Lesson 01 Demo 03

Performing CRUD Operations with Mongo Shell

Objective: To perform CRUD operations with Mongo Shell by installing mongosh and performing update operations

Tools Required: VS Code

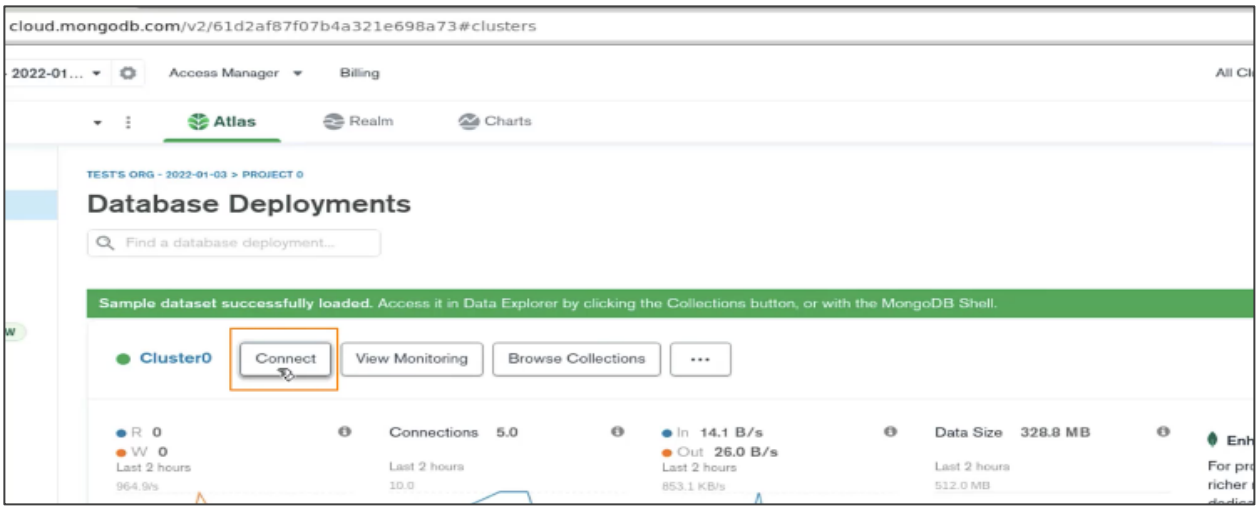
Prerequisites: None

Steps to be followed:

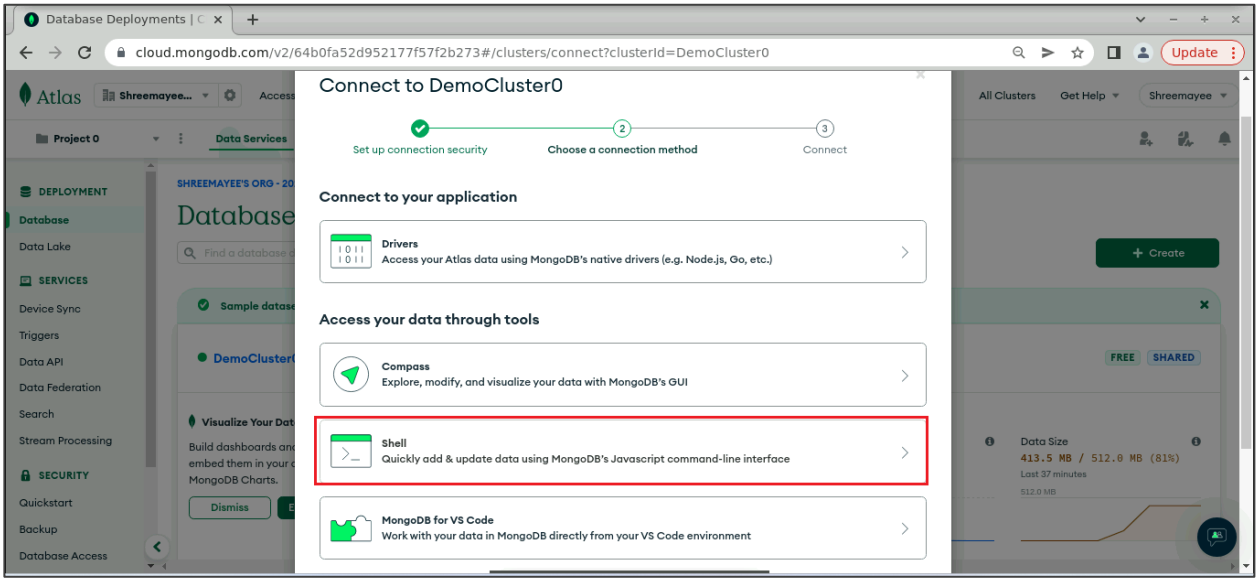
1. Install mongosh
2. Install Mongo Shell
3. Connect to MongoDB
4. Execute commands
5. Create a document
6. Insert more documents
7. Perform update operations

Step 1: Install mongosh

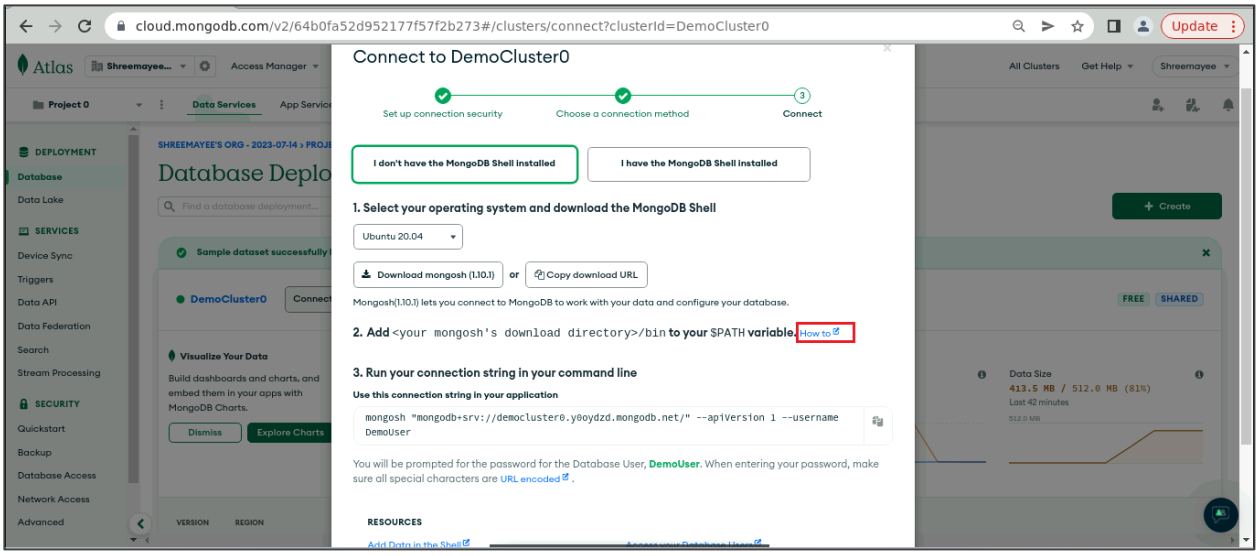
1.1 On the main interface of MongoDB Atlas, click on **Connect**



1.2 Select the **Shell** option to connect with MongoDB Shell



1.3 Click on **How to** and set the path variable for the MongoDB shell



1.4 Follow the instructions to import the public key used by the package management system

1 **Import the public key used by the package management system.**

From a terminal, issue the following command to import the MongoDB public GPG Key from https://www.mongodb.org/static/pgp/server-5.0.asc:

```
wget -qO - https://www.mongodb.org/static/pgp/server-5.0.asc | sudo apt-key add -
```

The operation should respond with an `OK`.

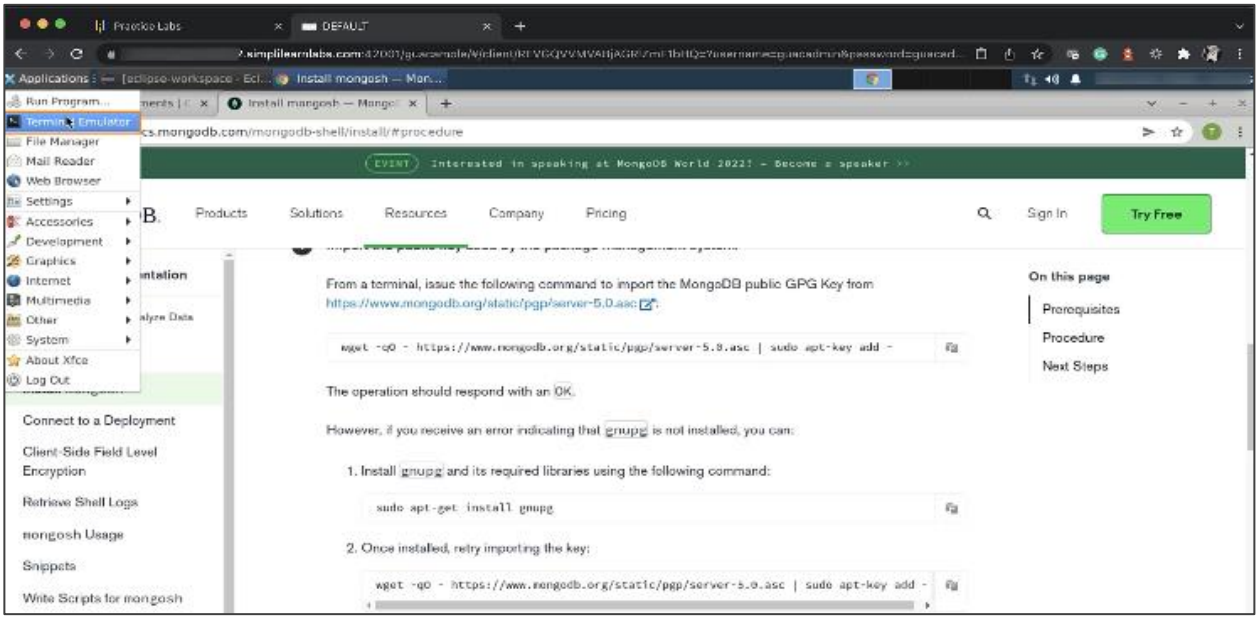
However, if you receive an error indicating that `gnupg` is not installed, you can:

1. Install `gnupg` and its required libraries using the following command:

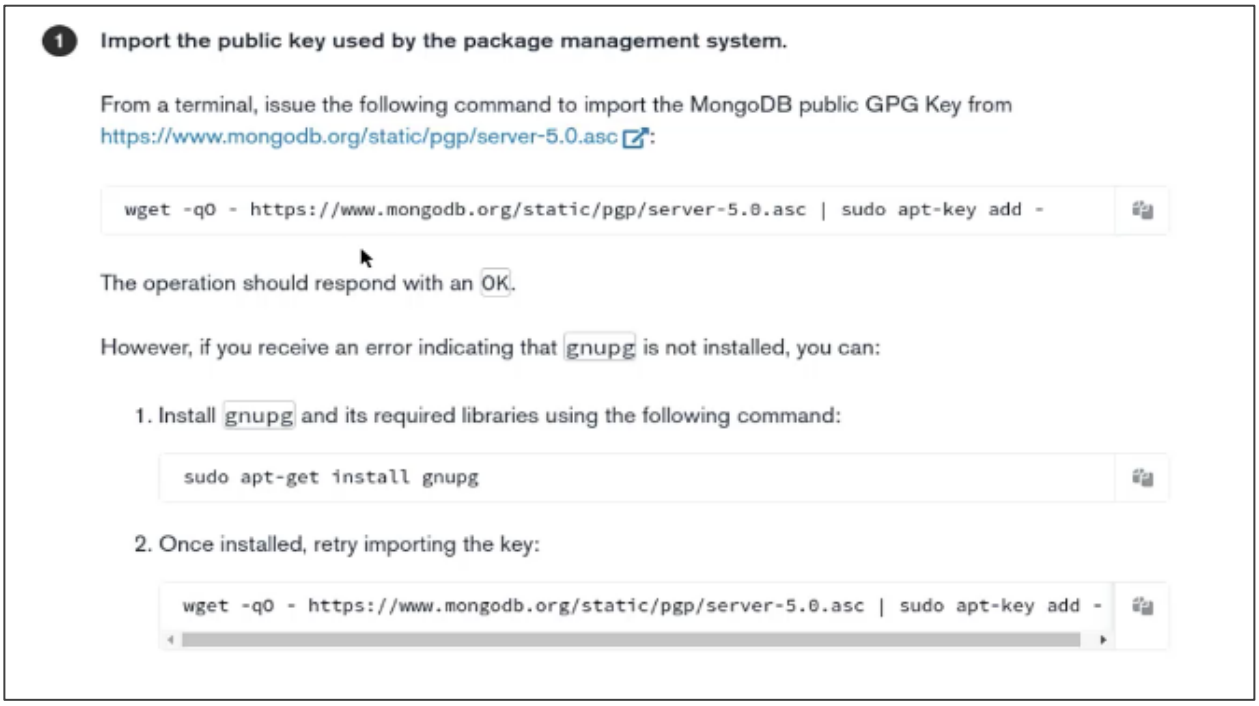
```
sudo apt-get install gnupg
```

2. Once installed, retry importing the key:

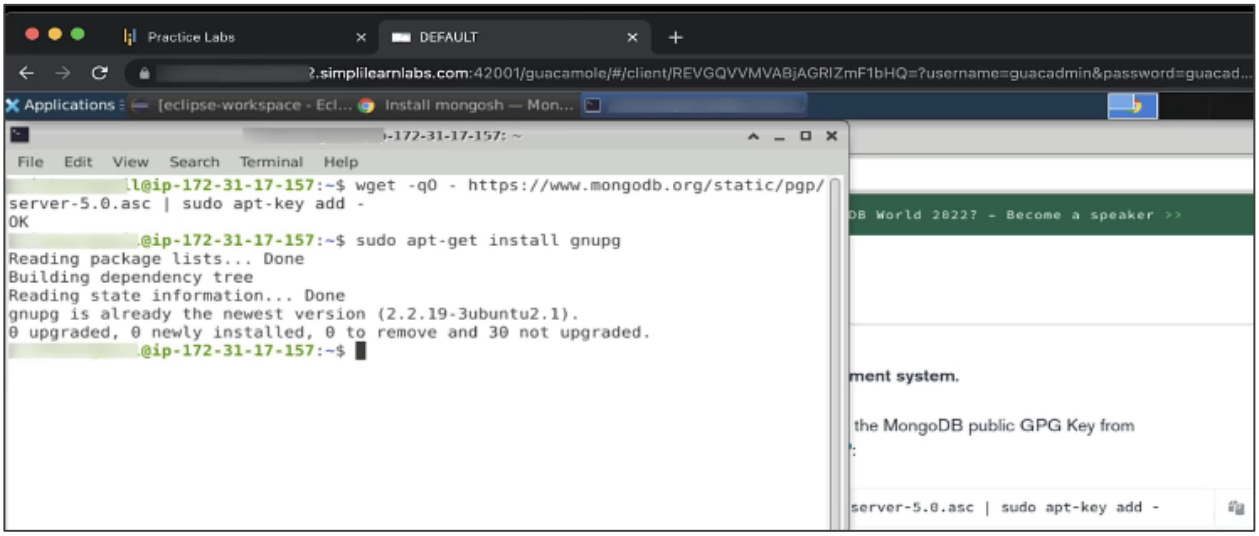
1.5 Open a **Terminal Emulator**



1.6 Copy the command below and paste it into the terminal to import the public GPG key:
sudo apt-get install gnupg

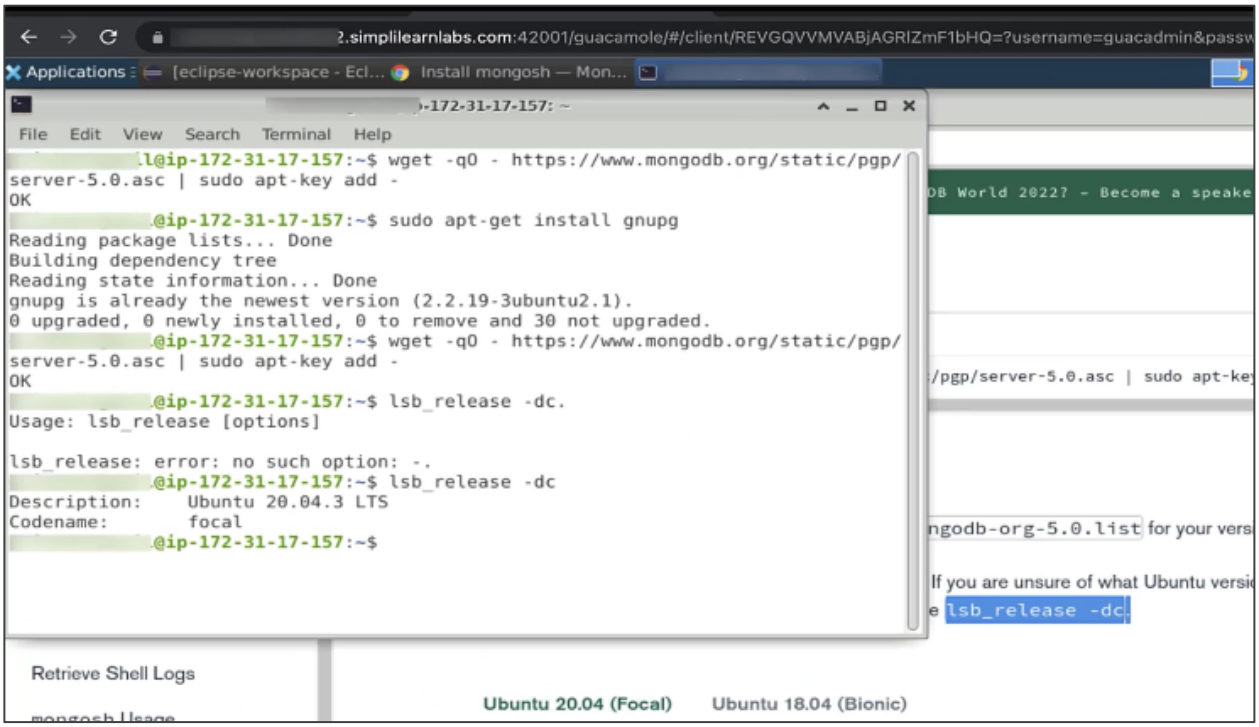


1.7 Paste the command in the terminal and press **Enter**

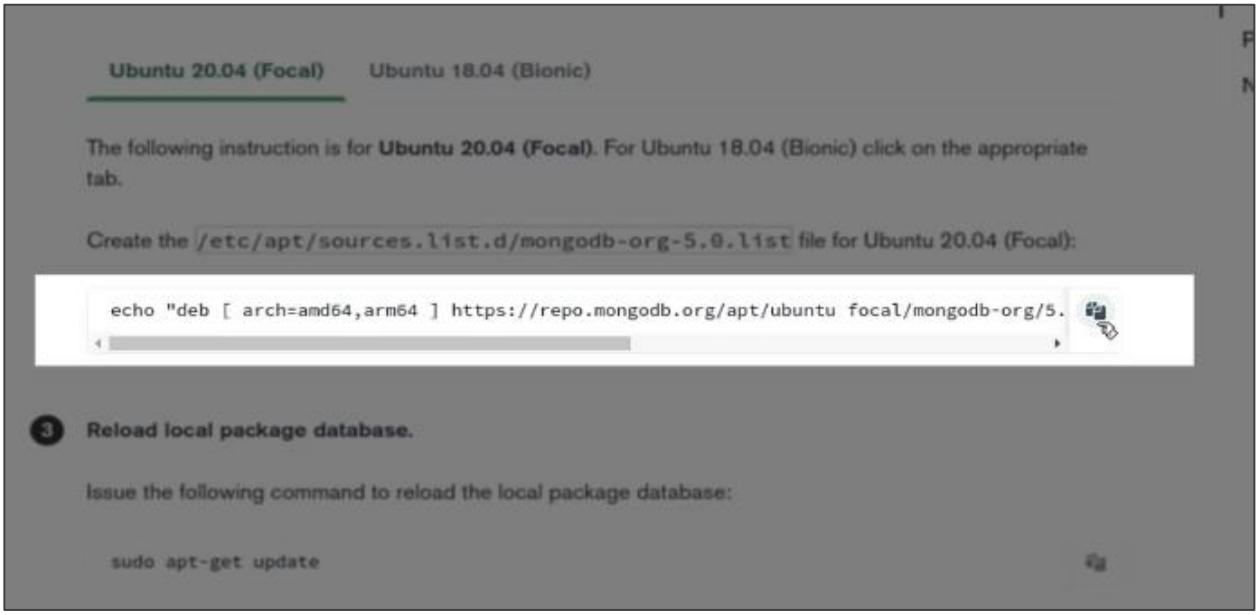


1.8 Execute the release command to check the version

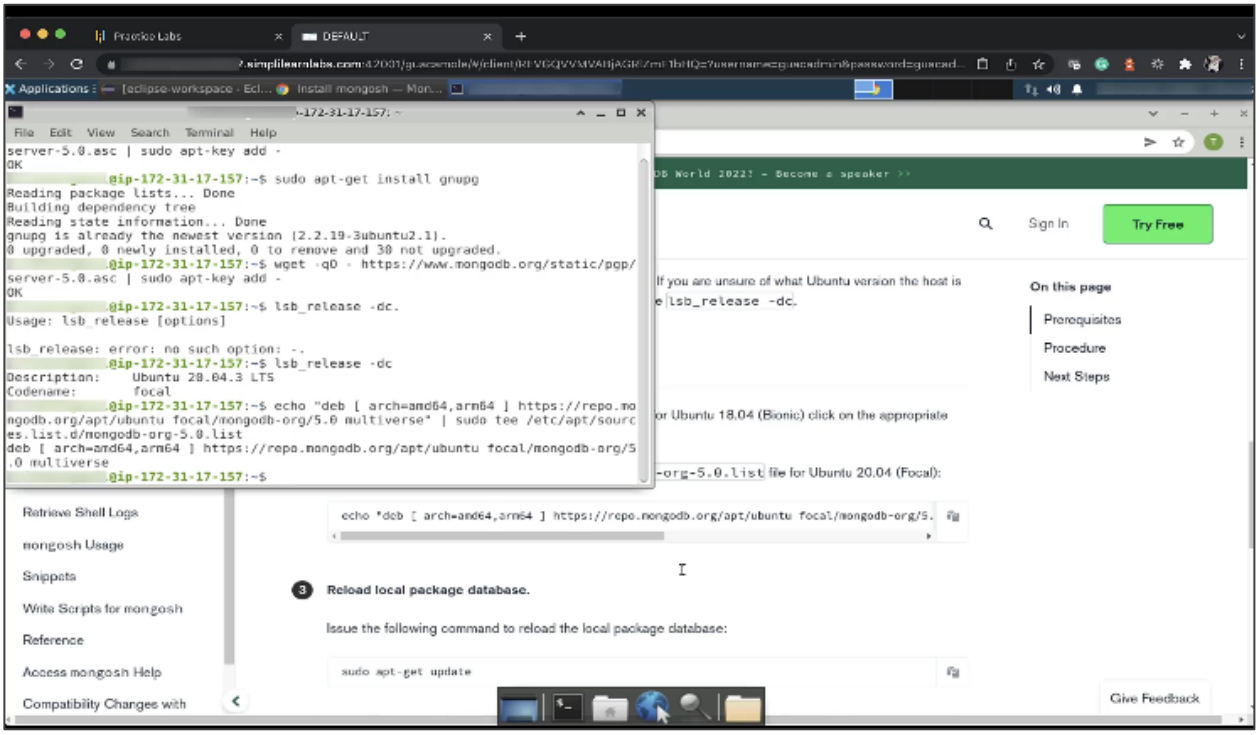
lsb_release -dc



1.9 Follow the Ubuntu 20.04 instructions. Create a file for Ubuntu 20.04 by copying the command from the screenshot below:

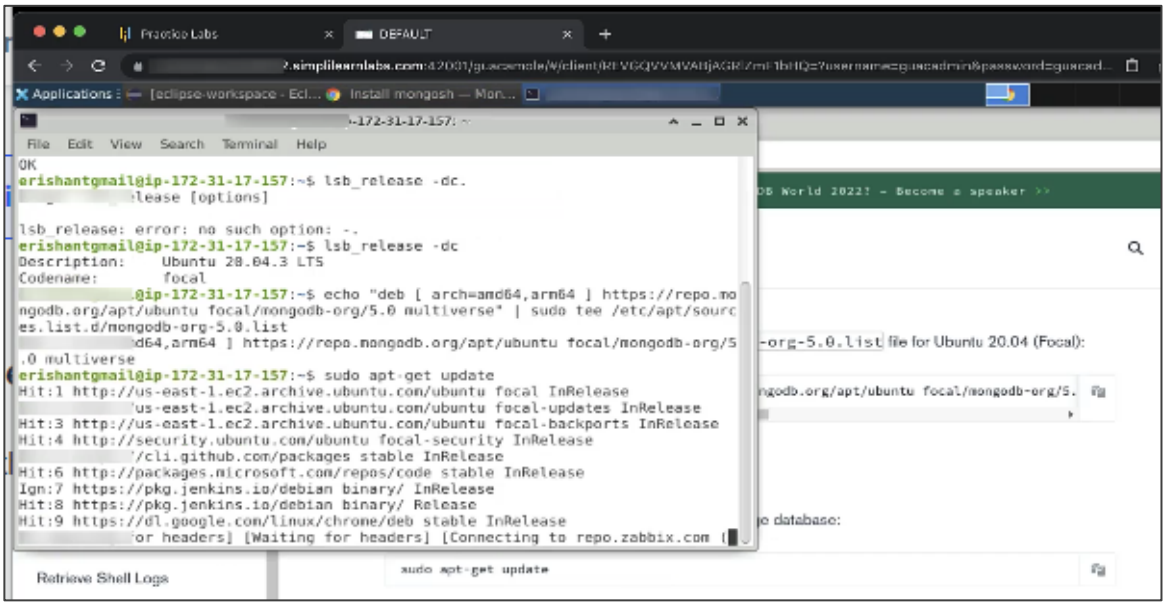


1.10 Paste it into the terminal



1.11 Once the file is created, reload the local package using the command below:

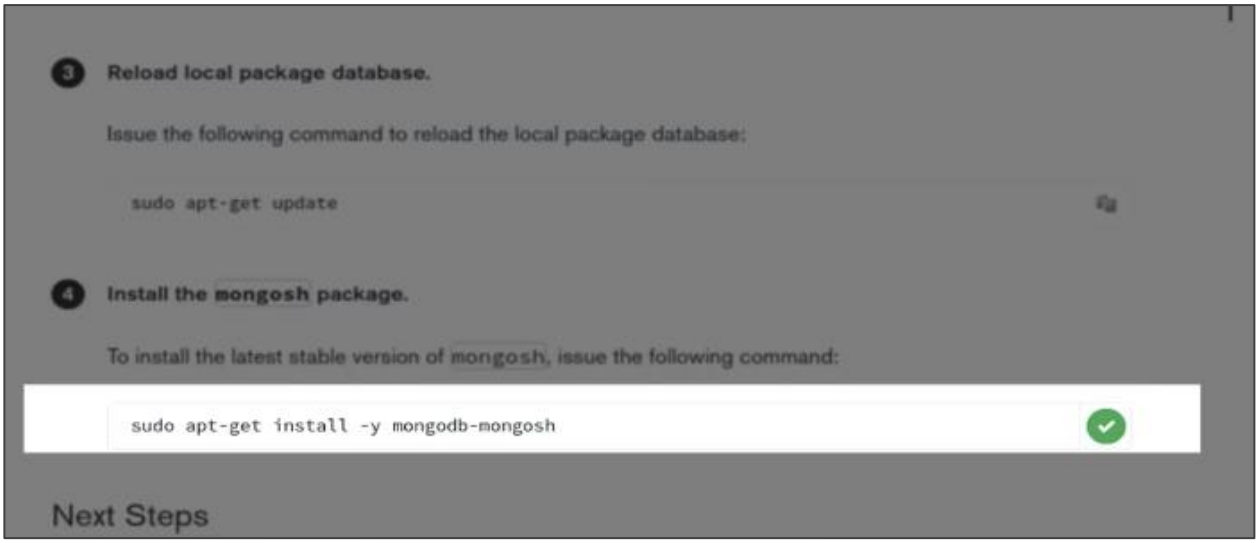
sudo apt-get update



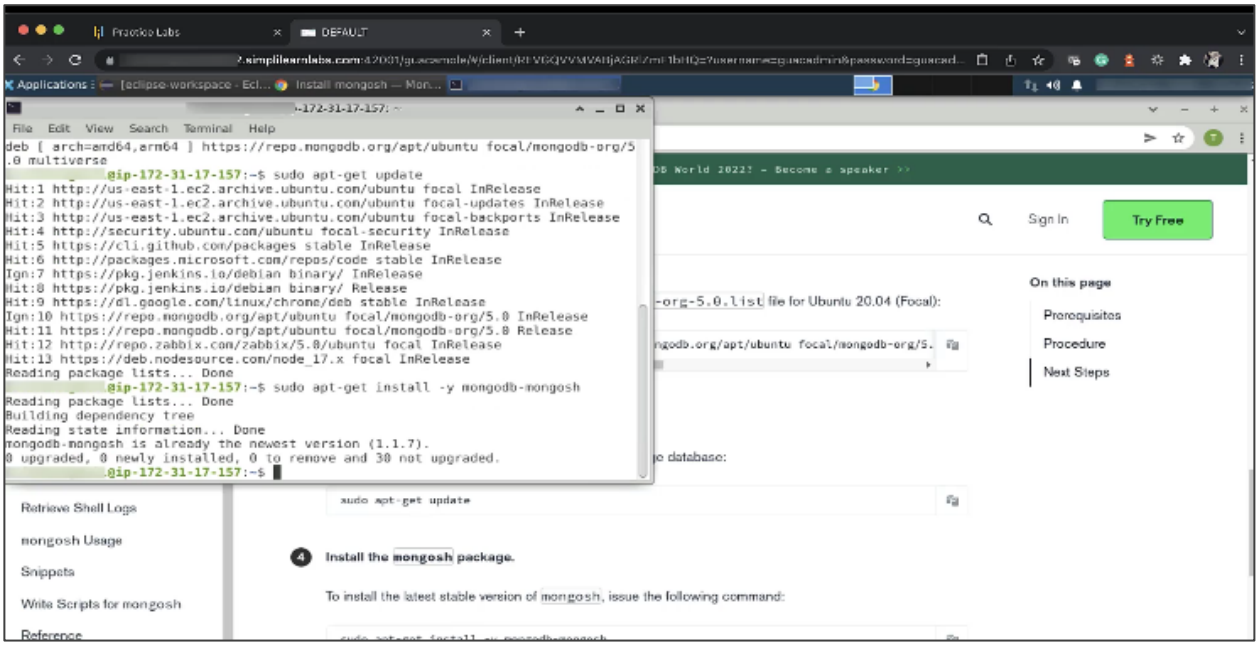
Step 2: Install Mongo Shell

2.1 Install the Mongo Shell using the command below:

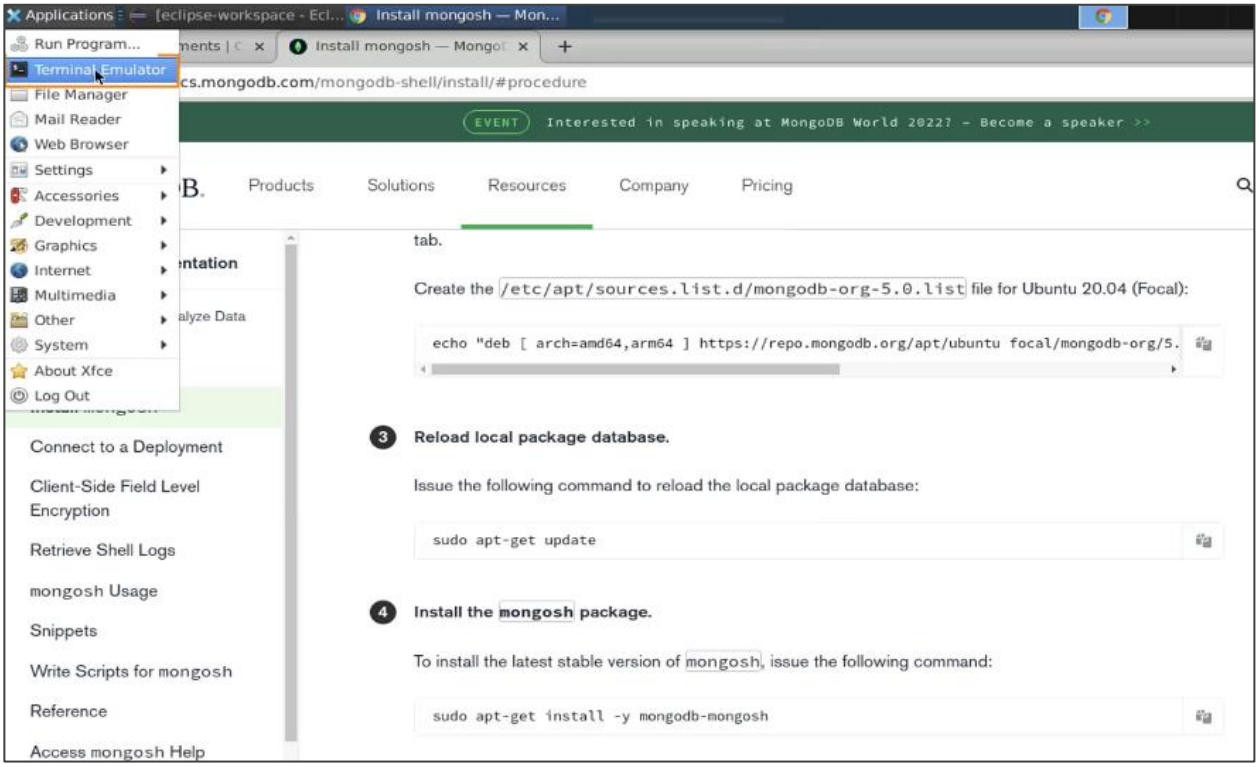
sudo apt-get install –y mongodb-mongosh



2.2 Paste the command in the terminal

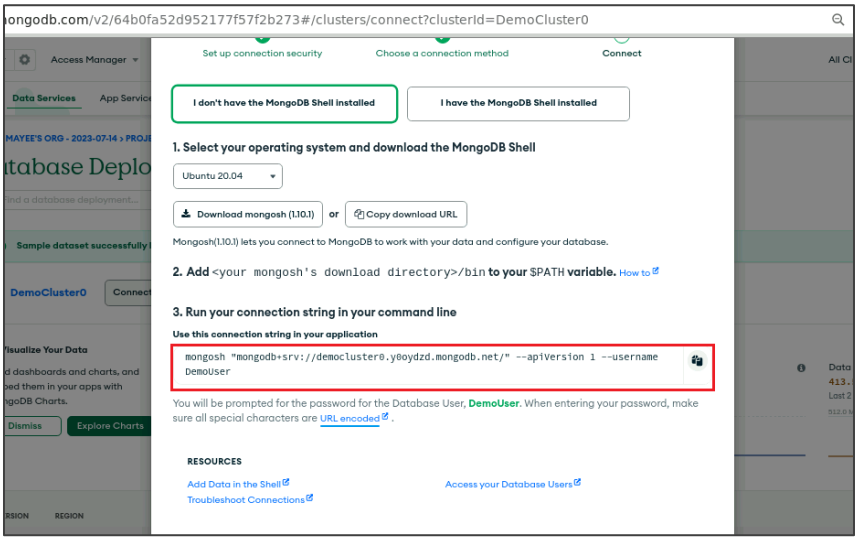


2.3 Close the terminal and reopen it by clicking on **Terminal Emulator**

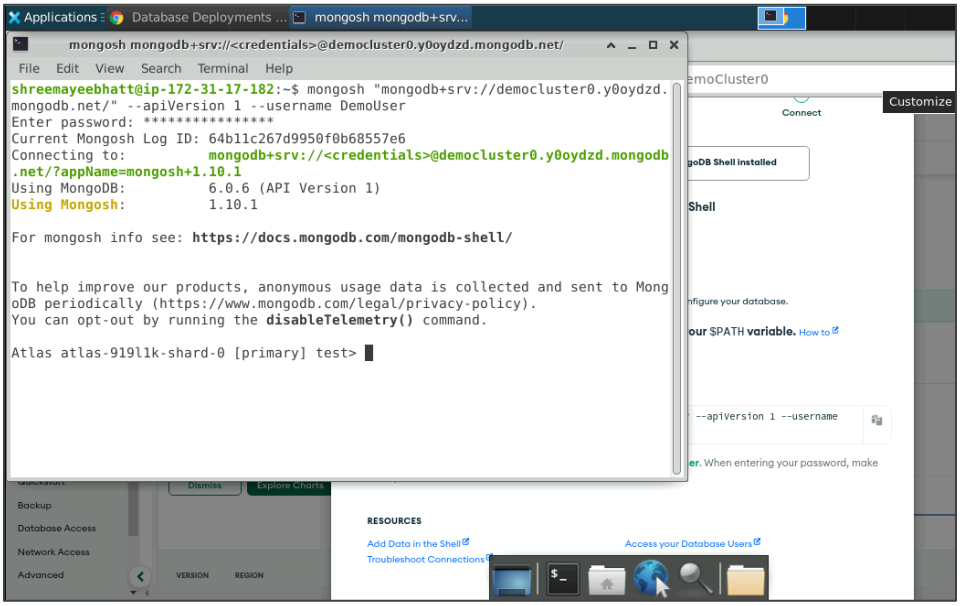


Step 3: Connect to MongoDB

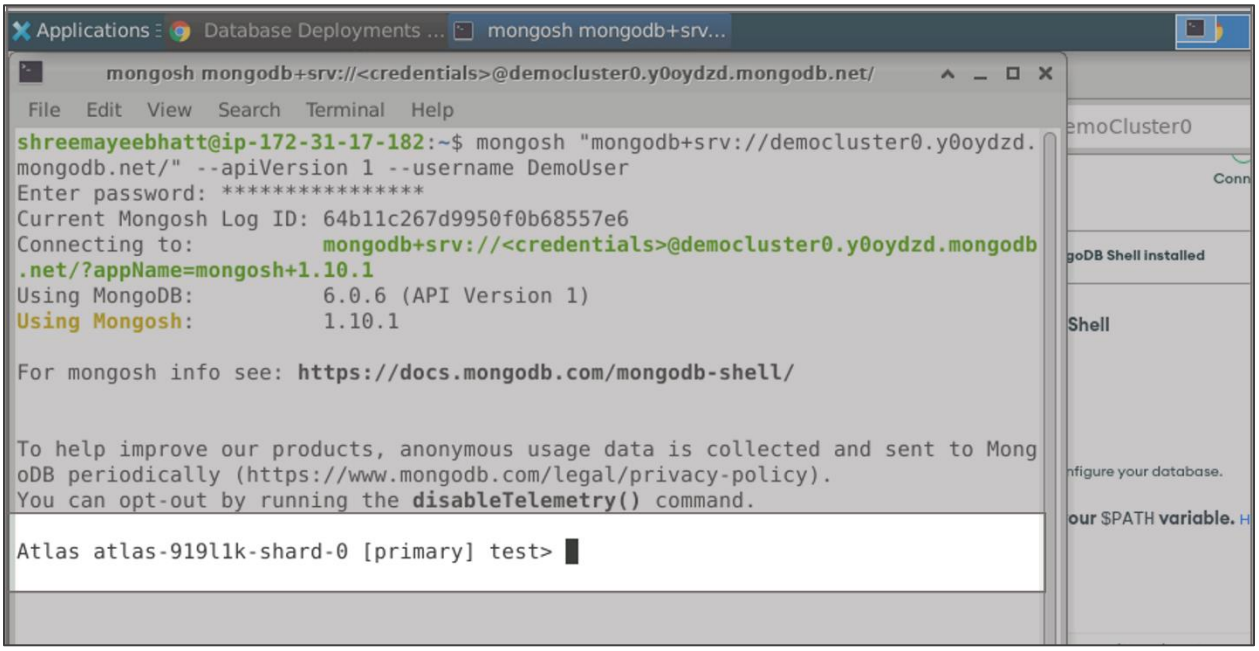
3.1 Connect to MongoDB by using the connection string in the application



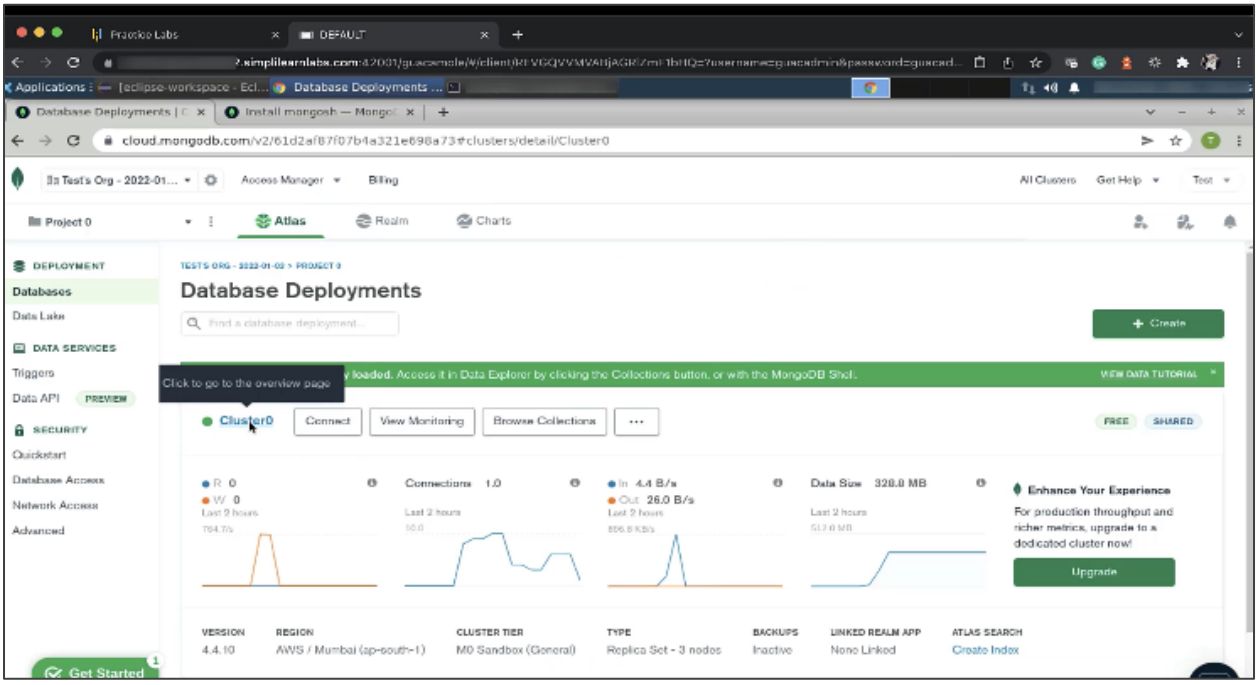
3.2 Paste it into the terminal and enter the password provided during account creation



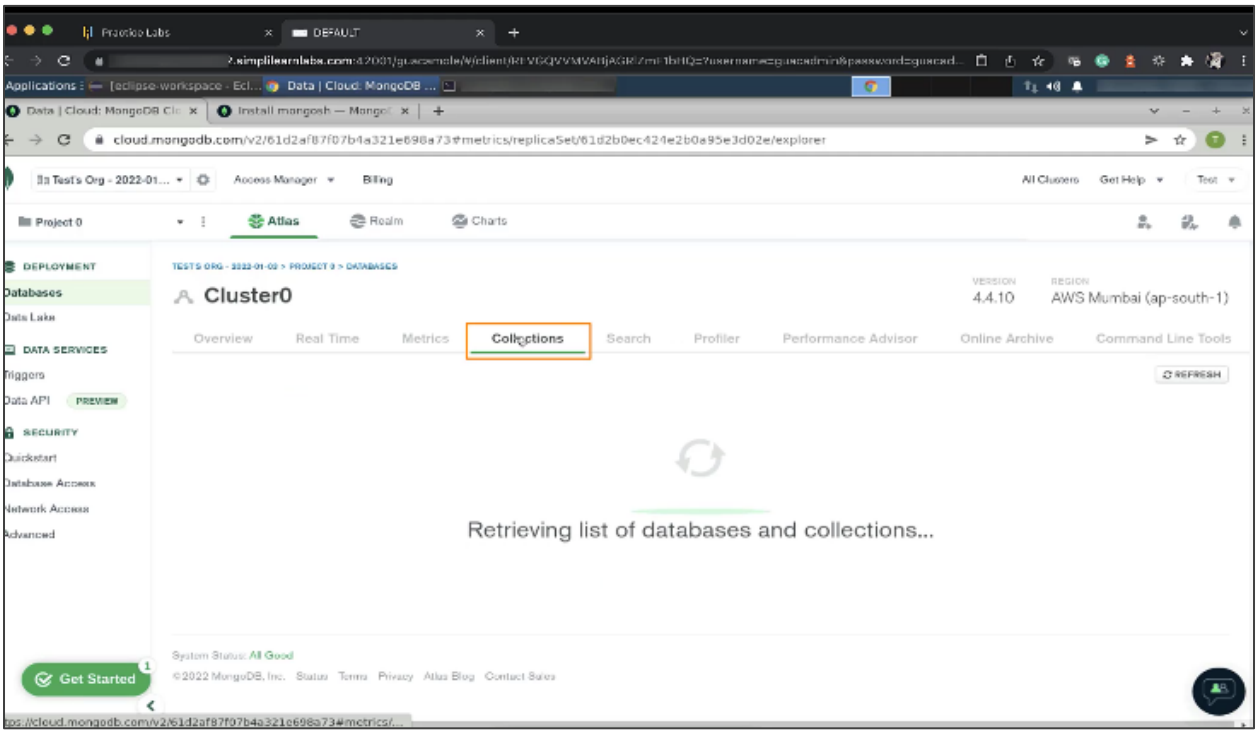
It takes some time to establish the connection; once it is established, you will enter the MongoDB shell.



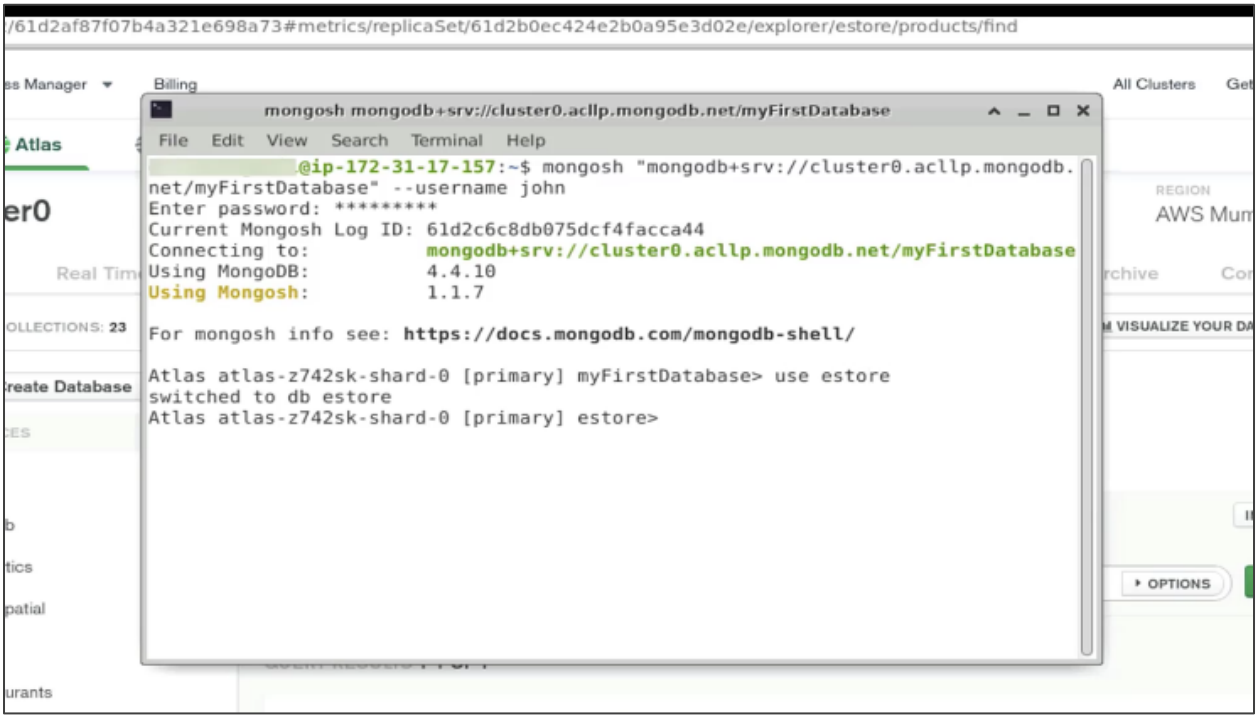
3.3 Select **Cluster0**



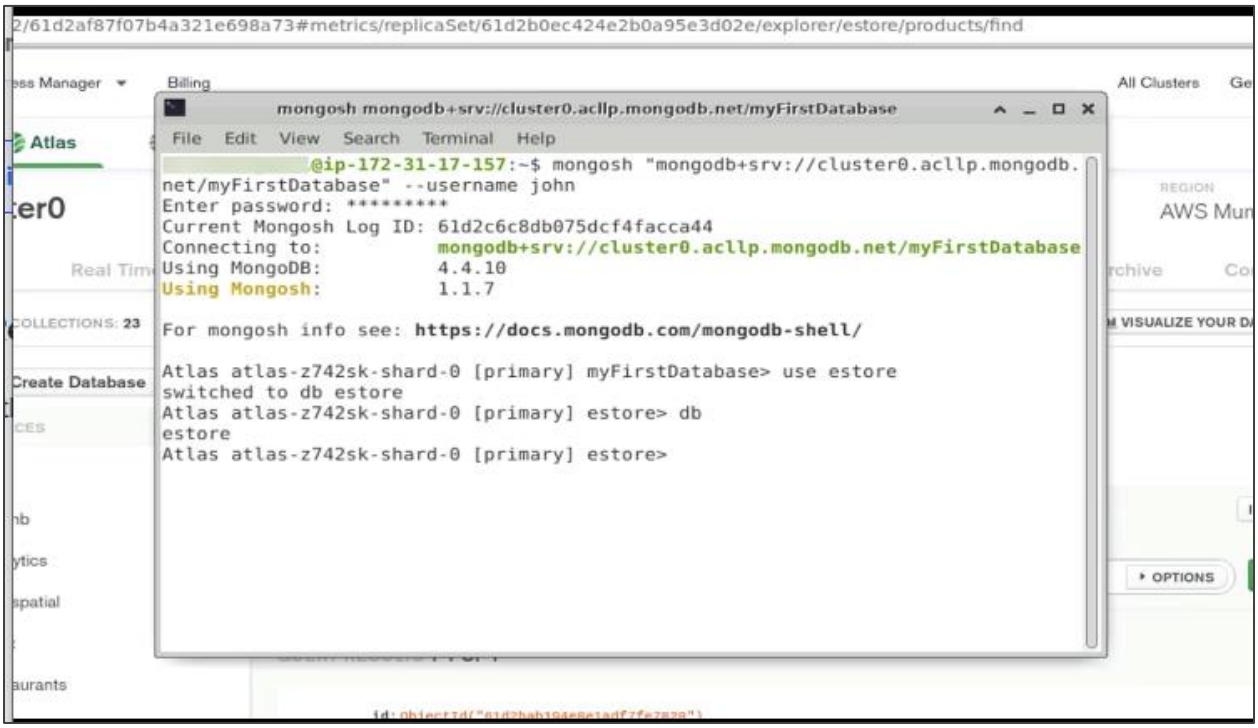
3.4 Navigate to the **Collections** tab



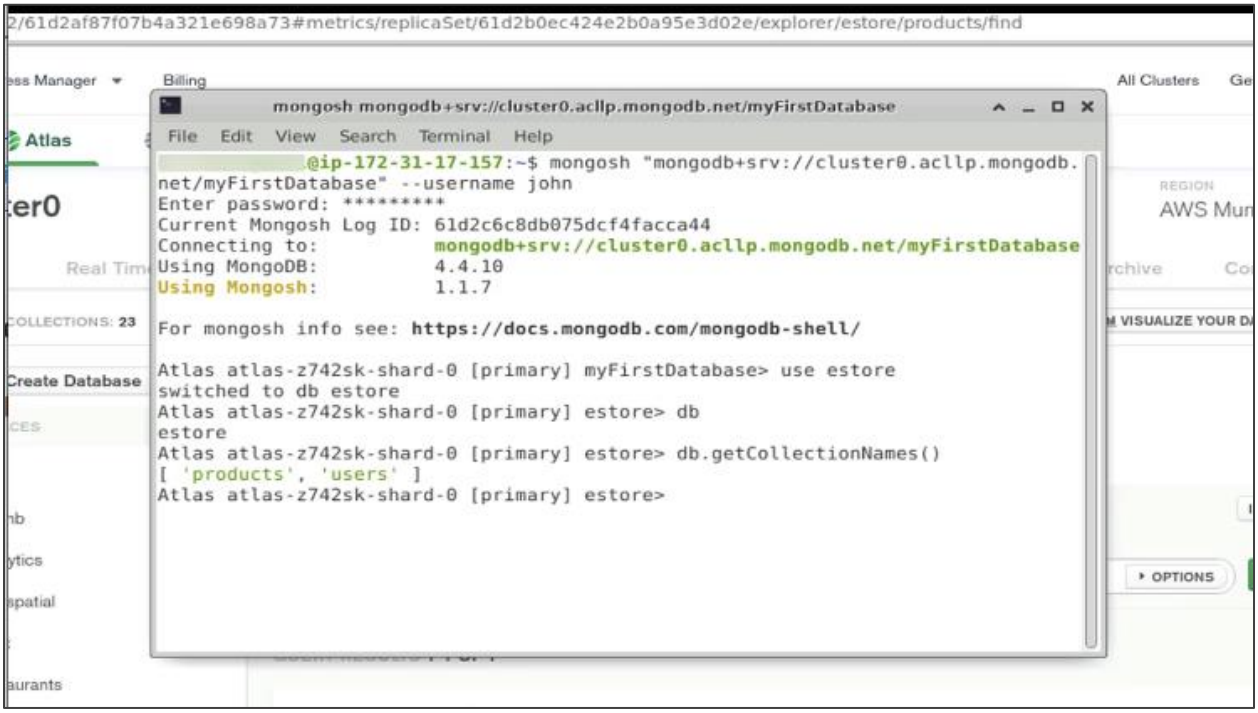
3.5 Enter **use estore** in the terminal window



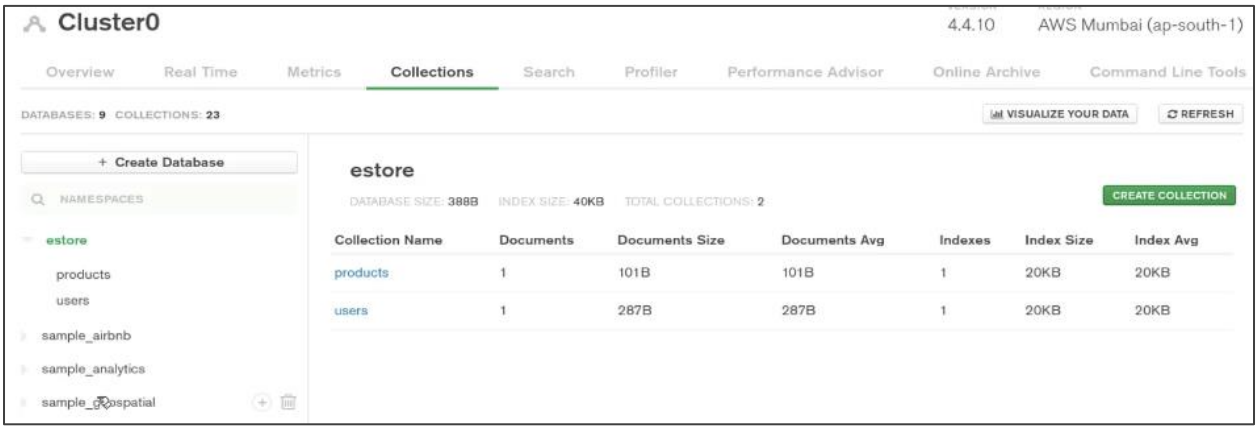
3.6 Enter the **db** command to understand which database you are working on



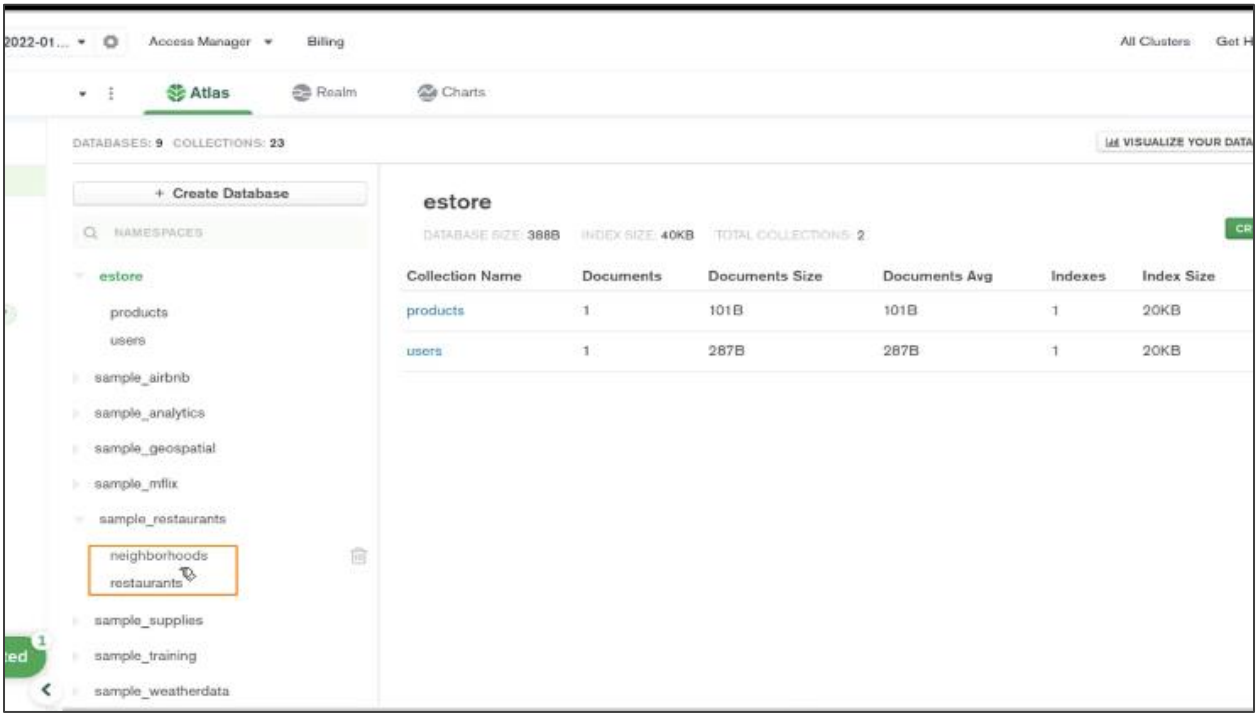
3.7 Get the collection name using the **db.getCollectionNames()** command



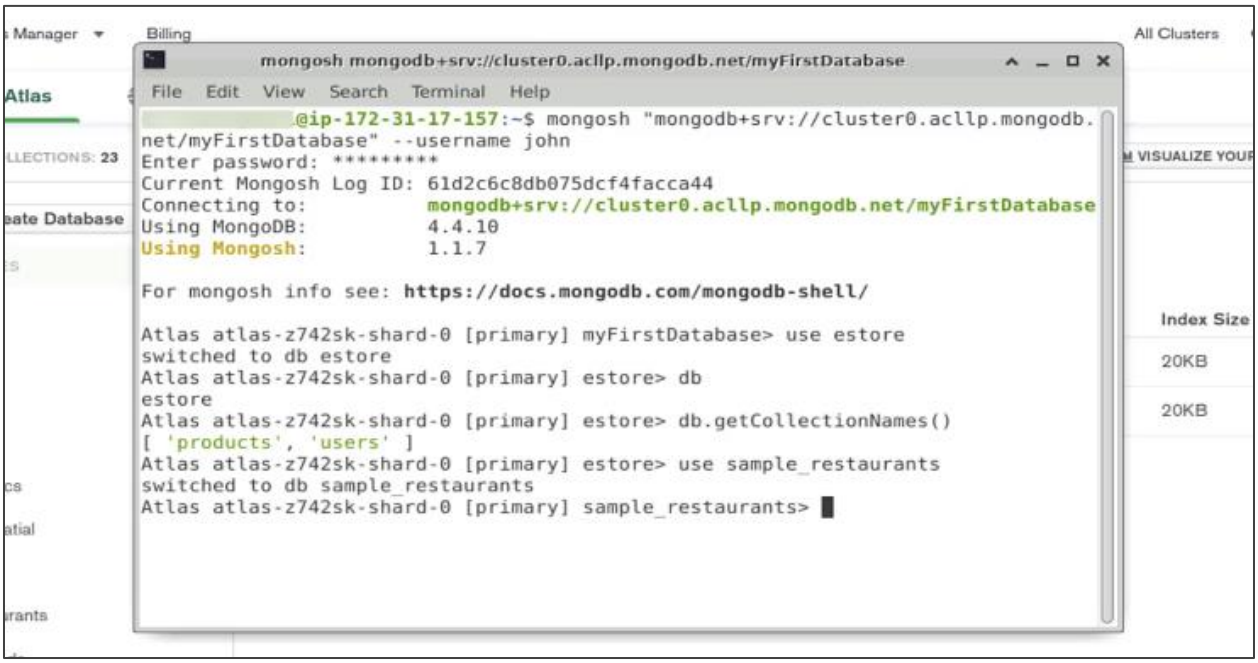
3.8 Open the **estore** collection. You will see the products and users listed.



3.9 Select **Sample_restaurants** from the collections



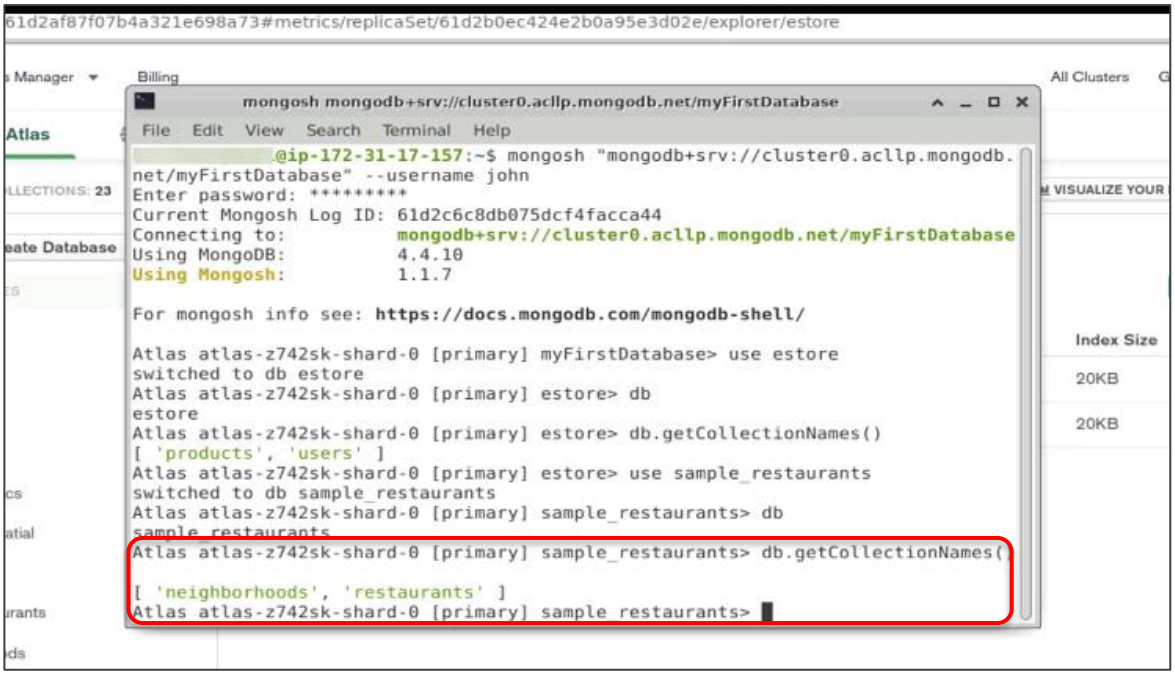
3.10 Type **sample_restaurants** in the terminal to access the **sample_restaurants** database



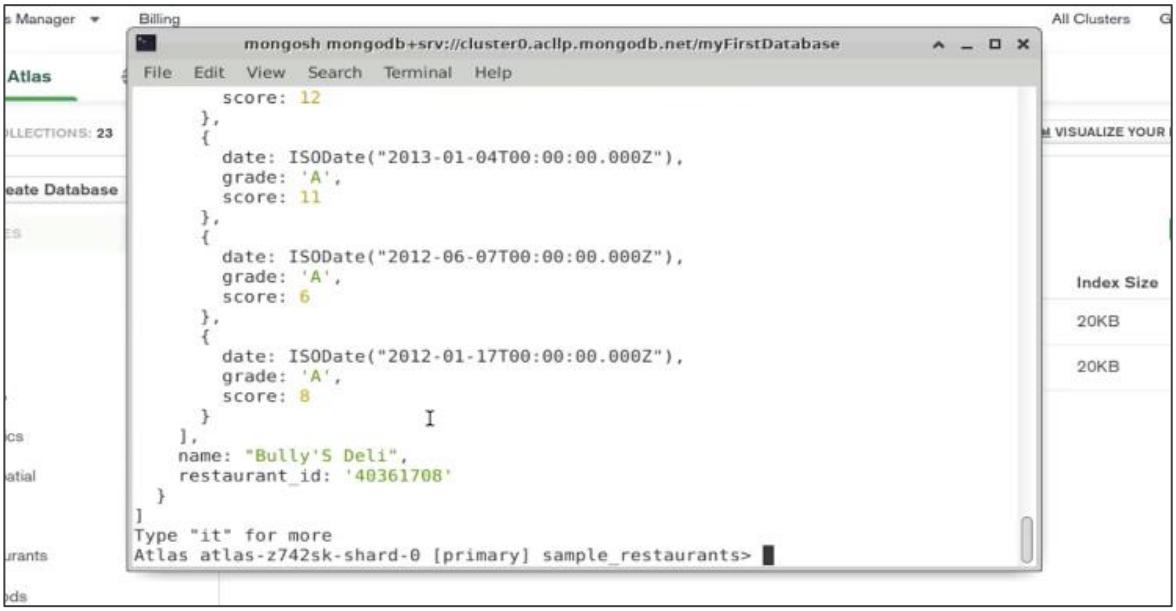
Step 4: Execute commands

4.1 Execute the command below to get the collection names:

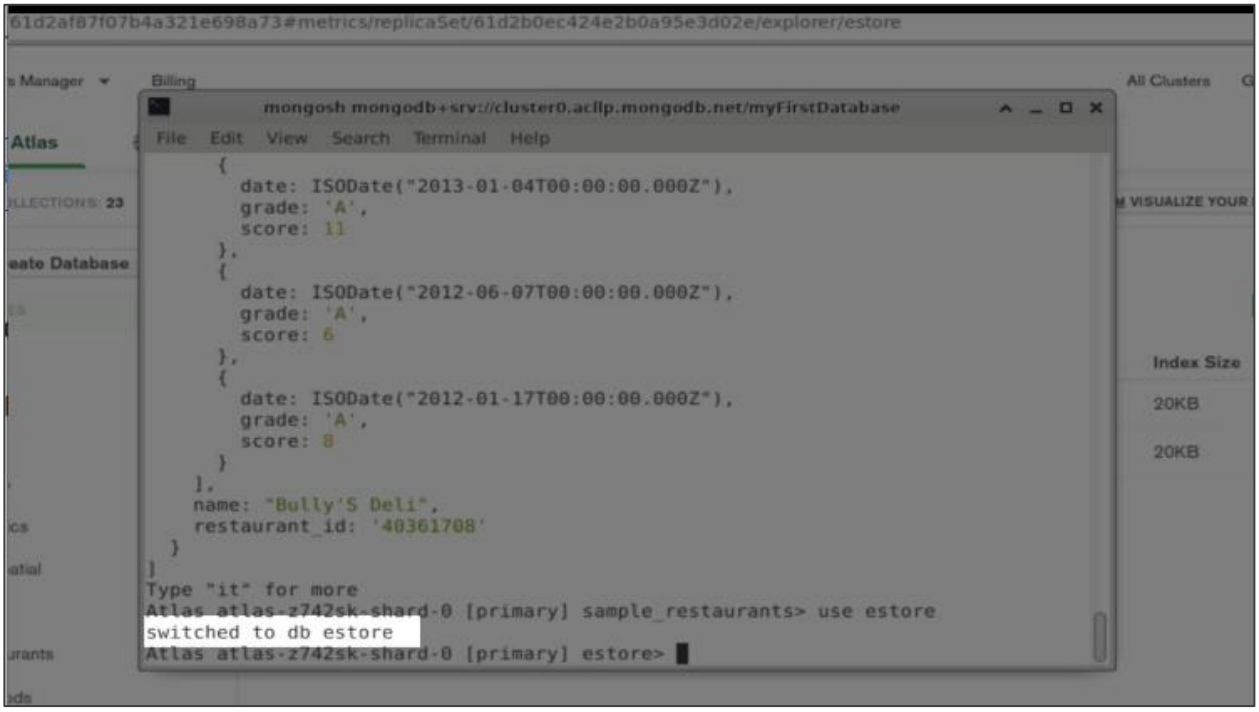
Db.getCollectionNames()



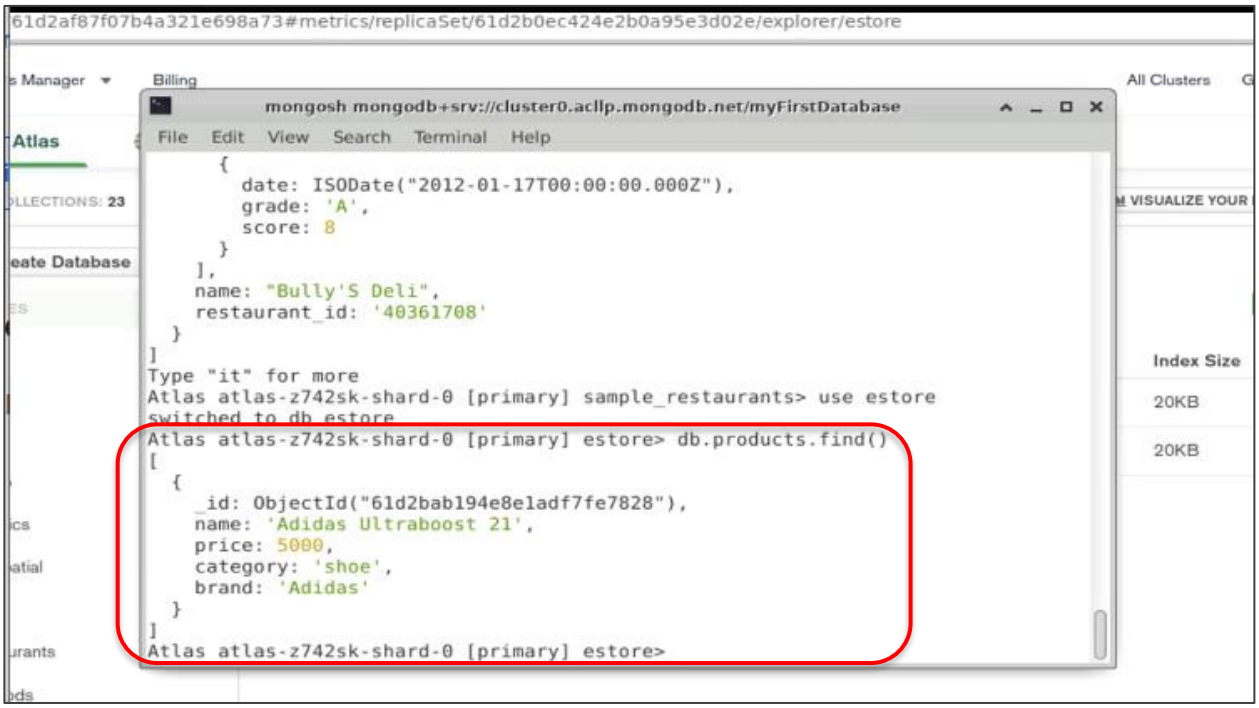
4.2 Execute the **db.restaurants.find()** command to determine the documents inside the collection



4.3 Enter the **estore** command once again and the database will switch to estore

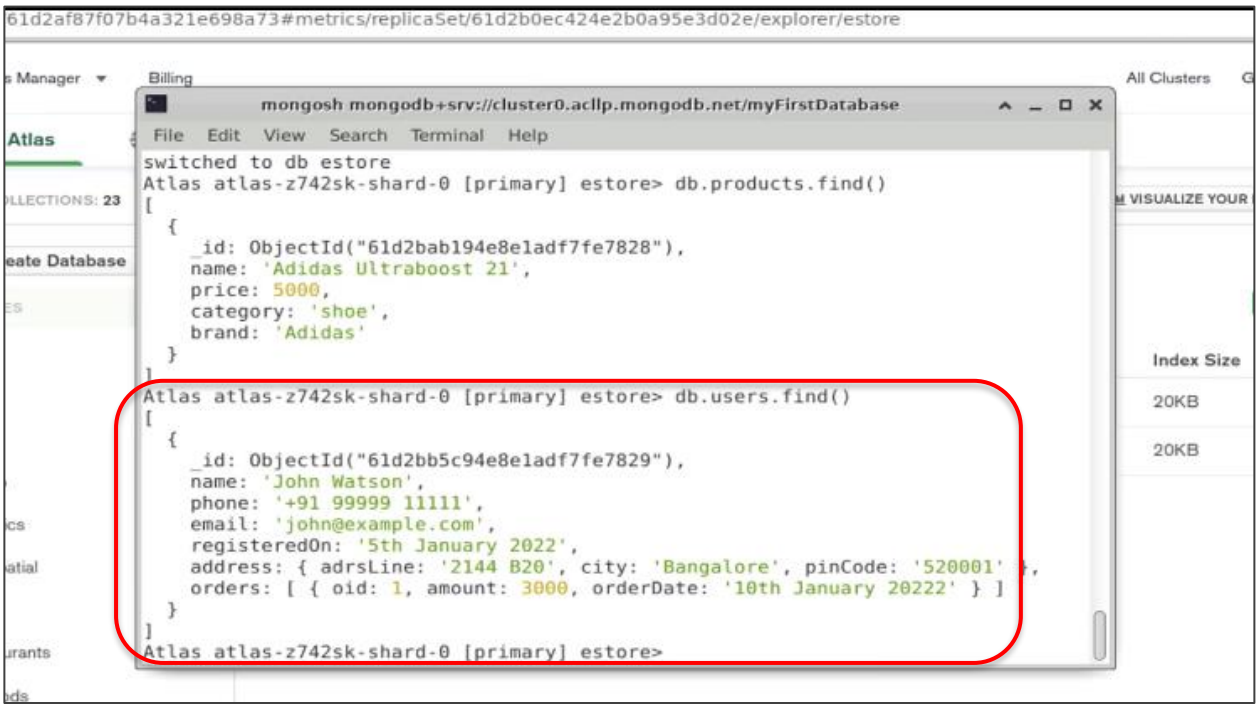


4.4 Use **db.products.find()** to list the products

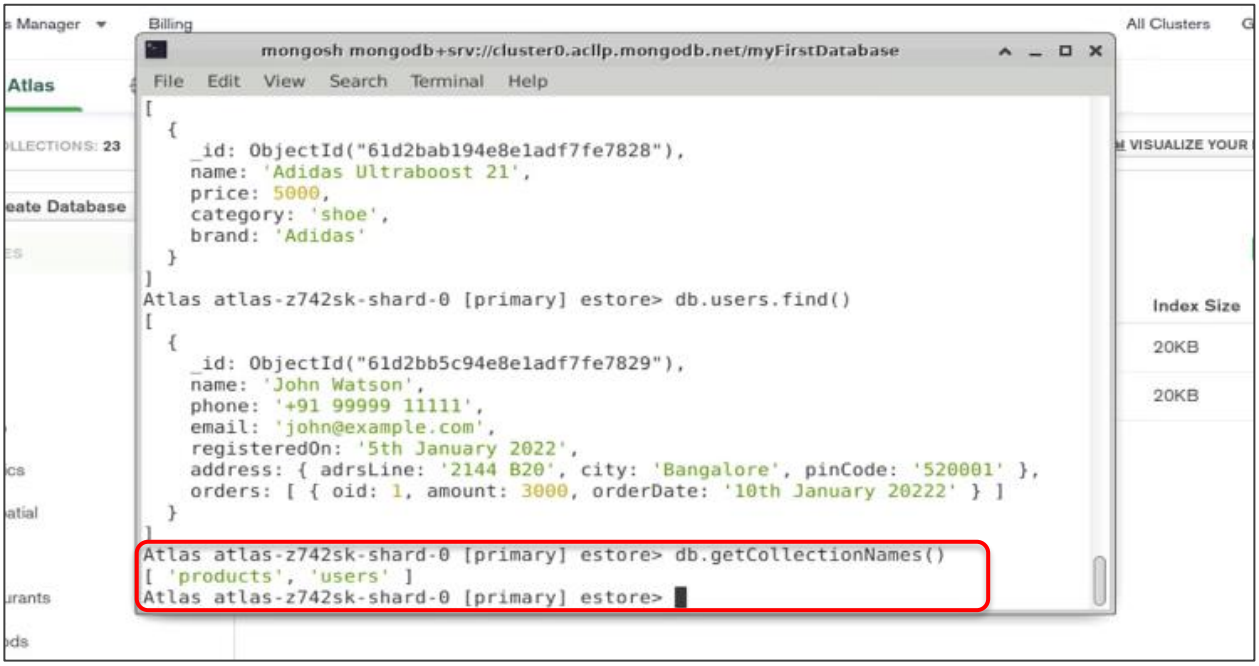


You can see that there is only one product listed in the terminal.

4.5 Similarly, use **db.users.find()** to determine the data

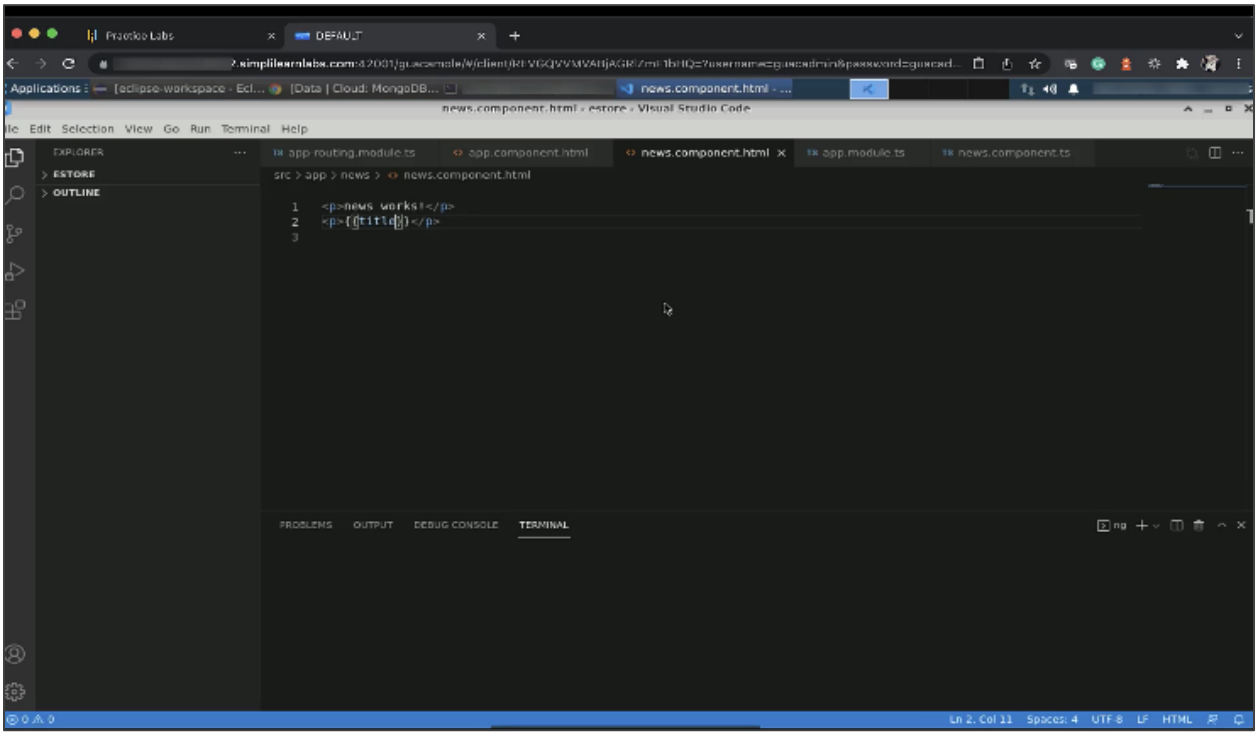


4.6 Get the collection names again using the **db.getCollectionNames()** command

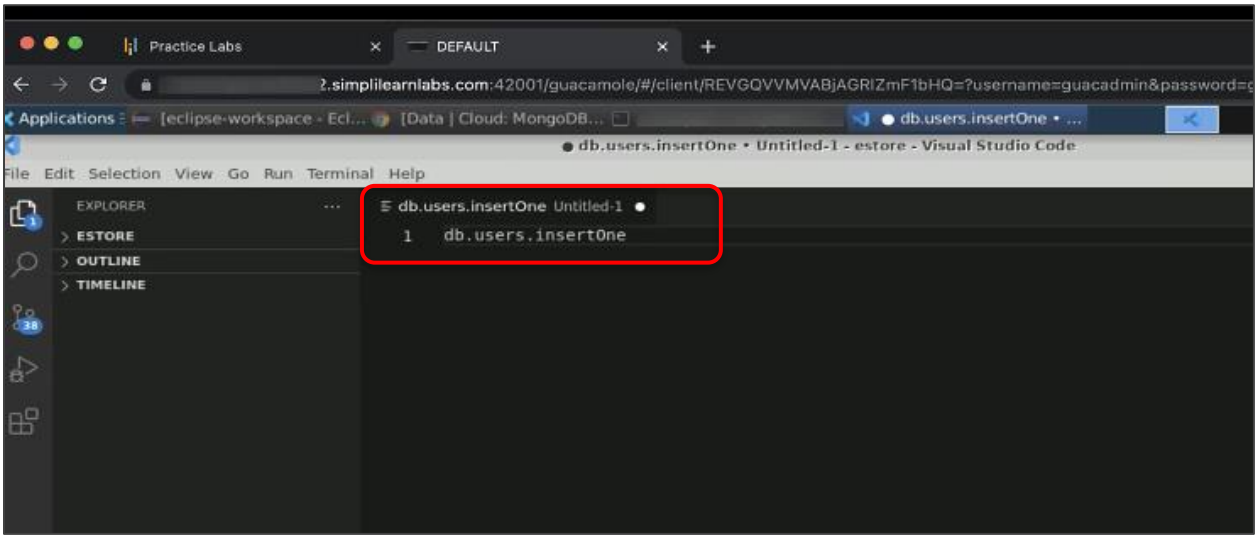


Step 5: Create a document

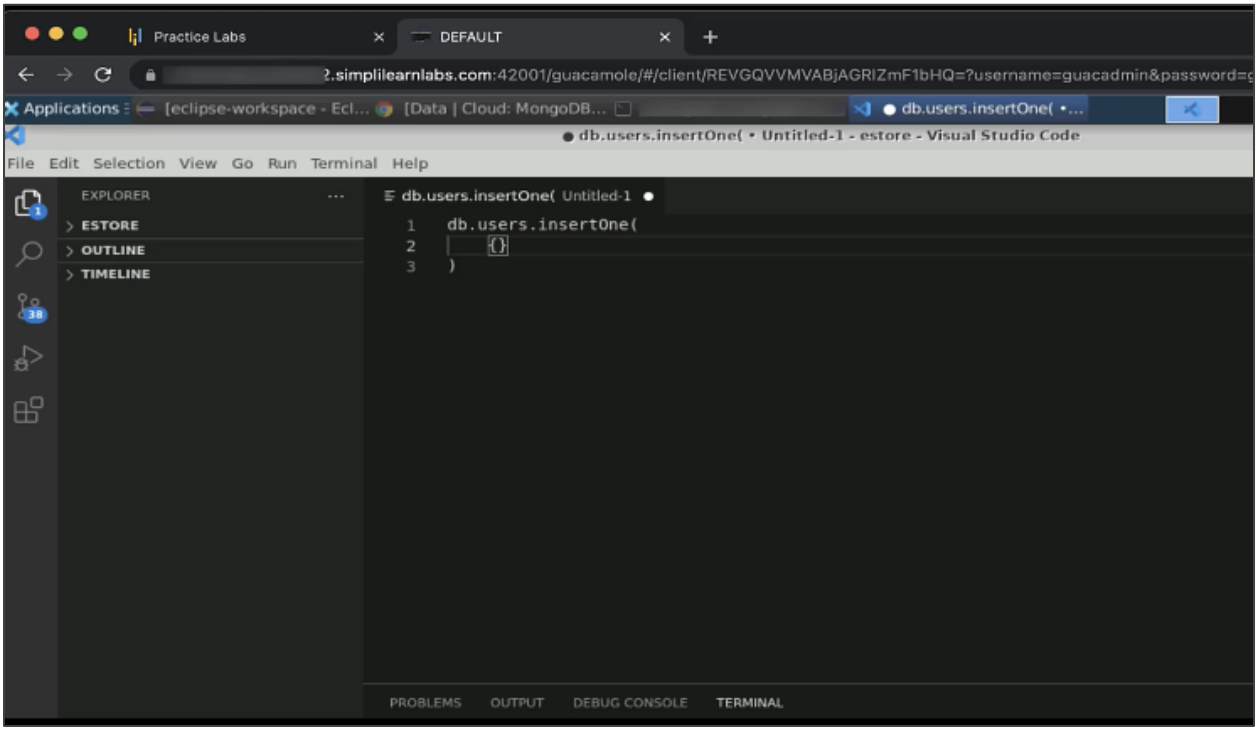
5.1 Open the Visual Studio Code editor



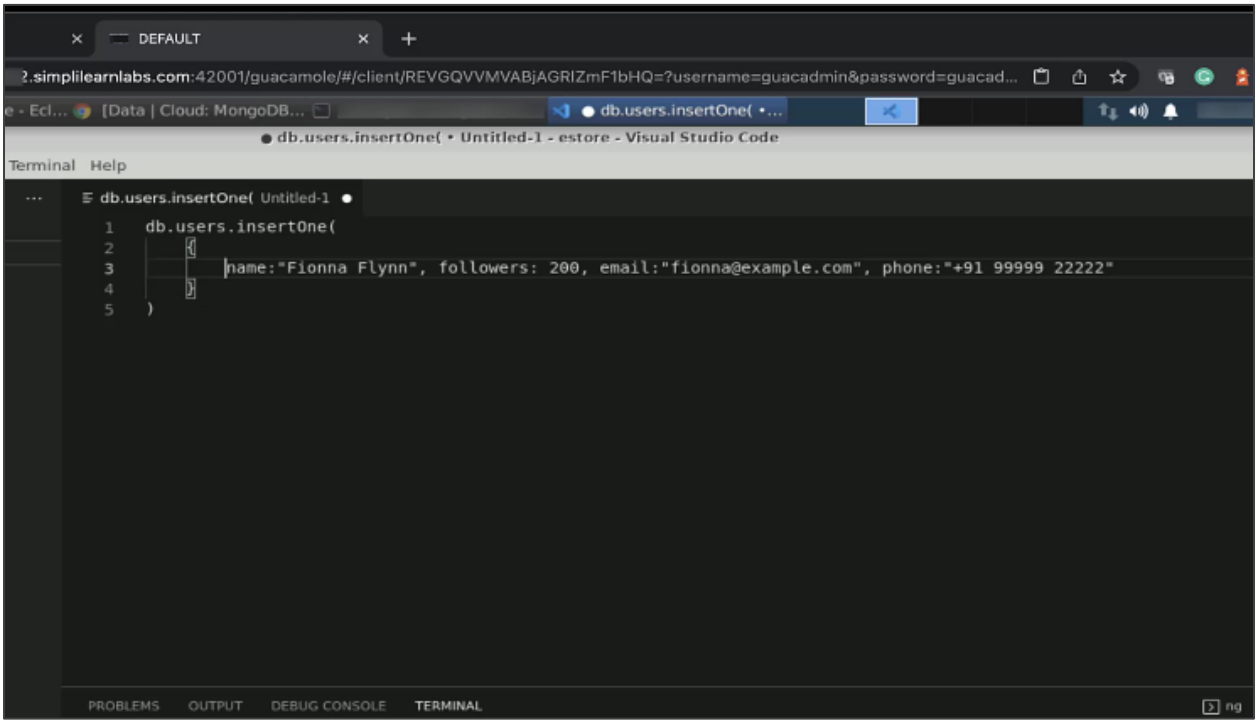
5.2 Create a new file and use the **insertOne** method to insert a document inside the database



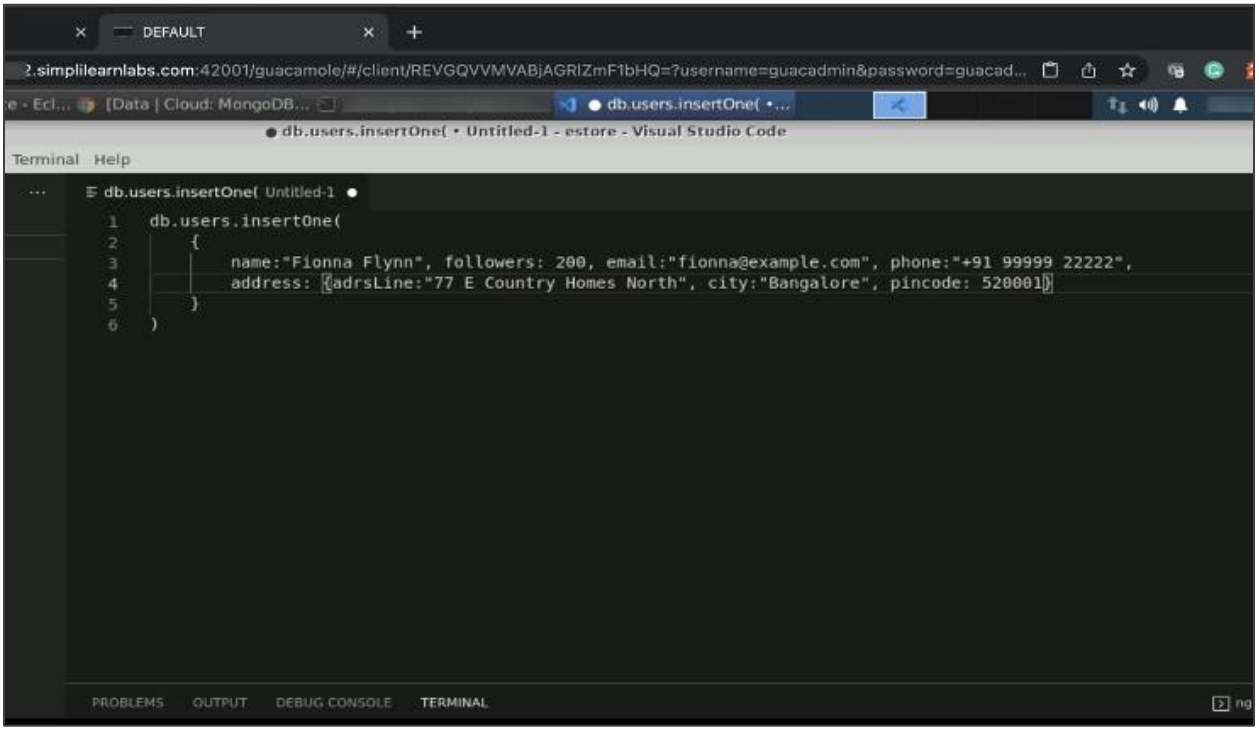
5.3 Create a document as per the JSON standards



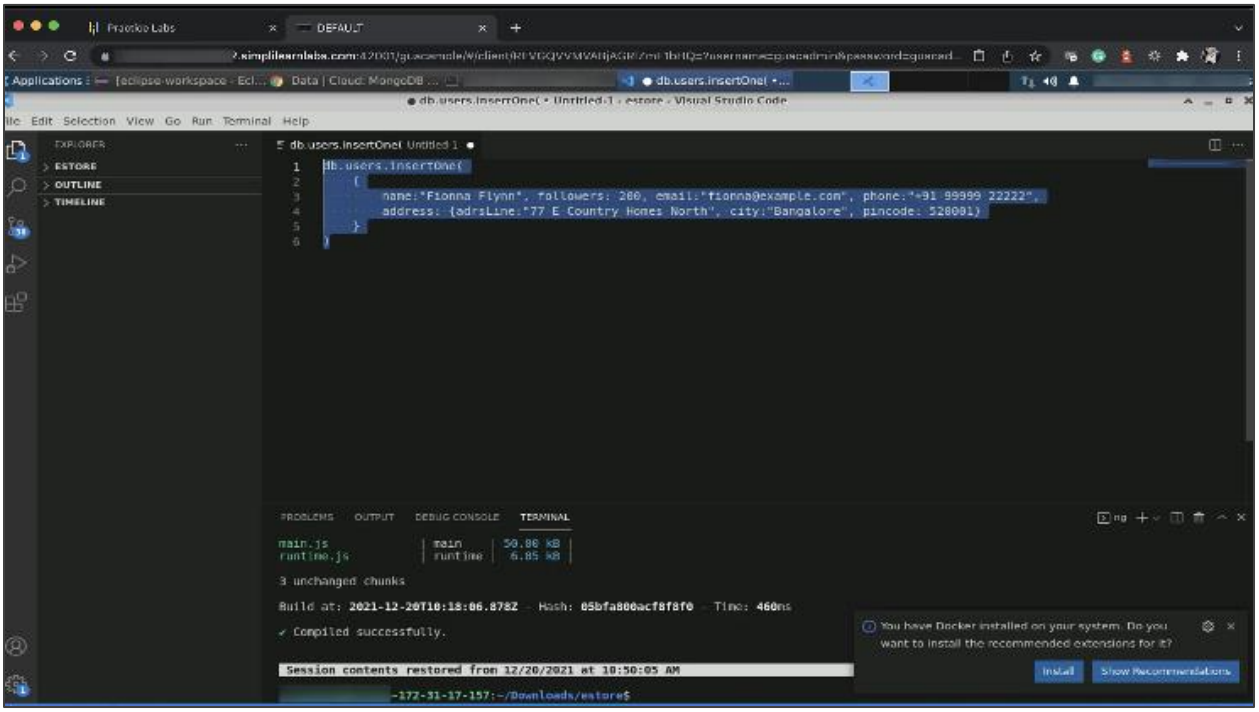
5.4 Give the attributes in key-value pairs, consisting of **name**, **followers**, **email**, and **phone**



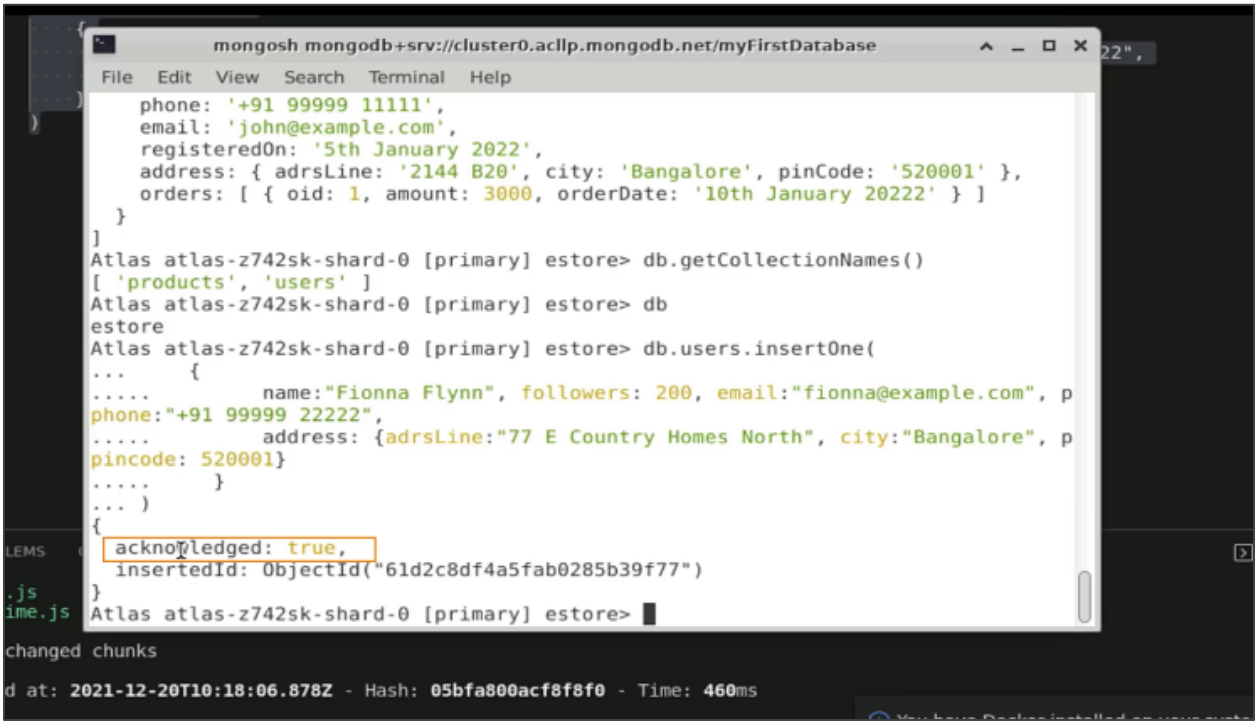
5.5 Add an **address** attribute to represent another object



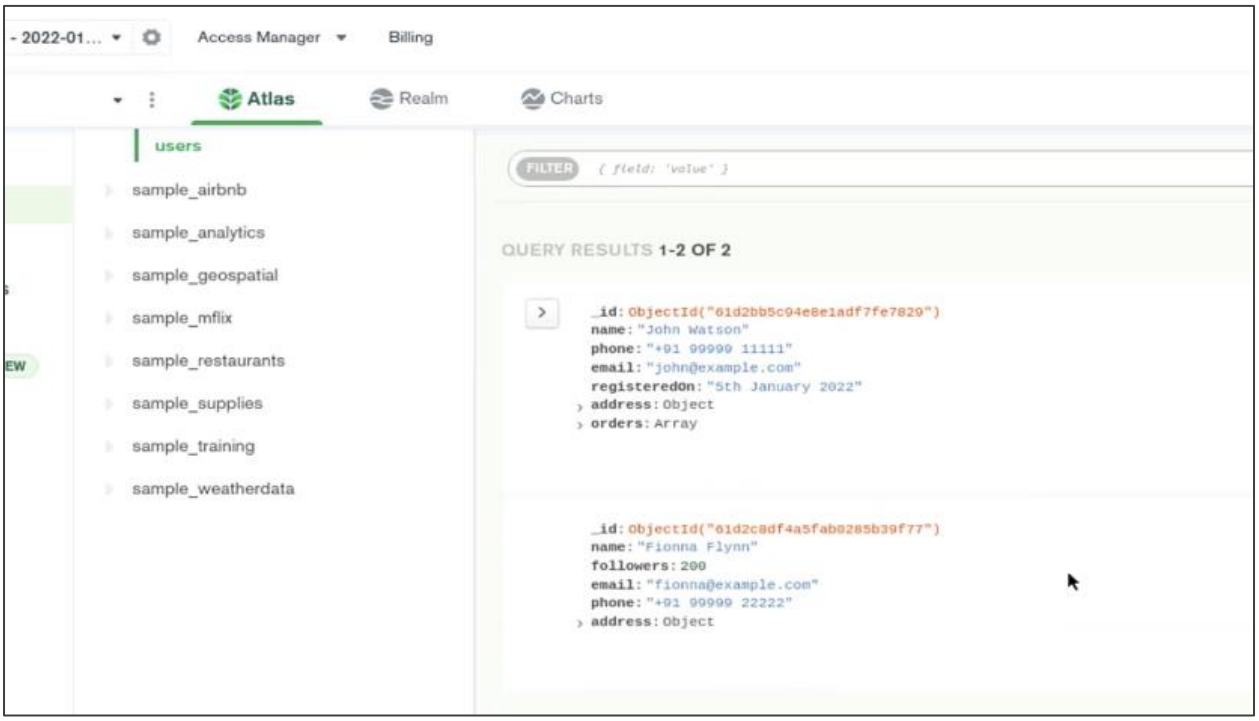
5.6 Copy the snippet



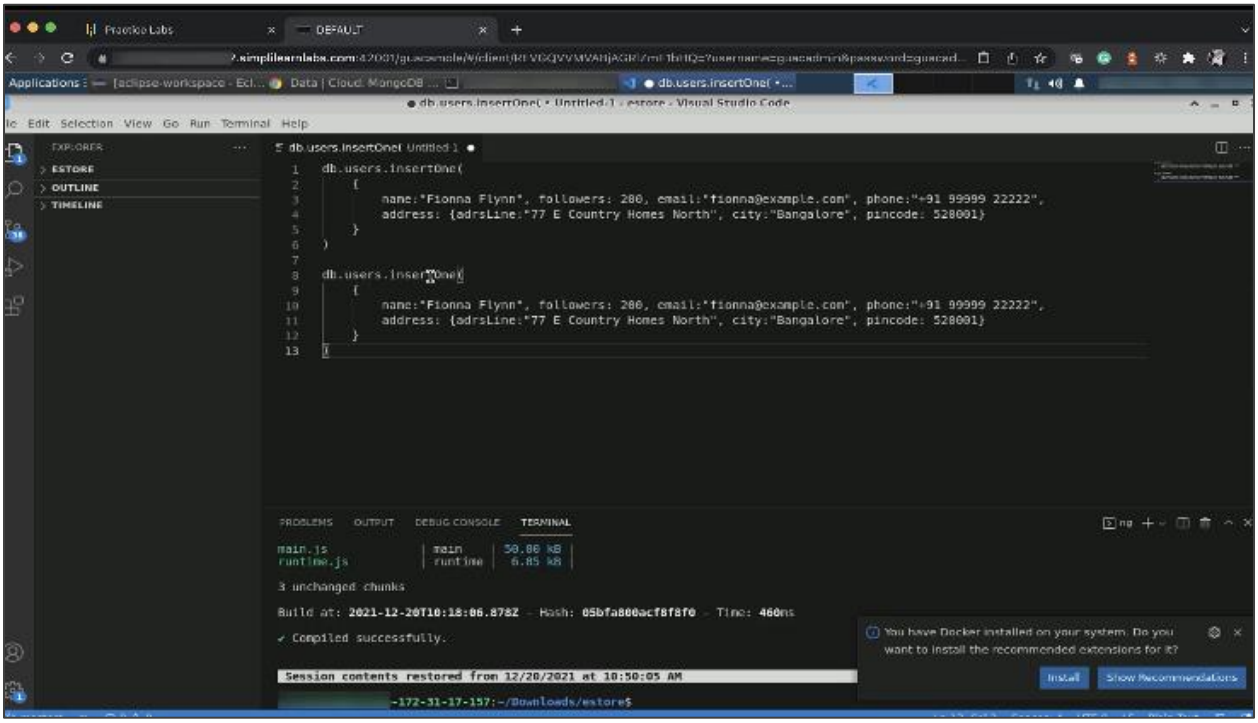
5.7 Paste it into the terminal in **MongoDB Shell** and press **Enter**



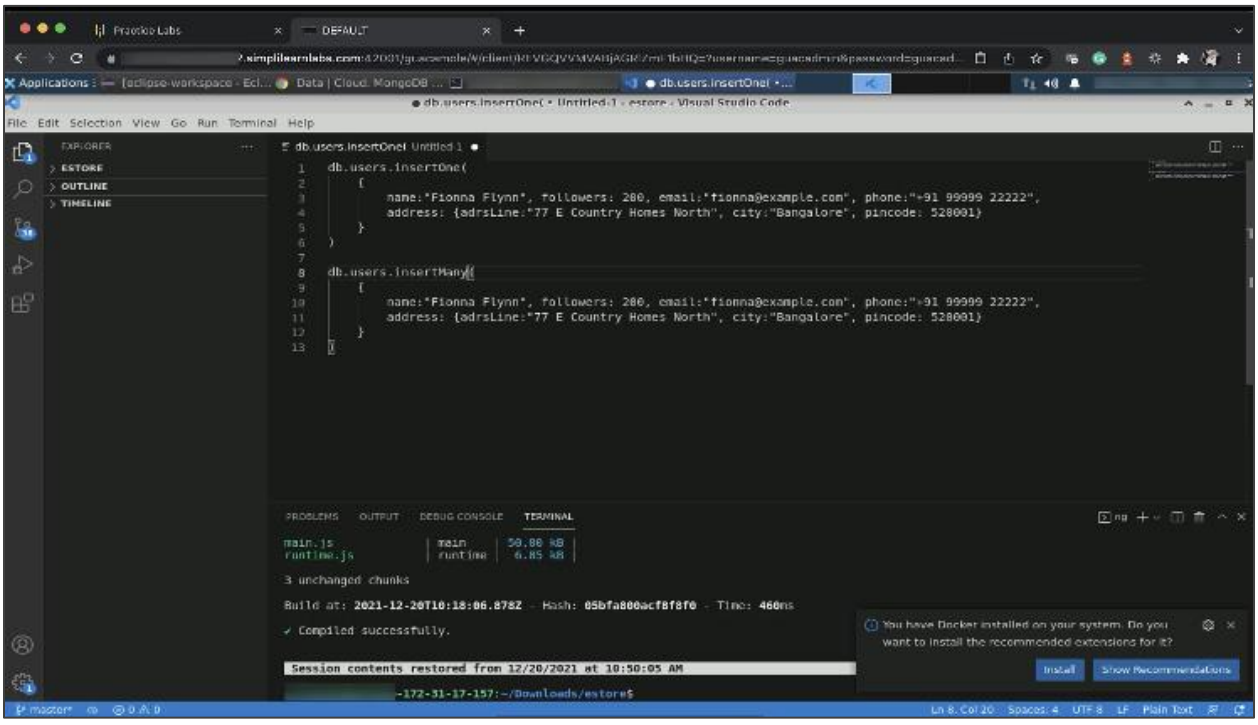
5.8 Refresh the **users** collection. You can see that another document was created.



5.9 Paste the same snippet in Visual Studio Code

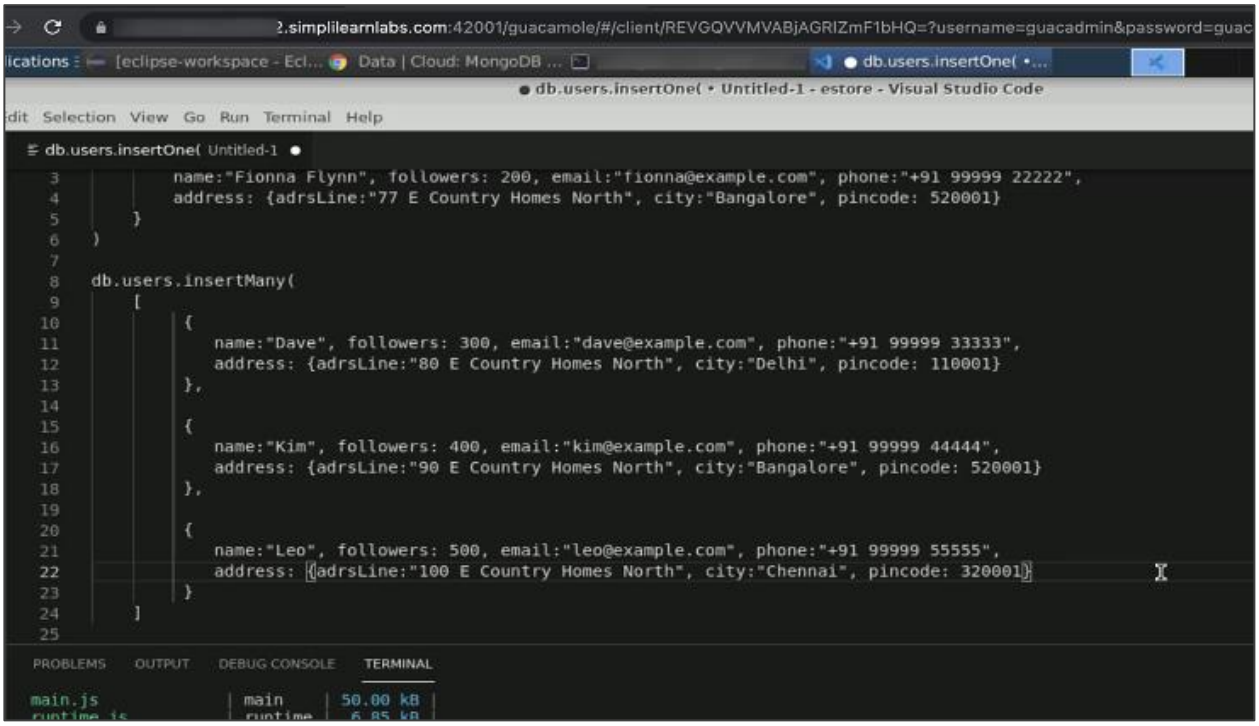


5.10 Use the **db.users.InsertMany()** method to pass an array of documents

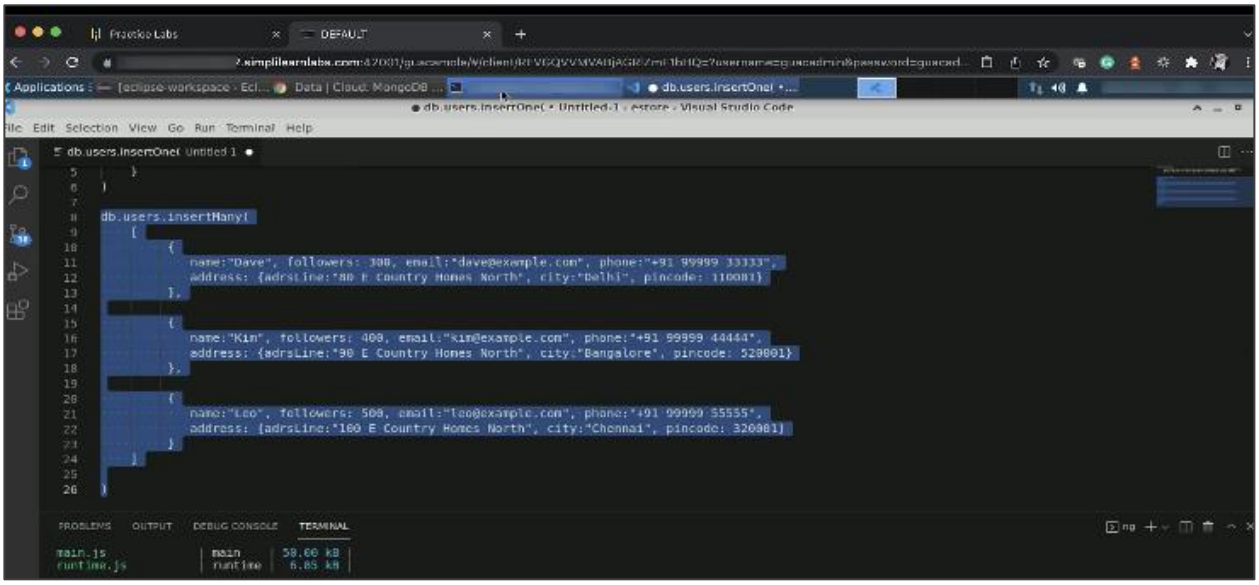


Step 6: Insert more documents

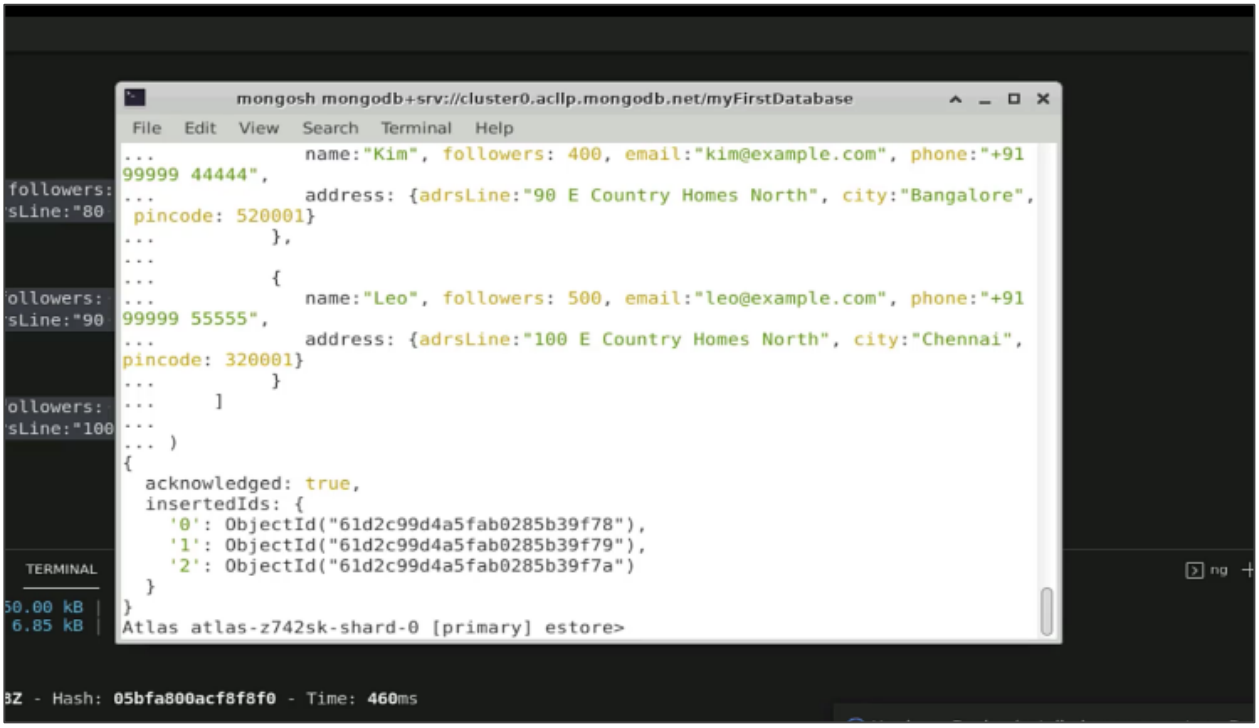
6.1 Insert some more documents with different values



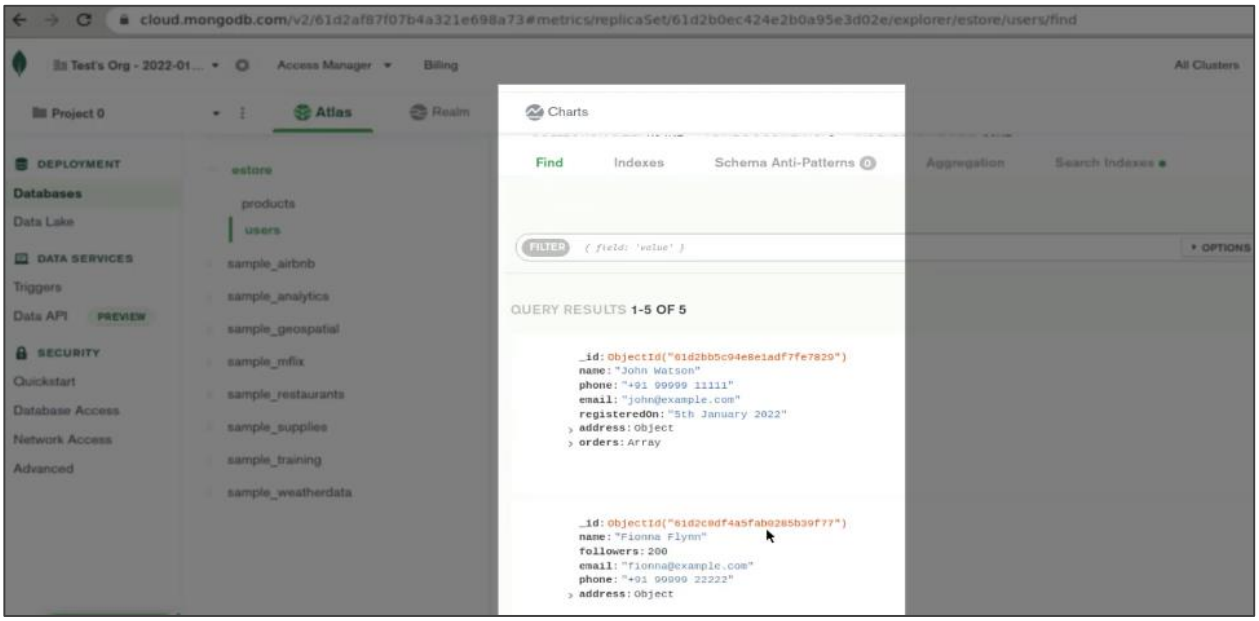
6.2 Copy the code



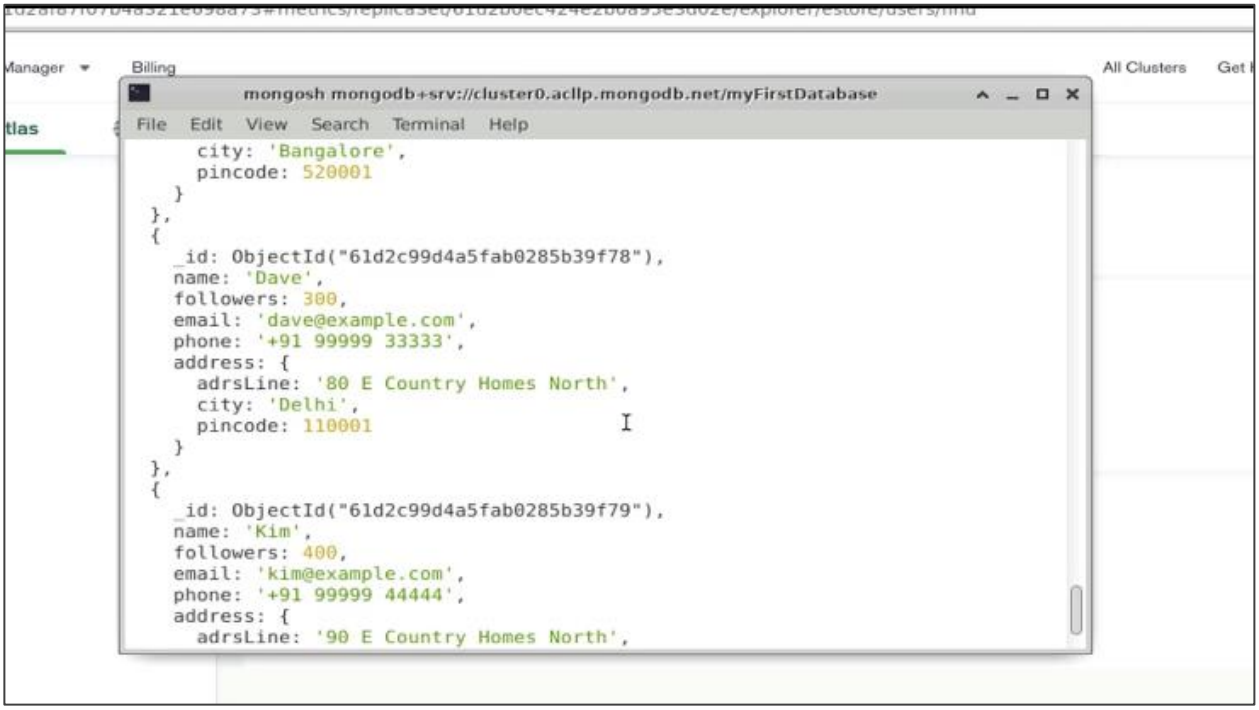
6.3 Paste it into the terminal and execute it



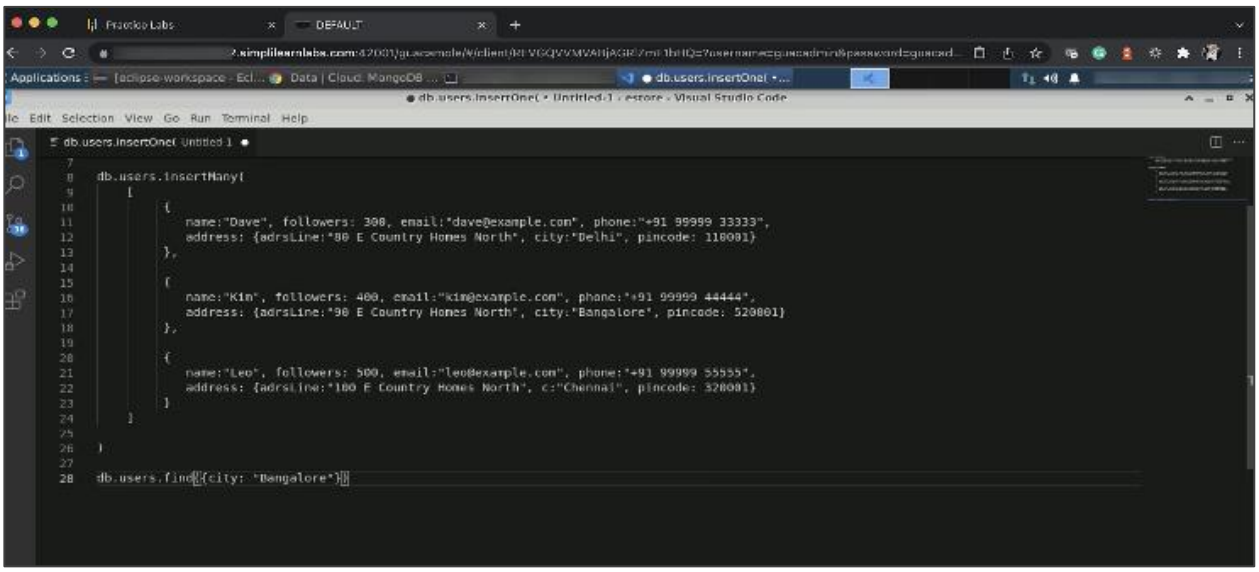
6.4 Refresh the **users** collection page. You can see more documents updated in the database.



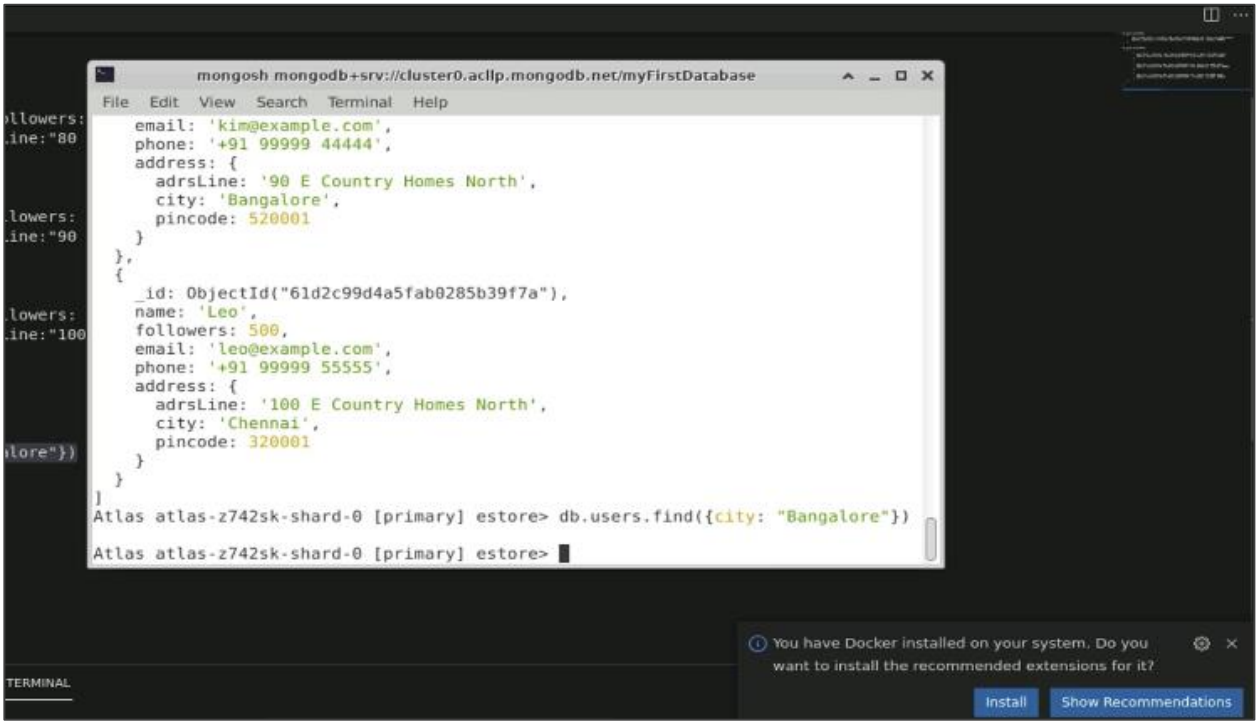
6.5 Now, execute the find function on users using the **db.users.find()** function



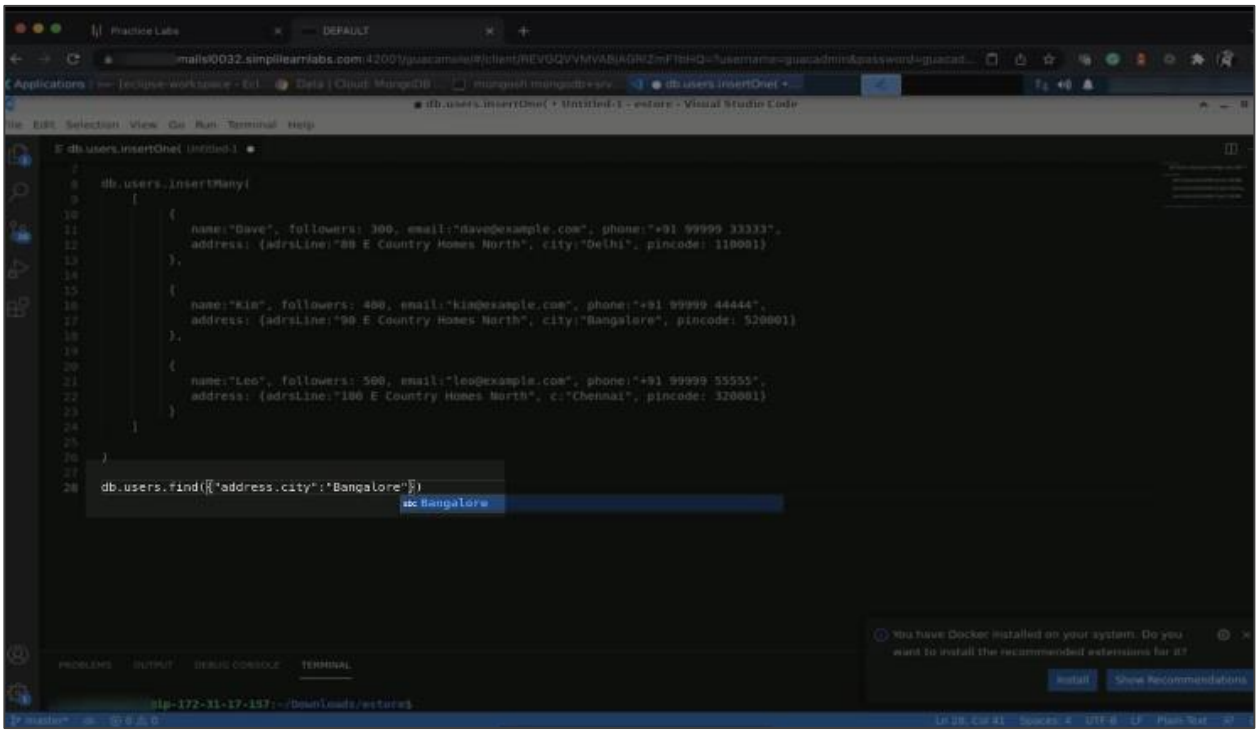
6.6 Use the **db.users.find(city: "Bangalore")** query to find users with the city as Bengaluru



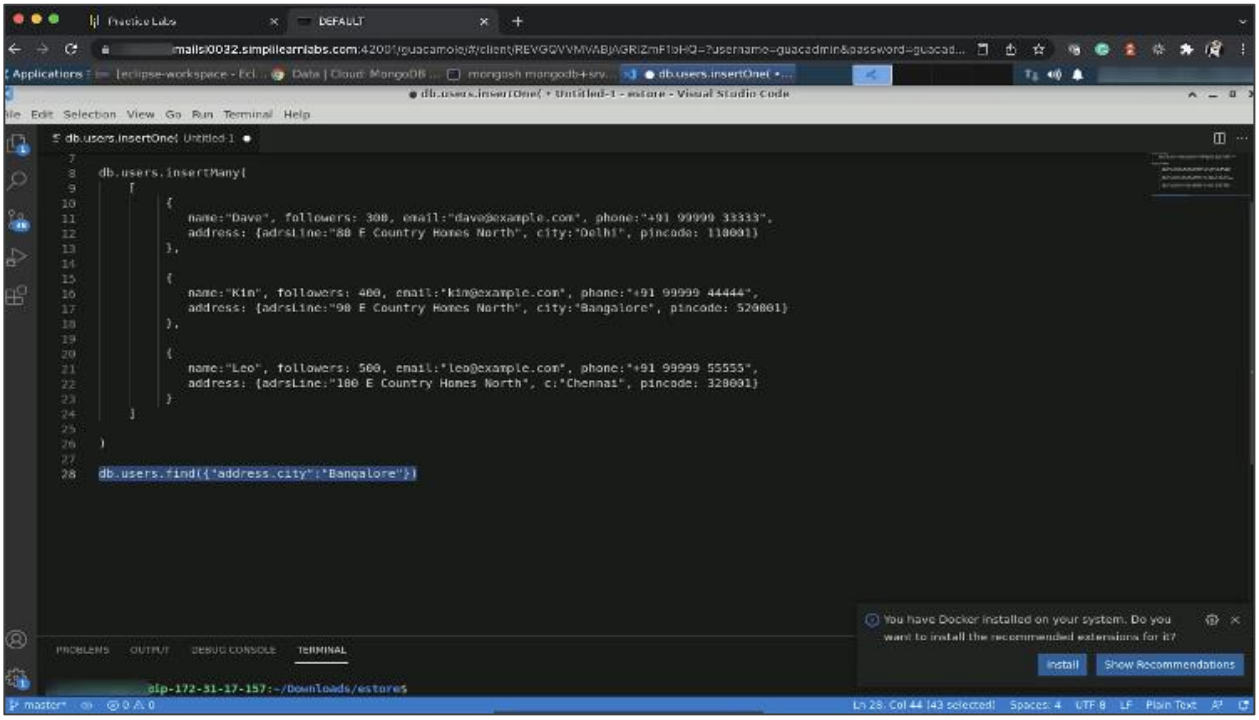
6.7 Execute the query in the terminal



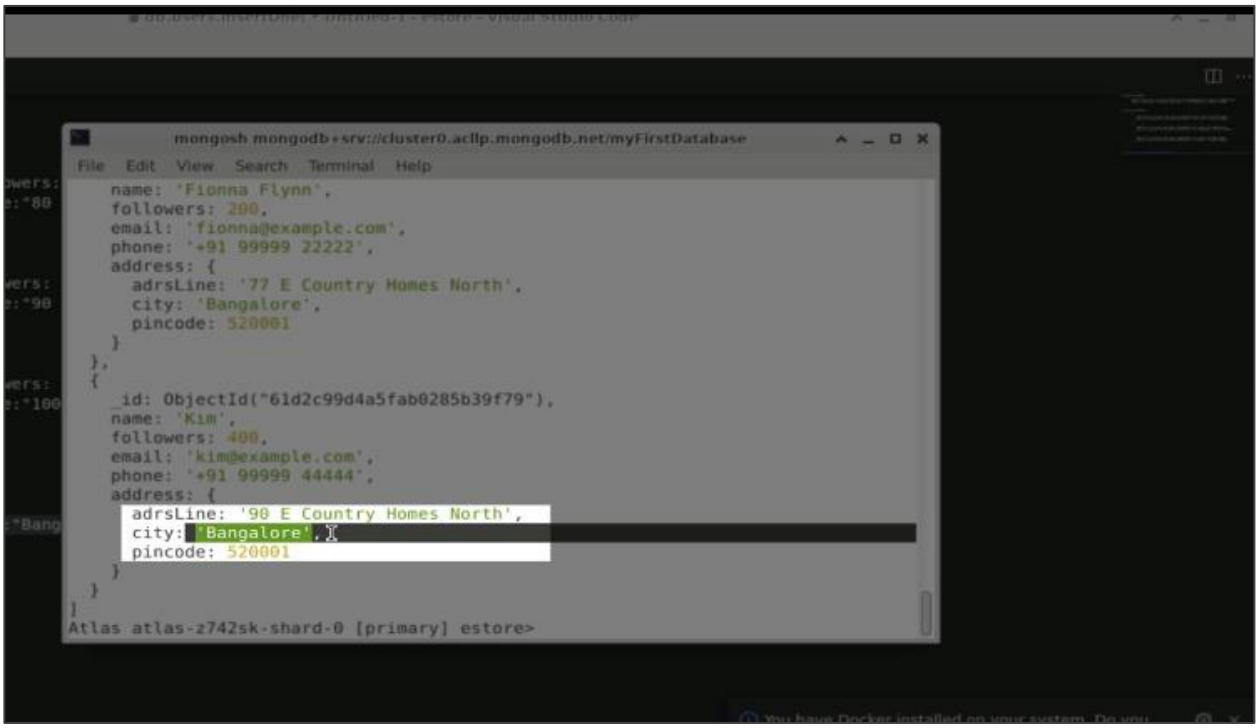
6.8 Change the key value to **address.city**



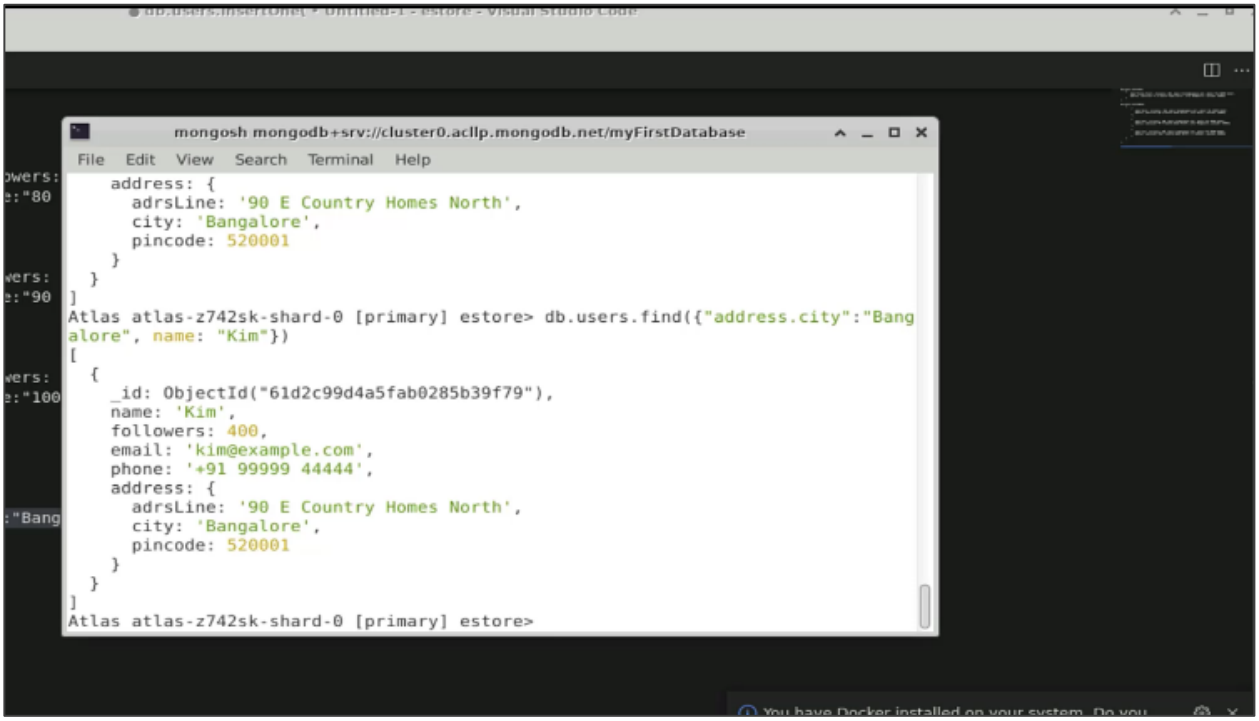
6.9 Copy the query



6.10 Paste it into the terminal

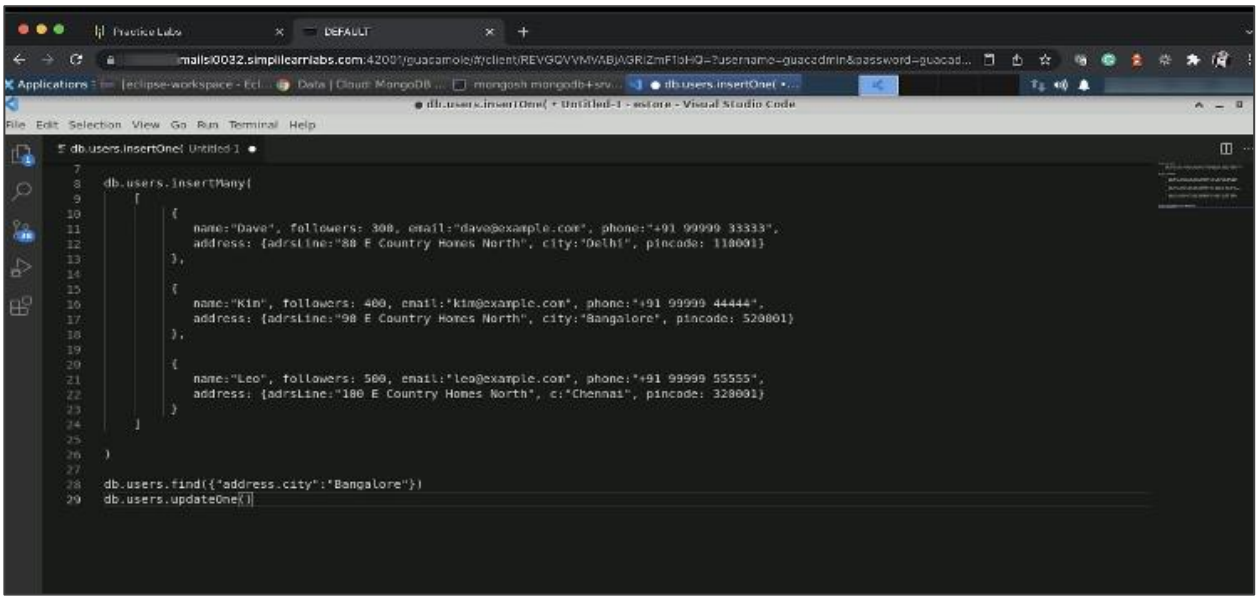


6.11 Add a find statement to list the results of user **Kim** in **Bengaluru**



Step 7: Perform update operations

7.1 Perform the update operation using the **updateOne()** function



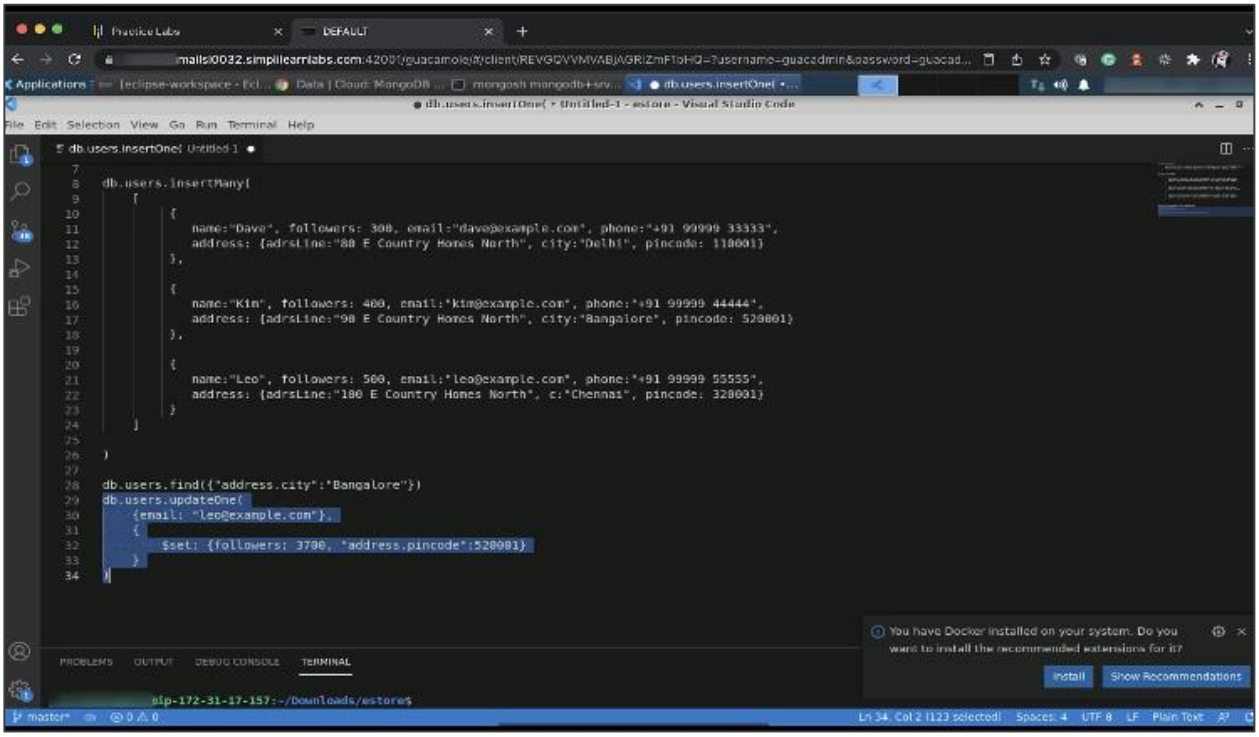
7.2 Enter conditions to update, such as **email**

```
db.users.insertOne( Untitled-1 ●
 7
 8  db.users.insertMany(
 9      [
10          {
11             name:"Dave", followers: 300, email:"dave@example.com", phone:"+91 99999 33333",
12             address: {adrsLine:"80 E Country Homes North", city:"Delhi", pincode: 110001}
13          },
14
15          {
16             name:"Kim", followers: 400, email:"kim@example.com", phone:"+91 99999 44444",
17             address: {adrsLine:"90 E Country Homes North", city:"Bangalore", pincode: 520001}
18          },
19
20          {
21             name:"Leo", followers: 500, email:"leo@example.com", phone:"+91 99999 55555",
22             address: {adrsLine:"100 E Country Homes North", c:"Chennai", pincode: 320001}
23          }
24      ]
25
26  )
27
28  db.users.find({"address.city":"Bangalore"})
29  db.users.updateOne(
30      {email: "leo@example.com"}
31  )
```

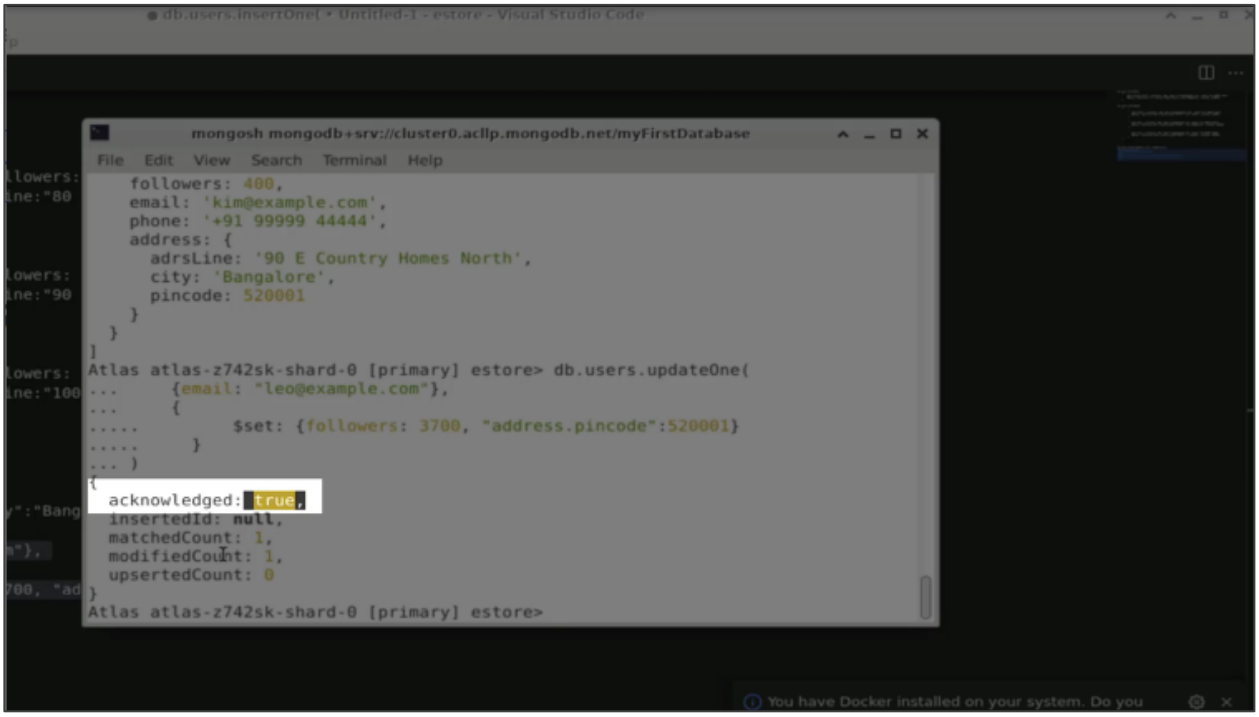
7.3 Set the details to update using the **set** variable that specifies the followers and pincode (zip code)

```
db.users.insertOne( Untitled-1 ●
 7
 8  db.users.insertMany(
 9      [
10          {
11             name:"Dave", followers: 300, email:"dave@example.com", phone:"+91 99999 33333",
12             address: {adrsLine:"80 E Country Homes North", city:"Delhi", pincode: 110001}
13          },
14
15          {
16             name:"Kim", followers: 400, email:"kim@example.com", phone:"+91 99999 44444",
17             address: {adrsLine:"90 E Country Homes North", city:"Bangalore", pincode: 520001}
18          },
19
20          {
21             name:"Leo", followers: 500, email:"leo@example.com", phone:"+91 99999 55555",
22             address: {adrsLine:"100 E Country Homes North", c:"Chennai", pincode: 320001}
23          }
24      ]
25
26  )
27
28  db.users.find({"address.city":"Bangalore"})
29  db.users.updateOne(
30      {email: "leo@example.com"}
31  )
```

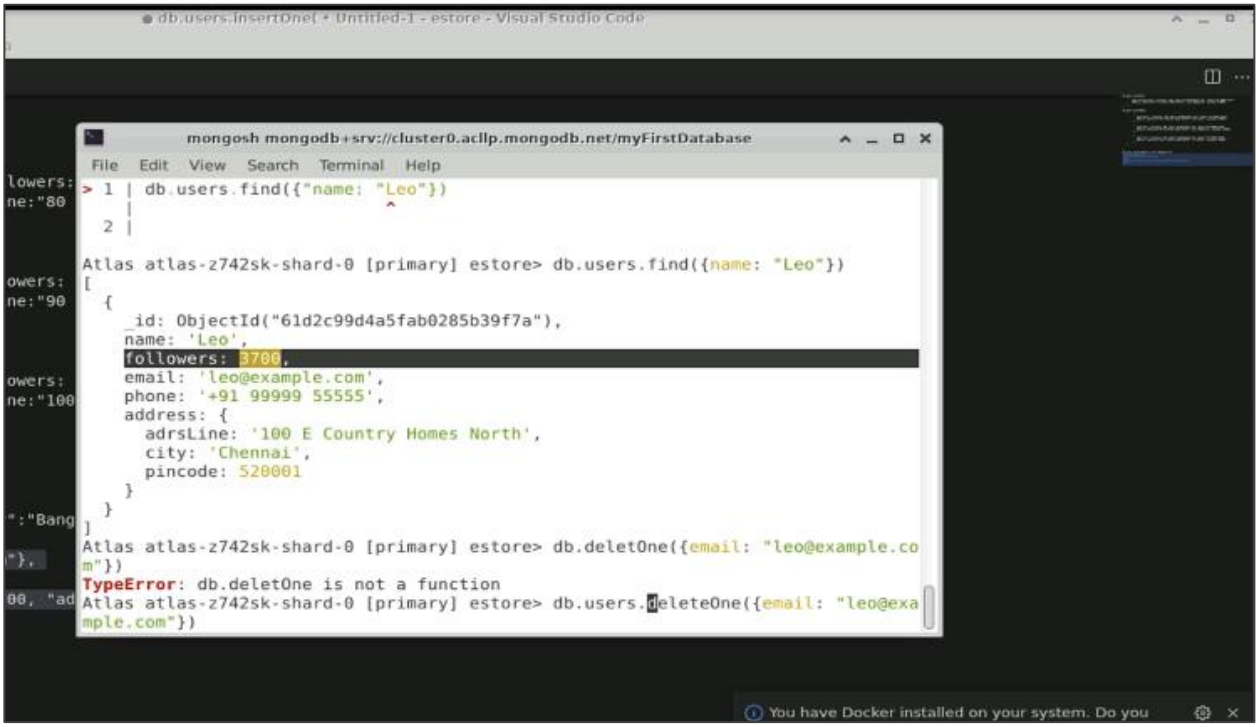
7.4 Copy the snippet



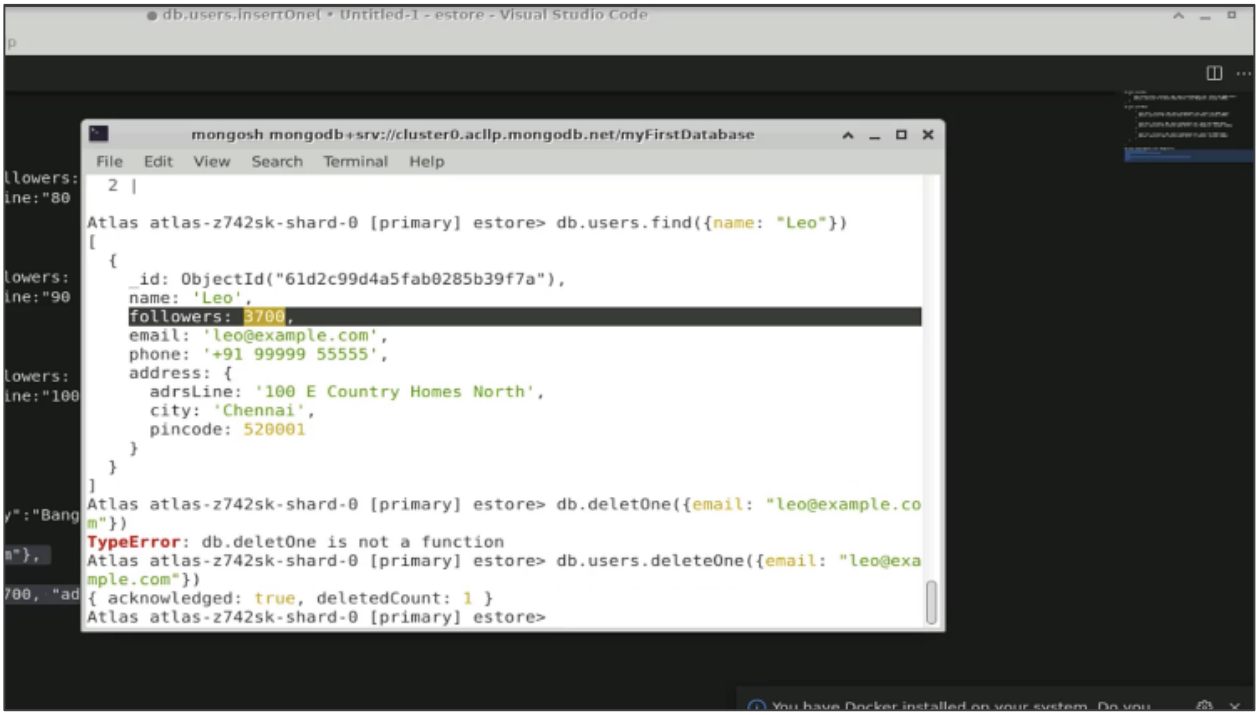
7.5 Paste it and press **Enter.** You can see that the acknowledgment is **true.**



7.6 Write **db.deleteOne()** and add a condition to delete one of the documents

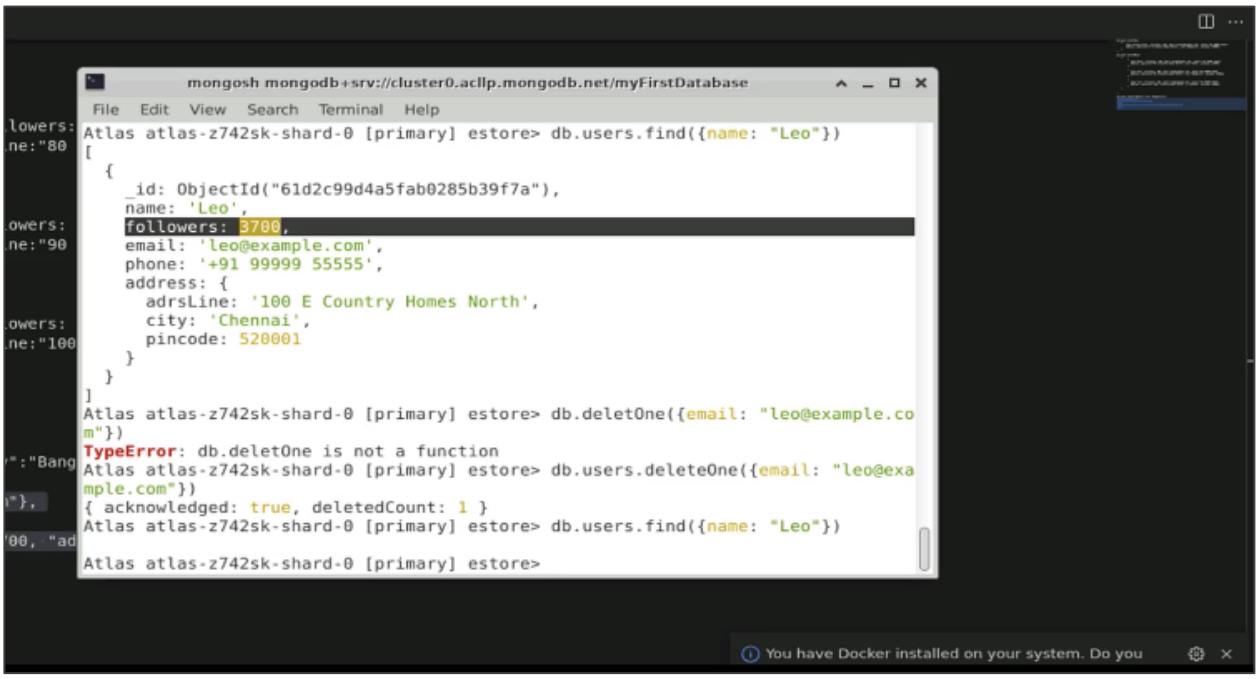


7.7 Update the query as **db.users.deleteOne()** and press **Enter**



You can see that the acknowledgment is **true** and one of the documents is deleted.

7.8 Execute another command named **Leo**



No such document is available.

These CRUD operations can be executed and will remain the same for any API. This applies to programming languages as well, for which these methods and functions remain the same.

By following these steps, you have successfully performed CRUD operations with Mongo Shell by installing mongosh and performing update operations.