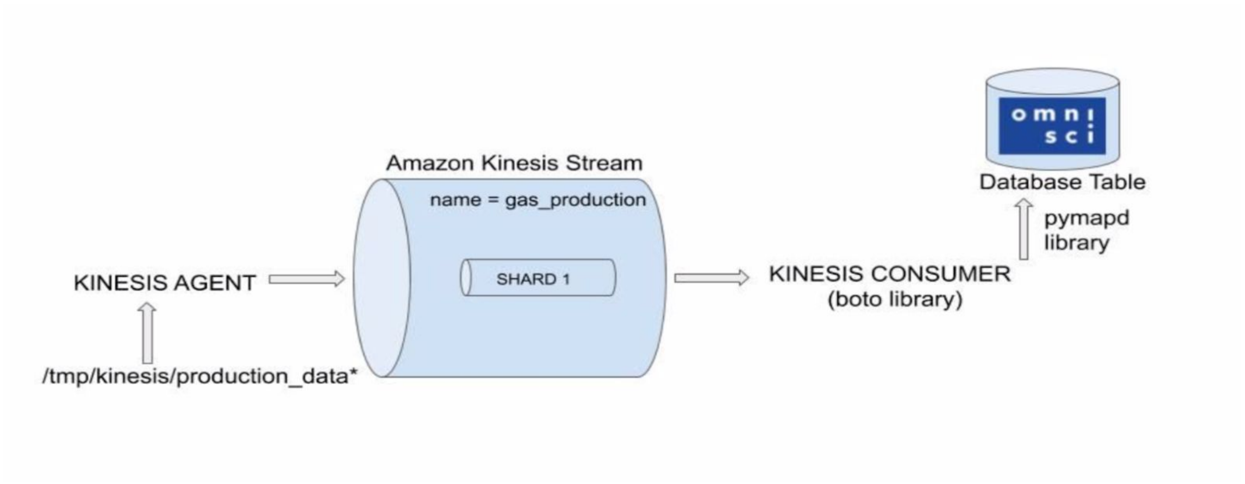
AWS Kinesis - OmniSci Worksheet



Setup Kinesis Producer

- Launch the Amazon Linux2 AMI which has the AWS CLI packages preinstalled.
  Medium to Large instance
  Minimum disk capacity of 50 GB
  Internet access (public subnet or through NAT Gateway)

- Attach IAM Role with full access to Kinesis & CloudWatch.
  Download the JSON file from S3 with which you can create a Policy with:
    AmazonKinesisFullAccess
    CloudWatchFullAccess

```
wget https://mapd-veda.s3-us-west-1.amazonaws.com/kinesis_cloudwatch_role.json
```

  On AWS Console -> **Services** -> **IAM** -> **Policies** -> **Create Policy** -> **JSON**
  Cut/paste the json file from above and select **Review Policy**
  Assign **Name** (eg: Kinesis_WS) and **Description**
  Create Policy

  On AWS Console **-> Services -> IAM -> Roles -> Create Role**
  Select **EC2**
  Select **Permissions**
  Specify the policy name from above in **Filter Policies**
  Select **Tags**
  Key=**name** Value=**Kinesis_WS**
  Select **Next**
  Assign Role Name = **Kinesis_WS_Role**

Create Role

Attach the newly created role to the EC2 instance:
Instance -> Instance Settings -> Attach/Replace IAM Role -> Select Role -> Apply

- Setup AWS configuration with access and secret key credentials
```
aws configure
```
Test if the AWS credentials are setup correctly by listing the user's S3 storage
```
aws s3 ls
```

- Create a Kinesis Stream called **bike-status** with a single shard
```
aws kinesis create-stream --stream-name bike-status --shard-count 1
aws kinesis list-streams
aws kinesis describe-stream --stream-name bike-status
```

- Install Kinesis Agent to create a producer
```
sudo yum install -y
https://s3.amazonaws.com/streaming-data-agent/aws-kinesis-agent-latest.amzn1.no
arch.rpm
```

- Setup Kinesis Producer configuration file **/etc/aws-kinesis/agent.json.** Replace the existing file with the contents below.
```
sudo vi /etc/aws-kinesis/agent.json
{
  "cloudwatch.emitMetrics": true,
  "kinesis.endpoint": "",
  "firehose.endpoint": "",
  "flows": [
    {
      "filePattern": "/tmp/kinesis/bike-status*",
      "kinesisStream": "bike-status",
      "partitionKeyOption": "RANDOM"
    }
  ]
}
```

- Start the Kinesis Agent and configure it to start at bootup
```
sudo service aws-kinesis-agent start
sudo service aws-kinesis-agent status
sudo chkconfig aws-kinesis-agent on
```

- Monitor the Kinesis agent at /var/log/aws-kinesis-agent/aws-kinesis-agent.log
```
tail -f /var/log/aws-kinesis-agent/aws-kinesis-agent.log
```

Setup Kinesis Consumer

Setup the Kinesis Consumer on the same server or spin up another Amazon Linux2 instance.
For the workshop we will spin up a single Linux instance for running both the producer and consumer.

- Install Anaconda Python Environment
  ```
  wget https://repo.anaconda.com/archive/Anaconda3-2019.07-Linux-x86_64.sh
  chmod 755 Anaconda3-2019.07-Linux-x86_64.sh
  ./Anaconda3-2019.07-Linux-x86_64.sh
  ```

  The Anaconda environment is initialized by the changes to **.bashrc** which gets executed at the time of login. To activate the Anaconda environment in your current login type:
  ```
  source ~/.bashrc
  ```

- Install boto3 library for kinesis consumer support and pymapd library for OmniSci API access.
  ```
  conda install boto3
  conda install -c conda-forge pymapd
  ```

OmniSci Cloud Instance

Get your own instance with a GPU in OmniSci Cloud by signing up at:
https://omnisci.cloud/accounts/register_from_event/kinesis2019

- Generating API Keys for Cloud Access

Select **DEVELOPER** menu from **SETTINGS** which is at the upper right-hand corner of the OmniSci Immerse user interface, and **Create Write keys**. Save the API Key Name and API Key Secret securely. You will need these keys when using the python API to access your OmniSci cloud instance. Refer to the appendix in this document for a brief overview of the available pymapd APIs.

Streaming data to OmniSci using Kinesis

- Copy the list of bike status endpoints from S3.
  ```
  wget https://mapd-veda.s3-us-west-1.amazonaws.com/gbfs_endpoints.csv
  ```

  Also, download the producer and consumer python programs from S3.
  ```
  wget https://mapd-veda.s3-us-west-1.amazonaws.com/bikestream.py
  wget https://mapd-veda.s3-us-west-1.amazonaws.com/streamconsumer.py
  ```

- Start the python program to feed the Kinesis Producer with bike status data stream. You may need to call the program with `python3` if you have different environment.
  ```
  python bikestream.py
  ```

  The bikestream program will get the bike status from the various endpoints and write it to a CSV file **/tmp/kinesis/bike-status.csv**. Note that the producer configuration (json) is looking for data input to files that match the /tmp/kinesis/bike-status* pattern.
  ```
  head -10 /tmp/kinesis/bike-status.csv
  ```

- Start the python Kinesis consumer program that will read the data from the producer and create a Pandas dataframe for 1000 records at a time and write to the table **bike-status** on OmniSci.
  NOTE: The code is using **us-east region** by default**.**

In the streamconsumer.py file, replace your cloud instance’s access and secret key in the function connect_to_omnisci().

```
connection = connect_to_omnisci(“Access Key”, “Secret Key”,
“use2-api.mapd.cloud”, “mapd”, True)
```

Run the consumer code:

```
python streamconsumer.py
```

- Confirm that the Kinesis workflow is working using OmniSci Immerse user interface. Go to the DATA MANAGER and look at the bike_status table to make sure that the number of rows is increasing every time you run the producer code.

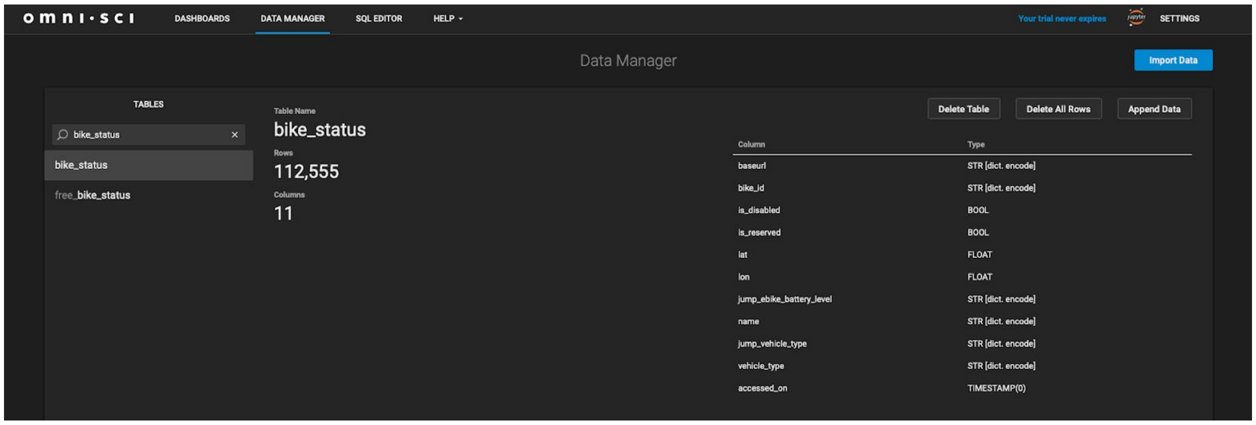


Table Name
bike_status
Rows
112,555
Columns
11

Column	Type
baseurl	STR [dict. encode]
bike_id	STR [dict. encode]
is_disabled	BOOL
is_reserved	BOOL
lat	FLOAT
lon	FLOAT
jump_ebike_battery_level	STR [dict. encode]
name	STR [dict. encode]
jump_vehicle_type	STR [dict. encode]
vehicle_type	STR [dict. encode]
accessed_on	TIMESTAMP(0)

References

Blog explaining simple end-to-end example with source code for using pymapd with OmniSci.
https://www.omnisci.com/blog/using-pymapd-to-load-data-to-omnisci-cloud/

GitHub - Pymapd Workshop with material for this workshop and useful Jupyter Notebooks
https://github.com/omnisci/pymapd-workshop

GitHub - General Bikeshare Feed Information
https://github.com/NABSA/gbfs

Blog on using OmniSci with Ford GoBike data:
https://www.omnisci.com/blog/taking-a-spin-with-ford-gobike

Blog on using OmniSci with Divvy Bike data:
https://www.omnisci.com/blog/oh-the-places-youll-go-analyzing-chicago-divvy-bike-share-data

Appendix: Using Python API to access OmniSci Cloud Instance

Install Pymapd

Pymapd is the python DB API compliant interface for OmniSci.
You can install pymapd from conda-forge

```
conda install -c conda-forge pymapd
```

Or with pip

```
pip install pymapd
```

Generating API Keys for OmniSci Cloud

Select DEVELOPER menu from SETTINGS which is at the upper right-hand corner of the Immerse user interface, and Create Write keys. Save the API Key Name and API Key Secret.

Connect to OmniSci

We'll use the OmniSci API key name for write access and the corresponding API key secret and initialize the connection variable. All API access goes through the same host, use2-api.mapd.cloud, which then automatically forwards the request to the correct cloud instance based on the key name. The default database name is mapd and you will connect using HTTPS port 443.

```
user_str = 'API Key Name'
password_str = 'API Key Secret'
host_str = 'use2-api.mapd.cloud'
dbname_str = 'mapd'
connection = connect(user=user_str, password=password_str, host=host_str,
dbname=dbname_str, port=443, protocol='https')
```

Load Data Into OmniSci Cloud

Read the data from a local CSV file into a pandas dataframe:

```
df = pd.read_csv("replace with your filename")
```

Load the dataframe into the OmniSci table using load_table API, the table is created if it does not exist:

```
table_name = 'toSF'
connection.load_table(table_name, df)
```

You can also load data into OmniSci from data files stored on AWS S3:

```
connection.execute("COPY toSF FROM 's3://mybucketname/myfilenameCSV'")
```

This method is particularly useful for loading Geospatial data files (ESRI Shape file & GeoJSON) into OmniSci Cloud:

```
connection.execute("COPY toSF FROM 's3://mybucketname/myshapefile.zip' WITH
(geo='true')")
```

Other Useful pymapd Commands

List tables in the database:

```
list_of_tables = connection.get_tables()
print('\n'.join(list_of_tables))
```

Get details of a preloaded table:

```
table_details = connection.get_table_details('toSF’)
print(table_details)
```

Delete table:

```
table_name = 'toSF'
command = 'drop table if exists %s' % (table_name)
connection.execute(command)
```