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

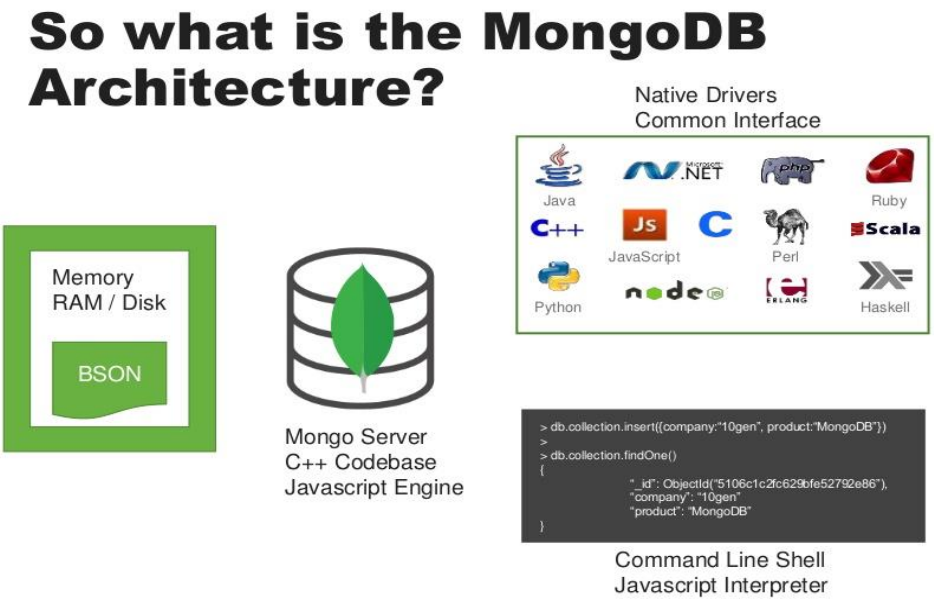
1. MongoDB Overview

1.1. 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

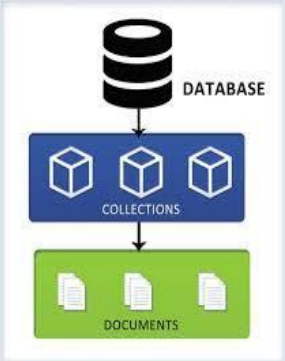
Ref. Examples of MongoDB users: https://www.mongodb.com/mongodb-scale

1.2. MongoDB Architecture



1.3. MongoDB Terminology

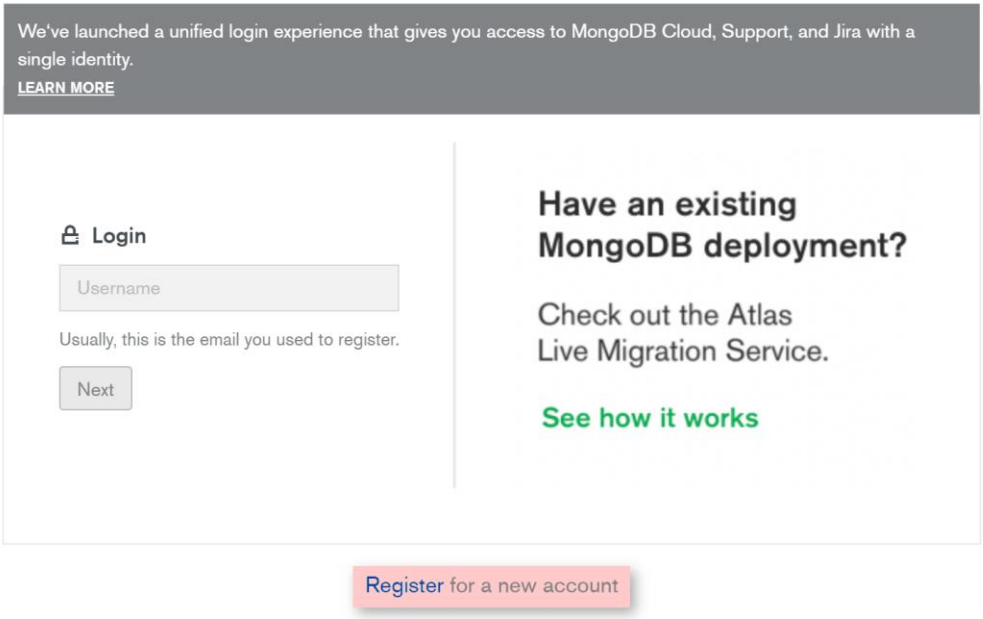
RDBMS		MongoDB
Database	→	Database
Table	→	Collection
Index	→	Index
Row	→	Document
Join	→	Embedding & Linking

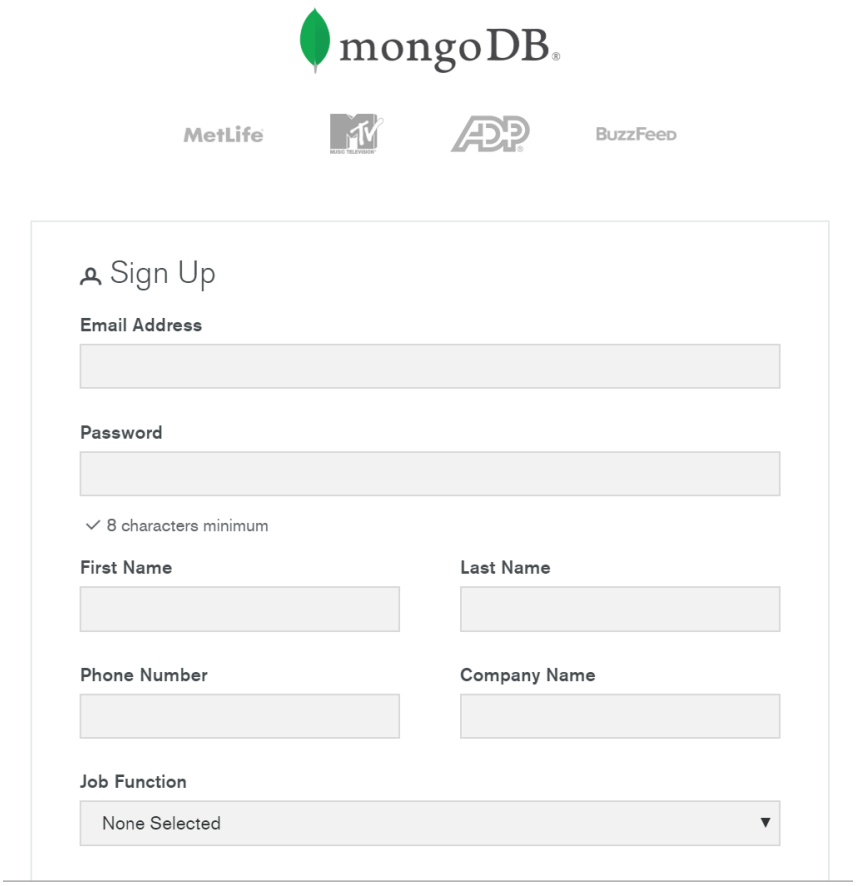


2. Set up our cluster (= Database Server) in Mongo DB cloud

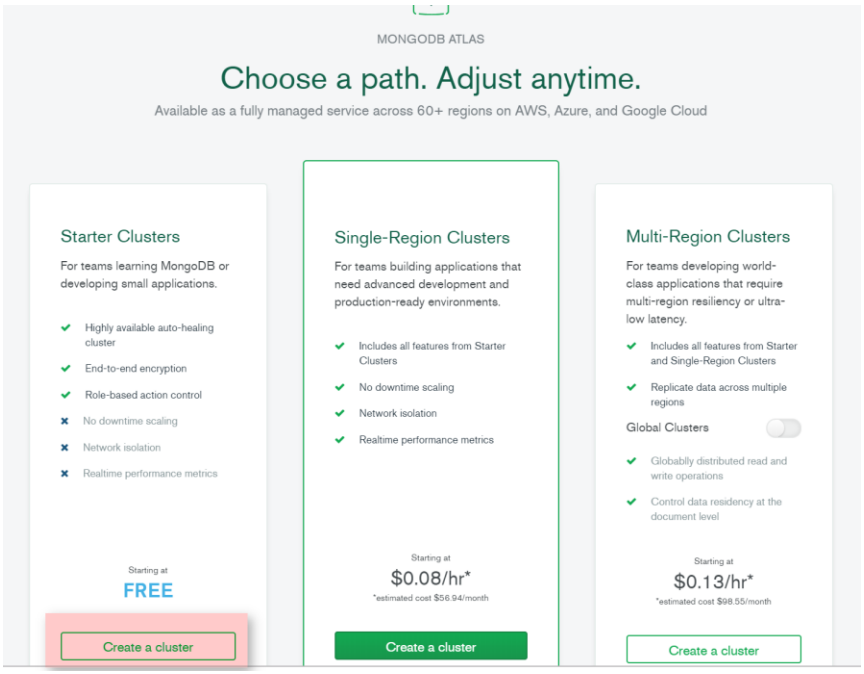
https://cloud.mongodb.com/

2.1 Sign up

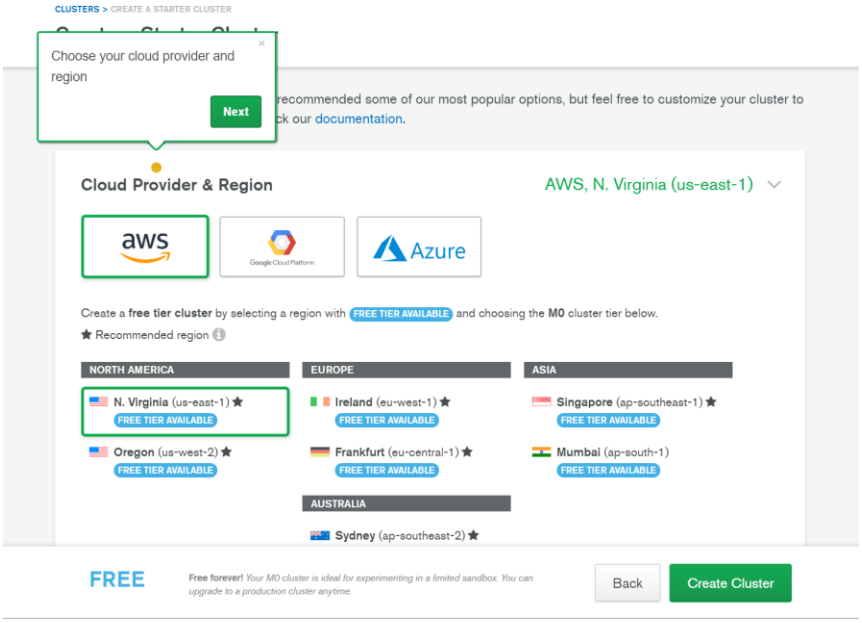


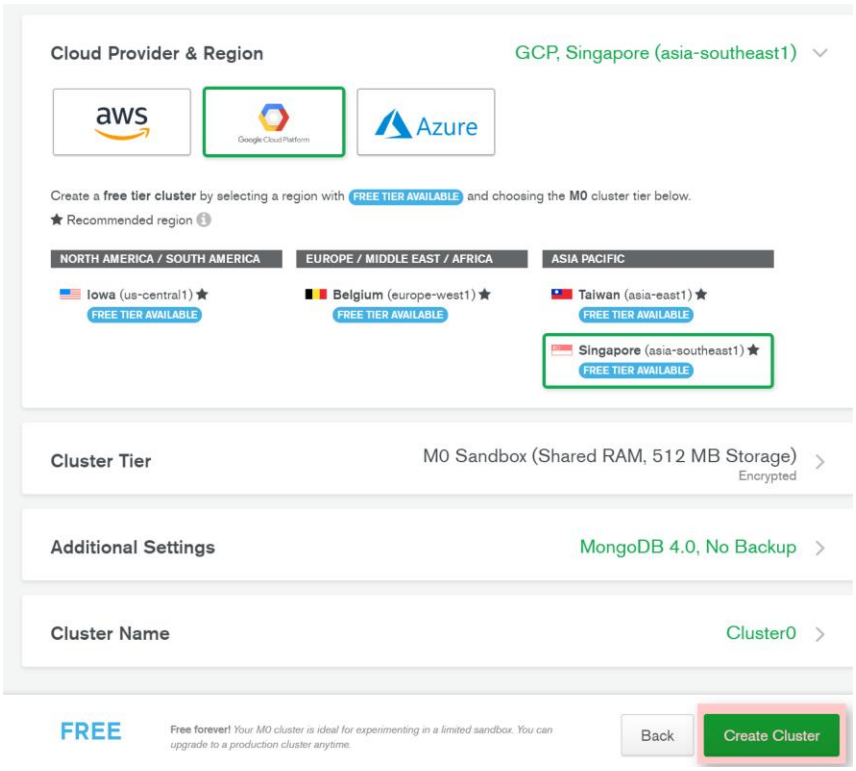


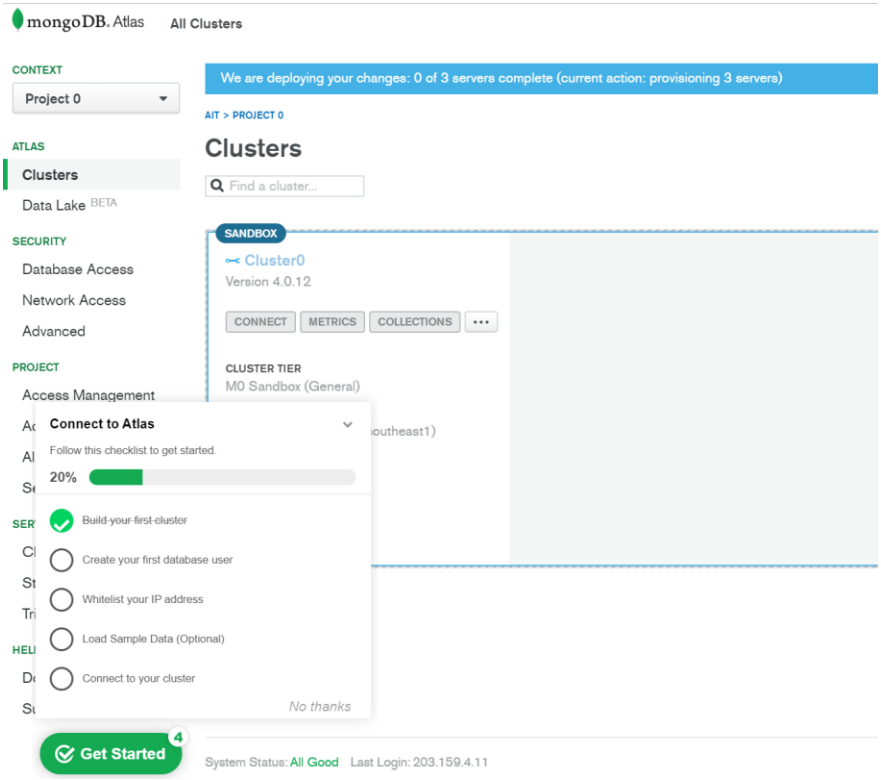
2.2 Create a Cluster



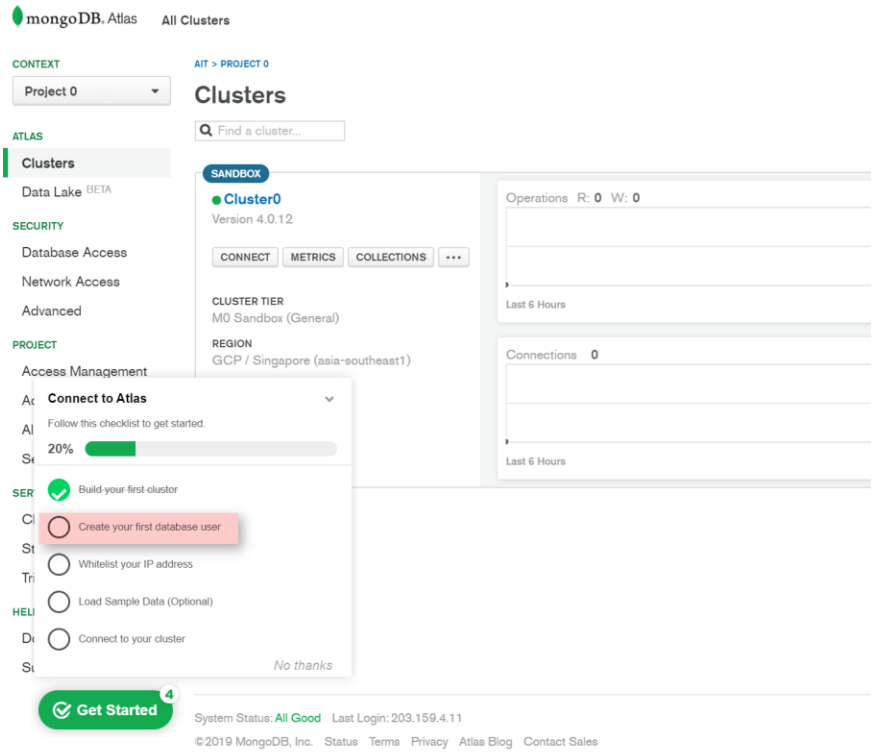
Select Cloud Provider and Region and Click “Create Cluster”

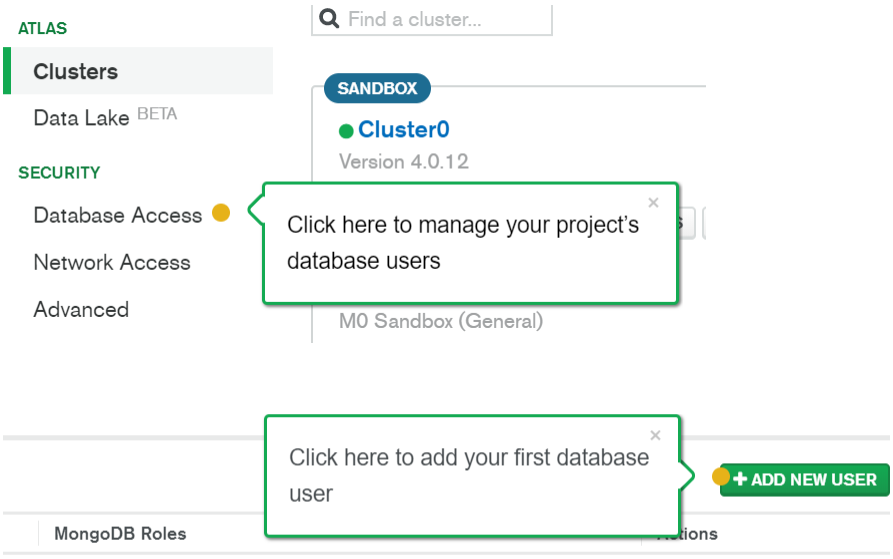


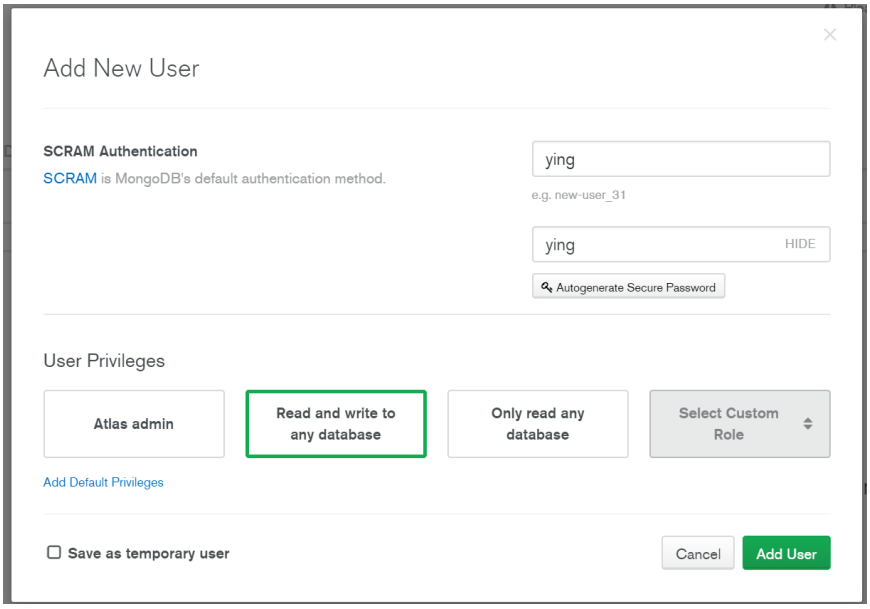


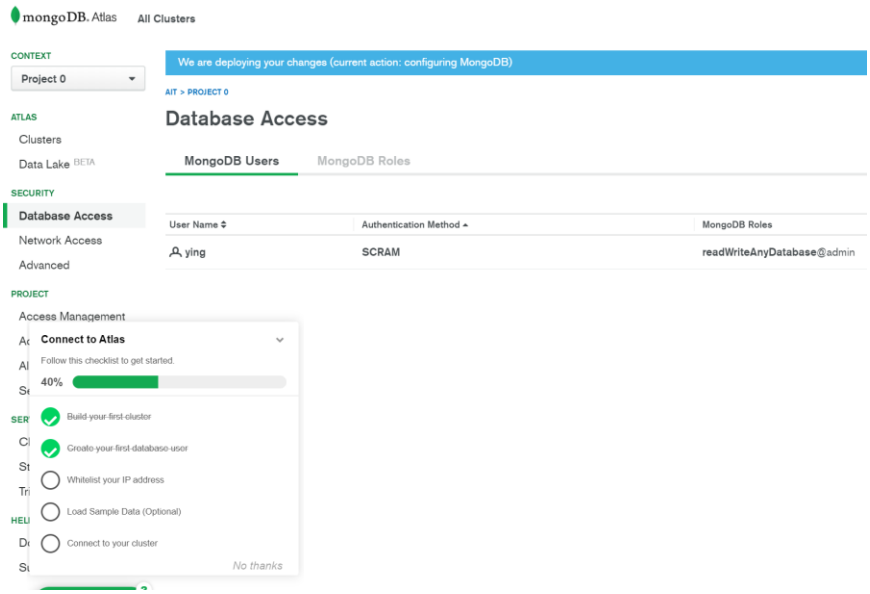


2.3 Create your first database user

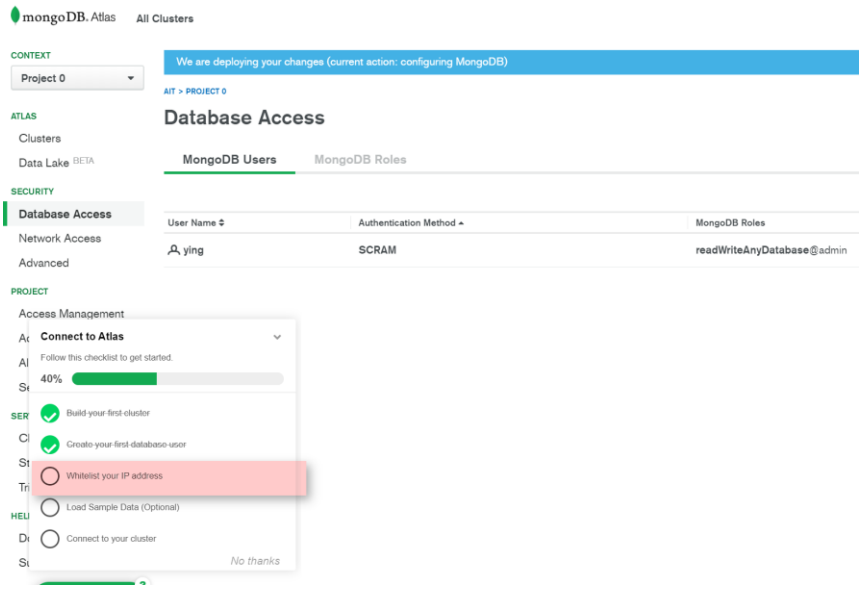


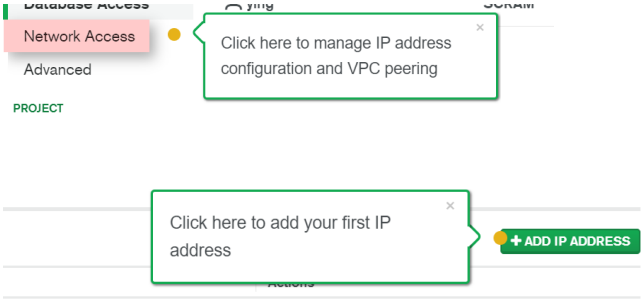

ATLAS
Clusters
Data Lake BETA
SECURITY
Database Access
Network Access
Advanced
Find a cluster...
SANDBOX
Cluster0
Version 4.0.12
Click here to manage your project's database users
M0 Sandbox (General)
Click here to add your first database user
+ ADD NEW USER
MongoDB Roles

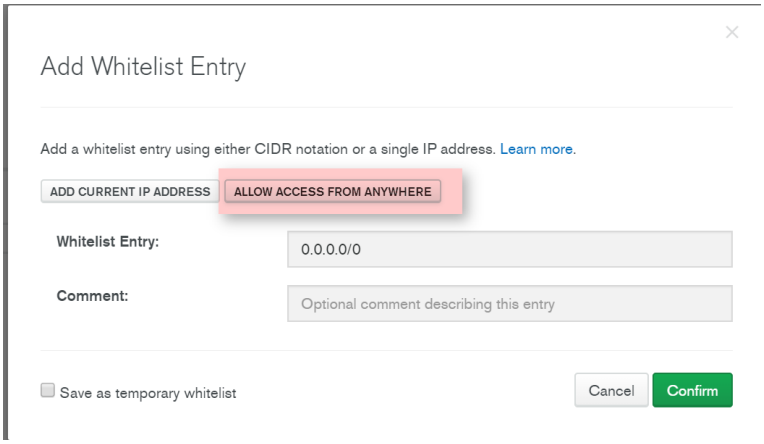

Add New User
SCRAM Authentication
SCRAM is MongoDB's default authentication method.
ying
e.g. new-user_31
ying
HIDE
Autogenerate Secure Password
User Privileges
Atlas admin
Read and write to any database
Only read any database
Select Custom Role
Add Default Privileges
Save as temporary user
Cancel
Add User



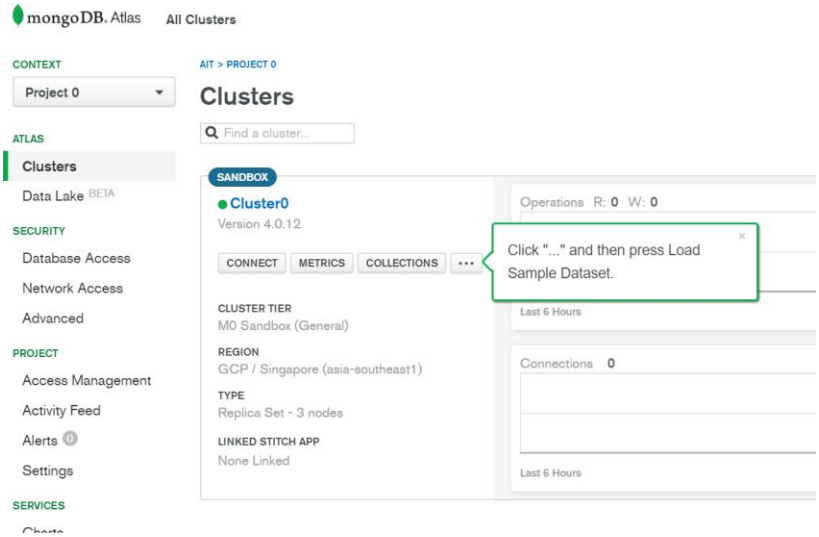
1.4 Allow Hosts that can access the Cluster

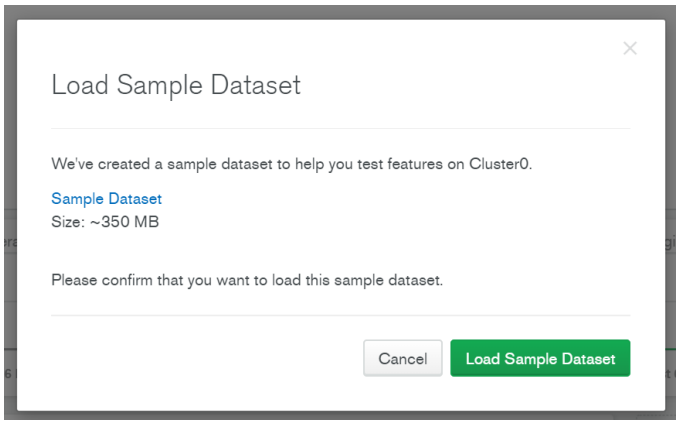




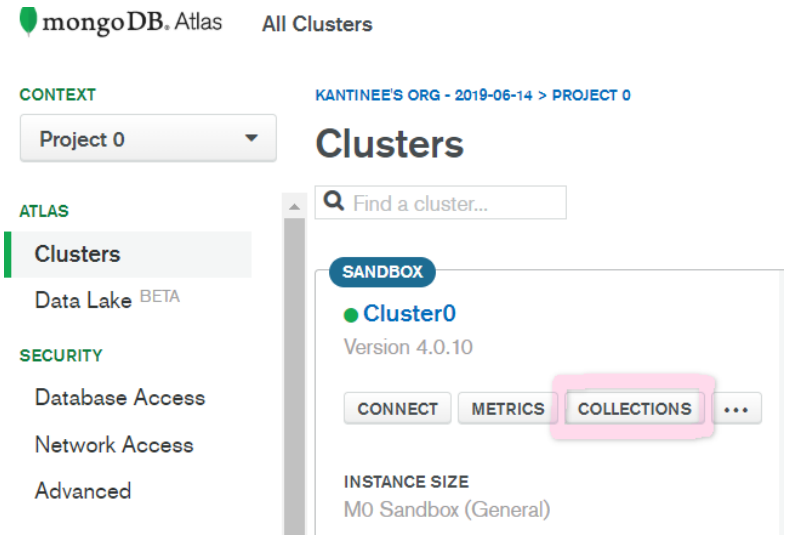


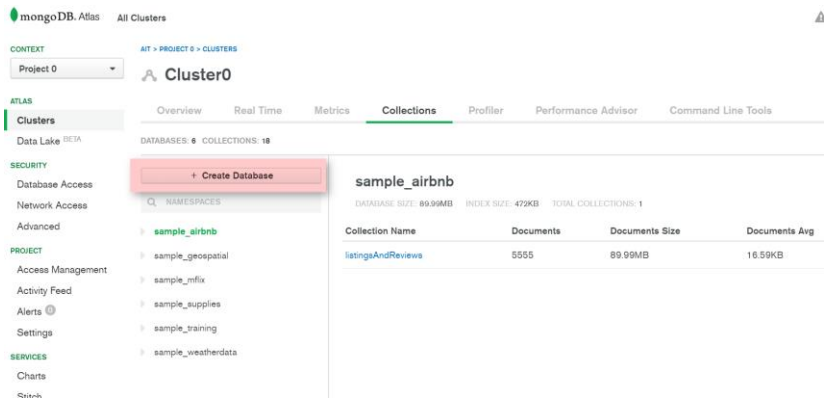
1.5 (Optional) Load Sample DB



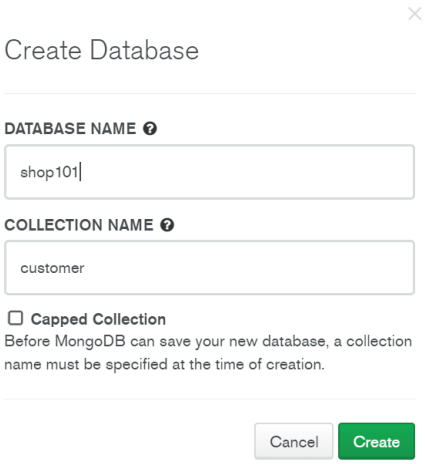


1.6 Create Database in Mongo Atlas

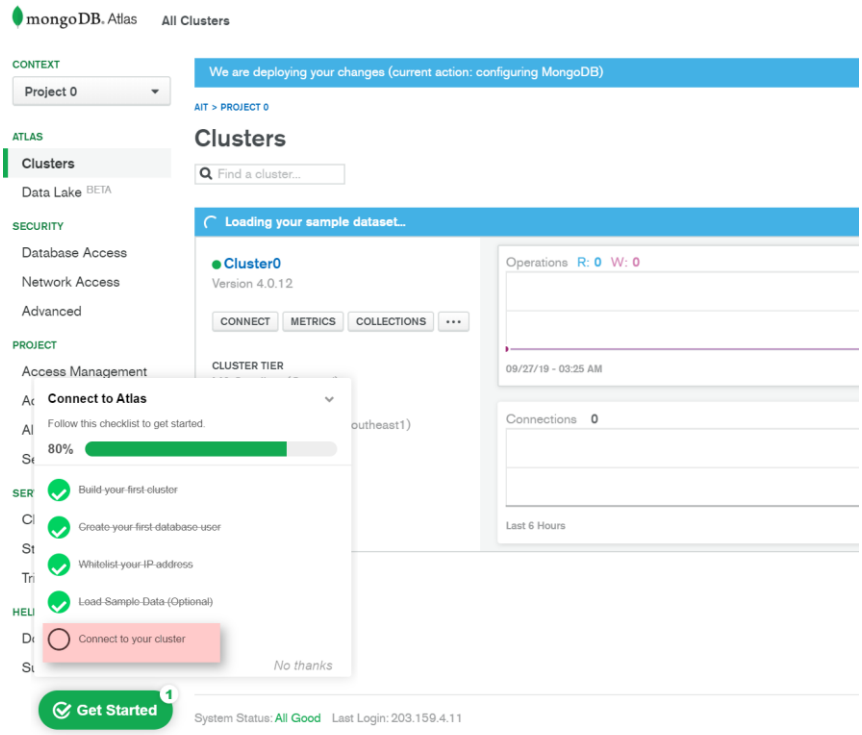


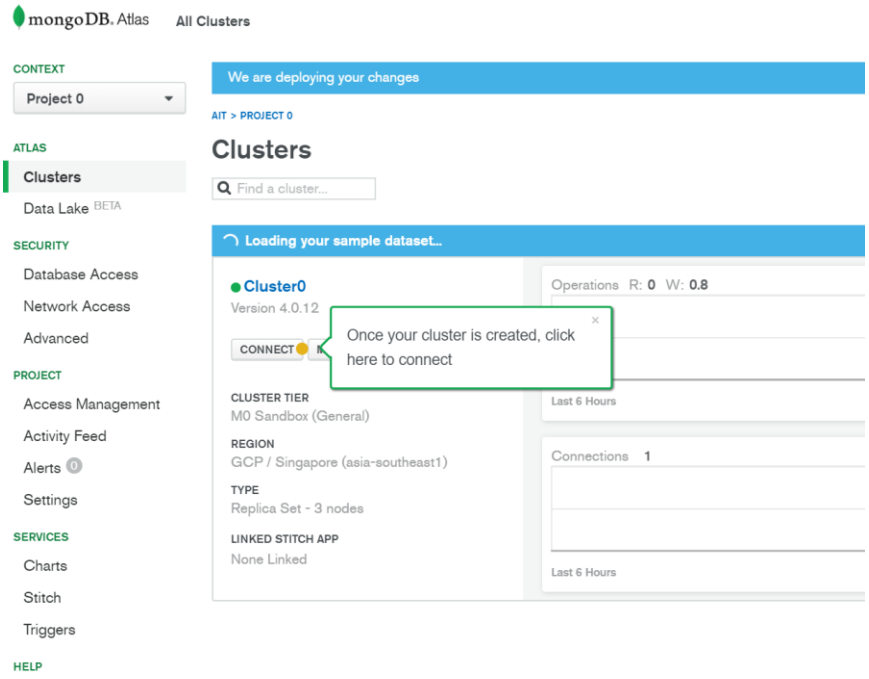


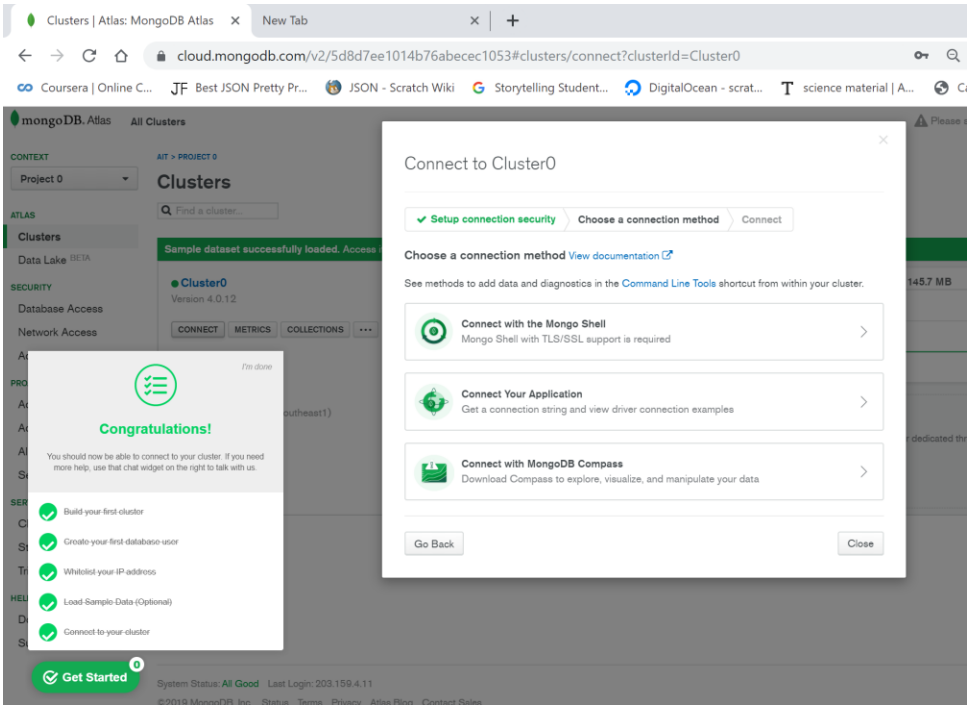
Create Database name =shop101 and Collection name = customer.



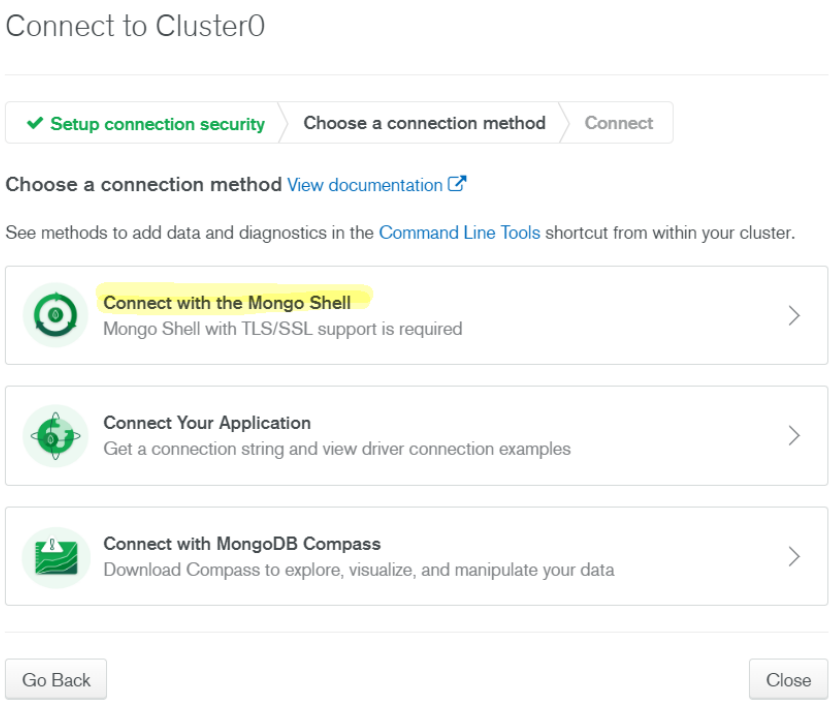
1.7 Connect to the Cluster using the User. There are 3 methods to connect: Mongo Shell, Your App and Mongo Compass.

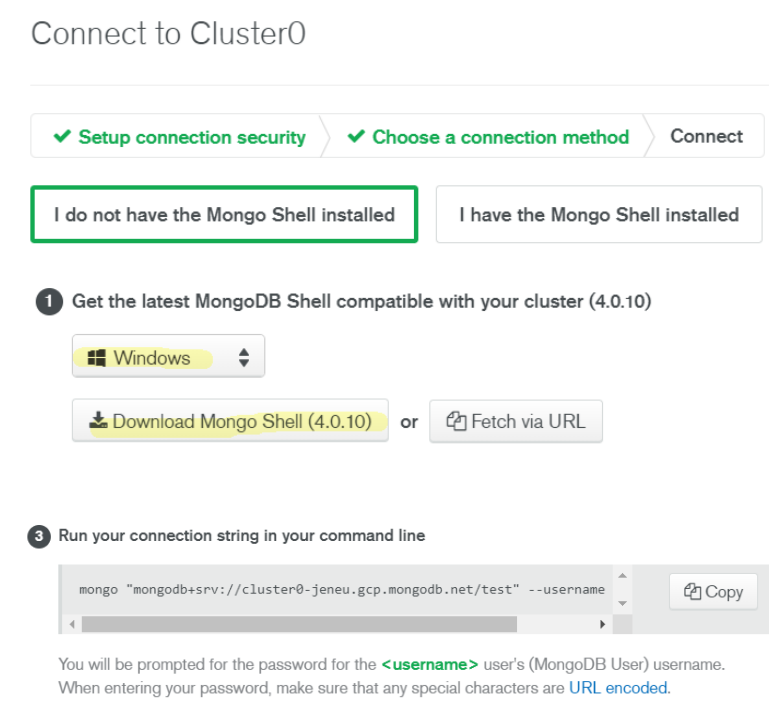


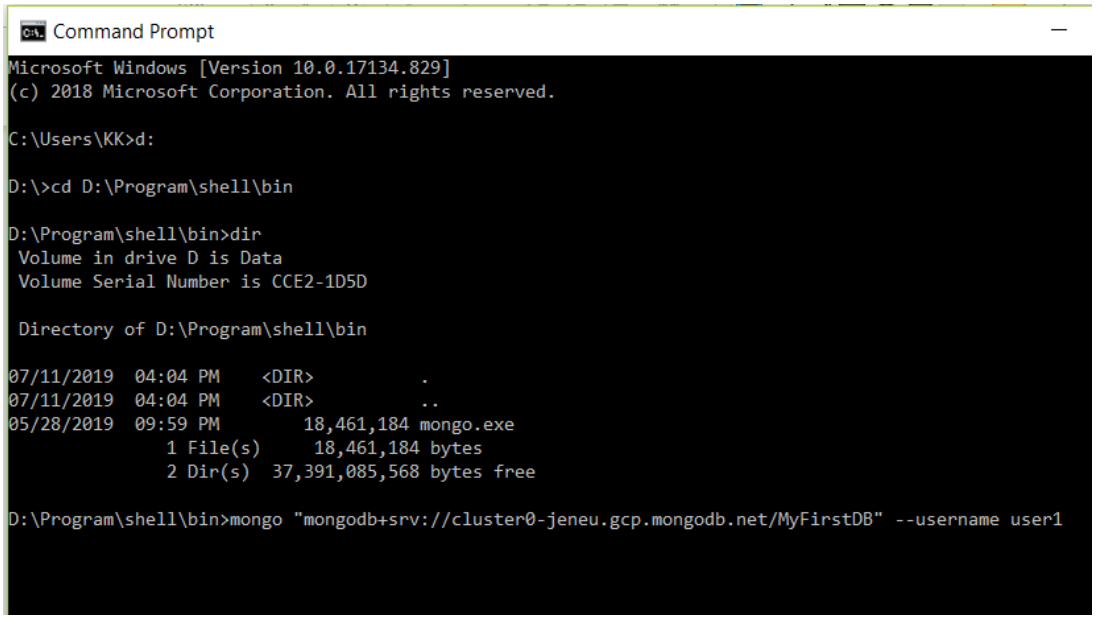


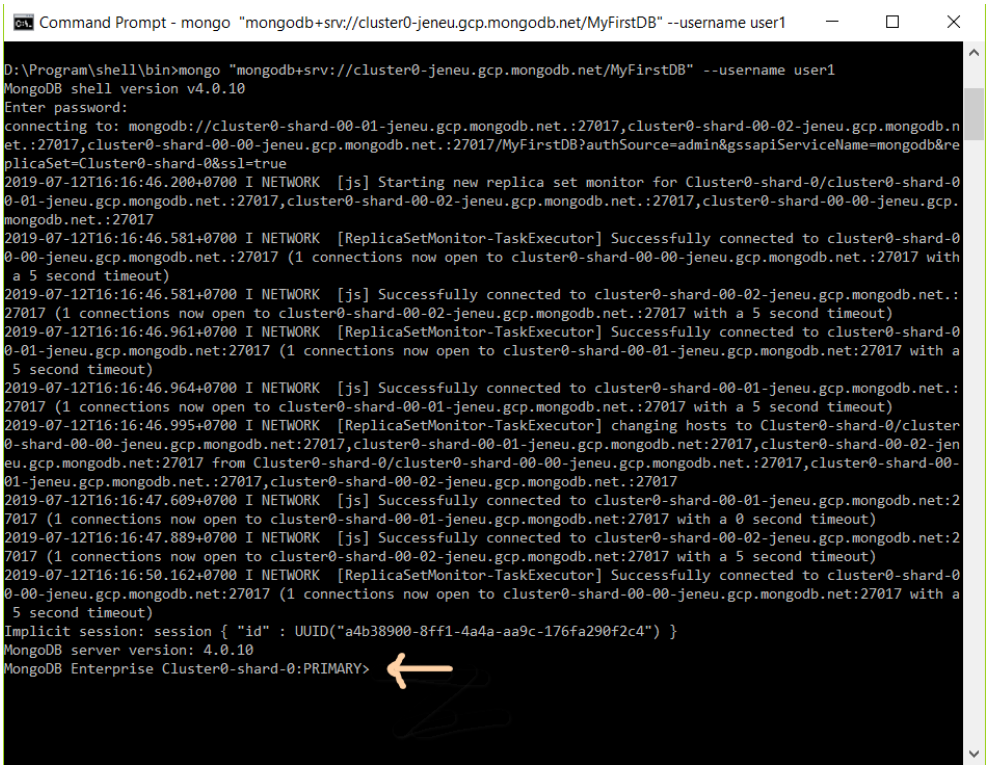


Install Shell or Compass and using the connection string to connect. Example of Connection string: mongodb+srv://ying:<password>@cluster0-dfi1d.gcp.mongodb.net/<database name>
In this tutorial we will use Shell




Connect to Cluster0
Setup connection security
Choose a connection method
Connect
I do not have the Mongo Shell installed
I have the Mongo Shell installed
1 Get the latest MongoDB Shell compatible with your cluster (4.0.10)
Windows
Download Mongo Shell (4.0.10)
or
Fetch via URL
3 Run your connection string in your command line
mongo "mongodb+srv://cluster0-jeneu.gcp.mongodb.net/test" --username
Copy
You will be prompted for the password for the <username> user's (MongoDB User) username.
When entering your password, make sure that any special characters are URL encoded.


Command Prompt
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Users\KK>d:
D:\>cd D:\Program\shell\bin
D:\Program\shell\bin>dir
Volume in drive D is Data
Volume Serial Number is CCE2-1D5D
Directory of D:\Program\shell\bin
07/11/2019 04:04 PM <DIR> .
07/11/2019 04:04 PM <DIR> ..
05/28/2019 09:59 PM 18,461,184 mongo.exe
1 File(s) 18,461,184 bytes
2 Dir(s) 37,391,085,568 bytes free
D:\Program\shell\bin>mongo "mongodb+srv://cluster0-jeneu.gcp.mongodb.net/MyFirstDB" --username user1



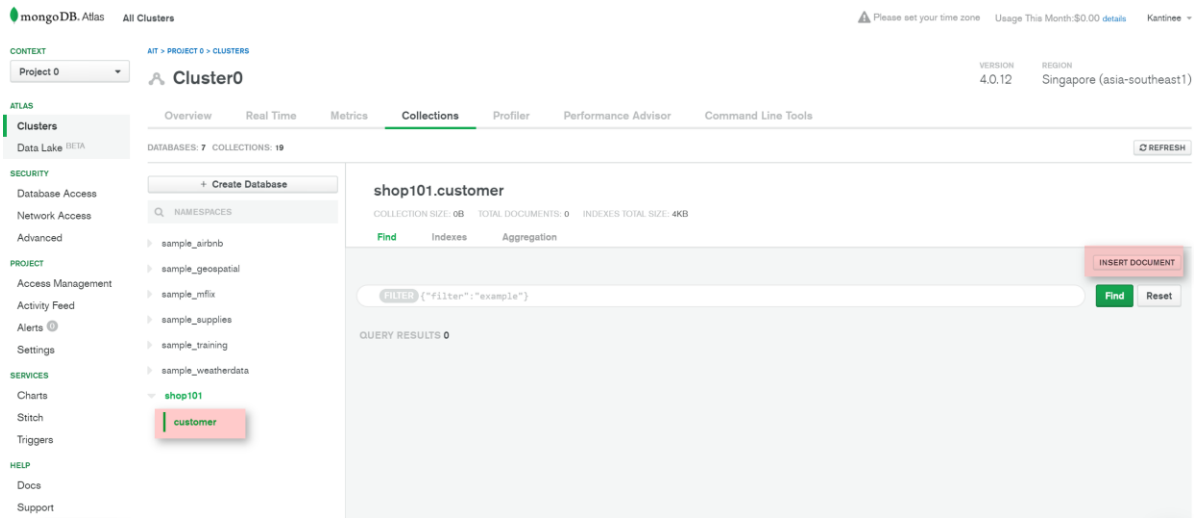
mongo "mongodb+srv://cluster0-jeneu.gcp.mongodb.net/<database name=shop101>" --username <username>

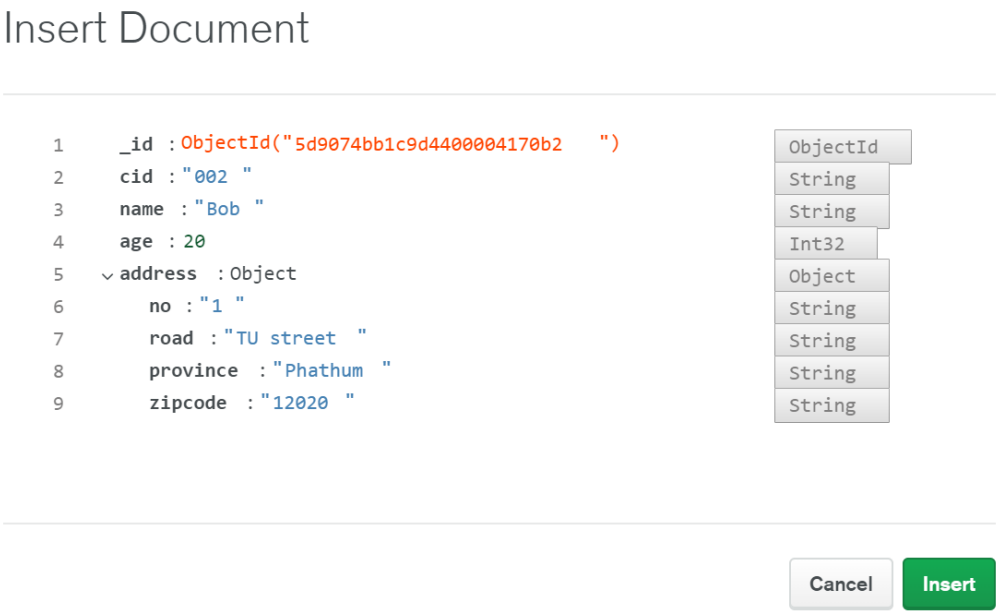
3. Insert, Update, Delete Document

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”





3.2 Insert inventory using Mongo Shell

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

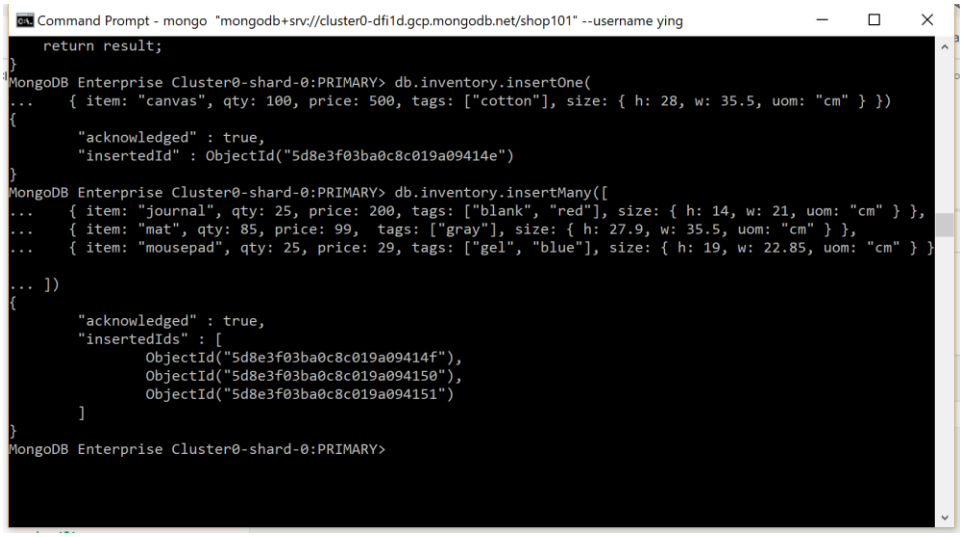
Insert Command

db.collection.insertMany()

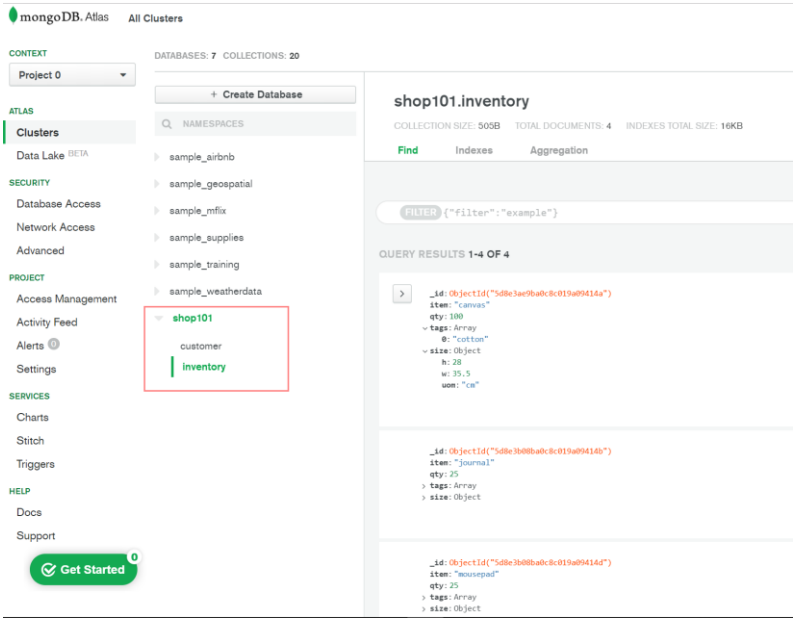
Syntax:

```
db.collection.insertMany(
    [ <document 1> , <document 2>, ... ]
)
```

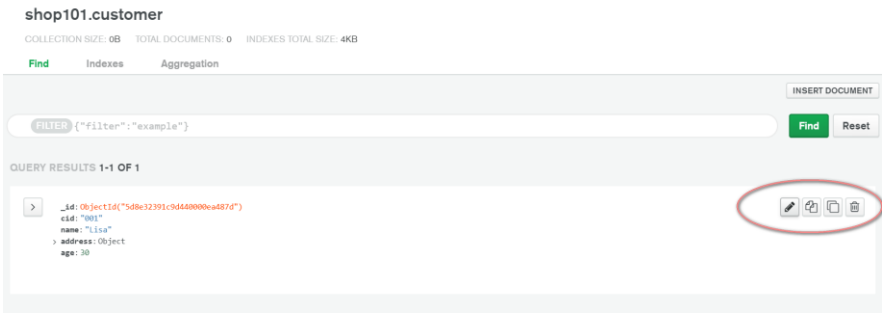
db.inventory.insertOne(
{ item: "canvas", qty: 100, price: 500, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } })
db.inventory.insertMany([
{ item: "pen", qty: 25, price: 200, tags: ["red"], size: { h: 14, w: 1, uom: "cm" } },
{ item: "mat", qty: 85, price: 99, tags: ["gray"], size: { h: 27.9, 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 })

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.find().pretty()
{
        "_id" : ObjectId("5d8f62648fb57f941738209b"),
        "item" : [
                {
                        "pid" : "canvas",
                        "qty" : 1,
                        "price" : 500
                },
                {
                        "pid" : "pen",
                        "qty" : 1,
                        "price" : 200
                }
        ],
        "customer" : "Lisa",
        "total" : 700,
        "couponused" : true
}
{
        "_id" : ObjectId("5d8f63588fb57f941738209c"),
        "item" : [
                {
                        "pid" : "mousepad",
                        "qty" : 1,
                        "price" : 29
                }
        ],
        "customer" : "Bob",
        "total" : 29,
        "couponused" : true
}
{
        "_id" : ObjectId("5d9027dd8fb57f941738209d"),
        "item" : [
                {
                        "pid" : "mat",
                        "qty" : 1,
                        "price" : 99
                }
        ],
        "customer" : "Bob",
        "total" : 99,
        "couponused" : false
}
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

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”**}})

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.updateMany( { }, {$set: {status: "A"}})
{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.updateMany( {name:"Lisa" }, {$set: {status: "I"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

4. Query Data

db.collection.find()

`db.collection.find(query, projection)`

- **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.

https://docs.mongodb.com/manual/reference/method/db.collection.find/#db.collection.find

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

4.1 Select

SQL SELECT Statements	MongoDB find() Statements
`SELECT *` `FROM Customer`	db.customer.find().pretty()

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.find().pretty()
{
        "_id" : ObjectId("5d8e32391c9d440000ea487d"),
        "cid" : "001",
        "name" : "Lisa",
        "address" : {
                "no" : "99",
                "road" : "main street",
                "province" : "BKK"
        },
        "age" : 30,
        "status" : "I"
}
{
        "_id" : ObjectId("5d8e34411c9d440000ea487f"),
        "cid" : "002",
        "name" : "Bob",
        "address" : {
                "no" : "1",
                "road" : "TU street",
                "province" : "Phathum",
                "zipcode" : "12020"
        },
        "age" : 20,
        "status" : "A"
}
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

- Projection (return some fields)

```
SELECT cid,name
FROM Customer
```

```
db.Customer.find(
    { },
    {cid:1, name:1,_id:0}
)
```

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.find({},{cid:1,name:1,_id:0})
{ "cid" : "001", "name" : "Lisa" }
{ "cid" : "002", "name" : "Bob" }
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

4.2 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})

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.find().sort({name:1})
{ "_id" : ObjectId("5d8e34411c9d440000ea487f"), "cid" : "002", "name" : "Bob", "address" : { "no" : "1"
, "road" : "TU street", "province" : "Phathum", "zipcode" : "12020" }, "age" : 20, "status" : "A" }
{ "_id" : ObjectId("5d8e32391c9d440000ea487d"), "cid" : "001", "name" : "Lisa", "address" : { "no" : "9
9", "road" : "main street", "province" : "BKK" }, "age" : 30, "status" : "I" }
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.find().sort({name:-1})
{ "_id" : ObjectId("5d8e32391c9d440000ea487d"), "cid" : "001", "name" : "Lisa", "address" : { "no" : "9
9", "road" : "main street", "province" : "BKK" }, "age" : 30, "status" : "I" }
{ "_id" : ObjectId("5d8e34411c9d440000ea487f"), "cid" : "002", "name" : "Bob", "address" : { "no" : "1"
, "road" : "TU street", "province" : "Phathum", "zipcode" : "12020" }, "age" : 20, "status" : "A" }
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

4.3 Where condition

4.3.1 =, !=, >,<, <=,>=

```
SELECT *
FROM Customer
WHERE Name ="Lisa"
```

```
db.Customer.find(
      {name ="Lisa"}
)
```

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.find({name:"Lisa"}).pretty()
{
        "_id" : ObjectId("5d8e32391c9d440000ea487d"),
        "cid" : "001",
        "name" : "Lisa",
        "address" : {
                "no" : "99",
                "road" : "main street",
                "province" : "BKK"
        },
        "age" : 30
}
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

```
SELECT *
FROM Customer
WHERE name != "Lisa"
```

```
db.Customer.find(
    { name : { $ne:"Lisa" } }
)
```

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.find({name:{$ne:"Lisa"}}).pretty()
{
        "_id" : ObjectId("5d8e34411c9d440000ea487f"),
        "cid" : "002",
        "name" : "Bob",
        "address" : {
                "no" : "1",
                "road" : "TU street",
                "province" : "Phathum",
                "zipcode" : "12020"
        },
        "age" : 20
}
```

```
SELECT *
FROM Customer
WHERE age > 25
```

```
db.Customer.find(
    { age: { $gt: 25 } }
)
```

```
SELECT *
FROM Customer
WHERE age < 25
```

```
db.Customer.find(
   { age: { $lt: 25 } }
)
```

Query Operators

- $eq: Matches values that are equal to a specified value.
- $gt: Matches values that are greater than a specified value.
- $gte: Matches values that are greater than or equal to a specified value.
- $in: Matches any of the values specified in an array.
- $lt: Matches values that are less than a specified value.
- $lte: Matches values that are less than or equal to a specified value.
- $ne: Matches all values that are not equal to a specified value.
- $nin: Matches none of the values specified in an array.

4.3.2 AND

```
SELECT *
FROM Customer
WHERE age > 25
AND   age <= 50
```

```
db.Customer.find(
   { age: { $gt: 25, $lte: 50 } }
)
```

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

4.3.3 OR

```
SELECT *
FROM Customer
WHERE status = "A"
OR age = 20
```

```
db.Customer.find(
    { $or: [ { status: "A" } , { age: 20 } ] }
)
```

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.find({$or:[{status:"A"},{age: 20}]}).pretty()
{
        "_id" : ObjectId("5d8e34411c9d440000ea487f"),
        "cid" : "002",
        "name" : "Bob",
        "address" : {
                "no" : "1",
                "road" : "TU street",
                "province" : "Phathum",
                "zipcode" : "12020"
        },
        "age" : 20,
        "status" : "A"
}
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

4.3.4 LIKE

```
SELECT *
FROM   Customer
WHERE  name   like "%is%"
```

```
db. Customer.find( { name   : /is/   } )

-or-

db.Customer .find( { name   : { $regex:  /is/   } } )
```

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.find({name :/is/}).pretty()
{
        "_id" : ObjectId("5d8e32391c9d440000ea487d"),
        "cid" : "001",
        "name" : "Lisa",
        "address" : {
                "no" : "99",
                "road" : "main street",
                "province" : "BKK"
        },
        "age" : 30,
        "status" : "I"
}
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

4.3.5 IN

```
db.Customer .find( { status: { $in: [ "A", "D" ] } } )
```

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.customer.find({status:{$in: ["A","D"]}}).pretty()
{
        "_id" : ObjectId("5d8e34411c9d440000ea487f"),
        "cid" : "002",
        "name" : "Bob",
        "address" : {
                "no" : "1",
                "road" : "TU street",
                "province" : "Phathum",
                "zipcode" : "12020"
        },
        "age" : 20,
        "status" : "A"
}
```

4.3.6 Exists

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

```
db.customer.find( {
  address.zipcode: { $exists: false }
} )
```

4.4 Query on Nested field

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

- Select all documents where uom of size is in “in” unit

```
db.inventory.find( { "size.uom": "in" } )
```

- selects items where their height less than 15 CM.

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.inventory.find( { "size.h": { $lt: 15 }, "size.uom": "cm" } ).pretty()
{
        "_id" : ObjectId("5d8e3f03ba0c8c019a09414f"),
        "item" : "pen",
        "qty" : 25,
        "price" : 200,
        "tags" : [
                "red"
        ],
        "size" : {
                "h" : 14,
                "w" : 1,
                "uom" : "cm"
        }
}
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

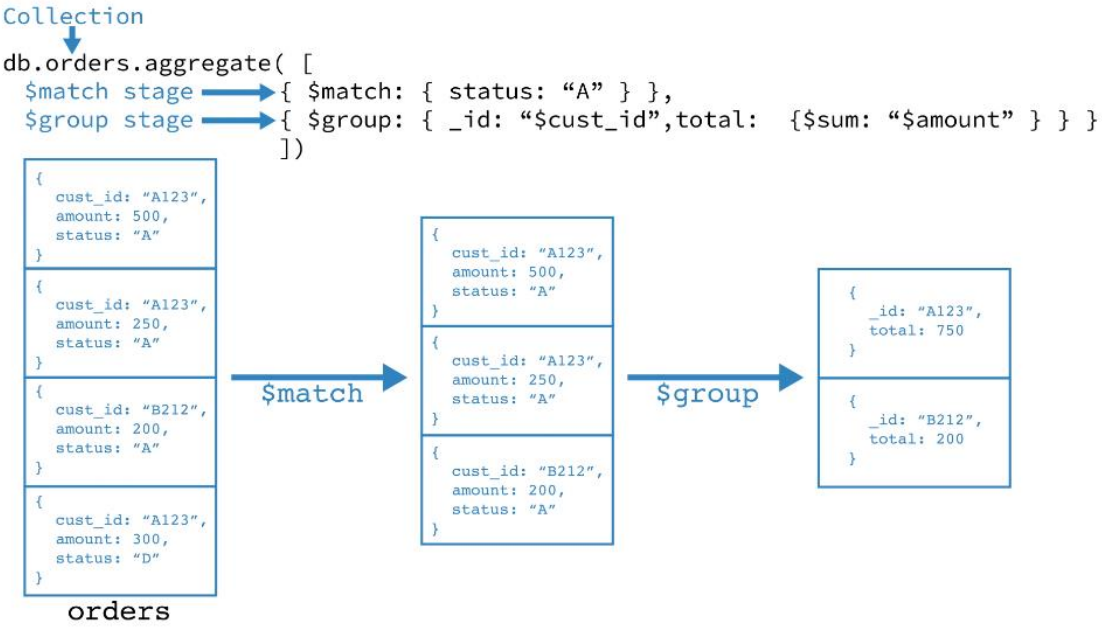
.AGGREGATE

4.5 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.

SQL Terms, Functions, and Concepts	MongoDB Aggregation Operators
WHERE	`$match`
GROUP BY	`$group`
HAVING	`$match`
SELECT	`$project`
ORDER BY	`$sort`
LIMIT	`$limit`
SUM()	`$sum`
COUNT()	`$sum` `$sortByCount`
join	`$lookup`

Example



4.5.1 Sum

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

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([    { $match: { couponused:true } },
  { $group: { _id: "$customer", total: { $sum: "$total" } } } ])
{ "_id" : "Bob", "total" : 29 }
{ "_id" : "Lisa", "total" : 700 }
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

All sales record:

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.find().pretty()
{
        "_id" : ObjectId("5d8f62648fb57f941738209b"),
        "item" : [
                {
                        "pid" : "canvas",
                        "qty" : 1,
                        "price" : 500
                },
                {
                        "pid" : "pen",
                        "qty" : 1,
                        "price" : 200
                }
        ],
        "customer" : "Lisa",
        "total" : 700,
        "couponused" : true
}
{
        "_id" : ObjectId("5d8f63588fb57f941738209c"),
        "item" : [
                {
                        "pid" : "mousepad",
                        "qty" : 1,
                        "price" : 29
                }
        ],
        "customer" : "Bob",
        "total" : 29,
        "couponused" : true
}
{
        "_id" : ObjectId("5d9027dd8fb57f941738209d"),
        "item" : [
                {
                        "pid" : "mat",
                        "qty" : 1,
                        "price" : 99
                }
        ],
        "customer" : "Bob",
        "total" : 99,
        "couponused" : false
}
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

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

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([{ $group: { _id: "$customer", total: { $sum: "$total" } } } ])
{ "_id" : "Bob", "total" : 128 }
{ "_id" : "Lisa", "total" : 700 }
```

4.5.2 Count

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([{ $group: { _id: "$customer", total: { $sum: 1 } } } ])
{ "_id" : "Bob", "total" : 2 }
{ "_id" : "Lisa", "total" : 1 }
```

Note .FIND() also has COUNT() function

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.find({customer:"Lisa"}).count()
1
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.find({customer:"Bob"}).count()
2
```

4.5.3 Average, Min, Max

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.find().pretty()
{
        "_id" : ObjectId("5d8f62648fb57f941738209b"),
        "item" : [
                {
                        "pid" : "canvas",
                        "qty" : 1,
                        "price" : 500
                },
                {
                        "pid" : "pen",
                        "qty" : 1,
                        "price" : 200
                }
        ],
        "customer" : "Lisa",
        "total" : 700,
        "couponused" : true
}
{
        "_id" : ObjectId("5d8f63588fb57f941738209c"),
        "item" : [
                {
                        "pid" : "mousepad",
                        "qty" : 1,
                        "price" : 29
                }
        ],
        "customer" : "Bob",
        "total" : 29,
        "couponused" : true
}
{
        "_id" : ObjectId("5d9027dd8fb57f941738209d"),
        "item" : [
                {
                        "pid" : "mat",
                        "qty" : 1,
                        "price" : 99
                }
        ],
        "customer" : "Bob",
        "total" : 99,
        "couponused" : false
}
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

```
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" } } }
                    ])
```

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([    { $group: { _id: "$customer", average: { $avg: "$total" } } } ])
{ "_id" : "Bob", "average" : 64 }
{ "_id" : "Lisa", "average" : 700 }
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([    { $group: { _id: "$customer", minimum: { $min: "$total" } } } ])
{ "_id" : "Bob", "minimum" : 29 }
{ "_id" : "Lisa", "minimum" : 700 }
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([    { $group: { _id: "$customer", maximum: { $max: "$total" } } } ])
{ "_id" : "Bob", "maximum" : 99 }
{ "_id" : "Lisa", "maximum" : 700 }
```

4.5.4 Having

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

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate
([{ $group: { _id: "$customer", total: { $sum: "$total" } } } ,
 {$match: {total: {$gt: 150 }}}])
{ "_id" : "Lisa", "total" : 700 }
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

4.5.5 Sort and Limit

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

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([{$group:{_id:"$customer",total: {$sum:"$total"}}}, { $sort: { total: -1 }}])
{ "_id" : "Lisa", "total" : 700 }
{ "_id" : "Bob", "total" : 128 }
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([{$group:{_id:"$customer",total: {$sum:"$total"}}}, { $sort: { total: 1 }}])
{ "_id" : "Bob", "total" : 128 }
{ "_id" : "Lisa", "total" : 700 }
```

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

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([{$group:{_id:"$customer",total: {$sum:"$total"}}}, { $sort: { total: -1 }},{$limit:1}])
{ "_id" : "Lisa", "total" : 700 }
MongoDB Enterprise Cluster0-shard-0:PRIMARY>
```

4.6 Join

- Syntax

```
{
   $lookup:
     {
       from: <collection to join>,
       localField: <field from the input documents>,
       foreignField: <field from the documents of the "from" collection>,
       as: <output array field>
     }
}
```

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

```
db.orders.insert([
   { "_id" : 1, "item" : "almonds", "price" : 12, "quantity" : 2 },
   { "_id" : 2, "item" : "pecans", "price" : 20, "quantity" : 1 },
   { "_id" : 3  }
])
```

Orders

Inventory

```
db.inventory.insert([
   { "_id" : 1, "sku" : "almonds", description: "product 1", "instock" : 120 },
   { "_id" : 2, "sku" : "bread", description: "product 2", "instock" : 80 },
   { "_id" : 3, "sku" : "cashews", description: "product 3", "instock" : 60 },
   { "_id" : 4, "sku" : "pecans", description: "product 4", "instock" : 70 },
   { "_id" : 5, "sku": null, description: "Incomplete" },
   { "_id" : 6 }
])
```

Command:

Join orders กับ Inventory

```
db.orders.aggregate([
   {
     $lookup:
       {
         from: "inventory",
         localField: "item",
         foreignField: "sku",
         as: "inventory_docs"
       }
  }
])
```

- Insert Inventory info into Orders
- Orders.item = Inventory.sku
- New Field = Inventory_docs

Result:

The operation returns the following documents:

```
{
   "_id" : 1,
   "item" : "almonds",
   "price" : 12,
   "quantity" : 2,
   "inventory_docs" : [
      { "_id" : 1, "sku" : "almonds", "description" : "product 1", "instock" : 120 }
   ]
}
{
```

Try our database:

- Include customer info into sale

```
db.sales.aggregate([
  {
    $lookup:
     {
       from: "customer",
       localField: "customer",
       foreignField: "name",
       as: "cust_info"
     }
  }
]).pretty()
```

```
MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.sales.aggregate([
...     {
...       $lookup:
...         {
...           from: "customer",
...           localField: "customer",
...           foreignField: "name",
...           as: "cust_info"
...         }
...    }
... ]).pretty()
{
        "_id" : ObjectId("5d8f62648fb57f941738209b"),
        "item" : [
                {
                        "pid" : "canvas",
                        "qty" : 1,
                        "price" : 500
                },
                {
                        "pid" : "pen",
                        "qty" : 1,
                        "price" : 200
                }
        ],
        "customer" : "Lisa",
        "total" : 700,
        "couponused" : true,
        "cust_info" : [
                {
                        "_id" : ObjectId("5d8e32391c9d440000ea487d"),
                        "cid" : "001",
                        "name" : "Lisa",
                        "address" : {
                                "no" : "99",
                                "road" : "main street",
                                "province" : "BKK"
                        },
                        "age" : 30,
                        "status" : "I"
                }
        ]
}
```

- 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

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

	A	B	C	D	E	F	G	H
1				Peer Feedback Form				
2		Reviewer Name	xx	xx	xx			
3		Meeting Room#	xx					
4								
5								
6	Team#	Project Title	Member#	Presenter Name	Time Taken (min)	Summary of Presentation	Strengths	Suggestions
7	19	Bike sharing	19-1	Mr. XYZ	15	good presentation	precise and easy to understand	practice presentation skills
8	20							
9								
10								
11								
12								
13								

```
{
"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.

3. Insert the peer review into the feedback table in the database.
4. 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