PROJECT FOR AWS

1-Created 2 servers Linux with apache web server with load balance and auto scaling.

The load balance will check the health of the application on Ec2 and if this unhealthy

It will automatically destroy it and created another one. Use cloud formation to deploy

All services.

2-Upload a file in S3 and it triggers a lambda function call to resize the picture and store the result in another S3 bucket.

3-Load a excel file into an S3 bucket, trigger GLUE to store the data in the excel into dynamoDB.

|  |
| --- |
| import boto3 |
|  | import os |
|  | import os.path |
|  | from pathlib import Path |
|  | import sys |
|  | from PIL import Image |
|  | import PIL.Image |
|  |  |
|  | # Create an S3 with boto3 |
|  | s3 = boto3.client('s3') |
|  |  |
|  | # Define a function to resize images; |
|  | def resize\_image(image\_path, resized\_path): |
|  | with Image.open(image\_path) as image: |
|  | image.thumbnail((200, 200)) |
|  | image.save(resized\_path) |
|  |  |
|  | def lambda\_handler(event, context): |
|  |  |
|  | # Read object key from event |
|  | key = event["Records"][0]['s3']['object']['key'] |
|  | object\_key = key.replace("+", " ") #correct potential issues with key name |
|  |  |
|  | # Construct download path (where the file uploaded from bucket will be saved) |
|  | down\_path = '/tmp/' + os.path.basename(object\_key) |
|  |  |
|  | # Construct upload path (where resized image will be saved locally) |
|  | path = os.path.join('/tmp', 'resized') |
|  | if os.path.exists(path) == False: #test the existence of the directory before creation |
|  | os.mkdir(path) |
|  | up\_path = path + os.path.basename(object\_key) |
|  |  |
|  | # Extract the object from origin bucket in S3 |
|  | s3.download\_file('originbucket-jak', object\_key, down\_path) |
|  |  |
|  | # call the resizing function |
|  | resize\_image(down\_path, up\_path) |
|  |  |
|  | # Upload the resized image to the destination bucket |
|  | s3.upload\_file(up\_path, 'destbucket-jak', object\_key) |