Getting Started with MongoDB Cloud Lab

a) **Set up a MongoDB Atlas cluster**

b) **Connect to the MongoDB Atlas cluster using a Node.js application**

c) **Perform basic CRUD (Create, Read, Update, Delete) operations**

MongoDB can be set up locally by downloading it from the MongoDB official website. Alternatively, you can use a cloud-based solution like MongoDB Atlas.

For a quick start, let's use MongoDB Atlas:

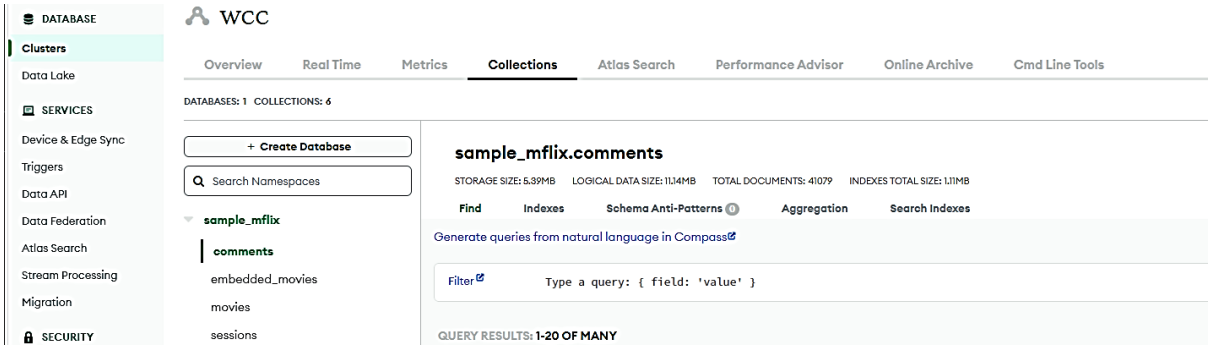
1. **Sign up** on MongoDB Atlas.

Create an Atlas Account

- Visit MongoDB Atlas and sign up for a free account.

2. **Create a cluster**

- Log in to your Atlas account
- Click on "Build a Cluster"
- Select the free tier (M0) and choose a cloud provider and region.
- Click "Create Cluster" and wait for the cluster to be created (this may take a few minutes)



3. **Create a database user** and whitelist your IP address

- Navigate to "Database Access"
- Click "Add New Database User"
- Create a user with a username and password. Note these credentials for later
- Set permissions to "Atlas Admin" or "Read and Write to any database"

4. **Whitelist Your IP Address**:

- Go to "Network Access"
- Click "Add IP Address"
- Add your current IP address or allow access from all IPs (use cautiously)

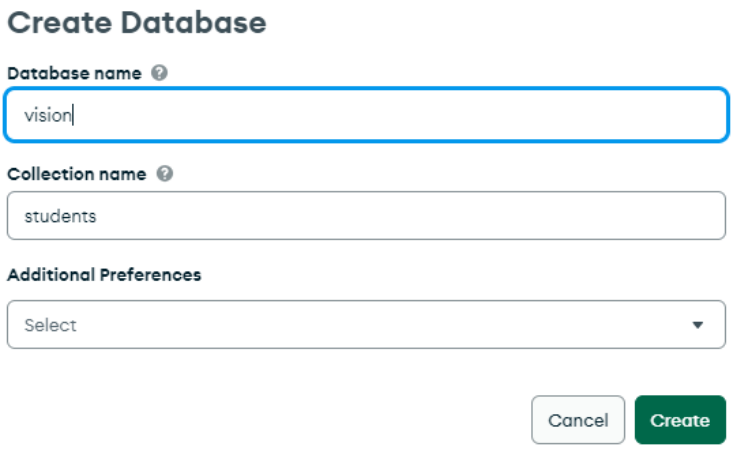
5. **Get the connection string**.

For Example:

mongodb+srv://
db_username>:<db_password>@wcc.r1yus.mongodb.net/?retryWrites=true&w=majority&appName=wcc

Note: Replace <db_password> with the password for the <db_username> database user.

6. **Create MongoDB Database**



a) Create a Node.js Application

Create a new directory for your project and navigate into it:

mkdir lab3

cd lab3

Initialize a new Node.js project:

npm init -y

Install the necessary packages:

npm install express-generator -g

npx express-generator mdb

Mongoose: An ODM (Object Data Modelling) library for MongoDB and Node.js

npm install express mongoose

Body-Parser: Middleware to parse incoming request bodies

npm install express mongoose body-parser

By adding the following code to app.js file after you declare the express application.

```
// Set up mongoose connection

const mongoose = require("mongoose");

mongoose.set("strictQuery", false);

const mongoDB = "mongodb+srv://whitecliffe-mc
Ky4XskoddDCaKlGb@cluster0.duazcta.mongodb.net/?retryWrites=true&w=majority ";



main().catch((err) => console.log(err));

async function main() {

  await mongoose.connect(mongoDB);

}
```

Run the application by using the command **node app.js** in terminal and place screenshot below, a sample screenshot is there:

```
PS C:\Users\[redacted] > node app.js
Congrates! Database is connected
```

Create, View, Update, and Delete Documents

Use the Atlas UI to manage documents inside your collections. Documents are individual records in a MongoDB collection and are the basic unit of data in MongoDB.

Viewing documents and collections in the Atlas UI can provide a high-level overview of database schema.

Click ***Insert Document***.

The document editor appears with the _id field with an ObjectId value that reflects the time of its generation and not the insertion time of the document. As such, the ObjectId does not represent a strict insertion order.

Modify the document.

- To add a new field after an existing field, hover over the field and click on the plus sign that appears over the field's line number.
- To delete a field, hover over the field and click on the x sign that appears to the left of the field's line number. You cannot delete the _id field.
- To edit a field name, value, or type, click on the field name, value, or type.

Click ***Insert***.

Data can also be imported with a Json format because we are going to insert it in MongoDB database. Use the provided file to import data it.