Project2: Machine Learning Cloud Service

Overview

Artificial intelligence and machine learning are steadily making their way into enterprise applications in areas such as customer support, fraud detection, and business intelligence. There is every reason to believe that much of it will happen in the cloud.

Benefits of ML in the Cloud

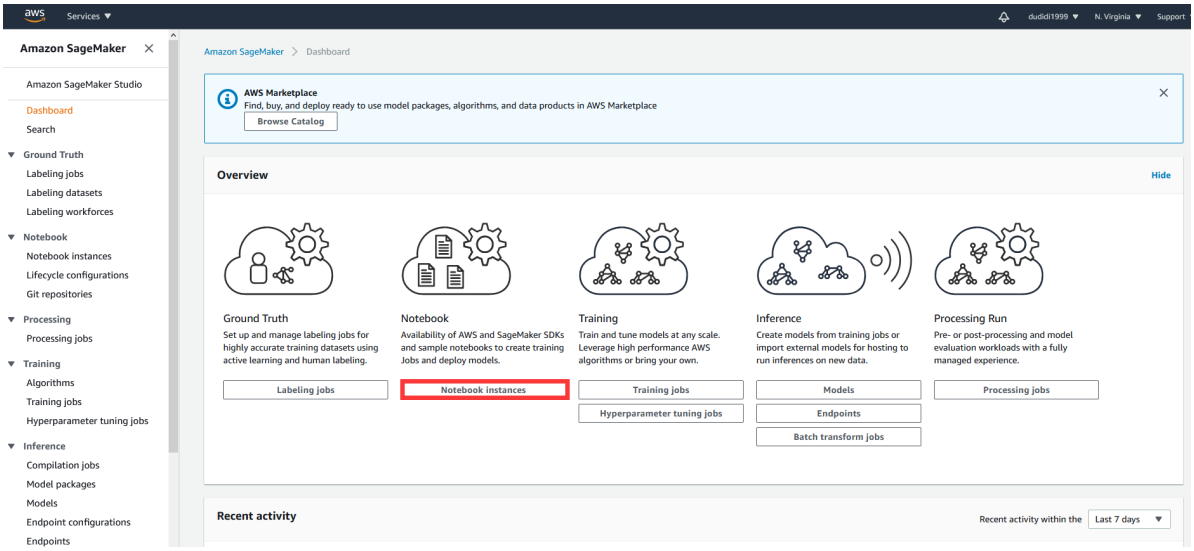
- The cloud's pay-per-use model is good for bursty AI or machine learning workloads.
- The cloud makes it easy for enterprises to experiment with machine learning capabilities and scale up as projects go into production and demand increases.
- The cloud makes intelligent capabilities accessible without requiring advanced skills in artificial intelligence or data science.
- AWS, Microsoft Azure, and Google Cloud Platform offer many machine learning options that don't require deep knowledge of AI, machine learning theory, or a team of data scientists.

Getting Started

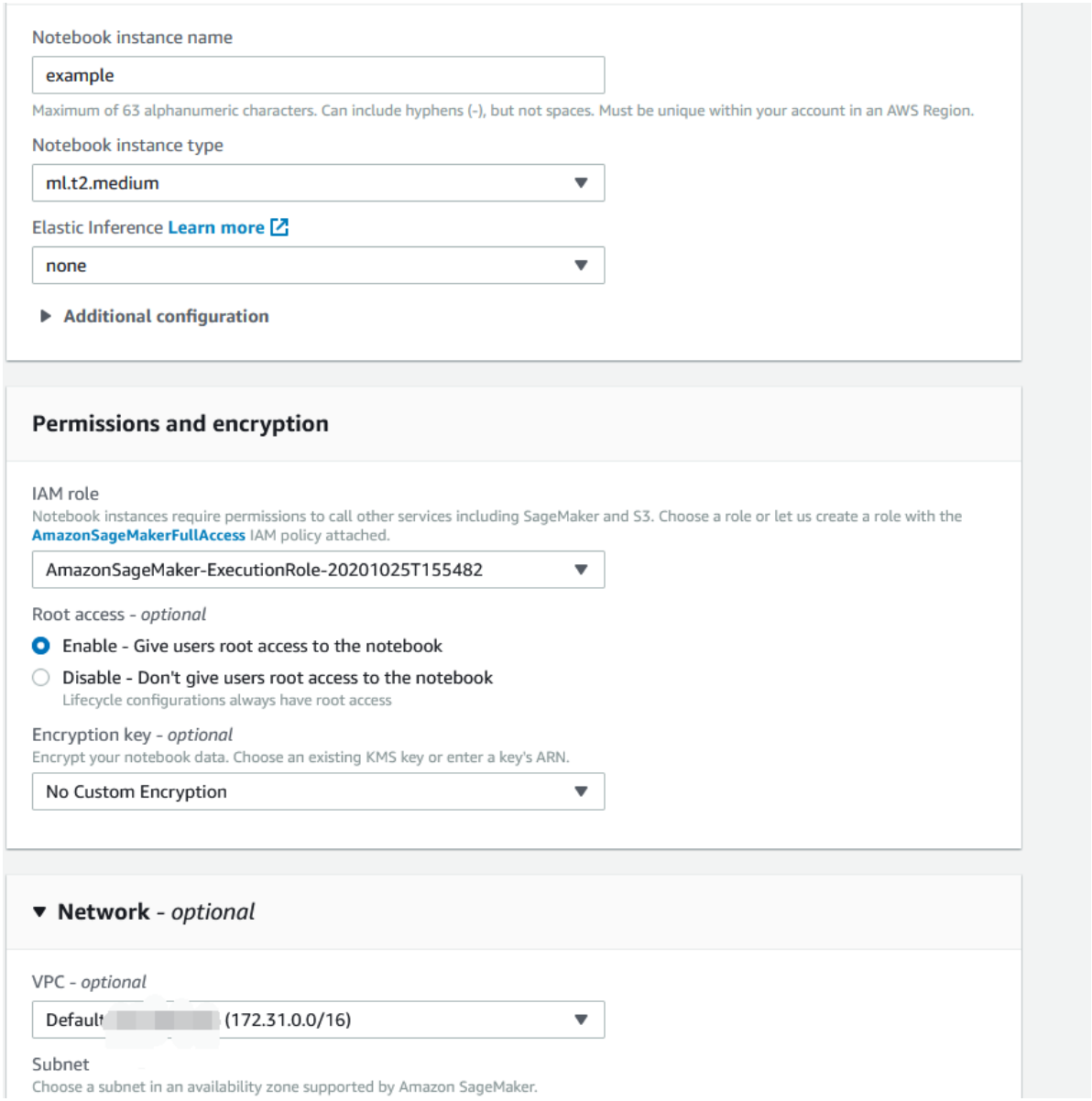
Here use AWS as an example. AWS has many free services. Commonly used EC2, S3, RDS, etc. all support a one-year limited free service, and Sagemaker also has a free experience service. Sagemaker is AWS's machine learning training platform, hosting Jupyter Notebook, and many commonly used machine learning algorithms are built-in.

Creating Instance

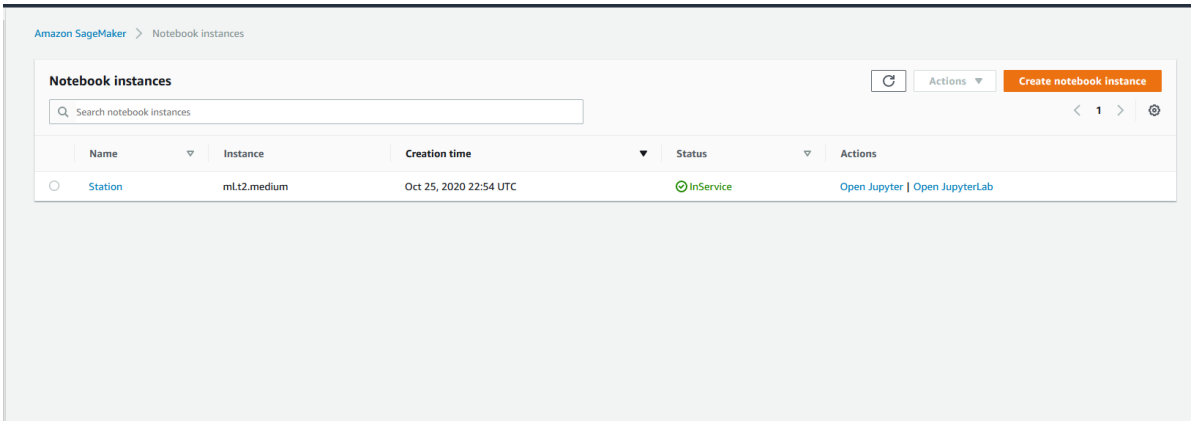
In dashboard of sagemaker, we can create notebook instances.



Give it a name. Then create a IAMrole, select VPC and subset. Omit other options.

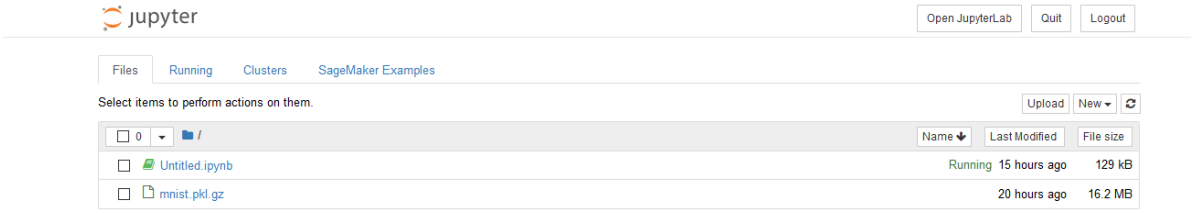


Wait until progress finishes.

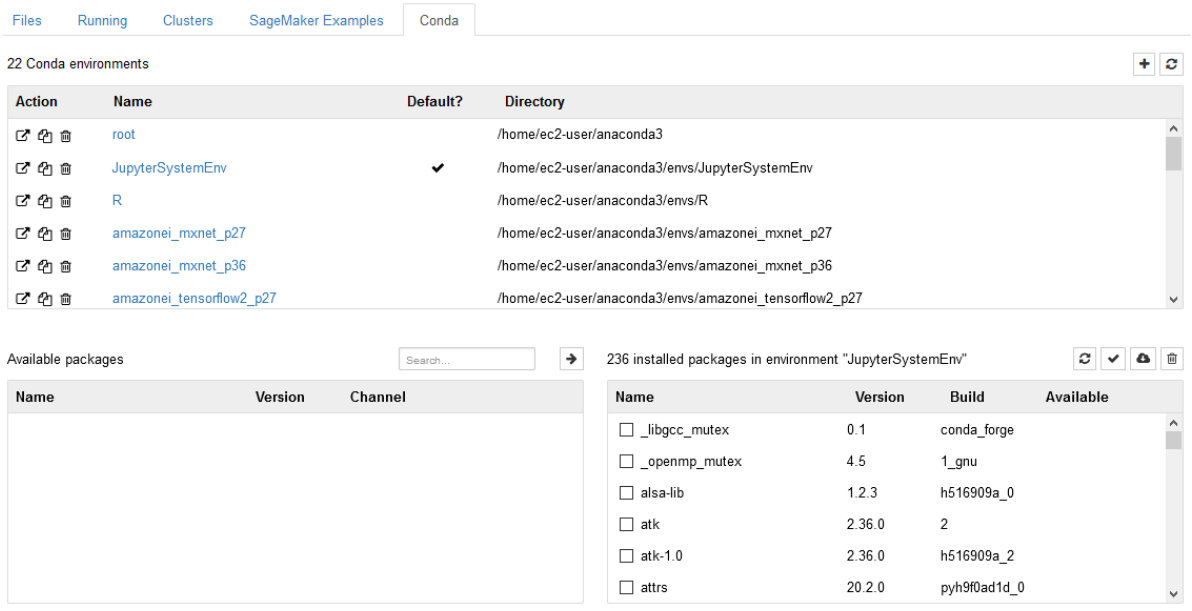


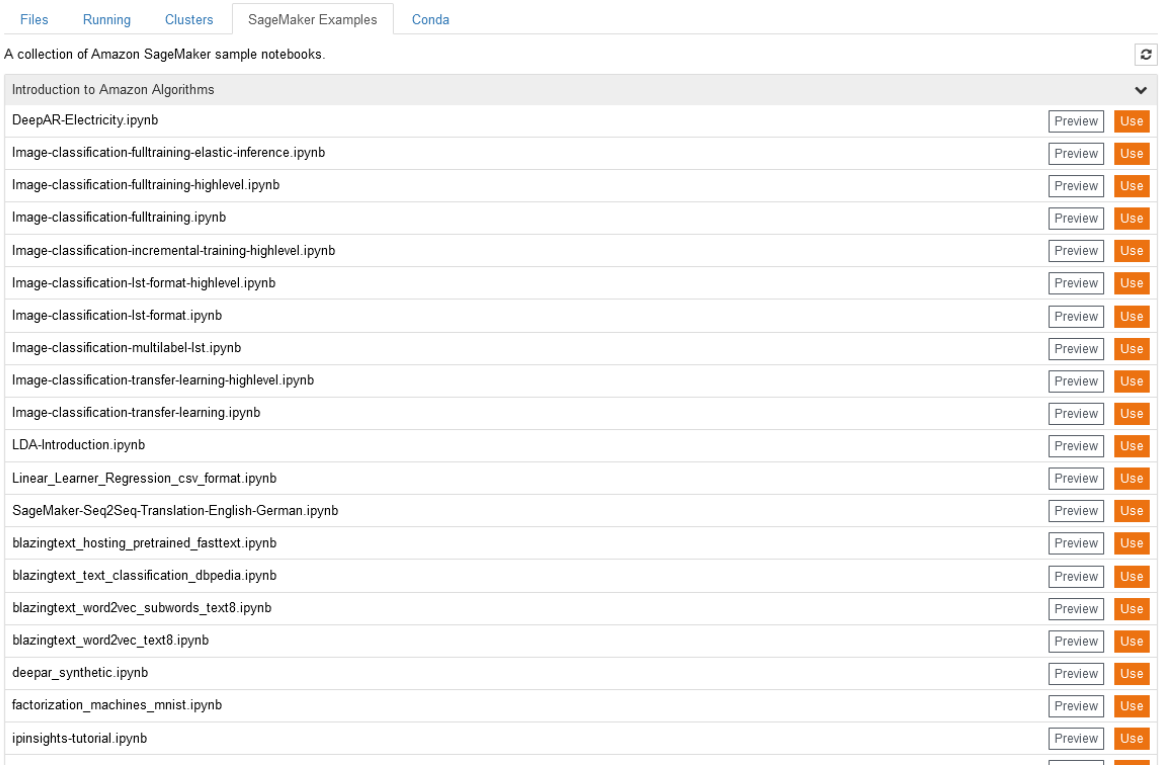
Open Jupyter

Click open Jupyter and it will direct to Jupyter. Here is Jupyter Notebook provided by Sagemaker.

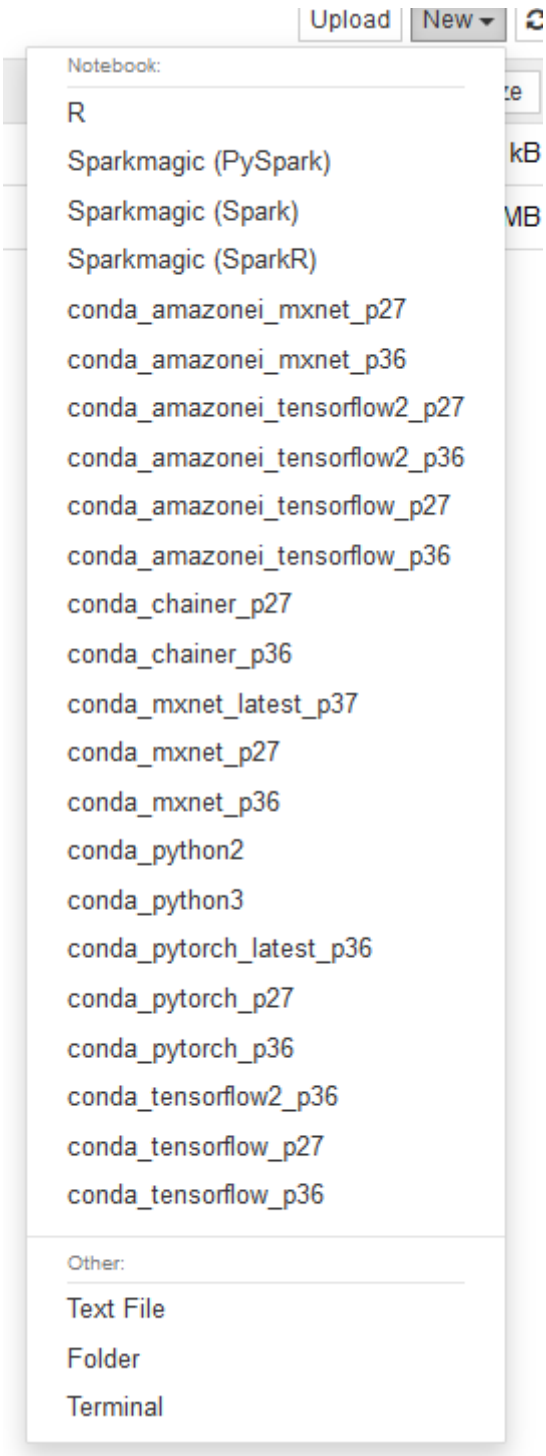


Its appearance looks same as local Jypyter Notebook. But Sagemaker provides more options. Conda tag is anaconda management page and Sagemaker Example provides online demo by AWS.

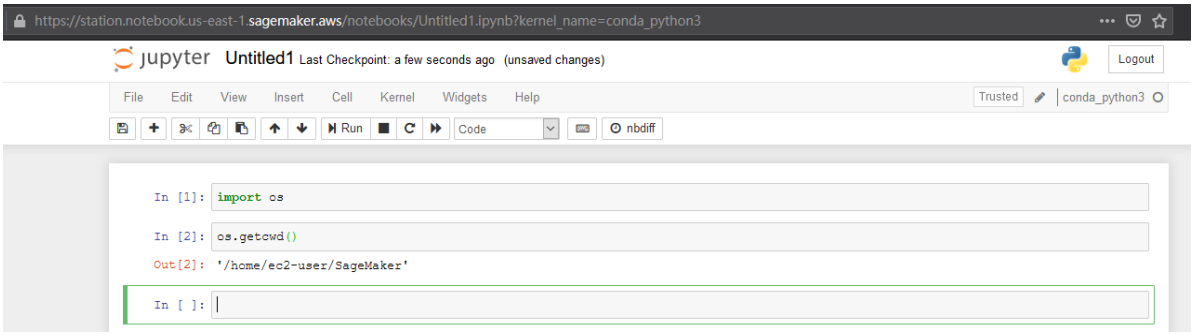




The new button has more file types, which can create Spark, tensorflow and pytorch project files. The concept in Jupyter is called kernal. Adding a Kernal can add a new file type that can be created. It is quite powerful and does not require users. The environment is still very powerful if it depends on its own configuration.

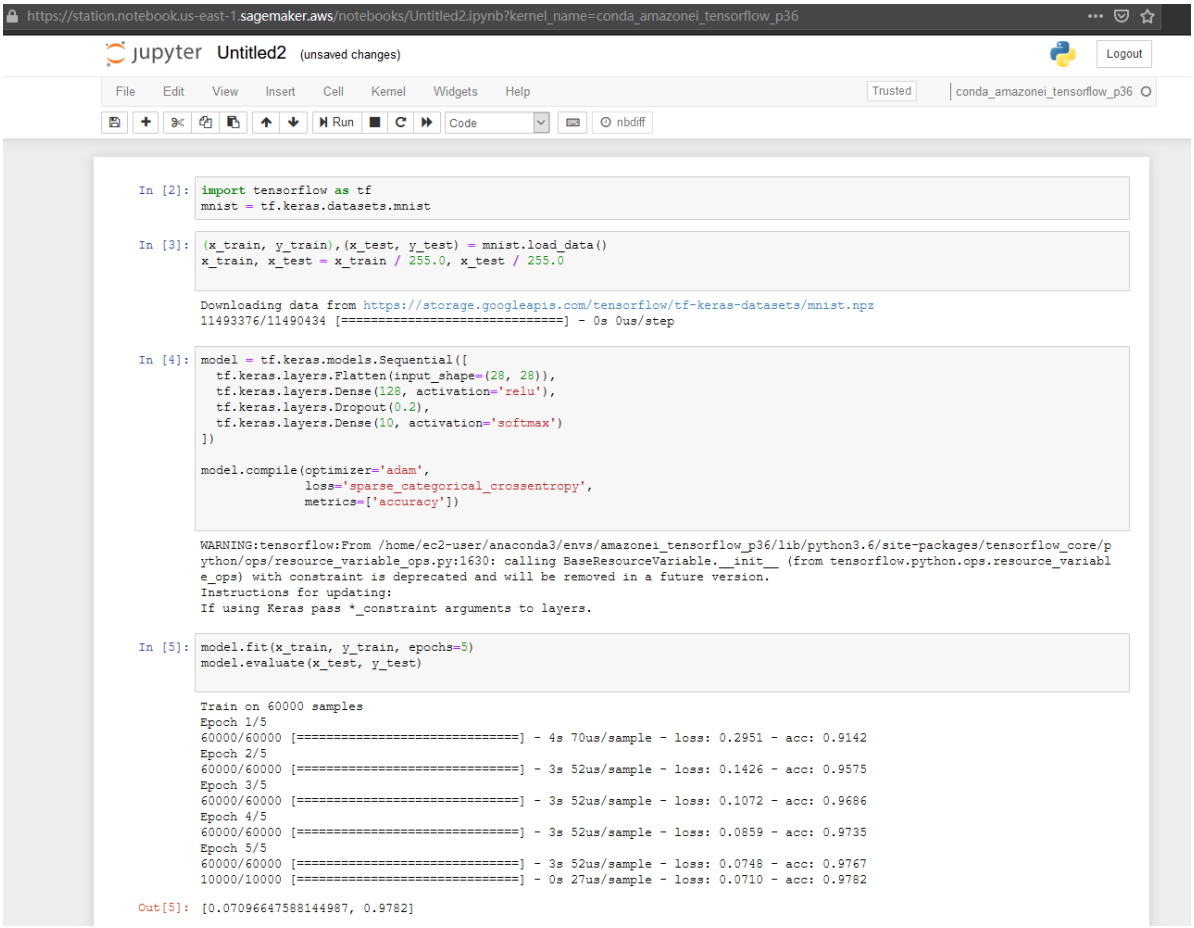


Create Python3 File



Create Tensorflow File

Put the case code on Tensorflow official website to do a simple model training.



Epoch5, the final training accuracy is 97.8%, not bad.

If you simply use Tensorflow, you don't see any advantages, because these operations can also be performed locally, but the speed of downloading the data set is a bit slow.

AWS has many great products, but Sagemaker itself is not very innovative. Many functions are the original IpythonNotebook functions, and they need to be paid. For general users, normal use is still a bit expensive. If there are a lot of training requirements, it is of course very useful. The difference between local computing power and cloud t2, medium is not a little bit different.

Similarly, Google has a tool called Colaboratory, which is actually almost the same as Jupyter Notebook, and the latter is free.

Let's try again using SageMaker's built-in algorithm, and use kmeans to make a simple classification. The following is a brief look at how to use Sagemaker for training, this part will see the advantages of cloud machine learning.

Using kmeans of Sagemaker to build a model

Prequisites and Preprocessing

Permissions and environment variables

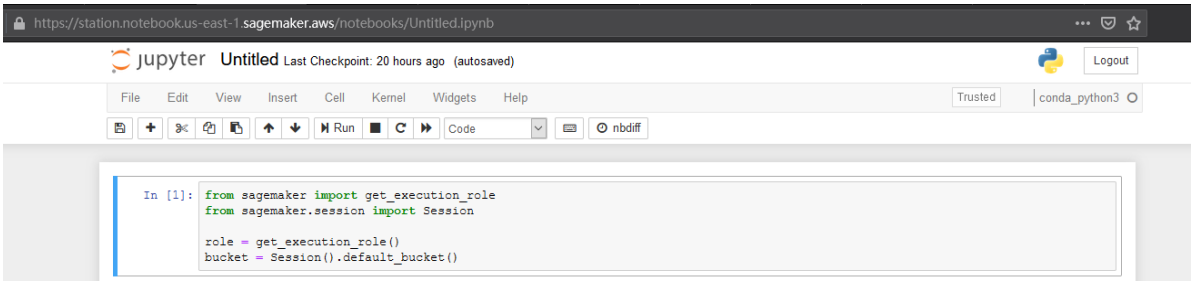
Here we set up the linkage and authentication to AWS services. There are two parts to this:

1. The roles used to give learning and hosting access to your data. Here we extract the role you created earlier for accessing your notebook. See the documentation if you want to specify a different role.

2. The S3 bucket name that you want to use for training and model data. Here we use a default in the form of `sagemaker-{region}-{AWS account Id}`, but you may specify a different one if you wish.

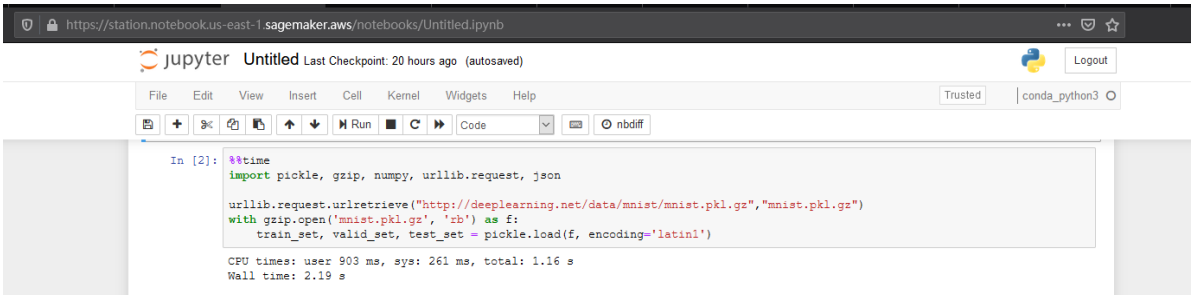
Import data

Obtain execution permissions, and create an s3bucket to store these training data. (AWS provides free 5G s3 space).

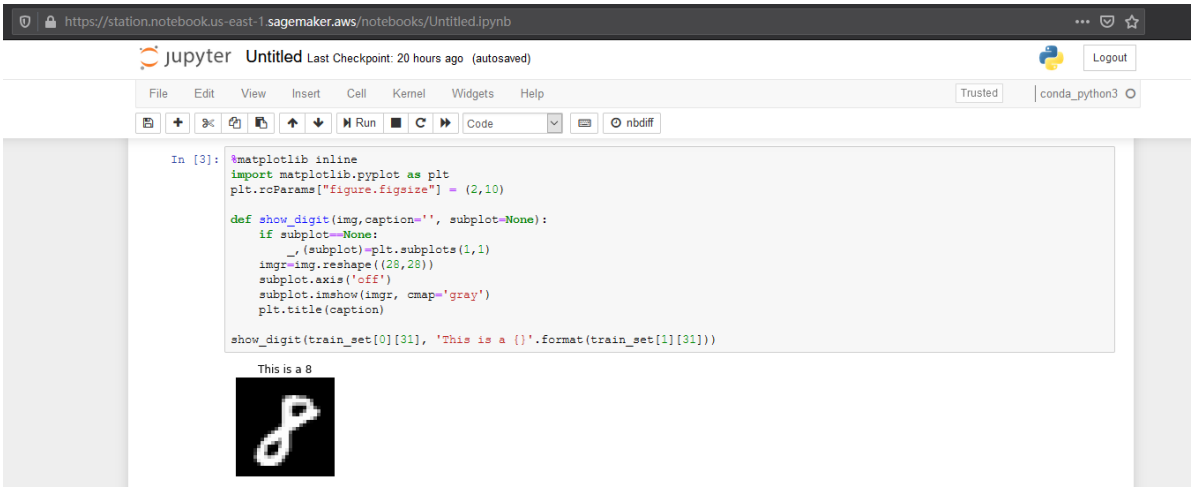


The training uses the Minist data set, 70,000 small 28x28 pixel pictures, the pictures are handwritten 0-9 numbers. The next step is data acquisition, data processing and other operations, Sagemaker quickly completed.

The imported data is divided into three categories, one is training data, the other is verification data, and the other is test data.



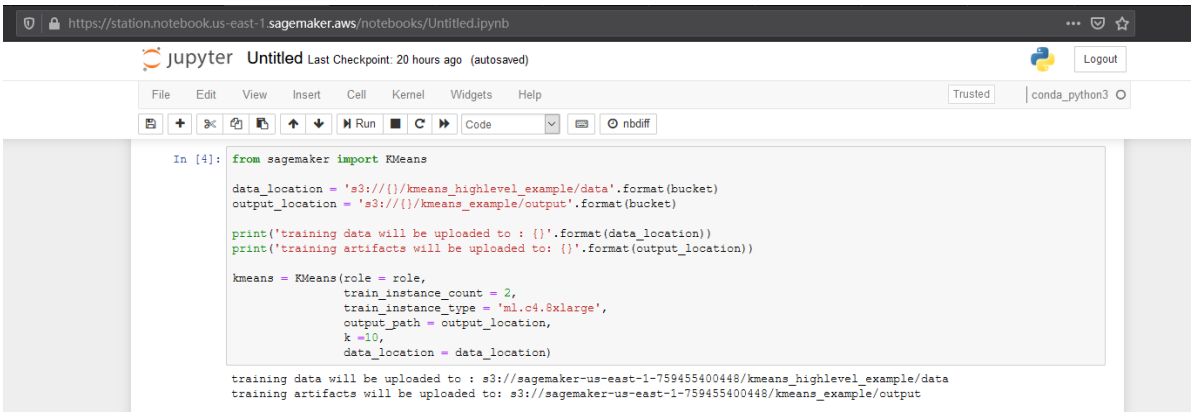
The data is imported successfully, and then in the next step, look at the composition of the data and see what the picture looks like.



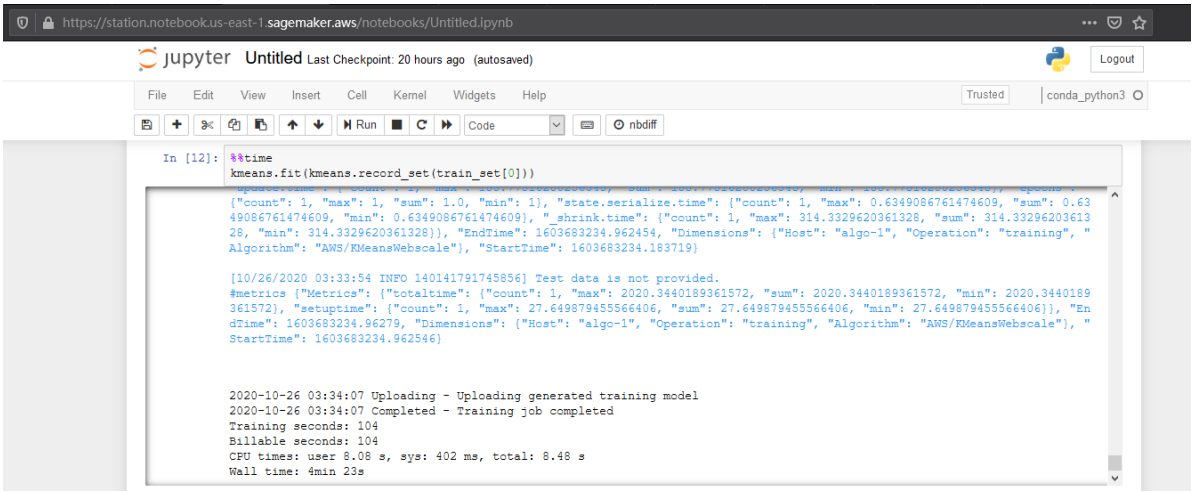
The above steps are usually the work of machine learning data processing, because the data itself is very standardized, so our technology did not do anything, because it is a clustering classification algorithm, so there is no need to tag.

Start training

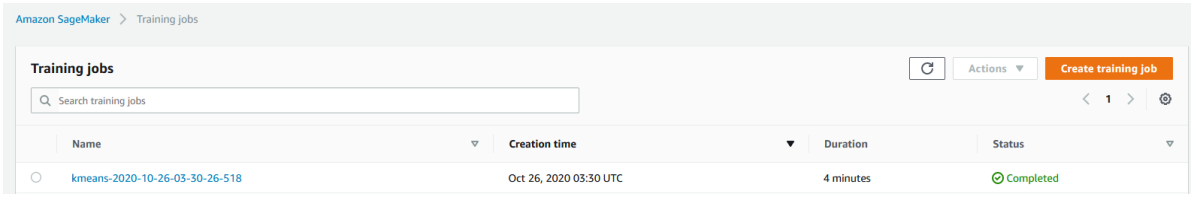
Sagemake of AWS implements its own kmeas. In this operation step, the basic framework of kmeans is constructed, two instances for training are created, and the input data and output data path are set. The value of k is set to 10. The value of k is very critical, which means that the trained model is divided into 10 clusters, and all data is in these ten clusters.



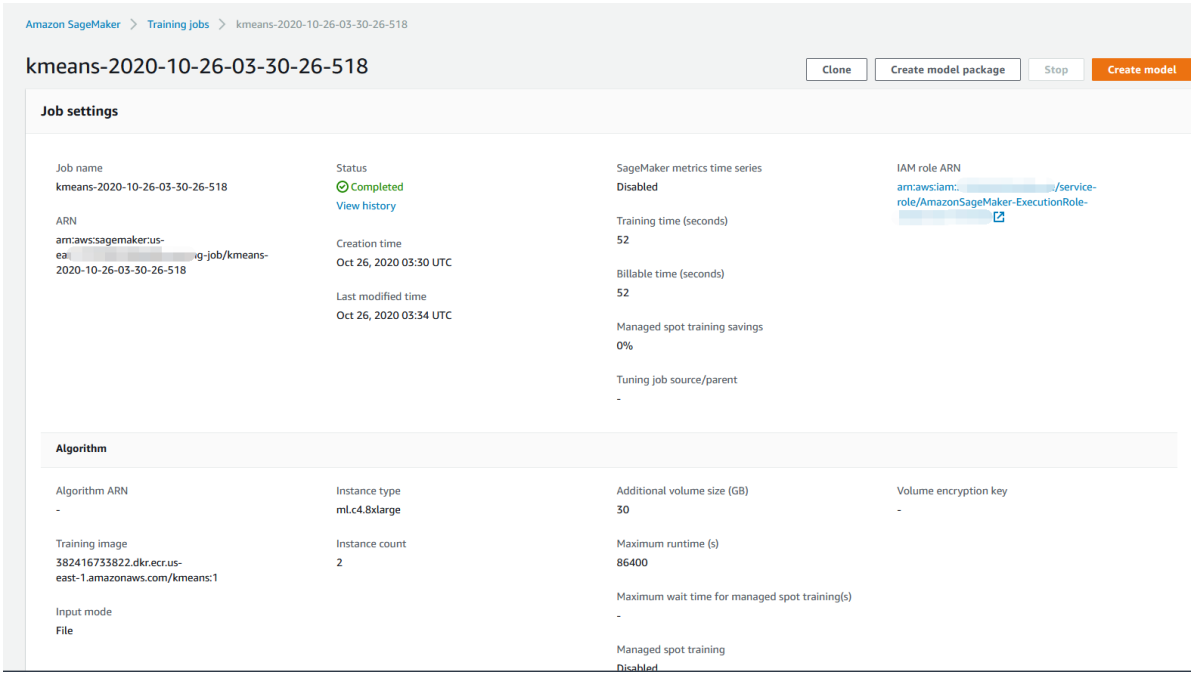
Call kmeans's .fit function for training. This .fit method should also be defined by sagemaker itself. The trained model is also uploaded, making the original simple kmens algorithm simpler.



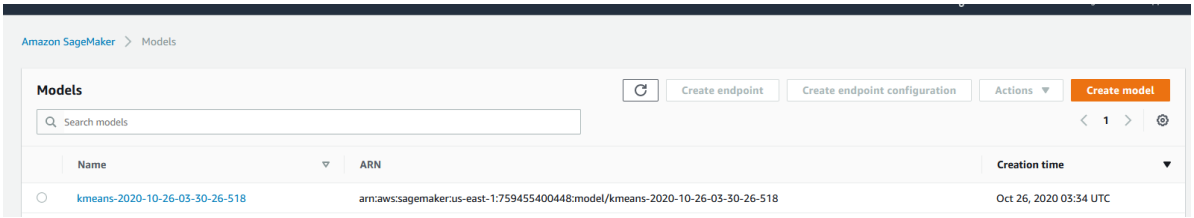
In console of Sagemaker, a training job takes 4 minutes.



Click it and we can see more information about it,

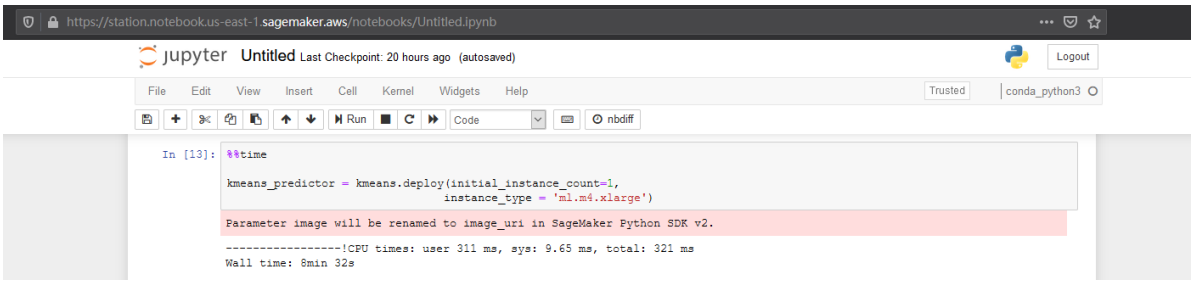


In the Sagemaker console, you can see the model you just trained.

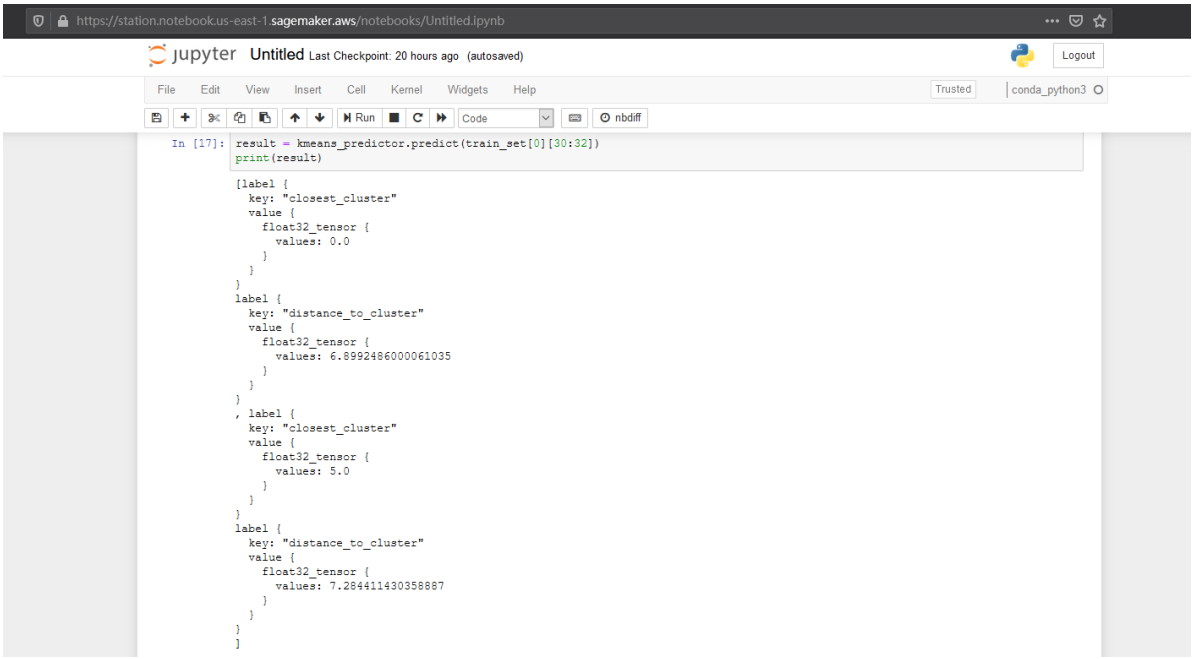


Deploy

Next step is to deploy the trained model on the m4.xlarge instance, which takes 8 minutes.

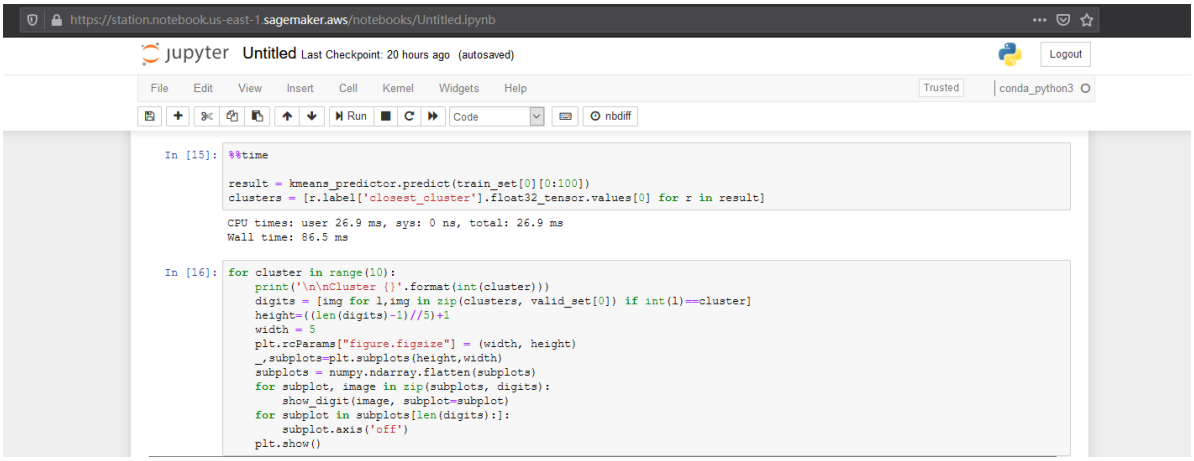


Bring a piece of data into the model, look at which cluster the data is closest to, and see how far it is from the closest cluster.



Result

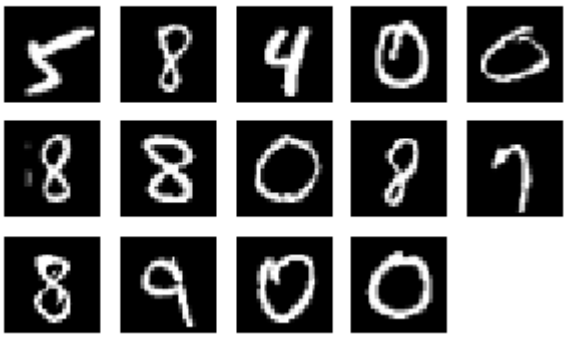
Find out 100 verification data, divide them into 0-9, 10 categories, 10 clusters. If the numbers in a cluster are the same, it proves that the recognition is not bad.



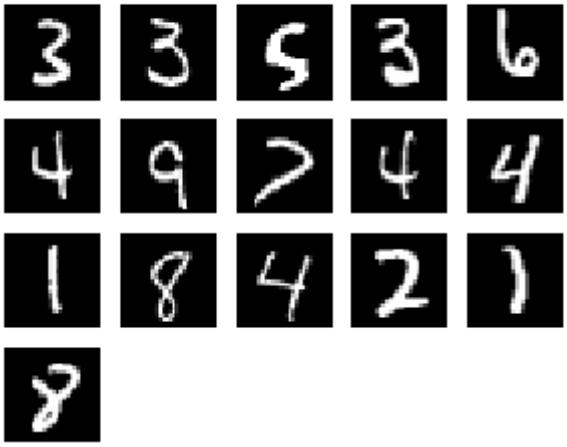
Cluster 3

Cluster 0

Cluster 2



Cluster 5



Cluster 4

End

Finally, don't forget to delete the training instance you created.

```
In [18]: print(kmeans_predictor.endpoint)

         kmeans-2020-10-26-03-30-26-518

In [ ]: import sagemaker
        sagemaker.Session().delete_endpoint(kmeans_predictor.endpoint)
```