MongoDB CRUD

Estimated time needed: **30** minutes

Objectives

After completing this lab, you will be able to:

- Create documents in MongoDB with the insert method
- Read documents by listing them, counting them and matching them to a query
- Update and delete documents in MongoDB based on specific criteria

About Skills Network Cloud IDE

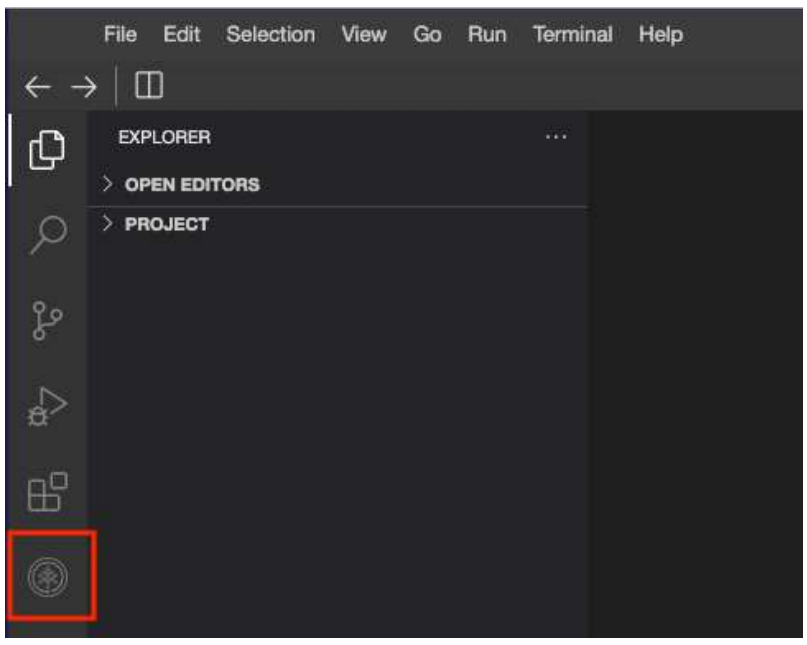
Skills Network Cloud IDE (based on Theia and Docker) provides an environment for hands on labs for course and project related labs. Theia is an open source IDE (Integrated Development Environment), that can be run on desktop or on the cloud. To complete this lab, we will be using the Cloud IDE based on Theia and MongoDB running in a Docker container.

Important Notice about this lab environment

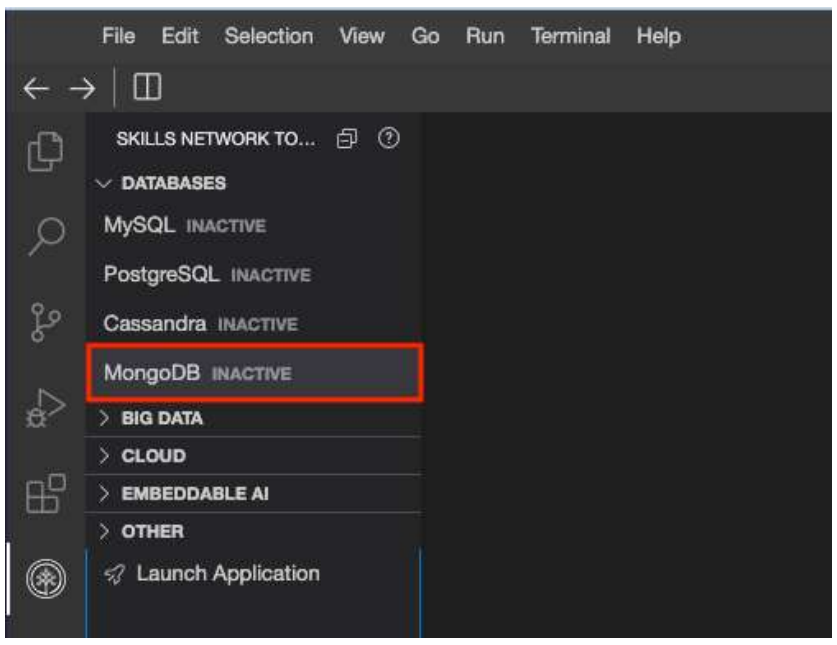
Please be aware that sessions for this lab environment are not persisted. Every time you connect to this lab, a new environment is created for you. Any data you may have saved in the earlier session would get lost. Plan to complete these labs in a single session, to avoid losing your data.

Set-up: Start MongoDB

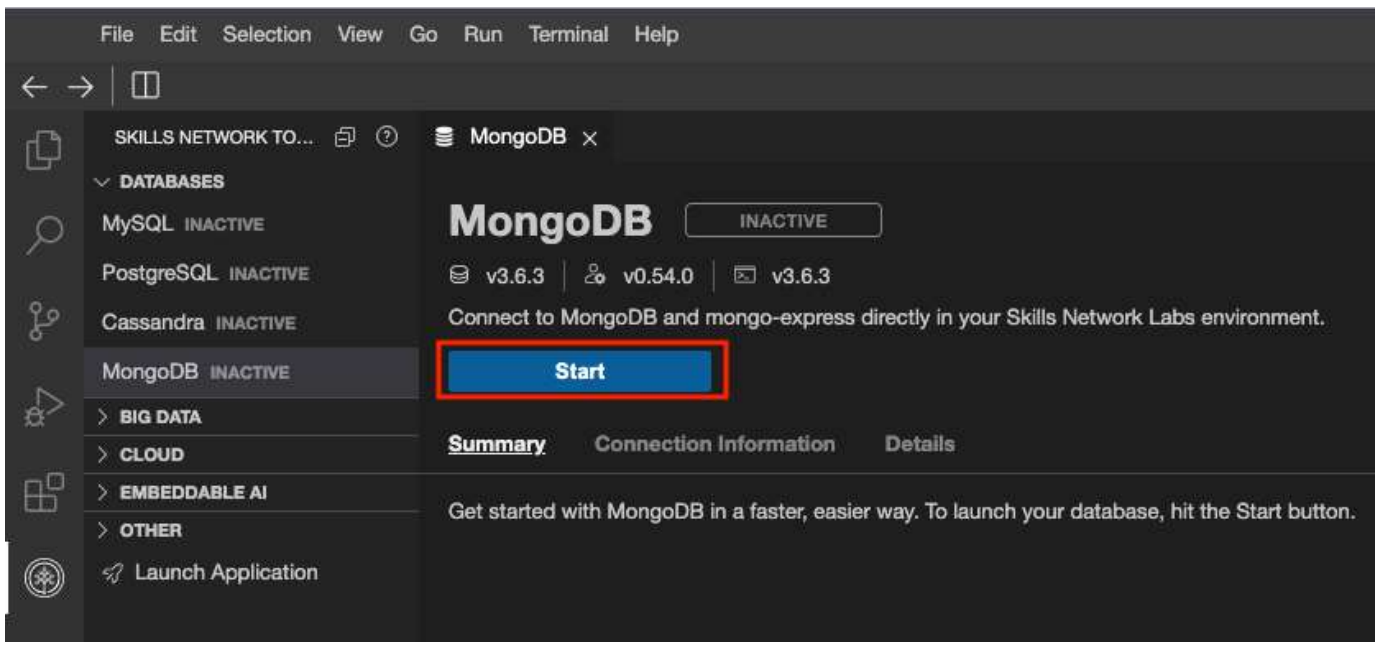
Navigate to Skills Network Toolbox.



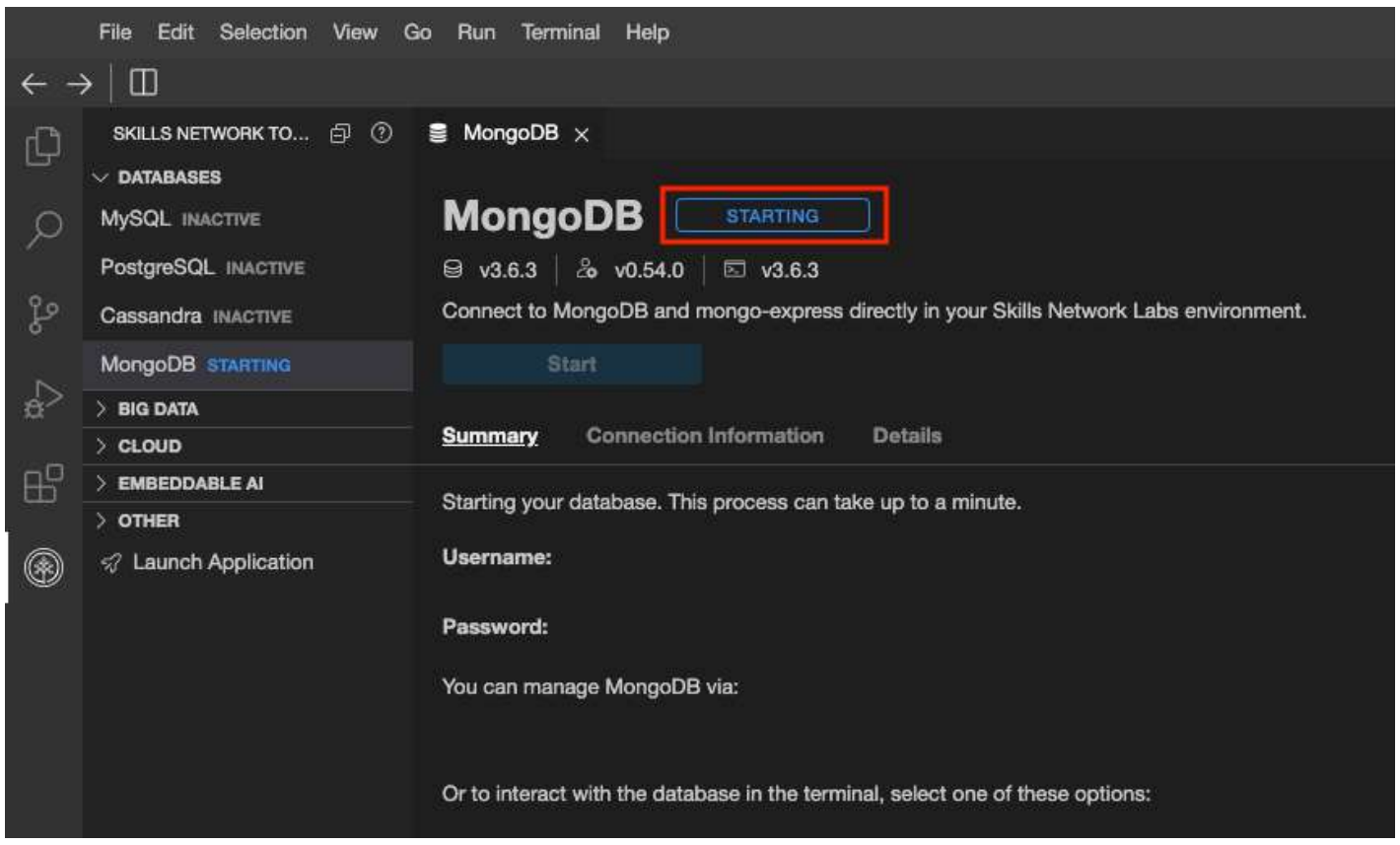
You will notice MongoDB listed there, but inactive. Which means the database is not available to use.



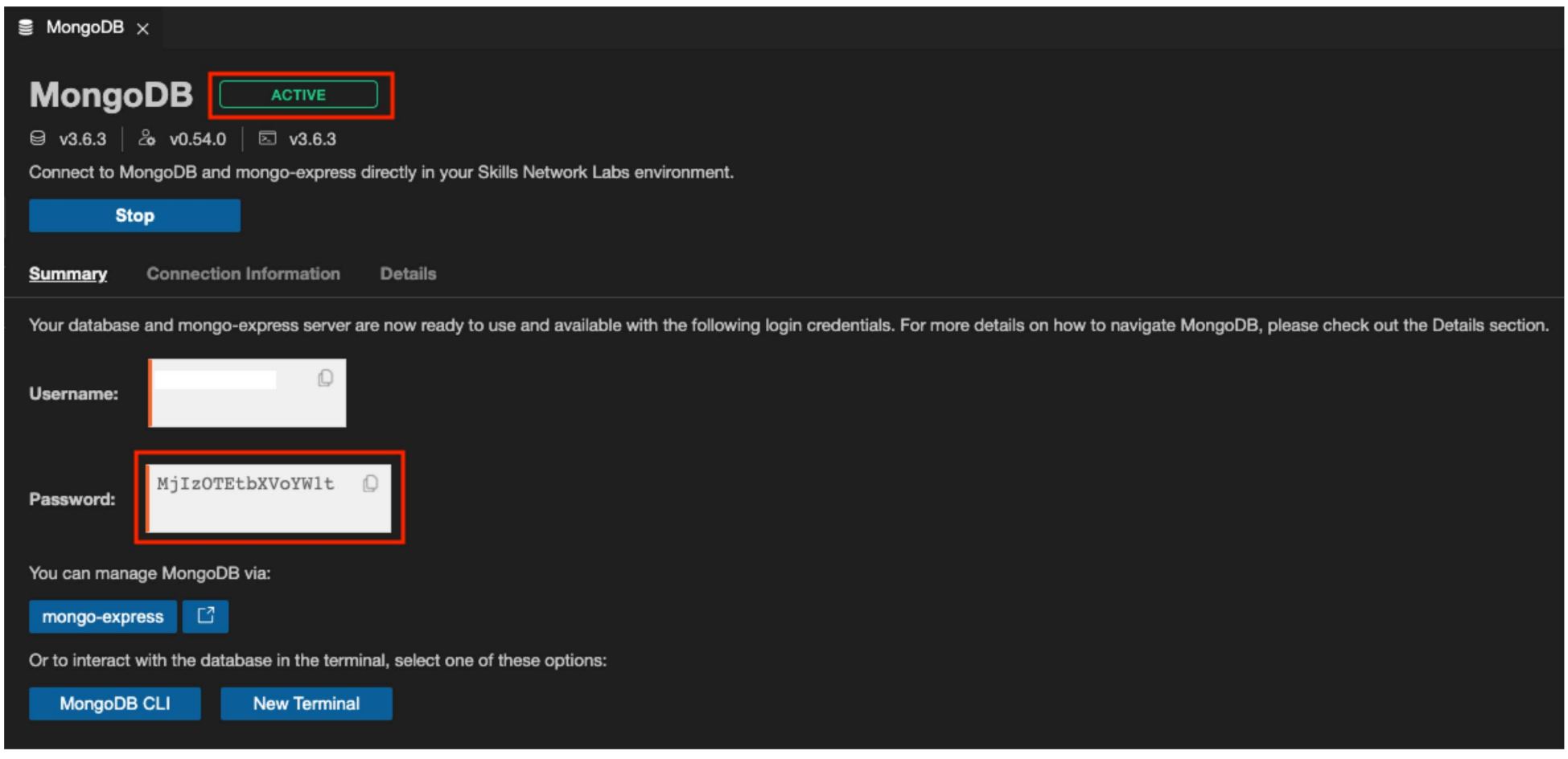
Once you click on it, you will see more details about it and a button to start it.



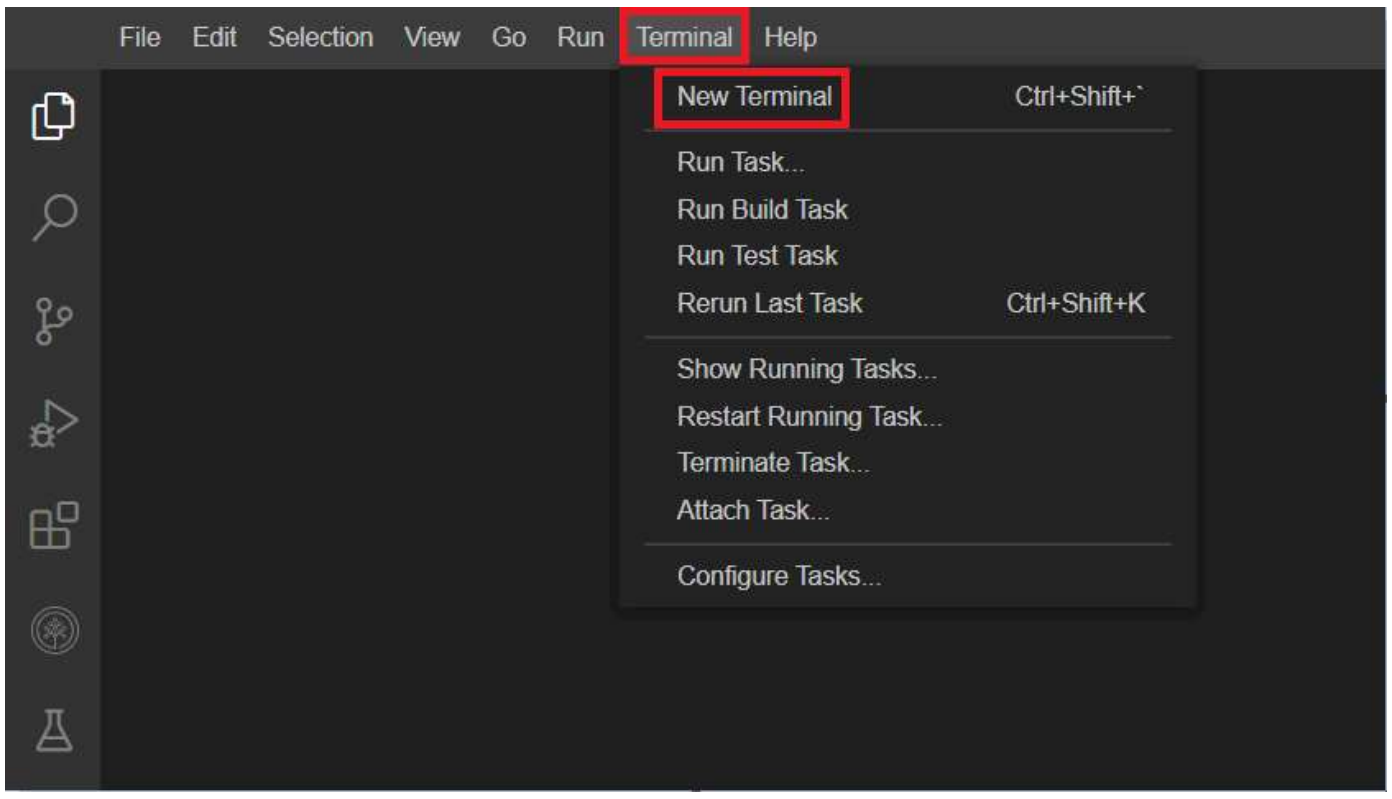
Clicking on the start button will run a background process to configure and start your MongoDB server.



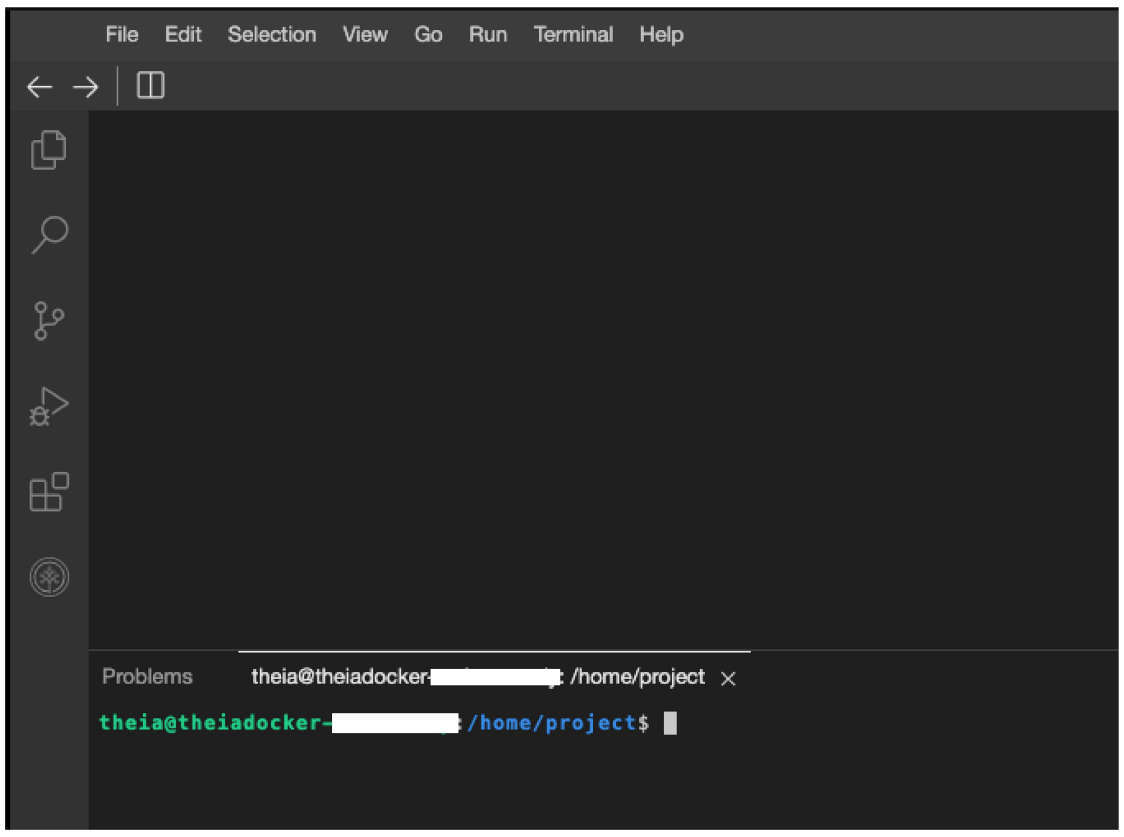
Shortly after that, your server is ready for use. This deployment has access control enabled and MongoDB enforces authentication. So, take note of the password as you will need it to login as `root` user.



You can now open terminal and enter details yourself.



This will open a new terminal at the bottom of the screen as in the image below.



Run the below command on the newly opened terminal. (You can copy the code by clicking on the little copy button on the bottom right of the codeblock below and then paste it, wherever you wish.)

```
1. 1
```

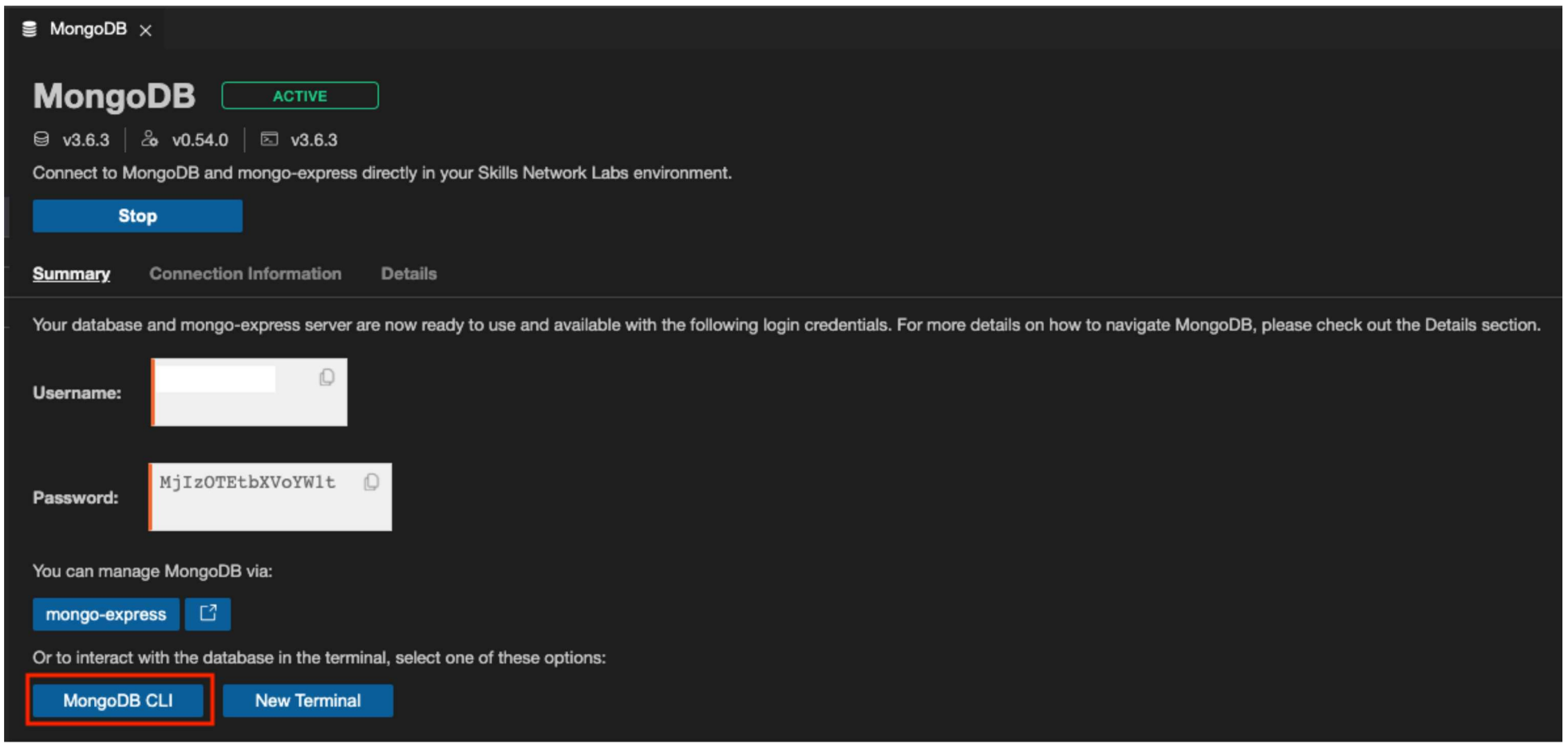
```
1. mongosh -u root -p PASSWORD --authenticationDatabase admin local
```

Copied! Executed!

```
theia@theiadocker- /home/project$ mongosh -u root -p MTc3MDUtbXVoYW1t --authenticationDatabase admin
 local
Current Mongosh Log ID: 646f9447f39eb3e6e51c6363
Connecting to:          mongodb://<credentials>@127.0.0.1:27017/local?directConnection=true&serverSelectionTi
meoutMS=2000&authSource=admin&appName=mongosh+1.8.0
Using MongoDB:          3.6.3
Using Mongosh:          1.8.0

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

------
   The server generated these startup warnings when booting
   2023-05-25T16:50:00.585+0000:
   2023-05-25T16:50:00.585+0000: ** WARNING: Using the XFS filesystem is strongly recommended with the WiredT
iger storage engine
   2023-05-25T16:50:00.585+0000: **          See http://dochub.mongodb.org/core/prodnotes-filesystem
   2023-05-25T16:50:01.480+0000:
   2023-05-25T16:50:01.480+0000: ** WARNING: You are running on a NUMA machine.
   2023-05-25T16:50:01.480+0000: **          We suggest launching mongod like this to avoid performance probl
ems:
   2023-05-25T16:50:01.480+0000: **              numactl --interleave=all mongod [other options]
   2023-05-25T16:50:01.480+0000:
------

local>
```

The command contains the username and password to connect to mongodb server (the text after the **-p** option is the password). Your output would be different from the one shown above. Copy the command given to you, and keep it handy. You will need it in the next step.

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



Exercise 1 - Getting the database and collection ready

Select the training database

*Select the **training** database.*

▼ Click here for Hint

`use` command helps you switch context to a particular database.

▼ Click here for Solution

```
1. 1
1. use training
```

Copied!

Create languages collection

*Create a collection named **languages**.*

▼ Click here for Hint

`createCollection` command lets you create a collection.

▼ Click here for Solution

```
1. 1
1. db.createCollection("languages")
```

Copied!

Exercise 2 - Create documents

1. Run the below commands in mongo client to insert two documents into the collection `languages` one at a time.

```
1. 1
2. 2
1. db.languages.insertOne({"name":"java","type":"object oriented"})
2. db.languages.insertOne({"name":"python","type":"general purpose","versions":201})
```

Copied!

2. To insert more than one document at the same time, you can use `insertMany` command; which accepts an array as the argument.

```
1. 1
2. 2
3. 3
4. 4
5. 5
1. db.languages.insertMany([
2.     {"name":"scala","type":"functional"},
3.     {"name":"c","type":"procedural"},
4.     {"name":"c++","type":"object oriented"}
5. ])
```

Copied!

Exercise 3 - Read documents

Let's try out different ways of querying documents.

1. Find how many documents in `languages` collection.

```
1. 1
1. db.languages.countDocuments()
```

Copied!

2. List the first document in the collection.

```
1. 1
1. db.languages.findOne()
```

Copied!

3. List all documents in the collection.

```
1. 1
1. db.languages.find()
```
Copied!

4. List first 3 documents in the collection.

```
1. 1
1. db.languages.find().limit(3)
```
Copied!

5. Query for "python" language.

```
1. 1
1. db.languages.find({"name":"python"})
```
Copied!

6. Query for "object oriented" languages.

```
1. 1
1. db.languages.find({"type":"object oriented"})
```
Copied!

7. Use projection to only `project` specific fields. Using a projection document you can specify what fields we wish to see or skip in the output.

   This command lists all the documents with only `name` field in the output.

```
1. 1
1. db.languages.find({},{"name":1})
```
Copied!

8. This command lists all the documents without the `name` field in the output.

```
1. 1
1. db.languages.find({},{"name":0})
```
Copied!

9. This command lists all the `object oriented` languages with only `name` field in the output.

```
1. 1
1. db.languages.find({"type":"object oriented"},{"name":1})
```
Copied!

Exercise 4 - Update documents

You will now update documents based on a criteria.

1. Add a field to all documents

   The `updateMany` command is used to update documents in a mongodb collection, and it has the following generic syntax.

```
1. 1
1. db.collection.updateMany(<filter>,<update>)
```
Copied!

   Here we are adding a field `description` with value `programming language` to all documents.

```
1. 1
1. db.languages.updateMany({},{$set:{"description":"programming language"}})
```
Copied!

2. Set the `creater` for python language.

```
1. 1
1. db.languages.updateMany({"name":"python"},{$set:{"creator":"Guido van Rossum"}})
```
Copied!

3. Set a field named `compiled` with a value `true` for all the `object oriented` languages.

```
1. 1
1. db.languages.updateMany({"type":"object oriented"},{$set:{"compiled":true}})
```
Copied!

4. Increment `version` for `python` by `1`.

```
1. 1
1. db.languages.updateOne({"name":"python"},{$inc:{"version":1}})
```
Copied!

Exercise 5 - Delete documents

Delete documents based on a criteria.

1. Delete one `scala` language document.

```
1. 1
1. db.languages.deleteOne({"name":"scala"})
```
Copied!

2. Delete all `object oriented` languages.

```
1. 1
1. db.languages.deleteMany({"type":"object oriented"})
```
Copied!

3. Delete all the documents in a collection.

```
1. 1
1. db.languages.deleteMany({})
```
Copied!

Practice exercises

Run the below code on mongo console. It will insert 5 documents, which will serve as sample data for the next steps.

```
1. 1
```

```
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
```

```
1. use training
2. db.languages.insertMany([
3.     {"name":"java","type":"object oriented"},
4.     {"name":"python","type":"general purpose"},
5.     {"name":"scala","type":"functional"},
6.     {"name":"c","type":"procedural"},
7.     {"name":"c++","type":"object oriented"}
8. ])
```

Copied!

1. Problem:

   Insert an entry for `Haskell` programming language which is of type `functional`.

▼ Click here for Hint

   use the command `db.collection.insert()` or `db.collection.insertOne()`

▼ Click here for Solution

On the mongo client run the below commands.

```
1. 1
```

```
1. db.languages.insertOne({"name":"Haskell","type":"functional"})
```

Copied!

2. Problem:

   Query all languages with `type` as `functional`.

▼ Click here for Hint

   use the command `db.collection.find(<filter>)` with suitable options

▼ Click here for Solution

On the mongo client run the below commands.

```
1. 1
```

```
1. db.languages.find({"type":"functional"})
```

Copied!

3. Problem:

   Add `Bjarne Stroustrup` as creator for c++.

▼ Click here for Hint

   use the command `db.collection.updateOne(<filter>,<change>)` with suitable options

▼ Click here for Solution

On the mongo client run the below commands.

```
1. 1
```

```
1. db.languages.updateOne({"name":"c++"},{$set:{"creator":"Bjarne Stroustrup"}})
```

Copied!

4. Problem:

   Delete all functional programming languages.

▼ Click here for Hint

   use the command `db.collection.deleteMany(<criteria>)` with suitable options

▼ Click here for Solution

On the mongo client run the below commands.

```
1. 1
```

```
1. db.languages.deleteMany({"type":"functional"})
```

Copied!

5. Problem:

   Disconnect from the mongodb server.

▼ Click here for Solution

Run the below command on the terminal.

```
1. 1
```

```
1. exit
```

Copied!

Summary

In this lab, you have gained the understanding of CRUD operations in MongoDB.

Author(s)

[Muhammad Yahya]