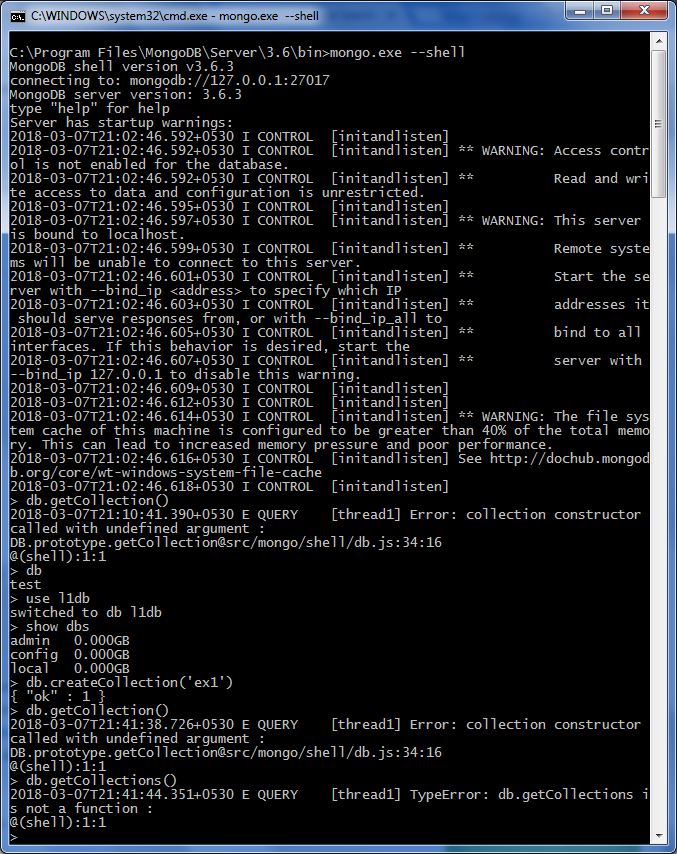
Hands on assignments:

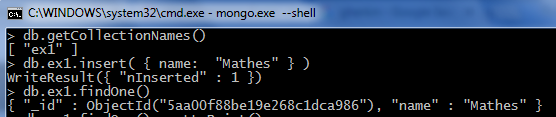
* 1. Create a Collection and insert documents with varying number of keys
  2. Insert documents with a specific value for the primary key (\_id)
  3. Use find() to display all the documents
  4. Use find()to select few documents from the collection, based on a **condition**
  5. Use ‘Java Script’ loop and insert multiple documents with ‘random’ numbers
     1. Explore on ‘sort()’, ‘skip()’
  6. Create a collection with ‘Nested’ JSON Documents
     1. E.g. ‘Address’ for every ‘Customer’
  7. Create a Collection with documents using Array of Embedded documents
     1. E.g. array of ‘course’ documents with ‘courses’ key for each Student
  8. Use ‘update()’
     1. To change the value of a field
        + $set -> change the field value
        + $unset ->change /add /remove the field (e.g. adding a **discount** field only for few customers)
        + $inc -> increment (only for numeric fields)
     2. To replace the document with another document (changing the schema)
     3. To update multiple records, override the default behavior, by using the switches:
        + {multi:1} 🡪 Multiple ‘true’
        + {upsert:1} 🡪 update + insert
  9. Update on Arrays / Collections
     1. Create a Collection with one of the field as an array and explore on
* $ push
* $pushAll
* $pull
* $pullAll
  1. Remove
     1. Remove few selected documents from a Collection
     2. Remove only one row with ‘justOne’ switch
  2. Indexes:
     1. Create an user defined Index on a Collection, based on a field in ‘descending’ order
     2. Practice ‘unique’ & ‘dropDups’ switches with ‘ensureIndex()’
  3. Aggregation
     1. Insert multiple ‘order’ documents into a Collection
        + With fields – custID,ord\_Date, amount&status
     2. Use aggregate() for displaying the total amount of orders with status as ‘processed’
     3. Use aggregate() for displaying the number of orders with status as ‘cancelled’
  4. Replication
     1. Create a Replica Set with 3 replicas
     2. Write data using ‘primary’
     3. Read data from ‘secondary’
     4. Check the FailOver
        + Kill the existing ‘Primary’ instance
        + Verify the new ‘Primary’ instance
  5. Sharding
     1. Configure a ‘config’ server
     2. Use ‘mongos’ service
     3. Enable Sharding at DB level and also at Collection Level
     4. Configure Index on a field and make it as the shard key
     5. Insert thousands of documents into the Collection
     6. Check if the data spill over to other Shards
  6. Working with MogoDB from Java application
     1. Download Mongo Driver for Java (mongo-java-driver-2.12.3.jar)
     2. Add it to the ‘Java Project’ libraries
     3. Create different methods in a Java Class for
        + Creating a Collection in MongoDB
          1. E.g. ‘Orders’
        + Inserting document into a Collection
          1. Takes ‘order’(BasicDBObject) document and Collection name (DBCollection) as arguments
        + Updating a Document
        + Deleting a Document
     4. Create a main class and invoke the methods & perform CRUD operations.
        + MongoClient & DB types

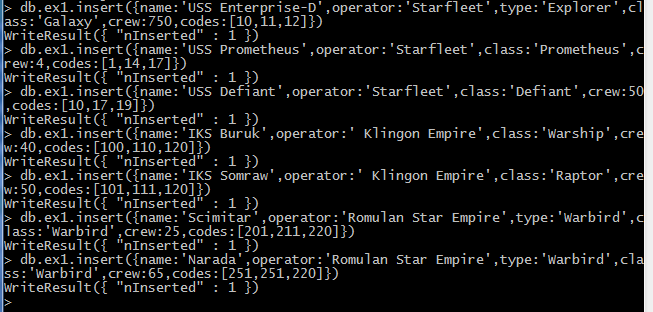
7 Mar 2018

**Create a Collection and insert documents with varying number of keys**

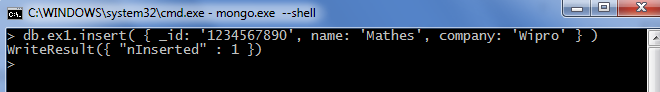
Start mongodb: mongod --dbpath "C:\Users\M24531\Desktop\Trainings\MongoDb\excerciseFiles\data\db"



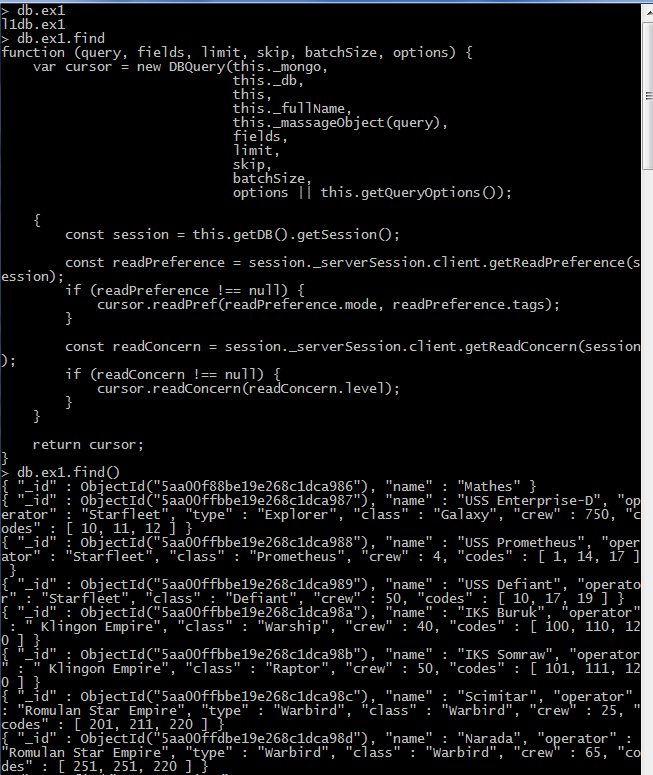




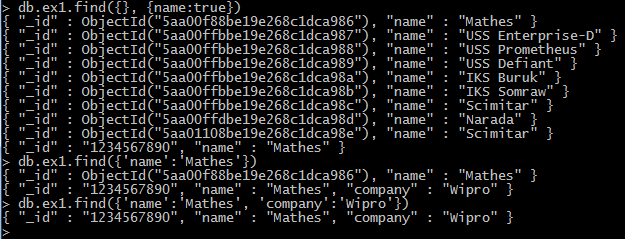
**Insert documents with a specific value for the primary key (\_id)**



**Use find() to display all the documents**

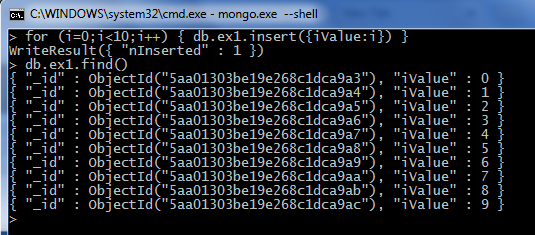
**Use find()to select few documents from the collection, based on a condition**



12 March 2018

**Use ‘Java Script’ loop and insert multiple documents with ‘random’ numbers**

Insert numbers using loop



Insert random numbers

db.ex1.insert({rand: \_rand()})

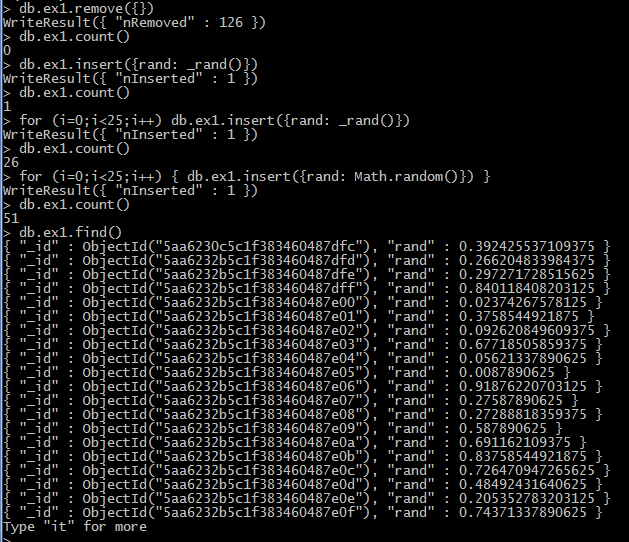
for (i=0;i<25;i++) db.ex1.insert({rand: \_rand()})

db.ex1.count()

for (i=0;i<25;i++) { db.ex1.insert({rand: Math.random()}) }

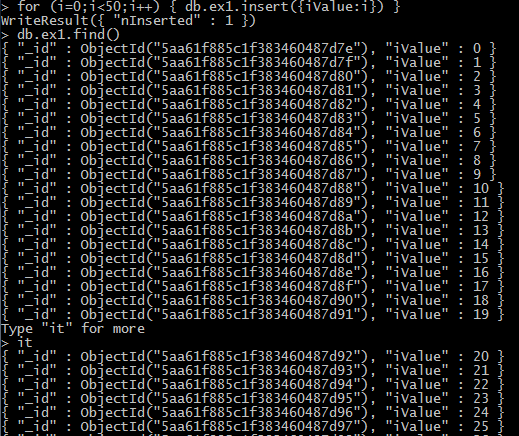
db.ex1.count()

db.ex1.find()



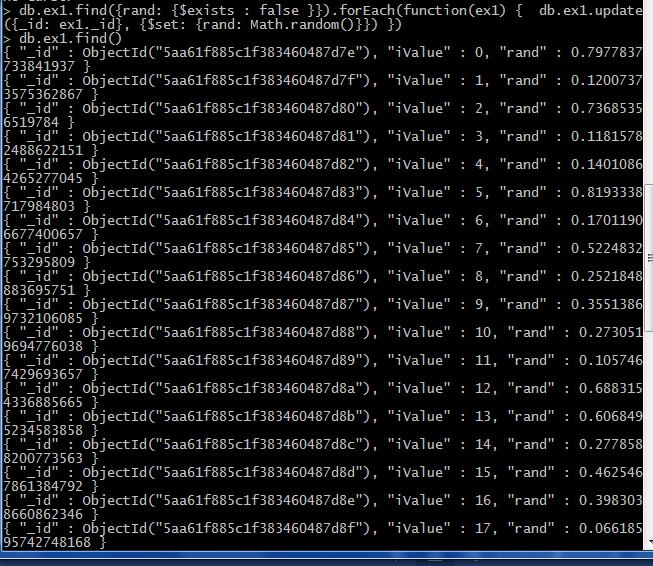
Insert some numbers

> for (i=0;i<50;i++) { db.ex1.insert({iValue:i}) }



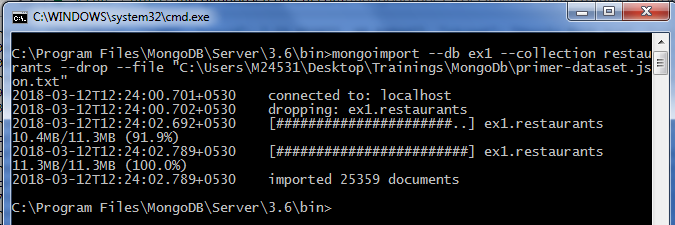
Update them with random numbers

> db.ex1.find({rand: {$exists : false }}).forEach(function(ex1) { db.ex1.update({\_id: ex1.\_id}, {$set: {rand: Math.random()}}) })

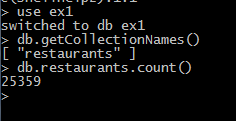


**Explore on ‘sort()’, ‘skip()’**

Importing data from - <https://docs.mongodb.com/getting-started/shell/import-data/>

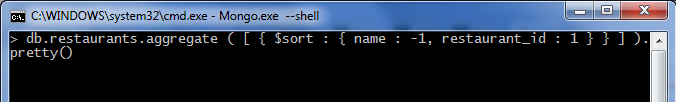


Check if collection and data are available



**Use sort function**

db.restaurants.aggregate ( [ { $sort : { name : -1, restaurant\_id : 1 } } ] ).pretty()



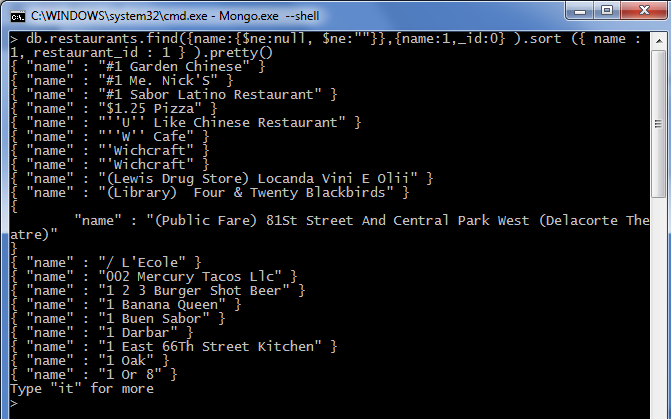


Sort and find non null name columns

db.restaurants.find({name:{$ne:null, $ne:""}}).sort ({ name : 1, restaurant\_id : 1 } ).limit(2).pretty()

Return only the names column – this way I’ll know that this data is sorted by name

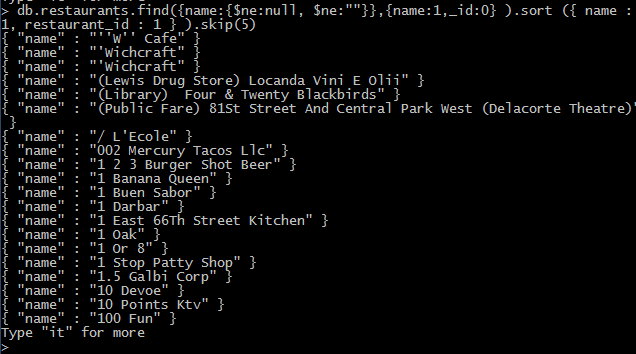
db.restaurants.find({name:{$ne:null, $ne:""}},{name:1,\_id:0} ).sort ({ name : 1, restaurant\_id : 1 } ).pretty()



**Use skip function**

Skipped the first 5 results.

db.restaurants.find({name:{$ne:null, $ne:""}},{name:1,\_id:0} ).sort ({ name : 1, restaurant\_id : 1 } ).skip(5)



**Create a collection with ‘Nested’ JSON Documents**

E.g. ‘Address’ for every ‘Customer’

db.ex1.insert (

{

cusName: "Mathes",

age:29,

companyDetails: {

name: "Wipro Technologies",

address: {

line1 : "CDC2, Wipro Street",

line2 : "Sholinganalur",

line3 : "Chennai",

pincode: 600119

}

}

}

)



13 March 2018

**Create a Collection with documents using Array of Embedded documents**

E.g. array of ‘course’ documents with ‘courses’ key for each Student

db.ex1.insert(

{

students: [{

firstName:'Matheswaran',

lastName:'Kanagarajan',

age: 29,

courses:[{

name:'Ethereum Developer',

code:'DAPP101',

duration:'6 months'

},

{

name:'Blueprism Developer',

code:'BPSM101',

duration:'3 months'

},

{

name:'Javascript Developer',

code:'JS101',

duration:'12 months'

}]

},

{

firstName:'Prasanna',

lastName:'Kalyanasundaram',

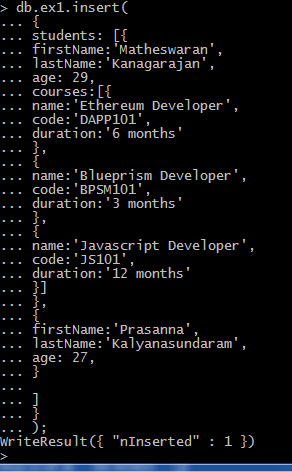
age: 27,

}

]

}

);



db.ex1.insertMany( {[{ students: [{ firstName:'Matheswaran', lastName:'Kanagarajan', age: 29, courses:[{ name:'Ethereum Developer', code:'DAPP101', duration:'6 months' }, { name:'Blueprism Developer', code:'BPSM101', duration:'3 months' }, { name:'Javascript

Developer', code:'JS101', duration:'12 months' }] }, { firstName:'Prasanna', lastName:'Kalyanasundaram', age: 27, courses:[{ name:'Ethereum Sr Developer', code:'DAPP102', duration:'6 months' }, { name:'Blueprism Architect', code:'BPSM201', duration:'3

months' }, { name:'Javascript Architect', code:'JS202', duration:'12 months' }] } ] }] } );

mk –check this – throws error

**Use ‘update()’**

**i. To change the value of a field**

db.inventory.insertMany( [

{ item: "journal", instock: [ { warehouse: "A", qty: 5 }, { warehouse: "C", qty: 15 } ] },

{ item: "notebook", instock: [ { warehouse: "C", qty: 5 } ] },

{ item: "paper", instock: [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty: 15 } ] },

{ item: "planner", instock: [ { warehouse: "A", qty: 40 }, { warehouse: "B", qty: 5 } ] },

{ item: "postcard", instock: [ { warehouse: "B", qty: 15 }, { warehouse: "C", qty: 35 } ] }

]);

$set -> change the field value

db.inventory.update(

{"\_id" : ObjectId("5aa77ae95c1f383460487e37") } ,

{$set:

{

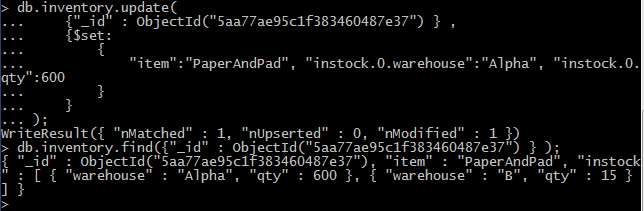
"item":"PaperAndPad", "instock.0.warehouse":"Alpha", "instock.0.qty":600

}

}

);

db.inventory.find({"\_id" : ObjectId("5aa77ae95c1f383460487e37") } );



$unset ->change /add /remove the field (e.g. adding a discount field only for few customers)

db.inventory.update(

{"\_id" : ObjectId("5aa77ae95c1f383460487e37") } ,

{$unset:

{

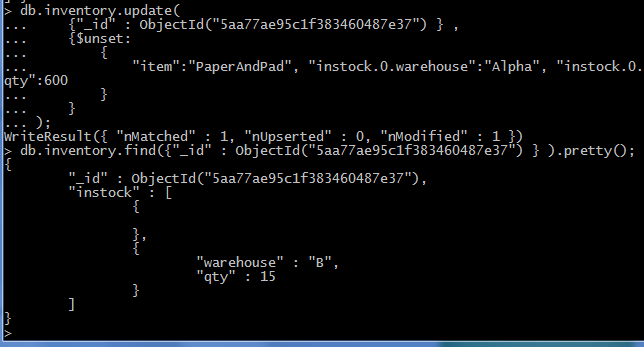
"item":"PaperAndPad", "instock.0.warehouse":"Alpha", "instock.0.qty":600

}

}

);

db.inventory.find({"\_id" : ObjectId("5aa77ae95c1f383460487e37") } ).pretty();



$inc -> increment (only for numeric fields)

db.inventory.find({"item" : "journal"}).pretty();

db.inventory.update(

{"item" : "journal"} ,

{$inc:

{

"instock.0.qty":-15,

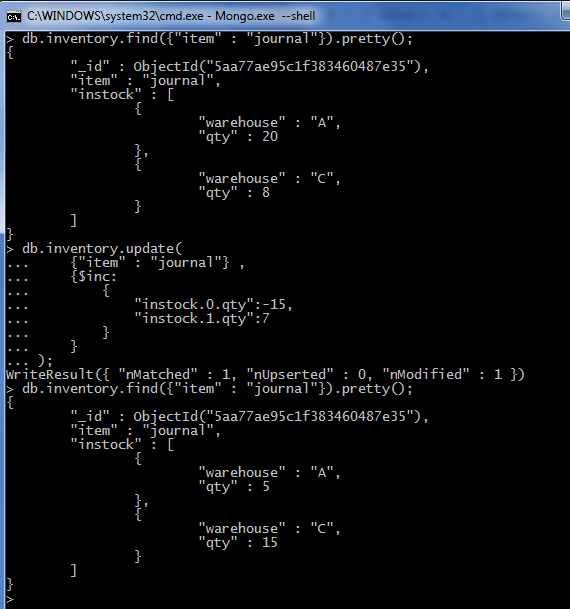
"instock.1.qty":7

}

}

);

db.inventory.find({"item" : "journal"}).pretty();



**ii. To replace the document with another document (changing the schema)**

mk – continue from here

https://docs.mongodb.com/manual/tutorial/update-documents/

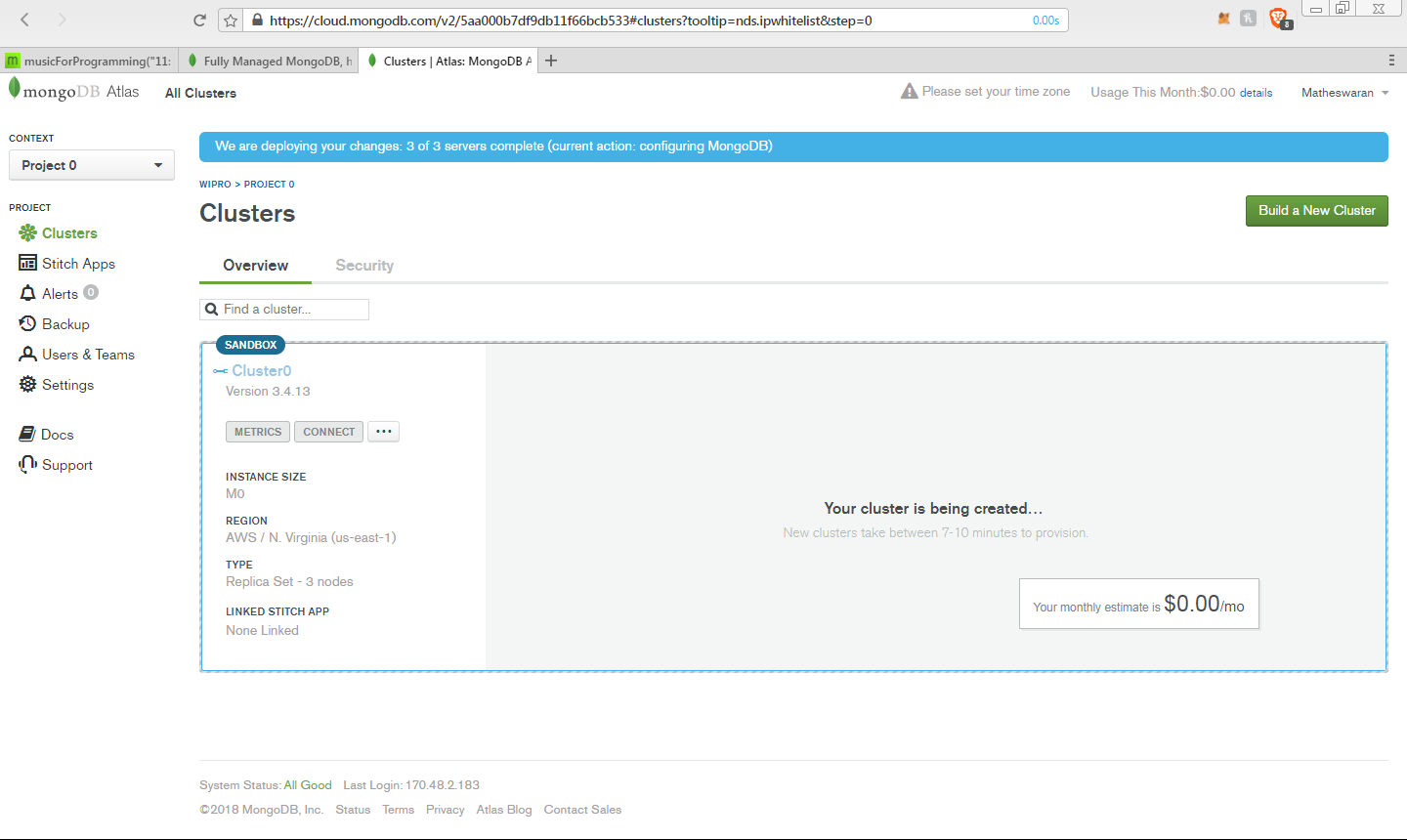
**Update on Arrays / Collections**

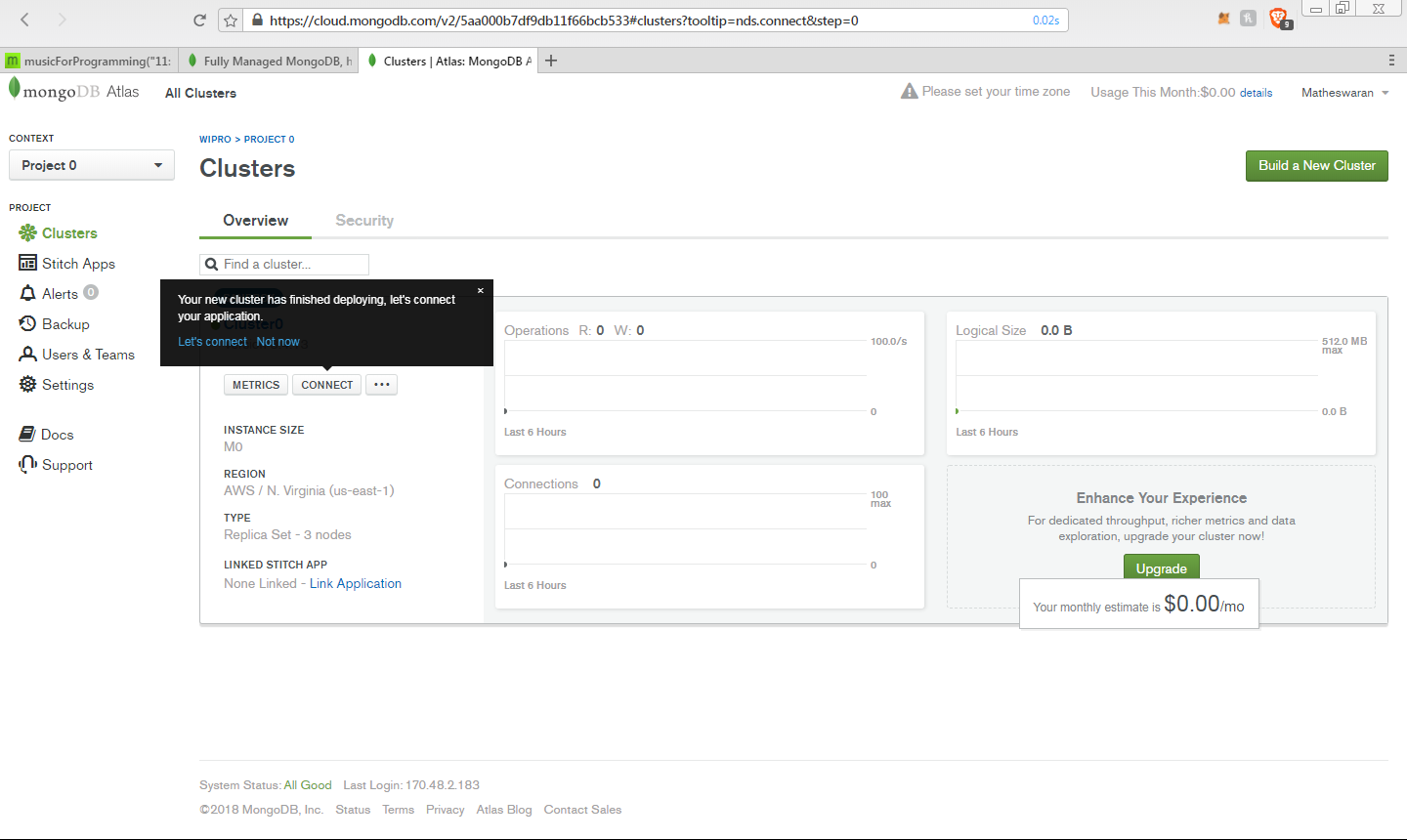
* Create a Collection with one of the field as an array and explore on
* $ push
* $pushAll
* $pull
* $pullAll

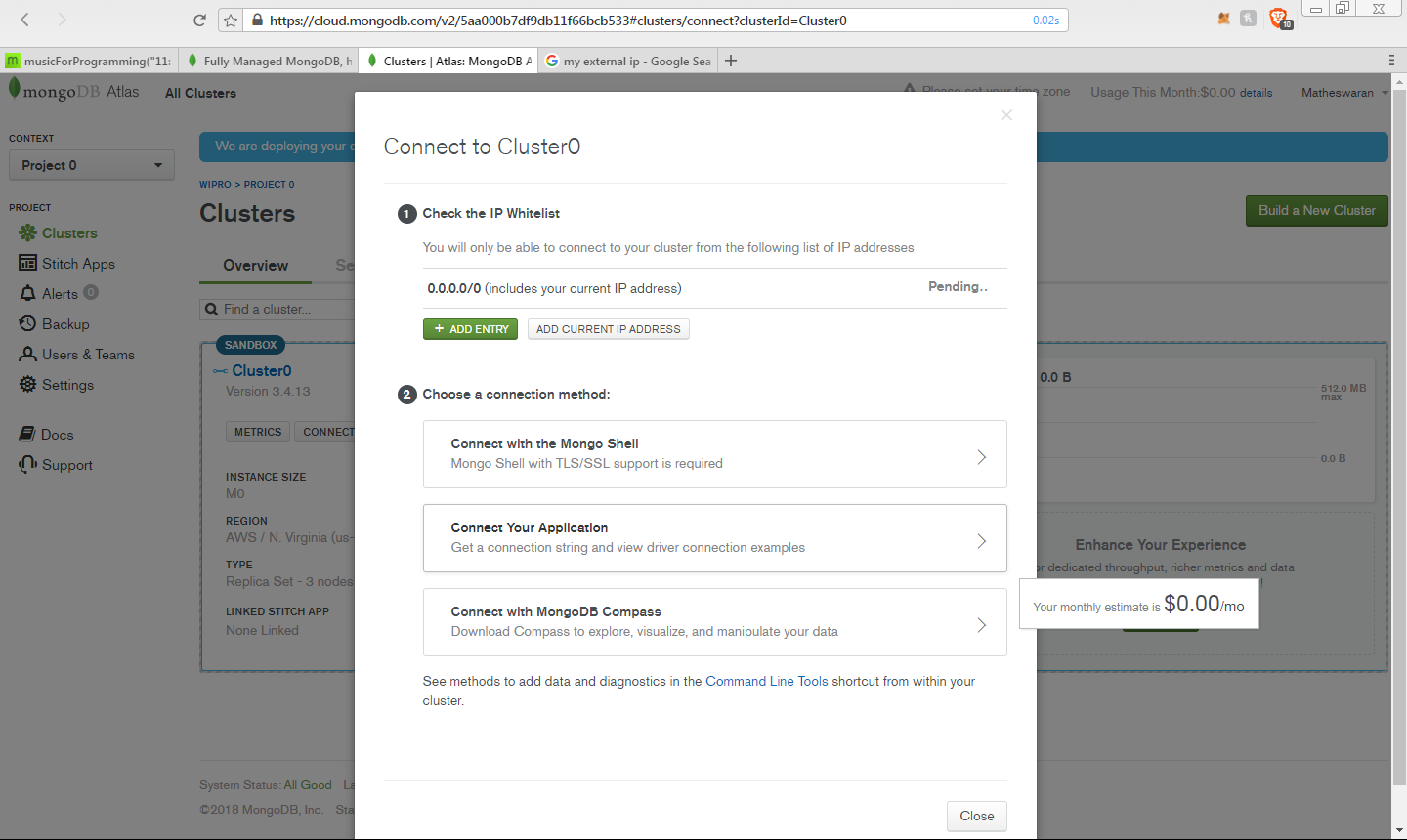
**Notes & Reference**

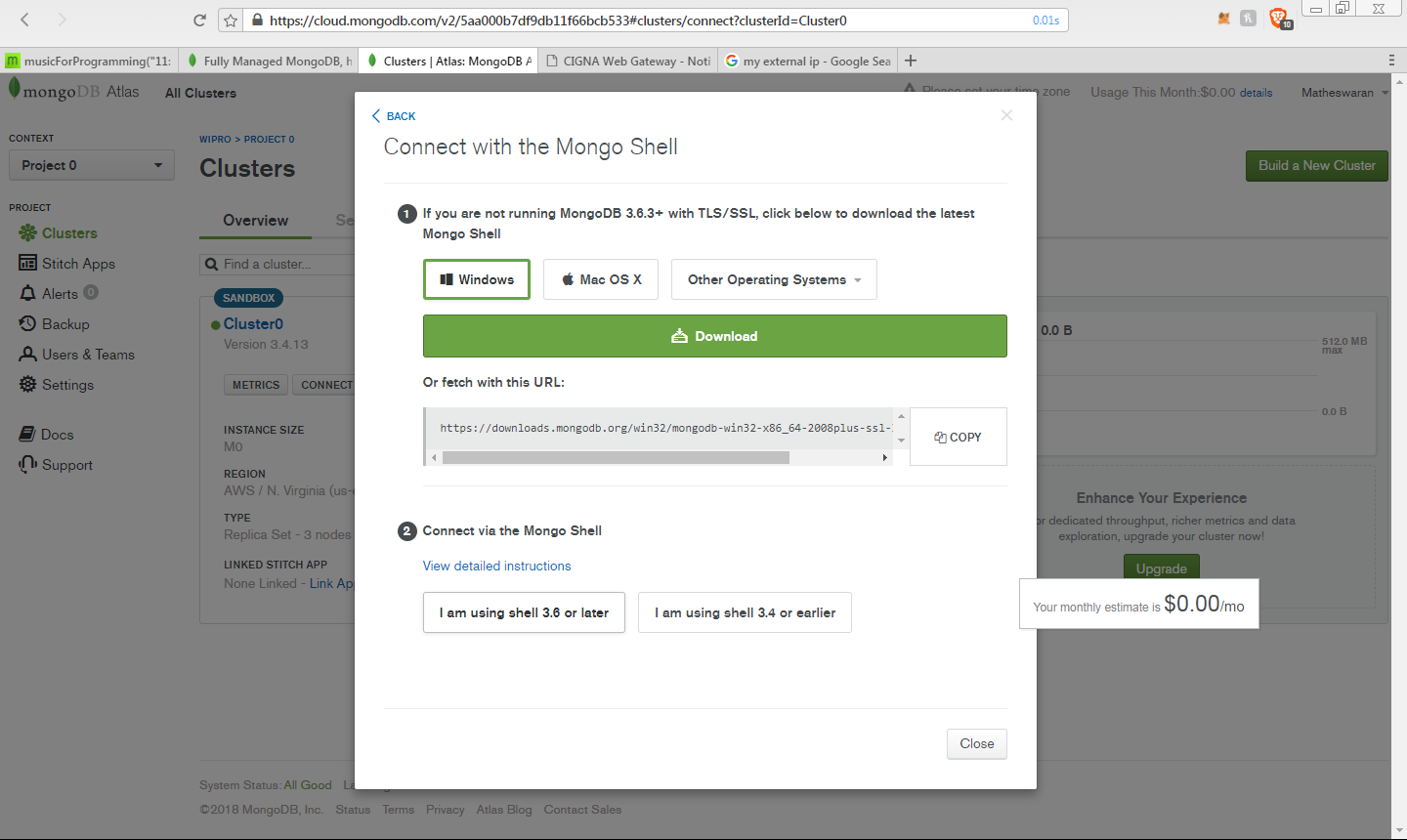
|  |  |
| --- | --- |
| **Start server** | mongod --dbpath "C:\Users\M24531\Desktop\Trainings\MongoDb\excerciseFiles\data\db" |
| **Start shell** | Mongo.exe --shell |

**Cloud setup - wip**









mongodb://mongouser:Wipro@123@cluster0-shard-00-00-shxy0.mongodb.net:27017,cluster0-shard-00-01-shxy0.mongodb.net:27017,cluster0-shard-00-02-shxy0.mongodb.net:27017/admin?replicaSet=Cluster0-shard-0&ssl=true