JEAN CARLOS DA CRUZ

AWS Machine Learning Engineer Nanodegree Program

Operationalizing Machine Learning on SageMaker Project

WRITEUP

1- Instance type

I chose the ml.t2.medium instance for my work because it has 2 vCPUS and 4GB of memory available. That setup is more than enough for the work I needed to accomplish.

2- EC2 Instance

I chose the t2.medium instance for my EC2 work because it is comparable to the sagemaker instance I used and found it is more than enough to accomplish the EC2 part as well.

3- Lambda Function

In order to accomplish the lambda part of the project I first defined the endpoint with the same name as I deployed using SageMaker in the *lamdafunction.py* since the lambda function receives a JSON with an URL of an IMAGE then starts a SageMaker runtime object to start the endpoint. The response is stored as the response body of the lambda function's output.

4- AWS Workspace Security

I decided to attach the AmazonSageMakerFullAccess policy to the Lambda's function roles. That is a role that grants full access to any operation in Sagemaker. In my opinion that is too permissive since I just wanted to invoke the endpoint, but unfortunately I did not find a less permissive role to it.

5- Concurrency and Autoscaling

I decided to set the reserved concurrency to 10 so the Lambda function can process 10 requests at the same time, for this training reason it is more than enough. I also configured autoscaling for the project's endpoint. I set the maximum instance to 2 with the threshold of 5 invocations per instance and if there is a case with more than 5 invocations, another instance will be spinned up. I set the in and out duration to 30 seconds to avoid any downtime.