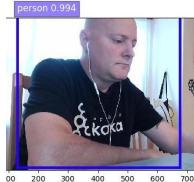




Unlocking Financial Data with Real-Time Pipelines

Tim Spann
Principal Developer Advocate

14-December-2023



OSA CON 23



UNLOCKING FINANCIAL DATA
WITH REAL-TIME PIPELINES

Timothy Spann
Principal Developer Advocate
for Data in Motion at Cloudera



**ENTERPRISE
DATA CLOUD**

CLOUDERA



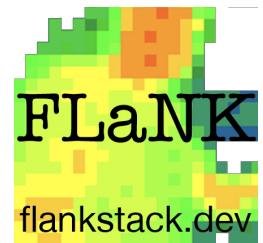
CLOUDERA



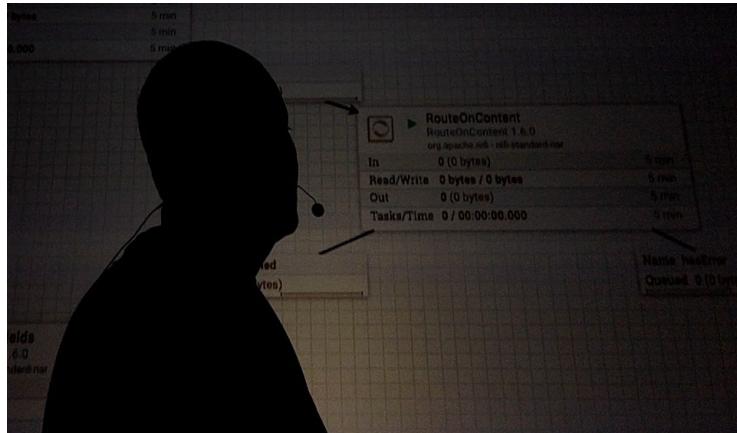
**EDGE
2AI**

CLOUDERA





Agenda (30 minutes)



Introduction

Overview

Apache Kafka

Apache Flink

Apache NiFi

Use Cases

Demos

FLaNK Stack Weekly by Tim Spann



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

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



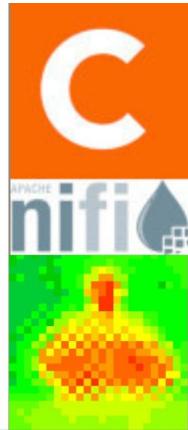
This week in Apache NiFi, Apache Flink, Apache Kafka, ML, AI, Apache Spark, Apache Iceberg, Python, Java and Open Source friends.

Future of Data - NYC + NJ + Philly + 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



@PaasDev

Tim Spann

Twitter: @PaasDev // Blog: datainmotion.dev

Principal Developer Advocate.

Princeton Future of Data Meetup.

ex-Pivotal, ex-Hortonworks, ex-StreamNative, ex-PwC

<https://medium.com/@tspann>

<https://github.com/tspannhw>



 DZone. REF CARDS TREND REPORTS EXPERTS

Top IoT Experts



Tim Spann
Principal Developer Advocate,
Cloudera
<https://github.com/tspannhw/SpeakerProfile/>
Tim Spann is a Principal Developer Advocate in Data in Motion for Cloudera. He works with Apache NiFi, Apache Pulsar, Apache...



Financial institutions thrive on accurate and timely data to drive critical decision-making processes, risk assessments, and regulatory compliance. However, managing and processing vast amounts of financial data in real-time can be a daunting task. To overcome this challenge, modern data engineering solutions have emerged, combining powerful technologies like Apache Flink, Apache NiFi, Apache Kafka, and Iceberg to create efficient and reliable real-time data pipelines. In this talk, we will explore how this technology stack can unlock the full potential of financial data, enabling organizations to make data-driven decisions swiftly and with confidence.

Introduction: Financial institutions operate in a fast-paced environment where real-time access to accurate and reliable data is crucial. Traditional batch processing falls short when it comes to handling rapidly changing financial markets and responding to customer demands promptly. In this talk, we will delve into the power of real-time data pipelines, utilizing the strengths of Apache Flink, Apache NiFi, Apache Kafka, and Iceberg, to unlock the potential of financial data.

Key Points to be Covered:

Introduction to Real-Time Data Pipelines: a. The limitations of traditional batch processing in the financial domain. b. Understanding the need for real-time data processing.

Apache Flink: Powering Real-Time Stream Processing: a. Overview of Apache Flink and its role in real-time stream processing. b. Use cases for Apache Flink in the financial industry. c. How Flink enables fast, scalable, and fault-tolerant processing of streaming financial data.

Apache Kafka: Building Resilient Event Streaming Platforms: a. Introduction to Apache Kafka and its role as a distributed streaming platform. b. Kafka's capabilities in handling high-throughput, fault-tolerant, and real-time data streaming. c. Integration of Kafka with financial data sources and consumers.

Apache NiFi: Data Ingestion and Flow Management: a. Overview of Apache NiFi and its role in data ingestion and flow management. b. Data integration and transformation capabilities of NiFi for financial data. c. Utilizing NiFi to collect and process financial data from diverse sources.

Iceberg: Efficient Data Lake Management: a. Understanding Iceberg and its role in managing large-scale data lakes. b. Iceberg's schema evolution and table-level metadata capabilities. c. How Iceberg simplifies data lake management in financial institutions.

Real-World Use Cases: a. Real-time fraud detection using Flink, Kafka, and NiFi. b. Portfolio risk analysis with Iceberg and Flink. c. Streamlined regulatory reporting leveraging all four technologies.

Best Practices and Considerations: a. Architectural considerations when building real-time financial data pipelines. b. Ensuring data integrity, security, and compliance in real-time pipelines. c. Scalability and performance optimization techniques.

Conclusion: In this talk, we will demonstrate the power of combining Apache Flink, Apache NiFi, Apache Kafka, and Iceberg to unlock financial data's true potential. Attendees will gain insights into how these technologies can empower financial institutions to make informed decisions, respond to market changes swiftly, and comply with regulations effectively. Join us to explore the world of real-time data pipelines and revolutionize financial data management.

OVERVIEW

CUNK ON FLANK

A woman with blonde hair, wearing a blue blazer over a light-colored top, is shown from the chest up. She has a surprised or shocked expression, with her mouth slightly open and eyes wide. The background shows a city street with buildings and a sidewalk.

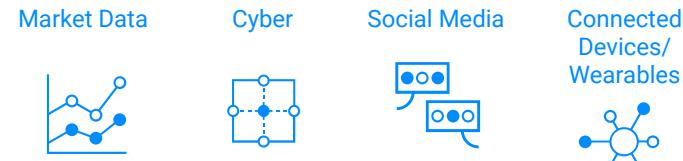
THEY SAY IT'S FOR FAST DATA

IS IT SO FAST THAT IT
ARRIVES BEFORE I ASKED FOR IT

DATA VELOCITY in FINANCIAL SERVICES

Streaming capabilities vary, all enhance insight

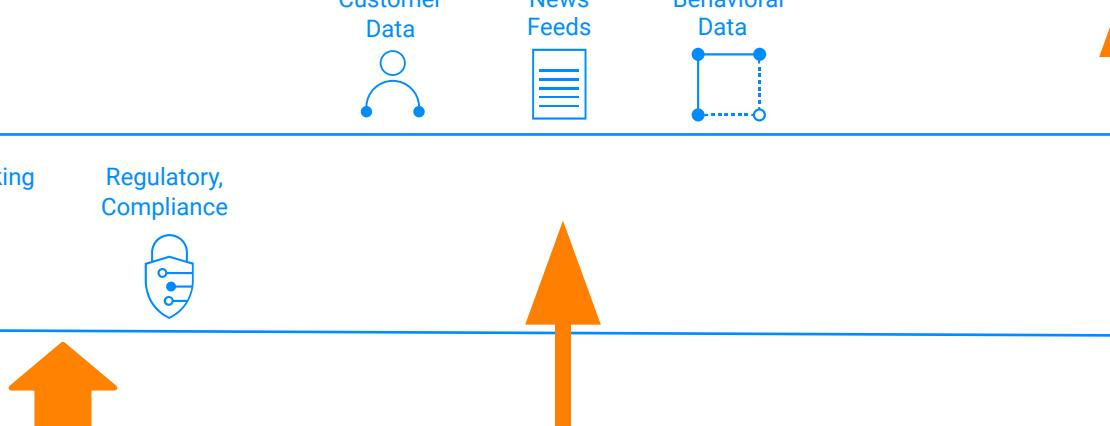
Real-Time Streaming



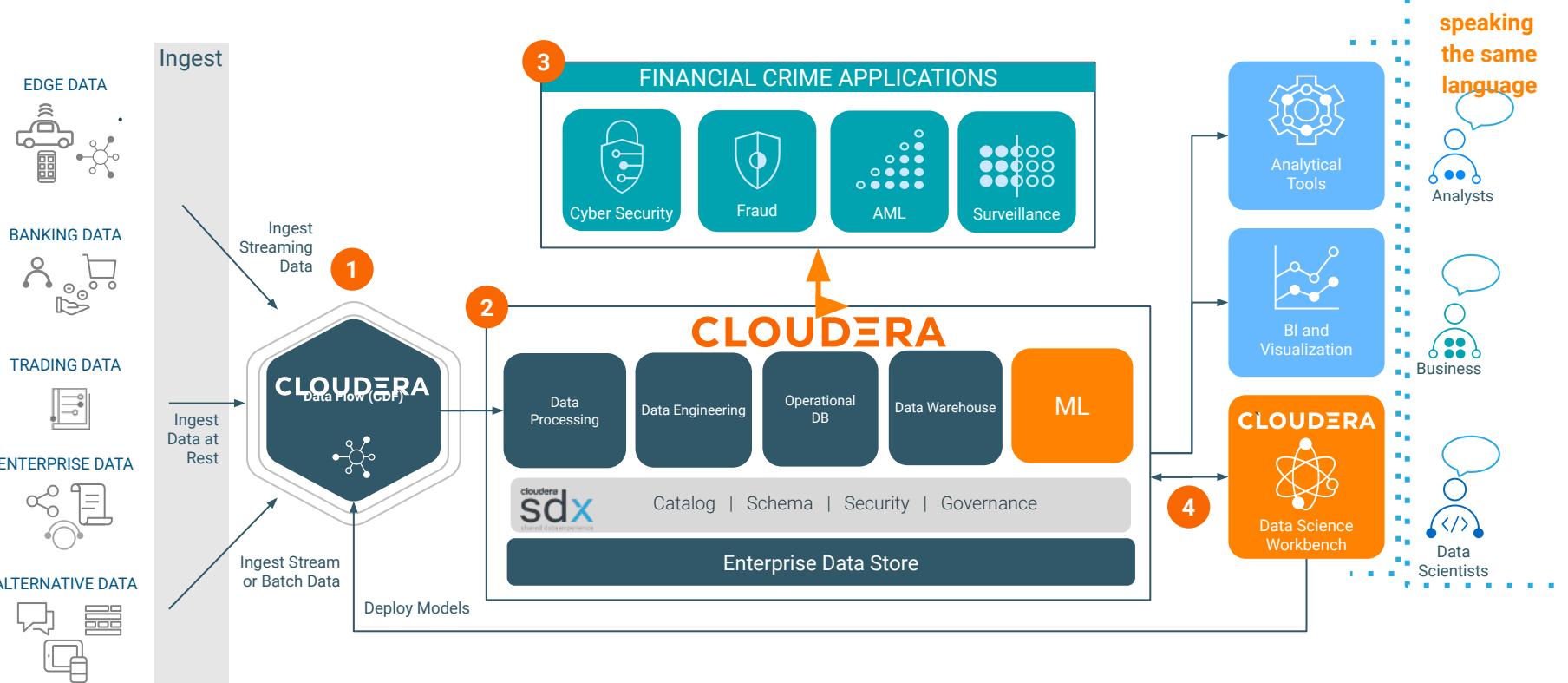
Near-Real Time Streaming



Normal Streaming

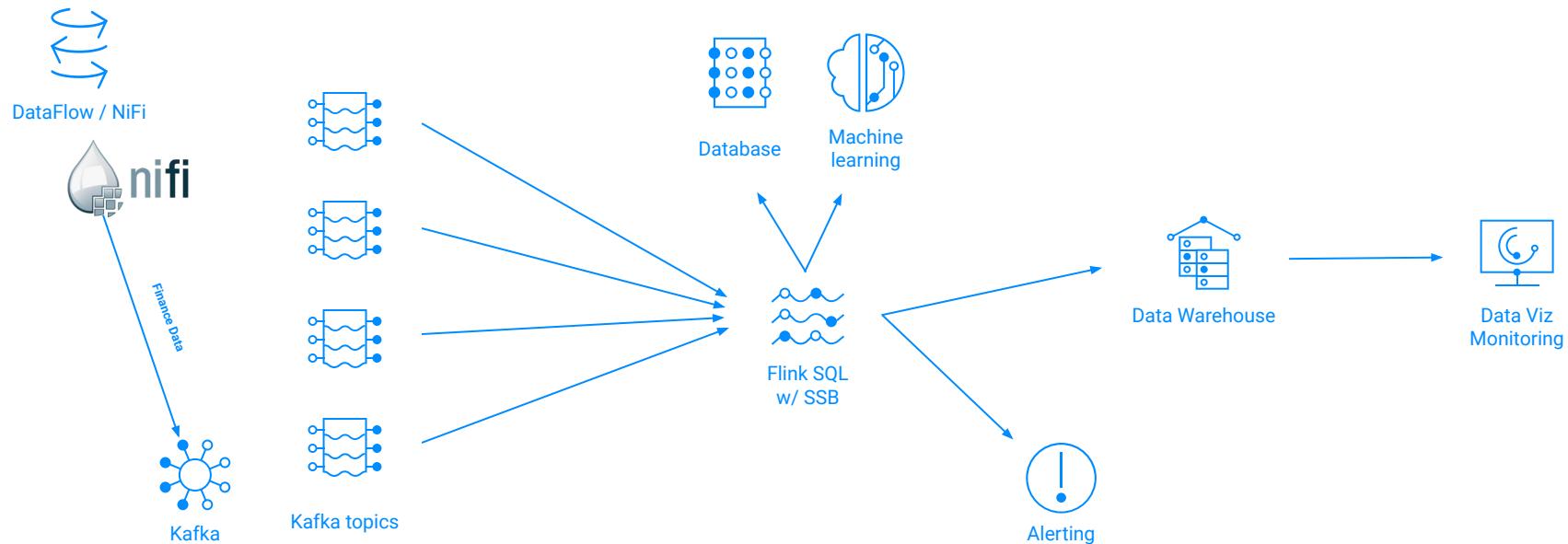


NEXT GEN PLATFORM FOR TACKLING FINANCIAL CRIME

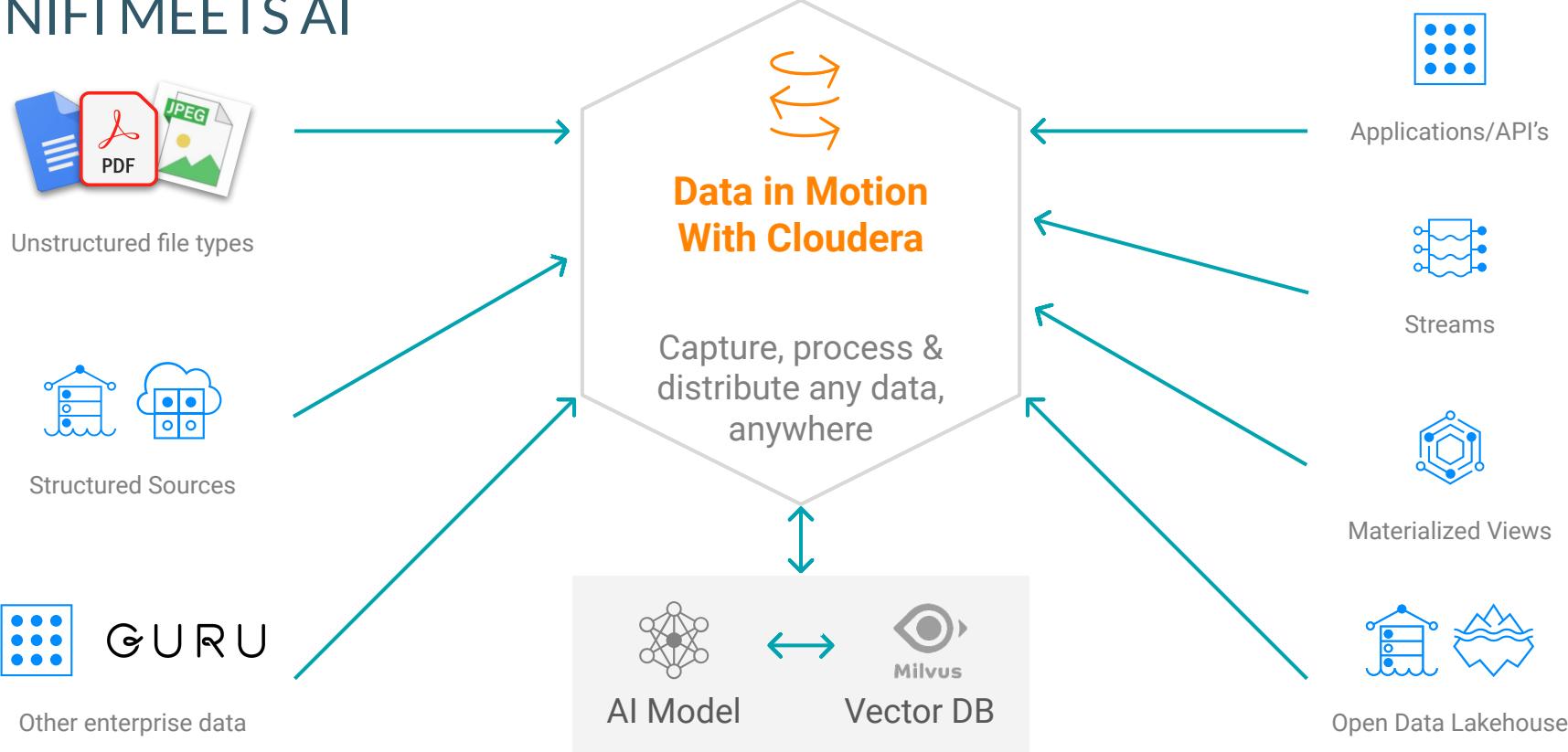


Architecture in the context of Financial Use Cases

Kafka & Flink (Flink SQL with Stream SQL Builder) for real time analytics

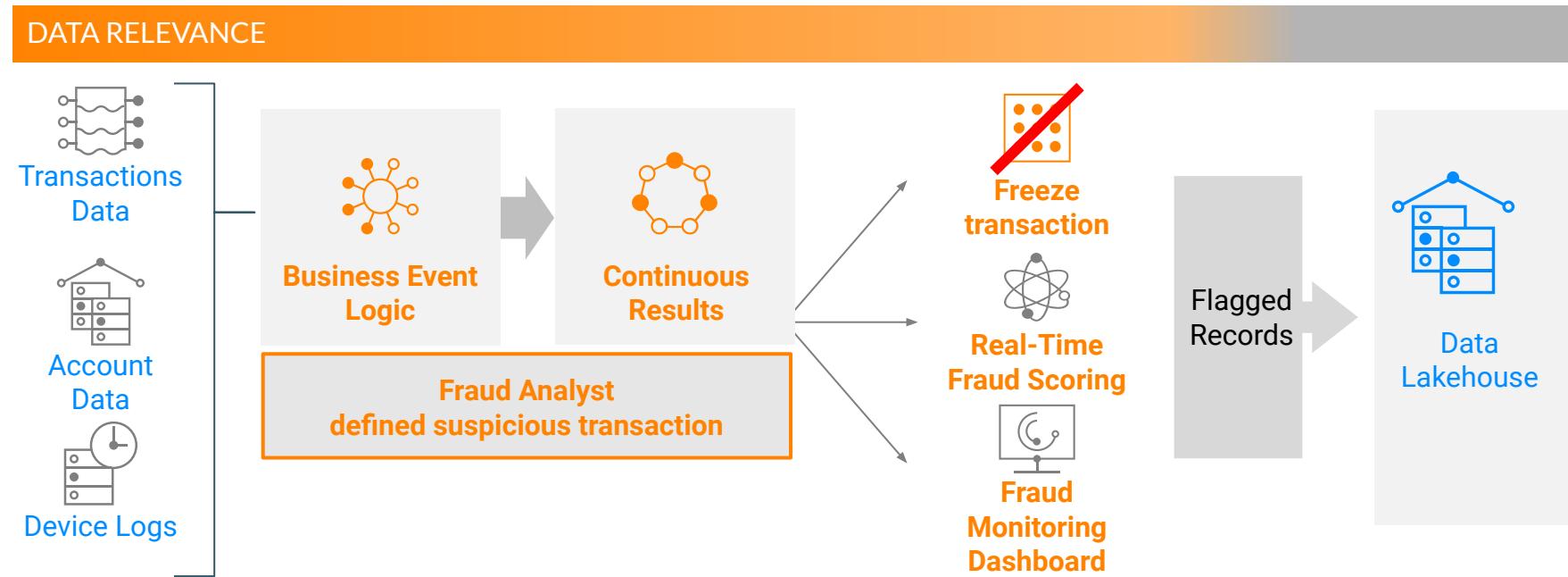


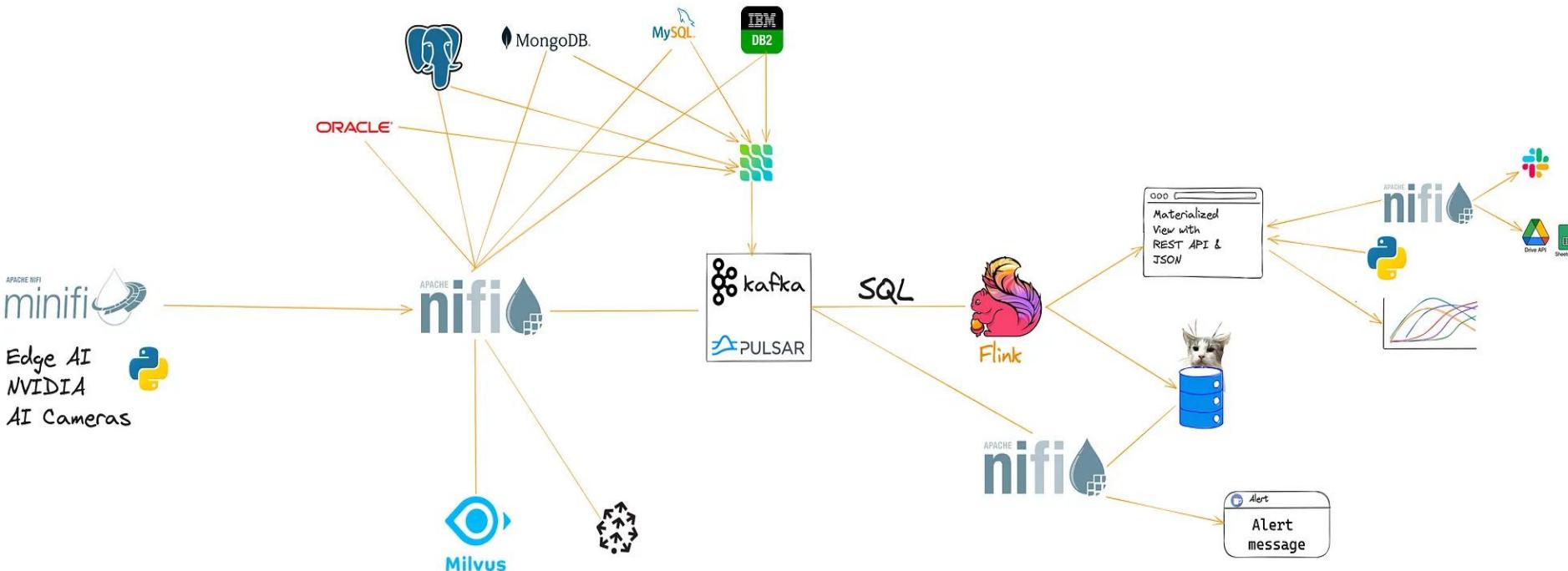
NIFI MEETS AI

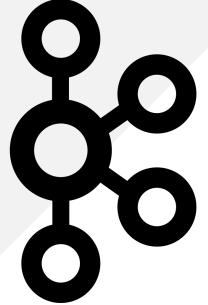


Stop Fraud When It Happens—Real Life Example

Simplified example of deployed use case







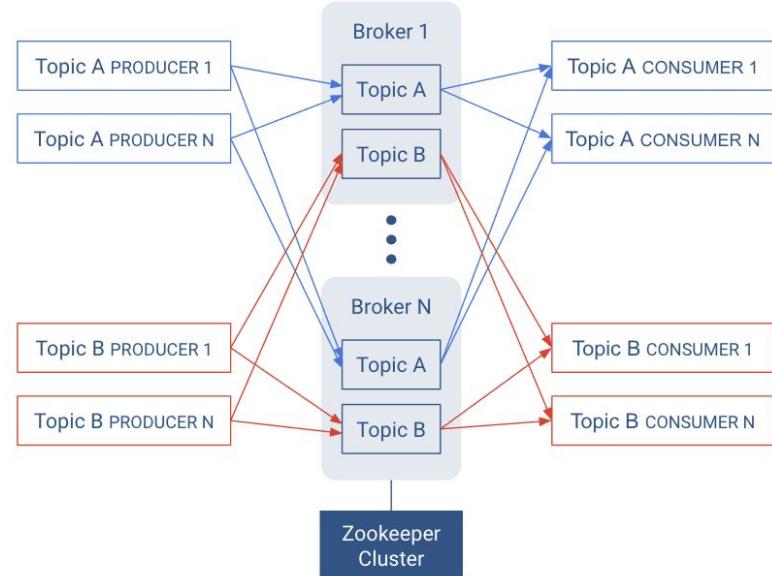
APACHE KAFKA

STREAMS MESSAGING WITH KAFKA



WriteToKafka		PublishKafka2RecordCDP 1.0.0.2.2.2.0-127 com.cloudera - nifi-cdf-kafka-2-nar
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

- Highly reliable distributed messaging system.
- Decouple applications, enables many-to-many patterns.
- Publish-Subscribe semantics.
- Horizontal scalability.
- Efficient implementation to operate at speed with big data volumes.
- Organized by topic to support several use cases.





APACHE FLINK

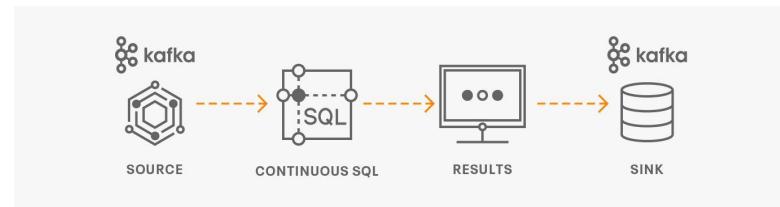
CONTINUOUS SQL

- SSB is a Continuous SQL engine
- It's SQL, but a slightly different mental model, but with big implications

Traditional Parse/Execute/Fetch model



Continuous SQL Model



Hint: The query is boundless and never finishes, and time matters

AKA: `SELECT * FROM foo WHERE 1=0 -- will run forever`

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.

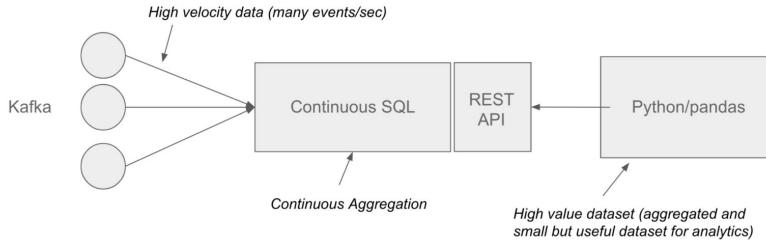
Simplifies access to data in Kafka & Flink. Connectors to batch data in HDFS, Kudu, Hive, S3, JDBC, CDC and more

Enrich streaming data with batch data in a single tool

```
CREATE TABLE `kafka_table_1670513700` (
  `col_str` STRING,
  `col_int` INT,
  `col_ts` TIMESTAMP(3),
  WATERMARK FOR `col_ts` AS `col_ts` - INTERVAL '5' SECOND
) WITH (
  'connector' = 'kafka',
  'topic' = 'yourTopicName',
  'bootstrap.servers' = '...', -- Comma separated list of Kafka brokers,
  'format' = 'json',
  'value-decoder.type-as' = 'plain',
  'ignore-parse-error.per-message' = 'false' -- Optional flag to specify whether to ignore parse errors per message by default.
)
Note, only one of 'topic-pattern' and 'topic' can be specified for sources. When the table is used as sink, the topic name is the topic to write data to. Note topic list is not supported for sinks.
-- 'value-decoder.type-as' can be set to 'plain' or 'json' by default.
-- 'ignore-parse-error.per-message' is false by default.
-- 'json.fail-on-missing-field' = 'false' -- Optional flag to specify whether to fail if a field is missing or not, false by default.
-- 'json.ignore-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.
-- 'json.map-null-key.literal' = 'null' -- Optional flag to specify string literal for null keys when 'map-null-key.mode' is LITERAL, '\"null\"' by default.
-- 'json.map-null-key.mode' = 'FAIL' -- Optional flag to control the handling mode when serializing null key for map data, FAIL by default.
Option DROP will drop null key entries for map data. Option LITERAL will use 'map-null-key.literal' as key literal.
[08/12/2022, 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] CREATE TABLE: `kafka_table_1670513700`(`col_str` STRING, `col_int` INT, `col_ts` TIMESTAMP(3), properties.bootstrap.servers='...', schema.0.name=col_str, schema.0.type=STRING, properties.bootstrap.servers='...', schema.1.name=col_ts, schema.1.type=TIMESTAMP(3), connector=kafka, schema.watermark.b.rounding=col_ts, schema.watermark.b.strategy,data-type=TIMESTAMP(3), topic='...', schema.0.name=col_str, identifier: l(ssb, ssb_default))
[08/12/2022, 16:35:02] [INFO] Active job started
[08/12/2022, 16:35:12] [INFO] yourTable 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
```



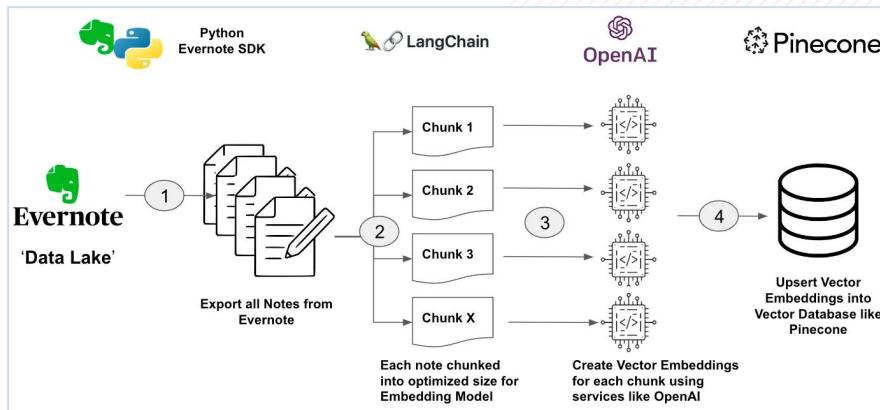
```
[90]: import pandas as pd
[91]: mv = "https://xxxxxxxxxx"
[92]: df = pd.read_json(mv)
[93]: len(df.keys())
[93]: 5
[95]: df['ts'] = pd.to_datetime(df['ts'])
[97]: df.dtypes
[97]: max_amount      int64
       sum_amount      int64
       thecount        int64
       ts              datetime64[ns]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



APACHE NIFI

NiFi 2.0.0-M1 is here... <https://medium.com/cloudera-inc/getting-ready-for-apache-nifi-2-0-5a5e6a67f450>

- First-class citizen Python API
- Rules Engine
- NiFi Stateless at Process Group level
- Java 21 (virtual threads, perf improvements, etc)



Closing the gap between data engineers and data scientists...

- Export documentation (Sharepoint, OCR) to build the knowledge base powering your chatbot
- Scrape the internet (Sitemap) to build the knowledge base powering your chatbot
- Real-time streaming ingest of Slack to build the knowledge base powering your chatbot

Evernote AI Chatbot Powered by Apache NiFi using OpenAI, Pinecone & Langchain

The screenshot shows a conversation history for an Evernote AI Chatbot. The interface includes a message input field, a list of messages, and a message preview area. The messages are as follows:

- You: When did my right elbow pain occur? (with a smiley face icon)
- Bot: The right elbow pain occurred on January 11, 2023. (with a smiley face icon)
- You: What doctor did I see for it? (with a smiley face icon)
- Bot: You saw Dr. Patel for your right elbow pain. (with a smiley face icon)
- You: What was the cause for the right elbow pain and what did the doctor suggest? (with a smiley face icon)
- Bot: The cause for the right elbow pain is tennis elbow. The doctor suggested a cortisone shot in the right elbow and taking Diclofenac for prescription pain relief. (with a smiley face icon)
- You: What was the dosage for Diclofenac? (with a smiley face icon)
- Bot: The dosage for Diclofenac is 50 mg, 3 times a week. (with a smiley face icon)

At the bottom, there is a message input field labeled 'You:' and a 'Send' button.

<https://medium.com/@george.vetticaden/accelerating-ai-data-pipelines-building-an-evernote-chatbot-with-apache-nifi-2-0-and-generative-ai-9d977466ff4c>

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 grey circle labeled "DROP". Two green arrows originate from the "RECEIVE" and "JOIN" circles and point to a separate window titled "Provenance Event".

Provenance Event

DETAILS

ATTRIBUTES

CONTENT

Attribute Values

filename	328717796819631
	No value previously set
kafka.offset	44815
	No value previously set
kafka.partition	6
	No value previously set
kafka.topic	nifi-testing
	No value previously set
path	/
	No value previously set
uuid	328717796819631-44800-10519073-0E

DEMO



Cloudera Stream Processing Community Edition



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](#)



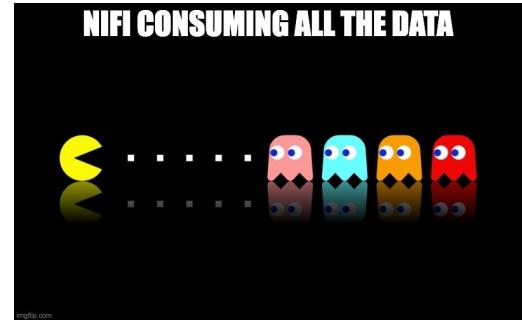
- **Zero to Flink in less than an hour**
 - Experiment with features
 - Develop apps locally
- One docker compose file of CSP which includes:
 - All dependencies required to run
 - Kafka, Kafka Connect and Flink
 - Streams Messaging Manager
 - Schema Registry
 - **SQL Stream Builder**
- Licensed under the Cloudera Community License
- Community Group Hub (Discussion Forum) for CSP
- Find it on [docs.cloudera.com](#) under Applications

Open Source Edition

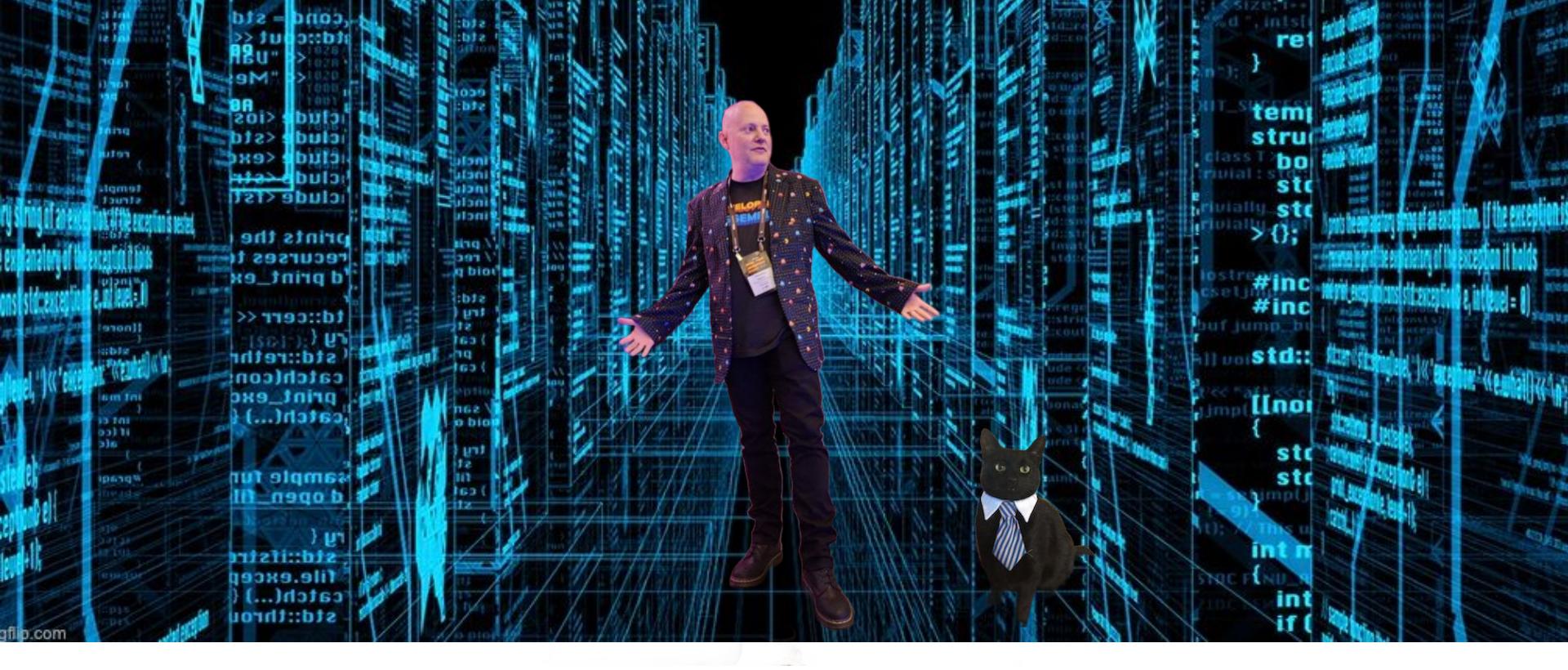


- Apache NiFi in Docker
- Runs in Docker
- Try new features quickly
- Develop <https://hub.docker.com/r/apache/nifi> applications locally

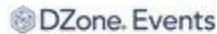
- Docker NiFi
 - `docker run --name nifi -p 8443:8443 -d -e SINGLE_USER_CREDENTIALS_USERNAME=admin -e SINGLE_USER_CREDENTIALS_PASSWORD=ctsBtRBKHRAx69EqUghvvgEvjnaLjFEB apache/nifi:latest`
 - Licensed under the ASF License
 - **Unsupported**



ed.



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



Data Pipelines Virtual Roundtable

Friday, October 27, 2023 | 12 PM ET

REGISTER NOW



Timothy Spann
Principal Developer Advocate,
Cloudera



Eric Sammer
CEO,
Decodable



Jesse Davis
Moderator,
DZone Chief Technologist



Amol Dongre
Sr Director of Product Management,
Informatica



Miguel Lorenzo
VP of Engineering,
Nextall

SPONSORED BY
  Decodable / Informatica

<https://events.dzone.com/dzone/Data-Pipelines-Investigating-the-Modern-Day-Stack>

THE FUTURE OF THE WORLD



TH^ON^G Y^OU

