FRANKFURT UNIVERSITY OF APPLIED SCIENCES

High Integrity Systems
Euro Currency Note Identification using AWS Sagemaker

Summer Semester 2022
(Professor Dr. Christian Baun)

Group Members:

Muhamad Haseeb Anwar	1381906
Moeez Ur Rehman	1378333
Sahrish Kanwal	1382303
Harmain Haider	1359598


Adobe Stock | #146192283
ONE HUNDREDS
FIVE HUNDREDS
100 EURO
500 EURO
200 EURO
BCE ECB EZB EKT EKP 2002

But…
It’s Just a talk….:(
Oops… sorry…

Agenda

- What is Amazon SageMaker?
- What are the main capability of Amazon SageMaker?
- Components of Amazon SageMaker
- How Amazon SageMaker works with other AWS AI Services
- About SageMaker inference endpoint, Aws lambda functions, Aws Api gateway output, AWS Amplify
- Machine Learning Model Workflow
- About S3
- How do I get started using Amazon SageMaker?
  - Create,Train,Deploy your Model
- Q&A

Amazon SageMaker

Amazon SageMaker is a cloud machine-learning platform that was launched in November 2017. SageMaker enables developers to create, train, and deploy machine-learning models in the cloud. SageMaker also enables developers to deploy ML models on embedded systems.

OR

A managed service that provides the quickest and easiest way for your **data scientists** and **developers** to get **Machine Learning** models from idea to production.

What are the main capability of Amazon SageMaker?

Amazon SageMaker enables you to deploy inference pipelines so you can pass raw input data and execute pre-processing, predictions, and post-processing on real-time and batch inference. You can build feature data processing and feature engineering pipelines, and deploy these as part of the inference pipelines.

Amazon SageMaker's Components

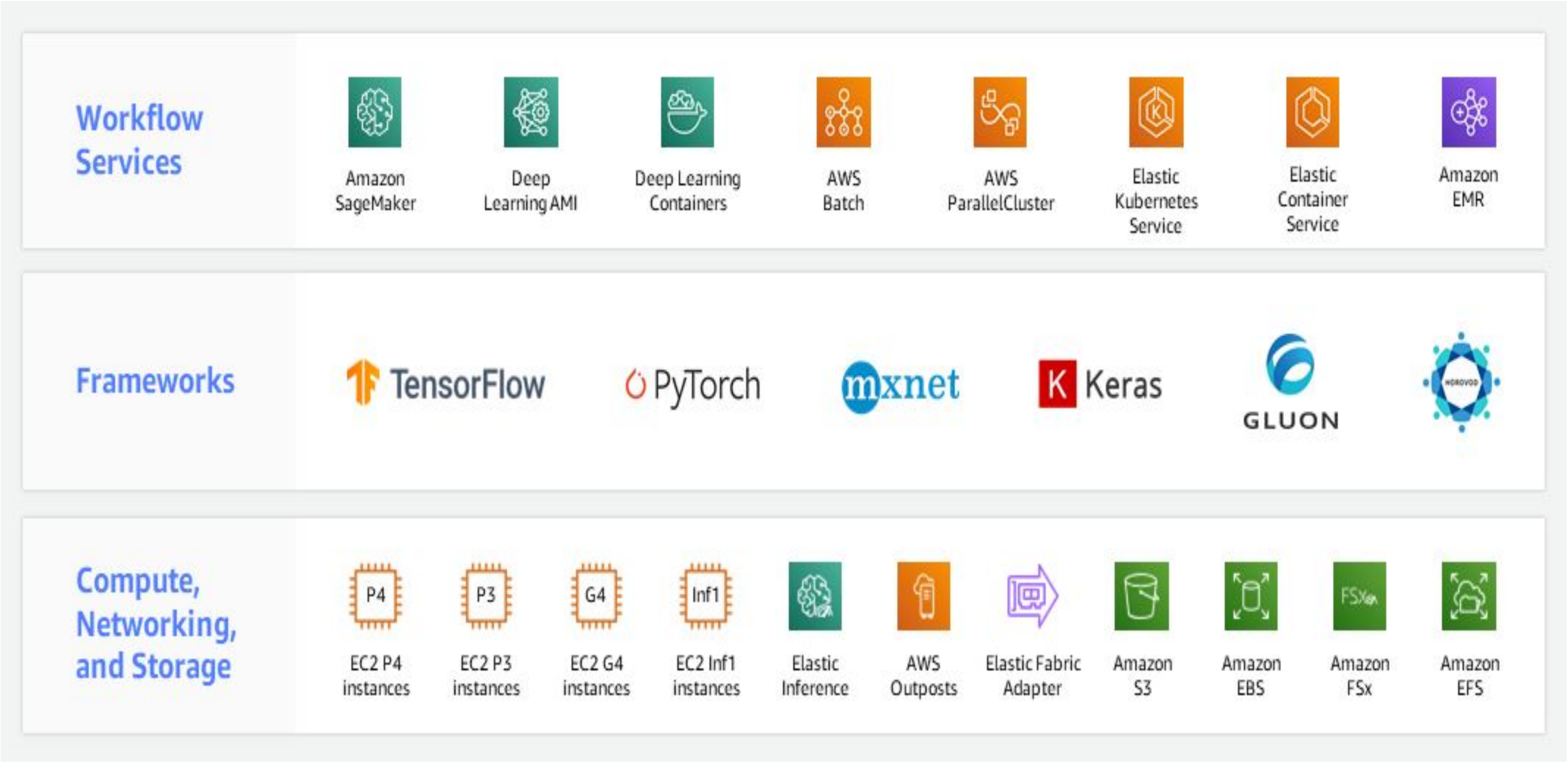
- Hyperparameter tuning job. ...
- Selecting the best hyperparameters. ...
- Training job with the best hyperparameters. ...
- Creating a model for deployment. ...
- **Deploying the inference endpoint.**

AWS machine learning infrastructure

You can choose a broad set of Machine Learning Services

AWS offers below mentioned services

- **Workflow services**, make it easier for you to manage and scale your underlying ML infrastructure.
- **Framework,** The next layer highlights that AWS ML infrastructure supports all of the leading ML frameworks.
- **Compute Networking and storage**, The bottom layer shows examples of compute, networking, and storage services that constitute the foundational blocks of ML infrastructure.



Ref: https://aws.amazon.com/machine-learning/infrastructure/

AWS lambda functions

- AWS Lambda is a serverless compute service that runs your code in response to events and automatically manages the underlying compute resources for you.

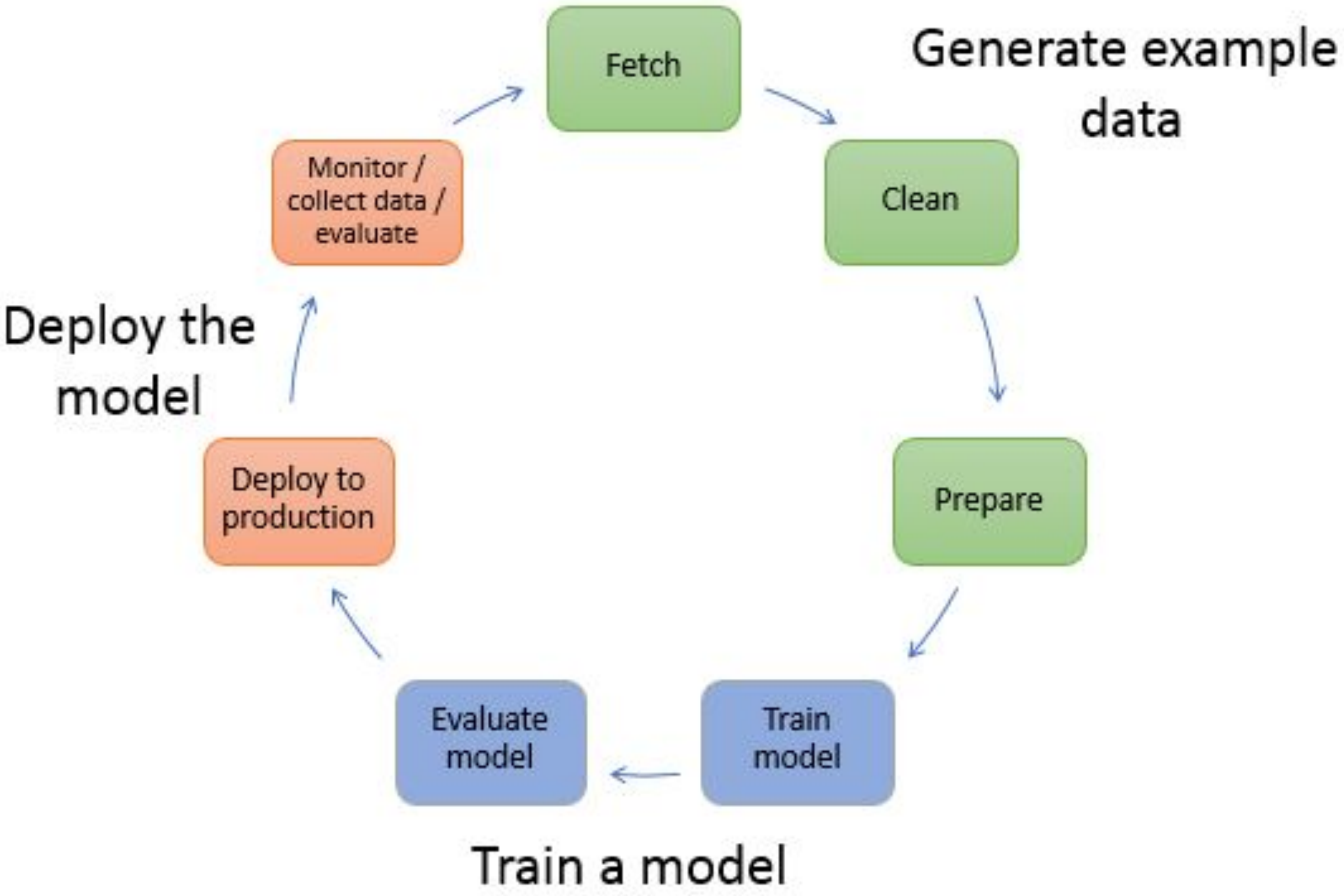
AWS Api gateway endpoint

- API Gateway is an AWS service that supports the following: Creating, deploying, and managing a RESTful application programming interface (API) to expose backend HTTP endpoints, AWS Lambda functions, or other AWS services.
-

AWS Amplify

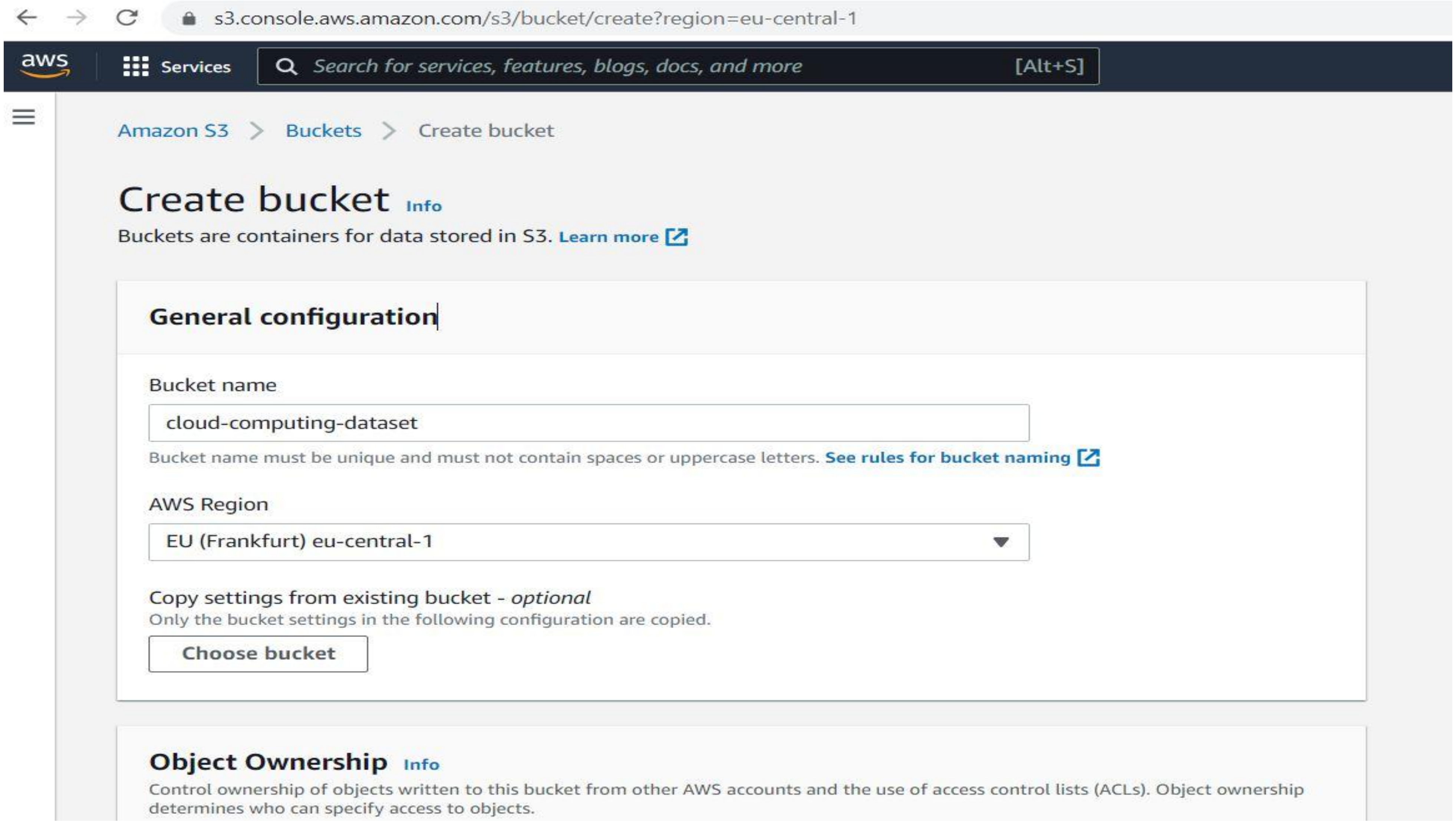
- AWS Amplify is a set of purpose-built tools and features that lets frontend web and mobile developers quickly and easily build full-stack applications on AWS, with the flexibility to leverage the breadth of AWS services as your use cases evolve.

Machine Learning Model Workflow

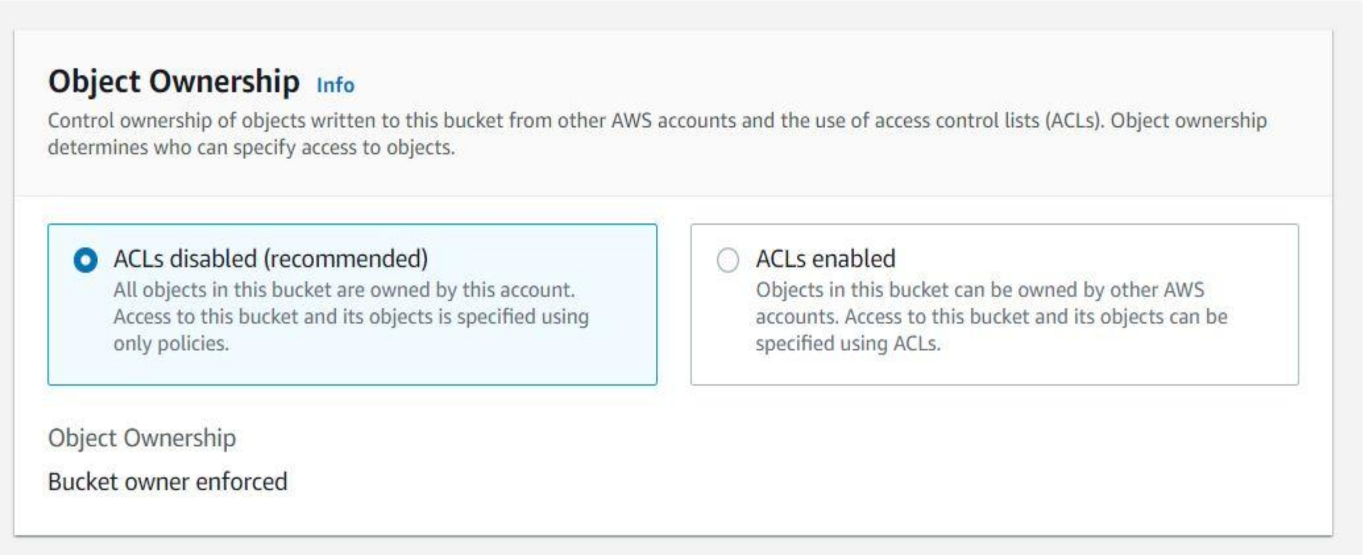


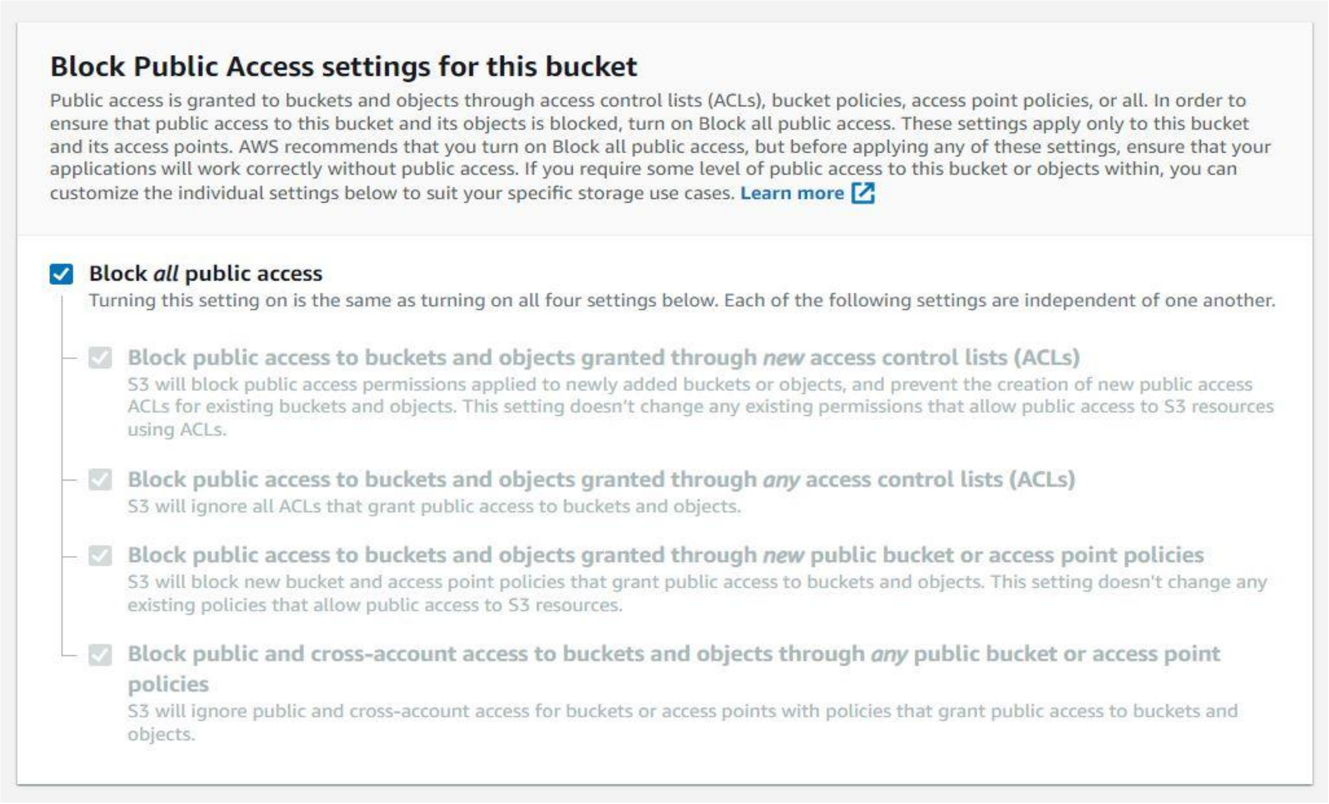
Ref: https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-mlconcepts.html

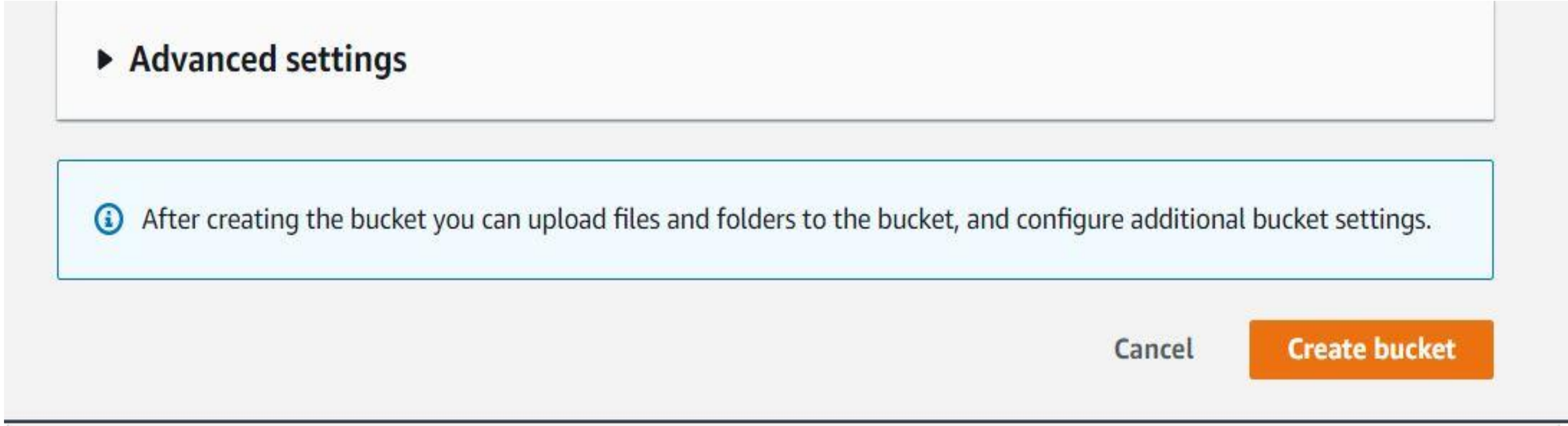
Creation of S3 Bucket

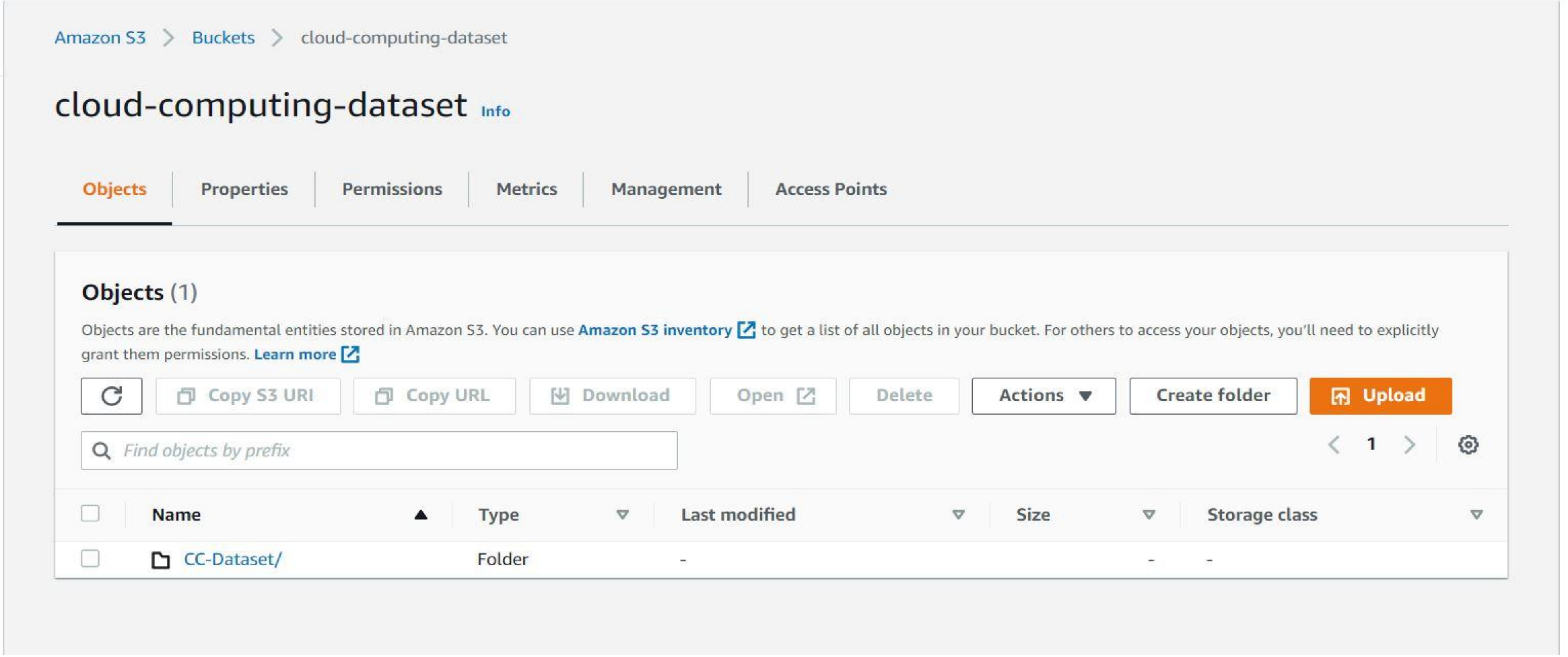


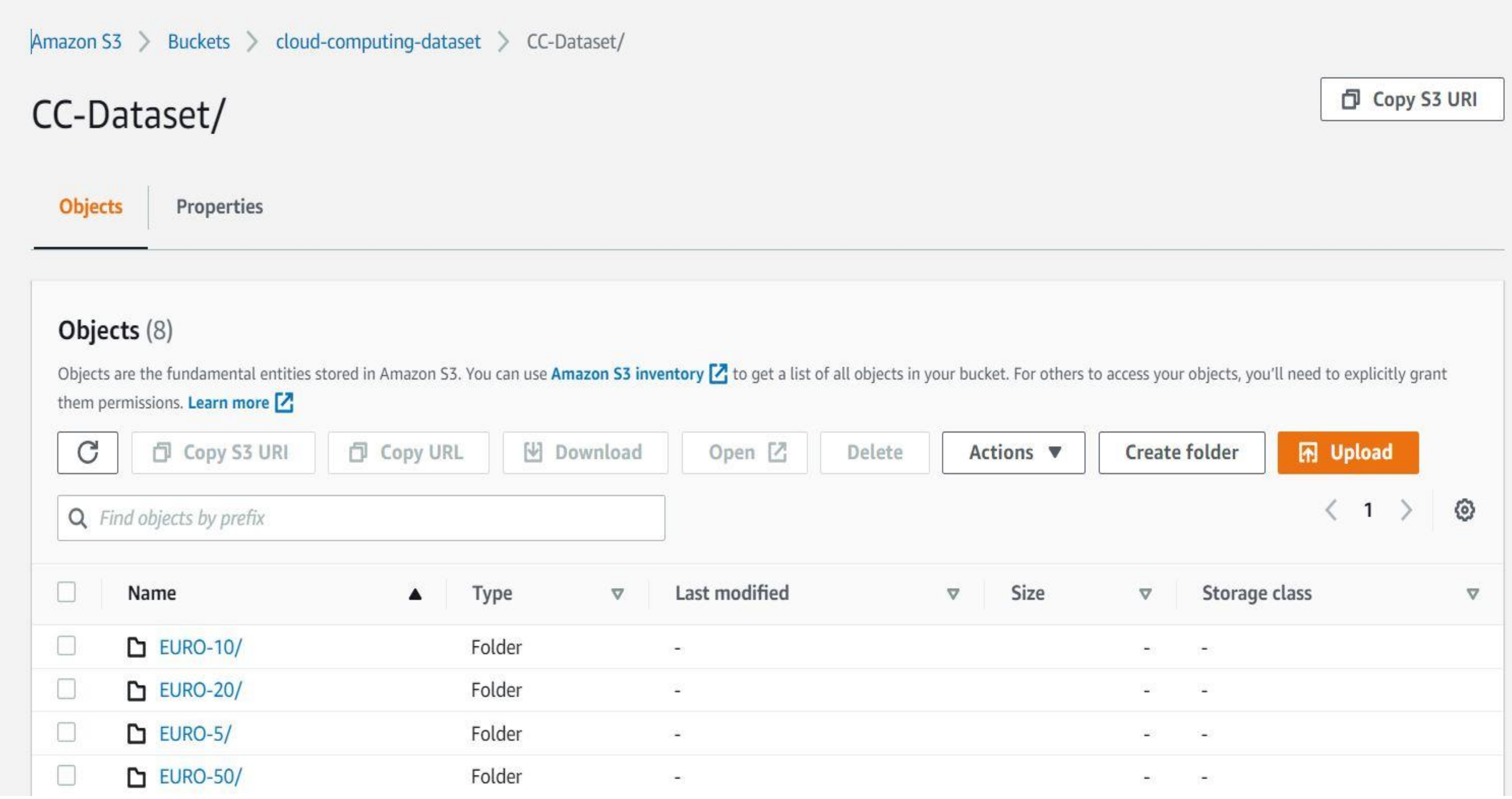
Setting up Security

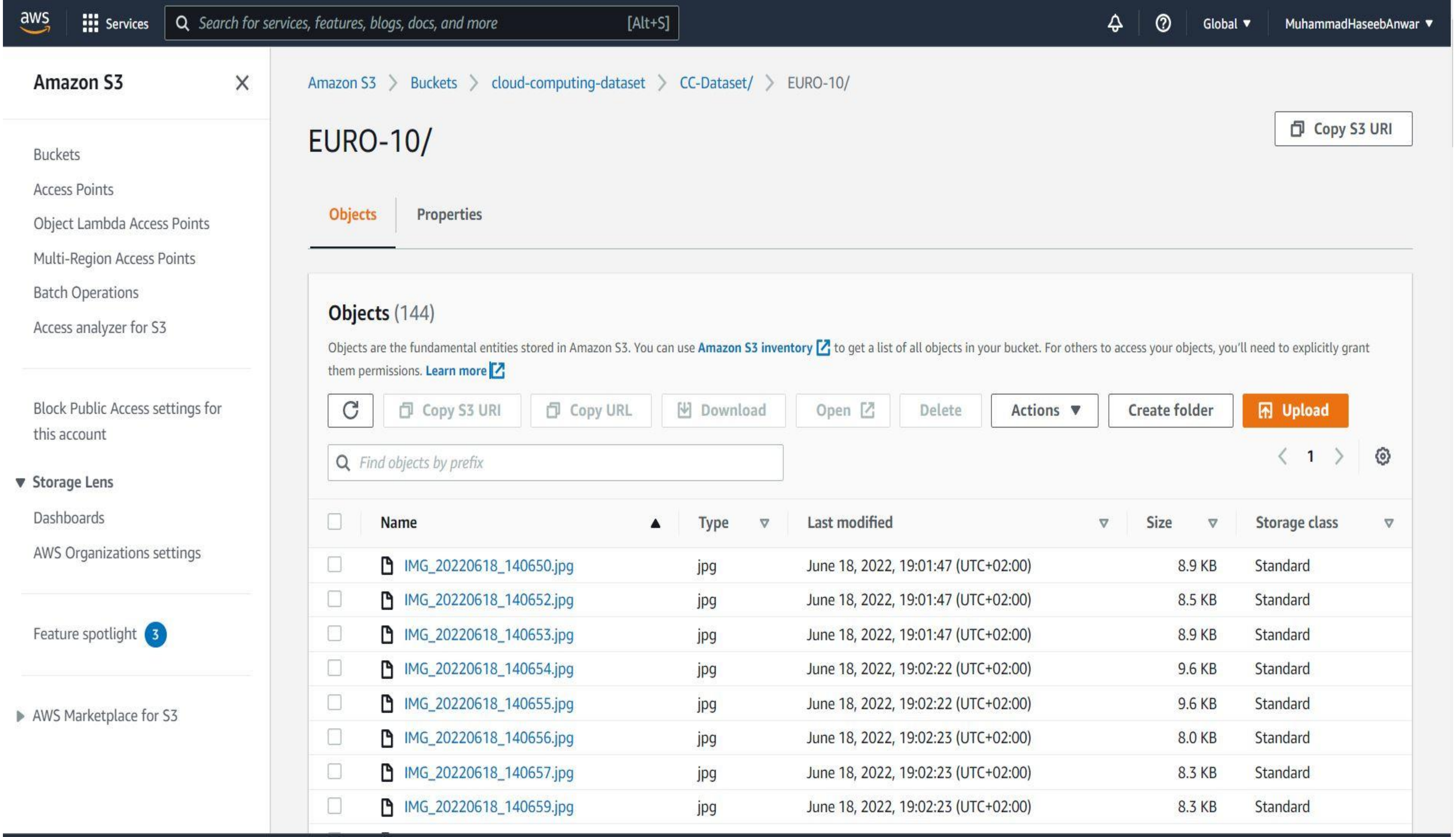





Advanced settings
After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings.
Cancel
Create bucket

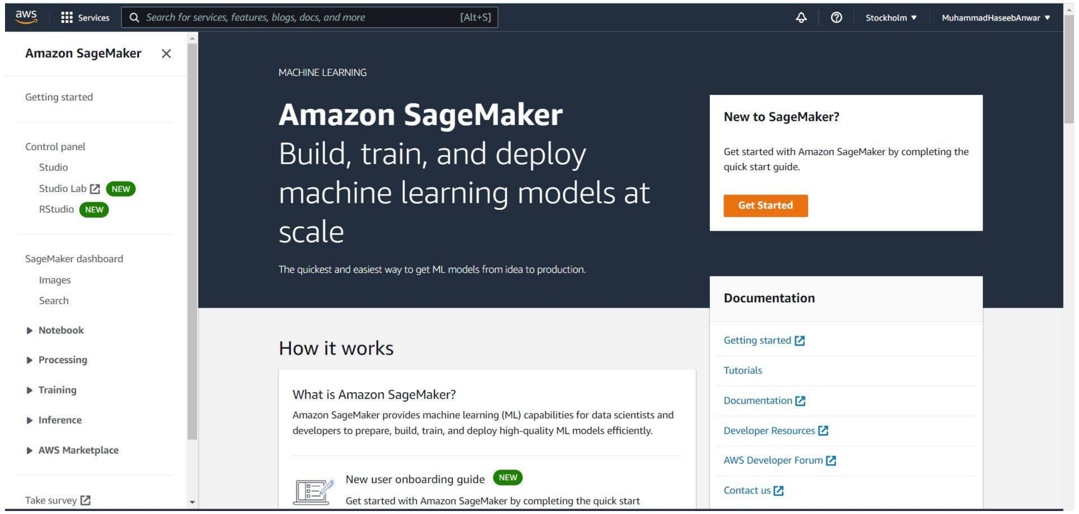

Amazon S3 > Buckets > cloud-computing-dataset
cloud-computing-dataset Info
Objects
Properties
Permissions
Metrics
Management
Access Points
Objects (1)
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Find objects by prefix
1
Name
Type
Last modified
Size
Storage class
CC-Dataset/
Folder
-
-
-

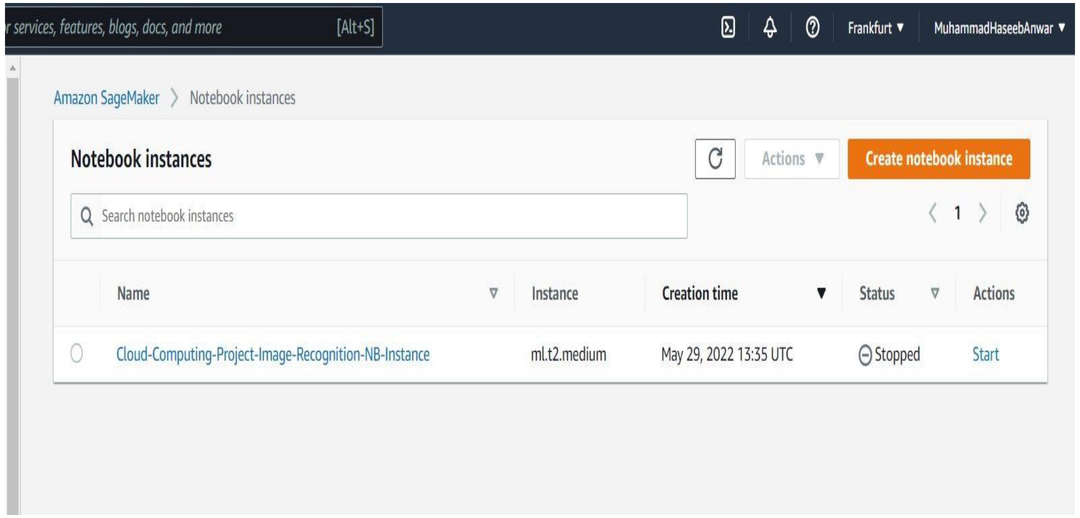

Amazon S3 > Buckets > cloud-computing-dataset > CC-Dataset/
CC-Dataset/
Copy S3 URI
Objects
Properties
Objects (8)
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Find objects by prefix
1
Name
Type
Last modified
Size
Storage class
EURO-10/
Folder
EURO-20/
Folder
EURO-5/
Folder
EURO-50/
Folder


aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Global
MuhammadHaseebAnwar
Amazon S3
Buckets
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
Access analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
AWS Organizations settings
Feature spotlight 3
AWS Marketplace for S3
Amazon S3 > Buckets > cloud-computing-dataset > CC-Dataset/ > EURO-10/
EURO-10/
Copy S3 URI
Objects
Properties
Objects (144)
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Find objects by prefix
1
Name
Type
Last modified
Size
Storage class
IMG_20220618_140650.jpg jpg June 18, 2022, 19:01:47 (UTC+02:00) 8.9 KB Standard
IMG_20220618_140652.jpg jpg June 18, 2022, 19:01:47 (UTC+02:00) 8.5 KB Standard
IMG_20220618_140653.jpg jpg June 18, 2022, 19:01:47 (UTC+02:00) 8.9 KB Standard
IMG_20220618_140654.jpg jpg June 18, 2022, 19:02:22 (UTC+02:00) 9.6 KB Standard
IMG_20220618_140655.jpg jpg June 18, 2022, 19:02:22 (UTC+02:00) 9.6 KB Standard
IMG_20220618_140656.jpg jpg June 18, 2022, 19:02:23 (UTC+02:00) 8.0 KB Standard
IMG_20220618_140657.jpg jpg June 18, 2022, 19:02:23 (UTC+02:00) 8.3 KB Standard
IMG_20220618_140659.jpg jpg June 18, 2022, 19:02:23 (UTC+02:00) 8.3 KB Standard

How do I get started using Amazon SageMaker?

- Create Notebook instance within sagemaker, choose your desired region, we have chosen eu-central-1
- Provide name of your notebook instance, and choose your instance type
- create a new IAM role or use already created IAM role
- After our notebook instance is created, we need to start this notebook instance after that we are going to write our model





Create your model

- Stored the name of S3 bucket and primary folder into variables so you can access them later.
- Importing the sagemaker library and setting up environment.
- We have used Sagemaker's built in **Image classification Algorithm** which is based on **Supervised Learning** and uses **Conventional Neural Networks (CNN).**

```
[12]: ###################### CLOUD COMPUTING PROJECT ##############################
######### Training machine learning models in the AWS Sagemaker ############
######### Supervised by: Prof. Dr. Christian Baun ###########################

######### Team Members:
######### Muhammad Haseeb Anwar
######### Moeez Ur Rehman
######### Sehrish Kanwal
######### Harmain Haidar

# S3 Bucket Name
bucket_name='cloud-computing-dataset'

# Our Main Folder inside the S3 bucket Which has subfolders of our classes
# One Sub-Folder will be considered as One Class
dataset_name = 'CC-Dataset'

print('Name of the bucket is: '+dataset_name)
print('Name of dataset folder is: '+bucket_name)
```

```
Name of the bucket is: CC-Dataset
Name of dataset folder is: cloud-computing-dataset
```

```
[13]: #Setting Up Our Environment

# Importing Sagemaker
# getting execution role of notebook
# defining algorithm type in Image_Uri method
import sagemaker
from sagemaker import get_execution_role
from sagemaker.amazon.amazon_estimator import get_image_uri

role = get_execution_role()
session = sagemaker.Session()
#sagemaker.image_uris.retrieve
#get_image_uri
image_uris = sagemaker.image_uris.retrieve(region=session.boto_region_name, framework ='image-classification')

print('Region name: '+session.boto_region_name)
print('Algorithm Used: image-classification ')
```

```
Region name: eu-central-1
Algorithm Used: image-classification
```

Continue…

- For data loading here, we are using **RecordIO** format for training. (RecordIO format converts images into binary data exchange formats)
- Recommendation: Sagemaker recommended storing images as records and packing them together, the major benefit is that Stored images in RecordIO format greatly reduce the size of the dataset on the disk.

```
[7]: # The SageMaker Image Classification algorithm supports both RecordIO and Image (jpg,Jpeg) formats
     # to train the images, in our project we are going to use the RecordIO format for training.

     # Why RecordIO?
     # Sagemaker recommend storing images as records and packing them together, the major benifit is
     # Storing images in Recordio format greatly reduces the size of the dataset on the disk.

     #here we specifiy the path of the script which converts images into RecordIO files

     BASE_DIRECTORY='/tmp'

     %env BASE_DIRECTORY=$BASE_DIRECTORY
     %env S3_BUCKET_NAME = $bucket_name
     %env DATASET_NAME = $dataset_name

     import sys,os

     suffix='/mxnet/tools/im2rec.py'
     im2rec = list(filter( (lambda x: os.path.isfile(x + suffix )), sys.path))[0] + suffix
     %env IM2REC=$im2rec

     env: BASE_DIR=/tmp
     env: S3_DATA_BUCKET_NAME=cloud-computing-dataset
     env: DATASET_NAME=CC-Dataset
     env: IM2REC=/home/ec2-user/anaconda3/envs/mxnet_p36/lib/python3.6/site-packages/mxnet/tools/im2rec.py
```

```
[8]: # The script below Pulls our images from S3 bucket

     !aws s3 sync s3://$S3_BUCKET_NAME/$DATASET_NAME $BASE_DIRECTORY/$DATASET_NAME --quiet

     print('Images have been Pulled!!! ')

     Images have been Pulled!!!
```

Continue…

- Transform the fetched images into RecordIO file, we have kept the training ratio to 70%, while Testing ratio to 30%. The RecordIO files will be created in this step with the above ratio.
- Upload created RecordIO files back into the S3 bucket, which then be used as an input for training of our model.

```
%%bash
# Now here we use the IM2REC script to convert our images which we fetched from S3 bucket into RecordIO files

# Delete if there are already created Recio files in our working directory

cd $BASE_DIR
rm *.rec
rm *.lst

# We want to create 2 LST files first, One for training and One for testing, along with saving the class of each image
# The output of the LST files command includes a list of all of our label classes
# We are specifying here the training and testing ration, 70% Training Ratio and 30% Testing Ratio

echo "Creating LST files"
python $IM2REC --list --recursive --pass-through --test-ratio=0.3 --train-ratio=0.7 $DATASET_NAME $DATASET_NAME > ${DATASET_NAME}_classes

echo "Label classes:"
cat ${DATASET_NAME}_classes

# Then we create RecordIO files from the LST files
echo "Creating RecordIO files"
python $IM2REC --num-thread=4 ${DATASET_NAME}_train.lst $DATASET_NAME
python $IM2REC --num-thread=4 ${DATASET_NAME}_test.lst $DATASET_NAME
ls -lh *.rec
```

```
Creating LST files
Label classes:
EURO-10 0
EURO-20 1
EURO-5 2
EURO-50 3
Creating RecordIO files
Creating .rec file from /tmp/CC-Dataset_train.lst in /tmp
time: 0.39785265922546387  count: 0
Creating .rec file from /tmp/CC-Dataset_test.lst in /tmp
```

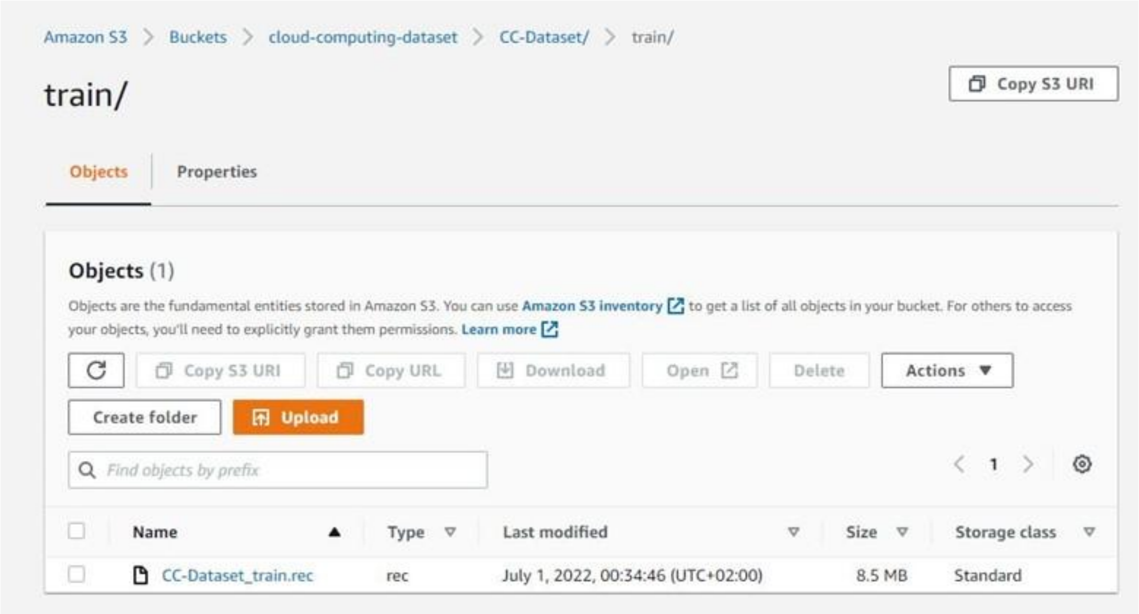
[15]:

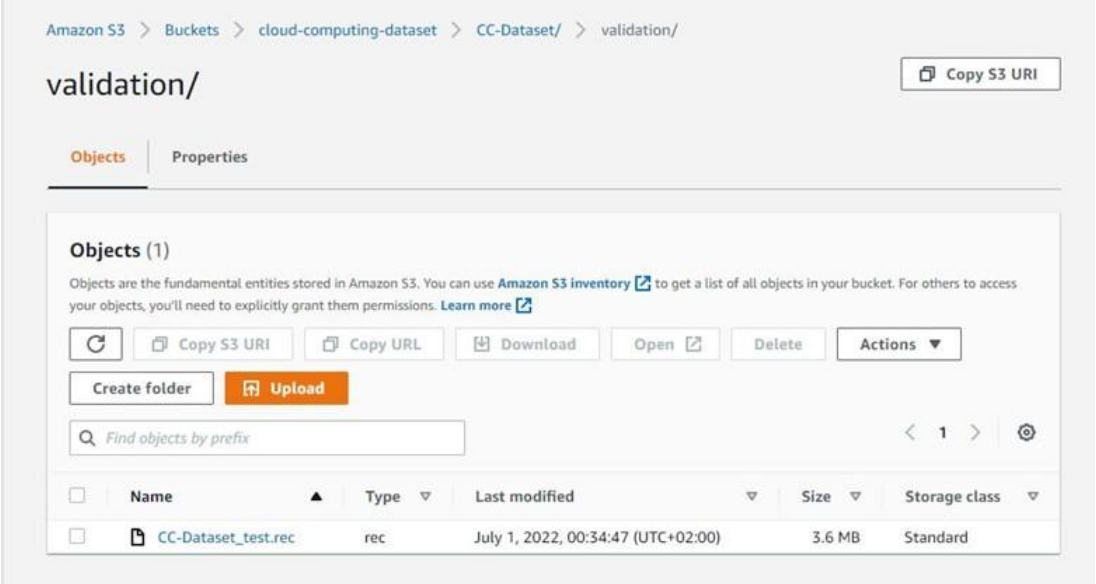
```
# We are now Uploading our train and test RecordIO files to S3 bucket

bucket = bucket_name

print (bucket)

training_path_s3 = 's3://{}/{}/train/'.format(bucket, dataset_name)
validation_path_S3 = 's3://{}/{}/validation/'.format(bucket, dataset_name)
print(training_path_s3)
print(validation_path_S3)

# Delete any existing data
!aws s3 rm s3://{bucket}/{dataset_name}/train --recursive
!aws s3 rm s3://{bucket}/{dataset_name}/validation --recursive

# Upload the rec files to the train and validation folders
!aws s3 cp /tmp/{dataset_name}_train.rec $training_path_s3
!aws s3 cp /tmp/{dataset_name}_test.rec $validation_path_S3
```

```
cloud-computing-dataset
s3://cloud-computing-dataset/CC-Dataset/train/
s3://cloud-computing-dataset/CC-Dataset/validation/
upload: ../../../tmp/CC-Dataset_train.rec to s3://cloud-computing-dataset/CC-Dataset/train/CC-Dataset_train.rec
upload: ../../../tmp/CC-Dataset_test.rec to s3://cloud-computing-dataset/CC-Dataset/validation/CC-Dataset_test.rec
```

Model looks like..

- The uploaded RecordIO files in our S3 bucket will look like this.
- We have now done our preprocessing; the data is ready to be trained. Now are going towards the process of training our model using the created RecordIO files.





Train your model

- Define the RecordIO paths to the training and validation functions by using inputs.TrainingInput() method.
- Define the output location of the model and initialize the estimator function in parallel.

```
[20]:
# Documentation of the function sagemaker.inputs.TrainingInput is available here
# https://sagemaker.readthedocs.io/en/stable/api/utility/inputs.html#sagemaker.inputs.TrainingInput
# Create a definition for input data used by an SageMaker training job.

train_data = sagemaker.inputs.TrainingInput(
    training_path_s3,
    distribution='FullyReplicated',
    content_type='application/x-recordio',
    s3_data_type='S3Prefix'
)

validation_data = sagemaker.inputs.TrainingInput(
    validation_path_S3,
    distribution='FullyReplicated',
    content_type='application/x-recordio',
    s3_data_type='S3Prefix'
)

data_channels = {'train': train_data, 'validation': validation_data}

print(train_data)
print(validation_data)

<sagemaker.inputs.TrainingInput object at 0x7f813d49e2e8>
<sagemaker.inputs.TrainingInput object at 0x7f813d49e320>
```

```
[43]:
# The are defining the output location for model

s3_output_location = 's3://{}/{}/output'.format(bucket, dataset_name)

# we have used ml.p3.2xlarge isntance for traning
image_classifier = sagemaker.estimator.Estimator(
    role=role,
    image_uri=image_uris,
    instance_count=1,
    instance_type='ml.p3.2xlarge',
    output_path=s3_output_location,
    sagemaker_session=session
)
print('done')

done
```

Continue…

- Image classification Hyperparameters,
  - **image shape** as 3,244,244 which is same as the image shape of our RecordIO files.
  - **number of classes** which in our case are 4,
  - **Augmentation type** here is important as we are taking the color into account, so we have chosen ‘crop color’.
- The training job is started and will provide the path of the model where it will be stored.

```
[58]: num_classes=! ls -l {base_dir}/{dataset_name} | wc -l
num_classes=int(num_classes[0]) - 3
num_training_samples=! cat {base_dir}/{dataset_name}_train.lst | wc -l
num_training_samples = int(num_training_samples[0])

# Details on Sagemaker built-in Image Classifier hyperparameters
# available here: https://docs.aws.amazon.com/sagemaker/latest/dg/IC-Hyperparameter.html

base_hyperparameters=dict(
    use_pretrained_model=1,
    image_shape='3,224,224',
    num_classes=num_classes,
    augmentation_type='crop_color', #taking corresponding Hue-Saturation-Lightness into account
    num_training_samples=num_training_samples,
)

# These are hyperparameters are important which can affect the model training success:
hyperparameters={
    **base_hyperparameters,
    **dict(
        learning_rate=0.001,
        mini_batch_size=5,
    )
}

image_classifier.set_hyperparameters(**hyperparameters)

hyperparameters
print('No of tranining Samples: '+str(num_training_samples))
print ('No of Classes: '+str(num_classes))

No of tranining Samples: 464
No of Classes: 4
```

```
[*]: %%time

import time
now = str(int(time.time()))
training_job_name = 'IC-' + dataset_name.replace('_', '-') + '-' + now

image_classifier.fit(inputs=data_channels, job_name=training_job_name, logs=True)

job = image_classifier.latest_training_job
model_path = f"{BASE_DIR}/{job.name}"

print(f"\n\n Finished training! The model is available for download at: {image_classifier.output_path}/{job.name}/output/model.tar.gz")

2022-06-30 23:08:48 Starting - Starting the training job...ProfilerReport-1656630528: InProgress
...
2022-06-30 23:09:40 Starting - Preparing the instances for training......
2022-06-30 23:10:43 Downloading - Downloading input data..
```

Continue…

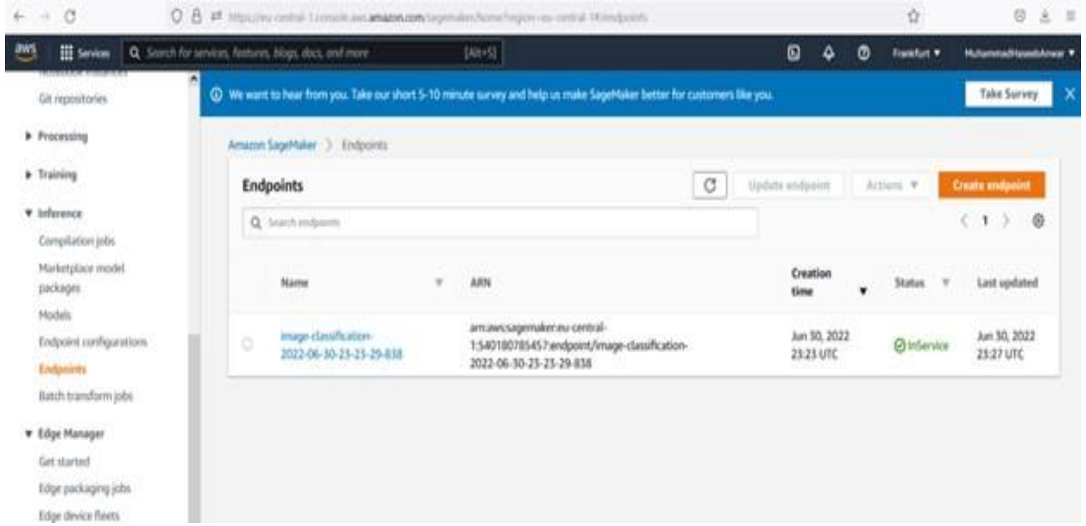
```
2022-06-30 23:13:04 Training - Training image download completed. Training in progress.[23:13:11] /opt/brazil-pkg-cache/packages/AIAlgorithm
sMXNet/AIAlgorithmsMXNet-1.3.x_ecl_Cuda_10.1.x.11282.0/AL2_x86_64/generic-flavor/src/src/operator/nn/./cudnn/./cudnn_algoreg-inl.h:97: Runni
ng performance tests to find the best convolution algorithm, this can take a while... (setting env variable MXNET_CUDNN_AUTOTUNE_DEFAULT to
0 to disable)
[06/30/2022 23:13:15 INFO 140493106341696] Epoch[0] Batch [20]#011Speed: 23.524 samples/sec#011accuracy=0.428571
[06/30/2022 23:13:16 INFO 140493106341696] Epoch[0] Batch [40]#011Speed: 32.955 samples/sec#011accuracy=0.609756
[06/30/2022 23:13:18 INFO 140493106341696] Epoch[0] Batch [60]#011Speed: 38.072 samples/sec#011accuracy=0.708197
[06/30/2022 23:13:20 INFO 140493106341696] Epoch[0] Batch [80]#011Speed: 41.102 samples/sec#011accuracy=0.767901
[06/30/2022 23:13:21 INFO 140493106341696] Epoch[0] Train-accuracy=0.784783
[06/30/2022 23:13:21 INFO 140493106341696] Epoch[0] Time cost=10.725
[06/30/2022 23:13:22 INFO 140493106341696] Epoch[0] Validation-accuracy=0.995000
[06/30/2022 23:13:23 INFO 140493106341696] Storing the best model with validation accuracy: 0.995000
[06/30/2022 23:13:23 INFO 140493106341696] Saved checkpoint to "/opt/ml/model/image-classification-0001.params"
[06/30/2022 23:13:25 INFO 140493106341696] Epoch[1] Batch [20]#011Speed: 55.178 samples/sec#011accuracy=0.895238
[06/30/2022 23:13:27 INFO 140493106341696] Epoch[1] Batch [40]#011Speed: 55.481 samples/sec#011accuracy=0.902439
[06/30/2022 23:13:28 INFO 140493106341696] Epoch[1] Batch [60]#011Speed: 55.671 samples/sec#011accuracy=0.927869
[06/30/2022 23:13:30 INFO 140493106341696] Epoch[1] Batch [80]#011Speed: 55.449 samples/sec#011accuracy=0.938272
[06/30/2022 23:13:31 INFO 140493106341696] Epoch[1] Train-accuracy=0.936957
[06/30/2022 23:13:31 INFO 140493106341696] Epoch[1] Time cost=8.199
[06/30/2022 23:13:32 INFO 140493106341696] Epoch[1] Validation-accuracy=1.000000
[06/30/2022 23:13:33 INFO 140493106341696] Storing the best model with validation accuracy: 1.000000
[06/30/2022 23:13:33 INFO 140493106341696] Saved checkpoint to "/opt/ml/model/image-classification-0002.params"
[06/30/2022 23:13:35 INFO 140493106341696] Epoch[2] Batch [20]#011Speed: 54.774 samples/sec#011accuracy=0.914286
[06/30/2022 23:13:37 INFO 140493106341696] Epoch[2] Batch [40]#011Speed: 55.316 samples/sec#011accuracy=0.951220
[06/30/2022 23:13:39 INFO 140493106341696] Epoch[2] Batch [60]#011Speed: 55.483 samples/sec#011accuracy=0.963934
[06/30/2022 23:13:40 INFO 140493106341696] Epoch[2] Batch [80]#011Speed: 55.151 samples/sec#011accuracy=0.967901
[06/30/2022 23:13:41 INFO 140493106341696] Epoch[2] Train-accuracy=0.971739
[06/30/2022 23:13:41 INFO 140493106341696] Epoch[2] Time cost=8.231
[06/30/2022 23:13:42 INFO 140493106341696] Epoch[2] Validation-accuracy=1.000000
[06/30/2022 23:13:45 INFO 140493106341696] Epoch[3] Batch [20]#011Speed: 53.828 samples/sec#011accuracy=0.990476
[06/30/2022 23:13:47 INFO 140493106341696] Epoch[3] Batch [40]#011Speed: 55.007 samples/sec#011accuracy=0.990244
[06/30/2022 23:13:48 INFO 140493106341696] Epoch[3] Batch [60]#011Speed: 55.255 samples/sec#011accuracy=0.977049
[06/30/2022 23:13:50 INFO 140493106341696] Epoch[3] Batch [80]#011Speed: 55.150 samples/sec#011accuracy=0.980247
[06/30/2022 23:13:51 INFO 140493106341696] Epoch[3] Train-accuracy=0.976087
[06/30/2022 23:13:51 INFO 140493106341696] Epoch[3] Time cost=8.235
```

```python
print(f"\n\n Finished training! The model is available for download at: {image_classifier.output_path}/{job.name}/output/model.tar.gz")
```

```
 Finished training! The model is available for download at: s3://cloud-computing-dataset/CC-Dataset/output/IC-CC-Dataset-1656630528/output/m
odel.tar.gz
```

Deploy your model

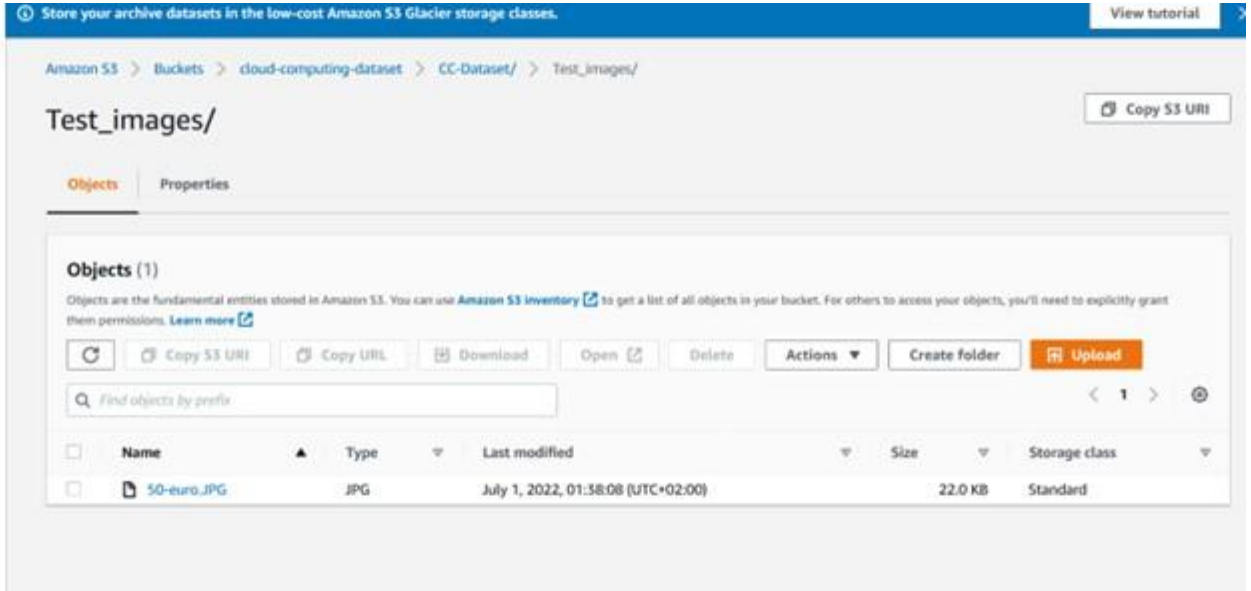
- Deployed endpoint is available and in service now.
- Create a function which invokes the endpoint and returns the result of prediction.
- Upload a sample image into S3 Bucket, We have used a 50 euro image and uploaded in into the folder test_images on S3.
- we are getting our image from S3 and saving them into a variable called Euro50.
- We are calling our classify_deployed function to predict our image, and the result is shown in below image.




```
[66]: # If we want to check out model' prediction through this notebook instance
# we will create a function which will call our endpoint here and return the model prediction
# we will have to upload some test images to our s3 bucket which this method will use

import json
import numpy as np
import os

def classify_deployed(file_name, classes):
    payload = None
    with open(file_name, 'rb') as f:
        payload = f.read()
        payload = bytearray(payload)

    result = deployed_endpoint.predict(payload, initial_args={'ContentType': 'image/jpeg'})

    #result = json.loads(deployed_endpoint.predict(payload))
    #result = deployed_endpoint.predict(payload)
    #best_prob_index = np.argmax(result)
    #return (classes[best_prob_index], result[best_prob_index])

    resultarray = (result.decode('UTF-8')[1:len(result)-1]).split(",")

    for i in range(len(classes)):
        print(classes[i] + ":" + str(resultarray[i]))
    return result


print("Function created")

Function created
```

Continue…

- The **EURO-50** image has the highest prediction value, our model is **99.9% confident** that the provided image in of 50 Euro note.



[68]:
```
# but for this we have to make our image public

!wget -O test.jpg https://cloud-computing-dataset.s3.eu-central-1.amazonaws.com/CC-Dataset/Test_images/50-euro.JPG
Euro50 = "test.jpg"
# test image
from IPython.display import Image

Image(Euro50)
```

```
--2022-06-30 23:39:43--  https://cloud-computing-dataset.s3.eu-central-1.amazonaws.com/CC-Dataset/Test_images/50-euro.JPG
Resolving cloud-computing-dataset.s3.eu-central-1.amazonaws.com (cloud-computing-dataset.s3.eu-central-1.amazonaws.com)... 52.219.47.144
Connecting to cloud-computing-dataset.s3.eu-central-1.amazonaws.com (cloud-computing-dataset.s3.eu-central-1.amazonaws.com)|52.219.47.144|:4
43... connected.
HTTP request sent, awaiting response... 200 OK
Length: 22513 (22K) [image/jpeg]
Saving to: ‘test.jpg’

test.jpg            100%[===================>]  21.99K  --.-KB/s    in 0.001s

2022-06-30 23:39:44 (21.8 MB/s) - ‘test.jpg’ saved [22513/22513]
```

[68]:


```
43... connected.
HTTP request sent, awaiting response... 200 OK
Length: 22513 (22K) [image/jpeg]
Saving to: ‘test.jpg’

test.jpg            100%[===================>]  21.99K  --.-KB/s    in 0.001s

2022-06-30 23:39:44 (21.8 MB/s) - ‘test.jpg’ saved [22513/22513]
```

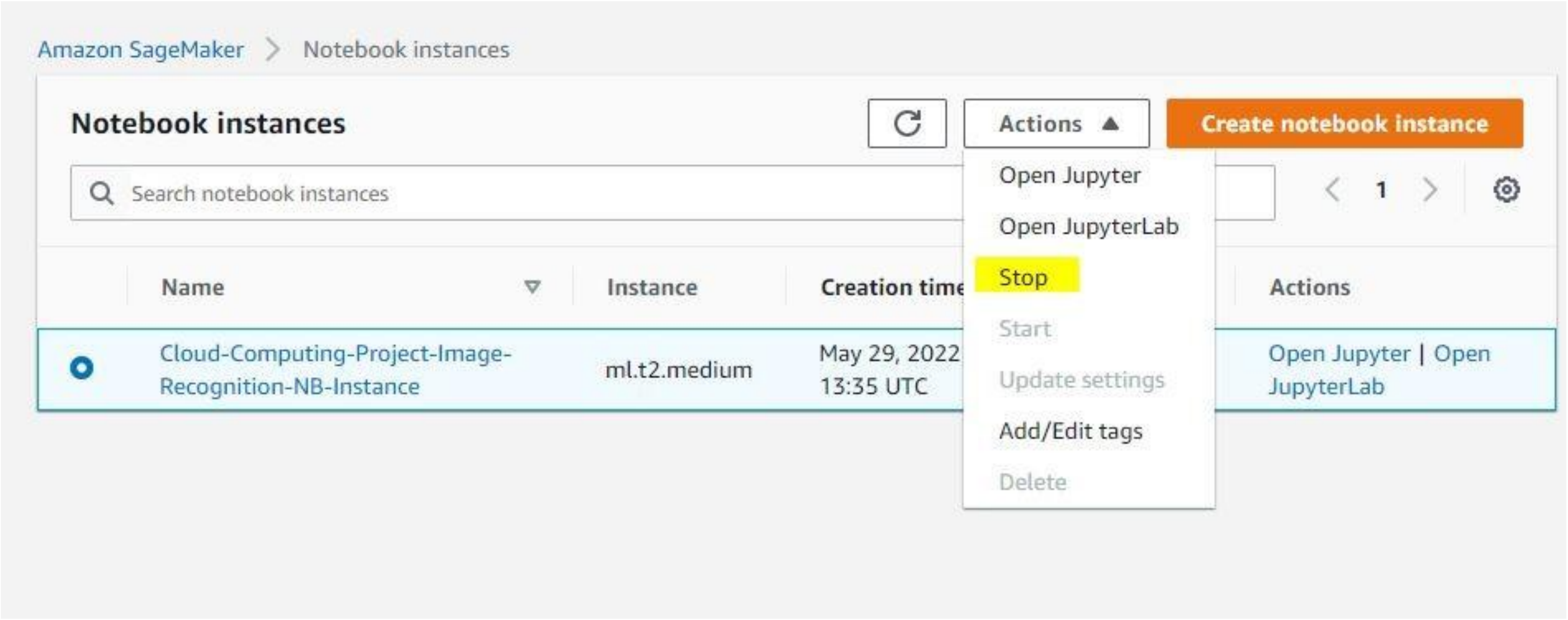
[68]:

[70]:
```
object_categories = [
    "EURO-10",
    "EURO-5",
    "EURO-20",
    "EURO-50"
]

output = classify_deployed(Euro50,object_categories)
```

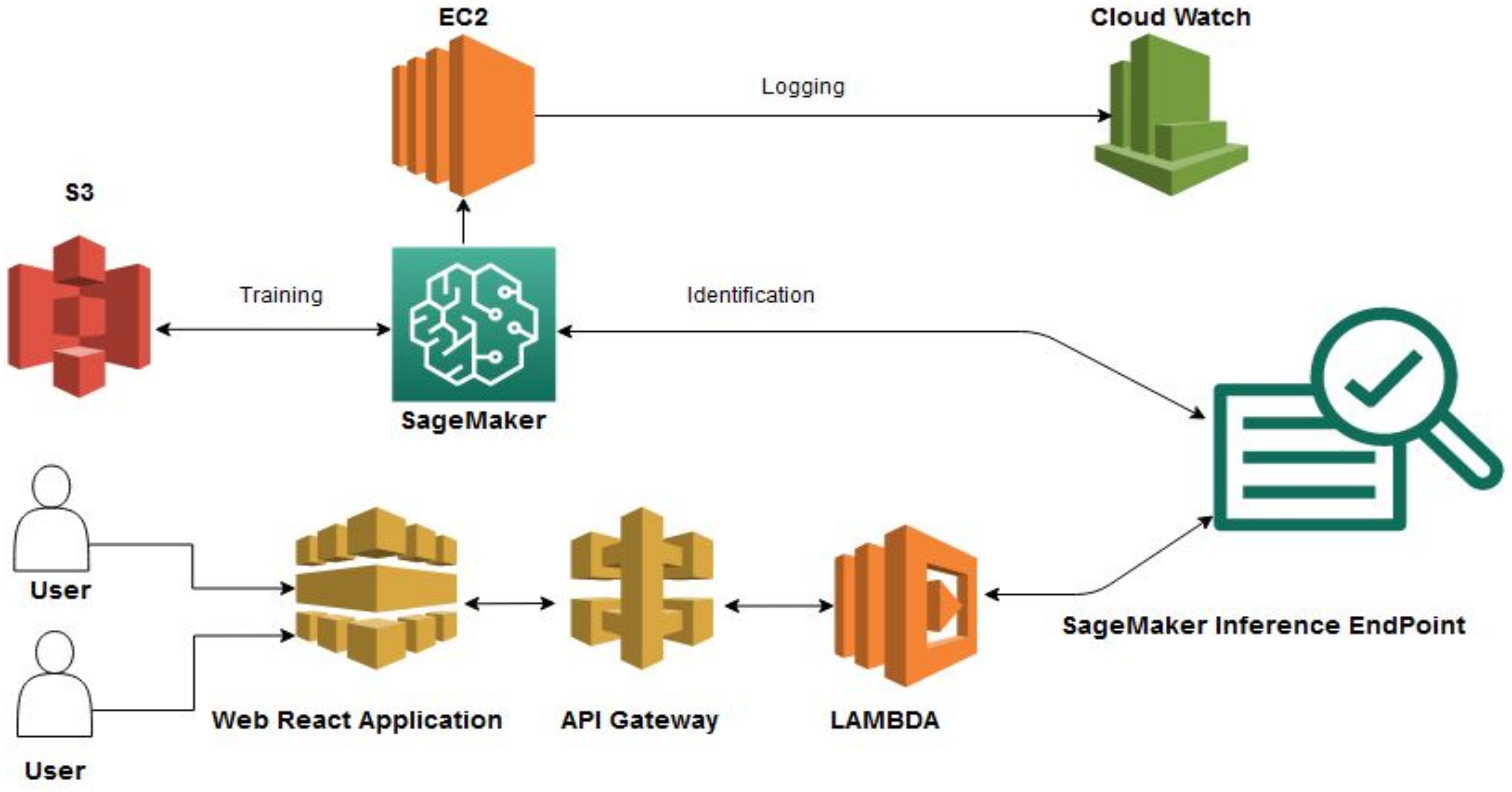
```
EURO-10:2.640426464495249e-05
EURO-5: 4.2157054849667475e-05
EURO-20: 9.781115659279749e-05
EURO-50: 0.9998335838317871
```

Alert…

- **Important:** Important is to stop the notebook after you have done creating the model, otherwise AWS will keep charging you for the time the notebook is in service.



Flow Diagram



Web Application

The web application is created for demo purposes of the model deployed on AWS Sagemaker. It is created using React libraries in addition to using amplify library which streamlines the connection of the web application with the AWS cloud setup. For setting up amplify in the project we need to install amplify and with in the project directory execute the command: amplify init Now we need to setup an API endpoint on the AWS API Gateway. We use the command: amplify add api We further follow the steps in the process executed by the command to make a POST Rest api. When the api end point is created locally we push the setup on the AWS using: amplify push The web application consists of 2 main components: Image Capture: This component uses the camera of the device to capture the image to be identified

ImageCapture.js U ×

src > components > ImageCapture.js > ImageCapture > render

```
    // Moeez comments: This component opens and captures or screenshots
    // the image in the camera session which is then used for image recognition
    return (
      <div>
        <div>
          <Webcam
            audio={false}
            height={IMAGE_HEIGHT}
            width={IMAGE_WIDTH}
            ref={this.setRef}
            screenshotFormat="image/jpeg"
            screenshotWidth={IMAGE_WIDTH}
            videoConstraints={videoConstraints}
          />
        </div>

        <Form.Button onClick={this.handleCapture}>Classify</Form.Button>
      </div>
    );
  }
}
```

ClassifiedImage.js U ×

src > components > ClassifiedImage.js > ...

```
  // Moeez comments: This component creates a React UI Card which is consists of a
  // Header, Meta and a Description element.
  render() {
    return (
      <Card style={{width: '224px'}}>
        <Image src={this.props.imageSrc} />
        <Card.Content>
          <Card.Header>
            { this.state.bestLabel ? this.state.bestLabel : "Loading..." }
          </Card.Header>
          <Card.Meta>
            { this.state.bestLabelScore ? this.state.bestLabelScore : "" }
          </Card.Meta>
          <Card.Description>
            <Accordion defaultActiveIndex={-1} panels={this.accordionPanels()} />
          </Card.Description>
        </Card.Content>
      </Card>
    )
  }
}
```

Continue…

App.js M ×

src > App.js > App > render

```
{/* Moeez comments: This is form to take inputs which are used to send the
post request to and the labels are used to present the result of specific
categories. */}
<Form>
  <Form.Group widths='equal'>
    <Form.Input label='SM Endpoint Name' placeholder='Please enter Sagemaker endpoint name'
    name='endpointName' onChange={this.changeHandler} value={this.state.endpointName} />

    <Form.Input label='SM Endpoint Region' placeholder='Please enter sagemaker endpoint region'
    name='endpointRegion' onChange={this.changeHandler} value={this.state.endpointRegion} />

    <Form.Input label='Labels' placeholder='Please enter space delimited list of labels'
    name='labels' onChange={this.changeHandler} value={this.state.labels} />
  </Form.Group>

  <Form.Group widths='equal'>
    <ImageCapture onCapture={this.classify}/>
  </Form.Group>
</Form>
```

App.js M ×

src > App.js > App > render

```

  {/* Moeez comments: Finally the results are grouped together and displayed */}
<CardGroup>
  { this.state.imageSources.map((src, index) =>
  <ClassifiedImage key={"img"+index} imageSrc={src} classifier={this.classifier} />) }
</CardGroup>
```

App.js M ×

src > App.js > App > render

```

// Moeez comments: This function calls the AWS API Gateway endpointName
// and returns the categories with their predictions as the result
classifier = async (imageSrc) => {
  const base64Image = new Buffer(imageSrc.replace(/^data:image\/\w+;base64,/, ""), 'base64')
  const { predictions } = await API.post(
    aws_exports.aws_cloud_logic_custom[0].name,
    '/classify',
    {
      body: {
        base64Image,
        endpointName: this.state.endpointName,
        endpointRegion: this.state.endpointRegion,
      },
    }
  );
  const topProbIndex = argMax(predictions);
  const labels = [].concat(this.state.labels.split(' '));
  labels.sort();
  return {
    labels: labels, predictions, topProbIndex: topProbIndex
  }
}
```

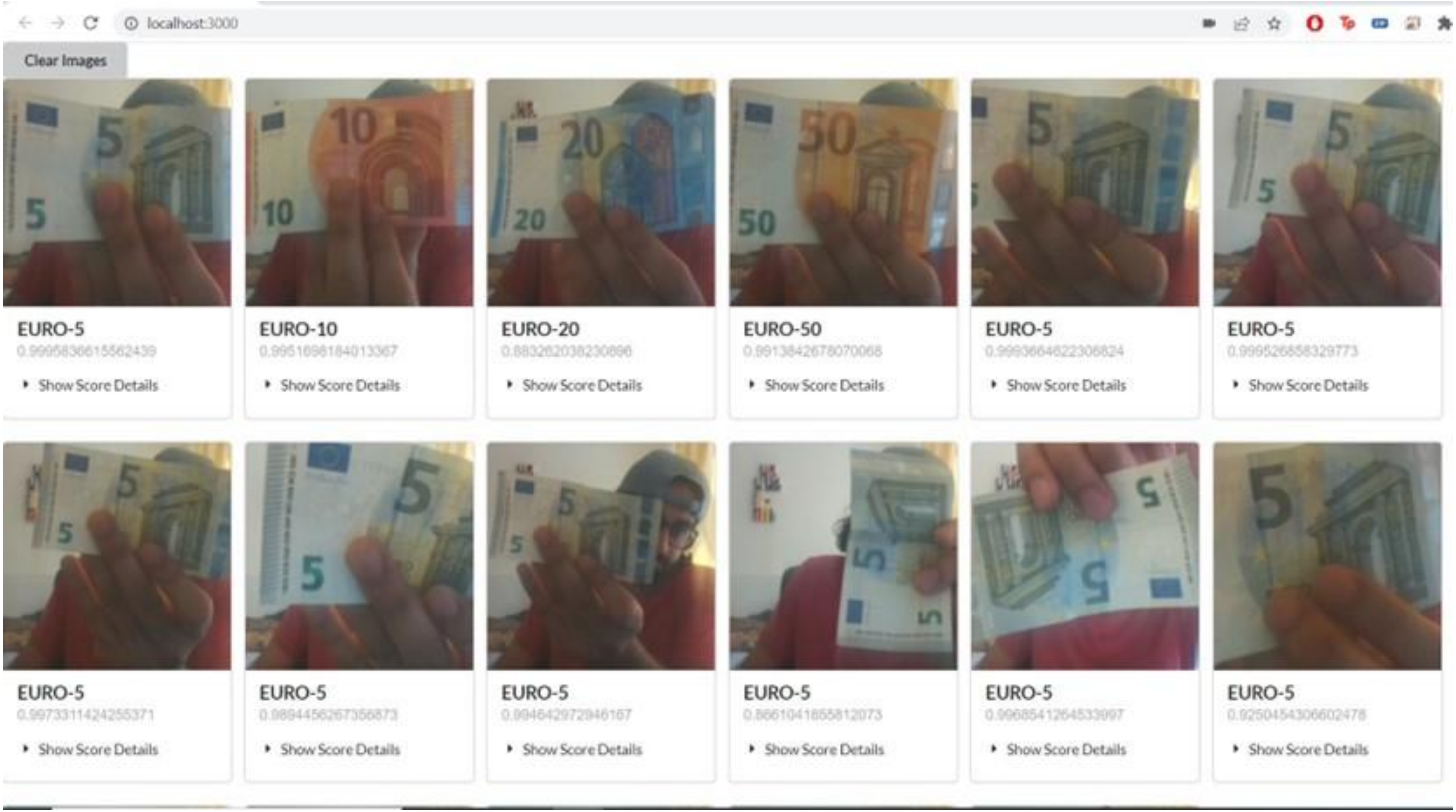
Continue…

App.js M ×

src > App.js > App > render

```
// Moeez comments: This function calls the AWS API Gateway endpointName
// and returns the categories with their predictions as the result
classifier = async (imageSrc) => {
  const base64Image = new Buffer(imageSrc.replace(/^data:image\/\w+;base64,/, ""), 'base64')
  const { predictions } = await API.post(
    aws_exports.aws_cloud_logic_custom[0].name,
    '/classify',
    {
      body: {
        base64Image,
        endpointName: this.state.endpointName,
        endpointRegion: this.state.endpointRegion,
      },
    }
  );
  const topProbIndex = argMax(predictions);
  const labels = [].concat(this.state.labels.split(' '));
  labels.sort();
  return {
    labels: labels, predictions, topProbIndex: topProbIndex
  }
}
```


HURRAH!


localhost:3000
Clear Images
EURO-5
Show Score Details
EURO-10
Show Score Details
EURO-20
Show Score Details
EURO-50
Show Score Details
EURO-5
Show Score Details
EURO-5
Show Score Details
EURO-5
Show Score Details
EURO-5
Show Score Details
EURO-5
Show Score Details
EURO-5
Show Score Details
EURO-5
Show Score Details
EURO-5
Show Score Details


QUESTIONS?
COMMENTS?
CONCERNS?

Thank you😊