

Operationalizing AWS ML Project

Technical Write Up

Alejandro Santillan Iturres

Operationalizing AWS ML Project

As part of requirements for the AWS MLEND

Udacity Project 4: AWS Machine Learning Engineer Nanodegree Program (2021-2022)

In this project we are requested to accomplish five steps towards applying acquired skills about most important services that AWS offer to the operative part of training and deploying Machine Learning models.

The project consists in 5 Steps

Step 1: Train and deploy a model on a Sagemaker notebook (one of the modalities that Sagemaker offer amongst several more).

Step 2: Perform a similar task on an EC2 instance.

Step 3: Create a Lambda function that will consume your model inference capabilities via endpoints.

Step 4: Security and testing.

Step 5: Set up concurrency for your lambda function and auto-scaling for your deployed endpoint.

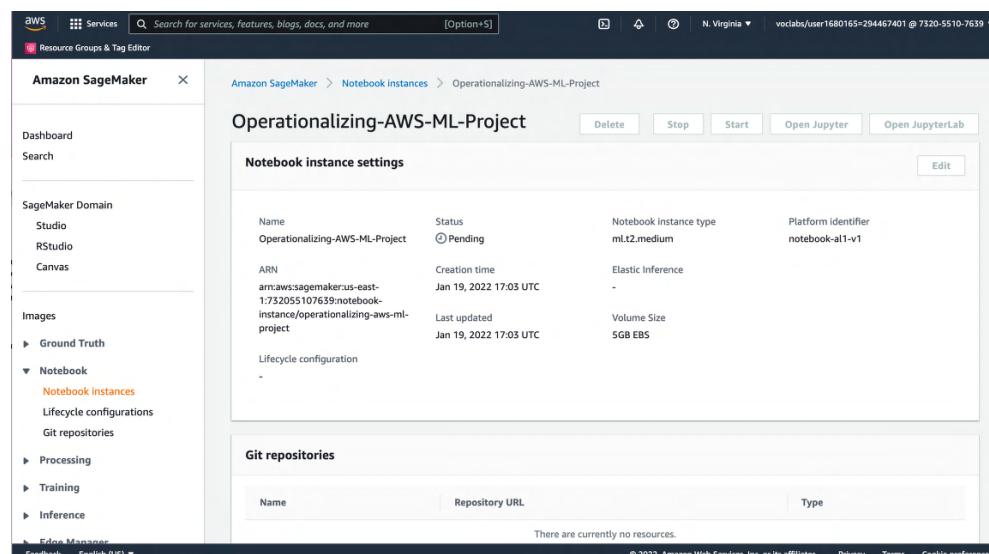
Step 0: Project Set Up and Installation

You will need to have a AWS account, which can be opened for free at <https://aws.amazon.com> You will be able to follow all the steps in this project in a day of work and with a cost of around \$4.00 Follow the instructions in the write up file located here: https://github.com/alelasantillan/Operationalizing_AWS_ML_Project/blob/main/writeup.pdf

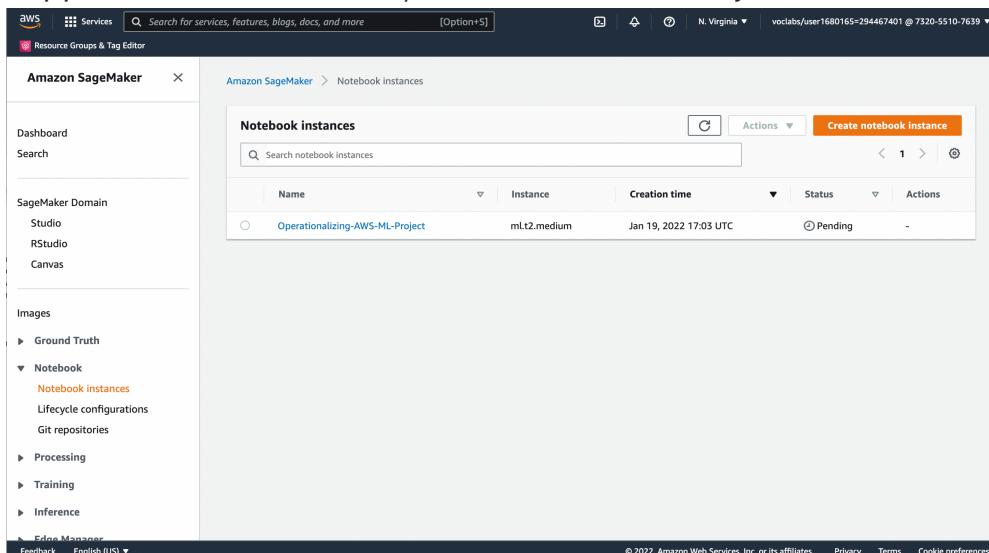
Step 1: Train and deploy a model on a Sagemaker notebook

Notebook Instance setup

1.1 Created a *ml.t2.medium* instance named Operationalizing-AWS-ML-Project. The cost of this instance is not published, but certainly is less than \$0.05/hour, which is the price for *ml.t3.medium* according AWS: <https://aws.amazon.com/sagemaker/pricing/>. This instance will allow me to perform code debugging without incurring in great costs. In general the computing resources are consumed by the processes launched by the notebook rather than the notebook itself, so no much is required in this instance. Once the project is running, I will consider increasing capacity along with the stress testing I will eventually perform. In previous experiences I've used also *ml.m5.large* (\$0.115) with sucess, but this time I will take more care of resources used since in the former project I ended too close to the limit.



I've launched the Notebook Instance, but it took a long time to be ready. It happens from time to time, but is not usual. You just have to wait:



The screenshot shows the Amazon SageMaker console interface. On the left, there's a sidebar with various options like Dashboard, Search, SageMaker Domain, Studio, RStudio, Canvas, Images, Processing, Training, Inference, and Edge Manager. The 'Notebook instances' section is currently selected under the 'Notebook' category. The main area displays a table titled 'Notebook instances' with one row. The row contains the following information:

Name	Instance	Creation time	Status	Actions
Operationalizing-AWS-ML-Project	ml.t2.medium	Jan 19, 2022 17:03 UTC	Pending	-

At the bottom of the page, there are links for Feedback, English (US), and cookie preferences.

1.2 I uploaded the `train_and_deploy-solution.ipynb` into the SageMaker notebook instance, as well as the files `hpo.py` and `infernce2.py` to run the Hyperparameter Optimization part, the training-debugging part and the endpoint deploy part. I adjusted the bucket name in all occurrences and changed the instance types for running the three different process: two `ml.m5.xlarge` for the hyperparameter optimization and training-debugging and `ml.m5.large` for deploy of endpoint for inferences.

1.3 Created a bucket named "udacitiesolution-alela" and changed the notebook to use that bucket. Run the train_and_deploy-solution.ipynb first cells referred about data collection, unzipping and synchronization with s3 and the cells created the images folders, and images into the bucket.

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with 'Amazon S3' selected. Under 'Buckets', it lists 'udacitiesolution-alela'. Below that, there are sections for 'Access Points', 'Object Lambda Access Points', 'Multi-Region Access Points', 'Batch Operations', and 'Access analyzer for S3'. Further down are sections for 'Block Public Access settings for this account', 'Storage Lens', 'Dashboards', and 'AWS Organizations settings'. At the bottom of the sidebar are 'Feedback' and 'English (US)'. The main area shows the 'udacitiesolution-alela' bucket details. The 'Info' tab is selected. Below it, tabs for 'Objects', 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points' are visible. The 'Objects' tab is selected. It displays 'Objects (3)'. A table lists three objects: 'test/' (Folder), 'train/' (Folder), and 'valid/' (Folder). The table columns are Name, Type, Last modified, Size, and Storage class. At the top of the object list are buttons for 'Create folder' and 'Upload'. Below the table is a search bar with 'Find objects by prefix' and navigation arrows. The footer of the page includes links for 'Feedback', 'English (US)', '© 2022, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

1.4 Run the following cells of the notebook to perform Hyperparameter optimization. This computation takes some time, depending on the instance type you choose to run computation.

I reserved the values of the optimization to perform the training of the model. At this point using the smdebug module, we perform debugging of the model to avoid the following problems that can show up in any training: overfitting, vanishing gradients, poor weight initialization or overtraining. Once the model is trained this way, we create another identical model but with multi instance. The multi instance training resulted in: Training seconds: 4221 whereas the single instance just:

Training seconds: 1339

Name	Creation time	Duration	Status
pytorch-training-220119-1731-002-de29cfdb	Jan 19, 2022 17:32 UTC	-	InProgress
Pytorch-training-220119-1731-001-0be8fe49	Jan 19, 2022 17:31 UTC	-	InProgress

This are the details of each training instance:

Job name	Status	SageMaker metrics time series	IAM role ARN
pytorch-training-220119-1731-001-0be8fe49	InProgress - Starting View history	Disabled	arn:aws:iam::732055107639:role/service-role/AmazonSageMaker-ExecutionRole-20220119T111125

And this is the cell code of the notebook that determined the two training instances to accelerate computation:

```
In [7]: estimator = PyTorch(
    entry_point="hpo.py",
    base_job_name='pytorch_dog_hpo',
    role=role,
    framework_version="1.4.0",
    instance_count=1,
    instance_type="ml.m5.xlarge",
    py_version="py3"
)

tuner = HyperparameterTuner(
    estimator,
    objective_metric_name,
    hyperparameter_ranges,
    metric_definitions,
    max_jobs=2,
    max_parallel_jobs=2,
    objective_type=objective_type
)
```

And the tuning job can also be seen from here:

Name	Status	Training completed/total	Creation time	Duration
Pytorch-training-220119-1740	Completed	2 / 2	Jan 19, 2022 17:40 UTC	30 minutes
Pytorch-training-220119-1731	Failed	0 / 2 2 Failed	Jan 19, 2022 17:31 UTC	6 minutes

Finally, when those jobs completed execution, we have the following result on the cell:

```
In [23]: hyperparameters = {"batch_size": int(best_estimator.hyperparameters()['batch_size'].replace('"', '')), \
                           "learning_rate": best_estimator.hyperparameters()['learning_rate']}
hyperparameters

Out[23]: {'batch_size': 128, 'learning_rate': '0.012257979272044682'}
```

We can keep this values to use them later for the training-debugging of the model with optimal parameters.

1.5 We deployed two endpoints for inference in both single instance and multi instance and performed the prediction for the same data and we obtained different results as well as different inference times. We kept the logs of both invocations to see if there is some sensitive difference but inference times were similar. We should instead perform a lot of requests to see how the endpoints latency behaves in case of higher throughput. The code for the single instance:

Creating an Estimator

```
In [29]: #adjust this cell to accomplish multi-instance training
estimator = PyTorch(
    entry_point='hpo.py',
    base_job_name='dog-pytorch',
    role=role,
    instance_count=1,
    instance_type='ml.m5.xlarge',
    framework_version='1.4.0',
    py_version='py3',
    hyperparameters=hyperparameters,
    ## Debugger and Profiler parameters
    rules = rules,
    debugger_hook_config=hook_config,
    profiler_config=profiler_config,
)

In [30]: estimator.fit({'training': "s3://udacitiesolution-alela/"}, wait=False)
```

The training for the single instance estimator produced by that code:

Name	Creation time	Duration	Status
dog-pytorch-2022-01-19-18-50-59-190	Jan 19, 2022 18:50 UTC	-	InProgress
dog-pytorch-2022-01-19-18-22-106	Jan 19, 2022 18:18 UTC	24 minutes	Completed
dog-pytorch-2022-01-19-18-04-448	Jan 19, 2022 18:18 UTC	25 minutes	Completed
pytorch-training-220119-1740-002-45cf5b8b	Jan 19, 2022 17:41 UTC	23 minutes	Completed
pytorch-training-220119-1740-001-e02276c2	Jan 19, 2022 17:41 UTC	26 minutes	Completed
pytorch-training-220119-1731-002-de29cfdb	Jan 19, 2022 17:32 UTC	5 minutes	Failed
pytorch-training-220119-1731-001-0be8fe49	Jan 19, 2022 17:31 UTC	5 minutes	Failed

Process jobs completed to avoid overfitting, poor weight initialization, overtraining and vanishing gradients:

Name	ARN	Creation time	Duration	Status
dog-pytorch-2022-01-19-18--ProfilerReport-Of5deb92	arn:aws:sagemaker:us-east-1:732055107639:processing-job/dog-pytorch-2022-01-19-18--profilerreport-Of5deb92	Jan 19, 2022 18:51 UTC	2 minutes	Completed
dog-pytorch-2022-01-19-18--PoorWeightInitialization-668eca02	arn:aws:sagemaker:us-east-1:732055107639:processing-job/dog-pytorch-2022-01-19-18--poorweightinitialization-668eca02	Jan 19, 2022 18:51 UTC	2 minutes	Completed
dog-pytorch-2022-01-19-18--Overtraining-284d9b93	arn:aws:sagemaker:us-east-1:732055107639:processing-job/dog-pytorch-2022-01-19-18--overtraining-284d9b93	Jan 19, 2022 18:51 UTC	2 minutes	Completed
dog-pytorch-2022-01-19-18--Overfit-d1db968	arn:aws:sagemaker:us-east-1:732055107639:processing-job/dog-pytorch-2022-01-19-18--overfit-d1db968	Jan 19, 2022 18:51 UTC	2 minutes	Completed
dog-pytorch-2022-01-19-18--VanishingGradient-e35814a0	arn:aws:sagemaker:us-east-1:732055107639:processing-job/dog-pytorch-2022-01-19-18--vanishinggradient-e35814a0	Jan 19, 2022 18:51 UTC	2 minutes	Completed
dog-pytorch-2022-01-19-18--PoorWeightInitialization-9f4c3f00	arn:aws:sagemaker:us-east-1:732055107639:processing-job/dog-pytorch-2022-01-19-18--poorweightinitialization-9f4c3f00	Jan 19, 2022 18:18 UTC	13 minutes	Completed
dog-pytorch-2022-01-19-18--Overtraining-bfb8f6c6	arn:aws:sagemaker:us-east-1:732055107639:processing-job/dog-pytorch-2022-01-19-18--overtraining-bfb8f6c6	Jan 19, 2022 18:18 UTC	24 minutes	Completed
dog-pytorch-2022-01-19-18--Overfit-5de69a3	arn:aws:sagemaker:us-east-1:732055107639:processing-job/dog-pytorch-2022-01-19-18--overfit-5de69a3	Jan 19, 2022 18:18 UTC	24 minutes	Completed
dog-pytorch-2022-01-19-18--VanishingGradient-d94e9436	arn:aws:sagemaker:us-east-1:732055107639:processing-job/dog-pytorch-2022-01-19-18--vanishinggradient-d94e9436	Jan 19, 2022 18:18 UTC	24 minutes	Completed
dog-pytorch-2022-01-19-18--ProfilerReport-1ecefb6b	arn:aws:sagemaker:us-east-1:732055107639:processing-job/dog-pytorch-2022-01-19-18--profilerreport-1ecefb6b	Jan 19, 2022 18:18 UTC	24 minutes	Completed

**1.6 Analogously, we performed same computation for the multi-instance model:
The code for the multi instance:**

Creating an Estimator - Multi-Instance Training,

```
In [37]: ###in this cell, create and fit an estimator using multi-instance training
estimator_multi_instance = PyTorch(
    entry_point='hpo.py',
    base_job_name='dog-pytorch',
    role=role,
    instance_count=3,
    instance_type='ml.m5.xlarge',
    framework_version='1.4.0',
    py_version='py3',
    hyperparameters=hyperparameters,
    # Debugger and Profiler parameters
    rules = rules,
    debugger_hook_config=hook_config,
    profiler_config=profiler_config,
)

In [38]: estimator_multi_instance.fit({"training": "s3://udacitysolution-alela/"}, wait=False)
```

The training for the multi instance estimator produced by that code:

Name	Creation time	Duration	Status
dog-pytorch-2022-01-19-18-50-59-190	Jan 19, 2022 18:50 UTC	25 minutes	Completed
dog-pytorch-2022-01-19-18-18-22-106	Jan 19, 2022 18:18 UTC	24 minutes	Completed
dog-pytorch-2022-01-19-18-04-448	Jan 19, 2022 18:18 UTC	25 minutes	Completed
pytorch-training-220119-1740-002-45cf5b8b	Jan 19, 2022 17:41 UTC	23 minutes	Completed
pytorch-training-220119-1740-001-e02276c2	Jan 19, 2022 17:41 UTC	26 minutes	Completed
pytorch-training-220119-1731-002-de29cfdb	Jan 19, 2022 17:32 UTC	5 minutes	Failed
pytorch-training-220119-1731-001-0be8fe49	Jan 19, 2022 17:31 UTC	5 minutes	Failed

Process jobs completed to avoid overfitting, poor weight initialization, overtraining and vanishing gradients:

Name	ARN	Creation time	Duration	Status
dog-pytorch-2022-01-19-18-Overfitting-284d9b93	arn:aws:sagemaker:us-east-1:732055107639:processing-job/dog-pytorch-2022-01-19-18-overfitting-284d9b93	Jan 19, 2022 18:51 UTC	24 minutes	Completed
dog-pytorch-2022-01-19-18-PoorWeightInitialization-868eca02	arn:aws:sagemaker:us-east-1:732055107639:processing-job/dog-pytorch-2022-01-19-18-poorweightinitialization-868eca02	Jan 19, 2022 18:51 UTC	14 minutes	Completed
dog-pytorch-2022-01-19-18-Overtraining-284d9b93	arn:aws:sagemaker:us-east-1:732055107639:processing-job/dog-pytorch-2022-01-19-18-overtraining-284d9b93	Jan 19, 2022 18:51 UTC	9 minutes	Failed
dog-pytorch-2022-01-19-18-Overfitting-d1db968d	arn:aws:sagemaker:us-east-1:732055107639:processing-job/dog-pytorch-2022-01-19-18-overfitting-d1db968d	Jan 19, 2022 18:51 UTC	25 minutes	Completed
dog-pytorch-2022-01-19-18-VanishingGradient-e55814a0	arn:aws:sagemaker:us-east-1:732055107639:processing-job/dog-pytorch-2022-01-19-18-vanishinggradient-e55814a0	Jan 19, 2022 18:51 UTC	10 minutes	Failed
dog-pytorch-2022-01-19-18-PoorWeightInitialization-9f4c3f00	arn:aws:sagemaker:us-east-1:732055107639:processing-job/dog-pytorch-2022-01-19-18-poorweightinitialization-9f4c3f00	Jan 19, 2022 18:18 UTC	13 minutes	Completed
dog-pytorch-2022-01-19-18-Overtraining-bfb8fc6c	arn:aws:sagemaker:us-east-1:732055107639:processing-job/dog-pytorch-2022-01-19-18-overtraining-bfb8fc6c	Jan 19, 2022 18:18 UTC	24 minutes	Completed
dog-pytorch-2022-01-19-18-Overfitting-5de869a3	arn:aws:sagemaker:us-east-1:732055107639:processing-job/dog-pytorch-2022-01-19-18-overfitting-5de869a3	Jan 19, 2022 18:18 UTC	24 minutes	Completed
dog-pytorch-2022-01-19-18-VanishingGradient-d94e9436	arn:aws:sagemaker:us-east-1:732055107639:processing-job/dog-pytorch-2022-01-19-18-vanishinggradient-d94e9436	Jan 19, 2022 18:18 UTC	24 minutes	Completed
dog-pytorch-2022-01-19-18-ProfilerReport-1ecafbf6	arn:aws:sagemaker:us-east-1:732055107639:processing-job/dog-pytorch-2022-01-19-18-profilerreport-1ecafbf6	Jan 19, 2022 18:18 UTC	24 minutes	Completed

1.7 After creating this two endpoints, the final version of the notebook is the one in this repo and we deleted the endpoints and stop the notebook instance to avoid charges.

Name	ARN	Creation time	Status	Last updated
pytorch-inference-2022-01-19-21-39-12-151	arn:aws:sagemaker:us-east-1:732055107639:endpoint/pytorch-inference-2022-01-19-21-39-12-151	Jan 19, 2022 21:39 UTC	Creating	Jan 19, 2022 21:39 UTC
pytorch-inference-2022-01-19-20-54-22-720	arn:aws:sagemaker:us-east-1:732055107639:endpoint/pytorch-inference-2022-01-19-20-54-22-720	Jan 19, 2022 20:54 UTC	InService	Jan 19, 2022 20:56 UTC

Step 2: Perform a similar task on an EC2 instance.

EC2 Instance setup

2.1 We choose first to launch a t2.micro since it's free tier to avoid costs, if I consider it insufficient, I will retry with a larger instance. Anyway the load is not in the EC2, as it was not on the notebook in sagemaker, but in the jobs launched for hpo and training.

To compare: In the sagemaker task we used ml.t2.medium for the notebook (very light work) and two ml.m5.xlarge for the trainings and ml.m5.large for inferences. The total costs of performing the tasks with sagemaker were \$4.03 The total costs of EC2 using same combination of resources were much less than that, but the jobs were different too.

The screenshot shows the AWS Cost Explorer interface. On the left, there's a sidebar with various navigation links like Payments, Credits, Purchase orders, Cost & Usage Reports, Cost Categories, Cost allocation tags, Cost Management, Cost Explorer, Budgets, Budgets Reports, Savings Plans, Preferences, Billing preferences, Payment methods, Consolidated billing, and Tax settings. The main content area is titled "Summary" and "AWS Service Charges". It lists charges for CloudWatch (US East (N. Virginia)), Data Transfer, Elastic File System, Key Management Service, Lambda, SageMaker, Simple Queue Service, and Simple Storage Service. The total AWS Service Charges listed are \$4.70. The currency is USD. A note at the bottom states: "Usage and recurring charges for this statement period will be charged on your next billing date. Estimated charges shown on this page, or shown on any notifications that we send to you, may differ from your actual charges for this statement period. This is because estimated charges presented on this page do not include usage charges incurred during this statement period after the date you view this page. Information about estimated charges sent to you in a notification do not include usage charges incurred during this statement period after the date we send you the notification. One-time fees and subscription charges are assessed separately from usage and recurring charges, on the date that they occur." At the bottom right, there are links for Feedback, English (US), © 2022, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

I choose the AMI Amazon Deep Learning because it comes with ML learning environment integrated already.

The screenshot shows the "Step 1: Choose an Amazon Machine Image (AMI)" screen of the New Launch Instance Wizard. The top navigation bar includes links for Choose AMI, Choose Instance Type, Configure Instance, Add Storage, Add Tags, Configure Security Group, and Review. A message box says: "You've been invited to try an early, beta iteration of the new launch instance wizard. We will continue to improve the experience over the next few months. We're asking customers for their feedback on this early release. To exit the new launch instance wizard at any time, choose the Cancel button." Below this, there are three options under "Amazon Linux": "Deep Learning AMI (Ubuntu 18.04) Version 55.0 - ami-029536273cb04d4d9" (selected), "Deep Learning AMI GPU PyTorch 1.10 (Amazon Linux 2) 20211115 - ami-01a16356ed2a310d1", and "Deep Learning AMI (Amazon Linux 2) Version 57.0 - ami-06ada98f5df02a2d2d". Each option has a "Select" button and a note about 64-bit (x86) architecture. The sidebar on the left shows categories: My AMIs (0), AWS Marketplace (161), Community AMIs (697), and a checkbox for Free tier only. At the bottom, there are links for Feedback, English (US), © 2022, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

And finally launched the instance:

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with various navigation options like New EC2 Experience, EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances (New), Dedicated Hosts, Scheduled Instances, Capacity Reservations, and Images (AMIs New, AMI Catalog). The main area is titled 'Instances (1/1)'. It shows a table with one row for the instance 'i-099ec51280ed14886'. The columns include Name, Instance ID, Instance state, Instance type, Status check, and Alarm status. The instance is listed as 'Running' (t2.micro). Below the table, there's a detailed view for 'Instance: i-099ec51280ed14886' with tabs for Details, Security, Networking, Storage, Status checks, Monitoring, and Tags. Under the Details tab, it shows Instance ID (i-099ec51280ed14886), Public IPv4 address (54.86.143.113), Private IPv4 addresses (172.31.80.215), Public IPv6 DNS (ec2-54-86-143-113.compute-1.amazonaws.com), and Instance state (Running).

Later on, it turned out that for amazon deep learning free tier is not available for the Amazon Deep Learning AMI and when installing torch by doing: pip install torch there was a memory problem. For that reason I stop the instance and I re launched a ml.t3.medium and connected to this new instance with the same information I had in the t2.micro in a matter of seconds.

2.2 I created the dir TrainedModels and downloaded and unzipped there the file: <https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip> using wget and unzip commands

```
=====
|   |   |_) Deep Learning AMI GPU PyTorch 1.10.0 (Amazon Linux 2)
|_ \ \ \ \ \
=====
* To activate pre-built pytorch environment, run: 'source activate pytorch'
* To activate base conda environment upon login, run: 'conda config --set auto_activate_base true'
* NVIDIA driver version: 470.57.02
* CUDA version: 11.3

AWS Deep Learning AMI Homepage: https://aws.amazon.com/machine-learning/amis/
Release Notes: https://docs.aws.amazon.com/dlami/latest/devguide/appendix-ami-release-notes.html
Support: https://forums.aws.amazon.com/forum.jspa?forumID=263
For a fully managed experience, check out Amazon SageMaker at https://aws.amazon.com/sagemaker
Security scan reports for python packages are located at: /opt/aws/dlami/info/
=====
No packages needed for security or python packages available
Run sudo dnf update to apply all updates
[root@ip-172-31-88-16 ~] # wget https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip
--2022-01-20 11:43:42 - https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip
Resolving s3-us-west-1.amazonaws.com (s3-us-west-1.amazonaws.com)... 3.5.163.157
Connecting to s3-us-west-1.amazonaws.com (s3-us-west-1.amazonaws.com)|3.5.163.157|:443... connected.
HTTP request sent, awaiting response...
200 OK
Length: 1132023110 (1.1G) [application/zip]
Saving to: 'dogImages.zip'

19% [=====>]
28% [=====>]
30% [=====>]
34% [=====>]
73% [=====>]

i-0702da6c590ca0da5
Public IP: 52.1.79.62 Private IP: 172.31.88.16
```

2.3 Created the file solution.py and I pasted the contents of the script ec2train1.py

```

test_data_loader = torch.utils.data.DataLoader(test_data, batch_size=batch_size, shuffle=True)
validation_data = torchvision.datasets.ImageFolder(root=validation_data_path, transform=test_transform)
validation_data_loader = torch.utils.data.DataLoader(validation_data, batch_size=batch_size, shuffle=True)

return train_data_loader, test_data_loader, validation_data_loader

batch_size=2
learning_rate=1e-4
train_loader, test_loader, validation_loader=create_data_loaders('dogImages',batch_size)
model=net()

criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.fc.parameters(), lr=learning_rate)

logger.info("Starting Model Training")
model.train()
train_loader, validation_loader, criterion, optimizer
torch.save(model.state_dict(), 'TrainedModels/model.pth')
print('saved')

solution.py" 147L, 48428 written
[ec2-user@ip-172-31-80-215 ~]$ wc -l solution.py
147 solution.py
[ec2-user@ip-172-31-80-215 ~]$ ls
logImages dogImages.zip solution.py TrainedModels
[ec2-user@ip-172-31-80-215 ~]$ ls dogImages
test train valid
[ec2-user@ip-172-31-80-215 ~]$ ls dogImages/test/ | tail -n 4
30_West_highland_pony
31_Wirehaired_pointing_griffon
32_Xoloitzcuintli
33_Yorkshire_terrrier
[ec2-user@ip-172-31-80-215 ~]$ █

```

i-099ec51280ed14886

Public IPs: 54.86.143.113 Private IPs: 172.31.80.215

2.4 Run the solution.py and took a screenshot of the model into the TrainedModels directory After inspecting the code in solution.py I can see that it performs the same tasks that were performed in the notebook of step 1 (train_and_deploy-solution.ipynb). It was adapted to work in a typical linux distro but with some changes as follows:

```

with tokenize.open(f'{filename}') as f:
    for line in f:
        if line.startswith('#'):
            continue
        buffer.append(line)
        encoding = detect_encoding(buffer)
        readline = detect_encoding(buffer, readline)
        first = read_or_stop()
        File "/opt/conda/lib/python3.9/tokenize.py", line 363, in detect_encoding
            first = read_or_stop()
        File "/opt/conda/lib/python3.9/tokenize.py", line 321, in read_or_stop
            return readline()
KeyboardInterrupt

[root@ip-172-31-88-16 ~]# ^C
[root@ip-172-31-88-16 ~]# python solution.py
Traceback (most recent call last):
  File "/root/solution.py", line 5, in <module>
    import torchvision
ModuleNotFoundError: No module named 'torchvision'
[root@ip-172-31-88-16 ~]# pip install torchvision
Collecting torchvision
  Downloading torchvision-0.11.2-cp39-cp39-manylinux1_x86_64.whl (23.2 MB)
[...]
Collecting pillow==8.3.0,>=5.3.0
  Downloading Pillow-9.0.0-cp39-cp39-manylinux_2_17_x86_64_manylinux2014_x86_64.whl (4.3 MB)
[...]
Requirement already satisfied: numpy in /opt/conda/lib/python3.9/site-packages (from torchvision) (1.22.1)
Requirement already satisfied: torch<=1.10.1 in /opt/conda/lib/python3.9/site-packages (from torchvision) (1.10.1)
Requirement already satisfied: typing_extensions in /opt/conda/lib/python3.9/site-packages (from torch==1.10.1->torchvision) (4.0.1)
Installing collected packages: torch, torchvision
Successfully installed pillow-9.0.0 torchvision-0.11.2
WARNING: Running pip as root will break packages and permissions. You should install packages reliably by using venv: https://pip.pypa.io/warnings/venv
[root@ip-172-31-88-16 ~]# python solution.py
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, 345MB/s]
Starting Model Training

```

i-0702da6c590ca0da5

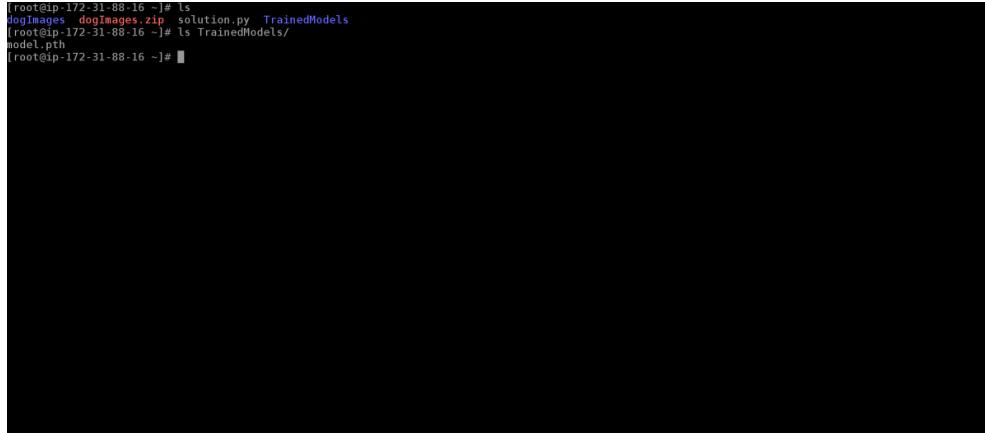
Public IPs: 44.201.181.170 Private IPs: 172.31.88.16

Considerations about the code into the solution.py file:

The code resembles the one used in hpo.py, but it has no smdebug module to perform the final debugging, so the result will be less effective. As well, hyperparameters are fixed, so there is no hyperparameter optimization. Also, this code does not perform the deploy of the endpoint. All that will have to be worked later.

2.5 Finally the job ended as follows: Execution time start 6:57 7:24 ended. And the proof of the job run well is the model saved into the directory:

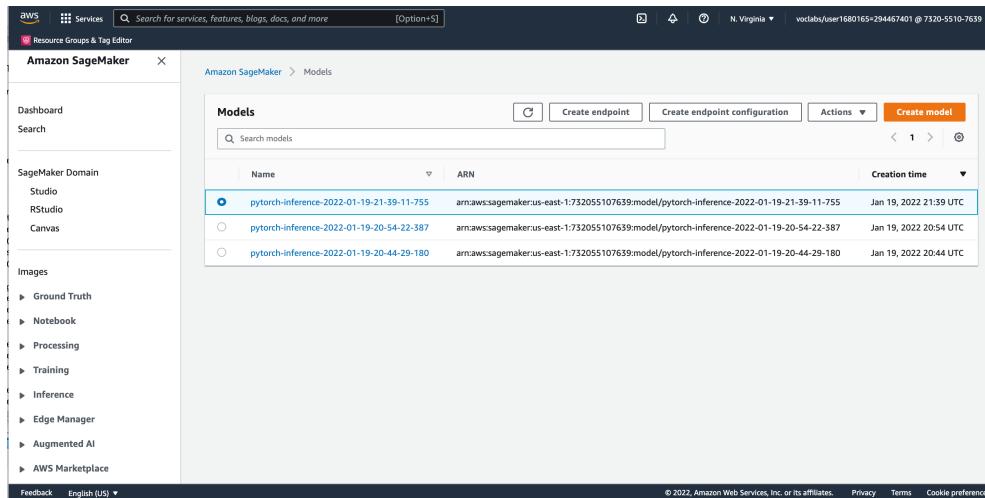
```
root@ip-172-31-88-16:~# ls
deployImages  dogImages.zip  solution.py  TrainedModels
root@ip-172-31-88-16:~# ls TrainedModels/
model.pth
[root@ip-172-31-88-16 ~]#
```



i-0702da6c590ca0da5
Public IPs: 44.201.181.170 Private IPs: 172.31.88.16

Step 3: Create a Lambda function that will consume your model inference capabilites via endpoints.

3.1 For this task I had to re create the endpoint I deleted yeasterday. I have the models for the endpoint configuration created for both multi-instance and single-instance I went to models in SageMaker and all the models created were there. I decided to use the multi-instance one:



Name	ARN	Creation time
pytorch-inference-2022-01-19-21-39-11-755	arn:aws:sagemaker:us-east-1:732055107639:model/pytorch-inference-2022-01-19-21-39-11-755	Jan 19, 2022 21:39 UTC
pytorch-inference-2022-01-19-20-54-22-387	arn:aws:sagemaker:us-east-1:732055107639:model/pytorch-inference-2022-01-19-20-54-22-387	Jan 19, 2022 20:54 UTC
pytorch-inference-2022-01-19-20-44-29-180	arn:aws:sagemaker:us-east-1:732055107639:model/pytorch-inference-2022-01-19-20-44-29-180	Jan 19, 2022 20:44 UTC

3.2 Then I went to endpoints on SageMaker and created the endpoint using the multi-instance model shown above and I choose a new name for the endpoint.

Name	ARN	Creation time	Status	Last updated
endpoint-multi-instance	arn:aws:sagemaker:us-east-1:732055107639:endpoint/endpoint-multi-instance	Jan 20, 2022 12:39 UTC	InService	Jan 20, 2022 12:42 UTC

Step 4: Security and testing.

4.1 I tested the lambda function using the following test: {"url":

"<https://s3.amazonaws.com/cdn-origin-etr.akc.org/wp-content/uploads/2017/11/20113314/Carolina-Dog-standing-outdoors.jpg>" } But I run into an error when testing.

```

{
    "errorMessage": "An error occurred (AccessDeniedException) when calling the InvokeEndpoint operation: User: arn:aws:sts::732055107639:assumed-role/lambda-project-operatorLambdaRole;Action: invoke;Resource: arn:aws:lambda:us-east-1:732055107639:function:lambda-project-operatorLambdaFunction;RequestId: 22a389fd-9ee1-4746-9d78-e4032ed334e;StatusCode: 403;Timestamp: 2022-01-20T12:44:44Z;Type: ClientError"
}

```

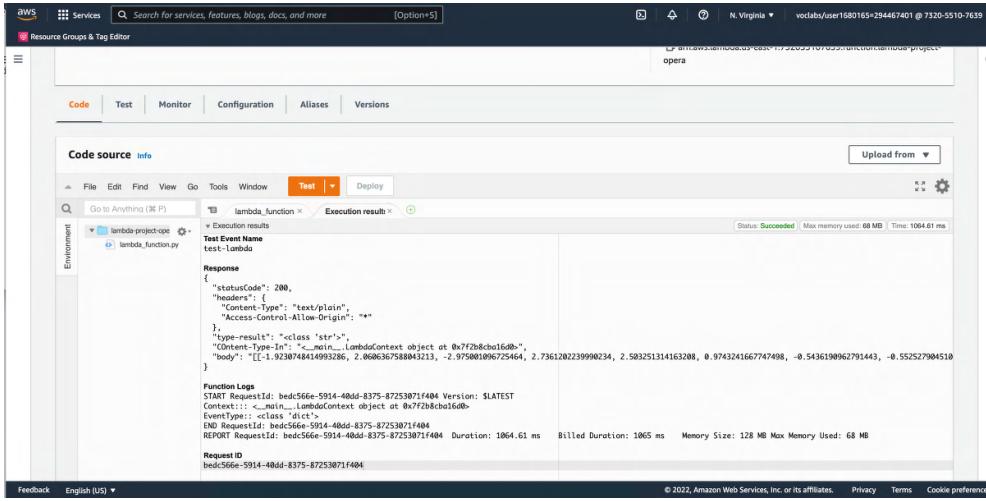
This error was caused because of Lambda running on a role that has no permissions over SageMaker nor Over s3.

The screenshot shows the AWS Lambda console. A function named 'lambda-project-opera' is selected. The 'Configuration' tab is active. In the 'Execution role' section, the role name is 'lambda-project-opera-role-3fgj8z1'. The sidebar on the left has 'Permissions' selected. Other tabs like 'Code', 'Test', 'Monitor', and 'Aliases' are also visible.

To solve that we added the policies for sagemakerfullaccess and s3fullaccess to the execution role.

The screenshot shows the AWS IAM 'Roles' page. It displays the 'Summary' for the role 'lambda-project-opera-role-3fgj8z1'. Under the 'Permissions' tab, three policies are listed: 'AWSLambdaBasicExecutionRole', 'AmazonS3FullAccess', and 'AmazonSageMakerFullAccess'. The sidebar on the left has 'Roles' selected.

4.2 I rerun the test and now it worked!



The screenshot shows the AWS Lambda Test interface. At the top, there's a navigation bar with tabs for Code, Test, Monitor, Configuration, Aliases, and Versions. The Test tab is selected. Below the navigation bar, there's a toolbar with File, Edit, Find, View, Go, Tools, Window, a Test button, Deploy, and a dropdown for "Upload from". To the right of the toolbar is a status bar showing "Status: Succeeded | Max memory used: 68 MB | Time: 1064.61 ms". The main area has tabs for Environment, Code source, and Info. The Code source tab is active, displaying a Python file named "lambda_function.py". The file contains a single function definition:

```
def lambda_handler(event, context):  
    response = {  
        "statusCode": 200,  
        "headers": {},  
        "Content-Type": "text/plain",  
        "Access-Control-Allow-Origin": "*"  
    }  
    type(result) == <class 'str'>  
    "Content-Type-In": <__main__.LambdaContext object at 0x7f72b8cb16d0>,  
    "body": "[[-1.9230748414993286, 2.0606367588043213, -2.975001096725464, 2.736120239990234, 2.503251314163208, 0.9743241667747498, -0.5436190962791443, -0.552527904510]
```

Below the code, there's a "Function Log" section with log entries:

```
START RequestId: bedc566e-5914-48dd-8375-87253071f404 Version: $LATEST  
Context::: <__main__.LambdaContext object at 0x7f72b8cb16d0>  
EventTypes::: <class 'dict'>  
END RequestId: bedc566e-5914-48dd-8375-87253071f404  
REPORT RequestId: bedc566e-5914-48dd-8375-87253071f404 Duration: 1064.61 ms Billed Duration: 1065 ms Memory Size: 128 MB Max Memory Used: 68 MB  
Request ID  
bedc566e-5914-48dd-8375-87253071f404
```

Here is the complete response from the endpoint:

Test Event Name test-lambda

```
Response { "statusCode": 200, "headers": { "Content-Type": "text/plain", "Access-Control-Allow-Origin": "*" }, "type-result": "<class 'str'>", "Content-Type-In": "<main.LambdaContext object at 0x7f2b8cba16d0>", "body": "[[-1.9230748414993286, 2.0606367588043213, -2.975001096725464, 2.7361202239990234, 2.503251314163208, 0.9743241667747498, -0.5436190962791443, -0.552527904510498, -6.05665397644043, 1.721237063407898, 2.425633668899536, 1.1596161127090454, 0.23340359330177307, 1.1787495613098145, -1.682868480682373, 0.04093865305185318, -3.0608670711517334, -1.0202009677886963, 0.5772457122802734, 2.3727235794067383, 1.725218415260315, -0.061895087361335754, -0.507676362991333, -3.4325389862060547, -2.3014237880706787, -6.103825092315674, 2.229447841644287, -1.3559074401855469, 0.214759960770607, -0.4876475930213928, 2.2932322025299072, -1.8778115510940552, -2.646745443344116, -0.4442533254623413, -1.4839411973953247, 0.018974244594573975, 0.5177682042121887, -0.4669727385044098, -0.7576069235801697, -1.7498409748077393, 1.264542818069458, 1.4518052339553833, 2.3175251483917236, 0.6133086681365967, 2.315450429916382, -2.0466291904449463, 1.5150392055511475, -0.7387037873268127, 0.36982569098472595, -0.008169978857040405, 1.6424503326416016, -2.1007399559020996, -2.434668779373169, -0.008525431156158447, -4.083022117614746, -0.004357367753982544, -2.7408454418182373, -0.48699548840522766, -3.6401796340942383, -1.6554871797561646, -1.546553373336792, -1.7185020446777344, -2.040588855743408, -4.549625396728516, -2.8638806343078613, -2.800689697265625, 1.9408971071243286, -0.13130821287631989, -1.1525102853775024, -1.2089629173278809, 3.5017807483673096, -2.380239725112915, -1.1759413480758667, -1.4099303483963013, -2.453126907348633, -2.063462018966675, -3.7739031314849854, -1.0499930381774902, 1.1904747486114502, -1.380522608757019, 1.0462281703948975, -5.719635009765625, 0.3957575261592865, 1.9719444513320923, -4.87777853012085, -5.302970886230469, -0.5300191640853882, -3.840769052505493, -2.534928798675537, 1.4356107711791992, -3.2125887870788574, 0.5481349229812622, -3.9147355556488037, -1.0964760780334473, 1.0998213291168213, 0.7646914124488831, -1.695220708847046, -1.466501235961914, -3.896580696105957, -4.843873023986816, -9.731215476989746, -2.6416592597961426, 1.6428296566009521, -2.0402002334594727, -0.4763360619544983, -0.48405128717422485, -2.3959767818450928, 1.7910501956939697, 3.1037676334381104, 1.1668872833251953, -0.15623413026332855,
```

```
0.057766884565353394, -3.2649059295654297, -1.6079130172729492,
-3.212416410446167, 0.8164639472961426, -0.5811924934387207,
2.154860258102417, -2.4100425243377686, 0.4797557294368744,
-0.26428431272506714, -1.8739787340164185, -0.9146941304206848,
0.05285979062318802, -6.775721073150635, 0.4937722682952881,
-3.6679766178131104, 1.0705149173736572, -0.2790021300315857,
-3.6403069496154785, -7.05629825592041, -2.225168466567993,
-5.770076274871826]]}" }
```

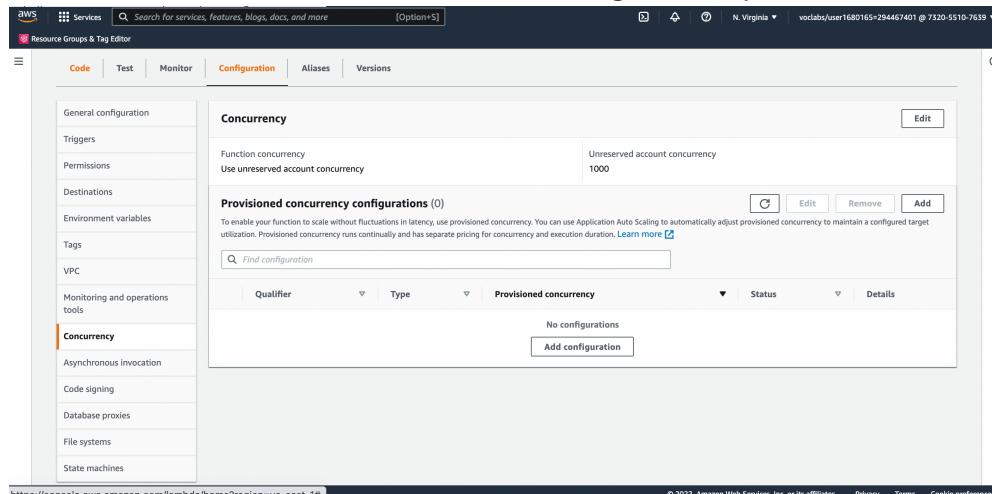
Function Logs START RequestId: bedc566e-5914-40dd-8375-87253071f404
 Version: \$LATEST Context::: <main.LambdaContext object at 0x7f2b8cba16d0>
 EventType::: <class 'dict'> END RequestId: bedc566e-5914-40dd-8375-87253071f404 REPORT RequestId: bedc566e-5914-40dd-8375-87253071f404 Duration: 1064.61 ms Billed Duration: 1065 ms Memory Size: 128 MB Max Memory Used: 68 MB

Request ID bedc566e-5914-40dd-8375-87253071f404

Step 5: Set up concurrence for your lambda function and auto-scaling for your deployed endpoint.

Concurrency

5.1 To let the lamdba answer requests in a parallel fashion we added concurrency from the configuration of the lambda function. The concurrency setup can be located in left menu of the lambda configuration pane.



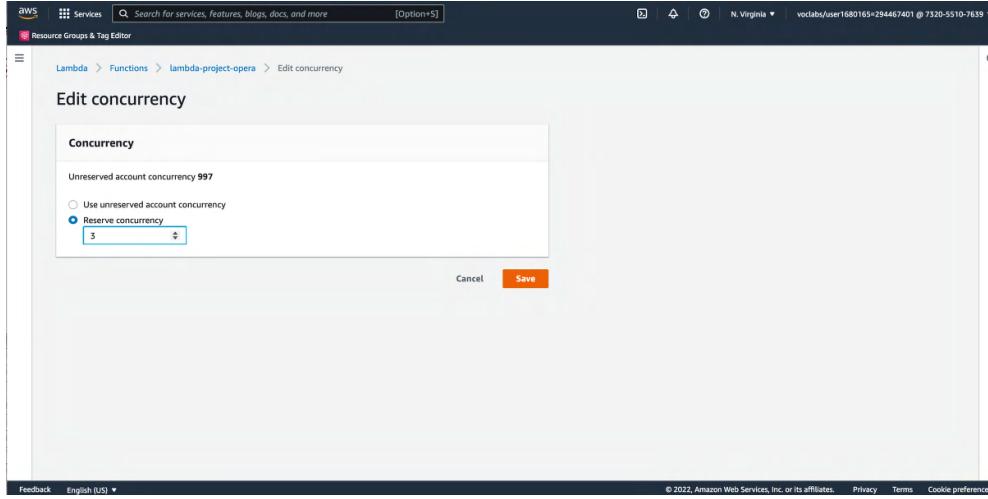
5.2 First we must set up the version1 of the function and using the edit button in Concurrency pane we selected reserved concurrency of 3 (to avoid costs while the endpoint is deployed).

The screenshot shows the AWS Lambda console interface. At the top, there's a search bar and navigation links for services, features, blogs, docs, and more. Below the search bar, it says 'N. Virginia' and 'vocabs/user1680165+294467401 @ 7320-5510-7639'. The main area is titled 'lambda-project-opera' and has a 'Function overview' section. It shows a thumbnail for the function, a 'Layers' section (0 layers), and buttons for '+ Add trigger' and '+ Add destination'. On the right, there's a 'Description' field, 'Last modified' (27 minutes ago), and a 'Function ARN' (arn:aws:lambda:us-east-1:732055107639:function:lambda-project-opera). Below this are tabs for Code, Test, Monitor, Configuration, Aliases, and Versions, with 'Versions' being the active tab. Under 'Versions', it lists Version 1 with the description 'This is a version to increase concurrency', last modified 27 minutes ago, and architecture x86_64. There are 'Delete' and 'Publish new version' buttons.

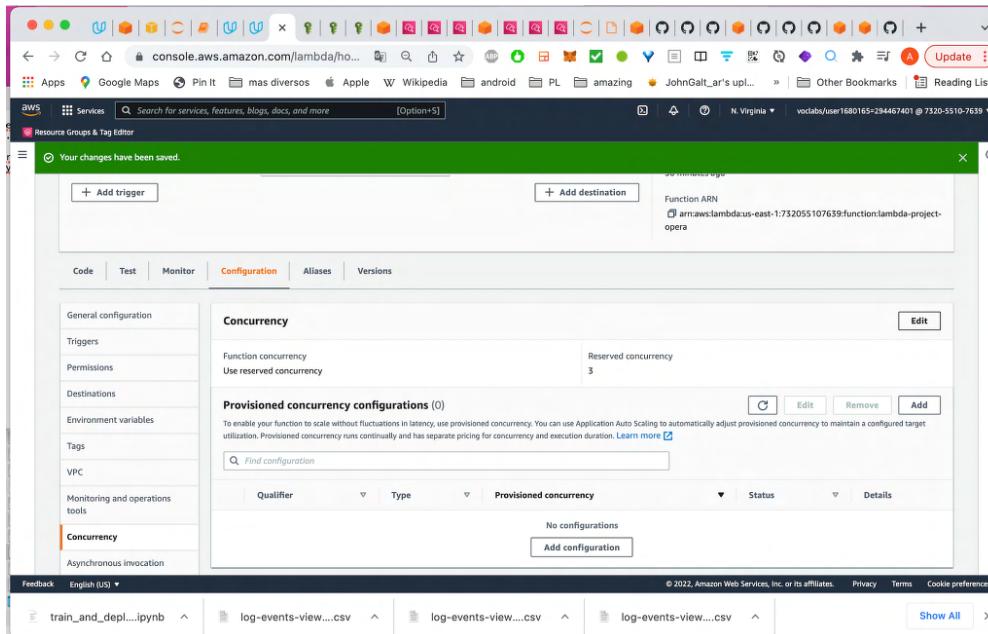
5.3 There are two types of concurrency settings: Reserved concurrency price is low, but latency could be high. Provisioned concurrency is always available and so, more costly.

The screenshot shows the AWS Lambda Configuration page for the same function. The left sidebar has tabs for General configuration, Triggers, Permissions, Destinations, Environment variables, Tags, VPC, Monitoring and operations tools, and Concurrency (which is selected and highlighted with a red border). The main content area is titled 'Concurrency'. It shows 'Function concurrency' (Unreserved account concurrency: 1000) and a 'Provisioned concurrency configurations' section. This section includes a note about enabling scale without fluctuations in latency using provisioned concurrency, a 'Find configuration' search bar, and a table for 'Provisioned concurrency'. The table has columns for Qualifier, Type, Provisioned concurrency, Status, and Details. It currently shows 'No configurations' and has an 'Add configuration' button. Red boxes highlight the 'Edit' button in the 'Unreserved account concurrency' section and the 'Add configuration' button in the 'Provisioned concurrency configurations' section, with red arrows pointing to each from the left.

5.4 I will configure concurrency only for reserved instances:

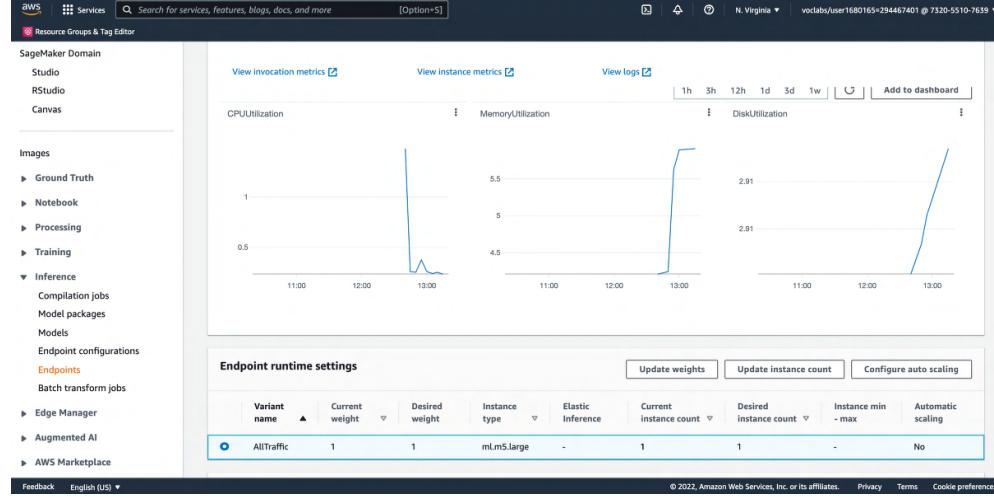


5.5 Finally the reserved concurrency setting is ready.

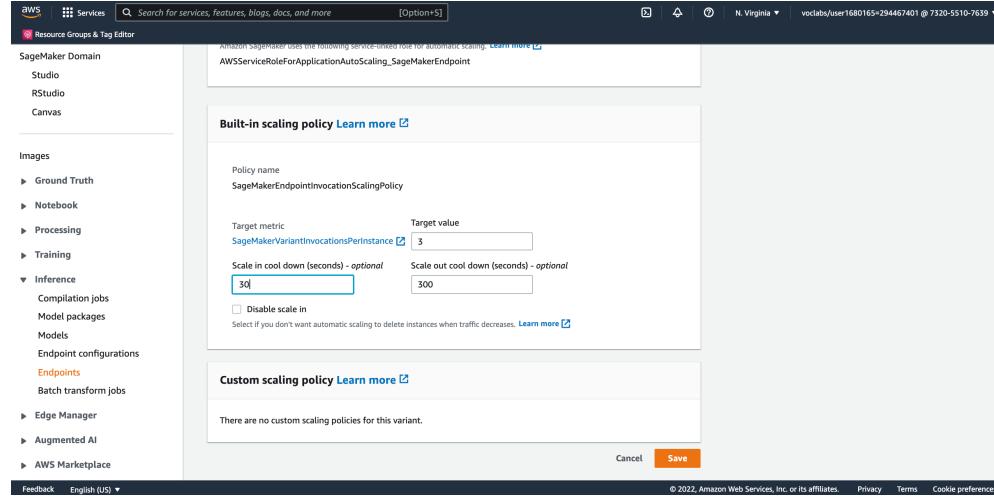


Auto scaling

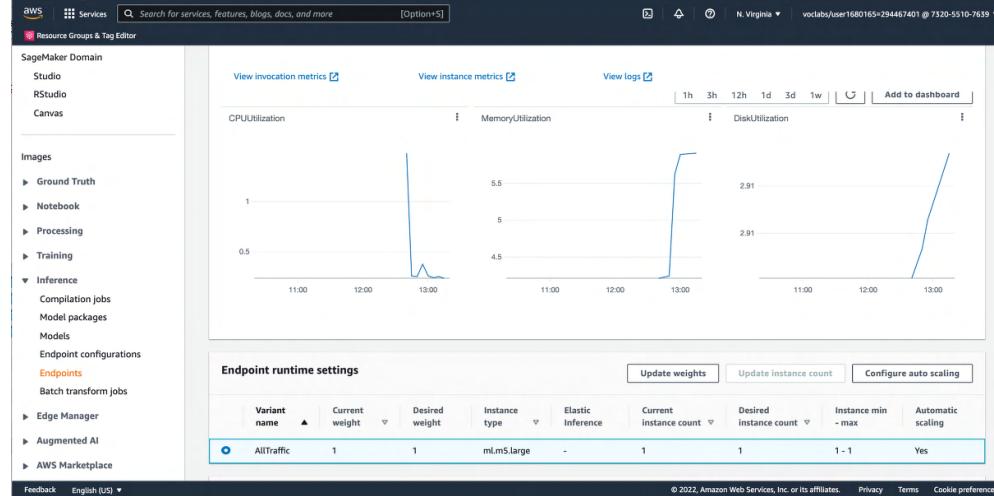
5.6 I will scale our endpoint to scale to more instances and with some short scale in cool down time, and some longer scale out cool down time.



5.7 I choose 3 instances of auto scaling as well as 3 reserved concurrency to be able to deal with triple increase in demand.



5.8 Finally auto-scaling is set!



Final words:

AWS was able to let us quickly create this model and deploy it in a way that is easily scalable and secure as shown in Step 3 and Step 4. This is one of the strongest features of the AWS Sagemaker for quick and professional Machine Learning solutions.