

Write Up of the Project: Operationalizing an AWS Machine Learning

Samin Poudel*

This project is started with an objective of learning the process of adjusting, improving, configuring, and preparing a trained model for production graded deployment using the different tool of AWS Sagemaker.

Notebook Instance, Training and Deployment

We started the project by creating a notebook instance named AWS-Chapter5-project with Standard Type of Notebook **ml.t2.medium** as shown in Fig 1. This instance was chosen as it is the most optimal choice in terms of optimizing cost. An image classification task might need instances with higher RAM and capacity, but the idea was to check if the image classification task can be performed with optimal cost in a reasonable time.

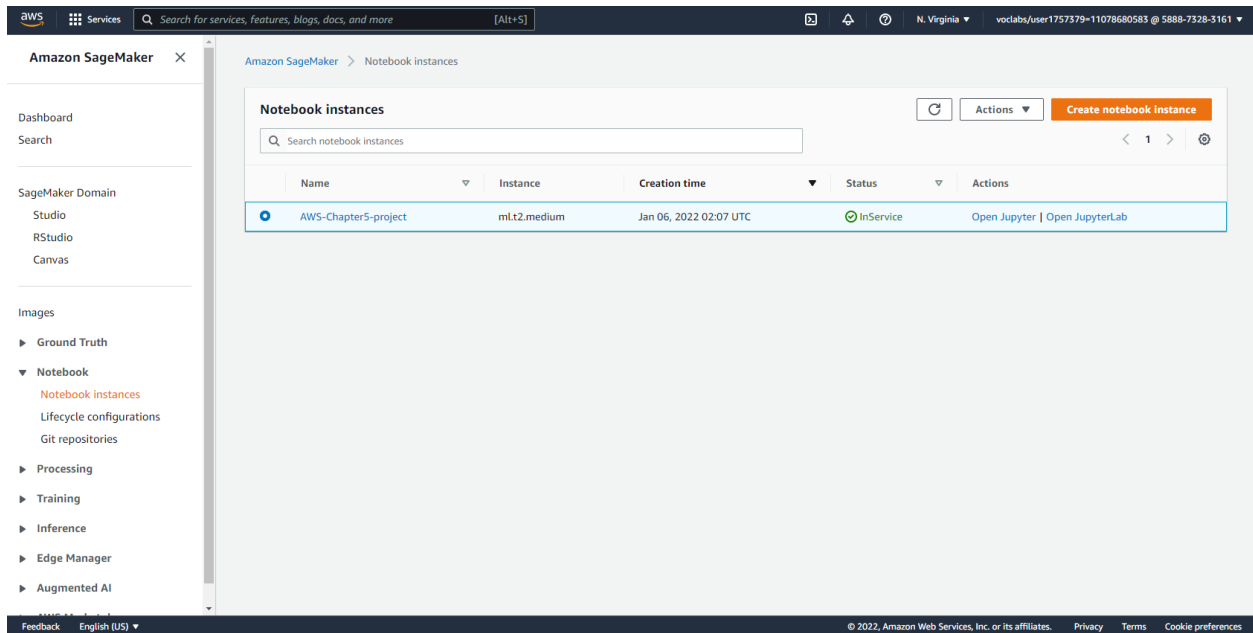


Figure 1. notebook Instance

After that using the created instance, the dog breed data was downloaded and then uploaded to a S3 bucket named chapter5project by running code in the jupyter notebook named train_and_deploy_solution.ipynb. The S3 bucket with data is as shown in Fig. 2.

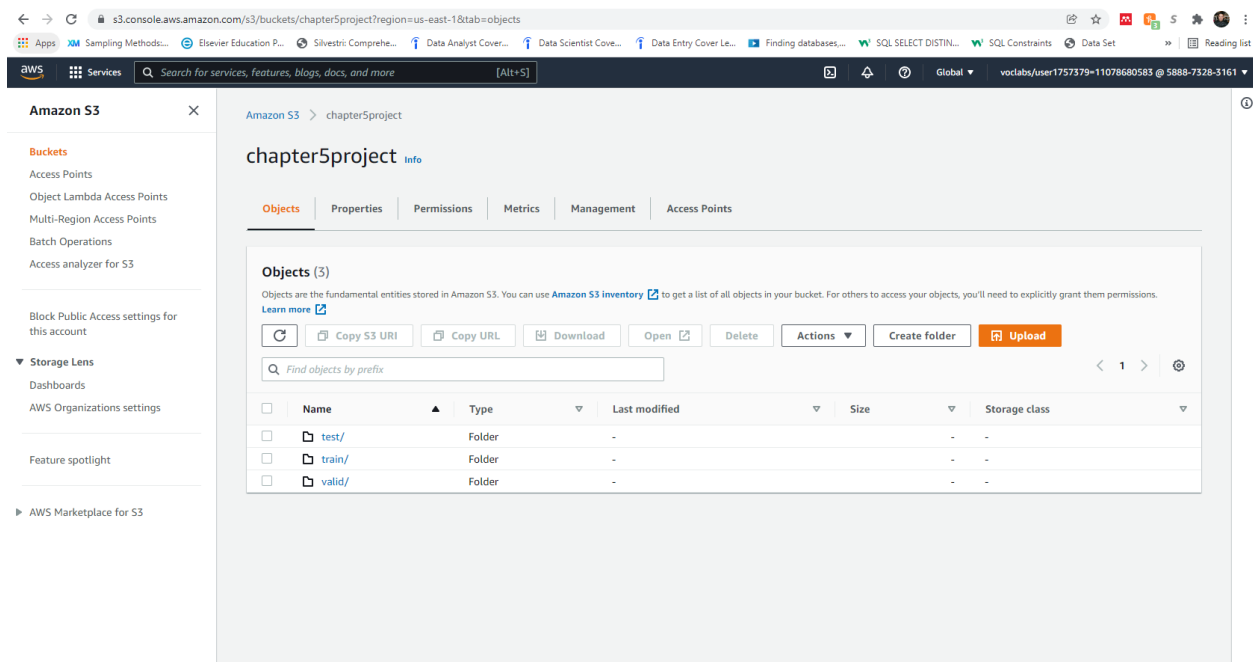


Figure 2. S3 Bucket with Data Uploaded

After that we performed hyperparameter tuning, single instance training of model, multi-instance training of model. Next, we deployed the endpoints for both single instance and multi-instance models. The screenshots of tuning, training and deployment are attached and the process of performing these tasks is as followed in the train_and_deploy_solution.ipynb. file

EC2 Training

Next step was to perform the training in the EC2. The EC2 instance was created, and the training job was performed in it. The AMI assigned to the EC2 instance was Deep Learning AMI (Amazon linux 2) Version 56.0 to make sure the required modules were available for the python code we were executing.

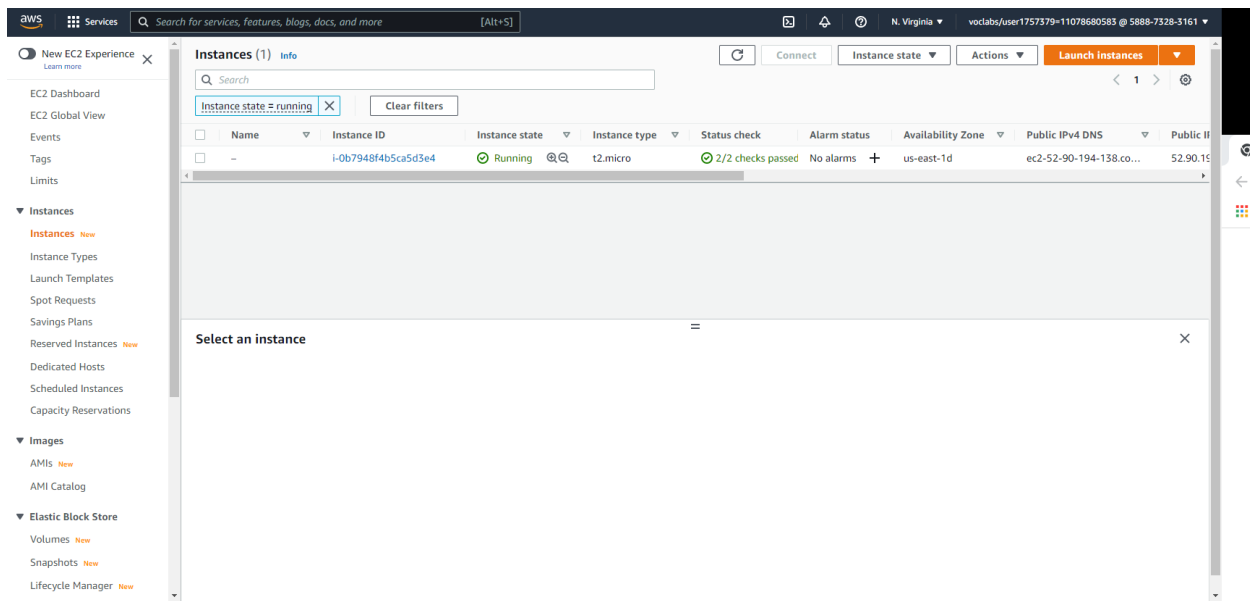


Figure 3. Running EC2 instance

```
For a fully managed experience, check out Amazon SageMaker at https://aws.amazon.com/sagemaker
When using INF1 type instances, please update regularly using the instructions at: https://github.com/aws/aws-neuron-sdk/tree/master/release-notes
Security Scan reports for python packages are located at: /opt/aws/dlami/info/
=====
No packages needed for security: 3 packages available
Run "sudo yum update" to apply all updates.
[root@ip-172-31-94-82 ~]# source activate amazonei_pytorch_latest_p37
Could not find conda environment: amazonei_pytorch_latest_p37
You can list all discoverable environments with 'conda info --envs'.

[root@ip-172-31-94-82 ~]# ls
dogImages dogImages.zip solution.py TrainedModels
[root@ip-172-31-94-82 ~]# python solution.py
Traceback (most recent call last):
  File "solution.py", line 2, in <module>
    import torch
ModuleNotFoundError: No module named 'torch'
[root@ip-172-31-94-82 ~]# source activate amazonei_pytorch_latest_p37
ln: failed to create symbolic link '/root/anaconda3/envs/amazonei_pytorch_latest_p37/bin/ei': No such file or directory
ln: failed to create symbolic link '/root/anaconda3/envs/amazonei_pytorch_latest_p37/bin/health_check': No such file or directory
Please run 'python ~/anaconda3/bin/EISetupValidator.py' if you experience issues using Amazon EI service.
(amazoni_pytorch_latest_p37) [root@ip-172-31-94-82 ~]# ls
dogImages dogImages.zip solution.py TrainedModels
(amazoni_pytorch_latest_p37) [root@ip-172-31-94-82 ~]# python solution.py
Downloading: "https://download.pytorch.org/models/resnet50-19c8e357.pth" to /root/.cache/torch/checkpoints/resnet50-19c8e357.pth
100% | 97.8M/97.8M [00:00<00:00, 113MB/s]
Starting Model Training
/pytorch/aten/src/ATen/native/BinaryOps.cpp:81: UserWarning: Integer division of tensors using div or / is deprecated, and in a future release div will perform true division
as in Python 3. Use true_divide or floor_divide (// in Python) instead.
saved
(amazoni_pytorch_latest_p37) [root@ip-172-31-94-82 ~]# cd Trained models
-bash: cd: Trained: No such file or directory
(amazoni_pytorch_latest_p37) [root@ip-172-31-94-82 ~]# cd TrainedModels
(amazoni_pytorch_latest_p37) [root@ip-172-31-94-82 TrainedModels]# ls
model.pth
(amazoni_pytorch_latest_p37) [root@ip-172-31-94-82 TrainedModels]#
```

Figure 4. Execution in EC2

We had to perform the execution of the model in EC2 like one time execution script, while the one in SageMaker notebook had the flexibility of step by step. SageMaker gives us the ability to tune the best hyperparameters and use them, but in EC2 script hyperparameters are to be predefined. EC2 are cost effective than SageMaker, while SageMaker seemed to be more user friendly than the EC2 code.

Lambda Function Set Up, Security and Testing

A Lambda function was set up to invoke the endpoint we created in step 1 of this project as in Fig. 5.

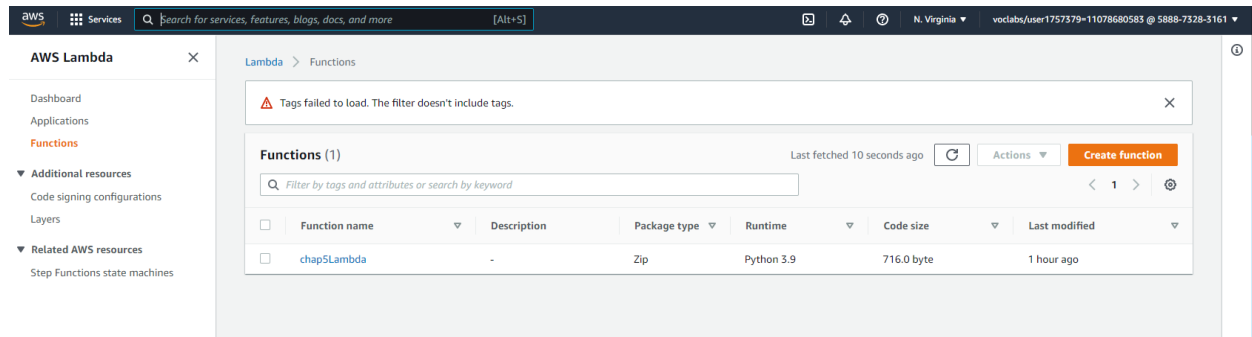


Figure 5. Lambda Function

Then the given test case was applied to lambda but the execution failed as the lambda Function did not have the policy attached to access sagemaker and invoke the endpoint. Therefore, we added the Sagemaker Full Access policy to the lambda Function as shown in Fig6. and the execution with the test case was possible as shown in Fig. 7. The testing of Lambda Function gave list of 33 numbers which the prediction of the class membership of the picture submitted from the test case.

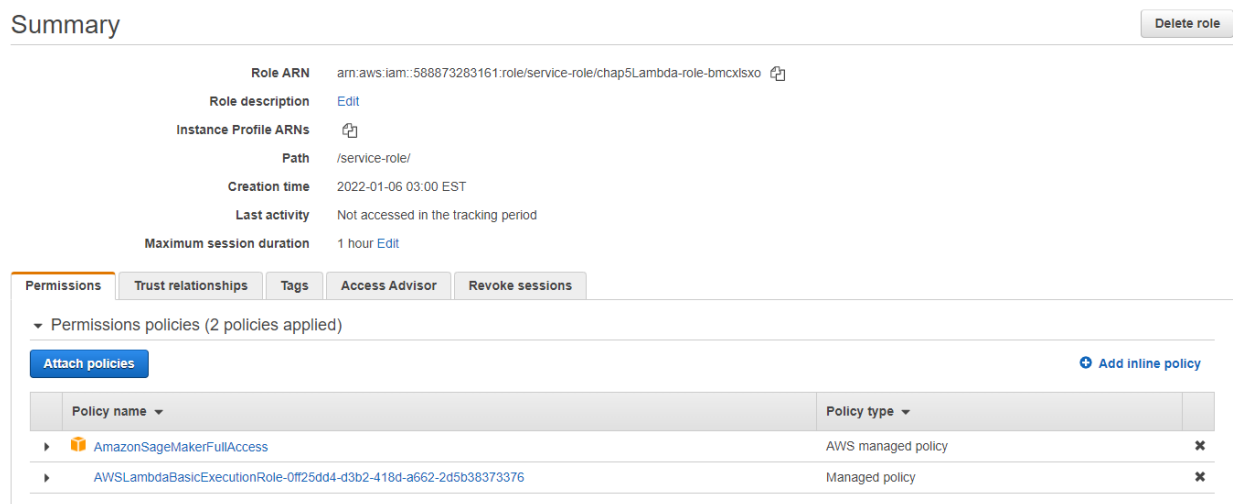


Figure 6 Lambda Function: Policy Added

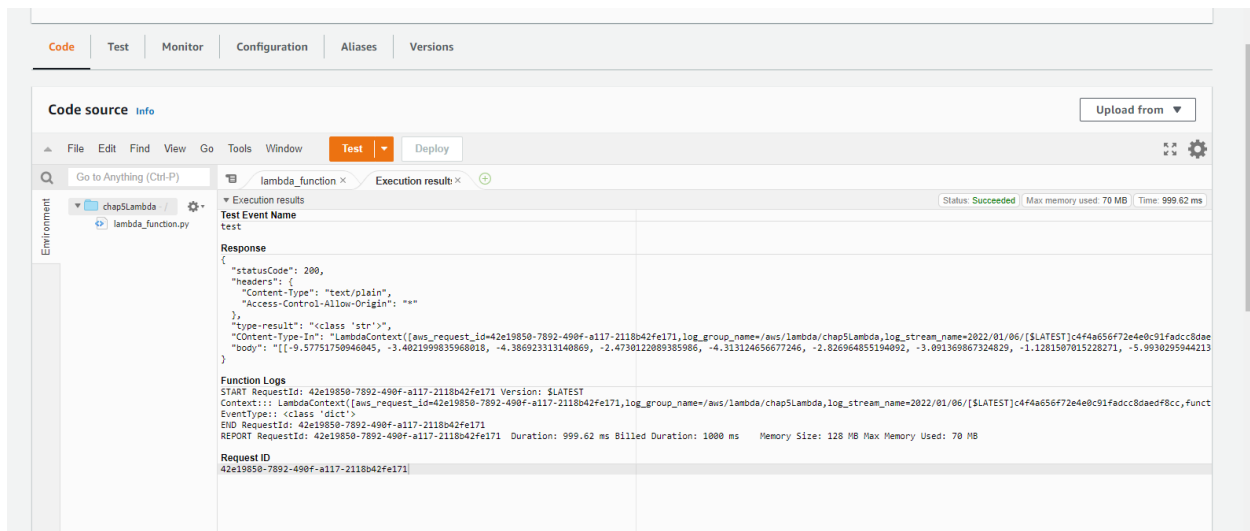


Figure 7. Lambda Function Successful Execution with the Test Case

A list of IAM roles and the is as shown in Fig. 8 and the two policies for lambda Function is already shown in the Fig. 6.

Identity and Access Management (IAM)			
Role name	Trusted entities	Last activity	
<input type="checkbox"/> AmazonSageMaker-ExecutionRole-20220105T210671	AWS Service: sagemaker	30 minutes ago	
<input type="checkbox"/> AWSServiceRoleForAWSCloud9	AWS Service: cloud9 (Service-Linked Role)	-	
<input type="checkbox"/> AWSServiceRoleForCloudWatchEvents	AWS Service: events (Service-Linked Role)	-	
<input type="checkbox"/> AWSServiceRoleForElasticCache	AWS Service: elasticache (Service-Linked Role)	-	
<input type="checkbox"/> AWSServiceRoleForGlobalAccelerator	AWS Service: globalaccelerator (Service-Linked Role)	-	
<input type="checkbox"/> AWSServiceRoleForOrganizations	AWS Service: organizations (Service-Linked Role)	-	
<input type="checkbox"/> AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)	-	
<input type="checkbox"/> AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linked Role)	-	
<input type="checkbox"/> chap5Lambda-role-bmcxisxo	AWS Service: lambda	-	
<input type="checkbox"/> EMR_AutoScaling_DefaultRole	AWS Service: application-autoscaling, and 1 more	-	
<input type="checkbox"/> EMR_DefaultRole	AWS Service: elasticmapreduce	-	
<input type="checkbox"/> EMR_EC2_DefaultRole	AWS Service: ec2	-	
<input type="checkbox"/> robomaker_students	AWS Service: greengrass, and 3 more	-	
<input type="checkbox"/> vocareum	Account: 137949000194	-	
<input type="checkbox"/> voclabs	Account: 137949000194	-	
<input type="checkbox"/> vocstartsoft	Account: 137949000194	-	

Figure 8. A List of IAM Roles

Addition of AmazonSageMakerFullAccess policy may add some sort of insecurity, so for more secure set up can be based on the location and the role of the endpoint that the lambda function is trying to invoke.

Concurrency and Auto-Scaling

The concurrency for lambda function was created by setting up a version for it. The task of the concurrency is to help during the high traffic input requests, but at it may add some cost, care must be taken to add the amount of concurrency. Similar idea goes for autoscaling of the Endpoints. The screenshot of concurrency of lambda function is as shown in Fig. 9. And, the autoscaling of an endpoint is as shown in Fig. 10.

The screenshot displays the AWS Lambda console interface for a function named 'chap5Lambda'. The 'Configuration' tab is selected, and the 'Concurrency' section is active. The 'Function concurrency' is set to 'Use unreserved account concurrency', and the 'Unreserved account concurrency' is 999. Below this, the 'Provisioned concurrency configurations (1)' section shows a single configuration with a 'Qualifier' of '1', a 'Type' of 'version', and a 'Provisioned concurrency' of '1'. The status is 'Ready'.

Qualifier	Type	Provisioned concurrency	Status	Details
1	version	1	Ready	-

Figure 9 Concurrency of Lambda Function

Amazon SageMaker

×

Dashboard

Search

SageMaker Domain

Studio

RStudio

Canvas

Images

▶ Ground Truth

▶ Notebook

▶ Processing

▶ Training

▼ Inference

Compilation jobs

Model packages

Models

Endpoint configurations

Endpoints

Batch transform jobs

▶ Edge Manager

▶ Augmented AI

▶ AWS Marketplace

Amazon SageMaker > Endpoints > pytorch-inference-2022-01-06-03-57-09-618 > AllTraffic

Configure variant automatic scaling

Deregister auto scaling

Variant automatic scaling [Learn more](#)

Variant name	Instance type	Current instance count
AllTraffic	m5.large	1
	Elastic inference	Current weight
	-	1

Minimum instance count

Maximum instance count

1 ~ 7

IAM role

Amazon SageMaker uses the following service-linked role for automatic scaling. [Learn more](#)

AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint

Built-in scaling policy [Learn more](#)

Policy name

SageMakerEndpointInvocationScalingPolicy

Target metric

SageMakerVariantInvocationsPerInstance [Learn more](#)

Target value

10

Scale in cool down (seconds) - optional

Scale out cool down (seconds) - optional

25 25

☐ Disable scale in

Select if you don't want automatic scaling to delete instances when traffic decreases. [Learn more](#)

Custom scaling policy [Learn more](#)

There are no custom scaling policies for this variant.

Cancel

Save

Figure 10 Auto-scaling of an Endpoint