Lab 2.1: Exploring AWS CloudShell and the AWS Cloud9 IDE

Lab overview and objectives

In this lab, you will take on the role of Sofía. You will connect to an AWS CloudShell environment and explore its capabilities. You will also launch an instance of AWS Cloud9, connect to it, and explore the layout and functionality of its integrated development environment (IDE).

After completing this lab, you should be able to do the following:

- **Connect to AWS CloudShell and run AWS Command Line Interface (AWS CLI) commands and AWS SDK code from it.**
- **Create an AWS Cloud9 development environment and connect to the browser-based IDE.**
- **Copy files to and from Amazon Simple Storage Service (Amazon S3), CloudShell, and AWS Cloud9.**
- **Install the AWS SDK for Python (Boto3) on an AWS Cloud9 instance.**
- **Use the AWS Cloud9 development environment to create files and run code files.**

Duration

This lab requires approximately 45 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

Frank and Martha are a married team who own and operate a small café business that sells desserts and coffee.

Their daughter, Sofía, works at the café. Sofía is pursuing a degree in cloud computing at a local university in the evenings and on the weekends. She has Python development skills and is learning more about how to develop solutions in the cloud.

Sofía is eager to start developing a web presence for the café. She thinks that before she starts coding, it would be a good idea to decide on a development environment for developing and running her code. She decides to explore at least two options that are available on AWS.

When you *start* the lab, the only pre-created resource in the AWS account is an empty S3 bucket.

However, by the *end* of this lab, you will have created an AWS Cloud9 instance and performed the actions that are shown below:

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 want the lab instructions to display across the entire browser window, you can hide the terminal in the browser panel by deselecting the Terminal check box on the top right corner of this page.

Task 1: Exploring AWS CloudShell

AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. In this first task, you will connect to CloudShell and explore its capabilities, as shown in this diagram.

5. **In the AWS Management Console, at the top of the screen, choose the AWS CloudShell icon.**

6. **A new browser tab opens with the AWS CloudShell interface.**

If a "*Welcome to AWS CloudShell*" pop-up window opens, choose Close.

It might take 1–2 minutes for the terminal to become available.

You should be able to access a terminal window with a prompt.

7. **Verify that the AWS CLI is installed.**
    - **At the CloudShell prompt, run the following command: aws --version**
    - **In the output after aws-cli, the version indicates that CloudShell is using AWS CLI version 2.x.x by default.**

8. **Test the ability to run an AWS CLI command.**
    - **Run the following simple AWS CLI command: aws s3 ls**
    - **A list of the S3 buckets that exist in the account is returned.**

An empty sample bucket was automatically created when you started the lab. The bucket name should appear in the result set.

9. **From the Actions menu, choose Tabs layout > Split into columns.**
    - **A second terminal window opens. This action demonstrates that you can have multiple terminal panels open at the same time.**

10. **Test the ability to run SDK for Python code.**
    - **Open the context (right-click) menu for the following link, and download the file to your computer:**

list-buckets.py

- **From the Actions menu, choose Files > Upload file, and then choose Select file.**
- **In the File Upload window, scroll to the list-buckets.py file that you downloaded, choose it, and then choose Open.**
- **Choose Upload.**
- **Close the *File upload successful* notification.**
- **In the terminal on the right side, run the following command: cat list-buckets.py**

The output shows the contents of the file that you uploaded:

```
import boto3
```

session = boto3.Session()

s3_client = session.client('s3')

b = s3_client.list_buckets()

for item in b['Buckets']:

print(item['Name'])

- **In the terminal on the right, run the SDK for Python code by issuing the following command: python3 list-buckets.py**

The name of the S3 bucket is returned.

- **Compare this output with the AWS CLI command output in the terminal on the left.**

You have now used two different programmatic ways to retrieve a list of the S3 buckets that exist in the AWS account.

11. **Copy a file from CloudShell to an S3 bucket.**
    - **Copy the name of the bucket that includes *-sample bucket-* in the name.**
    - **To copy the list-buckets.py file to the bucket, go to the terminal on the left and run the following command (replace <*bucket-name*> with your actual bucket name):**

aws s3 cp list-buckets.py s3://<bucket-name>

If the upload is successful, an output similar to the following example is returned:

upload: ./list-buckets.py to s3://<bucket-name>/list-buckets.py

12.
13. **When you use AWS CloudShell, you have persistent storage of 1 GB for each AWS Region at no additional cost.**
14. **The persistent storage is located in your home directory ($HOME) and is private to you. If you run the df -H /home command in a terminal, the amount of storage that's available in your CloudShell environment is returned.**
15. **Data in your home directory persists between sessions. If you must store more than 1 GB, you can access an S3 bucket from CloudShell.**

Update from the café

Sofía was impressed with how quickly she could run commands and code in AWS CloudShell. She can already envision how she can use it to run PowerShell and other scripts. However, to build a website, she wants to use a fully-featured integrated development environment (IDE) where she can visually write, edit, run, and debug her code. CloudShell provides the vi and vim terminal-based text editors, but it doesn't provide many of the additional features that Sofía is looking for.

Faythe, a friend of Sofía, is an experienced AWS developer and consultant. When Faythe visited the café this morning to buy pastries, Sofía mentioned her search for a development environment and how she explored CloudShell. Faythe was impressed that Sofía knew about CloudShell and agreed that it's a useful tool.

Meanwhile, Faythe suggested exploring the features of AWS Cloud9. "Based on your description of the features that you want to use, I think you will like it! You can get started quickly by going to the AWS Cloud9 service page and launching an instance."

Sofía is eager to explore AWS Cloud9 which is what you will do next!

Task 2: Creating an AWS Cloud9 instance

In this task, you will create an AWS Cloud9 instance.

12. **Return to the AWS Management Console browser window.**

13. **In the search bar at the top of the AWS Management Console, enter Cloud9 and choose it.**

14. **Choose Create environment.**

15. **In the Name field, enter MyCloud9.**

16. **In the New EC2 instance section, choose Additional Instance Types.**

17. **From the Additional instance types dropdown list, choose t2.medium.**

18. **In the Network settings section below Connection, choose Secure Shell (SSH).**

19. **Choose Create.**

A message indicates that the AWS Cloud9 environment is being created. Within 1–2 minutes, a message indicates that your AWS Cloud9 environment was successfully created.

20. **Choose the Open link to open your AWS Cloud9 IDE.**

Task 3: Exploring the AWS Cloud9 IDE

In this task, you will explore some features of the AWS Cloud9 IDE and learn how to use them. You will also interact with Amazon Storage Service (Amazon S3) from the AWS Cloud9 environment. Finally, you will author a Hello World webpage in AWS Cloud9 and host the webpage in Amazon S3.

21. **Observe the IDE user interface (UI).**
    - **The bottom of the screen is a Bash terminal. This terminal provides functionality that's similar to AWS CloudShell.**
    - **The left side of the screen is the navigation pane, which shows the file system.**

This AWS Cloud9 environment runs on an Amazon Elastic Compute Cloud (Amazon EC2) instance that's now running in your AWS account. Notice that the root directory is named *MyCloud9*, which is the name that you used for your AWS Cloud9 environment.

22. **Copy a file from Amazon S3 to your local storage in AWS Cloud9 by using the Bash terminal to run an AWS CLI command.**
    - **In the Bash terminal return the name of your bucket by running the following command: aws s3 ls**
    - **Next, download the list-buckets.py file from Amazon S3 to the local storage on AWS Cloud9 by running the following command.**

Replace <*bucket-name*> with your actual bucket name. Also be sure to include the period (.) at the end of the command, which indicates that the file should be downloaded to the current directory.

aws s3 cp s3://<bucket-name>/list-buckets.py .

The list-buckets.py file should now be listed in the navigation pane.

23. **Open a code file that uses the SDK for Python and run it.**
    - **Double-click the list-buckets.py file so that it opens in the file editor window. The code displays.**
    - **Choose View > Syntax and verify that Python is the chosen syntax. If it's not, choose Python now.**
    - **At the top of the window, choose the Run icon.**

The command does *not* run successfully. Instead, a new tab opens in the bottom panel and you see an error message: *ModuleNotFoundError: No module named 'boto3'*

Python version 3 is already pre-installed on your AWS Cloud9 instance. However, the SDK for Python libraries are not installed.

- **Return to the Bash terminal and install the SDK for Python libraries by running this command:**

sudo pip3 install boto3

- **Try running the Python code again.**

This time, it should succeed and display the S3 bucket name.

You can also invoke the same code from the Bash terminal directly by running the python3 list-buckets.py command. The Run button is a convenience.

24. **Create a new file and upload it to Amazon S3 by using the AWS Explorer in AWS Cloud9.**
    - **Choose File > New From Template > HTML File.**
    - **Between the <body></body> tags, add the following text: Hello World.**
    - **Choose File > Save and save the file as index.html.**
    - **On the left side of the AWS Cloud9 editor, choose the AWS icon.**
    - **Choose Add regions to AWS Explorer..., in the search box enter us-east-1, and then press Enter.**
    - **Expand the Region—for example, *US East (N. Virginia)*—expand S3, and then expand your bucket.**

The contents of the bucket are listed.

- **Open the context (right-click) menu for the bucket name and choose Upload Files.**
- **In the dialog box that appears, choose index.html, and then choose Upload.**

The file is uploaded from the AWS Cloud9 instance to the S3 bucket.

You can also use this AWS Explorer interface to download files from Amazon S3 to AWS Cloud9.

Update from the café

Sofía is pleased that she identified an IDE that has the features that she needs to develop the café website. She likes that AWS Cloud9 offers a graphical text editor, a file browser, a terminal for running AWS CLI commands, and code that uses the AWS SDKs. She's also glad that she knows about the features of CloudShell because she can open it from the AWS Management Console. CloudShell also provides some features that are similar to AWS Cloud9 but without the need to run an EC2 instance.

In the next lab, Sofia will use AWS Cloud9 to accomplish her development objectives.

Submitting your work

25. **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 checking the Terminal checkbox in the top right. This will ensure that the lab instructions remain visible after you choose Submit.

26. **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.

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

Lab complete

Congratulations! You have completed the lab.

28. **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.*

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

Appendix

s3-permissions.py:

```
import boto3

BUCKET_NAME = 'YOUR_BUCKET_NAME'
FILE_NAME = 'index.html'

# setup s3 client named s3_client
s3_client = boto3.client('s3')

# create a function to put s3 bucket ownership controls with Rules set to BucketOwnerPreferred
def put_bucket_ownership_controls():
   response = s3_client.put_bucket_ownership_controls(
      Bucket=BUCKET_NAME,
      OwnershipControls={
         'Rules': [
            {
               'ObjectOwnership': 'BucketOwnerPreferred'
            },
         ]
      }
```

```
    )
    return response

# create a function to set public access block values to false
def set_public_access_block():

    response = s3_client.put_public_access_block(
        Bucket=BUCKET_NAME,
        PublicAccessBlockConfiguration={
            'BlockPublicAcls': False,
            'IgnorePublicAcls': False,
            'BlockPublicPolicy': False,
            'RestrictPublicBuckets': False
        }
    )
    return response

# create a function to allow public access to FILE_NAME
def allow_public_access_to_file():
    response = s3_client.put_object_acl(
        Bucket=BUCKET_NAME,
        Key=FILE_NAME,
        ACL='public-read'
    )
    return response

# call the functions
put_bucket_ownership_controls()
set_public_access_block()
allow_public_access_to_file()
```