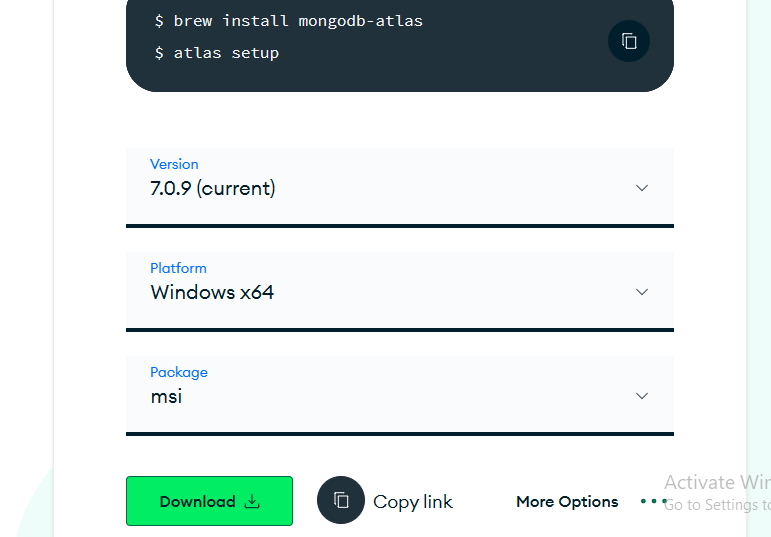
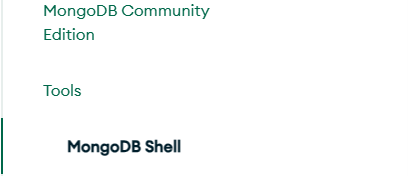
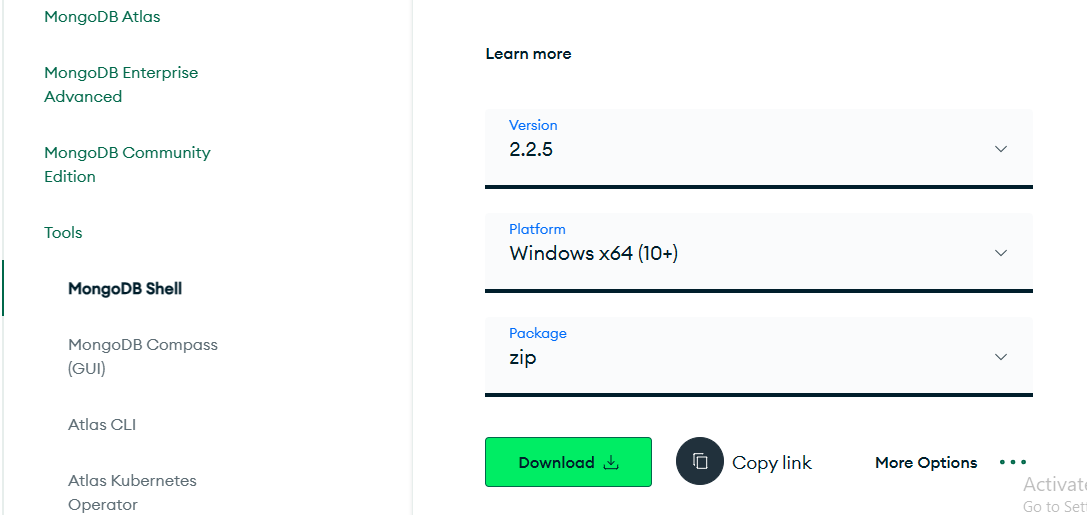
**MONGO DB**

[Download MongoDB Community Server | MongoDB](https://www.mongodb.com/try/download/community) ---link for mongo db download



Select latest version.----then install after installation click on localhost link(default)---then –connect--- then we can create database and collection or we can asscess it by creating using mongoDB Shell





To install mongo db shell ---- download .zip file then extract in the mongodb location after place inside near server and extract then open the cmd in the same loction then continue the commands by the link

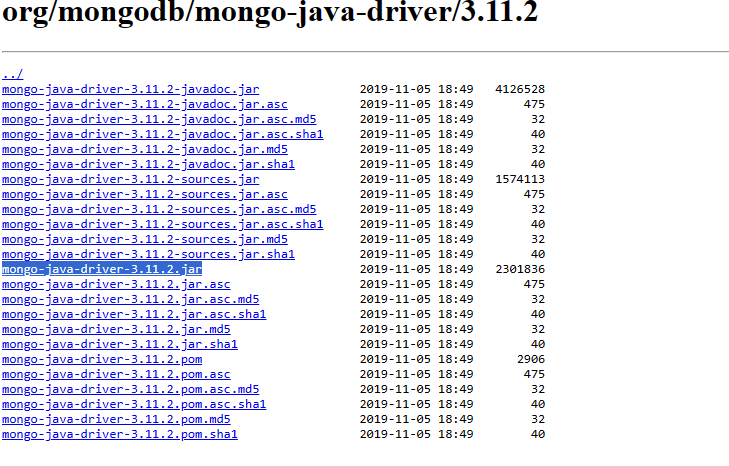
After that copy the bin path add to the **environmental variable**--- system variable ---path –add ---new --- paste the path of mongosh bin directory

Close cmd prompt and open again then type **mongosh**

**To Run in Eclipse: start the mongo db server before starting the ecplise execution the command to start the server is go to the mongodb location open cmd thn type “**mongod”

**then the server starts**

[Central Repository: org/mongodb/mongo-java-driver/3.11.2 (maven.org)](https://repo1.maven.org/maven2/org/mongodb/mongo-java-driver/3.11.2/) ---- link to download jar file



To run MongoDB code in Java within Eclipse, you need to follow these steps:

### Step 1: Download MongoDB Java Driver

First, download the MongoDB Java driver JAR file from the official MongoDB website or Maven repository.

### Step 2: Add MongoDB Java Driver to Eclipse Project

1. Right-click on your Eclipse project.

2. Select "Build Path" > "Configure Build Path...".

3. In the "Libraries" tab, click on "Add External JARs...".

4. Navigate to the location where you downloaded the MongoDB Java driver JAR file, select it, and click "Open".

5. Click "Apply and Close".

### Step 3: Write Java Code to Connect to MongoDB

Here's a simple example of Java code to connect to MongoDB, insert a document, and retrieve documents from a collection:

```java

import com.mongodb.client.MongoClients;

import com.mongodb.client.MongoClient;

import com.mongodb.client.MongoDatabase;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.model.Filters;

import com.mongodb.client.result.InsertOneResult;

import org.bson.Document;

public class MongoDBExample {

public static void main(String[] args) {

// Connect to MongoDB

try (MongoClient mongoClient = MongoClients.create("mongodb://localhost:27017")) {

// Get database

MongoDatabase database = mongoClient.getDatabase("testdb");

// Get collection

MongoCollection<Document> collection = database.getCollection("testCollection");

// Insert document

Document document = new Document("name", "John")

.append("age", 30)

.append("city", "New York");

InsertOneResult result = collection.insertOne(document);

System.out.println("Inserted document id: " + result.getInsertedId());

// Find documents

for (Document doc : collection.find(Filters.eq("city", "New York"))) {

System.out.println(doc.toJson());

}

} catch (Exception e) {

System.err.println("Exception: " + e.getMessage());

}

}

}

```

### Step 4: Run the Java Code

Once you have written the Java code in Eclipse, you can simply run it as a Java application by right-clicking on the class file and selecting "Run As" > "Java Application".

### Step 5: Verify MongoDB Connection

Ensure that MongoDB is running on your local machine on the default port (27017) or adjust the connection URL accordingly. Also, make sure you have a database named "testdb" and a collection named "testCollection" in your MongoDB instance.

By following these steps, you can run MongoDB code in Java within Eclipse. Make sure to handle exceptions appropriately and close resources properly in a real-world application.