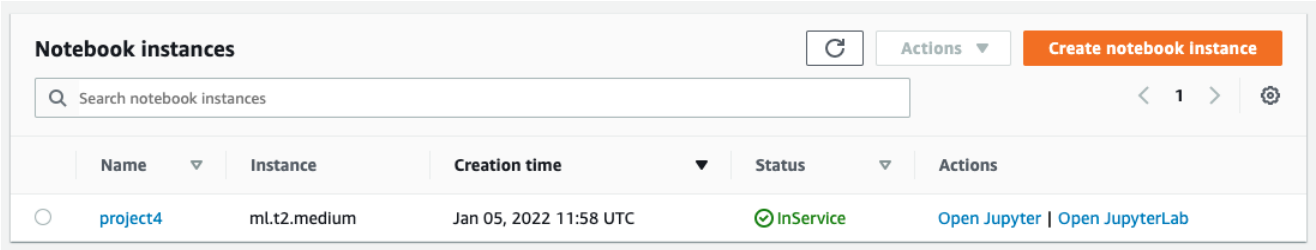
Step1

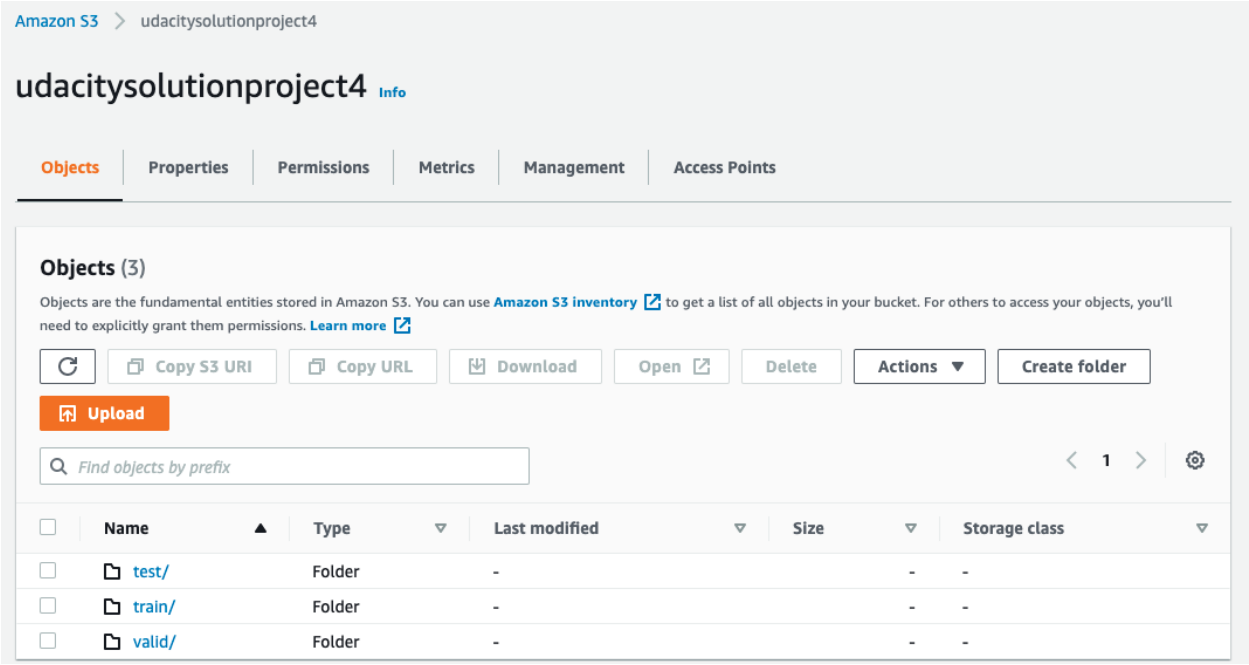
Initial setup, training and deployment

Firstly, I created a **ml.t2.medium** Sagemaker instance, which is within Free tier and still quite good for downloading, and uploading the dataset. For the training and deployment, inside the notebook, we create another instance which is suitable for image classification.



Download data to an S3 bucket

Once the notebook instance is "In Service", we open a jupyterLab and first download the dataset and unzip it, then create a bucket in S3 and upload the unzipped dataset to the bucket.



Training and Deployment

To use a pre-trained model, we set a hyperparameter range. The estimator uses a **ml.m5.xlarge** instance which is appropriate for image processing. After finding the best training job out of two training jobs (**pytorch-training** …), we run the estimator with the hyperparameter of the best training job. The outcome of the training with the best hyperparameters is named **dog-pytorch… .**

dog-pytorch-2022-01-05-12-43-18-817	Jan 05, 2022 12:43 UTC	23 minutes	Completed
pytorch-training-220105-1215-002-21462345	Jan 05, 2022 12:15 UTC	22 minutes	Completed
pytorch-training-220105-1215-001-4c175d11	Jan 05, 2022 12:15 UTC	21 minutes	Completed

After the training job is completed the endpoint is deployed.

pytorch-inference-2022-01-05-13-21-52-770 Delete

Endpoint settings

Name	Status	Type	URL
pytorch-inference-2022-01-05-13-21-52-770	InService	Real-time	https://runtime.sagemaker.us-east-1.amazonaws.com/endpoints/pytorch-inference-2022-01-05-13-21-52-770/invocations
ARN	Creation time	Last updated	Learn more about the API
arn:aws:sagemaker:us-east-1:119699166201:endpoint/pytorch-inference-2022-01-05-13-21-52-770	Wed Jan 05 2022 15:21:53 GMT+0200 (Eastern European Standard Time)	Wed Jan 05 2022 15:24:29 GMT+0200 (Eastern European Standard Time)	

Multi-instance training

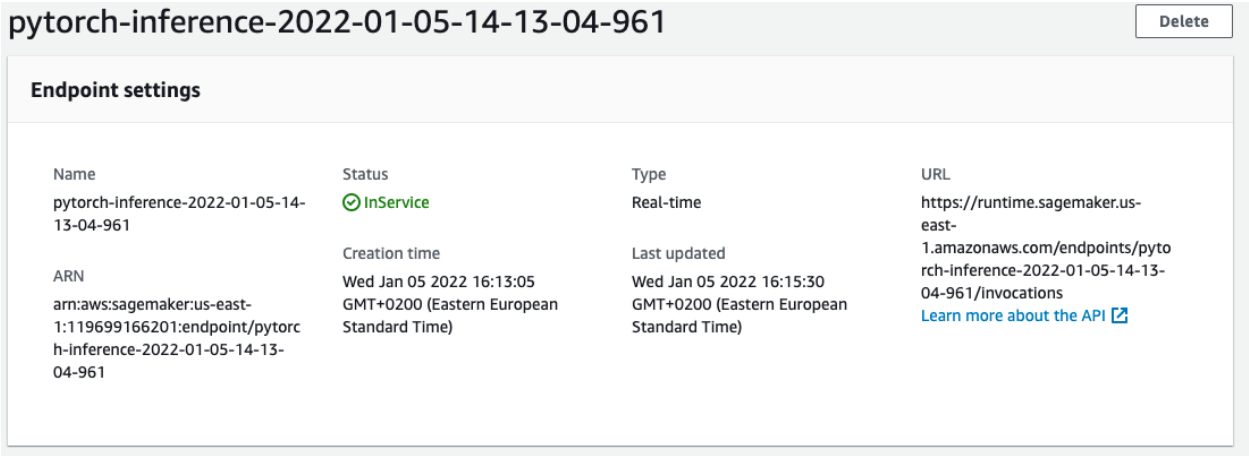
I changed the estimator setting to have 4 instances for multiple instances training.

Training jobs Actions Create training job

Search training jobs 1

Name	Creation time	Duration	Status
dog-pytorch-multi-2022-01-05-13-46-39-508	Jan 05, 2022 13:46 UTC	24 minutes	Completed

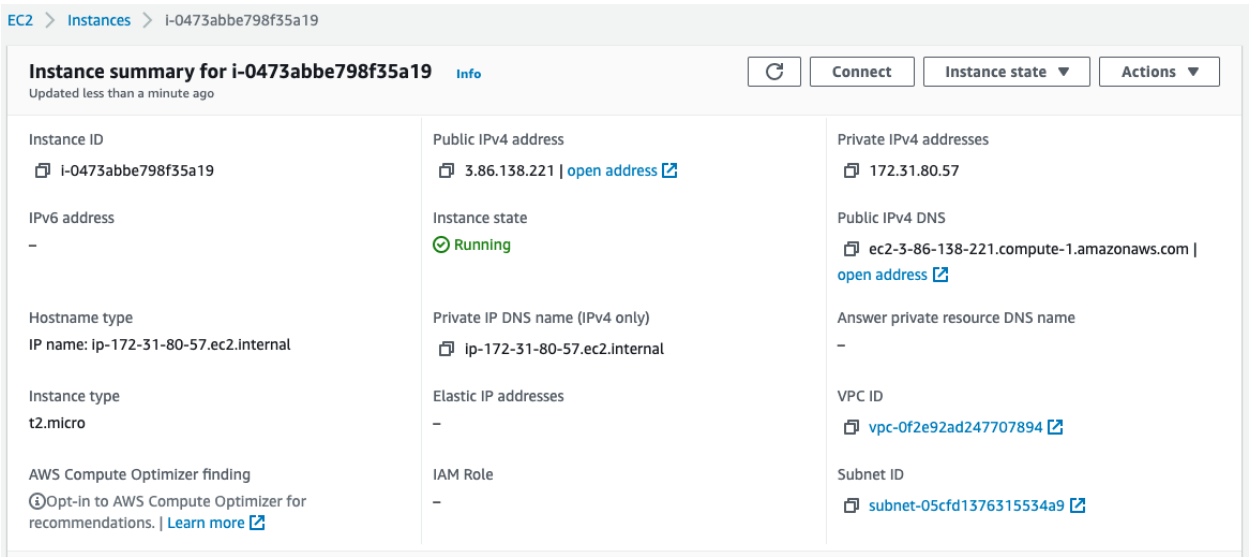
After completing the training, I deployed the endpoint.



Step 2

EC2 Setup

In the EC2 dashboard, I launched an EC2 instance with "**Deep Learning AMI (Amazon Linux 2) Version 56.0**" and a **t2.micro** instance which was free tier. After the instance was launched, I needed to connect to it via my computer terminal. The SSH connection was created with "**ssh -i "my-key-pair.pem" ec2-user@ec2-3-86-138-221.compute-1.amazonaws.com**" command.



Training and saving on EC2

Once the ssh connection was established, I followed the instruction and downloaded the dataset, created a training script named **solution.py** , and trained the model once again within EC2. The trained model was saved in the TrainedModels directory:

```
fsiavash — ec2-user@ip-172-31-80-57:~ — ssh -i my-key-pair.pem ec2-u...
  inflating: dogImages/valid/132.Xoloitzcuintli/Xoloitzcuintli_08299.jpg
  inflating: dogImages/valid/132.Xoloitzcuintli/Xoloitzcuintli_08301.jpg
  inflating: dogImages/valid/132.Xoloitzcuintli/Xoloitzcuintli_08304.jpg
   creating: dogImages/valid/133.Yorkshire_terrier/
  inflating: dogImages/valid/133.Yorkshire_terrier/Yorkshire_terrier_08333.jpg
  inflating: dogImages/valid/133.Yorkshire_terrier/Yorkshire_terrier_08334.jpg
  inflating: dogImages/valid/133.Yorkshire_terrier/Yorkshire_terrier_08336.jpg
  inflating: dogImages/valid/133.Yorkshire_terrier/Yorkshire_terrier_08348.jpg
[[ec2-user@ip-172-31-80-57 ~]$ ls                                          ]
 LICENSE                README     dogImages      examples  tools
 Nvidia_Cloud_EULA.pdf  anaconda3  dogImages.zip  src       tutorials
[[ec2-user@ip-172-31-80-57 ~]$ mkdir TrainedModels                          ]
[[ec2-user@ip-172-31-80-57 ~]$ vim solution.py                              ]
[[ec2-user@ip-172-31-80-57 ~]$ ls                                          ]
 LICENSE                TrainedModels  dogImages.zip  src
 Nvidia_Cloud_EULA.pdf  anaconda3      examples       tools
 README                 dogImages      solution.py    tutorials
[[ec2-user@ip-172-31-80-57 ~]$ vim solution.py                              ]
[[ec2-user@ip-172-31-80-57 ~]$ python solution.py                           ]
 Traceback (most recent call last):
   File "solution.py", line 1, in <module>
     import numpy as np
 ImportError: No module named numpy
[[ec2-user@ip-172-31-80-57 ~]$ source activate pytorch_latest_p37           ]
[(pytorch_latest_p37) [ec2-user@ip-172-31-80-57 ~]$ python solution.py      ]
 Downloading: "https://download.pytorch.org/models/resnet50-19c8e357.pth" to /hom
 e/ec2-user/.cache/torch/hub/checkpoints/resnet50-19c8e357.pth
 100%|[=================================================]| 97.8M/97.8M [00:00<00:00, 114MB/s]
 Starting Model Training
 saved
[(pytorch_latest_p37) [ec2-user@ip-172-31-80-57 ~]$ ls TrainedModels/      ]
 model.pth
[(pytorch_latest_p37) [ec2-user@ip-172-31-80-57 ~]$ ls                     ]
 LICENSE                TrainedModels  dogImages.zip  src
 Nvidia_Cloud_EULA.pdf  anaconda3      examples       tools
 README                 dogImages      solution.py    tutorials
[(pytorch_latest_p37) [ec2-user@ip-172-31-80-57 ~]$ ls TrainedModels/      ]
 model.pth
 (pytorch_latest_p37) [ec2-user@ip-172-31-80-57 ~]$
```

Comparing EC2 and Sagemaker

The codes that trained the models are the same in both instances. The main differences between these two approaches are 1) The EC2 does not save the dataset or upload the trained model into S3, while Sagemaker downloads the dataset in S3 and saves the trained model here as well. 2)Sagemaker uses external scripts for hpo and inference, whereas in the EC2 only one script includes all functionalities. 3) Sagemaker enables testing the

deployment of the endpoint within itself, while for EC2 a Lambda Function is required to test the deployment.

Step 3

Setting up a Lambda function

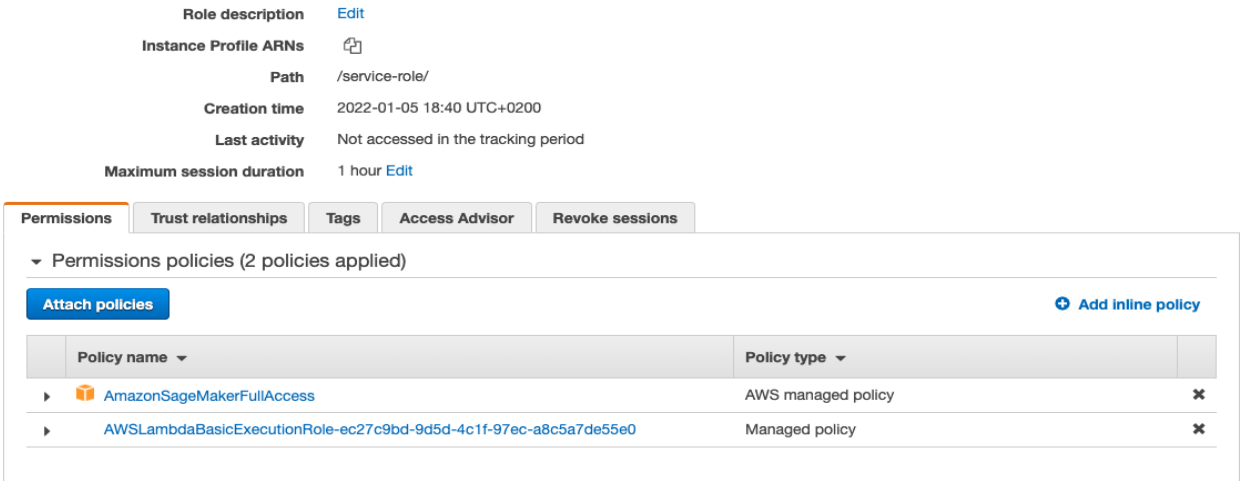
I changed the default name of the endpoint in the Lambdafunction.py in the starter file to the name of the deployed endpoint. Then, I copied the code and created a lambda function instance and added the code in there. The lambda function includes following functionalities:

- Creates a Boto3 client to communicate to the endpoint
- Gets the input data which is a url of an image
- Sends the image to the endpoint and receives the response
- The response body includes an array of 133 items, each represents a prediction score of a dog breed.
- By getting the index of the maximum item from the array, we can see what is the breed prediction of the input image.

Step 4

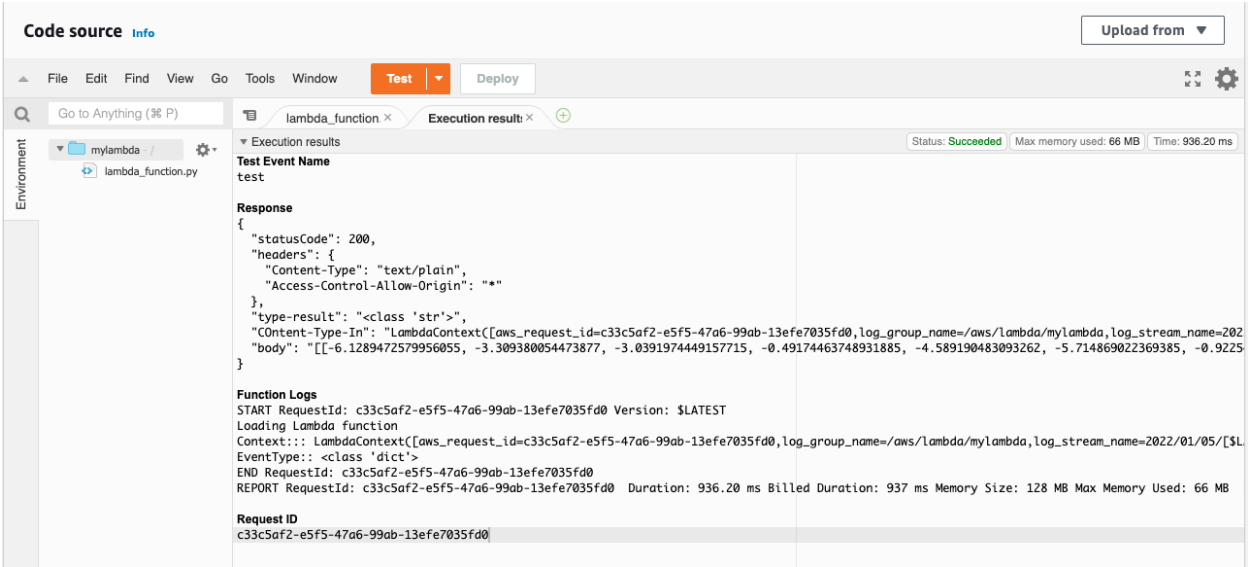
Lambda function security

In order to set the appropriate permission, we attached a policy as shown in the figure below.



Lambda function testing

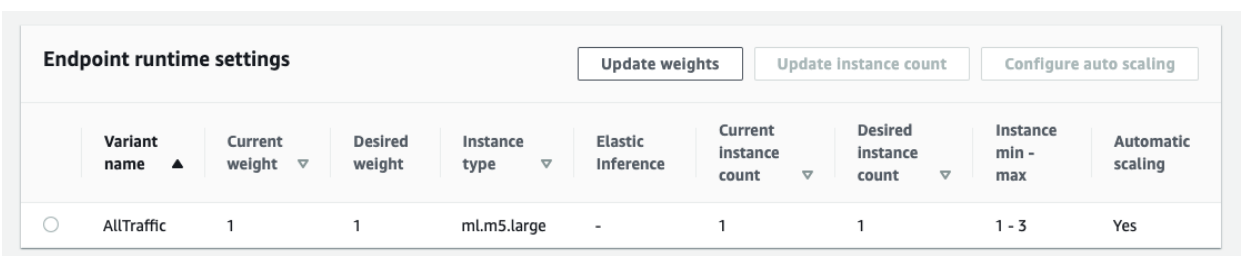
For testing the lambda function, we created a test with an image url as the input. The result of the test is shown below.



Step 5

Concurrency and auto scaling

To create concurrency in the lambda function, I changed the configuration and created a new version of the function. I provisioned 1 concurrency. For my current project this is more than required, however, to practice the procedure, I also added the auto scaling to the deployed endpoint. The min instance count is set to 1 and max is set to 3, scale-in and scale-out are 30 sec.



Variant name	Current weight	Desired weight	Instance type	Elastic Inference	Current instance count	Desired instance count	Instance min - max	Automatic scaling
AllTraffic	1	1	ml.m5.large	-	1	1	1 - 3	Yes