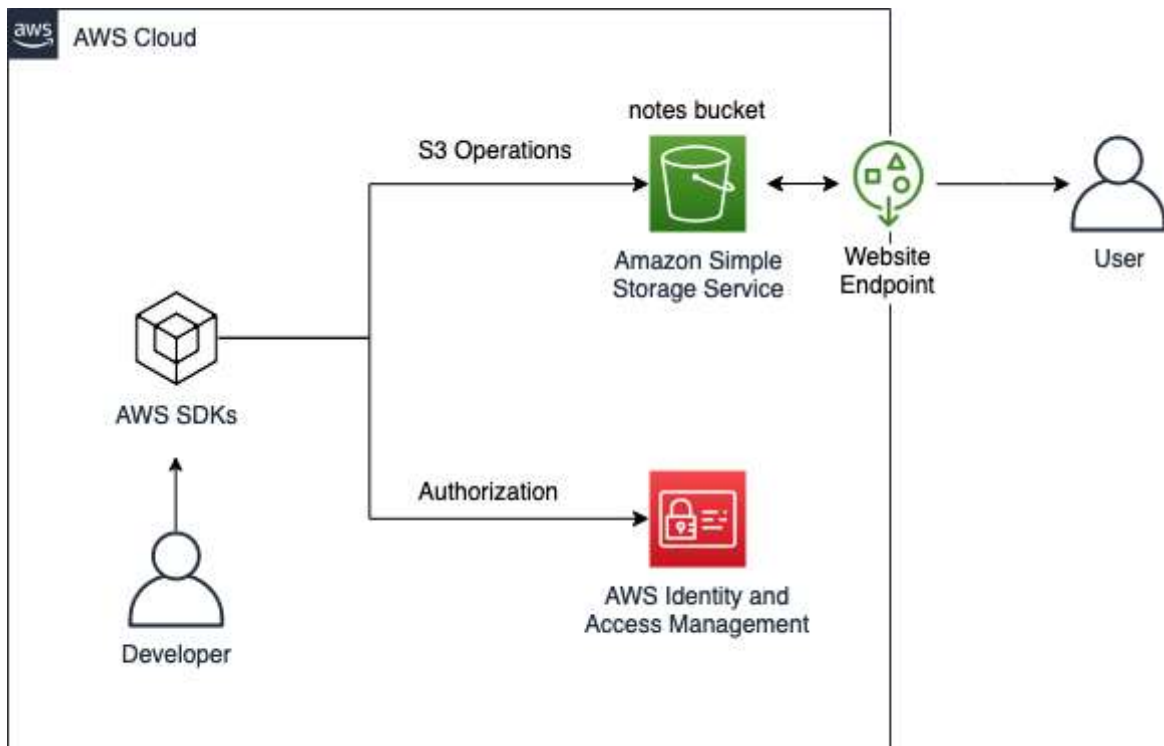


Lab 2 (Java) - Develop Solutions Using Amazon S3

Lab Overview

Now that you are familiar with how the developer environment is set up, you will start developing a portion of the application that uses Amazon Simple Storage Service (Amazon S3) as a storage service and web hosting service. You will create an Amazon S3 bucket to store notes as objects used in your application. Acceptable formats for the application notes are: csv, json, and eventually mp3 audio files in later labs. You will learn how to process the data in your files and manage updates. Next, you will use Amazon S3 to host your static website. You will use the clues provided to complete the missing pieces of the code at TODO placeholders. Once the error free code is filled in, you will run the program and review the results.



After completing this lab, you will be able to:

- Interact with Amazon S3 programmatically using AWS SDKs and the AWS CLI.
- Create a bucket using waiters and verify service exceptions codes.
- Build requests needed to upload an Amazon S3 object with metadata attached.
- Build requests to download an object from the bucket, process data, and upload the object back to bucket.
- Configure a bucket to host the website and sync the source files using the AWS CLI.

Objectives

Duration

This lab requires approximately **45 minutes** to complete.


Task 0: Connect to your development environment

Task 0: Connect to your development environment

In this task, you will connect to your development environment.

1. Choose a method to connect to the development environment from the options below:

- [Connect to Your Windows Dev Instance using Apache Guacamole](#)
- [Connect to Your Windows Dev Instance from a MacOS Machine using RDP](#)
- [Connect to Your Windows Dev Instance from a Windows Machine using RDP](#)

 **NOTE:** When lab instructions in subsequent sections require you to issue commands, use the **IntelliJ Terminal**.

Consider: This lab is designed for both experienced and newer developers:


- For more experienced developers who enjoy a challenge, there are **High-Level Instructions** before each task that should provide you enough information to help you complete the task.
- Once you complete the updates test your code to ensure it works, troubleshoot if needed, and then move on to the next task.
- For newer developers, there are **Detailed Instructions** to guide you through each step of the lab.

Task 0.1 - Open and configure IntelliJ

2. Open **IntelliJ** from the desktop icon

- From the **LICENSE AGREEMENT** window, select the check box that says, **I confirm that I have read and accept the terms of this User Agreement** and then choose **Continue**
- From the **DATA SHARING** window, choose **Don't Send**

3. From the **Welcome** screen, choose **Open** > navigate to **C:\code\java** > select **pom.xml** > **OK** > **Open as Project** > ☒ **Trust projects in C:/code** > **Trust Project**

 **CAUTION:** Wait for the underlying processes to complete as they download plugins/dependencies and other tasks that can cause errors if you do not wait for those tasks to complete. You can observe this status at the bottom of the screen.

4. Close the **Tip of the Day** window.

5. You will now assign the SDK to the project. Choose **File > Project Structure... > Project SDK:** 11 Amazon Corretto version 11.0.11 > **OK**

Task 1: Create a new Amazon S3 bucket

Task 1: Create a new Amazon S3 bucket

In this task, you will use the developer environment and IDE. You will develop code to create a notes bucket. The bucket requires a unique name and for you to specify the appropriate region. You will check if the bucket already exists, and that the assumed role has the right permissions using Amazon S3 service exception codes. To help save time, we provide basic coding components for each script, and you will add the missing code.

Pseudocode: *createBucket.java*

- **GetInputs** method initializes variables including bucket name and lab region based on the `config.properties` file.
- Creates an Amazon S3 client and closes an Amazon S3 client.
- **bucketExisting** method checks to see if an existing bucket name exists. It uses the head bucket operation and response codes to make this determination.
- **createBucket** method creates your Amazon S3 bucket. It builds create bucket request, runs createBucket operations, and defines Amazon S3 waiter to check if the bucket is ready for use.

You need to complete all the **TODO** sections of the `C:\code\java\src\main\java\dev.labs.s3\createBucket.java` file with the proper code to successfully create the Amazon S3 bucket using AWS SDK for Java.

High-Level Instructions:


- Open `C:\code\java\src\main\resources\config.properties`. Update the variable `lab_region` with the region **US West (Oregon)** value shown to the left of these instructions and the `bucket_name` variable with a value consisting of `lab2-notes-bucl` followed by your

first and last initial and your zip code.

- Complete the **TODO 1** to create an Amazon S3 service client.
- Complete the **TODO 2** to verify if the generated bucket name is valid to use by creating a HeadBucket object.
- Complete the **TODO 3** to create an Amazon S3 waiter.
- Complete the **TODO 4** to create the build request to create the bucket.
- Test/build/run the script, and resolve any errors.
- Verify the bucket is created using **Java** or the **aws cli**.

Detailed Instructions:

Task 1.1: Create an Amazon S3 service client

 **NOTE:** Review the **Java** [documentation](#) for creating an Amazon S3 bucket to learn about the **s3.Bucket** class.

1. Open the **crudOperations** project. From the **Project** pane, expand listing under **java [crudOperations] C:\code\java > src > main > resources > config.properties** .

2. Update the following variables and **save the file** when finished:

- Variable for **lab_region** to the **Region** value to the left of these  instructions.


Example:

```
lab_region=us-west-2
```

- Variable for **bucket_name** pre-pended with `lab2-notes-buck`
- Then your **first** and **last** initials followed by your **zip code** (or any 5 numbers).

Example:

```
bucket_name=lab2-notes-bucket-mh75135
```

 **NOTE:** These variables will be referred to by additional files used throughout this lab. You can close this file as it will not be used again in this lab.

3. From the **Project** pane, open **java [crudOperations] C:\code\java > src > main > java > dev.labs.s3 > createBucket**

4. **Copy/Paste:** Replace the code in **TODO 1** with the correct snippet below to create an Amazon S3 client.

- Choice A

Copy Code

```
S3Client s3 = S3Client.builder()  
    .build();
```

- Choice B

Copy Code

```
S3Client s3 = S3Client.builder()  
    .region(labRegion)  
    .build();
```

Answer: You can find the solution to this step in the [TODO 1 Solution](#) section at the bottom of these instructions.

Task 1.2: Create a HeadBucket object

Next, you need to determine if the bucket name you wish to create already exists. You can refer to the [HeadBucketResponse Class documentation](#) to see how this should be constructed.

5. **Copy/Paste:** Replace the code in **TODO 2** with the correct snippet below:

- Choice A

Copy Code

```
HeadBucketRequest request = HeadBucketRequest.builder()  
    .bucket(bucketName);
```

- Choice B

Copy Code

```
HeadBucketRequest request = HeadBucketRequest.builder()  
    .bucket(bucketName)  
    .build();
```

Answer: You can find the solution to this step in the [TODO 2 Solution](#) section at the bottom of these instructions.

Task 1.3: Create an Amazon S3 waiter

6. Update the next section of the script to create an [Amazon S3 Waiter](#).

7. **Copy/Paste:** Replace the code in **TODO 3** with the correct snippet below:

- Choice A

Copy Code

```
S3Waiter s3Waiter = s3Client.waiter();
```

- Choice B

Copy Code

```
S3Waiter s3Waiter = s3Resource.waiter();
```

Answer: You can find the solution to this step in the [TODO 3 Solution](#) section at the bottom of these instructions.

Task 1.4: Build a request to create bucket

The last step before running the code is to create a [build request](#) to create the Amazon S3 bucket.

8. **Copy/Paste:** Replace the code in **TODO 4** with the correct snippet below and save your changes.

- Choice A

Copy Code

```
CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
    .build.bucket(bucketName)
```

- Choice B

Copy Code


```
CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
    .bucket(bucketName)
    .build();
```

Answer: You can find the solution to this step in the [TODO 4 Solution](#) section at the bottom of these instructions.

Task 1.5: Test your code

Now it is time to run the code to test the script and ensure that it functions as expected. Make sure there are no syntax errors and that you have saved the changes to the `createBucket` file.

9. Choose **Build > Recompile 'createBucket.java'**

 **NOTE:** Review the status bar at the bottom and verify that it finishes **Recompiling** before proceeding.

10. **Command:** Use **Maven** with the following command from the terminal:

Copy Code

```
mvn -q exec:java -Dexec.mainClass="dev.labs.s3.createBucket"
```

✓ Expected output:


```
```txt
```

```
**** This is OUTPUT ONLY. ****
```

```
C:\code\java>mvn -q exec:java -
Dexec.mainClass="dev.labs.s3.createBucket"
```

```
Head Bucket operation...
No such bucket exists.
```

```
Creating bucket: lab2-notes-bucket-mh75135
Waiting on HeadBucketResponse()
Bucket "lab2-notes-bucket-mh75135" is ready.
```
```

 **NOTE:** If you were to run it again, it would show that the bucket already exists and would not try to create a new one.

✓ Expected output:

```
*****
**** This is OUTPUT ONLY. ****
*****
C:\code\java>mvn -q exec:java -
Dexec.mainClass="dev.labs.s3.createBucket"

Head Bucket operation...
    This bucket already exists!
```

Congratulations! You have successfully created a bucket for use in this lab.

Task 2: Upload an object to Amazon S3



Task 3: Process the data from an object stored in Amazon S3



Task 3: Process the data from an object stored in Amazon S3

In this task, you will develop a program to process the notes from the file `notes.csv` which you uploaded to your `lab2-notes-bucket-<>` bucket in the previous task. You will download the `notes.csv` object from the Amazon S3 bucket to your local folder. You will read the data

from the file and convert it into JSON format so that this file can be used to load data into the database in the proceeding labs. Once converted, you will upload it back to the bucket. You will read the response and output the tag information to verify the upload is successful.

Pseudocode: *processObject.java*

- **getInput** method initializes variables including bucket name and the lab region based on the `config.properties` file.
 - Update the **convertS3Object** method to do the following:
 - Build the request using bucket name and key to download object
 - Get response in bytes
 - Write to a local file on your disk
 - Convert the downloaded notes to JSON format, write to a new file, and catch exceptions

High-Level Instructions:

- Complete **TODO 7** to build and run the request for **getObject** using **object key name** and **bucket name** as parameters.
- Complete **TODO 8** to build and run the request for **putObject** using file path, file name, metadata, and bucket name as parameters.
- Test/build/run the script, and resolve any errors. It will also read the **eTag** from the response body and display on the screen using catch for `S3Exceptions` for any errors while uploading.
- Download the `notes.json` file to confirm the contents have been converted from the **CSV** format to the **JSON** format.

Detailed Instructions:

Task 3.1: Build a GetObjectRequest

You will now create a **getObjectRequest** to download the `notes.csv` file from your bucket.

1. From the **Project** pane, expand listing under **java [crudOperations]** `C:\code\java > src > main > java > dev.labs.s3` and open the `processObject` file.
2. **Copy/Paste:** Once you have reviewed the documentation, you should be able to choose the correct code snippet to replace the code in **TODO 7**.

- Choice A

Copy Code

```
GetObjectMetadataRequest getObjectMetadata = GetObjectMetadataRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .build();
```

- Choice B

[Copy Code](#)

```
GetObjectRequest getObjectRequest = GetObjectRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .build();
```

Answer: You can find the solution to this step in the [TODO 7 Solution](#) Solution section at the bottom of these instructions.

Task 3.2: Build a PutObjectRequest

In this task, you will build the request to upload the `notes.json` file that was just converted from the **CSV** format to the **JSON** format.

3. Continue editing the `C:\code\java\processObject` file.

4. **Copy/Paste:** Review the code snippets below and choose the correct one to replace the code in the **TODO 8** section and save the changes.

- Choice A

[Copy Code](#)

```
s3.getObject(putObject,
    RequestBody.fromFile(Paths.get(objectNewKey)));
```

- Choice B

[Copy Code](#)


```
s3.putObject(putObject,
    RequestBody.fromFile(Paths.get(objectNewKey)));
```

Answer: You can find the solution to this step in the [TODO 8 Solution](#) section at the bottom of these instructions.

Task 3.3: Test your code

Now it is time to run the code to test the script and ensure it functions as expected. Make sure there are no syntax errors and that you have saved the changes to the `processObject` file.

5. Choose **Build > Recompile 'processObject.java'**

 **NOTE:** Review the status bar at the bottom and verify that it finishes **Recompiling** before proceeding.

6. **Command:** Use **Maven** from the command line with the following command from the **terminal**:

Copy Code

```
mvn -q exec:java -Dexec.mainClass="dev.labs.s3.processObject"
```

Expected output:

```
*****  
**** This is OUTPUT ONLY. ****  
*****
```

```
C:\code\java>mvn -q exec:java -  
Dexec.mainClass="dev.labs.s3.processObject"
```

```
Retrieving notes.csv from bucket...
```

```
Object downloaded from Amazon S3 is written to:  
C:\code\java\localFile.csv
```

```
Converting notes.csv to json format...
```

```
Converted file is written to: C:\code\java\notes.json
```

```
Uploading notes.json file to Amazon S3...
```

```
Object uploaded. Tag information:"82e4c6abab63ec87157f6a2372a99bcd"
```

Congratulations! You have successfully created and object in the Amazon S3 bucket.

Task 4: Configure static website hosting on an Amazon S3 bucket

Task 4: Configure static website hosting on an Amazon S3 bucket

In this task, you will use AWS CLI commands to configure your lab2-notes-bucket-<> bucket to host the website for your application. You will use the provided index and error documents provided to you to configure the website. You will set the bucket policy so that web pages can be accessed over the internet. You will upload index and error files to Amazon S3 prior to accessing the site.

High-Level Instructions:

- Static files needed for your website are provided in the **html** folder.
- Run **s3api put-bucket-website** with bucket name and website.json (index, error, and routing http/https) as parameters.
- Run **s3api put-bucket-policy** with bucket name and policy.json as parameters to grant access permissions for your objects.
- Run **s3 sync** with the bucket name and folder as parameters to copy your HTML files.
- Identify the Amazon S3 website link for your bucket and open it in your browser to review files.

Detailed Instructions:

Task 4.1: Configure website hosting

You will now review the **website.json** file and apply this configuration file to setup your Amazon S3 static website.

1. Open the `C:\code\java\html\website.json` file.

This file specifies the values to be used for the `IndexDocument` and the `ErrorDocument`.

2. Verify the files `index.html` and `error.html` exist in the `html` directory.
3. **Command:** Run the following command to change to the `C:\code\java` directory:

Copy Code

```
cd C:\code\java
```

✅ Expected Output:

None, unless there is an error.

4. **Command:** Run the command below to get the value to be able to set the **myBucket** variable to your `lab2-notes-bu` name, which will be used in the next command:

Copy Code

```
aws s3api list-buckets --output text --query "Buckets[?contains(Name, `lab2-notes-bucket-`)  
== `true`] | [0].Name"
```

✅ Expected output:

```
*****  
**** This is OUTPUT ONLY. ****  
*****
```

```
lab2-notes-bucket-<initialsAndZipCode>
```

5. **Command:** Copy the value for the bucket name from the previous command and add it to run the command below to set the **myBucket** variable:

Copy Code

```
set myBucket=<bucketName>
```

Example:

```
set myBucket=lab2-notes-bucket-mh75135
```

✅ Expected Output:

None, unless there is an error.

6. **Command:** Run the command below to configure the website. The documentation for [put bucket website](#) specifies the configuration of the Amazon S3 website for your bucket.

📌 **NOTE:** The **myBucket** variable will be used for this command to specify your bucket name automatically.

Copy Code

```
aws s3api put-bucket-website --bucket "%myBucket%" --website-configuration  
file://html/website.json
```

✅ Expected output:

None, unless there is an error.

Task 4.2: Update and add a bucket policy


The bucket policy has already been created on your behalf. This policy grants the permission to read the objects in your bucket. Your only task is to update the **bucket name**.

7. From IntelliJ, open `C:\code\java\html\policy.json` and update the value for `<my-bucket>` to your bucket's name and save the changes to the file.

Example:

```
"arn:aws:s3:::lab2-notes-bucket-mh75135/*"
```

8. **Command:** Using the same **Terminal** session, run the following command to [put the bucket policy](#):

 **NOTE:** The **myBucket** variable will be used again for this command to specify your bucket name automatically.

Copy Code

```
aws s3api put-bucket-policy --bucket "%myBucket%" --policy file://html/policy.json
```

 **Expected output:**

None, unless there is an error.

9. Run the command below to change directories into the **html** directory:


Copy Code

```
cd C:\code\java\html
```

 **Expected Output:**

None, unless there is an error.

10. **Command:** Run the command below to sync the files in the **html** folder with your bucket. This includes the `index.html` and `error.html` files along with any images used in those webpages.

 **NOTE:** The **myBucket** variable will be used again for this command to specify your bucket name automatically.

Copy Code

```
aws s3 sync . s3://"%myBucket%"
```

 **Expected output:**

```
*****  
**** This is OUTPUT ONLY. ****  
*****
```

```
C:\code\java\html>aws s3 sync . s3://lab2-notes-bucket-mh75135
```

```
upload: .\error.html to s3://lab2-notes-bucket-mh75135/error.html
```

```
upload: .\index.html to s3://lab2-notes-bucket-mh75135/index.html
```

```
upload: .\policy.json to s3://lab2-notes-bucket-mh75135/policy.json
```

```
upload: .\website_redirect.json to s3://lab2-notes-bucket-mh75135/website_redirect.json
```

```
upload: .\website.json to s3://lab2-notes-bucket-mh75135/website.json
```

Task 4.3: Test your website

Now it is time to verify that you can access your Amazon S3-based website.

11. Run the command below to create the **region** variable after setting the **LabRegion value** to the value us-west-2 the left of these  instructions:

Copy Code

```
set region=<LabRegion>
```

Expected Output:

None, unless there is an error.

12. Review the [Amazon S3 Website Endpoints](#) documentation to determine which of the two commands below to run to generate the URL to access your website based on your region value to the left of these instructions:

 **NOTE:** Depending on the endpoint, the URL with contain **s3-website-** or **s3-website.** in the URL.

- **Choice A:**

Copy Code

```
echo. && echo You can now access the website using the following URL... && echo. && echo  
http://%mybucket%.s3-website-%region%.amazonaws.com
```

- **Choice B:**

Copy Code

```
echo. && echo You can now access the website using the following URL... && echo. && echo  
http://%mybucket%.s3-website.%region%.amazonaws.com
```

✓ Expected Output:

```
*****  
**** This is OUTPUT ONLY. ****  
*****
```

You can now access the website using the following URL..

`http://lab2-notes-bucket-iii-00000.s3-website-us-west-2.amazonaws.com`

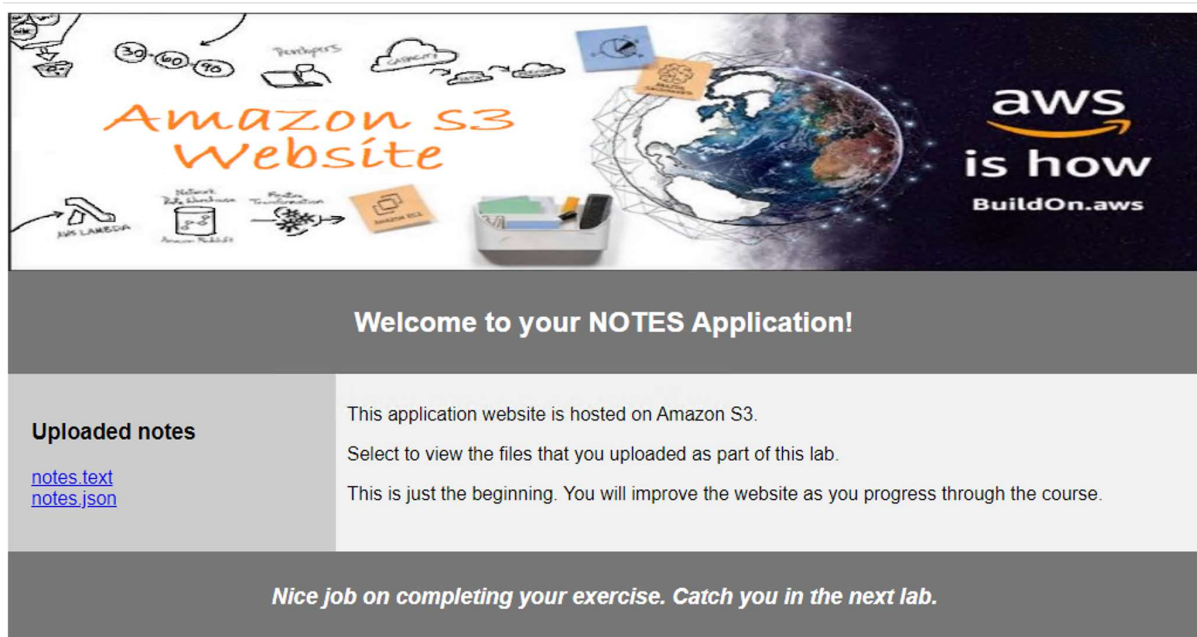
Or...

You can now access the website using the following URL..

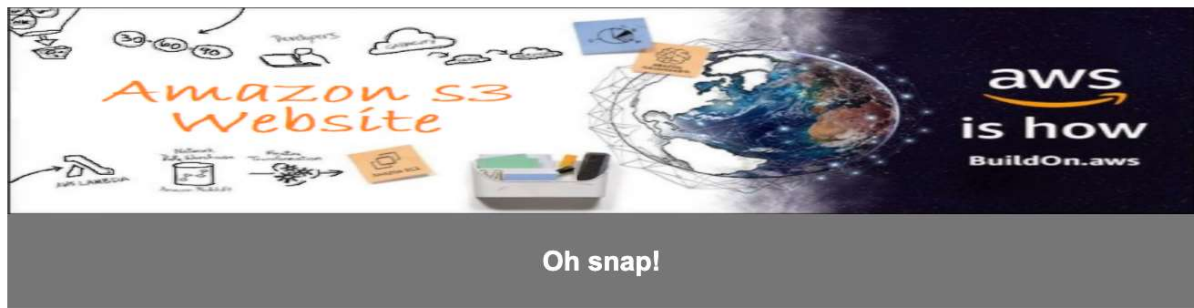
`http://lab2-notes-bucket-iii-00000.s3-website.us-west-2.amazonaws.com`

13. **Copy/Paste:** Copy the URL returned at the end of the command, open a new browser tab, paste the URL, and press **ENTER**.

- You should see the following for your **index.html** page:



- You should see the following for your **error.html** page:



The file you are looking for is not found on this site.

Verify the link and try again.

Congratulations! You have successfully configured Amazon S3 to function as a static web server.


Optional Challenge: Configure static website hosting on an Amazon S3 bucket using the AWS SDK for Java

Optional Challenge: Configure static website hosting on an Amazon S3 bucket using the AWS SDK for Java

In this task you will develop a program to configure your bucket to host the website for your application. You will use the index and error documents provided to configure the website. You will set the bucket policy so web pages can be accessed over the internet. You will upload index and error files to Amazon S3 prior to accessing the site.

High-Level Instructions:

- Assign your bucket name, index, and error pages names.
- Create and open a Amazon S3 service client.
- Build the request for WebsiteConfiguration using index and error pages as parameters.
- Build and run the request for putBucketWebsiteRequest using bucket name and website configuration as parameters.
- Build the request GetBucketWebsiteRequest and run getBucketWebsite to read bucket configuration. Catch any S3Exceptions while uploading.

 **NOTE:** If you need help along the way, you can refer to the solution file named `hostS3Website_` in the **solutions** folder.

Code Challenge Solutions

TODO 1 Solution

- Choice A is not correct because it is missing the **region** property.
- Choice B is the correct code snippet .**

Copy Code

```
S3Client s3 = S3Client.builder()
    .region(labRegion)
    .build();
```

[Return to the instructions](#)

TODO 2 Solution

- Choice A is incorrect because the **build** parameter is missing.
- Choice B is the correct code snippet.**

Copy Code

```
HeadBucketRequest request = HeadBucketRequest.builder()
    .bucket(bucketName)
    .build();
```

[Return to the instructions](#)

TODO 3 Solution

- Choice A is the correct code snippet.**

Copy Code

```
S3Waiter s3Waiter = s3Client.waiter();
```

- Choice B is incorrect because the syntax specifies an **S3Resource.waiter** which does not exist.

[Return to the instructions](#)

TODO 4 Solution

- Choice A is incorrect because the syntax is incorrect.
- **Choice B is the correct code snippet.**

Copy Code

```
CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
    .bucket(bucketName)
    .build();
```

[Return to the instructions](#)

TODO 5 Solution

- Choice A is incorrect because it is using `metadata.get` when it should be calling `metadata.put`.

Choice B is the correct code snippet.

Copy Code

```
Map<String, String> metadata = new HashMap<>();
metadata.put("x-amz-meta-myVal2", "lab2-testing-upload");
```

[Return to the instructions](#)

TODO 6 Solution

- Choice A is incorrect because it is missing the **.key(objectKey)** property.
- **Choice B is the correct code snippet.**

Copy Code

```
PutObjectRequest putObject = PutObjectRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
```

```
.metadata(metadata)
.build();
```

[Return to the instructions](#)

TODO 7 Solution

- Choice A is incorrect because it gets the metadata for the specified Amazon S3 object without actually fetching the object itself.
- **Choice B is the correct code snippet.**

[Copy Code](#)

```
GetObjectRequest getObjectRequest = GetObjectRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .build();
```

[Return to the instructions](#)

TODO 8 Solution

- Choice A is incorrect because **s3.getObject** is the wrong action.
- **Choice B is the correct code snippet.**

[Copy Code](#)

```
s3.putObject(putObject,
    RequestBody.fromFile(Paths.get(objectNewKey)));
```

[Return to Instructions](#)

RDP Connection Options



RDP Connection Options

Connect to Your Windows Dev Instance using Apache Guacamole

In this task you manually verify the **Public IPv4 address** is running by connecting to a **Apache Guacamole Server** hosted on the Amazon EC2 instance.

1. Search for **EC2** using the **Unified Search Bar** at the top of the AWS Console, and choose the service from the listed results.
2. Choose **Instances** from the Instances sub-menu in the Amazon EC2 navigation bar.
3. Fill the checkbox ☒ for the **Apache Guacamole Server**
4. Scroll down in **Details** tab and copy **Public IPV4 address** value and paste in a **notepad**
5. Copy the **GuacamoleLink** `http://Public IPv4 address` and paste it into a browser replacing *Public IPV4 address* of the **Apache Guacamole Server** copied from the previous task. .
6. Go to the Apache Guacamole sign-in, in the browser and sign in using the following steps:
 - For **Username**, enter: `student`
 - For **Password**, use value for `AWSPa$` .
 - Choose **Log In**.

The connection to your remote instance should start momentarily. Once you open a connection, you will see an image of the Dev instance desktop. You can interact with this image just as you would your normal desktop or any remote desktop client.

You are now connected to your Windows Dev instance in the browser using Guacamole.

 **NOTE:** You will be prompted with a Networks pop-up window asking: **Do you want to allow your PC to be discoverable by other PCs and devices on this network?** Choose **No**.

 **NOTE:** How to copy and paste when using Guacamole browser session?

- Open the Clipboard editor by choosing **Ctrl -> Alt -> Shift** on Windows or **command -> control -> shift** on MacOS.
- Copy your text from lab instructions and paste it to the Clipboard editor of the session.
- Close the Clipboard editor by choosing **Ctrl -> Alt -> Shift** on Windows or **command -> control -> shift** on MacOS.
- Now, you can use regular paste commands in your session from Clipboard. You can also either edit or replace the clipboard for the subsequent copies.

[Return to the instructions](#)

Connect to the Windows Dev Instance from a MacOS Machine

In this section, you connect to a Windows Amazon EC2 instance from your MacOS machine.

In this task you manually verify the **Public IPv4 address** is running by connecting to a **Windows Dev Instance** hosted on the Amazon EC2 instance.

7. Search for using the **Unified Search Bar** at the top of the AWS Console, and choose the service from the listed results.
8. Choose **Instances** from the Instances sub-menu in the Amazon EC2 navigation bar.
9. Fill the checkbox ☒ for the **Windows Dev Instance**
10. Scroll down in **Details** tab and copy **Public IPV4 address** value and paste in a **notepad**
11. Install Microsoft Remote Desktop if it is not already installed. To install, complete the following:
 - At the top of the screen select the Apple icon.
 - Select **About This Mac**.
 - Take note of your MacOS version
 - From the Dock launch **App store**.
 - Search for the following string: Microsoft Remote Desktop .
 - Choose **Install** or **GET** to install the appropriate version.
 - At the time of this writing, the current MacOS version is OSX 10.15.7 and you install Microsoft Remote Desktop Version 10.5.2. The connection instructions will be based on this version.
12. To open **Microsoft Remote Desktop** on the Dock select **Launchpad**, then select **Microsoft Remote Desktop**.
13. To create a new connection, from the top menu select **Connections > Add PC** and fill in the following fields:
 - **PC name:** *Paste the **Public IPV4 address** of the Windows Dev Instance copied from the previous task.*
 - **User account:**
 - **Friendly name:**
 - **Reconnect if the connection is dropped:** *Select this option if not already selected .*
 - Choose **Add**.

14. For **Password**, use value for `AWSPa$`.
15. Go back to your Microsoft Remote Desktop connection window, and open the context menu for the **Windows Dev Instance** connection and choose **Open**
16. You will see the following message, **You are connecting to the RDP host "IP.Address.Listed.Here". There certificate couldn't be verified back to a root certificate. Your connection may not be secure. Do you want to continue?** Choose **Continue**.
17. Once prompted for the password, enter the password that you copied to the clipboard and choose **Continue**.

Result

Your connection to your remote instance should start momentarily. When lab instructions in subsequent sections require a command window, open or use a PowerShell window.

 **NOTE:** You will be prompted with a Networks pop-up window asking: **Do you want to allow your PC to be discoverable by other PCs and devices on this network?** Choose **No**.

[Return to the instructions](#)

Connect to Your Windows Dev Instance from a Windows Machine

In this section, you connect to a Windows Amazon EC2 instance .

In this task you manually verify the **Public IPv4 address** is running by connecting to a **Windows Dev Instance** hosted on the Amazon EC2 instance.

18. Search for `EC2` using the **Unified Search Bar** at the top of the AWS Console, and choose the service from the listed results.
19. Choose **Instances** from the Instances sub-menu in the Amazon EC2 navigation bar.
20. Fill the checkbox ☒ for the **Windows Dev Instance**
21. Scroll down in **Details** tab and copy **Public IPV4 address** value and paste in a **notepad**
22. Open the Remote Desktop Connection application on your computer.
23. Choose the **Start** icon, and choose the **Search** icon. Type in `Remote Desktop` . Choose the application when it appears in the **Programs** list.

24. For **Computer**, paste the **Public IPV4 address** of the *Windows Dev Instance* copied from the previous task.


25. Choose **Connect**.

26. Remote Desktop Connection will prompt you with a login dialog asking for your username and password. By default, the application will use your current Windows username and domain. To change this, choose **Use another account**.

 **NOTE:** On Windows 10, select **More Choices** before choosing **Use a different account**.

27. For your login credentials, use the following values:

- For **User name**, enter:
- For **Password**, use value for .

 **NOTE:** The `\` in the user name is important because it tells Remote Desktop Connection that you are logging in as the local Administrator and not as a domain user.

28. To connect to your instance, choose **OK**. If you receive a prompt that the certificate used to verify the connection was not a known, trusted root certificate, choose **Yes**.

Result

Your connection to your remote instance should start momentarily. When lab instructions in subsequent sections require a command window, open or use a Powershell window.

 **NOTE:** You will be prompted with a Networks pop-up window asking: **Do you want to allow your PC to be discoverable by other PCs and devices on this network?** Choose **No**.

[Return to the instructions](#)

Summary

Congratulations on completing the lab! For the **Java** version you can now:

- Interact with Amazon S3 programmatically using AWS SDKs and the AWS CLI.
- Create a bucket using waiters and verify service exceptions codes.
- Build requests needed to upload an Amazon S3 object with metadata attached.
- Build requests to download an object from the bucket, process data, and upload the object back to bucket.
- Configure a bucket to host the website and sync the source files using the AWS CLI.

Additional Resources

[AWSJavaSDK Documentation](#)

[HeadBucketResponse Class documentation](#)

[Amazon S3 Waiter](#)

[Build Request](#)

[Class ObjectMetadata](#)

[Class PutObjectRequest](#)

[Put Bucket Website](#)

[Put Bucket Policy](#)

End Lab

[Click to go up](#)

Lab Complete