Lab6: MongoDB

### **Objectives**: Study NoSQL Document Model using MongoDB

**Estimated Time :** 3 hours

Lab Instruction

## Outline

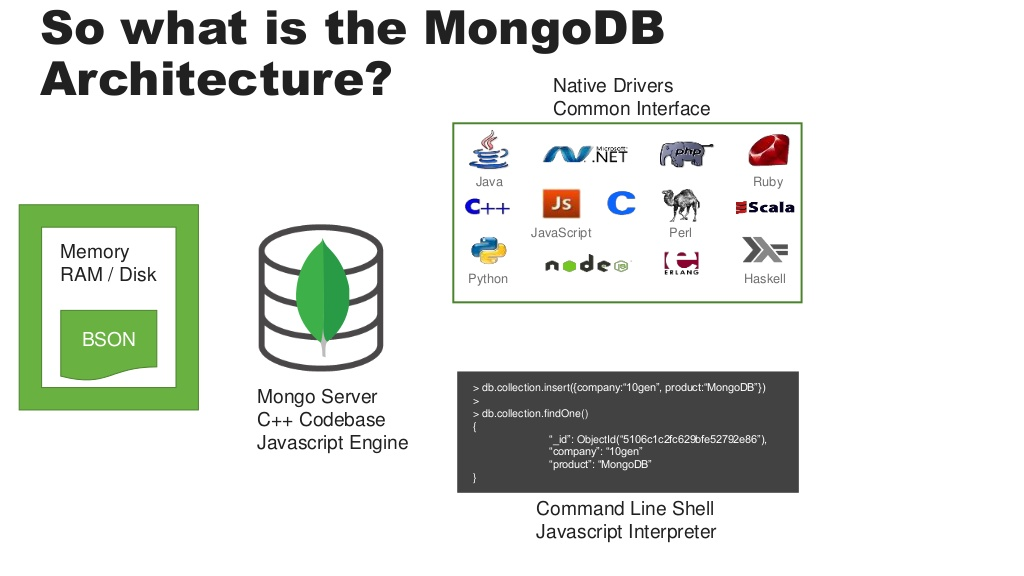
1. Mongo DB Overview
2. Setup Database on cloud
   * Create database
   * Create database user
   * Connect to database
3. CRUD Operations
   * Insert Document
   * Update Document
   * Delete Document
   * Query Data
   * Basic Query
   * Query Nested Field
   * Aggregate Function
   * Join

### MongoDB Overview

* 1. About MongoDB
* Stores data in flexible, JSON-like documents
* Fields can vary from document to document and data structure can be changed over time
* Distributed database, high availability, horizontal scaling, and geographic distribution
* Free to use
* Scalability
  + Performance Scale: Sustaining 100,000+ database read and writes per second while maintaining strict latency SLAs
  + Data Scale: Storing 1 billion+ documents in the database
  + Cluster Scale: Distributing the database across 100+ nodes, often in multiple data centers

*Ref. Examples of MongoDB users:* [*https://www.mongodb.com/mongodb-scale*](https://www.mongodb.com/mongodb-scale)

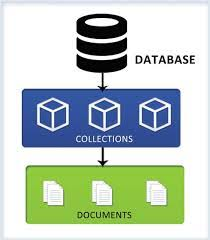
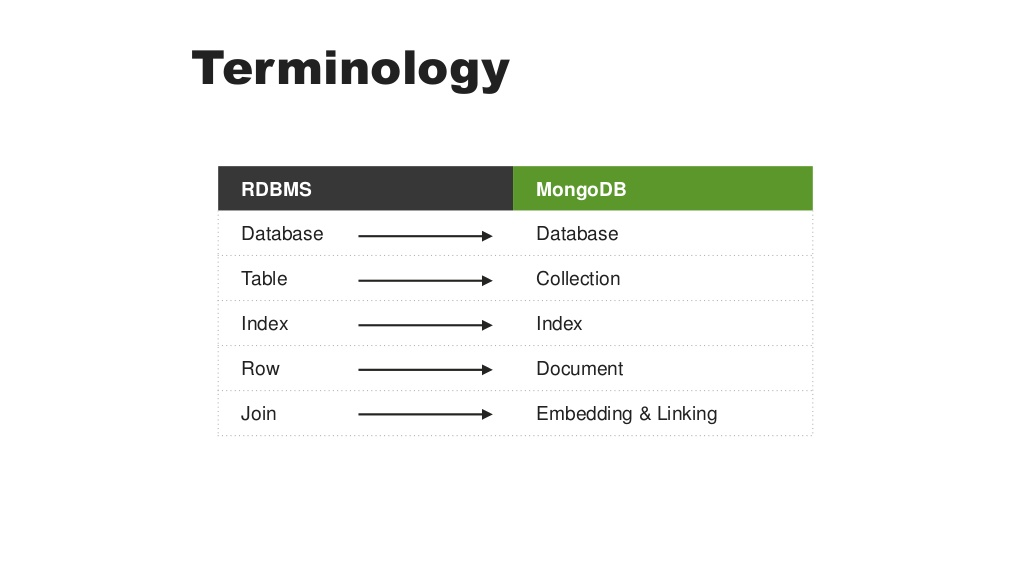
* 1. MongoDB Architecture



**MongoDB Atlas**

**MongoDB Shell / MongoDB Compass**

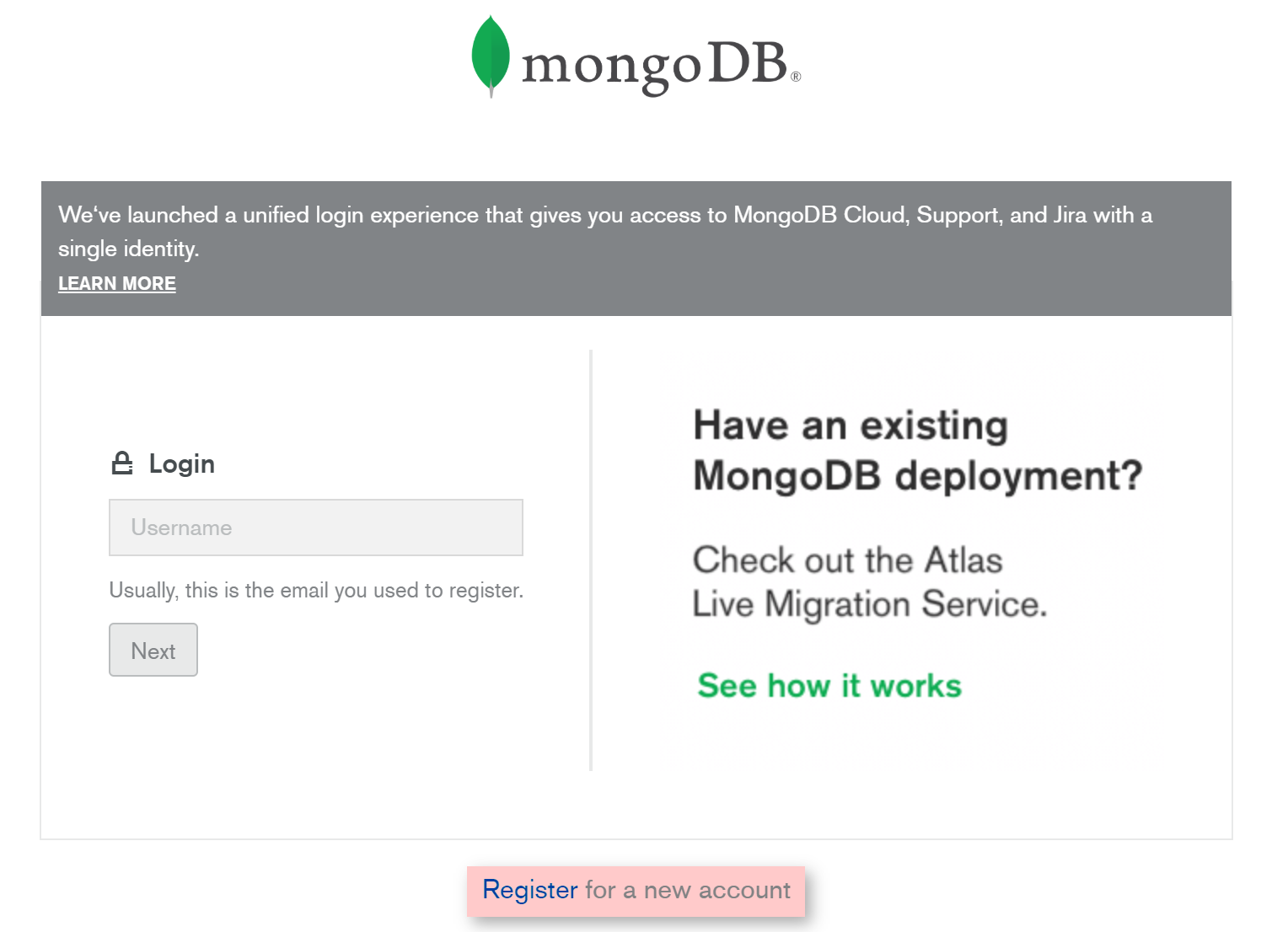
* 1. MongoDB Terminology

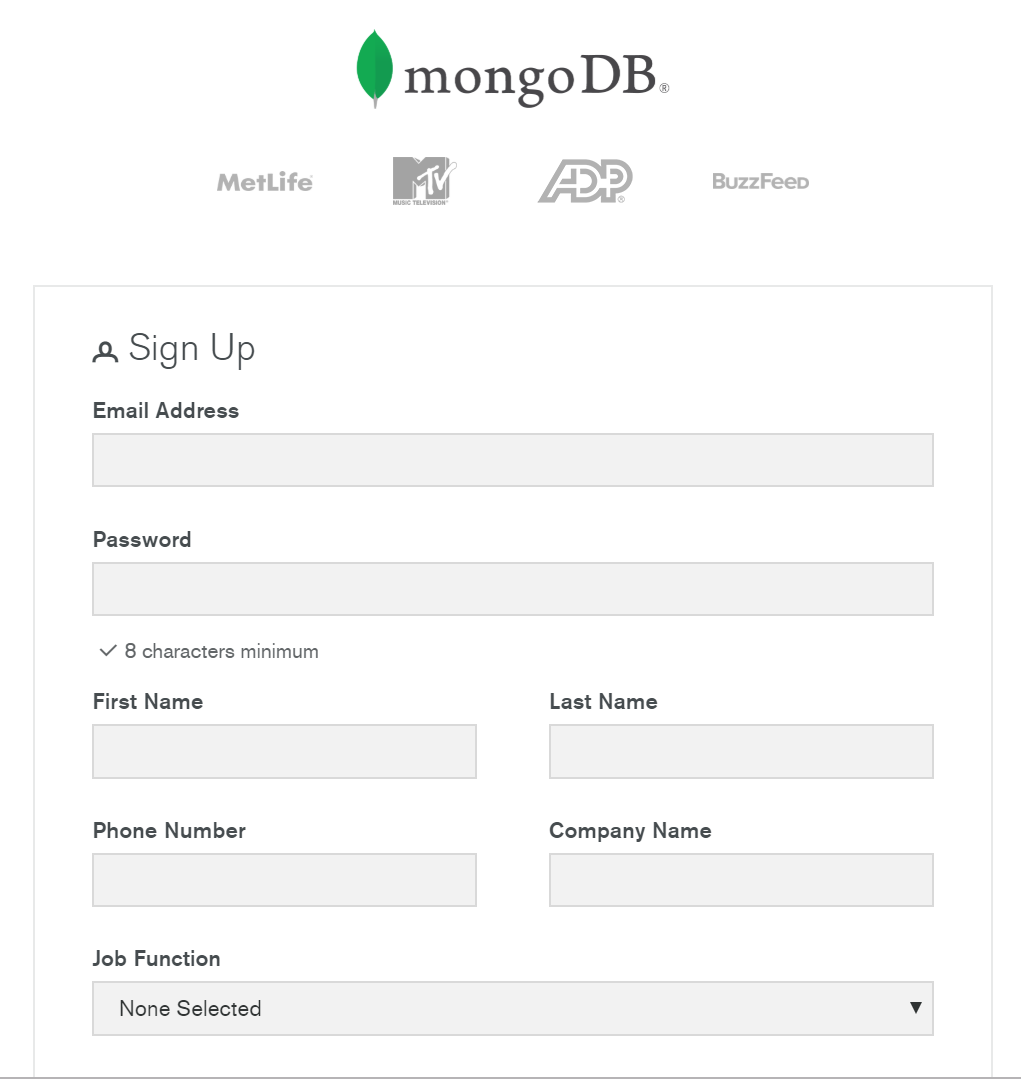


### Set up our cluster (= Database Server) in Mongo DB cloud

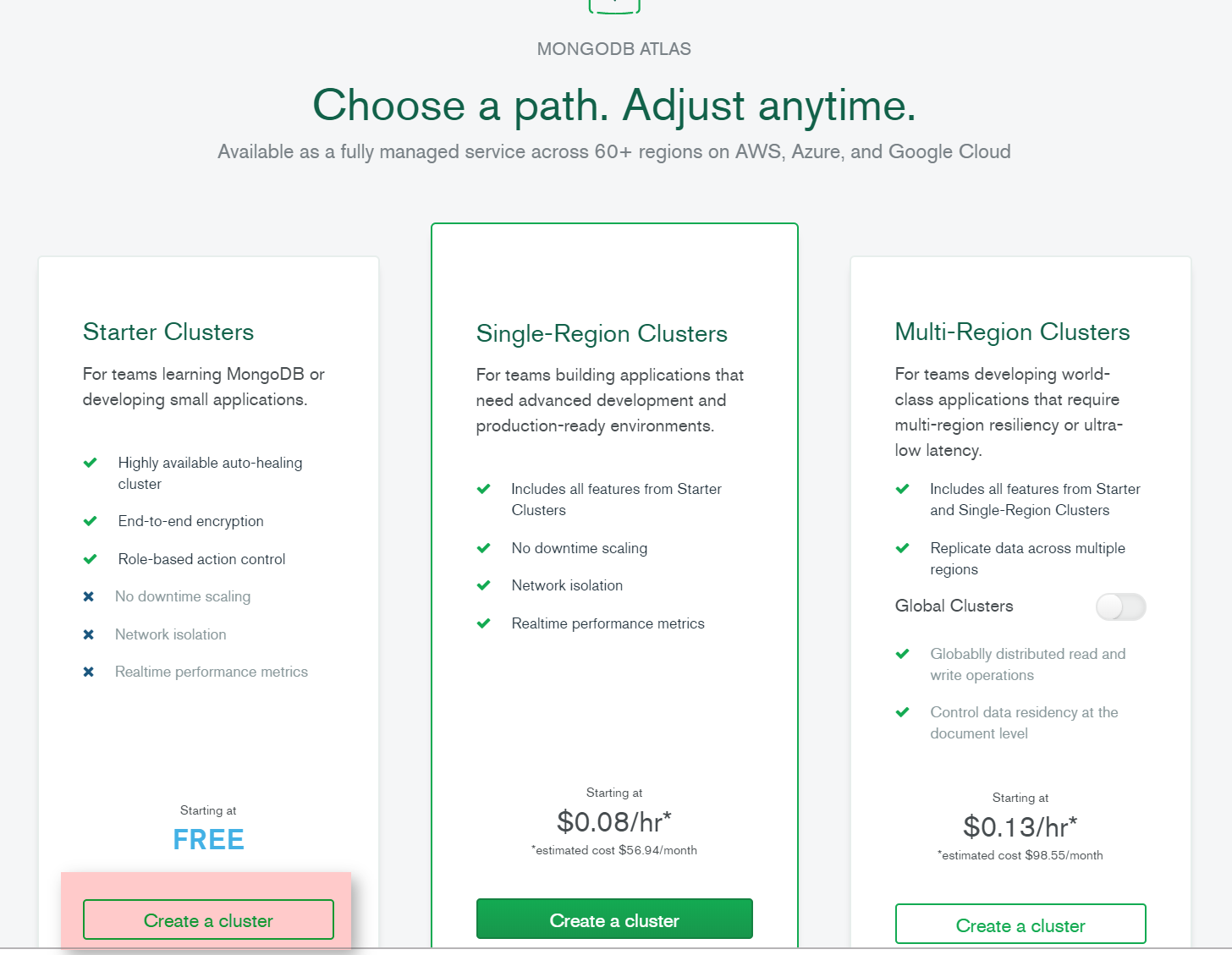
<https://cloud.mongodb.com/>

2.1 Sign up

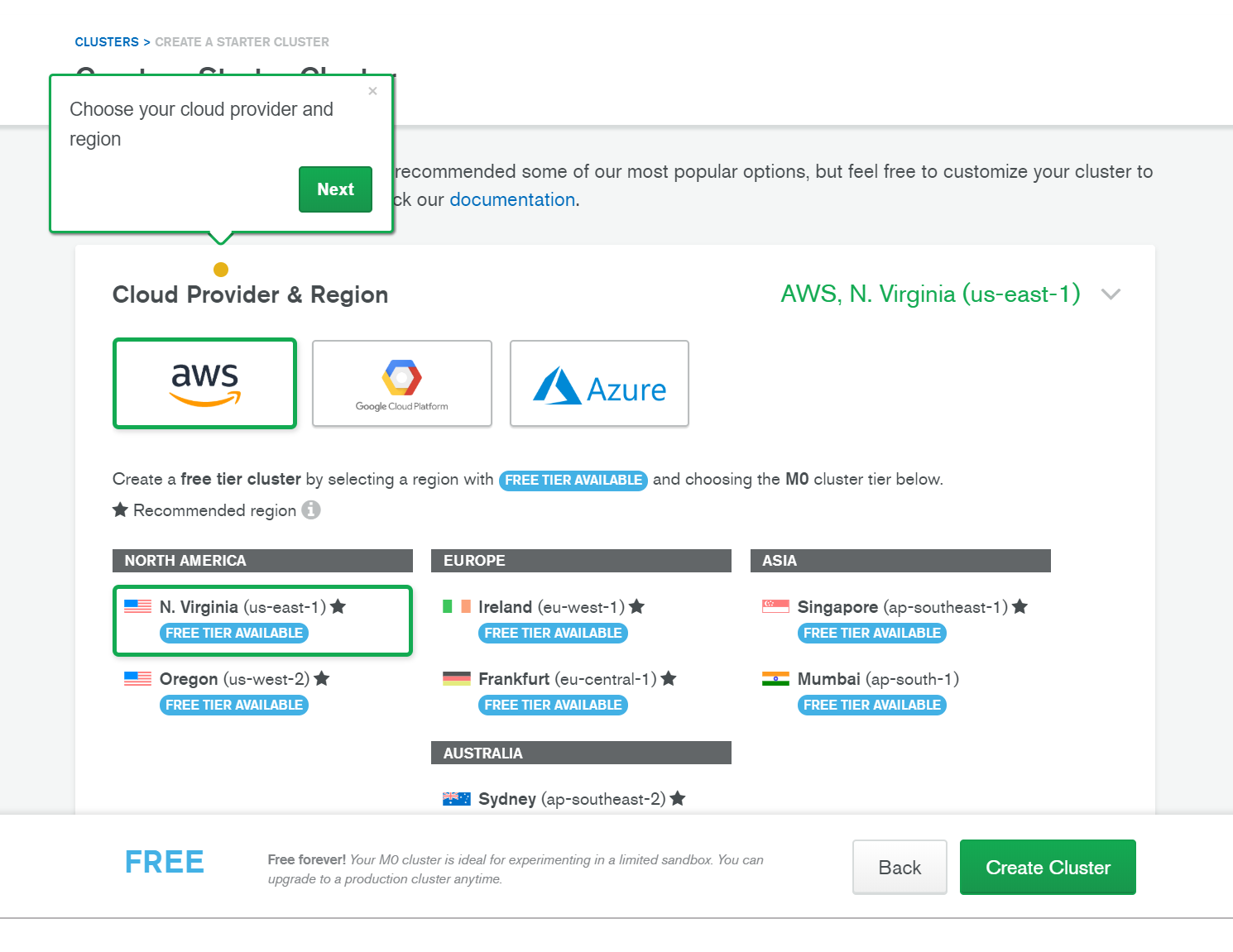


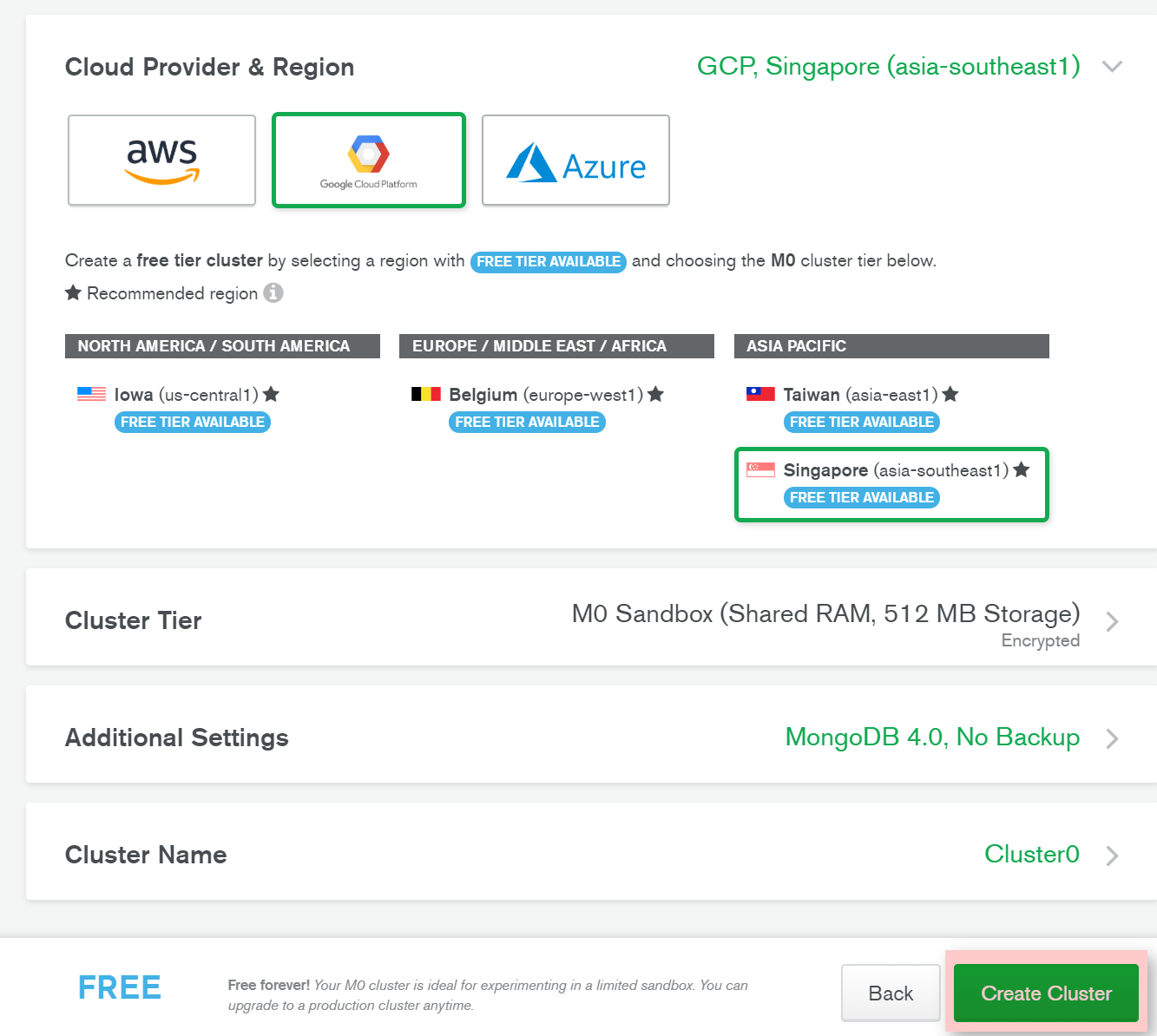


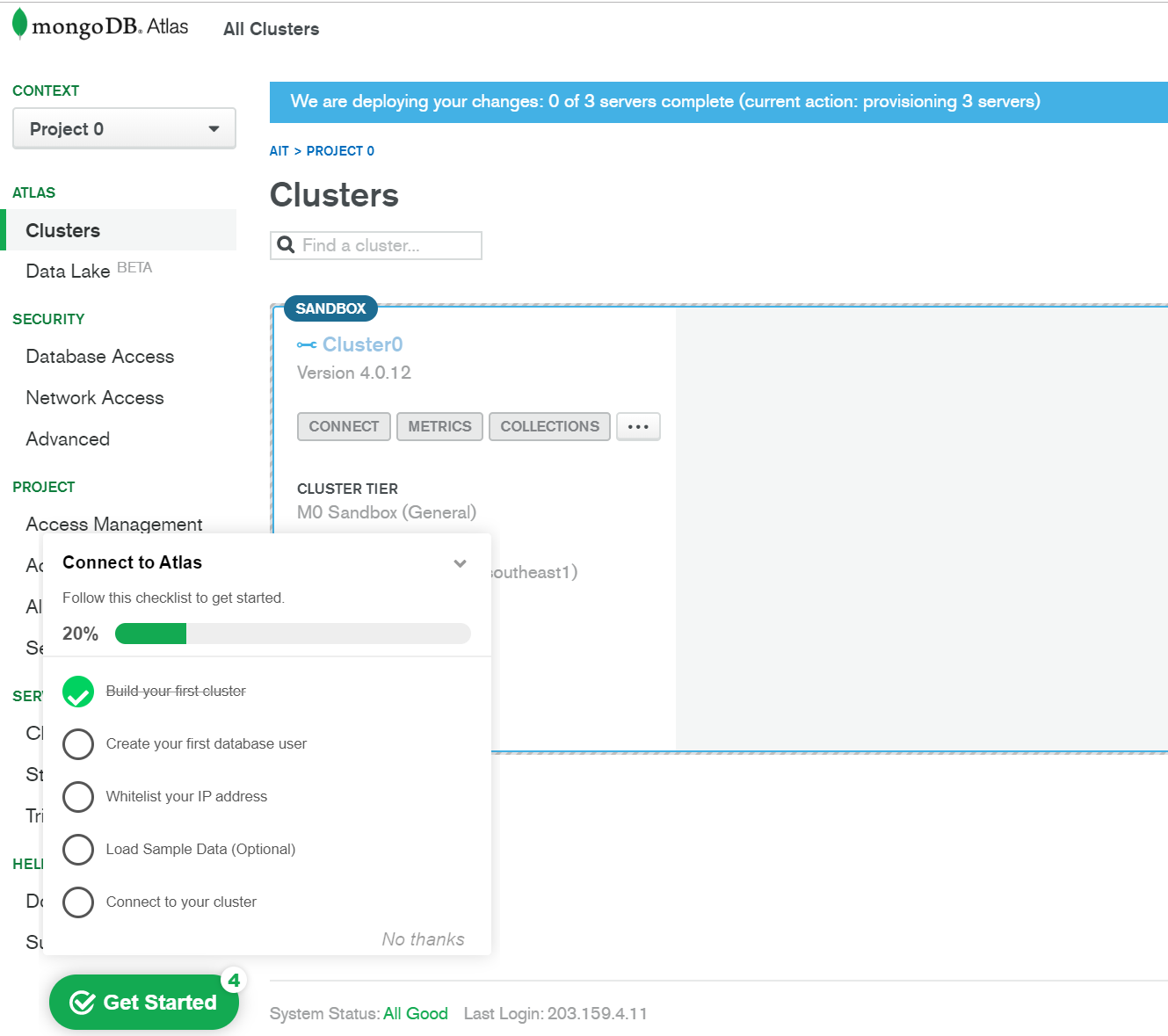
2.2 Create a Cluster



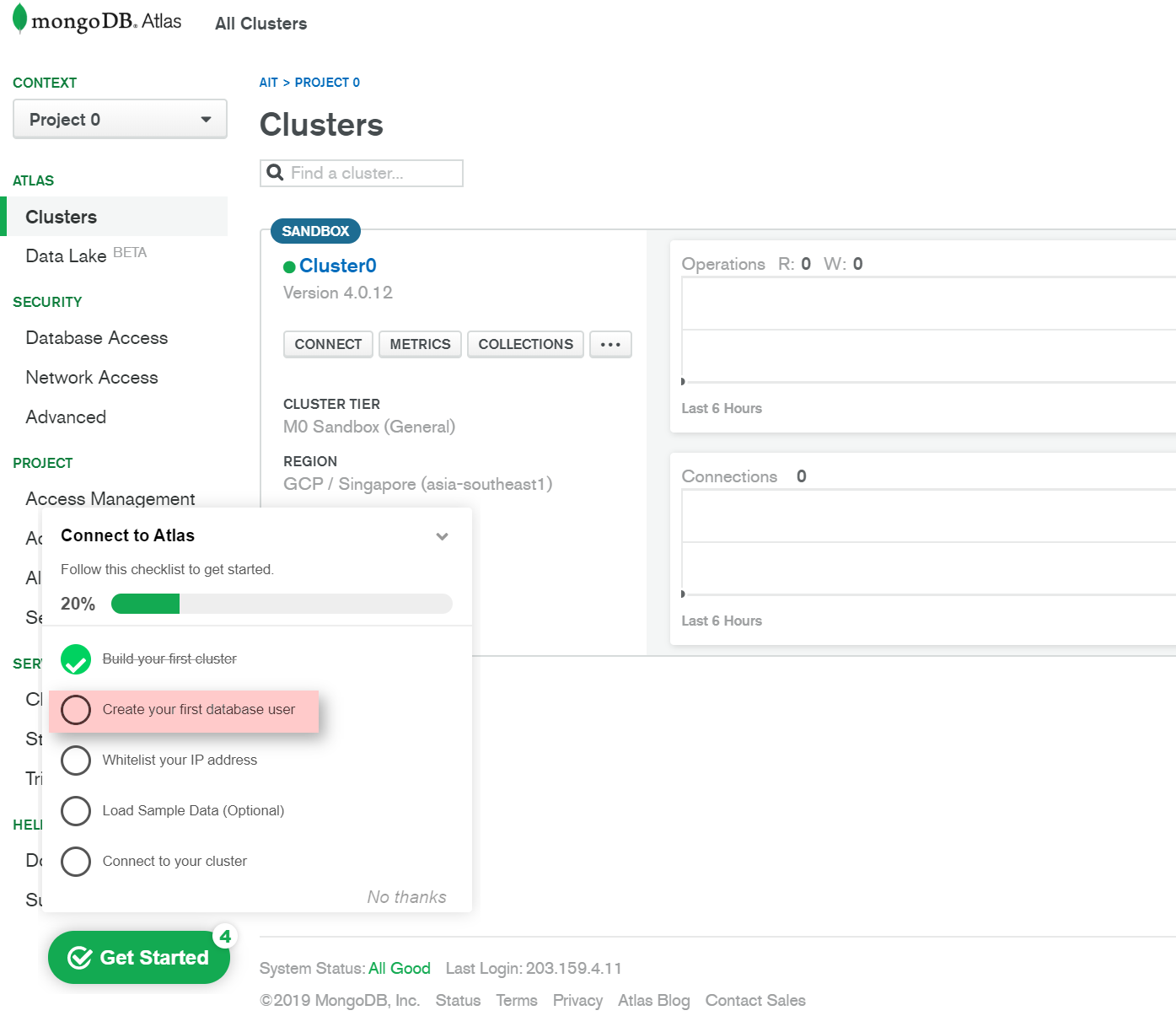
Select Cloud Provider and Region and Click “Create Cluster”

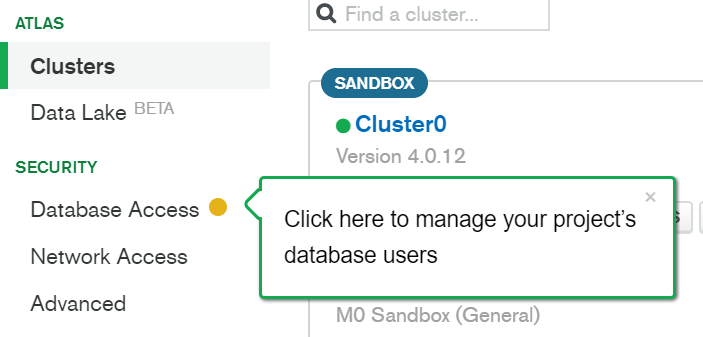


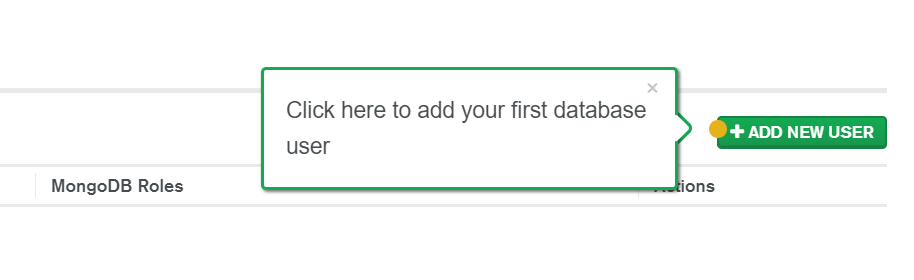


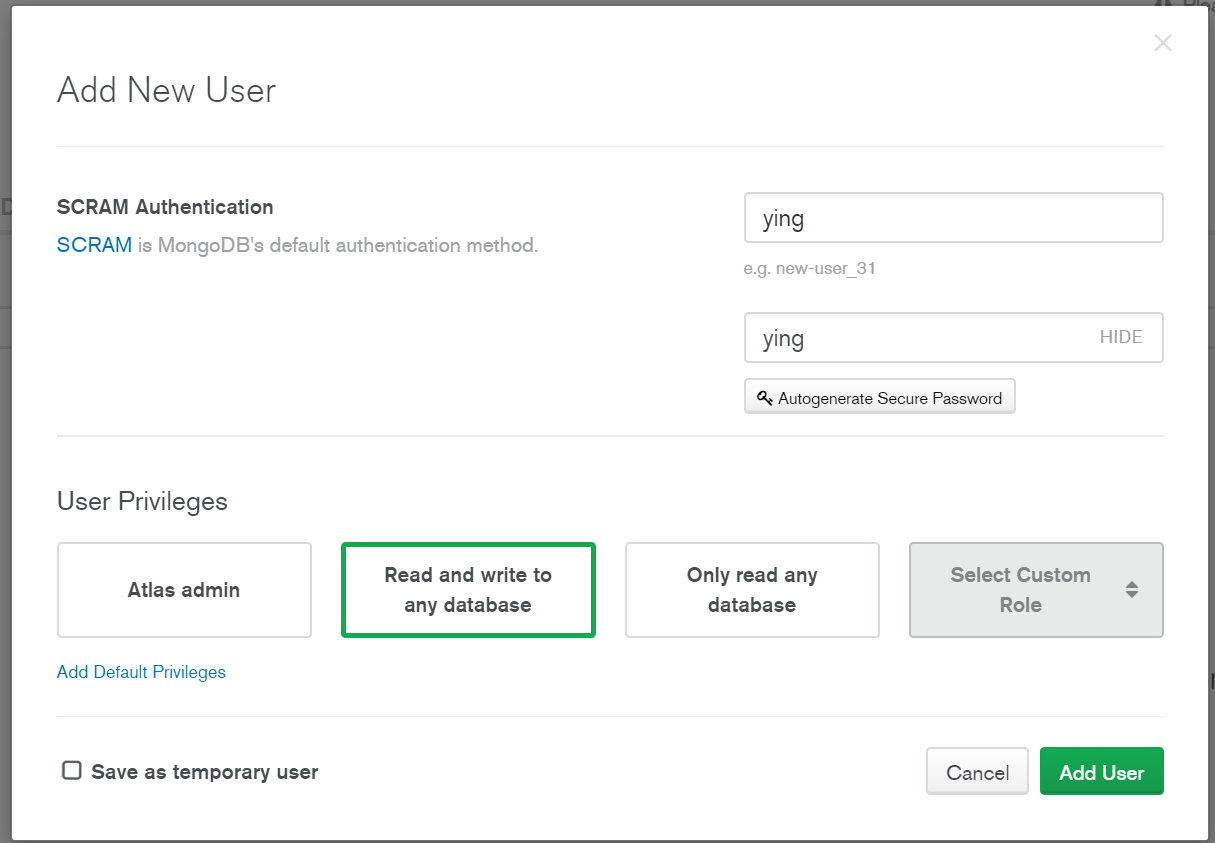


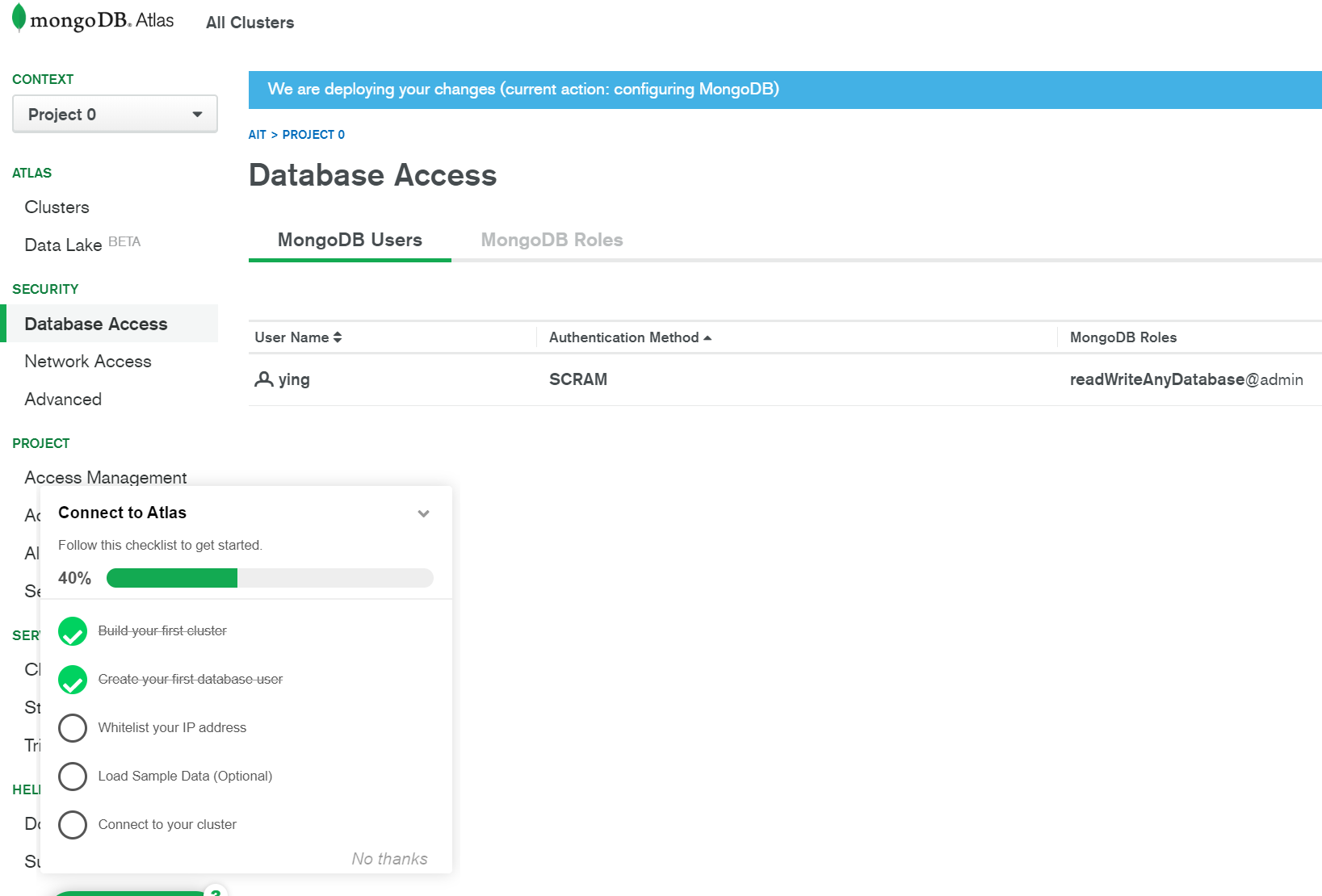
2.3 Create your first database user



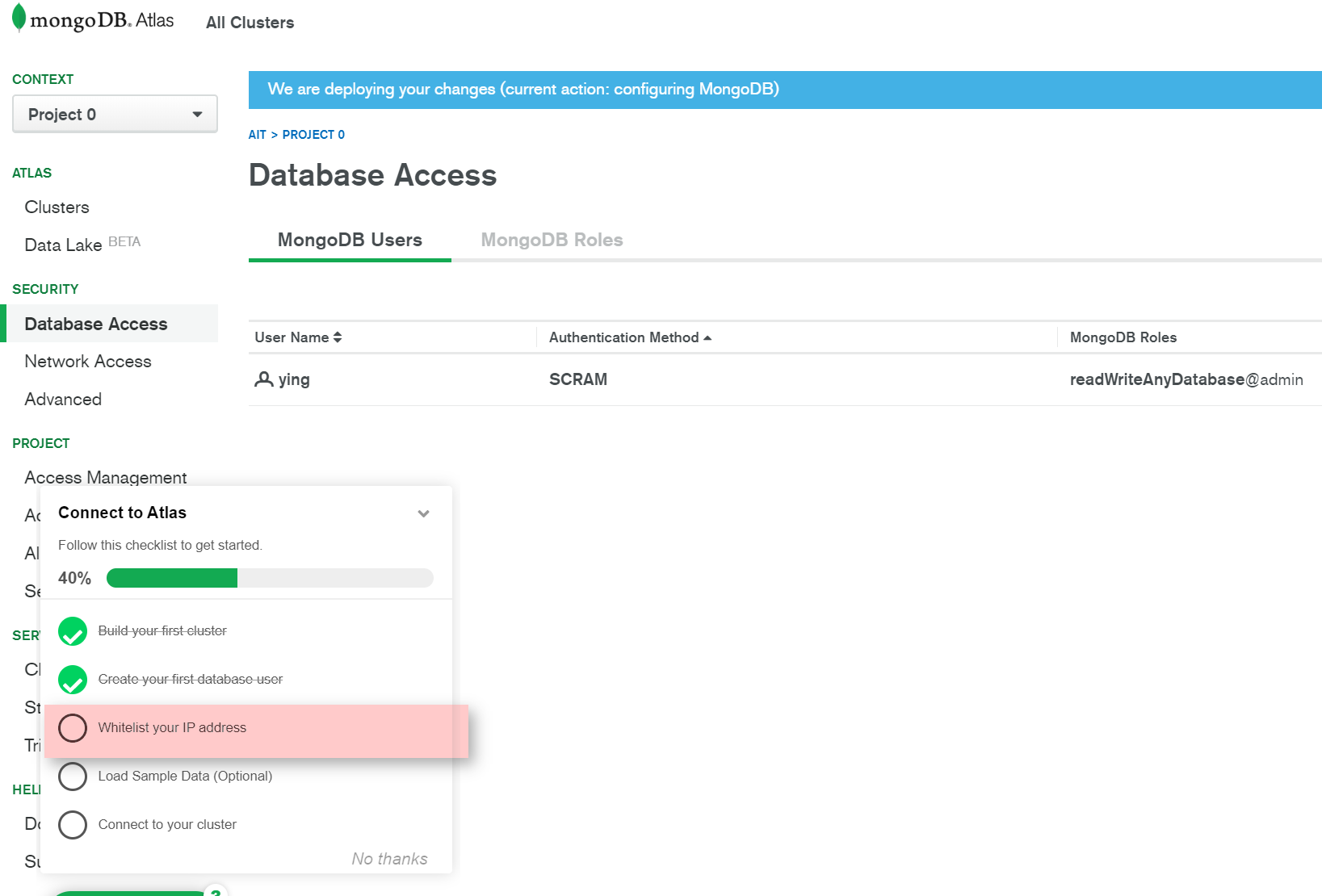


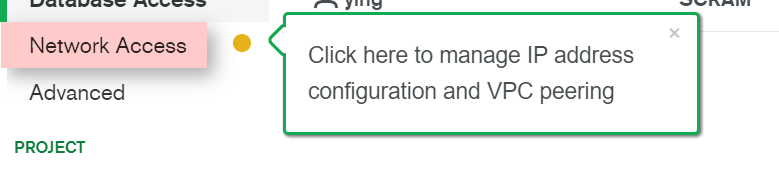


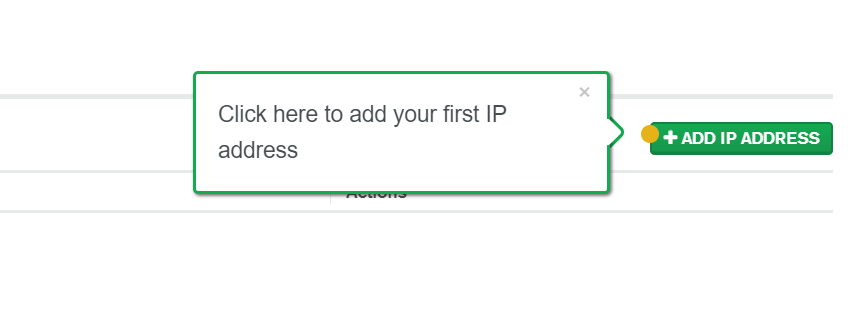


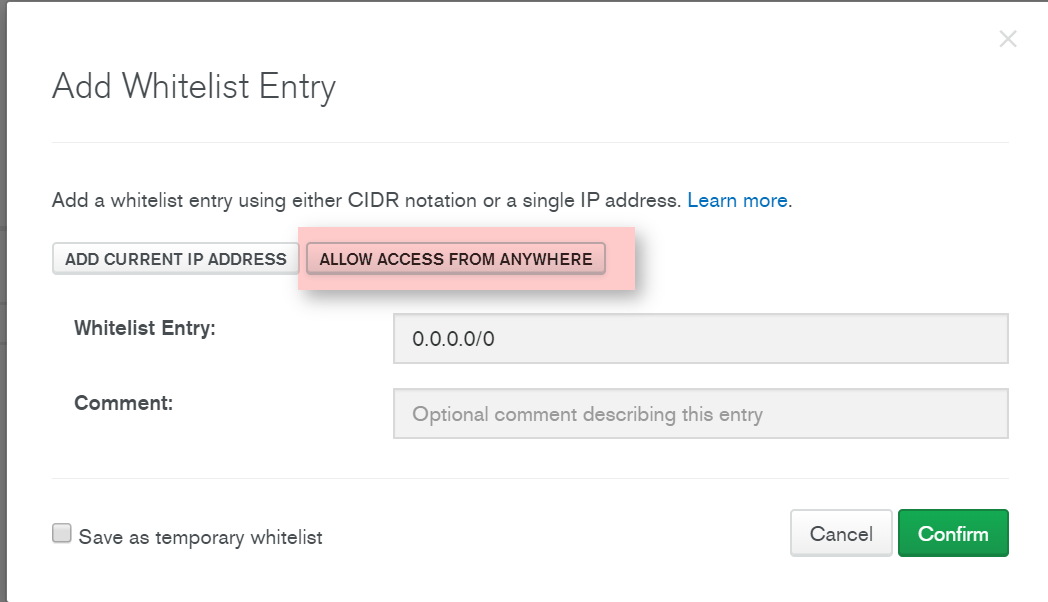


* 1. Allow Hosts that can access the Cluster

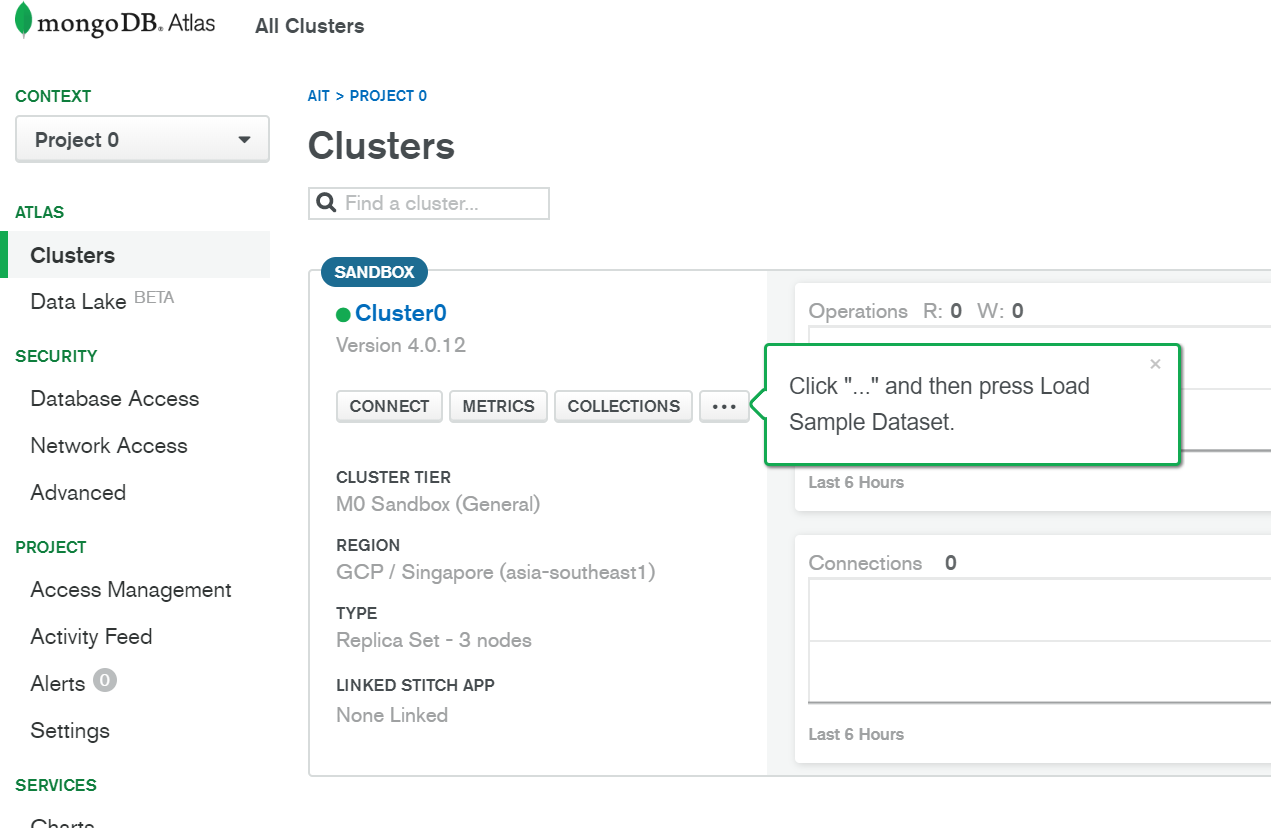


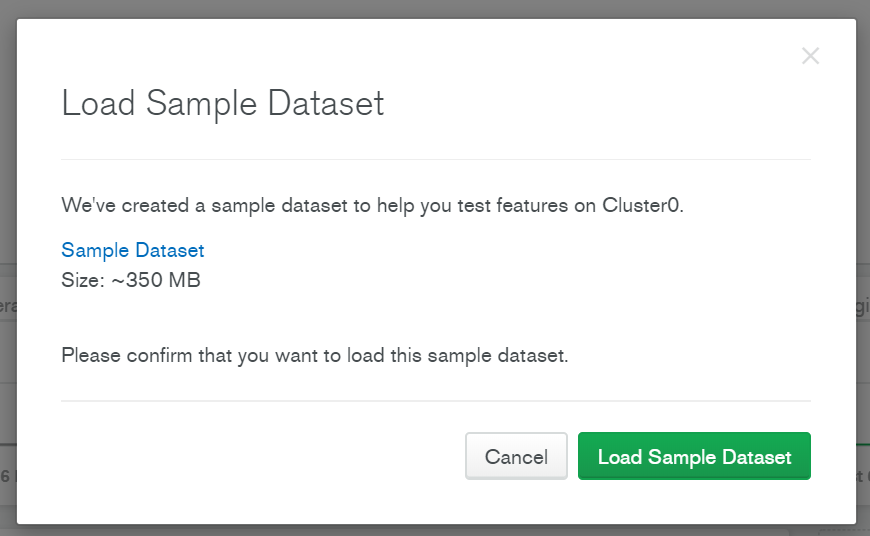




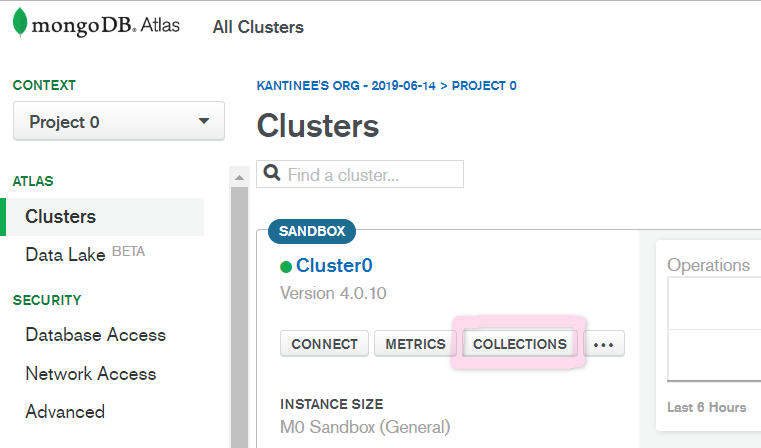


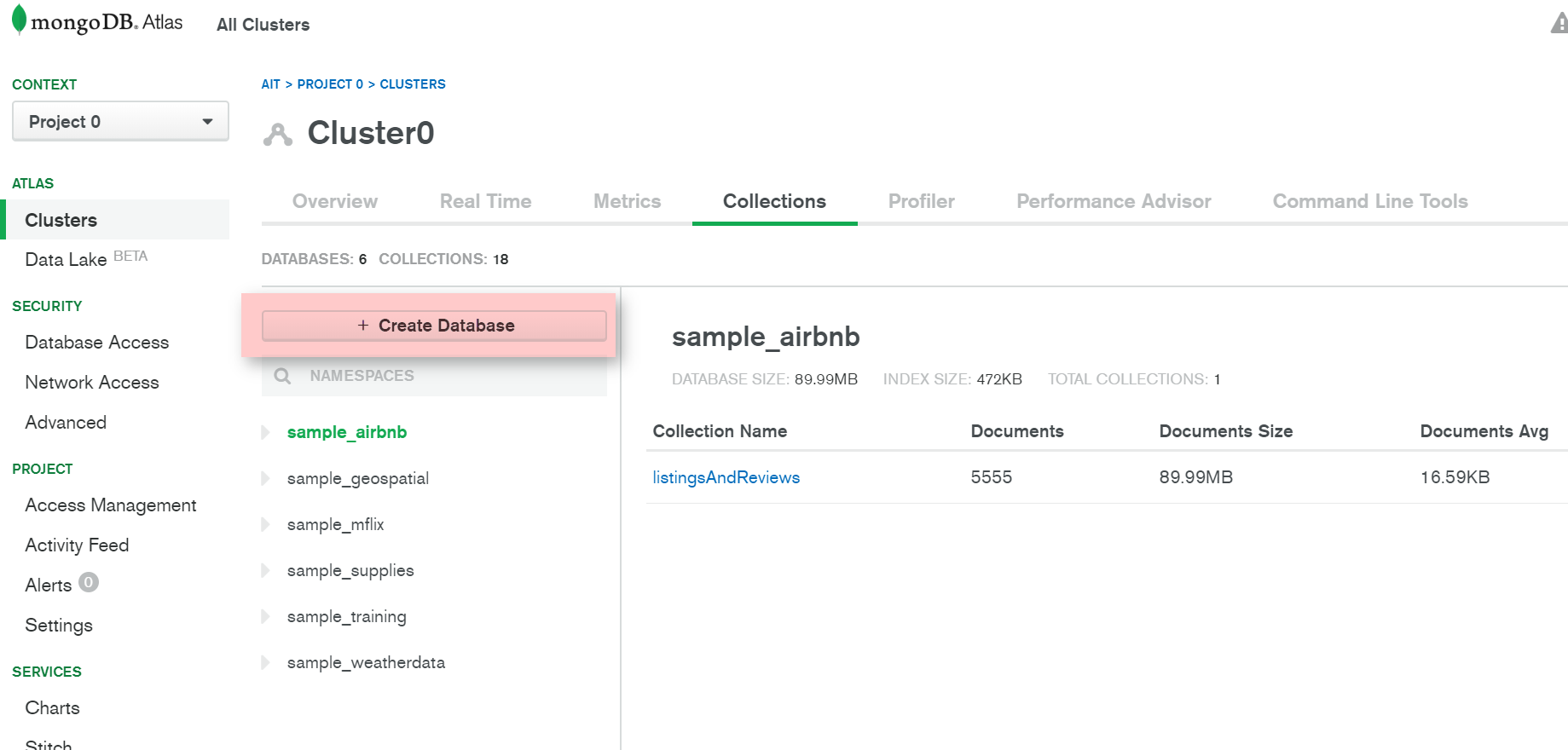
* 1. (Optional) Load Sample DB



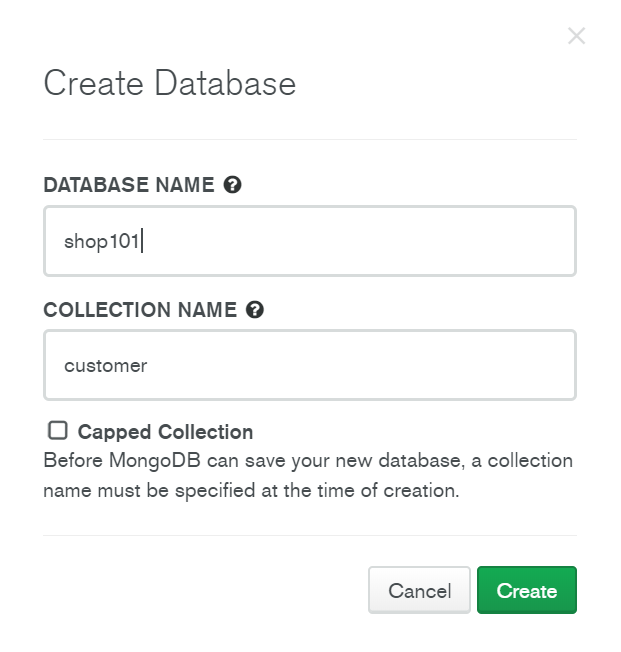


* 1. Create Database in Mongo Atlas

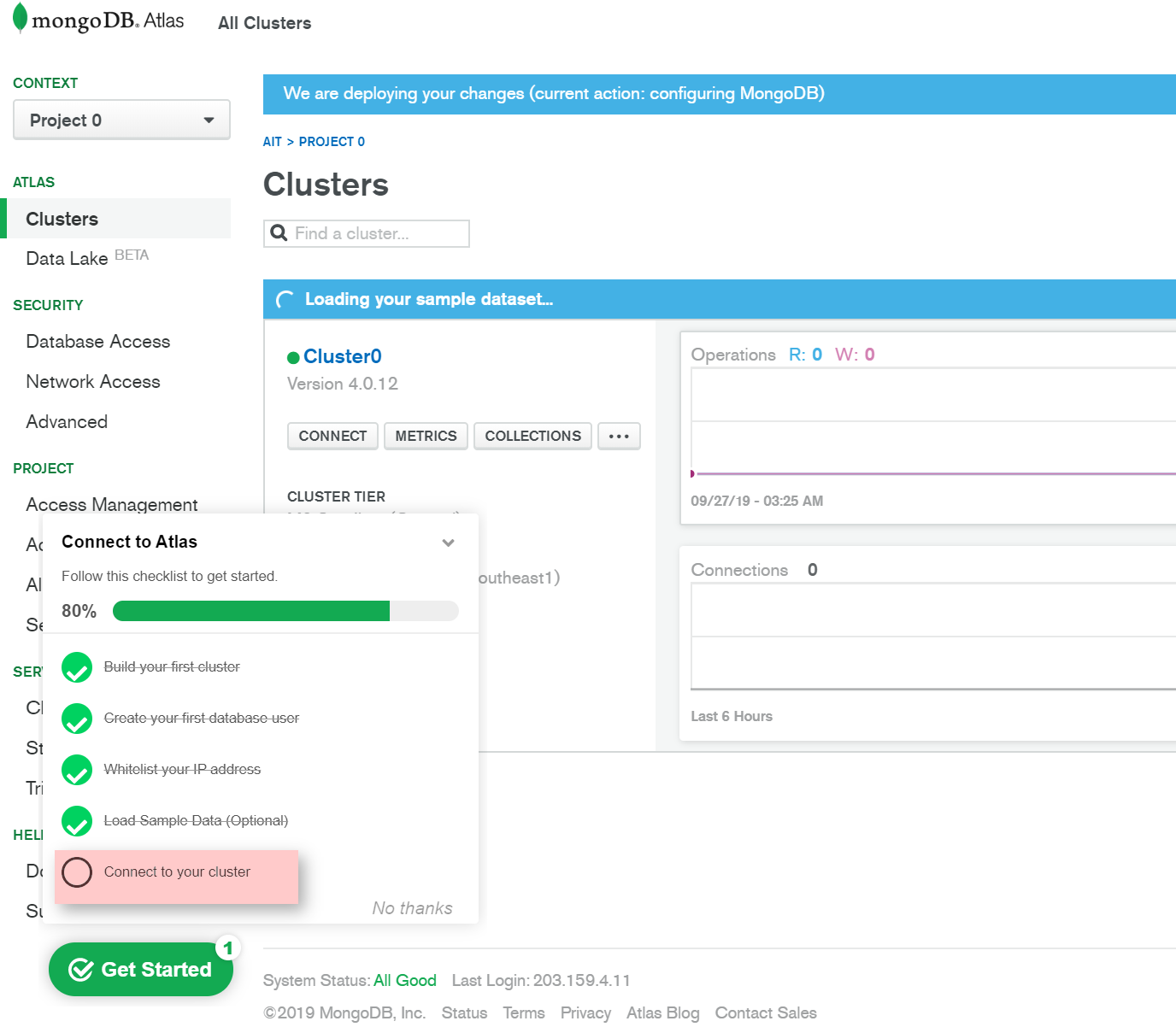


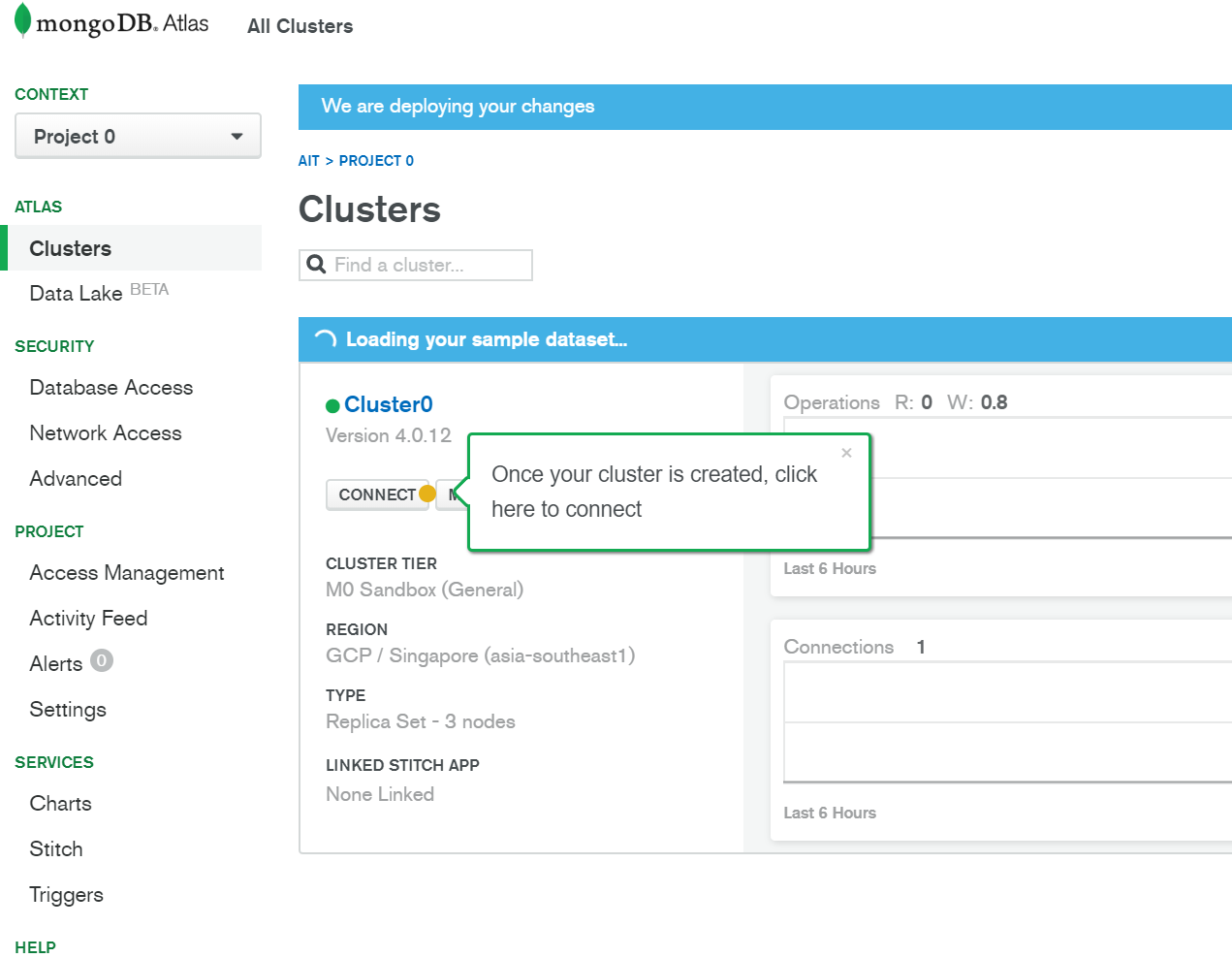


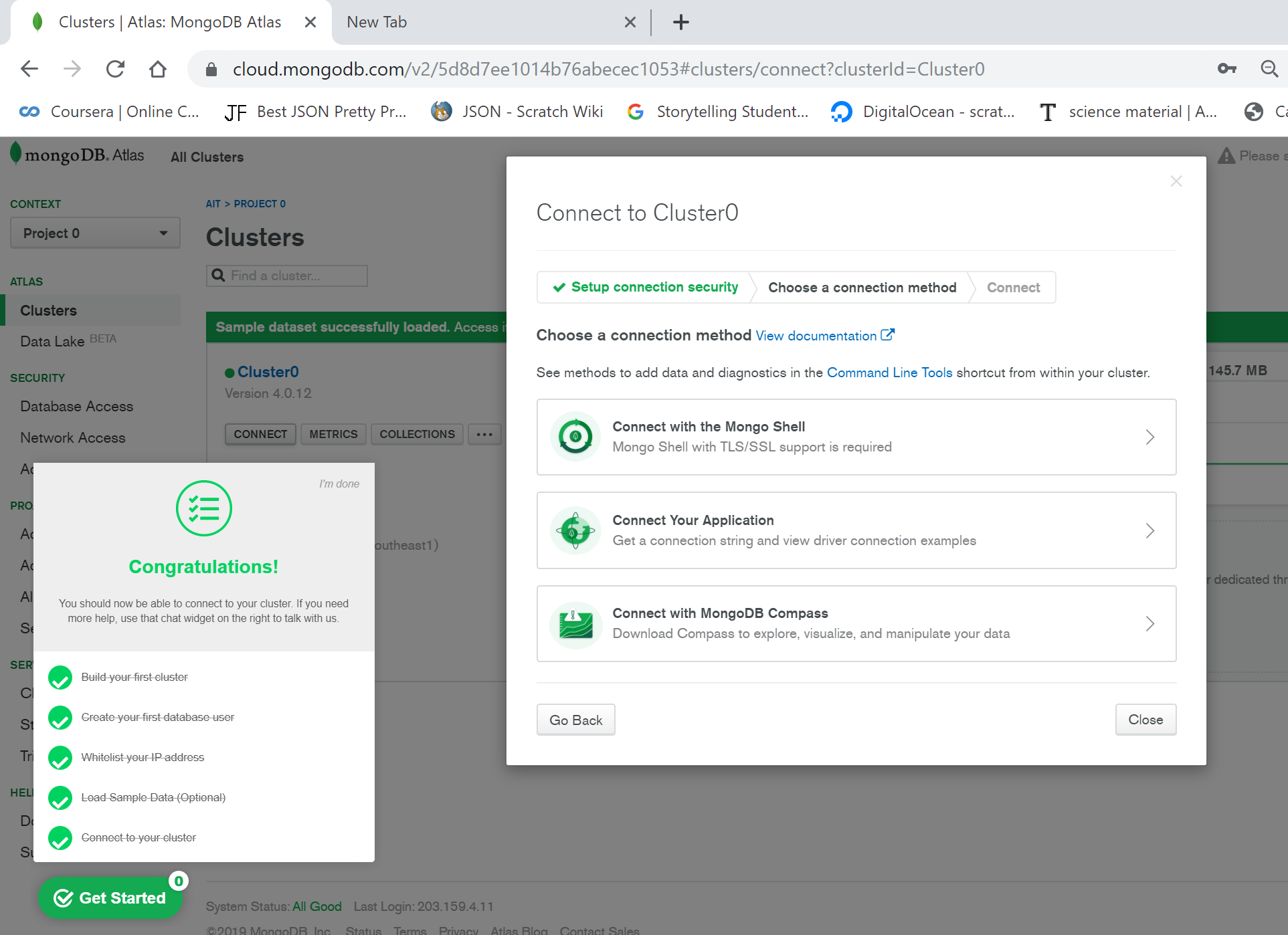
Create Database name =shop101 and Collection name = customer.



* 1. Connect to the Cluster using the User. There are 3 methods to connect: Mongo Shell, Your App and Mongo Compass.

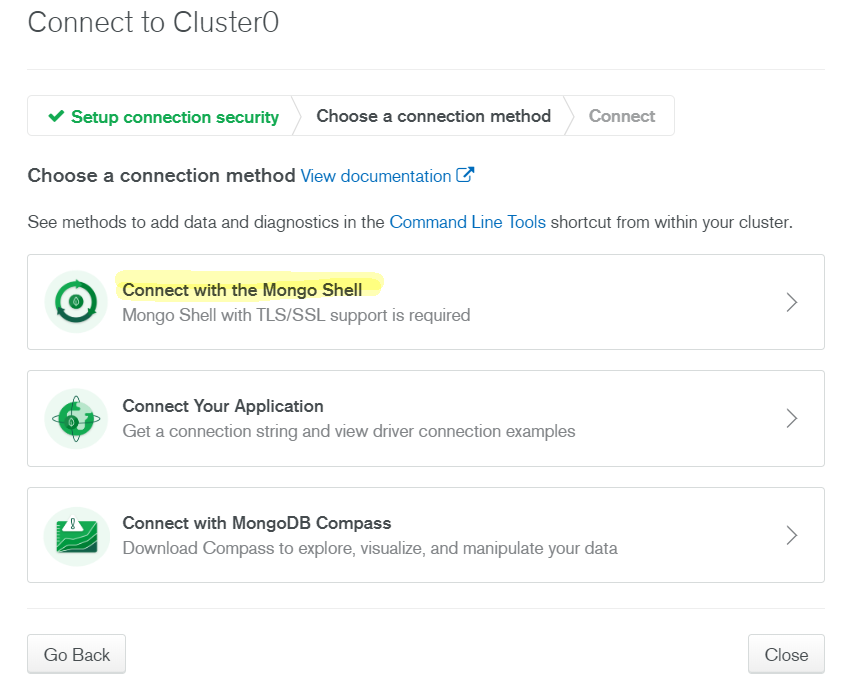


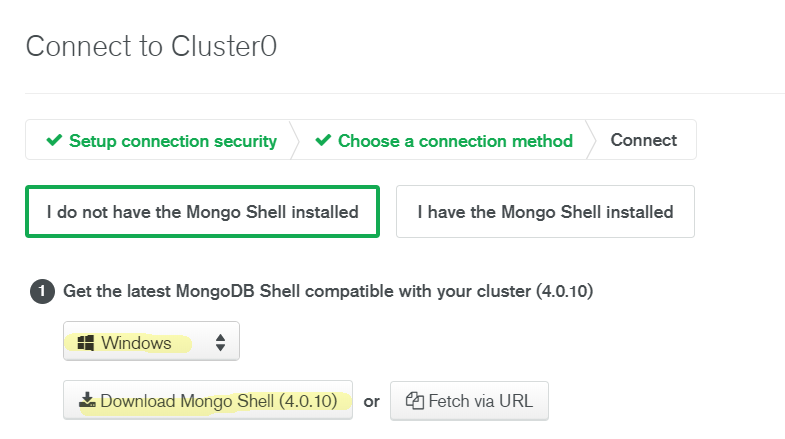


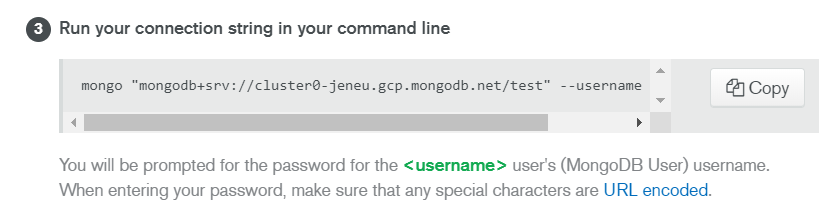


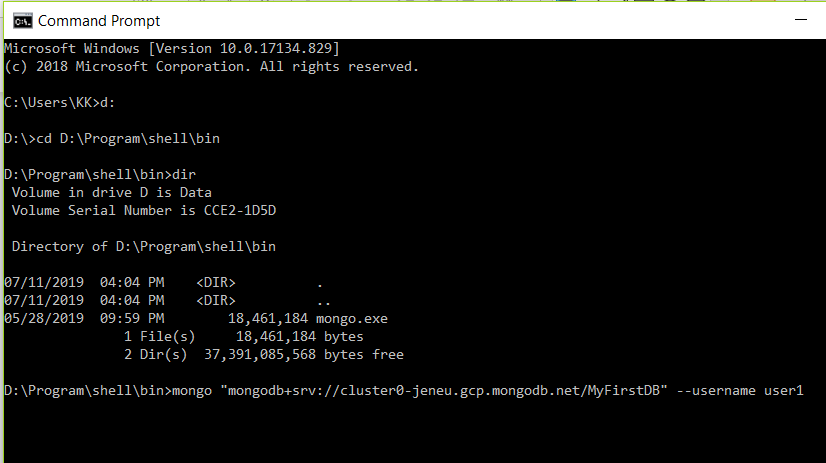
Install Shell or Compass and using the connection string to connect. Example of Connection string: mongodb+srv://ying:<password>@cluster0-dfi1d.gcp.mongodb.net/<database name>

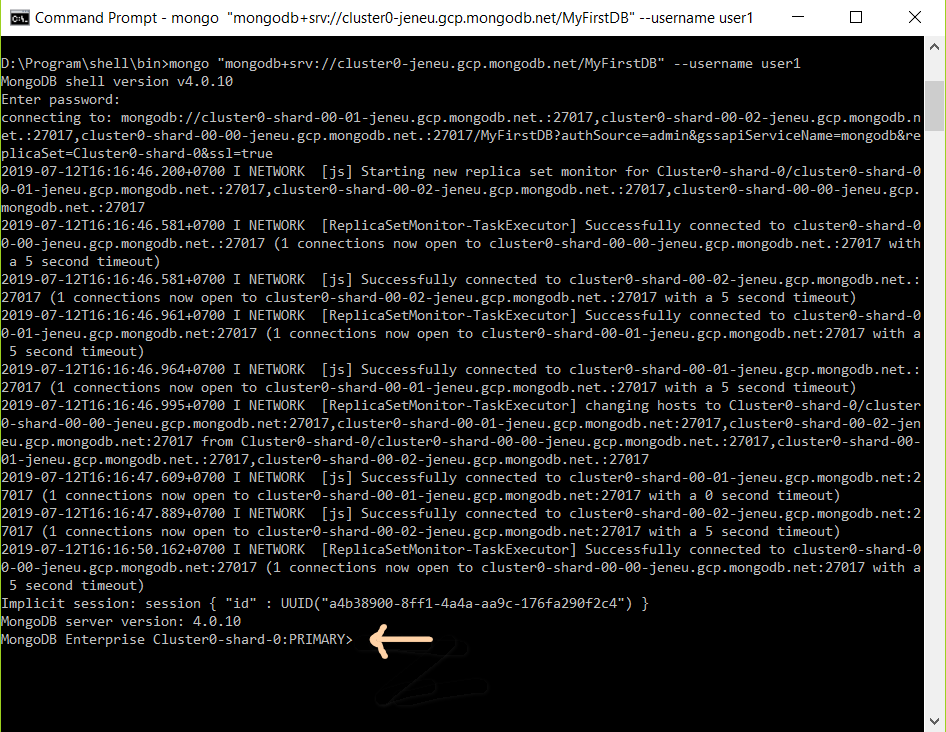
*In this tutorial we will use Shell*











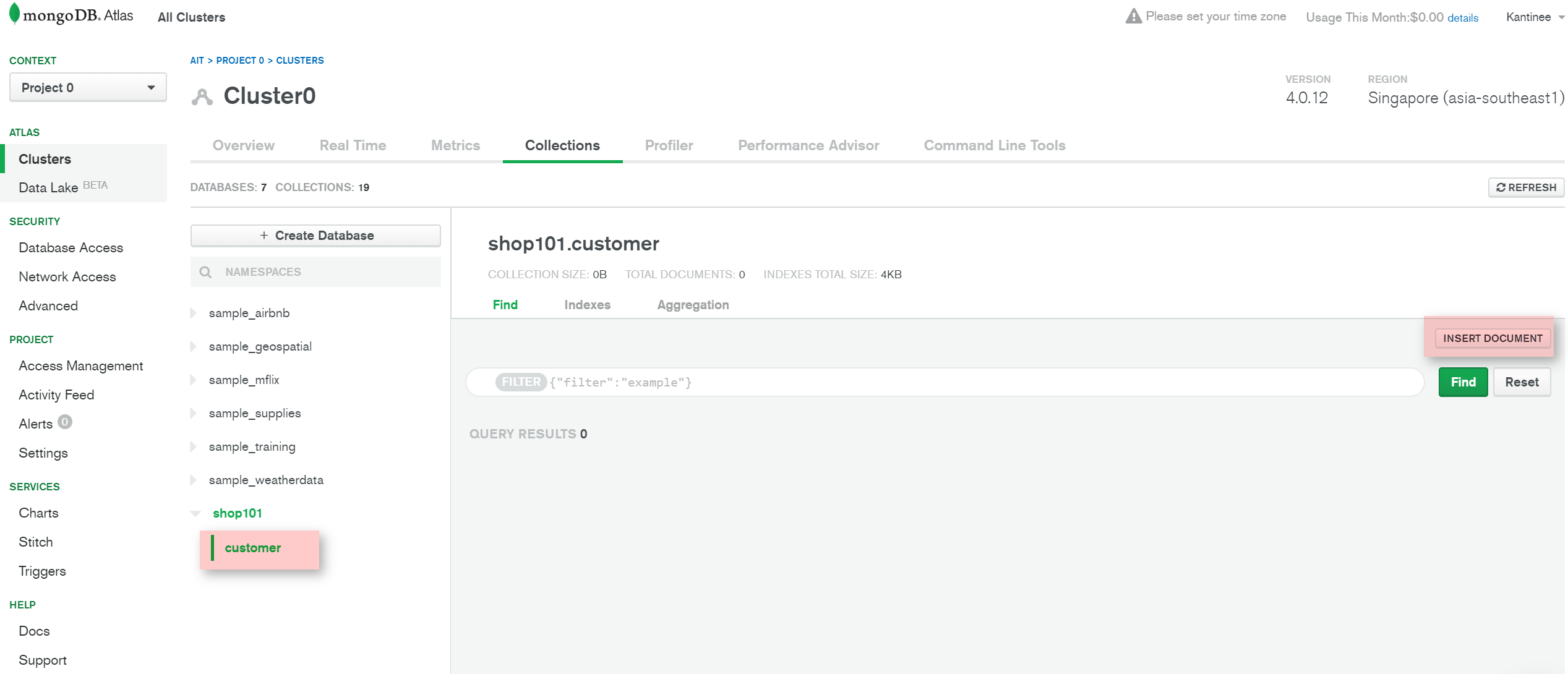
mongo "mongodb+srv://cluster0-jeneu.gcp.mongodb.net/<database name=shop101>" --username <username>

### Insert, Update, Delete Document

You can do DDL operation in Mongo Atlas, Mongo Compass or Mongo Shell.

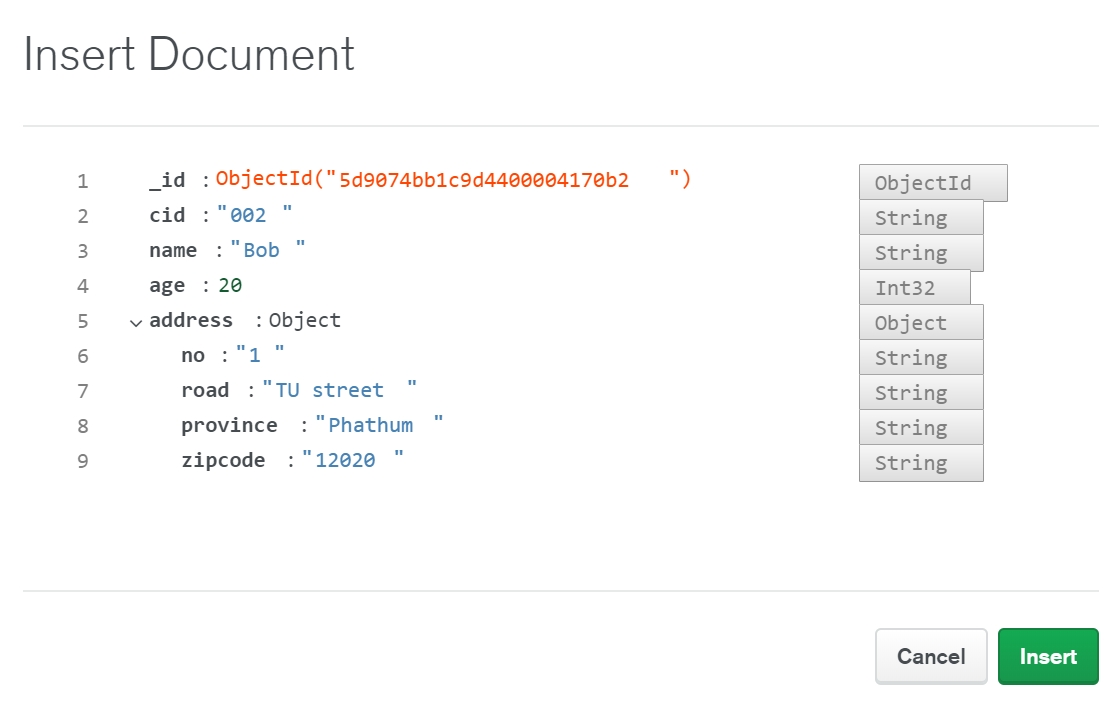
3.1 Insert customers in “shop101” database using **Mongo Atlas**

Click “Collection” -> click “customer” -> click “Insert Document”



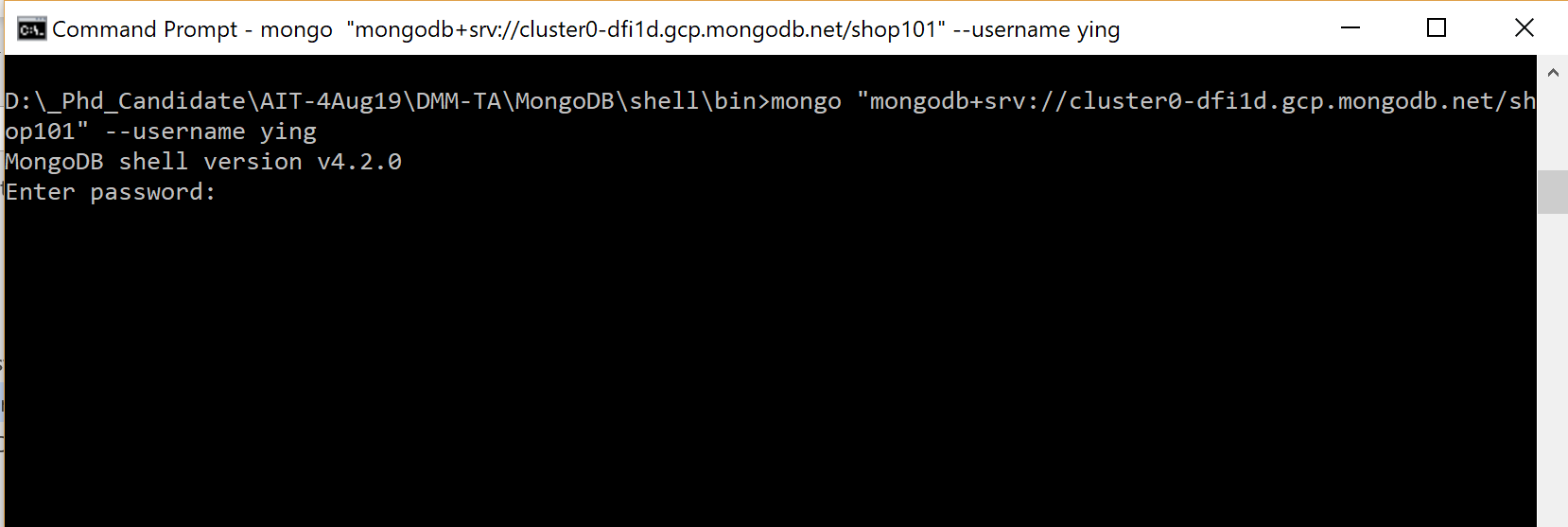
A screenshot of a cell phone

Description automatically generated

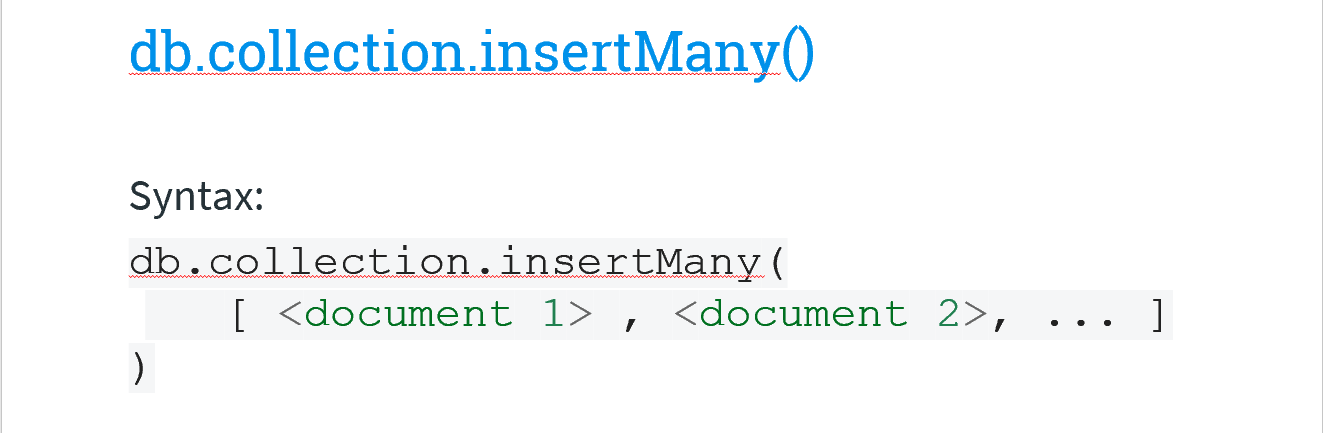


3.2 Insert inventory using Mongo Shell

Get connection string from Atlas = (mongo "mongodb+srv://cluster0-dfi1d.gcp.mongodb.net/shop101" --username ying)



**Insert Command**



*db.inventory.insertOne(*

*{ item: "canvas", qty: 100, price: 500, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } })*

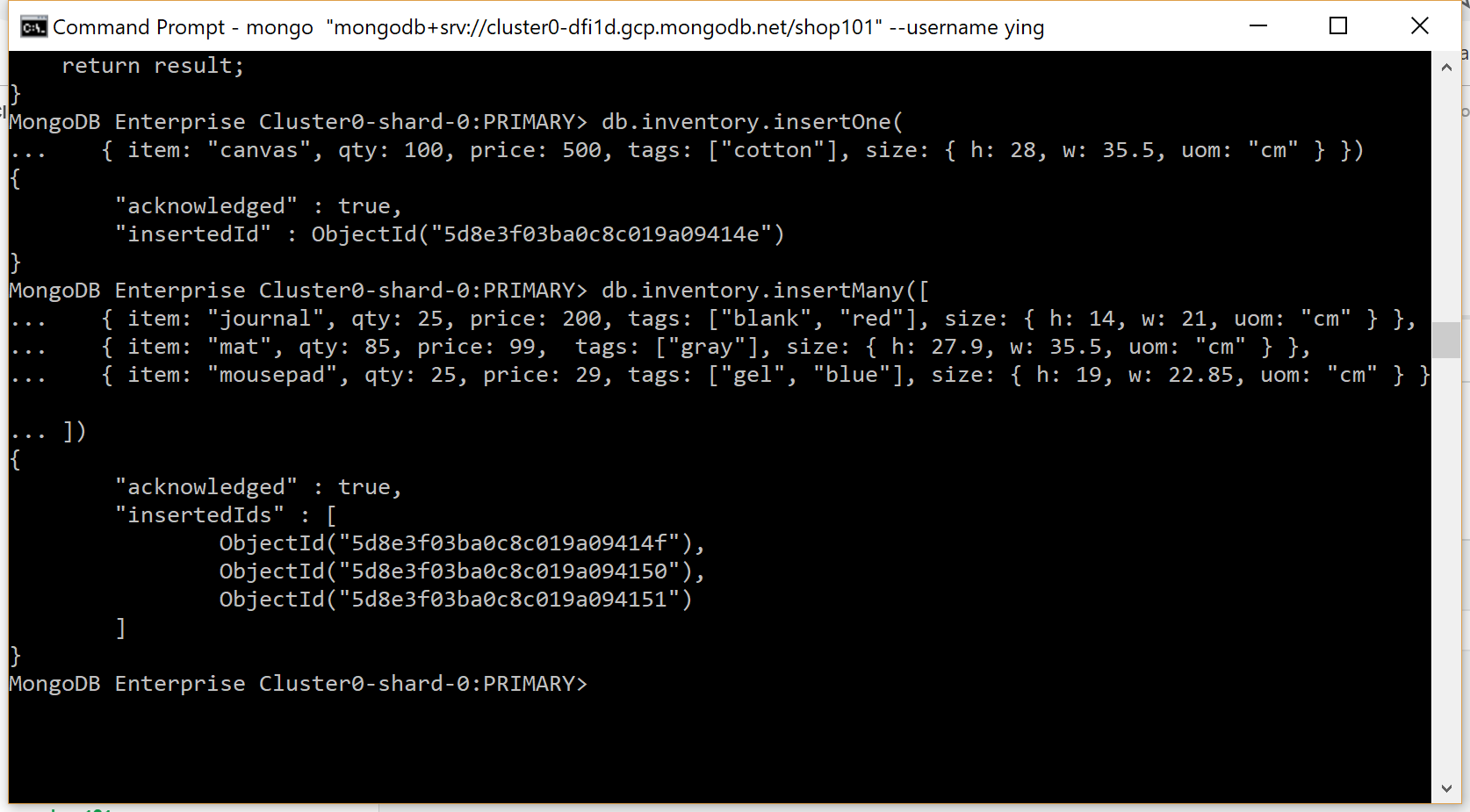
*db.inventory.insertMany([*

*{ item: "pen", qty: 25, price: 200, tags: ["red"], size: { h: 14, w: 1, uom: "cm" } },*

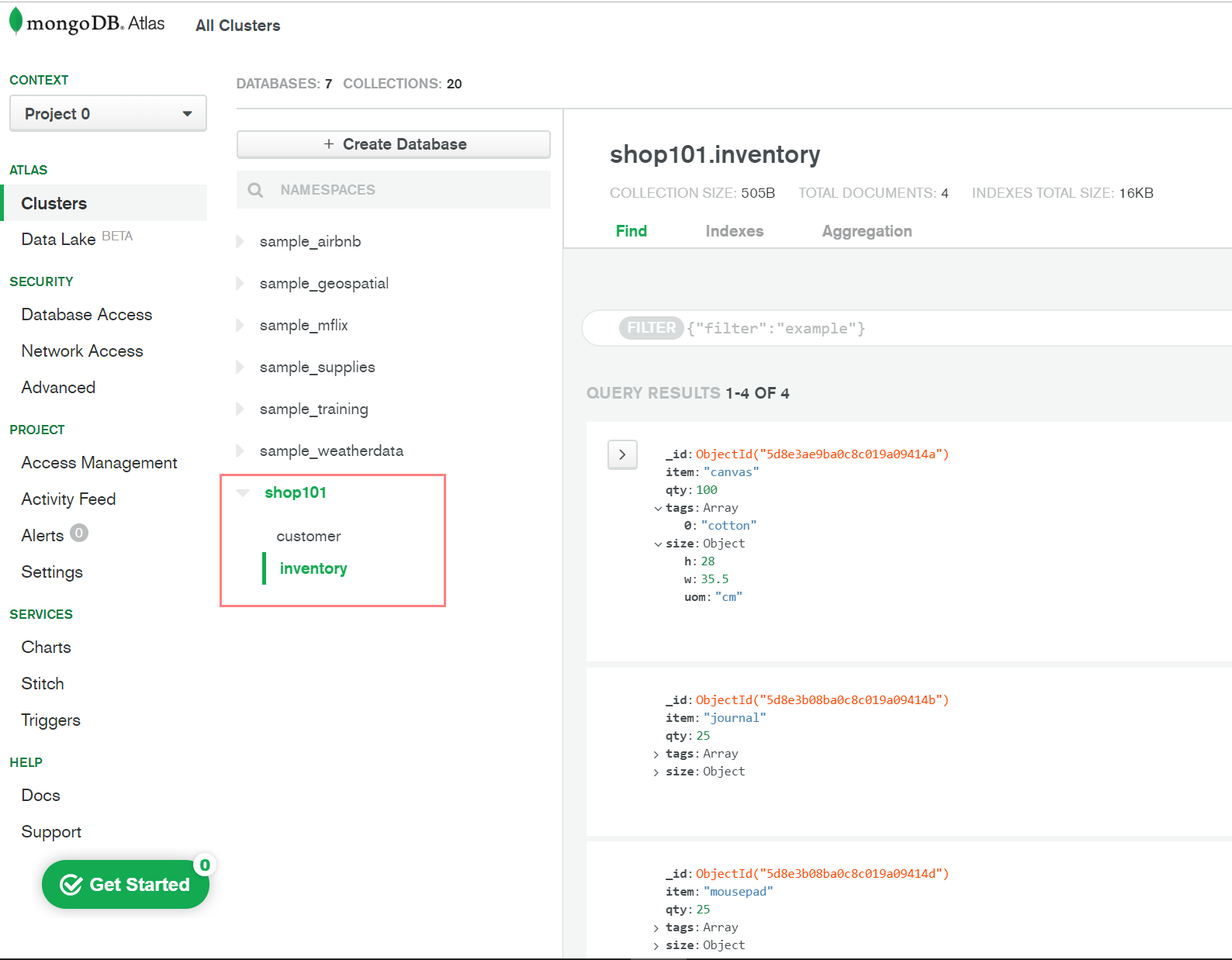
*{ item: "mat", qty: 85, price: 99, tags: ["gray"], size: { h: 27.9, w: 35.5, uom: "cm" } },*

*{ item: "mousepad", qty: 25, price: 29, tags: ["gel", "blue"], size: { h: 19, w: 22.85, uom: "in" } }*

*])*

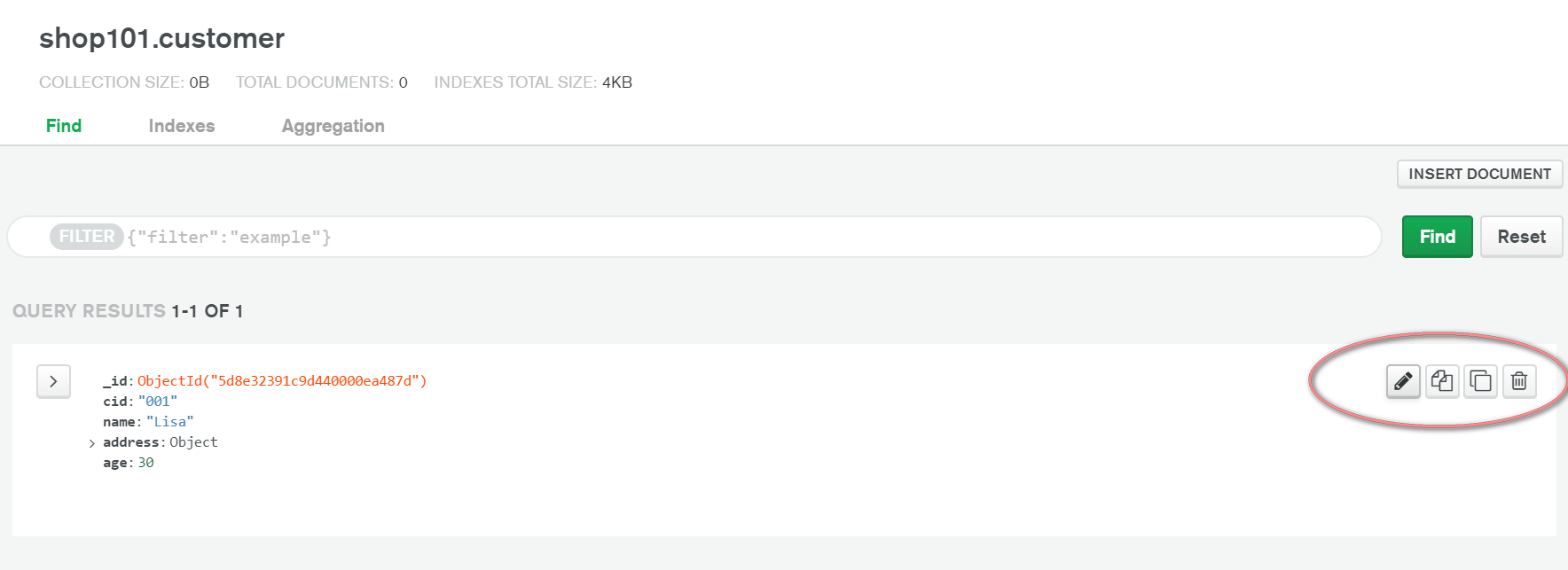


Verify



Note: if you want to UPDATE or DELETE:

@Atlas you can click the document and click small buttons on the right corner.



(@Shell see more command in <https://docs.mongodb.com/manual/crud/>. )

3.4 Insert sales in “shop101” database using **Mongo Shell**

*db.sales.insertOne(*

*{ item: [{pid: "canvas", qty: 1, price: 500},{pid: "pen", qty: 1, price: 200}], customer:"Lisa" , total:700 , couponused:true })*

*db.sales.insertOne(*

*{ item: [{pid: "mousepad", qty: 1, price: 29}], customer:"Bob" , total:29, couponused:true })*

*db.sales.insertOne(*

*{ item: [{pid: "mat", qty: 1, price: 99}], customer:"Bob" , total:99, couponused:false })*

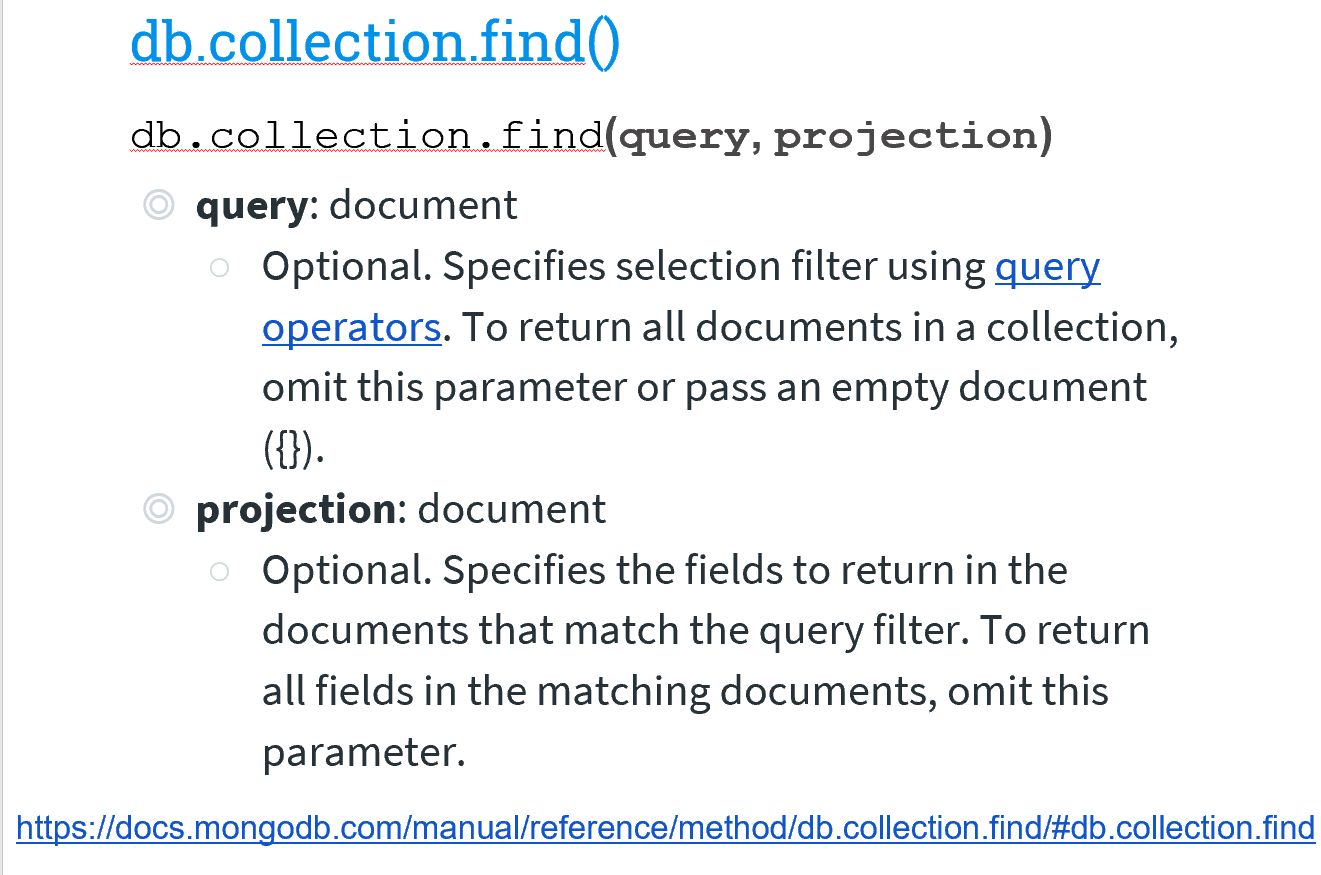


3.5 Update customer add status field

|  |  |
| --- | --- |
| Alter table customer  add status char(1)  +  Update customer  set status =”A” | db.customer.updateMany(  { },  {$set: {status: **“A”**}}  ) |
| Update customer  set status =”I”  where name =”Lisa” | db.customer.updateMany(  {name:”Lisa” },  {$set: {status: **“I”**}}  ) |

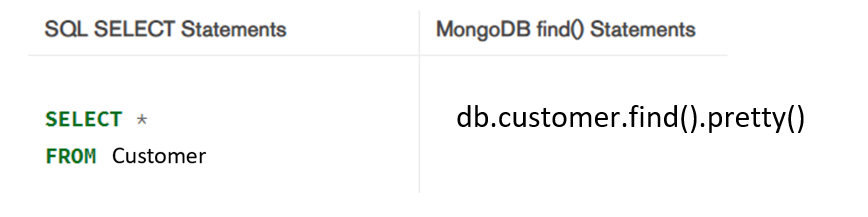


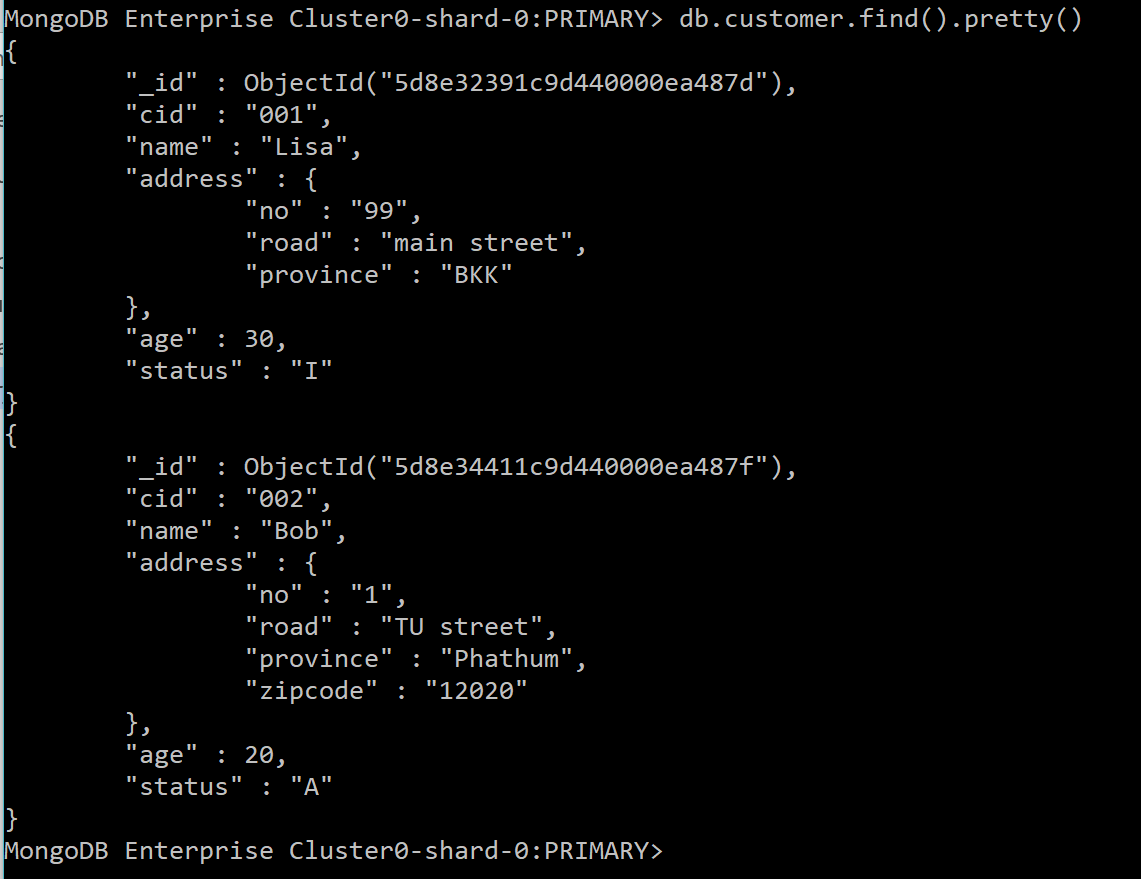
### Query Data



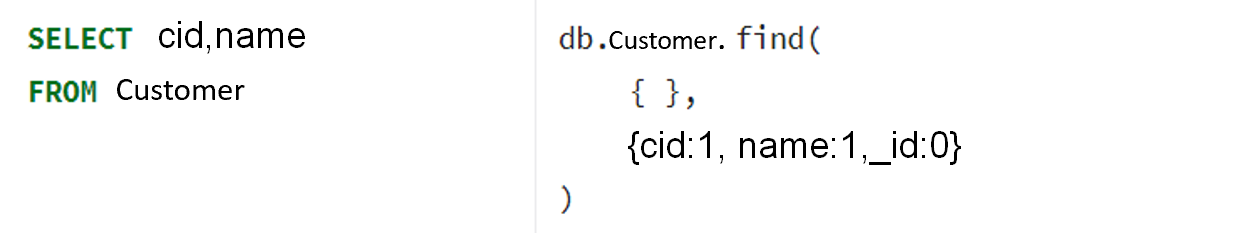
*Note: pretty() - displays the results in a formatted way.*

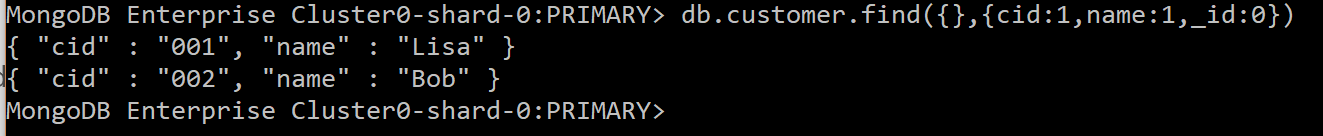
* 1. Select





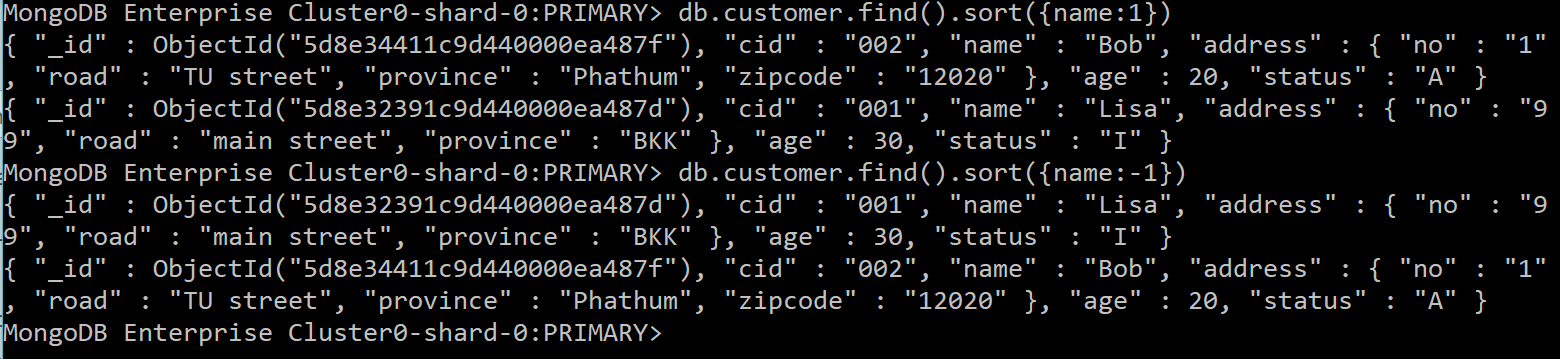
* Projection (return some fields)



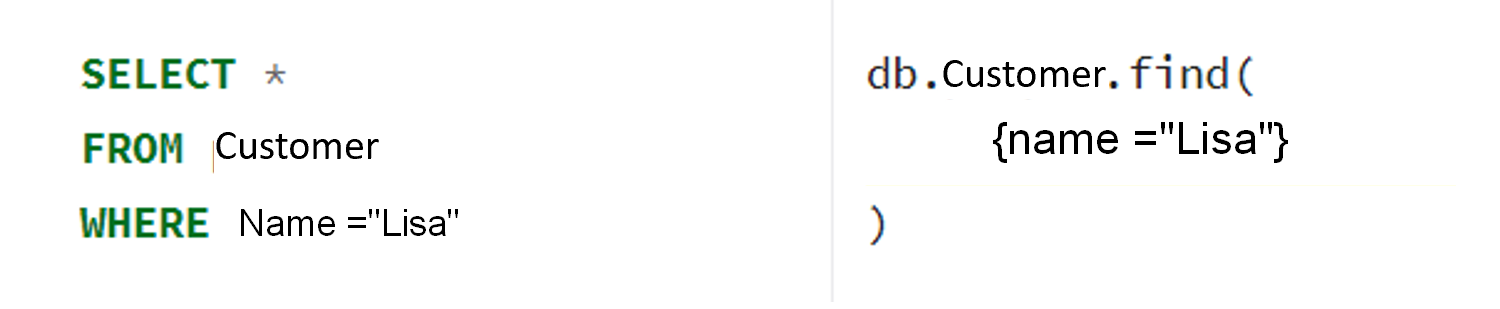


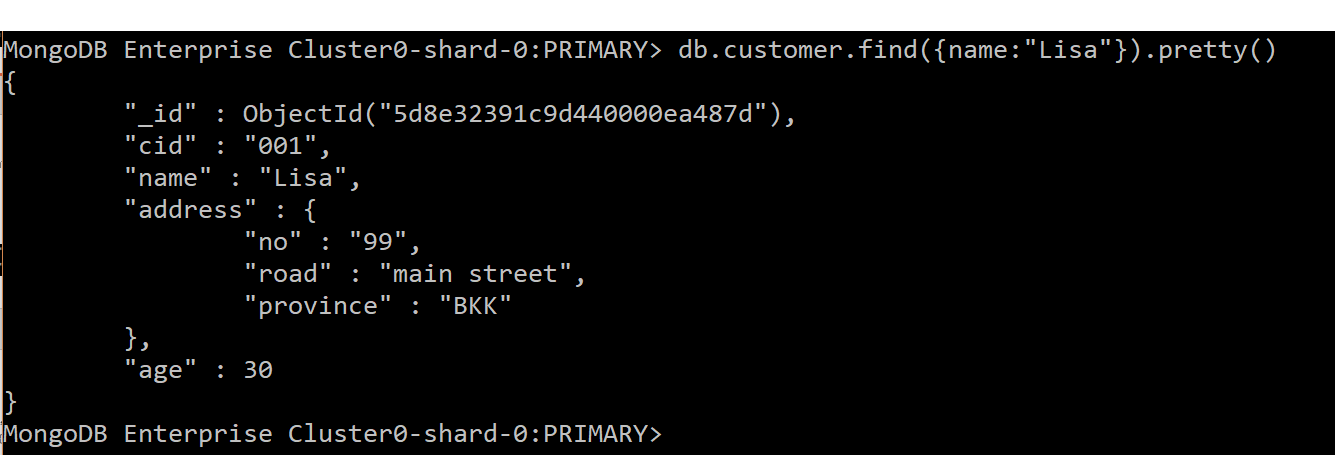
* 1. Order by

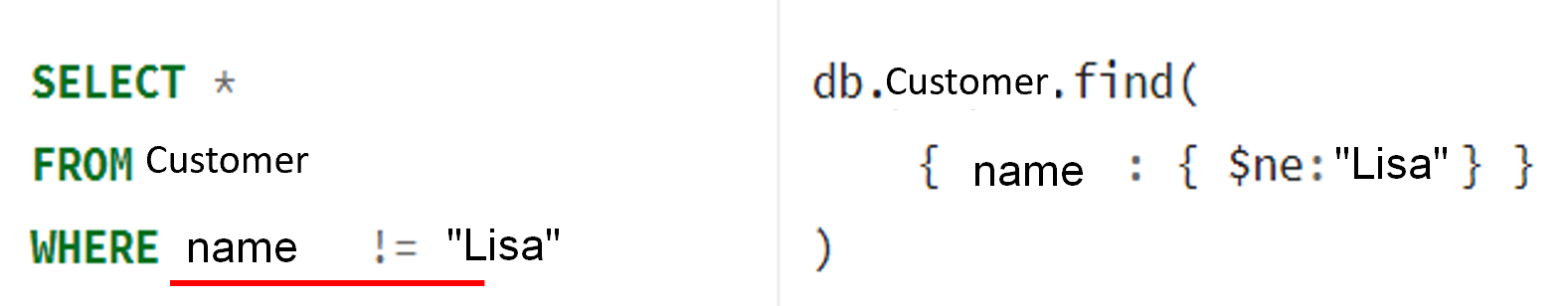
|  |  |
| --- | --- |
| Select \* from customer order by name | db.customer.find().sort({name:1}) |
| Select \* from customer order by name desc | db.customer.find().sort({name:-1}) |

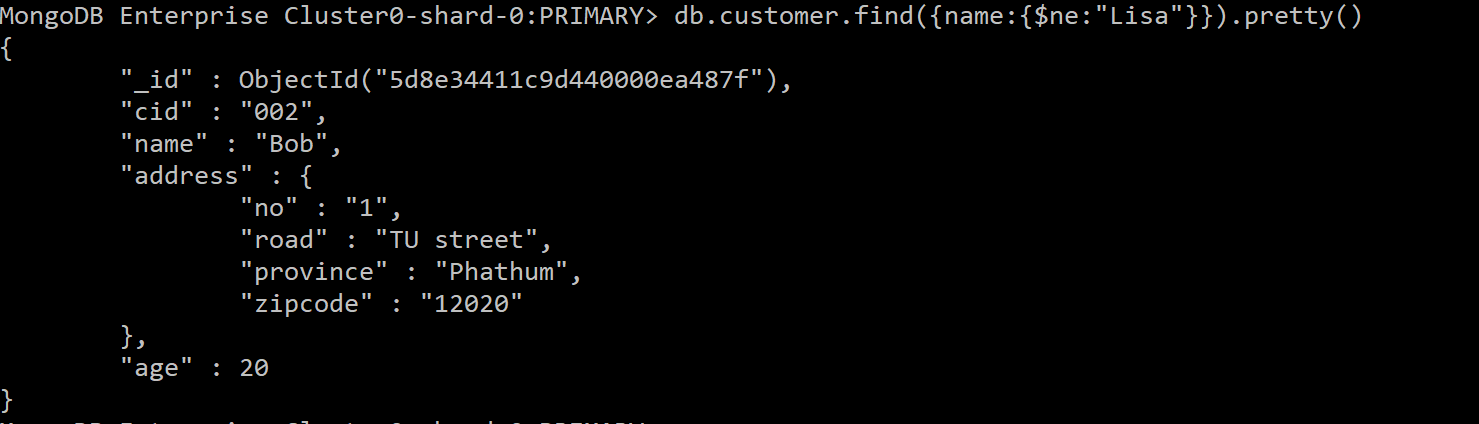


* 1. Where condition
     1. =, !=, >,<, <=,>=

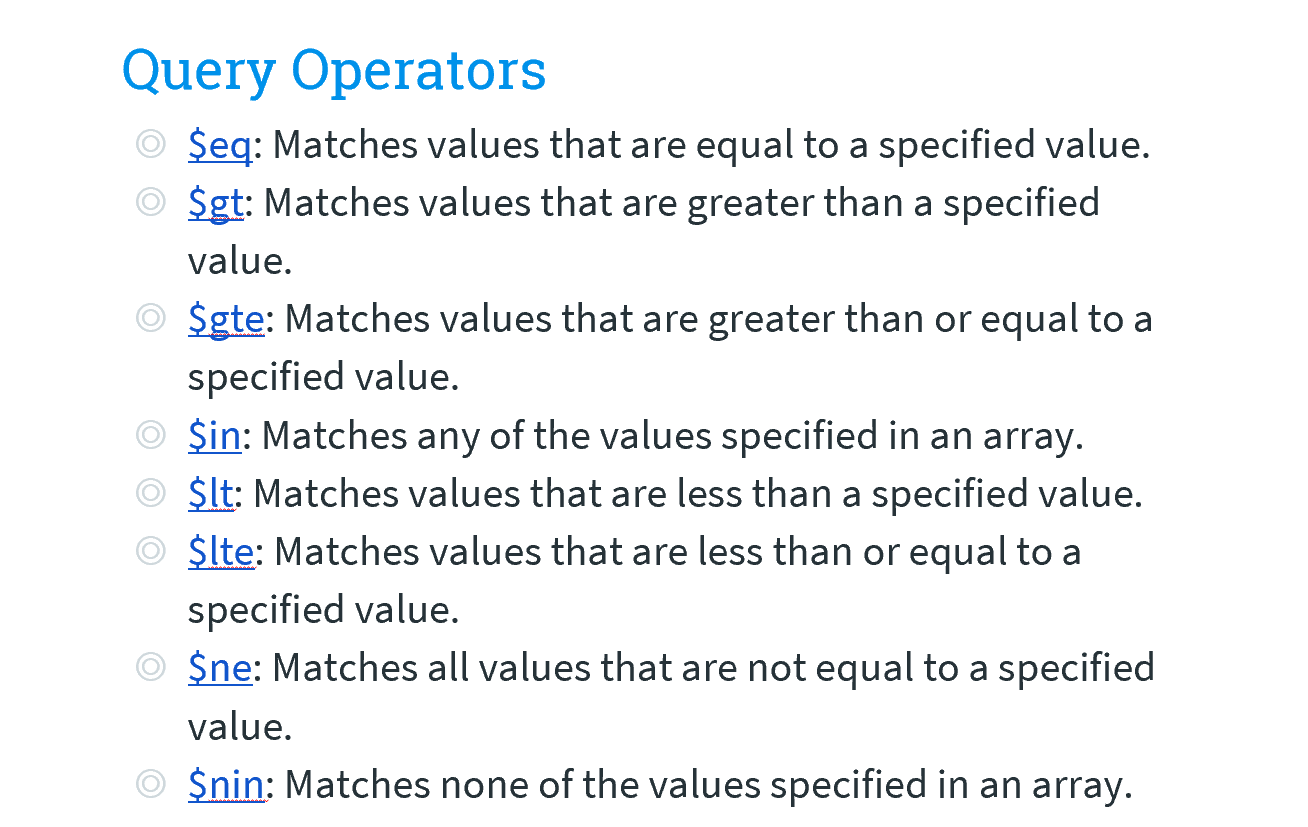




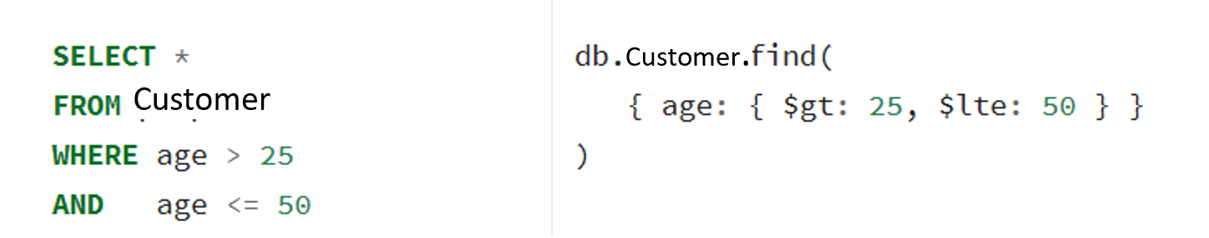






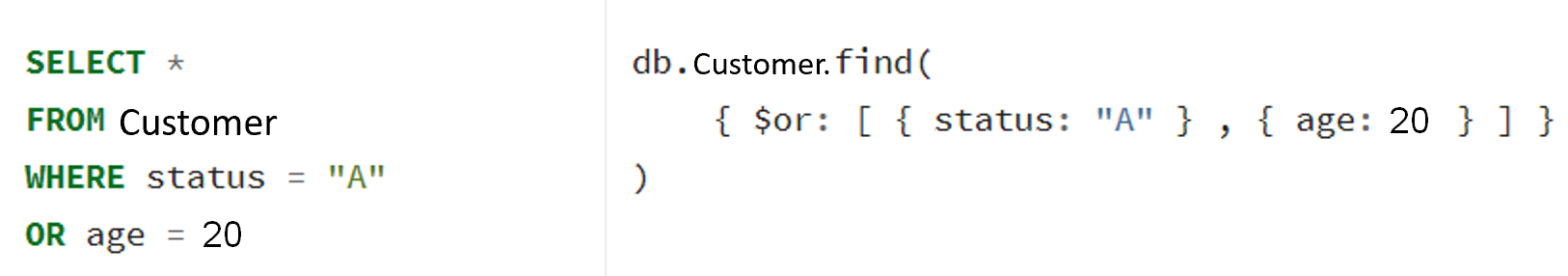


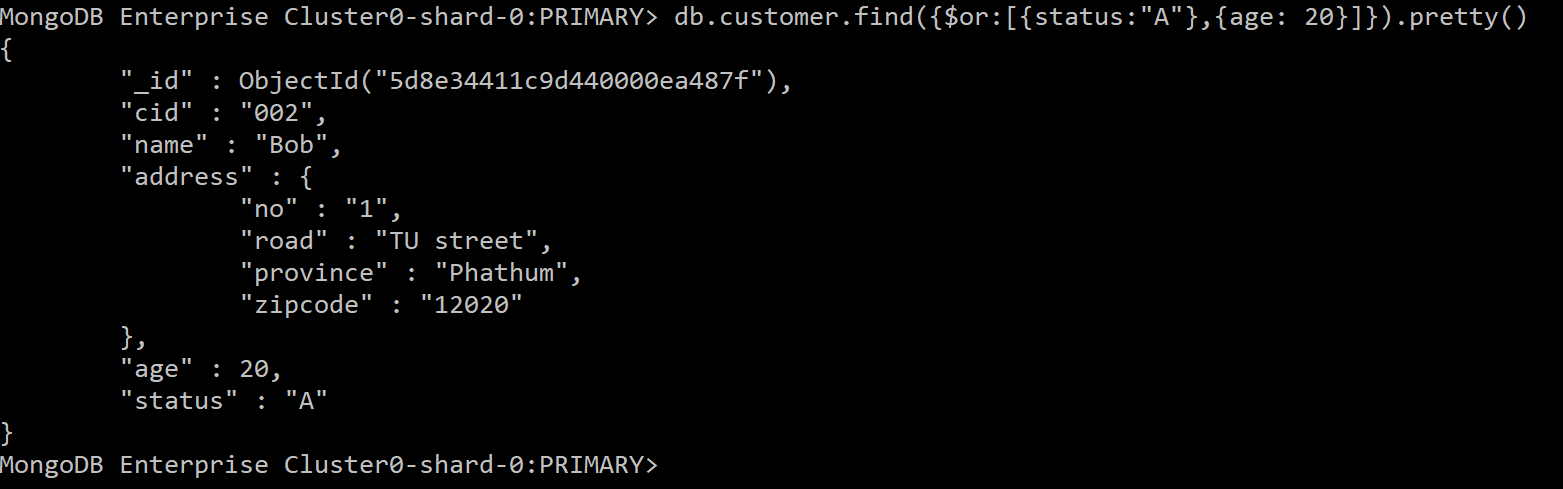
* + 1. AND



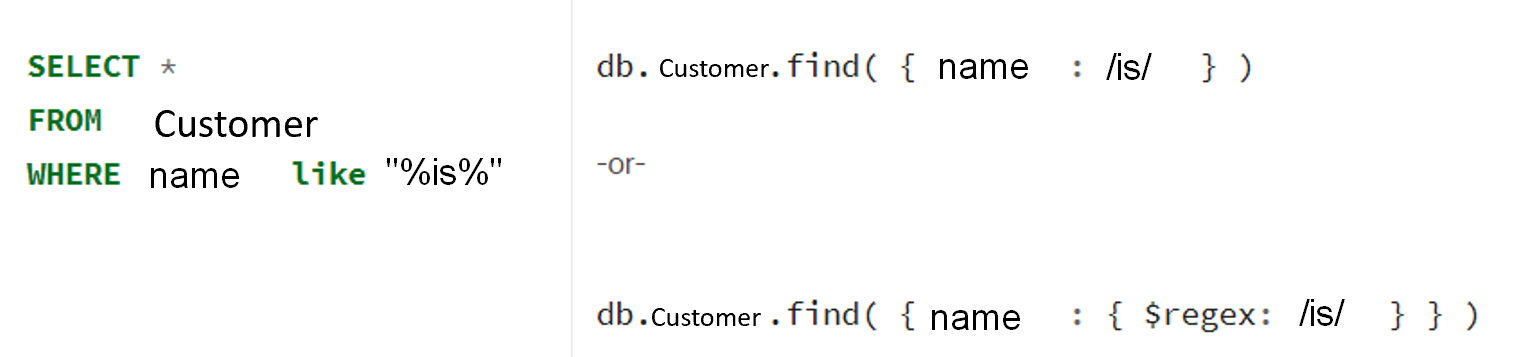
db.customer.find({$and : [{status: "A"}, {age:20}]})

* + 1. OR



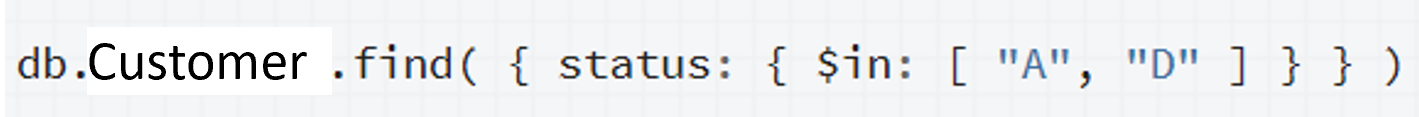


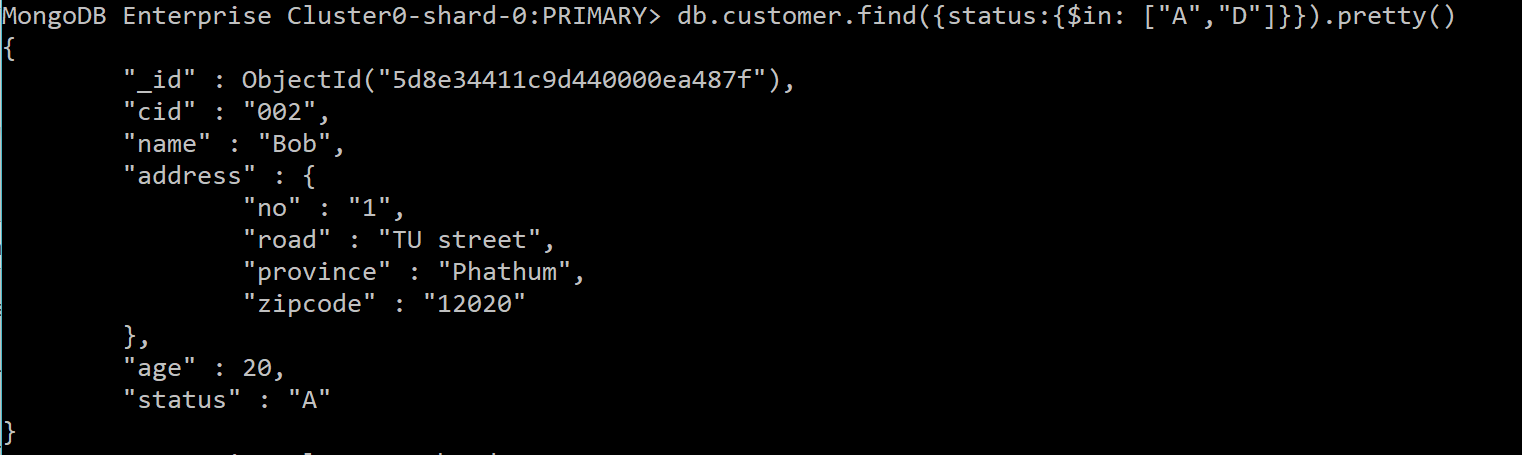
* + 1. LIKE





* + 1. IN





* + 1. Exists

Returns all the documents from the customer collection where zipcode field does not exists:

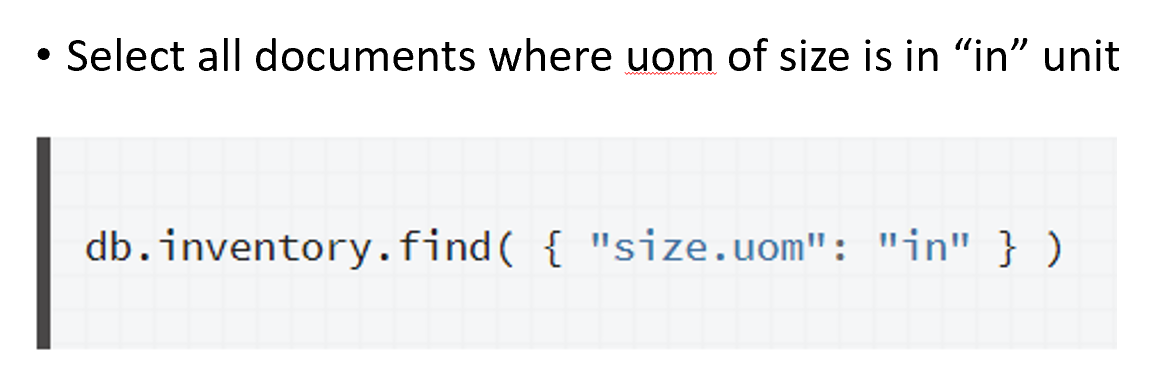
db.customer.find( {

address.zipcode: { $exists: false }

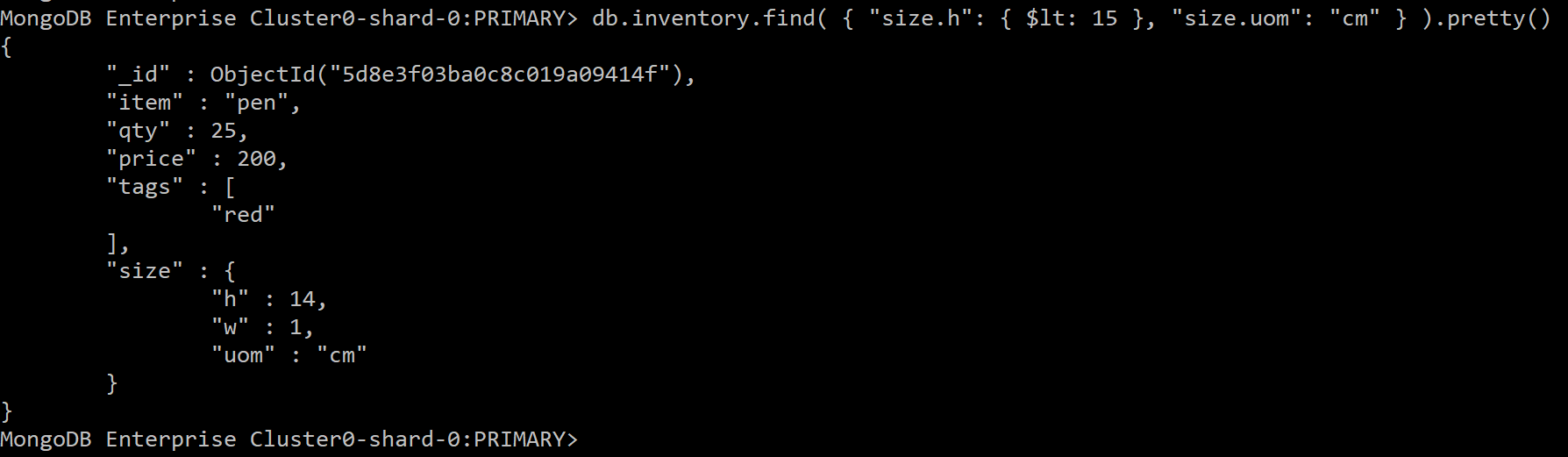
} )

* 1. Query on Nested field

Uses **dot notation** to access fields in an embedded document:



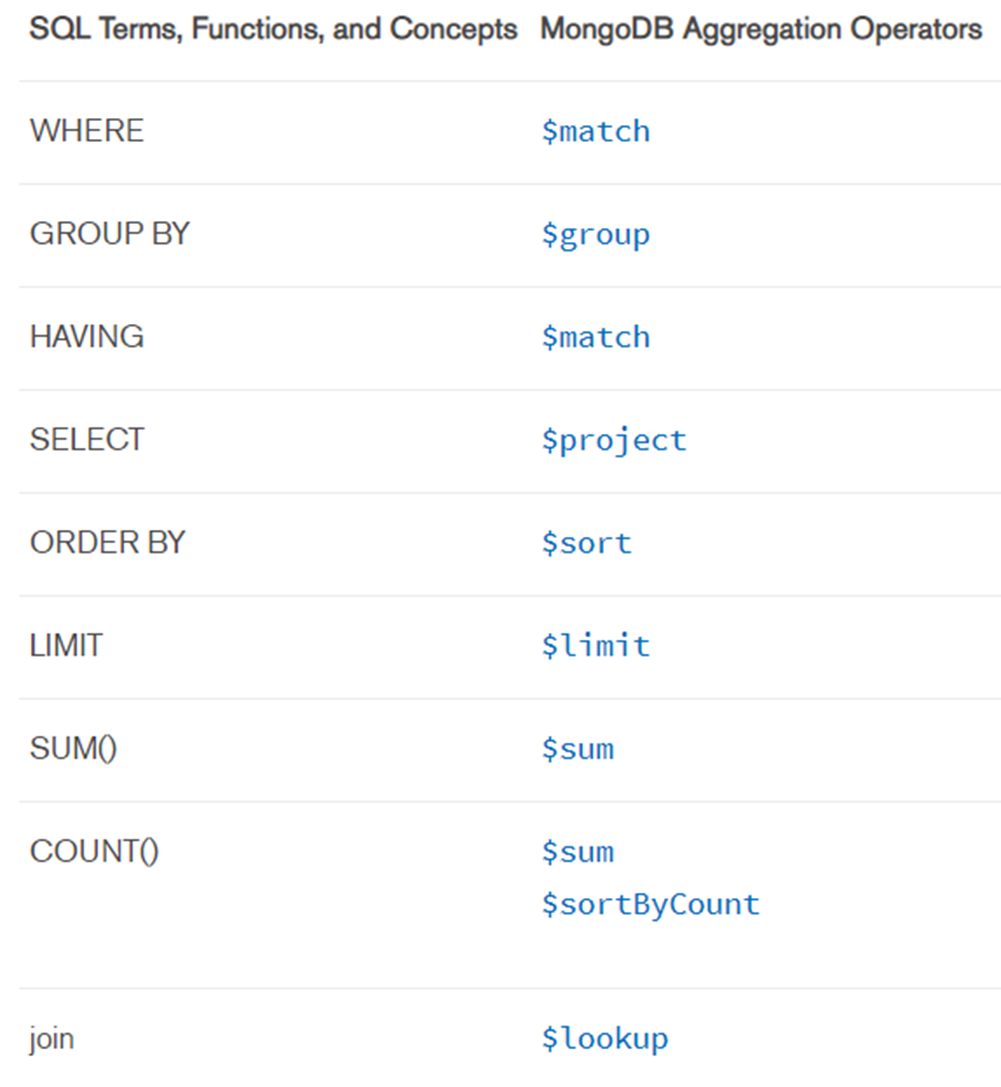
* selects items where their height less than 15 CM.



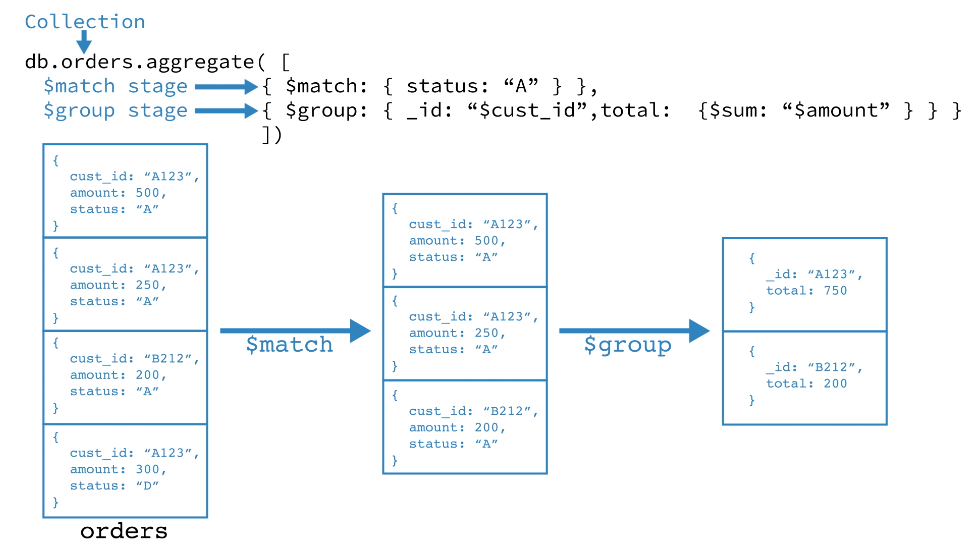
.AGGREGATE

* 1. Aggregate

Aggregation operations group values from multiple documents together and can perform a variety of operations on the grouped data to return a single result.



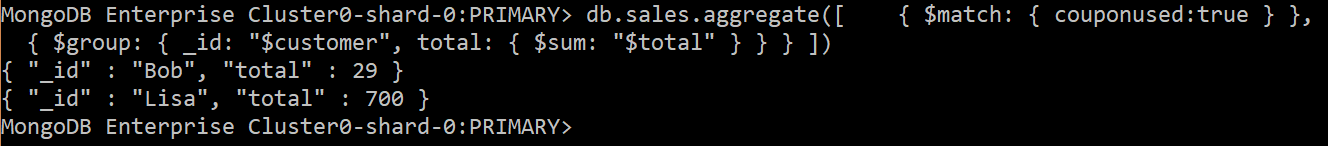
Example



* + 1. Sum

db.sales.aggregate([ { $match: { couponused:true } },   
 { $group: { \_id: "$customer", total: { $sum: "$total" } } }

])

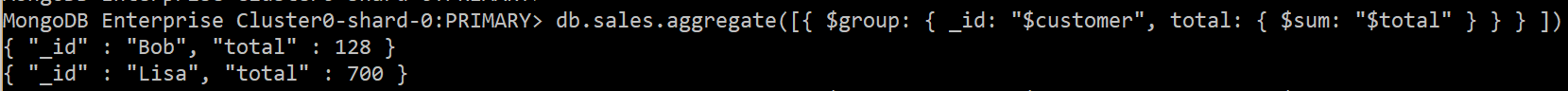


All sales record:

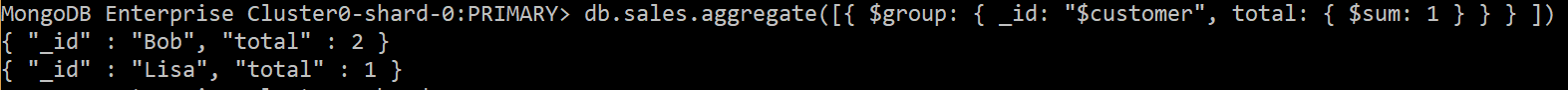


db.sales.aggregate([   
 { $group: { \_id: "$customer", total: { $sum: "$total" } } }

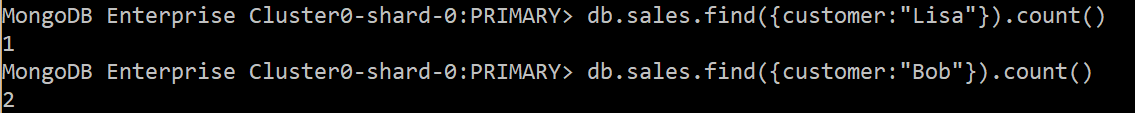
])



* + 1. Count



*Note* .FIND() also has COUNT() function



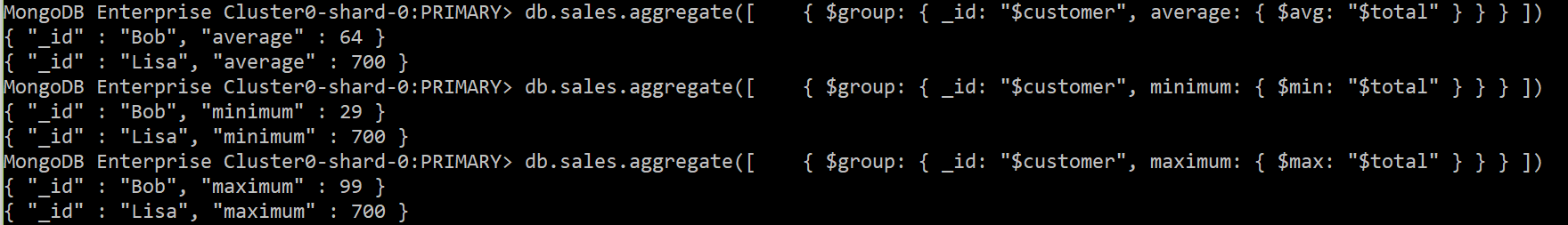
* + 1. Average, Min, Max



db.sales.aggregate([   
 { $group: { \_id: "$customer", average: { $avg: "$total" } } }   
 ])

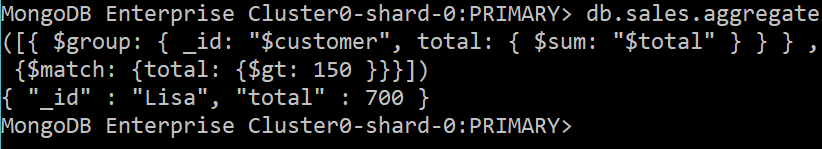
db.sales.aggregate([   
 { $group: { \_id: "$customer", minimum: { $min: "$total" } } }   
 ])

db.sales.aggregate([   
 { $group: { \_id: "$customer", maximum: { $max: "$total" } } }   
 ])



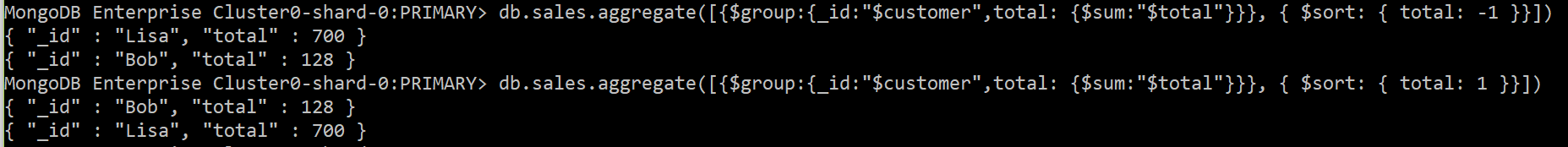
* + 1. Having

db.sales.aggregate([{ $group: { \_id: "$customer", total: { $sum: "$total" } } } ,  
 {$match: {total: {$gt: 150 }}}  
 ])

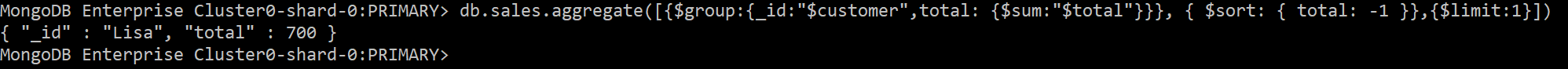


* + 1. Sort and Limit

db.sales.aggregate([  
{$group:{\_id:"$customer",total: {$sum:"$total"}}},   
{ $sort: { total: -1 } }  
])



db.sales.aggregate([  
{$group:{\_id:"$customer",total: {$sum:"$total"}}},   
{ $sort: { total: -1 }},  
{$limit:1}  
])



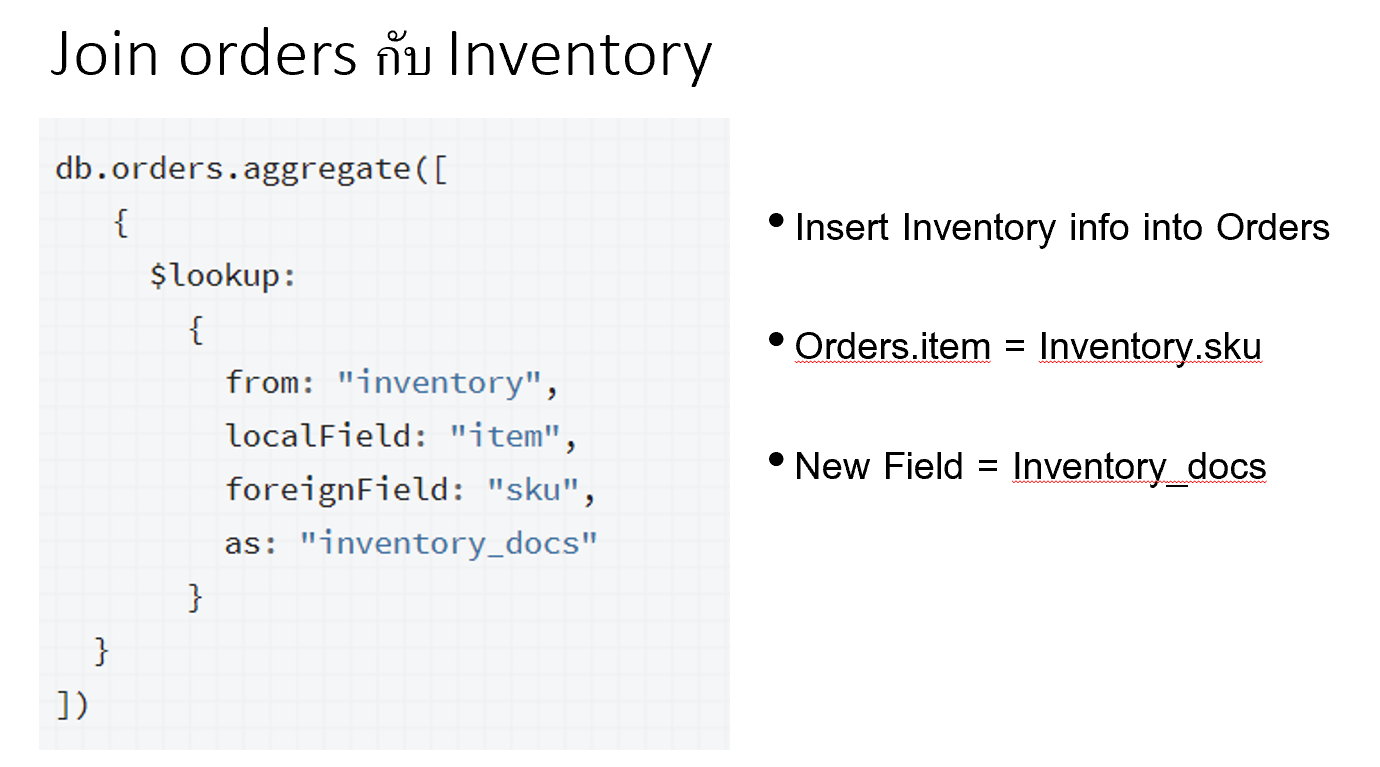
* 1. Join



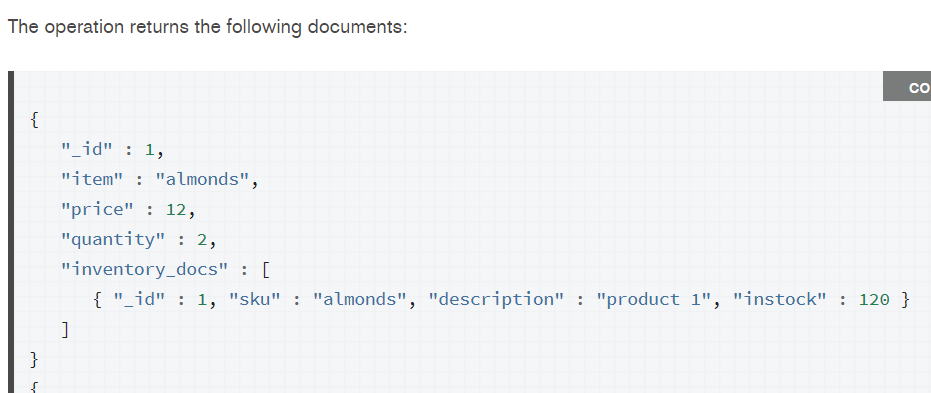
Example from [https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/index.html#lookup-single-equality](https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/index.html) :



Command:



Result:



Try our database:

* Include customer info into sale

db.sales.aggregate([

{

$lookup:

{

from: "customer",

localField: "customer",

foreignField: "name",

as: "cust\_info"

}

}

]).pretty()



* Include the info under customer in sale

db.sales.aggregate([

{

$lookup:

{

from: "customer",

localField: "customer",

foreignField: "name",

as: "customer.info"

}

}

])



**Reference:**

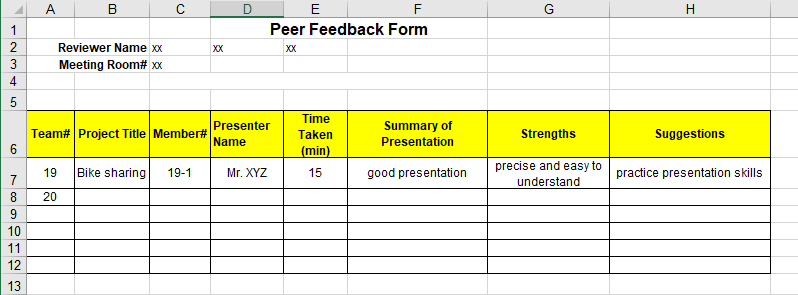
<https://docs.mongodb.com/manual/crud/>  
<https://docs.mongodb.com/manual/>  
<https://docs.mongodb.com/manual/reference/sql-comparison/#examples>  
<https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/index.html#lookup-single-equality>

Lab Assignment

1. Install Mongo Shell and connect to RDBProject database using the below command

***mongo "mongodb+srv://dmmcluster.f1uoi.gcp.mongodb.net/RDBProject" --username st\_dmm***

1. Convert your Peer-review feedback into the below JSON format and insert into feedback collection



{  
"member\_id": "19-1",  
"presenter\_name": "Mr. XYZ",  
"project\_id" : "19",   
"project\_name": "Bike sharing",  
"summary\_of\_presentation": "the presentation was well prepared and\_ ",  
"strengths": [ "precise", "easy to understand" ],  
"suggestions": ["practice presentation skills" ]  
…  
}

… means you can add any other information because NOSQL is schemaless.

1. Insert the peer review into the feedback table in the database.
2. Write MongoDB command to retrieve the following information, capture the running screen and the returned result.   
   [TASK1] List your presentation feedback reviewed by friends.

[TASK2] How many strengths are provided in total?

[TASK3] What is the average number of suggestion for a reviewer?

Submission System: Google Classroom

Total TASKS: 3