**Lab 8  Understanding MongoDB Atlas (V2)**

**Task 1: Setting up the MongoDB database**

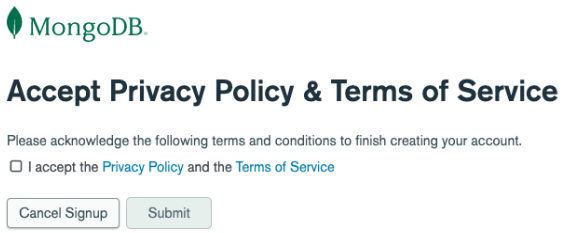
## 1.1 Create and Deploy an Atlas Cluster

You will first need to create an account with MongoDB Atlas (this is free, and just requires that you enter basic contact details and acknowledge their terms of service).

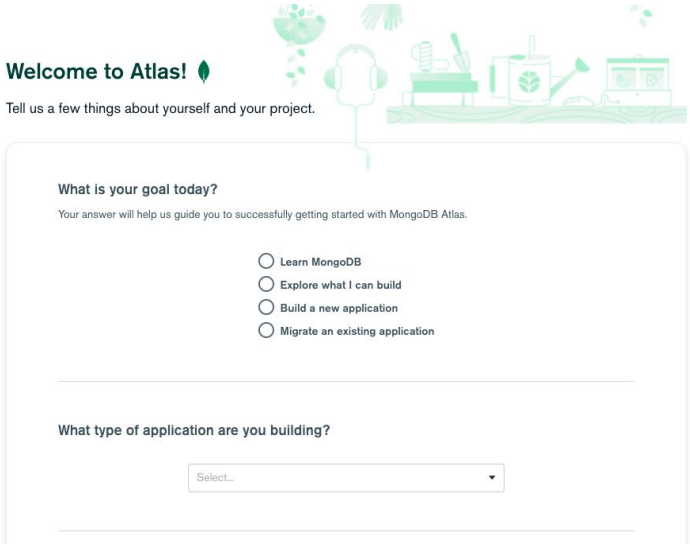
- https://www.mongodb.com/cloud/atlas/register

After logging in, you'll be taken to the home screen via https://cloud.mongodb.com/v2

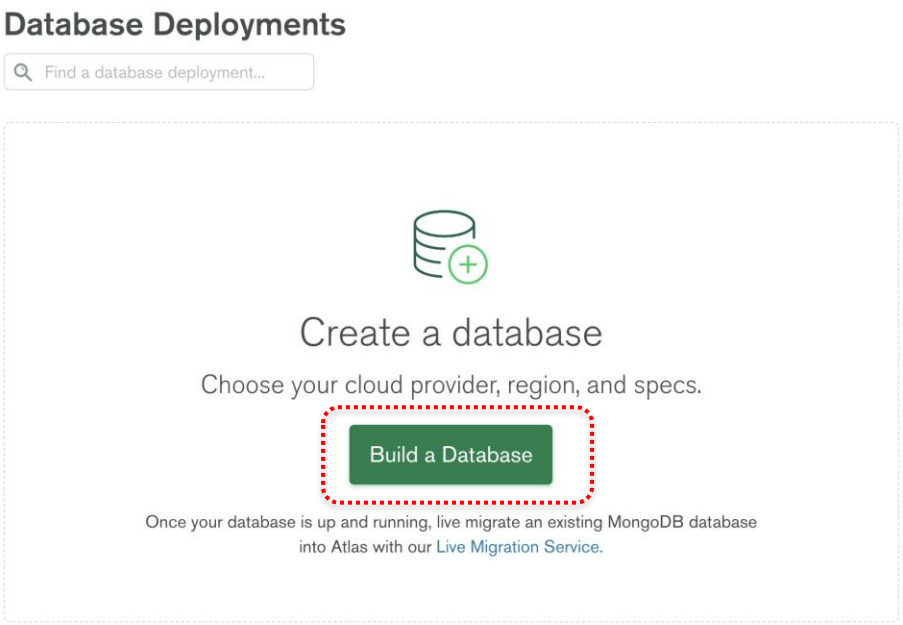
- Accept privacy policy and the terms of service and click the Submit button.




- Fill in data based on your interest on MongoDB

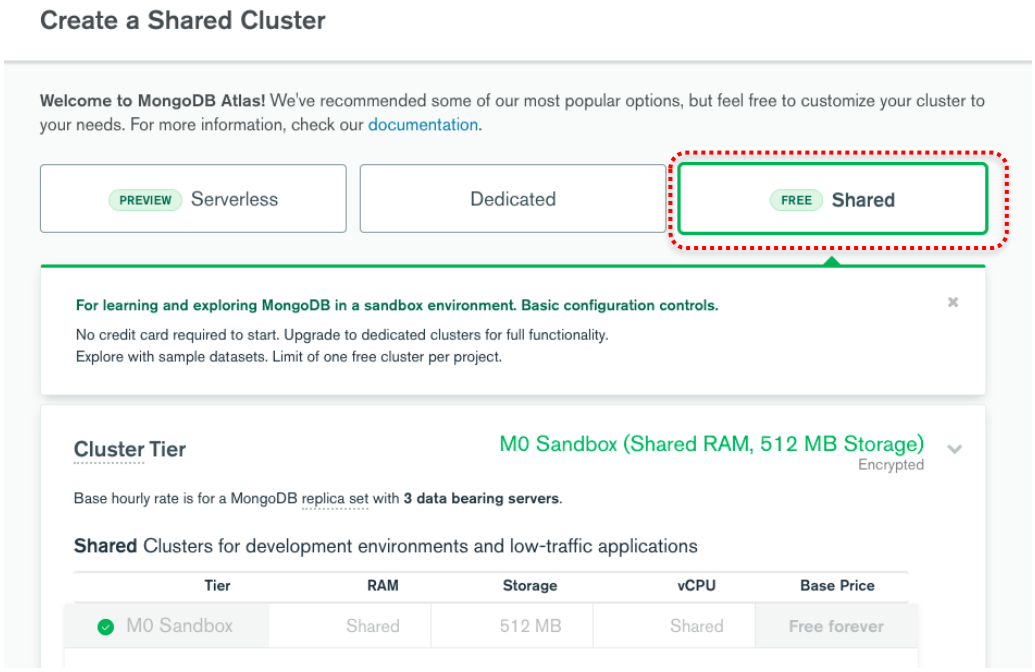



- Click **Build a Database** button in the Overview section.

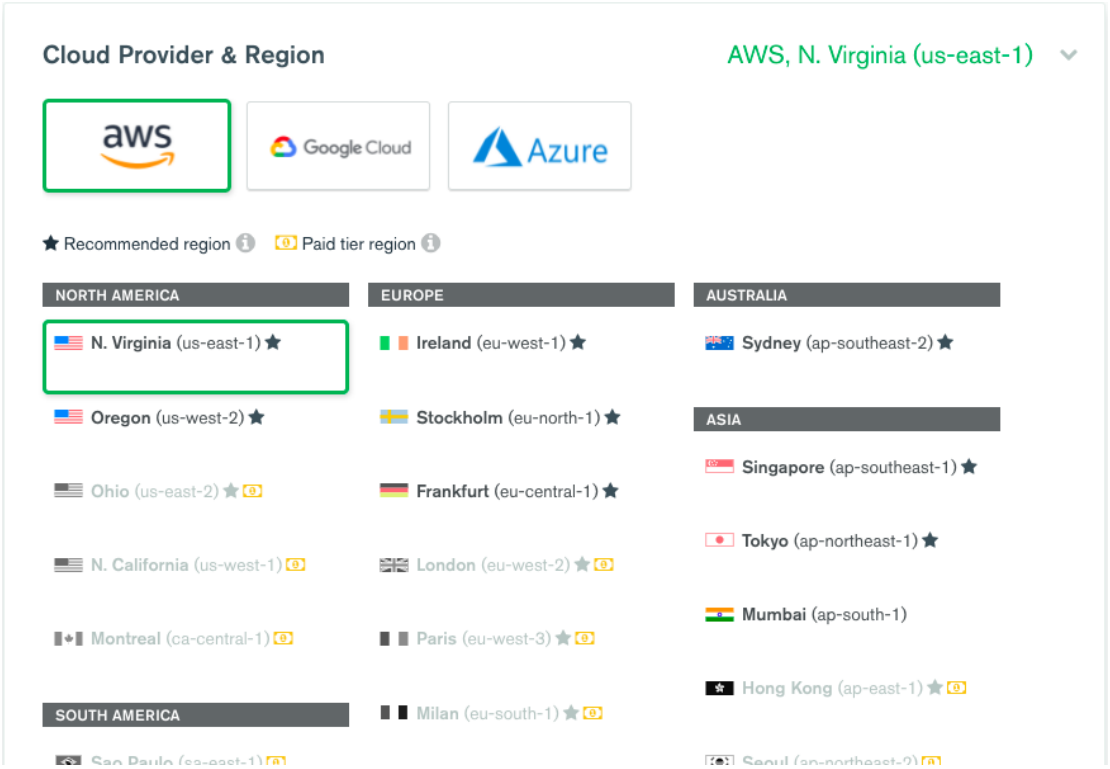



**Note:** If you cannot see the above button, you may need to create an organization first (specify an organization name as your first name) and create a new project (specify a project name as “project0”).

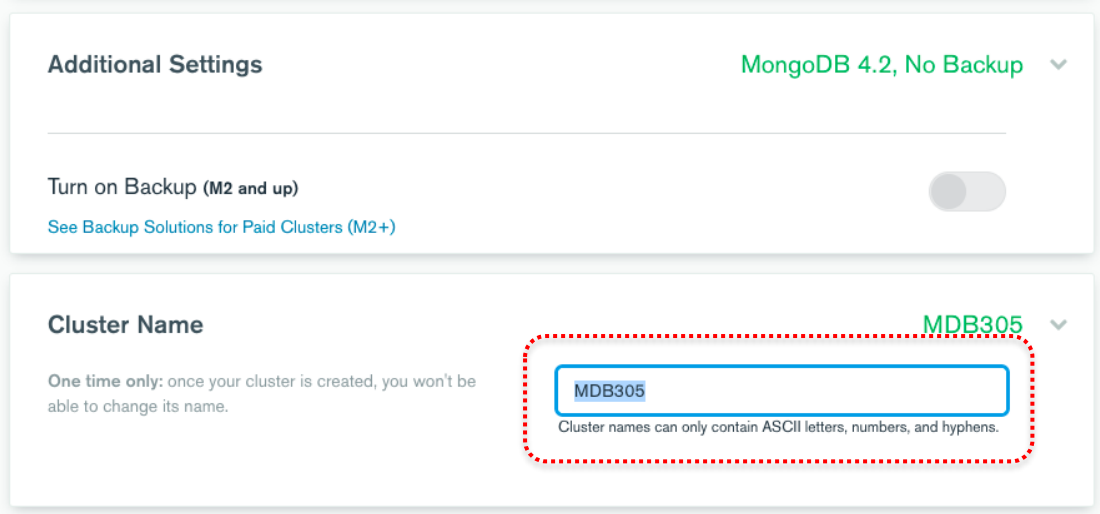
- Create a **shared** cluster (as free of charge)



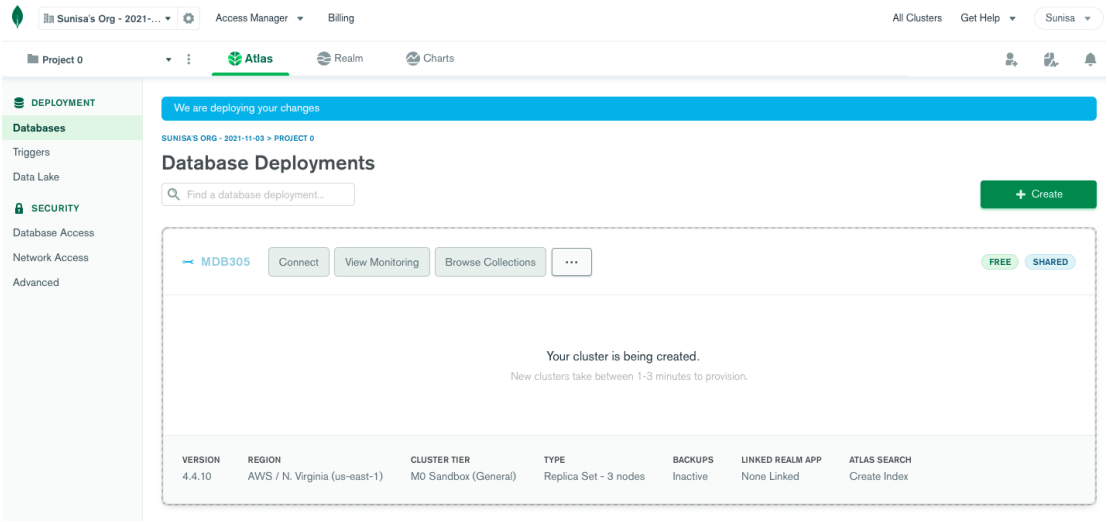
- Select any provider from the *Cloud Provider & Region* section. Different providers offer different regions.
- Select any region marked "FREE TIER AVAILABLE".



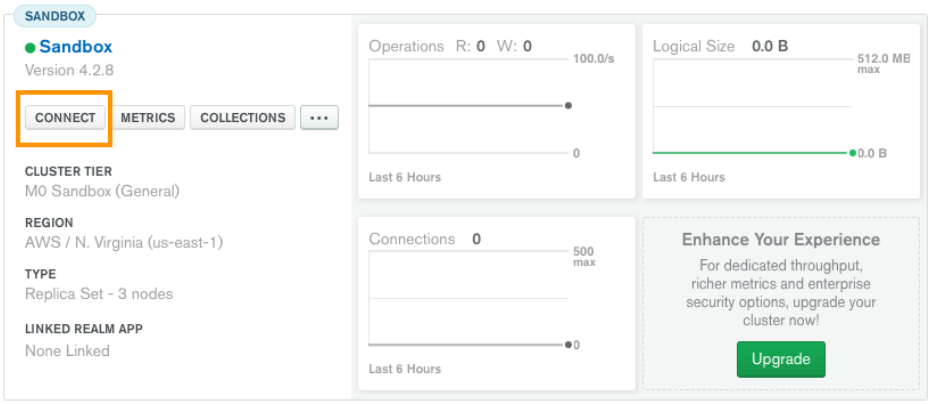

- Specify the cluster named **MDB305**



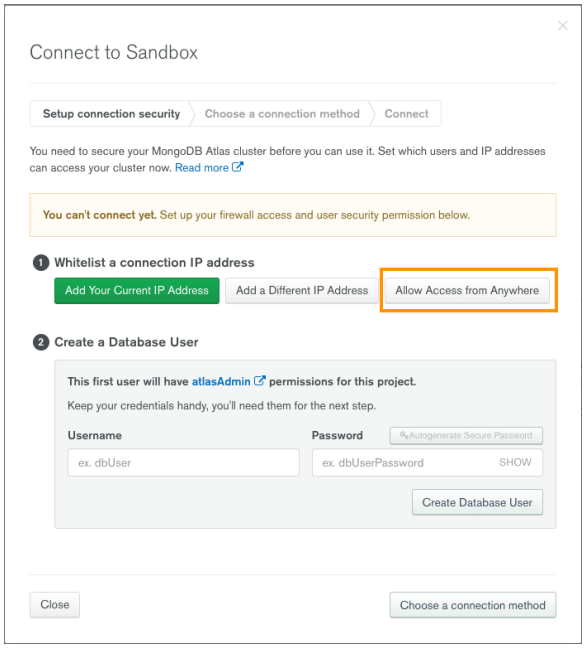
- Click the **Create Cluster** button (*This step might take a minute or two to complete*).



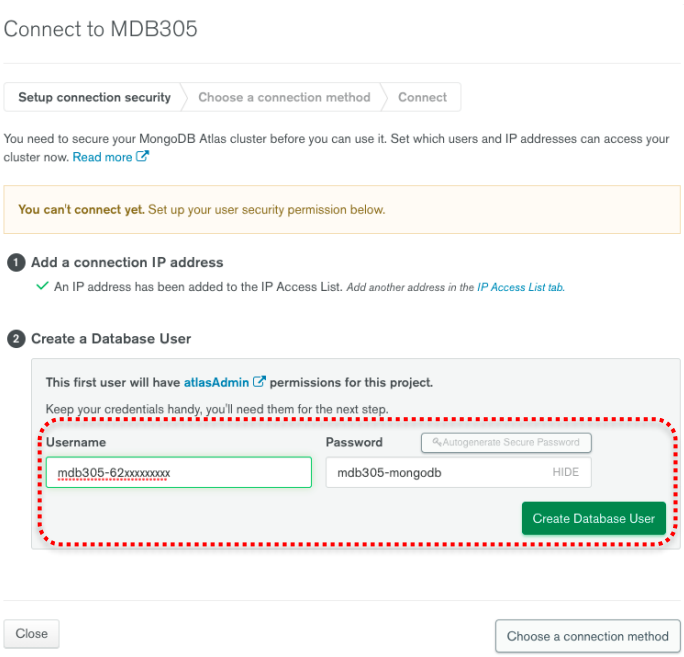

- Now that you have an Atlas cluster you need to grant access to your IP Address and create a Database User.
    - Select **Connect** from the cluster view.



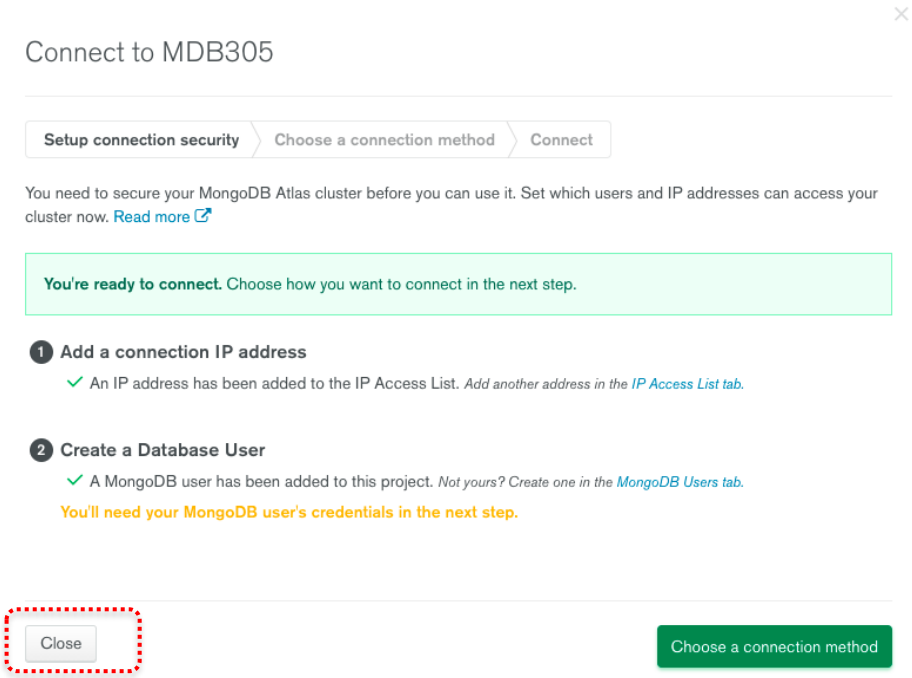

    - Select the *right-most* option **Allow Access from Anywhere.** This will open a form with `0.0.0.0/0` pre-seeded for the IP Address. and confirm your selection by clicking on **Add IP Address**. Allowing access from anywhere is not a good security practice. Clusters that are used for production should not have this enabled.




- Create a Database User
    - **username**: mdb305-*<student-id>*
    - **password**: mdb305-mongodb
- Click on the **Create Database User** button

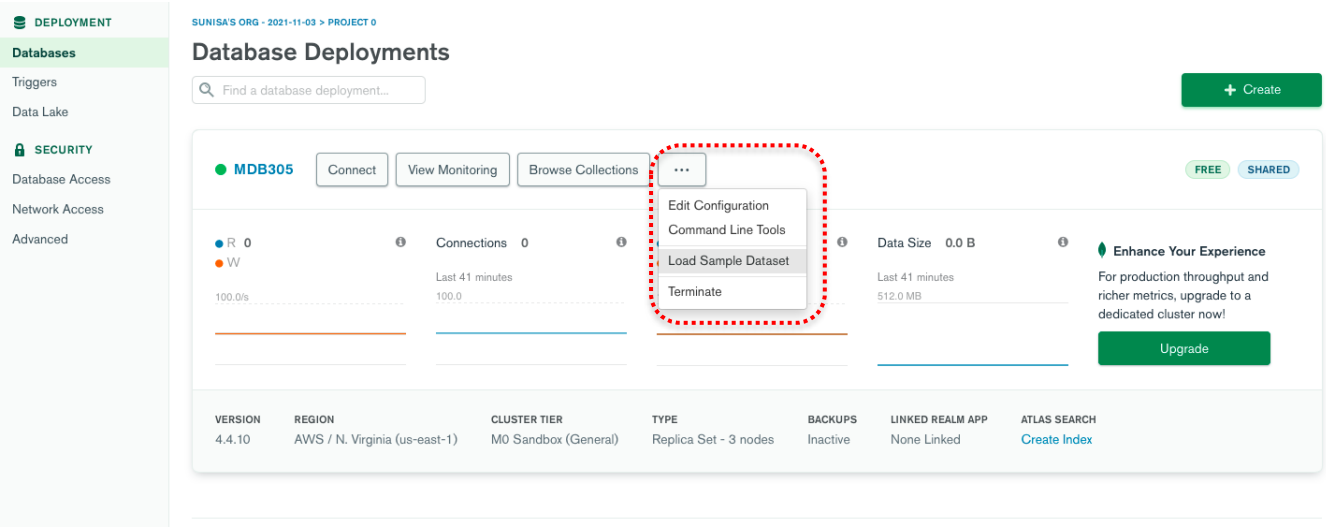


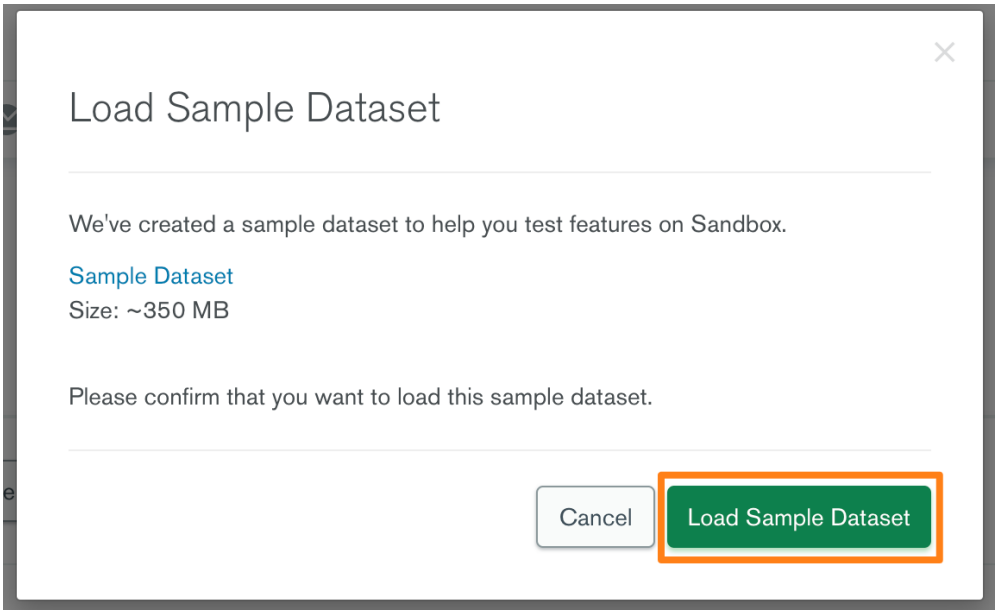

- Close the Connection menu at the lower left corner of the window.



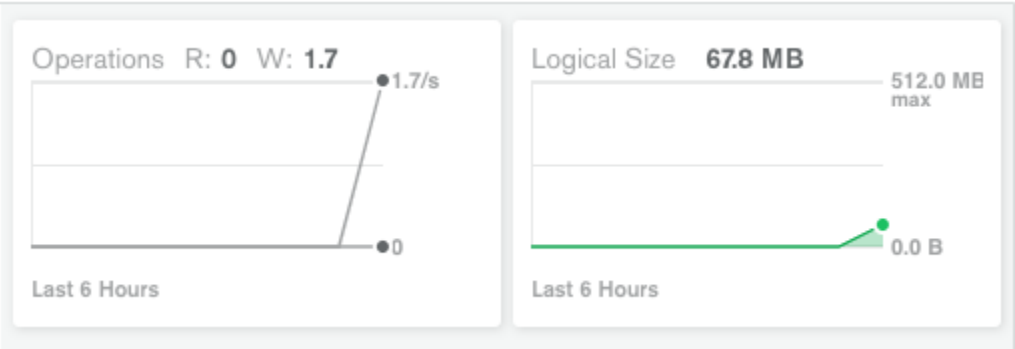
## 1.2 Load the Sample Dataset

Select the "…" option in the cluster menu -> choose the "Load Sample Dataset" option, then confirm your choice.

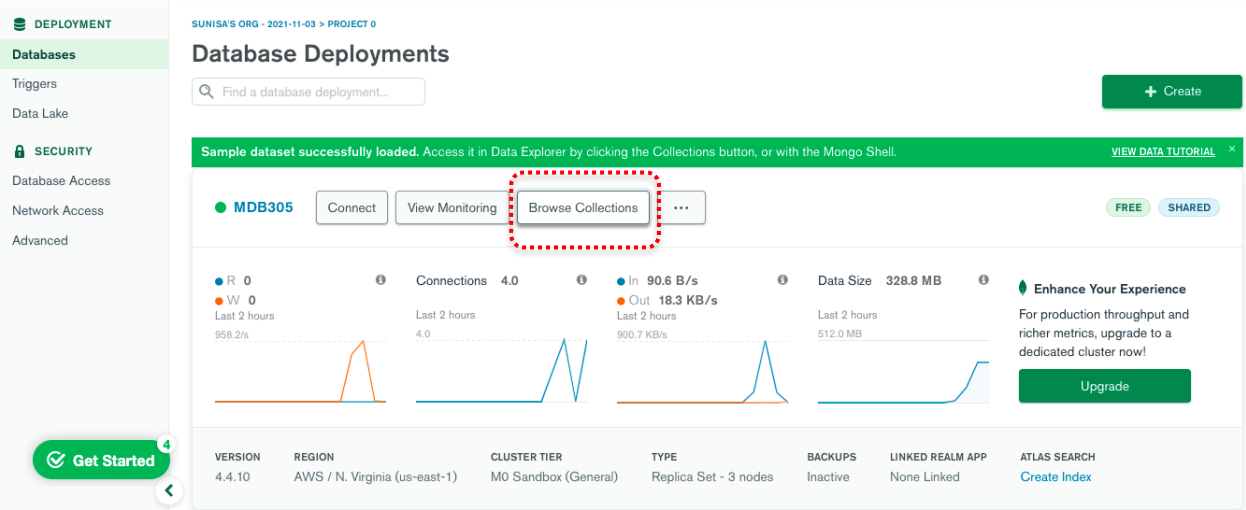



When the dataset is loaded the graph labeled "Logical Size" on the right side of the screen should go up and display the size of the dataset that is above zero and below 512 MB. Your graph may look different than the picture below.

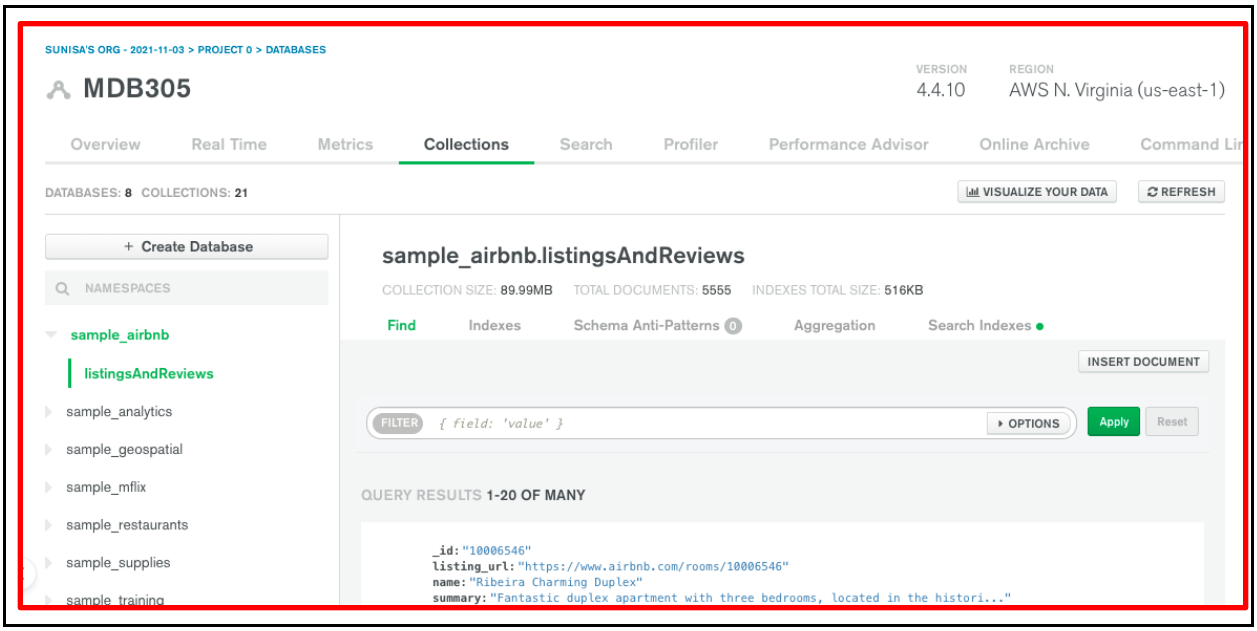



- *This step might take 5-10 minutes to complete.*
- *Click on the **Browse Collections** button to view the sample databases.*




***** Checkpoint 1*****

After you completed the tasks 1.1-1.2, **please capture "a screen of your cluster with Collections tab" here.** The picture should show your organization name and project name

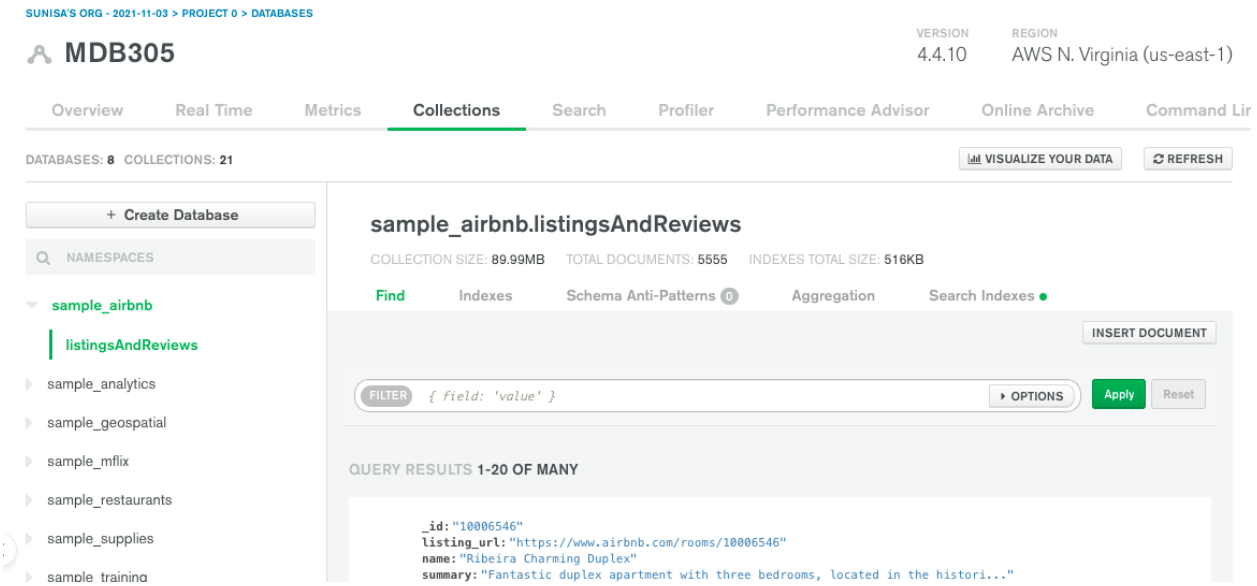



## Place your screenshot below

**1.3 Understand the Sample Dataset**

***** Checkpoint 2*****

- view documents in each collection of each database and **answer the following questions: A, B and C.**




A: How many databases are created in the sample dataset? Ans.: _____

B: How many collections are created in the database named SAMPLE_TRANING?

Please identify the collection names in this database.

Ans.: _____

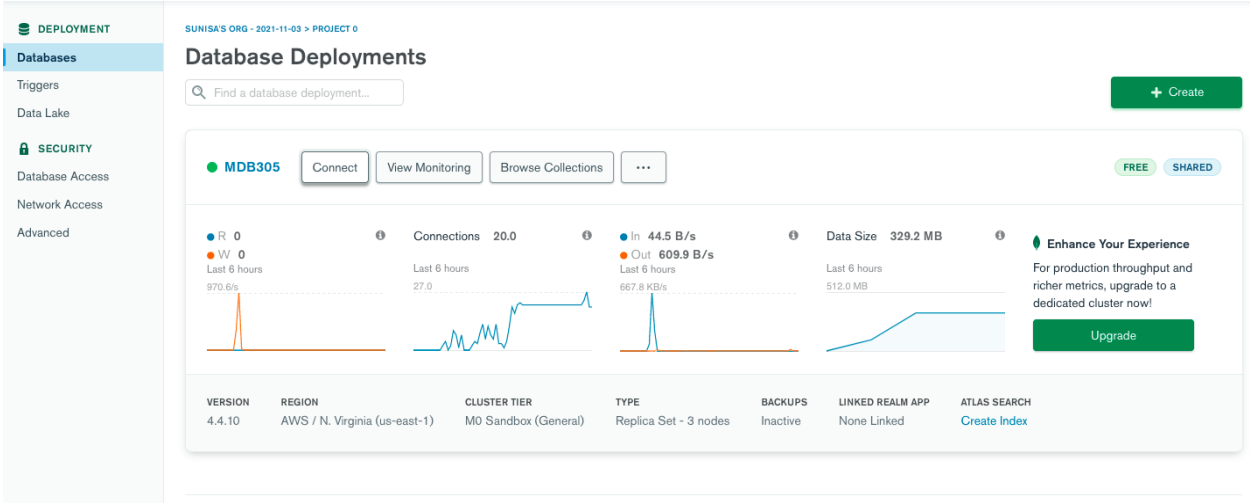
C: Please explain metadata (fields) stored in the SAMPLE_TRANING.ZIPS namespace.

Ans.: _____

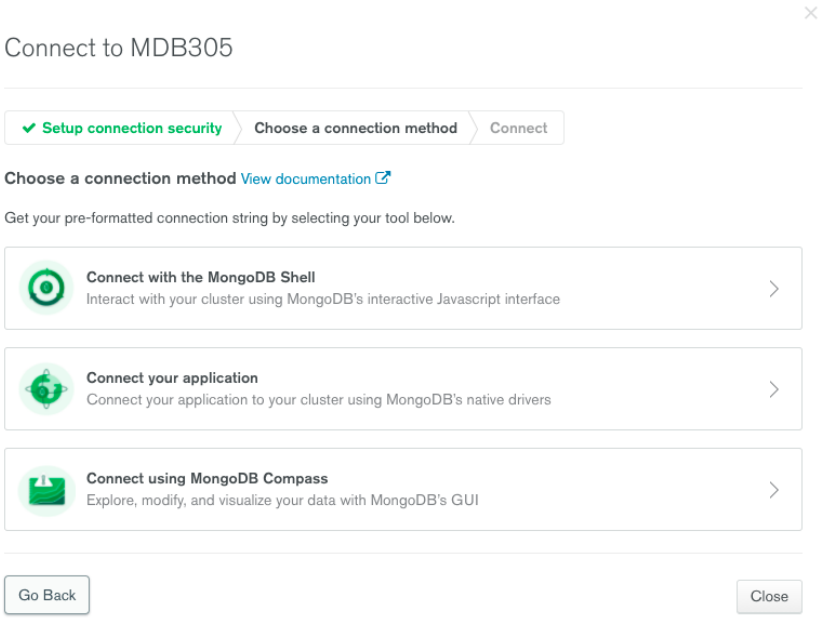
## Task 2: Connect to your cluster using the MongoDB Compass

### 2.1 Install the MongoDB Compass

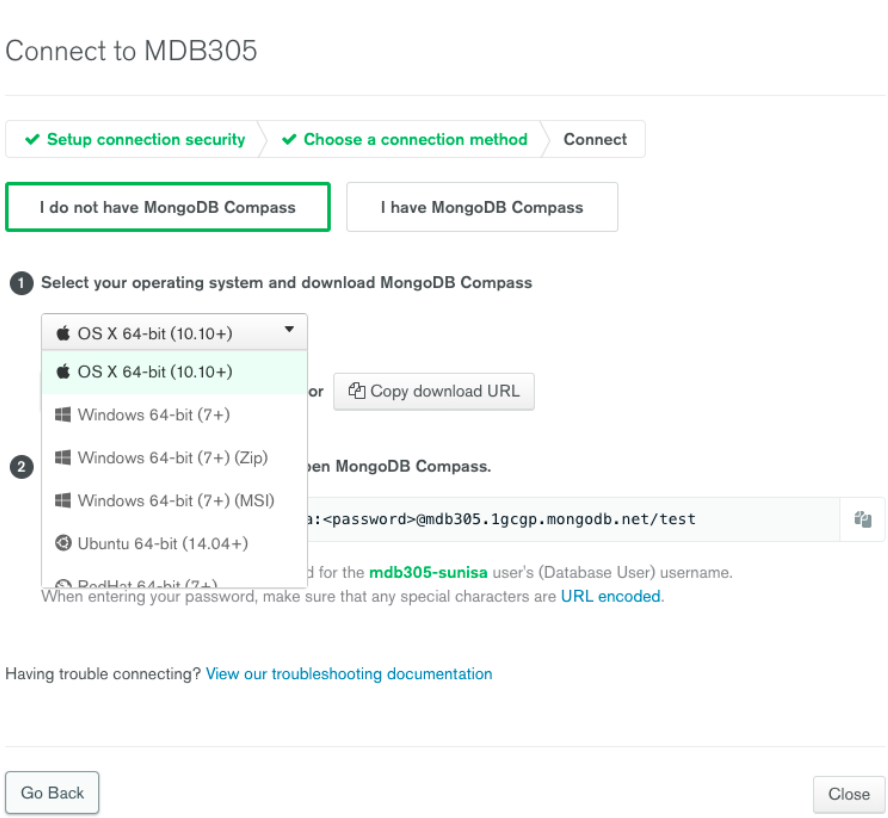
- Click on the **Database** menu to see the database view and click on the **Connect** button.



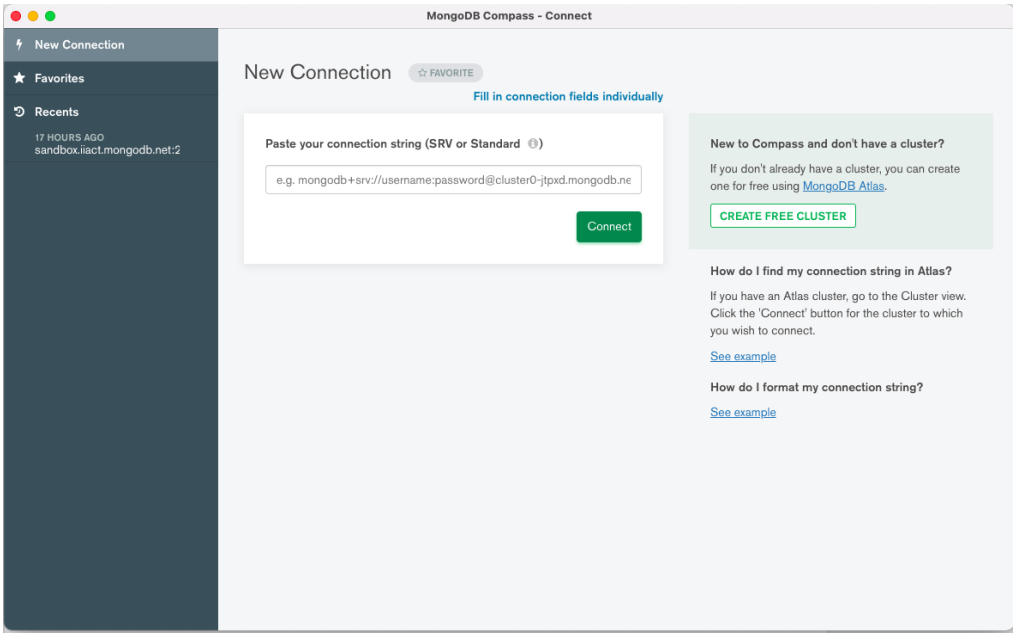

- Select a connection method as **Connect using MongoDB Compass**




- Select the "**I do not have MongoDB Compass**" option.
- Download a MongoDB Compass depending on your OS and install it.

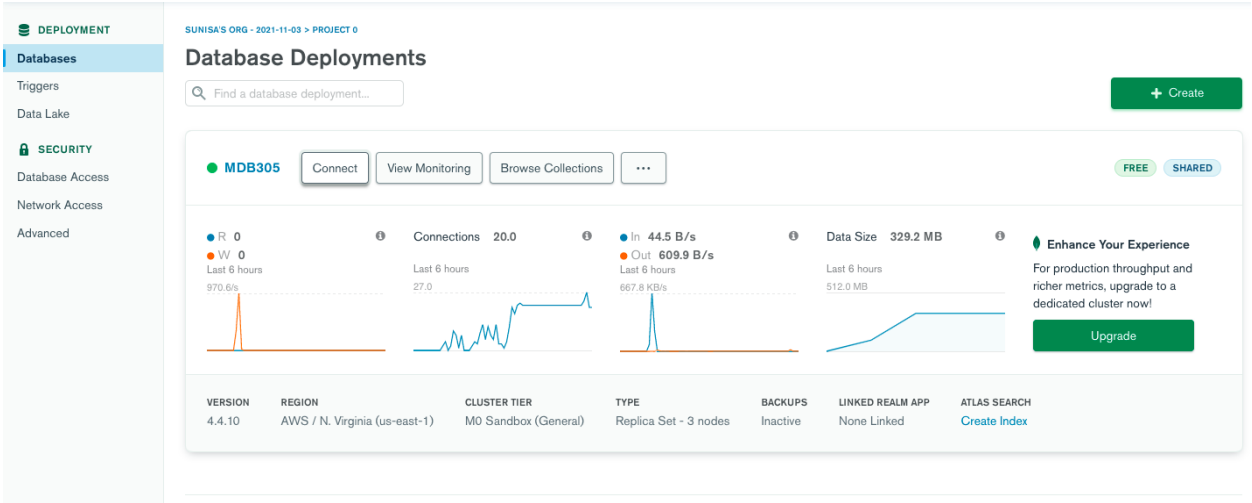


- After the installation, open the MongoDB Compass program. You should specify the connection string to connect to your cluster. If you are not sure about the correct connection string, you will do the next step.

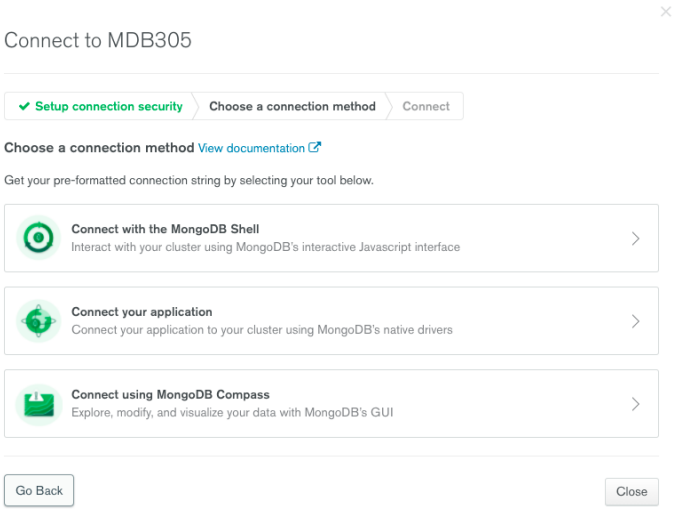



### 2.2 set up the MongoDB Compass connection

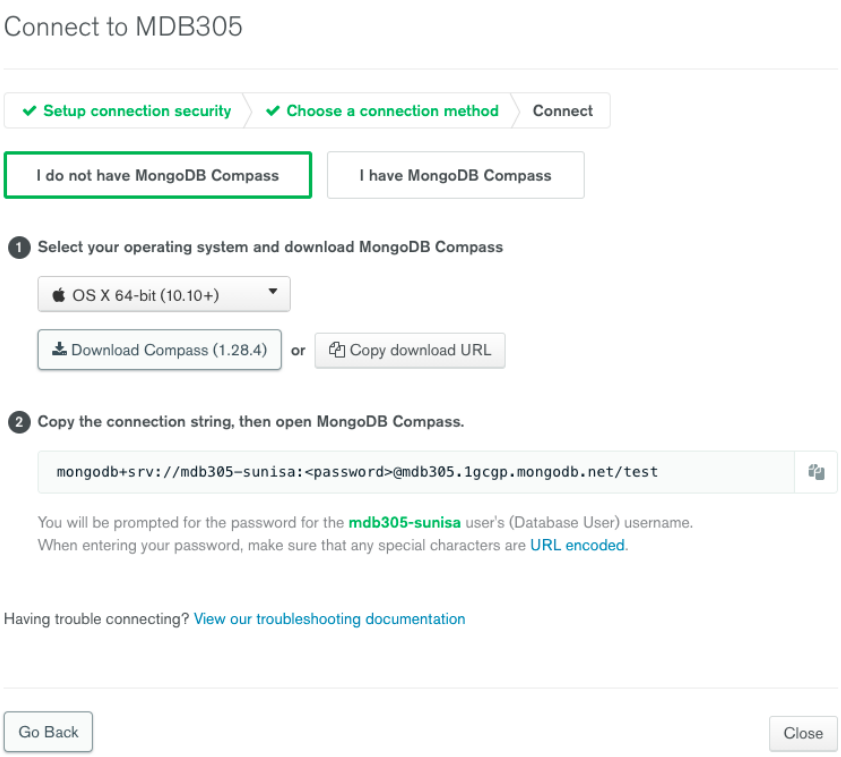
- Repeat these steps again
- Click on the **Database** menu to see the database homepage and click on the **Connect** button.




- Select a connection method as **Connect using MongoDB Compass**

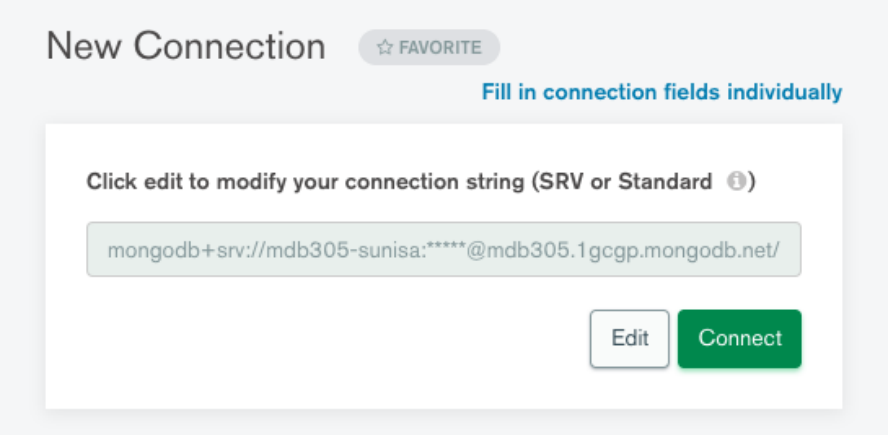



- You just copy the connection string in **no.2** only and close the window, then switch to the MongoDB Compass.

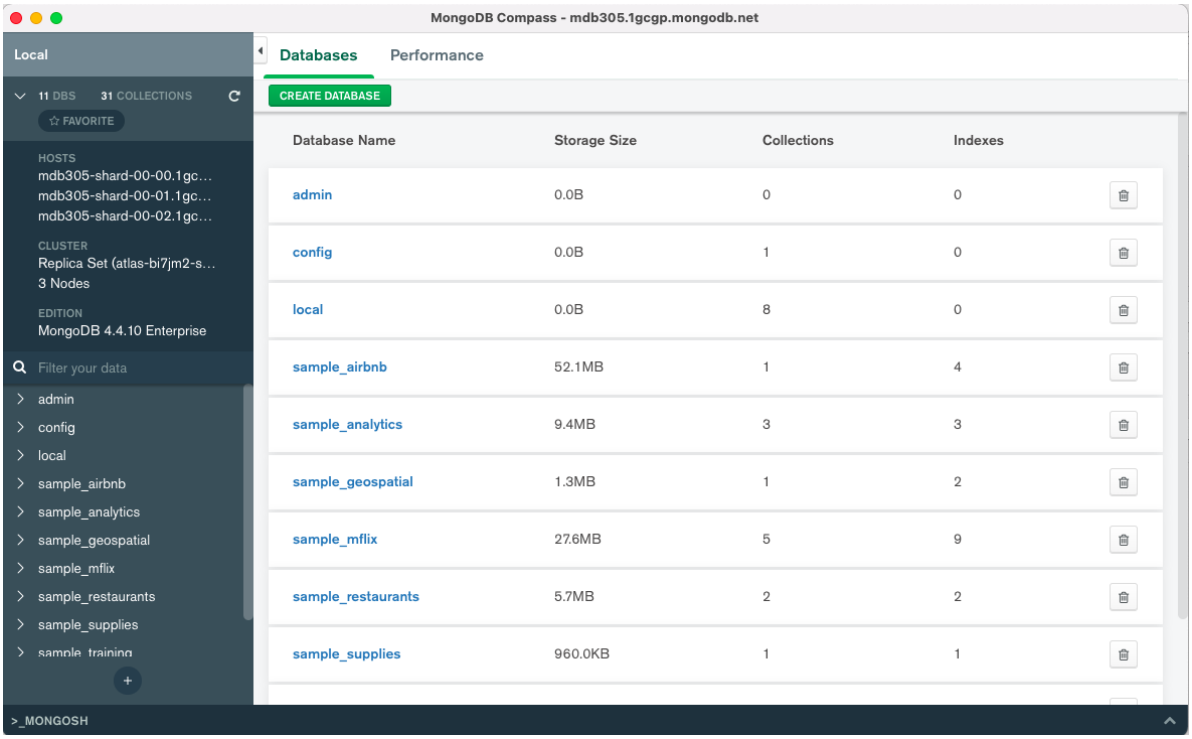



**2.3 Connect to your cluster**

- In the MongoDB Compass program, paste the connection string and specify your password.

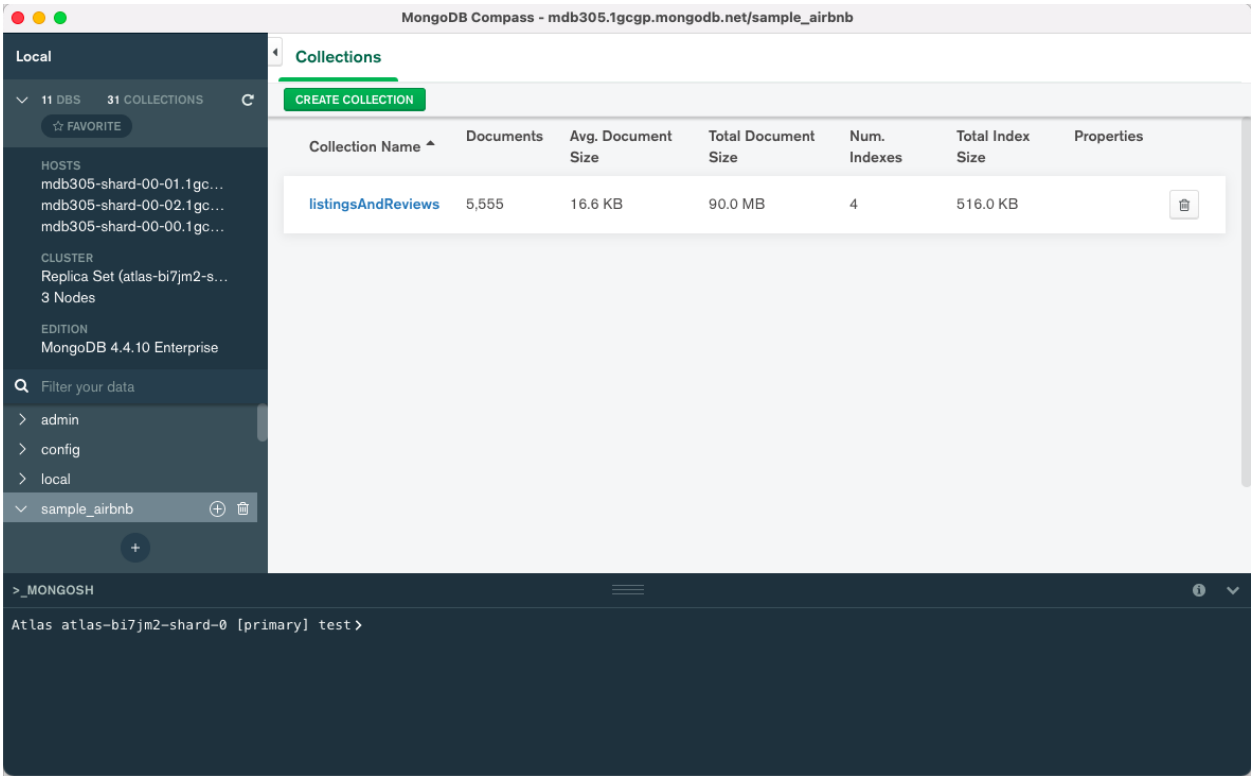



- If you can connect to your cluster via the MongoDB Compass, you will see the list of the databases as below:




**Task 3: Use basic MongoDB commands**

- Run MongoDB commands by clicking on the **MONGOSH** (MongoDB Shell) menu at the lower left corner of the window.




- **Run the following commands**

  > show dbs;

  > use <database_name>;

  > show collections;

**FIND() Collection Method:**

**Resource:**

https://docs.mongodb.com/v4.2/reference/method/db.collection.find/

`db.collection.find(query, projection)`: Selects documents in a collection or view and returns a cursor to the selected documents.

| Parameter | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| query | document | Optional. Specifies selection filter using query operators. To return all documents in a collection, omit this parameter or pass an empty document ( { } ). |
| projection | document | Optional. Specifies the fields to return in the documents that match the query filter. To return all fields in the matching documents, omit this parameter. For details, see Projection. |

**Returns:** A cursor to the documents that match the `query` criteria. When the `find()` method "returns documents," the method is actually returning a cursor to the documents.

**Query**

**{ field1: <value> }**

**{ field:{$lt:value}} // $lt, $lte, $gt, $gte, $ne**

**{ field1: <value>, field2: <value> ... }**

**Projection**

The `projection` parameter determines which fields are returned in the matching documents. The `projection` parameter takes a document of the following form:

```
{ field1: <value>, field2: <value> ... }
```

The <value> can be any of the following:

1 or true to include the field in the return documents.

0 or false to exclude the field.

**Example:**

- The following query uses `$gt` to return documents where the value of `qty` is greater than `4`.

```
db.collection.find( { qty: { $gt: 4 } } )
```

- The following operation finds all documents in the bios collection and returns only the name field, contribs field and _id field:

```
db.bios.find( { }, { name: 1, contribs: 1 } )
```

**COUNT() Collection Method:**

Resource: https://docs.mongodb.com/v4.2/reference/method/db.collection.count/

`db.collection.count(query, options)` Returns the count of documents that would match a `find()` query for the collection or view. The `db.collection.count()` method does not perform the `find()` operation but instead counts and returns the number of results that match a query.

| Parameter | Type | Description |
|---|---|---|
| `query` | document | The query selection criteria. |
| `options` | document | Optional. Extra options for modifying the count. |

Note: count() is equivalent to the db.collection.find(query).count() construct.

***** Checkpoint 3*****

**Write a mongoDB command to answer the following questions:**

3.1 Write a command to use the SAMPLE_TRANING database

```
> use sample_training
< 'switched to db sample_training'
```

3.2 Write a command to view documents of the zips collection that population is less than 10000.

```
> db.zips.find({pop: { $lt: 10000}})
< { _id: ObjectId("5c8eccc1caa187d17ca6ed16"),
    city: 'ALPINE',
    zip: '35014',
    loc: { y: 33.331165, x: 86.208934 },
    pop: 3062,
    state: 'AL' }
  { _id: ObjectId("5c8eccc1caa187d17ca6ed18"),
    city: 'ACMAR',
    zip: '35004',
    loc: { y: 33.584132, x: 86.51557 },
    pop: 6055,
    state: 'AL' }
  { _id: ObjectId("5c8eccc1caa187d17ca6ed19"),
    city: 'BAILEYTON',
```

3.3 Write a command to display _id, city and pop fields of documents in the zips collection that population is less than 10000.

```
> db.zips.find({pop: { $lt: 10000}},{_id: 1 ,city: 1,pop: 1})
< { _id: ObjectId("5c8eccc1caa187d17ca6ed16"),
    city: 'ALPINE',
    pop: 3062 }
  { _id: ObjectId("5c8eccc1caa187d17ca6ed18"),
    city: 'ACMAR',
    pop: 6055 }
  { _id: ObjectId("5c8eccc1caa187d17ca6ed19"),
    city: 'BAILEYTON',
    pop: 1781 }
  { _id: ObjectId("5c8eccc1caa187d17ca6ed1b"),
    city: 'BLOUNTSVILLE',
    pop: 9058 }
  { _id: ObjectId("5c8eccc1caa187d17ca6ed1c"),
    city: 'BRIERFIELD',
```

3.4 Write a command to view all fields of documents excluding the end station location and the start station location fields in the trips collection that the user type is "Subscriber" and the start station id is 268.

```
> db.trips.find({usertype: "Subscriber","start station id": 268},{"start station location": 0,"end station location": 0})
< { _id: ObjectId("572bb8222b288919b68abf5b"),
    tripduration: 889,
    'start station id': 268,
    'start station name': 'Howard St & Centre St',
    'end station id': 3002,
    'end station name': 'South End Ave & Liberty St',
    bikeid: 22794,
    usertype: 'Subscriber',
    'birth year': 1961,
    'start time': 2016-01-01T00:01:06.000Z,
    'stop time': 2016-01-01T00:15:56.000Z }
  { _id: ObjectId("572bb8222b288919b68abf5d"),
    tripduration: 923,
    'start station id': 268,
    'start station name': 'Howard St & Centre St',
    'end station id': 3002,
    'end station name': 'South End Ave & Liberty St',
    bikeid: 22285,
    usertype: 'Subscriber',
    'birth year': 1958,
    'start time': 2016-01-01T00:00:41.000Z,
    'stop time': 2016-01-01T00:16:04.000Z }
```

3.5 Write a command to answer "How many documents of the trips collection that have a user type named "Subscriber" and the birth year is equal to 1964 ?".

```
> db.trips.find({usertype: "Subscriber","birth year": 1964},{start_station_location: 0,end_station_location: 0}).count()
< 99
```