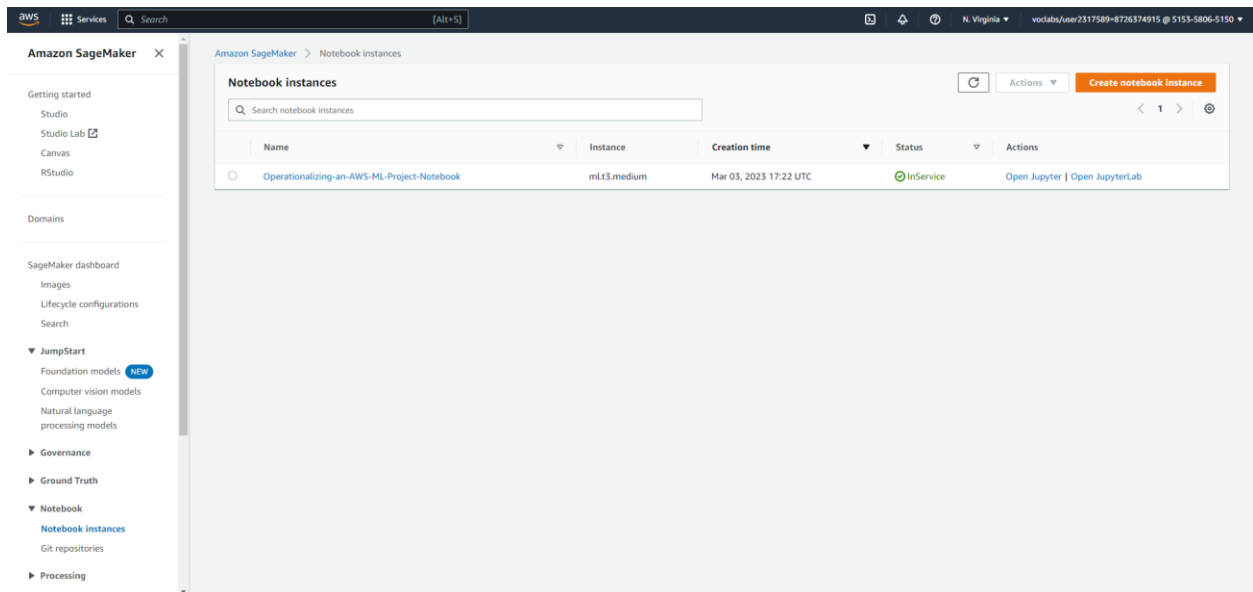


## Initial setup, training and deployment

### 1- Initial Setup

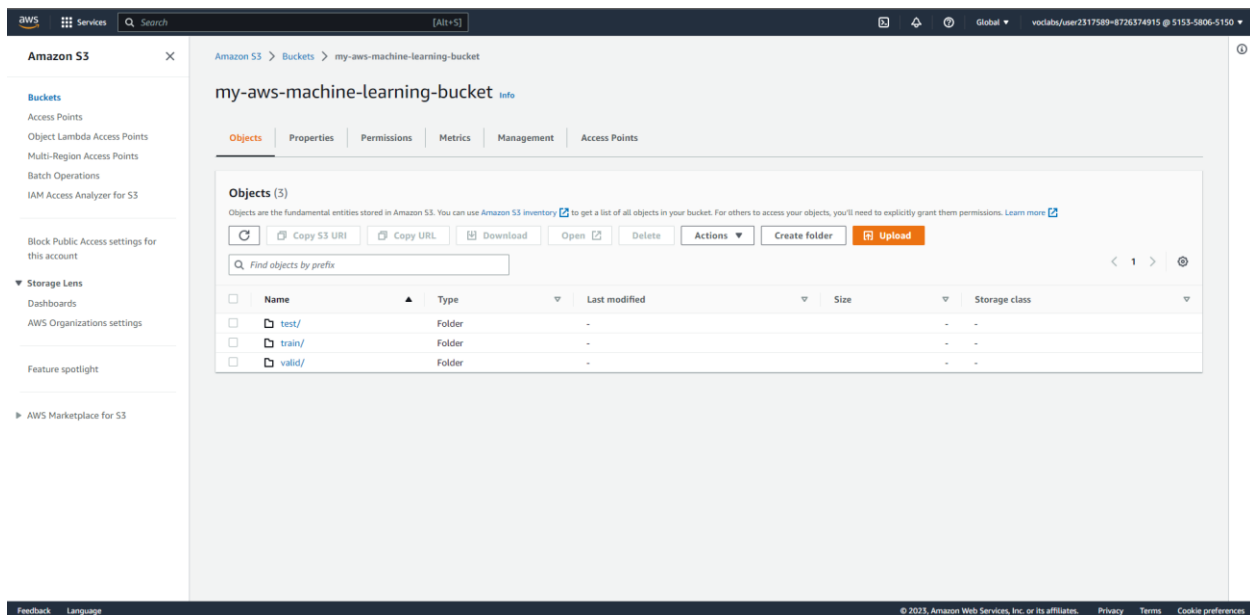
I chose the instance type "ml.t2. medium" for the Notebook instance because:

- I don't need a very powerful CPU in terms of computation and high RAM to run the notebook cells.
- In order to avoid high costs, we should select a laptop with a low cost per hour and reasonably good CPU and RAM, because we need to keep this laptop running for a long time while we work on the project.
- Looking at the type of instance and its price, "ml.t2. medium" has the lowest possible cost for our work.



### 2- Download data to an S3 bucket

Here is the Cloud bucket where our dataset resides so all our work will be saved and our model will train from.



### 3- Training and Deployment

Here we launched a hyperparameter tuning job to try to look for the best parameters that our model can train with, and we are using an “ml. g4dn.xlarge” for faster training.

pytorch-training-230303-1841 Stop tuning job

**Hyperparameter tuning job summary**

Name pytorch-training-230303-1841	Status Completed	Approx. total training duration 40 minute(s)
ARN arn:aws:sagemaker-us-east-1:515358065150:hyper-parameter-tuning-job/pytorch-training-230303-1841	Creation time Mar 03, 2023 18:41 UTC	
	Last modified time Mar 03, 2023 19:27 UTC	

Best training job | **Training jobs** | Training job definitions | Tuning Job configuration | Tags

**Training job status counter**

Completed 2 | In Progress 0 | Stopped 0 | Failed 0 ( Retryable: 0, Non-retryable: 0 )

**Training jobs**

Sorting by objective metric value will display only jobs that have metric values.

Search training jobs

	Name	Status	Objective metric value	Creation time	Training Duration
<input type="radio"/>	pytorch-training-230303-1841-002-88cc3128	Completed	1019	Mar 03, 2023 19:06 UTC	20 minute(s)
<input type="radio"/>	pytorch-training-230303-1841-001-07e2593e	Completed	150	Mar 03, 2023 18:41 UTC	20 minute(s)

#### Describe the tuning results

```
[7]: exp = HyperparameterTuningJobAnalytics(
      hyperparameter_tuning_job_name='pytorch-training-230303-1841')
      jobs = exp.dataframe()
      jobs.sort_values('FinalObjectiveValue', ascending=0)
```

	batch_size	learning_rate	TrainingJobName	TrainingJobStatus	FinalObjectiveValue	TrainingStartTime	TrainingEndTime	TrainingElapsedTimeSeconds
0	"256"	0.022315	pytorch-training-230303-1841-002-88cc3128	Completed	1019.0	2023-03-03 19:06:31+00:00	2023-03-03 19:26:33+00:00	1202.0
1	"32"	0.017952	pytorch-training-230303-1841-001-07e2593e	Completed	150.0	2023-03-03 18:43:16+00:00	2023-03-03 19:03:27+00:00	1211.0

After training the model with the best parameters and also changed the instance type to "ml.m5.2xlarge". here we have the result.

Amazon SageMaker > Training jobs > dog-pytorch-2023-03-20-47-30-344

dog-pytorch-2023-03-20-47-30-344

Clone Create model package Stop Create model

### Job settings

Job name dog-pytorch-2023-03-20-47-30-344	Status Completed <a href="#">View History</a>	SageMaker metrics time series Enabled	IAM role ARN <a href="#">arnaws:iam::515358065150:role/service-role/AmazonSageMaker-ExecutionRole-20230101T201145</a>
ARN arnaws:sagemaker:us-east-1:515358065150:training-job/dog-pytorch-2023-03-20-47-30-344	Creation time Mar 03, 2023 20:47 UTC	Training time (seconds) 801	
	Last modified time Mar 03, 2023 21:02 UTC	Billable time (seconds) 801	
		Managed spot training savings 0%	
		Tuning job source/parent -	

### Algorithm

Algorithm ARN -	Additional volume size (GB) 30	Maximum wait time for managed spot training(s) -	Volume encryption key -
Training image 763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training:1.4.0-cpu-py3	Maximum runtime (s) 86400	Managed spot training Disabled	
Input mode File			

Instance group	Instance type	Instance count	Keep alive period
-	ml.m5.2xlarge	1	-

## 4- Multi-instance training

Here we altered the code for multi-instance training by increasing the instance count to 4.

Amazon SageMaker > Training jobs > dog-pytorch-2023-03-21-19-45-746

dog-pytorch-2023-03-21-19-45-746

Clone Create model package Stop Create model

### Job settings

Job name dog-pytorch-2023-03-21-19-45-746	Status Completed <a href="#">View History</a>	SageMaker metrics time series Enabled	IAM role ARN <a href="#">arnaws:iam::515358065150:role/service-role/AmazonSageMaker-ExecutionRole-20230101T201145</a>
ARN arnaws:sagemaker:us-east-1:515358065150:training-job/dog-pytorch-2023-03-21-19-45-746	Creation time Mar 03, 2023 21:19 UTC	Training time (seconds) 752	
	Last modified time Mar 03, 2023 21:34 UTC	Billable time (seconds) 752	
		Managed spot training savings 0%	
		Tuning job source/parent -	

### Algorithm

Algorithm ARN -	Additional volume size (GB) 30	Maximum wait time for managed spot training(s) -	Volume encryption key -
Training image 763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training:1.4.0-cpu-py3	Maximum runtime (s) 86400	Managed spot training Disabled	
Input mode File			

Instance group	Instance type	Instance count	Keep alive period
-	ml.m5.2xlarge	4	-

## 5- Deploying endpoints for single and multi-instance training:

single instance endpoint: "pytorch-inference-2023-03-04-21-11-25-196" and multi-instance endpoint: "pytorch-inference-2023-03-04-21-44-05-645"

Amazon SageMaker > Endpoints

Endpoints

Search endpoints

Update endpoint Actions Create endpoint

	Name	ARN	Creation time	Status	Last updated
	pytorch-inference-2023-03-04-21-44-05-645	arn:aws:sagemaker:us-east-1:515358065150:endpoint/pytorch-inference-2023-03-04-21-44-05-645	Mar 04, 2023 21:44 UTC	InService	Mar 04, 2023 21:46 UTC
	pytorch-inference-2023-03-04-21-11-25-196	arn:aws:sagemaker:us-east-1:515358065150:endpoint/pytorch-inference-2023-03-04-21-11-25-196	Mar 04, 2023 21:11 UTC	InService	Mar 04, 2023 21:13 UTC

## EC2 Setup

### 1- Preparing for EC2 model training

I used a t2.xlarge instance with the Deep Learning AMI GPU Pytorch 1.13.1 (Amazon Linux 2) Image. This setup has good performance and cost. After taking all the parameters into accounts for the training and duration, I've concluded that the T2 instance is the way to go by referring to AWS documentation:

“Amazon EC2 T2 instances are Burstable Performance Instances that provide a baseline level of CPU performance with the ability to burst above the baseline. T2 Unlimited instances can sustain high CPU performance for as long as a workload needs it.”

And the best to go for a medium-sized instance so that we don't have to pay more while we do any other work then training

Instances (1/1) Info

Find instance by attribute or tag (case-sensitive)

Instance state: running Clear filters

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 ...	Public IPv4 address	Elastic IP	IPv6 IPs
operationalizing-aws-project	i-025dbaadc86e0e11c	Running	t2.xlarge	2/2 checks passed	No alarms	us-east-1c	ec2-34-207-24...	34.207.244.3	-	-

Instance: i-025dbaadc86e0e11c (operationalizing-aws-project)

Auto-assigned IP addresses: 34.207.244.3 [Public IP]

IAM Role: -

Instance details Info

Platform: Linux/UNIX (Inferred)

Platform details: Linux/UNIX

Stop protection: Disabled

Instance auto-recovery: Default

AMI Launch index: 0

VPC ID: vpc-005867564a563b71f

Subnet ID: subnet-096f07d8b2a2b75cb

AMI ID: ic3550dee3f5b

Deep Learning AMI GPU PyTorch 1.13.1 (Amazon Linux 2) 20230226

Launch time: Sat Mar 04 2023 20:39:54 GMT+0100 (Central European Standard Time) (about 2 hours)

Lifecycle: normal

Key pair name: project4

AWS Compute Optimizer: running

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

Auto Scaling Group name: -

Monitoring: disabled

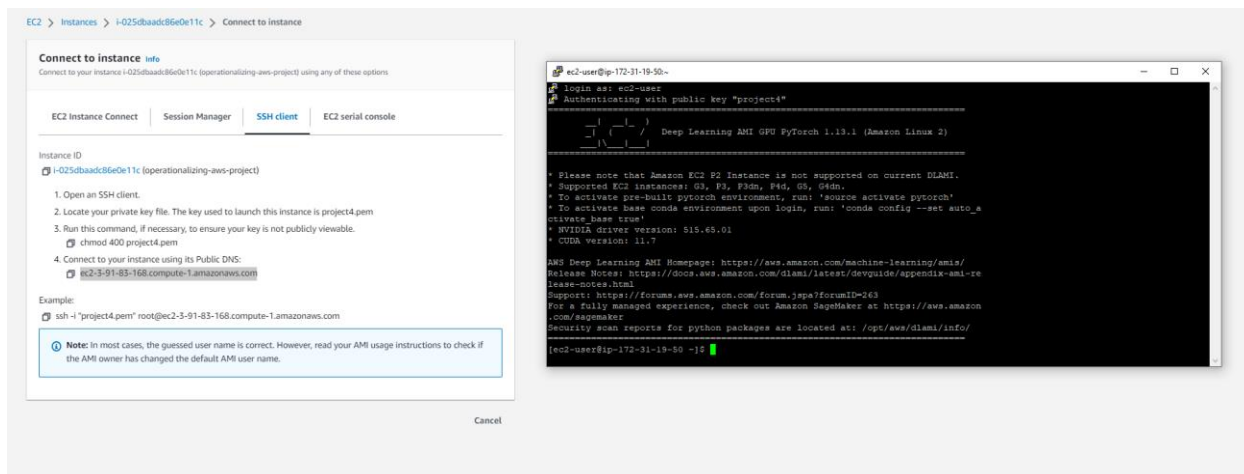
Termination protection: Disabled

AMI location: amazon/Deep Learning AMI GPU PyTorch 1.13.1 (Amazon Linux 2) 20230226

Stop-hibernate behavior: disabled

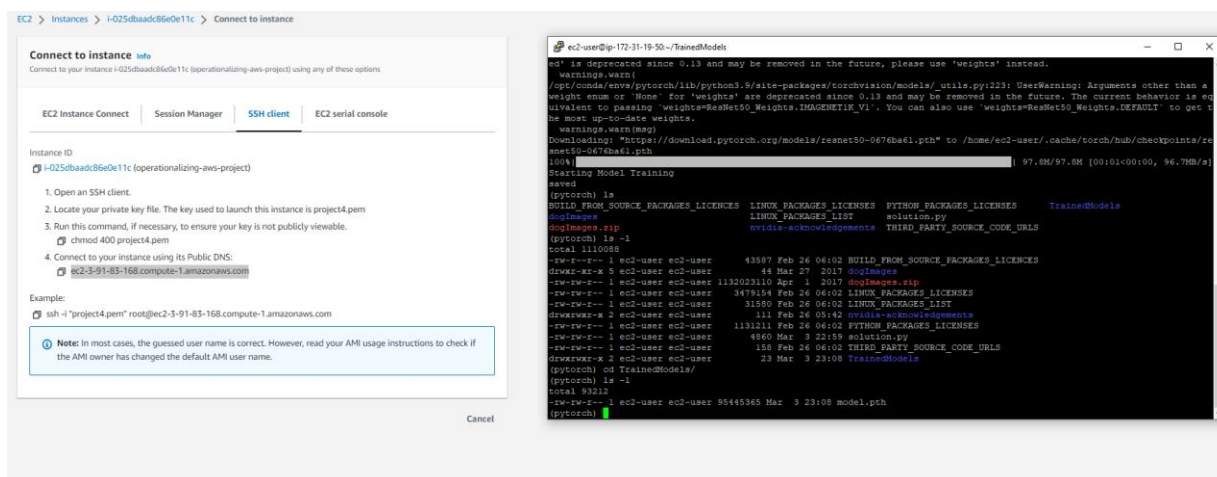
State transition reason: -

After launching the instance with Putty (an SSH Terminal), I've activated the “Pytorch” environment so we can run our code without any dependencies error.



## 2- Training and saving on EC2

After training the model was saved in the specified folder.



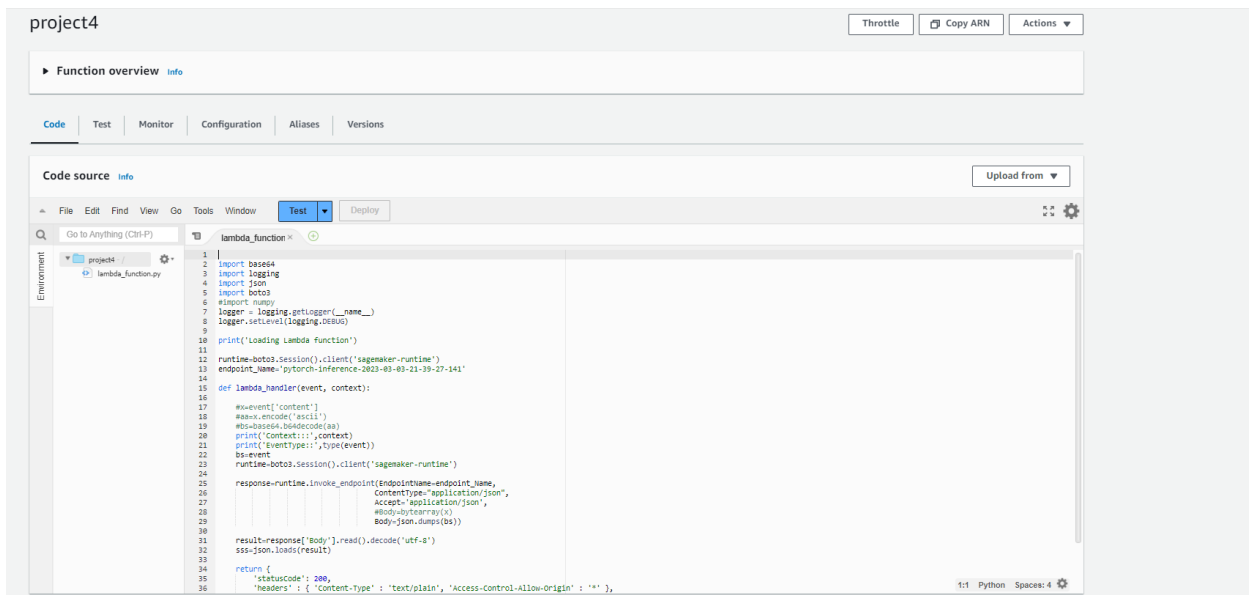
In the python code, we didn't need an Estimator or Tuner, all functions were handled by the code. and all variables like hyperparameters and output locations were hard-coded, so no arguments had to be passed to the script.

One thing is that deploying the trained model will take some work since the model was served locally.

## Setting up a Lambda function

Here, we will use this lambda function to invoke our multi-instance training endpoint. As with the endpoint invocation from our Notebook, we need to serialize our input to JSON Format and then pass it to the endpoint, after which we will dump the result.

Replacing the endpoint name with our multi-instance endpoint



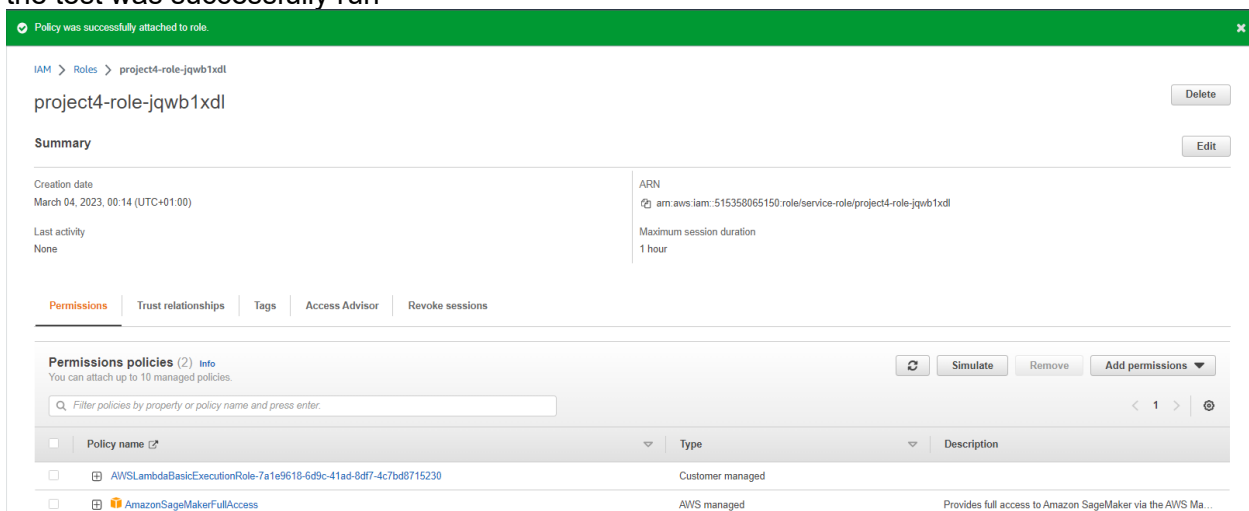
The screenshot shows the AWS Lambda console interface for a function named 'project4'. The 'Code source' tab is active, displaying a Python script. The script imports boto3, logging, json, and numpy. It defines a lambda\_handler function that takes an event and context as input. The handler prints the event, then uses boto3 to create a SageMaker runtime client. It then calls the 'invoke\_endpoint' method of the client, passing the endpoint name 'pytorch-inference-2023-03-21-141'. The response is read and decoded, and the result is returned as a JSON object. The console also shows the 'Function overview' tab with 'Throttle', 'Copy ARN', and 'Actions' buttons.

```
1 import boto3
2 import logging
3 import json
4 import numpy
5
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-2023-03-21-141'
13
14 def lambda_handler(event, context):
15
16     #event['content']
17     #event['content'].encode('ascii')
18     #event['content'].encode('utf-8')
19     print('Context:', context)
20     print('Event type:', type(event))
21     bs=event
22     runtime=boto3.session().client('sagemaker-runtime')
23
24     response=runtime.invoke_endpoint(endpoint_name=endpoint_name,
25                                     ContentType='application/json',
26                                     Accept='application/json',
27                                     Body=json.dumps(bs))
28
29     result=response['Body'].read().decode('utf-8')
30     json.loads(result)
31
32     return {
33         'statusCode': 200,
34         'headers': { 'Content-Type': 'text/plain', 'Access-Control-Allow-Origin': '*' },
35     }
```

## Lambda function security

### 1- Lambda function testing

after executing the lambda function event I got an error message, tracing the log I found it was an AccessDeniedException error. which means lambda didn't have access to Sagemaker resources, after modifying the lambda function role to have the "SageMakerFullAccess" policy. the test was successfully run



The screenshot shows the AWS IAM console interface for a role named 'project4-role-jqwb1xd1'. The 'Summary' tab is active, displaying the role's details. A green notification bar at the top indicates 'Policy was successfully attached to role.' The 'Permissions' tab is also visible, showing a list of attached policies. The policies listed are 'AWSLambdaBasicExecutionRole-7a1e9618-6d9c-41ad-8df7-4c7bd8715230' and 'AmazonSageMakerFullAccess'. The 'AmazonSageMakerFullAccess' policy is highlighted, showing its description: 'Provides full access to Amazon SageMaker via the AWS Ma...'

Policy was successfully attached to role.

project4-role-jqwb1xd1

Summary

Creation date: March 04, 2023, 00:14 (UTC+01:00)

Last activity: None

ARN: arn:aws:iam:515358065150:role/service-role/project4-role-jqwb1xd1

Maximum session duration: 1 hour

Permissions policies (2)

Policy name	Type	Description
AWSLambdaBasicExecutionRole-7a1e9618-6d9c-41ad-8df7-4c7bd8715230	Customer managed	
AmazonSageMakerFullAccess	AWS managed	Provides full access to Amazon SageMaker via the AWS Ma...

🟢 The test event **testing** was successfully saved.

► **Function overview** [info](#)

[Code](#) | [Test](#) | [Monitor](#) | [Configuration](#) | [Aliases](#) | [Versions](#)

**Code source** [info](#) Upload from ▼

File Edit Find View Go Tools Window **Test** Deploy

Go to Anything (Ctrl-P)

Environment

project4

lambda\_function.py

Execution results

Test Event Name: testing

Status: **Succeeded** | Max memory used: 75 MB | Time: 1358.01 ms

**Response**

```
{
  "statusCode": 200,
  "headers": {
    "Content-Type": "text/plain",
    "Access-Control-Allow-Origin": "*"
  },
  "type-result": "<class 'str'>",
  "Content-Type-In": "<bootstrap.LambdaContext object at 0x7fa2db902550>",
  "body": "[[-7.2419867515563965, -4.272039413452148, -4.949213981628418, -2.1663122177124023, -4.419911861419678, -5.0522284507751465, -2.7705905437469482, -1.6769870519638062, -4.451426029205322, -0.14113]]"
```

**Function Logs**

```
START RequestId: 1d712b44-3436-4bea-bcea-d8a51bb8249e Version: $LATEST
Context:: bootstrap.LambdaContext object at 0x7fa2db902550
Event type:: <class 'dict'>
END RequestId: 1d712b44-3436-4bea-bcea-d8a51bb8249e
REPORT RequestId: 1d712b44-3436-4bea-bcea-d8a51bb8249e Duration: 1358.01 ms Billed Duration: 1359 ms Memory Size: 128 MB Max Memory Used: 75 MB
```

**Request ID**  
1d712b44-3436-4bea-bcea-d8a51bb8249e

a lambda function having “Full Access” to the SageMaker resources is not an ideal type of security I want to deploy in the future. the policies should be stricter and carefully considered. and each lambda function is allowed to query only one endpoint.

## Concurrency and Auto-scaling

### 1- Concurrency

First, I created a Version Config for the lambda function. I chose reserved concurrency because it would decrease latency issues in high-traffic situations above normal. The lambda function would be able to manage up to 5 simultaneous requests.

Lambda > Functions > project4

**project4** Throttle Copy ARN Actions ▼

► **Function overview** [info](#)

[Code](#) | [Test](#) | [Monitor](#) | **[Configuration](#)** | [Aliases](#) | [Versions](#)

**General configuration**

- Triggers
- Permissions
- Destinations
- Function URL
- Environment variables
- Tags
- VP
- Monitoring and operations tools
- Concurrency**
- Asynchronous invocation
- Code signing
- Database proxies
- File systems
- State machines

**Concurrency** Edit

Function concurrency: Use reserved concurrency

Reserved concurrency: 5

**Provisioned concurrency configurations**

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

Find configuration

Qualifier	Type	Provisioned concurrency	Status	Details
No configurations				

Add configuration

## 2- Auto-scaling

When configuring auto-scaling, I added these parameters to the endpoint runtime settings:

- A maximum number of instances of 3, a minimum of 1, and the target metric "SageMakerVariantInvocationsPerInstance" of 10 because I see it will fulfill the present requirement of our project also I added an input and output cooling time of 30 seconds.

Endpoint runtime settings										Update weights	Update instance count	Configure auto scaling
	Variant name ▲	Current weight ▼	Desired weight	Elastic Inference	Instance type ▼	Current instance count ▼	Desired instance count ▼	Instance min - max	Automatic scaling			
<input type="radio"/>	AllTraffic	1	1	-	ml.m5.large	1	1	1 - 3	Yes			