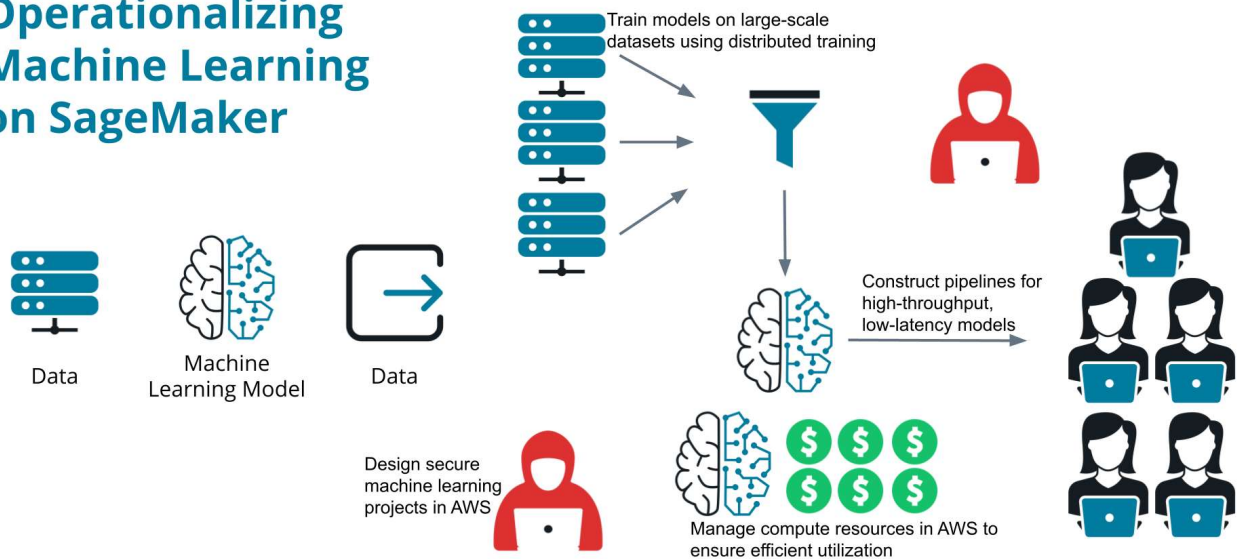


README.md

# Operationalizing an AWS Machine Learning.

## Operationalizing Machine Learning on SageMaker



[source Udacity](#)

In this project, I completed the following steps:

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

## Training and deployment on Sagemaker

Reason for using ml.t3.medium notebook instance:

Amazon SageMaker > Notebook instances

**Notebook instances** Actions Create notebook instance

Search notebook instances

Name	Instance	Creation time	Status	Actions
sagemaker-notebook-instance	ml.t3.medium	Nov 05, 2022 16:09 UTC	InService	Open Jupyter   Open JupyterLab

- Cost saving using this instance:
  - ml.t3.medium costs \$0.05 an hour this can be seen on the sagemaker [pricing page](#)
- This instance will be used for workloads like download of images and upload to S3 and will be used to create training jobs and deployment.
  - Training and Model deployments will depend on the compute power of instances ml.m5.xlarge used for both hyperparameter tuning and Estimator training. and ml.m5.large instance used for model deployment.

## Download data to an S3 bucket

Amazon S3 > Buckets

**Account snapshot** View Storage Lens dashboard

Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

**Buckets (1)** Info Copy ARN Empty Delete Create bucket

Buckets are containers for data stored in S3. [Learn more](#)

Find buckets by name

Name	AWS Region	Access	Creation date
sagemaker-bucket-05-11-2022	US East (N. Virginia) us-east-1	Bucket and objects not public	November 5, 2022, 17:30:04 (UTC+01:00)

After creating bucket, reference to the data path is as follows:

```
%%capture
!wget https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip
!unzip dogImages.zip
!aws s3 cp dogImages s3://sagemaker-bucket-05-11-2022/ --recursive

os.environ['SM_CHANNEL_TRAINING'] = 's3://sagemaker-bucket-05-11-2022/'
```

After downloading data from url, then unzip the dogImages.zip next is to download the data to S3 using the part s3://sagemaker-bucket-05-11-2022/

os.environ['SM\_CHANNEL\_TRAINING'] represents the Training path to the directory that contains the input data for this channel.

## Training and Deployment

```

estimator = PyTorch(
    entry_point='hpo.py',
    base_job_name='dog-pytorch',
    role=role,
    instance_count=1,
    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,
)

```

Training jobs <span>Info</span>						<div> <div>↺</div> <div>Actions</div> <div>↻</div> </div> <div>Create training job</div>	
<div> <div>🔍</div> <div>Search training jobs</div> </div>						<div> <div>&lt;</div> <div>1</div> <div>&gt;</div> <div>⚙️</div> </div>	
	Name	Creation time	Duration	Job status	Warm pool status		
<input type="radio"/>	<a href="#">dog-pytorch-2022-11-05-19-00-13-634</a>	Nov 05, 2022 19:00 UTC	23 minutes	✅ Completed	-		
<input type="radio"/>	<a href="#">dog-pytorch-2022-11-05-18-37-48-154</a>	Nov 05, 2022 18:37 UTC	3 minutes	❌ Failed	-		
<input type="radio"/>	<a href="#">dog-pytorch-2022-11-05-18-32-13-003</a>	Nov 05, 2022 18:32 UTC	3 minutes	❌ Failed	-		
<input type="radio"/>	<a href="#">pytorch-training-221105-1738-002-b4761cf3</a>	Nov 05, 2022 17:58 UTC	18 minutes	✅ Completed	⊖ Terminated		
<input type="radio"/>	<a href="#">pytorch-training-221105-1738-001-0a4bb29e</a>	Nov 05, 2022 17:38 UTC	20 minutes	✅ Completed	⊖ Reused		
<input type="radio"/>	<a href="#">pytorch-training-221105-1655-003-1404d1fa</a>	Nov 05, 2022 17:20 UTC	3 minutes	❌ Failed	⊖ Terminated		
<input type="radio"/>	<a href="#">pytorch-training-221105-1655-002-a33f7894</a>	Nov 05, 2022 17:00 UTC	18 minutes	✅ Completed	⊖ Reused		

This job `dog-pytorch-2022-11-05-19-00-13-634` is a training job trained. Using pytorch estimator, I trained my model using `instance_count=1` thus training the model in `hpo.py` using 1 instances of `ml.m5.xlarge`.

**SageMaker** training of your script `is` invoked when you call `fit` on the **PyTorch Estimat**

```
estimator.fit({"training": "s3://sagemaker-bucket-05-11-2022/"}, wait=False)
```



Amazon SageMaker > Endpoints

Endpoints ↻ Update endpoint Actions ▼ Create endpoint

Search endpoints

	Name ▼	ARN	Creation time ▼	Status ▼	Last updated
<input type="radio"/>	pytorch-inference-2022-11-05-19-35-47-532	arn:aws:sagemaker:us-east-1:690349917069:endpoint/pytorch-inference-2022-11-05-19-35-47-532	Nov 05, 2022 19:35 UTC	<span>✓</span> InService	Nov 05, 2022 19:38 UTC

After training, I deploy the endpoint. As seen above.

## Multi-instance Training

```
estimator = PyTorch(
    ...
    instance_count=3,
    instance_type='ml.m5.xlarge',
    ...
)
```

```
estimator.fit({"training": "s3://sagemaker-bucket-05-11-2022/"}, wait=False)
```

Log streams (11) ↻ Delete Create log stream Search all log stream

Search dog-pytorch-2022-11-05-20-02-56-524 × 3 matches ☐ Exact match

<input type="checkbox"/>	Log stream ▼	Last event time
<input type="checkbox"/>	dog-pytorch-2022-11-05-20-02-56-524/algo-3-1667678658	2022-11-05 21:06:00 (UTC+01:00)
<input type="checkbox"/>	dog-pytorch-2022-11-05-20-02-56-524/algo-1-1667678656	2022-11-05 21:05:58 (UTC+01:00)
<input type="checkbox"/>	dog-pytorch-2022-11-05-20-02-56-524/algo-2-1667678658	2022-11-05 21:05:55 (UTC+01:00)

Amazon SageMaker > Training jobs

Training jobs [Info](#) ↻ Actions ▼ Create training job

Search training jobs

	Name ▼	Creation time ▼	Duration	Job status ▼	Warm pool status
<input type="radio"/>	dog-pytorch-2022-11-05-20-02-56-524	Nov 05, 2022 20:02 UTC	20 minutes	<span>✓</span> Completed	-

Using pytorch estimator, I created multi-instance training using `instance_count=3` thus training the model in `hpo.py` using 3 instances of `ml.m5.xlarge`.

Deployed new endpoint that was trained on multiple instances

Amazon SageMaker > Endpoints

Endpoints					
<input type="text" value="Search endpoints"/>		<input type="button" value="Update endpoint"/>		<input type="button" value="Actions"/>	<input type="button" value="Create endpoint"/>
<div>&lt; 1 &gt; ⚙</div>					
	Name	ARN	Creation time	Status	Last updated
<input type="radio"/>	<a href="#">pytorch-inference-2022-11-05-20-43-15-461</a>	arn:aws:sagemaker:us-east-1:690349917069:endpoint/pytorch-inference-2022-11-05-20-43-15-461	Nov 05, 2022 20:43 UTC	<span>✔ InService</span>	Nov 05, 2022 20:45 UTC
<input type="radio"/>	<a href="#">pytorch-inference-2022-11-05-19-35-47-532</a>	arn:aws:sagemaker:us-east-1:690349917069:endpoint/pytorch-inference-2022-11-05-19-35-47-532	Nov 05, 2022 19:35 UTC	<span>✔ InService</span>	Nov 05, 2022 19:38 UTC

After which this training job was deployed. As seen above 2 endpoints are active, for the 2 trained jobs created with `starter/train_and_deploy-solution.ipynb` notebook.

## EC2 Training


**Quickstart AMIs (8)**  
Commonly used AMIs

**My AMIs (0)**  
Created by me

**AWS Marketplace AMIs (151)**  
AWS & trusted third-party AMIs


**Community AMIs (500)**  
Published by anyone

---

  
Amazon Linux  
**Verified provider**

**Deep Learning AMI GPU PyTorch 1.12.1 (Amazon Linux 2) 20221102**  
ami-0769cca279cc4c998 (64-bit (x86))  
Built with PyTorch conda environment, NVIDIA CUDA, cuDNN, NCCL, GPU Driver, Docker, NVIDIA-Docker and EFA support. For a fully managed experience, check: <https://aws.amazon.com/sagemaker>  
Platform: amazon    Root device type: ebs    Virtualization: hvm    ENA enabled: Yes

☒ 64-bit (x86)

  
Amazon Linux  
**Verified provider**

**Deep Learning AMI GPU TensorFlow 2.10.0 (Amazon Linux 2) 20221024**  
ami-005d340ef0bae27d5 (64-bit (x86))  
Built with AWS optimized TensorFlow, NVIDIA CUDA, cuDNN, NCCL, GPU Driver, Docker, NVIDIA-Docker and EFA support. For a fully managed experience, check: <https://aws.amazon.com/sagemaker>  
Platform: amazon    Root device type: ebs    Virtualization: hvm    ENA enabled: Yes

☒ 64-bit (x86)

An Amazon Deep Learning AMI Linux 2 was selected for the EC2 instance, this is to use deep learning libraries. Here I selected an Amazon Linux OS with PyTorch conda environment.



Amazon Machine Image (AMI)

Deep Learning AMI GPU PyTorch 1.12.1  
(Amazon Linux 2) 20221102  
ami-0769cca279cc4c998

Verified provider

  
Browse more AMIs  
Including AMIs from  
AWS, Marketplace and  
the Community

Catalog	Published	Architecture	Virtualization	Root device type	ENA Enabled
Quickstart AMIs	2022-11-02T09:17:47.000Z	x86_64	hvm	ebs	Yes

▼ Instance type [Info](#)

Instance type

t2.xlarge  
Family: t2 4 vCPU 16 GiB Memory  
On-Demand Linux pricing: 0.1856 USD per Hour  
On-Demand Windows pricing: 0.2266 USD per Hour

[Compare instance types](#)

<input checked="" type="checkbox"/>	Name	Instance ID	Instance state	Instance type
<input checked="" type="checkbox"/>	DL_Instance	i-0739af04b41db4a42	<span>Running</span>	t2.xlarge

This instance type selected is comparable has some cost savings when compared with m1.m5.xlarge used for sagemaker training.

Attributes	m1.m5.xlarge	t2.xlarge
CPU	4	4
Memory	16	16
Price per hour	\$0.23	\$0.1856
t2.xlarge instance is approximately 20% cheaper than m1.m5.xlarge even though there CPU and Memory attributes are same.		

Instance: i-0739af04b41db4a42 (DL\_Instance)

DetailsSecurityNetworkingStorageStatus checksMonitoringTags

▼ Root device details

Root device name  /dev/xvda	Root device type EBS	EBS optimization disabled
---	-------------------------	------------------------------

The training was done using EBS as storage.

Characteristics	S3	EBS	EFS	Glacier
Speed	Moderate	Fastest	Moderate	Slowest

Characteristics	S3	EBS	EFS	Glacier
Use case	Unstructured data	EC2-system storage	EC2-scalable storage	Long-term storage of huge datasets
Interface	Web interface	File system interface	Web & file system interface	File system interface

In sagemaker we used s3 for our training, but training on EC2 instance the training files is done on EBS which the fastest data store.

```

root@ip-172-31-84-10 ~]# source activate pytorch
(pytorch) [root@ip-172-31-84-10 ~]# wget https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip
--2022-11-06 09:37:36-- https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip
Resolving s3-us-west-1.amazonaws.com (s3-us-west-1.amazonaws.com)... 52.219.120.40
Connecting to s3-us-west-1.amazonaws.com (s3-us-west-1.amazonaws.com)|52.219.120.40|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1132023110 (1.1G) [application/zip]
Saving to: 'dogImages.zip'

 99% [=====>] 445,790,533 8.53MB/s et
inflating: dogImages/valid/133.Yorkshire terrier/Yorkshire terrier_08336.jpg
inflating: dogImages/valid/133.Yorkshire terrier/Yorkshire terrier_08348.jpg
(pytorch) [root@ip-172-31-84-10 ~]# mkdir TrainedModels
(pytorch) [root@ip-172-31-84-10 ~]# vim solution.py
criterion = nn.CrossEntropyLoss()

optimizer = optim.Adam(model.fc.parameters(), lr=learning_rate)
logger.info("Starting Model Training")
model=train(model, train_loader, validation_loader, criterion, optimizer)
torch.save(model.state_dict(), 'TrainedModels/model.pth')
print('saved')

(pytorch) [root@ip-172-31-84-10 ~]# python solution.py
opt/conda/envs/pytorch/lib/python3.9/site-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated and will be removed in 0.15, please use 'weights' instead.
  warnings.warn(
opt/conda/envs/pytorch/lib/python3.9/site-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or 'Weights' are deprecated since 0.13 and will be removed in 0.15. The current behavior is equivalent to passing 'weights=ResNet50_Weights.IMAGE'. You can also use 'weights=ResNet50_Weights.DEFAULT' to get the most 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
00%[ ] 97.8M/97.8M [00:01<00:00,
Starting Model Training

```

```
wget https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip
unzip dogImages.zip
```

```
mkdir TrainedModels
```

```
vim solution.py
```

```
:set paste
```

```
python solution.py
```

```
cd TrainedModels
```

Seen Above both screenshot and commands used for training in the EC2 instance. vim solution.py creates and open the vim text editor file solution.py in the EC2 terminal. :set paste enables pasting the code from starter/ec2train1.py into the solution.py .

python solution.py is a command used to run the code.

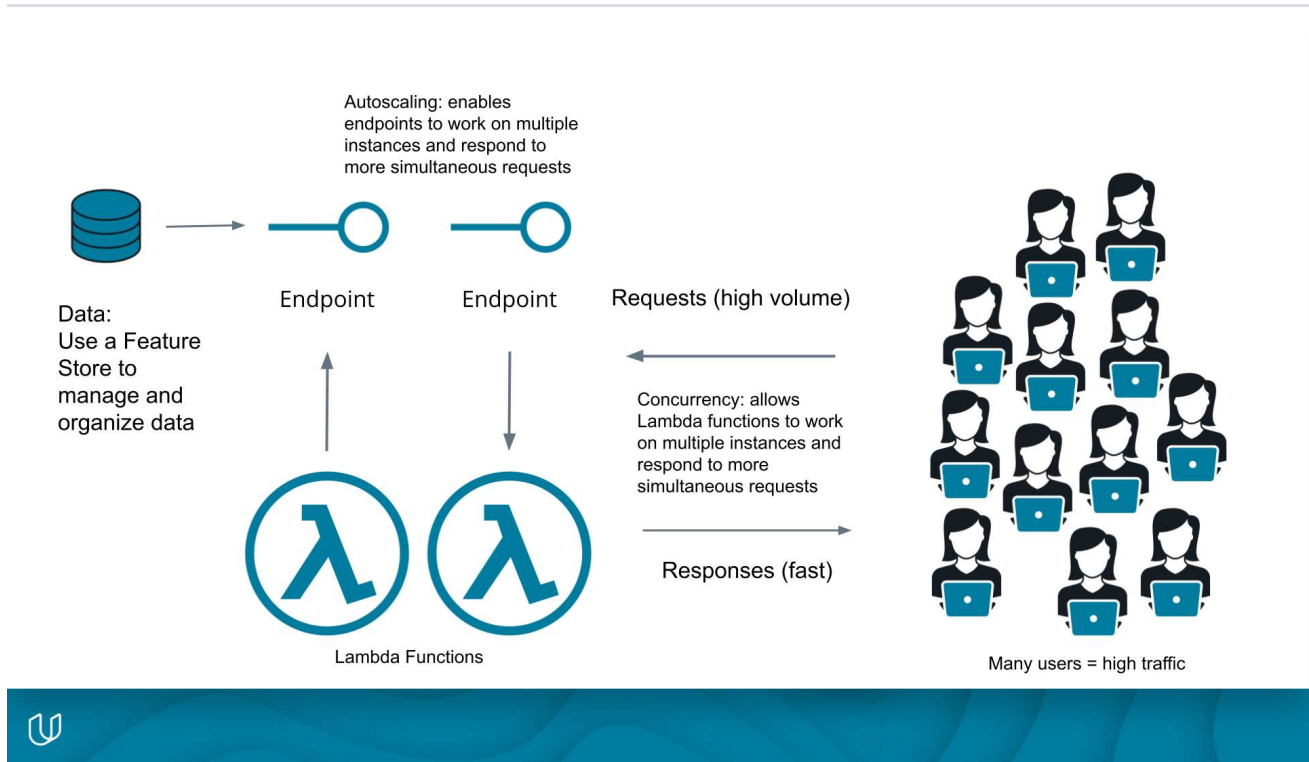
```
aws Services Search [Alt+S] N. Virginia voclabs/user1934616~9205698537 @ 6903-4991-7069
(pytorch) [root@ip-172-31-84-10 ~]# mkdir TrainedModels
(pytorch) [root@ip-172-31-84-10 ~]# vim solution.py
(pytorch) [root@ip-172-31-84-10 ~]# python solution.py
/opt/conda/envs/pytorch/lib/python3.9/site-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13.0 and will be removed in 0.15, please use 'weights' instead.
  warnings.warn(
/opt/conda/envs/pytorch/lib/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 will be removed in 0.15. The current behavior is equivalent to passing 'weights=ResNet50_Weights.IMAGENET1K_V1'. You can also use 'weights=ResNet50_Weights.DEFAULT' to get the most 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 [00:01<00:00, 93.5MB/s]
Starting Model Training
saved
(pytorch) [root@ip-172-31-84-10 ~]# dir
dogImages dogImages.zip solution.py TrainedModels
(pytorch) [root@ip-172-31-84-10 ~]# cd TrainedModels
(pytorch) [root@ip-172-31-84-10 TrainedModels]# dir
model.pth
(pytorch) [root@ip-172-31-84-10 TrainedModels]# exit
logout

i-0739af04b41db4a42 (DL_Instance)
PublicIPs: 54.88.204.103 PrivateIPs: 172.31.84.10
```

This shows the trained model as seen above `model.pth` saved in `TrainedModel` directory.

Writing on EC2 instance is cheaper and also faster, but not userfriendly when compared to Sagemaker Notebook or Studio.

## Construct pipeline for high throughput low latency models



[source Udacity](<https://www.udacity.com/course/>

## Lambda function setup

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.

I created a lambda function called `lambdafunction`. The code below invokes deployed endpoint which is used to make predictions of dogbreed images.



```

import base64
import logging
import json
import boto3
#import numpy
logger = logging.getLogger(__name__)
logger.setLevel(logging.DEBUG)

print('Loading Lambda function')

runtime=boto3.Session().client('sagemaker-runtime')
endpoint_Name='pytorch-inference-2022-11-05-20-43-15-461'##BradTestEndpoint'

def lambda_handler(event, context):

    #x=event['content']
    #aa=x.encode('ascii')
    #bs=base64.b64decode(aa)
    print('Context::',context)
    print('EventType::',type(event))
    bs=event
    runtime=boto3.Session().client('sagemaker-runtime')

    response=runtime.invoke_endpoint(EndpointName=endpoint_Name,
                                    ContentType="application/json",
                                    Accept='application/json',
                                    #Body=bytearray(x)
                                    Body=json.dumps(bs))

    result=response['Body'].read().decode('utf-8')
    sss=json.loads(result)

    return {
        'statusCode': 200,
        'headers' : { 'Content-Type' : 'text/plain', 'Access-Control-Allow-Origin' :
        'type-result':str(type(result)),
        'Content-Type-In':str(context),
        'body' : json.dumps(sss)
        #'updated_result':str(updated_result)

    }

```

runtime=boto3.Session().client('sagemaker-runtime') invokes the sagemaker runtime environment.

In the above code, runtime.invoke\_endpoint(...) invokes endpoint using name of the endpoint endpoint\_name , body provides input data, in the format specified in the ContentType which is application/json here is json format. This gets inference from the model hosted at this endpoint pytorch-inference-2022-11-05-20-43-15-461 .

this can be seen in the test event. as this is the event .

```
{  
    "url": "https://s3.amazonaws.com/cdn-origin-etr.akc.org/wp-content/uploads/2017/1  
}
```

Accept the desired MIME type of the inference in the response which is application/json here.

```
result=response['Body'].read().decode('utf-8')  
sss=json.loads(result)
```

read will return bytes. At least for Python 3 then you have to decode using the right encoding in this case utf-8 . The

loads() method can be used to parse a valid JSON string and convert it into a Python Dictionary. It is mainly used for deserializing native string, byte, or byte array which consists of JSON data into Python Dictionary. [geeksforgeeks](#)

Then returns a dictionary of 133 classes with there likelihood values.

## Security and testing

The Lambda function can only invoke my endpoint if there is proper security policies attached to it.

Attach a security policy to lambda function that allows it access to sagemaker endpoints.By attaching AmazonSageMakerFullAccess it grants this Lambda function access to the endpoints and other sagemaker services.

Go to Anything (Ctrl-P)

lambdafunction - /

lambda\_function.py

```

1 import base64
2 import logging
3 import json
4 import boto3
5 #import numpy
6 logger = logging.getLogger(__name__)
7 logger.setLevel(logging.DEBUG)
8
9 print('Loading Lambda function')
10
11 runtime=boto3.Session().client('sagemaker-runtime')
12 endpoint_Name='pytorch-inference-2022-11-05-20-43-15-461'#'BradTestEndpoint'
13
14 def lambda_handler(event, context):
15
16     #x=event['content']
17     #aa=x.encode('ascii')
18     #bs=base64.b64decode(aa)
19     print('Context:',context)
20     print('EventType:',type(event))
21     bs=event
22     runtime=boto3.Session().client('sagemaker-runtime')
23
24     response=runtime.invoke_endpoint(EndpointName=endpoint_Name,
25                                     ContentType="application/json",
26                                     Accept='application/json',
27                                     #Body=bytearray(x)
28                                     Body=json.dumps(bs))
29
30     result=response['Body'].read().decode('utf-8')
31     sss=json.loads(result)
32
33     return {
34         'statusCode': 200,
35         'headers': { 'Content-Type' : 'text/plain', 'Access-Control-Allow-Origin' : '*' },
36         'type-result':str(type(result)),
37         'Content-Type-In':str(context),
38     }

```

12:49 Python Spaces: 4

**Permissions policies (2) Info**

You can attach up to 10 managed policies.

Filter policies by property or policy name and press enter.

1

Policy name	Type	Description
AWSLambdaBasicExecutionRole-1c351308-ee79-4e48-b2ed-9592f7be94...	Customer managed	
AmazonSageMakerFullAccess	AWS managed	Provides full access to Arr

Test Event { "url": "<https://s3.amazonaws.com/cdn-origin-etr.akc.org/wp-content/uploads/2017/11/20113314/Carolina-Dog-standing-outdoors.jpg>"

}

Response (Predicted values)

Test Event Name Image-class-test

```
Response { "statusCode": 200, "headers": { "Content-Type": "text/plain", "Access-Control-Allow-Origin": "*" }, "type-result": "<class 'str'>", "Content-Type-In":  
"LambdaContext([aws_request_id=6389cc62-45ae-41ff-87ee-  
5eb8442d43b8,log_group_name=/aws/lambda/lambdafunction,log_stream_name=2022/11/  
/05/[$LATEST]f3938c0938764b45902528fc8dc3299c,function_name=lambdafunction,memory_limit_in_mb=128,function_version=$LATEST,invoked_function_arn=arn:aws:lambda:us-east-1:690349917069:function:lambdafunction,client_context=None,identity=CognitoIdentity([cognito_identity_id=None,cognito_identity_pool_id=None]))", "body": "[[-5.44008207321167, -4.952992916107178, -1.5317052602767944, -0.5046547651290894, -3.427722454071045, -3.5470330715179443, -2.1158523559570312, -1.5338290929794312, -7.150924205780029, 0.23522217571735382, -0.6767486929893494, -4.5792131423950195, -2.448819875717163, 1.5442739725112915, -4.79373025894165, -3.6146109104156494, -7.377564907073975, -2.806501865386963, -3.56970477104187, 0.6937685608863831, -4.370532989501953, -2.9022064208984375, -6.072688102722168, -4.791072368621826, -4.495667934417725, -4.914665699005127, -0.3236158490180969, -1.770508885383606, -5.172568321228027, -2.3332908153533936, -4.425060272216797, -2.528372049331665, -5.196396350860596, -1.9796003103256226, -8.67808723449707, -7.016977310180664, -4.221601486206055, -1.8513274192810059, -1.2895582914352417, -2.932392120361328, -2.3384647369384766, -3.72230863571167, -0.4528353810310364, -2.516489267349243, -0.7855870127677917, -7.66058874130249, -1.2515116930007935, 0.217243492603302, -3.0159168243408203, -1.0865718126296997, -0.9226453304290771, -6.505958557128906, -7.088664531707764, -2.6274538040161133, -5.69277811050415, -2.5954782962799072, -4.136444091796875, -6.413861274719238, -2.9214165210723877, -1.8026350736618042, -7.051815986633301, -6.990673542022705, -7.743916034698486, -6.82808780670166, -2.714198112487793, -6.730476379394531, 0.8651849031448364, -5.2978010177612305, -1.652145504951477, -0.9539076685905457, 0.27124127745628357, -3.6734120845794678, -5.905435562133789, -3.464625358581543, -5.734074115753174, -1.474758267402649, -6.283289432525635, -1.4405708312988281, -3.6478824615478516, -3.242371082305908, -1.1309401988983154, -4.781346321105957, -0.8546137809753418, -0.9007806777954102, -6.97314977645874, -5.638144493103027, -1.275240421295166, -6.873162269592285, -3.3538429737091064, -2.3007302284240723, -5.834065914154053, -3.1364731788635254, -2.340365409851074, -6.831973552703857, -3.429497241973877, -3.3244616985321045, -3.2669477462768555, -5.467846393585205, -5.630269527435303, -6.411980152130127, -7.864996433258057, -3.741847515106201, -2.0103819370269775, -5.227265357971191, -5.439854145050049, -5.761460304260254, -3.2689850330352783, -2.0540072917938232, -1.9230411052703857, -0.7347715497016907, -3.558453321456909, -1.4514901638031006, -6.587024211883545, -5.59173583984375, -5.957094669342041, -2.426581621170044, -7.154305934906006, -0.4399915337562561, -4.310878276824951, -0.7948324680328369, -2.5780420303344727, -4.45313835144043, -2.9790825843811035, -3.404550313949585, -6.894444942474365, -5.507934093475342, -3.2638163566589355, -2.16451096534729, -3.645242691040039, -5.3472371101379395, -5.278973579406738, -1.2079694271087646,
```

-5.8689680099487305]]]" } These are measurement of likelihood from each types of dogbreeds from the specified list of classes.

### **Any other Vulnerabilities that needs to be addressed.**

`FULLAccess` policies attached here too permissive. this endpoint can be secured. Less permissible policies can be attached, also for serverless applications like this, the preferred way to serve a backend application publicly is to use API Gateway. This can help you protect an API from malicious users or spikes in traffic.

Also using Amazon CloudWatch Events you can monitor unwanted users that logged on your endpoint.

## **Concurrency and auto-scaling**

### **Concurrency**

Concurrency refers to the ability of a lambda function to serve multiple requests simultaneously. Unreserved account concurrency 997 thus amount of lambda instances that I can use if I want to.

Provision concurrency 3 reserved concurrency. Provision concurrency means that I am creating instances that will always be on and always be there to reply to requests.



**Concurrency** Edit

Function concurrency: Unreserved account concurrency: 997

Use unreserved account concurrency

**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](#)

Refresh Edit Remove Add

Qualifier	Type	Provisioned concurrency	Status	Details
1	version	3	Ready	-

I used provision concurrency allowing 3 instance to be on always a this is a low traffic application.

Execution results Status: Succeeded Max memory used: 68 MB Time: 991.73 ms

**Test Event Name**  
Image-class-test

**Response**

```
{
  "statusCode": 200,
  "headers": {
    "Content-Type": "text/plain",
    "Access-Control-Allow-Origin": "*"
  },
  "type-result": "<class 'str'>",
  "Content-Type-In": "LambdaContext([aws_request_id=188dff98-413b-4659-8060-baa420a8e398, log_group_name=/aws/lambda/lambdafunction, log_stream_n",
  "body": "[[-5.44008207321167, -4.952992916107178, -1.5317052602767944, -0.5046547651290894, -3.427722454071045, -3.5470330715179443, -2.11585
```

**Function Logs**

```
Loading Lambda function
START RequestId: 188dff98-413b-4659-8060-baa420a8e398 Version: $LATEST
Context:: LambdaContext([aws_request_id=188dff98-413b-4659-8060-baa420a8e398, log_group_name=/aws/lambda/lambdafunction, log_stream_name=2022/11
EventType:: <class 'dict'>
END RequestId: 188dff98-413b-4659-8060-baa420a8e398
REPORT RequestId: 188dff98-413b-4659-8060-baa420a8e398 Duration: 991.73 ms Billed Duration: 992 ms Memory Size: 128 MB Max Memory Used: 68 MB
```

**Request ID**  
188dff98-413b-4659-8060-baa420a8e398

## Auto-Scaling

**Endpoint runtime settings** Update weights Update instance count Configure auto scaling

Variant name ▲	Current weight ▼	Desired weight	Instance type ▼	Elastic Inference	Current instance count ▼	Desired instance count ▼	Instance min - max	Automatic scaling
AllTraffic	1	1	ml.m5.large	-	1	1	1 - 3	Yes

Minimum instance count=1 and Maximum instance count=3 Target-value: Our endpoint need to decide when to reproly to traffic. Here if we recieve 100 invokations that is simultaneous, that will be a signal that it needs to create a new instance.

So this a very responsive endpoint

Scale in cool down. will only take 30 seconds of elevated traffic for the endpoint to deploy more instances, and Scale out cool down is 30 seconds of decreased traffic for the endpoint to delete its instances.

## References

---

1. [Udacity AWS ML Engineer](#)
2. [SageMaker Pricing](#)
3. [Using Pytorch](#)
4. [Sagemaker runtime Lambda](#)
5. [AWS Machine Learning WorkFlow](#)
6. [Json.load\(\) geeksforgeeks](#)
7. [public endpoints](#)
8. [SageMaker Workshop](#)