module_2.md

Module 2: API-based solution

Resources

- AWS Solutions Library: AI-Powered Health Data Masking
- Deployment Guide: HTML, PDF
- Source code: GitHub: awslabs/ai-powered-health-data-masking

Process

Step 1: Launch CloudFormation Stack

- Ensure you have launched the CloudFormation stack as described in Module 1, Step 1.
- From the CloudFormation console, open the **Outputs** tab of your stack and note the value of **ApiGatewayId**

Step 2: Create an IAM Policy

- Reference: Implementation Guide, Step 2
- Open the AWS IAM console
- Select **Policies** then **Create Policy**
- Select the **JSON** tab and replace the policy contents with the text copied from below
  - Replace `ACCOUNTID` with your 12-digit account ID
  - Replace `APIGATEWAYID` with the API Gateway ID from above
  - Optionally, replace **us-east-1** with your current region, and prod with your staging environment name, if modified

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke",
        "apigateway:PUT",
        "apigateway:POST",
        "apigateway:GET"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:ACCOUNTID:APIGATEWAYID/prod/*",
        "arn:aws:apigateway:us-east-1::/restapis/APIGATEWAYID/resources/*"
      ]
    }
  ]
}
```

- Click **Review Policy**
- Give your policy a memorable name then click **Create policy**

Step 3: Attach the policy to the SageMaker role

- In the IAM console, select **Roles** and locate the customer-managed role for SageMaker created in Module 1, Step 3 (it will start with **AmazonSageMaker-ExecutionRole-**)
- Select **Attach policies** and select the policy just created in Step 2, above
- Click **Attach policy** and verify that the role now has 5 policies attached

Step 4: Test the API

- Reference: Implementation Guide, Appendix B
- Return to the SageMaker instance from Solution 1
- Select **New** → **Terminal**
- Run `bash` and `cd` to the `SageMaker` directory
- Create a new Python script in this directory in one of two ways:
  - Use **vi** or **emacs** to create the file in the terminal window
  - From the **Jupyter** page, Select **new** → **Text File**

Text Masking

- Use the code below, entering the values from the CloudFormation outputs. This script is also in the course resources in `python/api_textmask.py`
  - `api_id` : **ApiGatewayId**
  - `resource_id` : **TextMaskResourceId**

```
import boto3
import json

# Calls POST on /text/mask and invokes Lambda function `mask_text/lambda_function.py`

client = boto3.client('apigateway')

api_id = 'YOUR_API_ID'
resource_id = 'YOUR_TEXTMASKRESOURCEID'

payload = {
    "text": "PERSON INFORMATION\nName: SALAZAR, CARLOS\nMRN: RQ36114734\nED Arrival Time: 11/12/2011 18:15\nSex: Male\nDOB: 2/11/1961",
    "phiDetectionThreshold": 0.9
    }
response = client.test_invoke_method(
    restApiId=api_id,
    resourceId=resource_id,
    httpMethod='POST',
    headers={"Content-Type": "application/json"},
    body=json.dumps(payload))

print(response['body'])
```

- Save the file with a `.py` extension
- Run the code: `python <yourfile.py>`
- Notes:
  - This calls the **POST** method on `/text/mask` resource, which invokes the Lambda method in `/mask_text/`
  - Lambda invocations are logged in CloudWatch Logs **/aws/lambda/<stack-name>-<lambda-name>**
  - Lambda logging can be configured with the environment variable **LOG_LEVEL** in the Lambda console (INFO, ERROR etc)
  - API Gateway logging may be configured in **Stages** -> **Logs/Tracing**

Image Masking

- Use the code below, entering the values from the CloudFormation outputs. This script is also in the course resources in `python/api_imagemask.py`
    - `api_id` : **ApiGatewayId**
    - `resource_id` : **ImageMaskResourceId**
    - `s3_bucket` : Bucket containing your image
    - `s3_key` : Name of image file

```
import boto3
import json

# Calls POST on /image/mask

client = boto3.client('apigateway')

api_id = 'YOUR_API_ID'
resource_id = 'YOUR_IMAGEMASKRESOURCEID'
s3_bucket = 'YOUR_S3_IMAGE_BUCKET'
s3_key = 'YOUR_S3_IMAGE_KEY'

destination_key = 'masked/' + s3_key
payload = {
    "phiDetectionThreshold": 0.5,
    "s3Bucket": s3_bucket,
    "s3Key": s3_key,
    "destinationBucket": s3_bucket,
    "destinationKey": destination_key
    }
response = client.test_invoke_method(
    restApiId=api_id,
    resourceId=resource_id,
    httpMethod='POST',
    headers={"Content-Type": "application/json"},
    body=json.dumps(payload)
)

print(response['body'])
```