MongoDB

Surisetti Lochani Vilehya

Emp ID : 43317

INDEX:

INTRODUCTION

Some important terminology

1. Database : collection of data in a structured way
2. SQL : structured query language
3. RDBMS : relational database management system

What is NOSQL?

It is also called non-SQL alias NON relational

In simple terms: it is a key-value database

A NoSQL database provides:

- A mechanism for storage and retrieval of data
- The data is NOT modeled in relational or tabular format
- A large variations and flavors of NoSQL are available in market - MongoDB, amazon, documentDB, google datastore, amazon DynamoDB etc.
- Extremely useful, powerful, and high-performance database in large big data applications large distributed network architecture apps etc.

MongoDB

- It is a free and open-source cross platform document-oriented database
- It is classified as NoSQL database program
- It uses JSON like document with schemas

Document-oriented database provides API's or a query/update language that exposes the ability to query or update based on the internal structure in the document

- MongoDB documents are composed of field and value pairs and have the following structure

```
{
Field1:value1,
Field2:value2
}
```

MongoDB is NOT?

- It is NOT a RDBMS system
- It does not have any concept of joins
- It is not tough or complicated

Languages supported by MongoDB

- PHP
- Node JS
- Python
- Java
- C#
- C++

DATABASE,DOCUMENTS AND COLLECTIONS

What is a database?

- It is described as a physical container for collections
- It can have any number of collections
- Each database gets its own set of files on the file system
- A MongoDB server can host multiple databases inside it
- Single or more collections
- One or more collections together becomes a database

What is a collection?

- Collection is a group of MongoDB documents
- You can relate a collection as a "Table"
- Unlike tables, collections does not have any schema definition
- Unlike RDBMS database – collections do not have any concept of JOINS
- However, we can achieve joins functionality using aggregations in MongoDB
- A set of documents become a collection
- It can have any number of documents
- It can have a dynamic schema : same or different
- User defined schema in MongoDB and they are NOT static or fixed
- It do not enforce any schema (no joins concept)

What is a document?

- Document in MongoDB is set of "key-value" pairs
- Every document in MongoDB has a unique value via key "_id"
- It have flexible and dynamic schema
- Its schema is user defined and is not fixed or static
- It can hold any data as along as they are valid data types in MongoDB
- Documents within collection can have different schema or fields
- Documents within a collection are related data belonging to a particular subject

- It can have any type of database – if it is a valid MongoDB data type
- A schema can be different for different documents
- MongoDB will add a key automatically for each document called **"_id"**
- MongoDB valid document example

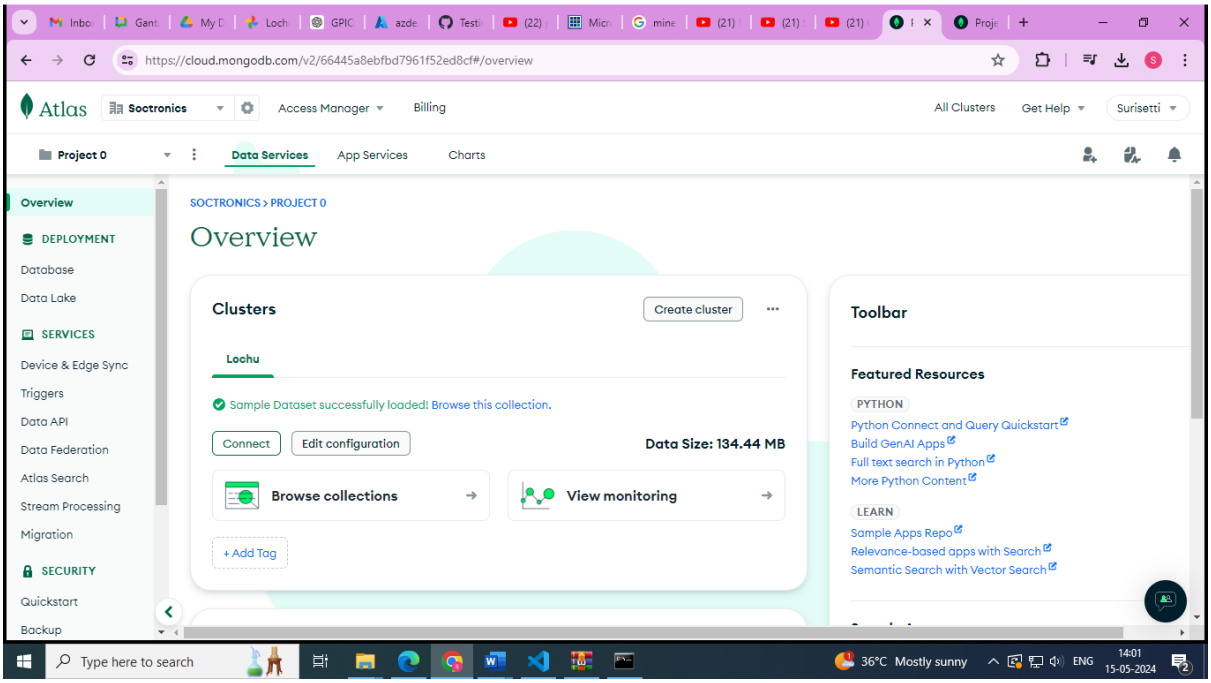
```
{
        "firstname" : "Sridhar" ,
        "lastname": "ram" ,
        "email" : "email@test.com"
},
{
        "firstname" : "Sridhar" ,
        "lastname": "ram" ,
        "email" : "email@test.com" ,
        "address" : "Hyderabad"
}
```

RDMS – relational database management systems

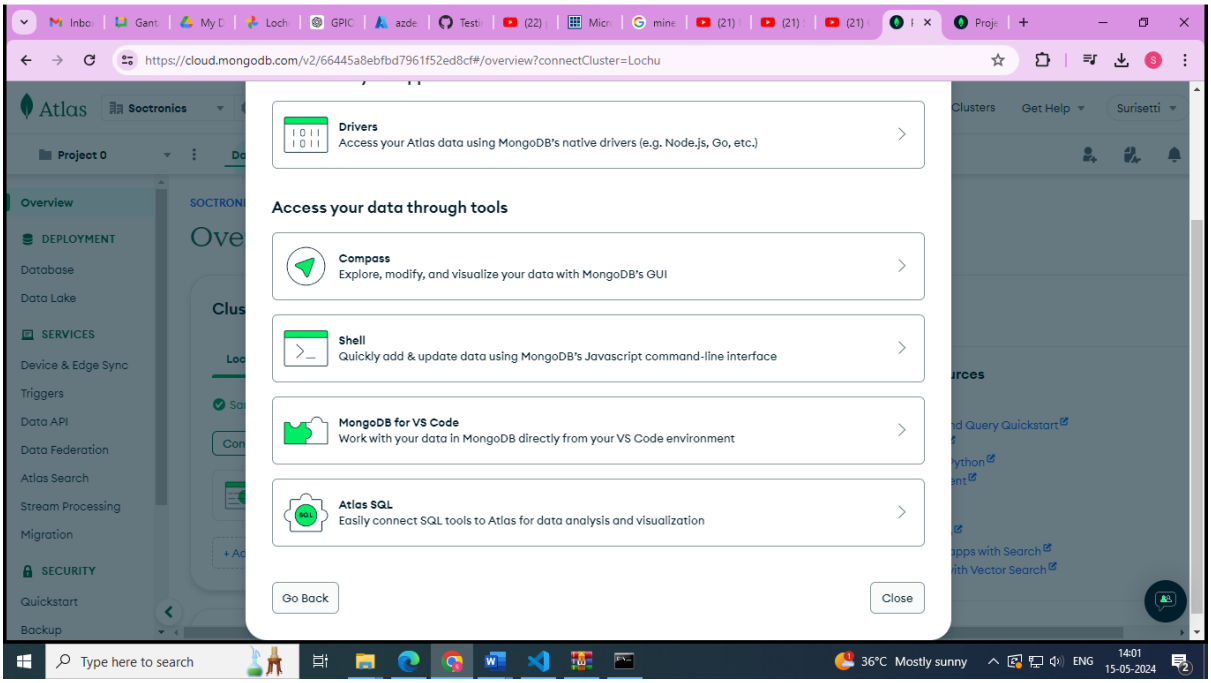
- Tables, columns, and rows
- Table – stands for entity
  - Person
  - Users
  - Courses
- Columns – fields inside the table e.g. person
  - First name
  - Last name
  - Email
  - Mobile
  - Address
- Rows – actual data

CONNECT TO MongoDB

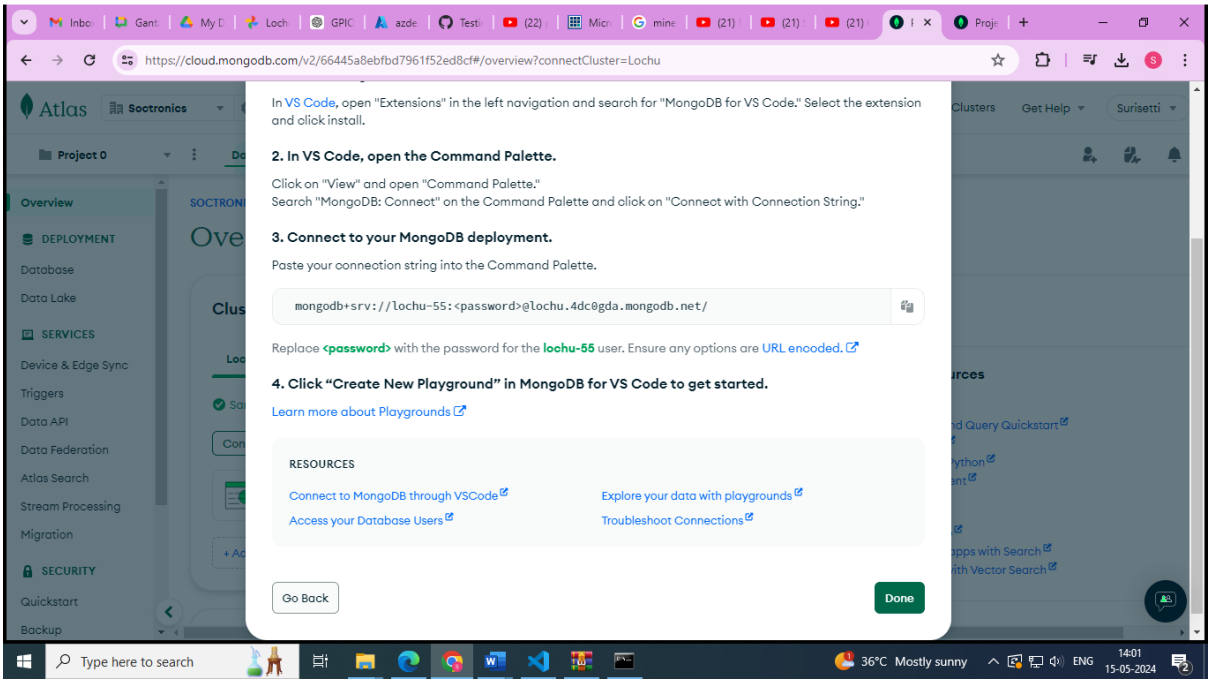
- Create an account in the mongodb atlas: https://www.mongodb.com/try
- Create one cluster in it

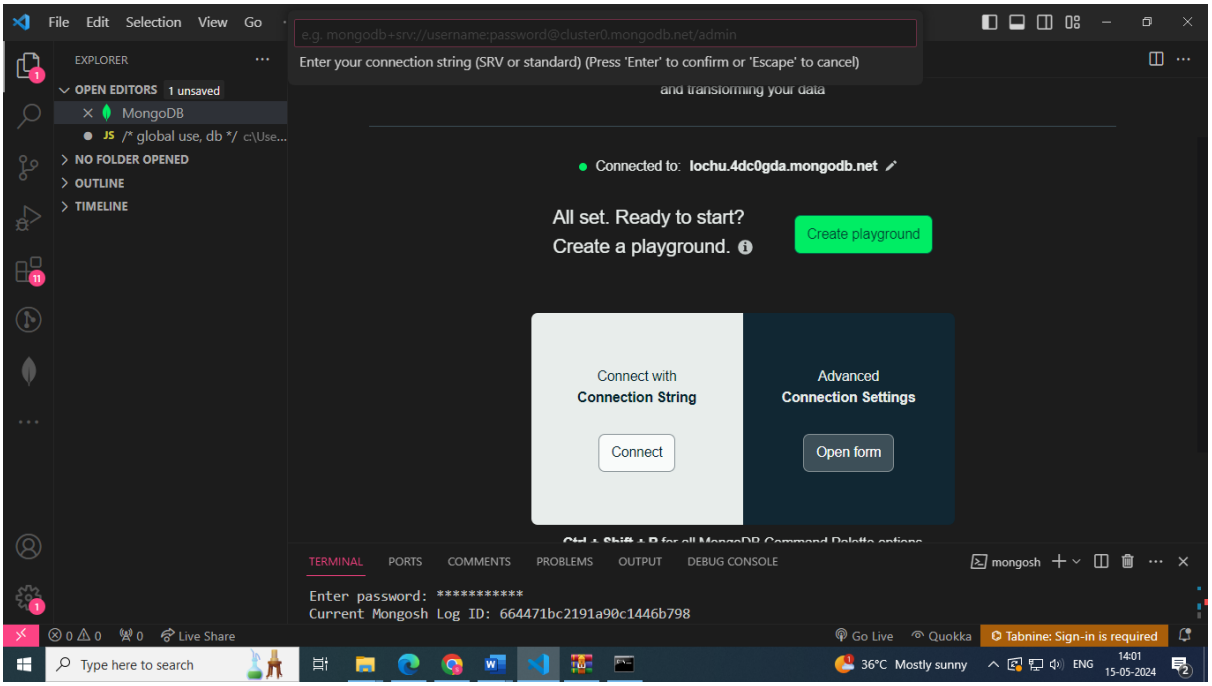


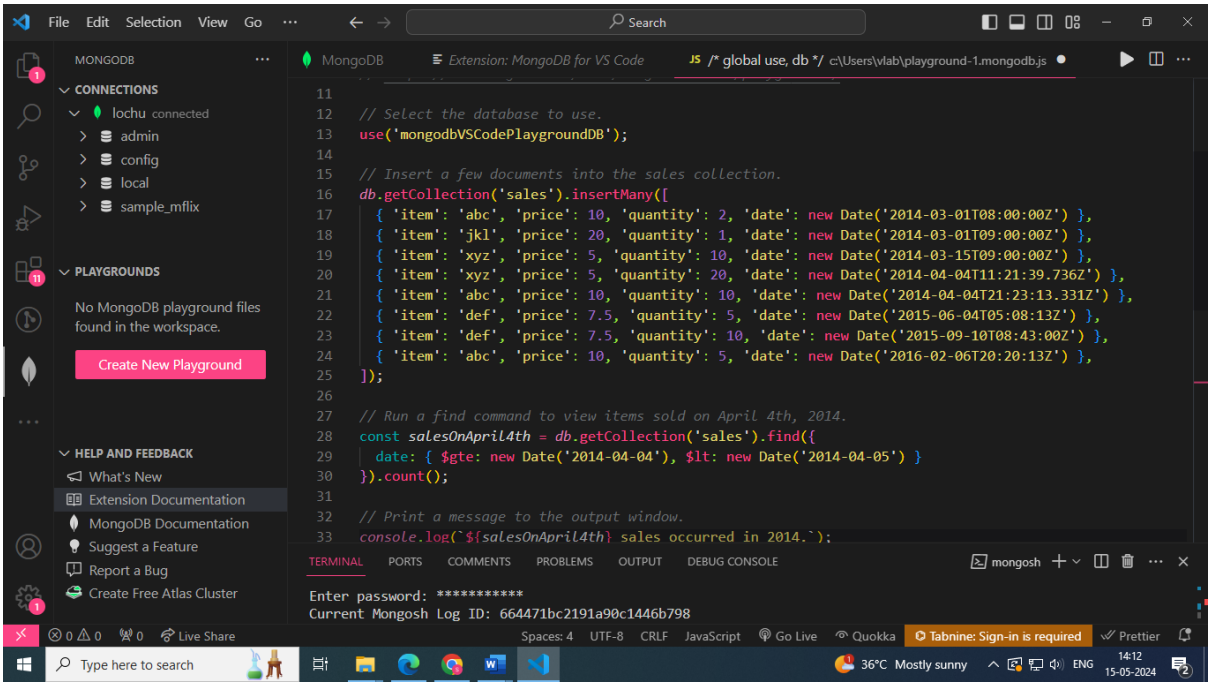
- Connect the cluster through mongoDB for VS code



- Copy the connection string from it and paste in the vs code connection string tab after installing the extension **MongoDB for VScode** in virtual studio







- To activate mongoDb shell → Go to mogodb atlass → click on the connect option → select the shell option → install mongosh zip file→extract the contents → go to bin folder containing the mongosh.exe file → copy the command to connect to mongosh
- Example : "**.\mongosh "mongodb+srv://lochu.4dc0gda.mongodb.net/" --apiVersion 1 --username lochu-55**"

PS C:\Users\vlab\mongodb\bin> .\mongosh "mongodb+srv://lochu.4dc0gda.mongodb.net/" --apiVersion 1 --username lochu-55

Enter password: **********

Current Mongosh Log ID: 664582ab2b2117203146b798

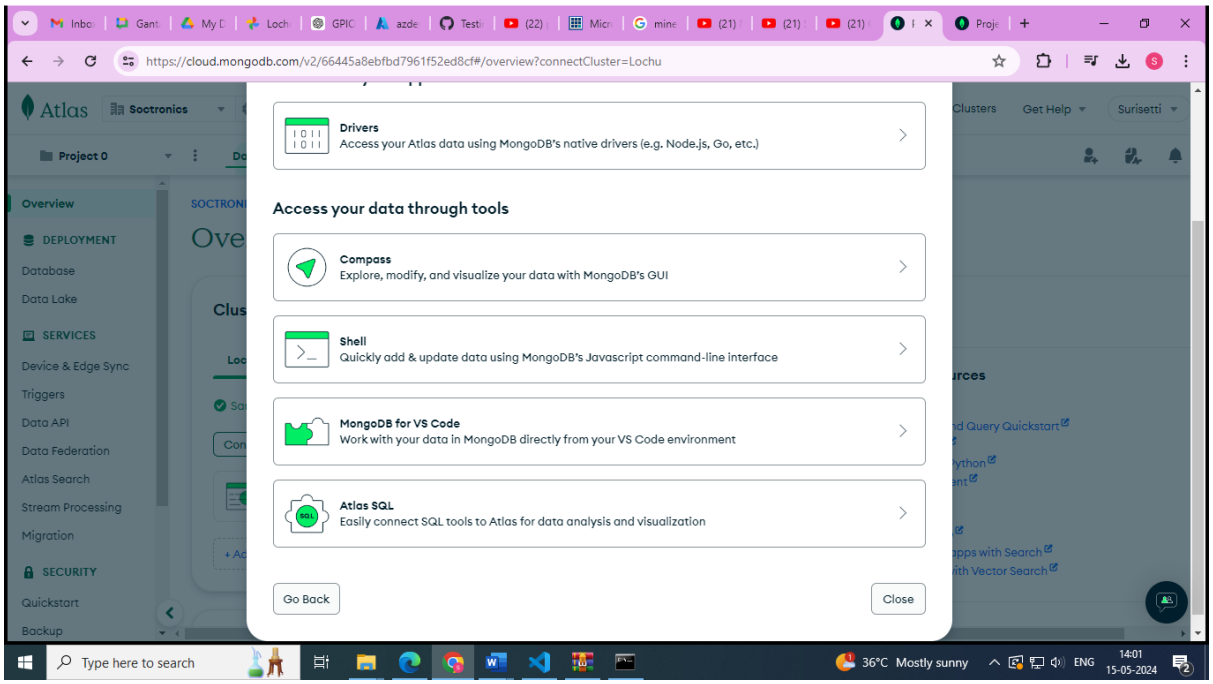
Connecting to:
mongodb+srv://<credentials>@lochu.4dc0gda.mongodb.net/?appName=mongosh+2.2.5

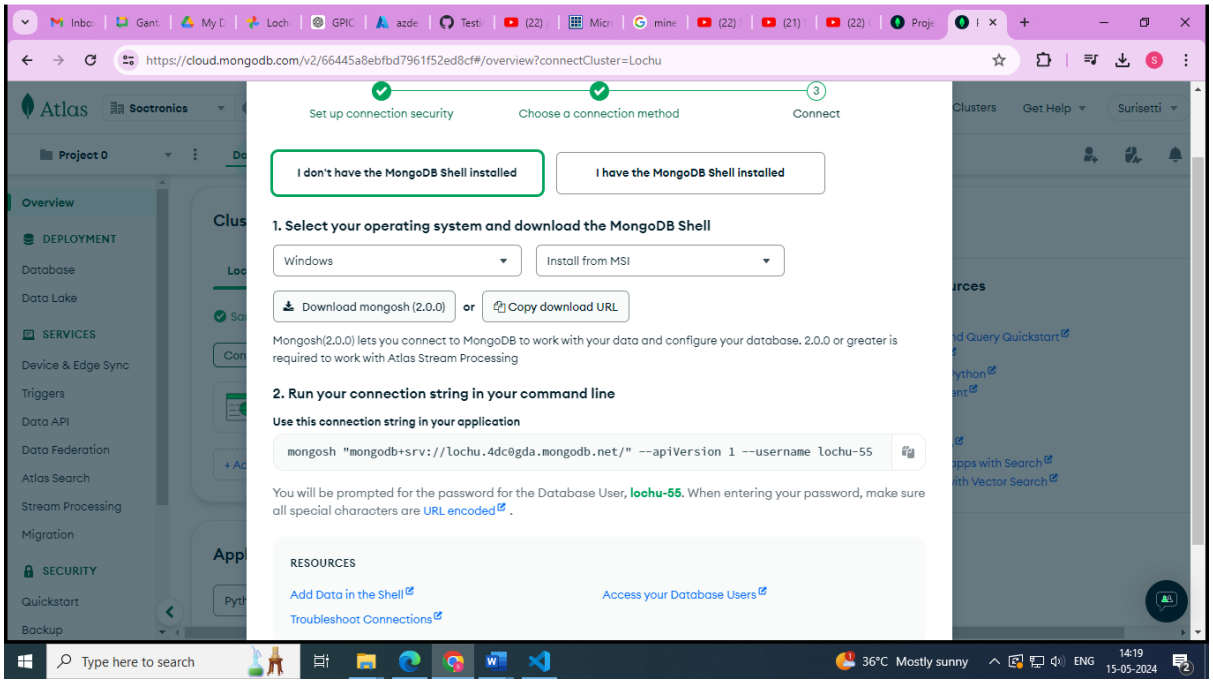
Using MongoDB: 7.0.8 (API Version 1)

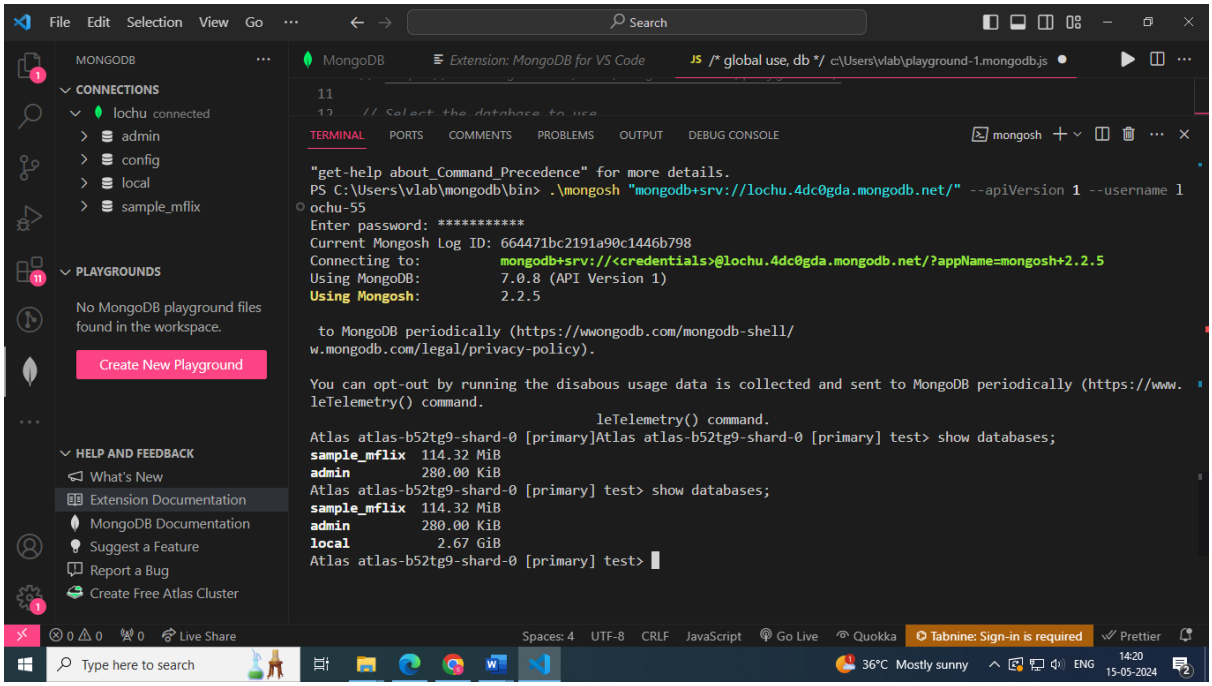
Using Mongosh: 2.2.5

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

Atlas atlas-b52tg9-shard-0 [primary] test>


https://cloud.mongodb.com/v2/66445a8ebfbd7961f52ed8cf#/overview?connectCluster=Lochu
Drivers
Access your Atlas data using MongoDB's native drivers (e.g. Node.js, Go, etc.)
Access your data through tools
Compass
Explore, modify, and visualize your data with MongoDB's GUI
Shell
Quickly add & update data using MongoDB's Javascript command-line interface
MongoDB for VS Code
Work with your data in MongoDB directly from your VS Code environment
Atlas SQL
Easily connect SQL tools to Atlas for data analysis and visualization
Go Back
Close
Type here to search
36°C Mostly sunny
ENG
14:01
15-05-2024


https://cloud.mongodb.com/v2/66445a8ebfbd7961f52ed8cf#/overview?connectCluster=Lochu
Set up connection security
Choose a connection method
3
Connect
I don't have the MongoDB Shell installed
I have the MongoDB Shell installed
1. Select your operating system and download the MongoDB Shell
Windows
Install from MSI
Download mongosh (2.0.0)
or
Copy download URL
Mongosh(2.0.0) lets you connect to MongoDB to work with your data and configure your database. 2.0.0 or greater is required to work with Atlas Stream Processing
2. Run your connection string in your command line
Use this connection string in your application
mongosh "mongodb+srv://lochu.4dc0gda.mongodb.net/" --apiVersion 1 --username lochu-55
You will be prompted for the password for the Database User, lochu-55. When entering your password, make sure all special characters are URL encoded .
RESOURCES
Add Data in the Shell
Access your Database Users
Troubleshoot Connections
Type here to search
36°C Mostly sunny
ENG
14:19
15-05-2024


File Edit Selection View Go
Search
MONGODB
CONNECTIONS
lochu connected
admin
config
local
sample_mflix
PLAYGROUNDS
No MongoDB playground files found in the workspace.
Create New Playground
HELP AND FEEDBACK
What's New
Extension Documentation
MongoDB Documentation
Suggest a Feature
Report a Bug
Create Free Atlas Cluster
MongoDB
Extension: MongoDB for VS Code
/* global use, db */ c:\Users\vlab\playground-1.mongodb.js
TERMINAL PORTS COMMENTS PROBLEMS OUTPUT DEBUG CONSOLE
mongosh
"get-help about_Command_Precedence" for more details.
PS C:\Users\vlab\mongodb\bin> .\mongosh "mongodb+srv://lochu.4dc0gda.mongodb.net/" --apiVersion 1 --username lochu-55
Enter password: **********
Current Mongosh Log ID: 664471bc2191a90c1446b798
Connecting to: mongodb+srv://<credentials>@lochu.4dc0gda.mongodb.net/?appName=mongosh+2.2.5
Using MongoDB: 7.0.8 (API Version 1)
Using Mongosh: 2.2.5
to MongoDB periodically (https://wwongodb.com/mongodb-shell/
w.mongodb.com/legal/privacy-policy).
You can opt-out by running the disabous usage data is collected and sent to MongoDB periodically (https://www.
leTelemetry() command.
leTelemetry() command.
Atlas atlas-b52tg9-shard-0 [primary]Atlas atlas-b52tg9-shard-0 [primary] test> show databases;
sample_mflix 114.32 MiB
admin 280.00 KiB
Atlas atlas-b52tg9-shard-0 [primary] test> show databases;
sample_mflix 114.32 MiB
admin 280.00 KiB
local 2.67 GiB
Atlas atlas-b52tg9-shard-0 [primary] test>
Live Share
Spaces: 4 UTF-8 CRLF JavaScript Go Live Quokka
Tabnine: Sign-in is required
Prettier
Type here to search
36°C Mostly sunny
ENG
14:20
15-05-2024

CREATE AND DROP DATABASES

Creating databases

- Use<database-name>
- The newly created database will NOT visible unless we insert any document inside it
- Make sure we see the message : **"switched to db <database-name>"**
- Check your working space DB

Dropping databases

- To remove database, use the command : **db.dropDatabase()**

IN MongoDB SHELL:

Atlas atlas-b52tg9-shard-0 [primary] test> db

test

Atlas atlas-b52tg9-shard-0 [primary] test> db.dropDatabase()

{ ok: 1, dropped: 'test' }

Atlas atlas-b52tg9-shard-0 [primary] test> show databases;

sample_mflix 114.32 MiB

admin 280.00 KiB

commands	usage
show databases;	list all databases in MongoDB
use <database-name>	select the database workspace as current working database
db	show your current working database
use <database-name to be dropped> db.dropDatabase()	to drop database, first we need to select the database

CREATE AND DROP COLLECTIONS

commands	usage
db.createCollection(name,options)	Creating collections
db.collection.drop()	Dropping collections

Process to create collection through MongoDB shell in VS code:

Atlas atlas-b52tg9-shard-0 [primary] new_database> use lochu-55

switched to db lochu-55

Atlas atlas-b52tg9-shard-0 [primary] lochu-55>show databases;

lochu-55 8.00 KiB

sample_mflix 71.63 MiB

admin 280.00 KiB

local 2.67 GiB

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> db

lochu-55

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> db.createCollection("Products");

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> db.createCollection("items");

{ ok: 1 }

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> db

lochu-55

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> db.Products.drop()

true

DATA TYPES

- BSON
- JSON
- Integer
- Boolean
- Double
- Arrays
- Object
- Null
- Date
- Timestamp
- Object Id
- Code

➔ **String** − This is the most commonly used datatype to store the data. String in MongoDB must be UTF-8 valid.
➔ **Integer** − This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon your server.
➔ **Boolean** − This type is used to store a boolean (true/ false) value.
➔ **Double** − This type is used to store floating point values.
➔ **Min/ Max keys** − This type is used to compare a value against the lowest and highest BSON elements.
➔ **Arrays** − This type is used to store arrays or list or multiple values into one key.
➔ **Timestamp** − ctimestamp. This can be handy for recording when a document has been modified or added.
➔ **Object** − This datatype is used for embedded documents.
➔ **Null** − This type is used to store a Null value.
➔ **Symbol** − This datatype is used identically to a string; however, it's generally reserved for languages that use a specific symbol type.
➔ **Date** − This datatype is used to store the current date or time in UNIX time format. You can specify your own date time by creating object of Date and passing day, month, year into it.
➔ **Object ID** − This datatype is used to store the document's ID.
➔ **Binary data** − This datatype is used to store binary data.
➔ **Code** − This datatype is used to store JavaScript code into the document.
➔ **Regular expression** − This datatype is used to store regular expression.

Difference between BSON and JSON

- JSON based databases usually return query results which can be effortlessly parsed , having modest and nix transformation, straightforwardly using JavaScript along with most well-liked programming languages
- In the case of MongoDB , data representation is done in JSON document format, but here the JSON is binary-encoded , which is termed as BSON

- BSON is the extended version of the JSON model, which is providing additional data types, makes performance to be competent o encode and decode in diverse languages and ordered fields
- JSON
    - {“name”:” “, “address” : “ “, ….}
    - Easy to parse , easy to render, most languages
- BSON
    - MongoDB datatype
    - Stores and process data
    - Binary encoded JSON → BSON
    - It has some extended data types which are not supported by JSON (date, timestamp, object ID)

CRUD OPERATIONS ON COLLECTIONS

INSERT DOCUMENT

- Insert is used for creating new documents inside the collection
- To insert any document into collection

  Db.<collection-name>.insert({“name”:”lochu”})

- To insert many documents at once into collection

  Db.<collection-name>.insertMany{
  {“name” : “lochu”},
  {“name” : “sai”}
)
- We can insert any number od documents into the collection
- Every document that we insert will have a unique key “_id”
- The value for this key is always unique and 24 characters
- _id is the primary key in your collection
- We can change _id , but it is tricky. DO NOT DO THIS

Atlas atlas-b52tg9-shard-0 [primary] test> **use lochu-55**

switched to db lochu-55

Atlas atlas-b52tg9-shard-0 [primary] lochu-55>
db.student_data.insert({"name":"lochu"});

DeprecationWarning: Collection.insert() is

deprecated. Use insertOne, insertMany, or bulkWrite.

{

acknowledged: true,

insertedIds: { '0': ObjectId('6645a59f2b2117203146b799') }

Atlas atlas-b52tg9-shard-0 [primary] lochu-55>
db.student_data.insertOne({"name":"veena"});

{

acknowledged: true,

insertedId: ObjectId('6645a6122b2117203146b79a')

Atlas atlas-b52tg9-shard-0 [primary] lochu-55>
db.student_data.insertMany([{"name":"veena","address":"hyderabad"},{"name":"veena","address":"vijayawada"}]);

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6645a7a92b2117203146b79b'),
    '1': ObjectId('6645a7a92b2117203146b79c')
  }
}
```

UPDATE DOCUMENTS

- To update any document into collection , update can be applied with
  - update
  - updateOne
  - updateMany
- command to update

```
db.<collection-name..update(
{“name”:”lochu”},
{
        $set:{
                “key” : “value”
        }
})
```

- example
  - **db.student_data.updateOne(**

```
{ "name": "lochu" },{
        $set: {
        "age": "20",
        "address": "hyderabad"
        }
 }
)
```

  - **db.student_data.updateMany(**

```
{ "age": "20" },{
        $set: {
```

```
                    "branch": "ECE"
                    }
                }
            );
```

```
Atlas atlas-b52tg9-shard-0 [primary] lochu-55> db.student_data.updateOne(
 // Query criteria
...   { "name": "veena" },   {
...     $set: { // Update operation to set new values
...       "age": "20",
...       "address": "bangalore"
...     }
...   }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-b52tg9-shard-0 [primary] lochu-55> db.student_data.updateOne(
...   { "name": "veena" },   {
...     $set: {
...       "age": "20",
...       "address": "bangalore"
...     }
...   }
... )
{
  acknowledged: true,
```

```
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> db.student_data.updateOne(
...   { "name": "lochu" },{    $set: {
...         "age": "20",
...         "address": "hyderabad"
...     }
...   }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}


Atlas atlas-b52tg9-shard-0 [primary] lochu-55> db.student_data.updateMany(
...   { "age": "20" },{
...     $set: {
...           "branch": "ECE"
...                       }
...     }
... );
{
```

```
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

READ DOCUMENTS FROM COLLECTIONS

Command	Usage
find()	Find all documentsin collection Db.collection.find()
findOne()	Find first document in collection
find({“key1”:”value” , “key2”:”value2”});	By setting query conditions
findOneAndReplace({“key1”:”value” , “key2”:”value2”},<replacement>);	To find based on key and value and replace it with replacement string
findOneAndDelete({“key1”:”value” , “key2”:”value2”});	To find and delete the document
findOneAndUpdate({“key1”:”value”} ,{$set : { “key2”:”value2”}});	To find and update the document with required fields

→find()

```
Atlas atlas-b52tg9-shard-0 [primary] lochu-55> show databases;
lochu-55       80.00 KiB
sample_mflix   71.57 MiB
admin          280.00 KiB
local          2.67 GiB
Atlas atlas-b52tg9-shard-0 [primary] lochu-55> db.student_data.find()
[
  {
    _id: ObjectId('6645a59f2b2117203146b799'),
```

```
  name: 'lochu',
  address: 'hyderabad',
  age: '20',
  branch: 'ECE'
},
{
  _id: ObjectId('6645a6122b2117203146b79a'),
  name: 'veena',
  address: 'bangalore',
  age: '20',
  branch: 'ECE'
},
{
  _id: ObjectId('6645a7a92b2117203146b79b'),
  name: 'veena',
  address: 'bangalore'
},
{
  _id: ObjectId('6645a7a92b2117203146b79c'),
  name: 'veena',
  address: 'bangalore'
}
```

→findOne()

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> db.student_data.findOne()

```
{
  _id: ObjectId('6645a59f2b2117203146b799'),
  name: 'lochu',
  address: 'hyderabad',
  age: '20',
```

```
  branch: 'ECE'
}
```

```
Atlas atlas-b52tg9-shard-0 [primary] lochu-55>
db.student_data.find({"address":"hyderabad"});
[
  {
    _id: ObjectId('6645a59f2b2117203146b799'),
    name: 'lochu',
    address: 'hyderabad',
    age: '20',
    branch: 'ECE'
  }
]
```

Atlas atlas-b52tg9-shard-0 [primary] lochu-55>
db.student_data.find({"address":"banglore"},{"age":"20"});

Atlas atlas-b52tg9-shard-0 [primary] lochu-55>
db.student_data.find({"address":"bangalore"},{"age":"20"});

```
[
  { _id: ObjectId('6645a6122b2117203146b79a'), age: '20' },
  { _id: ObjectId('6645a7a92b2117203146b79b'), age: '20' },
  { _id: ObjectId('6645a7a92b2117203146b79c'), age: '20' }
```

→ findOneAndReplace()

Atlas atlas-b52tg9-shard-0 [primary] lochu-55>
db.student_data.findOneAndReplace({"branch":
"ECE"},{"mobile":"8341130962","name":"lochu"});

null

Atlas atlas-b52tg9-shard-0 [primary] lochu-55>
db.student_data.findOneAndReplace({"mobile":"8341130962"},{"name":"lochu","age
":"20"});

{ _id: ObjectId('6645a59f2b2117203146b799'), mobile: '8341130962' }

Atlas atlas-b52tg9-shard-0 [primary] lochu-55>
db.student_data.findOneAndReplace({"name":"veena"},{"age":"22","name":"sai"});

{

_id: ObjectId('6645a7a92b2117203146b79b'),

name: 'veena',

address: 'bangalore'

}

→ findOneAndDelete()

Atlas atlas-b52tg9-shard-0 [primary] lochu-55>
db.student_data.findOneAndDelete({"name":"veena"});

{

_id: ObjectId('6645a7a92b2117203146b79c'),

name: 'veena',

address: 'bangalore'

}

→ findOneAndUpdate()

Atlas atlas-b52tg9-shard-0 [primary] lochu-55>
db.student_data.findOneAndUpdate({"mobile":"8341130962"},{$set:{"name":"divya"
}});

DELETE DOCUMENTS FROM COLLECTIONS

Command	Usage
deleteOne()	Delete a particular document of specified condition(criteria)
deleteMany({})	Will delete many documents at once When passed with empty curly braces , it will delete all documents in collection

→deleteOne()

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> **db.student_data.delete**

db.student_data.deleteMany db.student_data.deleteOne

Atlas atlas-b52tg9-shard-0 [primary] lochu-55>
db.student_data.deleteOne({"name":"sai"});

{ acknowledged: true, deletedCount: 1 }

→deleteMany()

Atlas atlas-b52tg9-shard-0 [primary] lochu-55>
db.student_data.insertMany([{"name":"veena","address":"hyderaba":"hyderabad"},
{"name":"veena","address":"vijayawada"}]);

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6645c67a2b2117203146b79d'),
    '1': ObjectId('6645c67a2b2117203146b79e')
  }
}
```

Atlas atlas-b52tg9-shard-0 [primary] lochu-55>
db.student_data.deleteMany({"name":"veena"});

{ acknowledged: true, deletedCount: 2 }

QUERIES

Conditions in find method

- we can use multiple operations inside find method
- using operators, we add more search power to find method
- various conditions that can be used are
    - equality – eq
    - less than - lt
    - less than equal - lte
    - greater than – gt
    - greater than equal - gte
    - not equal

- all conditions "must" be satisfied to return the document
    - $and – and operation
      →match all conditions mentioned in find method
      →e.g. db.leads.find({$and:[{},{}]})
    - $or – or operation
      →match any condition in find method

Let us consider there are three documents in the collection named "student_data" within database "lochu-55"


```
{
  "_id": {
    "$oid": "6645d00d2b2117203146b79f"
  },
  "name": "siva",
  "age": "54",
  "address": "pune",
  "salary": "75000",
  "mobile": "7780783562"
}


{
  "_id": {
    "$oid": "6645a59f2b2117203146b799"
  },
  "name": "lochu",
  "age": "20",
  "address": "vijayawada",
  "mobile": "9502700962",
  "salary": "20000"
```

```
    }
```


```
{
  "_id": {
    "$oid": "6645a6122b2117203146b79a"
  },
  "mobile": "8341130962",
  "name": "divya",
  "address": "hyderabad",
  "age": "18",
  "salary": "25000"
}
```


→find() – returns all the documents in the collection

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> **db.student_data.find()**


```
[
  {
    _id: ObjectId('6645a59f2b2117203146b799'),
    name: 'lochu',
    age: '20',
    address: 'vijayawada',        y] lochu-55>
    mobile: '9502700962',
    salary: '20000'
  },
  {
    _id: ObjectId('6645a6122b2117203146b79a'),
    mobile: '8341130962',
    name: 'divya',
    address: 'hyderabad',
    age: '18',
    salary: '25000'
  },
  {
    _id: ObjectId('6645d00d2b2117203146b79f'),
```

```
    name: 'siva',
    age: '54',
    address: 'pune',
    salary: '75000',
    mobile: '7780783562'
  }
]
```

→find with conditons

Atlas atlas-b52tg9-shard-0 [primary] lochu-55>
db.student_data.find({"address":"pune"})

```
[
  {
    _id: ObjectId('6645d00d2b2117203146b79f'),03146b79f'),
    name: 'siva',
    age: '54',
    address: 'pune',
    salary: '75000',
    mobile: '7780783562'
]
```

→find method with multiple conditions, like $and operator

Atlas atlas-b52tg9-shard-0 [primary] lochu-55>
db.student_data.find({"address":"pune","age":"54"})

```
[
  {
    _id: ObjectId('6645d00d2b2117203146b79f'),
    name: 'siva',
    age: '54',
```

```
    address: 'pune',
    salary: '75000',
    mobile: '7780783562'
  }
]
```

→find method with age less than or equal 50

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> **db.student_data.find({"age":{$lte : "50"}})**

```
[
  {
    _id: ObjectId('6645a59f2b2117203146b799'),
    name: 'lochu',
    age: '20',
    address: 'vijayawada',
    mobile: '9502700962',
    salary: '20000'
  },
  {
    _id: ObjectId('6645a6122b2117203146b79a'),
    mobile: '8341130962',
    name: 'divya',
    address: 'hyderabad',
    age: '18',
    salary: '25000'
  }
```

→find method with age greater than or equal 50

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> db.student_data.find({"age":{$gte : "50"}})

```
[
  {
    _id: ObjectId('6645d00d2b2117203146b79f'),
    name: 'siva',
    age: '54',
    address: 'pune',
    salary: '75000',
    mobile: '7780783562'
  }
]
```

→find method with age equal to 20

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> db.student_data.find({"age":{$eq : "22"}})

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> db.student_data.find({"age":{$eq : "20"}})

```
[
  {
    _id: ObjectId('6645a59f2b2117203146b799'),
    name: 'lochu',
    age: '20',
    address: 'vijayawada',
    mobile: '9502700962',
    salary: '20000'
  }
]
```

→find using $and operator

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> db.student_data.find({$and : [{"address":"pune"},{"age":{$gt :"50"}}]})

```
[
  {
    _id: ObjectId('6645d00d2b2117203146b79f'),
    name: 'siva',
    age: '54',
    address: 'pune',
    salary: '75000',
    mobile: '7780783562'
  }
]
```

→find using $or operator

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> db.student_data.find({$or : [{"salary":{$lt :"25000"}},{"age":{$gt :"50"}}]})

```
[
  {
    _id: ObjectId('6645a59f2b2117203146b799'),
    name: 'lochu',
    age: '20',
    address: 'vijayawada',
    mobile: '9502700962',
    salary: '20000'
  },
  {
    _id: ObjectId('6645d00d2b2117203146b79f'),
    name: 'siva',
    age: '54',
    address: 'pune',
```

salary: '75000',

mobile: '7780783562'

}

SELECTING FIELDS

- To select specific fields from documents from collections , we need to specify the field as “1” or “0”
  - Db.leads.find({<condition>},{“tax”:1,”_id”:0})

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> **db.student_data.find({},{"age":1})**

```
[
  { _id: ObjectId('6645a59f2b2117203146b799'), age: '20' },
  { _id: ObjectId('6645a6122b2117203146b79a'), age: '18' },
  { _id: ObjectId('6645d00d2b2117203146b79f'), age: '54' }
]
```

Atlas atlas-b52tg9-shard-0 [primary] lochu-55>
db.student_data.find({},{"age":1,"_id":0})

[{ age: '20' }, { age: '18' }, { age: '54' }]

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> **db.student_data.find({"age" : {$lte : "50"}},{"age":1,"_id":0})**

[{ age: '20' }, { age: '18' }]

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> **db.student_data.find({"age" : {$lte : "50"}},{"name":1,"age":1,"_id":0})**

[{ name: 'lochu', age: '20' }, { name: 'divya', age: '18' }]

PROJECTION

- Projection is nothing but another name for selecting fields for showing or hiding
- We need to specify the field as “1” or “0’
- Db.leads.find({<condition>},{“tax”:1,”_id”:0})
- By default it will bring up all keys/values from all documents in collections
- Find({},{“tax”:1,”name”:1,”_id”:0}
    - ➔ MongoDB gets all matching documents
    - ➔ Extract the keys that we have asked for
    - ➔ This result set is returned/projected

AGGREGATION

What is aggregation?

- Aggregate is very similar to the find command, where you can provide the criteria for you the form of JSON documents
- The key element in aggregation is called the pipeline
- It also helps us in performing few operations like min, max, sum etc.
- The command to use aggregation s:
    - **db.leads.aggregate (pipeline,options)**
    - **What’s pipeline?**
        - ➔ A sequence of data aggregation operations or stages
        - ➔ Pipeline is an array
    - **What are options?**
        - ➔ Documents can be passed as well
- **What are valid aggregate stages?**
    - $count
    - $group
    - $limit
    - $lookup
    - $match
    - $merge
    - $sort
    - $project
    - $unwind
    - $unset
    - And many more …..
- **Pipeline definition**

```
pipeline = [
        {…},
        {…},
        {…},
];
```

- **Example:**

```
Var pipeline = [
        {$group : {“_id” : “$city”}},
        {$sort : {“name”:1}},
        {$limit : 4}
];
db.leads.aggregate(pipeline)
```

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> **var pipeline = [**

... {$sort : {"name":1}}

...];

db.student_data.aggregate(pipeline)


```
[
  {
    _id: ObjectId('6645a6122b2117203146b79a'),
    mobile: '8341130962',
    name: 'divya',
    address: 'hyderabad',
    age: '18',
    salary: '25000'
  },
  {
    _id: ObjectId('6645a59f2b2117203146b799'),
    name: 'lochu',
    age: '20',
    address: 'vijayawada',
    mobile: '9502700962',
    salary: '20000'
  },
  {
```

```
    _id: ObjectId('6645d00d2b2117203146b79f'),
    name: 'siva',
    age: '54',
    address: 'pune',
    salary: '75000',
    mobile: '7780783562'
  }
]
```

Atlas atlas-b52tg9-shard-0 [primary] lochu-55>

```
var pipeline = [
{ $group: { "_id": "address" } },
{ $limit: 2 }
];
db.student_data.aggregate(pipeline);
```

[{ _id: 'address' }]

LIMIT AND SKIP

We may not always want all documents all the time

- Db.collection.find().limit(4)

We may want to skip documents we don't need

- Db.collection.find().skip(3)
- Always remember – skip will skip sequentially not random

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> **db.student_data.find().limit(2)**

```
[
  {
    _id: ObjectId('6645a59f2b2117203146b799'),
    name: 'lochu',
```

```
    age: '20',
    address: 'vijayawada',
    mobile: '9502700962',
    salary: '20000'
  },
  {
    _id: ObjectId('6645a6122b2117203146b79a'),
    mobile: '8341130962',
    name: 'divya',
    address: 'hyderabad',
    age: '18',
    salary: '25000'
  }
]
```

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> **db.student_data.find().skip(1)**

```
[
  {
    _id: ObjectId('6645a6122b2117203146b79a'),
    mobile: '8341130962',
    name: 'divya',
    address: 'hyderabad',
    age: '18',
    salary: '25000'
  },
  {
    _id: ObjectId('6645d00d2b2117203146b79f'),
    name: 'siva',
```

```
    age: '54',
    address: 'pune',
    salary: '75000',
    mobile: '7780783562'
  }
]
```

SORTING

- We will need to sort the record set before passing it to next logical operation
- To sort we can use the below command
  - ➔ Db.collection.find().sort({“leadname”:1})
  - ➔ 1 : ascending
  - ➔ -1 : descending

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> **db.student_data.find().sort({"age":-1})**


```
[
  {
    _id: ObjectId('6645d00d2b2117203146b79f'),
    name: 'siva',
    age: '54',
    address: 'pune',
    salary: '75000',
    mobile: '7780783562'
  },
  {
    _id: ObjectId('6645a59f2b2117203146b799'),
    name: 'lochu',
    age: '20',
    address: 'vijayawada',
    mobile: '9502700962',
    salary: '20000'
  },
  {
    _id: ObjectId('6645a6122b2117203146b79a'),
    mobile: '8341130962',
    name: 'divya',
    address: 'hyderabad',
    age: '18',
    salary: '25000'
```

```
  }
]
```

Atlas atlas-b52tg9-shard-0 [primary] lochu-55>
db.student_data.find().sort({"name":1})

```
[
  {
    _id: ObjectId('6645a6122b2117203146b79a'),
    mobile: '8341130962',
    name: 'divya',
    address: 'hyderabad',
    age: '18',
    salary: '25000'
  },
  {
    _id: ObjectId('6645a59f2b2117203146b799'),
    name: 'lochu',
    age: '20',
    address: 'vijayawada',
    mobile: '9502700962',
    salary: '20000'
  },
  {
    _id: ObjectId('6645d00d2b2117203146b79f'),
    name: 'siva',
    age: '54',
    address: 'pune',
    salary: '75000',
    mobile: '7780783562'
  }
]
```

CREATING INDEXES

- Indexes are the fastest way to find information
- Indexes concept is same as that you already would know in SQL
- By default – every collection will have an index on "_id" key
- How do we create an index on collection?
  - ➔ Db.leads.ensureIndex({"name":1})
  - ➔ Db.leads.createIndex({"name":-1})
  - ➔ 1 : creates index in ascending order
  - ➔ -1 : creates index in descending order
- You can drop a particular index using the dropIndex() method of MongoDB
  - ➔ db.COLLECTION_NAME.dropIndex({KEY:1})
- getIndexes() method : This method returns the description of all the indexes int the collection.\
  - ➔ db.COLLECTION_NAME.getIndexes()

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> **db.student_data.ensureIndex({"name":1})**

['name_1']

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> **db.student_data.createIndex({"age":-1})**

age_-1

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> **db.student_data.dropIndex({"age":-1})**

{
 nIndexesWas: 3,
 ok: 1,
 '$clusterTime': {
  clusterTime: Timestamp({ t: 1715921382, i: 2 }),
  signature: {
   hash: Binary.createFromBase64('w2/6PwyZU3SiEMhpFJKIbUD69tg=', 0),
   keyId: Long('7327032886341664774')
  }
 },

```
  operationTime: Timestamp({ t: 1715921382, i: 2 })
}
```

Atlas atlas-b52tg9-shard-0 [primary] lochu-55> db.student_data.getIndexes()

```
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'name_1' }
]
```

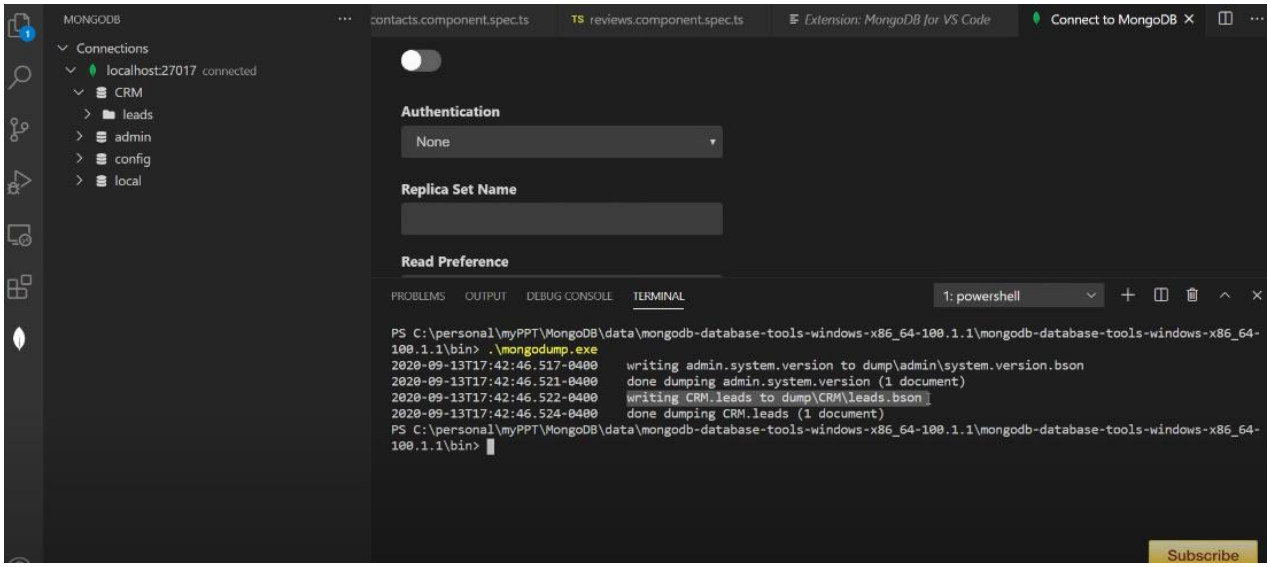
BACK UP AND RESTORE

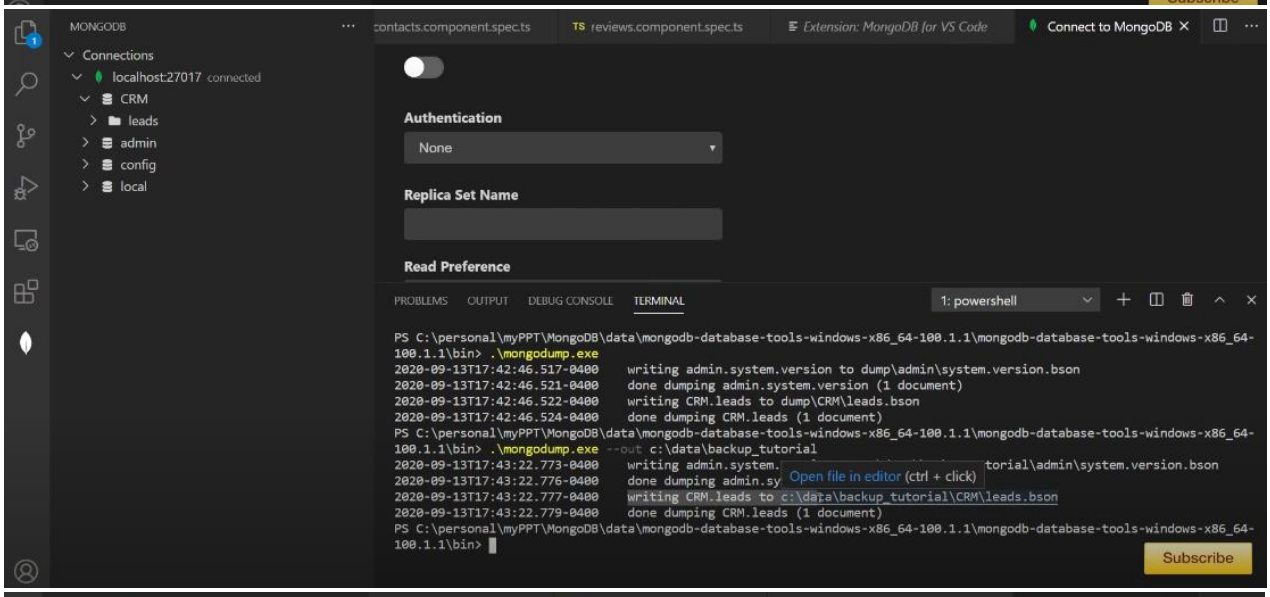
Steps to create back up of MongoDB data

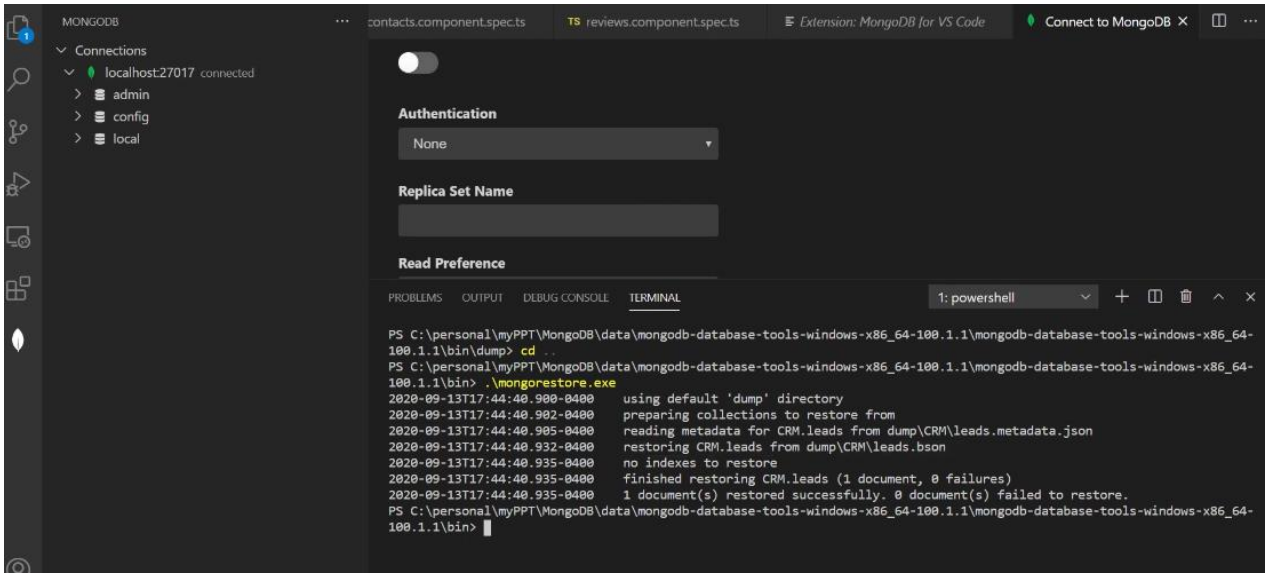
- First create a folder where you want save your data
- Next, run the command “mongodump.exe”
- Verify the data dump and folder dumped correctly

Steps to restore of MongoDB data

- Make sure you have got the backup of the data dump from MongoDB
- Next, run the command “mongostore.exe”
- Verify the data dump and folder dumped correctly


MONGODB
Connections
localhost:27017 connected
CRM
leads
admin
config
local
Authentication
None
Replica Set Name
Read Preference
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: powershell
PS C:\personal\myPPT\MongoDB\data\mongodb-database-tools-windows-x86_64-100.1.1\mongodb-database-tools-windows-x86_64-100.1.1\bin> .\mongodump.exe
2020-09-13T17:42:46.517-0400 writing admin.system.version to dump\admin\system.version.bson
2020-09-13T17:42:46.521-0400 done dumping admin.system.version (1 document)
2020-09-13T17:42:46.522-0400 writing CRM.leads to dump\CRM\leads.bson
2020-09-13T17:42:46.524-0400 done dumping CRM.leads (1 document)
PS C:\personal\myPPT\MongoDB\data\mongodb-database-tools-windows-x86_64-100.1.1\mongodb-database-tools-windows-x86_64-100.1.1\bin>
Subscribe


MONGODB
Connections
localhost:27017 connected
CRM
leads
admin
config
local
Authentication
None
Replica Set Name
Read Preference
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: powershell
PS C:\personal\myPPT\MongoDB\data\mongodb-database-tools-windows-x86_64-100.1.1\mongodb-database-tools-windows-x86_64-100.1.1\bin> .\mongodump.exe
2020-09-13T17:42:46.517-0400 writing admin.system.version to dump\admin\system.version.bson
2020-09-13T17:42:46.521-0400 done dumping admin.system.version (1 document)
2020-09-13T17:42:46.522-0400 writing CRM.leads to dump\CRM\leads.bson
2020-09-13T17:42:46.524-0400 done dumping CRM.leads (1 document)
PS C:\personal\myPPT\MongoDB\data\mongodb-database-tools-windows-x86_64-100.1.1\mongodb-database-tools-windows-x86_64-100.1.1\bin> .\mongodump.exe --out c:\data\backup_tutorial
2020-09-13T17:43:22.773-0400 writing admin.system.
tutorial\admin\system.version.bson
2020-09-13T17:43:22.776-0400 done dumping admin.sy
Open file in editor (ctrl + click)
2020-09-13T17:43:22.777-0400 writing CRM.leads to c:\data\backup_tutorial\CRM\leads.bson
2020-09-13T17:43:22.779-0400 done dumping CRM.leads (1 document)
PS C:\personal\myPPT\MongoDB\data\mongodb-database-tools-windows-x86_64-100.1.1\mongodb-database-tools-windows-x86_64-100.1.1\bin>
Subscribe


MONGODB
Connections
localhost:27017 connected
admin
config
local
Authentication
None
Replica Set Name
Read Preference
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: powershell
PS C:\personal\myPPT\MongoDB\data\mongodb-database-tools-windows-x86_64-100.1.1\mongodb-database-tools-windows-x86_64-100.1.1\bin\dump> cd ..
PS C:\personal\myPPT\MongoDB\data\mongodb-database-tools-windows-x86_64-100.1.1\mongodb-database-tools-windows-x86_64-100.1.1\bin> .\mongorestore.exe
2020-09-13T17:44:40.900-0400 using default 'dump' directory
2020-09-13T17:44:40.902-0400 preparing collections to restore from
2020-09-13T17:44:40.905-0400 reading metadata for CRM.leads from dump\CRM\leads.metadata.json
2020-09-13T17:44:40.932-0400 restoring CRM.leads from dump\CRM\leads.bson
2020-09-13T17:44:40.935-0400 no indexes to restore
2020-09-13T17:44:40.935-0400 finished restoring CRM.leads (1 document, 0 failures)
2020-09-13T17:44:40.935-0400 1 document(s) restored successfully. 0 document(s) failed to restore.
PS C:\personal\myPPT\MongoDB\data\mongodb-database-tools-windows-x86_64-100.1.1\mongodb-database-tools-windows-x86_64-100.1.1\bin>