

**Estimated time needed:** 30 minutes

**Learning Objectives**

After completing this lab you will be able to:

* Access the MongoDB server using Node.
* Create a collection in MongoDB using Node.
* Create, retrieve, update and delete documents in a collection using Node.

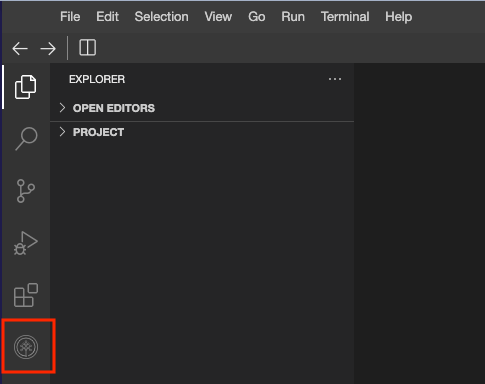
**About Skills Network Cloud IDE**

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).

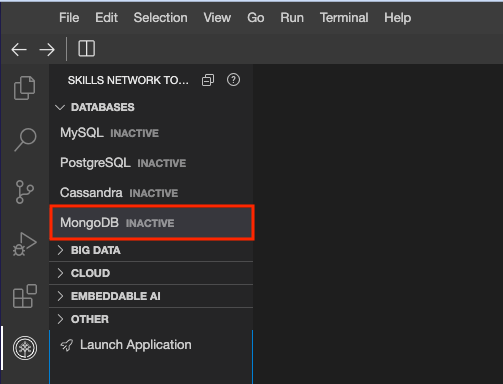
**Important Notice about this lab environment**

Please be aware that sessions for this lab environment do not persist. 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.

**Exercise 1 - 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 Create 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

**Connecting to MongoDB from Node**

1. Open a new terminal.

A screenshot of a computer

Description automatically generated

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. Open the file for editing and paste the following content in the file.
6. const mongoose = require('mongoose');
7. const Schema = mongoose.Schema;
8. const employees = new Schema({
9. emp\_name: {
10. type: String,
11. required: true
12. },
13. age: {
14. type: Number,
15. required: true,
16. },
17. location: {
18. type: String,
19. required: true
20. },
21. email: {
22. type: String,
23. required: true
24. }
25. });
26. module.exports = mongoose.model('employees', employees);
27. Run the following command to create a new file named app\_list.js. This is the application that will connect to the MongoDB and retrieve the employees collection and list it on the console.
28. touch app\_list.js
29. Open the file app\_list.js in the editor and paste the following content. This will connect to the MongoDB server that is running from the Node program and display all the records in the employees collection, when run.

*Make sure you replace the password with the password allocated by your instance of MongoDB server.*

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. })
9. Run the app\_list.js from the terminal.
10. node app\_list.js

You will see that there are no records in the collection because you haven't yet created any.

**InsertOne Document**

1. Run the following command to create a new file named app\_insertOne.js. This is the application that will connect to the MongoDB and add one document to the employees collection.
2. touch app\_insertOne.js
3. Open the file app\_insertOne.js in the editor and paste the following content.

*Make sure you replace the password with the password allocated by your instance of MongoDB server.*

1. const mongoose = require('mongoose');
2. const Schema = mongoose.Schema;
3. const Employee = require('./employee');
4. const uri = "mongodb://root:MjQ1MDYtbGF2YW55@localhost:27017";
5. mongoose.connect(uri,{'dbName':'employeeDB'});
6. //insertOne record into employee
7. let newEmployee = new Employee({
8. emp\_name: 'John Doe',
9. age: 37,
10. location: "Illinois",
11. email: "jdoe@somewhere.com"
12. });
13. newEmployee.save().then(function(){
14. Employee.find().then((data)=>{
15. console.log("\n\nDocuments in employees collection after insertOne")
16. console.log(data);
17. mongoose.connection.close();
18. })
19. }).catch(function(error){
20. console.log(error)
21. });
22. Run the app\_insertOne.js from the terminal.
23. node app\_insertOne.js

This should insert one document in the collection.

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

You will see that there is one document in the collection.

**InsertMany Documents**

1. Run the following command to create a new file named app\_insertMany.js. This is the application that will connect to the MongoDB and add many documents to the employees collection at once.
2. touch app\_insertMany.js
3. Open the file app\_insertMany.js in the editor and paste the following content.

*Make sure you replace the password with the password allocated by your instance of MongoDB server.*

1. const mongoose = require('mongoose');
2. const Schema = mongoose.Schema;
3. const Employee = require('./employee');
4. const uri = "mongodb://root:Mjc5MjktbGF2YW55@localhost:27017";
5. mongoose.connect(uri,{'dbName':'employeeDB'})
6. .then(() => {
7. console.log("Connected to MongoDB");
8. // insertMany records into employee
9. return Employee.insertMany([
10. { "emp\_name": "Ray Renolds", "age": 32, "location": "Austin", "email": "rayr@somewhere.com" },
11. { "emp\_name": "Matt Aniston", "age": 25, "location": "Houston", "email": "matta@somewhere.com" },
12. { "emp\_name": "Monica Perry", "age": 23, "location": "New Jersey", "email": "monicap@somewhere.com" },
13. { "emp\_name": "Rachel Tribbiani", "age": 28, "location": "Boston", "email": "rachelt@somewhere.com" }
14. ]);
15. })
16. .then(() => {
17. console.log("Records inserted successfully");
18. // Find all documents in employees collection after insertMany
19. return Employee.find();
20. })
21. .then((data) => {
22. console.log("\nDocuments in employees collection after insertMany:");
23. console.log(data);
24. })
25. .catch((error) => {
26. console.error("Error:", error);
27. })
28. .finally(() => {
29. mongoose.connection.close(); // Close the MongoDB connection
30. });
31. Run the app\_insertMany.js from the terminal.
32. node app\_insertMany.js

This should insert multiple documents in the collection.

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

You will see that there many documents in the collection.

**Update documents**

1. Run the following command to create a new file named app\_update.js. This is the application that will connect to the MongoDB and update the documents in the employees collection.
2. touch app\_update.js
3. Open the file app\_update.js in the editor and paste the following content.

*Make sure you replace the password with the password allocated by your instance of MongoDB server.*

1. const mongoose = require('mongoose');
2. const Schema = mongoose.Schema;
3. const Employee = require('./employee');
4. const uri = "mongodb://root:Mjc5MjktbGF2YW55@localhost:27017";
5. mongoose.connect(uri,{'dbName':'employeeDB'})
6. .then(() => {
7. console.log("Connected to MongoDB");
8. // Update one record in employee
9. return Employee.updateOne({ emp\_name: "John Doe" },
10. { email: "jdoe@somewhere.com" });
11. })
12. .then((updateOneResult) => {
13. console.log("Updated Docs for updateOne:", updateOneResult);
14. console.log("One record updated");
15. // Update many records in employees
16. return Employee.updateMany({ age: { $gt: 30 } },
17. { location: "New York" });
18. })
19. .then((updateManyResult) => {
20. console.log("Updated Docs for updateMany:", updateManyResult);
21. console.log("Many records updated");
22. })
23. .catch((error) => {
24. console.error("Error:", error);
25. })
26. .finally(() => {
27. mongoose.connection.close(); // Close the MongoDB connection
28. });
29. Run the app\_update.js from the terminal.
30. node app\_update.js

This should update the documents in the collection.

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

You will see that the documents are updated in the collection.

**Delete Documents**

1. Run the following command to create a new file named app\_delete.js. This is the application that will connect to the MongoDB and delete the documents from the employees collection.
2. touch app\_delete.js
3. Open the file app\_delete.js in the editor and paste the following content.

*Make sure you replace the password with the password allocated by your instance of MongoDB server.*

1. const mongoose = require('mongoose');
2. const Schema = mongoose.Schema;
3. const Employee = require('./employee');
4. const uri = "mongodb://root:Mjc5MjktbGF2YW55@localhost:27017";
5. mongoose.connect(uri,{'dbName':'employeeDB'})
6. .then(() => {
7. console.log("Connected to MongoDB");
8. // Delete one record from employees
9. return Employee.deleteOne({ age: { $lt: 30 }, location: "New York" });
10. })
11. .then((deleteOneResult) => {
12. console.log("Deleted document for deleteOne:", deleteOneResult);
13. // Delete many records from employees
14. return Employee.deleteMany({ emp\_name: { $regex: "R" } });
15. })
16. .then((deleteManyResult) => {
17. console.log("Deleted documents for deleteMany:", deleteManyResult);
18. })
19. .catch((error) => {
20. console.error("Error:", error);
21. })
22. .finally(() => {
23. mongoose.connection.close(); // Close the MongoDB connection
24. });
25. Run the app\_delete.js from the terminal.
26. node app\_delete.js

This should delete the documents from the collection.

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

You will see that the documents are deleted from the collection.

Congratulations! You have successfully connected to MongoDB server from Node.js and manipulated data.

**Authors**

Lavanya T S

**Other Contributors**

Rama