

goto;

GOTO **CHICAGO 2023**

#GOTOchgo

Event Streaming in the Cloud Native World with Apache Pulsar

Mary Grygleski
Streaming Developer Advocate
 @mgrygles



This slide deck can be accessed here:

<https://bit.ly/3OvWXiP>





Mary Grygleski

The Passionate Developer Advocate



@mgrygles



<https://www.linkedin.com/in/mary-grygleski/>



<https://www.twitch.tv/mgrygles>



<https://discord.gg/RMU4Juw>

Who is Mary?

Mary is a Senior Developer Advocate at DataStax, a leading Data Management Company that specializes in Database-as-a-Service, NoSQL, Big Data, Streaming, and the Cloud-Native platform. Previously she was with the Java and WebSphere/Open Source Advocacy team at IBM.

Based out of Chicago, Mary is a Java Champion and President and Executive Board Member of the Chicago Java Users Group (CJUG). She is also co-organizers for the Data, Cloud and AI In Chicago, Chicago Cloud, and IBM Cloud Chicago meetup groups.

She has extensive experience in product and application design, development, integration, and deployment experience, and specializes in Event-driven, Reactive Java, Open Source, and cloud-enabled distributed systems.

Senior Developer Advocate



Passionate Advocate



Java Champion

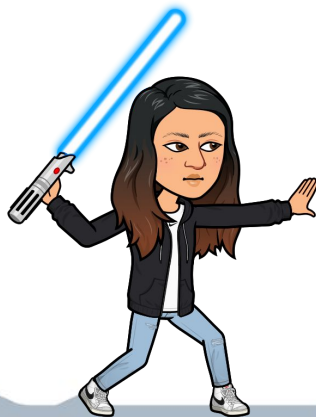
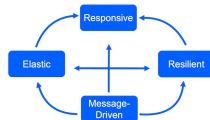


JAKARTA EE



Open Liberty

RED HAT OPENSIFT



- Streaming
- Distributed Systems
- Reactive Systems
- IoT/MQTT



mgrygles



mary-grygleski



mgrygles



mgrygles

AGENDA

The Many Facets of Computing Events

- Event Streaming
- Event Processing
- Complex Event Processing
- Event-Driven vs Message-Driven

Event Messaging Semantics / Patterns

- Pub/Sub
- Queue

How can Event Streaming improve your systems?

An Intro to Apache Pulsar

- Apache Zookeeper & Apache BookKeeper

How is Pulsar Different?

Highlighting a few Developer Features

DataStax Managed Cloud - Astra Streaming

Quick Demo



The Many Facets of Computing Events

What is an event, generically speaking?

From Merriam-Webster.com:

1 a :something just happens: OCCURRENCE

...

4 : the fundamental entity of observed physical reality represented by a point designated by three coordinates of place and one of time in the space-time continuum postulated by *the theory of relativity*

The alphabet soup of *event computing*

Data data data data data data data data

Event Sourcing

Event-Driven
Architecture

Stateful
Microservices

Serverless Apps

Event Messaging

Stateless
Microservices

Event Processing

Event Streaming

Event Storming

Reactive Systems

Event Streaming Processing

The practice of taking action on a series of data points that originate from a system that continuously creates data.

“Event” refers to each data point in the system,

“Stream” refers to the ongoing delivery of those events.

A series of events can also be referred to as “streaming data” or “data streams.”

Complex Event Processing

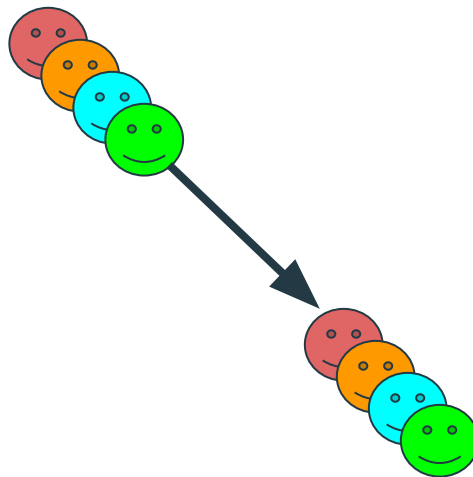
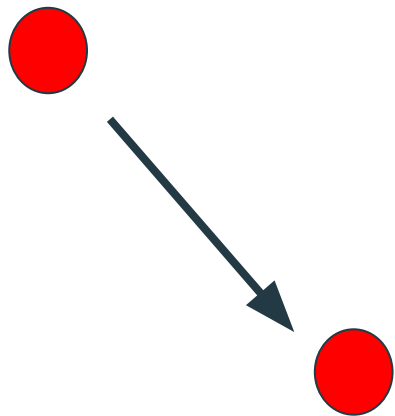
Complex event processing (CEP) is a set of techniques for capturing and analyzing streams of data as they arrive to identify opportunities or threats in real time. CEP enables systems and applications to respond to events, trends, and patterns in the data as they happen.

Event-Driven vs Message-Driven Messaging

Event-Driven -> Sender emits messages and interested subscribers can subscribe to the messages

Message-Driven -> Sender and Receiver are known to each other (address is known)

Event Approach vs Batch Processing

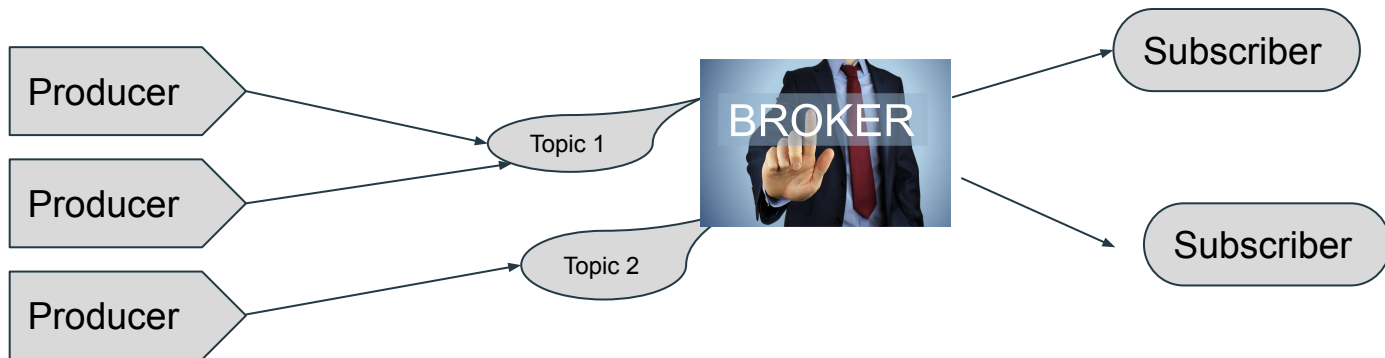




Event Messaging Semantics / Patterns

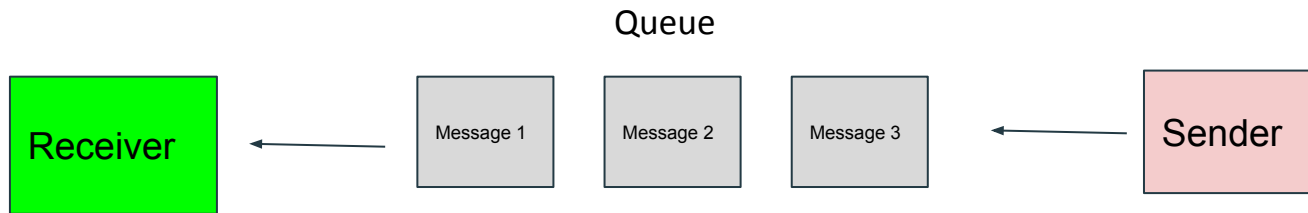
Streaming: Pub/Sub

- Publishing Client sends the data to the topic
- Broker is the middle-person / agent, owns the topic
- Subscribing Client receives the data from the topic



Message Queueing

A message queue is a form of asynchronous service-to-service communication used in serverless and microservices architectures. Messages are stored on the queue until they are processed and deleted. Each message is processed only once, by a single consumer. Message queues can be used to decouple heavyweight processing, to buffer or batch work, and to smooth spiky workloads.

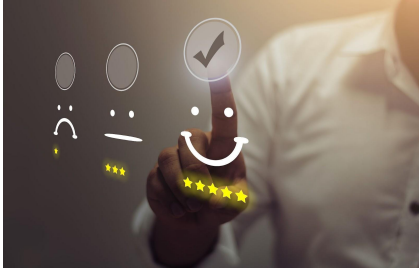


Message gets picked up once and is gone after that



Event Streaming – A Step beyond Event Messaging

What's driving this change?



Use real time data to enhance customer experiences and create a competitive advantage for your business



Use data pipelines to build AI/ML and smart models from time series event streams.



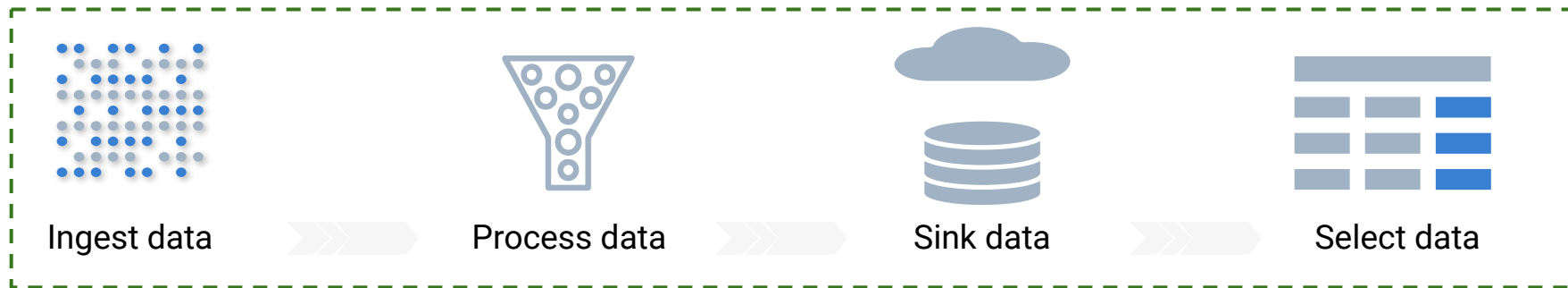
Scale to meet the demands of large volumes of data generated by application operating at the edge

Why event streaming

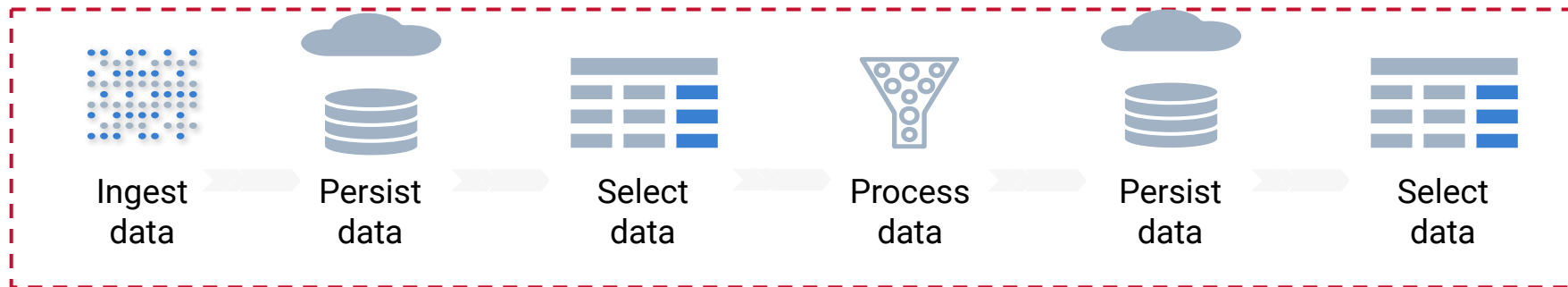
- Watch for events with “the system” or application
- Subscribe to topics to see certain event types
- Make decisions on data in real time. Not after the event.
- Ingest high frequency of messages with very low latency

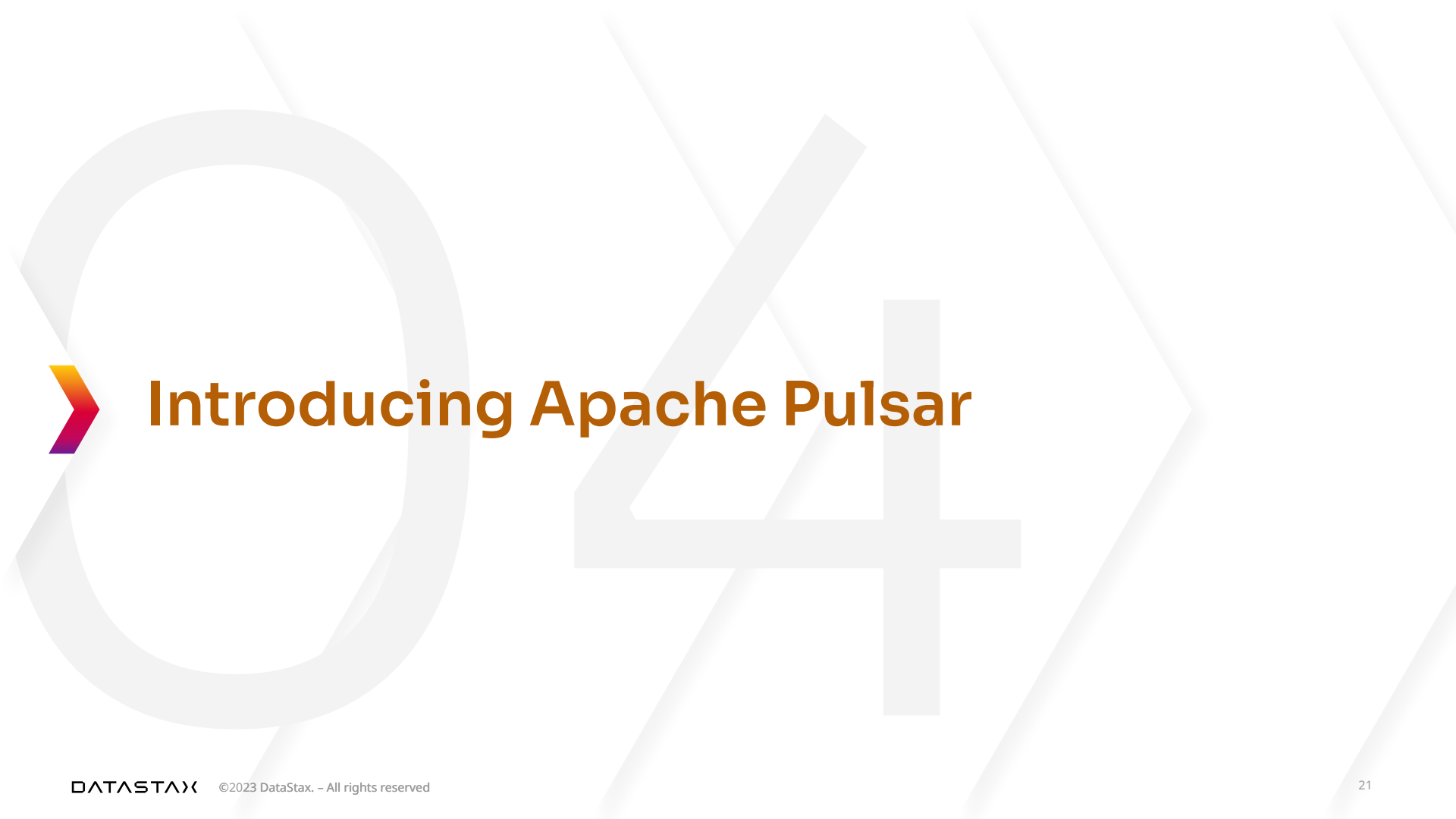
Streaming versus not streaming

Streaming



Not Streaming





Introducing Apache Pulsar

Open source

Created by Yahoo

Contributed to the Apache Software Foundation (ASF) in 2016

Top-level project (2018)

Cloud-native design

Cluster based

Multi-tenant

Simple client APIs (Java, C#, Python, Go, ...)

→ Separate compute and storage!

Guaranteed message delivery

If a message successfully reaches a Pulsar broker, it will be delivered to its intended target.

Light-weight serverless functions framework

Create complex processing logic within a Pulsar cluster (aka: data pipeline)

Tiered storage offloads

Offload data from hot/warm storage to cold/long-term storage when the data is aging out

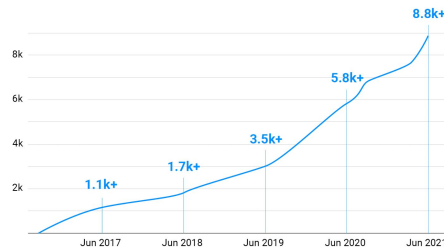
**Meet
Pulsar**

What is Apache Pulsar

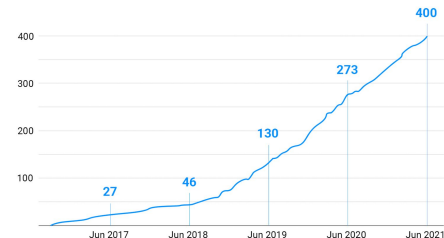
- Unified, distributed messaging and streaming platform
- Open source
 - Originally developed at Yahoo!
 - Contributed to the Apache Software Foundation (ASF) in 2016
 - Top-level project (2018)
- Cloud Native
 - K8s
 - Multi-cloud and hybrid-cloud

Four Reasons Why Apache Pulsar is Essential to the Modern Data Stack

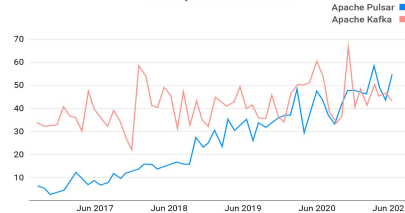
Apache Pulsar Github Stars



Apache Pulsar Contributors



Monthly Active Contributors



Source: APIDevs.com

Who else is using Pulsar?



Q5

➤ How is Pulsar Different ?

Pulsar Components

Producer

Client application sending messages to topic managed by Broker

Consumer

Client application reading messages from a topic managed by Broker

Broker

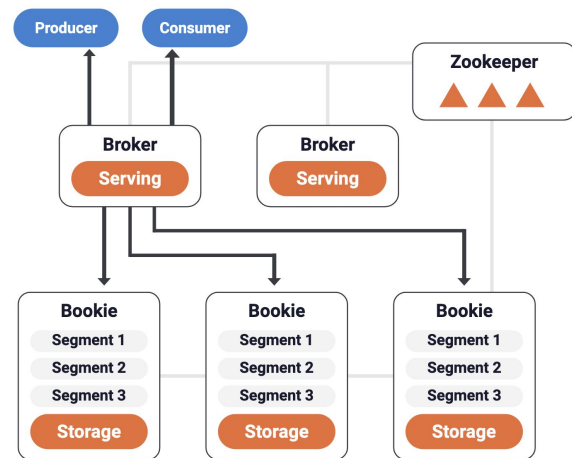
A stateless process that handles incoming message, message dispatching, communicates with the Pulsar configuration store, and stores messages in BookKeeper instances

BookKeeper

Persistent message store

ZooKeeper

Holds cluster metadata, handles coordination tasks between Pulsar clusters



Design Principle: Tiered Architecture Design, continued

Traditional Multi-Node Architecture

Distributed architecture supports horizontal scaling

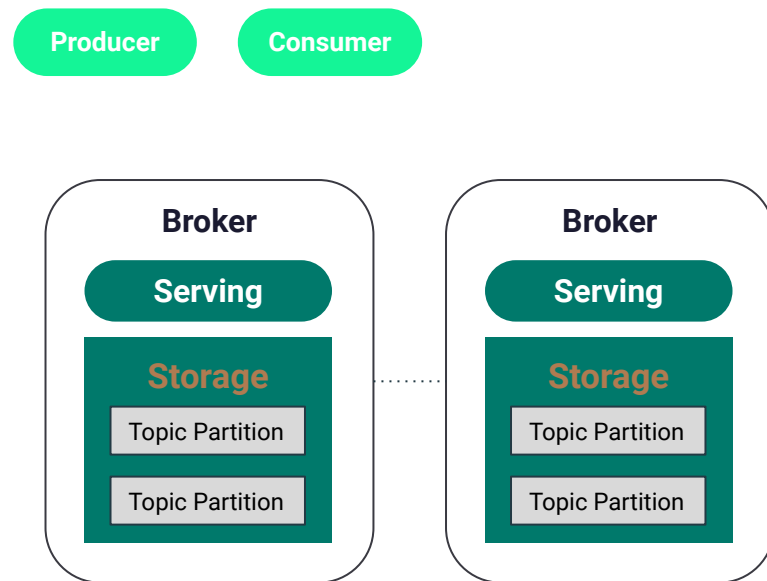
Partitioned topic abstraction masks complexity from consumers

Common Challenges

Scaling requires partition rebalancing

Tightly coupled persistence and message serving capabilities impose high cost on historical data.

Trade offs to support partitioned topics came at the expense of messaging semantics needed for use cases such as queuing.

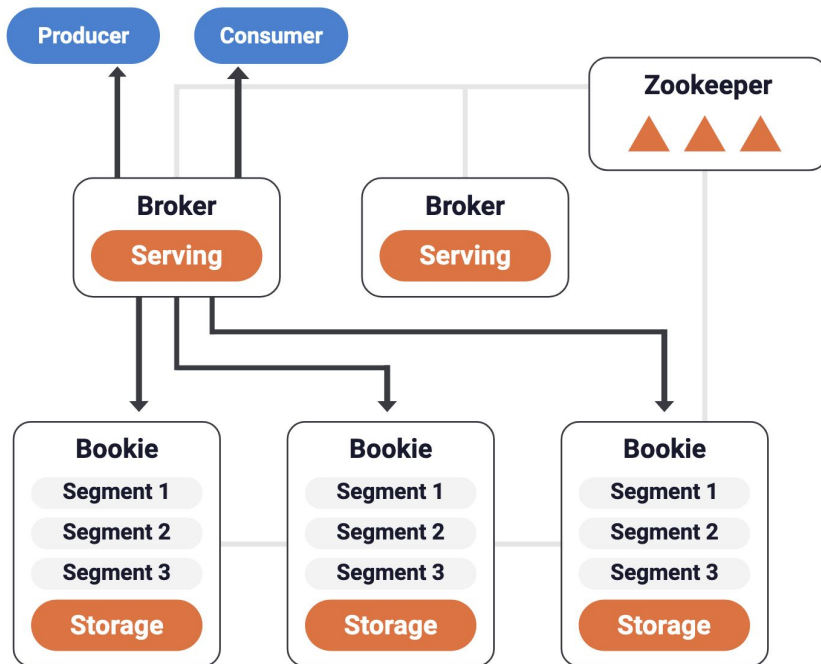


Design Principle: Tiered Architecture Design, continued

Pulsar's Multi-Node

Architecture

- What's the big deal?
 - Fast, Low impact, horizontal scaling
 - Reduced CAPEX and OPEX
- Broker
 - Stateless
 - Built-in load balancing
 - Instantaneous scaling
 - Zero impact disaster recovery
- Bookie
 - Scalable, WAL based, fault-tolerant, low latency storage service
 - Tunable consistency for message replication
 - Ensemble Size, Write Quorum, Ack Quorum
 - Fast write guarantee through Journals
 - Segment-centric data persistence via Ledgers



Apache Pulsar Solves the Problems of Bolt-on

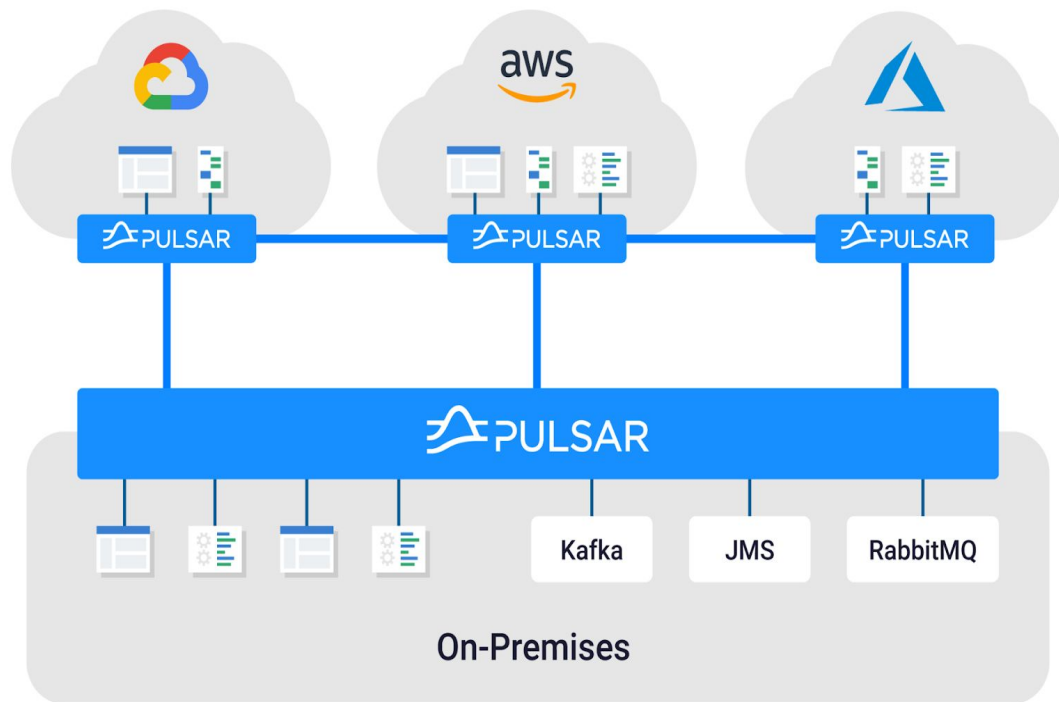
Apache Pulsar represents the **Next Generation of Enterprise Messaging**

Unified Solution for

- Pub/Sub
- Queuing
- Streaming
- Message mediation & enrichment

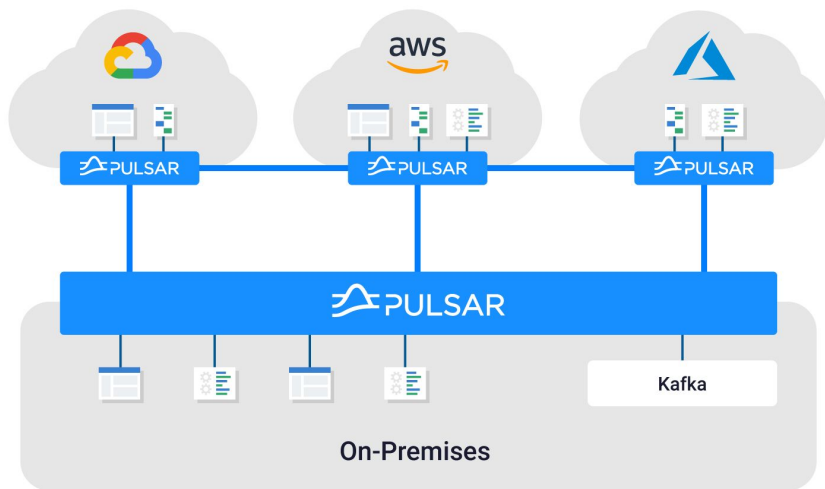
Out of the Box Capabilities Include

- Cloud, on-prem & hybrid
- Geo-replication
- Multi-region support
- Data lake integration
- And much, much more...

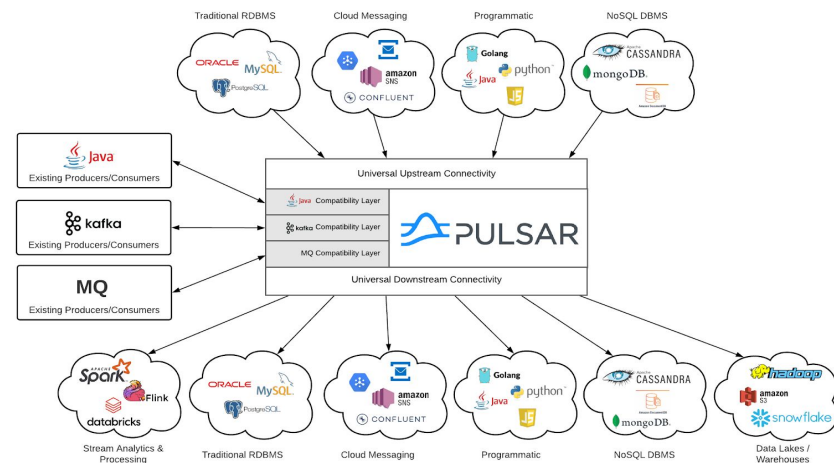


Apache Pulsar as an Unified Platform

- Unified infrastructure with built-in geo-replication
 - Multi-cloud, hybrid-cloud, multi-region



- Unified enterprise messaging/streaming backbone
 - Wire-level messaging protocol compatibility



How is Pulsar Different?

Pulsar's next generation architecture provides

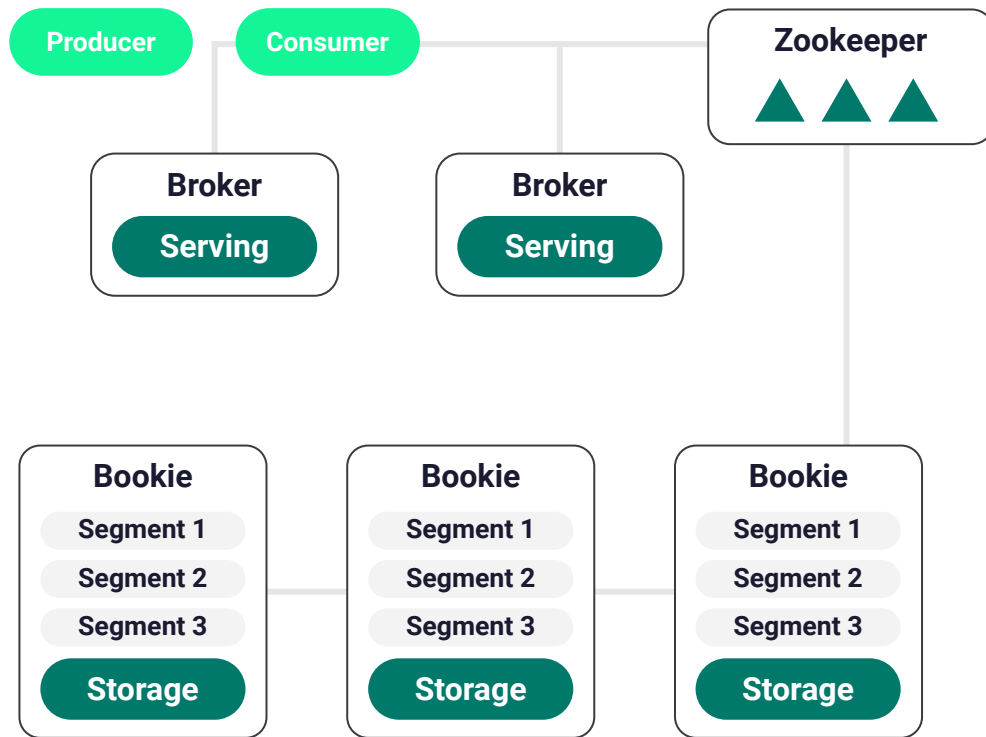
Distributed, tiered architecture

Separates compute from storage

Zookeeper holds metadata for the cluster

Stateless Broker handles producers and consumers

Storage is handled by Apache Bookkeeper



Pulsar Features



Rapid Patching, Zero Downtime, etc



Multi-Tenant

Soft Isolation via read/write I/O separation, independent storage quotas, and message flow control and throttling mechanisms

Hard Isolation via physically separate brokers and/or bookies for tenants



Identity and Access Management

Pluggable Authentication supporting TLS authentication, JWT, OAuth 2.0, Athenz, Kerberos

Role-based authorization providing control at the cluster, tenant, message producer and consumer level.

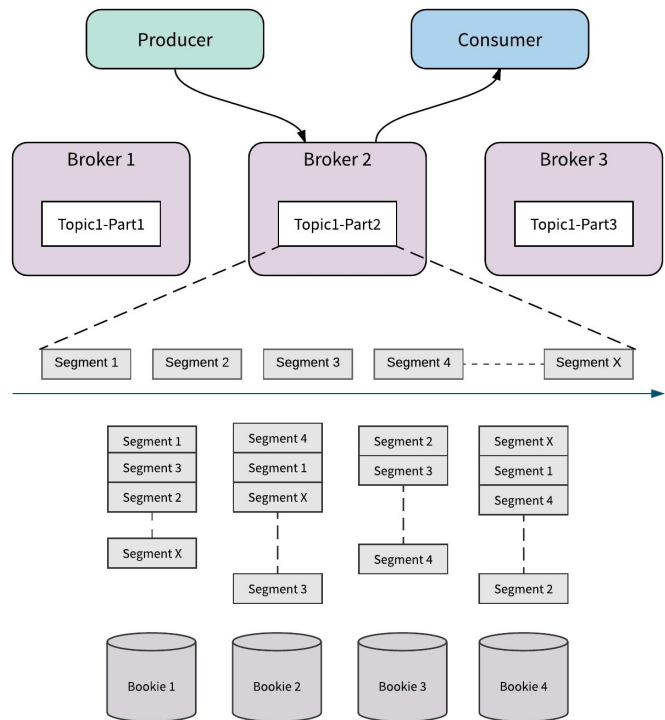


End-to-End Encryption

In transit TLS encryption and application managed content encryption.

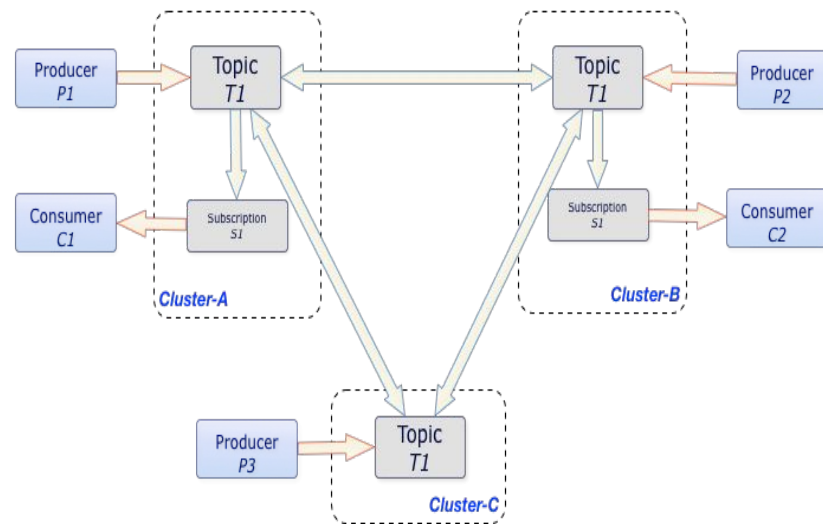
Key Differentiator 1: Separation between Compute and Storage

- Distributed, tiered architecture
 - Separates compute from storage
 - Independent scaling
- Stateless Broker handles producers and consumers
 - Intelligent, automatic load balancing
- Storage is handled by Apache BookKeeper
 - **Segment-centric** message storage management
- **Fast and Low Impact Horizontal Scaling Capability**



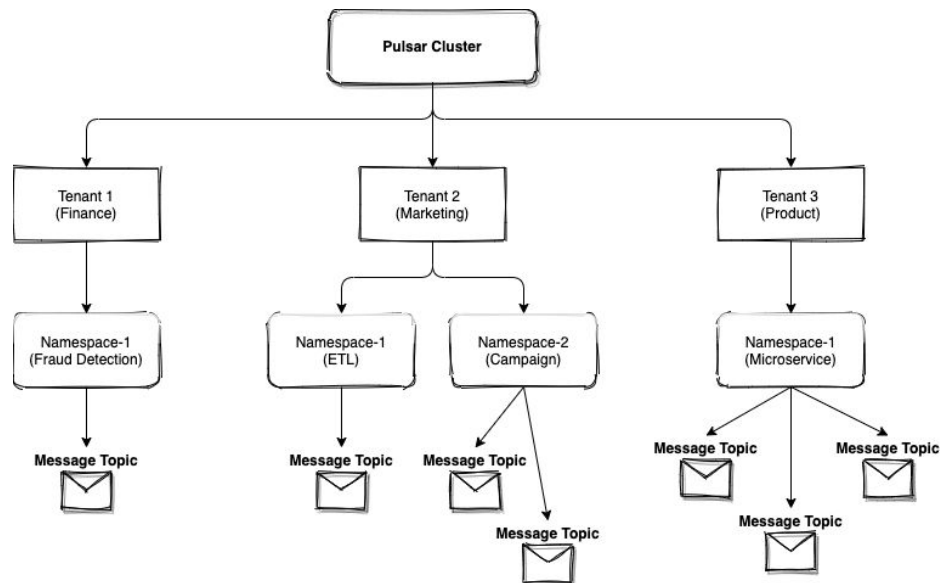
Key Differentiator 2: Native Geo-Replication

- Hands off, real time message replication across data centers
- Flexible message replication mode and patterns
 - Synchronous vs Asynchronous
 - Active-Active, Active-Passive
 - Selective message replication
- Capabilities to meet Data Compliance requirements across geo-regions



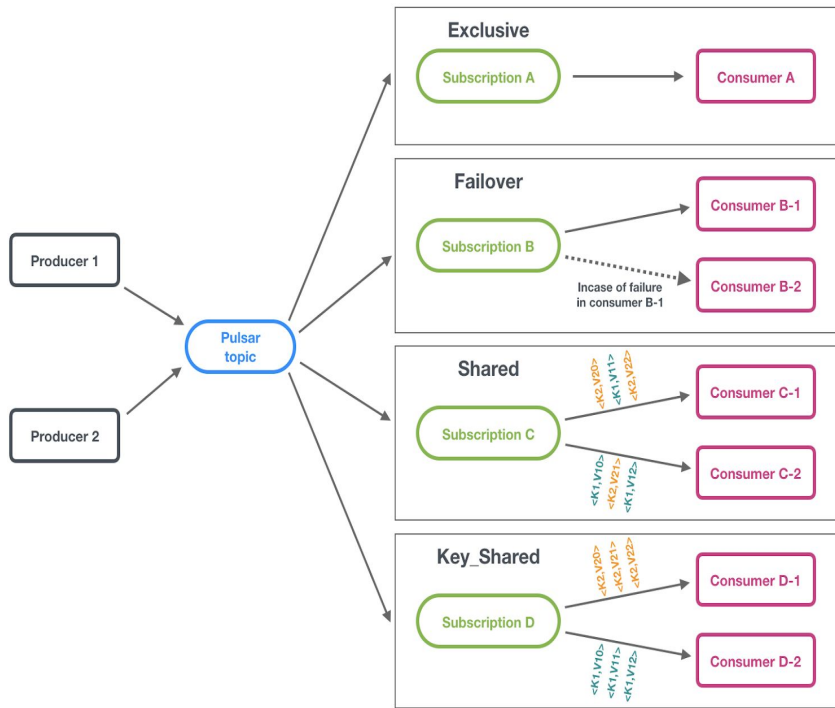
Key Differentiator 3 : Multi-Tenancy

- Consolidated messaging/streaming platform
 - Operation simplicity
- Effective permission control within business domain context
 - Security, compliance, auditing
- Better IT resource utilization. Reduce Total Cost of Ownership (TCO)
 - Storage Quota
 - Message flow control and throttling mechanisms
 - Physically separate brokers and/or bookies for tenants



Key Differentiator 4 : Flexible Message Processing Model

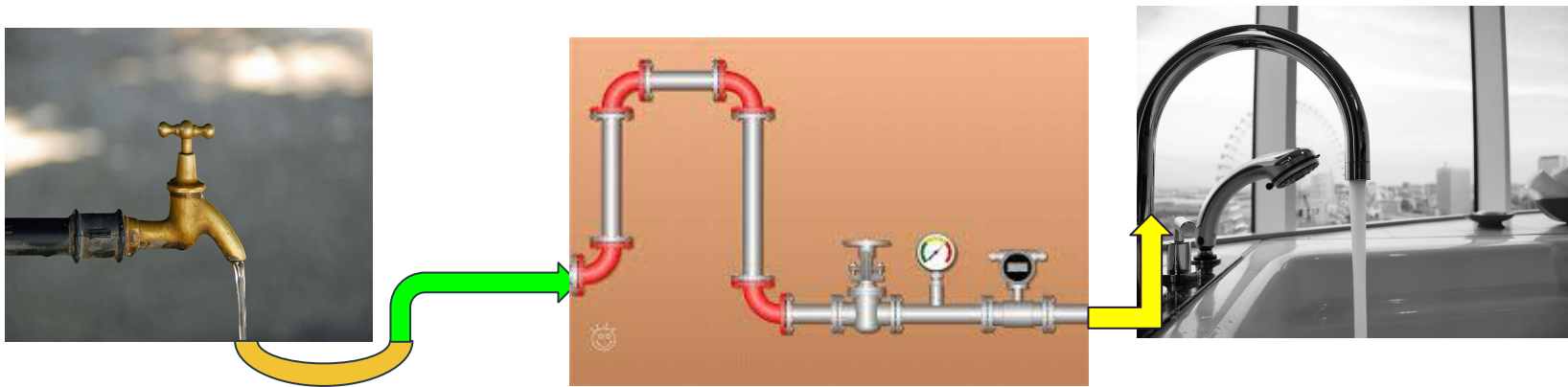
- Out-of-the box multi-subscription modes
 - Exclusive
 - Failover
 - Shared
 - Key_Shared
- Good fit with Queuing use case as well
 - Kafka has challenges for this



▶ Highlighting A Few Developer Features

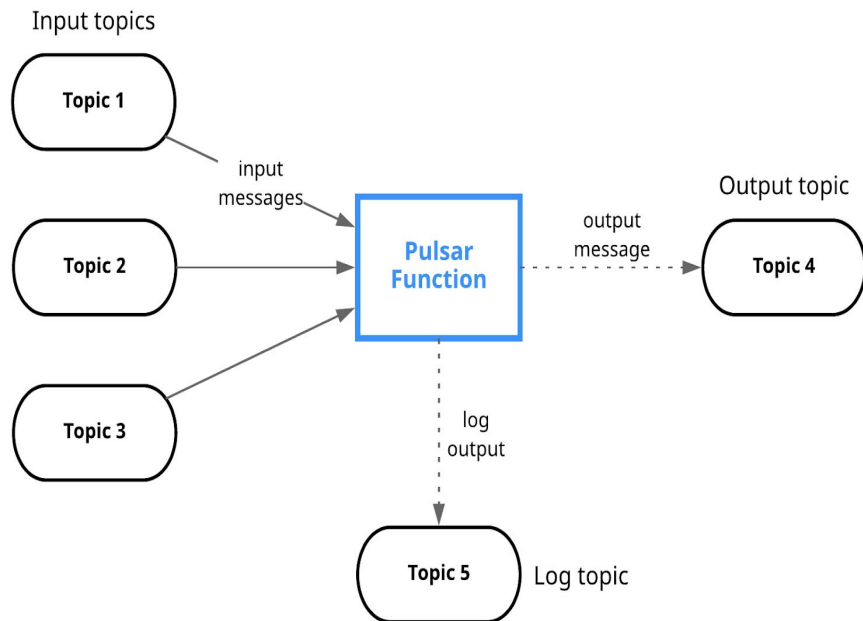
› Data Pipeline Illustrated

“Function” is there to transform the data in the most efficient way!



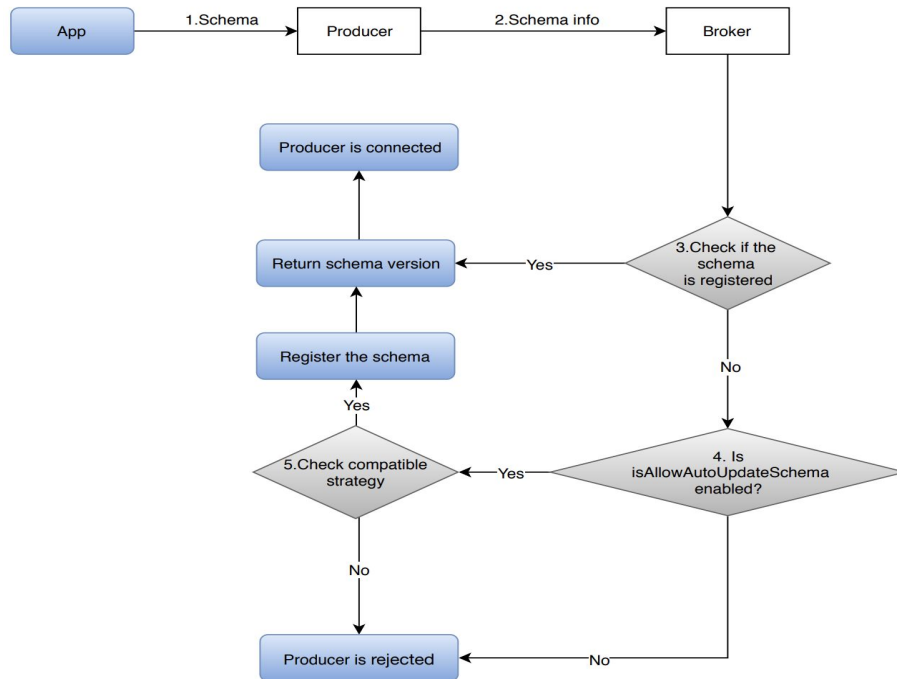
› Pulsar Functions

- ❖ Allows complex streaming processing
- ❖ Light-weight
- ❖ Function-as-a-service (“inspired by” AWS Lambda, Google Function, ...)
- ❖ Main languages:
 - Java
 - Python
 - Go



› Pulsar Schema

- ❖ If you do not want to worry about serializing and deserializing your data transfer over the wire
- ❖ Built-in schema registry
- ❖ Schema type
 - Primitive
 - Key/value pair
 - Avro, JSON, Protobuf
- ❖ Schema evolution
 - Version
 - Compatibility
- ❖ Schema Management
 - Automatic
 - Manual



› Pulsar I/O

Build your own source and sink connectors!





DataStax Managed Cloud – Astra

Pulsar meets you where you are

Astra Streaming

Managed Pulsar

Luna Streaming

Enterprise Support
Pulsar

Open Source

Community Driven
Pulsar

Pulsar as -the- messaging substrate

01

Pulsar Native

02

Starlight for
RabbitMQ

03

Starlight for Kafka

04

Starlight for JMS

DataStax Investment in Streaming Related Products

Customer Managed / On Prem

Luna Streaming

Support offering for Apache Pulsar or DataStax Luna Streaming distribution of Pulsar.

CDC for Cassandra

Support offering for our customer-managed CDC solution for DSE / OSS C*. Uses Pulsar under the covers.

Cloud Offerings

Astra Streaming

Apache Pulsar as a Service, fully cloud offering that is part of Astra.

CDC for Astra DB

CDC solution for Astra DB that pushes data changes into **Astra Streaming.**

Rich Ecosystem of Connectors and Clients (as of Jan 2023)



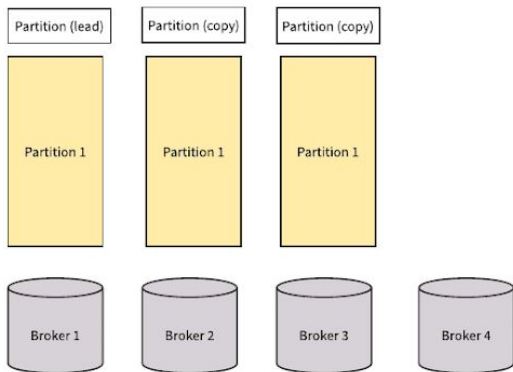
Partition-Centric vs. Segment-Centric

Apache Kafka

Logical partition view



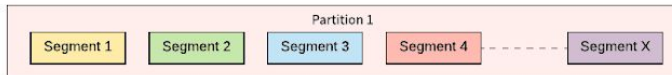
Distribution



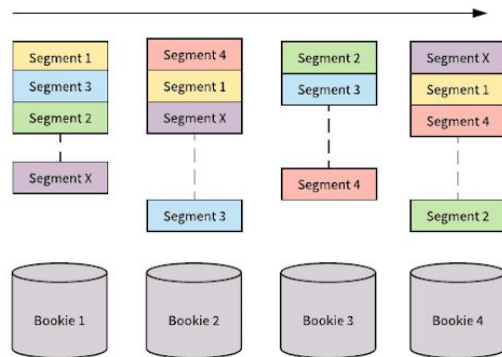
Kafka Partitions — All log segments are replicated in order across brokers (replication = 3 here).

Apache Pulsar/BookKeeper

Logical topic/
partition view



Segment
distribution

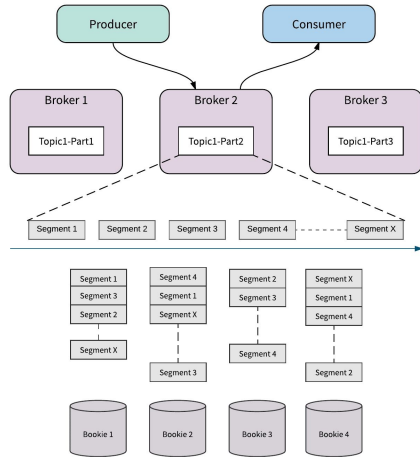


Pulsar/BookKeeper Stream — All log segment are replicated to a configurable number of bookies (replication = 3 here) across N possible bookies (N = 4 here). Log segments are evenly distributed to achieve horizontal scalability with no rebalancing.

Architecture Advantage of Pulsar

- Compute and Storage Separation

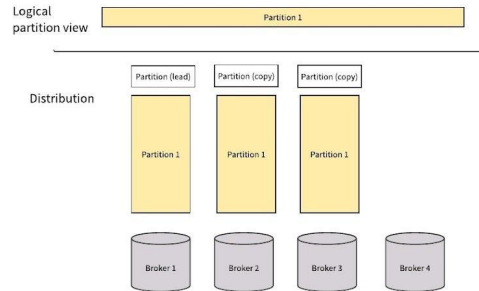
- Stateless brokers
- Independent scalability
- Instantaneous broker scaling and disaster recovery



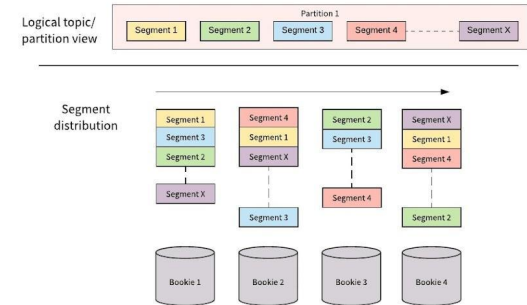
- Segment-Oriented Log Management

- Segment (of a Partition) as the smallest replication unit
- Efficient storage utilization; Unbounded partition storage
- Truly horizontal scalability
- Fast and low impact scaling and disaster recovery

Apache Kafka



Apache Pulsar/BookKeeper



Quick Demo

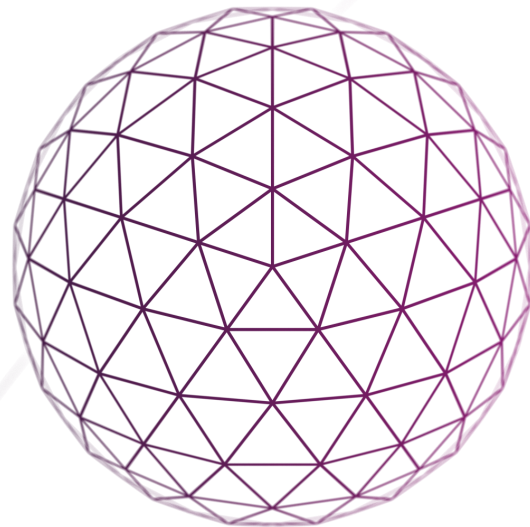


Sample code:

<https://github.com/mgrygles-lab/PulsarClientTestOne>



Appendix / Resources





Resources – Apache Pulsar and Astra from DataStax



<https://pulsar.apache.org/>



<https://bookkeeper.apache.org/>



<https://zookeeper.apache.org>

DATASTAX

ASTRA DB

<https://astra.datastax.com>

DATASTAX

ASTRA STREAMING

<https://www.datastax.com/products/astra-streaming>

<https://www.datastax.com/products/luna-streaming>

DATASTAX

LUNA STREAMING

Documentation for Streaming:

<https://docs.datastax.com/en/streaming/streaming/index.html>

DATASTAX

ASTRA

CDC for Astra: <https://docs.datastax.com/en/astra/docs/astream-cdc.html>



Community Resources – Apache Pulsar

Community Info

Apache Pulsar Community Info (Slack, Mailing Lists, StackOverflow, WeChat): <http://pulsar.apache.org/en/contact/>

Pulsar Slack (how to sign up): <https://apache-pulsar.herokuapp.com/>

Source Code

Apache Pulsar: <https://github.com/apache/pulsar>

DataStax on GitHub: <https://github.com/datastax>

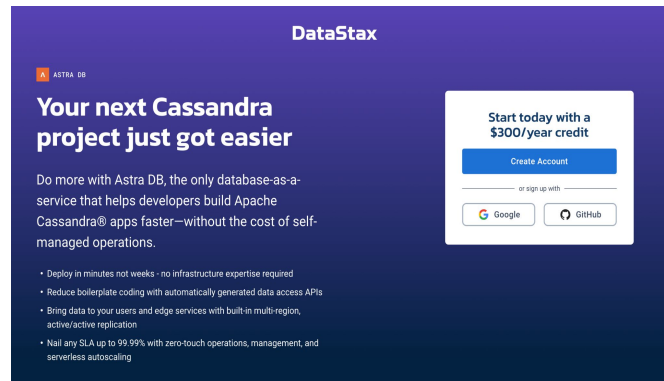
Starlight for Kafka (from DataStax) - <https://github.com/datastax/starlight-for-kafka>

Starlight for JMS (from DataStax) - <https://github.com/datastax/pulsar-jms> / <https://www.datastax.com/starlight/jms>

Starlight for RabbitMQ (from DataStax) - <https://github.com/datastax/starlight-for-rabbitmq>

DATASTAX

ASTRA



1.- Create an Astra account at

<https://www.datastax.com/lp/next-cassandra-project>

2.- Add a payment method, enter **OpenSource200** for an additional \$200 in credits

Check out **5 Minutes About Pulsar** on YouTube



<https://bit.ly/3bgkRxJ>

Follow Mary's Twitch Stream

(Different topics: Java, Open Source, Distributed Messaging, Event-Streaming, Cloud, DevOps, etc)

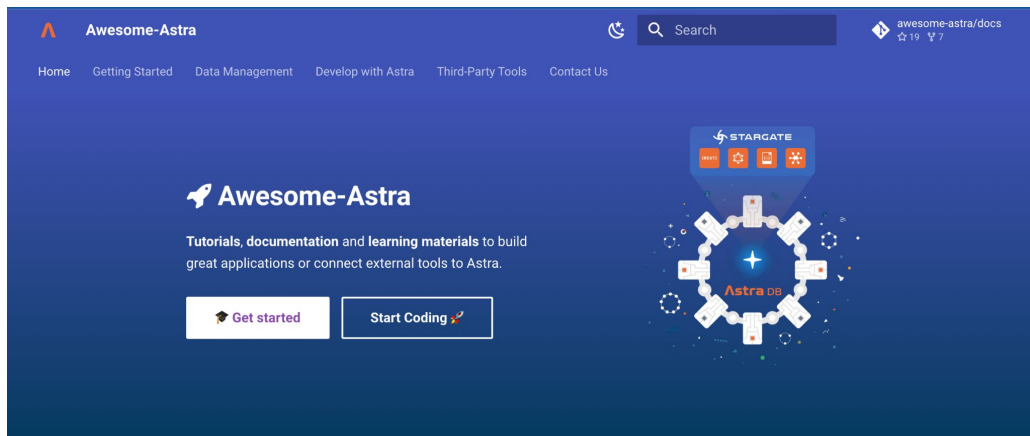
Wednesday at 2pm-US/CST



<https://twitch.tv/mgrygles>

How to start coding all of this?

Check out **Awesome-Astra**



<https://awesome-astra.github.io/docs/>

Benchmark report: Kafka vs Pulsar (from StreamNative)

<https://streamnative.io/pulsar/pulsar-vs-kafka>

Don't forget to
vote for this session
in the **GOTO Guide app**

DATASIX



Thank You

Mary Grygleski



<https://www.linkedin.com/in/mary-grygleski/>



[@mgrygles](https://twitter.com/mgrygles)



<https://discord.gg/RMU4Juw>

