Cloud Tools Guidance

Author: Michael Kohs George Vetticaden Timothy Spann
Date: 04/19/2023
Last Updated: 5/01/2023

Notice

This document assumes that you have registered for an account, activated it and logged into the CDP Sandbox. This is for authorized users only who have attended the webinar and have read the training materials.

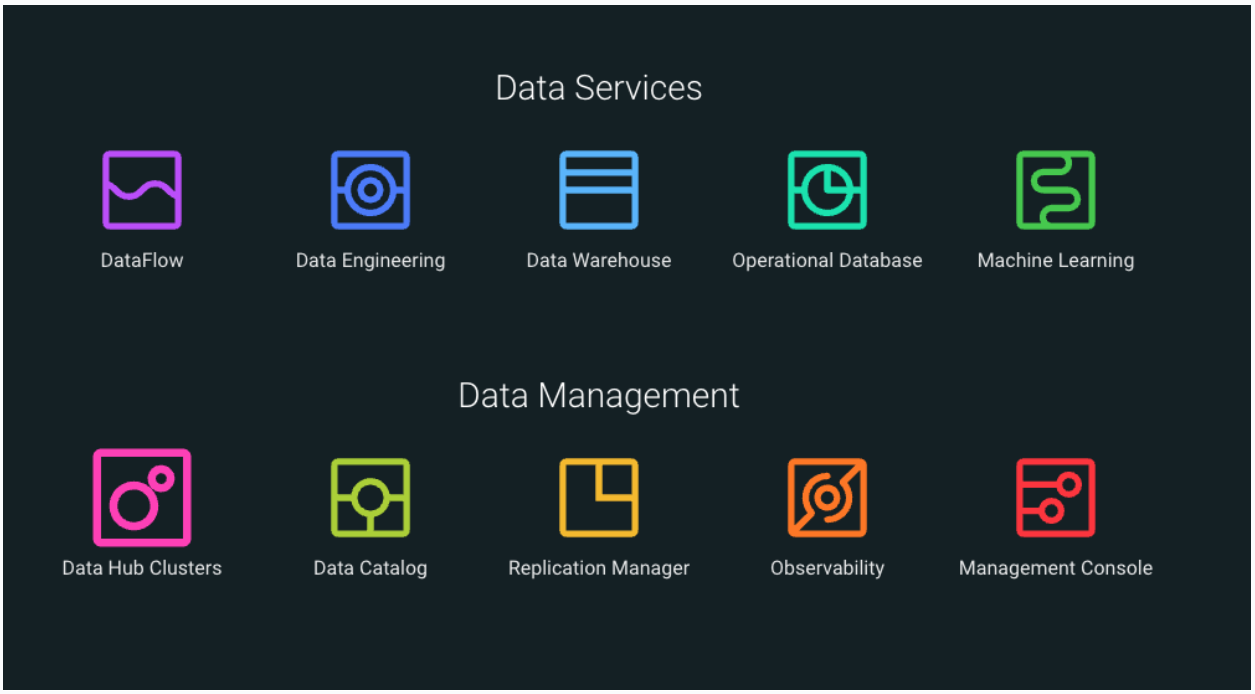
A short guide and references are listed here.

THIS IS NOT FOR USE WITH THE FIRST TWO TUTORIALS. THIS IS FOR BUILDING ASSETS FOR YOUR OWN NEW FLOWS.

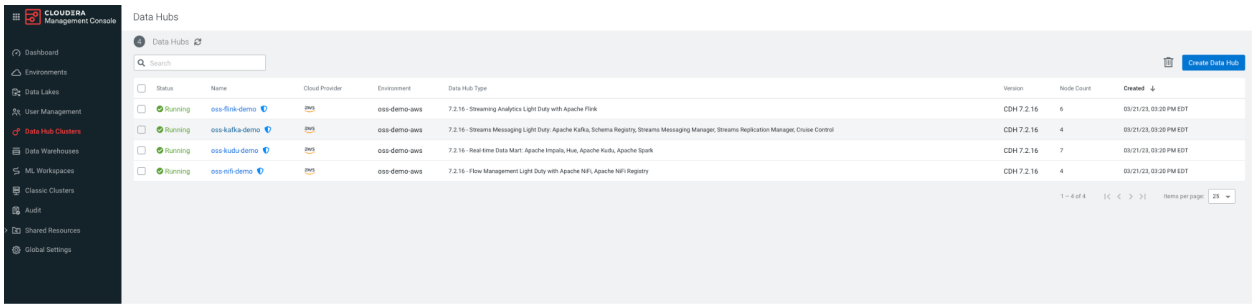
1. How To Build Data Assets

1.1 Create a Kafka Topic

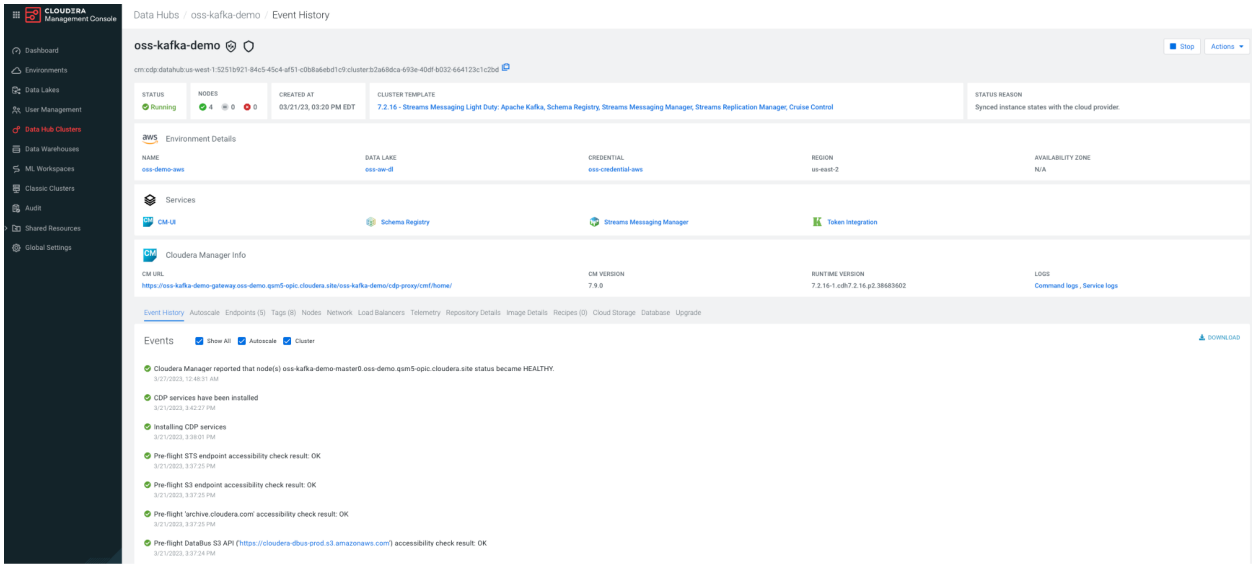
1. Navigate to **Data Hub Clusters**



2. Navigate to the **oss-kafka-demo** cluster

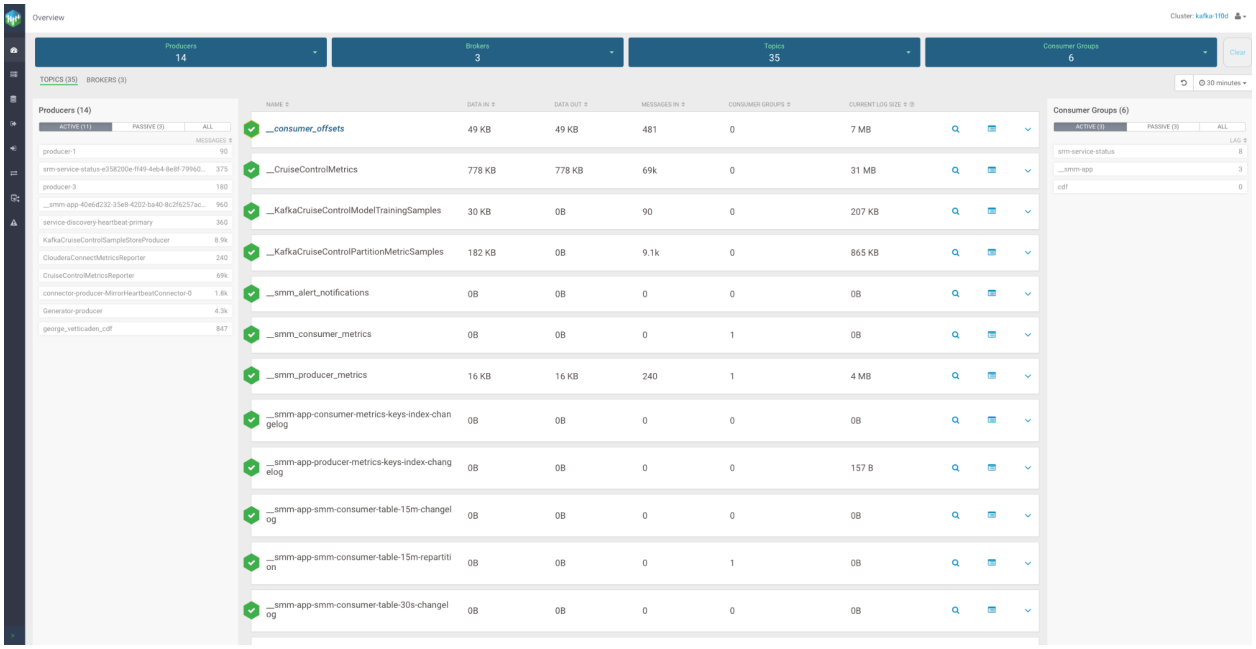


3. Navigate to Streams Messaging Manager

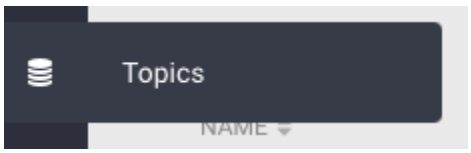


Info: Streams Messaging Manager (SMM) is a tool for working with Apache Kafka.

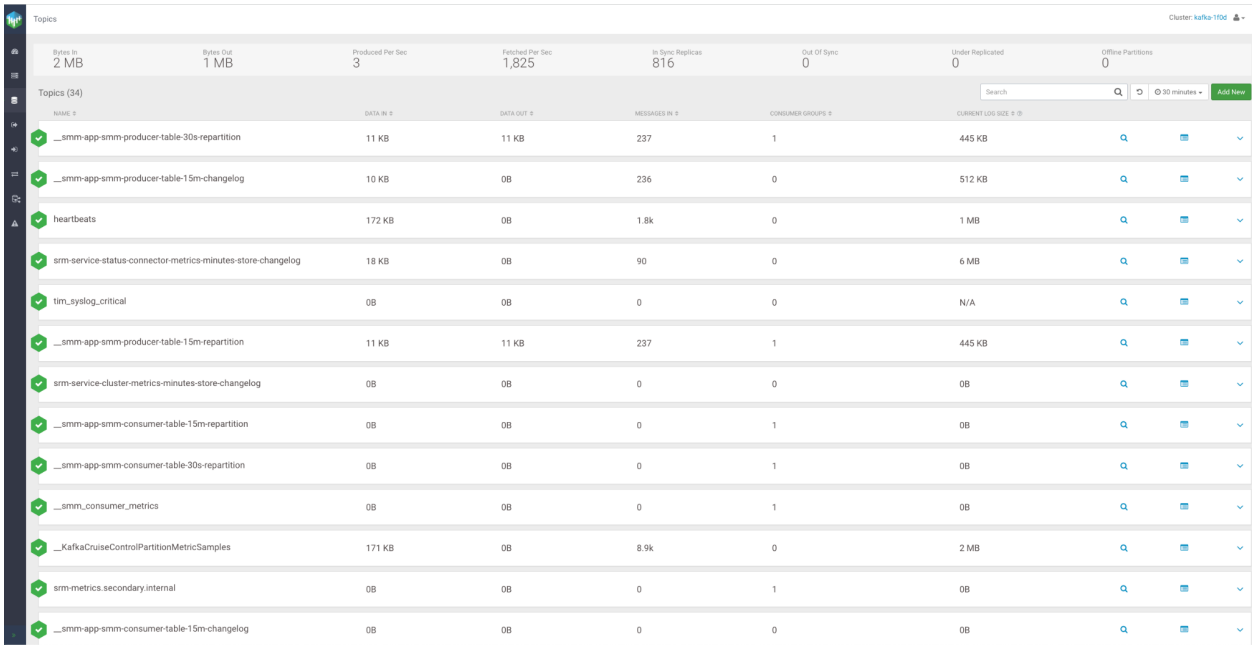
4. Now that you are in SMM.



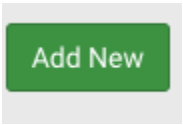
5. Navigate to the round icon third from the top, click this **Topic** button.



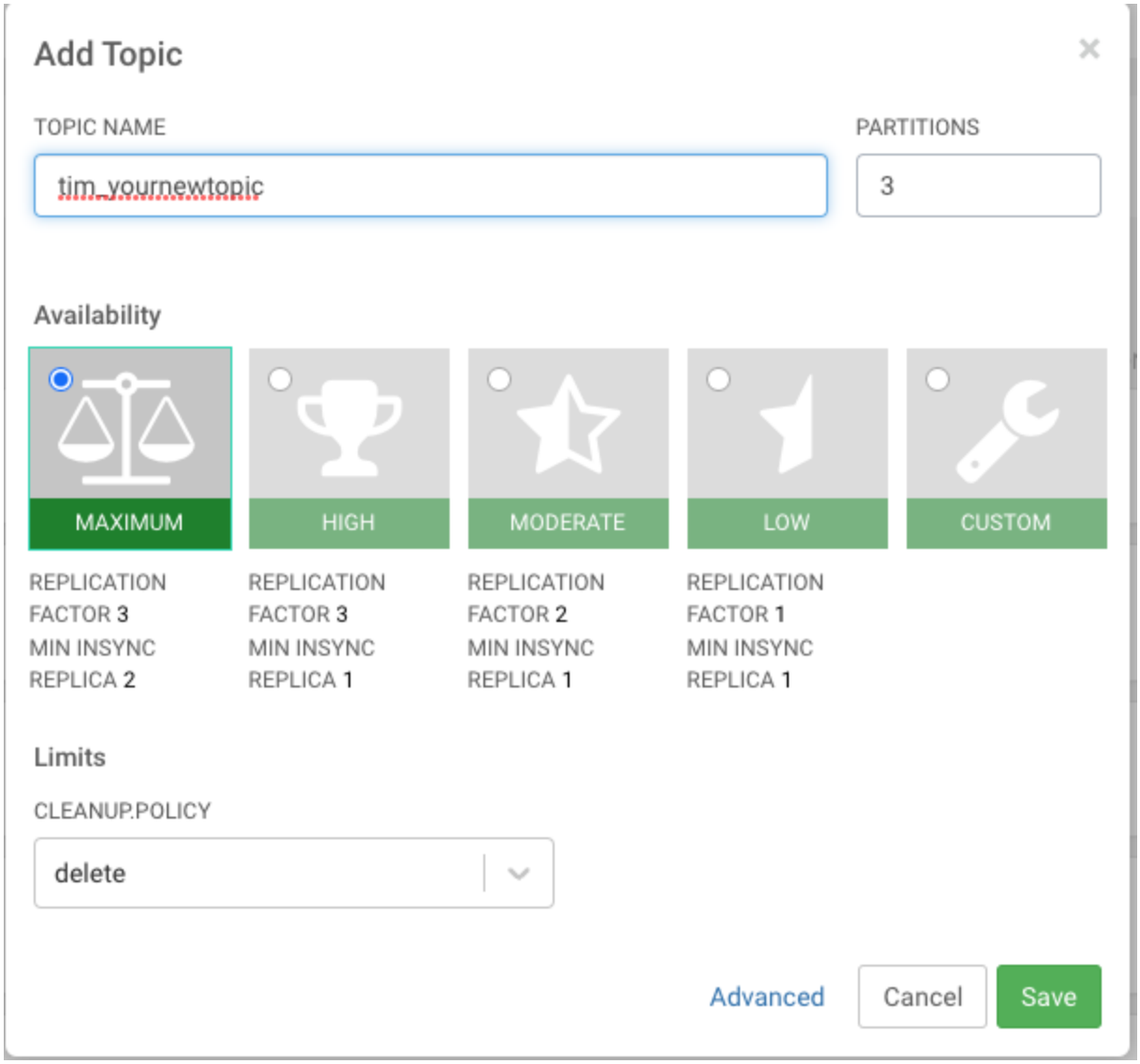
6. You are now in the Topic browser.



7. Click **Add New** to build a new topic.

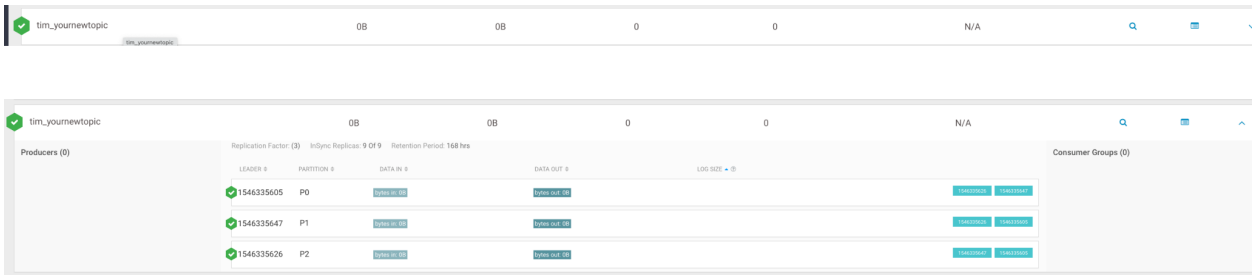


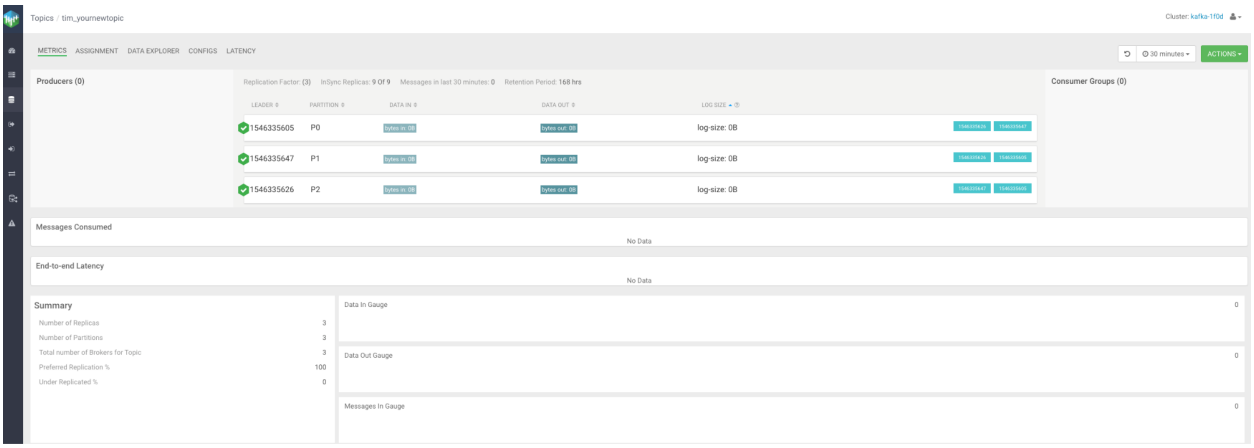
8. Enter the name of your topic prefixed with your **“Workload User Name“** <yourusername>_yournewtopic, ex: tim_younewtopic.



9. Enter the name of your topic prefixed with your Workload User Name, ex: **tim_younewtopic**. For settings you should create it with **(3 partitions, cleanup.policy: delete, availability maximum)** as shown above.

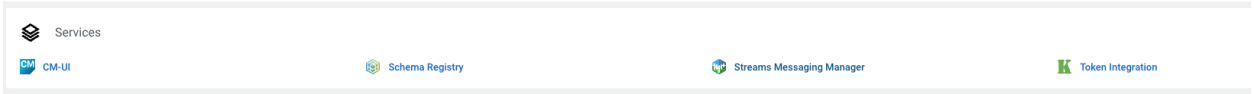
Congratulations! You have built a new topic.

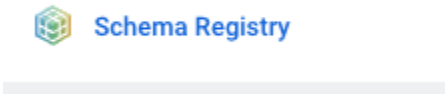




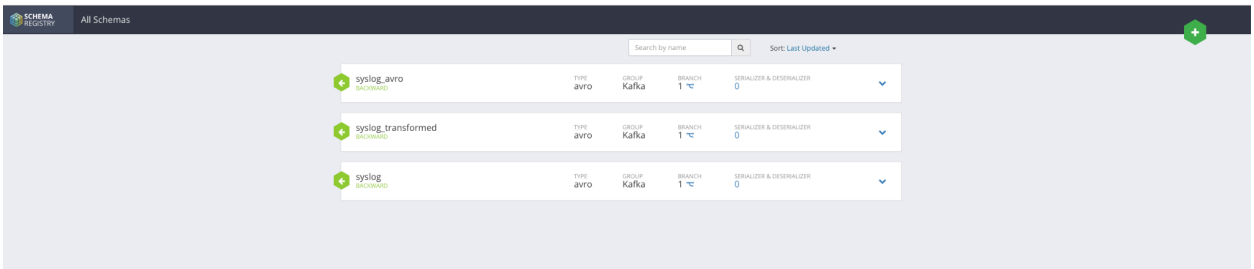
1.2 Create a Schema If You Need One. Not Required For Using Kafka Topics or Tutorials.

1. Navigate to **Schema Registry** from the Kafka Data Hub.



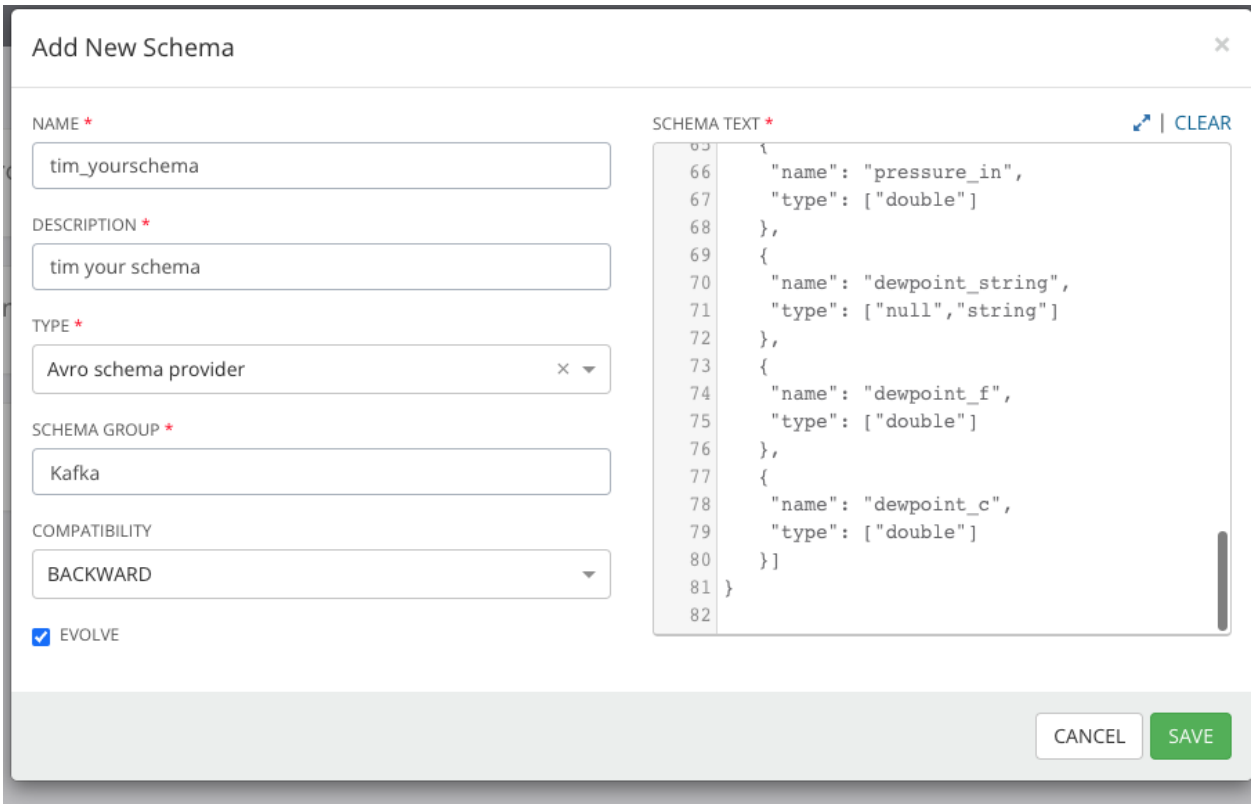


2. You will see existing schemas.



3. Click the white plus sign in the gray hexagon to create a new schema.

4. You can now add a new schema by entering a unique name starting with your **Workload User Name (ex: tim),** followed by a short description and then the schema text as shown. If you need examples, see the github list at the end of this guide.

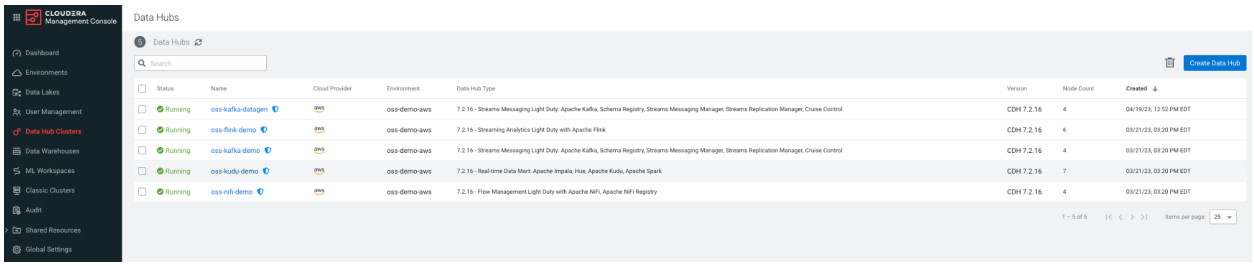


5. Click Save and you have a new schema. If there were errors they will be shown and you can fix them. For more help see, Schema Registry Documentation and Schema Registry Public Cloud.

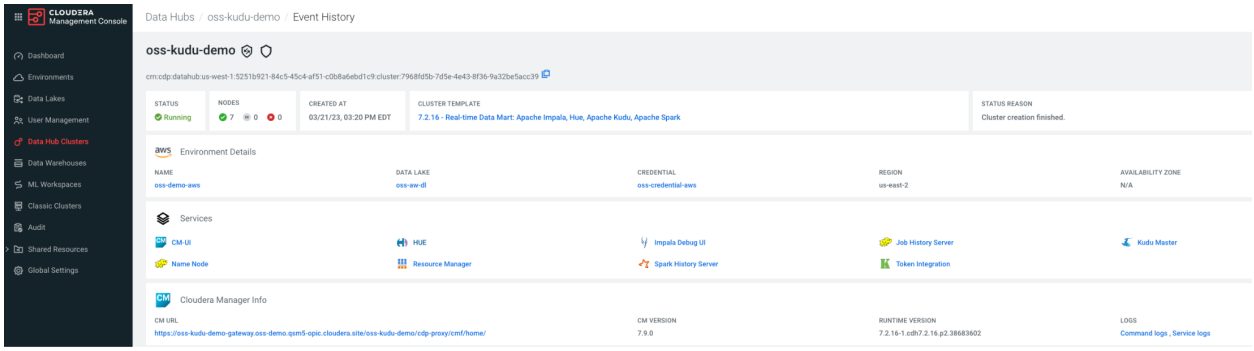
Congratulations! You have built a new schema. Start using it in your DataFlow application.

1.3 Create an Apache Iceberg Table

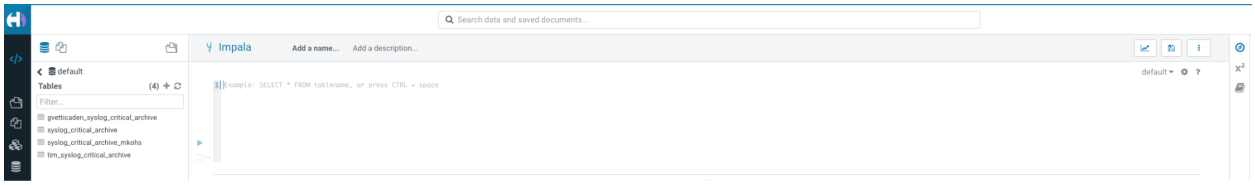
1. Navigate to **oss-kudu-demo** from the Data Hubs list



2. Navigate to **Hue** from the Kudu Data Hub.



3. Inside of **Hue** you can now create your table.



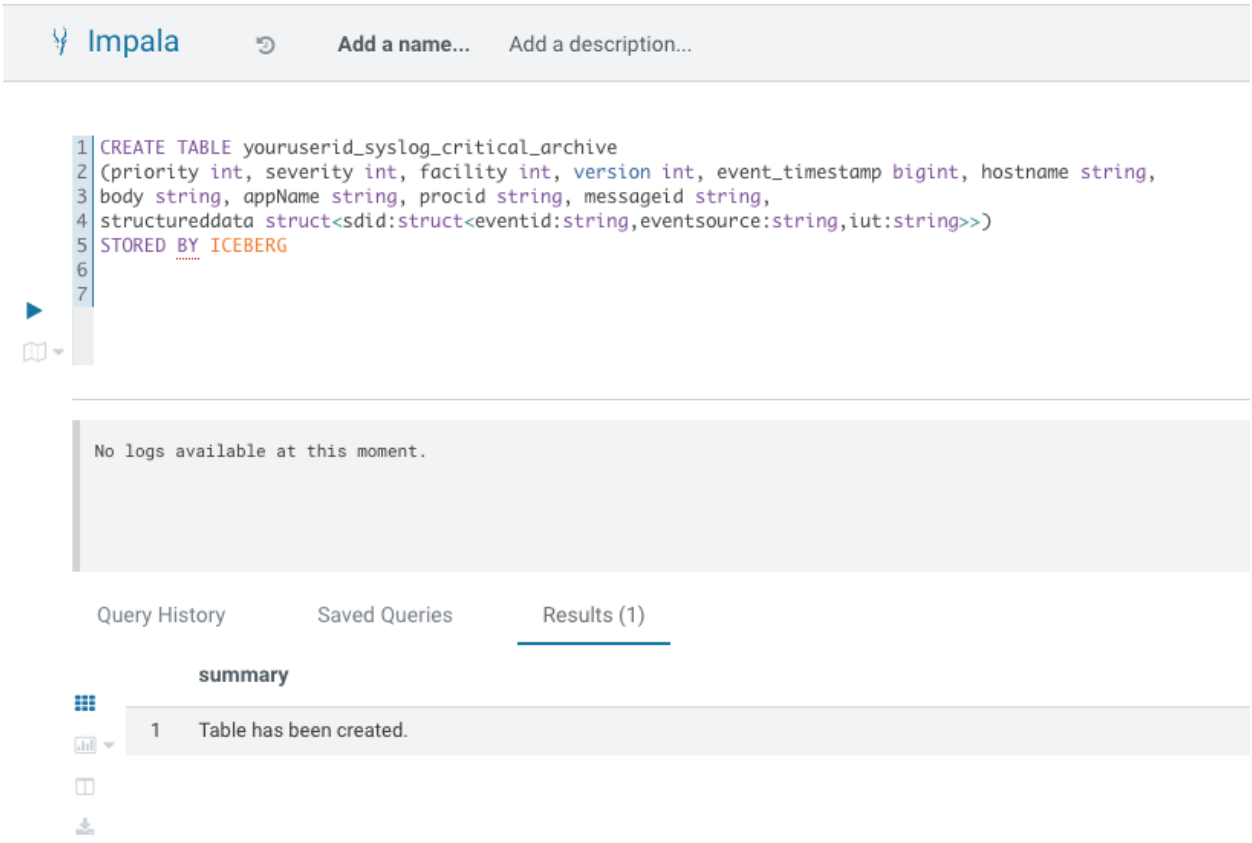
4. Navigate to your database, this was created for you.

Info: The database name pattern is your email address and then all special characters are replaced with **underscore** and then **_db** is appended to that to make the db name and the ranger policy is created to limit access to just the user and those that are in the admin group. For example:

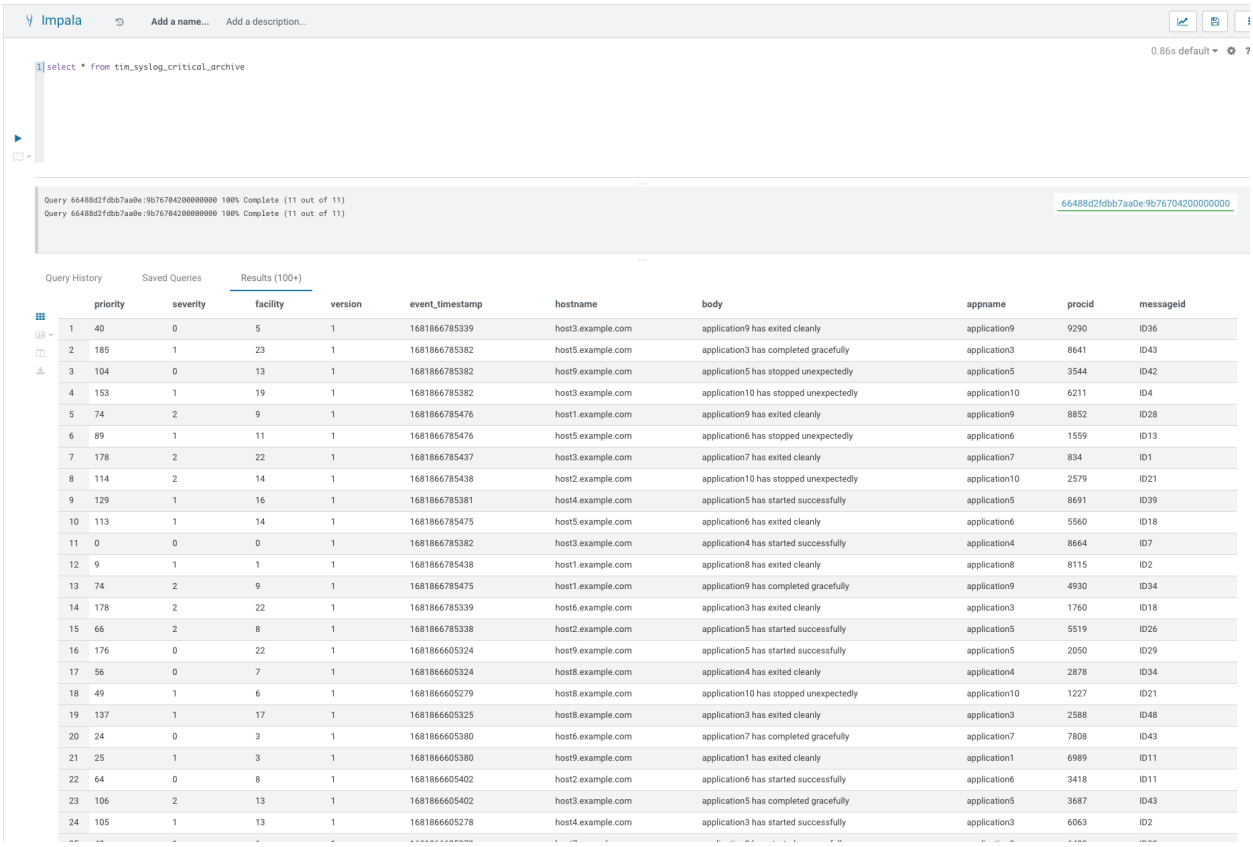
5. Create your Apache Iceberg table, it must be prefixed with your **Work Load User Name (userid)**.

```
CREATE TABLE <<userid>>_syslog_critical_archive
(priority int, severity int, facility int, version int, event_timestamp bigint, hostname string,
body string, appName string, procid string, messageid string,
structureddata struct<sdid:struct<eventid:string,eventsource:string,iut:string>>)
STORED BY ICEBERG
```

6. Your table is created in **s3a://oss-uat2/iceberg/**



7. Once you have sent data to your table, you can query it.



	priority	severity	facility	version	event_timestamp	hostname	body	appname	procid	messageid
1	40	0	5	1	1681866785339	host3.example.com	application9 has exited cleanly	application9	9290	ID36
2	185	1	23	1	1681866785382	host5.example.com	application3 has completed gracefully	application3	8641	ID43
3	104	0	13	1	1681866785382	host9.example.com	application5 has stopped unexpectedly	application5	3544	ID42
4	153	1	19	1	1681866785382	host3.example.com	application10 has stopped unexpectedly	application10	6211	ID4
5	74	2	9	1	1681866785476	host1.example.com	application9 has exited cleanly	application9	8852	ID28
6	89	1	11	1	1681866785476	host5.example.com	application6 has stopped unexpectedly	application6	1559	ID13
7	178	2	22	1	1681866785437	host3.example.com	application7 has exited cleanly	application7	834	ID1
8	114	2	14	1	1681866785438	host2.example.com	application10 has stopped unexpectedly	application10	2579	ID21
9	129	1	16	1	1681866785381	host4.example.com	application5 has started successfully	application5	8691	ID39
10	113	1	14	1	1681866785475	host5.example.com	application6 has exited cleanly	application6	5560	ID18
11	0	0	0	1	1681866785382	host3.example.com	application4 has started successfully	application4	8664	ID7
12	9	1	1	1	1681866785438	host1.example.com	application8 has exited cleanly	application8	8115	ID2
13	74	2	9	1	1681866785475	host1.example.com	application9 has completed gracefully	application9	4930	ID34
14	178	2	22	1	1681866785339	host6.example.com	application3 has exited cleanly	application3	1760	ID18
15	66	2	8	1	1681866785338	host2.example.com	application5 has started successfully	application5	5519	ID26
16	176	0	22	1	1681866605324	host9.example.com	application5 has started successfully	application5	2050	ID29
17	56	0	7	1	1681866605324	host8.example.com	application4 has exited cleanly	application4	2878	ID34
18	49	1	6	1	1681866605279	host8.example.com	application10 has stopped unexpectedly	application10	1227	ID21
19	137	1	17	1	1681866605325	host8.example.com	application3 has exited cleanly	application3	2588	ID48
20	24	0	3	1	1681866605380	host6.example.com	application7 has completed gracefully	application7	7808	ID43
21	25	1	3	1	1681866605380	host9.example.com	application1 has exited cleanly	application1	6989	ID11
22	64	0	8	1	1681866605402	host2.example.com	application6 has started successfully	application6	3418	ID11
23	106	2	13	1	1681866605402	host3.example.com	application5 has completed gracefully	application5	3687	ID43
24	105	1	13	1	1681866605278	host4.example.com	application3 has started successfully	application3	6063	ID2

Additional Documentation

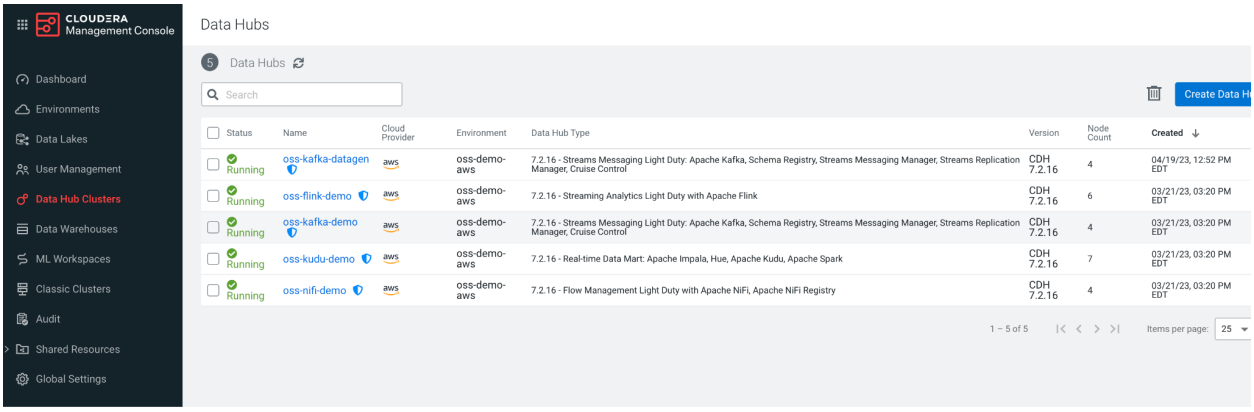
- Create a Table
- Query a Table
- Apache Iceberg Table Properties

2. Streaming Data Sets Available for Apps

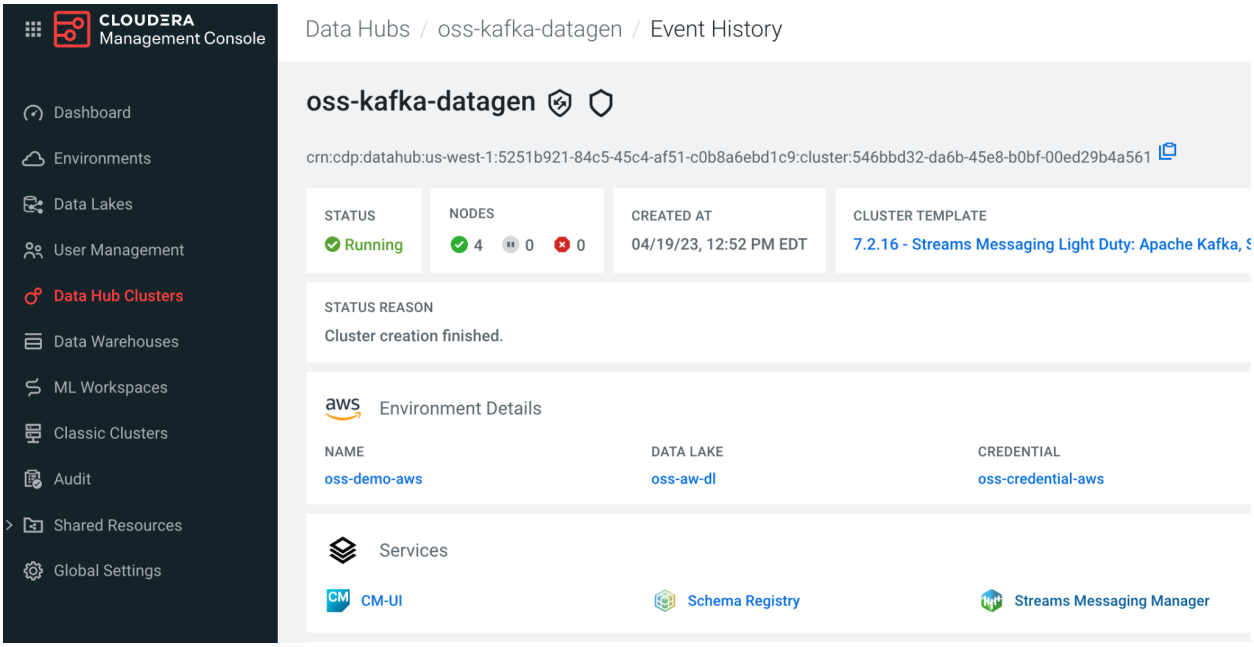
The following Kafka topics are being populated with streaming data for you.

These come from the read-only Kafka cluster.

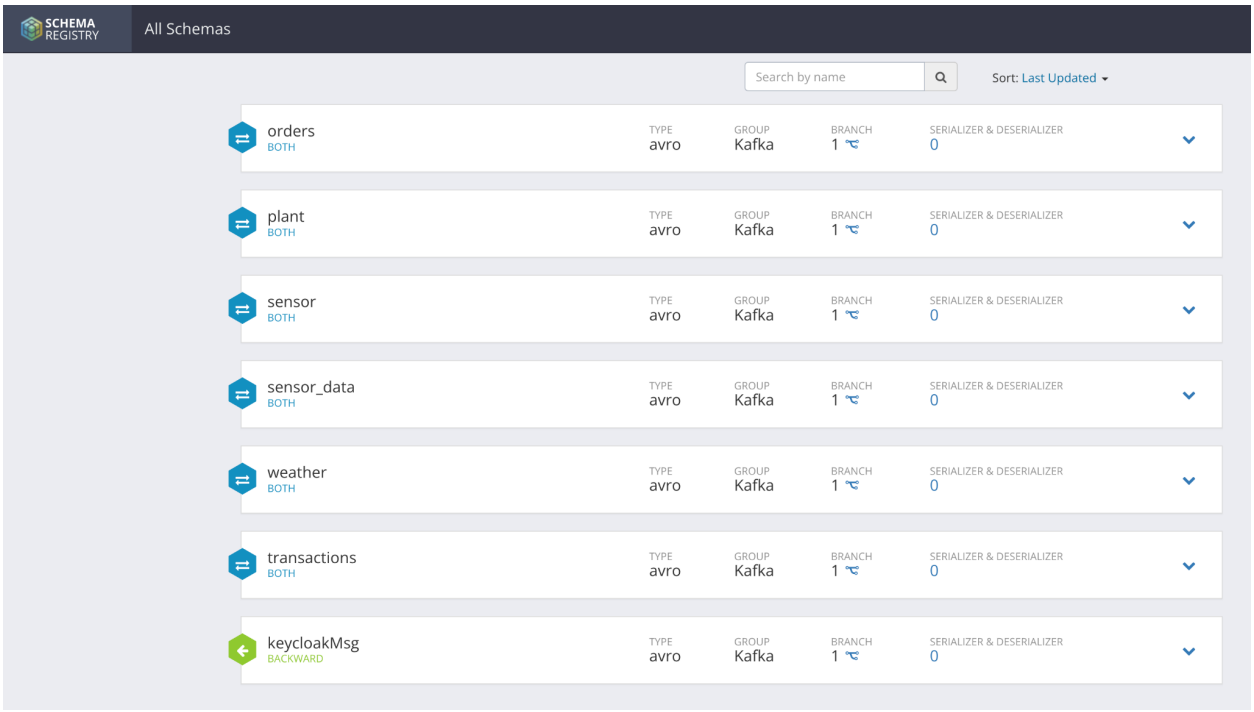
Navigate to the **Data Hub Clusters.**



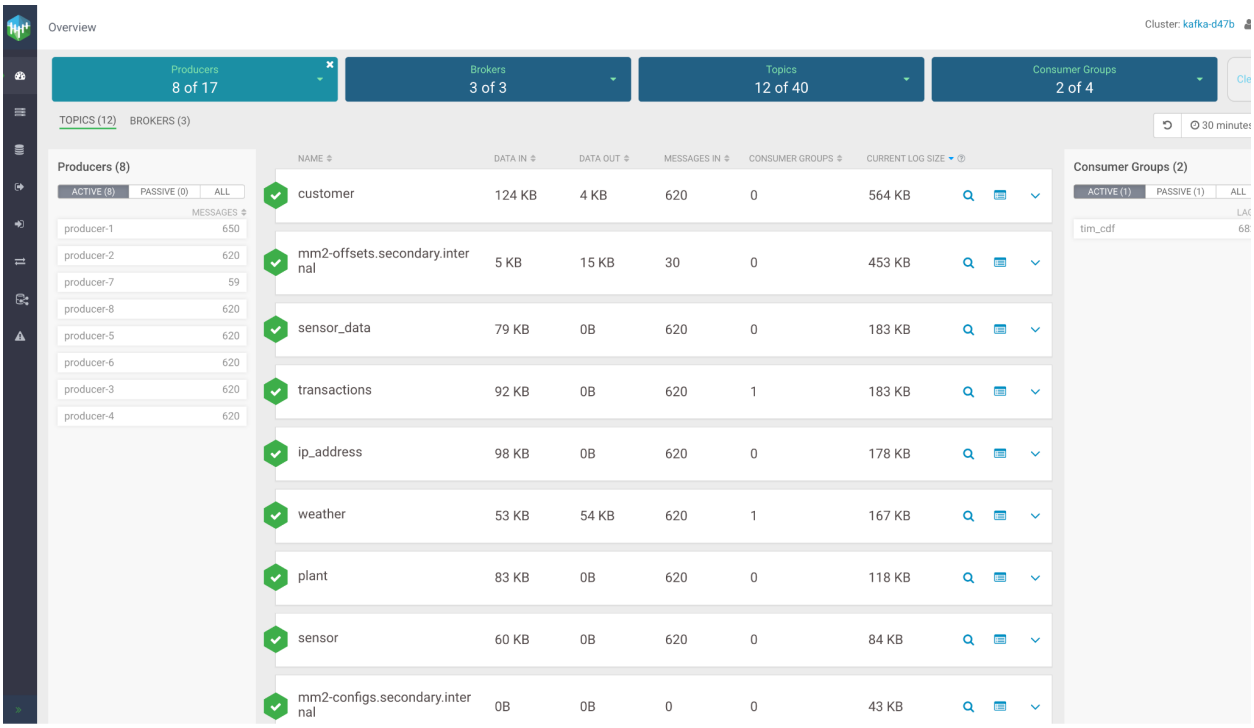
Click on **oss-kafka-datagen**.



Click **Schema Registry.**



Click **Streams Messaging Manager.**



Use these brokers to connect to them:

Brokers

oss-kafka-datagen-corebroker1.oss-demo.qsm5-opic.cloudera.site:9093,oss-kafka-datagen-corebroker0.oss-demo.qsm5-opic.cloudera.site:9093,oss-kafka-datagen-corebroker2.oss-demo.qsm5-opic.cloudera.site:9093

Use this link for Schema Registry

https://#{Schema2}:7790/api/v1

Schema Registry Parameter Hostname: Schema2

oss-kafka-datagen-master0.oss-demo.qsm5-opic.cloudera.site

To View Schemas in the Schema Registry click the icon from the datahub

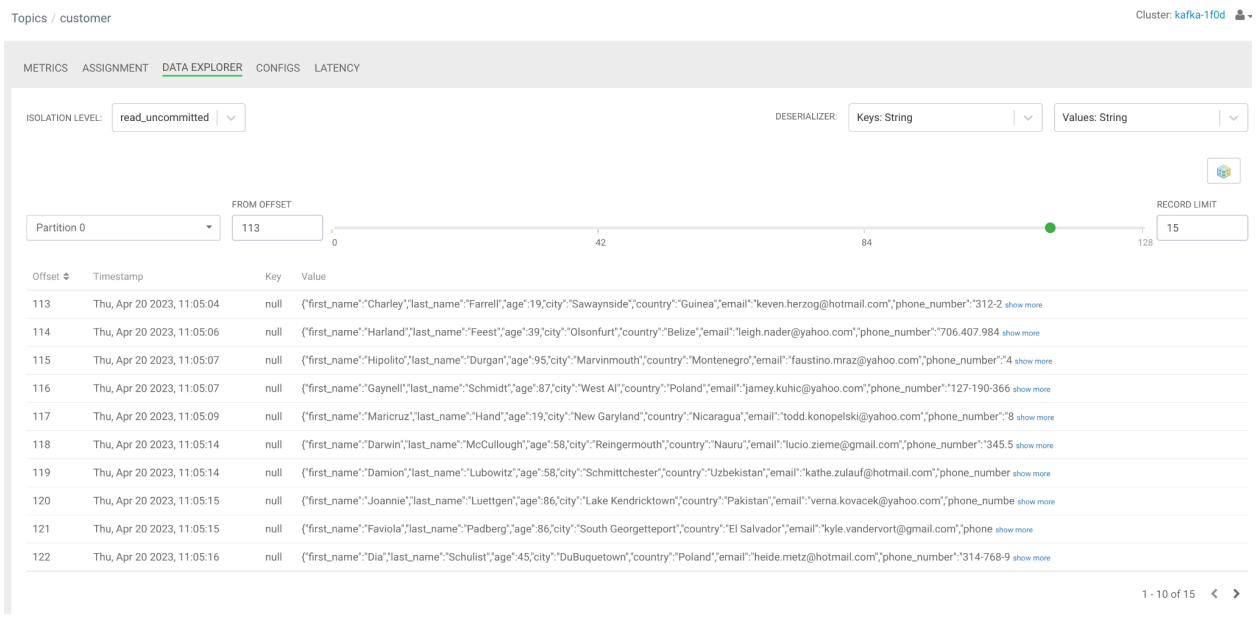
https://oss-kafka-datagen-gateway.oss-demo.qsm5-opic.cloudera.site/oss-kafka-datagen/cdp-proxy/schema-registry/ui/#/

Schemas
https://github.com/tspannhw/FLaNK-DataFlows/tree/main/schemas

Group ID: yourid_cdf

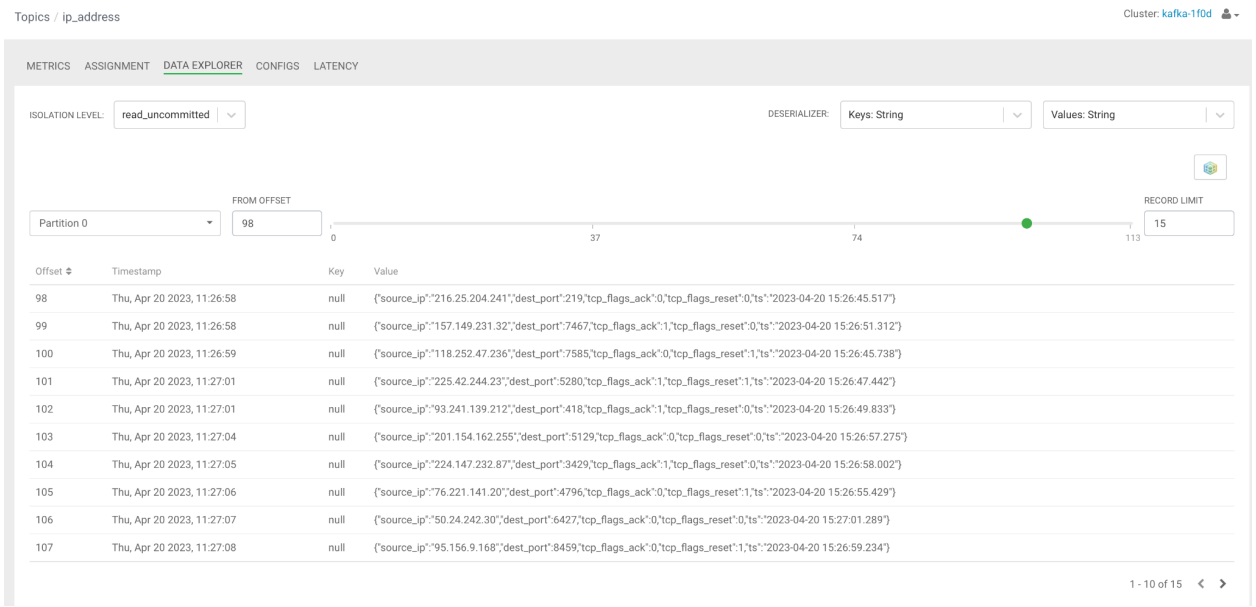
Customers (**customer**)

Example Row
{"first_name":"Charley","last_name":"Farrell","age":19,"city":"Sawaynside","country":"Guinea","email":"kveen.herzog@hotmail.com","phone_number":"312-269-6619"}



IP Tables (ip_address)

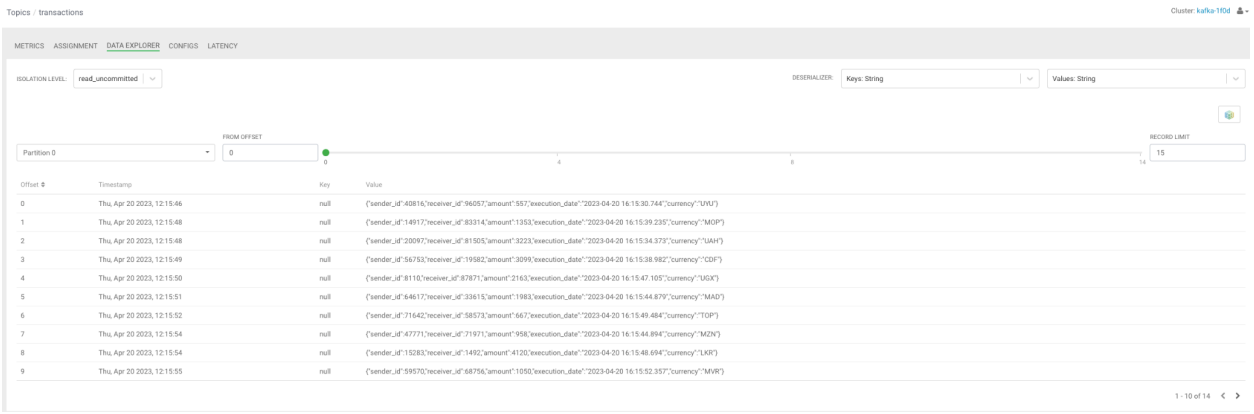
Example Row
{"source_ip":"216.25.204.241","dest_port":219,"tcp_flags_ack":0,"tcp_flags_reset":0,"ts":"2023-04-20 15:26:45.517"}



Orders (orders)

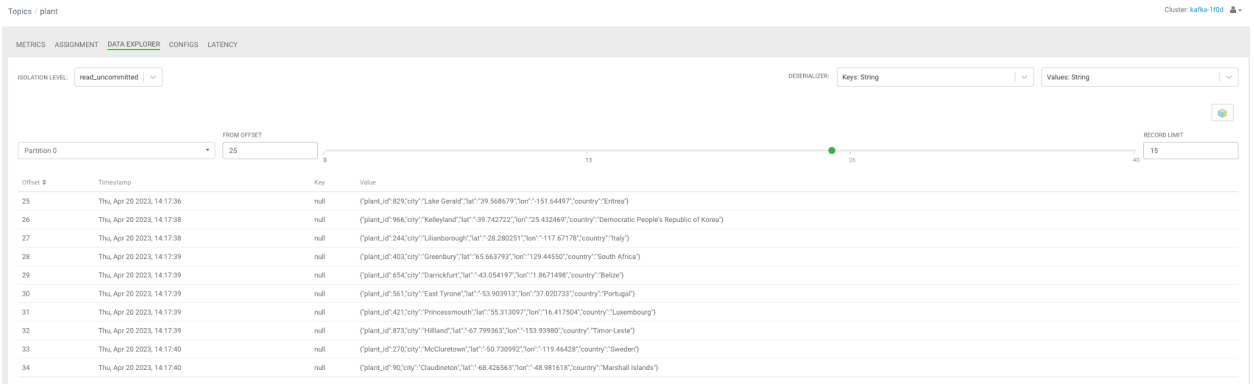
Example Row

{"order_id":84170282,"city":"Wintheiserton","street_address":"80206 Caroyln Lakes","amount":29,"order_time":"2023-04-20 13:25:06.097","order_status":"DELIVERED"}



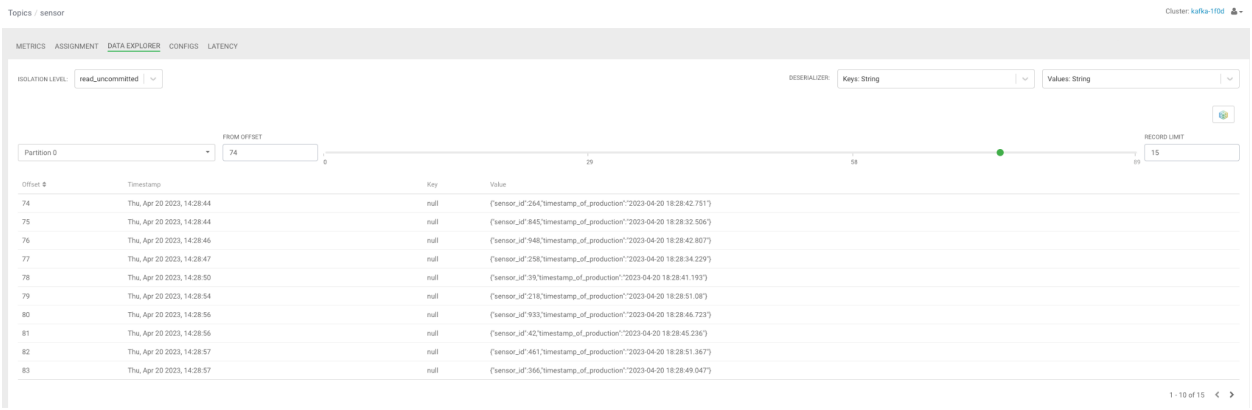
Plants (plant)

Example Row
{"plant_id":829,"city":"Lake Gerald","lat":"39.568679","lon":"-151.64497","country":"Eritrea"}



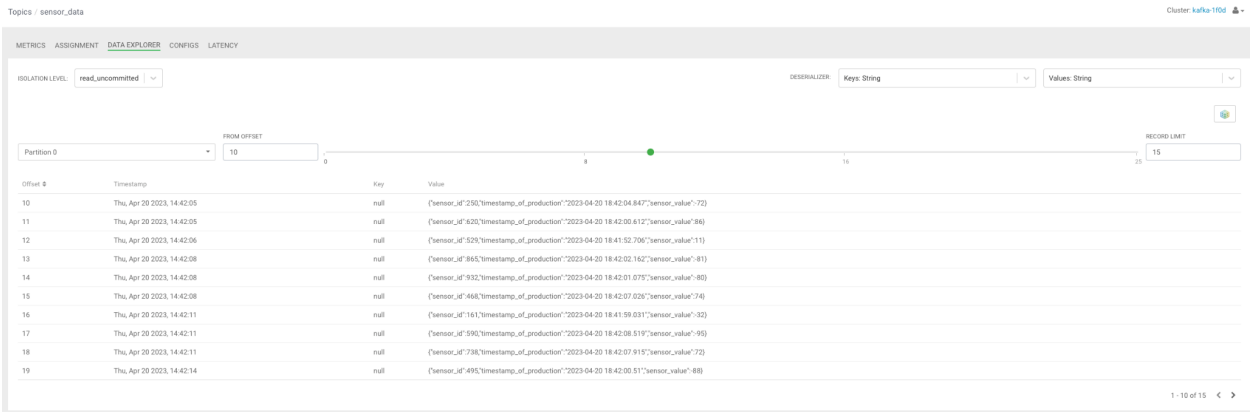
Sensors (sensor)

Example Row
{"sensor_id":264,"timestamp_of_production":"2023-04-20 18:28:42.751"}



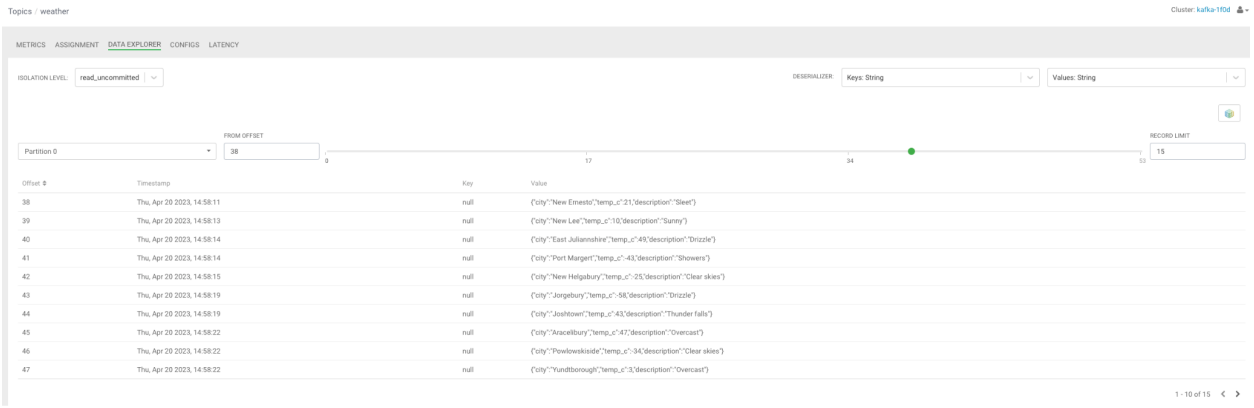
Sensor Data (sensor_data)

Example Row
{"sensor_id":250,"timestamp_of_production":"2023-04-20 18:42:04.847","sensor_value":-72}



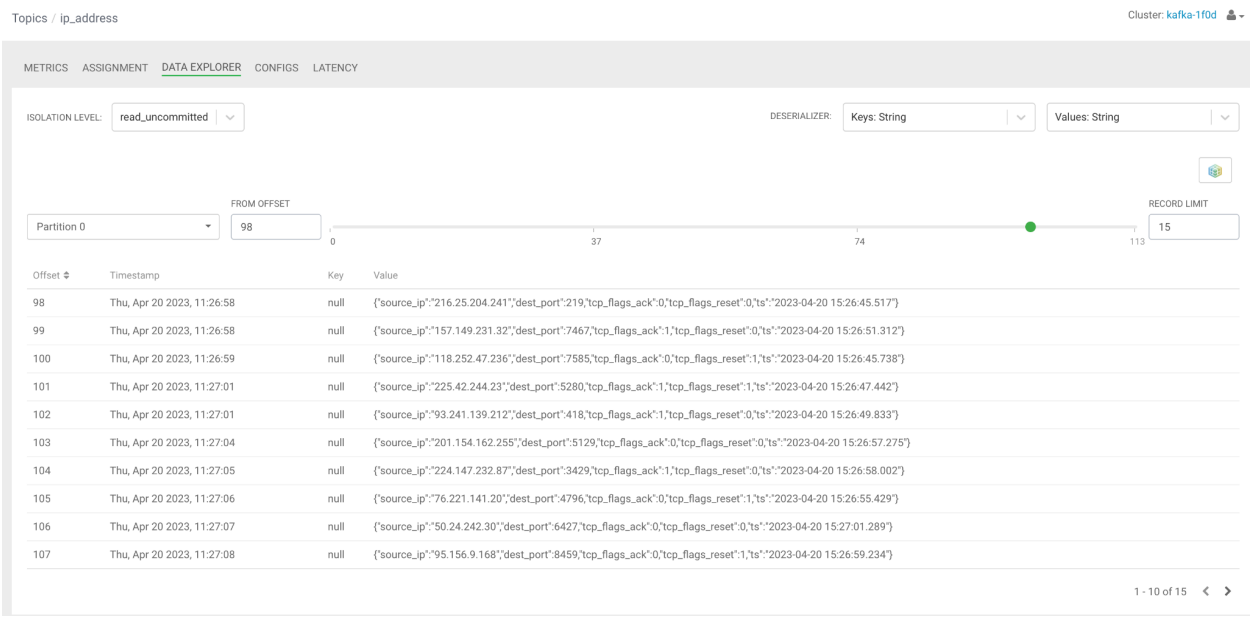
Weather (weather)

Example Row
{"city":"New Ernesto","temp_c":21,"description":"Sleet"}



Transactions (**transactions**)

Example Row
{"sender_id":40816,"receiver_id":96057,"amount":557,"execution_date":"2023-04-20 16:15:30.744","currency":"UYU"}



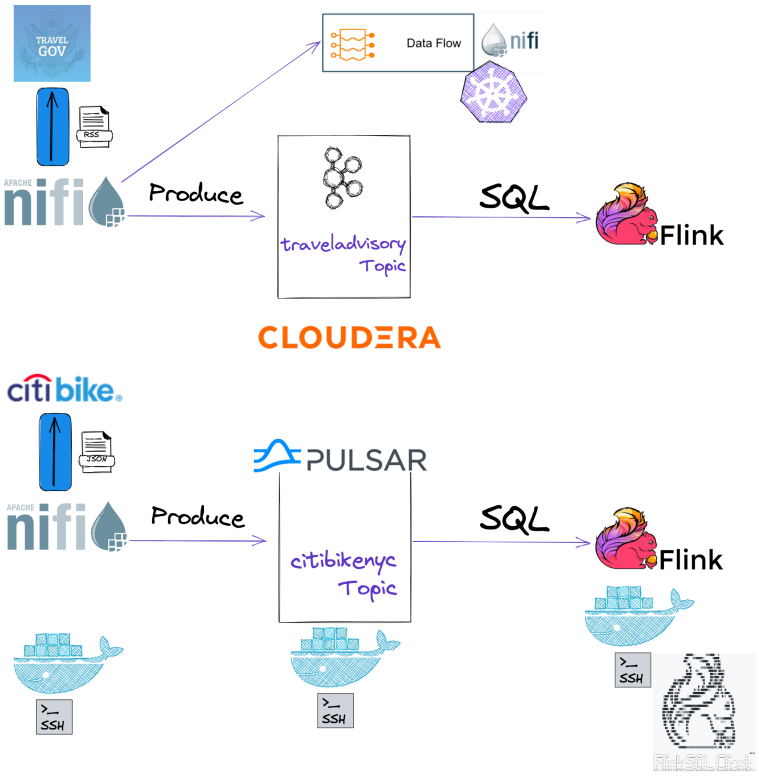
These are realistic generated data sources that you can use, they are available from read-only Kafka topics. These can be consumed by any developers in the sandbox.

Make sure you name your Kafka Consumer your Workload Username _ Some Name.

Ex: tim_customerdata_reader

3. Bring Your Own Data (Public Only)

- Data is **visible** and **downloadable** to all, make sure it is safe, free, **open**, public data.
- Public REST Feeds are good
    - Wikipedia https://docs.cloudera.com/dataflow/cloud/flow-designer-beginners-guide-readyflow/topics/cdf-flow-designer-getting-started-readyflow.html
    - https://gbfs.citibikenyc.com/gbfs/en/station_status.json
    - https://travel.state.gov/_res/rss/TAsTWs.xml
    - https://www.njtransit.com/rss/BusAdvisories_feed.xml
    - https://www.njtransit.com/rss/RailAdvisories_feed.xml
    - https://www.njtransit.com/rss/LightRailAdvisories_feed.xml
    - https://www.njtransit.com/rss/CustomerNotices_feed.xml
    - https://w1.weather.gov/xml/current_obs/all_xml.zip
    - https://dailymed.nlm.nih.gov/dailymed/services/v2/spls.json?page=1&pagesize=100
    - https://dailymed.nlm.nih.gov/dailymed/services/v2/drugnames.json?pagesize=100
    - https://dailymed.nlm.nih.gov/dailymed/rss.cfm
- Generic data files
    - https://aws.amazon.com/data-exchange
- Simulators
    - Use external data simulators via REST
    - Use GeneralFlowFile see: https://www.datainmotion.dev/2019/04/integration-testing-for-apache-nifi.html
- Schemas, Data Sources and Examples
    - https://github.com/tspannhw/FLaNK-AllTheStreams/
    - https://github.com/tspannhw/FLaNK-DataFlows
    - https://github.com/tspannhw/FLaNK-TravelAdvisory/
    - https://github.com/tspannhw/FLiP-Current22-LetsMonitorAllTheThings
    - https://github.com/tspannhw/create-nifi-kafka-flink-apps
    - https://www.datainmotion.dev/2021/01/flank-real-time-transit-information-for.html


TRAVEL GOV
RSS
Data Flow
nifi
APACHE nifi
Produce
traveladvisory Topic
SQL
Flink
CLOUDERA
citibike
JSON
PULSAR
APACHE nifi
Produce
citibikenyc Topic
SQL
Flink
>_ SSH
>_ SSH
>_ SSH