MongoDB

Benilton Carvalho & Guilherme Ludwig

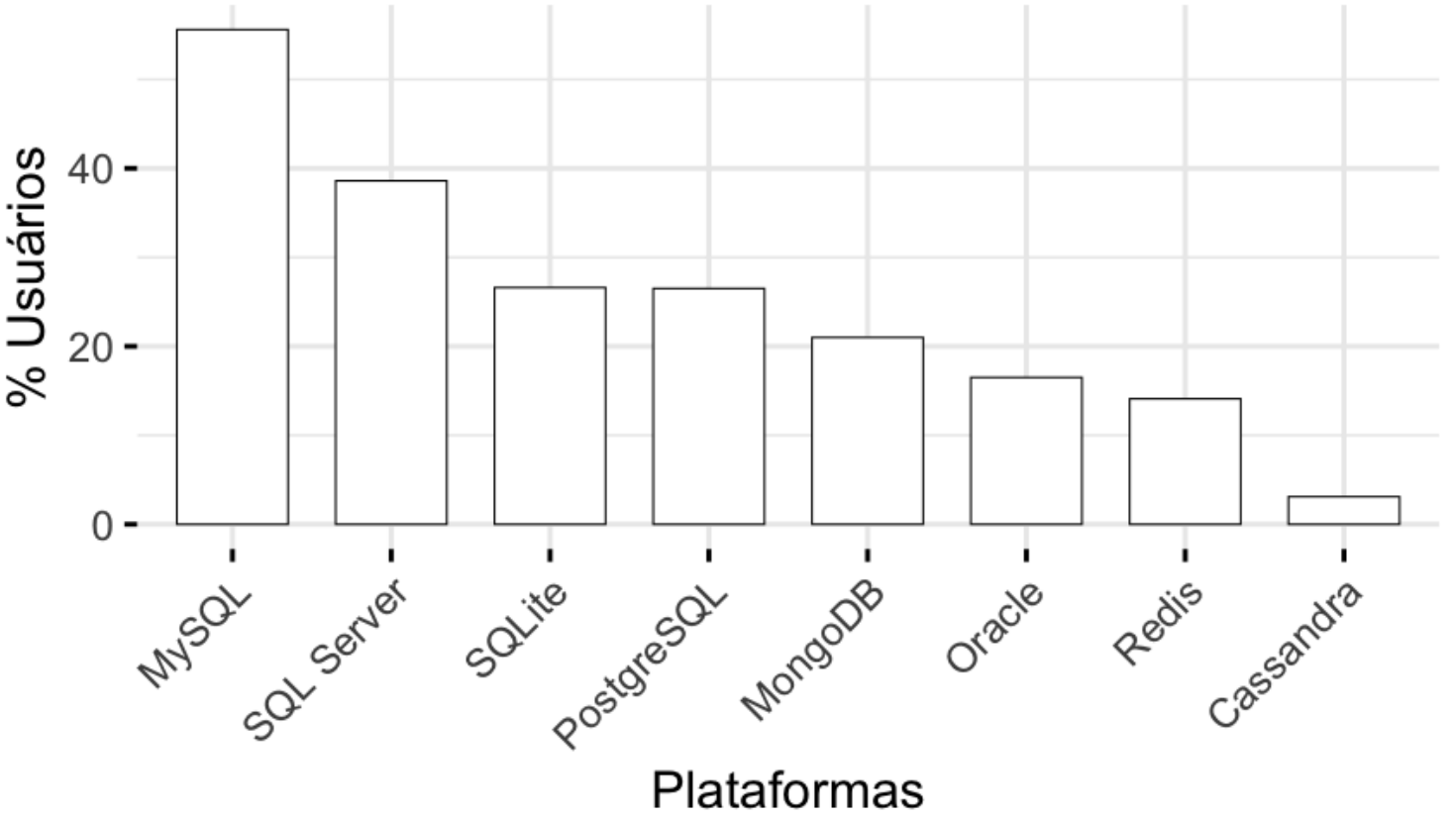
Introdução ao MongoDB

- Código aberto;
- Gratuito;
- Alta performance;
- Sem esquemas;
- Orientado a documentos;
- Implementado em C++;

Orientado a Documentos

- Orientado a documentos JSON;
- Lembrete:
  - documentos JSON possuem estrutura hierárquica;
  - podem ser facilmente utilizados pelo R ou outras ferramentas para realização de analítica;
  - suportam hierarquias complexas e mantém índices;

Popularidade no StackOverflow em 2017



Efeitos Práticos

- Cada documento é autossuficiente;
- Cada documento possui todas as informações de que possa precisar;
- Lembrete:
  - em SQL, evitam-se repetições e combinam-se tabelas via chaves;
- Evitam-se JOINs;
- Desenha-se a base de dados de forma que as *queries* busquem apenas uma chave e retornem todas as informações necessárias;
- Preço: espaço em disco;

Utilização de MongoDB

- Foco em *big data*;
- Escalonamento horizontal (*sharding*) - desempenho;
- Escalonamento vertical (*replica sets*) - multicore;
- Se os dados não possuem formato fixo, MongoDB é uma boa opção;
- (J/B)SON não possuem esquemas;
- Opção natural para sistemas web. Exemplo: Comércio eletrônico - detalhes de produtos;

Quando não utilizar MongoDB?

- Quando relacionamentos entre múltiplas entidades for essencial;
- Quando existirem múltiplas chaves externas e JOINs;
- Expectativas em MongoDB:
  - Documentos autossuficientes;
  - Mínimo de chaves;
  - etc;

Disponibilidade

- MongoDB Atlas - Database as a Service (AWS, GCP, Azure);
- Linux;
- MacOS;
- RHEL;
- Windows;

Uso do pacote `mongolite`

- Sempre monta-se uma conexão via `mongo()`;
- O arquivo pode ser remoto ou local;
- Contagem de registros via `con$count()`;
- Remoção de coleção via `con$drop()`;
- Inserção de coleção via `con$insert()`;

```
library(tibble)
library(mongolite)
myurl = "mongodb://readwrite:test@mongo.opencpu.org:43942/jeroen_test
con <- mongo("mtcars", url = myurl)
if(con$count() > 0) con$drop()
con$insert(mtcars)
```

```
## List of 5
##  $ nInserted  : num 32
##  $ nMatched   : num 0
##  $ nRemoved   : num 0
##  $ nUpserted  : num 0
##  $ writeErrors: list()
```

```
stopifnot(con$count() == nrow(mtcars))
```

Uso do pacote `mongolite`

- Seleção de dados presentes no banco de dados fia `con$find()`;
- No pacote `mongolite`, remover o objeto de conexão, `con`, já desconecta a sua sessão do banco de dados;
- Mas também existe o método `disconnect()` para realizar a desconexão;

```
mydata <- con$find()
stopifnot(all.equal(mydata, mtcars))
con$drop()
rm(con)
```

Coleções Maiores e Seleções mais Complexas

- Inserção de um conjunto de dados mais volumoso;

```
library(nycflights13)
## subconjunto pq o servidor eh publico
flights = flights[sample(nrow(flights), 10000), ]
m <- mongo(collection = "nycflights", url=myurl)
m$drop()
m$insert(flights)
```

```
## List of 5
##  $ nInserted  : num 10000
##  $ nMatched   : num 0
##  $ nRemoved   : num 0
##  $ nUpserted  : num 0
##  $ writeErrors: list()
```

Seleções mais Complexas

- `find()` é análogo ao `SELECT * FROM tabela`;
- É possível ordenar os dados já na seleção;
- As chamadas devem acontecer usando formato JSON;

```
m$count('{"month":1, "day":1}')
```

```
## [1] 19
```

```
jan1 <- m$find('{"month":1,"day":1}', sort='{"distance":-1}')
head(jan1) %>% as_tibble()
```

```
## # A tibble: 6 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1  2013     1     1     1720           1725        -5     2121
## 2  2013     1     1     1937           1905        32     2250
## 3  2013     1     1      628            630        -2     1016
## 4  2013     1     1     1059           1053         6     1342
## 5  2013     1     1     1730           1730         0     2126
## 6  2013     1     1     1959           2000        -1     2310
## # … with 12 more variables: sched_arr_time <int>, arr_delay <dbl>,
```

Ordenação em Grandes Bases

- Bases volumosas exigem a existência de um índice para permitir a ordenação;
- O índice pode ser adicionado via `index()`;
- `find()` aceita o argumento `sort=`.

```
# criacao de indice eh essencial para grandes volumes de dados
m$index(add = "distance")
```

```
##   v key._id key.distance       name                      ns
## 1 2       1           NA       _id_ jeroen_test.nycflights
## 2 2      NA            1 distance_1 jeroen_test.nycflights
```

```
allflights <- m$find(sort='{"distance":-1}')
```

Seleção de Colunas Específicas

- Utiliza-se `find()`;
- Adiciona-se o argumento `fields=`, que recebe a lista (em JSON) das variáveis de interesse;
- Ao especificar colunas de interesse, o MongoDB retorna uma coluna adicional, `_id`, que corresponde a um identificador interno do banco de dados;

```
# Select columns
jan1 <- m$find('{"month":1,"day":1}',
               fields = '{"_id":0, "distance":1, "carrier":1}')
head(jan1)
```

```
##   carrier distance
## 1      EV      266
## 2      UA     2227
## 3      DL     1096
## 4      UA      997
## 5      US      214
## 6      DL     1035
```

Identificação de Ocorrências Únicas

- O método `distinct()` retorna o que são valores únicos de um certo campo;
- Ele pode receber condições para serem avaliadas durante a execução;

```
# List unique values
m$distinct("carrier")
```

```
##  [1] "WN" "AA" "US" "9E" "EV" "MQ" "DL" "UA" "B6" "VX" "FL" "AS" "HA" "YV"
## [15] "OO" "F9"
```

```
m$distinct("carrier", '{"distance":{"$gt":3000}}')
```

```
## [1] "UA" "HA"
```

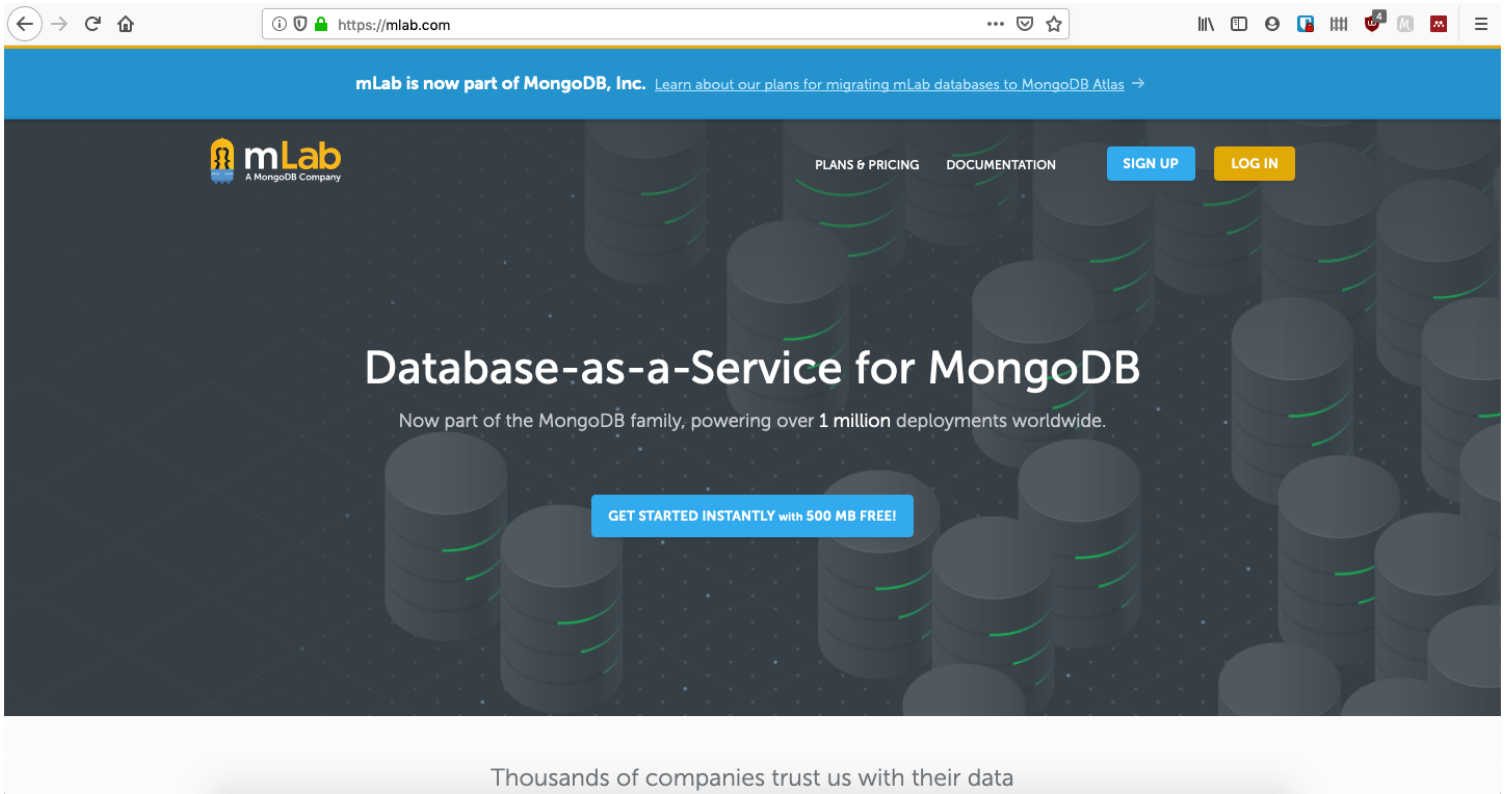
Tabulação de Dados em MongoDB

- O método `aggregate()` permite a tabulação de dados;
- Ele exige, em seu primeiro argumento, a apresentação de um *pipeline* para cálculos desejados;

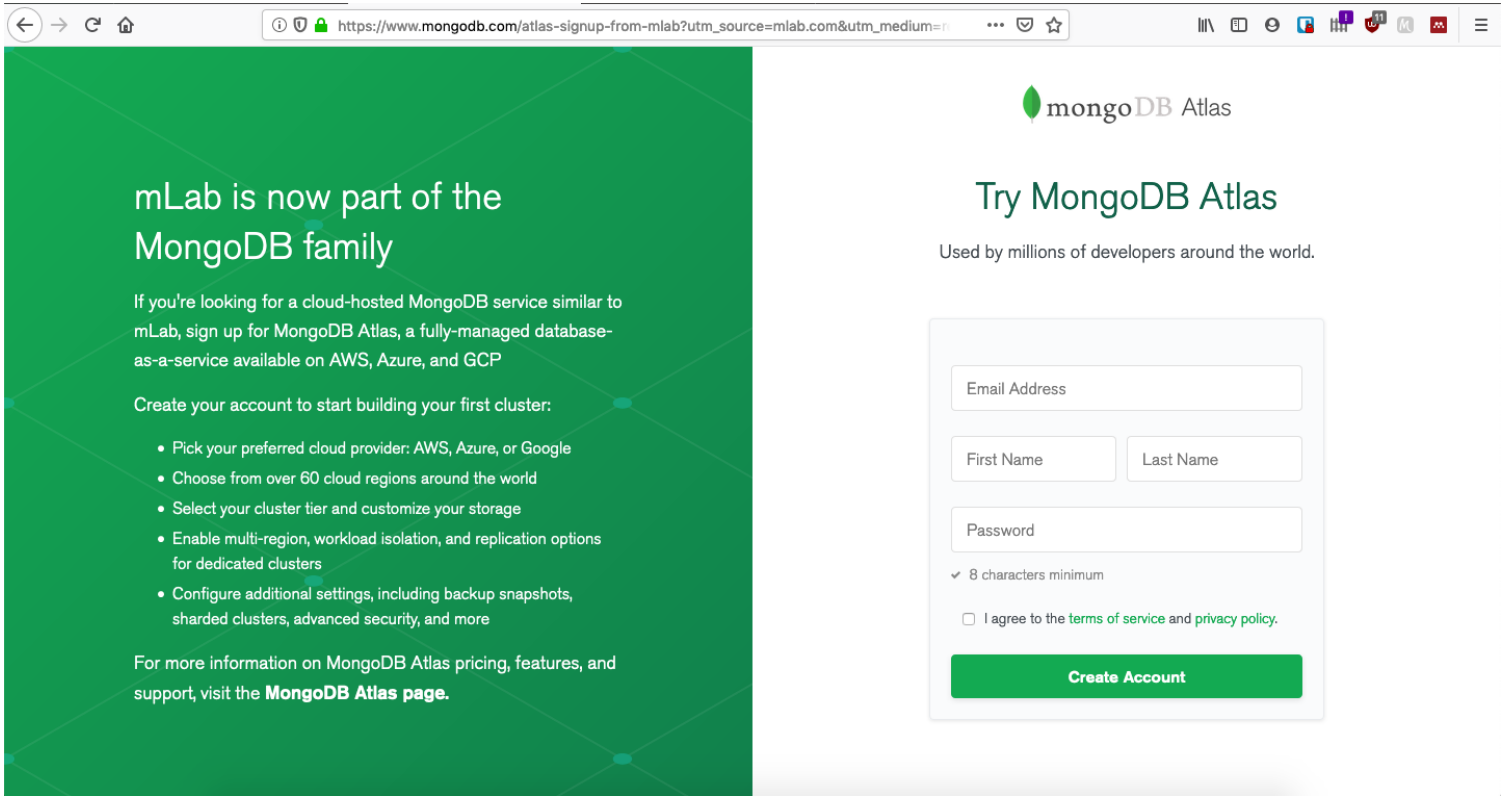
```
# Tabulate
m$aggregate('[{"$group": {"_id":"$carrier",
                          "count": {"$sum":1},
                          "average":{"$avg":"$distance"}}
            }]')
```

```
##    _id count   average
## 1   F9    14 1620.0000
## 2   OO     2  453.5000
## 3   AS    10 2402.0000
## 4   9E   536  551.2071
## 5   US   632  572.6487
## 6   HA     9 4983.0000
## 7   AA  1024 1362.7559
## 8   WN   359  995.4624
## 9   VX   149 2507.6510
## 10  YV    11  343.5455
## 11  FL   103  672.7087
```

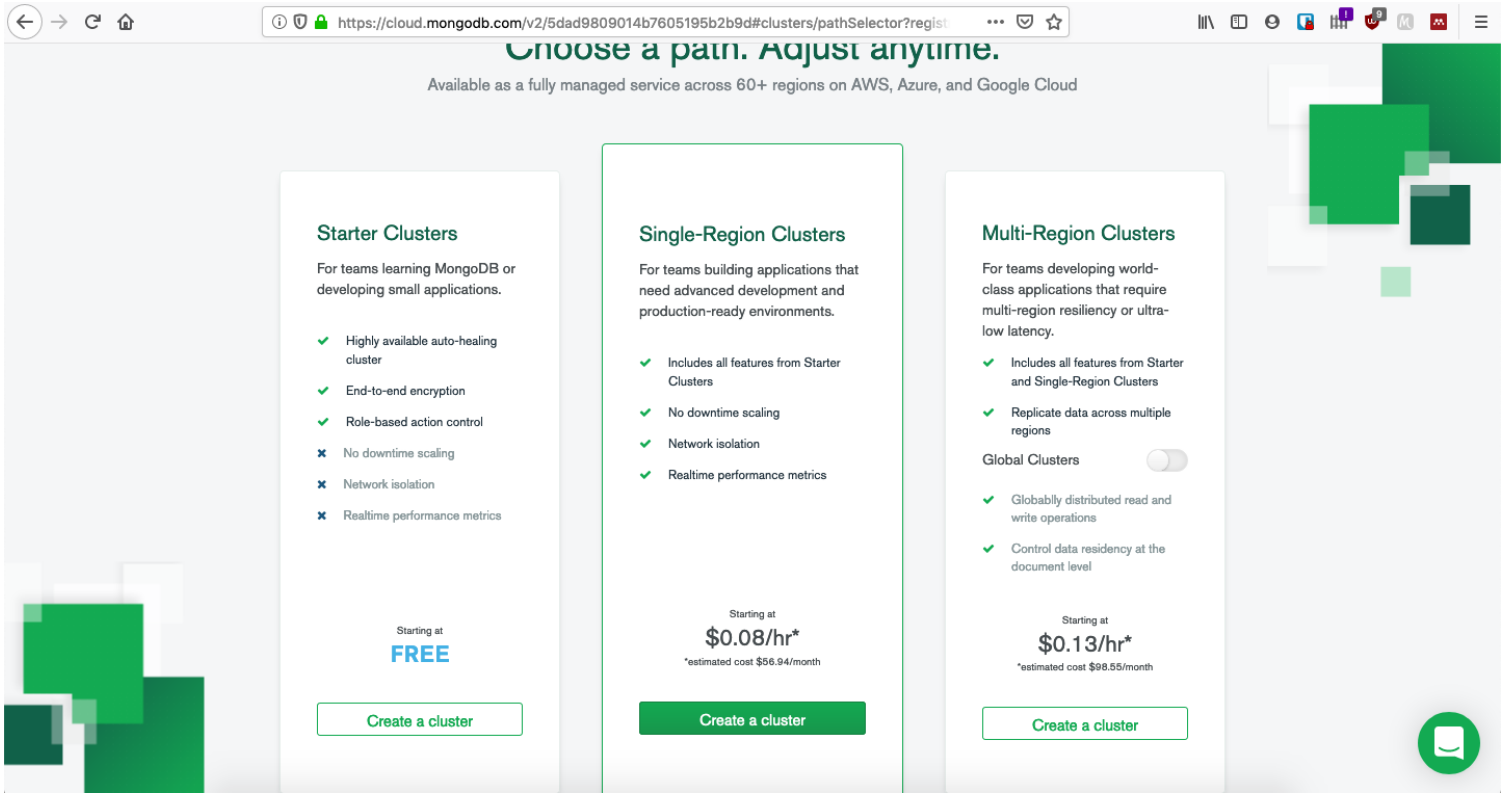
Criando sua instância para testes: mlab.com



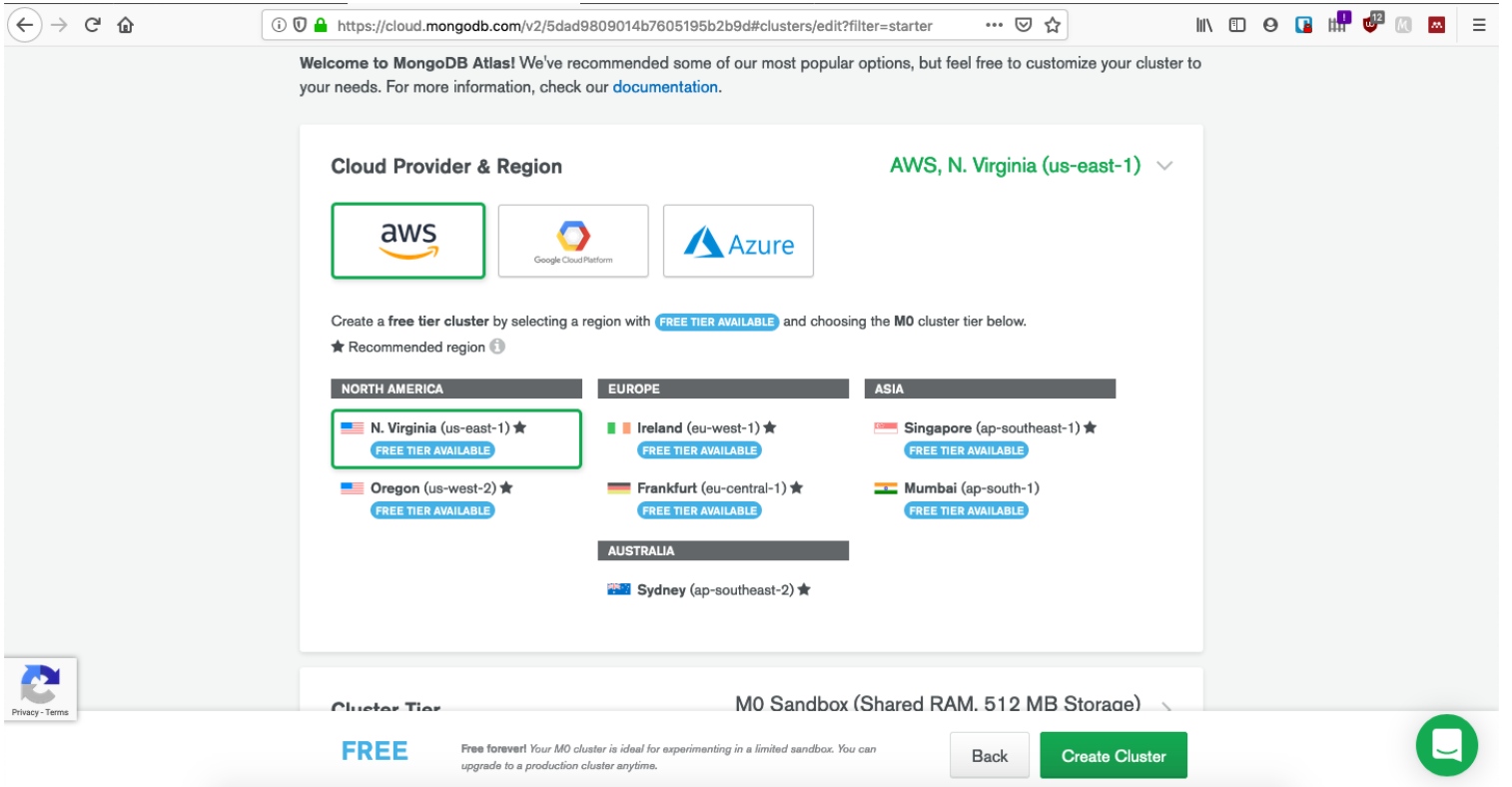
Criando sua instância para testes: CREATE ACCOUNT



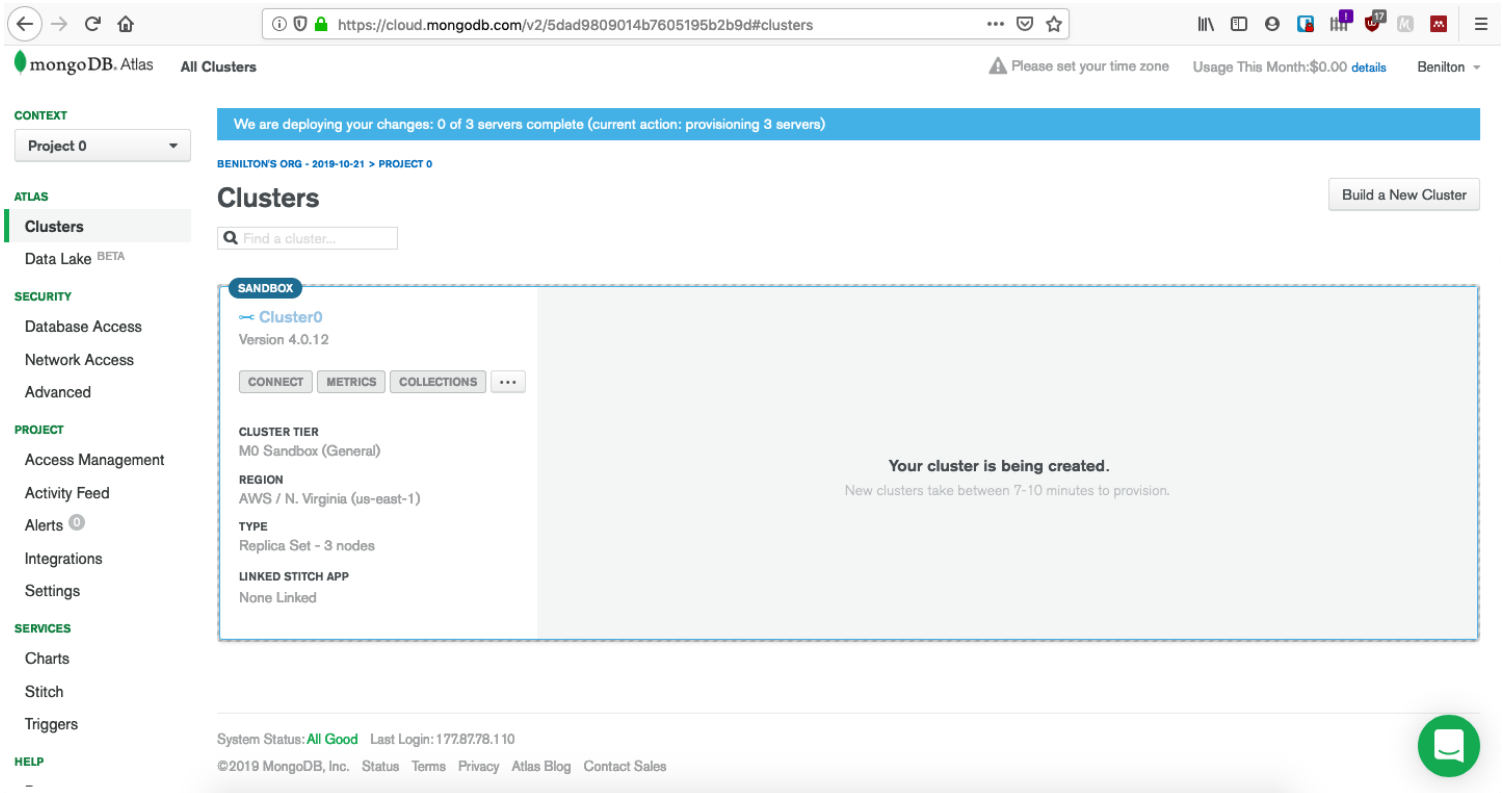
Criando sua instância para testes: FREE



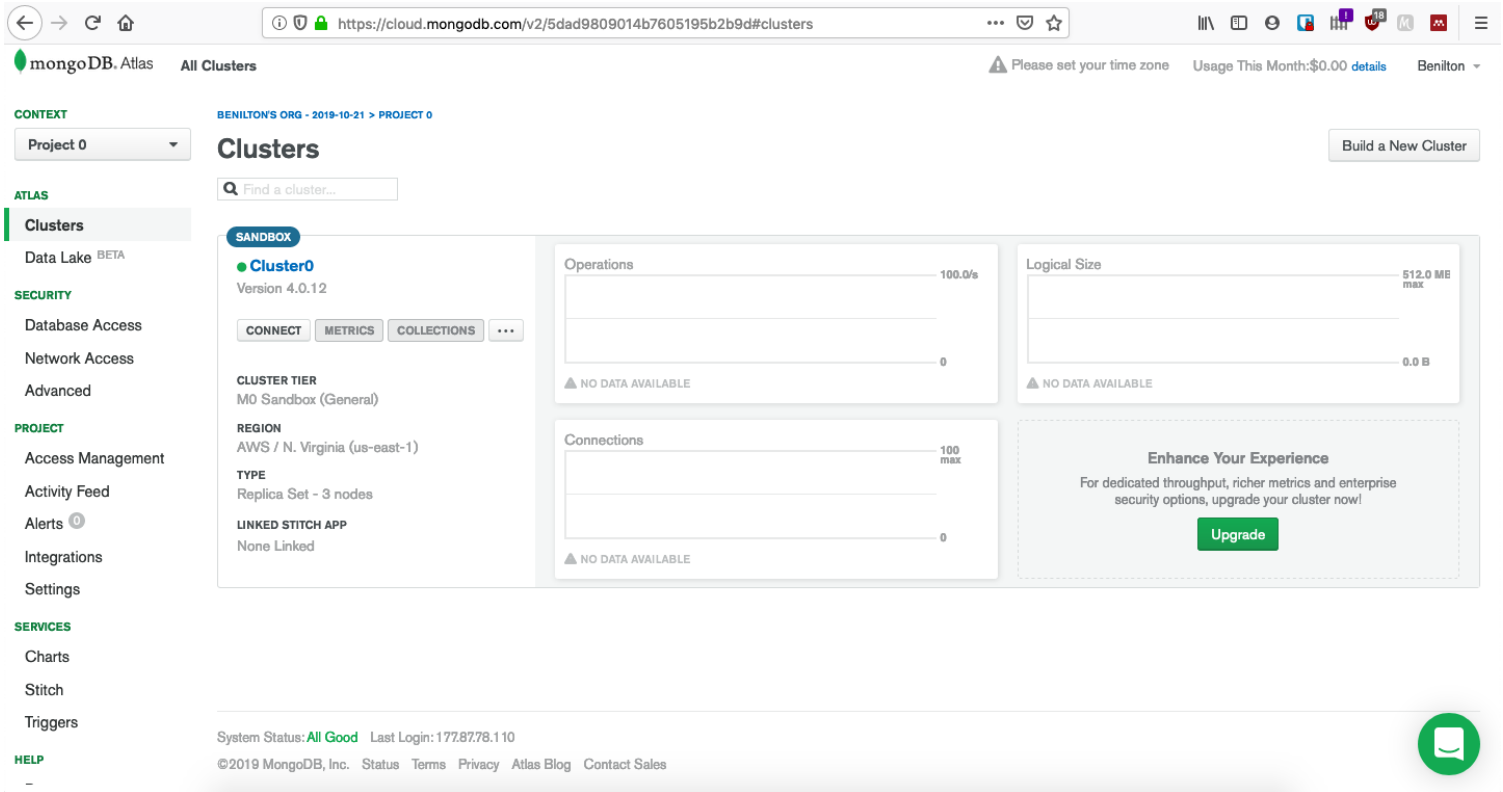
Criando sua instância para testes: CREATE



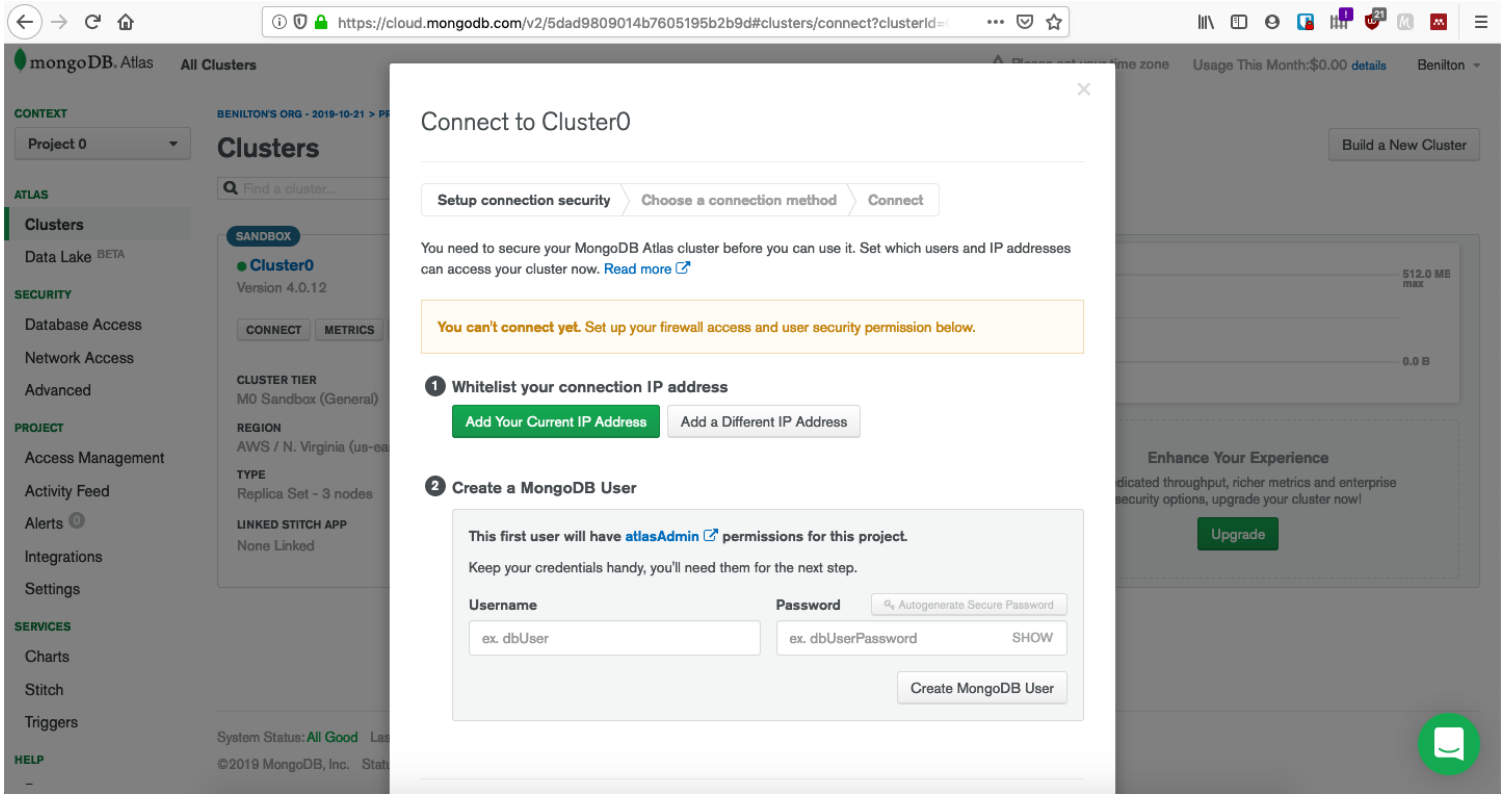
Criando sua instância para testes: Espere até criação



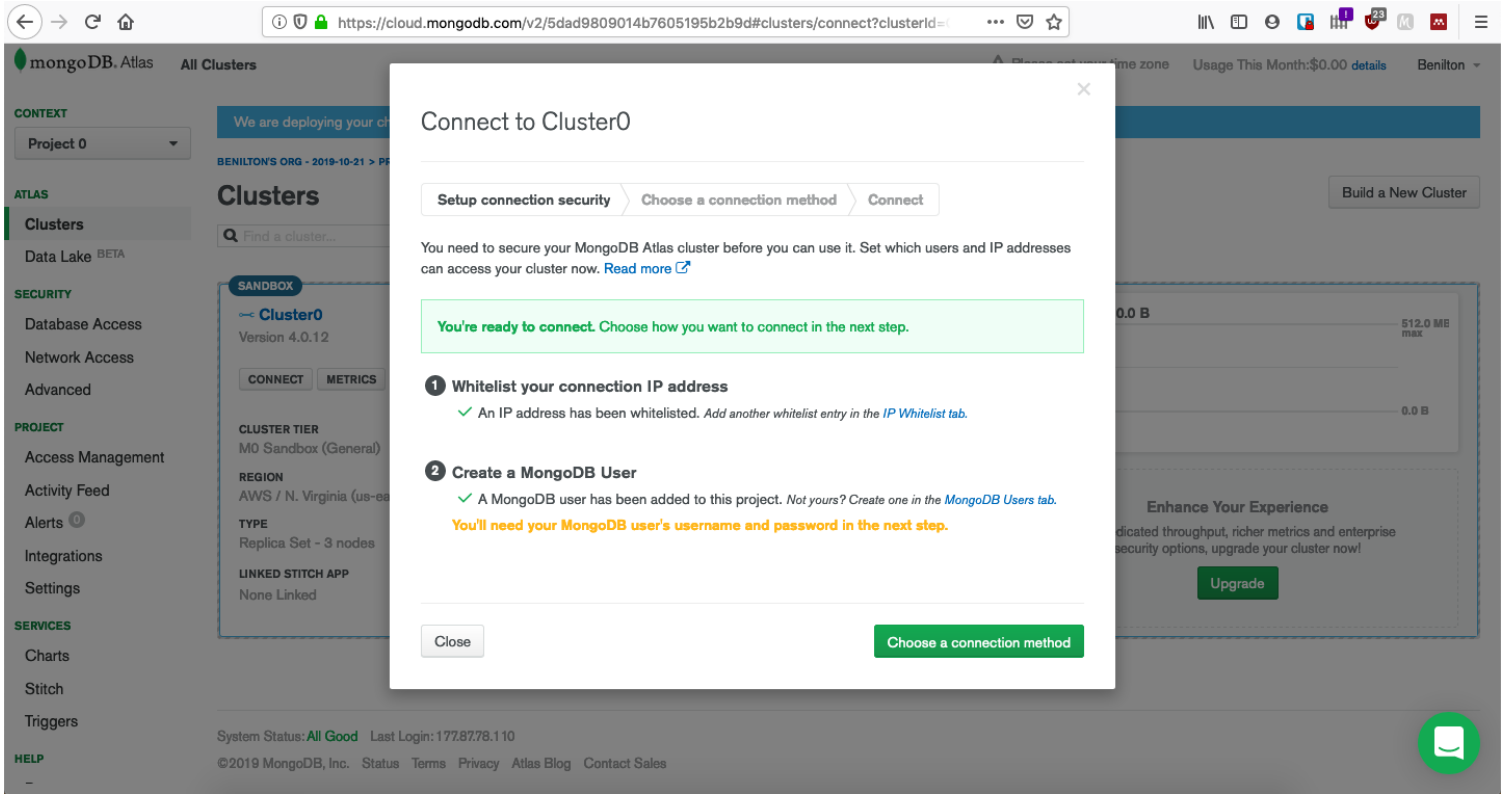
Criando sua instância para testes: Cluster pronto



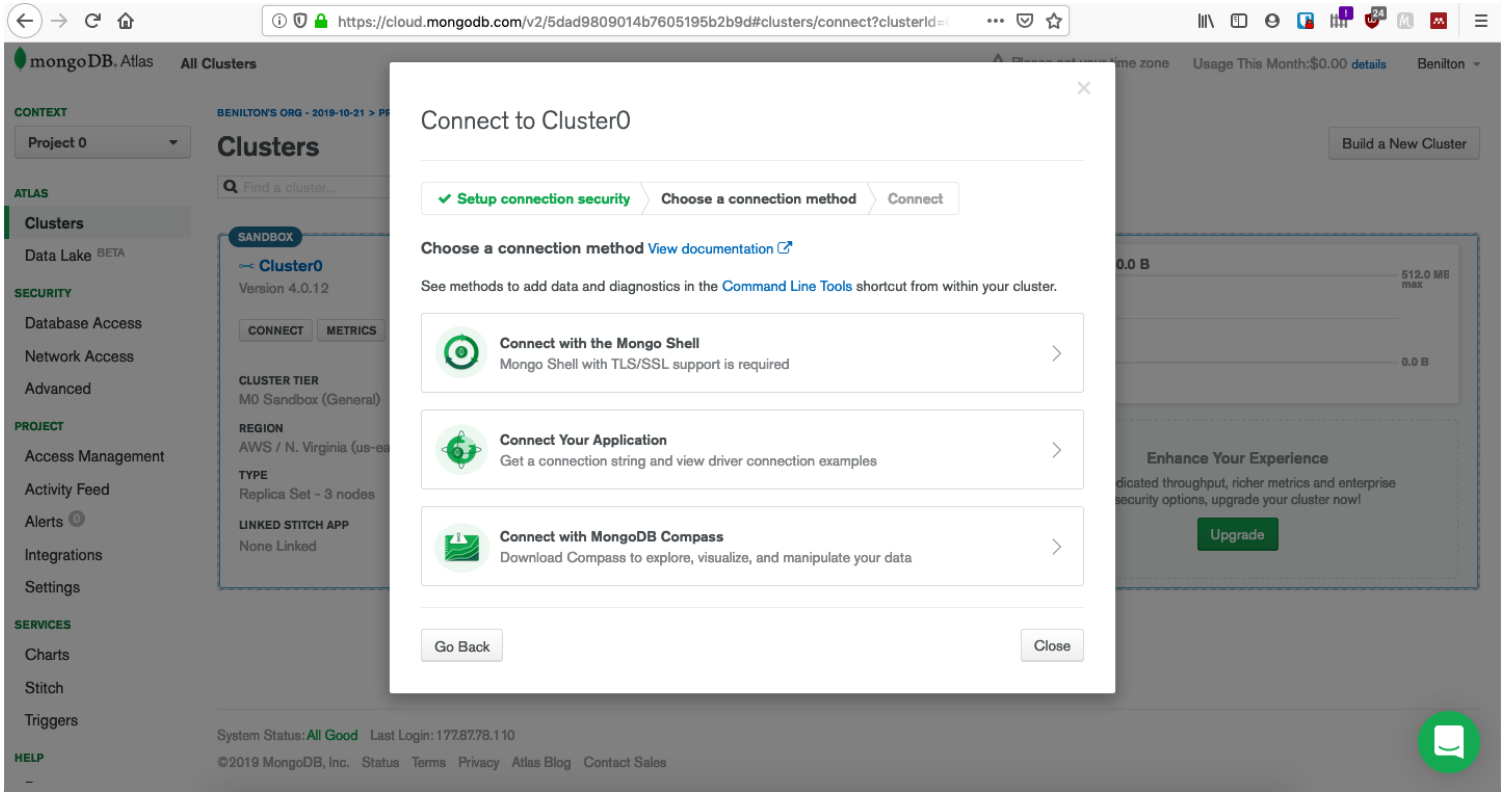
Criando sua instância para testes: Acesso



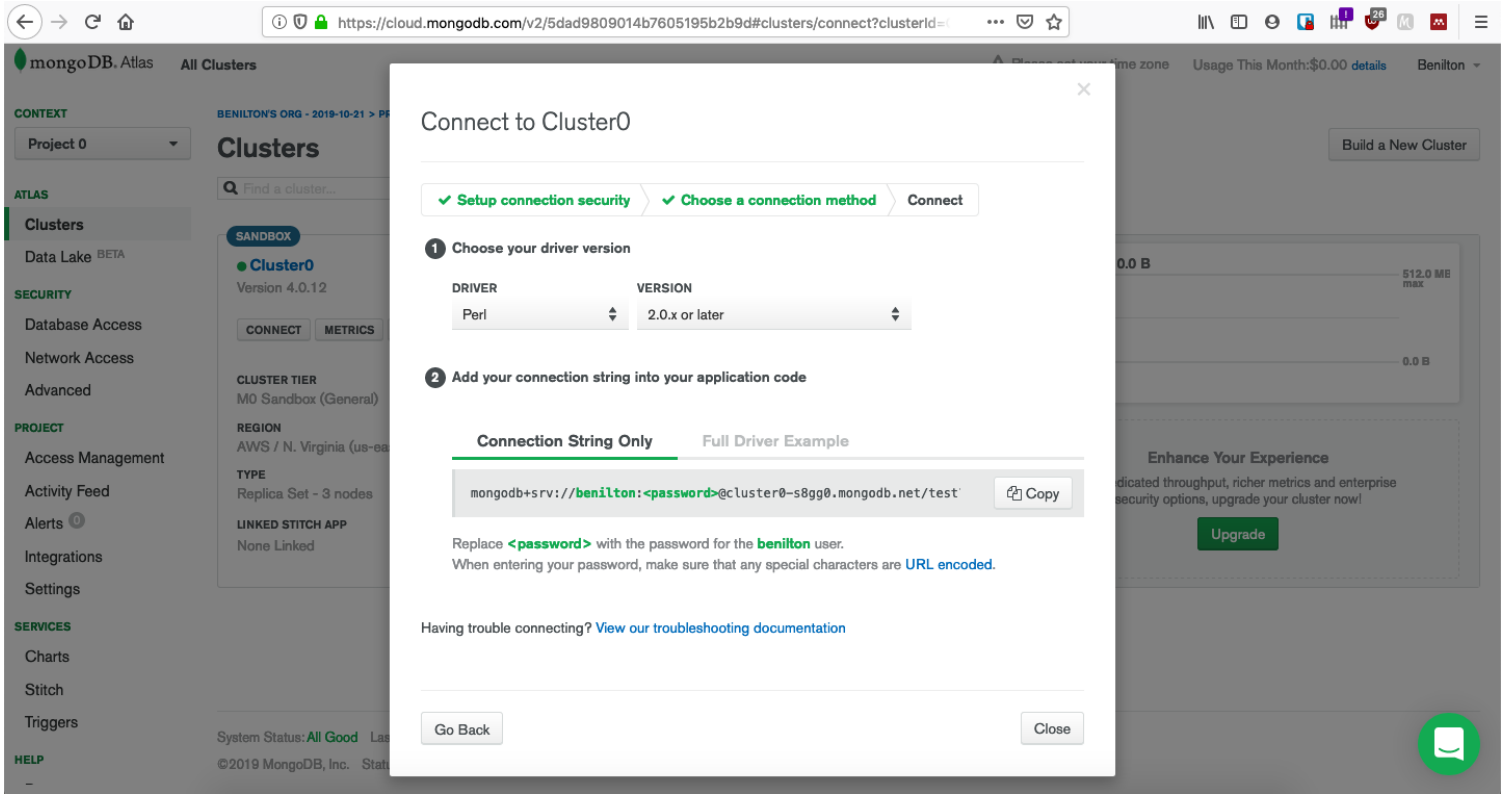
Criando sua instância para testes: Conexão



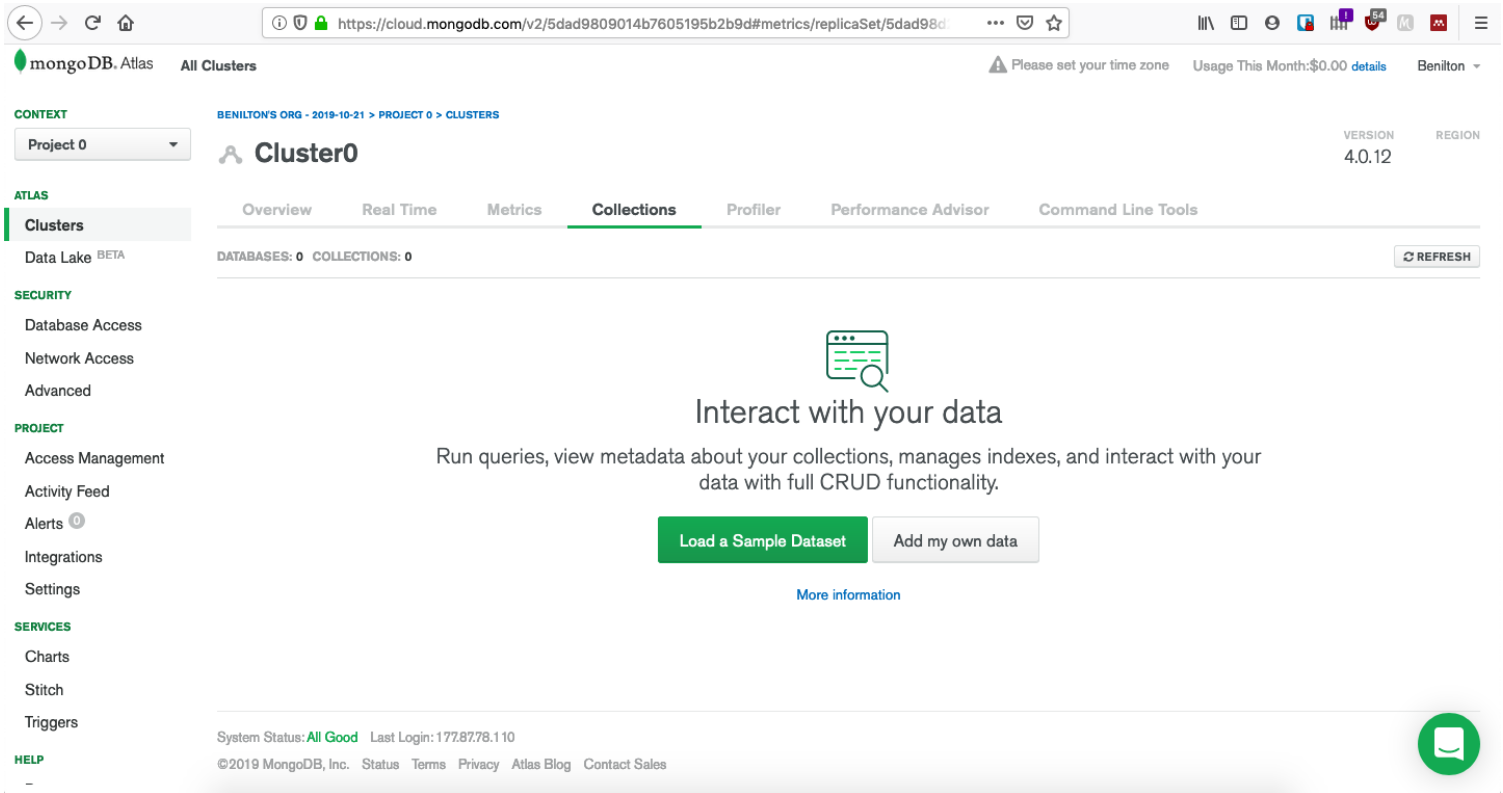
Criando sua instância para testes: Application



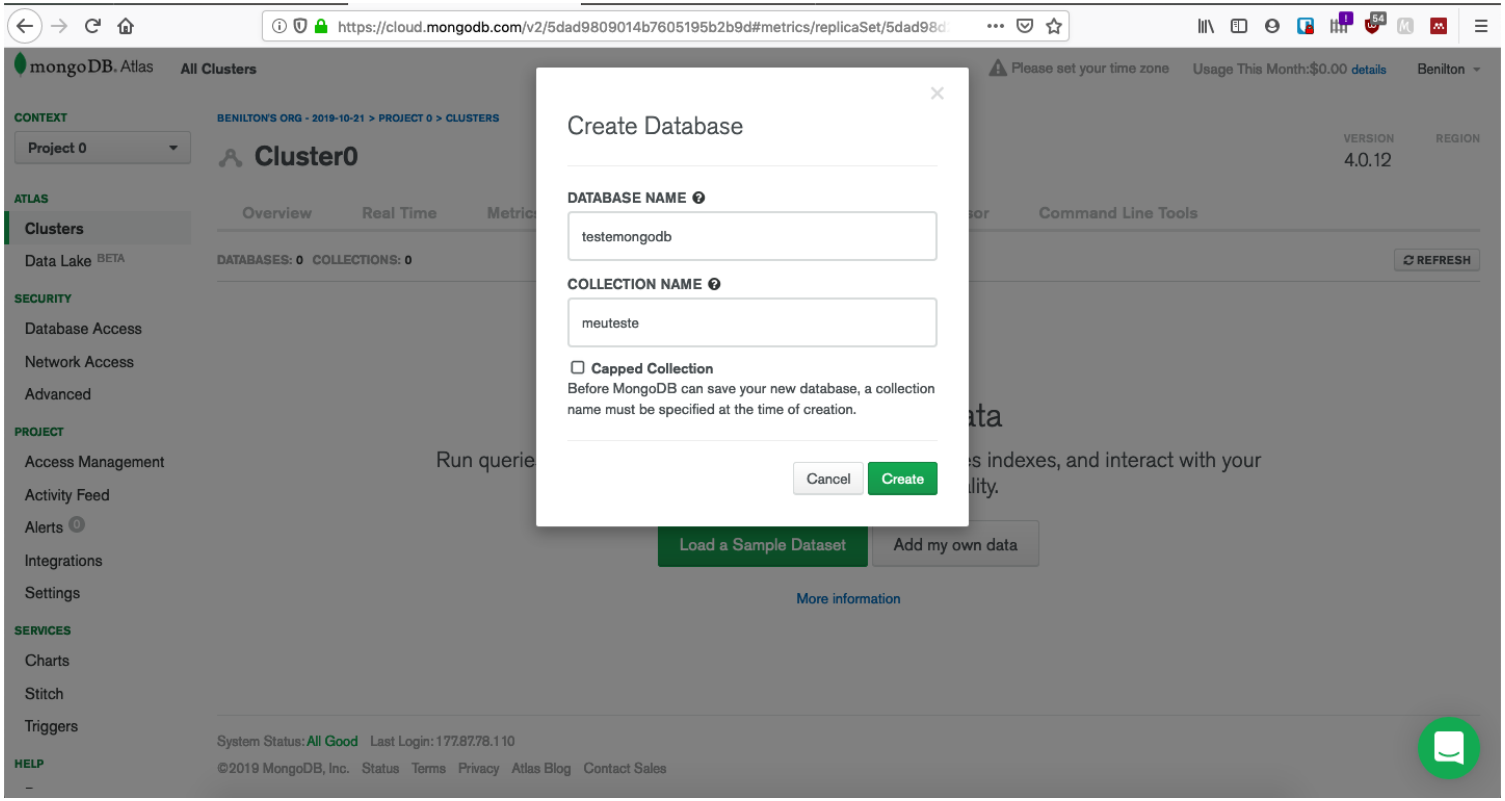
Criando sua instância para testes: Info para conexão



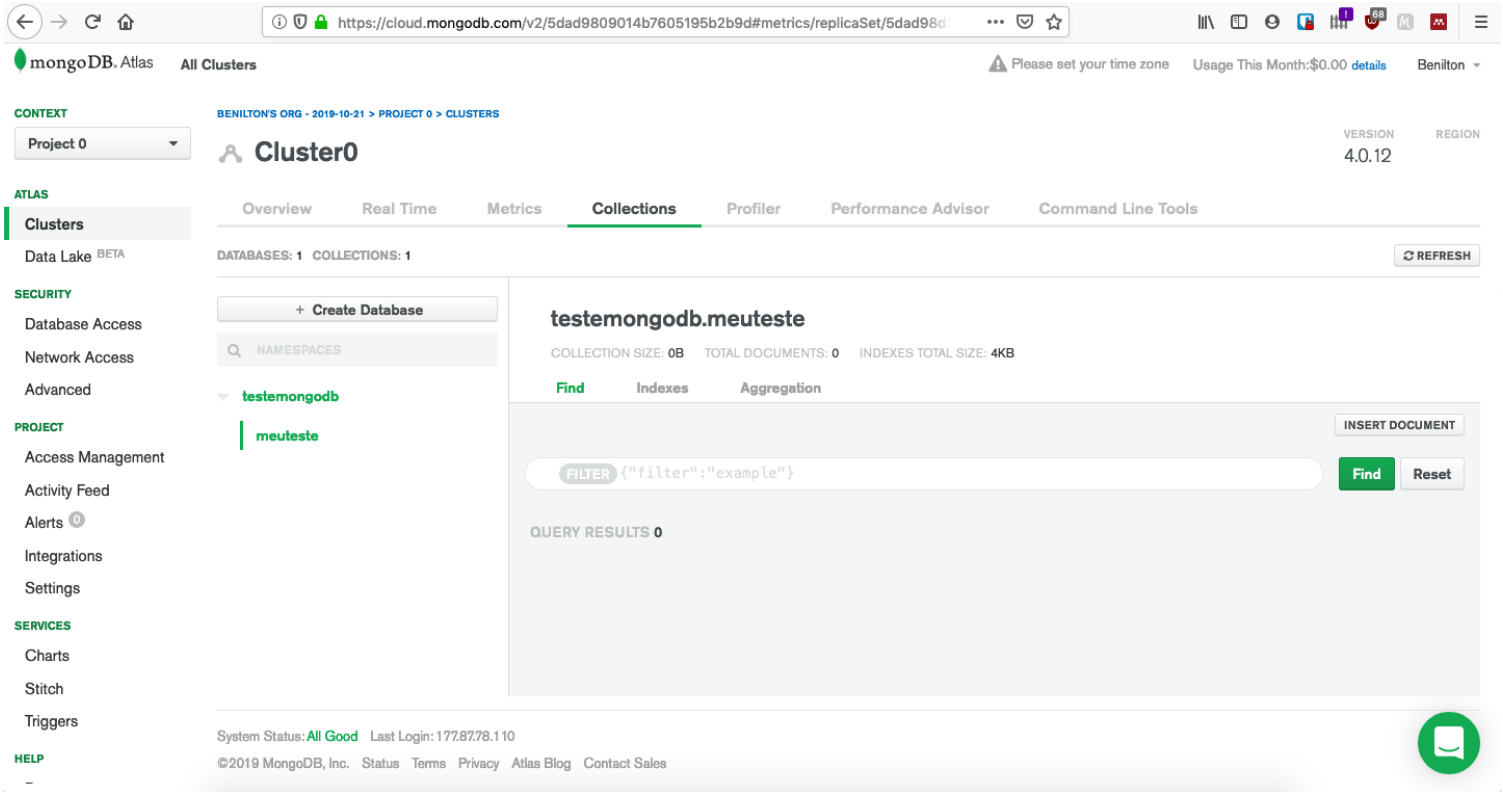
Criando sua instância para testes: ADD



Criando sua instância para testes: Banco e coleção



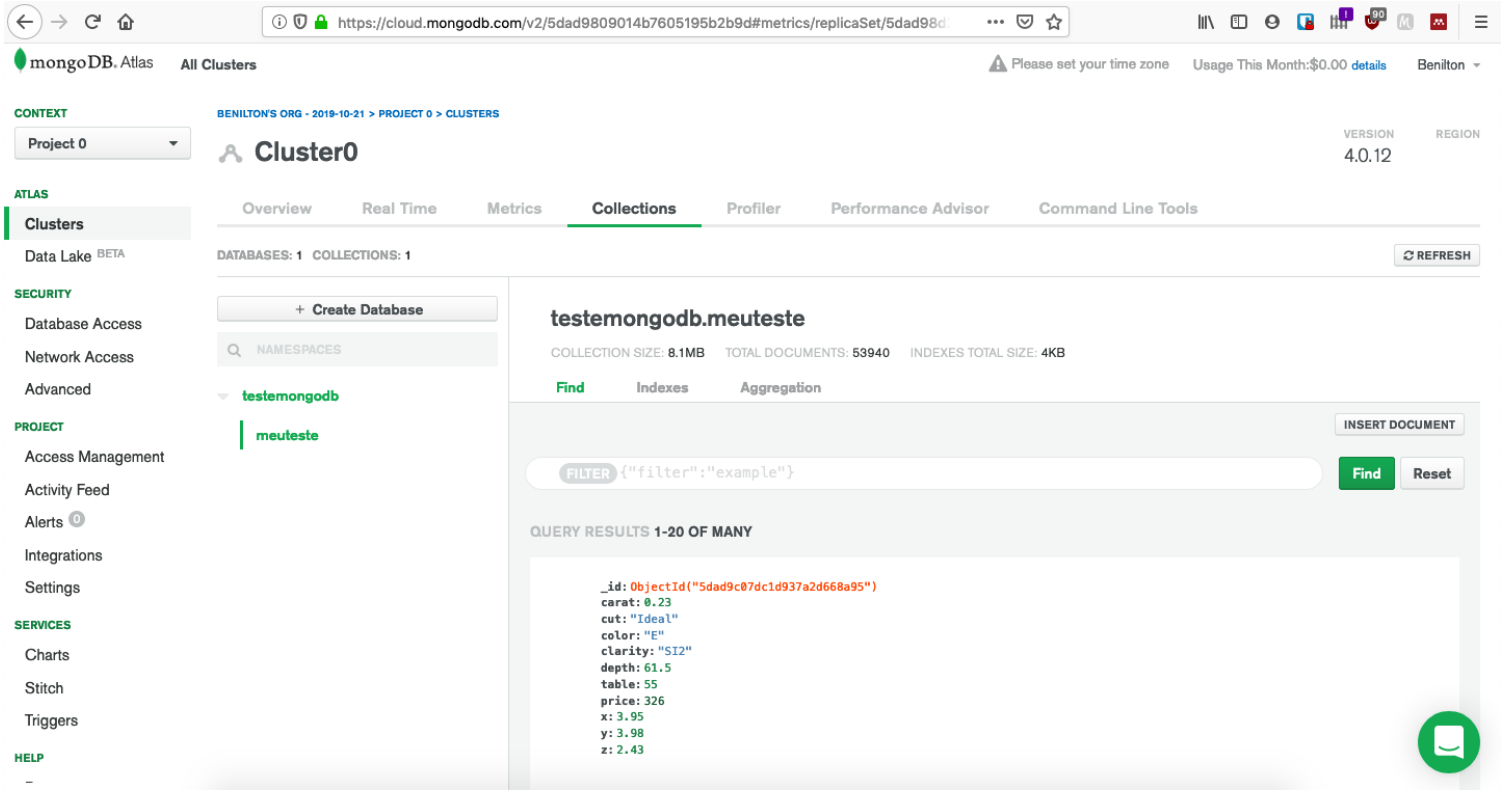
Criando sua instância para testes: Estrutura



Acesso via R

```
library(mongolite)
myurl = "mongodb+srv://benilton:minhasenha123@cluster0-s8gg0.mongodb.
myconn = mongo(collection="meuteste",
               db="testemongodb",
               url=myurl)
library(ggplot2)
myconn$insert(diamonds)
```

Criando sua instância para testes: Dados



Criando sua instância para testes: Manipulação Extra