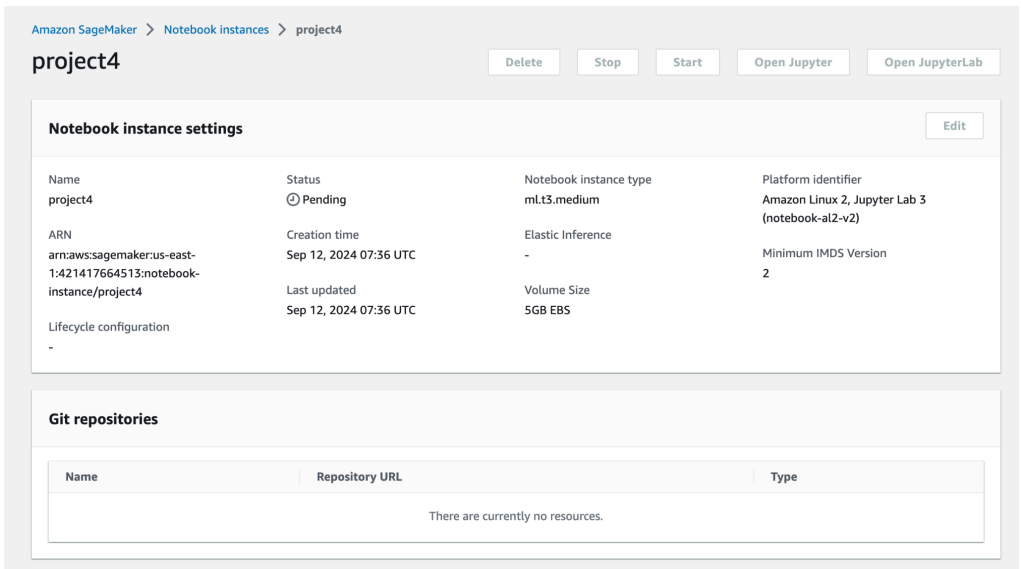


# Operationalizing ML on Sagemaker

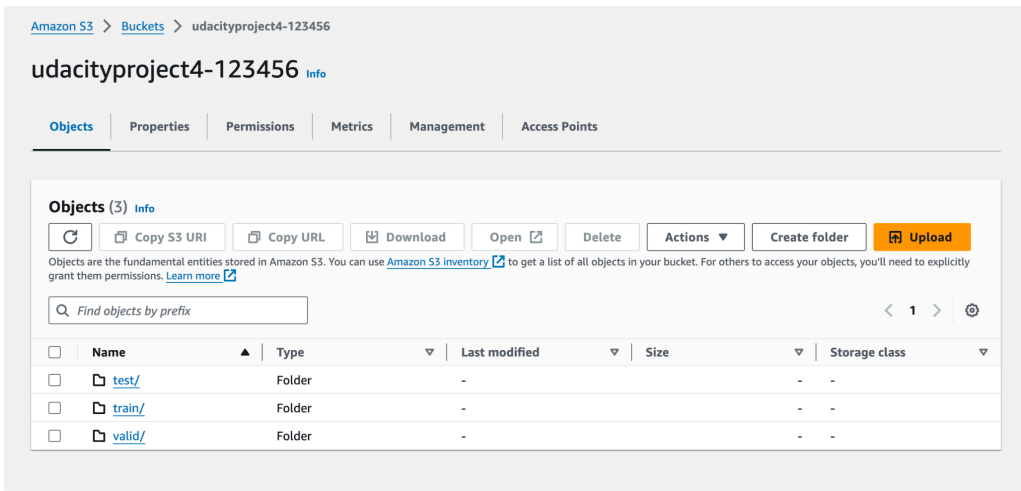
## Training and Deployment on Sagemaker

### Initial Setup

I chose the `ml.t3.medium` instance because it offers a good balance between cost and computing power for small to medium-sized training tasks. It provides sufficient CPU capacity, memory, and network performance to efficiently handle the training and deployment of a computer vision model without incurring excessive costs. The instance also launches quickly, making it an ideal choice for tasks that don't require intensive GPU resources but still need reasonable computational speed.



### S3 Setup



# Training and Deployment

## Hyperparameter Tuning

Amazon SageMaker

>

Hyperparameter tuning jobs

Hyperparameter tuning jobs

Add/Edit tags

Create hyperparameter tuning job

Q Search hyperparameter tuning jobs

< 1 > ⚙

	Name	Status	Training completed/total	Creation time	Duration
<input type="radio"/>	pytorch-training-240912-0755	Completed	2 / 2	9/12/2024, 10:55:28 AM	an hour

Amazon SageMaker

>

Training jobs

Training jobs Info

Actions

Create training job

Q Search training jobs

< 1 > ⚙

	Name	Creation time	Duration	Job status	Warm pool status	Time left
<input type="radio"/>	pytorch-training-240912-0755-002-9166700a	9/12/2024, 11:20:21 AM	19 minutes	Completed	Available	-
<input type="radio"/>	pytorch-training-240912-0755-001-7a2072bf	9/12/2024, 10:55:33 AM	21 minutes	Completed	Reused	-

Describe the tuning results

[7]:

exp = HyperparameterTuningJobAnalytics(  
 hyperparameter\_tuning\_job\_name='pytorch-training-240912-0755')  
  
jobs = exp.dataframe()  
  
jobs.sort\_values('FinalObjectiveValue', ascending=0)

[7]:

	batch_size	learning_rate	TrainingJobName	TrainingJobStatus	FinalObjectiveValue	TrainingStartTime	TrainingEndTime	TrainingElapsedTimeS
0	"256"	0.013478	pytorch-training-240912-0755-002-9166700a	Completed	726.0	2024-09-12 08:20:26+00:00	2024-09-12 08:39:49+00:00	
1	"64"	0.047914	pytorch-training-240912-0755-001-7a2072bf	Completed	290.0	2024-09-12 07:56:13+00:00	2024-09-12 08:16:32+00:00	

Prepare to perform Training on Best Estimator

[8]:

best\_estimator=tuner.best\_estimator()

2024-09-12 08:20:24 Starting - Preparing the instances for training  
2024-09-12 08:20:24 Downloading - Downloading the training image  
2024-09-12 08:20:24 Training - Training image download completed. Training in progress.  
2024-09-12 08:20:24 Uploading - Uploading generated training model  
2024-09-12 08:20:24 Completed - Resource reused by training job: pytorch-training-240912-0755-002-9166700a

[9]:

best\_estimator.hyperparameters()

{'\_tuning\_objective\_metric': 'Test Loss',  
 'batch\_size': '64',  
 'learning\_rate': '0.047914138854644156',  
 'sagemaker\_container\_log\_level': '20',  
 'sagemaker\_estimator\_class\_name': 'PyTorch',  
 'sagemaker\_estimator\_module': 'sagemaker.pytorch.estimator',  
 'sagemaker\_job\_name': 'pytorch\_dog\_hpo-2024-09-12-07-55-27-690',  
 'sagemaker\_program': 'hpo.py',  
 'sagemaker\_region': 'us-east-1',  
 'sagemaker\_submit\_directory': 's3://sagemaker-us-east-1-421417664513/pytorch\_dog\_hpo-2024-09-12-07-55-27-690/source/sourcedir.tar.gz'}

# Training

Amazon SageMaker > Training jobs

Training jobs Info

Search training jobs

Actions

Create training job

	Name	Creation time	Duration	Job status	Warm pool status	Time left
<input type="radio"/>	dog-pytorch-2024-09-12-08-44-49-464	9/12/2024, 11:44:50 AM	20 minutes	Completed	-	-
<input type="radio"/>	pytorch-training-240912-0755-002-9166700a	9/12/2024, 11:20:21 AM	19 minutes	Completed	Terminated	-
<input type="radio"/>	pytorch-training-240912-0755-001-7a2072bf	9/12/2024, 10:55:33 AM	21 minutes	Completed	Reused	-

# Deployed Endpoint

First the model is created:

Amazon SageMaker > Models

Models

Search models

Create endpoint

Create endpoint configuration

Actions

Create model

	Name	ARN	Creation time
<input type="radio"/>	pytorch-inference-2024-09-12-09-46-05-681	arn:aws:sagemaker:us-east-1:421417664513:model/pytorch-inference-2024-09-12-09-46-05-681	9/12/2024, 12:46:06 PM

Then the endpoint is deployed:

Amazon SageMaker > Endpoints

Endpoints

Search endpoints

Update endpoint

Actions

Create endpoint

	Name	ARN	Creation time	Status	Last updated
<input type="radio"/>	pytorch-inference-2024-09-12-09-46-06-379	arn:aws:sagemaker:us-east-1:421417664513:endpoint/pytorch-inference-2024-09-12-09-46-06-379	9/12/2024, 12:46:07 PM	InService	9/12/2024, 12:49:23 PM

# Multi Instance Training

## Creating an Estimator - Multi-Instance Training,

```
1: ###in this cell, create and fit an estimator using multi-instance training  
#adjust this cell to accomplish multi-instance training  
estimator = PyTorch(  
    entry_point='hpo.py',  
    base_job_name='dog-pytorch',  
    role=role,  
    instance_count=2, # Change this to 2 or more for multi-instance training  
    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,  
)  
  
2: estimator.fit({"training": "s3://udacityproject4-123456/"}, wait=False)
```

Changed the instance\_count = 2 to do multi-instance training.

## New Endpoint

Amazon SageMaker > Endpoints

Endpoints						Update endpoint	Actions	Create endpoint
<input type="text" value="Search endpoints"/>						< 1 >		
	Name	ARN	Creation time	Status	Last updated			
<input type="radio"/>	pytorch-inference-2024-09-12-10-23-18-191	arn:aws:sagemaker:us-east-1:421417664513:endpoint/pytorch-inference-2024-09-12-10-23-18-191	9/12/2024, 1:23:18 PM	InService	9/12/2024, 1:26:45 PM			

# EC2 Training

## EC2 Setup

I selected “Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.3 (Amazon Linux 2)” on the AMI list, so I can proceed with the AMI compatible with the Pytorch environment, which we are using. t2.micro is selected as the instance type since we are training a small model and it is eligible for free tier.

Instances (1) Info Last updated less than a minute ago Connect Instance state Actions Launch instances

All states < 1 > ⚙️

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
<input type="checkbox"/>	UdacityProject...	i-0c6f93905f27a23e9	Running	t2.micro	Initializing	View alarms	us-east-1d	ec2-44-27

```
aws Services 🔍 Search [Option+S]
```

```
#  
_#_#####  
--\#####\  
--\#####|  
--\##/  
--V-'  
---\_____  
---/_/_____  
---m/'_____
```

Amazon Linux 2  
  
AL2 End of Life is 2025-06-30.  
  
A newer version of Amazon Linux is available!  
  
Amazon Linux 2023, GA and supported until 2028-03-15.  
<https://aws.amazon.com/linux/amazon-linux-2023/>

1 package(s) needed for security, out of 6 available  
Run "sudo yum update" to apply all updates.

=====

AMI Name: Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.3.0 (Amazon Linux 2)  
Supported EC2 instances: G4dn, G5, G6, Gr6, G6e, P4, P4de, p5  
\* To activate pre-built pytorch environment, run: 'source activate pytorch'  
NVIDIA driver version: 535.183.01  
CUDA versions available: cuda-12.1  
Default CUDA version is 12.1

Release notes: <https://docs.aws.amazon.com/dlami/latest/devguide/appendix-ami-release-notes.html>  
AWS Deep Learning AMI Homepage: <https://aws.amazon.com/machine-learning/amis/>  
Developer Guide and Release Notes: <https://docs.aws.amazon.com/dlami/latest/devguide/what-is-dlami.html>  
Support: <https://forums.aws.amazon.com/forum.jspa?forumID=263>  
For a fully managed experience, check out Amazon SageMaker at <https://aws.amazon.com/sagemaker>

=====

```
[root@ip-172-31-95-163 ~]# █
```

Created the file and pasted the code, and the training is done.

```

root@ip-172-31-35-163:~# python solution.py
/opt/conda/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, please use 'weights' instead.
  warnings.warn(
/opt/conda/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 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 /root/.cache/torch/hub/checkpoints/resnet50-0676ba61.pth
0% | 0.00/97.8M [00:00<?, 6.62M/97.8M [00:00<001, 69.3
7% | 19.9M/97.8M [00:00<000, 106
20% | 30.2M/97.8M [00:00<000, 107
31% | 40.5M/97.8M [00:00<000, 105
41% | 50.5M/97.8M [00:00<000, 89.2
52% | 59.4M/97.8M [00:00<000, 79.6
61% | 67.4M/97.8M [00:00<000, 73.6
69% | 74.6M/97.8M [00:00<000, 67.5
76% | 84.5M/97.8M [00:01<000, 76.6
96% | 93.9M/97.8M [00:01<000, 81.6
100% | 97.8M/97.8M [00:01<000, 85.1
MB/s)
Starting Model Training
saved
root@ip-172-31-35-163:~#

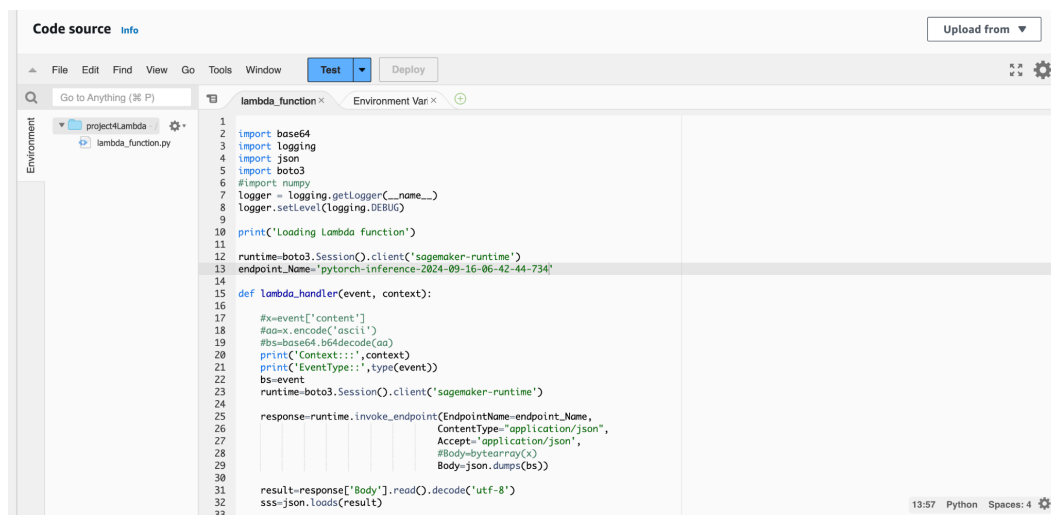
```

```
[root@ip-172-31-95-163 ~]# ls
dogImages  dogImages.zip  solution.py  TrainedModels
[root@ip-172-31-95-163 ~]# cd TrainedModels
[root@ip-172-31-95-163 TrainedModels]# ls
model.pth
[root@ip-172-31-95-163 TrainedModels]#
```

Firstly, we used a Python file on EC2 instead of a Jupyter Notebook, which needs a different environment to run. We also did not use the capabilities of Sagemaker such as hyperparameter tuning and instead used hard coded values. Other than these, the code is very similar to hpo.py, which we used as an entry point for our estimator when we were using the notebook.

# Lambda Function Setup

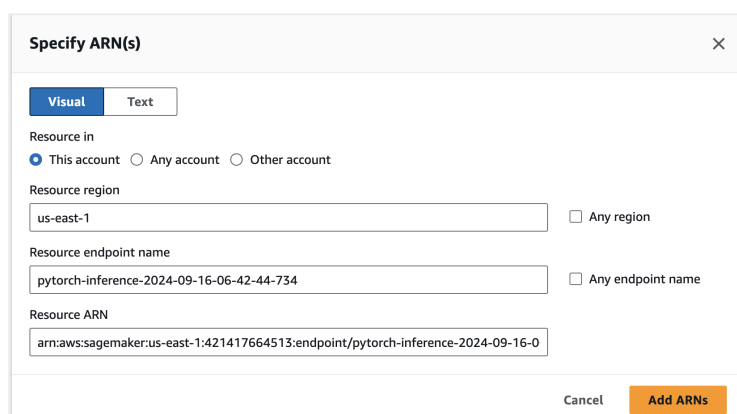
The `lambda_function.py` takes in an event and context, decodes the event using base64, and sends the resulting data as a request to a SageMaker endpoint. The function then handles the response from the endpoint, formatting it into a JSON object that includes HTTP status codes and headers. Additionally, it uses the Python logging module to record debug messages throughout the process.



```
1 import base64
2 import logging
3 import json
4 import boto3
5 import numpy
6
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-2024-09-16-06-42-44-734'
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)
```

# Lambda Function Security

We only added the policies we needed, Sagemaker execution, S3 access and Lambda execution as our permissions to ensure security and also to ensure that our user will not access other functions that we do not need at the moment. We need to be able to invoke the endpoint, so we need to attach the rule to ensure that. I created a rule to invoke the endpoint and attached this policy as well. Otherwise our test function does not work:



**Specify ARN(s)**

**Visual** Text





Resource in  
☒ This account ☐ Any account ☐ Other account

Resource region  
us-east-1 ☐ Any region

Resource endpoint name  
pytorch-inference-2024-09-16-06-42-44-734 ☐ Any endpoint name

Resource ARN  
arn:aws:sagemaker:us-east-1:421417664513:endpoint/pytorch-inference-2024-09-16-0

Cancel Add ARNs

Search		All types		< 1 >	⚙
<input type="checkbox"/>	Policy name	Type	Attached entities		
<input type="checkbox"/>	 <a href="#">AmazonS3FullAccess</a>	AWS managed	1		
<input type="checkbox"/>	 <a href="#">AmazonSageMaker-ExecutionPolicy-202...</a>	Customer managed	2		
<input type="checkbox"/>	 <a href="#">AWSLambdaBasicExecutionRole-1e070d...</a>	Customer managed	1		
<input type="checkbox"/>	 <a href="#">sagemaker-endpoint</a>	Customer managed	1		

## Lambda Function Testing

Used the json provided in the project page to test the lambda function:

Event name

ProjectTestEvent

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

☒ Private

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

☐ Shareable

This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

Template - optional

hello-world

Event JSON

Format JSON

1

<

>

}

2

"url": "https://s3.amazonaws.com/cdn-origin-etr.akc.org/wp-content/uploads/2017/11/2017-11-20-15-00-00.png"

3

}

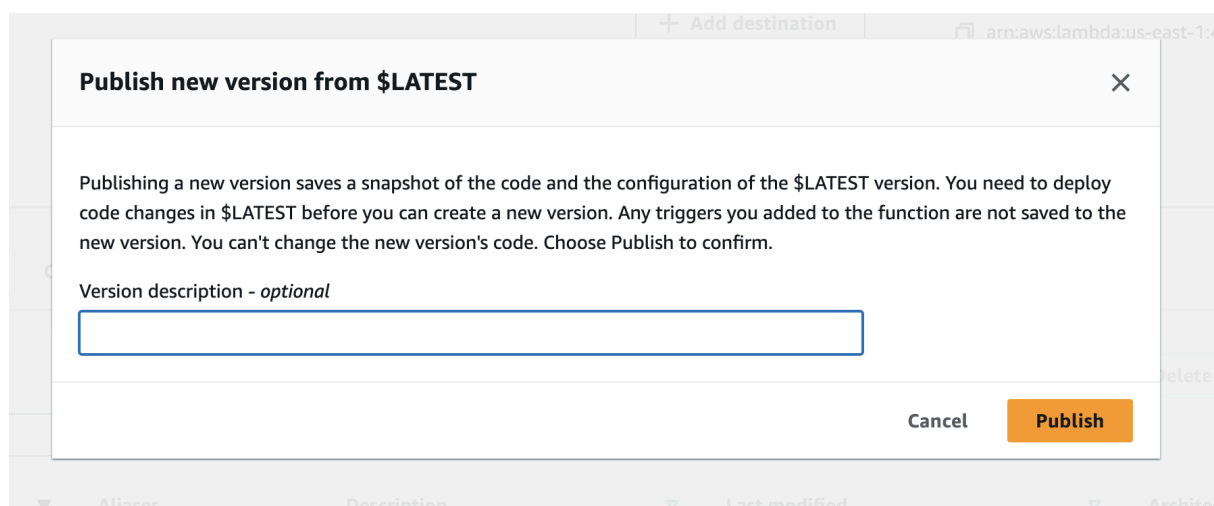
The screenshot shows the AWS Lambda console interface. At the top, there's a 'Code source' link and an 'Upload from' button. Below the navigation bar, the 'Test' button is highlighted in blue. The 'Execution results' tab is selected, showing a successful test execution. The response is a JSON object with status 200 and headers. The function logs show the execution details, including the request ID, context, and duration.

The account is safe since we only added certain policies, we delete the resources (the endpoint) after usage and keep monitoring the cost and other access.

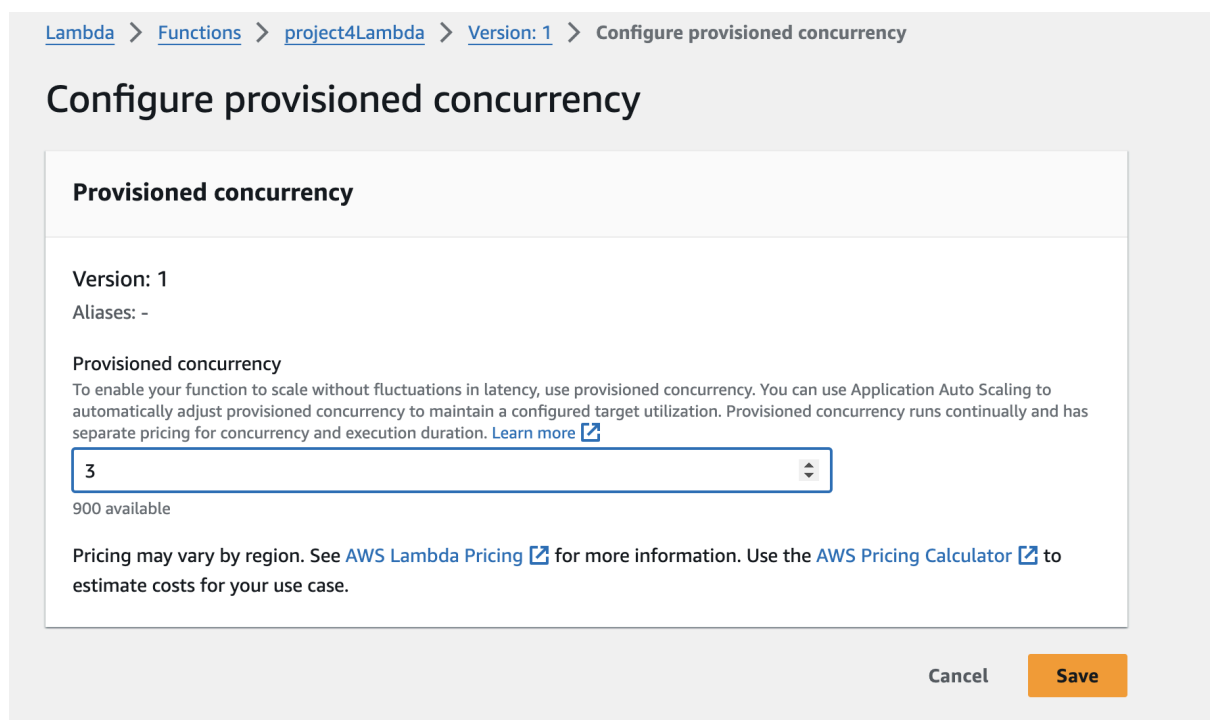
## Concurrency and Auto-Scaling

### Concurrency

I allocated three instances for the concurrency of my Lambda function to further reduce expenses.



The screenshot shows a modal dialog titled "Publish new version from \$LATEST". The dialog contains a close button (X) in the top right corner. The main text explains that publishing a new version saves a snapshot of the code and configuration of the \$LATEST version, and that any triggers added to the function are not saved to the new version. Below the text is a text input field labeled "Version description - optional". At the bottom right of the dialog are two buttons: "Cancel" and "Publish".



The screenshot shows the "Configure provisioned concurrency" page in the AWS Lambda console. The breadcrumb navigation at the top reads: "Lambda > Functions > project4Lambda > Version: 1 > Configure provisioned concurrency". The main heading is "Configure provisioned concurrency". Below this is a section titled "Provisioned concurrency". Inside this section, it shows "Version: 1" and "Aliases: -". The "Provisioned concurrency" section includes a text input field with the value "3" and a dropdown arrow. Below the input field, it says "900 available". At the bottom of the section, there is a note about pricing: "Pricing may vary by region. See [AWS Lambda Pricing](#) for more information. Use the [AWS Pricing Calculator](#) to estimate costs for your use case." At the bottom right of the page are two buttons: "Cancel" and "Save".



# Auto Scaling

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

Built-in scaling policy [Learn more](#)

Policy name

SageMakerEndpointInvocationScalingPolicy

Target metric

[SageMakerVariantInvocationsPerInstance](#)

Target value

100

Scale in cool down (seconds) - optional

10

Scale out cool down (seconds) - optional

300

☐ Disable scale in

Select if you don't want automatic scaling to delete instances when traffic decreases. [Learn more](#)

I set up auto scaling for my endpoint to activate when the number of invocations hits 100 per second, with a scale-in period of 10 seconds and a scale-out period of 300 seconds. This configuration was aimed at optimizing costs while managing high traffic.