



# Building Real-Time Pipelines with FLaNK: A Case Study with Transit Data

Tim Spann  
Principal Developer Advocate

July-2023

# Building Real-time Pipelines with FLaNK: A Case Study with Transit Data

In this session, we will explore the powerful combination of Apache Flink, Apache NiFi, and Apache Kafka for building real-time data processing pipelines. We will present a case study using the FLaNK-MTA project, which leverages these technologies to process and analyze real-time data from the New York City Metropolitan Transportation Authority (MTA). By integrating Flink, NiFi, and Kafka, FLaNK-MTA demonstrates how to efficiently collect, transform, and analyze high-volume data streams, enabling timely insights and decision-making.

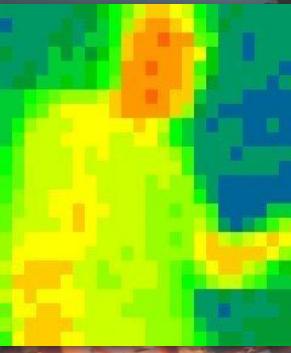
## Takeaways:

Understanding the integration of Apache Flink, Apache NiFi, and Apache Kafka for real-time data processing  
Insights into building scalable and fault-tolerant data processing pipelines  
Best practices for data collection, transformation, and analytics with FLaNK-MTA as a reference  
Knowledge of use cases and potential business impact of real-time data processing pipelines

<https://github.com/tspannhw/FLaNK-MTA/tree/main>

<https://medium.com/@tspann/finding-the-best-way-around-7491c76ca4cb>

# Tim Spann



@PaasDev [www.datainmotion.dev](http://www.datainmotion.dev)  
[github.com/tspannhw](https://github.com/tspannhw) [medium.com/@tspann](https://medium.com/@tspann)  
Principal Developer Advocate



Princeton Future of Data Meetup  
ex-Pivotal, ex-Hortonworks, ex-StreamNative,  
ex-PwC, ex-EY, ex-HPE.



0 53,639 / 153.08 MB 0 0 230 831 546 160 0 0 0 0 0 22:26:28 EDT



CMA CGM

MAERSK

CMA CGM

MAERSK  
SEALAND

MAERSK

HAMBURG SÜD

MAERSK

# OVERVIEW

Q. What is contain it everything?  
A. Wisdom.

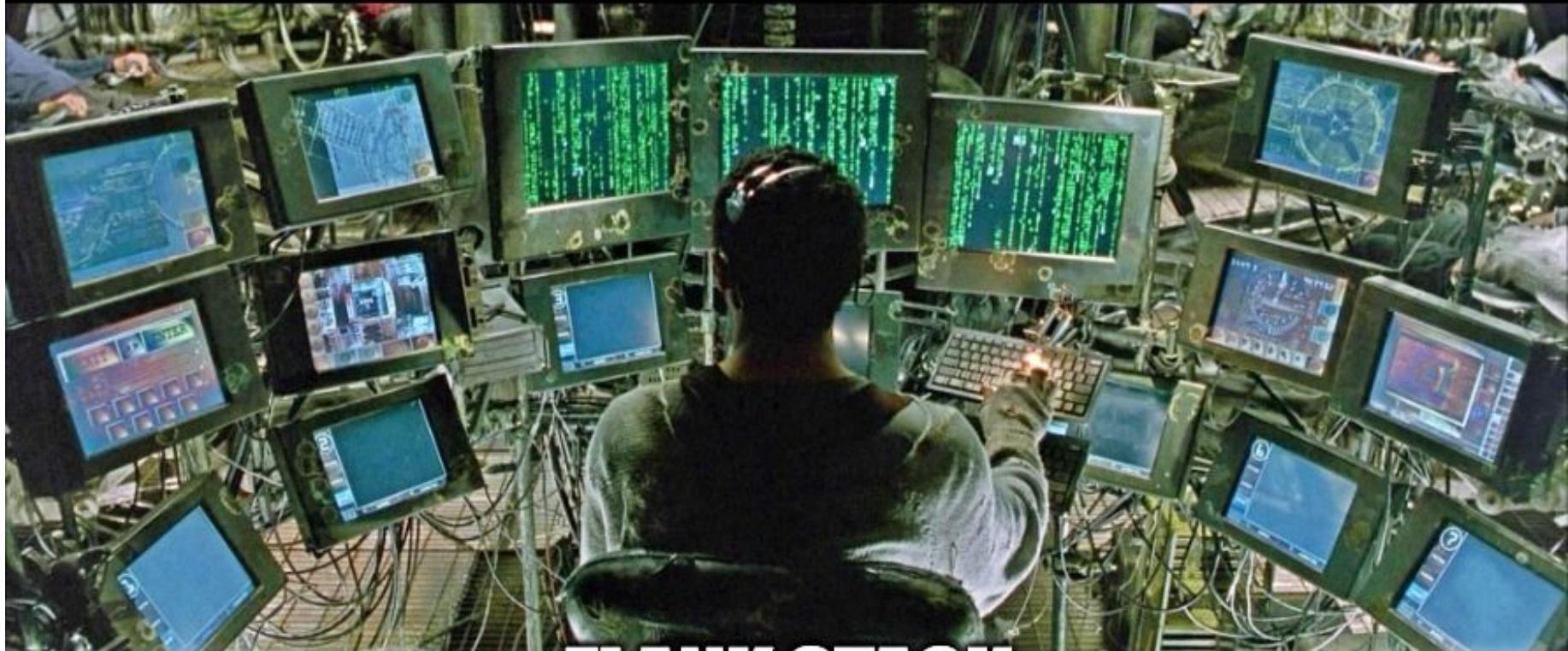
# Trains, Planes and Automobiles +++

Information Needed	Data Feed(s)
Local weather conditions	<ul style="list-style-type: none"><li>• XML, JSON, RSS</li></ul>
Mass transit status & alerts	<ul style="list-style-type: none"><li>• XML, JSON, RSS</li></ul>
Regional highways & tunnels	<ul style="list-style-type: none"><li>• GeoRSS, XML, ProtoBuf, JSON</li></ul>
Local social media	<ul style="list-style-type: none"><li>• JSON</li></ul>
ADS-B Plane Data	<ul style="list-style-type: none"><li>• JSON</li></ul>
Local air quality	<ul style="list-style-type: none"><li>• JSON</li></ul>

# REAL-TIME REQUIRES A PLATFORM



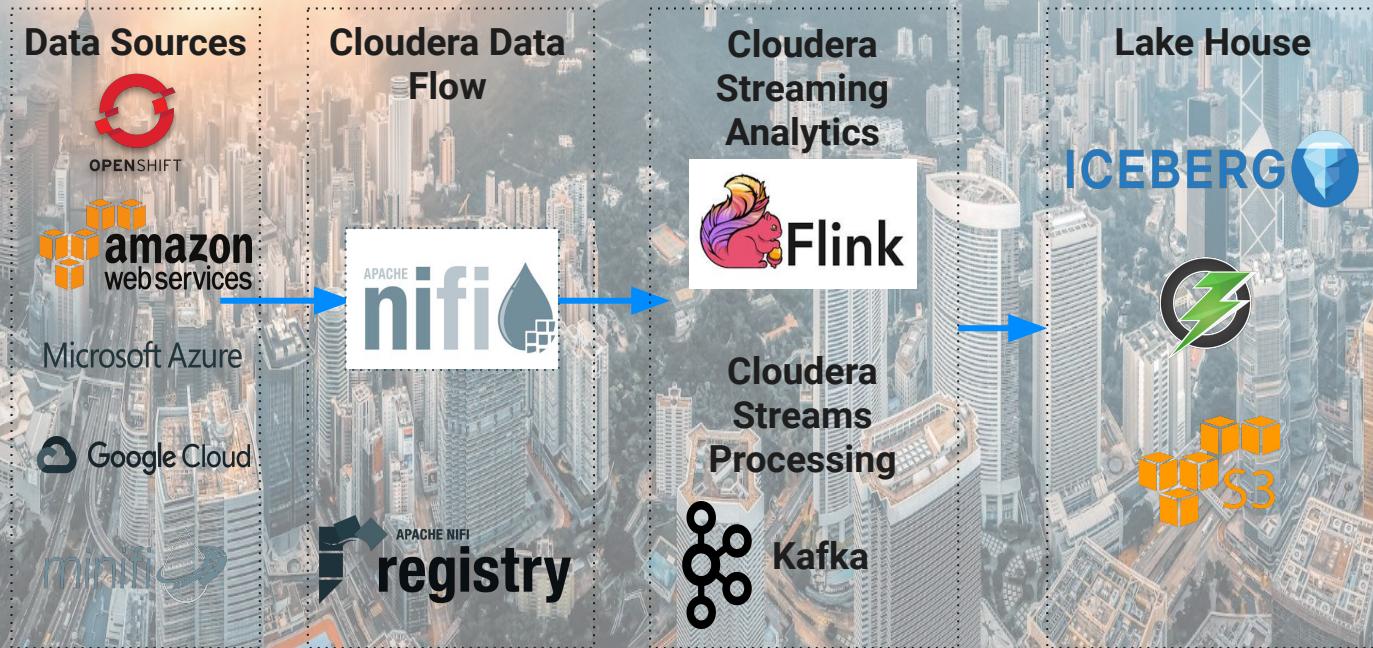
# ALL THE TRANSIT DATA



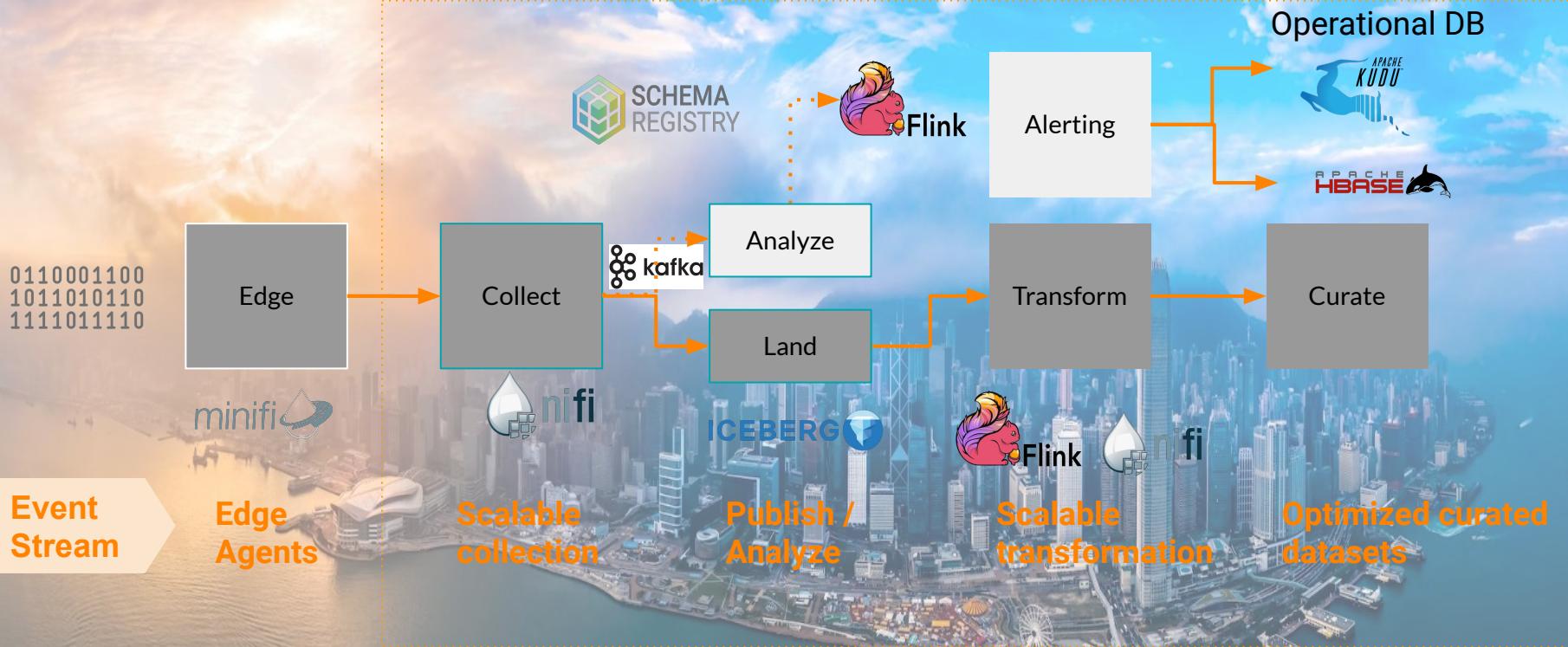
## FLANK STACK

# INGEST OF ALL TRANSIT DATA

Run collection and streaming on any cloud, server, container or VM



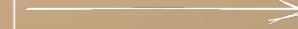
# CLOUDERA DATA PLATFORM

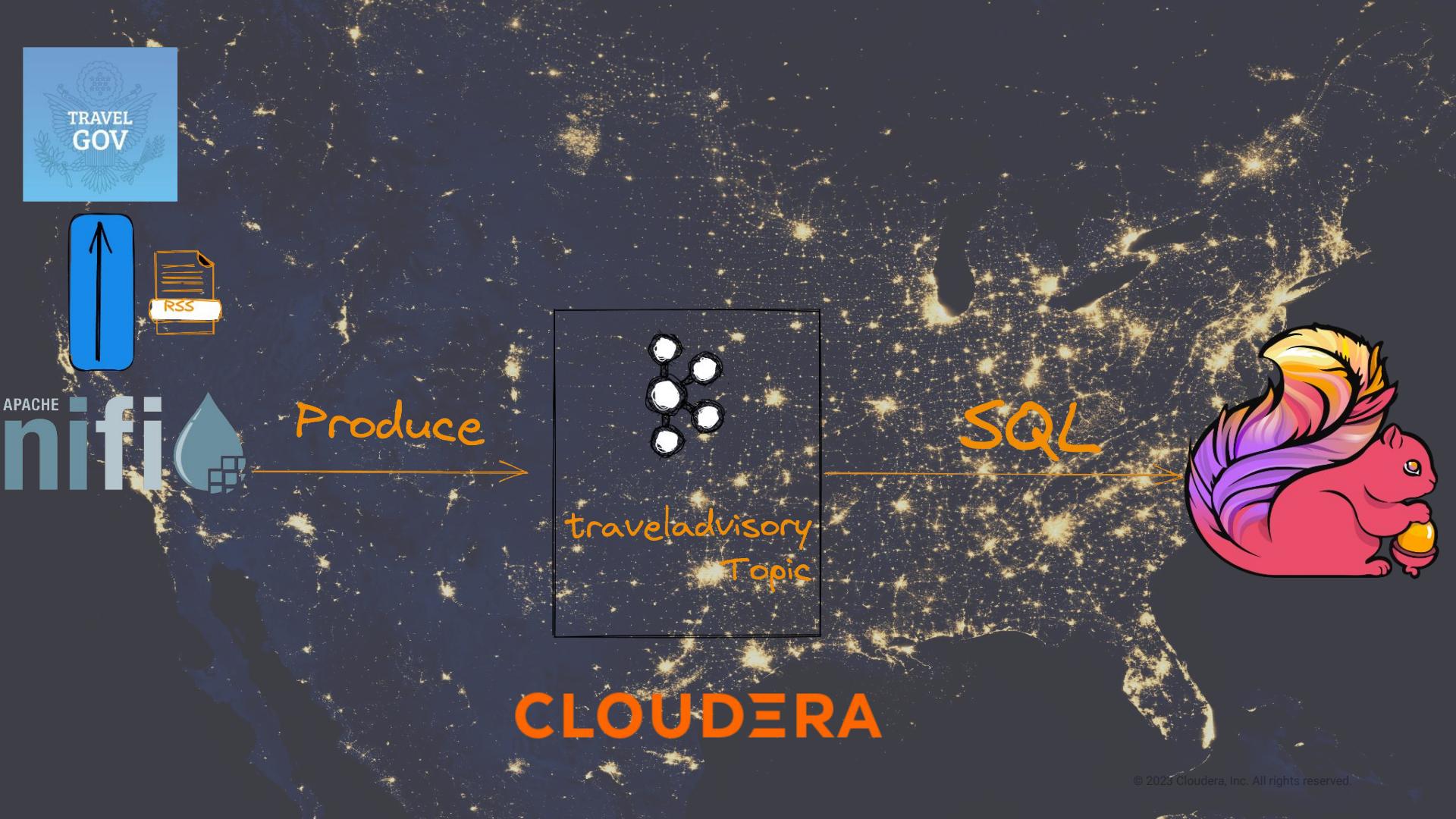


# Metropolitan Transportation Authority



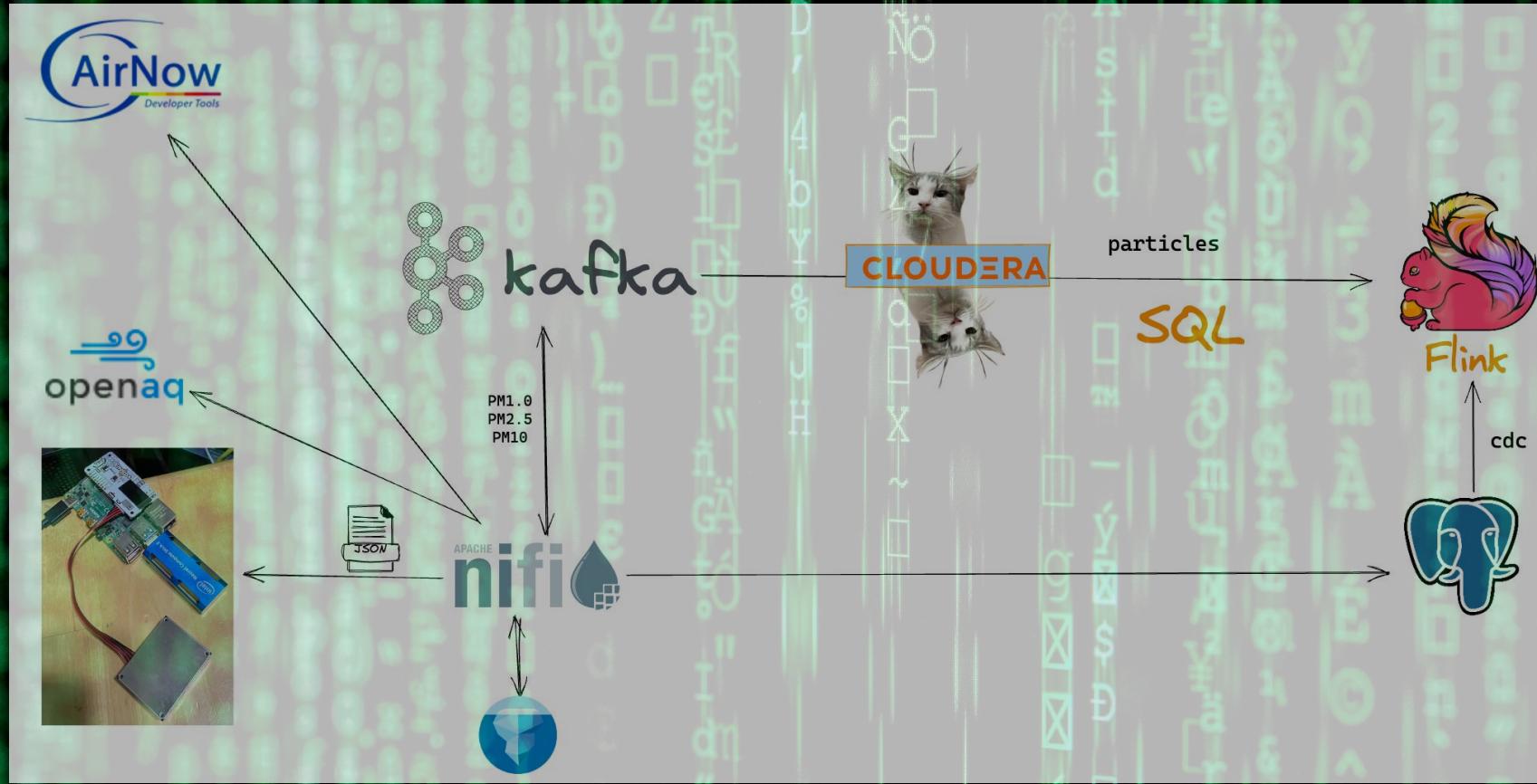
SQL







# Particulate Matter Sensor Pipeline





---

# APACHE NiFi - MiNiFi Agents



**NIFI For Ants**

# Cloudera Edge Management

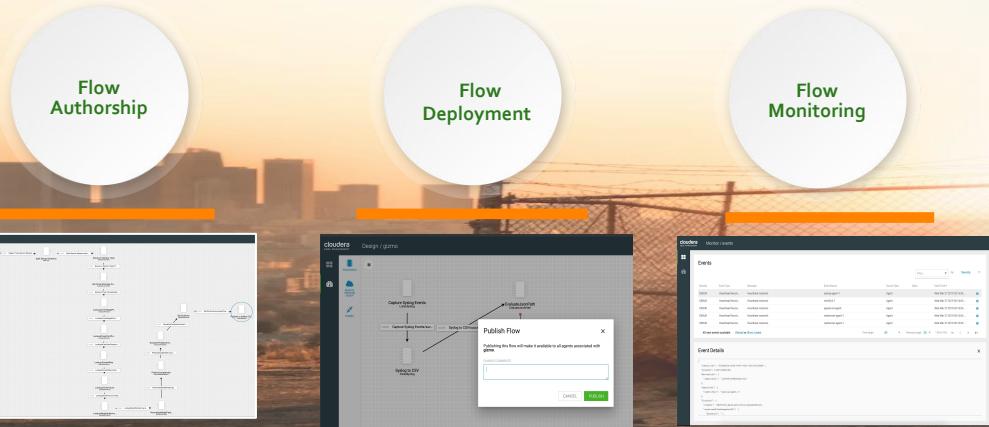
Edge device data collection and processing with easy to use central command and control

Edge Flow Manager

The screenshot shows the Cloudera Edge Flow Manager interface. It features a central workspace where a flow diagram is being designed. The diagram includes various nodes such as 'CloudWatch Metrics', 'Apache NiFi', 'Apache Beam', and 'Apache Flink'. Arrows indicate the flow of data between these components. On the left side of the workspace, there are tabs for 'Design / security cameras' and 'Process'. A modal window titled 'Managed Packages (Process) Configuration' is open, showing settings for 'Processor Name' (set to 'Managed Package'), 'Processor Type' (set to 'Apache Beam'), and 'Processor Version' (set to 'Apache Beam 3.0.0'). Below this, there are sections for 'Scheduling' and 'Properties'. The 'Properties' section contains a single entry: 'Hardcoding XML Definition Path' set to 'Apache Beam Operator 4.0.1 schema/specification.xml'. The background of the slide features a sunset over a city skyline.



A lightweight edge agent that implements the core features of Apache NiFi, focusing on data collection and processing at the edge



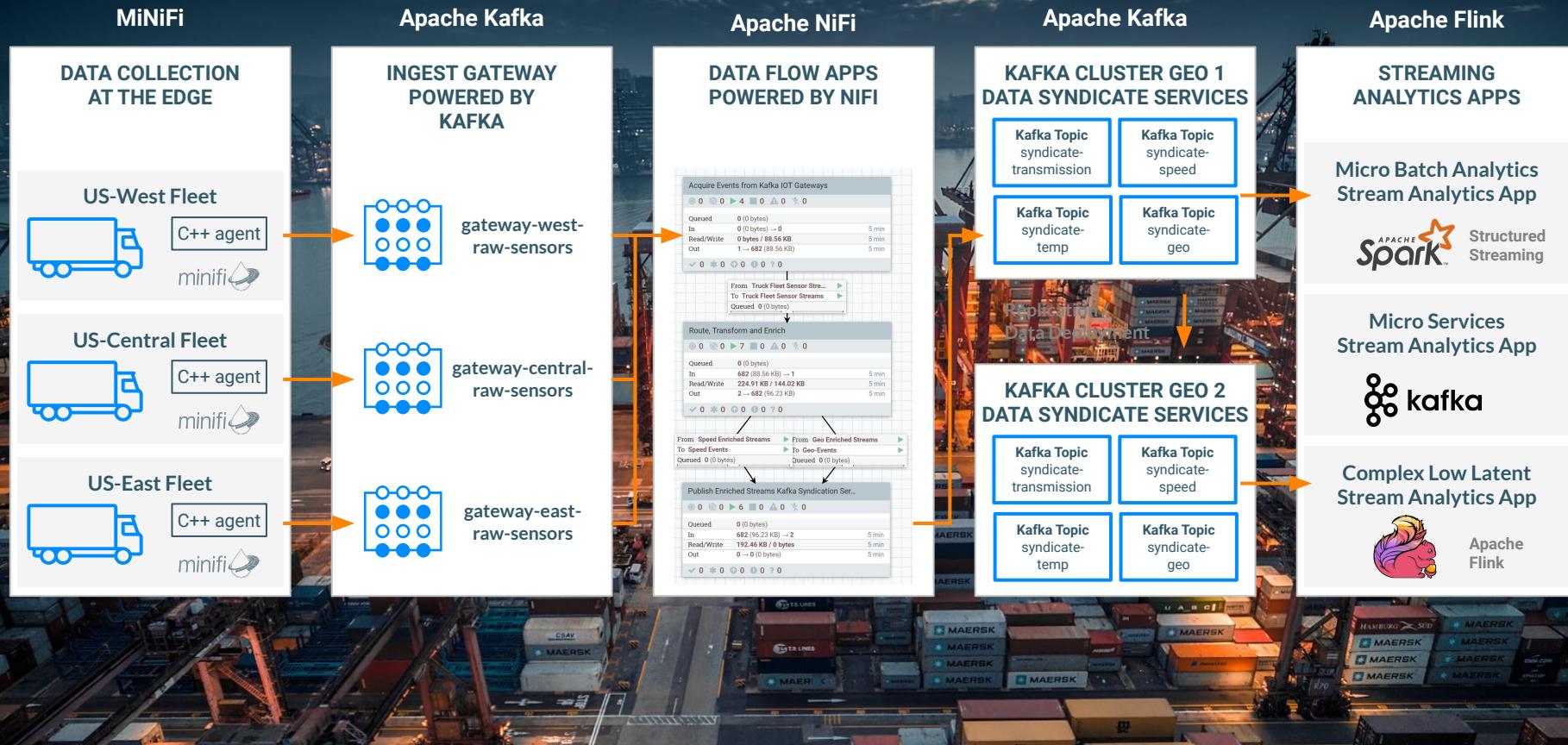
- Small footprint agent with MiNiFi
- Java and C++ agents
- Rich edge processors (edge collection & processing)
- End to end lineage and security
- Kubernetes support
- Central Command and Control (C2)
- Design and deploy to millions of agents
- Edge Applications lifecycle management
- Multitenancy with Agent classes
- Native integration with other CDF services

---

# APACHE KAFKA



# Apache Kafka



---

# APACHE FLINK



# SQL STREAM BUILDER (SSB)

Democratize access to real-time data with just SQL

**SQL STREAM BUILDER allows  
developers, analysts, and data  
scientists to write streaming  
applications with industry  
standard SQL.**

**No Java or Scala code  
development required.**

**Simplifies access to data in Kafka  
& Flink. Connectors to batch data  
in HDFS, Kudu, Hive, S3, JDBC,  
CDC and more**

**Enrich streaming data with batch  
data in a single tool**

The screenshot shows the SQL Stream Builder interface. On the left, the Explorer panel displays a project named 'ssb\_default' with several jobs and virtual tables listed. One job, 'elegant\_babbage', is selected. The Editor panel on the right contains a SQL query for creating a table from a Kafka topic:

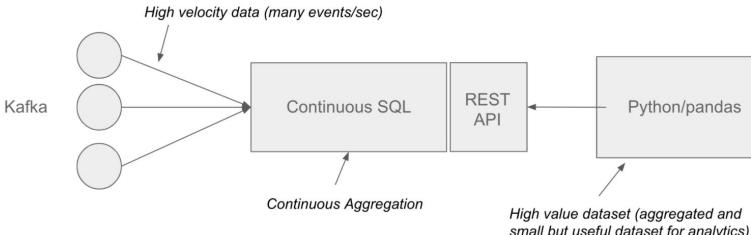
```
1 CREATE TABLE `kafka_table_1670513700` (
2   `col_str` STRING,
3   `col_int` INT,
4   `col_ts` TIMESTAMP(3),
5   WATERMARK FOR `col_ts` AS col_ts - INTERVAL '5' SECOND
6 ) WITH (
7   'connector' = 'kafka', -- Specify what connector to use, for Kafka it must use 'kafka'.
8   'format' = 'json', -- Comma separated list of Kafka brokers.
9   'topic' = 'elegant_babbage', -- Topic name to read data from when the table is used as source. It also supports topic list for source by separating topic by semicolon.
10   'bootstrap.servers' = '...', -- Comma separated list of Kafka brokers.
11   'value.deserializer' = 'org.apache.kafka.common.serialization.StringDeserializer', -- Optional flag to specify whether to encode all decimals as plain numbers instead of floating point representation.
12   'ignore.failed.records.per.seconds' = 'false', -- Optional flag to specify whether to fail if a field is missing or not, false by default.
13   'ignore.parse.errors' = 'false', -- Optional flag to skip fields and rows with parse errors instead of failing; fields are set to null in case of errors, false by default.
14   'map-null-key.literal' = 'null', -- Optional flag to specify string literal for null keys when 'map-null-key.mode' is LITERAL, '\"null\"' by default.
15   'map-null-key.mode' = 'FAIL' -- Optional flag to control the handling mode when serializing null key for map data, FAIL by default.
16 ) -- 'map-null-key.literal' = 'null' -- Optional flag to use 'map-null-key.literal' as key literal.
17 
```

The Log panel at the bottom shows the execution history of the job:

- [08/11/2022, 16:34:55] INFO Active job stopped
- [08/12/2022, 16:35:00] INFO Inserted kafka [json]connector DDL template
- [08/12/2022, 16:35:01] INFO Executing elegant\_babbage
- [08/12/2022, 16:35:02] INFO Creating table: [elegant\_babbage]
- [08/12/2022, 16:35:03] INFO schema: watermark.v0.routinememcol\_ts, schema:watermark.v0.strategy,data-type=TIMESTAMP(3), connector:kafka, schema:watermark.v0.routinememcol\_ts, ignoreIfExists: (false), isTemporary: (false) command executed successfully.
- [08/12/2022, 16:35:07] INFO Action job started
- [08/12/2022, 16:35:12] INFO youthul\_leavitt loaded into editor.
- [08/12/2022, 16:35:13] INFO unruffled\_thompson loaded into editor.
- [08/12/2022, 16:35:17] INFO youthful\_leavitt loaded into editor.
- [08/12/2022, 16:35:18] INFO elegant\_babbage loaded into editor.
- [08/12/2022, 16:35:24] INFO unruffled\_thompson loaded into editor.
- [08/12/2022, 16:35:27] INFO Inserted faker connector DDL template
- [08/12/2022, 16:35:30] INFO unruffled\_thompson loaded into editor.
- [08/12/2022, 16:35:32] INFO elegant\_babbage loaded into editor.

# SSB MATERIALIZED VIEWS

Key Takeaway; MV's allow data scientist, analyst and developers consume data from the firehose



```
SELECT userid,
       max(amount) as max_amount,
       sum(amount) as sum_amount,
       count(*) as thecount,
       tumble_end(eventTimestamp, interval '5' second) as ts
  FROM authorizations
 GROUP BY userid, tumble(eventTimestamp, interval '5' second)
 HAVING count(*) > 1
```

```
[90]: import pandas as pd
[91]: mv = "https://xxxxxxxxxx"
[92]: df = pd.read_json(mv)
[93]: len(df.keys())
[93]: 5
[95]: df['ts'] = pd.to_datetime(df['ts'])
[97]: df.dtypes
[97]: max_amount      int64
sum_amount        int64
thecount         int64
ts                datetime64[ns]
userid           int64
dtype: object
[98]: df.set_index('userid').sort_values(by=['thecount'], ascending=False).head()
[98]:
```

userid	max_amount	sum_amount	thecount	ts
787	34911	57304	10	2020-06-16 19:52:15
744	77407	95407	9	2020-06-16 19:52:15
78	88761	330397	9	2020-06-16 19:52:15
541	78762	282682	8	2020-06-16 19:52:15
926	85636	129728	8	2020-06-16 19:52:15

# Infer Tables from Kafka Topics with JSON or Avro

## Kafka Table

Table Name \*

Kafka Cluster \*

Data Format \*

Topic Name \*

Schema Definition

Event Time

Data Transformation

Properties

Deserialization

```
1  {
2    "type": "record",
3    "name": "inferredSchema",
4    "fields": [
5      {
6        "name": "bssid",
7        "type": "string",
8        "doc": "Type inferred from '\"\\\"\'"
9      },
10     {
11       "name": "channel",
12       "type": "string",
13       "doc": "Type inferred from '\"52\\\"\'"
14     },
15     {
16       "name": "channel_band",
17       "type": "string",
18       "doc": "Type inferred from '\"S\\\"\'"
19     },
20     {
21       "name": "channel_width",
22       "type": "string",
23       "doc": "Type inferred from '\"80\\\"\'"
24     },
25     {
26       "name": "country_code",
27       "type": "string",
28       "doc": "Type inferred from '\"\\\"\'"
29     },
30     {
31       "name": "interface",
32       "type": "string",
33       "doc": "Type inferred from '\"en0\\\"\'"
34     },
35   }
```



Schema is valid

Detect Schema

Cancel

Create and Review



tspann  
LOG OUT

0 53,639 / 153.08 MB 0 0 ▶ 230 ■ 831 ⚠ 546 ✅ 0 \* 0 ⚡ 0 ? 0 22:26:28 EDT



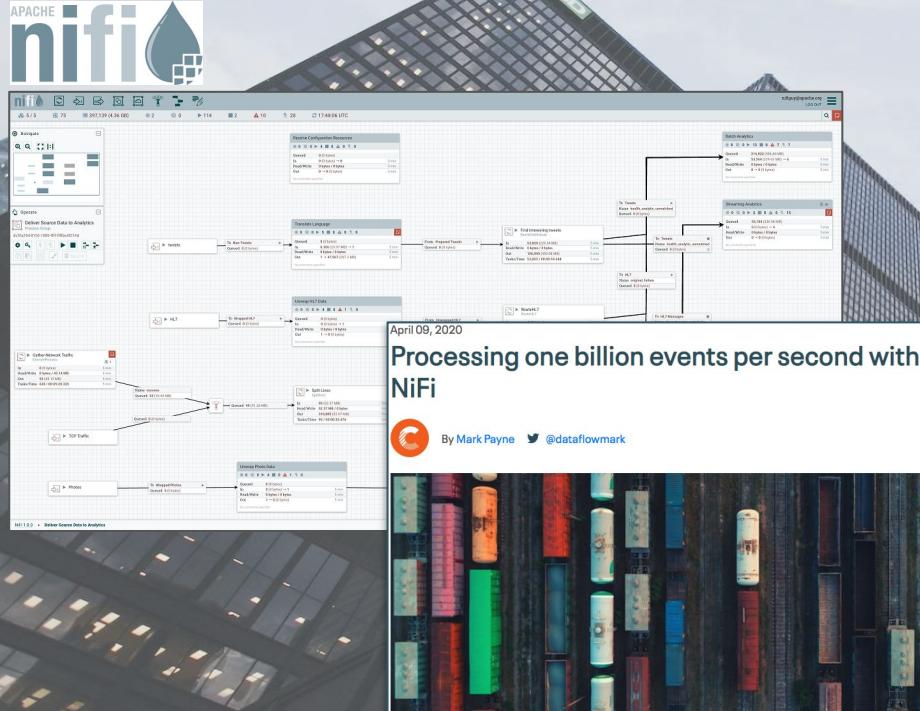
---

# DATAFLOW APACHE NIFI



# CLOUDERA DATAFLOW - POWERED BY APACHE NiFi

Ingest and manage data from edge-to-cloud using a no-code interface



- #1 data ingestion/movement engine
- Strong community
- Product maturity over 11 years
- Deploy on-premises or in the cloud
- Over 400+ pre-built processors
- Built-in data provenance
- Guaranteed delivery
- Throttling and Back pressure

# PROVENANCE

Displaying 13 of 104  
Oldest event available: 11/15/2016 13:34:50 EST

Showing the most recent events.

ConsumeKafka by component name

Date/Time	Type	FlowFile Uuid	Size	Component Name	Component Type
11/15/2016 13:35:03.8...	RECEIVE	379fc4f6-60e0-4151-9743-28...	44 bytes	ConsumeKafka	ConsumeKafka
11/15/2016 13:35:02.7...	RECEIVE	78f8c38b-89fc-4d00-a8d8-51...	44 bytes	ConsumeKafka	ConsumeKafka
11/15/2016 13:35:01.6...	RECEIVE	2bcd5124-bb78-489f-ad8a-7...	44 bytes	ConsumeKafka	ConsumeKafka

• Tracks data at each point as it flows through the system

• Records, indexes, and makes events available for display

• Handles fan-in/fan-out, i.e. merging and splitting data

• View attributes and content at given points in time

The diagram illustrates a data flow process. It starts with a red circle labeled "RECEIVE", which has an arrow pointing down to a grey circle labeled "JOIN". From the "JOIN" circle, an arrow points down to a blue circle labeled "DROP". Two green arrows originate from the "RECEIVE" and "JOIN" circles and point to a separate "Provenance Event" panel on the right.

**Provenance Event**

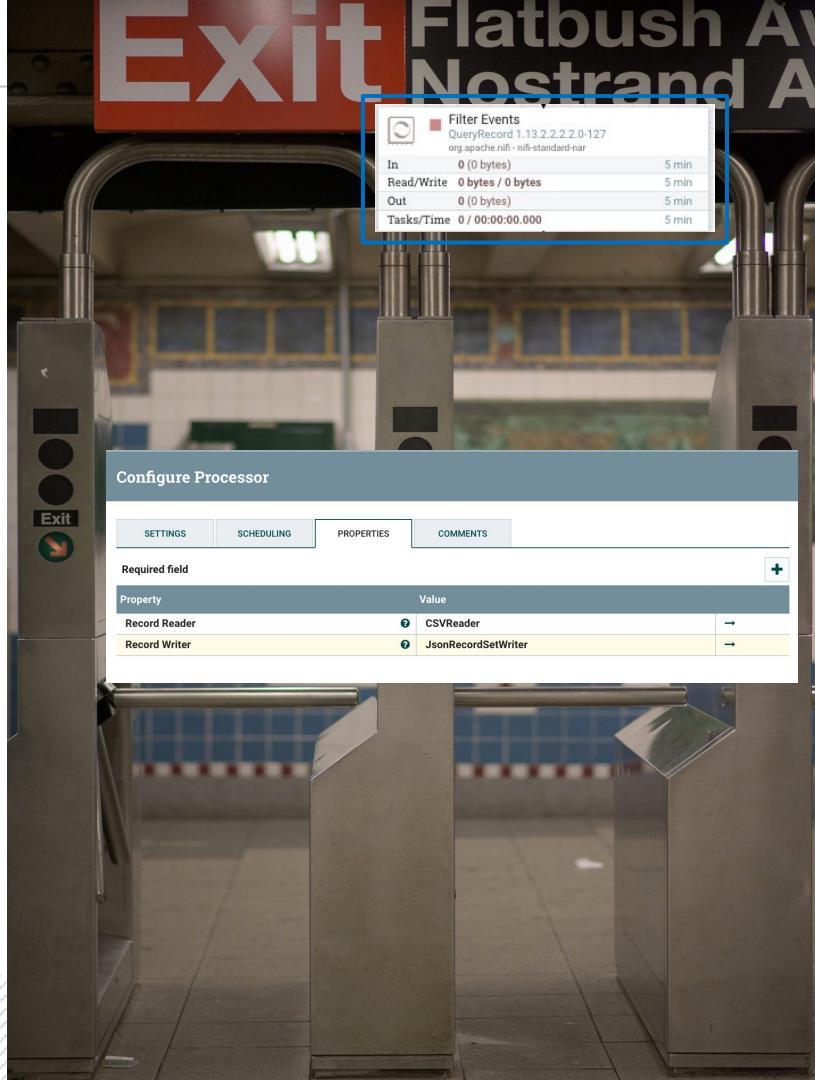
DETAILS ATTRIBUTES CONTENT

Attribute Values

filename	328717796819631
kafka.offset	44815
kafka.partition	6
kafka.topic	nifi-testing
path	/
uuid	32871623852144809510512672385

# RECORD-ORIENTED DATA WITH NIFI

- **Record Readers** - Avro, CSV, Grok, IPFIX, JSAN1, JSON, Parquet, Scripted, Syslog5424, Syslog, WindowsEvent, XML
- **Record Writers** - Avro, CSV, FreeFromText, Json, Parquet, Scripted, XML
- Record Reader and Writer support referencing a schema registry for retrieving schemas when necessary.
- Enable processors that accept any data format without having to worry about the parsing and serialization logic.
- Allows us to keep FlowFiles larger, each consisting of multiple records, which results in far better performance.

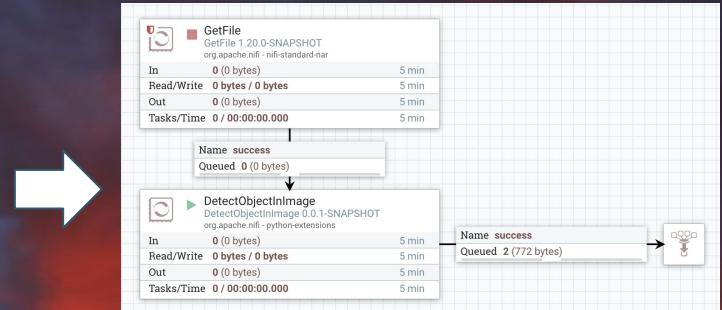


# APACHE NIFI WITH PYTHON CUSTOM PROCESSORS

## Python as a First Class Citizen

```
import cv2
import numpy as np
import json
from nifiapi.properties import PropertyDescriptor
from nifiapi.properties import ResourceDefinition
from nifiapi.flowfiletransform import FlowFileTransformResult
SCALE_FACTOR = 0.00392
NMS_THRESHOLD = 0.4 # non-maximum suppression threshold
CONFIDENCE_THRESHOLD = 0.6

class DetectObjectInImage:
    class Java:
        implements = ['org.apache.nifi.python.processor.FlowFileTransform']
        class ProcessorDescriptor:
            version = '0.0.1-SNAPSHOT'
            dependencies = ['numpy >= 1.23.5', 'opencv-python >= 4.6']
            required = True
            resource_definition = ResourceDefinition(allow_file = True)
        self.config_file = PropertyDescriptor(
            name = 'Network Config File',
            description = 'The text file containing the Network configuration. Supports Caffe (*.prototxt), TensorFlow (*.pbtxt), Darknet (*.cfg), and DLDT (*.xml)',
            required = False,
            resource_definition = ResourceDefinition(allow_file = True))
        self.class_name_file = PropertyDescriptor(
            name = 'Class Names File',
            description = 'A text file containing the names of the classes that may be detected by the model. Expected format is one class name per line, new-line terminated.',
            required = True,
            resource_definition = ResourceDefinition(allow_file = True))
        self.descriptors = [self.model_file, self.config_file, self.class_name_file]
    def getPropertyDescriptors(self):
        return self.descriptors
    def onScheduled(self, context):
        # read class name from text file
        class_name_file = context.getProperty(self.class_name_file.name).getValue()
        if class_name_file is None:
```



<https://github.com/apache/nifi/blob/614947e4ac6798ad80817e82514c39349d5faacb/nifi-docs/src/main/asciidoc/python-developer-guide.adoc>

# READYFLOW GALLERY

- Cloudera provided flow definitions
- Cover most common data flow use cases
- Optimized to work with CDP sources/destinations
- Can be deployed and adjusted as needed

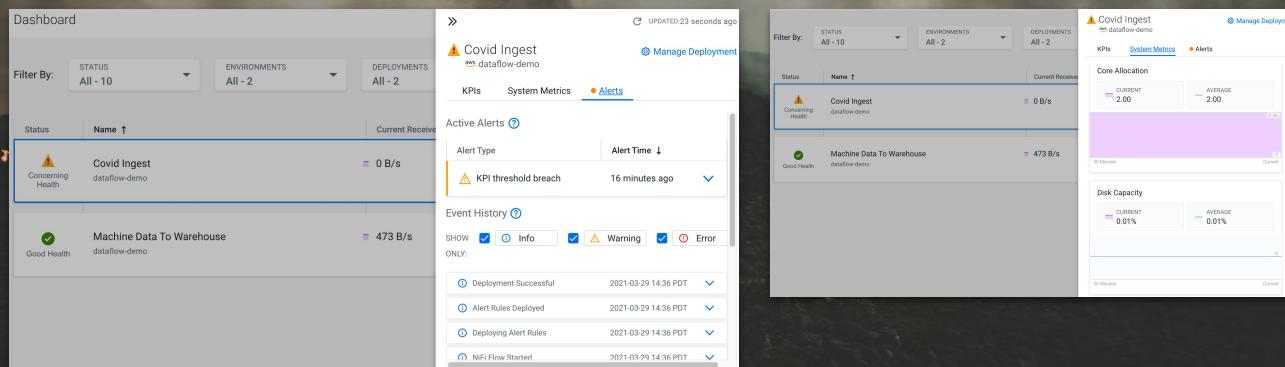
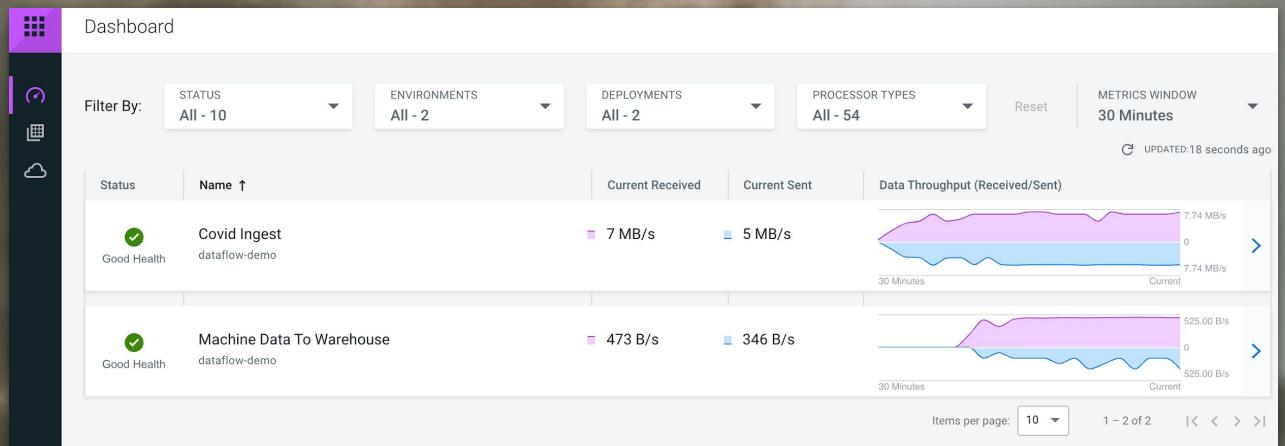
The screenshot shows the Cloudera ReadyFlow Gallery interface. On the left is a sidebar with navigation links: Dashboard, Catalog, ReadyFlow Gallery (which is selected and highlighted in purple), and Environments. Below the sidebar are help links: Help and alpha\_intcookie user cookie, and the version information 1.0.1-b570.

The main area is titled "ReadyFlow Gallery" and contains six flow definitions, each represented by a diagram of three nodes connected by arrows:

- Kafka filter to Kafka**: Version 1. Diagram: Source (Kafka icon) → Filter (wavy arrow icon) → Destination (Kafka icon). Description: Consumes JSON, CSV or Avro events from Kafka, filters them before writing them back to Kafka as JSON, CSV or Avro. [View Added Flow Definition](#)
- Kafka to Cloudera Operational Database**: Version 1. Diagram: Source (Kafka icon) → Sink (green square icon). Description: Consumes JSON, CSV or Avro events from Kafka and ingests them into Cloudera Operational Database (COD). [View Added Flow Definition](#)
- Kafka to Kafka**: Version 1. Diagram: Source (Kafka icon) → Sink (Kafka icon). Description: Consumes events from Kafka and writes them to another Kafka topic. [Add To Catalog](#)
- Kafka to Kudu**: Version 1. Diagram: Source (Kafka icon) → Sink (blue water icon). Description: Consumes JSON, CSV or Avro events from Kafka and ingests them into Kudu. [Add To Catalog](#)
- Kafka to S3 Avro**: Version 1. Diagram: Source (Kafka icon) → Sink (red cube icon). Description: Consumes JSON, CSV or Avro events from Kafka and writes Avro files to S3. [View Added Flow Definition](#)
- S3 to S3 Avro**: Version 1. Diagram: Source (red cube icon) → Sink (S3 icon). Description: Consumes JSON, CSV or Avro files from source S3 location and writes Avro files to a destination S3 location. [Add To Catalog](#)

# DASHBOARD

- Central Monitoring View
- Monitors flow deployments across CDP environments
- Monitors flow deployment health & performance
- Drill into flow deployment to monitor system metrics and deployment events



# I CAN SPEAK NIFI NOW

A cinematic photograph of Keanu Reeves as John Wick. He is shown from the chest up, wearing dark sunglasses and a leather jacket. His right hand is raised to his face, with his fingers resting near his eye. In the background, a large suspension bridge, resembling the Golden Gate Bridge, spans across a body of water under a clear sky.

## RESOURCES AND WRAP-UP

# Future of Data - Princeton + Virtual



<https://www.meetup.com/futureofdata-princeton/>

From Big Data to AI to Streaming to Containers to Cloud to Analytics to Cloud Storage to Fast Data to Machine Learning to Microservices to ...



CLOUDERA



# FUTURE OF DATA

AN OPEN SOURCE COMMUNITY



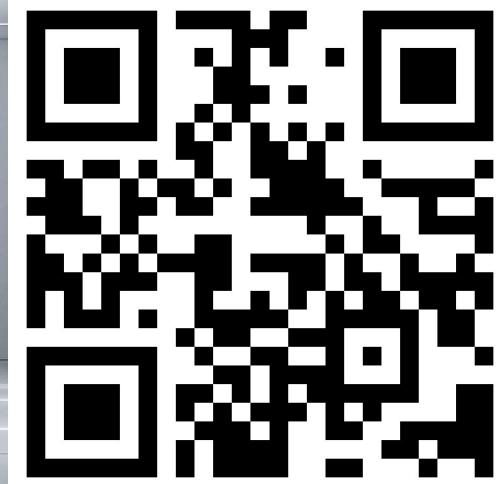
@PaasDev

© 2020 Cloudera, Inc. All rights reserved.

# FLaNK Stack Weekly



<https://bit.ly/32dAJft>



This week in Apache NiFi, Apache Flink, Apache Kafka, Apache Spark, Apache Iceberg, Python, Java, AI, ML, LLM and Open Source friends.

# CSP Community Edition

- Kafka, KConnect, SMM, SR, Flink, and SSB in Docker
- Runs in Docker
- Try new features quickly
- Develop applications locally



- Docker compose file of CSP to run from command line w/o any dependencies, including Flink, SQL Stream Builder, Kafka, Kafka Connect, Streams Messaging Manager and Schema Registry

○ \$> `docker compose up`

- Licensed under the Cloudera Community License
- Unsupported
- Community Group Hub for CSP
- Find it on [docs.cloudera.com](https://docs.cloudera.com) under Applications



## CSP Community Edition

A readily available, dockerized deployment of Apache Kafka and Apache Flink that allows you to test the features and capabilities of Cloudera Stream Processing.

[Learn More](#)

# Open Source Edition



- Apache NiFi in Docker
- Runs in Docker
- Try new features quickly
- Develop applications locally

- Docker NiFi

- ```
docker run --name nifi -p 8443:8443 -d -e
  SINGLE_USER_CREDENTIALS_USERNAME=admin -e
  SINGLE_USER_CREDENTIALS_PASSWORD=ctsBtRBKHRAx69EqUghv
  vgEvjnaLjFEB apache/nifi:latest
```

- Licensed under the ASF License
- Unsupported

<https://hub.docker.com/r/apache/nifi>



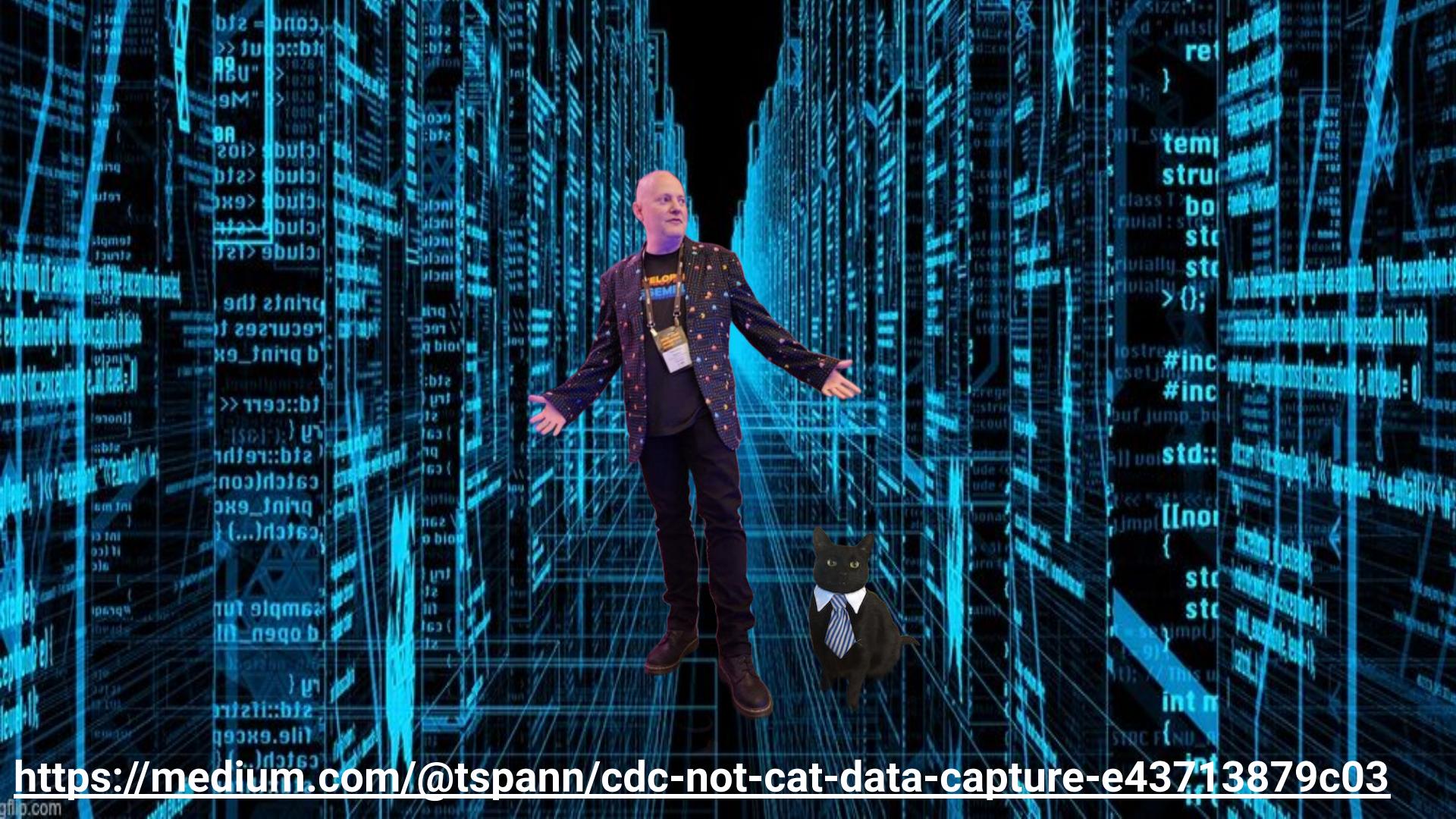
# Resources



[threads.net/@tspannhw](https://threads.net/@tspannhw)



**FLANK STACK**



<https://medium.com/@tspann/cdc-not-cat-data-capture-e43713879c03>

# COLLECTING DATA WITH NIFI

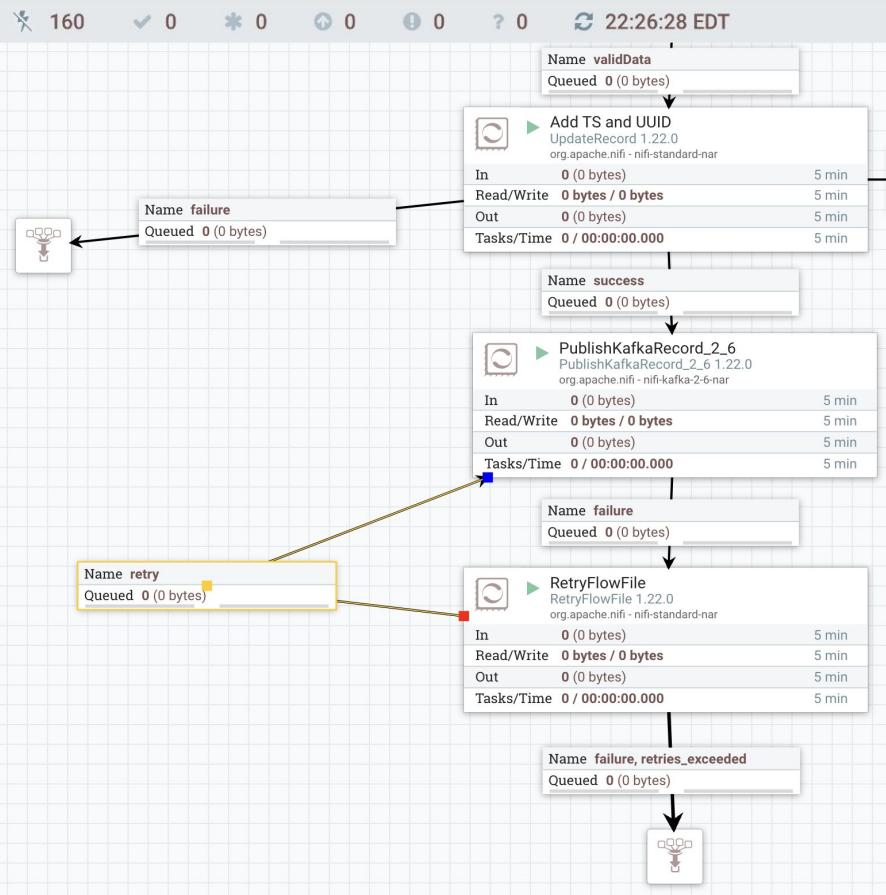
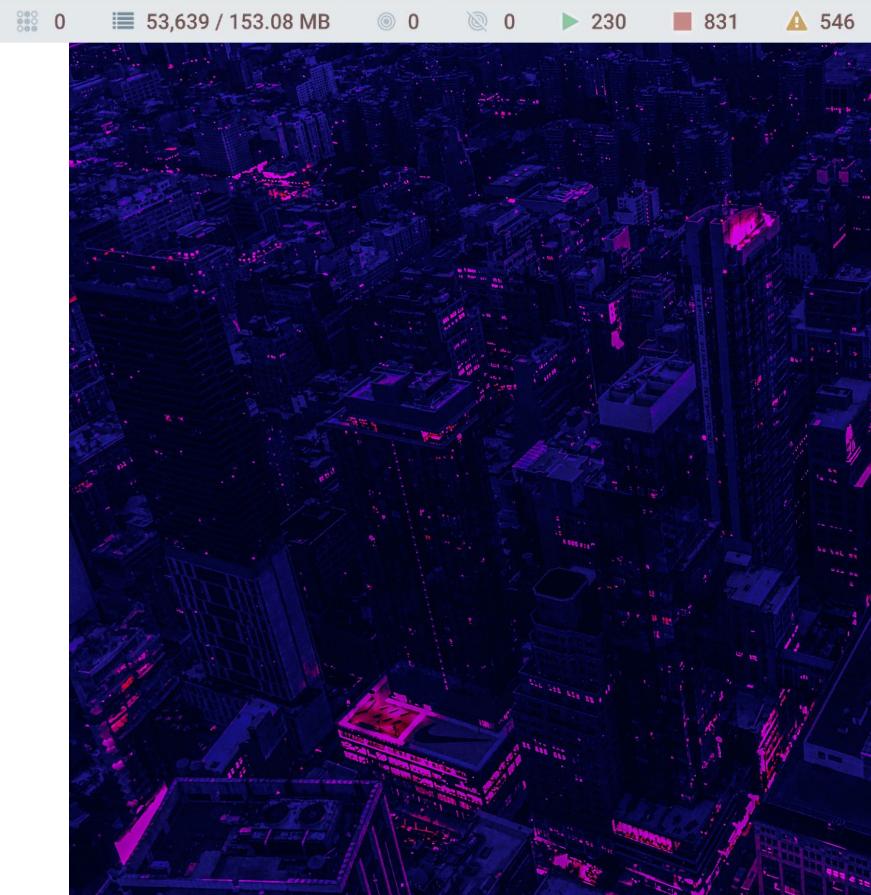
DATA INC

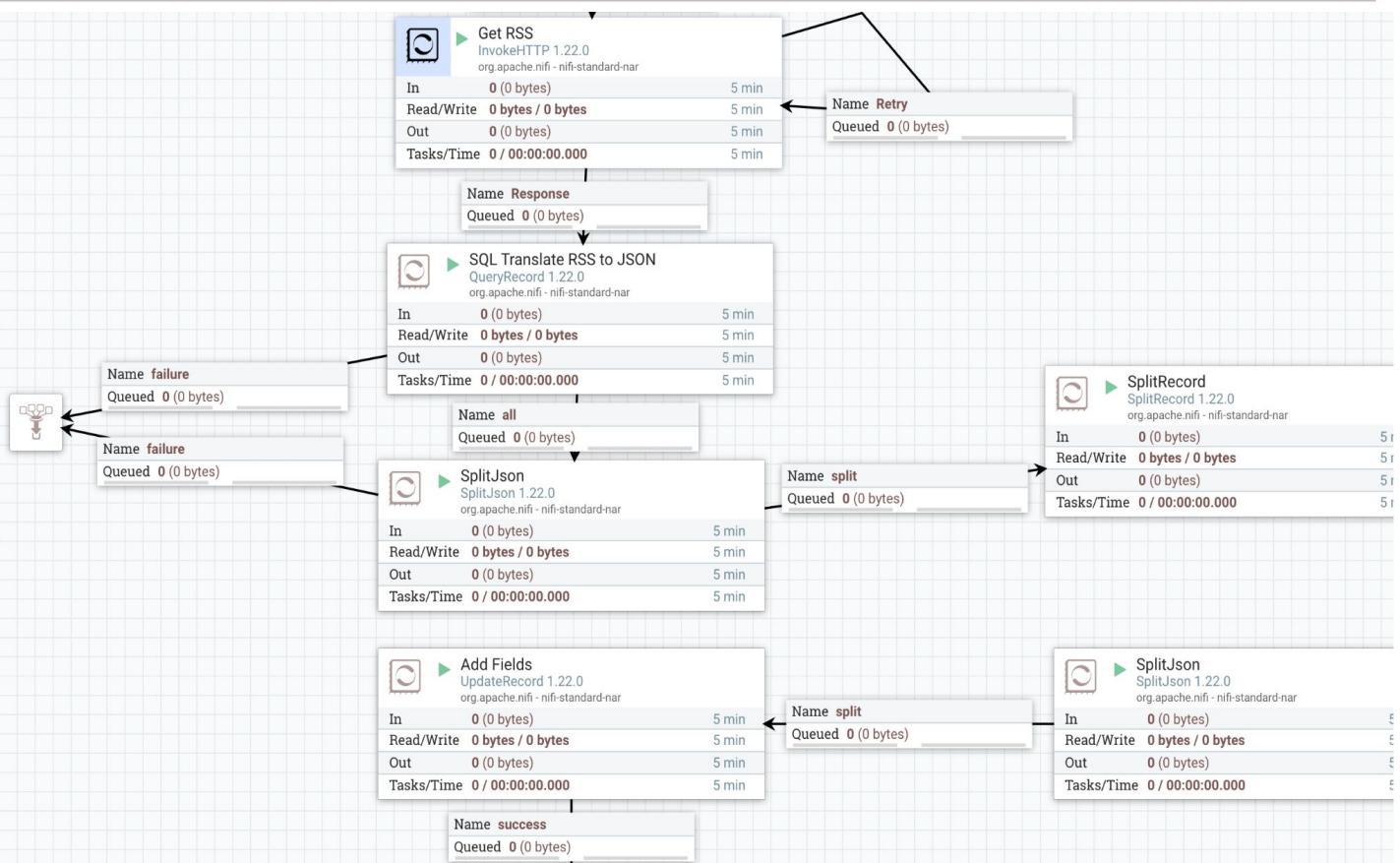
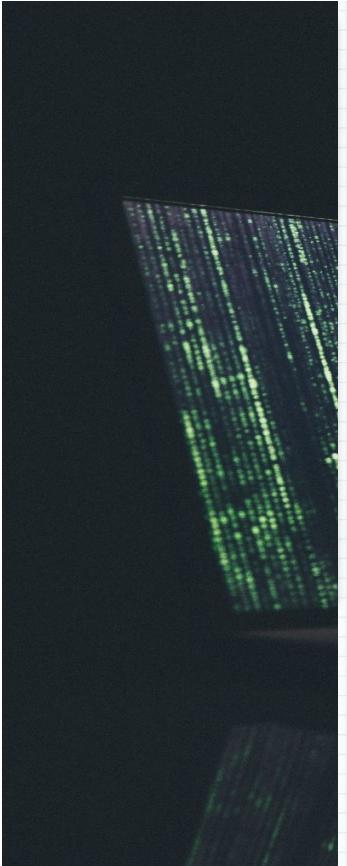
softsys

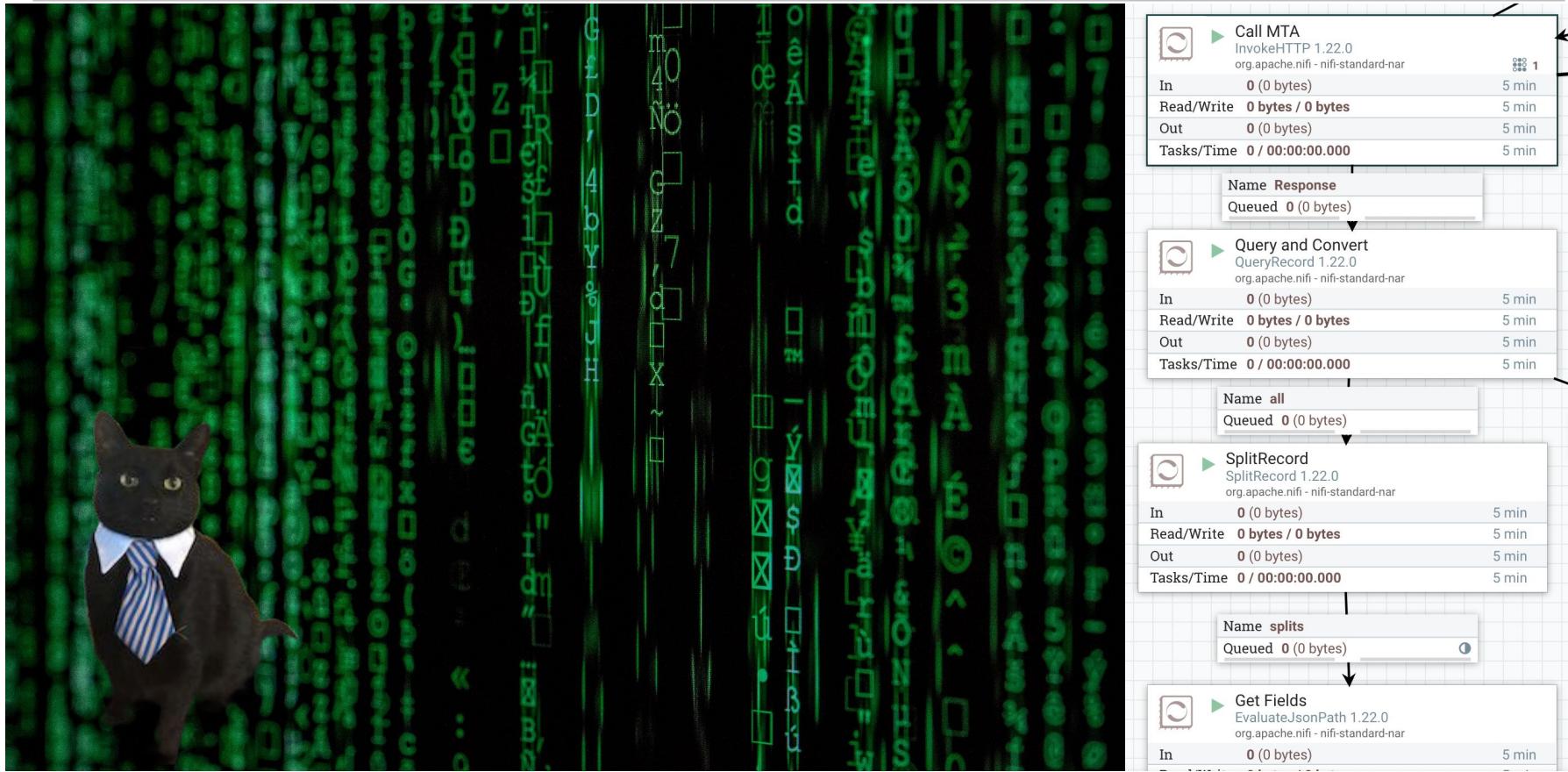
---

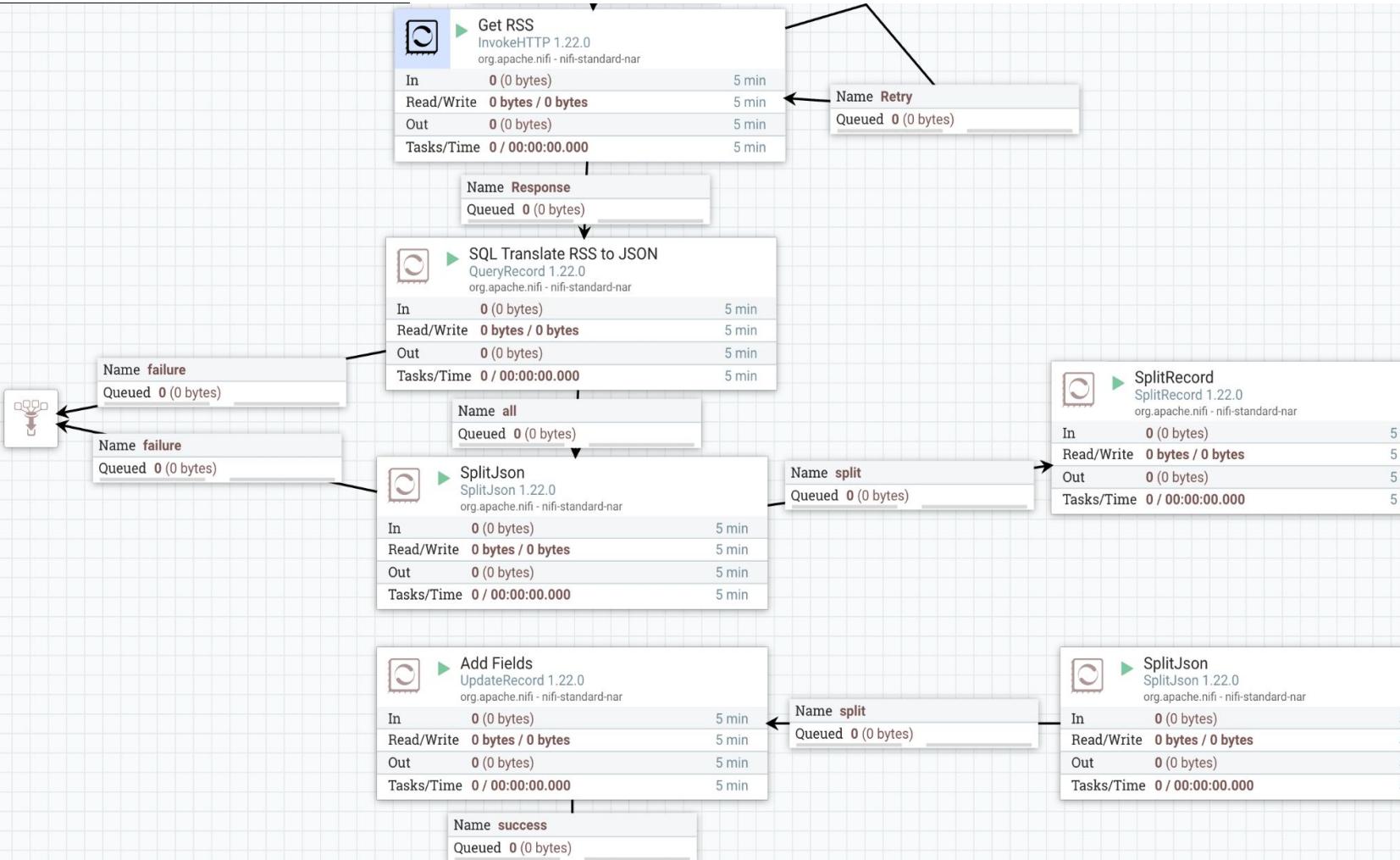
# WALK THROUGH

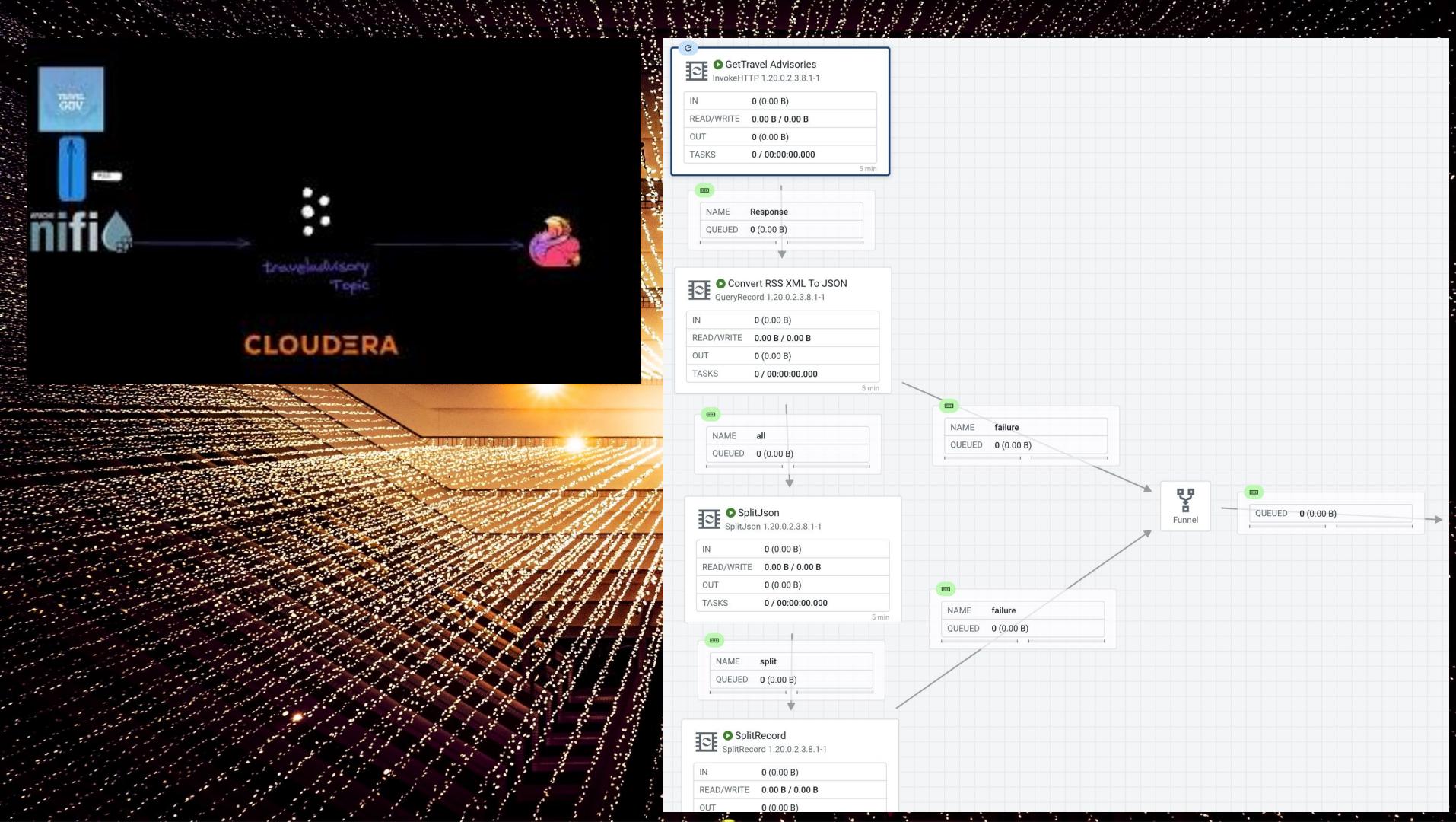


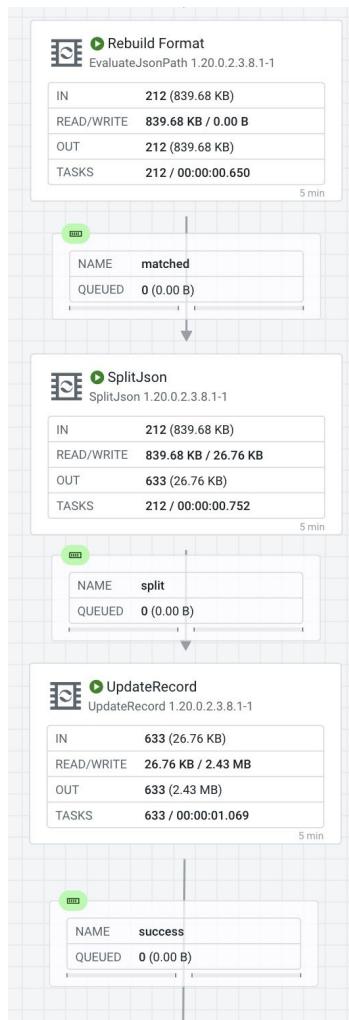


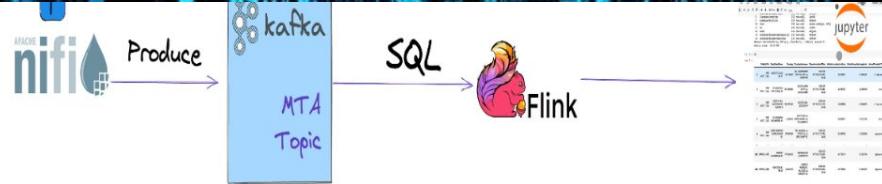












## CLOUDERA

Show 10 entries Search:

| Vehicle       | Stop                      | Destination                             | Expected Arrival              | How Close?    | Distance Away | # of Passengers |
|---------------|---------------------------|-----------------------------------------|-------------------------------|---------------|---------------|-----------------|
| MTA NYCT_9686 | 4 AV/E 10 ST              | EAST VILLAGE 8 ST via 5 AV              | 2023-07-11T14:53:00.000-04:00 | approaching   | 118           |                 |
| MTA NYCT_9651 | MURRAY ST/CHURCH ST       | BATTERY PK CITY via AVENUE C via PK ROW | 2023-07-11T14:51:10.862-04:00 | < 1 stop away | 351           | 5               |
| MTA NYCT_9639 | FT WASHINGTON AV/W 168 ST | MIDTOWN 32 ST & 5 AV via BWAY 5 AV      | 2023-07-11T14:51:06.076-04:00 | approaching   | 148           | 33              |
| MTA NYCT_8387 | W 231 ST/BROADWAY         | NORWOOD 205 ST STA                      | 2023-07-11T14:50:22.856-04:00 | < 1 stop away | 287           | 20              |
| MTA NYCT_9649 | CENTRE ST/CHAMBERS ST     | BATTERY PK CTY VESEY ST                 | 2023-07-11T14:50:09.630-04:00 | < 1 stop away | 317           | 14              |
| MTA NYCT_7890 | PENNSYLVANIA AV/PITKIN AV | SPRING CREEK GATEWAY MALL               | 2023-07-11T14:48:55.081-04:00 | < 1 stop away | 349           | 19              |
| MTA NYCT_8382 | OGDEN AV/W 162 ST         | PLIMPTON E.L. GRANT HWY                 | 2023-07-11T14:48:53.459-04:00 | < 1 stop away | 346           | 47              |
| MTA NYCT_8350 | KISSENA BL/45 AV          | FLUSHING MAIN ST STATION                | 2023-07-11T14:48:51.609-04:00 | < 1 stop away | 213           | 17              |
| MTA NYCT_8331 | ARCHER AV/SUTPHIN PL      | JAMAICA LIRR                            | 2023-07-11T14:48:47.233-04:00 | approaching   | 150           |                 |
| MTA NYCT_8388 | BROADWAY/W 178 ST         | MANHTTNVILLE W 125 via BWAY via AMSTRDM | 2023-07-11T14:48:44.718-04:00 | < 1 stop away | 161           | 21              |
| Vehicle       | Stop                      | Destination                             | Expected Arrival              | How Close?    | Distance Away | # of Passengers |

Showing 1 to 10 of 100 entries

Previous 1 2 3 4 5 ... 10 Next

