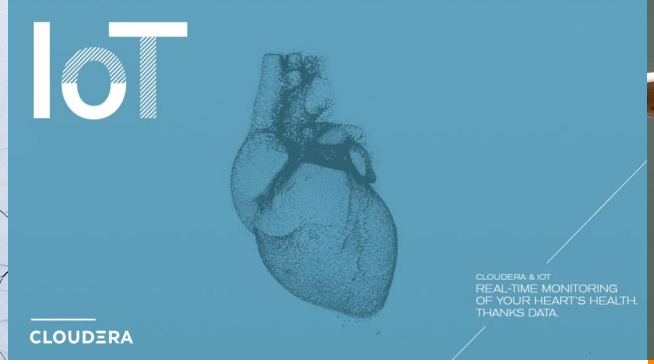
CLOUDERA

Streaming Data Pipeline Development

Tim Spann
Principal Developer Advocate

25-April-2023



CLOUDERA

EDGE
2AI
CLOUDERA

ENTERPRISE
DATA
CLOUD
CLOUDERA

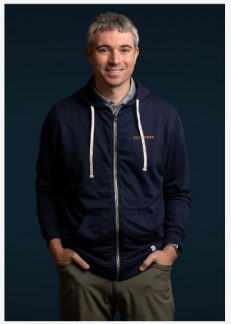
Meet the NiFi Committers

Wednesday, May 3, 2023
10am-11amPT/12pm-1pmCT/1pm-2pmET

Hosts

Joe Witt, VP of Engineering, Cloudera

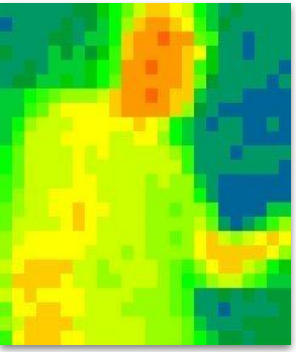
Mark Payne, Principal Engineer, Cloudera



Matt Gilman, Principal Engineer, Cloudera

https://attend.cloudera.com/nificommitters0503

FLaNK Stack



Tim Spann

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

Principal Developer Advocate.

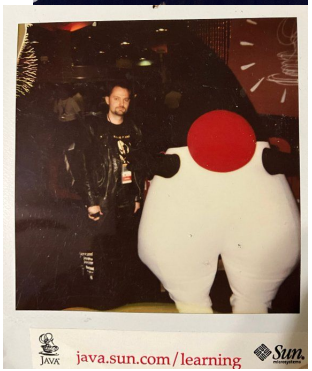
Princeton Future of Data Meetup.

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



Tim Spann
Principal Developer Advocate | Cloudera

Write the reasons someone should attend my meetup on Streaming Data Pipeline Development with Apache NiFi, Apache Kafka and Apache Flink

View other drafts

Here are some reasons why someone should attend your meetup on Streaming Data Pipeline Development with Apache NiFi, Apache Kafka and Apache Flink:

- Learn from experts in the field. The meetup will feature presentations from experts in Apache NiFi, Apache Kafka, and Apache Flink.
- Get hands-on experience. The meetup will also include hands-on workshops where you can learn how to use these technologies.
- Network with other developers. The meetup is a great opportunity to network with other developers who are interested in streaming data pipeline development.
- Stay up-to-date on the latest technologies. The meetup will cover the latest developments in Apache NiFi, Apache Kafka, and Apache Flink.
- Have fun! The meetup is a great opportunity to learn new things and network with other developers.

Google it

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.

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

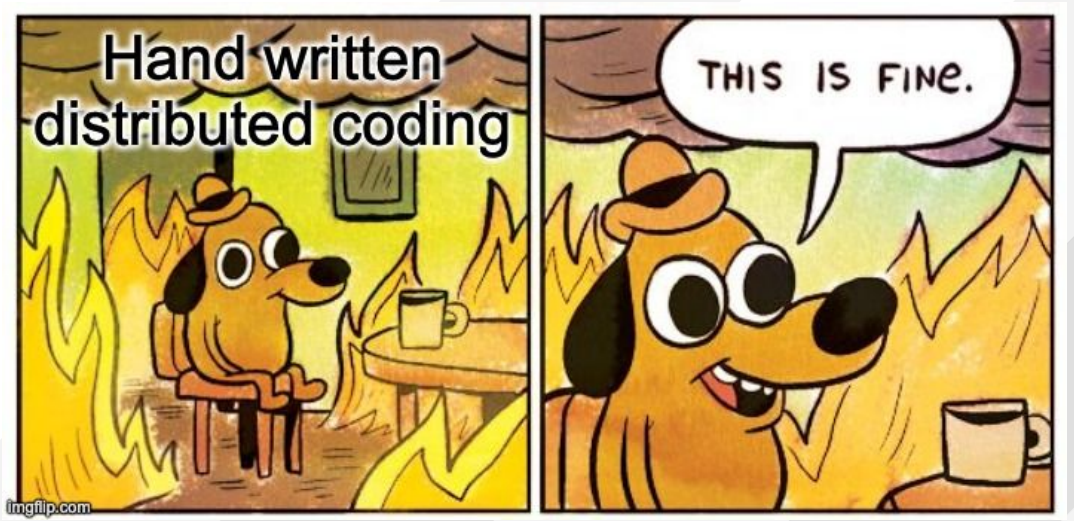
- Docker compose file of CSP to run from command line w/o any dependencies, including Flink, SQL Stream Builder, Kafka, Kafka Connect, Streams Messaging Manager and Schema Registry
  - `$> docker compose up`
- Licensed under the Cloudera Community License
- **Unsupported**
- Community Group Hub for CSP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

CSP Community Edition

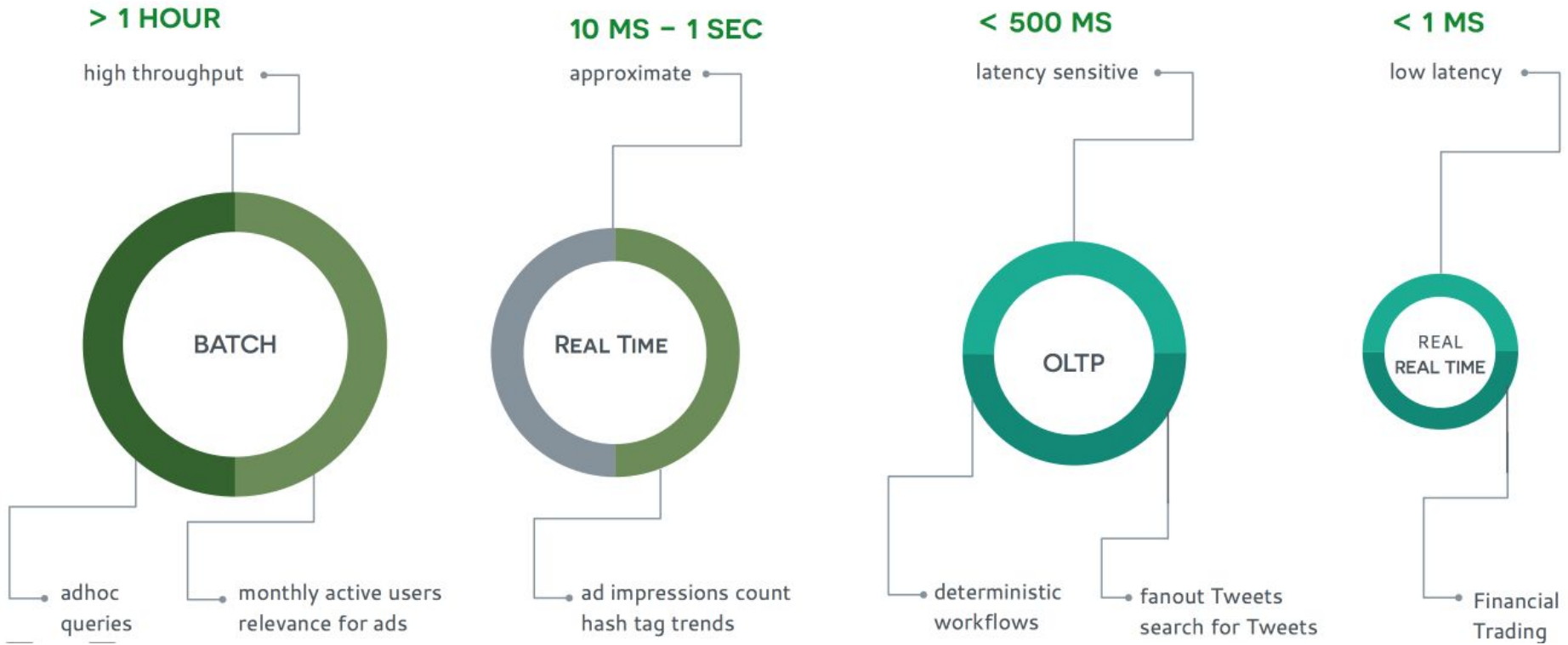
A readily available, dockerized deployment of Apache Kafka and Apache Flink that allows you to test the features and capabilities of Cloudera Stream Processing.

Learn More

STREAMING

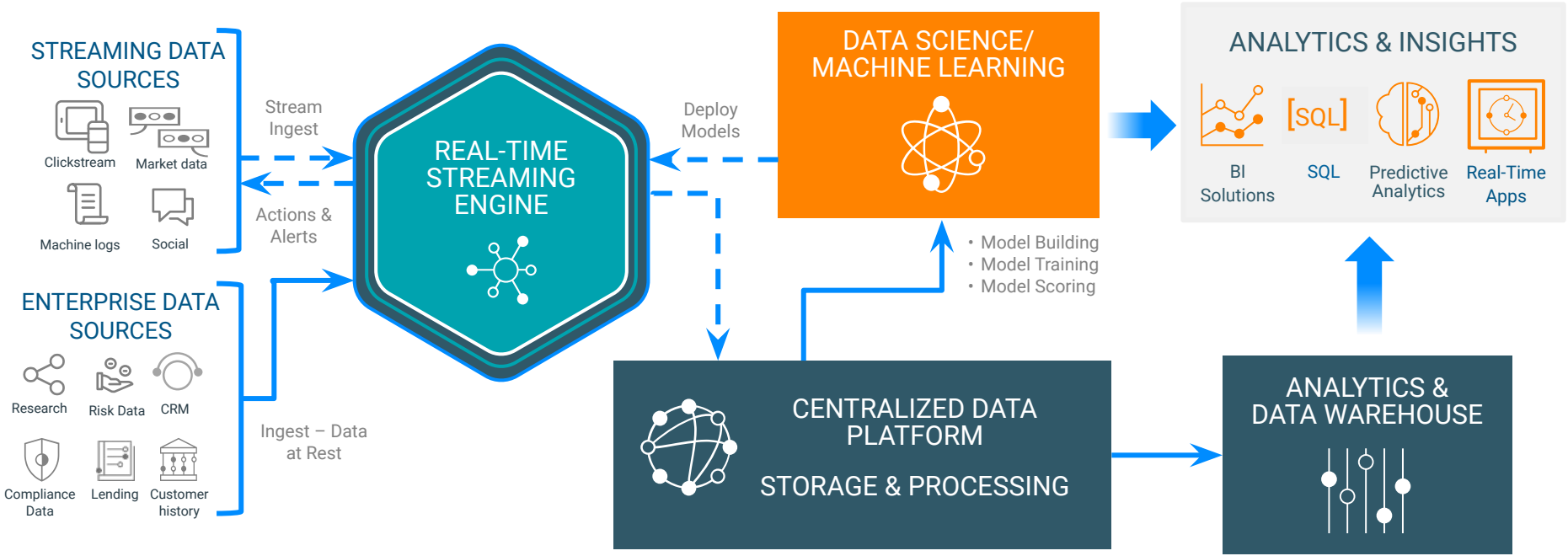


WHAT IS REAL-TIME?



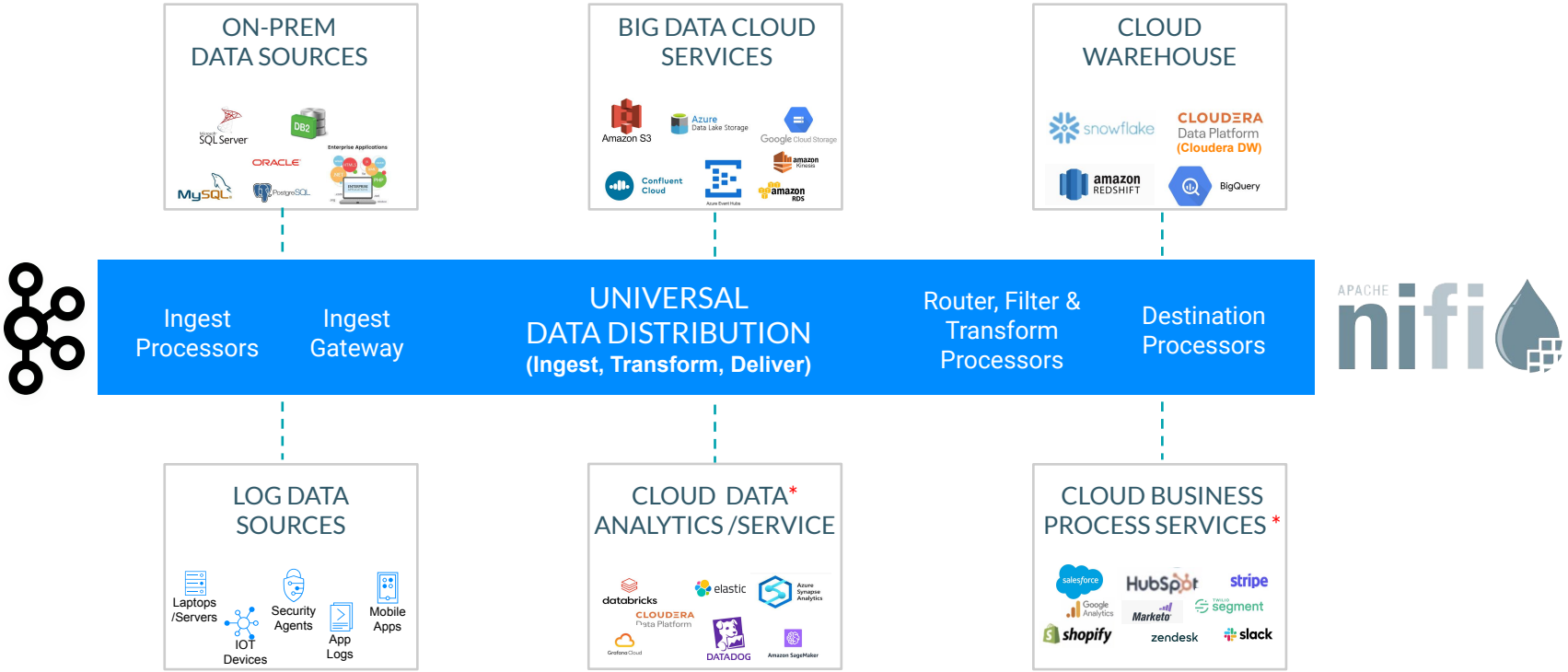
ENABLING ANALYTICS AND INSIGHTS ANYWHERE

Driving enterprise business value



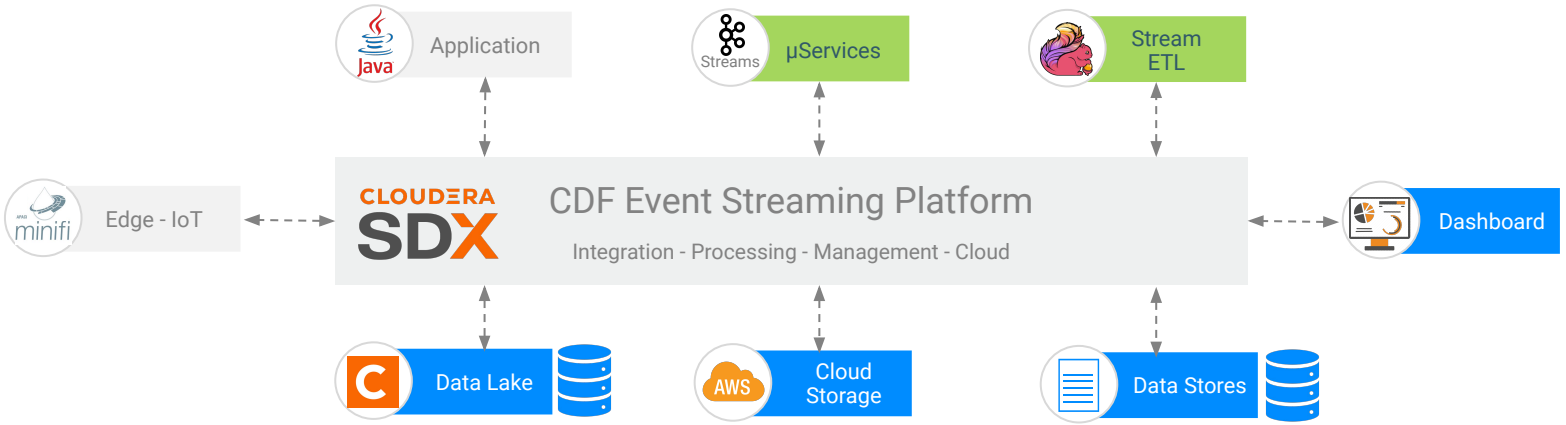
STREAMING FROM … TO .. WHILE ..

Data distribution as a first class citizen

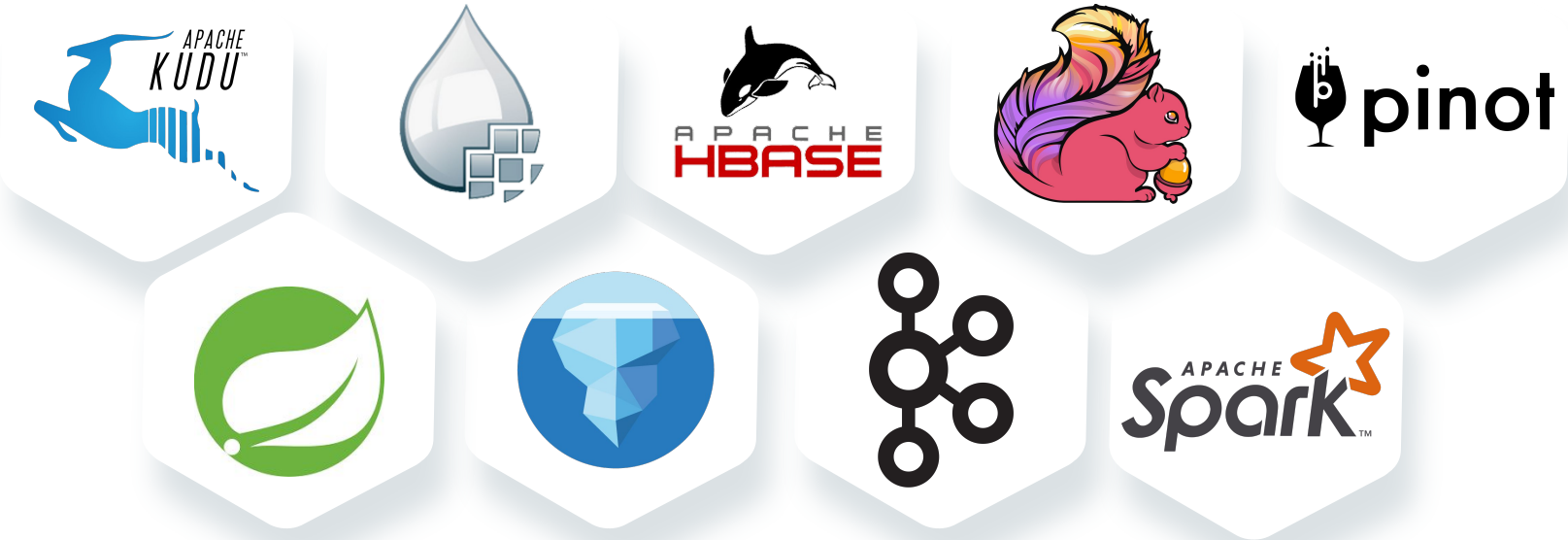


EVENT-DRIVEN ORGANIZATION

Modernize your data and applications



BUILDING REAL-TIME REQUIRES A TEAM

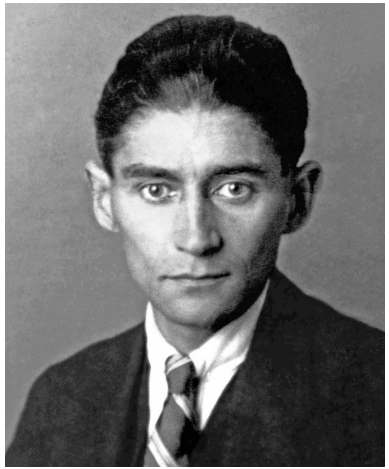


APACHE KAFKA

Yes, Franz, It's Kafka

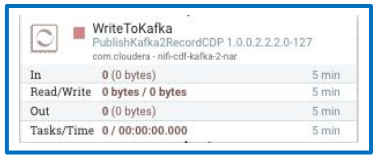
Let's do a metamorphosis on your data. Don't fear changing data.

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

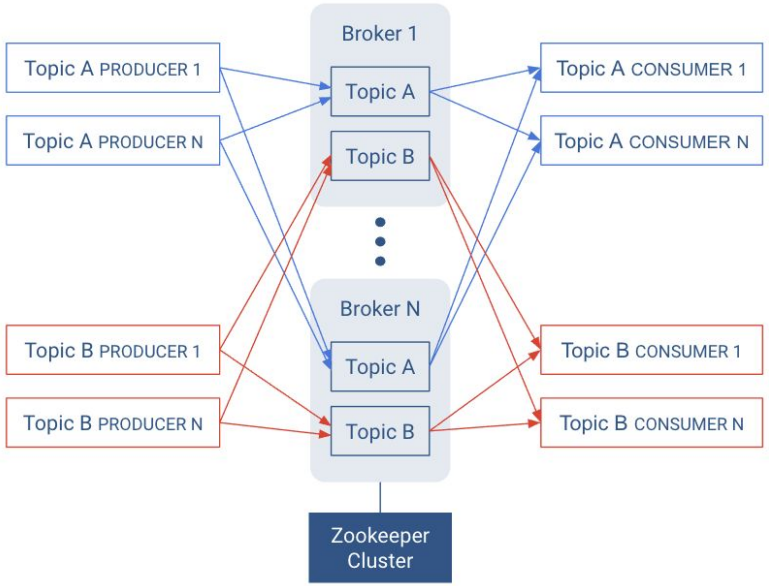


Franz Kafka was a German-speaking Bohemian novelist and short-story writer, widely regarded as one of the major figures of 20th-century literature. His work fuses elements of realism and the fantastic.
Wikipedia

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 Apache Kafka?

- **Distributed:** horizontally scalable
- **Partitioned:** the data is split-up and distributed across the brokers
- **Replicated:** allows for automatic failover
- **Unique:** Kafka does not track the consumption of messages (the consumers do)
- **Fast:** designed from the ground up with a focus on performance and throughput

– Kafka was built at Linkedin in 2011

– Open sourced as an Apache project

What is Can You Do With Apache Kafka?

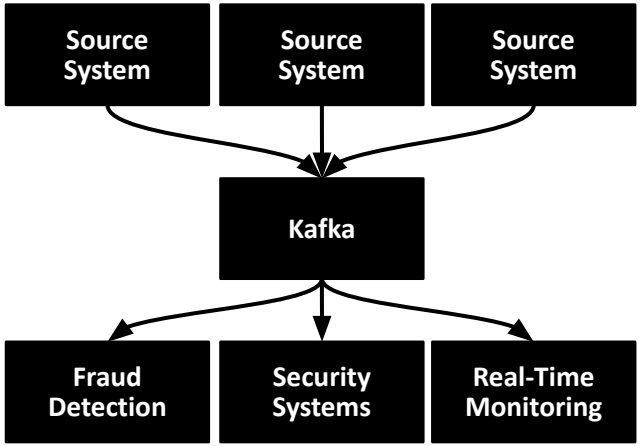
- Web site activity: track page views, searches, etc. in real time
- Events & log aggregation: particularly in distributed systems where messages come from multiple sources
- Monitoring and metrics: aggregate statistics from distributed applications and build a dashboard application
- Stream processing: 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

KAFKA TERMINOLOGY

- Kafka is a publish/subscribe messaging system comprised of the following components:
  - **Topic**: a message feed
  - **Producer**: a process that publishes messages to a topic
  - **Consumer**: a process that subscribes to a topic and processes its messages
  - **Broker**: a server in a Kafka cluster

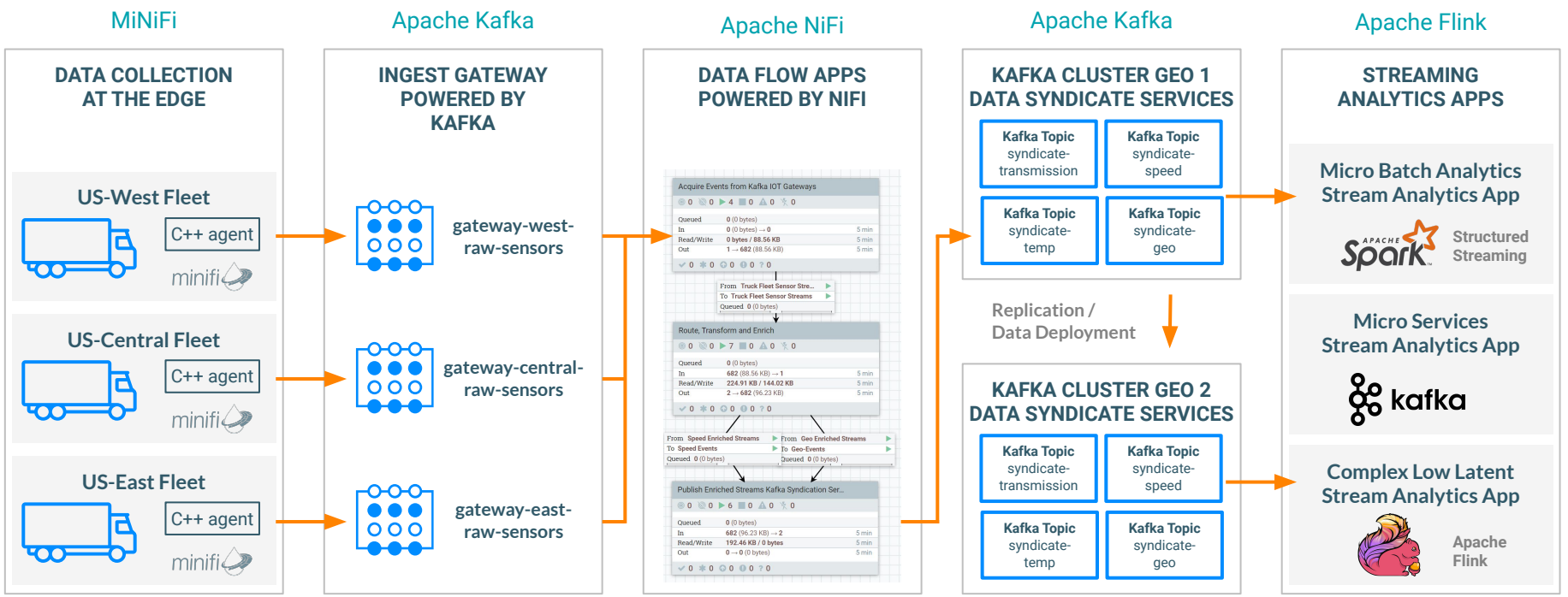
Apache Kafka

- 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



Many-To-Many
Publish-Subscribe

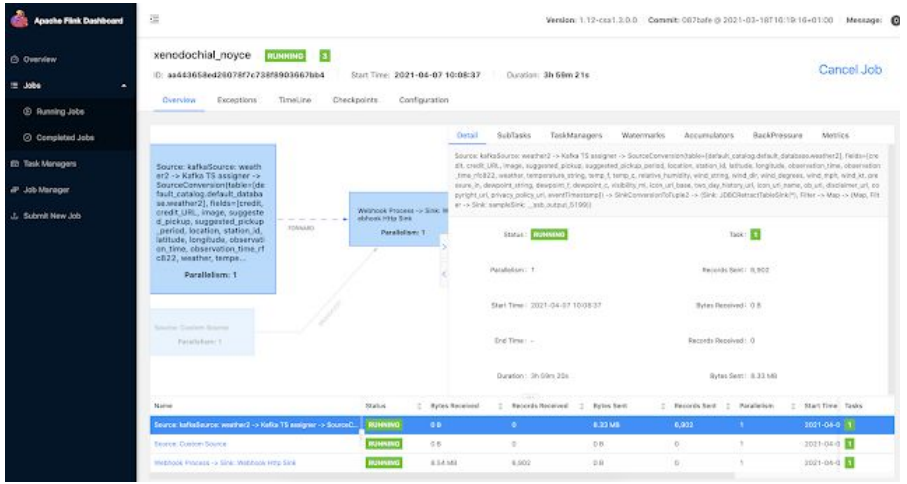
Apache Kafka



APACHE FLINK

Flink SQL

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

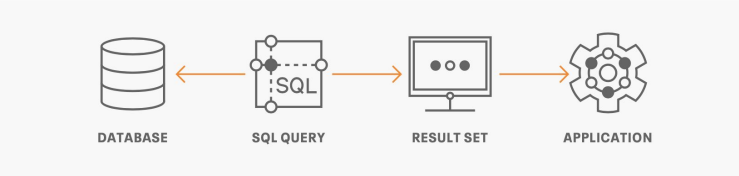


https://www.datainmotion.dev/2021/04/cloudera-sql-stream-builder-ssb-updated.html

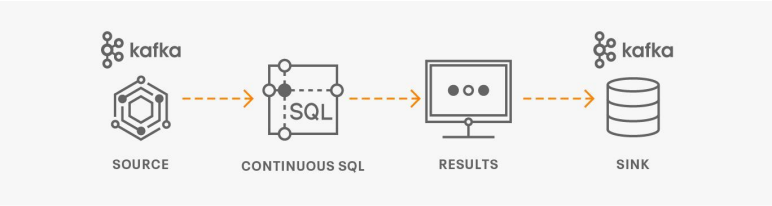
CONTINUOUS SQL

- 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`

Flink SQL

Key Takeaway: Rich SQL grammar with advanced time and aggregation tools

```
-- specify Kafka partition key on output
SELECT foo AS _eventKey FROM sensors

-- use event time timestamp from kafka
-- exactly once compatible
SELECT eventTimestamp FROM sensors

-- nested structures access
SELECT foo.'bar' FROM table; -- must quote nested
column

-- timestamps
SELECT * FROM payments
WHERE eventTimestamp > CURRENT_TIMESTAMP-interval
'10' second;

-- unnest
SELECT b.*, u.*
FROM bgp_avro b,
UNNEST(b.path) AS u(pathitem)
```

```
-- aggregations and windows
SELECT card,
MAX(amount) as theamount,
TUMBLE_END(eventTimestamp, interval '5' minute) as
ts
FROM payments
WHERE lat IS NOT NULL
AND lon IS NOT NULL
GROUP BY card,
TUMBLE(eventTimestamp, interval '5' minute)
HAVING COUNT(*) > 4  -- >4==fraud

-- try to do this ksql!
SELECT us_west.user_score+ap_south.user_score
FROM kafka_in_zone_us_west us_west
FULL OUTER JOIN kafka_in_zone_ap_south ap_south
ON us_west.user_id = ap_south.user_id;
```

CLOUDERA SQL STREAM BUILDER

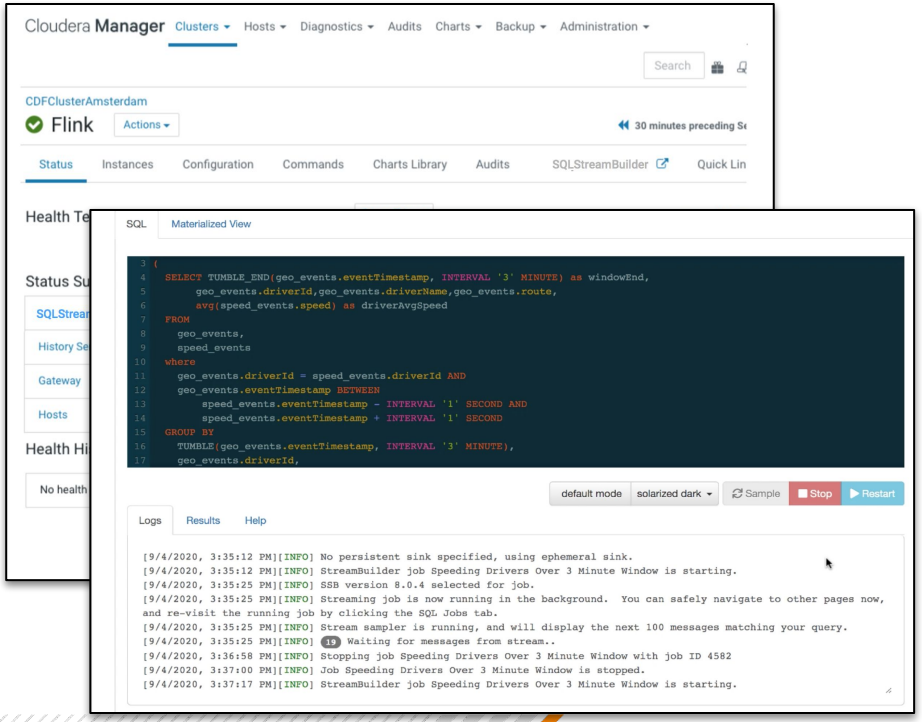
Making Streaming Analytics accessible to everyone with SQL

Application Developer

- Develop & test SQL queries with a powerful UI
- Expose streaming data to applications through materialized views
- Single button “Push to production” turns SQL queries into Flink application

Business Analyst

- Explore Streaming Data using SQL without learning new skills
- Build new real-time business reporting applications



SQL STREAM BUILDER (SSB)

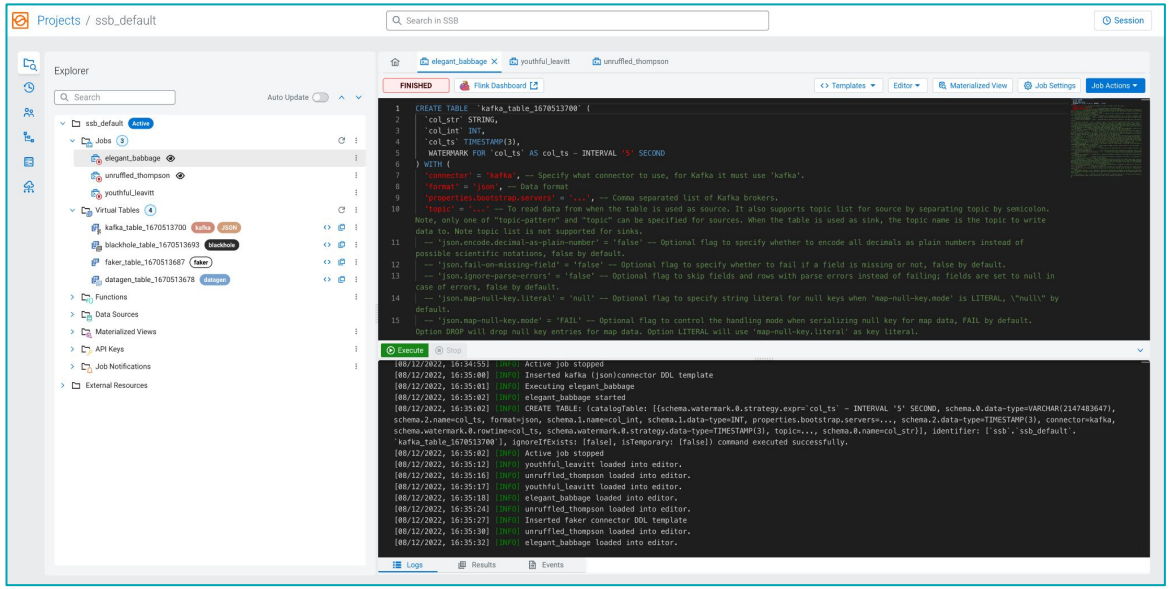
Democratize access to real-time data with just SQL

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

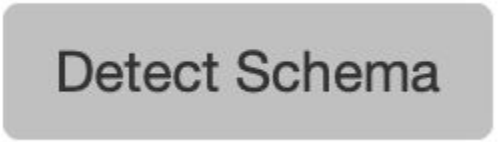
Enrich streaming data with batch data in a single tool



SCHEMA

Key Takeaway: Integrated with schema registry, also auto-detection for JSON types.

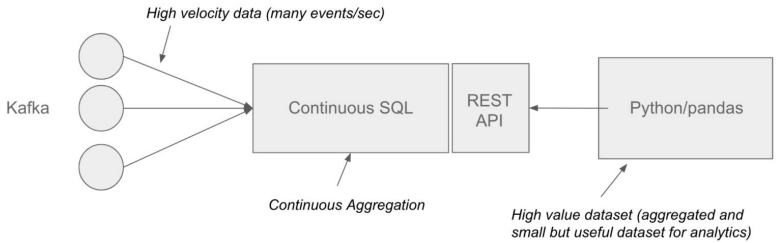
- AVRO - Schema Registry
- JSON - Schema Auto-detect
- Virtual Table design pattern
- Kafka Data Source auto-created in SSB




```
{
  "fields": [
    {
      "doc": "Type inferred from '215'",
      "name": "userid",
      "type": "long"
    },
    {
      "doc": "Type inferred from '94204'",
      "name": "amount",
      "type": "long"
    }
  ],
  "name": "inferredSchema",
  "type": "record"
}
```

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
```

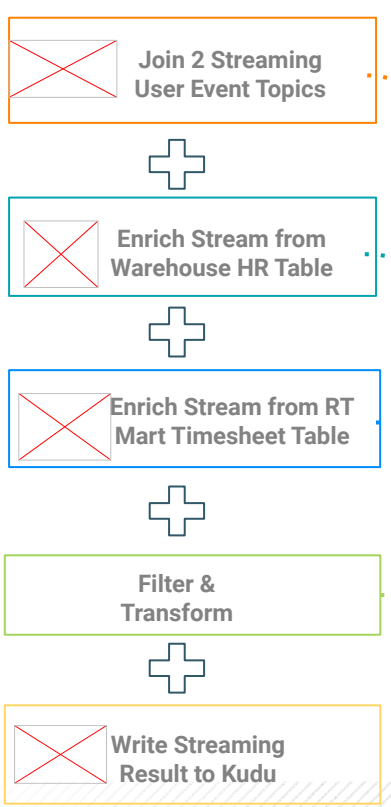
```
[90]: import pandas as pd
[91]: mv = "https://xxxyyyyzzz"
[92]: df = pd.read_json(mv)
[93]: len(df.keys())
[93]: 5
[95]: df['ts'] = pd.to_datetime(df['ts'])
[97]: df.dtypes
[97]: max_amount             int64
      sum_amount             int64
      thecount               int64
      ts          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

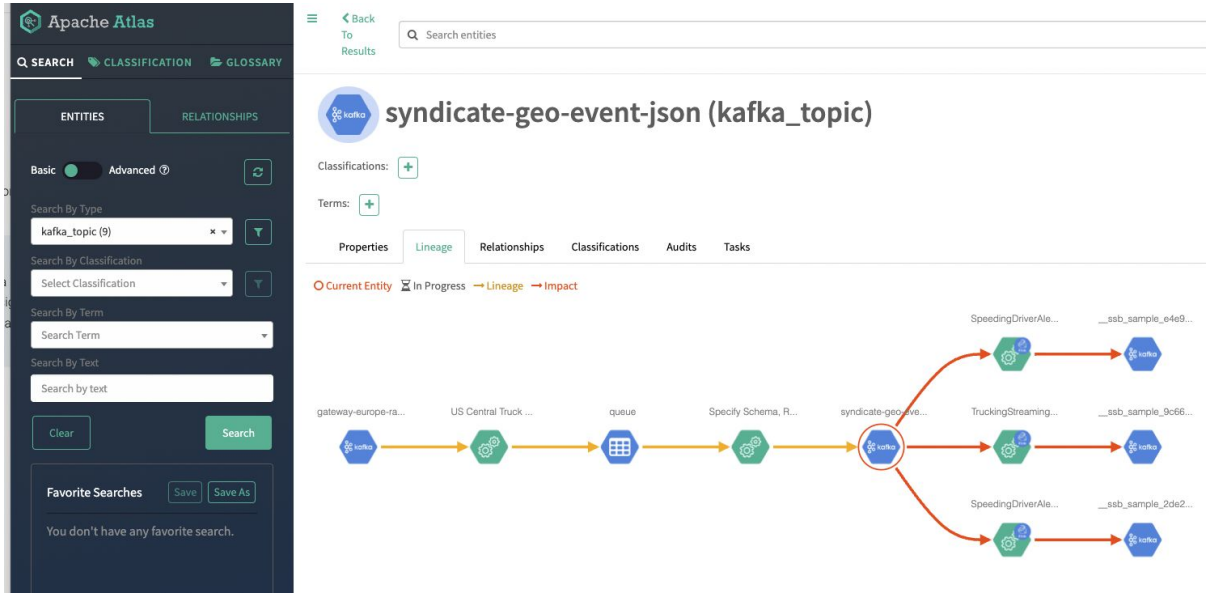



```
SELECT
      geo_event.eventTimestamp, geo_event.driverId, geo_event.eventTime,geo_event.eventSource,
      geo_event.truckId,geo_event.driverName,geo_event.routeId,geo_event.route,geo_event.eventType,
      geo_event.latitude, geo_event.longitude, geo_event.correlationId, geo_event.geoAddress,
      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

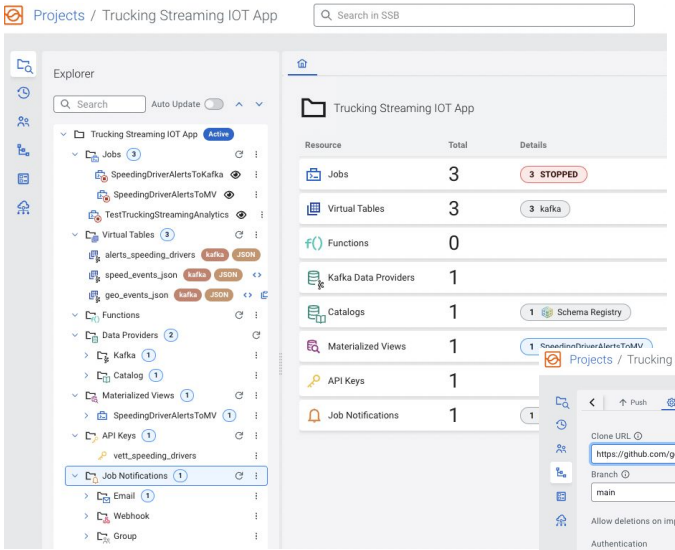


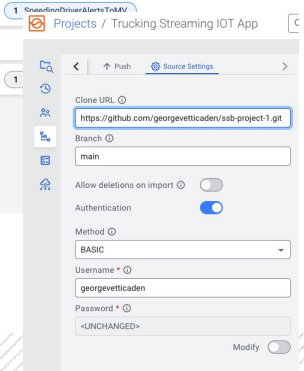
SSB Projects - Container Structure for All Assets of SQL Streaming Job

SDLC for Streaming SQL Applications With First Class Git Integration

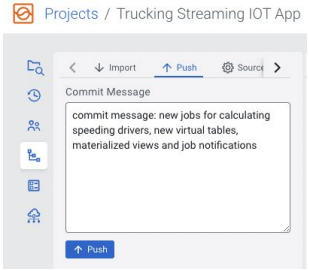
Project in SSB

Project Represented In Git

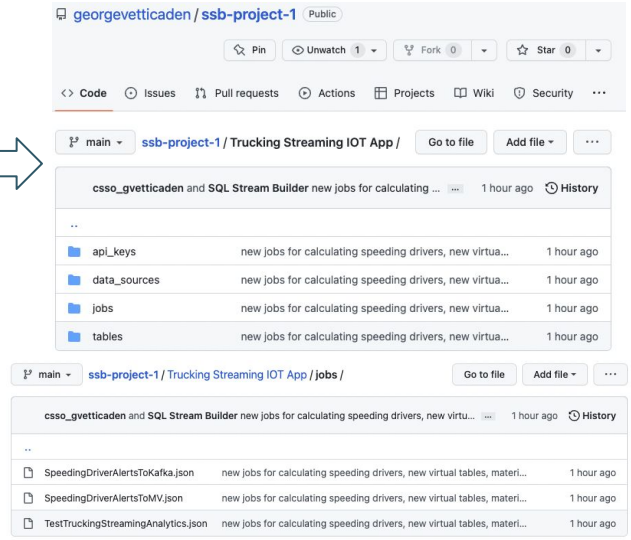




SSB Project provides the container structure for all the assets for your streaming app. Project is configured with a git repository



SSB allows you to push/import projects to/from Git

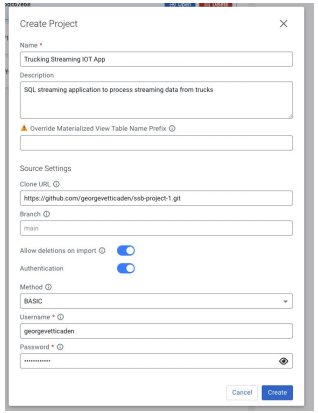


The streaming application assets in git within the project structure

SDLC Life Cycle with SSB Projects

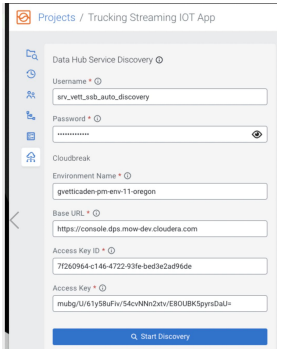
Step 1

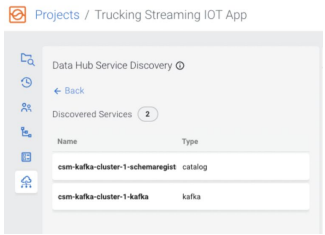
Create SSB Project & Configure Git Repo



Step 2

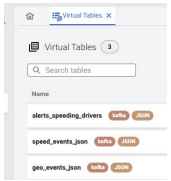
Run Service Discovery to register Kafka, Hive, etc

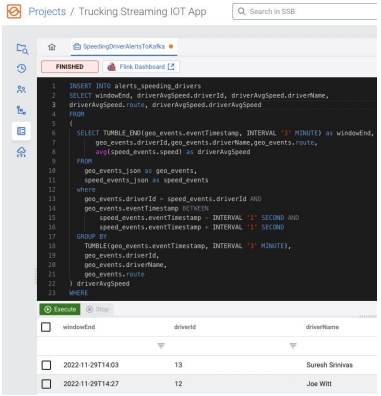




Step 3

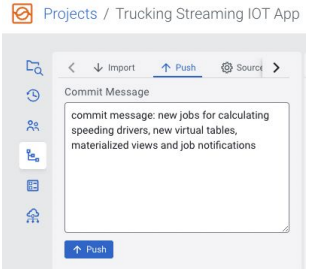
Create/Develop Streaming Assets & Test





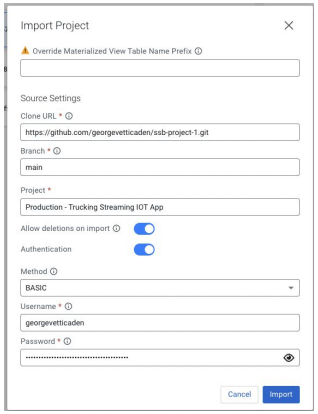
Step 4

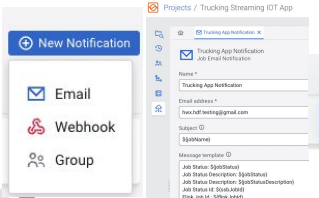
Check-in Project Into Git



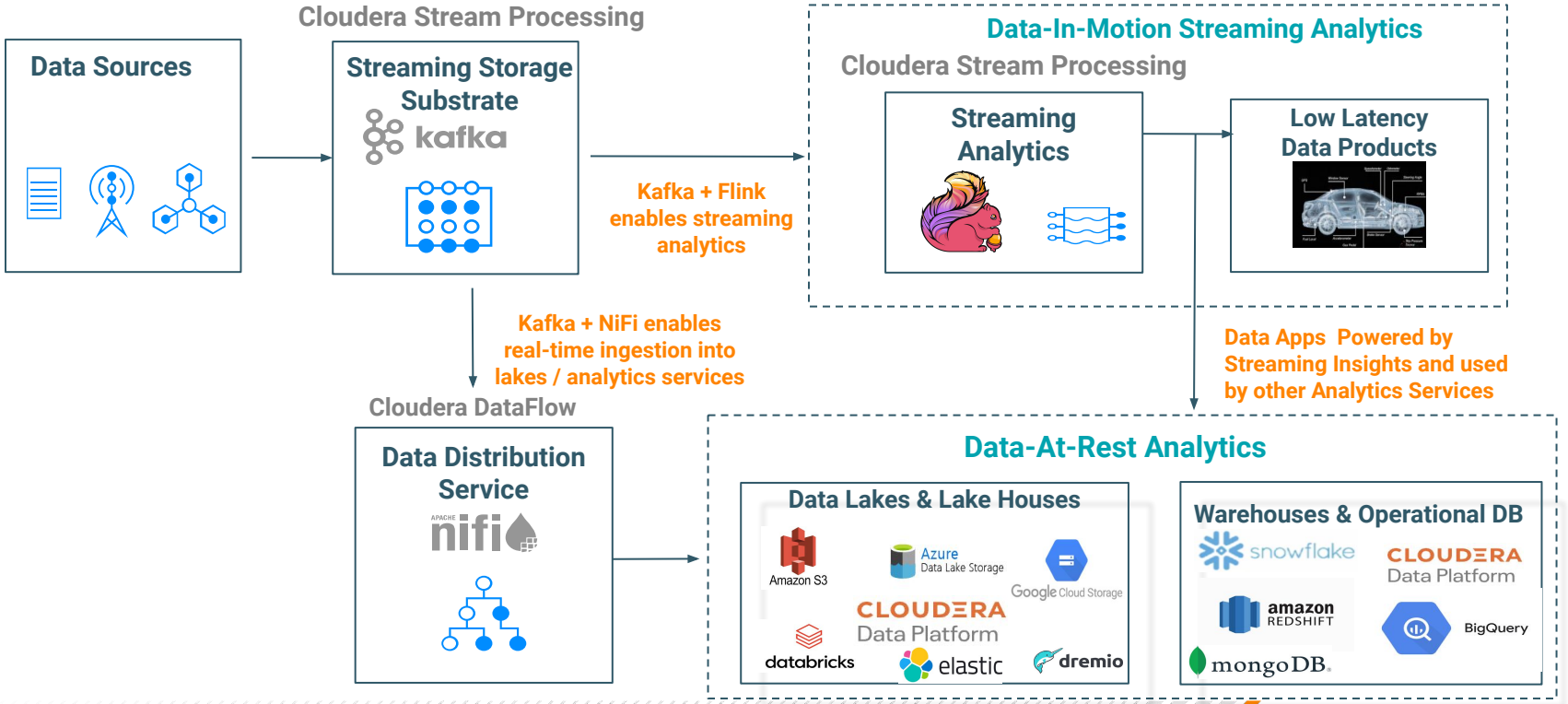
Step 5

Import Project from Git into SSB Prod, Setup Monitoring & Deploy





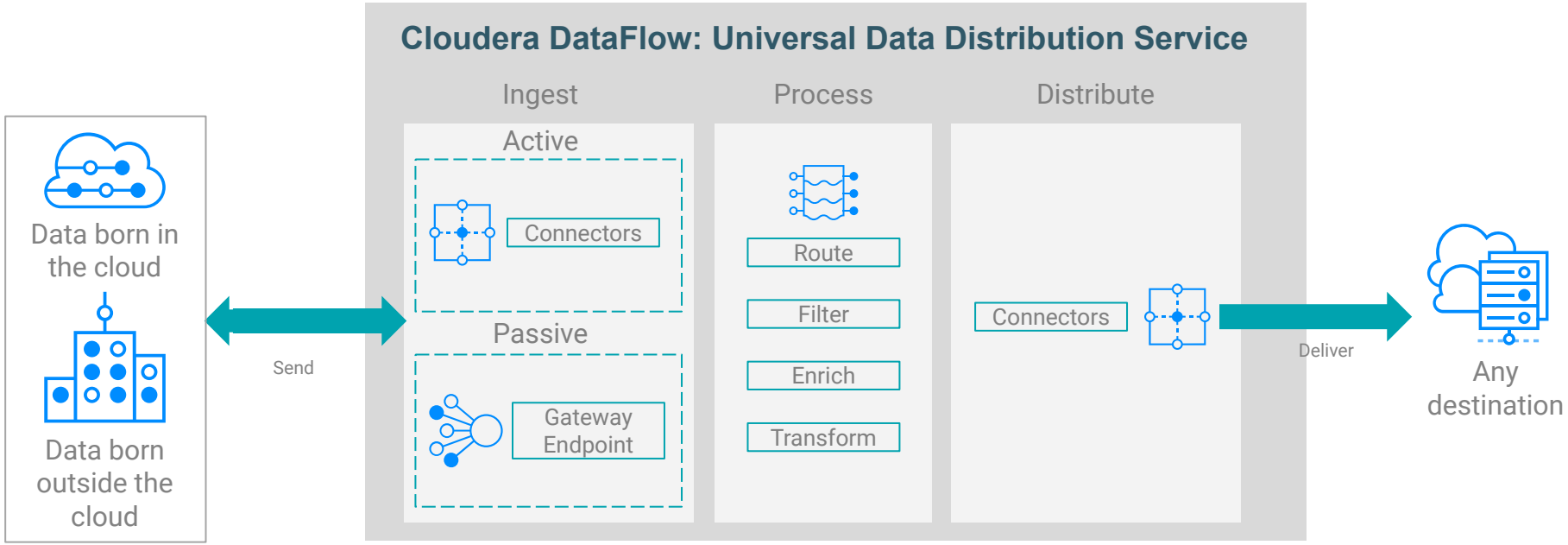
Moving Beyond Draining of Streams Into Lakes: Analytics-in-Stream



DATAFLOW APACHE NIFI

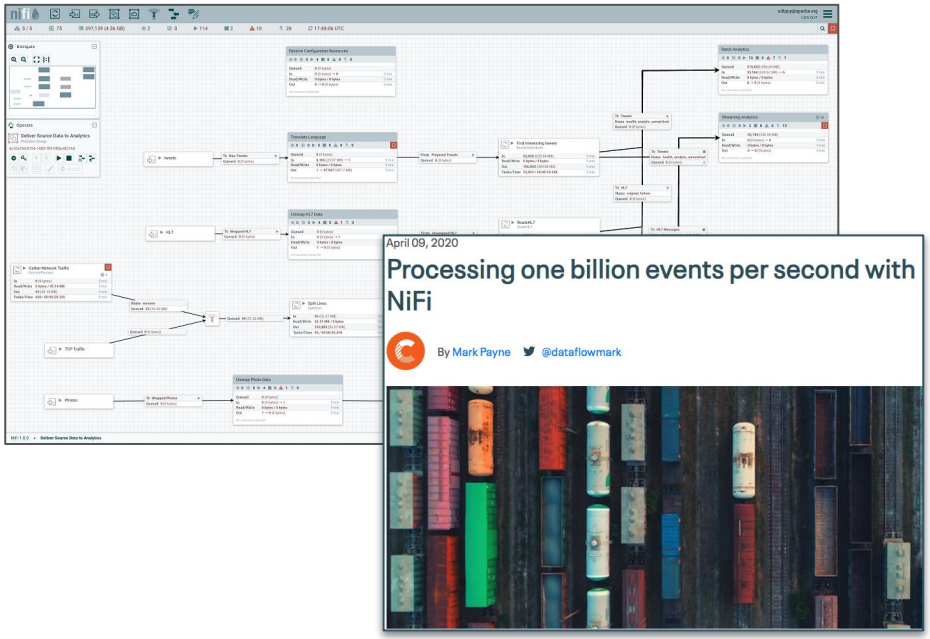
UNIVERSAL DATA DISTRIBUTION WITH CLOUDERA DATAFLOW (CDF)

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

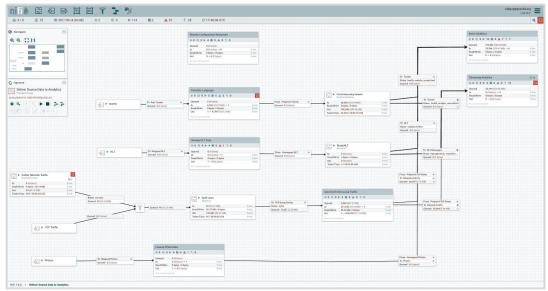


- #1 data ingestion/movement engine
- Strong community
- Product maturity over 11 years
- Deploy on-premises or in the cloud
- Over 400+ pre-built processors
- Built-in data provenance
- Guaranteed delivery
- Throttling and Backpressure

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, HBase, 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

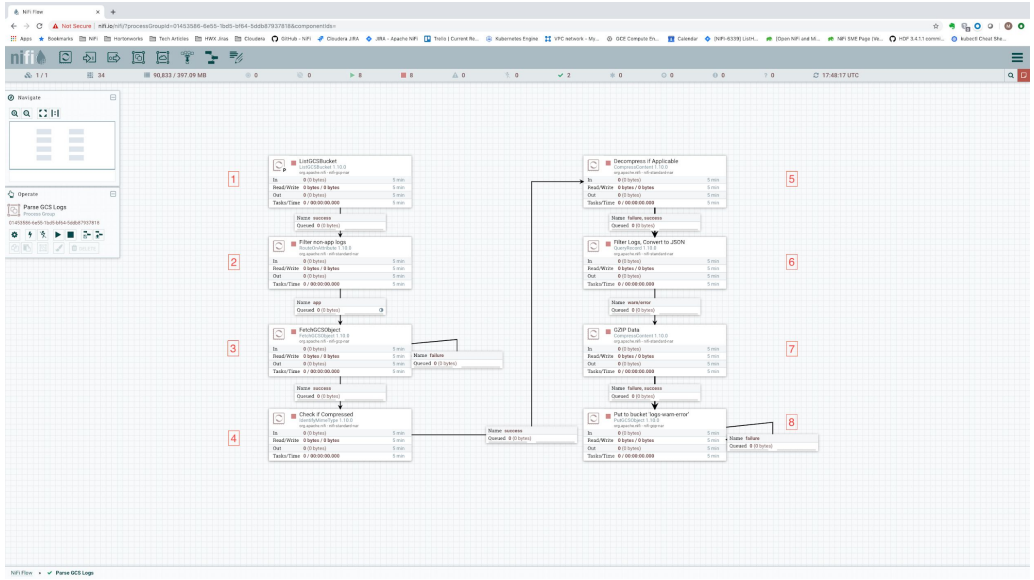
kafka, HIVE, Spark, HBase, APACHE STORM, Solr, SQL, Flink, DynamoDB, cassandra, amazon KINESIS, elastic, amazon S3, mongoDB, slack

- 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

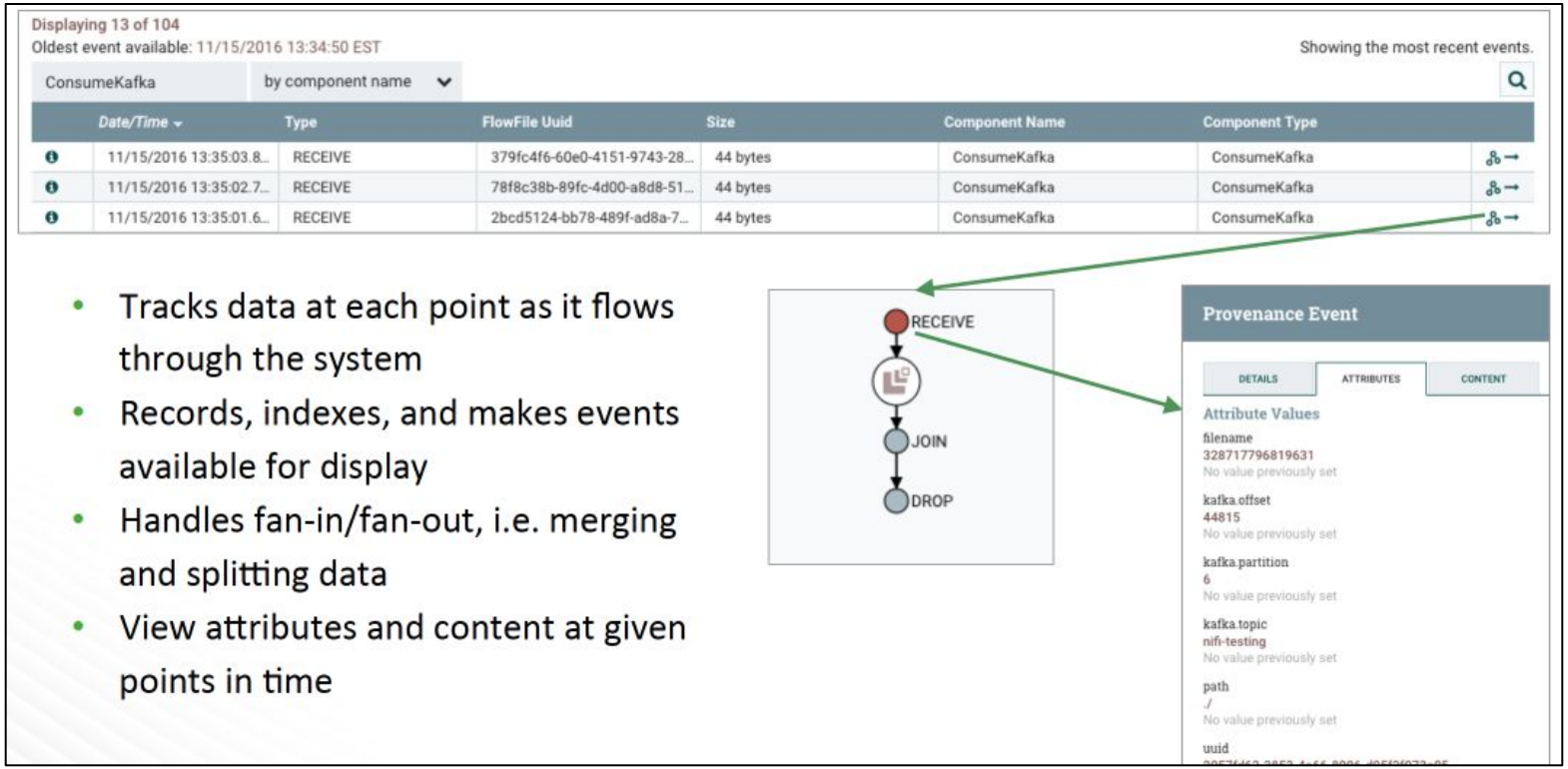
Processing one millions events per second with Apache NiFi

https://blog.cloudera.com/benchmarking-nifi-performance-and-scalability/



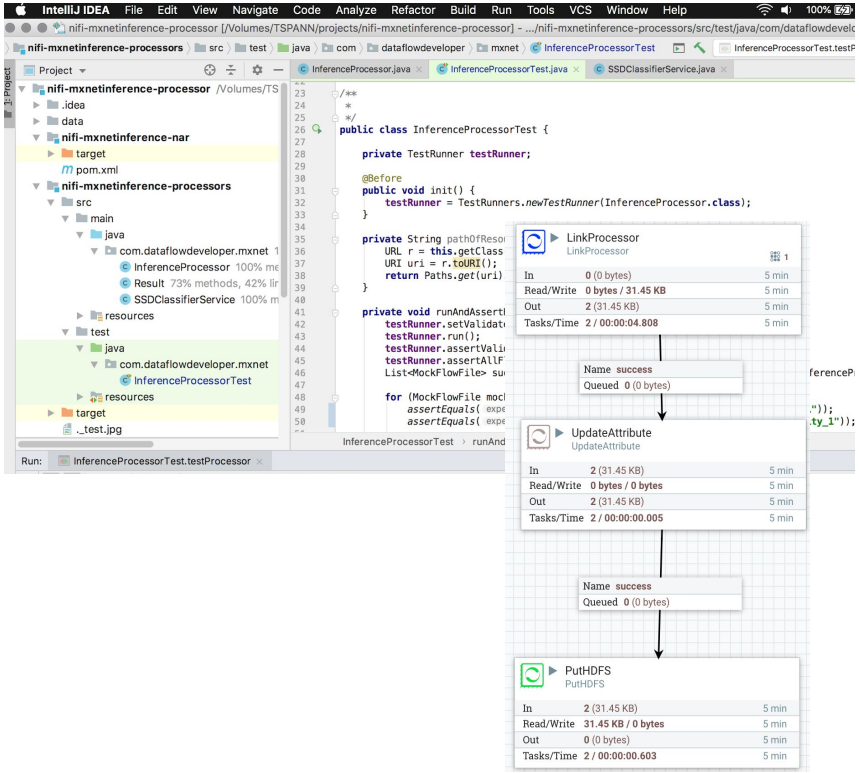
Nodes	Data rate/sec	Events/sec	Data rate/day	Events/day
1	192.5 MB	946,000	16.6 TB	81.7 Billion
5	881 MB	4.97 Million	76 TB	429.4 Billion
25	5.8 GB	26 Million	501 TB	2.25 Trillion
100	22 GB	90 Million	1.9 PB	7.8 Trillion
150	32.6 GB	141.3 Million	2.75 PB	12.2 Trillion

PROVENANCE



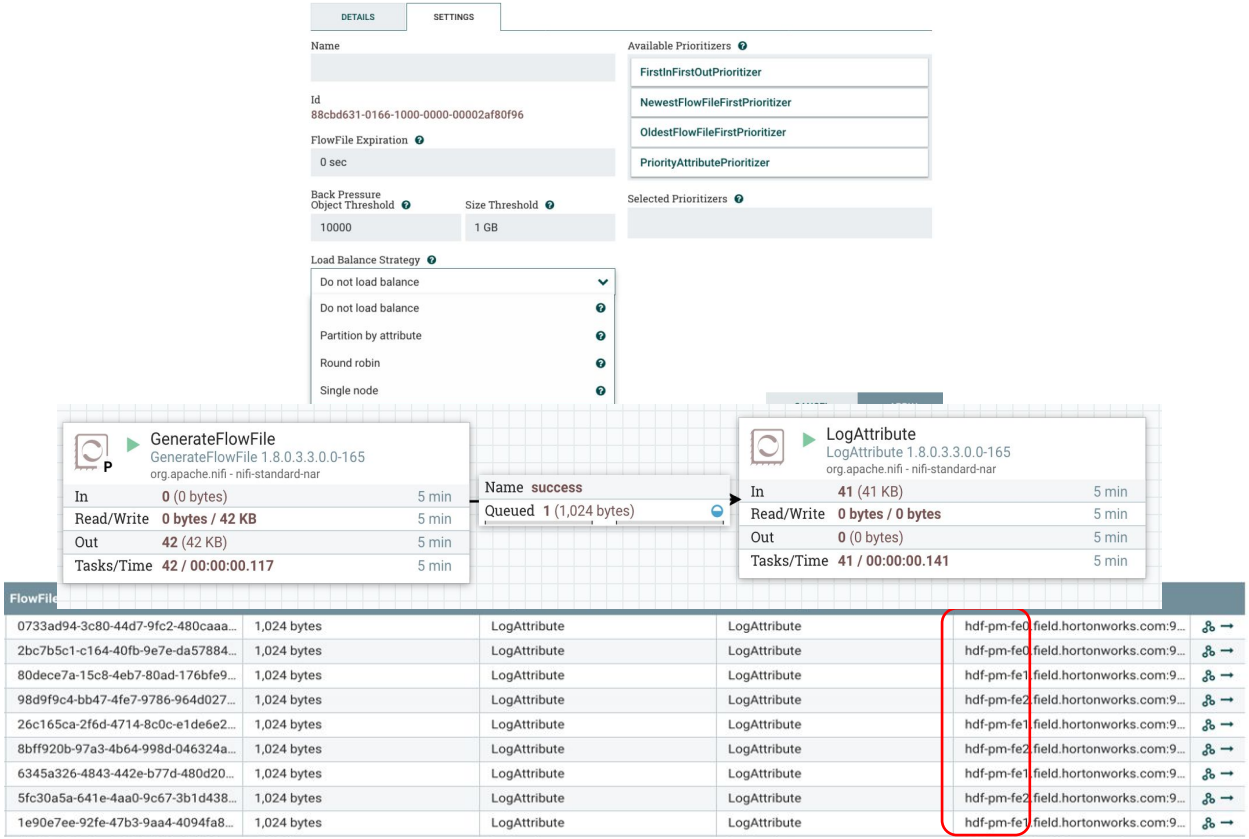
EXTENSIBILITY

- 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

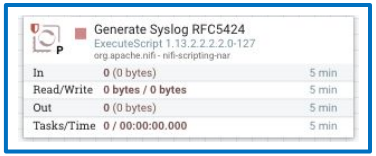


NiFi Load Balancing

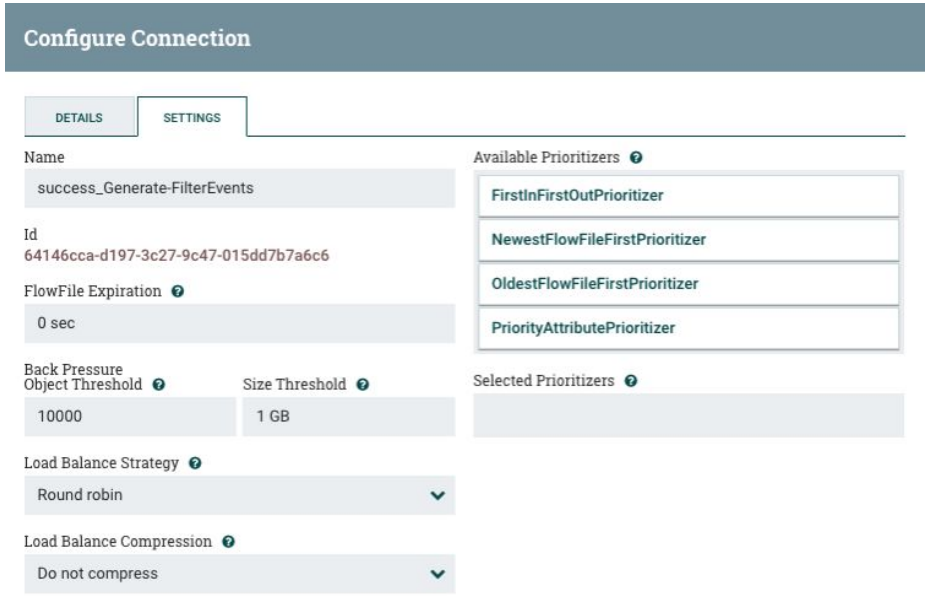
- Improve NiFi cluster throughput
- Defined at connection level
- Configurable balancing strategies
- Critical for scale up paradigm in Kubernetes
- Alleviates S2S balancing “hack” customers use



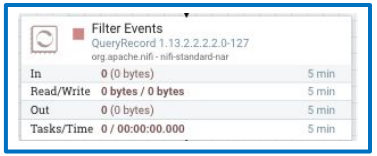
QUEUE CONFIGURATION



- **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.

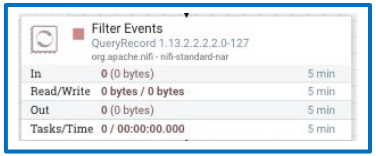


RECORD-ORIENTED DATA WITH NIFI

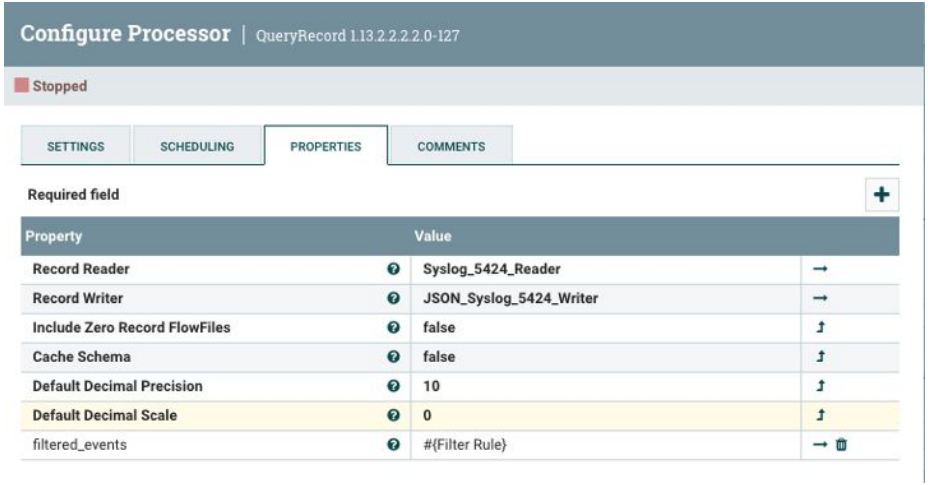


- **Record Readers** - Avro, CSV, Grok, IPFIX, JSAN1, JSON, Parquet, Scripted, Syslog5424, Syslog, WindowsEvent, XML
- **Record Writers** - Avro, CSV, FreeFromText, Json, Parquet, Scripted, XML
- Record Reader and Writer support referencing a schema registry for retrieving schemas when necessary.
- Enable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



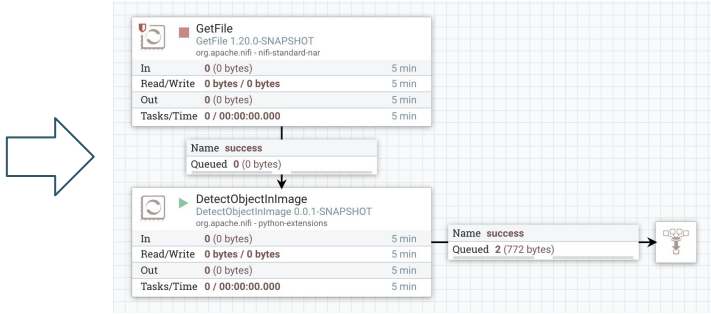
- Evaluates one or more SQL queries against the contents of a FlowFile.
- This can be used, for example, for field-specific filtering, transformation, and row-level filtering.
- Columns can be renamed, simple calculations and aggregations performed.
- The SQL statement must be valid ANSI SQL and is powered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



Apache NiFi with Python Custom Processors

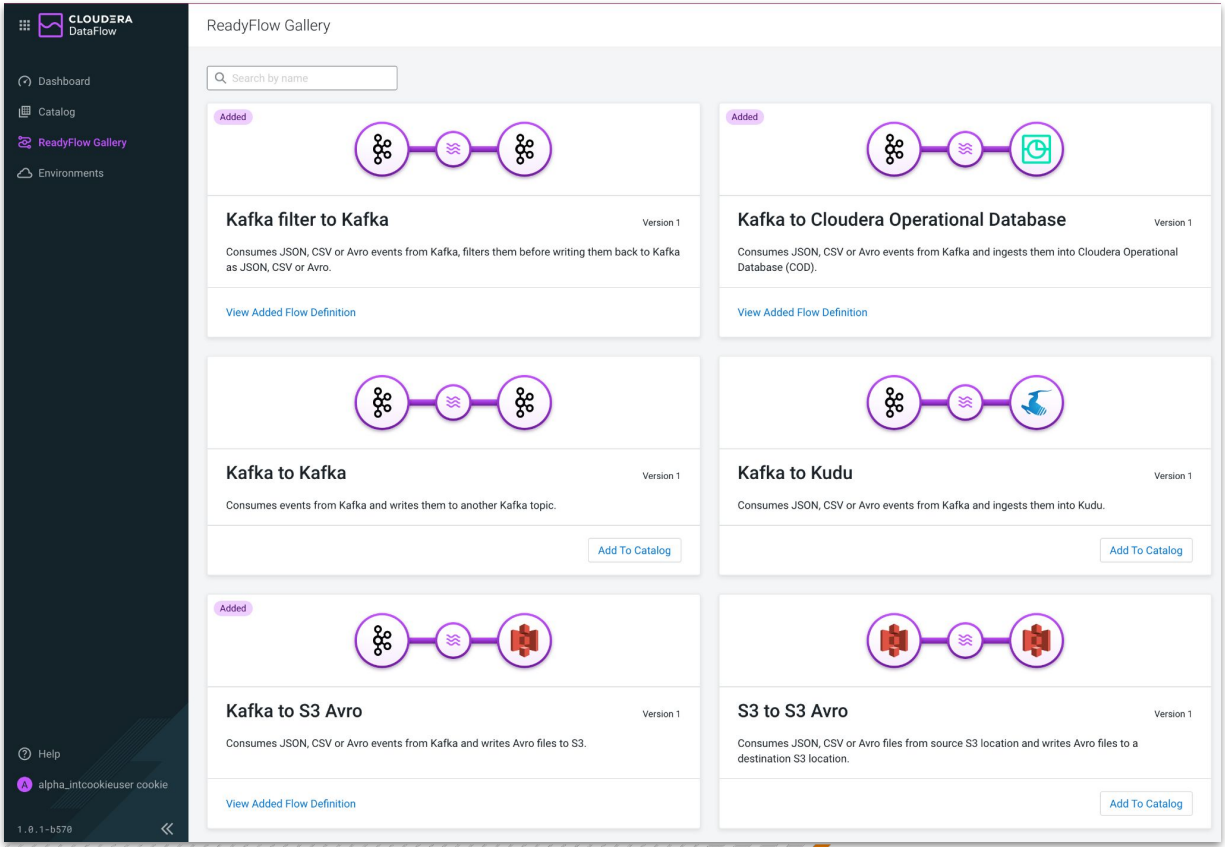
Python as a 1st class citizen

```
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:
```



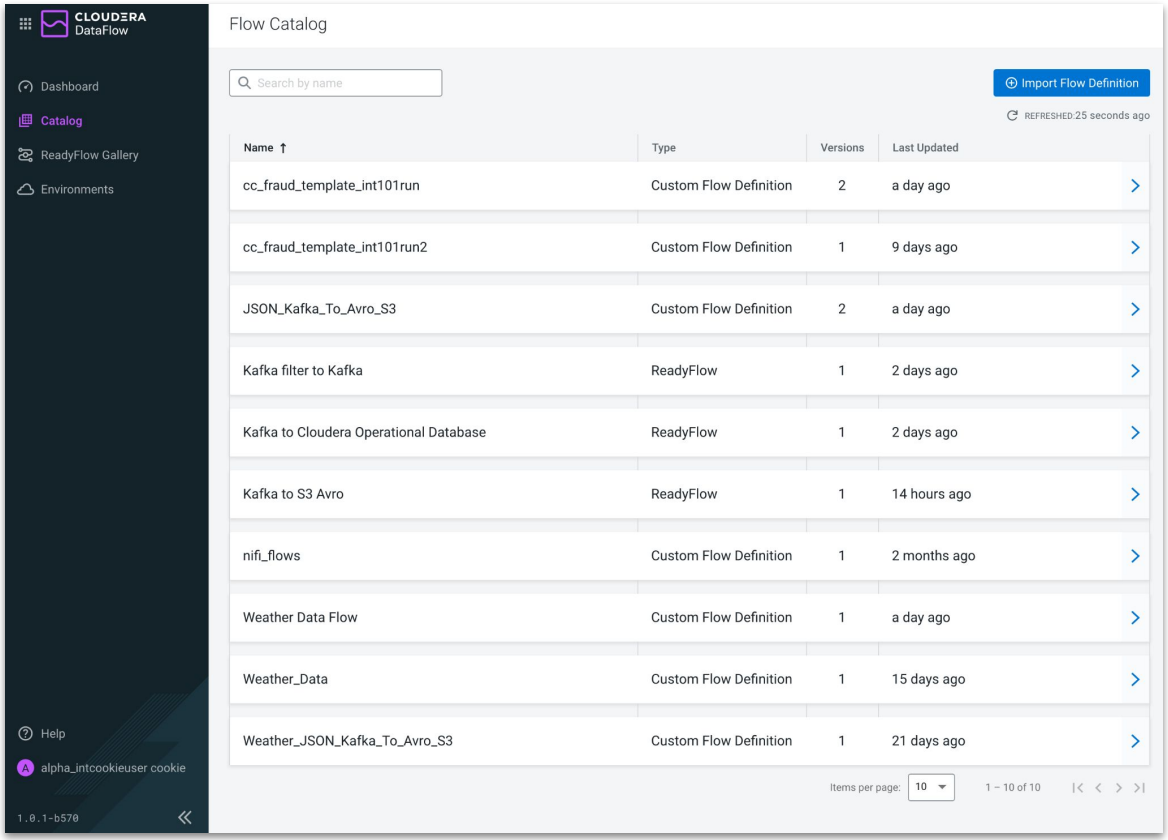
READYFLOW GALLERY

- 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

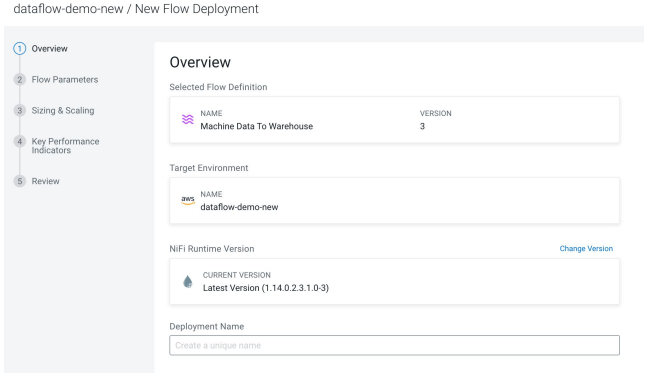
- Central repository for flow definitions
- Import existing NiFi flows
- Manage flow definitions
- Initiate flow deployments



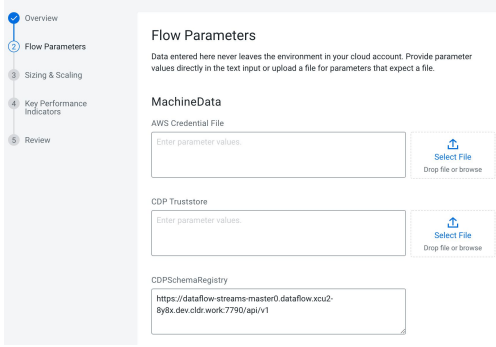
DEPLOYMENT WIZARD

- 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

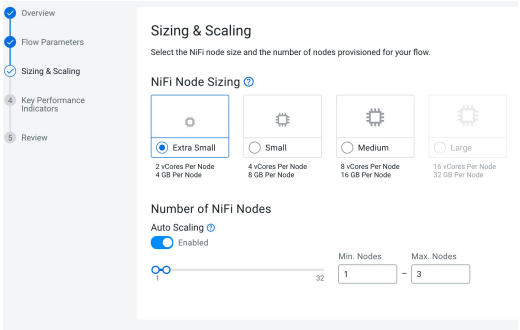
Start Deployment Wizard



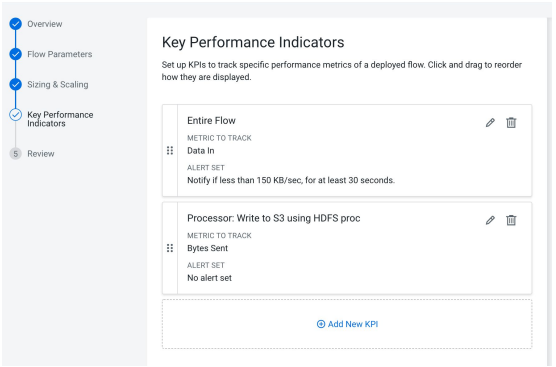
Provide Parameters



Configure Sizing & Scaling



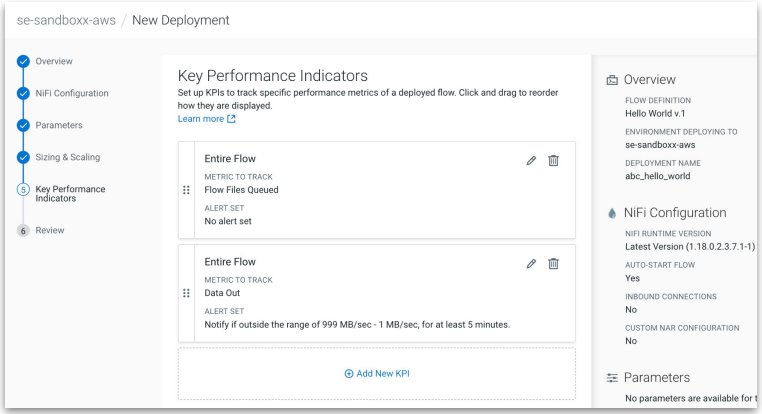
Define KPIs



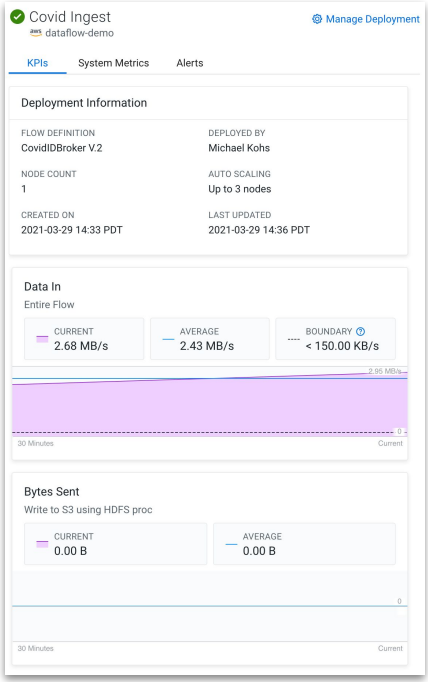
KEY PERFORMANCE INDICATORS

- 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

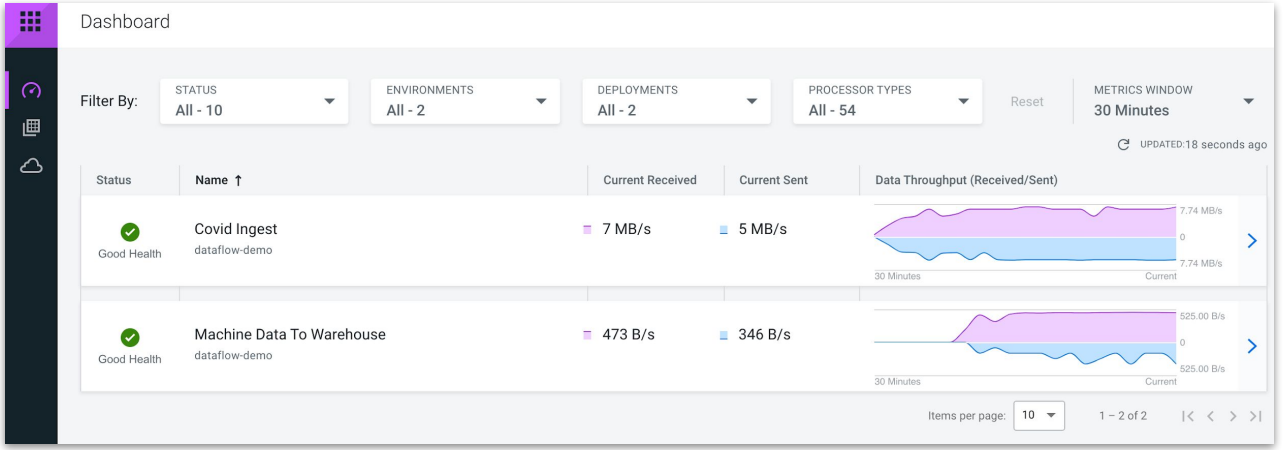


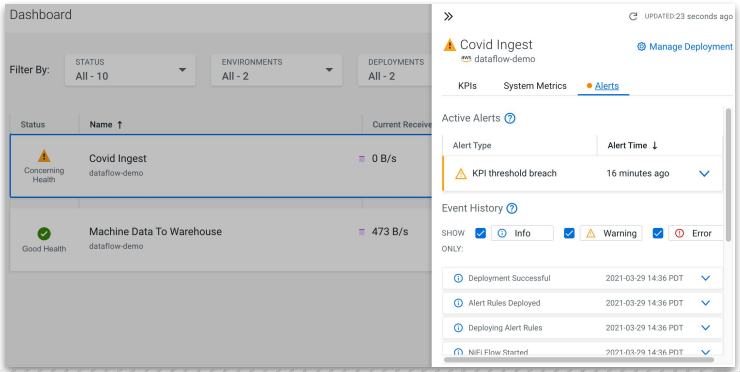
KPI Monitoring

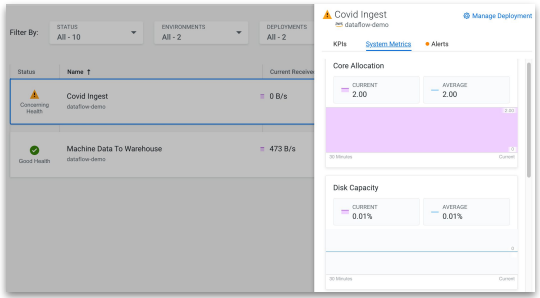


DASHBOARD

- 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

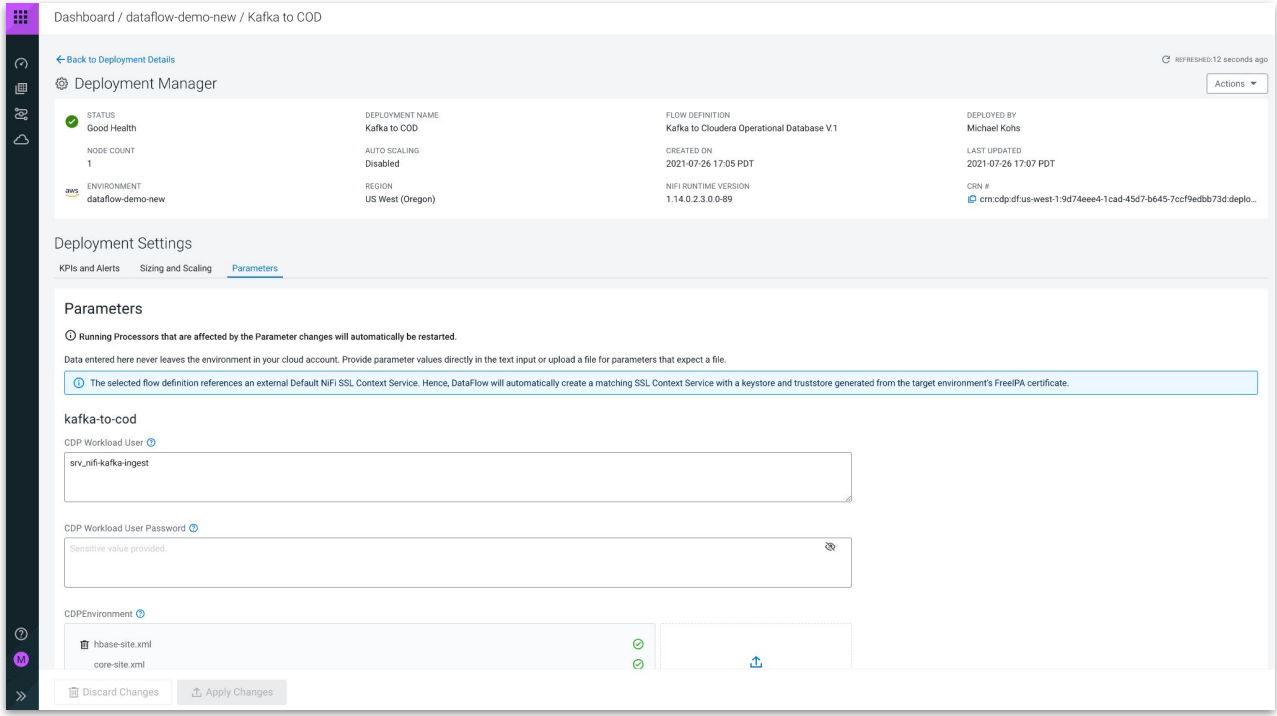






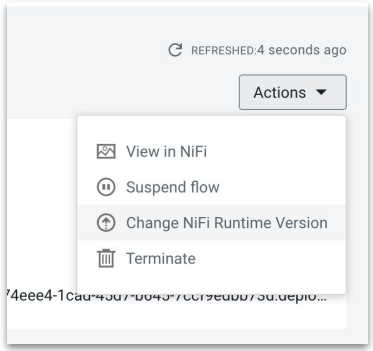
DEPLOYMENT MANAGER

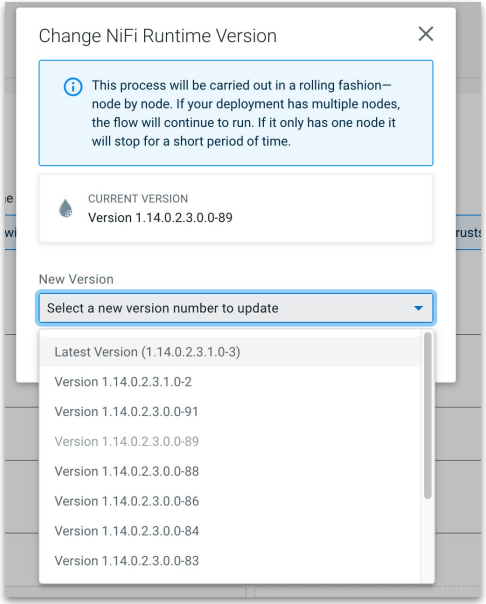
- Manage flow deployment lifecycle (Suspend/Start/Terminate)
- Add/Edit KPIs
- Change sizing configuration
- Update parameters
- Change NiFi version of the deployment
- Gateway to NiFi canvas



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)

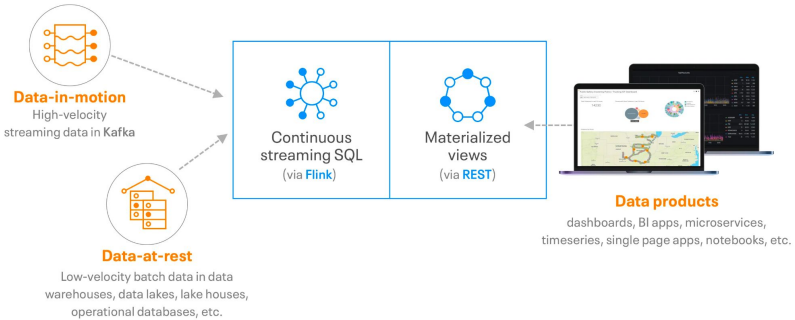




BEST PRACTICES

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)



Streaming Solutions

When to use what?

Routing vs Analytics
- Listeners
- Joins
- In-Memory

Operational Load

Current Skills

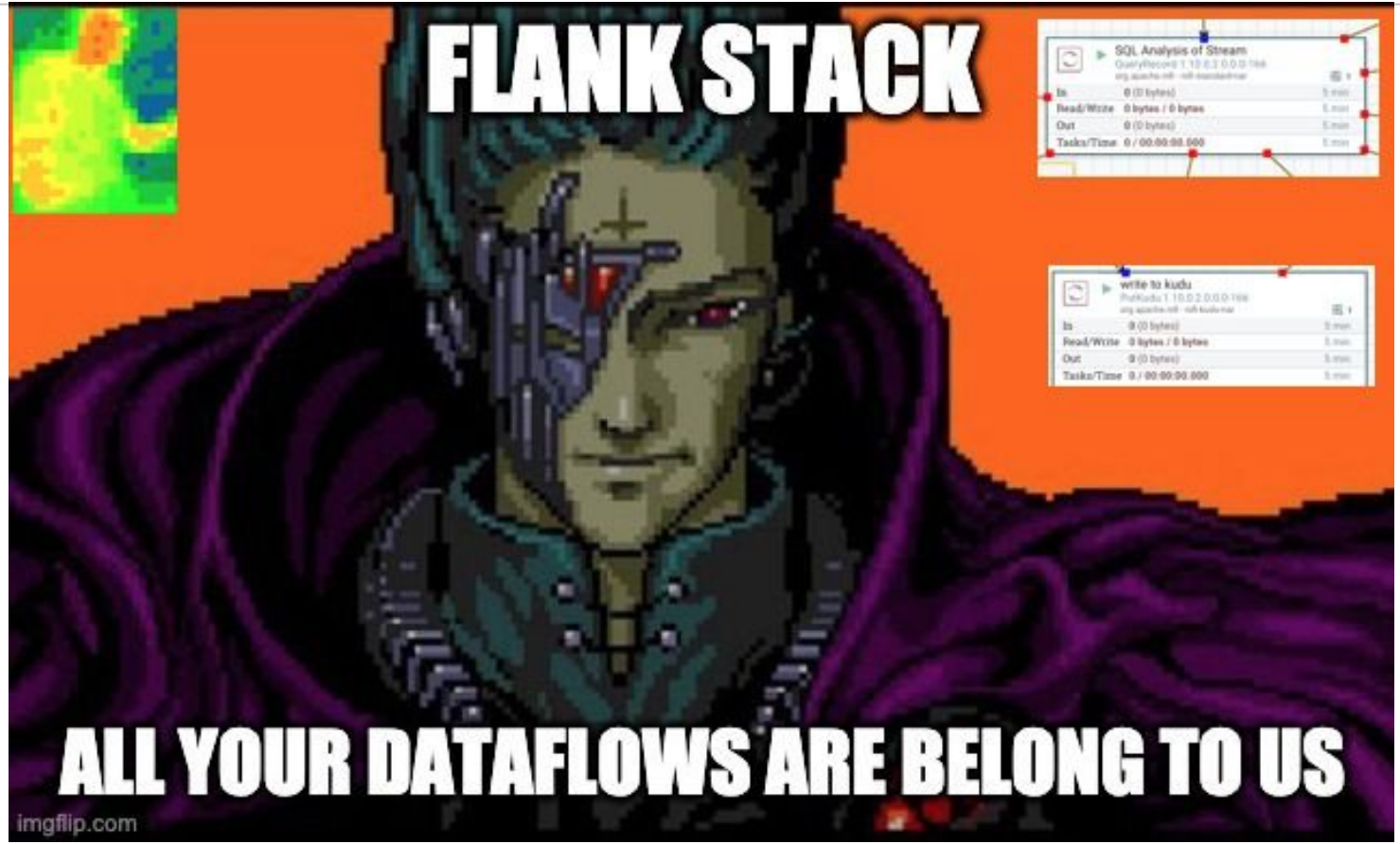
Use NiFi
- Doing more than just Syndication
- Not just small Kafka sized events
- Edge Management is needed
- Listener Type use cases that bind to ports
- Lightweight ETL, Lineage, Provenance, Message Replay

Use Flink
- Joining Streams
- Windowing
- Late Data Handling
- Streaming Analytics

Use KConnect
- Kafka Centric
- In-Memory Stateless

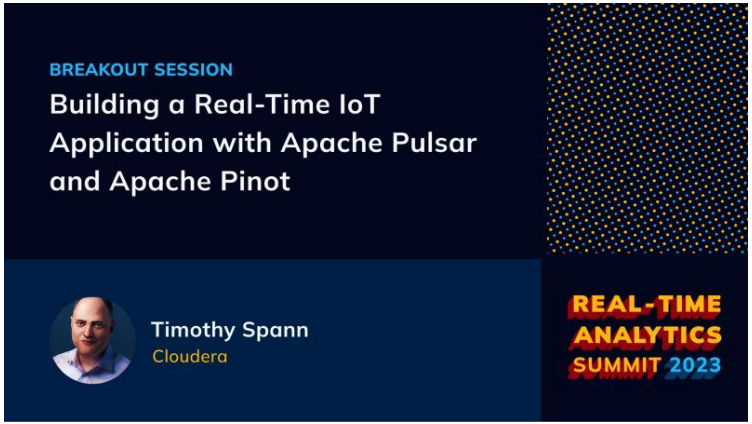
RESOURCES AND WRAP-UP

Resources



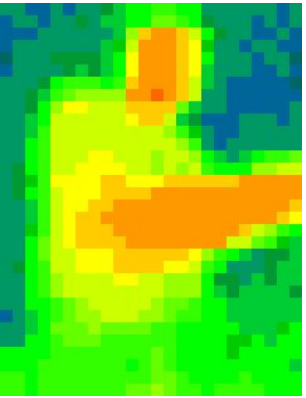
Upcoming Events

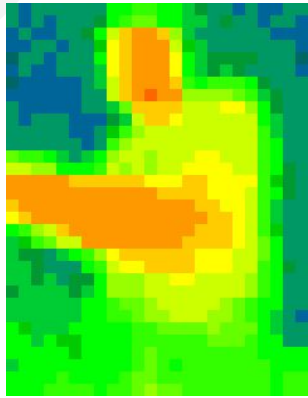
April 26



May 9

May 10





THANK YOU