WRITE-UP

1. **Student justifies the type of Sagemaker instance they created**

I chose the ml.t3.medium instance type for the following reasons:

- **Cost-Efficiency**: It is one of the lowest-cost options available in SageMaker, making it ideal for light workloads. This helps manage budget constraints during early stages of model development.
- **Balanced Performance:** While not as powerful as GPU-enabled instances, ml.t3.medium offers sufficient CPU-based computing power for running Jupyter notebooks, performing basic data analysis, and prototyping models.
- **Fast Launch Time**: Compared to more resource-heavy instances, t3.medium launches quickly, enabling faster iteration and productivity for initial development work.
- **Auto-scaling Option:** Since it's a burstable instance type, it can handle brief periods of high CPU usage, making it versatile for many standard tasks.

2. **Writeup contains a description of the EC2 instance created, including justification for the chosen type**

I chose the ml.t3.medium instance type for the reasons mentioned in my previous answer.

3. **Writeup containing EC2 code and Jupyter Notebook code comparison**

The EC2 code only train the data with ResNet50, whereas the Jupyter Notebook contains all the Sagemaker pipeline code, from training, deploying and testing.

4. **The writeup contains a description of potential vulnerabilities in the IAM setup, including the following: roles that have "FullAccess" policies attached, roles that are old or inactive, and roles with policies for functions that the project is no longer using.**

FullAccess policies are strongly discouraged in Production. The best approach is the Lambda role to Invoke a specific Sagemaker endpoint by following the Least Priviledge Access Principle.

5. **Write up on the lambda function**

- This Lambda is a simple “inference proxy” that:
  i. Receives an incoming event payload (expected to be JSON).
  ii. Forwards it to a SageMaker real-time endpoint.

Returns whatever prediction JSON the endpoint produces, wrapped in an HTTP-style response.

6. **The writeup clearly describes traffic, cost, and efficiency considerations for both concurrency and auto-scaling.**

Region default

In us-east-2, the default unreserved concurrency quota is 10 concurrent executions per endpoint. That means even if I enable provisioned concurrency, I won’t be able to scale above 10 unless I first request a service-quota increase.

Provisioned vs. Unreserved
Since I unreserved cap was already at 10, the console didn't surface any higher provisioned settings—provisioned concurrency only becomes adjustable once you have unreserved headroom or an approved limit bump.

Auto-Scaling Considerations
I configured target-tracking scaling rules on your SageMaker endpoint

Low-Traffic & Cost
With only a handful of inference calls, I found autoscaling overkill—and leaving a single instance running keeps costs minimal.