Operationalizing-an-AWS-ML-Project

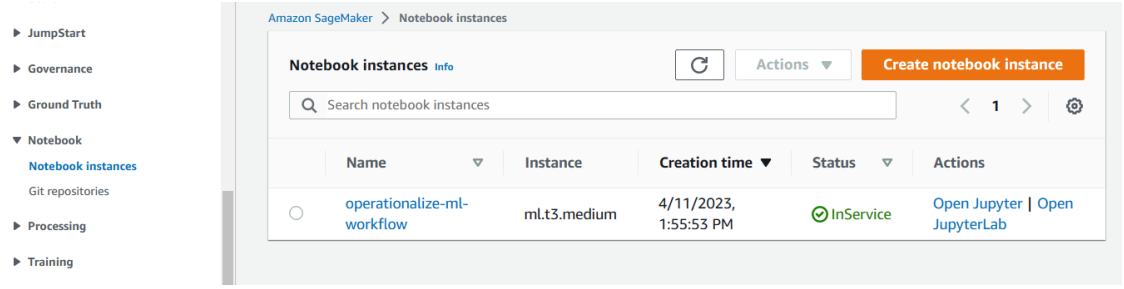
Dog Image Classification

In this project, you will complete the following steps:

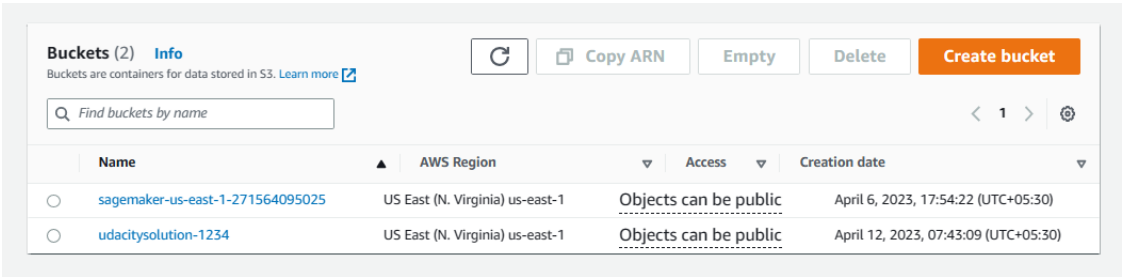
1. Train and deploy a model on Sagemaker, using the most appropriate instances. Set up multi-instance training in your Sagemaker notebook.
2. Adjust your Sagemaker notebooks to perform training and deployment on EC2.
3. Set up a Lambda function for your deployed model. Set up auto-scaling for your deployed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

Step 1: Training and deployment on Sagemaker

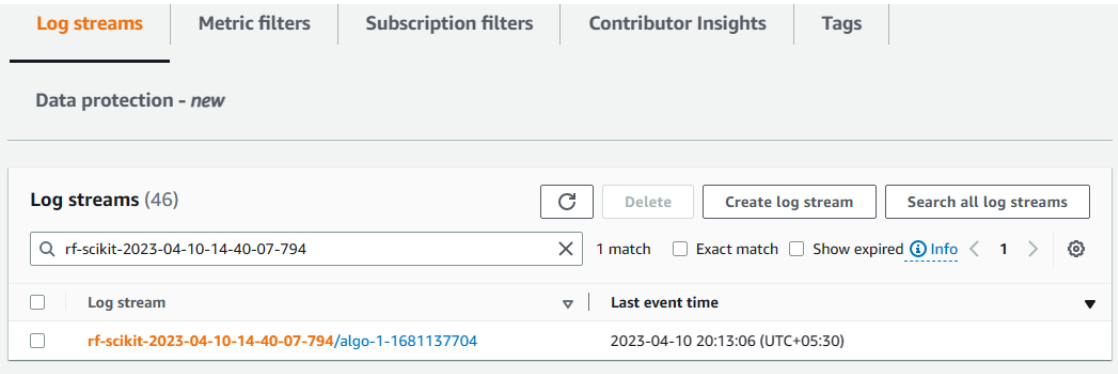
- **Created Sagemaker notebook instance** I have used ml.t3.medium as this is sufficient to run my notebook.



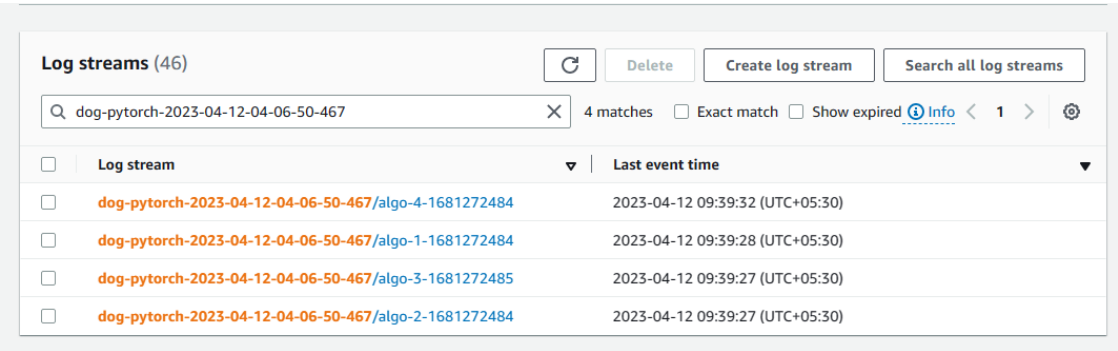
- **S3 bucket for the job** (udacitysolution-1234)



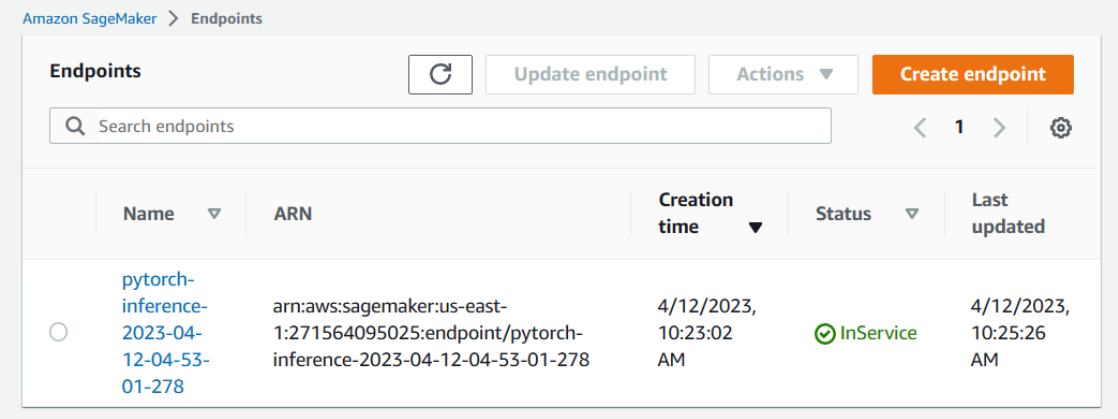
- **Single instance training** (1 epoch because of budget constraints)



- **Multi-instance training** (4 instances, 1 epoch because of budget constraints)

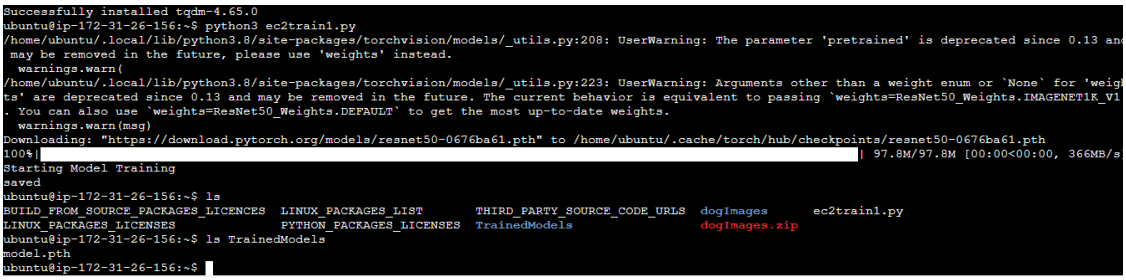


- **Deployment**



Step 2: EC2 Training

We can train model on EC2 instance as well. I chose AMI with required library already installed. Deep Learning AMI GPU PyTorch 2.0.0 has latest PyTorch version. Instance type selected was m5.xlarge because to low cost



The above image shows the EC2 instance and the terminal running the **ec2train1.py** script for training the model.

The adjusted code in ec2train1.py is very similar to the code in train_and_deploy-solution.ipynb. But there are few differences between the modules used - some modules can only be used in SageMaker. Much of the EC2 training code has also been adapted from the functions defined in the hpo.py starter script. Ec2train.py trains model with specific arguments while hpo.py takes argument for model by parsing through command line. The later code can train multiple model with different hyper parameters.

Step 3: Step 3: Lambda function setup

After training and deploying your model, setting up a Lambda function is an important next step. Lambda functions enable your model and its inferences to be accessed by API's and other programs, so it's a crucial part of production deployment.

Step 4: Lambda security setup and testing

- **Adding endpoints permission to lambda functions**
  Lambda function is going to invoke deployed endpoint. However, the lambda function will only be able to invoke endpoint if it has the proper security policies attached to it.
  Two security policy has been attached to the role:

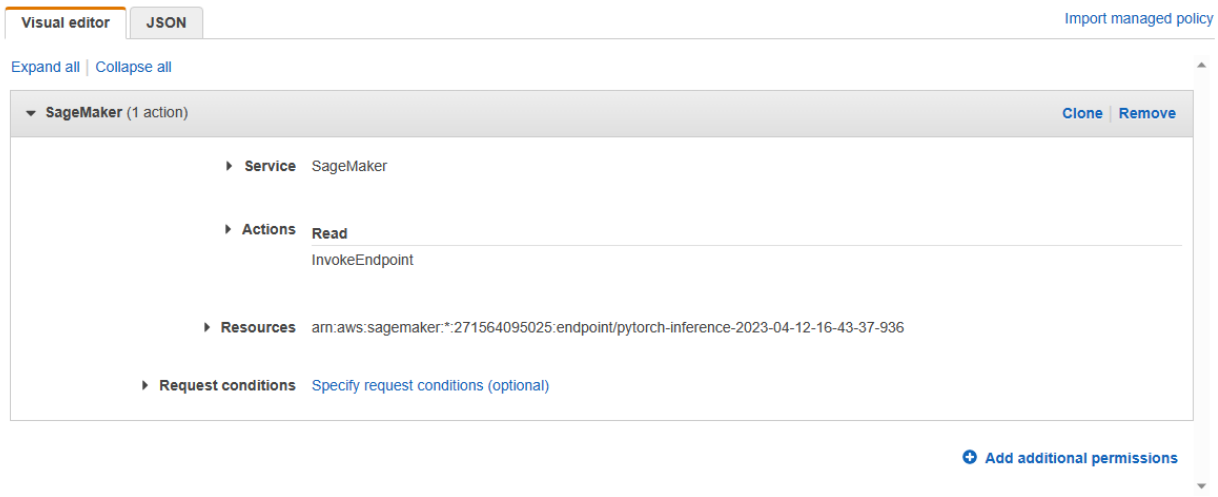
1. Basic Lambda function execution
2. Sagemaker endpoint invocation permission

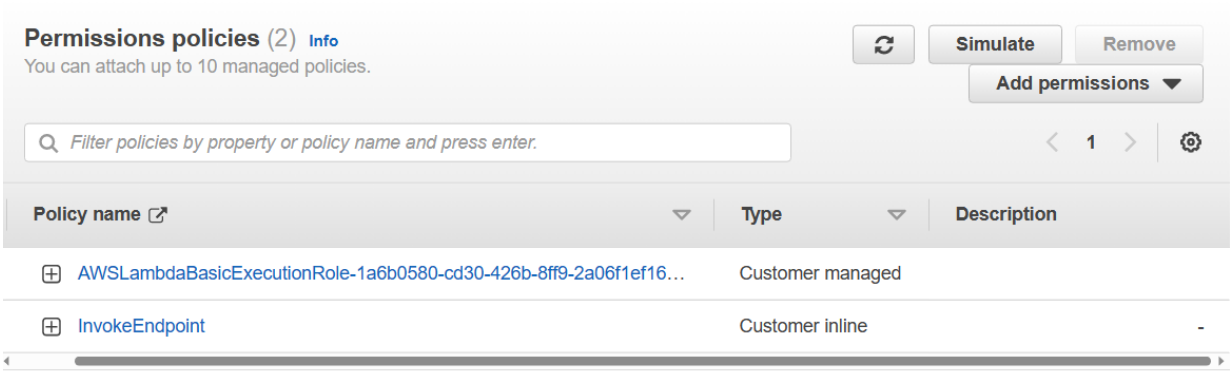
Vulnerability Assessment

- Giving 'Full Access' has potential to be exploited by malicious actor.
- Old and inactive roles are at the risk to compromise lambda function. These roles should be deleted.
- Roles with policies no longer in use has potential of unauthorized access. These policies should be removed.

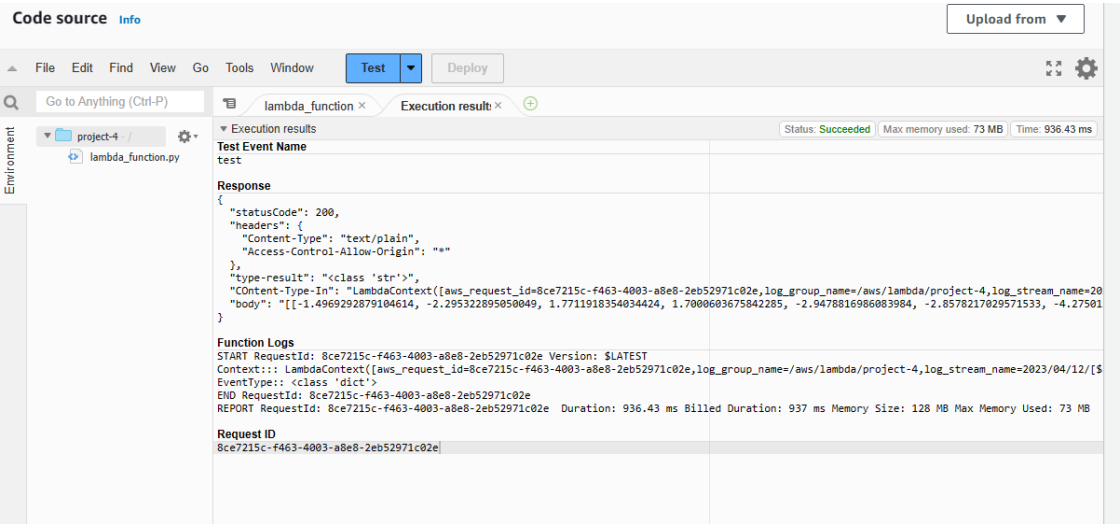
Creating policy with permission to only invoke specific endpoint.

```
{
    "Version": "2012-10-17",
    "Statement":
   [
     {
        "Sid": "VisualEditor0",
        "Effect": "Allow",
        "Action": "sagemaker:InvokeEndpoint",
        "Resource": "arn:aws:sagemaker:*:271564095025:endpoint/pytorch-inference-2023-04-12-16-
   43-37-936"
      }
   ]
}
```





- **Testing Lambda Function**



- **Response**


```
{
  "statusCode": 200,
  "headers": {
    "Content-Type": "text/plain",
    "Access-Control-Allow-Origin": "*"
  },
  "type-result": "<class 'str'>",
  "COntent-Type-In": "LambdaContext([aws_request_id=8ce7215c-f463-4003-a8e8-
2eb52971c02e,log_group_name=/aws/lambda/project-
4,log_stream_name=2023/04/12/[$LATEST]779f63f6e8f74c13a51970470403d45d,functi
on_name=project-
4,memory_limit_in_mb=128,function_version=$LATEST,invoked_function_arn=arn:aw
s:lambda:us-east-1:271564095025:function:project-
4,client_context=None,identity=CognitoIdentity([cognito_identity_id=None,cogn
ito_identity_pool_id=None])])",
  "body": "[[-1.4969292879104614, -2.295322895050049, 1.7711918354034424,
1.7000603675842285, -2.9478816986083984, -2.8578217029571533, -
4.275012493133545, -0.1858823001384735, -7.876079559326172,
3.485708713531494, 1.0156581401824951, -5.376102924346924,
0.08712732791900635, 1.043073058128357, -6.567814350128174, -
4.29003381729126, -6.980752468109131, 0.8602855205535889, -
1.9477488994598389, 3.517031192779541, 1.1386747360229492,
2.9857466220855713, -7.429625034332275, -4.872533321380615, -
6.990846633911133, -6.551341533660889, -2.713444471359253, -7.38569974899292,
-4.197381973266602, 0.5921437740325928, -0.23152270913124084, -
1.7627674341201782, -4.5388617515563965, 0.545304000377655, -
3.987135171890259, -3.03073787689209, -3.0044939517974854, -
0.24150973558425903, 1.3483713865280151, -1.685007095336914, -
1.484816312789917, 1.8437116146087646, 3.838435649871826, 0.6947075128555298,
-0.1137256920337677, -11.679482460021973, 1.3911057710647583, -
```

```
1.805575966835022, -2.34354567527771, 1.7085063457489014, -
0.9614053964614868, -8.280823707580566, -7.071294784545898,
0.16166920959949493, -0.5199874639511108, -0.843184232711792, -
6.246165752410889, -2.350785732269287, -1.3200641870498657, -
2.318167209625244, -4.5581793785095215, -8.424875259399414, -
7.389919281005859, -8.225934028625488, -5.960506916046143, -
6.004213333129883, 3.052112102508545, -0.9716691374778748,
0.1721428632736206, 1.1353175640106201, 5.675631046295166, -
4.881450176239014, -4.949894428253174, -4.2738471031188965, -
1.0440956354141235, 0.615058183670044, -7.484931468963623, -
3.1543450355529785, -4.828455448150635, -7.795574188232422,
2.6574225425720215, -8.622648239135742, 1.655287504196167,
1.2484384775161743, -8.302145957946777, -3.6932826042175293,
2.7479288578033447, -7.128697395324707, 1.1833446025848389,
2.0749804973602295, -8.263742446899414, -2.6031601428985596, -
2.9580190181732178, -6.628727436065674, -2.2038745880126953,
0.8574008941650391, 0.36140191555023193, 1.240427851676941, -
6.453752517700195, -9.306218147277832, -5.975752353668213, -
0.851379930973053, -3.8800625801086426, -4.767630577087402, -
2.8386266231536865, -5.776670932769775, -1.417716383934021,
2.4901983737945557, 1.7963000535964966, 0.9240027666091919,
1.3527525663375854, 1.043146014213562, -5.193169116973877, -
2.356482982635498, -7.384207248687744, -0.1713782399892807, -
4.5393853187561035, -2.649200916290283, -4.741747856140137,
0.618251621723175, 0.2919156551361084, -3.7748732566833496, -
3.141923666000366, -1.8757330179214478, -6.020362377166748, -
5.679347991943359, -1.4780714511871338, 1.1916701793670654, -
4.358892917633057, -7.019332408905029, -5.486293792724609, 3.572416067123413,
-3.9262099266052246]]"
}
```

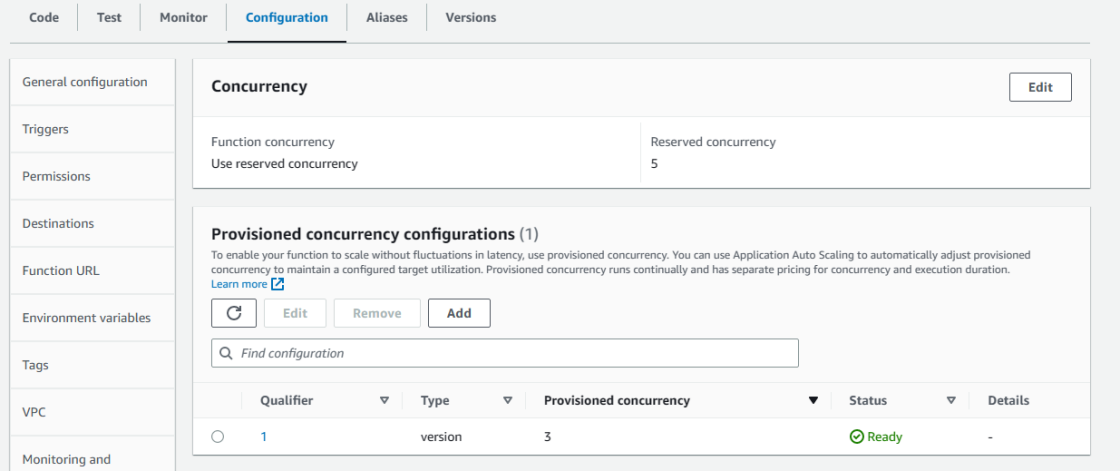

Step 5: Lambda concurrency setup and endpoint auto-scaling

- **Concurrency**

Setting up concurrency for your Lambda function. Concurrency will make your Lambda function better able to accommodate high traffic because it will enable your function to respond to multiple invocations at once. I reserved 5 instances and provisioned 3 of them.

> ***Provisioned concurrency****: computing resources that are available to be used immediately for requests to a Lambda function. Have low cost but the downside is that the maximum is a hard maximum. Thus, if your lambda function receives more request then there will be latency requests.*
>
> ***Reserved concurrency****: a set amount of computing resources that are reserved to be used for a Lambda function's concurrency. It creates instances that are always on and can reply to all traffic without requiring a wait for start-up times. Thus, have higher cost.*

```
Reserved instances: 5/1000
Provisioned instances: 3/5
```



- **Auto-scaling**

Sagemaker endpoints require automatic scaling to respond to high traffic. I enabled auto-scaling.

```
Minimum instances: 1
Maximum instances: 3
Target value: 20 //number of simultaneous requests which will trigger scaling
scale-in time: 30 s
scale-out time: 30 s
```