# AWS Machine Learning Project Writeup

## Step 1: Training and Deployment on SageMaker

I created an S3 bucket named 'udacity-p4-ml' and uploaded the training data. Using the provided notebook 'train\_and\_deploy-solution.ipynb', I trained a PyTorch model using the ResNet18 architecture. The model was trained with appropriate normalization and preprocessing steps. After training, the model was deployed to a SageMaker endpoint named ‘**pytorch-inference-2025-08-19-09-52-38-587'**.

## Step 2: EC2 Training

I launched an EC2 instance and connected to it via SSH. Using the script 'ec2train1.py', I trained the model locally and saved it to the 'TrainedModels' directory. A screenshot was taken to confirm the saved model file.

## Step 3: Lambda Function Setup

I created a Lambda function named 'InvokeModelFunction' and configured it to invoke the SageMaker endpoint. The function uses boto3 to send image data (via URL or base64) to the endpoint and returns the prediction results. The Lambda function was tested successfully and returned a list of class probabilities.

## Step 4: Security and Testing

I attached the 'AmazonSageMakerFullAccess' policy to the Lambda execution role to ensure proper permissions. The Lambda function was tested using a sample image URL and returned a list of 133 class probabilities. I reviewed IAM roles and policies and ensured that only necessary permissions were granted. Screenshots of the IAM dashboard and Lambda test results were captured.

## Step 5: Concurrency and Auto-Scaling

I published a new version of the Lambda function and created an alias to enable concurrency settings. Reserved concurrency was set to 5 to handle multiple simultaneous requests efficiently. For the SageMaker endpoint, I configured auto-scaling with appropriate min and max instance counts and target utilization. These settings were chosen to balance cost and performance based on expected traffic.

## Final Summary

All steps of the project have been completed successfully, including training, deployment, EC2 training, Lambda setup, security configuration, and concurrency/auto-scaling. Screenshots and code files have been collected for submission. The system is now capable of receiving image inputs and returning class predictions via a scalable and secure AWS infrastructure.