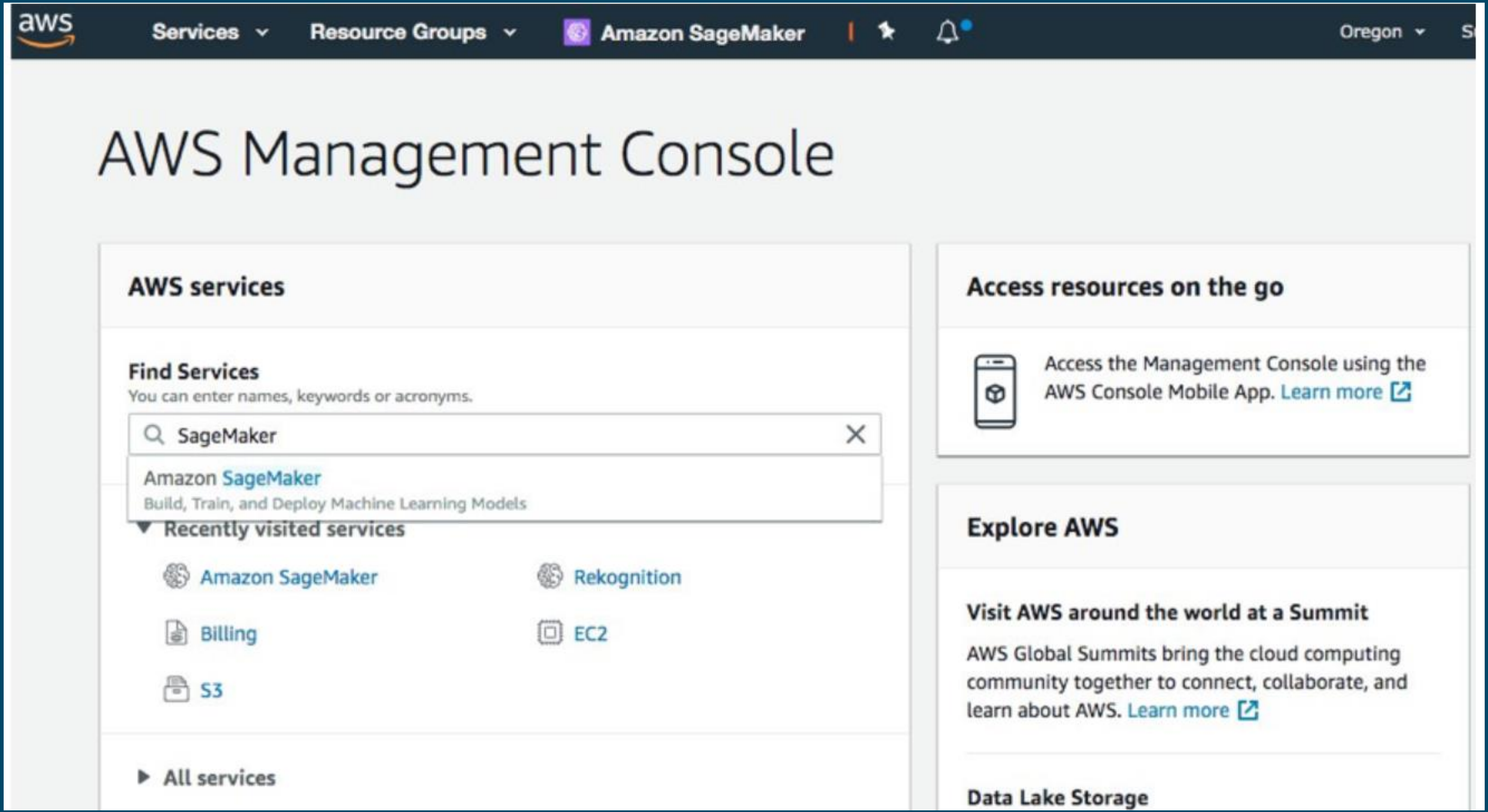
AWS SageMaker

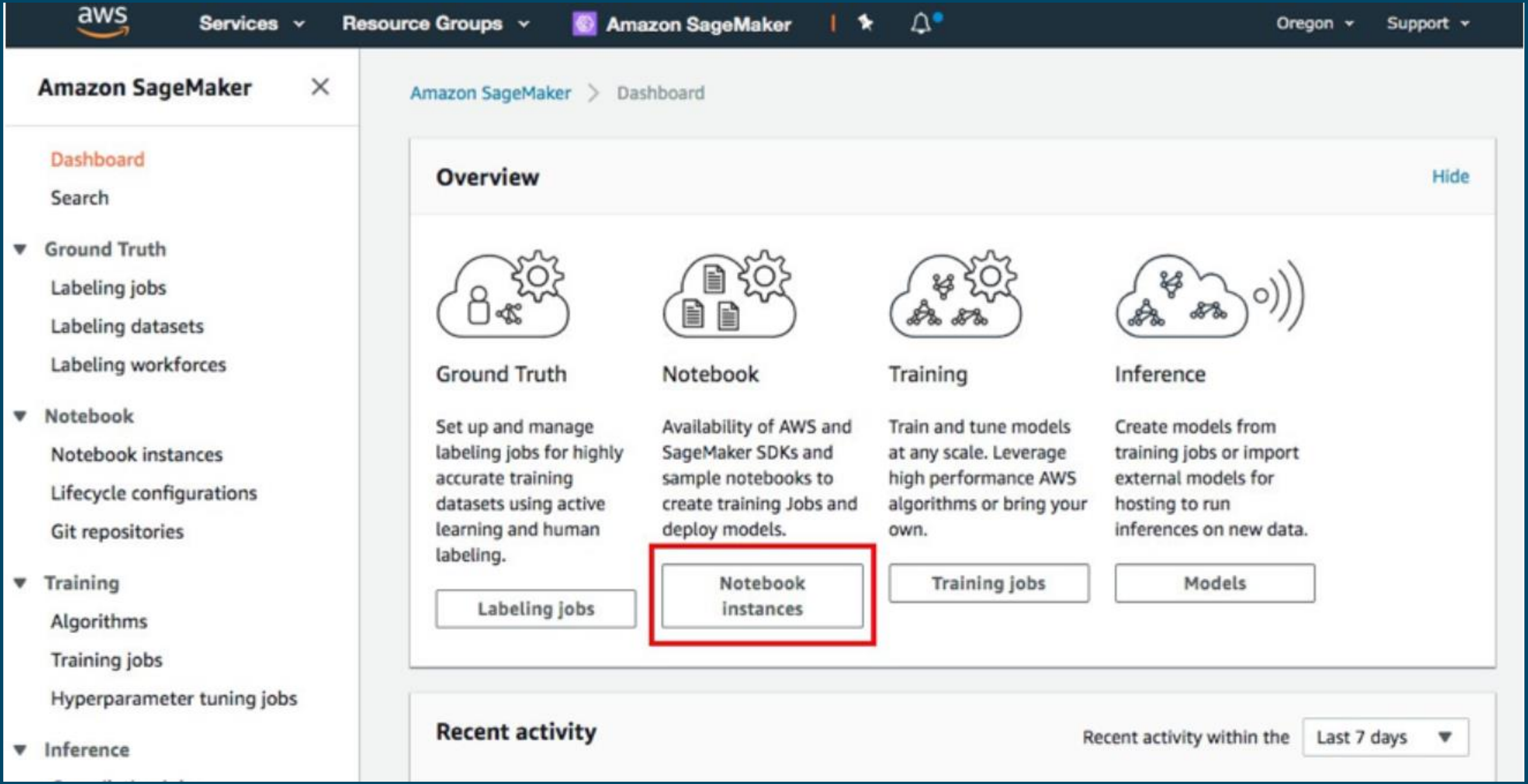
https://aws.amazon.com/getting-started/tutorials/build-train-deploy-machine-learning-model-sagemaker/

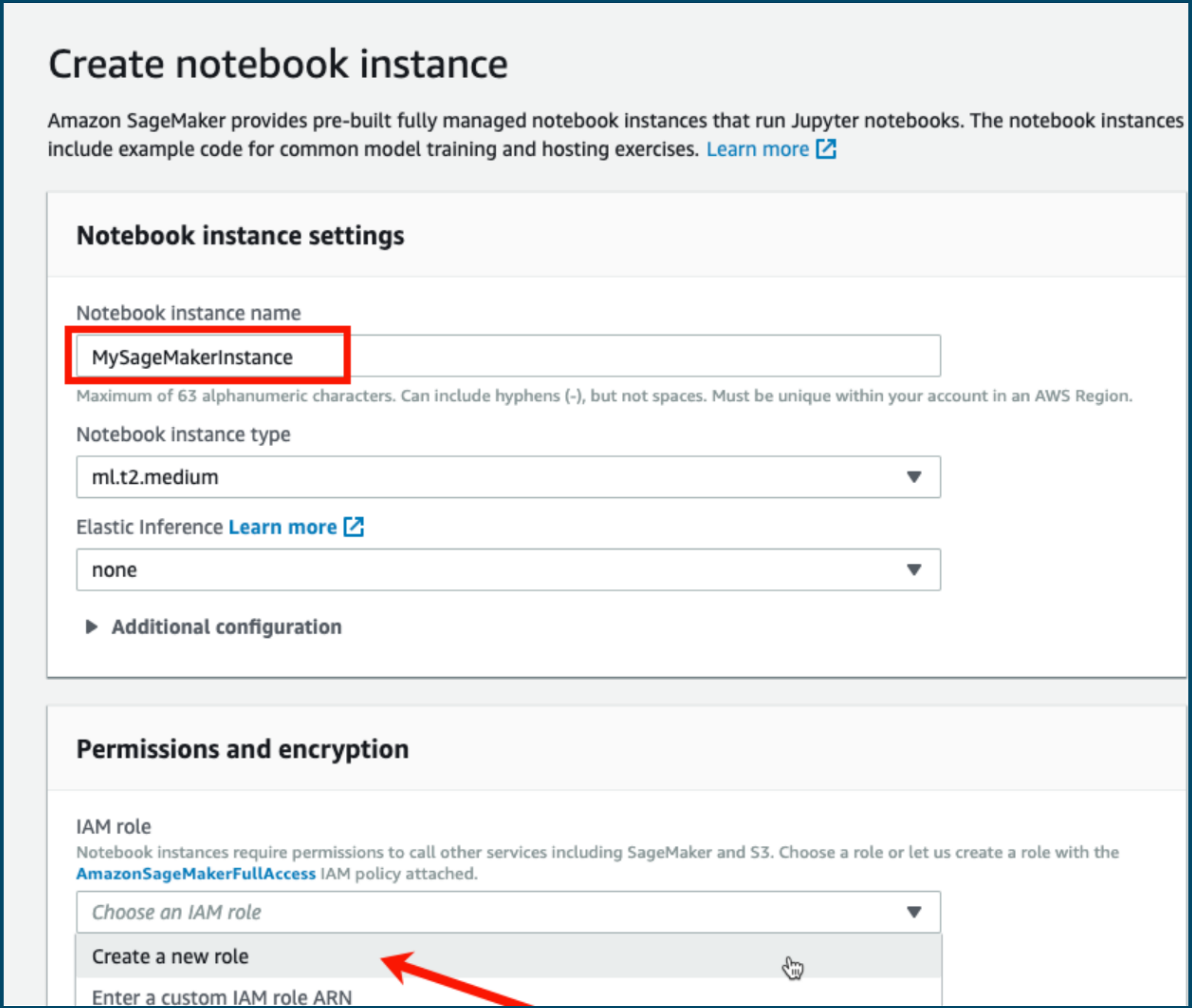
Create & deploy a model

- Create s3 file folder for data
- Create an XGBoost container
  - Which contains the XGBoost code, in a format known by the AWS API
- Train the model
- Deploy the model, in a new container
- Send the deployed model data
- → All model and containers managed by AWS API



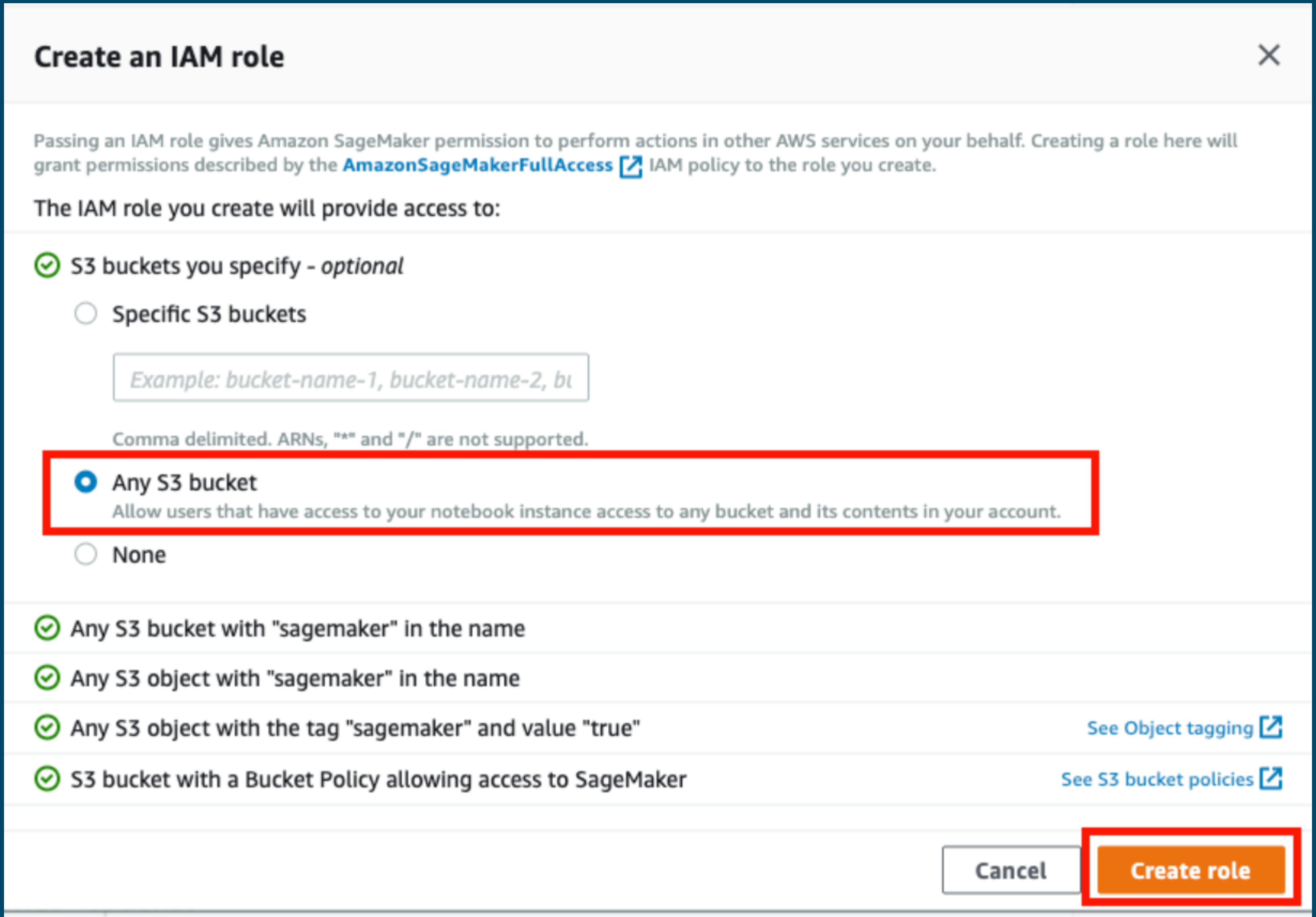
https://aws.amazon.com/console/


aws
Services
Resource Groups
Amazon SageMaker
Oregon
Support
Amazon SageMaker
Dashboard
Search
Ground Truth
Labeling jobs
Labeling datasets
Labeling workforces
Notebook
Notebook instances
Lifecycle configurations
Git repositories
Training
Algorithms
Training jobs
Hyperparameter tuning jobs
Inference
Amazon SageMaker > Dashboard
Overview
Hide
Ground Truth
Set up and manage labeling jobs for highly accurate training datasets using active learning and human labeling.
Labeling jobs
Notebook
Availability of AWS and SageMaker SDKs and sample notebooks to create training Jobs and deploy models.
Notebook instances
Training
Train and tune models at any scale. Leverage high performance AWS algorithms or bring your own.
Training jobs
Inference
Create models from training jobs or import external models for hosting to run inferences on new data.
Models
Recent activity
Recent activity within the
Last 7 days

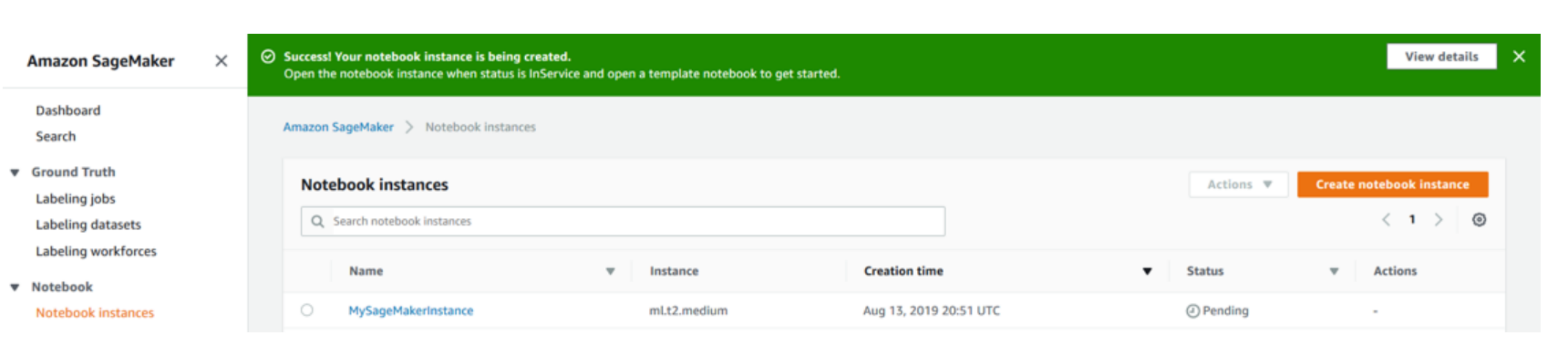


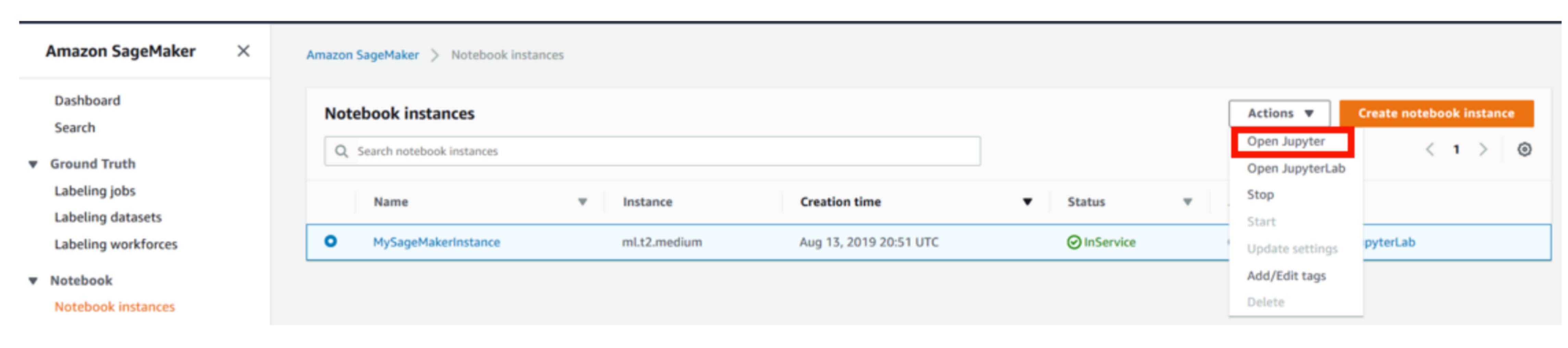
Create notebook instance
Amazon SageMaker provides pre-built fully managed notebook instances that run Jupyter notebooks. The notebook instances include example code for common model training and hosting exercises. Learn more
Notebook instance settings
Notebook instance name
MySageMakerInstance
Maximum 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
Additional configuration
Permissions and encryption
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.
Choose an IAM role
Create a new role
Enter a custom IAM role ARN

Prepare file space

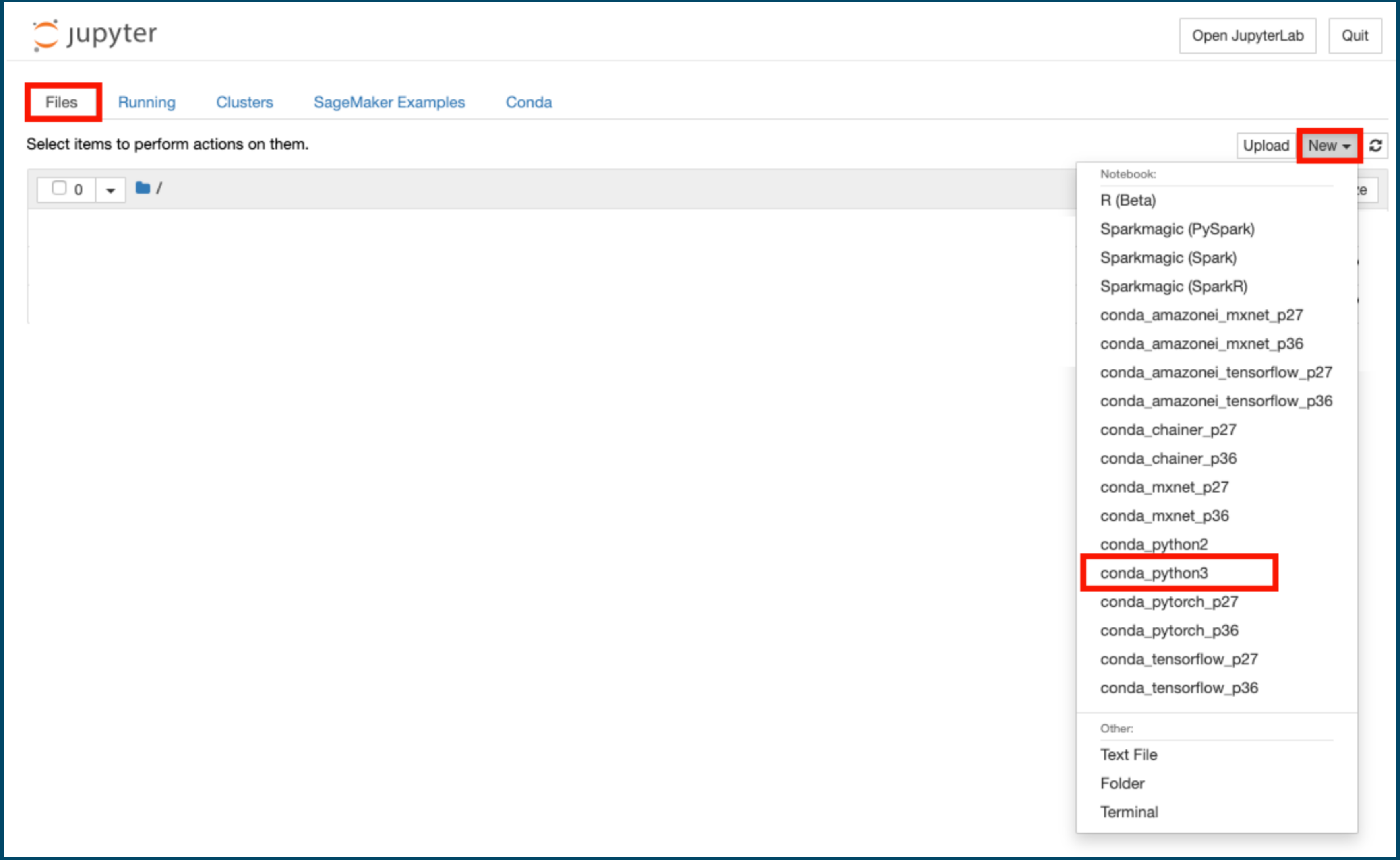


Wait for notebook instance to be ready




Amazon SageMaker
Dashboard
Search
Ground Truth
Labeling jobs
Labeling datasets
Labeling workforces
Notebook
Notebook instances
Amazon SageMaker > Notebook instances
Notebook instances
Actions
Create notebook instance
Search notebook instances
Open Jupyter
Open JupyterLab
Stop
Start
Update settings
Add/Edit tags
Delete
Name
Instance
Creation time
Status
MySageMakerInstance
ml.t2.medium
Aug 13, 2019 20:51 UTC
InService
pyterLab

Simple Python notebook



Prepare Docker XGBoost container to run

(the estimator inside the container is XGBoost)

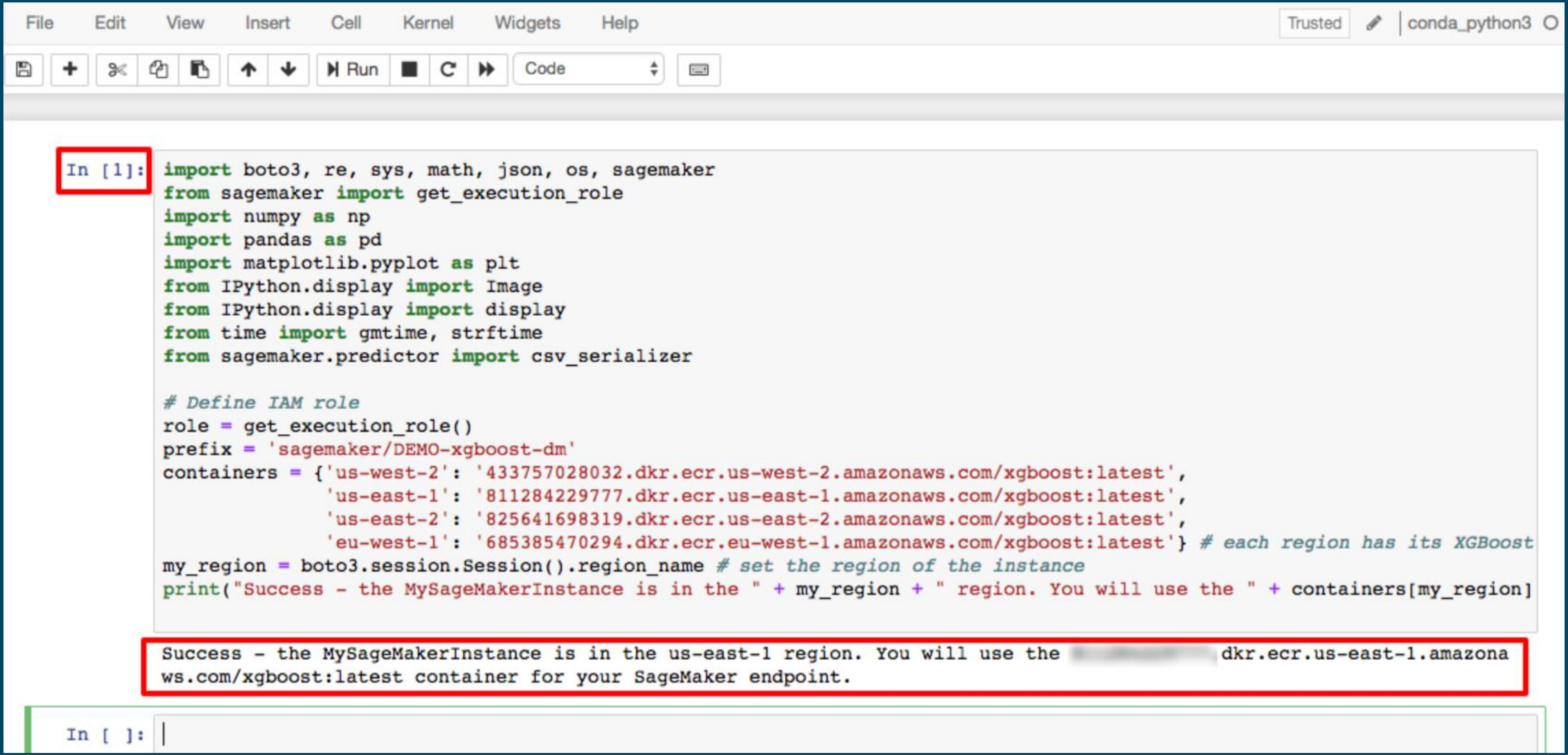
File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

In [*]:

```
import boto3, re, sys, math, json, os, sagemaker
from sagemaker import get_execution_role
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import Image
from IPython.display import display
from time import gmtime, strftime
from sagemaker.predictor import csv_serializer

# Define IAM role
role = get_execution_role()
prefix = 'sagemaker/DEMO-xgboost-dm'
containers = {'us-west-2': '433757028032.dkr.ecr.us-west-2.amazonaws.com/xgboost:latest',
              'us-east-1': '811284229777.dkr.ecr.us-east-1.amazonaws.com/xgboost:latest',
              'us-east-2': '825641698319.dkr.ecr.us-east-2.amazonaws.com/xgboost:latest',
              'eu-west-1': '685385470294.dkr.ecr.eu-west-1.amazonaws.com/xgboost:latest'} # each region has its XGBoost
my_region = boto3.session.Session().region_name # set the region of the instance
print("Success - the MySageMakerInstance is in the " + my_region + " region. You will use the " + containers[my_region]
```


File Edit View Insert Cell Kernel Widgets Help
Trusted
conda_python3
Run
Code
In [1]:
import boto3, re, sys, math, json, os, sagemaker
from sagemaker import get_execution_role
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import Image
from IPython.display import display
from time import gmtime, strftime
from sagemaker.predictor import csv_serializer
# Define IAM role
role = get_execution_role()
prefix = 'sagemaker/DEMO-xgboost-dm'
containers = {'us-west-2': '433757028032.dkr.ecr.us-west-2.amazonaws.com/xgboost:latest',
'us-east-1': '811284229777.dkr.ecr.us-east-1.amazonaws.com/xgboost:latest',
'us-east-2': '825641698319.dkr.ecr.us-east-2.amazonaws.com/xgboost:latest',
'eu-west-1': '685385470294.dkr.ecr.eu-west-1.amazonaws.com/xgboost:latest'} # each region has its XGBoost
my_region = boto3.session.Session().region_name # set the region of the instance
print("Success - the MySageMakerInstance is in the " + my_region + " region. You will use the " + containers[my_region]
Success - the MySageMakerInstance is in the us-east-1 region. You will use the dkr.ecr.us-east-1.amazonaws.com/xgboost:latest container for your SageMaker endpoint.
In [ ]:

A file folder (bucket) for processing

```
In [2]: bucket_name = 'testyourname' # <--- change this variable to a unique name for your bucket
        s3 = boto3.resource('s3')
        try:
            if  my_region == 'us-east-1':
              s3.create_bucket(Bucket=bucket_name)
            else:
              s3.create_bucket(Bucket=bucket_name, CreateBucketConfiguration={ 'LocationConstraint': my_region })
            print('S3 bucket created successfully')
        except Exception as e:
            print('S3 error: ',e)

        S3 bucket created successfully
```

Data available for processing

```
In [3]: try:
          urllib.request.urlretrieve ("https://d1.awsstatic.com/tmt/build-train-deploy-machine-learning-model-sagemaker/bank_cl
          print('Success: downloaded bank_clean.csv.')
        except Exception as e:
          print('Data load error: ',e)

        try:
          model_data = pd.read_csv('./bank_clean.csv',index_col=0)
          print('Success: Data loaded into dataframe.')
        except Exception as e:
            print('Data load error: ',e)
```

```
Success: downloaded bank_clean.csv.
Success: Data loaded into dataframe.
```

Split data

In [4]:
```
train_data, test_data = np.split(model_data.sample(frac=1, random_state=1729), [int(0.7 * len(model_data))])
print(train_data.shape, test_data.shape)
```

```
(28831, 61) (12357, 61)
```

Format the input data

```
# Prep the input data file
pd.concat([train_data['y_yes'], train_data.drop(['y_no', 'y_yes'], axis=1)], 
axis=1).to_csv('train.csv', index=False, header=False)
boto3.Session().resource('s3').Bucket(bucket_name).Object(os.path.join(prefix, 
'train/train.csv')).upload_file('train.csv’)
s3_input_train = sagemaker.s3_input(s3_data='s3://{}/{}/train'.format(bucket_name, prefix), 
content_type='csv’)
```

Create a sized instance of the XGBoost container

```
# Create the instance (Docker) with the XGBost estimator
sess = sagemaker.Session()
xgb = sagemaker.estimator.Estimator(containers[my_region],role, train_instance_count=1,
train_instance_type='ml.m4.xlarge',output_path='s3://{}/{}/output'.format(bucket_name,
prefix),sagemaker_session=sess)
xgb.set_hyperparameters(max_depth=5,eta=0.2,gamma=4,min_child_weight=6,subsample=0
.8,silent=0,objective='binary:logistic',num_round=100)
```

Run the container

```
In [7]: xgb.fit({'train': s3_input_train})

[17:36:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 10 extra nodes, 14 pruned nodes, max_depth=5
[93]#011train-error:0.095314
[17:36:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 24 extra nodes, 30 pruned nodes, max_depth=5
[94]#011train-error:0.095314
[17:36:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 24 pruned nodes, max_depth=3
[95]#011train-error:0.095314
[17:36:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 12 extra nodes, 30 pruned nodes, max_depth=5
[96]#011train-error:0.095279
[17:36:25] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 18 extra nodes, 12 pruned nodes, max_depth=5
[97]#011train-error:0.094828
[17:36:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 4 extra nodes, 22 pruned nodes, max_depth=2
[98]#011train-error:0.094863
[17:36:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 30 extra nodes, 12 pruned nodes, max_depth=5
[99]#011train-error:0.094759


2019-08-15 17:36:34 Uploading - Uploading generated training model
2019-08-15 17:36:34 Completed - Training job completed
Billable seconds: 56
```

Deploy the generated model

- Note that the tools know that this is a XGBoost container
  - Thus it knows where the model is stored
  - It uses that information to create a new container with

```
In [9]: xgb_predictor = xgb.deploy(initial_instance_count=1,instance_type='ml.m4.xlarge')

INFO:sagemaker:Creating model with name: xgboost-2018-07-13-14-29-39-425
INFO:sagemaker:Creating endpoint with name xgboost-2018-07-13-14-25-03-272

---------------------------------------------------------------!
```

Send data to the deployed model

```
In [20]: test_data_array = test_data.drop(['y_no', 'y_yes'], axis=1).values #load the data into an array
         xgb_predictor.content_type = 'text/csv' # set the data type for an inference
         xgb_predictor.serializer = csv_serializer # set the serializer type
         predictions = xgb_predictor.predict(test_data_array).decode('utf-8') # predict!
         predictions_array = np.fromstring(predictions[1:], sep=',') # and turn the prediction into an array
         print(predictions_array.shape)

         (12357,)
```

Create & deploy a model

- Create s3 file folder for data
- Create an XGBoost container
  - Which contains the XGBoost code, in a format known by the AWS API
- Train the model
- Deploy the model, in a new container
- Send the deployed model data
- → All model and containers managed by AWS API