Lesson 01 Demo 04

Configuring MongoDB with a Java Project

Objective: To configure MongoDB with Java project by creating a Java project and adding JAR files

Tools required: Eclipse IDE

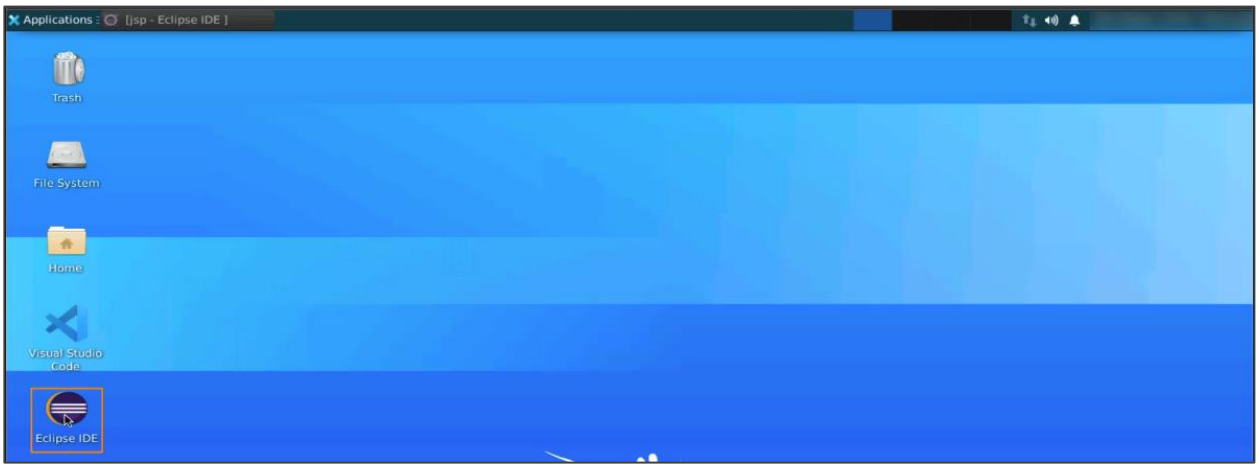
Prerequisites: None

Steps to be followed:

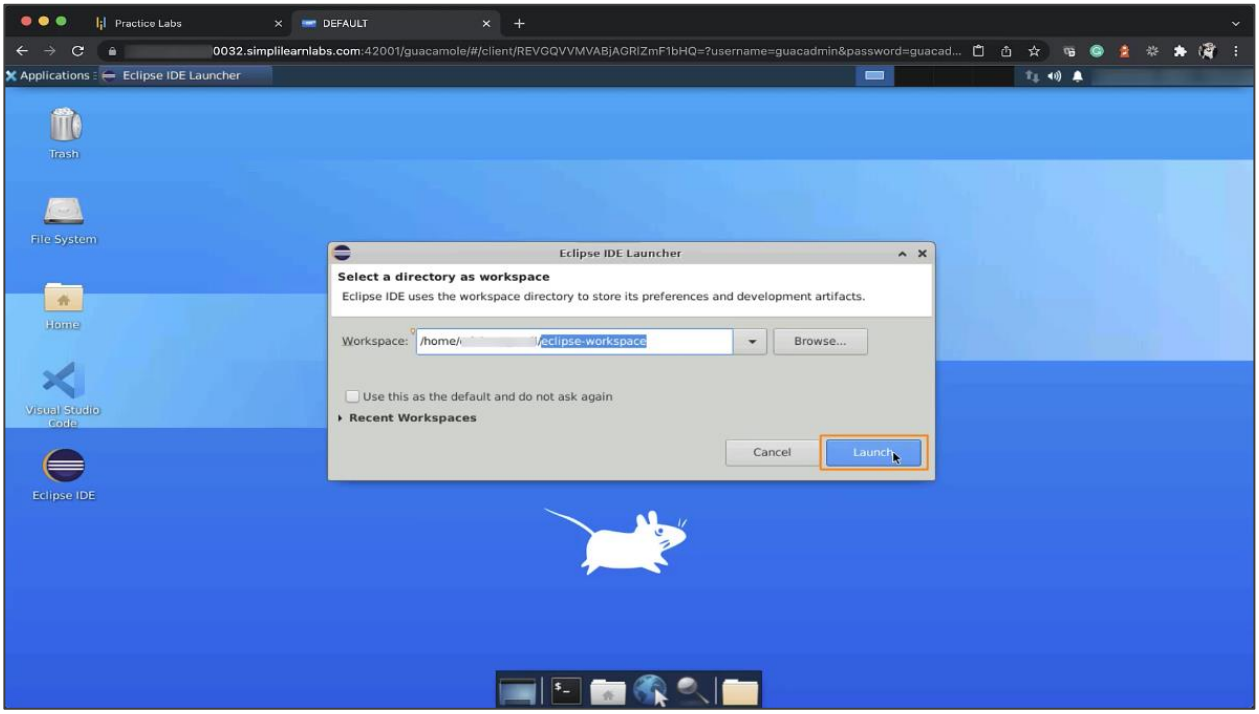
1. Create a Java project
2. Create a new class
3. Download JAR files
4. Add an external JAR
5. Create a new project
6. Add JAR files
7. Create a database

Step 1: Create a Java project

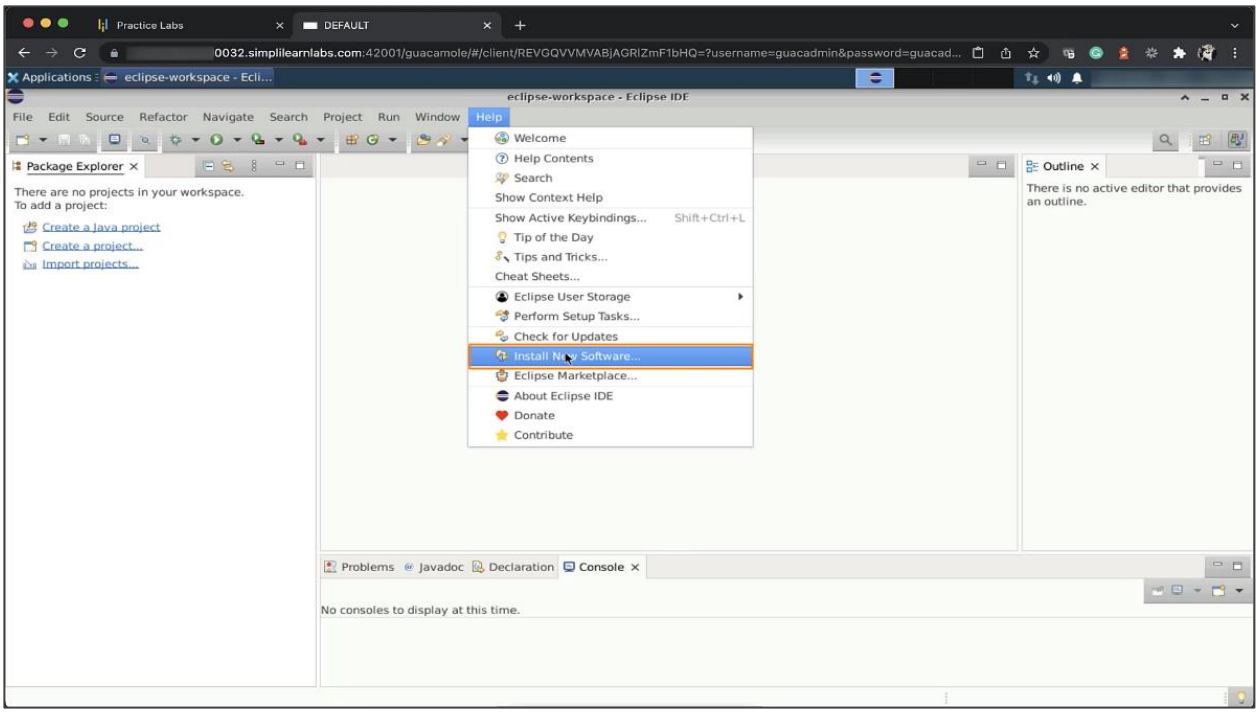
1.1 Open the Eclipse IDE



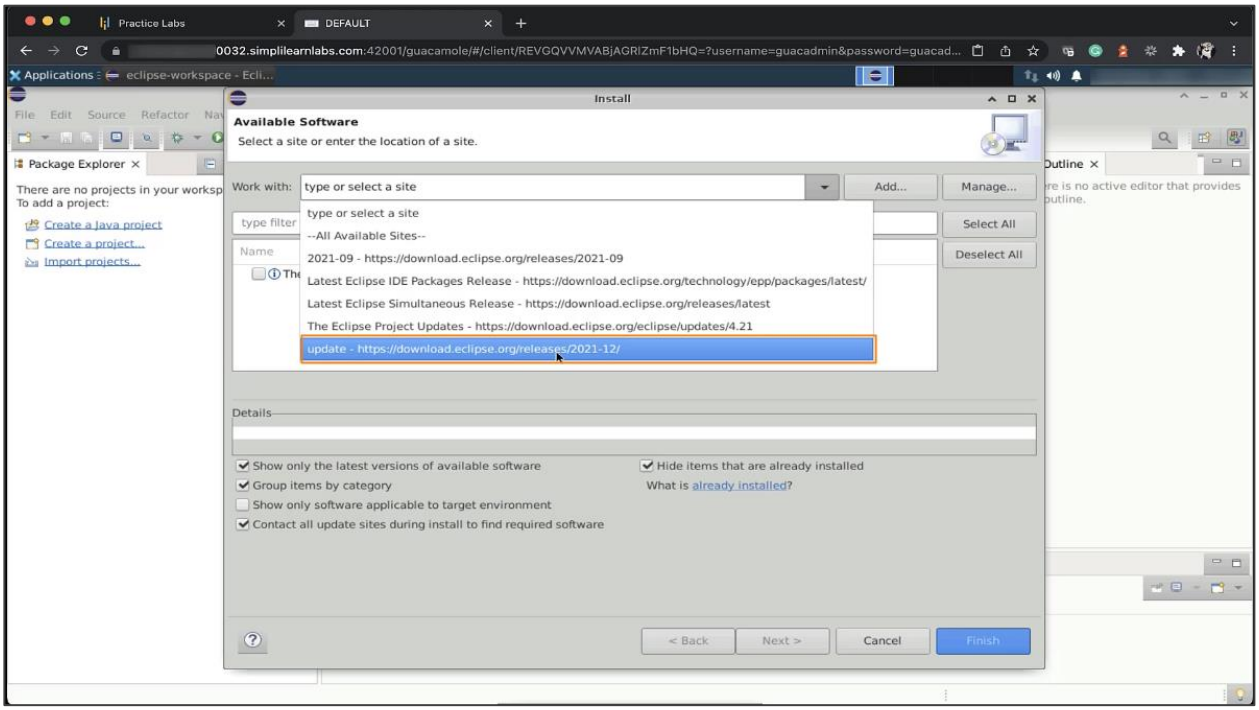
1.2 Keep the default workspace as the directory for the workspace and click on **Launch**



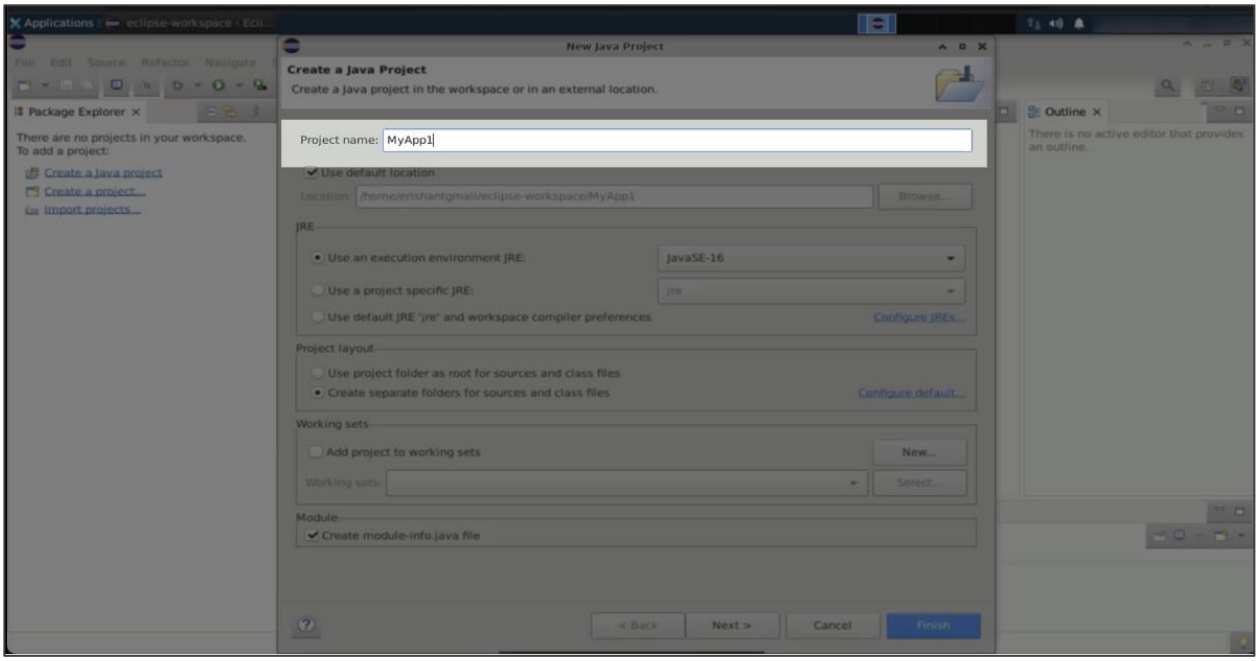
1.3 Click on the **Help** option and select the **Install New Software** option



1.4 Select the site **update - https://download.eclipse.org/release/2021-12** and click on **Finish**

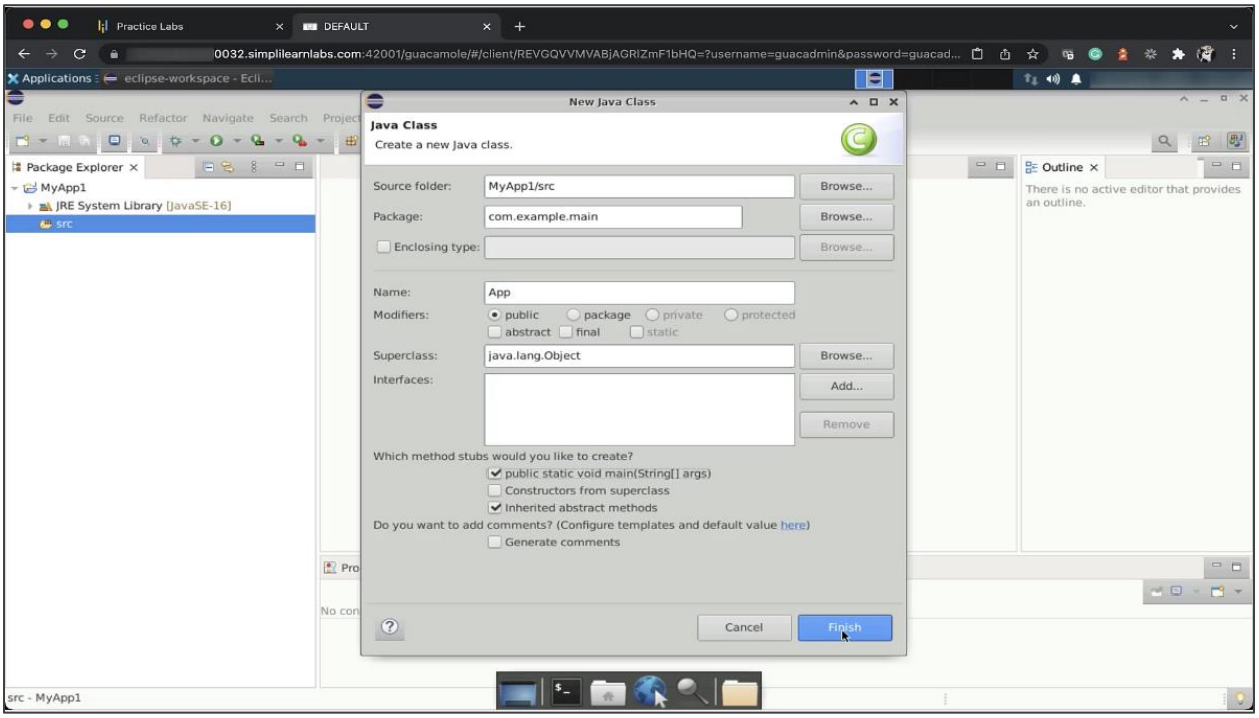


1.5 Click on **Create a Java Project**, name the project, click on **Next**, and then click on **Finish**

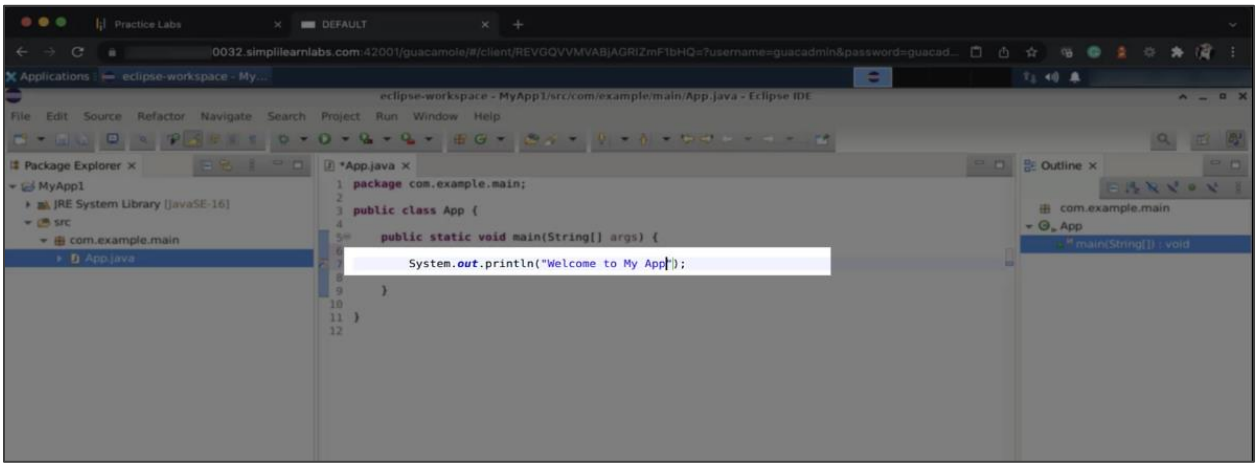


Step 2: Create a new class

2.1 Right-click on **src**, name the **source folder**, **package**, **name**, and click on **Finish**

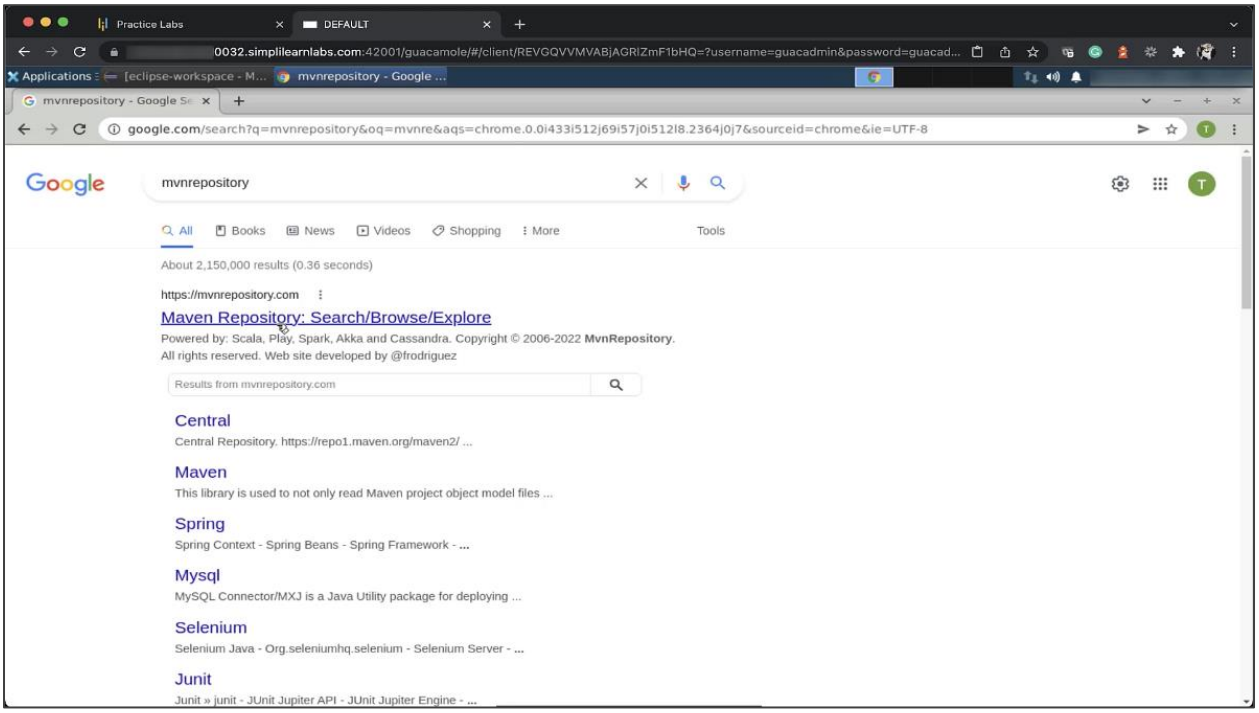


2.2 Write the print statement to display the welcome message (line 7):

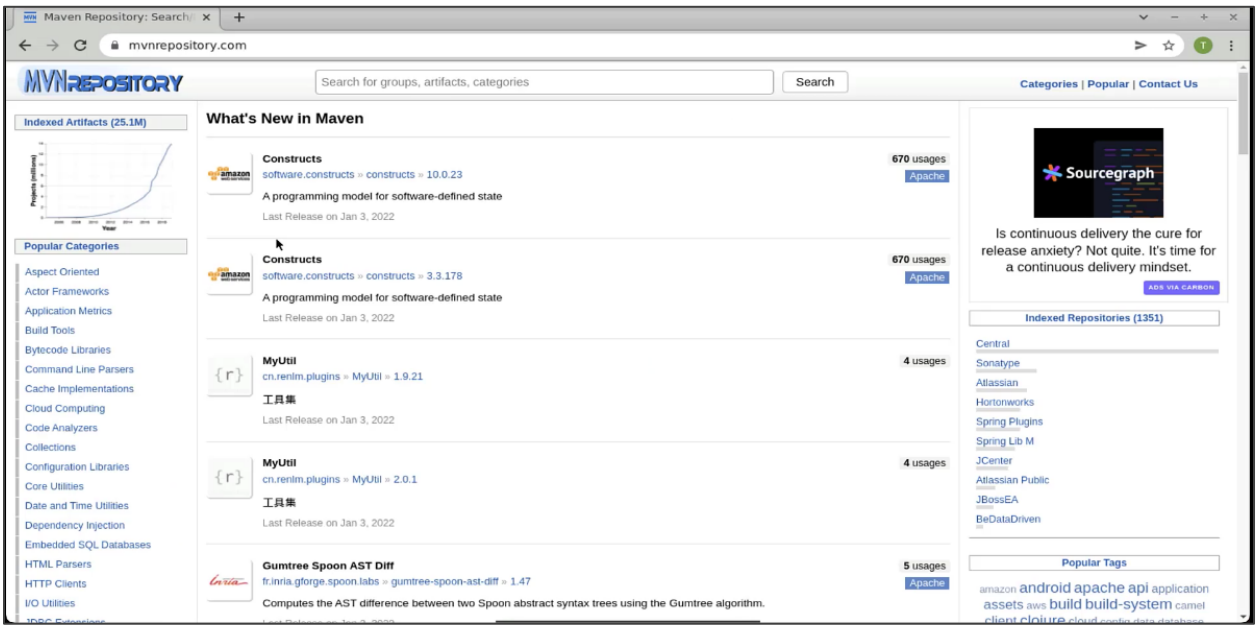


Step 3: Download JAR files

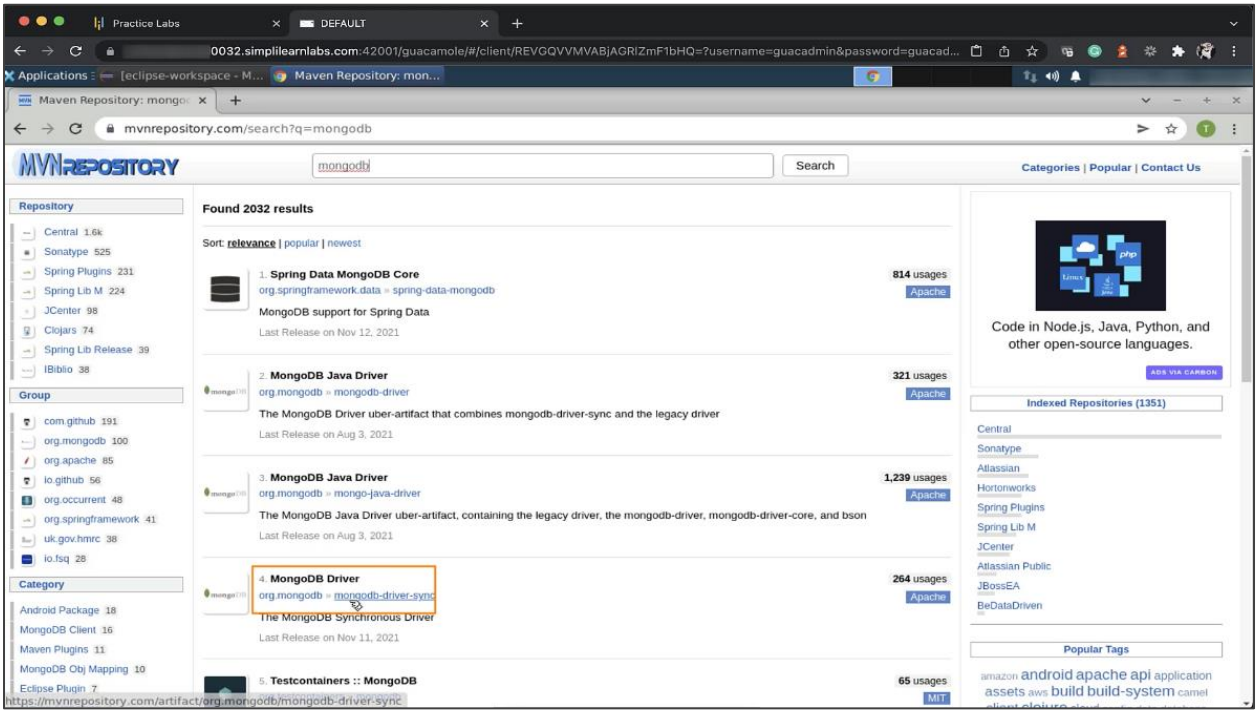
3.1 Go to the **browser**, search for **mvnrepository**, and open the link **mvnrepository.com**



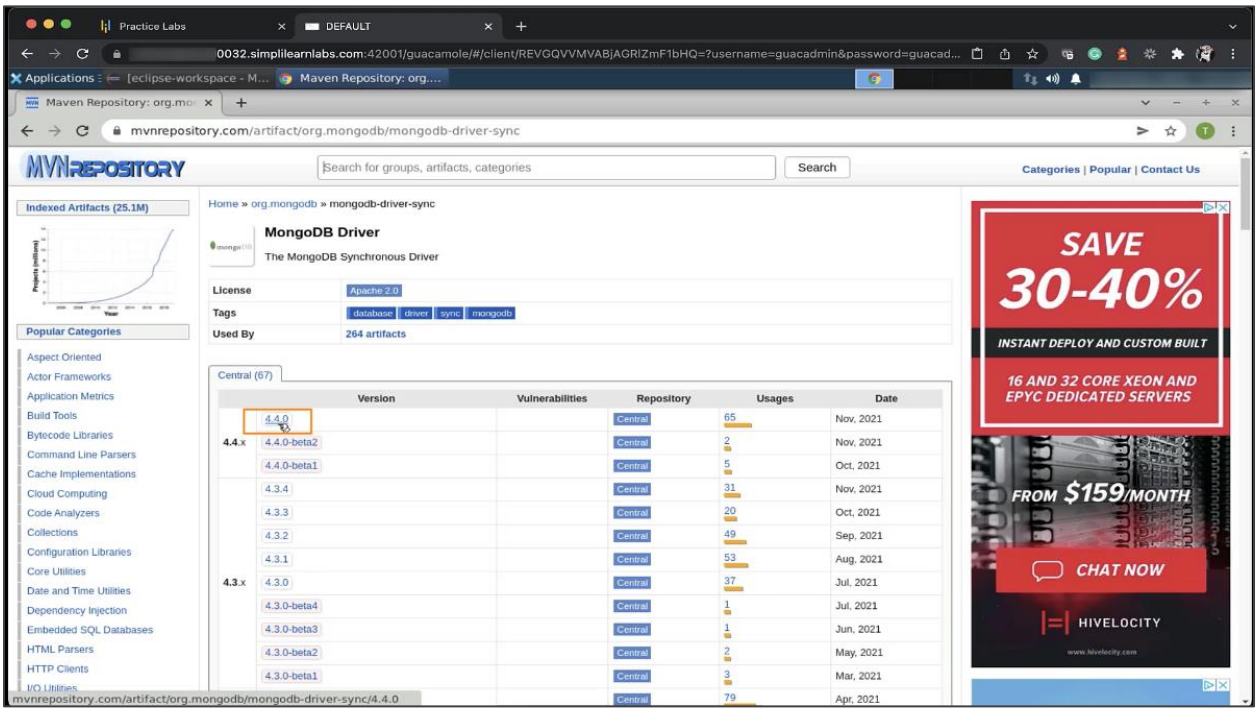
The following page can be seen:



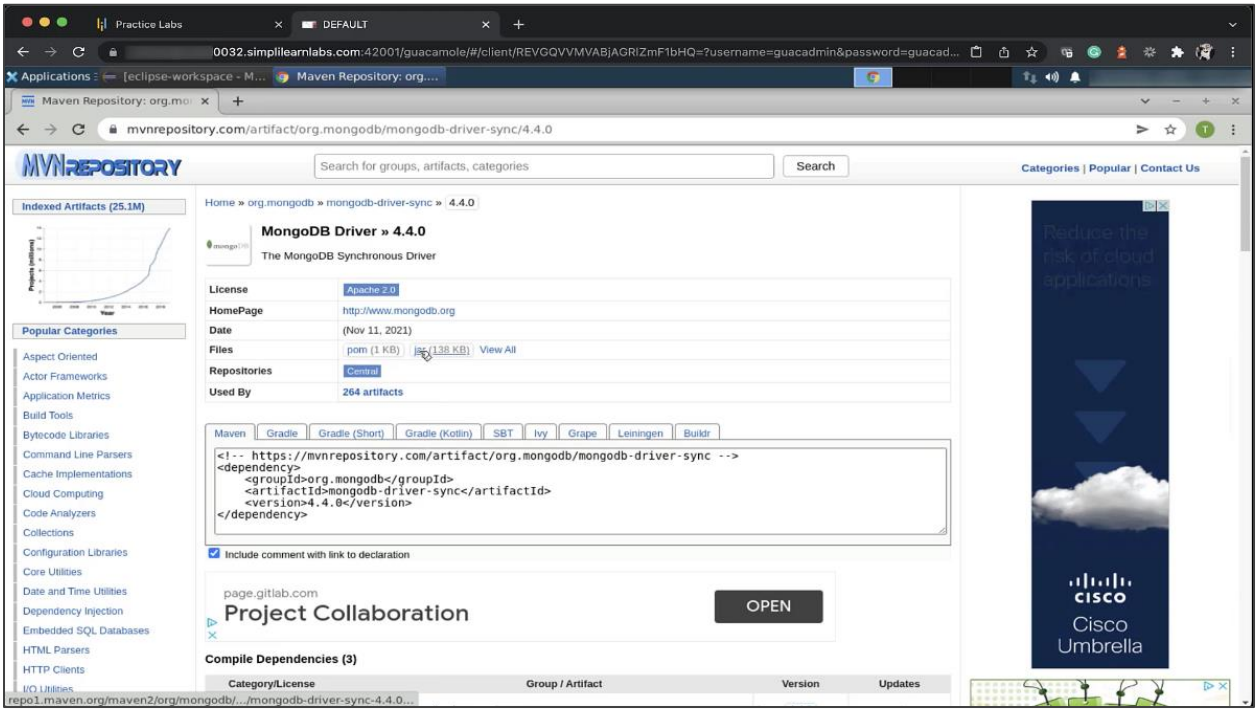
3.2 Search for **MongoDB Driver** and click on it



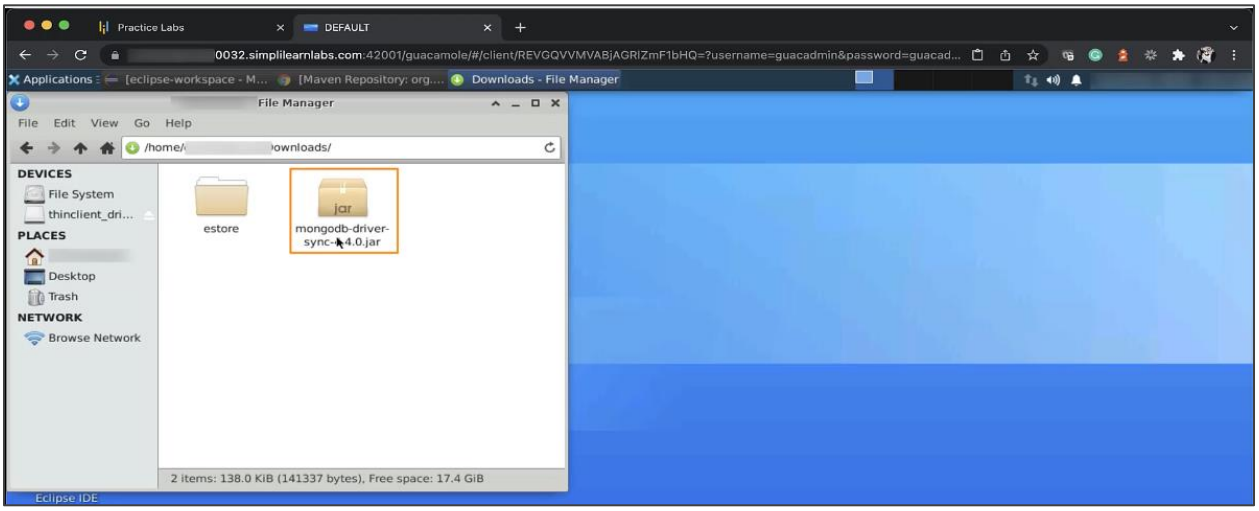
3.3 Open the latest version **4.4.0**



3.4 Click on the **JAR** to download the JAR files

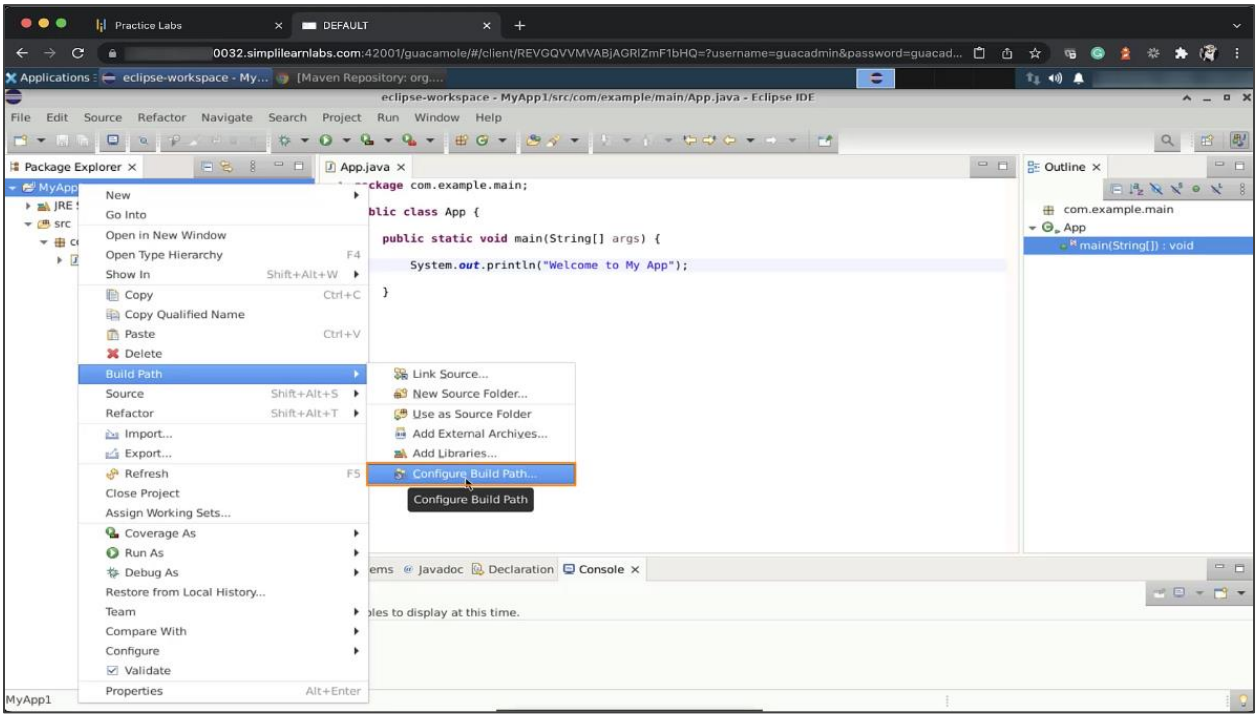


3.5 Now, go to the **Downloads** directory in **File Explorer** and you can see the downloaded JAR file there:

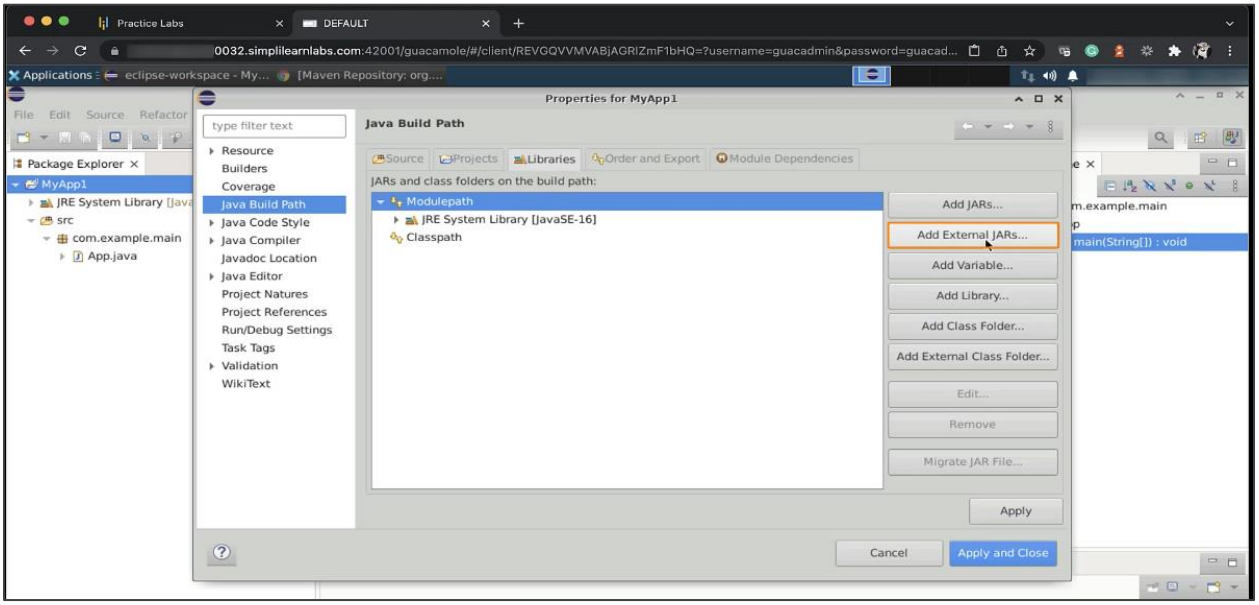


Step 4: Add the external JAR

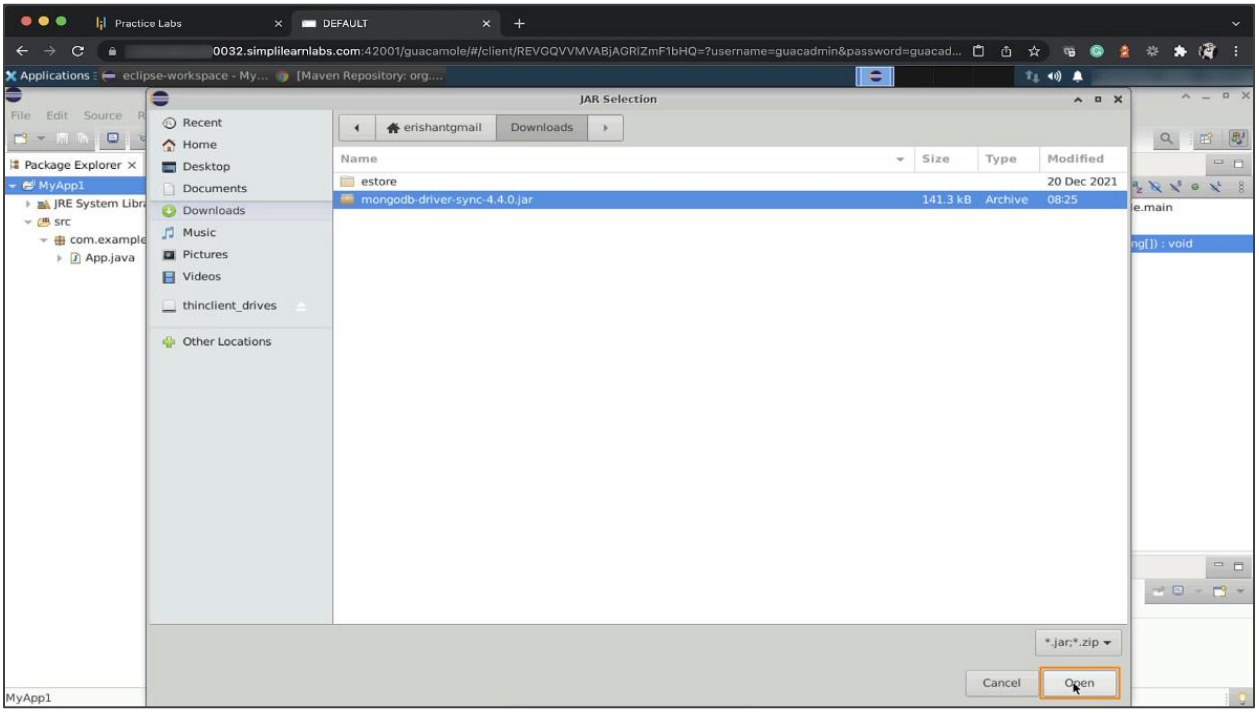
4.1 Go to Eclipse, right-click on the project, select **Build Path**, and select **Configure Build Path**



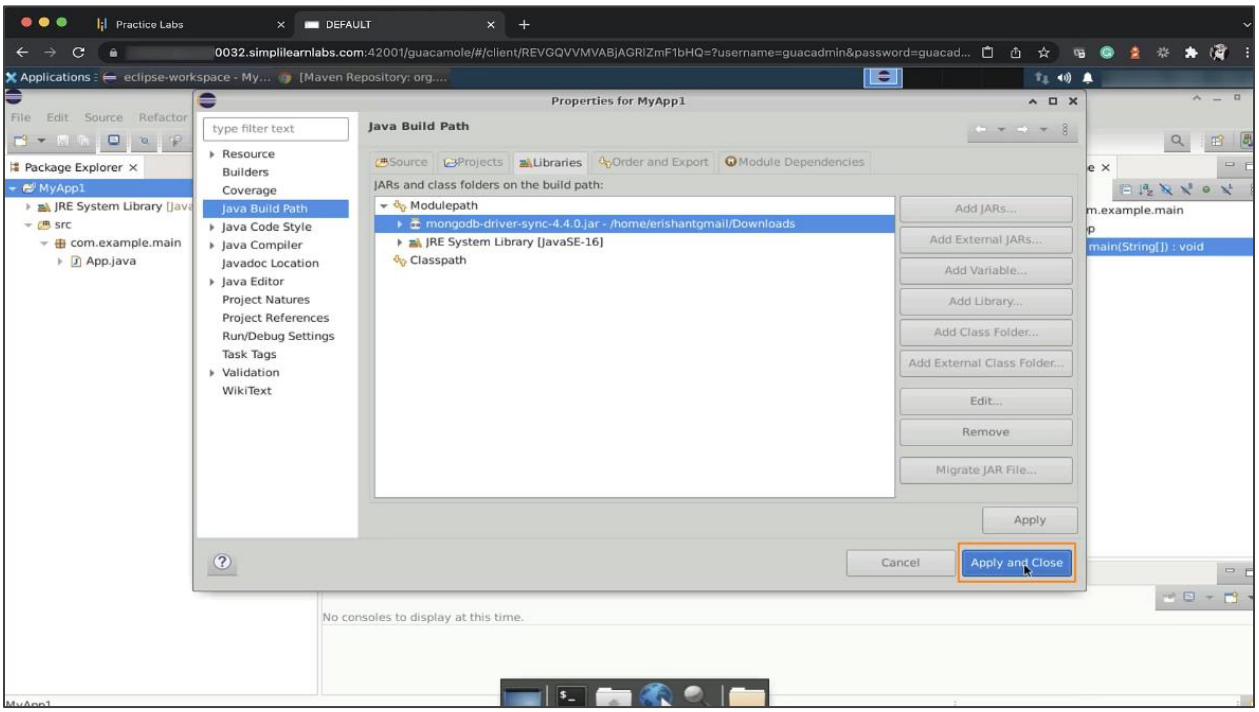
4.2 Go to the **Libraries** tab, select **Modulepath**, and click on **Add External JARs**

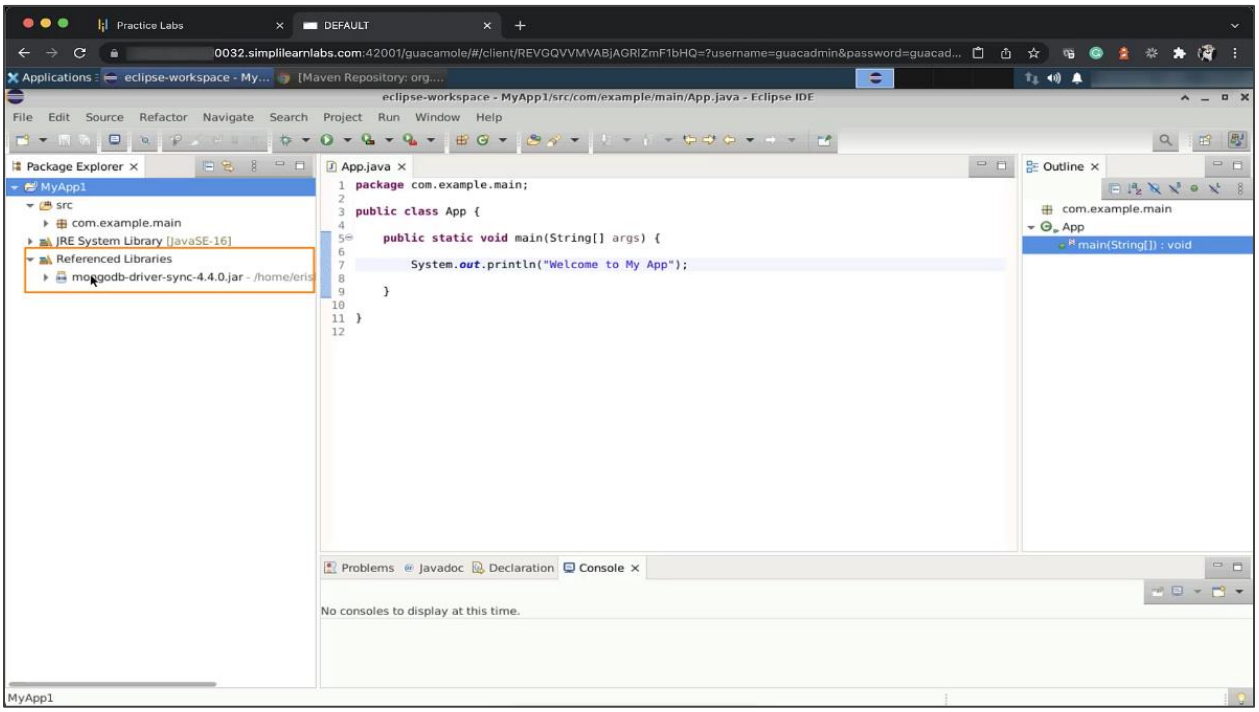


4.3 In the **Downloads** directory, select the JAR file and click on **Open**



4.4 Click on **Apply** and **Close**

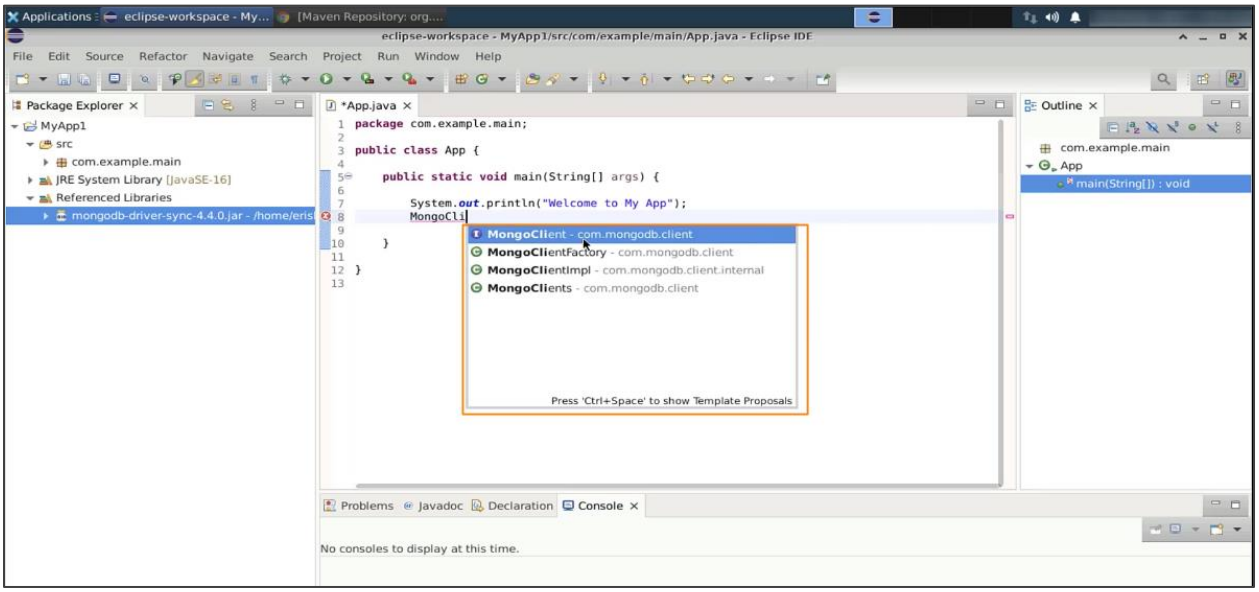




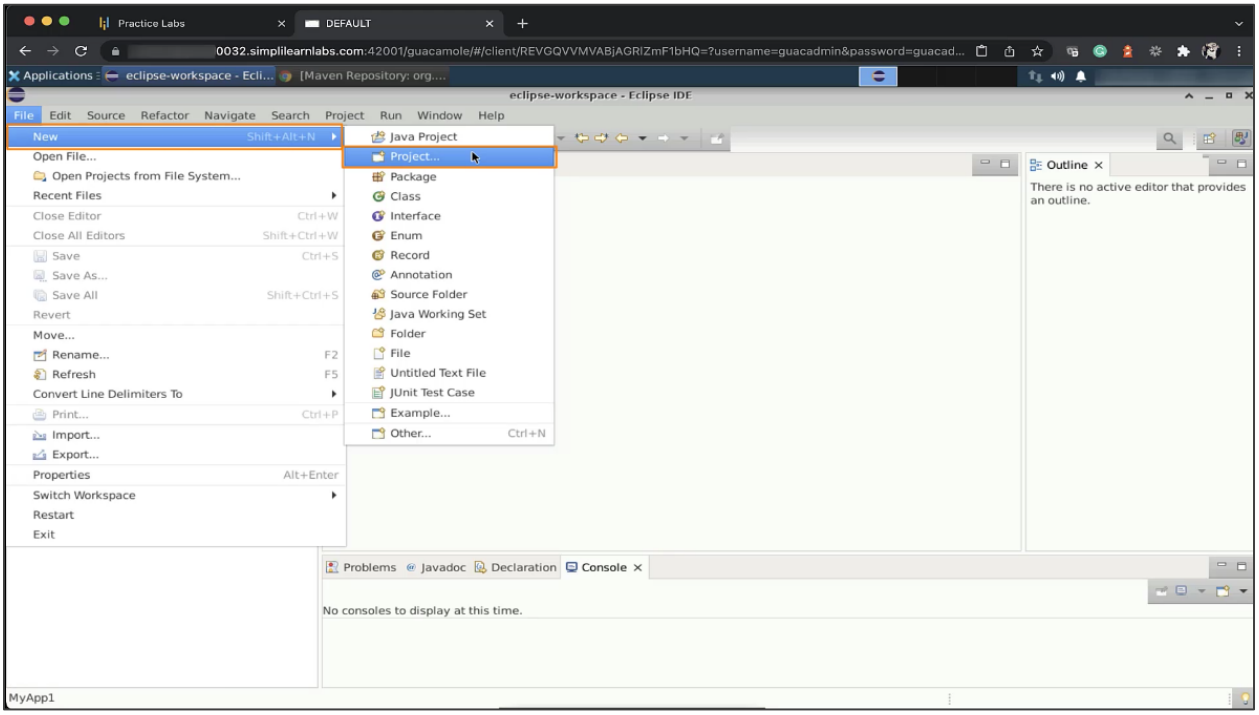
The JAR file has been added.

Step 5: Create a new project

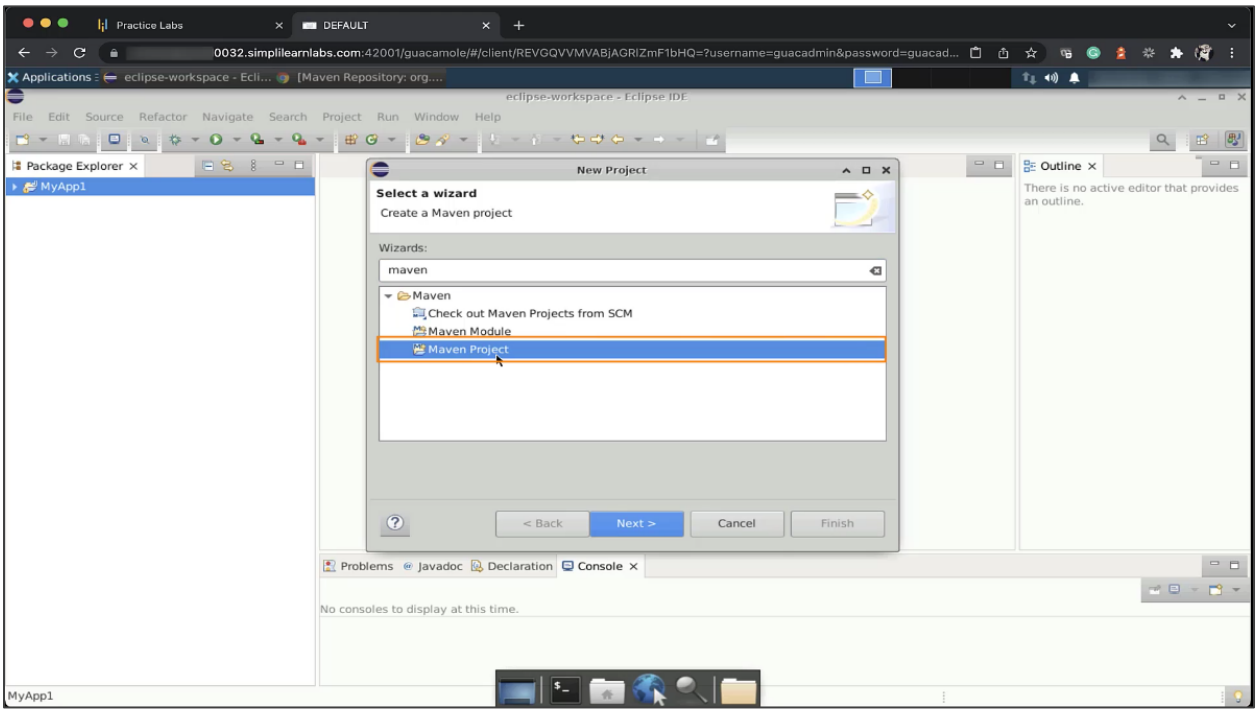
5.1 To check if the driver is working or not, write **MongoClient**, and it can be seen that the interface is available from the MongoDB client



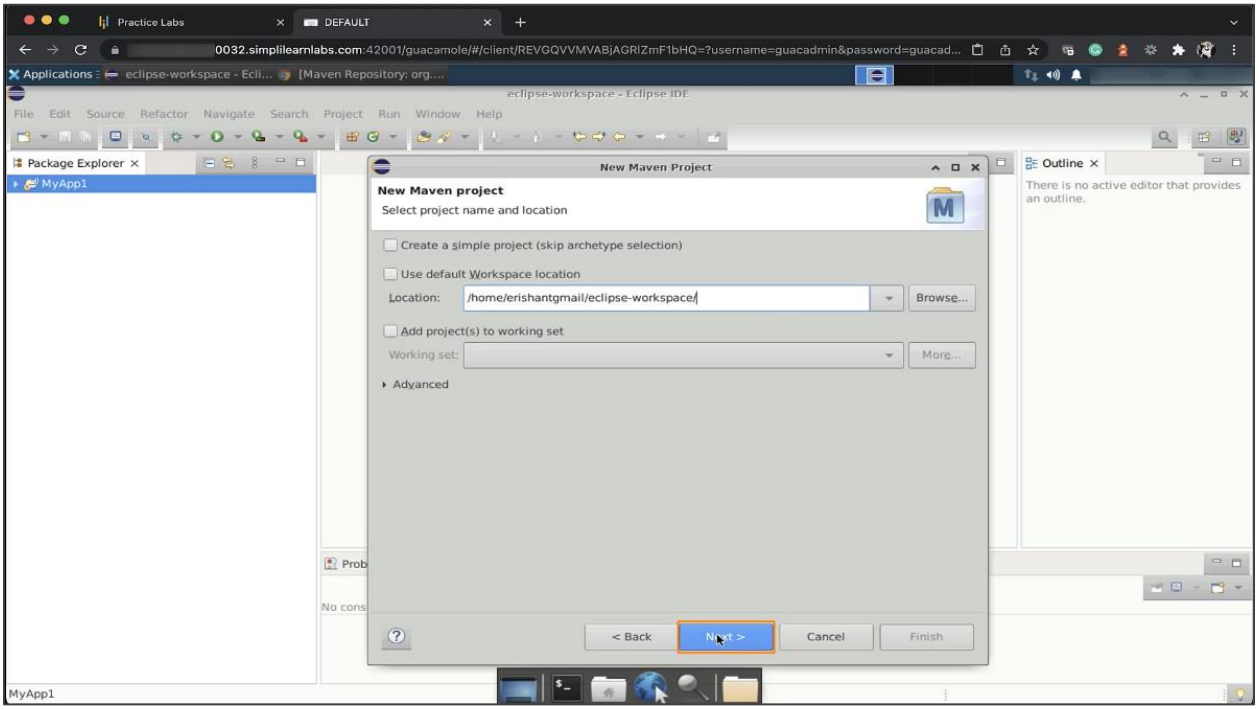
5.2 Go to **File**, select **New**, and select **Project** to create a new project



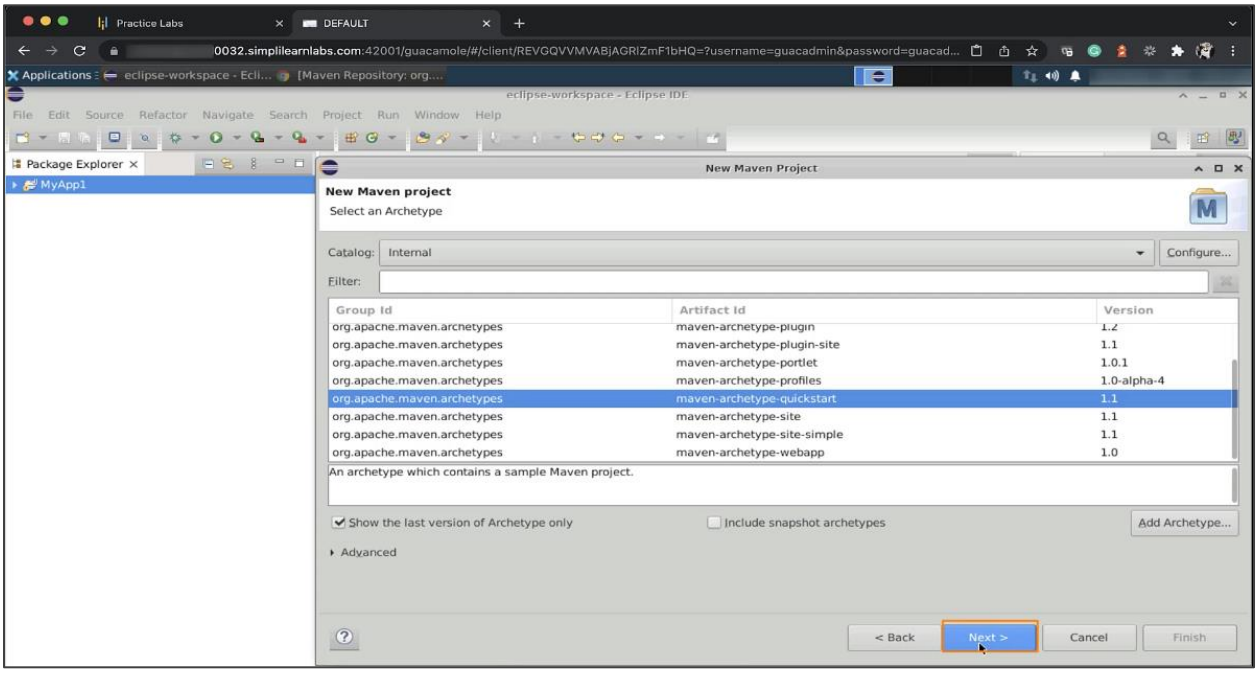
5.3 Search for maven, select **Maven Project**, and click on **Next**



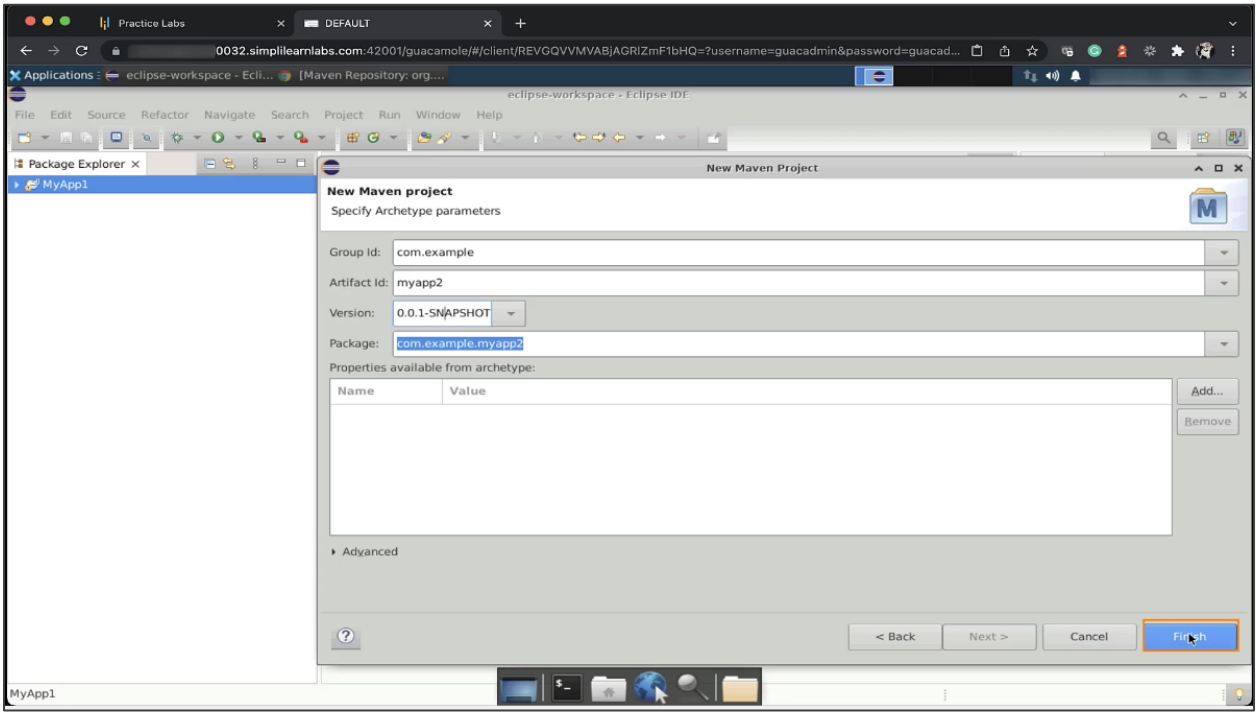
5.4 Select the default workspace location and click on **Next**

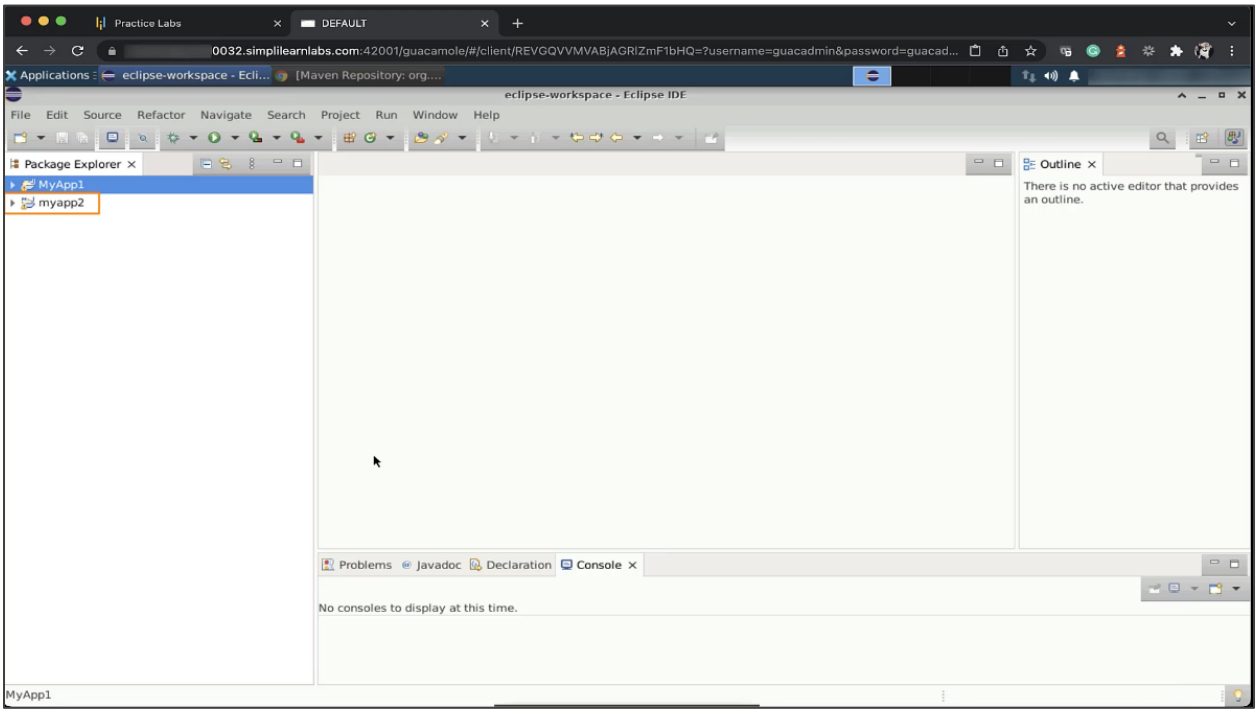


5.5 Select the archetype **maven-archetype-quickstart**, choose **Internal** in **Catalog**, and click on **Next**



5.6 Enter the **Group Id**, **Artifact Id**, **Version**, and click on **Finish**

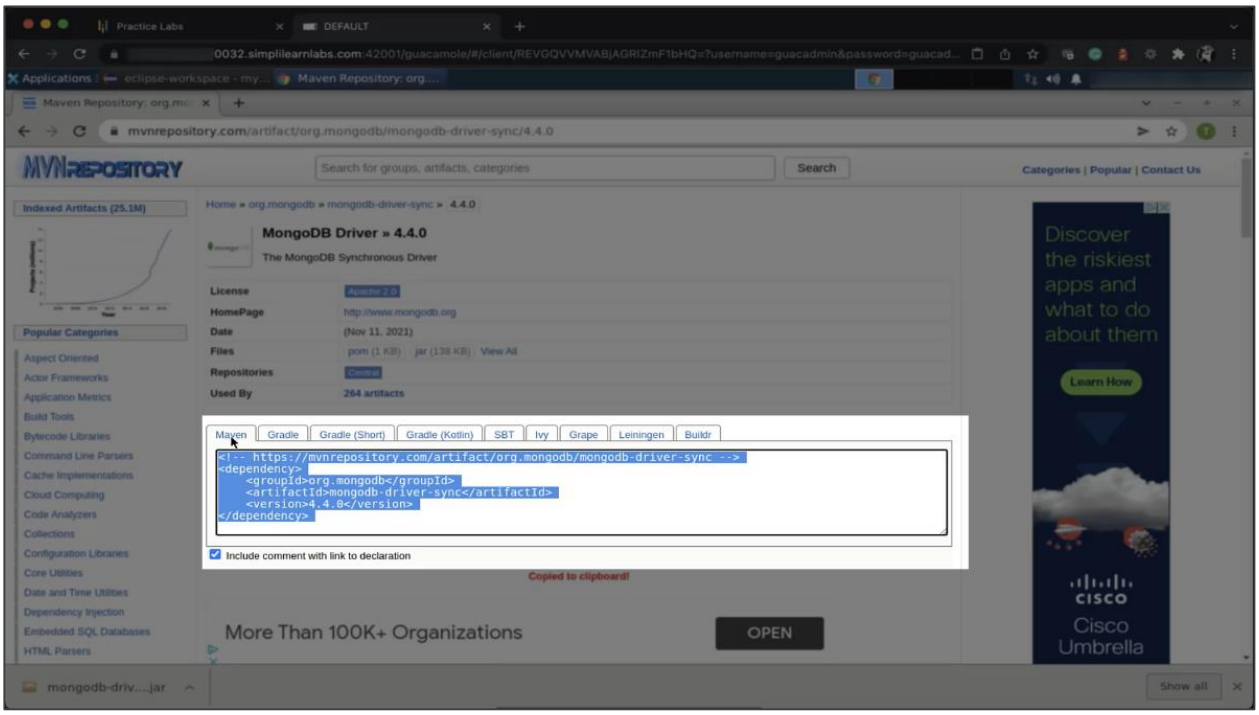




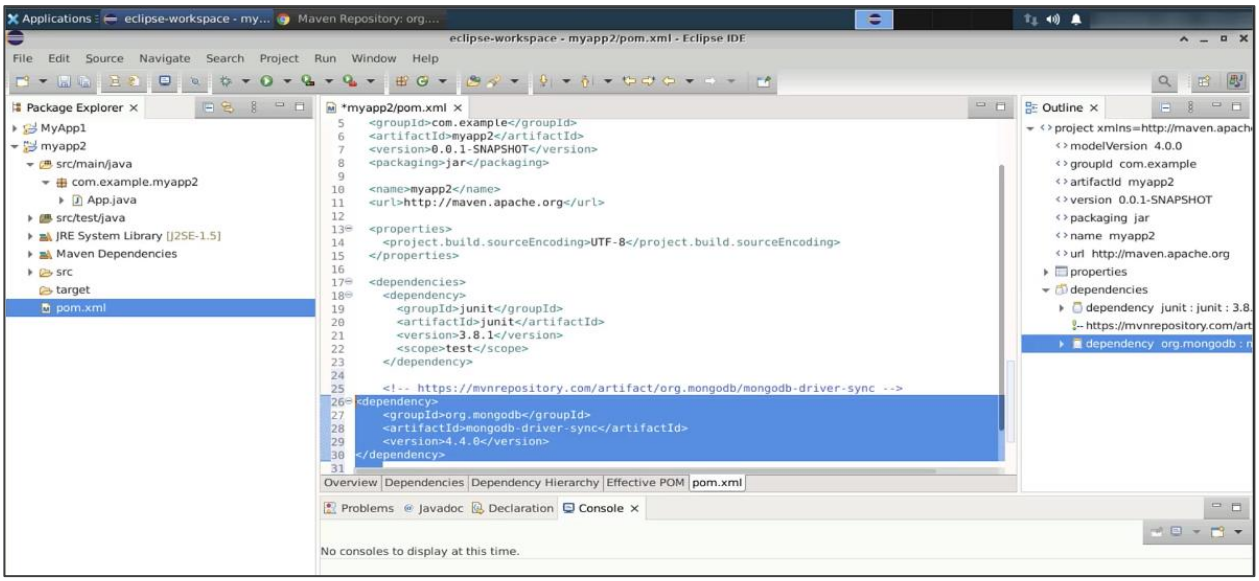
A new project is now created.

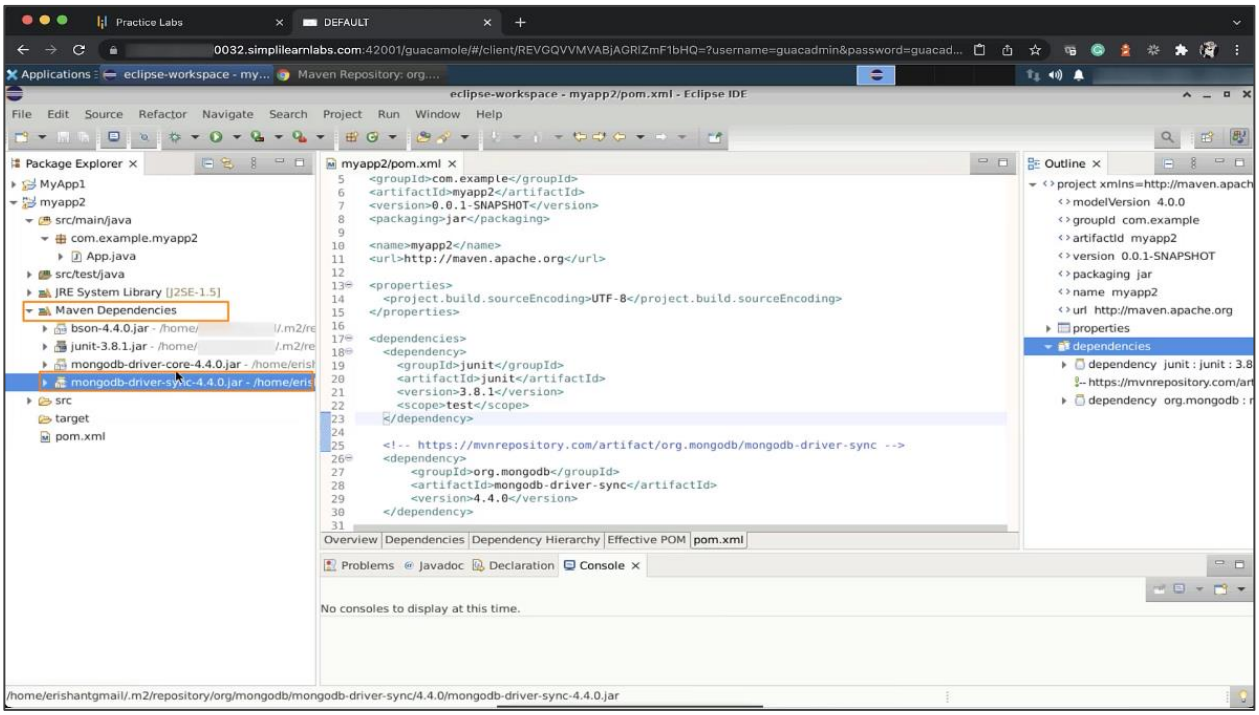
Step 6: Add JAR files

6.1 Now, go back to the **mvnrepository** in the browser and copy the dependency tag for Maven



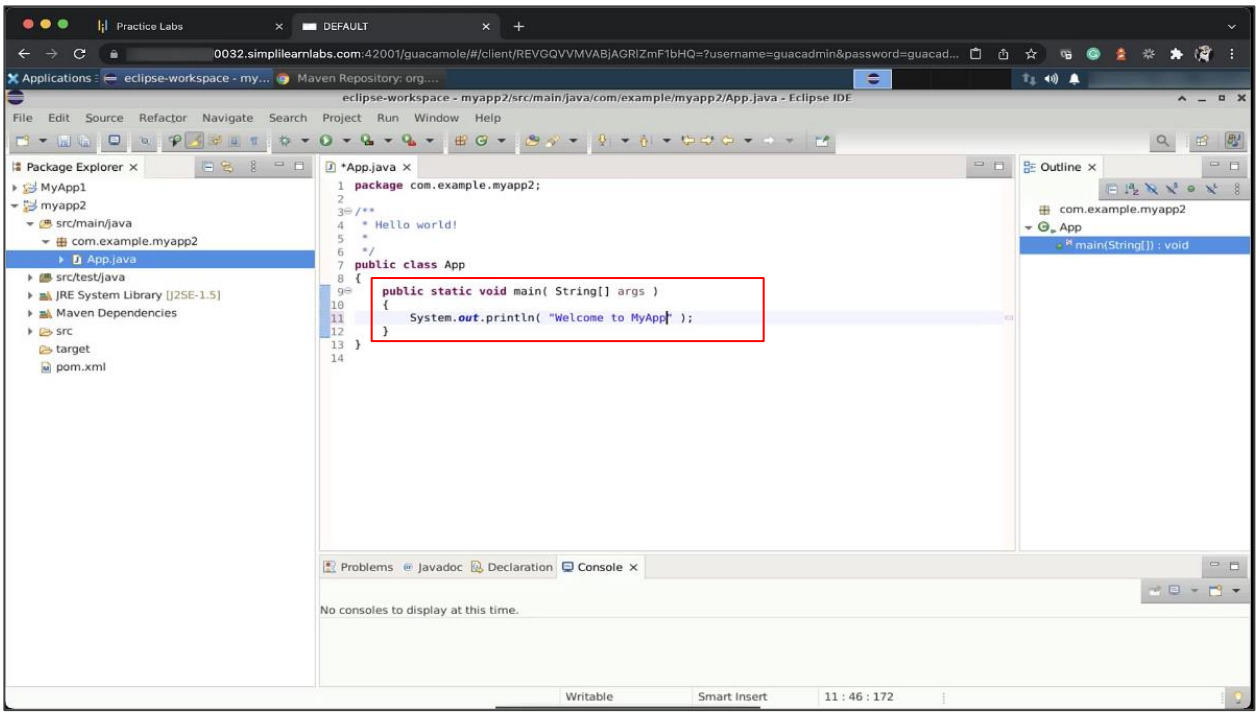
6.2 Paste the copied tag in the **pom.xml** file in Eclipse (lines 23 to 30):





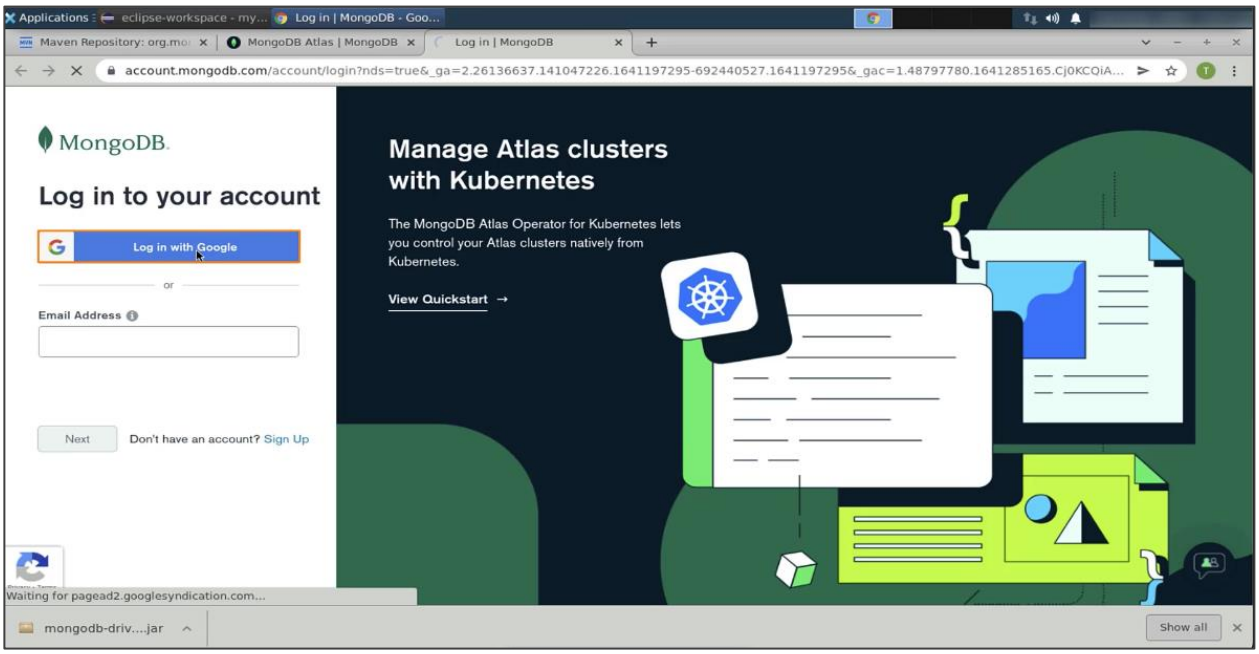
The JAR file has been added in the **Maven Dependencies**.

6.3 Go to the **App.java** file and change the message to **Welcome to MyApp** (line 11):

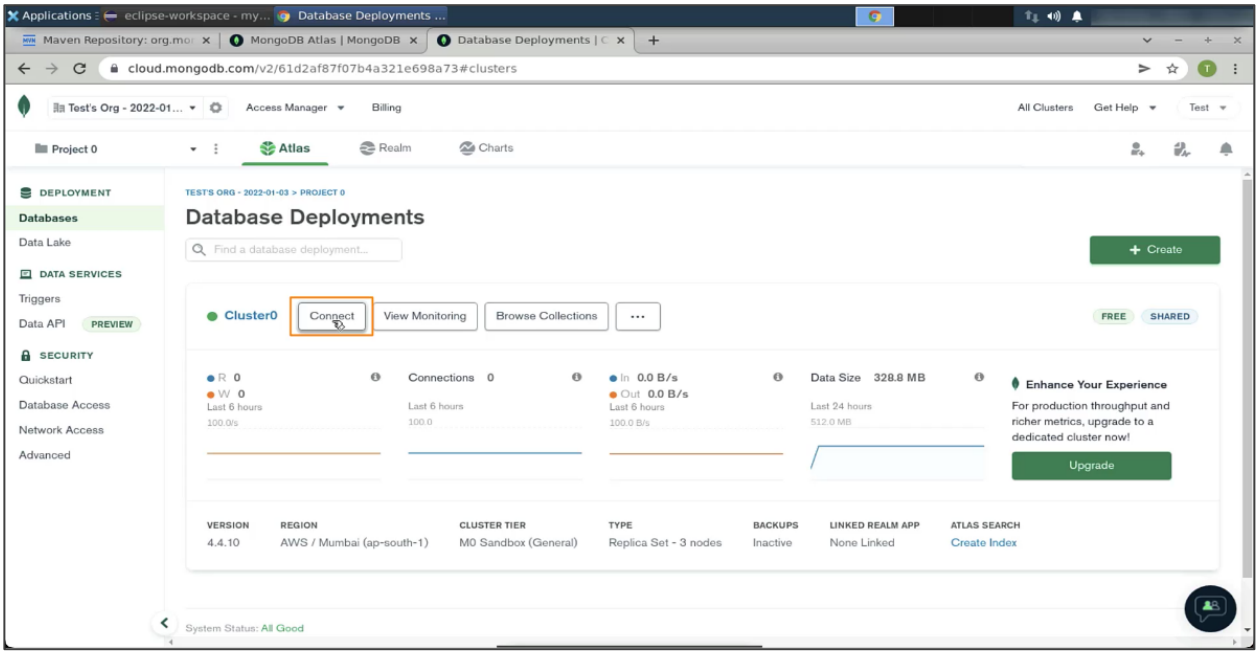


Step 7: Create a database

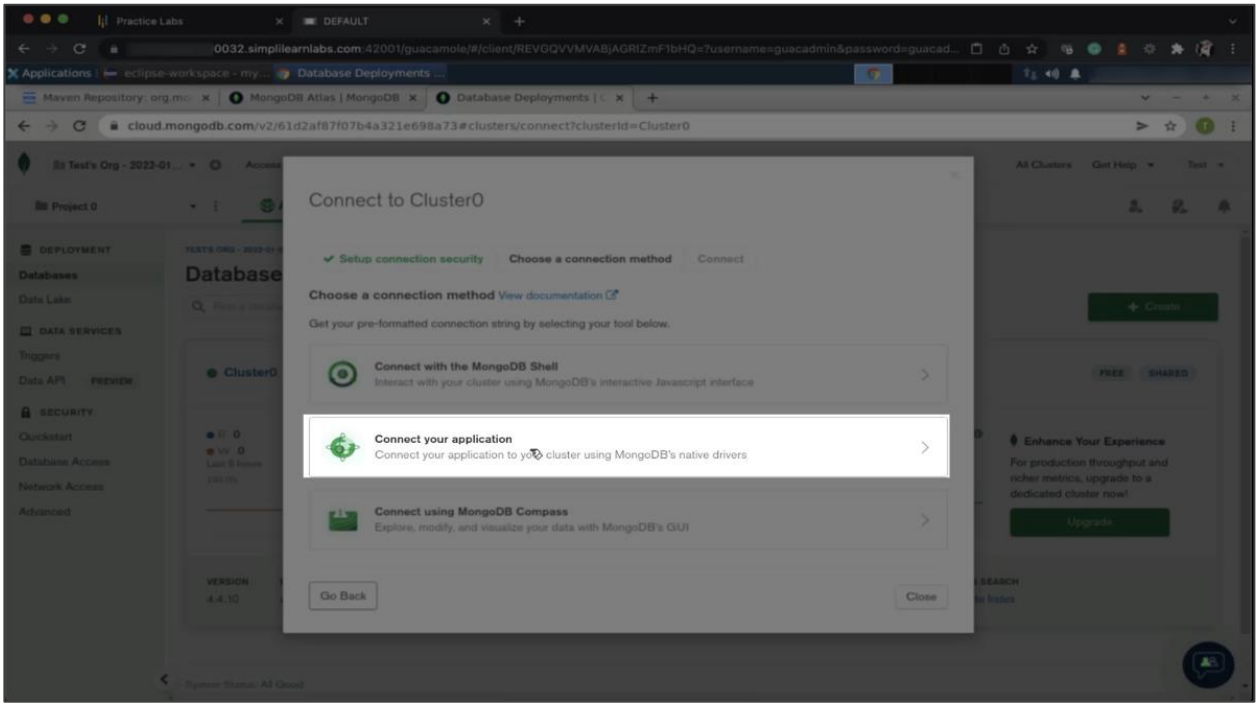
7.1 Now, go to the **browser** and open the **MongoDB** login, then click on **Log in with Google**



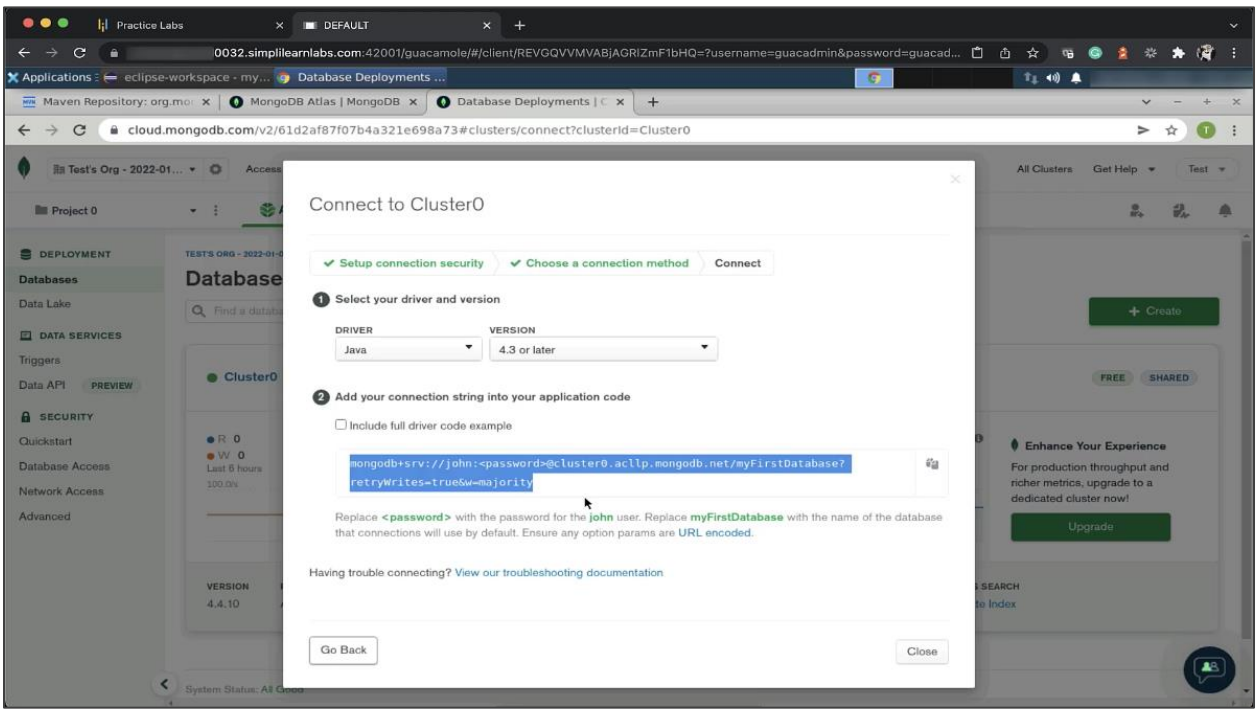
7.2 In **Database Deployments**, click on **Connect**



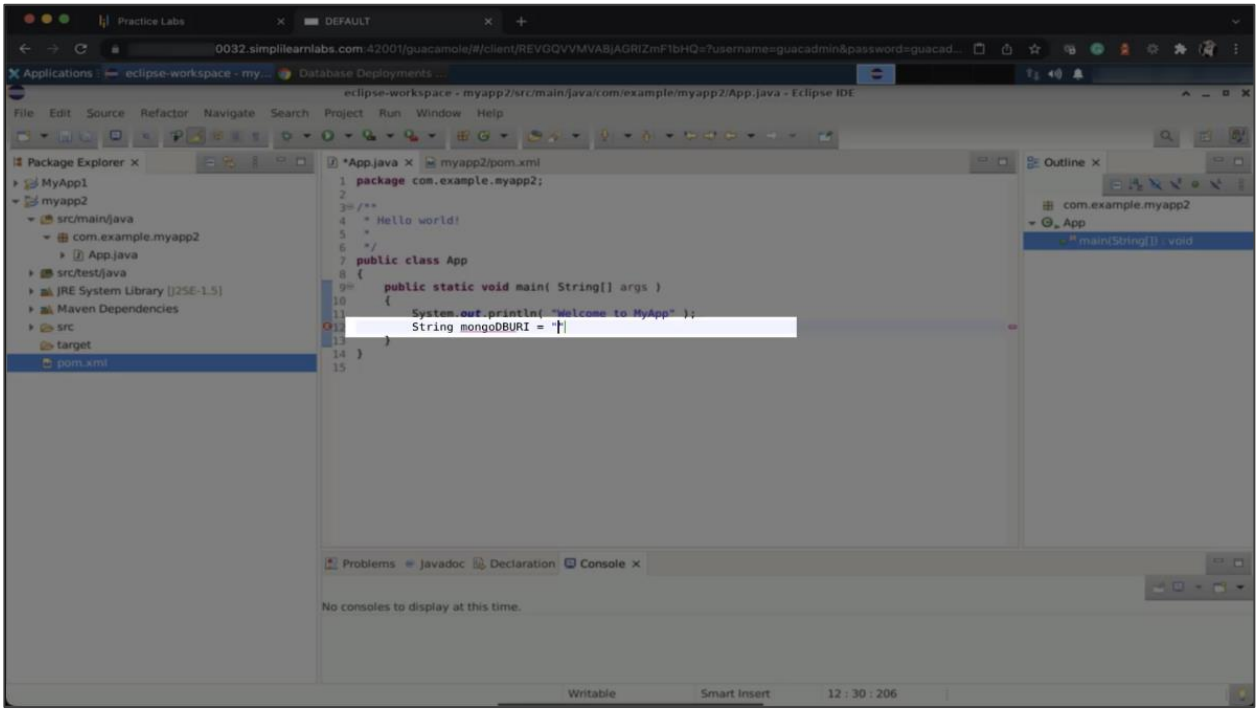
7.3 Select **Connect your application**



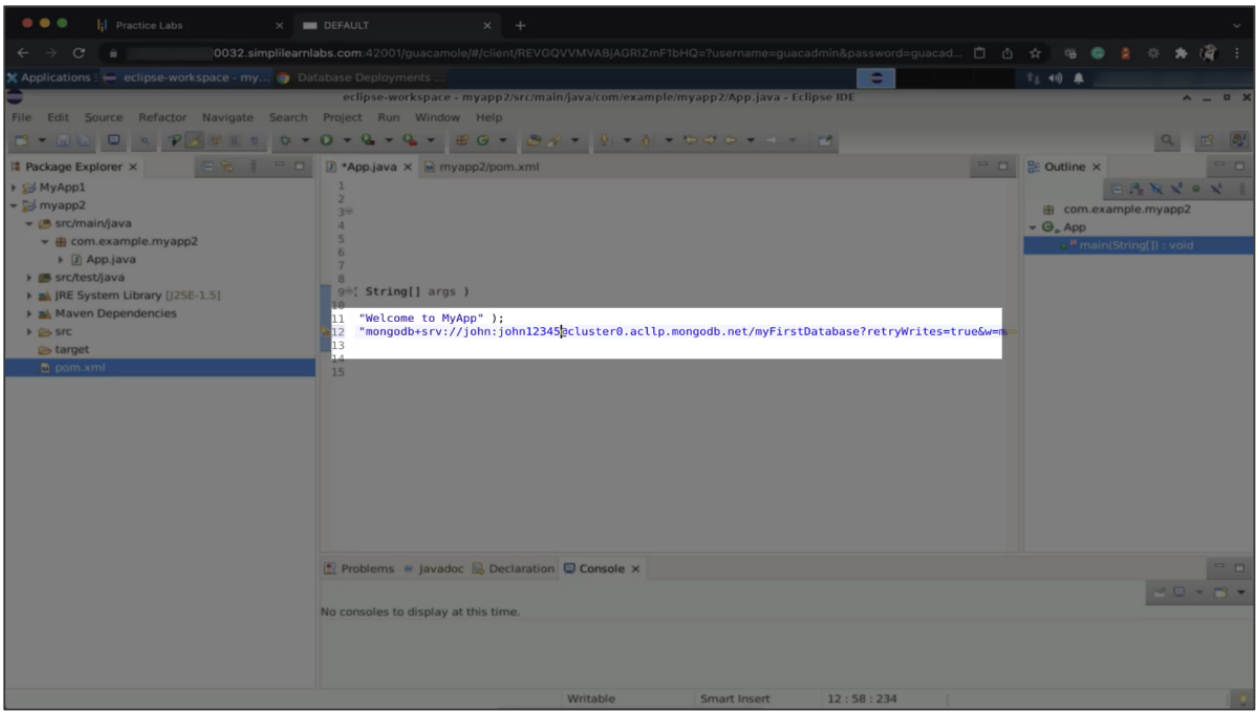
7.4 Copy the selected URL



7.5 Go to the **App.java** file and create a string **mongoDBURI** (line 12):



7.6 Paste the copied URL and edit the password to **john12345**

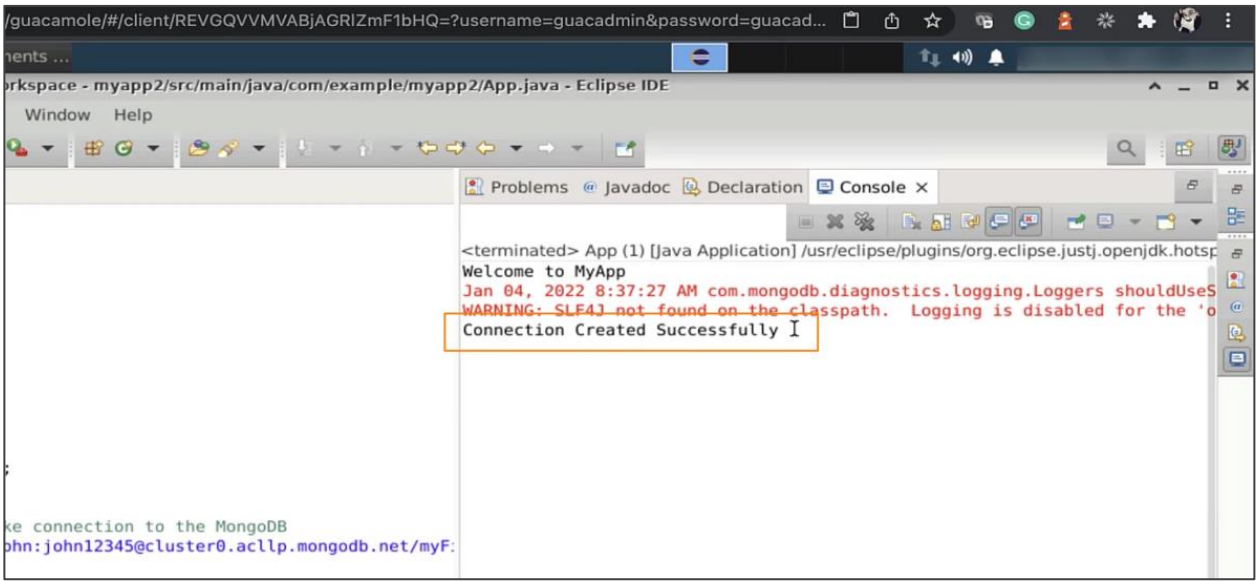


7.7 Go to the **welcome.jsp** page, create a connection with MongoDB using the API called **MongoClients**, and execute the **try-catch** block (from lines 17 to 26):

```
1  package com.example.myapp2;
2
3  import com.mongodb.client.MongoClient;
4  import com.mongodb.client.MongoClients;
5
6  /**
7   * Hello world!
8   *
9   */
10 public class App
11 {
12     public static void main( String[] args )
13     {
14         System.out.println( "Welcome to MyApp" );
15
16         try {
17             // URI i.e. Connection String to make connection to the MongoDB
18             String mongoDBURI = "mongodb+srv://john:john12345@cluster0.acllp.mongodb.net/myFirstDatabase?retryWrites=true&w=majority";
19
20             // 1. Connect to MongoDB
21             MongoClient client = MongoClients.create(mongoDBURI);
22             System.out.println("Connection Created Successfully");
23         } catch (Exception e) {
24             // TODO: handle exception
25             System.out.println("Something Went Wrong");
26             System.out.println("Message: "+e.getMessage());
27         }
28
29
30     }
31 }
32
```

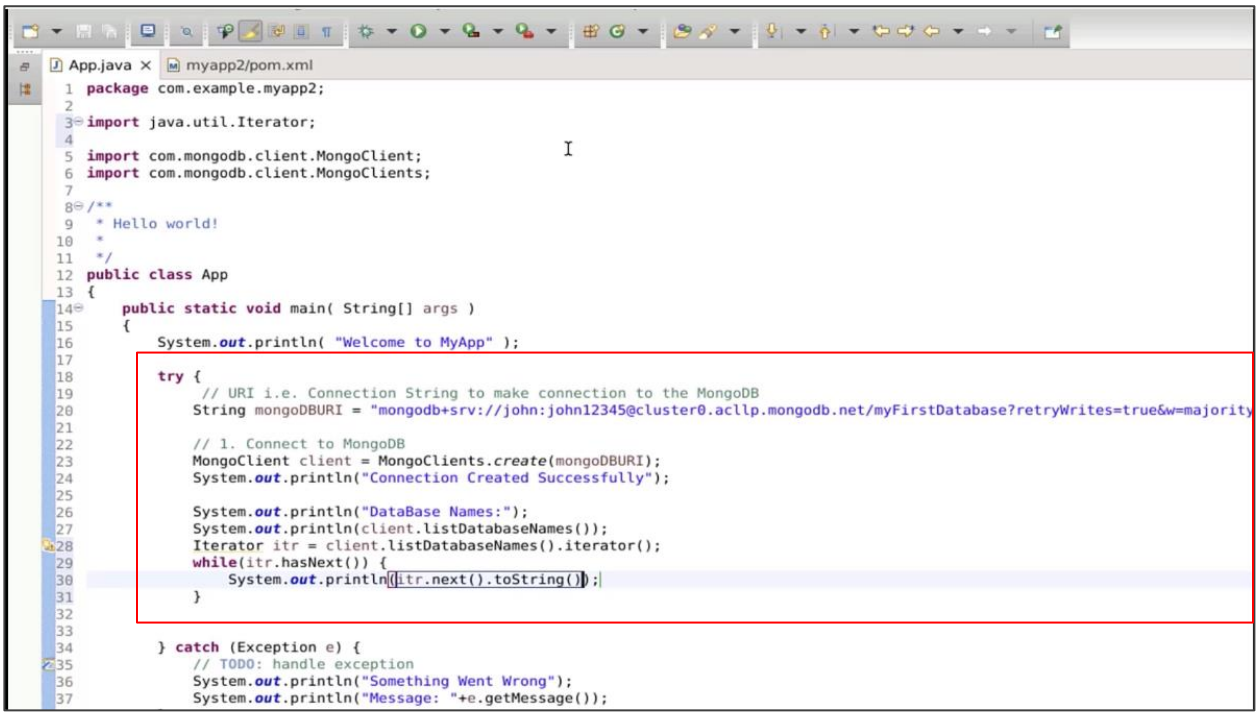
7.8 Run the code

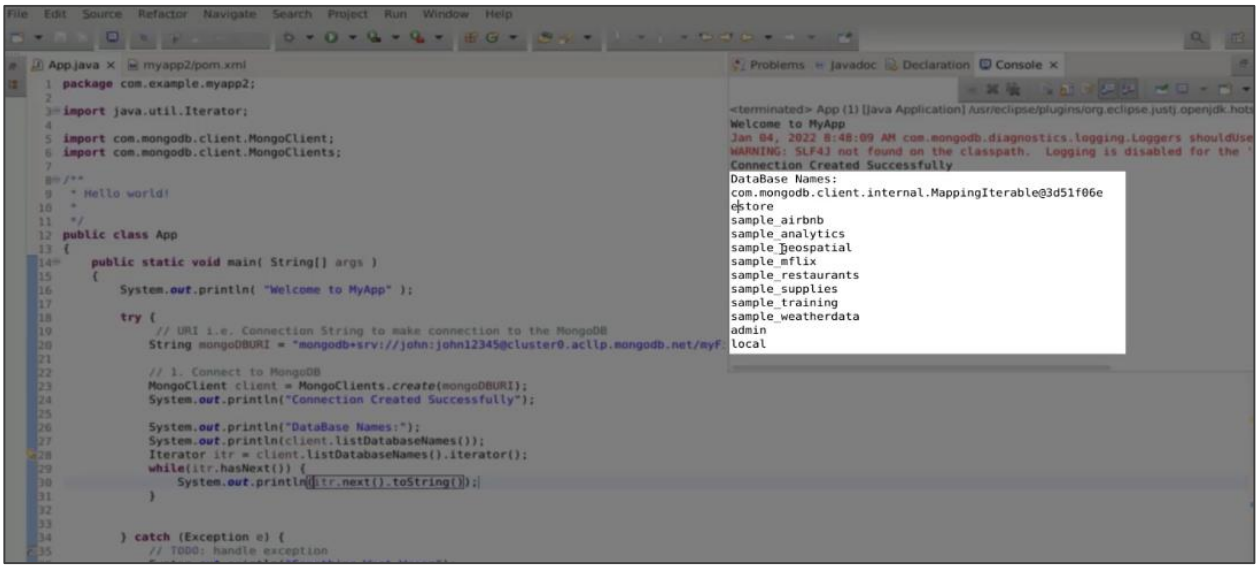
```
1  package com.example.myapp2;
2
3  import com.mongodb.client.MongoClient;
4  import com.mongodb.client.MongoClients;
5
6  /**
7   * Hello world!
8   *
9   */
10 public class App
11 {
12     public static void main( String[] args )
13     {
14         System.out.println( "Welcome to MyApp" );
15
16         try {
17             // URI i.e. Connection String to make connection to the MongoDB
18             String mongoDBURI = "mongodb+srv://john:john12345@cluster0.acllp.mongodb.net/myFirstDatabase?retryWrites=true&w=majority";
19
20             // 1. Connect to MongoDB
21             MongoClient client = MongoClients.create(mongoDBURI);
22             System.out.println("Connection Created Successfully");
23         } catch (Exception e) {
24             // TODO: handle exception
25             System.out.println("Something Went Wrong");
26             System.out.println("Message: "+e.getMessage());
27         }
28
29
30     }
31 }
32
```



The output can be seen as **Connection Created Successfully.** These are the logs coming from the **MongoDB** API.

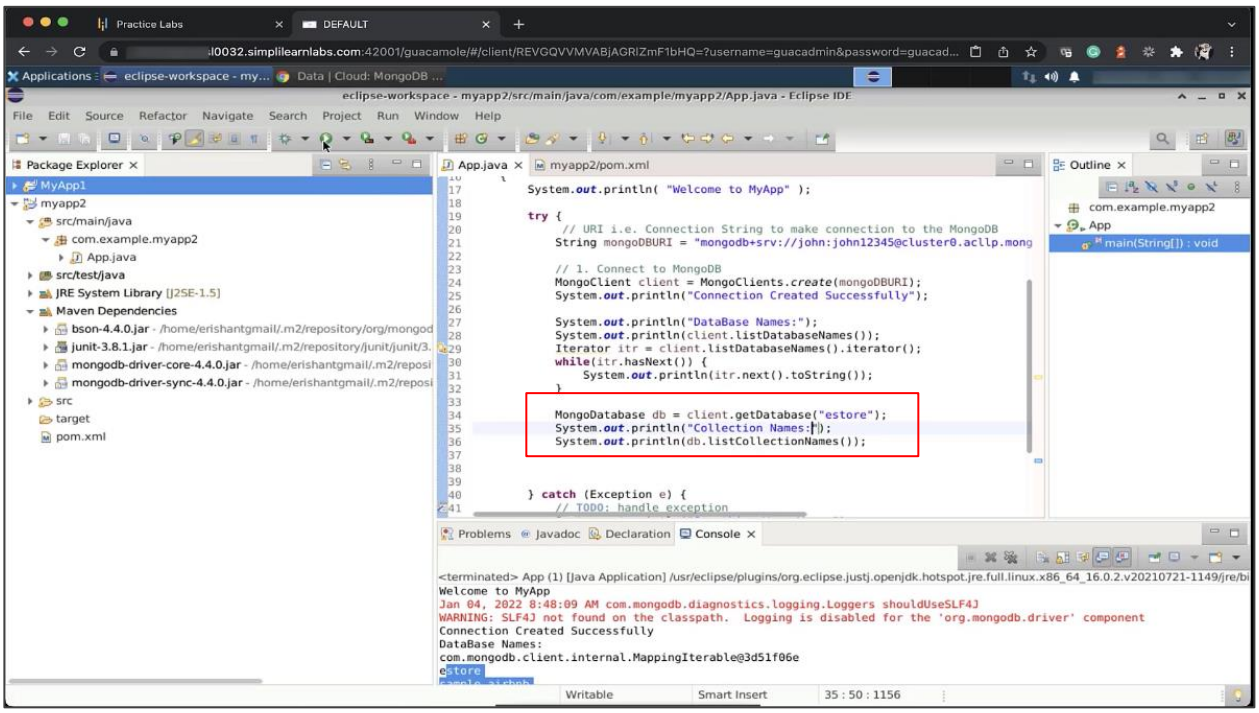
7.9 Test a few more parts and retrieve the documents using the **listDatabaseName** function (from lines 26 to 30) and run the code



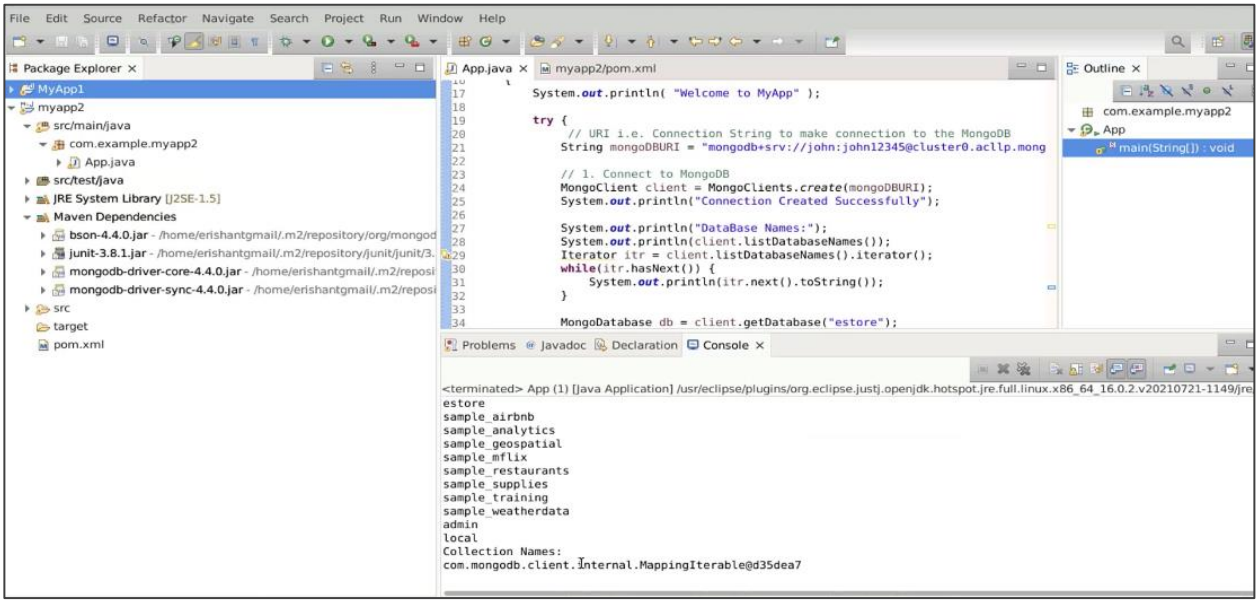


The database names can be seen in the output.

7.10 Now, get the database names and read the collections from them using the **getDatabase** function (from lines 34 to 36). Save and run the code.



You can see the databases in the **Console** tab.



There are two ways to create a project, either a Java project or Maven. In the Maven project, you can configure the dependencies in the pom.xml file, which will automatically add the required dependencies. For a Java project, all the dependencies must be added manually.

By following these steps, you have successfully configured MongoDB with a Java project by creating a Java project and adding JAR files.