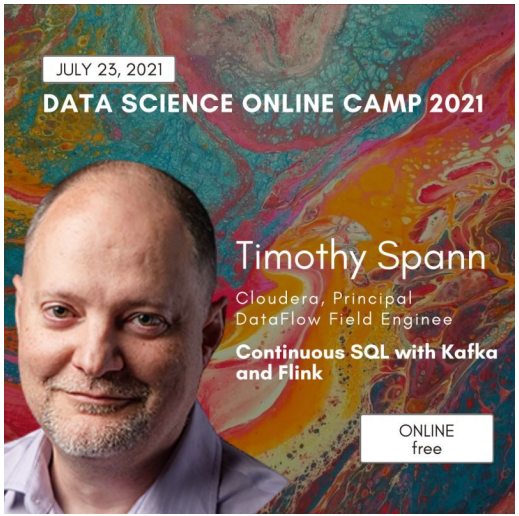
Continuous SQL with Apache Streaming



Timothy Spann
Developer Advocate

https://github.com/tspannhw/SpeakerProfile

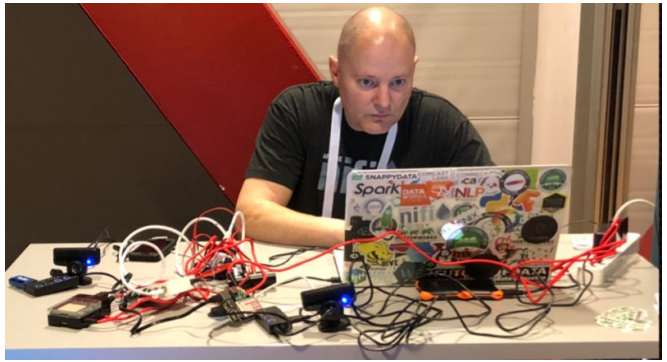

Tim
SPANN
https://github.com/tspannhw
https://www.datainmotion.dev/

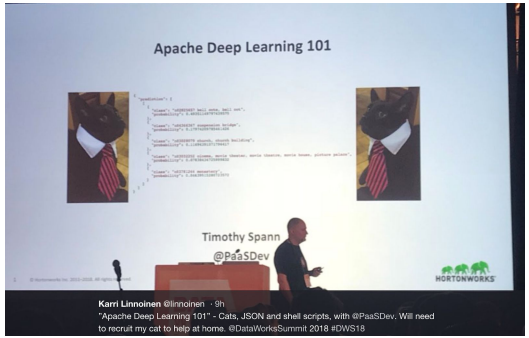
Speaker Bio

Developer Advocate

DZone Zone Leader and Big Data MVB;
@PaasDev

https://github.com/tspannhw https://www.datainmotion.dev/
https://github.com/tspannhw/SpeakerProfile
https://dev.to/tspannhw
https://sessionize.com/tspann/
https://www.slideshare.net/bunkertor

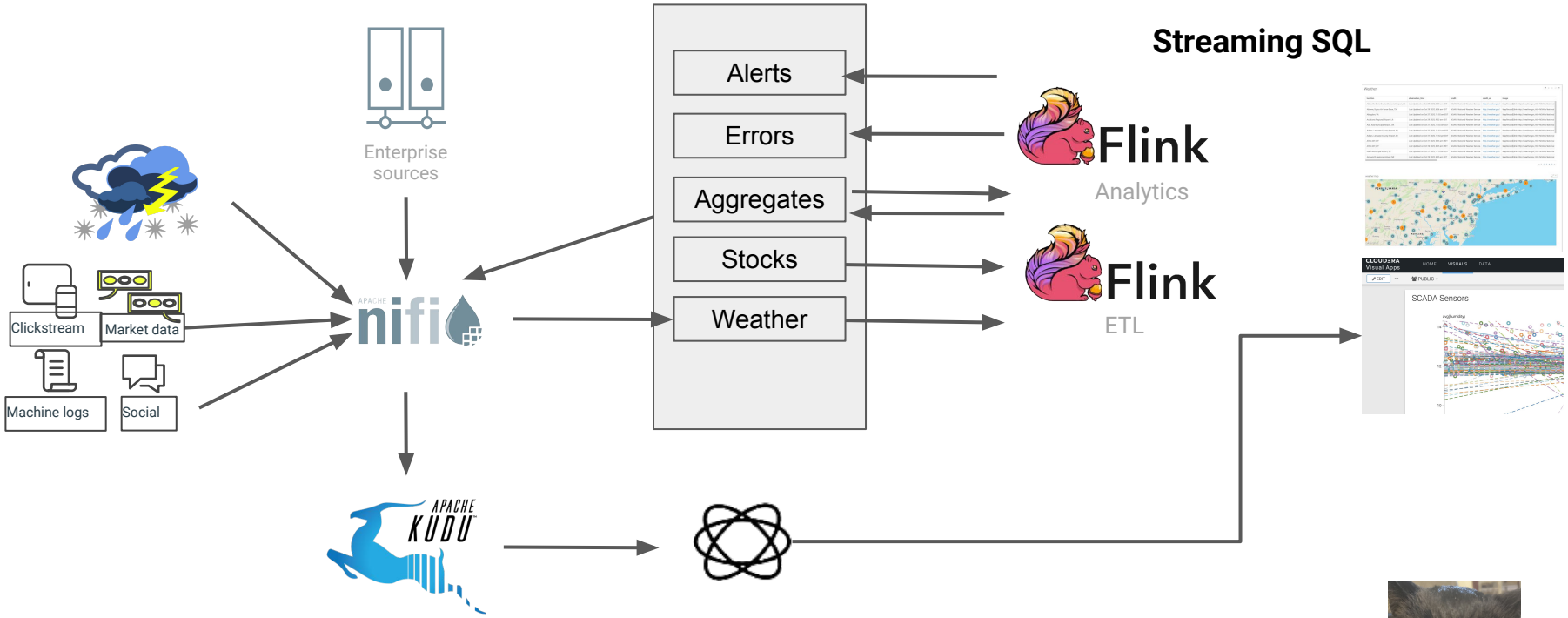




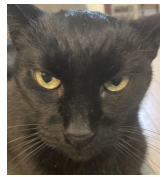
Today's Data. REST and Websocket JSON "stonks"

{"symbol":"CLDR",
"uuid":"10640832-f139-4b82-8780-e3ad37b3d0ce",
"ts":1618529574078,
"dt":1612098900000,
"datetime":"2021/01/31 08:15:00",
"open":"12.24500",
"close":"12.25500",
"high":"12.25500",
"volume":"12353",
"low":"12.24500"}

End to End Streaming Demo Pipeline

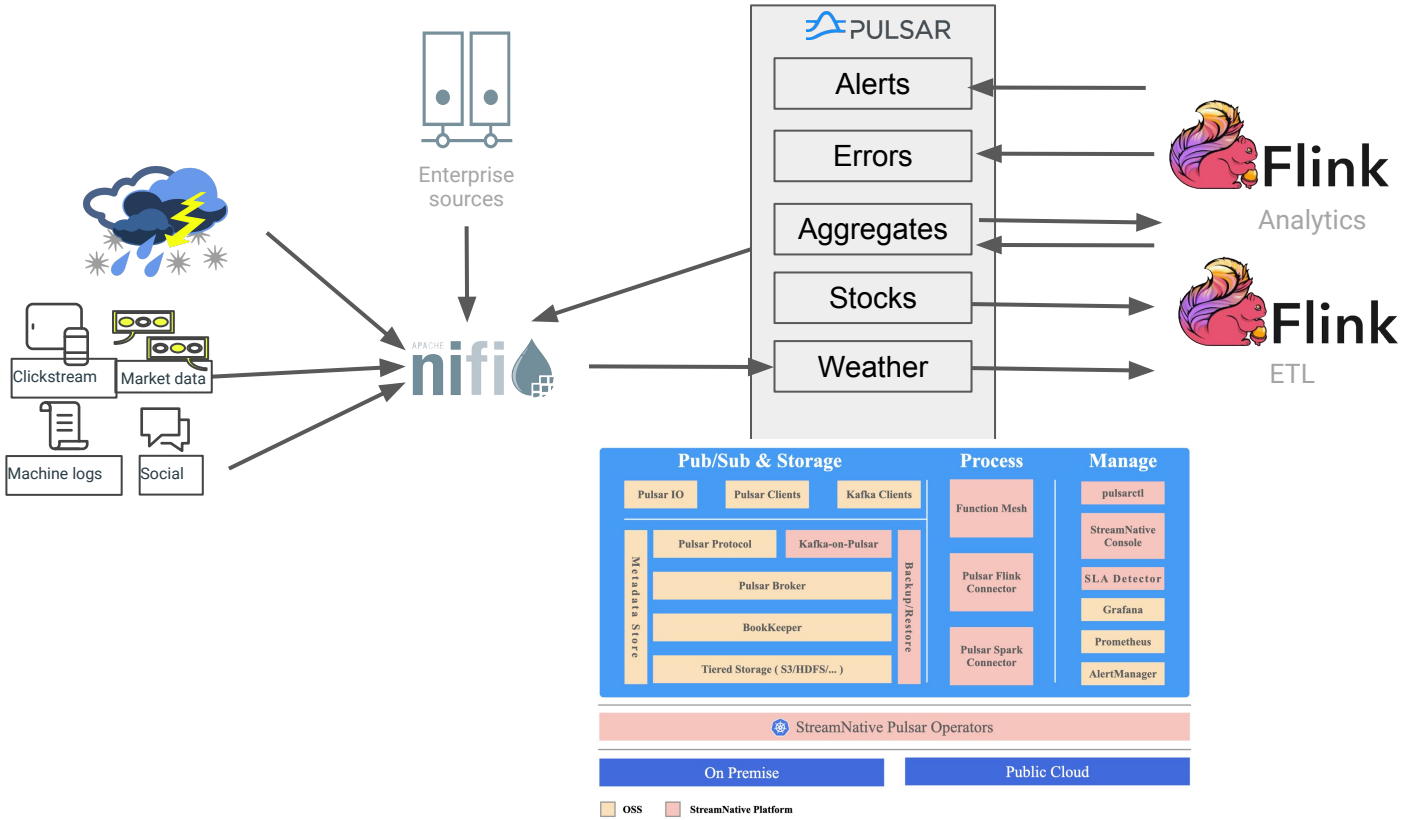


https://github.com/tspannhw/CloudDemo2021



End to End Streaming Demo Pipeline

Streaming SQL

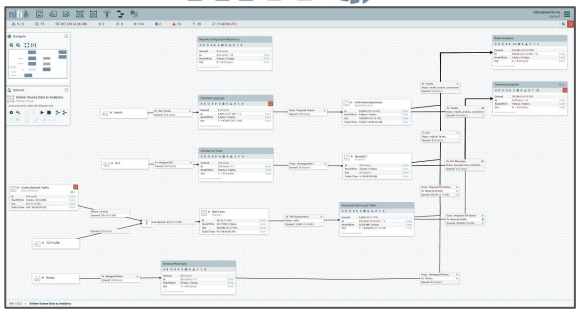


WHAT IS APACHE NIFI?

Apache NiFi is a scalable, real-time streaming data platform that collects, curates, and analyzes data so customers gain key insights for immediate actionable intelligence.

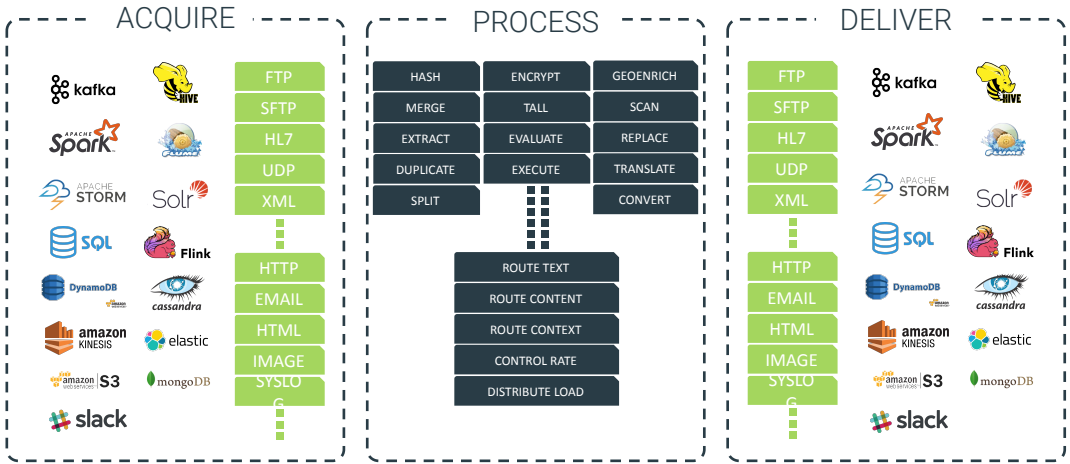
APACHE NIFI

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



APACHE NIFI registry

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

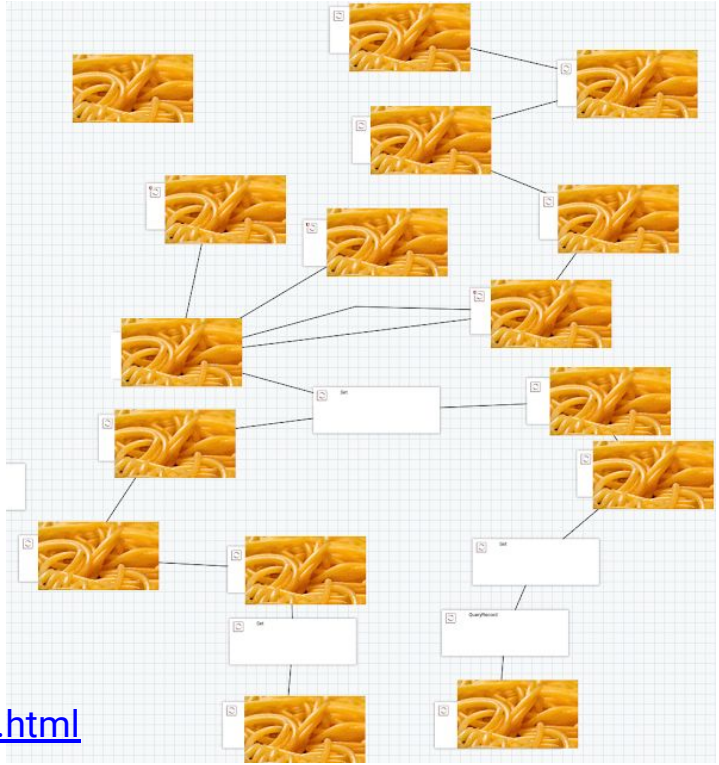


- Over 300 Prebuilt Processors
- Easy to build your own
- Parse, Enrich & Apply Schema
- Filter, Split, Merger & Route
- Throttle & Backpressure

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

No More Spaghetti Flows

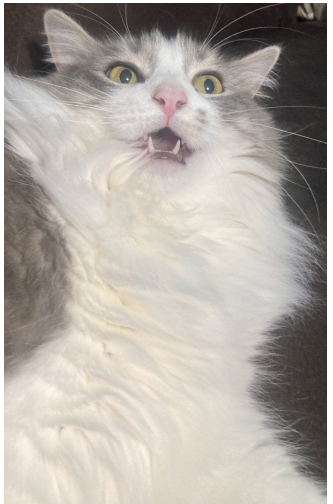
- Reduce, Reuse, Recycle. Use Parameters to reuse common modules.
- Put flows, reusable chunks into separate Process Groups.
- Write custom processors if you need new or specialized features
- Use Cloudera supported NiFi Processors
- Use Record Processors everywhere



https://www.datainmotion.dev/2020/06/no-more-spaghetti-flows.html

WHAT IS APACHE PULSAR?

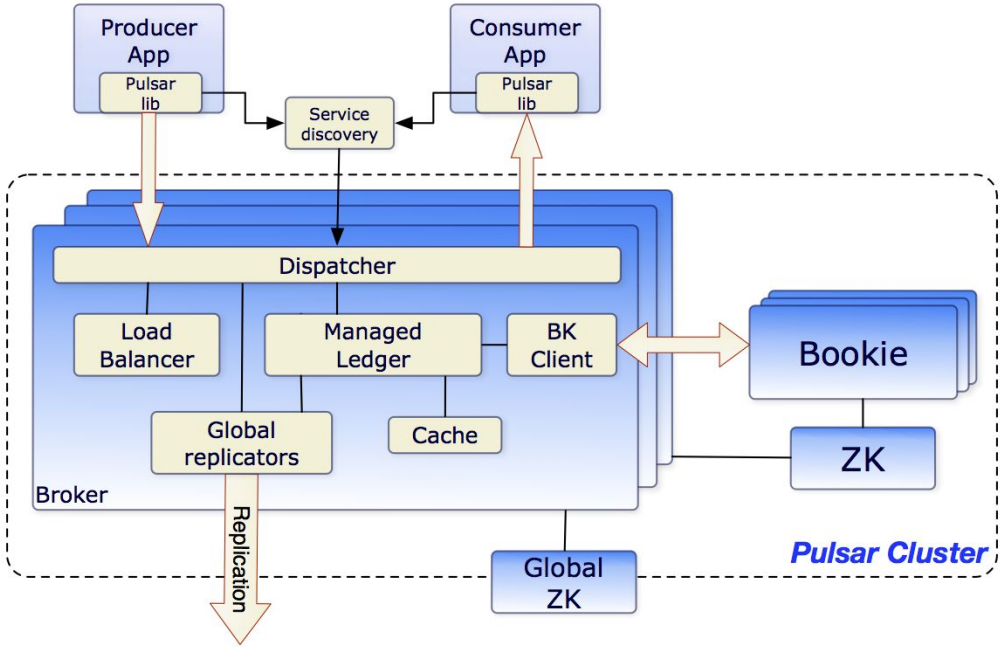
Apache Pulsar is an open source, cloud-native distributed messaging and streaming platform.



APACHE PULSAR

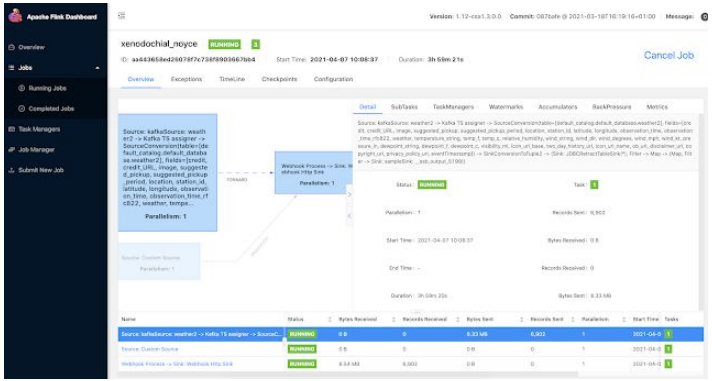
Enable Geo-Replicated Messaging

- 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
  - ...



Flink SQL

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



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

Flink SQL

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

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

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

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

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

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

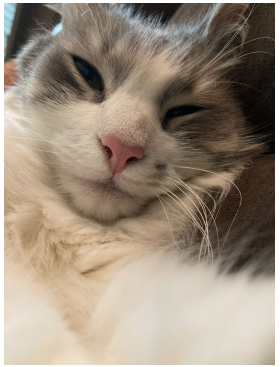
```
-- aggregations and windows
SELECT card,
MAX(amount) as theamount,
TUMBLE_END(eventTimestamp, interval '5' minute) as
ts
FROM payments
WHERE lat IS NOT NULL
AND lon IS NOT NULL
GROUP BY card,
TUMBLE(eventTimestamp, interval '5' minute)
HAVING COUNT(*) > 4  -- >4==fraud

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

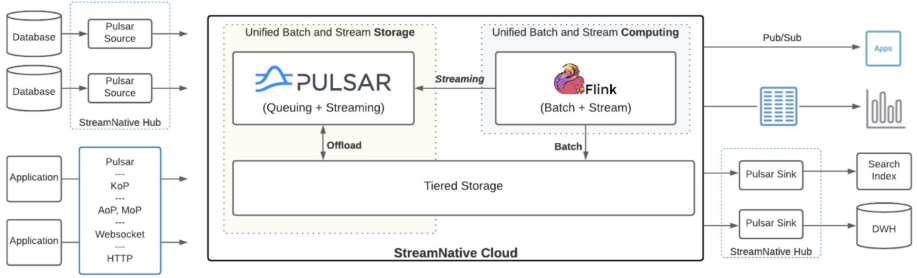
Flink SQL

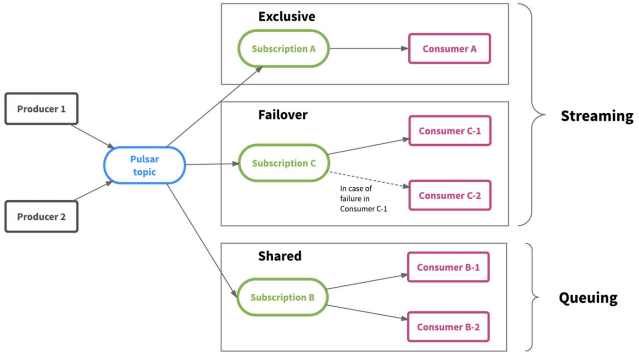
SELECT location, station_id, latitude, longitude, observation_time, weather, temperature_string, relative_humidity, wind_string, wind_dir, wind_degrees, wind_mph, pressure_in, dewpoint_string, dewpoint_f, dewpoint_c FROM weather2 **WHERE** location is not null and location <> 'null' and trim(location) <> '' and location like '%NJ'

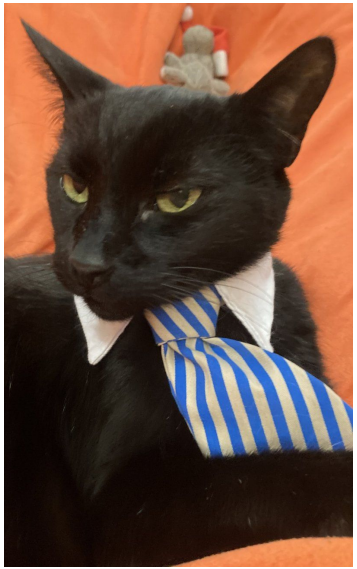
SELECT HOP_END(eventTimestamp, INTERVAL '1' SECOND, **INTERVAL** '30' SECOND) as windowEnd, **count**("close") as closeCount, **sum**(cast("close" as float)) as closeSum, **avg**(cast("close" as float)) as closeAverage, **min**("close") as closeMin, **max**("close") as closeMax, **sum**(case when "close" > 14 then 1 else 0 end) as stockGreaterThan14 **FROM** stocksraw **GROUP BY HOP**(eventTimestamp, INTERVAL '1' SECOND, INTERVAL '30' SECOND)



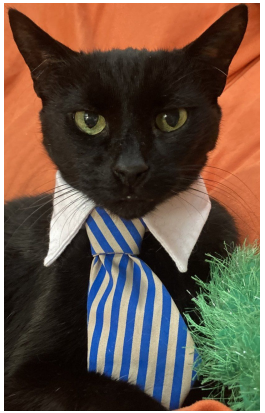
Upcoming - Flink + Pulsar (FLiP)







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



LET’S CONNECT!

@PaasDev