

## Step1

### Initial setup, training and deployment

Firstly, I created a **ml.t2.medium** Sagemaker instance, which is within Free tier and still quite good for downloading, and uploading the dataset. For the training and deployment, inside the notebook, we create another instance which is suitable for image classification.

Notebook instances

Actions

Create notebook instance

Search notebook instances

<

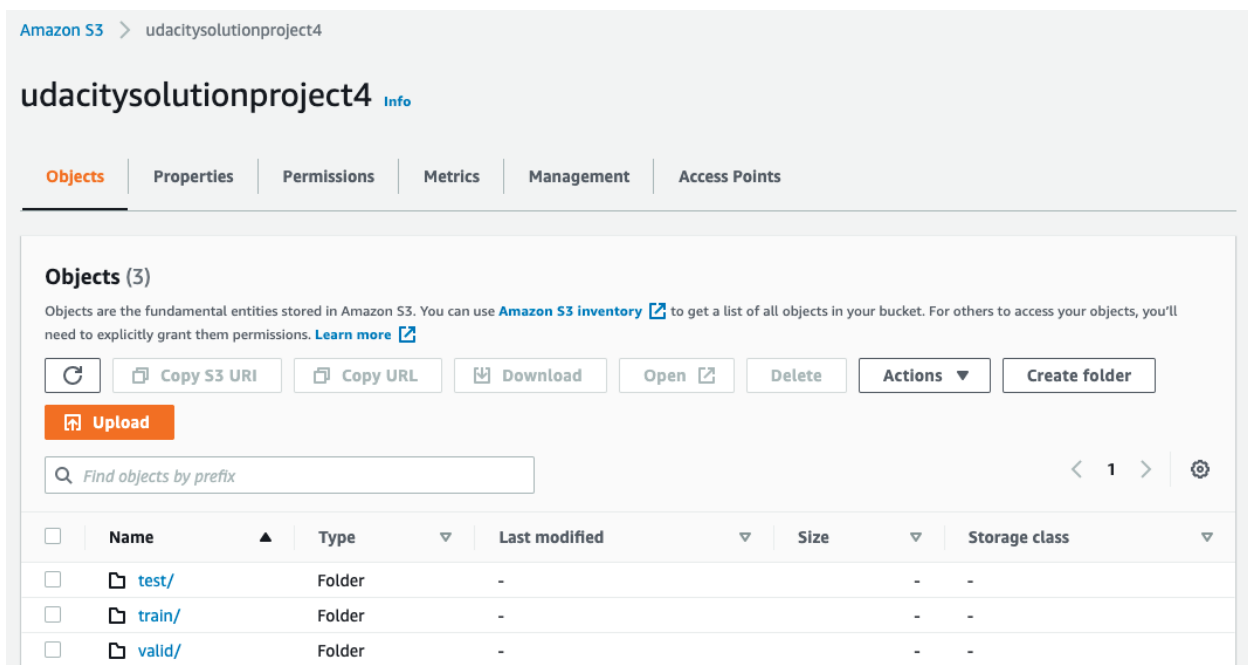
1

>

|                        | Name     | Instance     | Creation time          | Status                          | Actions                                   |
|------------------------|----------|--------------|------------------------|---------------------------------|---|
| <div><div></div></div> | project4 | ml.t2.medium | Jan 05, 2022 11:58 UTC | <div><div></div>InService</div> | <div>Open Jupyter   Open JupyterLab</div> |

### Download data to an S3 bucket

Once the notebook instance is “In Service”, we open a jupyterLab and first download the dataset and unzip it, then create a bucket in S3 and upload the unzipped dataset to the bucket.



| Amazon S3 > udacitysolutionproject4  |                                      |                                   |                                   |                               |                                 |         |               |
|--|--------------------------------------|-----------------------------------|-----------------------------------|-------------------------------|---------------------------------|---------|---------------|
| udacitysolutionproject4 Info   |                                      |                                   |                                   |                               |                                 |         |               |
| Objects  | Properties                           | Permissions                       | Metrics                           | Management                    | Access Points                   |         |               |
| Objects (3)  |                                      |                                   |                                   |                               |                                 |         |               |
| Objects are the fundamental entities stored in Amazon S3. You can use <a href="#">Amazon S3 inventory</a> to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. <a href="#">Learn more</a> |                                      |                                   |                                   |                               |                                 |         |               |
| <input type="checkbox"/>   | <input type="checkbox"/> Copy S3 URI | <input type="checkbox"/> Copy URL | <input type="checkbox"/> Download | <input type="checkbox"/> Open | <input type="checkbox"/> Delete | Actions | Create folder |
| <input type="checkbox"/> Upload  |                                      |                                   |                                   |                               |                                 |         |               |
| Find objects by prefix   |                                      |                                   |                                   |                               |                                 |         |               |
| <input type="checkbox"/>   | Name                                 | Type                              | Last modified                     | Size                          | Storage class                   |         |               |
| <input type="checkbox"/>   | test/                                | Folder                            | -                                 | -                             | -                               |         |               |
| <input type="checkbox"/>   | train/                               | Folder                            | -                                 | -                             | -                               |         |               |
| <input type="checkbox"/>   | valid/                               | Folder                            | -                                 | -                             | -                               |         |               |

### Training and Deployment

To use a pre-trained model, we set a hyperparameter range. The estimator uses a **ml.m5.xlarge** instance which is appropriate for image processing. After finding the best training job out of two training jobs (**pytorch-training ...**), we run the estimator with the hyperparameter of the best training job. The outcome of the training with the best hyperparameters is named **dog-pytorch...**

|                       |   |                           |               |             |
|-----------------------|---|---------------------------|---------------|-------------|
| <input type="radio"/> | <a href="#">dog-pytorch-2022-01-05-12-43-18-817</a>       | Jan 05, 2022<br>12:43 UTC | 23<br>minutes | ✔ Completed |
| <input type="radio"/> | <a href="#">pytorch-training-220105-1215-002-21462345</a> | Jan 05, 2022<br>12:15 UTC | 22<br>minutes | ✔ Completed |
| <input type="radio"/> | <a href="#">pytorch-training-220105-1215-001-4c175d11</a> | Jan 05, 2022<br>12:15 UTC | 21<br>minutes | ✔ Completed |

After the training job is completed the endpoint is deployed.

**pytorch-inference-2022-01-05-13-21-52-770** Delete

**Endpoint settings**

|   |  |   |   |
|---|--|---|---|
| <b>Name</b><br>pytorch-inference-2022-01-05-13-21-52-770  | <b>Status</b><br>✔ InService   | <b>Type</b><br>Real-time  | <b>URL</b><br><a href="https://runtime.sagemaker.us-east-1.amazonaws.com/endpoints/pytorch-inference-2022-01-05-13-21-52-770/invocations">https://runtime.sagemaker.us-east-1.amazonaws.com/endpoints/pytorch-inference-2022-01-05-13-21-52-770/invocations</a><br><a href="#">Learn more about the API</a> |
| <b>ARN</b><br>arn:aws:sagemaker:us-east-1:119699166201:endpoint/pytorch-inference-2022-01-05-13-21-52-770 | <b>Creation time</b><br>Wed Jan 05 2022 15:21:53 GMT+0200 (Eastern European Standard Time) | <b>Last updated</b><br>Wed Jan 05 2022 15:24:29 GMT+0200 (Eastern European Standard Time) |   |

## Multi-instance training

I changed the estimator setting to have 4 instances for multiple instances training.

**Training jobs** ↻ Actions ▼ Create training job

< 1 > ⚙


|                       | Name ▼  | Creation time ▼           | Duration      | Status ▼    |
|-----------------------|---|---------------------------|---------------|-------------|
| <input type="radio"/> | <a href="#">dog-pytorch-multi-2022-01-05-13-46-39-508</a> | Jan 05, 2022<br>13:46 UTC | 24<br>minutes | ✔ Completed |

After completing the training, I deployed the endpoint.

pytorch-inference-2022-01-05-14-13-04-961

Delete

## Endpoint settings

| Name  | Status  | Type   | URL   |
|---|---|--|---|
| pytorch-inference-2022-01-05-14-13-04-961   |  InService | Real-time  | <a href="https://runtime.sagemaker.us-east-1.amazonaws.com/endpoints/pytorch-inference-2022-01-05-14-13-04-961/invocations">https://runtime.sagemaker.us-east-1.amazonaws.com/endpoints/pytorch-inference-2022-01-05-14-13-04-961/invocations</a><br><a href="#">Learn more about the API</a> |
| ARN   | Creation time   | Last updated   |   |
| arn:aws:sagemaker:us-east-1:119699166201:endpoint/pytorch-inference-2022-01-05-14-13-04-961 | Wed Jan 05 2022 16:13:05 GMT+0200 (Eastern European Standard Time)                          | Wed Jan 05 2022 16:15:30 GMT+0200 (Eastern European Standard Time) |   |

## Step 2

## EC2 Setup

In the EC2 dashboard, I launched an EC2 instance with “**Deep Learning AMI (Amazon Linux 2) Version 56.0**” and a **t2.micro** instance which was free tier. After the instance was launched, I needed to connect to it via my computer terminal. The SSH connection was created with “**ssh -i "my-key-pair.pem" ec2-user@ec2-3-86-138-221.compute-1.amazonaws.com**” command.

EC2 &gt; Instances &gt; i-0473abbe798f35a19

## Instance summary for i-0473abbe798f35a19

Info

Updated less than a minute ago



Connect

Instance state ▼

Actions ▼

## Instance ID

 i-0473abbe798f35a19

## IPv6 address

-

## Hostname type

IP name: ip-172-31-80-57.ec2.internal

## Instance type

t2.micro

## AWS Compute Optimizer finding

 Opt-in to AWS Compute Optimizer for recommendations. | [Learn more](#)


## Public IPv4 address

 3.86.138.221 | [open address](#)

## Instance state

 Running

## Private IP DNS name (IPv4 only)

 ip-172-31-80-57.ec2.internal

## Elastic IP addresses

-

## IAM Role

-

## Private IPv4 addresses

 172.31.80.57

## Public IPv4 DNS

 ec2-3-86-138-221.compute-1.amazonaws.com | [open address](#)

## Answer private resource DNS name

-

## VPC ID

 vpc-0f2e92ad247707894

## Subnet ID

 subnet-05cfd1376315534a9

## Training and saving on EC2

Once the ssh connection was established, I followed the instruction and downloaded the dataset, created a training script named **solution.py** , and trained the model once again within EC2. The trained model was saved in the TrainedModels directory:

```

❶ ❷ ❸ ❹ fsiavash — ec2-user@ip-172-31-80-57:~ — ssh -i my-key-pair.pem ec2-u...
inflating: dogImages/valid/132.Xoloitzcuintli/Xoloitzcuintli_08299.jpg
inflating: dogImages/valid/132.Xoloitzcuintli/Xoloitzcuintli_08301.jpg
inflating: dogImages/valid/132.Xoloitzcuintli/Xoloitzcuintli_08304.jpg
creating: dogImages/valid/133.Yorkshire_terrier/
inflating: dogImages/valid/133.Yorkshire_terrier/Yorkshire_terrier_08333.jpg
inflating: dogImages/valid/133.Yorkshire_terrier/Yorkshire_terrier_08334.jpg
inflating: dogImages/valid/133.Yorkshire_terrier/Yorkshire_terrier_08336.jpg
inflating: dogImages/valid/133.Yorkshire_terrier/Yorkshire_terrier_08348.jpg
[ec2-user@ip-172-31-80-57 ~]$ ls
LICENSE          README           dogImages        examples         tools
Nvidia_Cloud_EULA.pdf  anaconda3       dogImages.zip    src             tutorials
[ec2-user@ip-172-31-80-57 ~]$ mkdir TrainedModels
[ec2-user@ip-172-31-80-57 ~]$ vim solution.py
[ec2-user@ip-172-31-80-57 ~]$ ls
LICENSE          TrainedModels   dogImages.zip    src
Nvidia_Cloud_EULA.pdf  anaconda3       examples         tools
README           dogImages       solution.py      tutorials
[ec2-user@ip-172-31-80-57 ~]$ vim solution.py
[ec2-user@ip-172-31-80-57 ~]$ python solution.py
Traceback (most recent call last):
  File "solution.py", line 1, in <module>
    import numpy as np
ImportError: No module named numpy
[ec2-user@ip-172-31-80-57 ~]$ source activate pytorch_latest_p37
(pytorch_latest_p37) [ec2-user@ip-172-31-80-57 ~]$ python solution.py
Downloading: "https://download.pytorch.org/models/resnet50-19c8e357.pth" to /home/ec2-user/.cache/torch/hub/checkpoints/resnet50-19c8e357.pth
100%|██████████| 97.8M/97.8M [00:00<00:00, 114MB/s]
Starting Model Training
saved
(pytorch_latest_p37) [ec2-user@ip-172-31-80-57 ~]$ ls TrainedModels/
model.pth
(pytorch_latest_p37) [ec2-user@ip-172-31-80-57 ~]$ ls
LICENSE          TrainedModels   dogImages.zip    src
Nvidia_Cloud_EULA.pdf  anaconda3       examples         tools
README           dogImages       solution.py      tutorials
(pytorch_latest_p37) [ec2-user@ip-172-31-80-57 ~]$ ls TrainedModels/
model.pth
(pytorch_latest_p37) [ec2-user@ip-172-31-80-57 ~]$
```

## Comparing EC2 and Sagemaker

The codes that trained the models are the same in both instances. The main differences between these two approaches are 1) The EC2 does not save the dataset or upload the trained model into S3, while Sagemaker downloads the dataset in S3 and saves the trained model here as well. 2) Sagemaker uses external scripts for hpo and inference, whereas in the EC2 only one script includes all functionalities. 3) Sagemaker enables testing the

deployment of the endpoint within itself, while for EC2 a Lambda Function is required to test the deployment.

## Step 3

### Setting up a Lambda function

I changed the default name of the endpoint in the `Lambdafunction.py` in the starter file to the name of the deployed endpoint. Then, I copied the code and created a lambda function instance and added the code in there. The lambda function includes following functionalities:

- Creates a Boto3 client to communicate to the endpoint
- Gets the input data which is a url of an image
- Sends the image to the endpoint and receives the response
- The response body includes an array of 133 items, each represents a prediction score of a dog breed.
- By getting the index of the maximum item from the array, we can see what is the breed prediction of the input image.

## Step 4

### Lambda function security

In order to set the appropriate permission, we attached a policy as shown in the figure below.

The screenshot shows the AWS IAM console for an IAM role. The role is named `AWSLambdaBasicExecutionRole-ec27c9bd-9d5d-4c1f-97ec-a8c5a7de55e0`. It has the following details:

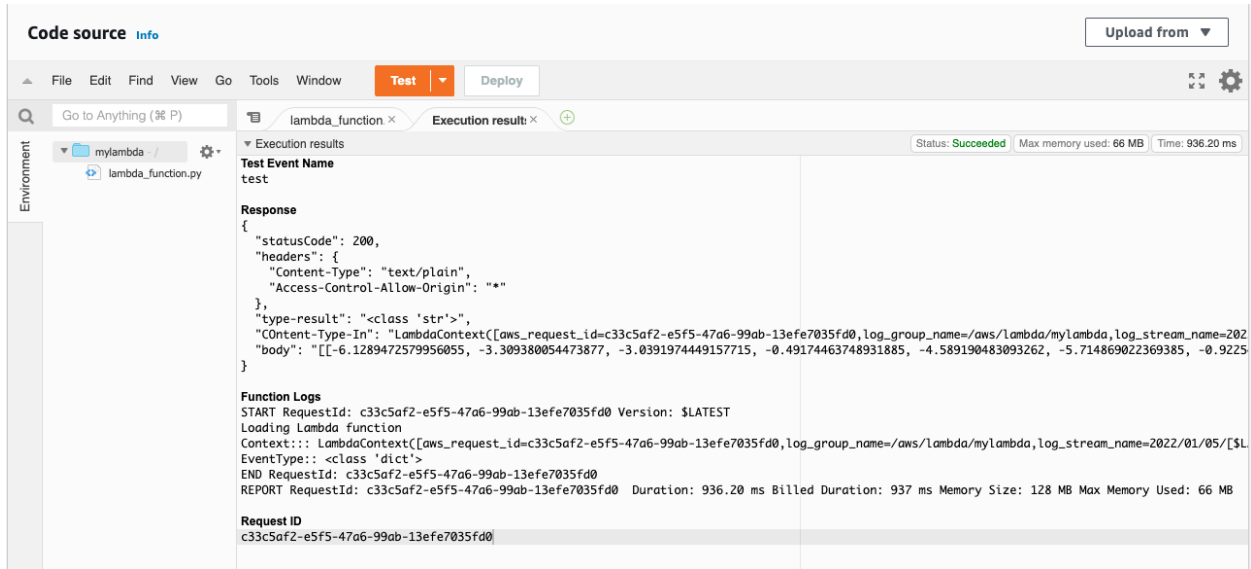
- Role description:** Edit
- Instance Profile ARNs:** [Icon]
- Path:** /service-role/
- Creation time:** 2022-01-05 18:40 UTC+0200
- Last activity:** Not accessed in the tracking period
- Maximum session duration:** 1 hour Edit

The **Permissions** tab is selected, showing 2 policies applied. The policies are:

| Policy name  | Policy type        |
|--|--------------------|
| <a href="#">AmazonSageMakerFullAccess</a>  | AWS managed policy |
| <a href="#">AWSLambdaBasicExecutionRole-ec27c9bd-9d5d-4c1f-97ec-a8c5a7de55e0</a> | Managed policy     |

## Lambda function testing

For testing the lambda function, we created a test with an image url as the input. The result of the test is shown below.



### Step 5

## Concurrency and auto scaling

To create concurrency in the lambda function, I changed the configuration and created a new version of the function. I provisioned 1 concurrency. For my current project this is more than required, however, to practice the procedure, I also added the auto scaling to the deployed endpoint. The min instance count is set to 1 and max is set to 3, scale-in and scale-out are 30 sec.

| Endpoint runtime settings |                |                  |                |                       |                   |                          |                          |                    |                   |
|---------------------------|----------------|------------------|----------------|-----------------------|-------------------|--------------------------|--------------------------|--------------------|-------------------|
|                           |                | Update weights   |                | Update instance count |                   | Configure auto scaling   |                          |                    |                   |
|                           | Variant name ▲ | Current weight ▼ | Desired weight | Instance type ▼       | Elastic Inference | Current instance count ▼ | Desired instance count ▼ | Instance min - max | Automatic scaling |
| <input type="radio"/>     | AllTraffic     | 1                | 1              | m1.m5.large           | -                 | 1                        | 1                        | 1 - 3              | Yes               |