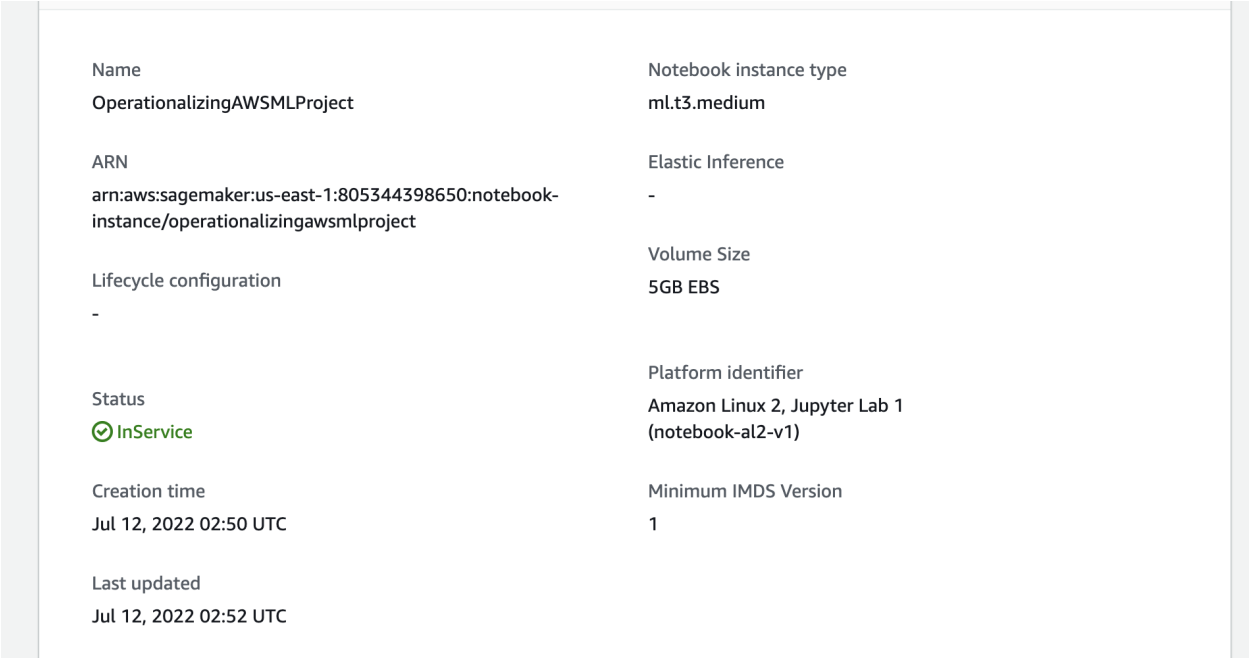
Operationalizing an AWS ML Project

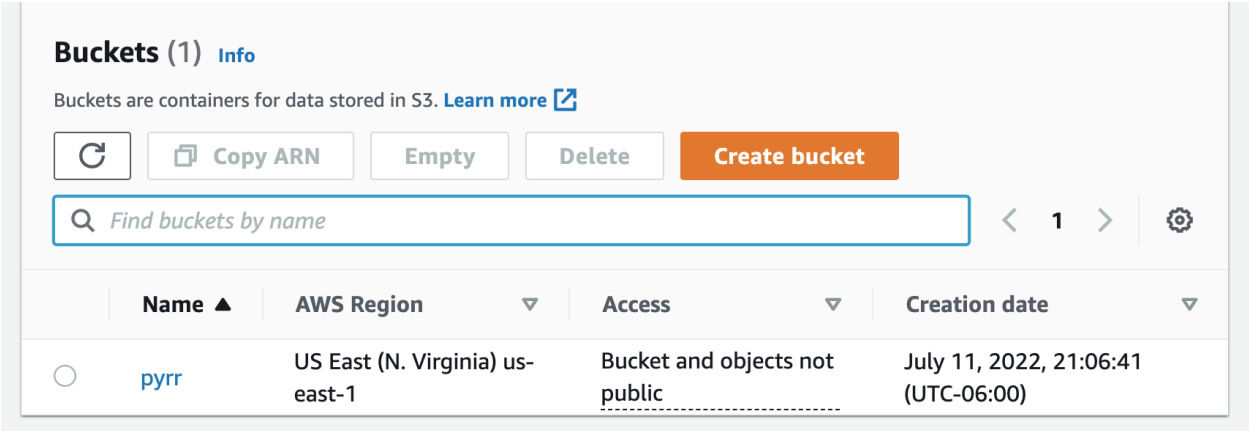
1. **Initial Set up**

Since this project does not require complex computing in the Notebook instance, considering to lower the cost, I decided to use "**ml.t3.medium**," which is also used popularly in past projects.



2. **Download data to an S3 bucket**

I've successfully set up an S3 bucket, pyrr, where I copy the data.



3. **Training and Deployment**

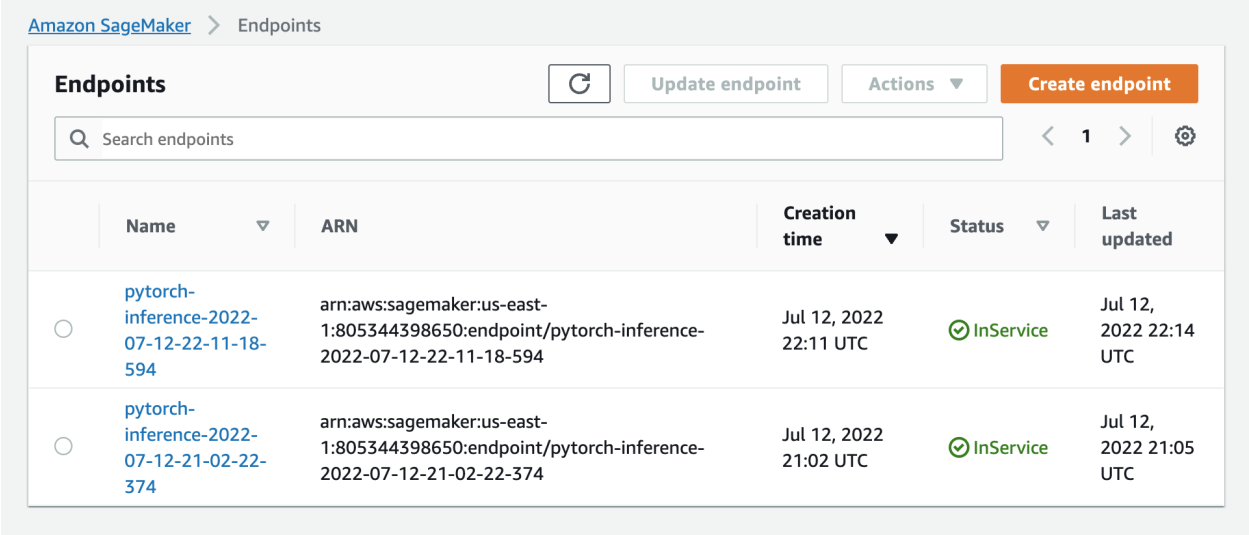
I deployed endpoints of both the single-instance training model and the multi-instance training model (for multi-instance model, I set parameter instance_count equal to 4)

Creating an Estimator - Multi-Instance Training,

In [33]:
```
###in this cell, create and fit an estimator using multi-instance training
multi_instance_estimator = PyTorch(
    entry_point='hpo.py',
    base_job_name='dog-pytorch',
    role=role,
    instance_count=4,
    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 [*]:
```
multi_instance_estimator.fit({"training": "s3://pyrr/"}, wait=True)
```

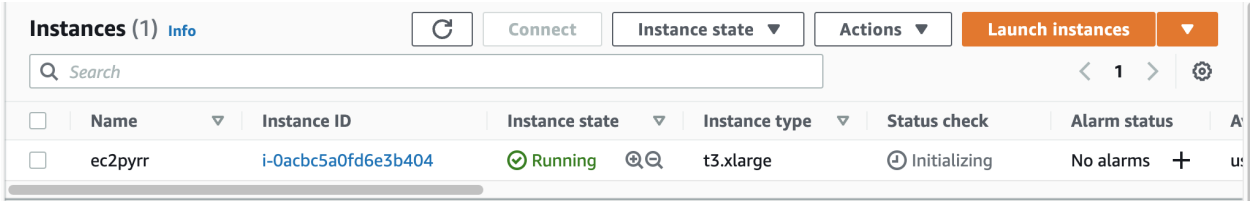
Then, I trained each estimator and deployed them respectively, and each endpoint successfully worked.



Tasks of Sagemakers part has been completed!

4. EC2

a. The type of AMI I choose for setting up my EC2 is **AWS Deep Learning AMI (Amazon Linux 2)**, with **t3.xlarge** instance. The reason I did not choose the recommended p3.2xlarge instances is because p3.2xlarge has 8 cpus and 61GB for memory, which would be a waste of resources for our project. The t3.xlarge instance has 4 cpus and 16GB memory, adequate for the project use. The AMI "**AWS Deep Learning AMI**" is also an popular choice for EC2 instance setting.



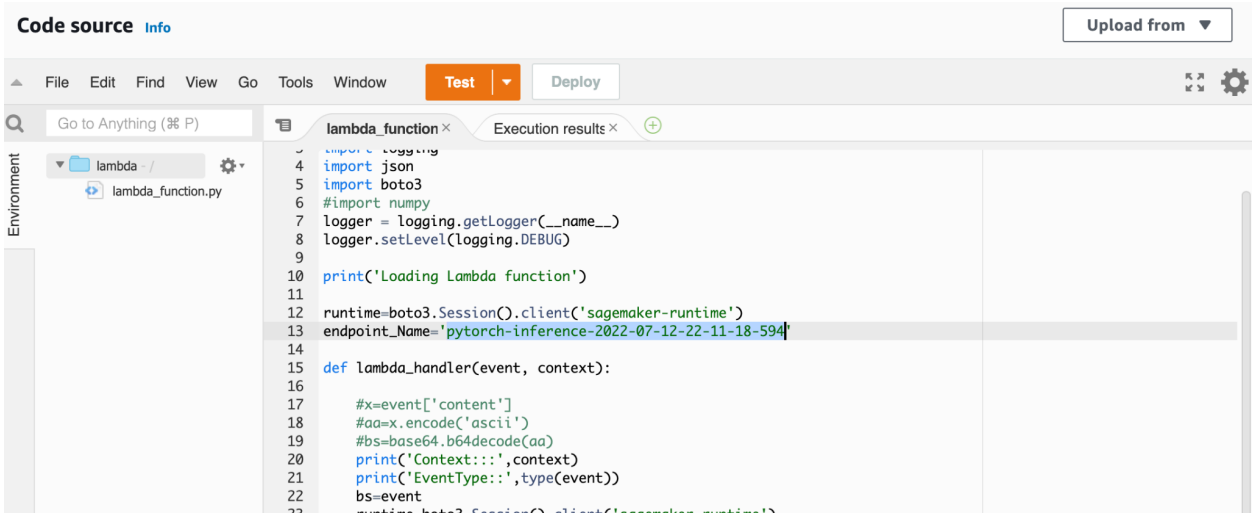
b. Comparing to the code from Sagemaker, the EC2 does not need manually tuning hyperparameters and setting up endpoints; instead, it does not take in any arguments, but

was completely operated within the EC2 instance. Also, considering the loading and saving the data, Sagemaker needs to save the data in certain S3 bucket, and then configure the right data path to train the model. However, training model in EC2 does not require us to specify the path but save the data within the current directory.

```
(pytorch_latest_p37) [root@ip-172-31-70-213 ~]# python3 solution.py
Downloading: "https://download.pytorch.org/models/resnet50-19c8e357.pth" to /root/.cache/torch/hub/checkpoints/resnet50-19c8e357.pth
100%|                                                                                | 97.8M/97.8M [00:00<00:00, 189MB/s
Starting Model Training
saved
(pytorch_latest_p37) [root@ip-172-31-70-213 ~]# ls
?  dogImages  dogImages.zip  solution.py  TrainedModels
(pytorch_latest_p37) [root@ip-172-31-70-213 ~]# cd TrainedModels/
(pytorch_latest_p37) [root@ip-172-31-70-213 TrainedModels]# ls
model.pth
(pytorch_latest_p37) [root@ip-172-31-70-213 TrainedModels]#
```

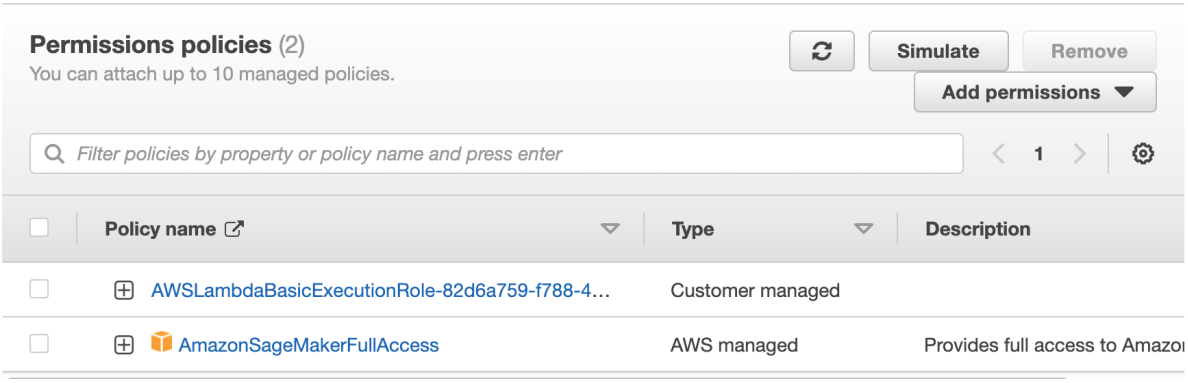
5. Lambda Function

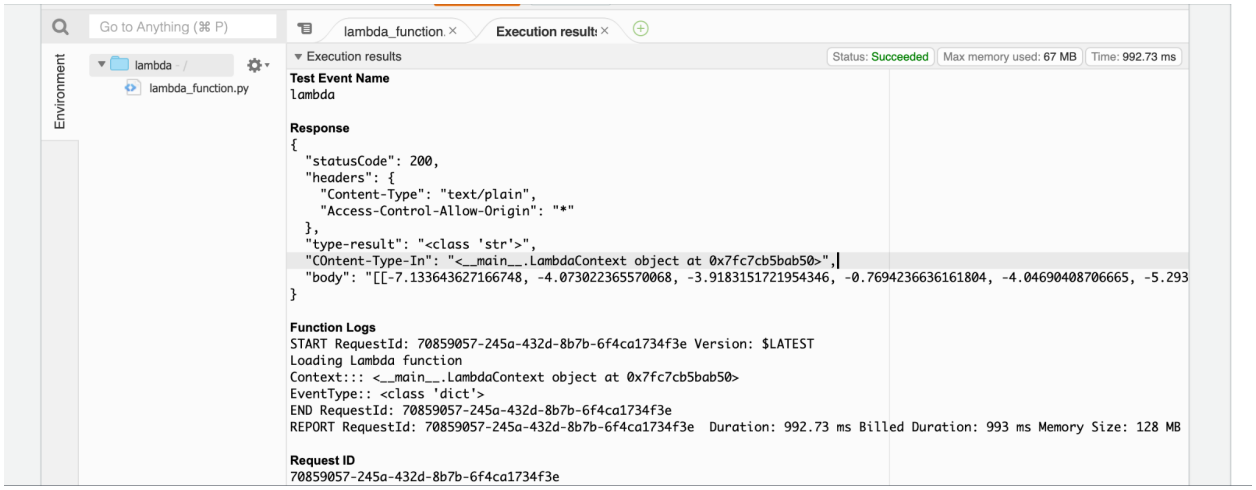
We could interact with the model through lambda functions to invoke deployed points. The endpoint I used is the multi-instance inference endpoint, “pytorch-inference-2022-07-12-22-11-18-594;” then, the lambda function will initiate a boto3.client session to invoke the endpoints. In this way, as long as the test input format is correct (.json), the lambda function would successfully invoke the multi-instance endpoint.



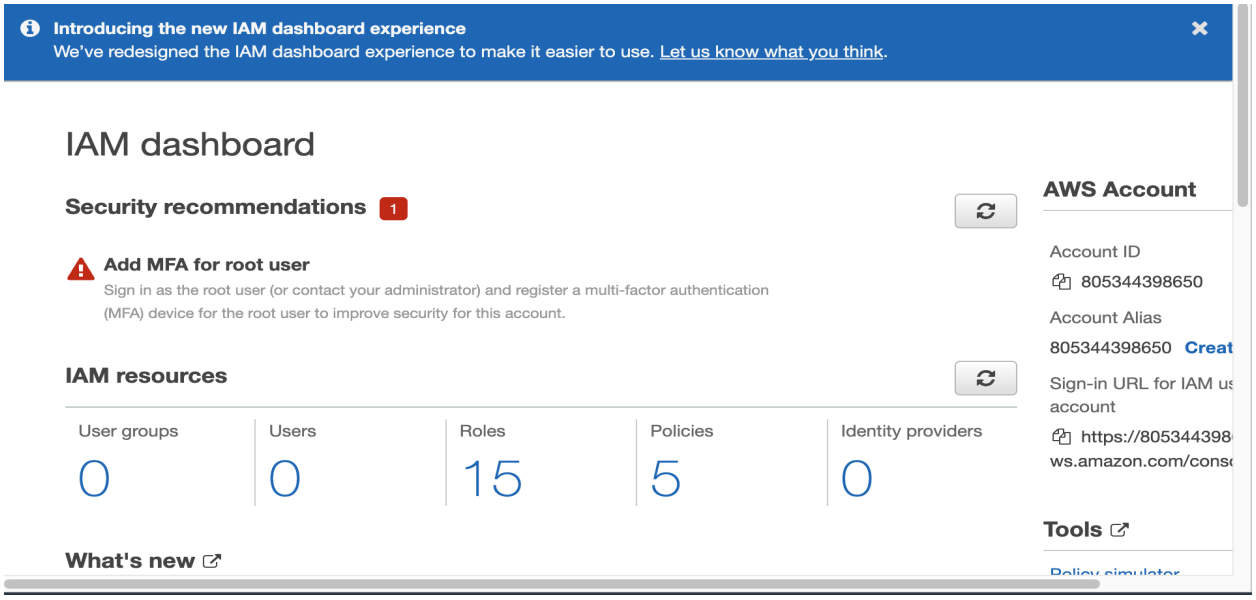
6. Security and Testing

After we attached the right policy term to our function, the expected outputs were generated. The test results, an 133-element array, are stored as values of the key Body. The images with testing outputs are attached below.



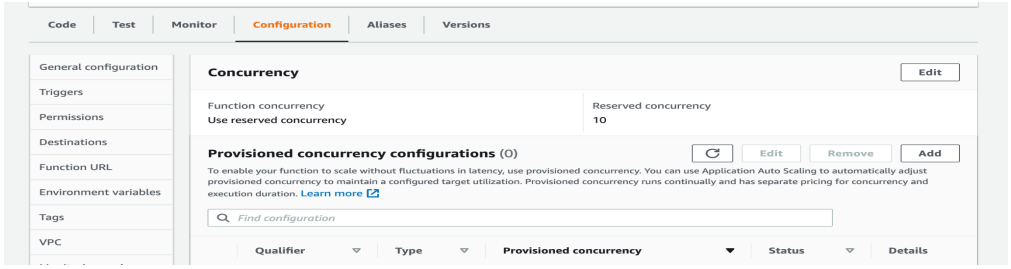


The IAM dashboard image is attached below, and there does exist several security concerns for our AWS workspace.



First, the "FullAccess" policies attached to roles such as lambda functions may be too permissive, and people could gain the full permissions to access the Sagemaker code through lambda functions, instead of the only endpoint we deployed, this set-up would be not secure and might cause problems. Also, we do not add MFA for root user, the employment of MFA could improve the security of account. Third, we should track the active roles, which means the inactive roles should not be given permissive policy accesses and perhaps should be deleted.

7. **Configuration of Concurrency and Auto-scaling**



The reserved concurrency is adequate for our project, since a large number of requests of lambda functions is not very likely, but I still set the maximum to 10. Also, I published a new version for our lambda functions.

Then, navigate to the Sagemaker and find the endpoint I deployed, and then click “configure auto-scaling,” we could set up auto-scaling for the endpoint. I choose the Maximum instance count to be 5, and 30 seconds for both scale in cool down and scale out cool down time.