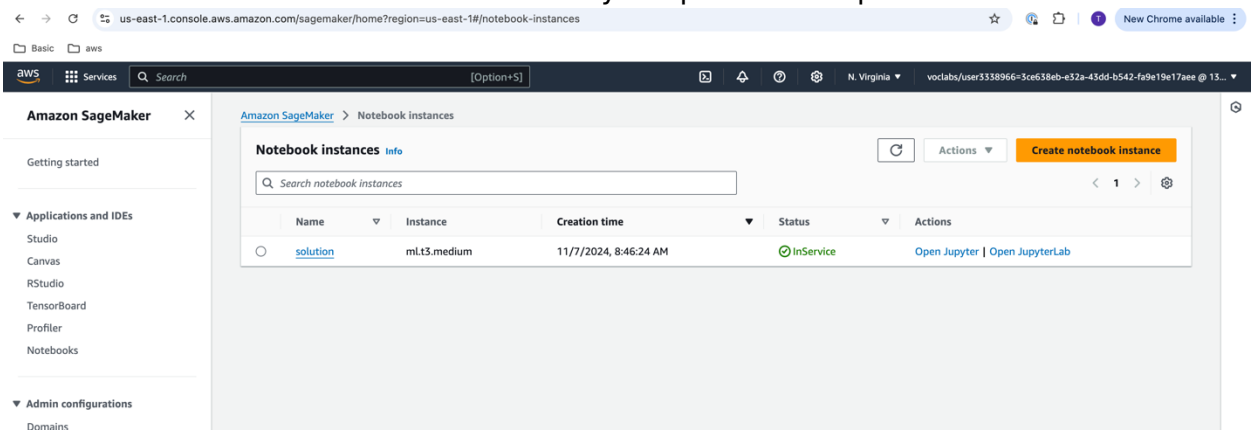
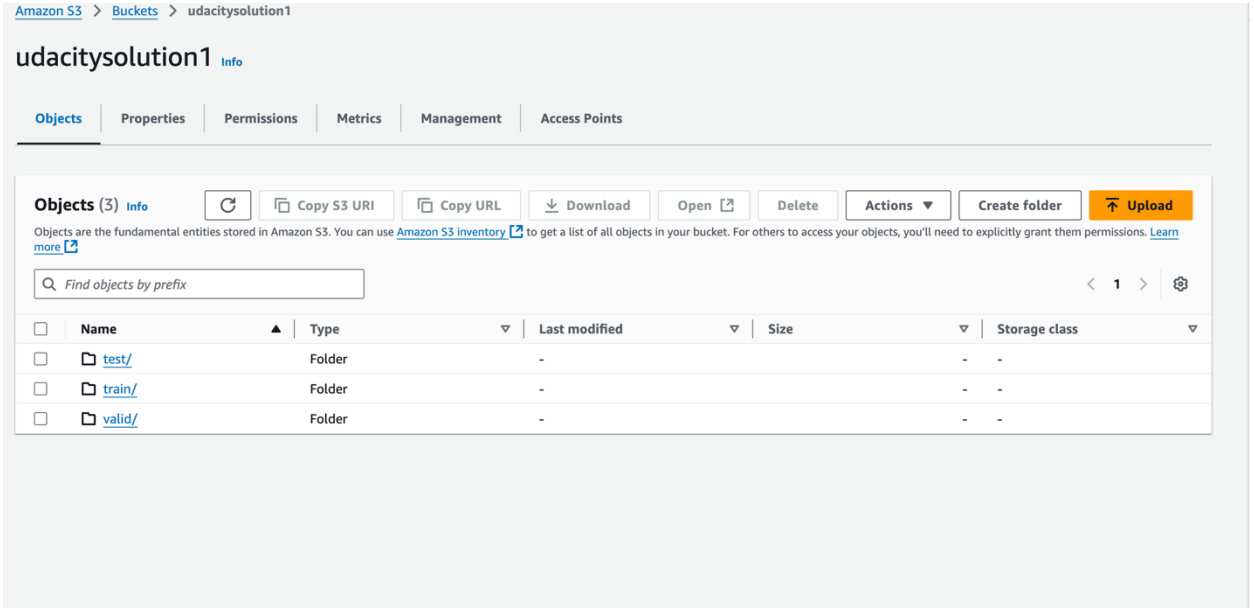
Step 1: Training and deployment on Sagemaker

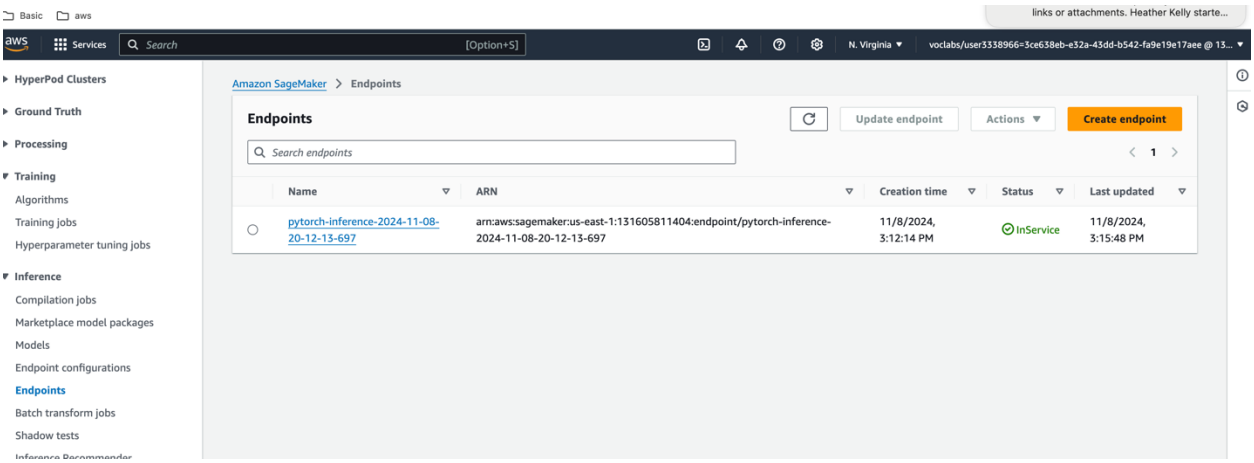
Created a Notebook instance with instance type ml.t3.medium which I felt will be sufficient which has 2 CPU and 4GB memory at a price of 0.05 per hour



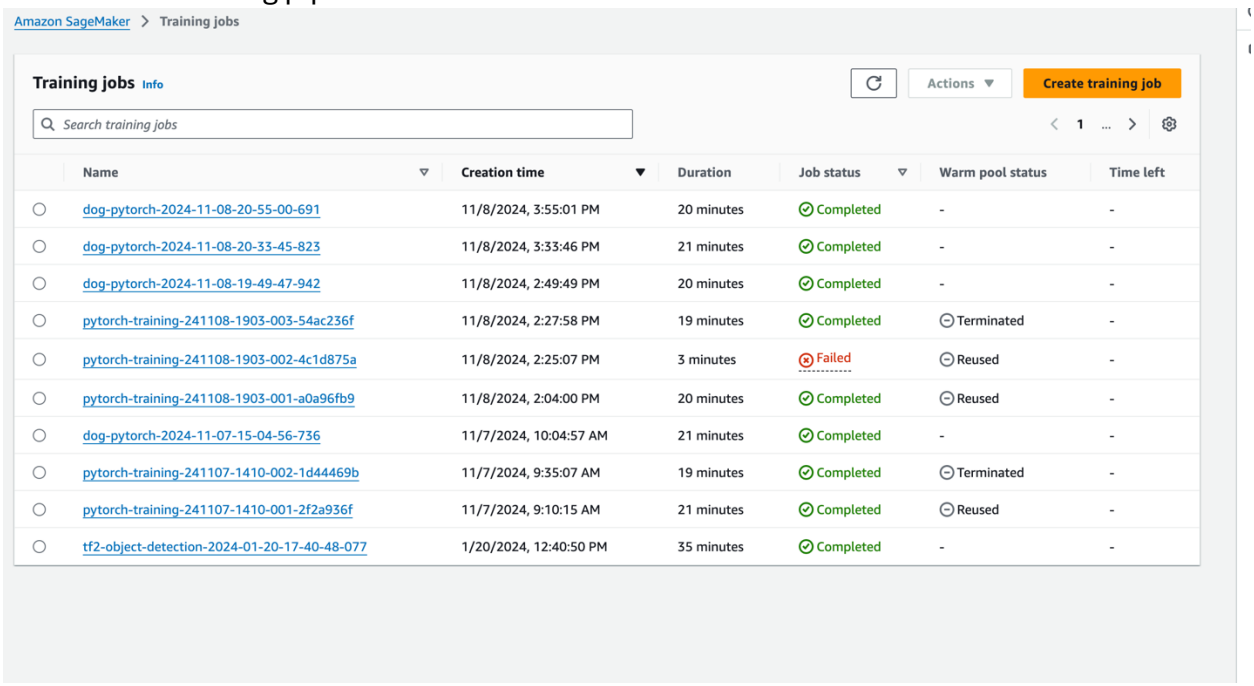
I am able to create a s3 bucket and upload the downloaded dogClassification data to the bucket



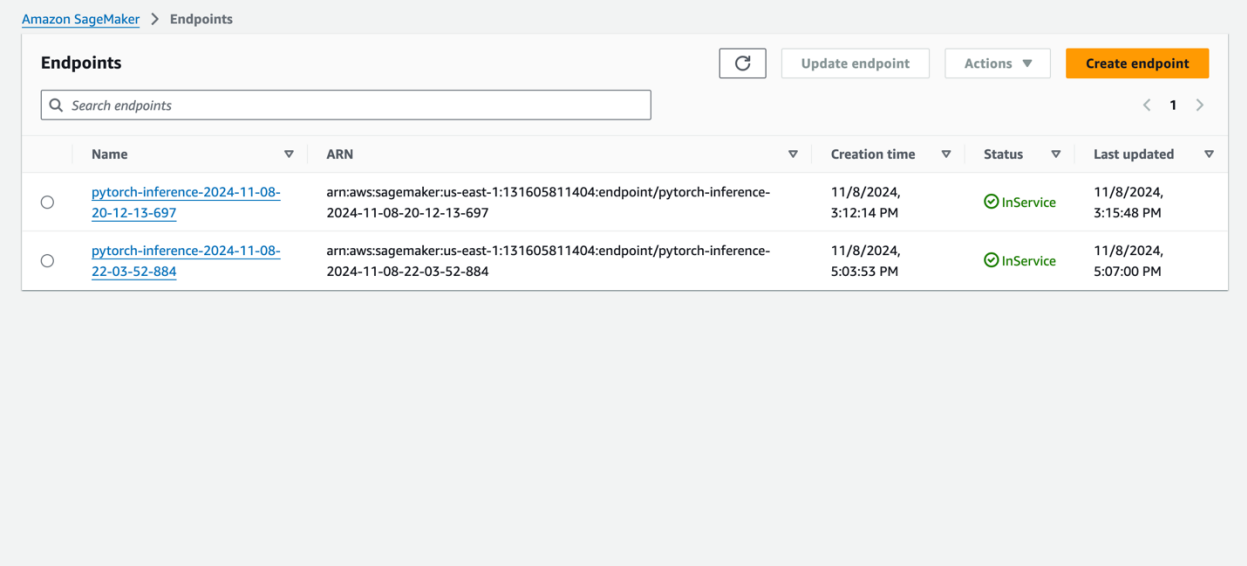
I was able to deploy the endpoint with training them on single instance



Some of the training pipelines

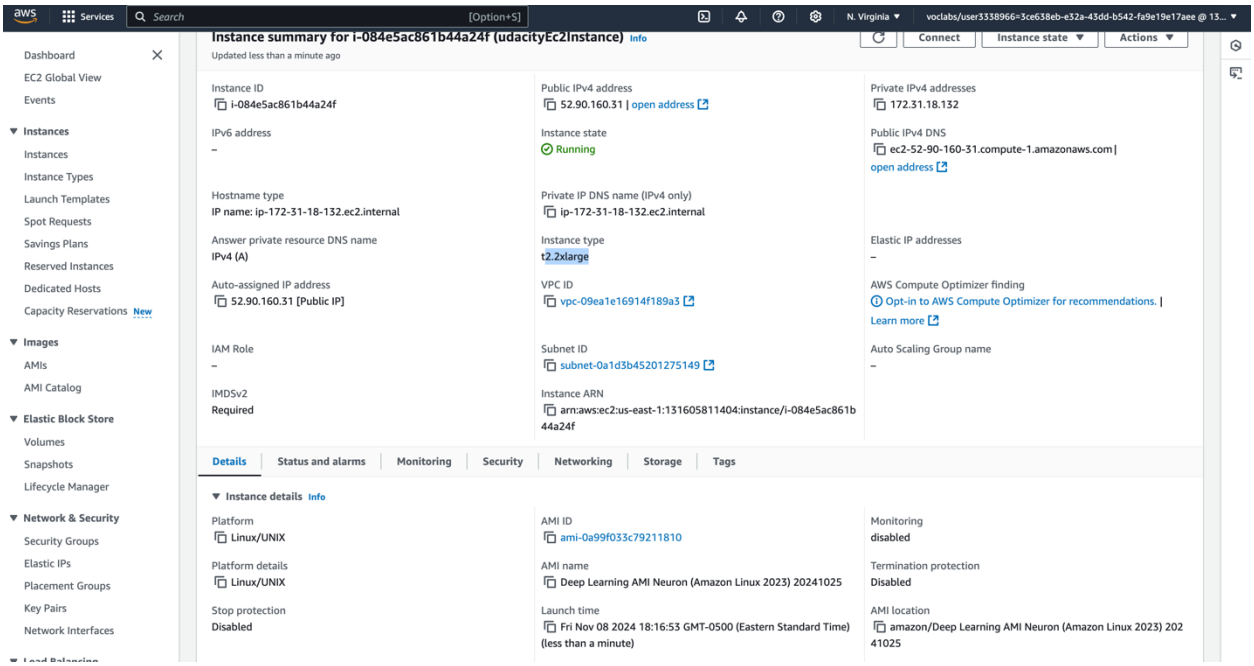


I have trained the model using multiple instances and deployed the endpoint



Step 2: EC2 Training

I create an EC2 instance with instance type t**2.2xlarge** image **Deep Learning AMI Neuron (Amazon Linux 2023) 20241025**



The reason for choosing the 2x large so It has space for all the modules installation and also the compute power for training the model

```
[root@ip-172-31-18-132 ~]# python3 solution.py
/usr/local/lib64/python3.9/site-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the future, please
 use 'weights' instead.
  warnings.warn(
/usr/local/lib64/python3.9/site-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or `None` for 'weights' are deprecated since 0.13 and may
 be removed in the future. The current behavior is equivalent to passing `weights=ResNet50_Weights.IMAGENET1K_V1`. You can also use `weights=ResNet50_Weights.DEFAULT` to get the mo
st up-to-date weights.
  warnings.warn(msg)
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, 117MB/s]
Starting Model Training
saved
[root@ip-172-31-18-132 ~]#
[root@ip-172-31-18-132 ~]# ls
TrainedModels  dogImages  dogImages.zip  solution.py
[root@ip-172-31-18-132 ~]# cd TrainedModels/
[root@ip-172-31-18-132 TrainedModels]# ls
model.pth
[root@ip-172-31-18-132 TrainedModels]#
```

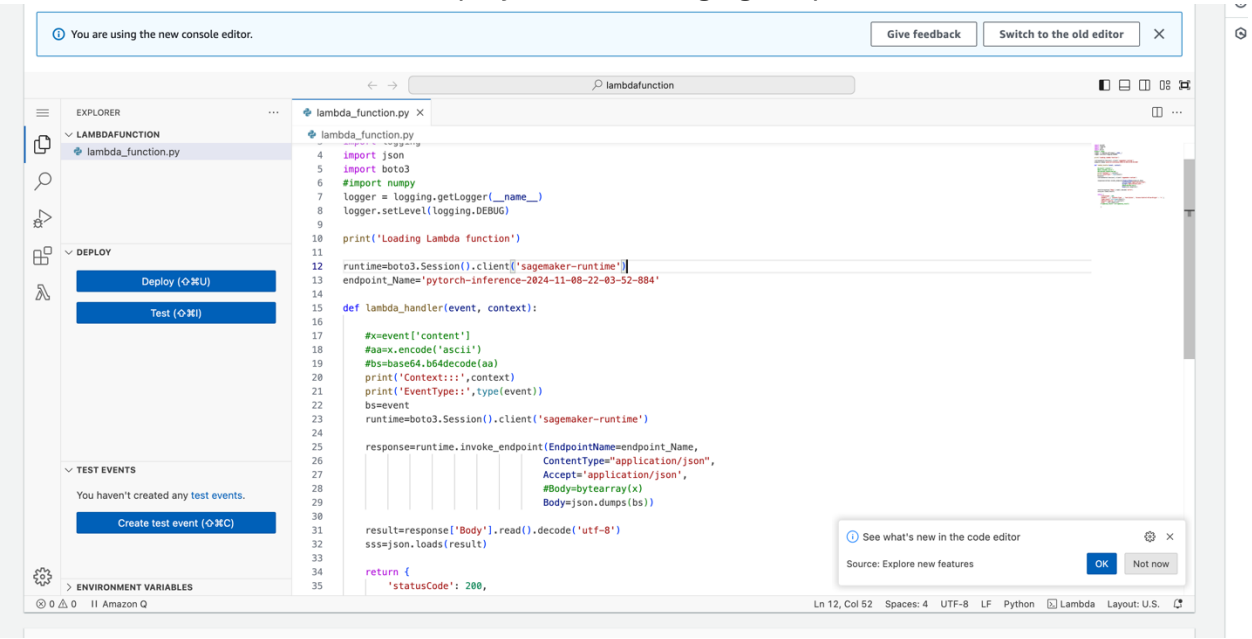
i-084e5ac861b44a24f (udacityEc2Instance)

PublicIPs: 52.90.160.31 PrivateIPs: 172.31.18.132

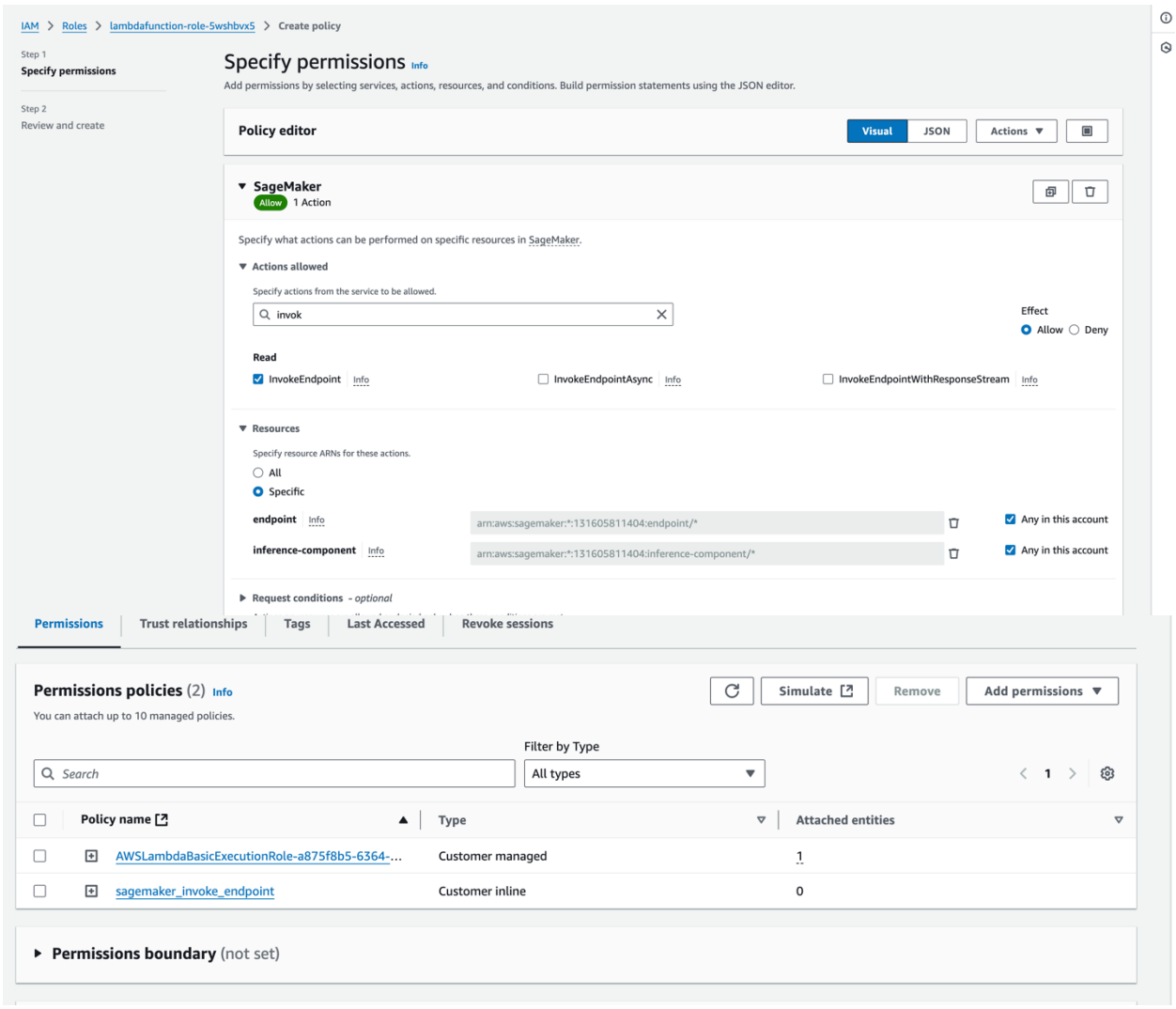
The trained model is saved to the Trained Model folder
The code ec2_train .py is almost similar to solution.ipny with same change in modules where solution.ipny has most modules related to sagemaker and the configuration related to sagemaker

Step 3: Lambda function setup

The lambda function has been deployed after changing endpoint



Step 4: Security and testing
I have added policy to the role of the lambda so that lambda can invoke the endpoint



The result of trigerring lambda is


```
{
  "statusCode": 200,
  "headers": {
    "Content-Type": "text/plain",
    "Access-Control-Allow-Origin": "*"
  },
  "type-result": "<class 'str'>",
  "COntent-Type-In": "<__main__.LambdaContext object at 0x7fab0ce3fbe0>",
  "body": "[[-18.778697967529297, -12.238218307495117, -5.308145046234131, -
7.89225435256958, -8.937215805053711, -10.688363075256348, -6.234222888946533, -
6.378856658935547, -9.672450065612793, -2.4265635013580322, -0.5124581456184387, -
9.953339576721191, -4.059317588806152, 0.5916768312454224, -9.475948333740234, -
6.651721000671387, -17.350299835205078, -4.182865142822266, -11.536008834838867, -
1.5100040435791016, -7.54235315322876, -4.557830810546875, -18.007892608642578, -
16.71207046508789, -12.820658683776855, -15.838006973266602, -5.058803558349609, -
4.795527458190918, -12.920251846313477, -4.0984625816345215, -8.94543170928955, -
```

```
10.278924942016602, -12.265420913696289, -9.051471710205078, -11.998316764831543, -
12.272721290588379, -8.479098320007324, -8.839516639709473, -5.938298225402832, -
13.204375267028809, -8.606587409973145, -8.593269348144531, -3.1887593269348145, -
8.698216438293457, -4.468384265899658, -17.074504852294922, -6.402591705322266, -
3.553925037384033, -7.13292121887207, -5.517165184020996, -7.332523822784424, -
17.604164123535156, -11.837915420532227, -6.383935451507568, -8.944275856018066, -
5.059283256530762, -11.892801284790039, -13.640637397766113, -7.336037635803223, -
7.442246913909912, -12.98865032196045, -13.984745979309082, -16.092927932739258, -
18.646669387817383, -8.718060493469238, -13.956052780151367, -3.1447694301605225, -
9.287219047546387, -8.4979829788208, -5.886133670806885, -1.5280661582946777, -
9.075248718261719, -9.046012878417969, -11.540383338928223, -9.35733413696289, -
7.213979721069336, -13.234991073608398, -8.41640567779541, -14.476627349853516, -
11.949639320373535, -2.8843283653259277, -16.363872528076172, -3.945138454437256, -
4.844686508178711, -11.802663803100586, -9.587656021118164, -5.377264976501465, -
14.948714256286621, -6.3733930587768555, -5.787425994873047, -18.488615036010742, -
10.916004180908203, -12.297137260437012, -15.696701049804688, -12.077312469482422, -
5.768967151641846, -7.77634859085083, -6.686862945556641, -14.79699420928955, -
15.238365173339844, -14.860189437866211, -6.409668445587158, -6.675621032714844, -
12.182988166809082, -11.968731880187988, -14.455435752868652, -7.31479549407959, -
5.257838726043701, -3.034386396408081, -0.9662284851074219, -4.283210754394531, -
4.098550319671631, -19.749507904052734, -8.340622901916504, -11.919856071472168, -
4.4515299797058105, -16.721647262573242, -4.307643890380859, -12.503721237182617, -
4.501818656921387, -4.698922634124756, -7.755297660827637, -7.737709045410156, -
10.31639575958252, -16.912992477416992, -12.368294715881348, -7.440530300140381, -
3.266435146331787, -11.047531127929688, -11.450682640075684, -13.404930114746094, -
6.659083366394043, -9.455876350402832]]"
}
```

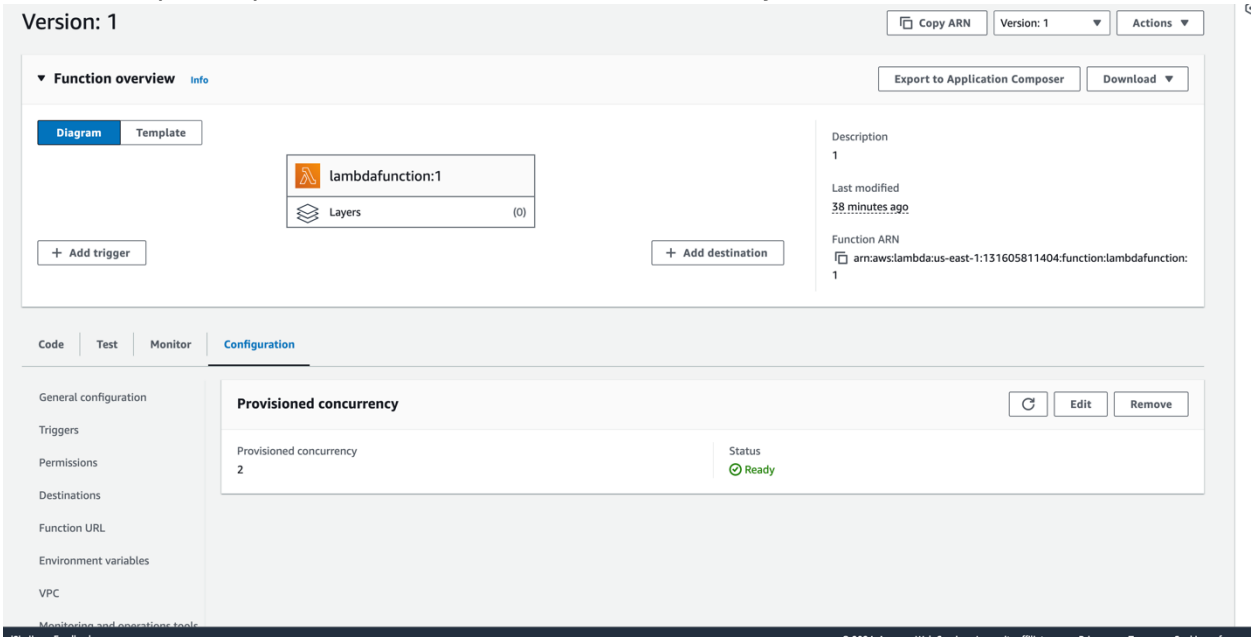

Logs

Loading Lambda function
START RequestId: e13d55fa-4a71-48ca-a470-c74e8c459356 Version: $LATEST
Context::: <__main__.LambdaContext object at 0x7fab0ce3fbe0>
EventType:: <class 'dict'>
END RequestId: e13d55fa-4a71-48ca-a470-c74e8c459356
REPORT RequestId: e13d55fa-4a71-48ca-a470-c74e8c459356 Duration: 1729.34 ms
Billed Duration: 1730 ms Memory Size: 128 MB Max Memory Used: 81 MB
Init Duration: 488.20 ms

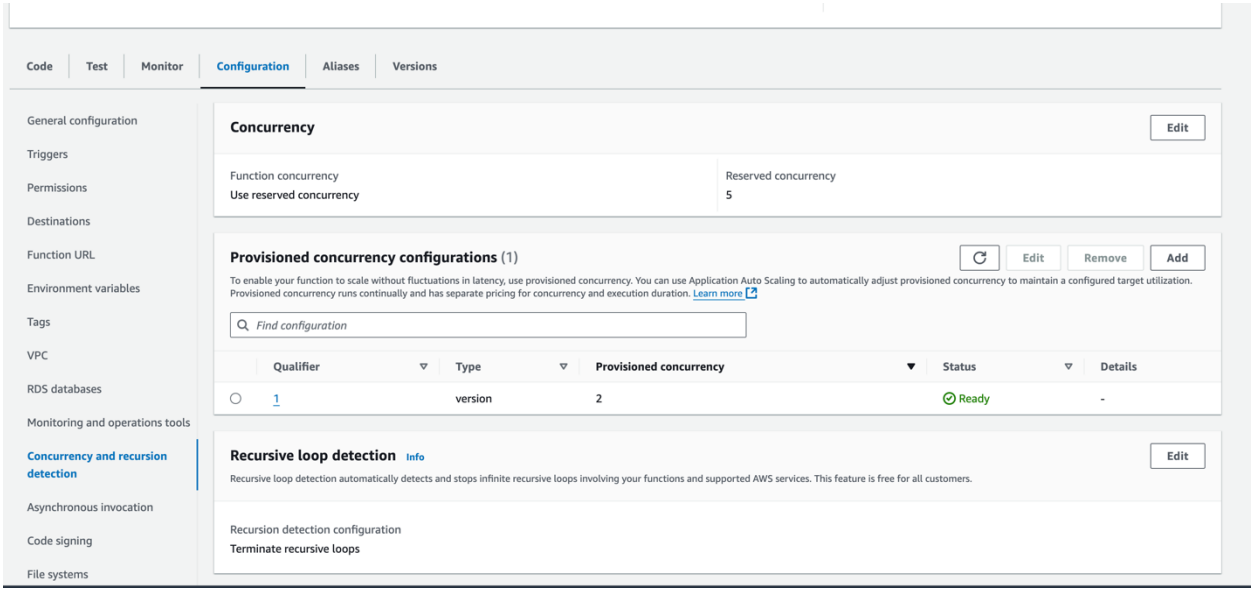
There are vulnerabilities to the lambda as this can be accessed for any where. To add more security we can add the vpn to the lambda so that lambda can be accessed based on security group inbound and outbound rules .

Step 5: Concurrency and auto-scaling

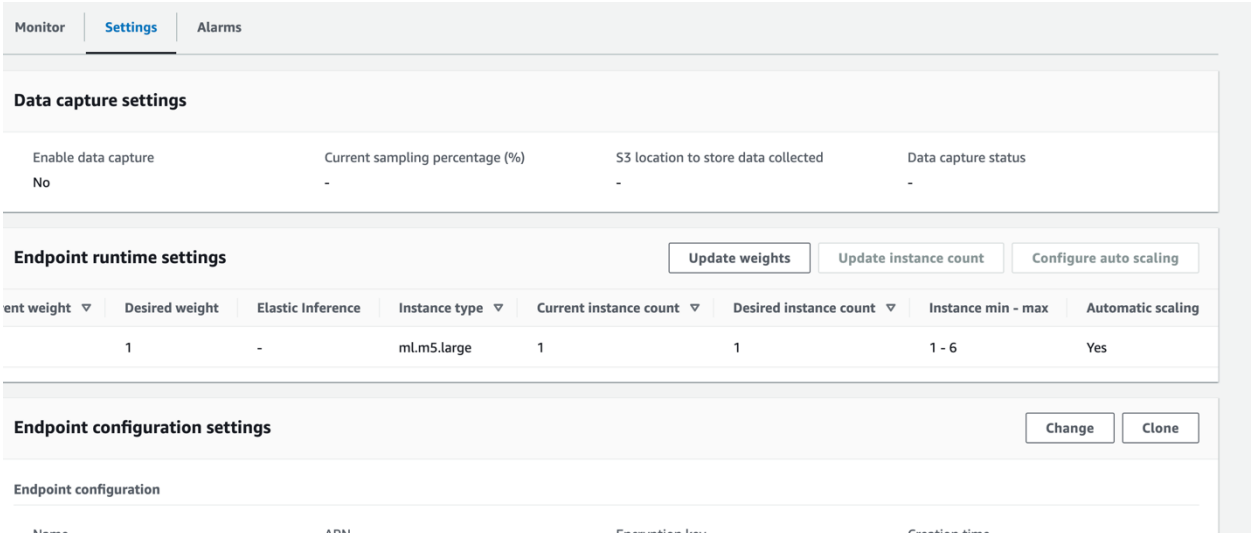
I have set up both provisioned and reserved concurrency for the lambda



Provisioned concurrency of 2 and reserved concurrency of 5



I have set up autoscaling of 6 and cooldown period of scale out and scale in is set to 300 sec


Monitor
Settings
Alarms
Data capture settings
Enable data capture
No
Current sampling percentage (%)
-
S3 location to store data collected
-
Data capture status
-
Endpoint runtime settings
Update weights
Update instance count
Configure auto scaling
ent weight
Desired weight
Elastic Inference
Instance type
Current instance count
Desired instance count
Instance min - max
Automatic scaling
1
-
ml.m5.large
1
1
1 - 6
Yes
Endpoint configuration settings
Change
Clone
Endpoint configuration