Operationalizing an AWS ML Project

Project developed for AWS Machine Learning Engineer nanodegree offered by Udacity (2023):

- Notebook instance chosen: ml.t3.medium was chosen as a notebook instance because it provides a balance between performance and cost.
- EC2 instance chosen: m5.xlarge was chosen because it offers a balance of compute power and memory resources. The m5.xlarge instance type is part of the Amazon EC2 M5 family, which is optimized for general-purpose workloads. It provides a good balance of compute, memory, and networking resources, making it suitable for a wide range of applications.
- Differences between train_and_deploy-solution.ipynb and ec2train1.py: The code in train_and_deploy-solution.ipynb and ec2train1.py serves the same purpose, which is to train a machine learning model. However, there are a few differences between the two. First, the code in train_and_deploy-solution.ipynb is specifically written to work in Amazon SageMaker, while the code in ec2train1.py is adapted to run on an EC2 instance. This means that some modules used in train_and_deploy-solution.ipynb can only be used in SageMaker. Second, the code in ec2train1.py has been adapted from the functions defined in the hpo.py starter script. This adaptation was necessary to make the code compatible with running on an EC2 instance. Overall, the differences between the two codes lie in the modules used and the adaptations made to run on different platforms.
- Explanation about lambda function: This Lambda function is designed to receive an event, which is expected to be a JSON object, invoke a SageMaker endpoint using the event data, and return the response from the endpoint as a JSON-encoded body in the Lambda response.
- Vulnerabilities of AWS workspace: It's important to strike a balance between functionality and security when granting permissions to your Lambda function's IAM role. Consider the specific requirements and risk tolerance of your application and follow the principle of least privilege to minimize potential vulnerabilities. Regularly review and update your security measures to align with evolving threats and best practices.
- Concurrency choices: By choosing a reserved concurrency of 5, you ensure that your Lambda function operates within resource limitations, maintains stability, distributes the workload evenly, and provides cost control. The provisioned concurrency of 2 ensures instant responsiveness, predictable performance, cost optimization, and resilience to handle spikes in traffic while minimizing idle resource costs. These choices strike a balance between concurrency requirements, performance objectives, and cost efficiency.
- Auto scaling choices: The maximum instance count of 3 ensures that the autoscaling group can scale up to a maximum of three instances to handle increased demand while avoiding unnecessary overprovisioning. The target value of 20 defines the desired average CPU utilization or other metric per instance, ensuring that the group maintains a balanced workload distribution. The scale in and scale out cool down

periods of 30 seconds provide a buffer to prevent rapid scaling actions, allowing time for the system to stabilize and avoid unnecessary oscillations. These choices enable efficient resource allocation and responsiveness to fluctuating workloads while maintaining stability and cost control.