The project aimed to train and deploy an image classification model on AWS the following are steps and decisions I made during the process and are as follows:

The goal is to minimize cost as much as possible so to start with, to create a notebook instance I chose ml.m5.xlarge because it is the cheapest and a bit faster compared to other cheapest instances type available cheap instances type on AWS.

Waited for some time for the notebook instance to be in service, launched the notebook instance by open Jupyter, and selected conda_python3 to open the environment that allowed allow me to train and deploy the model.
I downloaded the dataset and unzip it and then copied it to S3. The dataset uploaded to my bucket contains train, test, and validation files.

After that, I started the training process with 1 instance and it took 1690 seconds to complete the training job and billed 1690 seconds.

I then repeated the same process on multiple instance training and chose 5 instances and it trained a bit faster compared to training on 1 instance.

The training job was saved in the bucket I created under S3. The next step I did was to deploy the model and then later set up a lambda function for the deployed endpoint.

Meanwhile, I have set up auto-scaling for the deployed endpoint and since we are not expecting much traffic, I have set the auto-scaling as follows: scale down to 120 seconds, scale-out cool down to 30 seconds, and the target value of 20. Doing this to save resources in the AWS account.

Proceeding to the lambda function, testing the lambda function with default policy will result in an error however when added when I added the necessary policies in order for the lambda function to execute its job or access the deployed endpoint. And in this regard, I have added SageMakerFullAccess Policy to enable the Lambda function to access all the resources in the AWS

including the endpoint, and run the test, the Lambda function was giving status code 200 meaning the that Lambda function is able to communicate with the deployed endpoint.

Again, in this project, we do not expect high traffic and for this reason, I have decided to configure reserved concurrency only, choosing this because this e deployed endpoint model and again we are not receiving high traffic for this project thus I chose reserved concurrency of 4 because this is the cheapest compared to provisioned concurrency.

Started the invocation but could not work until I have added the necessary security policies to enable the Lambda function to access the endpoint, in this case, I added the SageMakeFullAccess policy.

On the other hand, I trained the same model on EC2 and since this project is for exploration and learning purposes, I have opted for spot instance. Spot instances are way cheaper compared to standard instances and overall using this type of instance is a great way to lower costs in the AWS. Training on EC2 will result in saving the training model in a created folder instead of s3 and the terminal command is wide to accomplish the task.

After training and deploying the model, I stopped the notebook instance and deleted deployed endpoint as well as the lambda function to avoid further charges.

Have used several AWS tools to adjust, improve and configure and prepare the model for production-grade deployment. The skills demonstrated in the project include optimizing advance,ed Machine Learning pipelines for efficiency, speed and security.