

operationalizing-an-aws-ml-project

Dog Image Classification

In this project, you will accomplish the following tasks:

1. Utilize SageMaker to train and deploy a model, selecting the most suitable instance types. Configure multi-instance training within your SageMaker notebook.
2. Modify your SageMaker notebooks to facilitate training and deployment on EC2 instances.
3. Establish a Lambda function associated with your deployed model. Configure auto-scaling for the deployed endpoint and manage concurrency for the Lambda function.
4. Implement proper security measures for your machine learning pipeline.

Project Steps:

Step 1: Training and deployment on Sagemaker

- Created sagemaker notebook instance I have used ml.t3.medium as this is suffiecnt to run my notebook.

Amazon SageMaker

Notebook instances

Notebook instances

Info

Refresh

Actions

Create notebook instance

Search notebook instances

< 1 > Settings

	Name	Instance	Creation time	Status	Actions
	solution	ml.t3.medium	5/16/2023, 10:31:56 AM	InService	Open Jupyter Open JupyterLab

- Created S3 bucket for the job (udacitysolution-418228298154)

Buckets (6)

Info

Refresh

Copy ARN

Empty

Delete

Create bucket

Buckets are containers for data stored in S3. Learn more

Find buckets by name

< 1 > Settings

	Name	AWS Region	Access	Creation date
	udacitysolution-418228298154	US East (N. Virginia) us-east-1	Objects can be public	May 16, 2023, 10:35:29 (UTC+07:00)

- Executed single instance training (1 epoch because of budget constraints)

CloudWatch

Favorites and recents

Dashboards

Alarms

▼ Logs

Log groups

Logs Insights

Metrics

X-Ray traces

Events

Application monitoring

Insights

Settings

Getting Started

/aws/sagemaker/TrainingJobs

Actions

View in Logs Insights

Search log group

▼ Log group details

ARN
arn:aws:logs:us-east-1:418228298154:log-group:/aws/sagemaker/TrainingJobs:*

Creation time
4 hours ago

Retention
Never expire

Stored bytes
-

Metric filters
0

Subscription filters
0

Contributor Insights rules
-

Data protection - new
Inactive

Sensitive data found - new
-

KMS key ID
-

Log streams

Metric filters

Subscription filters

Contributor Insights

Tags

Data protection - new

Log streams (30)

dog-pytorch-2023-05-16-06-12-58-987

1 match

Exact match

Show expired

Info

1

Log stream

Last event time

dog-pytorch-2023-05-16-06-12-58-987/algo-1-1684217657

2023-05-16 13:15:35 (UTC+07:00)

- Executed multi-instance training (4 instances, 1 epoch because of budget constraints)

CloudWatch

Favorites and recents

Dashboards

Alarms

▼ Logs

Log groups

Logs Insights

Metrics

X-Ray traces

Events

Application monitoring

Insights

Settings

Getting Started

CloudWatch > Log groups > /aws/sagemaker/TrainingJobs

/aws/sagemaker/TrainingJobs

Actions

View in Logs Insights

Search log group

▼ Log group details

ARN
arn:aws:logs:us-east-1:418228298154:log-group:/aws/sagemaker/TrainingJobs:*

Creation time
4 hours ago

Retention
Never expire

Stored bytes
-

Metric filters
0

Subscription filters
0

Contributor Insights rules
-

Data protection - new
Inactive

Sensitive data found - new
-

KMS key ID
-

Log streams

Metric filters

Subscription filters

Contributor Insights

Tags

Data protection - new

Log streams (30)

dog-pytorch-2023-05-16-06-16-14-289

4 matches

Exact match

Show expired

Info

1

Log stream

Last event time

dog-pytorch-2023-05-16-06-16-14-289/algo-1-1684217856

2023-05-16 13:18:58 (UTC+07:00)

dog-pytorch-2023-05-16-06-16-14-289/algo-3-1684217857

2023-05-16 13:18:58 (UTC+07:00)

dog-pytorch-2023-05-16-06-16-14-289/algo-4-1684217857

2023-05-16 13:18:55 (UTC+07:00)

dog-pytorch-2023-05-16-06-16-14-289/algo-2-1684217857

2023-05-16 13:18:55 (UTC+07:00)

- Model deployment

Amazon SageMaker > Endpoints

Endpoints

Update endpoint

Actions

Create endpoint

Search endpoints

1

	Name	ARN	Creation time	Status	Last updated
	pytorch-inference-2023-05-16-06-47-36-854	arn:aws:sagemaker:us-east-1:418228298154:endpoint/pytorch-inference-2023-05-16-06-47-36-854	5/16/2023, 1:47:37 PM	InService	5/16/2023, 1:50:12 PM

Step 2: EC2 Training

- Model training can also be carried out on an EC2 instance. For this purpose, I opted for an Ubuntu 20.04 that already had the necessary libraries installed. I chose the `g4dn.xlarge` instance, as it comes equipped with the latest version of PyTorch.

```
ubuntu@ip-172-31-20-77:~$ python3 solution.py
/home/ubuntu/.local/lib/python3.8/site-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the future, please use 'weights' instead.
  warnings.warn(msg)
/home/ubuntu/.local/lib/python3.8/site-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or `None` for 'weights' are deprecated since 0.13 and may be removed in the future. 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 /home/ubuntu/.cache/torch/hub/checkpoints/resnet50-0676ba61.pth
100% |██████████████████████████████████████████████████████████████████████████████| 97.8M/97.8M [00:00<00:00, 370MB/s]

Starting Model Training
saved
ubuntu@ip-172-31-20-77:~$ ls
TrainedModels  dogImages  dogImages.zip  requirements.txt  solution.py
ubuntu@ip-172-31-20-77:~$
```

- The image above illustrates an EC2 instance executing the `ec2train1.py` script to train the model.
- The code modifications in `ec2train1.py` bear a significant resemblance to the code in `train_and_deploy-solution.ipynb`. However, there are a few distinctions due to some modules that are exclusive to SageMaker. A large portion of the EC2 training code has been modified from the functions outlined in the `hpo.py` starter script. While `ec2train.py` trains the model with specific arguments, `hpo.py` retrieves arguments for the model through command line parsing. This latter script is capable of training multiple models with varying hyperparameters.

Step 3: Lambda function setup

- Once your model has been trained and deployed, the subsequent crucial step is to establish a Lambda function. This function facilitates the access to your model and its predictions by APIs and other applications, thereby serving as a vital component of production deployment.

The screenshot shows a code editor with a file named `lambda_function.py`. The code is a Python lambda function that interacts with the Sagemaker runtime. It imports `base64`, `logging`, `json`, and `boto3`. It sets up a logger and prints a message. The `lambda_handler` function takes an event and context, decodes the event content, and then uses `boto3` to invoke a specific Sagemaker endpoint. The endpoint name is `pytorch-inference-2023-05-16-06-47-36-854`. The function returns the response body as JSON.

```

1 |
2 | import base64
3 | import logging
4 | import json
5 | import boto3
6 | #import numpy
7 | logger = logging.getLogger(__name__)
8 | logger.setLevel(logging.DEBUG)
9 |
10 | print('Loading Lambda function')
11 |
12 | runtime=boto3.Session().client('sagemaker-runtime')
13 | endpoint_Name='pytorch-inference-2023-05-16-06-47-36-854'
14 |
15 | def lambda_handler(event, context):
16 |
17 |     #x=event['content']
18 |     #aa=x.encode('ascii')
19 |     #bs=base64.b64decode(aa)
20 |     print('Context:::', context)
21 |     print('EventType:::', type(event))
22 |     bs=event
23 |     runtime=boto3.Session().client('sagemaker-runtime')
24 |
25 |     response=runtime.invoke_endpoint(EndpointName=endpoint_Name,
26 |                                     ContentType="application/json",
27 |                                     Accept='application/json',
28 |                                     #Body=bytearray(x)
29 |                                     Body=json.dumps(bs))
30 |
31 |     result=response['Body'].read().decode('utf-8')
32 |     sss=json.loads(result)
33 |

```

Step 4: Lambda policy and testing

- Creating policy with permission to only invoke specific endpoint.

[IAM](#) > [Policies](#) > [invoke-endpoint](#) > [Edit policy](#)

Step 1

Modify permissions

Modify permissions in invoke-endpoint [Info](#)

Change or add permissions by choosing services, actions, and conditions. Build permission statements using the JSON editor.

Step 2

Review and save

Policy editor

Visual

JSON

Actions ▼

```

1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "VisualEditor0",
6       "Effect": "Allow",
7       "Action": "sagemaker:InvokeEndpoint",
8       "Resource": "arn:aws:sagemaker:*:418228298154:endpoint/pytorch-
9         inference-2023-05-16-06-47-36-854"
10    }
11  ]

```

Edit statement

Select a statement

Select an existing statement in the policy or add a new statement.

[+ Add new statement](#)

role-sagemaker

Delete

Allows Lambda functions to call AWS services on your behalf.

Summary

Edit

Creation date	ARN
May 16, 2023, 16:06 (UTC+07:00)	arn:aws:iam::418228298154:role/role-sagemaker
Last activity	Maximum session duration
None	1 hour

- Permissions
- Trust relationships
- Tags
- Access Advisor
- Revoke sessions

Permissions policies (2) [Info](#)

↻

Simulate


Remove

Add permissions ▾

🔍

Filter policies by property or policy name and press enter.

< 1 > ⚙️

<input type="checkbox"/>	Policy name ↗	Type	Description
<input type="checkbox"/>	+ invoke-endpoint	Customer managed	
<input type="checkbox"/>	+  AdministratorAccess	AWS managed - job function	Provides full access to AWS :

- Testing lambda function

The test event test was successfully saved.

Code source [Info](#)

Upload from ▾

File Edit Find View Go Tools Window

Test ▾ Deploy

⌕ Go to Anything (% P)

lambda_function.x Execution result: x +

Environment

lambda1 - /

lambda_function.py

Execution results

Status: Succeeded Max memory used: 77 MB Time: 1537.56 ms

Test Event Name

test

Response

{
 "statusCode": 200,
 "headers": {
 "Content-Type": "text/plain",
 "Access-Control-Allow-Origin": "*" }
 },
 "type-result": "<class 'str'>,"
 "Content-Type-In": "<__main__.LambdaContext object at 0x7f92b67a6af0>,"
 "body": "[[-3.8026819229125977, -2.0859713554382324, -1.539570927619934, -1.3080164194107056, -3.339965343475342, -4.404972553253174, -1.020"

Function Logs

START RequestId: 1ca242b0-0a85-418d-89c0-89f3bc51caa8 Version: \$LATEST
Context::: <__main__.LambdaContext object at 0x7f92b67a6af0>
Event type:: <class 'dict'>
END RequestId: 1ca242b0-0a85-418d-89c0-89f3bc51caa8
REPORT RequestId: 1ca242b0-0a85-418d-89c0-89f3bc51caa8 Duration: 1537.56 ms Billed Duration: 1538 ms Memory Size: 128 MB Max Memory Used:

Request ID

1ca242b0-0a85-418d-89c0-89f3bc51caa8

- Response:

Response

```
{  
  "statusCode": 200,  
  "headers": {  
    "Content-Type": "text/plain",  
    "Access-Control-Allow-Origin": "*" }  
  },
```

```

"type-result": "<class 'str'>",
"Content-Type-In": "<__main__.LambdaContext object at 0x7f92b67a6af0>",
"body": "[[-3.8026819229125977, -2.0859713554382324, -1.539570927619934,
-1.3080164194107056, -3.339965343475342, -4.404972553253174,
-1.0209825038909912, -1.900415062904358, -2.956531286239624,
-0.31846800446510315, -1.2855305671691895, -4.488164901733398,
-1.0597424507141113, 1.0620988607406616, -3.998521566390991,
-3.0901365280151367, -4.347545623779297, -2.194549560546875,
-2.292271375656128, 1.4613531827926636, -2.8937950134277344,
-1.1955442428588867, -4.603621482849121, -2.92773699760437,
-2.6350646018981934, -3.281313180923462, -2.2812507152557373,
-3.261805534362793, -2.8130531311035156, -1.1878055334091187,
-0.9282594323158264, -3.1857945919036865, -4.730673789978027,
-1.5436851978302002, -4.537162780761719, -2.8843417167663574,
-3.3408946990966797, -2.4748477935791016, -1.0711184740066528,
-2.507316827774048, -2.1006064414978027, -2.077411413192749,
1.8813728094100952, -1.3100594282150269, -2.0014002323150635,
-5.724288463592529, -1.0919257402420044, -0.9209285378456116,
-1.01749849319458, -0.3974989354610443, -3.730351686477661,
-5.149448394775391, -4.111176490783691, -1.7322322130203247,
-3.022367238998413, -1.5153255462646484, -2.9911558628082275,
-3.6611413955688477, -1.9944188594818115, -2.1147618293762207,
-4.525941371917725, -2.997464895248413, -4.339841842651367,
-4.683041572570801, -2.261636734008789, -3.0569100379943848,
-0.9021389484405518, -4.452295303344727, -2.18411922454834,
-0.2539750933647156, 0.6183738112449646, -2.6141915321350098,
-3.3500468730926514, -3.0878090858459473, -3.2058985233306885,
-1.1343743801116943, -5.269481182098389, -1.3220785856246948,
-3.5936434268951416, -2.4593327045440674, -0.47329947352409363,
-3.876897096633911, 0.4124978184700012, -2.1235811710357666,
-5.330963611602783, -4.047287940979004, -1.3646618127822876,
-2.864323854446411, -1.7422258853912354, -1.4358495473861694,
-5.686724662780762, -4.1877665519714355, -2.343993663787842,
-3.5264501571655273, -2.2768118381500244, -0.4151455760002136,
-0.6376075148582458, -1.7722325325012207, -3.9547088146209717,
-2.759162425994873, -3.774449586868286, -1.6567118167877197,
-1.488266110420227, -3.67352294921875, -3.1619045734405518,
-3.5365829467773438, -2.6133177280426025, -1.365824580192566,
-3.008732795715332, -0.39713871479034424, -0.5222817063331604,
-0.2639937400817871, -4.13600492477417, -1.5596508979797363,
-3.58133864402771, -2.406162977218628, -6.019803524017334,
-0.6986405253410339, -3.070871591567993, -0.1717919111251831,
0.023802954703569412, -3.0725650787353516, -3.4439449310302734,
-2.497530221939087, -3.286989450454712, -2.190408229827881,
-1.2030950784683228, 0.4249667227268219, -3.0472755432128906,
-4.850414752960205, -1.4908115863800049, -1.322715163230896,
-2.5584988594055176]]]"
}

```

Step 5: Lambda concurrency setup and endpoint auto-scaling

- Concurrency

reserved instances: 5/1000
provisioned instances: 3/5

+ Add trigger

+ Add destination

24 minutes ago

Function ARN
arn:aws:lambda:us-east-1:418228298154:function:lamb
da1

Function URL [Info](#)
-

Code | Test | Monitor | Configuration | Aliases | Versions

General configuration

Triggers

Permissions

Destinations

Function URL

Environment variables

Tags

VPC

Monitoring and operations tools

Concurrency

Concurrency

Edit

Function concurrency

Reserved concurrency

Use reserved concurrency

5

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](#)

↻

Edit

Remove

Add

Find configuration

	Qualifier	Type	Provisioned concurrency	Status	Details
<input type="radio"/>	1	version	3	Ready	-

- Auto-scaling

Sagemaker endpoints require automatic scaling to respond to high traffic.

minimum instances: 1
maximum instances: 3
target value: 3 // number of simulataneous requests which will trigger scaling
scale-in time: 300 s
scale-out time: 300 s

localhost:6419

7/8

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"/>	<input checked="" type="radio"/> P	AllTraffic	1	1	-	ml.m5.large	1	1	1 - 3	Yes

Endpoint configuration settings

Change

Clone

Endpoint configuration

Name	ARN	Encryption key	Creation time
pytorch-inference-2023-05-16-06-47-36-854	arn:aws:sagemaker:us-east-1:418228298154:endpoint-config/pytorch-inference-2023-05-16-06-47-36-854	-	5/16/2023, 1:47:37 PM