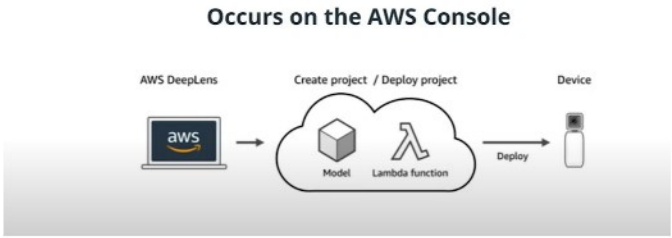
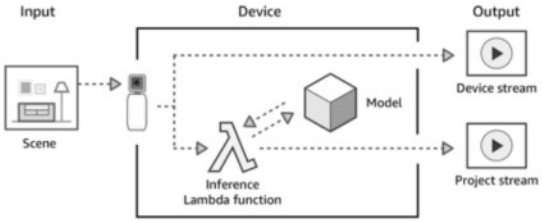
AWS

Wednesday, September 1, 2021 10:44 PM

Visão Computacional

AWS DeepLens





How AWS DeepLens works

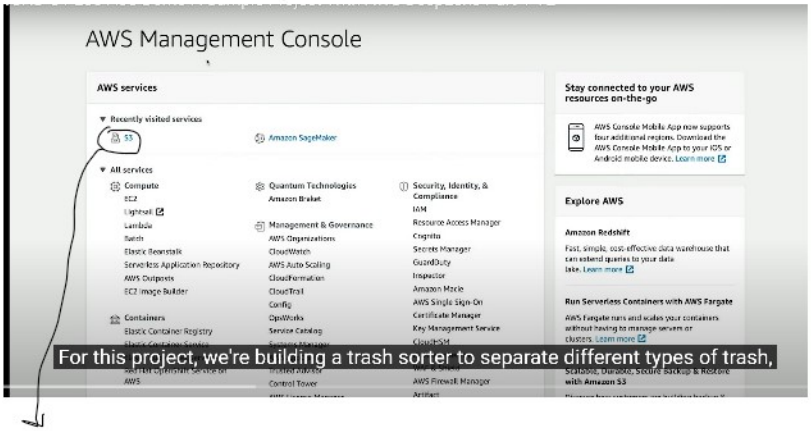
- First, you use the AWS console to create your project, store your data, and train your model.
- Then, you use your trained model on the AWS DeepLens device. On the device, the video stream from the camera is processed, inference is performed, and the output from inference is passed into two output streams:
- **Device stream** – The video stream passed through without processing.
- **Project stream** – The results of the model's processing of the video frames.

Four key components are required for an AWS DeepLens–based project.

1. **Collect your data:** Collect data and store it in an Amazon S3 bucket.
2. **Train your model:** Use a Jupyter Notebook in Amazon SageMaker to train your model.
3. **Deploy your model:** Use AWS Lambda to deploy the trained model to your AWS DeepLens device.
4. **View model output:** Use Amazon IoT Greenrass to view your model's output after the model is deployed.

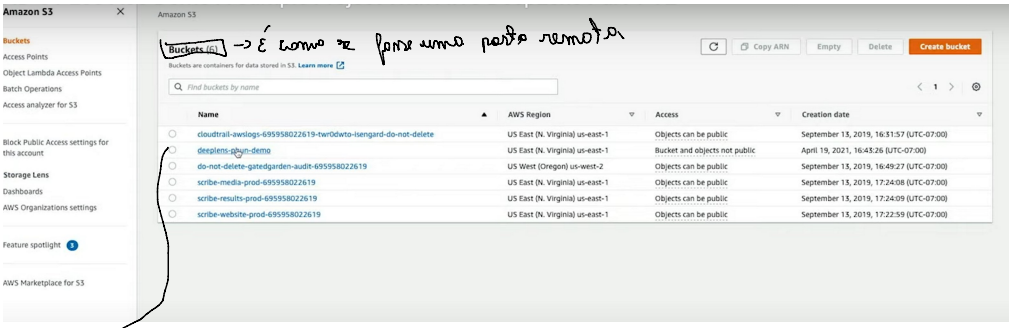
Passo a Passo ?

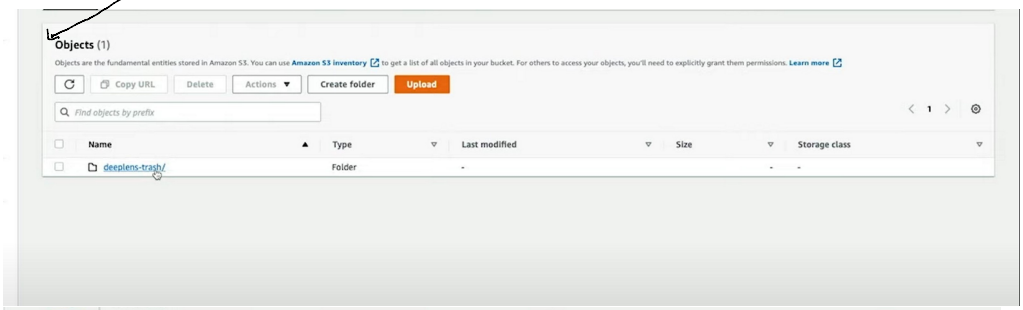
completo

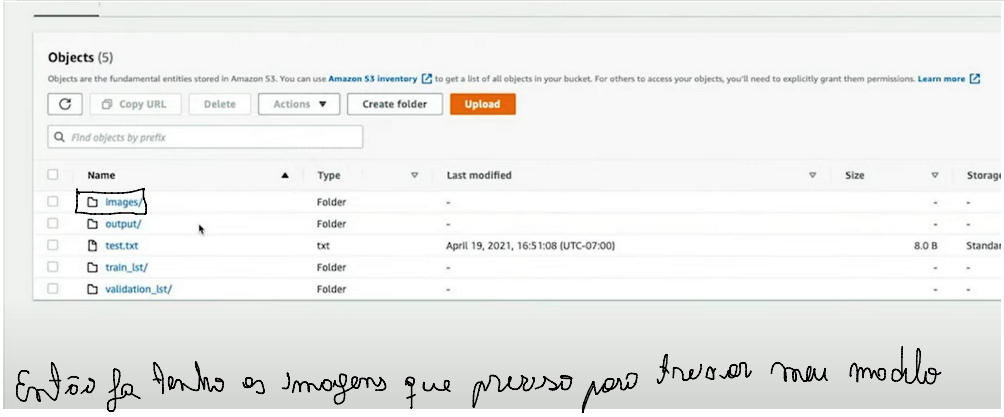


ND065 AWSND C1 L03 A08 Demo A Sample Project With AWS DeepLens Part 1 V2

https://youtu.be/d0auYkIX0VU

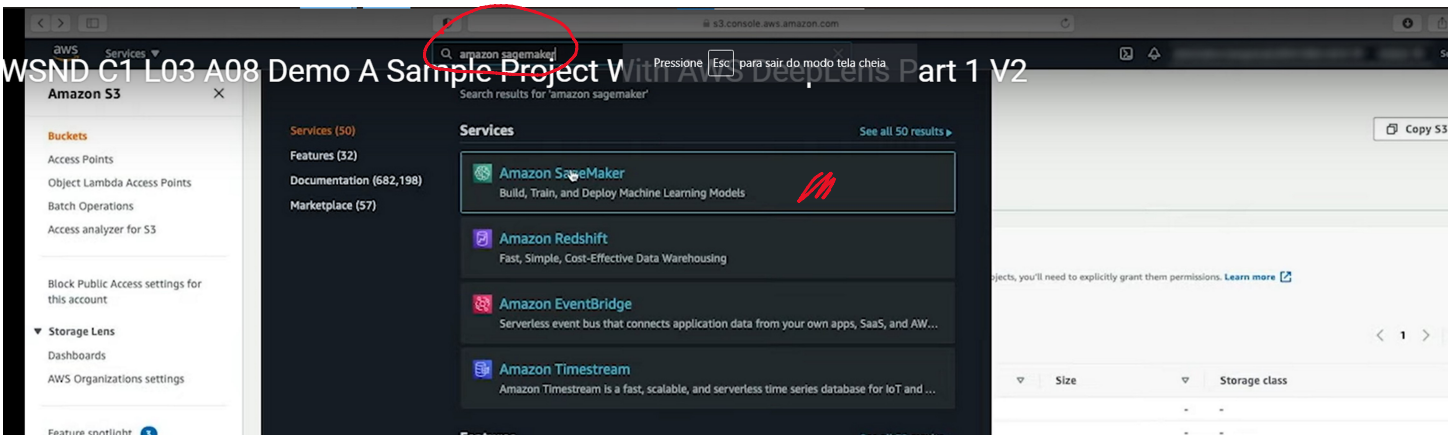


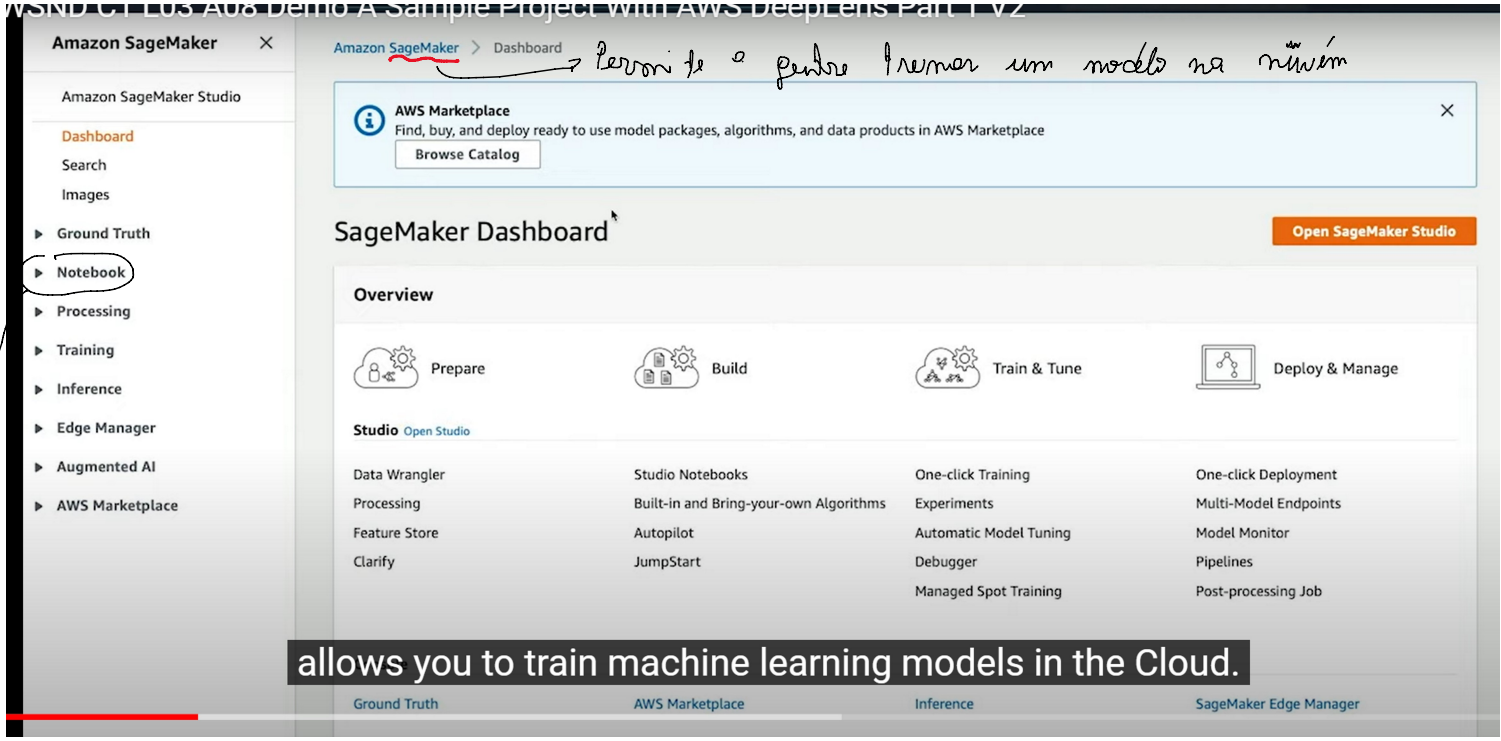


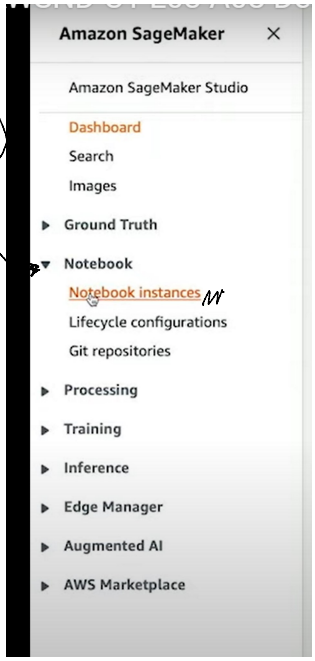


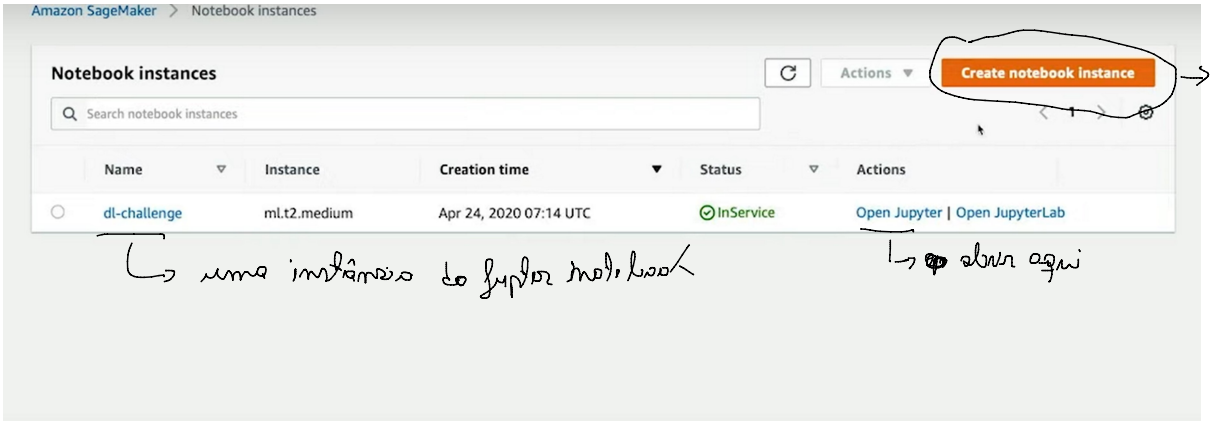
Então já tenho as imagens que preciso para treinar meu modelo

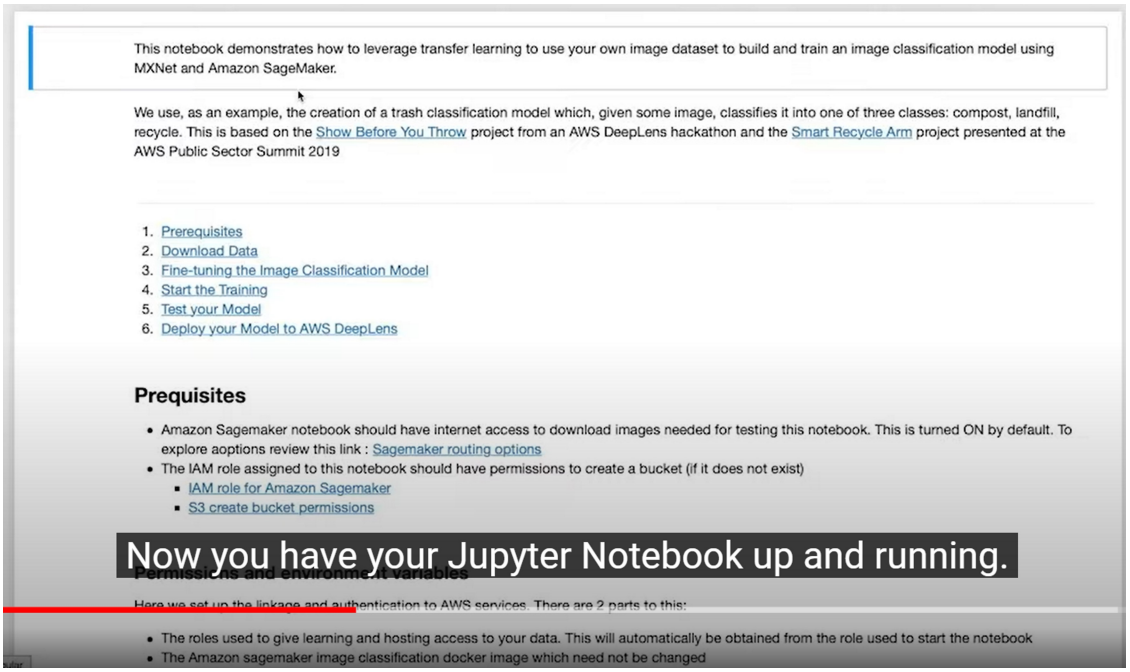
Agora vamos para o SageMaker usando a barra de pesquisa




Amazon SageMaker
Amazon SageMaker Studio
Dashboard
Search
Images
Ground Truth
Notebook
Processing
Training
Inference
Edge Manager
Augmented AI
AWS Marketplace
Amazon SageMaker > Dashboard
AWS Marketplace
Find, buy, and deploy ready to use model packages, algorithms, and data products in AWS Marketplace
Browse Catalog
SageMaker Dashboard
Open SageMaker Studio
Overview
Prepare
Build
Train & Tune
Deploy & Manage
Studio Open Studio
Data Wrangler
Processing
Feature Store
Clarify
Studio Notebooks
Built-in and Bring-your-own Algorithms
Autopilot
JumpStart
One-click Training
Experiments
Automatic Model Tuning
Debugger
Managed Spot Training
One-click Deployment
Multi-Model Endpoints
Model Monitor
Pipelines
Post-processing Job
allows you to train machine learning models in the Cloud.
Ground Truth
AWS Marketplace
Inference
SageMaker Edge Manager


Amazon SageMaker
Amazon SageMaker Studio
Dashboard
Search
Images
Ground Truth
Notebook
Notebook instances
Lifecycle configurations
Git repositories
Processing
Training
Inference
Edge Manager
Augmented AI
AWS Marketplace


Amazon SageMaker > Notebook instances
Notebook instances
Actions
Create notebook instance
Search notebook instances
Name
Instance
Creation time
Status
Actions
dl-challenge
ml.t2.medium
Apr 24, 2020 07:14 UTC
InService
Open Jupyter | Open JupyterLab



This notebook demonstrates how to leverage transfer learning to use your own image dataset to build and train an image classification model using MXNet and Amazon SageMaker.

We use, as an example, the creation of a trash classification model which, given some image, classifies it into one of three classes: compost, landfill, recycle. This is based on the Show Before You Throw project from an AWS DeepLens hackathon and the Smart Recycle Arm project presented at the AWS Public Sector Summit 2019

1. Prerequisites
2. Download Data
3. Fine-tuning the Image Classification Model
4. Start the Training
5. Test your Model
6. Deploy your Model to AWS DeepLens

Prequisites

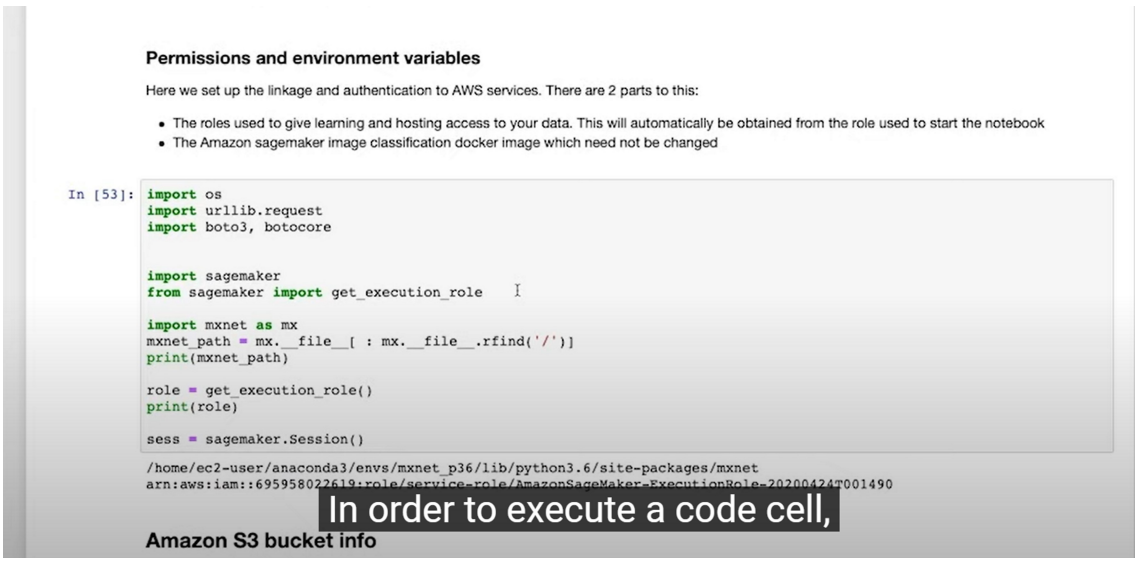
- Amazon Sagemaker notebook should have internet access to download images needed for testing this notebook. This is turned ON by default. To explore aoptions review this link : Sagemaker routing options
- The IAM role assigned to this notebook should have permissions to create a bucket (if it does not exist)
  - IAM role for Amazon Sagemaker
  - S3 create bucket permissions

Now you have your Jupyter Notebook up and running.

Permissions and environment variables

Here we set up the linkage and authentication to AWS services. There are 2 parts to this:

- The roles used to give learning and hosting access to your data. This will automatically be obtained from the role used to start the notebook
- The Amazon sagemaker image classification docker image which need not be changed



Permissions and environment variables

Here we set up the linkage and authentication to AWS services. There are 2 parts to this:

- The roles used to give learning and hosting access to your data. This will automatically be obtained from the role used to start the notebook
- The Amazon sagemaker image classification docker image which need not be changed

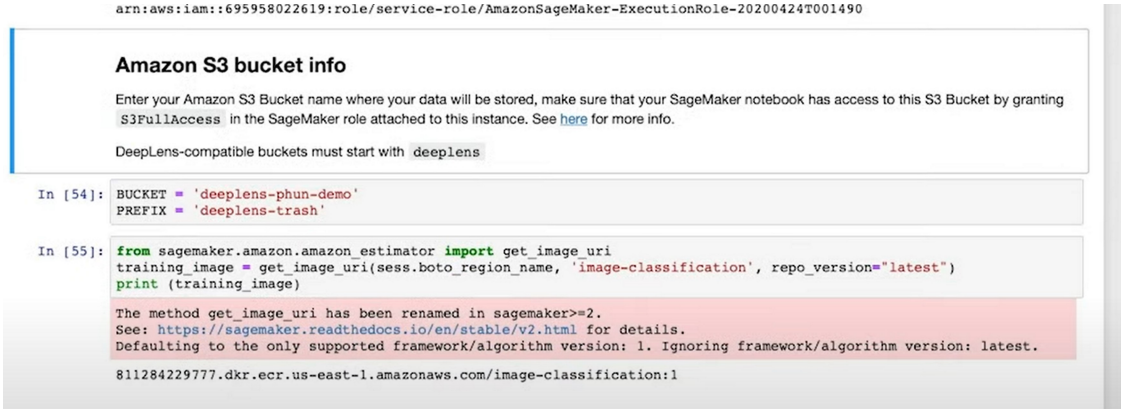
In [53]:
```
import os
import urllib.request
import boto3, botocore


import sagemaker
from sagemaker import get_execution_role

import mxnet as mx
mxnet_path = mx.__file__[ : mx.__file__.rfind('/')]
print(mxnet_path)

role = get_execution_role()
print(role)

sess = sagemaker.Session()
```

```
/home/ec2-user/anaconda3/envs/mxnet_p36/lib/python3.6/site-packages/mxnet
arn:aws:iam::695958022619:role/service-role/AmazonSageMaker-ExecutionRole-20200424T001490
```

In order to execute a code cell,

Amazon S3 bucket info

```
arn:aws:iam::695958022619:role/service-role/AmazonSageMaker-ExecutionRole-20200424T001490
```



Amazon S3 bucket info

Enter your Amazon S3 Bucket name where your data will be stored, make sure that your SageMaker notebook has access to this S3 Bucket by granting `S3FullAccess` in the SageMaker role attached to this instance. See here for more info.

DeepLens-compatible buckets must start with `deeplens`

In [54]:
```
BUCKET = 'deeplens-phun-demo'
PREFIX = 'deeplens-trash'
```

In [55]:
```
from sagemaker.amazon.amazon_estimator import get_image_uri
training_image = get_image_uri(sess.boto_region_name, 'image-classification', repo_version="latest")
print (training_image)
```

```
The method get_image_uri has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
Defaulting to the only supported framework/algorithm version: 1. Ignoring framework/algorithm version: latest.
811284229777.dkr.ecr.us-east-1.amazonaws.com/image-classification:1
```

Prepare data

It is assumed that your custom dataset's images are present in an S3 bucket and that different classes are separated by named folders, as shown in the following directory structure:

```
|-deeplens-bucket
   |-deeplens-trash

    |-images

        |-Compost

        |-Landfill

        |-Recycle
```

Since we are providing the data for you in this example, first we'll download the example data, unzip it and upload it to your bucket.

```
In [*]: !wget https://deeplens-public.s3.amazonaws.com/samples/deeplens-trash/trash-images.zip
```

```
In [57]: !rm -rf data/ && mkdir -p data
         !mkdir -p data/images
         !unzip -qq trash-images.zip -d data/images
         !rm trash-images.zip
```

.
.
.

Fine-tuning the Image Classification Model

Now that we are done with all the setup that is needed, we are ready to train our trash detector. To begin, let us create a `sageMaker.estimator.Estimator` object. This estimator will launch the training job.

Training parameters

There are two kinds of parameters that need to be set for training. The first one are the parameters for the training job. These include:

- **Training instance count**: This is the number of instances on which to run the training. When the number of instances is greater than one, then the image classification algorithm will run in distributed settings.
- **Training instance type**: This indicates the type of machine on which to run the training. Typically, we use GPU instances for these training
- **Output path**: This the s3 folder in which the training output is stored

```
In [15]: s3_output_location = 's3://{}/{}/output'.format(BUCKET, PREFIX)
         ic = sagemaker.estimator.Estimator(training_image,
                                            role,
                                            train_instance_count=1,
                                            train_instance_type='ml.p2.xlarge',
                                            train_volume_size = 50,
                                            train_max_run = 360000,
                                            input_mode= 'File',
                                            output_path=s3_output_location,
                                            sagemaker_session=sess,
                                            base_job_name='ic-trash')
```

and how many computers do we need to train the model?

```
train_instance_type has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
train_max_run has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
train_volume_size has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
```

.
.
.

Deploy to a Sagemaker endpoint

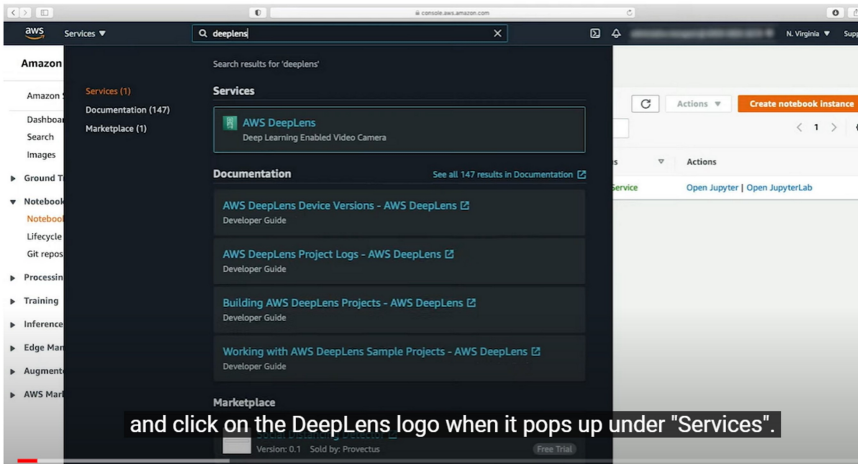
After training your model is complete, you can test your model by asking it to predict the class of a sample trash image that the model has not seen before. This step is called inference.

Amazon SageMaker provides an HTTPS endpoint where your machine learning model is available to provide inferences. For more information see the Amazon SageMaker documentation.

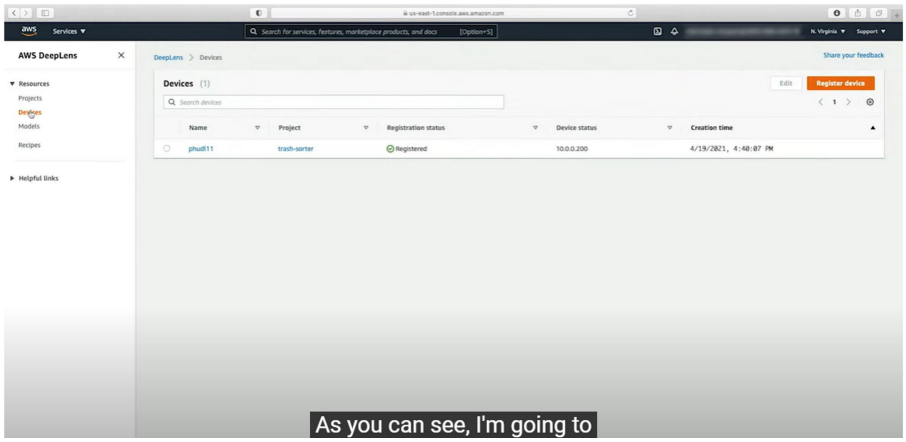
```
In [23]: ic_infer = ic.deploy(initial_instance_count=1, instance_type='ml.t2.medium')
         ----------------------!
```

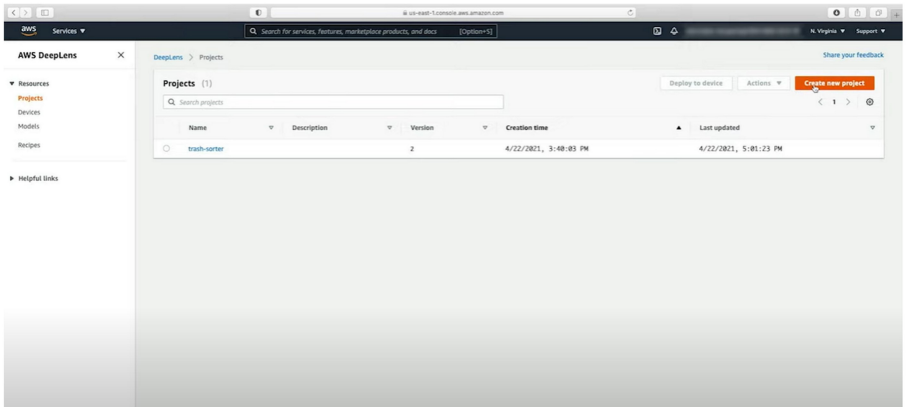
.
.
.

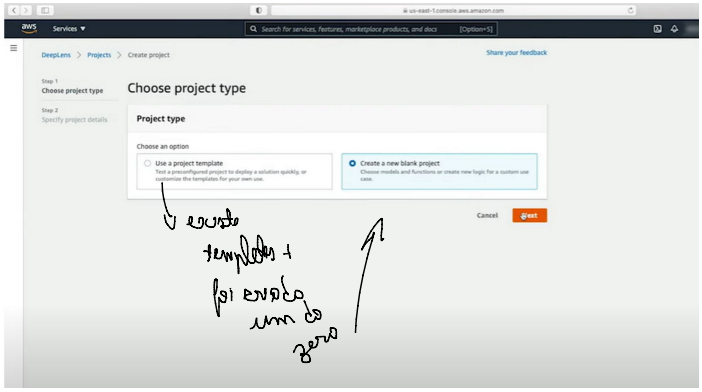
Demo Part 2: Deployment and Testing https://youtu.be/sy33KRApsWE

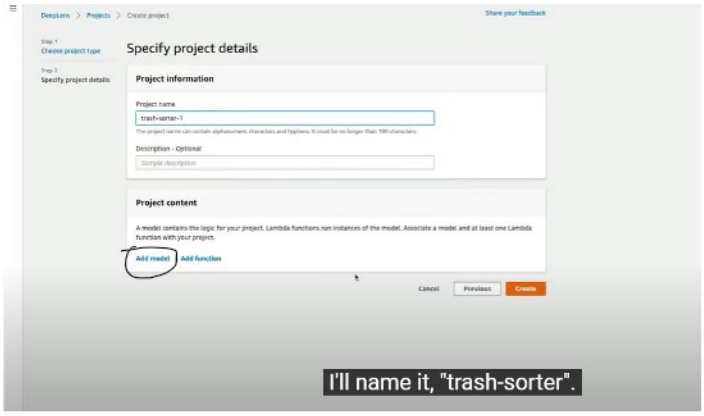


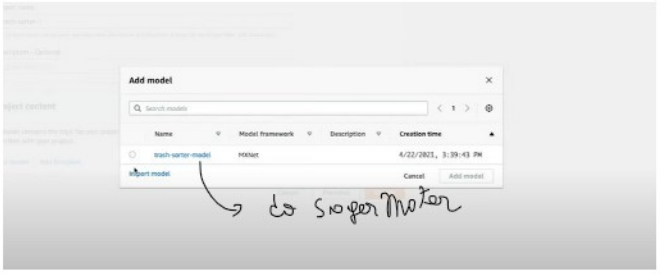
Demo Part 2: Deployment and Testing



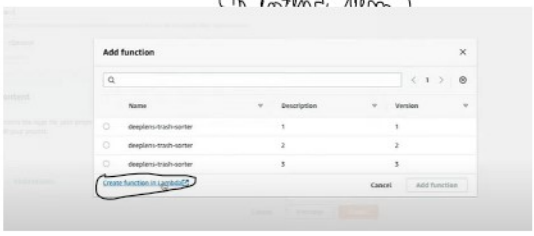






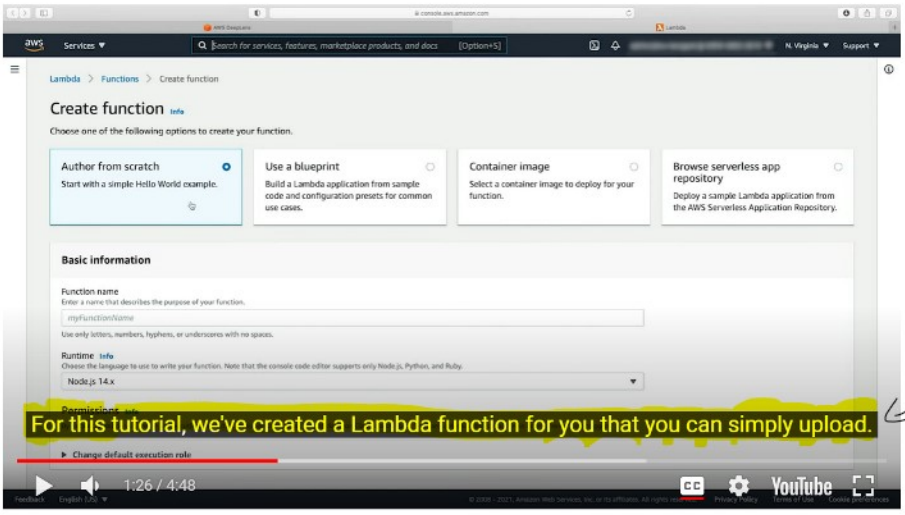


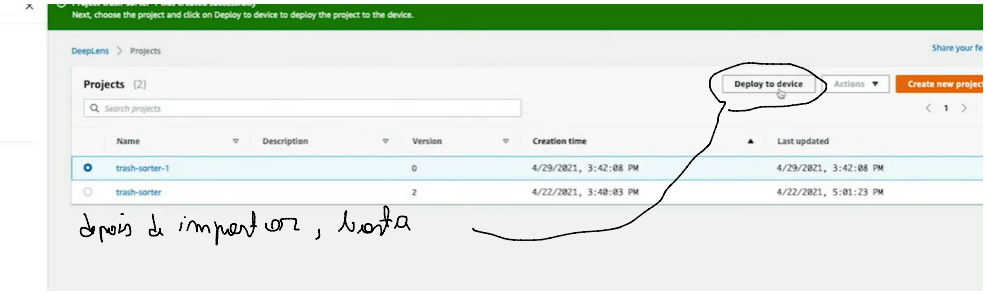
Lambda function hosts the code that will be running inference on AWS DeepLens.

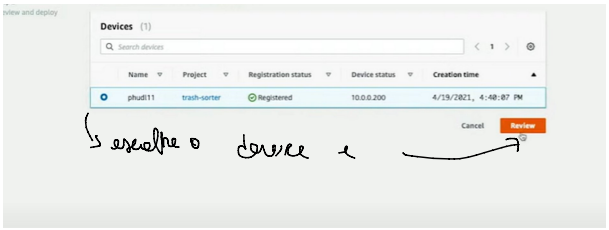


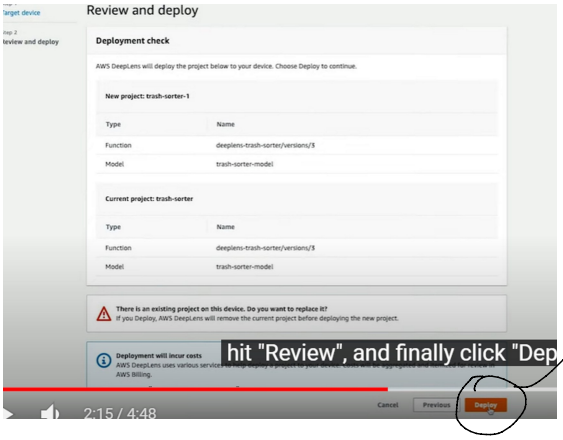
Next, you used AWS Lambda to deploy your model onto an AWS DeepLens device. Finally, once your model has been deployed to your device, you can use AWS IoT Greengrass to view the inference output from your model actively running on your AWS DeepLens device.

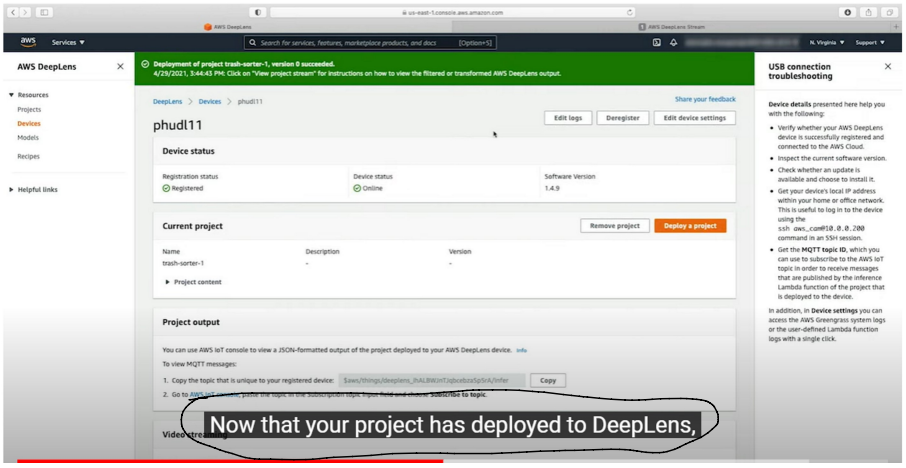
From <https://classroom.udacity.com/nanodegrees/nd065/parts/a5a4c41f-9cc7-48bd-9f00-582f35a7da53/modules/885b136b-2ca3-453a-8df1-4ea4b436b5da/lessons/8b79bd0c-6a77-40bc-8f96-b669c36d6103/concepts/50185845-e148-42e0-a709-bff908026ed2>

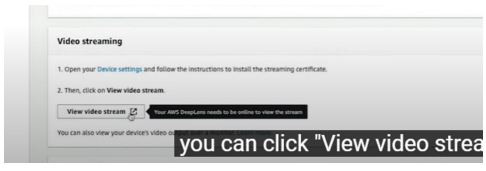


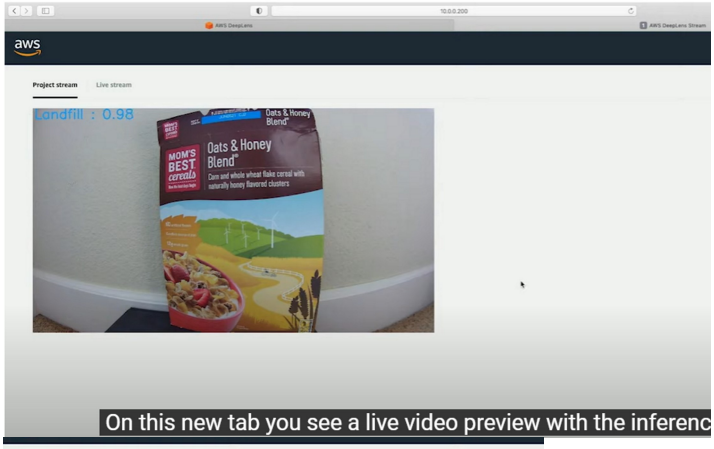

Next, choose the project and click on Deploy to device to deploy the project to the device.
DeepLens > Projects
Projects (2)
Deploy to device
Actions
Create new project
Name
Description
Version
Creation time
Last updated
trash-sorter-1
0
4/29/2021, 3:42:08 PM
4/29/2021, 3:42:08 PM
trash-sorter
2
4/22/2021, 3:40:03 PM
4/22/2021, 5:01:23 PM
depois de importar, basta

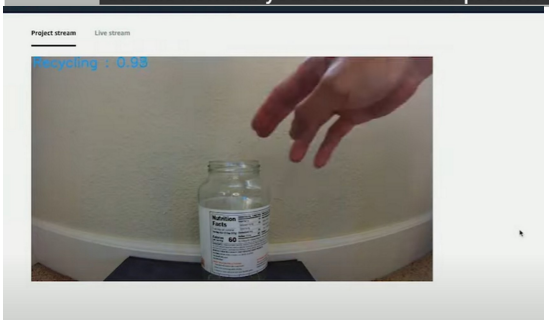

Devices (1)
Name
Project
Registration status
Device status
Creation time
phudl11
trash-sorter
Registered
10.0.0.200
4/19/2021, 4:40:07 PM
Cancel
Review
escolhe o device e


Review and deploy
Deployment check
AWS DeepLens will deploy the project below to your device. Choose Deploy to continue.
New project: trash-sorter-1
Current project: trash-sorter
There is an existing project on this device. Do you want to replace it?
If you Deploy, AWS DeepLens will remove the current project before deploying the new project.
Deployment will incur costs
hit "Review", and finally click "Dep
2:15 / 4:48
Cancel
Previous
Deploy
deploy


AWS DeepLens
Deployment of project trash-sorter-1, version 0 succeeded.
4/29/2021, 3:44:43 PM: Click on "View project stream" for instructions on how to view the filtered or transformed AWS DeepLens output.
DeepLens > Devices > phudl11
phudl11
Edit logs
Deregister
Edit device settings
Device status
Registration status
Registered
Device status
Online
Software Version
1.4.9
Current project
Remove project
Deploy a project
Project output
USB connection troubleshooting
Now that your project has deployed to DeepLens,


Video streaming
1. Open your Device settings and follow the instructions to install the streaming certificate.
2. Then, click on View video stream.
View video stream
you can click "View video strea


10.0.0.200
aws
Project stream
Live stream
Landfill : 0.98
Oats & Honey Blend
MOM'S BEST cereals
Oats & Honey Blend
Corn and whole wheat flake cereal with naturally honey flavored clusters
On this new tab you see a live video preview with the inferenc


Project stream
Live stream
Recycling : 0.93
Nutrition Facts