Explore Cloud9 Environment and Storage (S3) as part of a small project

We will undertake a small project that will span in sessions. We will build a simple web application where you can upload photos that will be stored in an Amazon S3 bucket (session-1). We will then process the photos with Amazon Rekognition to generate labels (session-2) . At the beginning, we will use Amazon Cloud9 as an IDE for writing, running and debugging the code.

Objective: (Session-1)

- To Explore Cloud9, a cloud-based IDE to write, run, and debug code within a browser.
- Build the Amazon S3 uploader component of the application

General Instructions:

- Each lab is to be done individually. However you can discuss with others, but effort should finally be yours.
- Make sure to sign in to your AWS account with the AWS IAM user **credentials and not root.**

References:

- https://docs.aws.amazon.com/cloud9/latest/user-guide/tutorial.html
- https://docs.aws.amazon.com/AmazonS3/latest/userguide/GetStartedWithS3.html
- https://soshace.com/uploading-files-to-amazon-s3-with-flask-form-part1-uploading-small-files/

Lab Instructions:

1. Explore AWS Cloud9 environment

AWS Cloud9 is a cloud-based integrated development environment (IDE). You can access it via a browser. You can write, run, and debug your code within. It supports popular programming languages, including JavaScript, Python, PHP etc. It also comes with a terminal that includes sudo privileges to the EC2 instance that is hosting the development environment and a preauthenticated AWS Command Line Interface.

- In the AWS Console, Click **All Services** → **Developer Tools** → **Cloud9**.
- Click **Create environment** at the top-right corner.
- For **Name**, type **Photo-Upload-App** (or any name you like)
- In **New EC2 instance** section choose “4 hours” as timeout value
- leave everything as is except under **Network settings** in **VPC Settings**, For **Amazon Virtual Private Cloud (VPC)** choose the **aws-vpc** (or whatever you created in earlier exercise) . For **Subnet**, choose a **public subnet** (any one) in aws-vpc
- Review the details and then click **Create**. This should launch your AWS Cloud9 environment in a few minutes.Click On **Open**
- There is a terminal window on the bottom pane. The terminal provides a remote login to the instance on which the AWS Cloud9 environment is hosted.
    - **Explore the terminal** by typing this command:
      `aws ec2 describe-instances`
      This should give a JSON output with all the information of the EC2 instances in your account.
    - Install **Boto 3** on your AWS Cloud9 instance by typing this command:

- `sudo pip3 install boto3`
- At the terminal, type **python3** and press ENTER. Explore the Python Boto 3 APIs by executing these commands:
  - ```
    import boto3
    client = boto3.client('ec2')
    client.describe_instances()
    ```
- Press **CTRL+D** to exit the Python interpreter.

2. Create an Amazon S3 bucket to store the application photos.

Before we code the web application, we will need a handle to an S3 bucket to store photos. In this section, you will create an Amazon S3 bucket.

- In the AWS Console, click **All Services → Storage → S3**
- Click **Create bucket** on the right middle.
- For **Bucket name**, type a unique bucket name (e.g. photo-gallery-100).
- Leave the rest as default. Click **Create**.

3. Explore, run and launch the web application

You do not have to write any code for this exercise, we will use some existing code for this exercise. However, it is good if you go through the code and understand it at least at some high level.
    ```

- In the AWS Cloud9 terminal, ensure you are in the ~/environment directory else, do "`cd ~/environment`"
- `You should now see the folder` **FlaskApp** in your labDirectory. You need to create same directory structure as FlaskApp in your cloud9 terminal and also copy the file contents of FlaskApp directory(in labDirectory) and paste them into the files present in cloud9
- Here is some high-level explanation.
    - The **FlaskApp** folder contains all the relevant files
    - The requirements.txt contains all the libraries that are needed for this project. As you will see below, we will use pip to install these.
    - The static and templates folders contain the style/javascript/html files to render the webpage
    - The app.py is the main file. Some configuration details and some utilities (e.g resize image,allowed extensions) are in the util.py which is imported into the app.py file.
    - Amazon S3 client has been created to interact with Amazon S3 via the Boto 3 API.
- First navigate to the **FlaskApp** folder `and then` Install the requirements for the project by executing the command below in your AWS Cloud9 terminal.
    - **`cd FlaskApp`**
    - **`sudo pip3 install -r requirement.txt`**
- To run the app.py code:
    - Select **FlaskApp/app.py** in the tree view.
    - Next we need to set the environment variables. To set it do the following:

- Run these commands one by one in another Terminal inside **~/environment** directory:
- **echo "export BUCKET_NAME=<Your Bucket Name>" >> ~/.bashrc**
- **source ~/.bashrc**
- **echo "export SECRET_KEY=<Your Secret Key>" >> ~/.bashrc**
- **source ~/.bashrc**
- **(**The Flask app uses a secret value to encrypt session variables. So, some random characters needed for encryption.)
- On the top menu bar, click **Run.** In case you see this error **“**ImportError: cannot import name 'PROTOCOL_TLS' from 'urllib3.util.ssl_' “, reinstall boto3 again (`pip3 install boto3`)
- If you run the app.py, it should run successfully. In the terminal, you will see a message like this:

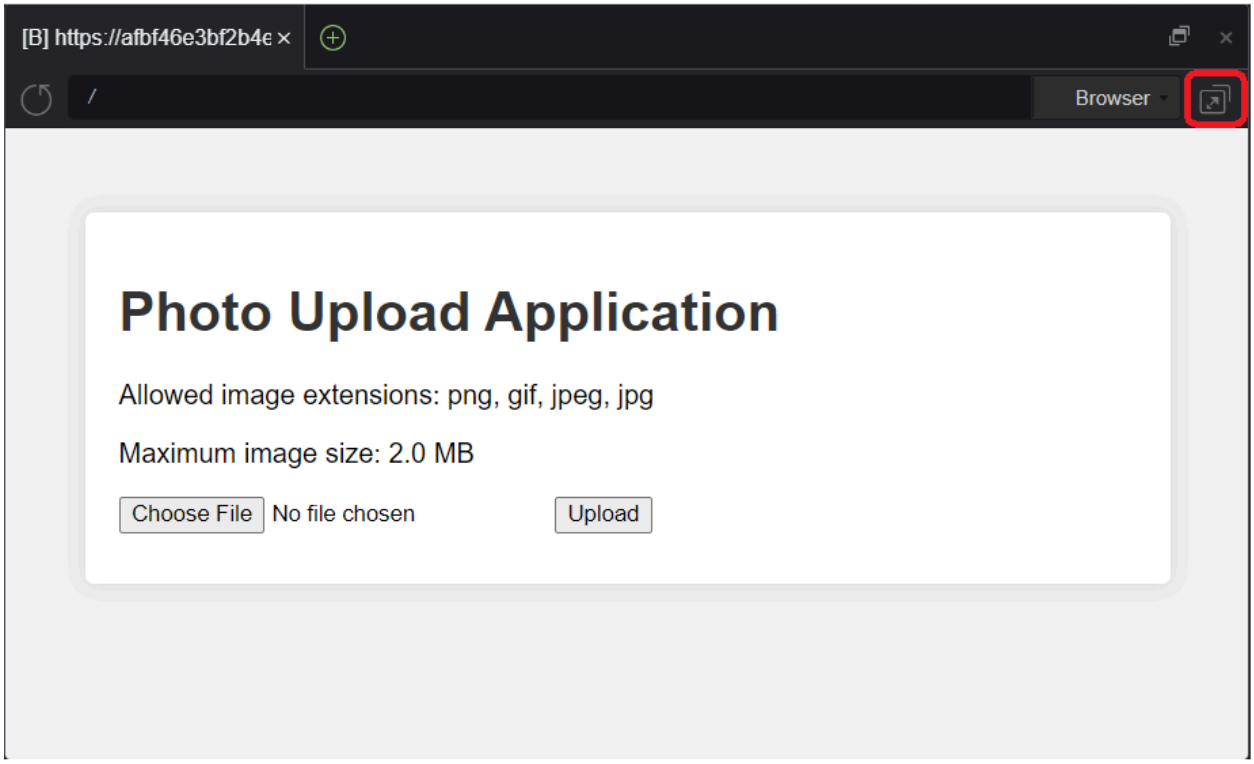
  `Running on http://0.0.0.0:8080/`

  That means the Python Flask app is now running on port 8080 of the AWS Cloud9 instance.
- If for some reason port 8080 is not working then you can try out different port numbers by just updating the below line in app.py

  app.run(host='0.0.0.0', port=<put your port number here>, debug=True)

- You can now test the application. For this, in the very top row (where File, Edit etc are there), click **Preview -> Preview Running Application**

- You should see the application running in a small window in the AWS Cloud9 environment.
- Pop out the application in a new window by clicking the **Pop Out** (marked in red) button shown in the screenshot below.



- The application is now running in the EC2 instance that supports the cloud9 environment.
- Try uploading a few photos via the app. You will notice that the photos are stored in the Amazon S3 bucket that you created earlier.

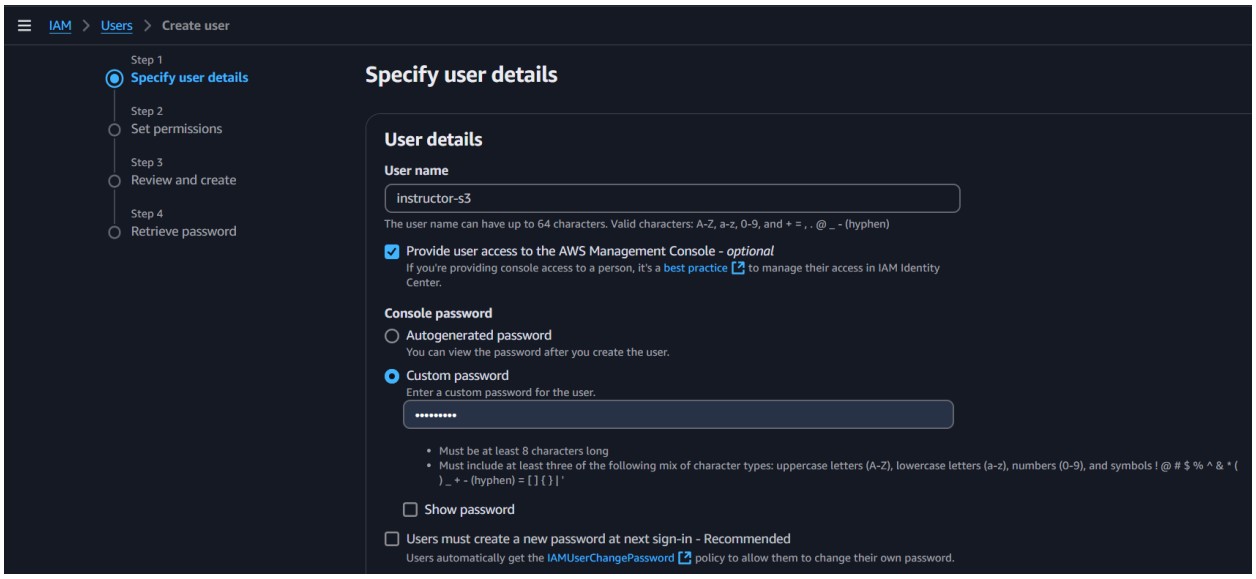
- To make the application public

1. First go to the **EC2 Dashboard**

2. Inside **Resources** section click on **Security groups**
3. Select the Security group corresponding to the AWS Cloud9 instance and inside the **inbound rules** tab click on **Edit inbound rules.**
4. Our application is running on port **8080**(or any other port of your choice), so we need to add an inbound rule which will allow all http requests on port 8080 from the external world.
5. Click on **Add Rule**, For Type, select **HTTP**, For Port Range, type 8080(or any other port). For **Source** keep custom and For **Source type** 0.0.0.0/0 or select “Anywhere”.
6. In order to access your application publicly, copy the Public IP of the instance and go to any other tab in your browser and type **http://<public IP>:<port>**
   After clicking **enter** you will be able to access your application publicly

4. Create an IAM user for the instructor to grade your application

For the purpose of checking your application’s correctness you need to create an AWS IAM user(as you did in AWS IAM activity) for the instructor. Make sure to check and uncheck necessary fields as shown in the below image while creating the user for easier workflow.



Attach the policy **instructor_iam_policy** which is present inside labDirectory.Go to the script and replace "**<s3 bucket name>**" with your Bucket name. You will then generate the access keys for the user. Those access keys will be used by the instructor to make programmatic calls to AWS services via AWS CLI or APIs.

5. Submission Instructions

You need to fill the following details:

Public IP and PORT of the application,Access key ID and SECRET_ACCESS_KEY for the Instructor IAM user account and your S3 Bucket name in the **data.json** file which is present in the current directory which looks like

```
{
    "public_ip": "PUBLIC_IP_OF_THE_APPLICATION",
    "port": "PORT_NUMBER",
    "INSTRUCTOR Access key ID" : "ACCESS_KEY_ID",
```

"INSTRUCTOR Secret Acess Key" : "SECRET_ACCESS_KEY",

"s3 bucket name" : "YOUR_BUCKET",

“Region” : “Your Bucket Region”

}

6. Delete the IAM user for instructor

- Delete the IAM user that you created for the instructor after evaluation is done.

7. Errors you may face during the lab and how to resolve them

- **Boto3 had to be reinstalled to fix :** ImportError: cannot import name 'PROTOCOL_TLS' from 'urllib3.util.ssl_' (/usr/local/lib/python3.7/site-packages/urllib3/util/ssl_.py)
- Port 8080+ (to avoid used port issues)
- If you encounter the error: Error uploading file: Failed to upload An error occurred (ExpiredToken) when calling the PutObject operation: The provided token has expired.
  simply logout and login again in the aws console and restart the flask application in cloud 9