

# Operationalizing a Machine Learning Project

## Notebook instance creation

We chose a ml.t2.medium instance type. This is because the actual notebook performs light operations and we don't need more power. The delegated instances perform the heavy workload, but we'll see that later.

Create notebook instance

Amazon SageMaker provides pre-built fully managed notebook instances that run Jupyter notebooks. The notebook instances include example code for common model training and hosting exercises. [Learn more](#)

Notebook instance settings

Notebook instance name

my-notebook-instance-1

Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

Notebook instance type

ml.t2.medium

Elastic Inference [Learn more](#)

none

Amazon SageMaker Notebook Instance is ending its standard support on Amazon Linux AMI (AL1). [Learn more](#)

Platform identifier [Learn more](#)

notebook-al1-v1

Additional configuration

Then, we create a new role. We only need the notebook to access to the edgarin-mlend bucket, so we configure the new role that way:

Create an IAM role

Passing an IAM role gives Amazon SageMaker permission to perform actions in other AWS services on your behalf. Creating a role here will grant permissions described by the [AmazonSageMakerFullAccess](#) IAM policy to the role you create.

The IAM role you create will provide access to:

S3 buckets you specify - optional

Any S3 bucket

Allow users that have access to your notebook instance access to any bucket and its contents in your account.

Specific S3 buckets

edgarin-mlend

Comma delimited. ARNs, "\*" and "/" are not supported.

None

Any S3 bucket with "sagemaker" in the name

Any S3 object with "sagemaker" in the name

Any S3 object with the tag "sagemaker" and value "true"

[See Object tagging](#)

S3 bucket with a Bucket Policy allowing access to SageMaker

[See S3 bucket policies](#)

Cancel

Create role

Next step is to configure the other security settings. This includes

- Selecting the newly created role
- Disable root access to the notebook
- Select a specific vpc, subnet and security group

IAM role

Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the [AmazonSageMakerFullAccess](#) IAM policy attached.

AmazonSageMaker-ExecutionRole-20220128T222182

Success! You created an IAM role.

AmazonSageMaker-ExecutionRole-20220128T222182

Root access - optional

Enable - Give users root access to the notebook

Disable - Don't give users root access to the notebook

Lifecycle configurations always have root access

Encryption key - optional

Encrypt your notebook data. Choose an existing KMS key or enter a key's ARN.

No Custom Encryption

▼ Network - optional

VPC - optional

Default vpc-0964b125a0511c8d9 (172.31.0.0/16)

Subnet

Choose a subnet in an availability zone supported by Amazon SageMaker.

subnet-02c9f4933f917c4de (172.31.0.0/20) | us-east-1a

Security group(s)

sg-0d5c3160fea95c893 (default)

Direct internet access

Enable — Access the internet directly through Amazon SageMaker

Disable — Access the internet through a VPC

To train or host models from a notebook, you need internet access. To enable internet access, make sure that your VPC has a NAT gateway and your security group allows outbound connections. [Learn more](#)

We also created a bucket to store the training data and the artifacts (the aforementioned edgarin-mlend bucket).

Amazon S3

edgarin-mlend

edgarin-mlend

Info

Objects

Properties

Permissions

Metrics

Management

Access Points

Bucket overview

AWS Region

US East (N. Virginia) us-east-1

Amazon Resource Name (ARN)

arn:aws:s3:::edgarin-mlend

Creation date

January 13, 2022, 11:33:43 (UTC-04:00)

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Edit

Bucket Versioning

Disabled

# Training and Endpoint deployment

All the process is performed in the `train-and-deploy.ipynb` notebook. In a nutshell, it does the following

- Data setup
- Training and HPO
- Best model deployment

After all this is done, the endpoint is deployed

Amazon SageMaker > Endpoints > pytorch-inference-2022-01-29-02-47-47-895

pytorch-inference-2022-01-29-02-47-47-895

Delete

Endpoint settings

Name

pytorch-inference-2022-01-29-02-47-47-895

Type

Real-time

ARN

arn:aws:sagemaker:us-east-1:784712499272:endpoint/pytorch-inference-2022-01-29-02-47-47-895

Last updated

Fri Jan 28 2022 22:50:34 GMT-0400 (hora de Bolivia)

Status

InService

URL

https://runtime.sagemaker.us-east-1.amazonaws.com/endpoints/pytorch-inference-2022-01-29-02-47-47-895/invocations

Creation time

Fri Jan 28 2022 22:47:48 GMT-0400 (hora de Bolivia)

Learn more about the API

## Training on EC2 instance

The choice is an `m1.m5.xlarge` instance to be able to train in a reasonable amount of time without much cost. Given this will be a 1 time run, a spot instance was chosen.  
If the code had had CUDA, we would have chosen a `p2.xlarge` instance (cheapest with good GPU).

	mac1	mac1.metal	12	32	EBS only	Yes	25 Gigabit
<input checked="" type="checkbox"/>	p2	p2.xlarge	4	61	EBS only	Yes	High
<input type="checkbox"/>	p2	p2.8xlarge	32	488	EBS only	Yes	10 Gigabit

For the security group, we chose to access it only from my ip with a keypair (ssh)

1. Choose AMI2. Choose Instance Type3. Configure Instance4. Add Storage5. Add Tags6. Configure Security Group

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow spe server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS port

Learn more about Amazon EC2 security groups.

Assign a security group:

Create a new security group

Select an existing security group

Security group name:

p2-security-group

Description:

Securit group created 2022-01-29T17:51:14.166-04:00

Type

SSH

Protocol

TCP

Port Range

22

Source

My IP

181.188.

Add Rule

Then, training is triggered via ssh This creates a model artifact in `TrainedModels/model.pth`

```
ec2-user@ip-172-31-24-188:~  
$ conda init <SHELL_NAME>  
  
Currently supported shells are:  
- bash  
- fish  
- tcsh  
- xonsh  
- zsh  
- powershell  
  
See 'conda init --help' for more information and options.  
  
IMPORTANT: You may need to close and restart your shell after running 'conda init'.  
  
[ec2-user@ip-172-31-24-188 ~]$ source activate pytorch  
(pytorch) [ec2-user@ip-172-31-24-188 ~]$ python solution.py  
Downloading: "https://download.pytorch.org/models/resnet50-0676ba61.pth" to /home/ec2-user/.cache/torch/hub/checkpoints/resnet50-0676ba61.pth  
100% | 97.8M/97.8M [00:00<00:00, 244MB/s]  
Starting Model Training  
saved  
(pytorch) [ec2-user@ip-172-31-24-188 ~]$
```

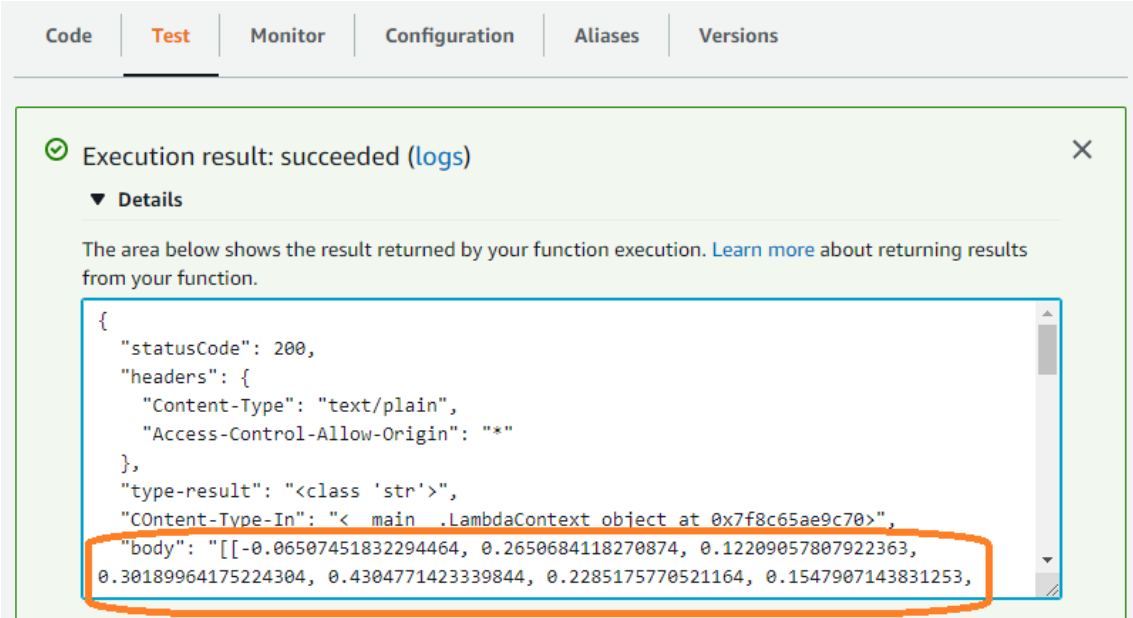
This script (ec2train1.py) is essentially the same as the one trained in script mode (hpo.py) with the following minor differences:

- hpo.py does have logging
- hpo.py is parameterized by command line or env vars (learning rate, batch size, file locations), whereas ec2train1.py has them hardcoded In fact, we could use hpo.py for both purposes, as long as we send the params when calling it.

## Endpoint invocation from Lambda

We create a lambda function that will basically take a dog image URL and will invoke the deployed endpoint to perform the prediction. It uses boto3's invoke\_endpoint api, sending a json with the image url. Let's keep in mind that behind the scenes, the inference2.py script will transform this url to a binary image that the endpoint can process.

Finally, the prediction -as defined in the training script- is an array of log probabilities for all breeds:



## About Lambda's security

Before executing the lambda, function, its execution role needs permissions to invoke the endpoint. So, a role with AWSSagemakerFullAccess policy was created:

Create role

Review

Provide the required information below and review this role before you create it.

Role name\*

lambda-sagemaker-role

Use alphanumeric and '+=, @, \_' characters. Maximum 64 characters.

Role description


Allows Lambda functions full access to Sagemaker, including endpoint invocation

Maximum 1000 characters. Use alphanumeric and '+=, @, \_' characters.

Trusted entities

AWS service: lambda.amazonaws.com

Policies

 AmazonSageMakerFullAccess [↗](#)

Permissions boundary

Permissions boundary is not set

No tags were added.

And then of course the role is attached to the lambda function.

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☒ Use an existing role

☐ Create a new role from AWS policy templates

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

lambda-sagemaker-role

↕

↻

[View the lambda-sagemaker-role role on the IAM console.](#)

Cancel

Save

In a real life scenario, it's strongly suggested to apply the least privilege principle. According to this, we would create a role with only permissions to invoke the lambda, but not full sagemaker access.

On the other hand, it's very important to configure resource based policies to define who specifically can execute the lambda function. For instance, if it's through API gateway, restrict the permissions to this service (and also configuring the correct authentication/authorization permissions in the api itself, but that's another story).

Finally, adding the lambda to a VPC needs to be considered, if applicable.

# Concurrency

We setup concurrency/autoscaling for both the lambda function and the endpoint

For lambda, we reserved 3 instances and also provisioned concurrency of 2 always-on with the possibility to scale to a 3rd instance.

Concurrency

Edit

Function concurrency

Reserved concurrency

Use reserved concurrency

3

Provisioned concurrency configurations (1)

↻

Edit


Remove

Add

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](#) [↗](#)

🔍 Find configuration

	Qualifier ▾	Type ▾	Provisioned concurrency ▾	Status ▾	Details
<input type="radio"/>	1	version	2	 Ready	-

For the endpoint, we configured autoscaling with a minimum of 2 and maximum of 3 instances

Configure variant automatic scaling

Deregister auto scaling

Variant automatic scaling

Variant name

AllTraffic

Instance type

ml.m5.large

Elastic Inference

-

Current instance count

2

Current weight

1

Minimum instance count

Maximum instance count

2

-

3

IAM role

Amazon SageMaker uses the following service-linked role for automatic scaling.

AWSServiceRoleForApplicationAutoScaling\_SageMakerEndpoint

Also, we want to spin up instances when there are 5 or more simultaneous requests (Typo: Screenshot shows 2 as target value but the final config was actually 5)

Built-in scaling policy

Policy name

SageMakerEndpointInvocationScalingPolicy

Target metric

SageMakerVariantInvocationsPerInstance

Target value

2

Scale in cool down (seconds) - optional

30

Scale out cool down (seconds) - optional

300

☐ Disable scale in

Select if you don't want automatic scaling to delete instances when traffic decreases.

Summarizing, we chose autoscaling of minimum 2 and maximum 3 instances for both lambda and endpoint. This is because due to our synchronous design, we can expect a 1-1 correspondence here. 1 lambda will derive the request to 1 endpoint.

However, if we had an async design, we could have many more lambdas to attend the requests, queue them and then have less endpoints to make the inferences as they get free.