Lab Information

Contents

AWS Services are loading.

aws training and certification

Lab 2 (Python) - Develop Solutions Using Amazon S3

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at *AWS Training and Certification* .

Duration

This lab will require around **45 minutes** to complete.

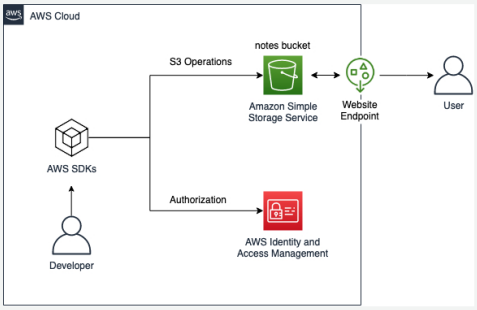
Prerequisites

This lab requires:

- Access to a Microsoft Windows or MacOS notebook computer with a WI-Fi connection.
- An Internet browser such as Chrome, Firefox, or IE9+.
- **Note:** Previous versions of Internet Explorer are not supported.
- **Note:** You can use an iPad or tablet device to access these directions in the lab console.
- **Additional information:** Review additional lab environment specific details in the Appendix.

Overview

Now that you are familiar with setting up a development environment, you will develop a portion of the application that uses Amazon S3 as both a storage and web hosting service. Your application will use Amazon S3 to store files in many formats for different purposes including, csv, json, and eventually mp3 files in later labs. You will also use Amazon S3 to host your static website. Use the clues given to fill out the missing pieces of the code at the TODO placeholders. Once the error free code is filled in, you will run the program and review the results.



OBJECTIVES

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.

Start lab

1. To launch the lab, at the top of the page, choose Start lab .

Caution: You must wait for the provisioned AWS services to be ready before you can continue.

2. To open the lab, choose Open Console .

You are automatically signed in to the AWS Management Console in a new web browser tab.

WARNING: Do not change the Region unless instructed.

COMMON SIGN-IN ERRORS

Error: You must first sign out

To logout, click here

If you see the message, **You must first log out before logging into a different AWS account:**

- Choose the **click here** link.
- Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
- Choose **Open Console** again.

Error: Choosing Start Lab has no effect

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

- Add the lab domain name to your pop-up or script blocker's allow list or turn it off.
- Refresh the page and try again.

Task 1: Create a new Amazon S3 bucket

You will utilize the AWS Cloud9 development environment to complete this task. You will develop code to create a **notes-bucket** with a unique name in the lab region. To save time, basic coding components are provided for each script and you add the missing code.

You will use the AWS SDK for Python (Boto3) documentation to complete the code marked as **TODO** in the scripts provided. The detailed instructions provide you with multiple choice selections.

All scripts in this lab are arranged in the same order to provide consistency so that they will be easier to read. Script dependencies are imported, the main function and any supporting functions are defined, the AWS SDK clients are initialized and then the main function is called. Each function has all needed values passed to ensure they are independent and easy to test and troubleshoot in the future.

Since all logic is processed from the main function, it is wrapped in a try block when called at the end of the script. Errors for the entire function are handled at this level. You can still target specific errors and handle them independently — which happens in the first code challenge.

The **labRepo/create-bucket.py** file is processed in the following order by the **main** function.

- Read the config.ini file and use the settings for all static values.
- **verifyBucketName** verifies that the bucket name is unique and valid.
- **createBucket** creates the Amazon S3 Bucket.
- **verifyBucket** informs the user when the bucket exists.

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.

High-Level Instructions

Note: Make sure the Cloud9 environment is **ready** . You can verify this by checking the folder structure for a folder named **labRepo** and a folder named **Lab-Is-Ready** . If you don't see them, just wait a few moments for the final configurations to complete.

Note: If you see the following message choose **Accept** .

- **.c9/project.settings have been changed on disk**
  - The project settings file (.c9/project.settings) was updated outside the IDE. If you did not make the changes we suggest reviewing the file before accepting. Do you want to accept the new settings?

- In the **labRepo/config.ini** file, replace the bucket_name value *NOTES_BUCKET_NAME* with the **NotesBucketName** value shown to the left of these instructions.
- Finish **labRepo/create-bucket.py** to create an Amazon S3 bucket.
- Complete the **TODO 1** section to create an Amazon S3 service client.
- Complete the **TODO 2** section to verify if the generated bucket name is valid to use.
- Complete **TODO 3** to add the new bucket to Amazon S3 in your lab account.
- Complete **TODO 4** to use a waiter to confirm that the bucket has been created.
- Run the script and resolve any errors.

Detailed Instructions

TASK 1.1: CONFIGURE YOUR ENVIRONMENT AND DEVELOP CODE TO CREATE AN AMAZON S3 SERVICE CLIENT

Your AWS Cloud9 environment has Python and Boto3 pre-installed for you to use.

3. From the **AWS Management Console** , use the **AWS search bar** to search for `Cloud9` and then choose the service from the list of results.

4. On the **Environments** page, next to the **Lab2** environment listing, choose **Open** .

Note: Make sure the Cloud9 environment is **ready** . You can verify this by checking the folder structure for a folder named **labRepo** and a folder named **Lab-Is-Ready** . If you don't see them, just wait a few moments for the final configurations to complete.

Note: If you see the following message choose **Accept** .

- **.c9/project.settings have been changed on disk**
  - The project settings file (.c9/project.settings) was updated outside the IDE. If you did not make the changes we suggest reviewing the file before accepting. Do you want to accept the new settings?

5. In the **Environment** window, expand the **labRepo** folder, and open **config.ini** .

This configuration file has all static values for every script in this lab. The only value you need to modify is the **bucket_name** . You will use a globally unique name for your new Amazon S3 bucket.

6. For the **bucket_name** , replace the *NOTES_BUCKET_NAME* with the **NotesBucketName** value from the left side of the lab page.

For example *bucket_name=notes-bucket-123456789*

CAUTION: This value must be entered in all lowercase. If the combined items resolve to an Amazon S3 bucket name that is already in use, modify the suffix by adding random letters or numbers to the end of it. When you run your script to create a bucket, you will be notified if the name is valid.

7. Save the **config.ini** file.

Note: After you confirm that the file is saved, you can keep it open to view the settings that are used in other scripts as you work through the lab.

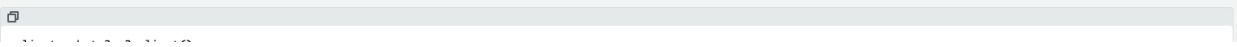
8. In the **Environment** window, expand the **labRepo** folder and open **create-bucket.py** .

As previously mentioned, all functions are initially defined in the scripts Near the end of the file, you will find the first TODO section after all of the functions are defined.

In the first TODO section, you need to choose and insert the correct code snippet to create an Amazon S3 service client.

9. **Copy/Paste:** Choose and insert the correct code snippet to complete TODO 1.

- Choice A



```
client = boto3.s3.client()
```

- Choice B

```
client = boto3.client('s3')
```

Answer: You can find the solution to this step in the TODO 1 solution section at the bottom of these instructions.

After your client is created, you invoke the main function and pass it the service client to use. If any unhandled client or parameter validation errors aren't handled within the main function, they will be revealed at this level and displayed in the output.

TASK 1.2: DEVELOP CODE TO VERIFY THAT YOUR PROPOSED BUCKET NAME IS VALID

The **main** function is used to orchestrate all script logic. Variables are read from the configuration file and passed to the supporting functions to help you visualize the flow of the script. In the next code challenge you will see how the script then creates a unique name for the bucket

In this TODO section, you choose and insert the correct logic to verify that the generated name is available for use when creating a new bucket.

CAUTION: Don't forget that Python uses indention to group blocks of code together. The indention of each code challenge should start at the **same level as the comments** that start and end the TODO sections.

10. **Copy/Paste:** Replace the code in **TODO 2** with the correct snippet below to complete the **VerifyBucketName** function.

- Choice A

```
s3Client.head_bucket(Bucket=bucket)
```

- Choice B

```
s3Client.list_buckets(Bucket=bucket)
```

Answer: You can find the solution to this step in the TODO 2 solution section at the bottom of these instructions.

TASK 1.3: DEVELOP CODE TO CREATE THE BUCKET IN YOUR LAB REGION

You now have a unique name for your new bucket. You can use that name to complete TODO 3 which finishes the createBucket function.

If your lab is running in any region except for **us-east-1** the **LocationConstraint** must be specified to host the bucket in that region. For **us-east-1** , you can't use the constraint. The script uses conditional logic to determine which set of parameters to use. You have to determine the best AWS SDK method to create a bucket.

11. **Copy/Paste:** Replace the code in **TODO 3** with the correct snippet below to complete the **createBucket** function.

- Choice A

```
if current_region == 'us-east-1':
        response = s3Client.create_bucket(Bucket=name)
    else:
        response = s3Client.create_bucket(
          Bucket=name,
          CreateBucketConfiguration={
              'LocationConstraint': current_region
          })
```

- Choice B

```
if current_region == 'us-east-1':
        response = s3Client.create(Bucket=name)
    else:
        response = s3Client.create(
          Bucket=name,
          CreateBucketConfiguration={
              'LocationConstraint': current_region
          })
```

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

TASK 1.4: DEVELOP CODE TO VERIFY THE BUCKET IS IN YOUR ACCOUNT, AND RUN YOUR SCRIPT

After creating the bucket, the script will need to verify that the bucket has been created in your account. For **TODO 4** , choose the correct option that will poll the Amazon S3 service until the bucket exists or the method times out.

12. **Copy/Paste:** Replace the code in **TODO 4** with the correct snippet below to complete the **verifyBucket** function.

- Choice A

```
s3Client.wait('bucket_exists', Bucket=bucket)
```

- Choice B

```
waiter = s3Client.get_waiter('bucket_exists')
    waiter.wait(Bucket=bucket)
```

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

Now that the **create-bucket.py** script is complete you can run it to create the bucket in your account.

13. Save and close the **create-bucket.py** file.

14. **Command:** In the AWS Cloud9 **terminal** run the command below to change directories to the **~/environment** directory:

```
cd ~/environment
```

Expected Output:

None, unless there is an error.

15. **Command:** Run the command below to run your completed script:

```
python labRepo/create-bucket.py
```

Expected output:

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

Start of create bucket script

Reading configuration file for bucket name...
Verifying that the bucket name is valid...
Existing Bucket Not Found, please proceed
notes-bucket-iii-00000

Creating notes-bucket-iii-00000 in us-west-2
Success!

Confirm that the bucket exists...
The bucket:notes-bucket-iii-00000 is now available.

End of create bucket script
```

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

Task 2: Upload an object to Amazon S3

In this task, you will complete the **create-object.py** script that uploads the **notes.csv** file from your local disk to the bucket you created in the previous task. This file contains sample notes that you will use in developing your application. You will add custom metadata before uploading the object to keep track of test files in your bucket.

The **labRepo/create-object.py** file is processed in the following order by the main function:

- Read the config.ini file and use the settings for all static values.
- **uploadObject** Uploads a file to Amazon S3.

High-Level Instructions

- Finish **labRepo\create-object.py** to add the **notes.csv** to your bucket.
- Complete **TODO 5** to create a notes.csv object.
- Run the script and confirm that there are no errors returned.

Detailed Instructions

TASK 2.1: DEVELOP CODE TO CREATE AN OBJECT IN AN AMAZON S3 BUCKET

16. Open the **labRepo/create-object.py** file.

In the **main()** function, the script reads variables from the configuration file. Review each variable to see what is being assigned. Notice that the configuration file has information about the file, its content, and information to be assigned as metadata.

The uploadObject function, is passed all of the variables that you gathered from the .ini file for use in uploading the file. The metadata parameter expects a dictionary object that you are building to be passed to it from the configuration.

You will use those parameters to create an object in your bucket in the single code challenge in this task.

17. **Copy/Paste:** Select the correct option to complete **TODO 5** and the **uploadObject** function.

- Choice A

```
response = s3Client.upload_file(
        Bucket=bucket,
        Key=key,
        Filename=name,
        ExtraArgs={
            'ContentType': contentType,
            'Metadata': metadata
            }
    )
```

- Choice B

```
response = s3Client.put_object(
        Bucket=bucket,
        Key=key,
        Filename=name,
        ContentType=contentType,
        Metadata=metadata
    )
```

Answer: You can find the solution to this step in the TODO 5 Solution section at the bottom of these instructions.

The **create-object.py** script is now complete, and you can run it to create the CSV object in your bucket.

18. Save and close the **create-object.py** file.

19. **Command:** Run the command below to change directories to the **~/environment** directory:

```
cd ~/environment
```

Expected Output:

None, unless there is an error.

20. **Command:** In the AWS Cloud9 terminal, run your completed script using the command below:

```
python labRepo/create-object.py
```

Expected output:

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

```
Start of create object script

Reading configuration file for bucket name...
Creating Object...
Finished creating object

End of create object script
```

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

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

One use of Amazon S3 is to aggregate data from different sources. Often, the source data output is not formatted for processing. Transformations, like the one performed in the **convert-csv-to-json.py** , are commonly used to address this issue.

In this task, you will complete the **convert-csv-to-json.py** script. This script retrieves the object data you uploaded in the last task. It then converts the data to a JSON string and creates a new object in the Amazon S3 Bucket.

Note: This JSON data will be used to load data into the database in a future lab.

The **labRepo/convert-csv-to-json.py** file is processed in the following order by the main function.

- Read the config.ini file and use the settings for all static values.
- **getCSVFile** downloads the contents of the notes.csv object as a string.
- **convertToJSON** converts the csv string to a JSON string.
- **createObject** creates a new Amazon S3 object with the converted JSON string.

High-Level Instructions

- Finish **labRepo\convert-csv-to-json.py** to process data in notes.csv.
- Complete **TODO 6** to download the contents of the csv file to memory.
- Complete **TODO 7** to upload the converted data to a new object.
- Run the script to convert the data and confirm that there are no errors.
- Download the JSON file to confirm the contents.

Detailed Instructions

TASK 3.1: DEVELOP CODE TO RETRIEVE THE OBJECT DATA FROM AMAZON S3

In the last task you uploaded an object to Amazon S3. In this task you will retrieve that object's contents for processing.

21. Open **convert-csv-to-json.py** in the AWS Cloud9 environment.

22. **Copy/Paste:** Choose and insert the correct choice to complete **TODO 6** and download the file contents to memory in the **getCSVFile** function.

- Choice A

```
bytes_buffer = s3Client.download_file(
      Bucket=bucket,
      Key=key)
```

- Choice B

```
s3Client.download_fileobj(
      Bucket=bucket,
      Key=key,
      Fileobj=bytes_buffer)
```

Answer: You can find the solution to this step in the TODO 6 Solution section at the bottom of these instructions.

TASK 3.2: DEVELOP CODE TO CREATE A NEW OBJECT WITH THE PROCESSED DATA

After the script downloads the data to memory, it will convert it from a comma delimited format to JSON. The conversion uses the column values on the first row as keys in the JSON file. This processed data also needs to be stored in Amazon S3.

In the next TODO, choose and insert the correct code snippet to create a new object with the processed data.

23. **Copy/Paste:** Choose and insert the correct option to complete **TODO 7** and create the new object in the **createObject** function.

- Choice A

```
s3Client.upload_file(
        Bucket=bucket,
        Key=key,
        Body=data,
        ExtraArgs={
            'ContentType': contentType,
            'Metadata': metadata
            }
    )
```

- Choice B

```
s3Client.put_object(
        Bucket=bucket,
        Key=key,
        Body=data,
        ContentType=contentType,
        Metadata=metadata
    )
```

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

TASK 3.3: RUN THE SCRIPT TO CREATE A NEW OBJECT IN JSON FORMAT

The **convert-csv-to-json.py** script is complete. You can run it to transform the CSV data to JSON.

24. Save and close the **convert-csv-to-json.py** file.

25. **Command:** Run the command below to change directories to the **~/environment** directory:

```
cd ~/environment
```

Expected Output:

None, unless there is an error.

26. **Command:** Run the command below to run your completed script:

```
python labRepo/convert-csv-to-json.py
```

Expected output:

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

Start of convert object script

Reading configuration file for bucket name...

Getting the CSV object from S3 bucket

Converting CSV string to JSON...
Creating the new JSON object on S3
Successfully Created Object

End of convert object script
```

Congratulations! You have successfully downloaded an object's contents, transformed it to the needed format, and created a new object with the output. The notes in this JSON file should match the notes in the variable from the previous task.

Task 4: Configure static website hosting on an Amazon S3 bucket with the AWS CLI

Many website configurations only need to be done once. In this case, it is more efficient to use the AWS Console or AWS CLI to configure them. In this task, you will use AWS CLI commands to configure your Amazon S3 bucket to host your application's website. This website will host the html files provided to you in the html folder of labRepo. You will set the bucket policy so that web pages can be accessed over the internet.

High-Level Instructions

- Upload the files in the **labRepo/html** folder to your Amazon S3 bucket.
- Enable Amazon S3 website hosting on your bucket using the index.html and error.html you uploaded.
- Apply the bucket policy provided in the **labRepo/policy.json** file to your bucket.
- View the website in your web browser to confirm that it displays as expected.

Detailed Instructions

27. **Command:** In the AWS Cloud9 **Terminal** window, use the **s3api service** to create a variable that contains your bucket name with the command below. Be sure to replace the *NOTES_BUCKET_NAME* with the **NotesBucketName** value from the left side of the lab page.

```
mybucket=NOTES_BUCKET_NAME
```

Expected Output:

None, unless there is an error.

CAUTION: You can run `echo $mybucket` in the terminal to confirm that it matches the bucket_name in the **config.ini** file. If it does not, manually set the variable.

28. Update the public-access-block permissions on the bucket.

```
aws s3api put-public-access-block --bucket $mybucket --public-access-block-configuration "BlockPublicPolicy=false,RestrictPublicBuckets=false"
```

29. **Command:** Run the command below to sync the files in the html folder with your bucket including the index.html and error.html files along with any assets used in those webpages:

```
aws s3 sync ~/environment/labRepo/html/. s3://$mybucket/
```

Similar expected output:

Note: Your **bucket name** will differ from the value shown here.

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

upload: html/error.html to s3://notes-bucket-iii-00000/error.html
upload: html/404.png to s3://notes-bucket-iii-00000/404.png
upload: html/index.html to s3://notes-bucket-iii-00000/index.html
upload: html/header.png to s3://notes-bucket-iii-00000/header.png
upload: html/styles.css to s3://notes-bucket-iii-00000/styles.css
```

30. **Command:** Run the command below to enable the Amazon S3 website hosting:

```
aws s3api put-bucket-website --bucket $mybucket --website-configuration file://~/environment/labRepo/website.json
```

Expected Output:

None, unless there is an error.

Note: The **website.json** file just specifies what objects to use for the index and error documents.

The bucket policy has already been created on your behalf. The policy has one statement that allows everyone to perform **s3:GetObject** on the specified resource. You will need to make one modification for the bucket name.

31. **Command:** Run the following command to update the current place holder for your S3 bucket name:

```
sed -i "s/\[BUCKET\]/$mybucket/g" ~/environment/labRepo/policy.json
```

Expected Output:

None, unless there is an error.

32. **Command:** Run the next command to verify the update is correct:

```
cat ~/environment/labRepo/policy.json
```

Similar expected output:

Note: Your **bucket name** will differ from the value shown here.

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

{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Principal": "*",
      "Action": ["s3:GetObject"],
      "Resource": "arn:aws:s3:::notes-bucket-iii-00000/*"
    }]
}
```

Note: If it is not updated correctly, manually update this file.

33. **Command:** Run the following commands to apply the bucket policy:

```
aws s3api put-bucket-policy --bucket $mybucket --policy file://~/environment/labRepo/policy.json
```

Expected Output:

None, unless there is an error.

34. **Command:** Run the following command to set the region value to a variable:

```
region=$(curl http://169.254.169.254/latest/meta-data/placement/region -s)
```

Expected Output:

None, unless there is an error.

35. 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 will contain **s3-website-** or **s3-website.** in the URL.

- **Choice A:**

```
printf "\nYou can now access the website at:\nhttp://$mybucket.s3-website-$region.amazonaws.com\n\n"
```

- **Choice B:**

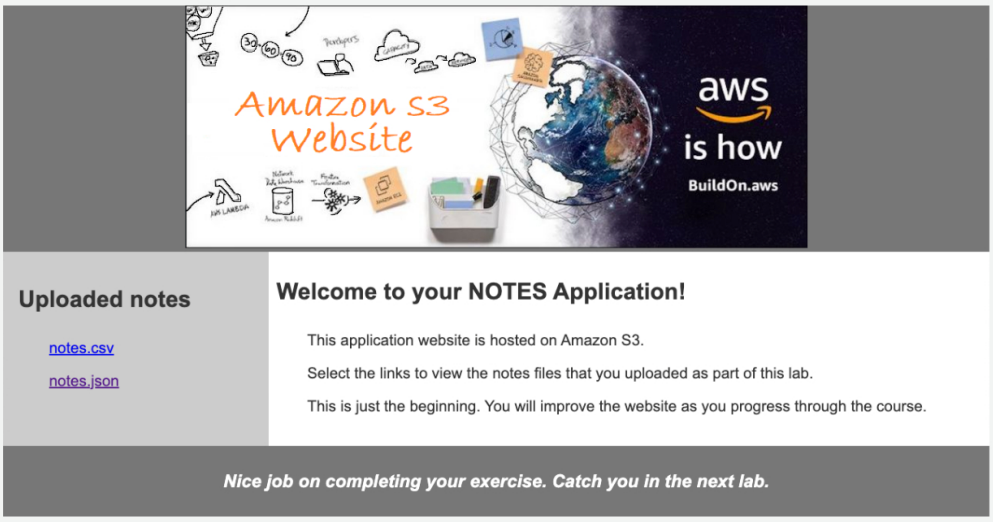
```
printf "\nYou can now access the website at:\nhttp://$mybucket.s3-website.$region.amazonaws.com\n\n"
```

Expected output:

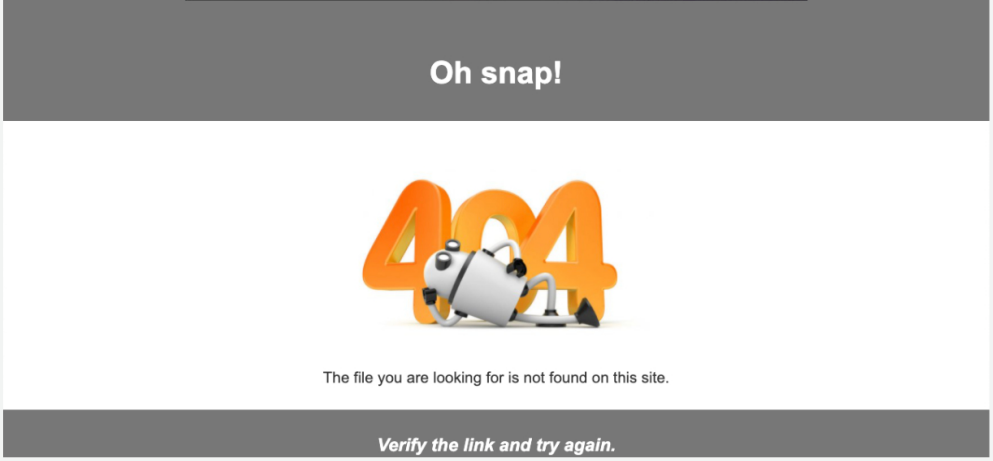
```
******************************
**** This is OUTPUT ONLY. ****
******************************

You can now access the website at:
http://notes-bucket-iii-00000.s3-website-us-west-2.amazonaws.com

Or...

You can now access the website at:
http://notes-bucket-iii-00000.s3-website.us-west-2.amazonaws.com
```

36. **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:



- Update the current URL with **/error.html** and you should see the following **error.html** page:



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 Python

Your website is configured and ready to go, but could you use what you've learned in this lab and the AWS SDK for Python (Boto3) documentation to complete the previous task with Python instead of the AWS CLI? Using the **labRepo/create-s3-website.py** file, complete 3 final TODO sections to test your skills. As always, there is a complete version of the file in the **Solutions** folder to reference if needed. The commands that you use are idempotent, allowing you to run the completed script to verify that it works.

The **labRepo/create-s3-website.py** file is processed in the following order by the main function:

- Read the config.ini file and use the settings for all static values.
- **uploadWebsiteFiles** uploads all of the files from the html folder to the bucket.
- **enableWebHosting** enables website hosting on the bucket.
- **allowAccessFromWeb** applies a bucket policy to the bucket that allows public access.

High-Level Instructions

- Finish **labRepo/create-s3-website.py** to process data in notes.csv.
- Complete **TODO 8** to upload the files to the bucket.
- Complete **TODO 9** to enable Amazon S3 web hosting.
- Complete **TODO 10** to apply the provided bucket policy to the website bucket .
- Run the script and confirm that there are no errors and that you can still access the website.

Expected output:

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

Starting create website function...

Reading configuration file for bucket name...
Uploading files for the website...
Enabling web hosting on the bucket...
Adding a bucket policy to allow traffic from the internet...

You can access the website at:

http://notes-bucket-iii-00000.s3-website.region.amazonaws.com

End create website function...
```

Summary

Congratulations on completing the lab! For the **Python** 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.

End lab

Follow these steps to close the console and end your lab.

37. Return to the **AWS Management Console** .

38. At the upper-right corner of the page, choose **AWSLabsUser** , and then choose **Sign out** .

39. Choose **End lab** and then confirm that you want to end your lab.

For more information about AWS Training and Certification, see *https://aws.amazon.com/training/* .

Your feedback is welcome and appreciated.
If you would like to share any feedback, suggestions, or corrections, please provide the details in our *AWS Training and Certification Contact Form* .

Additional Resources

- AWS SDK for Python (Boto3)

Code Challenge Solutions

TODO 1 SOLUTION

- Choice A is incorrect because the client is not specifying the **s3** parameter correctly.
- **Choice B is the correct code snippet** . You have to use the boto3 client class and pass it the name of the service client you wish to use.

```
client = boto3.client('s3')
```

Return to the instructions

TODO 2 SOLUTION

- **Choice A is the correct code snippet** . You would use head_bucket to get information about an existing bucket by name. The head_bucket method returns a **botocore.exceptions.ClientError** with the error code of 404 if the bucket name is not found in AWS.

```
s3Client.head_bucket(Bucket=bucket)
```

- Choice B is incorrect because list_buckets does not allow you to pass a *Bucket* property to limit it to one bucket. Additionally, the list_bucket method will only list buckets in your AWS account.

Return to the instructions

TODO 3 SOLUTION

- **Choice A is the correct code snippet.**

```
if current_region == 'us-east-1':
        response = s3Client.create_bucket(Bucket=name)
    else:
        response = s3Client.create_bucket(
          Bucket=name,
          CreateBucketConfiguration={
            'LocationConstraint': current_region
          })
```

- Choice B uses the incorrect syntax to create a bucket with the Amazon S3 client. If this code was using an Amazon S3 bucket resource, you could use the *bucket.create()* method. The bucket name would have to be specified before calling *create()* though, not passed with the call.

Return to the instructions

TODO 4 SOLUTION

- Choice A is incorrect because it does not specify a waiter.
- **Choice B** is correct. You have to create the waiter and then use it to poll Amazon S3 until the bucket is found.

```
waiter = s3Client.get_waiter('bucket_exists')
    waiter.wait(Bucket=bucket)
```

Return to the instructions

TODO 5 SOLUTION

- **Choice A is the correct code snippet.**

The parameters passed to upload_file are correct.

```
response = s3Client.upload_file(
        Bucket=bucket,
        Key=key,
        Filename=name,
        ExtraArgs={
            'ContentType': contentType,
            'Metadata': metadata
            }
    )
```

- Choice B is incorrect because the parameters passed to put_object would not be correct. Put_object requires the contents of the object passed in a data property, not the filename, to be transferred.

Return to the instructions

TODO 6 SOLUTION

- Choice A is incorrect because the output format and properties are not correct for download_file. Even if it was the correct syntax, download_file stores the data on your local disk instead of in-memory, which doesn't meet the requirements specified.
- **Choice B is the correct code snippet.** Download_fileobj will retrieve the contents to a file-like object, specified in the **Fileobj** parameter, in binary format that will be decoded. This is a managed transfer which will perform a multipart download in multiple threads if necessary.

```
s3Client.download_fileobj(
```

```
        Bucket=bucket,
        Key=key,
        Fileobj=bytes_buffer)
```

Return to the instructions

TODO 7 SOLUTION

- Choice A is incorrect because upload_file is used to upload a file from the local disk. For this solution there is no need to write the converted data to a disk when it can be uploaded directly.
- **Choice B is the correct code snippet.** Using put_object, is correct.

```
s3Client.put_object(
        Bucket=bucket,
        Key=key,
        Body=data,
        ContentType=contentType,
        Metadata=metadata
    )
```

Return to the instructions

Appendix

AWS SERVICES NOT USED IN THIS LAB

AWS services that are not used in this lab are deactivated in the lab environment. In addition, the capabilities of the services used in this lab are limited to what the lab requires. Expect errors when accessing other services or performing actions beyond those provided in this lab guide.

ICON KEY

Various icons are used throughout this lab to call attention to different types of instructions and notes. While not all of the icons will be used, the following list explains the purpose for each icon:

- **Command:** A command that you must run.
- **Expected output:** A sample output that you can use to verify the output of a command or edited file.
- **Note:** A note, tip, or important guidance.
- **Additional information:** Where to find more information.
- **Caution:** Information of special interest or importance (not so important to cause problems with the equipment or data if you miss it, but it could result in the need to repeat certain steps).
- **WARNING:** An action that is irreversible and could potentially impact the failure of a command or process (including warnings about configurations that cannot be changed after they are made).
- **Consider:** A moment to pause to consider how you might apply a concept in your own environment or to initiate a conversation about the topic at hand.
- **Copy/Paste:** A code block that displays the contents of a script or file you need to copy and paste that has been pre-created for you. When you need to copy only a certain part of a code block, there will be numbered **TODO** comments in the code.
- **Knowledge check:** An opportunity to check your knowledge and test what you have learned.
- **Security:** An opportunity to incorporate security best practices.
- **Refresh:** A time when you might need to refresh a web browser page or list to show new information.
- **Copy command:** A time when copying a command, script, or other text to a text editor (to edit specific variables within it) might be easier than editing directly in the command line or terminal.
- **Hint:** A hint to a question or challenge.
- **Answer:** An answer to a question or challenge.
- **Group effort:** A time when you must work together with another student to complete a task.

Return to the instructions