

CLOUDERA

Cloudera Data Flow CDF Workshop

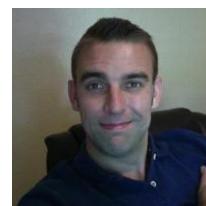
Data in Motion Field Team



Welcome to our Cloudera Data Flow Workshop

Who are we?

Cloudera Data in Motion Field Team



@PaasDev



Welcome to Your Local Cloudera Team

Who are we?

Camila Hiskey

Ifi Derekli, Senior Solutions Engineer - ifi@cloudera.com

Marty Lurie

Eran Orgad

William Brooks, Solutions Engineer - wbrooks@cloudera.com  @wcbdata

Today's Lead

Who am I?

Cloudera Data in Motion Field Engineer



@PaasDev

DZone Zone Leader and Big Data MVB;
Princeton NJ Future of Data Meetup;
ex-Pivotal Field Engineer;
Author of Apache Kafka RefCard
<https://github.com/tspannhw>
<https://www.datainmotion.dev/>



AGENDA

Cloudera DataFlow Overview

Cloudera Flow Management

Cloudera Edge Management

Cloudera Stream Processing

Cloudera Stream Analytics

Labs Architecture

Summary

AGENDA

8:30am-9:00am – Check In & Complimentary Breakfast

9:00am-9:15am – Welcome & Kickoff

9:15am-9:30am – Cloudera DataFlow (CDF)

9:30am-10:30am – Cloudera Flow Management - NiFi (Core concepts, Designing Data Flows)

10:30am-10:45am – Break

10:45am-11:15am – NiFi (Extending NiFi, Data Distribution, CI/CD)

11:15am-12:00am – Lab

12:00am-12:30pm – MiNiFi and Cloudera Edge Management (CEM)

12:30pm-1:15 pm – Complimentary Lunch/Lab

1:15pm-1:45pm – Cloudera Stream Processing (CSP) - Kafka, Schema Registry and Streams Messaging Manager (SMM)

1:45pm-3:30pm – Lab

3:30pm-4:00pm – Kafka Streams

4:00pm-4:30pm - SRM - Mirror Maker 2, Streams and Flink

4:45pm-5:00pm – Summary And Wrap-up

Labs are available at

<https://github.com/asdaraujo/edge2ai-workshop>

Your Environment:

<http://35.175.115.227>

Labs Summary:

- Lab 1 - On the Gateway host, run a simulator to send IoT sensors data to the MQTT broker.
- Lab 2 - On the Gateway host, configure and start MiNiFi, which will read from the MQTT broker, filter and forward to the NiFi cluster.
- Lab 3 - Create the MiNiFi flow on the Edge Flow Manager and publish it for the MiNiFi agent to start sending data to the NiFi cluster.
- Lab 4 - On Schema Registry, register the schema describing the data generated by the IoT sensors.
- Lab 5 - On the NiFi cluster, prepare the data and send it to the Kafka cluster.
- Lab 6 - On the Streams Messaging Manager (SMM) Web UI, monitor the Kafka cluster and confirm data is being ingested correctly.
- Lab 7 - Use the Edge Flow Manager to update existing edge flows and perform additional processing on the edge
- Lab 8 - Use NiFi to process each record, calling the Model endpoint and save results to Kudu.

<https://github.com/asdaraaujo/edge2ai-workshop#labs-summary>

LABS

Step by step instructions

<https://github.com/asdaraujo/edge2ai-workshop>

asdaraujo / edge2ai-workshop

Code Issues Pull requests Projects Wiki Security Insights Settings

Edge2AI Workshop

Manage topics

29 commits 2 branches 0 releases 2 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

asdaraujo Added SMM setup for truck demo

Latest commit 24c7982 22 hours ago

data Initial commit last month

images A few README improvements/corrections 21 days ago

setup Added SMM setup for truck demo 22 hours ago

.gitignore Added a web portal to serve environments 5 days ago

README.adoc A few README improvements/corrections 21 days ago

cdswiot_exp.py Initial commit last month

cdswiot_model.py Initial commit last month

mqtt.iot.config Added new Lab to filter data on the edge 23 days ago

mqtt.iot_simulator.py Added new Lab to filter data on the edge 23 days ago

sensor.avsc Added one more flow to the workshop and fixed a few more bugs 21 days ago

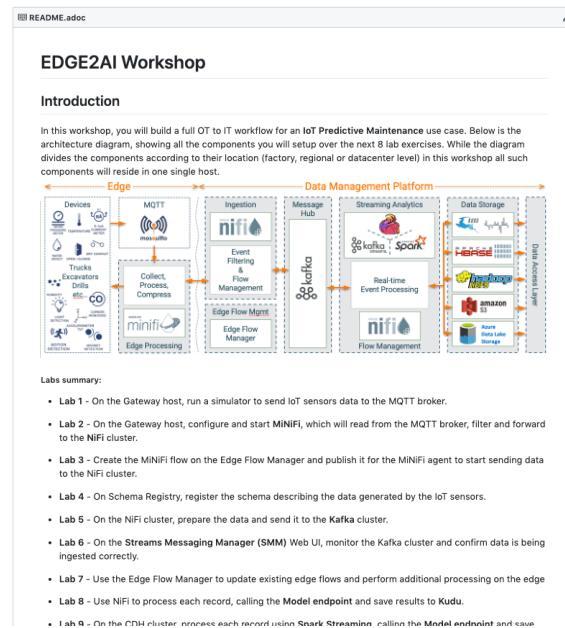
spark.iot.py Added one more flow to the workshop and fixed a few more bugs 21 days ago

README.adoc

EDGE2AI Workshop

Introduction

In this workshop, you will build a full OT to IT workflow for an IoT Predictive Maintenance use case. Below is the architecture diagram, showing all the components you will setup over the next 8 lab exercises. While the diagram divides the components according to their location (factory, regional or datacenter level) in this workshop all such components will reside in one single host.



HOW DO I GET ACCESS TO A CLUSTER?

Register for a Single-Node Cluster in AWS

admin/admin

<http://35.175.115.227>

EDGE2AI Workshop Home

Sign In

Email

araujo@cloudera.com

Sign In

EDGE2AI Workshop Home

Register

Confirm email

araujo@cloudera.com

Full name

Andre Araujo

Company

Cloudera

Register

Cancel

EDGE2AI Workshop Home



Andre Araujo
Cloudera

Logout

Cloudera Manager <http://34.215.190.44:7180/>

Edge Flow <http://34.215.190.44:10080/efm/ui/>

NiFi <http://34.215.190.44:8080/nifi/>

NiFi Registry <http://34.215.190.44:18080/nifi-registry/>

Schema Registry <http://34.215.190.44:7788/>

SMM <http://34.215.190.44:9991/>

Hue <http://34.215.190.44:8888/>

Cloudera Data Science Workbench <http://cdsw.34.215.190.44.nip.io/>

SSH Connection

Download key:

[Download SSH Key](#)

And then run:

```
chmod 400 workshop.pem  
ssh -i workshop.pem centos@34.215.190.44
```

HOW DO I GET ACCESS TO A CLUSTER? (continued)

The screenshot shows the Cloudera Manager interface with several service components highlighted:

- Cloudera Manager**: Located at the top left, showing the navigation bar and a search bar.
- NiFi Registry / Administration**: A central panel showing Buckets (0), Open Flow Overview (Producers: 30, Topics: 40, Brokers: 1), and a list of Producers (e.g., syndicate-speed-event-avro, syndicate-geo-event-avro).
- Schema Registry**: A separate window showing the Schema Registry interface with a list of topics and their schema details.
- Hue**: A separate window showing the Hue interface with a Projects view.
- NiFi**: A separate window showing the NiFi interface with a Process Group view.
- Cloudera Edge Flow Manager**: A separate window showing the Cloudera Edge Flow Manager interface.
- Streams Messaging Manager (SMM)**: A separate window showing the Streams Messaging Manager interface.

Each highlighted service has an orange square icon with a white 'X' next to its name.

CLOUDERA

© 2019 Cloudera, Inc. All rights reserved. 11

Cloudera Data Platform

- Public, private & hybrid cloud
- Shared data experience
- Powered by open source
- Analytics from the Edge to AI
- Unified data control plane

Control Plane

Identity | Orchestration | Management | Operations

Analytic experiences

DataFlow &
Streaming



Data
Engineering



Data
Warehouse



Operational
Database



Machine
Learning



Data
anywhere



Catalog | Schema | Migration | Security | Governance

Any
Infrastructure



KEY DIFFERENTIATORS

100% open source technology – Only vendor with this strategy; prevents vendor lock-in



300+ pre-built processors – Only product to offer such comprehensive connectivity from edge to enterprise



3 Streaming analytics engines – Only vendor to offer a choice of three streaming analytics engines to customers for all their streaming architecture needs



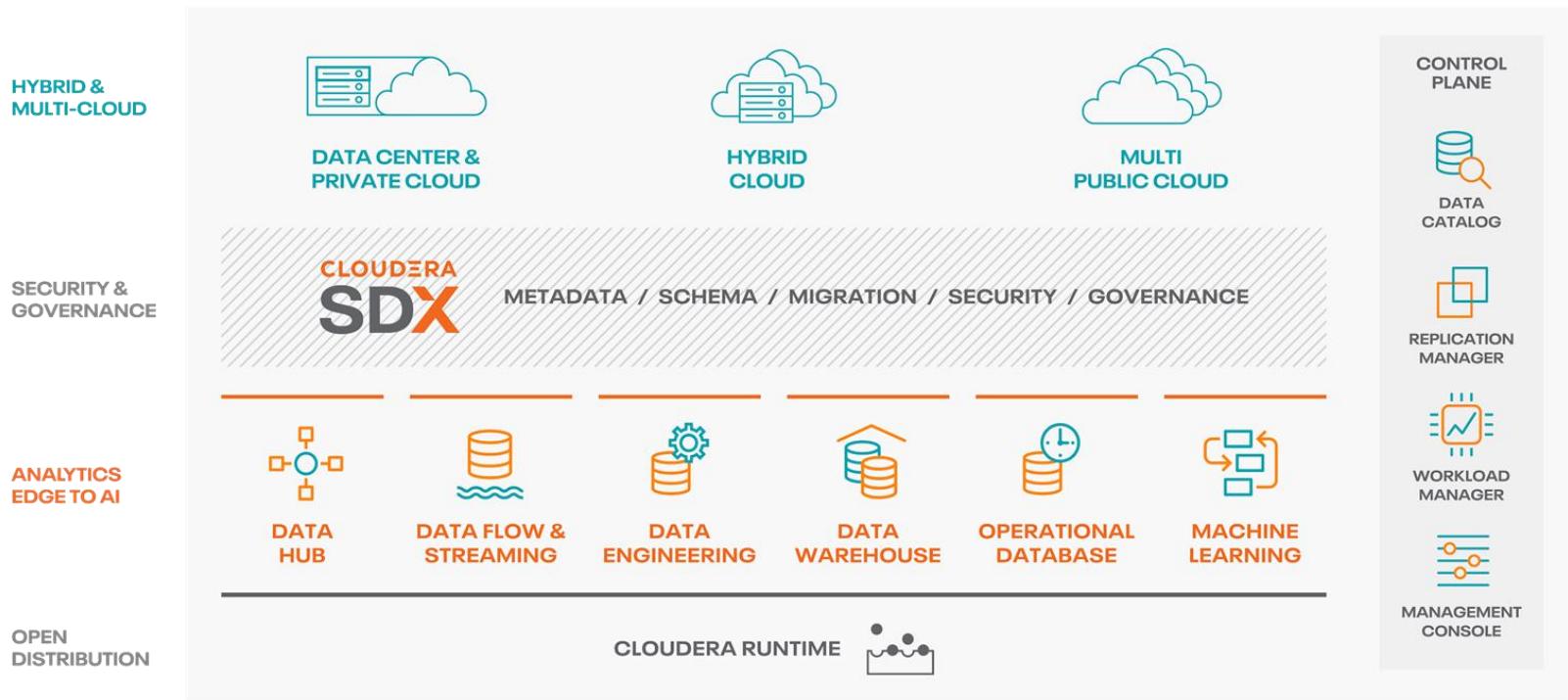
Built-in data provenance – Only product in the market to offer out-of-the-box data provenance on data-in-motion



Comprehensive streaming platform – Only big data vendor to offer a comprehensive streaming platform from real-time data ingestion, transformation, routing to descriptive, prescriptive and predictive analytics.

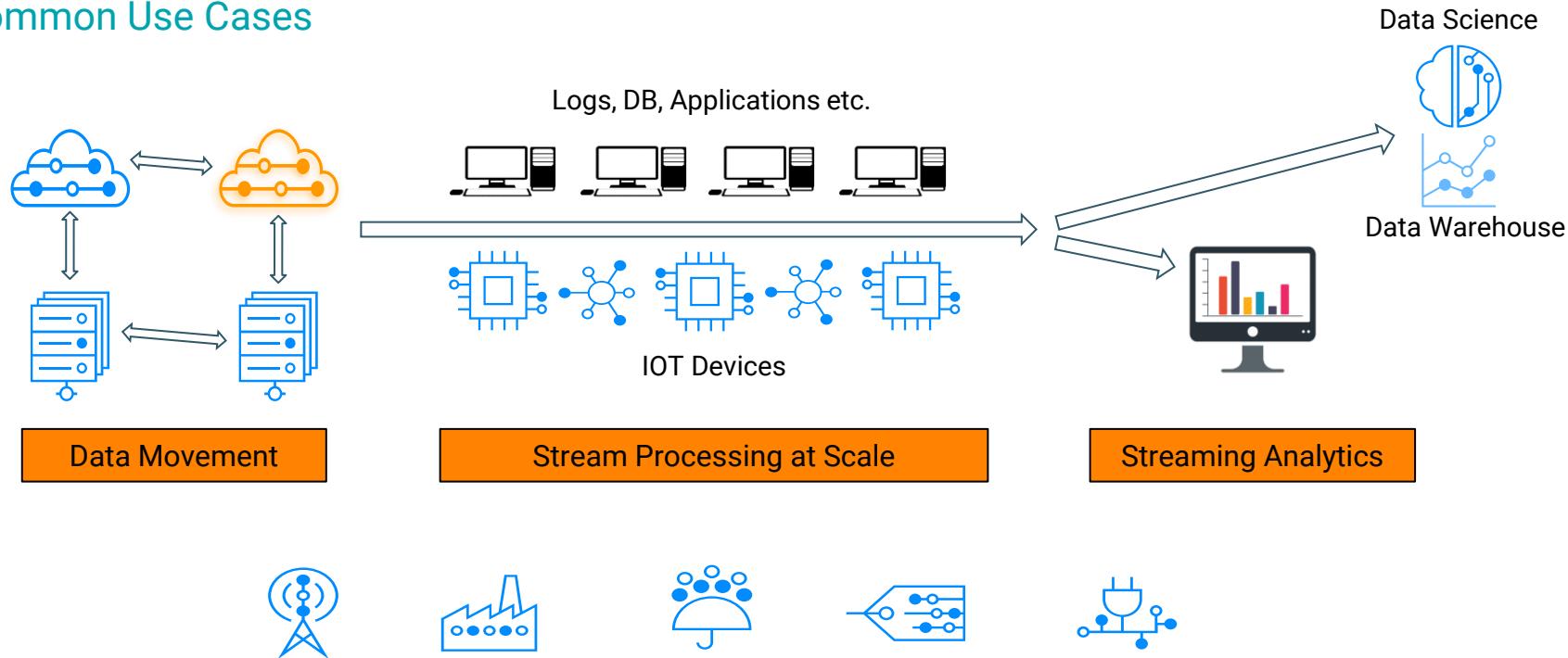


CLOUDERA DATA PLATFORM



Cloudera Data Flow

Common Use Cases



KEY TAKEAWAYS

Cloudera Data Platform

CDF is the only comprehensive streaming data platform for your end-to-end data journey

SMM and SRM

SMM is the cure to “Kafka Blindness” with a single pane of glass across all Kafka clusters

SRM is the answer to Kafka Replicator

Enterprise Capabilities

Only CDF offers out-of-the-box unified security & governance from Edge to AI

KEY CUSTOMER CHALLENGES

DATA INGESTION



High-volume streaming sources

Multiple message formats

Diverse protocols

Multi-vendor devices

REAL-TIME INSIGHTS



Analyzing continuous and rapid streaming data

Handling high streaming data volumes

Running predictive models on streaming data

VISIBILITY



Monitor end-to-end streaming data flows

Troubleshoot bottlenecks

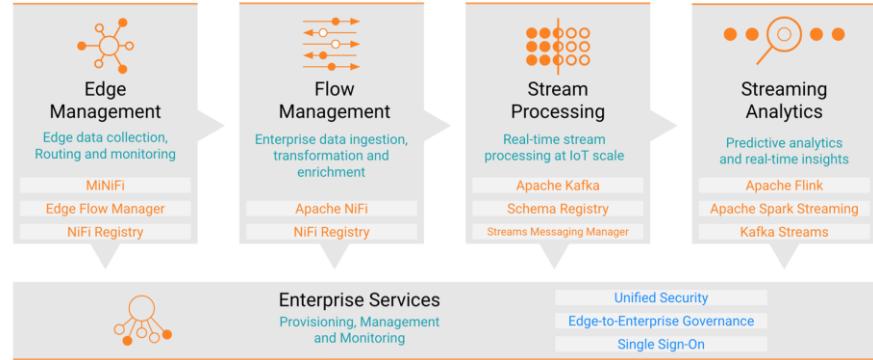
Understand consumption patterns

HISTORY OF CDF

Data-in-Motion:

- Comprehensive real-time streaming data platform
- Manage data-in-motion from edge-to-enterprise
- Power IoT-scale streaming architectures

Cloudera DataFlow Platform



Bring this to the edge with
connected platforms

Enable next generation
Modern Data Architecture

Mid-2000's
NiFi was developed
and used at NSA

2015
Onyara is
acquired
HDF is born

2018
Strong Streaming
Platform

- Support for Kafka 2.0
- SMM is introduced

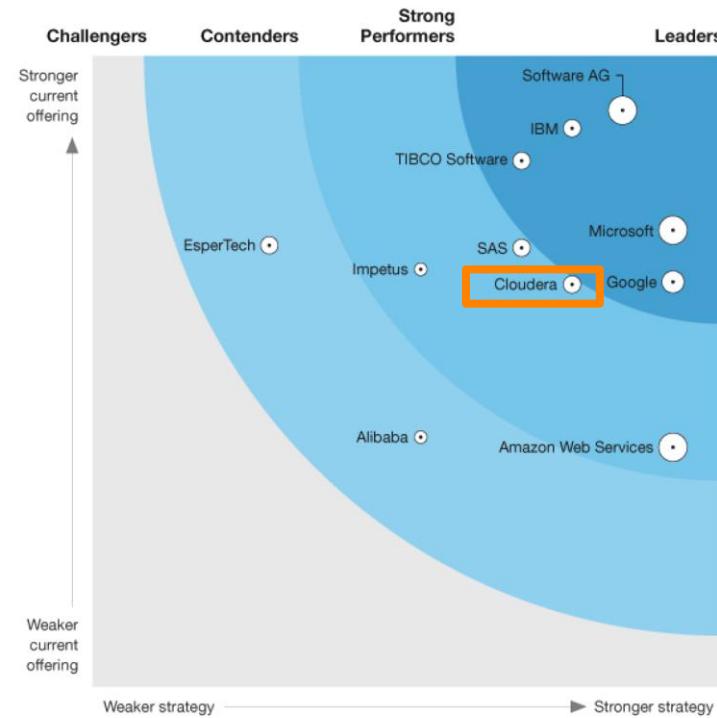
2019
Cloudera merger
Enable Edge
Intelligence

Tomorrow:
Edge-to-AI

CDF Debuts as Strong Performer in 2019 Forrester Wave for Streaming Analytics

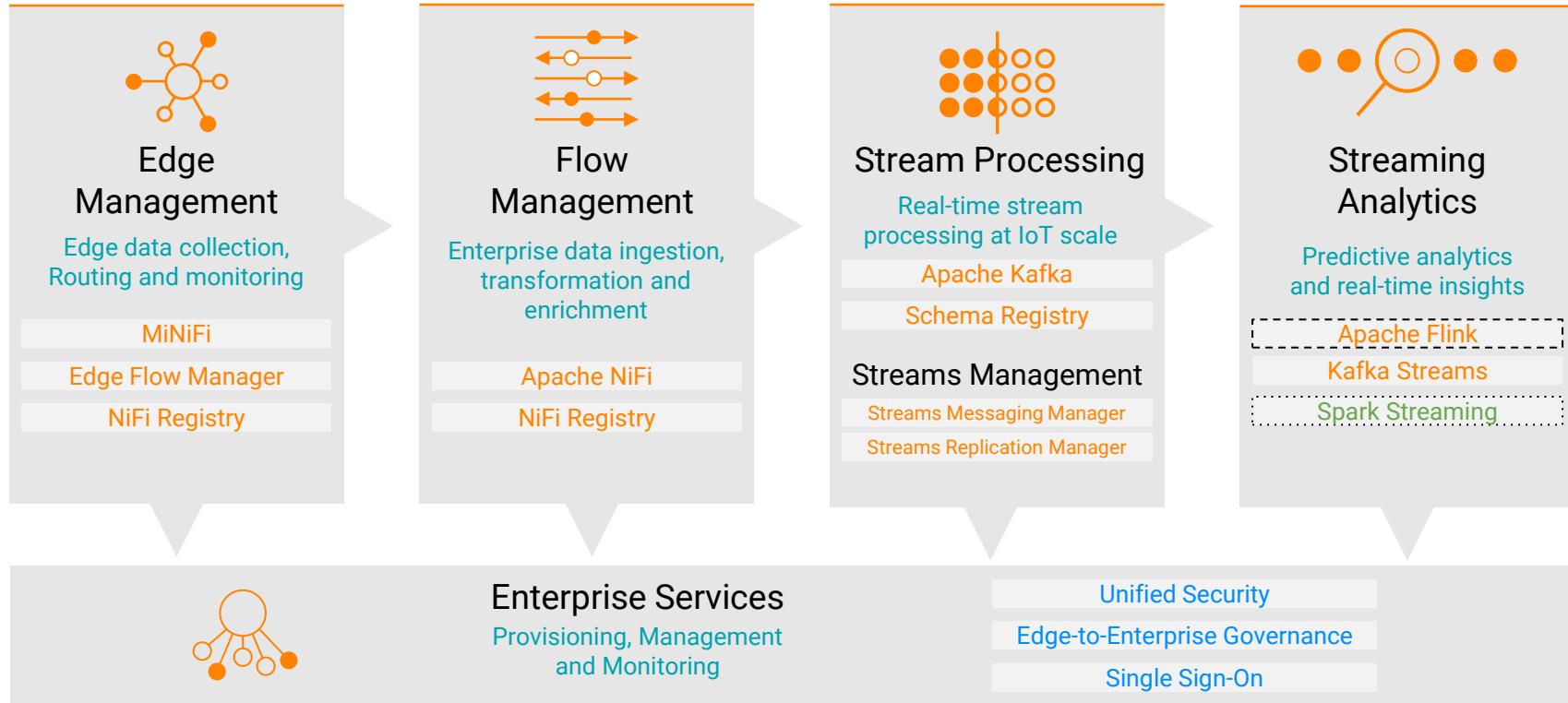
Key Highlights

- Cloudera has been named as a Strong Performer in the Forrester Wave for Streaming Analytics, Q3 2019.
- Our current offering is considered to be on par if not superior to the key cloud players like AWS, Google and Microsoft.
- Our key competitors, Confluent and Streamsets, are missing from this wave. This is a significant competitive advantage for us to press upon in the field.



Data Flow in the Enterprise

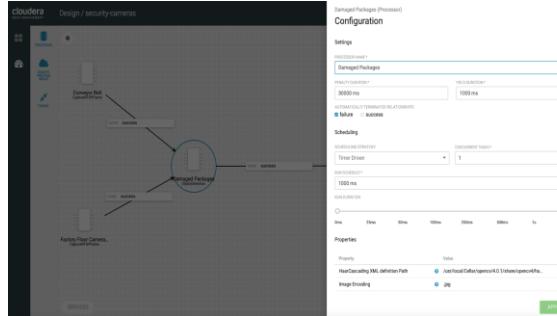
Cloudera DataFlow Platform



Cloudera EDGE Management

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

Edge Flow Manager

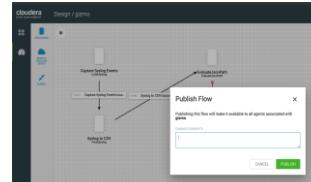


A lightweight edge agent that implements the core features of Apache NiFi, focusing on data collection and processing at the edge

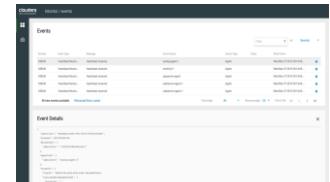
Flow Authorship



Flow Deployment



Flow Monitoring

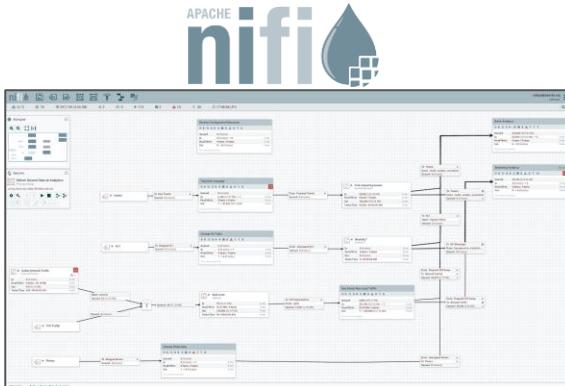


- Small footprint agent with MiNiFi
- Java and C++ agents
- Rich edge processors (edge collection & processing)
- End to end lineage and security

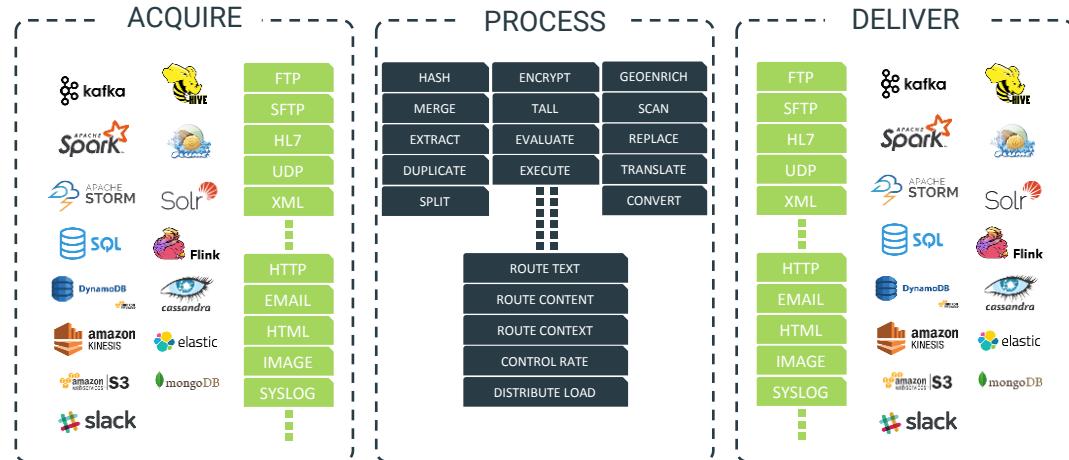
- Central Command and Control (C2)
- Design and deploy to thousands of agents
- Edge Applications lifecycle management
- Multitenancy with Agent classes
- Native integration with other CDF services

Cloudera Flow 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



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



- 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

Stream Processing and Analytics

Providing simpler ways of complex event processing at scale for the enterprise

 **kafka + KStreams**

“Pub & Sub”

Massively Scalable Publish -Subscribe Message Queue, API for building real-time microservices with Kafka Streams

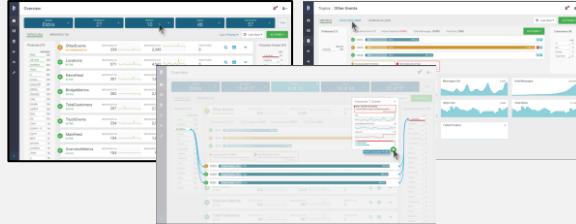
 **Flink**

“Process/Analyze”

Real-Time, Distributed Data and Task Processing Engine

 “Manage & Monitor”

Streams Messaging Manager

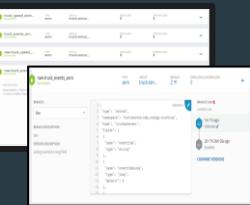


 **Streams Replication Manager (SRM)**

“Mirroring & DR”

Replicate data & configuration in Realtime, Smart clients for easy fail-over & fail-back, Support active active scenarios, Monitoring

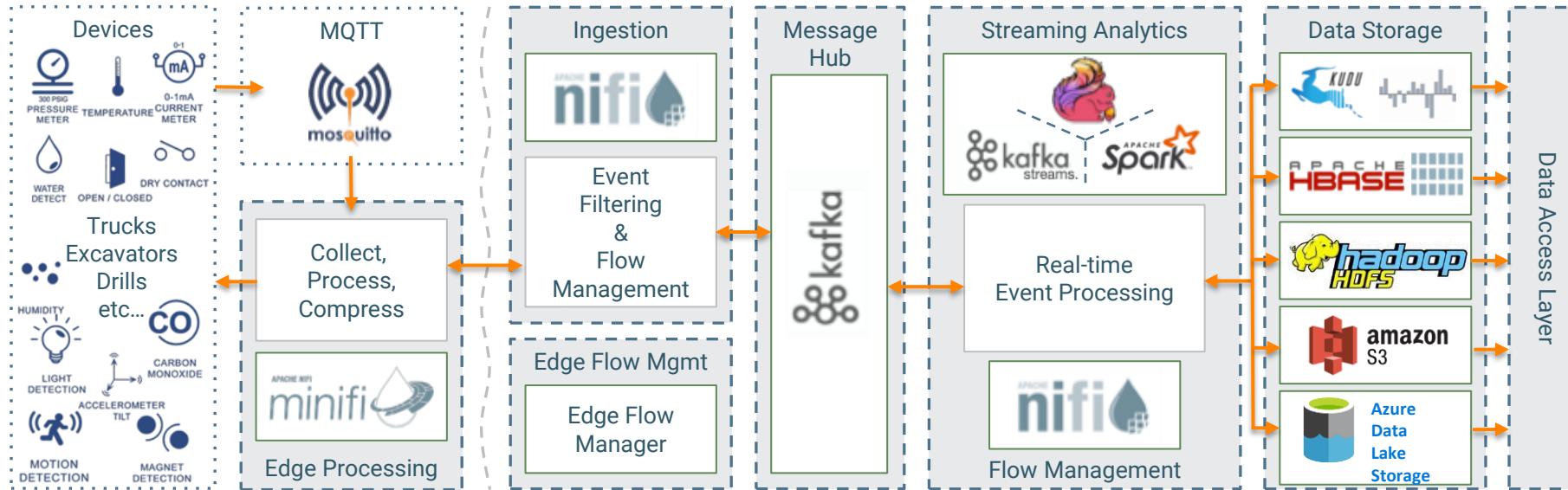
 **SCHEMA REGISTRY**



“Manage”

A shared schemas repository that allows applications to save, retrieve and reuse schemas and interact with each other

And by the way we will be building a complete IoT use case



Cloudera Flow Management

Apache NiFi

DATAFLOW MANAGEMENT:

The systematic process by which data
is acquired from all producers and
delivered to all consumers

NIFI OVERVIEW

WHY NIFI?

- Moving data is multifaceted in its challenges and these are present in different contexts at varying scopes
- Provide common tooling and extensions that are commonly needed but be flexible for extension
 - Leverage existing libraries and expansive Java ecosystem for functionality
 - Allow organizations to integrate with their existing infrastructure
- Empower folks managing your infrastructure to make changes and reason about issues that are occurring
 - Data Provenance to show context and data's journey
 - User Interface/Experience a key component

<https://dzone.com/articles/apache-nifi-10-cheatsheet>

THE NSA YEARS

- Created in 2006
- Improved over eight years
 - Simple initial vision – Visio for real-time dataflow management
- Key Lessons Learned
 - What scale means – down, up, and out
 - The fearsome force known as Compliance Requirements
 - The power of provenance!
 - Operational best-practices and anti-patterns
- NSA donated the codebase to the ASF in late 2014

APACHE NIFI

Key features



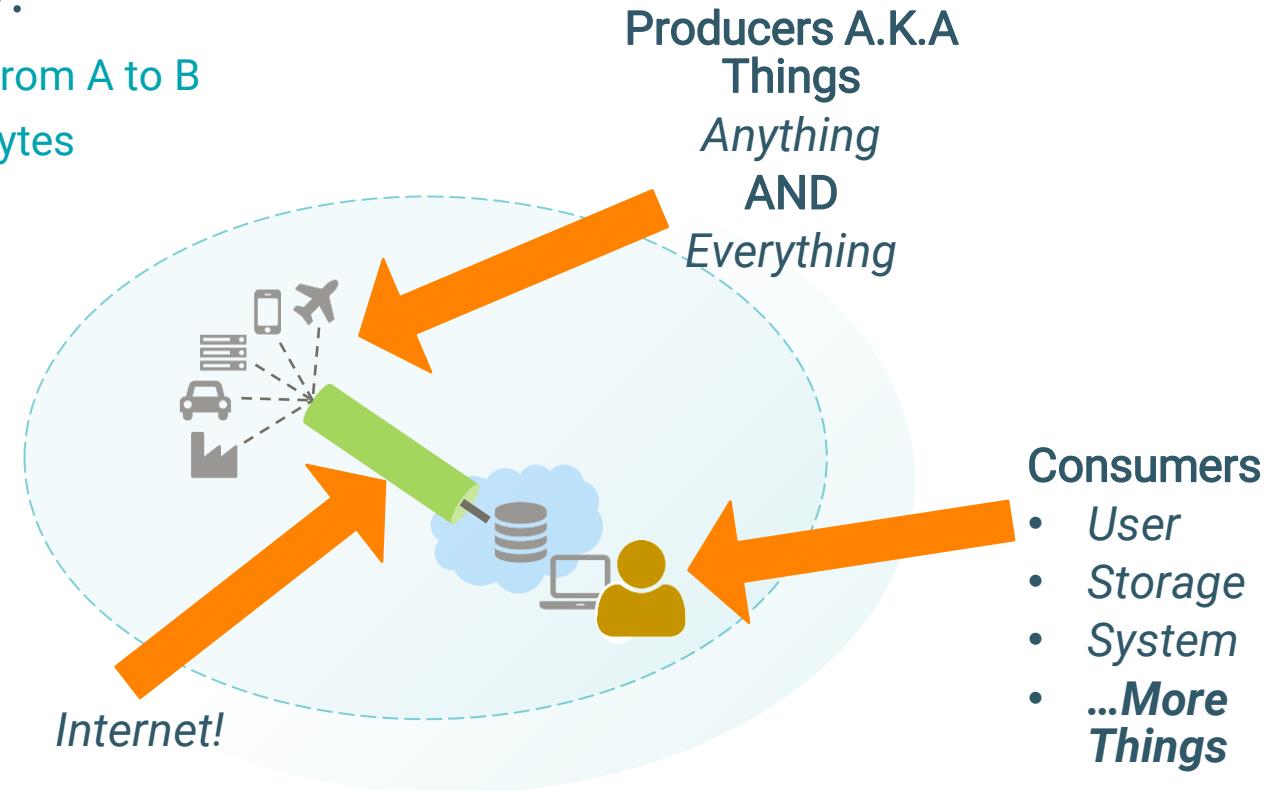
- Guaranteed delivery
- Data buffering
 - Backpressure
 - Pressure release
- Prioritized queuing
- Flow specific QoS
 - Latency vs. throughput
 - Loss tolerance
- Data provenance
- Supports push and pull models
- Recovery/recording a rolling log of fine-grained history
- Visual command and control
- Flow templates
- Pluggable/multi-role security
- Designed for extension
- Clustering

What is Dataflow?

Moving some content from A to B

Content could be any bytes

- Logs
- HTTP
- XML
- CSV
- Images
- Video
- Telemetry

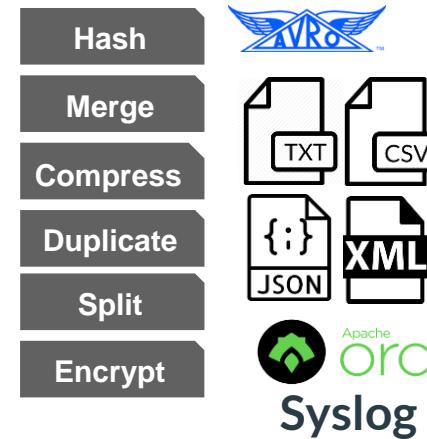


An Overview of Apache NiFi Capabilities

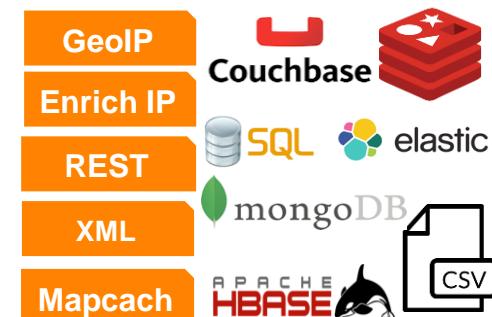
Data Ingest



Data Transformation

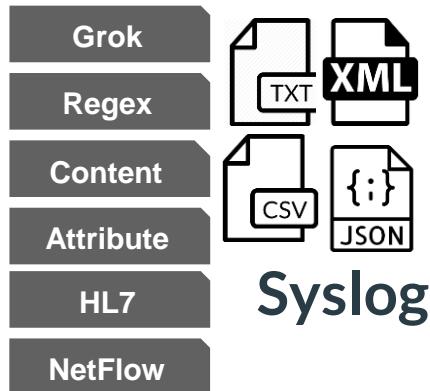


Data Enrichment

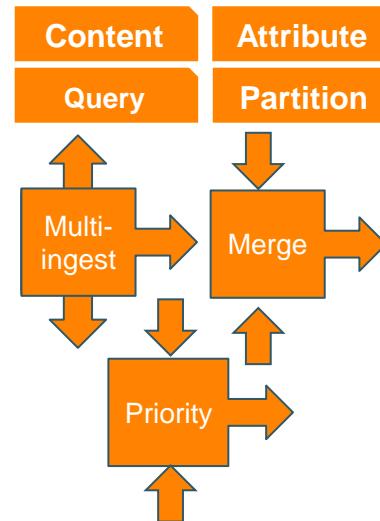


An Overview of Apache NiFi Capabilities

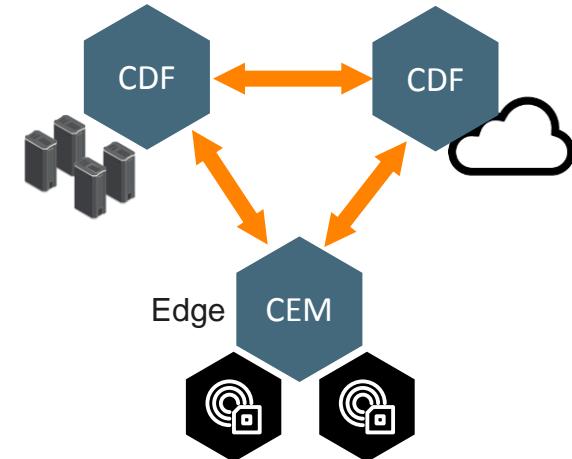
Parsing



Routing



Data Movement



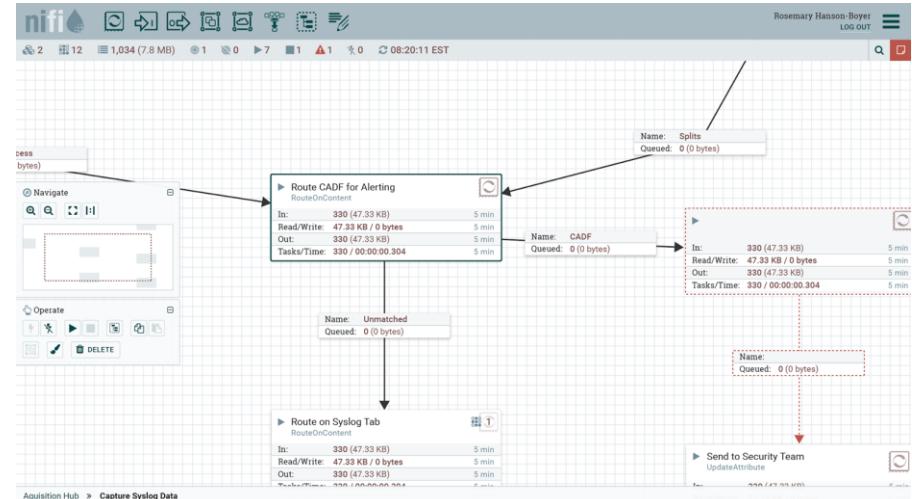
APACHE NIFI 1.9.2

Key New Features

Feature	Description	Apache JIRA
Nar Autoloading	Ability for Apache NiFi to load new nar bundles without restarting.	NIFI-5673
Remote Input/Output Ports	Remote input Input/Output Ports no longer require being placed in the root Process Group	NIFI-2933
PutKudu	Ability for Apache NiFi to write data into Apache Kudu on Cloudera Managed environments.	NIFI-5989
Kerberized Database Connection Pools	Ability for Apache NiFi to utilize JDBC connection pools that are secured via Kerberos (IE: Kudu & Impala)	NIFI-5985
Full Support for Cloud Processors	List too large for here. See release notes.	Multiple

High Level Capabilities

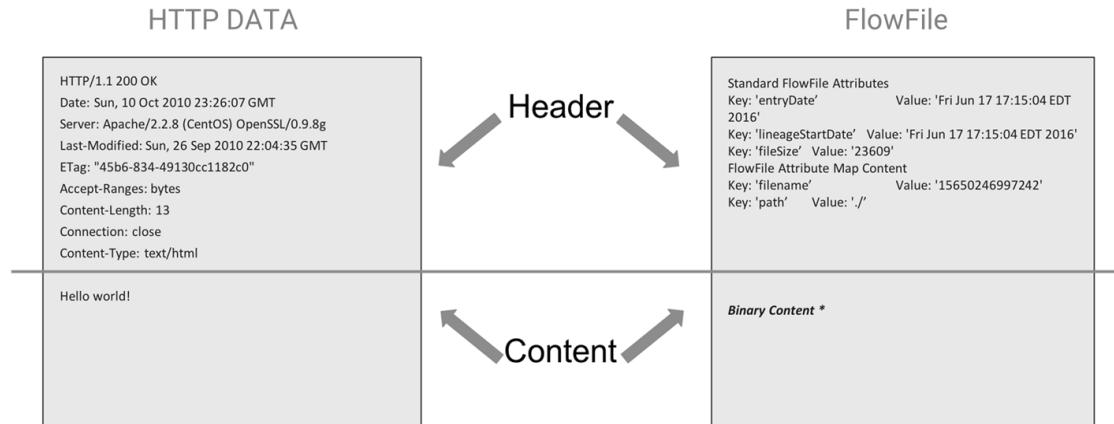
- Scale horizontal and vertically
 - Scale your data flow to millions event/s
 - Ingest TB to PB of data per day
- Adapt to your flow requirements
 - Back pressure & Dynamic prioritization
 - Loss tolerant vs guaranteed delivery
 - Low latency vs high throughput
- Secure
 - SSL, HTTPS, SFTP, etc.
 - Governance and data provenance
- Extensible
 - Build your own processors and Controller services (providers)
 - Integrate with external systems (Security, Monitoring, Governance, etc)



NIFI CORE CONCEPTS

NIFI—TERMINOLOGY

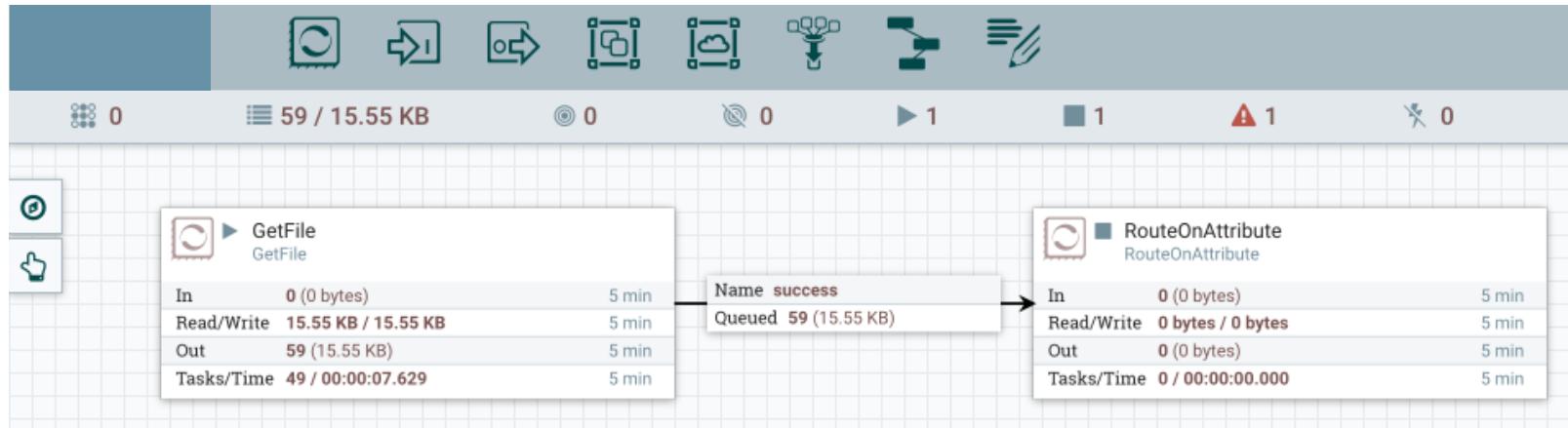
- FlowFile
 - Unit of data moving through the system
 - Content + Attributes (key/value pairs)
- Processor
 - Performs the work, can access FlowFiles
- Connection
 - Links between processors
 - Queues that can be dynamically prioritized
- Process Group
 - Set of processors and their connections
 - Receive data via input ports, send data via output ports



NIFI IS BASED ON FLOW BASED PROGRAMMING (FBP)

FBP Term	NiFi Term	Description
Information Packet	FlowFile	Each object moving through the system.
Black Box	FlowFile Processor	Performs the work, doing some combination of data routing, transformation, or mediation between systems.
Bounded Buffer	Connection	The linkage between processors, acting as queues and allowing various processes to interact at differing rates.
Scheduler	Flow Controller	Maintains the knowledge of how processes are connected, and manages the threads and allocations thereof which all processes use.
Subnet	Process Group	A set of processes and their connections, which can receive and send data via ports. A process group allows creation of entirely new component simply by composition of its components.

VISUAL COMMAND AND CONTROL



- Drag and drop processors to build a flow
- Start, stop, and configure components in real time
- View errors and corresponding error messages
- View statistics and health of data flow
- Create templates of common processor & connections

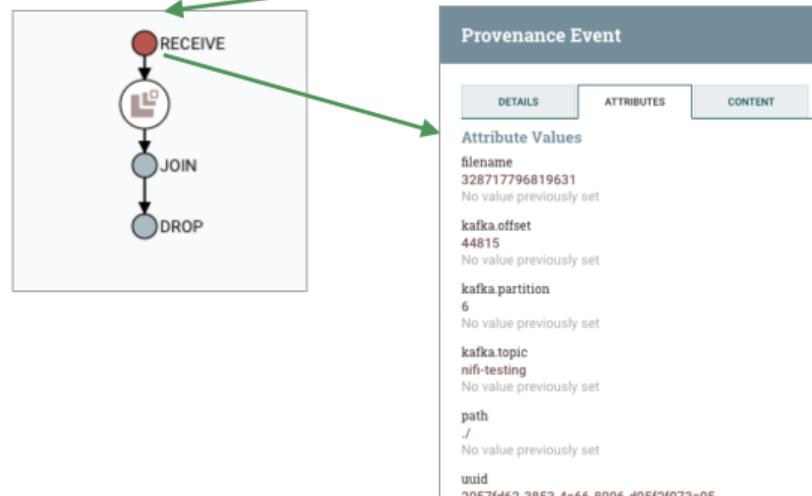
PROVENANCE/LINEAGE

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

Showing the most recent events.

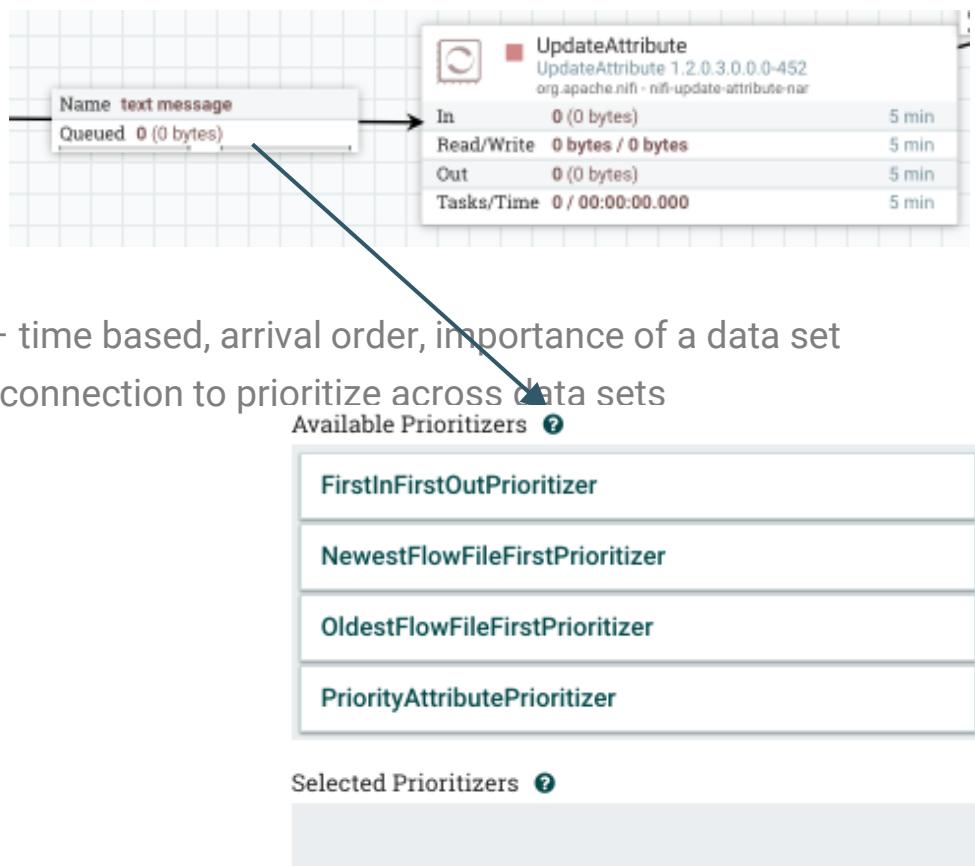
Date/Time	Type	FlowFile Uuid	Size	Component Name	Component Type	Actions
11/15/2016 13:35:03.8...	RECEIVE	379fc4f6-60e0-4151-9743-28...	44 bytes	ConsumeKafka	ConsumeKafka	
11/15/2016 13:35:02.7...	RECEIVE	78f8c38b-89fc-4d00-a8d8-51...	44 bytes	ConsumeKafka	ConsumeKafka	
11/15/2016 13:35:01.6...	RECEIVE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



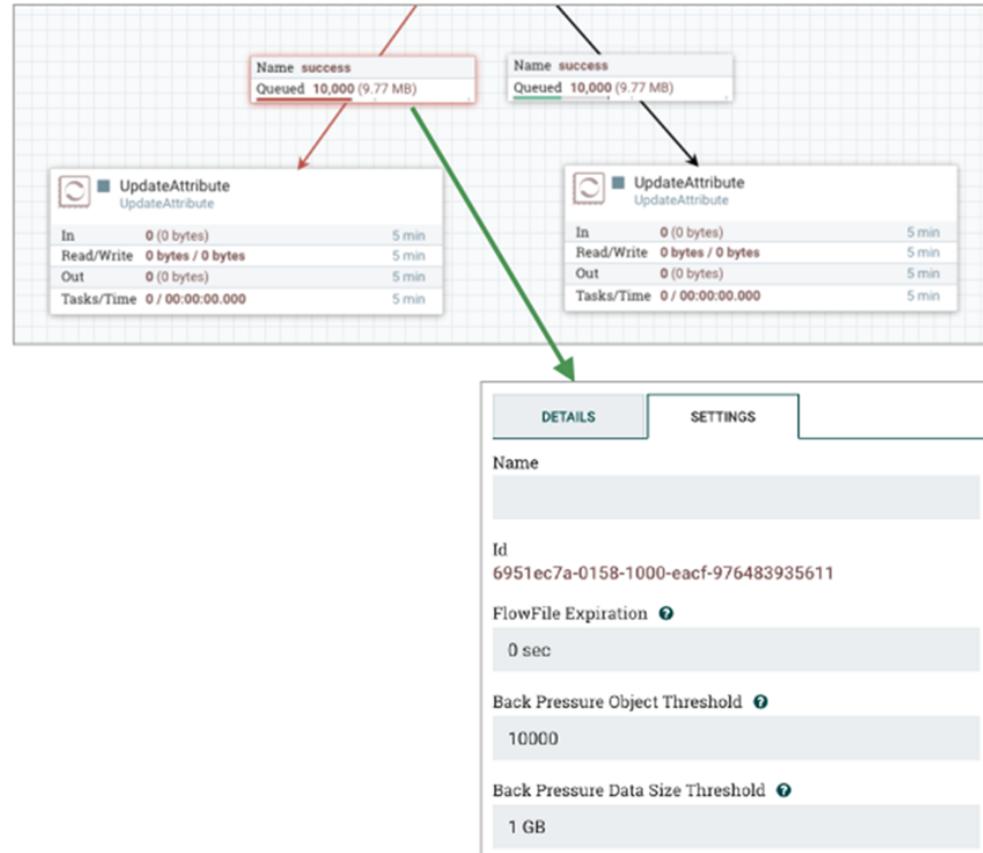
PRIORITIZATION

- Configure a prioritizer per connection
- Determine what is important for your data – time based, arrival order, importance of a data set
- Funnel many connections down to a single connection to prioritize across data sets
- Develop your own prioritizer if needed



BACK PRESSURE

- Configure back-pressure per connection
- Based on number of FlowFiles or total size of FlowFiles
- Upstream processor no longer scheduled to run until below threshold



LATENCY VS. THROUGHPUT

Choose between lower latency, or higher throughput on each processor

Configure Processor

SETTINGS SCHEDULING PROPERTIES COMMENTS

Scheduling Strategy ?
Timer driven

Concurrent Tasks ?
1

Execution ?
All nodes

Run Duration ?
0ms 25ms 50ms 100ms 250ms 500ms 1s 2s
Lower latency Higher throughput

Run Schedule ?
0 sec

The screenshot shows a configuration interface for a processor. At the top, there are tabs for SETTINGS, SCHEDULING, PROPERTIES, and COMMENTS. The SCHEDULING tab is active. Below it, there are sections for Scheduling Strategy (set to Timer driven), Concurrent Tasks (set to 1), and Execution (set to All nodes). On the right, there is a Run Duration slider with options from 0ms to 2s. The slider is currently at 0ms, which is labeled as 'Lower latency'. The 2s option is labeled as 'Higher throughput'.

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

The screenshot shows the NiFi interface with several components and their configurations:

- GenerateFlowFile**:
 - Available Prioritizers: FirstInFirstOutPrioritizer, NewestFlowFileFirstPrioritizer, OldestFlowFileFirstPrioritizer, PriorityAttributePrioritizer.
 - Selected Prioritizers: None.
 - Load Balance Strategy: Do not load balance.
 - Metric Table:

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 42 KB	5 min
Out	42 (42 KB)	5 min
Tasks/Time	42 / 00:00:00.117	5 min
- LogAttribute**:
 - Metric Table:

In	41 (41 KB)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	41 / 00:00:00.141	5 min
- FlowFile**: A table showing flowfile details:

FlowFile ID	Size	Processor	Processor	Processor	Processor
0733ad94-3c80-44d7-9fc2-480caa...	1,024 bytes	LogAttribute	LogAttribute	LogAttribute	LogAttribute
2bc7b5c1-c164-40fb-9e7e-d457884...	1,024 bytes	LogAttribute	LogAttribute	LogAttribute	LogAttribute
80dece7a-15c8-4eb7-80ad-176bfef9...	1,024 bytes	LogAttribute	LogAttribute	LogAttribute	LogAttribute
98d9f9c4-bb47-4fe7-9786-964d027...	1,024 bytes	LogAttribute	LogAttribute	LogAttribute	LogAttribute
26c165ca-2f6d-4714-8c0c-e1de6e2...	1,024 bytes	LogAttribute	LogAttribute	LogAttribute	LogAttribute
8bfff920b-97a3-4b64-998d-046324a...	1,024 bytes	LogAttribute	LogAttribute	LogAttribute	LogAttribute
6345a326-4843-442e-b77d-480d20...	1,024 bytes	LogAttribute	LogAttribute	LogAttribute	LogAttribute
5fc30a5a-641e-4aa0-9c67-3b1d438...	1,024 bytes	LogAttribute	LogAttribute	LogAttribute	LogAttribute
1e90e7ee-92fe-47b3-9aaa-4094fa8...	1,024 bytes	LogAttribute	LogAttribute	LogAttribute	LogAttribute

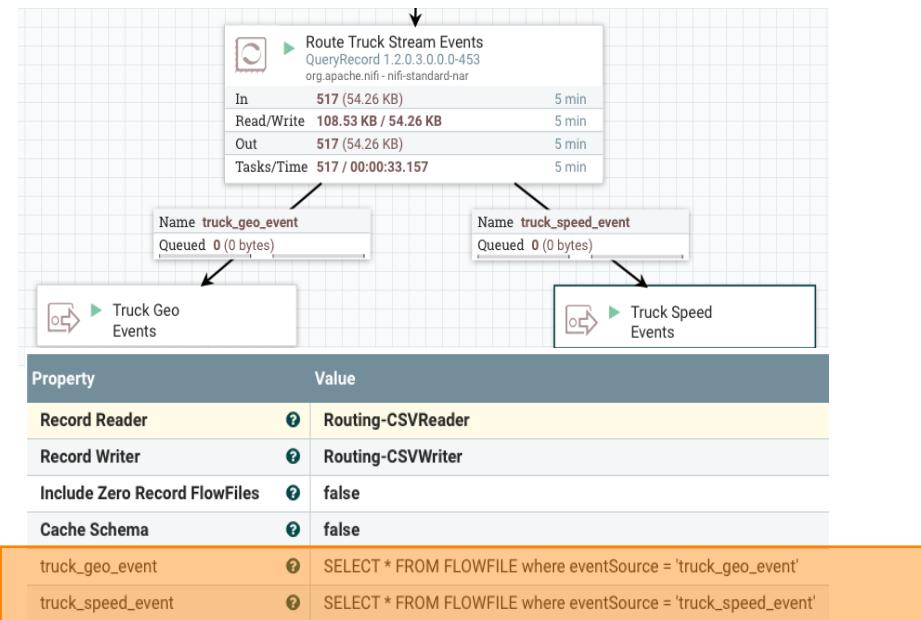
SQL BASED ROUTING WITH NiFi's QueryRecord Processor

QueryRecord Processor- Executes a SQL statement against records and writes the results to the flow file content.

CSVReader: Looking up schema from SR, it will converts CSV Records into ProcessRecords

SQL execution via Apache Calcite: execute configured SQL against the ProcessRecords for routing

CSVRecordSetWriter: Converts the result of the query from Process records into CSV for the for the flow file content



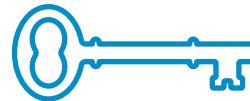
Do routing(routing geo and speed streams) using standard SQL as opposed to complex regular expressions.

Data Protection

Decrypting AES data inline

- **Example**
 - Encrypt data
 - Can use asymmetric encryption for destination-only decrypt
 - Can use symmetric encryption for speed
 - Handles standard AES and GPG ciphers out of the box

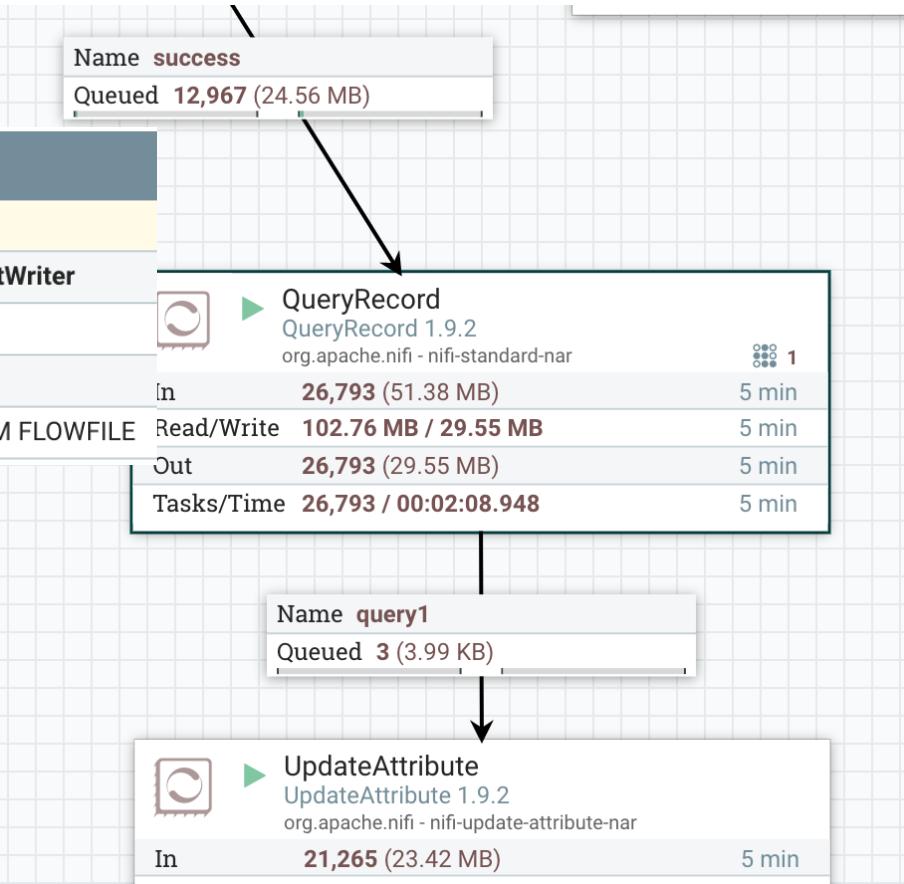
View as:	hex	▼
0x00000000	CB 54 29 E4 BA 64 3B 1B 55 85 84 70 22 9B 77 03	.T)..d;.U..p".w.
0x00000010	4E 69 46 69 49 56 2C 19 3C B1 DA AF 45 65 F9 A9	NiFiIIV,.<...Ee..
0x00000020	E8 D6 F7 B1 53 79 A1 11 81 83 96 FA E6 48 20 B5Sy.....H .
0x00000030	CD AE 71 66 03 C7 20 11 C0 DD 65 9E 4A 54 59 4B	..qf... .e.JTYK
0x00000040	4E 6A CD D7 5D 85 7D 68 C6 AF EF 70 F8 23 E0 D6	Nj...].}h....p.#..
0x00000050	3D 95 8E	=..



View as:		original	▼
1	This is a message at 2017/02/21 22:16:27.986Z		

XML -> JSON or AVRO

Property	Value
Record Reader	XMLReader
Record Writer	JsonRecordSetWriter
Include Zero Record FlowFiles	false
Cache Schema	true
query1	SELECT * FROM FLOWFILE



Advanced XML Processing

Property	Value
Schema Access Strategy	infer schema
Schema Registry	AvroSchemaRegistry
Schema Name	<code>\$(schema.name)</code>
Schema Version	No value set
Schema Branch	No value set
Schema Text	<code>\$(avro.schema)</code>
Schema Inference Cache	No value set
Expect Records as Array	false
Attribute Prefix	No value set
Field Name for Content	No value set
Date Format	No value set
Time Format	No value set
Timestamp Format	No value set

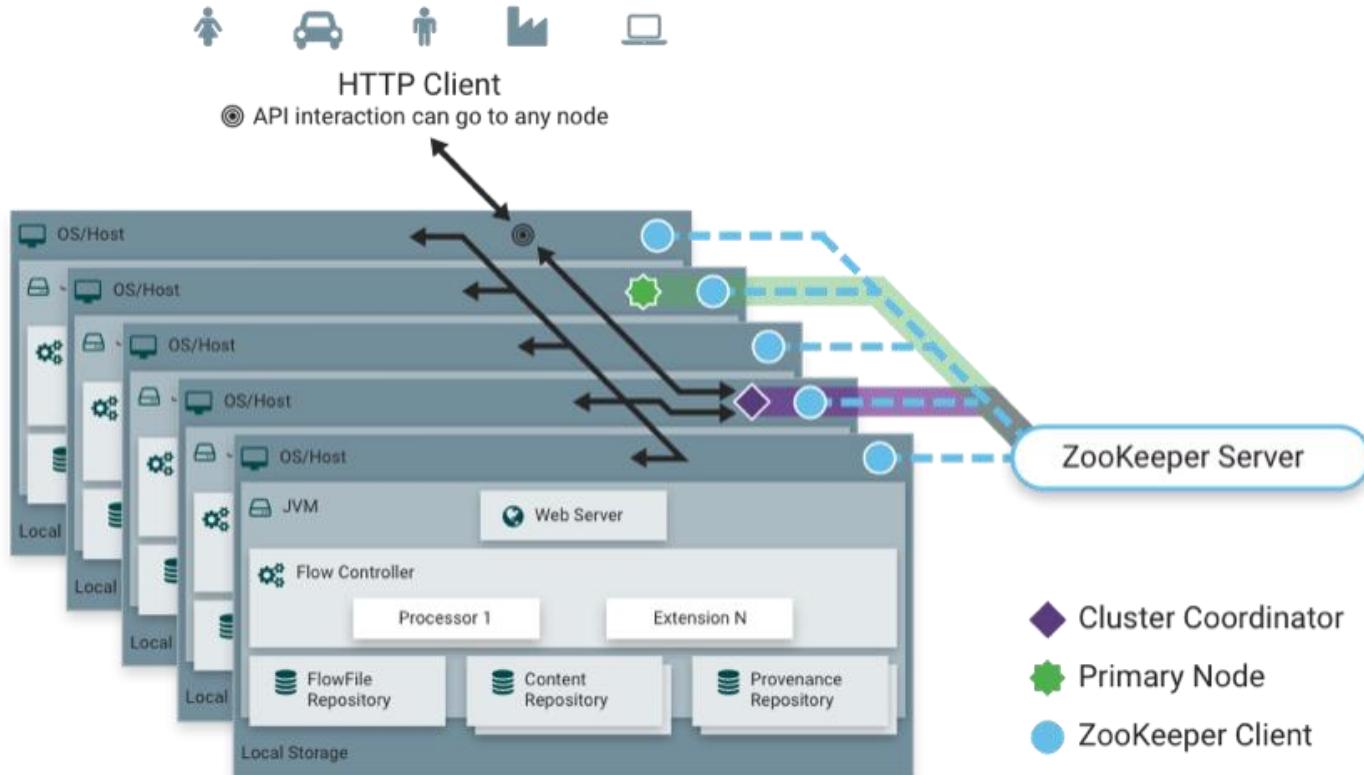
Property	Value
Record Reader	XMLReader
Record Writer	JsonRecordSetWriter
Include Zero Record FlowFiles	false
Cache Schema	true
query1	SELECT * FROM FLOWFILE

<https://pierrevillard.com/2018/06/28/nifi-1-7-xml-reader-writer-and-forkrecord-processor/>

<https://www.datainmotion.dev/2019/03/advanced-xml-processing-with-apache.html>

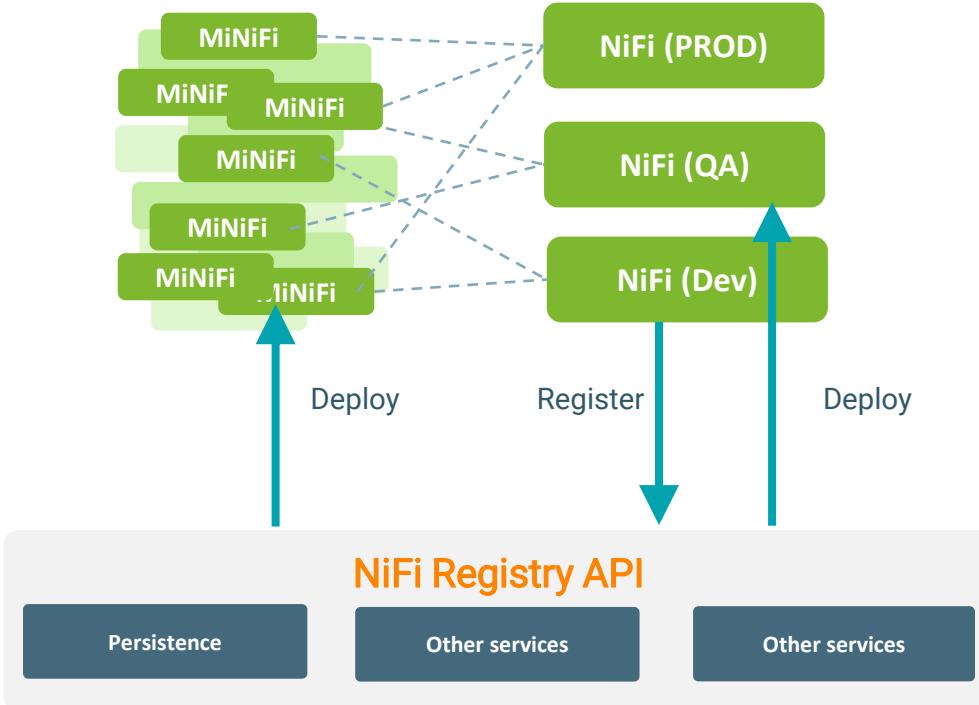
NIFI ARCHITECTURE

ARCHITECTURE



SOFTWARE DEVELOPMENT LIFECYCLE (SDLC)

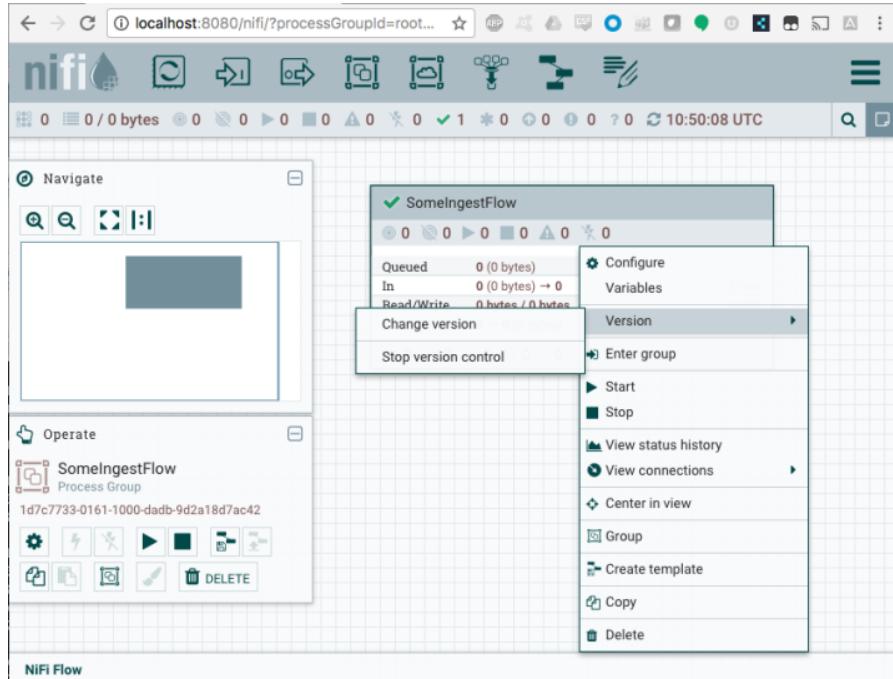
INCREASED DEVELOPER PRODUCTIVITY: APACHE NIFI REGISTRY



NiFi Registry

- Repository of versioned flows
- Portability
- Support multiple registries and interactions between them
- Design and deploy mechanism

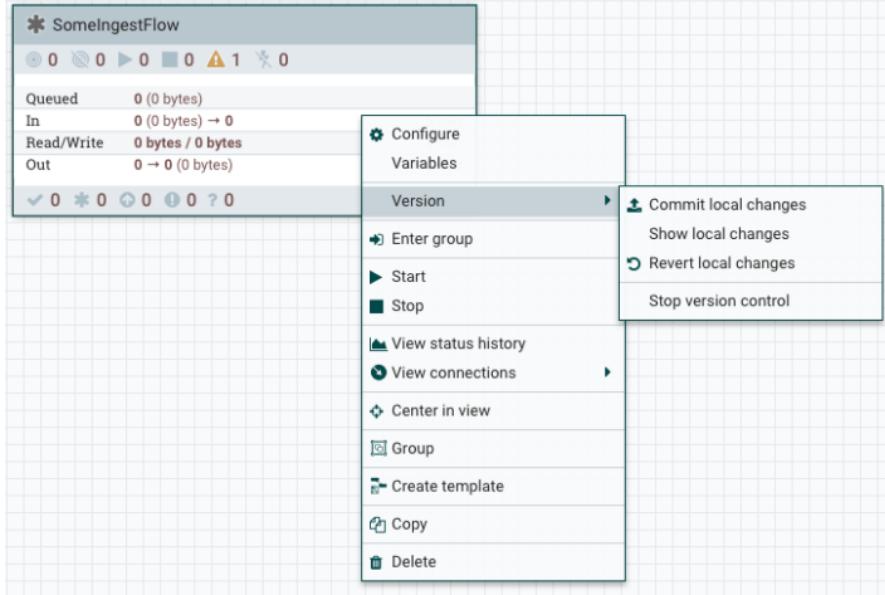
DESIGN & DEPLOY COMPLEMENTING COMMAND & CONTROL



- SDLC Dev: Place Process Groups under Version Control
- Make changes and commit to new version
- Roll versions back or forward

<https://community.cloudera.com/t5/Community-Articles/Building-a-Custom-Apache-NiFi-Operations-Dashboard-Part-1/ta-p/249060>

DESIGN & DEPLOY COMPLEMENTING COMMAND & CONTROL



- Get Notifications of local changes or new versions available in Repository
- Revert or Commit local changes via the GUI or Rest-API
- Use Rest-API to integrate with Jenkins, etc.

<https://medium.com/@abdelkrim.hadjidj/fdlc-towards-flow-development-life-cycle-with-nifi-registries-82e1ee866fab>

INTEGRATED FLOW REGISTRY SERVICE

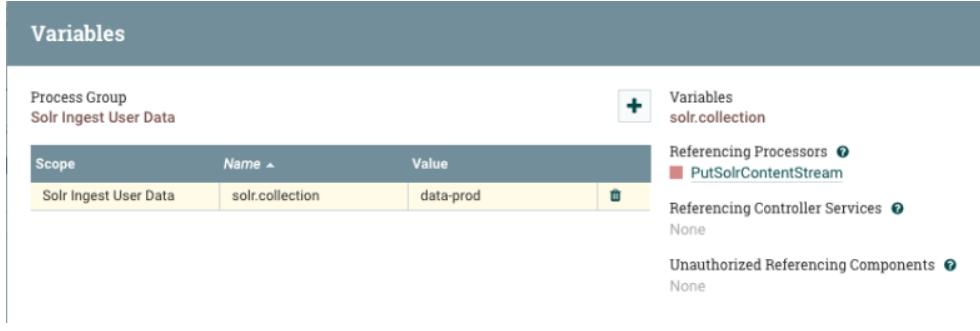
The screenshot shows the NiFi Registry interface. At the top, there's a navigation bar with a logo, the text 'NiFi Registry / All ▾', and user information 'anonymous'. Below the navigation, there's a search bar and a dropdown menu for sorting by 'Name (a - z)'. A sidebar on the left lists 'All buckets' and 'IngestFlow_A1'. The main content area displays two flow buckets:

- AnotherFlow - IngestFlow_A1**: Flow, VERSIONS 1. It has a 'DESCRIPTION' field containing 'also very important'. Under 'CHANGE LOG', it shows 'Version 1 - 2 minutes ago by anonymous' and 'Initial Commit Jan-22-2018 at 11:16 AM'.
- SomeIngestFlow - IngestFlow_A1**: Flow, VERSIONS 1.

- Integrated Flow Registry Service
- Sharable between NiFi environments for Dev/UAT/Prod promotion
- API or GUI driven
- Can be integrated with Enterprise Version Control e.g. GitLab
- ‘Buckets’ of Flows for security and access control

<https://community.cloudera.com/t5/Community-Articles/More-DevOps-for-HDF-Apache-NiFi-Registry-and-Friends/ta-p/248668>

INTEGRATED VARIABLE REGISTRY SERVICE



The screenshot shows the NiFi Variables interface. At the top, there's a header bar with the title "Variables". Below it, a table lists a single variable named "solr.collection". The table has columns for "Scope", "Name", and "Value". The "Scope" column shows "Solr Ingest User Data". The "Name" column shows "solr.collection". The "Value" column shows "data-prod". To the right of the table, there's a summary section with the following details:

- Variables: solr.collection
- Referencing Processors: PutSolrContentStream
- Referencing Controller Services: None
- Unauthorized Referencing Components: None

- Integrated Variable Registry
- Sets of key:value pairs available on every Process Group
- Referenced with NiFi Expression Language
- Dynamically changeable at runtime
- Use within Versioned Flows to set Environment Variables
- GUI or API driven

Mini NiFi (MiNiFi)

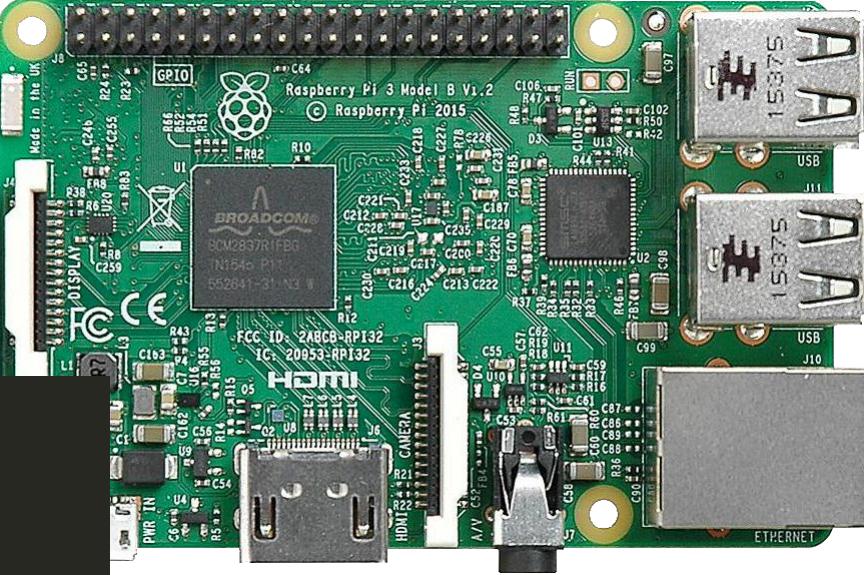
Apache NiFi Solves Everything*

- Runs on JVM
- Provides UI for flow design & monitoring
- Security built-in
 - TLS, authentication/authorization, encrypted data
- Handles practically any format/protocol

Apache NiFi for IoT

- NiFi supports AMQP, MQTT, UDP, TCP, HTTP(S), CEF, JMS, (S)FTP, AWS, Azure, GC, etc
- With a little pruning, NiFi can run on a Raspberry Pi

```
.  
├── bootstrap  
├── jcl-over-slf4j-1.7.12.jar  
├── jul-to-slf4j-1.7.12.jar  
├── log4j-over-slf4j-1.7.12.jar  
├── logback-classic-1.1.3.jar  
├── logback-core-1.1.3.jar  
├── nifi-api-0.6.1.jar  
├── nifi-documentation-0.6.1.jar  
├── nifi-framework-nar-0.6.1.nar  
├── nifi-html-nar-0.6.1.nar  
├── nifi-http-context-map-nar-0.6.1.nar  
├── nifi-jetty-bundle-0.6.1.nar  
├── nifi-kerberos-iaa-providers-nar-0.6.1.nar  
├── nifi-ldap-iaa-providers-nar-0.6.1.nar  
├── nifi-nar-utils-0.6.1.jar  
├── nifi-properties-0.6.1.jar  
├── nifi-provenance-repository-nar-0.6.1.nar  
├── nifi-runtime-0.6.1.jar  
├── nifi-scripting-nar-0.6.1.nar  
├── nifi-ssl-context-service-nar-0.6.1.nar  
├── nifi-standard-nar-0.6.1.nar  
├── nifi-standard-services-api-nar-0.6.1.nar  
└── nifi-update-attribute-nar-0.6.1.nar  
  slf4j-api-1.7.12.jar
```



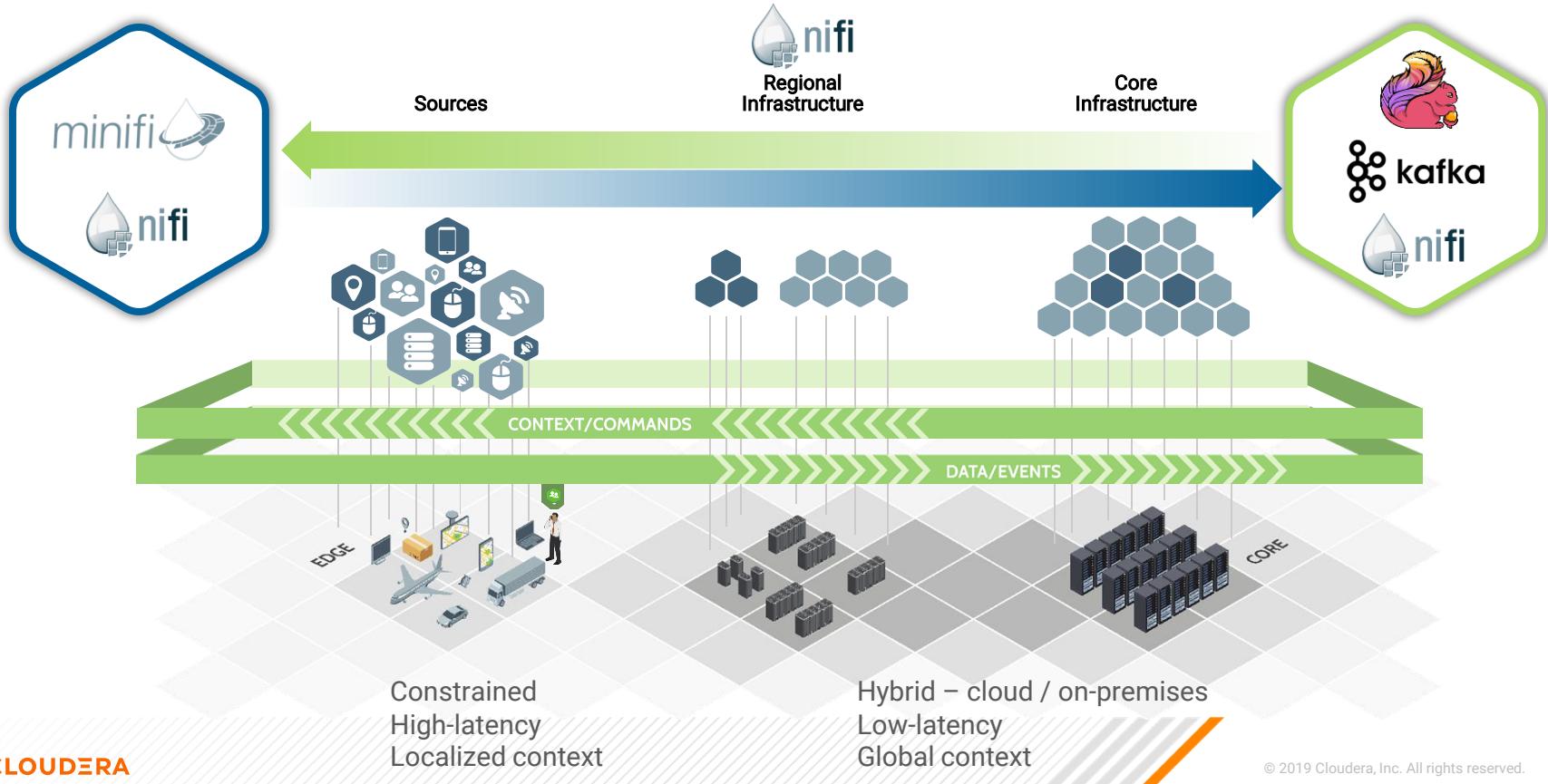
Edge Challenges

- Limited computing capability
- Limited power/network
- Restricted software library/platform availability
- No UI
- Physically inaccessible
- Not frequently updated
- Competing standards/protocols
- Scalability
- Privacy & Security

Apache NiFi is a an eco-system



End to End Data Management



OUTSIDE THE COMFORTS OF THE DATA CENTER

Realities of computing

- Limited computing capability
- Limited power/network
- Restricted software library/platform availability
- No UI
- Physically inaccessible
- Not frequently updated
- Competing standards/protocols
- Scalability
- Privacy & Security

Apache NiFi Subproject: MiNiFi

- Get the key parts of NiFi close to where data begins and provide bidirectional communication
- NiFi lives in the data center – give it an enterprise server or a cluster of them
- MiNiFi lives as close to where data is born and is a guest on that device or system
 - IoT
 - Connected car
 - Legacy hardware



Why build MiNiFi?

- NiFi is big
 - 1.7.0 release is 1.2 GB compressed
 - Can be modified to run in restricted environments, but requires manual surgery
 - Provides UI, provenance query, etc.
 - Runs on dedicated machines/clusters – “owns the box”
- MiNiFi lives at the edge
 - No UI
 - 0.5.0 Java binary is 67 MB, 0.5.0 C++ binary is 4.8 MB (**0.2.0 fits on a floppy disk**)
 - “Good guest”

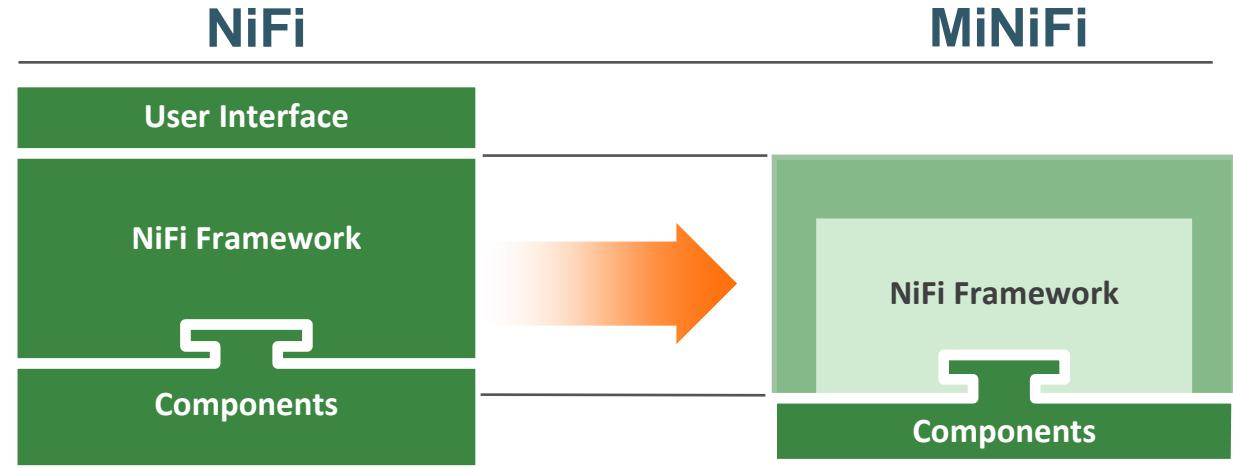
APACHE MINIFI C++ 0.6.0

Key New Features

Feature	Description	Apache JIRA
NiFi JNI Bindings	Ability to run any native NiFi processor via JNI. Makes all NiFi processors available to MiNiFi C++	MINIFCPP-740
Native Python Processors	Ability to write native MiNiFi C++ processors in Python	MINIFCPP-750
PublishKafka	Ability to write to secured Kafka instance	MINIFCPP-731
CoAP Support	Ability to communicate with EFM server over CoAP. Drastically reduces the network impact of heartbeats	MINIFCPP-759
Windows Support	Ability to install MiNiFi via a MSI	MINIFCPP-700

MiNiFi Agents

- C++ and Java agents
- Security & Data provenance
- Guaranteed delivery
- Data buffering, Backpressure
- Prioritized queuing, Flow QoS
- Loss tolerance
- TensorFlow support of Edge AI
- Supports IoT ecosystem



C++ and Java MiNiFi capabilities

Choosing the right agent

MiNiFi C++

- Smaller footprint
- Can work without JVM
- JNI support for all NiFi processors
- Native Python processors
- Lowest network utilization
- More performant
- More processors
 - GPS, MQTT, Camera, etc

MiNiFi Java

- Fully supports all NiFi components
- Requires a JVM
- Benefits from JVM monitoring
- More easily deployed
- Easier to update
- Easier to develop custom features for clients

C++ and Java MiNiFi capabilities

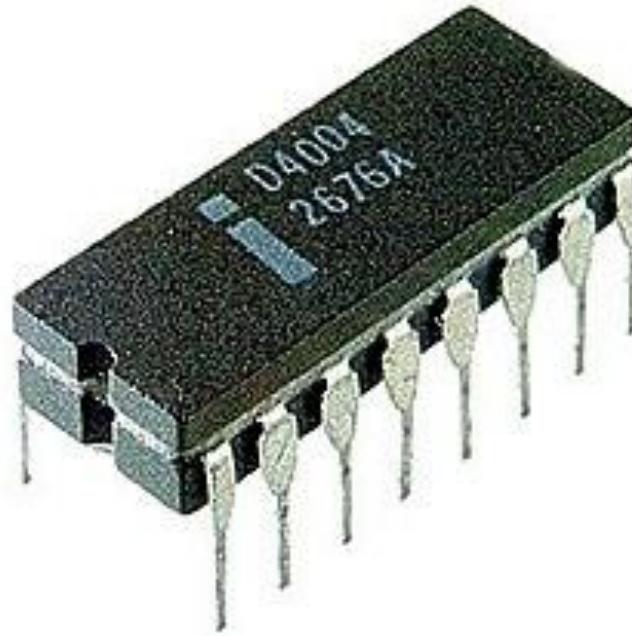
Choosing the right agent continued

- The two agents are 80% the same thing
- Java agent was short term solution while working on C++ agent
- C++ agent is now superior
- Java agent is still very valuable
 - Inherits all NiFi enhancements ... thanks Java!
- Start with C++ agent and move to Java if needed
- Eliminate customer confusion by just saying “agent” and prescribing an agent

MiNiFi

Where not to position

- Never position MiNiFi to run on low level hardware (microprocessors)
 - * Caveat: we are working on a C implementation for this ... not there yet.
- Requires local storage for data resiliency
- Requires Linux or Windows
- Requires networking interfaces



MiNiFi

Where to properly position

- IoT gateways with plethora of interfaces
- Mission critical servers where lightweight footprint for data collection is needed.
 - “Unclaimed environment”
- Ephemeral data collection endpoints
 - Containers, cloud instances, FaaS functions



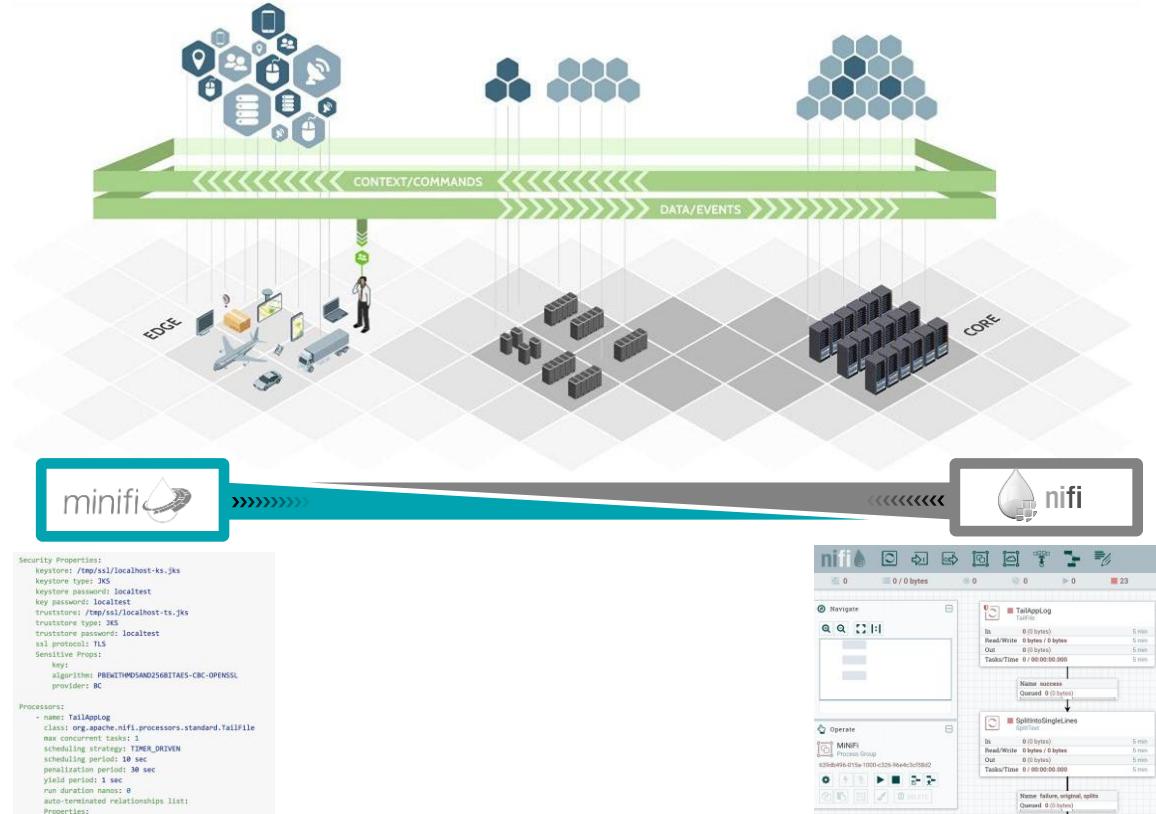
MiNiFi Exfil

Getting Data from the Edge to the Core

- Site-to-Site
 - NiFi protocol
 - Two implementations
 - Raw socket
 - HTTP(S)
 - Secured with mutual authentication TLS
- *HTTP(S), (S)FTP, JMS, Syslog, File, Email, Process also available*

How Does MiNiFi Interact With NiFi?

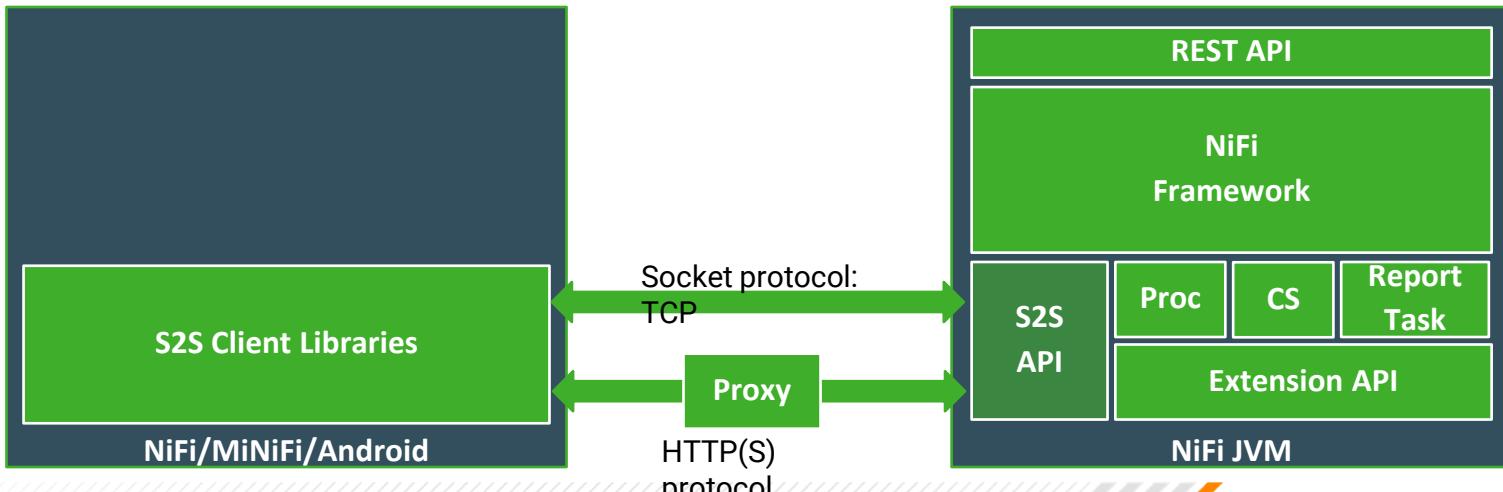
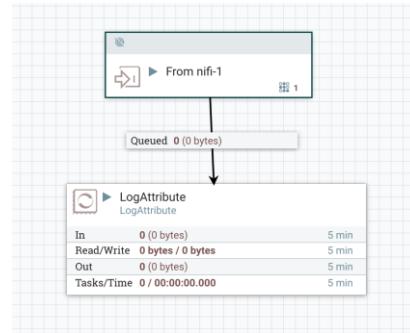
- MiNiFi
 - Receive flows
 - Collect data
 - Send for processing
- NiFi
 - Design flows
 - Aggregate data from many sources
 - Perform routing / analysis / SEP



NiFi Site-to-Site (S2S)

```
Security Properties:  
keystore: /tmp/nifi/localhost-ks.jks  
keyalias: localhost  
keystore password: localtest  
key password: localtest  
truststore: /tmp/nifi/localhost-ts.jks  
truststore type: JKS  
truststore password: localtest  
key algorithm: RSA  
key size: 1024  
Sensitive Prop:  
key  
algorithm: PBKDF2WITHSHA256BITAES+CBC+OPENSSL  
provider: BC  
  
Processors:  
- name: TailAppender  
  class: org.apache.nifi.processors.standard.TailFile  
  max concurrent tasks: 1  
  scheduling strategy: TIMER_DRIVEN  
  scheduling period: 10 sec  
  pool size: 1  
  max pool size: 10 sec  
  yield period: 1 sec  
  run duration: never  
  auto terminate: false  
  auto terminate relationship: list:  
    Properties:  
      - name: tail_appender_001  
        scaling filename pattern: nifi-app*  
        initial start position: beginning_of_file  
      - name: tail_appender_002  
        scaling filename pattern: nifi-app*  
        initial start position: beginning_of_file  
        max concurrent tasks: 2  
        scheduling period: 1 sec  
        pool size: 1  
        yield period: 0 sec  
        penalization period: 30 sec
```

- Guaranteed delivery
- 2-way SSL Encryption
- Built-in proxy capabilities



MINIFI: PRECEDENT FROM NIFI

A quick look at NiFi site to site

- Provides the semantics between two NiFi components across network boundaries
 - A custom protocol for inter-NiFi communication
 - Secure, Extensible, Load Balanced & Scalable Delivery to Cluster
- Extracted out to a client library which powers integration into popular frameworks like Apache Spark, Apache Storm, Apache Flink, and Apache Apex
- Attributes and the FlowFile format maintained

<https://nifi.apache.org/docs/nifi-docs/html/user-guide.html#site-to-site>

MINIFI: PRECEDENT FROM NIFI

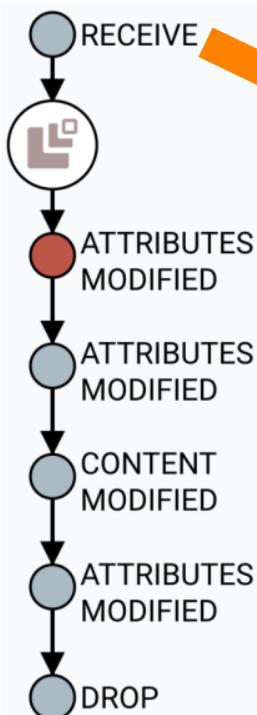
A deeper dive into provenance

- Fine-grained, event level access of interactions with FlowFiles
 - CREATE, RECEIVE, FETCH, SEND, DOWNLOAD, DROP, EXPIRE, FORK, JOIN ...
- Captures the associated attributes/metadata at the time of the event
- A map of a FlowFile's journey and how they relate to other FlowFiles in a system
 - MiNiFi enables us to get more and further illuminate the map of data processing

<http://nifi.apache.org/docs/nifi-docs/html/user-guide.html#data-provenance>

MINIFI: PRECEDENT FROM NIFI

RECEIVE event



Provenance Event

DETAILS	ATTRIBUTES	CONTENT
Time 04/04/2017 16:59:49.642 CEST		Parent FlowFiles (0) No parents
Event Duration < 1ms		Child FlowFiles (0) No children
Lineage Duration < 1ms		
Type RECEIVE		
FlowFileUuid e5a28106-e332-484f-9fd5-df4eb36015f0		
File Size 11 bytes		
Component Id 397820ce-015b-1000-b89d-7e7d65b8b671		
Component Name ListenHTTP		
Component Type ListenHTTP		

OK

APACHE NIFI - MINIFI

Departures from NiFi in getting the right fit

- The feedback loop is longer and not guaranteed
 - Removal of Web Server and UI
- Declarative configuration
 - Lends itself well to CM processes
 - Extensible interface to support varying formats
 - Currently provided in YAML
- Reduced set of bundled components

APACHE MINIFI: SCOPING

Provide all the key principles of NiFi in varying, smaller footprints

- **Go small:** Java – Write once, run anywhere*
 - Feature parity and reuse of core NiFi libraries
- **Go smaller:** C++ – Write once**, run anywhere
 - Go smallest: Write n-many times, run anywhere
Language libraries to support tagging, FlowFile format, Site to Site protocol, and provenance generation without a processing framework
 - Mobile: Android & iOS
 - Language SDKs

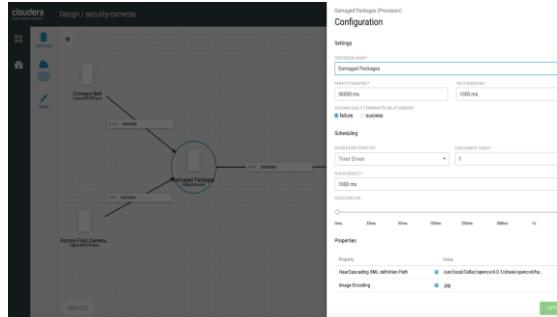


Edge Flow Manager

Cloudera EDGE Management

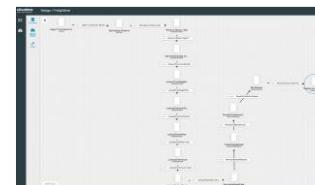
Edge device data collection and processing with easy to use central command and control

Edge Flow Manager

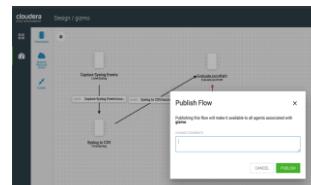


A lightweight edge agent that implements the core features of Apache NiFi, focusing on data collection and processing at the edge

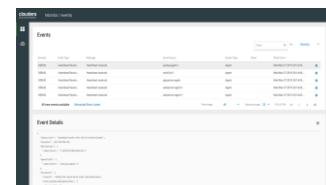
Flow Authorship



Flow Deployment



Flow Monitoring



- Small footprint agent with MiNiFi
- Java and C++ agents
- Rich edge processors (edge collection & processing)
- End to end lineage and security

- Central Command and Control (C2)
- Design and deploy to thousands of agents
- Edge Applications lifecycle management
- Multitenancy with Agent classes
- Native integration with other CDF services

EFM

Flow Authorship - Previously

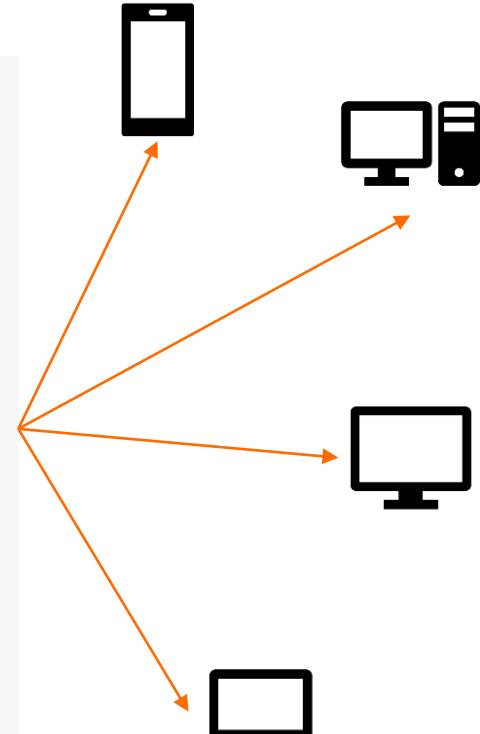
1. SSH to server
2. Enter credentials
3. SCP files to server
4. Stop MiNiFi Agent
5. Copy new config.yml file
6. Prep agent for restart
7. Restart MiNiFi agent
8. Tail logs to validate working
 - a. If not start from scratch!



Author Flow Manually

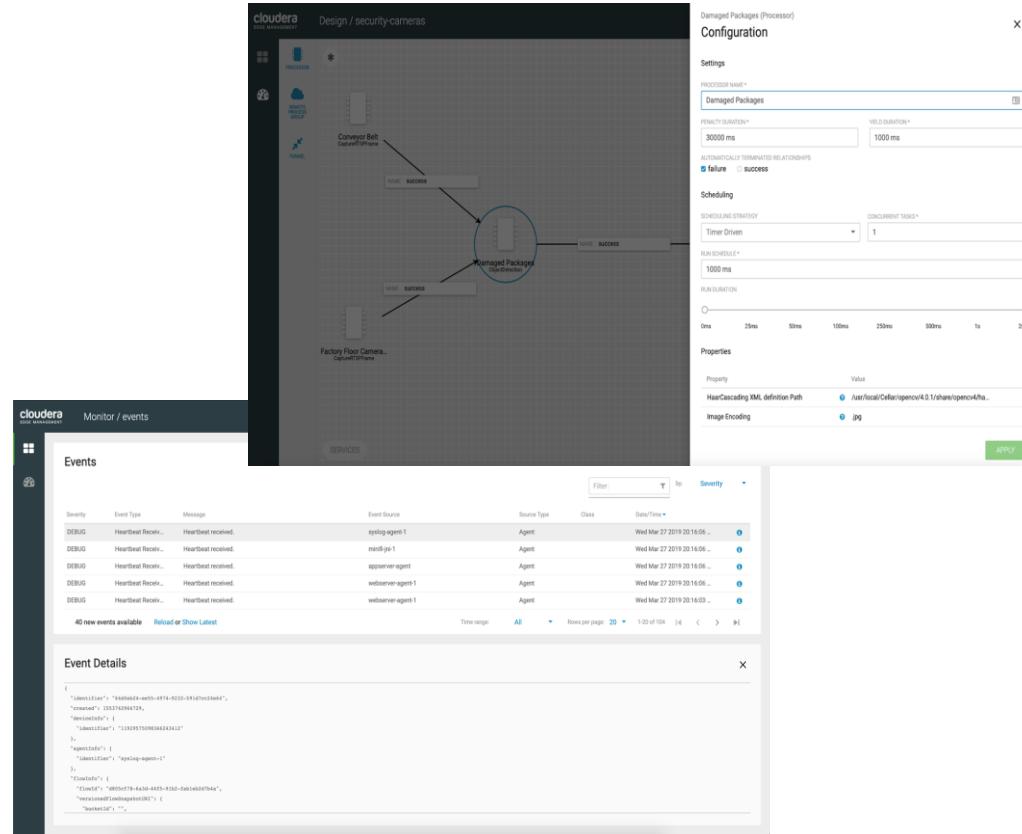
```
Flow Controller:  
id: 471deef6-2a6e-4a7d-912a-81cc17e3a205  
name: MiNiFi Flow  
  
Processors:  
- name: GetFile  
  id: 471deef6-2a6e-4a7d-912a-81cc17e3a206  
  class: org.apache.nifi.processors.standard.GetFile  
  max concurrent tasks: 1  
  scheduling strategy: TIMER_DRIVEN  
  scheduling period: 1 sec  
  penalization period: 30 sec  
  yield period: 1 sec  
  run duration nanos: 0  
  auto-terminated relationships list:  
  Properties:  
    Input Directory: /tmp/getfile  
    Keep Source File: true  
  
Connections:  
- name: TransferFilesToRPG  
  id: 471deef6-2a6e-4a7d-912a-81cc17e3a207  
  source name: GetFile  
  source id: 471deef6-2a6e-4a7d-912a-81cc17e3a206  
  source relationship name: success  
  destination id: 471deef6-2a6e-4a7d-912a-81cc17e3a204  
  max work queue size: 0  
  max work queue data size: 1 MB  
  flowfile expiration: 60 sec  
  
Remote Processing Groups:  
- name: NiFi Flow  
  id: 471deef6-2a6e-4a7d-912a-81cc17e3a208  
  url: http://localhost:8080/nifi  
  timeout: 30 secs  
  yield period: 10 sec  

```



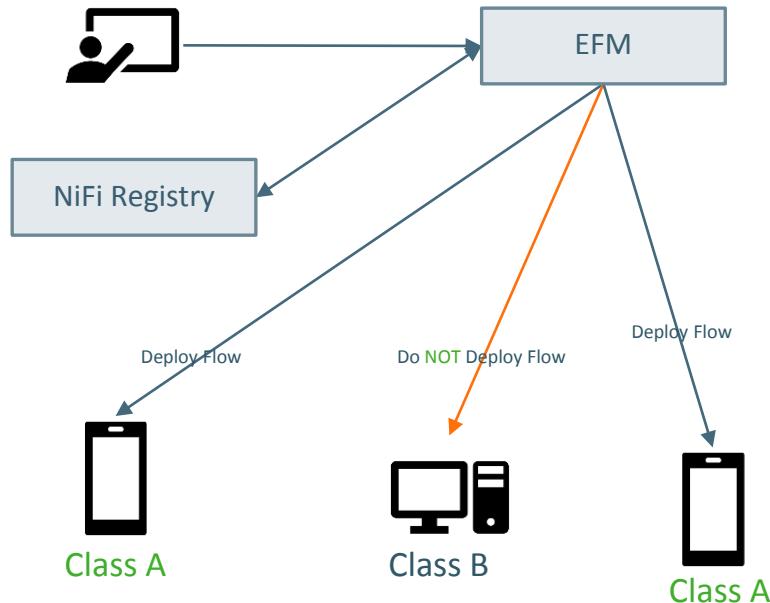
Edge Flow Manager (EFM)

- Manage thousands of deployed agents
 - NiFi-like user interface to develop and deploy flow files to the edge
 - Application lifecycle management
 - Update and deploy ML model files to the edge agents
 - Monitor thousands of edge agents
 - Integration with NiFi Registry



EFM Command & Control

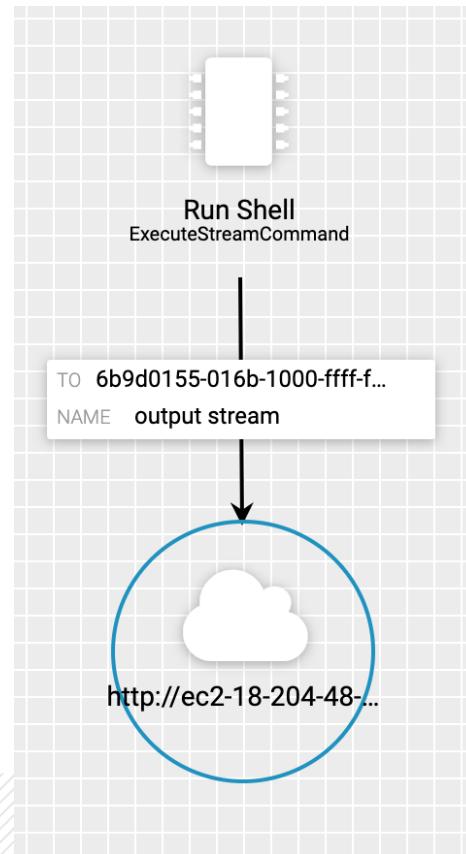
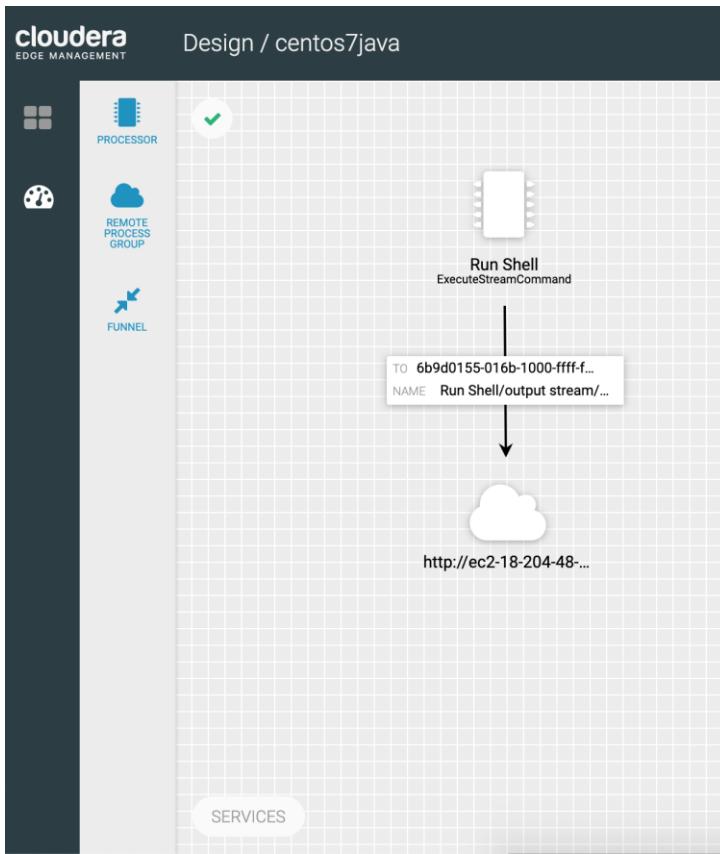
Flow Authorship & deployment



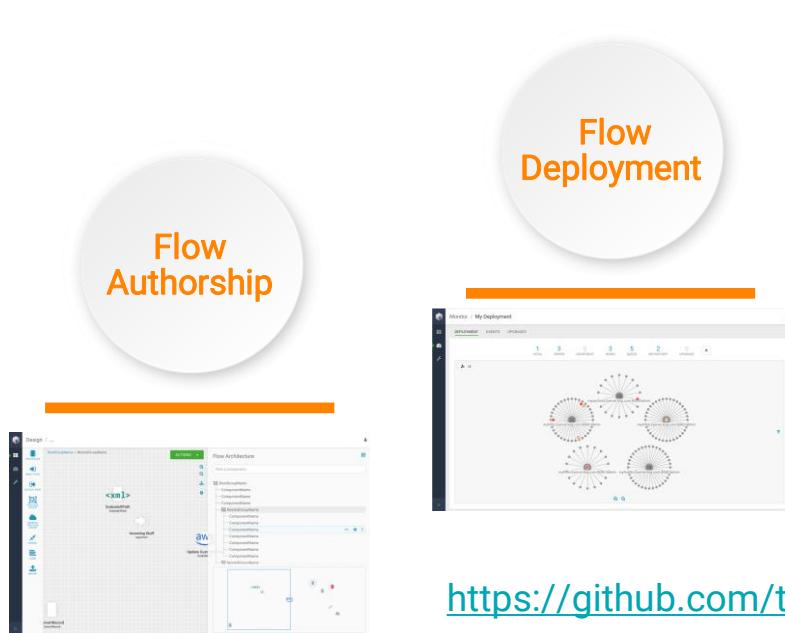
A screenshot of the Cloudera EFM UI. The left pane shows a flow diagram titled "Design / security-cameras" with nodes like "Conveyor Belt CameraFrame", "Damaged Packages ObjectDetection", and "Factory Floor Camera... CameraFrame". Success relationships are shown between these nodes. The right pane is a "Configuration" panel for the "Damaged Packages (Processor)" component. It includes fields for Processor Name (Damaged Packages), Penalty Duration (30000 ms), Yield Duration (1000 ms), Scheduling (Timer Driven, 1000 ms), and Properties (HaarCascading XML definition Path: /usr/local/Cellar/opencv/4.0.1/share/opencv4/haarcascade_smile.xml, Image Encoding: jpg). An "APPLY" button is at the bottom right.

- User designs flow in EFM UI
- Completed flow is saved to NiFi Registry
- **Class A** flow is pushed to **only class A** devices

EFM Demo



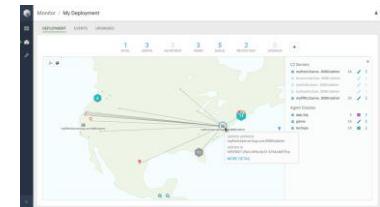
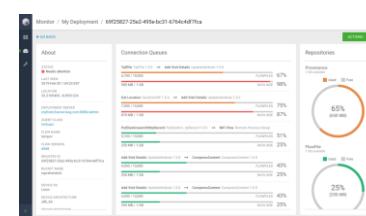
EFM ADDRESSES



Flow
Deployment

Flow
Monitoring

The
Disconnected
Edge



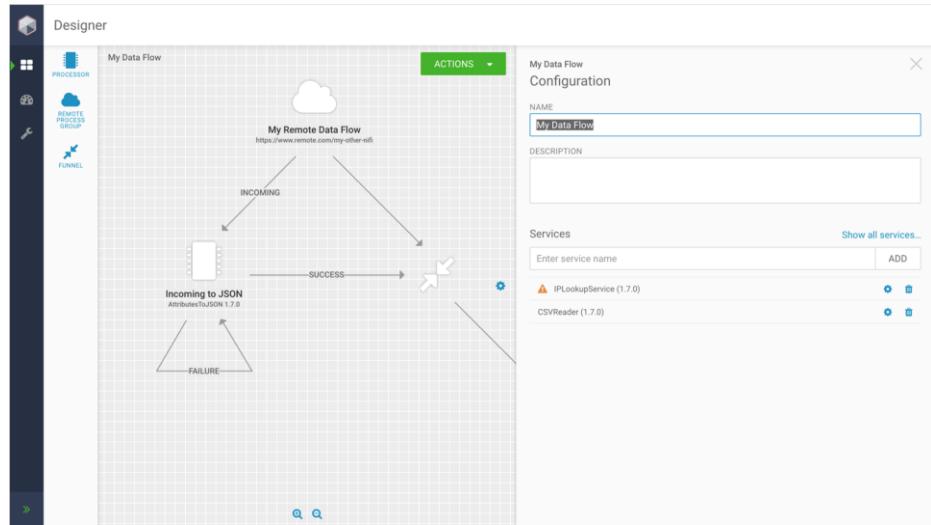
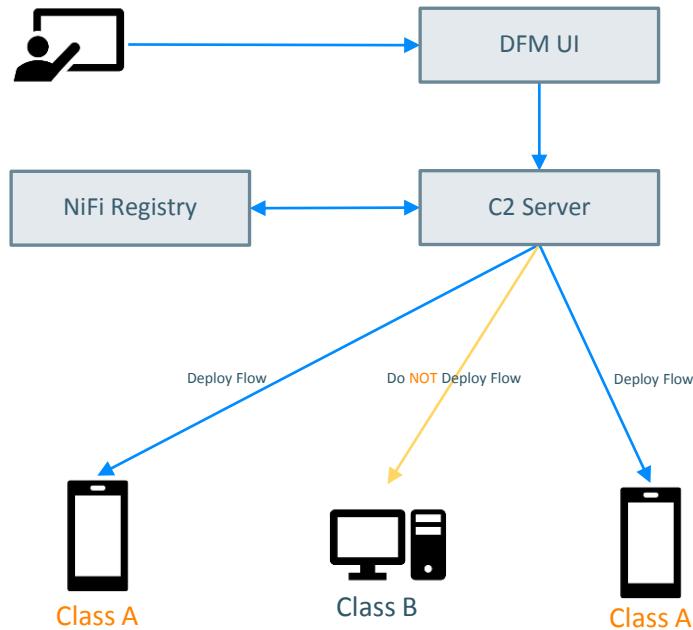
<https://github.com/tspannhw/CDF-Workshop/blob/master/efm.md>

TERMINOLOGY

- **Flow Authorship** – Process of a flow operator designing/editing a flow in the DFM UI and then deploying those flows to targeted edge devices
- **Flow Monitoring** – Operational insights into flows deployed on Edge Devices.
- **Device Manifest** – Payload included in the edge device's heartbeat to their correlating C2 server that fully describes the processors supported by the edge device. This in turn drives the processors that are available to the flow author in the DFM UI
- **Edge Device** – Device running either MiNiFi Java or MiNiFi C++
- **DFM** – Dataflow Manager, DPS app that provides the actual UI components to users
- **Class** – A taxonomical grouping of Edge Devices

FLOW AUTHORSHIP

FLOW AUTHORSHIP



- Flow is sent to C2 server to be delivered to edge
- Flow is saved to NiFi Registry as versioned flow
- Class A flow is pushed to only class A devices

FLOW AUTHORSHIP – NEW DATA FLOW REQUIREMENTS

- A Data Flow developer has been tasked with modifying all **gizmo** Data Flows because all upstream environments now require JSON formatted data
- Developer must be able to load any existing flows into the DFM Designer
- Developer needs the ability to view all possible processors/services available to them for the specific class of devices
 - This is a very powerful and unique feature we offer
- Once Development is complete the developer needs the capability to commit their changes to the NiFi Registry and Git where it is staged for deployment

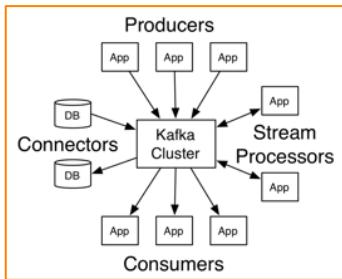
CSP : Cloudera Stream Processing

APACHE KAFKA CORE CONCEPTS

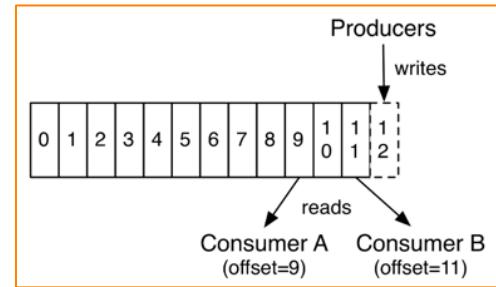
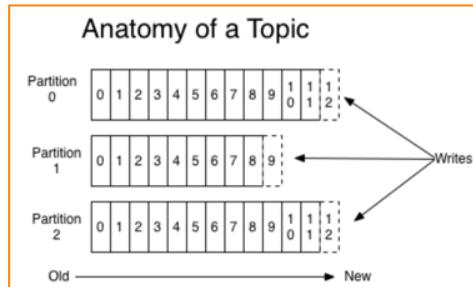
APACHE KAFKA BASICS

Kafka has 4 core APIs

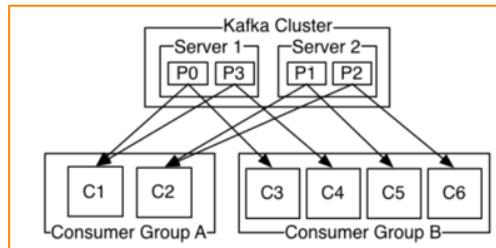
1. [Producer API](#)
2. [Consumer API](#)
3. [Streams API](#)
4. [Connector API](#)



Anatomy of a Kafka Topic



Kafka Consumers



WHAT IS APACHE KAFKA?



- According to the Kafka website:

Kafka is a distributed, partitioned, replicated commit log service. It provides the functionality of a messaging system, but with a unique design.

- **distributed**: horizontally scalable (just like Hadoop!)
- **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

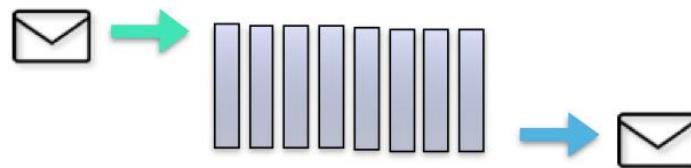
WHAT?



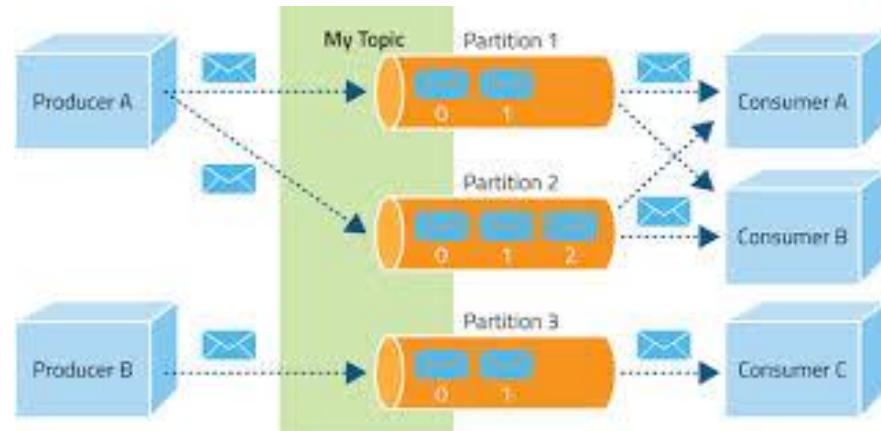
- A messaging system
- It can handle lots of messages
- Messages cannot get lost
- Message order is preserved
- Message will get delivered
- A producer sends the message and one or more consumers can read the message
- Kafka was built at LinkedIn in 2011
- Open sourced as an Apache project



Inside Kafka



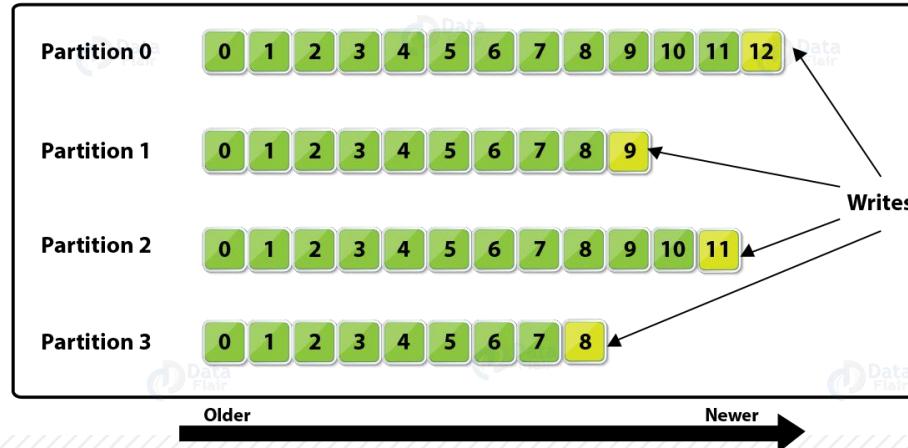
- Kafka topic and topic partition
- Kafka Broker
- Producers
- Consumers



How does Kafka preserve message order

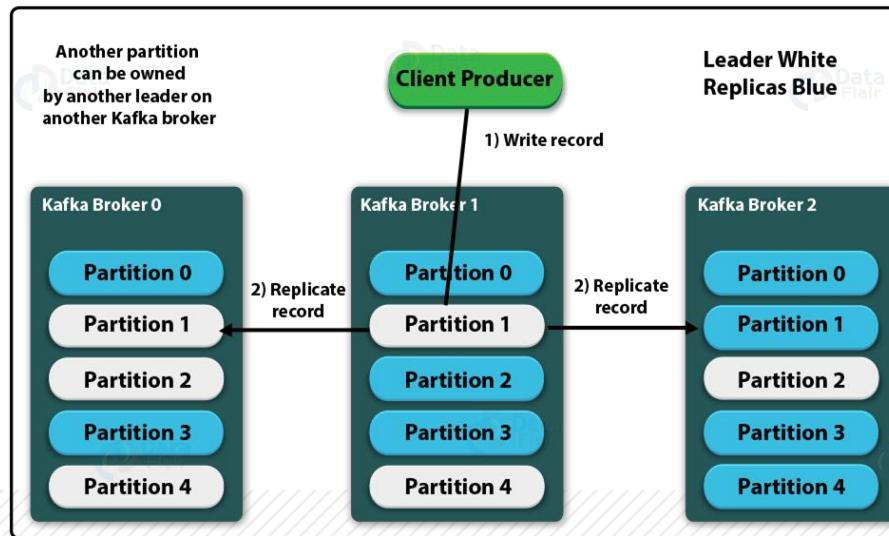
- Partition algorithm is fixed (hash on key)
- Stored as a log sequential write to a file
- Consume in order based on offset

Kafka Topic Partitions Layout



How does Kafka prevent data loss

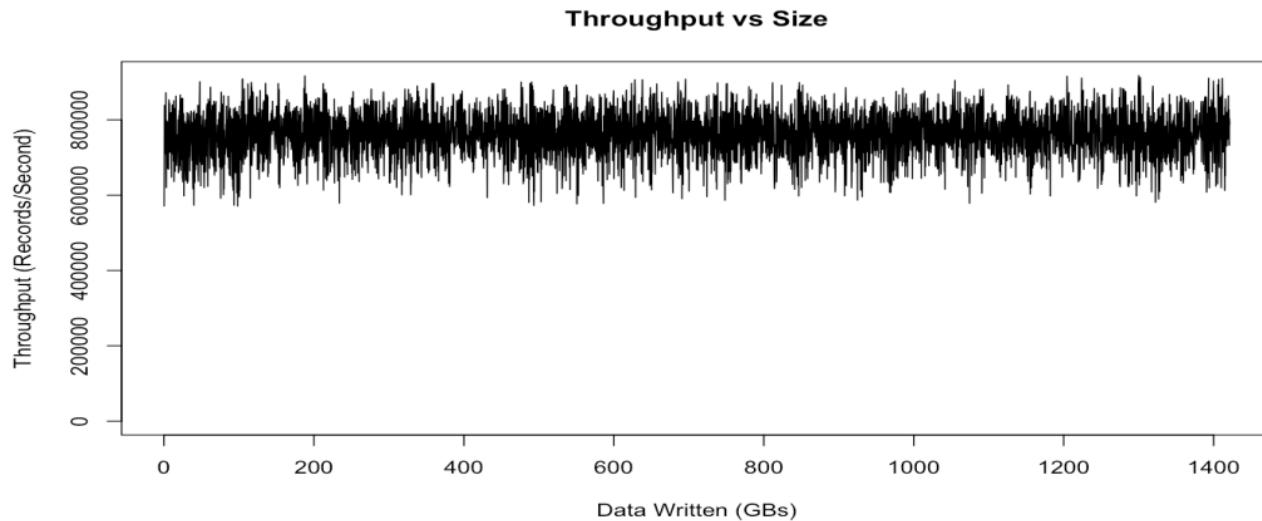
- Replicate, replicate, replicate
- Acknowledge you got the message
- Keep it even after it is consumed



HOW FAST IS KAFKA?

“Up to 2 million writes/sec on 3 cheap machines”

Using 3 producers on 3 different machines, 3x async replication



<http://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines>

WHY IS KAFKA SO FAST?

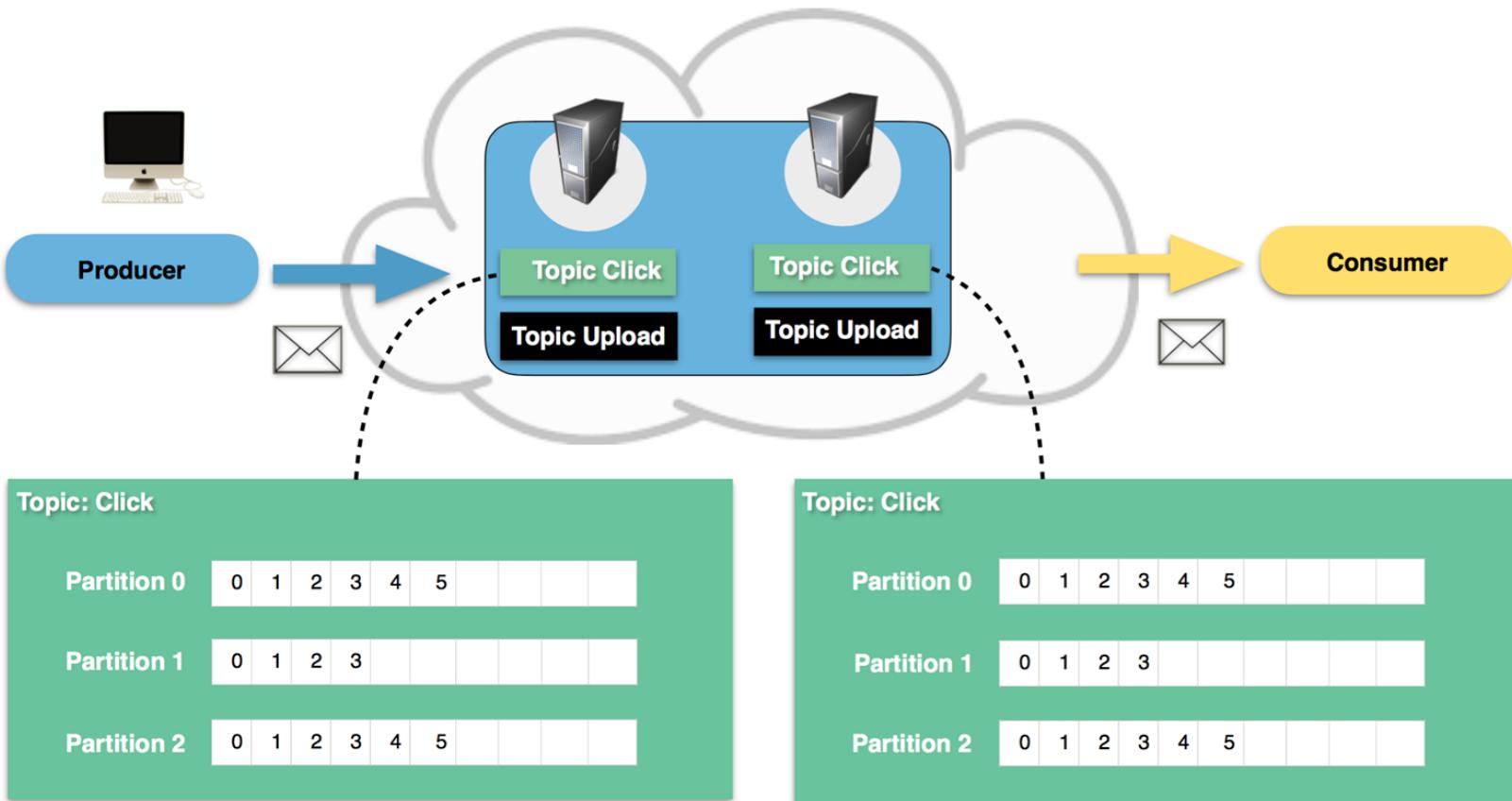
- **Fast writes:**
 - While Kafka persists all data to disk, essentially all writes go to the page cache of OS, i.e. RAM.
- **Fast reads:**
 - Very efficient to transfer data from page cache to a network socket
 - Linux: `sendfile()` system call
- **Fast writes + fast reads = fast Kafka!**
 - On a Kafka cluster where the consumers are mostly caught up, you will see no read activity on the disks as they will be serving data entirely from cache.

<http://kafka.apache.org/documentation.html#persistence>

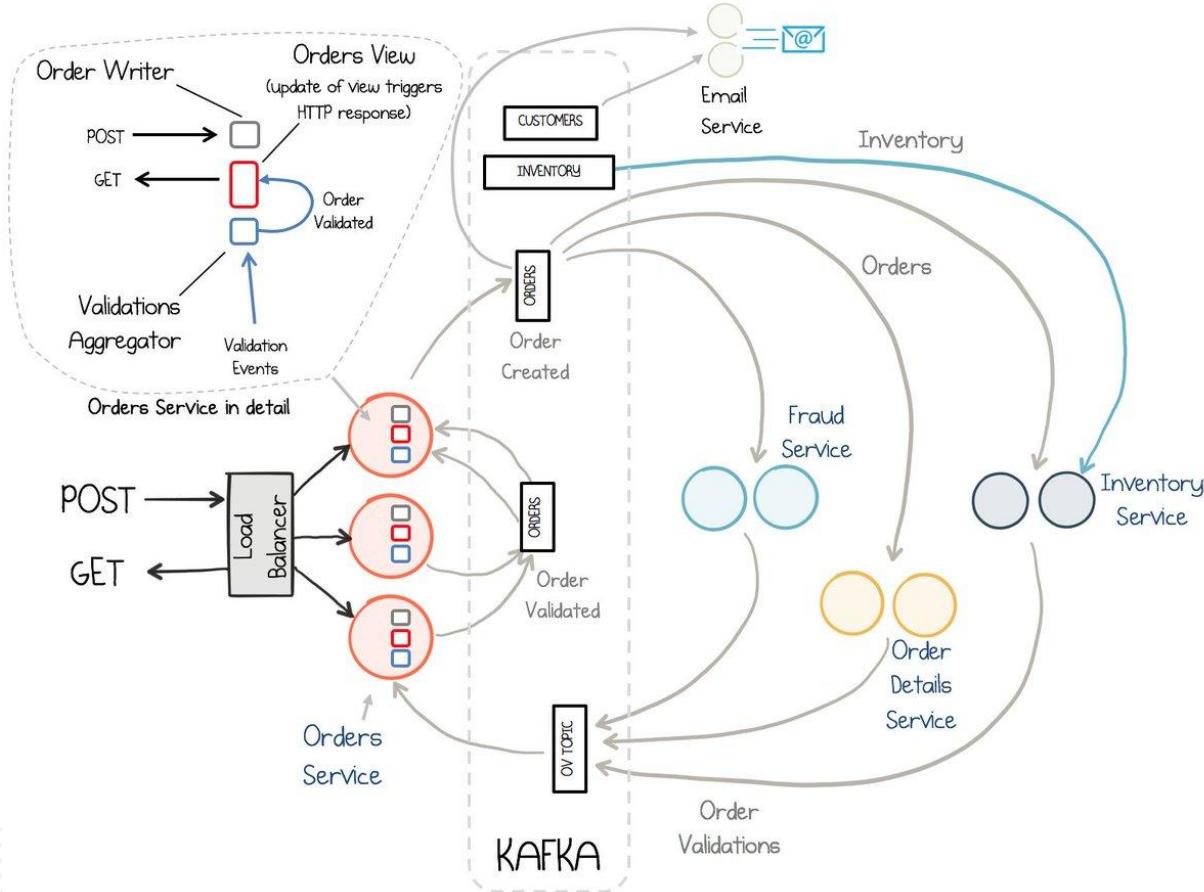
KAFKA USE CASES

- 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 Use Case: Clickstream Data



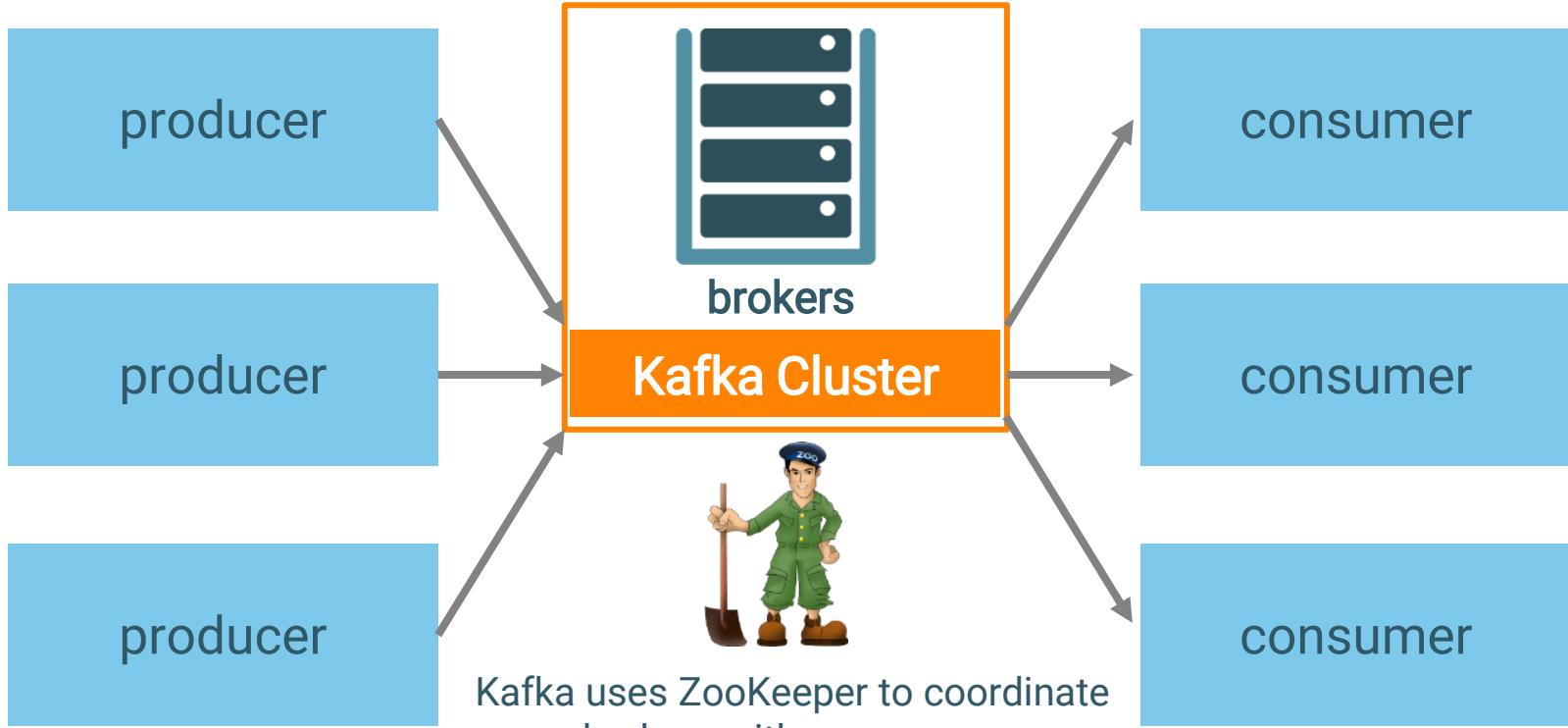
Kafka Use Case: Online ordering system



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

KAFKA COMPONENTS

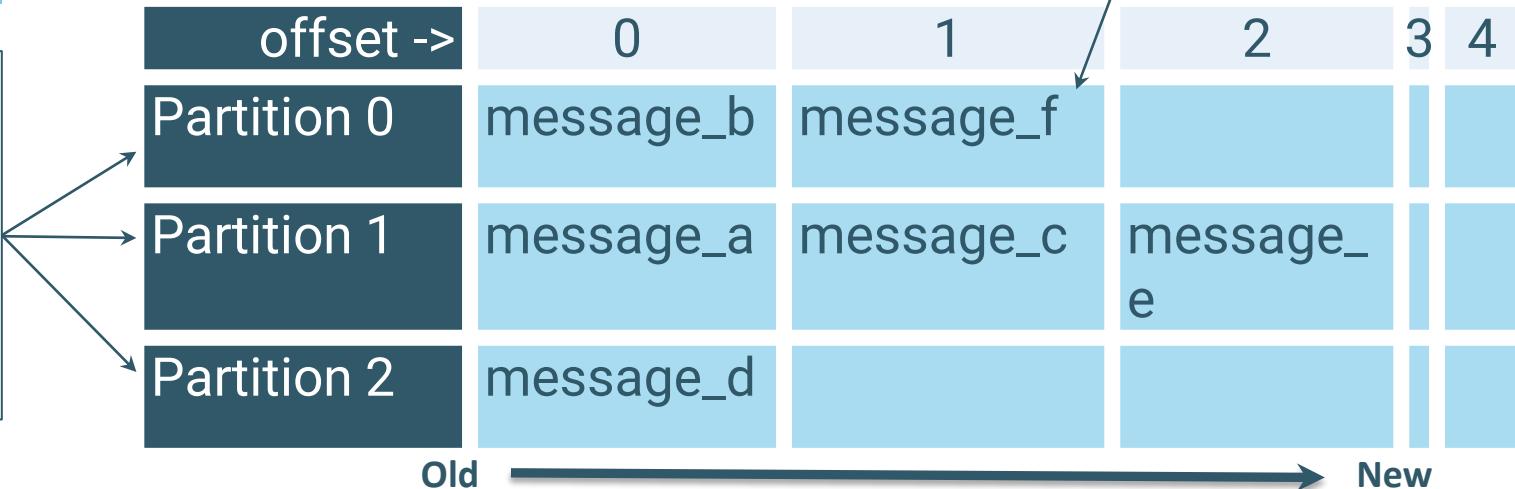
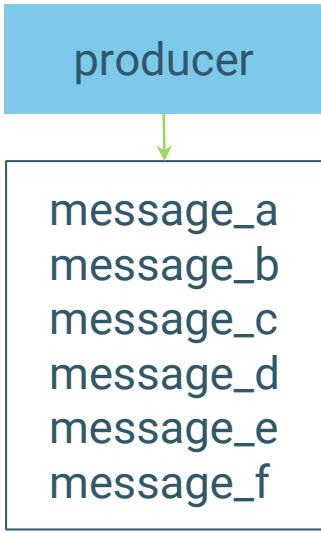


OVERVIEW OF TOPICS

- A **topic** is a name assigned to a feed to which messages are published
 - A topic in Kafka is partitioned
- Each **partition** is an ordered, immutable sequence of messages
 - it is continually appended to
 - each message is assigned a sequential id called an **offset**
- Messages are retained for a configurable amount of time (24 hours, 7 days, etc.)
- Each consumer retains its own offset in the partition
 - allows the consumer to go back and re-read messages without retaining the message
 - the offset is the only metadata that the consumer retains
 - different consumers maintain their own offset

PUBLISHING MESSAGES

1. A producer publishes messages to a topic



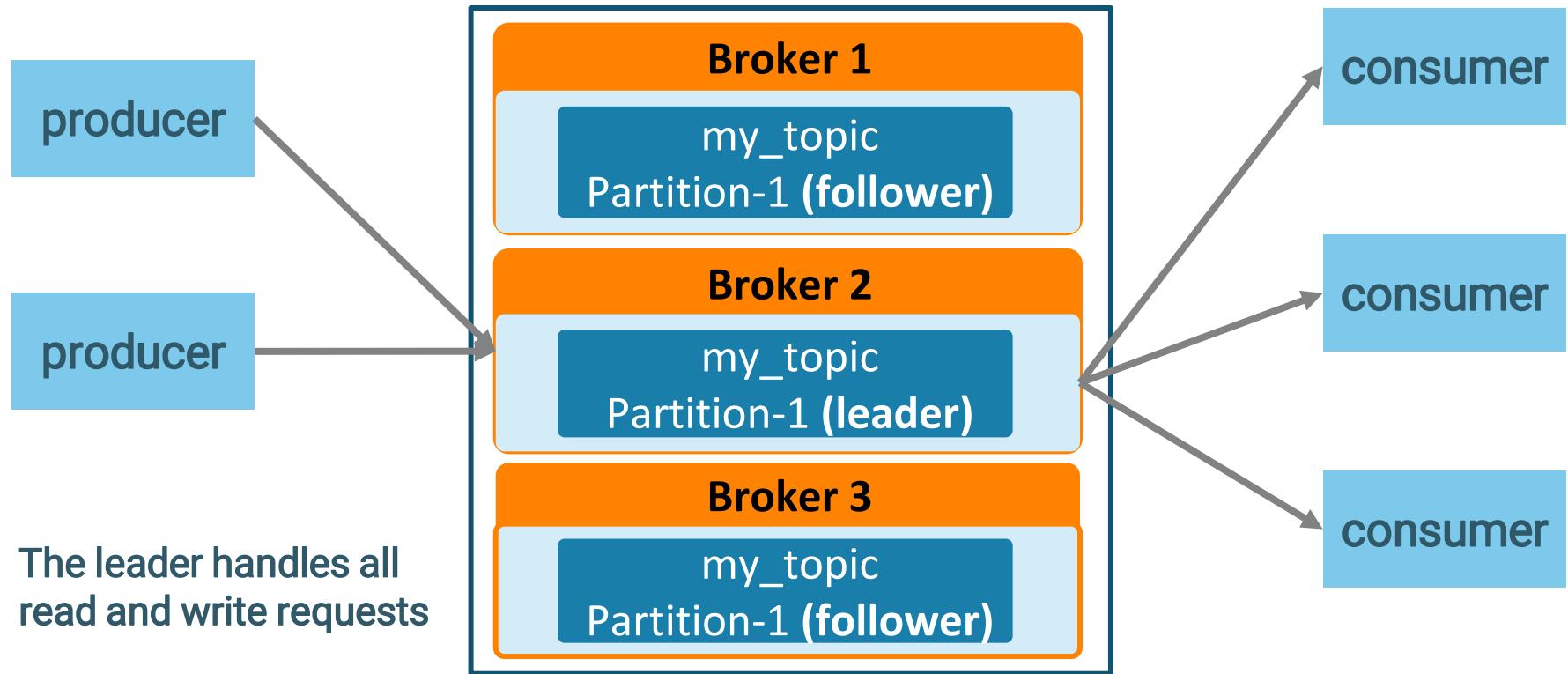
2. The producer decides which partition to send each message to

3. New messages are written to the end of the partition

UNDERSTANDING PARTITIONS

- Partitions are distributed across the cluster
- A partition is managed by a broker
- Each partition is **replicated** for fault tolerance
- A replicated partition has one broker that acts as the **leader**
- The other brokers of that partition act as **followers**
 - The followers passively replicate the leader
 - If the leader fails, one of the followers automatically becomes the new leader
 - Brokers distribute their roles as leaders and followers to maintain a well-balanced cluster

LEADER AND FOLLOWERS



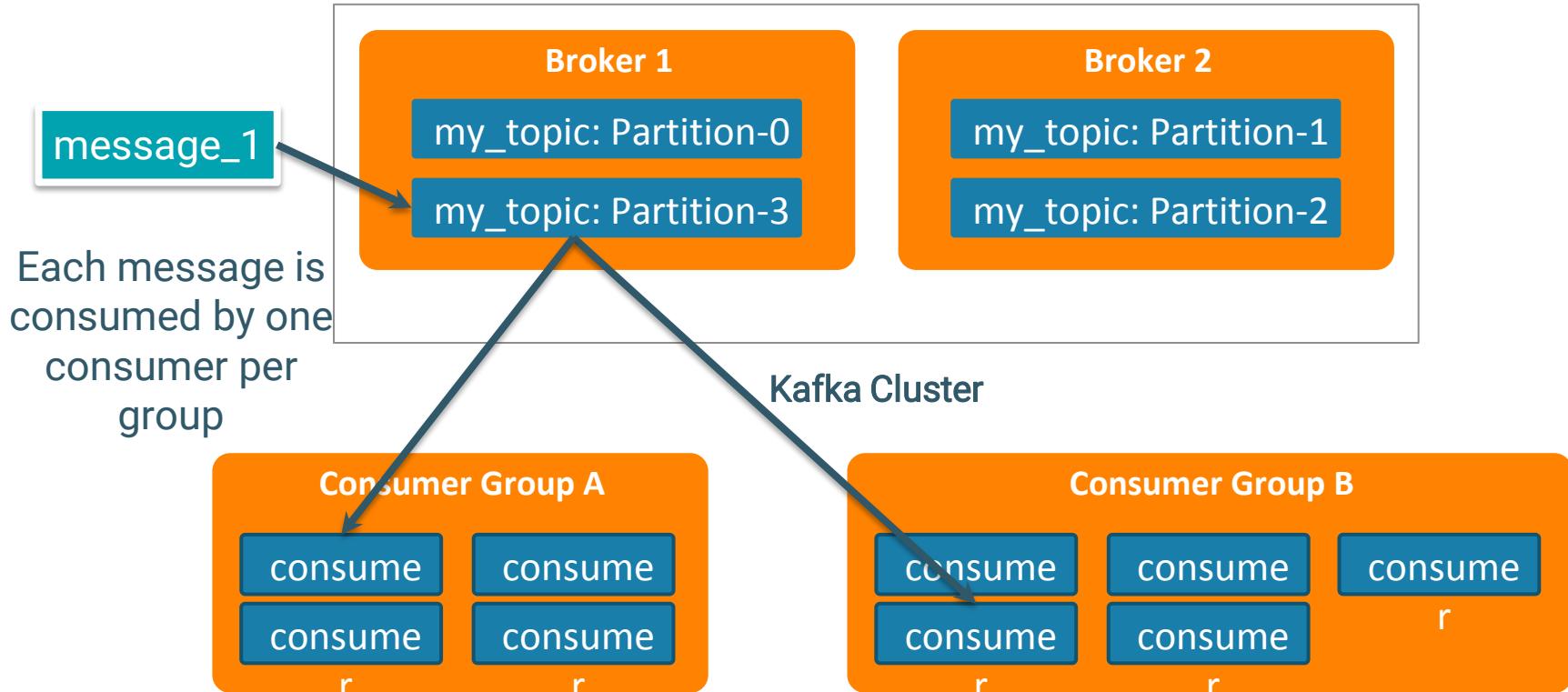
CONTROLLING PARTITIONING LOGIC

- The partitioning logic is performed by the producer
- This can happen various ways:
 - hash function (default behavior – the keys are hashed and divided by the # of partitioners)
 - random distribution (if the keys are null)
 - you can specify a **partitioner** using the `partitioner.class` config property (set to the name of a custom Java class that you write)

CONSUMING MESSAGES

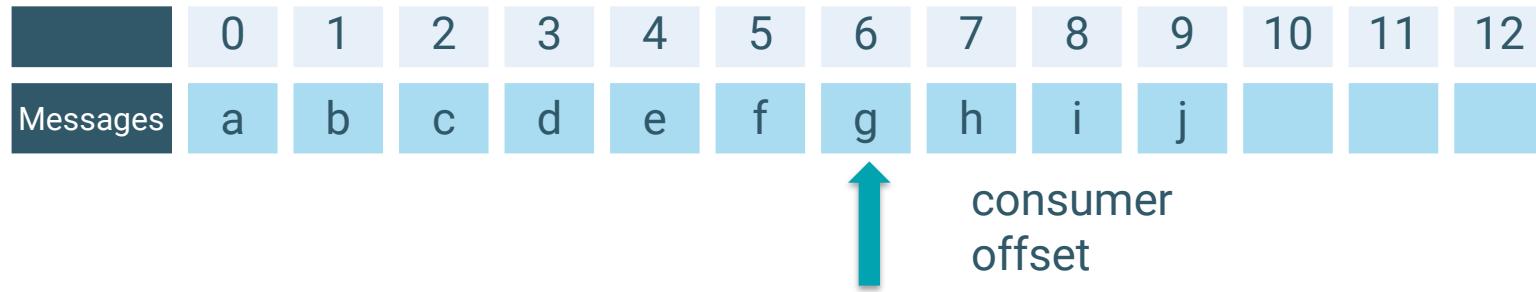
- Messages are consumed in Kafka by a **consumer group**
- Each individual consumer is labeled with a group name
- Each message in a topic is sent to one consumer in the group
- In other words, messages are consumed at the group level, not at the individual consumer level
 - This allows for fault tolerance and scalability of consumers
- This design allows for both queue and publish-subscribe models:
 - If you need a **queue** behavior, then simply place all consumers into the same group
 - If you need a **publish-subscribe** model, then create multiple consumer groups that subscribe to a topic

CONSUMER GROUPS



THE CONSUMER OFFSET

It is up to the consumer to maintain its offset in the partition (stored in a special topic named `_consumer_offsets`)



- This has several key benefits, including:
 - **performance:** there is no back-and-forth acknowledging of message consumption
 - **simplicity:** the consumer only has to maintain a single integer value for its state, which can be easily stored and shared between consumers (if a failure occurs)
 - **re-consume messages:** it becomes trivial for a consumer to re-consume messages

MESSAGE DELIVERY GUARANTEES

- Kafka guarantees at-least-once delivery by default
- At-most-once delivery is possible by disabling retries on the producer (when a commit fails)
- Exactly-once delivery is possible (with clever coordination of your consumers and the consumer offset)
- Other guarantees:
 - Messages in a partition are stored in the order that they were sent by the publisher
 - Each partition is consumed by exactly one consumer in the group
 - That consumer is the only reader in the group of that partition in the group
 - Messages are consumed in order
 - Messages committed to the log are not lost for up to N-1 broker failures

IN-SYNC REPLICAS

- Kafka replicates the messages in each partition across multiple brokers
- New messages are always appended to the leader
- A follower that keeps up is called an ISR, or in-sync replica, which means:
- A message is considered committed when all ISRs have a copy of the message

Kafka's Omnipresence Has Led to the Onset of “Kafka Blindness”

□ What is “Kafka Blindness”?

- Customers who use Kafka today struggle with monitoring / “seeing”/troubleshooting what is happening in their clusters

□ Who is Affected?

- Platform Operation Teams
- Developers / DevOps Teams
- Security / Governance Teams

□ What are the Symptoms?

- Difficulty seeing who is producing and consuming data
- Difficulty understanding the flow of data from producers -> topics ↗ consumers
- Difficulty troubleshooting/monitoring.

Streams Messaging Manager (SMM)

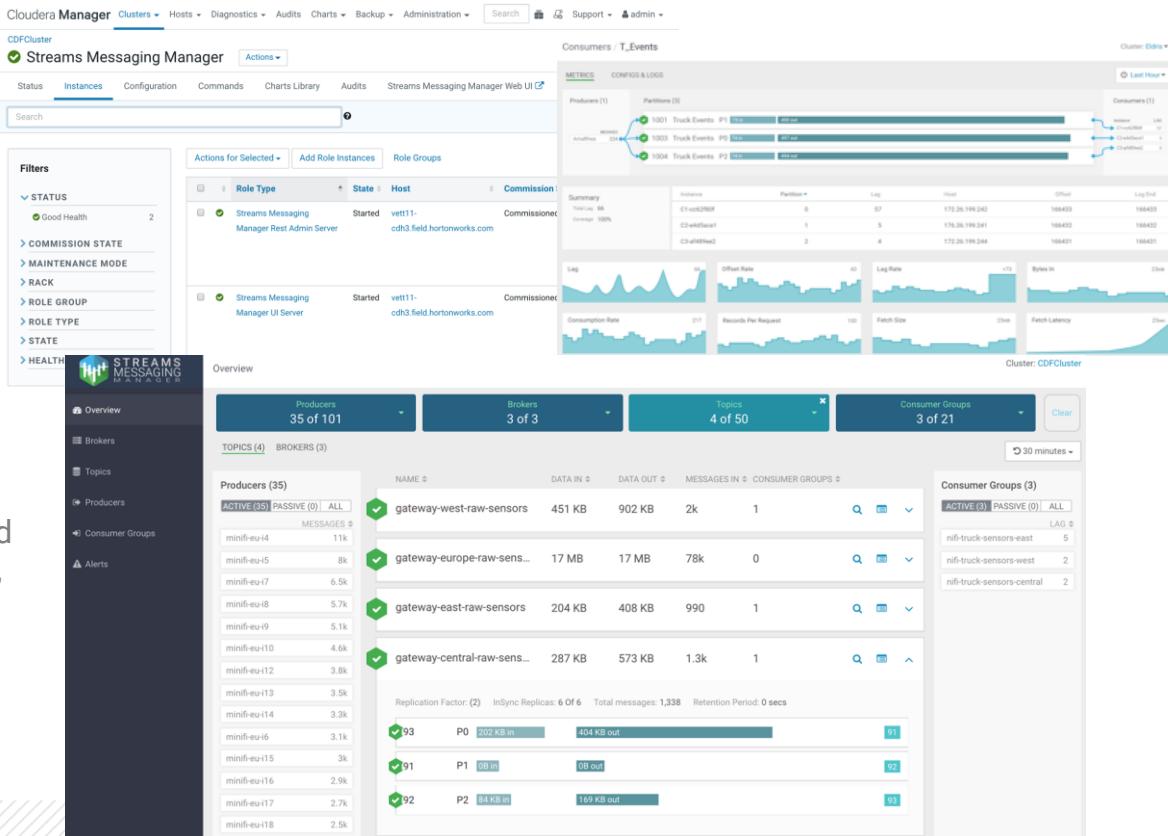
Curing Kafka Blindness

Problem Statement / Requirements

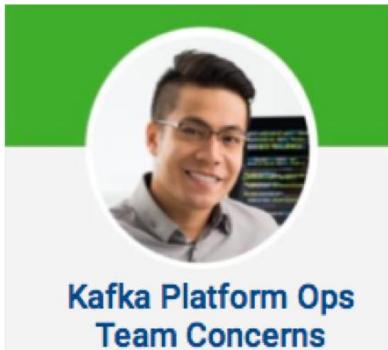
- Customers struggle with monitoring / “seeing”/ troubleshooting what is happening in their Kafka clusters
- They need an enterprise Kafka monitoring solution full integrated with platform services including CM and Sentry

Solution

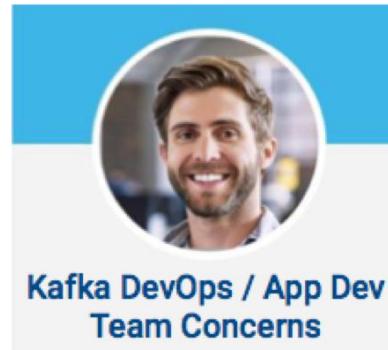
- SMM provides single monitoring Dashboard for Kafka Clusters across 4 entities: Broker, Producer, Topic, Consumer
- SMM integrated with CM Log Search
- Kerberized based authentication and rich access Control via Sentry
- DataPlane Platform NOT required



SMM Addresses the Distinct Needs of 3 Personas/Teams



Kafka Platform Ops
Team Concerns



Kafka DevOps / App Dev
Team Concerns



Governance / Security
Team Concerns

Concerned with monitoring the overall health of the cluster and the infrastructure it runs on

Concerned with monitoring the Kafka entities associated with their apps

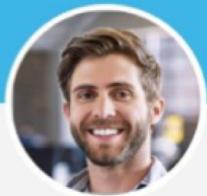
Concerned with audit, compliance, access control & chain of custody requirements

SMM for the Platform Ops Team



Do I have any offline topic partitions?	How many total active producers/consumers is there currently?	Which producers are generating the most data right now?
Which consumer group is falling behind the most?	Which producers are generating the most data right now?	Are any of my brokers running out of disk space?
Are any of my brokers down?	Are there any skewed partitions for a broker?	How full the cluster is being used, how much capacity do I have available per broker / cluster?
Which producers are generating the most data right now?	How many total active producers and consumers exist now?	Which partitions are located on each broker?
What is the throughput in/out for a given partition on that broker?	Are any of my brokers running hot? Which broker has the highest throughput in and out rates?	Which of my topics has produced/consumers the most messages over the last N minutes/hours?
What hosts are my brokers located on?	How many total topics does my Kafka cluster have?	
Are all my replicas in my topic in-synch?		

SMM for the DevOps/AppDev Teams



Kafka DevOps / App Dev
Team Concerns

Find all entities (producer, consumers, topics) associated with my app.

What brokers holds the partitions for my app topics?

What is the total number of messages into my topic over the last N minutes/hours?

Are there consumers in a consumer-group for a given topic slow/falling behind?

What topic(s) are the consumer group consuming messages from?

What is the retention rate for my app topics?

Who are all the producers and consumers connected to my app topics?

Did a consumer rebalance occur for a given topic?

How many active consumers instances are in a given consumer group?

Are any of my consumers/consumer-groups that are under-consuming?

What is the replication factor for my app topics?

What type of events are in my application topics? What does the event look like?

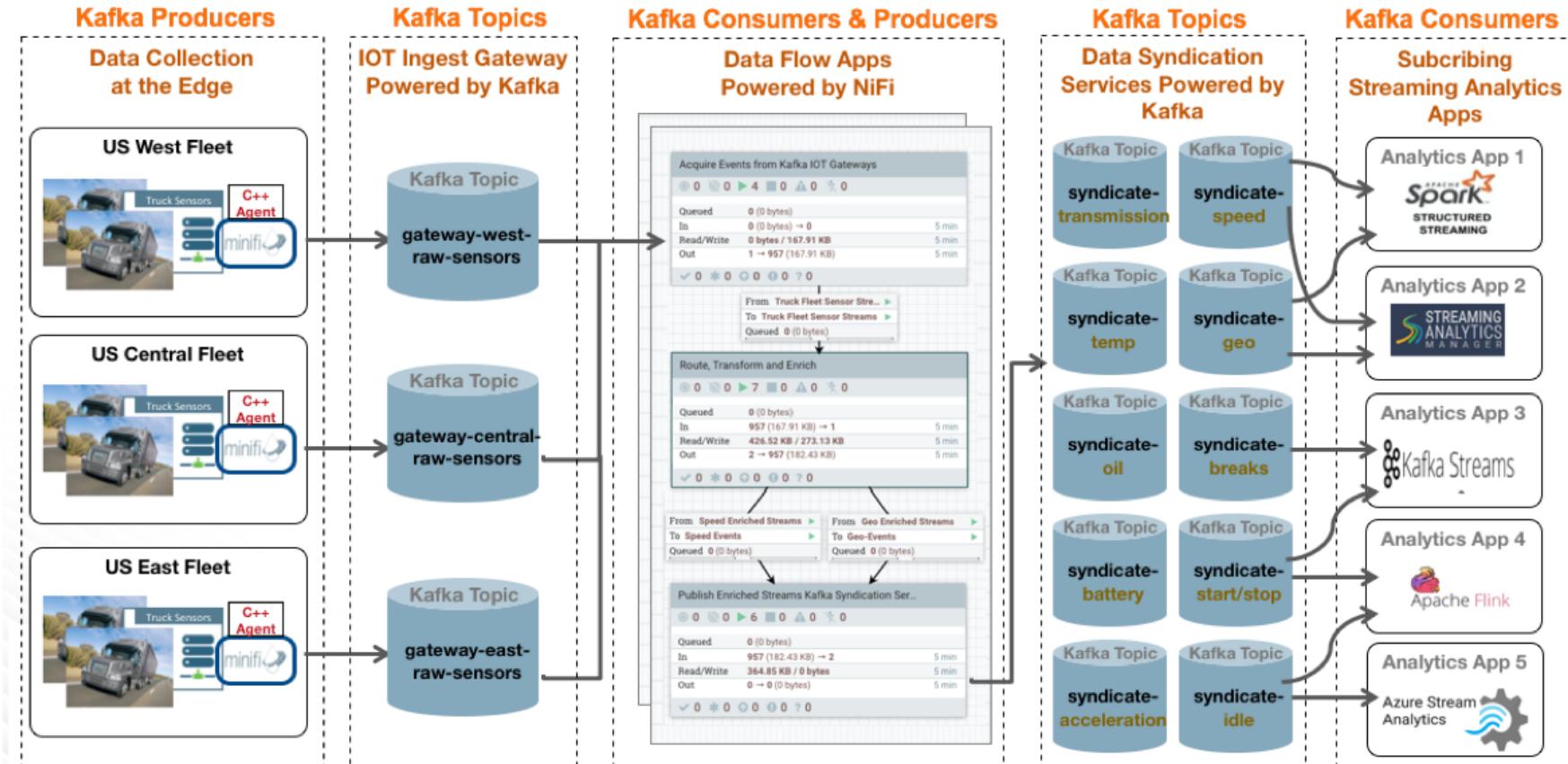
Are any of my consumers/consumer-groups that are over-consuming?

SMM for the Security and Governance Team



When was a topic created?	What is the schema for a given topic?	What is the lineage of a kafka topic?
How has the schema evolved for a given topic?	How does data flow across multiple kafka hops?	Who has edited the ACL policies of a given topic?
Which consumers have consumed from a topic?	What are the ACL policies for a given topic?	When were additional brokers added to the topic?
Which users/groups/service accounts have read from a given topic?	Which users/groups/service accounts have sent data to a topic?	Which producers have sent data to the topic?
When was the topic configuration last modified?		

Dev Ops / App Dev Persona – Monitoring the Streaming Truck App



Kafka Challenges

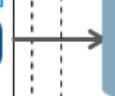
Shared Schema Registry

How do I associate schemas for messages in Kafka topics

Data Collection

Kafka as a Service

I want to provide my users self service capabilities for Kafka: creation of clusters, monitoring, management



US Central Fleet

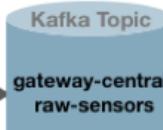
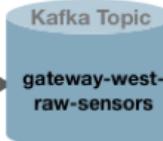


Balanced Kafka at Scale
As my clusters grow larger, I need the system to detect and automate balancing policies



Kafka Topics

IOT Ingest Gateway Powered by Kafka

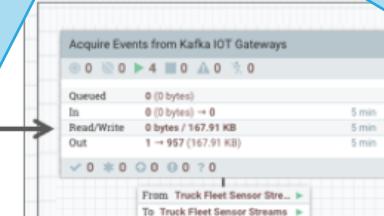


Agility and Self-Service

How do I develop, deploy & manage Kafka producers and consumers without code in self service manner

Consumers & Producers

Data Flow API Powered by NiFi



Kafka Replication

I need to replicate/move data across Kafka clusters

SQL/Access Patterns on Kafka

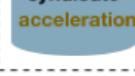
I want to treat Kafka as a table and execute SQL on it for ETL and analytics

Kafka Blindness

How do I manage/monitor the different kafka clusters, topics, consumers, and producers

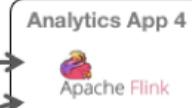
Kafka Topics

Data Syndication Services Powered by Kafka



Kafka Consumers

Subscribing Streaming Analytics Apps



Main Dashboard View

The Kafka cluster called OneNodeCluster is selected

Cluster: OneNodeCluster

Overview

Producers 29

Brokers 1

Topics 40

Consumer Groups 16

Clear

TOPICS (40) BROKERS (1)

30 minutes ▾

Producer Details

NAME	DATA IN	DATA OUT	MESSAGES	CONSUMER GROUPS
syndicate-tra...	783 KB	0B	3.5k	0
syndicate-sp...	0B	0B	0	0
syndicate-sp...	0B	0B	0	0
syndicate-oil	940 KB	0B	4.3k	0
syndicate-ge...	0B	0B	0	0
syndicate-ge...	0B	0B	0	3
syndicate-bat...	854 KB	0B	3.9k	0

Consumer Groups (16)

LAG
load-optimiz... 1.3m
fuel-micro-se... 0.6m
supply-chain... 0.4m
predictive-mi... 0.3m
energy-micro... 0.3m
audit-micro-... 0.2m
compliance-... 0.2m
adjudication-... 0.2m
approval-mic... 0.1m
nifi-truck-sen... 2
nifi-truck-sen... 2
flink-analytic... 0
kafka-strea... 0
spark-strea... 0

Producer List

NAME	MESSAGES
geo-critical-e...	89k
geo-critical-e...	45k
minifi-eu-i1	45k
load-optimiz...	42k
geo-critical-e...	30k
minifi-eu-i2	23k
geo-critical-e...	23k
fuel-apps	21k
geo-critical-e...	18k
minifi-eu-i3	15k
supply-chain...	14k
predictive-ap...	11k
energy-apps	8.5k
audit-apps	7.1k
compliance-...	6.1k

Find the Most Active Producer in my Cluster

STREAMS MESSAGING MANAGER

Overview Cluster: OneNodeCluster

Brokers: 1 Topics: 40 Consumer Groups: 16

Click on Messages to sort on messages sent by all producers in the last 30 mins

A Kafka producer called geo-critical-event-collector-i1 is the most active producer sending 89K messages in the last 30 mins

NAME	DATA IN	DATA OUT	MESSAGES	CONSUMER GROUPS	LAG
syndicate-tra...	783 KB	0B	3.5k	0	1.3m
syndicate-sp...	0B	0B	0	0	0.6m
syndicate-sp...	0B	0B	0	0	0.4m
syndicate-oil	940 KB	0B	4.3k	0	0.3m
syndicate-ge...	0B	0B	0	0	0.3m
syndicate-ge...	0B	0B	0	3	0.2m
syndicate-bat...	854 KB	0B	3.9k	0	0.2m

Consumer Groups (16)

- load-optimiz...
- fuel-micro-se...
- supply-chain...
- predictive-mi...
- energy-micro...
- audit-micro...
- compliance-...
- adjudication-...
- approval-mic...
- nifi-truck-sen...
- nifi-truck-sen...
- flink-analytic...
- kafka-strea...
- spark-strea...

Find the Consumer Who Has Fallen Behind the Most

The screenshot shows the Cloudera Streams Messaging Manager interface with the following details:

- Overview** tab selected.
- Cluster:** OneNodeCluster
- Producers:** 29
- Brokers:** 1
- Topics:** 40
- TOPICS (40) BROKERS (1)**
- Producers (29)** table:
 - syndicate-tra... (783 KB, 0B, 3.5k, 0)
 - syndicate-sp... (0B)
 - syndicate-sp... (0B, 0B, 0, 0)
 - syndicate-oil (940 KB, 0B, 4.3k, 0)
 - syndicate-ge... (0B, 0B, 0, 0)
 - syndicate-ge... (0B, 0B, 0, 3)
 - syndicate-bat... (854 KB, 0B, 3.9k, 0)
- Consumer Groups (16)** table:
 - load-optimiz... (1.3m)
 - fuel-micro-se... (0.6m)
 - supply-chain... (0.4m)
 - predictive-mi... (0.3m)
 - energy-micro... (0.3m)
 - audit-micro... (0.2m)
 - compliance-m... (0.2m)
 - adjudication-m... (0.2m)
 - approval-mic... (0.1m)
 - nifi-truck-sen... (2)
 - nifi-truck-sen... (2)
 - flink-analytic... (0)
 - kafka-strea... (0)
 - spark-strea... (0)
- A callout box highlights the consumer group "load-optimizer-micro-service" with the text: "Consumer group named load-optimizer-micro-service has significantly more lag (1.3m) than any other consumer in the cluster."
- A callout box highlights the "Consumer Groups" section with the text: "Click on LAG to sort on consumer lag across all consumers in the last 30 mins".

Broker Centric View – View Details of the Brokers in My Cluster

STREAMS MESSAGING MANAGER

Overview Cluster: OneNodeCluster

Producers 29 Brokers 40 Topics 40 Consumer Groups 16 Clear

TOPICS (40) BROKERS (1)

Click on the Brokers tab to see a broker centric view of the Dashboard

Producers (29)

ACTIVE PASSIV... ALL

MESSAGES ↴

geo-critical-e...	89k	8	ip-10-0-1-248.us-west...	93 MB	0.4m	194
geo-critical-e...	45k					
minifi-eu-i1	45k					
load-optimiz...	42k					
geo-critical-e...	30k					
minifi-eu-i2	23k					
geo-critical-e...	23k					
fuel-apps	21k					
geo-critical-e...	18k					
minifi-eu-i3	15k					
supply-chain...	14k					
predictive-ap...	11k					
energy-apps	8.5k					
audit-apps	7.1k					
compliance-...	6.1k					

PUTS MESSAGES IN PARTITIONS REPLICAS ↴

Consumer Groups (16)

ACTIVE PASSIV... ALL

LAG ↴

load-optimiz...	1.3m
fuel-micro-se...	0.6m
supply-chain...	0.4m
predictive-mi...	0.3m
energy-micro...	0.3m
audit-micro...	0.2m
compliance-...	0.2m
adjudication-...	0.2m
approval-mic...	0.1m
nifi-truck-sen...	2
nifi-truck-sen...	2
flink-analytic...	0
kafka-strea...	0
spark-strea...	0

RESERVED

Broker Centric View: Find my Hottest Broker – Broker with Highest Throughput In

The screenshot shows the Streams Messaging Manager interface with the following sections:

- Overview:** Shows a summary of Producers (59), Topics (27), and Consumer Groups (26). A green callout "Step 1" points to the "Producers" section.
- Producers:** A detailed view of 59 producers, including active, passive, and all. A green callout "Step 1" points to the "ACTIVE (59)" tab. A green callout "Analysis" points to the top producer, Broker 1001, with the text: "Broker 1001 has the highest rate of data in over the last 30 mins. 80K messages totaling 17MB".
- Topics:** A list of 27 topics, including geo-critical-event-collec..., minifi-eu-i6, audit-apps, geo-critical-event-collec..., minifi-eu-i7, compliance-apps, minifi-eu-i8, geo-critical-event-collec..., minifi-eu-i9, approval-apps, and minifi-eu-i10.
- Consumer Groups:** A list of 26 consumer groups, including route-micro-service, load-optimizer-micro-se..., fuel-micro-service, supply-chain-micro-ser..., predictive-micro-service, energy-micro-service, audit-micro-service, compliance-micro-servi..., adjudication-micro-servi..., approval-micro-service, flink-analytics-geo-event, kafka-streams-analytics..., spark-streaming-analyti..., nifi-truck-sensors-east, nifi-truck-sensors-west, nifi-truck-sensors-central, ranger_entities_consum..., and atlas.

Step 1
Click on the Brokers tab to see a broker centric view of the Dashboard

Step 2
Click on Throughput to sort on data in across all brokers

Analysis
Broker 1001 has the highest rate of data in over the last 30 mins. 80K messages totaling 17MB

Name	Throughput	Messages In	Partitions	Replicas
1001 c-dps-connected-dp13.field.hortonwor...	17MB	80k	21	36
1002 c-dps-connected-dp12.field.hortonwor...	16MB	75k	16	34
1005 c-dps-connected-dp11.field.hortonwor...	14MB	63k	15	31
1003 c-dps-connected-dp14.field.hortonwor...	10MB	42k	16	30
1004 c-dps-connected-dp15.field.hortonwor...	7MB	33k	14	30

This is a multi-node cluster. What you are working with is a single node cluster.

Find Partitions on a given Broker and Understand flow of data flow from Producer to selected Broker Partition to Consumer

The screenshot shows the Streams Messaging Manager interface with the following details:

- Overview:** Producers (1 of 83), Brokers (3 of 5), Topics (1 of 27).
- Producers:** nifi-syndicate-speed-avro (1.6k messages).
- Brokers:** 1001 (c-dps-connected-dp13.field.hortonworks.com:6...), throughput 25MB, partitions 21, replicas 36.
- Topics:** syndicate-speed-event-avro - P0 (Topic details: DATA IN 93385, DATA OUT 14260475). Other topics listed include syndicate-trans, syndicate-speed, syndicate-oil, syndicate-geo-ev, syndicate-battery, syndicate-all-geo, supply-chain, route-planning, predictive-alerts.
- Consumer Groups:** sam-speed-stream-consum, sam-speed-stream-consum, sam-speed-stream-consum.

Step 1: Click on Panel expand to get more details on the broker like all partitions that are stored on the broker.

Step 2 - Analysis: Note that partition 0 of topic syndicate-speed has high throughput-out on that partition.

Step 3: Click on the partition and see who are all the producers and consumers sending/consuming from that partition. There is 1 producer and 3 consumer groups explaining why the high throughput out vs in.

Analyze Detailed Broker Host Metrics – Cloudera Manager Integration

Overview

Cluster: OneNodeCluster

TOPICS (5) BROKERS (1)

NAME # THROUGHPUT # MESSAGES IN # PARTITIONS # REPLICAS #

ACTIVE (1...) PASSIVE (0...) ALL

MESSAGES

NAME	#	THROUGHPUT	MESSAGES IN	# PARTITIONS	# REPLICAS
ip-10-0-1-248.us-west-2.compute...	8	93 MB	0.4m	194	194
minifl-eu-i1	45k				
minifl-eu-i2	23k				
minifl-eu-i3	15k				
minifl-truck-w1	798				
minifl-truck-w2	660				
minifl-truck-w3	568				
minifl-truck-c1	496				
minifl-truck-c2	447				

FREE MEMORY ————— FREE DISK ————— CPU IDLE 14.98 LOAD AVERAGE 2.07

DISK I/O 1059780.20

Producer Consumer

ACTIVE (0...) PASSIVE (0...) ALL

LAG

NAME	#
nifi-truck-sensors...	2
nifi-truck-sensors...	2

Cloudera Manager Clusters Hosts Diagnostics Audits Charts Backup Administration

Search

OneNodeCluster / Kafka / ip-10-0-1-248

❗ Kafka Broker (id: 8) (Active Controller) Actions

30 minutes preceding S

Status Configuration Processes Commands Charts Library Audits Log Files Stacks Logs Quick Links

Health Tests Create Trigger

Host Health Suppress...
The health of this role's host is bad. The following health tests are bad:
agent parcel directory.

Show 7 Good

Log Directory Free Space Suppress...
This role has no Log Directory configured.

Charts

30m 1h 2

Messages Received ⓘ messages / sec
200
100
0
12:30 12:45
KAFKA_BROKER (ip-10-0-1-248.us-west-2.compu... 247

Bytes Received ⓘ bytes / second
39.1K/s
19.5K/s
0
12:30 12:45
KAFKA_BROKER (ip-10-0-1-248.us-west-2.co... 54.1K/s

Health History

Host Health Bad 10:41 PM

Host Health Concerning 9:40 PM

3 Became Good Sep 20 11:14 PM

3 Became Good Sep 20 11:11 PM

Bytes Fetched ⓘ bytes / second
500b/s
0
12:30 12:45
KAFKA_BROKER (ip-10-0-1-248.us-west-2.compu... 427b/s

Partitions ⓘ partitions
150
100
50
0
12:30 12:45
KAFKA_BROKER (ip-10-0-1-248.us-west-2.compu... 194

Leader Replicas ⓘ

Offline Partitions ⓘ

Click on the CM icon on the broker panel and the CM host detail view for that broker is displayed providing host level metrics and a view of other services running on that host

Keyword Search via Log Search

Overview

The screenshot shows the Cloudera Manager interface for a cluster named "OneNodeCluster". The top navigation bar includes tabs for Overview, Producers, Brokers, Topics, Consumer Groups, and Diagnostics. The Diagnostics tab is currently selected. A green callout box points to the "Logs" icon in the top right corner of the main content area.

Logs

Keywords: Enter Search Keywords or Regular Expression

Sources: Services (checked), Cloudera Manager Agent (checked), Cloudera Manager Server (checked)

Services: Kafka (selected)

Hosts: ip-10-0-1-248.us-west-2.compute.internal (selected)

Role Types: Kafka Broker (selected)

Minimum Log Level: WARN

Timeout (sec): 60

Search

30 minutes preceding Sep 22, 12:44 AM UTC

1 Machine(s) Searched (311ms), 0 Error(s), More Statistics

Hosts	Log Level	Time	Source	Message
ip-10-0-1-248.us-west-2.compute.internal	ERROR	September 22, 2019 12:14 AM	KafkaApis	[KafkaApi-8] Number of alive brokers '1' does not meet the required replication factor '3' for the transactions state topic (configured via 'transaction.state.log.replication.factor'). This error can be ignored if the cluster is starting up and not all brokers are up yet. View Log File
ip-10-0-1-248.us-west-2.compute.internal	ERROR	September 22, 2019 12:14 AM	KafkaApis	[KafkaApi-8] Number of alive brokers '1' does not meet the required replication factor '3' for the transactions state topic (configured via 'transaction.state.log.replication.factor'). This error can be ignored if the cluster is starting up and not all brokers are up yet. View Log File
ip-10-0-1-248.us-west-2.compute.internal	ERROR	September 22, 2019 12:14 AM	KafkaApis	[KafkaApi-8] Number of alive brokers '1' does not meet the required replication factor '3' for the transactions state topic (configured via 'transaction.state.log.replication.factor'). This error can be ignored if the cluster is starting up and not all brokers are up yet. View Log File

Click on the Log Search icon on the broker panel and the Log Search detail view is displayed. This enables you to search for specific keywords and to filter for specific log levels, components, and time ranges.

SMM DevOps/App Dev Use Cases

Topic Centric Dashboard View: Filter on Topics associated with my Topic

STREAMS MESSAGING MANAGER

Overview Cluster: orlandostreamcluster

Producers 83 | Brokers 5 | Topics 27 | Consumer Groups

TOPICS (27) BROKERS (5)

Producers (83)

	NAME	DATA IN	DATA OUT	MESSAGES IN	CONSUMER GROUPS
ACTIVE (83)	syndicate-all-geo-critical-eve...	26MB	0B	88k	0
PASSIVE (0)	route-planning	26MB	28	23k	1
ALL	gateway-europe-raw-sensors	18MB	0B	88k	0
	load-optimization	5MB	28KB	23k	1
	fuel-logistics	1MB	28KB	6.5k	1
	supply-chain	863KB	28KB	4.3k	1
	audit-events	804KB	27KB	3.9k	1
	compliance	697KB	29KB	3.4k	1
	predictive-alerts	648KB	28KB	3.3k	1

Topics (27)

Use the Filter to filter on topics and select all the IOT gateway topics

ACTIVE (18) PASSIVE (6) ALL

	NAME	DATA IN	DATA OUT	MESSAGES IN	CONSUMER GROUPS
ACTIVE (18)	gatew	26MB	0B	88k	0
PASSIVE (6)	route-micro-service	0.2m	0B	0	0
ALL	load-optimizer-micro-service	12k	0B	0	0
	fuel-micro-service	6k	0B	0	0
	supply-chain-micro-service	4k	0B	0	0
	predictive-micro-service	3k	0B	0	0
	energy-micro-service	2.3k	0B	0	0
	audit-micro-service	1.9k	0B	0	0
	compliance-micro-service	1.7k	0B	0	0
	adjudication-micro-service	1.4k	0B	0	0
	approval-micro-service	1.3k	0B	0	0
	flink-analytics-geo-event	525	0B	0	0
	kafka-streams-analytics-geo...	525	0B	0	0
	spark-streaming-analytics-g...	525	0B	0	0
	nifi-truck-sensors-west	2	0B	0	0
	nifi-truck-sensors-east	2	0B	0	0
	nifi-truck-sensors-central	1	0B	0	0
	ranger_entities_consumer	1	0B	0	0
	atlas	0	0B	0	0

Clear 0 minutes

Intelligent Filtering – Selected Topics causes Producers / Consumer to be Intelligent Filtered

Overview

Cluster: OneNodeCluster

The screenshot shows the Apache NiFi interface with four main sections: Producers, Brokers, Topics, and Consumer Groups.

- Producers:** Shows 12 of 29 producers. A green callout box labeled "Intelligent Filtering" states: "SMM automatically filters the producers associated with the selected topics. 12 of the 29 producers have been identified as sending data to the 5 topics selected".
- Topics:** Shows 5 of 0 topics. A green callout box labeled "User Action" states: "5 IOT Gateway topics have been selected".
- Consumer Groups:** Shows 3 of 17 consumer groups. A green callout box labeled "Intelligent Filtering" states: "SMM automatically filters the consumers associated with the selected topics. 3 of the 17 consumers have been identified as consuming data from the 5 topics selected".

The central Topics section displays a list of five selected topics:

NAME	DATA IN	DATA OUT	PRODUCERS	CONSUMERS	LAG
gateway-west-raw-sensors	411 KB	0B	5	0	2
gateway-europe-raw-sensor	16 MB	0B	5	0	0
gateway-east-raw-sensors	198 KB	0B	5	0	0
gateway-east-raw-sensor	0B	0B	0	0	0
gateway-central-raw-sensors	264 KB	0B	1.3k	1	0

Find the Hottest Topic – Topic With Highest Throughput-In

Cluster: OneNodeCluster

Overview

Producers 12 of 29

Brokers 1 of 1

Topics 5 of 40

Consumer Groups 3 of 17

Clear

TOPICS (5) BROKERS (1)

Analysis

Kafka topic called gateway-europe-raw-sensors has more data being sent to it than any other topic: 88K messages totaling 16 MB in the last 30 mins

Name	Data In	Data Out	Offset	Partition	Consumer Groups
gateway-west-raw-sensors	411 KB	0B	0	0	nifi-truck-sensors-west
gateway-europe-raw-sensors	16 MB	0B	0	0	nifi-truck-sensors-east
gateway-east-raw-sensors	198 KB	0B	0	0	nifi-truck-sensors-central
gateway-east-raw-sensor	0B	0B	0	0	
gateway-central-raw-sensors	264 KB	0B	1.3k	1	
minifi-eu-i2	23k				
minifi-eu-i3	15k				
minifi-truck-w1	788				
minifi-truck-w2	662				
minifi-truck-w3	562				
minifi-truck-c1	496				
minifi-truck-c2	442				
minifi-truck-c3	398				
minifi-truck-e1	364				
minifi-truck-e2	332				
minifi-truck-e3	304				

Step 1
Click on DATA IN to sort on data-in across all topics in the last 30 mins

gate

Name

gateway-west-raw-sensors

gateway-europe-raw-sensors

gateway-east-raw-sensors

gateway-east-raw-sensor

gateway-central-raw-sensors

Consumer Groups (3)

ACTIVE (3) PASSIVE (0) ALL

LAG

nifi-truck-sensors-west 2

nifi-truck-sensors-east 0

nifi-truck-sensors-central 0

How are the Partitions Laid out for the Topic? Who are the Producers and Consumers? Are there any Partition Skews?

Overview

Producers 12 of 30

Brokers 1 of 1

Topics 5 of 40

Consumers 3

Cluster: Unenforcedcluster

Step 1
Expand topic panel to see more details of the topic that has high data-in rates

Step 2
Click on the Topic to see who are all the producers sending data to the topic

TOPICS (5) BROKERS (1)

Producers (12)

ACTIVE (12) PASSIVE (0) ALL

Producer	Messages
minifi-eu-i1	11k
minifi-eu-i2	5.5k
minifi-eu-i3	3.7k
minifi-truck-w1	188
minifi-truck-w2	162
minifi-truck-w3	136
minifi-truck-c1	120
minifi-truck-c2	104
minifi-truck-c3	96
minifi-truck-e1	88
minifi-truck-e2	80
minifi-truck-e3	74

NAME: gateway-europe-raw-sensors

DATA IN: 96 KB

DATA OUT: 0B

MESSAGES IN: 486

CONSUMER GROUPS: 1

Replication Factor: (1) InSync Replicas

P0 1 MB in

P1 343 KB in 0B out

P2 1 MB in 0B out

P3 1 MB in 0B out

P4 195 KB in 0B out

Topic: gateway-europe-raw-sensors - P1

DATA IN 351606

DATA OUT 0

PROFILE FILTER EXPLORE

Consumer Groups (3)

ACTIVE (3) PASSIVE (0) ALL

Consumer Group	LAG
nifi-truck-sensors-west	2
nifi-truck-sensors-east	0
nifi-truck-sensors-ce...	0

ALL PARTITIONS

Analysis
Note that for each partition there is no data going out (0B) and we see no data going to any consumer groups. This means that while the topic has lots of producers, there is no consumers which could indicate a problem

RESERVED

2 0 1

How does Data Flow between Producers to Topics to Consumers?

The screenshot shows the Confluent Cloud UI with several annotations:

- Step 1**: Expand details of a topic that has consumers
- Step 2**: Click on the topic to see all producers sending data to it and all consumers consuming from it
- Analysis 1**: We have 3 truck producers from the west fleet sending data to gateway-west topic and a NiFi consumer called truck-sensors-west consuming from it
- Analysis 2**: Note that there is no data in 2 of the 4 partitions. This could be a partition/event key skew issue

Topics (4 of 25) details for gateway-europe-raw-sensors:

Partition	Replica	Data In	Data Out
1003	P0	0B in	0B out
1001	P1	0B in	0B out
1002	P2	54MB in	54MB out
1003	P3	82MB in	82MB out

Consumer Groups (3) details for nifi-truck-sensors-west:

Consumer Group	Count
nifi-truck-sensors-west	2
nifi-truck-sensors-east	2
nifi-truck-sensors-central	1

How does Data Flow between Producers to Topics to Consumers?

Overview

Cluster: OneNodeCluster

Producers
12 of 30

Step 2

Click on the topic to see all
producers sending data to
it and all consumers
consuming from it

Brokers
1 of 1

Topics
5 of 40

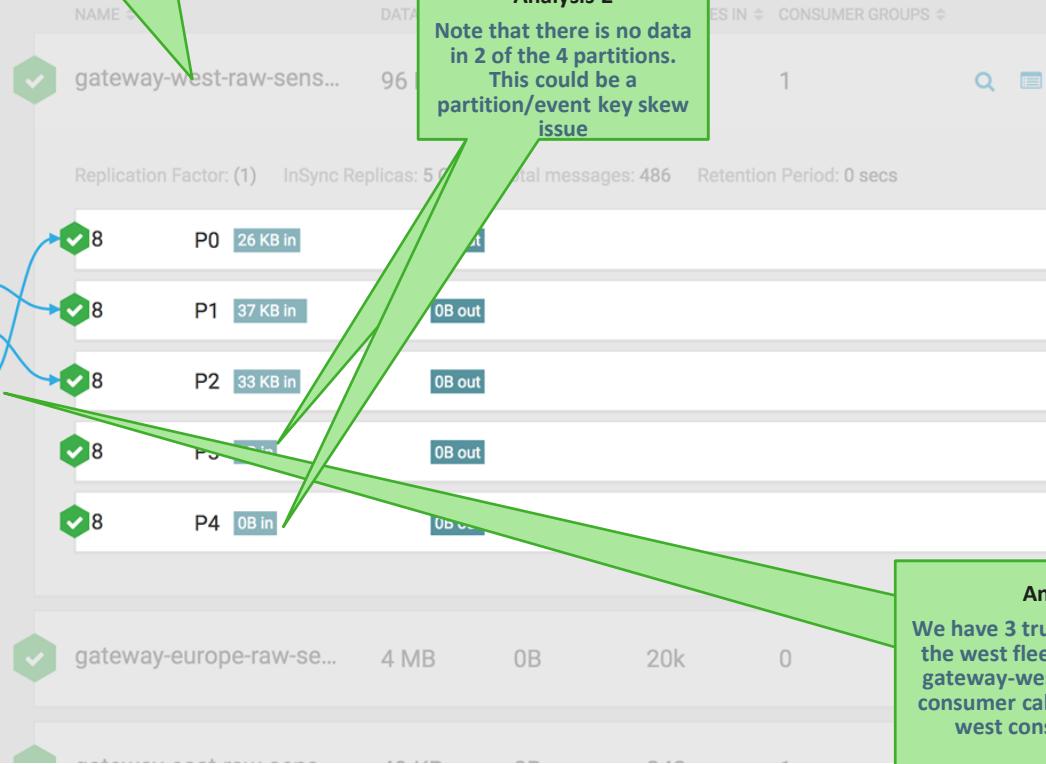
Step 1
Expand details of a topic
that has consumers

TOPICS (5) BROKERS (1)

Producers (12)

ACTIVE (12) PASSIVE (0) ALL

	MESSAGES
minifi-eu-i1	11k
minifi-eu-i2	5.5k
minifi-eu-i3	3.7k
minifi-truck-w1	188
minifi-truck-w2	162
minifi-truck-w3	136
minifi-truck-c1	120
minifi-truck-c2	104
minifi-truck-c3	96
minifi-truck-e1	88
minifi-truck-e2	80
minifi-truck-e3	74



Explore/Search Messages in the Kafka Topic

Overview

Producers
12 of 30

Brokers
1 of 1

Topics
5 of 40

TOPICS (5) BROKERS (1)

Producers (12)

ACTIVE (12) PASSIVE (0) ALL

MESSAGES (11k)

minifi-eu-i1

11k

minifi-eu-i2

5.5k

minifi-eu-i3

3.7k

minifi-truck-w1

188

minifi-truck-w2

162

minifi-truck-w3

136

minifi-truck-c1

120

minifi-truck-c2

104

minifi-truck-c3

96

minifi-truck-e1

88

NAME		DATA IN	DATA OUT	MESSAGES IN	CONSUMER COUNT
gateway-west-raw-sens...	96 KB	0B	186	1	1
Replication Factor: (1)	InSync Replicas: 1				
minifi-eu-i1	11k				
minifi-eu-i2	5.5k				
minifi-eu-i3	3.7k				
minifi-truck-w1	188	26 KB in	0B out	1	1
minifi-truck-w2	162	P0	37 KB in	0B out	1
minifi-truck-w3	136	P1	33 KB in	0B out	1
minifi-truck-c1	120	P2	0B in	0B out	1
minifi-truck-c2	104	P3	0B in	0B out	1
minifi-truck-c3	96	P4	0B in	0B out	1
minifi-truck-e1	88				

Topic: gateway-west-raw-sensors - P1
DATA IN 37824
DATA OUT 0
PROFILE FILTER EXPLORE

Click on the explorer icon to search for events in the Kafka Topic

Topics / gateway-west-raw-sensors

METRICS DATA EXPLORER CONFIGS LATENCY

DESERIALIZER: Keys: String Values: String

FROM OFFSET
Partition 1 409 0 141 282 TO OFFSET 424

Offset	Timestamp	Key	Value
409	Mon, Sep 23 2019, 9:20:35	10	2019-09-23 16:20:35.482 1569255635482 truck_speed_event 718 10 George Vetticaden 8 Saint Louis to Tulsa 66
410	Mon, Sep 23 2019, 9:20:39	10	2019-09-23 16:20:39.711 1569255639711 truck_geo_event 718 10 George Vetticaden 8 Saint Louis to Tulsa Normal 38.09 -91.3 1
411	Mon, Sep 23 2019, 9:20:39	10	2019-09-23 16:20:39.712 1569255639712 truck_speed_event 718 10 George Vetticaden 8 Saint Louis to Tulsa 58
412	Mon, Sep 23 2019, 9:20:44	10	2019-09-23 16:20:44.731 1569255644731 truck_geo_event 718 10 George Vetticaden 8 Saint Louis to Tulsa Normal 38.09 -91.44 1
413	Mon, Sep 23 2019, 9:20:44	10	2019-09-23 16:20:44.731 1569255644731 truck_speed_event 718 10 George Vetticaden 8 Saint Louis to Tulsa 72
414	Mon, Sep 23 2019, 9:20:49	10	2019-09-23 16:20:49.172 1569255649172 truck_geo_event 718 10 George Vetticaden 8 Saint Louis to Tulsa Normal 38.04 -91.55 1
415	Mon, Sep 23 2019, 9:20:49	10	2019-09-23 16:20:49.172 1569255649172 truck_speed_event 718 10 George Vetticaden 8 Saint Louis to Tulsa 69
416	Mon, Sep 23 2019, 9:20:53	10	2019-09-23 16:20:53.14 1569255653140 truck_geo_event 718 10 George Vetticaden 8 Saint Louis to Tulsa Normal 37.99 -91.69 1

SMM 1.2 New Features

Topic Lifecycle Management

The screenshot shows the Streams Messaging Manager interface. On the left, a sidebar menu includes: Overview, Brokers, Topics (selected), Producers, Consumer Groups, Alerts, and Help. The main area displays a list of topics: syndicate-geo-event-json-2, syndicate-geo-event-avro. Below the list is a detailed view of the selected topic, "syndicate-geo-event-json-2". The "Add Topic" dialog is open, showing the following configuration:

- TOPIC NAME:** syndicate-geo-event-json-2
- PARTITIONS:** 3
- Availability:** MAXIMUM (selected)
- REPLICATION FACTOR 3 MIN INSYNC REPLICA 2**
- HIGH:** REPPLICATION FACTOR 3 MIN INSYNC REPLICA 1
- MODERATE:** REPPLICATION FACTOR 2 MIN INSYNC REPLICA 1
- LOW:** REPPLICATION FACTOR 1 MIN INSYNC REPLICA 1
- CUSTOM:** (empty)
- Limits:** CLEANUPPOLICY: compact

At the bottom of the dialog are "Advanced", "Cancel", and "Save" buttons.

Add Topic
User friendly UI to create new topics. Simple and Advance features are available

Topic Update

The screenshot shows the Apache Kafka Metrics UI interface. On the left, there's a sidebar with various icons. The main area displays metrics for the cluster: Total Bytes In (64MB), Total Bytes Out (2MB), Produced Per Sec (243), Fetched Per Sec (705), Under Replicated (0), and Offline Partitions (0). Below this, a list of topics (35) is shown, with two topics selected: 'gdeleon-test' and 'testKnoxSetUp'. A green callout box points to the search bar at the top, containing the text: "Search Topic Search the Topic you would like to update. Then click on profile". Another green callout box points to the configuration section for 'testKnoxSetUp', containing the text: "Update Topic Click on Config and you can change Cleanup Policy or click Advanced to modify the configuration parameters." The configuration section includes tabs for METRICS, DATA EXPLORER, and CONFIGS. The CONFIGS tab is active, showing detailed configuration parameters for the topic.

Topics

Total Bytes In 64MB Total Bytes Out 2MB Produced Per Sec 243 Fetched Per Sec 705 Under Replicated 0 Offline Partitions 0

Topics (35)

NAME	DATA IN	DATA OUT	MESSAGES IN	CONSUMER GROUPS
gdeleon-test	0B	0B	0	0
testKnoxSetUp	0B	0B	-1	0

Search Topic
Search the Topic you would like to update. Then click on profile

Topics / testKnoxSetUp

METRICS DATA EXPLORER CONFIGS

TOPIC NAME testKnoxSetUp PARTITIONS 1

REPLICATION FACTOR 1

CLEANUPPOLICY compact,delete

MAX.MESSAGE.BYTES 1000000

RETENTION BYTES -1

RETENTION HOURS 168

COMPRESSION.TYPE producer

DELETE.RETENTION.HOURS 24

FILE.DELETE.DELAY.MS 40000

FLUSH.POLL.S 922337203685477507

FLUSH.MS 9223372036854775807

INDEX.INTERVAL.BYTES 1000000

simple Save

Name	Value
compression.type	producer
min.insync.replicas	1
segment.jitter.ms	0
cleanup.policy	delete
flush.ms	922337203685477507
segment.bytes	1073741824
retention.hours	168
flush.messages	922337203685477507
message.format.version	2.0-IV1
file.delete.delay.ms	60000
max.message.bytes	1000000
min.compaction.lag.ms	0
message.timestamp.type	CreateTime
preallocate	false
min.cleanable.dirty.ratio	0.5
index.interval.bytes	4096
unclean.leader.election.enable	false
retention.bytes	-1
delete.retention.hours	24
segment.ms	604800000
message.timestamp.difference.max.ms	922337203685477507
segment.index.bytes	10485760

Alerting – Create Notifier

Notifier

NAME
email_notifier

DESCRIPTION
Notifies via Email

PROVIDER
Email

FROM ADDRESS
smm.barcelona1@gmail.com

TO ADDRESS
smm.barcelona1@gmail.com

USERNAME
smm.barcelona1@gmail.com

1st Key Construct of Alerts

Alert Notifier

1. Email
2. http end point
3. Kafka topic

PASSWORD
.....

SMTP HOSTNAME
smtp.gmail.com

SMTP PORT
587

ENABLE AUTH
 ENABLE SSL ENABLE STARTTLS

PROTOCOL
smtp

ENABLE DEBUG

NOTIFIER RATE LIMIT

COUNT	DURATION
2	HOUR

Alerting – Create Alert

Alert Policy

NAME
High Lag - Kafka Streams Truck Join Micro Service

DESCRIPTION
High Lag - Kafka Streams Truck joining the speed and geo streams

EXECUTION INTERVAL IN SECONDS
60

EXECUTION DELAY IN SECONDS
300

ENABLE

Policy

COMPONENT TYPE
IF... Consumer

TARGET NAME
nifi-truck-sensors-west

+

ATTRIBUTE	CONDITION	VALUE
CONSUMER GROUP LAG	<	50

+

Action

NOTIFICATION
email_notifier

Preview

IF CONSUMER: nifi-truck-sensors-west has CONSUMER GROUP LAG > 50 THEN notify by email_notifier

Cancel Save

2nd Key Construct of Alerts

Alert Policy

1. Defined for 6 key entities (cluster, broker, topic, producer, consumer, latency, cluster replication)
2. Metrics defined on entities
3. Complex alerts
4. Includes notifier when triggered

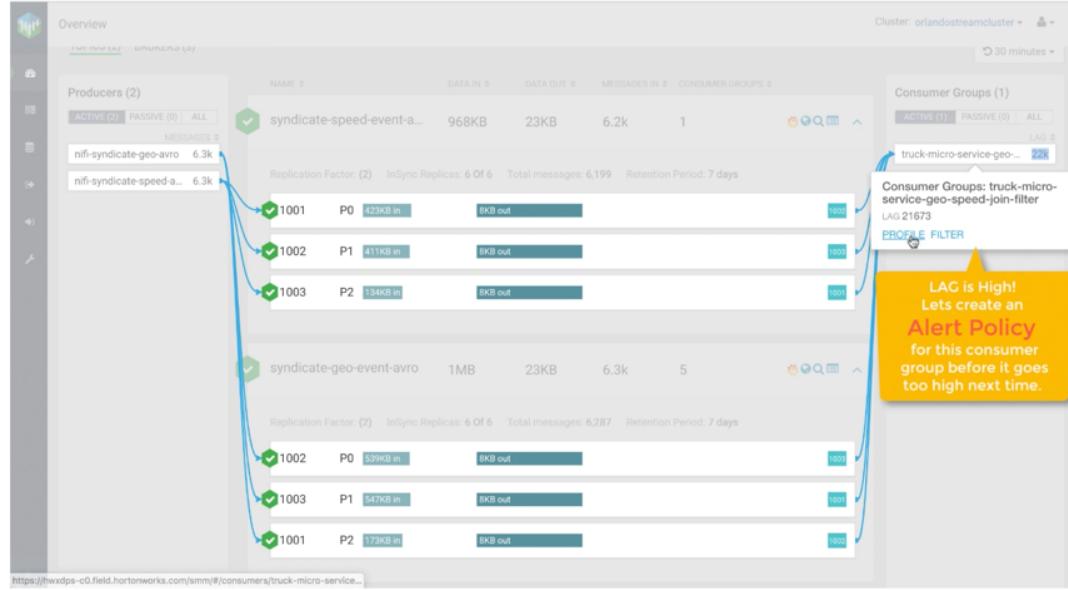
Alert History

HISTORY	ALERT POLICIES	NOTIFIERS			
Title	Timestamp	Component name	Type	State	Payload
Alert: Lag_Alert	2m 32s ago	load-optimizer-micro-service	CONSUMER	RAISED	Alert policy : 'Lag_Alert' For CONSUMER=load
Alert: Lag_Alert	7m 32s ago	load-optimizer-micro-service	CONSUMER	RAISED	Alert policy : 'Lag_Alert' For CONSUMER=load

Disable Alert

HISTORY	ALERT POLICIES	NOTIFIERS	ADD NEW
NAME	CONDITION	DESCRIPTION	ENABLE
gdeleon-alert	IF Topic: gdeleon-test has BYTES IN PER SEC >= 10	my alert	
Lag_Alert	IF Consumer: load-optimizer-micro-service has CONSUMER GROUP LAG >= 10		

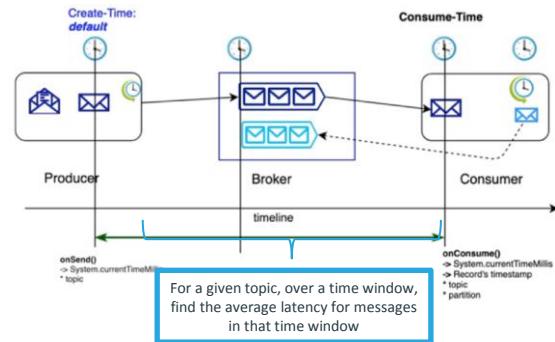
Example: Alerting on Micro-Service Consumer Group with High Lag



New DevOps Monitoring Capability with End to End Latency View

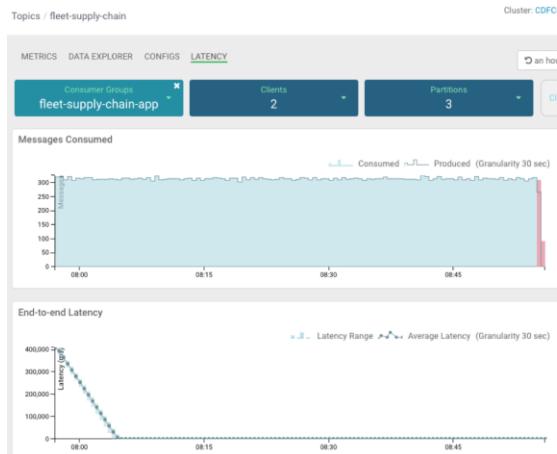
Problem Statement / Requirements

- Kafka DevOps teams need the ability to see an end to end latency view of messages produced and consumed across producers to topics/brokers to consumers
- Common DevOps questions include:
 - For the last five minutes, what is the average latency of messages consumed after being produced.
 - For the last hour, which topics have under consumption?



Solution / Benefit

- SMM 2.0 provides new monitoring view for end to end Latency
- End to End Latency view is powered by new embedded Kafka streams application that calculates end to end latencies and new interceptors that can be used to instrument producer and consumer clients.



Select a Topic of Interest

Overview Cluster: hdf_mpack ▾ ⌂

Producers 100 Brokers 3 Topics 13 Consumer Groups 2 Clear

TOPICS (13) BROKERS (3) 30 minutes ▾

Producers (100)

ACTIVE (100)	PASSIVE (0)	ALL (100)
MESSAGES ▾		
8-5-producer-23	24k	
8-5-producer-99	24k	
8-5-producer-37	24k	
8-5-producer-0	24k	
8-5-producer-45	24k	
8-5-producer-63	24k	
8-5-producer-96	24k	
8-5-producer-39	23k	
8-5-producer-20	23k	
8-5-producer-14	23k	
8-5-producer-81	23k	
8-5-producer-85	23k	

NAME ▾ DATA IN ▾ DATA OUT ▾ MESSAGES IN ▾ CONSUMER GROUPS ▾

TruckEvents2 616B 2KB 1.6k 0 🔍🔍🔍

Replication Factor: (2) InSync Replicas: 8 Of 8 Total messages: 1,620 Retention Period: 0 secs

✓ 2 P0 2KB in 2KB out 0
✓ 0 P1 2KB in 1KB out 1
✓ 1 P2 2KB in 2KB out 2
✓ 2 P3 2KB in 1KB out 1

TruckEvents 1KB 928B 997 0 🔍🔍🔍

Consumer Groups (2)

ACTIVE (0)	PASSIVE (2)	ALL (2)
LAG ▾		
ExampleTestGroup2	0	
ExampleTestGroup	0	

Select the End to End Latency Tab



SMM Atlas Integration

Explore Metadata about the Topic in Atlas

Topics / gateway-west-raw-sensors

METRICS DATA EXPLORER CONFIGS

Producers (3)

Replication Factor: (2) InSync Replicas: 8 Of 8 Total messages: 805 Retention Period: 7 days

	MESSAGES
minifi-truck-w1	298
minifi-truck-w3	230
minifi-truck-w2	263



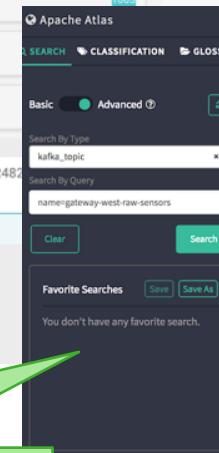
Summary

Number of Replicas	2
Number of Partitions	4
Total number of Brokers for Topic	5
Preferred Replication %	100
Under Replicated %	0

Atlas Integration is not available on the workshop cluster.

Cluster: orlandostreamcluster

Click on Atlas Link to see the metadata of the topic gateway-west-raw-sensors in Atlas



Key	Value
avgMessageSizeInBytes	0
avroSchema	
contactInfo	
description	
desiredRetentionInhrs	0
keyClassName	
maxThroughputPerSec	0
name	gateway-west-raw-sensors
numberOfEventsPerDay	0
owner	
partitionCount	0
partitionCountLocal	0
partitionCountNational	0
qualifiedName	gateway-west-raw-sensors@orlandostreamcluster

If Atlas does not come up then use this link and then pick Type as Kafka_Topic for search
<https://99.80.132.89:8443/pkuc-wv-smm-m/dp-proxy/atlas/>

Traverse the flow of data across multiple Kafka Topics using SMM and Atlas Integration

Topics / gateway-west-raw-sensors

METRICS DATA EXPLORER CONFIGS

Producers (3)

Replication Factor: (2) InSync Replicas: 8 Of 8 Total messages: 805 Retention Period

	MESSAGES
minifi-truck-w1	298
minifi-truck-w3	230
minifi-truck-w2	263

Summary

Number of Replicas
Number of Partitions
Total number of Brokers for Topic
Preferred Replication %
Under Replicated %

Question

The topic has one active consumer which is a NiFi consumer. Which Kafka topic if any is this NiFi Flow consumer publishing events to?

Cluster: orlando-streamcluster

Consumer Groups (1)

nifi-truck-sensors-west LAG 2

Step 1

Click on Atlas Icon to see lineage of the the topic gateway-west-raw-sensors

Apache Atlas

Basic Advanced

Search By Type: kafka_topic

Search By Query: name=gateway-west-raw-sensors

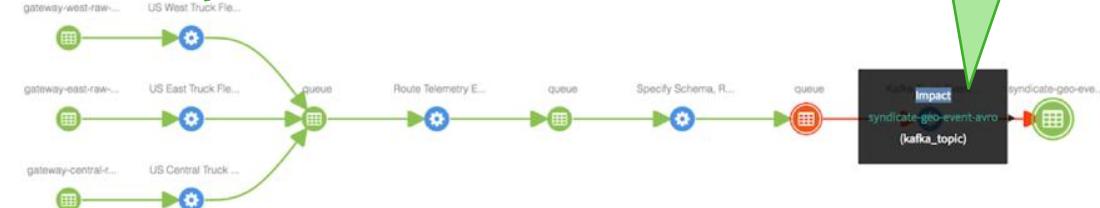
Favorite Searches: Save Save As

You don't have any favorite search.

Atlas Integration is not available on the workshop cluster.

gateway-west-raw-sensors (kafka_topic)

Classifications: +
Term: +
Properties Lineage Relationships Classifications Audits



Analysis

NiFi App consumes from the gateway-west-raw-sensors topic and publishes events to downstream Kafka topic called syndicate-geo-event-avro

SCHEMA REGISTRY

WHAT VALUE DOES SCHEMA REGISTRY PROVIDE?



- Data Governance
 - Centralized registry to provide reusable schema
 - Version management to define relationship between schemas
 - Validation to enable generic format conversion and generic routing
- Operational Efficiency
 - Centralized registry to avoid attaching schema to every piece of data
 - Version management to enable consumers and producers can evolve at different rates
 - Validation to ensure data quality

SCHEMA REGISTRY SUPPORT FOR DIFFERENT “STATES”

Enable, disable, archive

The screenshot shows the Schema Registry interface with two main sections:

test-6 BACKWARD (Left Panel):

- TYPE:** avro
- GROUP:** test-6
- BRANCH:** 1 ↘ 0
- SERIALIZER & DESERIALIZER:**

Branch: MASTER

Branch Description: 'MASTER' branch for schema metadata 'test-6'

Version Description: test

Schema (Version 1):

```
1 {  
2   "type": "record",  
3   "namespace": "hortonworks.hdp.refapp.truckin",  
4   "name": "truckgeoeventkafka",  
5   "fields": [  
6     {  
7       "name": "eventTime",  
8       "type": "string"  
9     },  
10    {  
11      "name": "eventTimeLong",  
12      "type": "long",  
13      "default": 0  
14    },  
15  ]  
}
```

test-5 BACKWARD (Right Panel):

- TYPE:** avro
- GROUP:** test-5
- BRANCH:** 1 ↘ 0
- SERIALIZER & DESERIALIZER:**

Branch: MASTER

Branch Description: 'MASTER' branch for schema metadata 'test-5'

Version Description: adding eventtime long

Schema (Version 2):

```
1 {  
2   "type": "record",  
3   "namespace": "hortonworks.hdp.refapp.truckin",  
4   "name": "truckgeoeventkafka",  
5   "fields": [  
6     {  
7       "name": "eventTime",  
8       "type": "string"  
9     },  
10    {  
11      "name": "eventTimeLong",  
12      "type": "long",  
13      "default": 0  
14    },  
15  ]  
}
```

Archived Versions:

- v2 27m 5s ago Archived
- v1 27m 42s ago Enabled

Compare Versions:

COMPARING DIFFERENT SCHEMA VERSIONS

Compare Schema Versions

BASE VERSION: 1 NEW VERSION: 2

DESCRIPTION: test-2 DESCRIPTION: adding eventtime long

V 1 vs. V 2

Unified	Split
1 1 {	
2 2 "type" : "record",	
3 3 "namespace" : "hortonworks.hdp.refapp.trucking",	
4 4 "name" : "truckgeoeventkafka",	
5 5 "fields" : [
6 6 { "name" : "eventTime" , "type" : "string" },	
7 7 { "name" : "eventTimeLong" , "type" : "long" , "default": 0 },	
8 8 { "name" : "eventSource" , "type" : "string" },	
9 9 { "name" : "truckId" , "type" : "int" },	
10 10 { "name" : "driverId" , "type" : "int"},	
11 11 { "name" : "driverName" , "type" : "string"},	

Close

TYPE: AVRO GROUP: test-2 BRANCH: 1 SERIALIZER & DESERIALIZER: 0

BRANCH: MASTER
CHANGE LOG

v2 54m 29s ago Enabled

v1 56m 32s ago Enabled

COMPARE VERSIONS

```
record,
": "hortonworks.hdp.refapp.trucking",
truckgeoeventkafka",
[

"eventTime",
"string"

"eventTimeLong",
"long",
": 0
```

LIFECYCLE ACTION 1 - ACTION

Fork Schema Version to Branch called Dev

The screenshot shows the Cloudera Schema Registry interface for a schema named 'raw-truck_events_avro'. The schema type is 'avro', group is 'truck-sen...', and there is 1 branch with 0 versions. The current branch is 'MASTER'. A modal dialog titled 'Fork a New Schema Branch' is open, prompting for a 'NAME*' (set to 'Dev') and a 'DESCRIPTION*' (set to 'Dev Fork'). The background shows the schema definition code:

```
1 {  
2   "type": "record",  
3   "namespace": "hortonworks.hdp.refapp.trucking",  
4   "name": "truckgeoevent",  
5   "fields": [  
6     {  
7       "name": "eventTime",  
8       "type": "string"  
9     },  
10    {  
11      "name": "eventSource",  
12      "type": "string"  
13    },  
14    {  
15      "name": "truckId"  
16    }  
17  ]  
18}  
19
```

Branch: MASTER
Version 1 (Forked 2d 17h 1m ago)

NAME*: Dev
DESCRIPTION*: Dev Fork

CANCEL SAVE

LIFECYCLE ACTION 2 - ACTION

Edit Version

DESCRIPTION *

adding eventtime long field

SCHEMA TEXT *

```
1 {
2   "type" : "record",
3   "namespace" : "hortonworks.hdp.refapp.trucking",
4   "name" : "truckgeoevent",
5   "fields" : [
6     { "name" : "eventTime" , "type" : "string" },
7     { "name" : "eventTimeLong" , "type" : "long" },
8     { "name" : "eventSource" , "type" : "string"
9     { "name" : "truckId" , "type" : "int" },
10    { "name" : "driverId" , "type" : "int" },
11    { "name" : "driverName" , "type" : "string" },
12    { "name" : "routeId" , "type" : "int" },
13    { "name" : "route" , "type" : "string" },
14    { "name" : "eventType" , "type" : "string" },
15    { "name" : "latitude" , "type" : "double" },
16    { "name" : "longitude" , "type" : "double" },
17    { "name" : "correlationId" , "type" : "long" }
18  ]
19 }
```

VALIDATE | CLEAR

raw-truck_events_avro
BACKWARD

TYPE: avro GROUP: truck-sen... BRANCH: 2 SERIALIZER & DESERIALIZER: 0

CANCEL SAVE

Edit the Forked Version, State: "INITIATED"

raw-truck_events_avro BACKWARD

TYPE: avro GROUP: truck-sen... BRANCH: 2 SERIALIZER & DESERIALIZER: 0

BRANCH: Dev

VERSION 2

BRANCH DESCRIPTION: dev

VERSION DESCRIPTION: adding eventtime long field

```
1 {
2   "type": "record",
3   "namespace": "hortonworks.hdp.refapp.trucking",
4   "name": "truckgeoevent",
5   "fields": [
6     {
7       "name": "eventTime",
8       "type": "string"
9     },
10    {
11      "name": "eventTimeLong",
12      "type": "long",
13      "default": 0
14    },
15  ]
16 }
```

BRANCH: Dev

CHANGE LOG

v2 0s ago INITIATED

v1 2d 17h 25m 15s ago Enabled

COMPARE VERSIONS

LIFECYCLE ACTION 3 - ACTION

Start Review, State: "InReview"

The screenshot shows two schema definitions for 'raw-truck_events_avro' in Cloudera Manager.

Top Schema (Version 2):

- Branch:** Dev
- Branch Description:** dev
- Version Description:** adding eventtime long field
- Type:** avro
- Group:** truck-sen...
- Branch:** 2 (with 0 changes)
- Serializer & Deserializer:** 0

```
1 {
2   "type": "record",
3   "namespace": "hortonworks.hdp.refapp.trucking",
4   "name": "truckgeoevent",
5   "fields": [
6     {
7       "name": "eventTime",
8       "type": "string"
9     },
10    {
11      "name": "eventTimeLong",
12      "type": "long",
13      "default": 0
14    }
15  ]
```

Bottom Schema (Version 1):

- Branch:** Dev
- Branch Description:** dev
- Version Description:** adding eventtime long field
- Type:** avro
- Group:** truck-sen...
- Branch:** 2 (with 0 changes)
- Serializer & Deserializer:** 0

```
1 {
2   "type": "record",
3   "namespace": "hortonworks.hdp.refapp.trucking",
4   "name": "truckgeoevent",
5   "fields": [
6     {
7       "name": "eventTime",
8       "type": "string"
9     }
10  ]
```

Change Log (Version 2):

- v2: 41s ago (INITIATED)
- v1: 2d 17h 26m 26s ago (Enabled)

Compare Versions: 1m 11s ago (InReview)

LIFECYCLE ACTION 4 - ACTION

Finish Review, State: "Reviewed"

The screenshot shows the Cloudera Manager interface for managing schema versions. At the top, there's a summary for the schema 'raw-truck_events_avro' (BACKWARD), which is of type 'avro', part of group 'truck-sen...', has 2 branches, and 0 serializers/deserializers.

Version 2 (Active):

- Branch: Dev
- Branch Description: dev
- Version Description: adding eventtime long field
- Code Snippet:

```
1 {
2   "type": "record",
3   "namespace": "hortonworks.hdp.refapp.trucking",
4   "name": "truckgeoevent",
5   "fields": [
6     {
7       "name": "eventTime",
8       "type": "string"
9     },
10    {
11      "name": "eventTimeLong",
12      "type": "long",
13      "default": 0
14    }
15 }
```
- BRANCH: Dev (with a red exclamation mark)
- CHANGE LOG

Version 1 (Historical):

- Branch: Dev
- Branch Description: dev
- Version Description: adding eventtime long field
- Code Snippet:

```
1 {
2   "type": "record",
3   "namespace": "hortonworks.hdp.refapp.trucking",
4   "name": "truckgeoevent",
5   "fields": [
6     {
7       "name": "eventTime",
8       "type": "string"
9     }
10    {
11      "name": "eventTimeLong",
12      "type": "long",
13      "default": 0
14    }
15 }
```
- BRANCH: Dev (with a red exclamation mark)
- CHANGE LOG
- v2 2m 55s ago Reviewed (with a blue pencil icon)
- v1 2d 17h 28m 10s ago Enabled
- COMPARE VERSIONS

LIFECYCLE ACTION 5 - ACTION: ENABLE, STATE: "ENABLED"

raw-truck_events_avro
BACKWARD

TYPE: avro
GROUP: truck-sen...
BRANCH: 2
SERIALIZER & DESERIALIZER: 0

BRANCH: Dev
CHANGE LOG

raw-truck_events_avro
BACKWARD

TYPE: avro
GROUP: truck-sen...
BRANCH: 2
SERIALIZER & DESERIALIZER: 0

BRANCH: Dev
CHANGE LOG

BRANCH: Dev
VERSION 2

raw-truck_events_avro
BACKWARD

TYPE: avro
GROUP: truck-sen...
BRANCH: 2
SERIALIZER & DESERIALIZER: 0

BRANCH: Dev
CHANGE LOG

3m 40s ago
Enabled

2d 17h 28m 55s ago
Enabled

COMPUTE VERSIONS

```
1 {
2 "type": "record",
3 "namespace": "hortonworks.hdp.refapp.trucking",
4 "name": "truckgeoevent",
5 "fields": [
6   {
7     "name": "eventTime",
8     "type": "string"
9   },
10  {
11    "name": "eventTimeLong",
12    "type": "long",
13    "default": 0
14  },
15 }
```

LIFECYCLE ACTION 6 - ACTION: MERGE BACK TO MASTER

The screenshot shows a schema editor interface for a file named `raw-truck_events_avro`. The file is set to **BACKWARD** compatibility. It has a **TYPE avro**, belongs to the **GROUP truck-sen...** group, and is part of **BRANCH 2** with **0** commits. The **SERIALIZER & DESERIALIZER** section is collapsed.

BRANCH : Dev

BRANCH DESCRIPTION : dev

VERSION DESCRIPTION : adding eventtime long field

The code editor displays the following Avro schema:

```
1 {  
2   "type": "record",  
3   "namespace": "hortonworks.hdp.refapp.trucking",  
4   "name": "truckgeoevent",  
5   "fields": [  
6     {  
7       "name": "eventTime",  
8       "type": "string"  
9     },  
10    {  
11      "name": "eventTimeLong",  
12    }  
13  ]  
14}  
15
```

The version is labeled **VERSION 2** with a **MERGE** button. To the right, the **CHANGE LOG** shows two entries:

- v2 3m 40s ago Enabled **Merge**
- v1 2d 17h 28m 55s ago Enabled

A **COMPARE VERSIONS** button is also present.

A modal dialog box in the foreground asks: **Are you sure you want to merge this branch?** with **No** and **Yes** buttons.

COMPLETION OF SCHEMA LIFECYCLE

Merged Schema from Dev Branch to Master

raw-truck_events_avro
BACKWARD

TYPE: avro GROUP: truck-sen... BRANCH: 2 🌐 0 SERIALIZER & DESERIALIZER

BRANCH: MASTER

VERSION 3

BRANCH: MASTER CHANGE LOG

v3 0s ago Enabled

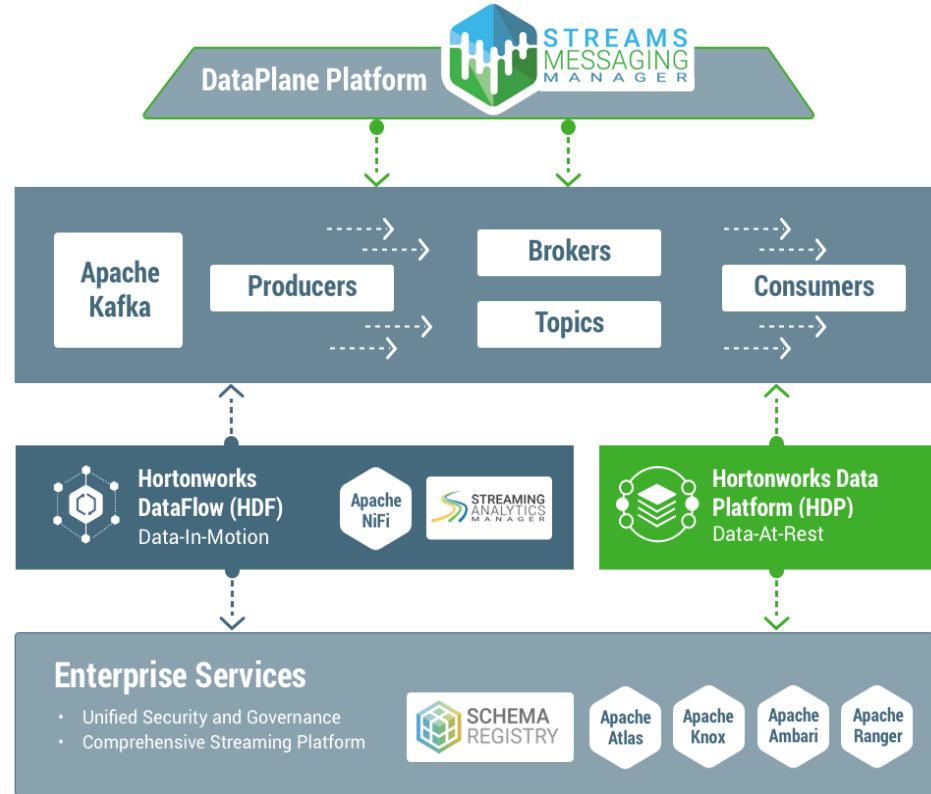
Schema Branches

```
1 {  
2   "type": "record",  
3   "namespace": "hortonworks.hdp.refapp.trucking",  
4   "name": "truckgeoevent",  
5   "fields": [  
6     {  
7       "name": "eventTime",  
8       "type": "string"  
9     },  
10    {  
11      "name": "eventTimeLong",  
12      "type": "long",  
13      "default": 0  
14    },  
15  ]
```

APACHE KAFKA TOPIC CREATION

Streams Messaging Manager (SMM)

- Open Source tool to Cure the “Kafka Blindness”
- Single Monitoring Dashboard for all your Kafka Clusters across 4 entities
- Notification on available metrics
- Designed for the Enterprise
 - Support for Secure Kafka cluster
 - Rich Access Control Policies (ACLS)
 - Supports multiple Kafka Clusters
- REST as a First Class Citizen





CRUD

Total Bytes In
1,234GBTotal Bytes Out
12,345GBProduced Per Sec
17.7kBFetched Per Sec
11.6kBIn Sync Replicas
192Out of Sync
0Under Replicated
3Offline
17

Topics (46)



Last Hour

ADD

NAME	MESSAGES
Alternate Metrics	ACTIVE
Baracuda	ACTIVE
BarnFields	ACTIVE
BulkMetric	ACTIVE
BulkLoad	ACTIVE
Producers (1)	Replication Factor: 2
P1 1001	Adrastea 234
P2 1003	MESSAGES
P3 1004	

Add Topic

TOPIC NAME *

Production001023

PARTITIONS

3

Availability



Maximum

REPLICATION FACTOR 3
MIN. INSYNC REPLICAS 2

High

REPLICATION FACTOR 3
MIN. INSYNC REPLICAS 1

Moderate

REPLICATION FACTOR 2
MIN. INSYNC REPLICAS 1

Low

REPLICATION FACTOR 1
MIN. INSYNC REPLICAS 1

Limits

CLEANUP POLICY *

Delete

RETAIN FOR

2

Weeks

MAX TOPIC SIZE

10 GB

MAX MESSAGE SIZE

25 MB

ADVANCED

CANCEL

SAVE

REPLICATION 2

PARTITIONS 3

Add Topic

TOPIC NAME * Production001023 PARTITIONS 3

MIN.INSYNC.REPLICAS 2

REPLICATION.FACTOR 3

CLEANUP.POLICY delete

MAX.MESSAGE.BYTES 10000011

RETENTION.BYTES 100000000000

RETENTION.MS 43200000

COMPRESSION.TYPE producer

SIMPLE CANCEL SAVE

Create a Kafka Topic

The screenshot shows the Cloudera Manager interface for managing Kafka topics. On the left, there's a sidebar with various icons. The main area displays statistics like 'Total Bytes In' (129 KB) and 'Total Bytes Out' (153 KB). Below this is a list of 'Topics (15)' with checkboxes next to them. A modal window titled 'Add Topic' is open in the center. It contains fields for 'TOPIC NAME' (cleandata) and 'PARTITIONS' (1). Below these are five availability levels: 'MAXIMUM' (green), 'HIGH' (green), 'MODERATE' (yellow), 'LOW' (gray), and 'CUSTOM' (gray). Under 'REPLICATION', it shows 'FACTOR 2' with 'MIN INSTYNC' and 'REPLICA 2'. The 'CLEANUP POLICY' dropdown is set to 'compact'. At the bottom of the modal are 'Advanced', 'Cancel', and 'Save' buttons. The background shows a summary of consumer groups with counts of 'Under Replicated' (0) and 'Offline Partitions' (0).

Create a Kafka Topic

Add Topic

TOPIC NAME cleandata PARTITIONS 1

Availability

- MAXIMUM
- HIGH
- MODERATE
- LOW
- CUSTOM

REPLICATION FACTOR 3
MIN INSYNC REPLICA 2

REPLICATION FACTOR 3
MIN INSYNC REPLICA 1

REPLICATION FACTOR 2
MIN INSYNC REPLICA 1

REPLICATION FACTOR 1
MIN INSYNC REPLICA 1

Limits

CLEANUP.POLICY

- Select...
- compact
- delete
- compact,delete

Advanced Cancel Save

0

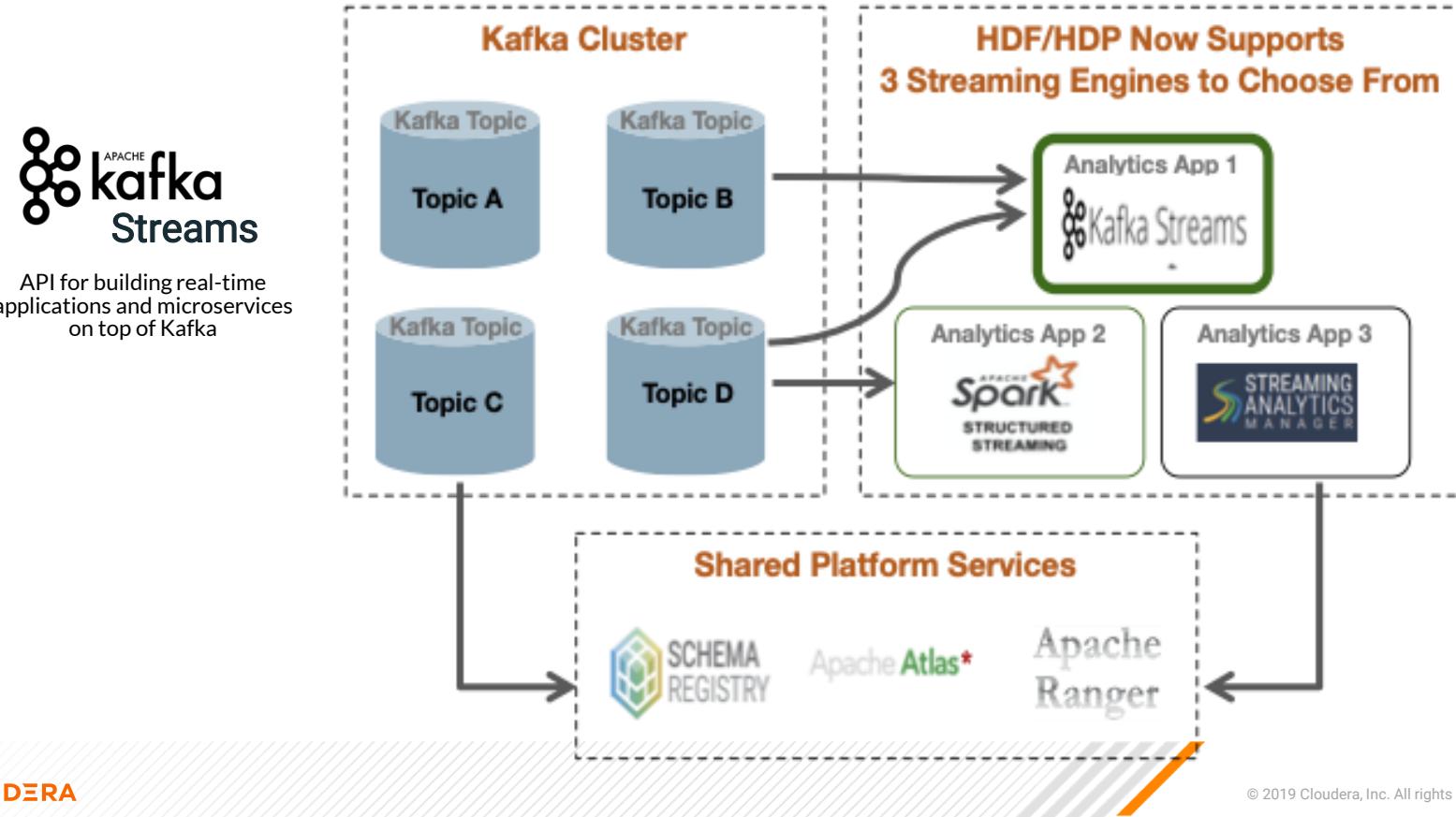
- Topic Name
- Partitions (1 or more)
- Pick a Template with replicas and replication factor
- Carefully select a Cleanup Policy **DELETE**
 - (your messages need a key)

Create Kafka Topics For Kafka Streams Application

- Topic Name: **streams-plaintext-input**
 - Partitions: 1
 - Replicas: 1
 - Delete
- Topic Name: **streams-wordcount-output**
 - Partitions: 1
 - Replicas: 1
 - Delete

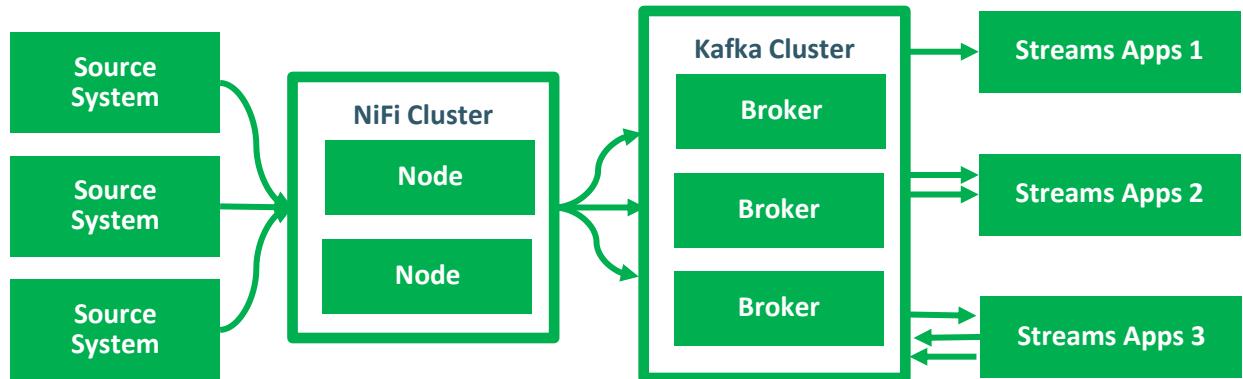
KAFKA STREAMS

Secure and Governed Microservices with Kafka Streams Support



Kafka Streams

- A client library for applications and microservices on top of Kafka
- Written in Java or Scala
- Elastic, highly scalable, fault-tolerant
- Supports At-Least-Once and Exactly-once semantics inside Kafka
- Deploy to containers, VMs, bare metal, cloud, K8, devices.



Kafka Streams Library

Stream Processing Style	Event a time
Delivery Guarantee	Exactly Once (within Kafka)
State Management	Implicit support with RocksDB
Fault Tolerance	Implicit support with internal Kafka topics / ZK
Advanced feature: Joins/ Aggregations/ Windowing	Stream/Stream joins, Stream/Table joins, joins with only message key, no OOO aggregation processors
Advanced Feature: Watermarking (late arriving data)	Not fully supported
Latency	Low
Throughput	Medium
APIS	Simple APIs (Compositional)

Kafka Streams Library

SQL DSL	Nothing in Apache Kafka.
Lambda Architecture	No Supported
GUI Tooling / Code-Less Approach	Not Supported
Maturity	Relatively New
Use Cases	Lightweight eventing based microservices, ETL
Cluster Requirement	No

KAFKA STREAMS WALK THROUGH

Topic - Create

Add Topic

TOPIC NAME

PARTITIONS

Availability



MAXIMUM



HIGH



MODERATE



LOW



CUSTOM

REPLICATION

FACTOR 3

MIN INSYNC

REPLICA 2

REPLICATION

FACTOR 3

MIN INSYNC

REPLICA 1

REPLICATION

FACTOR 2

MIN INSYNC

REPLICA 1

REPLICATION

FACTOR 1

MIN INSYNC

REPLICA 1

Limits

CLEANUP.POLICY

[Advanced](#)[Cancel](#)[Save](#)

Topics							
Total Bytes In	Total Bytes Out	Produced Per Sec	Fetched Per Sec	In Sync Replicas	Out Of Sync	Under Replicated	Offline Partitions
88 MB	2 MB	212	169	196	0	0	0
Topics (42)							
NAME	DATA IN	DATA OUT	MESSAGES IN	CONSUMER GROUPS			
streams-plaintext-input	0B	0B	0	0	Stream	Edit	View
streams-wordcount-output	0B	0B	0	0	Stream	Edit	View

Setup Your Development Environment

sudo su

yum install maven curl wget unzip zip git -y

mkdir /opt/demo

chmod -R 777 /opt/demo

cd /opt/demo

Generate a Kafka Streams Project

```
mvn archetype:generate \
  -DarchetypeGroupId=org.apache.kafka \
  -DarchetypeArtifactId=streams-quickstart-java \
  -DarchetypeVersion=2.2.0 \
  -DgroupId=streams.examples \
  -DartifactId=streams.examples \
  -Dversion=0.1 \
  -Dpackage=myapps
```

Update Word Count Kafka Connection

Edit the file

/opt/demostreams.examples/src/main/java/myapps/**WordCount.java**

```
Properties props = new Properties();
props.put(StreamsConfig.APPLICATION_ID_CONFIG,
"streams-pipe");
props.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG,
"<YOURAWSINTERNALIPNAME>:9092");
```

Build a Kafka Streams Project

```
rm /opt/demostreams.examples/src/main/java/myapps/Pipe.java  
rm  
/opt/demostreams.examples/src/main/java/myapps/LineSplit.java  
mvn clean package
```

KAFKA TOPIC INTERACTION

Produce Kafka Messages

```
/opt/cloudera/parcels/CDH/lib/kafka/bin/kafka-console-producer.sh --broker-list  
<YOURAWSINTERNALIPNAME>:9092 --topic streams-plaintext-input
```

Instructions: Then type messages with a return.

Run Kafka Streams Java Application

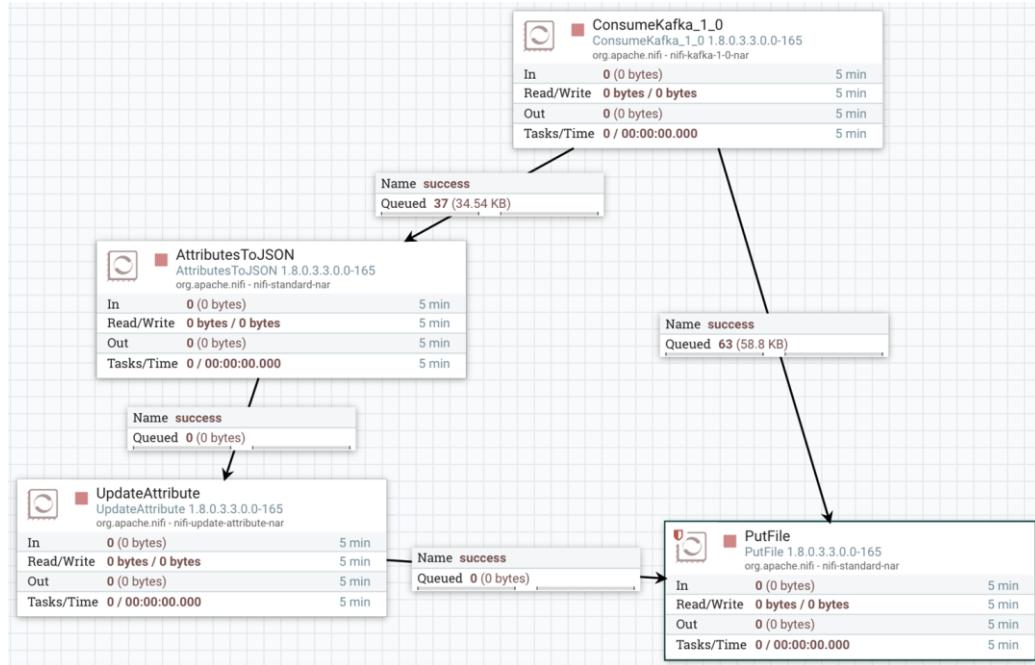
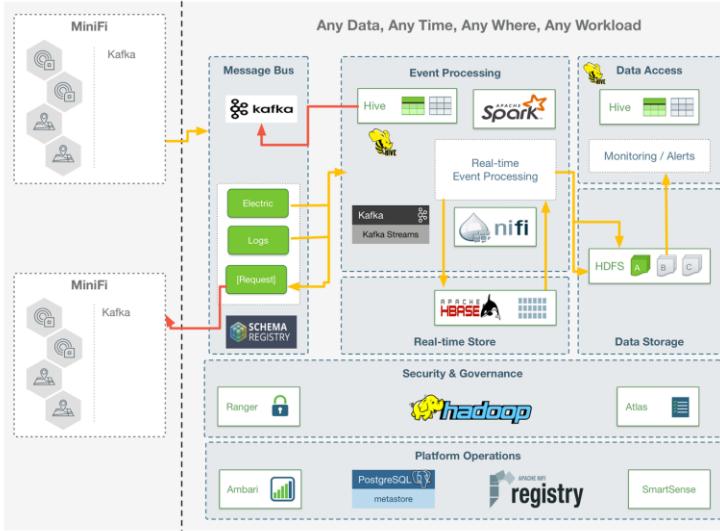
```
mvn exec:java -Dexec.mainClass=myapps.WordCount
```

Consume Kafka Messages

```
/opt/cloudera/parcels/CDH-6.3.0-1.cdh6.3.0.p0.1279813/lib/kafka/bin/kafka-console-consumer.sh --bootstrap-server <YOURAWSINTERNALIPNAME>:9092 \
--topic streams-wordcount-output \
--from-beginning \
--formatter kafka.tools.DefaultMessageFormatter \
--property print.key=true \
--property print.value=true \
--property key.deserializer=org.apache.kafka.common.serialization.StringDeserializer \
--property value.deserializer=org.apache.kafka.common.serialization.LongDeserializer
```

CONSUME AND PRODUCE KAFKA MESSAGES

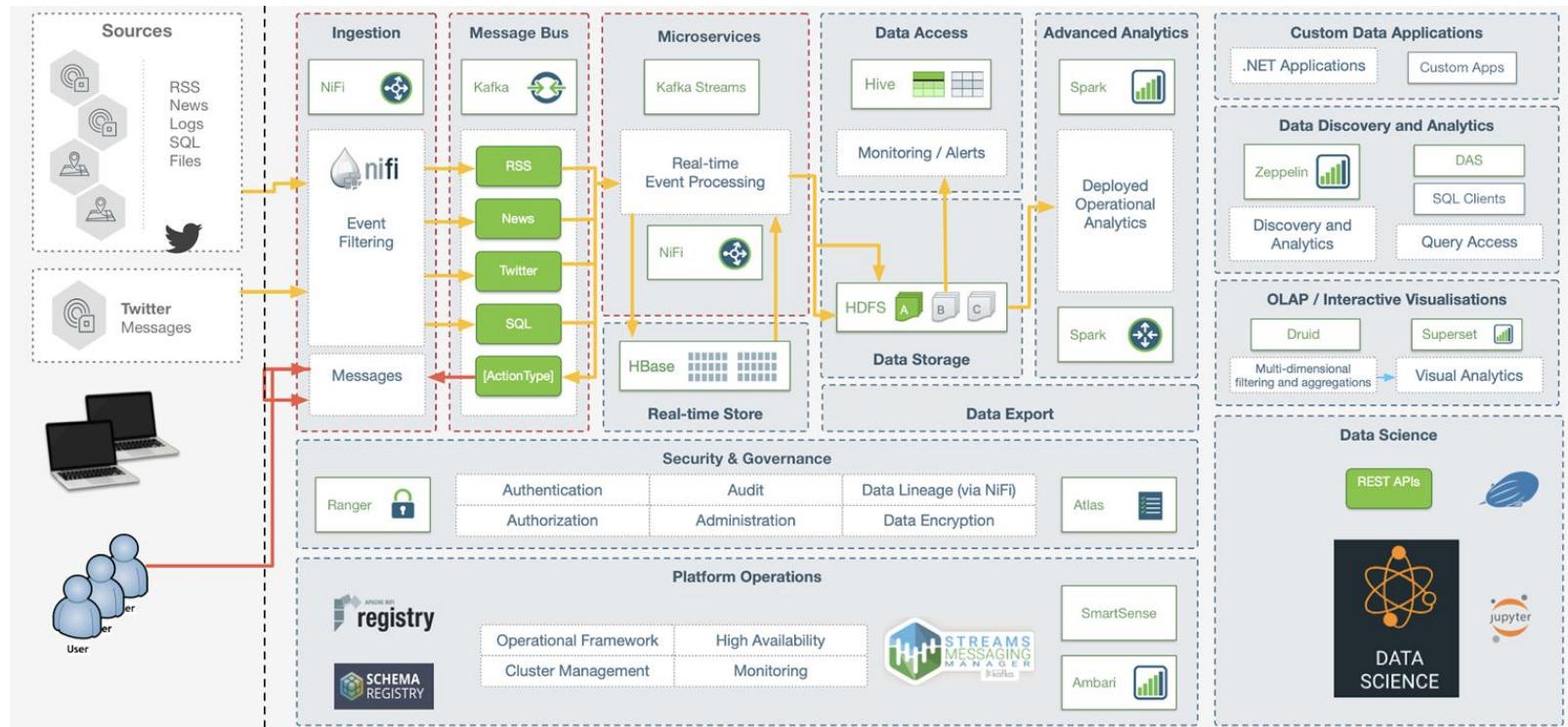
IoT Edge Use Cases with Apache Kafka and Apache NiFi - MiniFi



<https://community.cloudera.com/t5/Community-Articles/IoT-Edge-Use-Cases-with-Apache-Kafka-and-Apache-NiFi-MiniFi/ta-p/249232>

KAFKA STREAMS ADDITIONAL EXAMPLE

Kafka Streams Example Architecture



Kafka Streams

```
Timer timer = new Timer();

timer.schedule(new DisplayStatus(), 0, 120000);

final StreamsBuilder builder = new StreamsBuilder();
KStream<String, String> source = builder.stream("bme680");

source.foreach((key, value) -> processValues(key, value));
source.to("bme680out");

final Topology topology = builder.build();
final KafkaStreams streams = new KafkaStreams(topology, props);
final CountDownLatch latch = new CountDownLatch(1);
https://www.datainmotion.dev/2019/03/iot-series-sensors-utilizing-breakout\_28.html
```

Kafka Streams

```
Runtime.getRuntime().addShutdownHook(new Thread(STREAMS_SHUTDOWN_HOOK) {
    @Override
    public void run() {
        log.error(SHUTDOWN);
        streams.close();
        latch.countDown();
    }
}) ;

try {
    streams.start();
    latch.await();
} catch (Throwable e) {
    System.exit(1);
}
```

Kafka Streams Example 2

The screenshot shows an IDE interface with the following details:

- Project Structure:** The project is named "kstreams" and contains a "src" directory with "main" and "java" sub-directories. Under "java", there is a package "com.dataflowdeveloper.kstream" containing a class "BME680". The "resources" folder within "main" contains "log4j2.properties" and "log4j2_p.swp".
- Code Editor:** The file "BME680.java" is open, showing Java code for initializing MQTT connections. A yellow highlight covers the entire code block.
- Run Tab:** The "Run" tab shows the command being run: "/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/bin/java ...". The output shows SLF4J loading information and a warning about a static logger binder.

```
super();
initMQTT();

/**
 * initialize MQTT connections
 */
private void initMQTT() {
    String publisherId = UUID.randomUUID().toString();
    try {
        MqttClientPersistence persistence = new MemoryPersistence();
        publisher = new MqttClient(MQTT_BROKER, publisherId, persistence);
    } catch (MqttException e) {
        log.error(MQTT_FAILURE, e);
    }
    MqttConnectOptions options = new MqttConnectOptions();
    options.setAutomaticReconnect(true);
    options.setCleanSession(true);
    options.setConnectionTimeout(CONNECTION_TIMEOUT);
    try {
        this.publisher.connect(options);
    } catch (MqttException e) {
        log.error(MQTT_CONNECTION_FAILURE, e);
    }
}
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/bin/java ...
objc[80616]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/bin/java ...
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
```

Cloudera Managed Kafka

Cloudera Manager [Clusters](#) [Hosts](#) [Diagnostics](#) [Audits](#) [Charts](#) [Backup](#) [Administration](#) [Search](#) [Support](#) [admin](#)

nyc [KAFKA-2](#) [Actions](#) [24 hours preceding Sep 12, 7:54 PM UTC](#)

Status Instances Configuration Commands [Charts Library](#) Audits Quick Links [Hide Descriptions](#) [30m](#) [1h](#) [2h](#) [6h](#) [12h](#) [1d](#) [7d](#) [30d](#)

Status Page Charts

Topics [Filter](#)

- [_smm_consumer_metrics](#)
- [_smm_producer_metrics](#)
- [global-retail-pos](#)
- [global-retail-store](#) **global-retail-store**
- [global-retail-web](#)
- [heartbeats](#)
- [local-retail](#)
- [mm2-configs.paris.internal](#)
- [mm2-offset-syncs.paris.internal](#)
- [mm2-offsets.paris.internal](#)

[Events](#)

Total Bytes Received Across Kafka Broker Topics
The sum of the **Bytes Received** metric computed across all this entity's descendant Kafka Broker Topic entities.

bytes / second Every 10 minut...
Thu 12 06 AM 12 PM 06 PM
CD-KAFKA-plvdjnsn:global-retail-store, total_kafka_b... 0

Total Bytes Fetched Across Kafka Broker Topics
The sum of the **Bytes Fetched** metric computed across all this entity's descendant Kafka Broker Topic entities.

bytes / second Every 10 minut...
Thu 12 06 AM 12 PM 06 PM
CD-KAFKA-plvdjnsn:global-retail-store, total_kafka_b... 0

Total Bytes Rejected Across Kafka Broker Topics
The sum of the **Bytes Rejected** metric computed across all this entity's descendant Kafka Broker Topic entities.

bytes / second Every 10 minut...
Thu 12 06 AM 12 PM 06 PM
CD-KAFKA-plvdjnsn:global-retail-store, total_kafka_b... 0

Total Messages Received Across Kafka Broker Topics
The sum of the **Messages Received** metric computed across all this entity's descendant Kafka Broker Topic entities.

messages / se... Every 10 minut...
Thu 12 06 AM 12 PM 06 PM
CD-KAFKA-plvdjnsn:global-retail-store, total_kafka_b... 0

Total Rejected Message Batches Across Kafka Broker Topics
The sum of the **Rejected Message Batches** metric computed across all this entity's descendant Kafka Broker Topic entities.

message_batc... Every 10 minut...
Thu 12 06 AM 12 PM 06 PM
CD-KAFKA-plvdjnsn:global-retail-store, total_kafka_r... 0

Total Fetch Request Failures Across Kafka Broker Topics
The sum of the **Fetch Request Failures** metric computed across all this entity's descendant Kafka Broker Topic entities.

fetch_requ... Every 10 minut...
Thu 12 06 AM 12 PM 06 PM
CD-KAFKA-plvdjnsn:global-retail-store, total_kafka_f... 0

[Feedback](#)

CLOUDERA

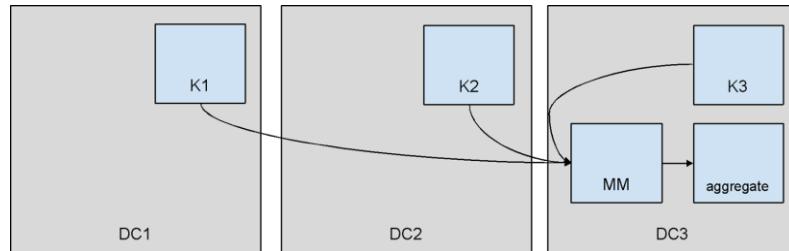
Kafka Replication

Kafka Replication Use Cases

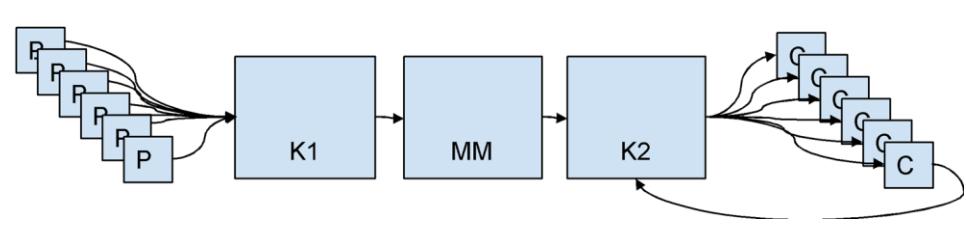
Disaster Recovery <p>In an event of a partial or complete datacenter disaster, providing failover/failback to a secondary cluster in a different region / DC</p>	Geo-Locality <p>Active-active geo-localized deployments allows users to access a near-by data center to optimize their architecture for low latency and high performance.</p>	Data Movement / Deployment <p>Use Kafka to synchronize data between on-prem applications and cloud deployments</p>
Centralized Analytics <p>Aggregate data from multiple Kafka clusters into one location for organization-wide analytics</p>	Workload Isolation <p>Creation of different envs for SDLC: Dev, Test, Prod. Clusters for specific use case cases (ETL, ingestion, analytics, etc)</p>	Legal / Compliance <p>Since different regions have different data storage and security requirements, clusters need to be created in region but data still needs to be shared.</p>

Various Replication Deployments Based on Use Cases

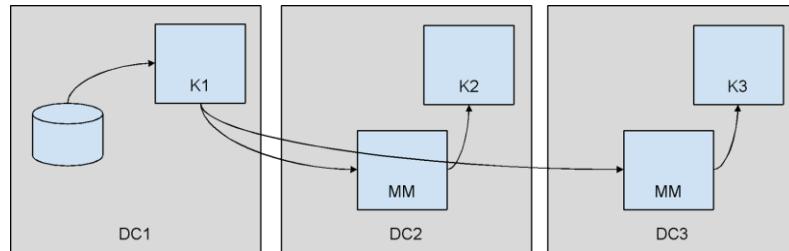
Aggregation



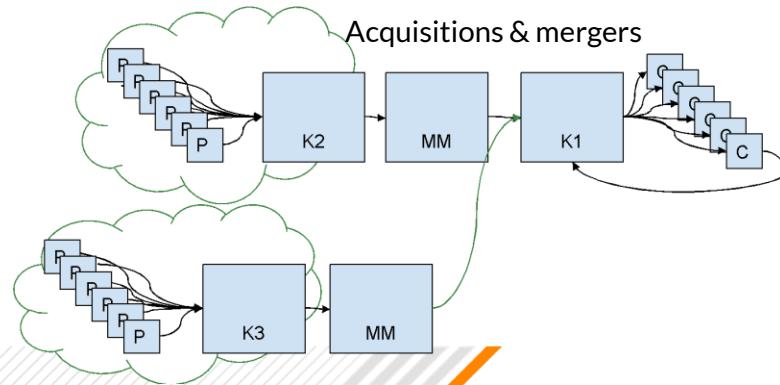
Segmentation



Data Deployment



Acquisitions & mergers



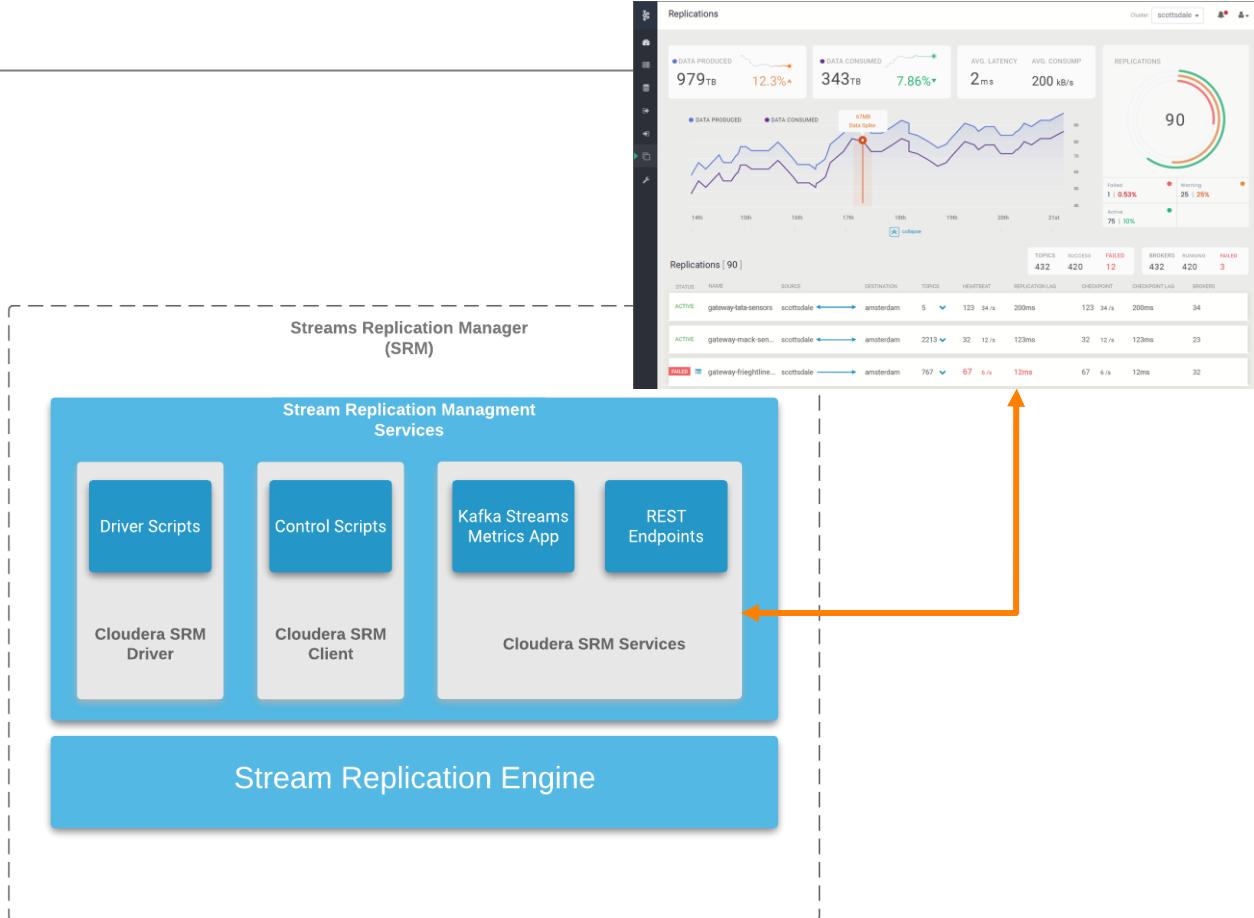
Legacy MirrorMaker

Limitations with Legacy MirrorMaker(MM) for Replication Use Cases

Challenge / Limitation	Description
Static Whitelists and Blacklists	Changes to replication jobs require restart of MirrorMaker cluster.
No Configuration/Metadata Synch	Topic configuration changes in the origin cluster are not detected and propagated to the destination cluster. New Topics are not detected
Scalability and Throughput Limitations due to Rebalances	Cannot scale replication processes as Kafka traffic increases
Lack of Monitoring and Operational Support	No tooling to install and manage replication cluster. Difficult to monitor replication lag, consumer and producer metrics for replication workflows
No Disaster Recovery, Migration, Failover	MM doesn't support active/active without capabilities such as topic renaming, etc. No support for client failover/failover out of the box.

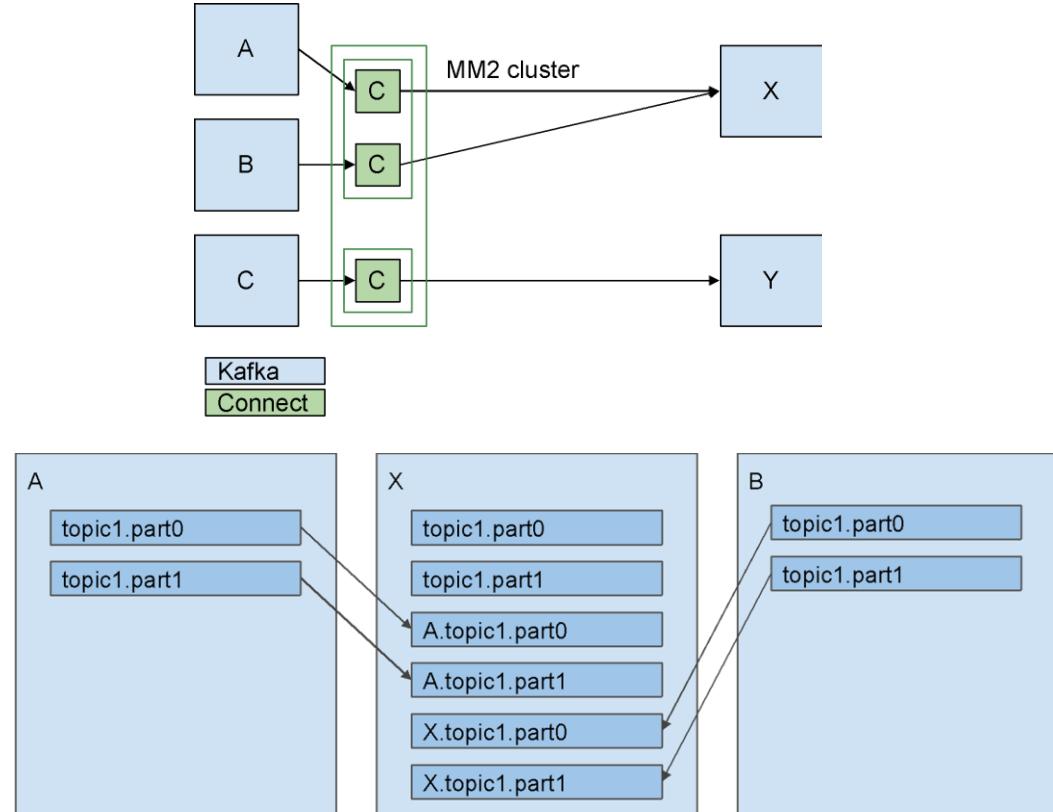
Streams Replication Manager (SRM)

- Easy installation & automation
- Lifecycle management
- Management and monitoring of replication flows across clusters with Streams Messaging Manager
- Smart client for failover and fallback



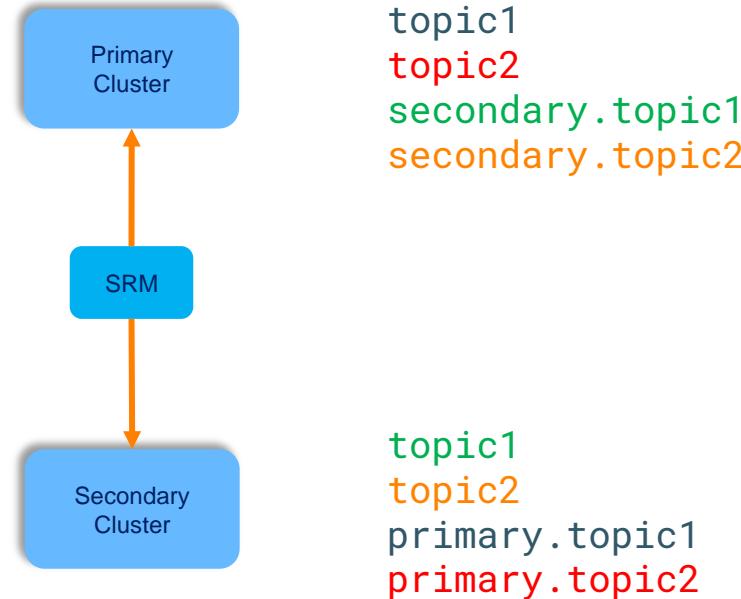
SRM Engine

- Supports active-active, multi-cluster, cross DC replication & other complex scenarios
- Leverage Kafka Connect for scalability and HA
- Replicate data and configurations (ACL, partitioning, new topics, etc)
- Offset translation



Remote topics

- Replicated topics are renamed according to `ReplicationPolicy`.
- Default policy : `<source>.topic`
- Can implement custom policies



4 Key Concepts in MM2

Remote Topics

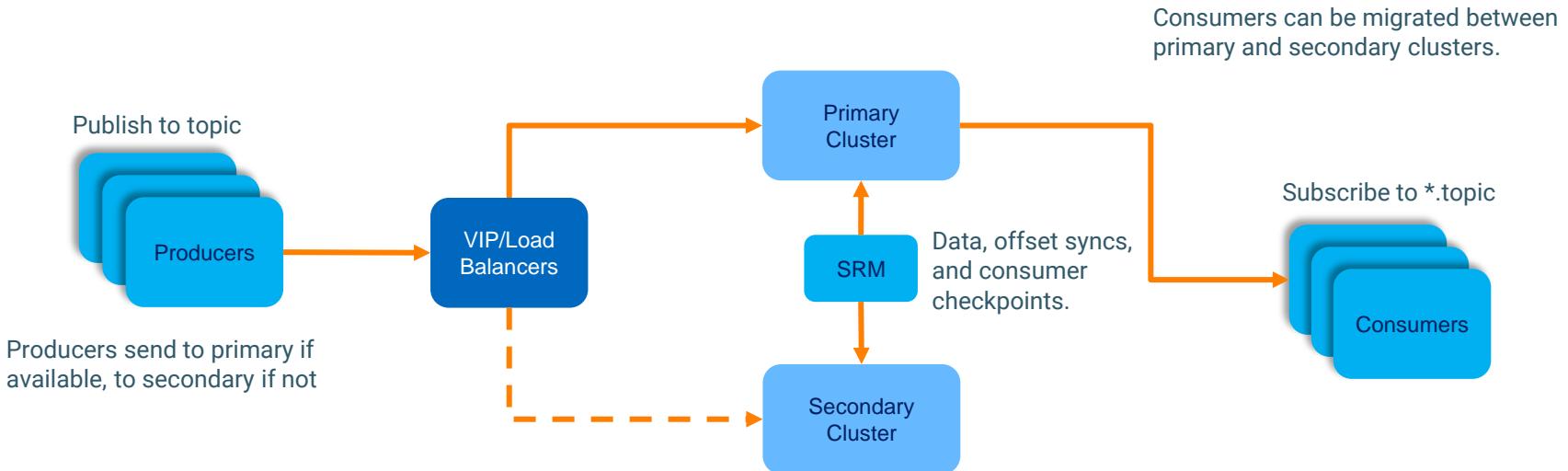
HeartBeats

Offset Syncs

Checkpoints

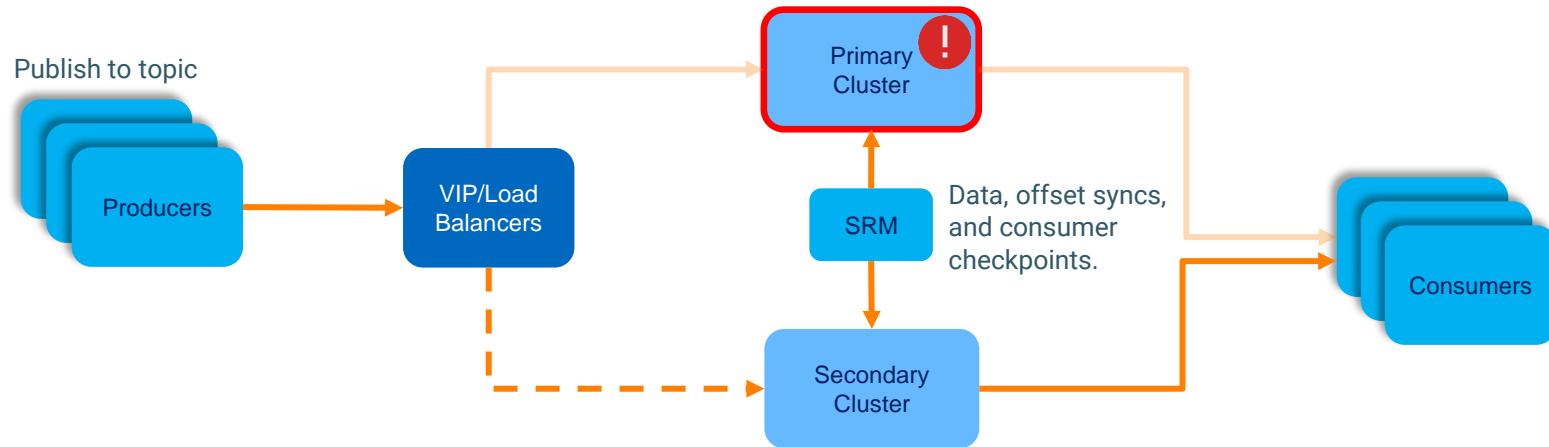
Active Passive

Active Passive



Active Passive

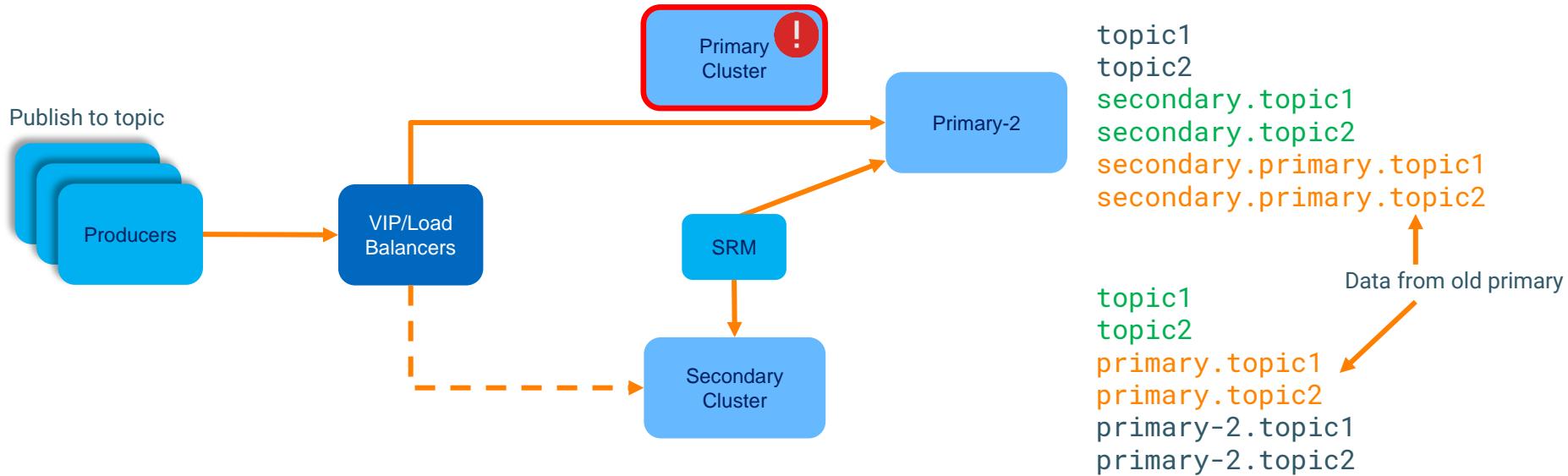
Primary down: failover and migrate consumers



```
$ srm-control offsets --bootstrap-server :9092 --source primary --group foo --export > out.csv  
$ kafka-consumer-groups --bootstrap-server B_host:9092 --reset-offsets --group foo --execute --from-file out.csv
```

Active Passive

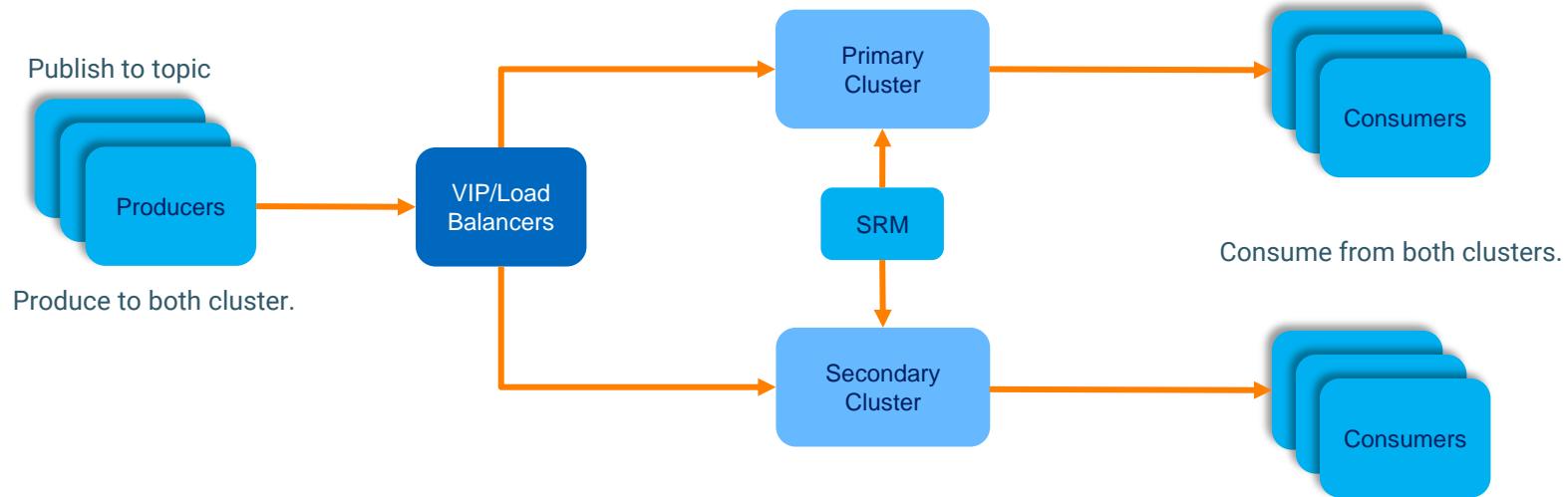
Primary permanently lost? Recover from secondary.



Active Active

Active Active: Cross DC Consumer Groups or Cross DC Replication

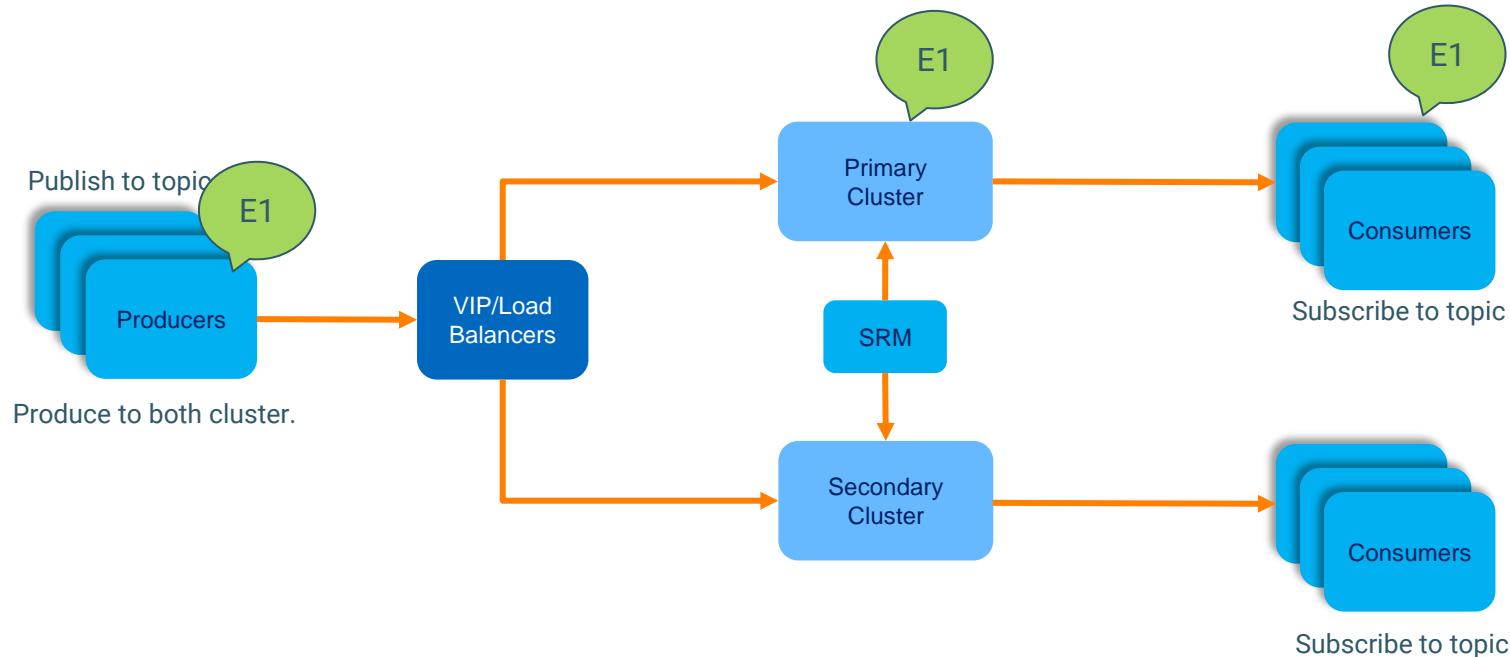
Consumer subscription defines the patterns



A/ Cross DC Consumer Groups

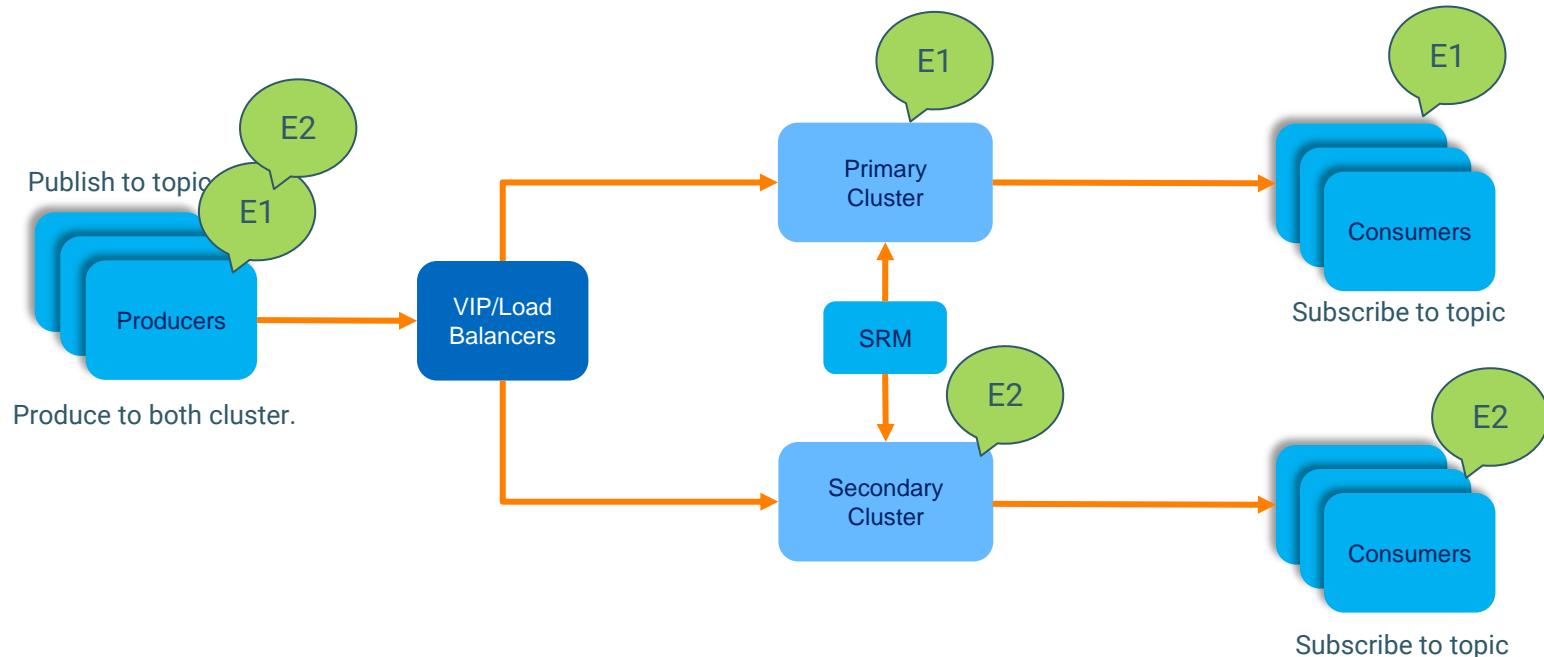
Cross DC consumer groups

Only one consumer processes each record



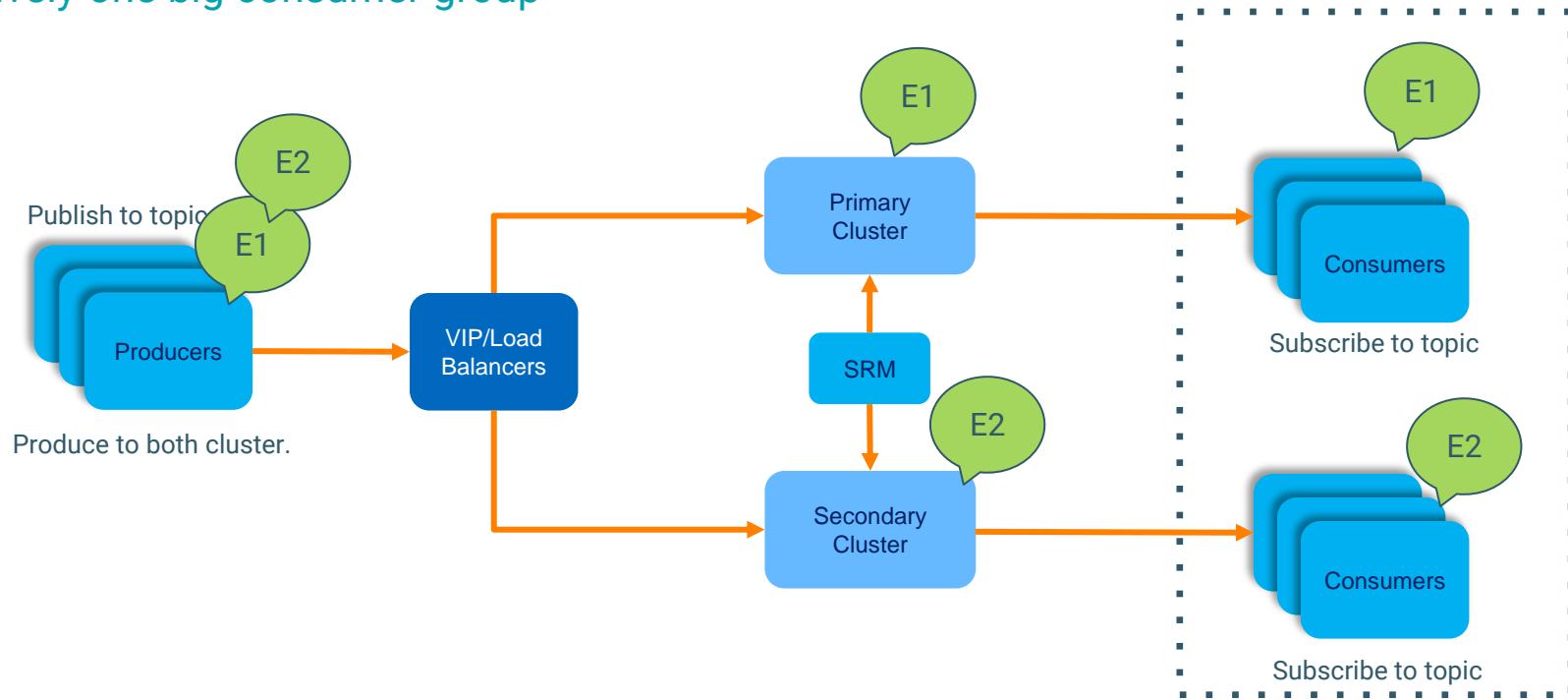
Cross DC consumer groups

Only one consumer processes each record



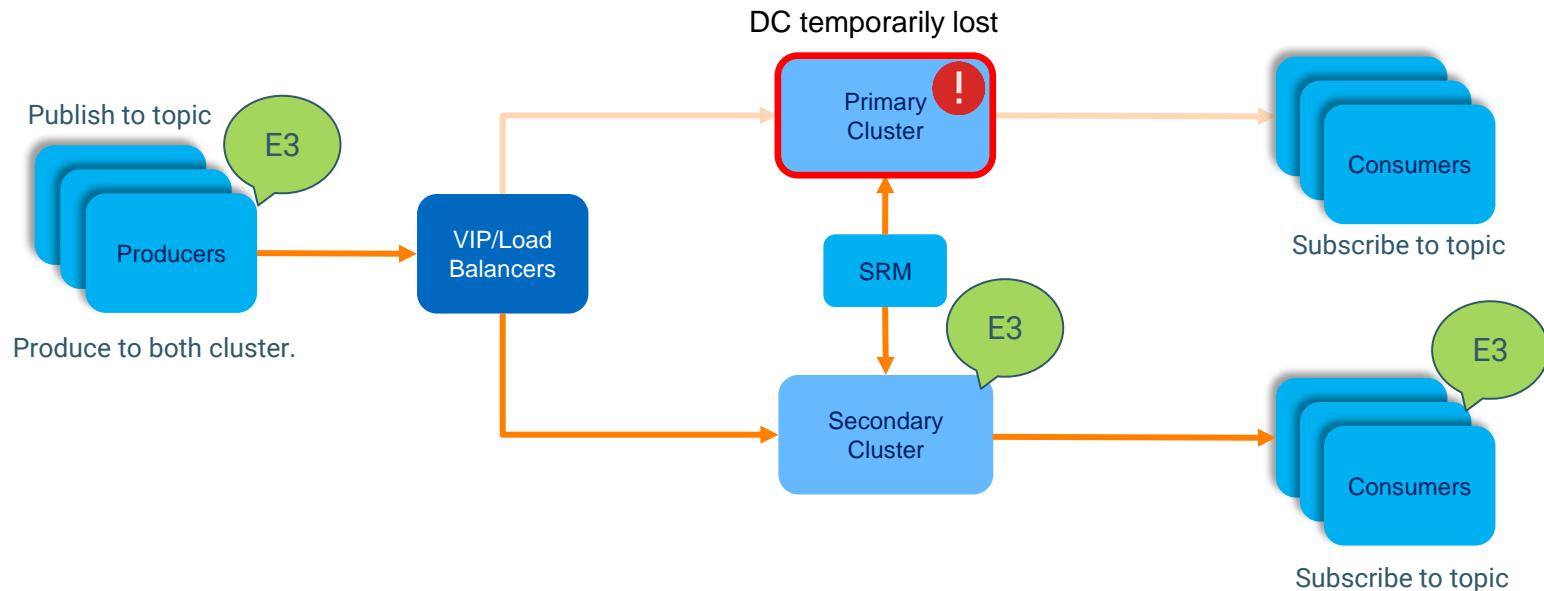
Cross DC consumer groups

Effectively one big consumer group



Cross DC consumer groups

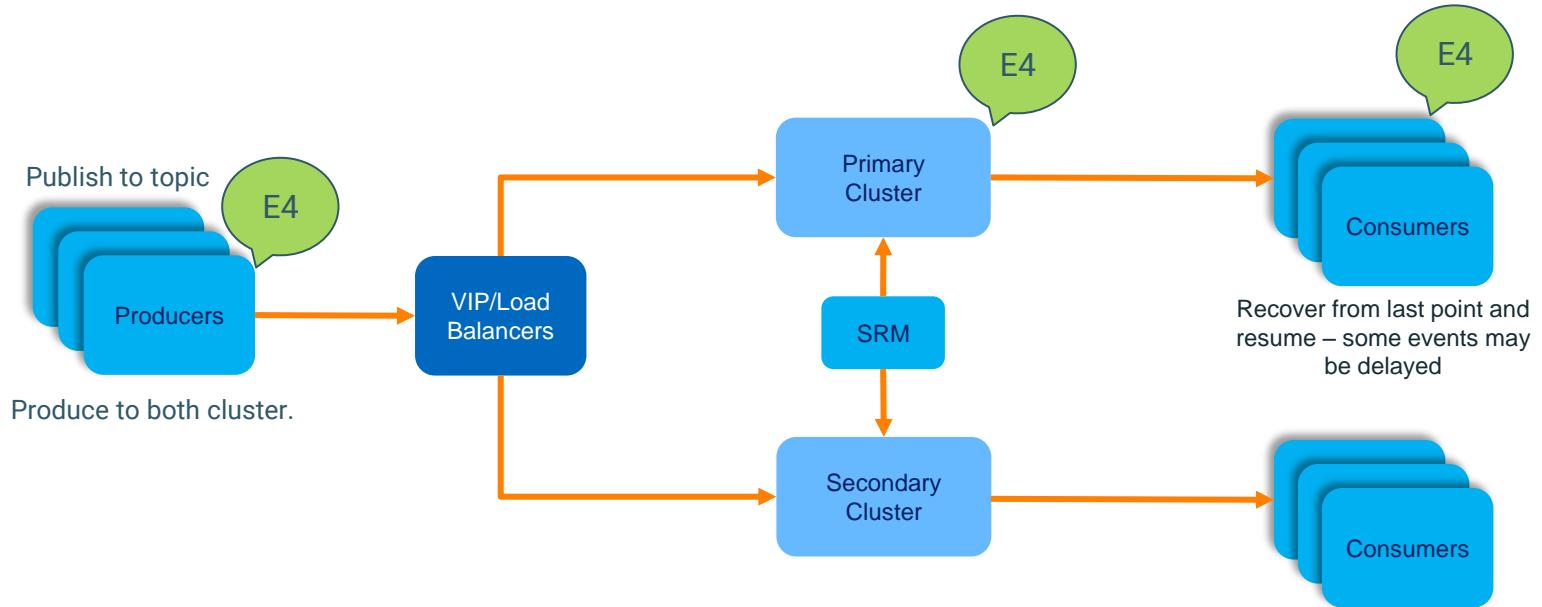
What it takes to fail-over? Nothing



Cross DC consumer groups

What it takes to fail-back? Nothing also

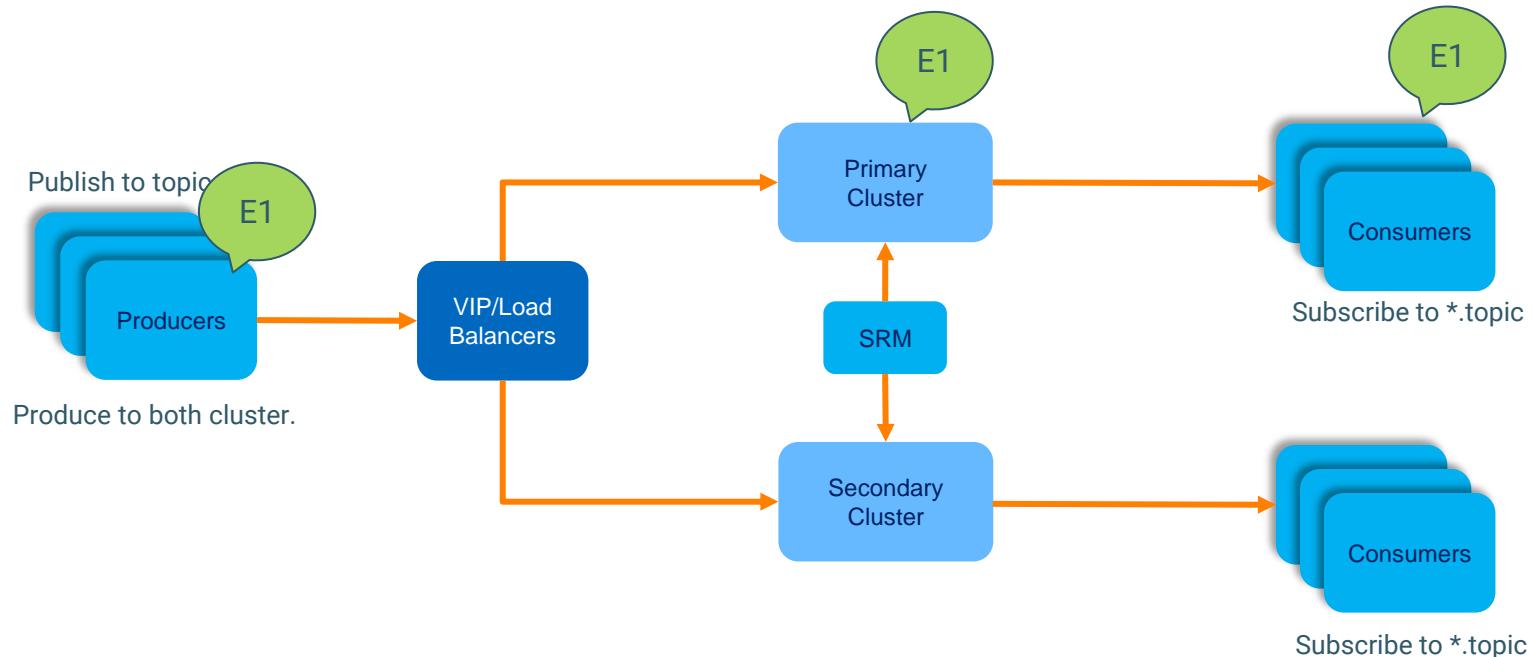
DC issue resolved !



B/ Cross DC Replication

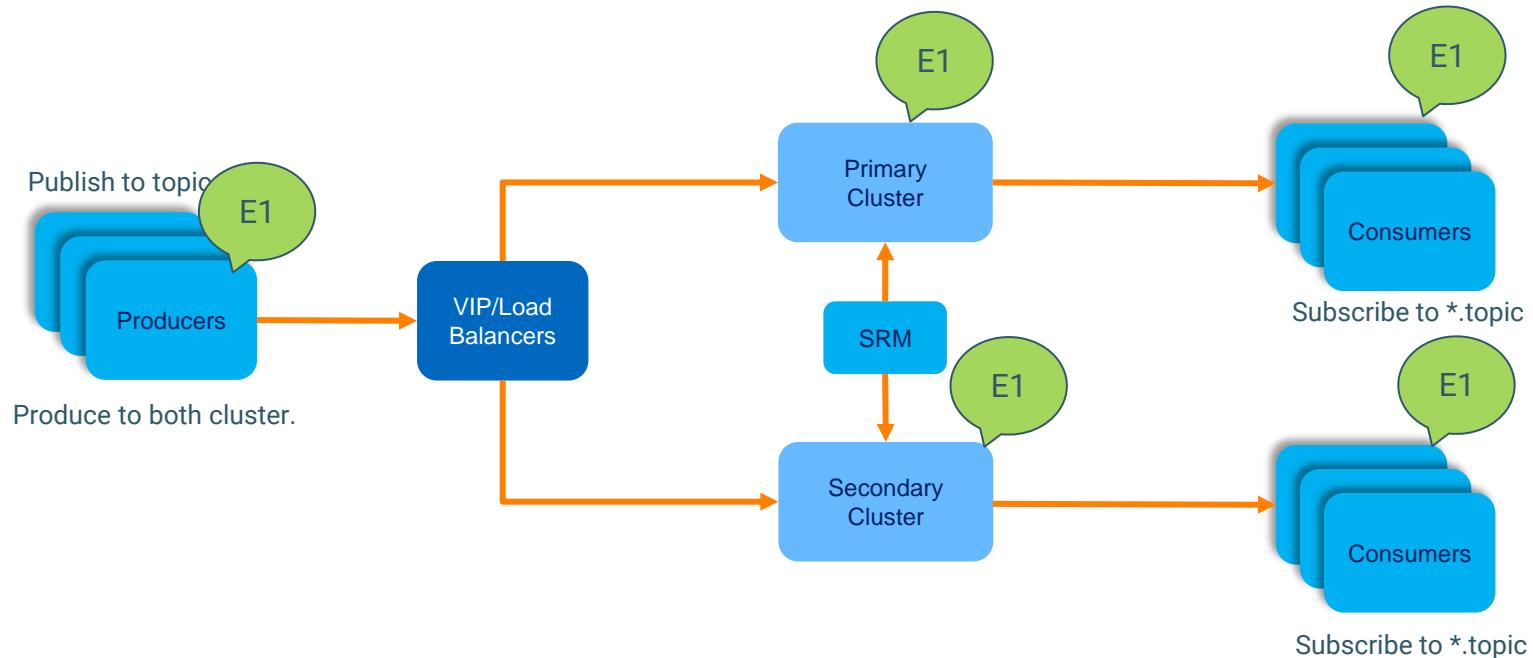
Cross Data Center Replication XDCR

All consumers process all records



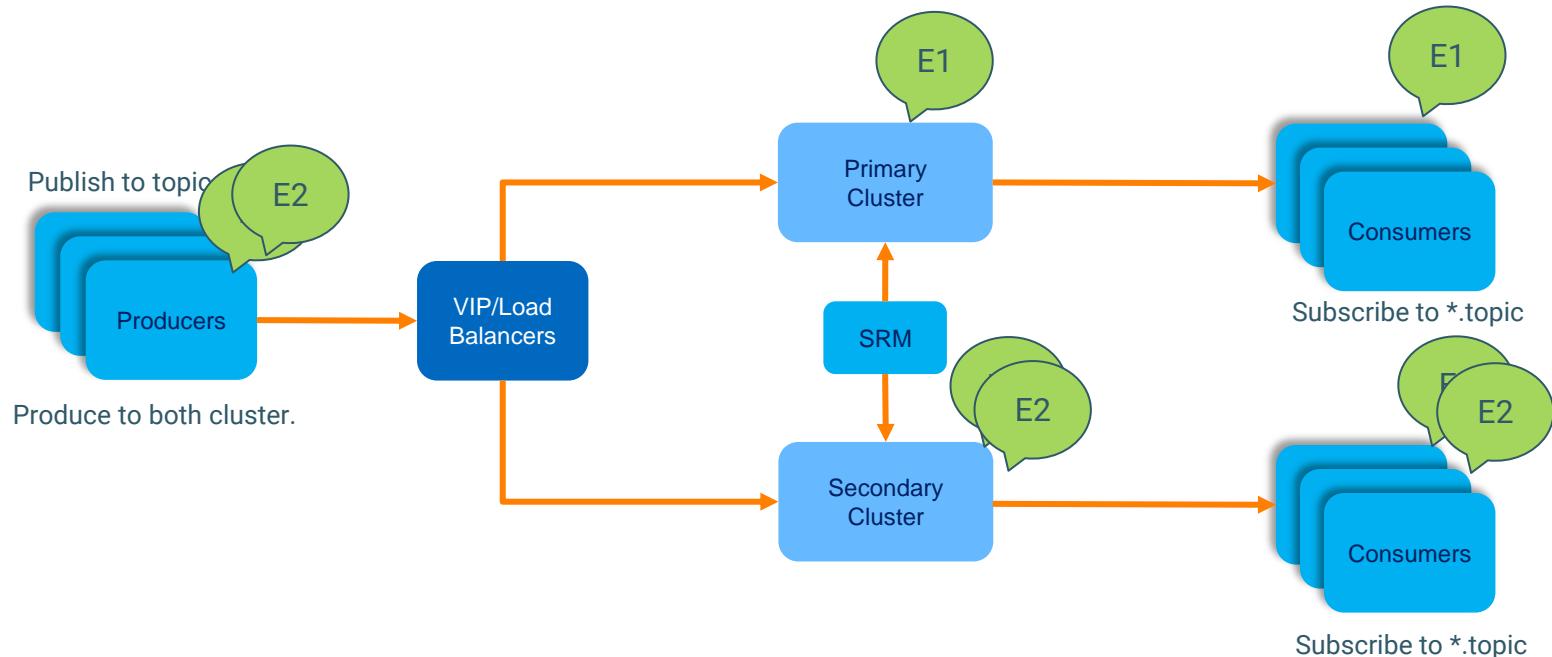
Cross Data Center Replication XDCR

All consumers process all records



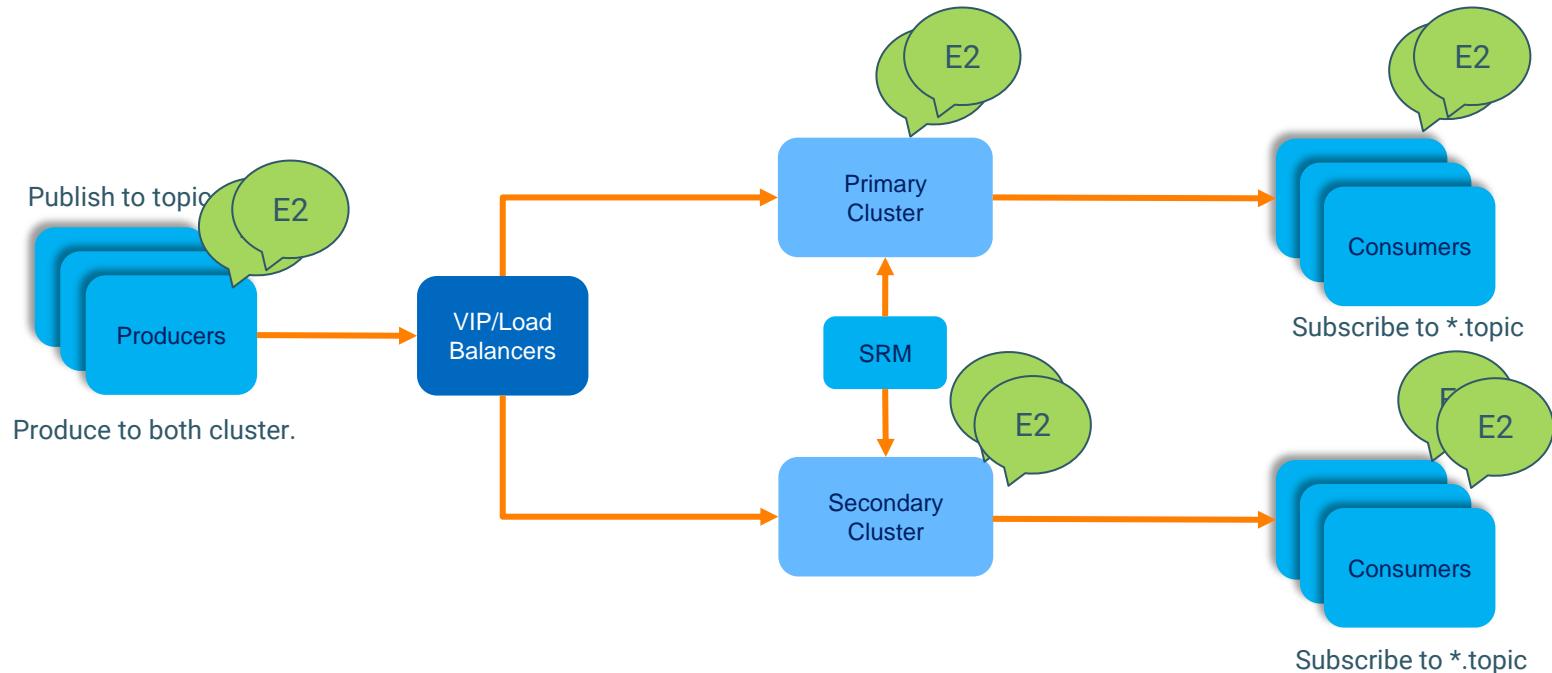
Cross Data Center Replication XDCR

All consumers process all records



Cross Data Center Replication XDCR

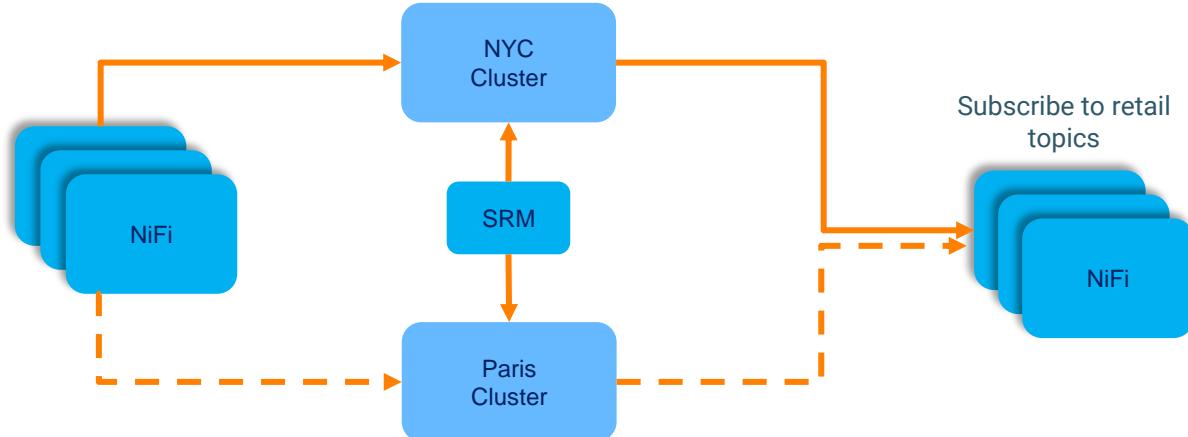
All consumers process all records



Demo DR with SRM

Use cases and scenarios

- WW retailer with critical use cases on Kafka
- Two clusters : NYC & Paris
- NiFi publish/consume events to/from Kafka
- global-retail-* topics are critical and should be replicated
- Local-retail-* are not replicated



STREAMING CORE CONCEPTS

TODAY IF A BYTE OF DATA WAS 1 GALLON OF
WATER WE COULD FILL AN AVERAGE HOUSE IN 10
SECONDS, BY 2020 IT WILL TAKE ONLY 2.

< 1% OF THE DATA BEING GENERATED BY
THE 30,000 SENSORS ON AN OFFSHORE OIL
RIG IS CURRENTLY USED WHEN MAKING
DECISIONS ABOUT MAINTENANCE
PLANNING

SOURCE: 2017 McKinsey Global Institute analysis

DISTRIBUTED STREAMING PROCESSING ENGINES

Quarks | Gearpump | Concord | Heron | Pulsar | Spark |
SQLStream | EsperTech | Google Cloud | S4 | Flink |
DataFlow | Storm | Samza | Impetus | Kafka Streams

Apache Apex | Amazon Kinesis | MemSQL | Bottlenose
Nerve Center | Striim | Azure Stream Analytics | Informatica |
Oracle Stream Analytics |

WS02 Complex Event Processor | IBM InfoSphere Streams |
Cisco Connected Streaming Analytics | SAS Event Stream
Processing | WS02 Complex Event Processor | TIBCO
StreamBase | Software AG Apama Streaming Analytics |
MongoDB | Ignite | XAP



Summary

Already using **Spark**?

Want unified **Batch/Stream**?

Need highest **Throughput**?

Don't need **low latency**?

Don't need advanced
Time/State features?



Like **Kafka**?

Only need **Microservices**?

Want **Low Latency**?

Want **Just Kafka**?

Don't need advanced
Time/State features OOTB?



Like **Flink**?

Want **Flexibility**?

Want **best-in-class**
Technology?

Want **Low Latency**?



Kafka Streams and Spark Structured Streaming

Picking the Right Streaming Engine Based on the Use Case / Requirement

Non-Functional Requirement (NFR) / Use Case	Spark Structured Streaming	Kafka Streams
Stream Processing Style	Micro-batching. Event at a time aka continuous mode is still experimental	Event a time
State Management	Implicit Support with HDFS	Implicit support with RocksDB
Fault Tolerance	Implicit Checkpoint support with HDFS	Implicit support with internal Kafka topics / ZK
Advanced feature: Joins/ Aggregations/ Windowing	Stream/Stream Joins, Stream/Table joins, joins using different keys, Windowing, OOO aggregation processors	Stream/Stream joins, Stream/Table joins, joins with only message key, no OOO aggregation processors
Advanced feature: watermarking	Supported	Not fully supported
Latency	Medium	Low
Throughput	High	Medium
APIs	Simple APIs (Declarative)	Simple APIs (Compositional)
SQL DSL	First Class feature within Spark	Nothing in Apache Kafka.
Lambda Architecture	First Class support. Write app and execute in streaming or batch mode	No Support
GUI Tooling / Code-Less Approach	Not Supported	Not Supported
Maturity	Maturing and Growing Community	Relatively New
Use Cases	complex stream processing use cases, ETL	Lightweight eventing based microservices, ETL
Cluster Requirement	Yes	No

Comparison: Technical Features

	Flink 1.9	Kafka Streams 2.3	Spark Structured Streaming 2.3	Storm 2.0 and Storm Trident
Approach / Position	Streaming First, modern class-leader	Message-Broker First, popular Streaming Add-on	Batch First, popular Streaming Add-on	Streaming First, legacy architecture
Streaming Model	1st class Stateful, (<500ms), Event-at-a-time	Stateful, (<500ms), Event-at-a-time	Stateful, (>1s), Microbatch	Natively stateless, (<1s), Event-at-a-time, Stateful plugins
Time Support	Event/Processing/Custom	Event/Processing	Event/Processing	Processing OOTB, Event developable
Processing Abstractions	Table/CEP/Graph/ML/SQL. Batch experimental	Table/SQL(KSQL). No Batch.	Table/ML/Graph. Unified APIs for Batch and Stream	Streaming only OOTB
SQL Abstraction	based on Apache Calcite, ANSI compliant	SQL-like DSL (KSQL), not ANSI compliant	ANSI SQL:2003 in Spark 2.3	Experimental since 1.2.3
Delivery Guarantee	Exactly Once Upstream, some Downstream	Exactly once, Kafka-only	Exactly Once Upstream, some Downstream	At least once (Trident exactly once Upstream)
State Mgmt	RocksDB, Configurable Snapshots, Queryable	RocksDB, Snapshot to Kafka Topic, Queryable	OSS Async on HDFS. RocksDB Databricks Only	External, not OOTB
Fault Tolerance / Resilience	Built-in, Checkpoints, Savepoints, Redistributable	BYO Microservice	Built-in	Built-in

Comparison: Operational Features

	Flink 1.9	Kafka Streams 2.3	Spark Structured Streaming 2.3	Storm 2.0 and Storm Trident
Deployment Model	Clustered, Kubernetes, Microservices	Microservices, Kubernetes	Clustered, Kubernetes	Clustered
Documentation	Good Technical Docs, growing examples and SO	Extensive Docs and Examples and SO coverage	Extensive Docs and Examples and SO coverage	Good for 1.x
Maturity / Community	Smaller but Fastest growing community with strong research and Production deployments	Newest, strong community with strong growth	Spark community is strong, but Streaming is a small quiet corner	Oldest framework, community eclipsed by newer Engines
Use Cases	Everything	Microservice / Event driven, Embedded in another Application	Unified ETL, semi-RT processing	IOT, complex event processing
Enterprise Management	Rich OSS, enhanced vendor offerings	Minimal OSS, Some via vendor offerings	Rich OSS, enhanced vendor offerings	Some integrations
Push Button Security	Complex, some OSS support, limited vendor offerings	Simple, some OSS support, good vendor offerings	Complex, good OSS support, good vendor offerings	Complex, good OSS support, good vendor offerings
Logging / Metrics	Usual OSS integrations, some vendor offerings	BYO Microservice	Good Logging integration	Good Logging integration
Scaling Up/Down	Not yet Autoscaling, but all requirements available	BYO Microservice	Not yet Autoscaling, but all requirements available	Management Tools help, but Tuning is challenging

Streaming Analytics on CDF

- Simple operation: split, join, filtering, enrichment, etc
- Time window operators
- Complex Event Processing
- Pattern matching
- Predictive and Prescriptive Analytics



Publish subscribe
high-throughput, low-latency broker



Leverage Spark micro-batching capabilities for HDP/CDH customers



API for building real-time applications and microservices on top of Kafka



MOST SIGNIFICANT REQUIREMENTS

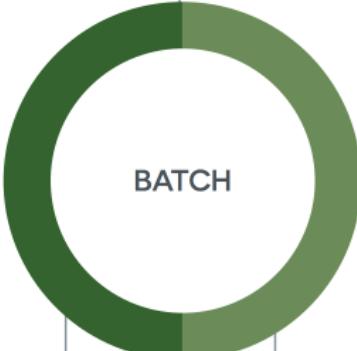
1. Process an instance at a time, and inspect it (at most) once
2. Use a limited amount of time to process each instance
3. Use a limited amount of memory
4. Be ready to give an answer (prediction, clustering, pattern) at any time
5. Adapt to temporal changes

**THINKING ABOUT...
TIME**

WHAT IS REAL-TIME?

> 1 HOUR

high throughput



adhoc queries

monthly active users relevance for ads

10 MS – 1 SEC

approximate



ad impressions count hash tag trends

< 500 MS

latency sensitive



deterministic workflows

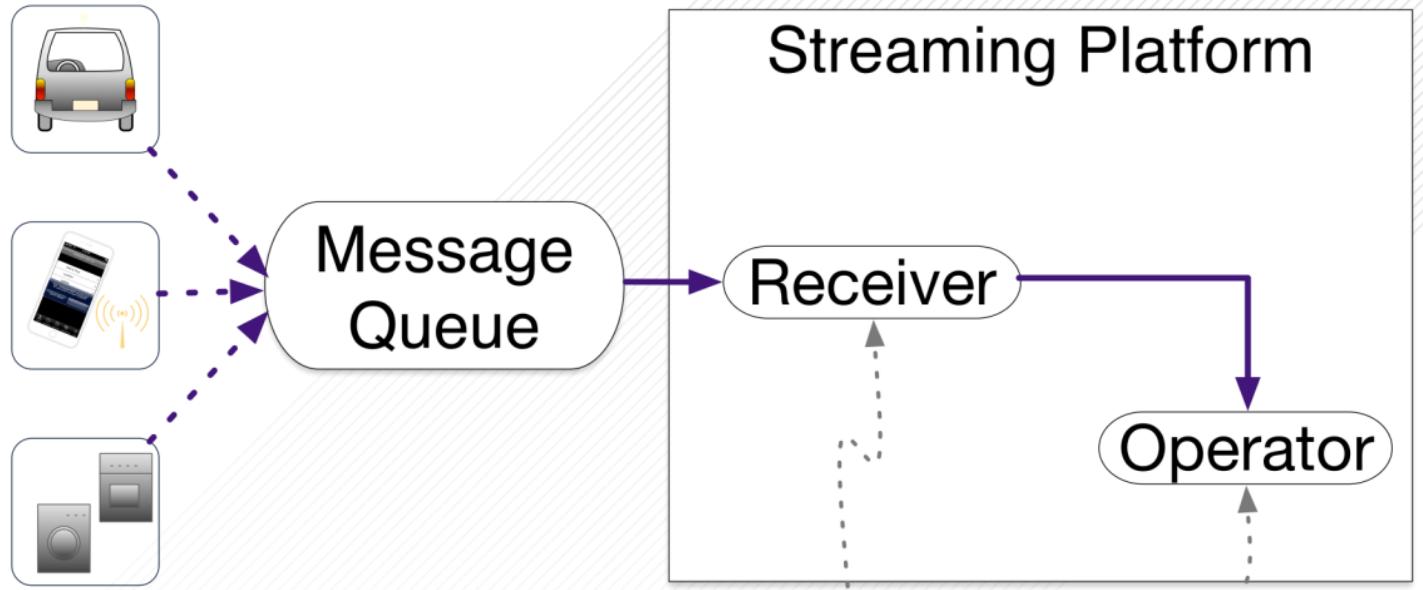
fanout Tweets search for Tweets

< 1 MS

low latency



Financial Trading

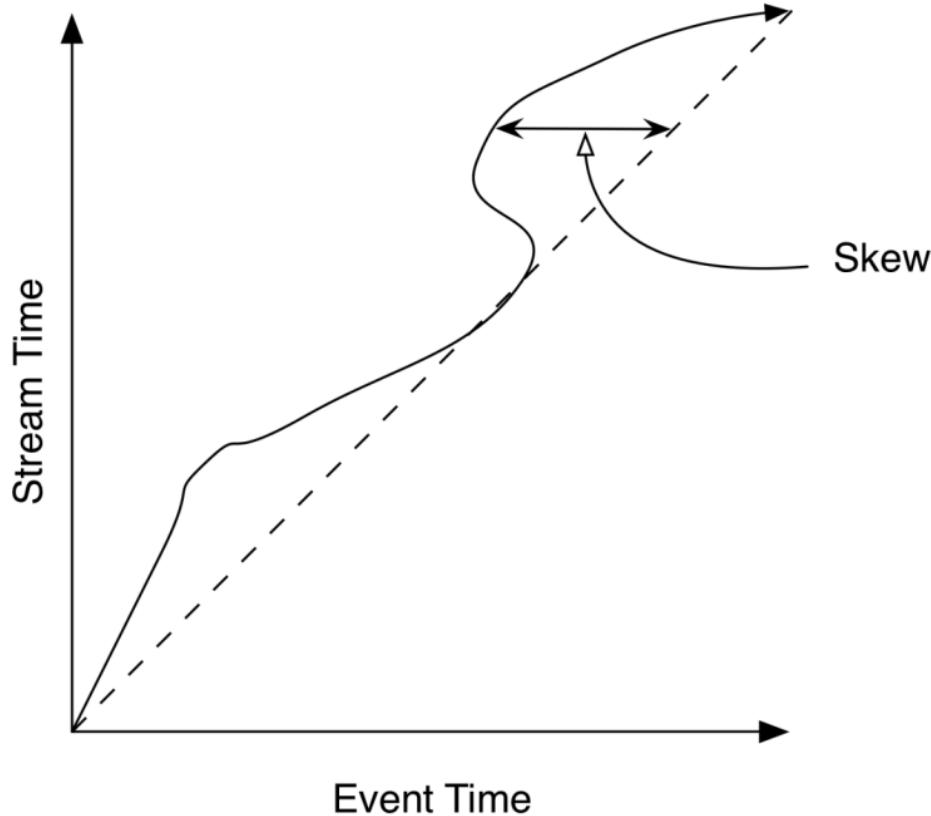


Event
Time

Stream Time
(Ingestion)

Processing
Time

TIME SKEW



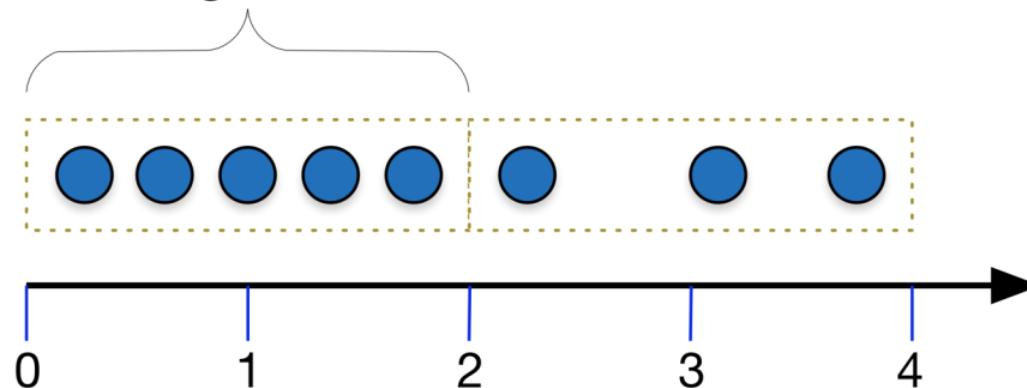
THINKING ABOUT...
WINDOWS

TUMBLING WINDOWS

TUMBLING TIME WINDOWING

Window length

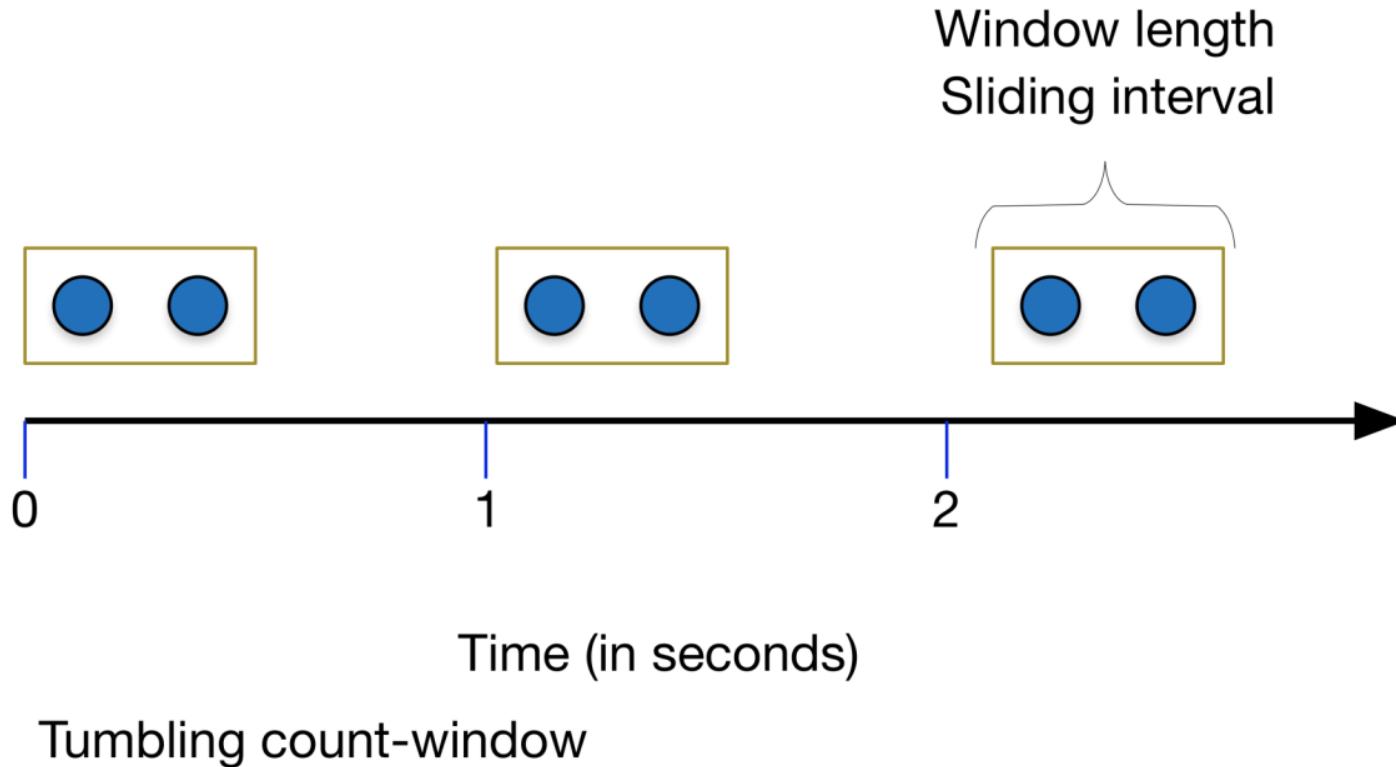
Sliding interval



Time (in seconds)

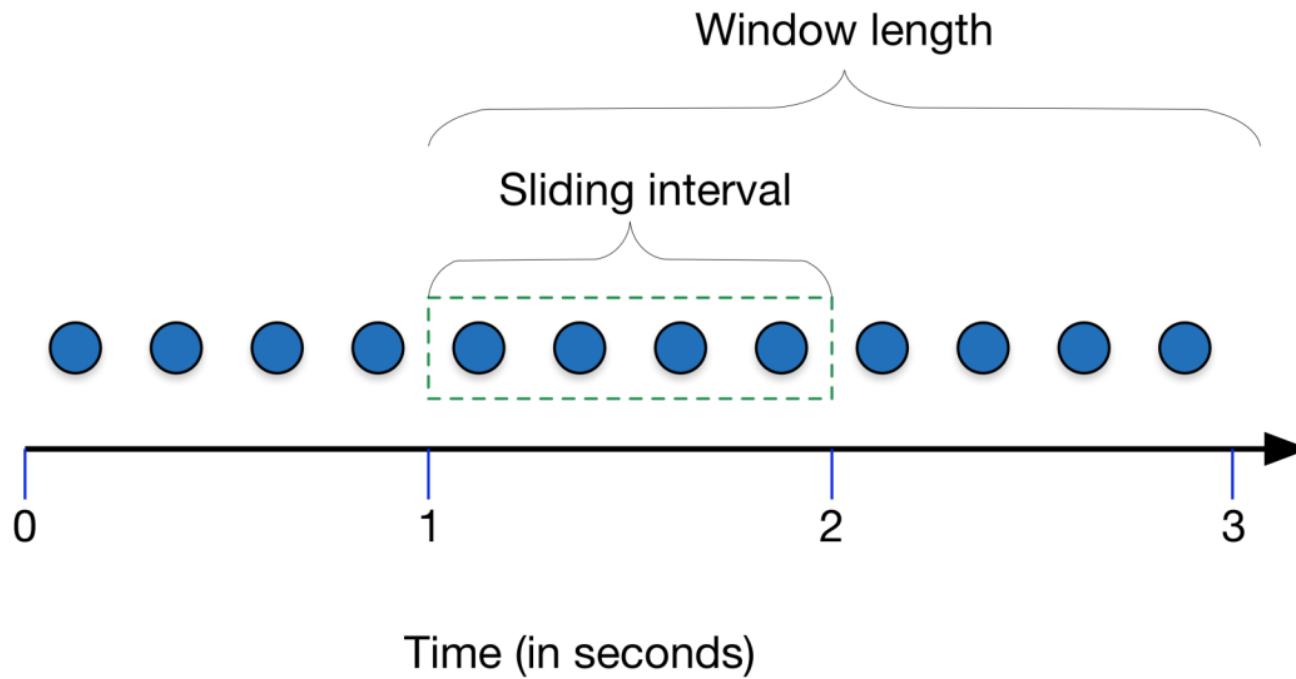
Tumbling temporal window

TUMBLING COUNT WINDOWING

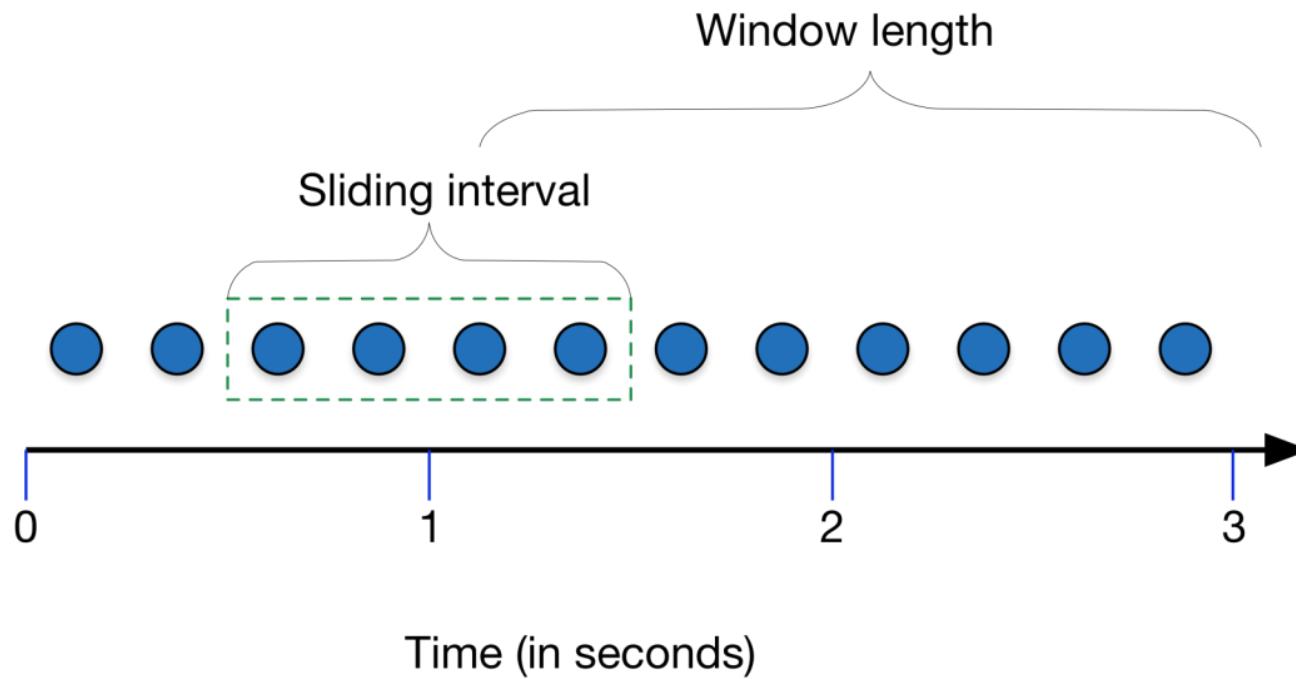


SLIDING WINDOWS

SLIDING TIME WINDOW

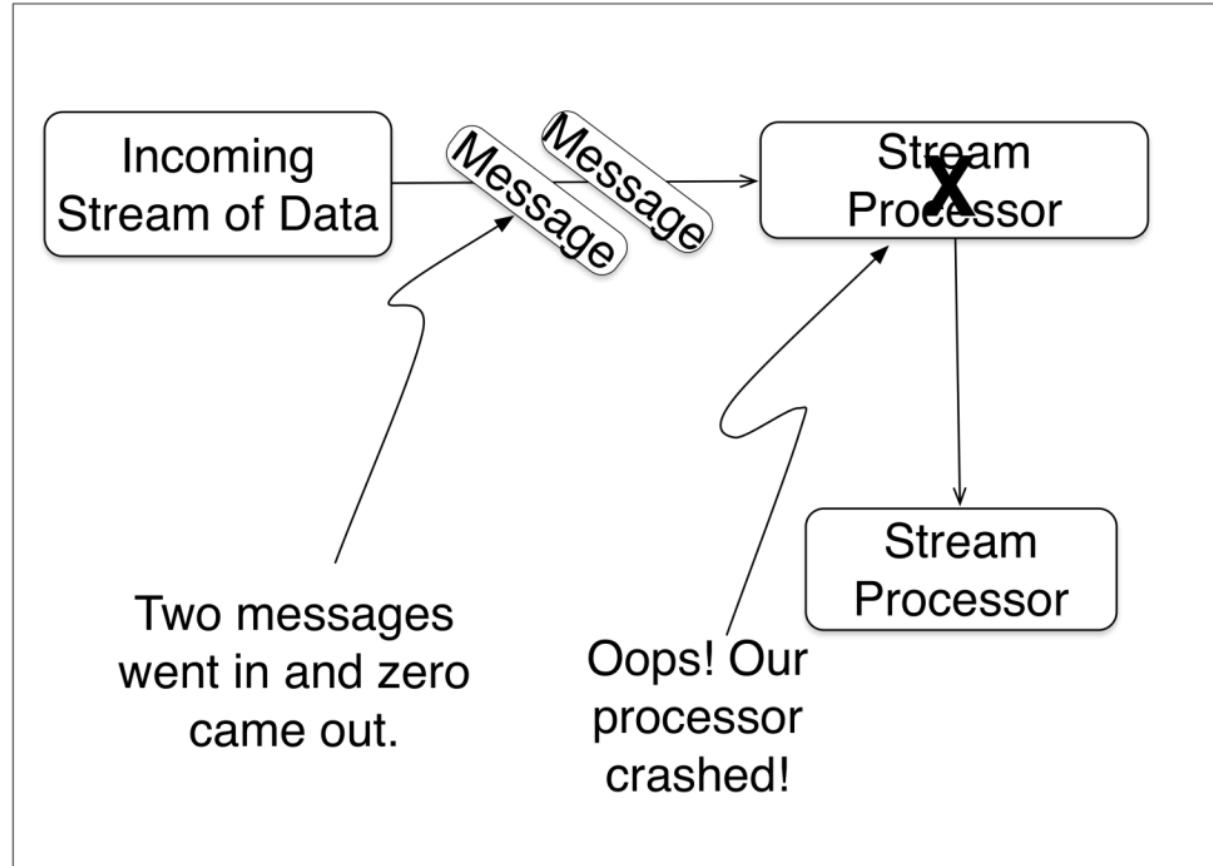


SLIDING COUNT WINDOW

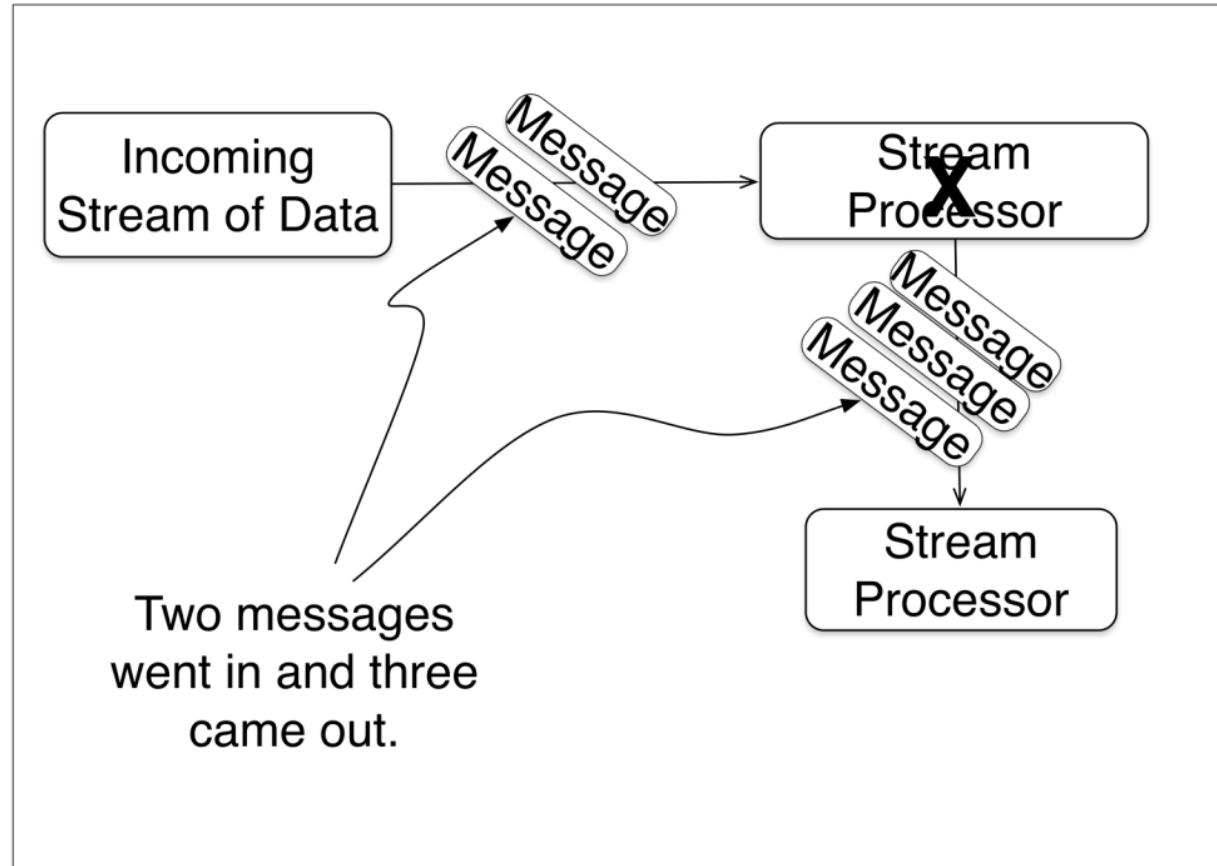


THINKING ABOUT...
SEMANTICS

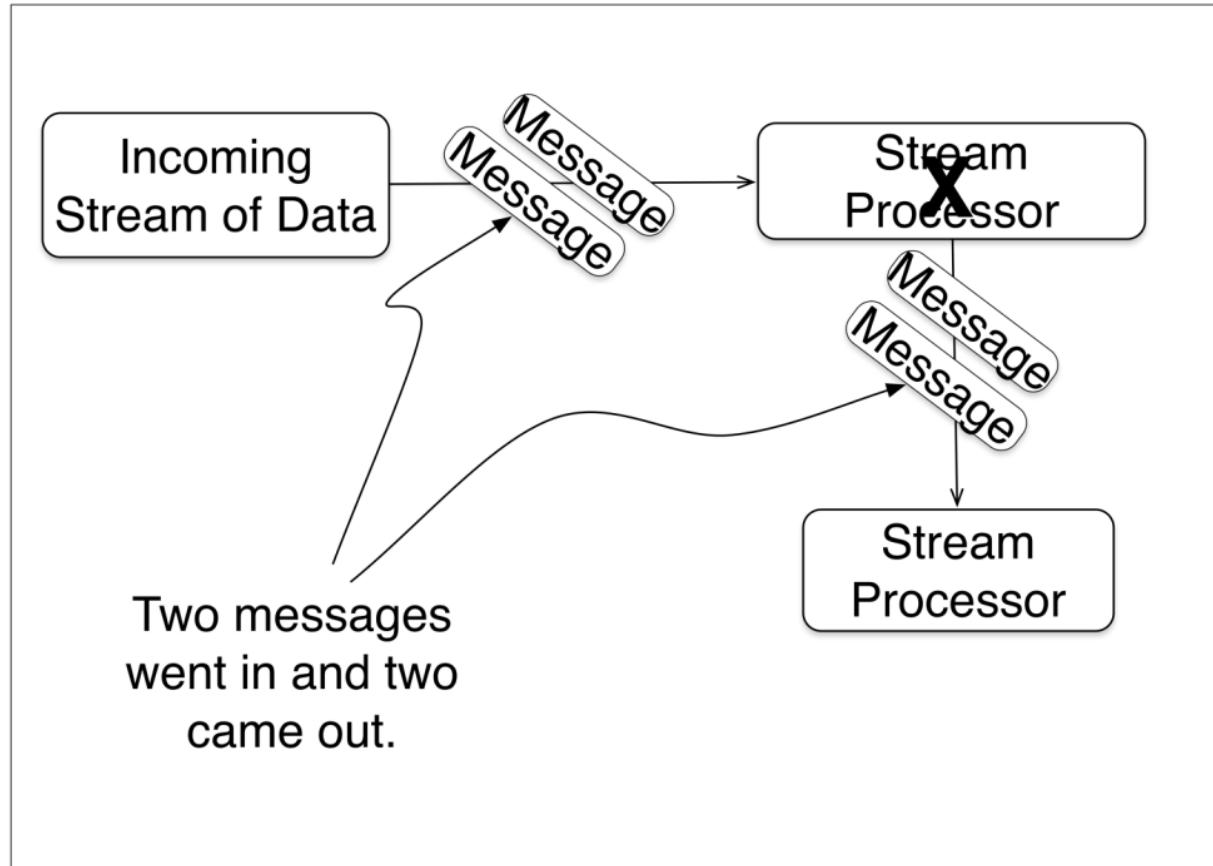
AT-MOST-ONCE



AT-LEAST-ONCE



EXACTLY-ONCE



Final Lab Architecture

PREDICTING MACHINE FAILURE



Photos by Dominik Vanyi on Unsplash



TEMPERATURE SENSORS

Photo by Kait Dahms on Unsplash



- Temperatures of different truck components
- 12 per truck
- Transmitted by radio to a local field office
- Data is published to a MQTT server

TRUCK HEALTH MODEL

- Cloudera Data Science Workbench (CDSW)
 - Data exploration
 - Model creation, training and deployment
- REST API

The screenshot shows the Cloudera Data Science Workbench (CDSW) interface. The left sidebar has a dark theme with white text and icons. It includes sections for Overview, Sessions, Experiments, Models (which is currently selected), Jobs, Files, Team, and Settings. The main content area is titled "IoT Prediction Model". At the top right of this area are buttons for Deployed, Stop, Restart, and Deploy New Build. Below this is a sub-menu with tabs for Overview, Deployments, Builds, Monitoring, and Settings. The Overview tab is active. Under "Model Details", there is a table with columns for Model Id, Deployment, Build, Deployed By, Comment, Kernel, Engine Image, File, and Function. The "Test Model" section contains an "Input" field with a JSON sample code block, a "Test" button, and a "Reset" button. The "Result" section shows a table with columns for Status, Response, and Replica ID. The "Status" row shows a green "success" status. The "Response" row shows a JSON object with a single key "result": 1. The "Replica ID" row shows the identifier "iot-prediction-model-1-1-79d76dcf44-fwqvm". At the bottom right of the main content area, the text "1.6.0.1294376 (46715e4)" is visible.

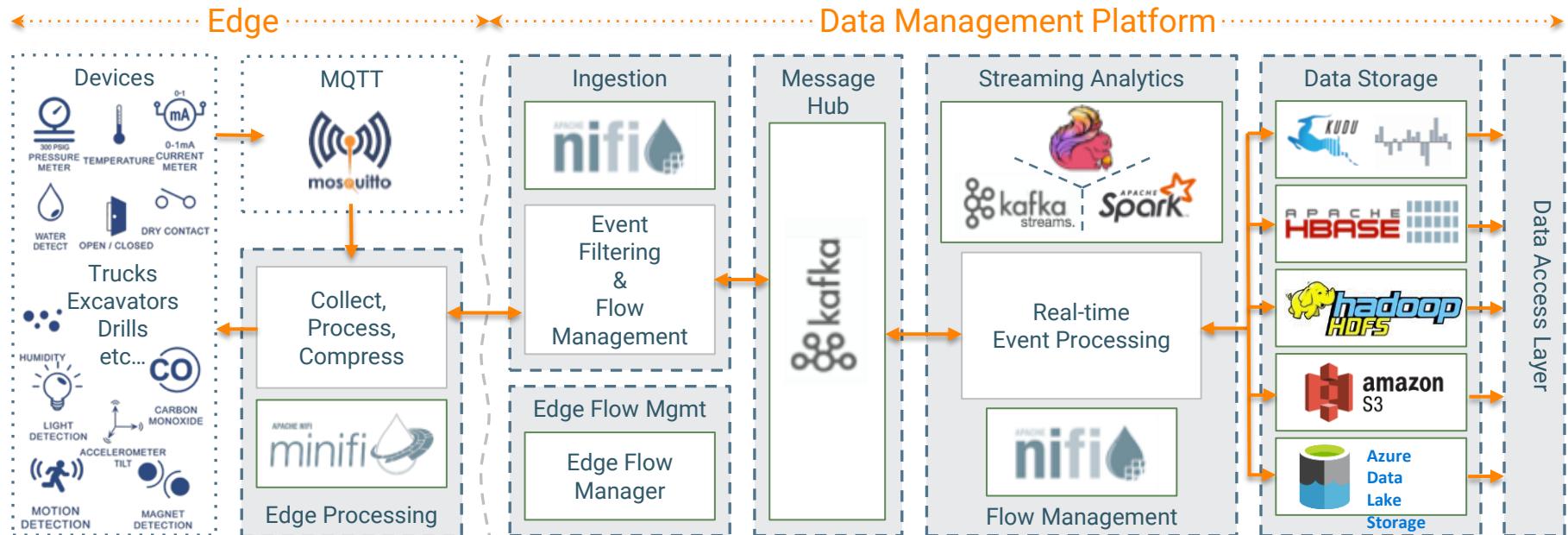
Model Id	1
Deployment	1
Build	1
Deployed By	admin
Comment	Initial revision.
Kernel	python3
Engine Image	Base Image v8
File	cdsw.iot_model.py
Function	predict

Status	success
Response	{ "result": 1 }
Replica ID	iot-prediction-model-1-1-79d76dcf44-fwqvm

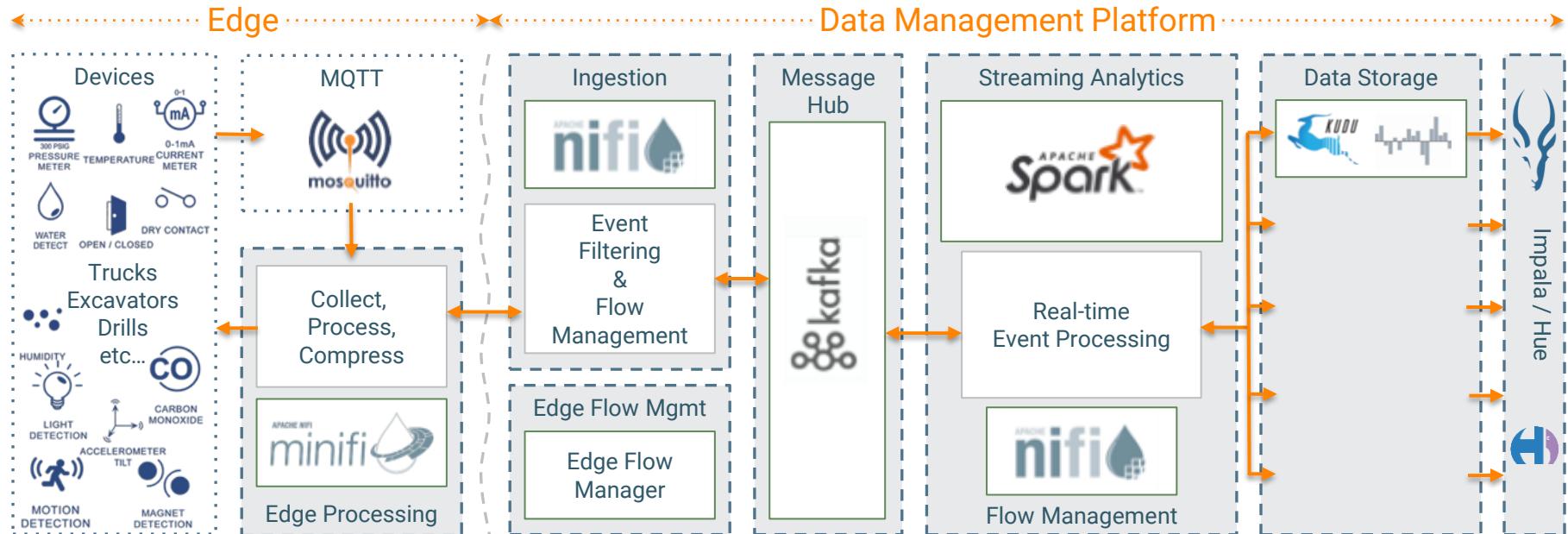
STREAMING AND PREDICTING

- Process the real-time data and it's collected from truck sensors
- Forward data from remote locations to our central Apache NiFi and to our main Apache Kafka message bus
- Read sensor readings from Apache Kafka and enrich the data
 - Call the deployed health model API to predict truck failures
- Update the prediction results in Apache Kudu
- Enable fast analytics on Apache Kudu

REFERENCE ARCHITECTURE



AND THAT'S WHAT WE DID TODAY!



Summary

Visualization

Apache Hue



SQL and Query Editor & Performance Diagnostics
Tool for the Cloudera Data Platform

A screenshot of the Cloudera SQL and Query Editor interface. At the top, there's a search bar for saved documents and a status bar showing "0.70s Database default Type text". The main area has tabs for "Query" and "Tables".

Tables

Table	Type	Columns
sensors	Text	uuid (string) end (string) ipaddress (string) top1pct (double) top1 (string) cputemp (string) gputemp (string) gputempf (string) gputempf (string) runtime (string) host (string) filename (string) imageinput (string) host_name (string) macaddress (string) te (string) systemtime (string) cpu (double) diskusage (string) memory (double) id (string)

Query

```
1 select * from sensors
2 order by systemtime desc
```

Query History

Query ID	Query	Status	Time
a94ee753c7068adb:bcf3428c00000000	select * from sensors	0% Complete (0 out of 16)	0.70s
a94ee753c7068adb:bcf3428c00000000	order by systemtime desc	0% Complete (0 out of 16)	0.70s

Saved Queries

Results (463)

uuid	end	ipaddress	top1pct	top1	cputemp	gputemp	gpu
1 nano_uuid_lwl_20190930162453	1569860698.3446946	192.168.1.200	28.564453125	teddy,teddy bear	28.0	28.0	82
2 nano_uuid_ykl_20190930162446	1569860690.9064593	192.168.1.200	33.3740234375	teddy,teddy bear	28.5	28.0	82
3 nano_uuid_spd_20190930162430	1569860675.1053476	192.168.1.200	36.62109375	teddy,teddy bear	28.5	28.0	82
4 nano_uuid_mjp_20190930162422	1569860668.015522	192.168.1.200	30.9814453125	teddy,teddy bear	29.5	28.5	83
5 nano_uuid_bvp_20190930162414	1569860659.530218	192.168.1.200	21.8017578125	teddy,teddy bear	28.5	28.0	82
6 nano_uuid_ady_20190930162407	1569860652.219771	192.168.1.200	32.2998046875	teddy,teddy bear	29.0	28.0	82
7 nano_uuid_wiq_20190930162353	1569860637.7323692	192.168.1.200	25.830078125	teddy,teddy bear	28.5	28.0	82
8 nano_uuid_ucm_20190930162346	1569860630.4993556	192.168.1.200	27.9052734375	teddy,teddy bear	29.5	28.0	82
9 nano_uuid_ery_20190930162339	1569860623.4607024	192.168.1.200	23.14453125	teddy,teddy bear	29.0	28.0	82
10 nano_uuid_cdx_20190930162332	1569860616.4739137	192.168.1.200	23.828125	teddy,teddy bear	29.0	28.0	82
11 nano_uuid_tbx_20190930162324	1569860609.5535312	192.168.1.200	26.5625	teddy,teddy bear	28.0	28.0	82

Visualization

A screenshot of the Cloudera Hue interface. The top navigation bar shows tabs for Cloudera, CEM, NIFI Flow, NIFI Rep, Schema, STREAM, localhost, Hue - Ed, Sign In, and a plus sign. The main title is "Not Secure | ec2-54-202-2-197.us-west-2.compute.amazonaws.com:8888/hue/editor?&type=Impala". A message at the top says, "You are accessing a non-optimized Hue, please switch to one of the available addresses: http://ip-10-0-1-84.us-west-2.compute.internal:8889". The left sidebar shows a "default" database with a single table "sensors". The main area has a "Query" dropdown menu open, showing options like Editor, Impala, Scheduler, Hive, Pig, Java, Spark, MapReduce, Shell, Sqoop 1, and Distcp. Below the dropdown is a search bar for saved documents. To the right of the search bar is a "Tables" section which currently displays "No tables identified". On the far right, there are "Jobs" and "admin" buttons. The bottom of the screen shows a footer with the URL "...amazonaws.com:8888/hue/editor/editor?&type=impala".

...amazonaws.com:8888/hue/editor/editor?&type=impala

Apache NiFi - Storage - Apache Kudu

Property	Value
Kudu Masters	7051
Table Name	impala::default.sensors
Kerberos Credentials Service	No value set
Skip head line	false
Record Reader	AvroReader
Insert Operation	INSERT
Flush Mode	AUTO_FLUSH_BACKGROUND
FlowFiles per Batch	1
Max Records per Batch	100

REFERENCES

- <https://www.datainmotion.dev/2019/05/cloudera-edge-management-introduction.html>
- <https://www.datainmotion.dev/2019/09/powering-edge-ai-for-sensor-reading.html>
- <https://www.datainmotion.dev/2019/08/google-coral-tpu-with-edge-devices-and.html>
- <https://www.datainmotion.dev/2019/08/rapid-iot-development-with-cloudera.html>
- <https://www.datainmotion.dev/2019/07/edge-data-processing-with-jetson-nano.html>
- https://github.com/purn1mak/SMM-NewYork/blob/master/SMM_Kafka_CrashCourse.pdf
- <https://docs.cloudera.com/cem/1.0.0/release-notes/topics/cem-whats-new.html>

REFERENCES

- https://www.slideshare.net/Hadoop_Summit/kafkasmm-crash-course
- https://www.slideshare.net/Hadoop_Summit/byop-custom-processor-development-with-apache-nifi
- https://www.slideshare.net/Hadoop_Summit/intelligently-collecting-data-at-the-edge-intro-to-apache-minifi-141236290
- https://www.slideshare.net/Hadoop_Summit/apache-nifi-crash-course-131483547

TH^{} N^{} Y^{} U^{}