

# **Developing Storage Solutions with Amazon Simple Storage Service (S3)**

## **(LAB-M06-01)**

Version Control	
Document	Developing Storage Solutions with Amazon S3
Owner	Ahmad Majeed Zahoory
Version	3.1
Last Change	22 <sup>nd</sup> May 2024
Description of Change	Task steps updated

**Lab duration:** 60 minutes

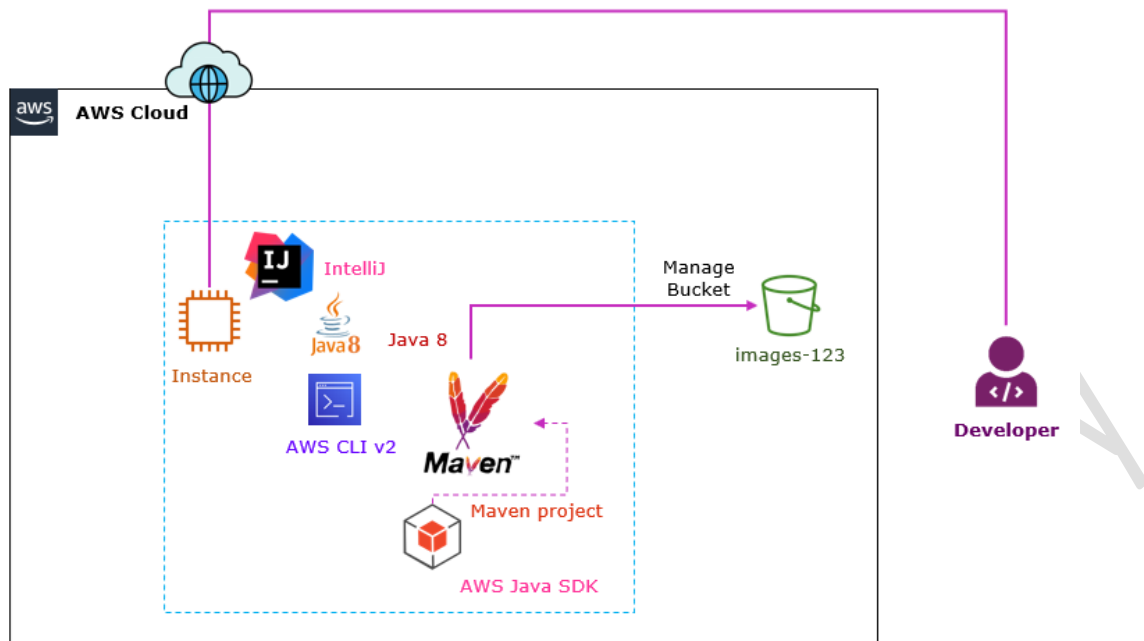
### **Lab scenario**

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 a bucket.
- Delete a bucket.
- List the existing buckets.
- List the bucket contents.
- Upload the objects in bucket.
- Get the objects from bucket.
- Upload the objects with metadata in bucket.
- Update metadata for the existing objects.
- Create the Pre-signed URL.
- Get the objects using Pre-signed URL.

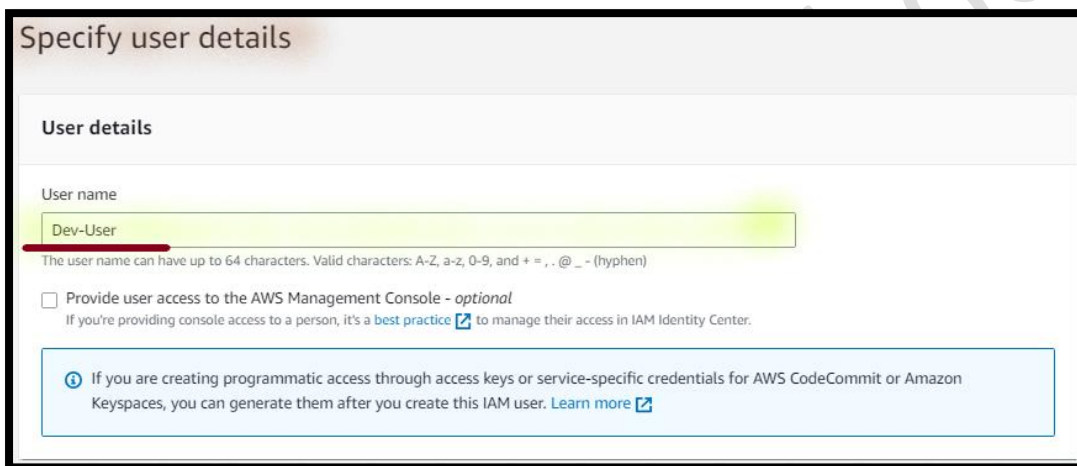


## Task 1: Create IAM User

In this task, you will create AWS IAM User with AWS S3 permission.

### Step 1: Create IAM User

1. In the **AWS Management Console**, on the **Services** menu, click **IAM**.
2. Select **Users**.
  - a. Select **Add users**.
    - i. In the **Specify user details** section:
      - a) **Username**: Write **Dev-User-YOUR-ID**.



Specify user details

User details

User name

Dev-User

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ \_ - (hyphen)

☐ Provide user access to the AWS Management Console - *optional*  
If you're providing console access to a person, it's a best practice [to manage their access in IAM Identity Center](#).

**i** If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

**Note:** Leave the other details as default.

- b) Select **Next**.
    - ii. In the **Set permissions** section:
      - a) Select **Attach policies directly**.
        - 1) In the **Search box**, write **AmazonS3FullAccess** and select **Enter Key**.
        - 2) Select **AmazonS3FullAccess**.

### Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

**Permissions options**

☐ Add user to group  
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

☐ Copy permissions  
Copy all group memberships, attached managed policies, and inline policies from an existing user.

☒ Attach policies directly  
Attach managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

**Permissions policies (1/1240)**  
Choose one or more policies to attach to your new user.

Filter by Type

Search: AmazonS3FullAccess X All types 1 match

<input checked="" type="checkbox"/>	Policy name	Type	Attached entities
<input checked="" type="checkbox"/>	AmazonS3FullAccess	AWS managed	0

3) Select **Next**.

iii. In the **Review and create** section.

**Note:** In the **Permission summary**, you can see the **AmazonS3FullAccess** policy.

### Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

**User details**

User name Dev-User	Console password type None	Require password reset No
-----------------------	-------------------------------	------------------------------

**Permissions summary**

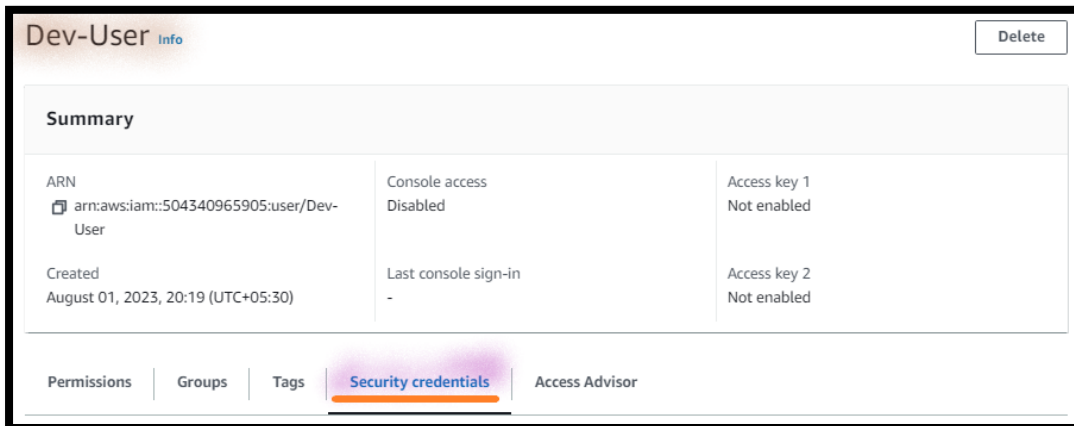
Name	Type	Used as
AmazonS3FullAccess	AWS managed	Permissions policy

a) Select **Create user**.

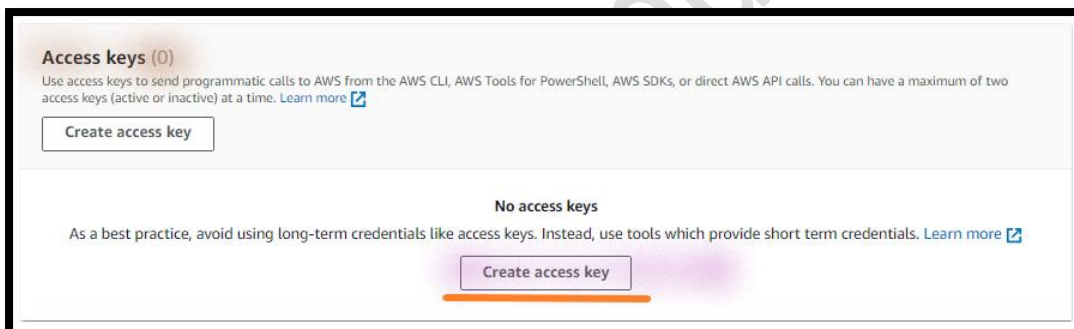
**Note:** You can see the message **"User created successfully"**.

## Step 2: Create IAM User Access Keys

3. From the **IAM** console.
  - a. Open the **Dev-User-YOUR-ID**.
    - i. Select **Security credentials**.



- a) **Access keys:** Select **Create access key**.



- I. In the **Access key best practices & alternatives** section:
  - A. **Use case:** Select **Command Line Interface (CLI)**.
  - B. **I understand the above recommendation and want to proceed to create an access key:** **Enable** the **Checkmark**.

## Access key best practices & alternatives Info

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

☒ **Command Line Interface (CLI)**  
You plan to use this access key to enable the AWS CLI to access your AWS account.


☐ **Local code**  
You plan to use this access key to enable application code in a local development environment to access your AWS account.

☐ **Application running on an AWS compute service**  
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

☐ **Third-party service**  
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

☐ **Application running outside AWS**  
You plan to use this access key to enable an application running on an on-premises host, or to use a local AWS client or third-party AWS plugin.

☐ **Other**  
Your use case is not listed here.

 **Alternatives recommended**

- Use [AWS CloudShell](#), a browser-based CLI, to run commands. [Learn more](#)
- Use the [AWS CLI V2](#) and enable authentication through a user in IAM Identity Center. [Learn more](#)

Confirmation

☒ I understand the above recommendation and want to proceed to create an access key.

C. Select **Next**.

II. In the **Set description tag** section:

A. **Description tag value:** Write **Development User for managing AWS services**.

### Set description tag - optional Info

The description for this access key will be attached to this user as a tag and shown alongside the access key.

**Description tag value**  
Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

Development User for managing AWS services

Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: \_ . : / = + - @

B. Select **Create access key**.



**Note:** You can see the message "Access key created".

III. In the **Retrieve access keys** section:

A. Select **Download .csv file**.

### Retrieve access keys Info

**Access key**  
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
 AKIAXK3IQNII6Q43ARHP	 ***** <a href="#">Show</a>

**Access key best practices**

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [Best practices for managing AWS access keys](#).

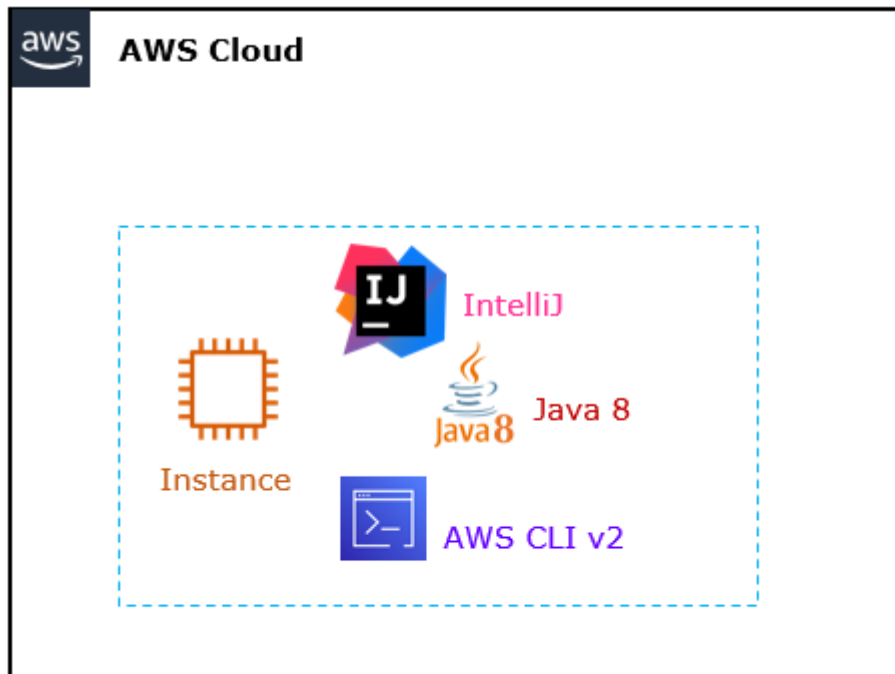
[Download .csv file](#) [Done](#)

**Note:** **Access key ID** and **Secret access key** details of the **Dev-User-YOUR-ID** gets downloaded in your **local desktop/ laptop**.

B. Select **Done**.

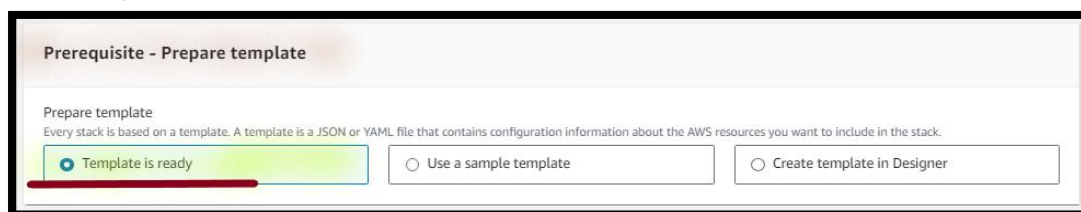
## Task 2: Build Development Environment

In this task, you will build the AWS EC2 instance to build development environment and install Java, IntelliJ and AWS CLI using the cloud formation template.



### Step 1: Create EC2 Instances

4. In the **AWS Management Console**, on the **Services** menu, search and select **CloudFormation**.
5. Choose the **YOUR ALLOCATED REGION**, region list to the right of your account information on the navigation bar.
6. Select **Create stack** and configure:
  - a. In the **Create stack** page:
    - i. **Prepare template**: Select **Template is ready**.





ii. **Template source:** Select **Upload a template file**.

iii. **Choose file:** Click on **Choose file**.

a) **Navigate** and **select** the **Dev-Instance.yaml** file.

**Note:** **Dev-Instance.yaml** template is provided with the Lab manual.

**Note:** AWS template **performing** the **following** tasks:

1. **Creating Windows instances**.
2. **Creating t2.medium** instance (**2 vCPU** and **4 GB memory**) [*This instance type attract charges*].
3. **Set** the **Administrator's Password**.

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

**Specify template**  
A template is a JSON or YAML file that describes your stack's resources and properties.

**Template source**  
Selecting a template generates an Amazon S3 URL where it will be stored.

☐ Amazon S3 URL ☒ Upload a template file

**Upload a template file**  
**Choose file**

Dev-Instance.yaml  
JSON or YAML formatted file

S3 URL: <https://s3.us-east-1.amazonaws.com/cf-templates-vib2vpwu8sff-us-east-1/2023-08-02T151409.027Zjy4-Dev-Instance.yaml> **View in Designer**

iv. Select **Next**.

b. In the **Specify stack details** page:

i. **Stack name:** Write **Dev-Instance-JV**.

**Stack name**

Stack name

Dev-Instance-JV

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

**Note:** Leave other details as default.

- ii. Select **Next**.
- c. In the **Configure stack options** page:

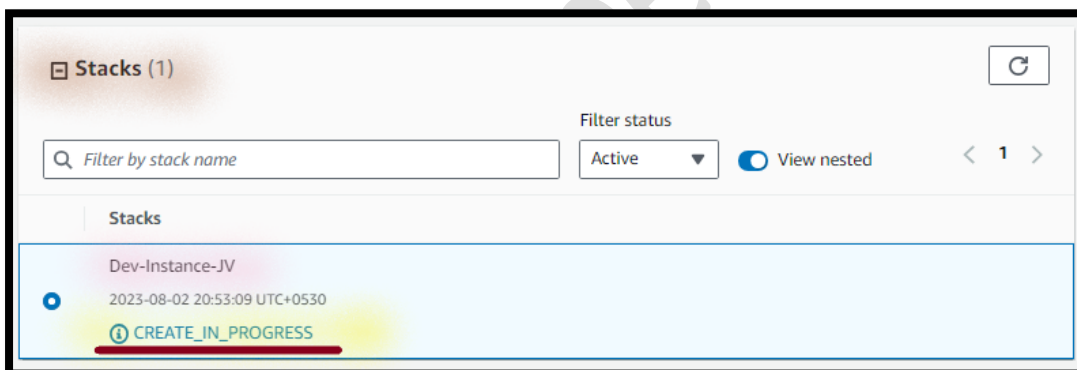
**Note:** Leave all the details as default.

- i. Select **Next**.
- d. In the **Review Dev-Instance-JV** page:

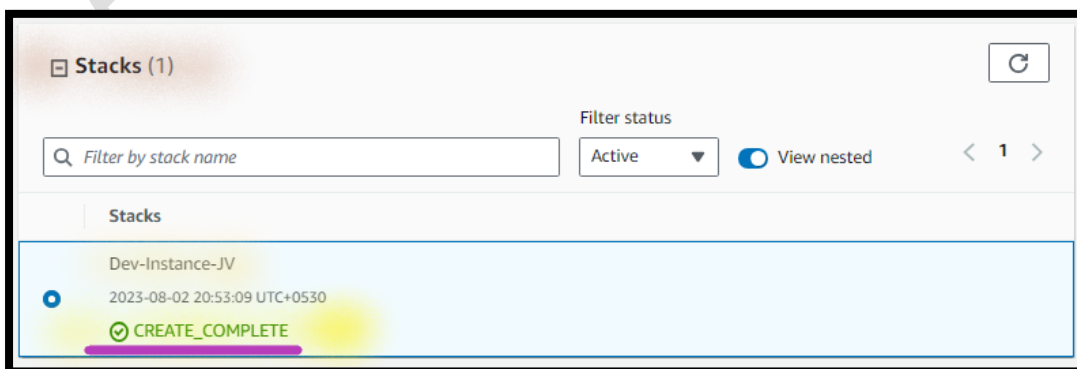
**Note:** Review all the details.

- i. Select **Submit**.

**Note:** You can see the **Stack** status as **CREATE\_IN\_PROGRESS**.



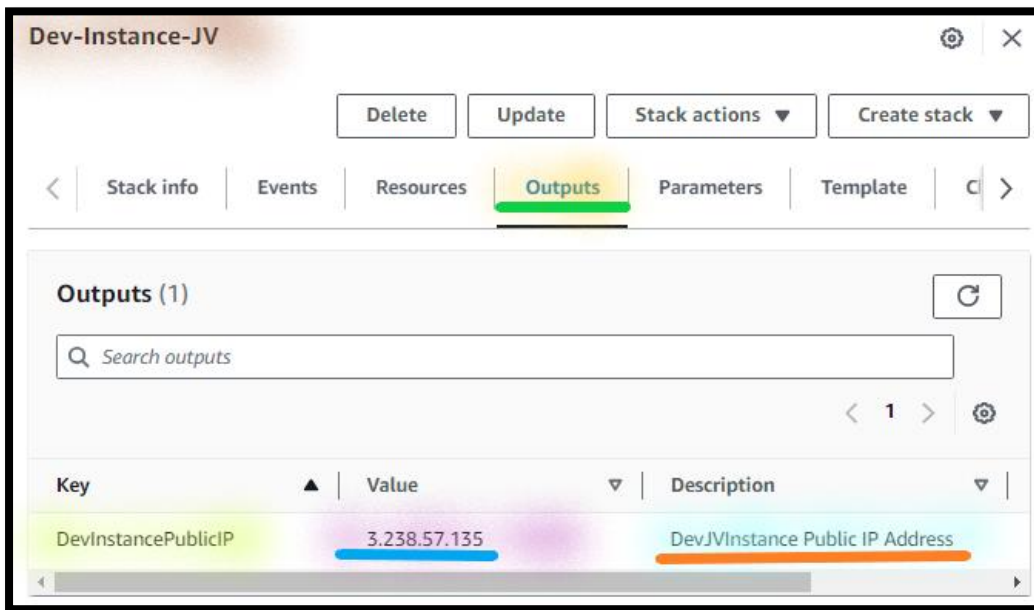
**Note:** **Wait**, till you can see the **Stack** status as **CREATE\_COMPLETE**. You can **Refresh** your screen



## Step 2: View the Output

7. From the **Dev-Instance-JV CloudFormation** console:
  - a. Select **Outputs**.

**Note:** Copy the **DevJVInstance Public IP** address in the **Notepad**.



## Step 3: Connect to Instance

8. From the **Local Desktop/ Laptop** (*Windows desktop*), right click on **Start** & **Run**.
  - a. In the **Open**, write **mstsc**.
  - b. Select **Ok**.
    - i. From the **Remote Desktop Connection**:
      - a) **Computer**: Write the **Public IP Address** of the **DevJVInstance**.
      - b) Select **Connect**.

**Note:** You can **get the prompt** to enter the **Username** and **Password**.

- 1) **Username:** Write **Administrator**.
- 2) **Password:** Write **lab-password@123**.
- 3) Select **Ok**.

#### Step 4: Install the Java Development Kit

9. From the **DevJVInstance** (*Windows Server 2022*).
  - a. **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://bitbucket.org/ahmadzahoory/dev/downloads/jdk-8u351-windows-x64.exe>

**Info:** You can also download the Java SE Development Kit 8 from the Oracle site.

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

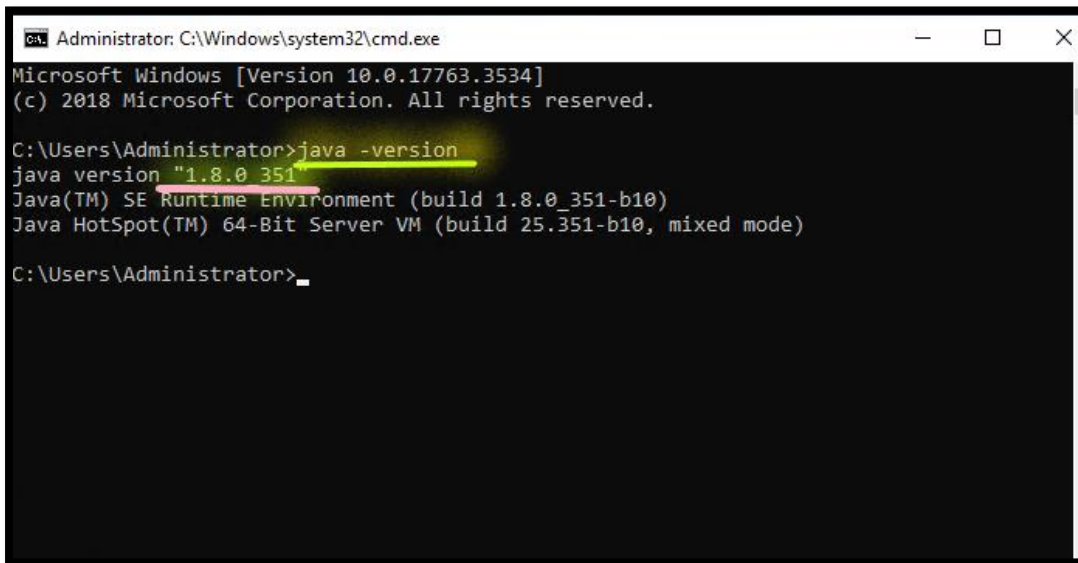
**Note:** **Wait**, till **Java SE Development Kit 8** install **successfully**.

#### Step 5: Check the JDK Version

10. From the **DevJVInstance**, right click on **Start** & **Run**.
  - a. In the **Open**, write **cmd**.
  - b. Select **Ok**.
    - i. From the **Command line interpreter**:
      - a) **Execute** the **below command** to **verify** the **Java version**:

Java -version

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



```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.17763.3534]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>java -version
java version "1.8.0_351"
Java(TM) SE Runtime Environment (build 1.8.0_351-b10)
Java HotSpot(TM) 64-Bit Server VM (build 25.351-b10, mixed mode)

C:\Users\Administrator>
```

## Step 6: Install the IntelliJ IDE

11. From the **DevJVInstance**.

- a. **Download** and **install** the **IntelliJ IDE 2024** for **Windows x64**.

**Note:** Use the below **URL** to download the **IntelliJ IDE 2024 Community edition**.

<https://bitbucket.org/ahmadzahoory/devenv/downloads/ideaIC-2024.1.1.exe>

**Info:** You can also download the IntelliJ from the intelliJ site.  
<https://www.jetbrains.com/idea/download/?section=windows>

**Note:** Wait till **IntelliJ IDE** install **successfully**.

**Note:** **Don't launch** the **IntelliJ IDE**.

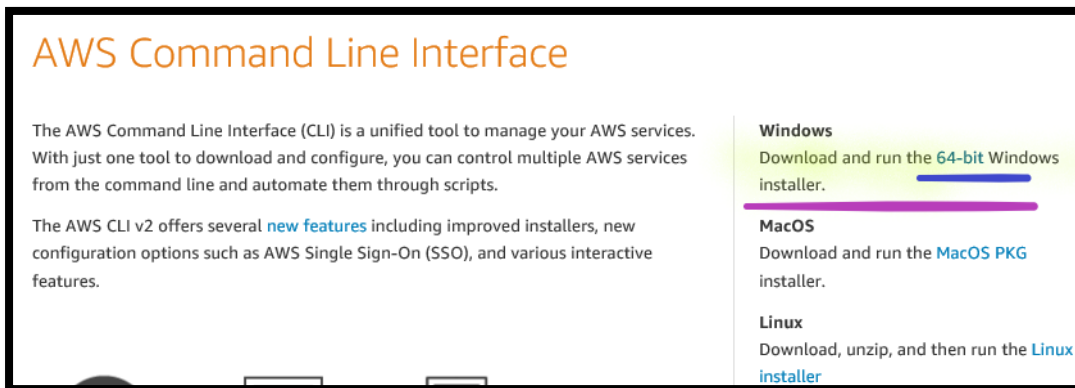
## Step 7: Install the AWS CLI V2

12. From the **DevJVInstance**.

- a. **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 **successfully**.

## Step 8: Check the AWS CLI Version

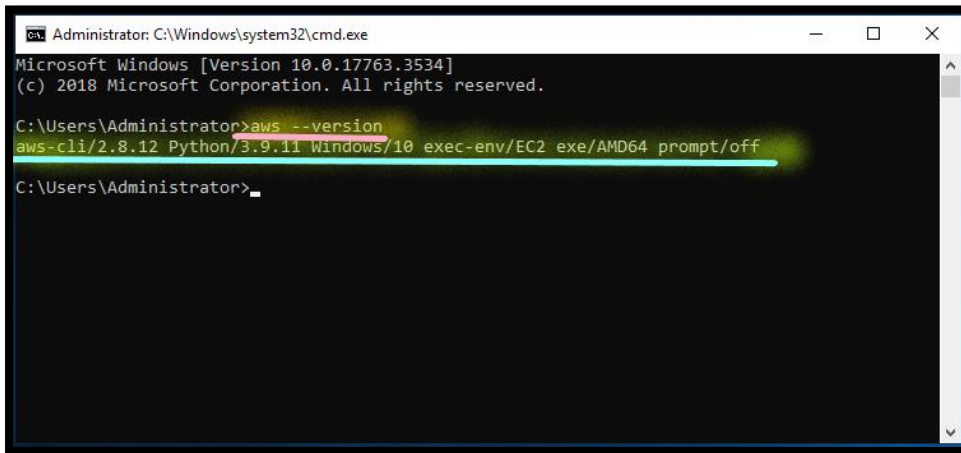
13. From the **DevJVInstance**, right click on **Start** & **Run**.

- a. In the **Open**, write **cmd**.
- b. Select **Ok**.
  - i. From the **Command line interpreter**:
    - a) **Execute** the **below command** to **verify** the **AWS version**.

```
aws --version
```

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

**Note:** If you can see the "'aws' is not recognized as an internal or external command" message, **Restart** the **DevJVInstance**.



```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.17763.3534]
(c) 2018 Microsoft Corporation. All rights reserved.

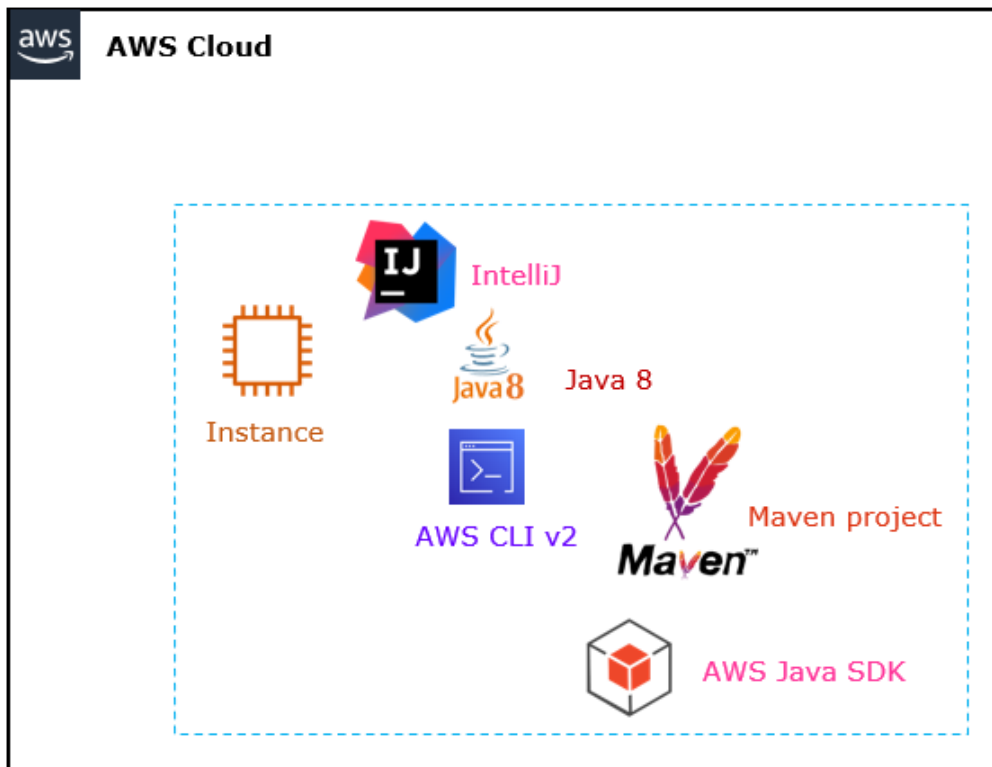
C:\Users\Administrator>aws --version
aws-cli/2.8.12 Python/3.9.11 Windows/10 exec-env/EC2 exe/AMD64 prompt/off

C:\Users\Administrator>
```

**Note:** Go to the next task. But **Don't close** the **DevJVInstance**.

## Task 3: Create Maven Project

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



### Step 1: Develop the Java Code

14. Unzip the **LAB-m06-01-Code.zip** (Java code).

**Note:** **lab-m06-01-code.zip** code file is available with the **Lab manual**.

**Note:** **Review** the **Code**, by **opening** in the **Notepad**.

### Step 2: Configure the Credentials and Configuration

15. From the **DevJVInstance**, right click on **Start** & **Run**.

- In the **Open**, write **cmd**.
- Select **Ok**.



i. From the **Command line interpreter**:

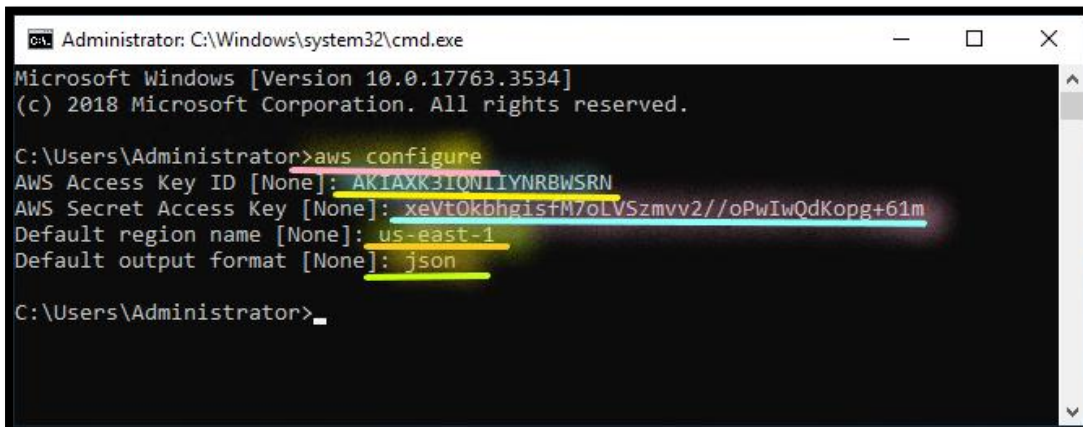
- a) **Execute** the **below command** to **configure** the **AWS credentials**.

```
aws configure
```

- a) **AWS Access Key ID**: Type **Dev-User-YOUR-ID's**, **access key** (copy from the .csv file), press **Enter** key to continue.
- b) **AWS Secret Access Key**: Type **Dev-User-YOUR-ID's**, **secret access key** (copy from the .csv file), press **Enter** key to continue.

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

- c) **Default region name**: Type **us-east-1**, press **Enter** key to continue.
- d) **Default output format**: Type **json**, press **Enter** key to continue.



```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.17763.3534]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>aws configure
AWS Access Key ID [None]: AKIAKK3IONTIYNRBWSRN
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json

C:\Users\Administrator>
```

- b) **Execute** the **below command** to **exit**.

```
exit
```

### Step 3: Verify the Configuration

16. From the **DevJVInstance**, right click on **Start** & **Run**.

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

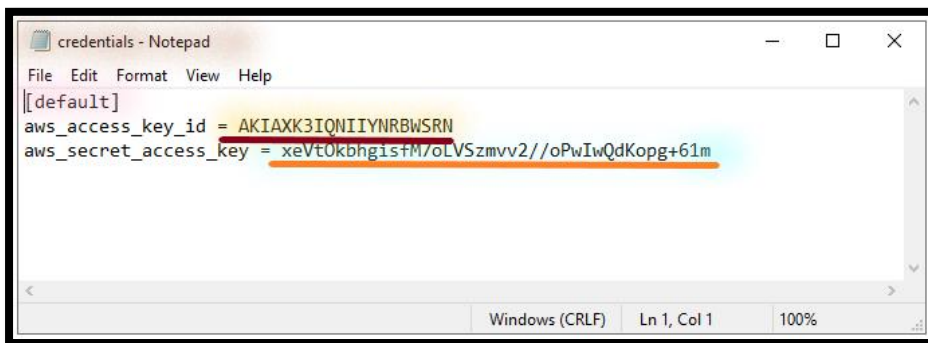
b. Select **Ok**.

i. From the **File explorer**:

a) **Open** the **.aws** folder.

1) **Open** the **Credentials** file in **Notepad**.

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



I. Select **File**.

II. Select **Exit**.

2) **Open** the **Config** file in **Notepad**.

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



- 1) Select **File**.
  - 2) Select **Exit**.
- b) **Close** the **File explorer**.

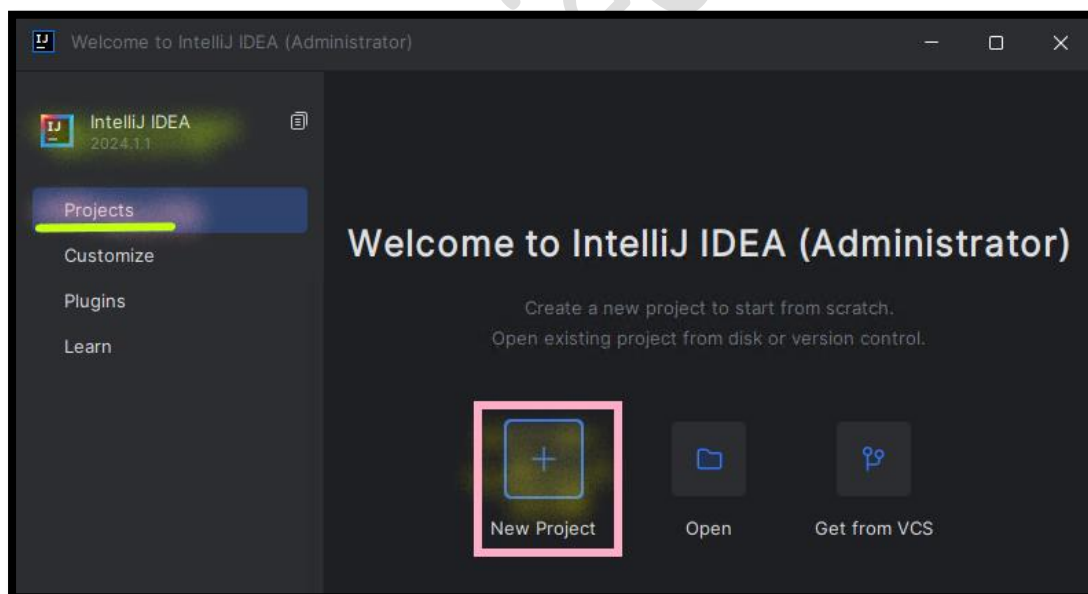
#### Step 4: Launch the IntelliJ IDE

17. From the **DevJVInstance**.
  - a. Open the **IntelliJ IDE**.

**Note:** **Wait**, till you can see the **IntelliJ IDE Workspace**.

#### Step 5: Create Maven Project

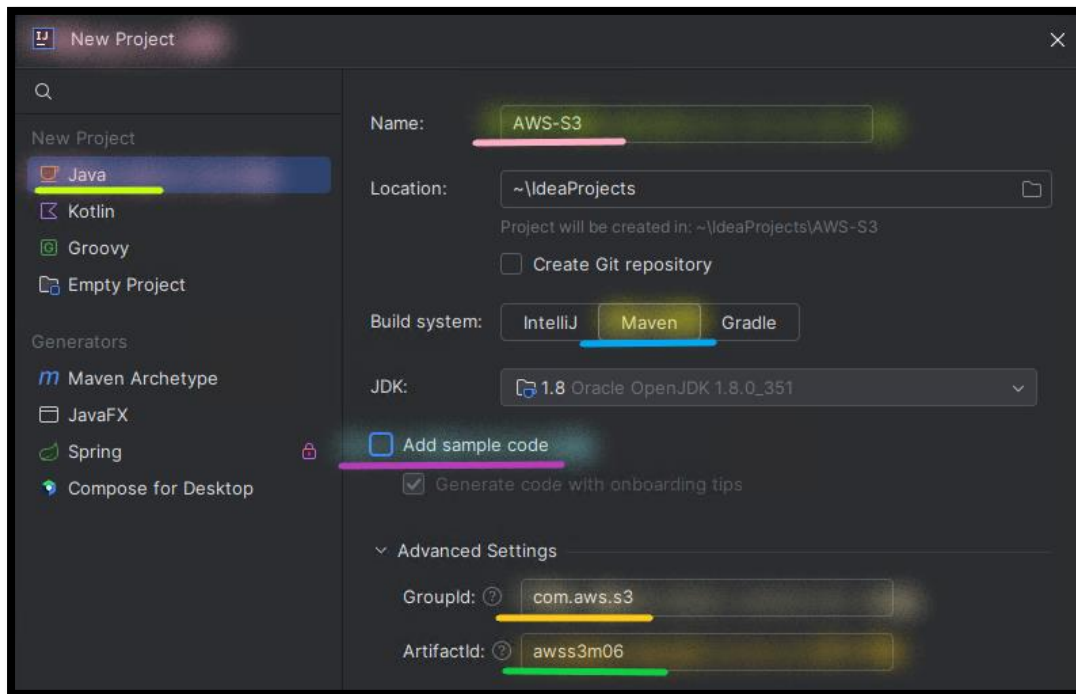
18. From the **IntelliJ IDE**.
  - a. Select **Project**.
    - i. Select **New Project**.



- ii. In the **New project** page:
  - a) Select **Java**.
    - 1) **Name**: Write **AWS-S3**.
    - 2) **Build system**: Select **Maven**.
    - 3) **Add sample code**: **Uncheck** the **Checkmark**.

4) **Expand** the **Advanced Settings**:

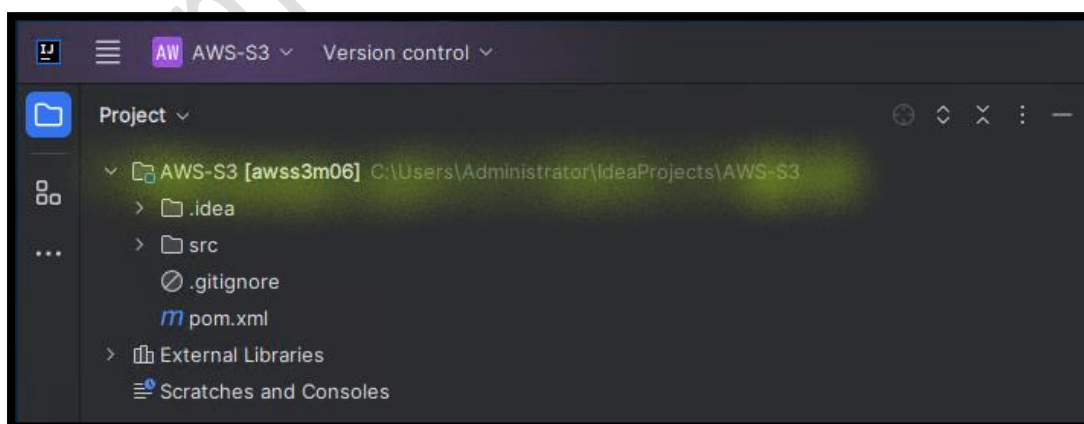
- I. **Group Id**: Write **com.aws.s3**.
- II. **Artifact Id**: Write **awss3m06**.



**Note:** Leave the other details as default.

5) Select **Create**.

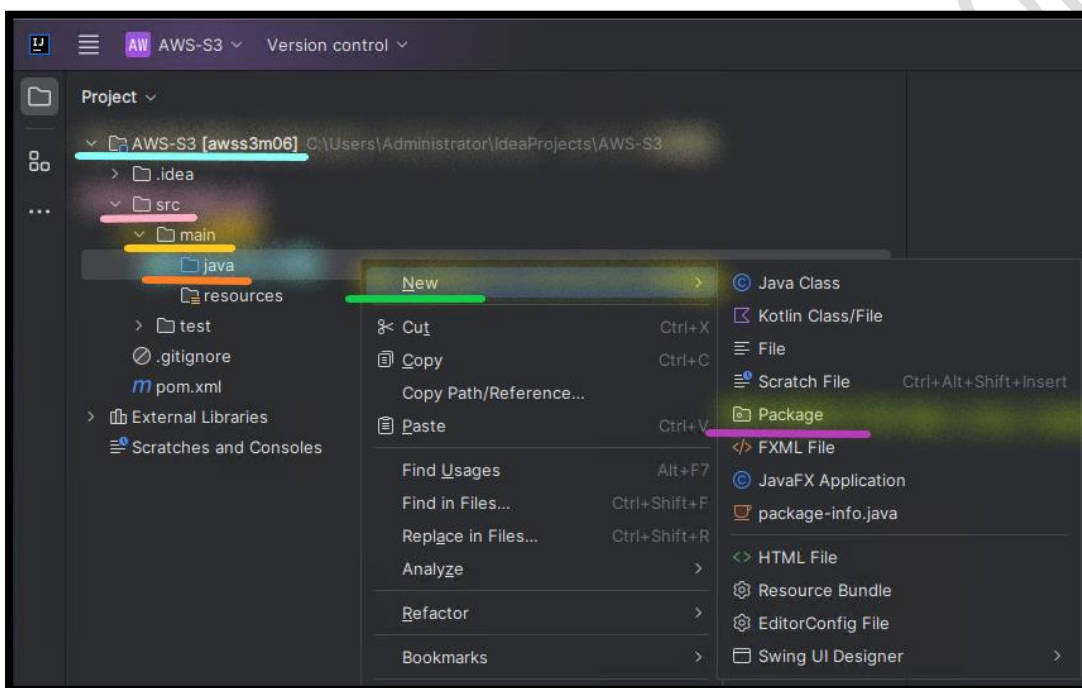
**Note:** You can see the **AWS-S3** project.



## Step 6: Create the Package

19. From the **IntelliJ IDE**.

- a. **Expand** the **awss3m06** project.
  - i. **Expand** the **src**.
    - a) **Expand** the **main**.
      - 1) Select the **Java**.
- b. **Right-click** on the **Java** project.
  - i. Select **New**.
    - a) Select **Package**.

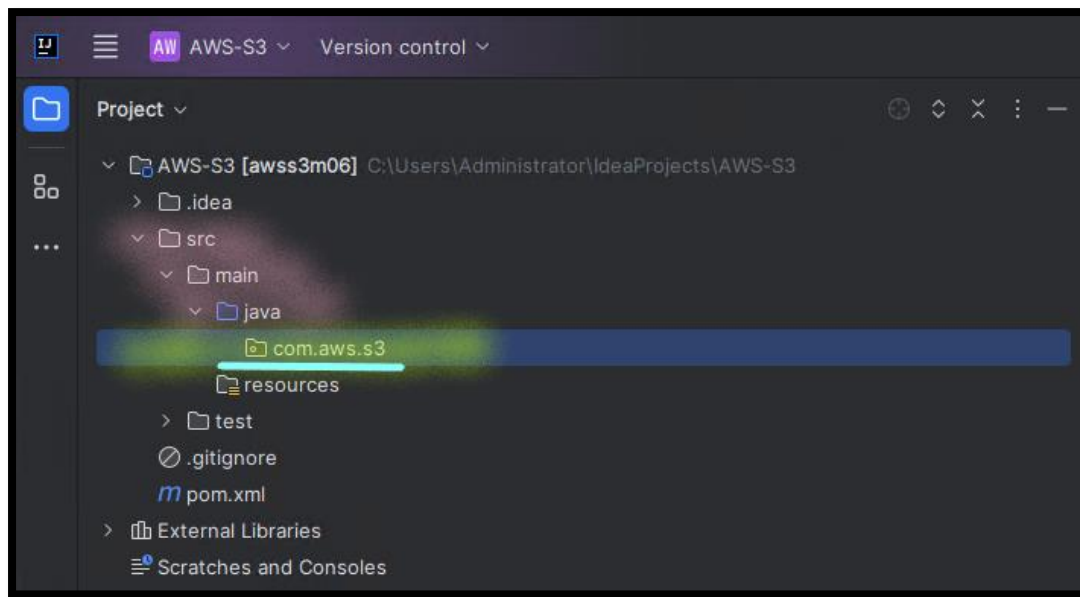


b) In the **New Package** section:

1) **Name:** Write **com.aws.s3**.

I. Press **Enter** key.

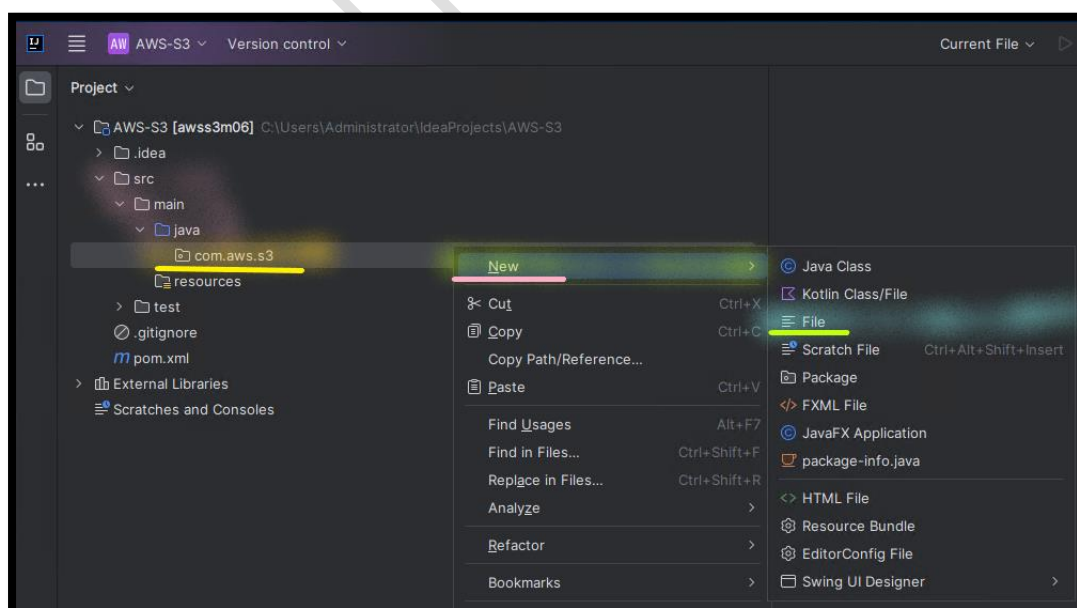
**Note:** You can see the **com.aws.s3** package under **src/main/java**.



## Step 7: Create the **Executor.java** File in the Package

20. From the **IntelliJ IDE**.

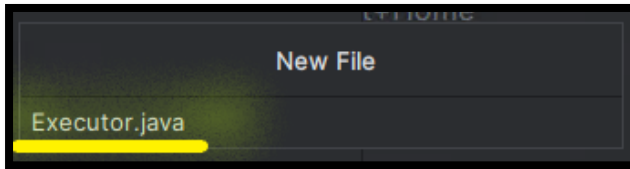
- a. **Expand** the **awss3m06** java project.
  - i. **Expand** the **src/main/java** resource path.
    - a) **Select** the **com.aws.s3** java package.
      - b) **Right-click** on the **com.aws.s3** java package.
        - 1) Select **New**.
        - 2) Select **File**.



c) In the **New File** section:

1) **Name:** Write **Executor.java**.

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



I. Press **Enter** key.

## Step 8: Create the **Utilities.java** File in the Package

21. From the **IntelliJ IDE**.

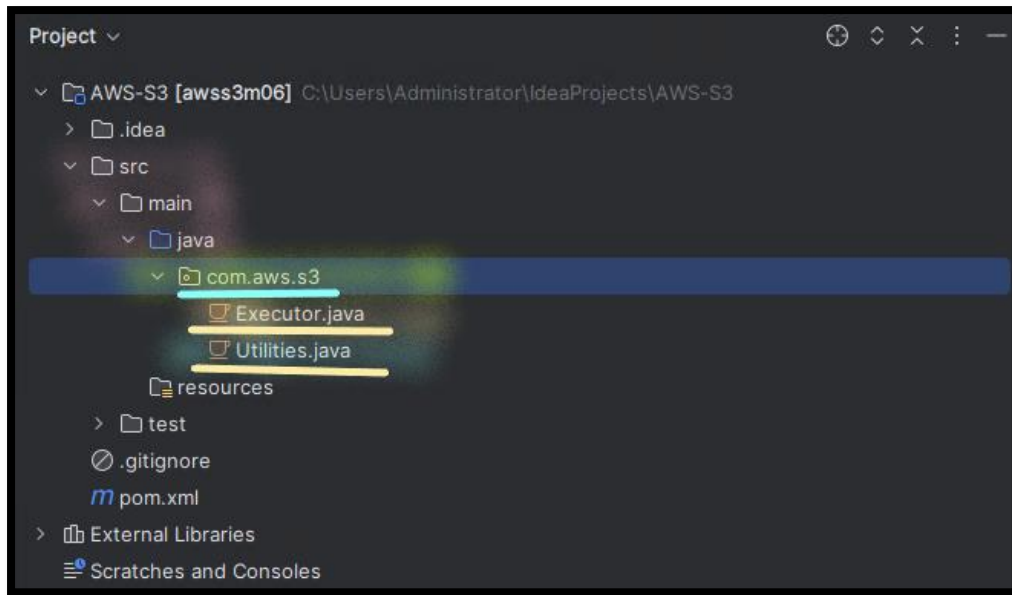
- a. **Expand** the **awss3m06** java project.
  - i. **Expand** the **src/main/java** resource path.
    - a) **Select** the **com.aws.s3** java project.
    - b) **Right-click** on the **com.aws.s3** java package.
      - 1) Select **New**.
      - 2) Select **File**.
    - c) In the **New File** section:
      - 1) **File name:** Write **Utilities.java**.

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

I. Press **Enter** key.

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





## Step 9: Update the Java Code

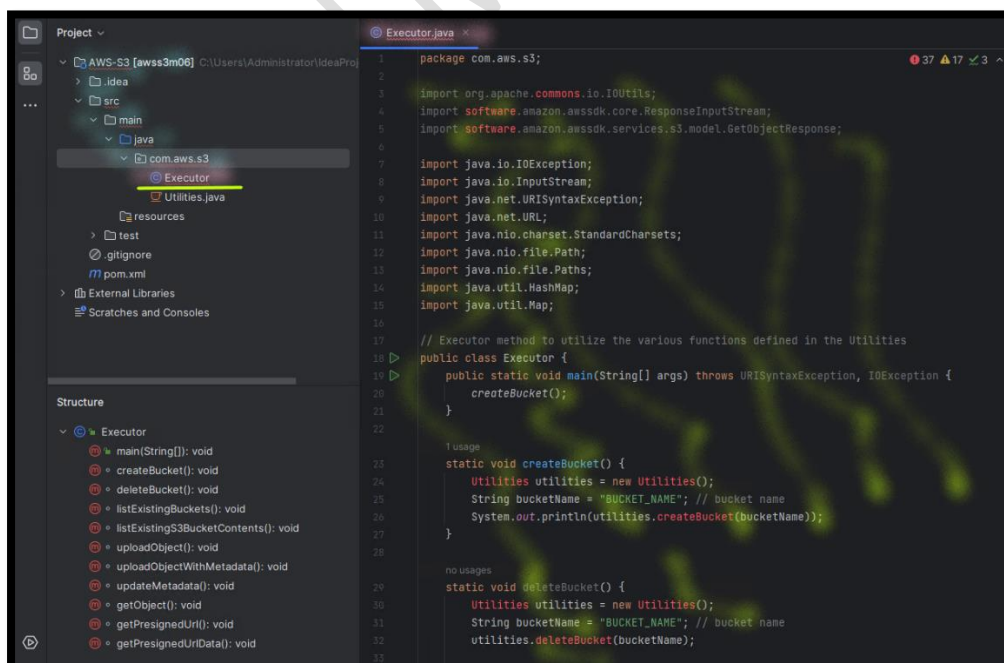
### Update the Executor.java

22. **Double-click** on the **Executor.java** file.

a. **Paste** the **Code** from **Executor.txt** file.

**Note:** **Code** is available with the **Lab manual**.

i. Select **CTRL + S** (to save).





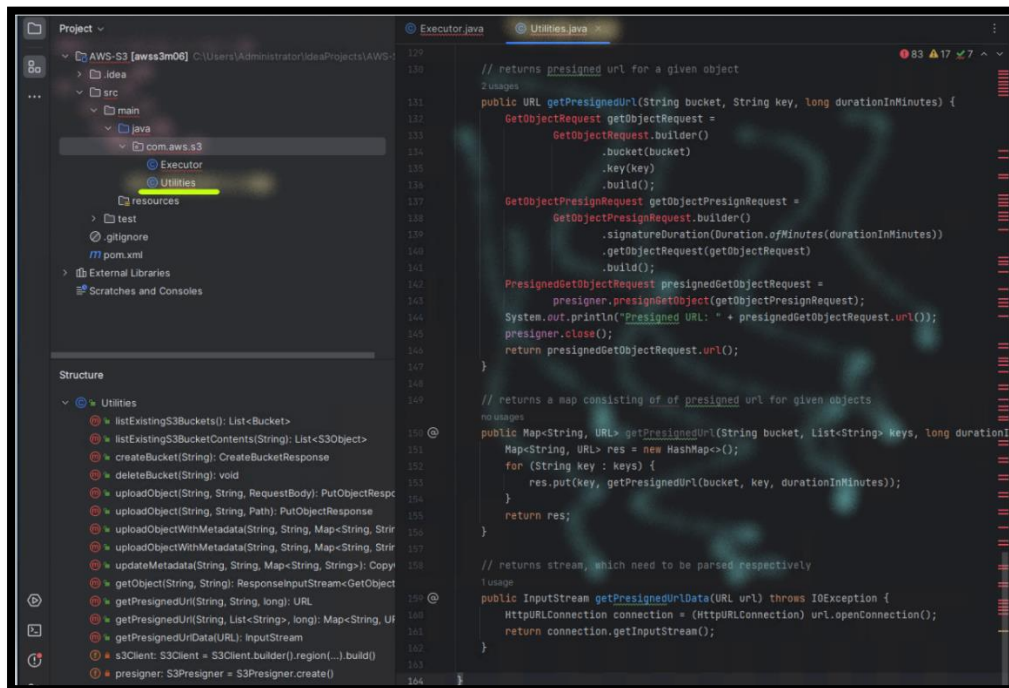
## Update the Utilities.java

23. **Double-click** on the **Utilities.java** java file.

- a. **Paste** the **Code** from **Utilities.txt** file.

**Note:** **Code** is available with the **Lab manual**.

- i. Select **CTRL + S** (to save).



## Update the Pom.xml

24. **Double-click** on the **pom.xml**.

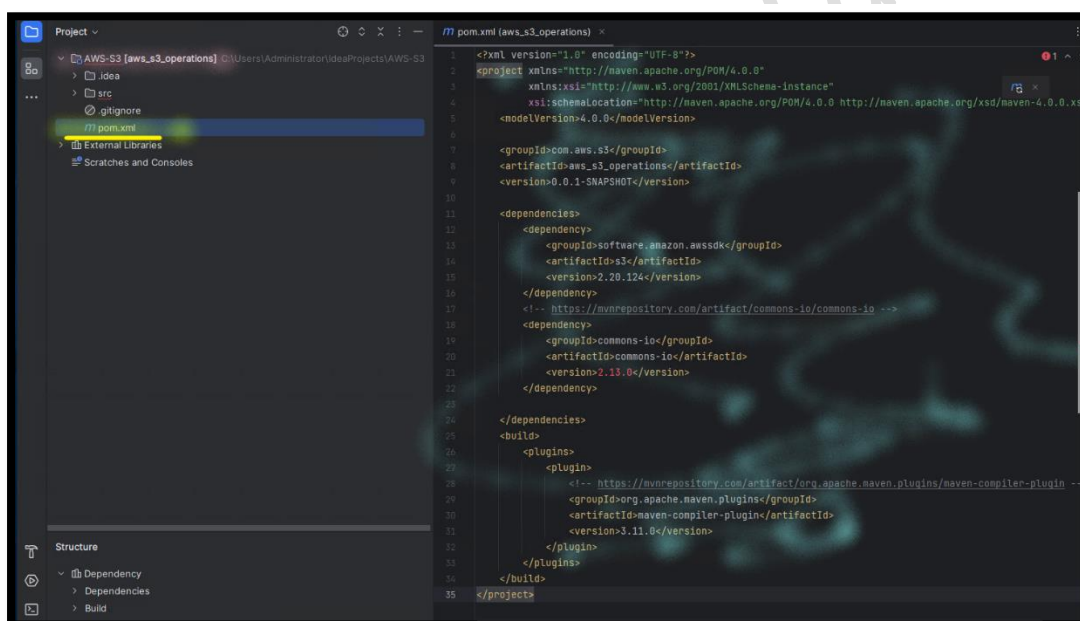
- a. **Remove** the **existing code**.
- b. **Copy** the **Code** from **pom.xml** file.

**Note:** **Code** is available with the **Lab manual**.

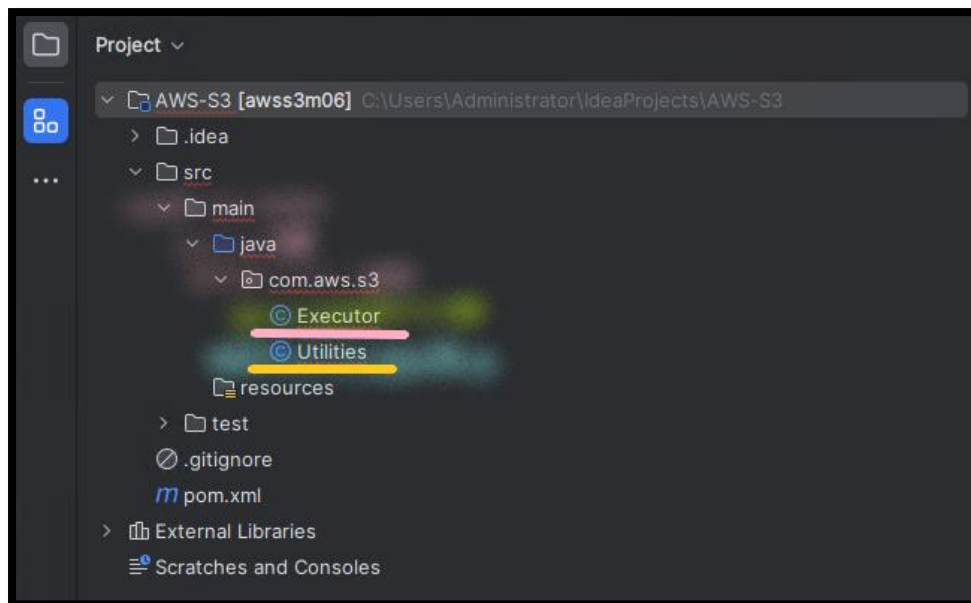
**Note:** In the **POM** you can see the:

1. **AWS Java SDK** [*software.amazon.awssdk*] - The AWS Java SDK for Amazon S3 module holds the client classes that are used for communicating with Amazon Simple Storage Service.
2. **Apache Common IO** [*common-io*] - The Apache Commons IO library contains utility classes, stream implementations, file filters, file comparators, endian transformation classes, and much more.
3. **Apache Maven Compiler Plugin** [*org.apache.maven.plugins*] - The Compiler Plugin is used to compile the sources of your project.

i. Select **CTRL + S** (to save).



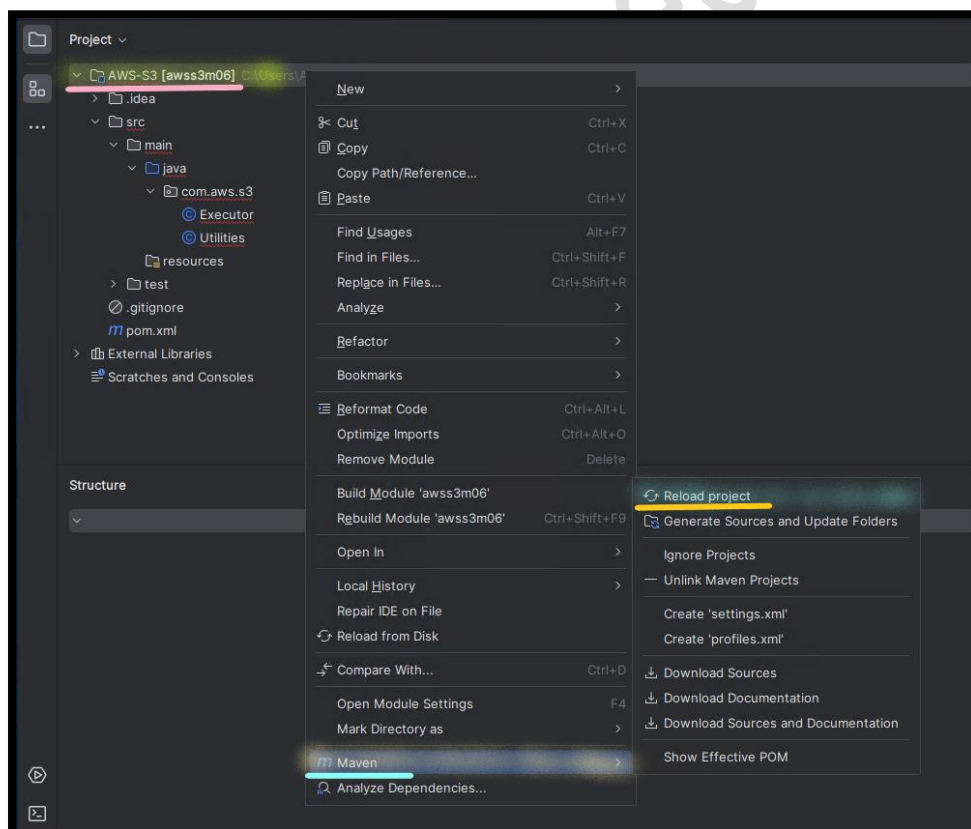
**Note:** You can see the **Error** against **Executor.java** and **Utilities.java** under Java package.



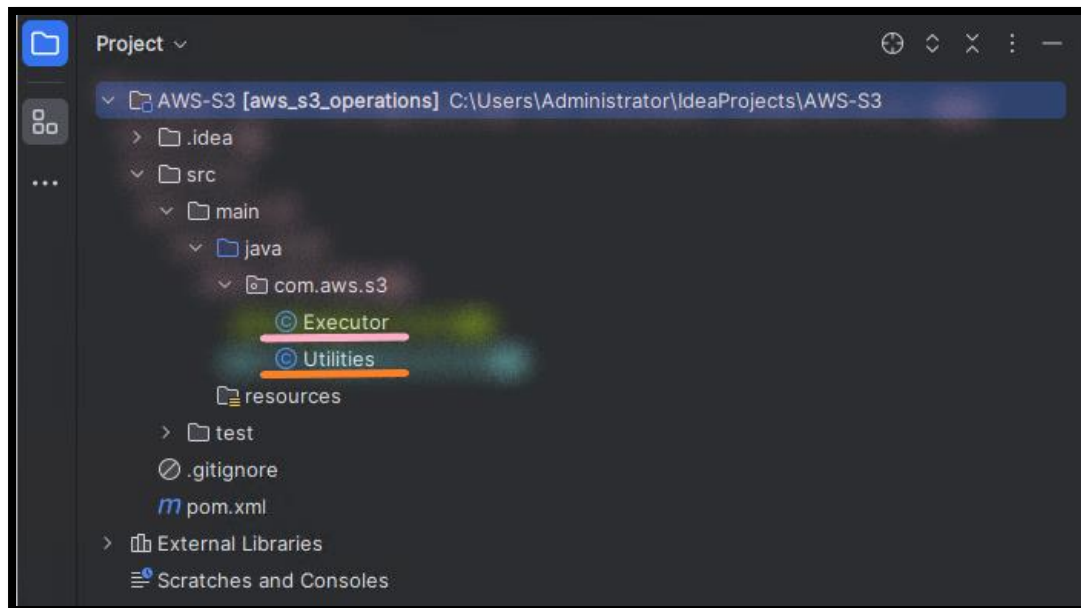
25. **Right-click** on the **awss3m06** java project.

a. Select **Maven**.

i. Select **Reload project**.

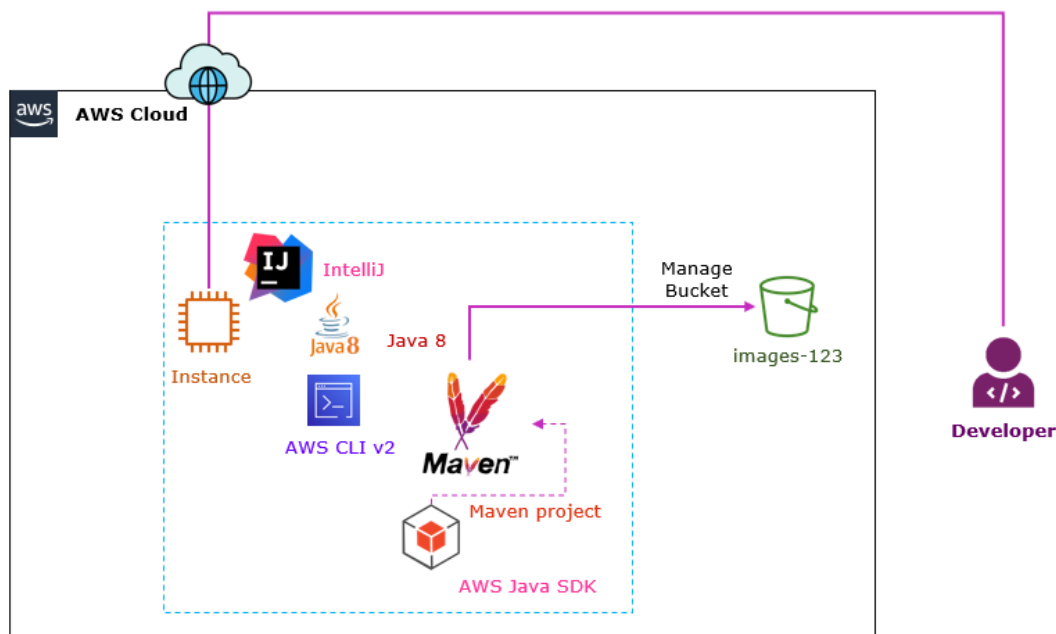


**Note:** **Wait** for **indexing** and **ensure** that you **Don't see** any **Errors** in the Java project.



## Task 4: Manage AWS S3 from IntelliJ

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



## Step 1: Copy the Objects

26. From the **DevJVInstance** (Windows Server 2022), right click on **Start** & **Run**.

a. In the **Open**, write **C:\**.

b. Select **Ok**.

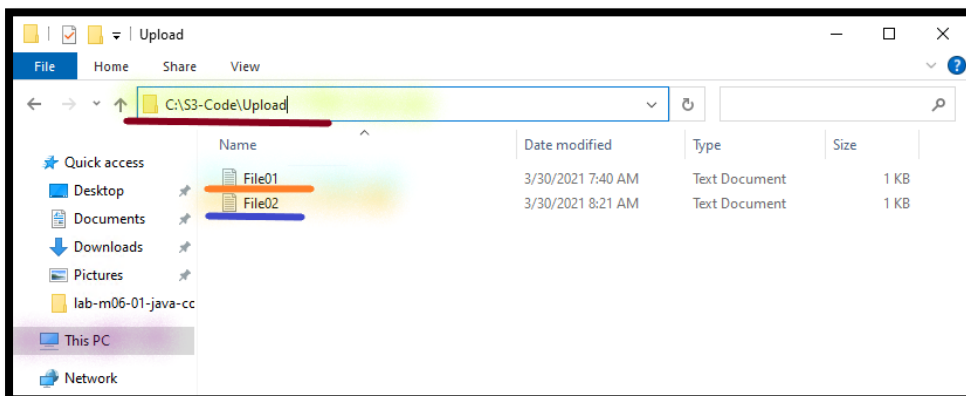
i. From the **File explorer**:

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

b) **Copy** the **Upload** folder in the **S3-Code** folder.

**Note:** **Upload** folder is available with the **Lab manual**.

**Note:** You can view the **File01.txt** and **File02.txt** file **content**.



## Step 2: Create New Bucket

27. From the **IntelliJ IDE**.

a. **Expand** the **awss3m06** java project.

i. **Expand** the **src/main/java** resource path.

a) **Select** the **com.aws.s3** java project.

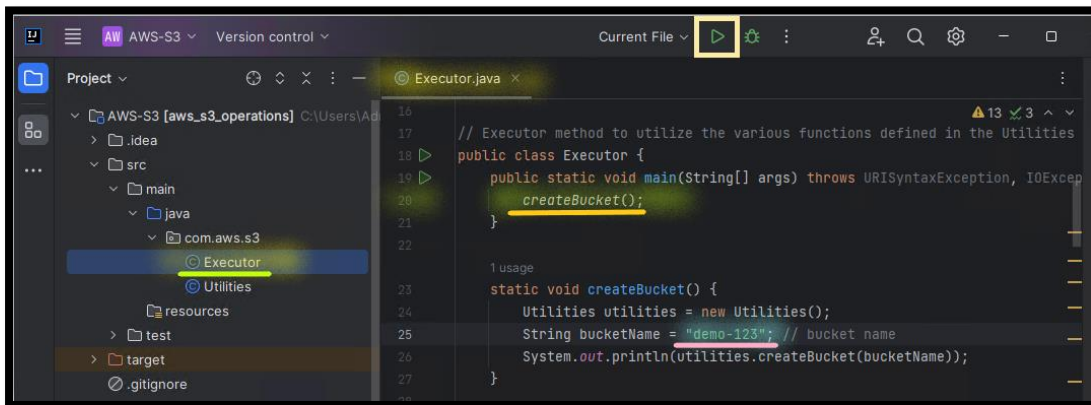
1) **Double-click** on the **Executor.java** java file.

28. 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 (" ").

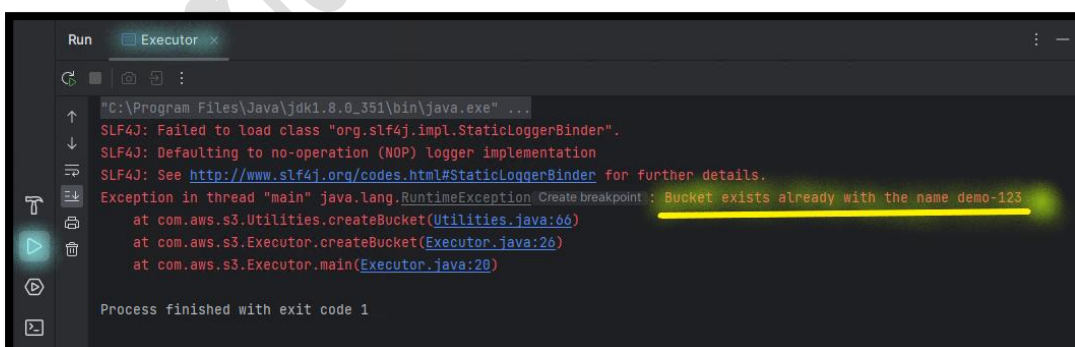
- i. **Execute** the **Run Executor**.



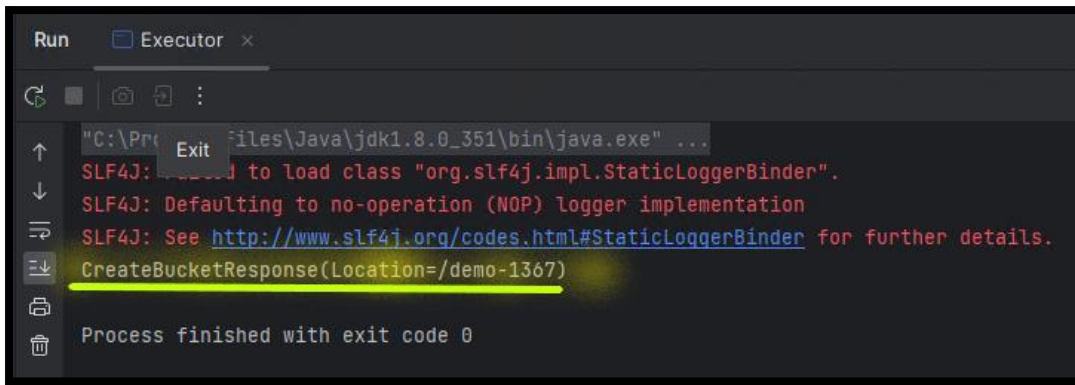
**Note:** If bucket **Created successfully**, in the **Console**, you will see the **"CreateBucketResponse"** message.

**Note:** If bucket **Already exist**, in the **Console**, you will see the **"Bucket exist already with the name demo-123"** message.

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



**Note:** **Replace 123** to make the **bucket name unique** and create the Bucket again.



```
Run Executor x
"C:\Program Files\Java\jdk1.8.0_351\bin\java.exe" ...
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
CreateBucketResponse(Location=/demo-1367)
Process finished with exit code 0
```

### Step 3: Create New [Second] Bucket

29. 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`.
  - i. **Execute** the **Run Executor**.

**Note:** If bucket **Created successfully**, in the **Console**, you will see the **"CreateBucketResponse"** message.

**Note:** If bucket **Already exist**, in the **Console**, you will see the **"Bucket exist already with the name images-123"** message.

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

### Step 4: List the Existing Buckets

30. From the `Executor.java`.

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

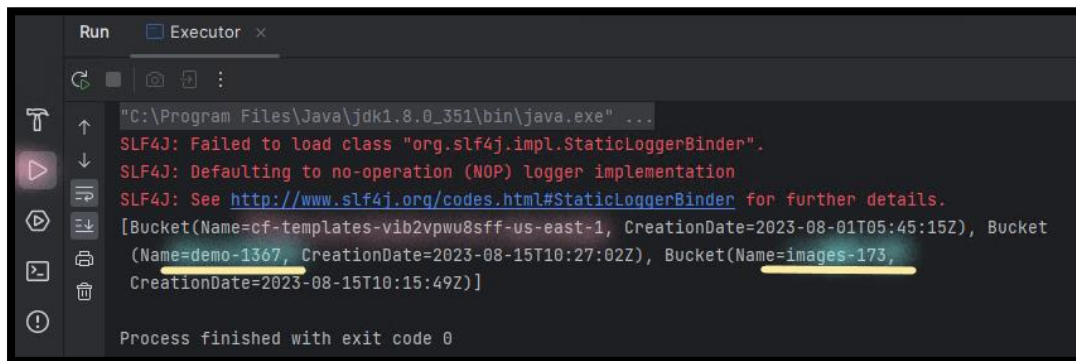
**Note:** Ensure there should be no space between `listExistingBuckets` and **start & end** bracket `()`. It should be like `listExistingBuckets()`.

- i. **Execute** the **Run Executor**.

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



**Note:** You may also see **other buckets** created by CloudFormation.



```
Run Executor x
"C:\Program Files\Java\jdk1.8.0_351\bin\java.exe" ...
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
[Bucket(Name=cf-templates-vib2vpwu8sff-us-east-1, CreationDate=2023-08-01T05:45:15Z), Bucket
(Name=demo-1367, CreationDate=2023-08-15T10:27:02Z), Bucket(Name=images-173,
CreationDate=2023-08-15T10:15:49Z)]
Process finished with exit code 0
```

## Step 5: Delete the Bucket

31. From the **Executor.java**.

- Replace the **listExistingBuckets** method with the **deleteBucket** method in the **line no. 20**.
- Replace the **BUCKET\_NAME** with the **demo-123**, bucket name (*which you have created in the previous step*) in the **line no. 31**.
  - Execute the **Run Executor**.

## Step 6: List the Existing Buckets

32. From the **Executor.java**.

- Replace the **deleteBucket** method with the **listExistingBuckets** method in the **line no. 20**.
  - Execute the **Run Executor**.

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

## Step 7: Upload the New Object (**File01.txt**)

33. From the **Executor.java**.

- 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. 49**.

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

- i. **Execute** the **Run Executor**.

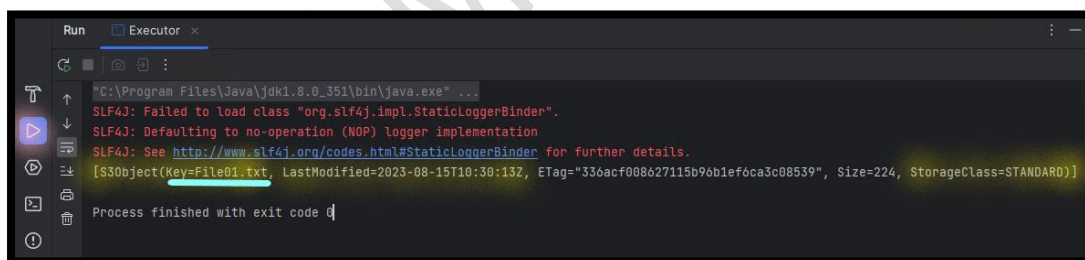
**Note:** If Object uploaded successfully, in the **Console**, you will see the **PutObjectResponse**.

## Step 8: List Bucket Objects

34. 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**.
  - i. **Execute** the **Run Executor**.

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



```
Run Executor x
"C:\Program Files\Java\jdk1.8.0_351\bin\java.exe" ...
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
[S3Object(Key=File01.txt, LastModified=2023-08-15T10:30:13Z, ETag="336acf008627115b96b1ef6ca3c08539", Size=224, StorageClass=STANDARD)]
Process finished with exit code 0
```

## Step 9: Get (Read) Existing Object (File01.txt)

35. 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 defined the file name (**File01.txt**).

- i. **Execute** the **Run Executor**.

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

```

Run Executor x
"C:\Program Files\Java\jdk1.8.0_351\bin\java.exe" ...
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can
manage a project's build, reporting and documentation from a central piece of information.
{}
Process finished with exit code 0
  
```

## Step 10: Upload New Object with Metadata (**File02.txt**)

36. 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 defined upload **file path** and file name (**File02.txt**) which you have copied in the S3-Code folder in C drive and **Project** and **Owner metadata**.

- i. **Execute** the **Run Executor**.

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

## Step 11: Get (Read) Existing Object (**File02.txt**)

37. From the **Executor.java**.

- a. **Replace** the **uploadObjectWithMetadata** method with the **getObject** method in the **line no. 20**.
- b. **Replace** the **File01.txt** with the **File02.txt**, object name in the **line no. 78**.

- i. **Execute** the **Run Executor**.

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

```

Run Executor x
"C:\Program Files\Java\jdk1.8.0_351\bin\java.exe" ...
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
To use the AWS SDK for Java in your project, you need to declare it as a dependency in your project's pom.xml file.
The AWS Java SDK for Amazon S3 module holds the client classes that are used for communicating with Amazon Simple Storage Service,
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3</artifactId>
  <version>2.13.9</version>
</dependency>

{owner=Ahmad, project=HRMS}
Process finished with exit code 0
  
```

## Step 12: Update Metadata with the Existing Object (**File01.txt**)

38. From the **Executor.java**.

- Replace** the **getObject** method with the **updateMetadata** method in the **line no. 20**.
- 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**.

- Execute** the **Run Executor**.

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

## Step 13: Get (Read) Existing Object (**File01.txt**)

39. From the **Executor.java**.

- Replace** the **updateMetadata** method with the **getObject** method in the **line no. 20**.
  - Replace** the **File02.txt** with the **File01.txt**, object name in the **line no. 78**.
- Execute** the **Run Executor**.

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

## Step 14: Create Pre-Signed URL

40. From the **Executor.java**.

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

- Execute** the **Run Executor**.

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

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

```

Run Executor x
"C:\Program Files\Java\jdk1.8.0_351\bin\java.exe" ...
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Presigned URL: https://images-123.s3.amazonaws.com/File02
.txt?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20230815T104009Z&X-Amz-SignedHeaders=host&X-Amz-Expires=1800&X-Amz-Credential=AKIA...
https://images-123.s3.amazonaws.com/File02.txt?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20230815T104009Z&X-Amz-SignedHeaders=host&X-Amz-Expires=1800&X-Amz-Credential=AKIA...
Process finished with exit code 0

```

## Step 15: Get (Read) Object using Pre-Signed URL

41. From the **Executor.java**.

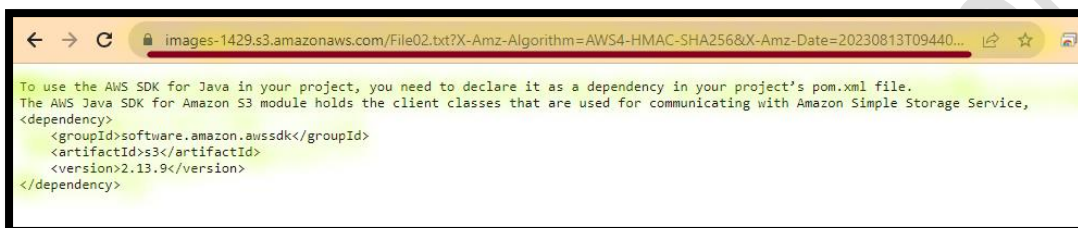
- Replace** the **getPresignedUrl** method with the **getPresignedUrlData** method in the **line no. 20**.
- Replace** the **PRE-SIGNED\_URL** with the **Pre-Signed URL** which you have created in the previous step in the **line no. 93**.
  - Execute** the **Run Executor**.

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

## Step 16: Access the Object from Browser

42. From your **Local Desktop/ Laptop**, open the **Browser**, write **Pre-Signed\_URL** of **File02.txt**, to access the **Object content**.

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



## Task 5: Install AWS Toolkit for IntelliJ

### Step 1: Close the Maven Project

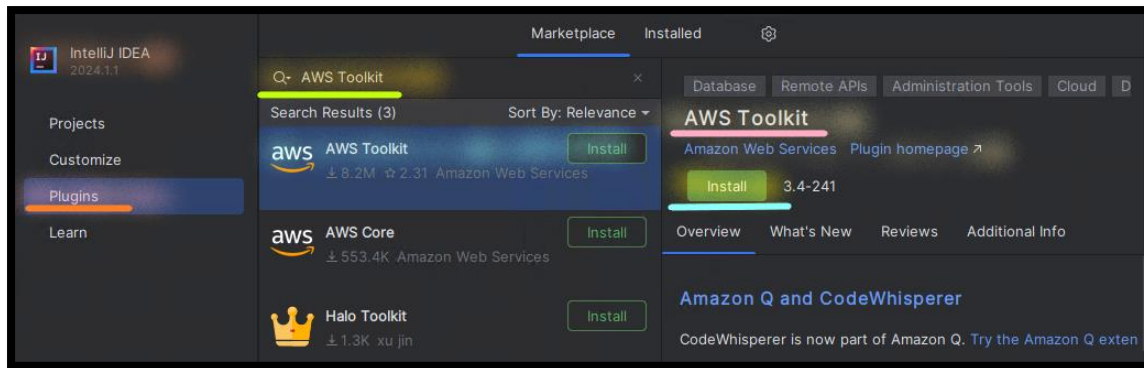
43. From the **IntelliJ IDE**.
- a. Select the **File**.
    - i. Select **Close Project**.

### Step 2: Install the AWS Toolkit

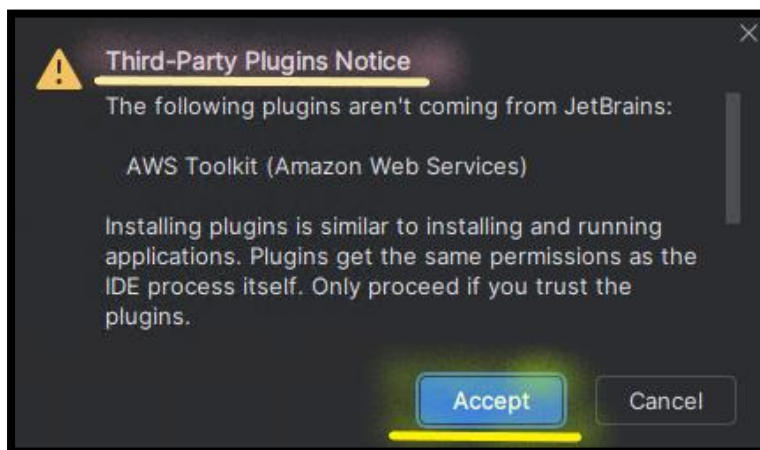
44. From the **IntelliJ IDE**.
- a. Select **Plugins**.
    - i. In the **Search box**, type **AWS Toolkit**.

**Note:** You can see the **AWS Toolkit**.

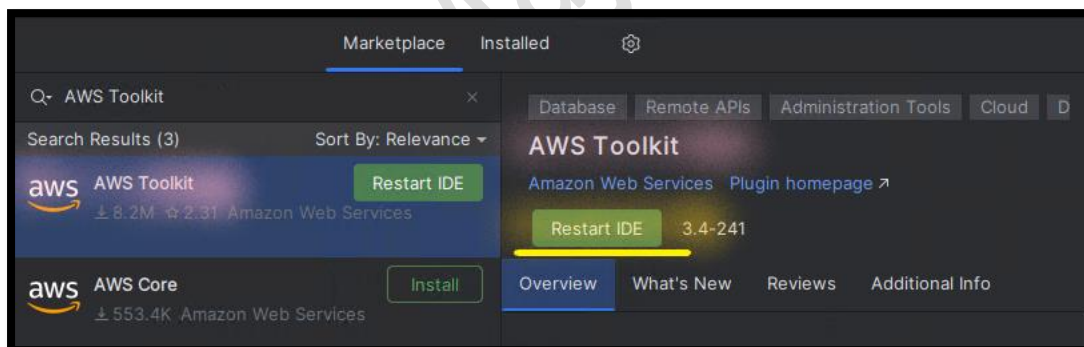
- a) Select **Install**.



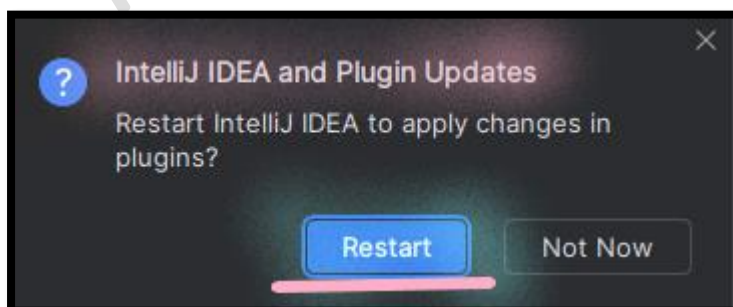
1) **Third-Party Plugins Notice:** Select **Accept**.



2) Select **Restart IDE**.



I. Select **Restart**.



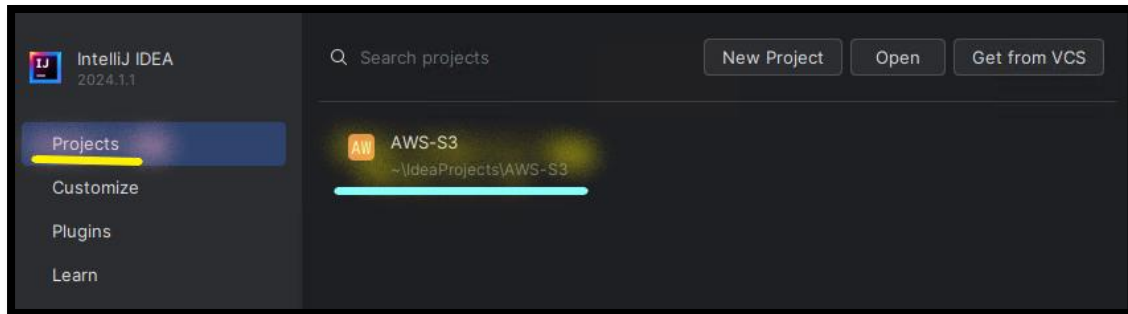


### Step 3: Install the AWS Toolkit

45. From the **IntelliJ IDE**.

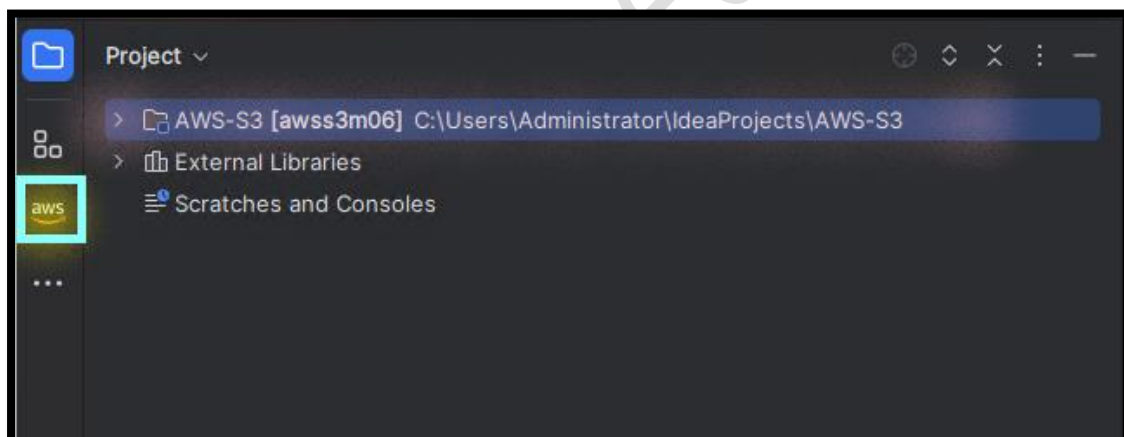
a. Select **Project**.

i. Select **AWS-S3**.



**Note:** You can see the **AWS-S3** project.

a) Select **AWS Toolkit**.

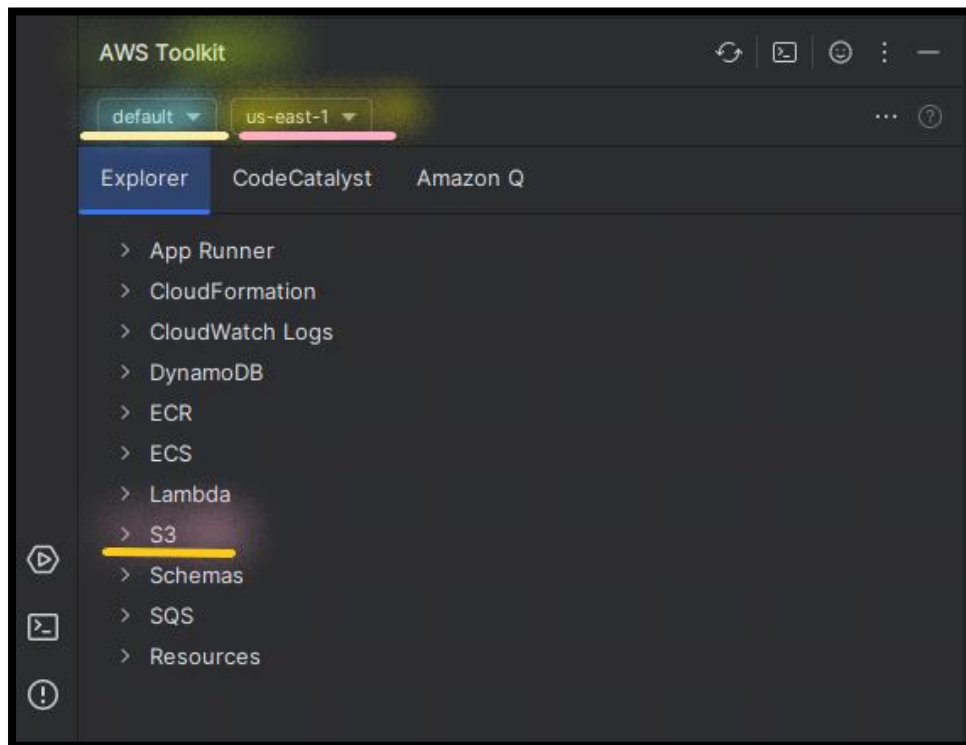


**Note:** You can see the **AWS Toolkit**.

**Note:** You can see the **Default** credentials profile.

**Note:** You can see the **Region**.

**Note:** You can see the **AWS Services**.



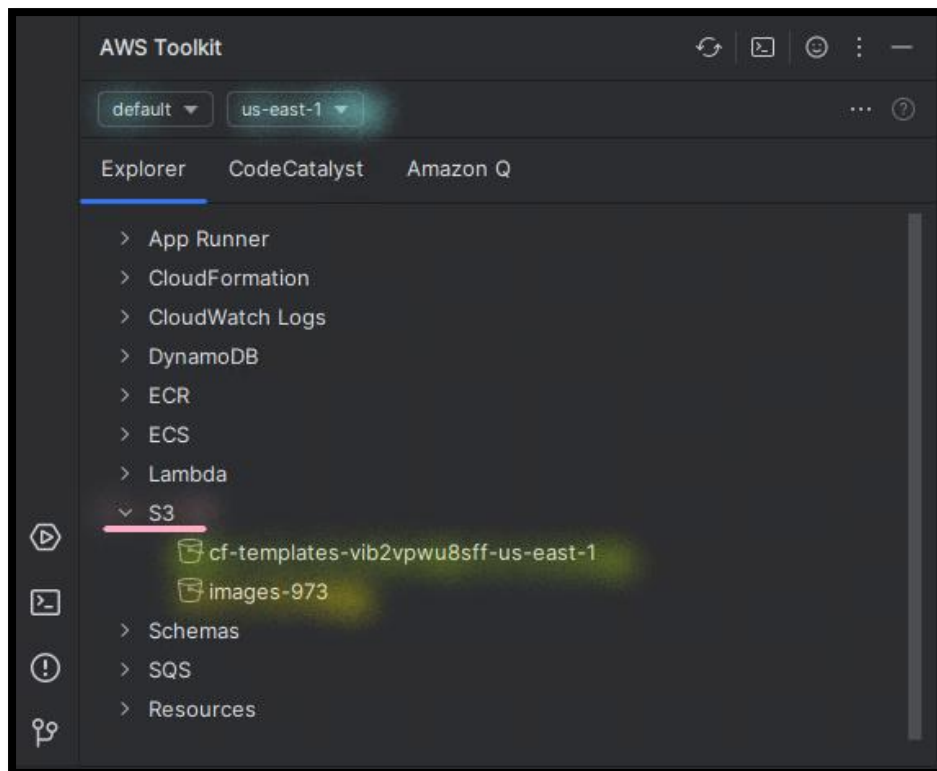
b) Expand **S3**.

**Note:** You can see the **Buckets**.

**Note:** You can see the **images** bucket.

**Note:** You can also see the **images** bucket and **another bucket** created via **cloudformation**.





**Note:** You can also create and upload new objects from AWS Toolkit.

## Task 6: Close the Project

### Step 1: Close the Maven Project

46. From the **IntelliJ IDE**.
  - a. Select the **File**.
    - i. Select **Close Project**.

### Step 2: Delete the Stack

47. In the **AWS Management Console**, on the **Services** menu, search and select **CloudFormation**.
48. Select **Stack**.
  - a. Select **Dev-Instance-JV**.
    - i. Select **Delete**.
      - i. Select **Delete stack**.

## Step 2: Delete the Buckets

49. In the **AWS Management Console**, on the **Services** menu, click **S3**.

50. Select **Buckets**.

### Delete Images-123 Bucket

- a. Select **images-123** bucket.
  - i. Select **Empty**.
    - a) **Type permanently delete** to delete all the objects.
    - b) Select **Empty**.
    - c) Select **Exit**.
  - b. Select **images-123** bucket.
    - i. Select **Delete**.
      - a) Type **images-123** bucket name to delete bucket.
      - b) Select **Delete bucket**.

### Delete Website-123 Bucket

- c. Select **images-123** bucket.
  - i. Select **Empty**.
    - a) **Type permanently delete** to delete all the objects.
    - b) Select **Empty**.
    - c) Select **Exit**.
  - d. Select **images-123** bucket.
    - i. Select **Delete**.
      - a) Type **images-123** bucket name to delete bucket.
      - b) Select **Delete bucket**.