How to connect to Denodo from Amazon SageMaker

Revision 20220127

Denodo Inc.: 525 University Avenue, Suite 31, Palo Alto, CA 94301. USA www.denodo.com

CONTENTS

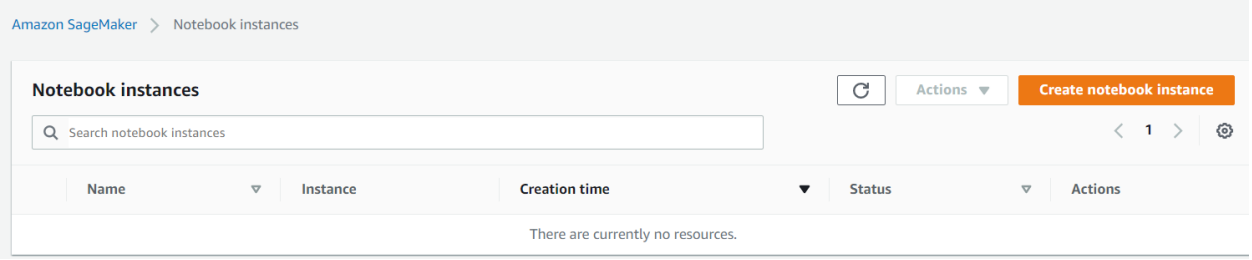
1 CONTENT

This document explains the steps involved in connecting to Denodo Virtual Dataport from AWS SageMaker.
Amazon SageMaker is a fully managed service that provides developers and data scientists with the ability to prepare, build, train, and deploy machine learning (ML) models quickly.

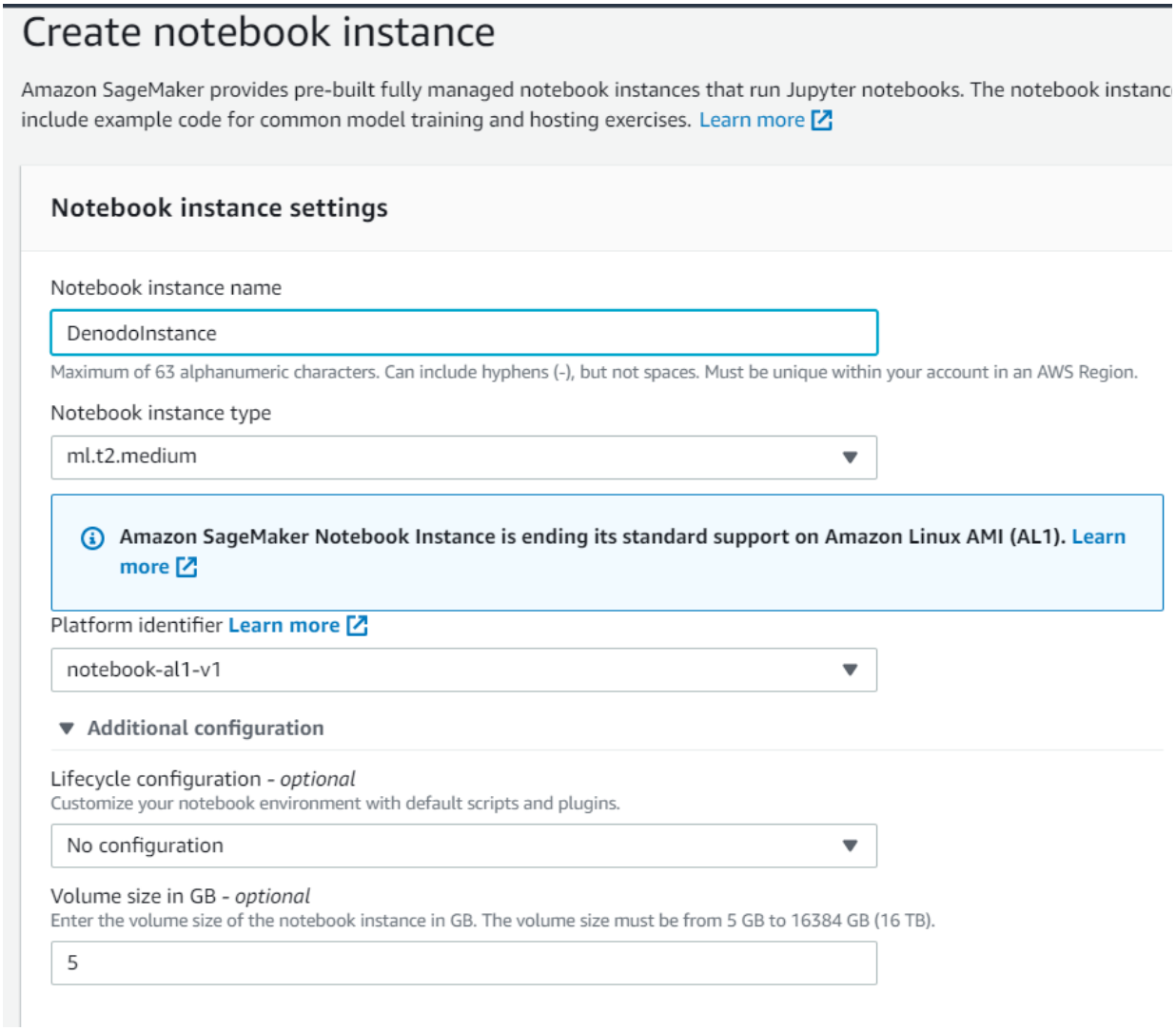
2 CREATING A NOTEBOOK INSTANCE

A notebook instance is a machine learning Amazon EC2 compute instance that runs the Jupyter Notebook App. Notebook instances are used to create and manage Jupyter notebooks for preprocessing data and to train and deploy machine learning models.
As a first step we are going to create a notebook instance.

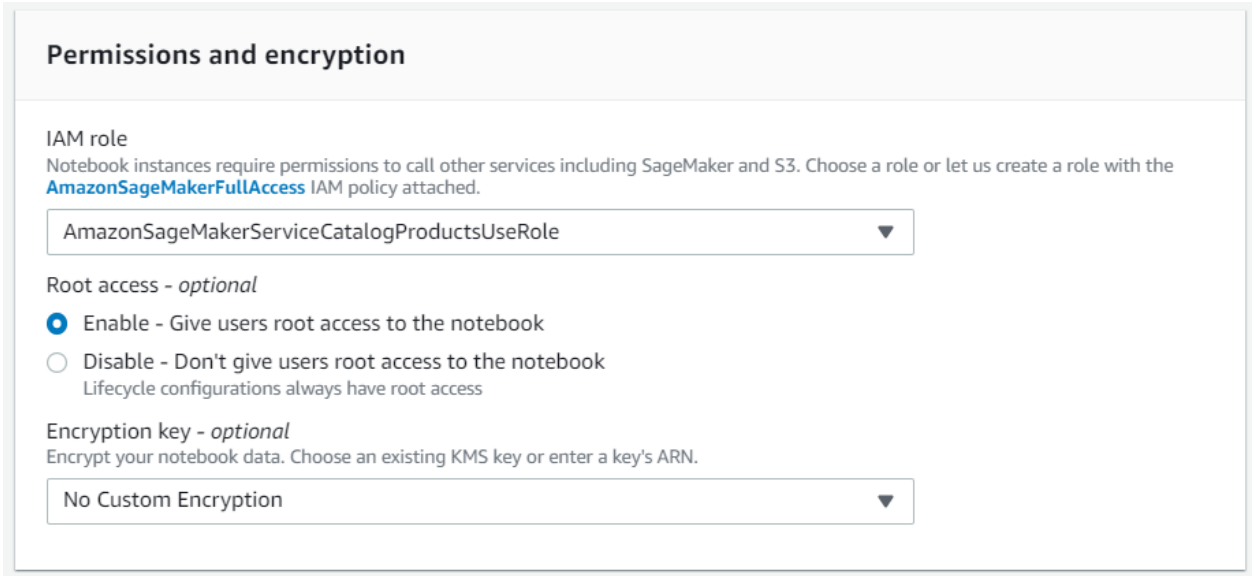
- Navigate to **`Amazon SageMaker > Notebook Instances`**.
- Click on **`Create notebook instance`**.



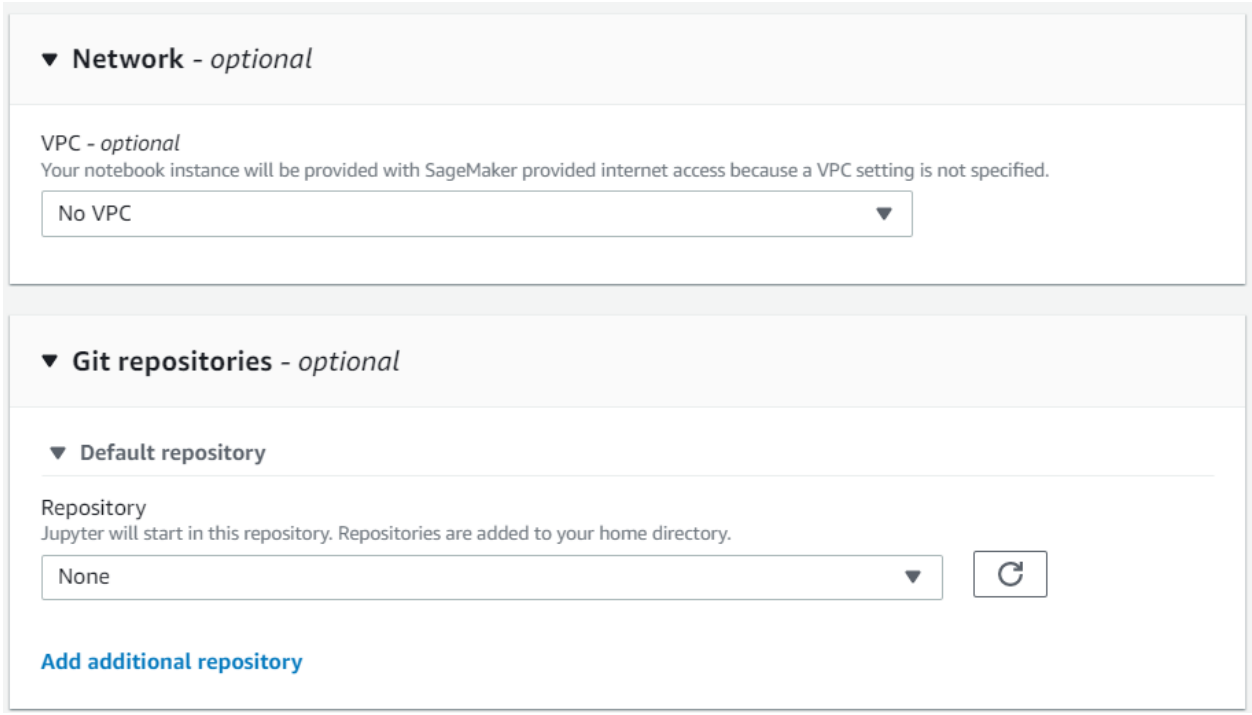
- Name the instance, for example **`DenodoInstance`**.
- Choose the required instance type from the available options, for this example, an `ml.t2.medium` instance is used.



- You may leave the `Additional configuration` section options as it is as they are optional.
- Next, configure the **`Permissions and encryption`** for the notebook instance.
- Choose the IAM Role that has permissions to access other services like S3, EC2 etc. Note that you can create a role or let SageMaker create one for you with the **`AmazonSageMakerFullAccess`** IAM policy attached.

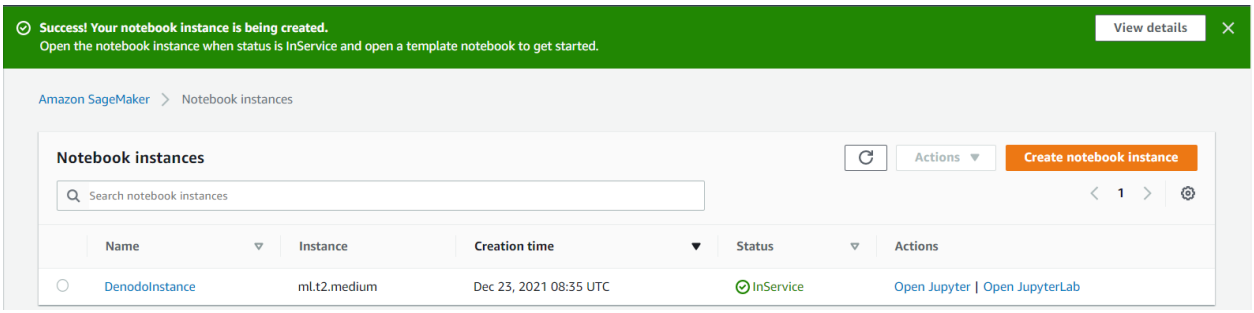


- Based on the requirement, we may decide whether to give root access for the users accessing the notebook.
- Scroll down to configure **`Network`** options. SageMaker allows you to configure the instance within a VPC. This will be useful in case the Notebook instance needs to be accessed only within the VPC. Note that if the VPC is configured to access the internet, then the instance will inherit it. For this example, we choose **`No VPC`** and SageMaker will provide internet access directly to the instance.



- The **`Git repositories`** section allows the instance to start Jupyter in the given repository. Leave this section blank.

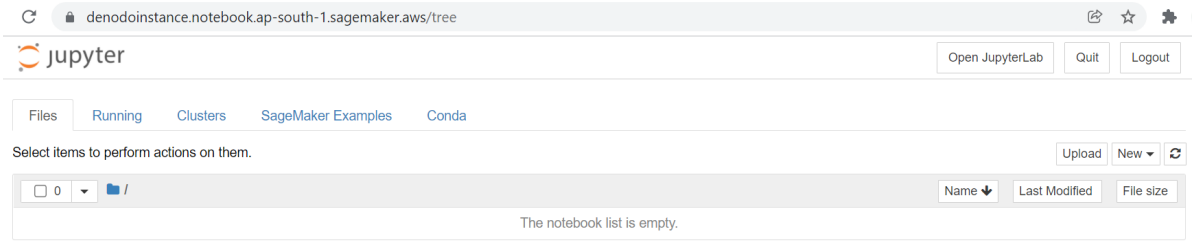
- Click on **`Create Notebook Instance`** to start the instance creation.



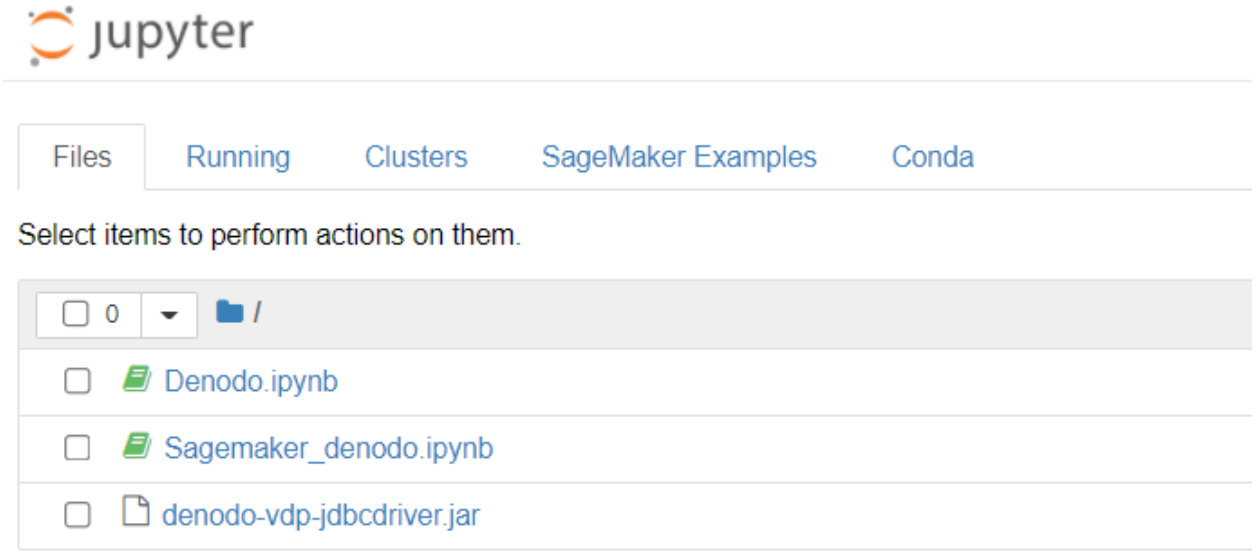
3 ACCESS JUPYTER / JUPYTERLAB

Amazon SageMaker creates Jupyter Notebook instances from which we can create notebooks and store them. AWS offers several pre-built notebooks for python libraries for AI/ML workloads. For this example, we will create a simple notebook and install the required Python libraries for establishing a connection to Denodo.

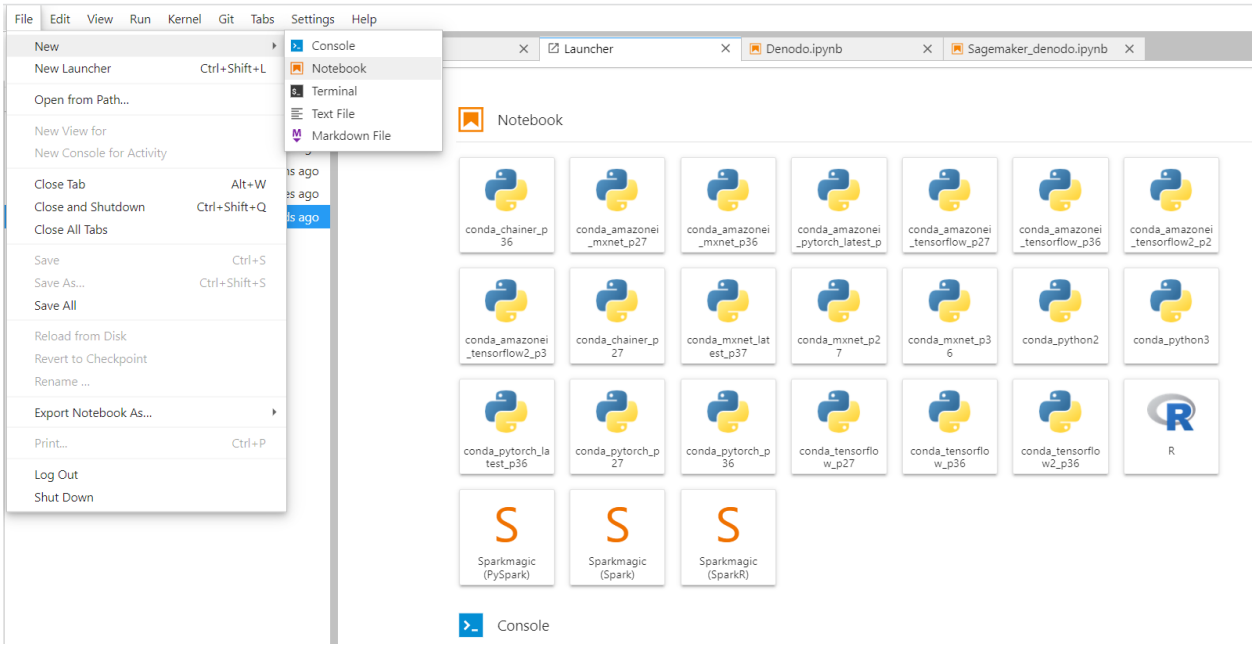
This **document** explains the list of required libraries required and we must install them by accessing the Terminal of the instance. We do have an option to install the libraries from the notebooks, but the Terminal option offers more control for the users and offers persistence.



From the Jupyter page, click on the **Upload** button to upload Denodo's JDBC driver.



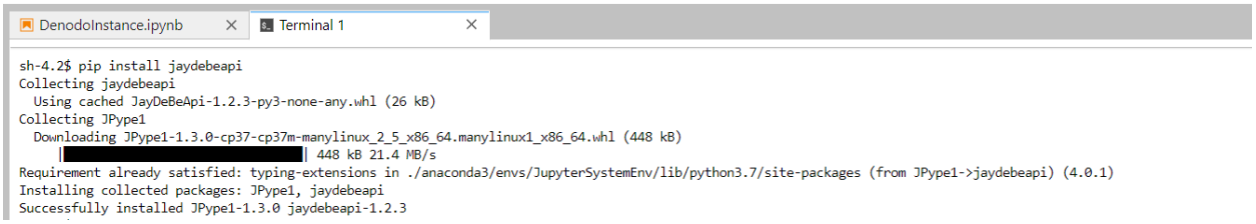
Once the JAR file has been uploaded, create a new Notebook by clicking "**`File > New > Notebook`**", from Launcher directly or choose **`conda_python3`** for running a python pre-installed notebook.


File
Edit
View
Run
Kernel
Git
Tabs
Settings
Help
New
New Launcher Ctrl+Shift+L
Open from Path...
New View for
New Console for Activity
Close Tab Alt+W
Close and Shutdown Ctrl+Shift+Q
Close All Tabs
Save Ctrl+S
Save As... Ctrl+Shift+S
Save All
Reload from Disk
Revert to Checkpoint
Rename ...
Export Notebook As...
Print... Ctrl+P
Log Out
Shut Down
Console
Notebook
Terminal
Text File
Markdown File
Launcher
Denodo.ipynb
Sagemaker_denodo.ipynb
Notebook
conda_chainer_p36
conda_amazonei_mxnet_p27
conda_amazonei_mxnet_p36
conda_amazonei_pytorch_latest_p
conda_amazonei_tensorflow_p27
conda_amazonei_tensorflow_p36
conda_amazonei_tensorflow2_p2
conda_amazonei_tensorflow2_p3
conda_chainer_p27
conda_mxnet_latest_p37
conda_mxnet_p27
conda_mxnet_p36
conda_python2
conda_python3
conda_pytorch_latest_p36
conda_pytorch_p27
conda_pytorch_p36
conda_tensorflow_p27
conda_tensorflow_p36
conda_tensorflow2_p36
R
Sparkmagic (PySpark)
Sparkmagic (Spark)
Sparkmagic (SparkR)
Console

4 INSTALL JAYDEBEAPI

Once a notebook instance has been created, the next step is to install the jaydebeapi python library for establishing a connection to Denodo.
To do so, navigate to **`File > New > Terminal`** and run the following command:

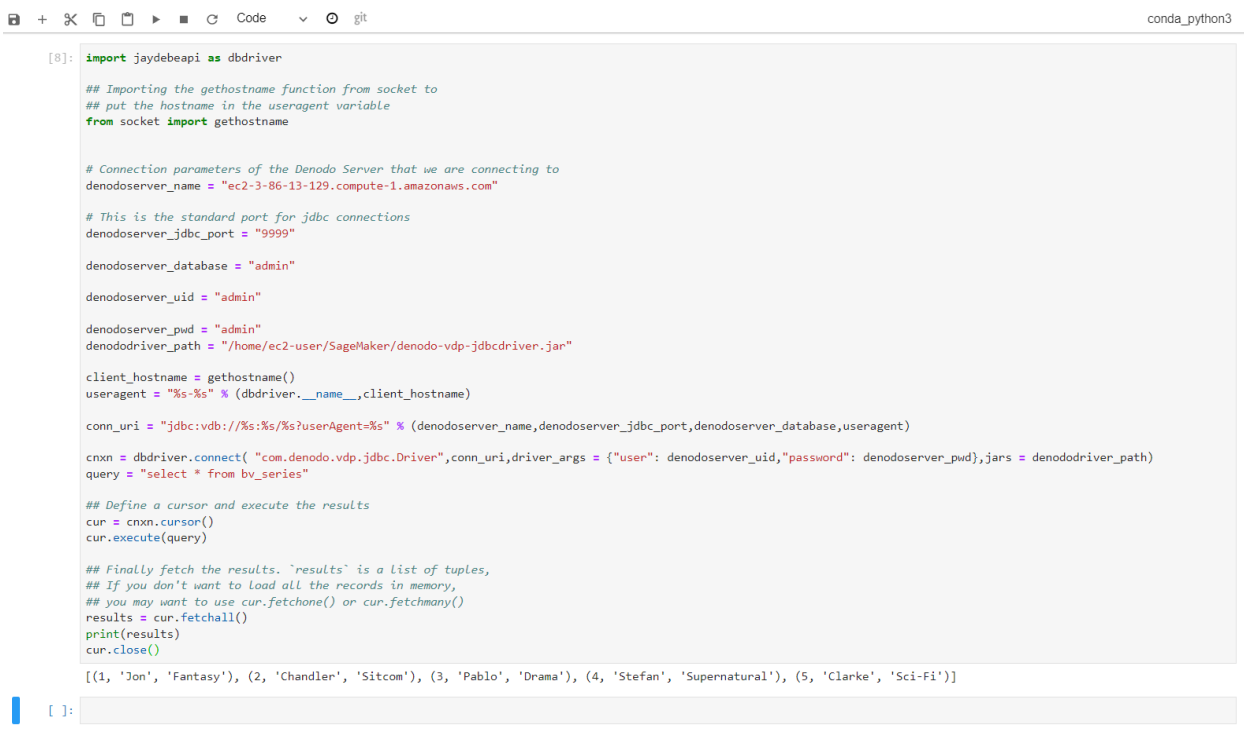
```
pip install jaydebeapi
```



Using the Terminal option, the library will be persisted on restart whereas other options will lose the library if the notebook is stopped/restarted. Once installed, you can run a `pip list` command that retrieves the list of libraries installed on the instance.

5 EXECUTE THE CODE

On the next cell, use the python code as mentioned below to connect to Denodo and list the results of querying a view (bv_series).

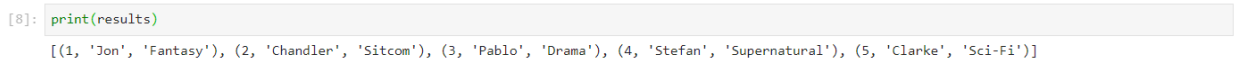


```
[8]: import jaydebeapi as dbdriver

## Importing the gethostname function from socket to
## put the hostname in the useragent variable
from socket import gethostname


# Connection parameters of the Denodo Server that we are connecting to
denodoserver_name = "ec2-3-86-13-129.compute-1.amazonaws.com"

# This is the standard port for jdbc connections
denodoserver_jdbc_port = "9999"

denodoserver_database = "admin"

denodoserver_uid = "admin"

denodoserver_pwd = "admin"
denododriver_path = "/home/ec2-user/SageMaker/denodo-vdp-jdbcdriver.jar"

client_hostname = gethostname()
useragent = "%s-%s" % (dbdriver.__name__,client_hostname)

conn_uri = "jdbc:vdb://%s:%s/%s?userAgent=%s" % (denodoserver_name,denodoserver_jdbc_port,denodoserver_database,useragent)

cnxn = dbdriver.connect( "com.denodo.vdp.jdbc.Driver",conn_uri,driver_args = {"user": denodoserver_uid,"password": denodoserver_pwd},jars = denododriver_path)
query = "select * from bv_series"

## Define a cursor and execute the results
cur = cnxn.cursor()
cur.execute(query)

## Finally fetch the results. `results` is a list of tuples,
## If you don't want to load all the records in memory,
## you may want to use cur.fetchone() or cur.fetchmany()
results = cur.fetchall()
print(results)
cur.close()

[(1, 'Jon', 'Fantasy'), (2, 'Chandler', 'Sitcom'), (3, 'Pablo', 'Drama'), (4, 'Stefan', 'Supernatural'), (5, 'Clarke', 'Sci-Fi')]

[ ]:
```

```
import jaydebeapi as dbdriver

## Importing the gethostname function from socket to
## put the hostname in the useragent variable
from socket import gethostname

# Connection parameters of the Denodo Server that we are connecting to
denodoserver_name = "<hostname>"
denodoserver_jdbc_port = "9999"
denodoserver_database = "admin"
denodoserver_uid = "<username>"
denodoserver_pwd = "<password>"

denododriver_path = "/home/ec2-user/SageMaker/denodo-vdp-jdbcdriver.jar"

client_hostname = gethostname()
useragent = "%s-%s" % (dbdriver.__name__,client_hostname)
```

```
conn_uri          =          "jdbc:vdb://%s:%s/%s?userAgent=%s"          %
(denodoserver_name,denodoserver_jdbc_port,denodoserver_database,useragent)

cnxn  =  dbdriver.connect("com.denodo.vdp.jdbc.Driver",conn_uri,driver_args  =
{"user": denodoserver_uid,"password": denodoserver_pwd},
jars = denododriver_path)

query = "select * from bv_series"

## Define a cursor and execute the results
cur = cnxn.cursor()
cur.execute(query)

## Finally fetch the results. `results` is a list of tuples,
## If you don't want to load all the records in memory,
## you may want to use cur.fetchone() or cur.fetchmany()
results = cur.fetchall()
print(results)
##To close the cursor after execution to ensure connections are closed
cur.close()
```

Replace the placeholders for `denodoserver_name`, `denodoserver_uid`, `denodoserver_pwd` fields accordingly. Notice the path of the JDBC driver has been given as "`/home/ec2-user/SageMaker/denodo-vdp-jdbcdriver.jar`". The location where the SageMaker stores the files uploaded to the Notebook instance can be found using the Terminal of the instance, `DenodoInstance` in this example.
After executing the cell, the print command is used to fetch the results of the executed query. Note that we are using a `fetchall()` cursor to store all the results in memory.



This way the views in Denodo Platform can be retrieved from AWS SageMaker and can also be used with various pre-built python libraries for Machine Learning workloads and AI use cases.

6 REFERENCES

Using Notebooks for Data Science with Denodo
How to connect to Denodo from Python - a starter for Data Scientists
Denodo in Data Science and Machine Learning Projects