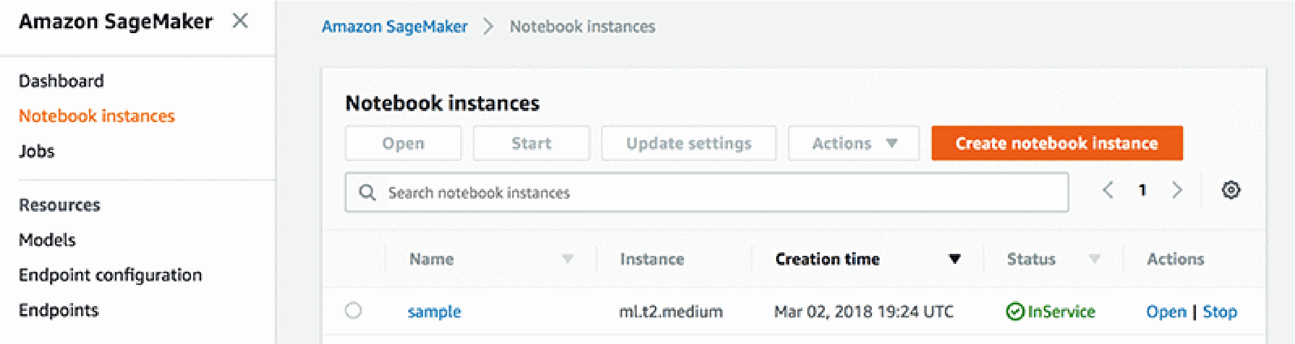
Connect Jupyter Notebook with CDP Data Lake over AWS Athena

Data analysis and management using Amazon SageMaker

The final step is to make the AudienceDB and EventDB tables definitions available in a Jupyter notebook instance of Amazon SageMaker. Jupyter notebooks are popularly used among data scientists to visualize data, perform statistical analysis, do data manipulations, and make the data ready for machine learning work.

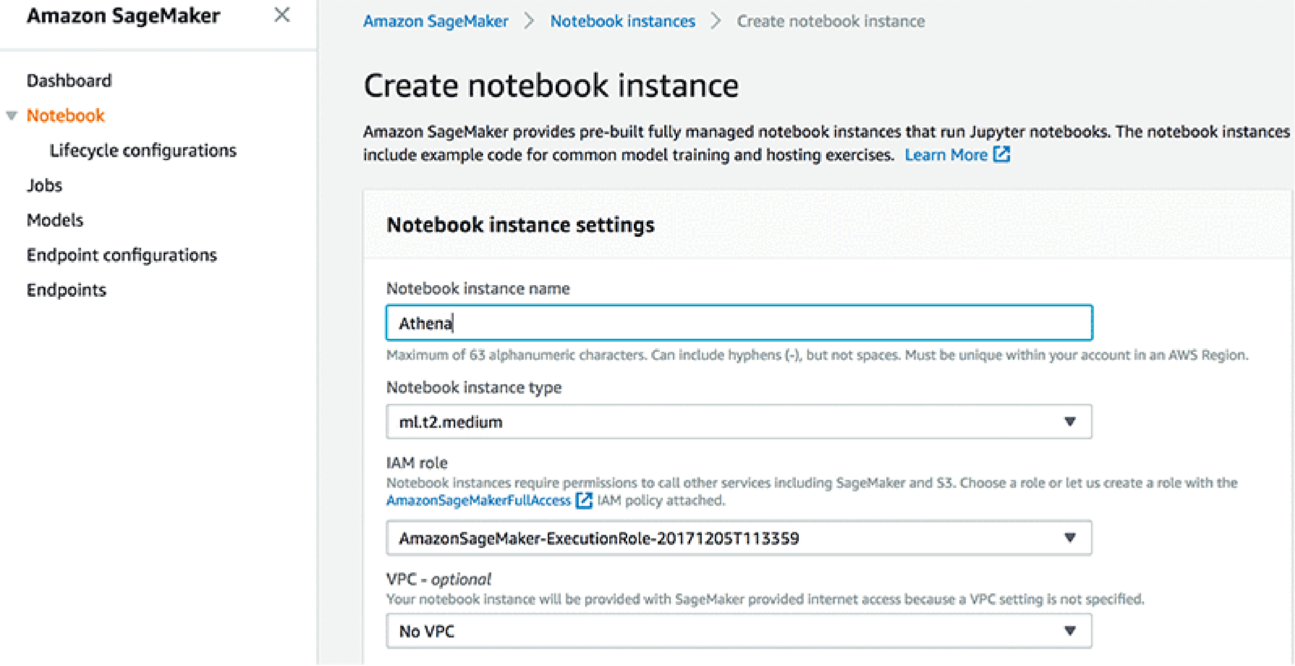
1. In the Amazon SageMaker console choose Create notebook instance.



Notebook instances

	Name	Instance	Creation time ▼	Status	Actions
○	sample	ml.t2.medium	Mar 02, 2018 19:24 UTC	InService	Open | Stop

2. Under Notebook Instance settings populate the Notebook instance name, choose an instance type, and a role for the notebook instances in Amazon SageMaker to interact with Amazon S3. The SageMaker execution role needs to have necessary permission to Athena, the S3 buckets where the data resides, and KMS if encrypted.



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: Athena

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

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.

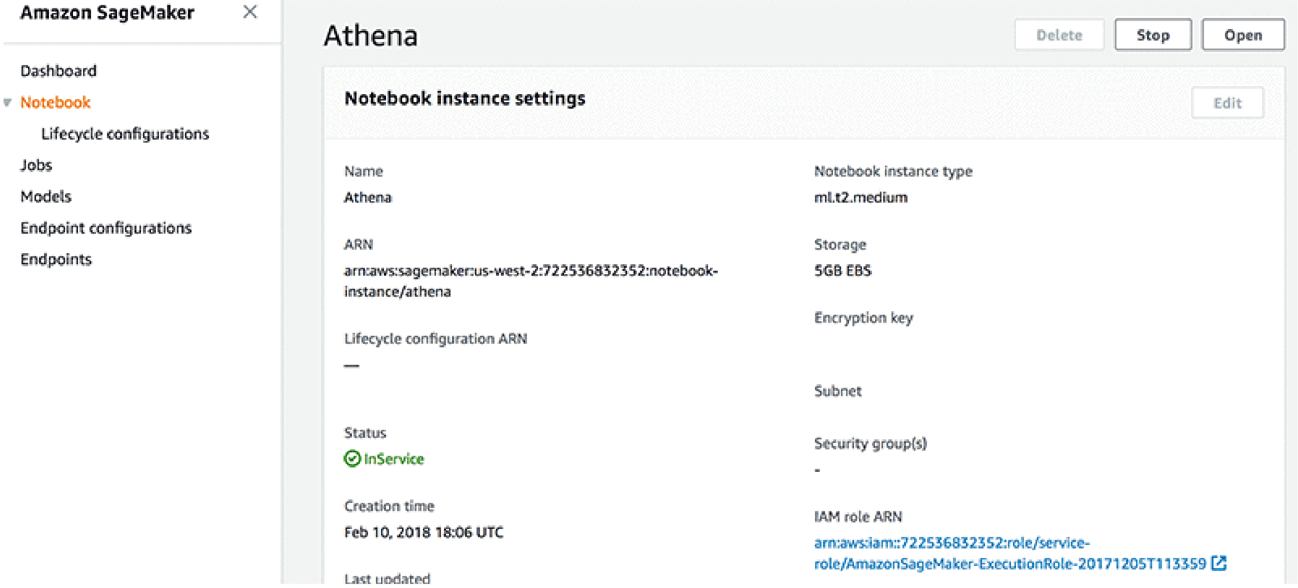
AmazonSageMaker-ExecutionRole-20171205T113359

VPC - *optional*

Your notebook instance will be provided with SageMaker provided internet access because a VPC setting is not specified.

No VPC

3. Wait for the Notebook instances to be created and the Status to change to InService.



Athena

Notebook instance settings

Name
Athena

ARN
arn:aws:sagemaker:us-west-2:722536832352:notebook-instance/athena

Lifecycle configuration ARN
—

Status
InService

Creation time
Feb 10, 2018 18:06 UTC

Last updated

Notebook instance type
ml.t2.medium

Storage
5GB EBS

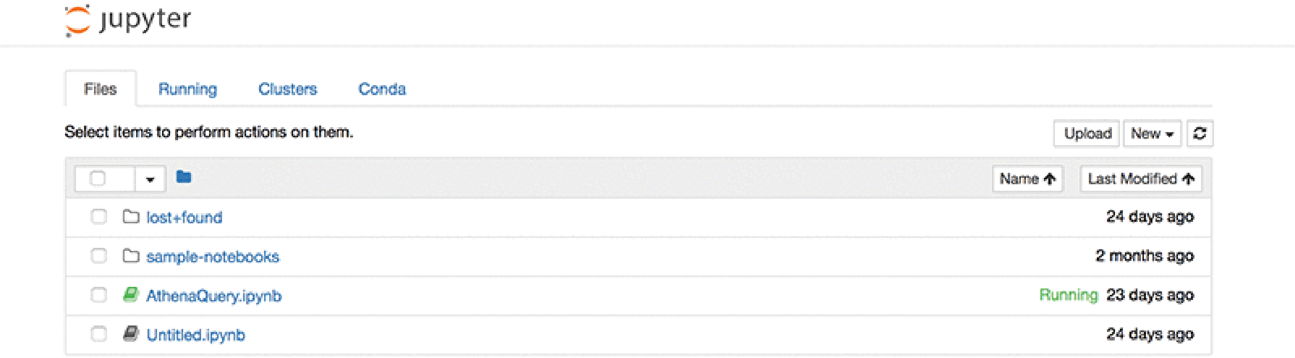
Encryption key

Subnet

Security group(s)
-

IAM role ARN
arn:aws:iam::722536832352:role/service-role/AmazonSageMaker-ExecutionRole-20171205T113359

4. Choose the Open link, which will open the notebook interface in a separate browser window.

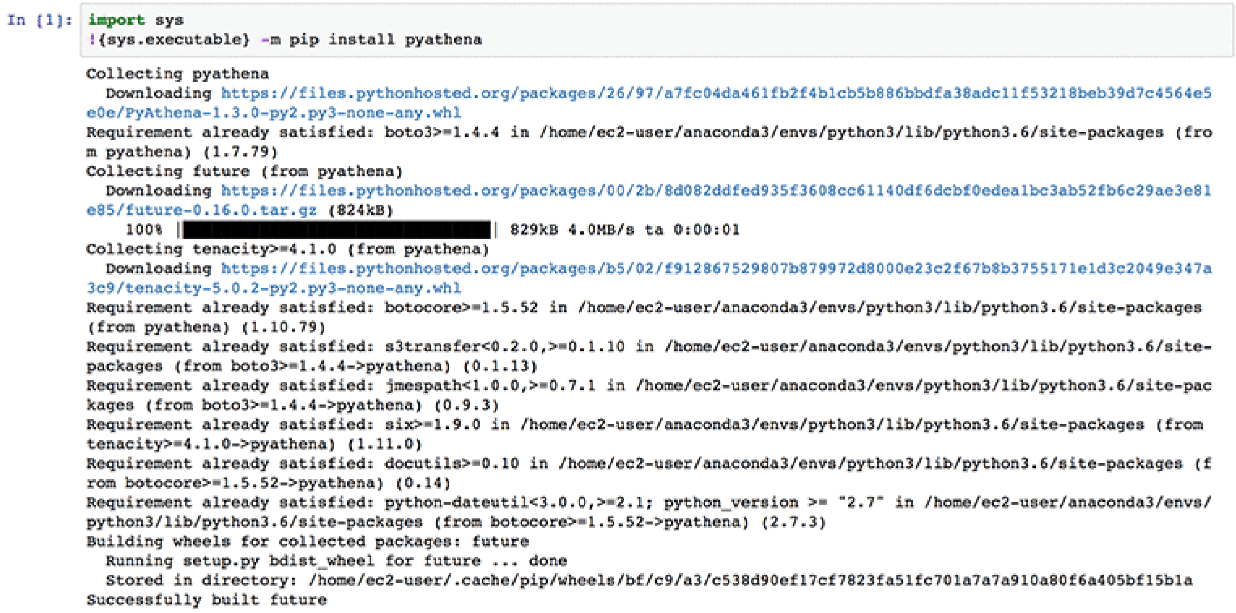


Select items to perform actions on them.

Name		Last Modified
lost+found		24 days ago
sample-notebooks		2 months ago
AthenaQuery.ipynb	Running	23 days ago
Untitled.ipynb		24 days ago

5. Click new to create a new notebook in Jupyter. Amazon SageMaker provides several kernels for Jupyter including support for Python 2 and 3, MXNet, TensorFlow, and PySpark. Choose Python as the kernel for this exercise as it comes with the Pandas library built in. Within the notebook, execute the following commands to install the Athena JDBC driver. PyAthena is a Python DB API 2.0 (PEP 249) compliant client for the Amazon Athena JDBC driver.

```
import sys
!{sys.executable} -m pip install PyAthena
```



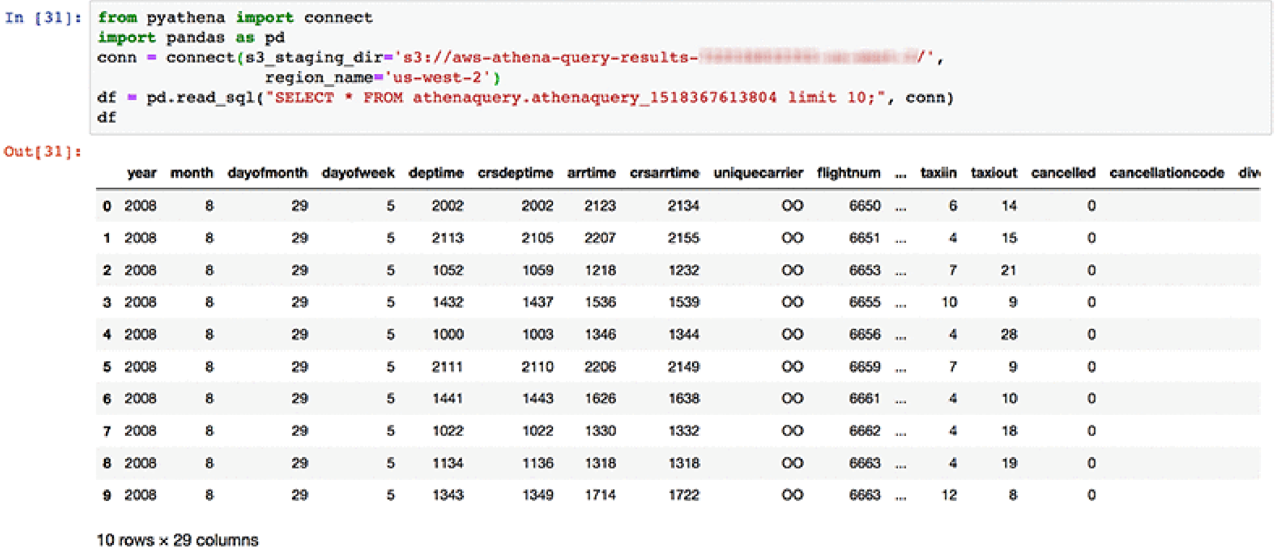
```
In [1]: import sys
        !{sys.executable} -m pip install pyathena

Collecting pyathena
  Downloading https://files.pythonhosted.org/packages/26/97/a7fc04da461fb2f4b1cb5b886bbdfa38adc11f53218beb39d7c4564e5
e0e/PyAthena-1.3.0-py2.py3-none-any.whl
Requirement already satisfied: boto3>=1.4.4 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (fro
m pyathena) (1.7.79)
Collecting future (from pyathena)
  Downloading https://files.pythonhosted.org/packages/00/2b/8d082ddfed935f3608cc61140df6dcbf0edea1bc3ab52fb6c29ae3e81
e85/future-0.16.0.tar.gz (824kB)
    100% |████████████████████████████████| 829kB 4.0MB/s ta 0:00:01
Collecting tenacity>=4.1.0 (from pyathena)
  Downloading https://files.pythonhosted.org/packages/b5/02/f912867529807b879972d8000e23c2f67b8b3755171e1d3c2049e347a
3c9/tenacity-5.0.2-py2.py3-none-any.whl
Requirement already satisfied: botocore>=1.5.52 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages
(from pyathena) (1.10.79)
Requirement already satisfied: s3transfer<0.2.0,>=0.1.10 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-
packages (from boto3>=1.4.4->pyathena) (0.1.13)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-pac
kages (from boto3>=1.4.4->pyathena) (0.9.3)
Requirement already satisfied: six>=1.9.0 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (from
tenacity>=4.1.0->pyathena) (1.11.0)
Requirement already satisfied: docutils>=0.10 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (f
rom botocore>=1.5.52->pyathena) (0.14)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1; python_version >= "2.7" in /home/ec2-user/anaconda3/envs/
python3/lib/python3.6/site-packages (from botocore>=1.5.52->pyathena) (2.7.3)
Building wheels for collected packages: future
  Running setup.py bdist_wheel for future ... done
  Stored in directory: /home/ec2-user/.cache/pip/wheels/bf/c9/a3/c538d90ef17cf7823fa51fc701a7a7a910a80f6a405bf15b1a
Successfully built future
```

6. After the Athena driver is installed, you can use the JDBC connection to connect to Athena and populate the Pandas data frames. For data scientists, working with data is typically divided into multiple stages: munging and cleaning data, analyzing / modeling it, then organizing the results of the analysis into a form suitable for plotting or tabular display. Pandas is the ideal tool for all of these tasks.

```
from pyathena import connect
import pandas as pd
conn = connect(s3_staging_dir='<ATHENA QUERY RESULTS LOCATION>',
               region_name='<YOUR REGION, for example, us-west-2>')

df = pd.read_sql("SELECT * FROM athenaquery.<YOUR TABLE NAME> limit 8;", conn)
df
```



```
In [31]: from pyathena import connect
         import pandas as pd
         conn = connect(s3_staging_dir='s3://aws-athena-query-results-[redacted]/',
                        region_name='us-west-2')
         df = pd.read_sql("SELECT * FROM athenaquery.athenaquery_1518367613804 limit 10;", conn)
         df
```

Out[31]:

	year	month	dayofmonth	dayofweek	deptime	crsdeptime	arrtime	crsarrtime	uniquecarrier	flightnum	...	taxiin	taxiout	cancelled	cancellationcode	div
0	2008	8	29	5	2002	2002	2123	2134	OO	6650	...	6	14	0		
1	2008	8	29	5	2113	2105	2207	2155	OO	6651	...	4	15	0		
2	2008	8	29	5	1052	1059	1218	1232	OO	6653	...	7	21	0		
3	2008	8	29	5	1432	1437	1536	1539	OO	6655	...	10	9	0		
4	2008	8	29	5	1000	1003	1346	1344	OO	6656	...	4	28	0		
5	2008	8	29	5	2111	2110	2206	2149	OO	6659	...	7	9	0		
6	2008	8	29	5	1441	1443	1626	1638	OO	6661	...	4	10	0		
7	2008	8	29	5	1022	1022	1330	1332	OO	6662	...	4	18	0		
8	2008	8	29	5	1134	1136	1318	1318	OO	6663	...	4	19	0		
9	2008	8	29	5	1343	1349	1714	1722	OO	6663	...	12	8	0		

10 rows × 29 columns

Conclusion

The solution described in this blog post provides an automated way to catalog the incoming data as it comes into the data store, and it provides the ability to query the data for data manipulation and analysis. In addition, this scenario sets the stage for building more ML models through feature engineering, training, and scoring to gain more insights into your data and deliver significant business outcomes.