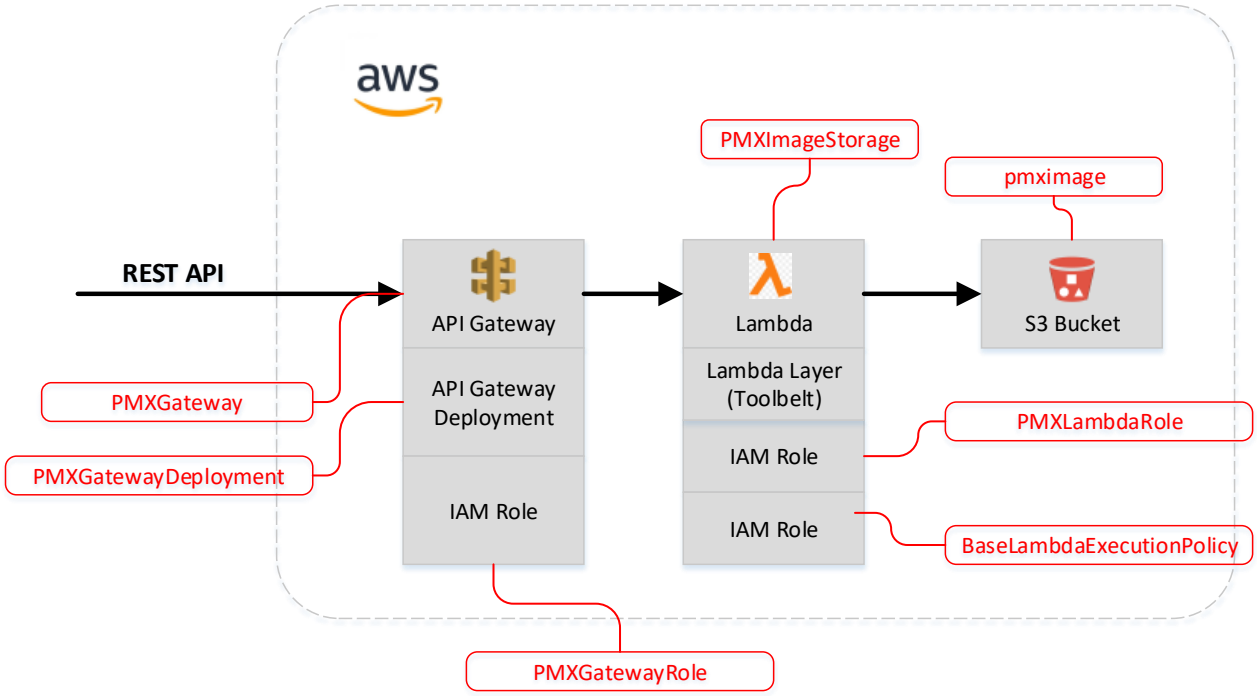
Image Gateway

Building Image Gateway using serverless architecture and hosting it via AWS

This **Image Gateway** has been built to provide the following functionality

a. Upload Images (DCM and NIFTY) for a Subject ID
b. Query for images by Subject ID
c. Download Images for a Subject ID



AWS Lambda service using Python 3.x tech stack has been used to implement this and **AWS API Gateway** service exposes the aforementioned functionality as REST API

The **Upload Image API** is a POST method, that accepts Subject ID and Image ID as QueryParameters . The image data is sent as a base64 encoded byte stream in the body of the request and the image type will be present in the Request Header Content-Type. I used Multipart Decoder from the requests-toolbelt python package to extract the image content from the request body. The extracted content is then written to s3 bucket. The object name is a combination of Subject ID (which is used as a prefix) and ImageID. I have used boto3 package

to perform the s3 operations. If the image is successfully stored, a 200 Ok response is sent

Query Image API is a GET method that lists all images stored for each of the Subject ID. This API accepts Subject ID as QueryParameter and provides list of image objects stored for the Subject ID. If Subject ID is invalid, then an empty list is returned

Download Image API is also a GET method that allows download of one image at a time. THis API accepts a valid Subject ID and Image ID and gets the image object from s3 bucket and sends it back in the response. The response is a json object shown as below

```
{
   "data":<base64 encoded image bytes>
}
```

If the Subject ID or Image ID is invalid, then the following json response is returned

```
{
   "message": "Internal server error"
}
```

The s3 bucket **'pmsimagestore'** is setup to be a private one with restricted access. Only the Lambda function is provided right IAM role to put and get objects from this bucket

Cloud formation script automates the steps to provision these services and assign the right roles and permissions.

This github repo contains all the required files to implement this solution

1. PMXImageStorage.py – Python code to upload, list and download images
2. Python.zip – Contains request-toolbelt python package. It should be uploaded as python.zip (and not by any other name) as a dependency in the lambda function via the console
3. PMX Image Gateway.postman_collection.json – This is a postman collection that contains all 3 API sample request. Replace the server url with the one that you have provisioned
4. pmx-template.yml – Cloud formation script to automate the service creation

This back-end service is being used to upload and download image files for the purpose of the android app being developed currently.