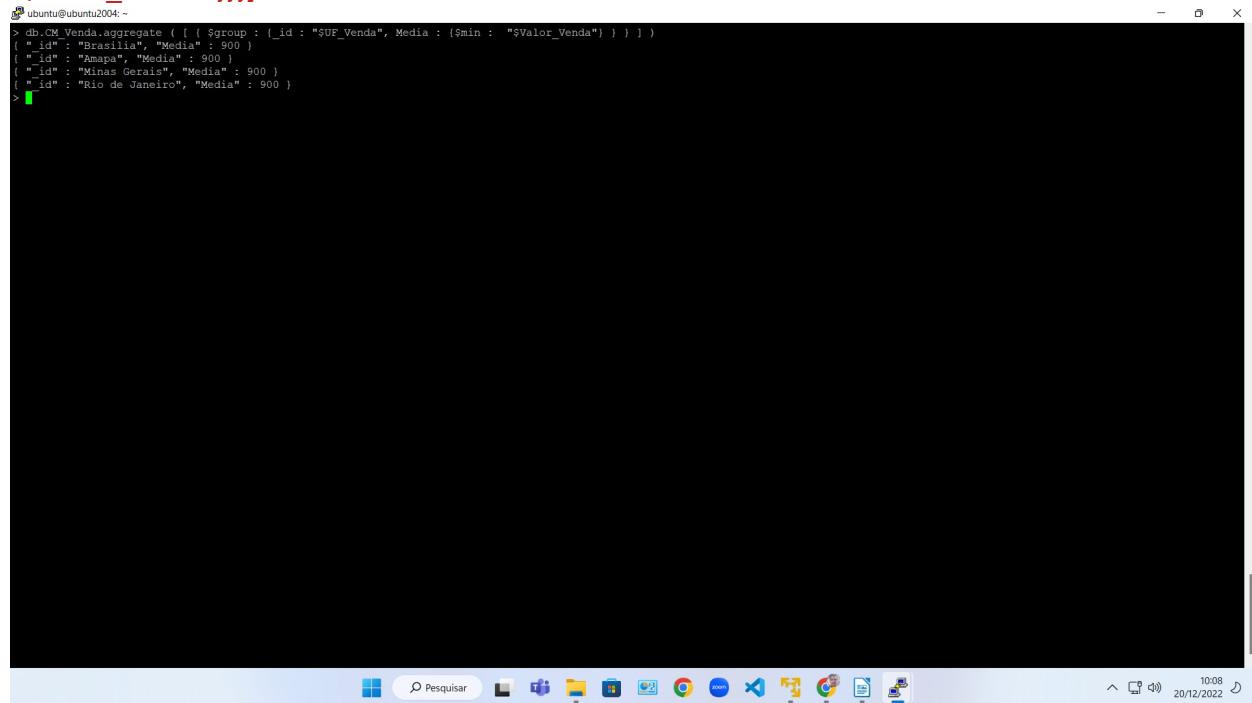
E. Criar uma consulta que mostre o maior valor de venda por UF.

```
db.CM_Venda.aggregate([{$group:{_id : "$UF_Venda", maximo_venda : {$max : "$Valor_Venda"}}}])
```

```
ubuntu@ubuntu2004:~\n> db.CM_Venda.aggregate ([ { $group : { _id : "$UF_Venda", Media : { $max : "$Valor_Venda" } } } ] )\n{ \"_id\" : \"Brasilia\", \"Media\" : 3000 }\n{ \"_id\" : \"Amapa\", \"Media\" : 3000 }\n{ \"_id\" : \"Minas Gerais\", \"Media\" : 3000 }\n{ \"_id\" : \"Rio de Janeiro\", \"Media\" : 3000 }\n> [
```

F. Criar uma consulta que mostre o menor valor de venda por UF.

```
db.CM_Venda.aggregate([{$group:{_id : "$UF_Venda", minimo_venda : {$min : "$Valor_Venda"}}}]
```



```
ubuntu@ubuntu2004:~$ db.CM_Venda.aggregate ( [ { $group : { _id : "$UF_Venda", Media : { $min : "$Valor_Venda" } } } ] )
{
  "_id": "Brasilia",
  "Media": 900
}
{
  "_id": "Amapa",
  "Media": 900
}
{
  "_id": "Minas Gerais",
  "Media": 900
}
{
  "_id": "Rio de Janeiro",
  "Media": 900
}
```

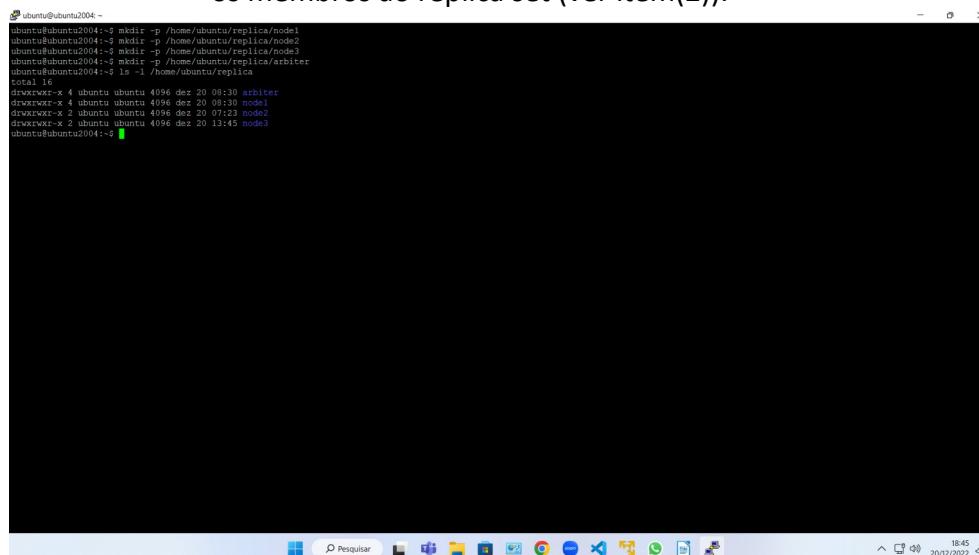
Replicação

1. Criar o replica set "&&_rsposmit".

2. Desenho da arquitetura:

- a.Um primário;
- b.Dois secundários;
- c.Um árbitro.Observações:

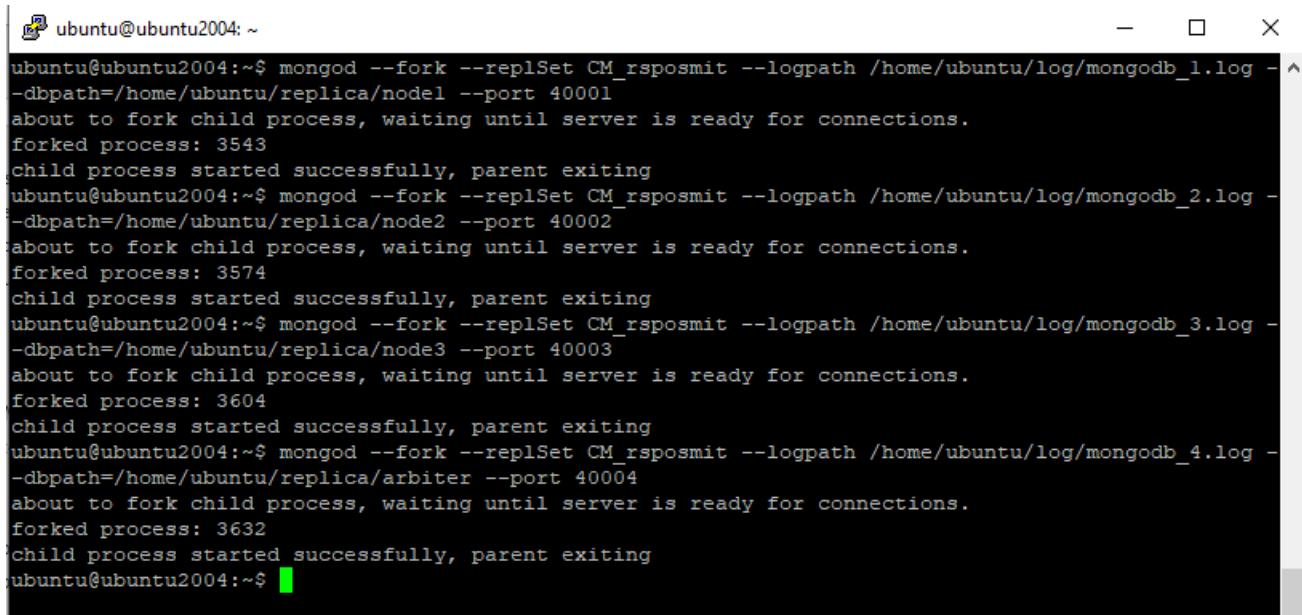
- I.Ativar os processos no nível do Sistema Operacional;
- II.Ativar o replica set através do MongoDB, lembrar de adicionar todos os membros ao replica set (ver item(2)).



```
ubuntu@ubuntu2004:~$ mkdir -p /home/ubuntu/replica/node1
ubuntu@ubuntu2004:~$ mkdir -p /home/ubuntu/replica/node2
ubuntu@ubuntu2004:~$ mkdir -p /home/ubuntu/replica/arbitro
ubuntu@ubuntu2004:~$ ls -l /home/ubuntu/replica
total 16
drwxrwxr-x 4 ubuntu ubuntu 4096 dez 20 08:30 arbitro
drwxrwxr-x 4 ubuntu ubuntu 4096 dez 20 08:30 node1
drwxrwxr-x 4 ubuntu ubuntu 4096 dez 20 07:23 node2
ubuntu@ubuntu2004:~$
```

I. Ativar o replica set na visão do sistema operacional.

```
mongod --fork --replSet CM_rsposmit --logpath /home/ubuntu/log/mongodb_1.log  
--dbpath=/home/ubuntu/replica/node1 --port 40001  
mongod --fork --replSet CM_rsposmit --logpath /home/ubuntu/log/mongodb_2.log  
--dbpath=/home/ubuntu/replica/node2 --port 40002  
mongod --fork --replSet CM_rsposmit --logpath /home/ubuntu/log/mongodb_3.log  
--dbpath=/home/ubuntu/replica/node3 --port 40003  
mongod --fork --replSet CM_rsposmit --logpath /home/ubuntu/log/mongodb_4.log  
--dbpath=/home/ubuntu/replica/arbiter --port 40004
```

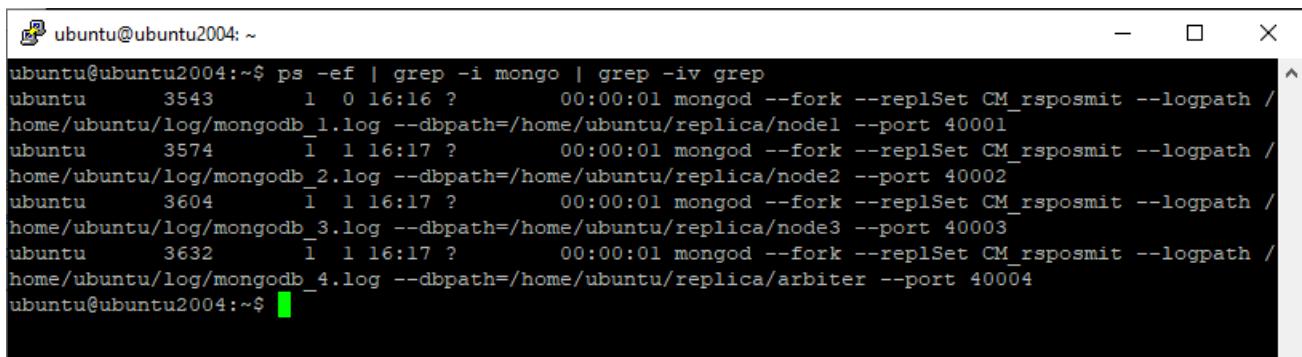


```
ubuntu@ubuntu2004:~$ mongod --fork --replSet CM_rsposmit --logpath /home/ubuntu/log/mongodb_1.log --dbpath=/home/ubuntu/replica/node1 --port 40001  
about to fork child process, waiting until server is ready for connections.  
forked process: 3543  
child process started successfully, parent exiting  
ubuntu@ubuntu2004:~$ mongod --fork --replSet CM_rsposmit --logpath /home/ubuntu/log/mongodb_2.log --dbpath=/home/ubuntu/replica/node2 --port 40002  
about to fork child process, waiting until server is ready for connections.  
forked process: 3574  
child process started successfully, parent exiting  
ubuntu@ubuntu2004:~$ mongod --fork --replSet CM_rsposmit --logpath /home/ubuntu/log/mongodb_3.log --dbpath=/home/ubuntu/replica/node3 --port 40003  
about to fork child process, waiting until server is ready for connections.  
forked process: 3604  
child process started successfully, parent exiting  
ubuntu@ubuntu2004:~$ mongod --fork --replSet CM_rsposmit --logpath /home/ubuntu/log/mongodb_4.log --dbpath=/home/ubuntu/replica/arbiter --port 40004  
about to fork child process, waiting until server is ready for connections.  
forked process: 3632  
child process started successfully, parent exiting  
ubuntu@ubuntu2004:~$
```

II. Ativar o replica set através do MongoDB, lembrar de adicionar todos os membros ao replica set (ver item (2)).

Iniciando o MongoDB Shell

```
mongo localhost:40001
```



```
ubuntu@ubuntu2004:~$ ps -ef | grep -i mongo | grep -iv grep  
ubuntu      3543      1  0 16:16 ?        00:00:01 mongod --fork --replSet CM_rsposmit --logpath /home/ubuntu/log/mongodb_1.log --dbpath=/home/ubuntu/replica/node1 --port 40001  
ubuntu      3574      1  1 16:17 ?        00:00:01 mongod --fork --replSet CM_rsposmit --logpath /home/ubuntu/log/mongodb_2.log --dbpath=/home/ubuntu/replica/node2 --port 40002  
ubuntu      3604      1  1 16:17 ?        00:00:01 mongod --fork --replSet CM_rsposmit --logpath /home/ubuntu/log/mongodb_3.log --dbpath=/home/ubuntu/replica/node3 --port 40003  
ubuntu      3632      1  1 16:17 ?        00:00:01 mongod --fork --replSet CM_rsposmit --logpath /home/ubuntu/log/mongodb_4.log --dbpath=/home/ubuntu/replica/arbiter --port 40004  
ubuntu@ubuntu2004:~$
```

Verificando novamente o status do replica set

rs_status()

```
ubuntu@ubuntu2004:~$ mongo localhost:40001
MongoDB shell version v3.6.8
connecting to: mongodb://localhost:40001/test
Implicit session: session { "id" : UUID("dbbcfaaa-6af7-4bf0-a0b4-49cf2e86884") }
MongoDB server version: 3.6.8
Server has startup warnings:
2022-12-20T16:16:59.144-0500 I STORAGE  [initandlisten]
2022-12-20T16:16:59.144-0500 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2022-12-20T16:16:59.144-0500 I STORAGE  [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2022-12-20T16:16:59.674-0500 I CONTROL  [initandlisten]
2022-12-20T16:16:59.674-0500 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2022-12-20T16:16:59.674-0500 I CONTROL  [initandlisten] ** Read and write access to data and configuration is unrestricted.
2022-12-20T16:16:59.675-0500 I CONTROL  [initandlisten]
2022-12-20T16:16:59.675-0500 I CONTROL  [initandlisten] ** WARNING: This server is bound to localhost.
2022-12-20T16:16:59.675-0500 I CONTROL  [initandlisten] ** Remote systems will be unable to connect to this server.
2022-12-20T16:16:59.675-0500 I CONTROL  [initandlisten] ** Start the server with --bind_ip <address> to specify which IP
2022-12-20T16:16:59.675-0500 I CONTROL  [initandlisten] ** addresses it should serve responses from, or with --bind_ip_all to
2022-12-20T16:16:59.675-0500 I CONTROL  [initandlisten] ** bind to all interfaces. If this behavior is desired, start the
2022-12-20T16:16:59.675-0500 I CONTROL  [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warning.
2022-12-20T16:16:59.675-0500 I CONTROL  [initandlisten]
>
```

```
ubuntu@ubuntu2004:~$ rs.initiate()
{
    "info2" : "no configuration specified. Using a default configuration for the set",
    "me" : "localhost:40001",
    "ok" : 1,
    "operationTime" : Timestamp(1671571313, 1),
    "$clusterTime" : {
        "clusterTime" : Timestamp(1671571313, 1),
        "signature" : {
            "hash" : BinData(0, "AAAAAAAAAAAAAAAAAAAAA="),
            "keyId" : NumberLong(0)
        }
    }
}
CM_rsposmit:OTHER>
CM_rsposmit:PRIMARY>
```

Mudando para o banco de dados CM_filmes

Use CM_filmes

```
ubuntu@ubuntu2004: ~
CM_rposmit:PRIMARY> use CM_filmes
switched to db CM_filmes
CM_rposmit:PRIMARY> db.createCollection("CM_col_filmes")
{
    "ok" : 1,
    "operationTime" : Timestamp(1671571500, 2),
    "$clusterTime" : {
        "clusterTime" : Timestamp(1671571500, 2),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAA="),
            "keyId" : NumberLong(0)
        }
    }
}
CM_rposmit:PRIMARY>
```

3. No nó primário criar a coleção "&&_col_filmes".

Inserir 5 documentos na coleção "&&_col_filmes".

```
db.CM_col_filmes.save({nome: "Adão Negro", genero: "Ação"})
```

```
db.CM_col_filmes.save({nome: "A Baleia Azul", genero: "Animação"})
```

```
db.CM_col_filmes.save({nome: "Vingadores: Guerra Infinita", genero: "Ação/Fantasia"})
```

```
db.CM_col_filmes.save({nome: "Somos tão Jovens", genero: "Biografia"})
```

```
db.CM_col_filmes.save({nome: "O Iluminado", genero: "Terror"})
```

```
ubuntu@ubuntu2004: ~
CM_rposmit:PRIMARY> db.CM_col_filmes.save({nome: "Adão Negro", genero: "Ação"})
WriteResult({ "nInserted" : 1 })
CM_rposmit:PRIMARY> db.CM_col_filmes.save({nome: "A Baleia Azul", genero: "Animação"})
WriteResult({ "nInserted" : 1 })
CM_rposmit:PRIMARY> db.CM_col_filmes.save({nome: "Vingadores: Guerra Infinita", genero: "Ação/Fantasia"})
WriteResult({ "nInserted" : 1 })
CM_rposmit:PRIMARY> db.CM_col_filmes.save({nome: "Somos tão Jovens", genero: "Biografia"})
WriteResult({ "nInserted" : 1 })
CM_rposmit:PRIMARY> db.CM_col_filmes.save({nome: "O Iluminado", genero: "Terror"})
WriteResult({ "nInserted" : 1 })
CM_rposmit:PRIMARY> db.CM_col_filmes.find()
[{"_id": ObjectId("63a22961b4376eb59cce7a96"), "nome": "Adão Negro", "genero": "Ação"}, {"_id": ObjectId("63a22961b4376eb59cce7a97"), "nome": "A Baleia Azul", "genero": "Animação"}, {"_id": ObjectId("63a22961b4376eb59cce7a98"), "nome": "Vingadores: Guerra Infinita", "genero": "Ação/Fantasia"}, {"_id": ObjectId("63a22961b4376eb59cce7a99"), "nome": "Somos tão Jovens", "genero": "Biografia"}, {"_id": ObjectId("63a22963b4376eb59cce7a9a"), "nome": "O Iluminado", "genero": "Terror"}]
CM_rposmit:PRIMARY>
```

4. Acessar um dos nós secundários.

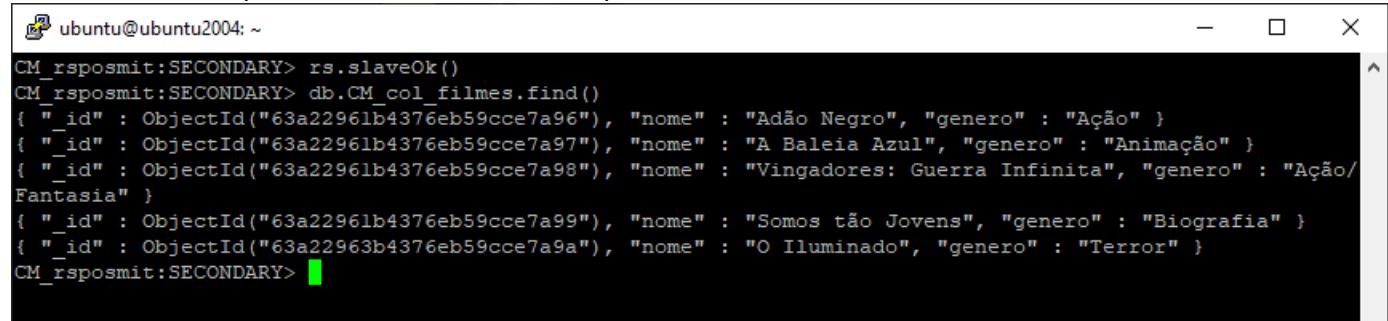
Efetuar a leitura da coleção "&&_col_filmes".

mongo localhost:40002

```
ubuntu@ubuntu2004:~$ mongo localhost:40002
MongoDB shell version v3.6.8
connecting to: mongodb://localhost:40002/test
Implicit session: session { "id" : UUID("808cee79-24b4-4279-8b56-a641f00a5148")}
}
MongoDB server version: 3.6.8
Server has startup warnings:
2022-12-20T16:17:28.320-0500 I STORAGE  [initandlisten]
2022-12-20T16:17:28.320-0500 I STORAGE  [initandlisten] ** WARNING: Using the XF
S filesystem is strongly recommended with the WiredTiger storage engine
2022-12-20T16:17:28.320-0500 I STORAGE  [initandlisten] **           See http://d
ochub.mongodb.org/core/prodnotes-filesystem
2022-12-20T16:17:28.882-0500 I CONTROL  [initandlisten]
2022-12-20T16:17:28.882-0500 I CONTROL  [initandlisten] ** WARNING: Access contr
ol is not enabled for the database.
2022-12-20T16:17:28.882-0500 I CONTROL  [initandlisten] **           Read and wri
te access to data and configuration is unrestricted.
2022-12-20T16:17:28.882-0500 I CONTROL  [initandlisten]
2022-12-20T16:17:28.882-0500 I CONTROL  [initandlisten] ** WARNING: This server
is bound to localhost.
2022-12-20T16:17:28.882-0500 I CONTROL  [initandlisten] **           Remote syste
ms will be unable to connect to this server.
2022-12-20T16:17:28.882-0500 I CONTROL  [initandlisten] **           Start the se
rver with --bind_ip <address> to specify which IP
2022-12-20T16:17:28.882-0500 I CONTROL  [initandlisten] **           addresses it
should serve responses from, or with --bind_ip_all to
2022-12-20T16:17:28.882-0500 I CONTROL  [initandlisten] **           bind to all
interfaces. If this behavior is desired, start the
2022-12-20T16:17:28.882-0500 I CONTROL  [initandlisten] **           server with
--bind_ip 127.0.0.1 to disable this warning.
2022-12-20T16:17:28.883-0500 I CONTROL  [initandlisten]
CM_rposmit:SECONDARY> █
```

```
ubuntu@ubuntu2004:~$ CM_rposmit:SECONDARY> use CM_filmes
switched to db CM_filmes
CM_rposmit:SECONDARY> db.CM_col_filmes.find()
Error: error: {
    "operationTime" : Timestamp(1671571984, 1),
    "ok" : 0,
    "errmsg" : "not master and slaveOk=false",
    "code" : 13435,
    "codeName" : "NotMasterNoSlaveOk",
    "$clusterTime" : {
        "clusterTime" : Timestamp(1671571984, 1),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAA="),
            "keyId" : NumberLong(0)
        }
    }
}
CM_rposmit:SECONDARY> █
```

Habilitar a leitura para o nós secundário do replica set



```
ubuntu@ubuntu2004: ~
CM_rsposmit:SECONDARY> rs.slaveOk()
CM_rsposmit:SECONDARY> db.CM_col_filmes.find()
{ "_id" : ObjectId("63a22961b4376eb59cce7a96") , "nome" : "Adão Negro", "genero" : "Ação" }
{ "_id" : ObjectId("63a22961b4376eb59cce7a97") , "nome" : "A Baleia Azul", "genero" : "Animação" }
{ "_id" : ObjectId("63a22961b4376eb59cce7a98") , "nome" : "Vingadores: Guerra Infinita", "genero" : "Ação/Fantasia" }
{ "_id" : ObjectId("63a22961b4376eb59cce7a99") , "nome" : "Somos tão Jovens", "genero" : "Biografia" }
{ "_id" : ObjectId("63a22963b4376eb59cce7a9a") , "nome" : "O Iluminado", "genero" : "Terror" }
CM_rsposmit:SECONDARY>
```

E. Particionamento

1.Criar um shard.

2.Desenho da arquitetura:

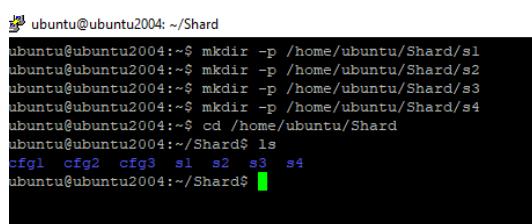
a.Três config servers (no Replica Set);

b.Quatro shard servers (sem Replica Set);

c.Um mongos.Observações:

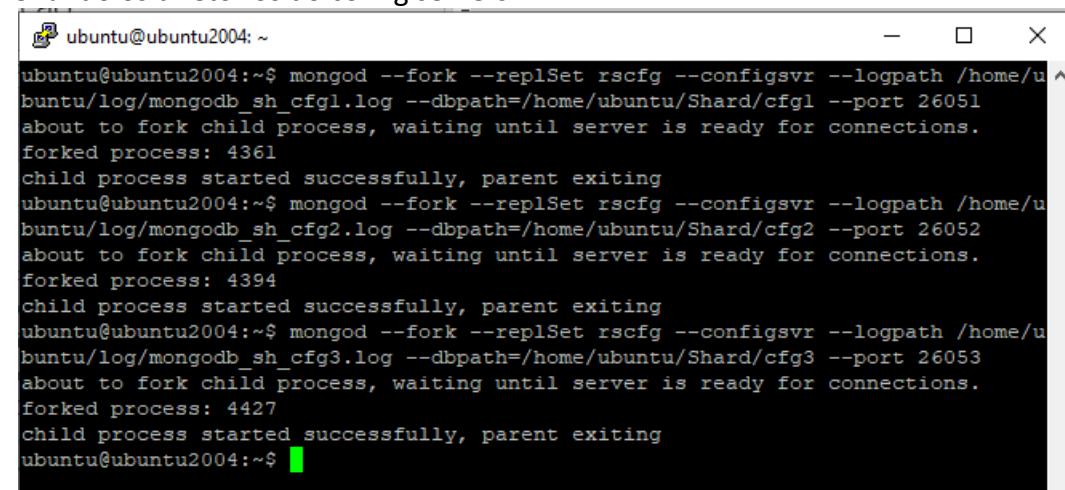
I.Ativar os processos no nível do Sistema Operacional;

II.Ativar o shard através do MongoDB, lembrar de adicionar os shard servers (ver item(2.b)).



```
ubuntu@ubuntu2004: ~/Shard
ubuntu@ubuntu2004:~$ mkdir -p /home/ubuntu/Shard/s1
ubuntu@ubuntu2004:~$ mkdir -p /home/ubuntu/Shard/s2
ubuntu@ubuntu2004:~$ mkdir -p /home/ubuntu/Shard/s3
ubuntu@ubuntu2004:~$ mkdir -p /home/ubuntu/Shard/s4
ubuntu@ubuntu2004:~$ cd /home/ubuntu/Shard
ubuntu@ubuntu2004:~/Shard$ ls
cfg1 cfg2 cfg3 s1 s2 s3 s4
ubuntu@ubuntu2004:~/Shard$
```

Criando os diretórios do config servers



```
ubuntu@ubuntu2004: ~
ubuntu@ubuntu2004:~$ mongod --fork --replSet rscfg --configsvr --logpath /home/ubuntu/log/mongodb_sh_cfg1.log --dbpath=/home/ubuntu/Shard/cfg1 --port 26051
about to fork child process, waiting until server is ready for connections.
forked process: 4361
child process started successfully, parent exiting
ubuntu@ubuntu2004:~$ mongod --fork --replSet rscfg --configsvr --logpath /home/ubuntu/log/mongodb_sh_cfg2.log --dbpath=/home/ubuntu/Shard/cfg2 --port 26052
about to fork child process, waiting until server is ready for connections.
forked process: 4394
child process started successfully, parent exiting
ubuntu@ubuntu2004:~$ mongod --fork --replSet rscfg --configsvr --logpath /home/ubuntu/log/mongodb_sh_cfg3.log --dbpath=/home/ubuntu/Shard/cfg3 --port 26053
about to fork child process, waiting until server is ready for connections.
forked process: 4427
child process started successfully, parent exiting
ubuntu@ubuntu2004:~$
```

```

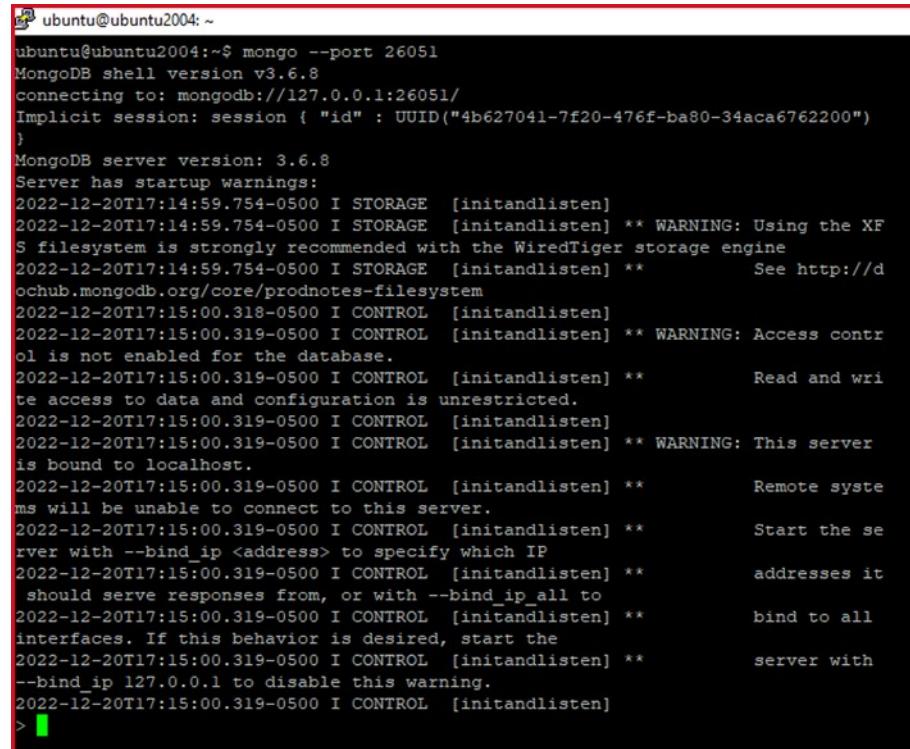
mongod --fork --replSet rscfg --configsvr --logpath /home/ubuntu/log/mongodb_sh_cfg1.log --
dbpath=/home/ubuntu/Shard/cfg1 --port 26051

mongod --fork --replSet rscfg --configsvr --logpath /home/ubuntu/log/mongodb_sh_cfg2.log --
dbpath=/home/ubuntu/Shard/cfg2 --port 26052

mongod --fork --replSet rscfg --configsvr --logpath /home/ubuntu/log/mongodb_sh_cfg3.log --
dbpath=/home/ubuntu/Shard/cfg3 --port 26053

```

mongo --port 26051



```

ubuntu@ubuntu2004:~$ mongo --port 26051
MongoDB shell version v3.6.8
connecting to: mongodb://127.0.0.1:26051/
Implicit session: session { "id" : UUID("4b627041-7f20-476f-ba80-34aca6762200") }
}
MongoDB server version: 3.6.8
Server has startup warnings:
2022-12-20T17:14:59.754-0500 I STORAGE  [initandlisten]
2022-12-20T17:14:59.754-0500 I STORAGE  [initandlisten] ** WARNING: Using the XF
S filesystem is strongly recommended with the WiredTiger storage engine
2022-12-20T17:14:59.754-0500 I STORAGE  [initandlisten] **             See http://d
ochub.mongodb.org/core/prodnotes-filesystem
2022-12-20T17:15:00.318-0500 I CONTROL  [initandlisten]
2022-12-20T17:15:00.319-0500 I CONTROL  [initandlisten] ** WARNING: Access contr
ol is not enabled for the database.
2022-12-20T17:15:00.319-0500 I CONTROL  [initandlisten] **             Read and wri
te access to data and configuration is unrestricted.
2022-12-20T17:15:00.319-0500 I CONTROL  [initandlisten]
2022-12-20T17:15:00.319-0500 I CONTROL  [initandlisten] ** WARNING: This server
is bound to localhost.
2022-12-20T17:15:00.319-0500 I CONTROL  [initandlisten] **             Remote syste
ms will be unable to connect to this server.
2022-12-20T17:15:00.319-0500 I CONTROL  [initandlisten] **             Start the se
rver with --bind_ip <address> to specify which IP
2022-12-20T17:15:00.319-0500 I CONTROL  [initandlisten] **             addresses it
2022-12-20T17:15:00.319-0500 I CONTROL  [initandlisten] **             bind to all
interfaces. If this behavior is desired, start the
2022-12-20T17:15:00.319-0500 I CONTROL  [initandlisten] **             server with
--bind_ip 127.0.0.1 to disable this warning.
2022-12-20T17:15:00.319-0500 I CONTROL  [initandlisten]
>

```

Iniciar o replica Set

```

rs.initiate({
    _id: "rscfg",      configsvr: true,
    members: [
        { _id : 0, host : "localhost:26051" },
        { _id : 1, host : "localhost:26052" },
        { _id : 2, host : "localhost:26053" }
    ]}
}
```

```
ubuntu@ubuntu2004:~$ mongod --fork --shardsvr --logpath /home/ubuntu/log/mongodb_sh_ss1.log --dbpath=/home/ubuntu/Shard/s1 --port 27051
about to fork child process, waiting until server is ready for connections.
forked process: 4672
child process started successfully, parent exiting
ubuntu@ubuntu2004:~$ mongod --fork --shardsvr --logpath /home/ubuntu/log/mongodb_sh_ss2.log --dbpath=/home/ubuntu/Shard/s2 --port 27052
about to fork child process, waiting until server is ready for connections.
forked process: 4700
child process started successfully, parent exiting
ubuntu@ubuntu2004:~$ mongod --fork --shardsvr --logpath /home/ubuntu/log/mongodb_sh_ss3.log --dbpath=/home/ubuntu/Shard/s3 --port 27053
about to fork child process, waiting until server is ready for connections.
forked process: 4725
child process started successfully, parent exiting
ubuntu@ubuntu2004:~$ mongod --fork --shardsvr --logpath /home/ubuntu/log/mongodb_sh_ss4.log --dbpath=/home/ubuntu/Shard/s4 --port 27054
about to fork child process, waiting until server is ready for connections.
forked process: 4755
child process started successfully, parent exiting
ubuntu@ubuntu2004:~$
```

Vinculado ao mongos (MongoDB Shard)

```
mongos --fork --configdb "rscfg/localhost:26051,localhost:26052,localhost:26053" --logpath
/home/ubuntu/log/mongos_sh.log --port 28000
```

```
ubuntu@ubuntu2004:~$ mongos --fork --configdb "rscfg/localhost:26051,localhost:26052,localhost:26053" --logpath /home/ubuntu/log/mongos_sh.log --port 28000
about to fork child process, waiting until server is ready for connections.
forked process: 4805
child process started successfully, parent exiting
ubuntu@ubuntu2004:~$ █
```

3. Conectar ao mongos.

- a. Ativar o particionamento para um database de sua escolha.
- b. Particionar uma coleção da sua escolha.
- c. Inserir para essa coleção 1.000 documentos através do comando [for].
- d. Mostrar a distribuição da coleção criada.

mongo localhost:28000

```
ubuntu@ubuntu2004: ~
ubuntu@ubuntu2004:~$ mongo localhost:28000
MongoDB shell version v3.6.8
connecting to: mongodb://localhost:28000/test
Implicit session: session { "id" : UUID("9e84078f-6415-44e3-860b-67c6f845609f") }
MongoDB server version: 3.6.8
Server has startup warnings:
2022-12-20T17:35:53.788-0500 I CONTROL  [main]
2022-12-20T17:35:53.789-0500 I CONTROL  [main] ** WARNING: Access control is not enabled for the database.
2022-12-20T17:35:53.789-0500 I CONTROL  [main] ** Read and write access to data and configuration is unrestricted.
2022-12-20T17:35:53.789-0500 I CONTROL  [main]
2022-12-20T17:35:53.789-0500 I CONTROL  [main] ** WARNING: This server is bound to localhost.
2022-12-20T17:35:53.789-0500 I CONTROL  [main] ** Remote systems will be unable to connect to this server.
2022-12-20T17:35:53.789-0500 I CONTROL  [main] ** Start the server with --bind_ip <address> to specify which IP
2022-12-20T17:35:53.789-0500 I CONTROL  [main] ** addresses it should serve responses from, or with --bind_ip_all to
2022-12-20T17:35:53.789-0500 I CONTROL  [main] ** bind to all interfaces. If this behavior is desired, start the
2022-12-20T17:35:53.789-0500 I CONTROL  [main] ** server with --bind_ip 127.0.0.1 to disable this warning.
2022-12-20T17:35:53.789-0500 I CONTROL  [main]
mongos> |
```

Acessando o banco de dados config

`use config`

Mostrando coleções deste database

`show collections`

Verificando a coleção shards

`db.shards.find()`

Adicionando o shard 1 indicando máquina:porta

`sh.addShard("localhost:27051")`

Adicionando o shard 2 indicando máquina:porta

`sh.addShard("localhost:27052")`

Adicionando o shard 3 indicando máquina:porta

`sh.addShard("localhost:27053")`

Adicionando o shard 4 indicando máquina:porta

`sh.addShard("localhost:27054")`

```
ubuntu@ubuntu2004: ~
mongos> use config
switched to db config
mongos>
mongos> show collections
chunks
lockpings
locks
migrations
mongos
shards
tags
transactions
version
mongos>
mongos> db.shards.find()
mongos>
mongos> sh.addShard("localhost:27051")
{
    "shardAdded" : "shard0000",
    "ok" : 1,
    "operationTime" : Timestamp(1671576934, 5),
    "$clusterTime" : {
        "clusterTime" : Timestamp(1671576934, 5),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAA="),
            "keyId" : NumberLong(0)
        }
    }
}
mongos> sh.addShard("localhost:27052")
{
    "shardAdded" : "shard0001",
    "ok" : 1,
    "operationTime" : Timestamp(1671576940, 3),
    "$clusterTime" : {
        "clusterTime" : Timestamp(1671576940, 3),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAA="),
            "keyId" : NumberLong(0)
        }
    }
}
mongos> sh.addShard("localhost:27053")
{
    "shardAdded" : "shard0002",
    "ok" : 1,
    "operationTime" : Timestamp(1671576946, 3),
    "$clusterTime" : {
        "clusterTime" : Timestamp(1671576946, 3),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAA="),
            "keyId" : NumberLong(0)
        }
    }
}
mongos> sh.addShard("localhost:27054")
{
    "shardAdded" : "shard0003",
    "ok" : 1,
    "operationTime" : Timestamp(1671576952, 3),
    "$clusterTime" : {
        "clusterTime" : Timestamp(1671576952, 3),
        "signature" : {
```

Ativar o particionamento para um database de sua escolha.

```
mongos> use loja
switched to db loja
mongos>
mongos> sh.enableSharding("loja")
{
    "ok" : 1,
    "operationTime" : Timestamp(1671577591, 9),
    "$clusterTime" : {
        "clusterTime" : Timestamp(1671577591, 9),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAA="),
            "keyId" : NumberLong(0)
        }
    }
}
mongos> █
```

Particionar uma coleção da sua escolha.

```
sh.shardCollection ("loja.clientes", {_id:1}, true)
mongos> sh.shardCollection ("loja.clientes", {_id:1}, true)
{
    "collectionsharded" : "loja.clientes",
    "collectionUUID" : UUID("e4c69bb5-d280-47c1-a829-54a07c7e6db0"),
    "ok" : 1,
    "operationTime" : Timestamp(1671577726, 11),
    "$clusterTime" : {
        "clusterTime" : Timestamp(1671577726, 11),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAA="),
            "keyId" : NumberLong(0)
        }
    }
}
mongos> █
```

Inserir para essa coleção 1.000 documentos através do comando [for].

```
for (var i=0; i < 1000; i++ ) { db.clientes.insert ( { x:i, y:3, uf :"RJ" } ) }
mongos> for (var i=0; i < 1000; i++ ) { db.clientes.insert ( { x:i, y:3, uf :"RJ" } ) }
WriteResult({ "nInserted" : 1 })
mongos> █
```

Mostrar a distribuição da coleção criada.

```
db.c.getShardDistribution()
mongos> db.clientes.getShardDistribution()

Shard shard0001 at localhost:27052
  data : 53KiB docs : 1000 chunks : 1
  estimated data per chunk : 53KiB
  estimated docs per chunk : 1000

Totals
  data : 53KiB docs : 1000 chunks : 1
  Shard shard0001 contains 100% data, 100% docs in cluster, avg obj size on shard : 55B

mongos> █
```

Storage Engines

1. Criar uma instância do MongoDB que use o storage engine mmapv1.

`mkdir /home/ubuntu/se_mmap`

```
ubuntu@ubuntu2004:~$ mkdir /home/ubuntu/se_mmap
ubuntu@ubuntu2004:~$ mongod --fork --storageEngine mmapv1 --logpath /home/ubuntu/log/mongodb_se_mmap.log --dbpath=/home/ubuntu/se_mmap --port 28000
about to fork child process, waiting until server is ready for connections.
forked process: 48950
child process started successfully, parent exiting
ubuntu@ubuntu2004:~$
```

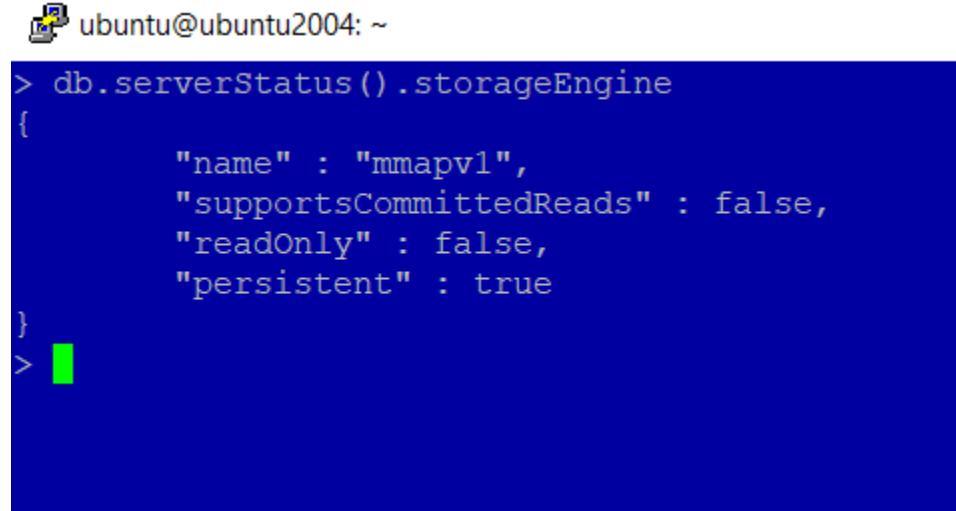
- A. Conectar a essa instância;

`mongod --fork --storageEngine mmapv1 --logpath /home/ubuntu/log/mongodb_se_mmap.log --dbpath=/home/ubuntu/se_mmap --port 28000`

```
ubuntu@ubuntu2004:~$ mongo localhost:29000/teste_journal
MongoDB shell version v3.6.8
connecting to: mongodb://localhost:29000/teste_journal
Implicit session: session { "id" : UUID("c898a400 52c0 4ba7 982f b662ad69fc68") }
MongoDB server version: 3.6.8
Server has startup warnings:
2022-12-21T15:16:16.825-0500 I STORAGE  [initandlisten]
2022-12-21T15:16:16.825-0500 I STORAGE  [initandlisten] ** WARNING: Using the XFS file
system is strongly recommended with the WiredTiger storage engine
2022-12-21T15:16:16.826-0500 I STORAGE  [initandlisten] **             see http://dochu
b.mongodb.org/core/prodnotes-filesystem
2022-12-21T15:16:17.674-0500 I CONTROL  [initandlisten]
2022-12-21T15:16:17.674-0500 I CONTROL  [initandlisten] ** WARNTNG: Access control i
s not enabled for the database.
2022-12-21T15:16:17.674-0500 I CONTROL  [initandlisten] **             Read and write a
ccess to data and configuration is unrestricted.
2022-12-21T15:16:17.674-0500 I CONTROL  [initandlisten]
2022-12-21T15:16:17.674-0500 I CONTROL  [initandlisten] ** WARNING: This server is b
ound to localhost.
2022-12-21T15:16:17.674-0500 I CONTROL  [initandlisten] **             Remote systems w
ill be unable to connect to this server.
2022-12-21T15:16:17.674-0500 I CONTROL  [initandlisten] **             Start the server
with --bind_ip <address> to specify which IP
2022-12-21T15:16:17.674-0500 I CONTROL  [initandlisten] **             addresses it sho
uld serve responses from, or with --bind_ip all to
2022-12-21T15:16:17.674-0500 I CONTROL  [initandlisten] **             bind to all inte
rfaces. If this behavior is desired, start the
2022-12-21T15:16:17.674-0500 I CONTROL  [initandlisten] **             server with --bi
nd_ip 127.0.0.1 to disable this warning.
2022-12-21T15:16:17.675-0500 I CONTROL  [initandlisten]
```

B. Verificar o storage engine corrente;

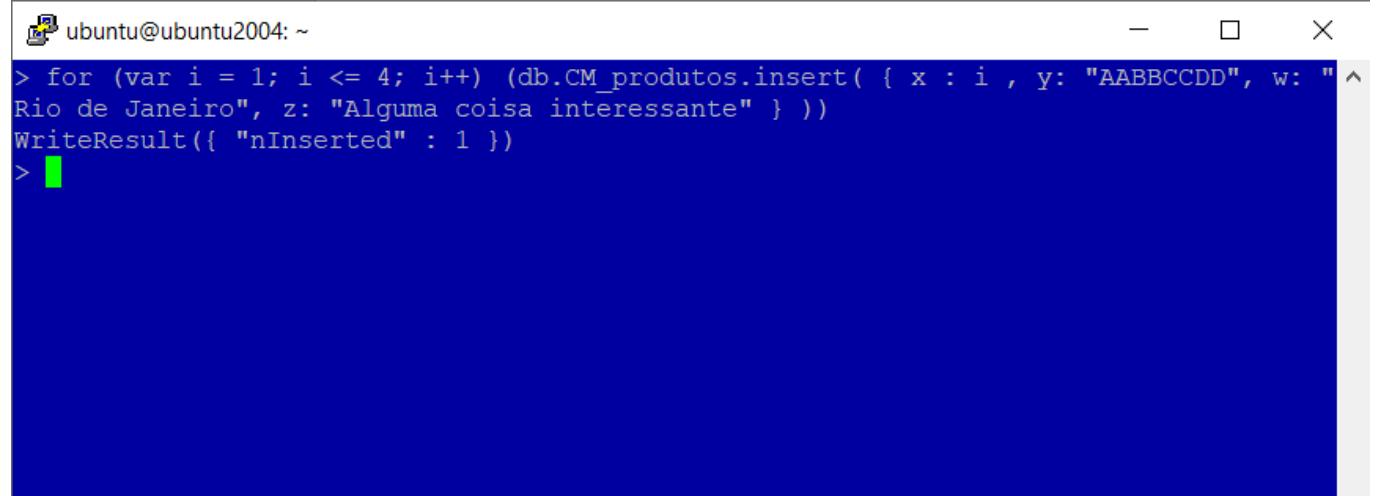
```
db.serverStatus().storageEngine
```



```
ubuntu@ubuntu2004: ~
> db.serverStatus().storageEngine
{
    "name" : "mmapv1",
    "supportsCommittedReads" : false,
    "readOnly" : false,
    "persistent" : true
}
>
```

C. Criar a coleção “&&_produtos” e inserir 4 documentos.

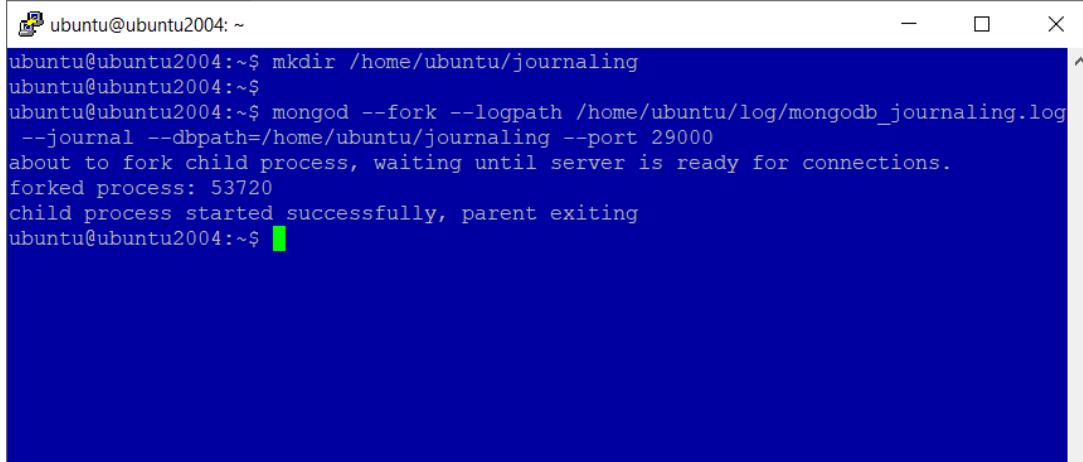
```
for (var i = 1; i <= 4; i++) (db.cadastro.insert( { x : i , y: "AABBCCDD", w: "Rio de Janeiro", z: "Alguma coisa interessante" } ))
```



```
ubuntu@ubuntu2004: ~
-
X
> for (var i = 1; i <= 4; i++) (db.CM_produtos.insert( { x : i , y: "AABBCCDD", w: "Rio de Janeiro", z: "Alguma coisa interessante" } ))
WriteResult({ "nInserted" : 1 })
>
```

2. Criar uma instância do MongoDB que use o storage engine wiredTiger.

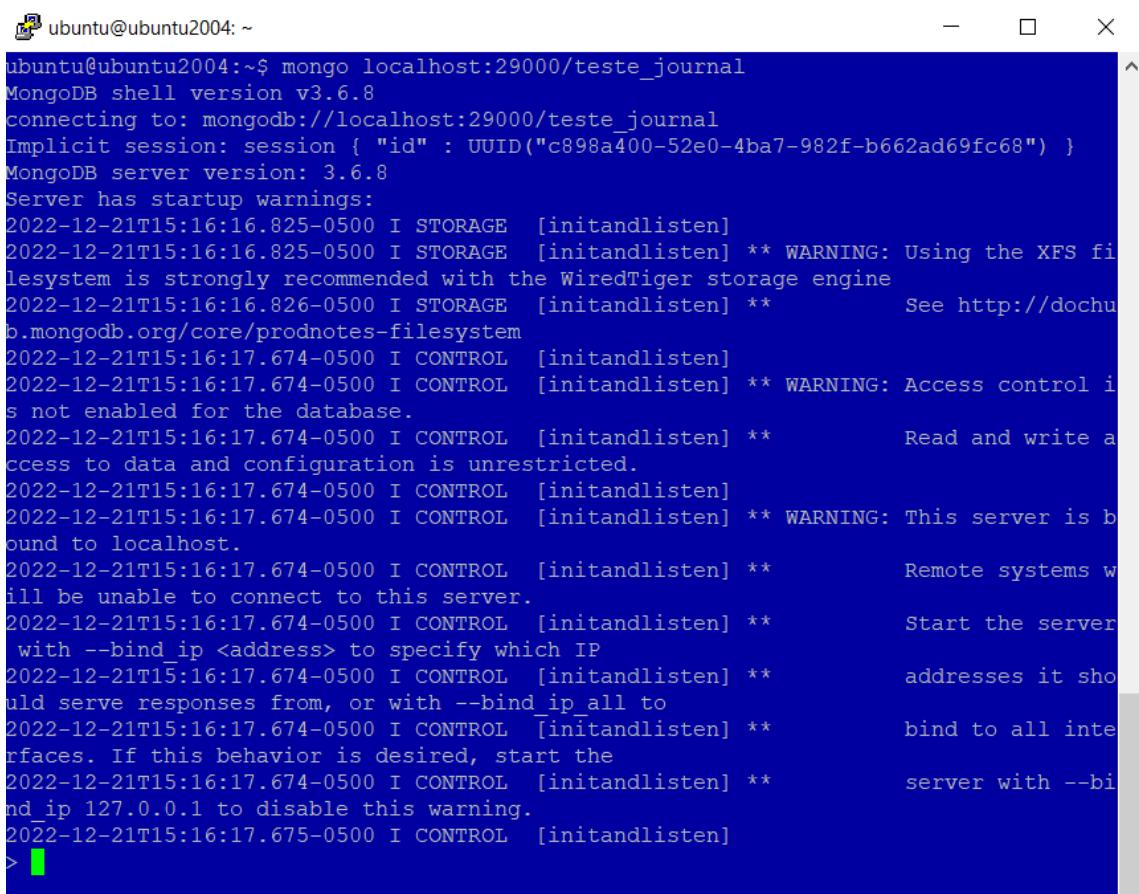
```
mongod --fork --logpath /home/ubuntu/log/mongodb_journaling.log --journal  
--dbpath=/home/ubuntu/journaling --port 29000
```



```
ubuntu@ubuntu2004:~$ mkdir /home/ubuntu/journaling  
ubuntu@ubuntu2004:~$ mongod --fork --logpath /home/ubuntu/log/mongodb_journaling.log  
--journal --dbpath=/home/ubuntu/journaling --port 29000  
about to fork child process, waiting until server is ready for connections.  
forked process: 53720  
child process started successfully, parent exiting  
ubuntu@ubuntu2004:~$
```

A. Conectar a essa instância;

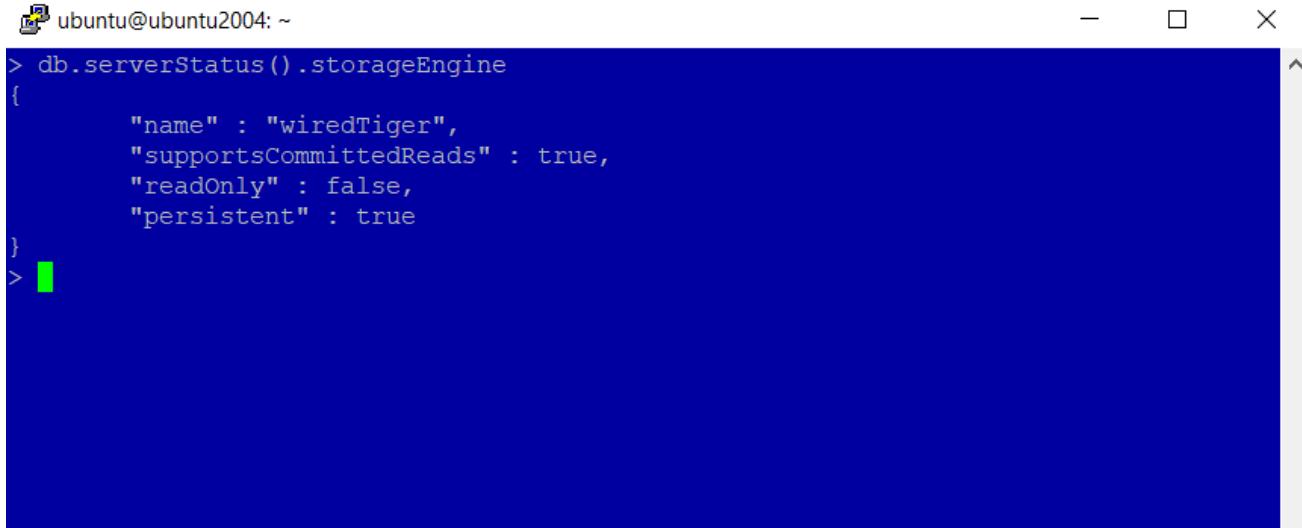
```
mongo localhost:29000/teste_journal
```



```
ubuntu@ubuntu2004:~$ mongo localhost:29000/teste_journal  
MongoDB shell version v3.6.8  
connecting to: mongodb://localhost:29000/teste_journal  
Implicit session: session { "id" : UUID("c898a400-52e0-4ba7-982f-b662ad69fc68") }  
MongoDB server version: 3.6.8  
Server has startup warnings:  
2022-12-21T15:16:16.825-0500 I STORAGE [initandlisten]  
2022-12-21T15:16:16.825-0500 I STORAGE [initandlisten] ** WARNING: Using the XFS file  
system is strongly recommended with the WiredTiger storage engine  
2022-12-21T15:16:16.826-0500 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem  
2022-12-21T15:16:17.674-0500 I CONTROL [initandlisten]  
2022-12-21T15:16:17.674-0500 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.  
2022-12-21T15:16:17.674-0500 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.  
2022-12-21T15:16:17.674-0500 I CONTROL [initandlisten]  
2022-12-21T15:16:17.674-0500 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.  
2022-12-21T15:16:17.674-0500 I CONTROL [initandlisten] ** Remote systems will be unable to connect to this server.  
2022-12-21T15:16:17.674-0500 I CONTROL [initandlisten] ** Start the server with --bind_ip <address> to specify which IP  
2022-12-21T15:16:17.674-0500 I CONTROL [initandlisten] ** addresses it should serve responses from, or with --bind_ip all to bind to all interfaces. If this behavior is desired, start the  
2022-12-21T15:16:17.674-0500 I CONTROL [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warning.  
2022-12-21T15:16:17.675-0500 I CONTROL [initandlisten]
```

B. Verificar o storage engine corrente;

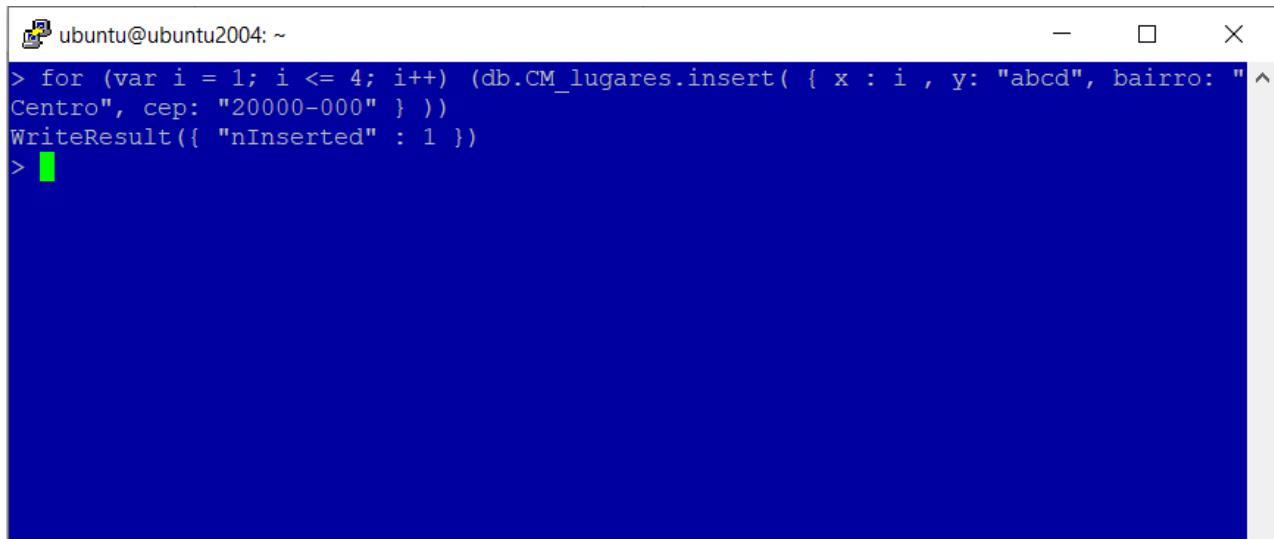
db.serverStatus().storageEngine



```
ubuntu@ubuntu2004: ~
> db.serverStatus().storageEngine
{
    "name" : "wiredTiger",
    "supportsCommittedReads" : true,
    "readOnly" : false,
    "persistent" : true
}
> [REDACTED]
```

C. Criar a coleção “`&&_lugares`” e inserir 4 documentos.

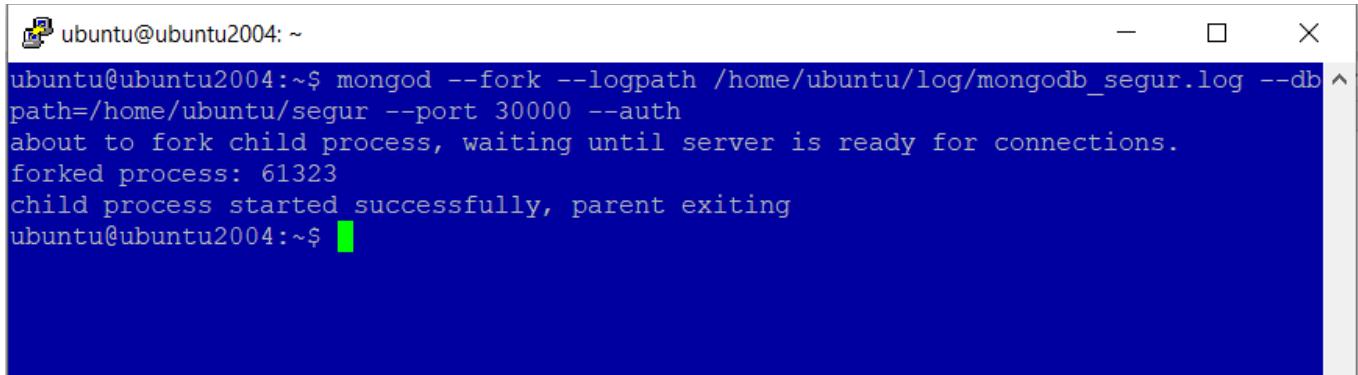
```
for (var i = 1; i <= 4; i++) (db.CM_lugares.insert( { x : i, y: "abcd", bairro: "Centro", cep: "20000-000" } ))
```



```
ubuntu@ubuntu2004: ~
> for (var i = 1; i <= 4; i++) (db.CM_lugares.insert( { x : i , y: "abcd", bairro: "Centro", cep: "20000-000" } ))
WriteResult({ "nInserted" : 1 })
> [REDACTED]
```

3. Criar uma instância do MongoDB que use segurança (autenticação e autorização)

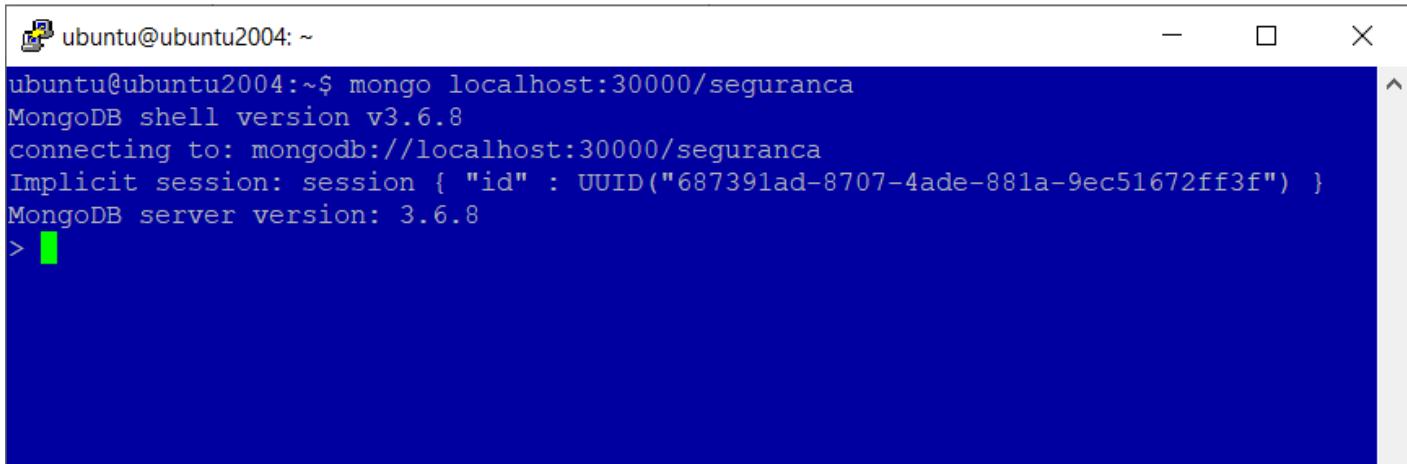
```
mongod --fork --logpath /home/ubuntu/log/mongodb_segur.log --dbpath=/home/ubuntu/segu  
--port 30000 --auth
```



```
ubuntu@ubuntu2004:~$ mongod --fork --logpath /home/ubuntu/log/mongodb_segur.log --db  
path=/home/ubuntu/segu --port 30000 --auth  
about to fork child process, waiting until server is ready for connections.  
forked process: 61323  
child process started successfully, parent exiting  
ubuntu@ubuntu2004:~$
```

A. Conectar a essa instância;

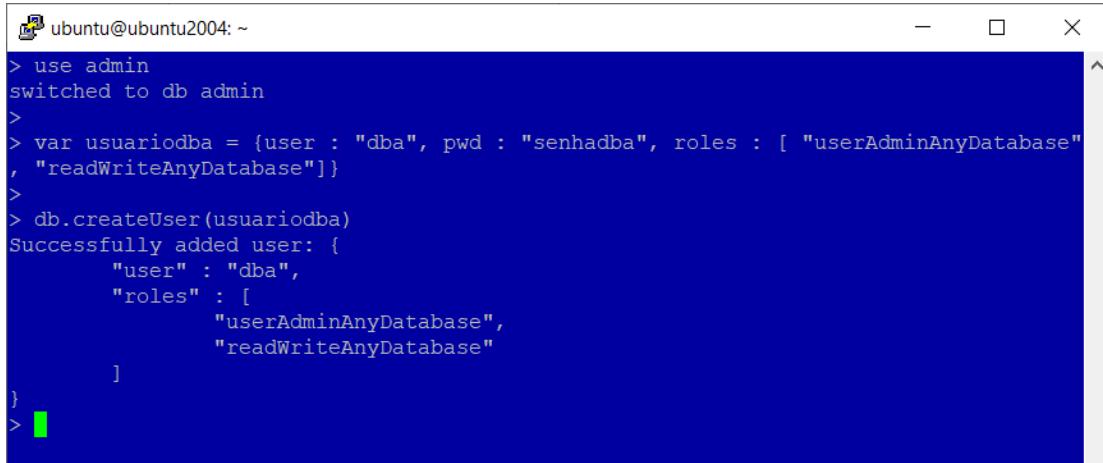
```
mongo localhost:30000/seguranca
```



```
ubuntu@ubuntu2004:~$ mongo localhost:30000/seguranca  
MongoDB shell version v3.6.8  
connecting to: mongodb://localhost:30000/seguranca  
Implicit session: session { "id" : UUID("687391ad-8707-4ade-881a-9ec51672ff3f") }  
MongoDB server version: 3.6.8  
>
```

B. Criar o usuário dba com a role root;

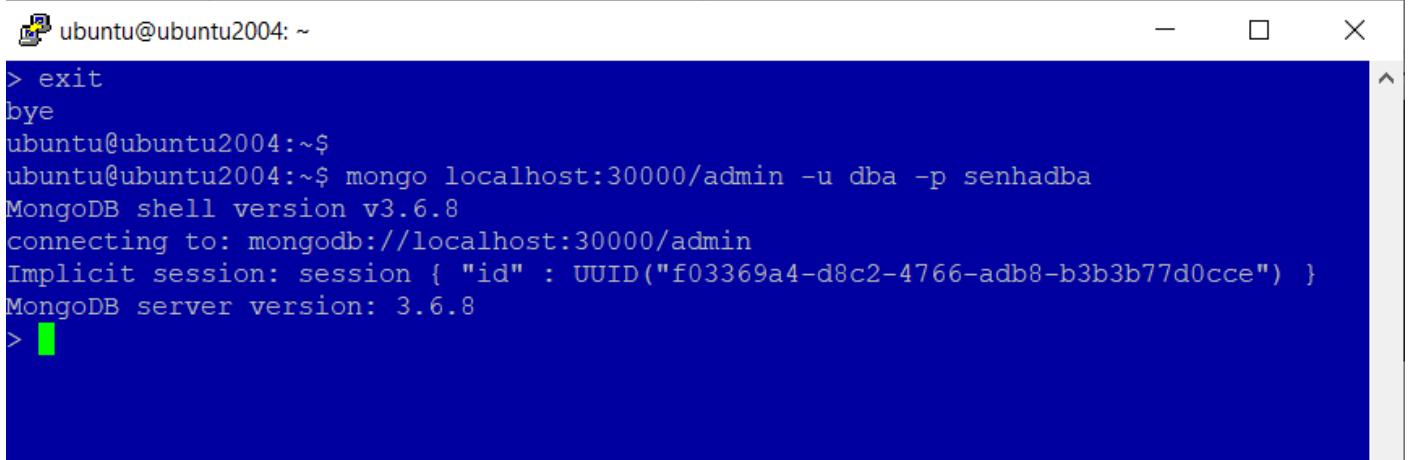
```
var usuariodba = { user : "dba", pwd : "senhadba", roles : [ "userAdminAnyDatabase",  
"readWriteAnyDatabase" ]}
```



```
ubuntu@ubuntu2004: ~  
> use admin  
switched to db admin  
>  
> var usuariodba = {user : "dba", pwd : "senhadba", roles : [ "userAdminAnyDatabase"  
, "readWriteAnyDatabase"]}  
>  
> db.createUser(usuariodba)  
Successfully added user: {  
    "user" : "dba",  
    "roles" : [  
        "userAdminAnyDatabase",  
        "readWriteAnyDatabase"  
    ]  
}  
> [
```

C. Conectar com o usuário dba;

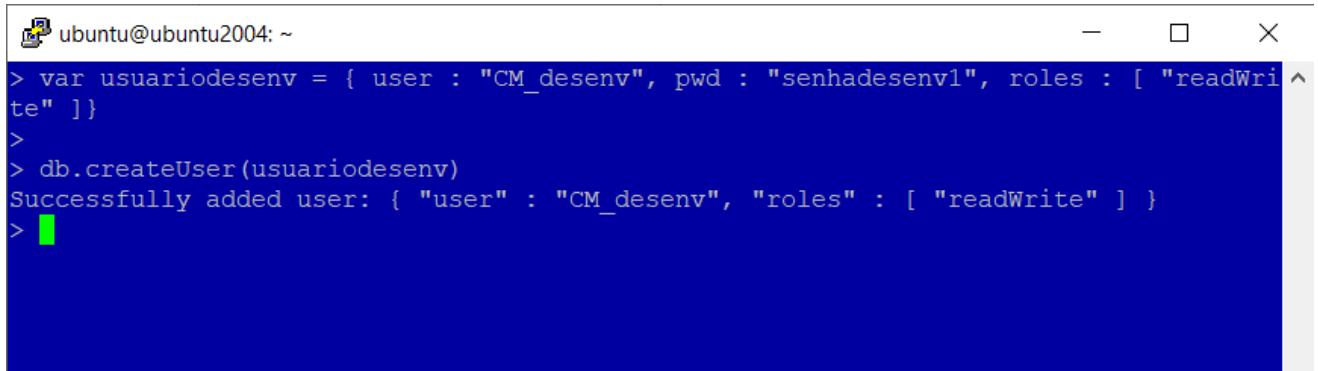
```
mongo localhost:30000/admin -u dba -p senhadba
```



```
ubuntu@ubuntu2004: ~  
> exit  
bye  
ubuntu@ubuntu2004:~$ mongo localhost:30000/admin -u dba -p senhadba  
MongoDB shell version v3.6.8  
connecting to: mongodb://localhost:30000/admin  
Implicit session: session { "id" : UUID("f03369a4-d8c2-4766-adb8-b3b3b77d0cce") }  
MongoDB server version: 3.6.8  
> [
```

D. Criar o usuário “&&_desenv” com a role readWrite no database “&&_rh”;

```
var usuariodesenv = { user : "desenv1", pwd : "senhadesenv1", roles : [ "readWrite" ]}
```



```
ubuntu@ubuntu2004: ~
> var usuariodesenv = { user : "CM_desenv", pwd : "senhadesenv1", roles : [ "readWrite" ] }
>
> db.createUser(usuariodesenv)
Successfully added user: { "user" : "CM_desenv", "roles" : [ "readWrite" ] }
>
```

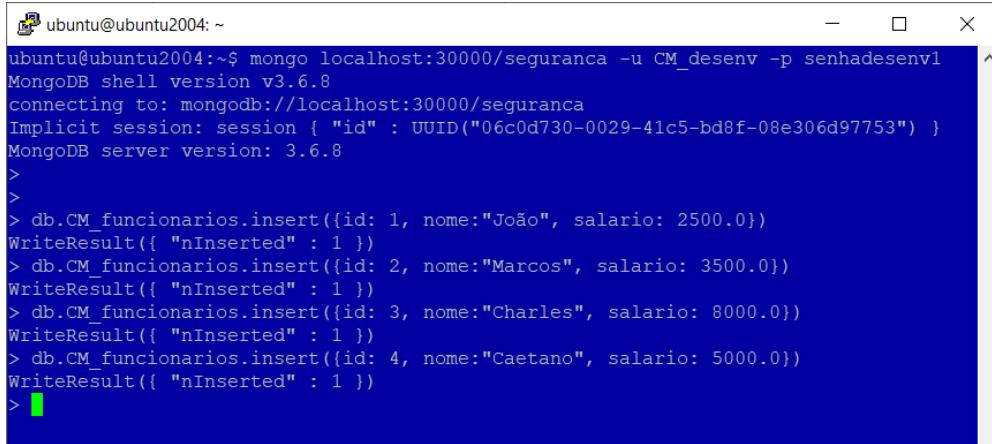
E. Conectar com o usuário “&&_desenv”

```
mongo localhost:30000/seguranca -u CM_desenv -p senhadesenv1
```



```
ubuntu@ubuntu2004: ~$ mongo localhost:30000/seguranca -u CM_desenv -p senhadesenv1
MongoDB shell version v3.6.8
connecting to: mongodb://localhost:30000/seguranca
Implicit session: session { "id" : UUID("06c0d730-0029-41c5-bd8f-08e306d97753") }
MongoDB server version: 3.6.8
>
```

F. Criar a coleção “&&_funcionarios” e inserir 4 documentos.

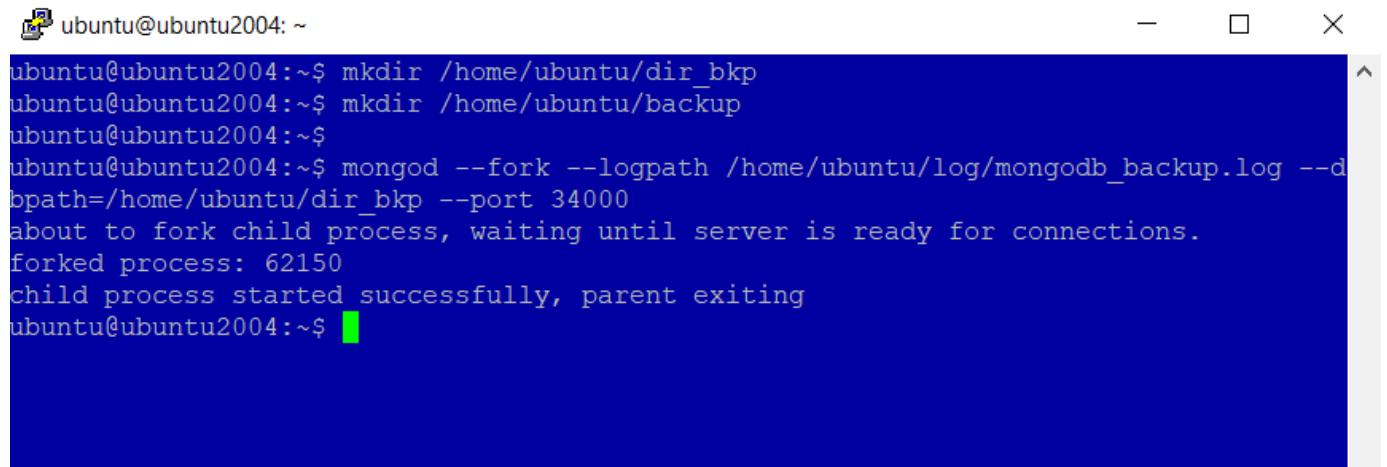


```
ubuntu@ubuntu2004: ~$ mongo localhost:30000/seguranca -u CM_desenv -p senhadesenv1
MongoDB shell version v3.6.8
connecting to: mongodb://localhost:30000/seguranca
Implicit session: session { "id" : UUID("06c0d730-0029-41c5-bd8f-08e306d97753") }
MongoDB server version: 3.6.8
>
>
> db.CM_funcionarios.insert({id: 1, nome:"João", salario: 2500.0})
WriteResult({ "nInserted" : 1 })
> db.CM_funcionarios.insert({id: 2, nome:"Marcos", salario: 3500.0})
WriteResult({ "nInserted" : 1 })
> db.CM_funcionarios.insert({id: 3, nome:"Charles", salario: 8000.0})
WriteResult({ "nInserted" : 1 })
> db.CM_funcionarios.insert({id: 4, nome:"Caetano", salario: 5000.0})
WriteResult({ "nInserted" : 1 })
>
```

Depuração, Backup/Restore

1. Criar uma instância do MongoDB.

```
mongod --fork --logpath /home/ubuntu/log/mongodb_backup.log  
--dbpath=/home/ubuntu/dir_bkp --port 34000
```

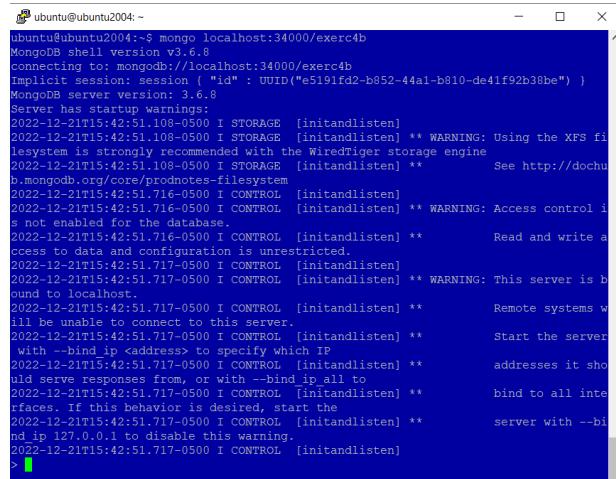


A terminal window titled "ubuntu@ubuntu2004: ~". The session shows the user creating a directory for backups and starting a MongoDB instance with the specified configuration. The output indicates the server is ready for connections and has forked a child process.

```
ubuntu@ubuntu2004:~$ mkdir /home/ubuntu/dir_bkp  
ubuntu@ubuntu2004:~$ mkdir /home/ubuntu/backup  
ubuntu@ubuntu2004:~$  
ubuntu@ubuntu2004:~$ mongod --fork --logpath /home/ubuntu/log/mongodb_backup.log --d  
bpath=/home/ubuntu/dir_bkp --port 34000  
about to fork child process, waiting until server is ready for connections.  
forked process: 62150  
child process started successfully, parent exiting  
ubuntu@ubuntu2004:~$
```

2. Conectar a essa instância (acessar o database exerc4b);

```
mongo localhost:34000/exerc4b
```

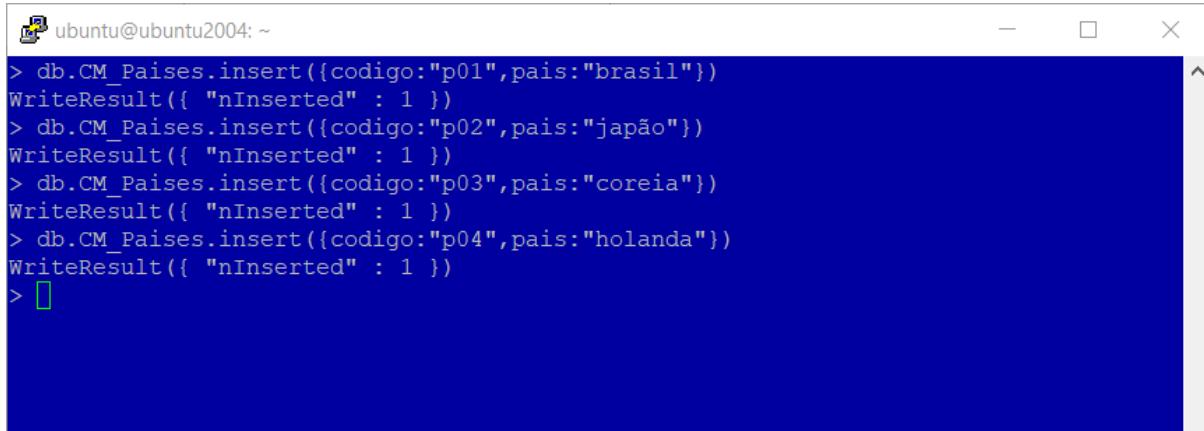


A terminal window titled "ubuntu@ubuntu2004: ~". The user connects to the MongoDB instance running on port 34000. The connection process shows various startup warnings related to storage engines and access control.

```
ubuntu@ubuntu2004:~$ mongo localhost:34000/exerc4b  
MongoDB shell version v3.6.8  
connecting to: mongodb://localhost:34000/exerc4b  
Implicit session: session { "id" : UUID("e5191fd2-b852-44a1-b810-de41f92b38be") }  
MongoDB server version: 3.6.8  
Server has startup warnings:  
2022-12-21T15:42:51.108-0500 I STORAGE [initandlisten]  
2022-12-21T15:42:51.108-0500 I STORAGE [initandlisten] ** WARNING: Using the XFS fi  
lesystem is strongly recommended with the WiredTiger storage engine  
2022-12-21T15:42:51.108-0500 I STORAGE [initandlisten] ** See http://dochu  
b.mongodb.org/core/prodnotes-filesystem  
2022-12-21T15:42:51.716-0500 I CONTROL [initandlisten]  
2022-12-21T15:42:51.716-0500 I CONTROL [initandlisten] ** WARNING: Access control i  
s not enabled for the database.  
2022-12-21T15:42:51.716-0500 I CONTROL [initandlisten] ** Read and write a  
ccess to data and configuration is unrestricted.  
2022-12-21T15:42:51.717-0500 I CONTROL [initandlisten]  
2022-12-21T15:42:51.717-0500 I CONTROL [initandlisten] ** WARNING: This server is b  
ound to localhost.  
2022-12-21T15:42:51.717-0500 I CONTROL [initandlisten] ** Remote systems w  
ill be unable to connect to this server.  
2022-12-21T15:42:51.717-0500 I CONTROL [initandlisten] ** Start the server  
with --bind_ip <address> to specify which IP  
2022-12-21T15:42:51.717-0500 I CONTROL [initandlisten] ** addresses it sho  
uld serve responses from, or with --bind_ip_all to  
2022-12-21T15:42:51.717-0500 I CONTROL [initandlisten] ** bind to all inter  
faces. If this behavior is desired, start the  
2022-12-21T15:42:51.717-0500 I CONTROL [initandlisten] ** server with --bi  
nd_ip 127.0.0.1 to disable this warning.  
2022-12-21T15:42:51.717-0500 I CONTROL [initandlisten]  
>
```

3. Criar a coleção "&&_Paises" e inserir 4 documentos.

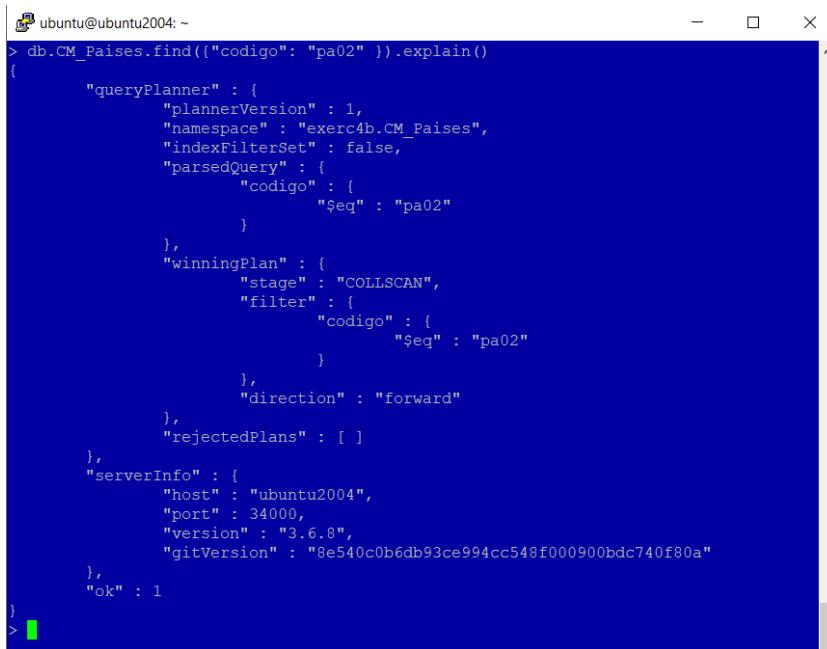
```
db.CM_Paises.insert({codigo:"p01",pais:"brasil"})
db.CM_Paises.insert({codigo:"p02",pais:"japão"})
db.CM_Paises.insert({codigo:"p03",pais:"coreia"})
db.CM_Paises.insert({codigo:"p04",pais:"holanda"})
```



```
ubuntu@ubuntu2004: ~
> db.CM_Paises.insert({codigo:"p01",pais:"brasil"})
WriteResult({ "nInserted" : 1 })
> db.CM_Paises.insert({codigo:"p02",pais:"japão"})
WriteResult({ "nInserted" : 1 })
> db.CM_Paises.insert({codigo:"p03",pais:"coreia"})
WriteResult({ "nInserted" : 1 })
> db.CM_Paises.insert({codigo:"p04",pais:"holanda"})
WriteResult({ "nInserted" : 1 })
> [ ]
```

4. Executar uma consulta para a coleção "&&_Paises" com um filtro de sua escolha com o explain.

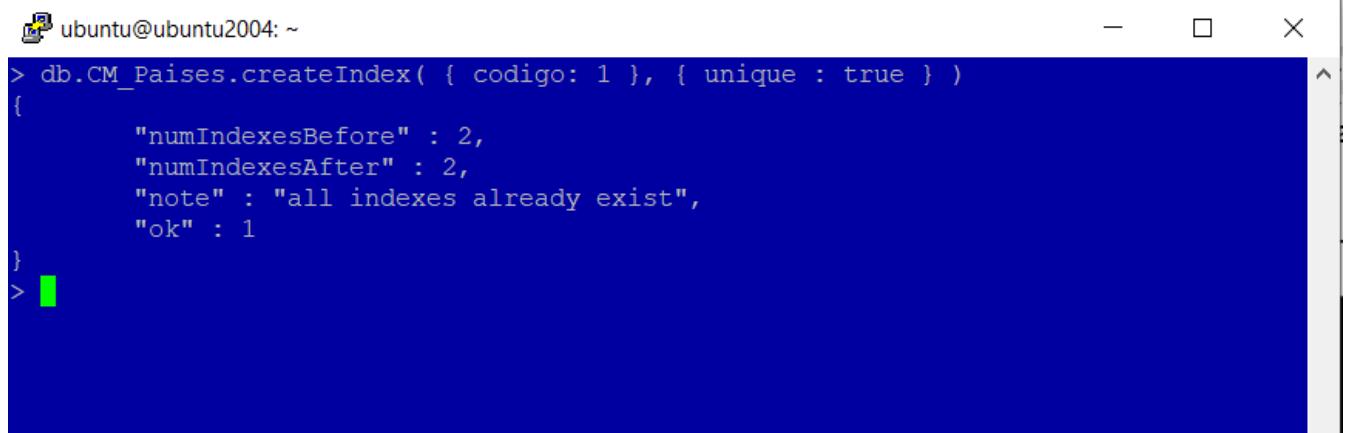
```
db.CM_Paises.find({"codigo": "pa02"}).explain()
```



```
ubuntu@ubuntu2004: ~
> db.CM_Paises.find({"codigo": "pa02"}).explain()
{
  "queryPlanner" : {
    "plannerVersion" : 1,
    "namespace" : "exerc4b.CM_Paises",
    "indexFilterSet" : false,
    "parsedQuery" : {
      "codigo" : {
        "$eq" : "pa02"
      }
    },
    "winningPlan" : {
      "stage" : "COLLSCAN",
      "filter" : {
        "codigo" : {
          "$eq" : "pa02"
        }
      },
      "direction" : "forward"
    },
    "rejectedPlans" : [ ]
  },
  "serverInfo" : {
    "host" : "ubuntu2004",
    "port" : 34000,
    "version" : "3.6.8",
    "gitVersion" : "8e540c0b6db93ce994cc548f000900bcd740f80a"
  },
  "ok" : 1
}
> [ ]
```

5. Criar um índice para um atributo que você fez o filtro no item (1.c)

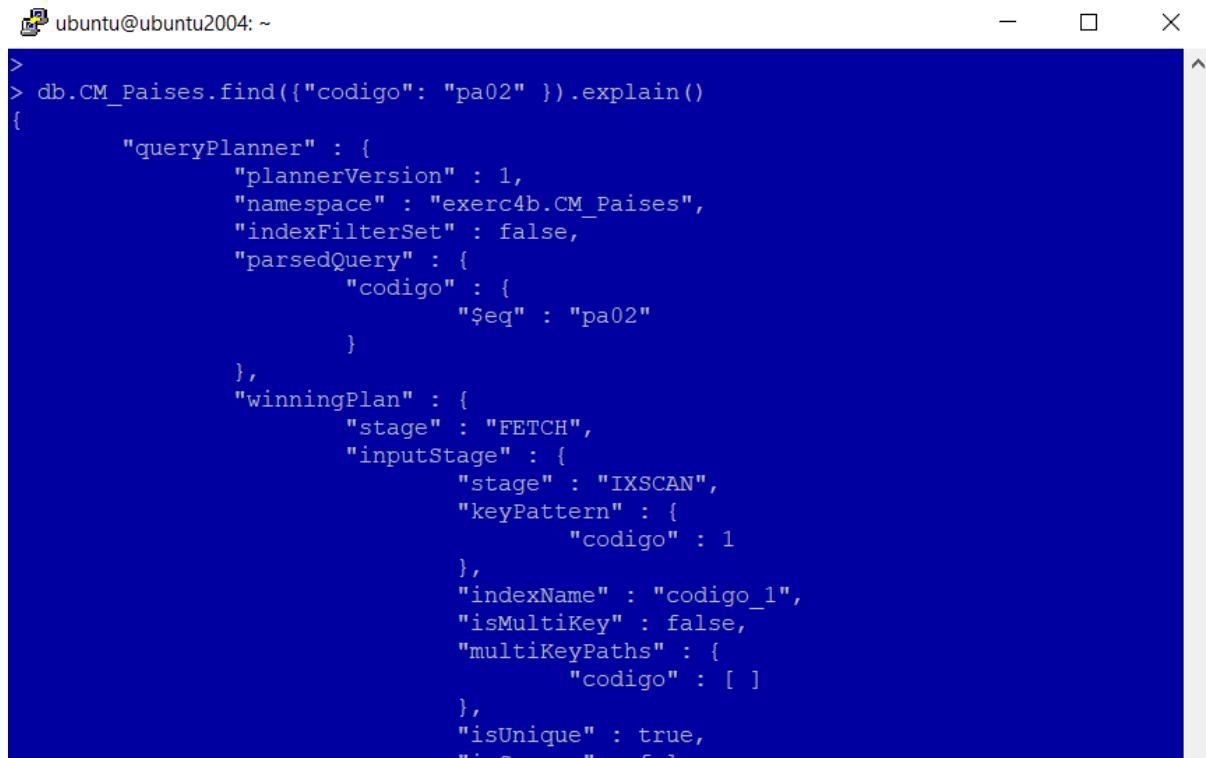
```
db.CM_Paises.createIndex( { codigo: 1 }, { unique : true } )
```



```
ubuntu@ubuntu2004: ~
> db.CM_Paises.createIndex( { codigo: 1 }, { unique : true } )
{
    "numIndexesBefore" : 2,
    "numIndexesAfter" : 2,
    "note" : "all indexes already exist",
    "ok" : 1
}
> █
```

6. Executar a mesma consulta do item (1.c).

```
db.CM_Paises.find({"codigo": "pa02" }).explain()
```

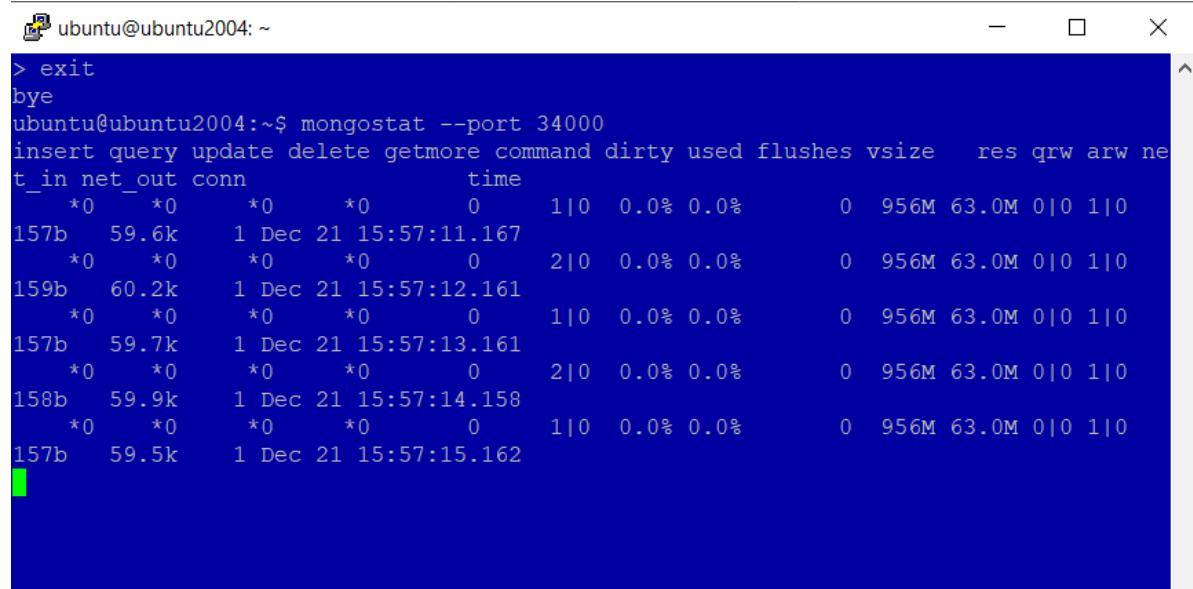


```
ubuntu@ubuntu2004: ~
>
> db.CM_Paises.find({ "codigo": "pa02" }).explain()
{
    "queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "exerc4b.CM_Paises",
        "indexFilterSet" : false,
        "parsedQuery" : {
            "codigo" : {
                "$eq" : "pa02"
            }
        },
        "winningPlan" : {
            "stage" : "FETCH",
            "inputStage" : {
                "stage" : "IXSCAN",
                "keyPattern" : {
                    "codigo" : 1
                },
                "indexName" : "codigo_1",
                "isMultiKey" : false,
                "multiKeyPaths" : {
                    "codigo" : [ ]
                },
                "isUnique" : true,
                "isSparse" : false
            }
        }
    }
}
```

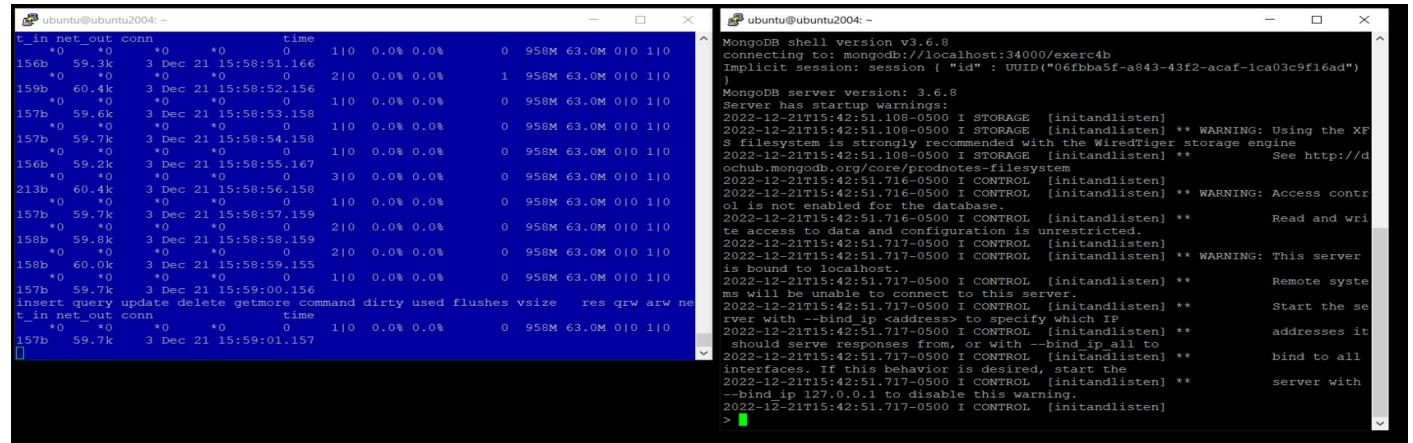
7. Criar a coleção “&&_Numero1” e inserir 50.000 documentos em paralelo; numa outra janela executar o utilitário mongostat.

```
mongostat --port 34000
```

```
for (var i = 1; i <= 50000; i++) { db.CM_Numer01.insert({ x : i }) }
```



```
> exit
bye
ubuntu@ubuntu2004:~$ mongostat --port 34000
insert query update delete getmore command dirty used flushes vsize    res qrw arw ne
t_in net_out conn          time
  *0   *0   *0   *0      0   1|0  0.0% 0.0%      0  956M 63.0M 0|0 1|0
157b  59.6k   1 Dec 21 15:57:11.167
  *0   *0   *0   *0      0   2|0  0.0% 0.0%      0  956M 63.0M 0|0 1|0
159b  60.2k   1 Dec 21 15:57:12.161
  *0   *0   *0   *0      0   1|0  0.0% 0.0%      0  956M 63.0M 0|0 1|0
157b  59.7k   1 Dec 21 15:57:13.161
  *0   *0   *0   *0      0   2|0  0.0% 0.0%      0  956M 63.0M 0|0 1|0
158b  59.9k   1 Dec 21 15:57:14.158
  *0   *0   *0   *0      0   1|0  0.0% 0.0%      0  956M 63.0M 0|0 1|0
157b  59.5k   1 Dec 21 15:57:15.162
```

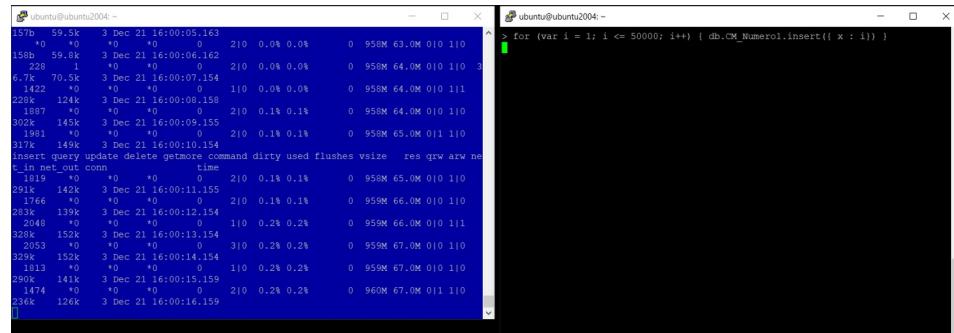


```
MongoDB shell version v3.6.8
connecting to: mongodb://localhost:34000/exerc4b
Implicit session: session { "id" : UUID("06fbba5f-a843-43f2-acaf-1ca03c9f16ad" )
}
MongoDB server version: 3.6.8
Server has startup warnings:
2022-12-21T15:42:51.108-0500 I STORAGE [initandlisten] 
2022-12-21T15:42:51.108-0500 I STORAGE [initandlisten] ** WARNING: Using the XFS file system as the storage engine is deprecated. Consider using the WiredTiger storage engine
2022-12-21T15:42:51.108-0500 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2022-12-21T15:42:51.716-0500 I CONTROL [initandlisten]
2022-12-21T15:42:51.716-0500 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2022-12-21T15:42:51.716-0500 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2022-12-21T15:42:51.717-0500 I CONTROL [initandlisten]
2022-12-21T15:42:51.717-0500 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2022-12-21T15:42:51.717-0500 I CONTROL [initandlisten] ** Remote systems will be unable to connect to this server
2022-12-21T15:42:51.717-0500 I CONTROL [initandlisten] ** Start the server with --bind_ip <address> to specify which IP
2022-12-21T15:42:51.717-0500 I CONTROL [initandlisten] ** addresses it should serve responses from, or with --bind_ip all to bind to all interfaces. If this behavior is desired, start the
2022-12-21T15:42:51.717-0500 I CONTROL [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warning
2022-12-21T15:42:51.717-0500 I CONTROL [initandlisten]
>
```

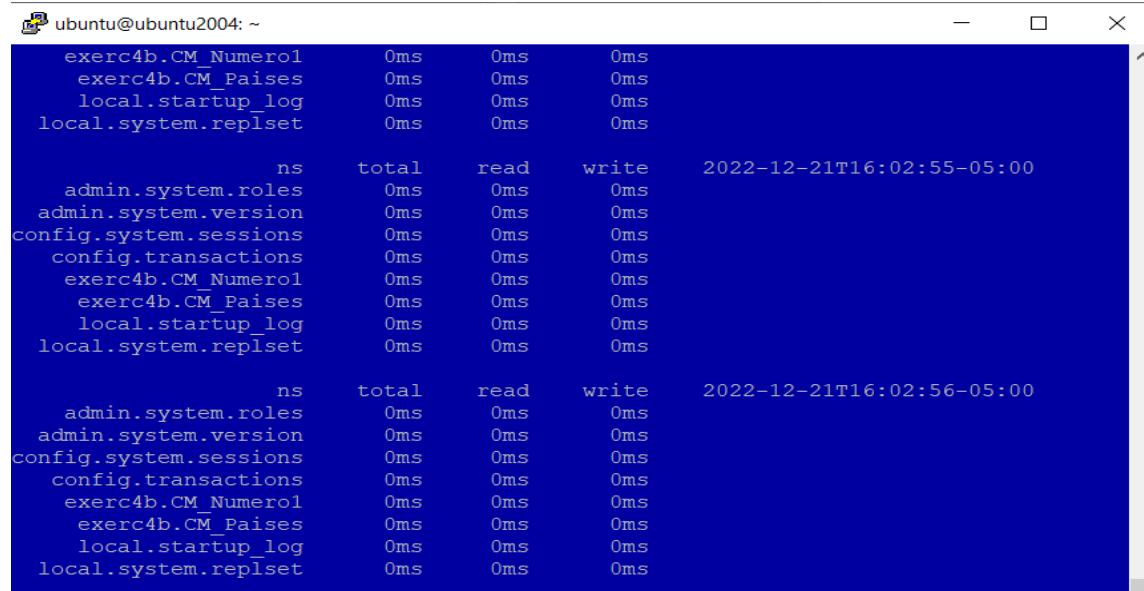
8. Criar a coleção “&&_Numero2” e inserir 50.000 documentos em paralelo; numa outra janela executar o utilitário mongotop.

mongotop --port 34000

```
for (var i = 1; i <= 50000; i++) { db.CM_Numer02.insert({ y : i }) }
```



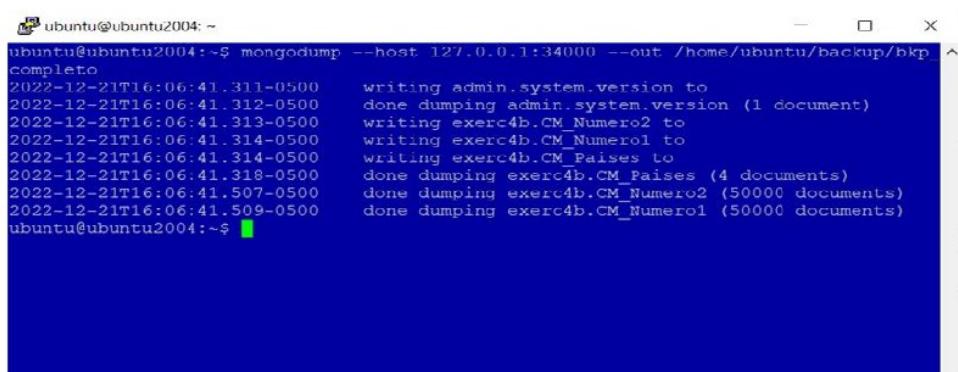
```
ubuntu@ubuntu2004: ~
for (var i = 1; i <= 50000; i++) { db.CM_Numer02.insert({ y : i }) }
```



```
ubuntu@ubuntu2004: ~
for (var i = 1; i <= 50000; i++) { db.CM_Numer02.insert({ y : i }) }
```

9. Sair do Mongo Shell e fazer um backup da instância inteira do MongoDB.

mongodump --host 127.0.0.1:34000 --out /home/ubuntu/backup/bkp_completo



```
ubuntu@ubuntu2004: ~$ mongodump --host 127.0.0.1:34000 --out /home/ubuntu/backup/bkp_completo
writing admin.system.version to
done dumping admin.system.version (1 document)
writing exerc4b.CM_Numer02 to
writing exerc4b.CM_Numer01 to
writing exerc4b.CM_Paises to
done dumping exerc4b.CM_Paises (4 documents)
done dumping exerc4b.CM_Numer02 (50000 documents)
done dumping exerc4b.CM_Numer01 (50000 documents)
```

10. Matar o processo do MongoDB.

```
kill -9 62150
```

```
ubuntu    62150      1  2 15:42 ?        00:00:41 mongod --fork --logpath /home/ubuntu/log/mongodb_backup.log --dbpath=/home/ubuntu/dir_bkp --port 34000
ubuntu@ubuntu2004:~$ kill -9 62150
ubuntu@ubuntu2004:~$
```

11. Apagar o diretório de dados do MongoDB.,

```
rm -r /home/ubuntu/dir_bkp
```

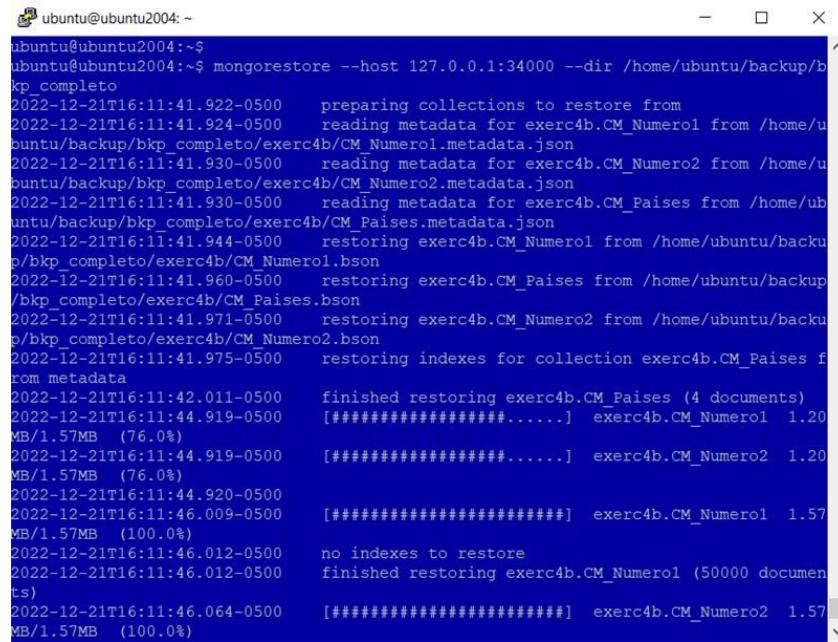
```
ubuntu@ubuntu2004: ~
ubuntu@ubuntu2004:~$ rm -r /home/ubuntu/dir_bkp
ubuntu@ubuntu2004:~$
```

12. Fazer o restore do MongoDB (o caminho tem que ser criado anteriormente).

```
mkdir /home/ubuntu/dir_bkp mongod --fork --logpath /home/ubuntu/log/mongodb_backup.log --dbpath=/home/ubuntu/dir_bkp --port 34000
```

```
ubuntu@ubuntu2004: ~
ubuntu@ubuntu2004:~$ mkdir /home/ubuntu/dir_bkp
ubuntu@ubuntu2004:~$ mongod --fork --logpath /home/ubuntu/log/mongodb_backup.log --dbpath=/home/ubuntu/dir_bkp --port 34000
about to fork child process, waiting until server is ready for connections.
forked process: 62326
child process started successfully, parent exiting
ubuntu@ubuntu2004:~$
```

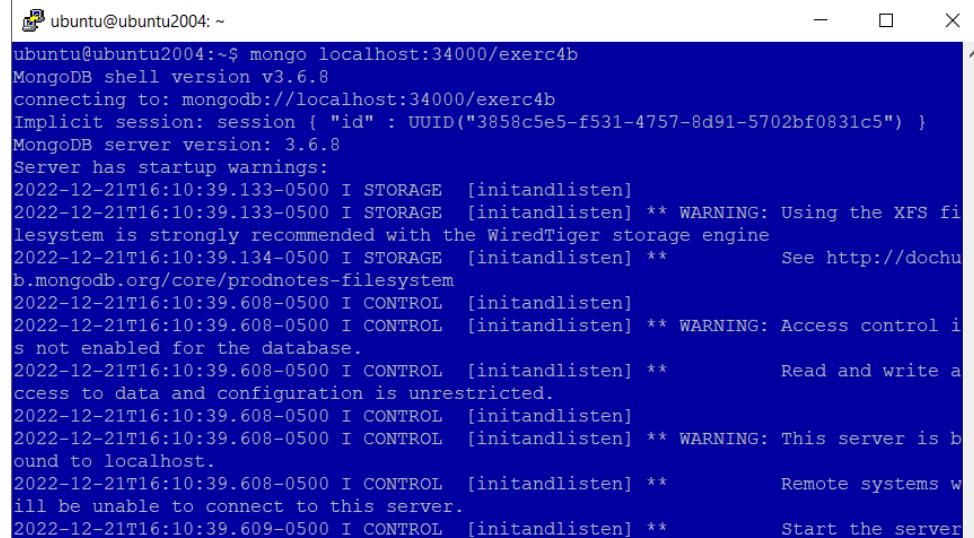
```
mongorestore --host 127.0.0.1:34000 --dir /home/ubuntu/backup/bkp_completo
```



```
ubuntu@ubuntu2004:~$ mongorestore --host 127.0.0.1:34000 --dir /home/ubuntu/backup/bkp_completo
2022-12-21T16:11:41.922-0500      preparing collections to restore from
2022-12-21T16:11:41.924-0500      reading metadata for exerc4b.CM_Numerol from /home/ubuntu/backup/bkp_completo/exerc4b/CM_Numerol.metadata.json
2022-12-21T16:11:41.930-0500      reading metadata for exerc4b.CM_Numerol from /home/ubuntu/backup/bkp_completo/exerc4b/CM_Numerol.metadata.json
2022-12-21T16:11:41.930-0500      reading metadata for exerc4b.CM_Numerol from /home/ubuntu/backup/bkp_completo/exerc4b/CM_Numerol.metadata.json
2022-12-21T16:11:41.930-0500      reading metadata for exerc4b.CM_Paises from /home/ubuntu/backup/bkp_completo/exerc4b/CM_Paises.metadata.json
2022-12-21T16:11:41.944-0500      restoring exerc4b.CM_Numerol from /home/ubuntu/backup/bkp_completo/exerc4b/CM_Numerol.bson
2022-12-21T16:11:41.960-0500      restoring exerc4b.CM_Paises from /home/ubuntu/backup/bkp_completo/exerc4b/CM_Paises.bson
2022-12-21T16:11:41.971-0500      restoring exerc4b.CM_Numerol from /home/ubuntu/backup/bkp_completo/exerc4b/CM_Numerol.bson
2022-12-21T16:11:41.975-0500      restoring indexes for collection exerc4b.CM_Paises from metadata
2022-12-21T16:11:42.011-0500      finished restoring exerc4b.CM_Paises (4 documents)
2022-12-21T16:11:44.919-0500      [#####] exerc4b.CM_Numerol 1.20 MB/1.57MB (76.0%)
2022-12-21T16:11:44.919-0500      [#####] exerc4b.CM_Numerol 1.20 MB/1.57MB (76.0%)
2022-12-21T16:11:44.920-0500      [#####] exerc4b.CM_Numerol 1.57 MB/1.57MB (100.0%)
2022-12-21T16:11:46.009-0500      no indexes to restore
2022-12-21T16:11:46.012-0500      finished restoring exerc4b.CM_Numerol (50000 documents)
2022-12-21T16:11:46.064-0500      [#####] exerc4b.CM_Numerol 1.57 MB/1.57MB (100.0%)
```

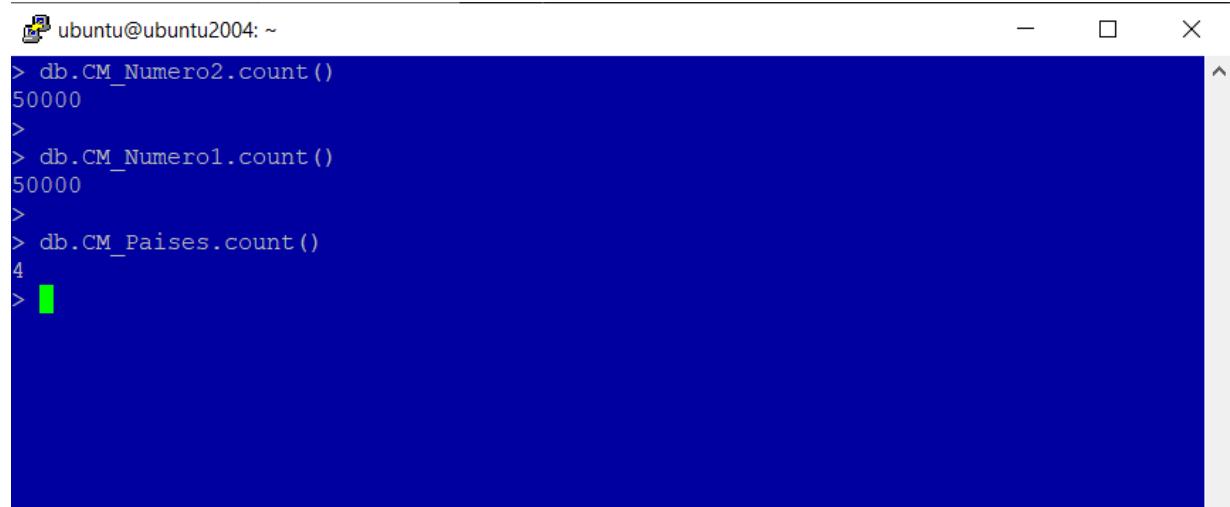
13. Acessar a instância e ir para o database exerc4b. Verificar o número de documentos por coleção.

```
mongo localhost:34000/exerc4b
```



```
ubuntu@ubuntu2004:~$ mongo localhost:34000/exerc4b
MongoDB shell version v3.6.8
connecting to: mongodb://localhost:34000/exerc4b
Implicit session: session { "id" : UUID("3858c5e5-f531-4757-8d91-5702bf0831c5") }
MongoDB server version: 3.6.8
Server has startup warnings:
2022-12-21T16:10:39.133-0500 I STORAGE  [initandlisten]
2022-12-21T16:10:39.133-0500 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2022-12-21T16:10:39.134-0500 I STORAGE  [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2022-12-21T16:10:39.608-0500 I CONTROL  [initandlisten]
2022-12-21T16:10:39.608-0500 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2022-12-21T16:10:39.608-0500 I CONTROL  [initandlisten] ** Read and write access to data and configuration is unrestricted.
2022-12-21T16:10:39.608-0500 I CONTROL  [initandlisten]
2022-12-21T16:10:39.608-0500 I CONTROL  [initandlisten] ** WARNING: This server is bound to localhost.
2022-12-21T16:10:39.608-0500 I CONTROL  [initandlisten] ** Remote systems will be unable to connect to this server.
2022-12-21T16:10:39.609-0500 I CONTROL  [initandlisten] ** Start the server
```

```
db.CM_Numero2.count()  
db.CM_Numero1.count()  
db.CM_Paises.count()
```



A screenshot of a terminal window titled "ubuntu@ubuntu2004: ~". The window contains the following text:

```
> db.CM_Numero2.count()
50000
>
> db.CM_Numerol.count()
50000
>
> db.CM_Paises.count()
4
> █
```