LAB 9 - MongoDB and NoSQL

Introduction to MongoDB and NoSQL Concepts

The Rise of NoSQL Databases

In the landscape of database technology, NoSQL databases have emerged as a powerful alternative to traditional relational database systems (RDBMS). Unlike RDBMS, which organize data into tables with rows and columns, NoSQL databases offer a variety of data models, including document, key-value, wide-column, and graph formats. This flexibility allows them to handle large volumes of structured, semi-structured, and unstructured data, making NoSQL databases particularly suited for big data and real-time web applications.

Why NoSQL?

- **Scalability:** NoSQL databases can scale out horizontally, making them ideal for applications that require high throughput and large volumes of data.
- **Flexibility:** They allow for a more flexible and dynamic schema. This is particularly beneficial for applications that evolve rapidly, as changes can be made without affecting the entire database.
- **Performance:** Optimized for specific data models, NoSQL databases can offer improved performance for certain types of applications, such as those requiring real-time analytics or full-text search.

MongoDB: A Document-Oriented Database

MongoDB is a leading NoSQL database that stores data in flexible, JSON-like documents. In MongoDB, each record is a document consisting of key-value pairs, similar to objects in JSON (JavaScript Object Notation). Documents are grouped into collections, which are analogous to tables in a relational database. However, unlike tables, each document in a collection can have a different structure. This document model is intuitive for developers, as it maps naturally to objects in programming languages.

Key Features of MongoDB

- **Document Model:** The document data model is a powerful way to store and retrieve data that allows developers to work with data more naturally.
- **Indexing:** MongoDB supports secondary indexes, allowing for efficient queries across any field or combination of fields within a document.
- **Aggregation Framework:** A powerful set of operations that allows for complex data aggregation and analysis directly within the database.
- **Replication:** MongoDB provides high availability with replica sets, a group of mongod instances that maintain the same data set.
- **Sharding:** For scalability, MongoDB can distribute data across multiple servers using sharding, enabling horizontal scaling.

MongoDB’s Document Model

At the heart of MongoDB is its document model. A MongoDB document is a group of key-value pairs. Documents have dynamic schema, meaning that documents in the same collection do not need to have the same set of fields, and the data type for a field can differ across documents.

Example of a MongoDB Document:

```
{
 "name": "John Doe",
 "age": 30,
 "address": {
   "street": "123 Elm Street",
   "city": "Anytown"
},
 "hobbies": ["reading", "cycling", "photography"]
}
```

In this example, the document represents a user with basic information, an embedded document (address), and an array of hobbies. This structure is highly readable and easy to work with, as it closely resembles data structures used in modern programming languages.

Understanding MongoDB Collections

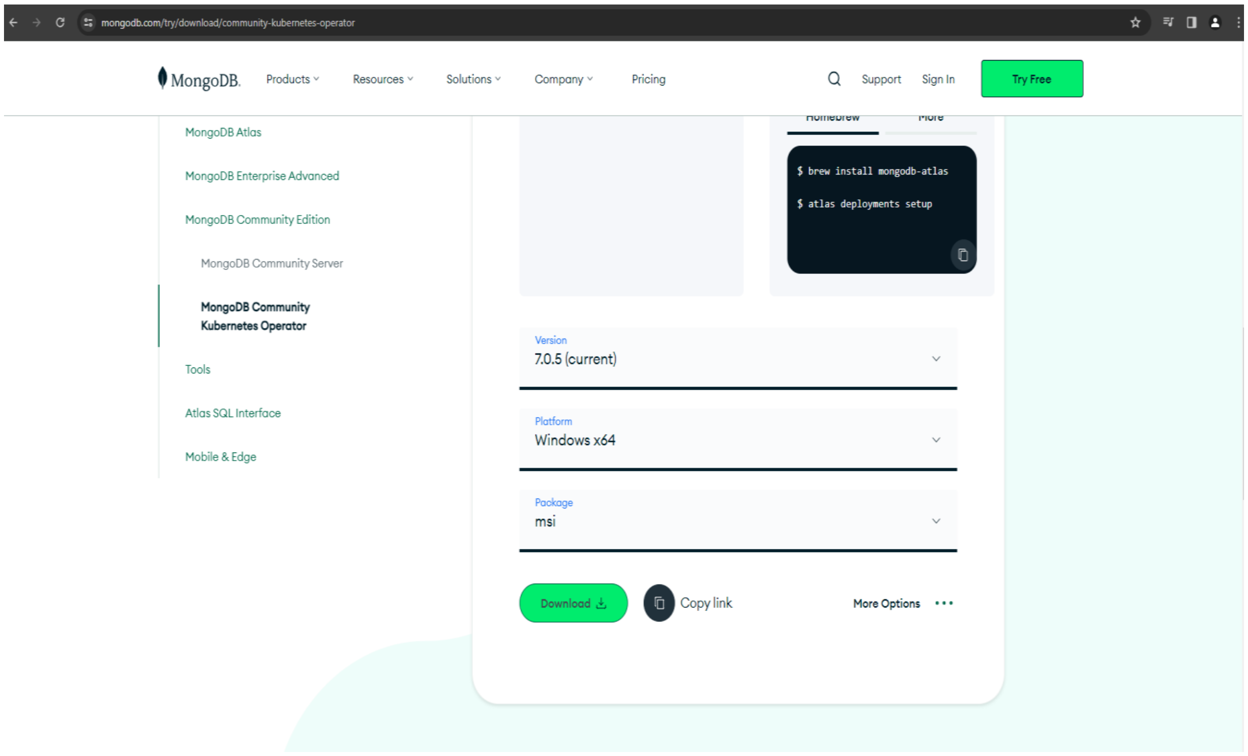
A collection in MongoDB is a group of documents. Collections are analogous to tables in relational databases but without a fixed schema. This means that documents within the same collection can have different fields. However, it is common practice to store documents with a similar structure in the same collection to maintain data consistency.

Advantages of MongoDB’s Schema-less Design

- **Flexibility:** Easily adapt to changes in your data model without the need to modify a central database schema.
- **Development Speed:** Accelerate development cycles by allowing developers to store data in a way that aligns closely with their application’s objects.

In the next sections, you'll learn how to set up your MongoDB environment, design a schema for a social media platform, and implement fundamental database operations. By simulating a real-world application, you'll gain hands-on experience with MongoDB's versatile capabilities, from basic CRUD operations to complex queries and aggregations.

Download link for mongodb compass is
https://www.mongodb.com/try/download/community-kubernetes-operator
You will have to select the package and download the .msi file according to your system specifications.

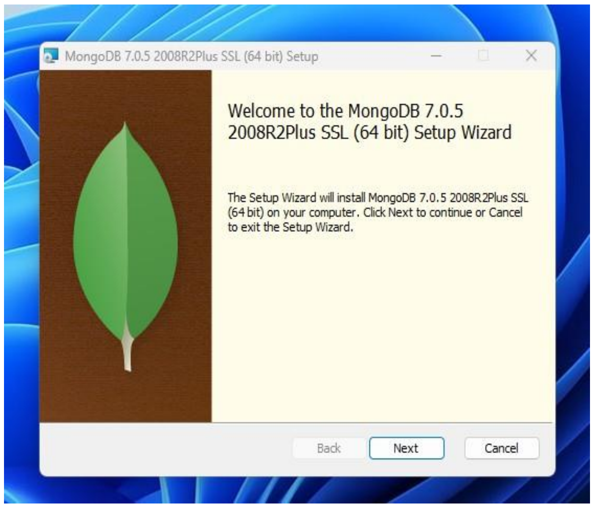


```
$ brew install mongodb-atlas

$ atlas deployments setup
```

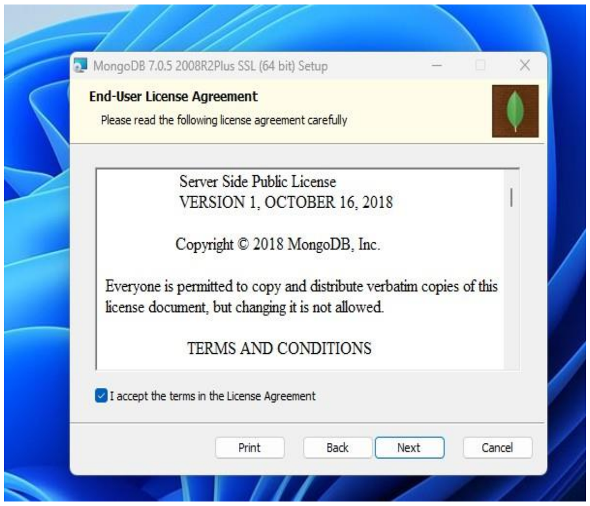
Version: 7.0.5 (current)
Platform: Windows x64
Package: msi

Now, you will have to install the downloaded .msi file. The steps are as shown as the following snapshots


MongoDB 7.0.5 2008R2Plus SSL (64 bit) Setup

Welcome to the MongoDB 7.0.5 2008R2Plus SSL (64 bit) Setup Wizard

The Setup Wizard will install MongoDB 7.0.5 2008R2Plus SSL (64 bit) on your computer. Click Next to continue or Cancel to exit the Setup Wizard.


MongoDB 7.0.5 2008R2Plus SSL (64 bit) Setup

End-User License Agreement

Please read the following license agreement carefully

Server Side Public License
VERSION 1, OCTOBER 16, 2018

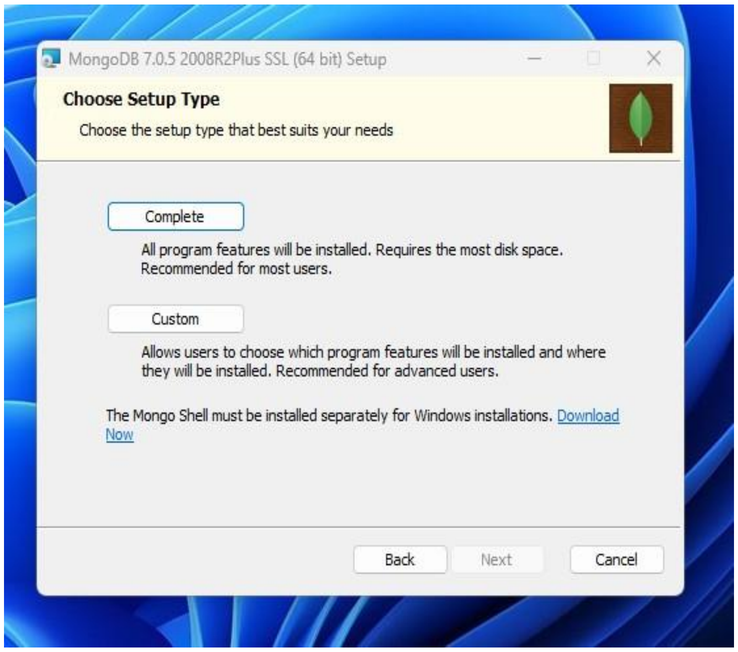
Copyright © 2018 MongoDB, Inc.

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

TERMS AND CONDITIONS

[✓] I accept the terms in the License Agreement

.
Check the box to accept the terms and conditions.


MongoDB 7.0.5 2008R2Plus SSL (64 bit) Setup

Choose Setup Type

Choose the setup type that best suits your needs

Complete

All program features will be installed. Requires the most disk space. Recommended for most users.

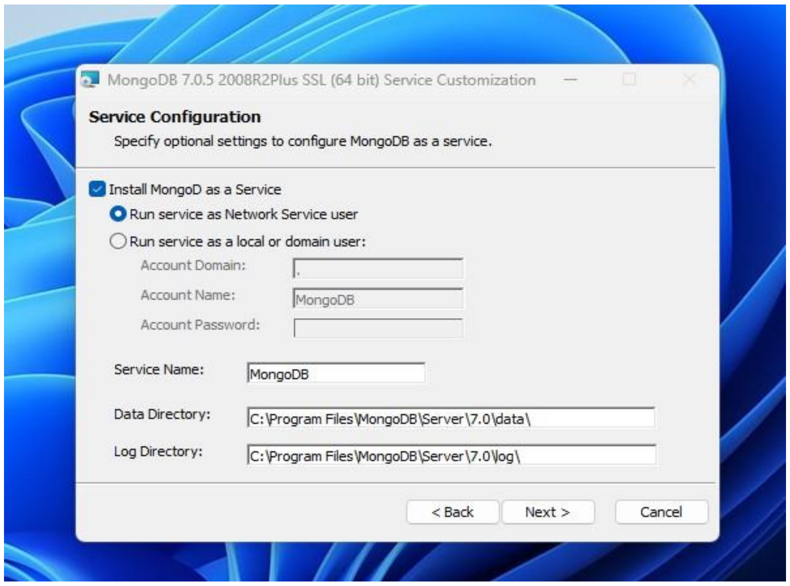
Custom

Allows users to choose which program features will be installed and where they will be installed. Recommended for advanced users.

The Mongo Shell must be installed separately for Windows installations. Download Now

Back Next Cancel

Select the **complete** setup type.


MongoDB 7.0.5 2008R2Plus SSL (64 bit) Service Customization

Service Configuration

Specify optional settings to configure MongoDB as a service.

[✓] Install MongoD as a Service

[● Run service as Network Service user]

○ Run service as a local or domain user:

Account Domain: .

Account Name: MongoDB

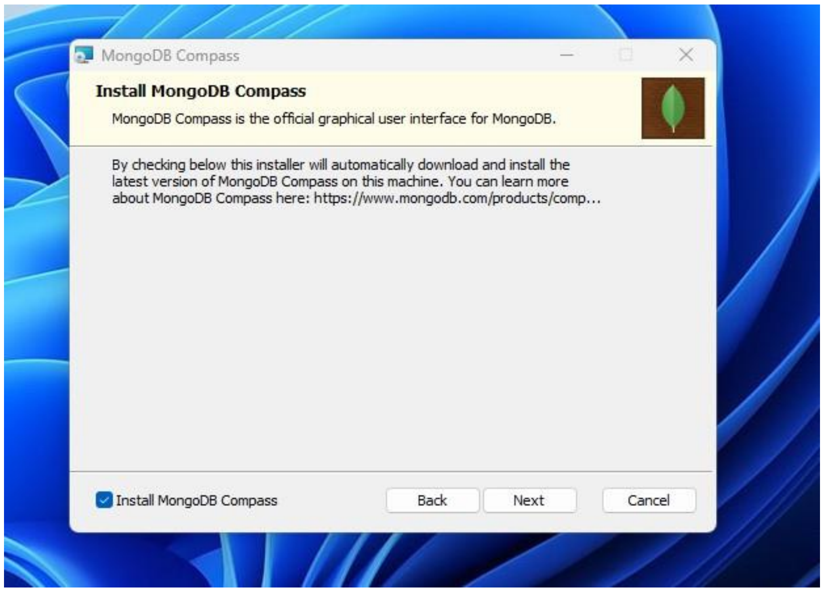
Account Password:

Service Name: MongoDB

Data Directory: C:\Program Files\MongoDB\Server\7.0\data\

Log Directory: C:\Program Files\MongoDB\Server\7.0\log\

< Back Next > Cancel


MongoDB Compass

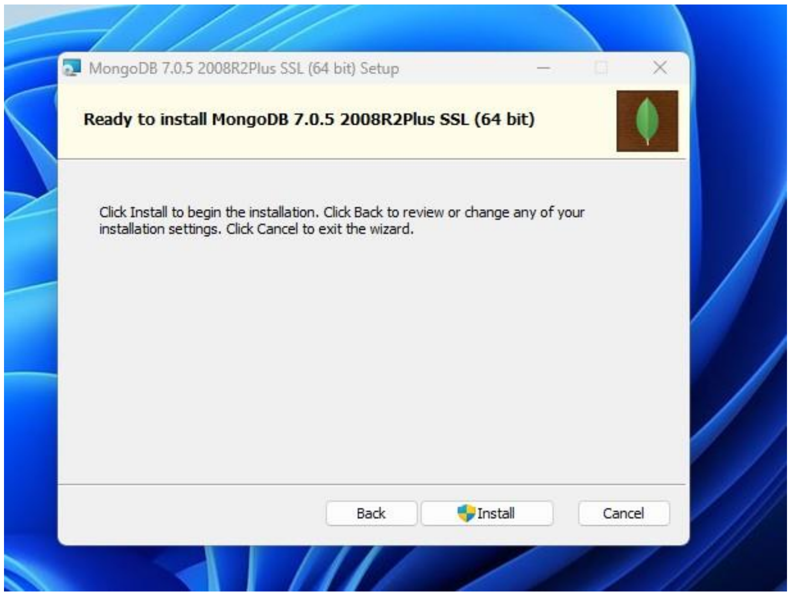
Install MongoDB Compass

MongoDB Compass is the official graphical user interface for MongoDB.

By checking below this installer will automatically download and install the latest version of MongoDB Compass on this machine. You can learn more about MongoDB Compass here: https://www.mongodb.com/products/comp...

☑ Install MongoDB Compass

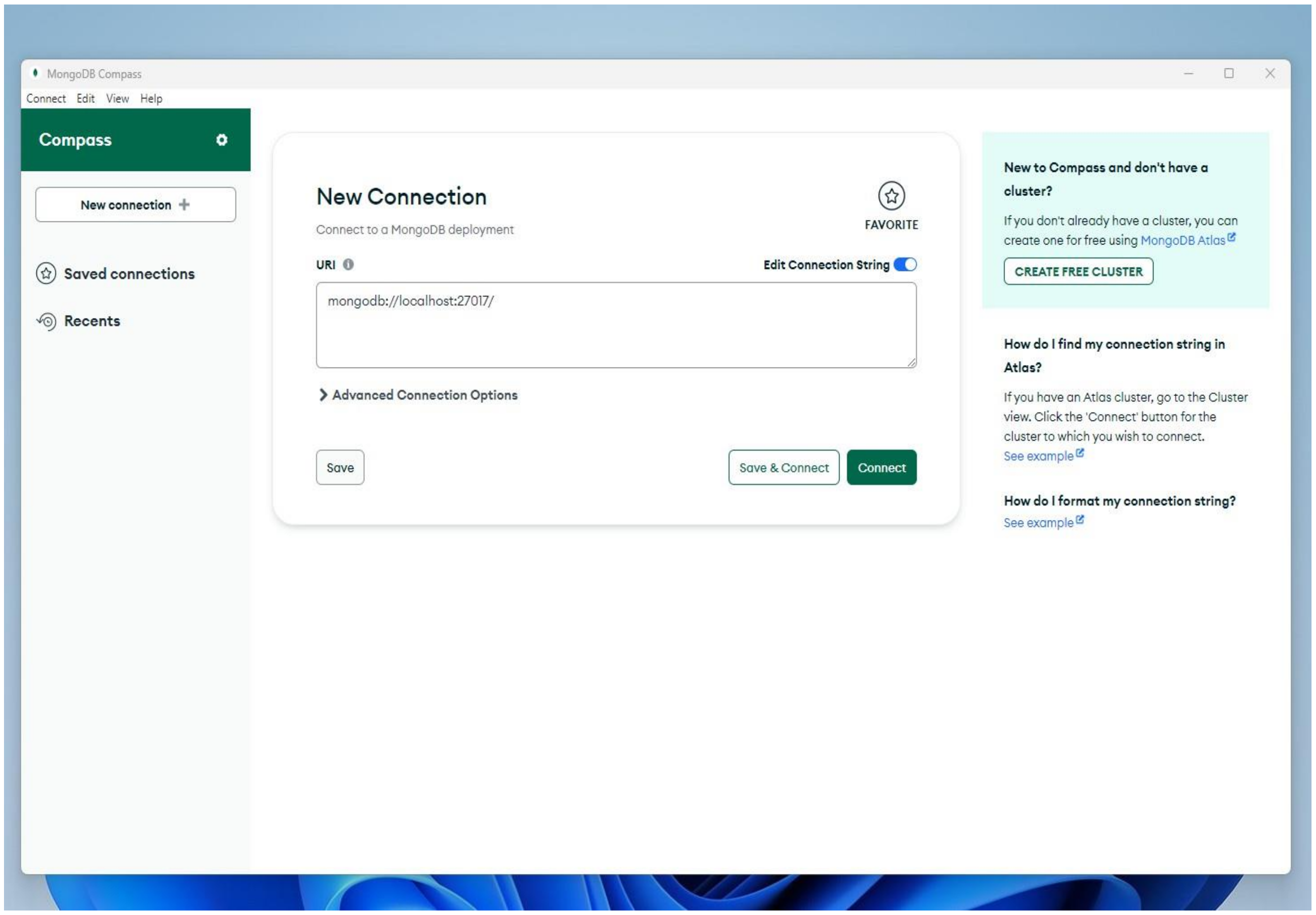
Check the box to install **mongoDB Compass** (UI for mongodb).


MongoDB 7.0.5 2008R2Plus SSL (64 bit) Setup

Ready to install MongoDB 7.0.5 2008R2Plus SSL (64 bit)

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.

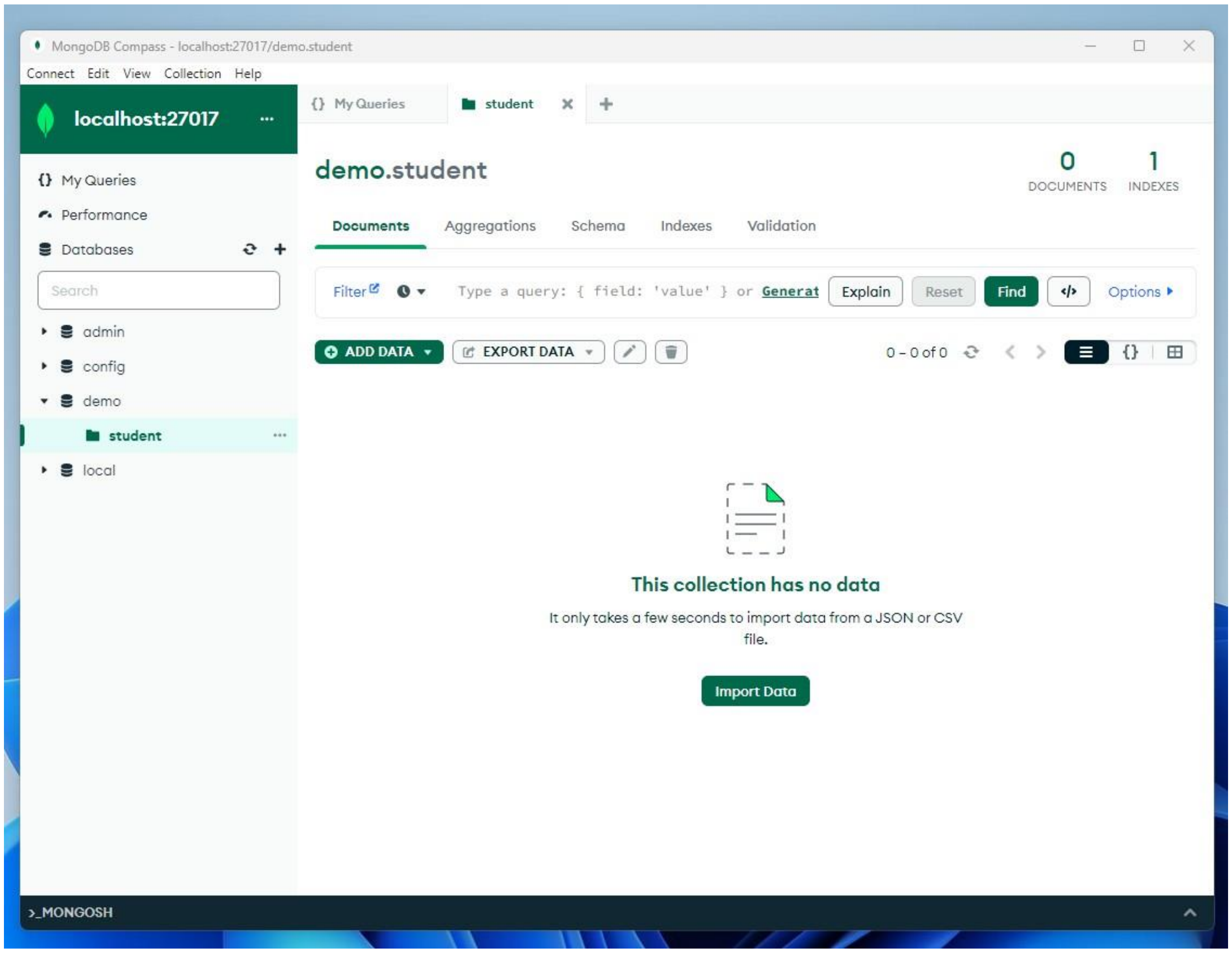
After successful installation, open MondoDB Compass. The figure below shows the interface of MongoDB Compass.


MongoDB Compass
Connect Edit View Help
Compass
New connection +
Saved connections
Recents
New Connection
Connect to a MongoDB deployment
FAVORITE
URI
Edit Connection String
> Advanced Connection Options
Save
Save & Connect
Connect
New to Compass and don't have a cluster?
If you don't already have a cluster, you can create one for free using MongoDB Atlas
CREATE FREE CLUSTER
How do I find my connection string in Atlas?
If you have an Atlas cluster, go to the Cluster view. Click the 'Connect' button for the cluster to which you wish to connect.
See example
How do I format my connection string?
See example

```
mongodb://localhost:27017/
```

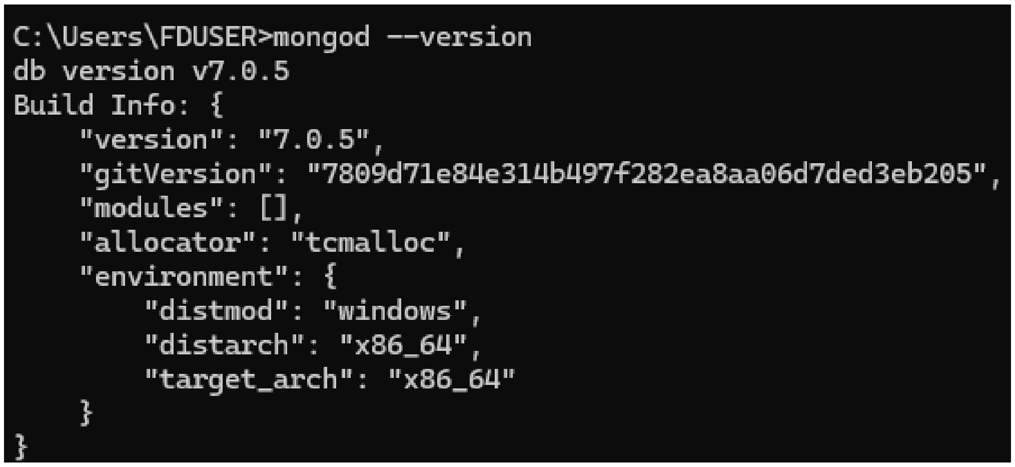
Click on connect, to connect to the local host.

Using the **+ icon** in the left panel, we have a new database named ‘demo’ and collection named ‘student’. Please try it yourself and see how it works.



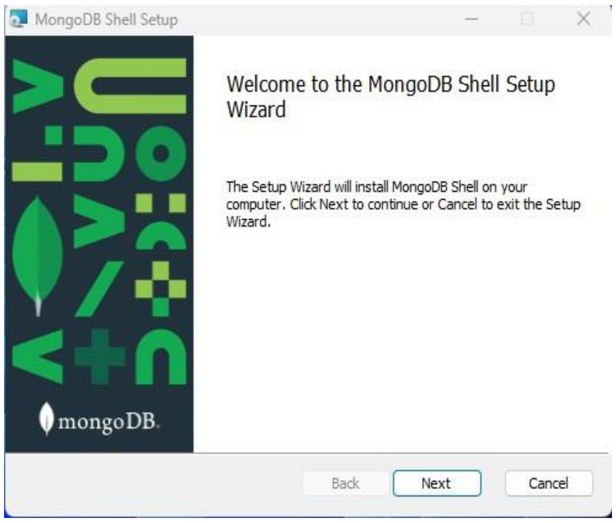
Now, you will have to edit the environment variable, by adding the path of mongodb compass. The path is as follows: C:\Program Files\MongoDB\Server\7.0\bin.

Now, open cmd and type **mongod –version,** to verify installation.



```
C:\Users\FDUSER>mongod --version
db version v7.0.5
Build Info: {
    "version": "7.0.5",
    "gitVersion": "7809d71e84e314b497f282ea8aa06d7ded3eb205",
    "modules": [],
    "allocator": "tcmalloc",
    "environment": {
        "distmod": "windows",
        "distarch": "x86_64",
        "target_arch": "x86_64"
    }
}
```

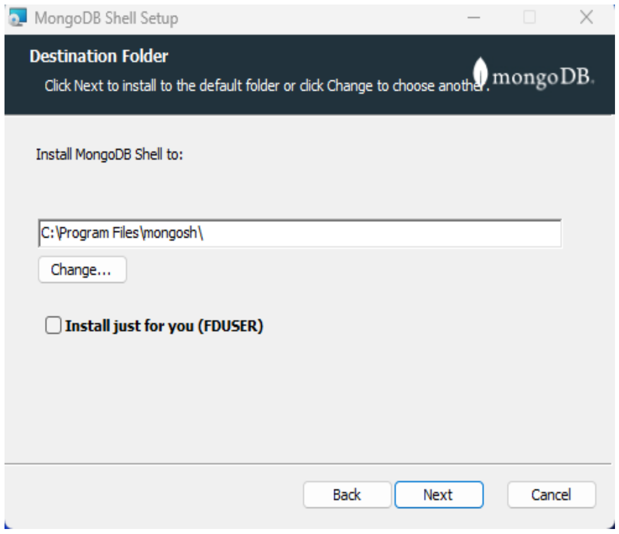
After this, let us install mongodb shell from https://www.mongodb.com/try/download/shell.



MongoDB Shell Setup

Welcome to the MongoDB Shell Setup Wizard

The Setup Wizard will install MongoDB Shell on your computer. Click Next to continue or Cancel to exit the Setup Wizard.



MongoDB Shell Setup

Destination Folder

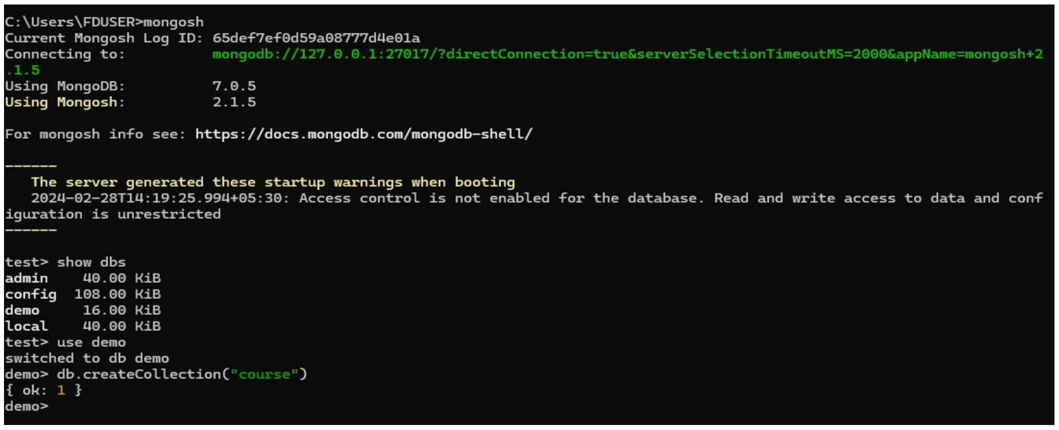
Click Next to install to the default folder or click Change to choose another.

Install MongoDB Shell to:

C:\Program Files\mongosh\

Install just for you (FDUSER)

After installation, go to cmd and type **'mongosh'.**
Type **show dbs** - this will show the existing databases
Similar to mysql, you may type **- use nameOfDatabase** — to use a particular database
To add a new collection, you may type, **db.createCollection("nameOfCollection")**



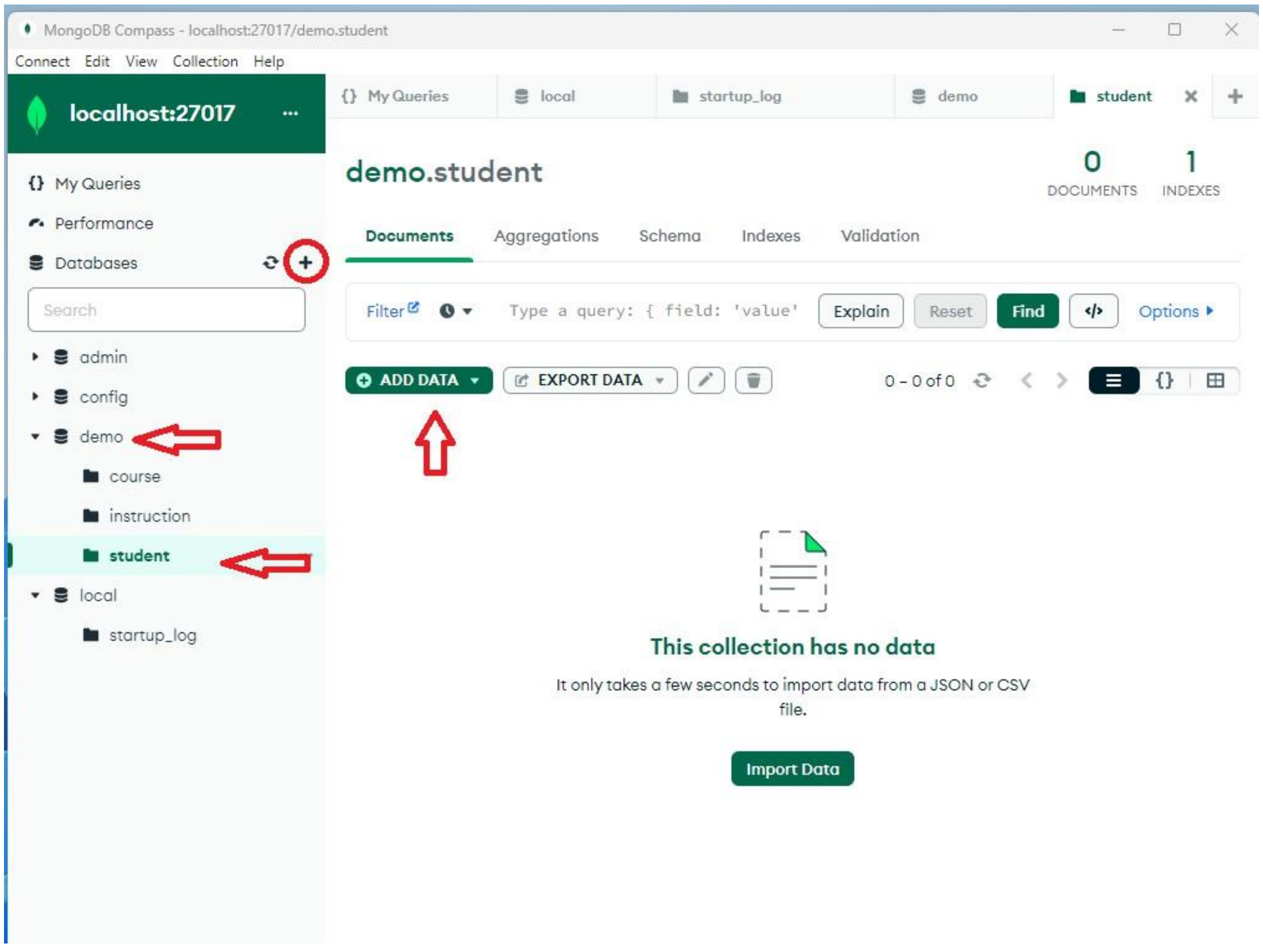
```
C:\Users\FDUSER>mongosh
Current Mongosh Log ID: 65def7ef0d59a08777d4e01a
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2
.1.5
Using MongoDB:          7.0.5
Using Mongosh:          2.1.5

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

------
   The server generated these startup warnings when booting
   2024-02-28T14:19:25.994+05:30: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
------

test> show dbs
admin    40.00 KiB
config  108.00 KiB
demo     16.00 KiB
local    40.00 KiB
test> use demo
switched to db demo
demo> db.createCollection("course")
{ ok: 1 }
demo>
```

First let us explore MongoDB Compass.



Using **+ icon**, you can add databases and corresponding collections.
We have added a database named lab9 with collections - post, comment and users.

3. Creating a Social Media Database

Objective

This section focuses on designing a schema for a simplified social media platform and implementing CRUD (Create, Read, Update, Delete) operations. By simulating the fundamental functionalities of social media—user profiles, posts, and comments—you'll gain practical experience with MongoDB's document model and operations.

Designing the Schema

Users Collection

The **users** collection will store user profiles, including basic information and an array of posts and comments made by the user. Here's an example schema:

```
{
 "_id": ObjectId("uniqueUserId"),
 "username": "user_name",
 "email": "user@example.com",
 "fullName": "Full Name",
 "bio": "Short user biography",
 "posts": [
 ObjectId("postId1"),
 ObjectId("postId2")
],
 "comments": [
   ObjectId("commentId1"),
   ObjectId("commentId2")
]
}
```

Posts Collection

The **posts** collection represents the messages or content users share. Each post is linked to a user and can have multiple comments.

```
{
 "_id": ObjectId("uniquePostId"),
 "user": ObjectId("userId"),
 "content": "This is the content of the post.",
 "timestamp": ISODate("2023-01-01T00:00:00Z"),
 "comments": [
   ObjectId("commentId1"),
   ObjectId("commentId2")
]
}
```

Comments Collection

Comments made on posts are stored in the **comments** collection. Each comment is associated with a user and a post.

```
{
 "_id": ObjectId("uniqueCommentId"),
 "post": ObjectId("postId"),
 "user": ObjectId("userId"),
 "content": "This is a comment.",
 "timestamp": ISODate("2023-01-02T00:00:00Z")
}
```

CRUD

CRUD operations represent the essential interactions with a database, encompassing the creation, reading, updating, and deletion of data. In the context of NoSQL databases, which can include document, key-value, wide-column, and graph databases, the specific syntax and operations can vary depending on the type of NoSQL database you're using. Here, I'll provide a brief overview focusing on MongoDB, a popular document-oriented NoSQL database, to illustrate CRUD operations with syntax examples.

Create (Insert Data)

To add new data to a collection:

Insert One Document

```
db.collectionName.insertOne({name: "Alice", age: 30});
```

Insert Multiple Documents

```
db.collectionName.insertMany([{name: "Bob", age: 25}, {name: "Charlie", age: 35}]);
```

Read (Query Data)

To retrieve data from a collection:

Find All Documents in a Collection

```
db.collectionName.find();
```

Find Documents Matching Criteria

```
db.collectionName.find({age: {$gt: 30}});
```

This finds documents where the age is greater than 30.

Projection (Selecting Specific Fields)

```
db.collectionName.find({}, {name: 1, _id: 0});
```

This returns only the name field of all documents.

Update (Modify Data)

To modify existing data in a collection:

Update One Document

```
db.collectionName.updateOne({name: "Alice"}, {$set: {age: 31}});
```

This updates the age of one document where the name is Alice.

Update Multiple Documents

```
db.collectionName.updateMany({age: {$gt: 30}}, {$inc: {age: 1}});
```

This increments the age by 1 for all documents where the age is greater than 30.

Delete (Remove Data)

To remove data from a collection:

Delete One Document

```
db.collectionName.deleteOne({name:  "Alice"});
```

Delete Multiple Documents

```
db.collectionName.deleteMany({age:  {$lt:  30}});
```

This deletes all documents where the age is less than 30.

Operation	Syntax	Example	RDBMS Equivalent
Equality	{<key>:<value>}	db.posts.find({"by":"tutorial"}).pretty()	where by = 'tutorials point'
Less Than	{<key>:{$lt:<value>}}	db.posts.find({"likes":{$lt:50}}).pretty()	where likes < 50
Less Than Equals	{<key>:{$lte:<value>}}	db.posts.find({"likes":{$lte:50}}).pretty()	where likes <= 50
Greater Than	{<key>:{$gt:<value>}}	db.posts.find({"likes":{$gt:50}}).pretty()	where likes > 50
Greater Than Equals	{<key>:{$gte:<value>}}	db.posts.find({"likes":{$gte:50}}).pretty()	where likes >= 50
Like	{<key>:{'$regex':<value>}}	db.posts.find({“title”: {'$regex': ‘How’} })	where title like ‘%How%’

Exercise Questions

NOTE: For all the questions, you may use a valid id from the existing collection

1. Retrieve comments made between February 1st, 2024 and February 28th, 2024.

2. Retrieve all posts made by a specific user by username. Hint: First, find the user's _id by their username, then query the posts collection using the user field.

3. Add a new comment. Hint: Insert a new document into the comments collection with the post field set to the post's _id.

4. Update a user's biography (bio).
   Hint: Use the updateOne method to update the bio field of the specified user's document.

5. Delete all comments made by a specific user from the comments collection, based on the user's _id