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.

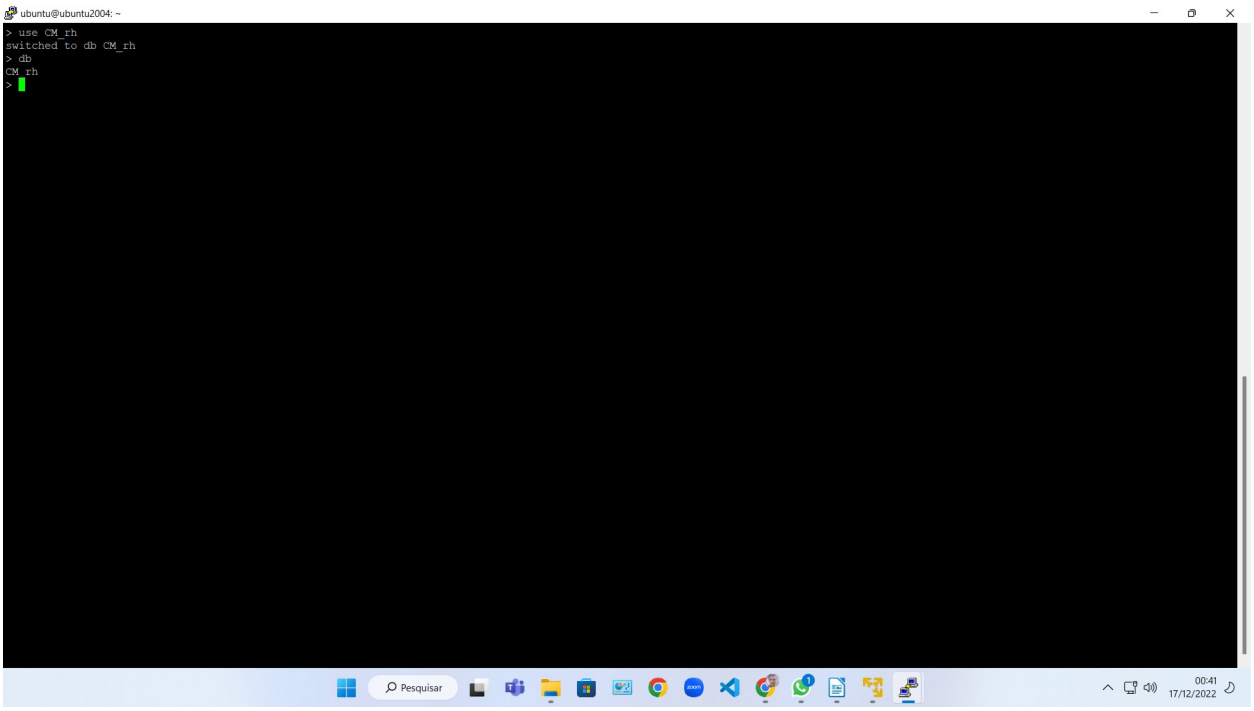
Redis: quando o assunto é banco de dados valores-chave, o Redis é o nome mais forte no quesito. Sua principal característica é a velocidade, muito útil para tratar um grande volume de dados em aplicações com muitas transações.

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

Parte 2 – Prática

1. Criar o database "&&_rh".

use CM_rh

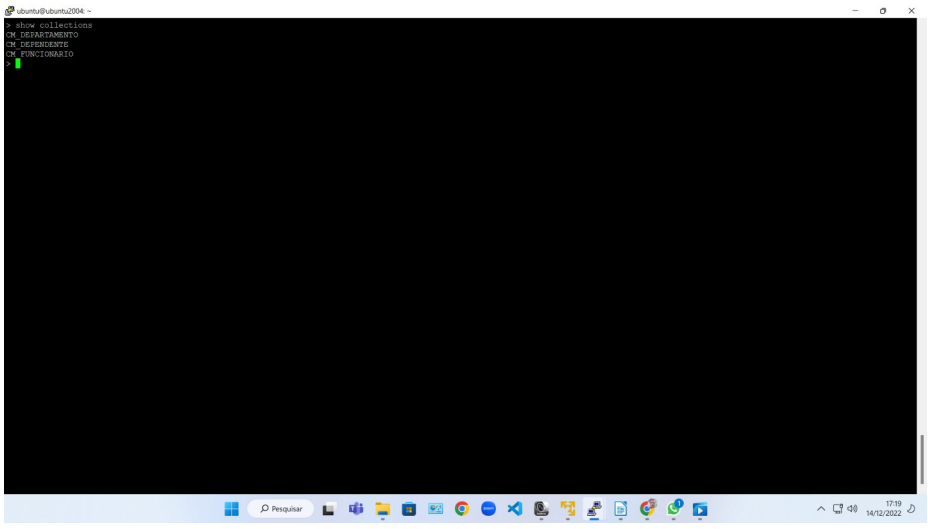


2. Criar as coleções: "&&_DEPARTAMENTO", "&&_FUNCIONARIO", "&&_DEPENDENTE".

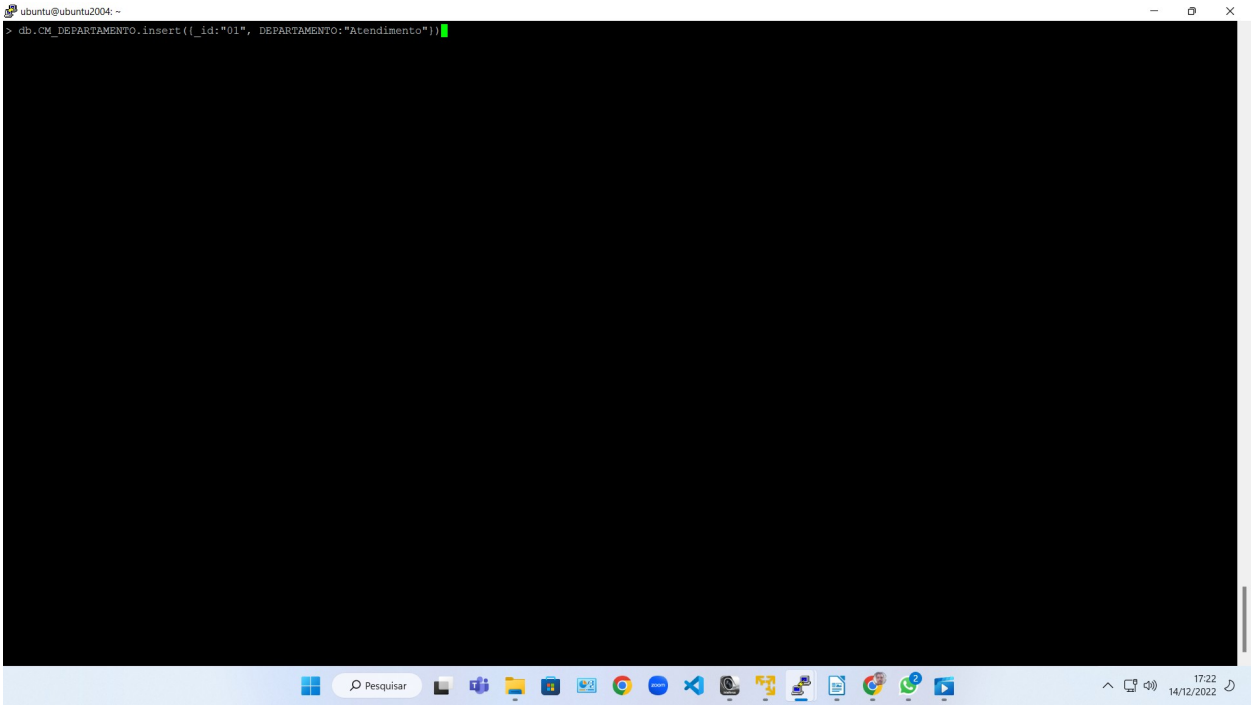
db.CM_FUNCIONARIO.insert({_id:"01" nome:"Carolina", salario:"1000"})

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

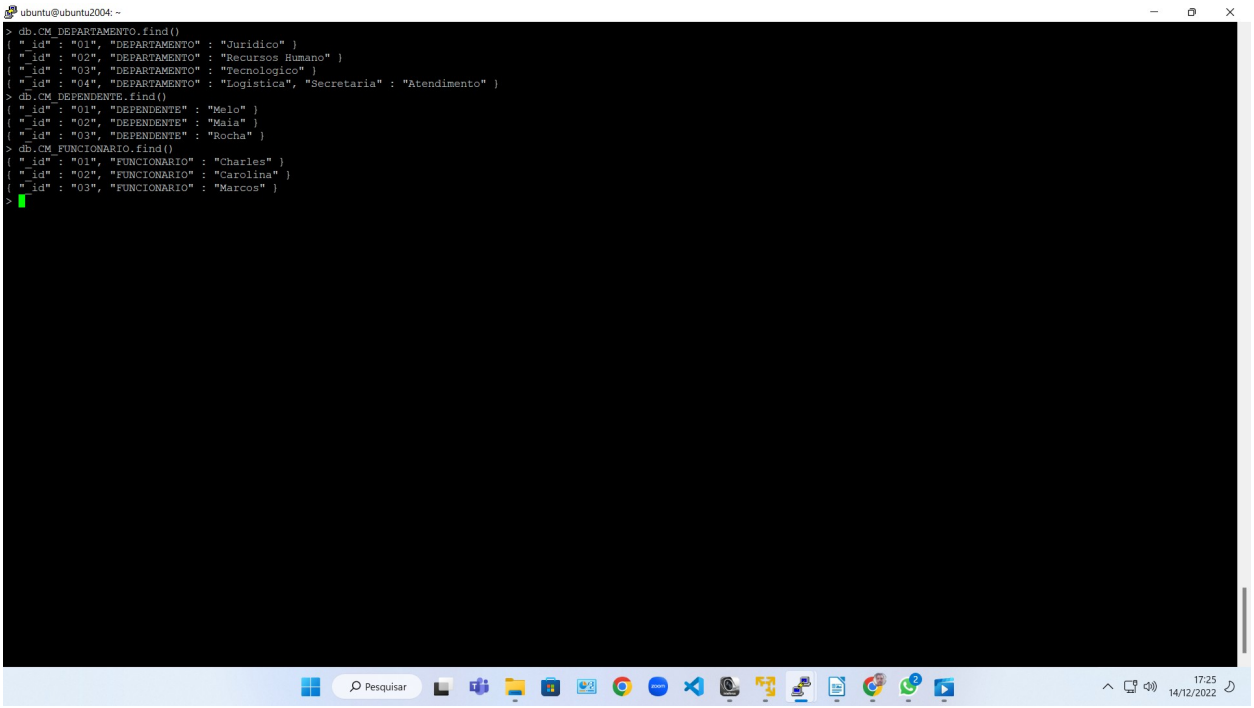
db.CM_DEPENDENTE.insert({_id:"01" dependente:"Melo"})



3. Inserir 3 documentos para cada coleção.

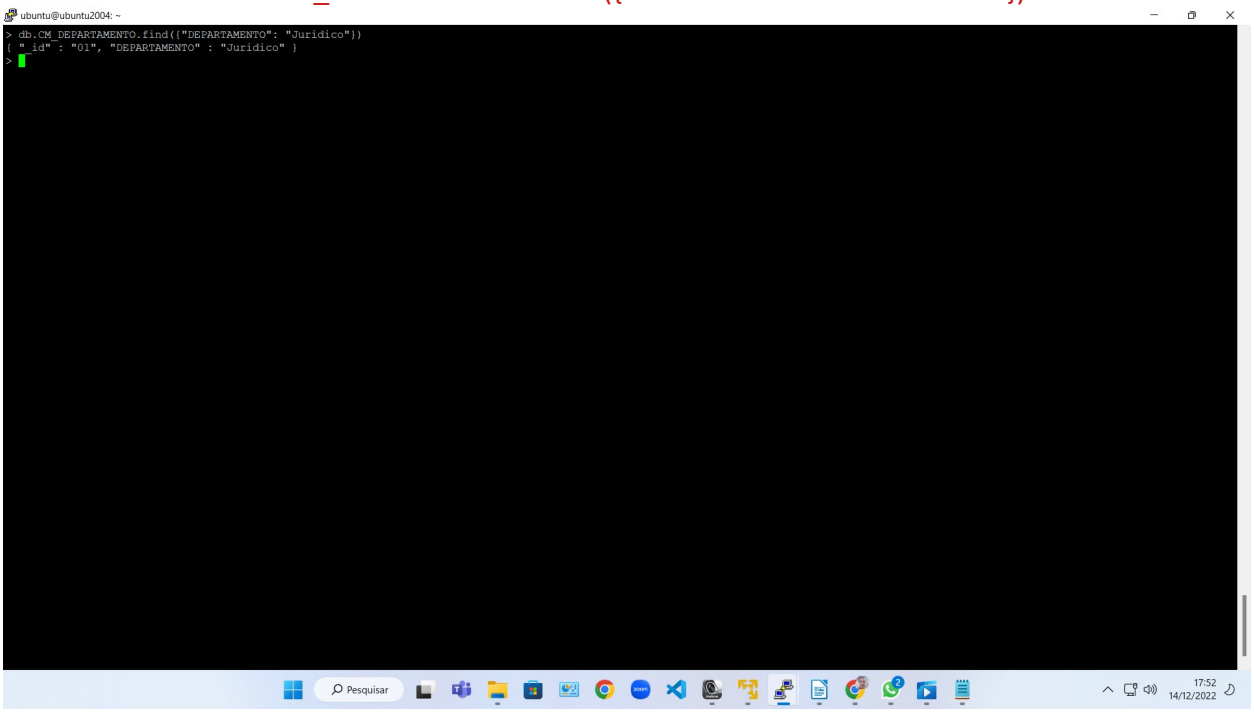


4. Mostrar o conteúdo de cada coleção.



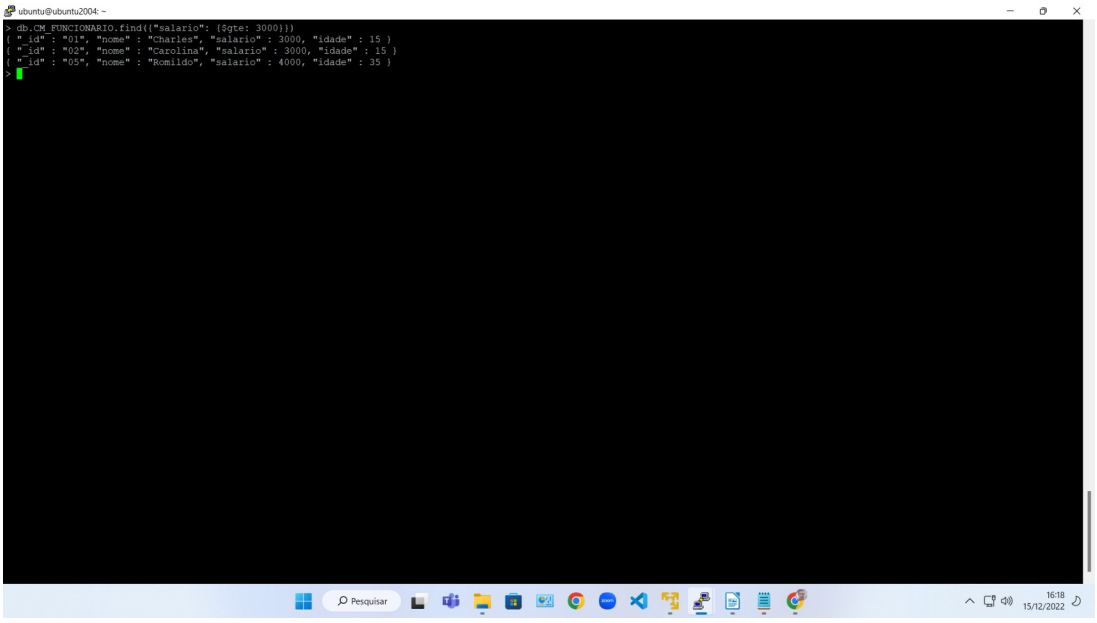
5. Para a coleção "&&_DEPARTAMENTO" fazer um filtro baseado numa descrição do departamento (igualdade).

db.CM_DEPARTAMENTO.find({“DEPARTAMENTO”: “Juridico”})



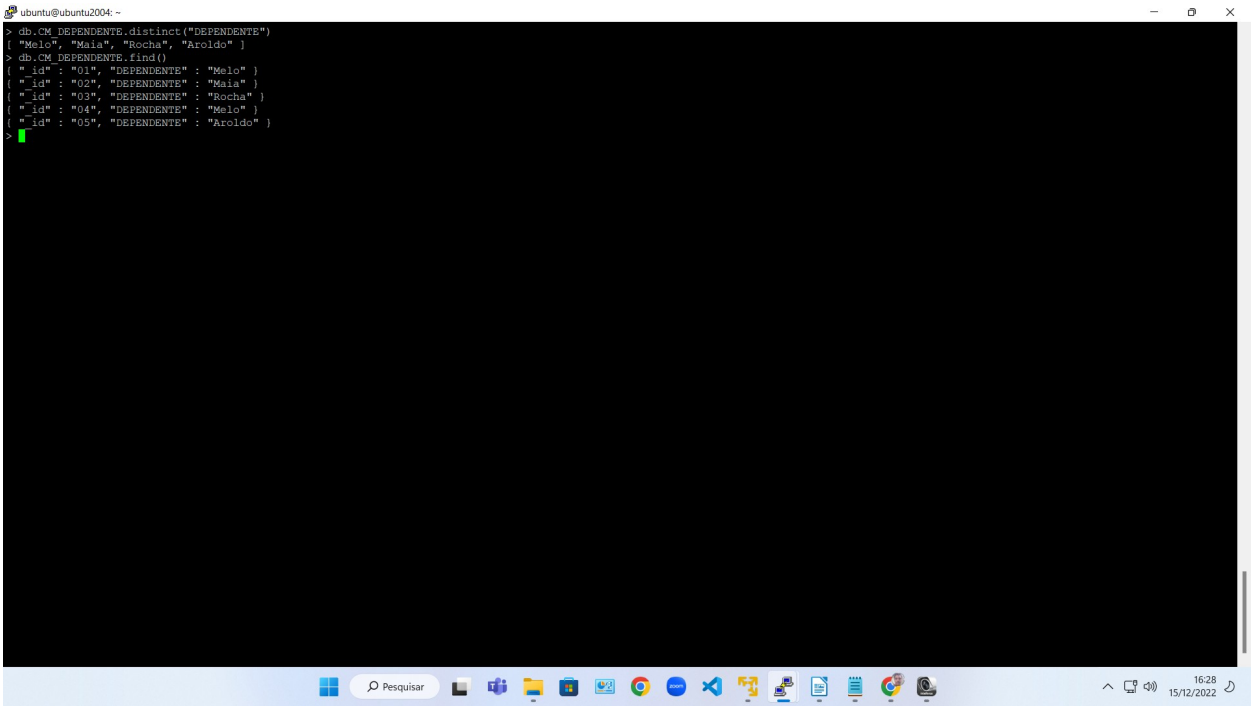
6. Para a coleção "&&_FUNCIONARIO" mostrar os funcionários que recebem salário acima de R $2000,00.

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



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

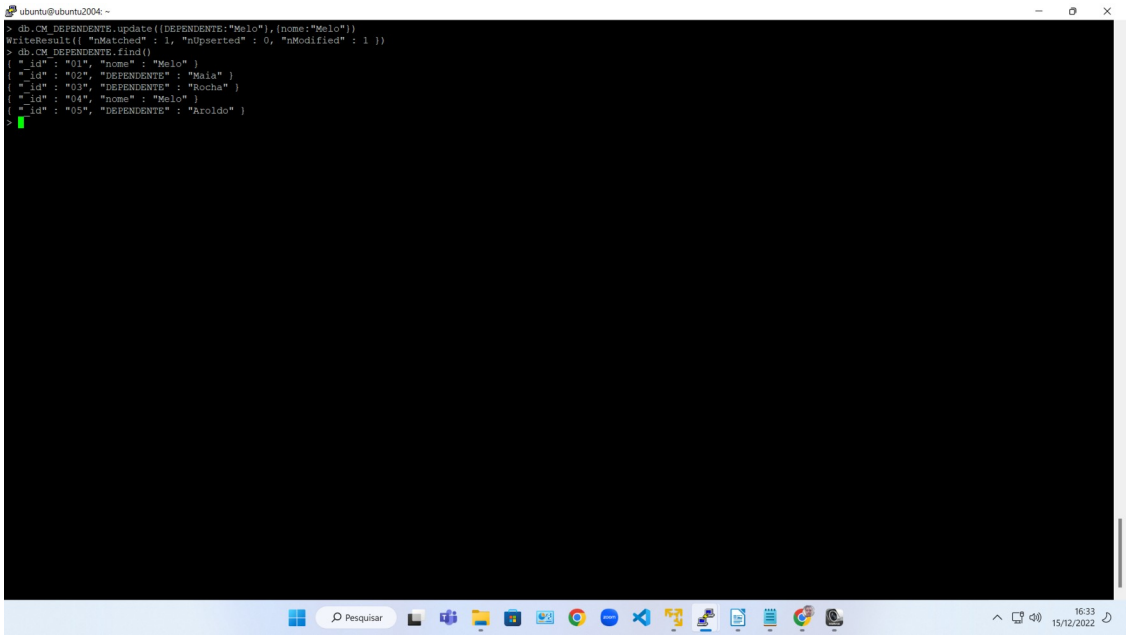
db.CM_DEPENDENTE.distinct("DEPENDENTE")



OBS: No caso coloquei o nome de DEPENDENTE

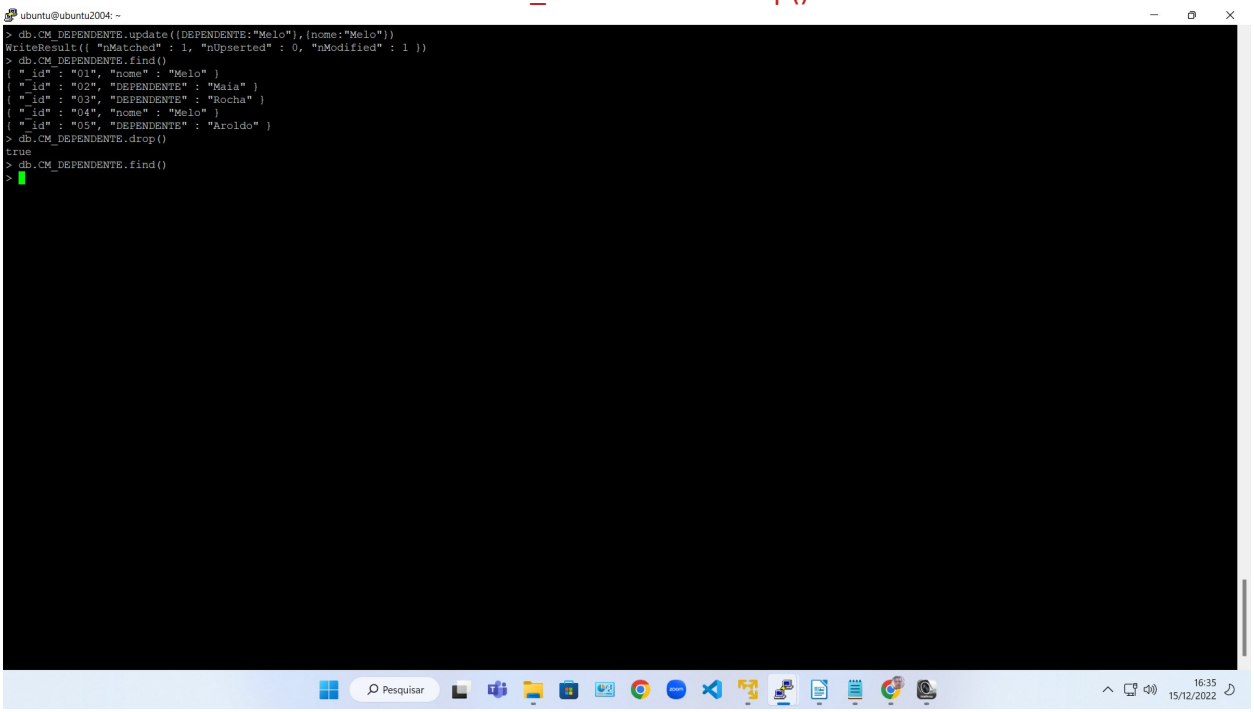
8. Executar um update para uma das coleções.

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



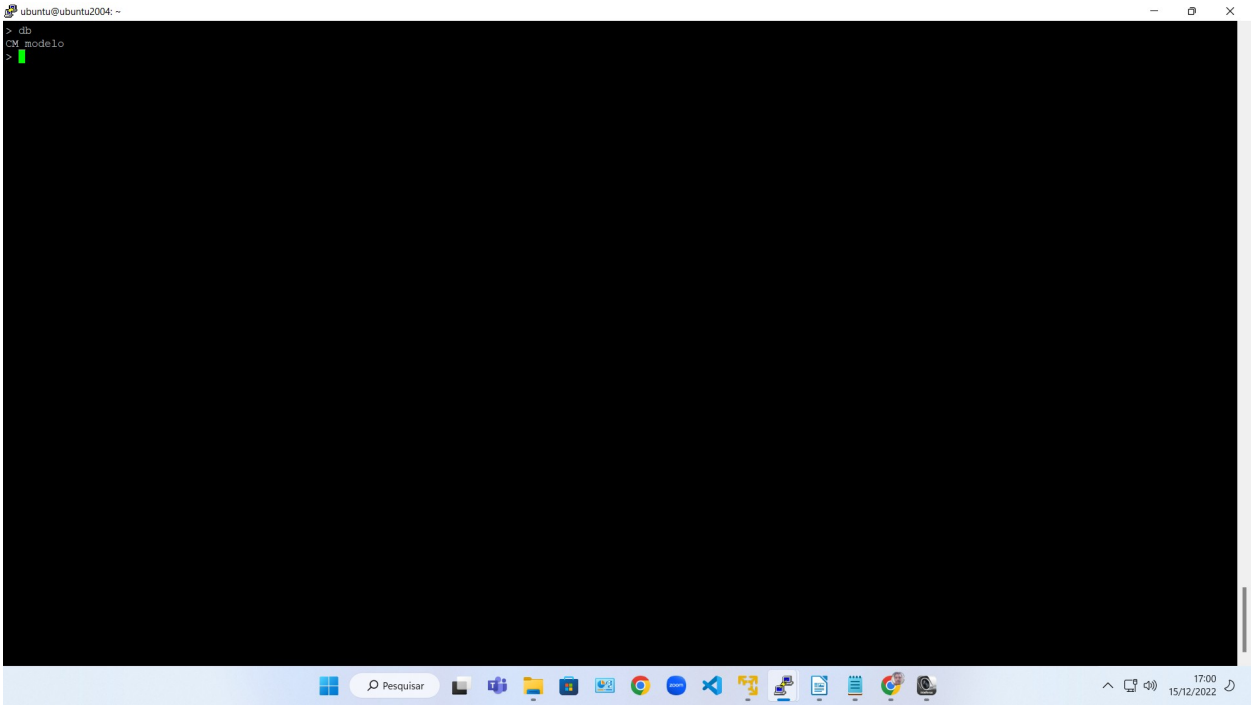
9. Executar um delete (remove) para uma das coleções.

db.CM_DEPENDENTE.drop()



Modelagem

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

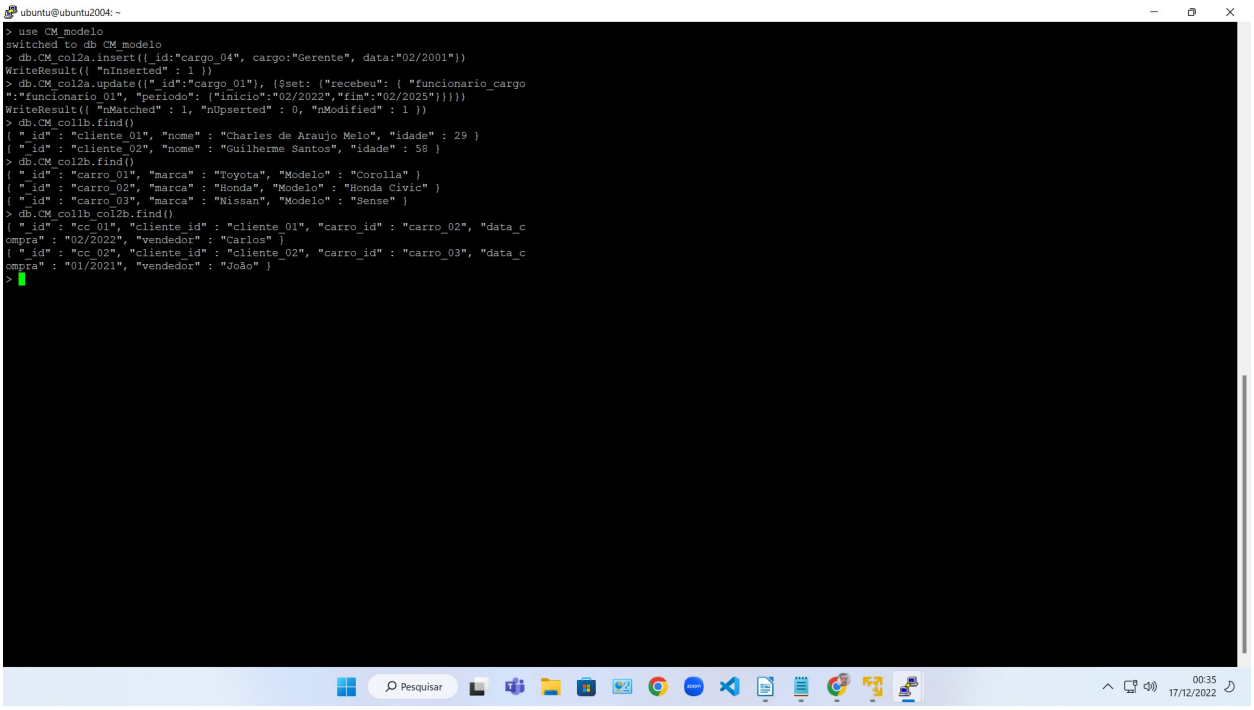


2. Criar um pequeno modelo que envolva duas coleções com as cardinalidades 1-N. O nome das coleções deve ser "&&_col1a" e "&&_col2a"

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"}}}})
```



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.

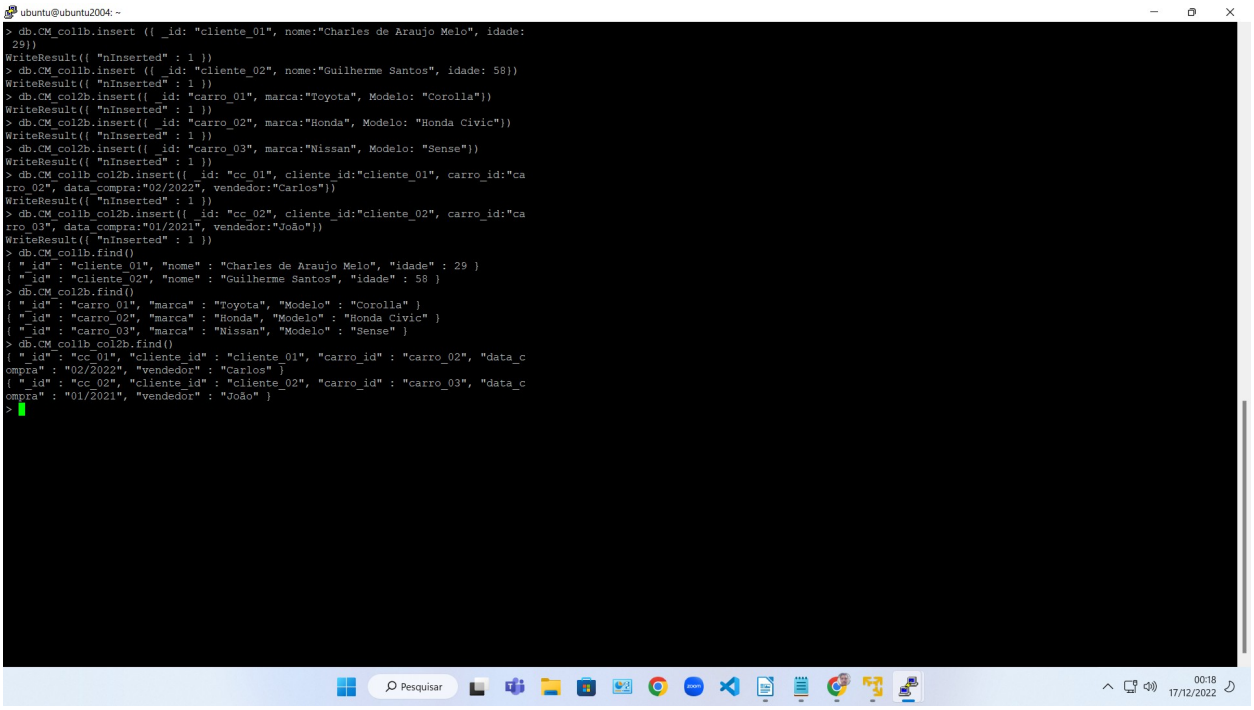
A. Criar duas coleções de nome (se você embarcar) ou;

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.

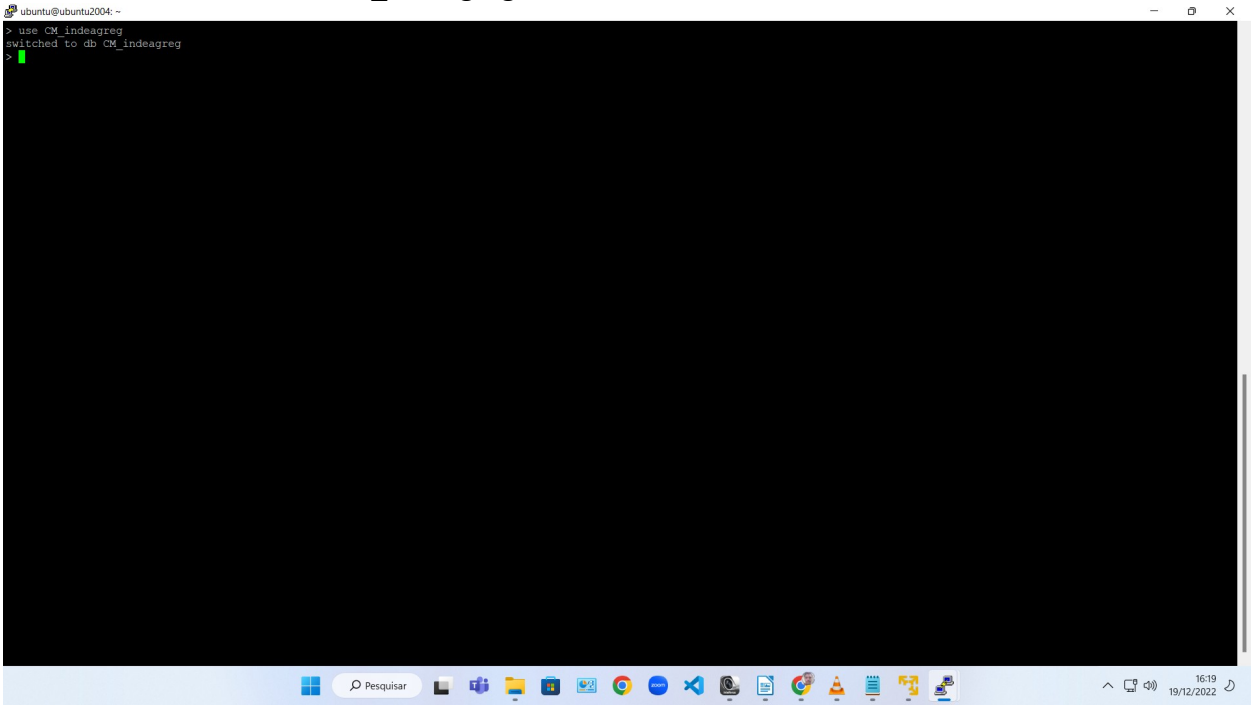
db.CM_col1b.insert ({ _id: "cliente_01", nome:"Charles de Araujo Melo", idade: 29})
db.CM_col1b.insert ({ _id: "cliente_02", nome:"Guilherme Santos", idade: 58})
db.CM_col2b.insert({ _id: "carro_01", marca:"Toyota", Modelo: "Corolla"})
db.CM_col2b.insert({ _id: "carro_02", marca:"Honda", Modelo: "Honda 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"})



Índices

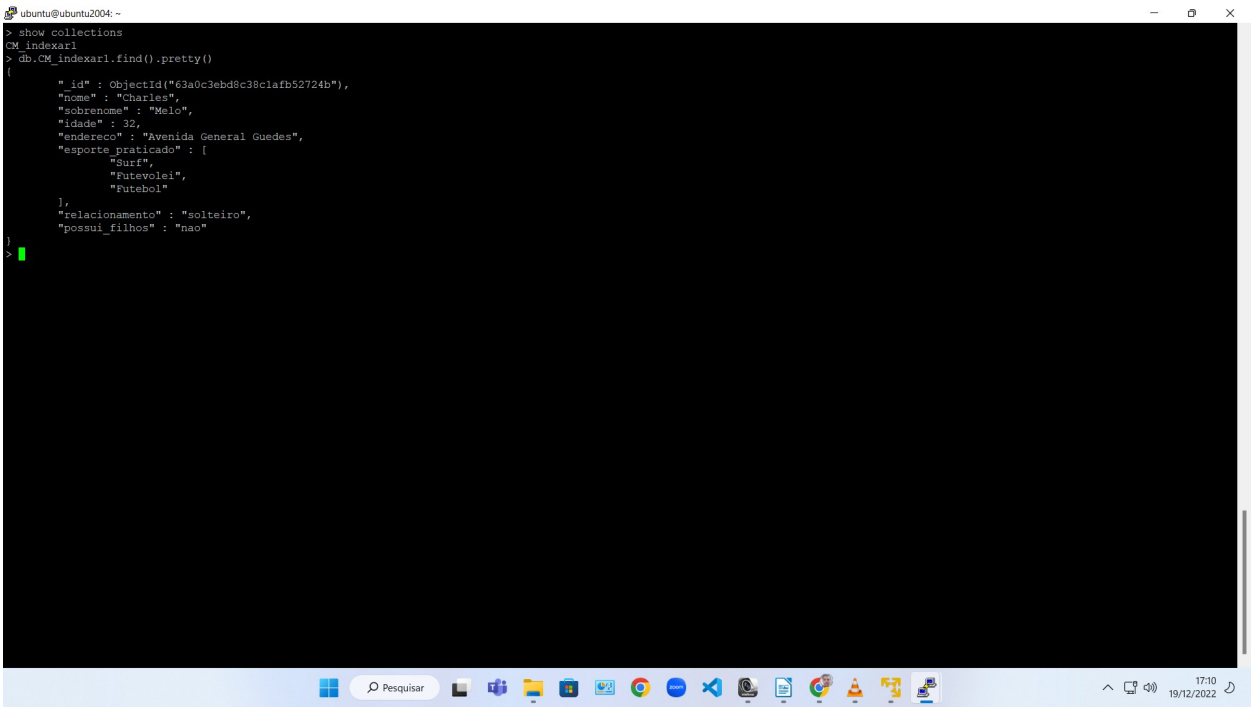
1. Criar o database "&&_indeagreg".

Use CM_indeagreg



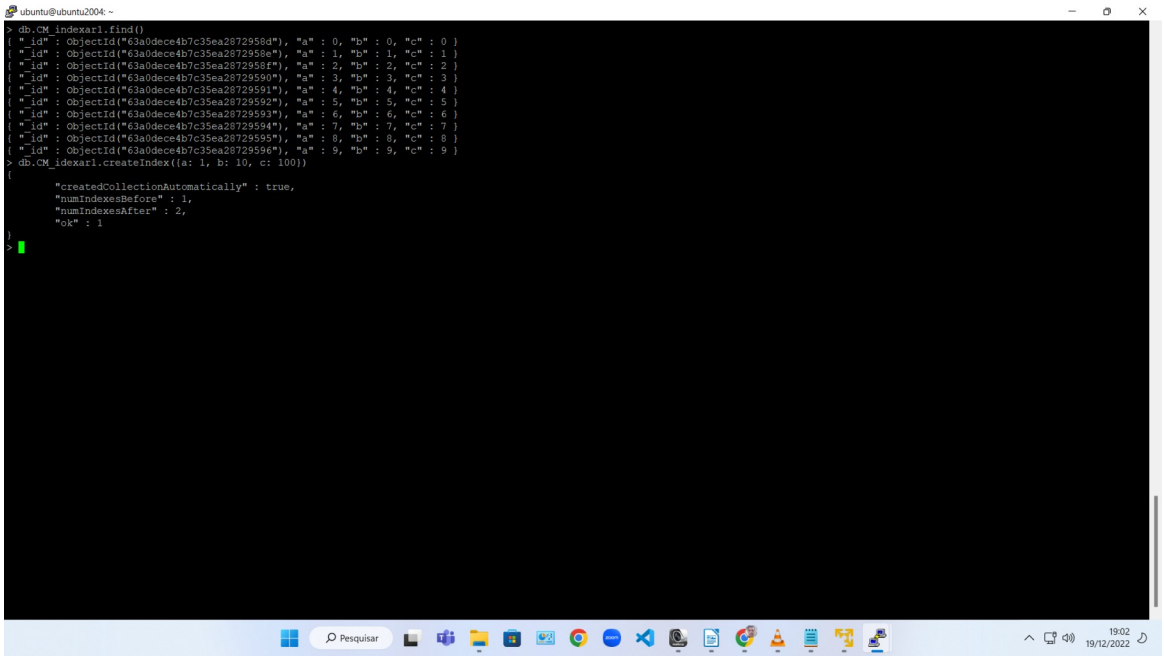
2. Criar uma coleção de nome "&&_indexar1" que apresente pelo menos 7 atributos (tem que ter pelo menos um atributo array).

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



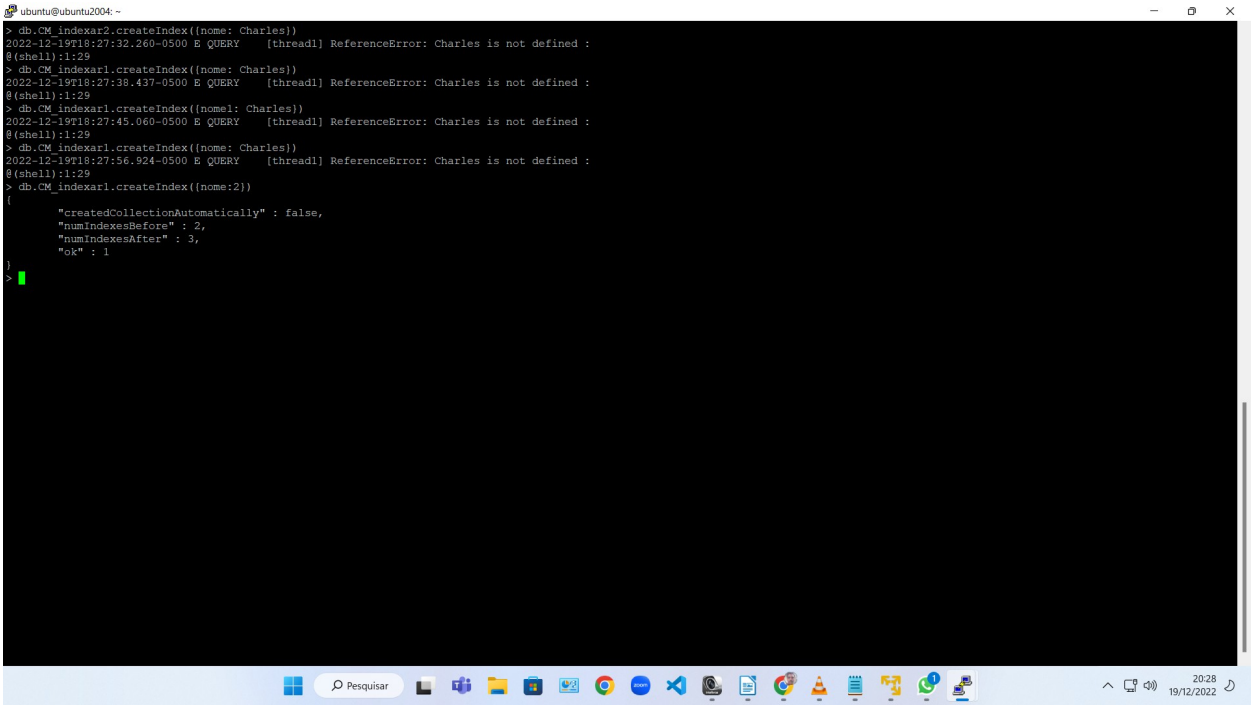
A. Criar um índice (não único e com ordenação ascendente).

```
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})
```



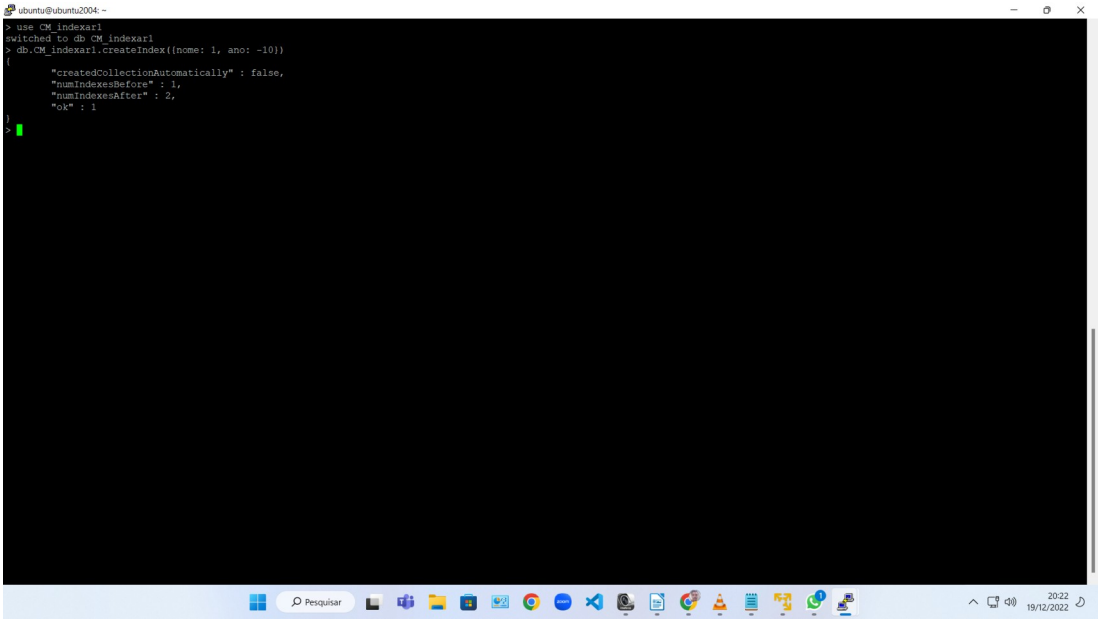
B. Criar um índice (único e com ordenação descendente).

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



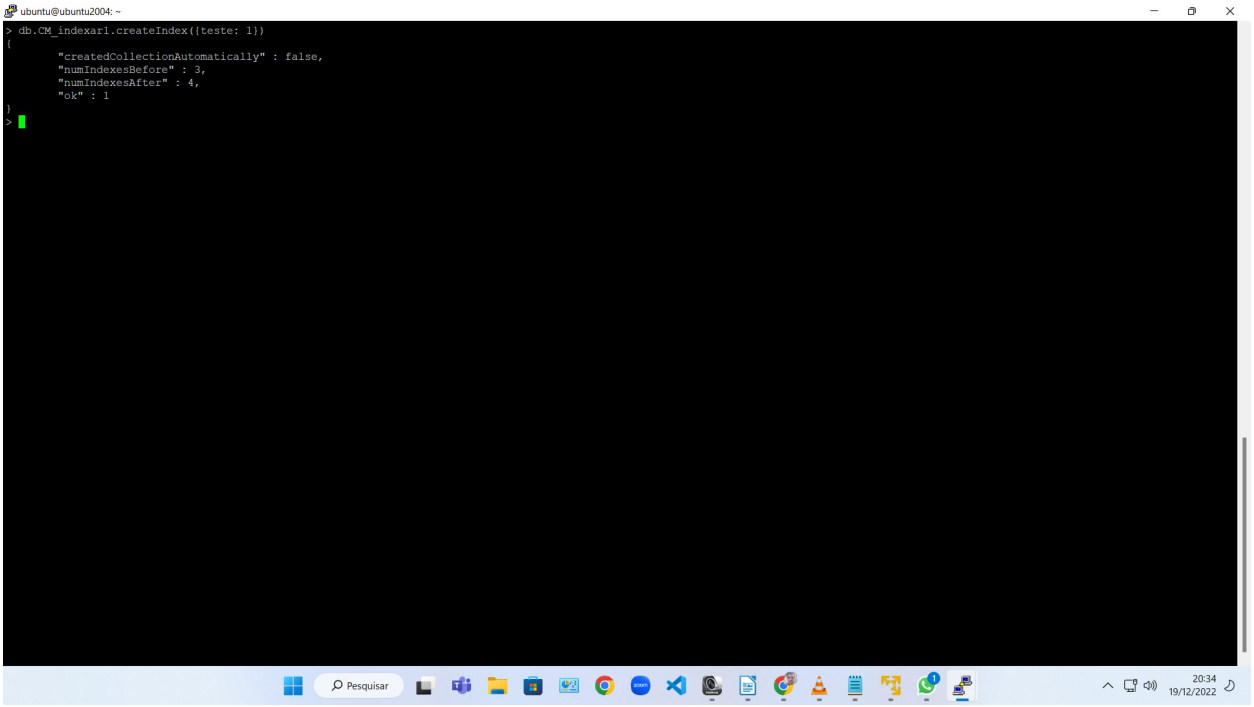
C. Criar um índice (com dois atributos).

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



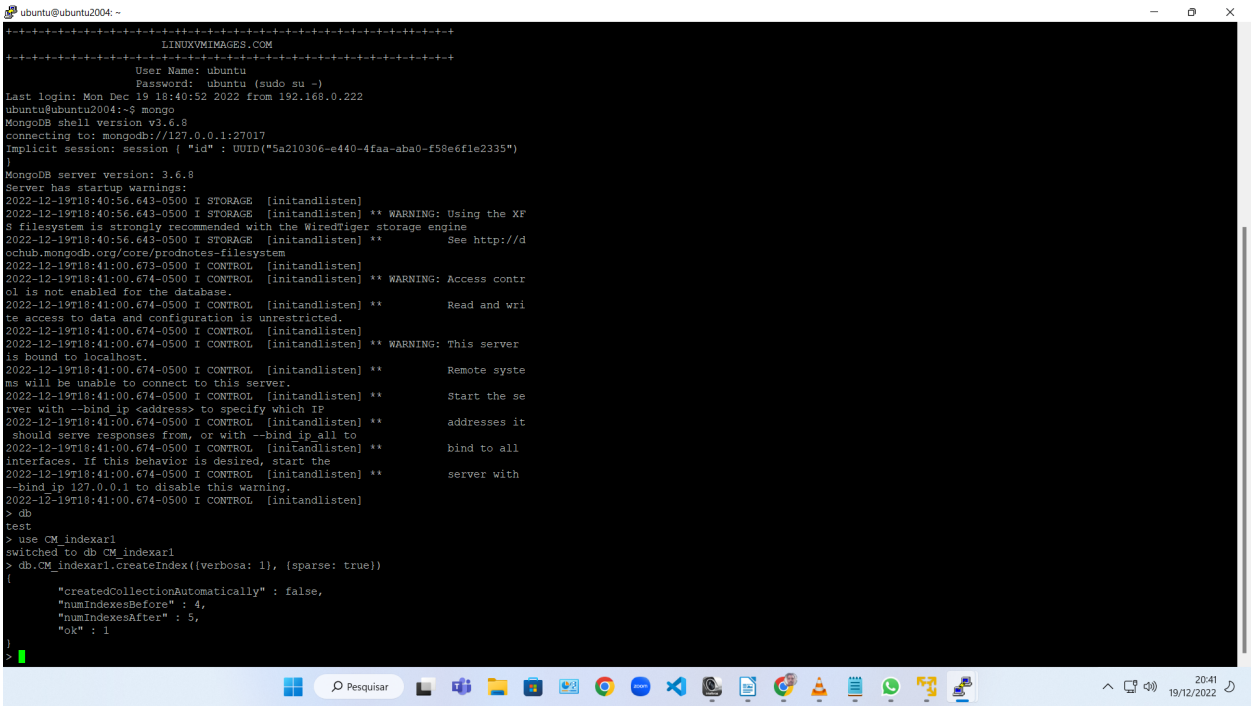
D. Criar um índice para um atributo array.

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



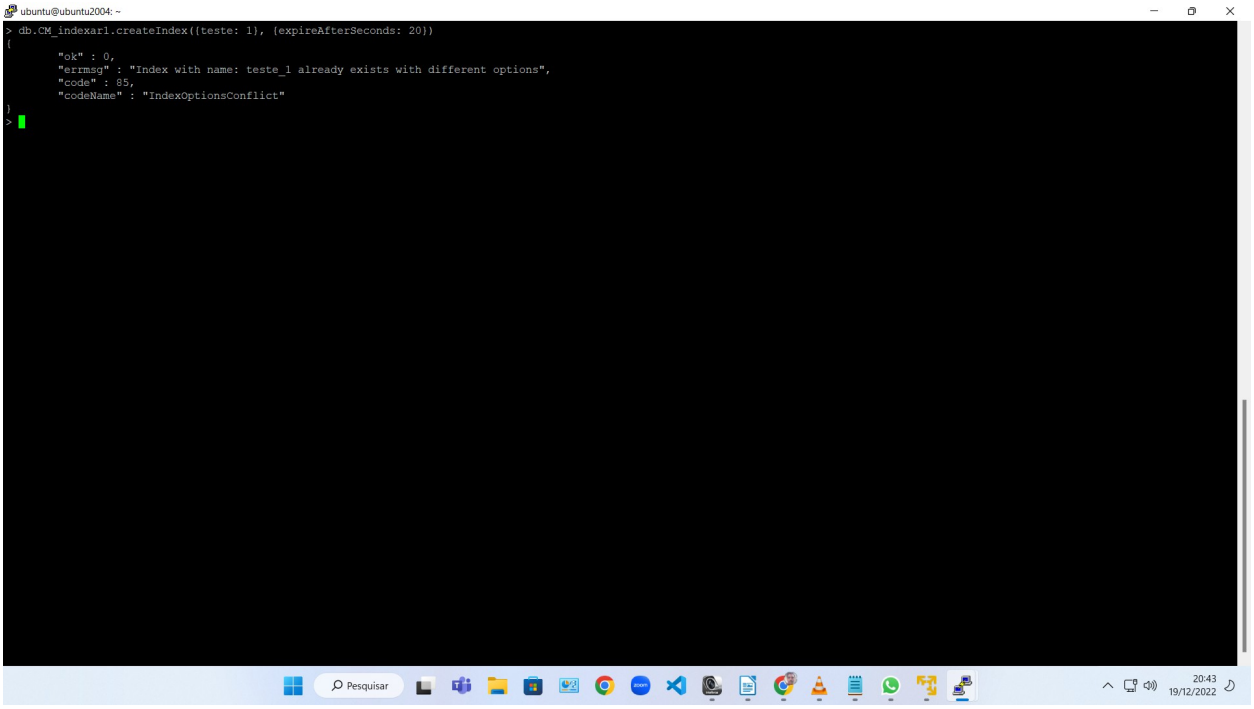
E. Criar um índice esparso (com um atributo).

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



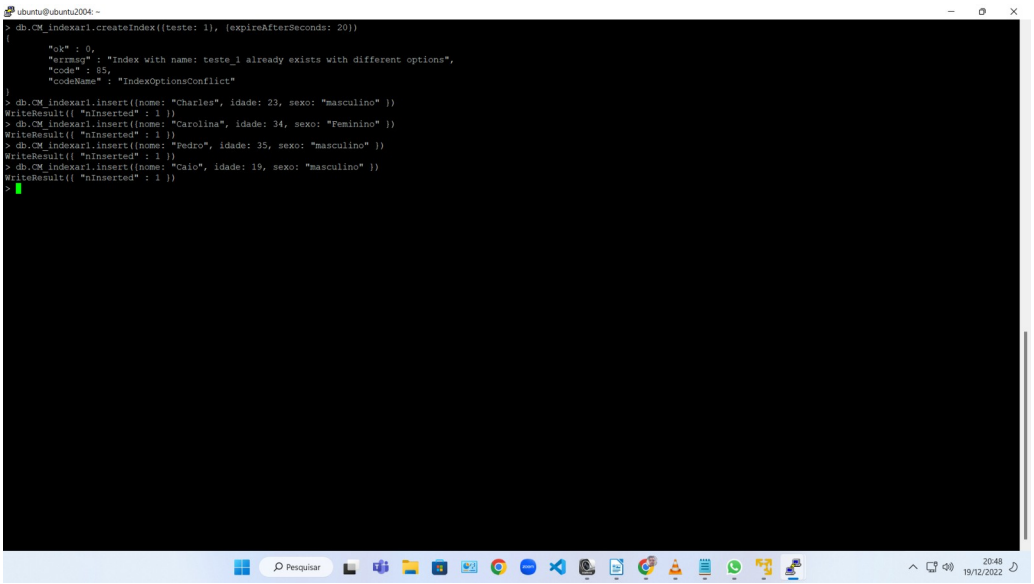
F. Criar um índice com tempo de vida - TTL (com um atributo e com expiração de 20 segundos).

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

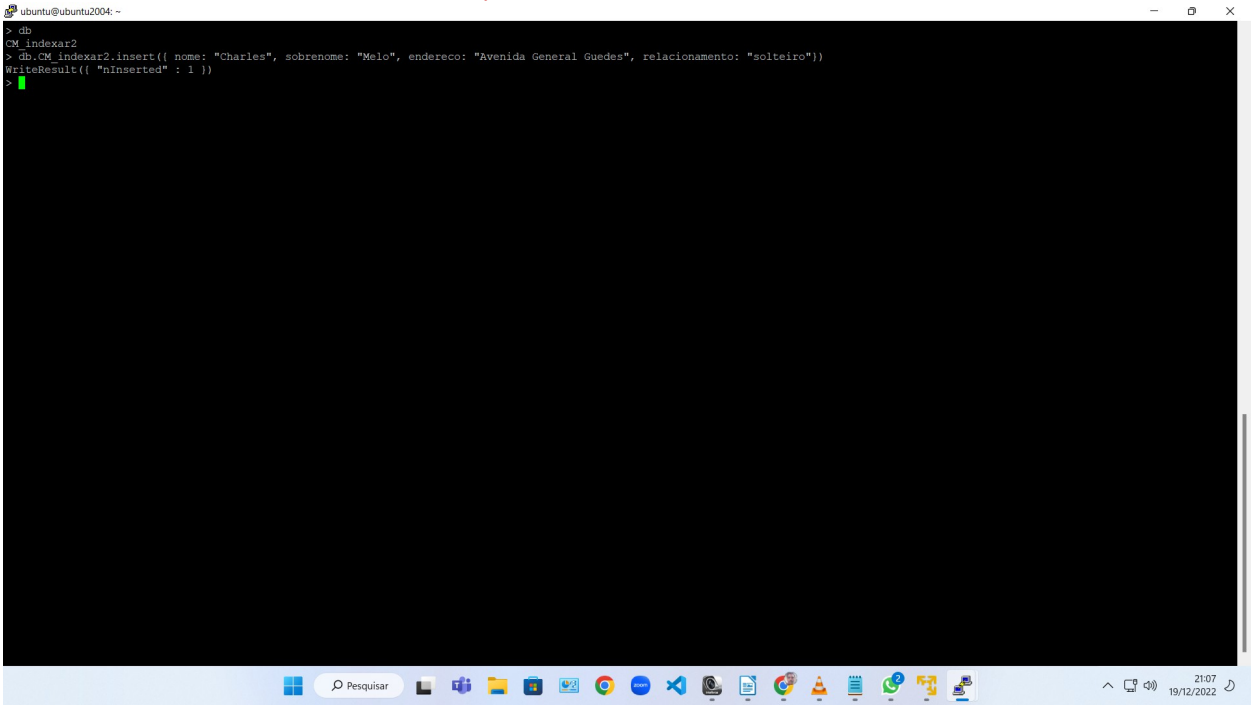


G. Inserir 4 documentos nessa coleção.

```
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" })
```

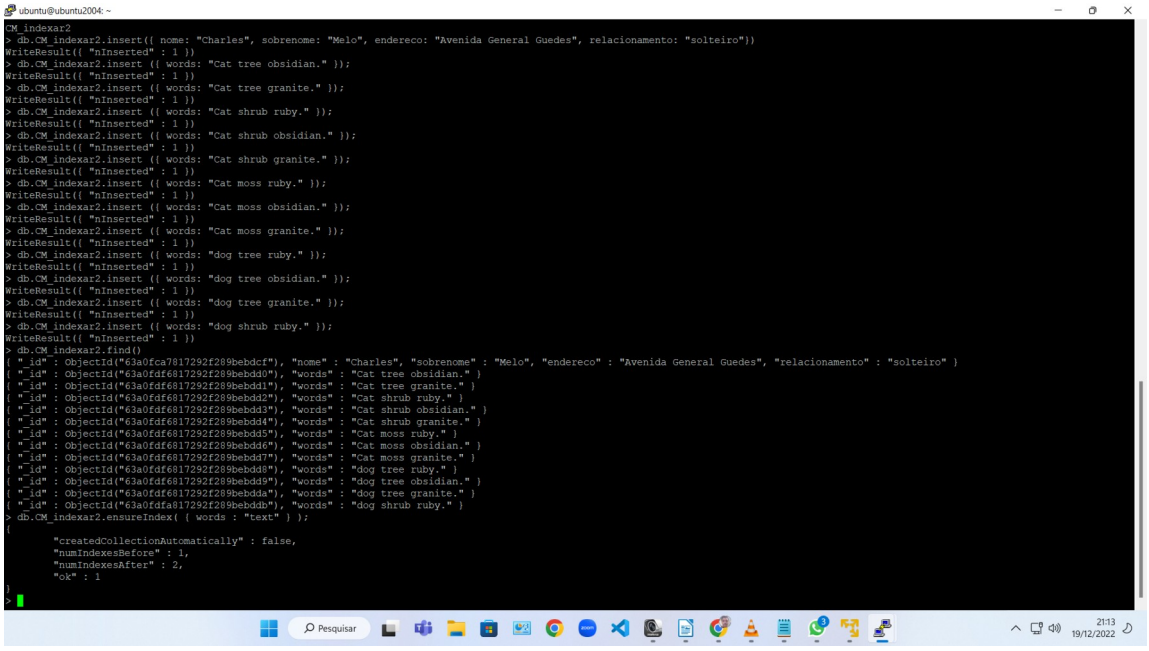


3. Criar uma segunda coleção de nome "&&_indexar2" (que apresente pelo menos 4 atributos string)
db.CM_indexar2.insert({ nome: "Charles", sobrenome: "Melo", endereco: "Avenida General Guedes", relacionamento: "solteiro"})



A. Criar um índice textual para essa segunda coleção.

B. Inserir 4 documentos nessa coleção.

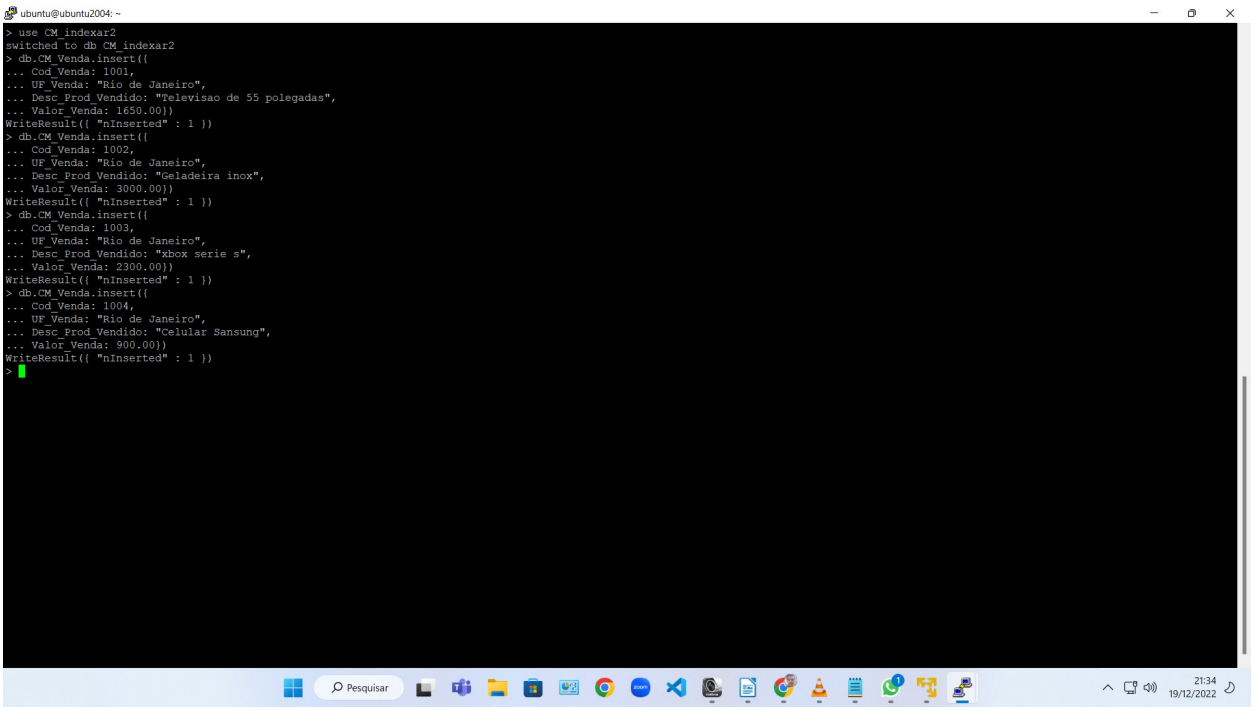


```
db.CM_indexar2.insert ({ words: "Cat tree obsidian." });
db.CM_indexar2.insert ({ words: "Cat tree granite." });
db.CM_indexar2.insert ({ words: "Cat shrub ruby." });
db.CM_indexar2.insert ({ words: "Cat shrub obsidian." });
```

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.

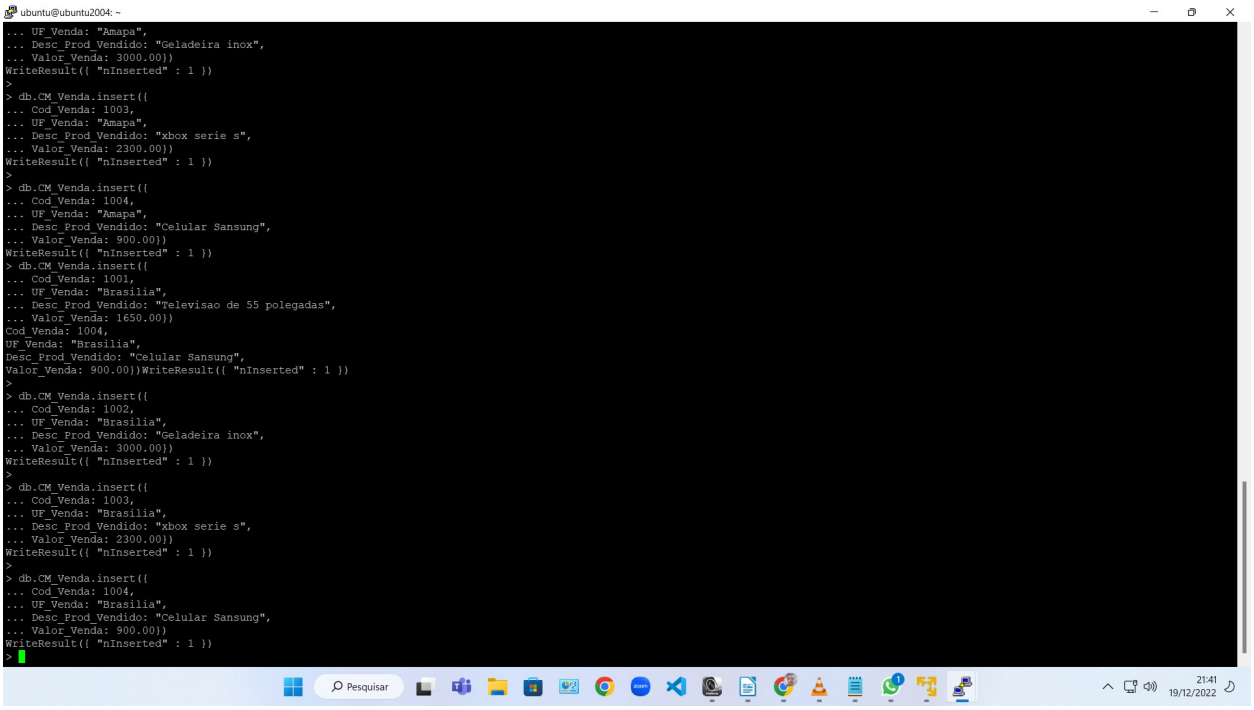
```
db.CM_Venda.insert({
Cod_Venda: 1001,
UF_Venda: "Rio de Janeiro",
Desc_Prod_Vendido: "Televisao de 55 polegadas",
Valor_Venda: 1650.00})

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})
```




```
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",
```

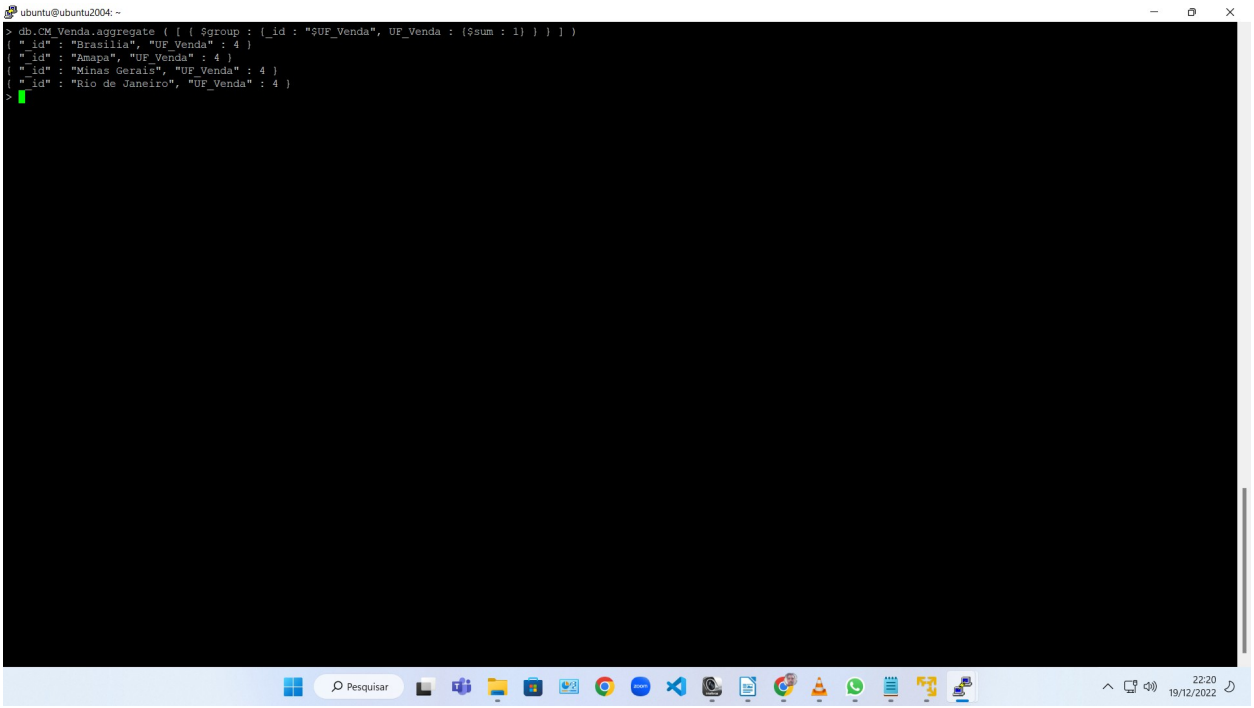
```
Valor_Venda: 900.00})

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})
```



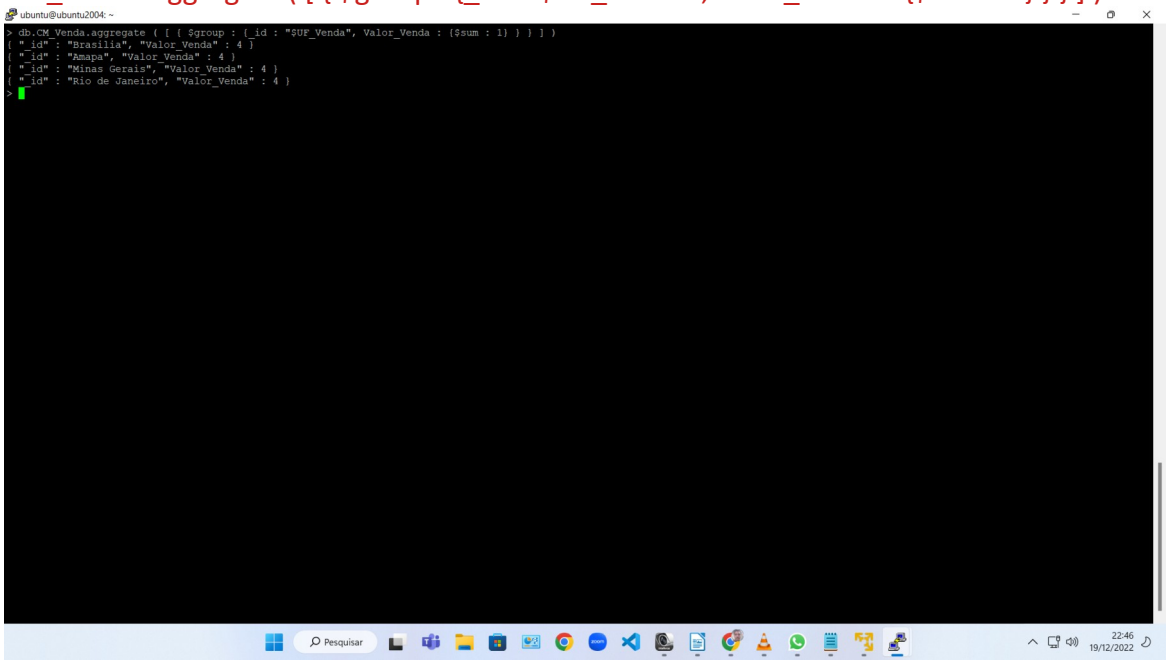
B. Criar uma consulta que mostre o número de documentos por UF.

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



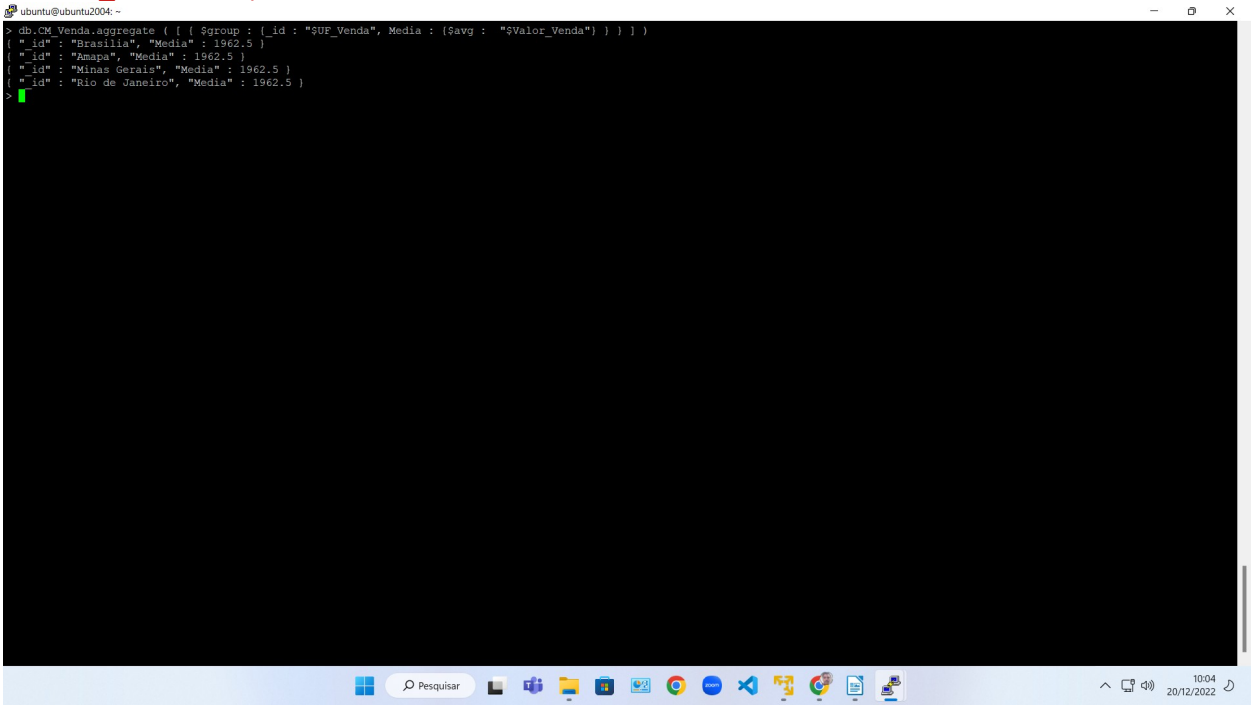
C. Criar uma consulta que mostre o valor total das vendas por UF.

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



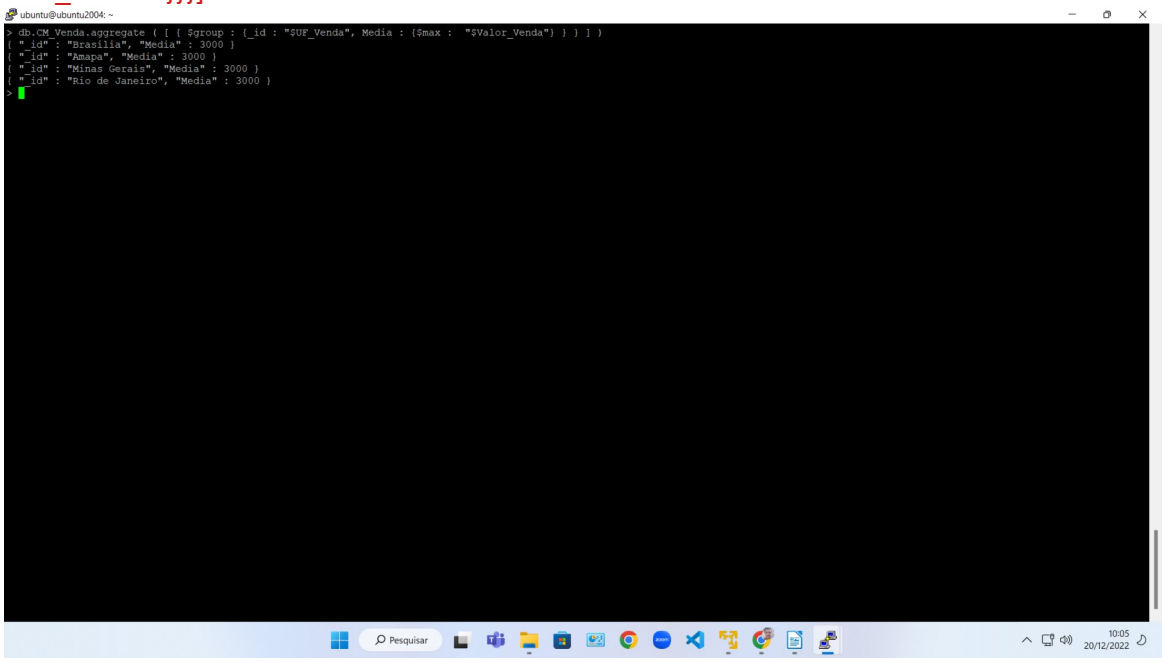
D. Criar uma consulta que mostre o valor de médias das vendas por UF.

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



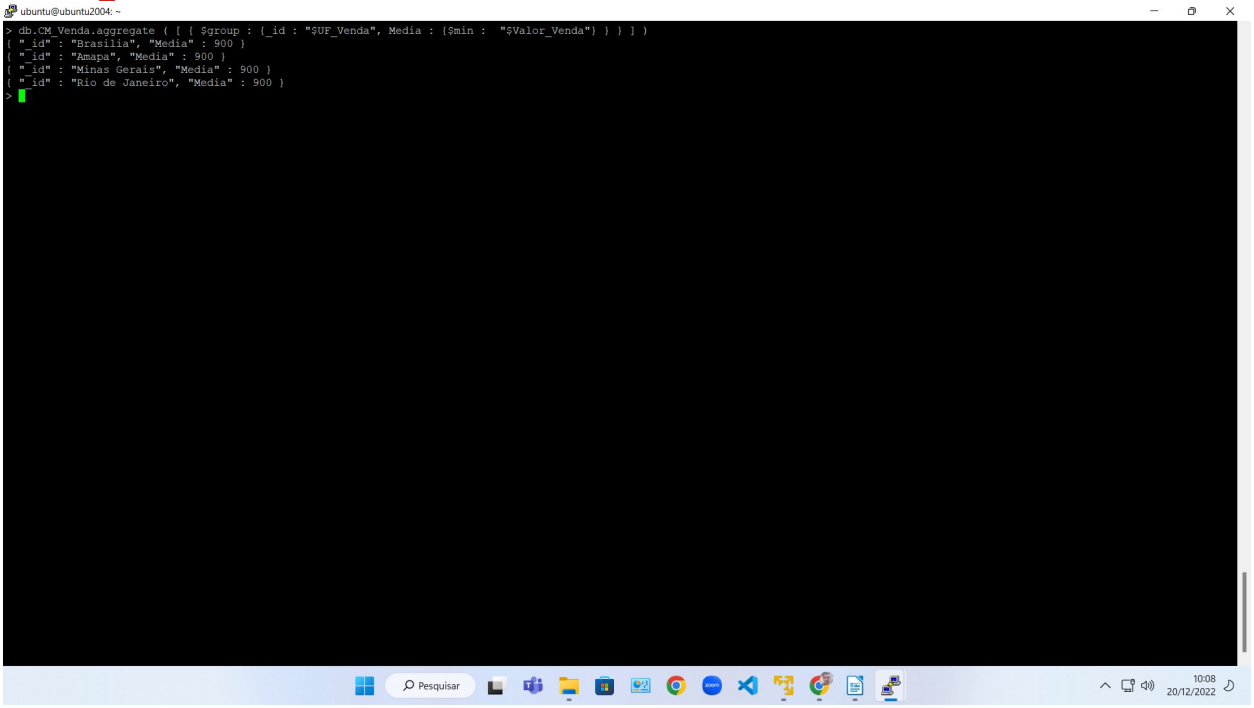
E. Criar uma consulta que mostre o maior valor de venda por UF.

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



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"}}}])
```



Replicação

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

2. Desenho da arquitetura:

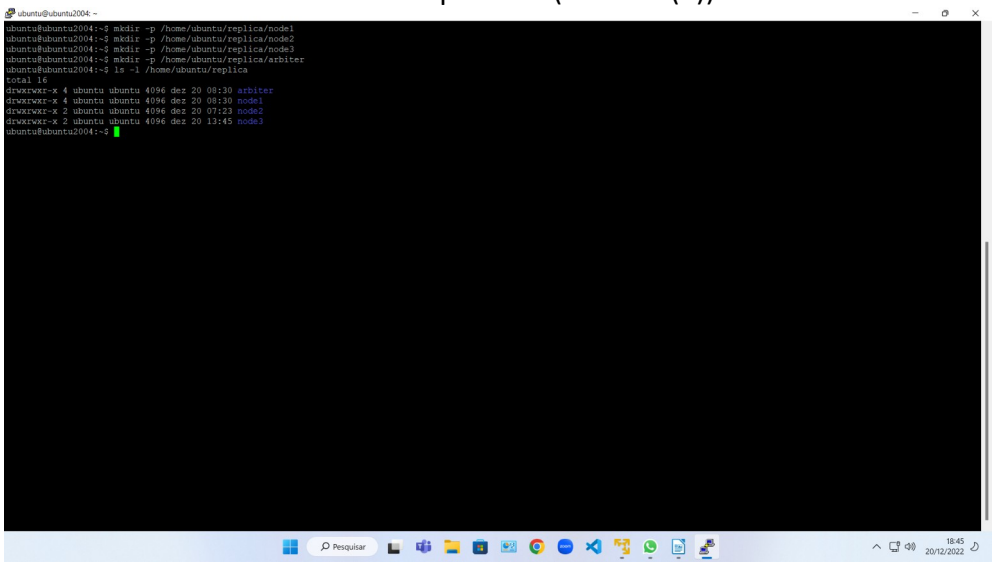
a.Um primário;

b.Dois secundários;

c.Um árbitro.Observações:

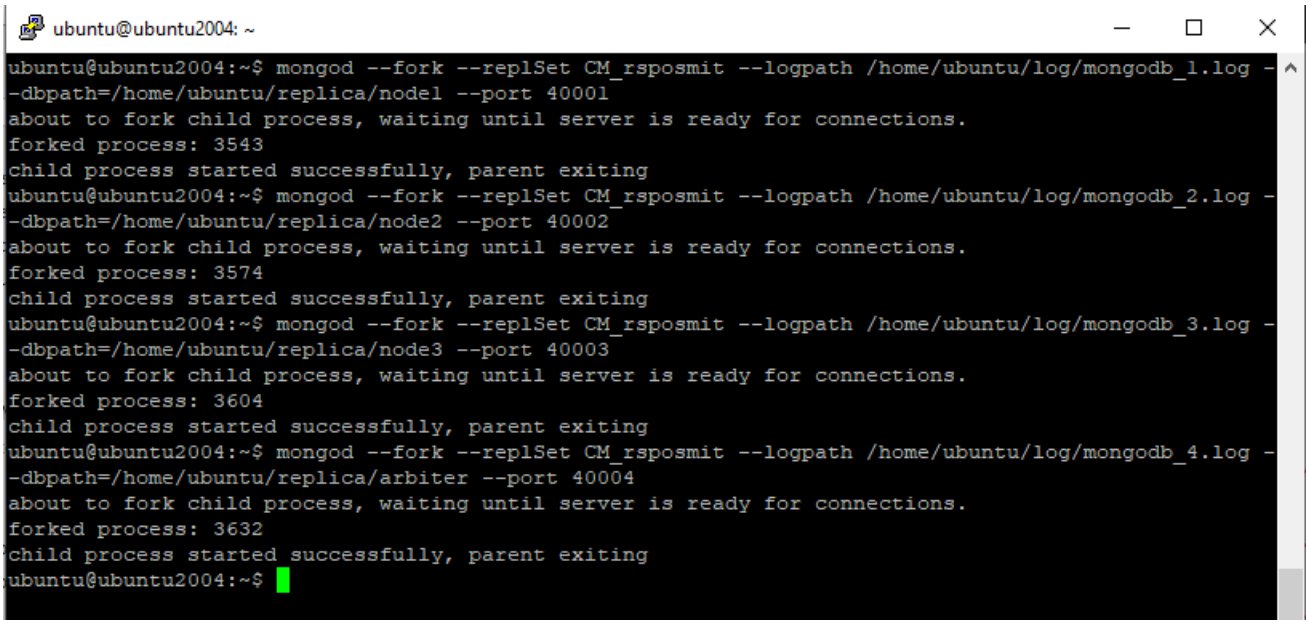
I.Ativar os processos no nível do Sistema Operacional;

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



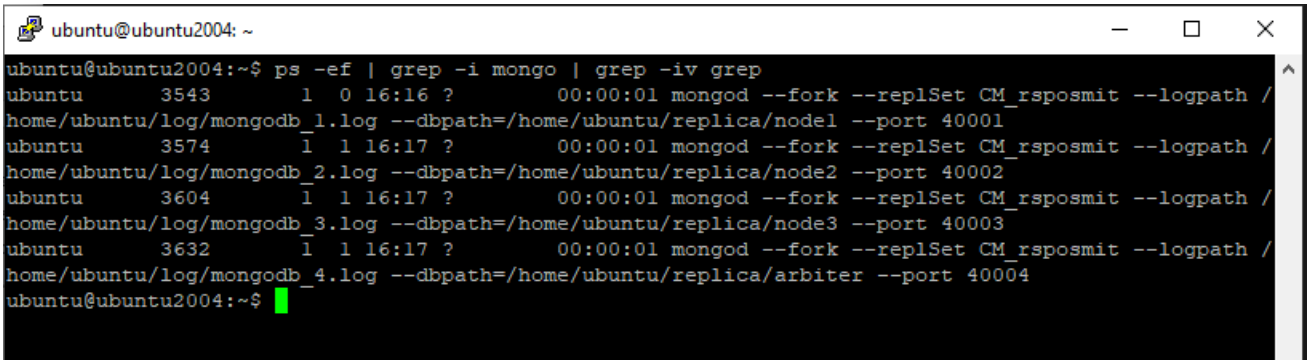
I. Ativar o replica set na visão do sistema operacional.

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



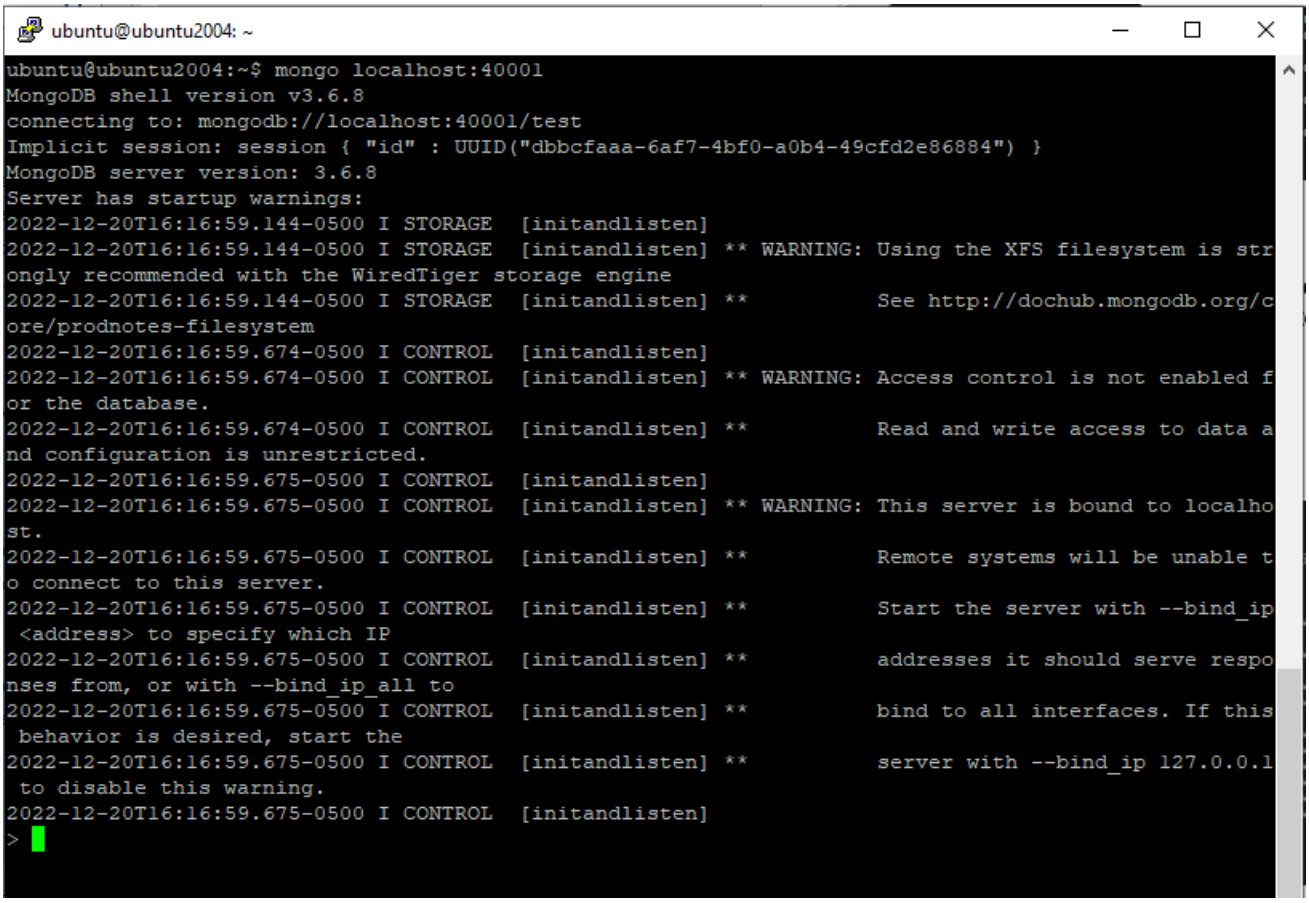
II. Ativar o replica set através do MongoDB, lembrar de adicionar todos os membros ao replica set (ver item (2)).

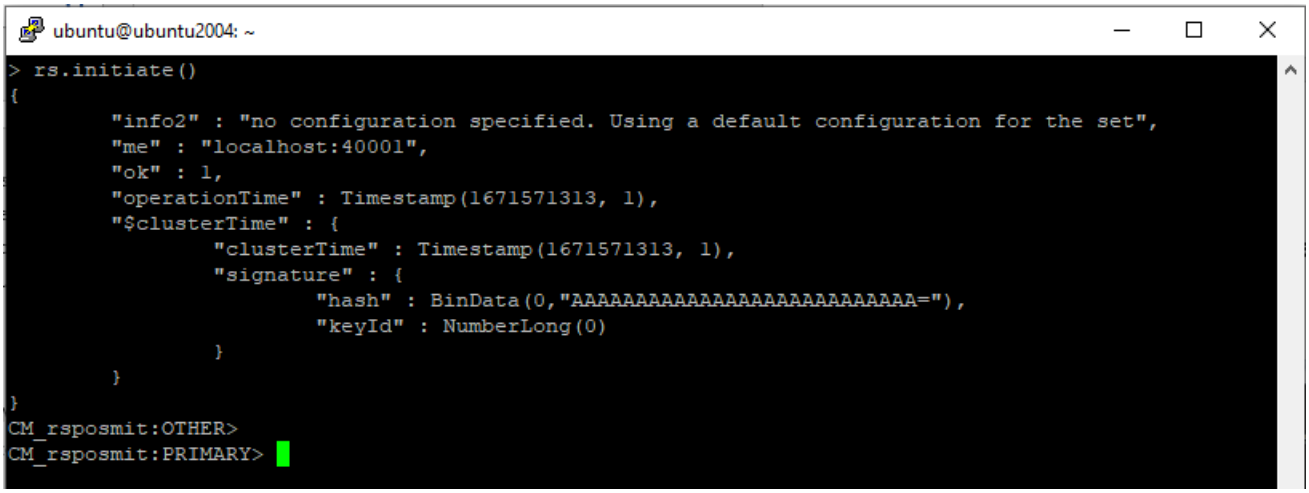
Iniciando o MongoDB Shell
mongo localhost:40001



Verificando novamente o status do replica set

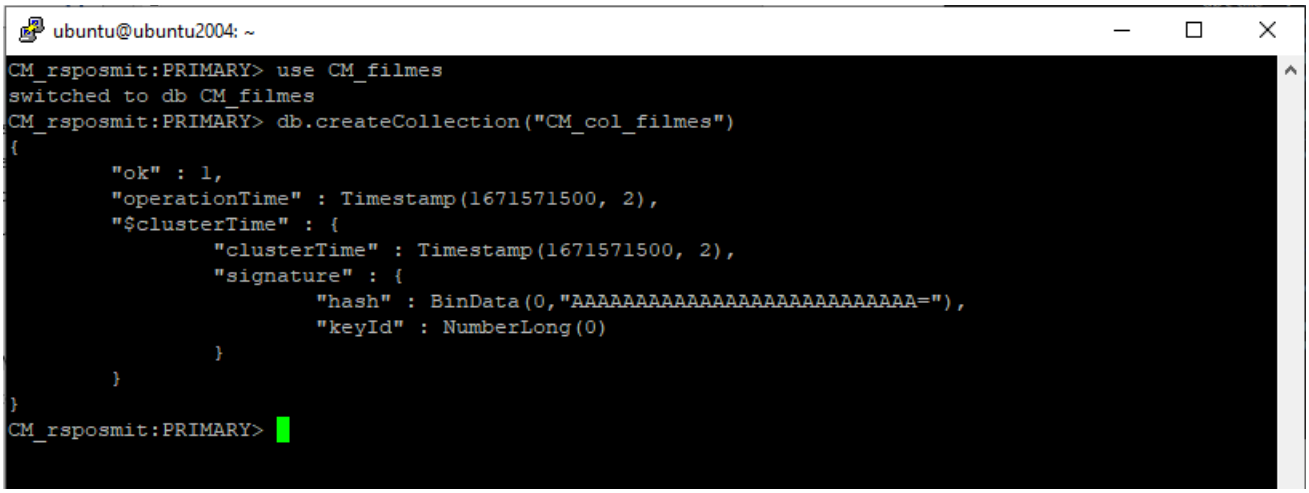
rs_status()





Mudando para o banco de dados CM_filmes

Use CM_filmes



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

Inserir 5 documentos na coleção "&&_col_filmes".

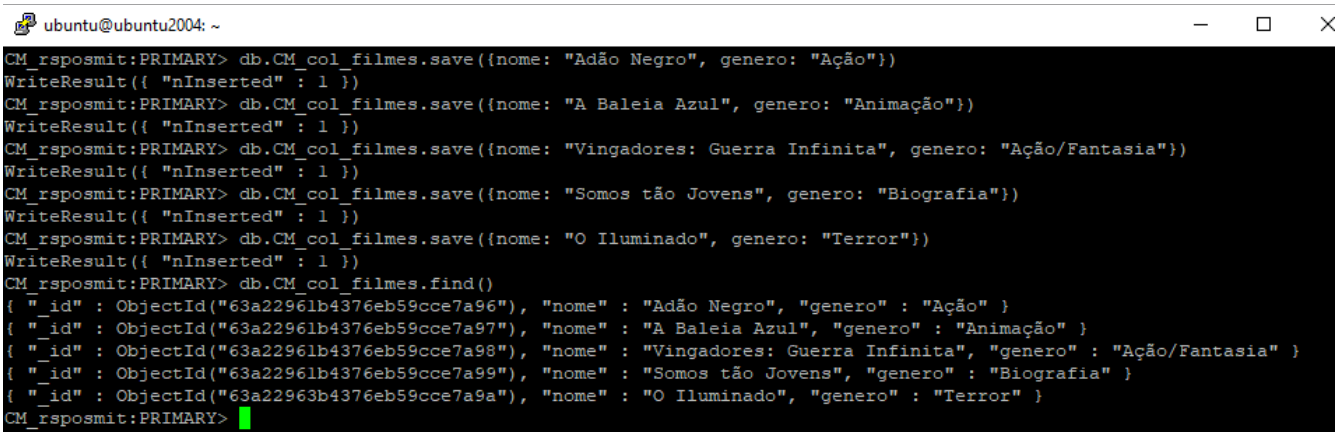
db.CM_col_filmes.save({nome: "Adão Negro", genero: "Ação"})

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

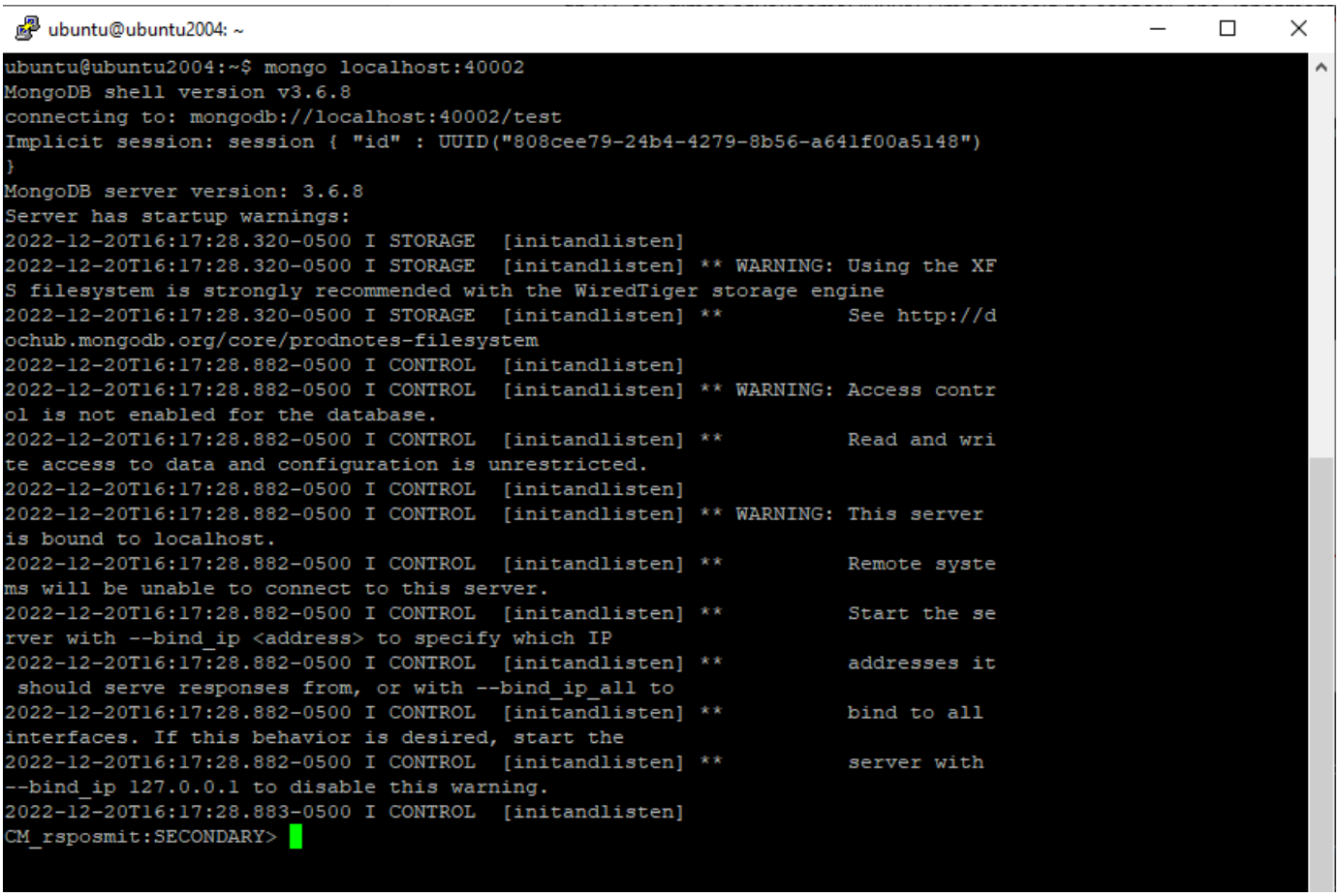
db.CM_col_filmes.save({nome: "Vingadores: Guerra Infinita", genero: "Ação/Fantasia"})

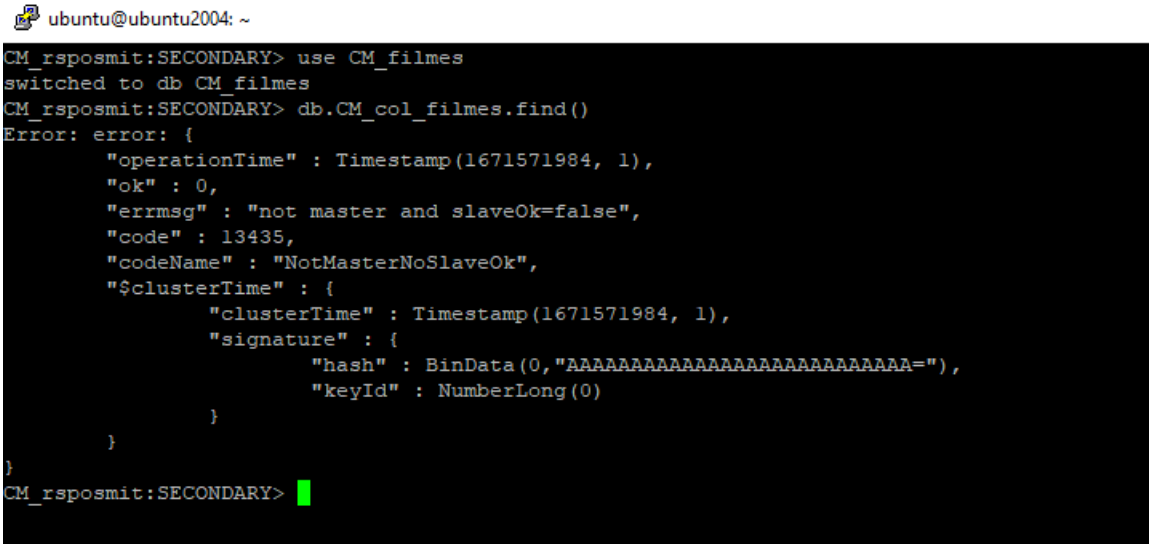
db.CM_col_filmes.save({nome: "Somos tão Jovens", genero: "Biografia"})

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

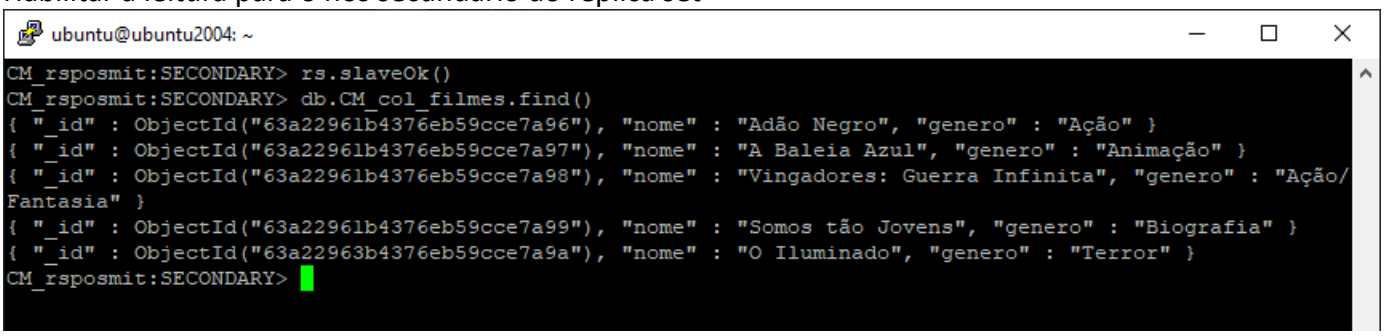


4. Acessar um dos nós secundários.
Efetuar a leitura da coleção "&&_col_filmes".
mongo localhost:40002





Habilitar a leitura para o nós secundário do replica set



E. Particionamento

1.Criar um shard.

2.Desenho da arquitetura:

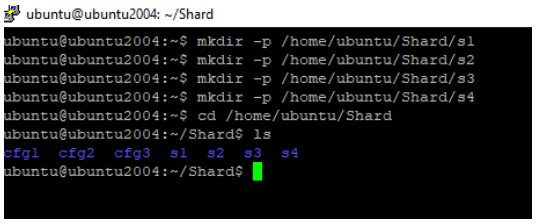
a.Três config servers (no Replica Set);

b.Quatro shard servers (sem Replica Set);

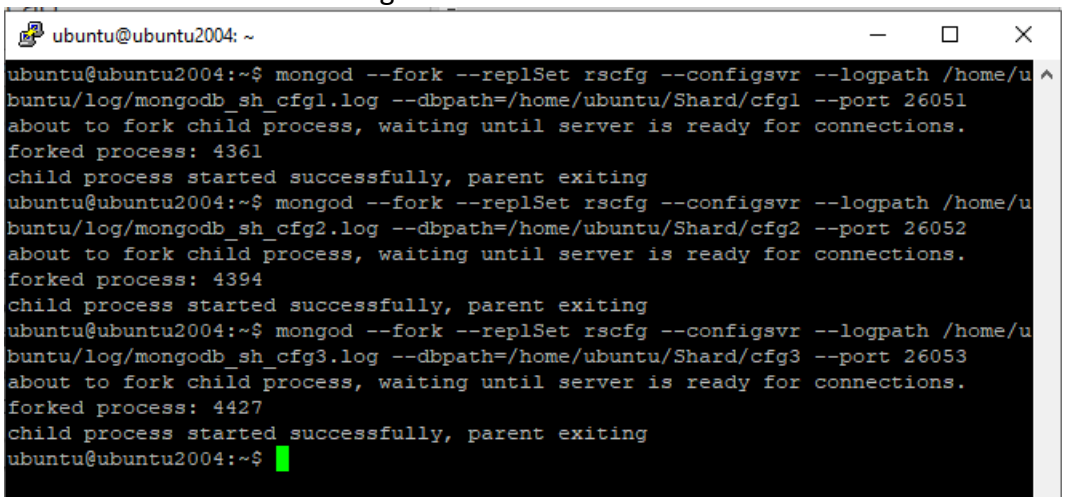
c.Um mongos.Observações:

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).



Criando os diretórios do config servers

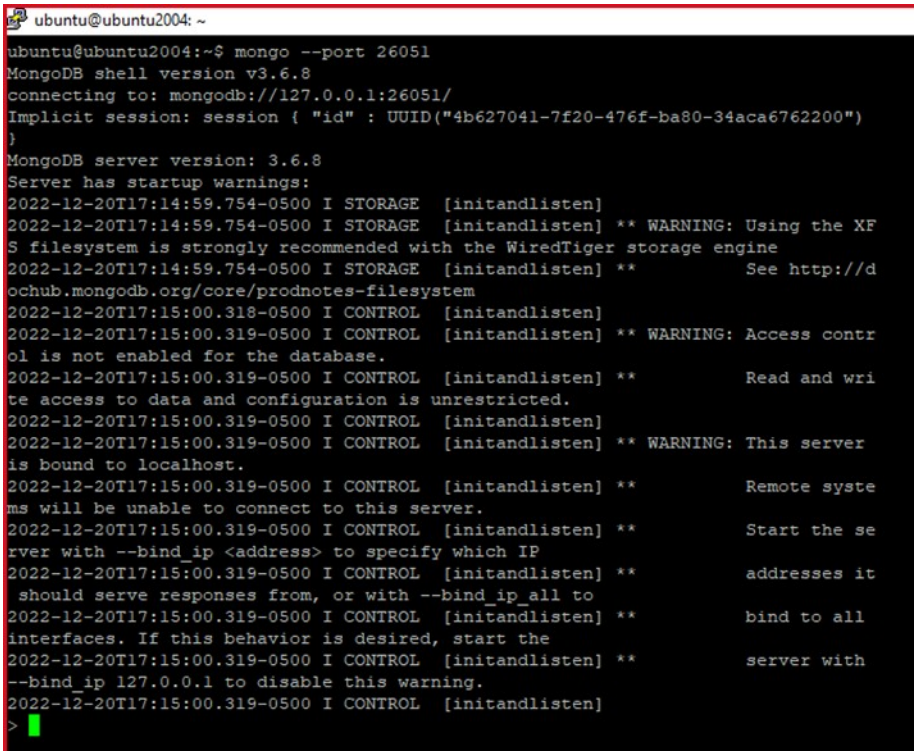


mongod --fork --replSet rscfg --configsvr --logpath /home/ubuntu/log/mongodb_sh_cfg1.log --dbpath=/home/ubuntu/Shard/cfg1 --port 26051

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



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: ~

```
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

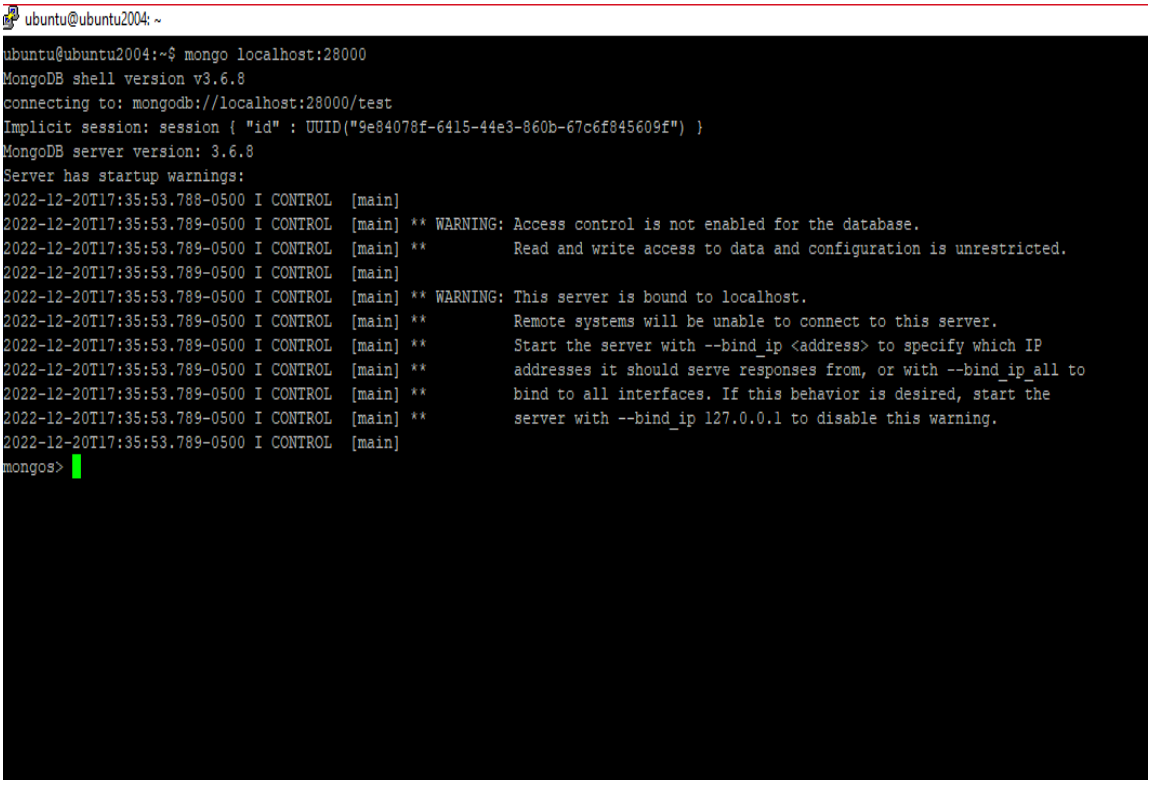
ubuntu@ubuntu2004: ~

```
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



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),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1671576934, 5),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
mongos> sh.addShard("localhost:27052")
{
        "shardAdded" : "shard0001",
        "ok" : 1,
        "operationTime" : Timestamp(1671576940, 3),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1671576940, 3),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
mongos> sh.addShard("localhost:27053")
{
        "shardAdded" : "shard0002",
        "ok" : 1,
        "operationTime" : Timestamp(1671576946, 3),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1671576946, 3),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
mongos> sh.addShard("localhost:27054")
{
        "shardAdded" : "shard0003",
        "ok" : 1,
        "operationTime" : Timestamp(1671576952, 3),
        "$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,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "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,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "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" }) }

ubuntu@ubuntu2004: ~

```
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()

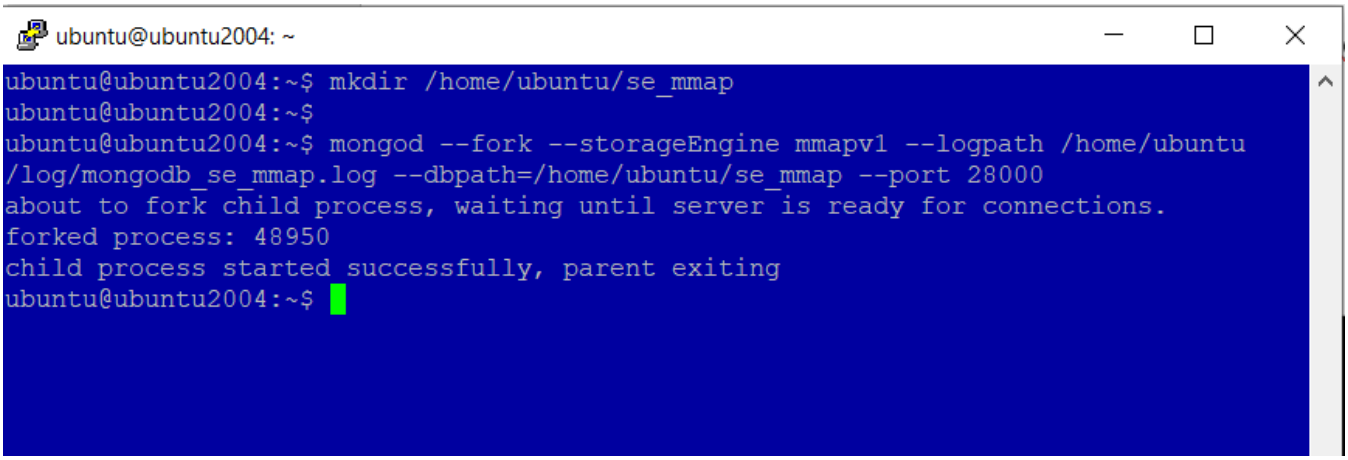
ubuntu@ubuntu2004: ~

```
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

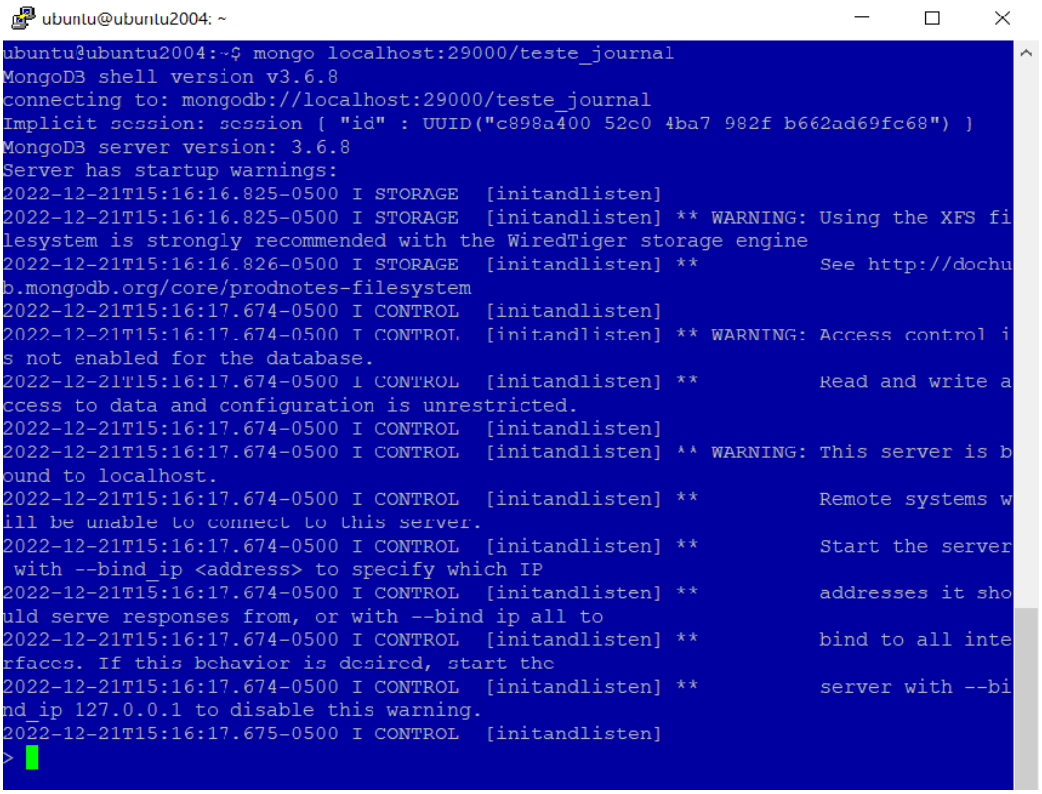
1. Criar uma instância do MongoDB que use o storage engine mmapv1.

mkdir /home/ubuntu/se_mmap



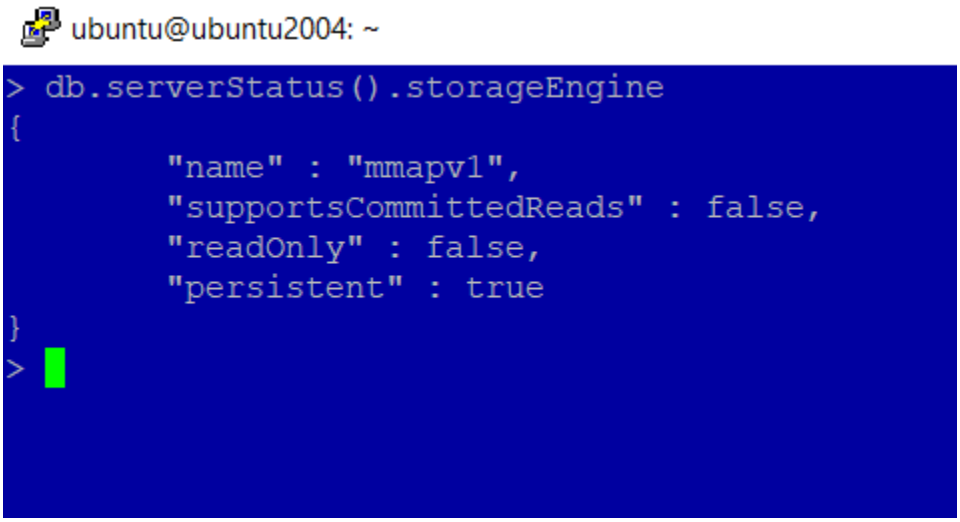
A. Conectar a essa instância;

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



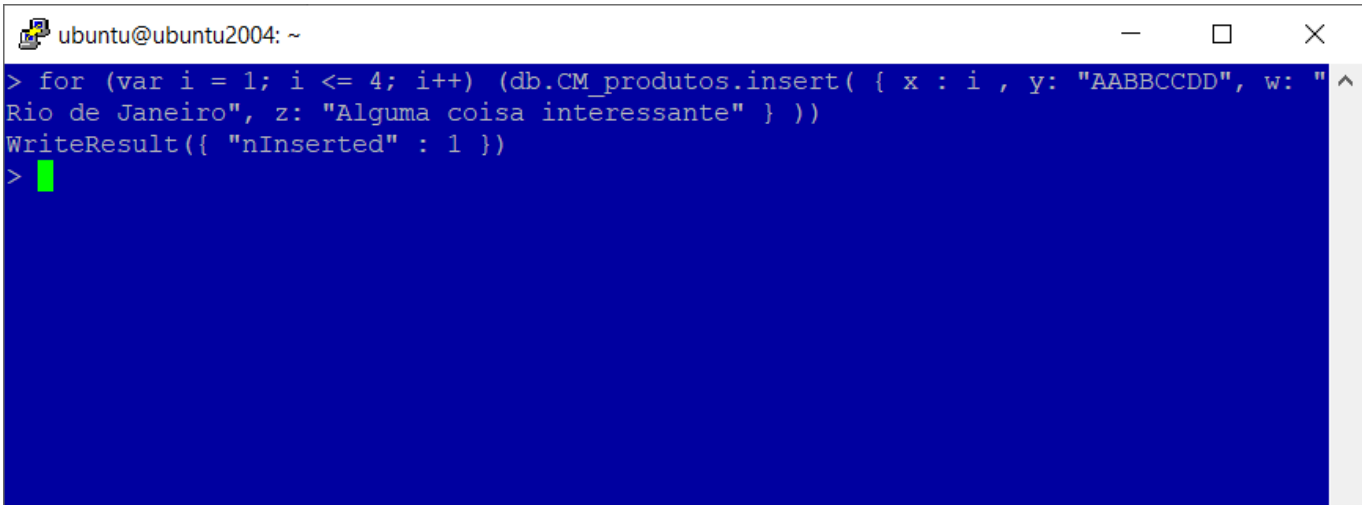
B. Verificar o storage engine corrente;

db.serverStatus().storageEngine



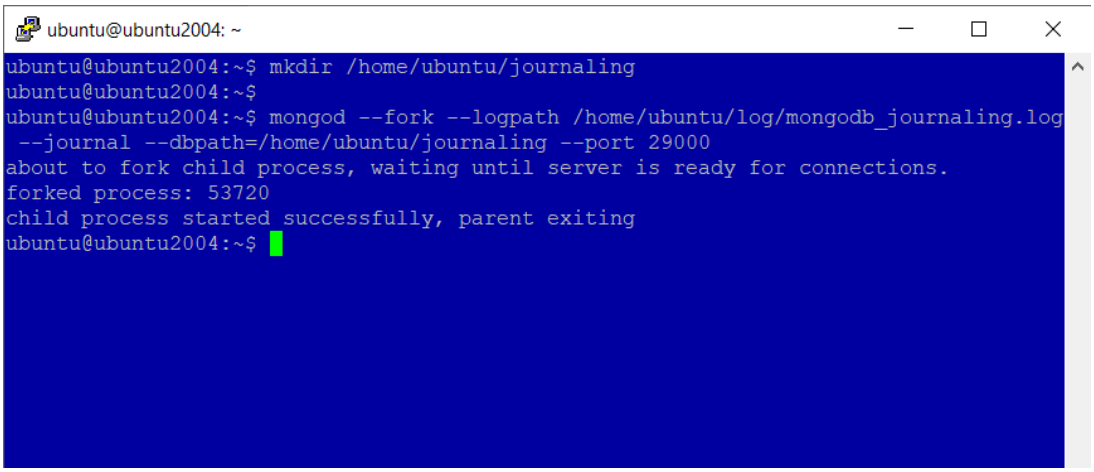
C. Criar a coleção “&&_produtos” e inserir 4 documentos.

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



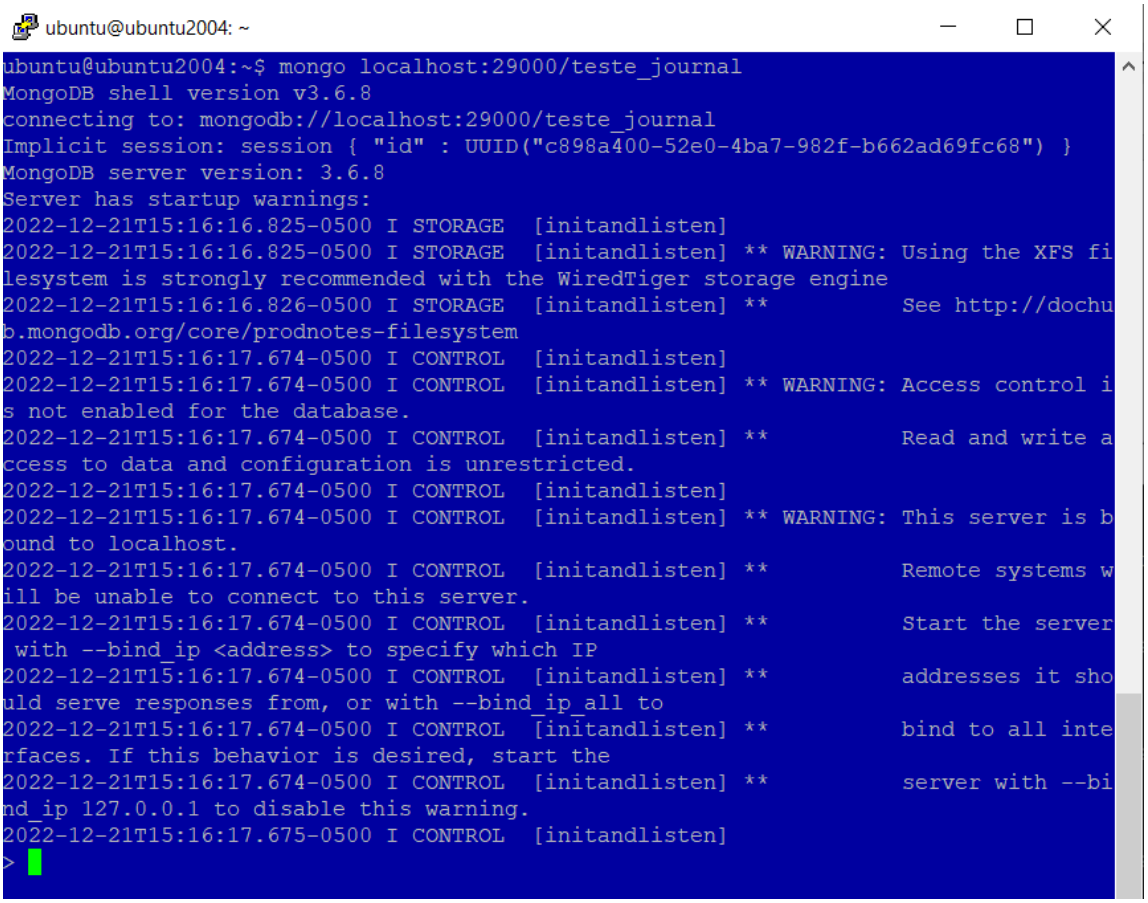
2. Criar uma instância do MongoDB que use o storage engine wiredTiger.

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



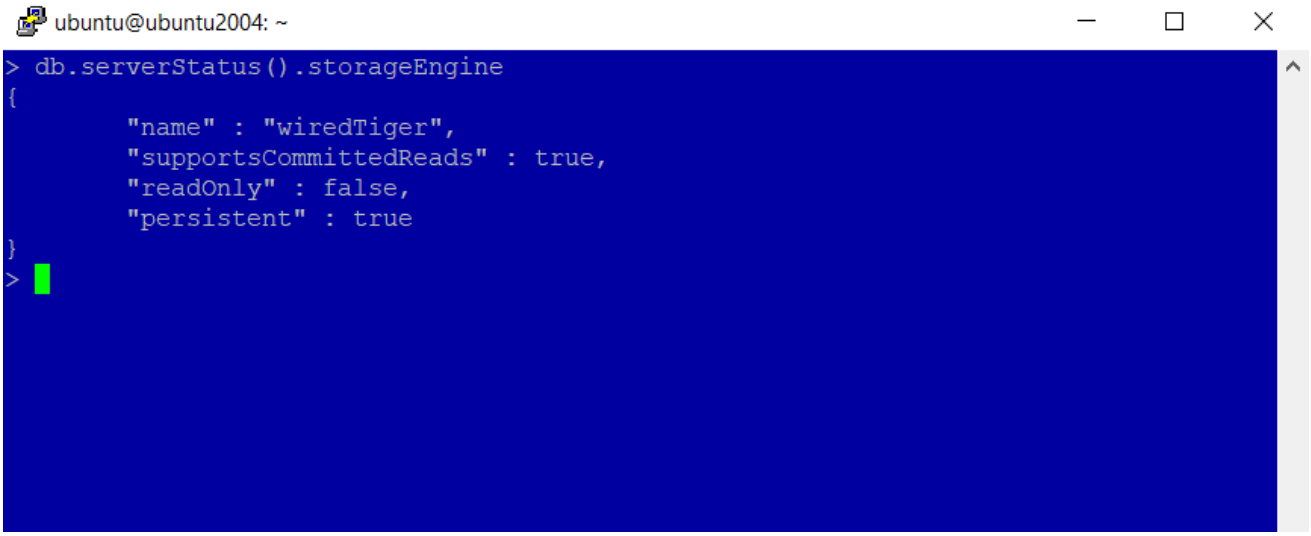
A. Conectar a essa instância;

mongo localhost:29000/teste_journal



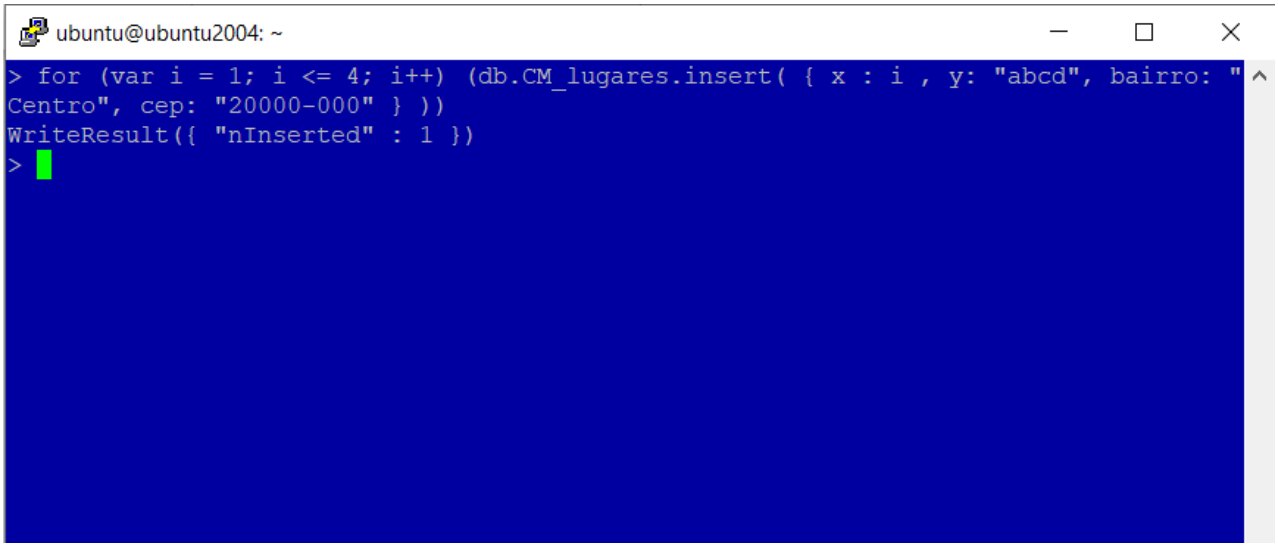
B. Verificar o storage engine corrente;

db.serverStatus().storageEngine



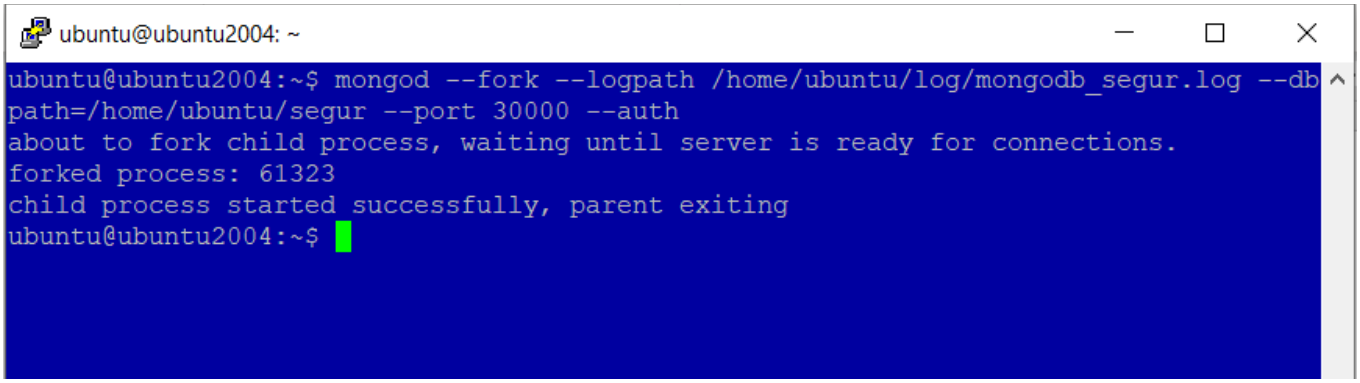
C. Criar a coleção "&&_lugares" e inserir 4 documentos.

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



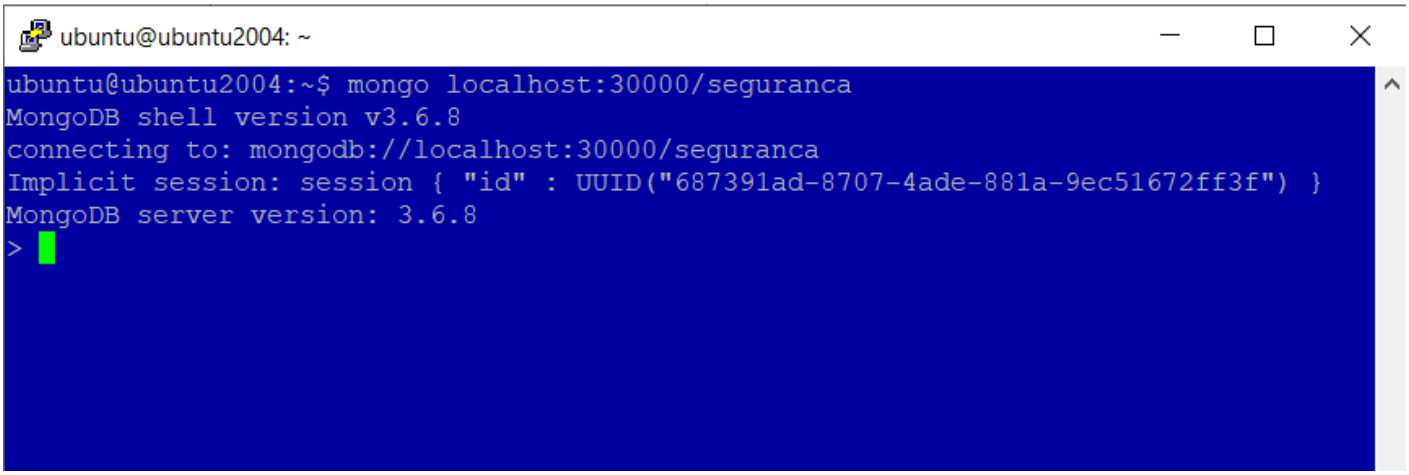
3. Criar uma instância do MongoDB que use segurança (autenticação e autorização)

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



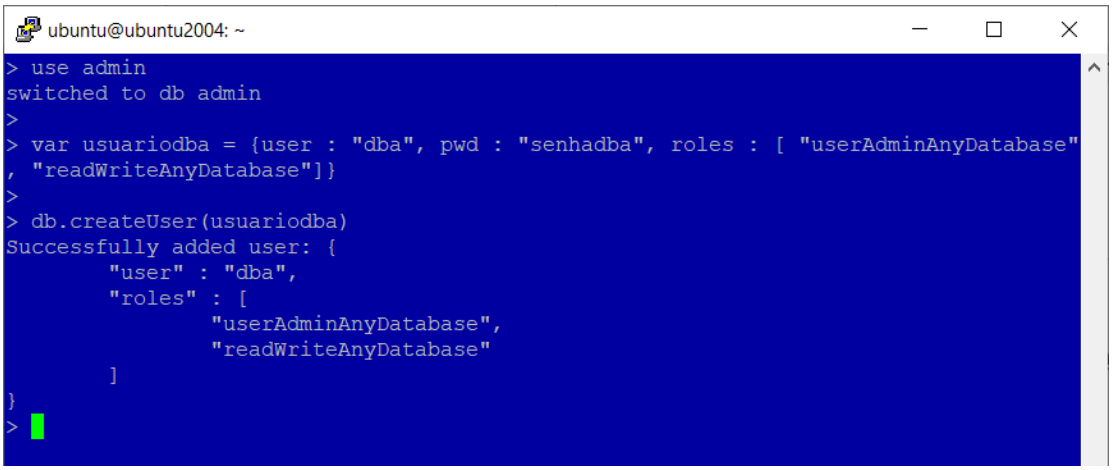
A. Conectar a essa instância;

mongo localhost:30000/seguranca



B. Criar o usuário dba com a role root;

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

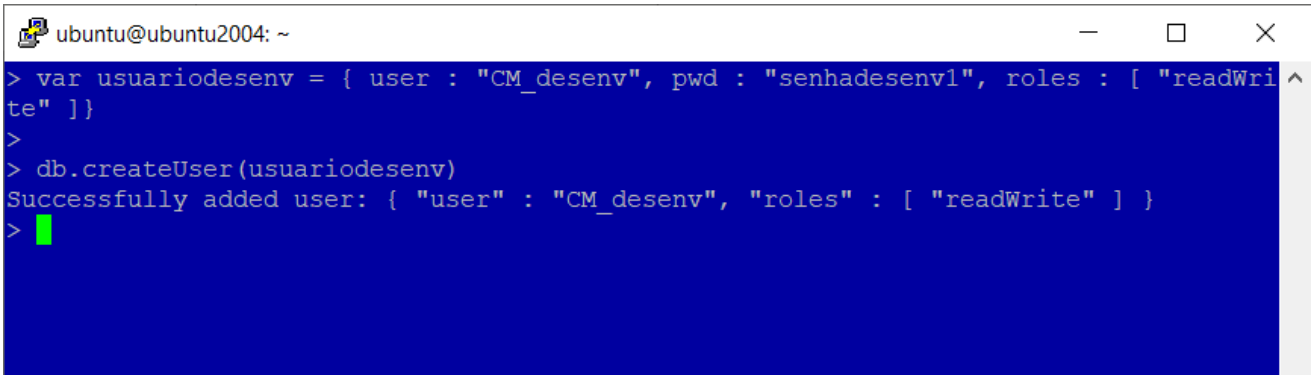


C. Conectar com o usuário dba;

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

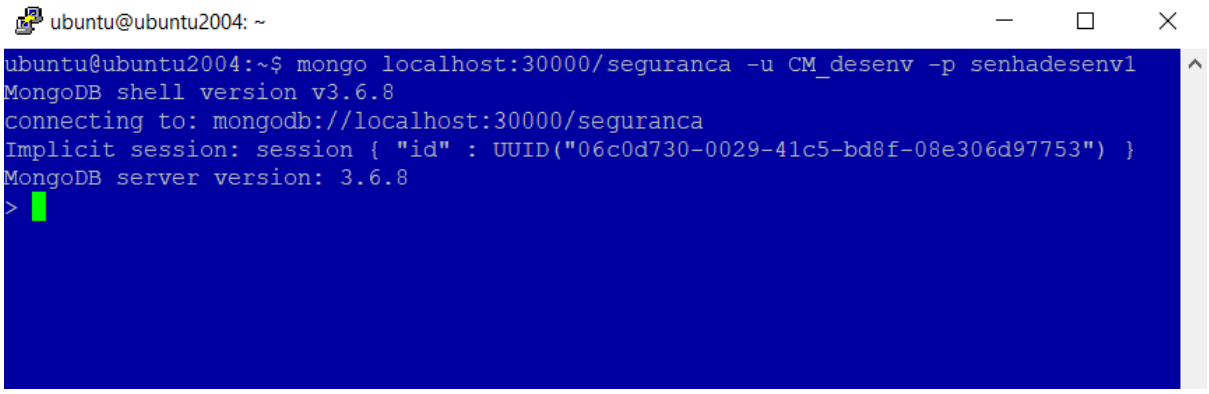
D. Criar o usuário “&&_desenv” com a role readWrite no database “&&_rh”;

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

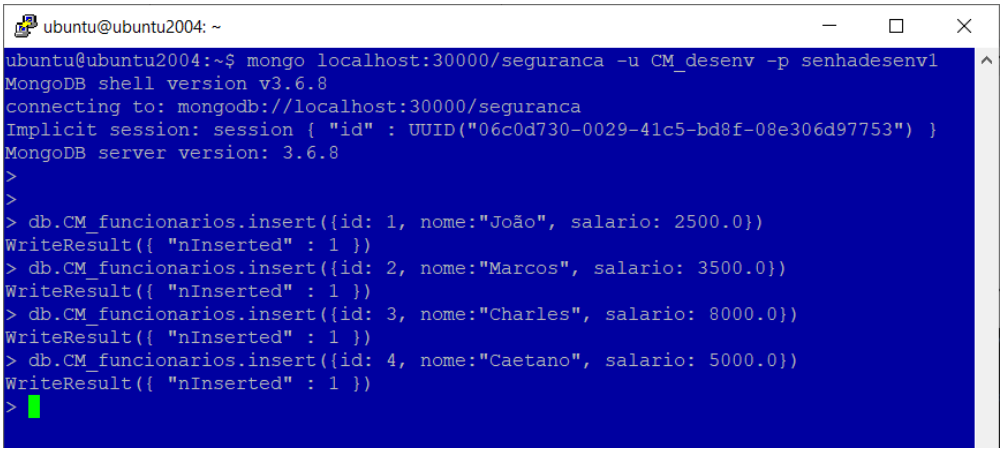


E. Conectar com o usuário “&&_desenv”

mongo localhost:30000/seguranca -u CM_desev -p senhadesenv1



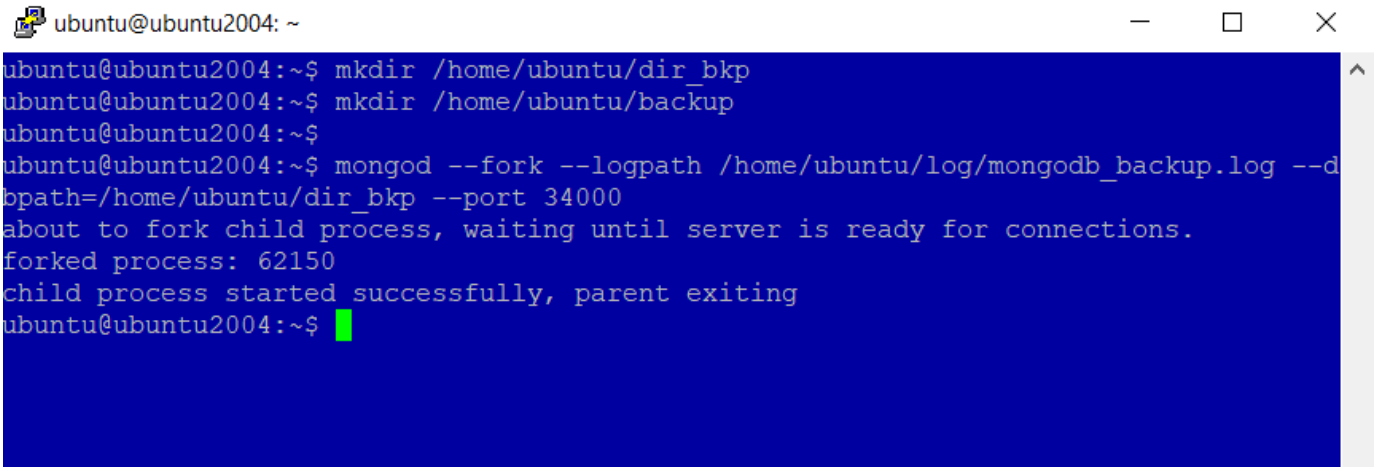
F. Criar a coleção “&&_funcionarios” e inserir 4 documentos.



Depuração, Backup/Restore

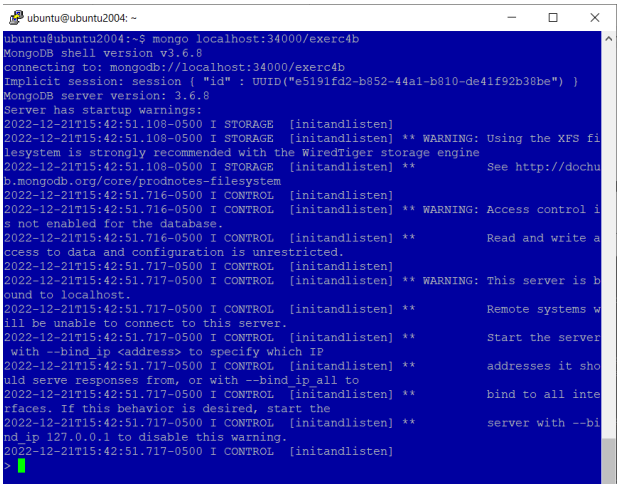
1. Criar uma instância do MongoDB.

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



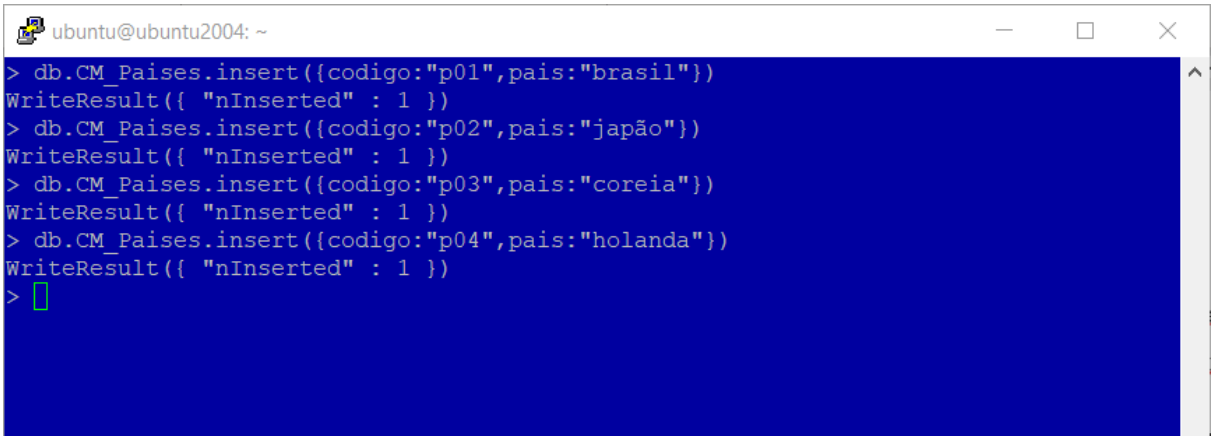
2. Conectar a essa instância (acessar o database exerc4b);

mongo localhost:34000/exerc4b



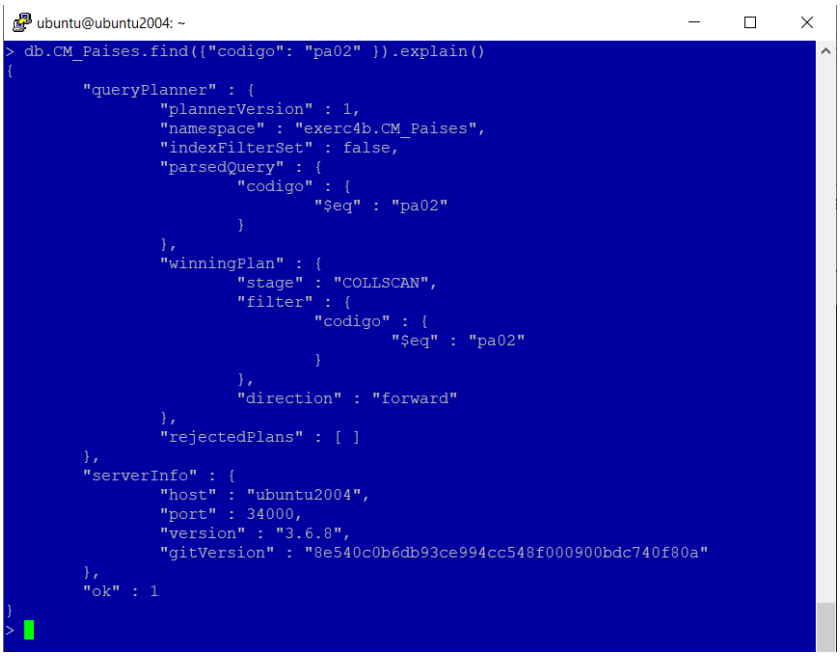
3. Criar a coleção "&&_Paises" e inserir 4 documentos.

```
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"})
```



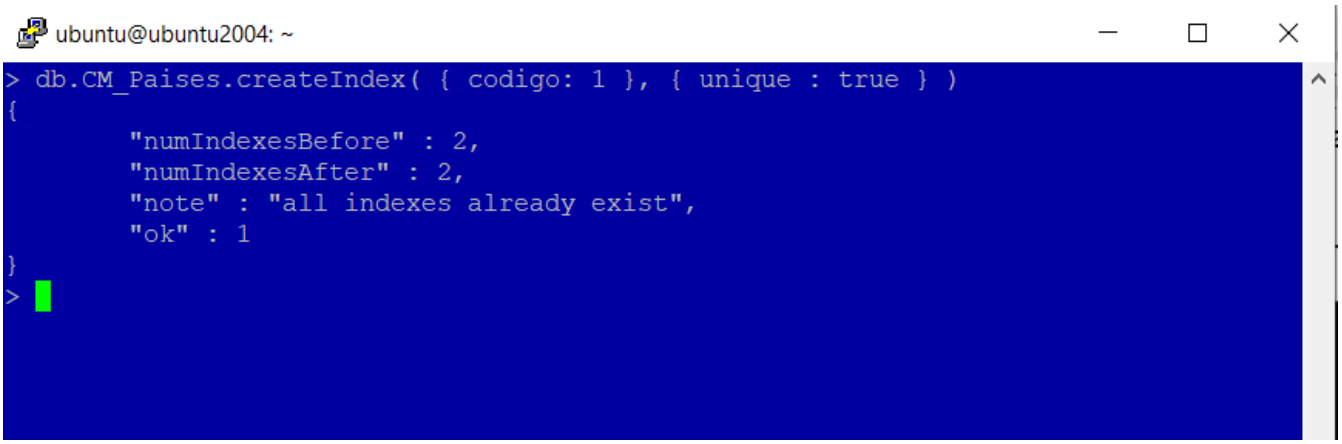
4. Executar uma consulta para a coleção "&&_Paises" com um filtro de sua escolha com o explain.

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



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

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



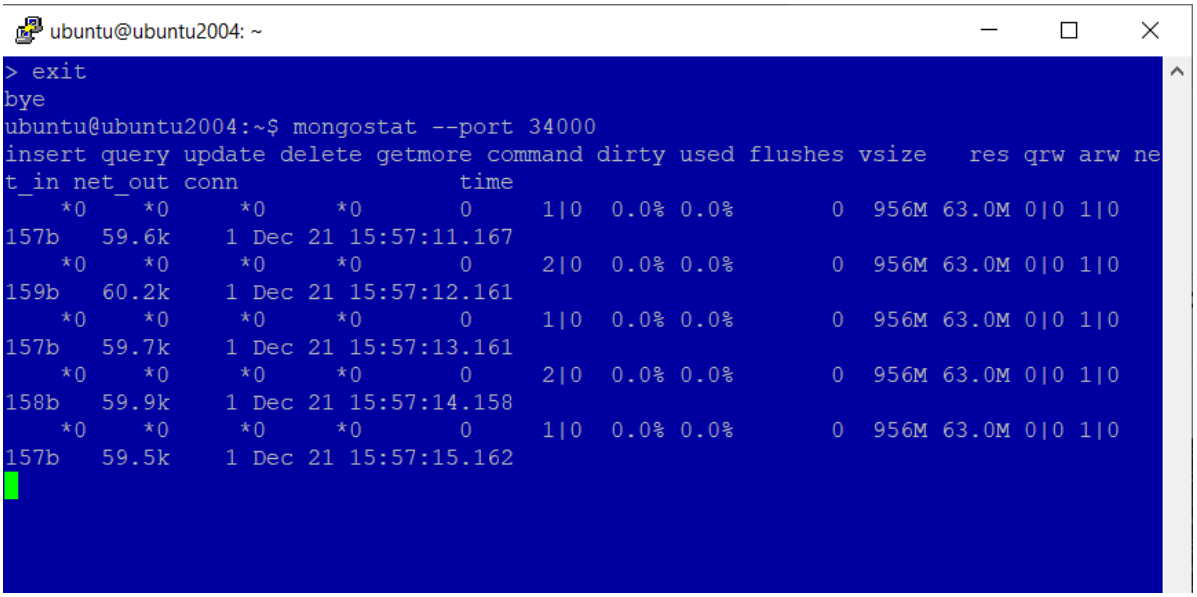
6. Executar a mesma consulta do item (1.c).

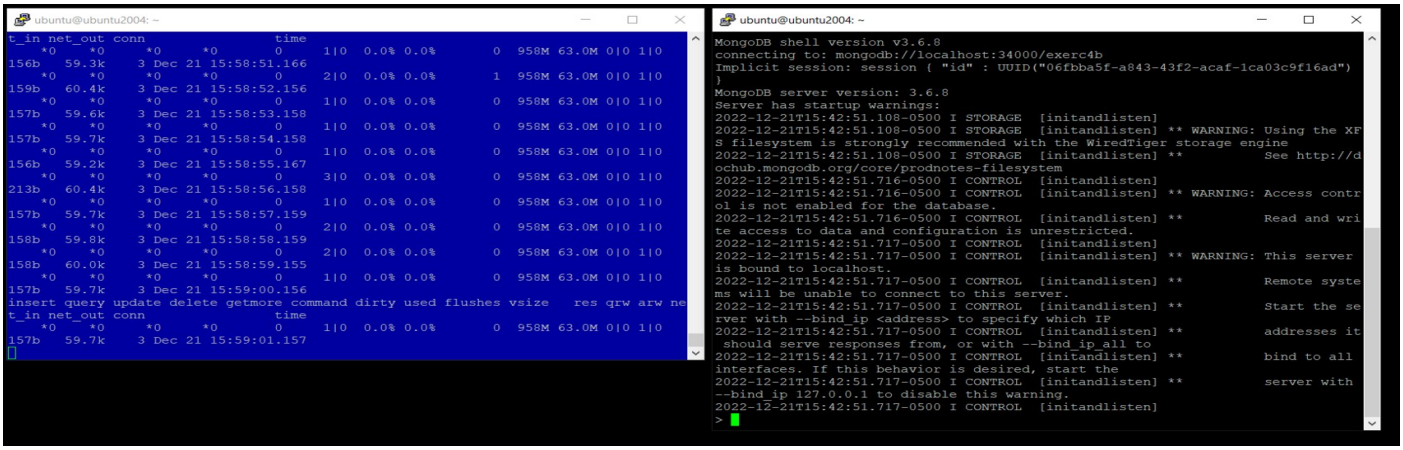
db.CM_Paises.find({"codigo": "pa02" }).explain()

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_Numero1.insert({ x : i}) }

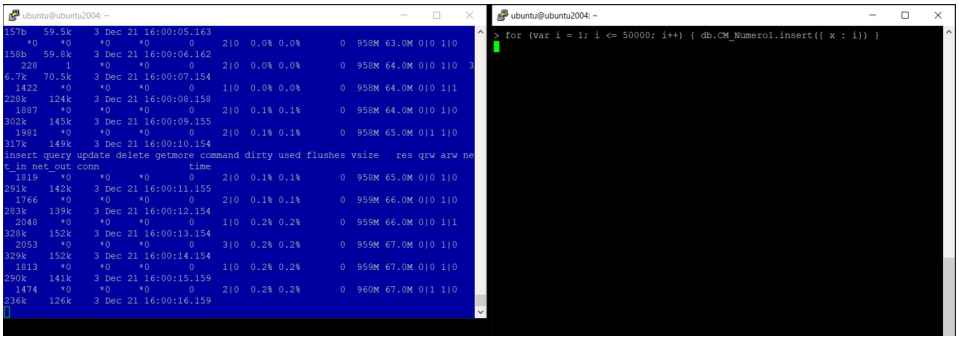


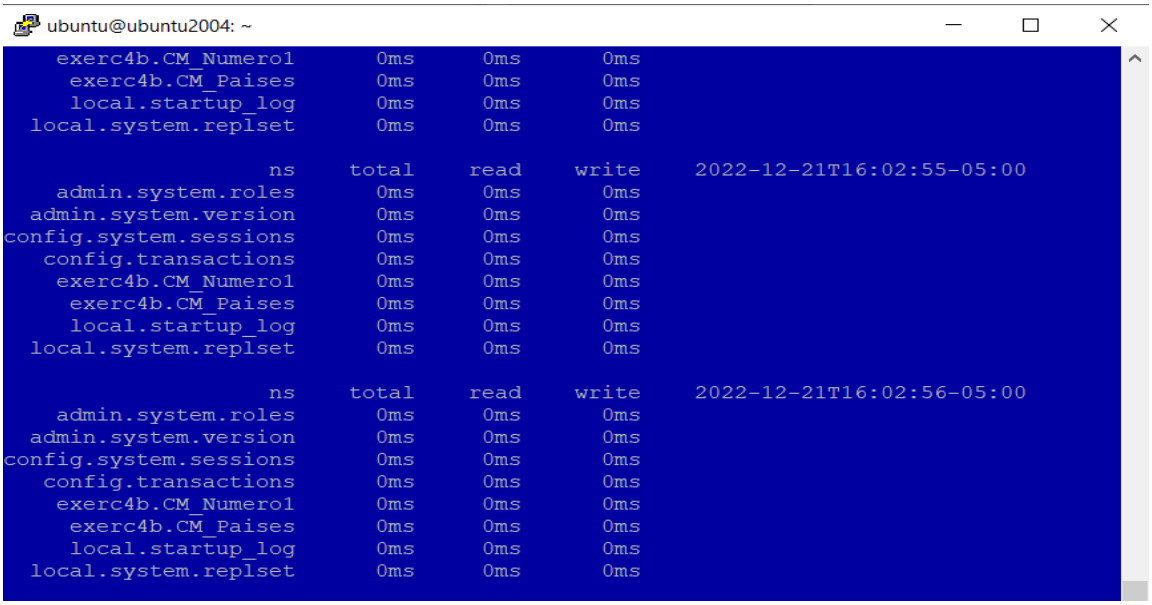


8. Criar a coleção “&&_Numero2” e inserir 50.000 documentos em paralelo; numa outra janela executar o utilitário mongotop.

mongotop --port 34000

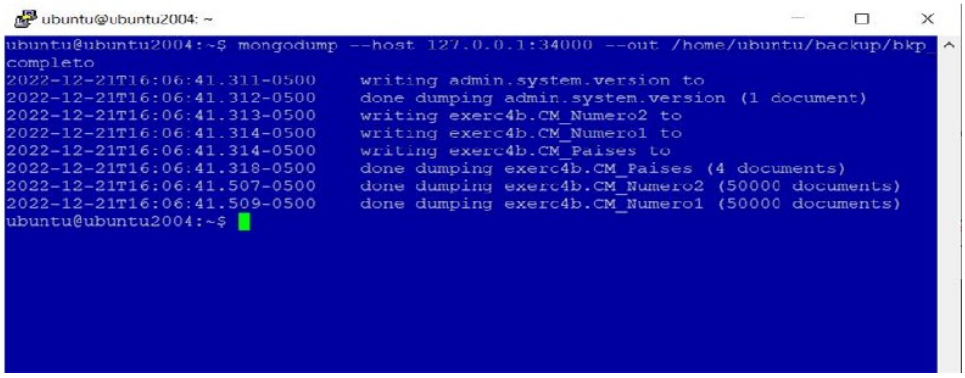
for (var i = 1; i <= 50000; i++) { db.CM_Numero2.insert({ y : i}) }





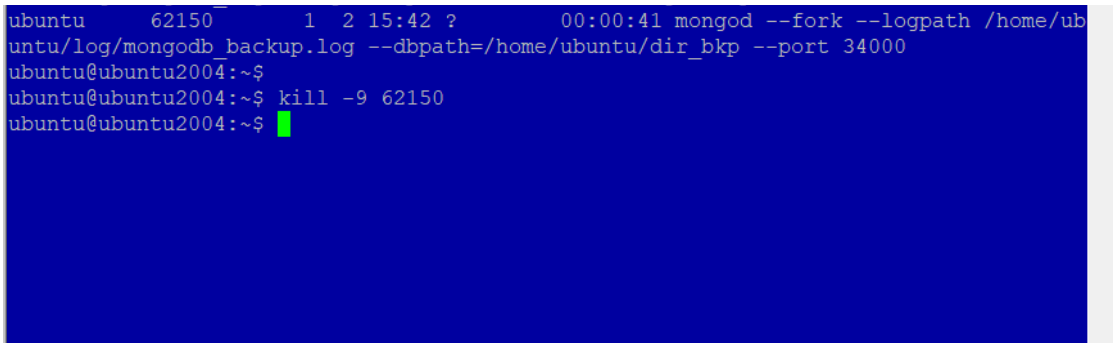
9. Sair do Mongo Shell e fazer um backup da instância inteira do MongoDB.

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



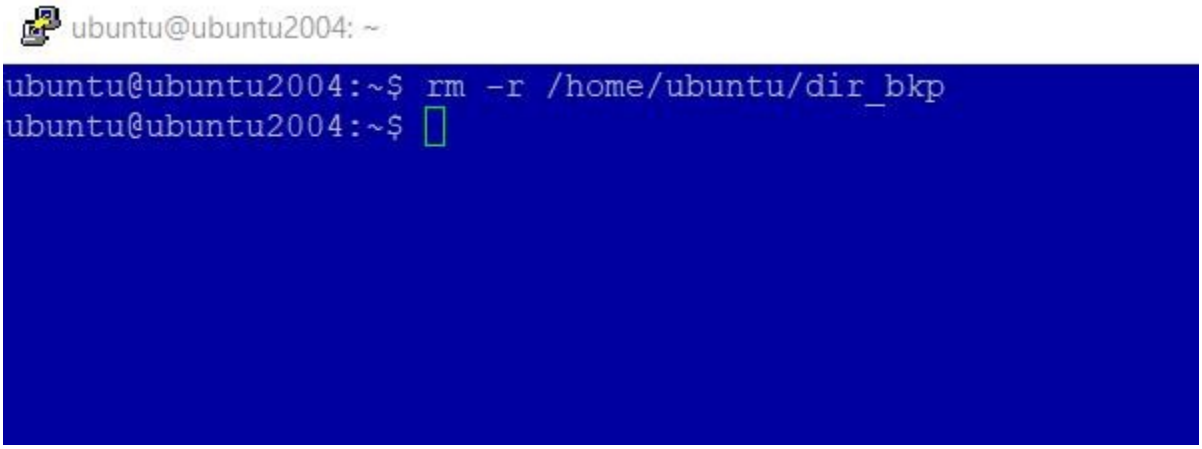
10. Matar o processo do MongoDB.

kill -9 62150



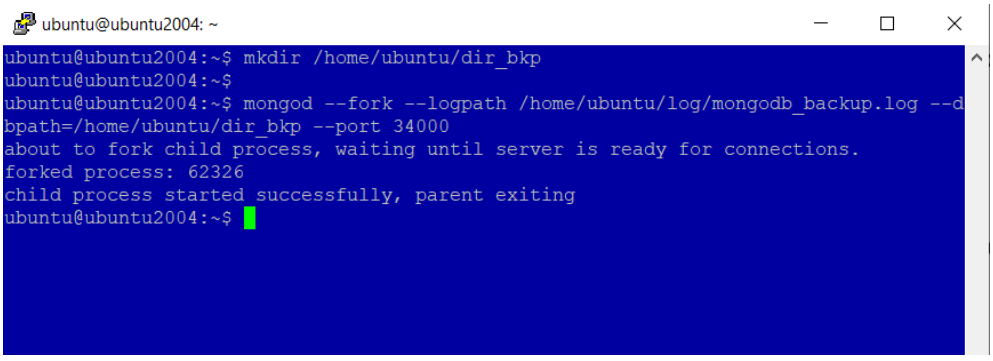
11. Apagar o diretório de dados do MongoDB.,

rm -r/home/ubuntu/dir_bkp

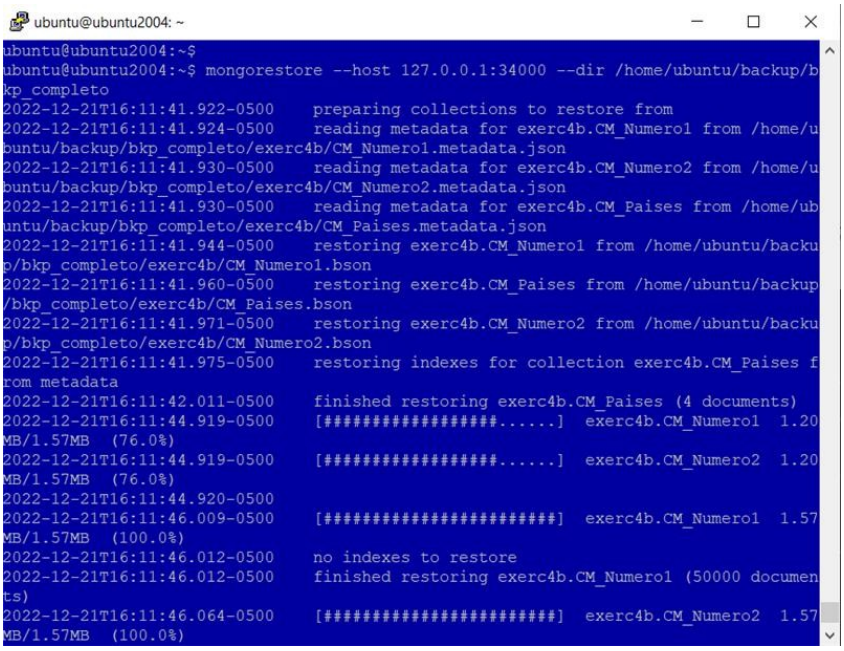


12. Fazer o restore do MongoDB (o caminho tem que ser criado anteriormente).

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

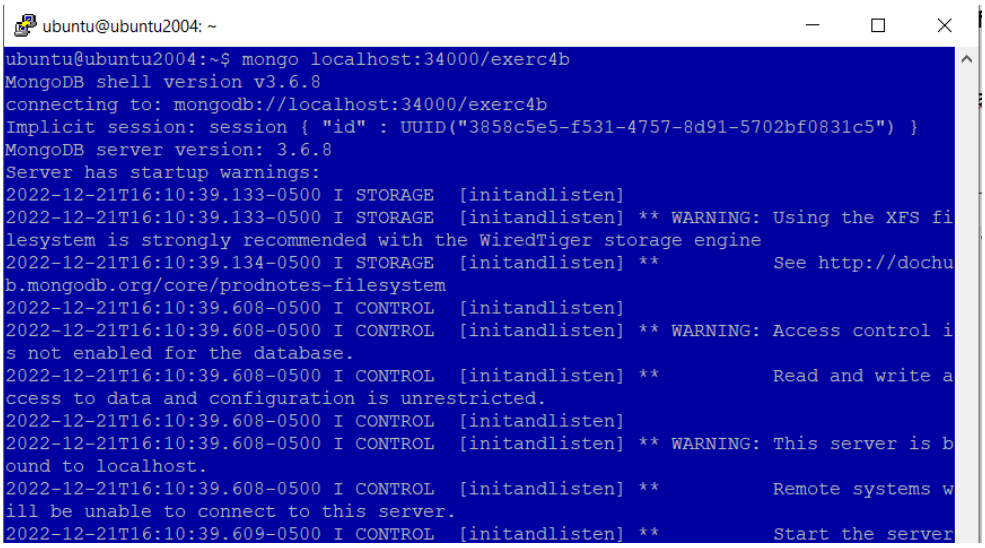


mongorestore --host 127.0.0.1:34000 --dir /home/ubuntu/backup/bkp_completo



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

mongo localhost:34000/exerc4b



db.CM_Numero2.count()

db.CM_Numero1.count()

db.CM_Paises.count()