Developing Storage Solutions with Amazon Simple Storage Service (S3) (LAB-M04-01)

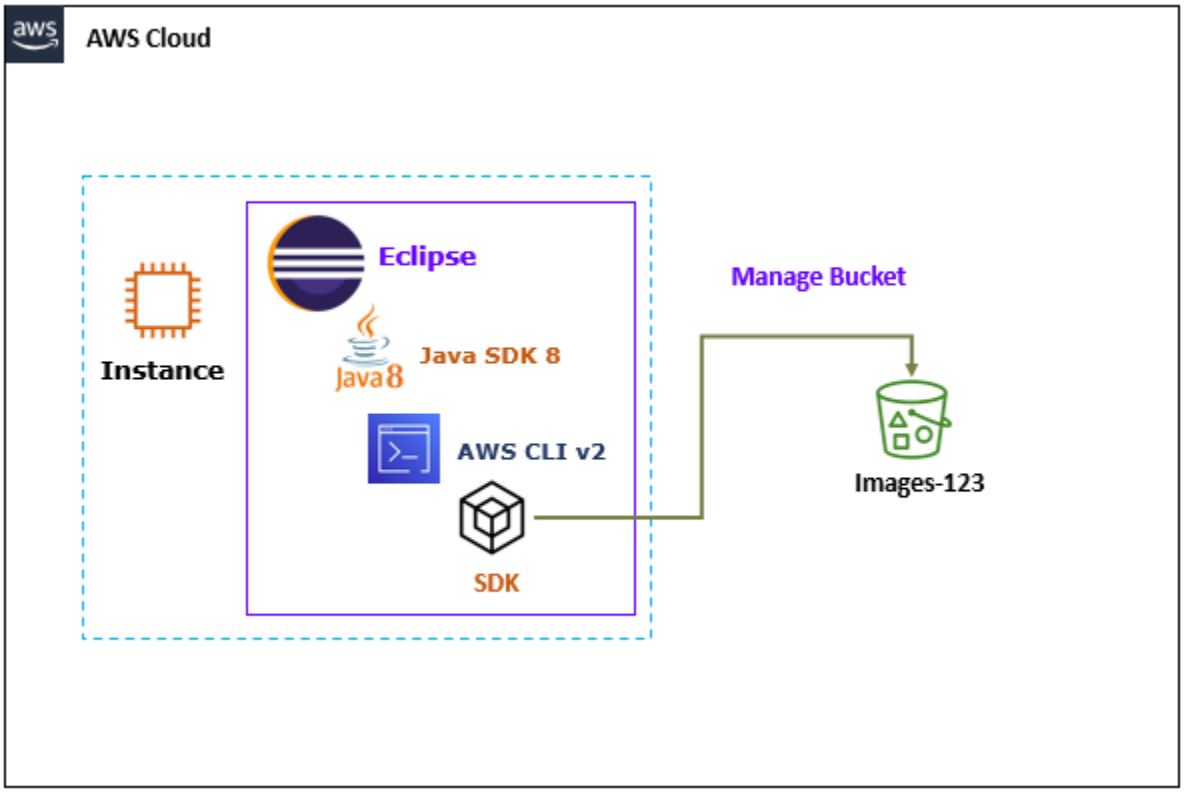
Lab scenario

You're preparing to store binary data in AWS. As a development group, your team has decided to use Java to manage the data from AWS storage programmatically.

Objectives

After you complete this lab, you will be able to:

- Create new Bucket.
- Delete the bucket.
- List the existing Buckets.
- List the buckets Content.
- Upload Object in the bucket.
- Get Object from the bucket.
- Upload Object with metadata in the bucket.
- Update metadata for the existing Object.
- Create Pre-signed URL.
- Get Object using Pre-signed URL.



Task 1: Create IAM User

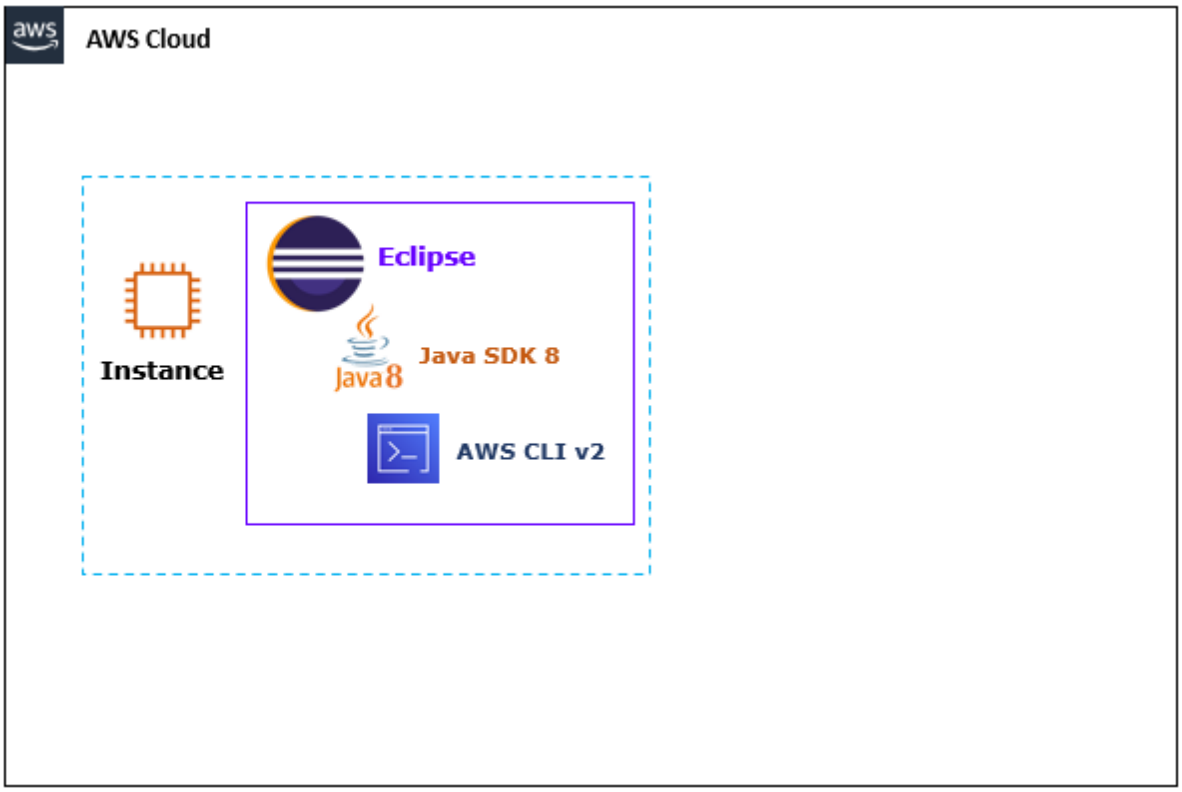
In this task, you will create AWS IAM User with Permission to manage S3.

Step 1: Create IAM User

1. In the **AWS Management Console**, on the **Services** menu, click **IAM**.
2. Select **Users**.
3. Select **Add user**.
    a. **User name**: Write **S3-User**.
        i. **Access type**: Select **Programmatic access**.
        ii. Select **Next: Permissions**.
    b. Select **Attach existing policies**.
        i. Search and Select **AmazonS3FullAccess**.
        ii. Select **Next: Tags**.
        iii. Select **Next: Review**.
    c. Select **Create users**.
    d. Select **Download .csv** file, to download the **Access key Id** & **Secret Access key details** in your local desktop/ laptop.

Task 2: Build Server for Development Environment

In this task, you will build the AWS Virtual machine to build development environment and install the Java, Eclipse and AWS CLI.



Step 1: Create EC2 Instance

4. Choose the **US East (N. Virginia)** region list to the right of your account information on the navigation bar.

5. In the **AWS Management Console**, on the **Services** menu, click **EC2**.

6. Click **Launch Instance**.

   a. In the **Choose Amazon Machine Image (AMI),** choose **Microsoft Windows Server 2019 Base**, click **Select**.

   b. In the **Choose Instance Type**, choose an **t2.medium**, click **Next: Configure Instance Details**.

Note: You can also use **t2.micro**, but the performance will be low to build development environment.

   c. In the **Configure Instance Details**, Select **Next: Add Storage**.

Note: Leave the detail as default.

d. In the **Add Storage**, Select **Next: Add Tags**.

Note: Leave the detail as default.

e. In the **Add Tags,** Select **Add Tag**.

i. **Key Name**: Write **Name**.

ii. **Value**: Write **Dev Instance**.

iii. Choose **Next: Configure security group**.

f. In the **Configure Security Group**.

Note: The wizard automatically defines the launch-wizard-*x* security group and creates an inbound rule to allow you to connect to your instance over RDP (**port 3389**).

i. Choose **Create a new security group**.

- **Security group name**: Write **Dev Instance SG**.
- **Description**: Write **Dev Instance Security Group**.

ii. Click **Review and Launch**.

g. Click **Launch**.

h. In the ***Select an existing key pair or create a new key pair***.

i. In the popover, select **Choose an existing key pair** and select **My-Dev-LAB-KP**.

ii. **Select I acknowledge ...**

iii. Click **Launch Instances**.

Step 2: Check the Dev Instance Status

7. In the **AWS Management Console**, on the **Services** menu, click **EC2**.
8. Click **Instance**.
9. Select **Dev Instance**.
    a. **Wait** for the **Instance State** to change to **Running** **state**.
    b. **Wait** for the **Status check** to change to **2/2 checks passed**.

Step 3: Copy the Dev Instance Public IP address

10. In the **AWS Management Console**, on the **Services** menu, click **EC2**.
11. Click **Instances**.
12. Select **Dev Instance**.
    a. ***Go below*** in the console and click on **Networking**.
    b. **Copy** the **Public IP address**.

Step 4: Generate Windows Password of Dev Instance

13. Select **Dev Instance**.
    a. Select **Actions**.
    b. Select **Security**.
    c. Select **Get Windows Password**.
        i. **Browse**: Navigate and Select **My-Dev-LAB-KP**.pem key pair.
        ii. Click on **Decrypt Password**.

Note: Windows will pop-up with **user name** and **password**.

Note: **Copy** the **user name** and **password** in the **Notepad**.

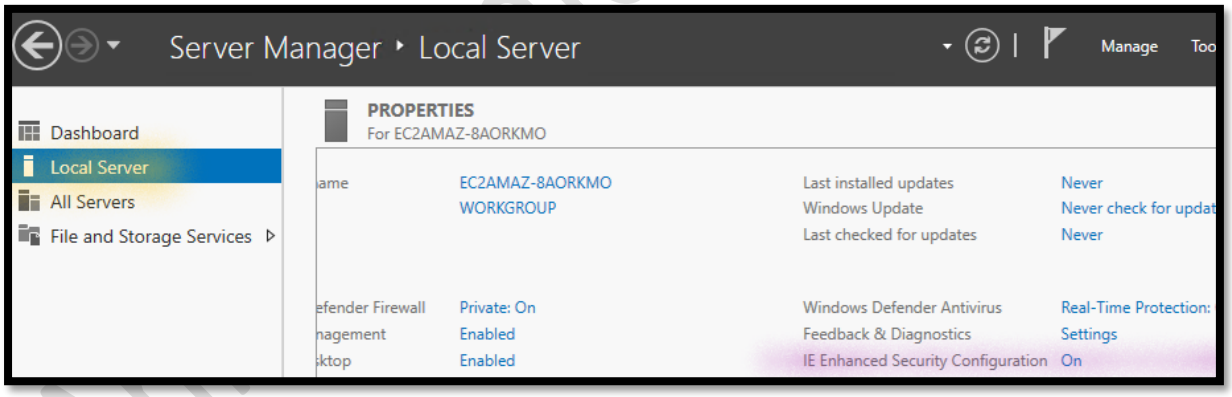
    d. Select **Close**.

Step 5: Remote Desktop to Dev Instance

14. From the **local Desktop/ Laptop** (Windows), right click on **Start** & **Run**.
15. In the open, write **mstsc**, press **Ok**.
    a. **Type** the **Public IP Address** of the **Dev Instance**.
    b. Click **Connect**.
    c. **Type** the **Username** and **Password** of the **Dev Instance** which you copied in the previous step and click **Ok**.
    d. Click on **Yes** to confirm this connection, if prompted with the security message.

Step 6: Install the Java SE Development Kit 8

16. From the **Dev Instance** (Windows 2019), right click on **Start** & **Run**.
    a. Go to **Start menu**, open **Server manager**.
    b. Select **Local Server**.
    c. Click in **On** showing against ***IE Enhanced Security Configuration***.



    d. Select **Off** in Administrator and Select **Ok**.
    e. **Refresh** you screen & now you can see **Off** showing against ***IE Enhanced Security Configuration***.
    f. **Close** the **Server manager**.
17. **Download** and **Install** the ***Java SE Development Kit 8*** for **Windows x64**.

Note: Use the below URL to download the **Java SE Development Kit 8**.

https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html

Solaris x64	92.66 MB	jdk-8u281-solaris-x64.tar.gz
Windows x86	154.69 MB	jdk-8u281-windows-i586.exe
Windows x64	166.97 MB	jdk-8u281-windows-x64.exe

Note: If you don't have Oracle account to download the JAVA SDK, then **create account** and download the Java SDK.

Note: **Wait**, till **Java SE Development Kit 8** install **succesfully**.

Step 7: Check the Java SE Development Kit 8 version

18. **From** the **Dev Instance**, right click on **Start** & **Run**.
19. In the **Open**, write **cmd**, press **Ok**.
    a. From the ***command line interpreter***, write **java -version**, press **Enter**.

Note: You can see the **Java SDK installed version**.

```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.17763.1817]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>java -version
java version "1.8.0_281"
Java(TM) SE Runtime Environment (build 1.8.0_281-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.281-b09, mixed mode)

C:\Users\Administrator>_
```

Step 8: Install the Eclipse IDE

20. **Download** and **Install** the ***Eclipse IDE***.

Note: Use the below URL to download the **Eclipse IDE**.

https://www.eclipse.org/downloads/

a. In the **Eclipse Installer**, Select ***Eclipse IDE for Java Developers***.

Note: **Wait**, till ***Eclipse IDE*** install **succesfully**.

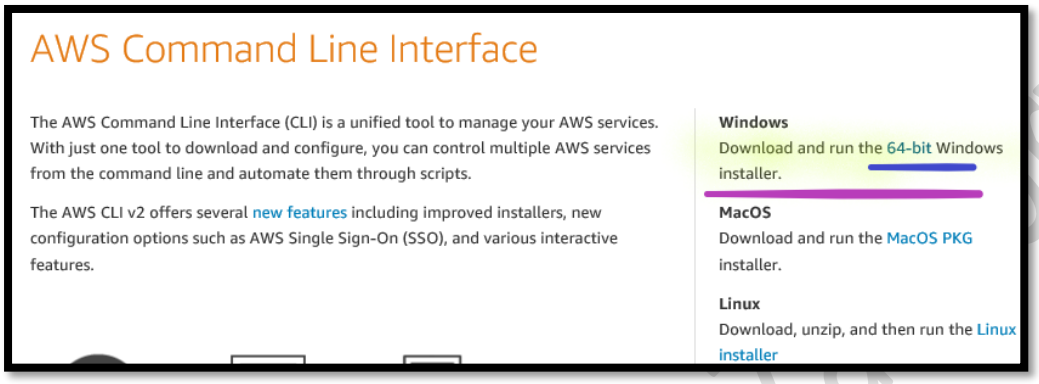
Note: **Don't** Launch the Eclipse IDE.

Step 9: Install the AWS CLI v2

21. **Download** and **Install** the ***AWS CLI v2***.

Note: Use the below URL to download the **AW CLI v2**.

https://aws.amazon.com/cli/



Note: **Wait**, till ***AWS CLI v2*** install **succesfully**.

Step 10: Check the AWS CLI version

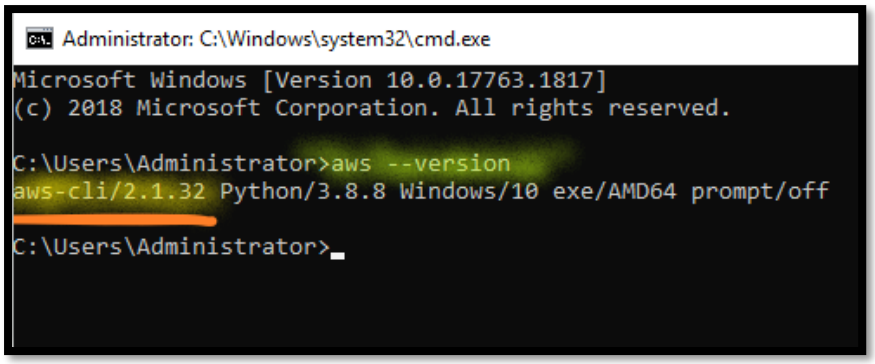
22. **From** the **Dev Instance**, right click on **Start** & **Run**.

23. In the **Open**, write **cmd**, press **Ok**.

    a. From the ***command line interpreter***, write **aws --version**, press **Enter**.

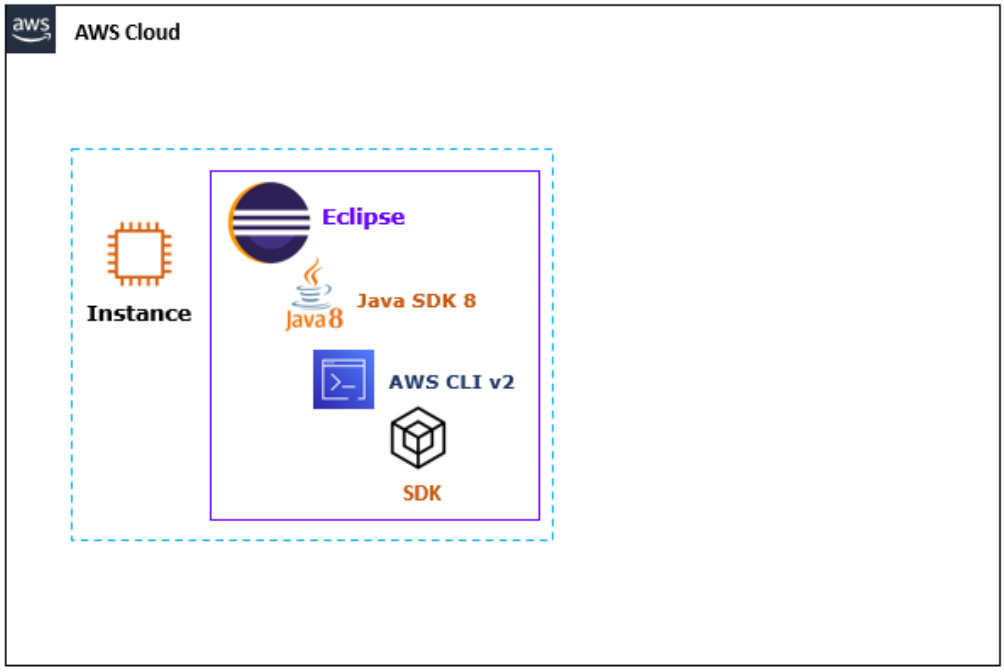
Note: You can see the **AWS CLI version**.

Note: If you can't see the version, **restart the virtual machine**.



Task 3: Create Maven Project

In this task, you will create Maven project to manage S3 programmatically.



Step 1: Copy the Java source code in the Dev Instance

24. **Unzip** the **LAB-04-01-JAVA-Code.zip** (**Java code**).

Note: **Lab-04-01-JAVA-Code.zip** code file is available with the Lab manual.

Note: **Review** the code after opening with the **Notepad**.

25. **Return** to the **Dev Instance** (Windows 2019).

26. **From** the **Dev Instance**, right click on **Start** & **Run**.

27. In the **Open**, write **C:**, press **Ok**.

    a. Create **S3-Code** folder in ***C drive***.

    b. Open the **S3-Code** folder.

    c. **Copy** the **Code files** in the **S3-Code** folder.

Note: You need to copy all the **three files** and **one folder**. Don't copy the zip file.

Step 2: Configure the Credentials and Configuration Settings

28. **From** the **Dev Instance**, right click on **Start** & **Run**.

29. In the **Open**, write **cmd**, press **Ok**.

    a. From the ***command line interpreter***, write **aws configure**, press **Enter**.

        i. **AWS Access Key ID**: **Type** the IAM User **S3-User** **access key** and **enter** to continue.

        ii. **AWS Secret Access Key**: **Type** the IAM User **S3-User** **secret access key** and **enter** to continue.

Note: You can **copy** the **access key** and **secret access key** of the IAM user **S3-User** from **.csv file** which you have downloaded in the previous step.

        i. **Default region name**: **Type** **us-east-1** and **enter** to continue.

        iii. **Default output format**: **Type** **json** and **enter** to continue.

```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.17763.1817]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>aws configure
AWS Access Key ID [None]: AKIAXK3IQNIIS62TQV6F
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json

C:\Users\Administrator>_
```

30. **From** the **Dev Instance**, right click on **Start** & **Run**.

    a. In the **Open**, write **C:\Users\Administrator**, press **Ok**.

    b. Open the, **.aws** folder.

    c. Open the **Credentials** file in **Notepad**.

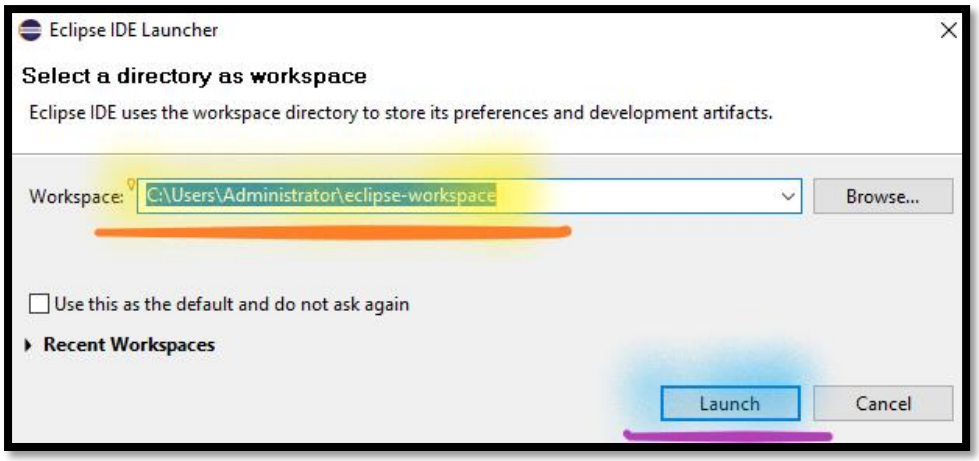
Note: You can see the **access key** and **secret access key** details.

    d. Open the **Config** file in **Notepad**.

Note: You can see the **region** and **output** format details.

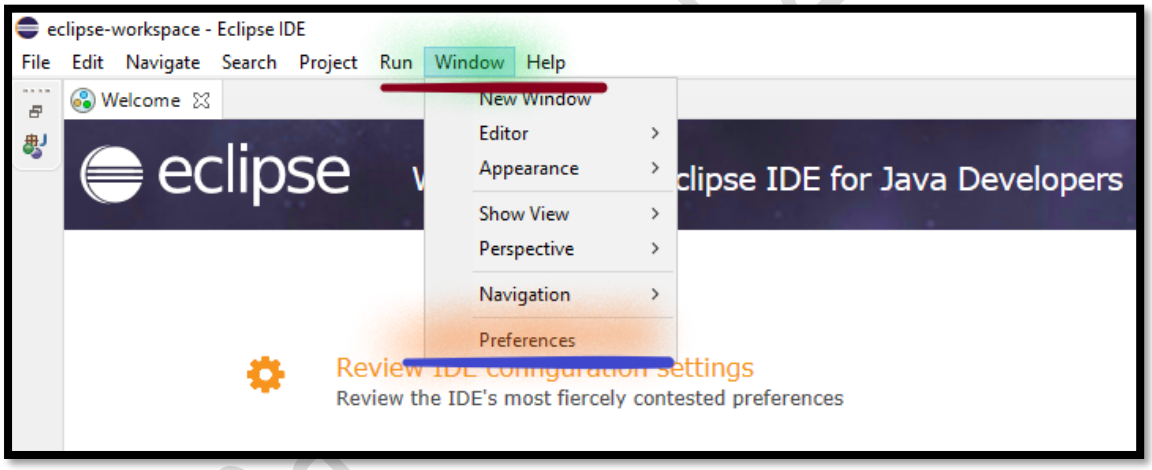
Step 3: Launch the Eclipse IDE

31. Select and Open the **Eclipse IDE for Java Developers**.
32. In the **Eclipse Ide Launcher**:
    a. **Workspace**: Leave the **default path**.
    b. Select the **Launch**.



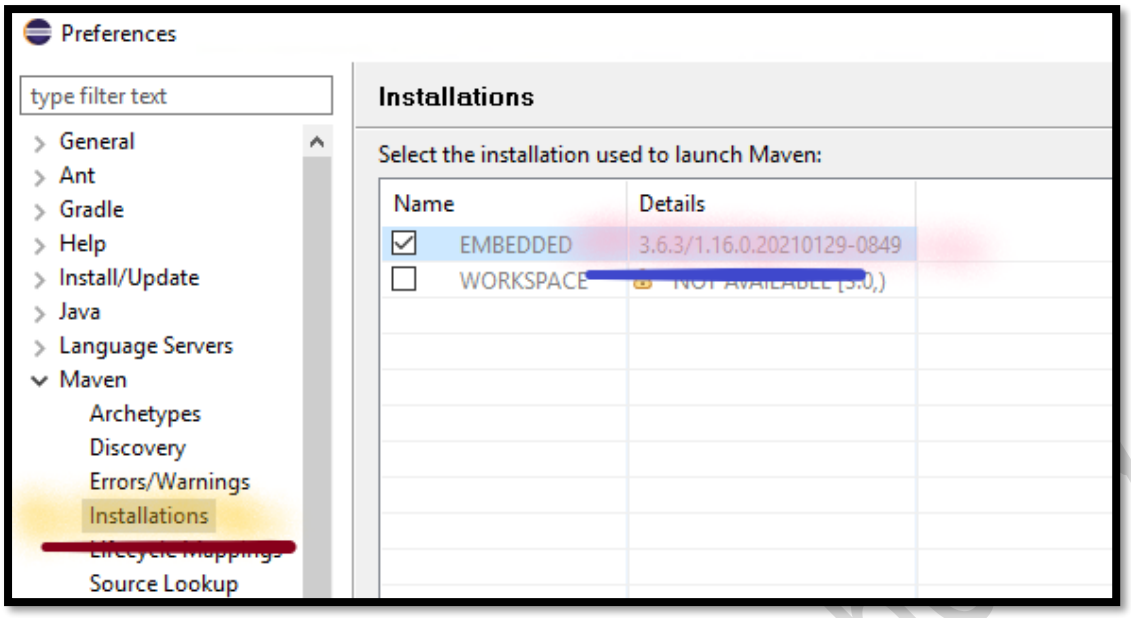
Step 4: Verify the Maven Installation

33. **From** the **Eclipse IDE**.
    a. Select the **Windows**.
    b. Select the **Preference**.



    c. **From** the **Preference page**:
        i. Expand the **Maven**.
        ii. Select the **Installations**.

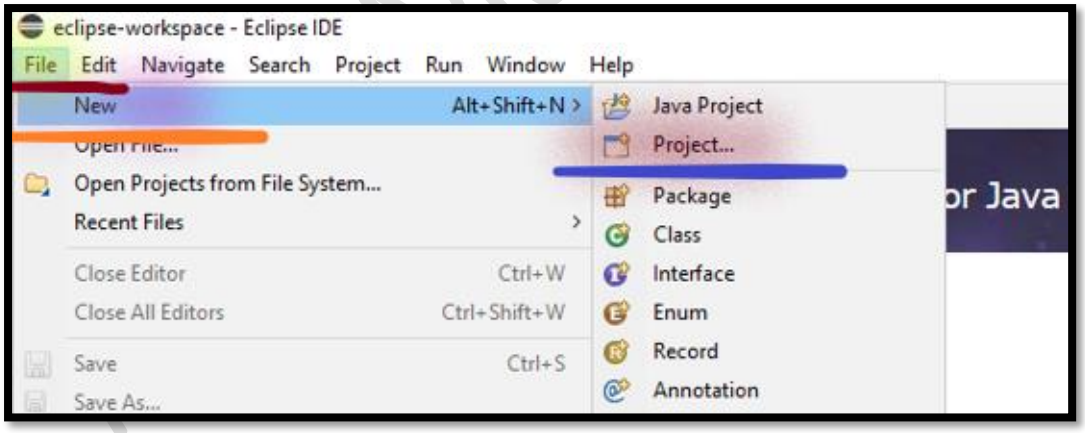
Note: You can see the **Maven version**.



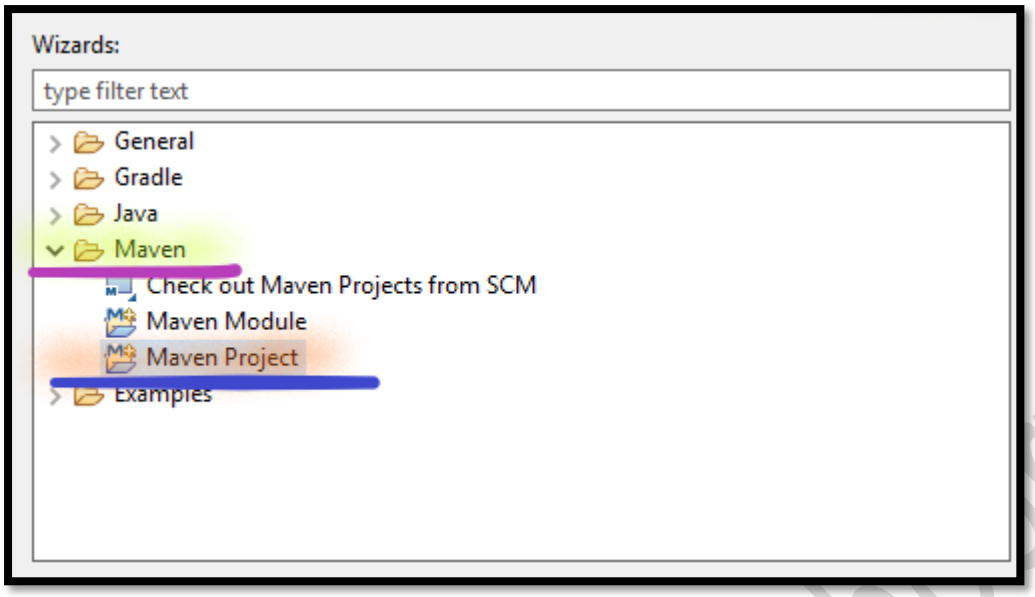
i. Select the **Cancel** to ***close the page***.

Step 5: Create the Maven Project

34. **From** the **Eclipse IDE**.
    a. Select the **File**.
    b. Select the **New**.
    c. Select **Project**.



d. In the **Select a Wizard** page:
    i. **Expand** the **Maven**.
    ii. Select the **Maven Project**.

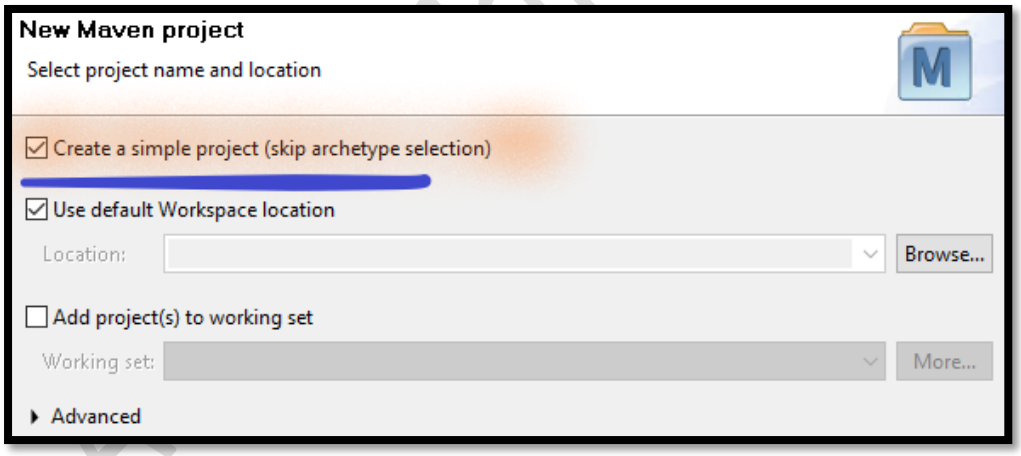


iii. Select the **Next**.

e. In the **Select project name and location** page:

i. Select the **Create a Simple project**.

Note: Leave the other details as default.



ii. Select the **Next**.

f. In the **Enter the group id for Artifact** page:

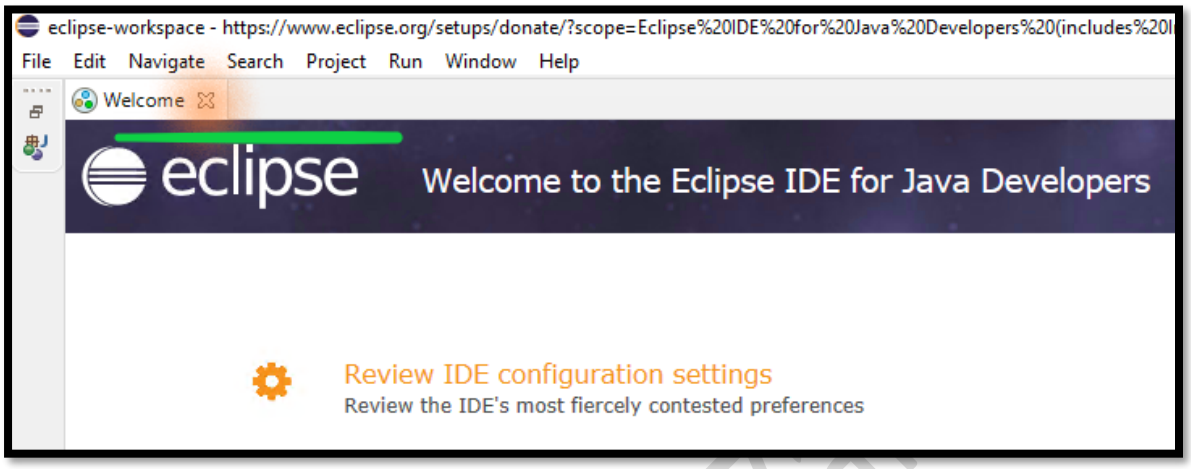
i. **Group Id**: Write **com.aws.s3**.

ii. **Artifact Id**: Write **awss3m04**.

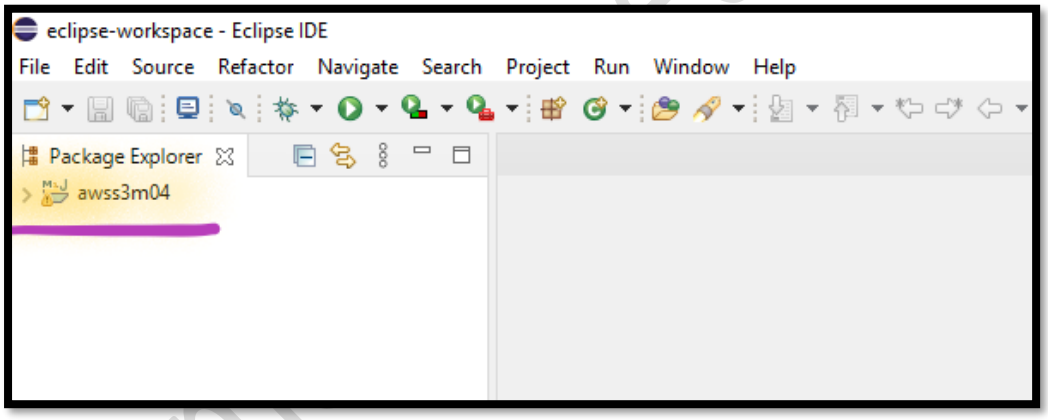
Note: Leave the other details as default.

iii. Select **Finish**.

Note: Close the **Welcome to the Eclipse IDE for Java Developers** page.



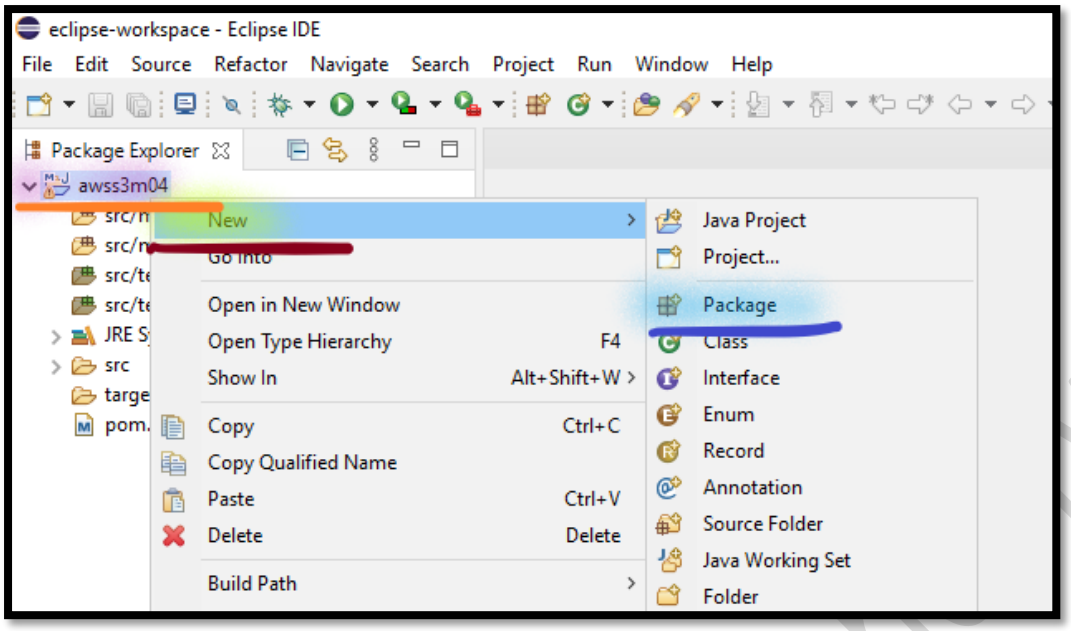
Note: You can see the Package explorer and your **awss3m04** project.



Note: If you can't see the Package explorer and your project, **Close the Welcome to Eclipse** page and Select **Windows --> Show view --> project explorer**.

Step 6: Create the Package

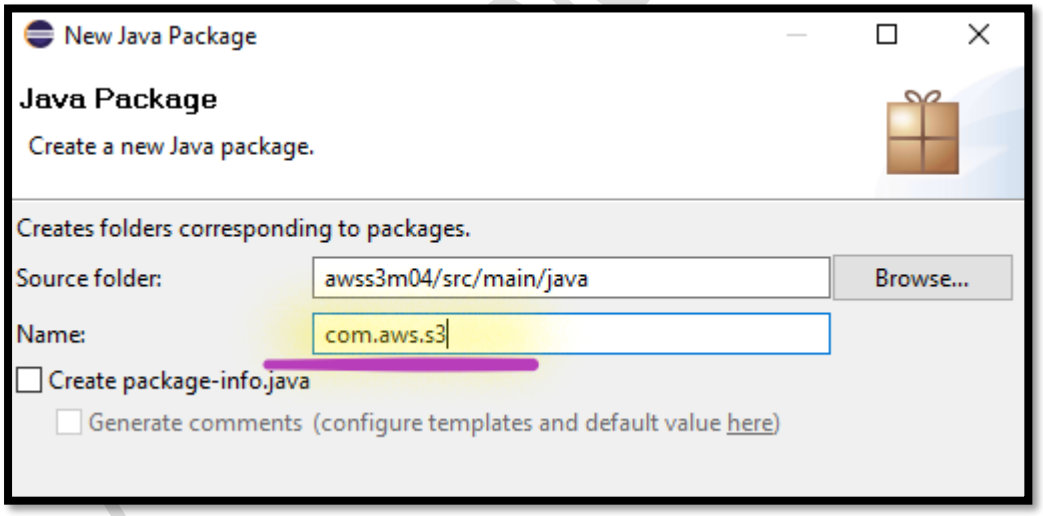
35. **Expand** the Java Project ***awss3m04***.
    a. **Right-click** on the **awss3m04** Java project.
    b. Select **New**.
    c. Select **Package**.



d. In the **Create a new Java package** page:

i. **Name**: Write **com.aws.s3**.

Note: Leave the other details as default.



ii. Select **Finish**.

Step 7: Create the File in the Package

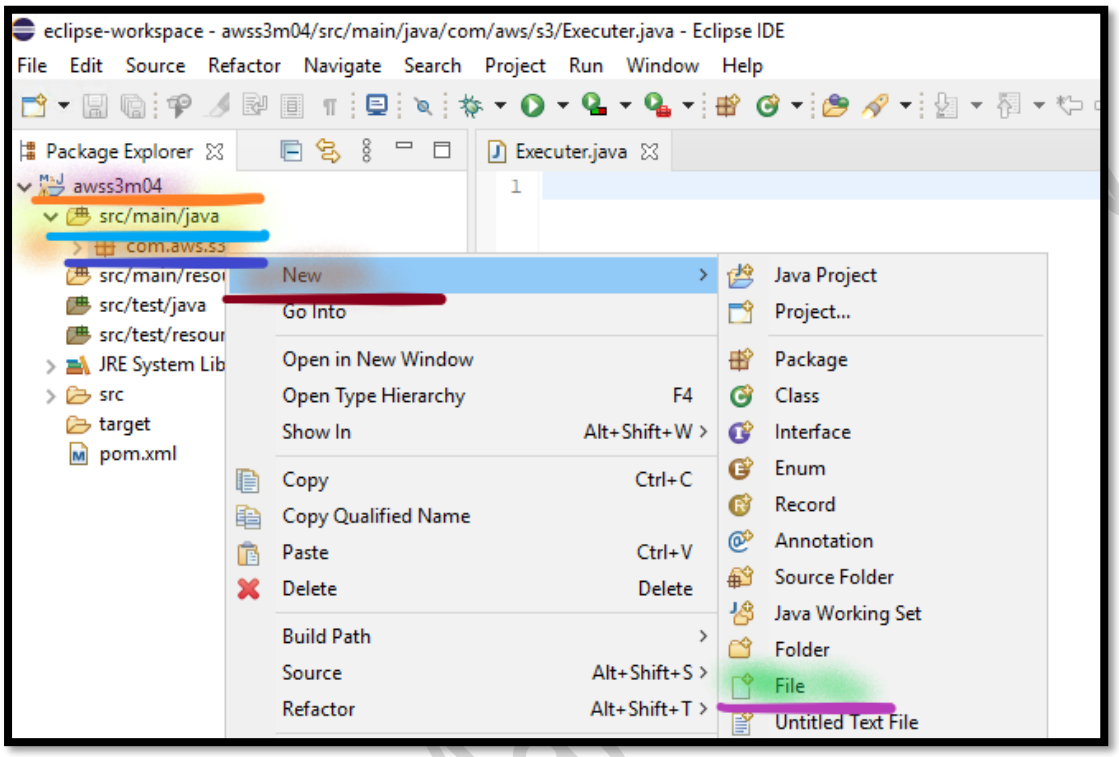
36. **Expand** the Java Project ***awss3m04***.
37. **Expand** the Resource path ***src/main/java***.
38. **Select** the Java Package ***com.aws.s3***.

Create Executor.java file

a. **Right-click** on the **com.aws.s3** Java package.

b. Select **New**.

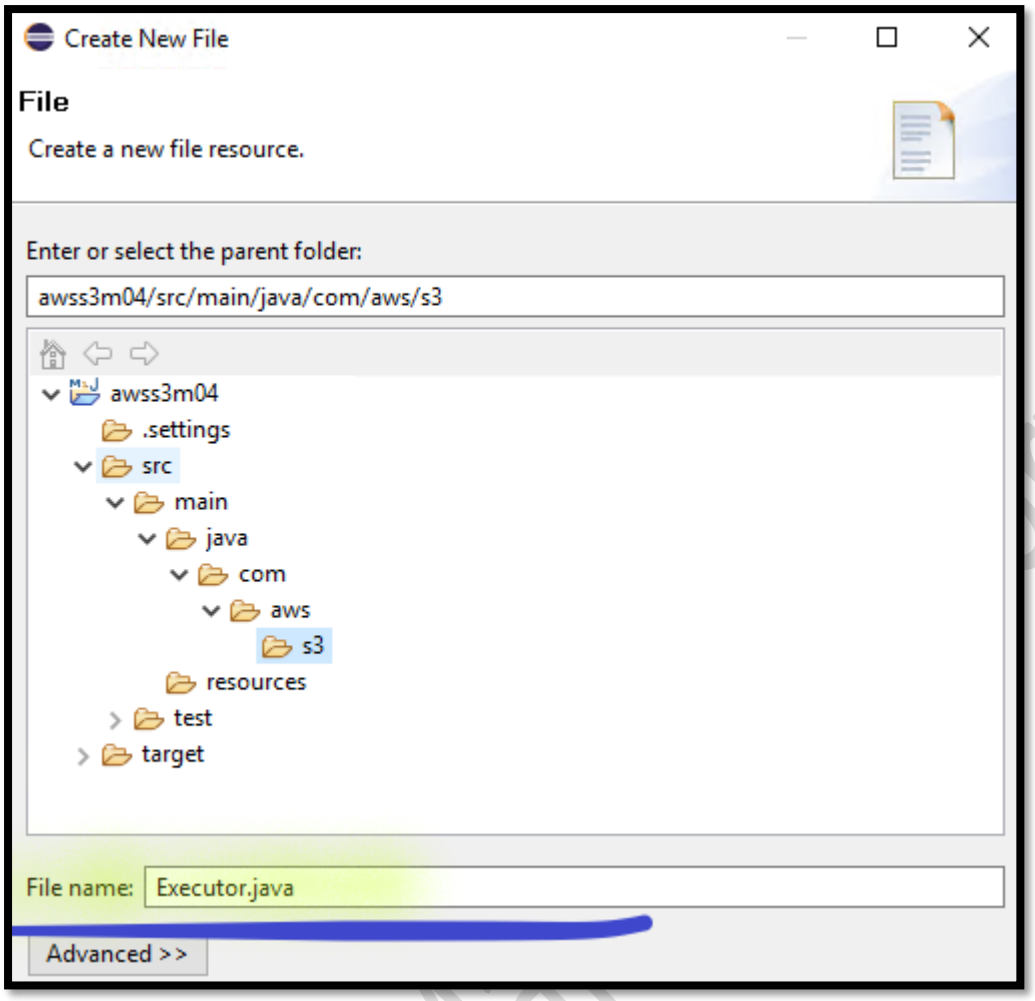
c. Select **File**.



d. In the **Create a new file resource** page:

   i. **File name**: Write **Executor.java**.

Note: Ensure that in the executor, **E** shoudl be **Capital**.



ii. Select **Finish**.

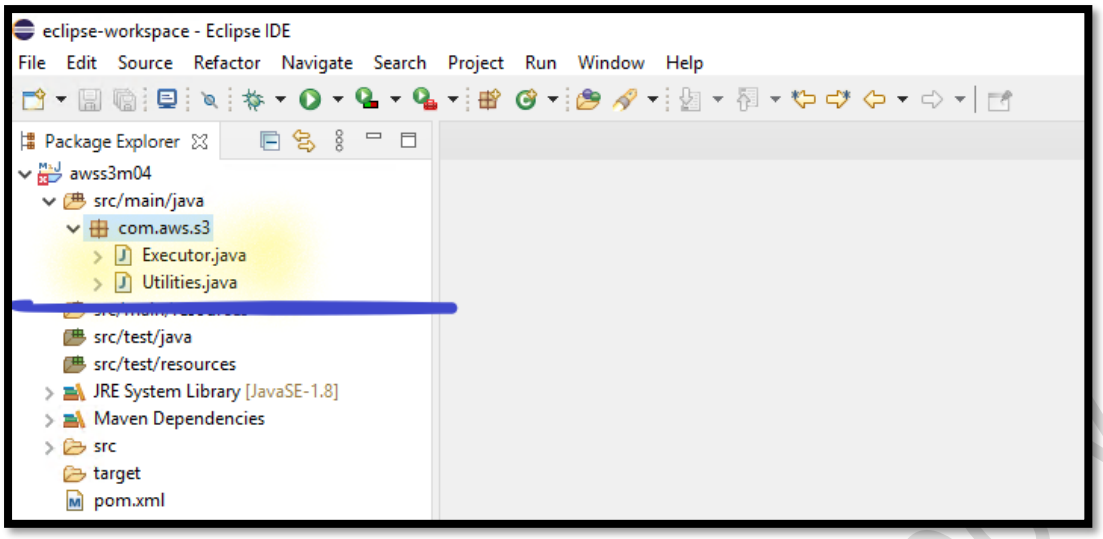
Create Utilities.java file

39. **Right-click** on the **com.aws.s3** Java package.
    a. Select **New**.
    b. Select **File**.
    c. In the **Create a new file resource** page:
        i. **File name**: Write **Utilities.java**.

Note: Ensure that in the utilities, **U** should be **Capital**.

ii. Select **Finish**.

Note: You can see the **Executor.java** and **Utilities.java** under Java package.



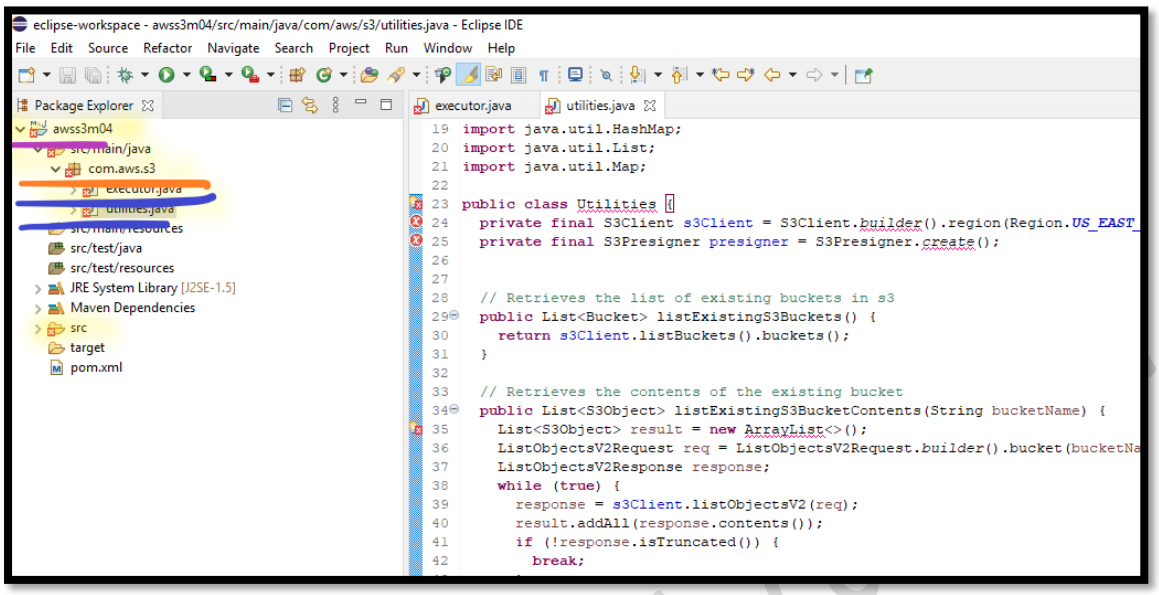
Step 8: Update the Java Code

40. **Double-click** on the Java file ***Executor.java***.
    a. **Paste** the ***Code*** from ***Executor.java*** file, which you copied in the **S3-Code folder** in the ***Dev Instance***.
    b. **From** the **Eclipse IDE**.
        i. Select the **File**.
        ii. Select the **Save**.

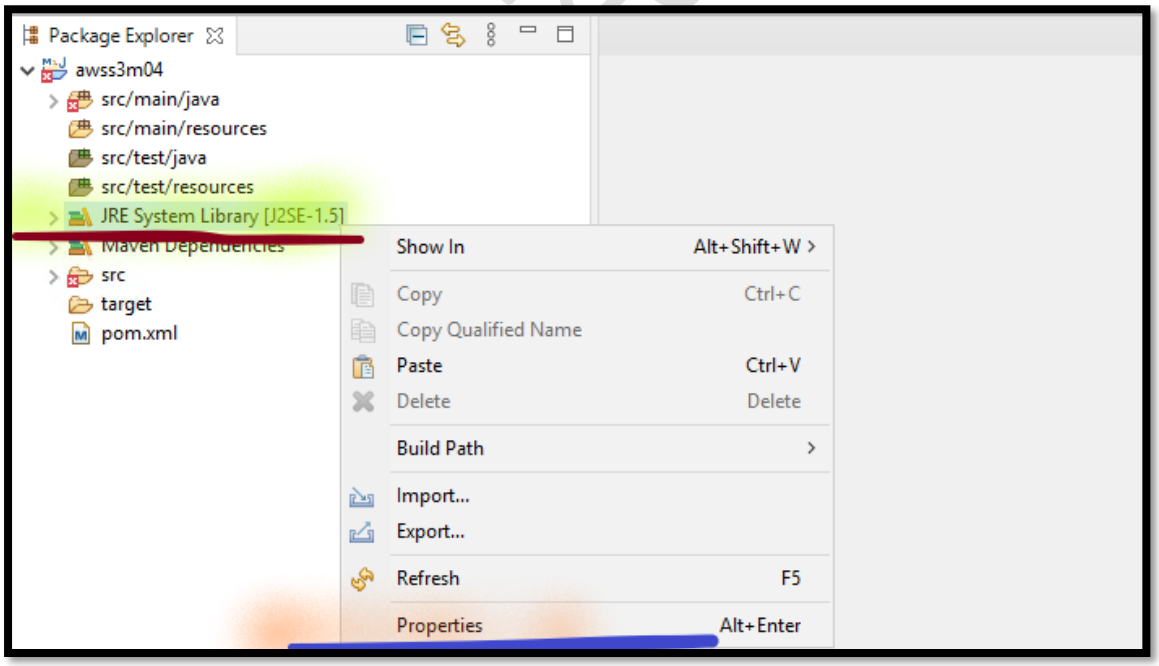
41. **Double-click** on the Java file ***Utilities.java***.
    a. **Paste** the ***Code*** from ***Utilities.java*** file, which you copied in the **S3-Code folder** in the ***Dev Instance***.
    b. **From** the **Eclipse IDE**.
        i. Select the **File**.
        ii. Select the **Save**.

42. **Double-click** on the ***pom.xml***.
    a. **Remove** the ***existing code***.
    b. **Copy** the ***Code*** from ***pom.xml*** file, which you copied in the **S3-Code folder** in the ***Dev Instance***.
    c. **From** the **Eclipse IDE**.
        i. Select the **File**.
        ii. Select the **Save**.

Note: You can see the **Error** against **Executor.java** and **Utilities.java** under Java package.

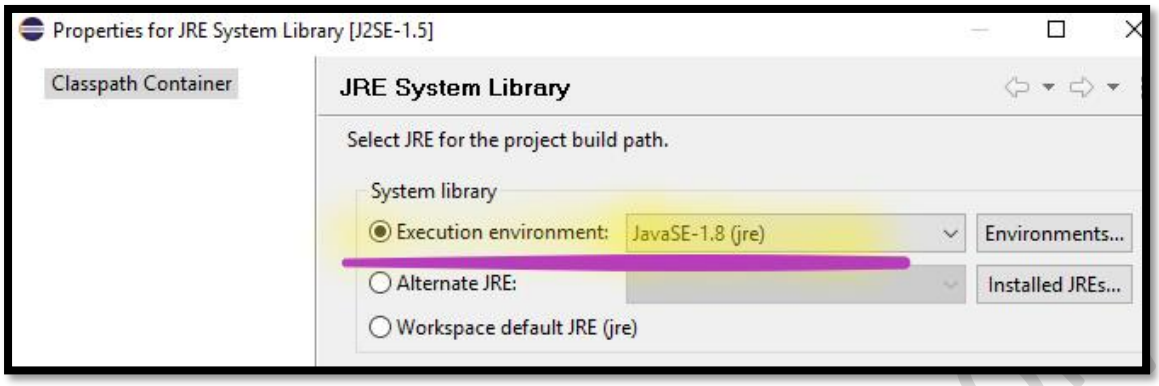


43. **Right-click** on the ***JRE System Library***.
    a. Select **Properties**.



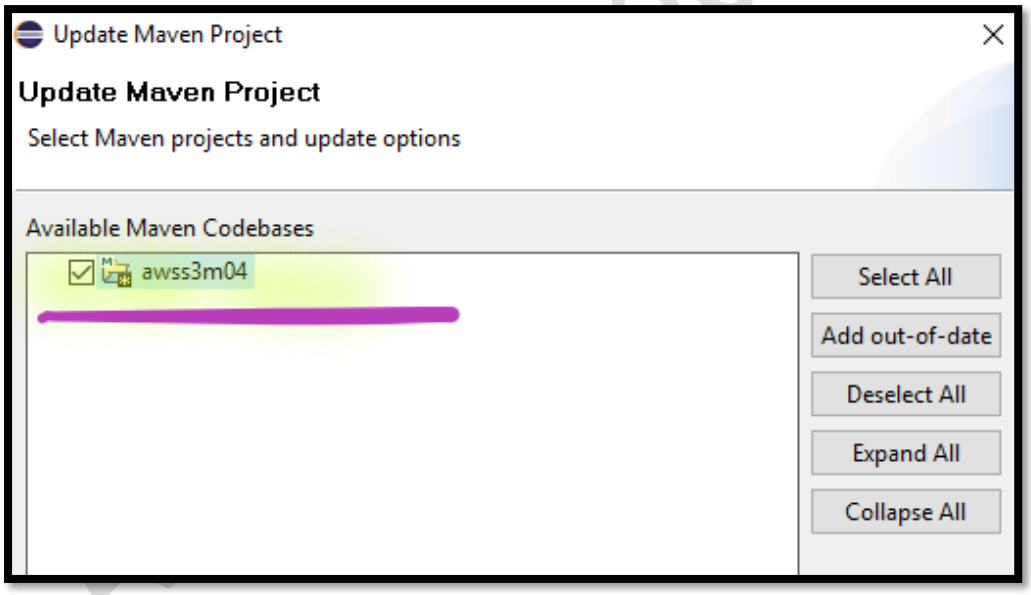
i. **Execution Environment**: Dropdown and Select **JavaSE-1.8 (jre)**.

Note: Leave the other details as default.



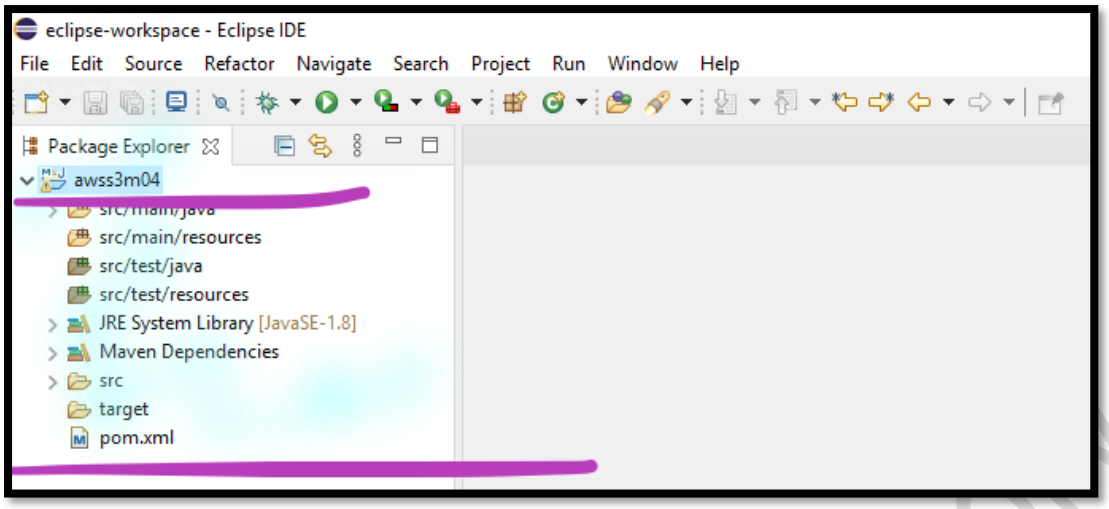
ii. Select **Apply and Close**.

44. **Right-click** on the Java Project ***awss3m04***.
    a. Select **Maven**.
    b. Select **Update project**.



c. Select **Ok**.

Note: Ensure that you **don't see any errors** in the Java project.



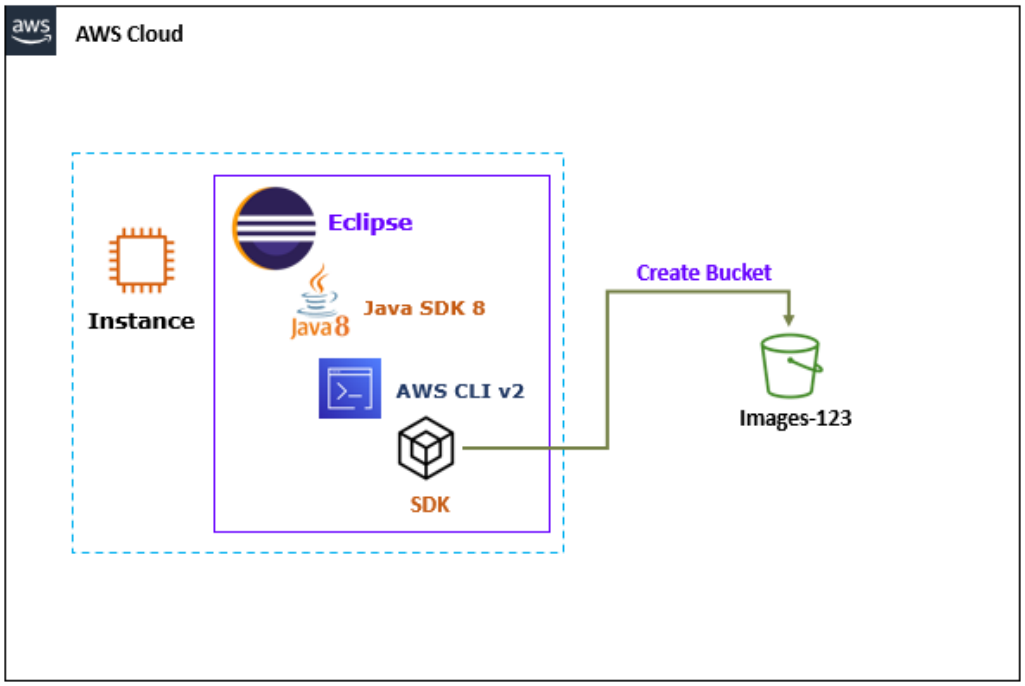
Task 4: Manage AWS S3 from Eclipse

In this task, you will manage AWS S3 from Eclipse using Java.

Step 1: Run the Code

45. **Expand** the Java Project ***awss3m04***.
46. **Expand** the Resource path ***src/main/java***.
47. **Expand** the Java Package ***com.aws.s3***.
    a. **Double-click** on the Java file ***Executor.java***.

Create new Bucket



48. You can see the ***createBucket*** method in the ***line no. 20***.

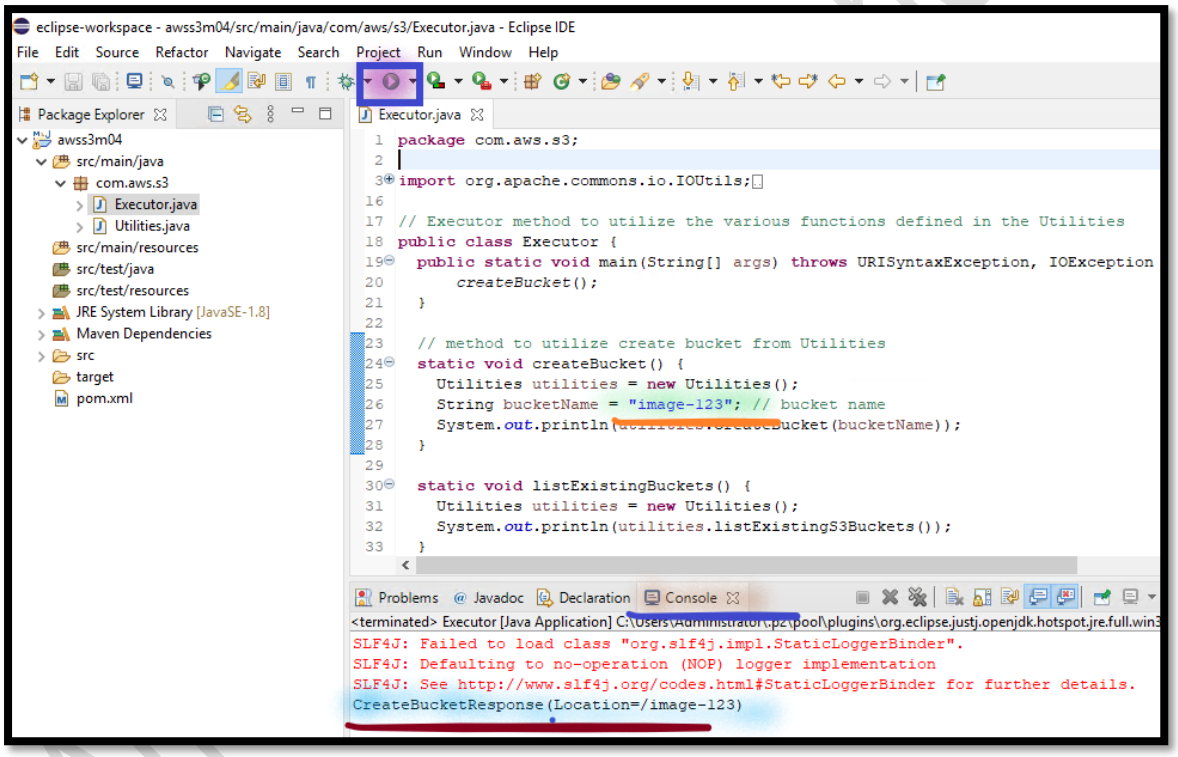
a. **Replace** the **BUCKET_NAME** with the **demo-123**, bucket name you want to create in the ***line no. 25***.

Note: **Don't Replace** the start and end quote (" ").

b. **From** the **Eclipse IDE**, Select the **File** and select the **Save**.

c. **Execute** the **Run Executor**.

Note: If bucket created succesfully, in the **Console**, you will see the **CreateBucketResponse**.



Note: If you are getting error **bucket already exist** in console, **Replace 123** to make the bucket name unique.

```
Executor.java
1  package com.aws.s3;
2
3⊕ import org.apache.commons.io.IOUtils;
16
17 // Executor method to utilize the various functions defined in the Utilities
18 public class Executor {
19⊖   public static void main(String[] args) throws URISyntaxException, IOException {
20        createBucket();
21    }
22
23⊖   static void createBucket() {
24      Utilities utilities = new Utilities();
25      String bucketName = "demo-12389"; // bucket name
26      System.out.println(utilities.createBucket(bucketName));
27    }

Problems  @ Javadoc  Declaration  Console
erminated> Executor [Java Application] C:\Users\Administrator\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0
LF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
LF4J: Defaulting to no-operation (NOP) logger implementation
LF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
xception in thread "main" java.lang.RuntimeException: The bucket demo-12389 already exists
        at com.aws.s3.Utilities.createBucket(Utilitie
        at com.aws.s3.Executor.createBucket(Executor.java:26)
        at com.aws.s3.Executor.main(Executor.java:20)
```

Create second Bucket

49. You can see the ***createBucket*** method in the ***line no. 20***.

    a. **Replace** the **demo-123** with the **images-123**, bucket name you want to create in the ***line no. 25***.

    b. **From** the **Eclipse IDE**, Select the **File** and select the **Save**.

    c. **Execute** the **Run Executor**.

Note: If bucket created succesfully, in the **Console**, you will see the **CreateBucketResponse**.

Note: If you are getting error **bucket already exist** in console, **Replace 123** to make the bucket name unique.

List the existing Buckets

50. **From** the **Executor.java**.

    a. **Replace** the ***createBucket*** method with the **listExistingBuckets** method in the ***line no. 20***.

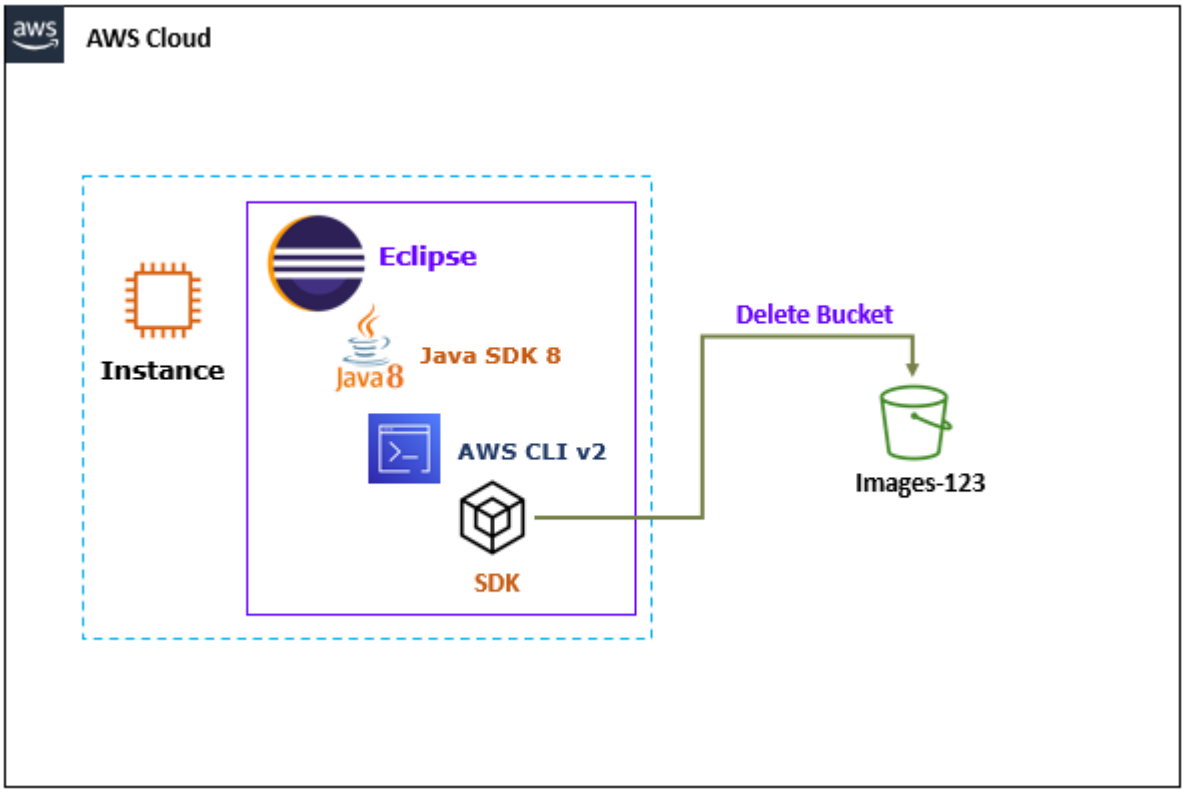
Note: Ensure there should be no space between **listExistingBuckets** and start & end bracket **()**. It should be like **listExistingBuckets()**.

    b. **From** the **Eclipse IDE**, Select the **File** and select the **Save**.

    c. **Execute** the **Run Executor**.

Note: In the **Console**, you will see the Bucket details with **Bucket Name** and **Bucket Creation Date** for **demo-123** and **images-123** buckets.

Delete the Buckets



51. **From** the **Executor.java**.

    a. **Replace** the ***createBucket*** method with the ***deleteBucket*** method in the ***line no. 20***.

b. **Replace** the **BUCKET_NAME** with the **demo-123**, bucket name (***which you have created in the previous step***) in the ***line no. 31***.

c. **From** the **Eclipse IDE**, Select the **File** and select the **Save**.

d. **Execute** the **Run Executor**.

List the existing Buckets

52. **From** the **Executor.java**.

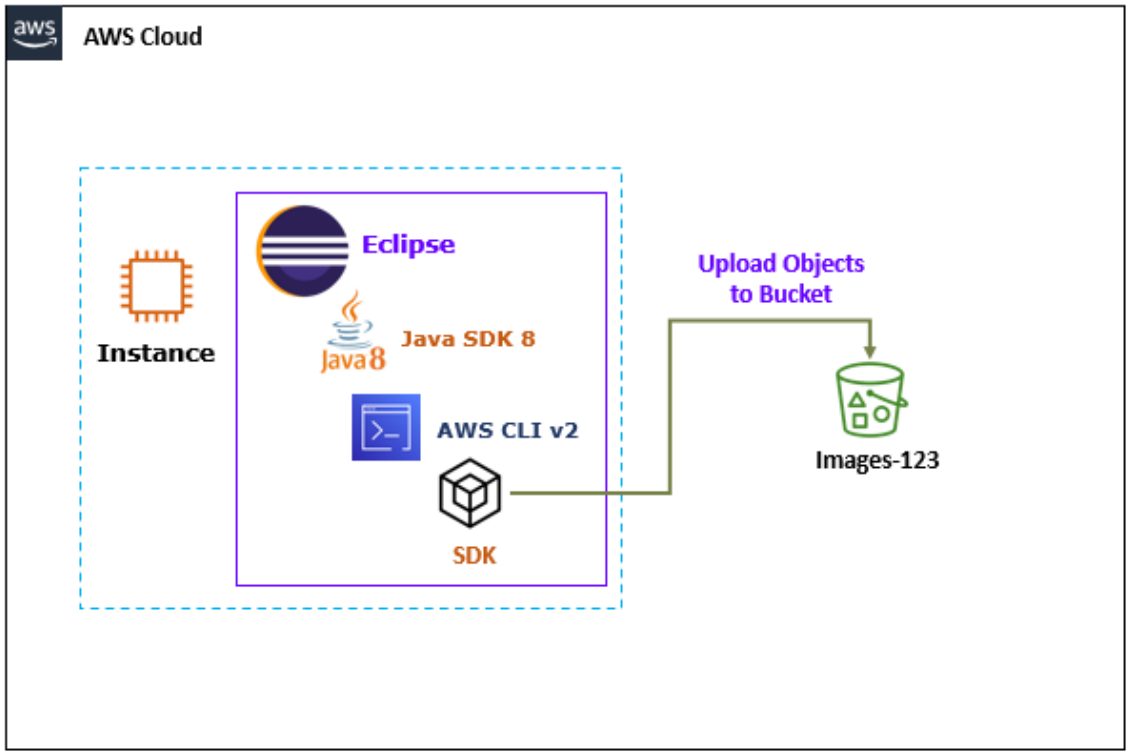
a. **Replace** the ***deleteBucket*** method with the **listExistingBuckets** method in the ***line no. 20***.

b. **From** the **Eclipse IDE**, Select the **File** and select the **Save**.

c. **Execute** the **Run Executor**.

Note: In the **Console**, you will see only the **images-123** bucket details.

Upload new Object



53. **From** the **Executor.java**.

   a. **Replace** the ***listExistingBuckets*** method with the **uploadObject** method in the ***line no. 20***.

   b. **Replace** the **BUCKET_NAME** with the **images-123**, bucket name (which you have created in the previous step) in the ***line no. 43***.

Note: In the code, we have already mentioned **upload file path**, (**File01.txt**) which you have copied in the S3-Code folder in C drive.

   c. **From** the **Eclipse IDE**, Select the **File** and select the **Save**.

   d. **Execute** the **Run Executor**.

Note: If Object uploaded succesfully, in the **Console**, you will see the **PutObjectResponse**.

List Bucket Objects

54. **From** the **Executor.java**.

   a. **Replace** the ***uploadObject*** method with the ***listExistingS3BucketContents*** method in the ***line no. 20***.

   b. **Replace** the **BUCKET_NAME** with the **images-123**, bucket name (which you have created in the previous step) in the ***line no. 43***.

   c. **From** the **Eclipse IDE**, Select the **File** and select the **Save**.

   d. **Execute** the **Run Executor**.

Note: In the **Console**, you will see the Object details with **Object Name**, **LastModified** and **Size**.

Get (read) existing Object

55. **From** the **Executor.java**.

   a. **Replace** the ***listExistingS3BucketContents*** method with the **getObject** method in the ***line no. 20***.

b. **Replace** the **BUCKET_NAME** with the **images-123**, bucket name (which you have created in the previous step) in the ***line no. 77***.

Note: In the code, we have already mentioned the file name (**File01.txt**).

c. **From** the **Eclipse IDE**, Select the **File** and select the **Save**.

d. **Execute** the **Run Executor**.

Note: In the **Console**, you will see the Object content.

Upload new Object with metadata

56. **From** the **Executor.java**.

a. **Replace** the ***getObject*** method with the ***uploadObjectWithMetadata*** method in the ***line no. 20***.

b. **Replace** the **BUCKET_NAME** with the **images-123**, bucket name (which you have created in the previous step) in the ***line no. 57***.

Note: In the code, we have already mentioned upload file path and file name (**File02.txt**) with Project and Owner metadata which you have copied in the S3-Code folder in C drive.

c. **From** the **Eclipse IDE**, Select the **File** and select the **Save**.

d. **Execute** the **Run Executor**.

Note: If Object uploaded succesfully, in the **Console**, you will see the **PutObjectResponse.**

Update metadata with the existing Object

57. **From** the **Executor.java**.

a. **Replace** the ***uploadObjectWithMetadata*** method with the ***updateMetadata*** method in the ***line no. 20***.

b. **Replace** the **BUCKET_NAME** with the **images-123**, bucket name (which you have created in the previous step) in the ***line no. 68***.

Note: In the code, we have already mentioned file name (**File01.txt**) and the Project metadata.

c. **From** the **Eclipse IDE**, Select the **File** and select the **Save**.

d. **Execute** the **Run Executor**.

Note: If Object uploaded succesfully, in the **Console**, you will see the **CopyObjectResponse**.

Get (read) existing Object (File01.txt)

58. **From** the **Executor.java**.

a. **Replace** the ***uploadObjectWithMetadata*** method with the **getObject** method in the ***line no. 20***.

Note: In the code, we have already mentioned the file name (**File01.txt**).

b. **From** the **Eclipse IDE**, Select the **File** and select the **Save**.

c. **Execute** the **Run Executor**.

Note: In the **Console**, you will see the **Object content** with **Metadata**.

Get (read) existing Object (File02.txt)

59. **From** the **Executor.java**.

a. You can see the **getObject** method in the ***line no. 20***.

b. **Replace** the **File01.txt** with the **File02.txt**, file name in the ***line no. 78***.

c. **From** the **Eclipse IDE**, Select the **File** and select the **Save**.

d. **Execute** the **Run Executor**.

Note: In the **Console**, you will see the **Object content** with **Metadata**.

Create Pre-Signed URL

60. **From** the **Executor.java**.

    a. **Replace** the ***getObject*** method with the ***getPresignedUrl*** method in the ***line no. 20***.

    b. **Replace** the **BUCKET_NAME** with the **images-123**, bucket name (which you have created in the previous step) in the ***line no. 86***.

Note: In the code, we have already mentioned file name (**File02.txt**).

    c. **From** the **Eclipse IDE**, Select the **File** and select the **Save**.

    d. **Execute** the **Run Executor**.

Note: In the **Console**, you will see the Pre-signed URL.

Note: **Copy** the Pre-signed URL in the **Notepad**.

Get (read) Object using Pre-Signed URL

61. **From** the **Executor.java**.

    a. **Replace** the ***getPresignedUrl*** method with the ***getPresignedUrlData*** method in the ***line no. 20***.

    b. **Replace** the **PRE-SIGNED_URL** with the Pre-Signed URL which you have created in the previous step in the ***line no. 93***.

    c. **From** the **Eclipse IDE**, Select the **File** and select the **Save**.

    d. **Execute** the **Run Executor**.

Note: In the **Console**, you will see the Object content.

Step 2: Access the Object from Browser

62. **From** the local desktop/ laptop **Web browser**, paste the **Pre-Signed_URL** of **File02.txt**, which you have copied in the previous step and access your Object content.

Note: You will see the **File02.txt** Object content.

Note

- Do not delete any resources you deployed in this lab.
- You will be using them in the next lab of this module.