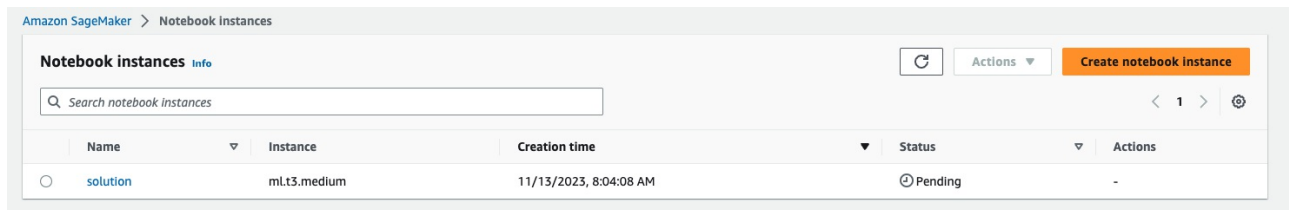


operationalized-dog-classifier

A dog classifier that will take advantage of multi-instance training and a scaled up inference endpoint.

Notebook setup

I choose to use an ml.t3.medium notebook type because it will be generally lower cost than using a larger instance. For just code editing using a smaller instance will save money, If I need something more robust for training or inference I will deploy that separately.

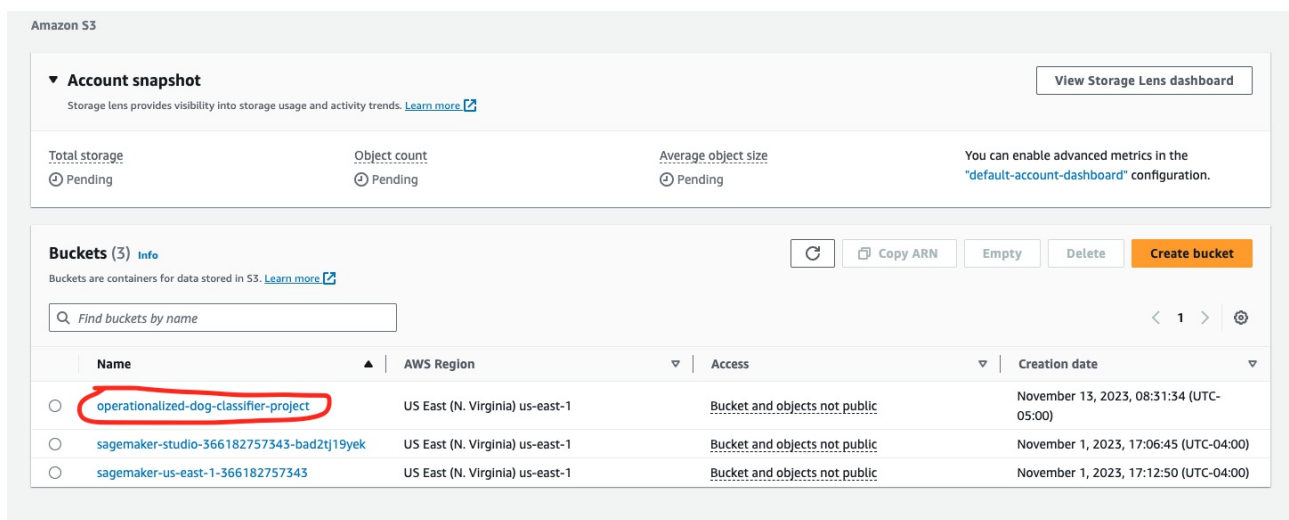


The screenshot shows the Amazon SageMaker Notebook Instances page. At the top, there's a search bar and a 'Create notebook instance' button. Below is a table with one instance named 'solution' of type 'ml.t3.medium', created on 11/13/2023 at 8:04:08 AM, with a status of 'Pending'.

Name	Instance	Creation time	Status	Actions
solution	ml.t3.medium	11/13/2023, 8:04:08 AM	Pending	-

Training and Deployment

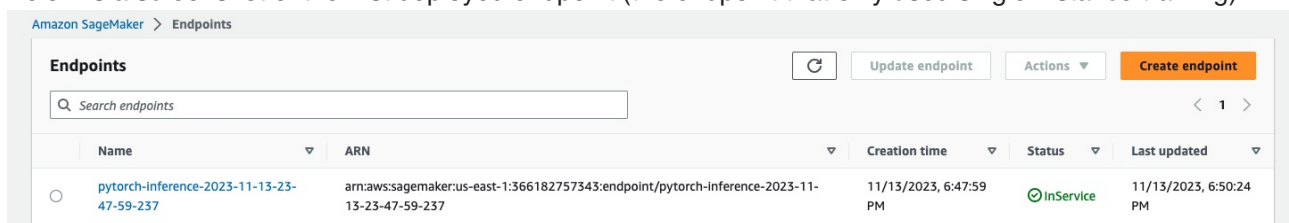
I created an S3 bucket using the sagemaker session object it is named "operationalized-dog-breed-classifier-project".



The screenshot shows the Amazon S3 Buckets page. It lists three buckets. The first bucket, 'operationalized-dog-classifier-project', is circled in red. It was created on November 13, 2023, at 08:31:34 UTC-05:00. The other two buckets are 'sagemaker-studio-366182757343-bad2tj19yek' and 'sagemaker-us-east-1-366182757343', both created on November 1, 2023.

Name	AWS Region	Access	Creation date
operationalized-dog-classifier-project	US East (N. Virginia) us-east-1	Bucket and objects not public	November 13, 2023, 08:31:34 (UTC-05:00)
sagemaker-studio-366182757343-bad2tj19yek	US East (N. Virginia) us-east-1	Bucket and objects not public	November 1, 2023, 17:06:45 (UTC-04:00)
sagemaker-us-east-1-366182757343	US East (N. Virginia) us-east-1	Bucket and objects not public	November 1, 2023, 17:12:50 (UTC-04:00)

Below is a screenshot of the first deployed endpoint (the endpoint that only used single instance training)



The screenshot shows the Amazon SageMaker Endpoints page. It lists one endpoint named 'pytorch-inference-2023-11-13-23-47-59-237'. It was created on 11/13/2023 at 6:47:59 PM and is currently 'InService'.

Name	ARN	Creation time	Status	Last updated
pytorch-inference-2023-11-13-23-47-59-237	arn:aws:sagemaker:us-east-1:366182757343:endpoint/pytorch-inference-2023-11-13-23-47-59-237	11/13/2023, 6:47:59 PM	InService	11/13/2023, 6:50:24 PM

Below is a screenshot of the 2nd deployed endpoint (the endpoint that only used multi instance training)

Amazon SageMaker > Endpoints

Endpoints

Update endpoint

Actions

Create endpoint

Search endpoints

1

	Name	ARN	Creation time	Status	Last updated
<div></div>	pytorch-inference-2023-11-14-13-03-52-459	arn:aws:sagemaker:us-east-1:366182757343:endpoint/pytorch-inference-2023-11-14-13-03-52-459	11/14/2023, 8:03:53 AM	<div><div></div>InService</div>	11/14/2023, 8:06:22 AM

Cost Effective Training (using EC2)

An alternative to using sagemaker is to use an EC2 instance directly (which is more cost effective). For this part of the project I setup an EC2 instance in the console.

▼ Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Quick Start

Amazon Linux
aws

macOS
Mac

Ubuntu
ubuntu

Windows
Microsoft

Red Hat
Red Hat

SUSE Li
SUS

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

Amazon Machine Image (AMI)

Deep Learning AMI GPU PyTorch 2.0.1 (Amazon Linux 2) 20231107
ami-09e2639b59ee94f7c (64-bit (x86))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Supported EC2 instances: P5, P4d, P4de, P3, P3dn, G5, G4dn, G3. Release notes:
<https://docs.aws.amazon.com/dlami/latest/devguide/appendix-ami-release-notes.html>

Architecture	AMI ID
64-bit (x86)	ami-09e2639b59ee94f7c

Verified provider

▼ Summary

Number of instances Info

Software Image (AMI)
Supported EC2 instances: P5, P4d, P4de, P3, P3dn, G5, G4dn, G3. Release notes:
<https://docs.aws.amazon.com/dlami/latest/devguide/appendix-ami-release-notes.html>
ami-09e2639b59ee94f7c

Virtual server type (instance type)
t3.medium

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 45 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million I/Os, 1 GB of

I choose this particular size of instance because I wanted it to be large enough not to take too long for training but not too large to cost too much. I also picked the "Deep Learning AMI GPU PyTorch 2.0.1 (Amazon Linux 2) 20231107" AMI because it would have the pytorch packages I needed already installed on the machine.

Here is a screenshot of the model location after it had been saved.

```
[root@ip-172-31-37-94 TrainedModels]# ls
model.pth
[root@ip-172-31-37-94 TrainedModels]#
```

Training a model this way is more cost effective but lacks certain convinces. For example if I wanted to deploy the model I trained in the EC2 instance I would then need to write a script to manually write the model artifact to an S3 instance (which would require me to setup permissions), then within the console (and sagemaker most likely) choose the model artifact to deploy the instance to an endpoint. This feels like something I would only do for very specialized instances when I had a very small budget.

Operationalizing Model

Creating Lambda

The lambda file basically calls the `invoke_endpoint` using a variable `endpoint_Name` which is naturally the endpoint I deployed via the notebook and sagemaker.

Testing Lambda Endpoint

For this part of the exercise I had to navigate to the IAM interface, find my inference lambda role then attach the AmazonSageMakerFullAccess policy. Attaching this policy is fine for this exercise but wouldn't be my first choice for a real world situation. Ideally in more real world situations we would like to limit access of the role, so if I could create a role that simply allowed the lambda to invoke endpoints, that would be a better policy to attach. Below is a screenshot of the AmazonSageMakerFullAccess attached to my lambdas role:

The screenshot shows the AWS IAM console interface for the role 'inference-lambda-role-6e156078'. The 'Summary' tab is active, displaying the role's creation date (November 17, 2023, 17:55 UTC-05:00), its ARN (arn:aws:iam::366182757343:role/service-role/inference-lambda-role-6e156078), and its maximum session duration (1 hour). Below the summary, the 'Permissions policies' tab is selected, showing a list of attached policies. The policies listed are 'AmazonSageMaker-ExecutionPolicy-20231101T170683' (Customer managed), 'AmazonSageMakerFullAccess' (AWS managed), and 'AWSLambdaBasicExecutionRole-384cd850-53ff-40a4-8...' (Customer managed). The 'AmazonSageMakerFullAccess' policy is highlighted, indicating it is the one being managed.

Policy name	Type	Attached entities
AmazonSageMaker-ExecutionPolicy-20231101T170683	Customer managed	2
AmazonSageMakerFullAccess	AWS managed	2
AWSLambdaBasicExecutionRole-384cd850-53ff-40a4-8...	Customer managed	1

the results of my endpoint invocation was:

The screenshot shows the AWS Lambda console interface for a function execution. The 'Details' tab is active, displaying the execution log and summary. The log shows a successful execution with a status code of 200 and a content type of 'text/plain'. The summary section provides details about the execution, including the request ID, duration, and resources used.

Summary	Execution details
Code SHA-256 DvcFmA2Uiz7L7WIVVCCvSwatUFomX3LNfKvUHiv+E=	Execution time 52 seconds ago (November 17, 2023 at 06:06 PM EST)
Request ID caaab536-dd35-4b49-81b7-1aafe1175754	Function version \$LATEST
Duration 1732.39 ms	Billed duration 1733 ms
Resources configured 128 MB	Max memory used 83 MB

```
{
  "statusCode": 200,
  "headers": {
    "Content-Type": "text/plain",
    "Access-Control-Allow-Origin": "*"
  },
  "type-result": "<class 'str'>",
  "Content-Type-In": "LambdaContext([aws_request_id=2238049f-c452-4a4e-be15-1aec928cc22e,log_group_name=/aws/lambda/inference-lambda,log_stream_name=2023/11/17/[$LATEST]ab70ffa2e1e14d33adcbd8cb263aab1b,function_name=inference-lambda,memory_limit_in_mb=128,function_version=$LATEST,invoked_function_arn=arn:aws:lambda:us-east-1:366182757343:function:inference-lambda,client_context=None,identity=CognitoIdentity([cognito_identity_id=None,cognito_identity_pool_id=None]))",
  "body": "[[0.5792363882064819, 0.7636755108833313, 0.1653096079826355, 0.4950832724571228, 0.8176630735397339, 0.7240684628486633, 0.5346078872680664, 0.3285769522190094, -0.10185138136148453, 0.2920922338962555, 0.6372276544570923, 0.5812382698059082, -0.08509815484285355, 0.37544023990631104, 0.8461633324623108, 0.021534280851483345, 0.10557816922664642, 0.19371265172958374, 0.14003700017929077, 0.3003043234348297, 0.41529303789138794, 0.00643235445022583, 0.1804734468460083, 0.6603132486343384, -0.030782438814640045, -0.09654703736305237, 0.5044122338294983, -0.022909216582775116, 0.6699318885803223, 0.615969181060791, 0.6364791393280029, 0.7549263834953308, -0.024541690945625305, 0.07810435444116592, 0.3004719018936157, 0.5528291463851929, 0.3313681483268738, 0.6378064751625061, 0.5468197464942932, 0.40666159987449646, 0.762763500213623, 0.6268402338027954, 0.35572952032089233, 0.5318848490715027, 0.17492461204528809, 0.7221673727035522, 0.6038680076599121, 0.2621477246284485, 0.40824612975120544, 0.39500364661216736, 0.714881181716919, 0.6262304186820984, 0.40533313155174255, 0.6996309161186218, 0.26447486877441406, 0.5697035789489746, 0.719495415687561, 0.5230414271354675, 0.37702247500419617, 0.45582693815231323, 0.19721060991287231, 0.23548755049705505, 0.13242028653621674, 0.1574971228837967, 0.37834662199020386, -0.09755107760429382, -0.3490096926689148, 0.5763421654701233, 0.25023671984672546, 0.016184493899345398, 0.5006518363952637, 0.36676883697509766, 0.2488367110490799, 0.33383965492248535, 0.0741475522518158, 0.2896389365196228, 0.051754407584667206, 0.03837300091981888, 0.10128553956747055, 0.09817423671483994, 0.5892516374588013, 0.5812305808067322, 0.35266634821891785, -0.10847659409046173, -0.21791903674602509, -0.007983347401022911, 0.5052909255027771, 0.47968733310699463, 0.48889875411987305, 0.6120469570159912, 0.28052493929862976, 0.2681379020214081, -0.11890796571969986, 0.1654740422964096, -0.0032714693807065487, 0.17335288226604462, -0.23192933201789856, 0.0732516348361969, -0.49823227524757385, -0.1330939382314682, 0.26317697763442993, -0.4642694294452667, 0.2237107902765274, 0.22803275287151337, -0.4031201899051666, 0.07531560212373734, 0.4183971881866455, -0.08876407891511917, 0.13764815032482147, -0.25683286786079407, 0.43035510182380676, 0.22272832691669464, 0.08310887962579727, 0.23699021339416504, 0.5412672162055969, -0.4890146553516388, 0.6523604989051819, 0.4549826979637146, -0.5567016005516052, -0.11672106385231018, -0.3893718719482422, 0.07198801636695862, 0.22364474833011627, 0.4074152410030365, -0.22290650010108948, -0.6037143468856812, -0.004977082833647728, -0.24122896790504456, 0.12222044169902802, -0.2430066615343094, 0.12427321076393127, -0.8702178597450256, -0.14377853274345398]]]"
}
```

Concurrency and Auto-Scaling

For Concurrency of my lambda function I simply configured it to reserve two instances, as this project will have no request volume I figured the cheapest option was best, obviously if I were receiving a larger number of requests I would need more instances reserved.

For my sagemaker endpoint Autoscalling I just choose to allow it to autoscale between one and two endpoints (again low volume) and have a scale in and scale out values at their defaults as I don't anticipating needing to scale up quickly (scale in) or scale down quickly (scale out).