



Paris Public Cloud Hands on Lab Workshop Guide

18 Oct 2023

Jacques Marchand

Charles Aad

Cristina Sanchez

TABLE OF CONTENTS

Foreword	2
1. Introduction	3
2. Data Flow Lab:	4
2.1. Goals	4
2.2. Lab 1 - Ingest Kafka streams to Iceberg table	4
2.3. Lab 2 - Stream Messaging Manager - Optional	19
3. Data Engineering	22
3.1. Goals	22
3.2. Lab 1 - Enrich the Ingested Iceberg table	22
4. Data Warehouse	32
4.1. Goals	32
4.2. Dashboard Development	32
5. Machine Learning	42
5.1. Goals	42
5.2. Create a Machine Learning Model for Churn Prediction	42
6. Optional Labs	60
6.1. Goals	60
6.2. ML -Deploy Applied Machine Learning Model - Optional	60
6.3. Add a third visual element - Optional	63
6.4. One more visual element - Optional	63
6.5. Data Discovery and SQL Analysis Using HUE Dashboard - Optional	64
6.6. Part 2: Add a New Field - Optional	68
7. Take-aways	80

Foreword

Document Status

This document does not form a contract or offer to contract.

Response Limitation

All products or company names are used for identification purposes only, and maybe trademarks of their respective owners.

Confidentiality

The material contained in this document represents proprietary information pertaining to Cloudera products and methods.

Validity

Cloudera does not take responsibility for the changes and product updates post publication of this document and does not commit to keeping it updated.

Change log

Date	Name	Change
Sept 2023	Alex Campos Simoes	Workshop creation
12 Oct 2023	Cristina Sánchez	Update of Workshop elements
13 Oct 2023	Jacques Marchand	Add optional exercices
16 Oct 2023	Charles Aad	Document merge, sanity check

1. Introduction

At Cloudera we believe that data can make what is impossible today, possible tomorrow. We empower people to transform complex data into clear and actionable insights. Cloudera delivers an enterprise data platform for any data, anywhere, from the Edge to AI and it's powered by the relentless innovation of the open source community, Cloudera advances digital transformation for the world's largest enterprises.

Today, Cloudera offers a mature and operational data lake stack. Since Cloudera is uniquely positioned in the on premises space, private cloud and public cloud, it can deliver a highly differentiated hybrid and multi-cloud vision.

The scope of this workshop is to experiment with the latest stack on public cloud through an overly simplified telco customer churn use case. All this is built for the purposes of experimentation.

This workshop guide is a step by step document to follow in order to deliver the workshop and is completed by

- A preparation guide (internal)
- A presentation designed to facilitate the experimentation. (to be delivered to you)

At Cloudera, we thank you for your confidence and for experimenting with our products.

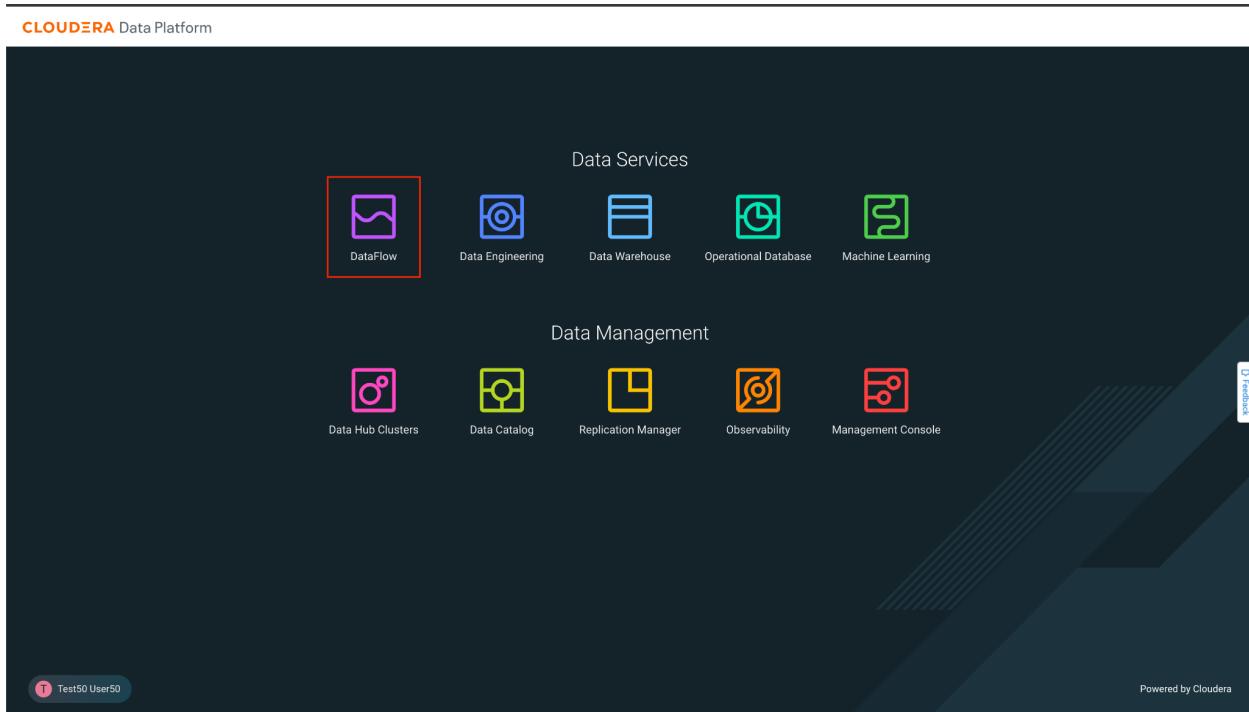
2. Data Flow Lab:

2.1. Goals

- Consume data from a Kafka topic
- Convert the data to Parquet format
- Store the data in a table in the Lakehouse

2.2. Lab 1 - Ingest Kafka streams to Iceberg table

1. Click on DataFlow from CDP PC Home:



2. Once in DataFlow, click on the option **Catalog** from the left menu. The data ingestion application templates are listed here. For the purpose of this workshop, we have created and published a template that allows you to read Kafka topic data and ingest/store it in the Lakehouse provided by CDP Public Cloud. Click on the Flow called **kafka_to_lakehouse** to

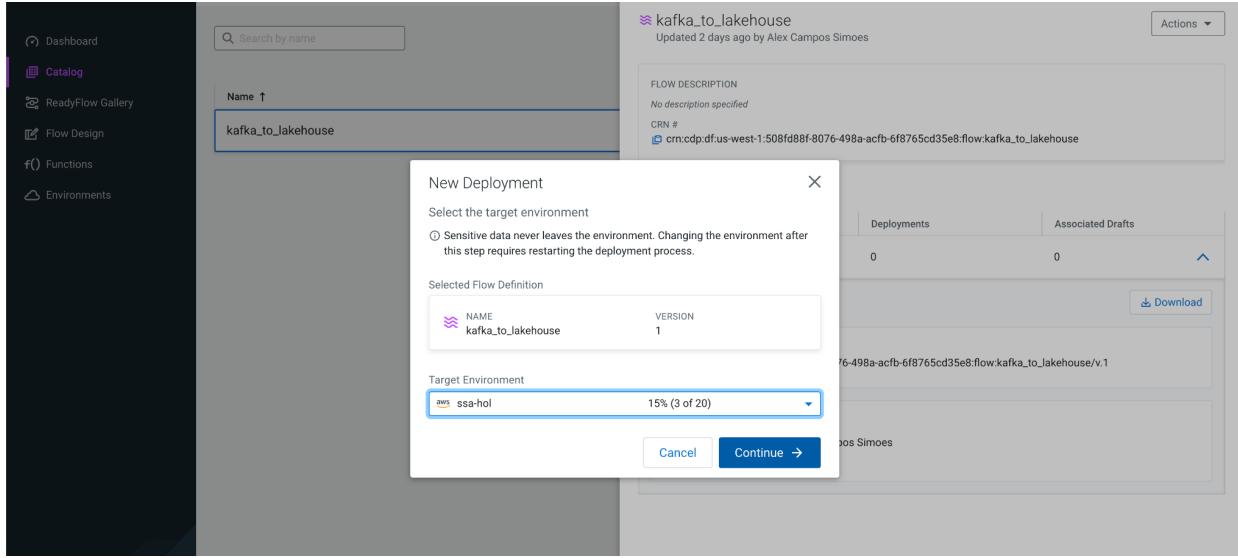
start deploying it.

The screenshot shows the Cloudera DataFlow interface. On the left is a dark sidebar with navigation links: Dashboard, Catalog (which is selected and highlighted in purple), ReadyFlow Gallery, Flow Design, Functions, Environments, Help, and a user profile for Test50 User50. The main content area is titled "Flow Catalog". It features a search bar at the top left and a button "Import Flow Definition" at the top right. Below the search bar is a table header with columns: Name ↑, Type, Versions, and Last Updated. A single row is visible in the table: "kafka_to_lakehouse" (Type: Custom Flow Definition, Versions: 1, Last Updated: 2 days ago). At the bottom of the table are pagination controls: "Items per page: 10", "1 - 1 of 1", and navigation arrows. The overall theme is dark with purple highlights for the active menu item.

3. When clicked, the following panel appears with the Flow information. It shows the available versions, creation date, creator user, and a button **Deploy** to start the deployment. Click on that button.

This screenshot shows a detailed view of the "kafka_to_lakehouse" flow. The left sidebar is identical to the previous screenshot. The main content area has a title "» kafka_to_lakehouse" with a subtitle "Updated 2 days ago by Alex Campos Simoes". To the right is a "Actions" button. Below this is a "FLOW DESCRIPTION" section with the note "No description specified". Underneath is a "CRN #": "crn:cdp:df:us-west-1:508fd88f-8076-498a-acfb-6f8765cd35e8:flow:kafka_to_lakehouse". There is also a checkbox "Only show deployed versions". A table below shows one version: "Version 1", "Deployments 0", and "Associated Drafts 0". A "Deploy" button with an arrow icon is located next to the deployment count. Further down are sections for "CRN #", "CREATED" (date: 2023-05-19 00:15 CEST by Alex Campos Simoes, note: "Initial Version"), and a "Download" button with a download icon. The overall layout is clean with a white background and light gray borders for the different sections.

4. The following popup window allows you to select the DataFlow cluster in which you want to deploy the Flow. In this case, the cluster to be selected is **ssa-hol**. The workshop instructor will tell you which environment to select. Once selected, click **Continue**.



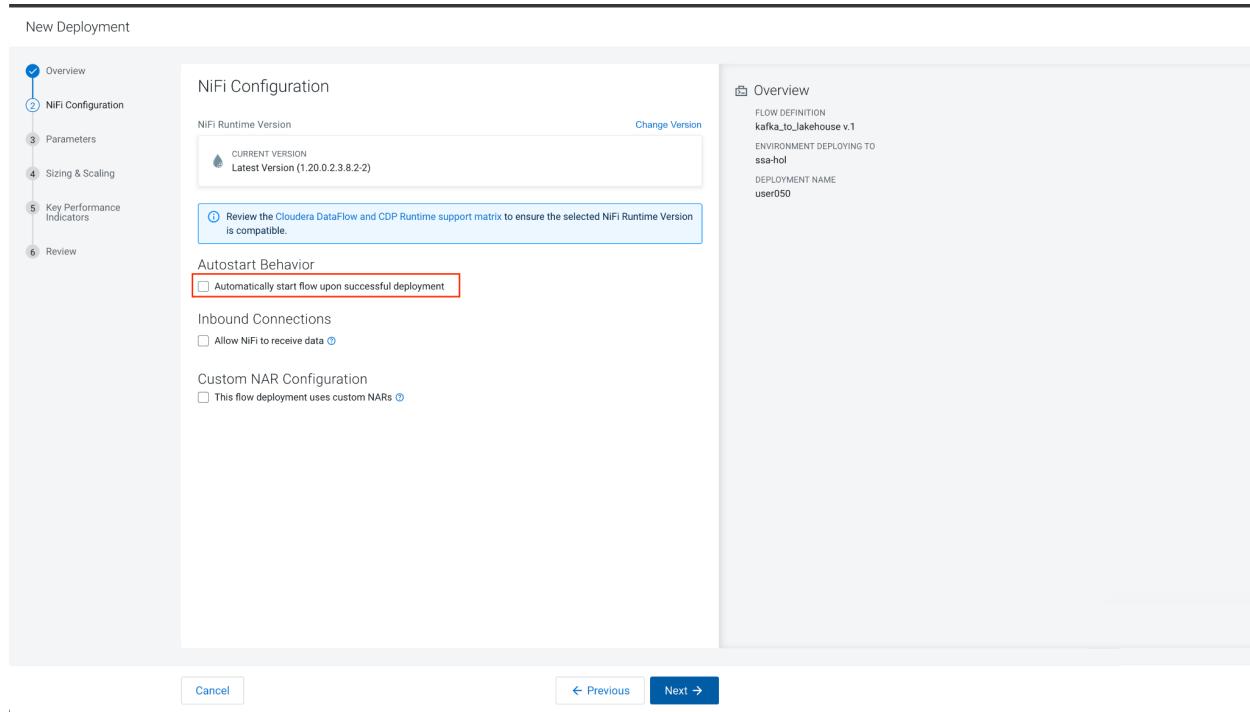
5. From this point, you will need to enter the Flow configuration. Start by assigning a name (**Deployment Name**) and click **Next**.

For the purposes of this workshop, please name the Flow with the assigned username -user050, for example.

The screenshot shows the 'Overview' step of the deployment wizard. The left sidebar lists steps 1 through 6. The main area shows the 'Deployment Name' field set to 'user050', which is highlighted in blue. Below it, the 'Selected Flow Definition' and 'Target Environment' sections are identical to the previous screenshot. At the bottom are 'Cancel' and 'Next →' buttons.

6. Uncheck the option **Automatically start flow upon successful deployment** and click **Next**.

We are going to run Flow step by step, so we don't want it to start automatically.



7. In this part of Parameters, you must enter the following values:

CDP Workload User Password: Enter the Workload Password shared at the beginning of the workshop.

CDP Workload Username: enter the assigned user number, *user050*, for example.

Database: enter the assigned user number, *user050*, for example. This database and the tables are already pre-created for you. We'll review it later.

Kafka Consumer Group Id: Enter a unique value using the assigned user. You can combine with the user id assigned for you.

Review that the parameters were entered correctly. Then click on **Next**.

New Deployment

Parameters

Data entered here never leaves the environment in your cloud account. Provide parameter values directly in the text input or upload a file for parameters that expect a file.

The selected flow definition references an external Default NiFi SSL Context Service. Hence, DataFlow will automatically create a matching SSL Context Service with a keystore and truststore generated from the target environment's FreeIPA certificate.

SHOW: Sensitive No value

parameters (7)

CDP Workload User Password 17/100K
.....

CDP Workload Username 7/100K
user050

CDPEnvironment 0/100K

core-site.xml
ssl-client.xml
hive-site.xml

Select File Drop file or browse

DataFlow automatically adds all required configuration files to interact with Data Lake services. Unnecessary files that are added won't impact the deployment process.

Cancel ← Previous Next →

New Deployment

CDPEnvironment 0/100K

core-site.xml
ssl-client.xml
hive-site.xml

Select File Drop file or browse

DataFlow automatically adds all required configuration files to interact with Data Lake services. Unnecessary files that are added won't impact the deployment process.

Database 7/100K
user050

Kafka Brokers 203/100K
realtime-ingestion-corebroker0.ssa-hol.yu1t-vbzg.cloudera.site:9093,realtime-ingestion-corebroker1.ssa-hol.yu1t-vbzg.cloudera.site:9093,realtime-ingestion-corebroker2.ssa-hol.yu1t-vbzg.cloudera.site:9093

Kafka Consumer Group Id 16/100K
Consumer_user050

Kafka Topic 10/100K
telco_data

Cancel ← Previous Next →

8. There is no need to configure auto scaling parameters, then click on **Next**

New Deployment

Sizing & Scaling
Select the NiFi node size and the number of nodes provisioned for your flow.

NiFi Node Sizing

<input checked="" type="radio"/> Extra Small	<input type="radio"/> Small	<input type="radio"/> Medium	<input type="radio"/> Large
2 vCores Per Node 4 GB Per Node	3 vCores Per Node 6 GB Per Node	6 vCores Per Node 12 GB Per Node	12 vCores Per Node 24 GB Per Node

Number of NiFi Nodes

Auto Scaling Disabled

Nodes:

Overview
FLOW DEFINITION: kafka_to_lakehouse v.1
ENVIRONMENT DEPLOYING TO: ssa-hol
DEPLOYMENT NAME: user050

NiFi Configuration
NIFI RUNTIME VERSION: Latest Version (1.20.0.2.3.8.2-2)
AUTO-START FLOW: No
INBOUND CONNECTIONS: No
CUSTOM NAR CONFIGURATION: No

Parameters
parameters
COP WORKLOAD USER PASSWORD: [Sensitive Value Provided]
COP WORKLOAD USERNAME: user050
COPENVIRONMENT: core-site.xml
ssl-client.xml
hive-site.xml
DATABASE: user050
KAFKA BROKERS: realtime-ingestion-corebroker0.ssa-hol.yu1t-vbzg.cloudera.site:9093,realtime-ingestion-corebroker1.ssa-hol.yu1t-vbzg.cloudera.site:9093,realtime-ingestion-corebroker2.ssa-hol.yu1t-vbzg.cloudera.site:9093

Cancel **← Previous** **Next →**

9. We are also not going to configure KPIs by now, then click on **Next** to continue the configuration.

New Deployment

Key Performance Indicators
Set up KPIs to track specific performance metrics of a deployed flow. Click and drag to reorder how they are displayed.
[Learn more](#)

Add New KPI

Overview
FLOW DEFINITION: kafka_to_lakehouse v.1
ENVIRONMENT DEPLOYING TO: ssa-hol
DEPLOYMENT NAME: user050

NiFi Configuration
NIFI RUNTIME VERSION: Latest Version (1.20.0.2.3.8.2-2)
AUTO-START FLOW: No
INBOUND CONNECTIONS: No
CUSTOM NAR CONFIGURATION: No

Parameters
parameters
COP WORKLOAD USER PASSWORD: [Sensitive Value Provided]
COP WORKLOAD USERNAME: user050
COPENVIRONMENT: core-site.xml
ssl-client.xml
hive-site.xml
DATABASE: user050
KAFKA BROKERS: realtime-ingestion-corebroker0.ssa-hol.yu1t-vbzg.cloudera.site:9093,realtime-ingestion-corebroker1.ssa-hol.yu1t-vbzg.cloudera.site:9093,realtime-ingestion-corebroker2.ssa-hol.yu1t-vbzg.cloudera.site:9093

Cancel **← Previous** **Next →**

10. Review all the information entered for your Flow, then click on **Deploy** to start the deployment process.

The screenshot shows the 'Review' step of a 'New Deployment' process. On the left, a sidebar lists steps: Overview, NiFi Configuration, Parameters, Sizing & Scaling, Key Performance Indicators, and Review (which is selected). The main area displays the following details:

- Overview**:
 - FLOW DEFINITION: kafka_to_lakehouse.v1
 - ENVIRONMENT DEPLOYING TO: ssa-hol
 - DEPLOYMENT NAME: user050
- NiFi Configuration**:
 - NIFI RUNTIME VERSION: Latest Version (1.20.0.2.3.8.2-2)
 - AUTO-START FLOW: No
 - INBOUND CONNECTIONS: No
 - CUSTOM NAR CONFIGURATION: No
- Parameters**:
 - parameters
 - CDF WORKLOAD USER PASSWORD: [Sensitive Value Provided]
 - CDF WORKLOAD USERNAME: user050
 - COPENVIRONMENT
 - core-site.xml
 - ssl-client.xml
 - hive-site.xml
 - DATABASE: user050
 - KAFKA BROKERS

At the bottom are 'Cancel', 'Previous', and 'Deploy' buttons.

11. The blue box indicates that the Flow deployment process has been started. By clicking on the button **Load More** you will be able to see the different stages of the deployment. After about 60 to 90 seconds approximately, the last event should be *Deployment Successful*.

The screenshot shows the Cloudera DataFlow 'Dashboard'. On the left, a sidebar includes links for Dashboard, Catalog, ReadyFlow Gallery, Flow Design, Functions, Environments, Help, and a note for 'Test50 User50'. The main dashboard area shows a table of flows, with one entry for 'user050' in the 'ssa-hol' environment, currently in a 'Deploying' state. To the right, a detailed view for 'user050' shows an 'Alerts' section with a message: 'Deployment Initiated' (Initiated deployment of [user050]). A red box highlights this message. Below it, the 'Event History' section shows a single event: 'Deployment Initiated' at '2023-05-21 00:09 CEST'. A 'Load More' button is visible at the bottom of the history list.

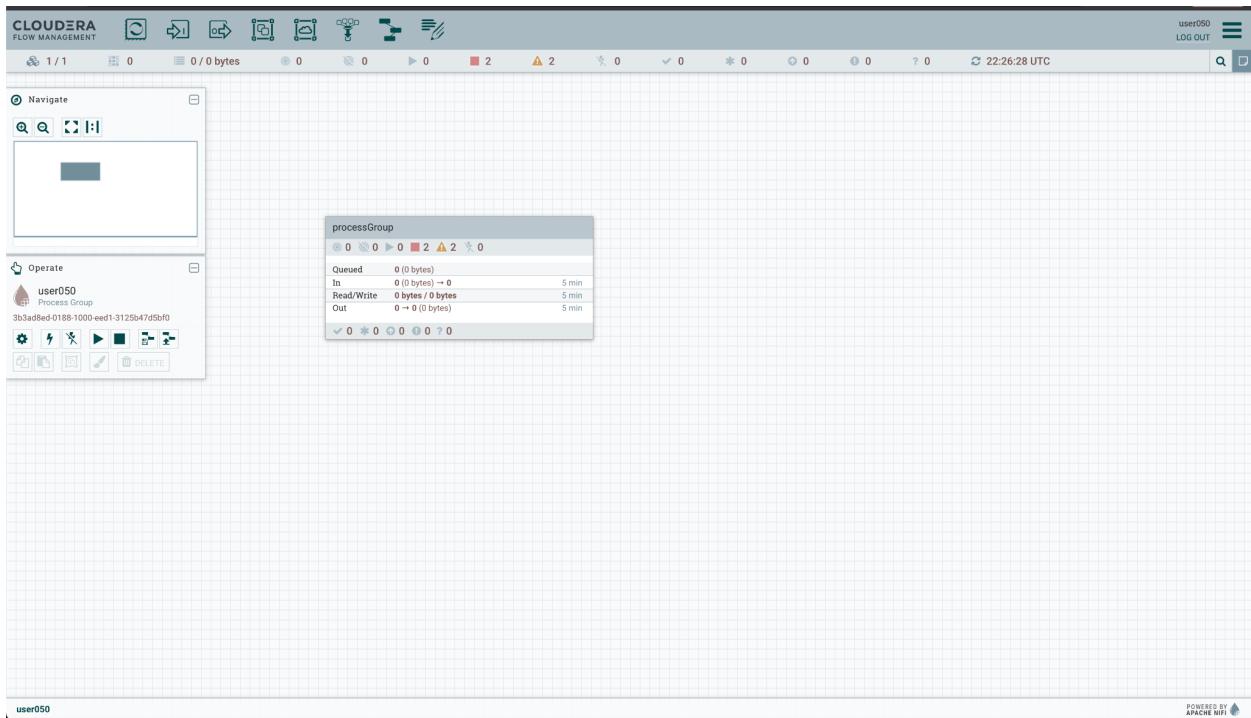
12. Once the deployment is finished, click on **Manage Deployment** to see the details of the recently deployed Flow.

The screenshot shows the Cloudera DataFlow interface. On the left is a sidebar with icons for Dashboard, Catalog, ReadyFlow Gallery, Flow Design, Functions, and Environments. The main area is titled 'Dashboard' and shows a table with one row for 'user050'. The table has columns for Status (Deploying) and Name (user050). Below the table, there's a section for 'Active Alerts' which says 'No alerts to display.' and a 'Event History' table with several log entries. A red box highlights the 'Manage Deployment' button at the top right of the dashboard area.

13. In this window you will see the Flow information displayed. It is time to execute the application processes from the graphical Flow Management interface. Click on **Actions -> View in NiFi**, to open Cloudera Flow Management canvas in a new window/tab.

The screenshot shows the 'Deployment Manager' page for 'user050'. The left sidebar is identical to the previous screenshot. The main area displays deployment details: STATUS (Suspended), DEPLOYMENT NAME (user050), FLOW DEFINITION (kafka_to_lakehouse V.1), NODE COUNT (1), AUTO SCALING (Disabled), REGION (US East(N. Virginia)), and NIFI RUNTIME VERSION (1.20.0.2.3.8.2-2). To the right of these details is an 'Actions' dropdown menu with options: View in NiFi, Start flow, Change NiFi Runtime Version, Restart Deployment, and Terminate. Below the details, there are sections for 'Deployment Settings' (with tabs for KPIs and Alerts, Sizing and Scaling, Parameters, and NiFi Configuration) and 'Key Performance Indicators' (with a placeholder for 'Add New KPI'). At the bottom are buttons for Discard Changes, Apply Changes, and Update Deployment CLI Command.

14. In the new window you should be able to see the Flow Management canvas with one process group (a box). The canvas is where the Flow Management applications are built. Double click on the box; the only visible box, which is a Process Group and should be titled **processGroup**.



15. When opening the Process Group, you should be able to see the Processors that compose the Flow application. To summarize, there are four Processors:

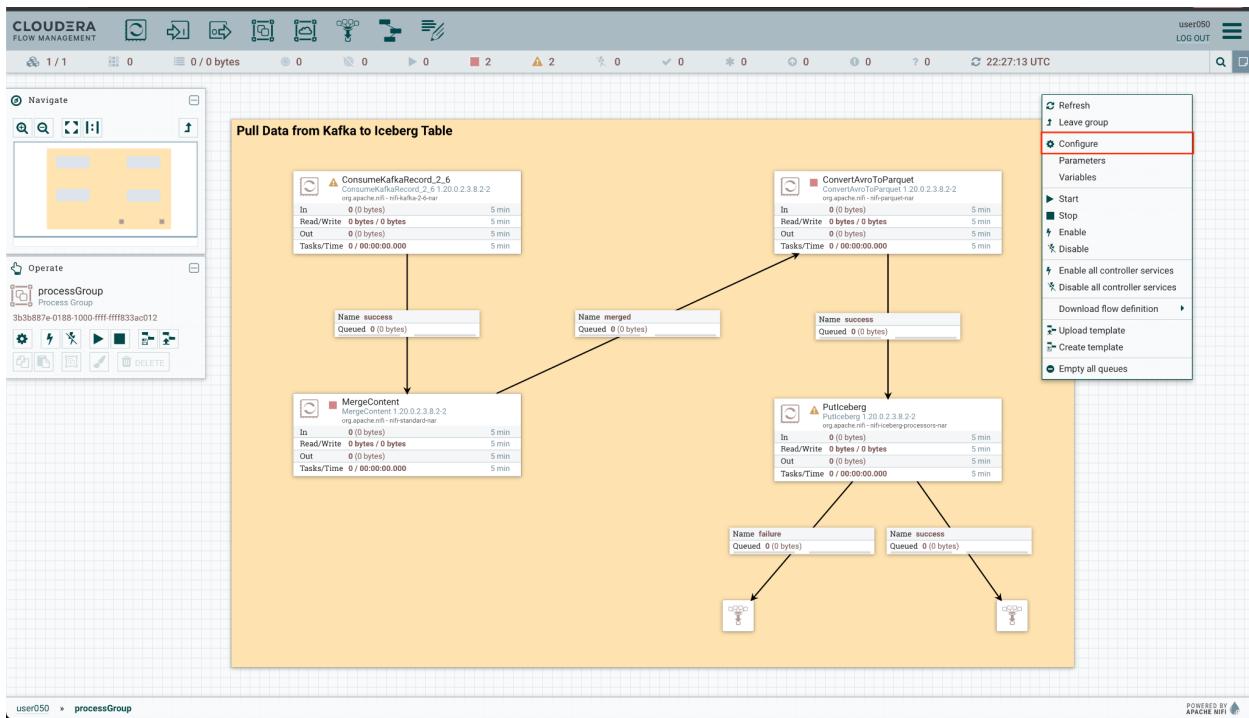
ConsumeKafkaRecord, processor to consume data from the Kafka topic, reading the data in JSON format and outputting in AVRO format.

MergeContent, to group the flow files and streamline the data flow.

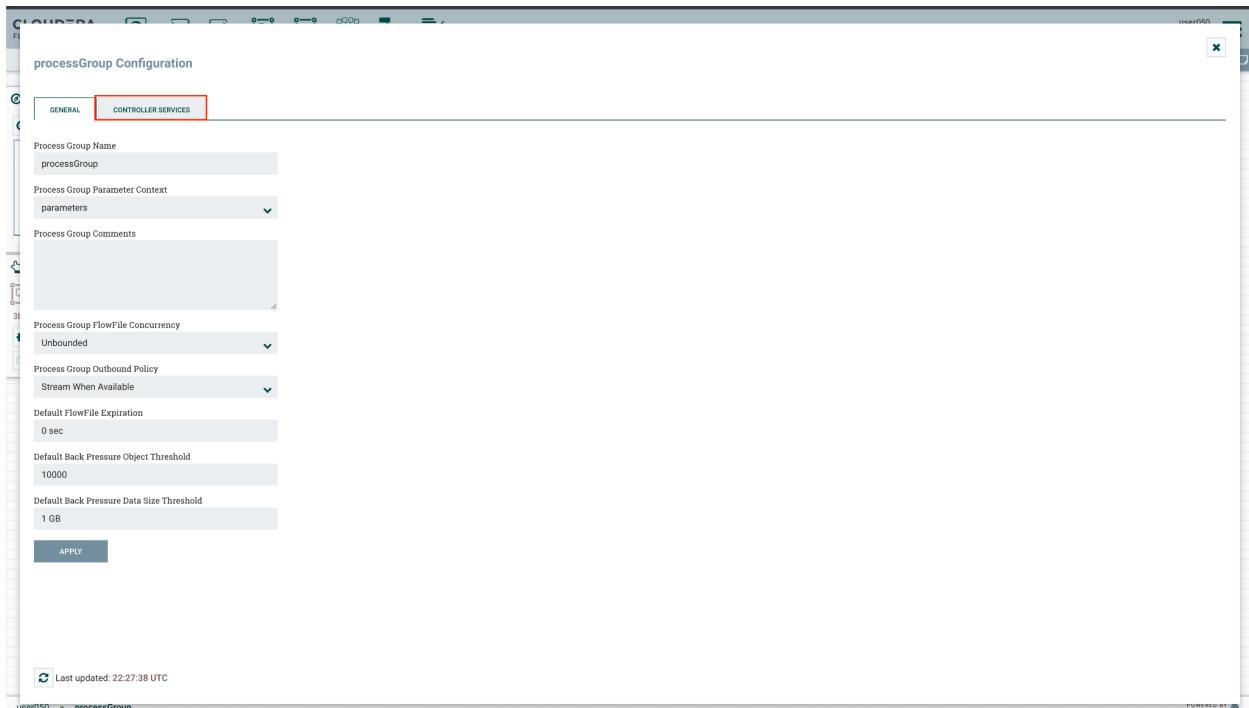
ConvertAvroToParquet, conversion needed to store the data in PARQUET format.

PutIceberg, to insert the data into the table in the Lakehouse. The destination table is called `telco_kafka_iceberg`, and each user has an assigned database (`user_id` is the name of the database).

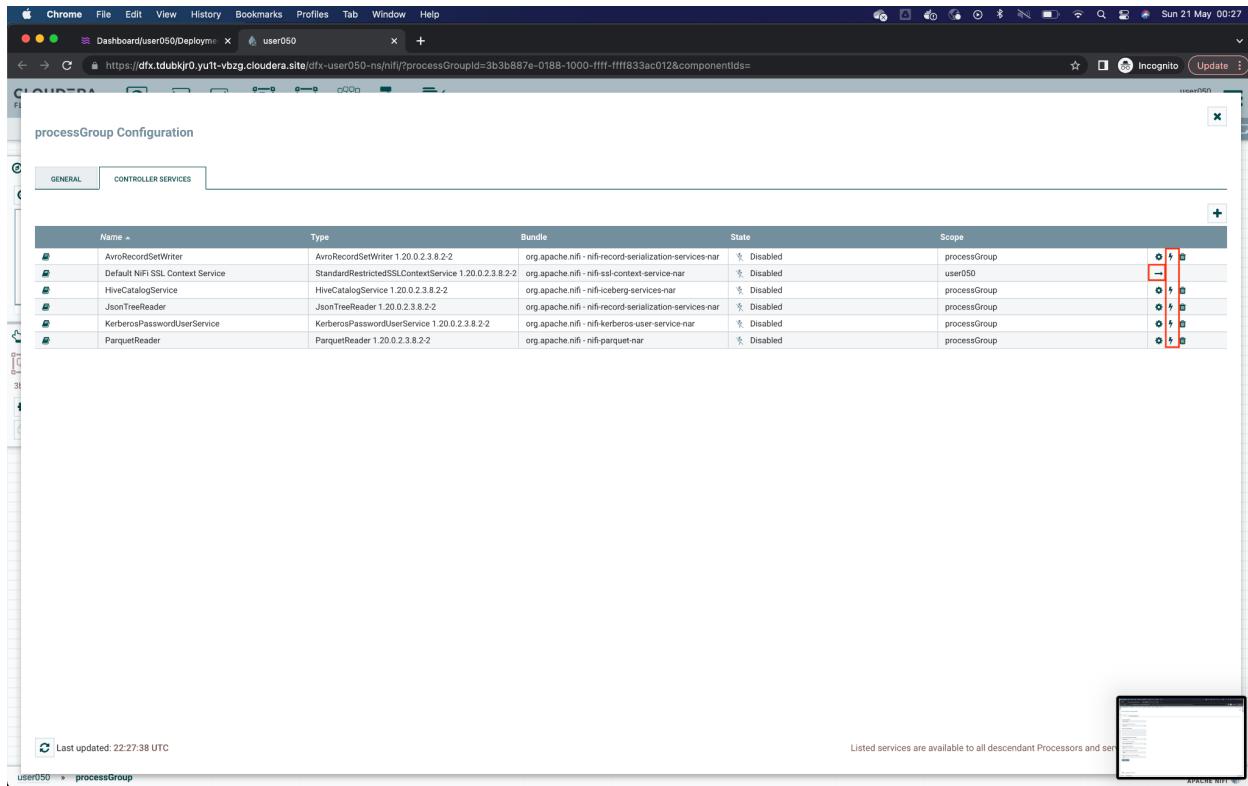
As you can see, the Processors are not started, and some have an error message/alert icon. The latter is because there are components of the data flow that must be activated before. To activate them - the *Controller Services* - right click on the canvas and click on the option **Configure** from the floating menu that appears.



16. In the pop-up window that opens, select the tab Controller Services.

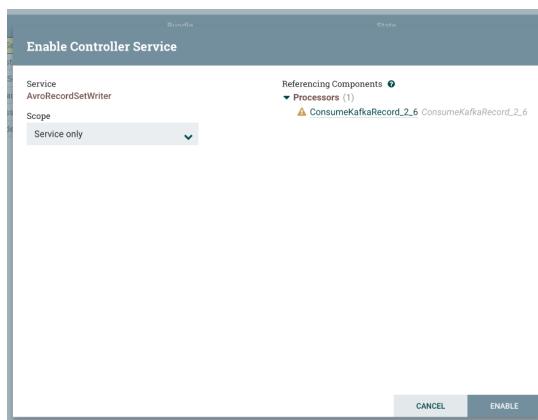


17. The **Controller Services** of the data flow. Each of them must be activated. The following Controllers must be activated first: **AvroRecordSetWriter**, **HiveCatalogService**, **JsonTreeReader**, **KerberosPasswordUserService** and **ParquetReader** clicking on the icon lightning  which appears on the right (marked in red).

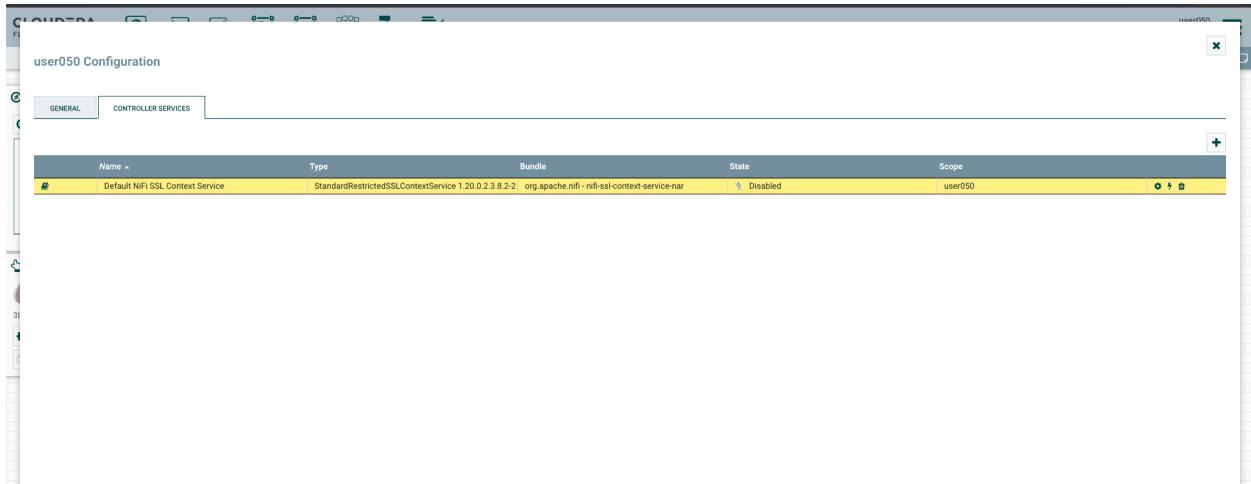


Name	Type	Bundle	State	Scope
AvroRecordSetWriter	AvroRecordSetWriter 1.20.0.2.3.8.2-2	org.apache.nifi - nifi-record-serialization-services-nar	Disabled	processGroup
Default NIFI SSL Context Service	StandardRestrictedSSLContextService 1.20.0.2.3.8.2-2	org.apache.nifi - nifi-ssl-context-service-nar	Disabled	user050
HiveCatalogService	HiveCatalogService 1.20.0.2.3.8.2-2	org.apache.nifi - nifi-iceberg-services-nar	Disabled	processGroup
JsonTreeReader	JsonTreeReader 1.20.0.2.3.8.2-2	org.apache.nifi - nifi-record-serialization-services-nar	Disabled	processGroup
KerberosPasswordUserService	KerberosPasswordUserService 1.20.0.2.3.8.2-2	org.apache.nifi - nifi-kerberos-user-service-nar	Disabled	processGroup
ParquetReader	ParquetReader 1.20.0.2.3.8.2-2	org.apache.nifi - nifi-parquet-nar	Disabled	processGroup

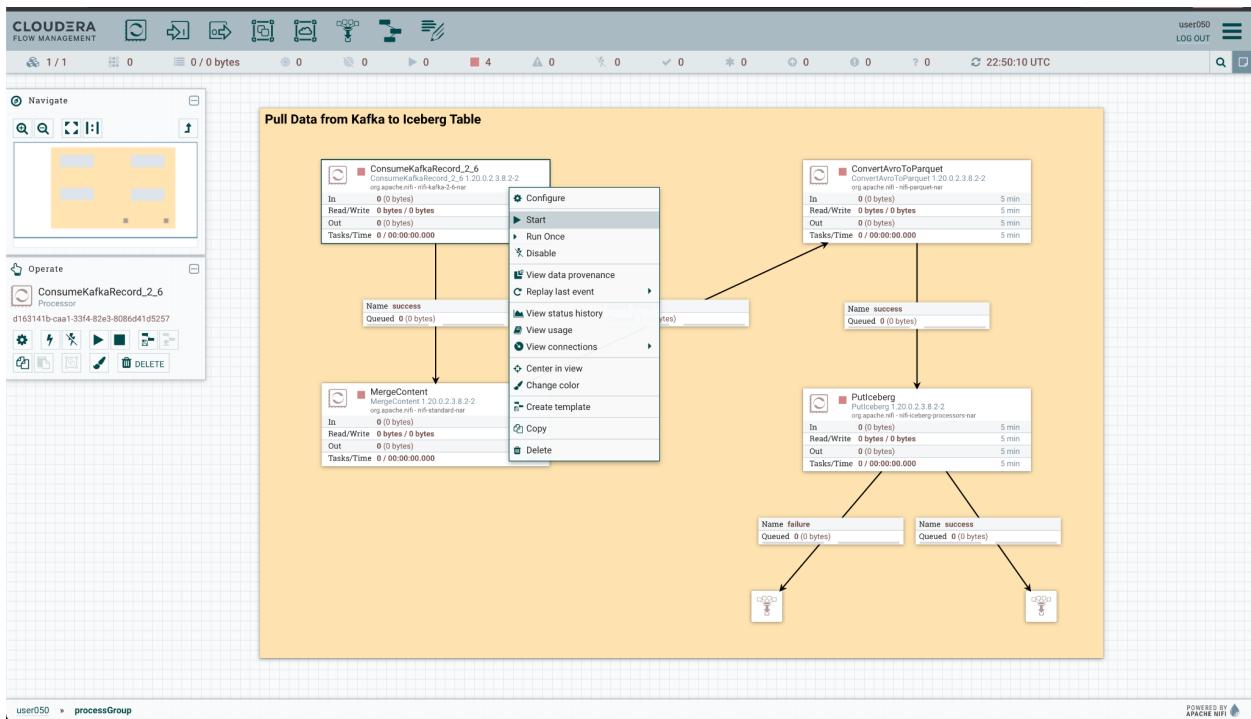
Click the button **Enable** in the enable confirmation window of each Controller Service. Then close that window to enable the next Controller Service.



To activate the Default NiFi SSL Context Service, you must click on the arrow →. Finally clicking on the lightning bolt icon ⚡ controller service is activated **Default NiFi SSL Context Service**, which will also present a window to enable it.

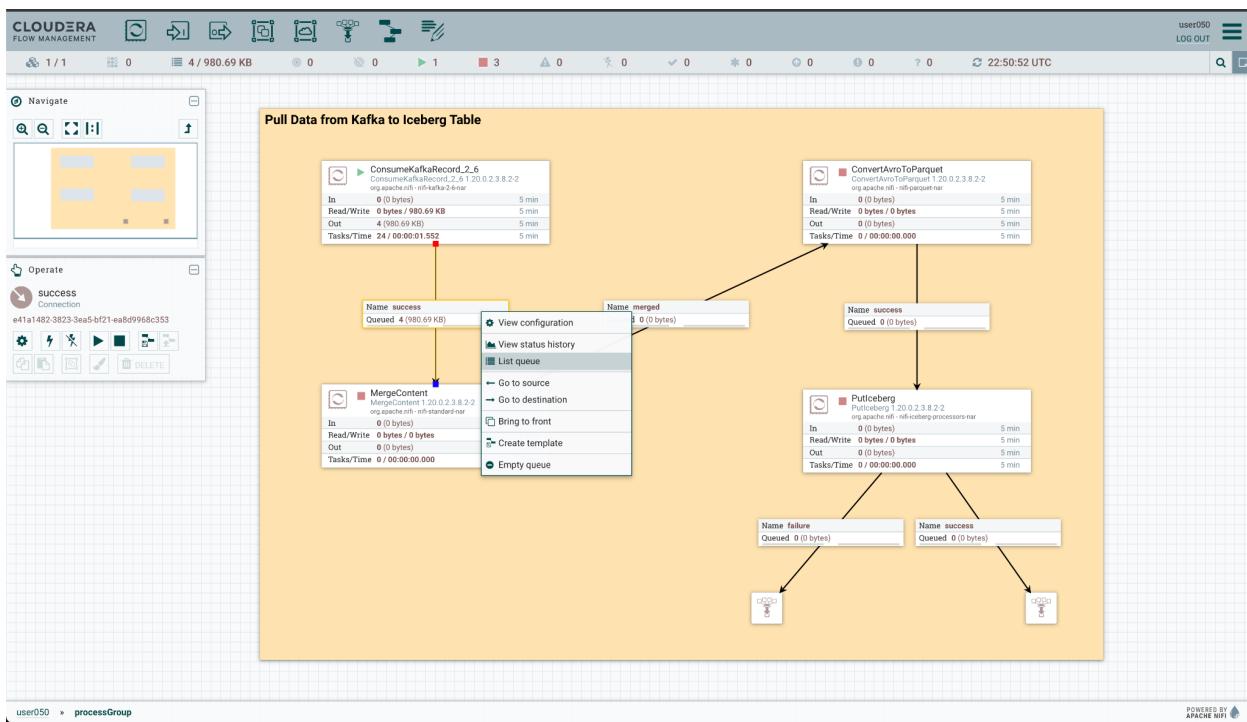


18. Close the Controller Services window, making sure all are enabled. Return to the Process Group by double-clicking on it. It's time to execute **Processors**. Start with **ConsumeKafkaRecord**, by right-clicking on it, and then clicking on **Start**. This will start consuming the Kafka topic data.

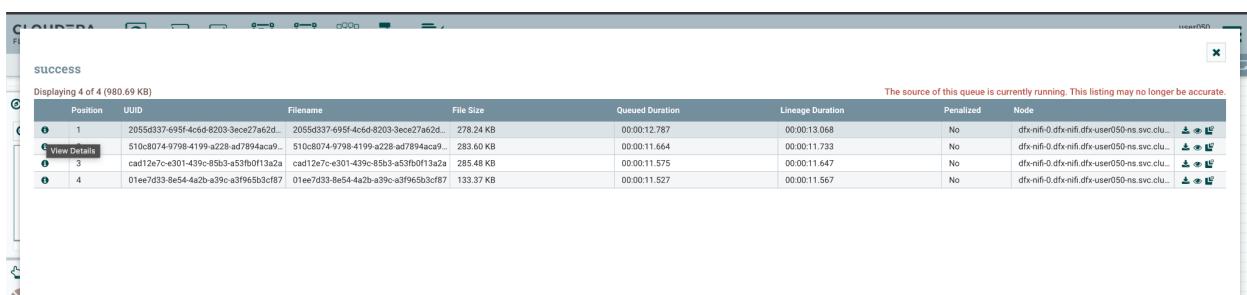


19. Flow Management allows us to see and access data in motion during the execution of the data flow. Between Processors **ConsumeKafkaRecord** (just started) and **MergeContent**, there is a connection. This connection is what joins the Processors and transmits data from one to the other.

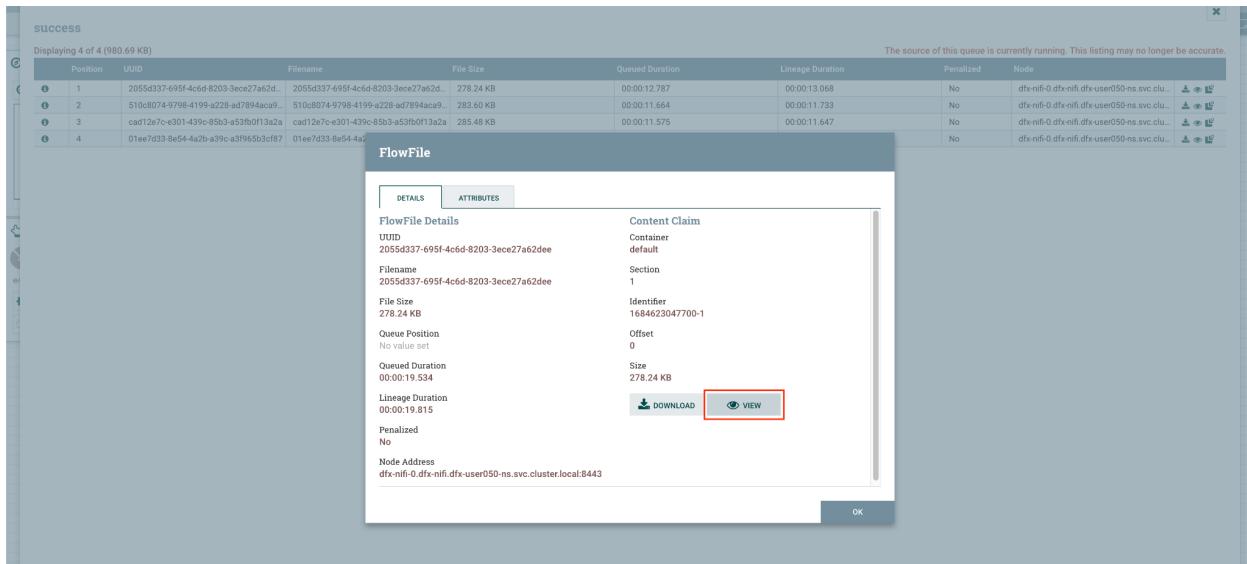
To check how much data is queued on this connection, refresh the counter by pressing the Ctrl+R (Windows) or Command+R (Mac) combination on the keyboard. This will allow the current metrics of the entire data stream to be updated. At some point there should be a number next to the legend **Queued** in the connection between **ConsumeKafkaRecord** and **MergeContent**. To see the queued data, right click on the connection and click on the option **List Queue**, opening a popup window.



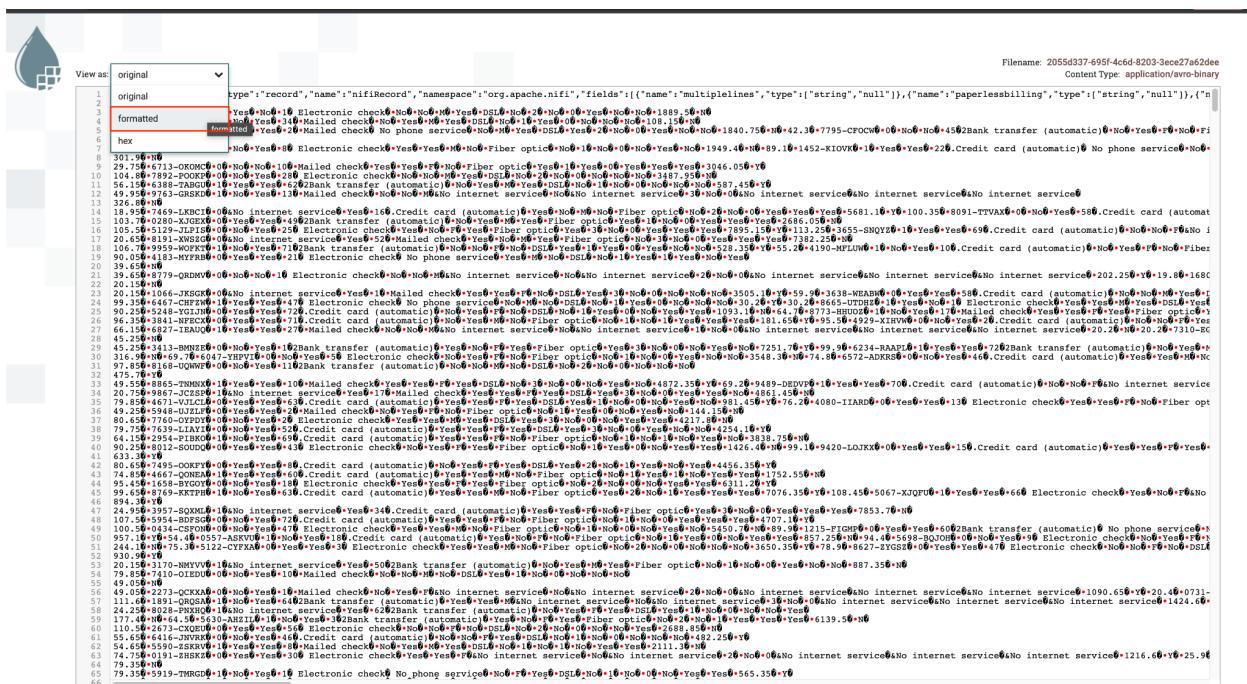
20. The next popup window lists the queued data. Click on the information icon (i) that appears on the left side to view the events.



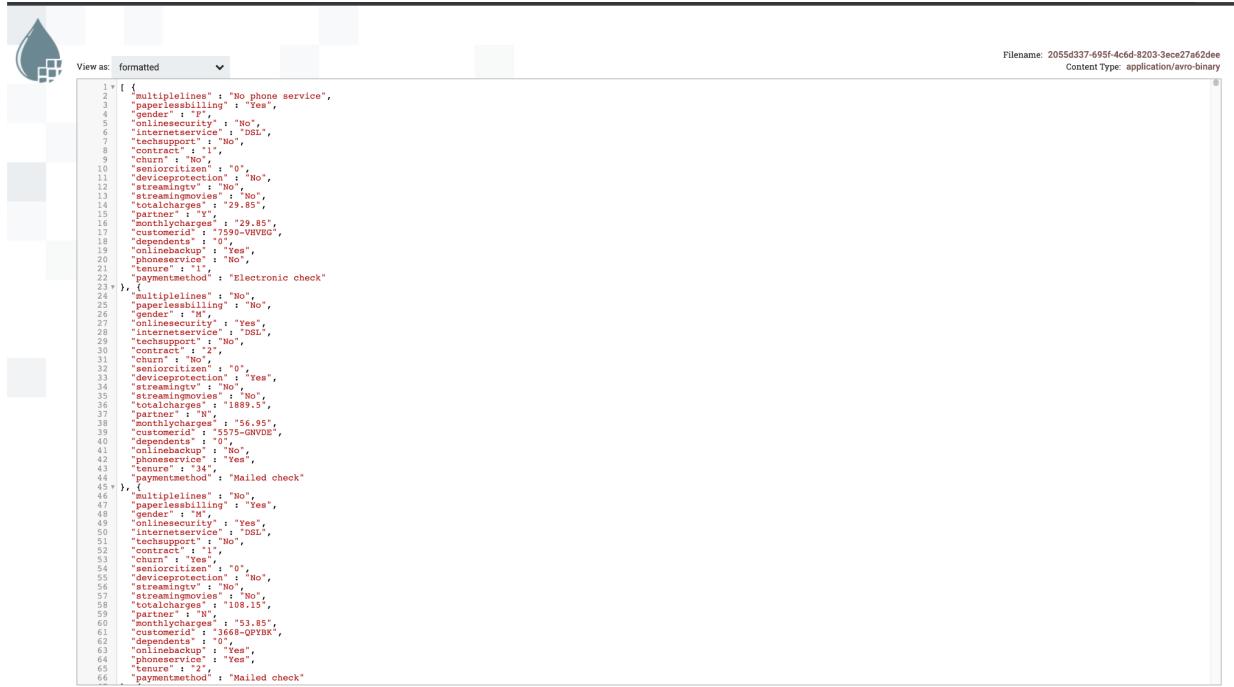
21. Once the FlowFile detail window appears, click on the button **VIEW** to open the content of consumed events.



22. The new window that opens shows the data of the FlowFile content. Being in AVRO format, it is not fully readable. A deserializer must be selected to correctly display the data. For this, in the upper left, select the option **formatted** from the menu **View as**.



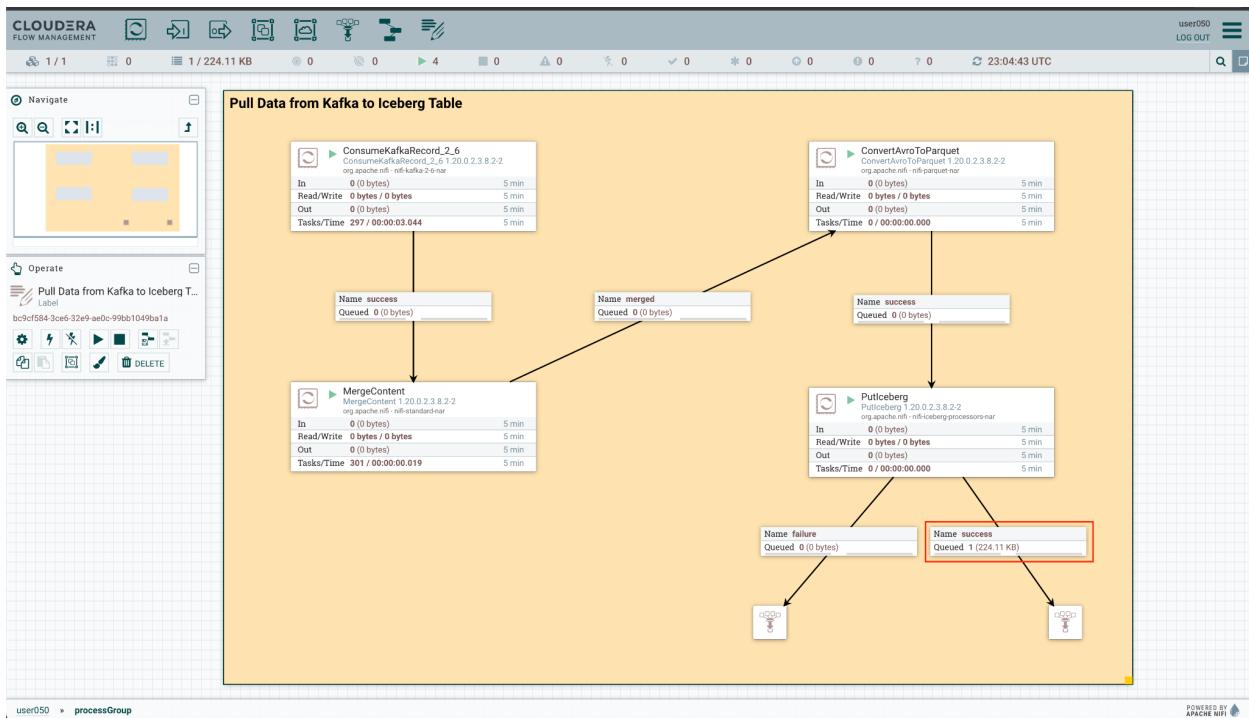
23. Now you can display the data correctly. Notice that the fields or attributes indicated at the beginning of the workshop appear. You can close that FlowFile window and the popups, returning to the canvas with the four Processors.



```
View as: formatted
1  {
2    "multipelines": "No phone service",
3    "paperlessbilling": "Yes",
4    "gender": "P",
5    "onlinesecurity": "No",
6    "internetservice": "DSL",
7    "techsupport": "No",
8    "contract": "1",
9    "churn": "No",
10   "seniorcitizen": "0",
11   "deviceprotection": "No",
12   "streamingtv": "No",
13   "streamingmovies": "No",
14   "totalcharges": "29.85",
15   "partner": "Y",
16   "monthlycharges": "29.85",
17   "customerid": "3100-WWBG",
18   "dependents": "0",
19   "onlinebackup": "Yes",
20   "phoneservice": "No",
21   "tenure": "1",
22   "paymentmethod": "Electronic check"
23 },
24 {
25   "multipelines": "No",
26   "paperlessbilling": "No",
27   "onlinesecurity": "Yes",
28   "internetservice": "DSL",
29   "techsupport": "No",
30   "contract": "2",
31   "churn": "Yes",
32   "seniorcitizen": "0",
33   "deviceprotection": "Yes",
34   "streamingtv": "No",
35   "streamingmovies": "No",
36   "totalcharges": "1889.5",
37   "customerid": "3575-QNVD",
38   "monthlycharges": "56.95",
39   "dependents": "1",
40   "onlinebackup": "No",
41   "phoneservice": "Yes",
42   "tenure": "34",
43   "paymentmethod": "Mailed check"
44 },
45 {
46   "multipelines": "No",
47   "paperlessbilling": "Yes",
48   "onlinesecurity": "Yes",
49   "internetservice": "DSL",
50   "techsupport": "Yes",
51   "contract": "1",
52   "churn": "Yes",
53   "seniorcitizen": "0",
54   "deviceprotection": "No",
55   "streamingtv": "No",
56   "streamingmovies": "No",
57   "totalcharges": "108.15",
58   "dependents": "1",
59   "monthlycharges": "53.85",
60   "customerid": "3668-QPYBK",
61   "onlinebackup": "Yes",
62   "phoneservice": "Yes",
63   "tenure": "2",
64   "paymentmethod": "Mailed check"
65 },
66 }
```

24. Continue running each of the Processors in order:**MergeContent**, after **ConvertAvroToParquet** and finally **PutIceberg**. Remember that you can refresh the flow counters with the combination Control+R or Command+R.

If the previous steps were executed correctly, the connection of the Processor **PutIceberg** to a funnel should be of type **success**.



2.3. Lab 2 - Stream Messaging Manager - Optional

1. On your Cloudera Data Platform landing page,
 - o Click on DataHub Clusters

2. The list of all your cluster loads

- Click on mtn-streams

The screenshot shows the Cloudera Management Console interface. On the left is a dark sidebar with various navigation options like Dashboard, Environments, Data Lakes, User Management, Data Hub Clusters (which is currently selected), Data Warehouses, ML Workspaces, Classic Clusters, Audit, Consumption, Shared Resources, and Global Settings. Below the sidebar, there's a help section and a user info section for 'Test50 User50'. The main content area is titled 'Data Hubs' and shows a table of cluster details. The table has columns for Status, Name, Cloud Provider, Environment, Data Hub Type, Version, Node Count, and Created. One row is visible: 'Running mtn-streams aws paris-atelier 7.2.17 - Streams Messaging Light Duty: Apache Kafka, Schema Registry, Streams Messaging Manager, Streams Replication Manager, Cruise Control CDH 7.2.17 4 10/09/23, 07:29 PM GMT+2'. At the bottom of the table, there are pagination controls (1-1 of 1) and a dropdown for 'Items per page'.

3. Your management page for the datalake loads. You can manage and scale your data lake, review history, endpoints, tags, telemetry.

- Click on Streams Messaging Manager

The screenshot shows the 'mtn-streams' cluster details page. The top navigation bar includes 'Data Hubs / mtn-streams / Autoscaling' and 'Actions' buttons. The main content area is divided into several sections: 'mtn-streams' (status: Running, nodes: 4, created at: 10/09/23, 07:29 PM GMT+2, cluster template: 7.2.17 - Streams Messaging Light Duty: Apache Kafka, Schema Registry, Streams Messaging Manager, Streams Replication Manager, Cruise Control); 'Status Reason' (Synced instance states with the cloud provider); 'Environment Details' (aws, environment: paris-atelier, data lake: paris-atelier-dl, credential: paris-atelier, region: us-east-1, availability zone: us-east-1a); 'Services' (CM UI, Schema Registry, Streams Messaging Manager, Token Integration); 'Cloudera Manager Info' (CM URL: https://mtn-streams-gateway.paris-at.pr0u-qsh6.cloudera.site/mtn-streams/cdp-proxy/cm/home/, CM version: 7.11.0, runtime version: 7.2.17-1.cdh7.2.17.p100.44441663, logs: Command logs, Service logs); and 'Autoscale' (event history, upgrade, autoscale, endpoints: 6). A note says 'Currently autoscale is disabled'.

4. Your streams messaging page opens. It has a list of producers, brokers, topics, and consumer groups.

- Click on topics

- Click on the topic: telco data

When loaded, this page will include all the data coming for your telco data CDRs.

Cluster: kafka-d91e User:050

Topics 1 BROKERS 3

NAME	DATA IN	DATA OUT	MESSAGES IN	CONSUMER GROUPS	CURRENT LOG SIZE
telco_data	0B	0B	0	0	0B

Consumer Groups (0)

- Then Click on explore (small icon to the right of the magnifying glass)

5. Your topics page loads, Click on Data Explorer

Topics / telco_data

METRICS ASSIGNMENT DATA EXPLORER CONFIGS LATENCY ACTIONS

Producers (2)

MESSAGES	LEADER	PARTITION	DATA IN	DATA OUT	LOG SIZE
producer-1 0	1546335837	P0	bytes in: 0B	bytes out: 0B	log-size: 0B
producer-2 0	1546335858	P1	bytes in: 0B	bytes out: 0B	log-size: 0B
	1546335816	P2	bytes in: 0B	bytes out: 0B	log-size: 0B
	1546335837	P3	bytes in: 0B	bytes out: 0B	log-size: 0B
	1546335858	P4	bytes in: 0B	bytes out: 0B	log-size: 0B

Consumer Groups (0)

Messages Consumed

No Data

End-to-end Latency

No Data

Summary

Data In Gauge 0

Number of Replicas 1

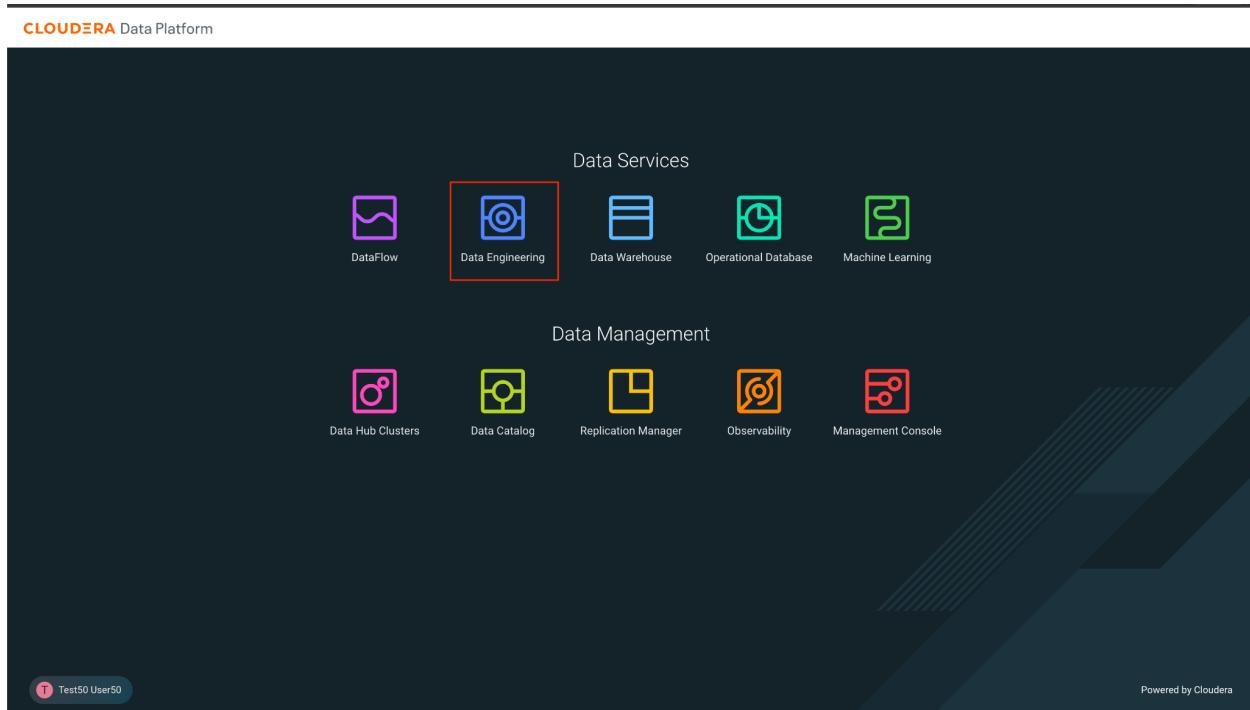
3. Data Engineering

3.1. Goals

- Run a data enrichment process
- Run a process to simulate changes to the data
- Configure the execution of a pipeline using low-code/no-code tools

3.2. Lab 1 - Enrich the Ingested Iceberg table

1. Click on DataFlow from CDP PC Home:



2. The Data Engineering Home shows all the actions that can be done, such as Jobs in Spark and pipelines in Airflow, Resources and useful information/documentation. Click on the option **Jobs** from the left menu to create a dataflow in Airflow.

The screenshot shows the Cloudera Data Engineering Home page. On the left, there is a sidebar with the following navigation options:

- Home
- Jobs
- Job Runs
- Sessions **Preview**
- Resources
- Administration

The main content area is titled "Welcome, Test50". It features several sections:

- Create**: Sub-sections include Spark Jobs (Create New, Schedule, Ad-Hoc Run), Airflow Pipelines (Upload DAG file, Build a Pipeline New), Sessions (New, Start New).
- Resources**: Sub-sections include File (Create New) and Python (Create New).
- Docs & Downloads**: Sub-sections include References (API Doc, Product Doc, Release Notes), Downloads (CLI Client, Migration Tool New).
- Virtual Clusters**: A section for Autoscaling Spark clusters to run Jobs. It shows one cluster named "aws ssa-de" with a single job named "ssa-de-cluster" (Spark 3.2.3). Resource usage is shown as 0 CPU, 0 MB MEMORY, and 0 JOBS.

3. Here the available tasks are listed. For the purposes of this workshop, two Jobs have been configured:

- **CDE-Table-Update**, generate random changes and enrich table to visualize Lakehouse Time Travel functionality.
- **CDE-Data-Enrichment**, process in Spark (Python) to enrich the data ingested from Kafka and save to a new table.

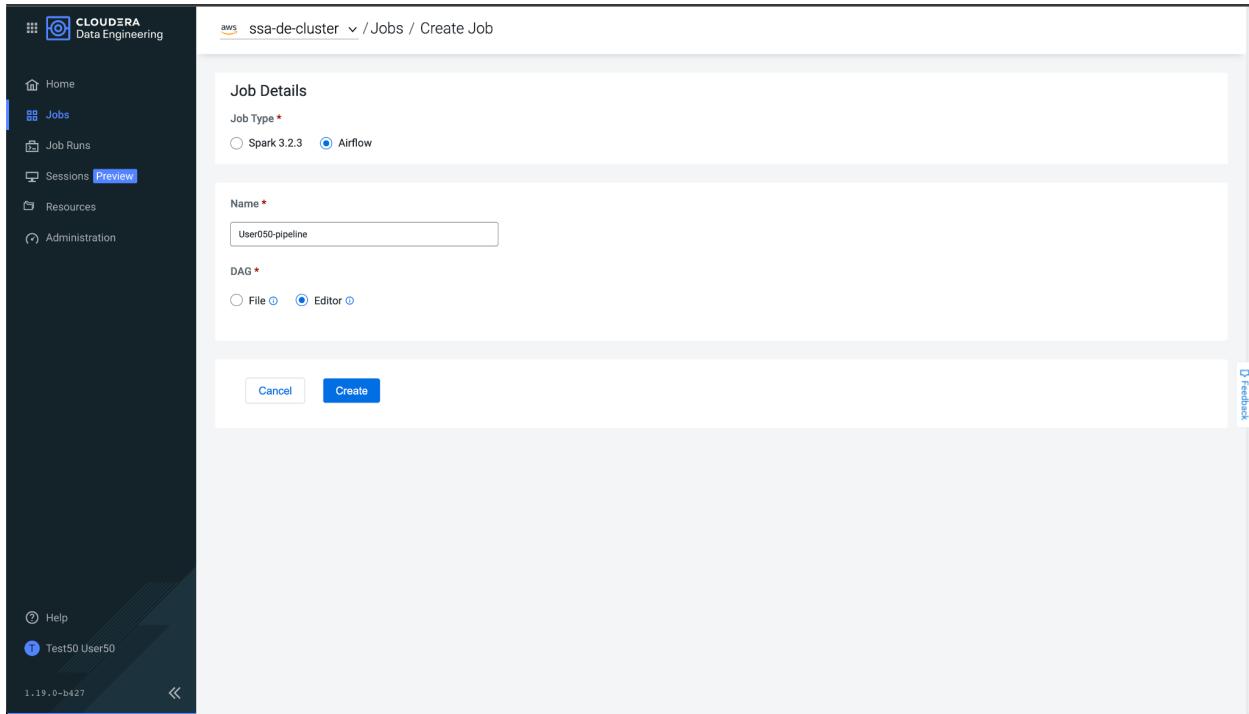
It is time to create our Job in Airflow. Click on **Create Job**.

The screenshot shows the Cloudera Data Engineering interface. On the left, there is a sidebar with the following navigation options: Home, Jobs (which is selected and highlighted in blue), Job Runs, Sessions (with a 'Preview' link), Resources, Administration, Help, and a user profile for 'Test50 User50'. Below the sidebar, the version number is listed as '1.19.0-b427'. The main content area is titled 'aws ssa-de-cluster / Jobs'. It features a search bar, filter dropdowns for 'Status' and 'Type', and a 'Create Job' button. A table lists two jobs: '_CDE-Table-Update' and '_CDE-Data-Enrichment', both of which are Spark type, Ad-Hoc schedule, and were modified on May 26, 2023. The table includes columns for Status, Job, Type, Schedule, Modified On, and Actions. At the bottom, there are pagination controls for 'Items per page: 10' and '1 - 2 of 2'.

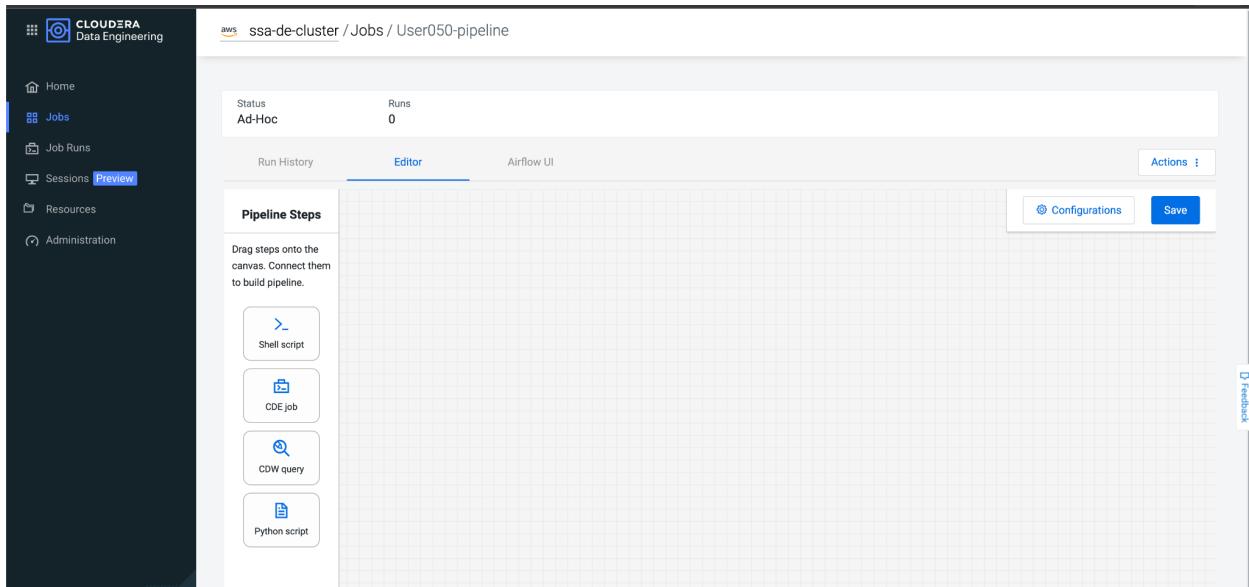
4. In the Job creation form, you must enter the following information:

- Job Type: Airflow
- Name: Use the naming <assigned user>-pipeline. Replace <assigned user> with the user assigned to you. For example, user050
- DAG: Editor, to graphically configure the task.

Once entering the values correctly, click on**Create**.

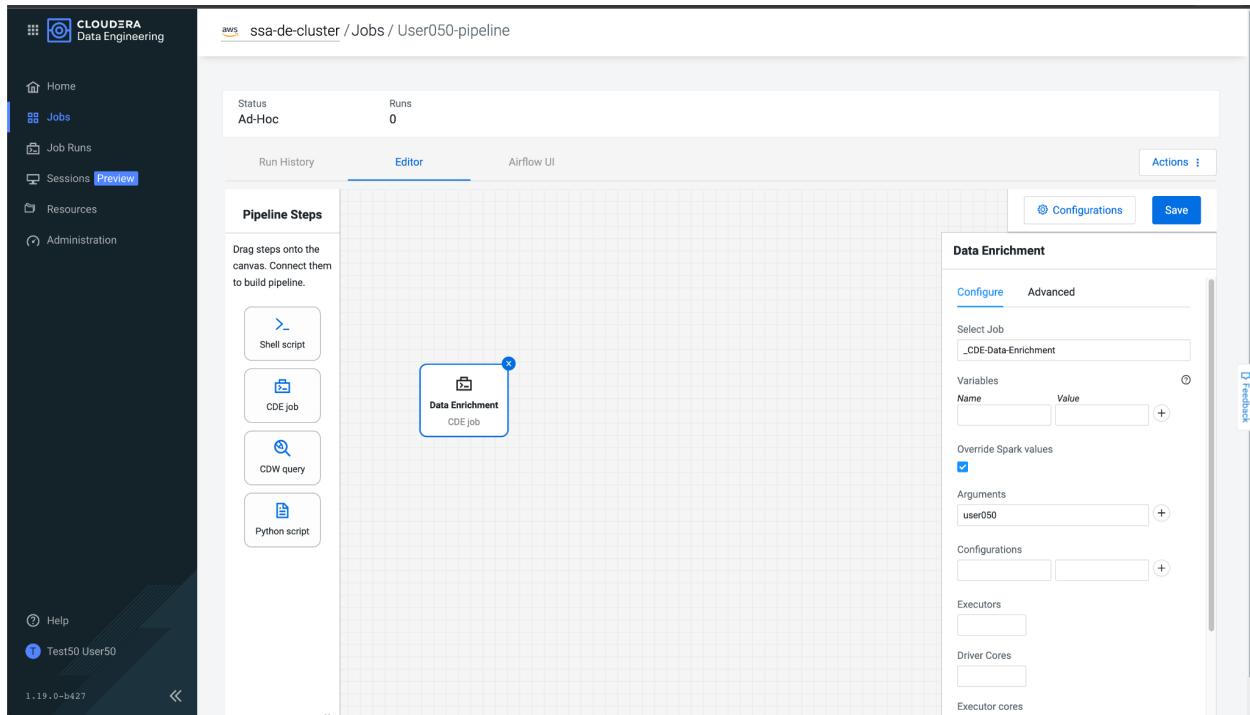


5. On the Job editing screen, select the Editor tab, and you will see the following canvas to drag the steps of the pipeline that we are going to create. In our case, we are going to create two CDE Jobs and relate them.



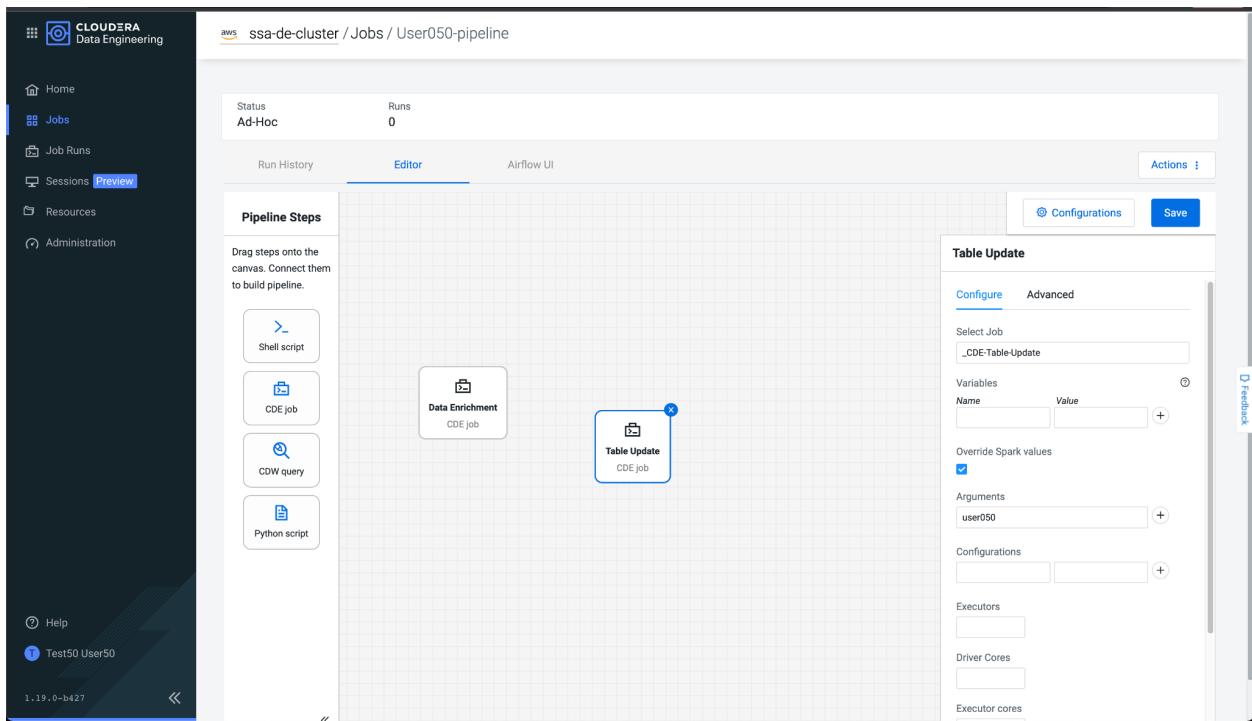
6. Let's start with the first Job. Click on the CDE Job button and drag onto the canvas, entering the following settings:

- **title/name:** Data Enrichment
- **Select Job:** select the Job_CDE-Data-Enrichment
- Check the checkbox **Override Spark values**. Additional options will appear below.
- **Arguments:** <assigned user>. Use the username assigned to you. For example, user050

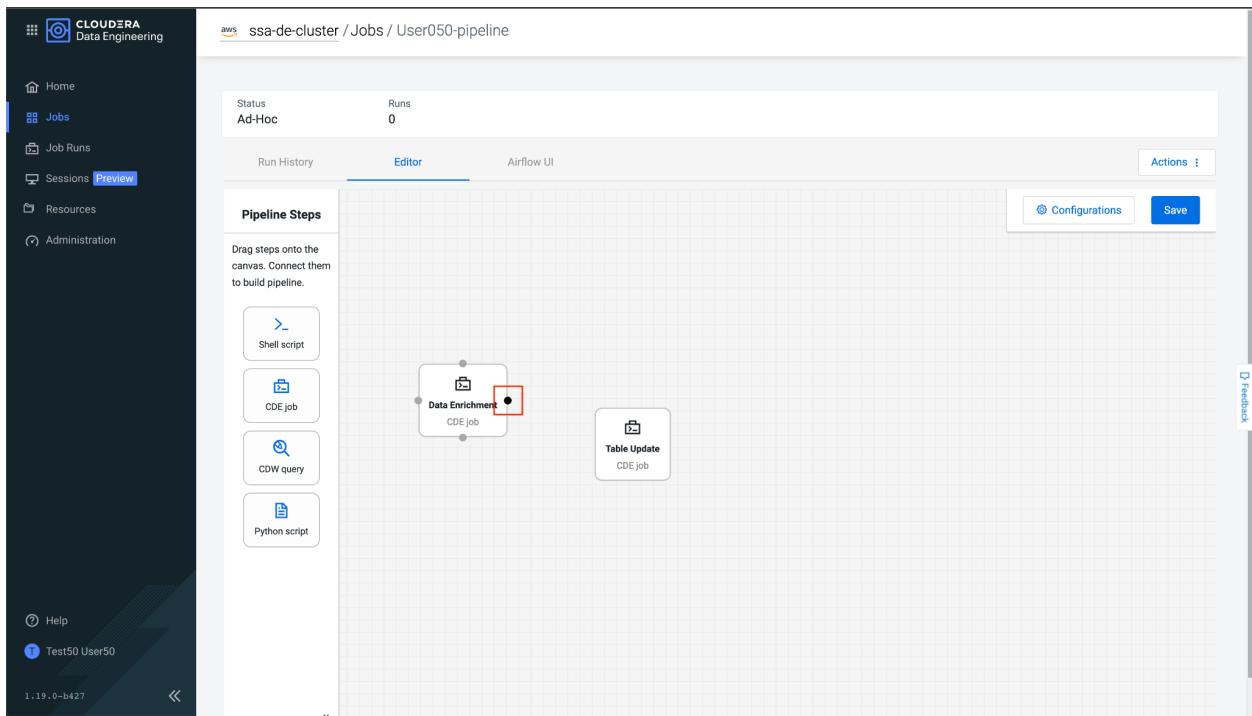


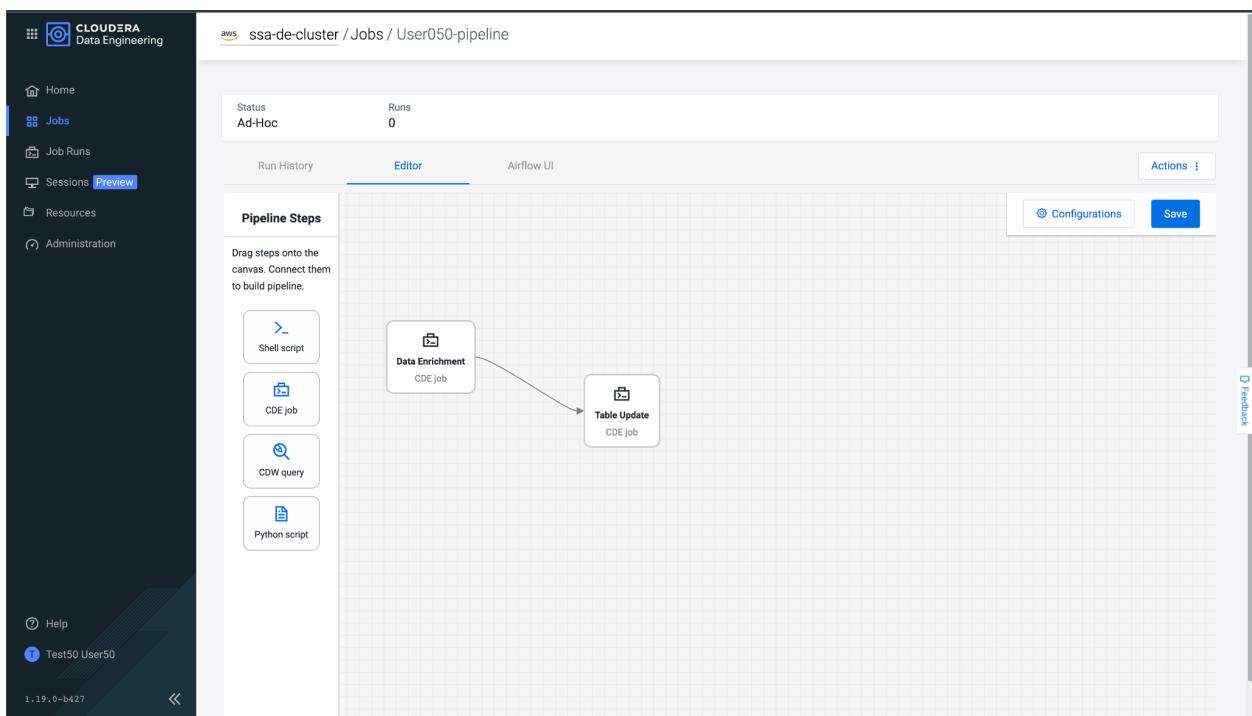
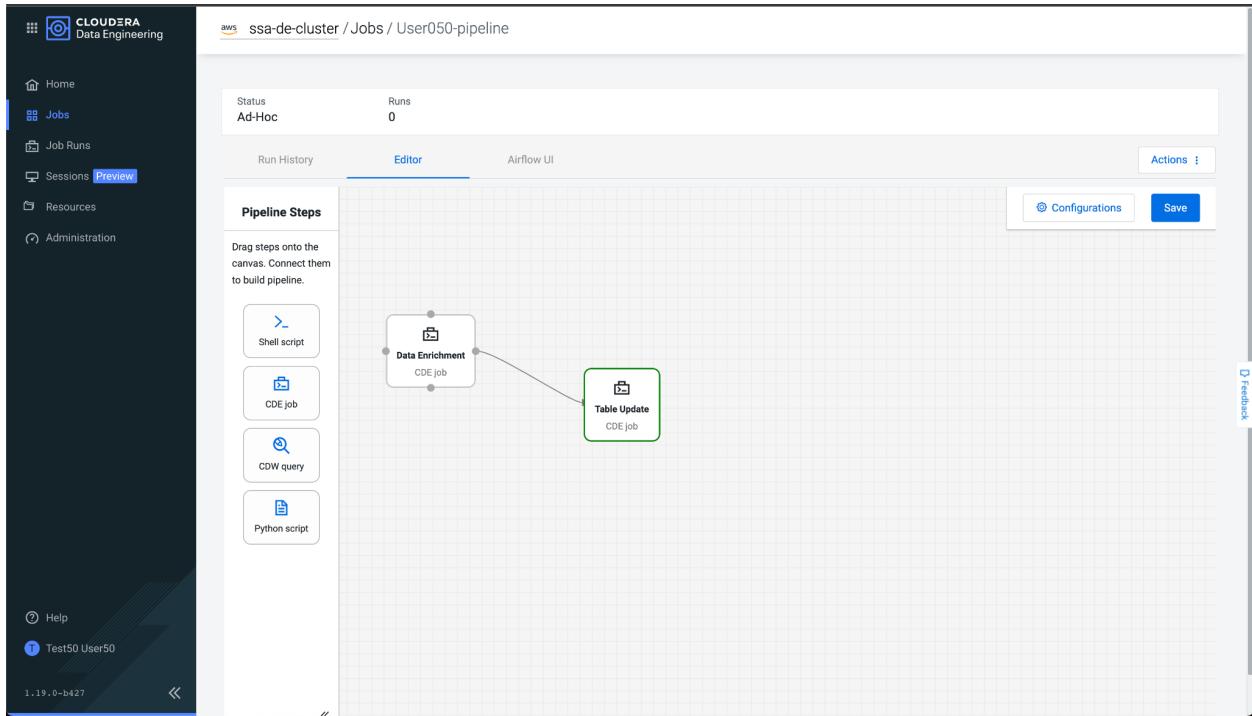
7. Configure the second Job. Click on the CDE Job button and drag onto the canvas, entering the following settings:

- **title/name:** Table Update
- **Select Job:** select the Job_CDE-Table-Update
- Check the checkbox **Override Spark values**. Additional options will appear below.
- **Arguments:** <assigned user>. Use the username assigned to you. For example, user050

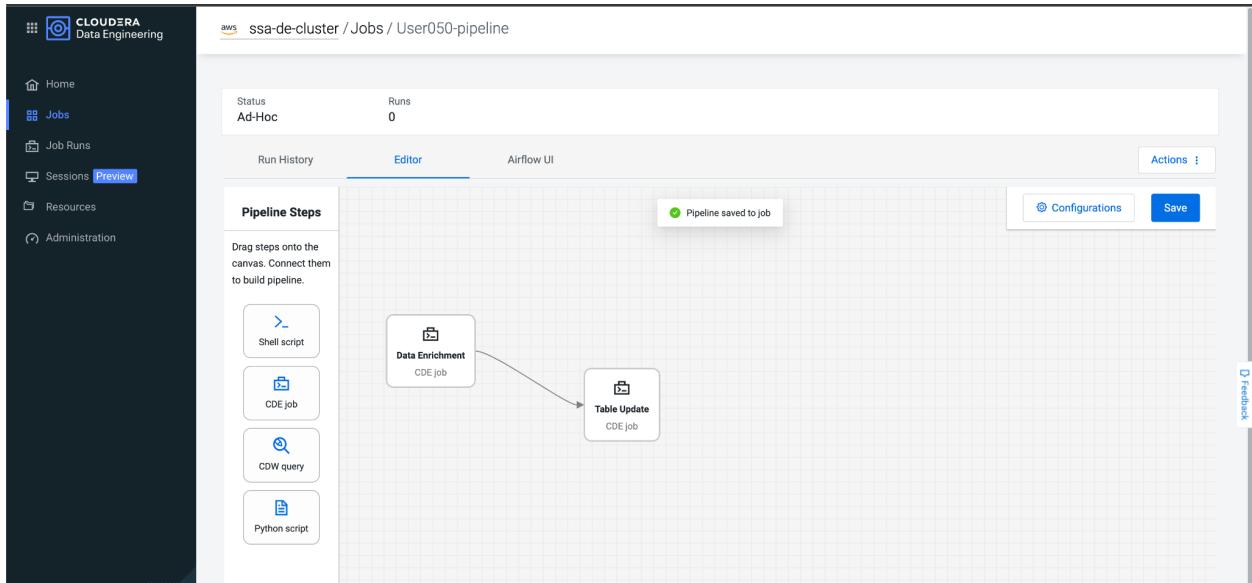


8. To set up the execution sequence, bind **Data Enrichment** with **Table Update**. For that, click on the right connector of the job of **Data Enrichment** and drag to the left connector of **Table Update**.

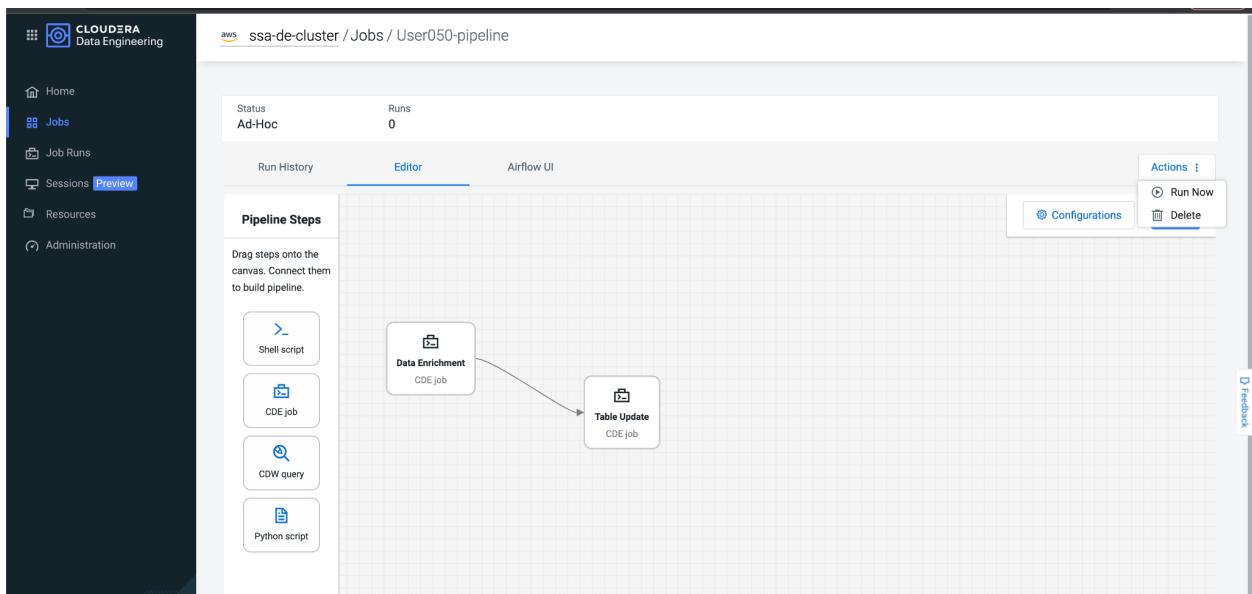




9. Once the Jobs have been joined, click on **Save** to save the settings made. You should see a message indicating **Pipeline saved to job**.



10. The time has come to run the pipeline. On the upper right side of the canvas, click **Actions** > **Run Now**.



11. You should see the pipeline execution screen, indicating that the execution has been initialized.

The screenshot shows the Cloudera Data Engineering interface. On the left, a sidebar menu includes Home, Jobs (selected), Job Runs, Sessions (Preview), Resources, Administration, Help, and Test50 User50. The main area displays a pipeline named "aws ssa-de-cluster / Jobs / User050-pipeline". A green notification bar at the top right says "A new run with id 7 has been initiated." Below it, a table shows "Status: Ad-Hoc" and "Runs: 0". There are tabs for Run History, Editor, and Airflow UI (selected). A search bar for "Search by Run Id" is present. A table lists one run: "Run ID: 7, Status: 7, Duration: 0, User: user050, Start Time: May 26, 2023, 1:32:09 PM". Pagination shows "Items per page: 10" and "1 - 1 of 1".

12. Click on the Airflow UI tab to see the execution detail of each step in the pipeline. The configured Data Enrichment and Table Update jobs are listed at the bottom left. The colors indicate the status of each job. Make sure the radio button **Auto-refresh** is enabled to automatically display the status of jobs.

The screenshot shows the Cloudera Data Engineering interface with the Airflow UI tab selected. The sidebar and pipeline details are the same as the previous screenshot. The main area now displays the Airflow UI interface for the "User050_pipeline". It shows a grid view of tasks. At the bottom left, two tasks are highlighted with a red box: "Data_Enrichment" and "Table_Update". The "Auto-refresh" button is turned on. On the right, the "DAG Details" section provides summary information: Total Runs Displayed (1), Total running (1), First Run Start (2023-05-26, 18:32:10 UTC), Last Run Start (2023-05-26, 18:32:10 UTC), and Max Run Duration (00:00:21).

13. You can see more information about the execution by clicking on the view **Graph**. Hovering the mouse over the Job name displays specific information for each step in the pipeline. Make sure the pipeline status is Success, which indicates that the entire pipeline was able to run without issue.

The screenshot shows the Cloudera Data Engineering UI. On the left is a sidebar with 'CLOUDERA Data Engineering' logo, navigation links (Home, Jobs, Job Runs, Sessions, Resources, Administration), and user info (Help, Test50 User50). The main area shows 'aws ssa-de-cluster / Jobs / User050-pipeline'. It has tabs for Status (Ad-Hoc, 1 Run), Run History, Editor, and Airflow UI (selected). Below is the DAG: User050_pipeline. The Graph tab is selected. A tooltip for the 'Data_Enrichment' task shows its status as 'success'. The DAG consists of two tasks: Data_Enrichment and Table_Update.

*The execution status appears next to the name of the pipeline (marked in red). If it is green and indicates **Success**, it means that the execution was successful.*

The screenshot shows the Cloudera Data Engineering UI. On the left is a sidebar with 'CLOUDERA Data Engineering' logo, navigation links (Home, Jobs, Job Runs, Sessions, Resources, Administration), and user info (Help, Test50 User50). The main area shows 'aws ssa-de-cluster / Jobs / User050-pipeline'. It has tabs for Status (Ad-Hoc, 1 Run), Run History, Editor, and Airflow UI (selected). Below is the DAG: User050_pipeline. The Graph tab is selected. A tooltip for the 'Table_Update' task shows its status as 'success'. The DAG consists of two tasks: Data_Enrichment and Table_Update.

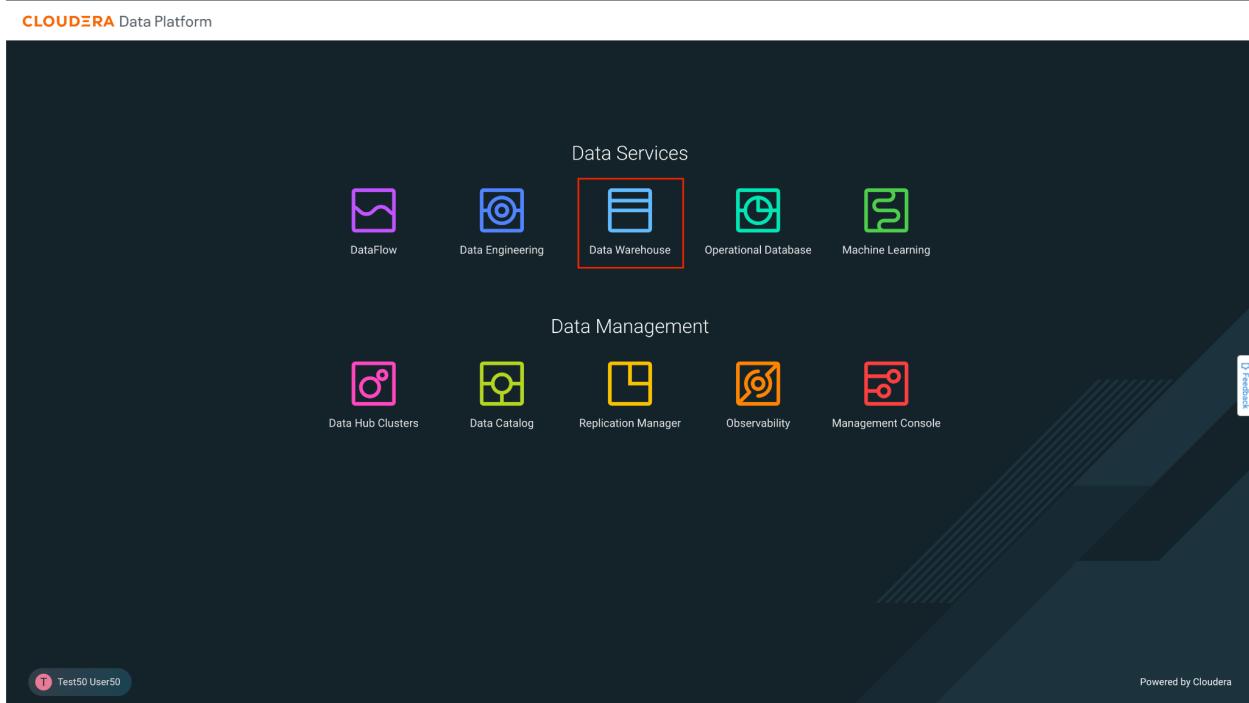
4. Data Warehouse

4.1. Goals

- Create a dataset pointing to the table
- Create a dashboard with metrics and dimensions

4.2. Dashboard Development

1. Click on Data Warehouse from CDP PC Home:



2. Data Warehouse welcome screen. Click on Data Visualization in the left menu.

3. In Data Visualization, click on the button **Data Viz** from which they were assigned.

4. Once in Data Visualization, go to the Data option from the top menu, and then to the Connector **ImpalaConn** from the left menu.

The screenshot shows the Cloudera Data Visualization interface. On the left, there's a sidebar with options like '% NEW CONNECTION', 'All Connections', and 'ImpalaConn' (which is highlighted with a red box). Below that is a section for 'samples'. The main area is titled 'Datasets' and shows a list of datasets. The columns in the table are 'Title/Table', 'ID', 'Created', 'Last Updated', 'Modified By', and '# Dashboards'. The datasets listed are: 'Food Stores Inspection in NYC' (ID: 12, Created: May 29, 2023), 'Cereals' (ID: 11, Created: May 29, 2023), 'World Life Expectancy' (ID: 9, Created: May 29, 2023), 'Earthquake Data January 2019' (ID: 10, Created: May 29, 2023), 'US State Populations Over Time' (ID: 7, Created: May 29, 2023), 'US County Population' (ID: 8, Created: May 29, 2023), 'Global Information Security Threats' (ID: 6, Created: May 29, 2023), and 'Restaurant Inspection SF' (ID: 5, Created: May 29, 2023). Each dataset entry includes a link to its details and a trash icon.

5. We have to create a new data source, for that, click on New Dataset and a window will appear to enter the information of the new data source.

The screenshot shows the 'New Dataset' dialog box open over the main Data Visualization interface. The dialog has tabs for 'Dataset' (selected), 'Add Data', and '...'. The main area of the dialog shows a table with columns 'Title/Table', 'ID', 'Created', 'Last Updated', 'Modified By', and '# Dashboards'. A single row is present with the message 'No data'.

6. Enter the information for the new data source:

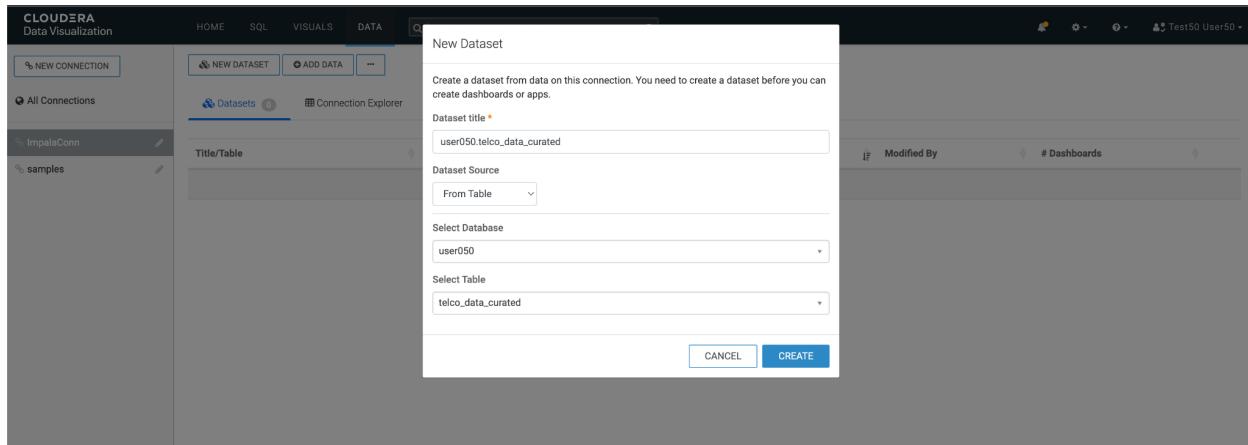
Dataset title: <assigned_user>.telco_curated_data

Dataset Source: From table

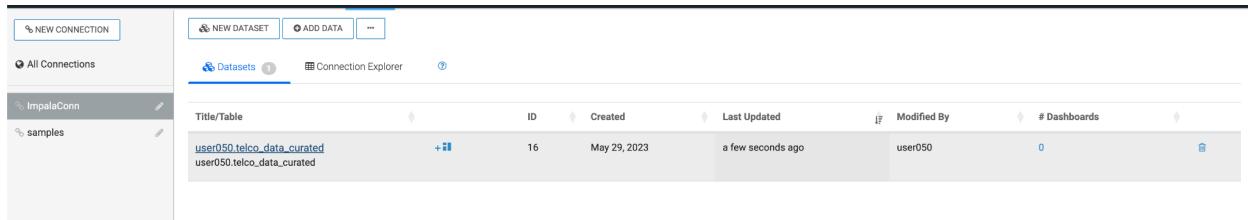
Select Database: <assigned_user>

Select Table: telco_data_curated

Click on Create to create the new Dataset.



7. The new Dataset should appear in the list. Click on the dataset that you just created.



8. Here you will see the details of the dataset.

The screenshot shows the 'Dataset Detail' page for a dataset named 'user050.telco_data_curated'. The left sidebar includes sections for Dataset Detail, Fields, Data Model, Time Modeling, Segments, Filter Associations, and Permissions. The main content area displays dataset metadata: Connection Type (Impala), Data Connection (ImpalaConn), Description (empty), Join Elimination (Enabled), Result Cache (From Connection), and Incremental Results (Disabled). It also shows creation details: ID 16, Created on May 29, 2023 06:15 PM, Created by user050, Last updated on May 29, 2023 06:15 PM, and Last updated by user050. Navigation buttons at the top right include 'CLONE DATASET' and 'NEW DASHBOARD'.

9. Click on **Fields** (left menu) to see the fields automatically captured during the dataset creation process.

The screenshot shows the 'Fields' section of the dataset 'user050.telco_data_curated'. The left sidebar includes sections for Dataset Detail, Fields, Data Model, Time Modeling, Segments, Filter Associations, and Permissions. The main content area is divided into 'Dimensions' and 'Measures'. The Dimensions section lists fields: multiplelines, paperlessbilling, gender, onlinesecurity, internetservice, techsupport, contract, churn, seniorcitizen, deviceprotection, streamingtv, streamingmovies, partner, customerid, dependents, onlinebackup, phoneservice, and paymentmethod. The Measures section lists fields: totalcharges, monthlycharges, and tenure. Navigation buttons at the top right include 'EDIT FIELDS' and 'NEW DASHBOARD'.

10. You can also preview the data from this screen. Click on **Data Model** (left menu) and then on the button **Show Data** that appears in the center.

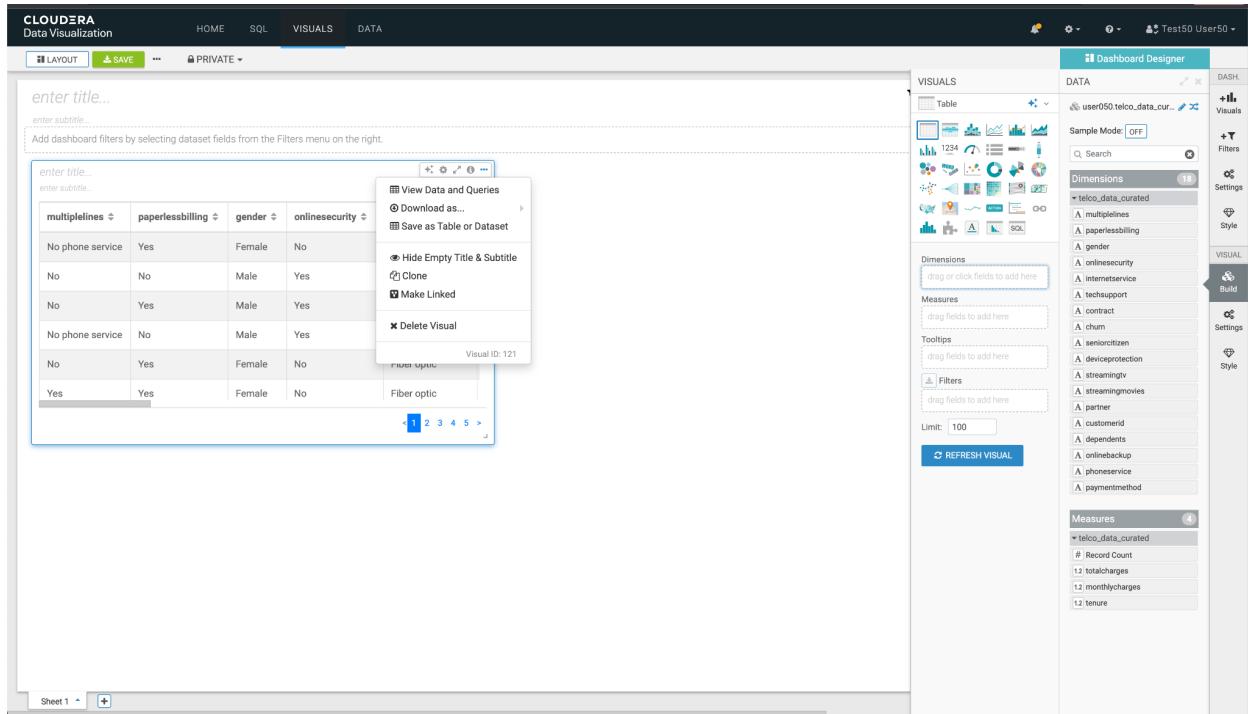
The screenshot shows the Cloudera Data Visualization interface. On the left, there's a sidebar with various options: Dataset Detail, Related Dashboards, Fields, Data Model (which is selected and highlighted in grey), Time Modeling, Segments, Filter Associations, and Permissions. In the main area, it says "Dataset: user050.telco_data_curated". Below that is a "Data Model" section with a "telco_data_curated" dataset. A prominent blue button labeled "SHOW DATA" is centered in this section. To its right is a "EDIT DATA MODEL" button. At the bottom of this section is a checkbox labeled "Apply Display Format". A red box highlights the "SHOW DATA" button.

11. At this moment, a query to the Virtual Warehouse is executed to retrieve the data from the data set. Notice the columns and values. Click New Dashboard to create a new dashboard.

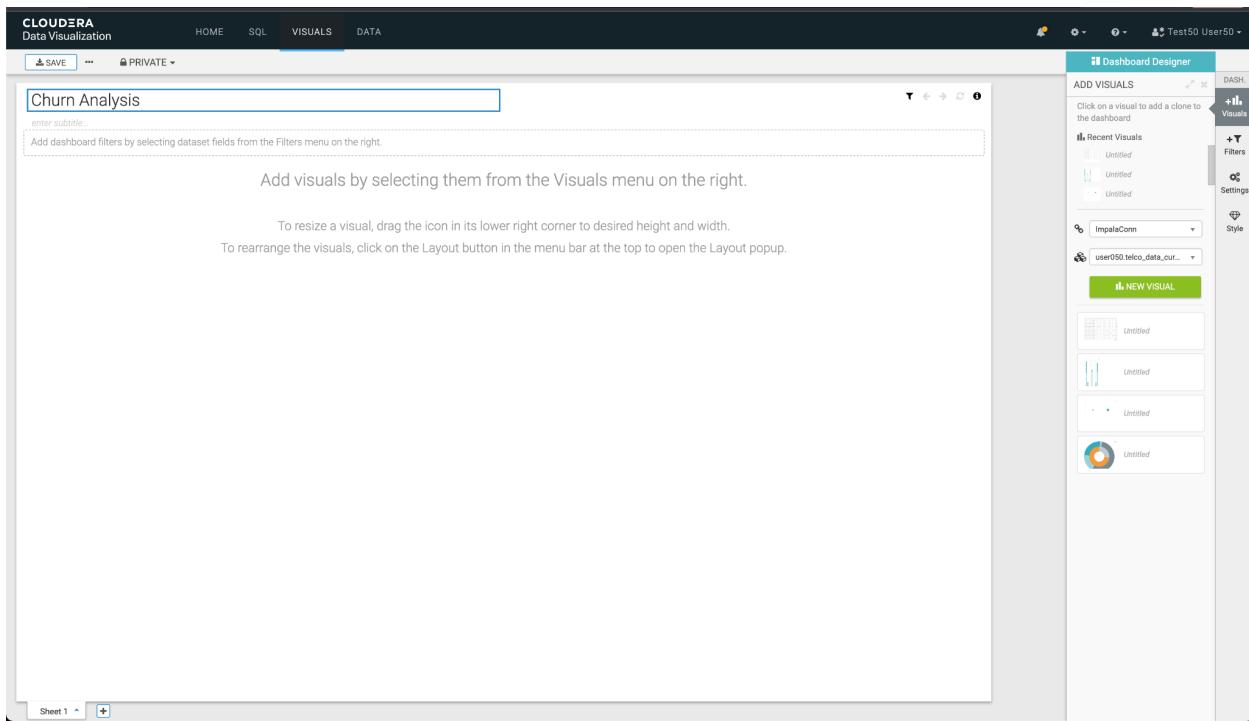
The screenshot shows the same Cloudera Data Visualization interface as the previous one, but now the "telco_data_curated" dataset is being previewed. The "SHOW DATA" button has been replaced by a "HIDE DATA" button. The "NEW DASHBOARD" button at the top right is highlighted with a red box. Below the preview area, a table titled "telco_data_curated" displays data for 10 rows. The columns listed are: multipletypes, paperlessbilling, gender, onlinesecurity, internetservice, techsupport, contract, churn, seniorcitizen, deviceprotection, streamingtv, streamingmovies, totalcharges, partner, monthlycharges, customerid, and d. The data rows show various combinations of these values for different customers.

multipletypes	paperlessbilling	gender	onlinesecurity	internetservice	techsupport	contract	churn	seniorcitizen	deviceprotection	streamingtv	streamingmovies	totalcharges	partner	monthlycharges	customerid	d
No phone service	Yes	Female	No	DSL	No	Month-to-month	No	0	No	No	No	29.850000381469727	Yes	32.602622985839844	7590-VHVEG	
No	No	Male	Yes	DSL	No	One year	No	0	Yes	No	No	1889.5	No	79.32872009277344	5575-GNVED	
No	Yes	Male	Yes	DSL	No	Month-to-month	Yes	0	No	No	No	108.1500015258789	No	53.849998474121094	3668-QPYBK	
No phone service	No	Male	Yes	DSL	Yes	One year	No	0	Yes	No	No	1840.75	No	39.008785247802734	7795-CFOCW	
No	Yes	Female	No	Fiber optic	No	Month-to-month	Yes	0	No	No	No	151.64999389648438	No	70.69999694824219	9237-HQITU	
Yes	Yes	Female	No	Fiber optic	No	Month-to-month	Yes	0	Yes	Yes	Yes	820.5	No	99.6500015258789	9305-COSKC	
Yes	Yes	Male	No	Fiber optic	No	Month-to-month	No	0	No	Yes	No	1949.4000244140625	No	154.11448669433594	1452-KIOVK	Yes
No phone service	No	Female	Yes	DSL	No	Month-to-month	No	0	No	No	No	301.8999938964844	No	46.75687789916992	6713-OKOMC	
Yes	Yes	Female	No	Fiber optic	Yes	Month-to-month	Yes	0	Yes	Yes	Yes	3046.050048828125	Yes	104.80000305175781	7892-POOKP	

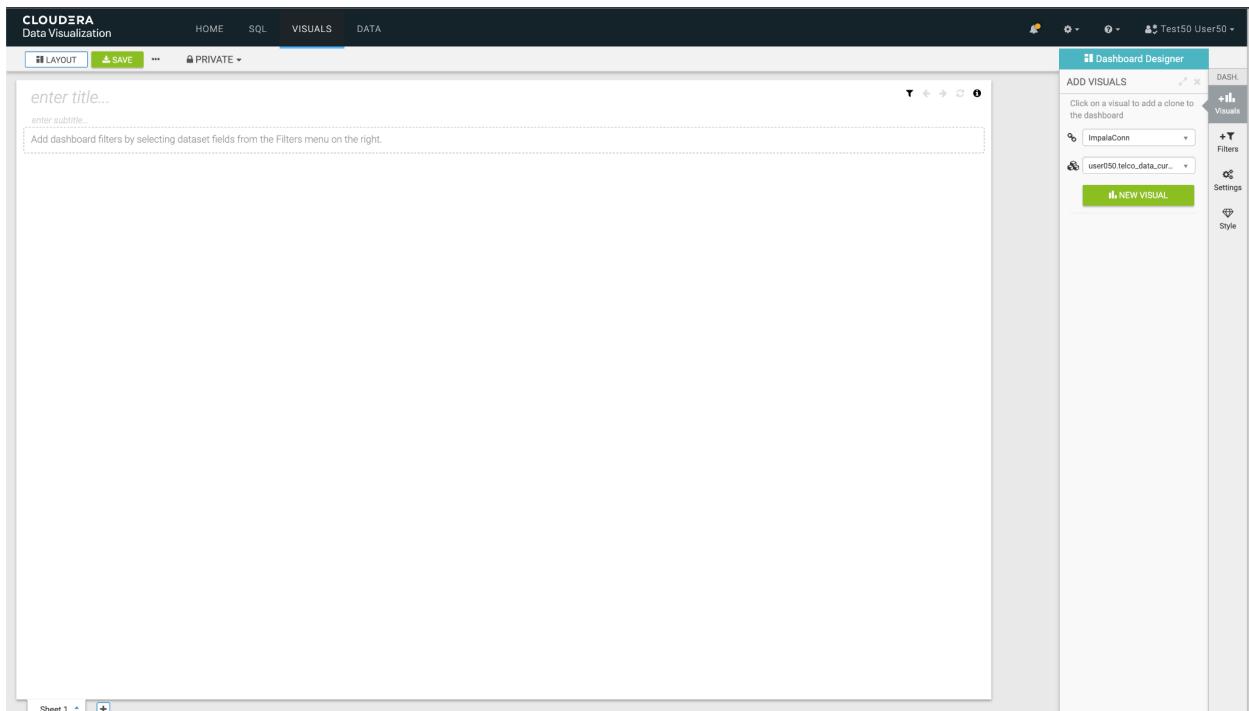
12. When opening the design canvas of a new panel, remove the element that is added by default, by clicking on the three dots (...) button at the top right of the element, and then clicking on the option **Delete Visual**



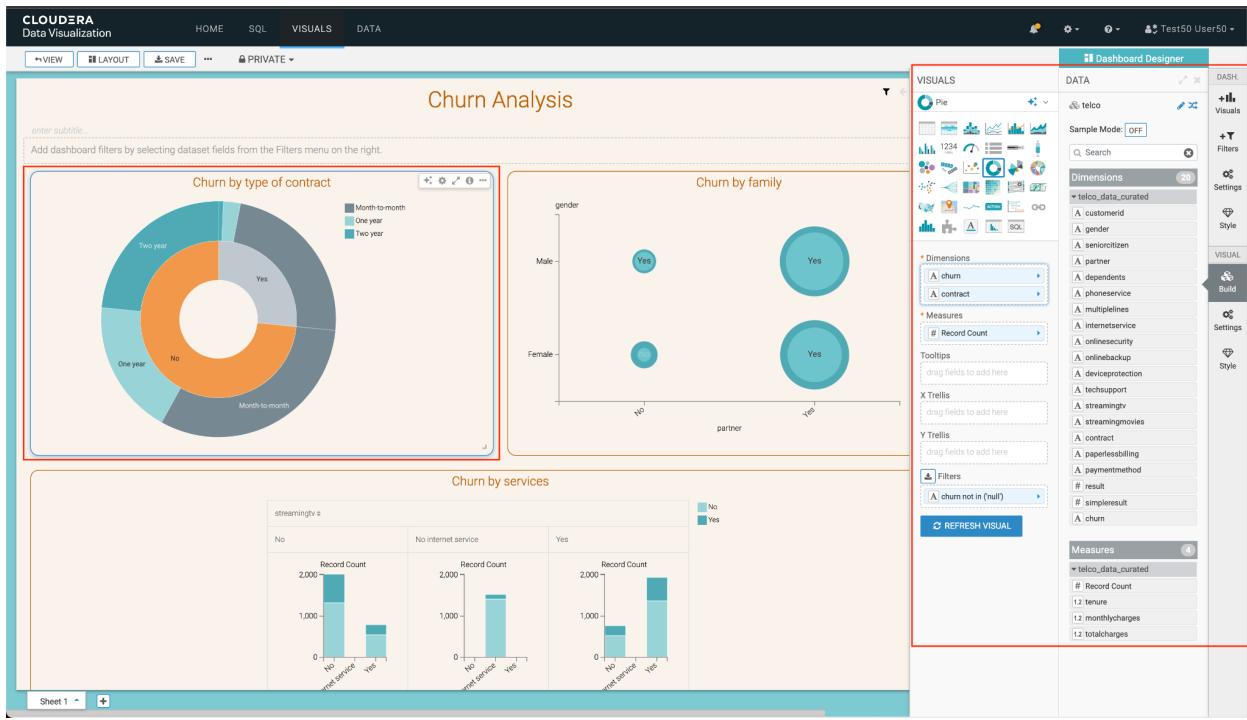
At the top of the canvas, in the enter title field, enter the name *Churn Analysis* to identify the dashboard.



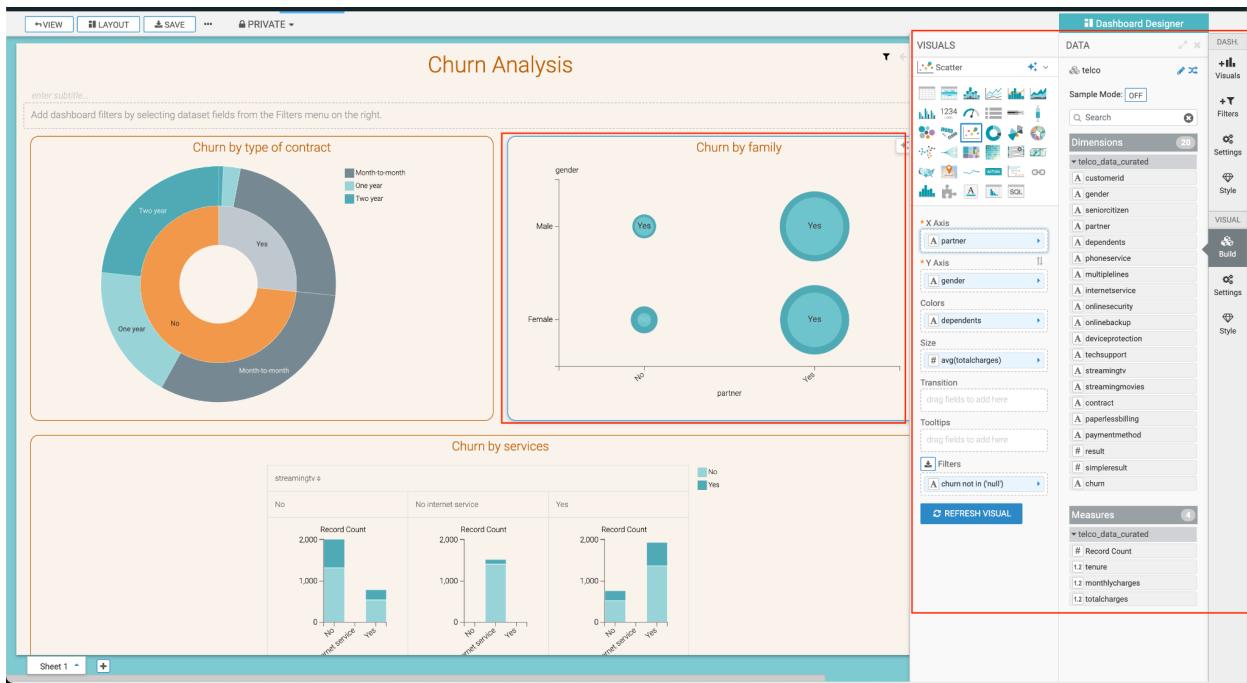
13. To add a new visual element, click on the button **Visuals** from the right menu, select the dataset that corresponds to them, and click on the button **New Visual**.



14. Add the first visual element, which is a pie chart with the dimensions **churn** and **contract**, with the metric of **Record count**. Once finished, click the button **Refresh Visual**.



15. Add the second visual element, which is a scatter chart with the dimension **partner** like X Axis, **gender** how Y Axis, **dependents** as Colors and **avg (total charges)** as Size. Once finished, click the button **Refresh Visual**.



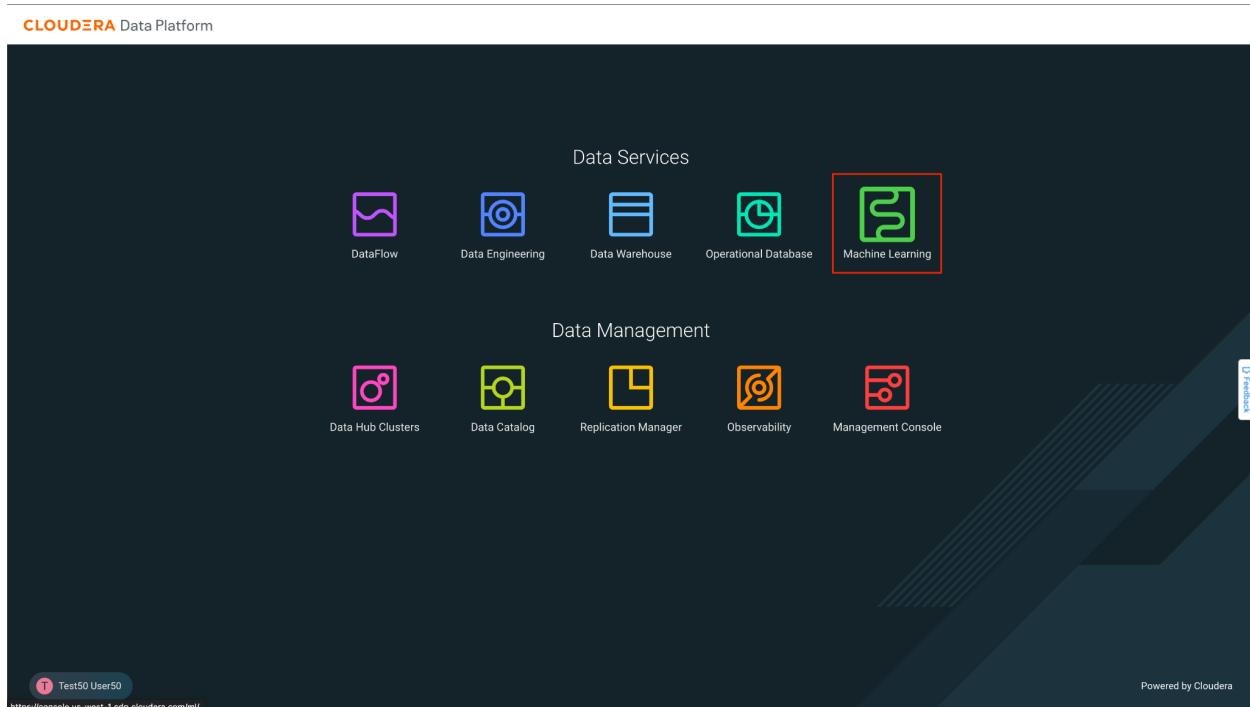
5. Machine Learning

5.1. Goals

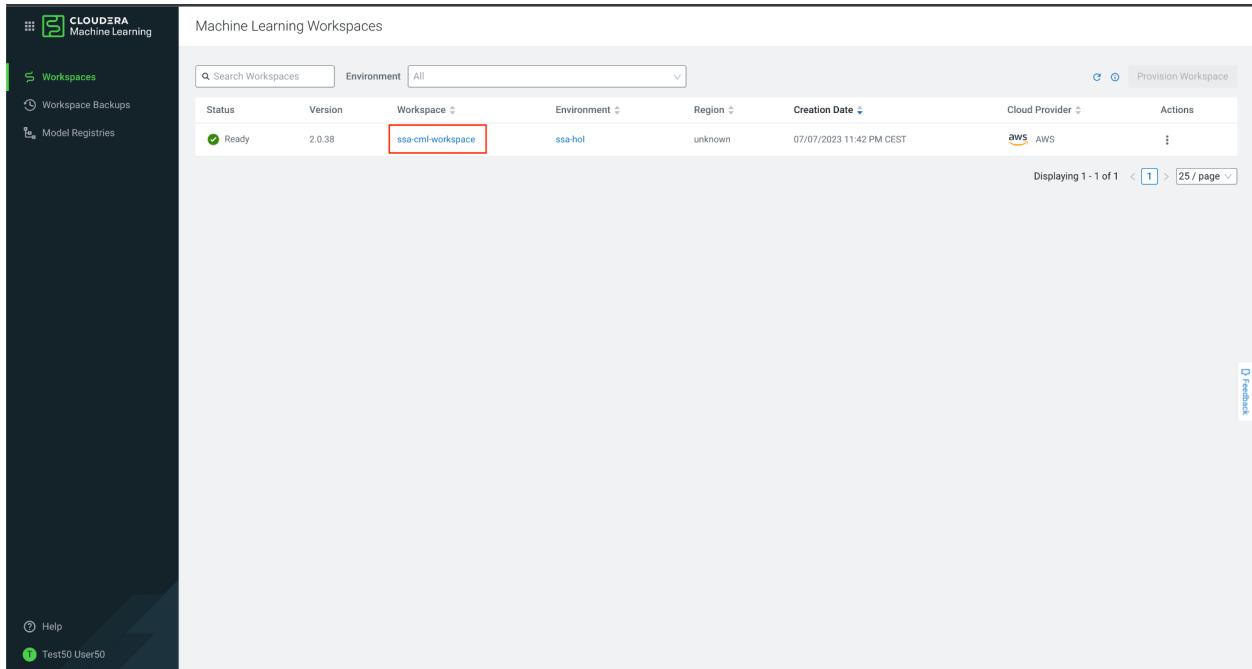
- Train a model to predict if a customer will churn
- Deploy/expose model as REST API

5.2. Create a Machine Learning Model for Churn Prediction

1. Click on Machine Learning from CDP PC Home:



2. This is a screen to select a Workspace, which is compute resource allocation for Data Science related jobs. Click on the only Workspace that appears.



The screenshot shows the 'Machine Learning Workspaces' page. On the left is a dark sidebar with the Cloudera Machine Learning logo and navigation links: 'Workspaces' (highlighted with a green bar), 'Workspace Backups', 'Model Registries', 'Help', and 'Test50 User50'. The main area has a light gray header with 'Machine Learning Workspaces' and search/filter options ('Search Workspaces', 'Environment: All'). Below is a table with the following data:

Status	Version	Workspace	Environment	Region	Creation Date	Cloud Provider	Actions
Ready	2.0.38	ssa-cml-workspace	ssa-hol	unknown	07/07/2023 11:42 PM CEST	aws AWS	⋮

At the bottom right of the table, it says 'Displaying 1 - 1 of 1 < 1 > 25 / page ▾'. There is also a 'Feedback' link on the right side of the page.

3. Once in the Workspace, you should see the following interface. Here are the projects you have created. It is time to create a new project. Click on the blue button **New Project**.

The screenshot shows the Cloudera Machine Learning interface. The left sidebar has a dark theme with white icons and text. The main area is titled 'Projects' and displays a message: 'You currently don't have any projects'. Below this, there's a brief description of what projects are and a 'New Project' button which is highlighted with a red box. At the bottom of the main area, it shows the workspace name 'ssa-cml-workspace' and the cloud provider 'AWS (AWS)'. The top right corner shows a user profile for 'user050'.

4. Enter the following information to create a new project:

Project Name: User0xx Telco Churn

Project Visibility: Private

Initial Setup, select Git

In the text field below HTTPS, enter the url of the git repo:

<https://github.com/camposalex/TelcoChurn>

New Project

Project Name

Project Description

Project Visibility Private - Only added collaborators can view the project Public - All authenticated users can view this project.

Initial Setup Blank Template AMPs Local Files Git

Provide the Git URL of the project to clone. Select the option that applies to your URL access.

HTTPS SSH

You are able to provide username/password.
e.g. https://username:password@mygithost.com/my/repository

Make sure to select **Python 3.7** in the Kernel selector. Click the button **Create Project**

Runtime setup

Basic Advanced

Basic configuration adds the most commonly used Editors for the Kernel of your choice. To fine-tune the Editors available in the project, choose the Advanced tab.

Kernel

Add GPU enabled Runtime variant

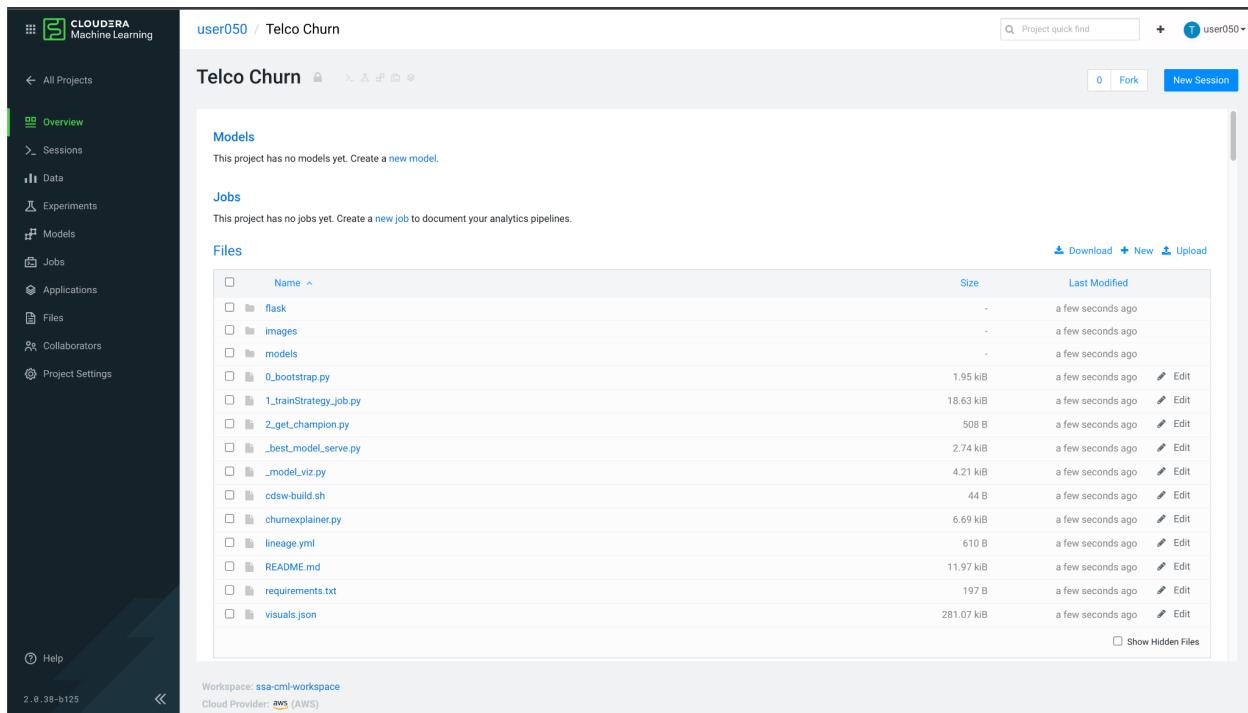
These runtimes will be added to the project:

- JupyterLab - Python 3.7 - Standard - 2023.05
- PBJ Workbench - Python 3.7 - Standard - 2023.05
- Workbench - Python 3.7 - Standard - 2023.05

5. Once the project is created, you should see the following screen:

Models, deploy and manage models as REST APIs to serve predictions.
Jobs, automate and orchestrate the execution of batch analytics workloads
Files, assets that are part of the project, such as files, scripts and code

This Telco Churn project consists of running three scripts. The way of execution is through a session, which is the allocation of isolated compute resources for each user. For this, you must click on the blue button **New Session**, located in the upper right.



The screenshot shows the Cloudera Machine Learning interface for the 'Telco Churn' project. The left sidebar includes links for All Projects, Overview, Sessions, Data, Experiments, Models, Jobs, Applications, Files, Collaborators, and Project Settings. The main content area displays the 'Telco Churn' project details. It features sections for 'Models' (indicating no models yet), 'Jobs' (indicating no jobs yet), and 'Files'. The 'Files' section lists the following contents:

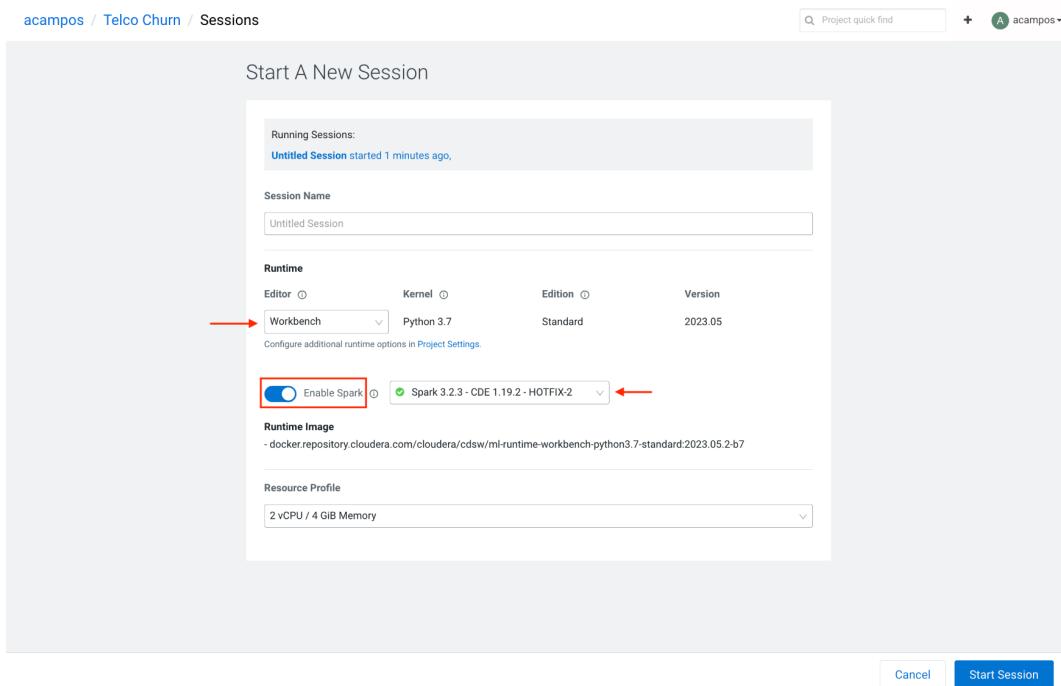
Name	Size	Last Modified
flask	-	a few seconds ago
images	-	a few seconds ago
models	-	a few seconds ago
0_bootstrap.py	1.95 kB	a few seconds ago
1_trainStrategy.job.py	18.63 kB	a few seconds ago
2_get_champion.py	508 B	a few seconds ago
_best_model_serve.py	2.74 kB	a few seconds ago
_model_viz.py	4.21 kB	a few seconds ago
cdsw-build.sh	44 B	a few seconds ago
chumexplainer.py	6.69 kB	a few seconds ago
lineage.yml	610 B	a few seconds ago
README.md	11.97 kB	a few seconds ago
requirements.txt	197 B	a few seconds ago
visuals.json	281.07 kB	a few seconds ago

At the bottom of the interface, it shows 'Workspace: ssa-cml-workspace' and 'Cloud Provider: AWS (AWS)'.

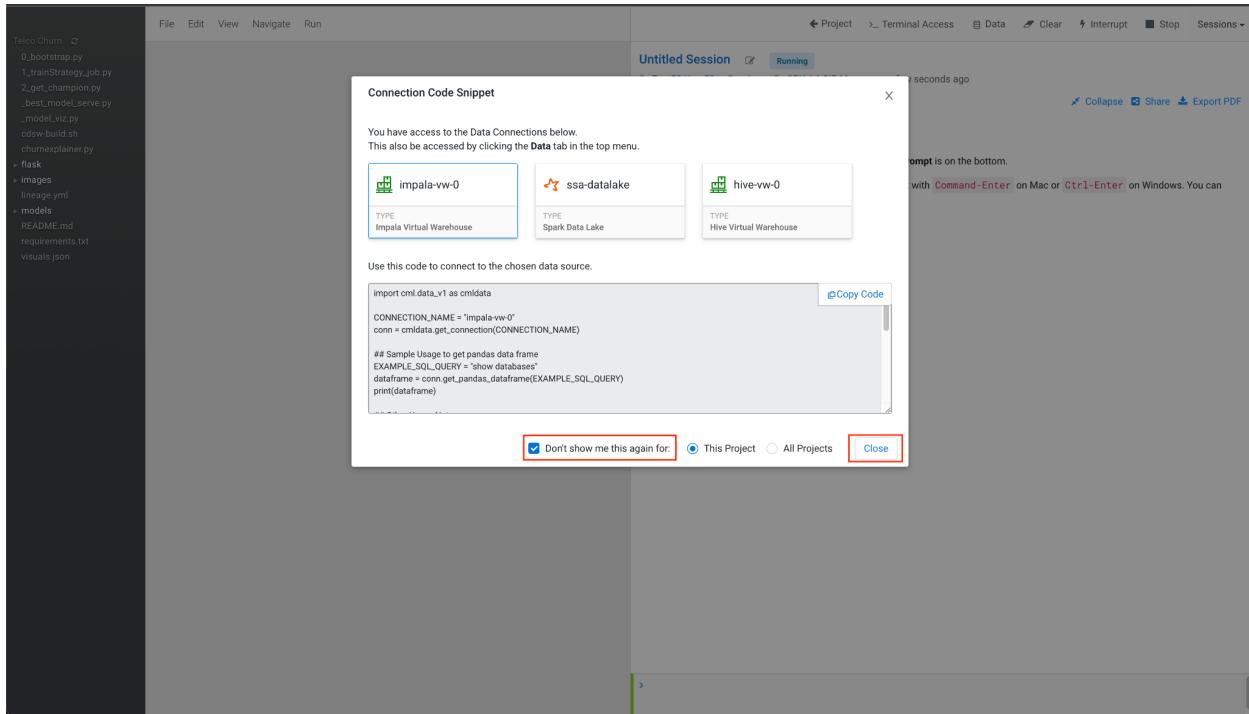
6. When starting a new session, make sure:

- Name the session: User0XX_Workbench_Session
- Select **Workbench** in the Editor selector.
- Enable **Spark**, marking the corresponding check.
- Select **Spark 3.2.x**, in the Spark version selector.
- Ressource Profile: 2vCPU / 4 GiB

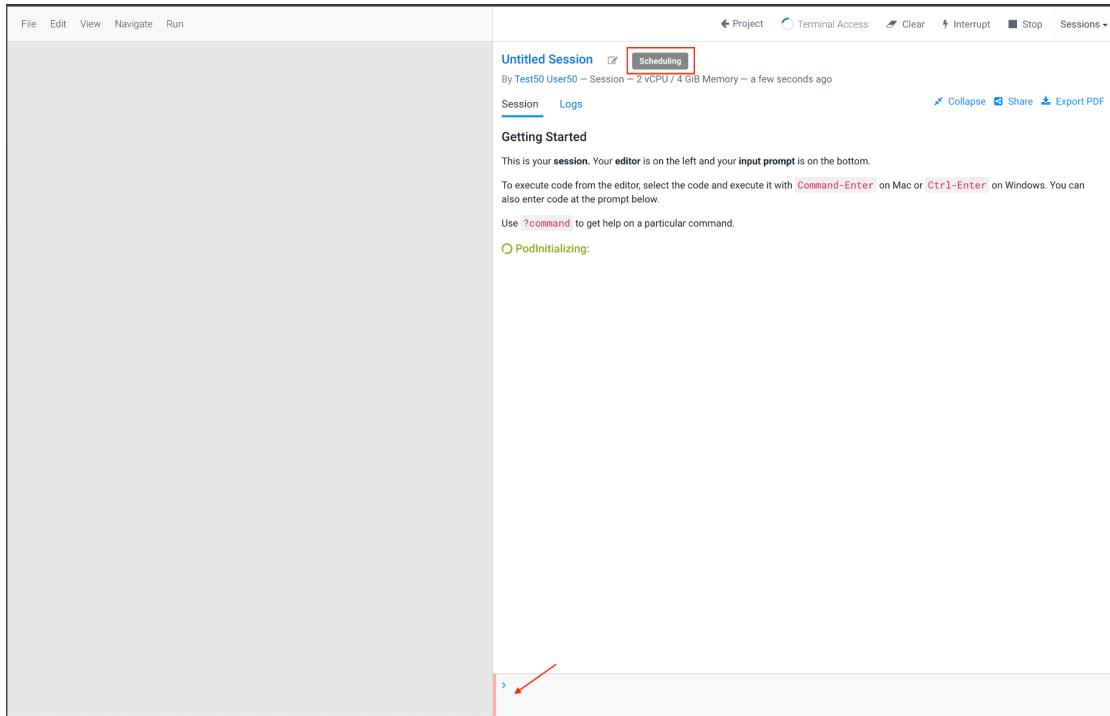
Click on the button **Start Session**



7. When you start a session for the first time, it will ask if you want to use a data connection. This project does not need this type of connection. mark the check of **Don't show me this again**, and then click the button **Close**, so this window will not appear anymore.



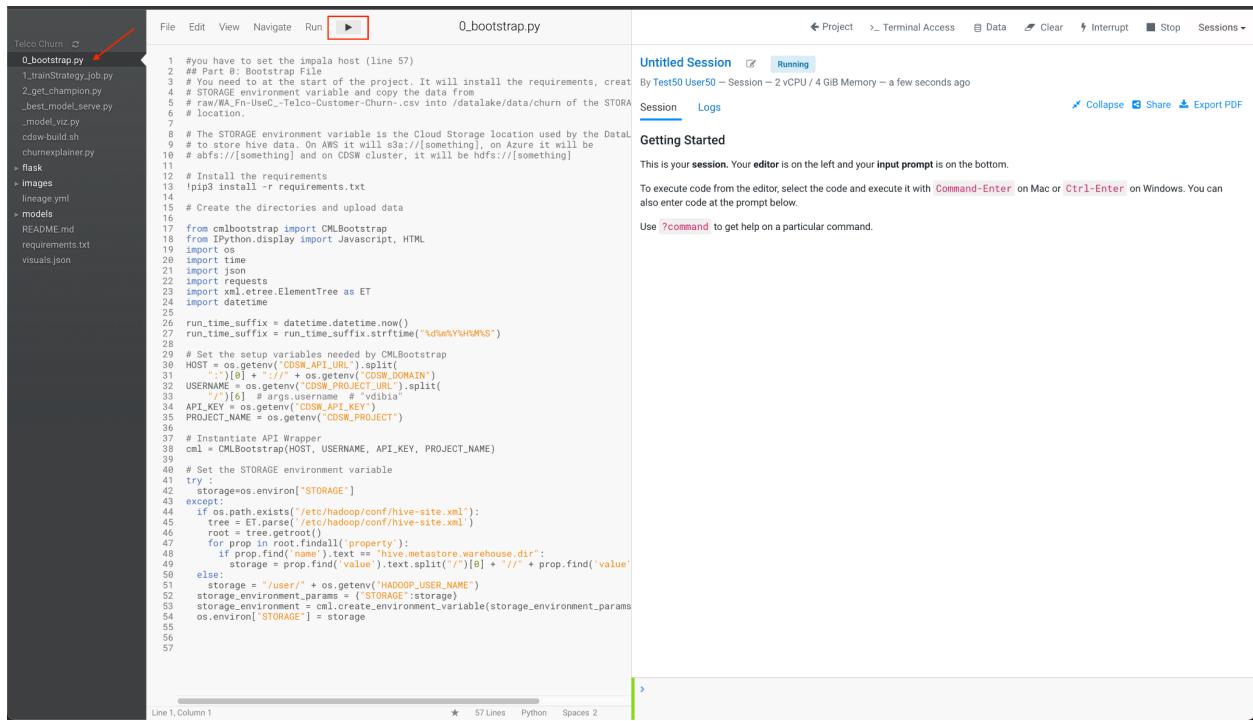
8. The editor/notebook located on the right side of the window will be in **Scheduling** status, and the bottom command bar flashing red. This means that CML is allocating computation for your session.



After a few seconds, the status changes to **Running**, and the command bar to green. This means that the session is ready to run code.

The screenshot shows a Jupyter Notebook interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Navigate', and 'Run' options. To the right of the bar are several icons: a left arrow labeled 'Project', a right arrow labeled 'Terminal Access', a square labeled 'Data', a brush icon labeled 'Clear', a lightning bolt icon labeled 'Interrupt', a stop sign icon labeled 'Stop', and a dropdown menu labeled 'Sessions'. Below the navigation bar, the main content area has a title 'Untitled Session' with a status bar indicating it is 'Running'. A red box highlights the 'Running' status. Below the title, it says 'By Test50 User50 — Session — 2 vCPU / 4 GiB Memory — a few seconds ago'. There are two tabs: 'Session' (underlined) and 'Logs'. To the right of the tabs are three buttons: 'Collapse', 'Share', and 'Export PDF'. A section titled 'Getting Started' follows, containing instructions about the session, editor, and input prompt. It also mentions using '?command' for help. At the bottom of the screen, there's a command input field with a green vertical cursor bar and a red arrow pointing to the prompt character (>).

9. The first script/code to run is **0_bootstrap.py**. This Python code configures the libraries required for the project and integration with Lakehouse tables you populated before. Select (just one click) the file in the bar located on the left side of the interface, this will make the code appear in the editor. Once the file is selected, click on the button  to run the code.



The screenshot shows the Jupyter Notebook interface. On the left, a sidebar lists files in the 'Telco Churn' directory, with '0_bootstrap.py' highlighted. The main pane displays the contents of '0_bootstrap.py'. A red arrow points to the play button icon at the top right of the code editor. The status bar at the bottom indicates 'Line 1, Column 1' and shows the code is running.

```

0_bootstrap.py
1 #you have to set the impala host (line 57)
2 ## Part 0: Bootstrap File
3 # You need to at the start of the project. It will install the requirements, creat
4 # the STORAE environment variable and copy the data from
5 # raw/TA-Fn-Used-Telco-Customer-Churn-.csv to /datalake/data/churn of the STORA
6 # location.
7
8 # The STORAE environment variable is the Cloud Storage location used by the Data
9 # to store hive data. On AWS it will s3a://[something], on Azure it will be
10 # abfs://[something] and on CDSW cluster, it will be hdfs://[something]
11
12 # Install the requirements
13 !pip3 install -r requirements.txt
14
15 # Create the directories and upload data
16
17 from cmlbootstrap import CMLBootstrap
18 from IPython.display import Javascript, HTML
19 import os
20 import time
21 import json
22 import requests
23 import xml.etree.ElementTree as ET
24 import datetime
25
26 run_time_suffix = datetime.datetime.now()
27 run_time_suffix = run_time_suffix.strftime("%d%b%Y%H%M%S")
28
29 # Set the setup variables needed by CMLBootstrap
30 HOST = os.getenv("CDSN_API_URL").split(
31     "/")[-1] + os.getenv("CDSN_DOMAIN")
32 USERNAME = os.getenv("CDSN_PROJECT_URL").split(
33     "/")[-1] # args.username # "vibial"
34 API_KEY = os.getenv("CDSN_API_KEY")
35 PROJECT_NAME = os.getenv("CDSN_PROJECT")
36
37 # Instantiate API Wrapper
38 cml = CMLBootstrap(HOST, USERNAME, API_KEY, PROJECT_NAME)
39
40 # Set the STORAGE environment variable
41 try :
42     storage=os.environ['STORAGE']
43 except:
44     if os.path.exists("/etc/hadoop/conf/hive-site.xml"):
45         tree = ET.parse('/etc/hadoop/conf/hive-site.xml')
46         root = tree.getroot()
47         for prop in root.findall('property'):
48             if prop.find('name').text == "hive.metastore.warehouse.dir":
49                 storage = prop.find('value').text.split('/')[-1] + "/" + prop.find('value').text
50             else:
51                 storage = "/user/" + os.getenv("HADOOP_USER_NAME")
52     storage_environment_params = {'STORAGE':storage}
53     storage_environment = cml.create.environment_variable(storage_environment_params)
54     os.environ['STORAGE'] = storage
55
56
57

```

When you start execution, you will see code output on the right side of the interface, and the bottom command bar flashing red, indicating that it is busy.

```

File Edit View Navigate Run ▶ 0_bootstrap.py
Telco Churn
0_bootstrap.py
1_trainStrategy_job.py
2_get_champion.py
3_best_model_serve.py
4_model_viz.py
cdsw-build.sh
churn_explainer.py
flask
images
lineage.yml
models
README.md
requirements.txt
visuals.json

File Edit View Navigate Run ▶ 0_bootstrap.py
Telco Churn
0_bootstrap.py
1 # you have to set the impala host (line 57)
2 ## Part 0: Bootstrap file
3 # You need to start at the start of the project. It will install the requirements, creat
4 # STORAGE environment variable and copy the data from
5 # raw/MA_Fn-UseC_-Telco-Customer-Churn-.csv into /datalake/data/churn of the STORA
6 # location
7
8 # The STORAGE environment variable is the Cloud Storage location used by the DataL
9 # to store hive data. On AWS it will s3a://[something], on Azure it will be
10 # abfs://[something] and on CDSW cluster, it will be hdfs://[something]
11
12 # Install the requirements
13 pip3 install -r requirements.txt
14
15 # Create the directories and upload data
16
17 from cmbootstrap import CMLBootstrap
18 from IPython.display import Javascript, HTML
19 import os
20 import time
21 import json
22 import requests
23 import xml.etree.ElementTree as ET
24 import datetime
25
26 run_time_suffix = datetime.datetime.now()
27 run_time_suffix = run_time_suffix.strftime("%d%b%Y%H%M%S")
28
29 # Set the setup variables needed by CMLBootstrap
30 HOST = os.getenv("CDSW_API_URL").split(
31     ":" )[0] + ":" + os.getenv("CDSW_DOMAIN")
32 USERNAME = os.getenv("CDSW_PROJECT_URL").split(
33     "/" )[6] # args.username # "vibbia"
34 API_KEY = os.getenv("CDSW_API_KEY")
35 PROJECT_NAME = os.getenv("CDSW_PROJECT")
36
37 # Instantiate API Wrapper
38 cml = CMLBootstrap(HOST, USERNAME, API_KEY, PROJECT_NAME)
39
40 # Set the STORAGE environment variable
41 try :
42     storage=os.environ['STORAGE']
43 except:
44     if os.path.exists('/etc/hadoop/conf/hive-site.xml'):
45         tree = ET.parse('/etc/hadoop/conf/hive-site.xml')
46         root = tree.getroot()
47         for prop in root.findall('property'):
48             if prop.find('name').text == "hive.metastore.warehouse.dir":
49                 storage = prop.find('value').text.split('/')[-1] + '/' + prop.find('value')
50             else:
51                 storage = "/user/" + os.getenv('HADOOP_USER_NAME')
52     storage_environment_params = ('STORAGE':storage)
53     storage_environment = cml.create_environment_variable(storage_environment_params
54     os.environ['STORAGE'] = storage
55
56
57

```

you have to set the impala host line57

Part 0: Bootstrap File

You need to start at the start of the project. It will install the requirements, creates the STORAGE environment variable and copy the data from raw/MA_Fn-UseC_-Telco-Customer-Churn-.csv into /datalake/data/churn of the STORAGE location.

The STORAGE environment variable is the Cloud Storage location used by the DataLake to store hive data. On AWS it will s3a://[something], on Azure it will be abfs://[something] and on CDSW cluster, it will be hdfs://[something].

Install the requirements

```
> !pip3 install -r requirements.txt
Collecting cmbootstrap (from -r requirements.txt (line 1))
  Cloning https://github.com/fastforwardlabs/cmbootstrap to /tmp/pip-install-qfytrqoz/cmbootstrap_d8f2aa8c04274970b6fc35b052aa5594
    Running command git clone --filter=blob:none --quiet https://github.com/fastforwardlabs/cmbootstrap /tmp/pip-install-qfytrqoz/cmbootstrap_d8f2aa8c04274970b6fc35b052aa5594
```

The green command bar indicates that the execution of the code has been finished. This bootstrap code takes 3-4 minutes to run.

```

File Edit View Navigate Run ▶ 0_bootstrap.py
Telco Churn
0_bootstrap.py
1 # you have to set the impala host (line 57)
2 ## Part 0: Bootstrap file
3 # You need to start at the start of the project. It will install the requirements, creat
4 # STORAGE environment variable and copy the data from
5 # raw/MA_Fn-UseC_-Telco-Customer-Churn-.csv into /datalake/data/churn of the STORA
6 # location
7
8 # The STORAGE environment variable is the Cloud Storage location used by the DataL
9 # to store hive data. On AWS it will s3a://[something], on Azure it will be
10 # abfs://[something] and on CDSW cluster, it will be hdfs://[something]
11
12 # Install the requirements
13 pip3 install -r requirements.txt
14
15 # Create the directories and upload data
16
17 from cmbootstrap import CMLBootstrap
18 from IPython.display import Javascript, HTML
19 import os
20 import time
21 import json
22 import requests
23 import xml.etree.ElementTree as ET
24 import datetime
25
26 run_time_suffix = datetime.datetime.now()
27 run_time_suffix = run_time_suffix.strftime("%d%b%Y%H%M%S")
28
29 # Set the setup variables needed by CMLBootstrap
30 HOST = os.getenv("CDSW_API_URL").split(
31     ":" )[0] + ":" + os.getenv("CDSW_DOMAIN")
32 USERNAME = os.getenv("CDSW_PROJECT_URL").split(
33     "/" )[6] # args.username # "vibbia"
34 API_KEY = os.getenv("CDSW_API_KEY")
35 PROJECT_NAME = os.getenv("CDSW_PROJECT")
36
37 # Instantiate API Wrapper
38 cml = CMLBootstrap(HOST, USERNAME, API_KEY, PROJECT_NAME)
39
40 # Set the STORAGE environment variable
41 try :
42     storage=os.environ['STORAGE']
43 except:
44     if os.path.exists('/etc/hadoop/conf/hive-site.xml'):
45         tree = ET.parse('/etc/hadoop/conf/hive-site.xml')
46         root = tree.getroot()
47         for prop in root.findall('property'):
48             if prop.find('name').text == "hive.metastore.warehouse.dir":
49                 storage = prop.find('value').text.split('/')[-1] + '/' + prop.find('value')
50             else:
51                 storage = "/user/" + os.getenv('HADOOP_USER_NAME')
52     storage_environment_params = ('STORAGE':storage)
53     storage_environment = cml.create_environment_variable(storage_environment_params
54     os.environ['STORAGE'] = storage
55
56
57

```

you have to set the impala host line57

Part 0: Bootstrap File

You need to start at the start of the project. It will install the requirements, creates the STORAGE environment variable and copy the data from raw/MA_Fn-UseC_-Telco-Customer-Churn-.csv into /datalake/data/churn of the STORAGE location.

The STORAGE environment variable is the Cloud Storage location used by the DataLake to store hive data. On AWS it will s3a://[something], on Azure it will be abfs://[something] and on CDSW cluster, it will be hdfs://[something].

Install the requirements

```
> !pip3 install -r requirements.txt
Collecting cmbootstrap (from -r requirements.txt (line 1))
  Cloning https://github.com/fastforwardlabs/cmbootstrap to /tmp/pip-install-qfytrqoz/cmbootstrap_d8f2aa8c04274970b6fc35b052aa5594
    Running command git clone --filter=blob:none --quiet https://github.com/fastforwardlabs/cmbootstrap /tmp/pip-install-qfytrqoz/cmbootstrap_d8f2aa8c04274970b6fc35b052aa5594
```

10. The second script/code to run is **1_trainStrategy_job.py**. This Python code will create the Experiment to run the model with three different hyper parameters and records the precision. Select (just one click) the file in the bar located on the left side of the interface, this will make the code appear in the editor. Once the file is selected, click on the button  to run the code. Once the execution is finished (approximately 1 minute), click on the button **Project**, located in the upper right bar of the session to go back to the project home.

11. Once back in project gome, click on the **Experiments** option, from the left menu, and then on **expRetrain** in the list of Experiments that appears.

The screenshot shows the Cloudera Machine Learning interface. The left sidebar has a dark theme with various project management and data science tools like Overview, Sessions, Data, Models, Jobs, Applications, Files, Collaborators, and Project Settings. The 'Experiments' icon is highlighted with a red arrow pointing to it. The main content area shows the 'Experiments' page for user010 / Telco Churn. It includes a search bar, a BETA status indicator, and a 'New Experiment' button. A table lists one experiment: 'expRetrain' created by 'Test10 User10' on '07/12/2023 8:36 PM'. At the bottom, there's a pagination message 'Displaying 1 - 1 of 1' and a '25 / page' dropdown.

Name	Creator	Created At	Last Updated
expRetrain	Test10 User10	07/12/2023 8:36 PM	-

Displaying 1 - 1 of 1 < 1 > 25 / page

12. On this screen you will see the three runs of this experiment. Look at the last column, where **precision** attribute displays. This is the precision that each hyper parameter is delivering.

user010 / Telco Churn / Experiments / expRetrain

Experiment BETA

Experiment Name: expRetrain
Experiment ID: pdj8-a6kp-bh2r-dehf
Artifact Location: /home/cdsw/experiments/pdj8-a6kp-bh2r-dehf

> Notes

Runs (3)

	Status	Start Time	Run Name	Duration	User	Source	Version	Models	algo	compute	dataset	precision
<input type="checkbox"/>	✓	2023-07-12 08:36:19	run_3619_0	5.2s	user010	<input type="checkbox"/> ipython3	8e811a	<input checked="" type="checkbox"/> sklearn	random forest	local	telco-churn	1
<input type="checkbox"/>	✓	2023-07-12 08:36:25	run_3619_1	3.8s	user010	<input type="checkbox"/> ipython3	8e811a	<input checked="" type="checkbox"/> sklearn	random forest	local	telco-churn	1
<input type="checkbox"/>	✓	2023-07-12 08:36:28	run_3619_2	4.0s	user010	<input type="checkbox"/> ipython3	8e811a	<input checked="" type="checkbox"/> sklearn	random forest	local	telco-churn	1

13. Let's go back to the session to run the last code. Since sessions run in Kubernetes containers, it's very easy to get back to where we were. Click on the option **Sessions** from the left menu, and later in the only session that will appear in the list.

Go to

- Sessions on the left menu
- Click on session name: User0XX_Workbench_Session
- If you didn't name your session when you started it (step 6), it should be called *Untitled Session*.

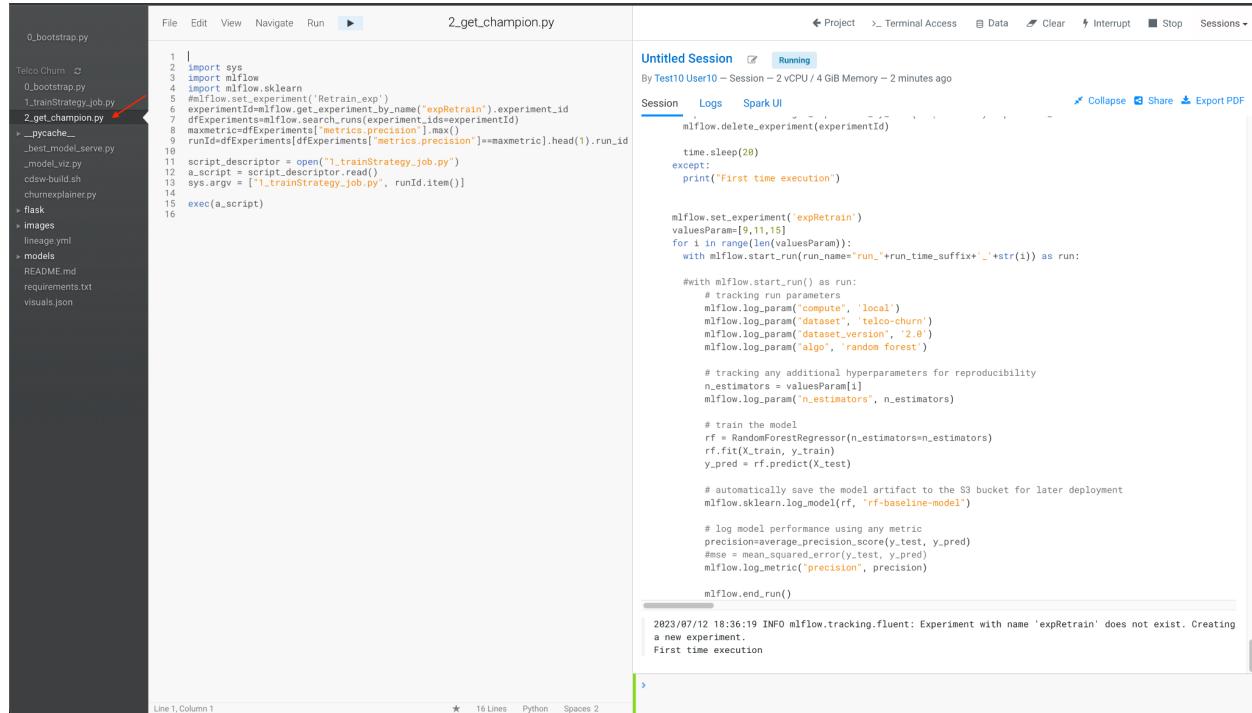
user010 / Telco Churn / Sessions

Sessions

Creator: All | Show Running Only:

Status	Session	Kernel	Creator	Created At	Duration	Actions
Running	Untitled Session	(Python 3.7 Workbench Standard)	Test10 User10	07/12/2023 8:35 PM	Running since 1m 43s	Edit Stop Delete

14. The third and last script/code to run is **2_get_champion.py**. This Python code takes the hyper parameter of the execution of the Experiment with better precision and deploys a Model as REST API, to be integrated in Data Visualization. Select (just one click) the file in the bar located on the left side of the interface, this will make the code appear in the editor. Once the file is selected, click on the button  to run the code.



```

File Edit View Navigate Run ▶ 2_get_champion.py

1 | import sys
2 | import mlflow
3 | import mlflow.sklearn
4 | mlflow.set_experiment('Retrain_experiment')
5 | experimentId=mlflow.get_experiment_by_name('retrain').experiment_id
6 | experiment=mlflow.get_experiment(experimentId)
7 | maxmetric=dfExperiments['metrics_precision'].max()
8 | maxmetric=dfExperiments['metrics_precision']==maxmetric].head(1).run_id
9 | runId=dfExperiments[dfExperiments['metrics_precision']==maxmetric].head(1).run_id
10 |
11 | script_descriptor = open("1_trainStrategy_job.py")
12 | a_script = script_descriptor.read()
13 | sys.argv = ["1_trainStrategy_job.py", runId.item()]
14 |
15 | exec(a_script)
16

```

Untitled Session  Running
By Test10 User10 — Session — 2 vCPU / 4 GiB Memory — 2 minutes ago

Session Logs Spark UI   

```

mlflow.delete_experiment(experimentId)

time.sleep(20)
except:
    print("First time execution")

mlflow.set_experiment('expRetrain')
valuesParam=[9,11,15]
for i in range(len(valuesParam)):
    with mlflow.start_run(run_name="run_" + run_time_suffix + '_' + str(i)) as run:
        with mlflow.start_run() as run:
            # tracking run parameters
            mlflow.log_param("compute", 'local')
            mlflow.log_param("dataset", 'telco-churn')
            mlflow.log_param("dataset_version", '2.0')
            mlflow.log_param("algo", 'random forest')

            # tracking any additional hyperparameters for reproducibility
            n_estimators = valuesParam[i]
            mlflow.log_param("n_estimators", n_estimators)

            # train the model
            rf = RandomForestRegressor(n_estimators=n_estimators)
            rf.fit(X_train, y_train)
            y_pred = rf.predict(X_test)

            # automatically save the model artifact to the S3 bucket for later deployment
            mlflow.sklearn.log_model(rf, "rf-baseline-model")

            # log model performance using any metric
            precision=average_precision_score(y_test, y_pred)
            mse = mean_squared_error(y_test, y_pred)
            mlflow.log_metric("precision", precision)

        mlflow.end_run()

2023/07/12 18:36:19 INFO mlflow.tracking.fluent: Experiment with name 'expRetrain' does not exist. Creating a new experiment.
First time execution

```

After a few seconds, you will see the following message “Deploying Model...” repeated several times, and the bottom command bar will be red.

```

File Edit View Navigate Run 2_get_champion.py
File Edit View Navigate Run 2_get_champion.py

1 import sys
2 import mlflow
3 import mlflow.sklearn
4 import pandas as pd
5 experimentId=mlflow.get_experiment_by_name("expRetrain").experiment_id
6 dfExperiments=mlflow.search_runs(experiment_ids=experimentId)
7 dfExperiments=dfExperiments[["metrics.precision"]].max()
8 runId=dfExperiments[dfExperiments[["metrics.precision"]]==maxmetric].head(1).run_id
9
10 script_descriptor = open("1_trainStrategy_job.py")
11 a_script = script_descriptor.read()
12 sys.argv = ["1_trainStrategy_job.py", runId.item()]
13
14 exec(a_script)
15
16

```

Untitled Session Running
By Test10 User10 - Session - 2 vCPU / 4 GiB Memory - 2 minutes ago

Session Logs Spark UI Collapse Share Export PDF

2023/11/10 09:13 AM MLflow tracking client. Experiment with name expRetrain does not exist. Creating a new experiment.

First time execution

```

>import sys
>import mlflow
>import mlflow.sklearn
mlflow.set_experiment('Retrain_exp')

>experimentId=mlflow.get_experiment_by_name("expRetrain").experiment_id
>dfExperiments=mlflow.search_runs(experiment_ids=experimentId)
>maxmetric=dfExperiments[["metrics.precision"]].max()
>runId=dfExperiments[dfExperiments[["metrics.precision"]]==maxmetric].head(1).run_id
>script_descriptor = open("1_trainStrategy_job.py")
>a_script = script_descriptor.read()
>sys.argv = ["1_trainStrategy_job.py", runId.item()]

/usr/local/bin/ipython3:1: FutureWarning: 'item' has been deprecated and will be removed in a future version
n
#/usr/local/bin/python3.7

>exec(a_script)

Starting Experiments
Creating Model
Creating new model
New model created with access key msqqkhgmf0lyt4ulz8ikvb7mbdph9gl
Deploying Model....
Model is deployed
Creating new model for visualization
New model created with access key mli7u0em8ypcxly6xid1c4a8g17q3fo1
Deploying Model....
Deploying Model.....

```

After about 2 minutes, the last message should be "Model is deployed", and the bar will be green. It means that the Deployment of the Model is complete.
 Click on the button **Project**, located in the upper right bar of the session to return to the home page of the project.

The screenshot shows a Jupyter Notebook environment. On the left, a sidebar displays the project structure for 'Telco Churn' with files like `0_bootstrap.py`, `1_trainStrategy_job.py`, `2_get_champion.py`, and `model_viz.py`. The main area shows a code cell for `2_get_champion.py` and its execution output in an 'Untitled Session' notebook.

```

File Edit View Navigate Run ▶ Project Untitled Session [Running]
By Test10 User10 - Session - 2 vCPU / 4 GiB Memory - 2 minutes ago
Session Logs Spark UI
Collapse Share Export PDF
> import sys
> import miflow
> import miflow.sklearn
> experimentId=miflow.get_experiment('Retrain_exp')
> dfExperiments=miflow.search_runs(experiment_ids=experimentId)
> dfExperiments=dfExperiments[dfExperiments['metrics_precision'].max()]
> runId=dfExperiments[dfExperiments['metrics_precision']==maxmetric].head(1).run_id
> script_descriptor = open("1_trainStrategy_job.py")
> a_script = script_descriptor.read()
> sys.argv = ["1_trainStrategy_job.py", runId.item()]
> exec(a_script)
Starting Experiments
Creating Model
Creating new model
New model created with access key msqqkhgmf0lyt4ulz81kvb7mbdph9gi
Deploying Model....
Model is deployed
Creating new model for visualization
New model created with access key ml17u8em8pcxly6xid1c4a8g17q3foi
Deploying Model....
Deploying Model....
Deploying Model....
Model is deployed

```

A red arrow points to the bottom right corner of the terminal output area, indicating where to click to see more content.

15. Once on the home page of the project, you will see the Model displayed. Click on the one that starts with **ModelViz_user0XX**.

The screenshot shows the CDP interface with the 'PROJECT' sidebar open. The 'Overview' tab is selected under 'PROJECT'. In the center, the 'Models' section displays a table with one row:

Model	Source	Status	Replicas	CPU	Memory	Last Deployed	Actions
ModelViz_user050	1._model...	Deployed	1 / 1	1	2.00 GiB	Oct 11, 2023, 03:47 PM	<button>Stop</button>

Below the table, the 'Jobs' section indicates 'This project has no jobs yet. Create a new job to document your analytics pipelines.'

The 'Files' section shows a file tree with the following structure:

- Name
- __pycache__
- flask
- images
- models
- 0_bootstrap.py
- 1_trainStrategy_job.py
- 2_get_champion.py
- best_model.serve.py
- model_viz.py

At the bottom, it shows 'Workspace: mi-paris-steller' and 'Cloud Provider: AWS (AWS)'.

16. Here you will see Model information and settings in the Overview tab.

The screenshot shows the Cloudera Machine Learning interface for the ModelOpsChurn_user010 model. The left sidebar includes links for All Projects, Overview, Sessions, Data, Experiments, Models (selected), Jobs, Applications, Files, Collaborators, and Project Settings. The main content area has tabs for Overview, Deployments, Builds, Monitoring, Logs, and Settings. The Overview tab is selected.

Description: Explain a given model prediction

Sample Code:

```
curl -H "Content-Type: application/json" -X POST https://modelservice.ml-1e5bb8eb-444.ssa-hol.yuit-vbzg.cloudera.site/model1?accessKey="msqkghmf0ly4ulz8kv7mbdph9g1","request":{"streamingtv":"No","monthlycharges":70.35,"phoneservice":"No","paperlessbilling":"No","partner":"No","onlinebackup":"No","gender":"female","contract":"Month-to-month","totalcharges":1397.475,"streamingmovies":"No","deviceprotection":"No","paymentmethod":"Bank transfer (automatic)","tenure":29,"dependents":"No","onlinesecurity":"No","multiplelines":"No","internetservice":"DSL","seniorcitizen":"No","techsupport":"No"} or curl -H "Content-Type: application/json" -X POST https://modelservice.ml-1e5bb8eb-444.ssa-hol.yuit-vbzg.cloudera.site/model1?accessKey="msqkghmf0ly4ulz8kv7mbdph9g1 -d {"request":{"streamingtv":"No","monthlycharges":70.35,"phoneservice":"No","paperlessbilling":"No","partner":"No","onlinebackup":"No","gender":"female","contract":"Month-to-month","totalcharges":1397.475,"streamingmovies":"No","deviceprotection":"No","paymentmethod":"Bank transfer (automatic)","tenure":29,"dependents":"No","onlinesecurity":"No","multiplelines":"No","internetservice":"DSL","seniorcitizen":"No","techsupport":"No"}'}
```

Model Details:

Source	Code
Model Id	19
Model CRN	cm.cdp.mlus-west-1:508fd88f-8076-498a-acfb-6f8765cd35e8 workspace:1e48b728-bcff-4867-8a54-830399e99355/fa9eb299-1c63-45f2-b60d-9a705eabfa5
Deployment Id	14
Deployment CRN	cm.cdp.mlus-west-1:508fd88f-8076-498a-acfb-6f8765cd35e8 workspace:1e48b728-bcff-4867-8a54-830399e99355/32aa37d1-af8b-4225-a86d-4ca5cc0f03109
Build Id	14
Build CRN	cm.cdp.mlus-west-1:508fd88f-8076-498a-acfb-6f8765cd35e8 workspace:1e48b728-bcff-4867-8a54-830399e99355/197c0cd8-b00e-4354-b63a-746fafe75e8
Deployed By	user010
Comment	Initial revision.
Runtime Image	Python 3.7 (Standard)
File	_best_model.serve.py
Function	explain

Model Resources:

Replicas	1
Total CPU	1 vCPUs
Total Memory	2.00 GiB

Test Model:

Input:

```
{"onlinesecurity": "No", "multiplelines": "No", "internetservice": "DSL", "seniorcitizen": "No", "techsupport": "No"}
```

Workspace: ssa-cml-workspace
Cloud Provider: AWS (AWS)

To test it and make a request to the model, scroll down, and click on the button **Test**, which will take the value in JSON format that is in the field **Input** and will make the request call to the model. What you see in the field **Result** is the response from the model in JSON format. If you wish, you can change some of the parameters of the **Input** field (for example, change some values from *Not* to *Yes*), and call the model again, and observe the value of the attribute *probability* of the response to see if there were any changes.

The screenshot shows the Cloudera Machine Learning interface for a project named 'user010'. The left sidebar includes links for All Projects, Overview, Sessions, Data, Experiments, Models (which is selected), Jobs, Applications, Files, Collaborators, and Project Settings. The main area displays the 'Overview' of a 'Test Model' named 'ModelOpsChurn_user010'. The 'Input' field contains a JSON object representing a customer profile. The 'Test' button is highlighted with a red box. The 'Result' section shows a green 'success' status and a detailed 'Response' object containing deployment information and a prediction probability of 0.5555555555555556. The 'Replica ID' is listed as 'modelopschurn-user010-19-14-6c5d7947ff-52kzg'. On the right side, there are sections for 'Comment' (Initial revision.), 'Runtime Image' (Python 3.7 (Standard)), 'File' (_best_model.serve.py), 'Function' (explain), and 'Model Resources' (Replicas: 1, Total CPU: 1 vCPUs, Total Memory: 2.00 GB). The bottom of the screen shows workspace and cloud provider information: 'Workspace: ssa-cml-workspace' and 'Cloud Provider: AWS (AWS)'.

6. Optional Labs

6.1. Goals

- Deploy Applied Machine Learning Models
- Discover and query data using HUE
- Add a new field that makes calls to the ML model
- Add the new field to the dashboard

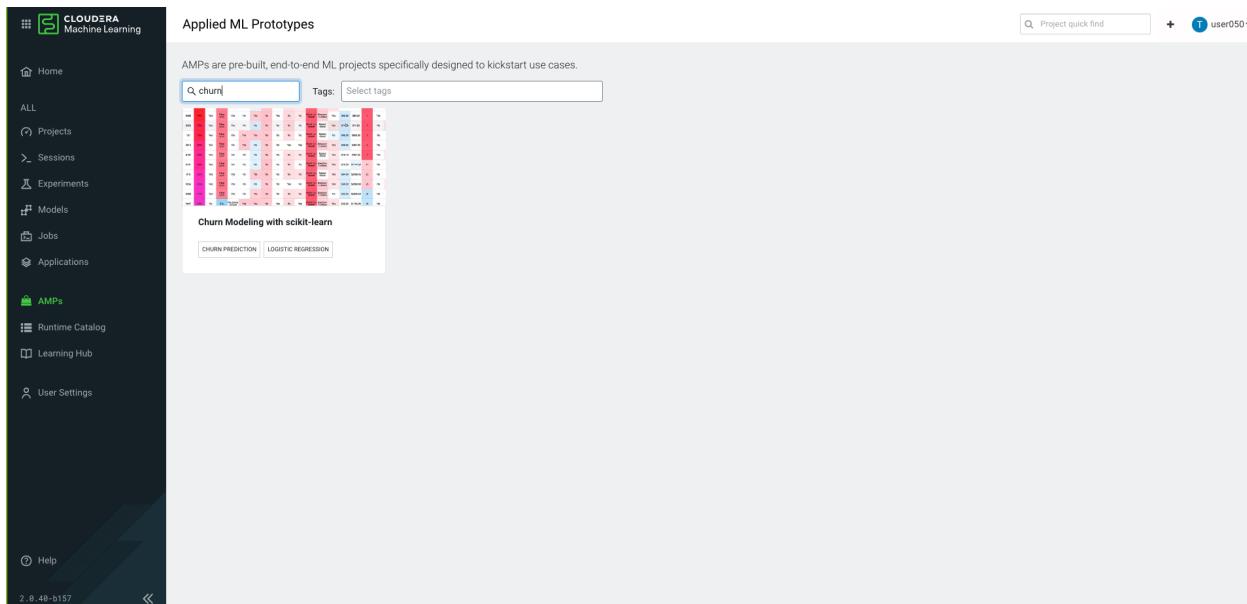
6.2. ML -Deploy Applied Machine Learning Model - Optional

1. In you Cloudera Machine learning home screen, click on AMPs on the left menu

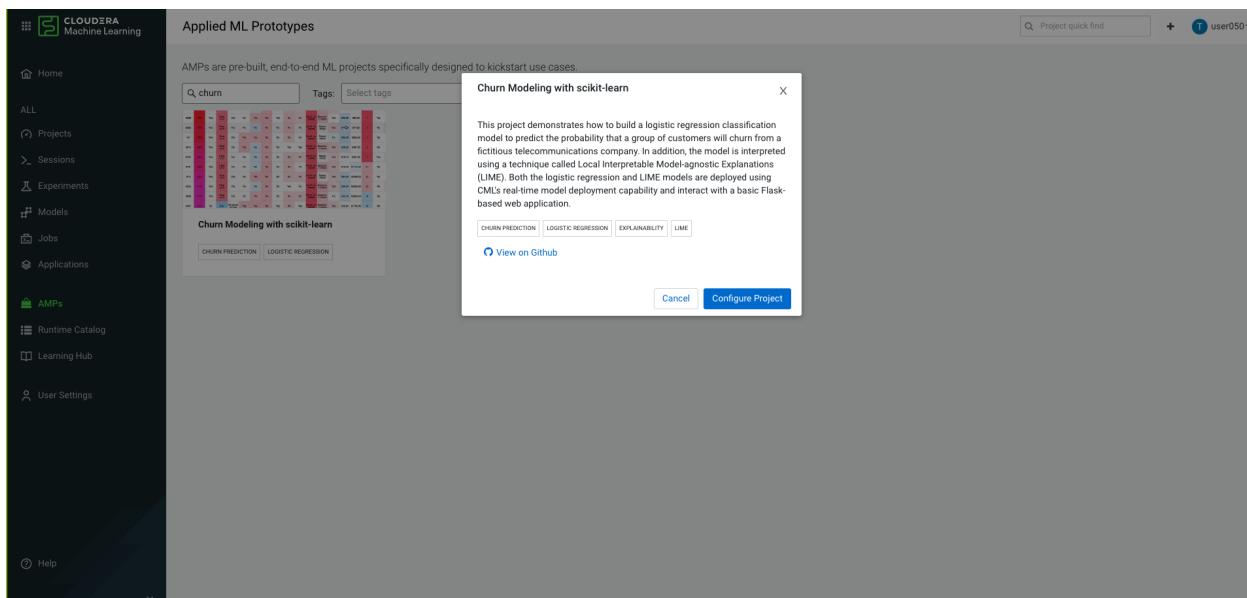
The screenshot shows the Cloudera Machine Learning interface. On the left, a sidebar menu includes options like Home, Projects, Sessions, Experiments, Models, Jobs, Applications, AMPs (which is selected), Runtime Catalog, Learning Hub, and User Settings. The main content area is titled "Applied ML Prototypes" and features a grid of tiles for different projects. Each tile contains a small image, the project name, and a brief description. For example, the "Text Summarization and more with Amazon Bedrock" project uses BEDROCK + LLM and includes a chart. The "Churn Modeling with scikit-learn" project uses CHURN PREDICTION + LOGISTIC REGRESSION and shows a line graph. Other visible projects include "Deep Learning for Image Analysis" (Computer Vision, IMAGE ANALYSIS), "Question Answering with WIKIPEDIA" (The Free Encyclopedia), and "Deep Learning for Question Answering" (AUTOMATED QUESTION ANSWERING, EXTRACTIVE QUESTK).

2. In the search bar on top, type: Churn

A tile will filter: "Churn Modeling with scikit-learn"



3. Click on the tile → Then click on Configure Project



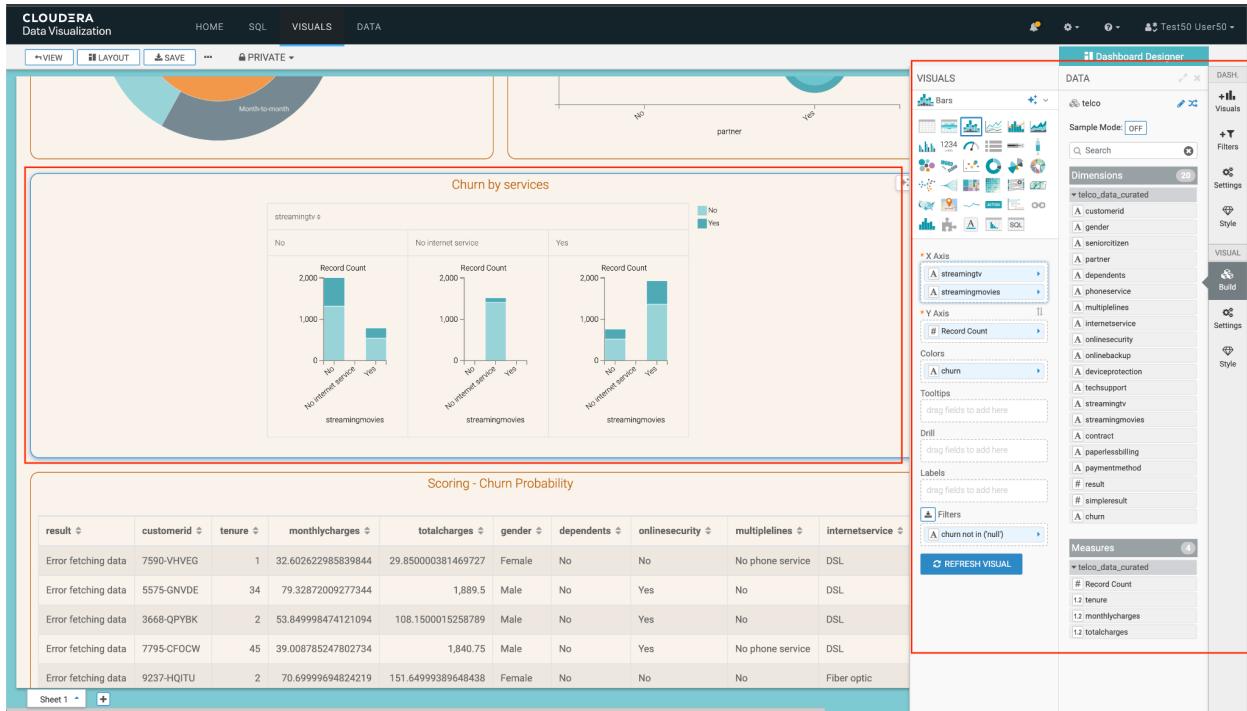
4. Project configuration screen launches. You can setup all the necessary steps for successful project configuration. Leave everything as-is.

Do Not Click Launch Project - this will be done by the facilitator.

5. Deployment launches

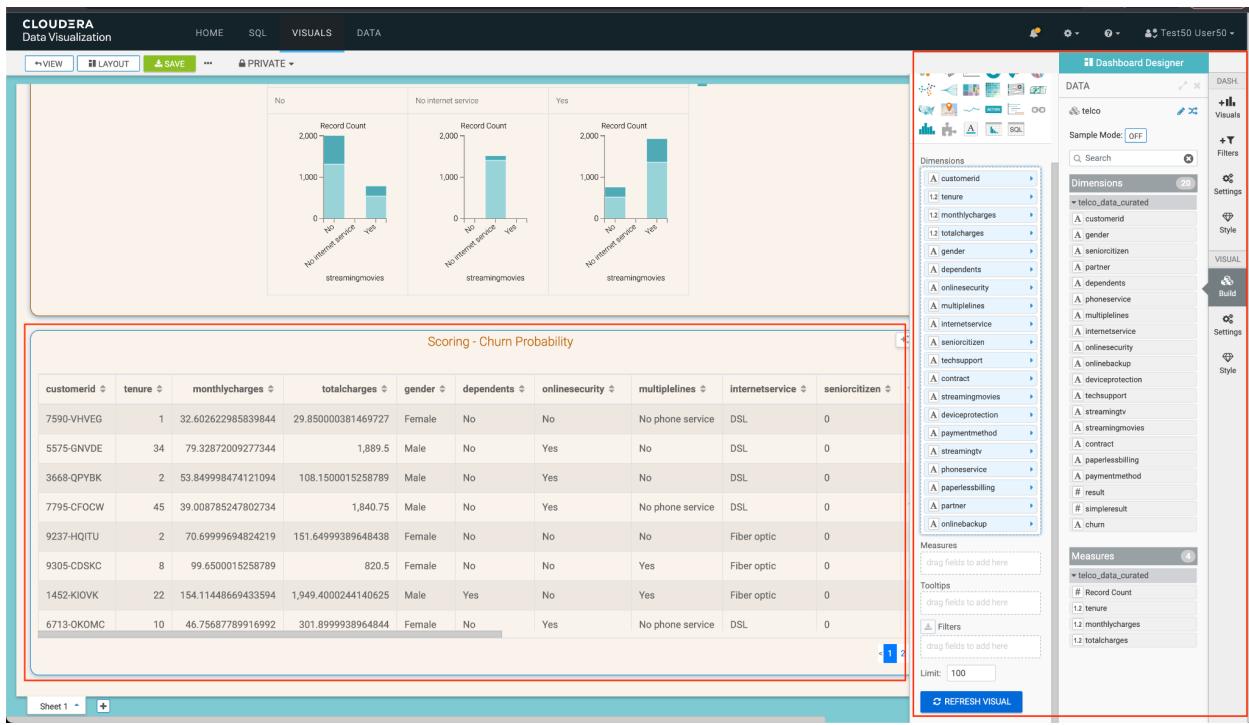
6.3. Add a third visual element - Optional

1. Add the third visual element, which is a bar chart with the dimensions **streamingtv** and **streamingmovies** like X Axis,
2. **Record Count** how Y Axis and **churn** how Colors. Once finished, click the button **Refresh Visual**.



6.4. One more visual element - Optional

1. If you do not want to call a machine learning model, please follow the below steps to add an extra element.
2. Add the fourth and last visual element, which is a table with the dimensions and metrics of the dataset. Be sure to add all 17 dimensions and 3 metrics to the table. Once finished, click the button **Refresh Visual**.



- Save the dashboard by clicking the button **Save** from the top menu.

6.5. Data Discovery and SQL Analysis Using HUE Dashboard - Optional

On your Cloudera landing page, go do Data Warehouse.

- Click on the **HUE** button, in your Data Warehouse dashboard.

Overview

The screenshot shows the Cloudera Data Platform Management Console interface. On the left, there's a sidebar with 'Environments' and a 'More...' button. The main area has three main sections: 'Create' (with 'See More' link), 'Query and visualize data' (with 'See More' link), and 'Guides and More' (with 'See More' link). Below these are two tabs: 'Database Catalogs' (2) and 'Virtual Warehouses' (2). The 'Database Catalogs' tab shows two entries: 'paris-atelier-dl-default' (Running, 7 cores, 17 GB memory, 2 VWHs) and 'tuya-matuu-dl-default' (Stopped, 7 cores, 17 GB memory, 0 VWHs). The 'Virtual Warehouses' tab shows three entries: 'paris-atelier-impala' (Running, 10 executors, 176 cores, 1396 GB memory, TYPE IMPALA), 'paris-atelier-hive' (Stopped, 0 executors, 36 cores, 284 GB memory, TYPE HIVE UNIFIED ANALYTICS COMPACTOR), and another 'paris-atelier-impala' entry (Running, 0 executors, 36 cores, 284 GB memory, TYPE IMPALA).

2. This is your HUE tool.

Hue is a web-based interface, simplifying data exploration, job scheduling, and management.

It offers a user-friendly environment for running SQL queries, managing files, and visualizing data. Hue enhances productivity by providing a centralized platform for big data ecosystem components, making tasks easier and more accessible.

The screenshot shows the Hue web interface. On the left is a sidebar with icons for Home, Environment, Tables, Jobs, and Help. The main area has a search bar at the top. Below it, there's a 'Tables' section for 'user050' with two entries: 'telco_data_curated' and 'telco_jcbeberg_kafka'. A central panel shows a query editor with a placeholder 'Example: SELECT * FROM tablename, or press CTRL + space'. Below the editor are tabs for 'Query History' (which is empty) and 'Saved Queries' (also empty). To the right, there's a 'Jobs' section and a 'Tables' section which says 'No tables identified.'

3. Inside your Hue window, run the following SQL statement:

- Run the following statement:

```
describe formatted user0XX.telco_data_curated
```

You will get a visual description of your table as shown below:

	name	type	comment
1	# col_name	data_type	comment
2		NULL	NULL
3	multiplelines	string	NULL
4	paperlessbilling	string	NULL
5	gender	string	NULL
6	onlinesecurity	string	NULL
7	internetservice	string	NULL
8	techsupport	string	NULL
9	contract	string	NULL
10	churn	string	NULL
11	seniorcitizen	string	NULL
12	deviceprotection	string	NULL
13	streamingtv	string	NULL
14	streamingmovies	string	NULL
15	totalcharges	float	NULL
16	partner	string	NULL
17	monthlycharges	float	NULL
18	customerid	string	NULL
19	dependents	string	NULL

b. We are interested in a few properties, scroll to line 32. Notice the location

32 Location: s3a://paris-atelier/my-data/warehouse/tablespace/external/hive/user050.db/telco_data_curated NULL

c. Scroll to line 52. Notice the table_Type

52	table_type	ICEBERG
----	------------	---------

4. Inside your Hue window, run the following SQL statement:

describe history user0XX.telco_data_curated

	creation_time	snapshot_id	parent_id	is_current_ancestor
1	2023-10-11 09:22:59.302000000	77495325292598376	NULL	TRUE
2	2023-10-11 09:23:37.192000000	821975665367142062	77495325292598376	TRUE

Notice the snapshot history for your tables. Next we will go and query them.

5. Inside your Hue window, run the following SQL statement:

```
SELECT count (*)
FROM user0XX.telco_data_curated
for SYSTEM_VERSION as of <first_snapshot_id>
```

count(*)
1 0

6. Inside your Hue window, run the following SQL statement:

```
SELECT count (*)
FROM user050.telco_data_curated
for SYSTEM_VERSION as of <last_snapshot_id>
```

count(*)
1 7043

1	7043
---	------

7. Explore with free SQL

6.6. Part 2: Add a New Field - Optional

1. Edit the previously created Dataset, in Data -> <user_assigned>.telco_data_curated.

The screenshot shows the Cloudera Data Visualization interface. On the left, there's a sidebar with connection management (New Connection, All Connections, ImpalaConn, samples). The main area is titled 'Datasets' and shows a table with one row: 'user050.telco_data_curated'. The table includes columns for Title/Table, ID, Created, Last Updated, Modified By, and # Dashboards. A '+' icon is next to the title.

2. Once in the Dataset, go to **Fields** in the left menu and then click on **Edit Field** to edit the fields of your dataset.

The screenshot shows the 'Fields' page for the dataset 'user050.telco_data_curated'. The left sidebar has 'Fields' selected. The main area is divided into 'Dimensions' and 'Measures'. The 'Dimensions' section lists 18 fields: multipelines, paperlessbilling, gender, onlinesecurity, internetservice, techsupport, contract, churn, seniorcitizen, deviceprotection, streamingtv, streamingmovies, partner, customerid, dependents, onlinebackup, phoneservice, and paymentmethod. The 'Measures' section lists 3 fields: totalcharges, monthlycharges, and tenure. There are buttons for 'EDIT FIELDS' and 'NEW DASHBOARD' at the top right.

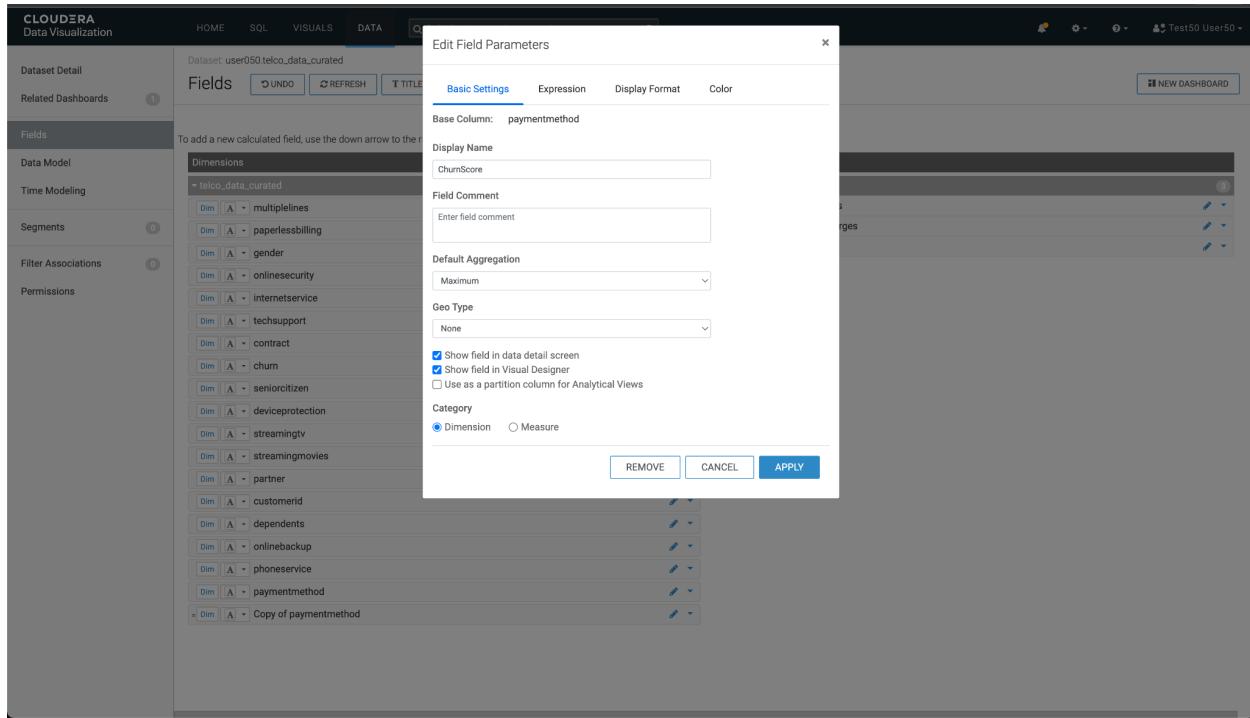
3. In the list of **Dimensions**, click the down arrow of the last field in the list, and select the option **Clone**.

The screenshot shows the Cloudera Data Visualization interface. On the left, there's a sidebar with sections like Dataset Detail, Related Dashboards, Fields, Data Model, Time Modeling, Segments, Filter Associations, and Permissions. The main area is titled 'Dataset: user050 telco_data_curated'. It has tabs for HOME, SQL, VISUALS, and DATA. Below the tabs is a search bar. The DATA tab is active, showing the 'Fields' section. Under 'Dimensions', there's a list of fields under 'telco_data_curated', including 'multipelines', 'paperlessbilling', 'gender', 'onlinesecurity', 'internetservice', 'techsupport', 'contract', 'churn', 'seniorcitizen', 'deviceprotection', 'streamingtv', 'streamingmovies', 'partner', 'customerid', 'dependents', 'onlinebackup', 'phoneservice', and 'paymentmethod'. The 'paymentmethod' field has a small downward arrow icon to its right. A context menu is open over this field, with 'Clone' being the highlighted option. To the right of the dimensions is a 'Measures' section with fields like 'totalcharges', 'monthlycharges', and 'tenure'. At the bottom right of the screen, there's a status bar with icons and the text 'Test50 User50'.

4. Once the field is cloned, click on the pencil next to the field to edit it.

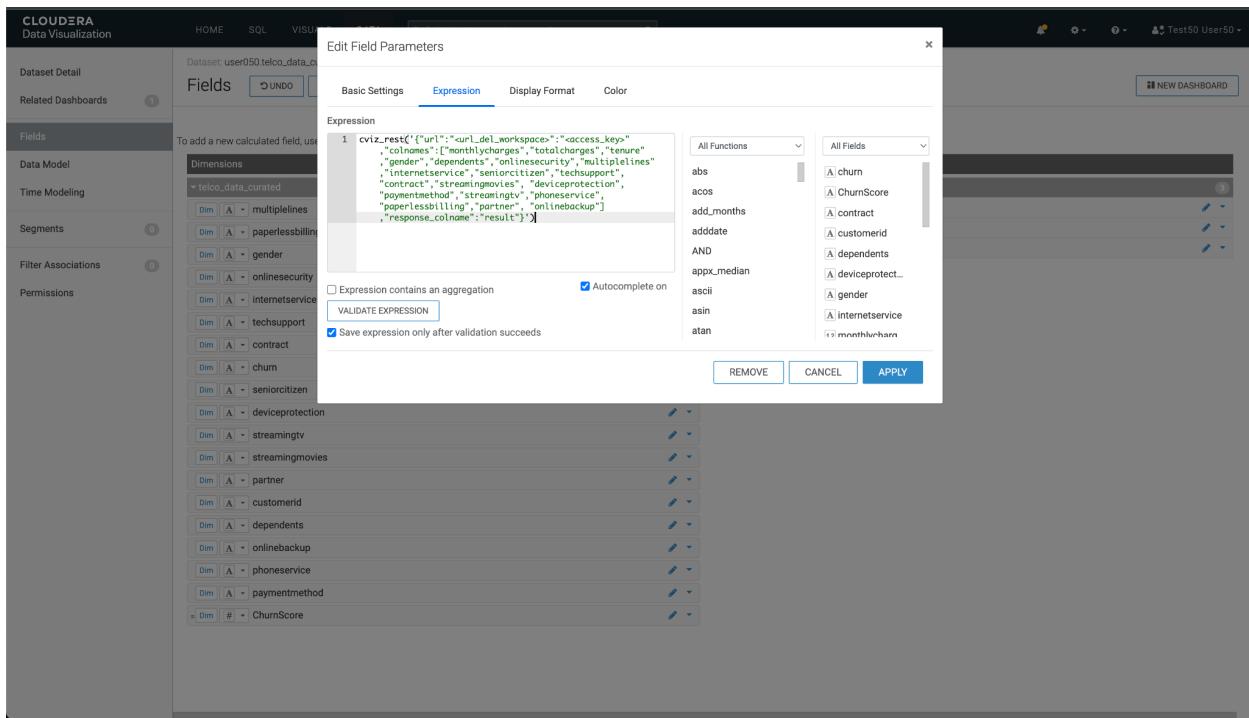
This screenshot shows the same interface as the previous one, but with a different state. The 'paymentmethod' dimension has been cloned, resulting in a new dimension named 'Copy of paymentmethod' at the bottom of the list. The 'Edit Field' button is visible to the right of this new dimension. The rest of the interface remains the same, with the sidebar, dataset details, and other dimensions listed above.

5. In the popup window that appears, enter the name of the new field in **Display Name**. We suggest that you enter *ChurnScore*.



6. Go to the Expressions tab and enter the following value in the Expression field. This will allow you to call the REST API of the Model you have previously deployed.

```
cviz_rest('{"url":"<url_del_workspace>","accessKey":"<access_key>","colnames":["monthlycharges","totalcharges","tenure","gender","dependents","onlinesecurity","multiplelines","internetservice","seniorcitizen","techsupport","contract","streamingmovies","deviceprotection","paymentmethod","streamingtvtv","phoneservice","paperlessbilling","partner","onlinebackup"],"response_colname":"result"}')
```



7. Being in CML in another tab of the web browser, go to the section of **Models** of your project, and click on the Model that begins with the name *Model/Viz*, followed by your assigned username.

Model	Source	Status	Replicas	CPU	Memory	Last Deployed	Actions
ModelViz_user050	13_mod...	Deployed	1 / 1	1	2.00 GiB	May 29, 2023, 03:54 PM	<button>Stop</button>
ModelOpsChurn_user050	11_best...	Deployed	1 / 1	1	2.00 GiB	May 29, 2023, 03:53 PM	<button>Stop</button>

Name	Runs / Failures	Duration	Status	Latest Run	Actions
deploy_best_model	0 / 0	00:00	Not Yet Run	-	<button>Run</button>
retrain	0 / 0	00:00	Not Yet Run	-	<button>Run</button>
avisoPerformance	0 / 0	00:00	Not Yet Run	-	<button>Run</button>
Check Model	0 / 0	00:00	Not Yet Run	-	<button>Run</button>

Name	Size	Last Modified
__pycache__	-	15 hours ago
flask	-	15 hours ago
images	-	15 hours ago
models	-	15 hours ago
raw	-	15 hours ago
0_bootstrap.py	1.95 kB	15 hours ago
0b_create_jobs.py	5.60 kB	15 hours ago

8. In the Overview tab, copy the URL that allows you to interact and call the workspace API.

The screenshot shows the Cloudera Machine Learning interface with the following details:

- Header:** CLOUDERA Machine Learning
- Breadcrumbs:** user050 / user050-telco-churn / Models / ModelViz_user050 / Overview
- Left Sidebar:**
 - All Projects
 - Overview
 - Sessions
 - Data
 - Experiments
 - Models** (selected)
 - Jobs
 - Applications
 - Files
 - Collaborators
 - Project Settings
- Page Content:**
 - ModelViz User Interface:** A visualization tool for a given model prediction. It includes:
 - Description: visualization a given model prediction
 - Sample Code: Shell, Python, R tabs. The Python tab contains a curl command to interact with the model service.
 - Sample Response: A JSON object representing the predicted output.
 - Model Details:** A table showing model metadata.

Source	Code
8	cm:cdp:mlus-west-1:508fd88f-8076-498a-acfb-6878c5cd35e8/workspace\$14194cb1c7e48cd-9989-b499a79ed5f6/dae534c1-b214-45eb-acd0-101e651ff8d8
Deployment Id	10
Deployment CRN	cm:cdp:mlus-west-1:508fd88f-8076-498a-acfb-6878c5cd35e8/workspace\$14194cb1c7e48cd-9989-b499a79ed5f6/cf985a5d-9870-4533-919a-d42ad0db56ed
Build Id	10
Build CRN	cm:cdp:mlus-west-1:508fd88f-8076-498a-acfb-6878c5cd35e8/workspace\$14194cb1c7e48cd-9989-b499a79ed5f6/0e00e2d9-80cb-4ee8-8304-79987673de32
Deployed By	user050
Comment	Initial revision.
Runtime Image	Python 3.7 (Standard)
File	13_model_viz.py
Function	predict
 - Model Resources:** A table showing deployment statistics.

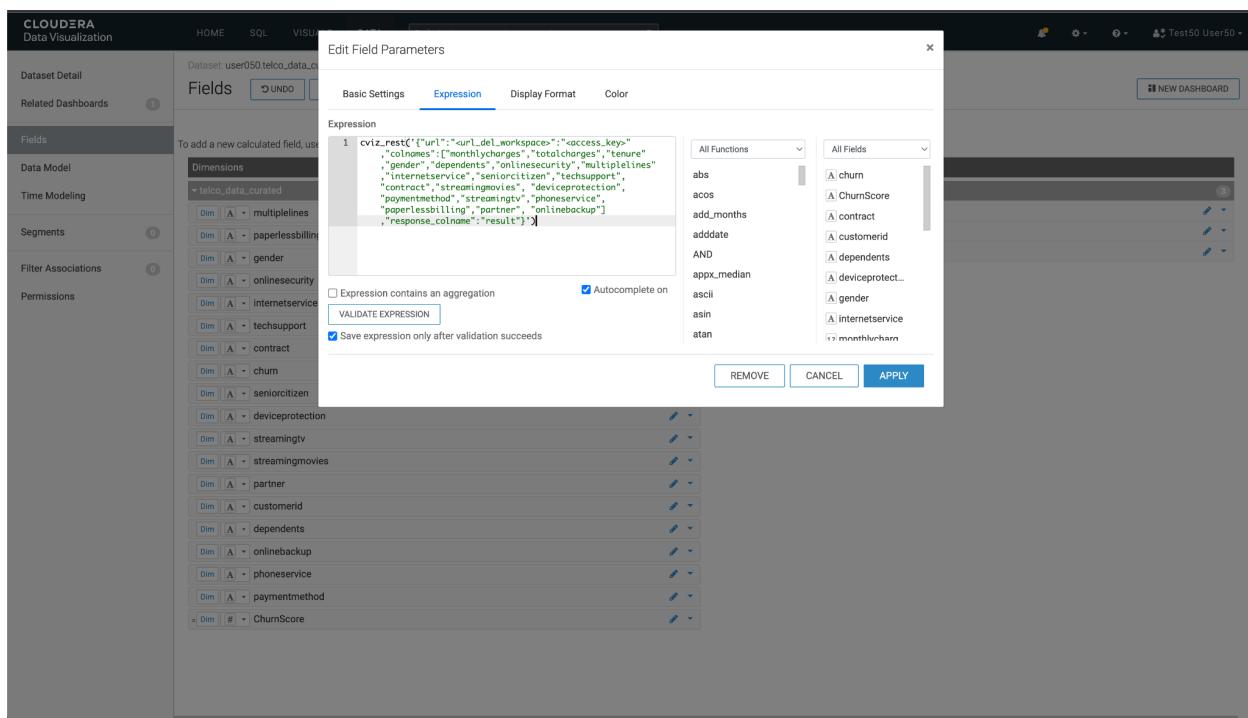
Replicas	1
Total CPU	1 vCPUs
Total Memory	2.00 GiB

Replace the copied value in the attribute <url_del_workspace> of the Expression field.

9. Returning to the CML, copy the accessKey of the model.

Replace the copied value in the attribute **<access_key>** of the Expression field. The format should be as follows, e.g.

```
cviz_rest('{"url":"https://modelservice.ml-b200bd6f-fb9.za-mtn-l.yu1t-vbzg.cloudera.site/model","accessKey":"mjy1fowabqiwpfjb19s9ht6xmuvy0f2j","colnames":["monthlycharges","totalcharges","tenure","gender","dependents","onlinesecurity","multiplelines","internetservice","seniorcitizen","techsupport","contract","streamingmovies","deviceprotection","paymentmethod","streamingtvtv","phoneservice","paperlessbilling","partner","onlinebackup"],"response_colname":"result"})
```



10. Finish the process of copying the *url del workspace* and the *accessKey*, click the Validate Expression button at the top of the window. If the message appears in green *Validation Successful*, Click on **Apply** to save the settings made.

The screenshot shows the Cloudera Data Visualization interface. A modal window titled 'Edit Field Parameters' is open, specifically on the 'Expression' tab. The expression input field contains the following JSON code:

```

1 cviz.restC["url": "https://modelservice.ml-369083c3-99e.ssa-nj.ykt-02.cloud.tencent.com:8081", "accessKey": "FamrJzq4X01GKmWm0b8Qc", "secretKey": ":[{"monthlycharges": "totalcharges", "tenure": "gender", "dependents": "onlinesecurity", "multiplelines": "internetservice", "seniorcitizen": "techsupport", "contract": "streamingmovies", "dependents": "deviceprotection", "paymentmethod": "streamingtv", "onlineservice": "paperlessbilling", "partner": "onlinebackup"}, {"response_colname": "result"}]

```

Below the expression editor, there are several checkboxes: 'Expression contains an aggregation', 'Autocomplete on VALIDATE EXPRESSION', and 'Save expression only after validation succeeds'. The 'VALIDATE EXPRESSION' button is highlighted in blue. At the bottom of the dialog, a green bar says 'Validation Successful!'. There are 'REMOVE', 'CANCEL', and 'APPLY' buttons at the bottom right.

11. The new field should appear in the list of fields. Change the data type, selecting the type *Integer*, which is represented by the symbol #

The screenshot shows the Cloudera Data Visualization interface with the 'Fields' section open. A dropdown menu is displayed over a dimension field, listing various data types: Boolean, # Integer, .12 Real, A String, and T Timestamp. The '# Integer' option is currently selected. To the right of the dropdown, there are sections for 'Dimensions' and 'Measures', each containing a list of fields from the 'telco_data_curated' dataset.

12. Finish the process by clicking on the green button with the legend **SAVE** in the top menu.

The screenshot shows the 'Dataset Detail' page in Cloudera Data Visualization. The left sidebar lists 'Data Model', 'Time Modeling', 'Segments', 'Filter Associations', and 'Permissions'. The main area displays the 'Fields' section for the dataset 'user050 telco_data_curated'. It is divided into 'Dimensions' and 'Measures' panels. The 'Dimensions' panel contains 19 items, and the 'Measures' panel contains 3 items. At the top right of the main area, there is a green 'SAVE' button with a white icon. Other buttons include 'UNDO', 'REFRESH', 'TITLE CASE', and 'Show Comments'. A 'NEW DASHBOARD' button is also present in the top right corner.

13. Return to the dashboard, selecting the option **VISUALS** from the top menu, and clicking on the name of the dashboard that was previously created.

The screenshot shows the 'VISUALS' dashboard in Cloudera Data Visualization. The left sidebar shows 'WORKSPACES' with 'Public' and 'Private' options. The main area displays a grid of various dashboards. One dashboard, titled 'Churn Analysis', is highlighted with a red box. Other dashboards visible include 'Deficiency Details', 'State of NYC', 'Sample App', 'Store Details', 'Cereal Comparisons', 'Earthquakes Around the World', 'Life Expectancy Dashboard', 'World Population & GDP Trends', 'Animated world population - GDP vs HI', 'US State Population Trends', 'Census Dashboard', 'Global Threats', 'Inspector View', 'Consumer View', 'Iris species w/ images', and 'Taxi rides application'. Action buttons at the top of the dashboard area include 'NEW DASHBOARD', 'NEW APP', 'MOVE TO WORKSPACE', 'EXPORT', and 'DELETE'.

14. Once in the dashboard, click on the button **Edit** which is in the upper left.

The screenshot shows a Cloudera Data Visualization dashboard. At the top, there are navigation tabs: HOME, SQL, VISUALS, and DATA. On the far right, it shows the user 'Test50 User50'. Below the tabs, there is an 'EDIT' button and a 'PRIVATE' dropdown. The main area contains three bar charts under the heading 'streamingtv s'. Each chart has 'Record Count' on the y-axis (0 to 2,000) and 'streamingmoves' on the x-axis. The first chart is for 'No internet service', the second for 'Yes', and the third for 'partner'. Each chart has two bars: 'No' (dark teal) and 'Yes' (light teal). Below the charts is a data grid with columns: totalcharges, monthlycharges, tenure, multiplelines, paperlessbilling, gender, onlinesecurity, internetservice, techsupport, contract, and chu. The data grid shows several rows of customer information. At the bottom of the grid, there are page navigation buttons: < 1 2 3 4 5 >.

totalcharges	monthlycharges	tenure	multiplelines	paperlessbilling	gender	onlinesecurity	internetservice	techsupport	contract	chu
29.850000381469727	32.602622985839844	1	No phone service	Yes	Female	No	DSL	No	Month-to-month	NK
1,889.5	79.32872009277344	34	No	No	Male	Yes	DSL	No	One year	NK
108.1500015258789	53.849998474121094	2	No	Yes	Male	Yes	DSL	No	Month-to-month	Ye
1,840.75	39.008785247802734	45	No phone service	No	Male	Yes	DSL	Yes	One year	Nc
151.64999389648438	70.69999694824219	2	No	Yes	Female	No	Fiber optic	No	Month-to-month	Ye
820.5	99.6500015258789	8	Yes	Yes	Female	No	Fiber optic	No	Month-to-month	Ye

15. Edit the lower table by clicking on it and then on the option **Build** from the right vertical menu. Add the new field, **ChurnScore**, at the beginning of the table, by clicking and dragging from the option **Dimensions** available.

The screenshot shows the Cloudera Data Visualization interface. On the left, there are three bar charts under the heading "streamingtv". Each chart has "Record Count" on the y-axis (0 to 2,000) and "streamingmovies" on the x-axis. The first chart is for "No internet service", the second for "Yes". The legend indicates "No" (light blue) and "Yes" (dark blue). The third chart is for "partner". Below the charts is a table with columns: totalcharges, monthlycharges, tenure, multiplelines, paperlessbilling, gender, onlinesecurity, internetservice, techsupport, and contract. The table contains several rows of data. On the right side of the interface is the "Dashboard Designer" panel. The "Dimensions" section is expanded, showing a list of fields. The field "ChurnScore" is highlighted with a red box. Other dimensions listed include totalcharges, monthlycharges, tenure, multiplelines, paperlessbilling, gender, onlinesecurity, internetservice, techsupport, contract, and partner. The "Measures" section shows "Record Count" selected. The "Visuals" section shows "Record Count" selected. The "Build" button is highlighted in the vertical toolbar on the right.

16. Click on the Refresh Visual button to update the data. The new column should appear **ChurnScore** then at the beginning of the table, with a value of numeric type. Finish the process by clicking the button **SAVE** from the top left menu.

The screenshot shows the Cloudera Data Visualization interface. At the top, there are navigation tabs: HOME, SQL, VISUALS, and DATA. Below the tabs, there are buttons for VIEW, LAYOUT, and SAVE, along with a PRIVATE dropdown. On the right side, there's a sidebar titled "Dashboard Designer" with sections for DATA, DASH., and VISUAL. The DATA section contains a tree view of dimensions and measures, including "telco_data_curated" and "Record Count". The VISUAL section shows three stacked bar charts under the title "streamingtv". The first chart is for "No internet service", the second for "Yes", and the third for "partner". Each chart has two bars: "No" (light blue) and "Yes" (dark blue). The Y-axis is labeled "Record Count" and ranges from 0 to 2,000. The X-axis is labeled "streamingmovies". Below the charts is a table with the following data:

	totalcharges	monthlycharges	tenure	multiplelines	paperlessbilling	gender	onlinesecurity	internetservice	techsupport
0	29.850000381469727	32.602622985839844	1	No phone service	Yes	Female	No	DSL	No
0	1,889.5	79.32872009277344	34	No	No	Male	Yes	DSL	No
0	108.1500015258789	53.849998474121094	2	No	Yes	Male	Yes	DSL	No
0	1,840.75	39.008785247802734	45	No phone service	No	Male	Yes	DSL	Yes
6	151.64999389548438	70.69999694824219	2	No	Yes	Female	No	Fiber optic	No
10	820.5	99.6500015258789	8	Yes	Yes	Female	No	Fiber optic	No

At the bottom right of the table area, there is a "REFRESH VISUAL" button. The status bar at the bottom left says "Sheet 1" and the bottom right says "1 2".

7. Take-aways

Cloudera Data Platform (CDP) is a hybrid data platform designed for unmatched freedom to choose—any cloud, any analytics, any data.

CDP delivers faster and easier data management and data analytics for data anywhere, with optimal performance, scalability, and security.

With CDP you get all the advantages of CDP Private Cloud and CDP Public Cloud for faster time to value and increased IT control.