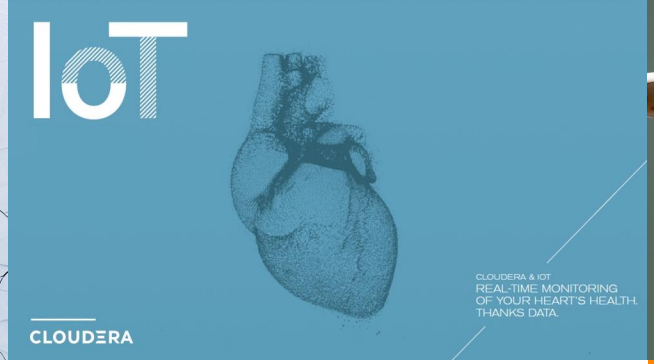
CLOUDERA

Mastering Data Streaming Pipelines

Tim Spann
Principal Developer Advocate

09-May-2023

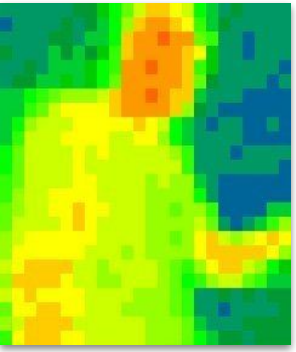


CLOUDERA

EDGE
2AI
CLOUDERA

ENTERPRISE
DATA
CLOUD
CLOUDERA

FLaNK Stack



Tim Spann

@PaasDev // Blog: **www.datainmotion.dev**

Principal Developer Advocate.

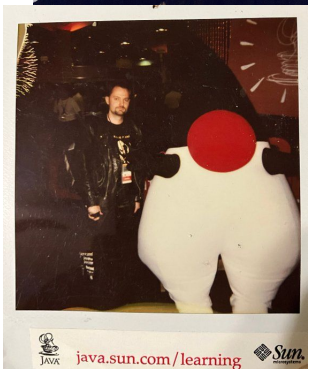
Princeton Future of Data Meetup.

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

https://medium.com/@tspann

https://github.com/tspannhw

Apache NiFi x Apache Kafka x Apache Flink x Java



FLiP Stack Weekly

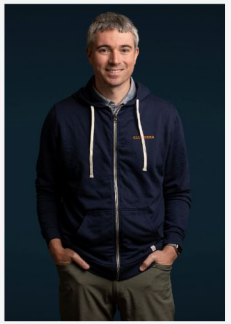
https://bit.ly/32dAJft

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

Hosts

Joe Witt, VP of Engineering, Cloudera

Mark Payne, Principal Engineer, Cloudera



Matt Gilman, Principal Engineer, Cloudera

See details here: https://lnkd.in/gCUVSvM6 (Passcode: a.7eZl5N).

Future of Data - Princeton + Virtual

FUTURE OF DATA

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 ...

ICEBERG

@PaasDev

FREE LEARNING ENVIRONMENT

CSP Community Edition

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

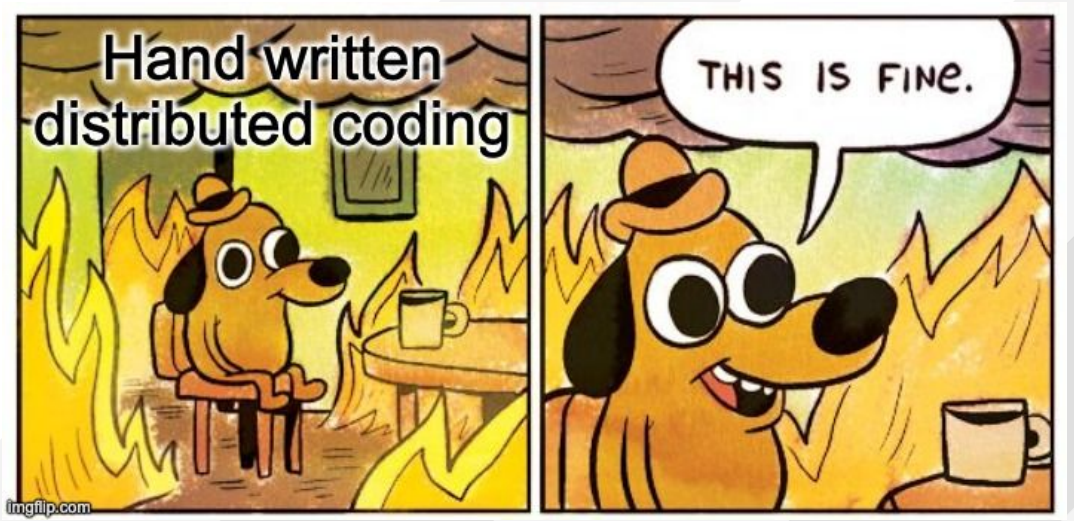
- Docker compose file of CSP to run from command line w/o any dependencies, including Flink, SQL Stream Builder, Kafka, Kafka Connect, Streams Messaging Manager and Schema Registry
  - `$> docker compose up`
- Licensed under the Cloudera Community License
- **Unsupported**
- Community Group Hub for CSP
- Find it on docs.cloudera.com under Applications

CSP Community Edition

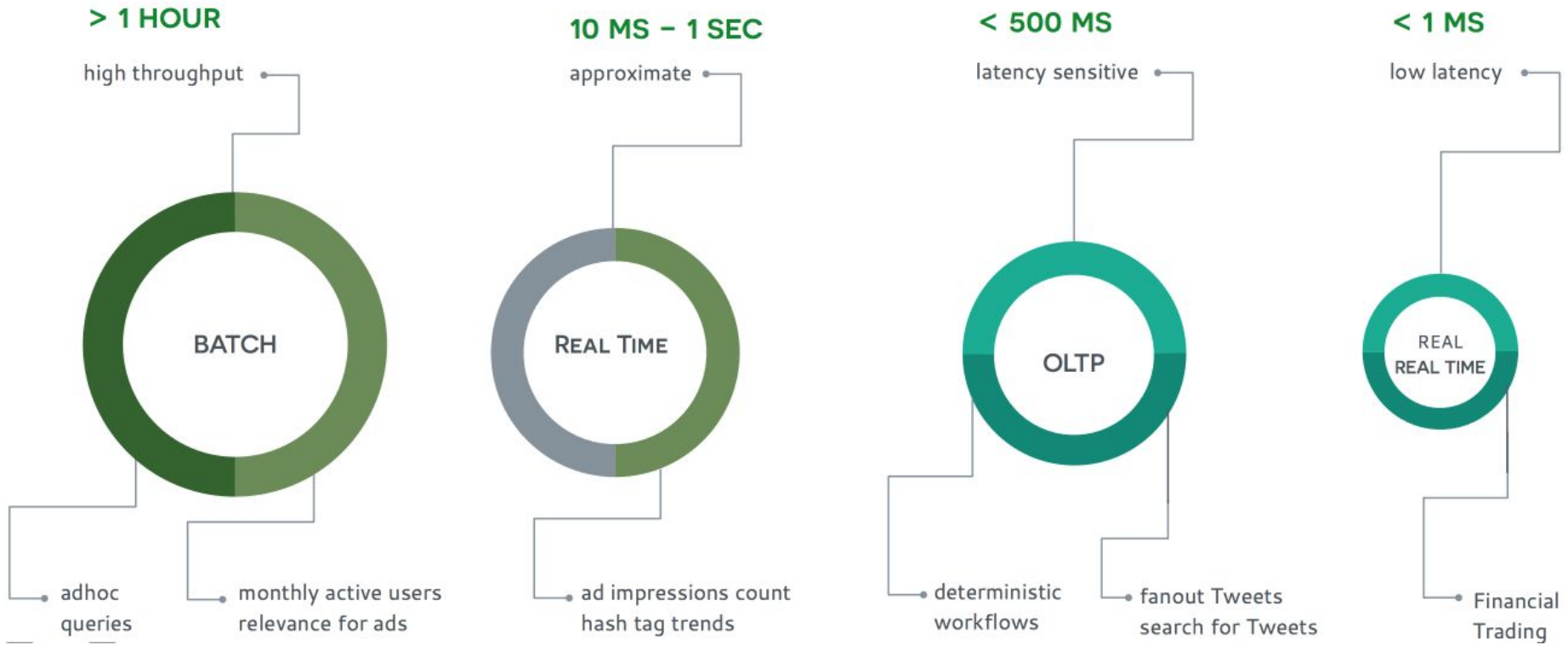
A readily available, dockerized deployment of Apache Kafka and Apache Flink that allows you to test the features and capabilities of Cloudera Stream Processing.

Learn More

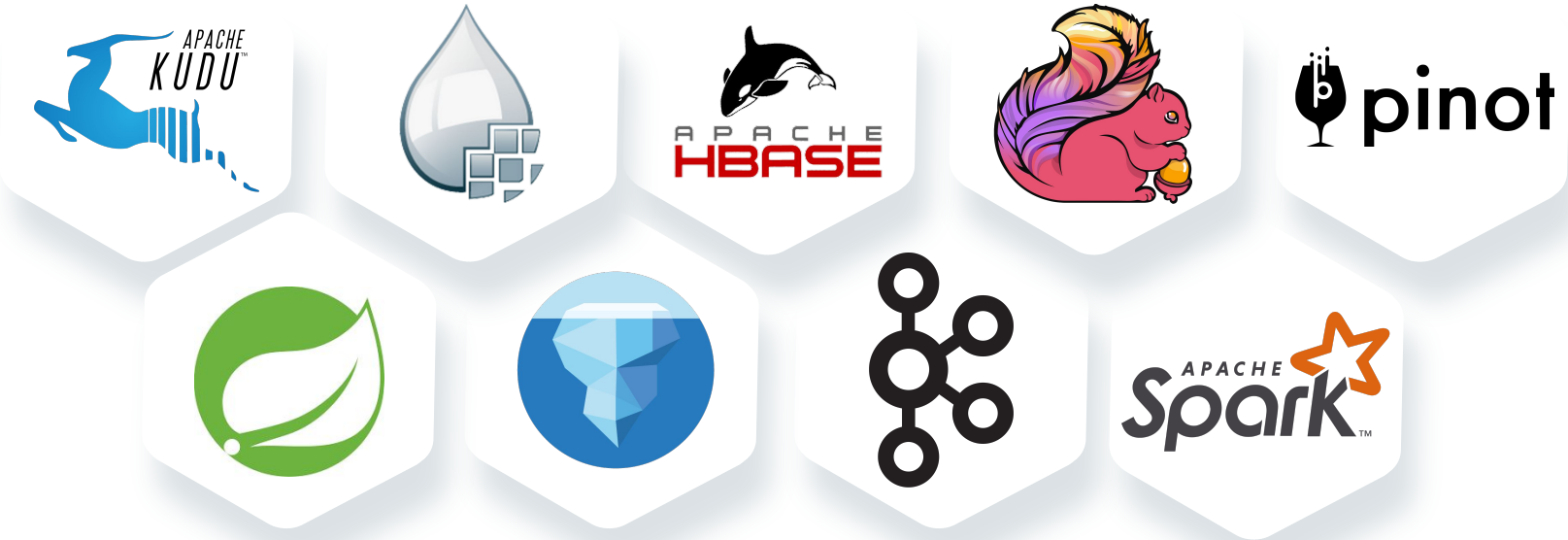
STREAMING

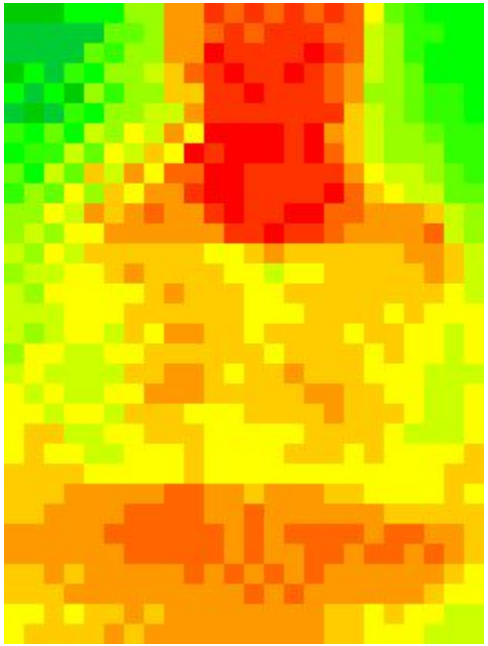


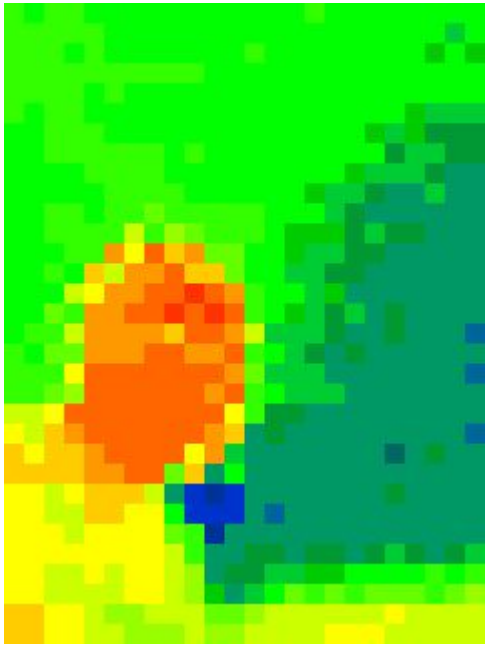
WHAT IS REAL-TIME?



BUILDING REAL-TIME REQUIRES A TEAM

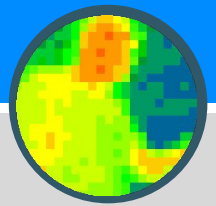






Streaming for Java Developers

Multiple users, frameworks, languages, devices, data sources & clusters

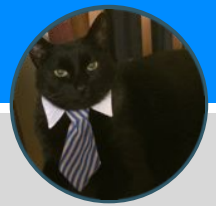


STREAMING ENGINEER

- Coding skills in Python, Java
- Experience with Apache Kafka or Pulsar
- Knowledge of database query languages such as SQL
- Knowledge of tools such as Apache Flink, Apache Spark and Apache NiFi

JAVA DEVELOPER

- Frameworks like Spring, Quarkus and micronaut
- Relational Databases, SQL
- Cloud
- Dev and Build Tools



CAT

- Expert in ETL (Eating, Ties and Laziness)
- Deep SME in Buzzwords
- No Coding Skills
- R&D into Lasers

AI

- Will Drive your Car?
- Will Fix Your Code?
- Will Beat You At Q-Bert
- Will Write my Next Talk

APACHE SPARK

APACHE SPARK

Data Engineering at Scale

- Multi-language support with Java, Scala, Python and R
- Batch and Microbatch
- Strong SQL support
- Machine Learning
- Jupyter and Apache Zeppelin notebook support

APACHE ICEBERG

A Flexible, Performant & Scalable Table Format

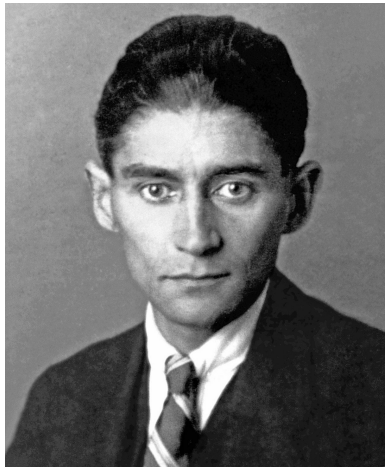
- Donated by Netflix to the Apache Foundation in 2018
- Flexibility
  - Hidden partitioning
  - Full schema evolution
- Data Warehouse Operations
  - Atomic Consistent Isolated Durable (ACID) Transactions
  - Time travel and rollback
- Supports best in class SQL performance
  - High performance at Petabyte scale

APACHE KAFKA

Yes, Franz, It's Kafka

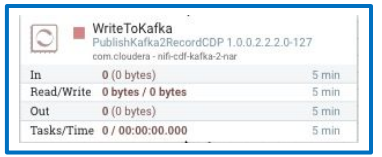
Let's do a metamorphosis on your data. Don't fear changing data.

You don't need to be a brilliant writer to stream data.

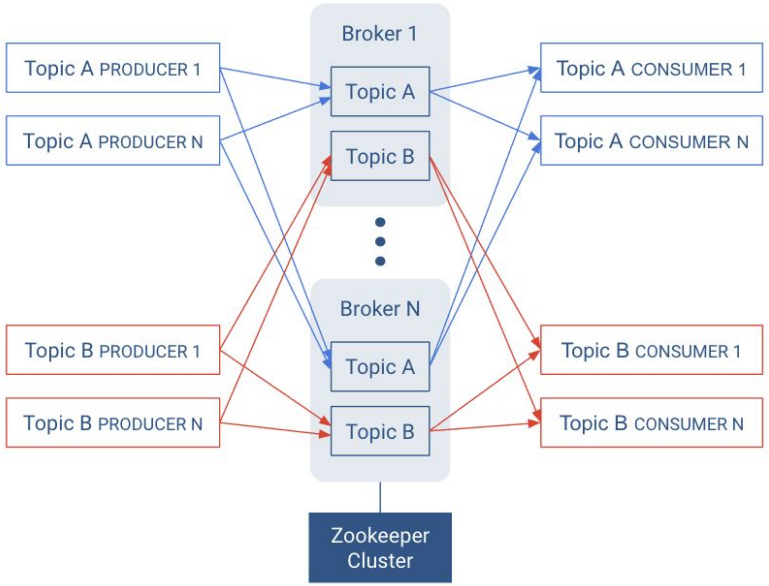


Franz Kafka was a German-speaking Bohemian novelist and short-story writer, widely regarded as one of the major figures of 20th-century literature. His work fuses elements of realism and the fantastic.
Wikipedia

STREAMS MESSAGING WITH KAFKA



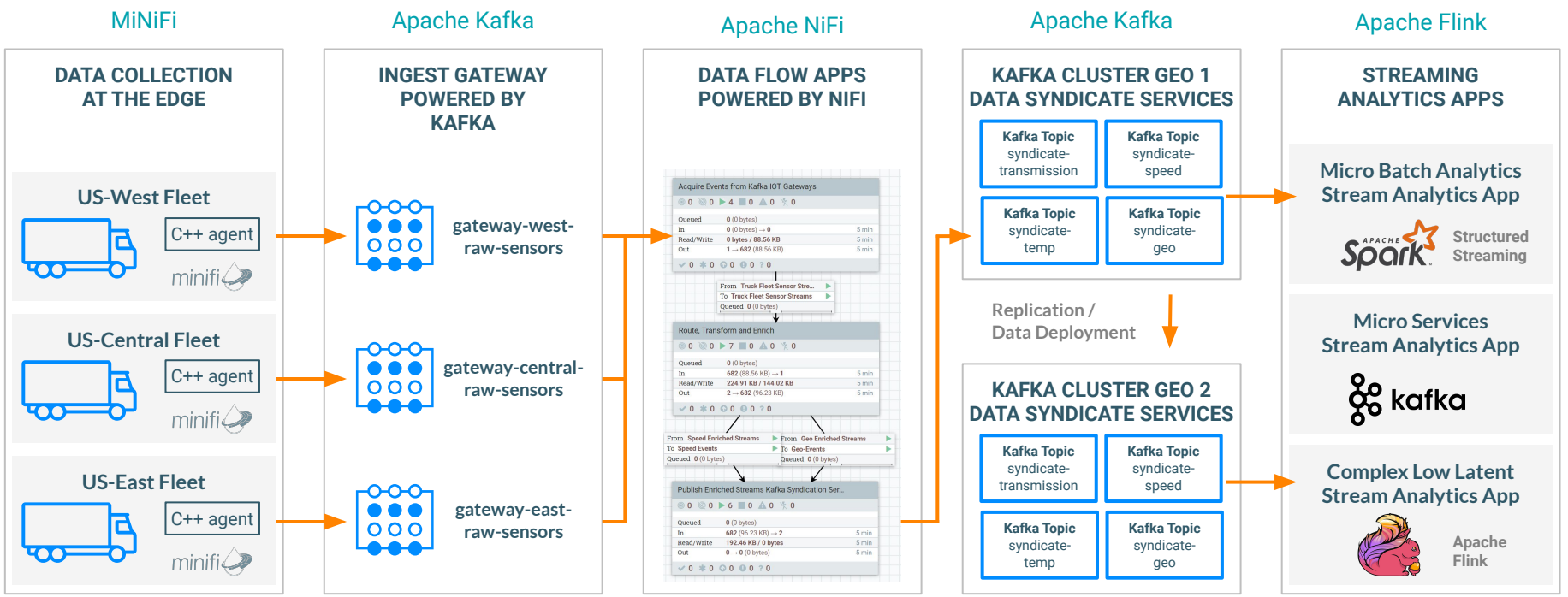
- Highly reliable distributed messaging system.
- Decouple applications, enables many-to-many patterns.
- Publish-Subscribe semantics.
- Horizontal scalability.
- Efficient implementation to operate at speed with big data volumes.
- Organized by topic to support several use cases.



What is Can You Do With Apache Kafka?

- Web site activity: track page views, searches, etc. in real time
- Events & log aggregation: particularly in distributed systems where messages come from multiple sources
- Monitoring and metrics: aggregate statistics from distributed applications and build a dashboard application
- Stream processing: process raw data, clean it up, and forward it on to another topic or messaging system
- Real-time data ingestion: fast processing of a very large volume of messages

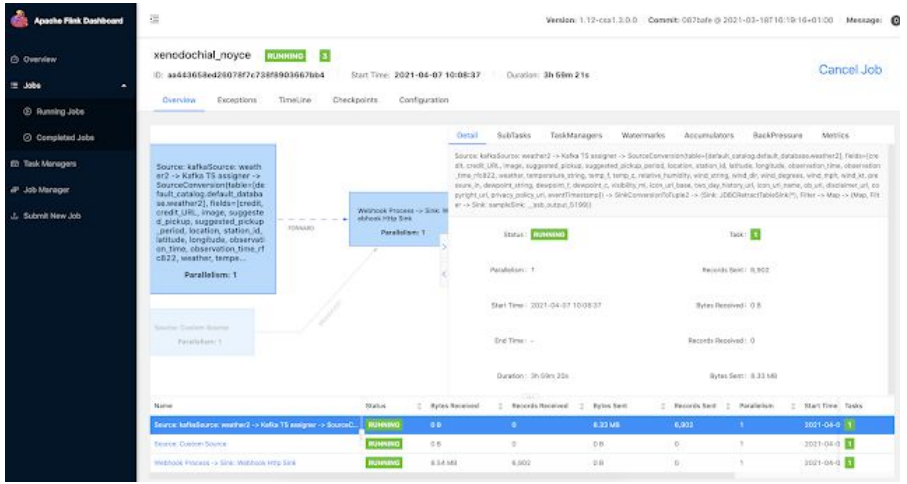
Apache Kafka



APACHE FLINK

Flink SQL

- Streaming Analytics
- Continuous SQL
- Continuous ETL
- Complex Event Processing
- Standard SQL Powered by Apache Calcite

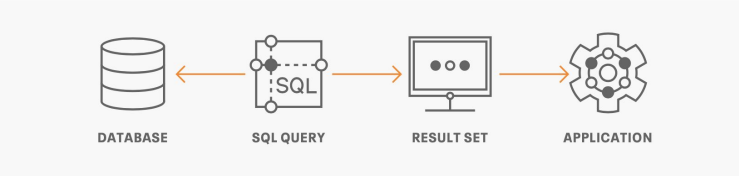


https://www.datainmotion.dev/2021/04/cloudera-sql-stream-builder-ssb-updated.html

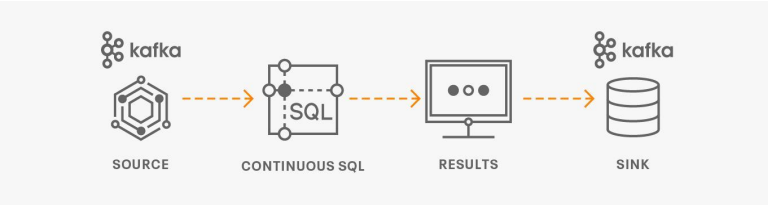
CONTINUOUS SQL

- 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)

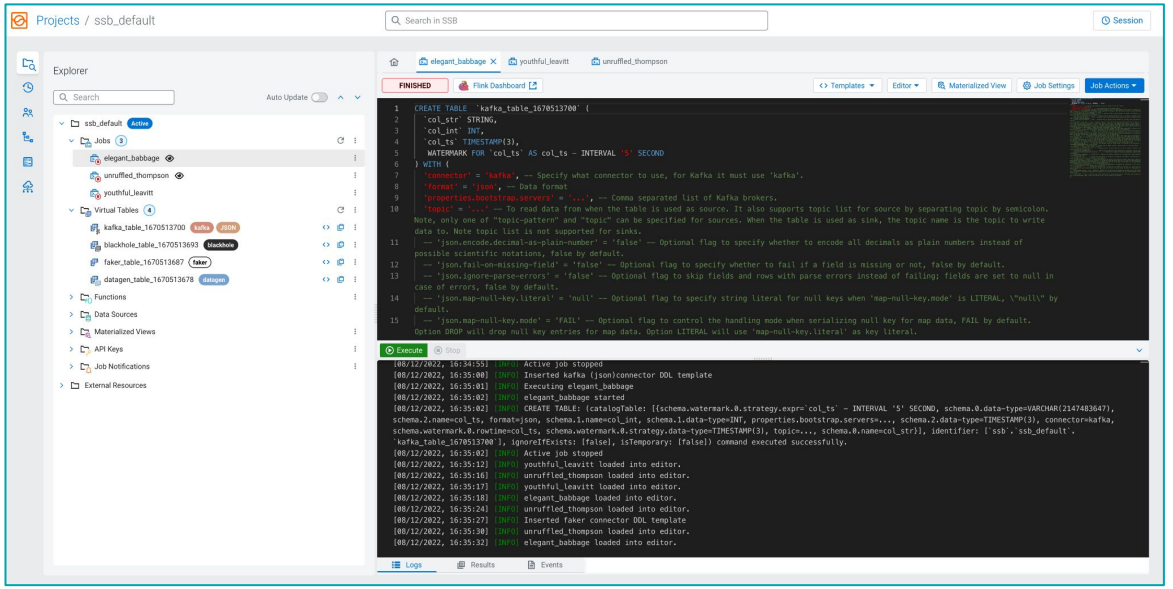
Democratize access to real-time data with just SQL

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

No Java or Scala code development required.

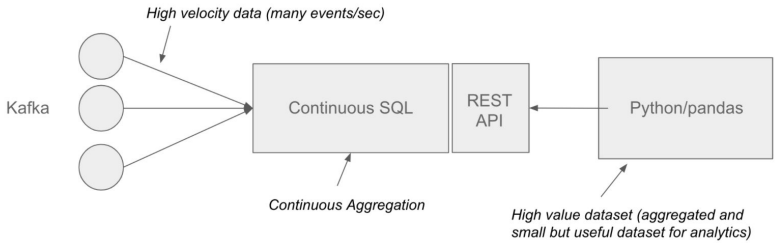
Simplifies access to data in Kafka & Flink. Connectors to batch data in HDFS, Kudu, Hive, S3, JDBC, CDC and more

Enrich streaming data with batch data in a single tool



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
```

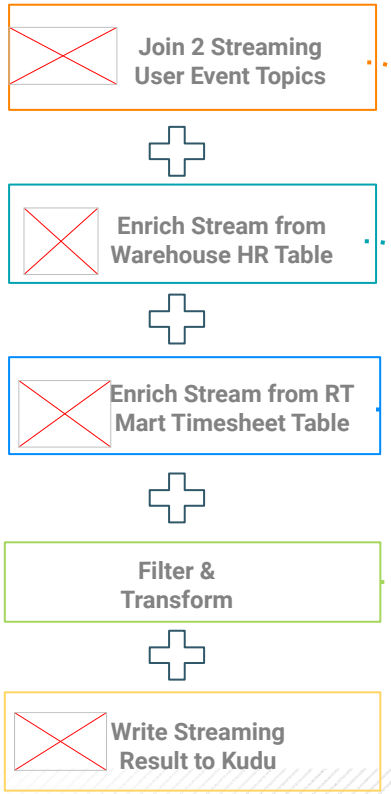
```
[90]: import pandas as pd
[91]: mv = "https://xxxyyyyzzz"
[92]: df = pd.read_json(mv)
[93]: len(df.keys())
[93]: 5
[95]: df['ts'] = pd.to_datetime(df['ts'])
[97]: df.dtypes
[97]: max_amount              int64
      sum_amount              int64
      thecount                int64
      ts             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	86761	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

```
[ ]:
```

Streaming ETL Data Pipeline Made Simple with SQL StreamBuilder

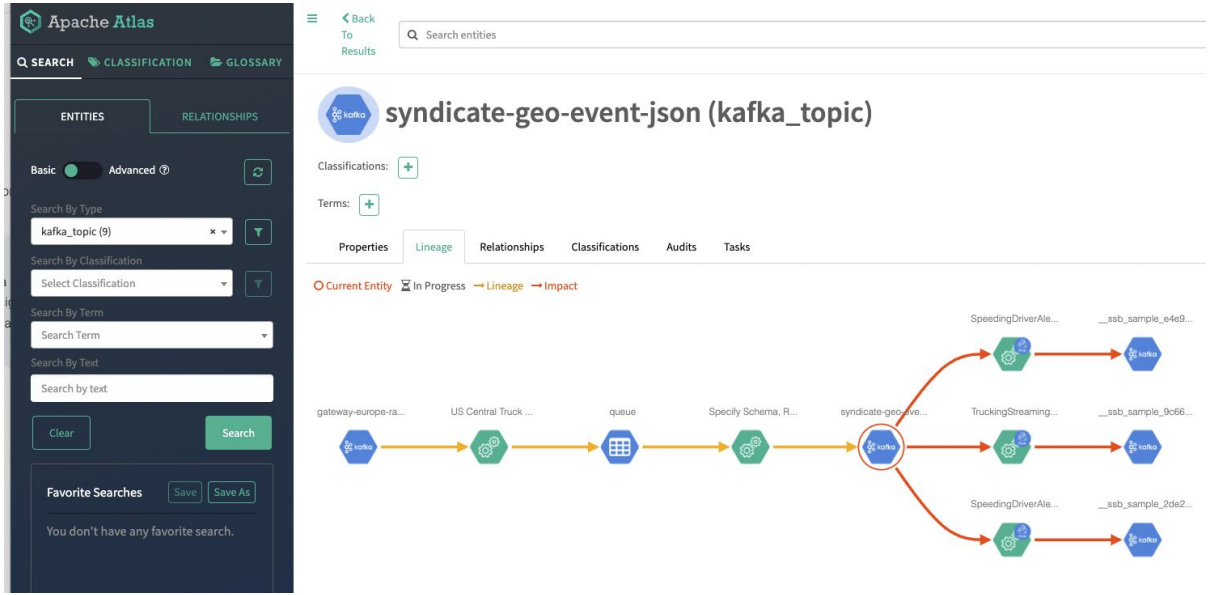



```
SELECT
      geo_event.eventTimestamp, geo_event.driverId, geo_event.eventTime,geo_event.eventSource,
      geo_event.truckId,geo_event.driverName,geo_event.routeId,geo_event.route,geo_event.eventType,
      geo_event.latitude, geo_event.longitude, geo_event.correlationId, geo_event.geoAddress,
      speed_event.speed,
      driver.certified, driver.wage_plan,
      timesheet.hours_logged, timesheet.miles_logged
 FROM
   geo_events_json as geo_event
   join speed_events_json as speed_event
   on (geo_event.driverId = speed_event.driverId)
   left join CDP_Hive_Catalog.employees_hr_hive_db.driver
      FOR  SYSTEM_TIME AS OF PROCTIME() driver
      on driver.driverid = geo_event.driverId
   left join `CDP_Kudu_Catalog`.`default_database`.`impala::employees_hr_kudu_impala_db.timesheet`
      FOR  SYSTEM_TIME AS OF PROCTIME() timesheet
      on (timesheet.driverid = geo_event.driverId  and timesheet_week = 1)
    where
   geo_event.eventTimestamp BETWEEN
      speed_event.eventTimestamp - INTERVAL '1' SECOND AND
      speed_event.eventTimestamp + INTERVAL '1' SECOND
   AND geo_event.eventType <> 'Normal'
   AND driver.wage_plan = 'hours'
   AND timesheet.hours_logged > 45
```

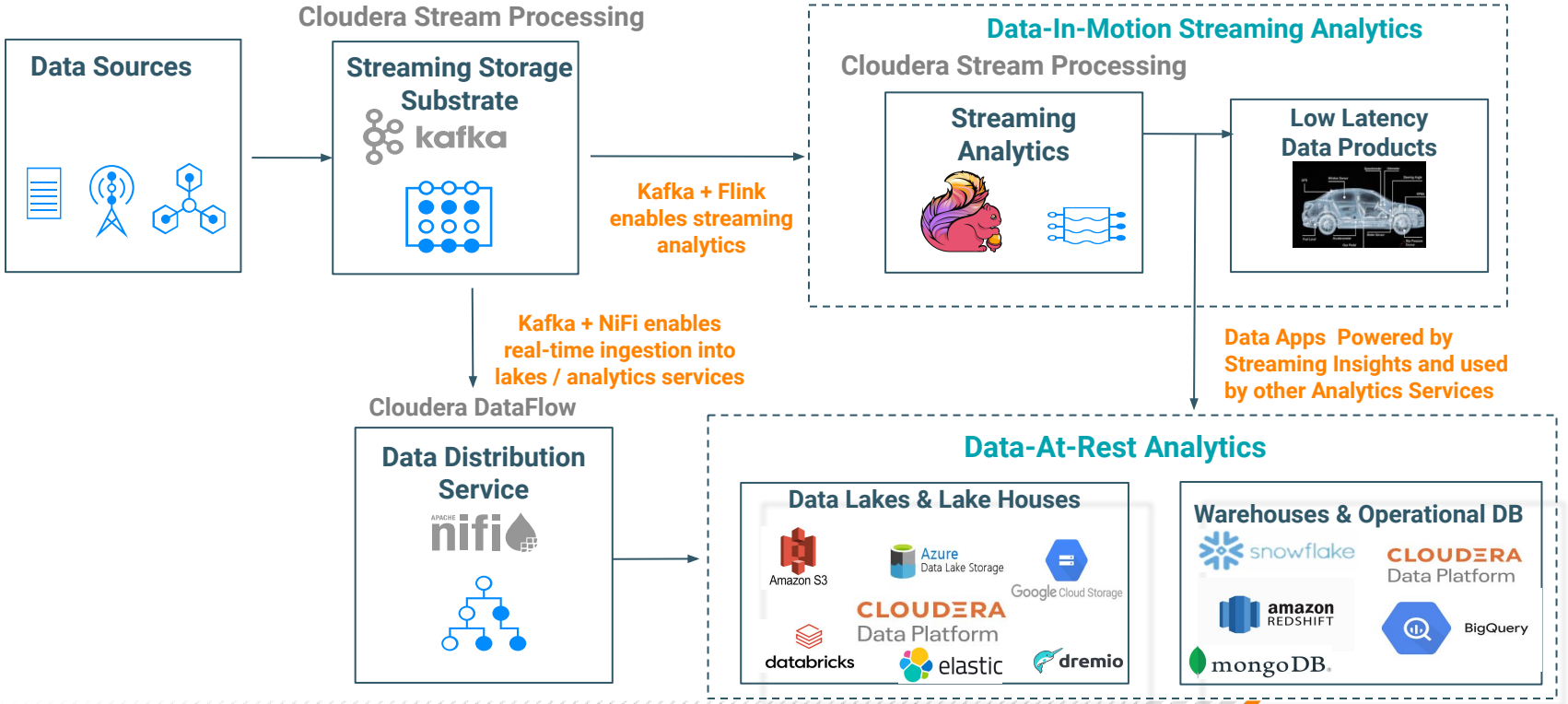
DATA GOVERNANCE FOR THE ENTIRE STREAMING PIPELINE

Streaming Data Lineage with SDX

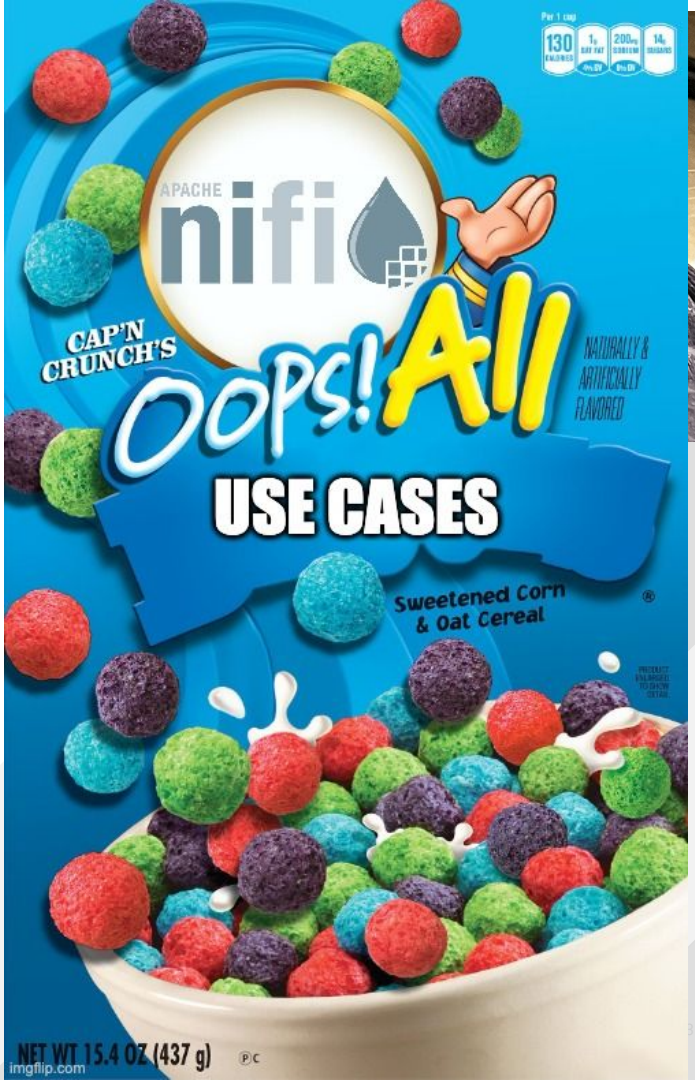
- Track Consumer, Producer, Topics and Consumer Group Lineage
- No changes required to Consumers or Producers
- End-To-End lineage from consumer to producer



Moving Beyond Draining of Streams Into Lakes: Analytics-in-Stream

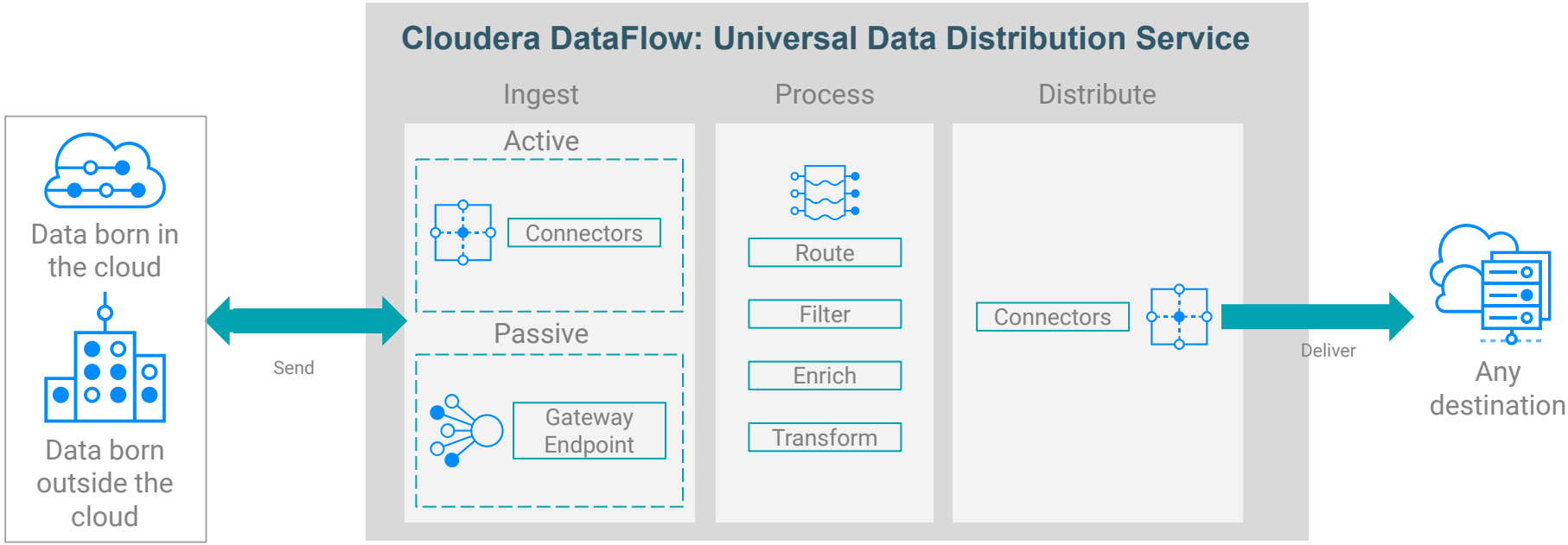


DATAFLOW APACHE NIFI



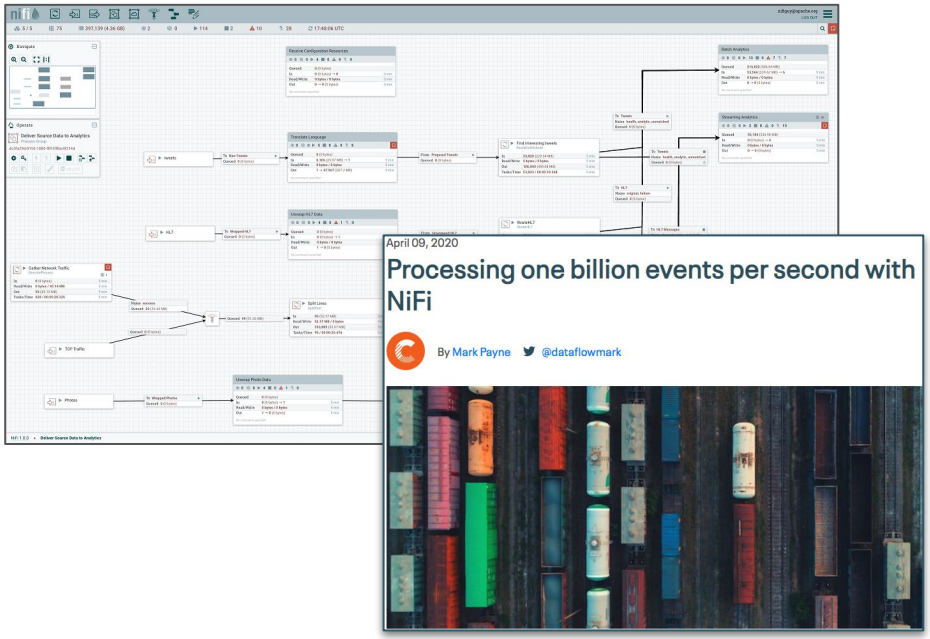
UNIVERSAL DATA DISTRIBUTION WITH CLOUDERA DATAFLOW (CDF)

Connect to Any Data Source Anywhere then Process and Deliver to Any Destination



CLOUDERA DATAFLOW - POWERED BY APACHE NiFi

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

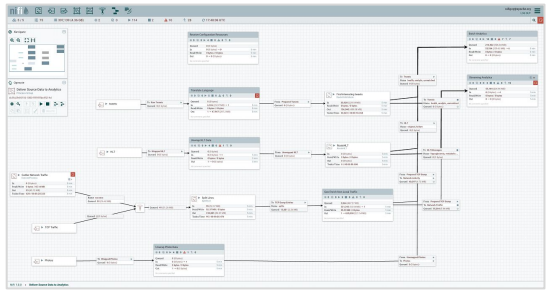


- #1 data ingestion/movement engine
- Strong community
- Product maturity over 11 years
- Deploy on-premises or in the cloud
- Over 400+ pre-built processors
- Built-in data provenance
- Guaranteed delivery
- Throttling and Back pressure

CLOUDERA FLOW AND EDGE MANAGEMENT

Enable easy ingestion, routing, management and delivery of any data anywhere *(Edge, cloud, data center)* to any downstream system with built in end-to-end security and provenance

minifi APACHE nifi



APACHE NIFI registry

Advanced tooling to industrialize flow development *(Flow Development Life Cycle)*

ACQUIRE

kafka, HIVE, Spark, APACHE STORM, Solr, SQL, Flink, DynamoDB, cassandra, amazon KINESIS, elastic, amazon S3, mongoDB, slack

FTP
SFTP
HL7
UDP
XML
HTTP
EMAIL
HTML
IMAGE
SYSLOG

PROCESS

HASH	EVALUATE
MERGE	EXECUTE
EXTRACT	GEOENRICH
DUPLICATE	SCAN
SPLIT	REPLACE
ENCRYPT	TRANSLATE
TALL	CONVERT
ROUTE TEXT	
ROUTE CONTENT	
ROUTE CONTEXT	
ROUTE RATE	
DISTRIBUTE LOAD	

DELIVER

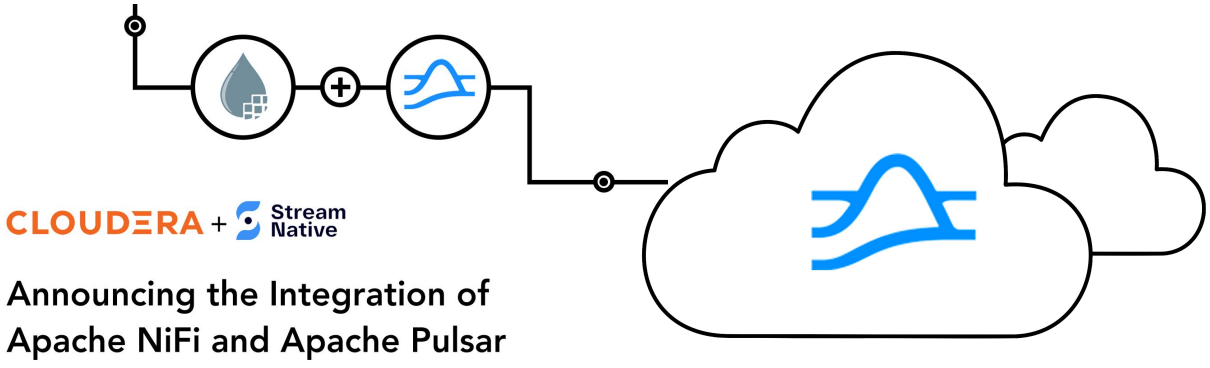
FTP
SFTP
HL7
UDP
XML
HTTP
EMAIL
HTML
IMAGE
SYSLOG

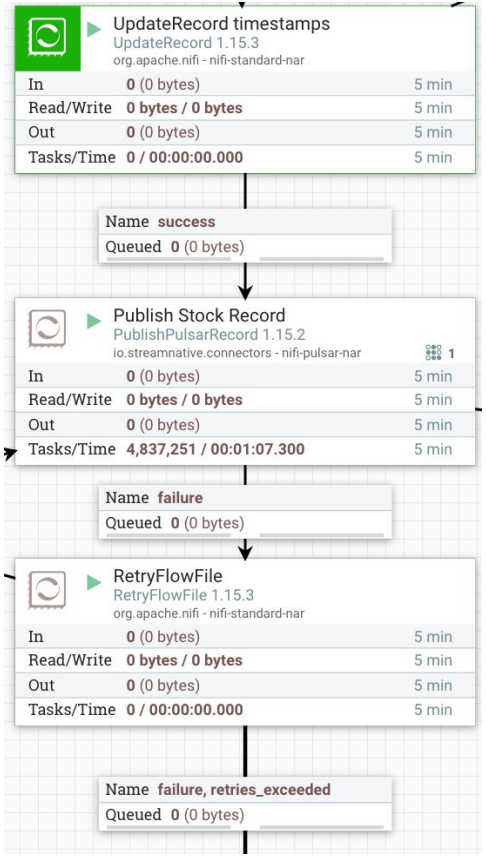
kafka, HIVE, Spark, APACHE STORM, Solr, SQL, Flink, DynamoDB, cassandra, amazon KINESIS, elastic, amazon S3, mongoDB, slack

- Over 300 Prebuilt Processors
- Easy to build your own
- Parse, Enrich & Apply Schema
- Filter, Split, Merger & Route
- Throttle & Backpressure

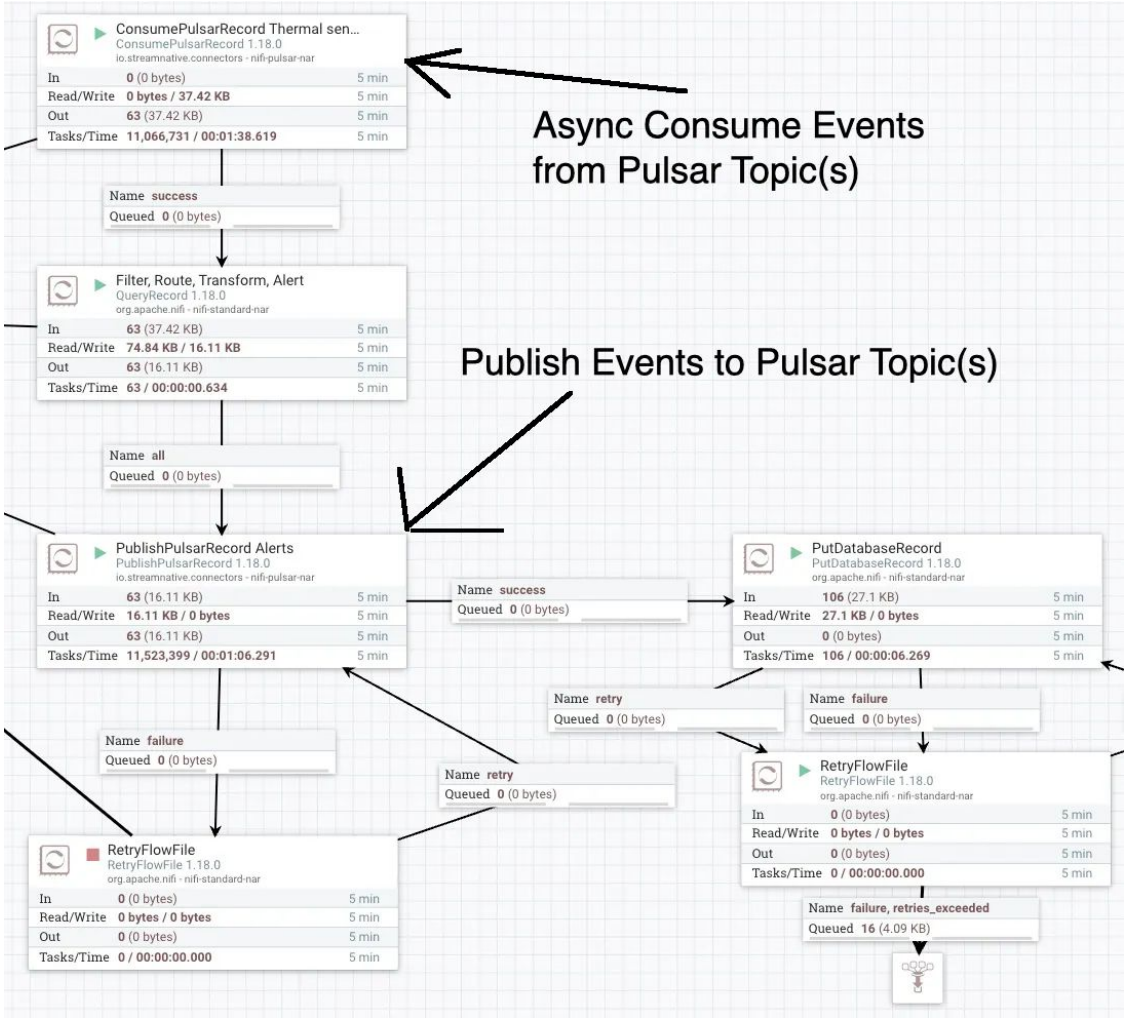
- Guaranteed Delivery
- Full data provenance from acquisition to delivery
- Diverse, Non-Traditional Sources
- Eco-system integration

Apache NiFi Pulsar Connector



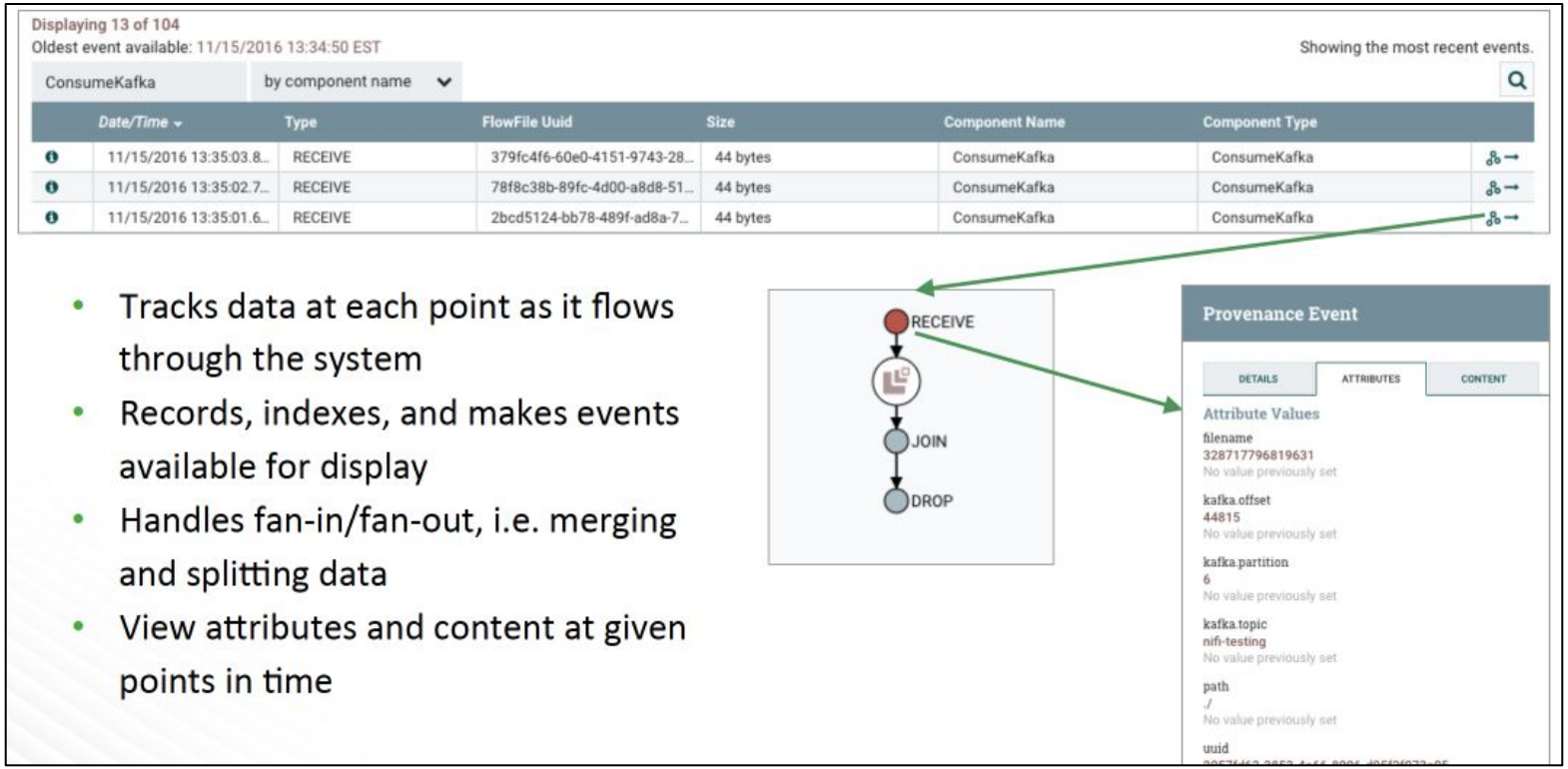


https://streamnative.io/apache-nifi-connector/



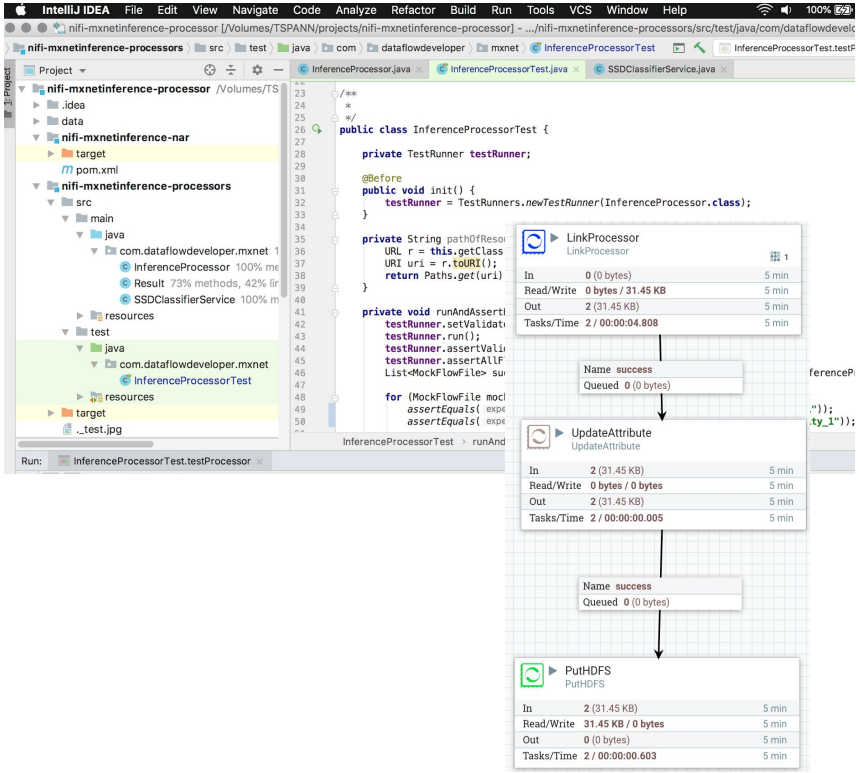
PULSAR SUMMIT

PROVENANCE



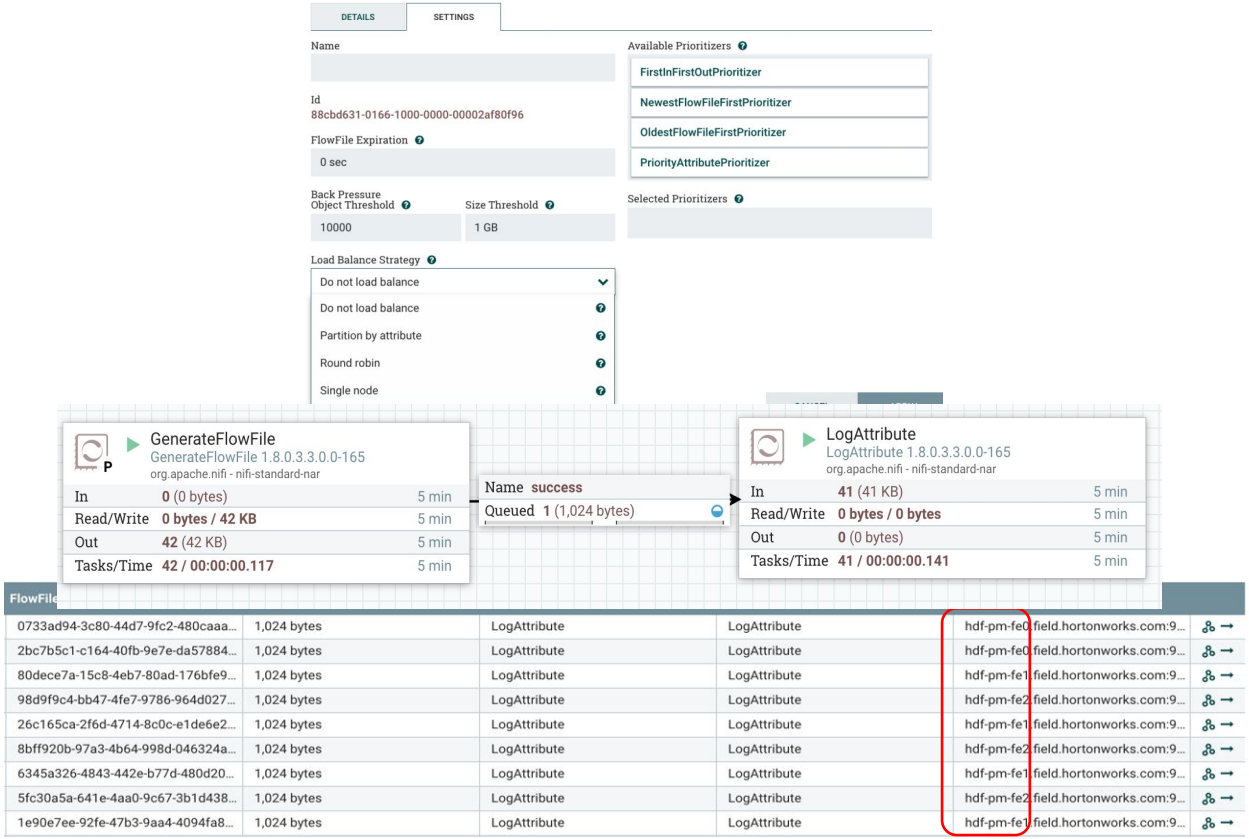
EXTENSIBILITY

- Built from the ground up with extensions in mind
- Service-loader pattern for…
  - Processors
  - Controller Services
  - Reporting Tasks
  - Prioritizers
- Extensions packaged as NiFi Archives (NARs)
  - Deploy NiFi lib directory and restart
  - Same model as standard components

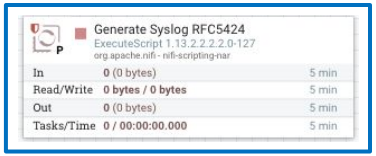


NiFi Load Balancing

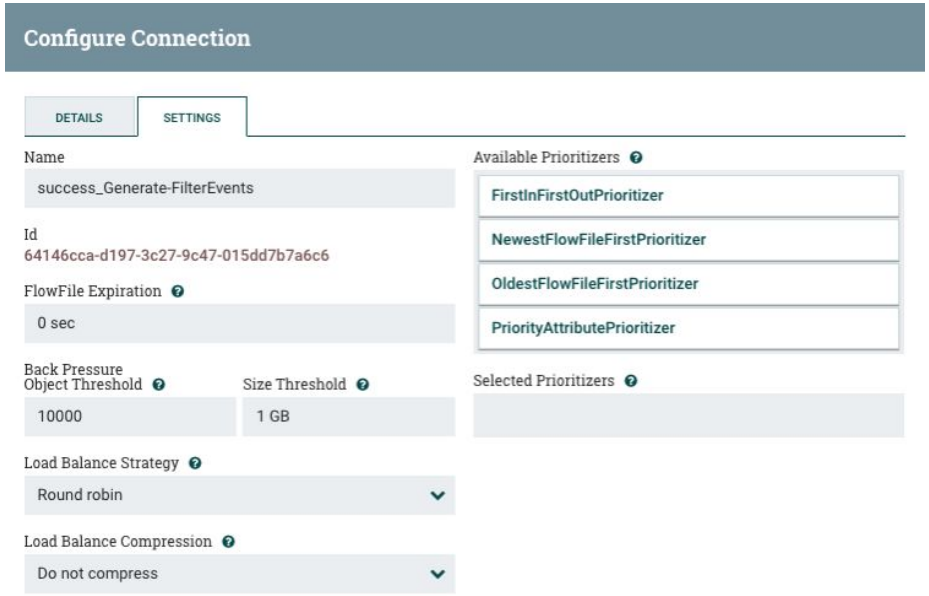
- Improve NiFi cluster throughput
- Defined at connection level
- Configurable balancing strategies
- Critical for scale up paradigm in Kubernetes
- Alleviates S2S balancing “hack” customers use



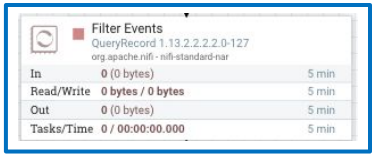
QUEUE CONFIGURATION



- **FlowFile Expiration -** Data that cannot be processed in a timely fashion can be automatically removed from the flow.
- **Back Pressure Thresholds -** Thresholds indicate how much data should be allowed to exist in the queue before the component that is the source of the Connection is no longer scheduled to run. This allows the system to avoid being overrun with data.
- **Load Balance Strategy –** Strategy to distribute the data in a flow across the nodes in the cluster. When enabled, compression can be configured on FlowFile contents and attributes.
- **Prioritization –** Determines the order in which flow files are processed.

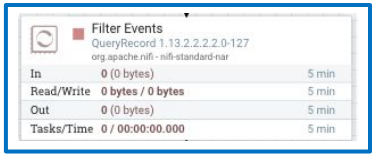


RECORD-ORIENTED DATA WITH NIFI

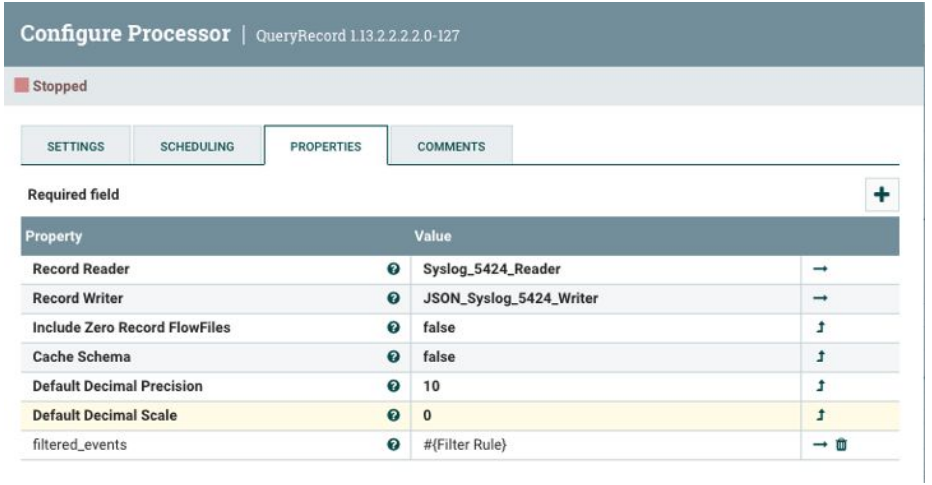


- **Record Readers** - Avro, CSV, Grok, IPFIX, JSAN1, JSON, Parquet, Scripted, Syslog5424, Syslog, WindowsEvent, XML
- **Record Writers** - Avro, CSV, FreeFromText, Json, Parquet, Scripted, XML
- Record Reader and Writer support referencing a schema registry for retrieving schemas when necessary.
- Enable 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.

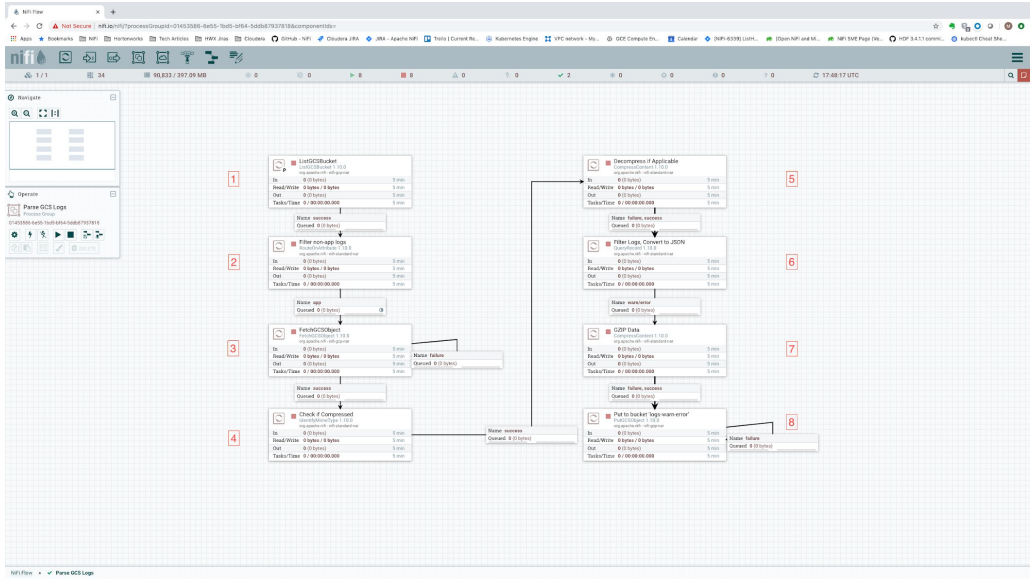
RUNNING SQL ON FLOWFILES



- Evaluates one or more SQL queries against the contents of a FlowFile.
- This can be used, for example, for field-specific filtering, transformation, and row-level filtering.
- Columns can be renamed, simple calculations and aggregations performed.
- The SQL statement must be valid ANSI SQL and is powered by Apache Calcite.



Processing one million events per second with NiFi

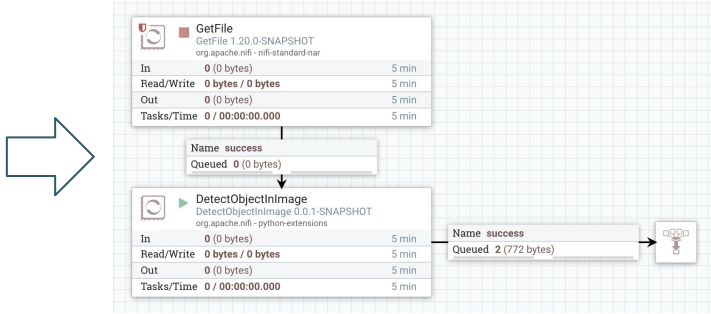


Nodes	Data rate/sec	Events/sec	Data rate/day	Events/day
1	192.5 MB	946,000	16.6 TB	81.7 Billion
5	881 MB	4.97 Million	76 TB	429.4 Billion
25	5.8 GB	26 Million	501 TB	2.25 Trillion
100	22 GB	90 Million	1.9 PB	7.8 Trillion
150	32.6 GB	141.3 Million	2.75 PB	12.2 Trillion

Apache NiFi with Python Custom Processors

Python as a 1st 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 = 0.5

class DetectObjectInImage:
    class Java:
        implements = ['org.apache.nifi.python.processor.FlowFileTransform']
    class ProcessorDetails:
        version = '0.0.1-SNAPSHOT'
        dependencies = ['numpy >= 1.23.5', 'opencv-python >= 4.6']

    def __init__(self, jvm=None, **kwargs):
        self.jvm = jvm

        # Build Property Descriptors
        self.model_file = PropertyDescriptor(
            name = 'Model File',
            description = 'The binary file containing the trained Deep Neural Network weights. Supports Caffe (*.caffemodel), TensorFlow (*.pb), Torch (*.t7, *.net), Darknet (*.weights), ' +
                          'DLDT (*.bin), and ONNX (*.onnx)',
            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 names from text file
        class_name_file = context.getProperty(self.class_name_file.name).getValue()
        if class_name_file is None:
```

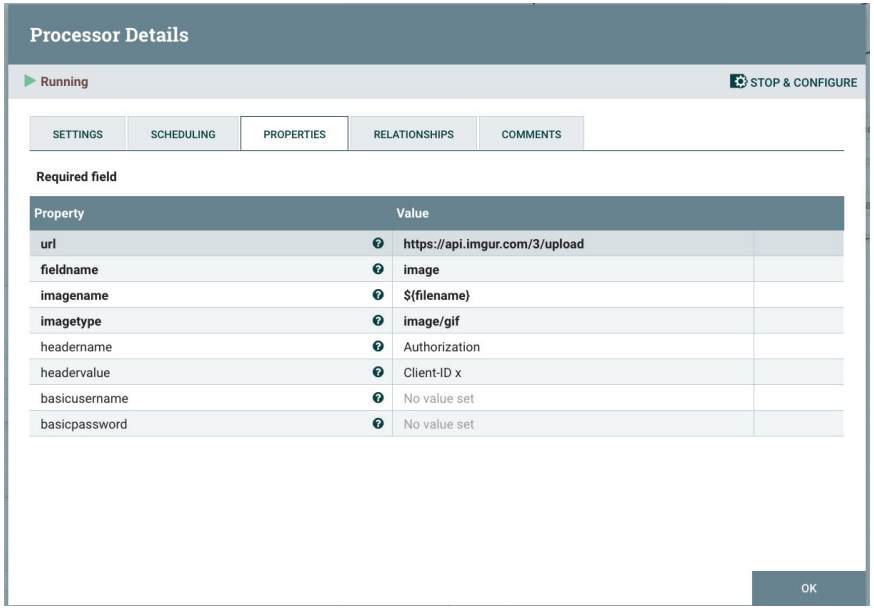


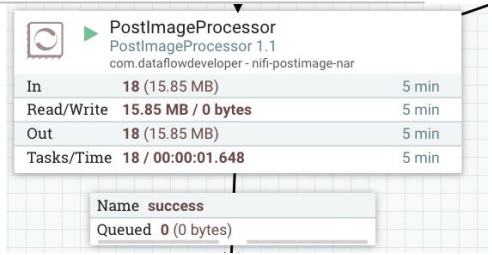
CUSTOM JAVA APPS

Custom Processors

https://github.com/tspannhw/nifi-extracttext-processor
https://github.com/tspannhw/nifi-tensorflow-processor
https://github.com/tspannhw/nifi-nlp-processor
https://github.com/tspannhw/nifi-convertjsontoddl-processor
https://github.com/tspannhw/nifi-corenlp-processor
https://github.com/tspannhw/nifi-imageextractor-processor
https://github.com/tspannhw/nifi-attributecleaner-processor
https://github.com/tspannhw/linkextractorprocessor
https://github.com/tspannhw/GetWebCamera
https://github.com/tspannhw/nifi-langdetect-processor
https://github.com/tspannhw/nifi-postimage-processor

Custom Processors in Java





Name
PostImageProcessor

Id
f62ebd85-3615-351b-6e72-812c68e68cf8

Type
PostImageProcessor 1.1

Bundle
com.dataflowdeveloper - nifi-postimage-nar

Penalty Duration
30 sec

Yield Duration
1 sec

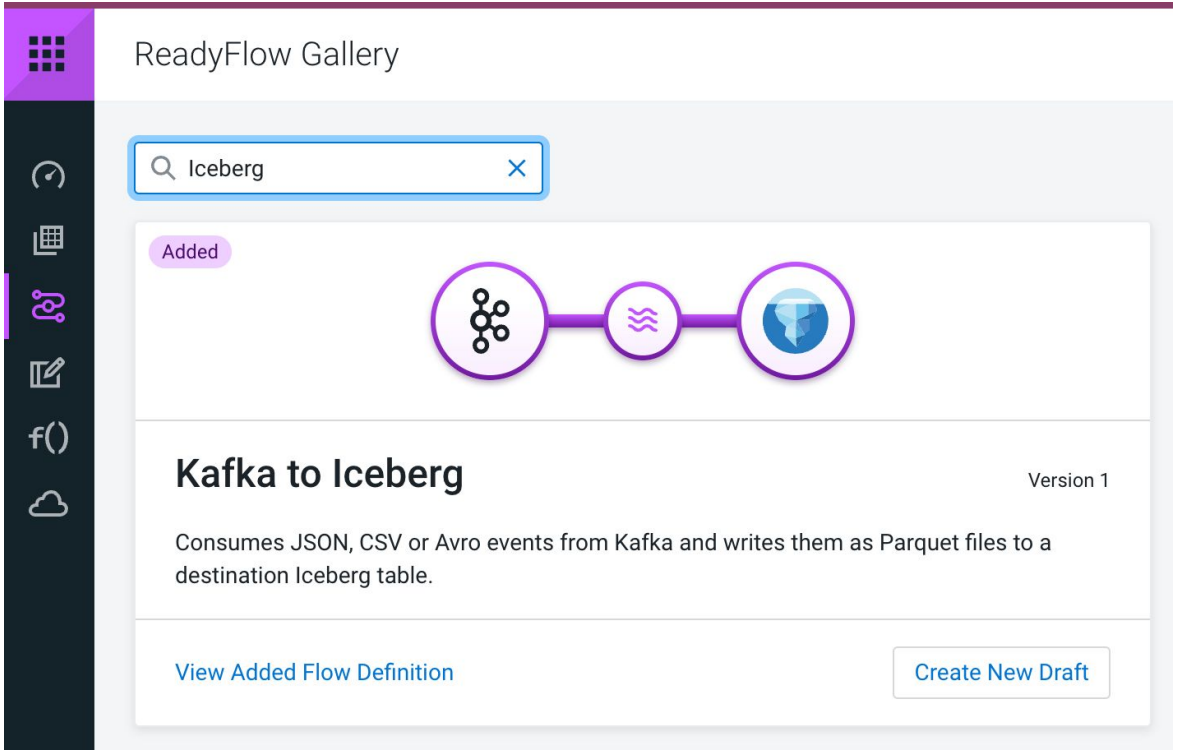
Bulletin Level
WARN

https://www.datainmotion.dev/2019/03/getting-started-with-custom-processor.html
https://www.datainmotion.dev/2019/03/posting-images-to-imgur-via-apache-nifi.html

```
public class TensorFlowProcessor extends AbstractProcessor {
        public static final PropertyDescriptor MODEL_DIR = new
PropertyDescriptor.Builder().name(MODEL_DIR_NAME)
                        .description("Model Directory").required(true).expressionLanguageSupported(true)
                        .addValidator(StandardValidators.NON_EMPTY_VALIDATOR).build();

@Override
        public void onTrigger(final ProcessContext context, final ProcessSession session) throws
ProcessException {
                FlowFile flowFile = session.get();
                if (flowFile == null) {
                        flowFile = session.create();
                }
                try {
                        flowFile.getAttributes();

}
```

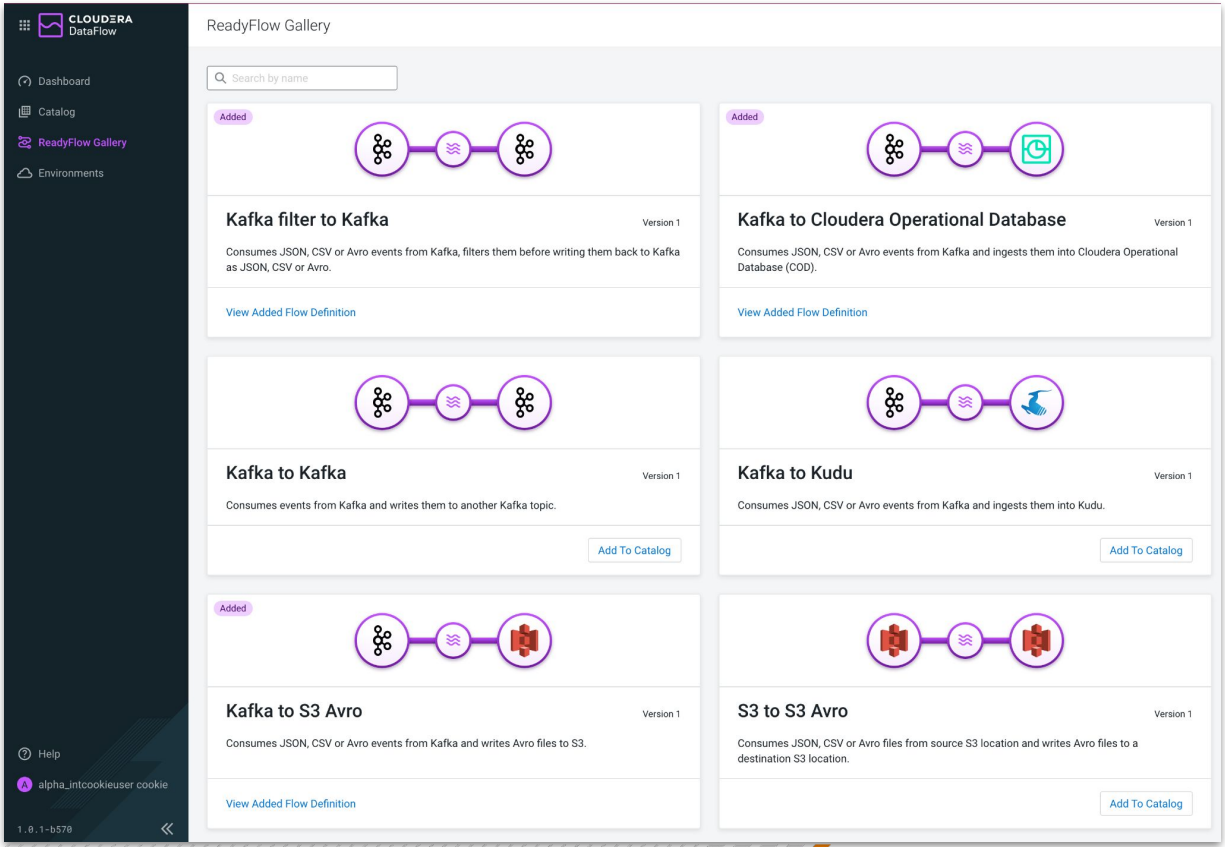
https://github.com/tspannhw/nifi-tensorflow-processor

CLOUDERA FLOW DESIGNER



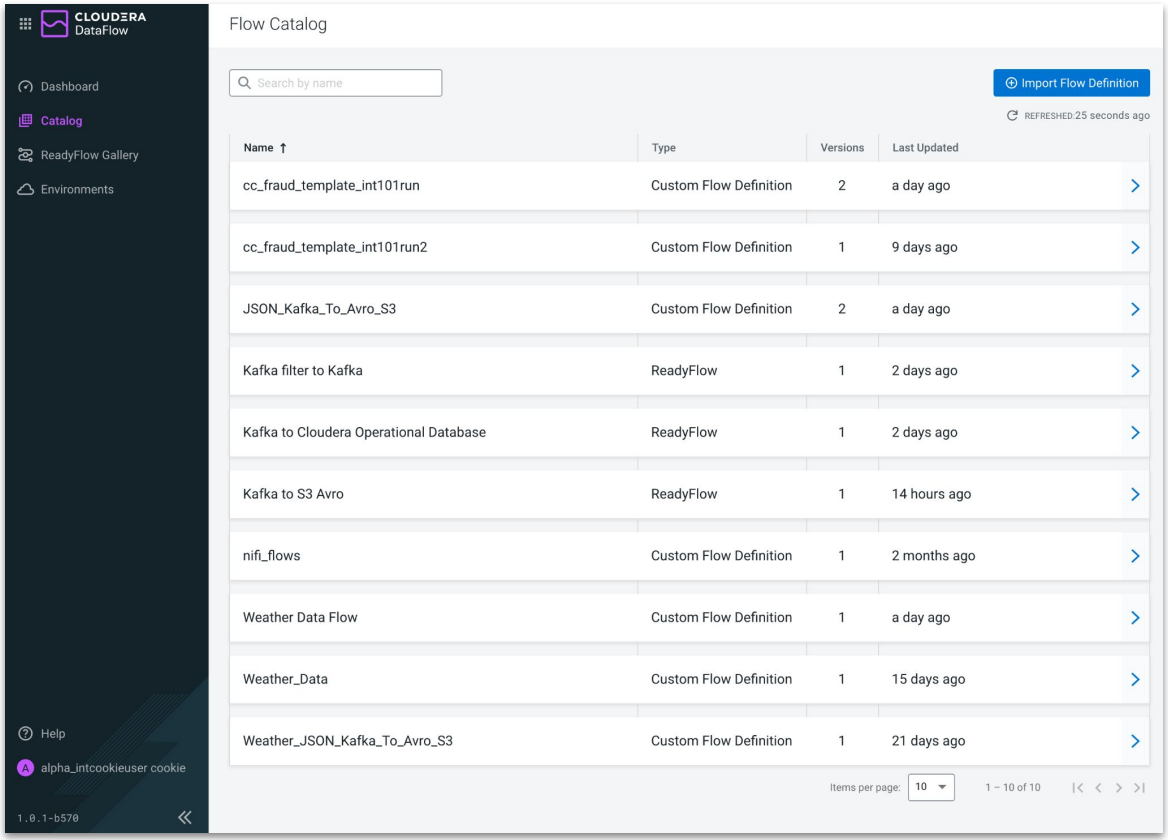
READYFLOW GALLERY

- 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



FLOW CATALOG

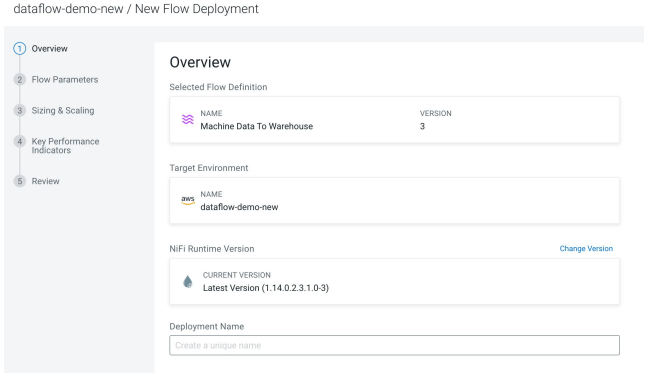
- Central repository for flow definitions
- Import existing NiFi flows
- Manage flow definitions
- Initiate flow deployments



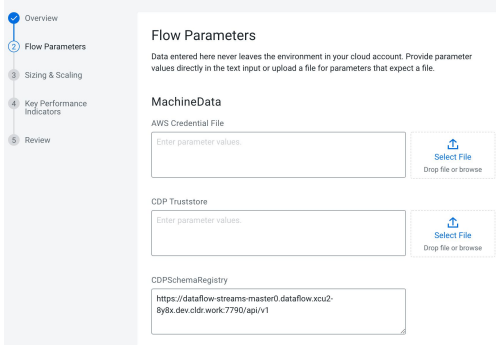
DEPLOYMENT WIZARD

- Turns flow definitions into flow deployments
- Guides users through providing required configuration
- Choose NiFi runtime version
- Pick from pre-defined NiFi node sizes
- Define KPIs for the deployment

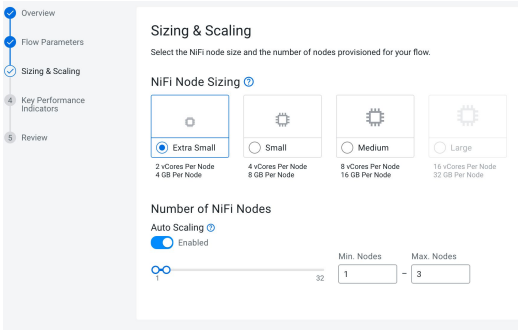
Start Deployment Wizard



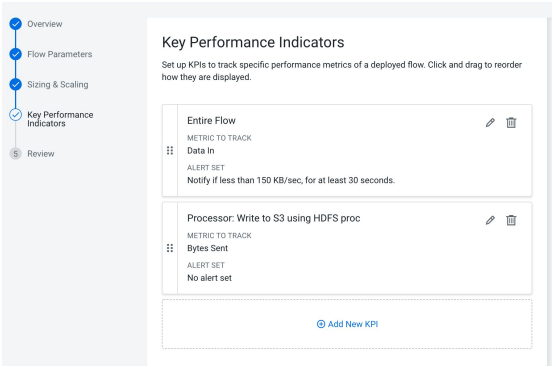
Provide Parameters



Configure Sizing & Scaling



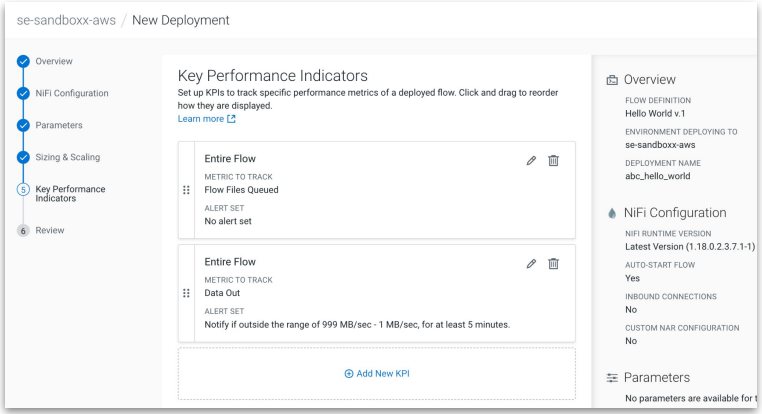
Define KPIs



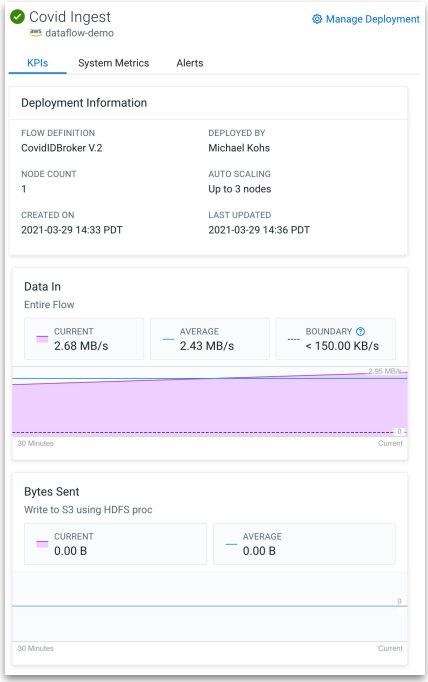
KEY PERFORMANCE INDICATORS

- Visibility into flow deployments
- Track high level flow performance
- Track in-depth NiFi component metrics
- Defined in Deployment Wizard
- Monitoring & Alerts in Deployment Details

KPI Definition in Deployment Wizard

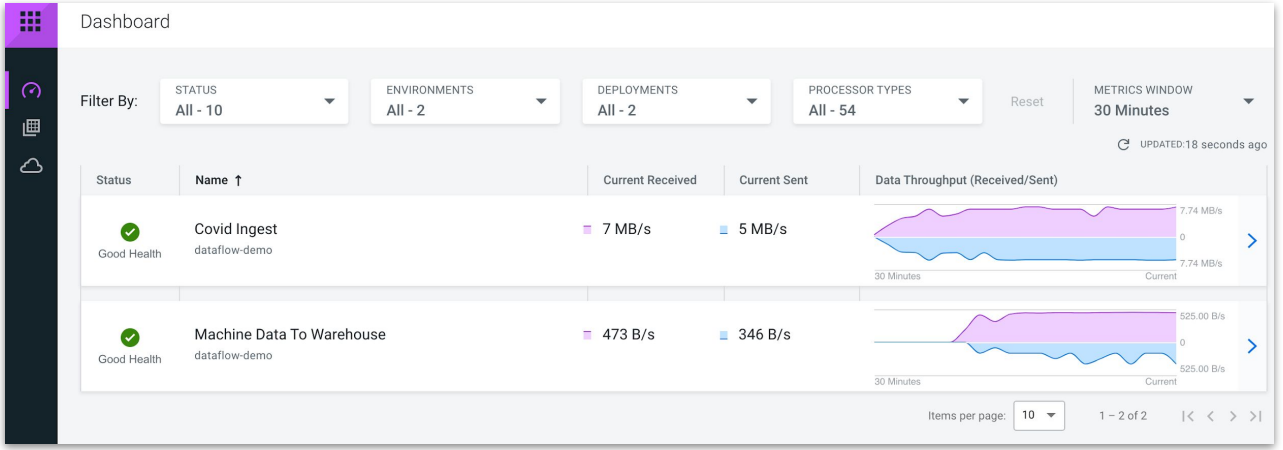


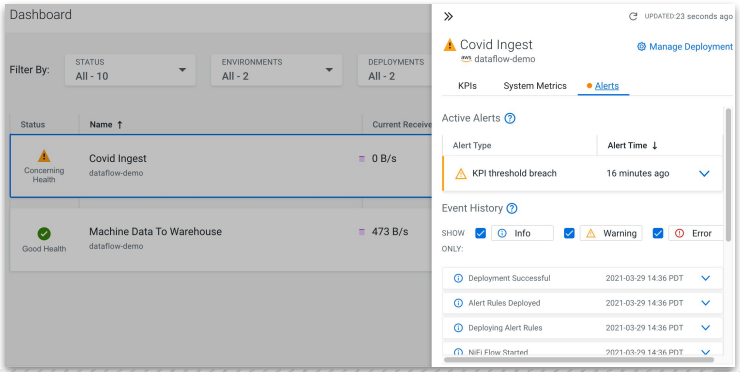
KPI Monitoring

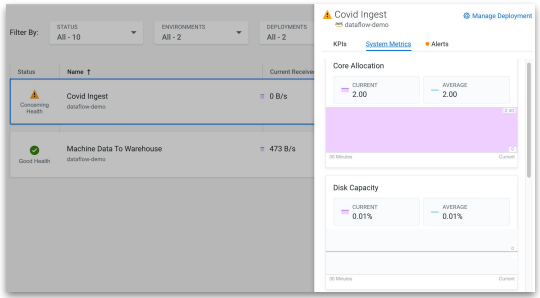


DASHBOARD

- 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

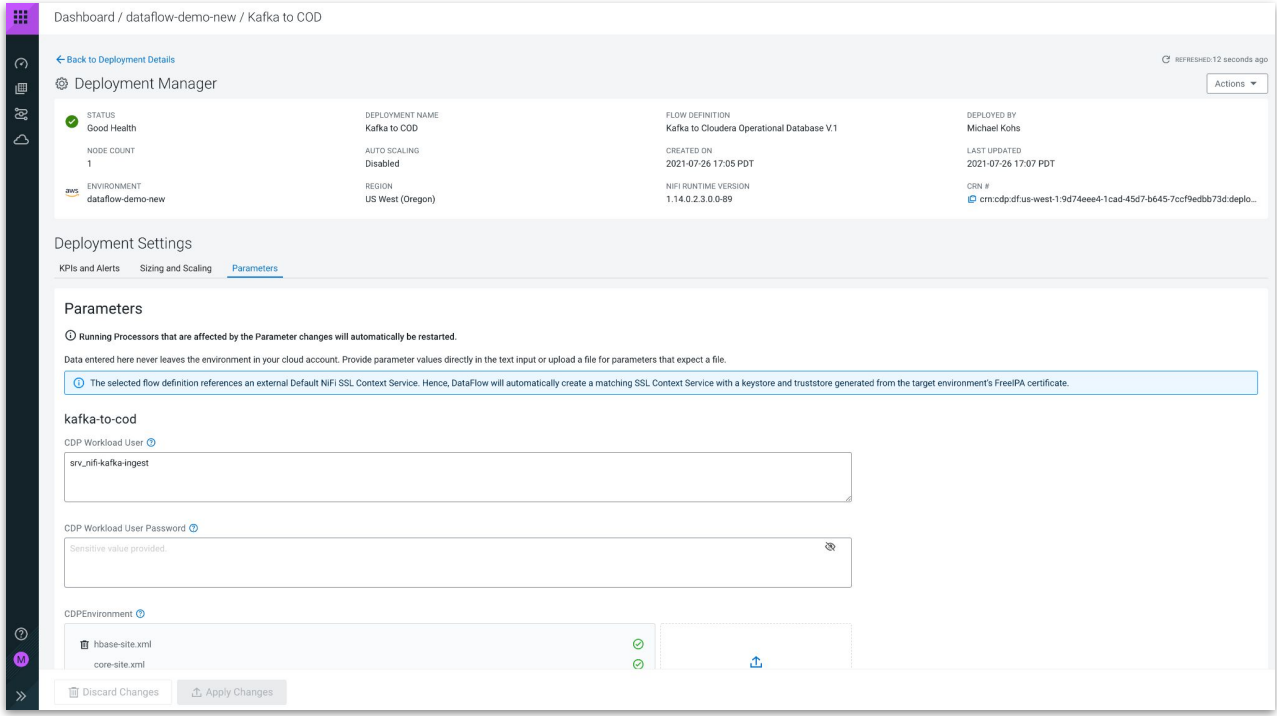






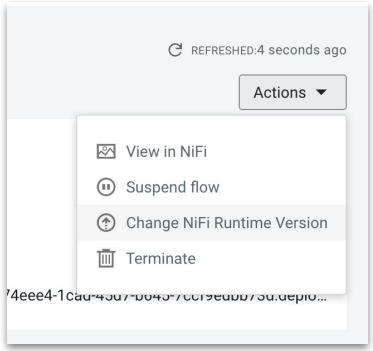
DEPLOYMENT MANAGER

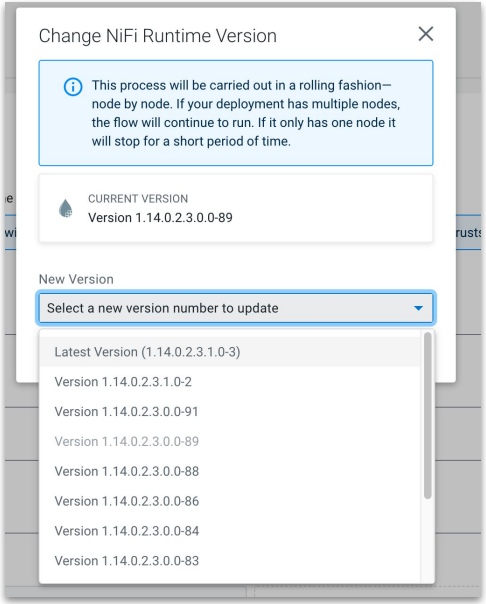
- Manage flow deployment lifecycle (Suspend/Start/Terminate)
- Add/Edit KPIs
- Change sizing configuration
- Update parameters
- Change NiFi version of the deployment
- Gateway to NiFi canvas



NIFI VERSION UPGRADES

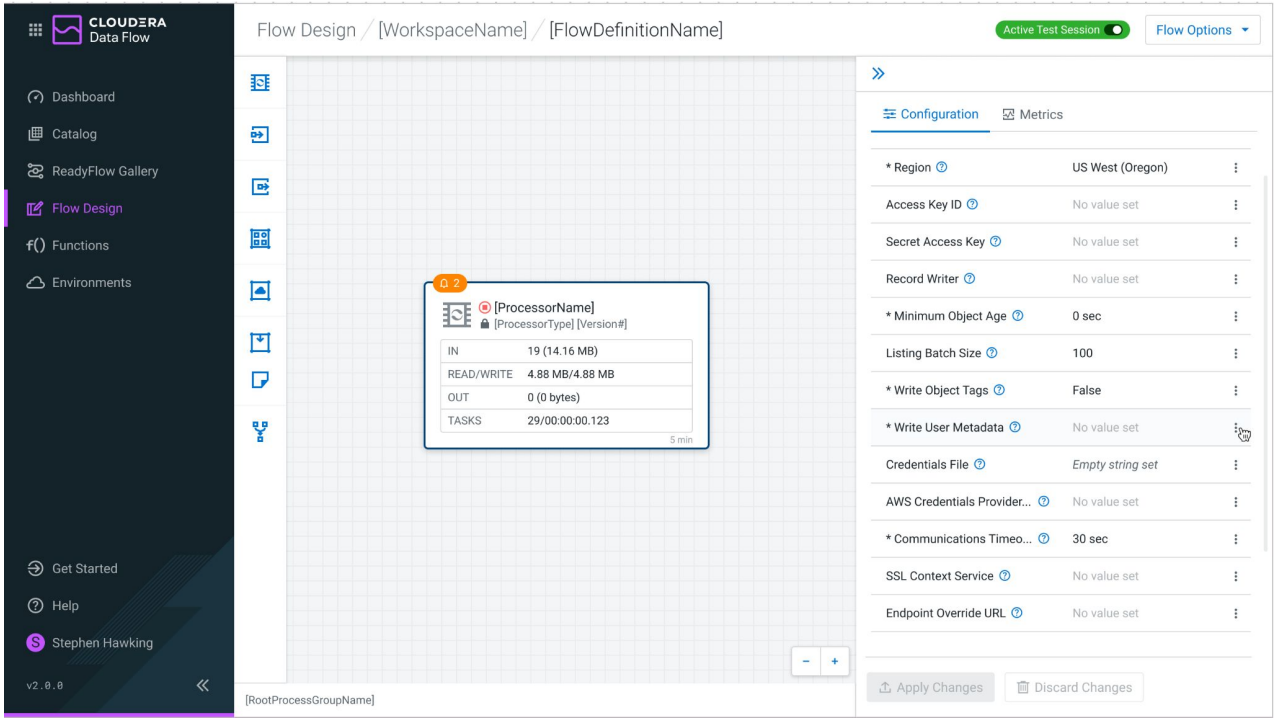
- Pick up NiFi hotfixes easily
- Upgrade (or downgrade) the hotfix version of existing deployments
- Rolling upgrade (if the deployment has >1 NiFi nodes)





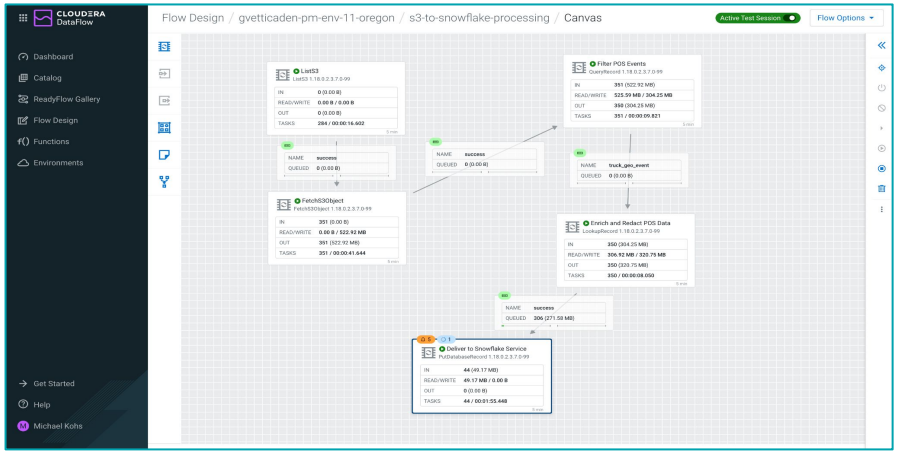
DATA FLOW DESIGN FOR EVERYONE

- Cloud-native data flow development
- Developers get their own sandbox
- Start developing flows without installing NiFi
- Redesigned visual canvas
- Optimized interaction patterns
- Integration into CDF-PC Catalog for versioning



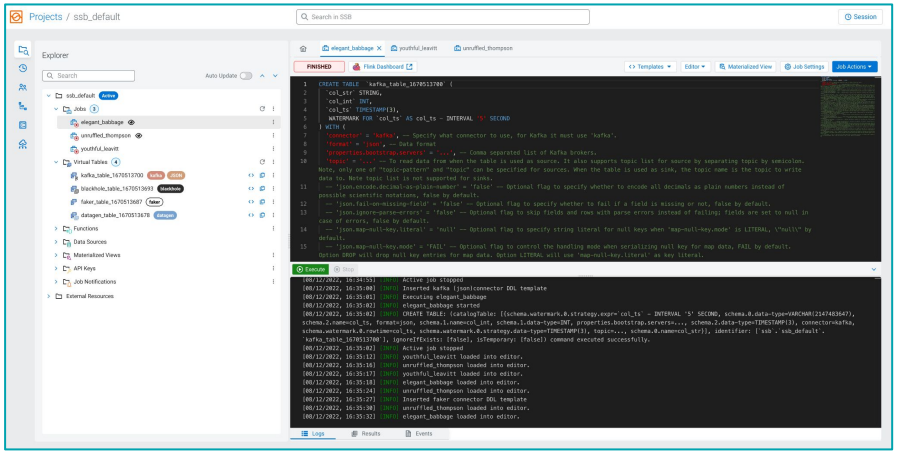
No-Code/Low-Code User Experience

DataFlow Designer



visual canvas + automatic provisioning=
Self-service pipeline development

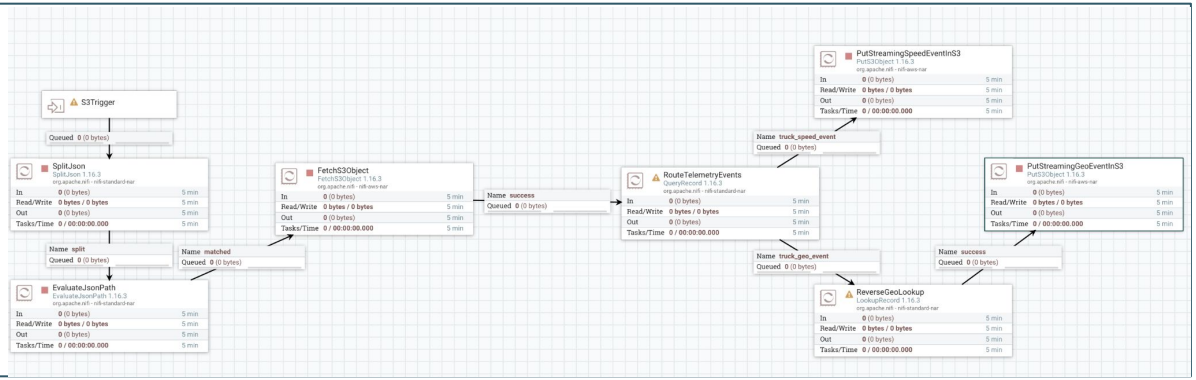
SQL Stream Builder



Instant data access + unified SQL processing =
Self-service streaming event processing

Development & Runtime of DataFlow Functions

Step1. Develop functions on local workstation or in CDP Public Cloud using **no-code**, UI designer



Step 2. Run functions on serverless compute services in AWS, Azure & GCP

AWS Lambda

Azure Functions

Google Cloud Functions

DataFlow Functions Use Cases

Trigger Based, Batch, Scheduled and Microservice Use Cases

Serverless Trigger-Based File Processing Pipeline

Develop & run data processing pipelines when files are created or updated in any of the cloud object stores

Example: When a photo is uploaded to object storage, a data flow is triggered which runs image resizing code and delivers resized image to different locations.

Serverless Workflows / Orchestration

Chain different low-code functions to build complex workflows

Example: Automate the handling of support tickets in a call center or orchestrate data movement across different cloud services.

Serverless Scheduled Tasks

Develop and run scheduled tasks without any code on pre-defined timed intervals

Example: Offload an external database running on-premises into the cloud once a day every morning at 4:00 a.m.

Serverless Microservices

Build and deploy serverless independent modules that power your applications microservices architecture

Example: Event-driven functions for easy communication between thousands of decoupled services that power a ride-sharing application.

Serverless Web APIs

Easily build endpoints for your web applications with HTTP APIs without any code using DFF and any of the cloud providers' function triggers

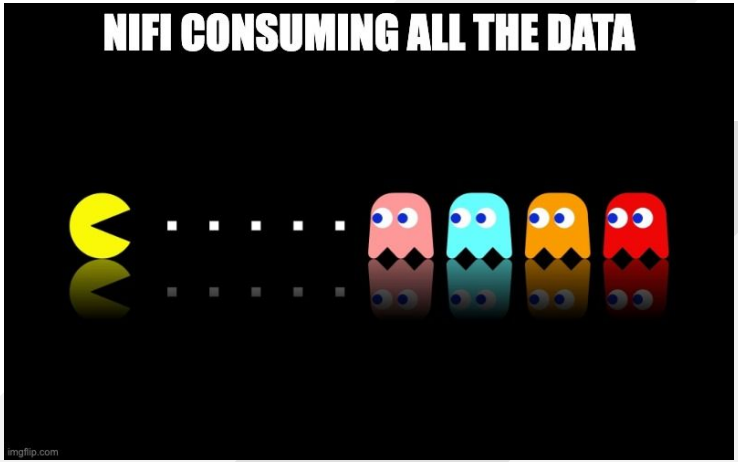
Example: Build high performant, scalable web applications across multiple data centers.

Serverless Customized Triggers

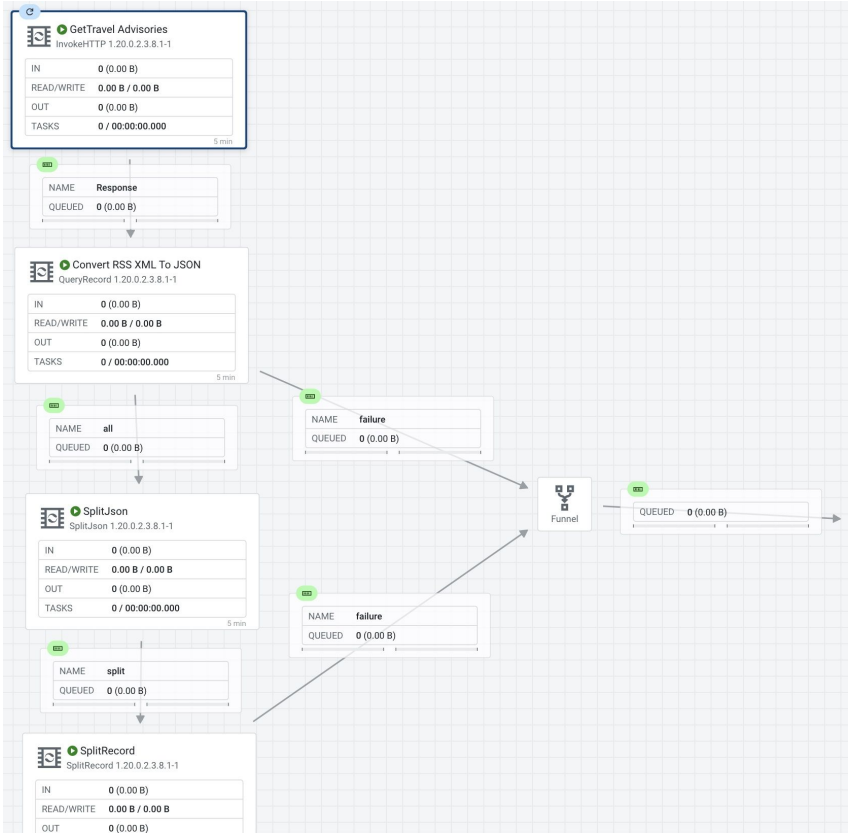
With the DFF State feature, build flows to create customized triggers allowing access to on-premises or external services

Example: Near real time offloading of files from a remote SFTP server.

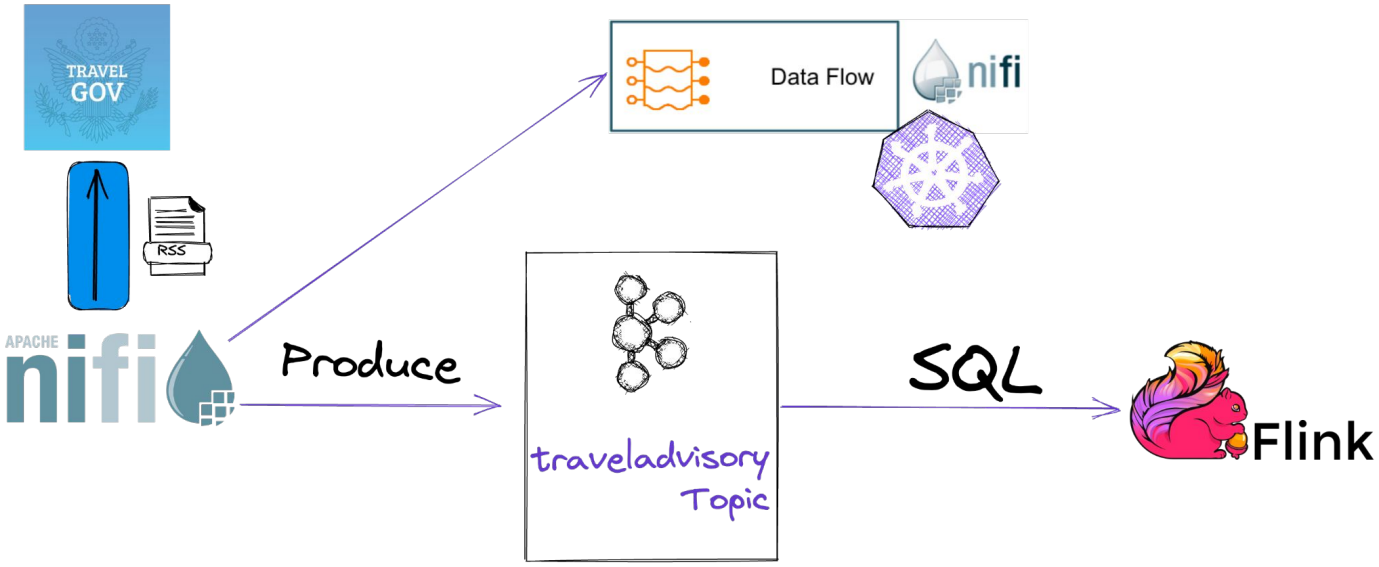
DEMOS



Cloudera Dataflow Demos

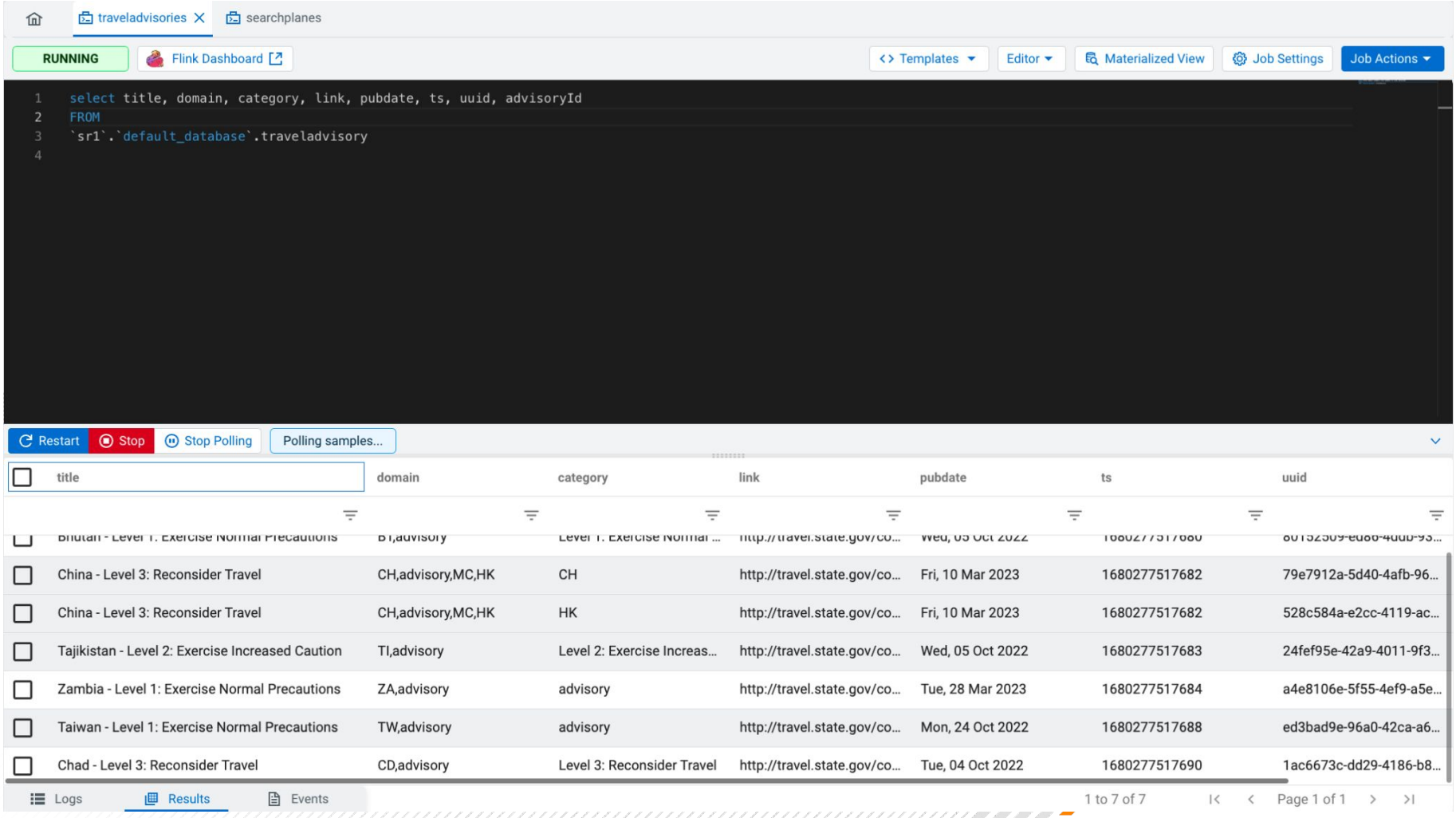


Examples

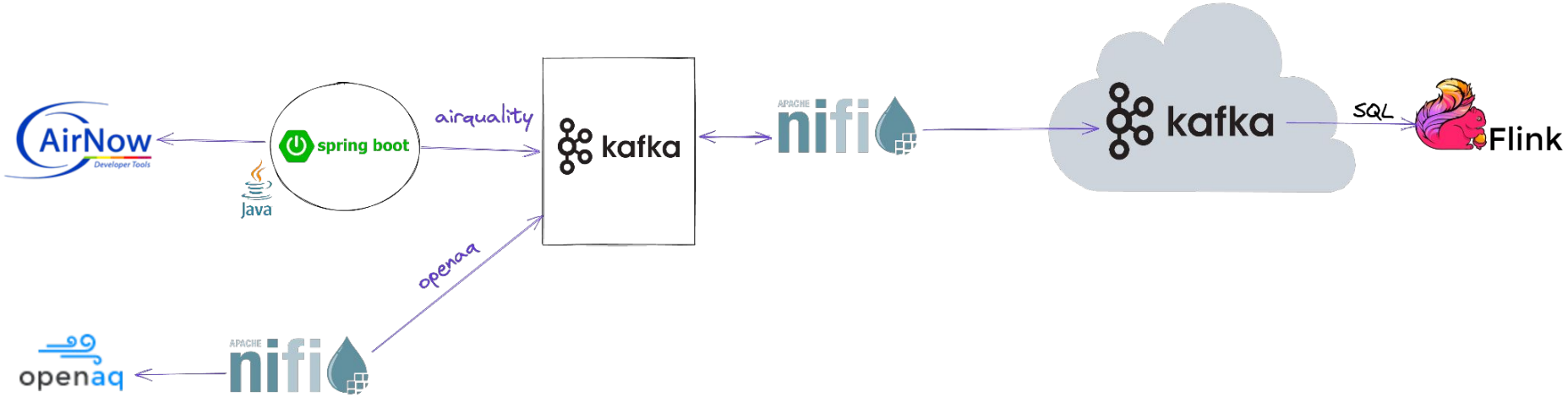


CLOUDERA

https://github.com/tspannhw/FLaNK-TravelAdvisory

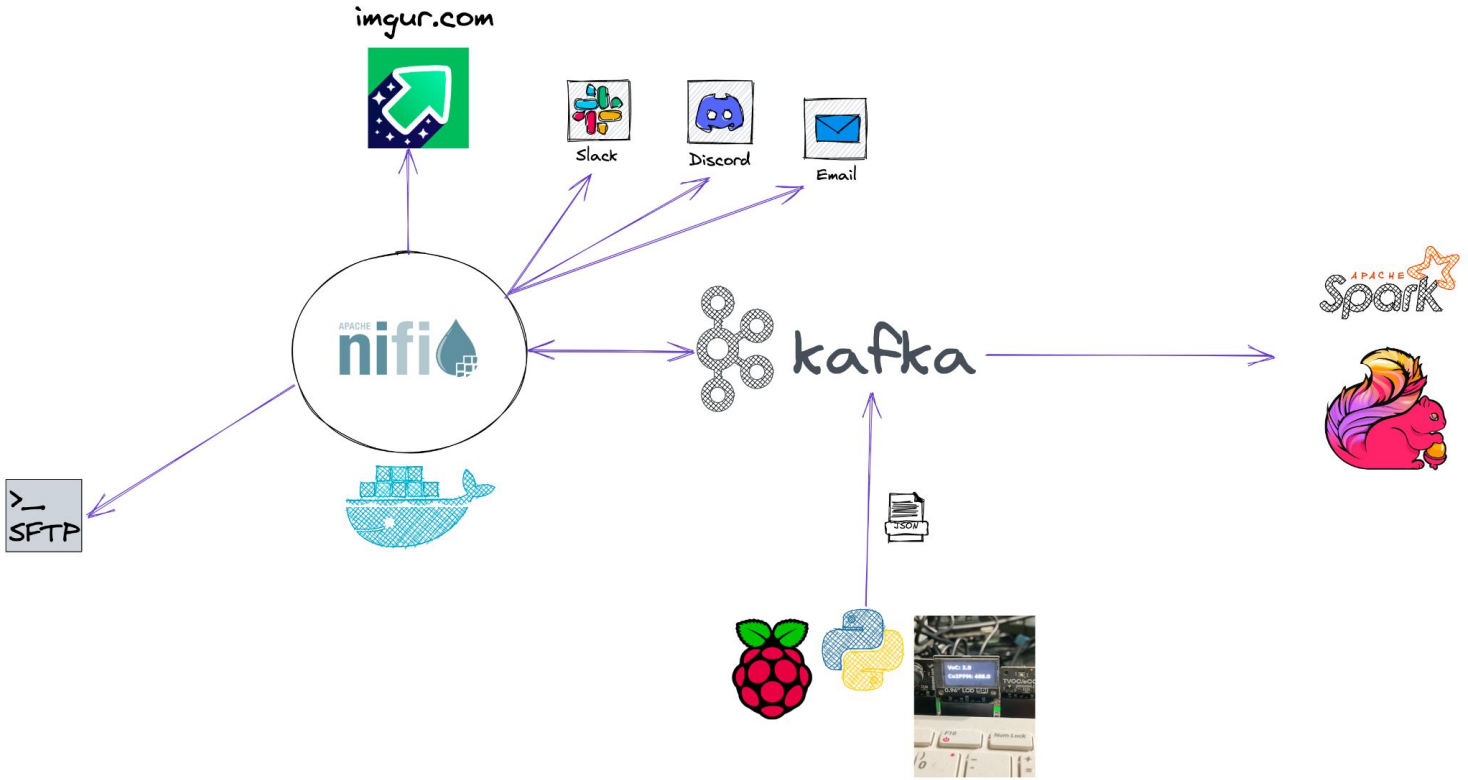

traveladvisories
searchplanes
RUNNING
Flink Dashboard
Templates
Editor
Materialized View
Job Settings
Job Actions
1 select title, domain, category, link, pubdate, ts, uuid, advisoryId
2 FROM
3 `sr1`.`default_database`.traveladvisory
4
Restart
Stop
Stop Polling
Polling samples...
title
domain
category
link
pubdate
ts
uuid
China - Level 3: Reconsider Travel
CH,advisory,MC,HK
CH
http://travel.state.gov/co...
Fri, 10 Mar 2023
1680277517682
79e7912a-5d40-4afb-96...
China - Level 3: Reconsider Travel
CH,advisory,MC,HK
HK
http://travel.state.gov/co...
Fri, 10 Mar 2023
1680277517682
528c584a-e2cc-4119-ac...
Tajikistan - Level 2: Exercise Increased Caution
TI,advisory
Level 2: Exercise Increas...
http://travel.state.gov/co...
Wed, 05 Oct 2022
1680277517683
24fef95e-42a9-4011-9f3...
Zambia - Level 1: Exercise Normal Precautions
ZA,advisory
advisory
http://travel.state.gov/co...
Tue, 28 Mar 2023
1680277517684
a4e8106e-5f55-4ef9-a5e...
Taiwan - Level 1: Exercise Normal Precautions
TW,advisory
advisory
http://travel.state.gov/co...
Mon, 24 Oct 2022
1680277517688
ed3bad9e-96a0-42ca-a6...
Chad - Level 3: Reconsider Travel
CD,advisory
Level 3: Reconsider Travel
http://travel.state.gov/co...
Tue, 04 Oct 2022
1680277517690
1ac6673c-dd29-4186-b8...
Logs
Results
Events
1 to 7 of 7
Page 1 of 1

Examples - Spring Boot - Apache Kafka - Apache NiFi - Apache Flink

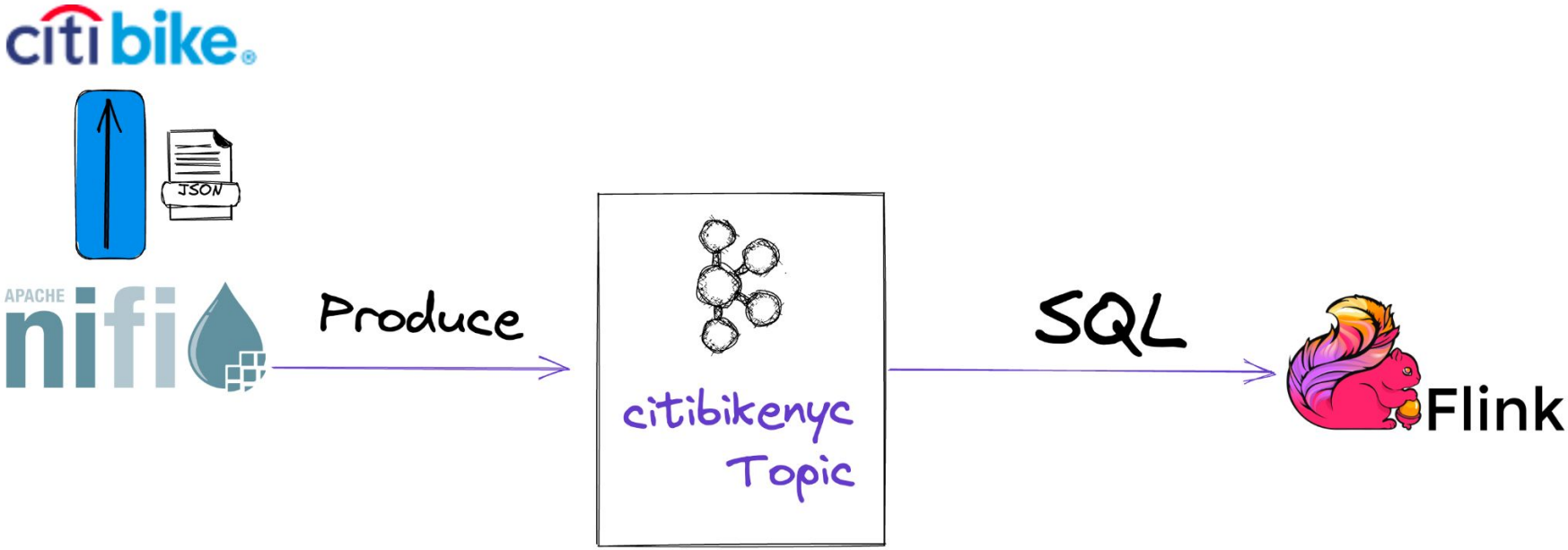


https://github.com/tspannhw/flank-airquality

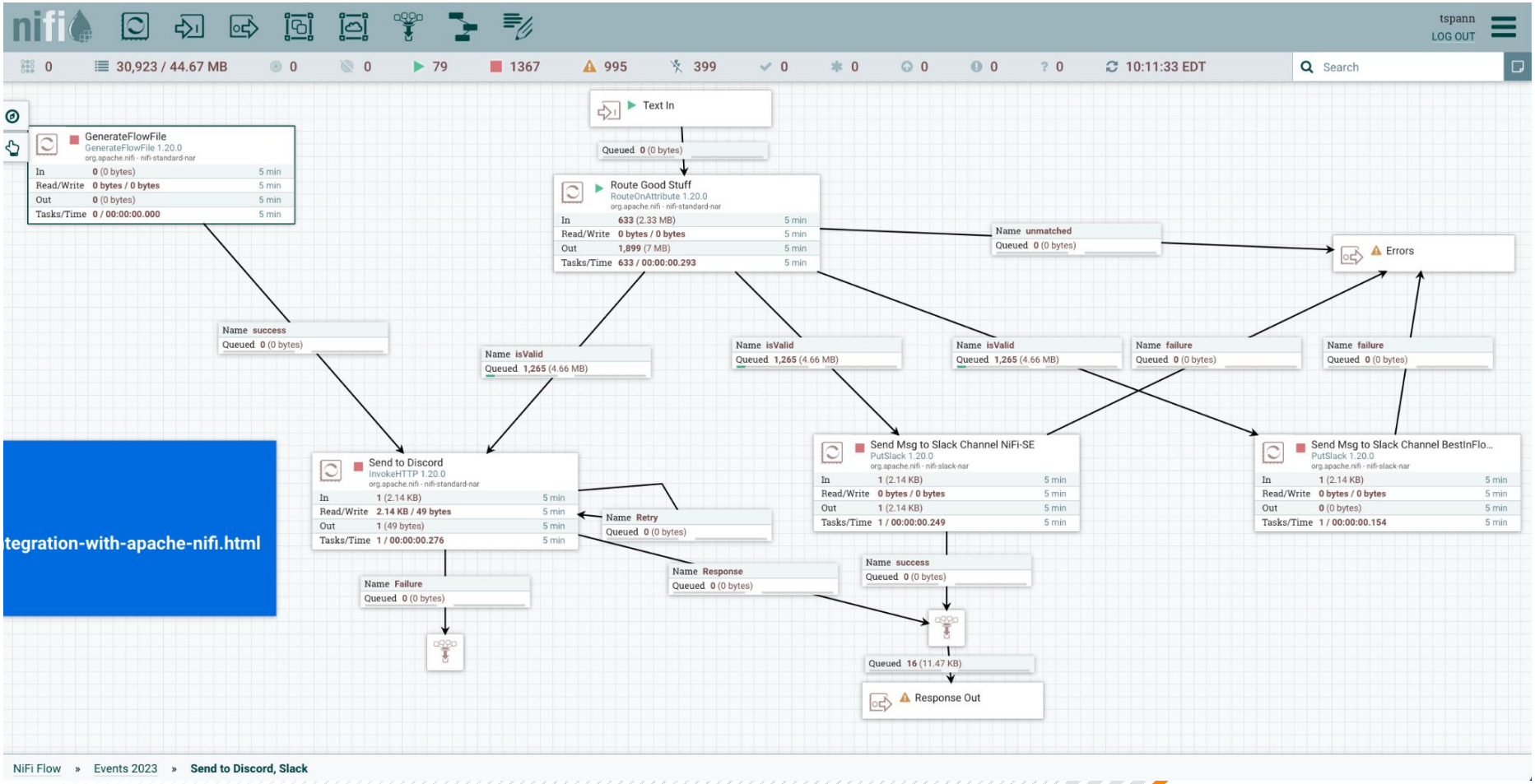
Examples - Python - Kafka - NiFi - Flink - Spark

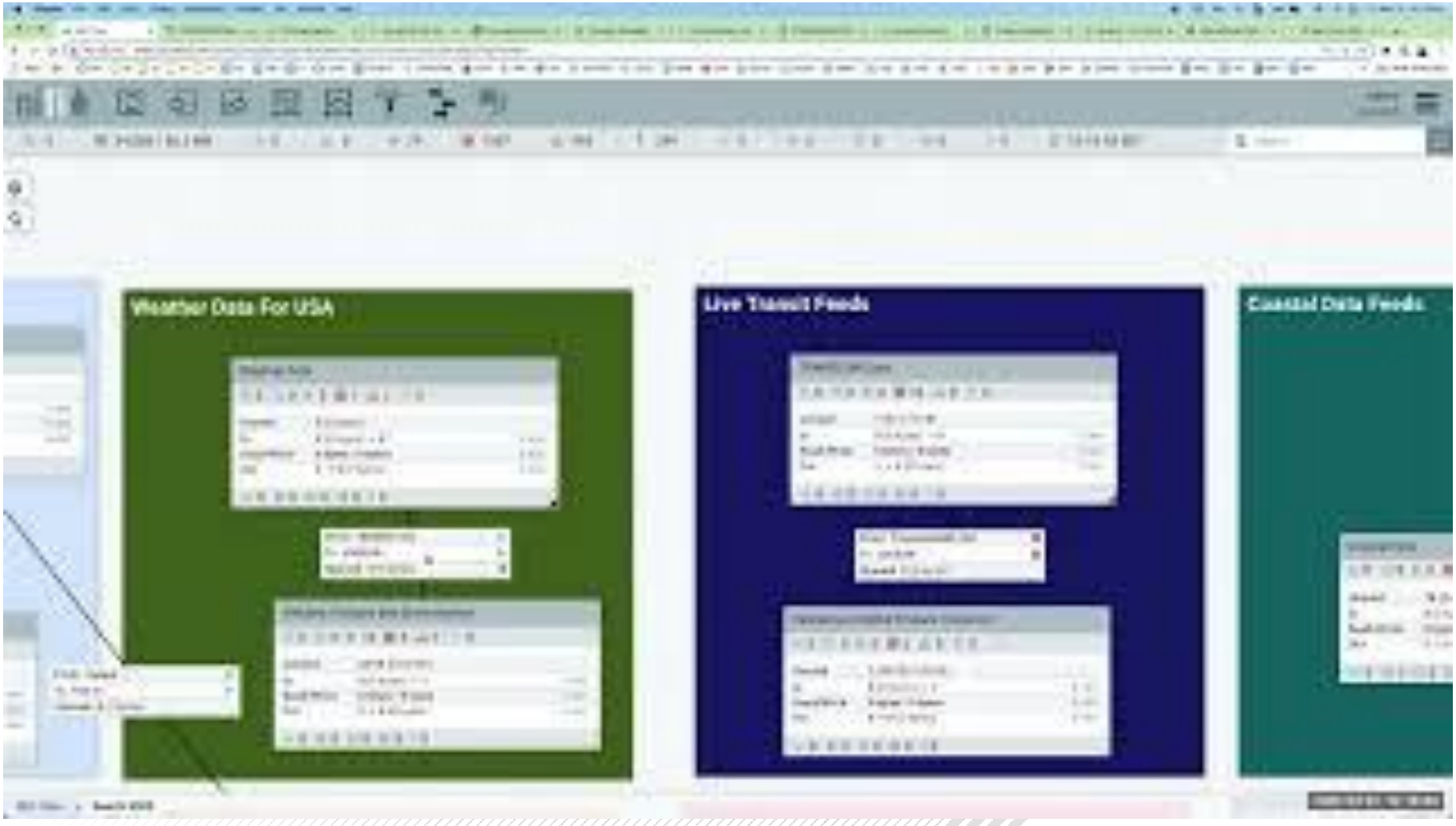


Examples - NiFi - Kafka - Flink



CLOUDERA

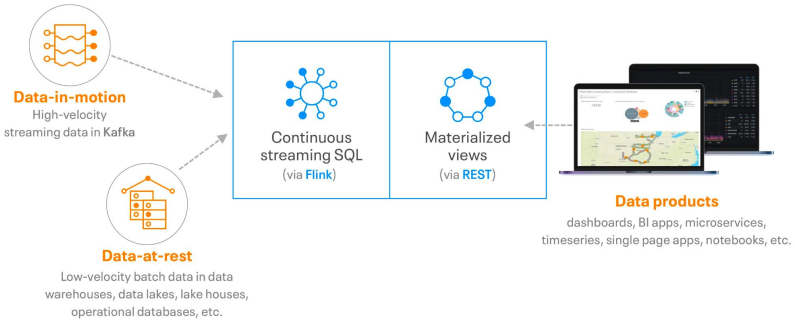



Weather Data For USA
Live Transit Feeds
Coastal Data Feeds

BEST PRACTICES

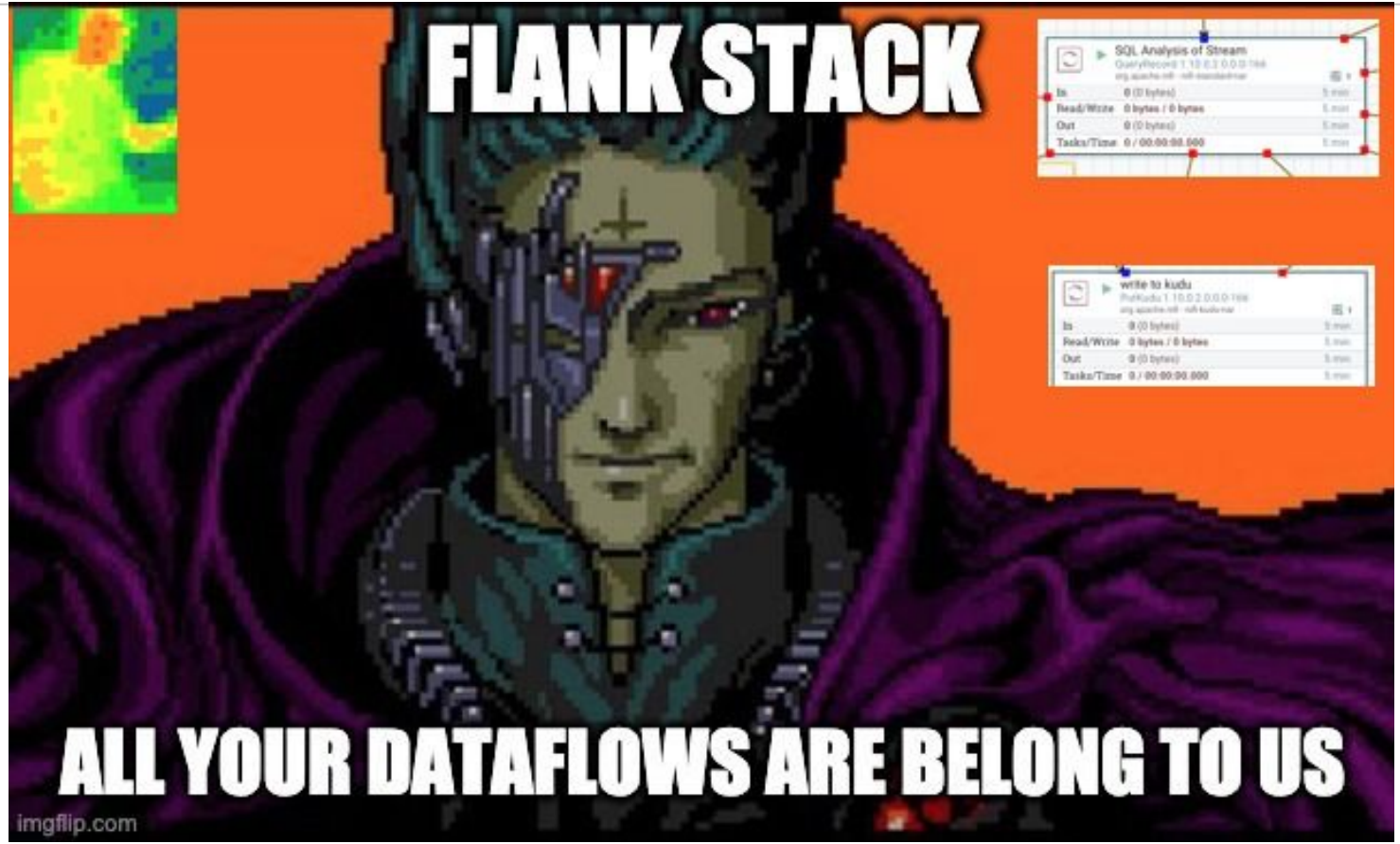
STREAMING TECH DEBT TIPS

- Version Control All Assets
- Managed Public Cloud like Cloudera
- Use DevOps and APIs
- Latest Java and Python
- Stream Sizing (NiFi, Kafka, Flink)



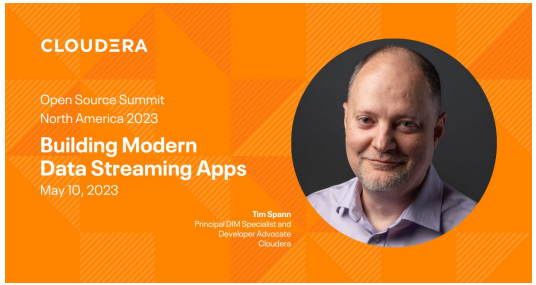
RESOURCES AND WRAP-UP

Resources

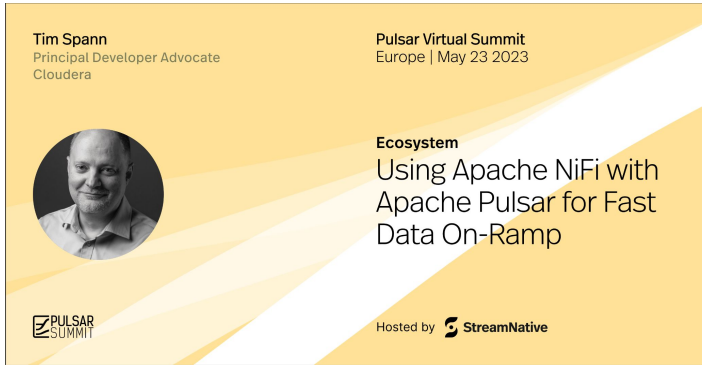


Upcoming Events

May 10



May 23



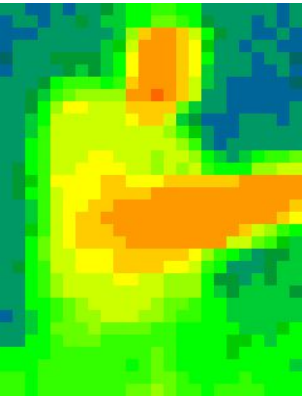
May 25

Streaming Resources

- https://dzone.com/articles/real-time-stream-processing-with-hazelcast-and-streamnative
- https://flipstackweekly.com/
- https://www.datainmotion.dev/
- https://www.flankstack.dev/
- https://github.com/tspannhw
- https://medium.com/@tspann
- https://medium.com/@tspann/predictions-for-streaming-in-2023-ad4d7395d714
- https://www.apachecon.com/acna2022/slides/04_Spann_Tim_Citizen_Streaming_Engineer.pdf

Spring Things

- https://spring.io/guides/gs/spring-boot/
- https://spring.io/projects/spring-amqp/
- https://spring.io/projects/spring-kafka/
- https://github.com/spring-projects/spring-integration-kafka
- https://github.com/spring-projects/spring-integration
- https://github.com/spring-projects/spring-data-relational
- https://github.com/spring-projects/spring-kafka
- https://github.com/spring-projects/spring-amqp



THANK YOU