Stream Native

Real-Time Streaming in Azure

Timothy Spann | Developer Advocate

cloud
lunch & learn

Real-Time Streaming in Azure

with Timothy Spann

13 October 2021

18:00 (GMT+1)

www.cloudlunchlearn.com

In collaboration with CloudFamily Microsoft

StreamNative

Tim Spann

Developer Advocate

DZone Zone Leader and Big Data MVB Data DJay

- https://www.datainmotion.dev/
- https://github.com/tspannhw/SpeakerProfile
- https://dev.to/tspannhw
- https://sessionize.com/tspann/

StreamNative

StreamNative

Founded by the original developers of Apache Pulsar and Apache BookKeeper, StreamNative builds a cloud-native event streaming platform that enables enterprises to easily access data as real-time event streams.

Apache Pulsar

StreamNative

Apache  is an open source, cloud-native distributed messaging and streaming platform.

What are the Benefits of Pulsar?

Multi-Tenancy

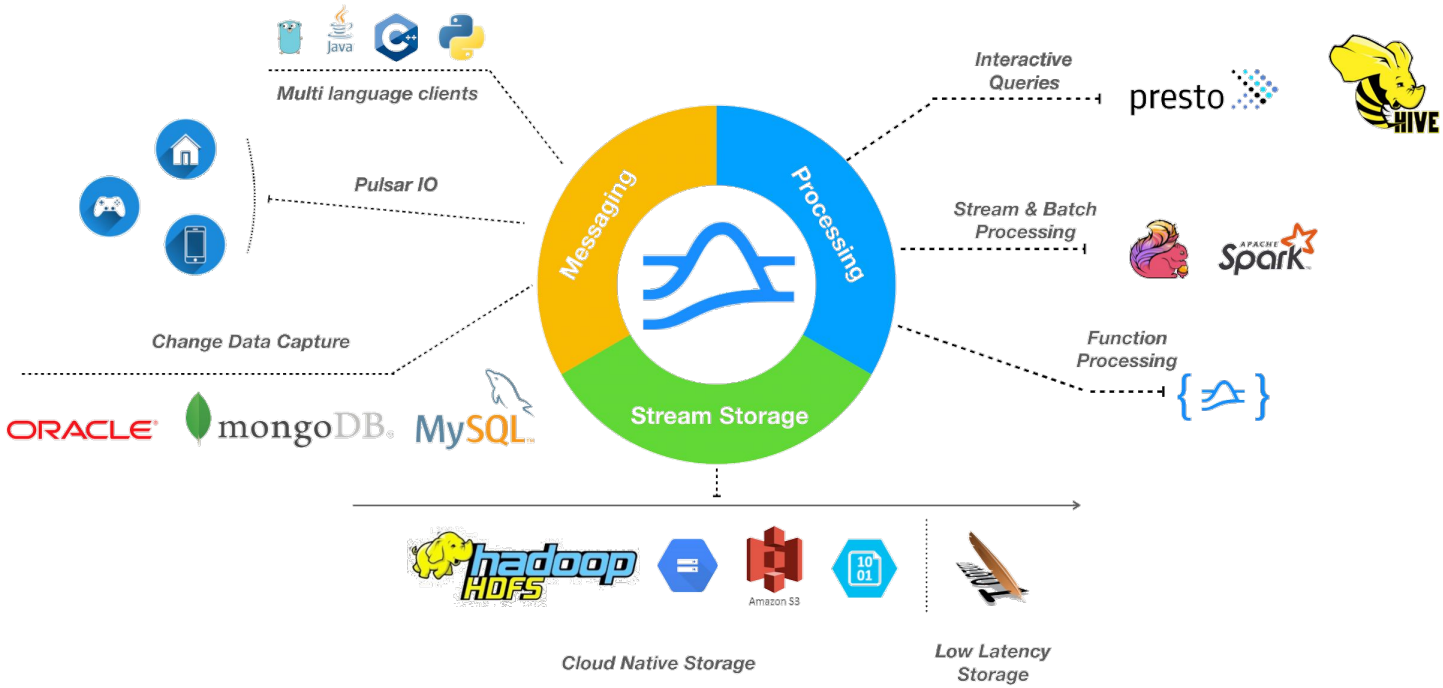
Scalability

Geo-Replication

Unified Messaging Model

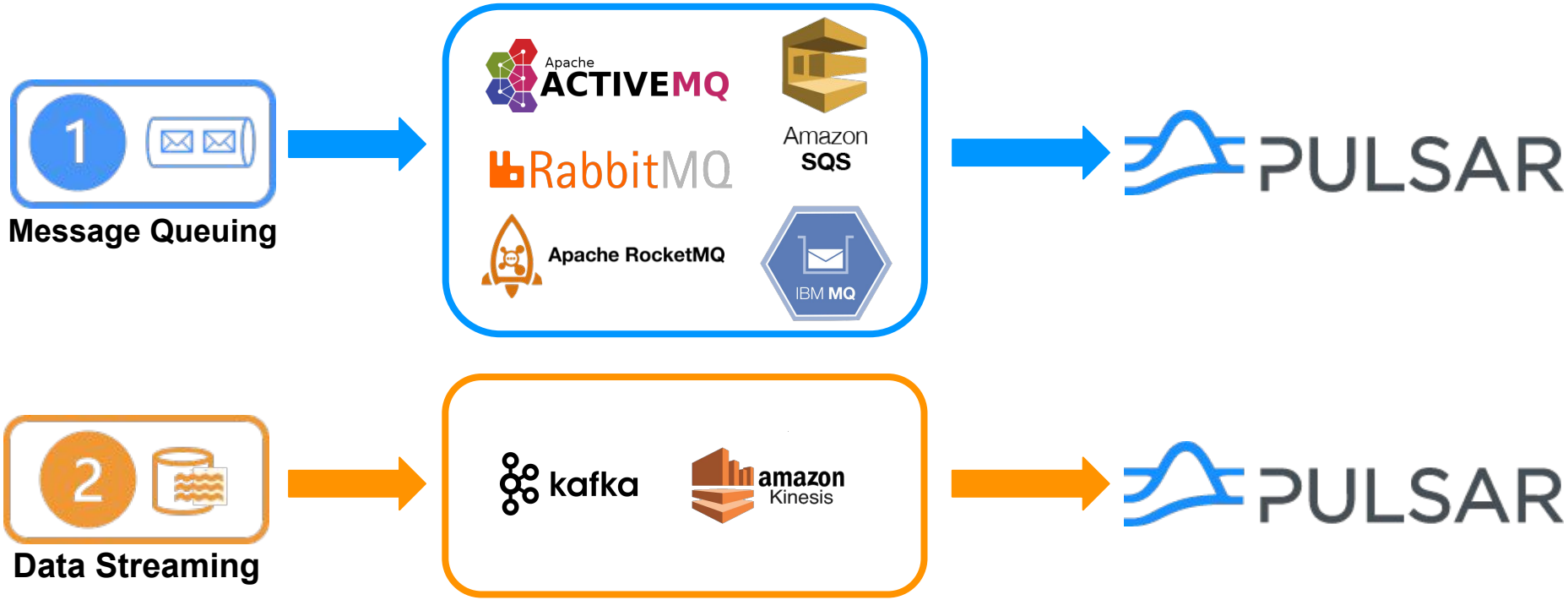
Data Durability

Apache Pulsar



StreamNative

A Unified Messaging Platform



StreamNative

The Difference Between Messaging & Streaming

Messaging

Messaging systems are ideal for work queues that do not require tasks to be performed in a particular order—for example, sending one email message to many recipients.

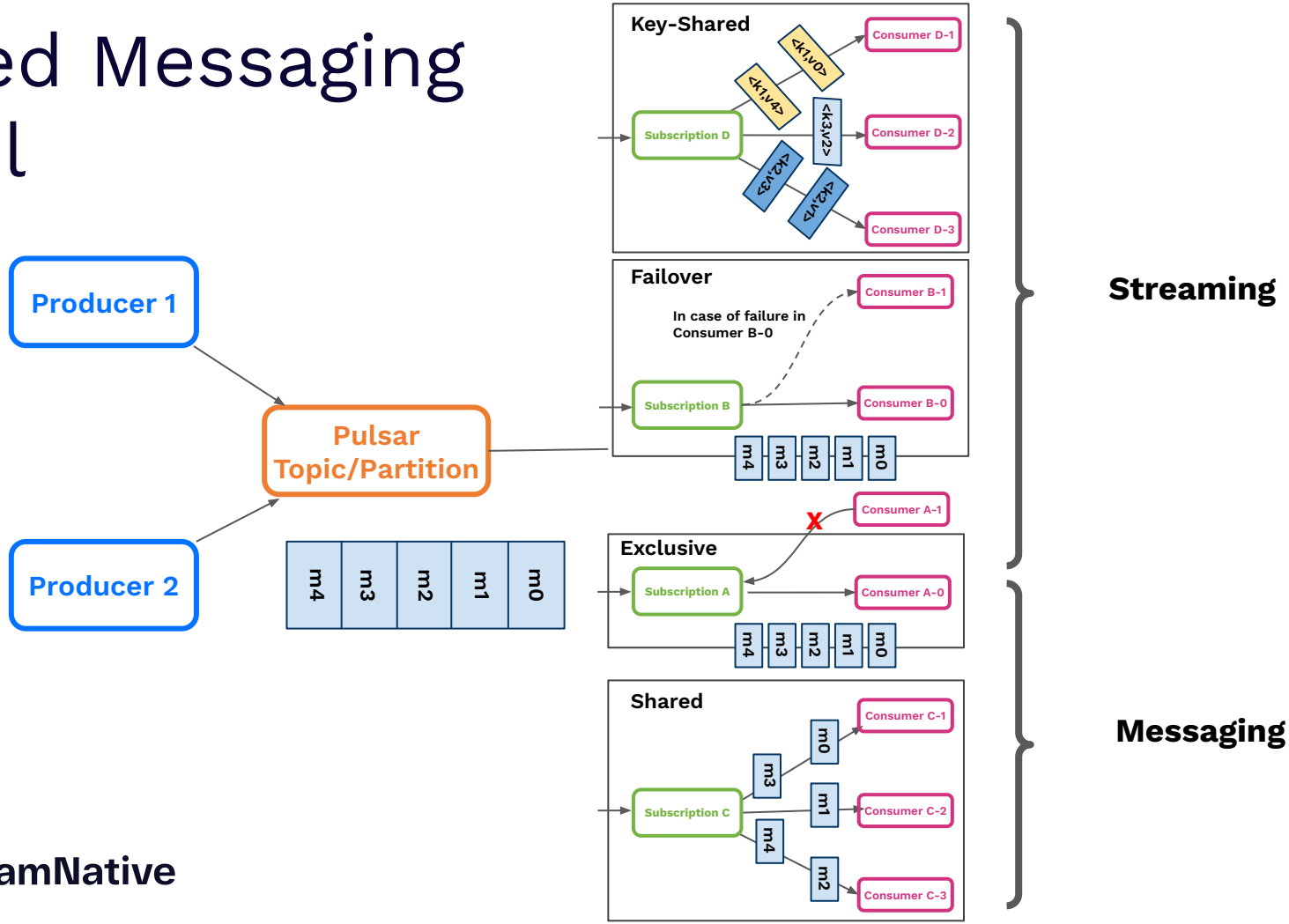
RabbitMQ and Amazon SQS are examples of popular queue-based message systems.

Streaming

Streaming works best in situations where the order of messages is important—for example, data ingestion.

Kafka and Amazon Kinesis are examples of messaging systems that use streaming semantics for consuming messages.

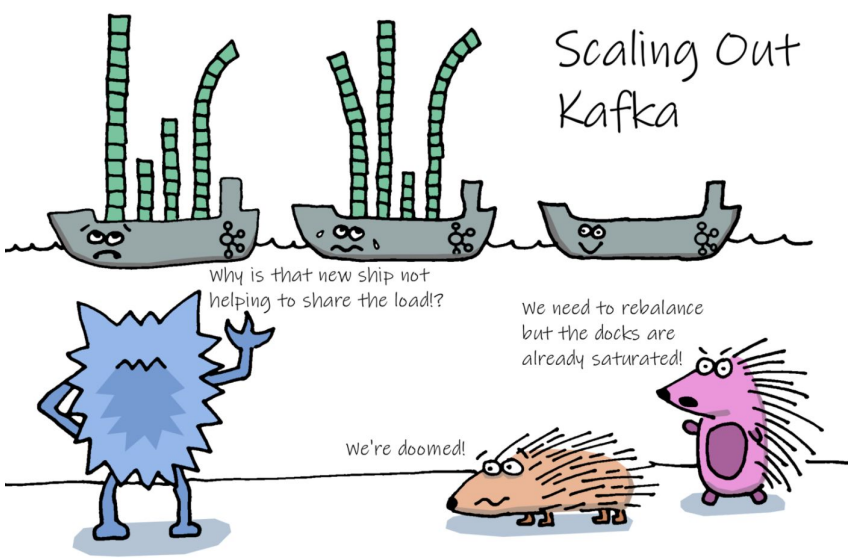
Unified Messaging Model

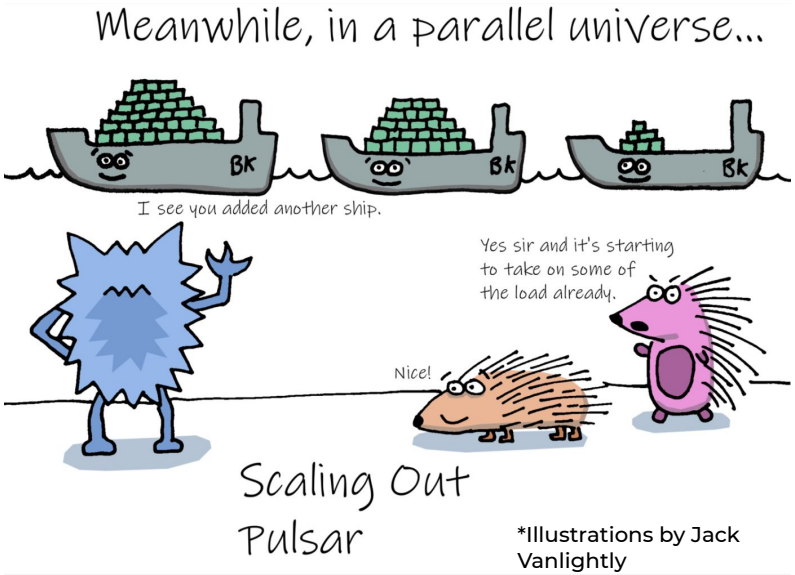


StreamNative

Pulsar is Cloud Native

- Rebalance free architecture via separate compute and storage for easy scalability
- Built for Kubernetes, with production ready helm charts and Kubernetes operators
- Infinite streams by leveraging cost effective cloud-storage like S3, GCS, etc
- Supports a broad range of workloads and teams through unified messaging model



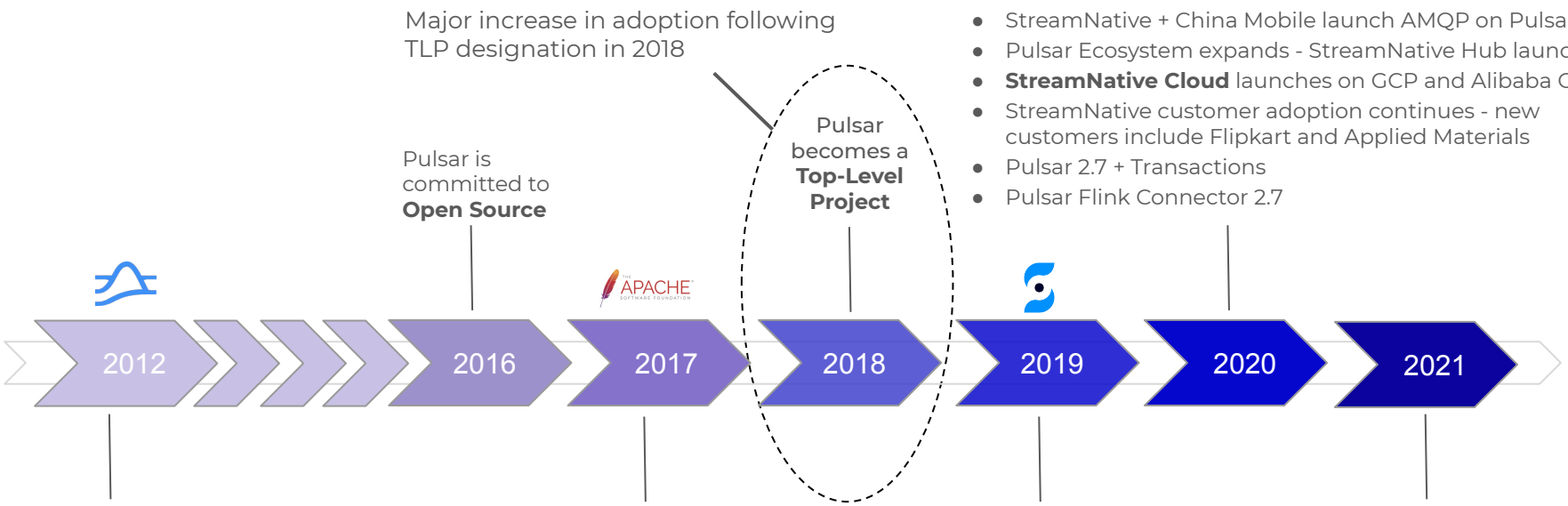


Pulsar is built for easy scale-out.

StreamNative

Pulsar Global Adoption

Key Milestones

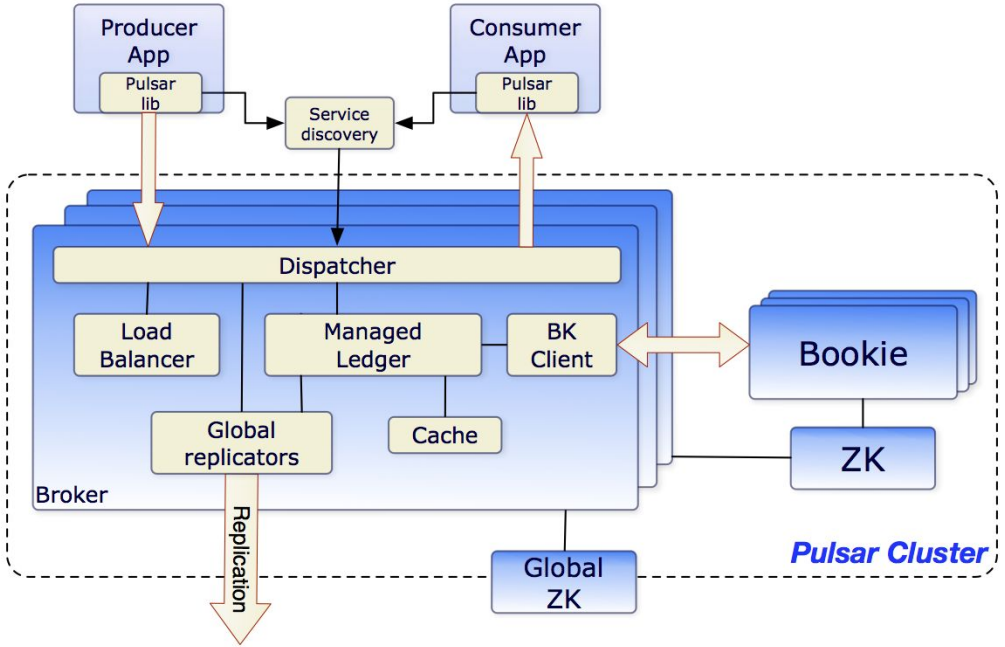


StreamNative

Apache Pulsar Overview

Enable Geo-Replicated Messaging

- Pub-Sub
- Geo-Replication
- Pulsar Functions
- Horizontal Scalability
- Multi-tenancy
- Tiered Persistent Storage
- Pulsar Connectors
- REST API
- CLI
- Many clients available
- Four Different Subscription Types
- Multi-Protocol Support
  - MQTT
  - AMQP
  - JMS
  - Kafka
  - ...



StreamNative

What is the Pulsar Ecosystem?

- **Functions and Connectors**
  - Functions: Lightweight stream processing
  - Connectors: Part of “Pulsar IO”, includes “Source” and “Sink” APIs
    - Files, Databases, Data tools, Cloud Services, etc
- **Protocol Handlers**
  - Allows Pulsar to handle additional protocols by an extendable API running in the broker
    - AoP (AMQP), KoP (Kafka), MoP (MQTT)

What is the Pulsar Ecosystem? (cont'd)

- **Processing Engines**
  - Supports modern processing engines
    - Flink and Spark, as well as Pulsar SQL (Presto/Trino)
- **Offloaders**
  - Allows data to be offloaded to cloud storage and used with existing Pulsar APIs
    - S3, GCP Cloud Storage, HDFS, File (NFS), Azure Blob Storage (in Pulsar 2.7.0)

Pulsar Functions

Provides a simple API to:

- Receive a message (consume)
- Process the message using your own code
- Send a message (produce)

Takes care of the boilerplate code so there is no need to create producers and consumers.

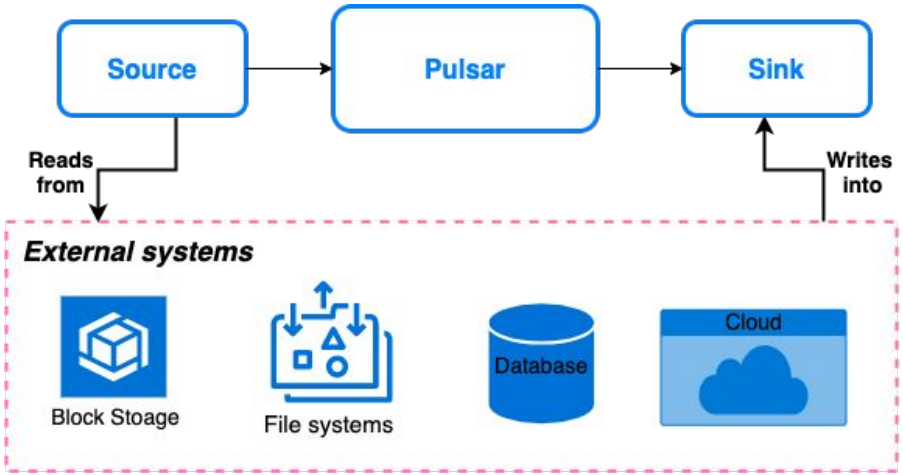
Moving Data In and Out of Pulsar

IO/Connectors are a simple way to integrate with external systems and move data in and out of Pulsar.

- Built on top of Pulsar Functions
- Built-in connectors - hub.streamnative.io

Use Azure BlobStore offloader with Pulsar

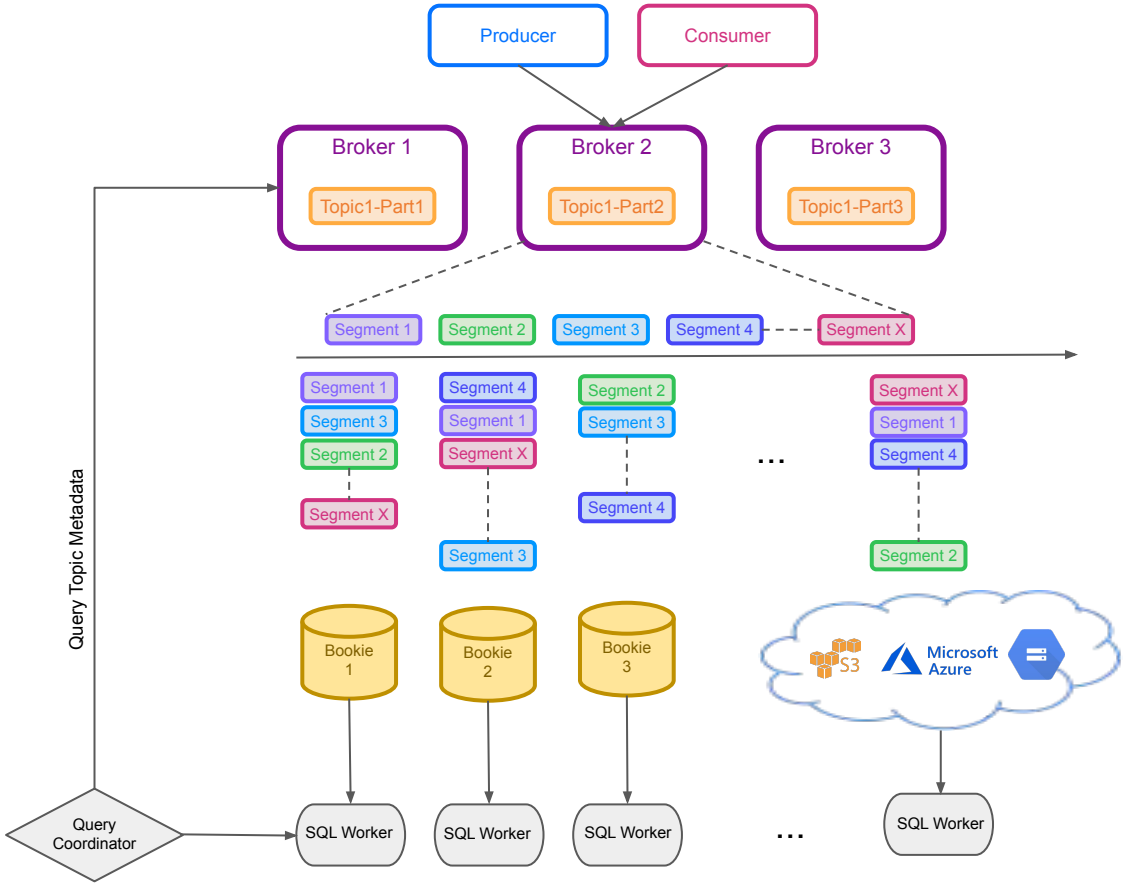
https://pulsar.apache.org/docs/en/tiered-storage-azure/



StreamNative

Pulsar SQL

Presto/Trino workers can read segments directly from bookies (or offloaded storage) in parallel.



trino

StreamNative

Query Your Topics with Pulsar SQL (Trino)

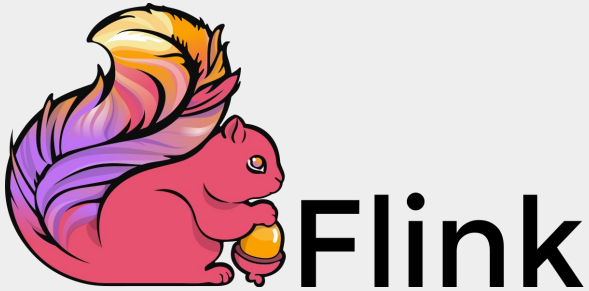
```
presto> select camera, cpu, cputempf, gputempf, memory, top1, top1pct, uuid, __publish_time__, __message_id__, __key__ from  pulsar."public/default".iotjetsonjson;
  camera    | cpu  | cputempf | gputempf | memory |       top1       |    top1pct     |                uuid                |    __publish_time__     | __message_id__ |
------------+------+----------+----------+--------+------------------+----------------+------------------------------------+-------------------------+----------------+
 /dev/video0 |  8.7 | 82       | 82       |   33.5 | microphone, mike | 18.85986328125 | xav_uuid_video0_lgl_20211001183019 | 2021-10-01 14:30:30.657 | (564,3,0)      |
 /dev/video0 |  8.7 | 82       | 82       |   33.6 | microphone, mike | 19.22607421875 | xav_uuid_video0_kpt_20211001183033 | 2021-10-01 14:30:44.380 | (564,4,0)      |
 /dev/video0 | 12.0 | 80       | 81       |   33.5 | microphone, mike | 12.53662109375 | xav_uuid_video0_gzd_20211001182930 | 2021-10-01 14:29:48.756 | (564,0,0)      |
 /dev/video0 |  8.5 | 82       | 82       |   33.6 | microphone, mike |        14.0625 | xav_uuid_video0_wlw_20211001182951 | 2021-10-01 14:30:02.919 | (564,1,0)      |
 /dev/video0 |  8.5 | 82       | 82       |   33.5 | microphone, mike |     29.8828125 | xav_uuid_video0_ulq_20211001183005 | 2021-10-01 14:30:16.787 | (564,2,0)      |
(5 rows)
(END)
```

```
presto> show tables in pulsar."public/default";
     Table
----------------
 generator_test
 iotjetsonjson
 mqtt-2
(3 rows)

Query 20211001_054538_00008_s8x23, FINISHED, 1 node
Splits: 19 total, 19 done (100.00%)
0:00 [3 rows, 105B] [14 rows/s, 493B/s]
```

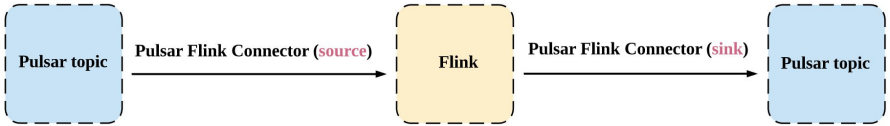
StreamNative

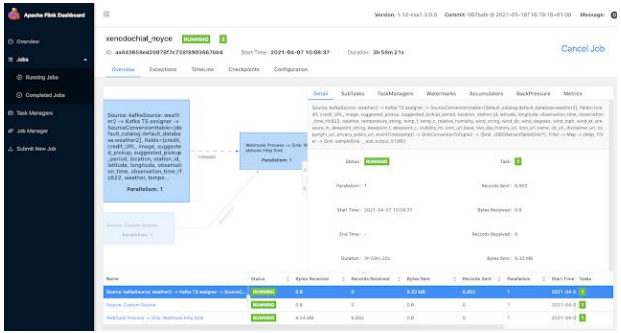
Apache Flink



StreamNative

Why Apache Flink?



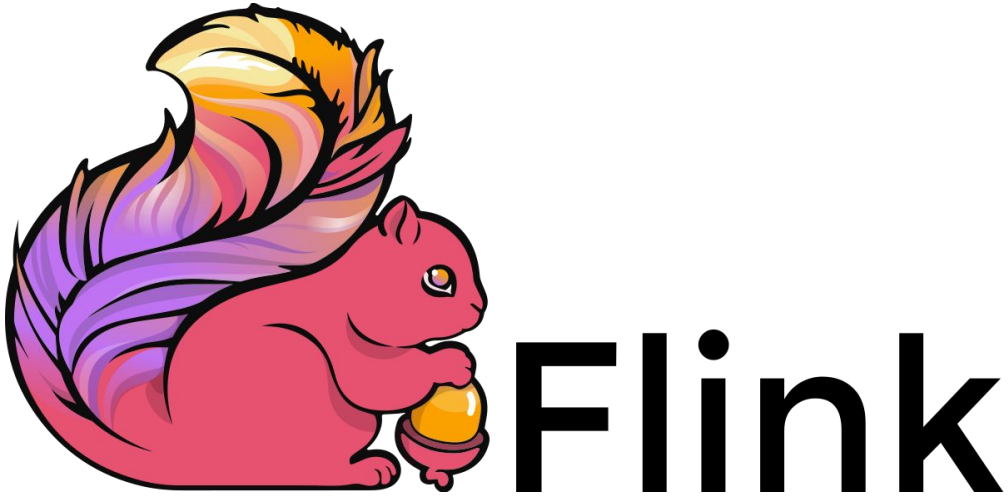


- Unified computing engine
- Batch processing is a special case of stream processing
- Stateful processing
- Massive Scalability
- Flink SQL for queries, inserts against Pulsar Topics
- Streaming Analytics
- Continuous SQL
- Continuous ETL
- Complex Event Processing
- Standard SQL Powered by Apache Calcite

StreamNative

Apache Flink

- Apache Flink is a distributed stream processing system.
- It is capable of providing high throughput, near real-time processing of streams from Pulsar.
- It is ideal for *ambitious* Stream Processing compared to Pulsar's model of lightweight Stream Processing.

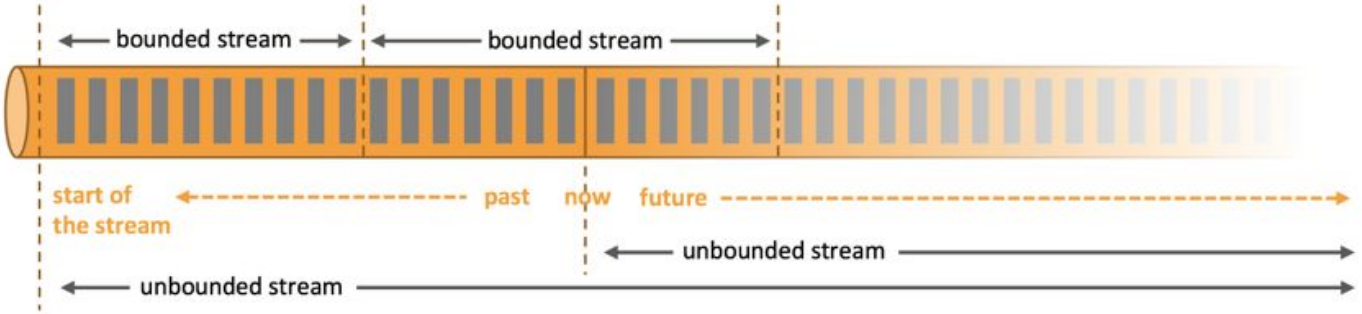


Some Apache Flink Users

StreamNative

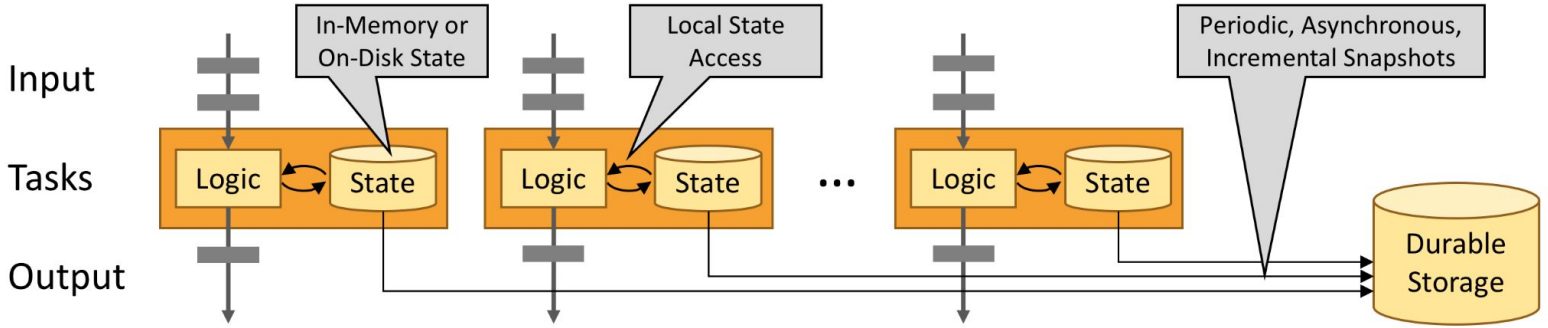
Everything Is a Stream

- Stream: Sequence of data which is made available over time
- All computation processes chunks of data over time producing results over time → Stream processing
  - E.g. reading data from disks is done in streaming fashion
- Events can be of various forms
- Decisive difference: Is my stream bounded or not?



How Flink Works

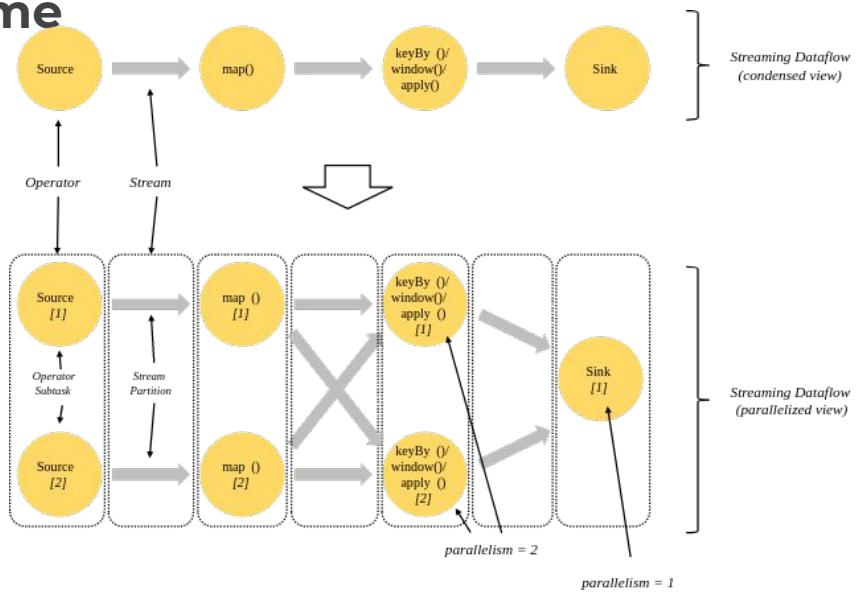
Flink Provides an API for **handling events** and **intermediate state and periodic snapshots** to ensure consistency



StreamNative

How Flink Works

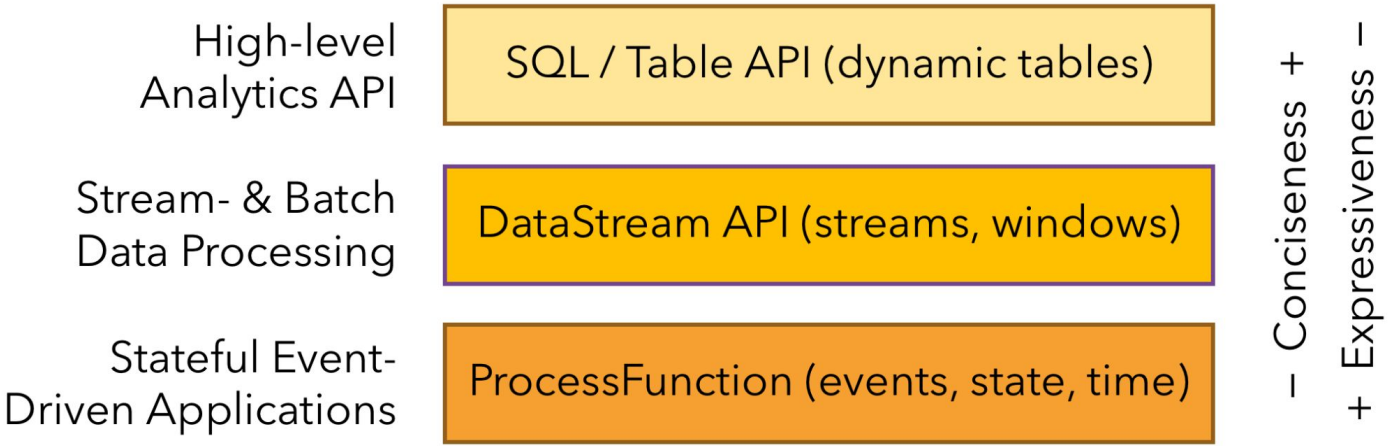
Flink apps are composed of a **graph of tasks** which is executed in a **distributed runtime**



StreamNative

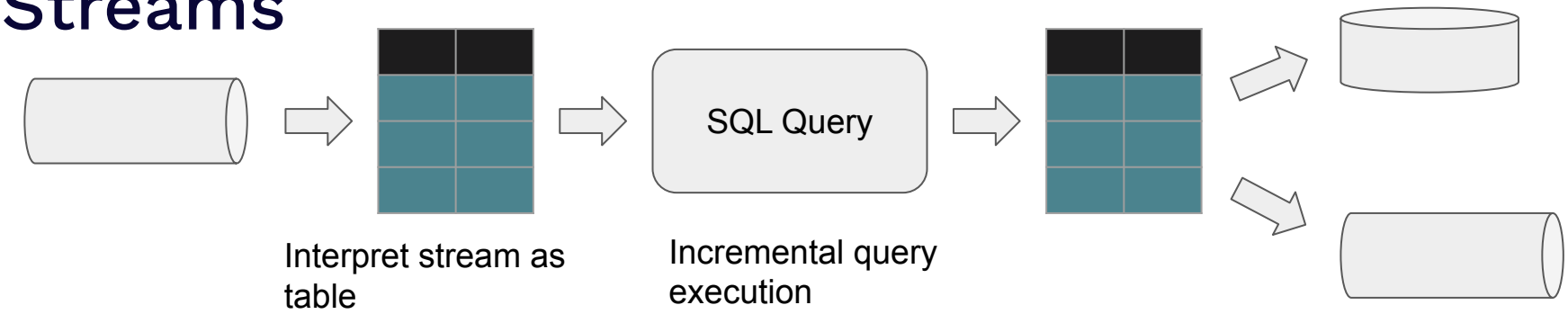
How Flink Works

Flink provides **multiple APIs at different layers of abstraction**



StreamNative

SQL / Table API: Running The Same Query On Streams




```
SELECT
      room,
      TUMBLE_END(rowtime, INTERVAL ‘1’ HOUR),
      AVG(temperature)
FROM
      sensors
GROUP BY
      TUMBLE(rowtime, INTERVAL ‘1’ HOUR), room
```

StreamNative Cloud

StreamNative Cloud

Powered by Apache Pulsar, StreamNative provides a cloud-native, real-time messaging and streaming platform to support multi-cloud and hybrid cloud strategies.

Cloud Native

Built for Containers

Flink SQL

StreamNative

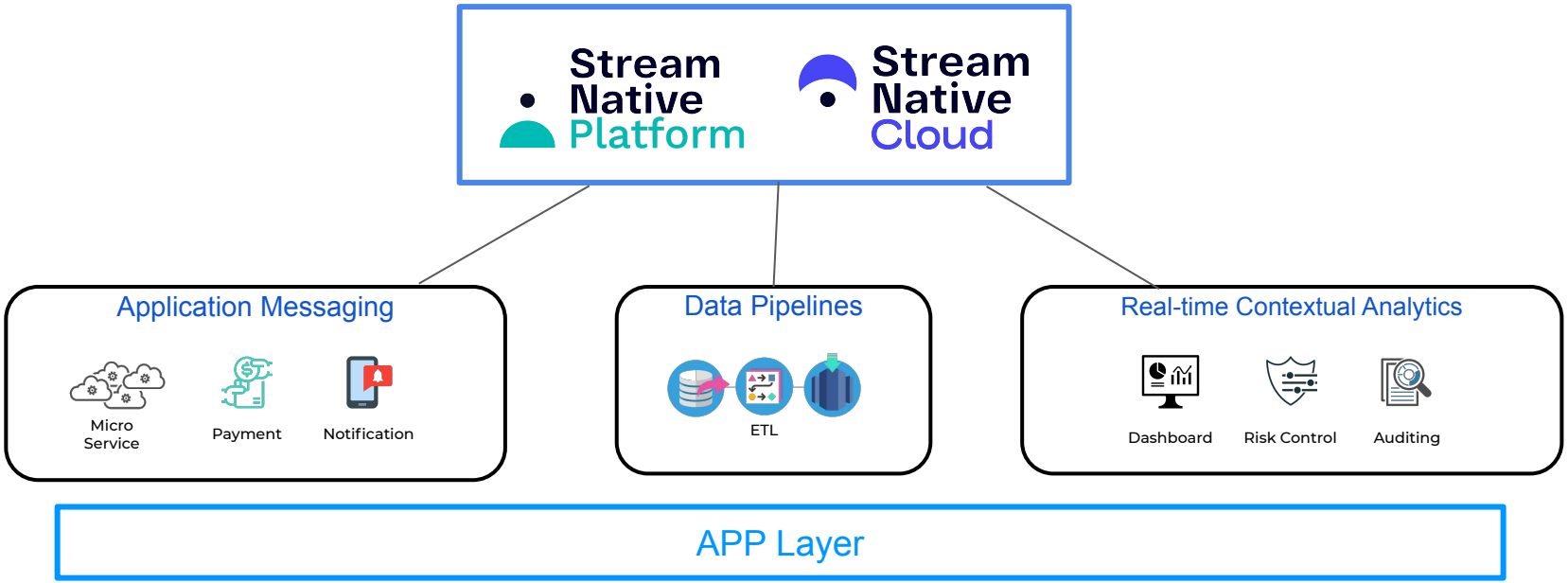
StreamNative

Powered by Apache Pulsar, StreamNative provides a cloud-native, unified messaging and streaming platform to support multi-cloud and hybrid cloud strategies.

Cloud Native

Built for Containers

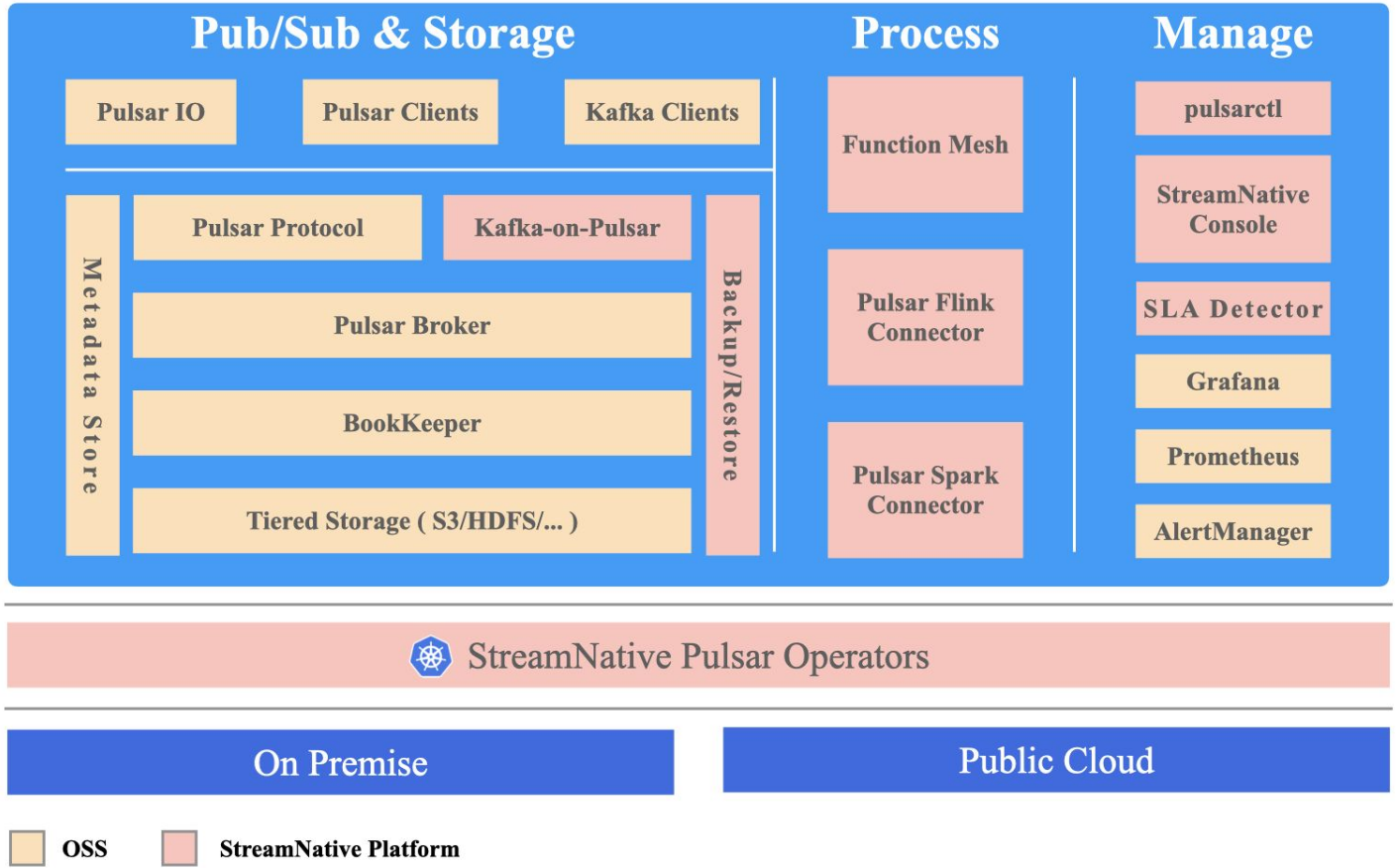
The StreamNative Offering



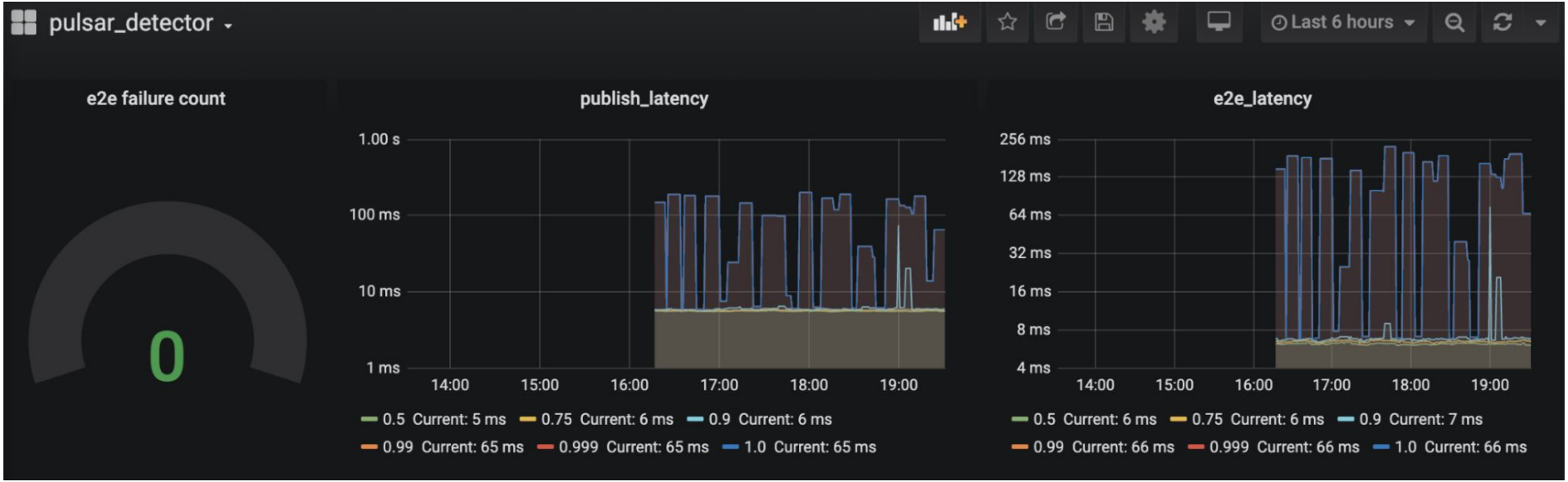
StreamNative

StreamNative Solution

StreamNative

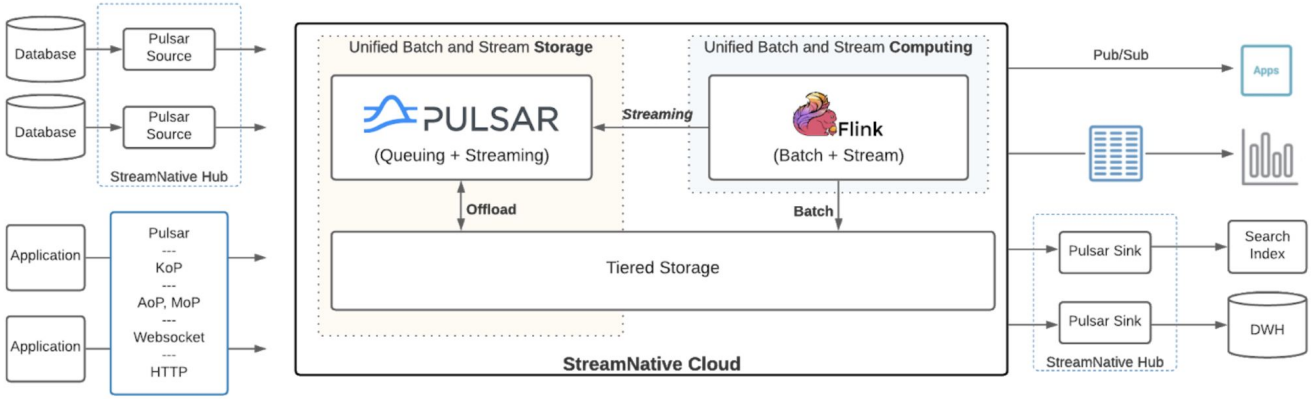


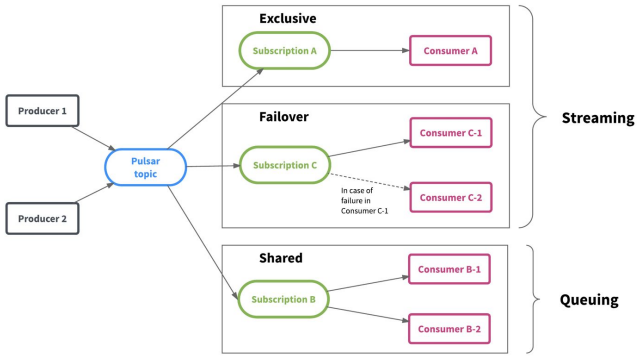
StreamNative

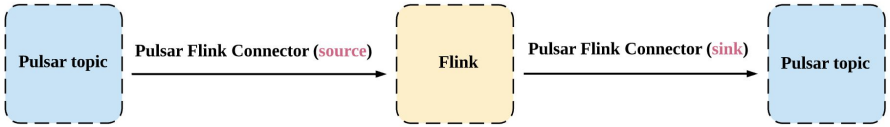


StreamNative

Flink + Pulsar







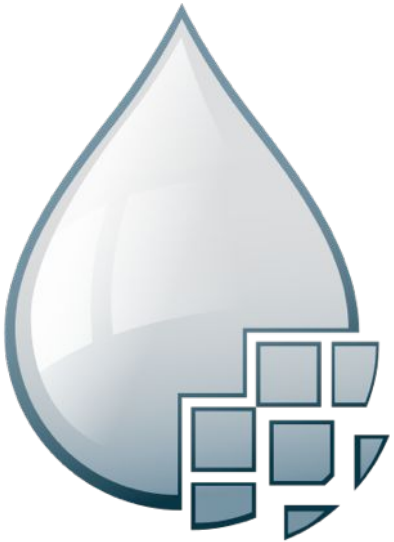
https://flink.apache.org/2019/05/03/pulsar-flink.html
https://github.com/streamnative/pulsar-flink
https://streamnative.io/en/blog/release/2021-04-20-flink-sql-on-streamnative-cloud

StreamNative

Apache NiFi

StreamNative

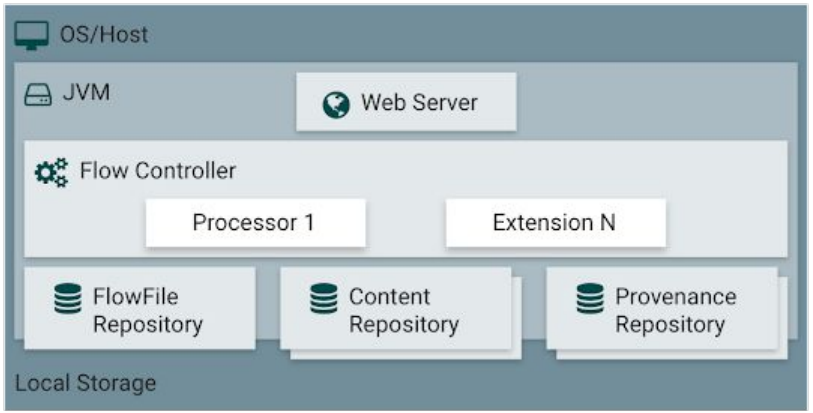
Why Apache NiFi?

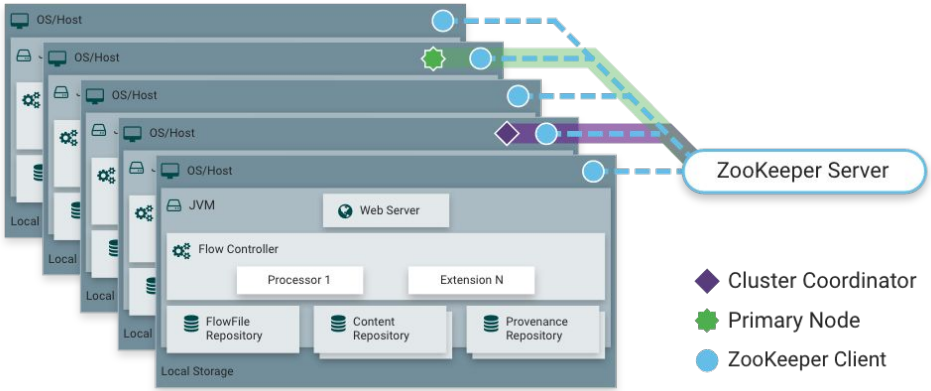


- Guaranteed delivery
- Data buffering
  - Backpressure
  - Pressure release
- Prioritized queuing
- Flow specific QoS
  - Latency vs. throughput
  - Loss tolerance
- Data provenance
- Supports push and pull models

- Hundreds of processors
- Visual command and control
- Over a sixty sources
- Flow templates
- Pluggable/multi-role security
- Designed for extension
- Clustering
- Version Control

Architecture





https://nifi.apache.org/docs/nifi-docs/html/overview.html

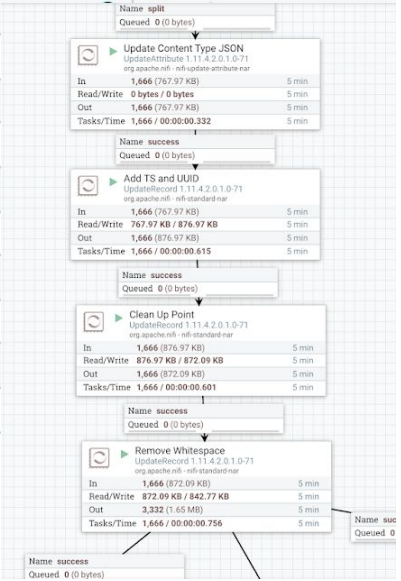
StreamNative

Record Processors

- XML, CSV, JSON, AVRO and more
- Schemas or Inferred Schemas
- Easily convert between them
- Support SQL with Apache Calcite

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

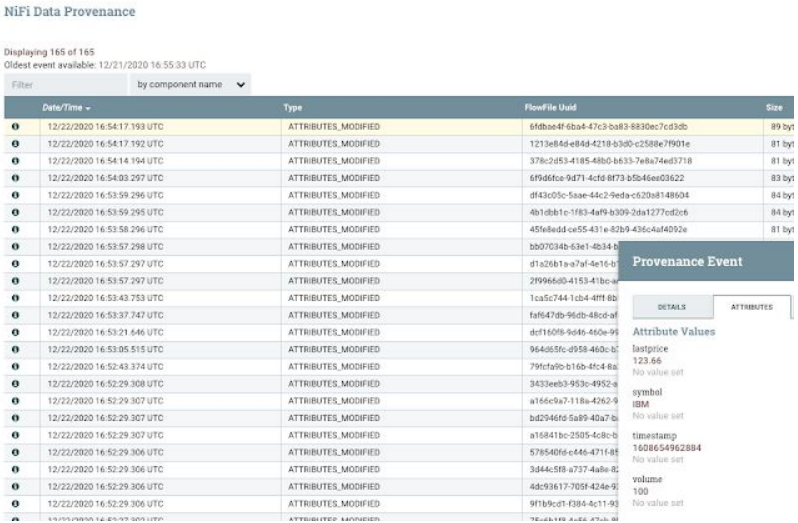
Property	Value
Schema Access Strategy	**Infer Schema**
Schema Registry	AvroSchemaRegistry
Schema Name	${schema.name}
Schema Version	
Schema Branch	
Schema Text	
Schema Inference Cache	
Expect Records as Array	
Attribute Prefix	
Field Name for Content	
Date Format	
Time Format	
Timestamp Format	



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

StreamNative

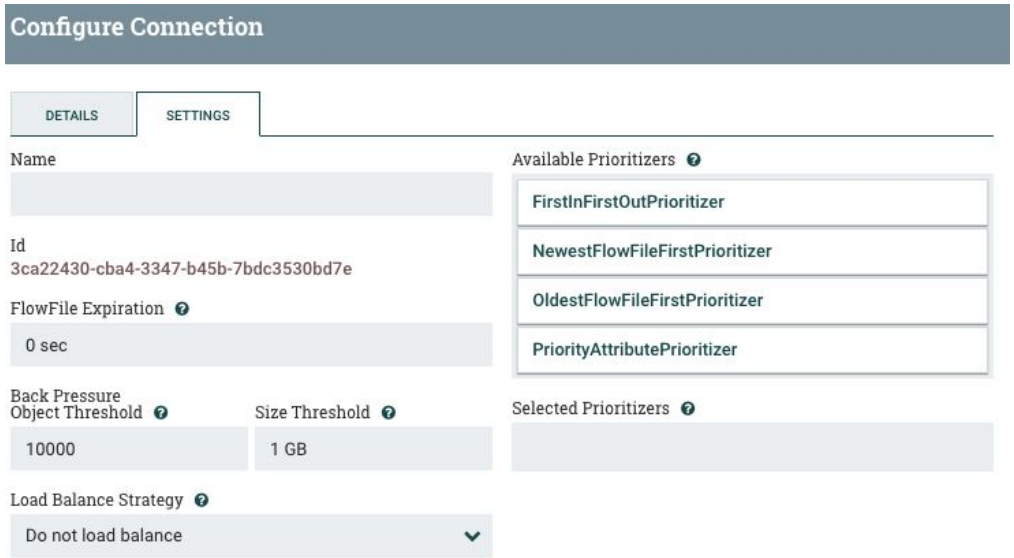
Provenance



https://www.datainmotion.dev/2021/01/automating-starting-services-in-apache.html

StreamNative

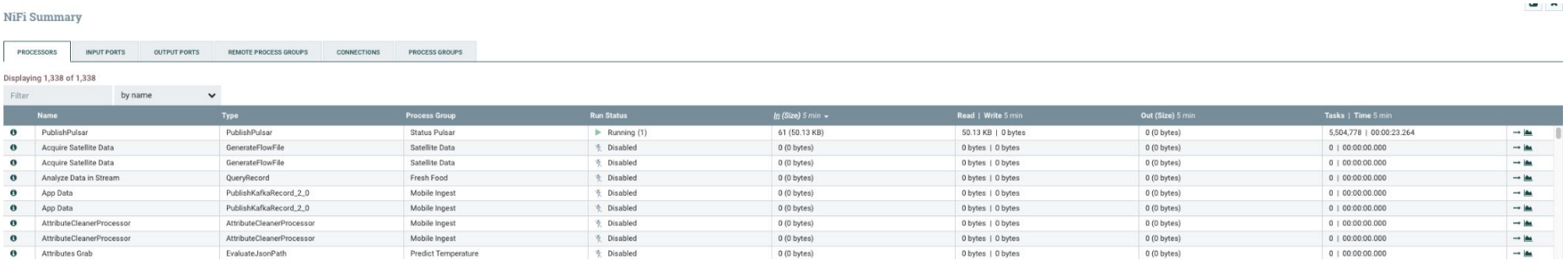
Backpressure & Prioritizers



https://www.datainmotion.dev/2019/11/exploring-apache-nifi-110-parameters.html

StreamNative

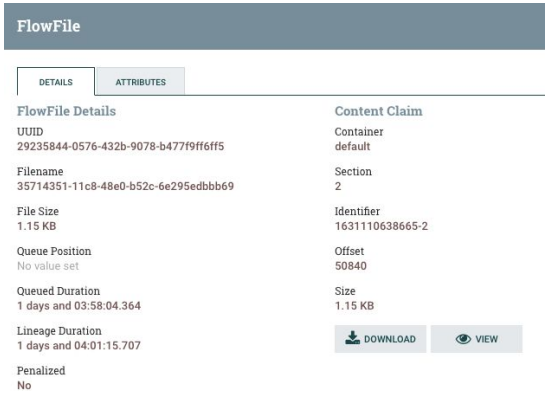
System Diagnostics

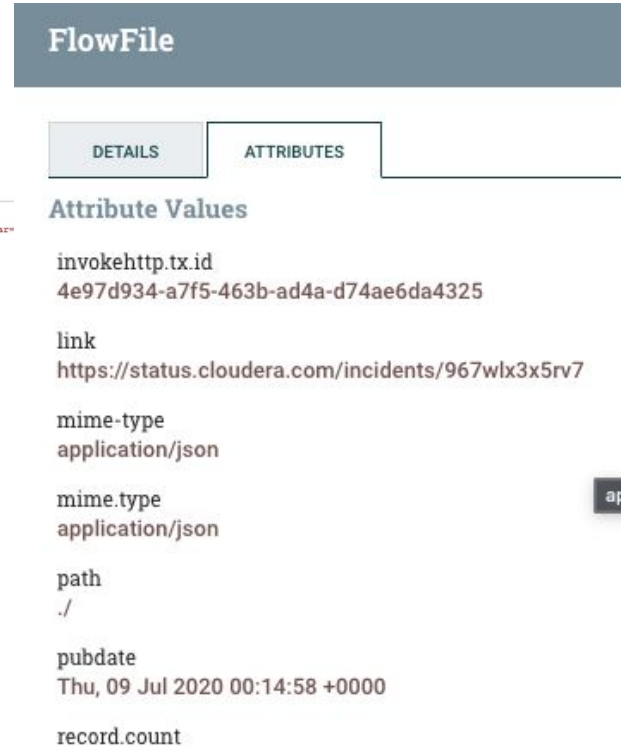


StreamNative

Flow File

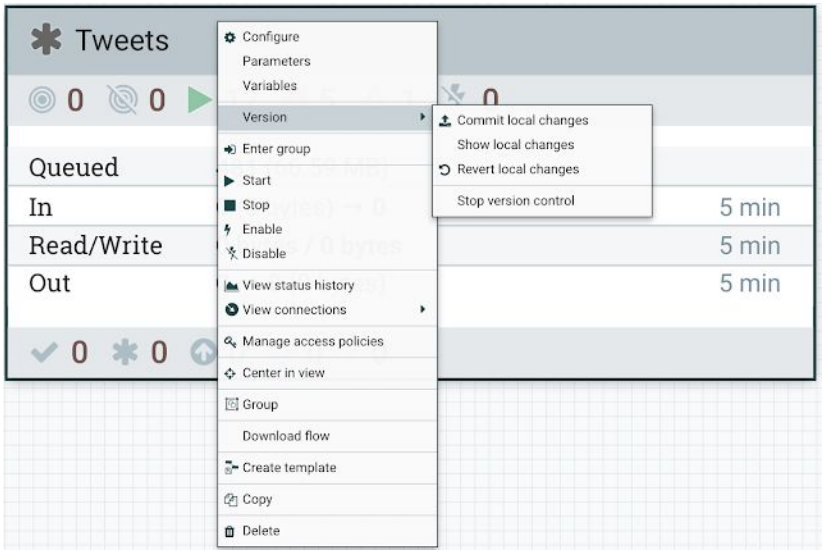
Flow Files are content and key/value pairs for attributes that are each event/message/file that has been introduced into NiFi.

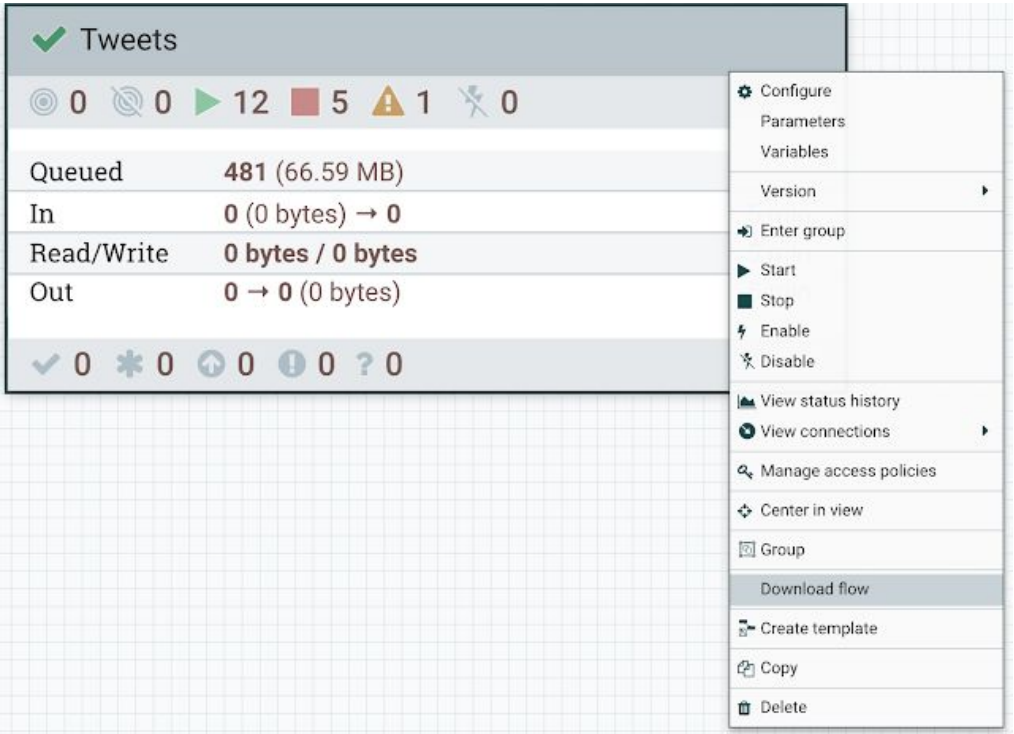




Stre https://nifi.apache.org/docs/nifi-docs/html/overview.html

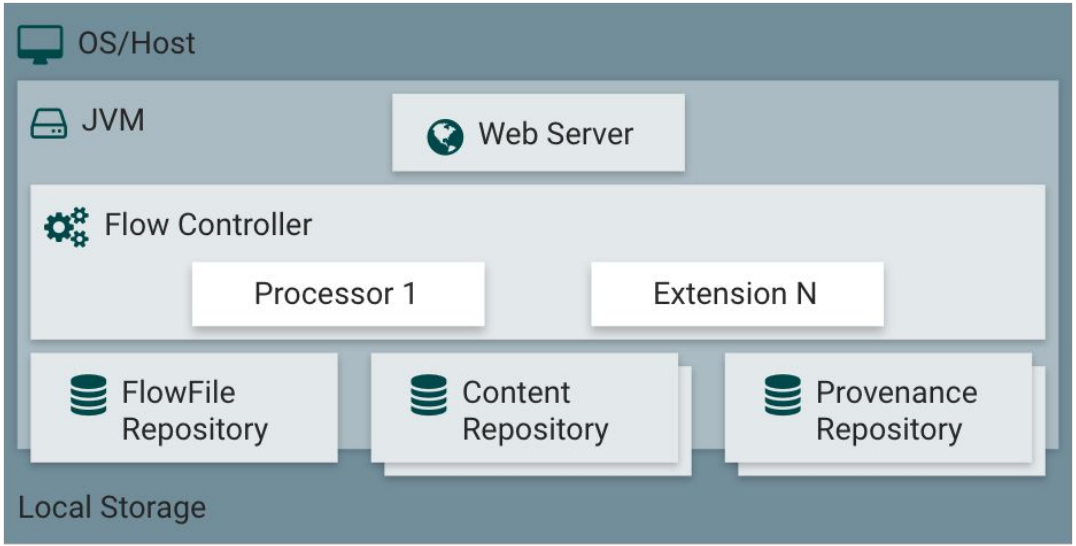
Version Control (Github and Beyond)





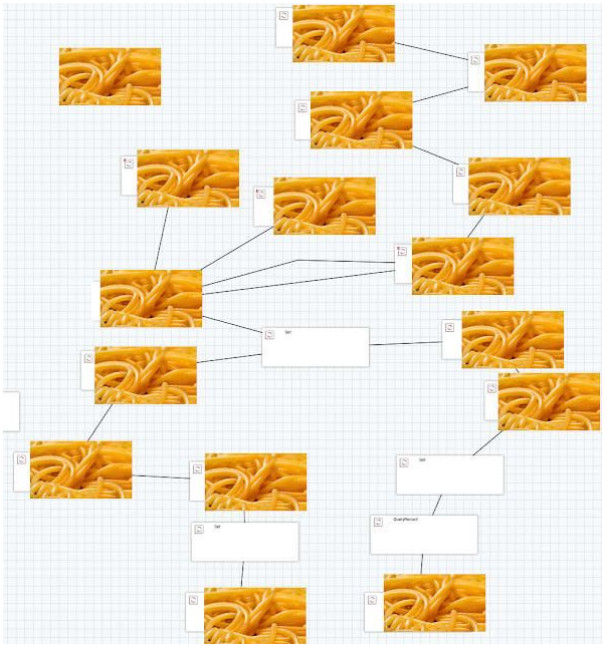
StreamNative

Repositories



https://nifi.apache.org/docs/nifi-docs/html/nifi-in-depth.html#repositories

No More Spaghetti Flows - DO NOT

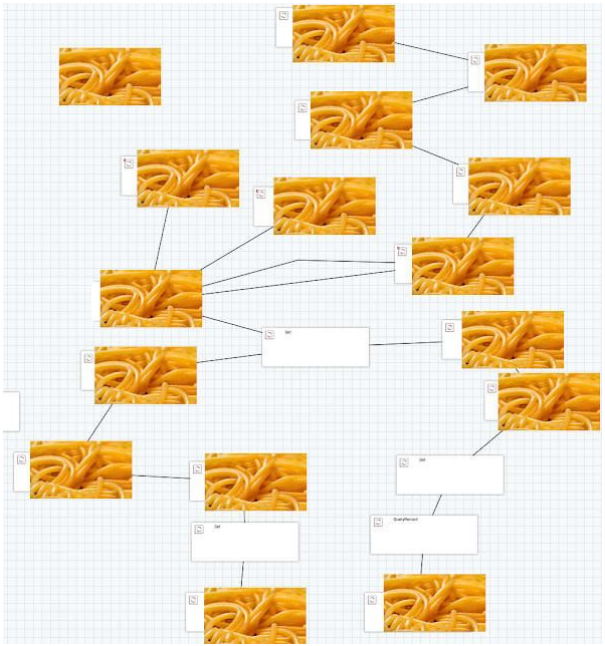


https://dev.to/tspannhw/no-more-spaghetti-flows-2emd

Do Not

- Do not Put 1,000 Flows on one workspace.
- If your flow has hundreds of steps, this is a Flow Smell. Investigate why.
- Do not Use ExecuteProcess, ExecuteScripts or a lot of Groovy scripts as a default, look for existing processors
- Do not Use Random Custom Processors you find that have no documentation or are unknown.
- Do not forget to upgrade, if you are running anything before Apache NiFi 1.14, upgrade now!
- Do not run on default 512M RAM.
- Do not run one node and think you have a highly available cluster.
- Do not split a file with millions of records to individual records in one shot without checking available space/memory and back pressure.
- Use Split processors only as an absolute last resort. Many processors are designed to work on FlowFiles that contain many records or many lines of text. Keeping the FlowFiles together instead of splitting them apart can often yield performance that is improved by 1-2 orders of magnitude.

No More Spaghetti Flows - DO

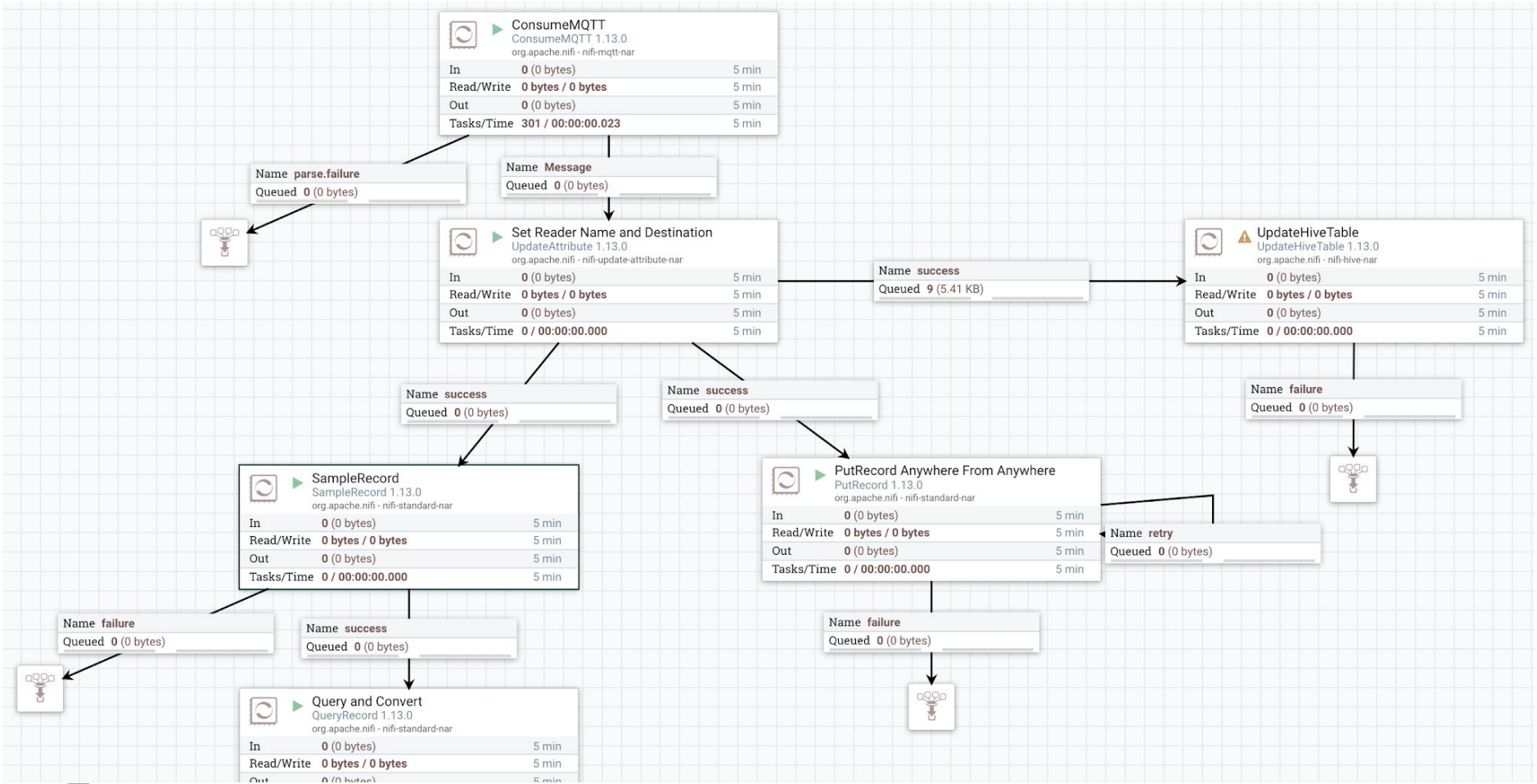


https://dev.to/tspannhw/no-more-spaghetti-flows-2emd

StreamNative

Do

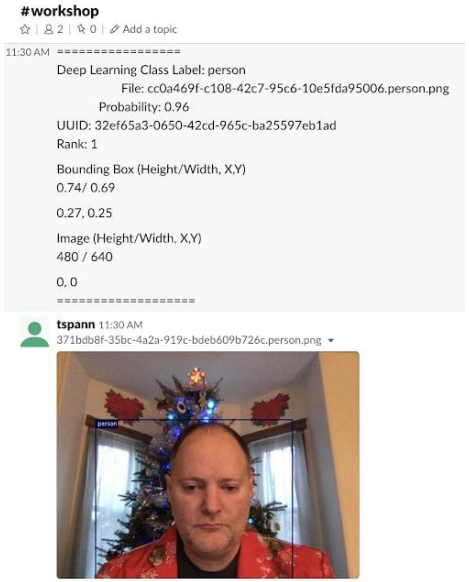
- Reduce, Reuse, Recycle. Use Parameters to reuse common modules.
- Put flows, reusable chunks (write to Slack, Database, Kafka) into separate Process Groups.
- Write custom processors if you need new or specialized features
- Use Record Processors everywhere
- Read the Docs!
- Use the NiFi Registry for version control.
- Use NiFi CLI and DevOps for Migrations.
- Walk through your flow and make sure you understand every step and it’s easy to read and follow. Is every processor used? Are there dead ends?
- Do run Zookeeper on different nodes from Apache NiFi.
- Use routing based on content and attributes to allow one flow to handle multiple nearly identical flows is better than deploying the same flow many times with tweaks to parameters in same cluster.
- Use the correct driver for your database. There's usually a couple different JDBC drivers.

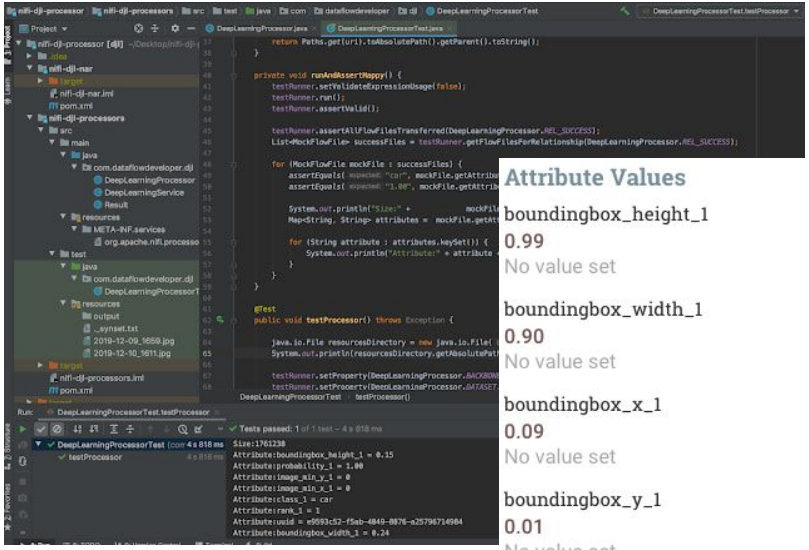


StreamNative

Apache MXNet Native Processor through DJL.AI for Apache NiFi

DJL





This processor uses the DJL.AI Java Interface

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

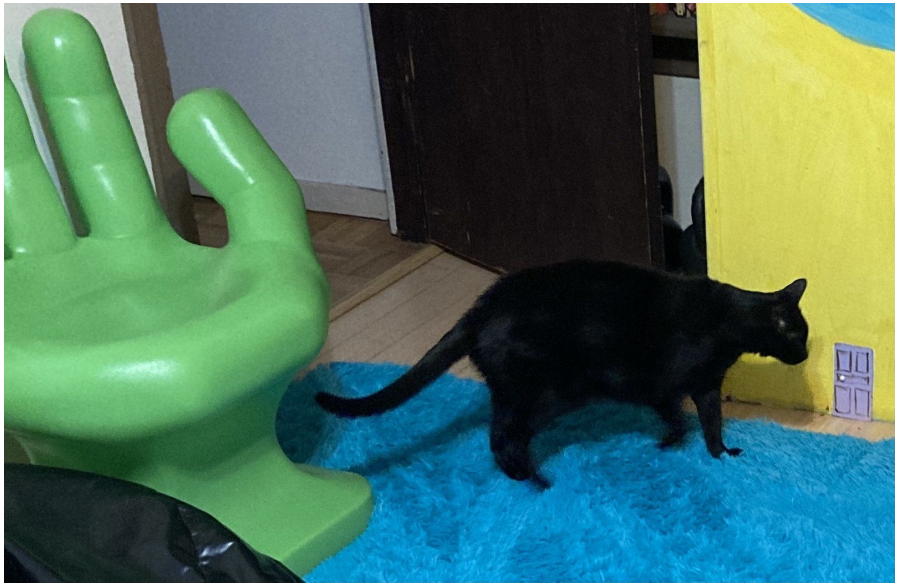
https://dev.to/tspannhw/easy-deep-learning-in-apache-nifi-with-djl-2d79

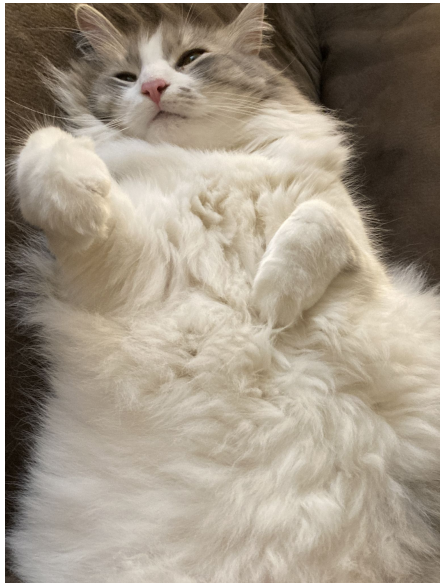
StreamNative

Use Cases

USE CASE

IoT Ingestion: High-volume streaming sources, multiple message formats, diverse protocols and multi-vendor devices creates data ingestion challenges.





StreamNative

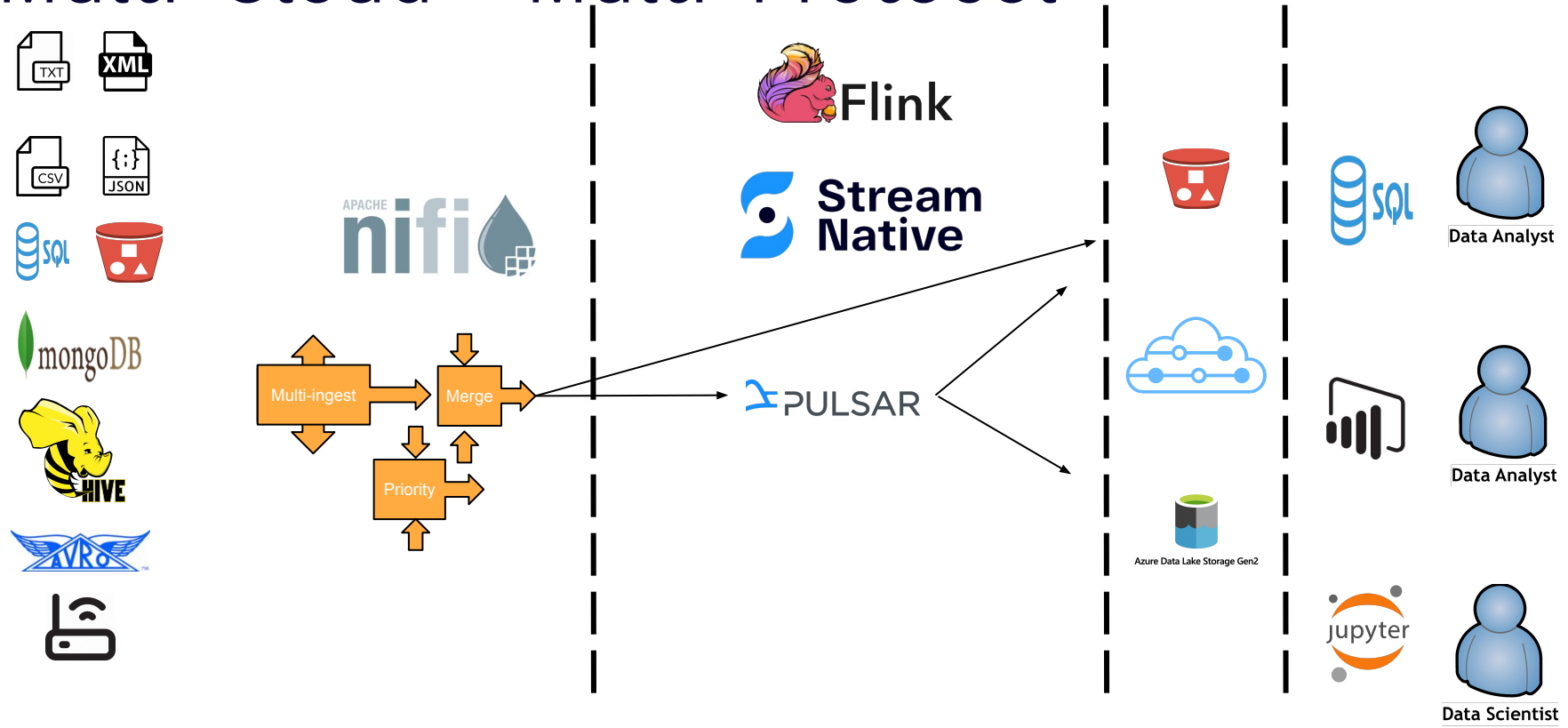
REST and Websocket JSON “stonks”

{"symbol":"MSFT",
"uuid":"10640832-f139-4b82-8780-e3ad37b3d0ce",
"ts":1618529574078,
"dt":1612098900000,
"datetime":"2021/01/31 08:15:00",
"open":"122.24500",
"close":"122.25500",
"high":"122.25500",
"volume":"12353",
"low":"12.24500"}

Best Practice Architectures

StreamNative

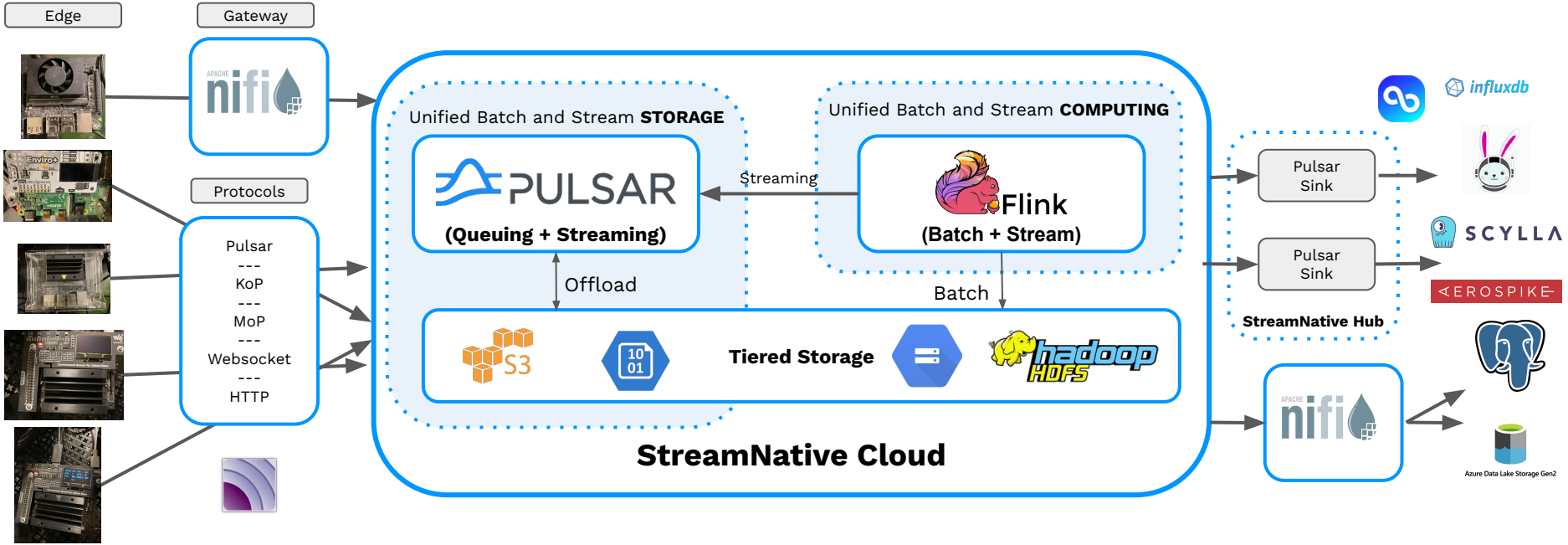
All Data - Anytime - Anywhere - Multi-Cloud - Multi-Protocol



StreamNative

End-to-End Streaming FLiP(N) Apps

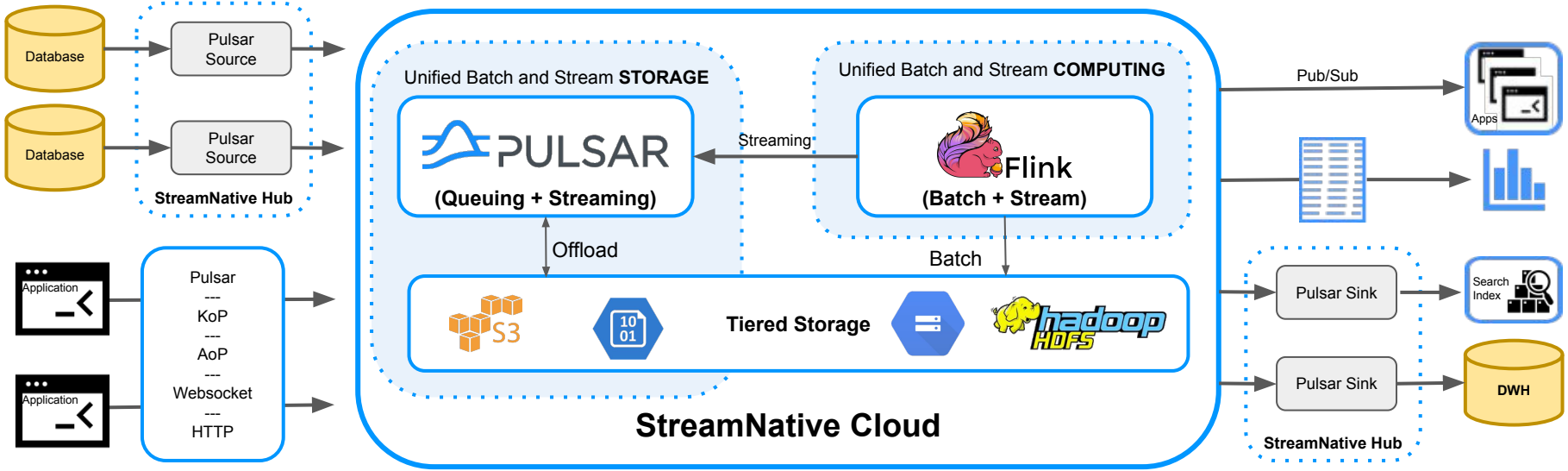
Apache Flink - Apache Pulsar - Apache NiFi <-> Events <-> Azure Data Stores



StreamNative

End-to-End Streaming Applications with Flink

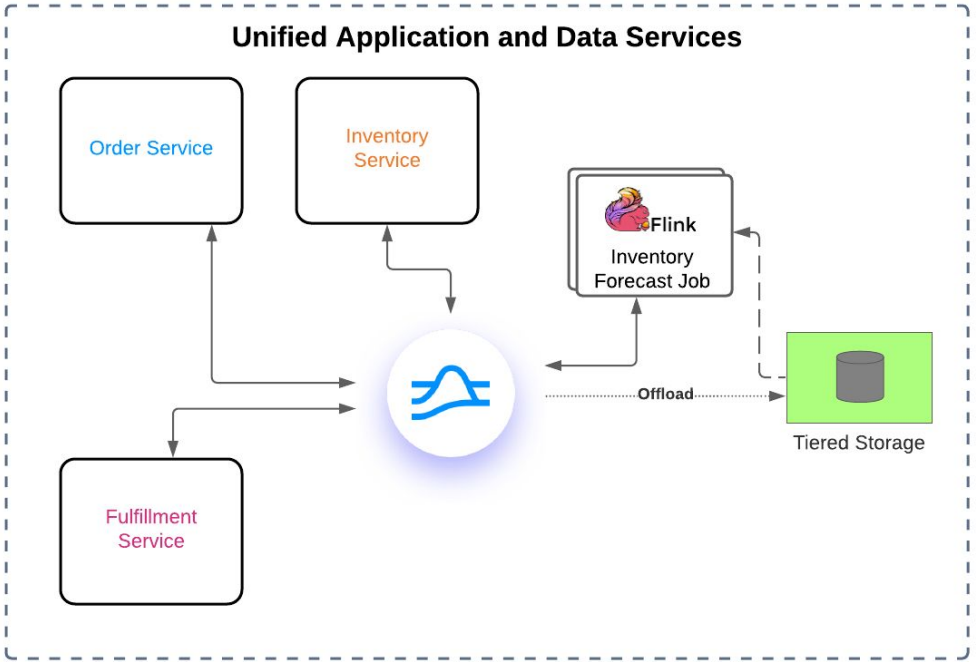
With Pulsar and Flink, StreamNative offers both **stream storage** and **stream compute** for a complete streaming solution.



StreamNative

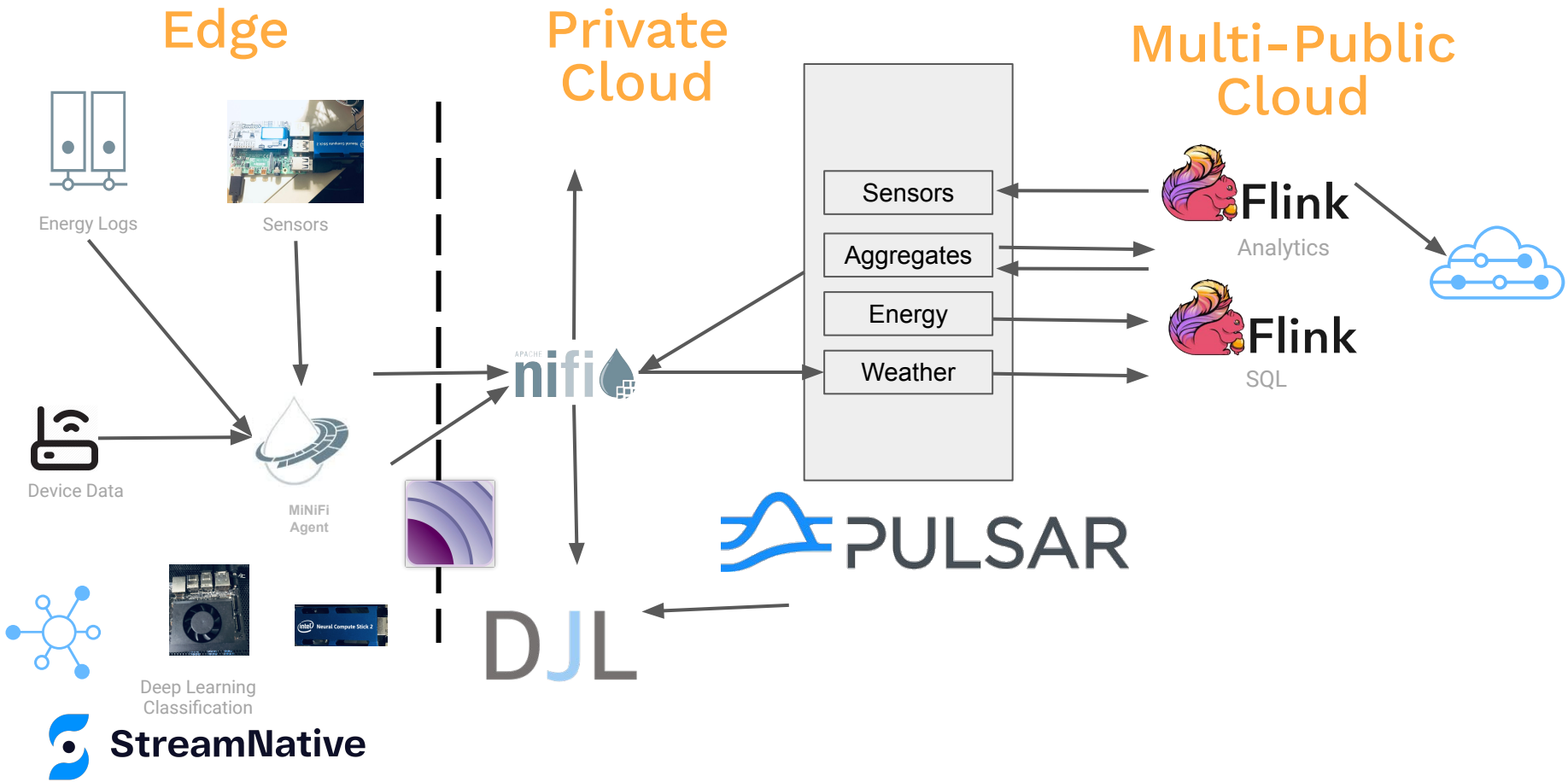
Example: E-Commerce with Pulsar

- Unified storage with access to underlying data
- Native tiered storage
- Single system to exchange data
- Teams share toolset



StreamNative

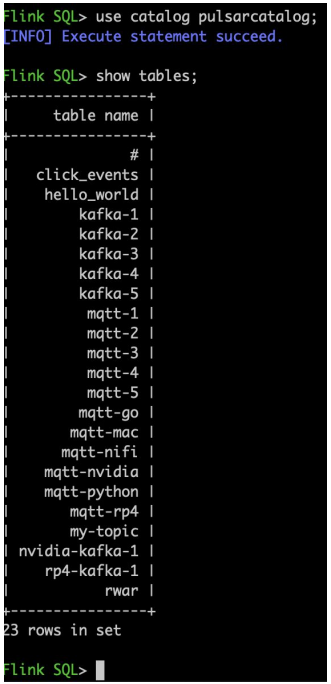
Edge AI to Cloud Streaming Pipeline



Demo

StreamNative

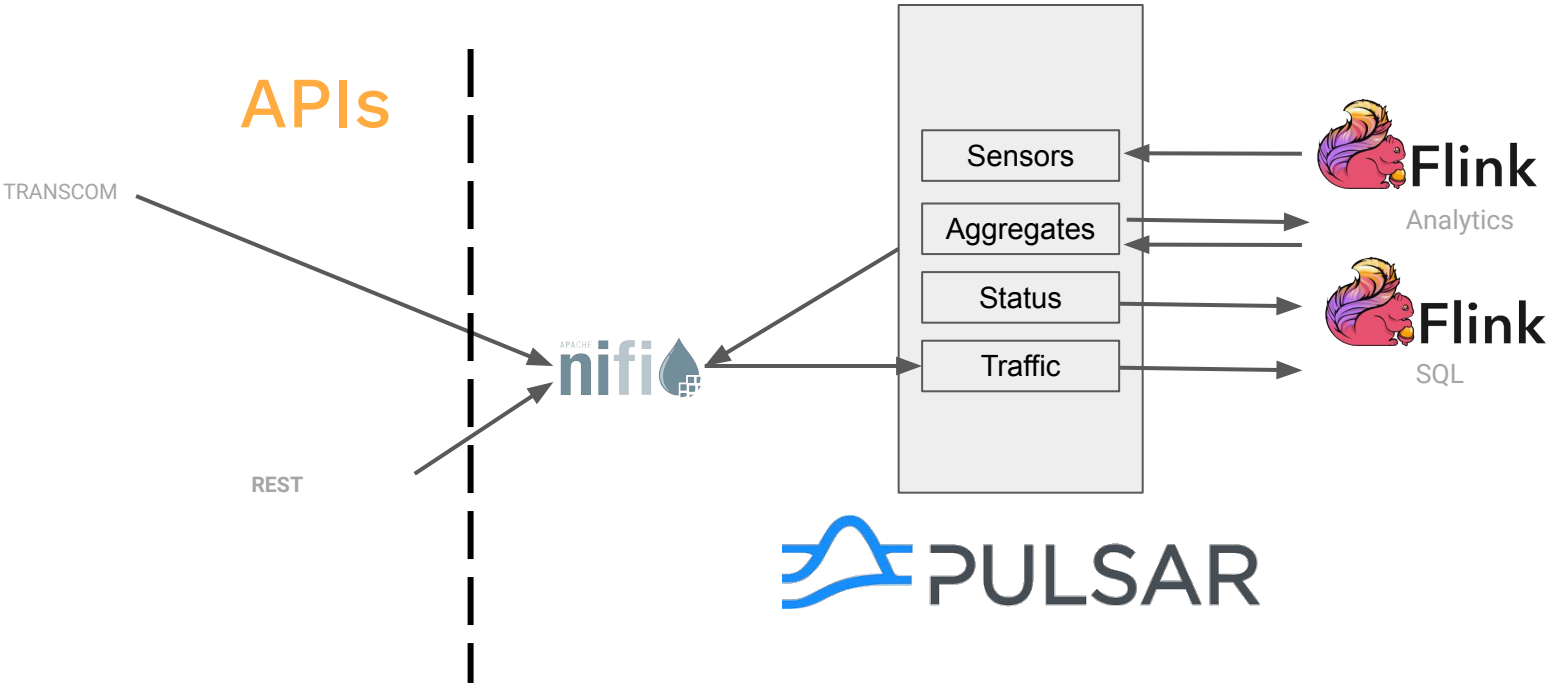
Demo Walkthrough



{"entriesAddedCounter":1,"numberOfEntries":1,"totalSize":651,"currentLedgerEntries":1,"currentLedgerSize":651,"lastLedgerCreatedTimestamp":"2021-09-13T16:13:06.6-04:00","waitingCursorsCount":0,"pendingAddEntriesCount":0,"lastConfirmedEntry":"7076:0","state":"LedgerOpened","ledgers":[{"ledgerId":7076,"entries":0,"size":0,"offloaded":false,"underReplicated":false}],"cursors":{},"schemaLedgers":[],"compactedLedger":{"ledgerId":-1,"entries":-1,"size":-1,"offloaded":false,"underReplicated":false}}

StreamNative

Real-Time Cloud Streaming Pipeline



StreamNative

Wrap-Up

StreamNative

Connect with the Community & Stay Up-To-Date

- Join the Pulsar Slack channel - *Apache-Pulsar.slack.com*
- Follow @streamnativeio and @apache_pulsar on Twitter
- Subscribe to Monthly Pulsar Newsletter for major news, events, project updates, and resources in the Pulsar community

Deeper Content

- https://www.datainmotion.dev/2020/04/building-search-indexes-with-apache.html
- https://github.com/tspannhw/nifi-solr-example
- https://github.com/streamnative/pulsar-flink
- https://www.linkedin.com/pulse/2021-schedule-tim-spann/
- https://github.com/tspannhw/SpeakerProfile/blob/main/2021/talks/20210729_HailHydrate!FromStreamtoLake_TimSpann.pdf
- https://streamnative.io/en/blog/release/2021-04-20-flink-sql-on-streamnative-cloud
- https://docs.streamnative.io/cloud/stable/compute/flink-sql

 **@PaasDev**  **timothyspann**

https://www.pulsardeveloper.com/

 StreamNative

Interested In Learning More?

Resources

[Flink SQL Cookbook]

[The Github Source for Flink SQL Demo]

[The GitHub Source for Demo]

Free eBooks

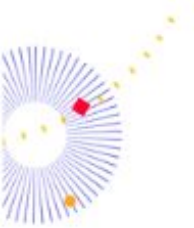
[Manning's Apache Pulsar in Action]

[O'Reilly Book]

Upcoming Events

[10/21] Trino Summit

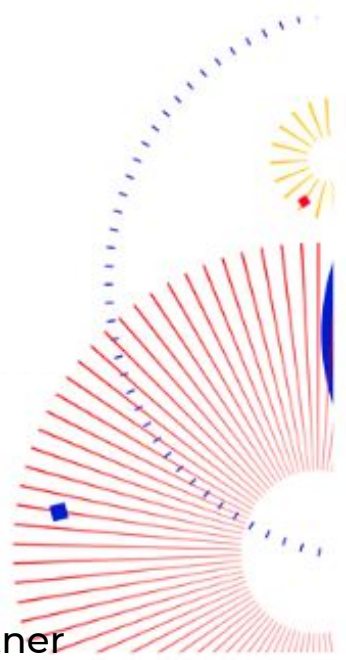
StreamNative



PULSAR SUMMIT

Pulsar Summit Asia

November 20-21, 2021



Contact us at partners@pulsar-summit.org to become a sponsor or partner

trino SUMMIT

10:45 - 11:25am EST
7:45 - 8:25am PST
3:45 - 4:25pm BST
8:15 - 8:55pm IST

FLiP Into Trino

Remember the days when you could wait until your batch data load was done and then you could run some simple queries or build stale dashboards? Those days are over, today you need instant analytics as the data is streaming in real-time. You need universal analytics where that data is. I will show you how to do this utilizing the latest cloud native open source tools. In this talk we will utilize Trino, Apache Pulsar, Pulsar SQL and Apache Flink to analyze instantly data from IoT, sensors, transportation systems, Logs, REST endpoints, XML, Images, PDFs, Documents, Text, semistructured data, unstructured data, structured data and a hundred data sources you could never dream of streaming before. I will teach how to use Pulsar SQL to run analytics on live data.

Tim Spann
Developer Advocate
StreamNative

David Kjerrumgaard
Developer Advocate
StreamNative

StreamNative

Wednesday, October 27, 2021

9:00 AM - 9:25 AM PDT

AI DEVWORLD | AI OPEN TALKS | AI FOR THE ENTERPRISE

Add

OPEN TALK (AI): Utilizing Apache PULSAR, Apache NiFi and MiNiFi for EdgeAI IoT at Scale

Timothy Spann
StreamNative, Developer Advocate

Thursday, October 28, 2021

9:00 AM - 9:25 AM PDT

API INNOVATION | EMERGING APIS

Add

PRO TALK (API): Apache NiFi 101: Introduction and Best Practices

Timothy Spann
StreamNative, Developer Advocate

Let’s Keep in Touch!

Tim Spann
Developer Advocate

@PassDev

https://www.linkedin.com/in/timothyspann

https://github.com/tspannhw

StreamNative

Questions