Operationalizing an AWS ML Project

Table of Contents

Notebook Setup

I have chosen the 'ml.t2.medium' instance type for the following reasons. The execution of the code does not require a very computationally powerful CPU and high RAM, hence we should look at smaller instances**.** To avoid high costs, we should select a notebook that is low in per hour cost while meeting the require CPU and RAM needs. Comparing 'ml.t2.medium' and 'ml.t3.medium', the former is cheaper due to slower boot time while having the same 2 vCPU and 4GB memory. Since boot time speed is not important, the former was chosen.

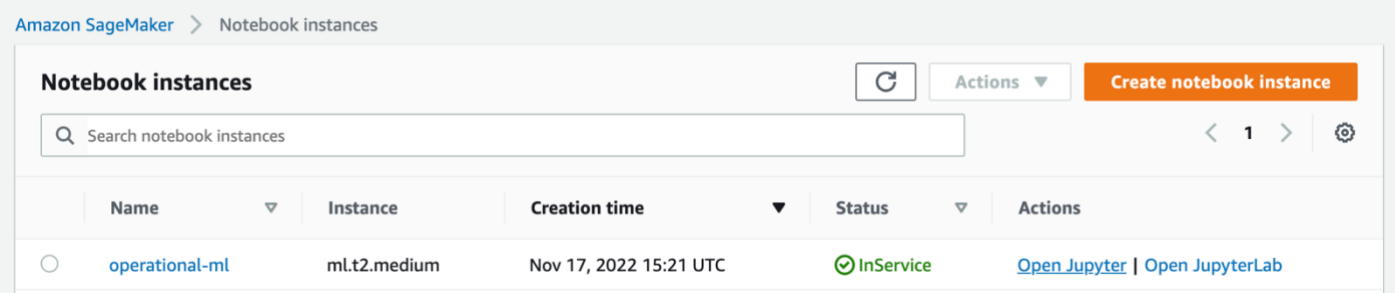

Figure 1. Sagemaker Notebook Instance

The dog breed dataset has been uploaded to the S3 bucket using sagemaker.

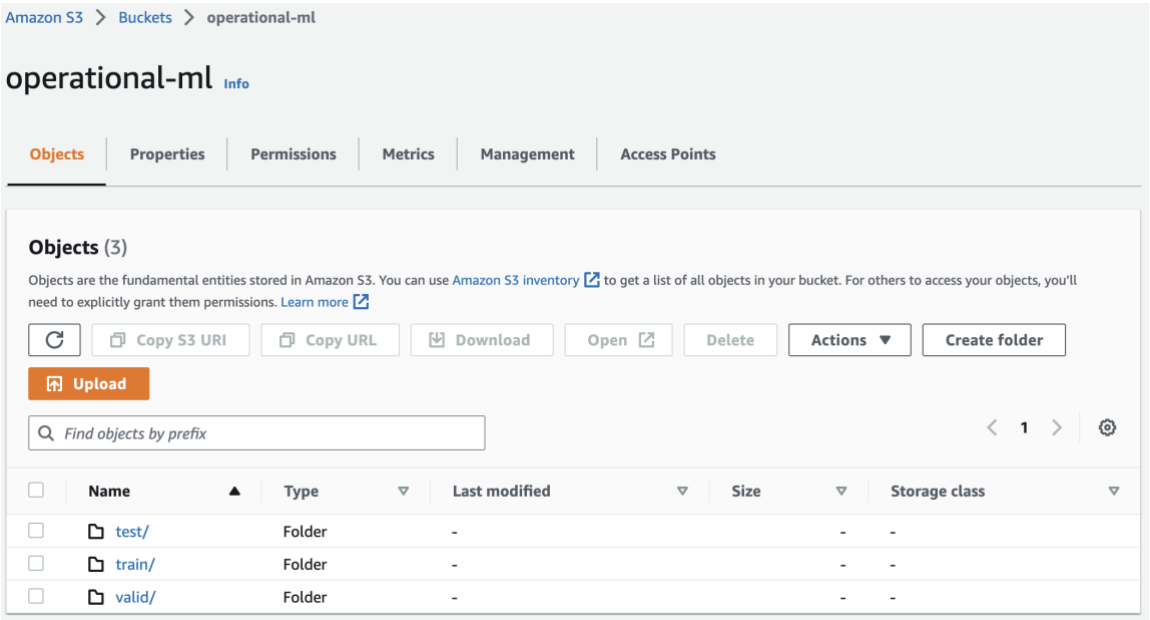

Figure 2. S3 Bucket

Sagemaker Training and Deployment

For hyperparameter tuning, the 'ml.m5.2xlarge' instance type, which has 8 vCPU and 32 GB of RAM at a cost of $0.461 per hour, was used. Training was done using a multi-instance format, with max jobs of 6 and max parallel jobs of 3.

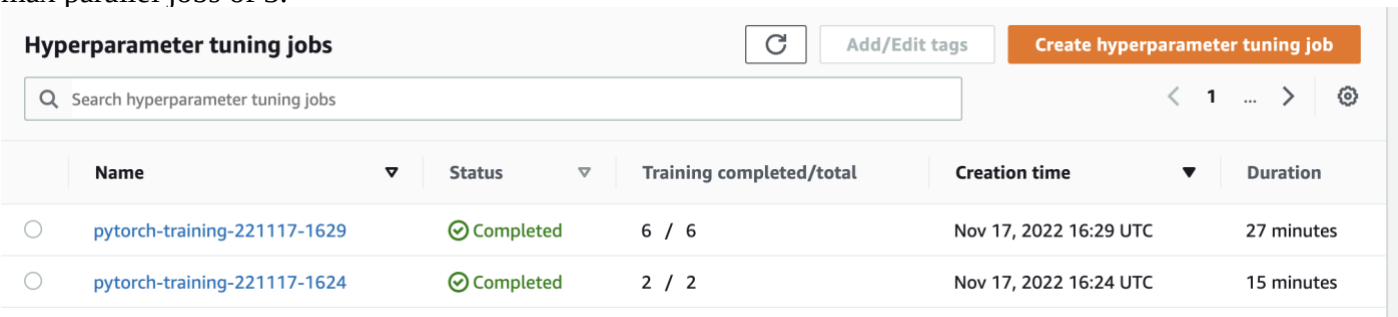

Figure 3. Hyperparameter Tuning Job

For training, the 'ml.m5.2xlarge' instance type, which has 8 vCPU and 32 GB of RAM at a cost of $0.461 per hour, was used. Training was done using a multi-instance format, with max jobs of 6 and max parallel jobs of 3.

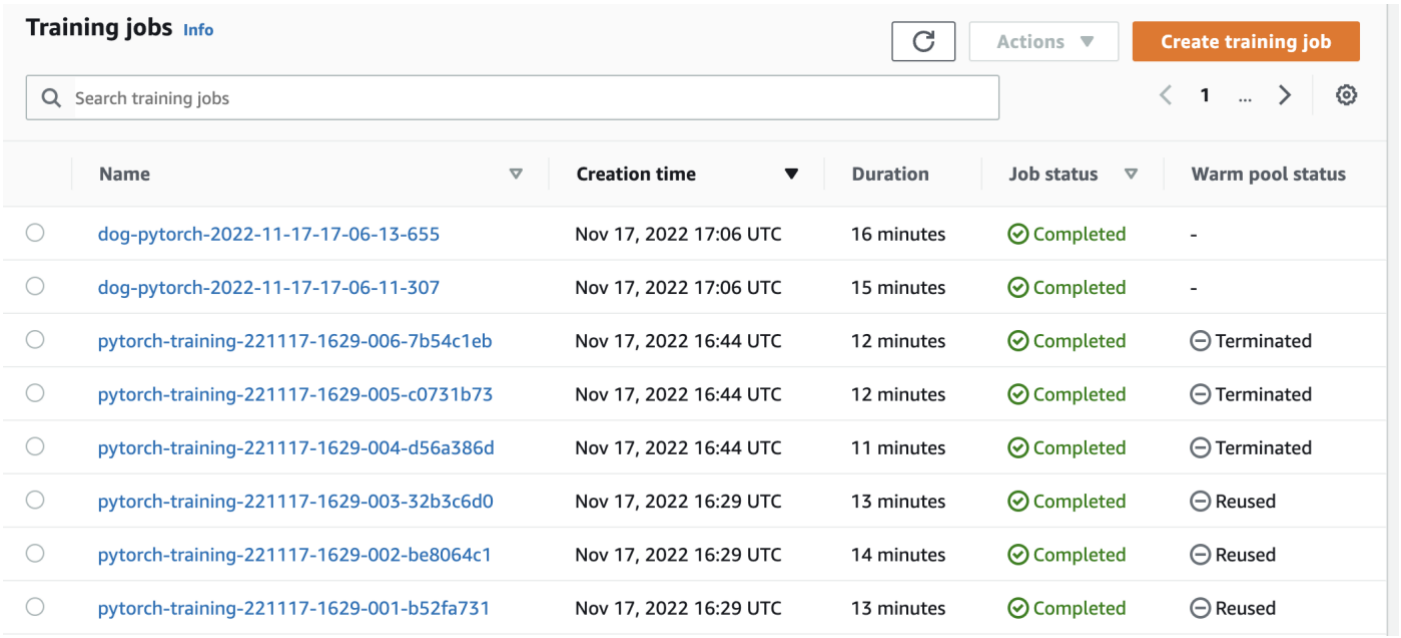

Figure 4. Multi-Instance Training Job

Multi instance deployed endpoint: pytorch-inference-2022-11-17-17-22-57-546

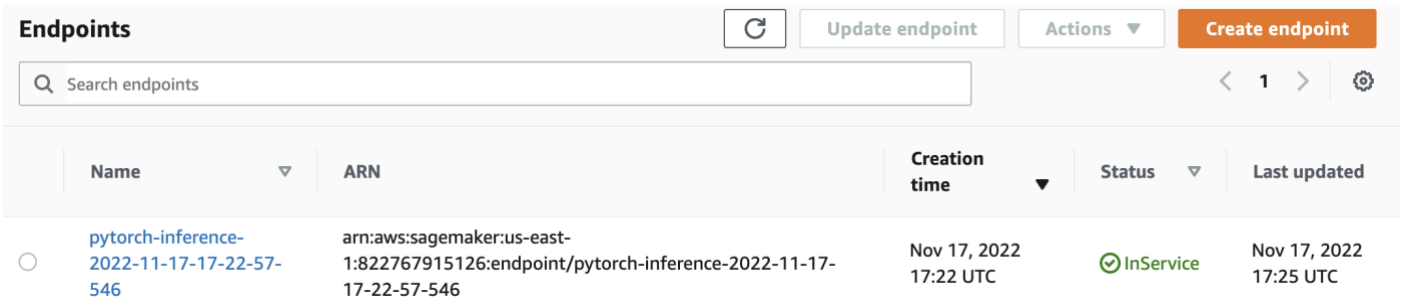

Figure 5. Sagemaker Endpoints

EC2 Training

The **t2.2xlarge** instance and the **Deep Learning AMI (Amazon Linux 2)** was used. Given that t2 instances can sustain high CPU performance for long periods without incurring extraordinary costs, this instance is a good mix of performance and affordability.

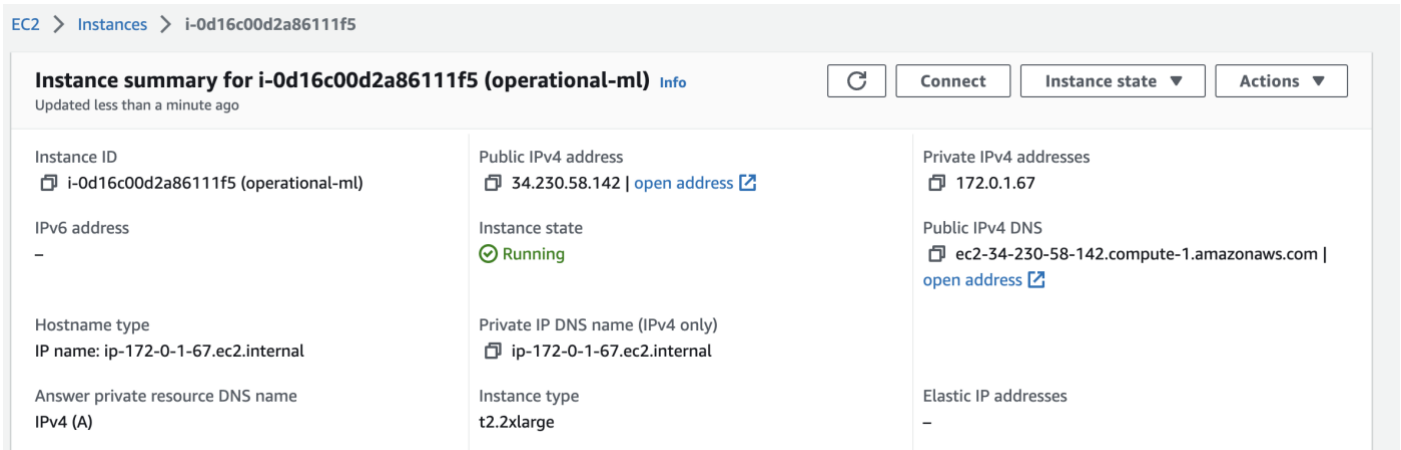

Figure 6. EC2 Instance Info

```
[root@ip-172-0-1-67 ~]# source activate pytorch
NOTE that the Amazon EC2 t2.2xlarge instance type is not supported by current Deep Learning AMI. Please review the DLAMI release notes https://docs.aws.amazon
.com/dlami/latest/devguide/appendix-ami-release-notes.html for supported Amazon EC2 instance types.
(pytorch) [root@ip-172-0-1-67 ~]# python solution.py
/opt/conda/envs/pytorch/lib/python3.9/site-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may
 be removed in the future, please use 'weights' instead.
  warnings.warn(
/opt/conda/envs/pytorch/lib/python3.9/site-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or `None` for 'weights'
are deprecated since 0.13 and may be removed in the future. The current behavior is equivalent to passing `weights=ResNet50_Weights.IMAGENET1K_V1`. You can al
so use `weights=ResNet50_Weights.DEFAULT` to get the most up-to-date weights.
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/resnet50-0676ba61.pth" to /root/.cache/torch/hub/checkpoints/resnet50-0676ba61.pth
100%|                                                                                                  | 97.8M/97.8M [00:00<00:00, 115MB/s]
Starting Model Training
saved
(pytorch) [root@ip-172-0-1-67 ~]# ls
dogImages  dogImages.zip  solution.py  TrainedModels
(pytorch) [root@ip-172-0-1-67 ~]# cd TrainedModels/
(pytorch) [root@ip-172-0-1-67 TrainedModels]# ls
model.pth
```

Figure 7. EC2 Training Saved - model.pth

Item	EC2	Sagemaker
Dataset/Model	Takes from local path	Takes from S3 bucket
Hyperparameter Tuning	Internal script	External script
Instances	Script and training job on the same instance	Script and training job on separate instance

Figure 8. Comparison of EC2 and Sagemaker scripts

Lambda functions

The lambda function will be used to invoke the deploy endpoint for the multi-instance trained model.

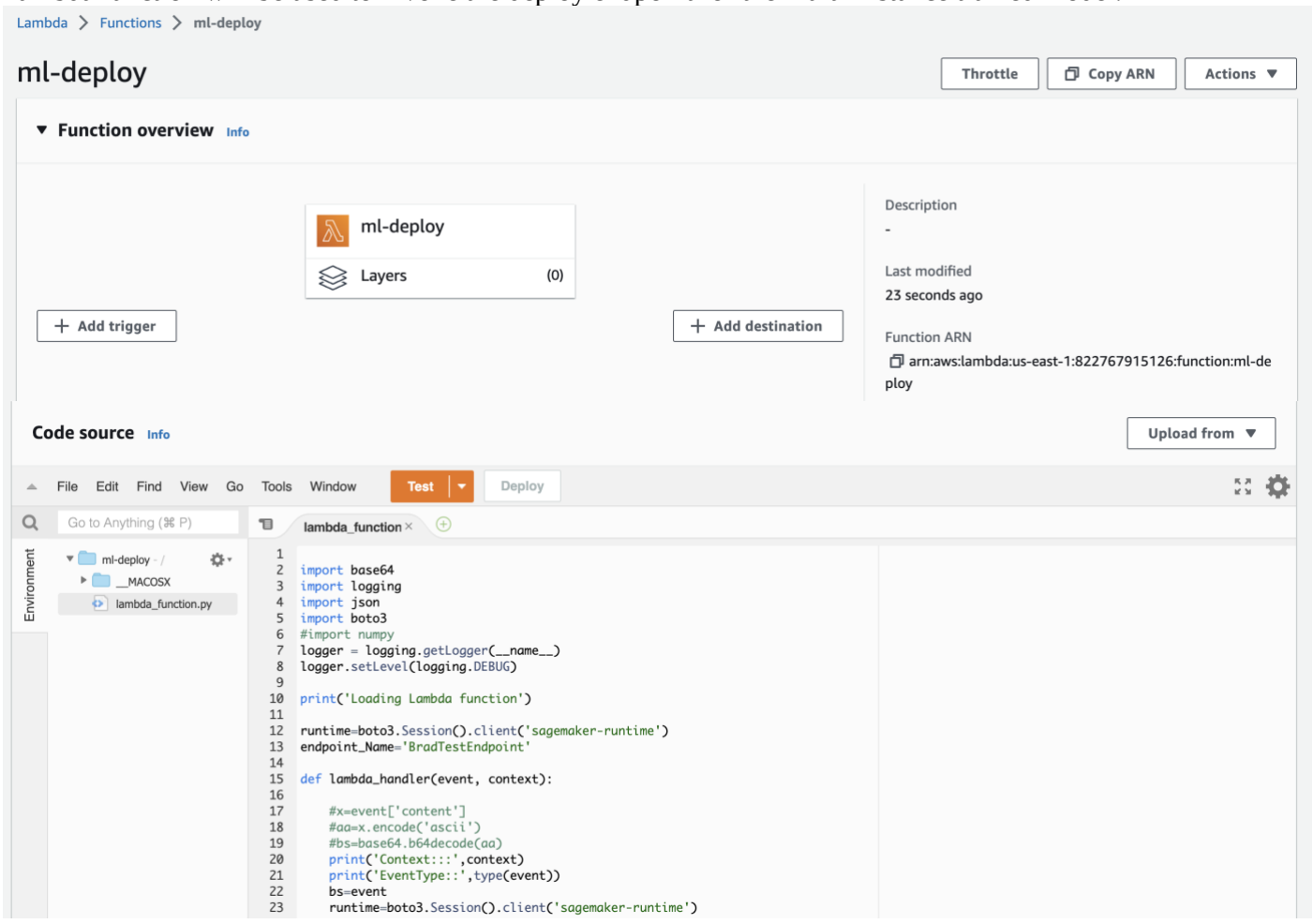

Figure 9. Lambda Function

Security and Testing

Test event was executed in lambda function.

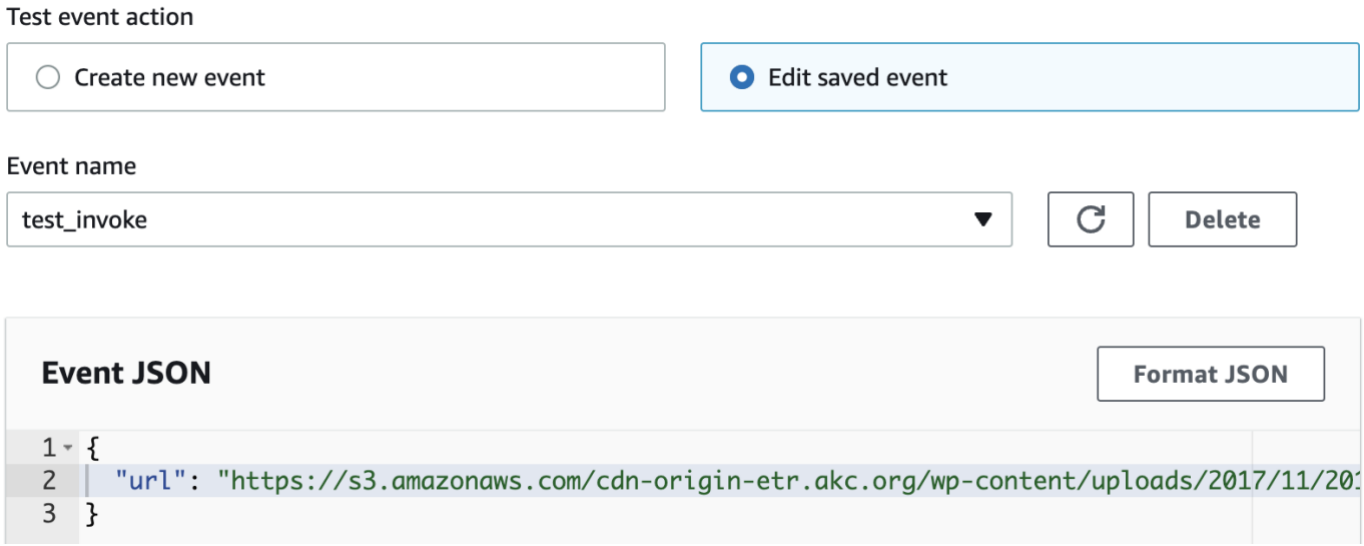

Figure 10. Lambda Function Test Event

Upon execution of test evet, we got 'AccessDeniedException' as the lambda function does not have access to Sagemaker.

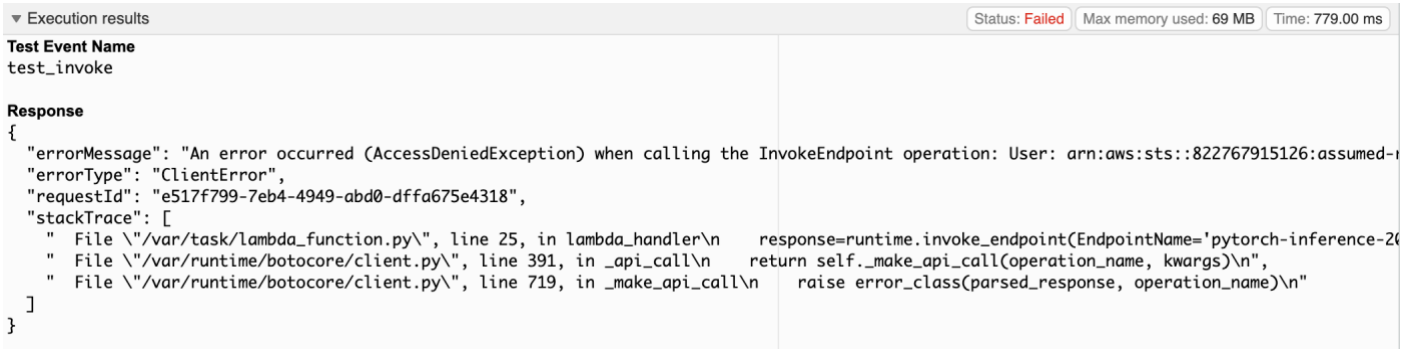

Figure 11. Lambda function test event failure response

The 'SageMakerFullAccess' policy was added to lambda function's role.

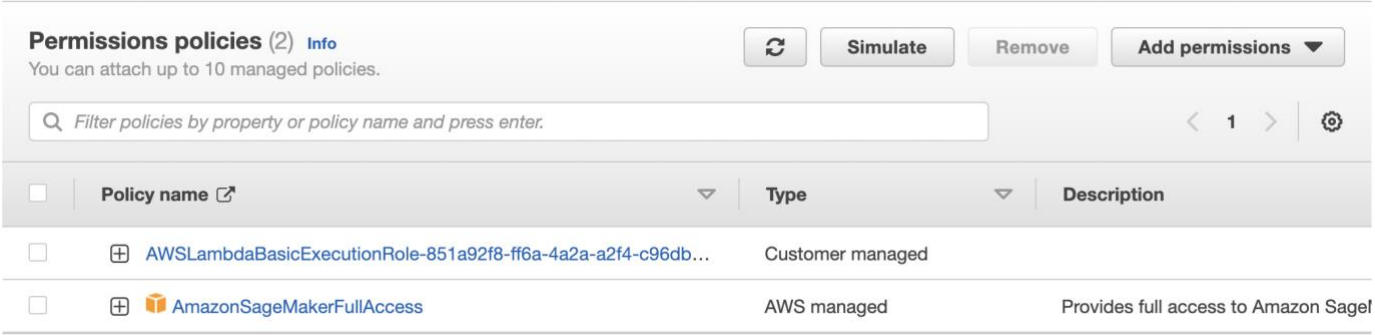

Figure 12. Lambda function role IAM permissions

Test event was successfully executed, and 33 dog breeds was indicated in the result.

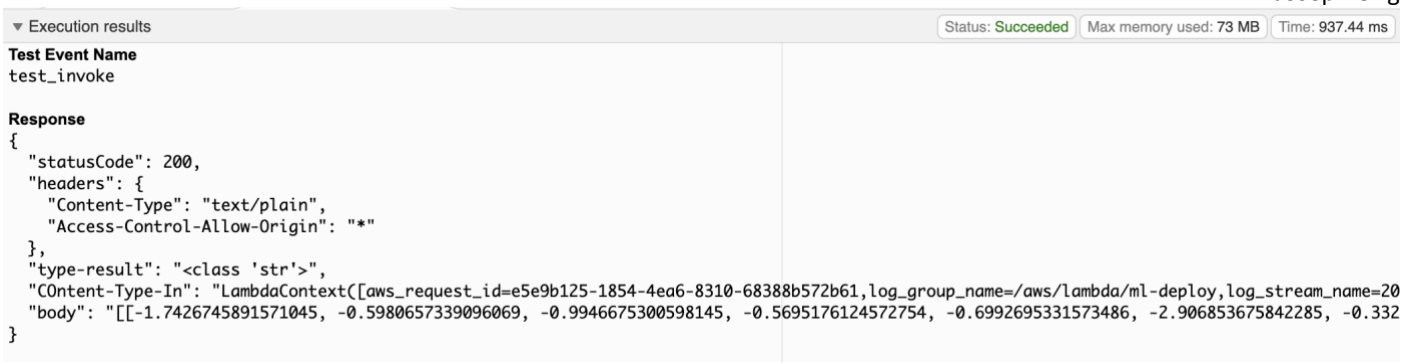

Figure 13. Lambda function test event success response

While we have given the lambda function full access to sagemaker as well as the sagemaker notebook full access to S3, it is possible to add further granular permissions to allow access to a specific notebook or s3 bucket.

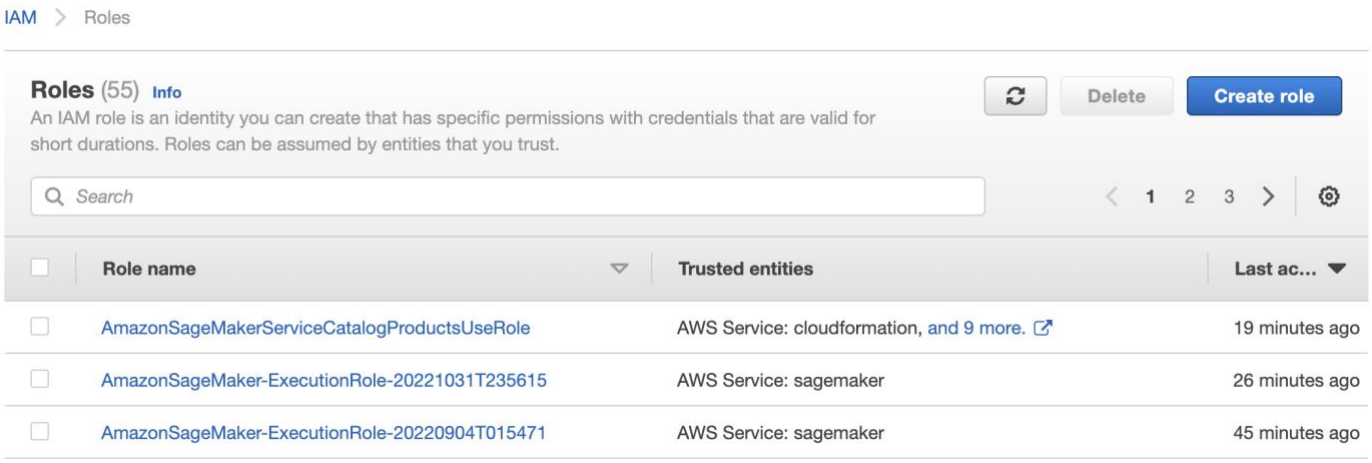

Figure 14. IAM Roles

Concurrency and Auto-scaling

Version configuration for our lambda function was created.

Concurrency is set to 5, which means that the function can handle up to 5 requests at the same time. We will only be using provisioned concurrency, as we are not given the maximum number of requests.

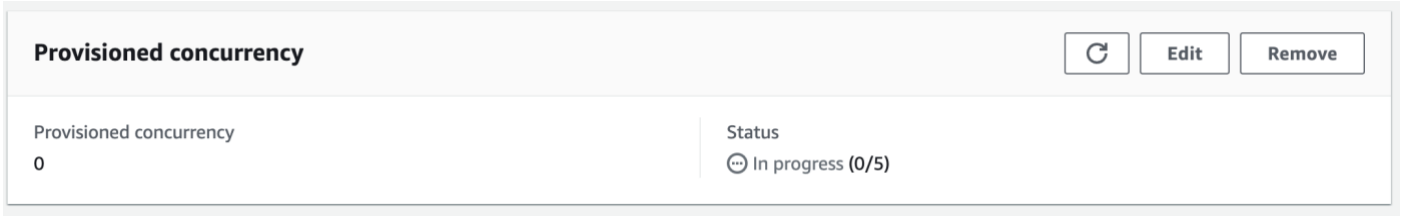

Figure 15. Lambda Function Provision Concurrency

Auto-scaling is set to a maximum of 3 instances, which means in time of high number of requests, 2 additional instances will be deployed. In addition, a scale-in and scale-out cooldown time of 30 seconds was used to ensure that users are not experiencing high latency.

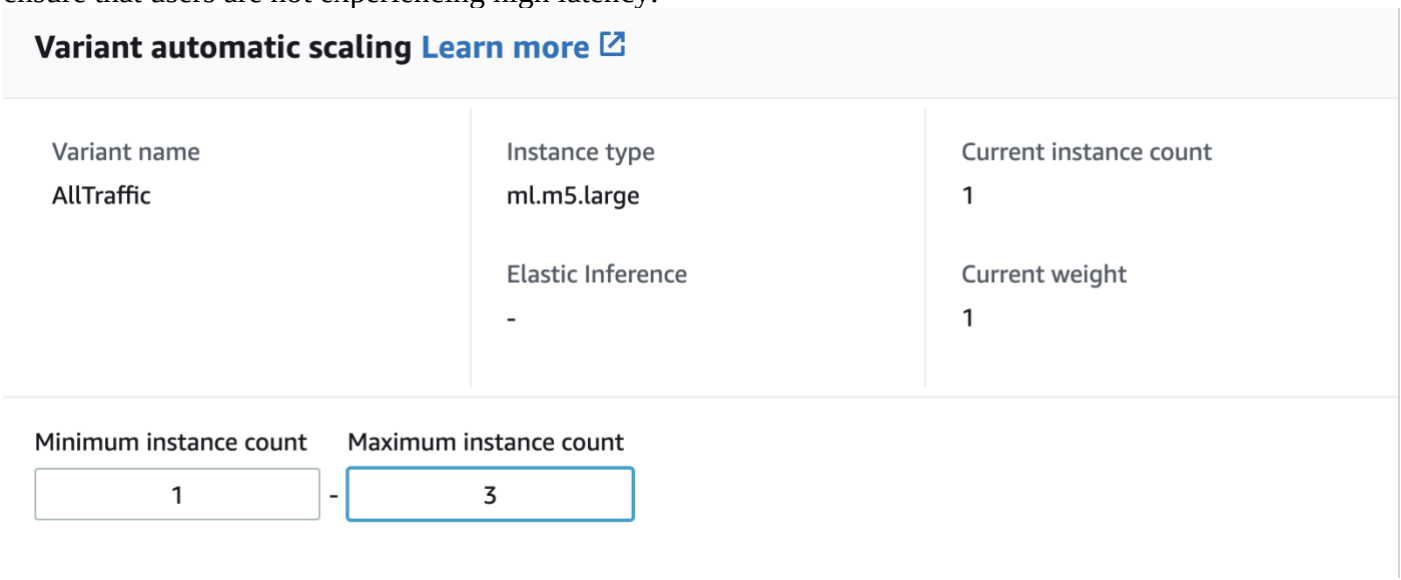

Figure 16. Endpoint Auto-scaling Config