Lab 3.1: Working with Amazon S3

Lab overview and objectives

In this lab, you use Amazon Simple Storage Service (Amazon S3) to host a static website. You will also implement architectural best practices to protect and manage your data.

After completing this lab, you should be able to:

- Create a static website by using Amazon S3.
- Apply a bucket policy on an S3 bucket to configure customized data protection.
- Upload objects to an S3 bucket by using the AWS SDK for Python (Boto3).
- Configure the website that is hosted on Amazon S3 to be accessible only from a specific IP address range, and test the configuration.

Duration

This lab will require approximately **60 minutes** to complete.

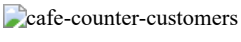
AWS service restrictions

In this lab environment, access to AWS services and service actions might be restricted to the ones that are needed to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond the ones that are described in this lab.

Scenario

Sofía is eager to start building a website for the café. She has some Python development skills and she's learning more about how to develop solutions on the cloud. Nikhil is a secondary school student who also works at the café. He has some skills in graphic design and a strong interest in learning about cloud computing.

The café has a single location in a large city. It mostly gains new customers when someone walks by, notices the café, and decides to try it. The café has a reputation for high-quality desserts and coffees, but their reputation is limited to people who have visited, or who have heard about them from their customers.


cafe-counter-customers

Sofía suggests to Frank and Martha (her parents and the owners of the café) that they should expand community awareness of what the café has to offer. The café doesn’t have a web presence, and it doesn’t currently use any cloud computing services. However, that situation is about to change.

In this lab, you will play the role of Sofía and take the first steps needed create a website for the café. It's time to get started!

When you *start* the lab, the following resources are already created for you in the AWS account:

At the *end* of this lab, your architecture should look like the following example:

cafe website architecture

Accessing the AWS Management Console

1. At the top of these instructions, choose Start Lab to launch your lab.

   A **Start Lab** panel opens, and it displays the lab status.

   Tip: If you need more time to complete the lab, choose the **Start Lab** button again to restart the timer for the environment.

2. Wait until you see the message *Lab status: ready*, then close the **Start Lab** panel by choosing the **X**.

3. At the top of these instructions, choose AWS.

   This opens the AWS Management Console in a new browser tab. The system will automatically log you in.

   Tip: If a new browser tab doesn't open, a banner or icon is usually at the top of your browser with a message that your browser is preventing the site from opening pop-up windows. Choose the banner or icon and then choose **Allow pop ups**.

4. Arrange the AWS Management Console tab so that it displays along side these instructions. Ideally, you will be able to see both browser tabs at the same time so that you can follow the lab steps more easily.

   Tip: If you would like the lab instructions to display across the entire browser window, you can hide the terminal in the browser panel by unchecking the Terminal checkbox in the top right.

A business request for the café: Create a static website

Sofía mentions to Nikhil that she would like to build a proof-of-concept website for Frank and Martha that will showcase the café's offerings visually. It would also provide business details, such as the location of the store, business hours, and telephone number. Nikhil is eager to see how she will build the website, and he agrees to create the graphics that she will use.

For this first challenge, you will take on the role of Sofia. You will use the AWS Command Line Interface (AWS CLI) and the SDK for Python to configure Amazon S3 so that it hosts a basic website for the café.

Task 1: Connecting to the AWS Cloud9 IDE and configuring the environment

In this first task, you will configure the AWS Cloud9 environment to support the development that you will work on during the rest of the lab.

5. Connect to the AWS Cloud9 IDE.
   - From the **Services** menu, search for and select **Cloud9**.

     You should see an existing IDE that's named **Cloud9 Instance**.
   - Choose **Open**.

     The AWS Cloud9 IDE loads in a new browser tab.

6. Install the AWS SDK for Python.
   - In the AWS Cloud9 bash terminal (located at the bottom of the IDE), run the following commands:

     ```
     sudo pip install boto3
     ```

     Note: You might see a note that you aren't using the latest version of pip, but you can ignore this message.

7. Download and extract the files that you will need for this lab.
   - In the same terminal, run the following command:

     ```
     xxxxxxxxxx
     wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCDEV-2-91558/02-lab-s3/code.zip -P /home/ec2-user/environment
     ```
   - The `code.zip` file is downloaded to the AWS Cloud9 instance. The file is listed in the left navigation pane.
   - Run the following commands to extract the file:

     ```
     xxxxxxxxxx
     unzip code.zip
     ```

Tip: You will use the files that you downloaded and extracted later in this lab.

8. Verify the AWS CLI version 2.

   - Verify the CLI version:

     ```
     xxxxxxxxxx
     aws --version
     ```

   - To confirm that AWS CLI version 2 is the installed version, the output should look similar to the following:

     ```
     xxxxxxxxxx
     aws-cli/2.15.9 Python/3.11.6 Linux/5.10.205-195.804.amzn2.x86_64 exe/x86_64.amzn.2 prompt/off
     ```

Task 2: Creating an S3 bucket by using the AWS CLI

In this task, you will create an S3 bucket to host your website. You will complete this task by using the AWS CLI.

9. In the AWS Cloud9 Bash terminal, run the following command, replacing <*bucket-name*> with your bucket name.

   For the *bucket name*, use the following items, separated by dashes (-):

   - Your initials in lowercase
   - Today's date in the format *YYYY-MM-DD*
   - The word *s3site*

   For example, *Sofia Martínez* might name the bucket *sm-2022-08-26-s3site*.

   ```
   xxxxxxxxxx
   aws s3api create-bucket --bucket <bucket-name> --region us-east-1
   ```

   The terminal should confirm that the bucket was created by returning output similar to this example:

   ```
   xxxxxxxxxx
   {
     "Location": "/sm-2022-08-26-s3site"
   }
   ```

   Tip: Copy the bucket name a text file in the AWS Cloud9 IDE or on your computer so that you can easily copy it during later steps in this lab. The bucket name does *not* include the leading slash (/) that was returned in the location name-value pair.

10. Use the AWS Management Console to confirm that the bucket was created, and observe the current permissions settings on the bucket.

    - In the AWS Management Console, choose **Services** and select **S3**. You should see that the bucket was created.

      The **Access** column for the bucket indicates that *Bucket and objects not public.*

    - Choose the bucket name and then choose **Permissions**.
    - Notice that **Block *all* public access** is turned on.
    - Choose **Edit**, de-select **Block *all* public access**.
    - Select **Block public access to buckets and objects granted through new access control lists (ACLs)**.
    - Select **Block public access to buckets and objects granted through any access control lists (ACLs)**.
    - Select **Block public and cross-account access to buckets and objects through any public bucket or access point policies**.
    - Choose **Save changes**.
    - In the confirm settings dialog, type in `confirm` and choose **Confirm**.

      Note: you will configure the bucket with a bucket policy later in this lab.

    - Scroll further down and observe that the bucket currently doesn't have a bucket policy attached to it.

Task 3: Setting a bucket policy on the bucket by using the SDK for Python

Now that Sofía has created the bucket, she wants to set permissions on the bucket. She wants Frank, Martha, and Nikhil to be able to access the first version of the website that she's building so that they can see the progress she's making. However, she doesn't want to expose the site to the outside world yet. She will accomplish this objective by configuring a *bucket policy* on the bucket. This bucket policy will enforce a condition.

11. Create the policy document.

    - In the **AWS Cloud9** IDE, in the navigation pane, choose the **Cloud9 Instance** directory.
    - Choose **File** > **New File** and then choose **File** > **Save**.
    - Name the empty file `website_security_policy.json` and choose **Save**.
    - In the `website_security_policy.json` file, paste the following code:

```
xxxxxxxxxx
{
       "Version": "2008-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": "*",
               "Action": "s3:GetObject",
               "Resource": [
                   "arn:aws:s3:::<bucket-name>/*",
                   "arn:aws:s3:::<bucket-name>"
               ],
               "Condition": {
                   "IpAddress": {
                       "aws:SourceIp": [
                           "<ip-address>/32"
                       ]
                   }
               }
           },
           {
               "Sid": "DenyOneObjectIfRequestNotSigned",
               "Effect": "Deny",
               "Principal": "*",
               "Action": "s3:GetObject",
               "Resource": "arn:aws:s3:::<bucket-name>/report.html",
               "Condition": {
                   "StringNotEquals": {
                       "s3:authtype": "REST-QUERY-STRING"
                   }
               }
           }
       ]
}
```

- In the policy document, replace all three `<bucket-name>` entries with your actual bucket name.
- Finally, replace `<ip-address>` with the IP address that is being used to connect your computer to the internet. You can find your IP address by visiting whatismyip.com

  Note: you won't be able to view the website if you use 0.0.0.0 as the allowed aws:SourceIp, because Amazon S3 is smart enough to recognize that means public access, and since you just set the bucket permissions to **Block *all* public access**, S3 will block your access. Therefore ensure you use your specific IPv4 address.

- **Save** the file.

  Analysis: This policy document consists of two statements. The first statement allows `GetObject` requests from your IP address. The second statement denies access to an object in the bucket that's named *report.html*, unless a specific condition is met. The `report.html` file doesn't exist in the bucket yet. However, in a later lab, you will upload the report.html file to the bucket and configure presigned URL access to it. The later lab will explain the details.

12. Apply the bucket policy to your bucket by using the SDK for Python.

    - In the navigation pane, expand the **python_3 directory** and open the **permissions.py** file (which contains starter code).
    - Edit the file by replacing <*bucket-name*> with the name of your bucket.
    - Save the changes.
    - Finally, in the terminal, navigate to the **python_3** directory and run the following code:

      ```
      xxxxxxxxxx

      cd python_3

      python3 permissions.py
      ```

      If the command completed successfully, you should see the message *DONE* in the terminal output.

      Optional: Return to the browser tab where the Amazon S3 console is open, and observe that the bucket policy has been applied to the bucket. (You might need to reload the console page.)

Task 4: Uploading objects to the bucket to create the website

Now that Sofía has configured permissions on the bucket, she needs to upload the website objects to the bucket. Fortunately, Sofía created the website code some time ago, and it's in the **resources** directory in the AWS Cloud9 IDE. Nikhil created the graphics for the website, and he's excited to see what the website will look like. Sofía is now ready to upload the files. To complete this task, she will use the AWS Cloud9 terminal to issue a simple recursive file upload command and disable caching (since the website is still being developed).

13. Run the code in the terminal. You should still be in the `python_3` directory.

    Be sure to replace `<bucket-name>` below with your actual bucket name

    ```
    xxxxxxxxxx

    aws s3 cp ../resources/website s3://<bucket-name>/ --recursive --cache-control "max-age=0"
    ```

Task 5: Testing access to the website

Now that Sofía has uploaded the website files to the S3 bucket, she must test the site and verify that it loads when a user accesses it through a virtual secure endpoint.

14. Load the website in a browser tab.

    - Return to the browser tab with the Amazon S3 console.
    - Choose your bucket name, and then choose **Objects**.

      If the files you just uploaded do not display, choose the refresh icon to view them.

    - Choose the **index.html** file.
    - Copy the **Object URL**. It will be in the following format. `https://<bucket-name>.s3.amazonaws.com/index.html`

      Note: In this lab scenario, the S3 bucket you created is intentionally *not* configured for static website hosting (a feature available in the bucket properties). Instead, you will access the website using the Object URL of the `index.html` file.

    - Verify that your website displays by pasting that full URI into your browser. Ensuring you are on the same network (as it will block anything other than the IPv4 you specified earlier)

15. Try to access the same URL from a location outside of your network.

- Ways that you can test outside access:
  - If you have mobile phone with a cellular network connection, try loading the same URL in a browser on the mobile phone.
  - Another way to test is by running the following command in the AWS Cloud9 Bash terminal (where <*bucket-name*> is the actual bucket name):

```
xxxxxxxxxx

curl https://<bucket-name>.s3.amazonaws.com/index.html
```

Analysis: Regardless whether you load the page from another location or use the curl command from the Cloud9 instance, an attempt is made to retrieve the page that you specified. It should return an *AccessDenied* error because your mobile device or the AWS Cloud9 instance (whichever one you chose to use) connects to the internet by using a different IP address than your computer.

The essential point is that you should *only* be able to access the website if the device uses the IPv4 address that's specified in the S3 bucket policy. In the café story, this IPv4 address is the café's network IP address.

16. Test that the website is loading.
    - Back in the browser tab where the website loads correctly, choose **Login** on the top right of the header.

      An alert will display saying `No API to call` This behavior is also expected at this point in the course.

      Analysis: In the next lab, you will begin to create an application programming interface (API) that the JavaScript code in this website will interact with. For now, however, that API doesn't exist yet.

Task 6: Analyzing the website code

Nikhil watched as Sofía configured the website. Now that the website is hosted and working, he asks Sofía to explain the code behind the site.

17. In the AWS Cloud9 IDE, browse to the **resources** > **website** directory and open the **index.html** file.
    - Notice that this HTML page references a number of CSS files in the head section
    - Towards the end of the **body** section, the code references a few JavaScript (.js) files. Some of these JavaScript files define the essential features of the site. One of these files is named `config.js` another one of them is named `pastries.js`.

18. Open the **config.js** file.
    - It contains an object literal that looks like this:

```
xxxxxxxxxx

window.COFFEE_CONFIG = {

API_GW_BASE_URL_STR: null,

COGNITO_LOGIN_BASE_URL_STR: null

};
```

As you continue through the labs in this course, at the appropriate time (for example, when you add Amazon API Gateway and Amazon Cognito to the website implementation) you will set values for these `keys` and overwrite the contents of this file on S3 with the changes. Then the behavior of the website will adapt to the enhancements accordingly.

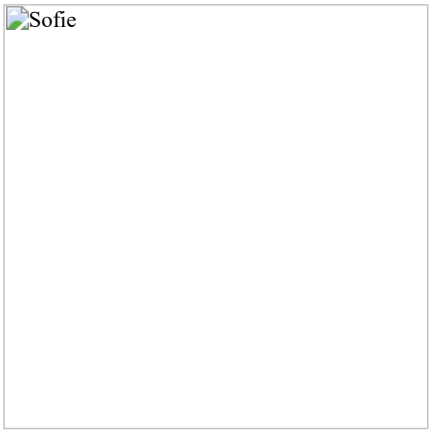
19. Open the **pastries.js** file.
    - Notice the logic defined in the *loadAllItems* function, that if the API_GW_BASE_URL_STR has a value of null (which it currently does, as you saw in config.js), then this function will print the items contained in the file named `all_products.json`.
    - Observe the *printItems* function details that are defined further down in the code.

20. Open the **all_products.json** file.
    - Notice that this JSON file contains information about all the products that display on the website.

      For now, this product information is stored in this file hosted on Amazon S3, however, in a later lab, you will migrate this product information into a database.

- *Optionally*, browse the other files that make up the website, to get a sense of how the website code works. The files have been left unobfuscated so you can read them if you wish. However you do *not* need to understand this application code in order to complete this course.

Update from the café



Sofia is pleased that she now has a basic website that's hosted on Amazon S3. Because it's still a work in progress, she made sure to configure the site so that it's only accessible from the café's network. Frank and Martha are both pleased with the results so far.

Sofia takes a minute to relax, but she's also already looking forward to the next task. She's planning to start developing the components that will make the site run dynamically by using REST API calls to a database backend.

Submitting your work

21. At the top of these instructions, choose Submit to record your progress and when prompted, choose **Yes**.

    Tip: If you previously hid the terminal in the browser panel, expose it again by selecting the **Terminal** checkbox. This action will ensure that the lab instructions remain visible after you choose **Submit**.

22. If the results don't display after a couple of minutes, return to the top of these instructions and choose Grades

    Tip: You can submit your work multiple times. After you change your work, choose **Submit** again. Your last submission is what will be recorded for this lab.

23. To find detailed feedback on your work, choose Details followed by **View Submission Report**.

Lab complete

Congratulations! You have completed the lab.

24. Choose End Lab at the top of this page, and then select Yes to confirm that you want to end the lab.

    A panel indicates that *DELETE has been initiated... You may close this message box now.*

25. Select the **X** in the top-right corner to close the panel.