Model Migration
and
Model Registry Using Amazon SageMaker

-SUHAS BORATE

PART-I
MODEL MIGRATION

Deploy A Locally Trained ML Model In Cloud Using AWS SageMaker

Training the Model

1. In local laptop, use Jupyter notebook and train a XGBoost classification model on the Auto Insurance data set.

2. Test the model and save the model file locally using joblib.

1. Train Model in Jupyter Notebook

- To train ML model in Jupyter Notebook follow the following steps:

1. Import required libraries
2. Upload Dataset into Jupyter Notebook from local system.
3. Data Preprocessing
4. Label Encoding
5. Feature Selection
6. Feature Scaling
7. Splitting Dataset into train and test set
8. Model Training (XGBoost)
9. Make Predictions on test data
10. Check performance of model – Accuracy Score, Confusion Matrix and Classification Report

- Take first 5 records from test data set and save this file as 'test_point_fraud.csv'.
- This 'test_point_fraud.csv' file is used to make predictions using locally trained model after that model is saved into joblib file.

```
import numpy as np

point_X = x_test[0:5]

print(point_X)

#point_X = np.expand_dims(point_X, axis=0)
#print(point_X)
#point_y = test_y[0]

np.savetxt("test_point_fraud.csv", point_X, delimiter=",")
```

```
     months_as_customer  age  policy_state  policy_csl  policy_deductable  \
521                   5   26             0           1               2000
737                 160   33             0           2               1000
740                 385   51             1           1               1000
660                 446   57             1           0               2000
411                  84   29             2           1               1000
```

2. Save the trained model using Joblib

- To save locally trained model (Jupyter Notebook model) use joblib library.
- First give name to joblib file. In this case it is "DEMO-local-xgboost-model-fraud-detection" and then dump that file using `joblib.dump(model, filename). #bt is XGBoostClassifier`
- `Then load the model file using joblib.load(filename)`

```
In [35]: model_file_name = "DEMO-local-xgboost-model-fraud-detection"
         import joblib
         joblib.dump(bt, model_file_name)

Out[35]: ['DEMO-local-xgboost-model-fraud-detection']

In [38]: bt1 = joblib.load(model_file_name)
```

- Make predictions on test data ("test_point_fraud.csv"- this file contains first 5 records from actual test data set) using joblib saved model.
- Here, prediction output is : [0,0,0,0,0]
- Make sure that , predication output for locally trained model (Jupyter Notebook Model) , Migrated Model and SageMaker Model Should be same.

```
In [39]: bt1.predict(mypayload)

Out[39]: array([0, 0, 0, 0, 0])
```

Deploying a locally trained Model on AWS cloud (Model Migration)

Pre-requisite for trying out the exercise -

- One needs to have a **free tier AWS account** to avail Amazon's cloud services including SageMaker, S3 etc. and **some familiarity with launching these services on AWS console**. Details on how to create a free tier AWS account and try launching these services on AWS console, can be found easily on the net.

- **Important** — Please note that SageMaker is NOT a free service and does incur some cost based on how long you run and use the resources like notebook instances. You have to properly cleanup all SageMaker resources, S3 buckets etc. after you try this deployment exercise. I have shared the details on how to cleanup at the end of this article.

Amazon SageMaker Studio

- Amazon SageMaker Studio provides a single, web-based visual interface where you can perform all ML development steps, improving data science team productivity by up to 10x.
- SageMaker Studio gives you complete access, control, and visibility into each step required to build, train, and deploy models.
- You can quickly upload data, create new notebooks, train and tune models, move back and forth between steps to adjust experiments, compare results, and deploy models to production all in one place, making you much more productive.
- All ML development activities including notebooks, experiment management, automatic model creation, debugging, and model and data drift detection can be performed within SageMaker Studio.

To onboard to the Domain using Quick setup

1. Open the SageMaker console.
2. Choose **Control Panel** at the top left of the page.
3. On the **Setup SageMaker Domain** page, choose **Quick setup**.
4. Under **User profile**, for **Name** keep the default name or create a new name. The name can be up to 63 characters. Valid characters: A-Z, a-z, 0-9, and - (hyphen).
5. For **Default execution role**, choose an option from the role selector. This is the default role that is assigned to the **Amazon SageMaker Domain** user profile.
6. If you choose **Enter a custom IAM role ARN**, the role must have at a minimum, an attached trust policy that grants SageMaker permission to assume the role. For more information, see SageMaker Roles.
7. Choose **Submit**.
8. From the pop-up window, select a Amazon Virtual Private Cloud (Amazon VPC) and subnet to use.
9. Choose **Save and continue**.
10. When **Status** is **Ready**, the user name that you specified is enabled and chosen. The **Add user** and **Delete user** buttons, and the **Launch app** link are also enabled.

To access Studio after you onboard

01

Open

- *Open the SageMaker console.*

02

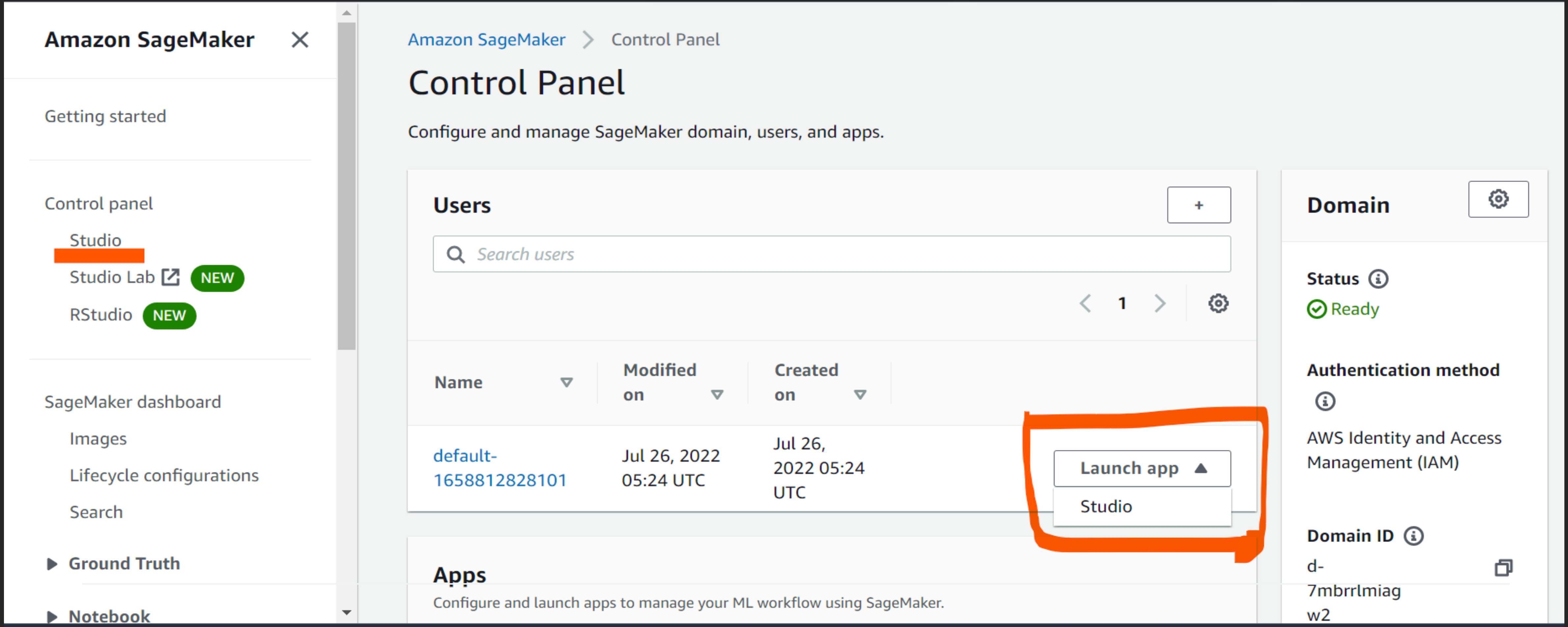
Choose

- *Choose Control Panel at the top left of the page.*

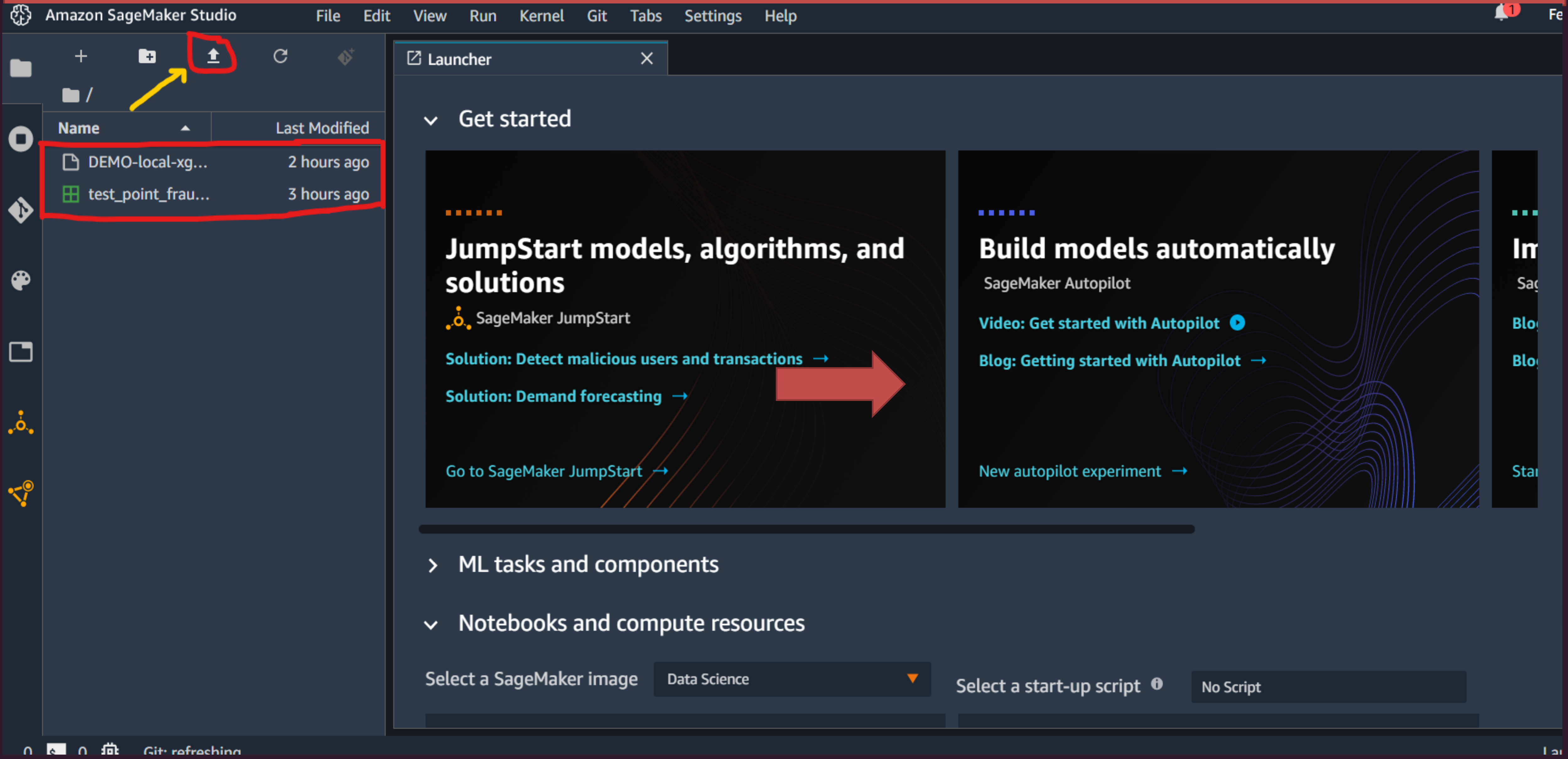
03

Choose

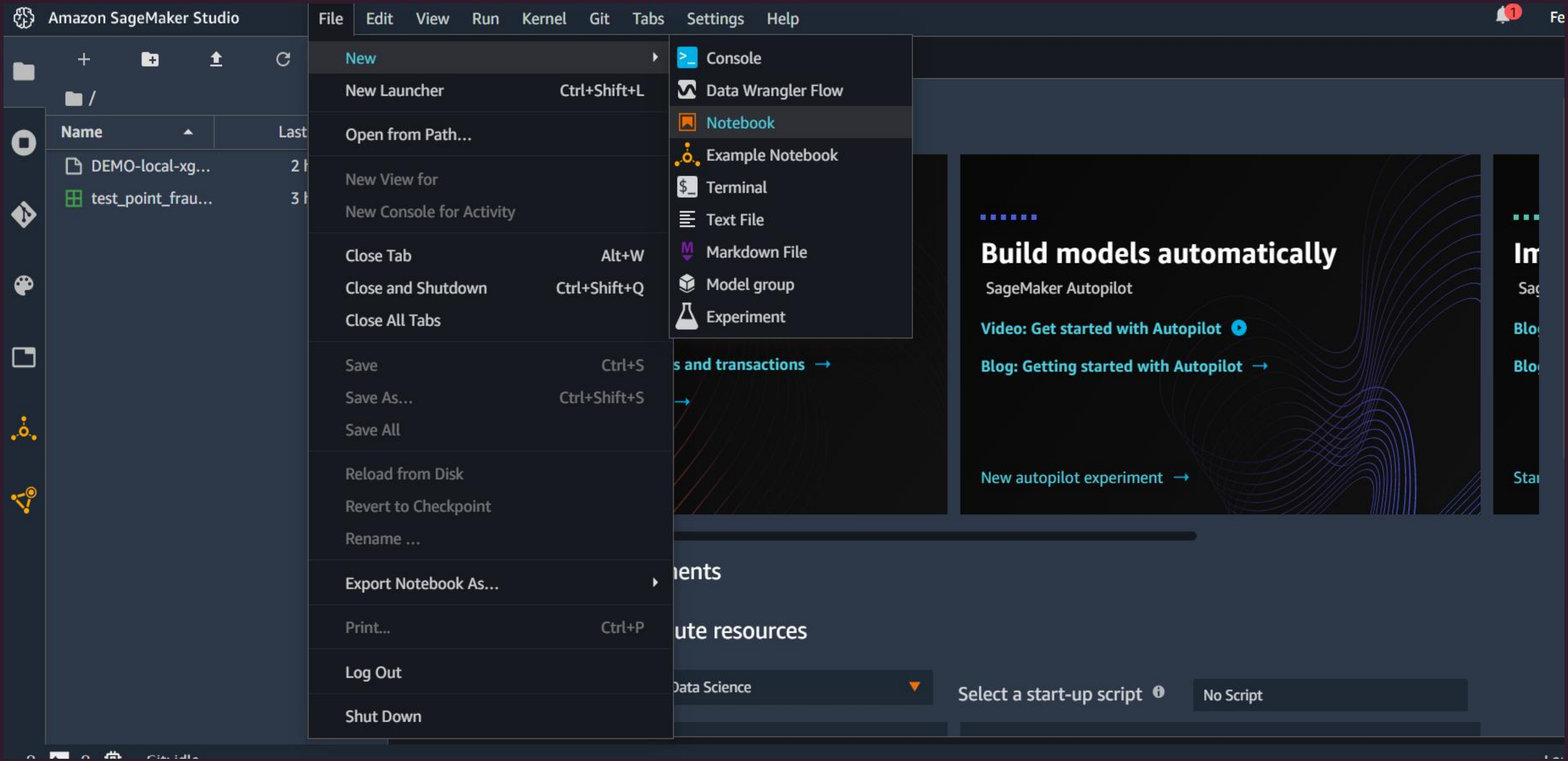
- *On the Control* Panel, choose your user name and then choose Launch app. Select Studio.



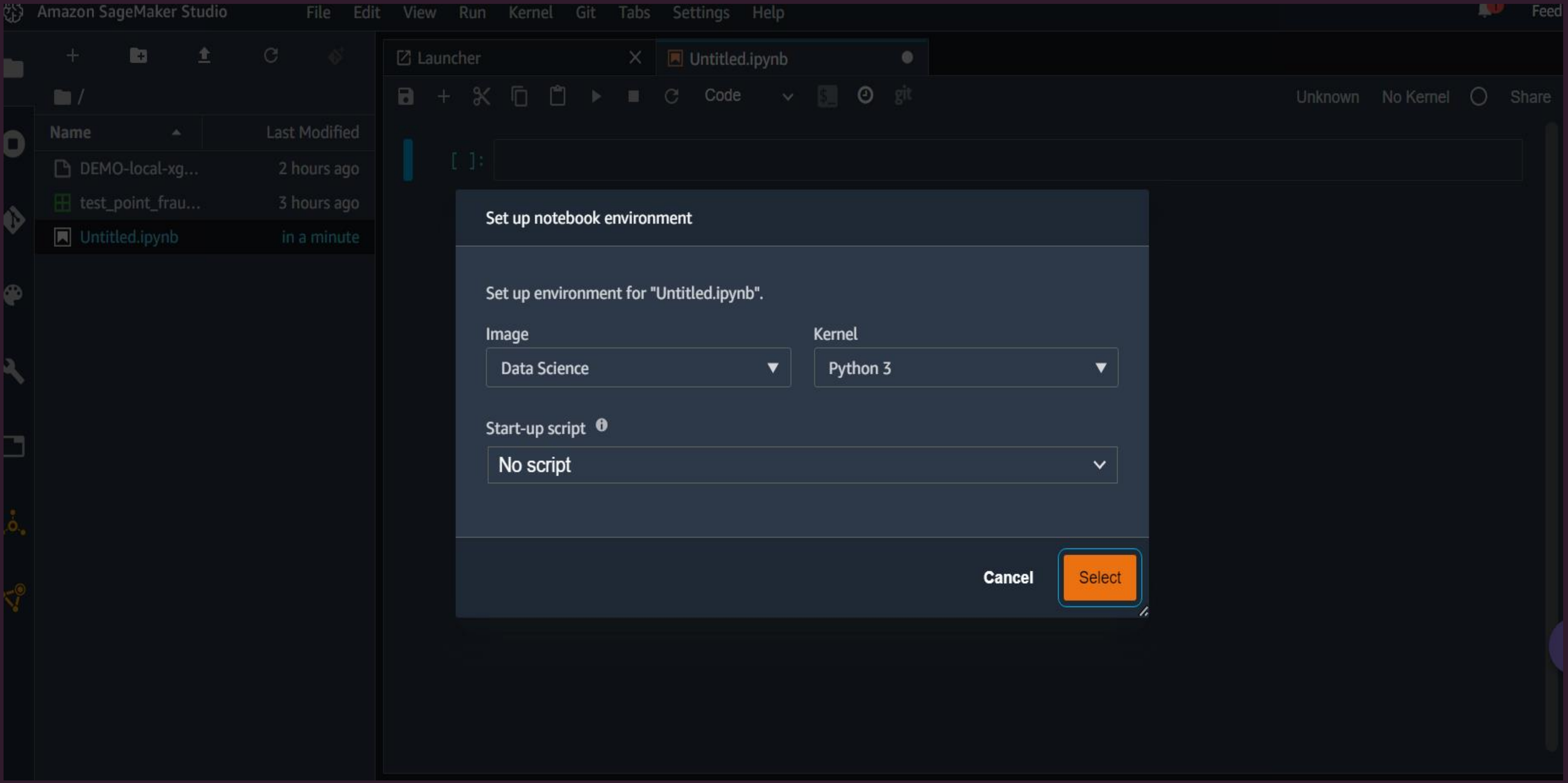
- Upload the locally trained model (DEMO-local-xgboost-model-fraud-detection) and the test_point_fraud.csv files to the SageMaker Studio.



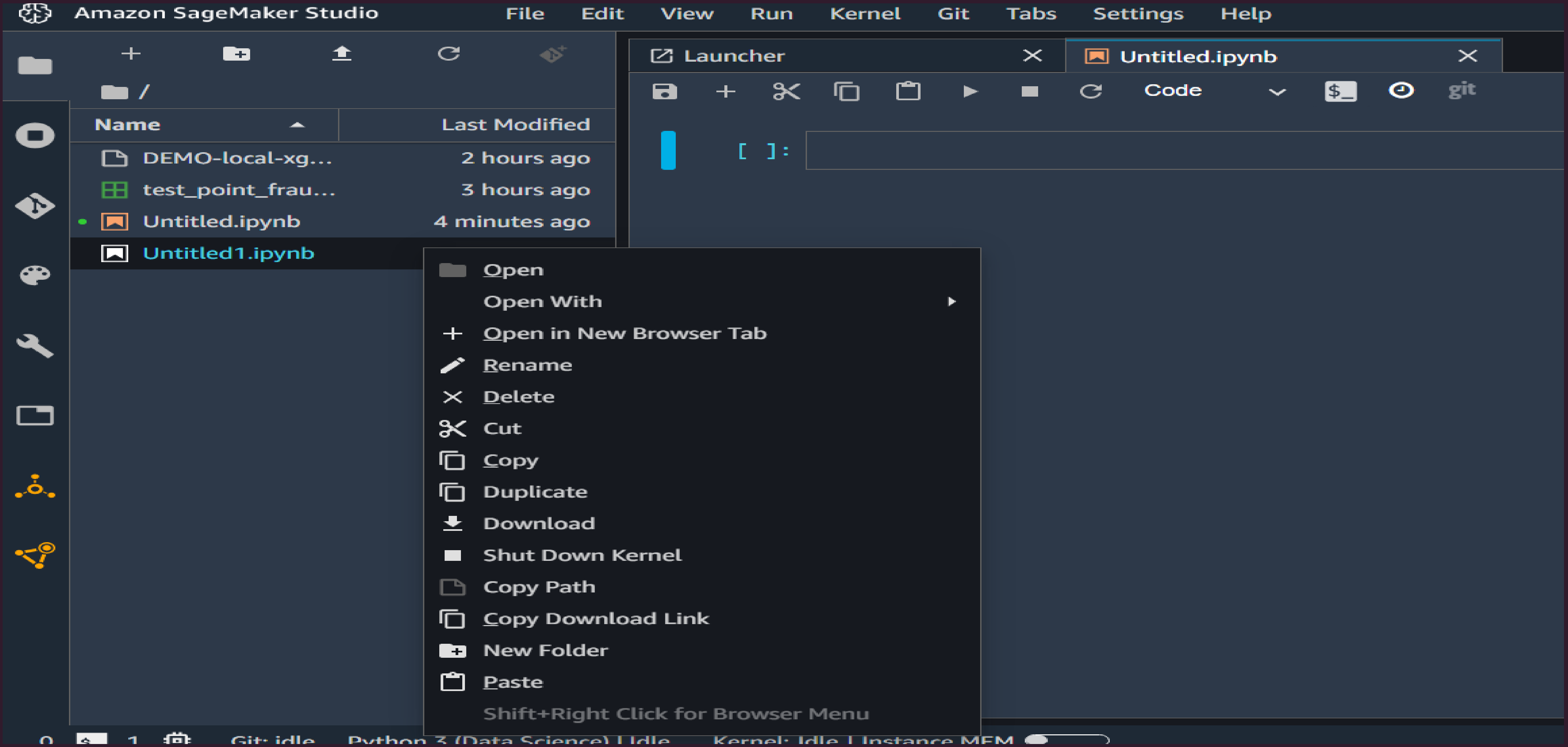
- **Create New Notebook in SageMaker Studio – Click on File-----> Click on New----->Click on Notebook**



Set up Notebook Environment as shown below



- **Rename notebook file as 'Auto-Insurance-Model-Registry' and start running file with code.**



• Import libraries and Create S3 bucket

Auto_Insurance_Model_Regis ×

Code | 2 vCPU + 4 GiB | Cluster | Python 3 (Data Science)

[2]:
```python
%%time

import os
import boto3
import sagemaker

from sagemaker import get_execution_role

region = boto3.Session().region_name

role = get_execution_role()
```

```
CPU times: user 688 ms, sys: 110 ms, total: 799 ms
Wall time: 1.22 s
```

Create S3 bucket

[3]:
```python
# This creates a default S3 bucket where we will upload our model.
bucket = sagemaker.Session().default_bucket()
```

[4]:
```python
bucket_path = "https://s3-{}.amazonaws.com/{}".format(region, bucket)
```

[5]:
```python
print(role)
print(region)
print(bucket)
```

- Install xgboost as it is needed for loading the model from joblib dump file and test it before deployment.
- Please note that, the XGBoost version should be same as the version with which the model was trained locally in laptop.
- In Our Case, XGBoost version is 1.5.0 , So Install the same version in SageMaker Studio Notebook.

```
[6]: !conda install -y -c conda-forge xgboost==1.5.0

Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/conda

  added / updated specs:
    - xgboost==1.5.0


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    _openmp_mutex-5.1          |            1_gnu          21 KB
    _py-xgboost-mutex-2.0      |            cpu_0           8 KB  conda-forge
    libgomp-11.2.0             |       h1234567_1         474 KB
    libxgboost-1.5.0           |       h295c915_1         2.0 MB
    openssl-1.1.1q             |       h7f8727e_0         2.5 MB
    py-xgboost-1.5.0           |   py37h06a4308_1         162 KB
    xgboost-1.5.0              |   py37h06a4308_1          25 KB
    ------------------------------------------------------------
```

- **Load the pre-trained Model (DEMO-local-xgboost-model-fraud-detection) and call & read the test file (test_point_fraud.csv) in SageMaker Studio Notebook.**

```
[8]: model_file_name = "DEMO-local-xgboost-model-fraud-detection"
```

```
[ ]: import joblib
     import xgboost

     mymodel = joblib.load(model_file_name)
```

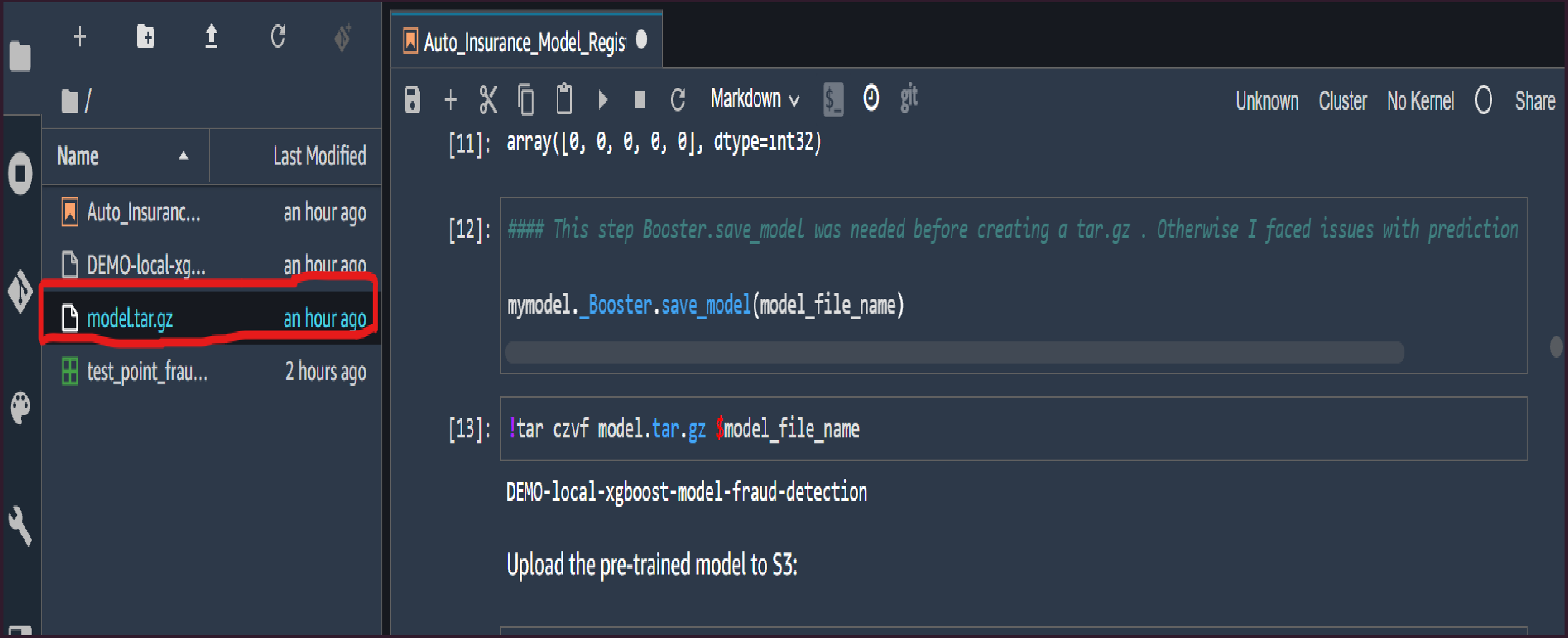
```
[9]: #import json
     import numpy as np


     file_name = (
         "test_point_fraud.csv"  # customize to your test file, will be 'mnist.single.test' if use data above
     )

     with open(file_name, "r") as f:
         mypayload = np.loadtxt(f, delimiter=",")

     print(mypayload)
```

```
[[ 5.00000e+00  2.60000e+01  0.00000e+00  1.00000e+00  2.00000e+03
   1.13702e+03  0.00000e+00  0.00000e+00  6.00000e+00  4.00000e+00
```

- **Test the model before deployment (i.e. Make predictions on test data)**
- **Note that, predictions should be same for locally trained (Jupyter Notebook) model and Model Uploaded in SageMaker Studio Notebook. Here, predictions are : [0,0,0,0,0]**

```
  0.00000e+00  0.00000e+00  6.28000e+04  6.28000e+03  1.25600e+04
  4.39600e+04  7.00000e+00  3.60000e+01  2.01200e+03]
 [ 8.40000e+01  2.90000e+01  2.00000e+00  1.00000e+00  1.00000e+03
  1.11717e+03  0.00000e+00  0.00000e+00  2.00000e+00  6.00000e+00
  1.80000e+01  1.00000e+00  0.00000e+00 -2.99000e+04  1.00000e+00
  1.00000e+00  3.00000e+00  4.00000e+00  4.00000e+00  0.00000e+00
  2.94000e+02  6.00000e+00  1.00000e+00  1.00000e+00  2.00000e+00
  0.00000e+00  1.00000e+00  6.82000e+03  6.20000e+02  1.24000e+03
  4.96000e+03  2.00000e+00  0.00000e+00  2.00500e+03]]

[10]: mymodel.predict(mypayload)

[10]: array([0, 0, 0, 0, 0], dtype=int32)
```

- **Now, First Save the model and then Create model.tar.gz file of the model.**
- **This step Booster.save_model is needed before creating a tar.gz . Otherwise, we will face issues with prediction on deployment.**



- **Upload the pre-trained model to S3 Bucket.**
- **Model Path : sagemaker-ap-south-1-208779919433/Sagemaker/DEMO-XGBoost-Auto-Insurance/DEMO-local-xgboost-model-fraud-detection/model.tar.gz**

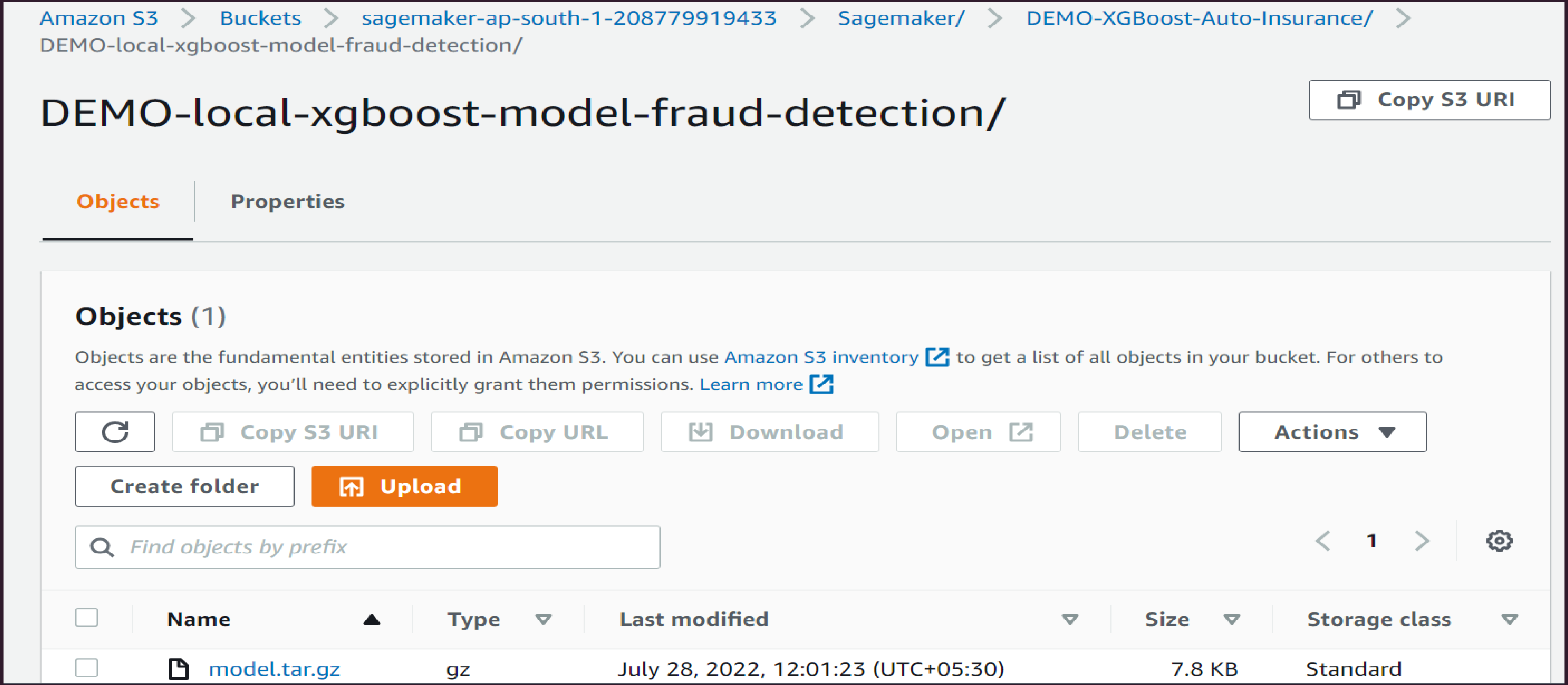
Upload the pre-trained model to S3:

[13]:
```
#### prefix in S3
prefix = "Sagemaker/DEMO-XGBoost-Auto-Insurance"

fObj = open("model.tar.gz", "rb")
key = os.path.join(prefix, model_file_name, "model.tar.gz")
print(key)
boto3.Session().resource("s3").Bucket(bucket).Object(key).upload_fileobj(fObj)
```

```
Sagemaker/DEMO-XGBoost-Auto-Insurance/DEMO-local-xgboost-model-fraud-detection/model.tar.gz
```

Set up hosting for the model:

- **View Uploaded Model in S3 Bucket: Amazon S3 > Buckets > sagemaker-ap-south-1-208779919433 > SageMaker > DEMO-XGBoost-Auto-Insurance > DEMO-local-xgboost-model-fraud-detection > model.tar.gz**



Import Model into Hosting

- This involves creating a SageMaker model from the model file previously uploaded to S3.
- Before creating SageMaker Model, we need to call SageMaker in-built container.
- Remember SageMaker having 3 in-built containers for classification models - XGBoost, Linear Learner and KNN.
- get_image_uri(region_name, repo_name, repo_version=1.5-1)
- region_name: name of the region, repo_name: name of the repo (e.g. xgboost), repo_version: **version of the repo**

[14]:
```
from sagemaker.amazon.amazon_estimator import get_image_uri

#### Get the built-in xgboost container image in Sagemaker to host our model
container = get_image_uri(boto3.Session().region_name, "xgboost", "1.5-1")
```

```
The method get_image_uri has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
```

- Create SageMaker Model: To Create SageMaker Model following parameters are required:
- ModelName= Name of Model File [DEMO-local-xgboost-model-fraud-detection]
- ExecutionRoleArn= SageMaker Role [get_execution_role()]
- PrimaryContainer = { "Image": container, "ModelDataUrl": model_url}

```
[15]: %%time
from time import gmtime, strftime

model_name = model_file_name + strftime("%Y-%m-%d-%H-%M-%S", gmtime())

model_url = "https://s3-{}.amazonaws.com/{}/{}".format(region, bucket, key)

sm_client = boto3.client("sagemaker")

print(model_url)

primary_container = {
    "Image": container,
    "ModelDataUrl": model_url,
}

create_model_response2 = sm_client.create_model(
    ModelName=model_name, ExecutionRoleArn=role, PrimaryContainer=primary_container
)

print(create_model_response2["ModelArn"])
```

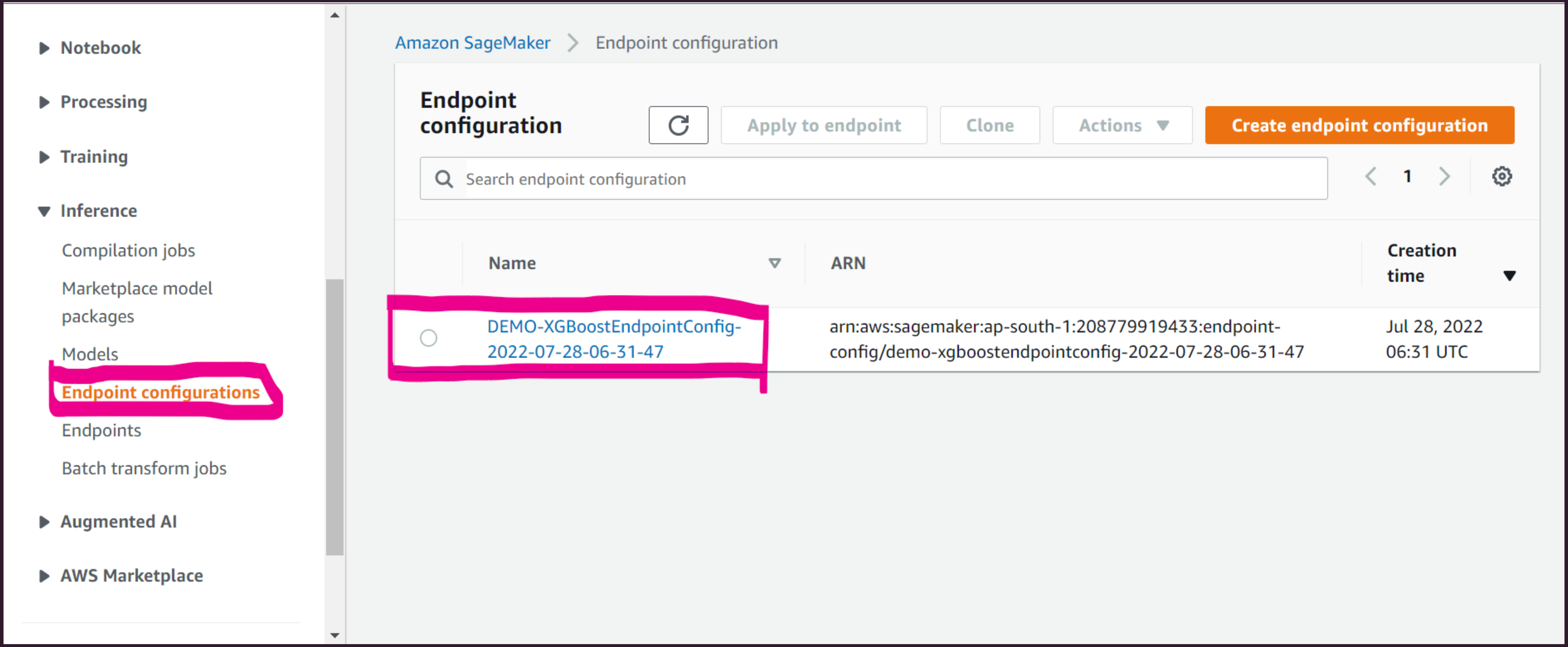
```
https://s3-ap-south-1.amazonaws.com/sagemaker-ap-south-1-208779919433/Sagemaker/DEMO-XGBoost-Auto-Insurance/DEMO-local-xgboost-model-fraud-detection/model.tar.gz
arn:aws:sagemaker:ap-south-1:208779919433:model/demo-local-xgboost-model-fraud-detection2022-07-26-08-38-09
CPU times: user 59.9 ms, sys: 16.8 ms, total: 76.7 ms
```

- Create endpoint configuration : Create an endpoint configuration, that describes the distribution of traffic across the models, whether split, shadowed, or sampled in some way. In addition, the endpoint configuration describes the instance type required for model deployment.

[16]:
```python
from time import gmtime, strftime

endpoint_config_name = "DEMO-XGBoostEndpointConfig-" + strftime("%Y-%m-%d-%H-%M-%S", gmtime())

print(endpoint_config_name)

create_endpoint_config_response = sm_client.create_endpoint_config(
    EndpointConfigName=endpoint_config_name,
    ProductionVariants=[
        {
            "InstanceType": "ml.m4.xlarge",
            "InitialInstanceCount": 1,
            "InitialVariantWeight": 1,
            "ModelName": model_name,
            "VariantName": "AllTraffic",
        }
    ],
)

print("Endpoint Config Arn: " + create_endpoint_config_response["EndpointConfigArn"])
```

```
DEMO-XGBoostEndpointConfig-2022-07-26-08-38-21
Endpoint Config Arn: arn:aws:sagemaker:ap-south-1:208779919433:endpoint-config/demo-xgboostendpointconfig-2022-07
-26-08-38-21
```

- After running the notebook till this point, you can see the endpoint configuration created under Sagemaker -> Inference -> Endpoints Configurations in AWS console.

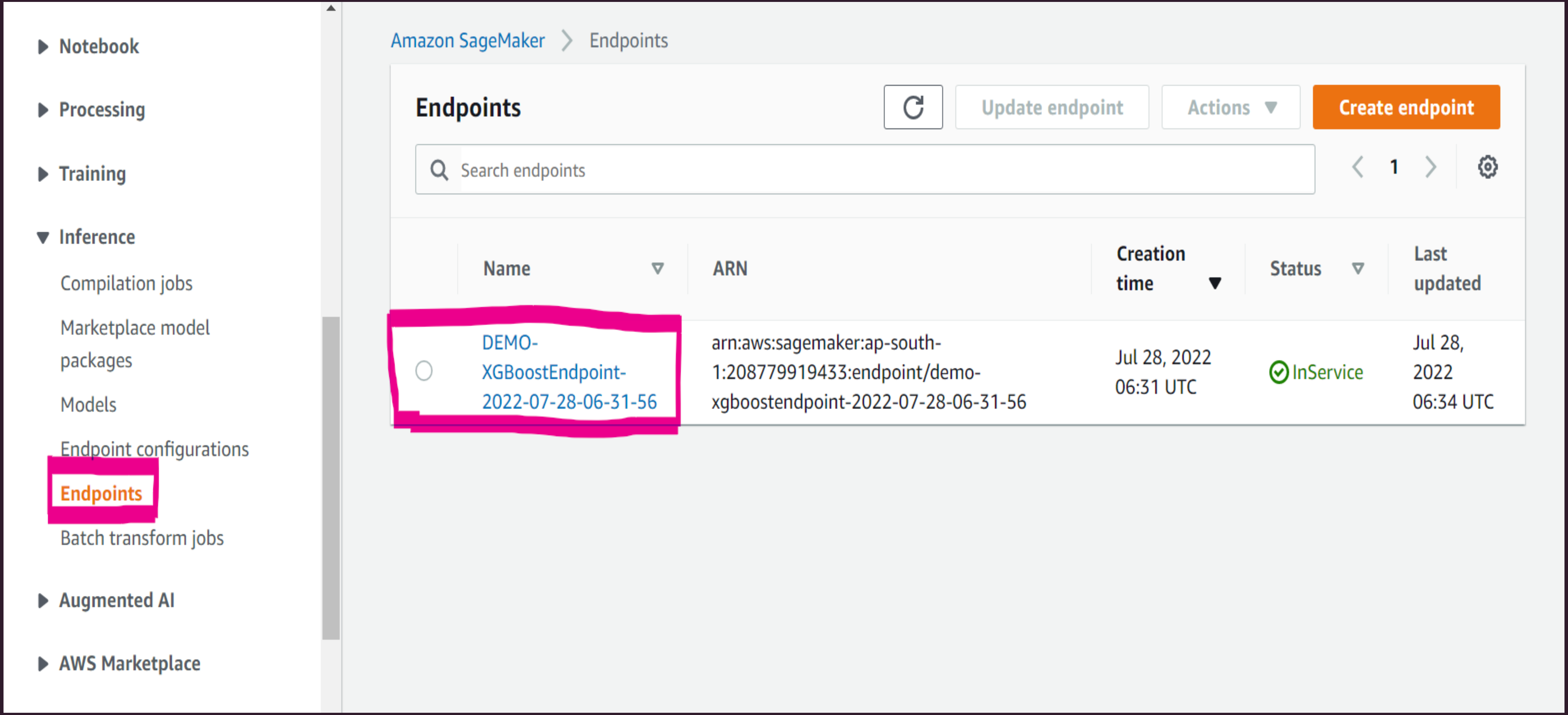


- Create endpoint: Lastly, you create the endpoint that serves up the model, through specifying the name and configuration defined above. The end result is an endpoint that can be validated and incorporated into production applications. This takes 3-4 minutes to complete.

[17]:
```
%%time
import time

endpoint_name = "DEMO-XGBoostEndpoint-" + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print(endpoint_name)
create_endpoint_response = sm_client.create_endpoint(
    EndpointName=endpoint_name, EndpointConfigName=endpoint_config_name
)
print(create_endpoint_response["EndpointArn"])

resp = sm_client.describe_endpoint(EndpointName=endpoint_name)
status = resp["EndpointStatus"]
print("Status: " + status)

while status == "Creating":
    time.sleep(60)
    resp = sm_client.describe_endpoint(EndpointName=endpoint_name)
    status = resp["EndpointStatus"]
    print("Status: " + status)

print("Arn: " + resp["EndpointArn"])
print("Status: " + status)
```

```
DEMO-XGBoostEndpoint-2022-07-26-08-38-30
arn:aws:sagemaker:ap-south-1:208779919433:endpoint/demo-xgboostendpoint-2022-07-26-08-38-30
Status: Creating
Status: Creating
```

- After running the notebook till this point, you can see the endpoint configuration created under SageMaker -> Inference -> Endpoints in AWS console.



- Validate the model for use:
- Now you can obtain the endpoint from the client library using the result from previous operations and generate classifications from the model using that endpoint.

```
[18]: runtime_client = boto3.client("runtime.sagemaker")
```

Lets generate the prediction. We'll pick csv data from the test data file

```
[19]: %%time
import json

file_name = (
    "test_point_fraud.csv"
)
with open(file_name, "r") as f:
    payload = f.read().strip()
print("Payload :\n")
print(payload)
print()
response = runtime_client.invoke_endpoint(
    EndpointName=endpoint_name, ContentType="text/csv", Body=payload
)
##print(response)
print("Results :\n")
print()
result = response["Body"].read().decode("ascii")
# Unpack response
print("\nPredicted Class Probabilities: {}.".format(result))
```

```
Payload :
```

- This is output of predictions on test data for SageMaker Model. Here, we will get output in terms of probabilities. We can set threshold (0.5) to convert output into exact value.
- Value < 0.5 = 0 and Value > 0.5 = 1.
- Here, all values are less than threshold (0.5) , so output will be ; [0,0,0,0,0]
- This output should be same as that of our previous prediction output in SageMaker as well as our local model output.

```
Results :



Predicted Class Probabilities: 0.07987239956855774
0.07372141629457474
0.07987239956855774
0.11815092712640762
0.10339038819074631
.
CPU times: user 13.2 ms, sys: 2.13 ms, total: 15.3 ms
Wall time: 135 ms
```

Register and Deploy Models with Model Registry

Model Registry: A central repository of trained models

- With the SageMaker model registry you can do the following:

1. Catalog models for production.
2. Manage model versions.
3. Associate metadata, such as training metrics, with a model.
4. Manage the approval status of a model.
5. Deploy models to production.
6. Automate model deployment with CI/CD.

Model Registry Structure

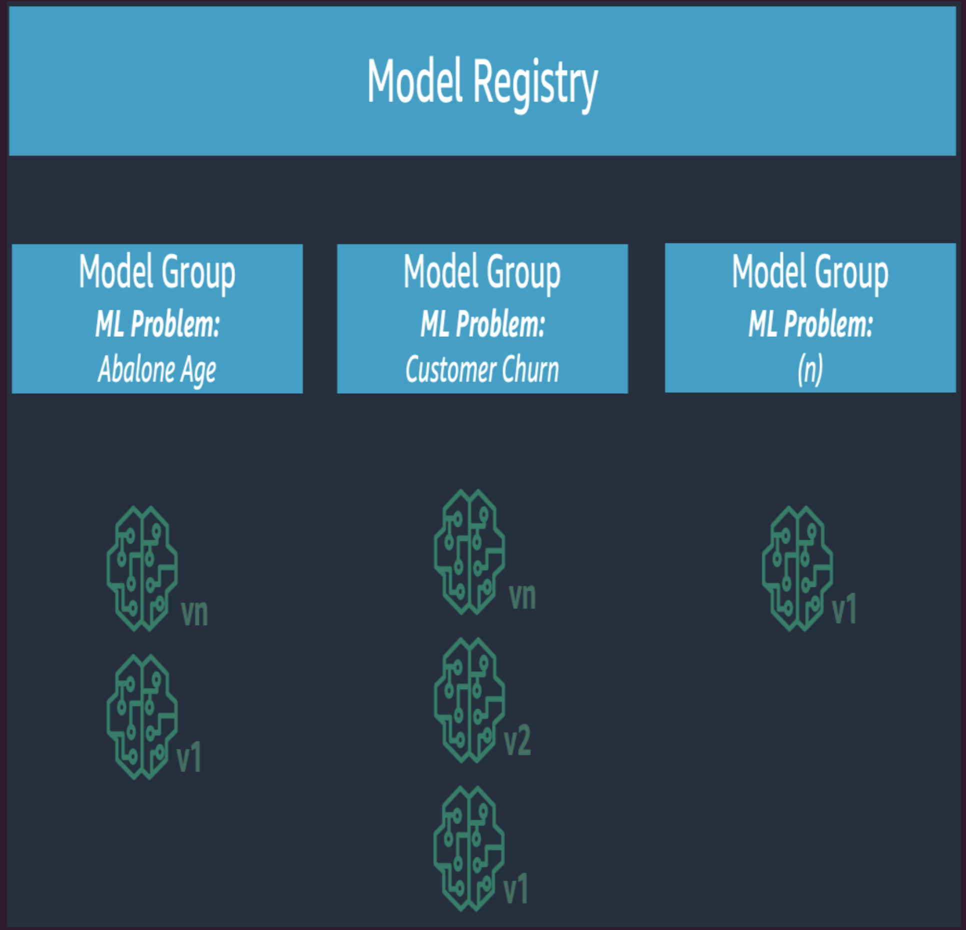
The SageMaker Model Registry is structured as several model groups with model packages in each group.

Each model package in a model group corresponds to a trained model.

The version of each model package is a numerical value that starts at 1 and is incremented with each new model package added to a model group.

For example, if 5 model packages are added to a model group, the model package versions will be 1, 2, 3, 4, and 5.

The example Model Registry shown in the following image contains 3 model groups, where each group contains the model packages related to a particular ML problem.



Model Registry- Steps

Create	Create a Model Group
Register	Register a Model Version
View	View Model Groups and Versions
View	View the Details of a Model Version
Update	Update the Approval Status of a Model
Deploy	Deploy a Model from the Registry
View	View the Deployment History of a Mode
Delete	Delete a Model Version
Delete	Delete a Model Group

1.Create a Model Group:

- A model group contains a group of versioned models. Create a model group by using either the AWS SDK for Python (Boto3) or Amazon SageMaker Studio.
- To create a model group by using Boto3, call the create_model_package_group method and specify a name and description as parameters.
- The following example shows how to create a model group. The response from the create_model_package_group call is the Amazon Resource Name (ARN) of the new model package group.

- First, import the required packages and set up the SageMaker Boto3 client.
- Secondly, create the model group.

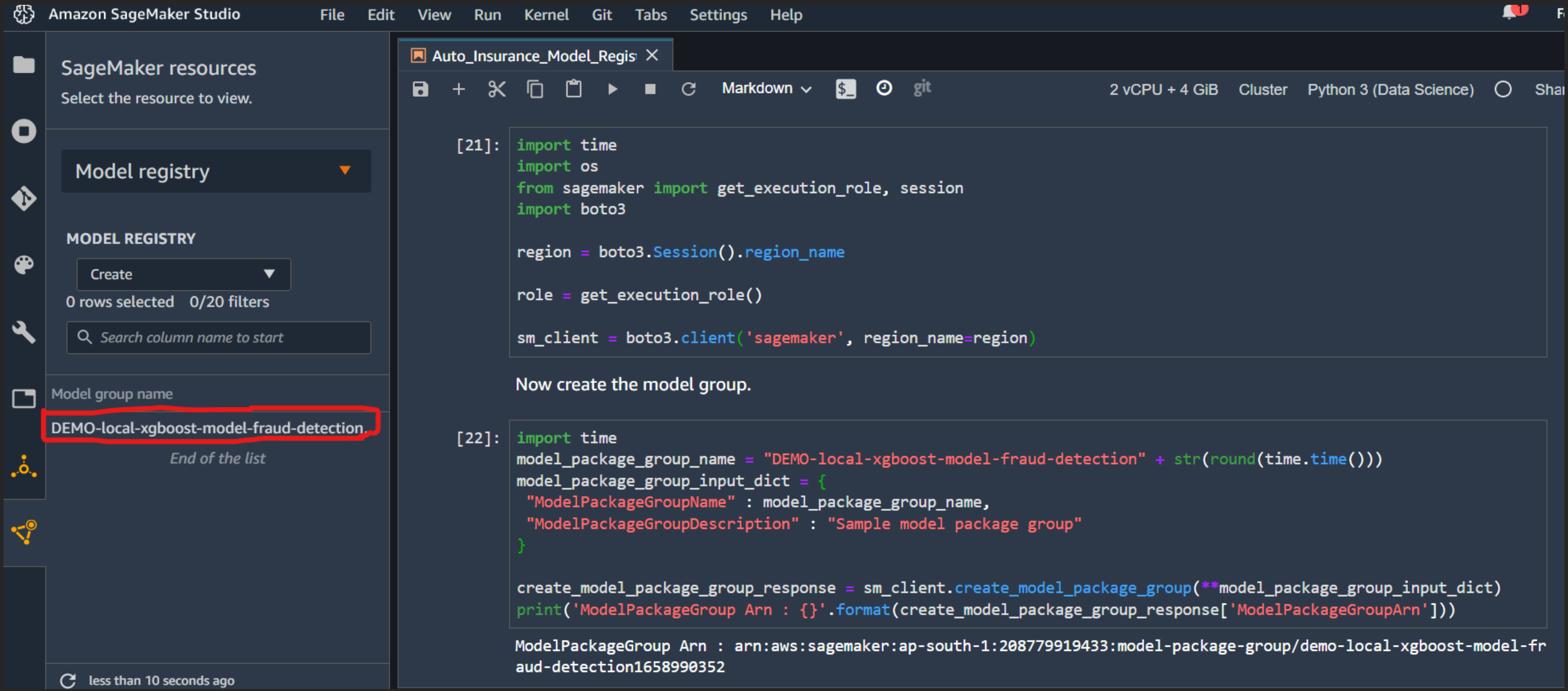
[20]:
```
import time
import os
from sagemaker import get_execution_role, session
import boto3

region = boto3.Session().region_name

role = get_execution_role()

sm_client = boto3.client('sagemaker', region_name=region)
```

Now create the model group.

[21]:
```
import time
model_package_group_name = "DEMO-local-xgboost-model-fraud-detection" + str(round(time.time()))
model_package_group_input_dict = {
 "ModelPackageGroupName" : model_package_group_name,
 "ModelPackageGroupDescription" : "Sample model package group"
}

create_model_package_group_response = sm_client.create_model_package_group(**model_package_group_input_dict)
print('ModelPackageGroup Arn : {}'.format(create_model_package_group_response['ModelPackageGroupArn']))
```

```
ModelPackageGroup Arn : arn:aws:sagemaker:ap-south-1:208779919433:model-package-group/demo-local-xgboost-model-fr
aud-detection1658824993
```

- Now, we can check created model group from Model Registry.
- View at left bottom of SageMaker Studio, the last option is SageMaker Resources.
- SageMaker Resources --> Select Resource to view (Model Registry) --> Select Model Group Name(`'DEMO-local-xgboost-model-fraud-detection1658824993'`)



2.Register a Model Version:

- You can register an Amazon SageMaker model by creating a model version that specifies the model group to which it belongs.
- A model version must include both the model artifacts (the trained weights of a model) and the inference code for the model.

- To register a model version by using Boto3, call the create_model_package method.
- First, you set up the parameter dictionary to pass to the create_model_package method.

- create_model_package_input_dict = { "ModelPackageGroupName" : "--Your model package group name--", "ModelPackageDescription" : "--Description about your model--","ModelApprovalStatus" : "PendingManualApproval" }   # Initially set the Pending Status
- create_model_package_input_dict.update(modelpackage_inference_specification)

[22]:
```python
# Specify the model source

model_url = "https://s3-{}.amazonaws.com/{}/{}".format(region, bucket, key)

modelpackage_inference_specification =  {
    "InferenceSpecification": {
      "Containers": [
         {
            "Image": container,
           "ModelDataUrl": model_url
         }
      ],
      "SupportedContentTypes": [ "text/csv" ],
      "SupportedResponseMIMETypes": [ "text/csv" ],
   }
 }

create_model_package_input_dict = {
    "ModelPackageGroupName" : model_package_group_name,
    "ModelPackageDescription" : "Model to detect wheather the claim is fraud or non-fraud",
    "ModelApprovalStatus" : "PendingManualApproval"
}
create_model_package_input_dict.update(modelpackage_inference_specification)
```

- Then you call the **create_model_package** method, passing in the parameter dictionary that you just set up

```
[23]: create_model_package_response = sm_client.create_model_package(**create_model_package_input_dict)
      model_package_arn = create_model_package_response["ModelPackageArn"]
      print('ModelPackage Version ARN : {}'.format(model_package_arn))
```

```
ModelPackage Version ARN : arn:aws:sagemaker:ap-south-1:208779919433:model-package/demo-local-xgboost-model-fraud
-detection1658824993/1
```

3.View Model Groups and Versions:

Model groups and versions help you organize your models. You can view a list of the model versions in a model group.

You can view all of the model versions that are associated with a model group.

If a model group represents all models that you train to address a specific ML problem, you can view all of those related models.

- To view model versions associated with a model group by using Boto3, call the *list_model_packages* method, and pass the name of the model group as the value of the ModelPackageGroupName parameter.
- It gives model package summary that includes *Model Package Group Name, ARN, Description, Creation Time,Package status, Approval Status etc.*
- In this step, Model Package ARN(Amazon Resource Name) is generated which is important to see the details of the model version , which will be our next stpe.

```
[24]: sm_client.list_model_packages(ModelPackageGroupName=model_package_group_name)
```


```
[24]: {'ModelPackageSummaryList': [{'ModelPackageGroupName': 'DEMO-local-xgboost-model-fraud-detection1658824993',
         'ModelPackageVersion': 1,
         'ModelPackageArn': 'arn:aws:sagemaker:ap-south-1:208779919433:model-package/demo-local-xgboost-model-fraud-det
      ection1658824993/1',
         'ModelPackageDescription': 'Model to detect wheather the claim is fraud or non-fraud',
         'CreationTime': datetime.datetime(2022, 7, 26, 8, 44, 29, 679000, tzinfo=tzlocal()),
         'ModelPackageStatus': 'Completed',
         'ModelApprovalStatus': 'PendingManualApproval'}],
       'ResponseMetadata': {'RequestId': '2d8c3a11-b659-4f28-b20d-a92f0dadc20c',
        'HTTPStatusCode': 200,
        'HTTPHeaders': {'x-amzn-requestid': '2d8c3a11-b659-4f28-b20d-a92f0dadc20c',
         'content-type': 'application/x-amz-json-1.1',
         'content-length': '457',
         'date': 'Tue, 26 Jul 2022 08:45:37 GMT'},
        'RetryAttempts': 0}}
```

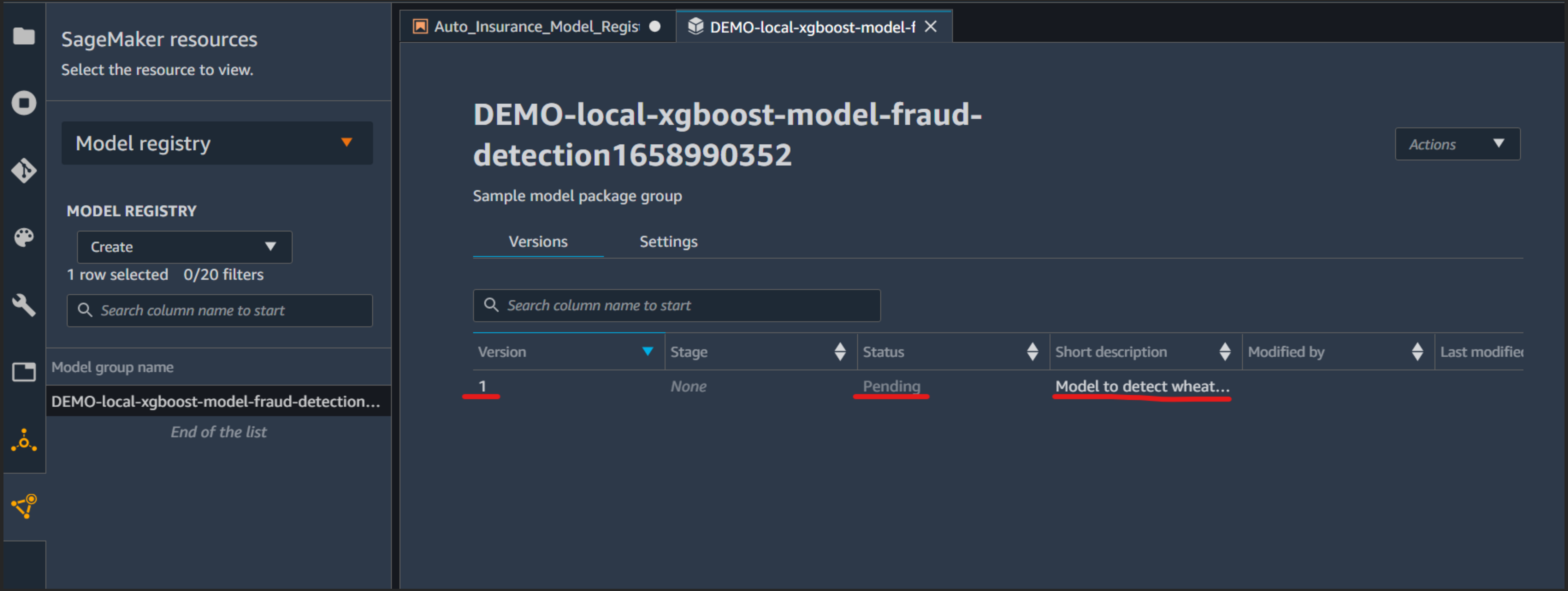
4.View the Details of a Model Version:

- Call describe_model_package to see the details of the model version. You pass in the ARN of a model version that you got in the output of the call to list_model_packages **(See the Step-3 for ARN)**
- The output of this call is a JSON object with the model version details.

```
[26]: sm_client.describe_model_package(ModelPackageName="arn:aws:sagemaker:ap-south-1:208779919433:model-package/demo-l
```


```
[26]: {'ModelPackageGroupName': 'DEMO-local-xgboost-model-fraud-detection1658824993',
 'ModelPackageVersion': 1,
 'ModelPackageArn': 'arn:aws:sagemaker:ap-south-1:208779919433:model-package/demo-local-xgboost-model-fraud-detec
tion1658824993/1',
 'ModelPackageDescription': 'Model to detect wheather the claim is fraud or non-fraud',
 'CreationTime': datetime.datetime(2022, 7, 26, 8, 44, 29, 679000, tzinfo=tzlocal()),
 'InferenceSpecification': {'Containers': [{'Image': '720646828776.dkr.ecr.ap-south-1.amazonaws.com/sagemaker-xgb
oost:1.5-1',
    'ImageDigest': 'sha256:900372db4cdb1f34e9eae2de344350b53b4d7c141e883a1e197fd3576c73d2a6',
    'ModelDataUrl': 'https://s3-ap-south-1.amazonaws.com/sagemaker-ap-south-1-208779919433/Sagemaker/DEMO-XGBoost
-Auto-Insurance/DEMO-local-xgboost-model-fraud-detection/model.tar.gz'}],
  'SupportedContentTypes': ['text/csv'],
  'SupportedResponseMIMETypes': ['text/csv']},
 'ModelPackageStatus': 'Completed',
 'ModelPackageStatusDetails': {'ValidationStatuses': [],
  'ImageScanStatuses': []},
 'CertifyForMarketplace': False,
 'ModelApprovalStatus': 'PendingManualApproval',
 'CreatedBy': {'UserProfileArn': 'arn:aws:sagemaker:ap-south-1:208779919433:user-profile/d-7mbrrlmiagw2/default-1
658812828101',
  'UserProfileName': 'default-1658812828101',
  'DomainId': 'd-7mbrrlmiagw2'},
```

- We can view the details of model group such as model version number, Stage, Status, Description of model , Modification Information from model registry.
- In this case, only one model is there, so we can see details of one model only. Stage is not defined here, default it is taking 'none'. Initially, status for all the model is set as 'Pending'. Later , we can update status to Approved once model is 'OK' to deploy in production and we can set Stage to 'Prod' (Production).



5.Update the Approval Status of a Model:

After you create a model version, you typically want to evaluate its performance before you deploy it to a production endpoint.

If it performs to your requirements, you can update the approval status of the model version to Approved.

Setting the status to Approved can initiate CI/CD deployment for the model.

If the model version does not perform to your requirements, you can update the approval status to Rejected.

5.Update the Approval Status of a Model:

- PendingManualApproval to Approved – initiates CI/CD deployment for the approved model version
- PendingManualApproval to Rejected – No action
- Rejected to Approved – initiates CI/CD deployment for the approved model version
- Approved to Rejected – initiates CI/CD to deploy the latest model version with an Approved status

5.Update the Approval Status of a Model:

- When you created the model version in Register a Model Version, you set the ModelApprovalStatus to PendingManualApproval.
- You update the approval status for the model by calling update_model_package.
- Note that you can automate this process by writing code that, for example, sets the approval status of a model depending on the result of an evaluation of some measure of the model's performance.
- You can also create a step in a pipeline that automatically deploys a new model version when it is approved.

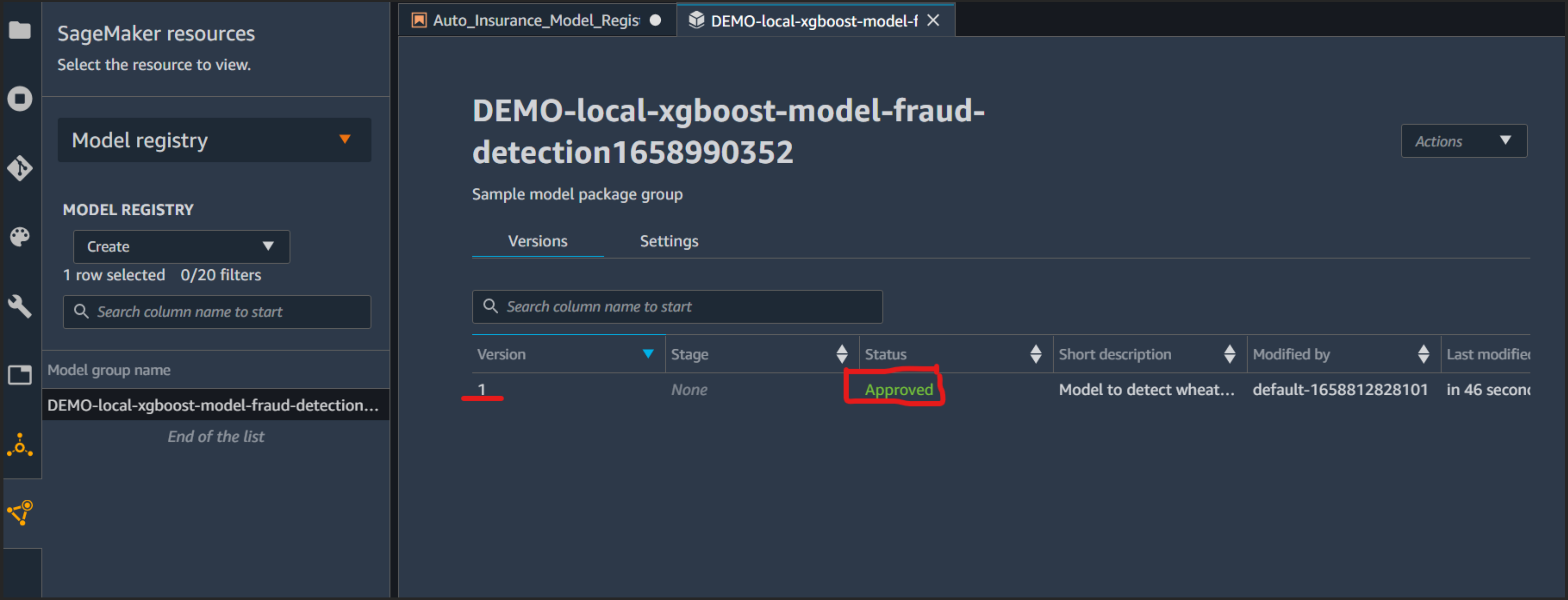
- Here, In first block of code , we updated **ModelApprovalStatus** from "*PendingManualApproval*" to "Approved" (highlighted below)
- In second step, we have repeated step no.3 to check updated status.

```
[27]: model_package_update_input_dict = {
          "ModelPackageArn" : model_package_arn,
          "ModelApprovalStatus" : "Approved"
      }
      model_package_update_response = sm_client.update_model_package(**model_package_update_input_dict)
```

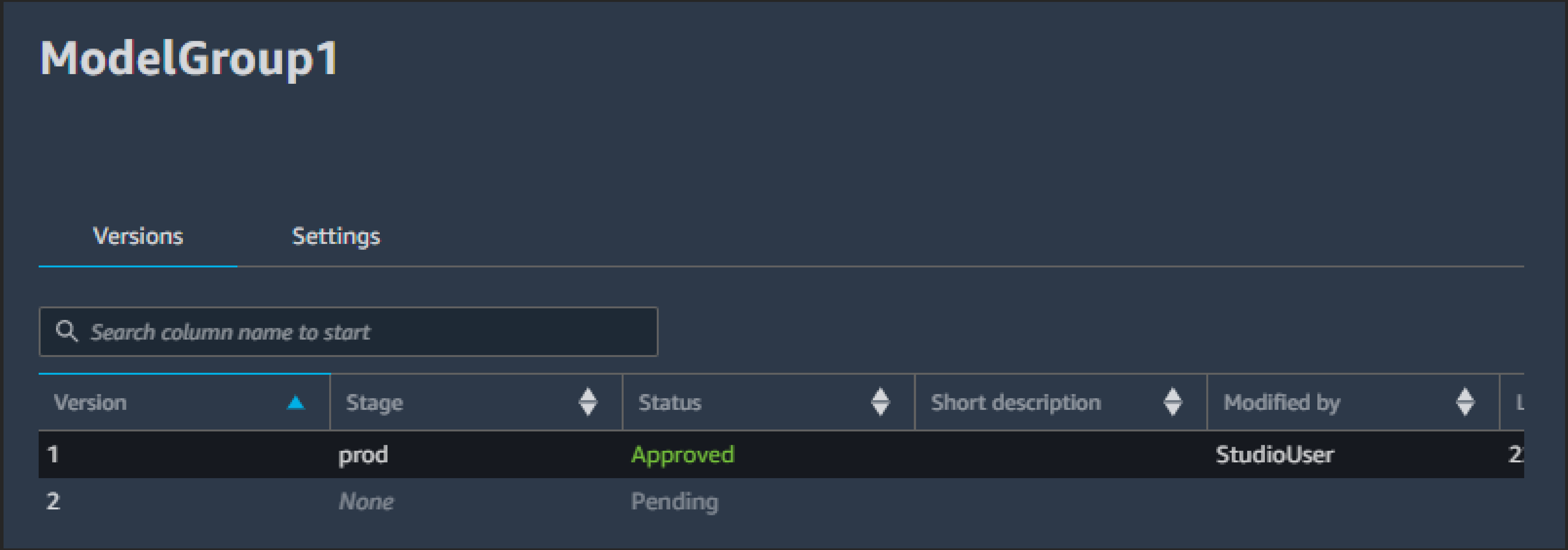
```
[28]: sm_client.list_model_packages(ModelPackageGroupName=model_package_group_name)
```


```
[28]: {'ModelPackageSummaryList': [{'ModelPackageGroupName': 'DEMO-local-xgboost-model-fraud-detection1658824993',
         'ModelPackageVersion': 1,
         'ModelPackageArn': 'arn:aws:sagemaker:ap-south-1:208779919433:model-package/demo-local-xgboost-model-fraud-det
      ection1658824993/1',
         'ModelPackageDescription': 'Model to detect wheather the claim is fraud or non-fraud',
         'CreationTime': datetime.datetime(2022, 7, 26, 8, 44, 29, 679000, tzinfo=tzlocal()),
         'ModelPackageStatus': 'Completed',
         'ModelApprovalStatus': 'Approved'}],
       'ResponseMetadata': {'RequestId': 'd512c257-8012-4037-9b86-e49a8adeafdc',
        'HTTPStatusCode': 200,
        'HTTPHeaders': {'x-amzn-requestid': 'd512c257-8012-4037-9b86-e49a8adeafdc',
         'content-type': 'application/x-amz-json-1.1',
         'content-length': '444',
         'date': 'Tue, 26 Jul 2022 08:48:52 GMT'},
        'RetryAttempts': 0}}
```

- Also, we can check updated status from Model Registry.
- View at left bottom of SageMaker Studio, the last option is SageMaker Resources.
- SageMaker Resources --> Select Resource to view (Model Registry) --> Select Model Group Name (`'DEMO-local-xgboost-model-fraud-detection1658824993'`)



- Version-2 model is under process (It not performs to your requirements yet). Therefore, it's Stage is 'none' and Status is 'Pending'.
- Version-1 model is ready for deployment in production(It performs to your requirements). Therefore, we can update it's Stage from **'none'** to **'prod'** and Status from **'Pending'** to **'Approved'**



6. Delete a Model Group:

- To delete a model group in Amazon SageMaker Studio, complete the following steps.

1.Go to Amazon SageMaker Studio.

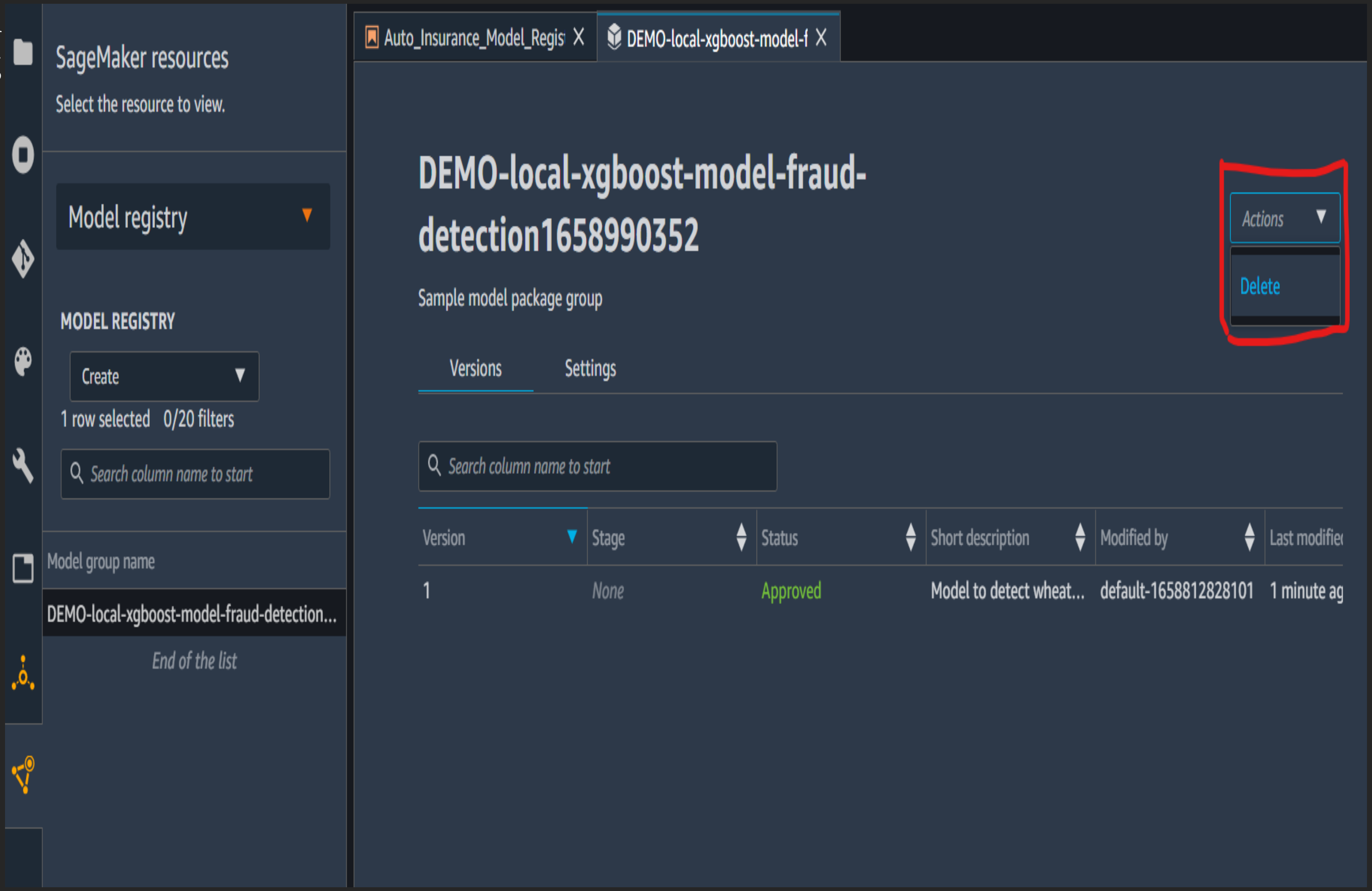
2.In the left navigation pane, choose the SageMaker Resources icon .

3.Choose Model registry in the dropdown menu at the top of the SageMaker resources panel. A list of your model groups appears.

4.From the model groups list, double-click the model group you want to delete. The model details tab opens to the right.

5.In the Actions dropdown menu in the top right corner of the model details tab, choose Delete.

6.In the confirmation dialog box, choose Delete.



Thank You....!!!