

5.10 – Dados no MongoDB

Nesta etapa, trabalharemos com armazenamento, a quarta etapa do pipeline dos dados. O banco de dados que iremos utilizar para realizar essa tarefa será o MongoDB.

1º passo: Criar uma conta no mLab / Configurar

Inicialmente, precisamos criar uma conta no site <https://mlab.com/>



Try MongoDB Atlas

Used by millions of developers around the world.

☒ 8 characters minimum

☐ I agree to the [terms of service](#) and [privacy policy](#).

Create Account

Feito isso, prosseguiremos com a configuração.

Na tela seguinte, podemos escolher a opção gratuita e após isso, teremos que escolher uma Cloud Provider. Amazon, Google e Azure são as opções que temos disponíveis. Pode-se escolher qualquer uma delas. Nessa atividade, irei utilizar a Cloud da Amazon. A região fica a seu critério.

Welcome to MongoDB Atlas! We've recommended some of our most popular options, but feel free to customize your cluster to your needs. For more information, check our [documentation](#).

Cloud Provider & Region

AWS, N. Virginia (us-east-1) ▾



Create a **free tier cluster** by selecting a region with **FREE TIER AVAILABLE** and choosing the **M0** cluster tier below.

★ Recommended region ⓘ

NORTH AMERICA	EUROPE	ASIA
<div>🇺🇸 N. Virginia (us-east-1) ★ FREE TIER AVAILABLE</div> <div>🇺🇸 Oregon (us-west-2) ★ FREE TIER AVAILABLE</div>	<div>🇮🇪 Ireland (eu-west-1) ★ FREE TIER AVAILABLE</div> <div>🇩🇪 Frankfurt (eu-central-1) ★ FREE TIER AVAILABLE</div>	<div>🇸🇬 Singapore (ap-southeast-1) ★ FREE TIER AVAILABLE</div> <div>🇮🇳 Mumbai (ap-south-1) FREE TIER AVAILABLE</div>
AUSTRALIA		
<div>🇦🇺 Sydney (ap-southeast-2) ★ FREE TIER AVAILABLE</div>		

Clicando em “Create Cluster” na parte inferior, teremos concluído essa fase inicial de definição de ferramentas de trabalho. Em seguida, somos redirecionados para a interface demonstrada na figura abaixo, onde iniciaremos nosso trabalho no MongoDB:

The screenshot shows the MongoDB Atlas interface. On the left is a sidebar with navigation links: CONTEXT (Project 0), ATLAS (Clusters, Data Lake, SECURITY, PROJECT), SERVICES (Charts, Stitch, Triggers), and HELP (Get Started, Support). The main content area is titled 'Clusters' and shows details for 'Cluster0'. It includes a 'CONNECT' button, 'METRICS' and 'COLLECTIONS' tabs, and a summary of cluster details: M0 Sandbox (General), Region: AWS / N. Virginia (us-east-1), Type: Replica Set - 3 nodes, and Linked Stitch App: None Linked. There are three charts: Operations (R: 0, W: 0), Logical Size (0.0 B), and Connections (0). An 'Upgrade' button is visible in the bottom right corner of the main content area.

O próximo passo é criarmos nosso usuário e senha para acessar a cluster:



Connect to Cluster0

Setup connection security

Choose a connection method

Connect

You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. [Read more](#)

You can't connect yet. Set up your firewall access and user security permission below.

1 Whitelist your connection IP address

Add Your Current IP Address

Add a Different IP Address

2 Create a MongoDB User

This first user will have [atlasAdmin](#) permissions for this project.

Keep your credentials handy, you'll need them for the next step.

Username

hugo

Password

Autogenerate Secure Password

.....

SHOW

Create MongoDB User

Close

Choose a connection method

Escolher o método de conexão: "Connect your application". E em seguida, acessar e salvar a sua string para conexão.

Connect to Cluster0

✖ Setup connection security

✔ Choose a connection method

Connect

You can't connect yet. Set up your firewall access in the first step.

1 Choose your driver version

DRIVER

Node.js

VERSION

3.0 or later

2 Add your connection string into your application code

Connection String Only

Full Driver Example

```
mongodb+srv://hugo:<password>@cluster0-uln1j.mongodb.net/test?retryWr
```

Copy

Replace **<password>** with the password for the **hugo** user.

When entering your password, make sure that any special characters are [URL encoded](#).

Having trouble connecting? [View our troubleshooting documentation](#)

Go Back

Close

Agora vamos configurar no menu security o acesso via rede.



Connect to Cluster0

Setup connection security

Choose a connection method

Connect

You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. [Read more](#)

You can't connect yet. Set up your firewall access below.

1 Whitelist your connection IP address

IP Address	Description (Optional)
<input type="text" value="0.0.0.0"/>	<input type="text" value="An optional comment describing this entry"/>
<div>Cancel Add IP Address</div>	

Iremos utilizar o IP 0.0.0.0 na configuração. Porque assim, possibilitaremos o acesso ao banco de dados a partir de qualquer máquina. E assim finalizamos nosso 1º passo.

2º passo: Criar a conexão com o atlas

Iremos importar as bibliotecas pymongo e pandas para poder dar continuidade no projeto.

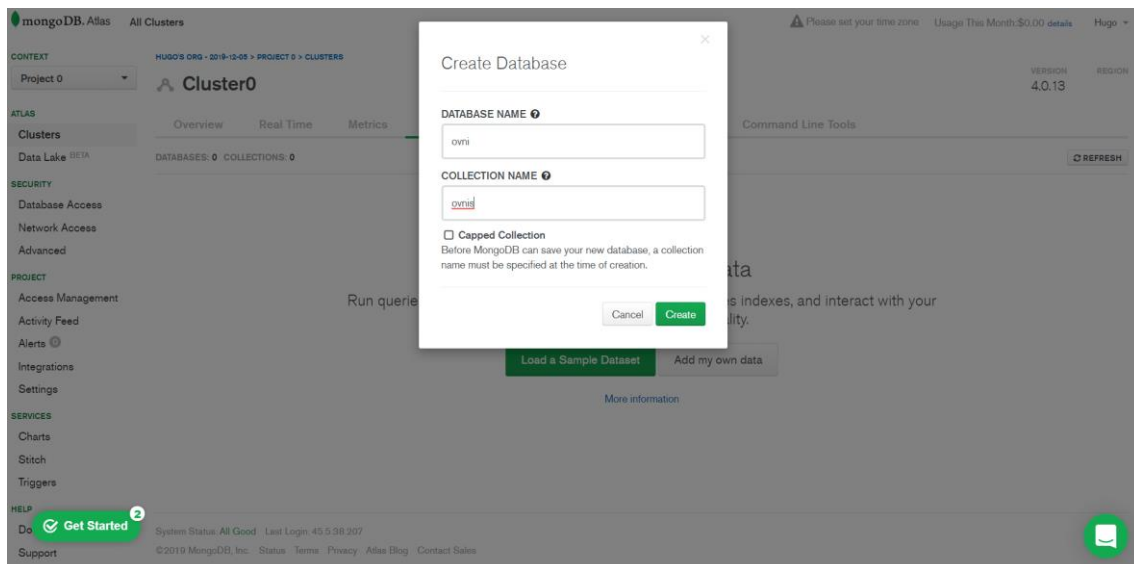
```
[1] 1 !pip install dnspython

[3] 1 import pymongo
    2 import pandas as pd

1 myclient = pymongo.MongoClient("mongodb+srv://hugo:senha123@cluster0-uln1j.mongodb.net/test?retryWrites=true&w=majori
2
3 db = myclient.ovni
4 collection = db.ovnis
```

Atribuímos à variável **myclient** a string de acesso que foi gerada pelo Atlas.

Logo após isso, na guia de “Collections” criaremos nossa base de dados e daremos nome à ela de **ovni** e a coleção de **ovnis**. Como mostra a figura a seguir:



Feito isso, iremos para o próximo passo.

3º passo: Inserir na coleção criada todos os registros do `df_OVNI_preparado`.

A primeira parte consiste em ler nosso data frame e convertê-lo em um json.

Para isso, precisaremos primeiramente importar a biblioteca **json** e logo após ler e converter o data frame. Como mostra a figura a seguir:

```
1 myclient = pymongo.MongoClient("mongodb+srv://hugo:senha123@cluster0-u1nj.mongodb.net/test?retryWrites=true&w=majority")
2
3 db = myclient.ovni
4 collection = db.ovnis
5
6 df = pd.read_csv('df_OVNI_preparado.csv')
7
8 df_json = df.to_json(orient='records')
9
10 json_ovni = json.loads(df_json)
```

Na variável **df** nós lemos o data frame e ao final do procedimento de conversão, temos a variável **json_ovni** com o nosso data frame convertido em json.

Agora iremos inserir nosso json no banco de dados.

```
1 inserir = collection.insert_many(json_ovni)
2 lista = list(collection.find({}))
```

Podemos notar que no nosso banco já aparecem os dados inseridos, como mostra a figura a seguir:

ovni.ovnis

COLLECTION SIZE: 0B TOTAL DOCUMENTS: 0 INDEXES TOTAL SIZE: 4KB

Find Indexes Aggregation

INSERT DOCUMENT

FILTER {"filter": "example"}

Find Reset

QUERY RESULTS 1-20 OF MANY

```
{
  "_id": ObjectId("5de876476b8f4ab661ae0078"),
  "Unnamed": 0,
  "City": "East Greenwich",
  "State": "RI",
  "Shape": "Disk",
  "Sight_Date": "29/01/97",
  "Sight_Time": "23:15",
  "Sight_Weekday": "Quarta-Feira",
  "Sight_Day": 29,
  "Sight_Month": 1
}
```

```
{
  "_id": ObjectId("5de876476b8f4ab661ae0079"),
  "Unnamed": 0,
  "City": "Flagstaff",
  "State": "AZ",
  "Shape": "Light",
  "Sight_Date": "26/01/97",
  "Sight_Time": "22:00",
  "Sight_Weekday": "Domingo",
  "Sight_Day": 26,
  "Sight_Month": 1
}
```

4º passo: Contar e mostrar quantos documentos há na coleção ovnis.

Nesse passo iremos contar e mostrar quantos documento há na coleção **ovnis**.

```
1 contar = collection.count()
```

```
80029
```

Utilizamos a função **count()** para poder contar e em seguida temos a saída esperada, como mostra a figura acima.

5º passo: Resgatar todos os documentos (registros) da coleção ovnis e ordenar por tipo (shape).

```
1 resgate = collection.find().sort("shape")
2 list(resgate)
```

Para resgatar e ordenar os documentos, utilizamos as funções **find()** e com a função **sort()** passamos o parâmetro desejado, que no nosso caso é o **shape**. E assim, obtemos o seguinte resultado:

```
[{'City': 'East Greenwich',
  'Shape': 'Disk',
  'Sight_Date': '29/01/97',
  'Sight_Day': 29,
  'Sight_Month': 1,
  'Sight_Time': '23:15',
  'Sight_Weekday': 'Quarta-Feira',
  'State': 'RI',
  'Unnamed: 0': 0,
  '_id': ObjectId('5de876476b8f4ab661ae0078')},
 {'City': 'Flagstaff',
  'Shape': 'Light',
  'Sight_Date': '26/01/97',
  'Sight_Day': 26,
  'Sight_Month': 1,
  'Sight_Time': '22:00',
  'Sight_Weekday': 'Domingo',
  'State': 'AZ',
  'Unnamed: 0': 1,
  '_id': ObjectId('5de876476b8f4ab661ae0079')},
 {'City': 'Marion',
  'Shape': 'Triangle',
  'Sight_Date': '25/01/97',
  'Sight_Day': 25,
  'Sight_Month': 1,
  'Sight_Time': '21:00',
  'Sight_Weekday': 'Sábado',
  'State': 'WI',
  'Unnamed: 0': 2,
  '_id': ObjectId('5de876476b8f4ab661ae007a')},
 {'City': 'Alta',
  'Shape': 'Other',
  'Sight_Date': '24/01/97',
  'Sight_Day': 24,
```

6º passo: Verificar quantas ocorrências existem por estado.

Nessa etapa iremos verificar quantas ocorrências existem por estado.

```
1 ocorrencia = list(collection.aggregate([{'$group': {'Views': {'$sum': 1},
2 '_id': '$State'}}], {'$sort': {'views': -1}}]))
```

E assim, obtemos o seguinte resultado:


```

[{'Views': 373, '_id': 'VT'},
 {'Views': 112, '_id': 'DC'},
 {'Views': 733, '_id': 'ID'},
 {'Views': 701, '_id': 'ME'},
 {'Views': 1075, '_id': 'CT'},
 {'Views': 1584, '_id': 'MA'},
 {'Views': 602, '_id': 'MT'},
 {'Views': 10463, '_id': 'CA'},
 {'Views': 416, '_id': 'NE'},
 {'Views': 1629, '_id': 'GA'},
 {'Views': 247, '_id': 'DE'},
 {'Views': 1400, '_id': 'TN'},
 {'Views': 3221, '_id': 'AZ'},
 {'Views': 238, '_id': 'WY'},
 {'Views': 894, '_id': 'UT'},
 {'Views': 985, '_id': 'KY'},
 {'Views': 2699, '_id': 'OH'},
 {'Views': 368, '_id': 'RI'},
 {'Views': 5077, '_id': 'FL'},
 {'Views': 693, '_id': 'AR'},
 {'Views': 997, '_id': 'NV'},
 {'Views': 1287, '_id': 'MN'},
 {'Views': 1520, '_id': 'WI'},
 {'Views': 2289, '_id': 'NC'},
 {'Views': 3690, '_id': 'TX'},
 {'Views': 2837, '_id': 'IL'},
 {'Views': 4338, '_id': 'WA'},
 {'Views': 1429, '_id': 'SC'},
 {'Views': 840, '_id': 'OK'},
 {'Views': 669, '_id': 'NH'},
 {'Views': 3492, '_id': 'NY'},
 {'Views': 145, '_id': 'ND'},
 {'Views': 4700, '_id': 'ND'}]

```

7º passo: Buscar todas as ocorrências da cidade Phoenix.

Agora iremos buscar as ocorrências da cidade de Phonex.



```
1 phoenix = list(collection.find({'City' : 'Phoenix'}))
```

Desta forma podemos obter as ocorrências desejadas, na figura abaixo temos o resultado obtido:

```

▶ [
  {
    'City': 'Phoenix',
    'Shape': 'Chevron',
    'Sight_Date': '01/01/97',
    'Sight_Day': 1,
    'Sight_Month': 1,
    'Sight_Time': '19:30',
    'Sight_Weekday': 'Quarta-Feira',
    'State': 'AZ',
    'Unnamed: 0': 36,
    '_id': ObjectId('5de876476b8f4ab661ae009c')
  },
  {
    'City': 'Phoenix',
    'Shape': 'Cigar',
    'Sight_Date': '11/02/97',
    'Sight_Day': 11,
    'Sight_Month': 2,
    'Sight_Time': '11:00',
    'Sight_Weekday': 'Domingo',
    'State': 'AZ',
    'Unnamed: 0': 57,
    '_id': ObjectId('5de876476b8f4ab661ae00b1')
  },
  {
    'City': 'Phoenix',
    'Shape': 'Other',
    'Sight_Date': '20/03/97',
    'Sight_Day': 20,
    'Sight_Month': 3,
    'Sight_Time': '00:00',
    'Sight_Weekday': 'Quinta-Feira',
    'State': 'AZ',
    'Unnamed: 0': 99,
    '_id': ObjectId('5de876476b8f4ab661ae00db')
  },
  {
    'City': 'Phoenix',
    'Shape': 'Light',
    'Sight_Date': '17/03/97',
    'Sight_Day': 17,

```

8º passo: Buscar as ocorrências do estado da Califórnia e ocultar o id de cada documento (registro)

Chegamos no nosso 8º e último passo dessa atividade, agora nós iremos buscar as ocorrências do estado da Califórnia e ocultar o id de cada documento (registro).

```

▶ 1 cali = list(collection.find({'City' : 'Phoenix'}, {'_id': 0}))

```

Da mesma forma do passo anterior podemos fazer a busca de ocorrências do estado, a única diferença é que agora podemos utilizar o número 0 como parâmetro para ocultar o ID. E assim finalizamos nossa atividade.

GITHUB

<https://github.com/HugoCalisto/datascience/>