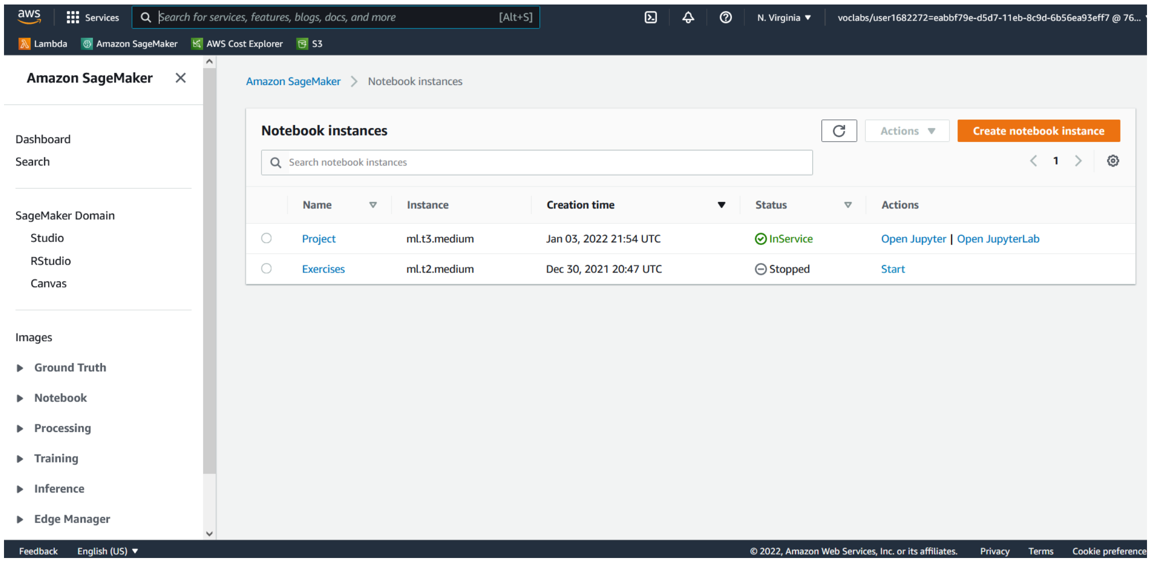
PROJECT WRITEUP

S. CHETTIH

1. Setup, training, and deployment on SageMaker

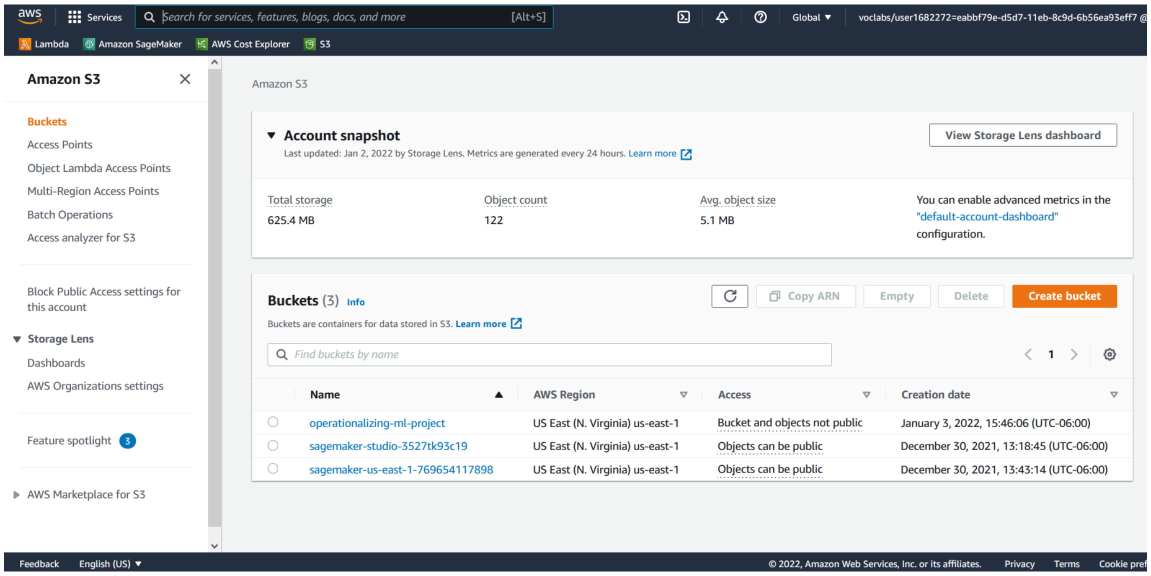
1.1. Notebook Instance. I chose a **ml.t3.medium** instance for my Jupyter notebook (Fig 1). I chose this instance type because it is eligible for the AWS free tier, and thus inexpensive, while still powerful enough for the code I planned to run on it. I knew I'd be installing packages, and downloading, unzipping, and uploading images using this instance, but model tuning, training, and deployment would happen on other instances. I also wanted the fast launch capability of ml.t3.medium, since I anticipated stopping and starting the notebook multiple times in the course of the project.



Name	Instance	Creation time	Status	Actions
Project	ml.t3.medium	Jan 03, 2022 21:54 UTC	InService	Open Jupyter | Open JupyterLab
Exercises	ml.t2.medium	Dec 30, 2021 20:47 UTC	Stopped	Start

Figure 1. Notebook instance

1.2. S3 Bucket. I created the S3 bucket `operationalizing-ml-project` to copy the data into (Fig 2). I changed the references in `train_and_deploy-solution.ipynb` to point to this bucket, and I uploaded the unzipped dogImages dataset to `s3://operationalizing-ml-project/dogImages/`.


aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Global
voclabs/user1682272=eabbf79e-d5d7-11eb-8c9d-6b56ea93eff7 @
Lambda
Amazon SageMaker
AWS Cost Explorer
S3
Amazon S3
Buckets
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
Access analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
AWS Organizations settings
Feature spotlight 3
AWS Marketplace for S3
View Storage Lens dashboard
Copy ARN
Empty
Delete
Create bucket
Find buckets by name
Feedback
English (US)
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie pref

Amazon S3

Account snapshot

Last updated: Jan 2, 2022 by Storage Lens. Metrics are generated every 24 hours. Learn more

Total storage	Object count	Avg. object size	
625.4 MB	122	5.1 MB	You can enable advanced metrics in the "default-account-dashboard" configuration.

Buckets (3) Info

Buckets are containers for data stored in S3. Learn more

Name	AWS Region	Access	Creation date
operationalizing-ml-project	US East (N. Virginia) us-east-1	Bucket and objects not public	January 3, 2022, 15:46:06 (UTC-06:00)
sagemaker-studio-3527tk93c19	US East (N. Virginia) us-east-1	Objects can be public	December 30, 2021, 13:18:45 (UTC-06:00)
sagemaker-us-east-1-769654117898	US East (N. Virginia) us-east-1	Objects can be public	December 30, 2021, 13:43:14 (UTC-06:00)

FIGURE 2. S3 Bucket

1.3. **Training and Deployment.** For hyperparameter tuning I used an **ml.m5.xlarge** instance with 2 jobs. This instance is part of the free tier for training in SageMaker, and from experience I knew this would take about 25-30 min, which was acceptable. The best hyperparameters were a batch size of 32 and a learning rate of approximately 0.0373.

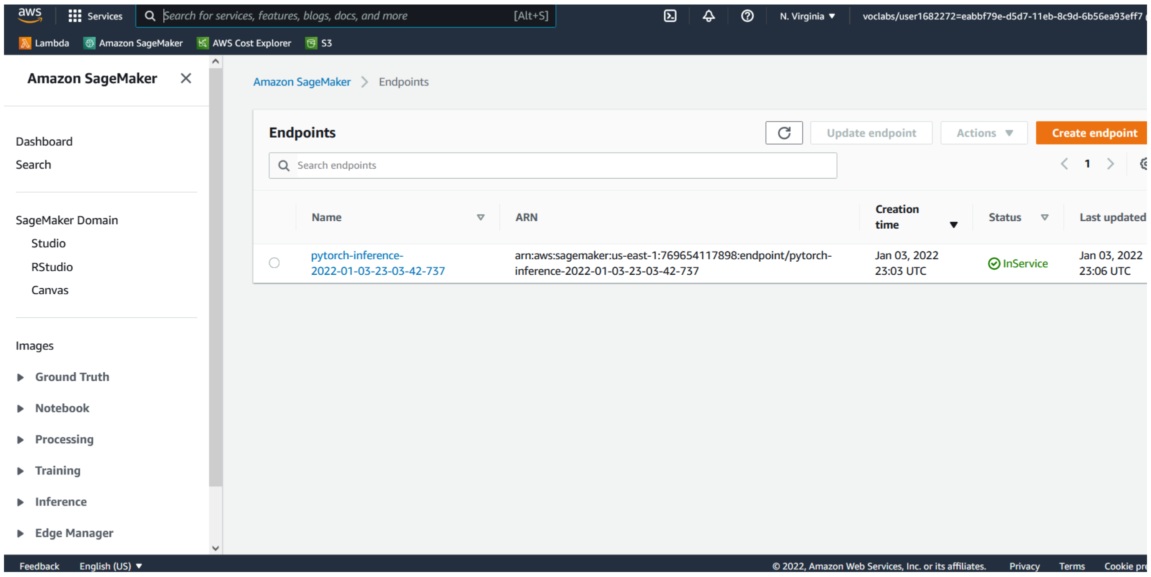
I used an **ml.m5xlarge** instance to train the estimator, using the best hyperparameters from my tuning job. Once it was trained, I deployed the model to an endpoint on a **ml.m5.large** instance. I chose this type of instance because it wouldn't need to handle very many or very large inference queries. The endpoint's name was `pytorch-inference-2022-01-03-23-03-42-737` (Fig 3).

After running the rest of the notebook, to confirm that the endpoint could perform inference, I went back and used 4 instances of type **ml.m5.xlarge** to perform multi-instance training. Once it was trained, I deployed to an endpoint on a **ml.m5.large** instance. The endpoint's name was `pytorch-inference-2022-01-04-03-32-53-734` (Fig 4).

2. EC2 TRAINING

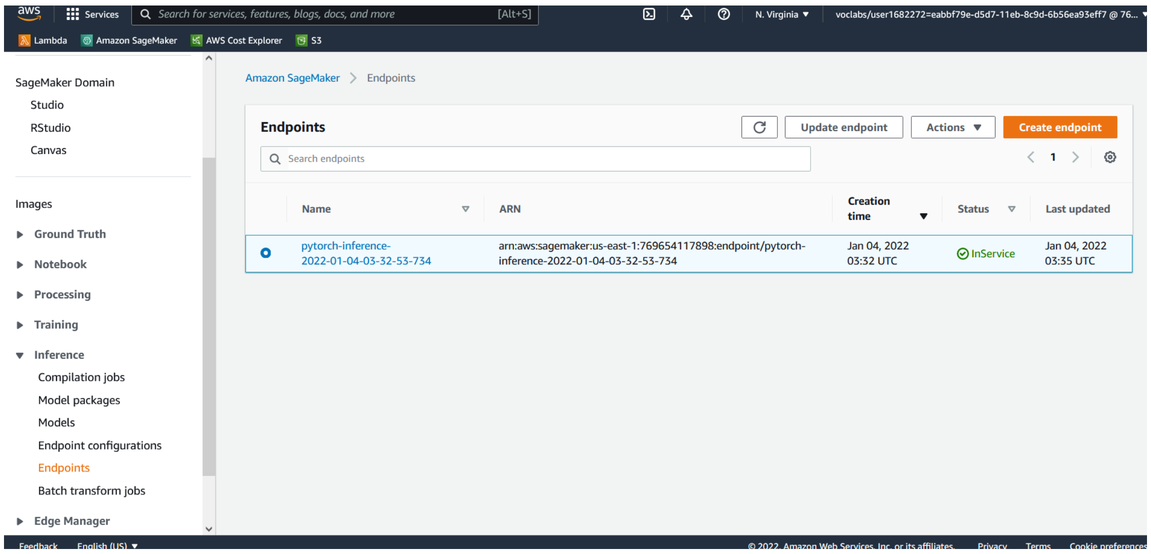
2.1. **EC2 Setup.** I chose the Deep Learning AMI (Amazon Linux 2) Version 56.0, running on an **m5.xlarge** instance (Fig 5). I knew this would be powerful enough to complete model training in a reasonable amount of time, and it wouldn't bankrupt my account if I needed to leave it on for a few hours to troubleshoot.

I launched the instance, downloaded the dataset, and trained the model, saving it in the TrainedModels directory (Fig 6).



Name	ARN	Creation time	Status	Last updated
pytorch-inference-2022-01-03-23-03-42-737	arn:aws:sagemaker:us-east-1:769654117898:endpoint/pytorch-inference-2022-01-03-23-03-42-737	Jan 03, 2022 23:03 UTC	InService	Jan 03, 2022 23:06 UTC

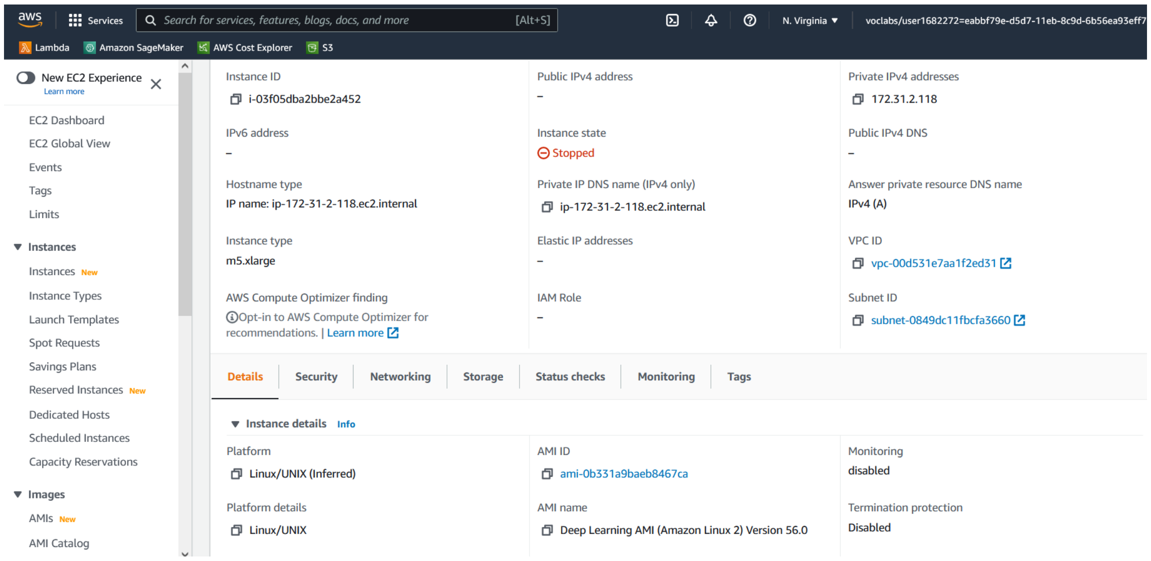
FIGURE 3. Endpoint from Single-Instance Training



Name	ARN	Creation time	Status	Last updated
pytorch-inference-2022-01-04-03-32-53-734	arn:aws:sagemaker:us-east-1:769654117898:endpoint/pytorch-inference-2022-01-04-03-32-53-734	Jan 04, 2022 03:32 UTC	InService	Jan 04, 2022 03:35 UTC

FIGURE 4. Endpoint from Multi-Instance Training

2.2. **EC2 vs SageMaker.** The code in `ec2train1.py` is similar to `train_and_deploy-solution.ipynb` plus `hpo.py`, with a few key differences. Firstly, the EC2 script performs all training locally. Contrast this to SageMaker, where the training data, model, and output are stored on S3, and estimators are fit on separate instances from the notebook where the code is run. This



Instance ID
i-03f05dba2bbe2a452

Public IPv4 address
–

Private IPv4 addresses
172.31.2.118

IPv6 address
–

Instance state
Stopped

Public IPv4 DNS
–

Hostname type
IP name: ip-172-31-2-118.ec2.internal

Private IP DNS name (IPv4 only)
ip-172-31-2-118.ec2.internal

Answer private resource DNS name
IPv4 (A)

Instance type
m5.xlarge

Elastic IP addresses
–

VPC ID
vpc-00d531e7aa1f2ed31

AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations. | Learn more

IAM Role
–

Subnet ID
subnet-0849dc11fbcfa3660

Instance details Info

Platform
Linux/UNIX (Inferred)

AMI ID
ami-0b331a9baeb8467ca

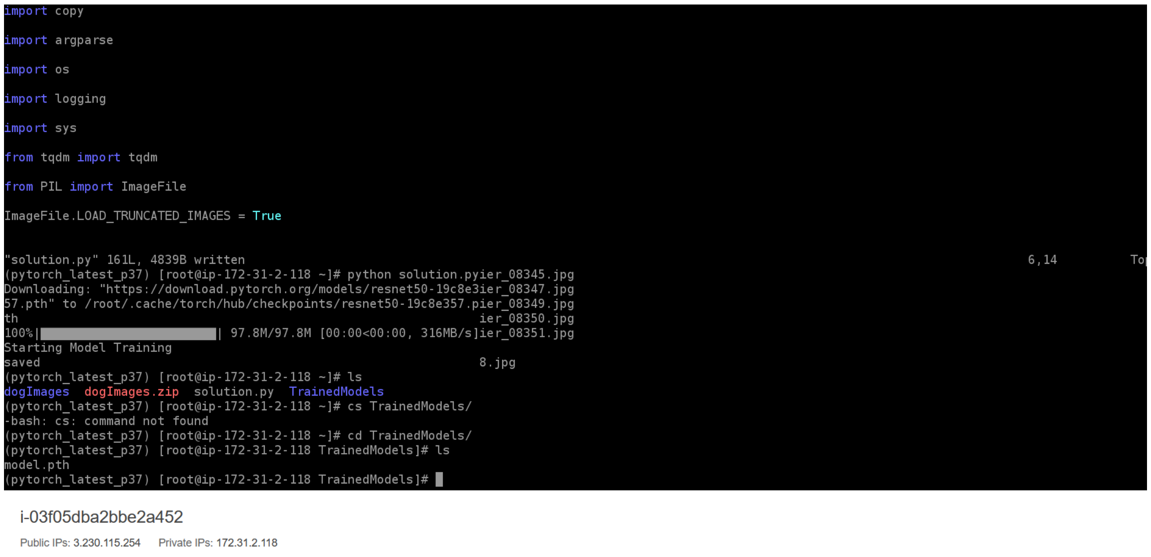
Monitoring
disabled

Platform details
Linux/UNIX

AMI name
Deep Learning AMI (Amazon Linux 2) Version 56.0

Termination protection
Disabled

FIGURE 5. EC2 Instance



```
import copy

import argparse

import os

import logging

import sys

from tqdm import tqdm

from PIL import ImageFile

ImageFile.LOAD_TRUNCATED_IMAGES = True


"solution.py" 161L, 4839B written                                                                              6,14          To
(pytorch_latest_p37) [root@ip-172-31-2-118 ~]# python solution.pyier_08345.jpg
Downloading: "https://download.pytorch.org/models/resnet50-19c8e3ier_08347.jpg
57.pth" to /root/.cache/torch/hub/checkpoints/resnet50-19c8e357.pier_08349.jpg
th                                                                  ier_08350.jpg
100%|████████████████████████| 97.8M/97.8M [00:00<00:00, 316MB/s]ier_08351.jpg
Starting Model Training
saved                                                               8.jpg
(pytorch_latest_p37) [root@ip-172-31-2-118 ~]# ls
dogImages  dogImages.zip  solution.py  TrainedModels
(pytorch_latest_p37) [root@ip-172-31-2-118 ~]# cs TrainedModels/
-bash: cs: command not found
(pytorch_latest_p37) [root@ip-172-31-2-118 ~]# cd TrainedModels/
(pytorch_latest_p37) [root@ip-172-31-2-118 TrainedModels]# ls
model.pth
(pytorch_latest_p37) [root@ip-172-31-2-118 TrainedModels]# █
```

i-03f05dba2bbe2a452

Public IPs: 3.230.115.254 Private IPs: 172.31.2.118

FIGURE 6. TrainedModels Directory

is part of why the EC2 script doesn't need a parser or a main function. In SageMaker, the parser passes on the learning rate and batch size, as well as the training, model, and output directories to the main function. In EC2, all of these are simply specified in the script. This is only possible because the dataset is available locally, the training is done locally, and the model can be saved locally. Finally, it's not possible to directly deploy the trained model to an endpoint from EC2. We only know how to deploy endpoints from models within SageMaker, so we'd need to get the model from the EC2 instance into SageMaker first before we could deploy an endpoint to perform real-time inference.

3. Lambda Function Setup

I used the `lambdafunction.py` starter file to set up a Lambda function called `dog-breed-prediction`. The function first creates a Boto3 client. This allows the function to interact with other AWS applications, like SageMaker. The function uses the client to invoke the endpoint I deployed from the multi-instance trained model, called `pytorch-inference-2022-01-04-03-32-53-734`. It expects the input/image in JSON format, and it returns a JSON file. This JSON file has a status code of 200 and the prediction from the endpoint in the key 'body'.

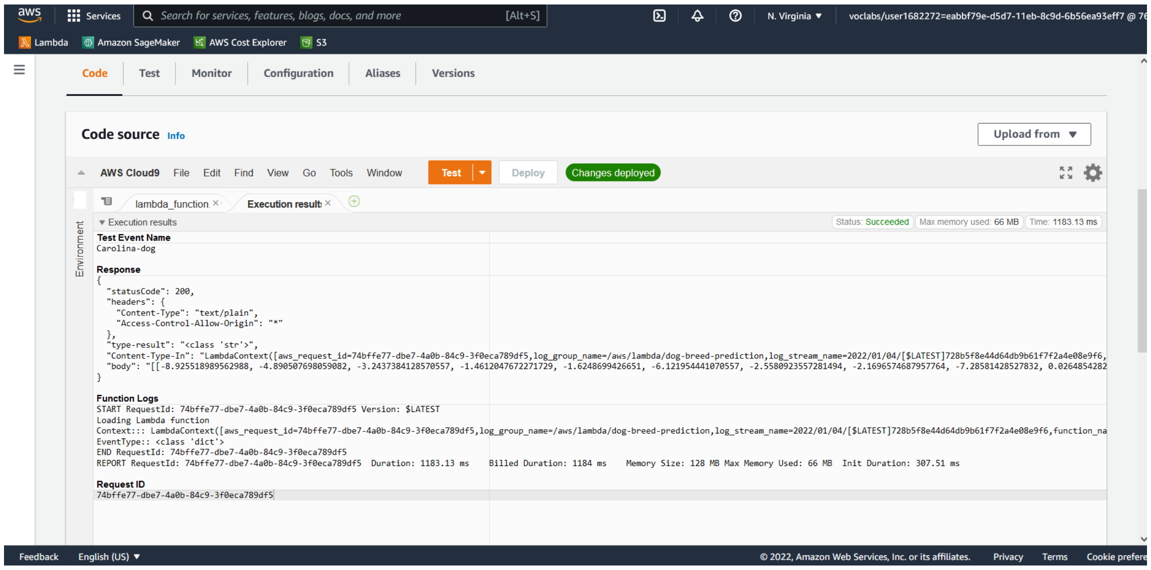
4. Lambda Function Security and Testing

4.1. **Testing.** After attaching the appropriate permissions to the Lambda function role, I ran a successful test of my lambda function (Fig 7). The returned list is

$$
\begin{aligned}
&[-8.925518989562988, -4.890507698059082, -3.2437384128570557, -1.4612047672271729,\\
&-1.6248699426651, -6.121954441070557, -2.5580923557281494, -2.1696574687957764,\\
&-7.28581428527832, 0.02648542821407318, 0.19917014241218567, -5.017480373382568,\\
&-3.6260993480682373, 1.1632394790649414, -5.876077175140381, -2.9551069736480713,\\
&-5.999130725860596, -2.639396905899048, -4.315248966217041, -0.2771189212799072,\\
&-4.760954856872559, -1.261879563331604, -8.60474681854248, -6.055394649505615,\\
&-5.513672828674316, -8.68757152557373, -4.001145839691162, -4.517764568328857,\\
&-5.7072601318359375, -2.0146870613098145, -5.00324010848999, -4.045901298522949,\\
&-7.567681789398193, -1.9930708408355713, -8.426380157470703, -4.777566909790039,\\
&-4.981318950653076, -3.4745962619781494, -0.4175407290458679, -5.553298473358154,\\
&-3.684882879257202, -2.5894155502319336, -0.07702390849590302, -3.7064366340637207,\\
&-1.2497320175170898, -9.527958869934082, -1.3992691040039062, -0.8133065104484558,\\
&-5.449779987335205, -1.8658945560455322, -3.7588181495666504, -6.388206958770752,\\
&-8.07963752746582, -4.794607162475586, -6.696590423583984, -2.123265504837036,\\
&-4.90000057220459, -4.982354640960693, -2.1294171810150146, -4.1486663818359375,\\
&-5.791005611419678, -7.688570976257324, -8.112510681152344, -8.65478229522705,\\
&-4.661163330078125, -7.993859767913818, 0.3516315519809723, -6.90894889831543,\\
&-3.3528940677642822, -2.145860433578491, -0.09199655801057816, -4.984414577484131,\\
&-5.115091323852539, -7.574524402618408, -5.158764362335205, -2.175959825515747,\\
&-9.419661521911621, -2.6756057739257812, -6.178658962249756, -5.340219974517822,\\
&-1.113185167312622, -7.447085380554199, -0.99626624584198, -1.6738048791885376,\\
&-6.99773645401001, -7.048044204711914, -1.6650669574737549, -8.459725379943848,
\end{aligned}
$$

$-3.6921746730804443, -0.7576943635940552, -7.3646063804626465, -5.9802021980285645,$
$-4.481424808502197, -5.951948642730713, -4.740312576293945, -2.958517551422119,$
$-4.381550312042236, -4.749039649963379, -7.411319732666016, -6.350480556488037,$
$-7.72014856338501, -3.0790865421295166, -4.492247104644775, -3.869389295578003,$
$-5.422338485717773, -8.145662307739258, -2.894341230392456, -1.49086594581604,$
$-1.7460885047912598, -1.9906584024429321, -2.62567138671875, -2.025052785873413,$
$-4.880069732666016, -4.984194755554199, -6.928412437438965, -1.6735512018203735,$
$-8.021923065185547, -0.8875415325164795, -6.566311359405518, -1.2155416011810303,$
$-4.719881534576416, -2.7002644538879395, -4.106622695922852, -5.600302696228027,$
$-6.946561336517334, -3.35860538482666, -3.570112466812134, -1.1193897724151611,$
$-6.629595756530762, -7.377218246459961, -4.372668266296387, -1.3083720207214355,$
$-2.9448294639587402]$

The highest number is in index 13, which means the model thinks this is most likely a



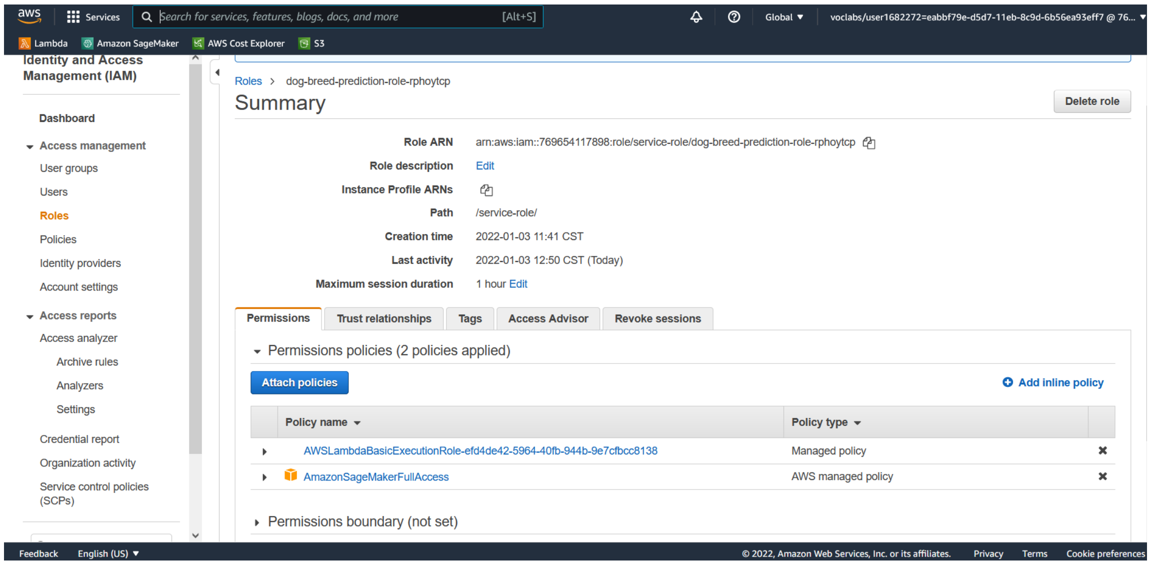
```
Test Event Name
Carolina-dog

Response
{
  "statusCode": 200,
  "headers": {
    "Content-Type": "text/plain",
    "Access-Control-Allow-Origin": "*"
  },
  "type-result": "<class 'str'>",
  "Content-Type-In": "LambdaContext([aws_request_id=74bffe77-dbe7-4a0b-84c9-3f0eca789df5,log_group_name=/aws/lambda/dog-breed-prediction,log_stream_name=2022/01/04/[$LATEST]728b5f8e44d64db9b61f7f2a4e08e9f6,
  "body": "[[-8.925518989562988, -4.890507698059082, -3.2437384128570557, -1.4612047672271729, -1.6248699426651, -6.121954441070557, -2.5580923557281494, -2.1696574687957764, -7.28581428527832, 0.0264854282
}

Function Logs
START RequestId: 74bffe77-dbe7-4a0b-84c9-3f0eca789df5 Version: $LATEST
Loading Lambda function
Context::: LambdaContext([aws_request_id=74bffe77-dbe7-4a0b-84c9-3f0eca789df5,log_group_name=/aws/lambda/dog-breed-prediction,log_stream_name=2022/01/04/[$LATEST]728b5f8e44d64db9b61f7f2a4e08e9f6,function_na
EventType:: <class 'dict'>
END RequestId: 74bffe77-dbe7-4a0b-84c9-3f0eca789df5
REPORT RequestId: 74bffe77-dbe7-4a0b-84c9-3f0eca789df5  Duration: 1183.13 ms    Billed Duration: 1184 ms    Memory Size: 128 MB Max Memory Used: 66 MB  Init Duration: 307.51 ms

Request ID
74bffe77-dbe7-4a0b-84c9-3f0eca789df5
```

FIGURE 7. Successful Lambda Test

Basenji. That's a decent guess, in my opinion, based on the body shape and coat type of the dog. The full test response can be found in the file `lambda_test_results.txt`.

4.2. **Security.** I took a screenshot of the policies that are attached to my Lambda function's role (Fig 8). In considering which policy to attach, I looked for one that would only allow the Lambda function to invoke endpoints, but I couldn't find one defined that narrowly. This is a vulnerability, since I've granted the Lambda function full access to

SageMaker. This could be used to get or delete objects from S3, create or delete VPC endpoints in EC2, get metric data from CloudWatch, and list the secrets in Secrets Manager, among others. This suggests that we should be careful who is authorized to create Lambda functions in the first place, and we should regularly check the Access Advisor for roles to see which services are being accessed and when, to revoke the permissions if we see anything untoward. We should practice basic operational security around AWS accounts, such as multi-factor authentication, privileged account workstations, and regular active-user/role audits. I recognize all of the IAM roles and services accessed in my account, since it is nearly brand-new, but it's worth keeping an eye on.



Summary

Role ARN	arn:aws:iam::769654117898:role/service-role/dog-breed-prediction-role-rphoytcp
Role description	Edit
Instance Profile ARNs	
Path	/service-role/
Creation time	2022-01-03 11:41 CST
Last activity	2022-01-03 12:50 CST (Today)
Maximum session duration	1 hour Edit

Permissions policies (2 policies applied)

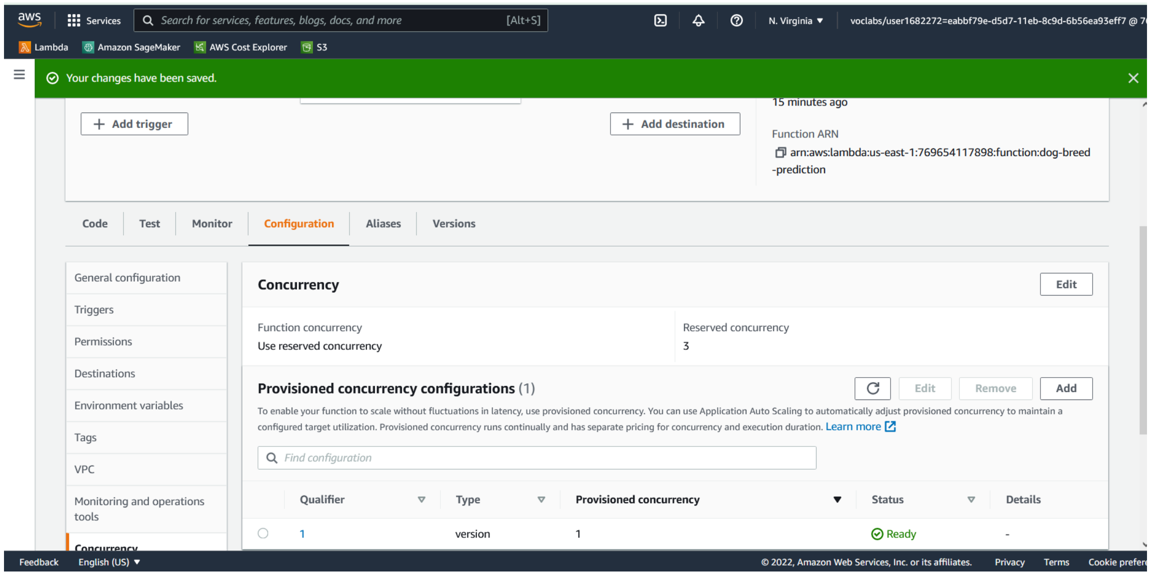
Policy name	Policy type
AWSLambdaBasicExecutionRole-efd4de42-5964-40fb-944b-9e7cfbcc8138	Managed policy
AmazonSageMakerFullAccess	AWS managed policy

Permissions boundary (not set)

FIGURE 8. Lambda Function IAM Role Policies

5. CONCURRENCY AND AUTO-SCALING

5.1. **Concurrency.** I configured a version of my Lambda function, and then I chose 3 reserved concurrency and 1 provisioned concurrency (Fig 9). I chose to add some of each kind of concurrency to demonstrate my knowledge, and these would be more than sufficient for my use of the Lambda function. Similarly, I configured auto-scaling for my deployed endpoint, with a min instance count of 1, a max of 3, and scale-in and scale-out cool down times of 30 seconds. Those settings were sufficient for the exercise we did earlier in the course, so I feel they're fine here as well.



voclabs/user1682272=eabbf79e-d5d7-11eb-8c9d-6b56ea93eff7 @ 7

Your changes have been saved.

15 minutes ago

Function ARN
arn:aws:lambda:us-east-1:769654117898:function:dog-breed-prediction

Concurrency

Function concurrency	Reserved concurrency
Use reserved concurrency	3

Provisioned concurrency configurations (1)

To enable your function to scale without fluctuations in latency, use provisioned concurrency. You can use Application Auto Scaling to automatically adjust provisioned concurrency to maintain a configured target utilization. Provisioned concurrency runs continually and has separate pricing for concurrency and execution duration. Learn more

Qualifier	Type	Provisioned concurrency	Status	Details
1	version	1	Ready	-

FIGURE 9. Lambda Concurrency Configuration

Thank you for your time!