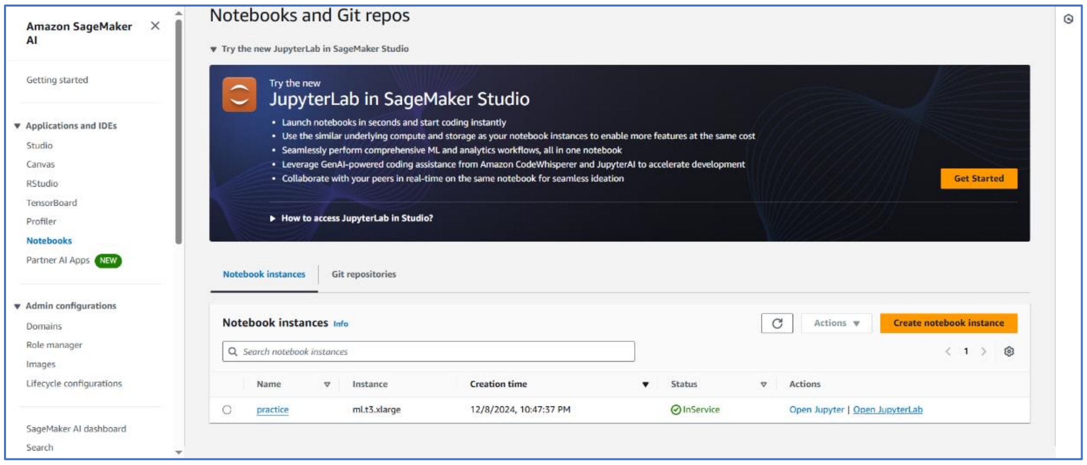
Operationalizing an AWS ML Project

Dog Image Classification

1. Training and deployment

- **Setup a Sagemaker instance**

  I chose the **ml.t3.xlarge** instance for running the notebook because it offers a good balance of cost and performance for development tasks. This instance provides sufficient CPU and memory resources to handle data preprocessing, model training, and basic hyperparameter tuning in SageMaker without incurring high costs, making it suitable for this stage of the project.



Notebooks and Git repos

Try the new JupyterLab in SageMaker Studio

Try the new
JupyterLab in SageMaker Studio

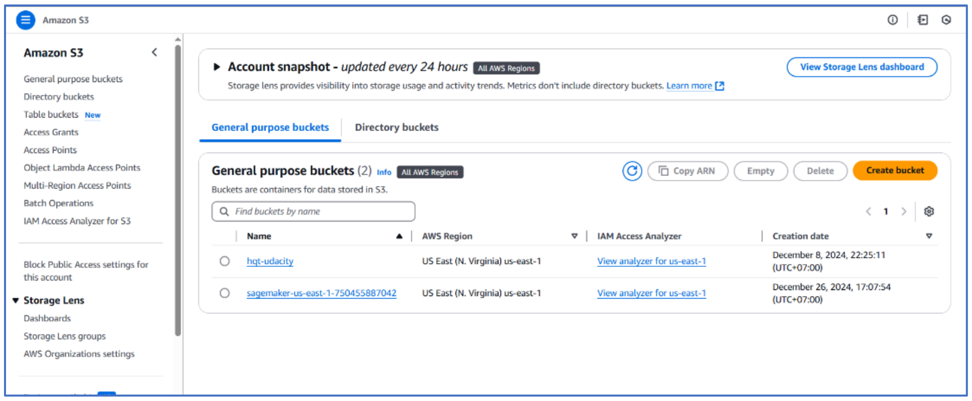
- Launch notebooks in seconds and start coding instantly
- Use the similar underlying compute and storage as your notebook instances to enable more features at the same cost
- Seamlessly perform comprehensive ML and analytics workflows, all in one notebook
- Leverage GenAI-powered coding assistance from Amazon CodeWhisperer and JupyterAI to accelerate development
- Collaborate with your peers in real-time on the same notebook for seamless ideation

How to access JupyterLab in Studio?

Notebook instances Info

Name	Instance	Creation time	Status	Actions
practice	ml.t3.xlarge	12/8/2024, 10:47:37 PM	InService	Open Jupyter | Open JupyterLab

- **Setup data storage on S3 (hqt-udacity bucket)**



Account snapshot - *updated every 24 hours* All AWS Regions

Storage lens provides visibility into storage usage and activity trends. Metrics don't include directory buckets. Learn more

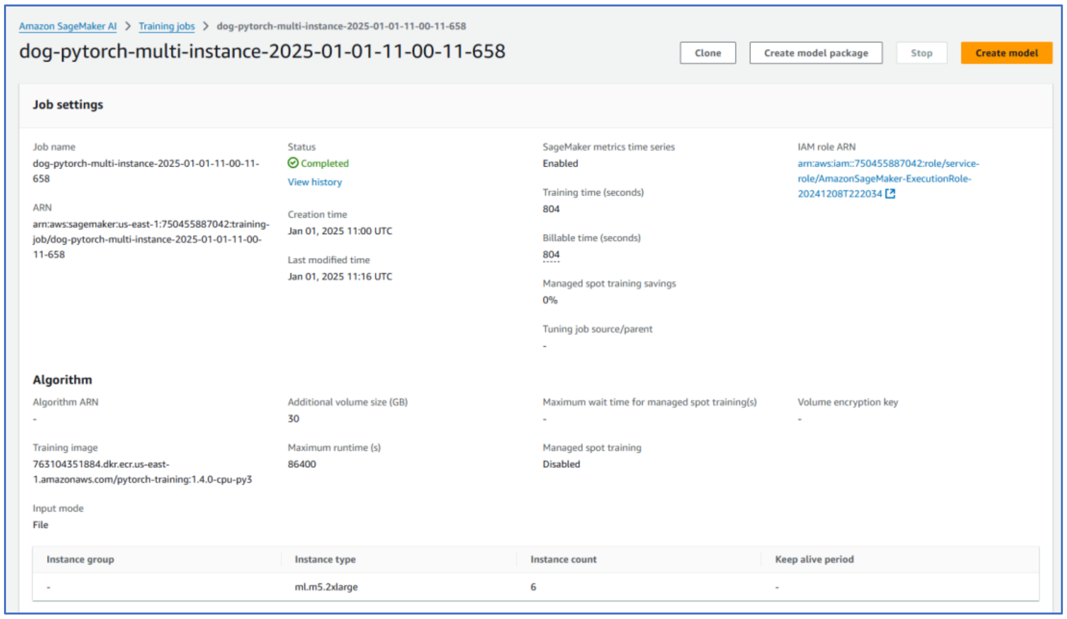
General purpose buckets (2) Info All AWS Regions

Buckets are containers for data stored in S3.

Name	AWS Region	IAM Access Analyzer	Creation date
hqt-udacity	US East (N. Virginia) us-east-1	View analyzer for us-east-1	December 8, 2024, 22:25:11 (UTC+07:00)
sagemaker-us-east-1-750455887042	US East (N. Virginia) us-east-1	View analyzer for us-east-1	December 26, 2024, 17:07:54 (UTC+07:00)

- **Multi-instance training**

  The training job uses 6 **ml.m5.2xlarge** instances with SageMaker's PyTorch estimator for distributed training. This instance type was chosen for its cost-effectiveness and balanced CPU and memory resources, suitable for workloads not requiring GPUs. SageMaker Debugger and Profiler were enabled to monitor and optimize the training process. This setup ensures scalability and reduces training time efficiently.


Amazon SageMaker AI > Training jobs > dog-pytorch-multi-instance-2025-01-01-11-00-11-658
Clone
Create model package
Stop
Create model

dog-pytorch-multi-instance-2025-01-01-11-00-11-658

Job settings

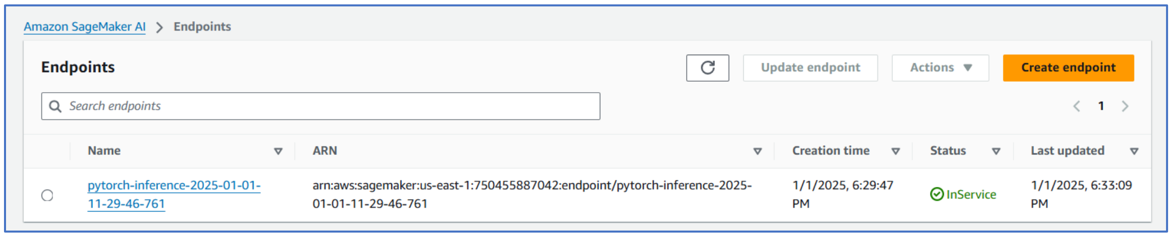
Job name	dog-pytorch-multi-instance-2025-01-01-11-00-11-658
ARN	arn:aws:sagemaker:us-east-1:750455887042:training-job/dog-pytorch-multi-instance-2025-01-01-11-00-11-658
Status	Completed (View history)
Creation time	Jan 01, 2025 11:00 UTC
Last modified time	Jan 01, 2025 11:16 UTC
SageMaker metrics time series	Enabled
Training time (seconds)	804
Billable time (seconds)	804
Managed spot training savings	0%
Tuning job source/parent	-
IAM role ARN	arn:aws:iam::750455887042:role/service-role/AmazonSageMaker-ExecutionRole-20241208T222034

Algorithm

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

Instance group	Instance type	Instance count	Keep alive period
-	ml.m5.2xlarge	6	-

- **Deploy endpoint**

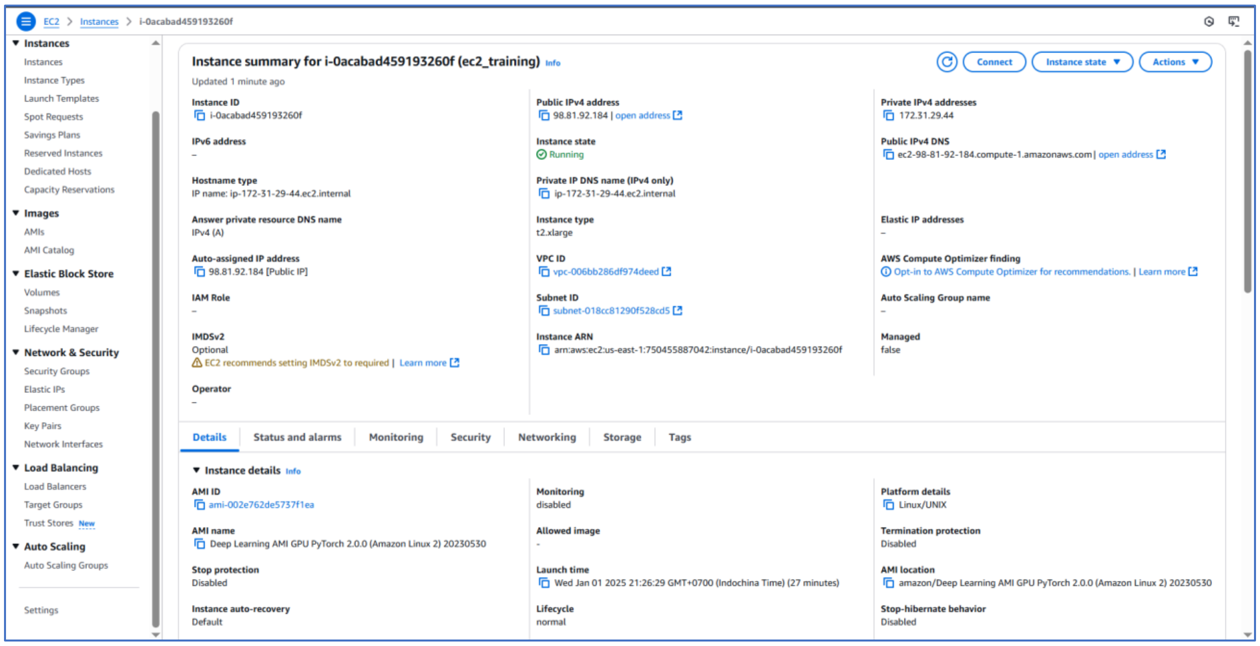


Name	ARN	Creation time	Status	Last updated
pytorch-inference-2025-01-01-11-29-46-761	arn:aws:sagemaker:us-east-1:750455887042:endpoint/pytorch-inference-2025-01-01-11-29-46-761	1/1/2025, 6:29:47 PM	InService	1/1/2025, 6:33:09 PM

2. EC2 Training

- **Create and open an EC2 instance**

  I used **Deep Learning AMI GPU PyTorch 2.0.0 (Amazon Linux 2)** which has built-in Pytorch, Numpy libraries. I setup **t2.xlarge** EC2 instance because it provides a cost-effective solution with 4 vCPUs and 16 GB of memory, sufficient for running training scripts and handling moderate computational tasks. It is ideal for development and testing phases where GPU acceleration is not critical, ensuring a balance between performance and cost efficiency.



Instance summary for i-0acabad459193260f (ec2_training) Info

Updated 1 minute ago

Instance ID
i-0acabad459193260f

Public IPv4 address
98.81.92.184 | open address

Private IPv4 addresses
172.31.29.44

IPv6 address
–

Instance state
Running

Public IPv4 DNS
ec2-98-81-92-184.compute-1.amazonaws.com | open address

Hostname type
IP name: ip-172-31-29-44.ec2.internal

Private IP DNS name (IPv4 only)
ip-172-31-29-44.ec2.internal

Answer private resource DNS name
IPv4 (A)

Instance type
t2.xlarge

Elastic IP addresses
–

Auto-assigned IP address
98.81.92.184 [Public IP]

VPC ID
vpc-006bb286df974deed

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

IAM Role
–

Subnet ID
subnet-018cc81290f528cd5

Auto Scaling Group name
–

IMDSv2
Optional
EC2 recommends setting IMDSv2 to required | Learn more

Instance ARN
arn:aws:ec2:us-east-1:750455887042:instance/i-0acabad459193260f

Managed
false

Operator
–

Instance details Info

AMI ID
ami-002e762de5737f1ea

Monitoring
disabled

Platform details
Linux/UNIX

AMI name
Deep Learning AMI GPU PyTorch 2.0.0 (Amazon Linux 2) 20230530

Allowed image
–

Termination protection
Disabled

Stop protection
Disabled

Launch time
Wed Jan 01 2025 21:26:29 GMT+0700 (Indochina Time) (27 minutes)

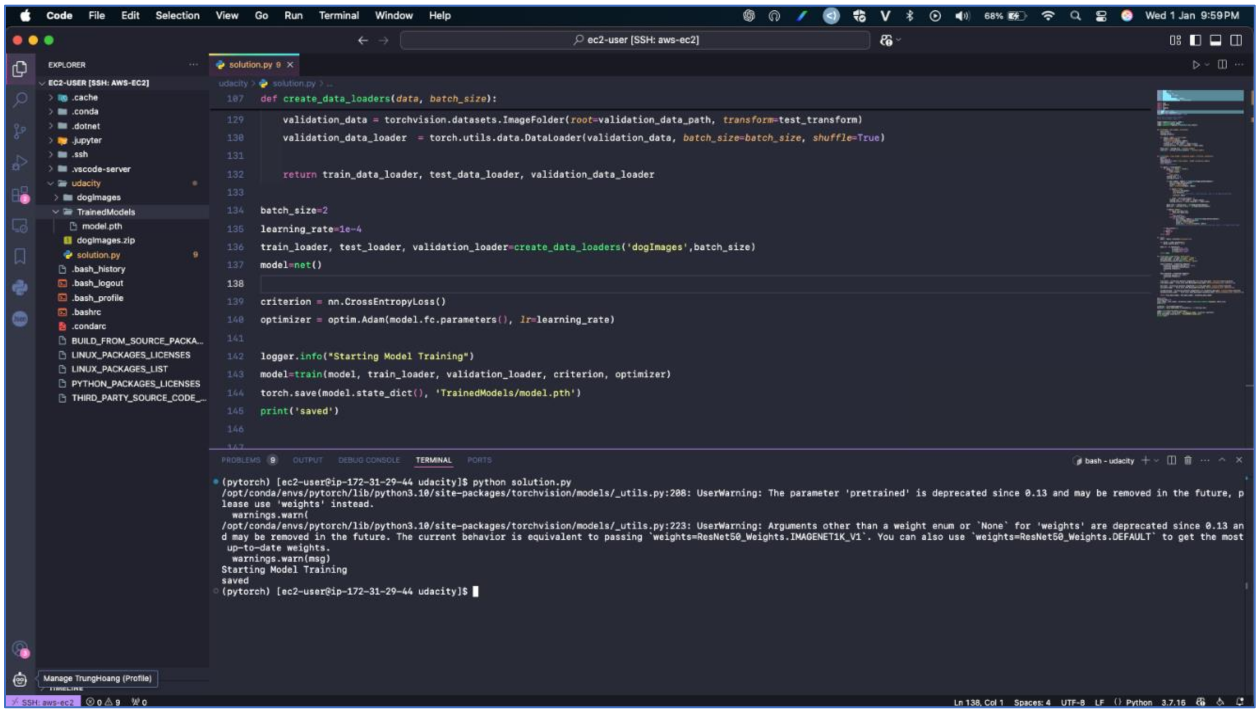
AMI location
amazon/Deep Learning AMI GPU PyTorch 2.0.0 (Amazon Linux 2) 20230530

Instance auto-recovery
Default

Lifecycle
normal

Stop-hibernate behavior
Disabled

Training model on EC2, here is the terminal run train on vscode remote ssh to EC2.



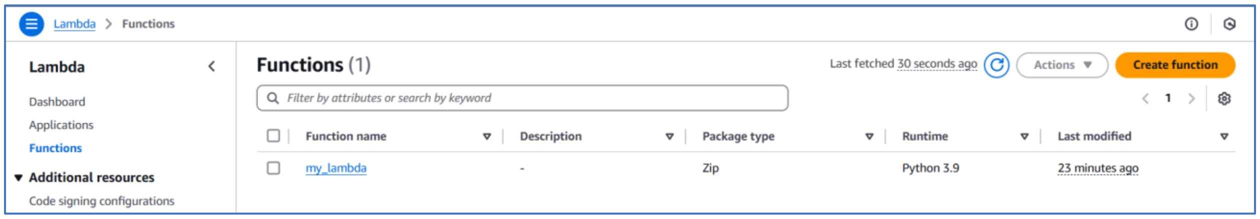
```python
def create_data_loaders(data, batch_size):
    validation_data = torchvision.datasets.ImageFolder(root=validation_data_path, transform=test_transform)
    validation_data_loader  = torch.utils.data.DataLoader(validation_data, batch_size=batch_size, shuffle=True)

    return train_data_loader, test_data_loader, validation_data_loader

batch_size=2
learning_rate=1e-4
train_loader, test_loader, validation_loader=create_data_loaders('dogImages',batch_size)
model=net()

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) [ec2-user@ip-172-31-29-44 udacity]$ python solution.py
/opt/conda/envs/pytorch/lib/python3.10/site-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the future, p
lease use 'weights' instead.
  warnings.warn(
/opt/conda/envs/pytorch/lib/python3.10/site-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or `None` for 'weights' are deprecated since 0.13 an
d 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)
Starting Model Training
saved
(pytorch) [ec2-user@ip-172-31-29-44 udacity]$
```

- **Comparison of EC2 instances (ec2train1.py) with SageMaker Notebook Instances (train_and_deploy-solution.ipynb and hpo.py)**
  - ec2train1.py is suitable for simple, small-scale training setups, while the notebook and hpo.py are designed for production-grade pipelines on AWS.
  - SageMaker-based training supports distributed learning, making it ideal for large datasets or high-performance requirements.
  - The notebook setup integrates seamlessly with AWS tools, whereas ec2train1.py requires manual configuration.
  - EC2 instances are highly customizable and scalable, allowing users to tailor the setup to their specific needs, such as the choice of machine learning frameworks (e.g., PyTorch or TensorFlow), CPU and memory specifications, or GPU support. They are particularly suitable for high-performance computing tasks, enabling faster training for large-scale machine learning models. Additionally, EC2 instances can be optimized to match specific workloads, providing flexibility and control over computational resources.
  - SageMaker notebook instances provide a pre-configured environment with popular machine learning frameworks and libraries, making setup quick and user-friendly. These instances are well-integrated with other AWS services, such as SageMaker for training and deploying models, and offer collaboration tools for seamless teamwork. This integration with the AWS ecosystem simplifies the workflow, making it ideal for machine learning engineering and operations tasks.
  - In summary, EC2 instances are better suited for customized, resource-intensive training needs, while SageMaker notebook instances are designed for rapid setup and efficient integration with AWS machine learning tools.

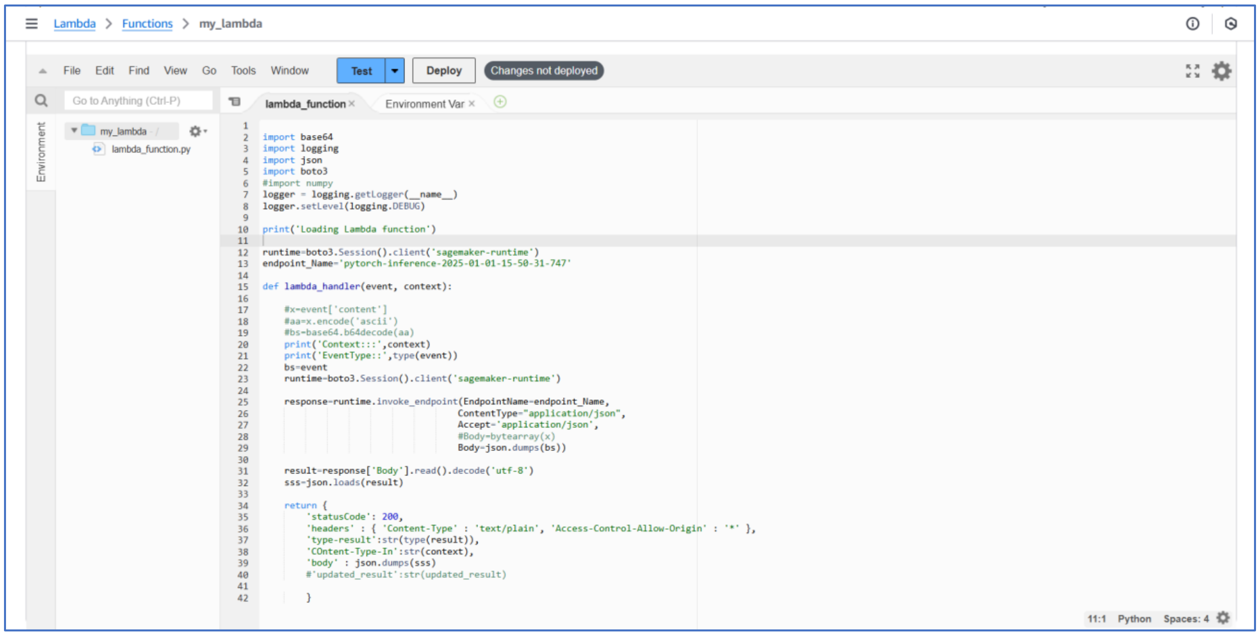
3. Lambda function setup

- Lambda function invokes the student's deployed endpoint



Functions (1)

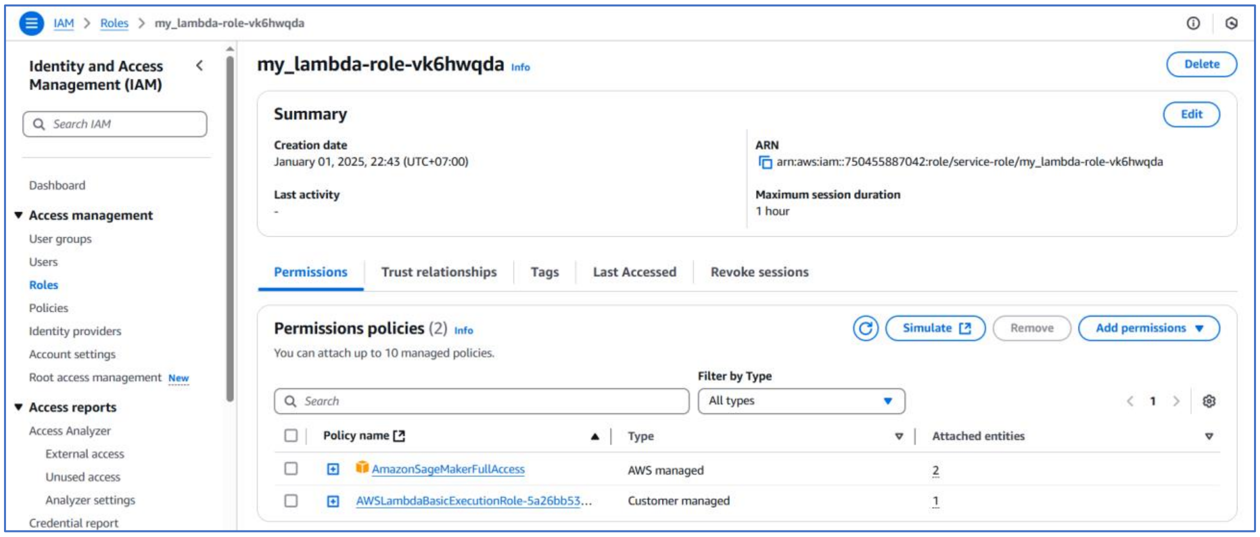
Function name	Description	Package type	Runtime	Last modified
my_lambda	-	Zip	Python 3.9	23 minutes ago


Lambda > Functions > my_lambda
File Edit Find View Go Tools Window Test Deploy Changes not deployed
Go to Anything (Ctrl-P)
lambda_function × Environment Var ×
Environment
my_lambda
lambda_function.py
11:1 Python Spaces: 4

```python

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-2025-01-01-15-50-31-747'

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)

        }
```

4. Security and testing

- **Attach a security policy to the role associated with the Lambda endpoint**



my_lambda-role-vk6hwqda Info

Summary

Creation date
January 01, 2025, 22:43 (UTC+07:00)

Last activity
-

ARN
arn:aws:iam::750455887042:role/service-role/my_lambda-role-vk6hwqda

Maximum session duration
1 hour

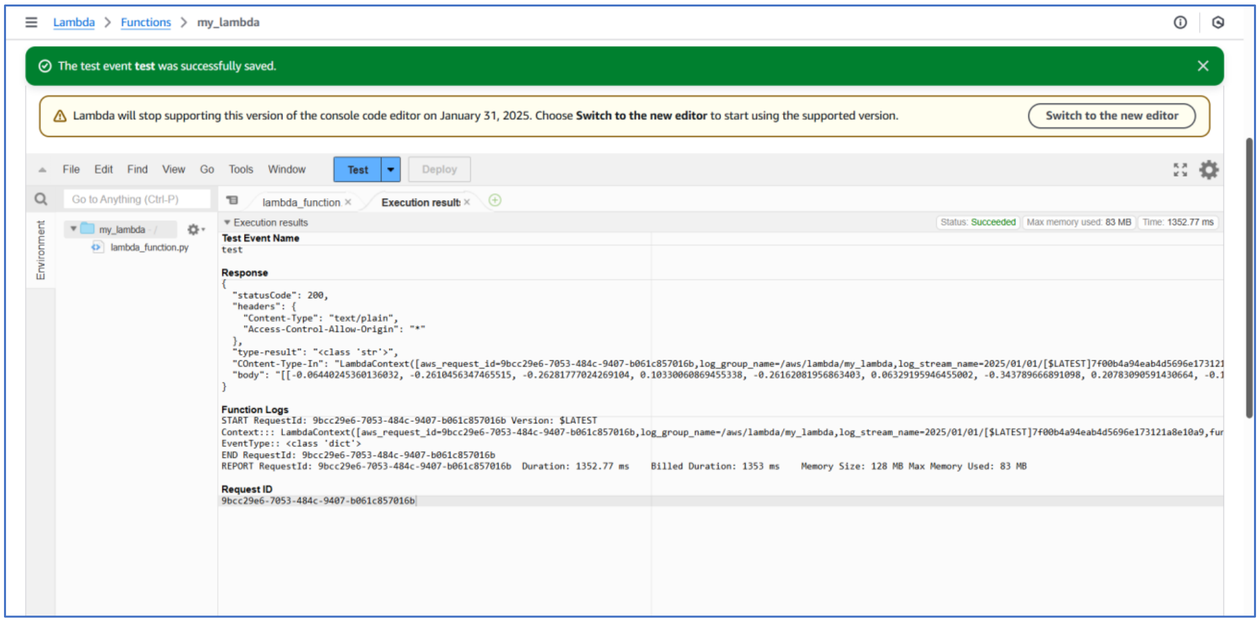
Permissions policies (2) Info

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
AmazonSageMakerFullAccess	AWS managed	2
AWSLambdaBasicExecutionRole-5a26bb53...	Customer managed	1

- **Test the Lambda function.**



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

Lambda will stop supporting this version of the console code editor on January 31, 2025. Choose **Switch to the new editor** to start using the supported version.

Status: Succeeded | Max memory used: 83 MB | Time: 1352.77 ms

Test Event Name

```
test
```

Response

```
{
  "statusCode": 200,
  "headers": {
    "Content-Type": "text/plain",
    "Access-Control-Allow-Origin": "*"
  },
  "type-result": "<class 'str'>",
  "COntent-Type-In": "LambdaContext([aws_request_id=9bcc29e6-7053-484c-9407-b061c857016b,log_group_name=/aws/lambda/my_lambda,log_stream_name=2025/01/01/[$LATEST]7f00b4a94eab4d5696e173121
  "body": "[[-0.06440245360136032, -0.2610456347465515, -0.26281777024269104, 0.10330060869455338, -0.26162081956863403, 0.06329195946455002, -0.343789666891098, 0.20783090591430664, -0.1
}
```

Function Logs

```
START RequestId: 9bcc29e6-7053-484c-9407-b061c857016b Version: $LATEST
Context::: LambdaContext([aws_request_id=9bcc29e6-7053-484c-9407-b061c857016b,log_group_name=/aws/lambda/my_lambda,log_stream_name=2025/01/01/[$LATEST]7f00b4a94eab4d5696e173121a8e10a9,fur
EventType:: <class 'dict'>
END RequestId: 9bcc29e6-7053-484c-9407-b061c857016b
REPORT RequestId: 9bcc29e6-7053-484c-9407-b061c857016b  Duration: 1352.77 ms    Billed Duration: 1353 ms    Memory Size: 128 MB Max Memory Used: 83 MB
```

Request ID

```
9bcc29e6-7053-484c-9407-b061c857016b
```

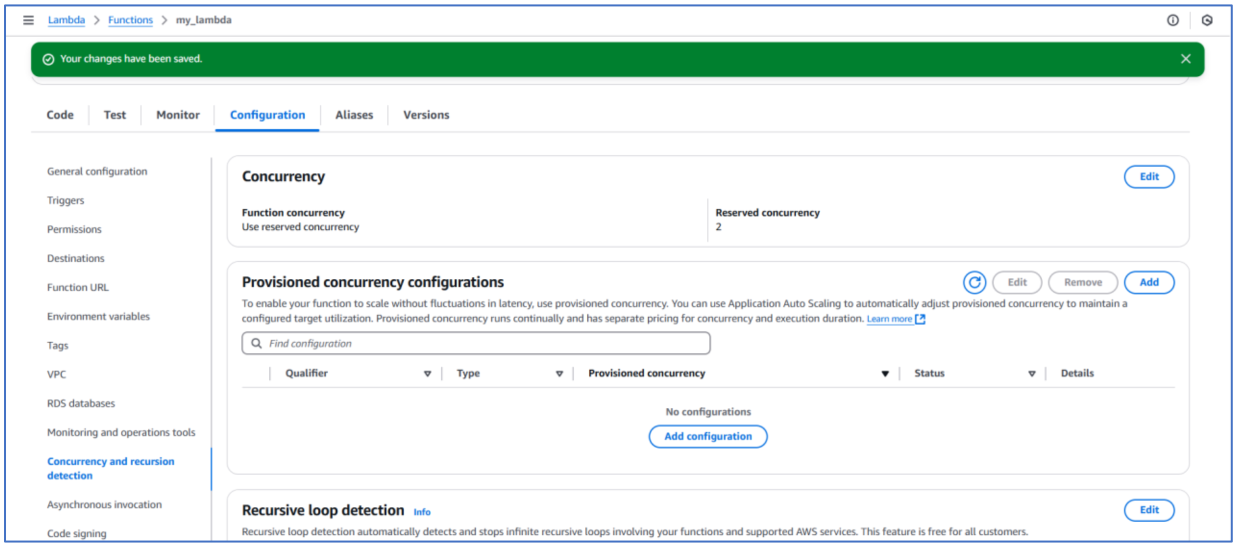
- **Identified Potential Vulnerabilities**

  - In the setup for the Lambda function, the IAM role assigned includes the **AmazonSageMakerFullAccess** policy to enable communication with the SageMaker inference endpoint. While this allows the function to execute the required tasks, it also introduces potential vulnerabilities that should be addressed for improved security.
  - The role attached to the Lambda function has AmazonSageMakerFullAccess, which provides broad permissions, including the ability to create, update, and delete SageMaker resources. If the Lambda function is compromised, an attacker could misuse these permissions to manipulate SageMaker resources, leading to data leaks, endpoint misuse, or increased costs.
  - The assigned role includes permissions for broader actions across all SageMaker resources. This violates the principle of least privilege, which states that roles should only have permissions necessary for their intended purpose.

5. Concurrency and auto-scaling

- **Lambda concurrency**

  In this project, the Lambda function was configured with Reserved Concurrency set to 2. By default, a Lambda function can handle only one request at a time. Configuring concurrency allows the function to handle multiple requests simultaneously by reserving the specified number of execution environments. The chosen concurrency level of 2 is appropriate for a low-demand service or a development/test environment. For production or high-traffic scenarios, this setting should be revisited and potentially increased or replaced with an auto-scaling approach to ensure efficient resource utilization and cost management.



Concurrency

Function concurrency: Use reserved concurrency

Reserved concurrency: 2

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

Qualifier	Type	Provisioned concurrency	Status	Details

No configurations

Recursive loop detection

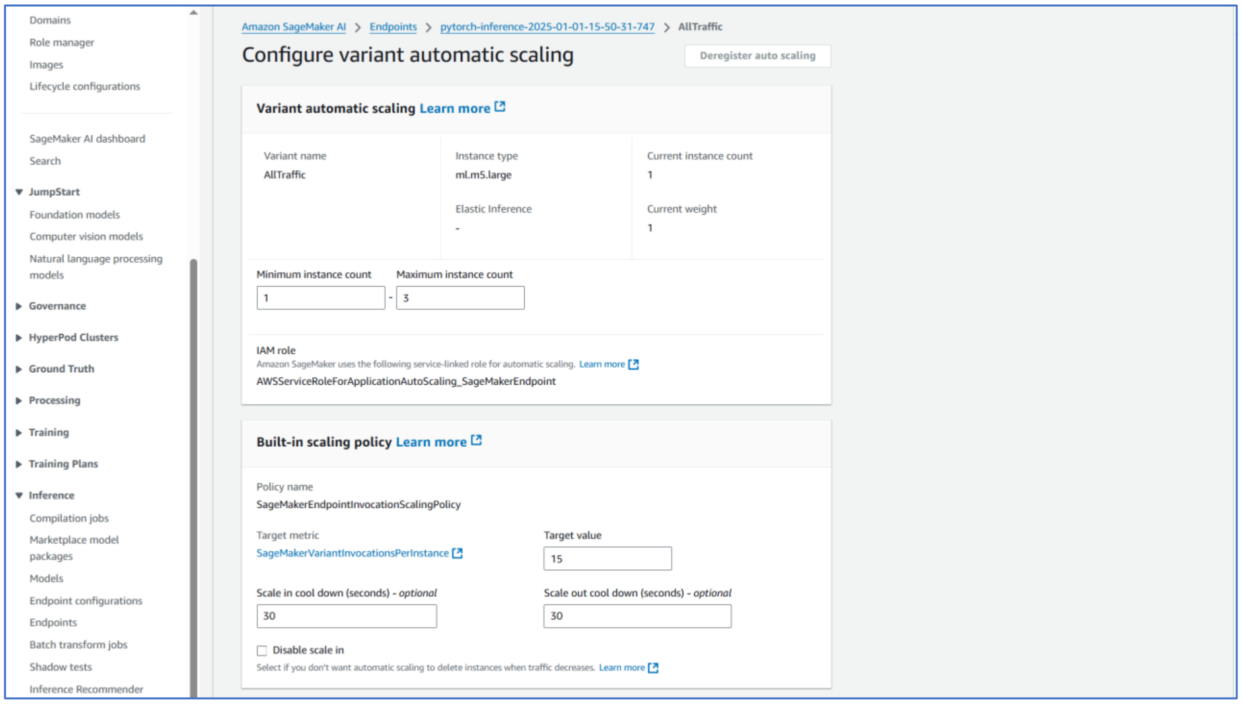
Recursive loop detection automatically detects and stops infinite recursive loops involving your functions and supported AWS services. This feature is free for all customers.

Considerations:

- Traffic: This configuration enables the Lambda function to handle up to two simultaneous requests. While suitable for low-traffic scenarios, it may not be sufficient for high-demand services.
- Cost: Provisioned concurrency incurs additional costs, as it pre-initializes environments to reduce cold start latency. By keeping the reserved concurrency value low (2), the additional cost is minimized.
- Efficiency: Setting concurrency to 2 balances reduced latency for simultaneous requests while avoiding unnecessary expenses. However, for scalable applications with fluctuating traffic, higher concurrency or dynamic scaling might be required.

- **Auto-Scaling Sagemaker Endpoint**

To ensure the endpoint can efficiently handle fluctuating traffic, auto-scaling was configured for the SageMaker endpoint. This allows the system to dynamically adjust resources based on the number of incoming requests.



Configure variant automatic scaling

Variant automatic scaling Learn more

Variant name: AllTraffic
Instance type: ml.m5.large
Current instance count: 1
Elastic Inference: -
Current weight: 1

Minimum instance count: 1 - Maximum instance count: 3

IAM role
Amazon SageMaker uses the following service-linked role for automatic scaling. Learn more
AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint

Built-in scaling policy Learn more

Policy name: SageMakerEndpointInvocationScalingPolicy
Target metric: SageMakerVariantInvocationsPerInstance
Target value: 15
Scale in cool down (seconds) - *optional*: 30
Scale out cool down (seconds) - *optional*: 30

☐ Disable scale in
Select if you don't want automatic scaling to delete instances when traffic decreases. Learn more

Considerations:

- Traffic: This setup allows the endpoint to efficiently handle up to 15 simultaneous requests per instance, with a maximum of 3 instances ensuring scalability for moderate traffic.
- Cost: Keeping the minimum instance count at 1 reduces idle resource costs, while the maximum of 3 ensures that scaling is limited to control expenses during peak demand.
- Efficiency: By setting reasonable delays for scaling actions, the configuration minimizes rapid instance changes, improving stability and ensuring resources are used effectively.