

Estimated time needed: **30** minutes

**Objectives**

After completing this lab you will be able to:

* Start and access the MongoDB server using the command-line interface
* Describe the process of listing and creating collections
* Perform basic operations on a collection such as inserting, counting, and listing documents
* Connect to the MongoDB server from Node and perform basic queries

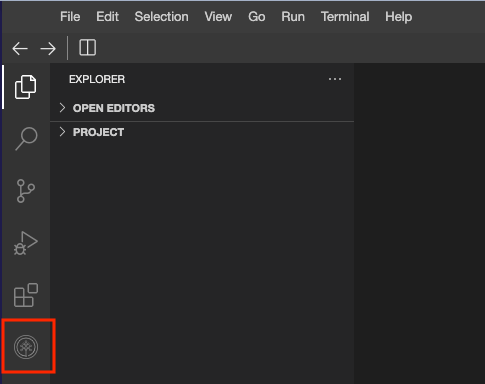
**About Skills Network Cloud IDE**

Skills Network Cloud IDE (based on Theia and Docker) provides an environment for hands on labs for course and project related labs. Theia is an open source IDE (Integrated Development Environment).

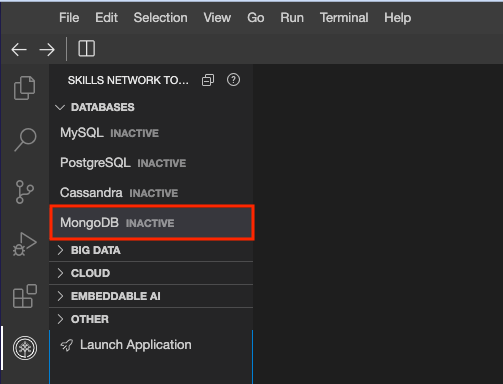
**Important Notice about this lab environment**

Please be aware that sessions for this lab environment are not persisted. Every time you connect to this lab, a new environment is created for you. Any data you may have saved in the earlier session would get lost. Plan to complete these labs in a single session, to avoid losing your data

**Start MongoDB server**



You will notice MongoDB listed there, but inactive. Which means the database is not available to use.



Once you click on it, you will see more details about it and a button to start it.

A screenshot of a computer

Description automatically generated

Clicking on the start button will run a background process to configure and start your MongoDB server.

A screenshot of a computer

Description automatically generated

Shortly after that, your server is ready for use. This deployment has access control enabled and MongoDB enforces authentication. So, **take note of the password** as you will need it to login as root user.

A screenshot of a computer

Description automatically generated

**Note:** For Password and other information click on Connection Information

A screenshot of a computer

Description automatically generated

You can simply click on MongoDB CLI which does that for you.

A screenshot of a computer

Description automatically generated

This will open a terminal with the Mongosh.

**Working with Mongo CLI**

1. On the Mongo client run the below command to check the version of the MongoDB Server.
2. db.version()

This will show the version of the mongodb server.

1. On the Mongo client, run the command below to get the list of databases on the server.
2. show dbs

This will print a list of the databases present on the server, including default and user-defined.

1. On the Mongo client run the command below to create a database.
2. use employeeDB

This will switch the context to the database named employeeDB. If the database employeeDB doesn't exist, MongoDB will create it for you. But that happens only when you create a collection inside the database. MongoDB creates databases and collections lazily i.e., it is created only when necessary.

1. On the Mongo client run the command below to create a collection named employee inside the employeeDB.
2. db.createCollection("employees")

This will create a collection name employees inside the employeeDB database.

1. On the Mongo client run the command below to list the collections.
2. show collections

This will print the list of collections in your current database.

**Data insertion**

1. On the Mongo client run the command below to insert data into the employee collection. Each data point is referred to as a document in MongoDB.
2. db.employees.insert({"emp\_name":"Ray Renolds","Department":"IT", "salary":7500, "onsite":false})

The above command inserts the JSON document {"emp\_name":"Ray Renolds","Department":"IT", "salary":7500, "onsite":false} into the collection.

1. Insert one more documents by running the command below.
2. db.employees.insert({"emp\_name":"Matt Aniston","Department":"HR", "salary":6000, "onsite":true})
3. To insert multiple documents, you can enclose them in square brackets. Run the command below to insert two documents at a time.
4. db.employees.insert([{"emp\_name":"Monica Perry","Department":"Admin", "salary":5000, "onsite":true},{"emp\_name":"Rachel Tribbiani","Department":"IT", "salary":9000, "onsite":false}])
5. On the Mongo client run the command below to count the number of documents.
6. db.employees.countDocuments()

This command gives you the number of documents in the collection.

1. On the Mongo client run the command below to list all documents in the employee collection.
2. db.employees.find()

This command lists all the documents in the collection employees

Notice that MongoDB automatically adds an \_id field to every document in order to uniquely identify the document.

1. On the Mongo client run the command below to disconnect from the MongoDB server.
2. exit

**Connect to Mongo from Node**

1. Open a New Terminal to work with.

A screenshot of a computer

Description automatically generated

1. Run the following command to create a package.json file, which will save all the dependencies. Give the package the name, empMgmtor any other name of your choice. Press enter to accept the default values and complete the initialization.
2. npm init

*This lab environment is transient. You will lose the directories and files you are creating after you log out of the lab session. But it is always a good practice to create package.json when you are actually working on projects to keep track of the dependencies installed. It will be helpful when you work on real world projects.*

1. In the terminal, run the following command to install the Node package named mongoose for connnecting to MongoDB server.
2. npm install mongoose
3. Run the following command to create a new file named employee.js in which we will define the schema for the employees collection. Node needs this to interpret the data it reads from the MongoDB.
4. touch employee.js
5. Paste the following content that defines the schema inside employee.js and save it.
6. const mongoose = require('mongoose');
7. const Schema = mongoose.Schema;
8. const employeesSchema = new Schema({
9. emp\_name: {
10. type: String,
11. required: true
12. },
13. Department: {
14. type: String,
15. required: true,
16. },
17. salary: {
18. type: Number,
19. required: true
20. },
21. onsite: {
22. type: Boolean,
23. default: false
24. }
25. });
26. module.exports = mongoose.model('employees', employeesSchema);

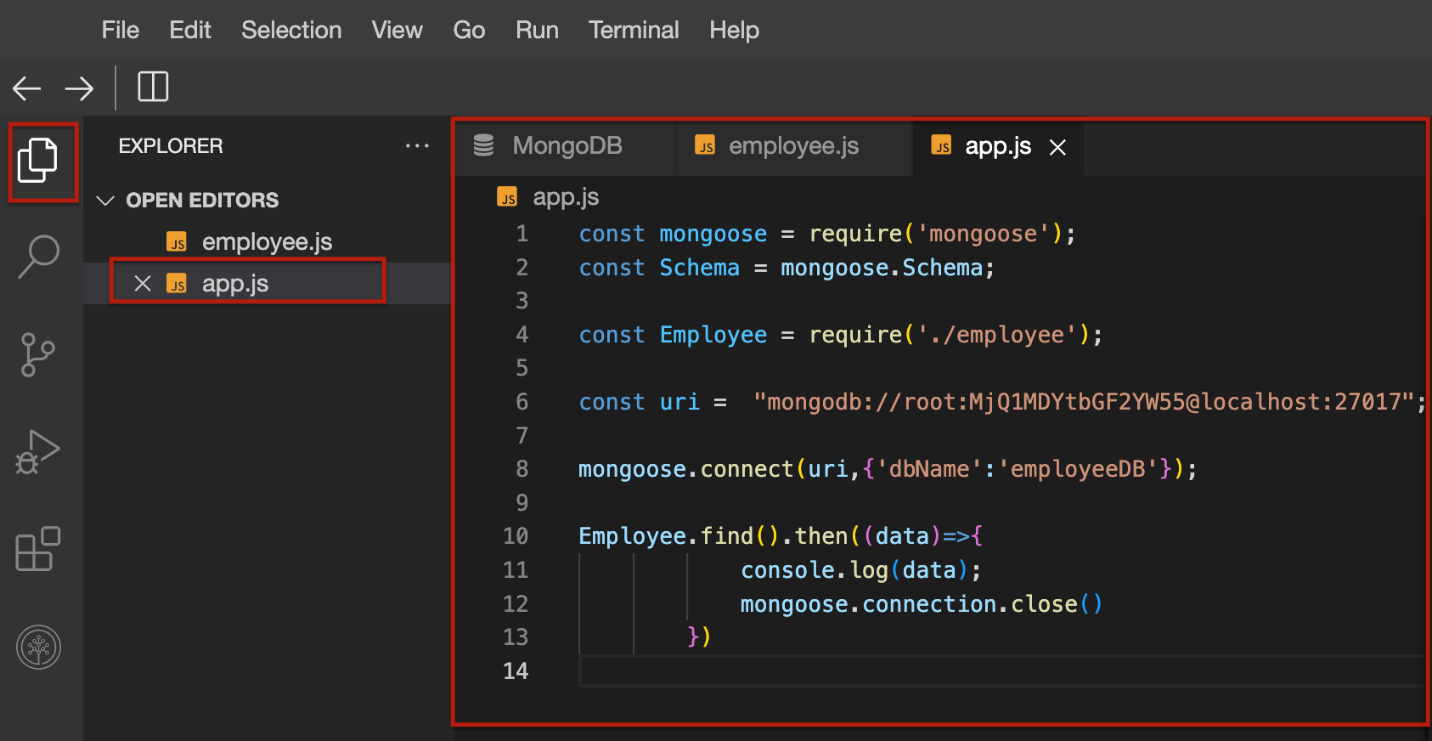
A screenshot of a computer

Description automatically generated

1. Run the following command to create a new file named app.js. This is the application that will connect to the MongoDB and retrieve the employees collection.
2. touch app.js
3. Open the file app.js in the editor and paste the following content. This will connect to the MongoDB server that is running from you node program and display all the records in the employees collection, when run.

*Note:  
-Make sure you replace the password with the password allocated by your instance of MongoDB server.  
-Placing the password in a URL is not a best practice. It has been used here for lab purposes only.*

1. const mongoose = require('mongoose');
2. const Employee = require('./employee');
3. const uri = "mongodb://root:MjQ1MDYtbGF2YW55@localhost:27017";
4. mongoose.connect(uri,{'dbName':'employeeDB'});
5. Employee.find().then((data)=>{
6. console.log(data);
7. mongoose.connection.close()
8. })



1. Run app.js from the terminal.
2. node app.js



You will see that all the records that you created earlier through the Mongosh CLI, will be retrieved.

**Summary**

In this lab you started the MongoDB server, connected to it from Mongosh, created a collection with data, connected to the MongoDB from Node and retrieved all the documents in the collection you created.

Congratulations! You have successfully connected to the MongoDB server from Node.js.

**Authors**

Lavanya T S

Rama