Operationalizing an AWS Machine Learning Project

Project Summary

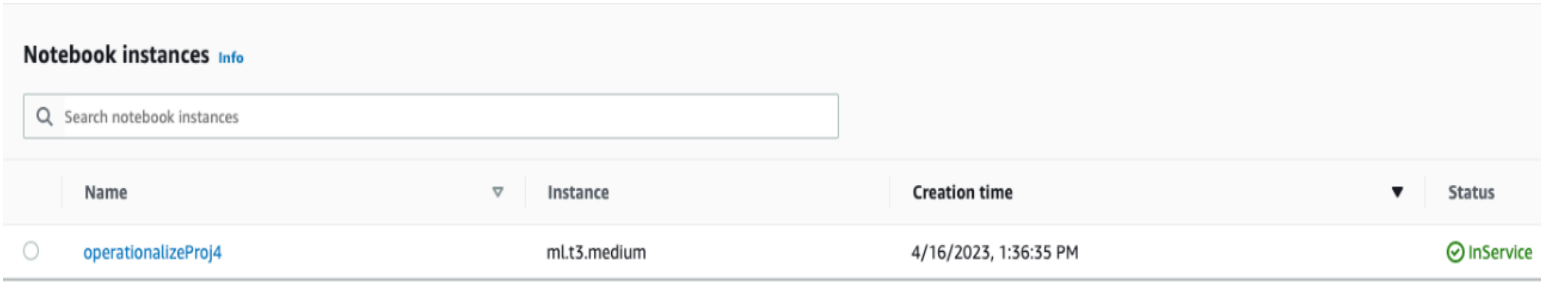
In this project, I completed the following steps:

- Train and deploy a model on Sagemaker, using the most appropriate instances. Set up multi-instance training in a Sagemaker notebook.
- Adjust a Sagemaker notebook to perform training and deployment on EC2.
- Set up a Lambda function for a deployed model. Set up auto-scaling for the deployed endpoint as well as concurrency for a Lambda function.
- Ensure that the security on the ML pipeline was set up properly.

Training and deployment on Sagemaker

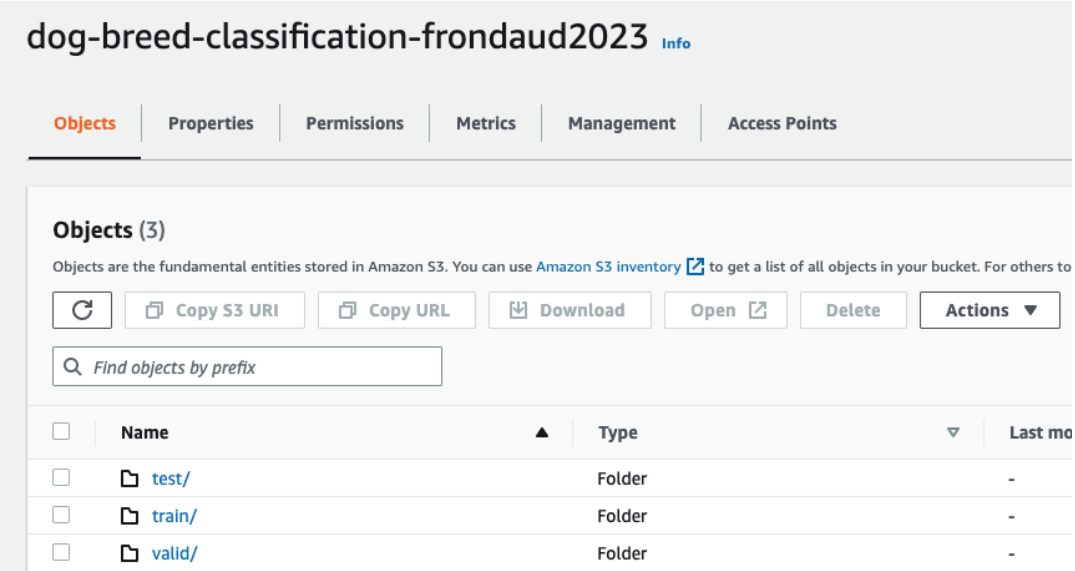
In the first step, I navigated to Sagemaker and started a notebook instance. There are many types of different instances, I chose ml.t3.medium.

- I chose this instance type because the need for CPU and RAM was not great, so I could get away with executing the notebook with no problem.
- Additionally, I took into account cost, as a big factor. I would need to keep the notebook in the “InServce” status as I completed additional steps.



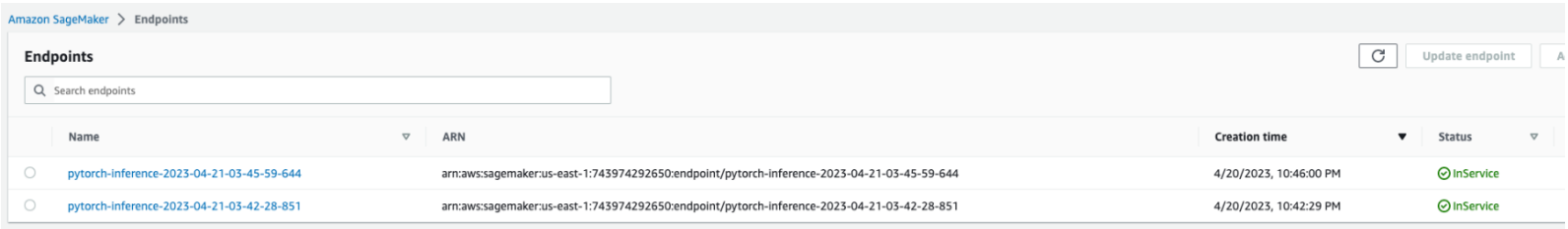
Download data to an S3 bucket

The next step was to create a bucket to hold the dog images for training, testing and validation



Training and Deployment

Finally we can appropriately train the model and deploy endpoints which can be used to make inferences:

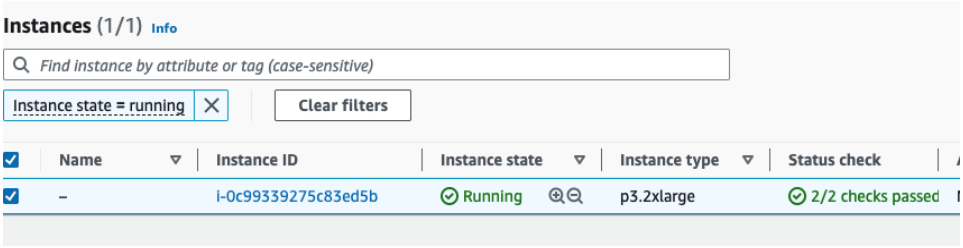


Multi instance training was used to train data more quickly via Sagemaker, which essentially uses multiple instances for gained performance.

EC2 Training

If using a personal AWS account its important to note, vCPU limits. In fact, certain EC2 instances require the a certain limit and the request process can take a few days.

In order to run the **Deep Learning AMI GPU Pytorch** there is a need to run more intensive instances for additional RAM and CPU. According to the documentation, this type of AMI supports only the following instances: G3, P3, P3dn, P4d, G5, and G4dn . These instances also cost more than other alternatives. Thus, in order balance overall power of the instances with the ML workload and cost I choose a **p3.2xlarge** instance, where I had to increase my vCPU limit on account to 8 from 0 for this instance.



Preparing for EC2 model training

Once the instance is ready, I connected to the instance, I downloaded the data and unzipped it, pasted in the training code, trained another model, and saved it to a directory TrainedModels

```
================================================================================
      __|  __|_  )
      _|  (     /   Deep Learning AMI GPU PyTorch 2.0.0 (Amazon Linux 2)
     ___|\___|___|
================================================================================

* Please note that Amazon EC2 P2 Instance is not supported on current DLAMI.
* Supported EC2 instances: G3, P3, P3dn, P4d, P4de, G5, G4dn.
* 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: 525.85.12
* CUDA version: 11.8

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; 10 packages available
Run "sudo yum update" to apply all updates.
[root@ip-172-31-84-165 ~]# ource activate pytorch
-bash: ource: command not found
[root@ip-172-31-84-165 ~]# source activate pytorch
(pytorch) [root@ip-172-31-84-165 ~]# wget https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip
--2023-04-21 01:59:05--  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)... 52.219.193.112
Connecting to s3-us-west-1.amazonaws.com (s3-us-west-1.amazonaws.com)|52.219.193.112|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1132023110 (1.1G) [application/zip]
Saving to: 'dogImages.zip'

90% [=============================================================================================================================
```

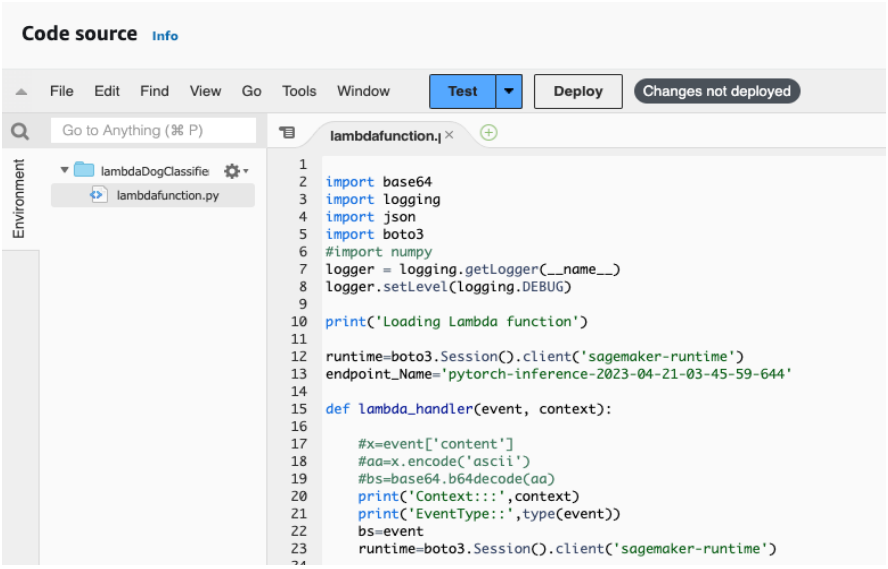
```
(pytorch) [root@ip-172-31-13-82 ~]# vim solution.py
(pytorch) [root@ip-172-31-13-82 ~]# python solution.py
/opt/conda/envs/pytorch/lib/python3.10/site-packages/torchvision/models/_utils.py:208: UserWarning: The parameter
  warnings.warn(
/opt/conda/envs/pytorch/lib/python3.10/site-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other
_Weights.IMAGENET1K_V1`. You can also 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/res
100%|
Starting Model Training
saved
```

```
(pytorch) [root@ip-172-31-13-82 ~]# ls
dogImages  dogImages.zip  solution.py  TrainedModels
(pytorch) [root@ip-172-31-13-82 ~]# cd TrainedModels
(pytorch) [root@ip-172-31-13-82 TrainedModels]# ls
model.pth
(pytorch) [root@ip-172-31-13-82 TrainedModels]#
```

It's important to note the differences between using EC2 and a Notebook Instance for training models. EC2 can be scaled based on computing needs and customized to meet specific requirements such as frameworks, memory size, GPU support, and number of CPUs. A notebook instance has a simple and fast setup time and comes configured with ML frameworks and libraries. In fact a notebook instance represents a more visual and user friendly experience to coding. Something that sets EC2 apart from Sagemaker is that training happens on the EC2 instances where the script is executed, however in the Notebook instances the training job runs separately from the notebook instance, which is flexible but more complex than EC2.

Lambda function setup

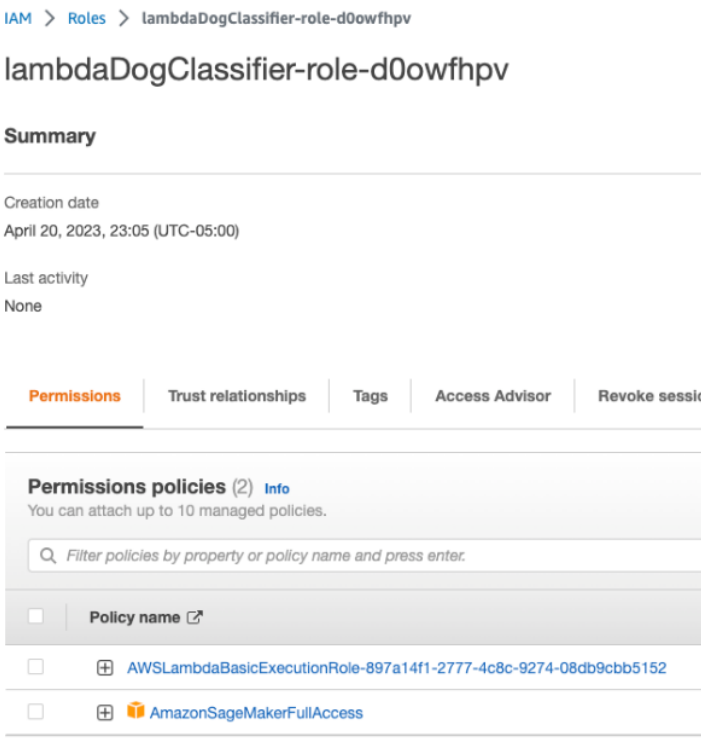
AWS Lambda represents a serverless compute method, which means no instances! Lambda functions enable the model to be accessed by API's and other programs and is nice way to productionize the endpoint. First we go Lambda and create a function:



The implemented function essentially takes an image input in the form of a json which can then be used to trigger the model's deployed Sagemaker endpoint. For this particular function I choose the endpoint created via multi instance training.

Security and testing

Before testing however, it was necessary to change the IAM permissions on my lambda role. By the principle of Least Privilege, which is an important security concept that aims to limit access rights for particular roles with the goal of reducing attack surfaces in case of breaches, the lambda role by default only has access to lambda once created. Thus, it was necessary to add sagemaker access.



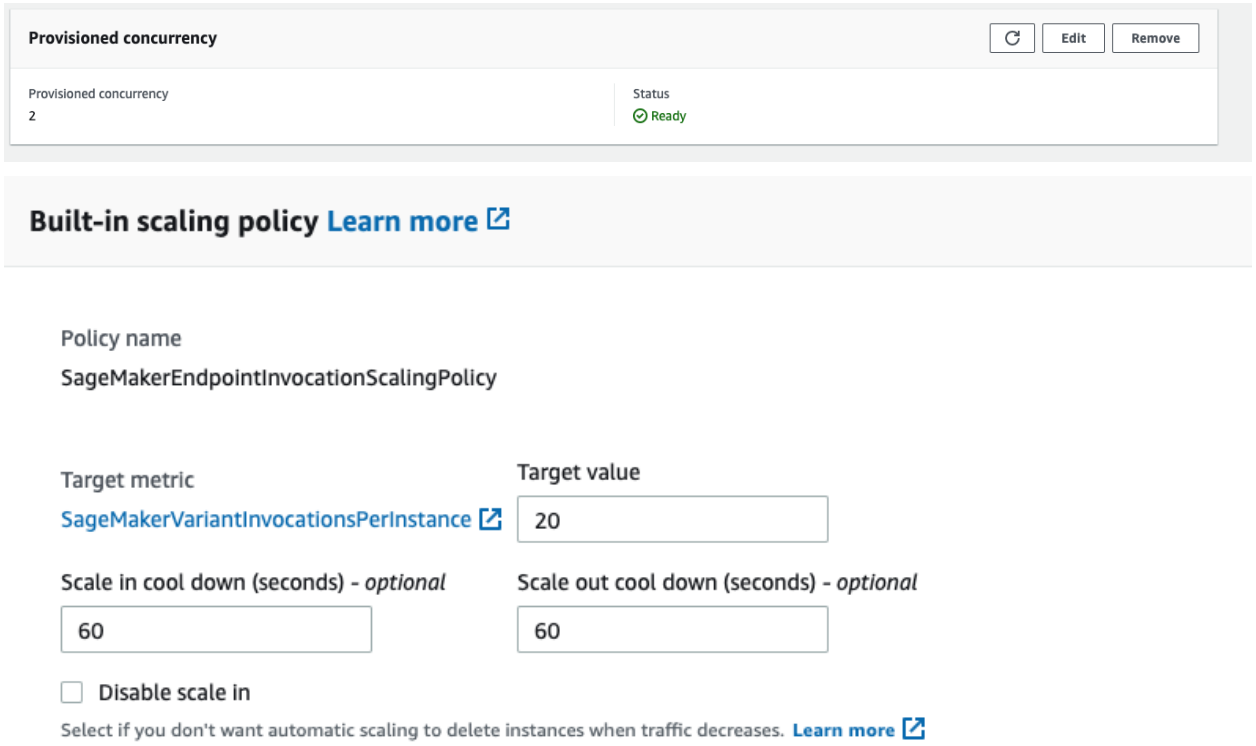
Once this change was made the lambda function could finally test the endpoint

The following output was recorded:

"body": "[[0.2588767409324646, 0.09894143790006638, 0.322409451007843,
-0.007990323007106781, -0.20250970125198364, -0.29159364104270935, 0.10301638394594193,
0.2741706669330597, -0.0027715100441128016, 0.22495269775390625, 0.12941083312034607,
0.017645571380853653, 0.058814387768507004, 0.03783517703413963, 0.43058541417121887,
0.2957042455673218, 0.2548891305923462, 0.17691609263420105, 0.040033672004938126,
0.19580364227294922, -0.007150987163186073, -0.08062895387411118, 0.1660713255405426,
-0.13472014665603638, 0.035164810717105865, 0.2174147516489029, 0.2269885540008545,
0.05585500970482826, -0.11938035488128662, 0.3458840847015381, 0.06528293341398239,
0.3050979971885681, 0.031008640304207802, 0.5489312410354614, 0.30281180143356323,
0.2528790235519409, 0.22086110711097717, 0.2480006068944931, 0.29302552342414856,
0.32339411973953247, 0.2210780531167984, 0.3440444767475128, -0.02003985457122326,
0.26709863543510437, 0.0498044528067112, 0.13902121782302856, 0.27410995960235596,
0.06922183185815811, 0.044728200882673264, 0.12145983427762985, -0.11906171590089798,
0.05043475702404976, 0.10200698673725128, -0.20856139063835144, 0.2775029242038727,
0.008527887985110283, -0.023639868944883347, 0.10491307824850082, 0.1604854315519333,
0.30252471566200256, 0.19880631566047668, -0.03914125636219978, 0.18622452020645142,
-0.14956839382648468, -0.07268647849559784, -0.05135641619563103, -0.11374072730541229,
0.22408194839954376, -0.029740270227193832, 0.08433953672647476, 0.2275959551334381,
0.1190701350569725, -0.02169419266283512, 0.1709669530391693, -0.05432465299963951,
0.2625986635684967, 0.18539397418498993, 0.18102101981639862, -0.2537115514278412,
0.1796039193868637, -0.002537184627726674, -0.02134590409696102, 0.12194988876581192,
-0.1437775194644928, -0.12089651823043823, 0.19338160753250122, 0.3965680003166199,
0.007405652664601803, 0.2724529206752777, 0.03968621790409088, -0.3307320177555084,
-0.15668264031410217, 0.14587153494358063, 0.0635010376572609, -0.2519446611404419,
0.028525292873382 57, 0.16476821899414062, 0.012352344579994678, -0.025489898398518562,
-0.12075569480657578, -0.20715875923633575, -0.02388007380068302, 0.30804678797721863,
0.11751990765333176, -0.0649586170911789, 0.16957727074623108, 0.19760893285274506,
-0.3229627311229706, -0.04248465597629547, -0.11183422803878784, 0.13752713799476624,
-0.059087738394737244, -0.09755914658308029, 0.08997048437595367, -0.23155313730239868,
-0.09681768715381622, -0.1579795926809311, -0.11281651258468628, 0.04184875264763832,
0.09951680153608322, 0.02948482148349285, -0.047525953501462936, -0.31040868163108826,
0.014951334334909916, -0.04846563935279846, -0.042383573949337006, -0.00736591499298811,
0.07168691605329514, 0.03488847613334656, 0.17158807814121246, -0.0005084844306111336,
-0.054658204317092896, -0.11197599768638611]]"

Concurrency and auto-scaling

Both these concepts are important to Lambda. Concurrency refers to the number of requests that a Lambda function can process and Auto scaling is a feature that allows Lambda to automatically scale the number of instances based on the request volume.



Clean up

After finishing the project its important to shut down all EC2 instances, Notebook instances, Endpoints and Lambda Functions in order to prevent a large amount of costs