OCTOBER 26, 2023

Platform Developer Roadmap: Leveraging new capabilities within AVEVA's Industrial Platform

Ecosystem Track

Joshua Kidd, Scott Dunham

AVEVA

Agenda

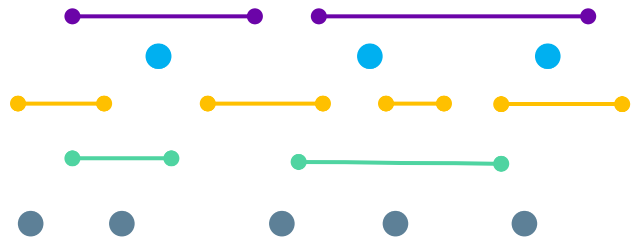
- What is new or in development on AVEVA Data Hub for developers
- Why the capabilities were built
- How will you interact with these capabilities

AVEVA

New data types and query patterns in AVEVA Data Hub

Event Data Store

Ability to store events with surrounding context and provide a rich contextual search API for retrieving the information

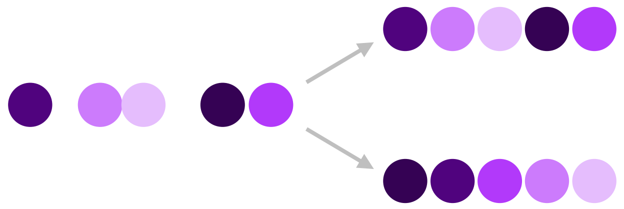


Data Views supports Parquet

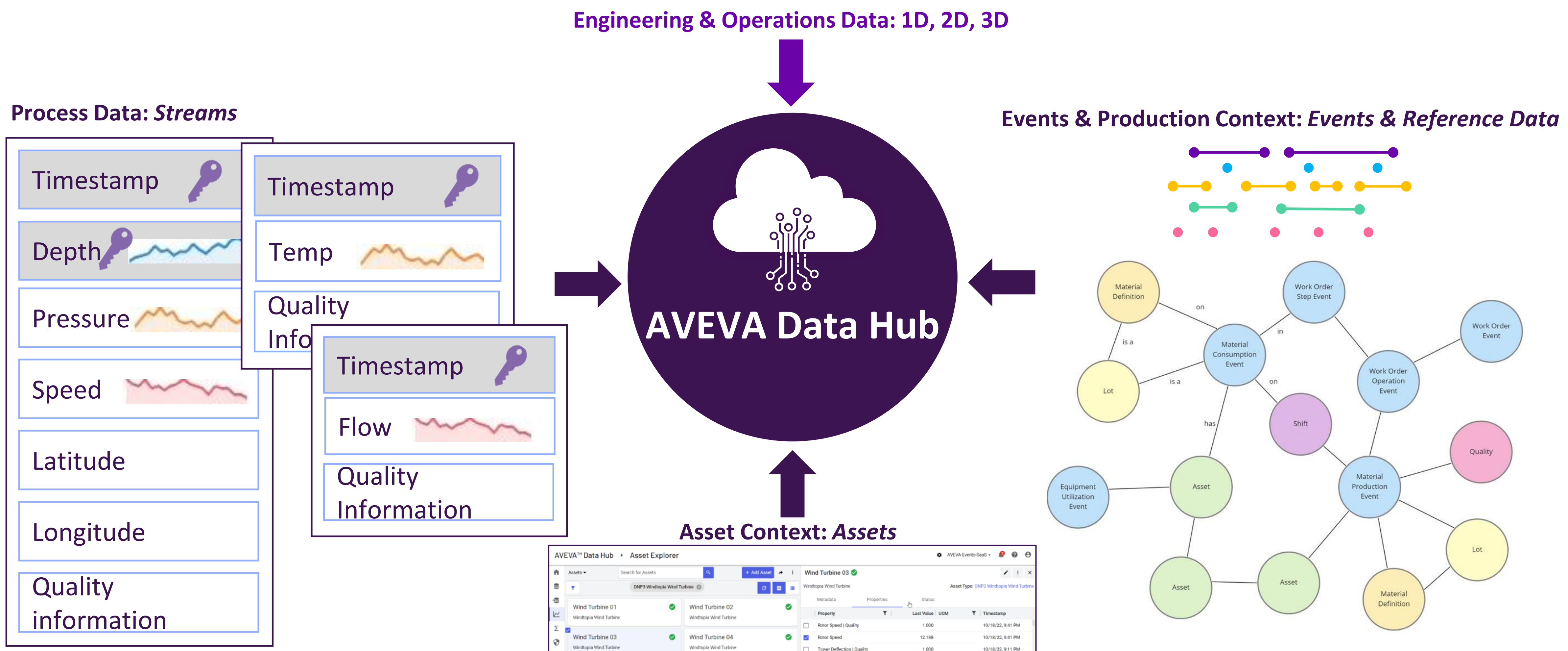
Ability to query data sets in a format familiar and common to data scientists

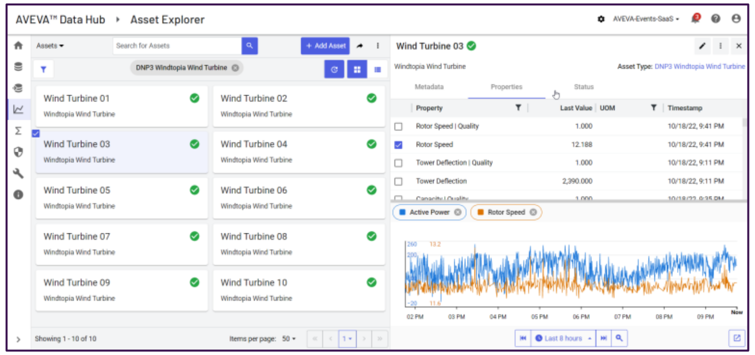
Change Broker

Ability to sign up for and query changes to stream data



Solid foundation to expanding the industrial data footprint




AVEVA™ Data Hub › Asset Explorer

AVEVA-Events-SaaS

Assets · Search for Assets · + Add Asset

DNP3 Windtopia Wind Turbine

Wind Turbine 01 — Windtopia Wind Turbine
Wind Turbine 02 — Windtopia Wind Turbine
Wind Turbine 03 — Windtopia Wind Turbine
Wind Turbine 04 — Windtopia Wind Turbine
Wind Turbine 05 — Windtopia Wind Turbine
Wind Turbine 06 — Windtopia Wind Turbine
Wind Turbine 07 — Windtopia Wind Turbine
Wind Turbine 08 — Windtopia Wind Turbine
Wind Turbine 09 — Windtopia Wind Turbine
Wind Turbine 10 — Windtopia Wind Turbine

Showing 1 - 10 of 10 · Items per page: 50

Wind Turbine 03

Windtopia Wind Turbine · Asset Type: DNP3 Windtopia Wind Turbine

Metadata · Properties · Status

Property	Last Value	UOM	Timestamp
Rotor Speed | Quality	1.000		10/18/22, 9:41 PM
Rotor Speed	12.188		10/18/22, 9:41 PM
Tower Deflection | Quality	1.000		10/18/22, 9:11 PM
Tower Deflection	2,390.000		10/18/22, 9:11 PM

Active Power · Rotor Speed

02 PM · 03 PM · 04 PM · 05 PM · 06 PM · 07 PM · 08 PM · 09 PM · Now

Last 8 hours

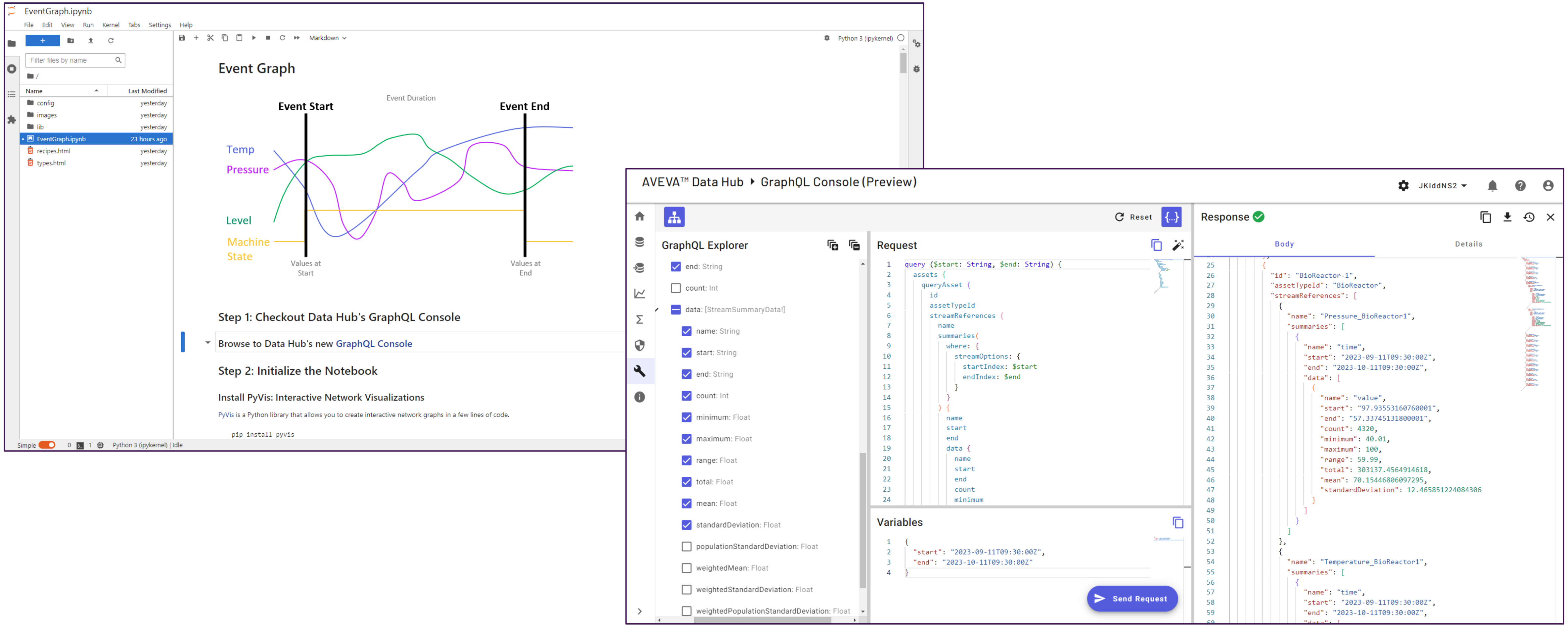
GraphQL

- A query language that provides a simple declarative way to retrieve data by accepting a nested object structure as a query rather than a text string
- A GraphQL schema is composed of types and operations {queries, mutations}

```
type BatchEvent {
  id: ID!
  name: String!
  description: String
  startTime: DateTime!
  endTime: DateTime
  duration: TimeSpan
  state: EventState
  asset: Asset
  unitOperationEvents: [UnitOperationEvent]!
}
```

```
query {
  queryBatchEvents(
    where: { startTime: { gt: “2023-09-01 11:00:00” } },
    options: { size: 10, sort: { startTime:  “DESC” } }
  ){
    id
    startTime
    duration
    state
    asset {
      id
      typeId
      streamReferences {
        name
      }
    }
  }
}
```

Tools: Python Notebook and Data Hub’s GraphQL Console



Name	Last Modified
config	yesterday
images	yesterday
lib	yesterday
EventGraph.ipynb	23 hours ago
recipes.html	yesterday
types.html	yesterday

Event Graph

Step 1: Checkout Data Hub's GraphQL Console

Browse to Data Hub's new GraphQL Console

Step 2: Initialize the Notebook

Install PyVis: Interactive Network Visualizations

PyVis is a Python library that allows you to create interactive network graphs in a few lines of code.

```
pip install pyvis
```

GraphQL Explorer

- ✓ end: String
- ☐ count: Int
- data: [StreamSummaryData!]
  - ✓ name: String
  - ✓ start: String
  - ✓ end: String
  - ✓ count: Int
  - ✓ minimum: Float
  - ✓ maximum: Float
  - ✓ range: Float
  - ✓ total: Float
  - ✓ mean: Float
  - ✓ standardDeviation: Float
  - ☐ populationStandardDeviation: Float
  - ☐ weightedMean: Float
  - ☐ weightedStandardDeviation: Float
  - ☐ weightedPopulationStandardDeviation: Float

Request

```
query ($start: String, $end: String) {
  assets {
    queryAsset {
      id
      assetTypeId
      streamReferences {
        name
        summaries(
          where: {
            streamOptions: {
              startIndex: $start
              endIndex: $end
            }
          }
        ) {
          name
          start
          end
          data {
            name
            start
            end
            count
            minimum
```

Variables

```
{
  "start": "2023-09-11T09:30:00Z",
  "end": "2023-10-11T09:30:00Z"
}
```

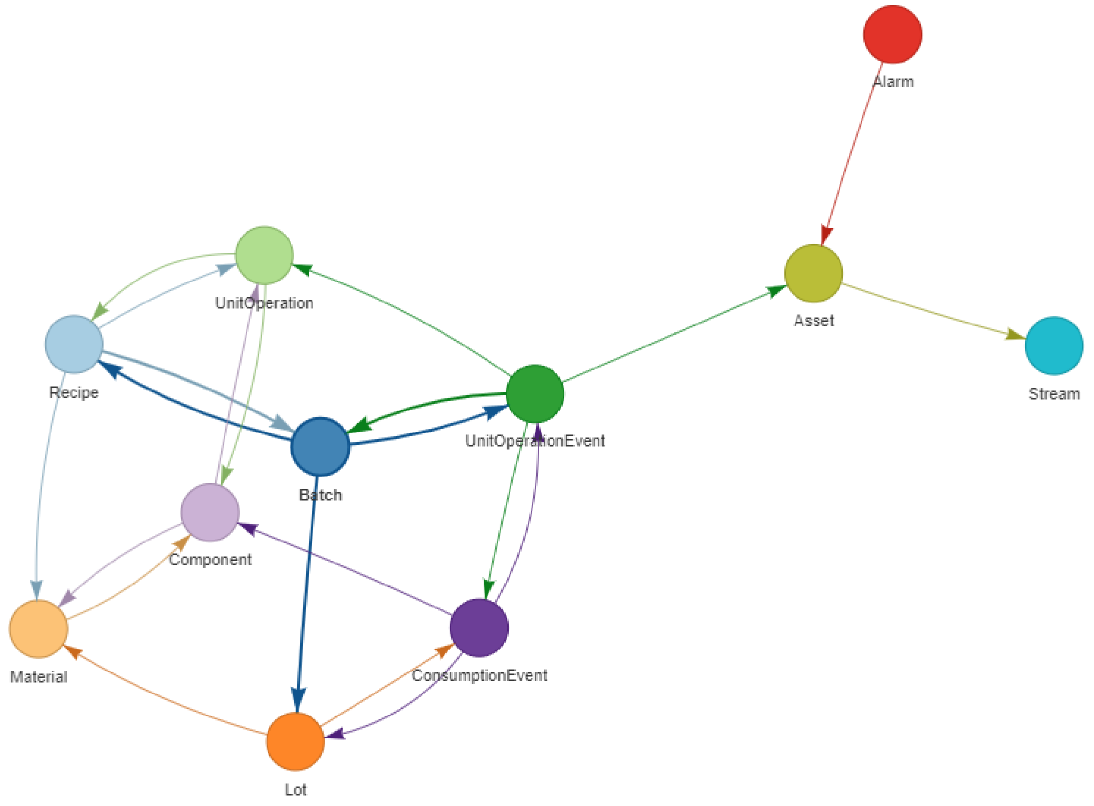
Response

```
    {
      "id": "BioReactor-1",
      "assetTypeId": "BioReactor",
      "streamReferences": [
        {
          "name": "Pressure_BioReactor1",
          "summaries": [
            {
              "name": "time",
              "start": "2023-09-11T09:30:00Z",
              "end": "2023-10-11T09:30:00Z",
              "data": [
                {
                  "name": "value",
                  "start": "97.93553160760001",
                  "end": "57.33745131800001",
                  "count": 4320,
                  "minimum": 40.01,
                  "maximum": 100,
                  "range": 59.99,
                  "total": 303137.4564914618,
                  "mean": 70.15446806097295,
                  "standardDeviation": 12.465851224084306
                }
              ]
            }
          ]
        },
        {
          "name": "Temperature_BioReactor1",
          "summaries": [
            {
              "name": "time",
              "start": "2023-09-11T09:30:00Z",
              "end": "2023-10-11T09:30:00Z",
```

Demo Scenario: Batch Manufacturing Process

GraphQL Schema

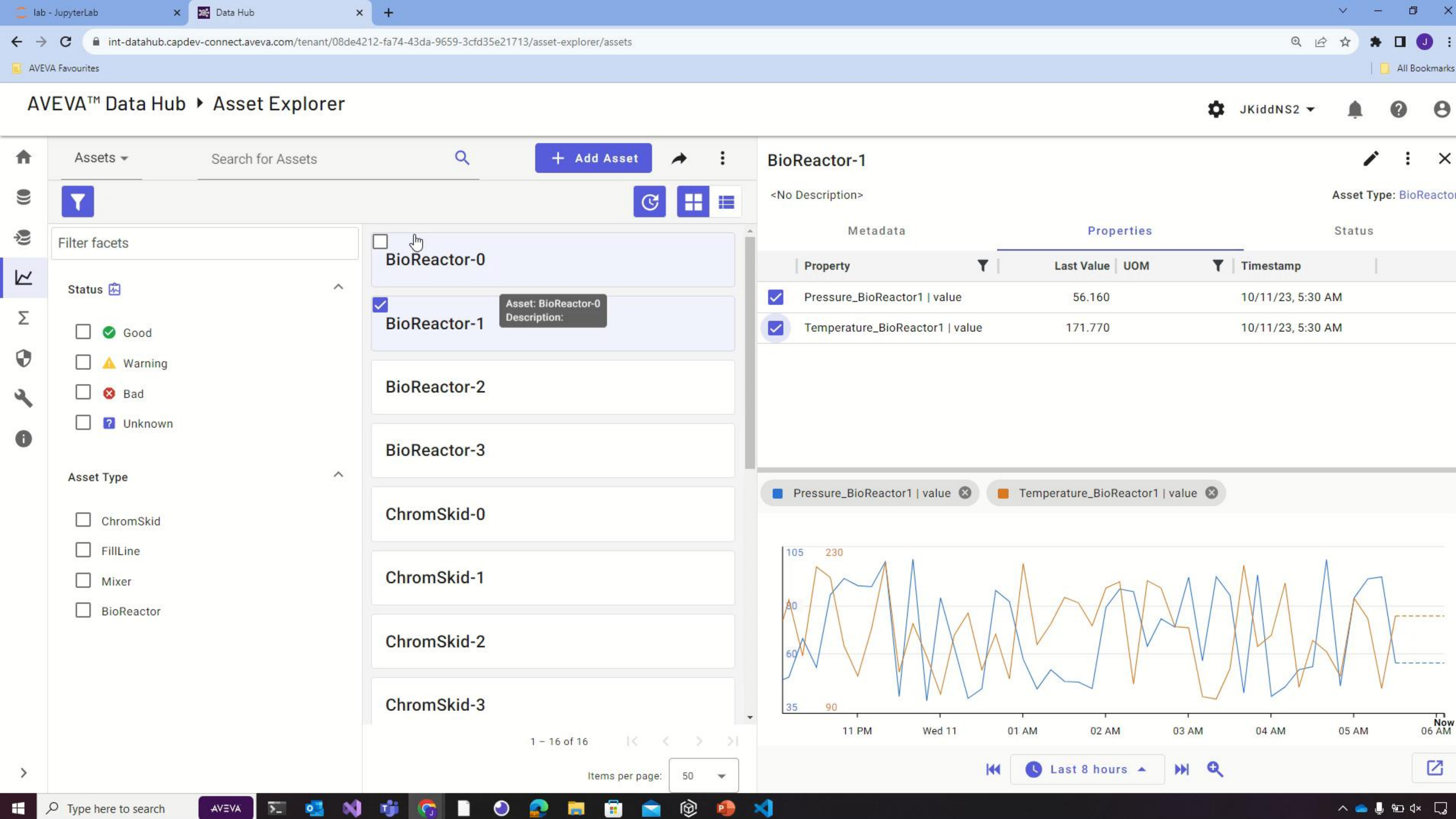
- **Reference Data Types (Context)**
  - *Material*: represents a material type definition. An instance may be created to define a raw material, process intermediate, or a finished product.
  - *Recipe*: defines the equipment, materials and process steps required to manufacturer a product.
  - *UnitOperation*: an individual process within a recipe.
  - *Component*: defines the sequence and target quantity of a raw material ingredient.
- **Event Types**
  - *Batch*: The complete execution record of a recipe instance that manufactured a material lot. Includes specifics about the equipment used, raw materials consumed, and process steps executed.
  - *UnitOperationEvent*: An individual process step executed as part of a batch. Includes the equipment used and the raw materials consumed.
  - *ConsumptionEvent*: Event representing the consumption of a raw material component within a unit operation event
  - *Lot*: An event the models a group of material containers that share a common lot number.
  - *Alarm*: An event representing a process deviation or anomaly.



Demo:

Create a Graph Schema, Upsert Data, and Perform Queries

AVEVA



lab - JupyterLab | Data Hub

int-datahub.capdev-connect.aveva.com/tenant/08de4212-fa74-43da-9659-3cfd35e21713/asset-explorer/assets

AVEVA Favourites | All Bookmarks

AVEVA™ Data Hub ▸ Asset Explorer

JKiddNS2

Assets | Search for Assets | + Add Asset

Filter facets

Status

- ☐ Good
- ☐ Warning
- ☐ Bad
- ☐ Unknown

Asset Type

- ☐ ChromSkid
- ☐ FillLine
- ☐ Mixer
- ☐ BioReactor

- ☐ BioReactor-0
- ☑ BioReactor-1
- BioReactor-2
- BioReactor-3
- ChromSkid-0
- ChromSkid-1
- ChromSkid-2
- ChromSkid-3

Asset: BioReactor-0
Description:

1 – 16 of 16

Items per page: 50

BioReactor-1

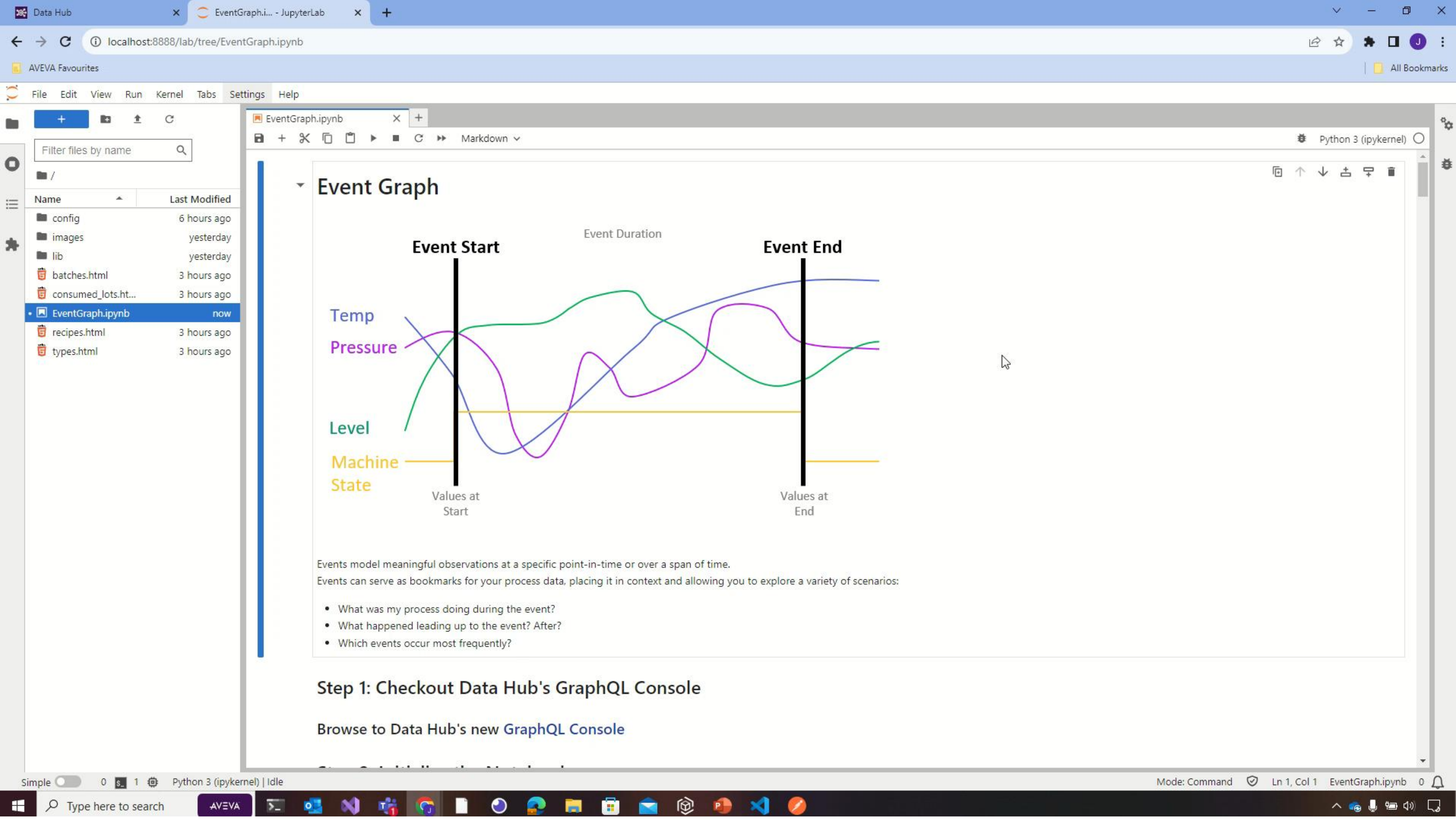
<No Description>

Asset Type: BioReactor

Metadata | Properties | Status

	Property	Last Value	UOM	Timestamp
☑	Pressure_BioReactor1 | value	56.160		10/11/23, 5:30 AM
☑	Temperature_BioReactor1 | value	171.770		10/11/23, 5:30 AM

Type here to search



Data Hub
EventGraph.i... - JupyterLab
localhost:8888/lab/tree/EventGraph.ipynb
AVEVA Favourites
All Bookmarks
File Edit View Run Kernel Tabs Settings Help
Filter files by name
EventGraph.ipynb
Markdown
Python 3 (ipykernel)

Name	Last Modified
config	6 hours ago
images	yesterday
lib	yesterday
batches.html	3 hours ago
consumed_lots.ht...	3 hours ago
EventGraph.ipynb	now
recipes.html	3 hours ago
types.html	3 hours ago

Event Graph

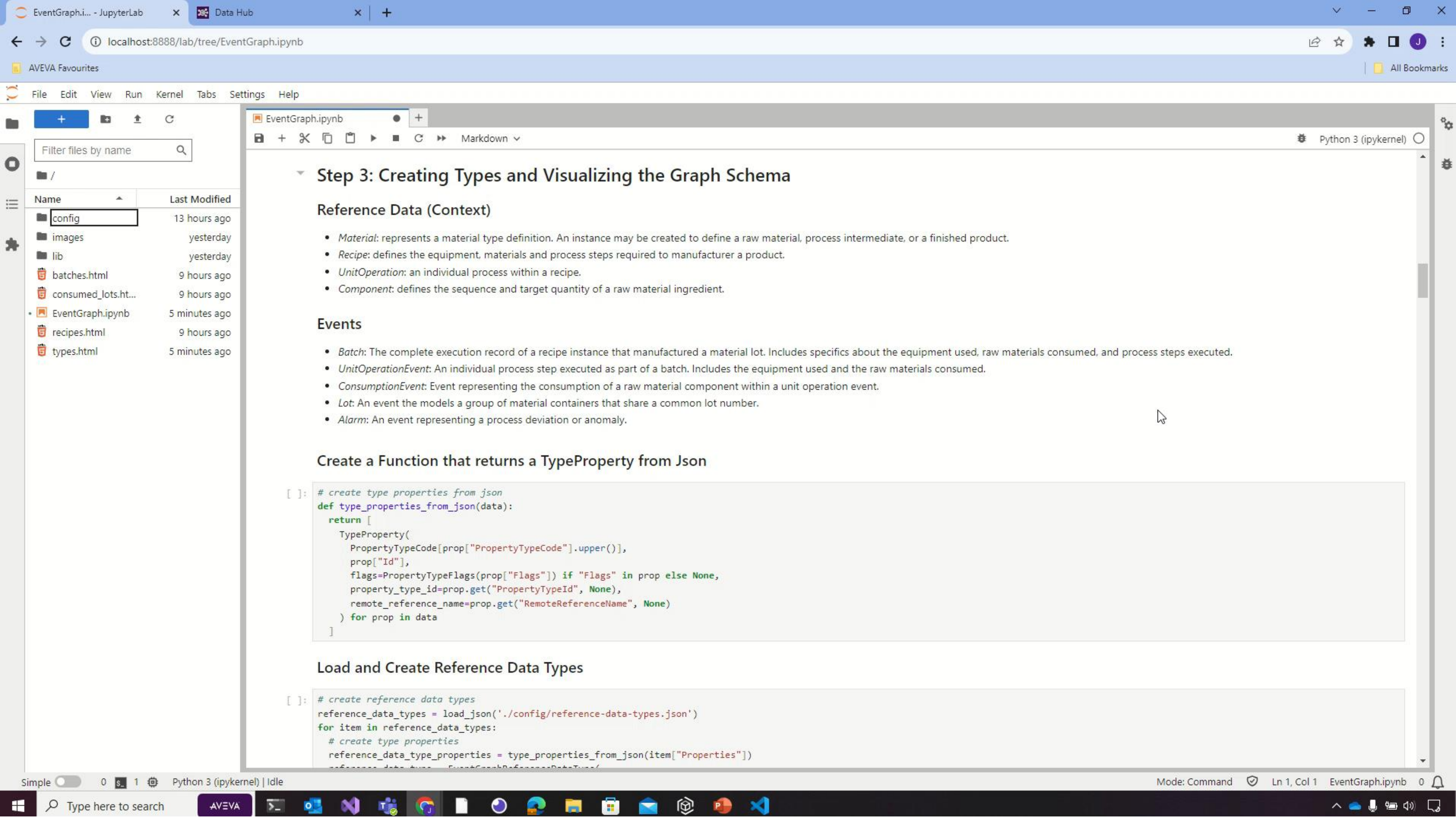
Events model meaningful observations at a specific point-in-time or over a span of time.
Events can serve as bookmarks for your process data, placing it in context and allowing you to explore a variety of scenarios:

- What was my process doing during the event?
- What happened leading up to the event? After?
- Which events occur most frequently?

Step 1: Checkout Data Hub's GraphQL Console

Browse to Data Hub's new GraphQL Console

Simple 0 1 Python 3 (ipykernel) | Idle
Mode: Command Ln 1, Col 1 EventGraph.ipynb 0
Type here to search
AVEVA



Step 3: Creating Types and Visualizing the Graph Schema

Reference Data (Context)

- *Material*: represents a material type definition. An instance may be created to define a raw material, process intermediate, or a finished product.
- *Recipe*: defines the equipment, materials and process steps required to manufacturer a product.
- *UnitOperation*: an individual process within a recipe.
- *Component*: defines the sequence and target quantity of a raw material ingredient.

Events

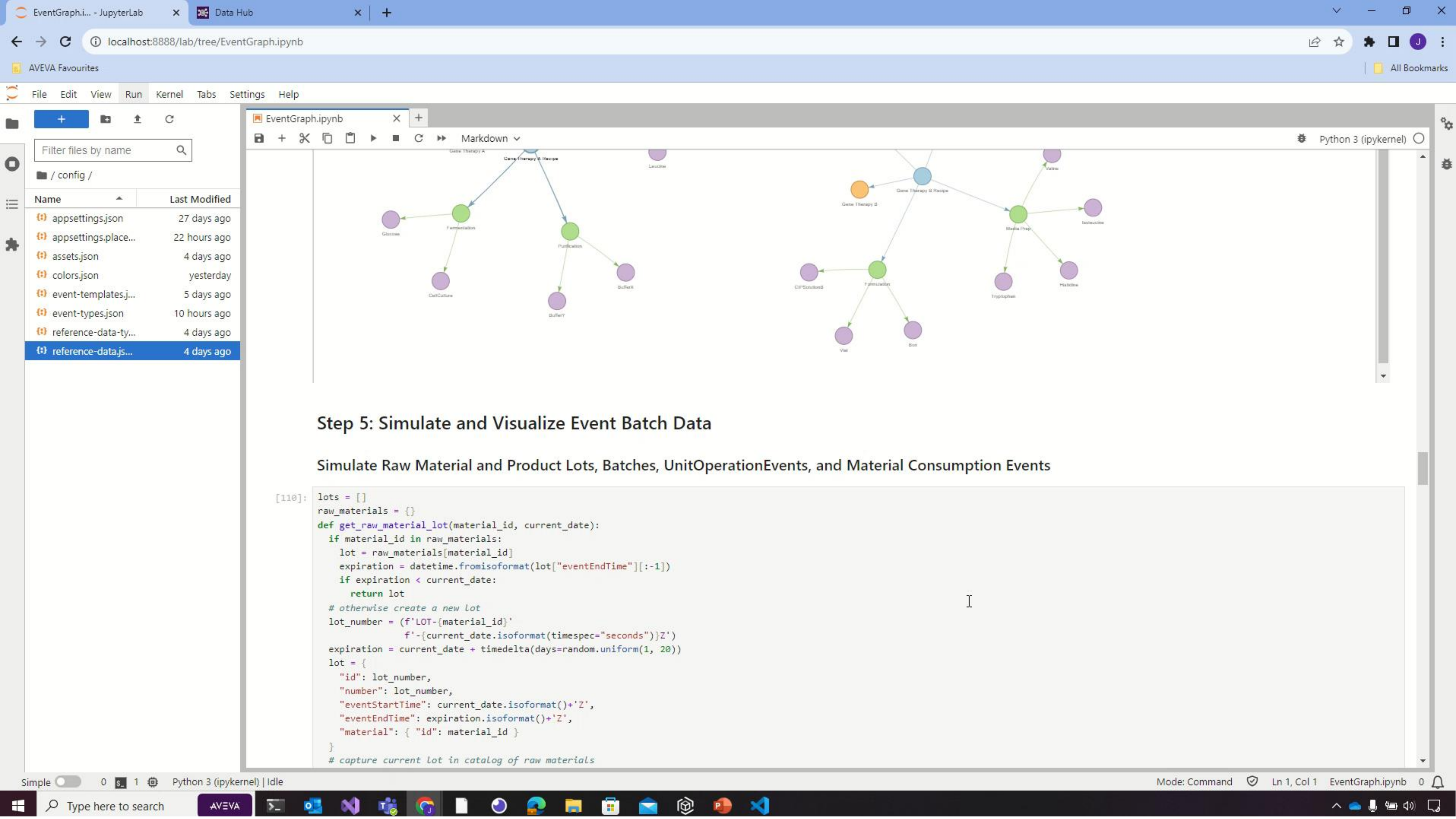
- *Batch*: The complete execution record of a recipe instance that manufactured a material lot. Includes specifics about the equipment used, raw materials consumed, and process steps executed.
- *UnitOperationEvent*: An individual process step executed as part of a batch. Includes the equipment used and the raw materials consumed.
- *ConsumptionEvent*: Event representing the consumption of a raw material component within a unit operation event.
- *Lot*: An event the models a group of material containers that share a common lot number.
- *Alarm*: An event representing a process deviation or anomaly.

Create a Function that returns a TypeProperty from Json

[ ]:
```
# create type properties from json
def type_properties_from_json(data):
  return [
    TypeProperty(
      PropertyTypeCode[prop["PropertyTypeCode"].upper()],
      prop["Id"],
      flags=PropertyTypeFlags(prop["Flags"]) if "Flags" in prop else None,
      property_type_id=prop.get("PropertyTypeId", None),
      remote_reference_name=prop.get("RemoteReferenceName", None)
    ) for prop in data
  ]
```

Load and Create Reference Data Types

[ ]:
```
# create reference data types
reference_data_types = load_json('./config/reference-data-types.json')
for item in reference_data_types:
  # create type properties
  reference_data_type_properties = type_properties_from_json(item["Properties"])
```

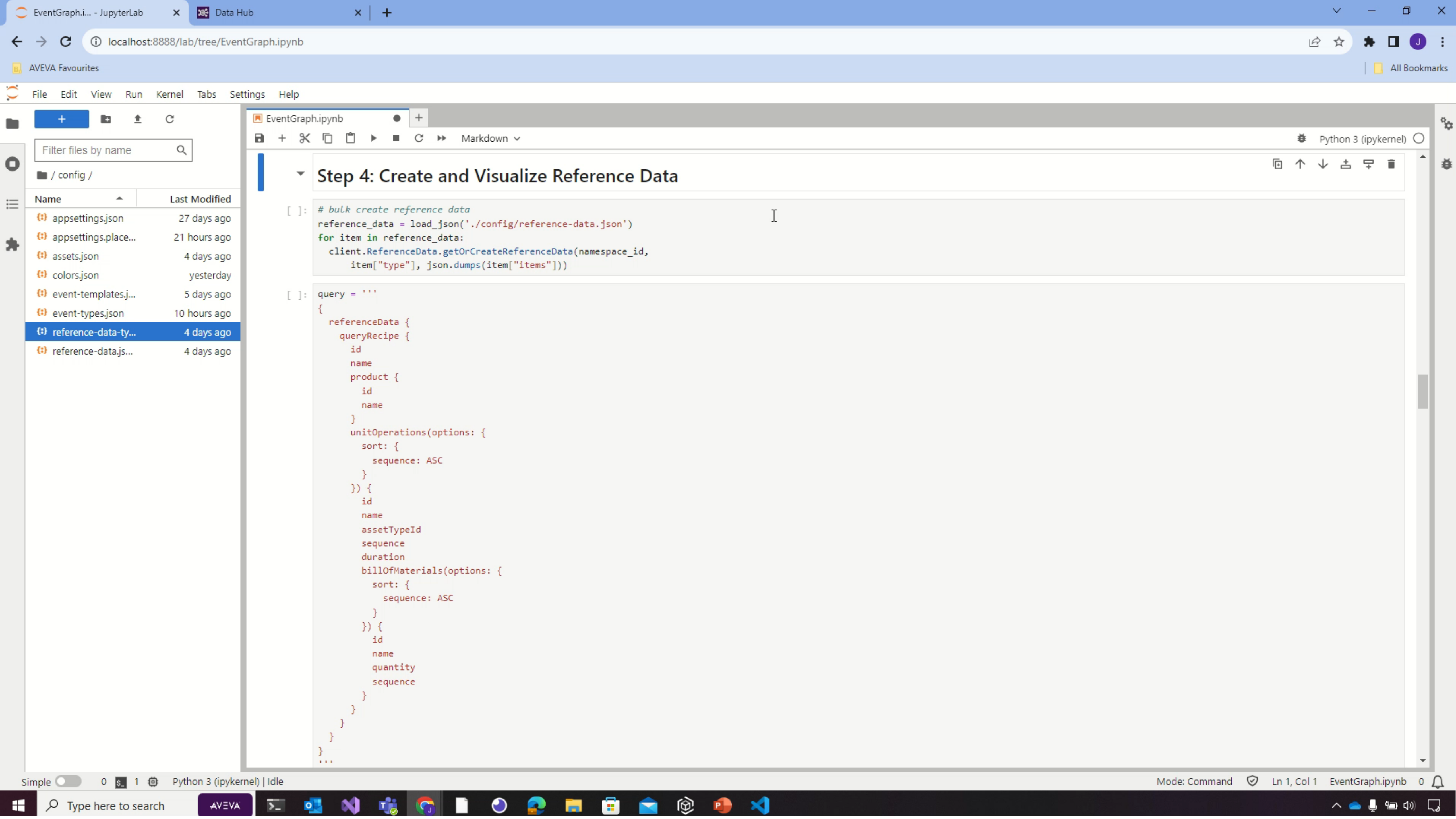


Name	Last Modified
appsettings.json	27 days ago
appsettings.place...	22 hours ago
assets.json	4 days ago
colors.json	yesterday
event-templates.j...	5 days ago
event-types.json	10 hours ago
reference-data-ty...	4 days ago
reference-data.js...	4 days ago

Step 5: Simulate and Visualize Event Batch Data

Simulate Raw Material and Product Lots, Batches, UnitOperationEvents, and Material Consumption Events

```
[110]:
lots = []
raw_materials = {}
def get_raw_material_lot(material_id, current_date):
  if material_id in raw_materials:
    lot = raw_materials[material_id]
    expiration = datetime.fromisoformat(lot["eventEndTime"][:-1])
    if expiration < current_date:
      return lot
  # otherwise create a new lot
  lot_number = (f'LOT-{material_id}'
                f'-{current_date.isoformat(timespec="seconds")}Z')
  expiration = current_date + timedelta(days=random.uniform(1, 20))
  lot = {
    "id": lot_number,
    "number": lot_number,
    "eventStartTime": current_date.isoformat()+'Z',
    "eventEndTime": expiration.isoformat()+'Z',
    "material": { "id": material_id }
  }
  # capture current lot in catalog of raw materials
```

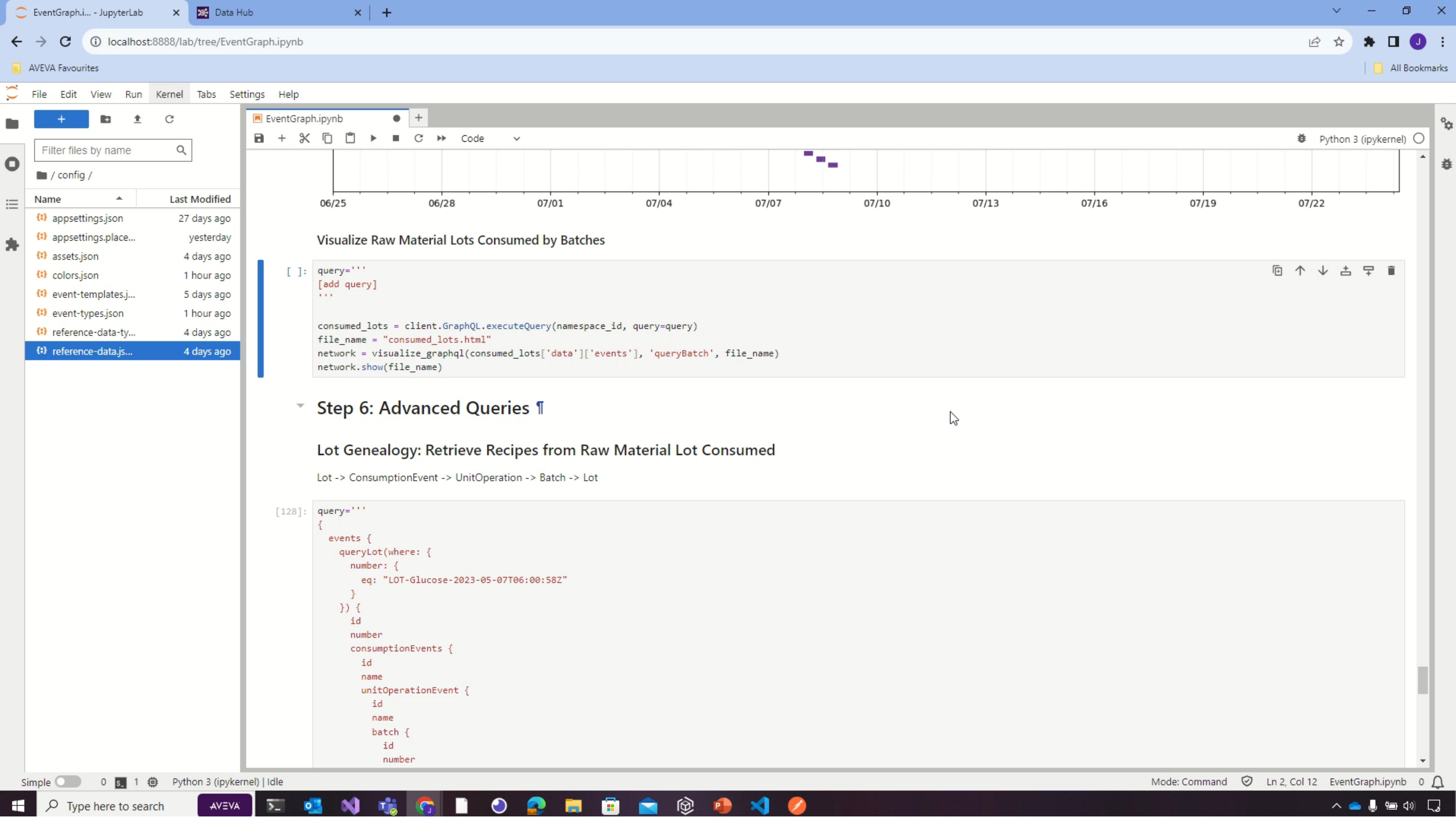


Name	Last Modified
appsettings.json	27 days ago
appsettings.place...	21 hours ago
assets.json	4 days ago
colors.json	yesterday
event-templates.j...	5 days ago
event-types.json	10 hours ago
reference-data-ty...	4 days ago
reference-data.js...	4 days ago

Step 4: Create and Visualize Reference Data

```
[ ]: # bulk create reference data
     reference_data = load_json('./config/reference-data.json')
     for item in reference_data:
       client.ReferenceData.getOrCreateReferenceData(namespace_id,
           item["type"], json.dumps(item["items"]))
```

```
[ ]: query = '''
     {
       referenceData {
         queryRecipe {
           id
           name
           product {
             id
             name
           }
           unitOperations(options: {
             sort: {
               sequence: ASC
             }
           }) {
             id
             name
             assetTypeId
             sequence
             duration
             billOfMaterials(options: {
               sort: {
                 sequence: ASC
               }
             }) {
               id
               name
               quantity
               sequence
             }
           }
         }
       }
     }
     '''
```



/ config /

Name	Last Modified
appsettings.json	27 days ago
appsettings.place...	yesterday
assets.json	4 days ago
colors.json	1 hour ago
event-templates.j...	5 days ago
event-types.json	1 hour ago
reference-data-ty...	4 days ago
reference-data.js...	4 days ago

Visualize Raw Material Lots Consumed by Batches

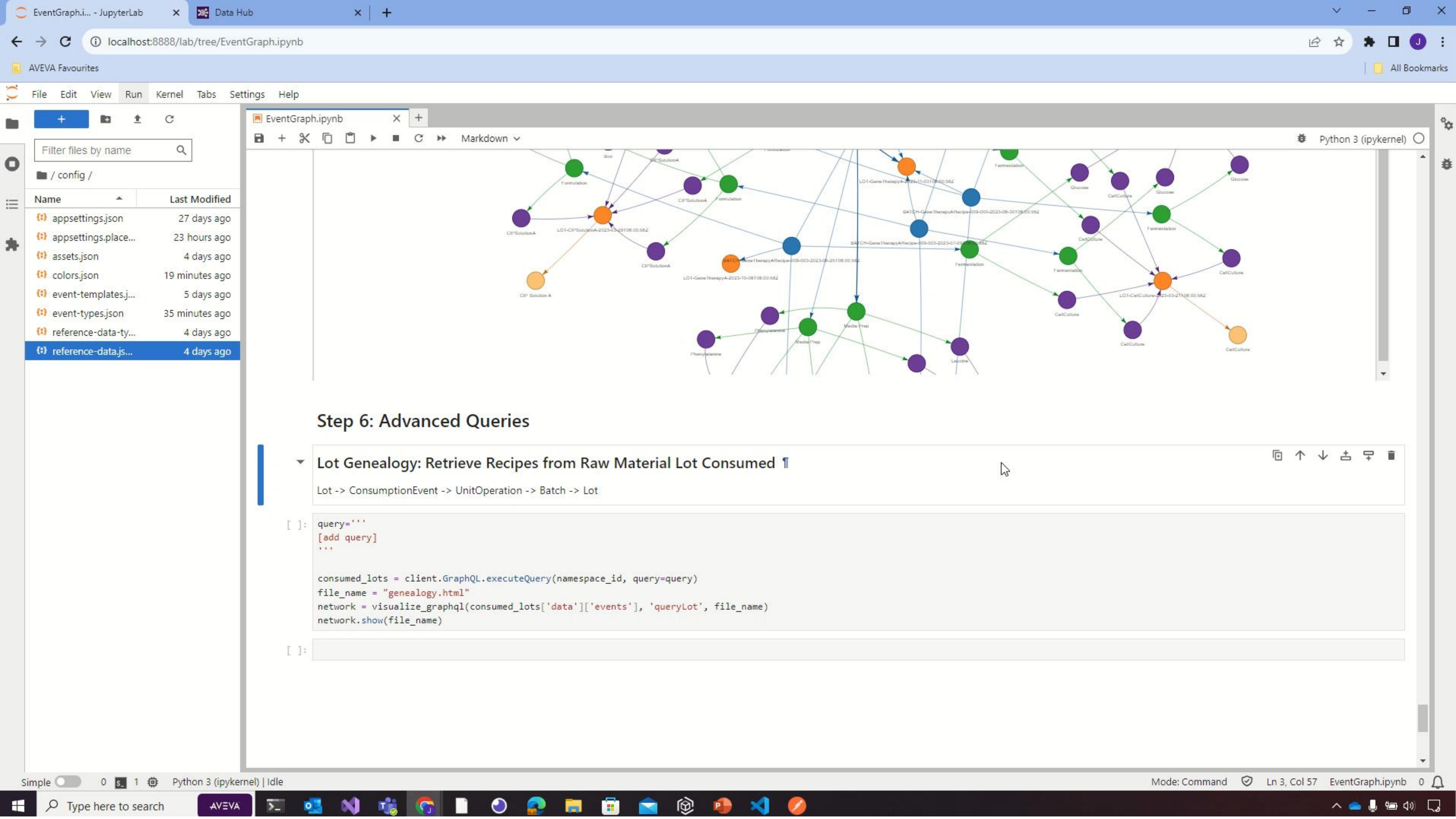
```
[ ]: query='''
[add query]
'''

consumed_lots = client.GraphQL.executeQuery(namespace_id, query=query)
file_name = "consumed_lots.html"
network = visualize_graphql(consumed_lots['data']['events'], 'queryBatch', file_name)
network.show(file_name)
```

Step 6: Advanced Queries ¶

Lot Genealogy: Retrieve Recipes from Raw Material Lot Consumed

Lot -> ConsumptionEvent -> UnitOperation -> Batch -> Lot

```
[128]: query='''
{
  events {
    queryLot(where: {
      number: {
        eq: "LOT-Glucose-2023-05-07T06:00:58Z"
      }
    }) {
      id
      number
      consumptionEvents {
        id
        name
        unitOperationEvent {
          id
          name
          batch {
            id
            number
```


EventGraph.i... - JupyterLab
Data Hub
localhost:8888/lab/tree/EventGraph.ipynb
AVEVA Favourites
All Bookmarks
File Edit View Run Kernel Tabs Settings Help
Filter files by name
/ config /
Name
Last Modified
appsettings.json 27 days ago
appsettings.place... 23 hours ago
assets.json 4 days ago
colors.json 19 minutes ago
event-templates.j... 5 days ago
event-types.json 35 minutes ago
reference-data-ty... 4 days ago
reference-data.js... 4 days ago
EventGraph.ipynb
Markdown
Python 3 (ipykernel)
Simple
Python 3 (ipykernel) | Idle
Mode: Command
Ln 3, Col 57
EventGraph.ipynb
Type here to search

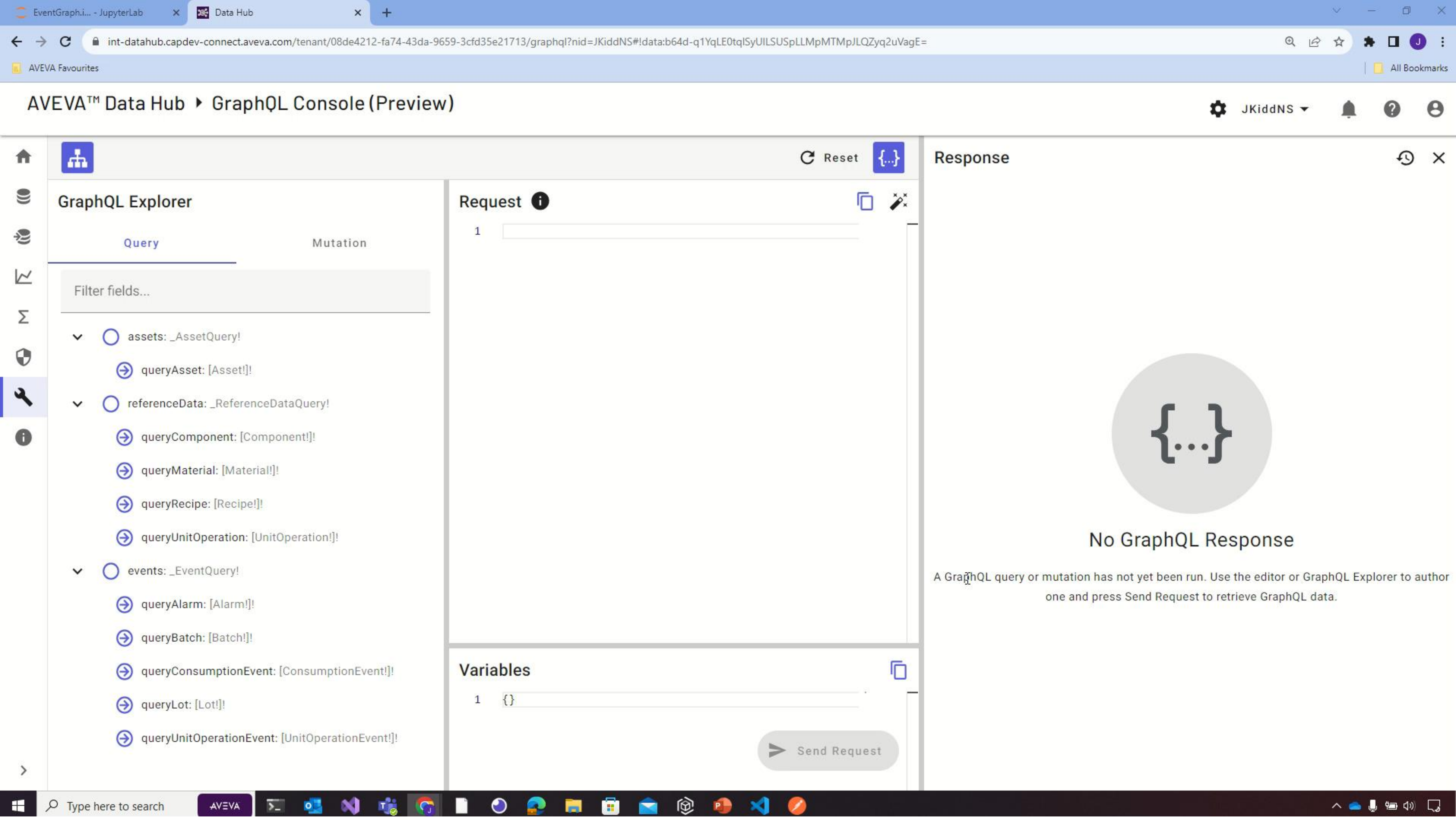
Step 6: Advanced Queries

Lot Genealogy: Retrieve Recipes from Raw Material Lot Consumed ¶

Lot -> ConsumptionEvent -> UnitOperation -> Batch -> Lot

```
[ ]: query='''
     [add query]
     '''

     consumed_lots = client.GraphQL.executeQuery(namespace_id, query=query)
     file_name = "genealogy.html"
     network = visualize_graphql(consumed_lots['data']['events'], 'queryLot', file_name)
     network.show(file_name)
```

```
[ ]:
```


EventGraph.i... - JupyterLab
Data Hub
int-datahub.capdev-connect.aveva.com/tenant/08de4212-fa74-43da-9659-3cfd35e21713/graphql?nid=JKiddNS#!data:b64d-q1YqLE0tqlSyUILSUSpLLMpMTMpJLQZyq2uVagE=
AVEVA Favourites
All Bookmarks
JKiddNS
Reset
Response
Request
Variables
Send Request
Type here to search

AVEVA™ Data Hub ▸ GraphQL Console (Preview)

GraphQL Explorer

Query | Mutation

Filter fields...

- assets: _AssetQuery!
  - queryAsset: [Asset!]!
- referenceData: _ReferenceDataQuery!
  - queryComponent: [Component!]!
  - queryMaterial: [Material!]!
  - queryRecipe: [Recipe!]!
  - queryUnitOperation: [UnitOperation!]!
- events: _EventQuery!
  - queryAlarm: [Alarm!]!
  - queryBatch: [Batch!]!
  - queryConsumptionEvent: [ConsumptionEvent!]!
  - queryLot: [Lot!]!
  - queryUnitOperationEvent: [UnitOperationEvent!]!

```
{}
```

No GraphQL Response

A GraphQL query or mutation has not yet been run. Use the editor or GraphQL Explorer to author one and press Send Request to retrieve GraphQL data.

New data types and query patterns in AVEVA Data Hub

Event Data Store

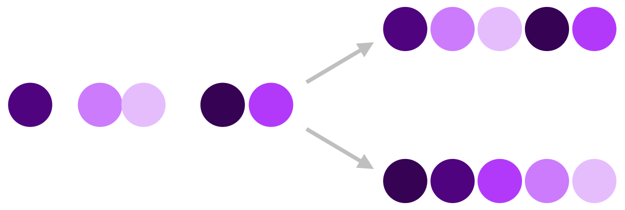
Ability to store event data and provide a rich contextual search API for retrieving the information

Data Views supports Parquet

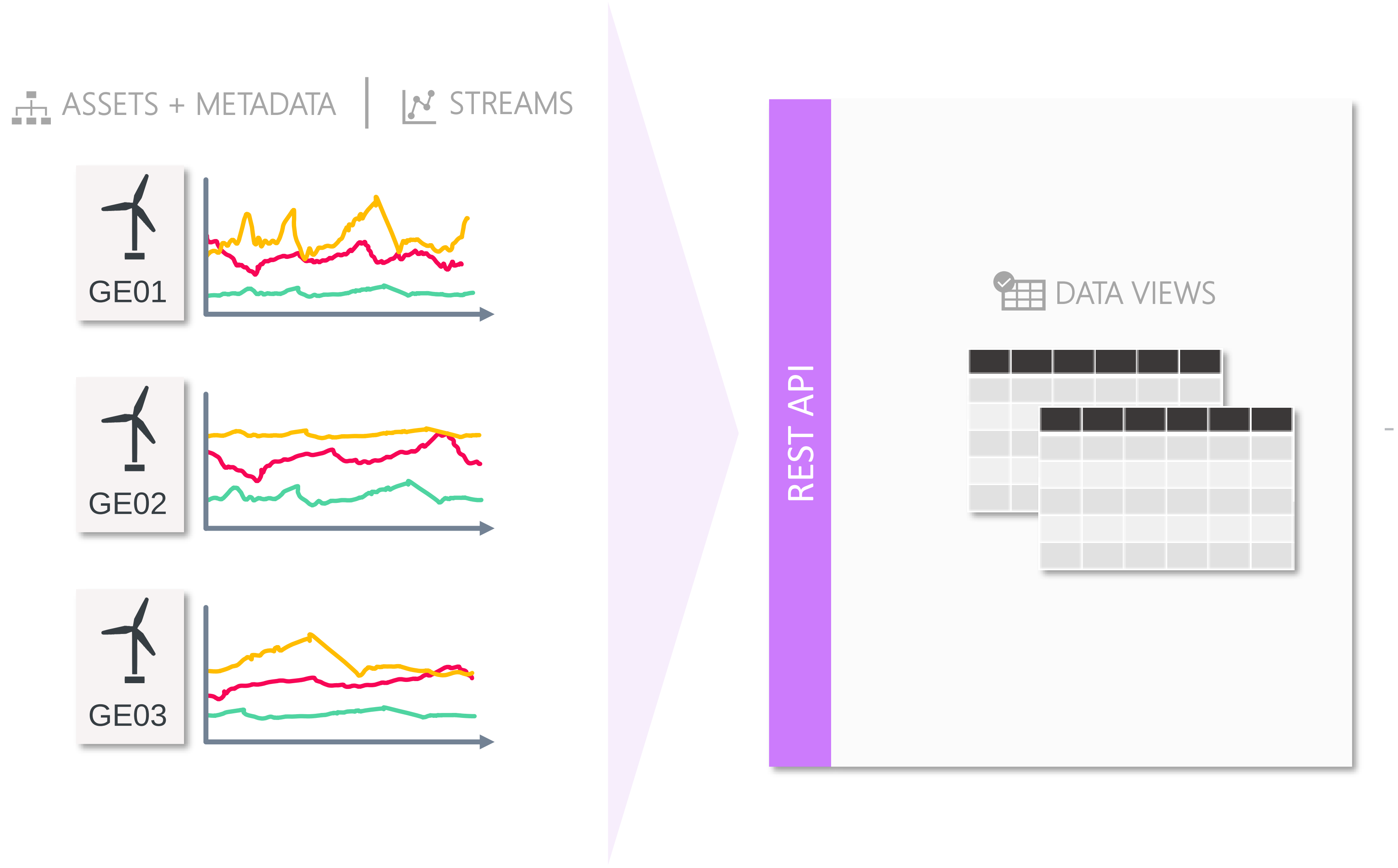
Ability to query data sets in a format familiar and common to data scientists

Change Broker

Ability to sign up for and query changes to stream data



Data Views supports a variety of formats



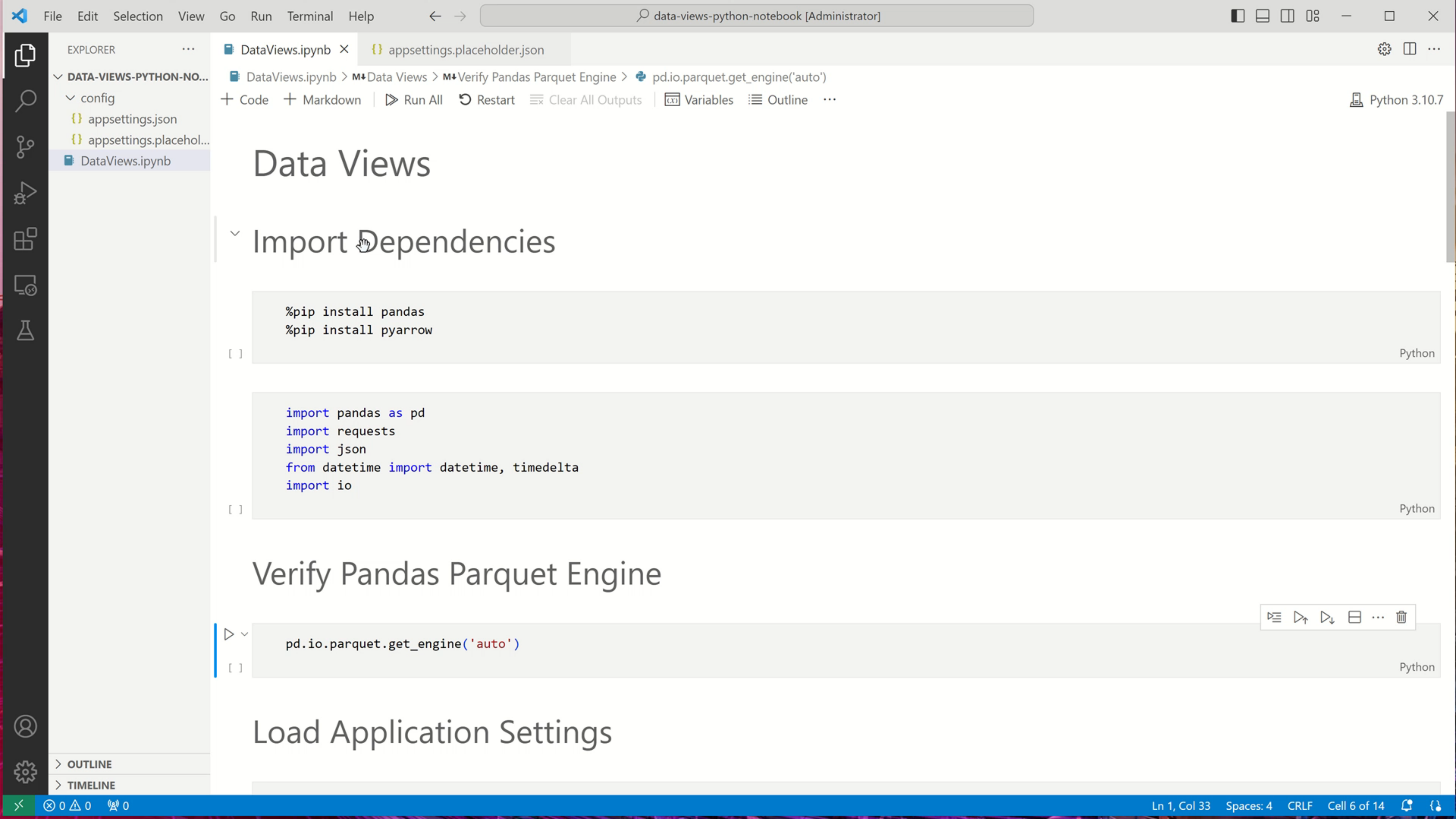
- Object-style JSON
- Table-style JSON
- CSV
- Apache Parquet (new!)

AVEVA

Demo:

DataViews Apache Parquet Format

AVEVA


File Edit Selection View Go Run Terminal Help
data-views-python-notebook [Administrator]
EXPLORER
DATA-VIEWS-PYTHON-NO...
config
appsettings.json
appsettings.placehol...
DataViews.ipynb
OUTLINE
TIMELINE
DataViews.ipynb
appsettings.placeholder.json
DataViews.ipynb > Data Views > Verify Pandas Parquet Engine > pd.io.parquet.get_engine('auto')
Code Markdown Run All Restart Clear All Outputs Variables Outline
Python 3.10.7
Ln 1, Col 33 Spaces: 4 CRLF Cell 6 of 14

Data Views

Import Dependencies

```
%pip install pandas
%pip install pyarrow
```

```
import pandas as pd
import requests
import json
from datetime import datetime, timedelta
import io
```

Verify Pandas Parquet Engine

```
pd.io.parquet.get_engine('auto')
```

Load Application Settings

New data types and query patterns in AVEVA Data Hub

Event Data Store

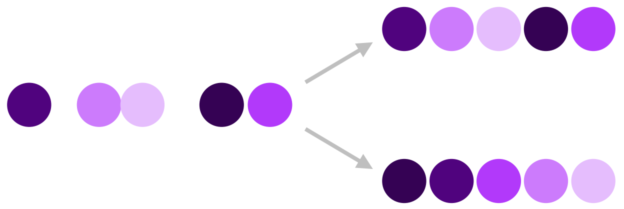
Ability to store event data and provide a rich contextual search API for retrieving the information

Data Views supports Parquet

Ability to query data sets in a format familiar and common to data scientists

Change Broker

Ability to sign up for and query changes to stream data



Industrial operations data is always changing…

Live data

Uploads of manual measurements

Recalculations

Edits of incorrect readings

… and there are consequences downstream

Less accurate predictions and remote monitoring

Inaccurate combined data sets for BI reporting

Service provider recommendations based on incomplete data

AVEVA

Change data can impair and complicate solutions

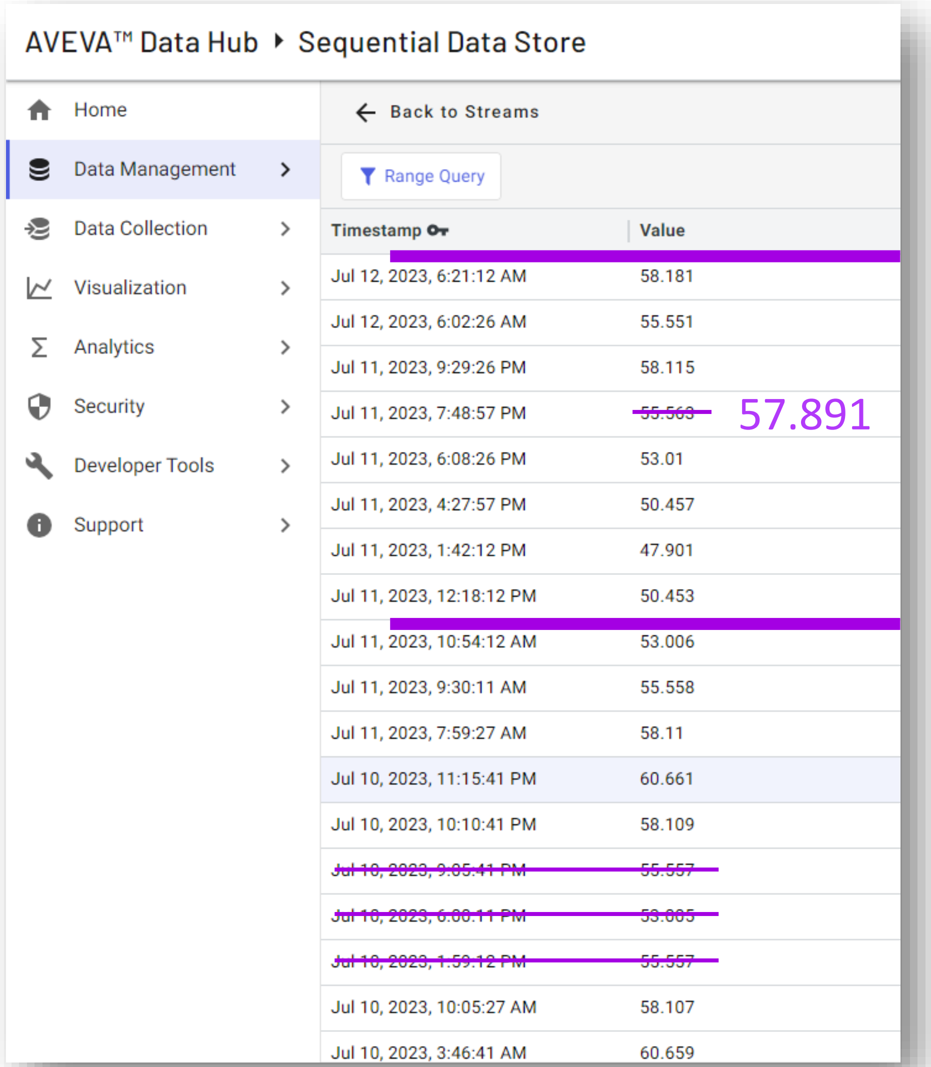
Customers and Partners surfaced common challenges

- **Reconciling destination with source** using large queries
- Taking on risk by **assuming no changes** in data
- **Managing unique data sharing solutions** for each trusted partner

AVEVA

Changes in Streams’ data can be queried in AVEVA Data Hub

(In Preview)



AVEVA™ Data Hub ▸ Sequential Data Store

Timestamp	Value
Jul 12, 2023, 6:21:12 AM	58.181
Jul 12, 2023, 6:02:26 AM	55.551
Jul 11, 2023, 9:29:26 PM	58.115
Jul 11, 2023, 7:48:57 PM	~~55.563~~ 57.891
Jul 11, 2023, 6:08:26 PM	53.01
Jul 11, 2023, 4:27:57 PM	50.457
Jul 11, 2023, 1:42:12 PM	47.901
Jul 11, 2023, 12:18:12 PM	50.453
Jul 11, 2023, 10:54:12 AM	53.006
Jul 11, 2023, 9:30:11 AM	55.558
Jul 11, 2023, 7:59:27 AM	58.11
Jul 10, 2023, 11:15:41 PM	60.661
Jul 10, 2023, 10:10:41 PM	58.109
~~Jul 10, 2023, 9:05:41 PM~~	~~55.557~~
~~Jul 10, 2023, 6:00:11 PM~~	~~53.005~~
~~Jul 10, 2023, 1:59:12 PM~~	~~55.557~~
Jul 10, 2023, 10:05:27 AM	58.107
Jul 10, 2023, 3:46:41 AM	60.659

Update

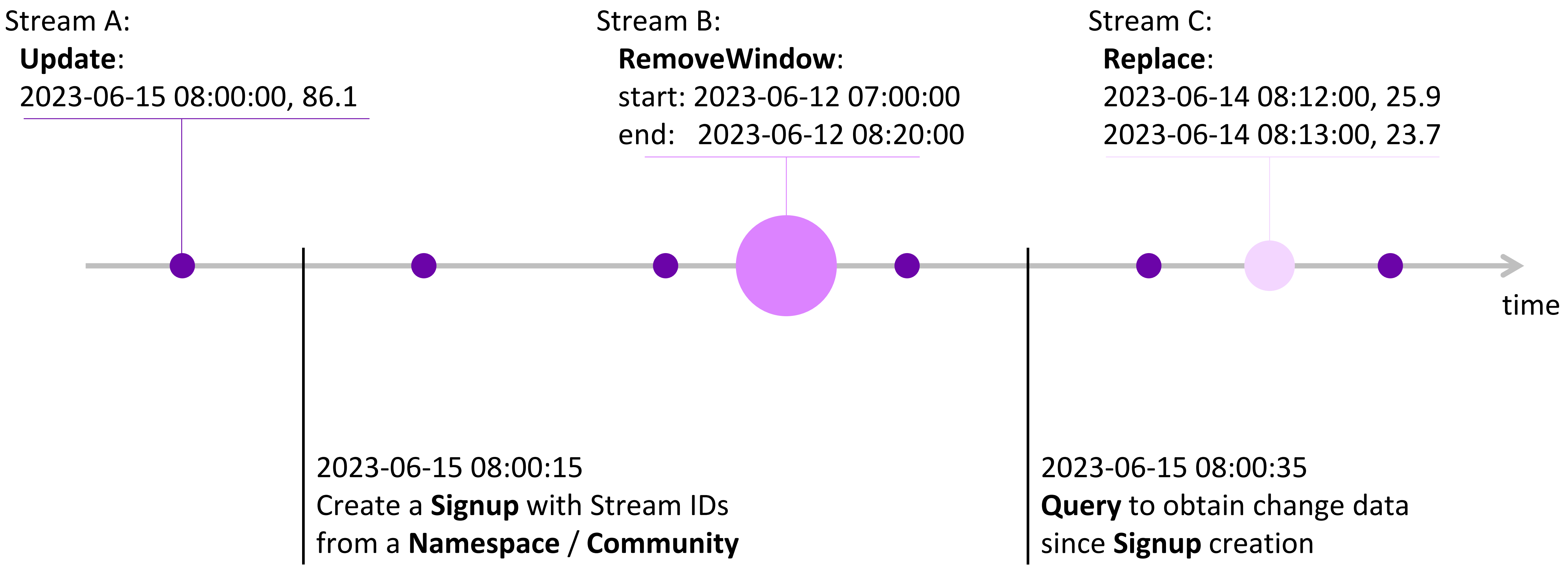
Replace

Insert

Remove

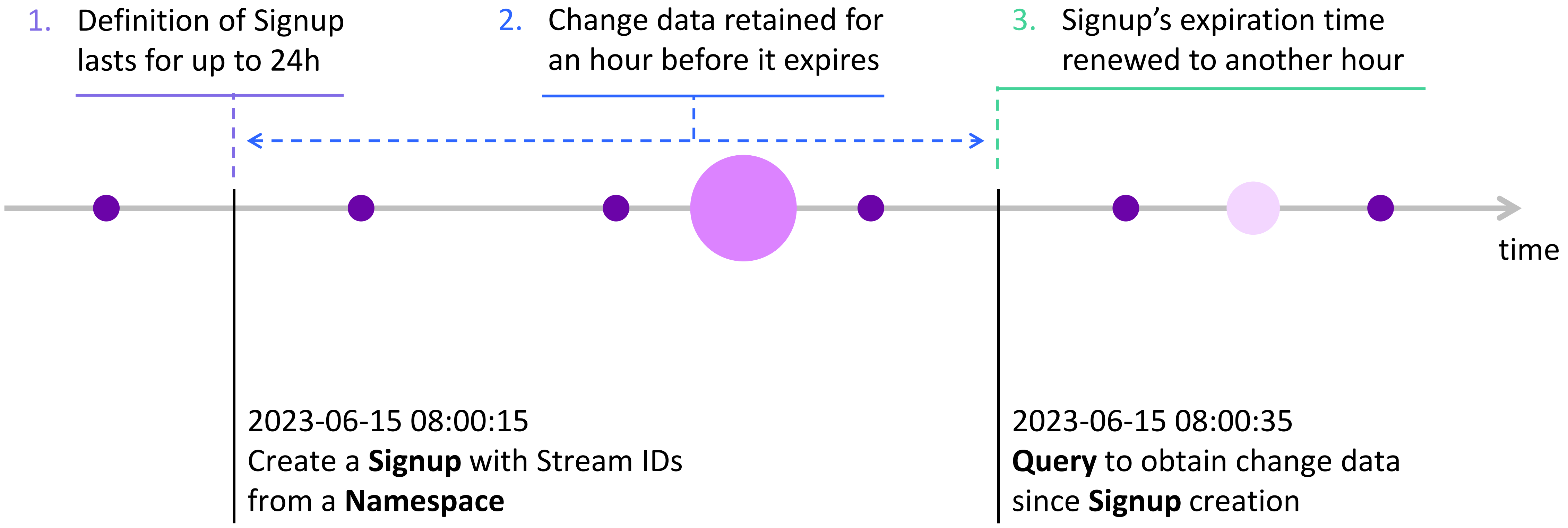
Use an API route to create Signups for change data on Streams

(In Preview)



Keep up with your data as it arrives

(In Preview)


1. Definition of Signup lasts for up to 24h
2. Change data retained for an hour before it expires
3. Signup's expiration time renewed to another hour
time
2023-06-15 08:00:15
Create a **Signup** with Stream IDs from a **Namespace**
2023-06-15 08:00:35
Query to obtain change data since **Signup** creation

AVEVA

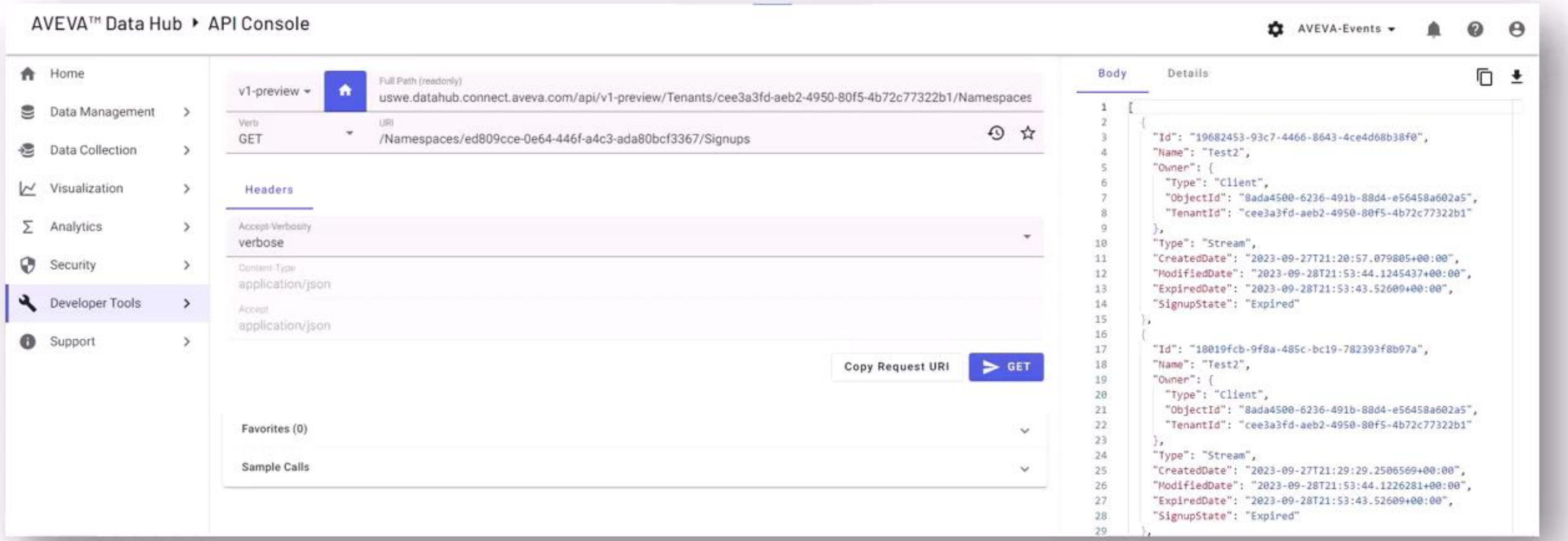
Demo:

Create and query a Signup

AVEVA

AVEVA™ Data Hub - Change Broker Demo

1. Query an Asset to get its Stream IDs
2. Use some of the Asset's Stream IDs to create a Signup
3. Query the Signup for change data
4. Edit the Signup to include all the Stream IDs for the Asset
5. Confirm the Signup's definition is updated



Version: v1-preview

Full Path (readonly): uswe.datahub.connect.aveva.com/api/v1-preview/Tenants/cee3a3fd-aeb2-4950-80f5-4b72c77322b1/Namespaces

Verb: GET

URI: /Namespaces/ed809cce-0e64-446f-a4c3-ada80bcf3367/Signups

Accept-Verbosity: verbose

Content-Type: application/json

Accept: application/json

```
[
  {
    "Id": "19682453-93c7-4466-8643-4ce4d68b38f0",
    "Name": "Test2",
    "Owner": {
      "Type": "Client",
      "ObjectId": "8ada4500-6236-491b-88d4-e56458a602a5",
      "TenantId": "cee3a3fd-aeb2-4950-80f5-4b72c77322b1"
    },
    "Type": "Stream",
    "CreatedDate": "2023-09-27T21:20:57.079805+00:00",
    "ModifiedDate": "2023-09-28T21:53:44.1245437+00:00",
    "ExpiredDate": "2023-09-28T21:53:43.52609+00:00",
    "SignupState": "Expired"
  },
  {
    "Id": "18019fcb-9f8a-485c-bc19-782393f8b97a",
    "Name": "Test2",
    "Owner": {
      "Type": "Client",
      "ObjectId": "8ada4500-6236-491b-88d4-e56458a602a5",
      "TenantId": "cee3a3fd-aeb2-4950-80f5-4b72c77322b1"
    },
    "Type": "Stream",
    "CreatedDate": "2023-09-27T21:29:29.2506569+00:00",
    "ModifiedDate": "2023-09-28T21:53:44.1226281+00:00",
    "ExpiredDate": "2023-09-28T21:53:43.52609+00:00",
    "SignupState": "Expired"
  },
```

AVEVA

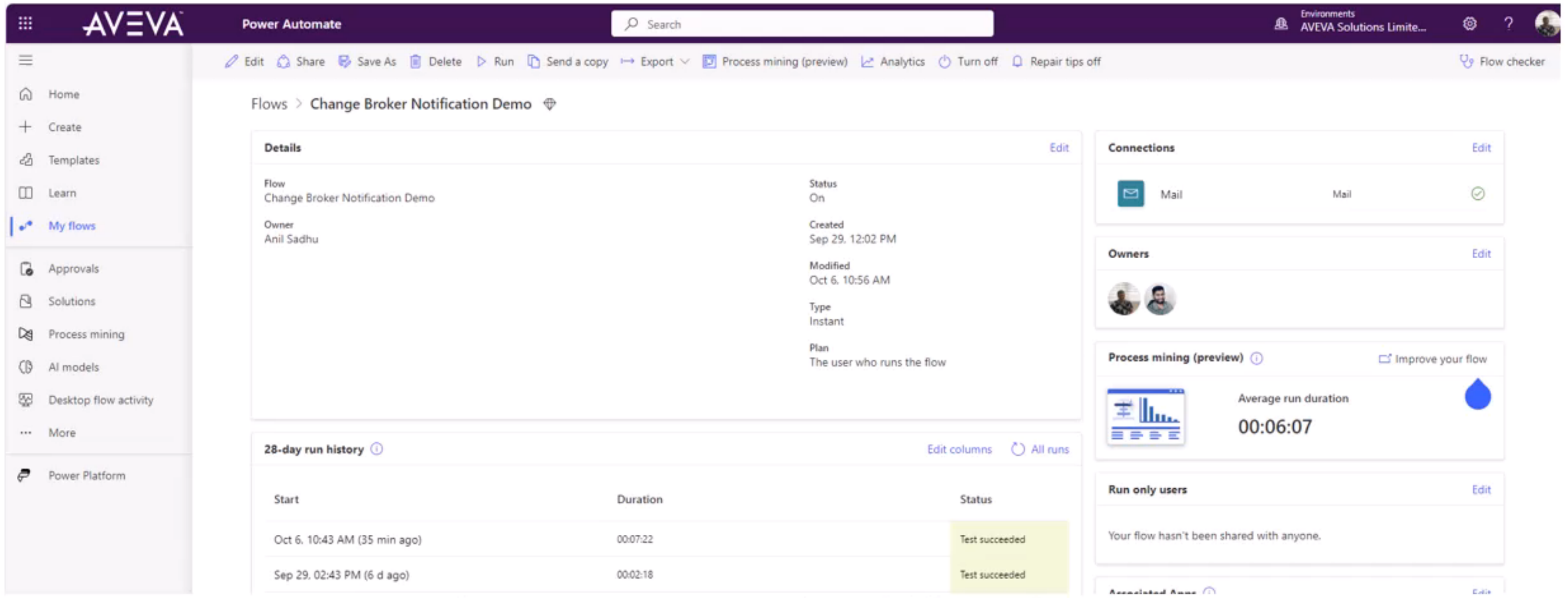
Demo:

Feed an External App, Algorithm, Notice

AVEVA

AVEVA™ Data Hub - Change Broker Demo 2

1. Configure your External App to create a Signup
2. Query the Signup for change data
3. Configure Algorithms or threshold Alerts or Notifications



Flows > Change Broker Notification Demo

Details Edit

Flow
Change Broker Notification Demo

Owner
Anil Sadhu

Status
On

Created
Sep 29, 12:02 PM

Modified
Oct 6, 10:56 AM

Type
Instant

Plan
The user who runs the flow

Connections Edit

Mail Mail

Owners Edit

Process mining (preview) Improve your flow

Average run duration
00:06:07

Run only users Edit

Your flow hasn't been shared with anyone.

28-day run history Edit columns All runs

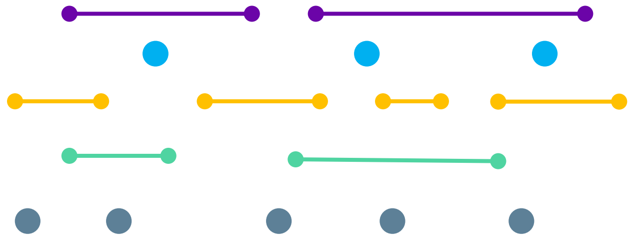
Start	Duration	Status
Oct 6, 10:43 AM (35 min ago)	00:07:22	Test succeeded
Sep 29, 02:43 PM (6 d ago)	00:02:18	Test succeeded

New data types and query patterns in AVEVA Data Hub

Event Data Store

Ability to store event data and provide a rich contextual search API for retrieving the information

In Preview!



Data Views supports Parquet

Ability to query data sets in a format familiar and common to data scientists

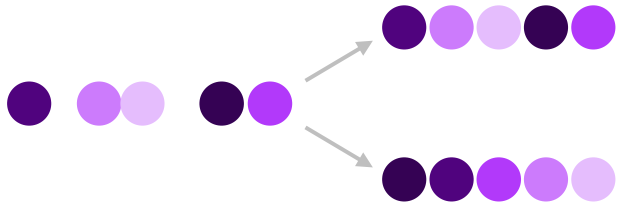
Released!

Change Broker

Ability to sign up for and query changes to stream data

In Preview!

Dec Release



Enabling developers on AVEVA Data Hub

- Update and enhance API Console to support new capabilities
- Release sample code to simplify getting started in various languages
- Support common data formats that integrate into a broader technology ecosystem

AVEVA Lighthouse Program

Join us in a Lighthouse Project to prove & shape the value of new technology for your business scenarios!

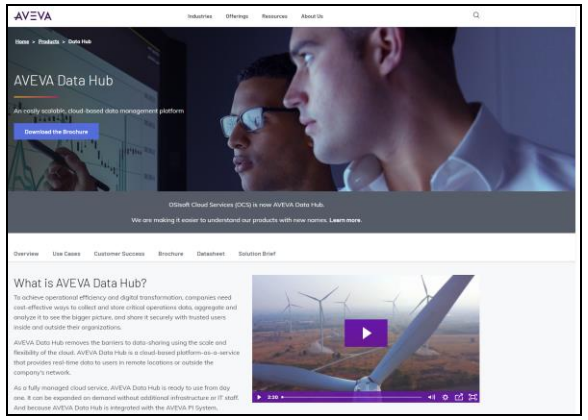
- **AVEVA provides:**
  - AVEVA pre-released software
  - Installation & configuration support
  - Technical R&D support
  - Program management
- **Participating customer provides:**
  - Viable scenario
  - Resources to use AVEVA pre-released software for your scenario
  - Product feedback on use of software for scenario
  - Willingness to document a success story and participate in a future AVEVA public presentation

Email us at lighthouse@aveva.com to engage with our team!

AVEVA

Where to find more information

Overview & Resources



AVEVA Industries Offerings Resources About Us

Home > Products > Data Hub

AVEVA Data Hub

An easily scalable, cloud-based data management platform

Download the Brochure

OSIsoft Cloud Services (OCS) is now AVEVA Data Hub.

We are making it easier to understand our products with new names. **Learn more.**

Overview Use Cases Customer Success Brochure Datasheet Solution Brief

What is AVEVA Data Hub?

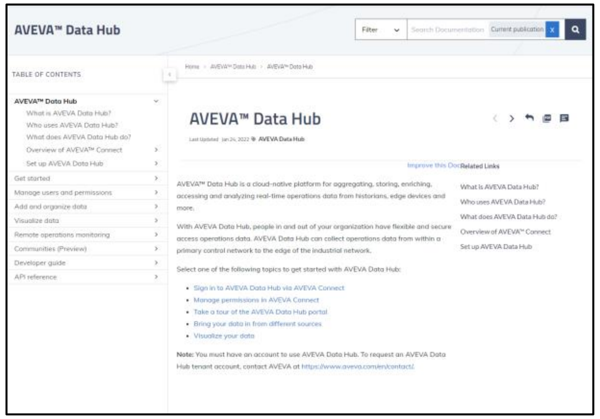
To achieve operational efficiency and digital transformation, companies need cost-effective ways to collect and store critical operations data, aggregate and analyze it to see the bigger picture, and share it securely with trusted users inside and outside their organizations.

AVEVA Data Hub removes the barriers to data-sharing using the scale and flexibility of the cloud. AVEVA Data Hub is a cloud-based platform-as-a-service that provides real-time data to users in remote locations or outside the company's network.

As a fully managed cloud service, AVEVA Data Hub is ready to use from day one. It can be expanded on demand without additional infrastructure or IT staff. And because AVEVA Data Hub is integrated with the AVEVA PI System,

https://www.aveva.com/en/products/data-hub/

Documentation



AVEVA™ Data Hub

Filter | Search Documentation | Current publication

TABLE OF CONTENTS

- AVEVA™ Data Hub
  - What is AVEVA Data Hub?
  - Who uses AVEVA Data Hub?
  - What does AVEVA Data Hub do?
  - Overview of AVEVA™ Connect
  - Set up AVEVA Data Hub
- Get started
- Manage users and permissions
- Add and organize data
- Visualize data
- Remote operations monitoring
- Communities (Preview)
- Developer guide
- API reference

Home > AVEVA™ Data Hub > AVEVA™ Data Hub

AVEVA™ Data Hub

Last Updated Jan 24, 2022 AVEVA Data Hub

Improve this Doc

Related Links

- What is AVEVA Data Hub?
- Who uses AVEVA Data Hub?
- What does AVEVA Data Hub do?
- Overview of AVEVA™ Connect
- Set up AVEVA Data Hub

AVEVA™ Data Hub is a cloud-native platform for aggregating, storing, enriching, accessing and analyzing real-time operations data from historians, edge devices and more.

With AVEVA Data Hub, people in and out of your organization have flexible and secure access operations data. AVEVA Data Hub can collect operations data from within a primary control network to the edge of the industrial network.

Select one of the following topics to get started with AVEVA Data Hub:

- Sign in to AVEVA Data Hub via AVEVA Connect
- Manage permissions in AVEVA Connect
- Take a tour of the AVEVA Data Hub portal
- Bring your data in from different sources
- Visualize your data

Note: You must have an account to use AVEVA Data Hub. To request an AVEVA Data Hub tenant account, contact AVEVA at https://www.aveva.com/en/contact/.

https://docs.aveva.com/bundle/data-hub/page/adh-content-portal-overview.html

Joshua Kidd

Software Architect

- AVEVA Solutions Limited
- joshua.kidd@aveva.com

Scott Dunham

Staff Technical Product Manager

- AVEVA Solutions Limited
- scott.dunham@aveva.com

AVEVA

Questions?

Please wait for the microphone.

State your name and company.

Please remember to...

Navigate to this session in the mobile app to complete the survey.

Thank you!

AVEVA

AVEVA

linkedin.com/company/aveva

@avevagroup

ABOUT AVEVA

AVEVA is a world leader in industrial software, providing engineering and operational solutions across multiple industries, including oil and gas, chemical, pharmaceutical, power and utilities, marine, renewables, and food and beverage. Our agnostic and open architecture helps organizations design, build, operate, maintain and optimize the complete lifecycle of complex industrial assets, from production plants and offshore platforms to manufactured consumer goods.

Over 20,000 enterprises in over 100 countries rely on AVEVA to help them deliver life's essentials: safe and reliable energy, food, medicines, infrastructure and more. By connecting people with trusted information and AI-enriched insights, AVEVA enables teams to engineer efficiently and optimize operations, driving growth and sustainability.

Named as one of the world's most innovative companies, AVEVA supports customers with open solutions and the expertise of more than 6,400 employees, 5,000 partners and 5,700 certified developers. The company is headquartered in Cambridge, UK.

Learn more at www.aveva.com

AVEVA