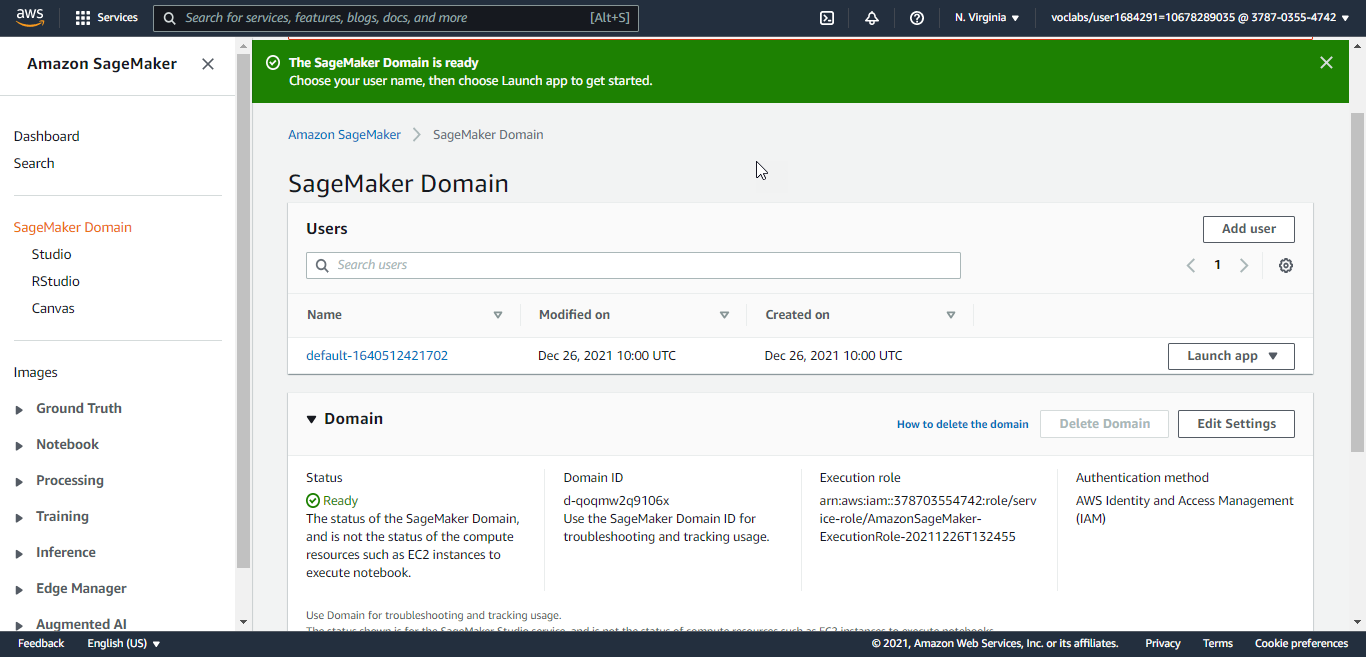
—---------------------------------------------Javad Saljooghi-----------------------------------------------------

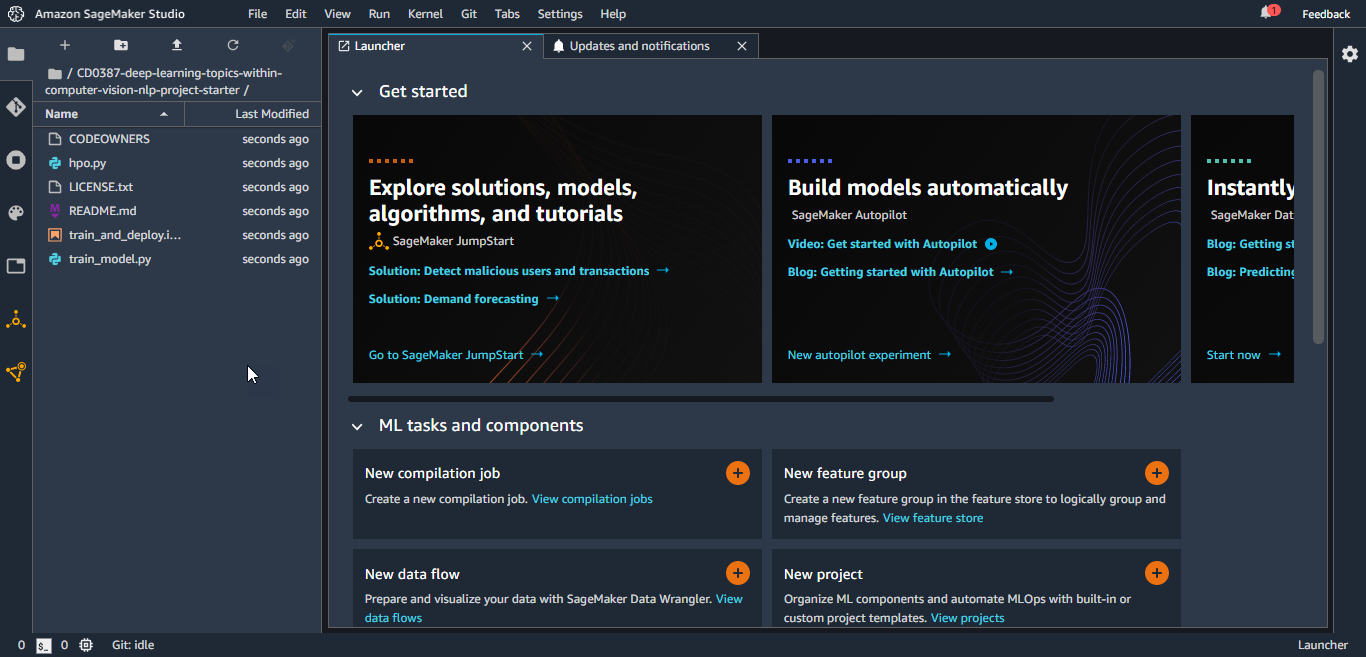
**Image Classification using AWS SageMaker–Project**

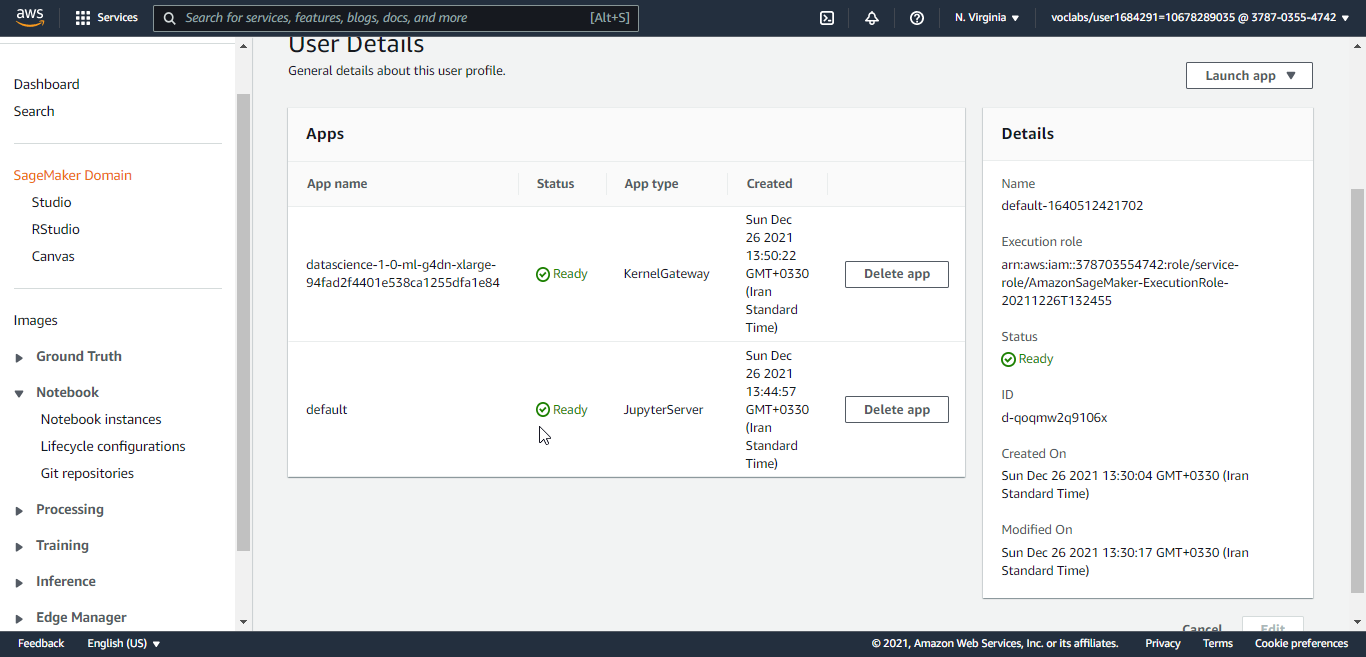
—-----------------------------------------------------------------------------------------------------------------------

Creating SageMaker studio domain:

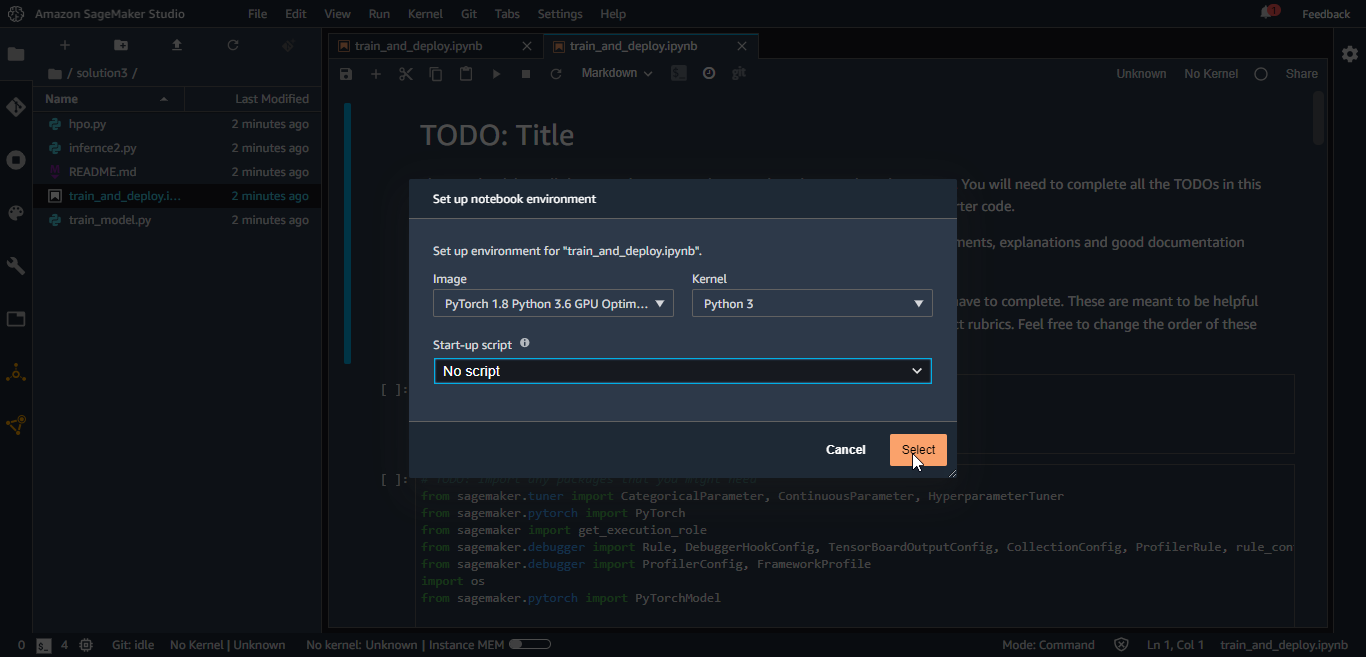


Clone the Repo:

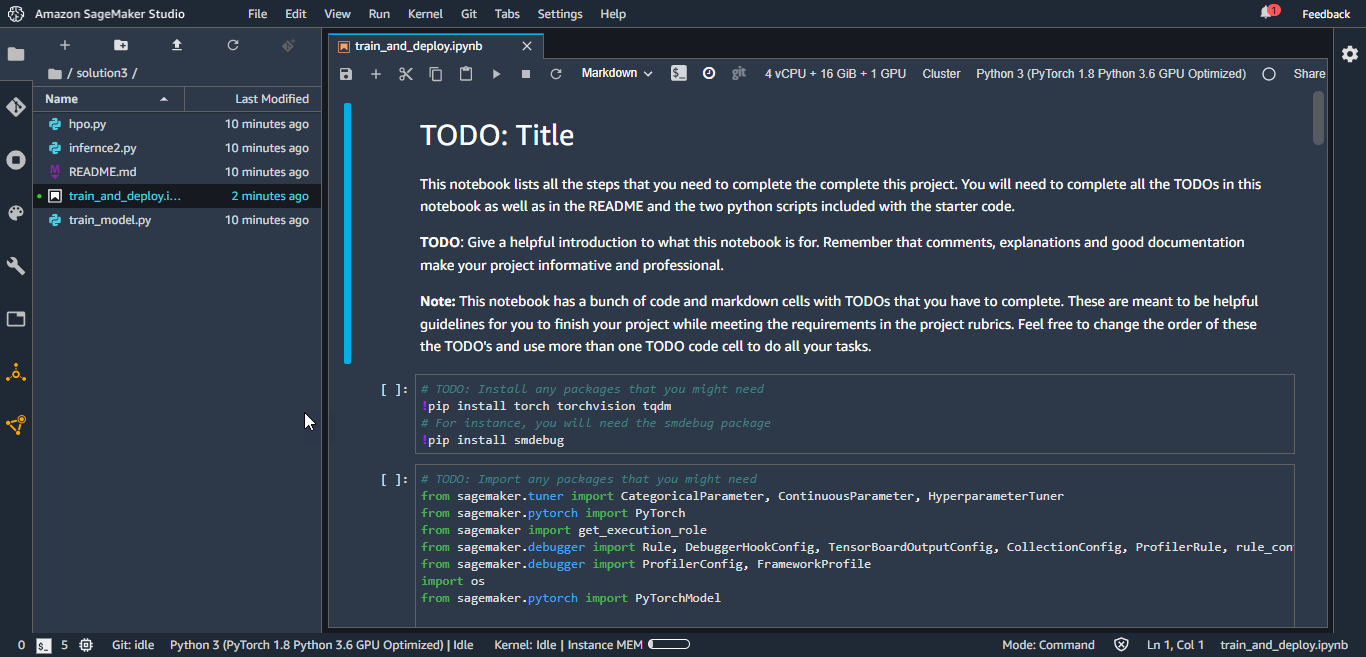


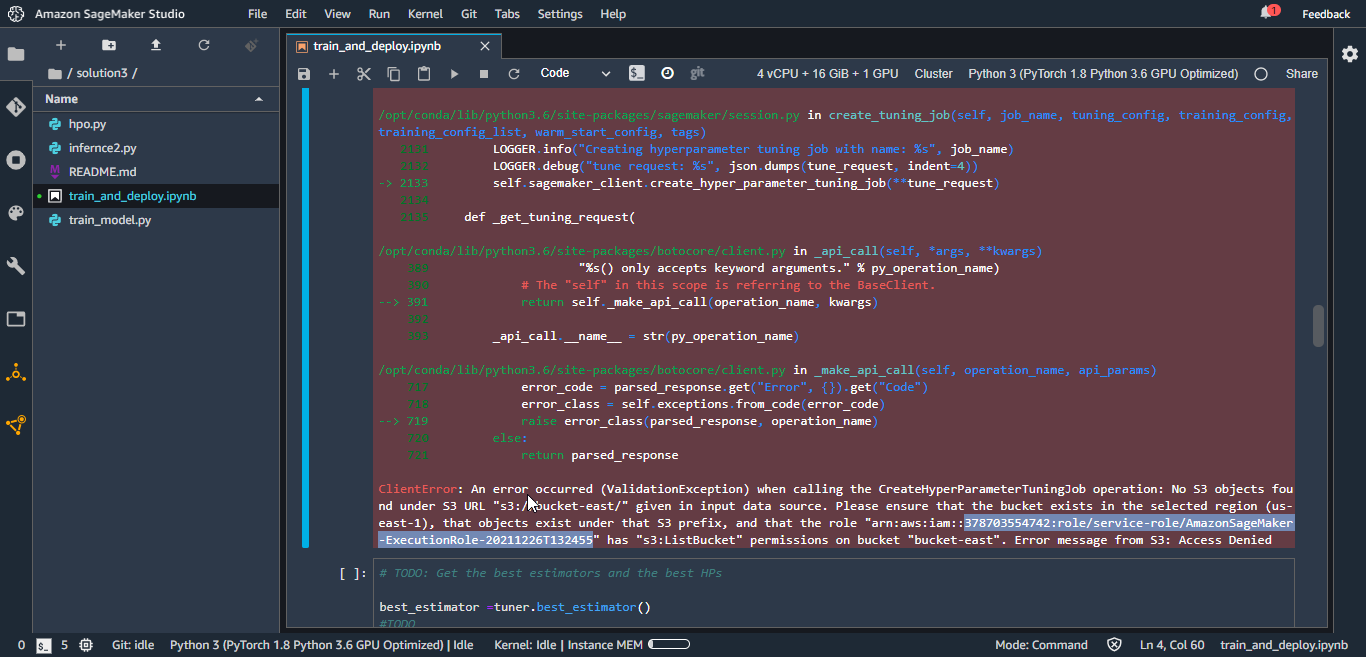


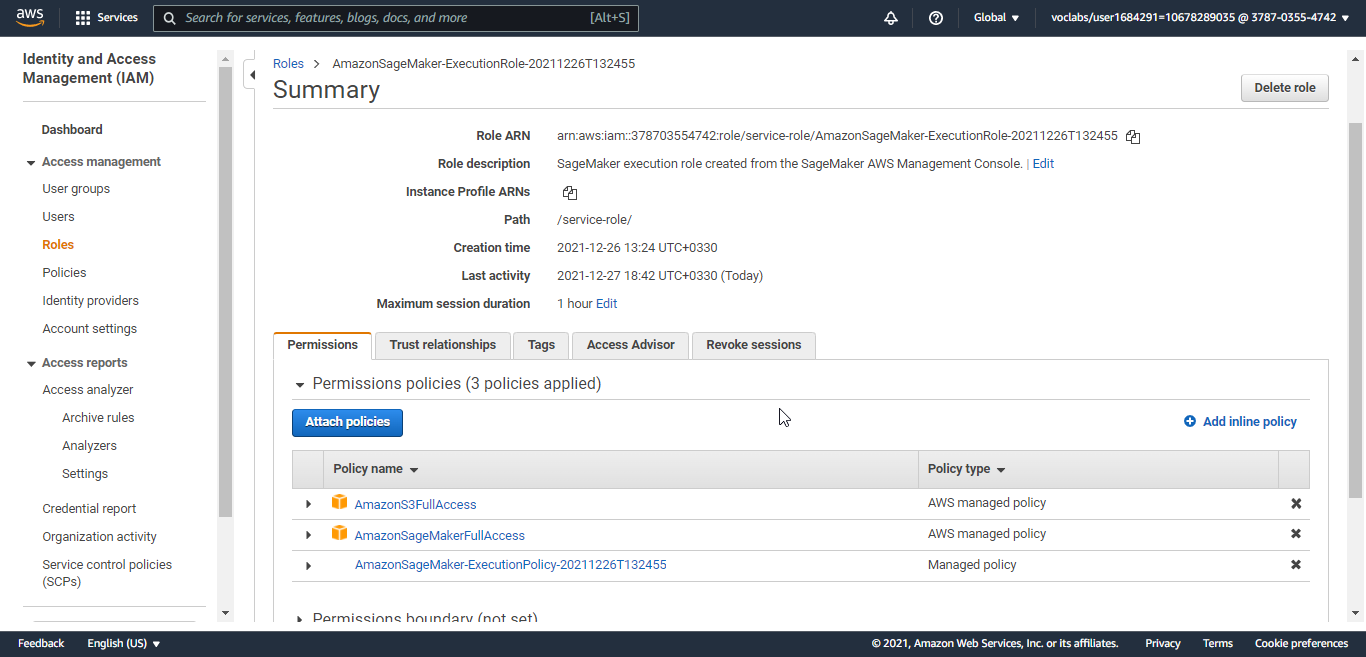
Starting kernel and environment folder:



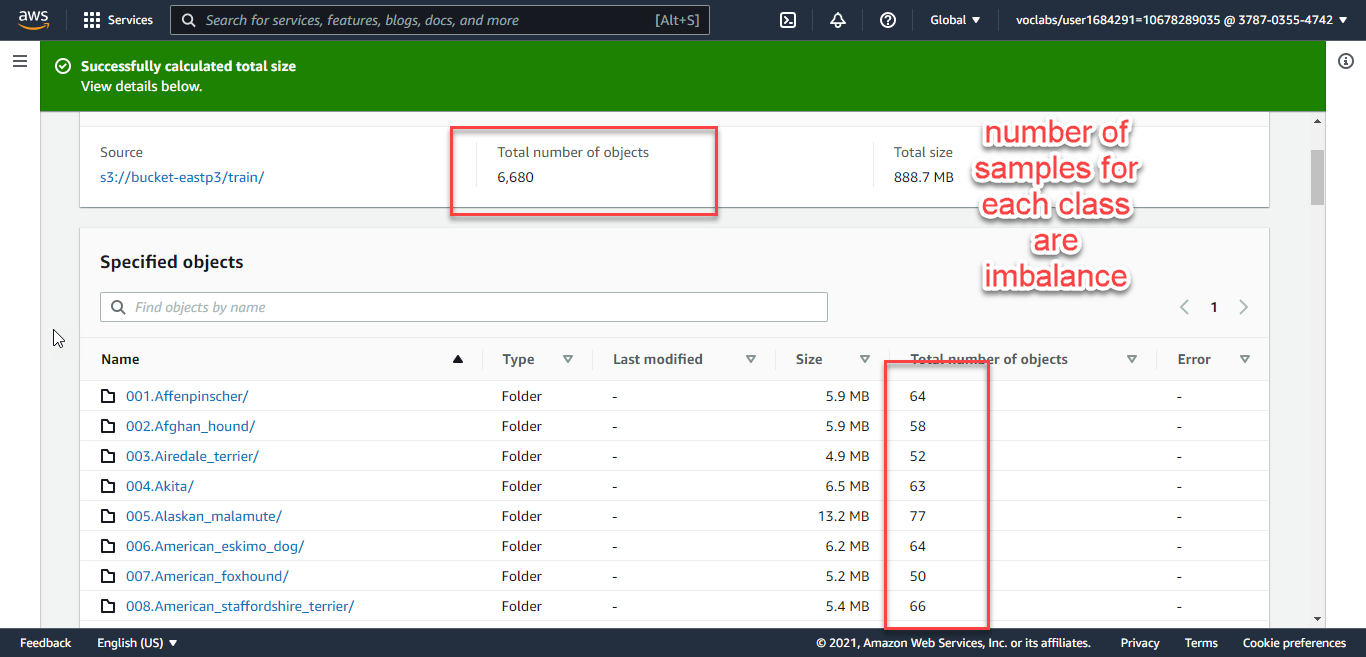
Since I'm going to do the project with PyTorch , I used PyTorch 1.8 Python 3.6 kernel and due to doing deep learning on images I used GPU optimized. but there is an Error with the method of sagemaker.get\_execution\_role(), from github I found that images of kernel that used has this bug ( <https://github.com/aws/sagemaker-python-sdk/issues/2186> ) and cannot find execution role even it is in /service-role/ path and the name of role contatines "ExceutionRole", so I decided to switch to a kernel that does not have this bug which is Python3-Data science kernel.

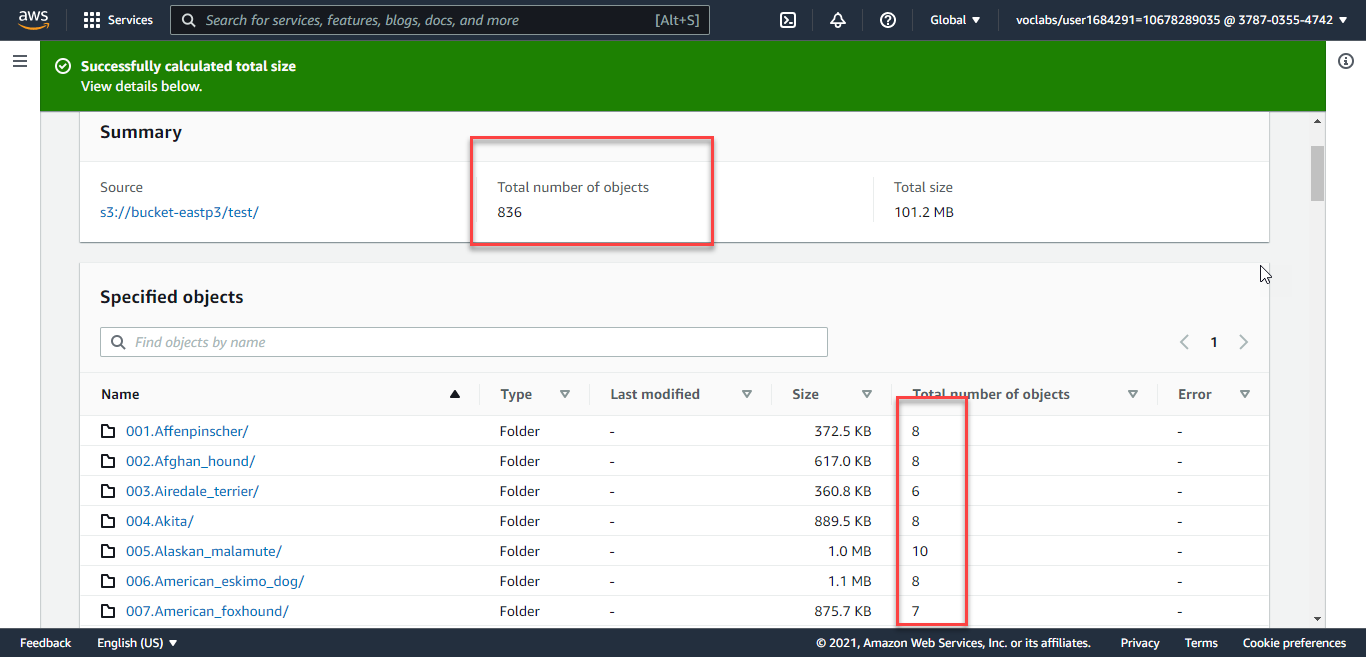


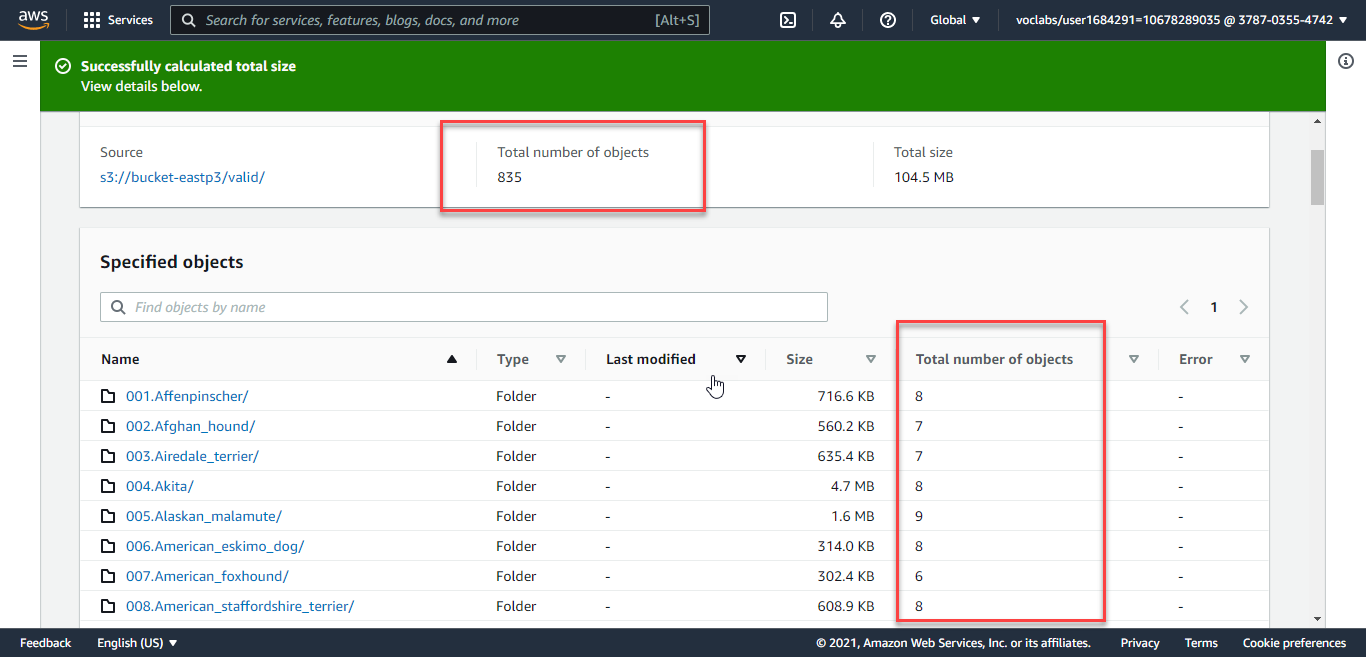




I used dogImages dataset which has 3 Folders, train(6680 samples-number of dog breeds sample are imbalance some has more pics with transformation like flipping and mirroring we can add more sample to give a same weight to all of our breeds so that during learning process all breeds be considered with equal weight), test (836 samples), validation (835 samples)…..and in each folder there are 121 dog breeds or classes(name of the folder)….the goal is by training and tuning our PyTorch classifier we could classify correctly new dog images….for this purpose we use images in test folder which has not seen by our ML and for the first time it encounters with them. Also for fine tuning hyperparameters we will use validation images in validation folder.

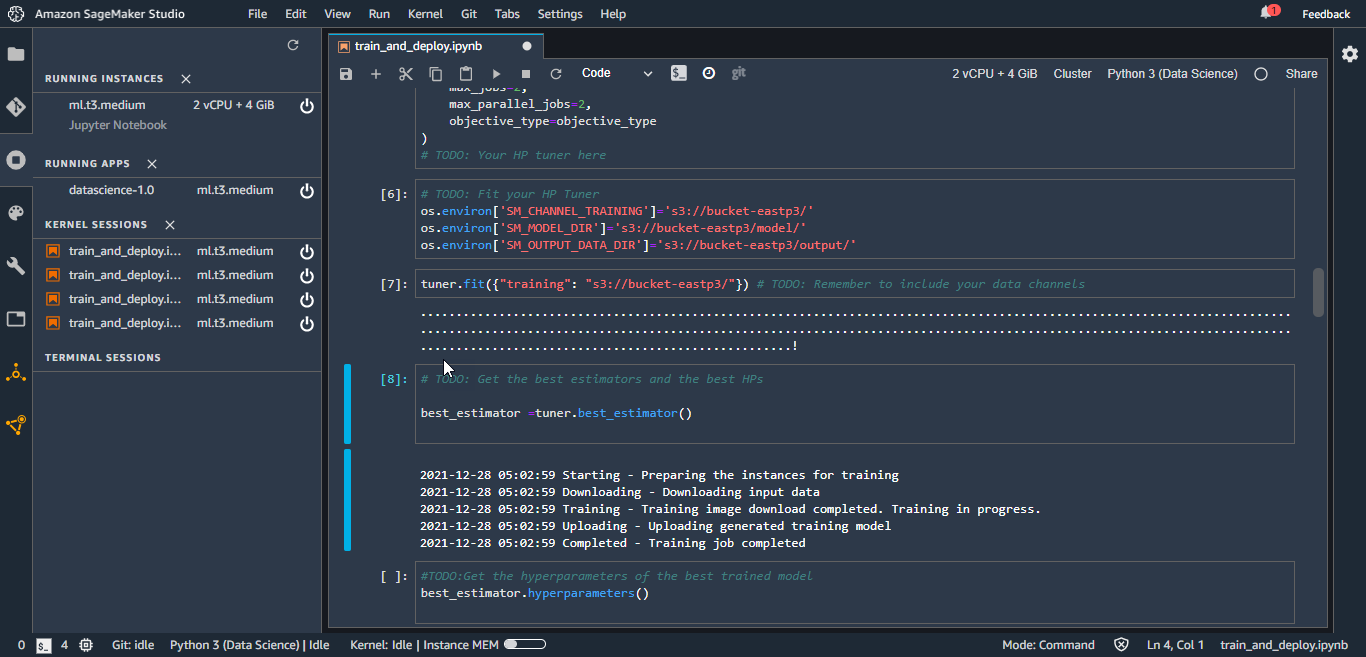


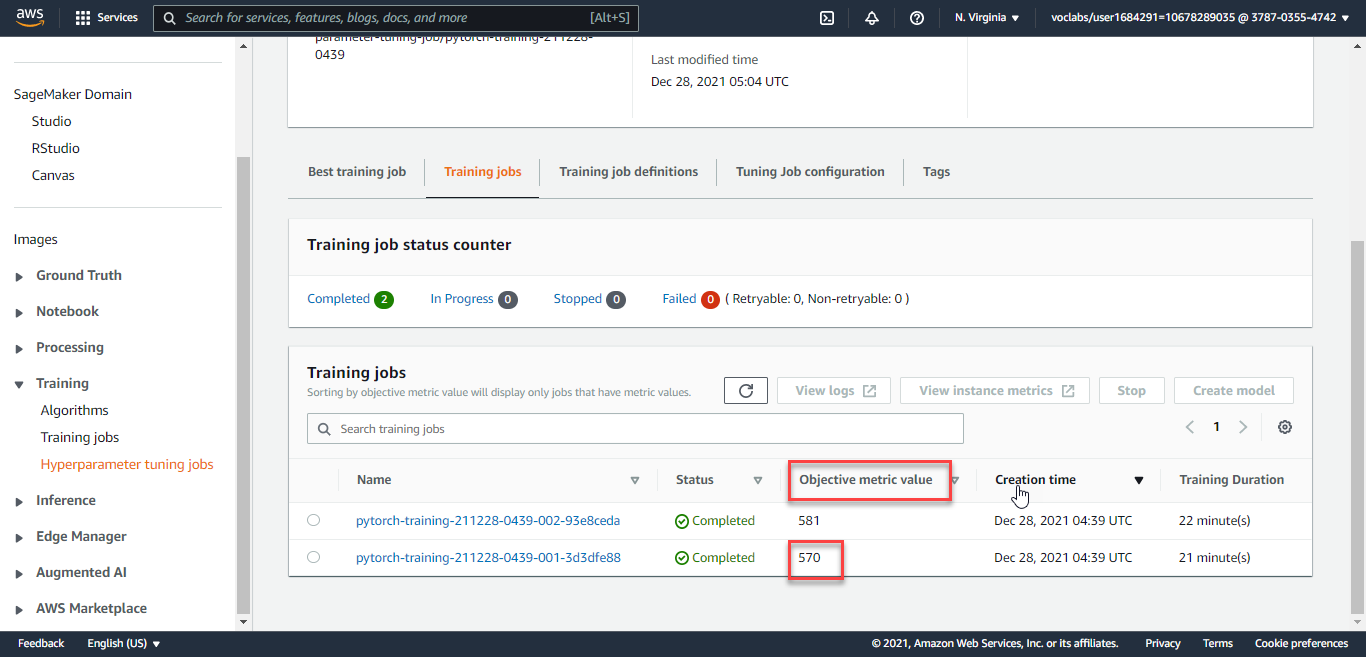


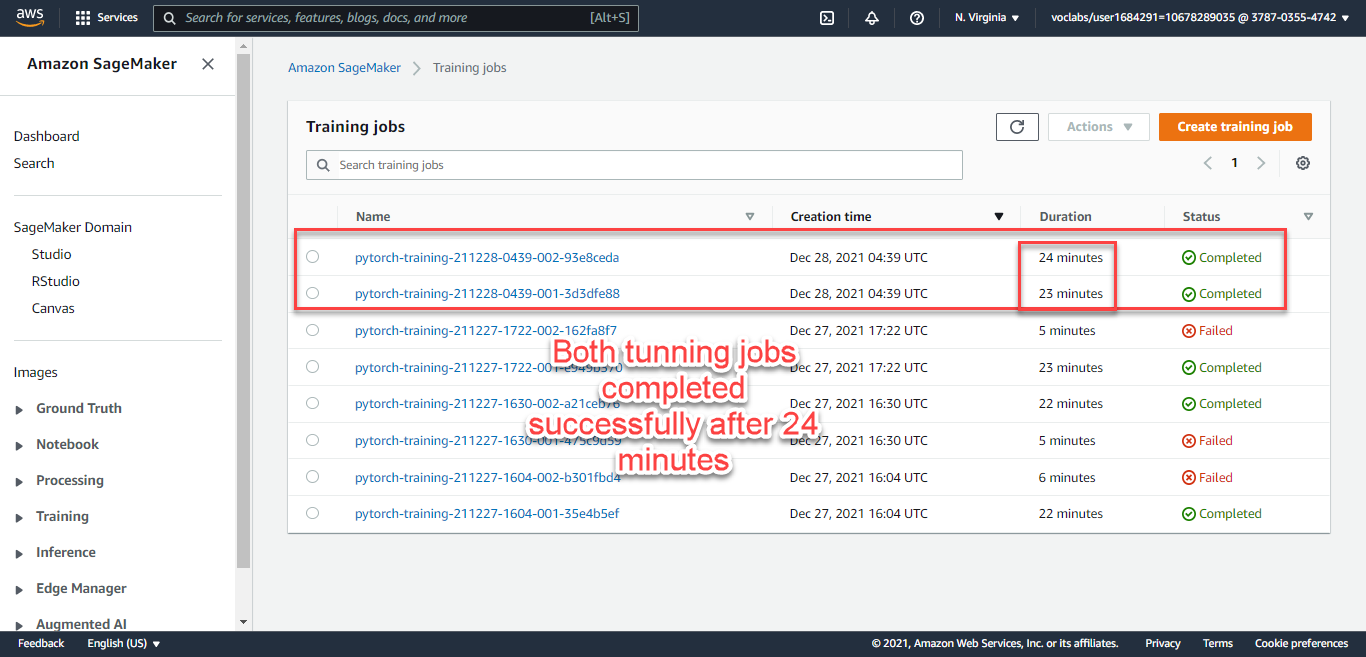


In total we have: 6680+836+835=8351 images; train=80%, test=10%,valid=10% split of data which is acceptable since deep learning needs more training data to calculate models weights and finding best model hyperparameters

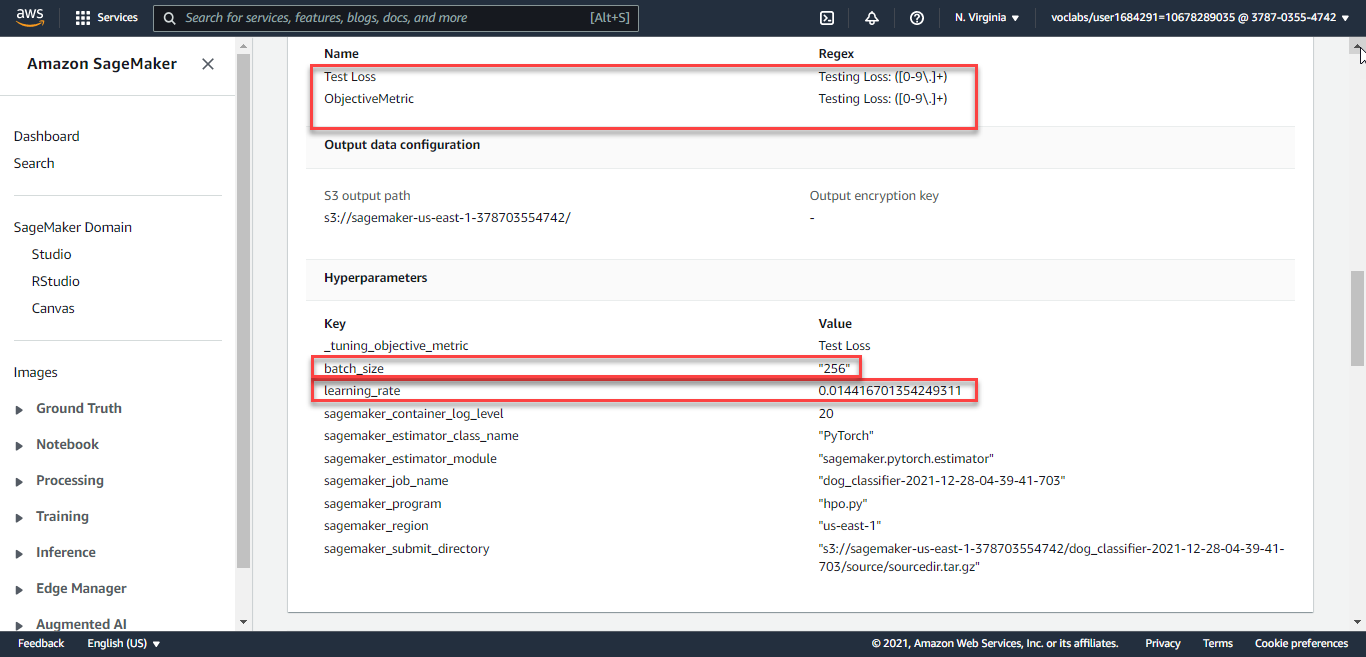
Training job:



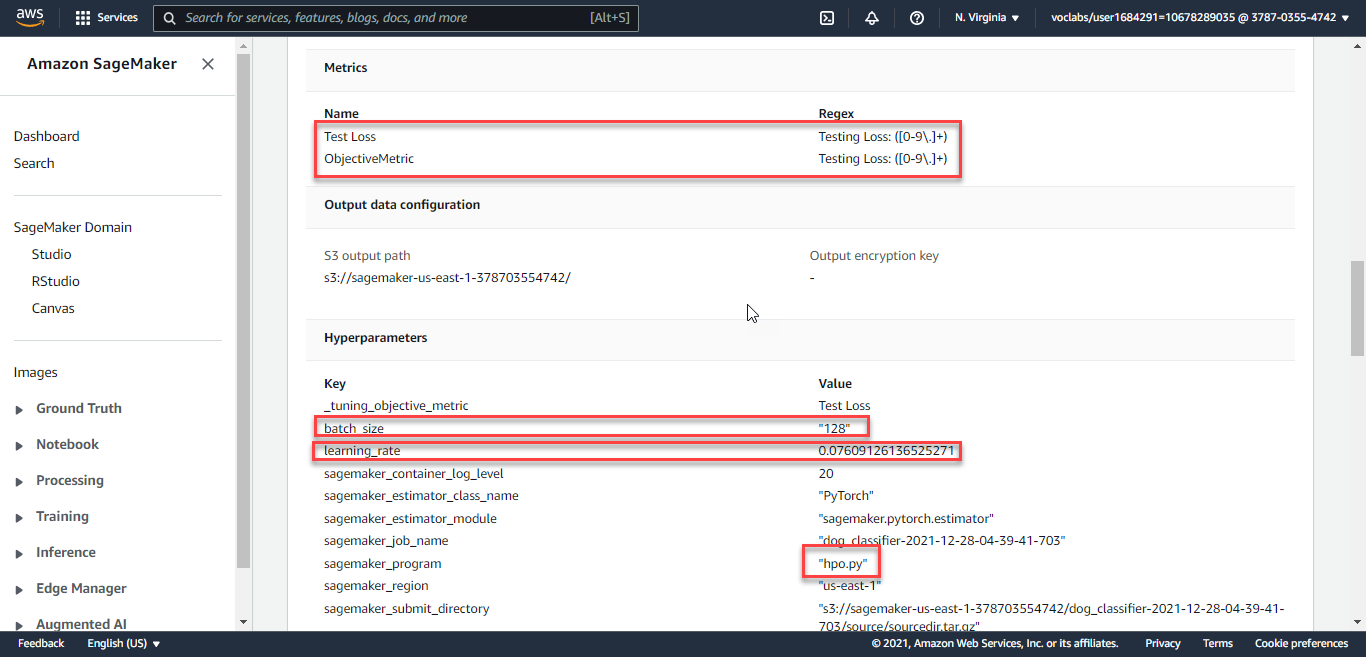




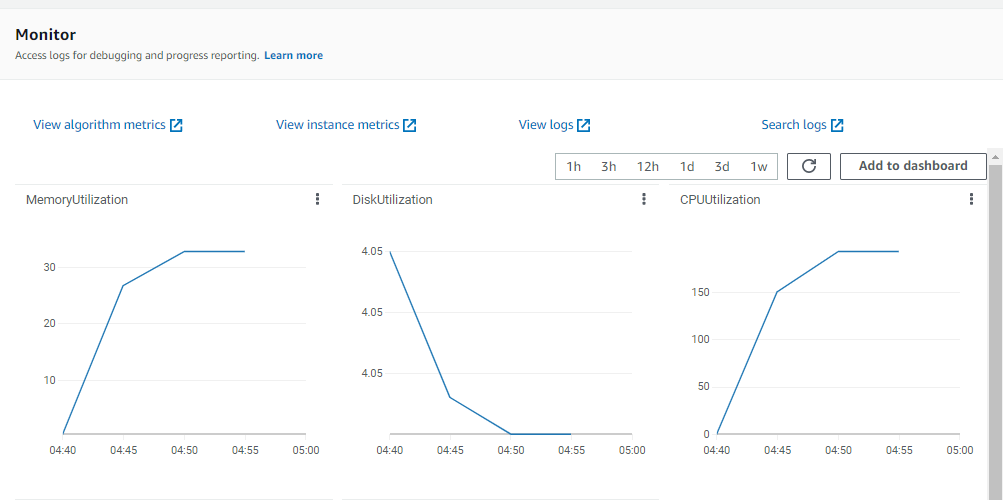
Training job 001 batch\_size getting from the range given is 256 and learning\_rate is 0.0144

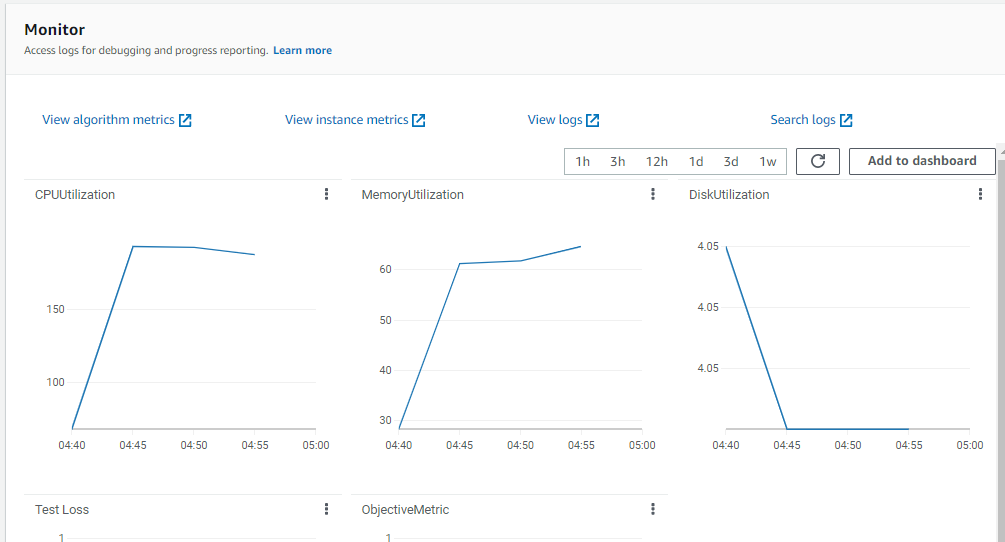


Training job 002 batch\_size getting from the range given is 128 and learning\_rate is 0.0760

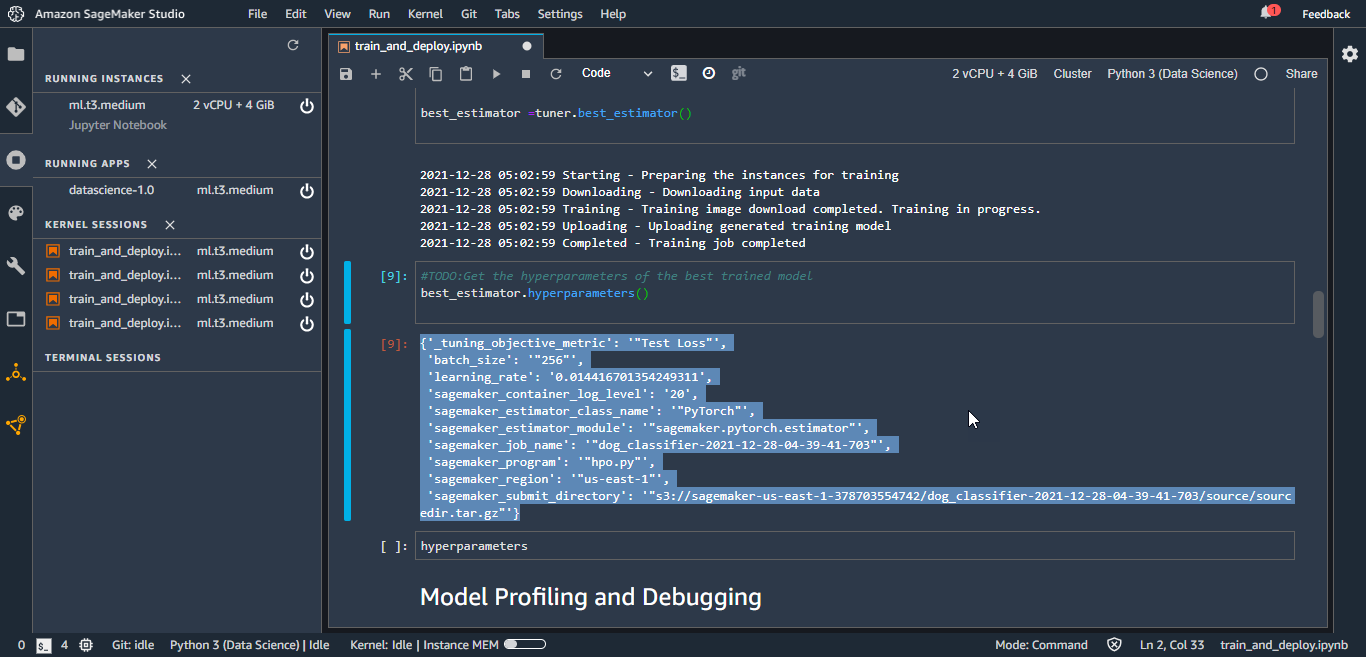


CPU and RAM and IO utilization during job training can be seen here:

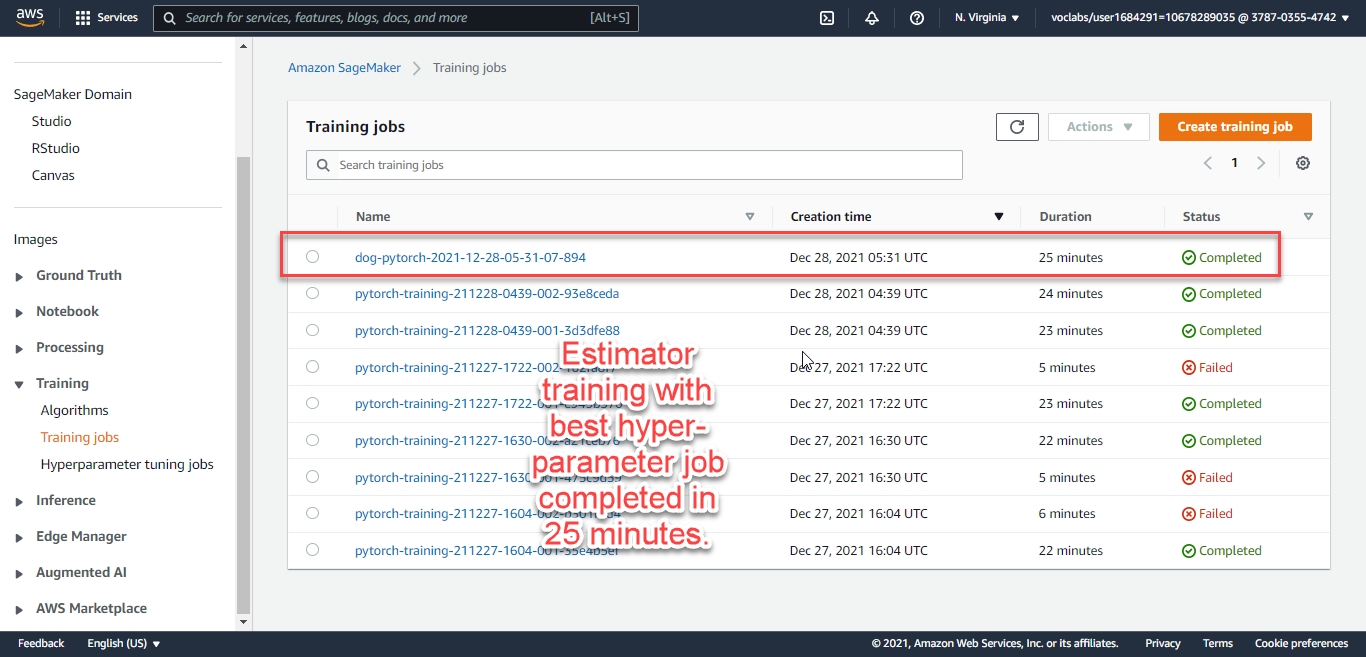




Best estimator hyperparameters are from job 001 which are:

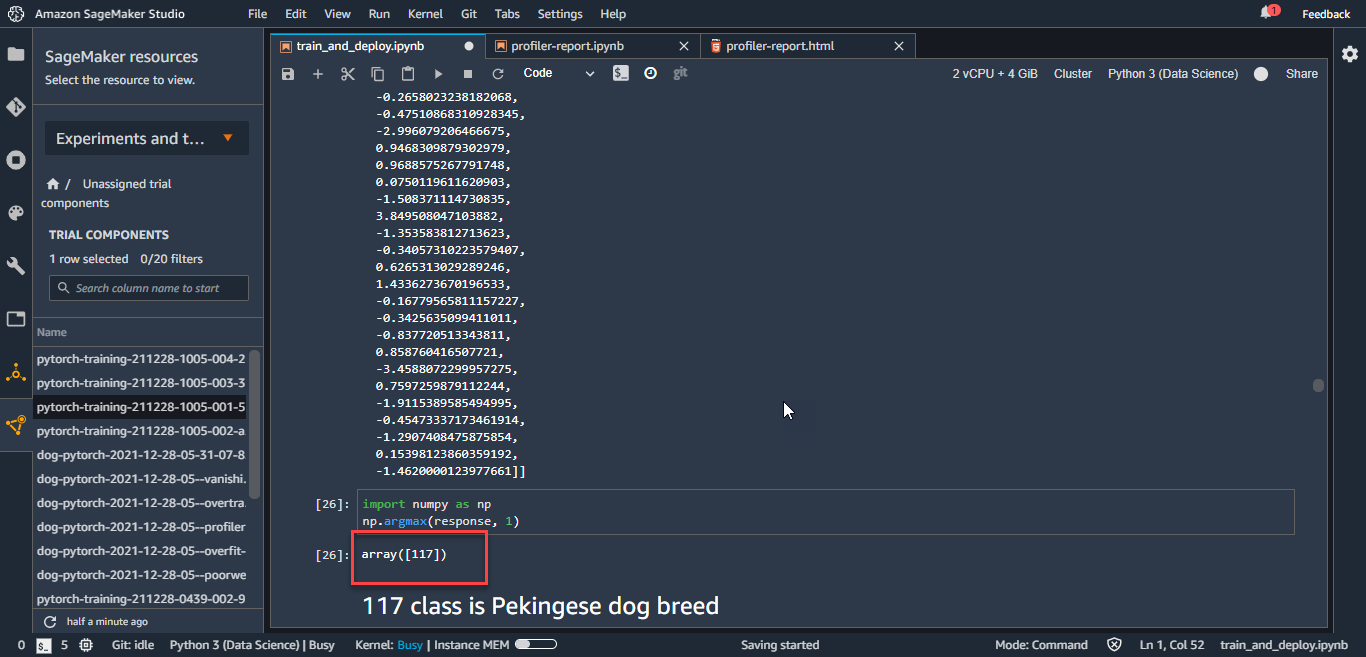


Model Profiling and Debugging:

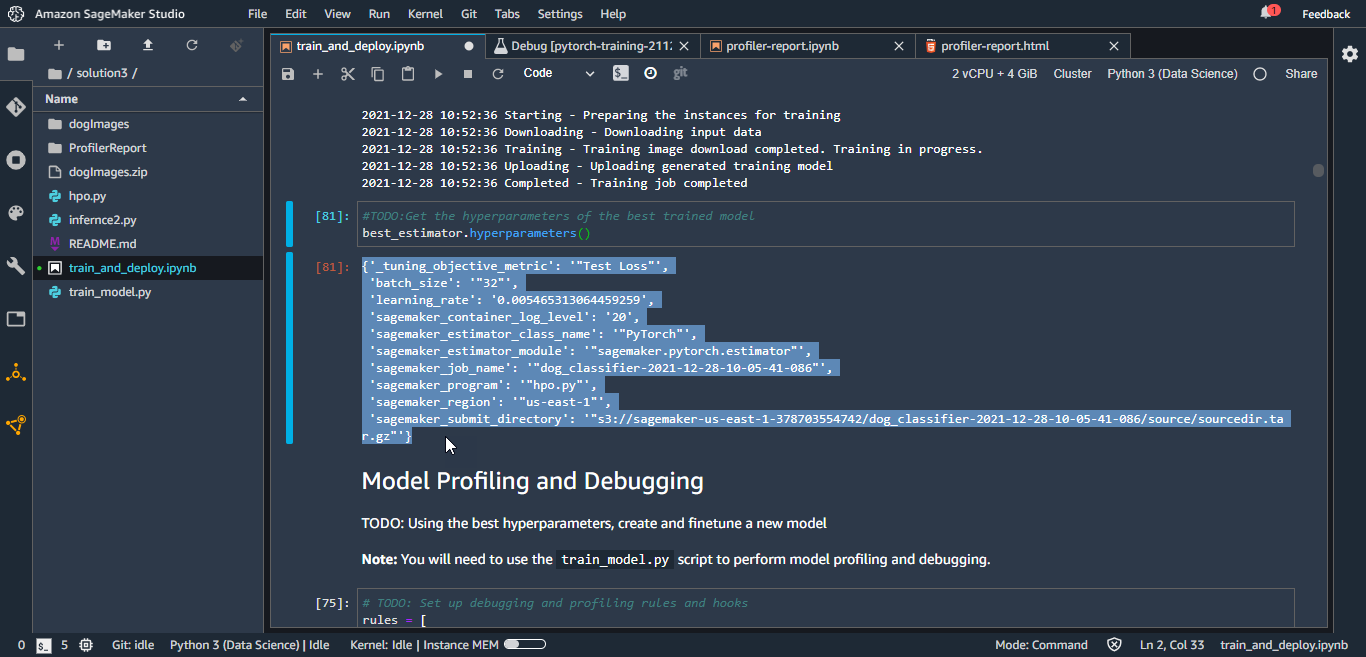


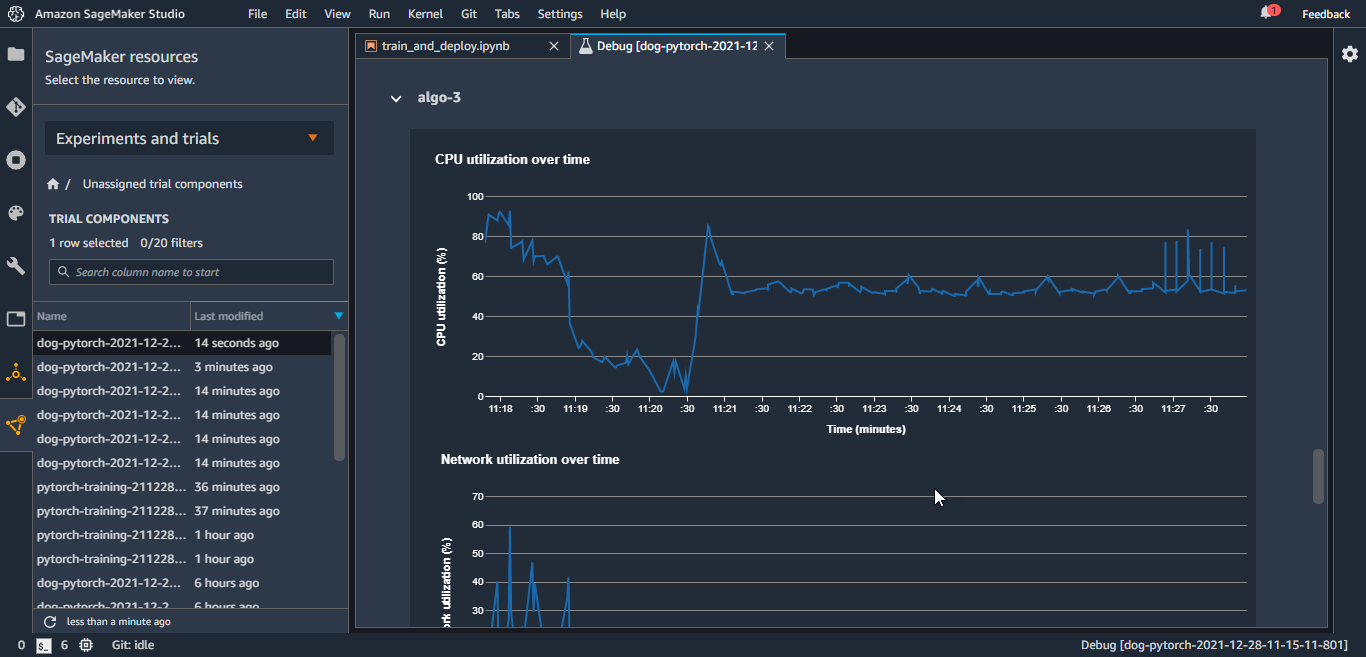
After deploying the Model, I invoked the endpoint with different dog pictures but from 133 classes for different dog breeds; it predicts the same class which is breed 117 with highest probability compared to other classes, therefore, I think my training needs to be done again since model is biased toward specific class (117).

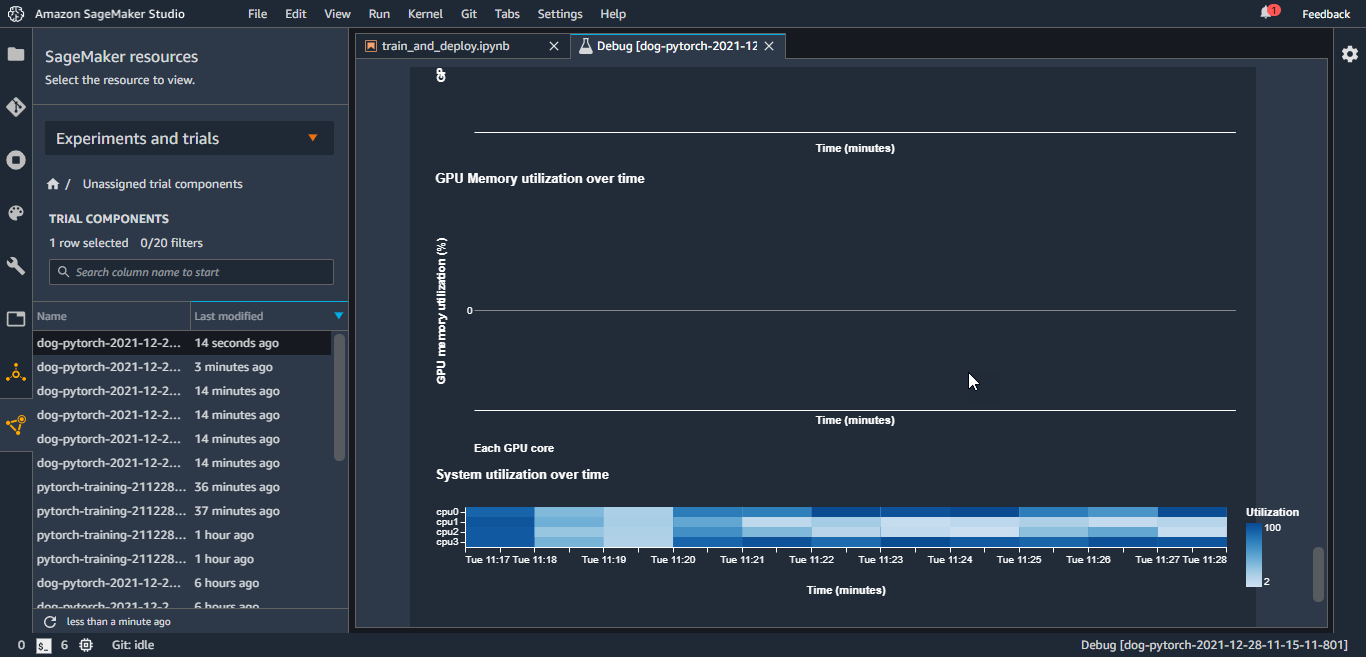
Changing hyperparameter range for batch\_size and learning\_rate and did the job for max\_job of 4 instead of 2:



The reports for new training with different hyperparameters tuning:







since selected kernel (data science does not utilizes GPU, GPU RAM and GPU are 0)

I did profile ML training Jobs with Amazon SageMaker Debugger by using this script, from this link: <https://aws.amazon.com/blogs/aws/profile-your-machine-learning-training-jobs-with-amazon-sagemaker-debugger/>

rules = [

    Rule.sagemaker(rule\_configs.vanishing\_gradient()),

    Rule.sagemaker(rule\_configs.overfit()),

    Rule.sagemaker(rule\_configs.overtraining()),

    Rule.sagemaker(rule\_configs.poor\_weight\_initialization()),

    ProfilerRule.sagemaker(rule\_configs.ProfilerReport()),

]

hook\_config = DebuggerHookConfig(

    hook\_parameters={

        "train.save\_interval": "1",

        "eval.save\_interval": "1"

    }

)

profiler\_config = ProfilerConfig(

    system\_monitor\_interval\_millis=200, framework\_profile\_params=FrameworkProfile(num\_steps=1)

)

Unfortunately, cannot see graphs that SageMaker auto generate it from ipynb file and .json files, put all of files and folders in profile\_output folder….I also changes the kernel to PyTorch 1.6 but still no access and having empty sections in reports.html file.

<https://sagemaker-examples.readthedocs.io/en/latest/sagemaker-debugger/tensorflow_nlp_sentiment_analysis/sentiment-analysis-tf-distributed-training-bringyourownscript.html>

Although I did not configure Debugger Hook to collect Tensors, I still have to get system metrics like CPU,GPU …usages on my report I get them on Sage Maker studio….

<https://docs.aws.amazon.com/sagemaker/latest/dg/debugger-configure-hook.html>

<https://knowledge.udacity.com/questions/771244>

<https://docs.aws.amazon.com/sagemaker/latest/dg/debugger-configure-framework-profiling.html>

<https://docs.aws.amazon.com/sagemaker/latest/dg/debugger-profiling-report.html>

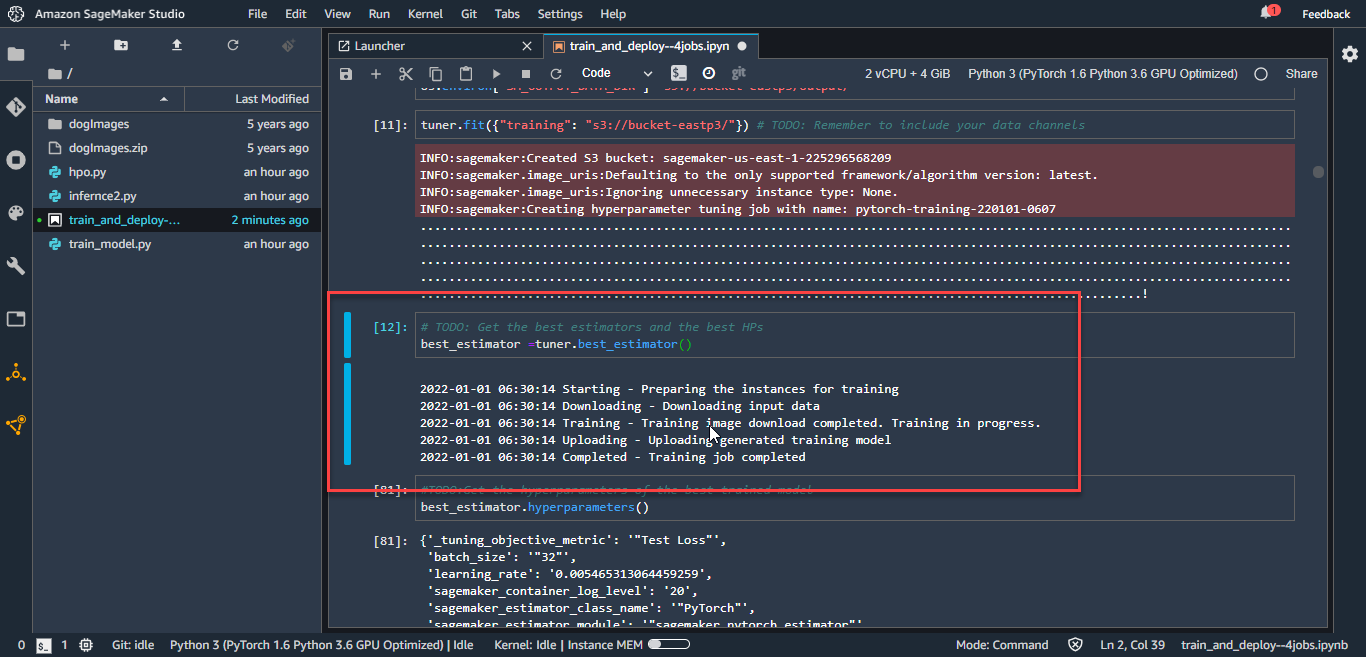
<https://github.com/awslabs/sagemaker-debugger/blob/master/docs/pytorch.md>

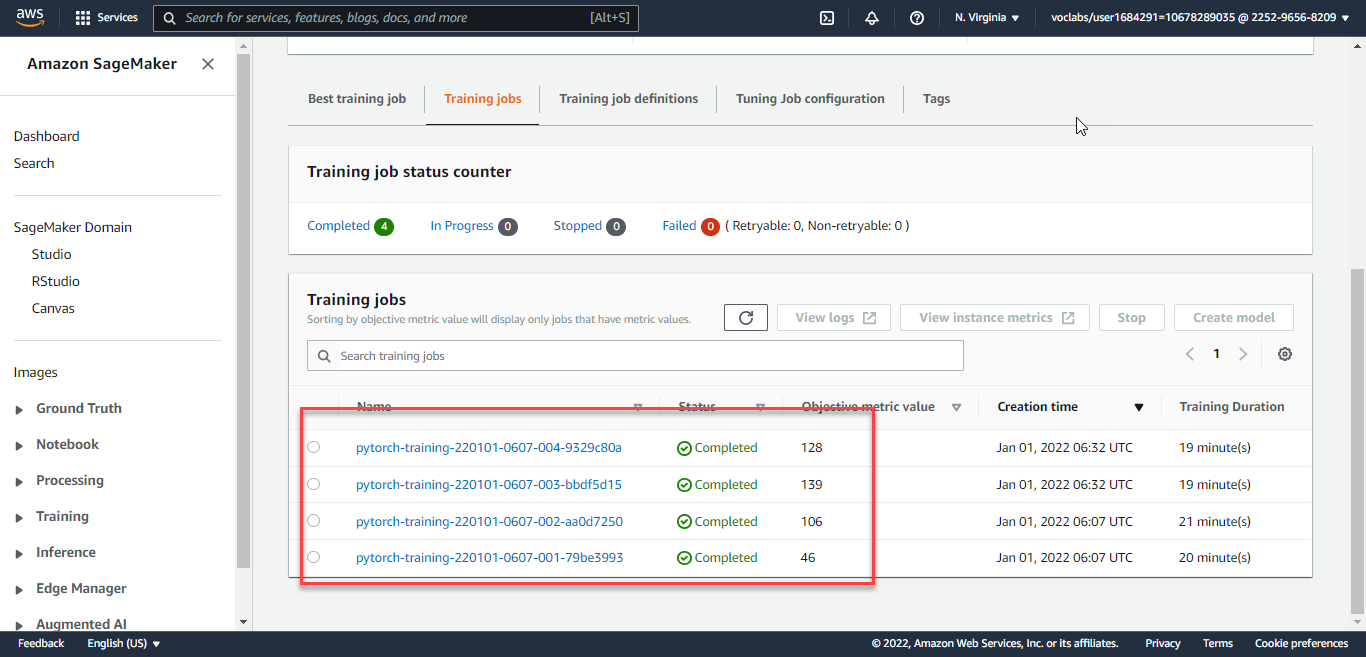
-------------—-----------------------Did the Training Again-----------------------------------------------------

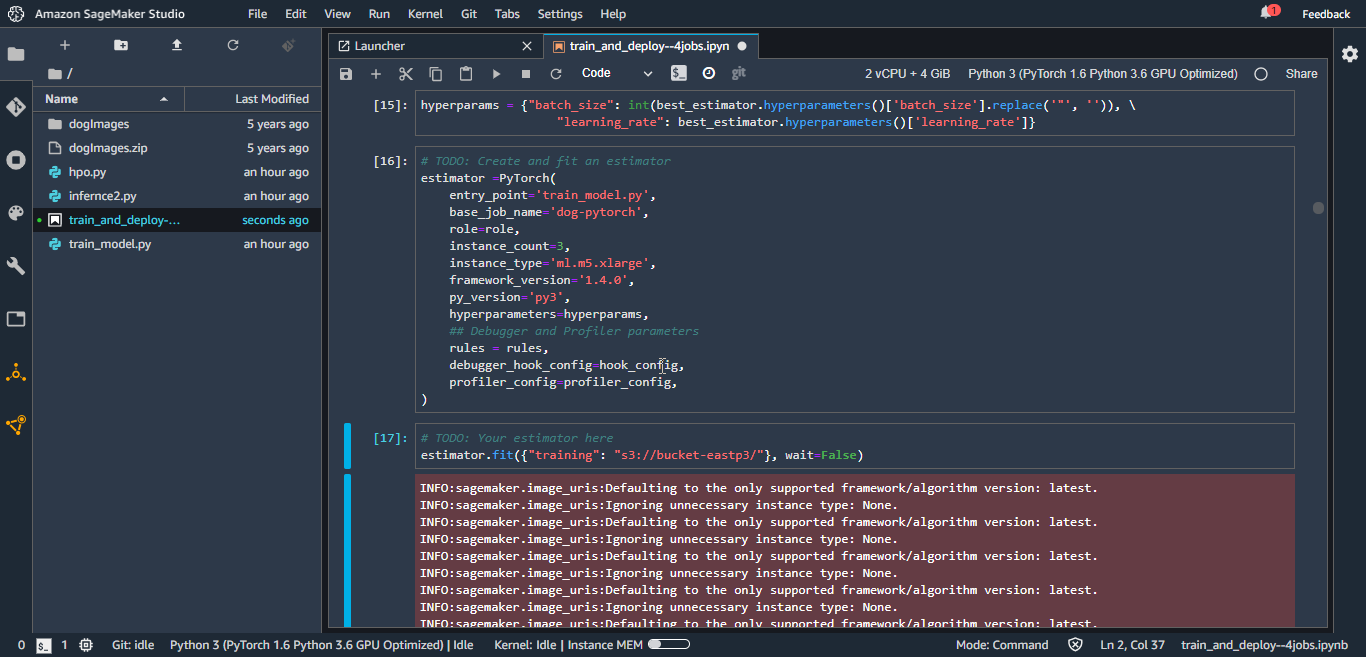
With the instruction of Mr. Peter L: <https://knowledge.udacity.com/questions/771244>

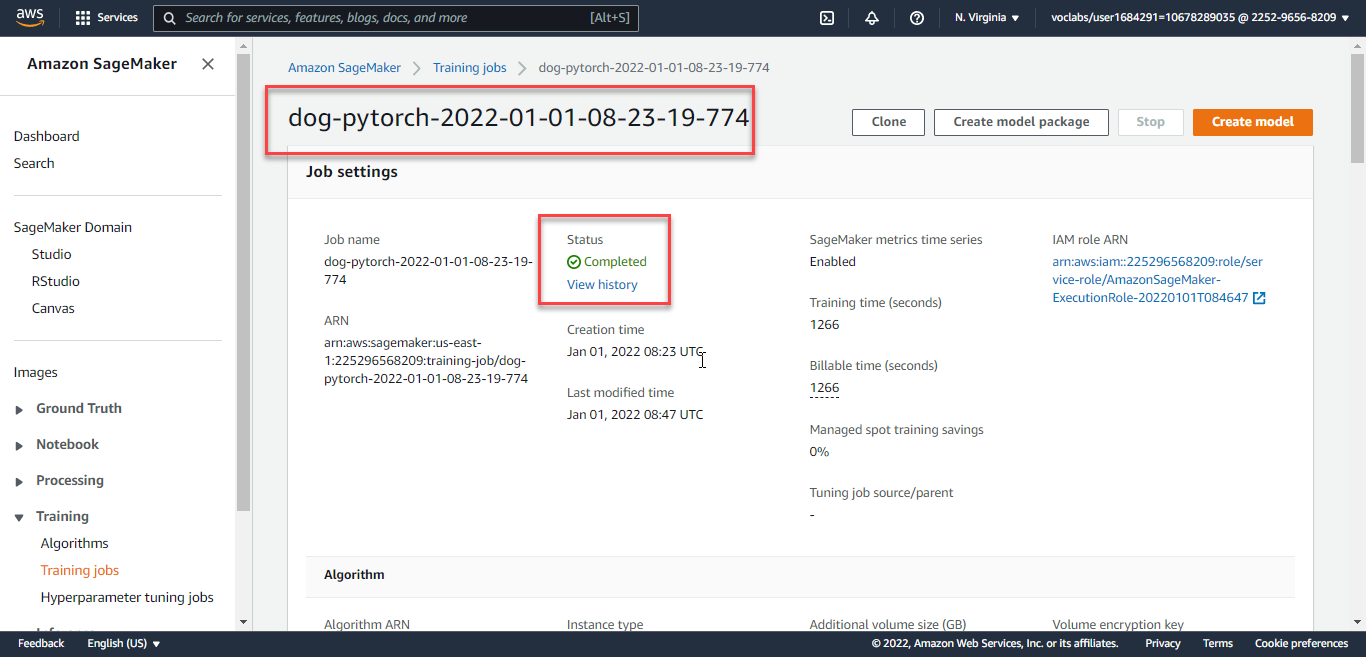
Added hook to python file entry point: <https://github.com/aws/amazon-sagemaker-examples/blob/master/sagemaker-debugger/pytorch_model_debugging/scripts/pytorch_mnist.py>

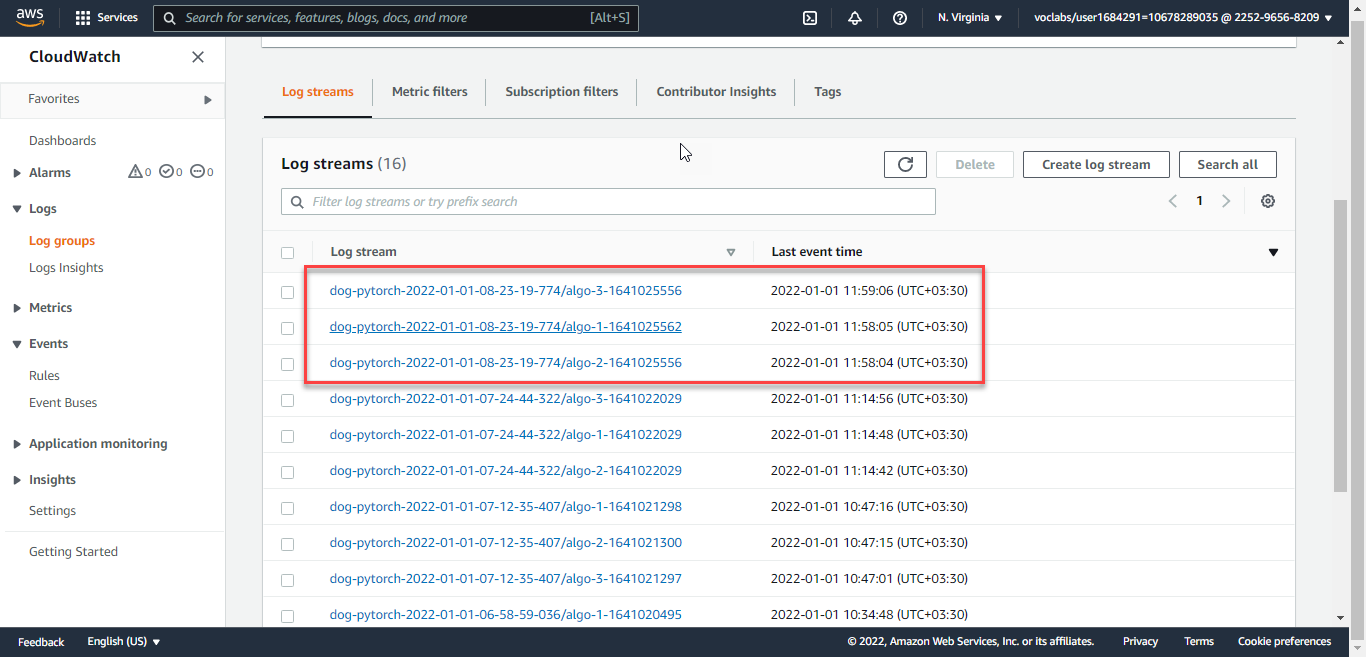
still an empty report….

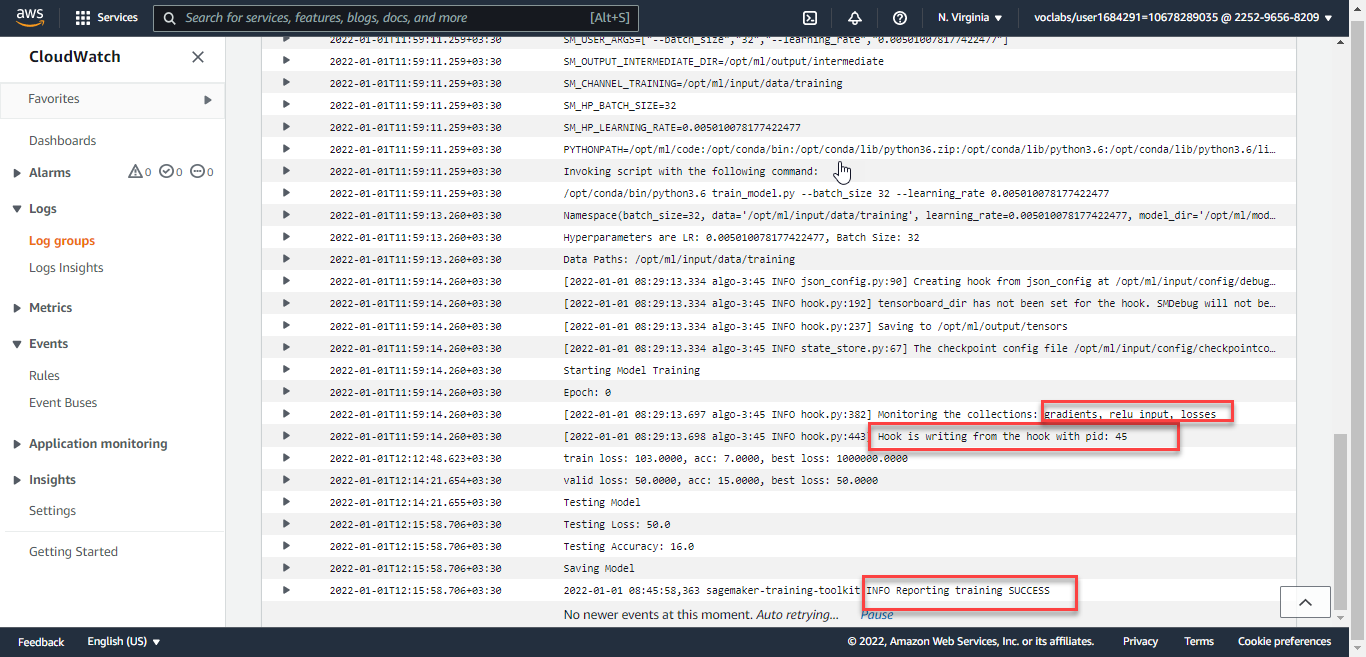


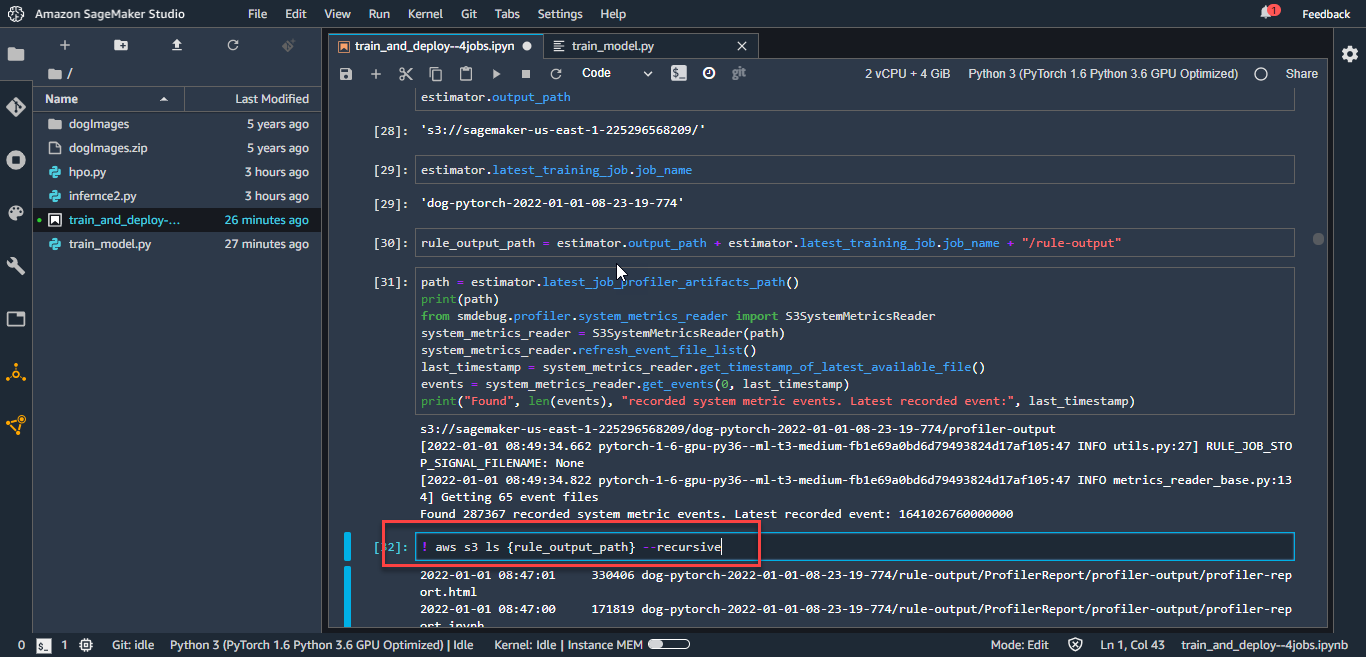






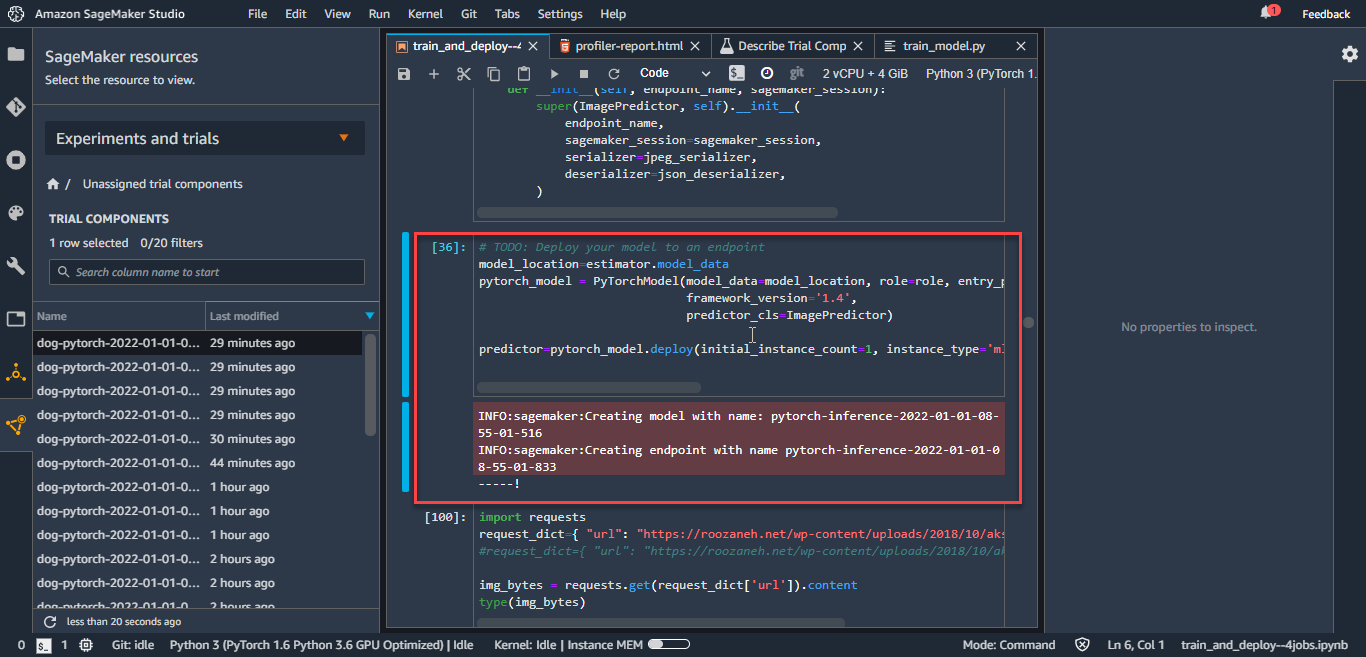


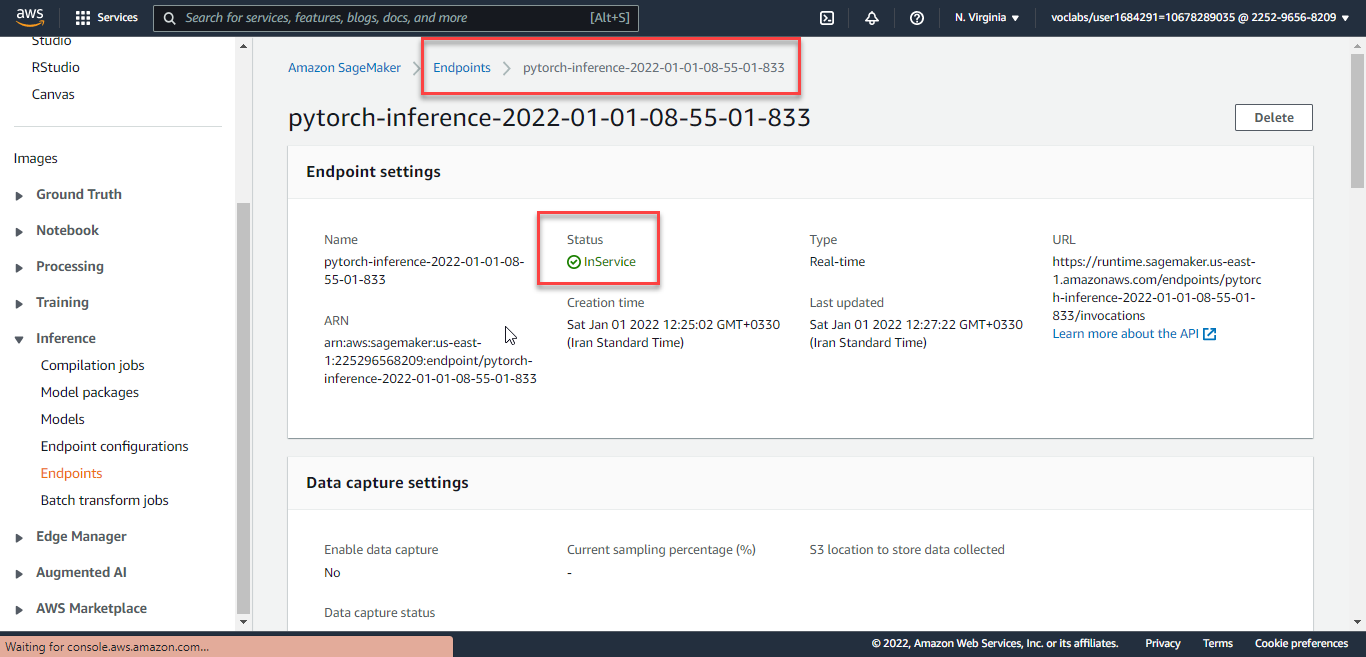






Deploying to endpoint:





Then for prediction use requests library and putting my image in dictionary with URL key, then with calling predict method on predictor , we get response NumPy array which has 133 classes probability of prediction, the index of maximum probability would be prediction of dog image classes…which is now dog breed 11…we can call endpoint many times and see the predictions…

**Standing Out (Packaging with docker):**

Since I used PyTorch library for DL, decided to use this reference: (<https://sagemaker-examples.readthedocs.io/en/latest/advanced_functionality/pytorch_extending_our_containers/pytorch_extending_our_containers.html> ) to Extend existing SageMaker PyTorch container and customize it with my code and environment. By extending the SageMaker PyTorch container we can utilize the existing training and hosting solution made to work on SageMaker.

Permissions:

Running this notebook requires permissions in addition to the normal SageMakerFullAccess permissions. This is because it creates new repositories in Amazon ECR. The easiest way to add these permissions is simply to add the managed policy AmazonEC2ContainerRegistryFullAccess to the role that you used to start your notebook instance. There’s no need to restart your notebook instance when you do this, the new permissions will be available immediately.

I use the following docker file for building image from base image:

"763104351884.dkr.ecr.$REGION.amazonaws.com/pytorch-inference-eia:1.3.1-cpu-py36-ubuntu16.04"

docker file:

ARG REGION=us-east-1

FROM 763104351884.dkr.ecr.$REGION.amazonaws.com/pytorch-inference-eia:1.3.1-cpu-py36-ubuntu16.04

ENV PATH="/opt/ml/code:${PATH}"

COPY /hpo.py /opt/ml/code

RUN pip3 install -r /opt/ml/code/requirements.txt

ENV SAGEMAKER\_SUBMIT\_DIRECTORY /opt/ml/code

ENV SAGEMAKER\_PROGRAM hpo.py

After creating the image and upload to amazon ECR registry , then can create container and do the training with our custom image packaged.