Event Messaging and Streaming with Apache Pulsar

Mary Grygleski
@mgrygles

David Dieruf
@DierufDavid

Streaming Developer Advocates @ DataStax

August 2022

This slide deck can be accessed here:

https://bit.ly/3dDbiJS

Thank You to the Code PaLOUsa Sponsors

Friends of Code PaLOUsa

Developer Advocate

PULSAR

ddieruf

ddieruf

@dierufdavid

.NET

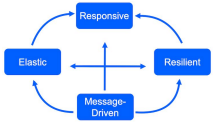
- Developer/Architect
- Specialize in cloud-native
- Kubernetes
- Application modernization
- CI/CD

David Dieruf

Senior Developer Advocate

Passionate Advocate

Java Champion

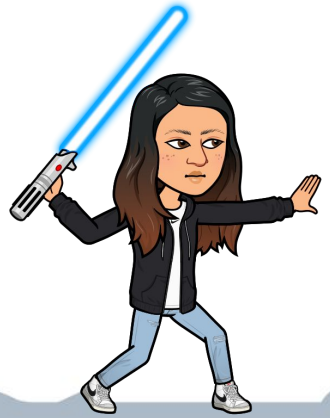


mgrygles

mary-grygleski

mgrygles

mgrygles



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

Mary Grygleski

DataStax Developers

AGENDA

The Many Facets of Computing Events

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

Event Processing Semantics

- Pub/Sub
- Queue

How can Event Streaming improve your systems?

A Brief Intro to Apache Pulsar

- Apache Zookeeper & Apache BookKeeper
- Highlighting a few Developer Features

Why Pulsar (extending Apache Kafka)

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***

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

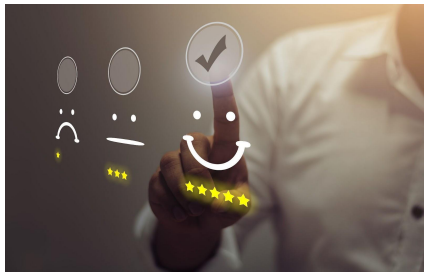
- Publishing Client sends the data
- Broker is the middle-person / agent
- Subscribing Client receives the data

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.

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

Apache Pulsar

Meet 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)

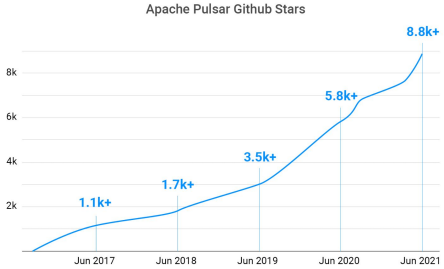
Tiered storage offloads

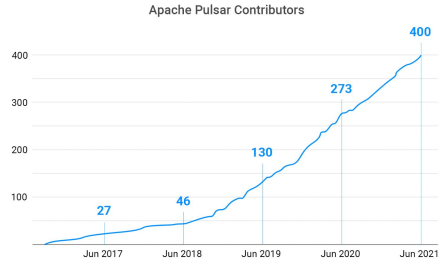
Offload data from hot/warm storage to cold/long-term storage when the data is aging out

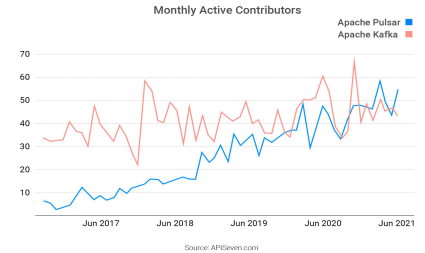
What is Apache Pulsar

- 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]

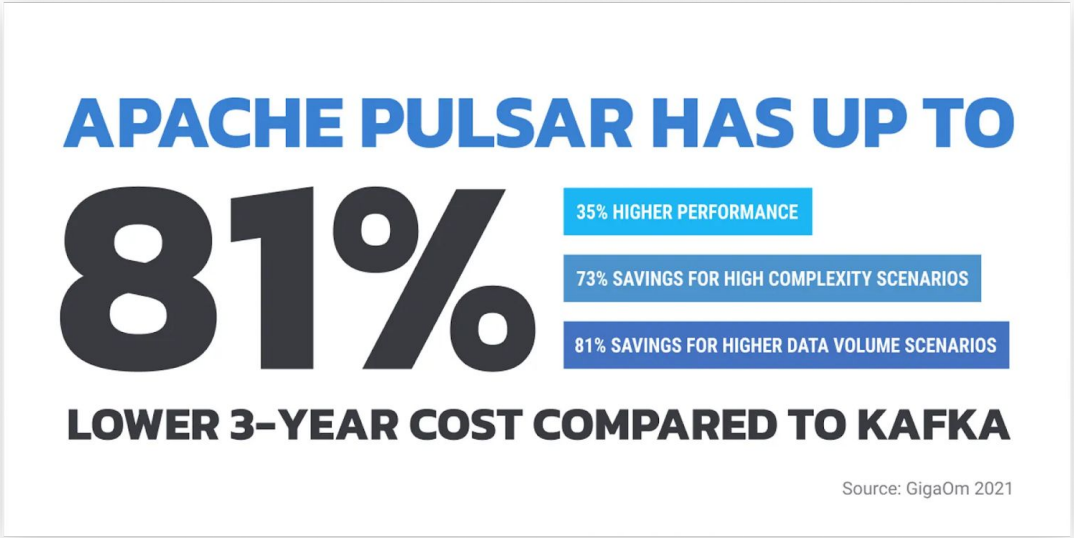






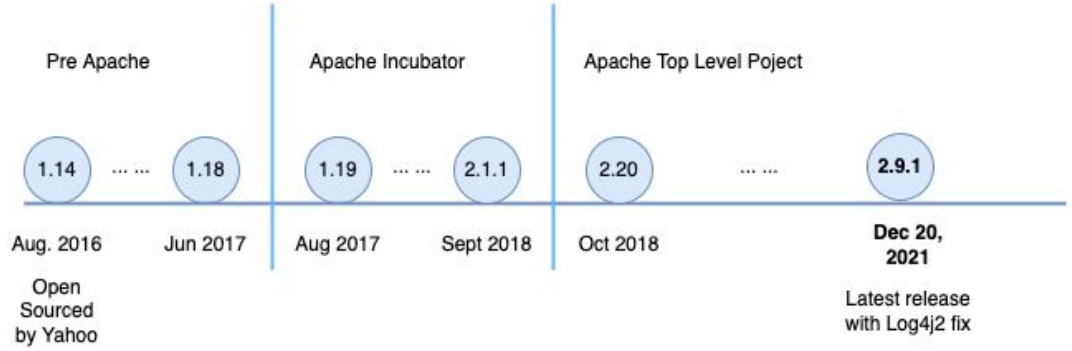
Who else is using Pulsar?

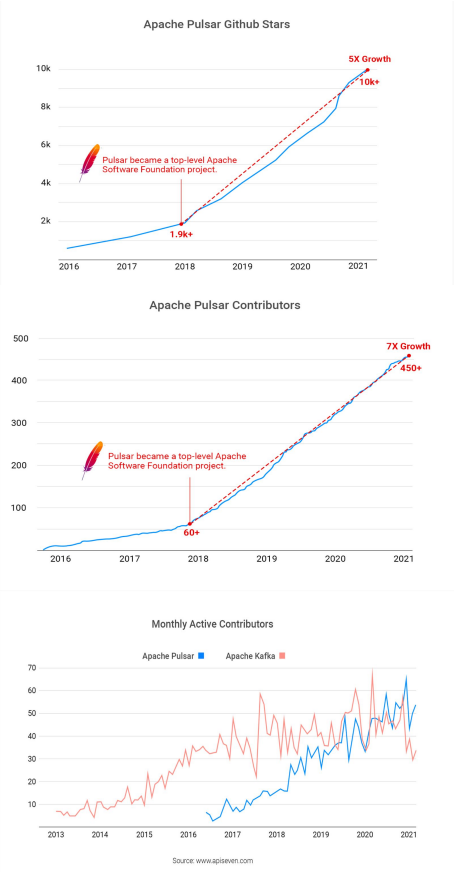
Apache Pulsar provides value across the enterprise



Brief History of Apache Pulsar

- **Cloud native**, **distributed, unified** messaging and streaming platform
- **Open source** as Apache TLP since Oct. 2018





Pulsar Is Different

Pulsar Component

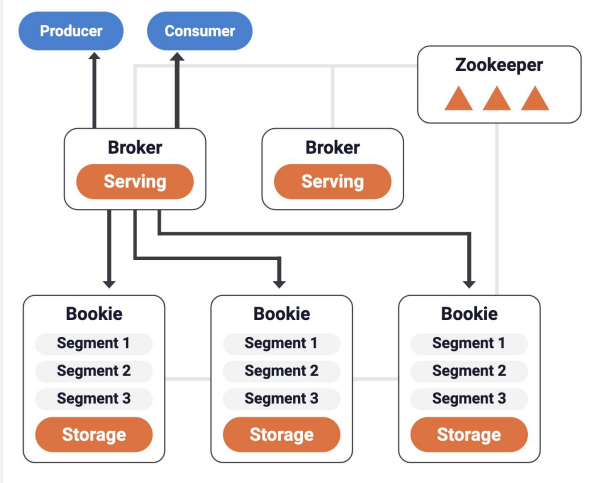
Producer
Client application sending messages to topic managed by Broker

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

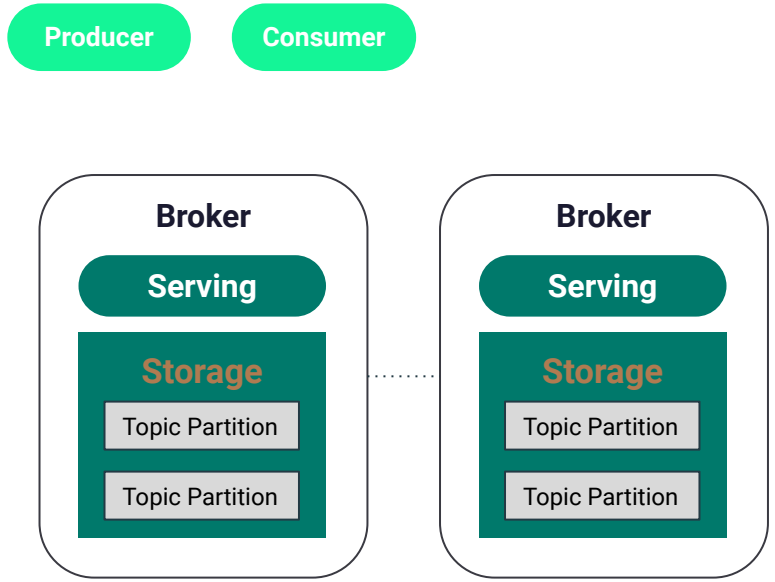
Partitioned topic abstraction masks complexity from consumers

Common Challenges

Scaling requires partition rebalancing

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

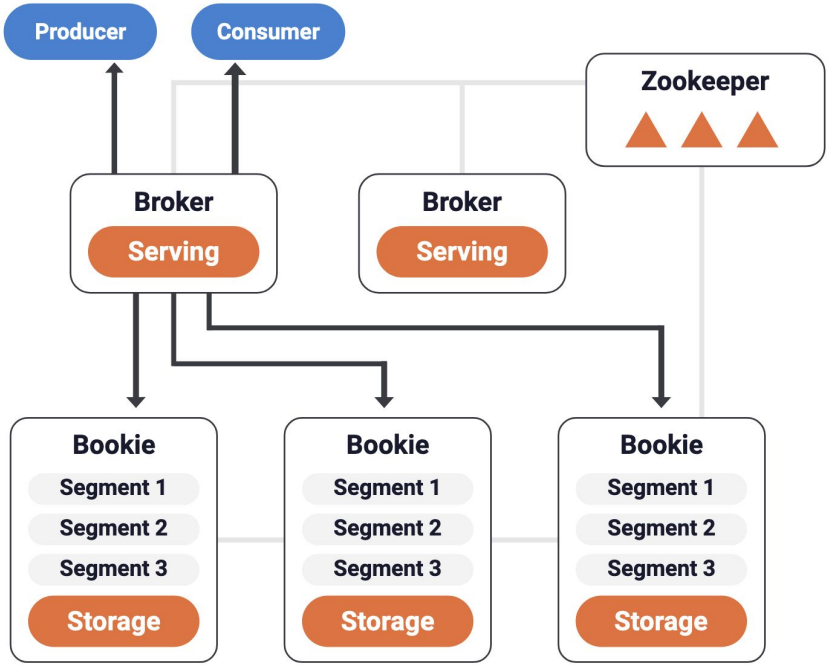
Trade offs to support partitioned topics came at the expense of messaging semantics needed for use cases such as queuing.



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

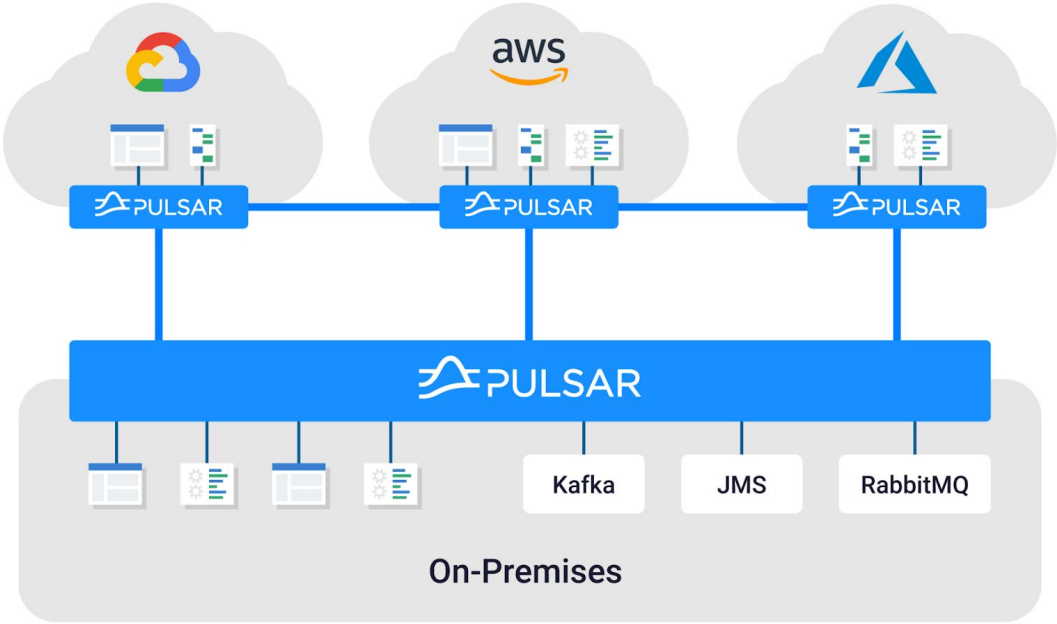
Apache Pulsar represents the **Next Generation of Enterprise Messaging**

Unified Solution for

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

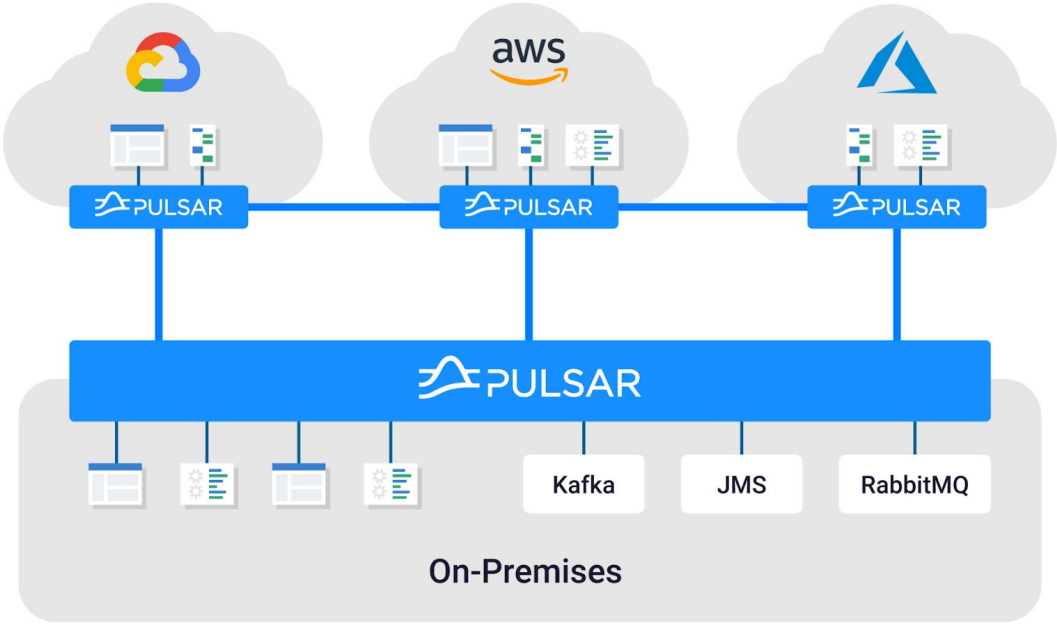
Out of the Box Capabilities Include

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



29

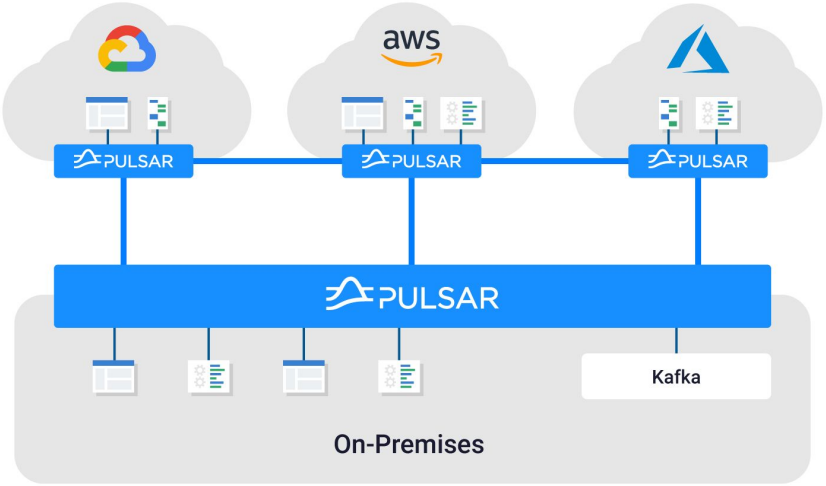
Pulsar Provides a Unified Platform for All Events in the Enterprise

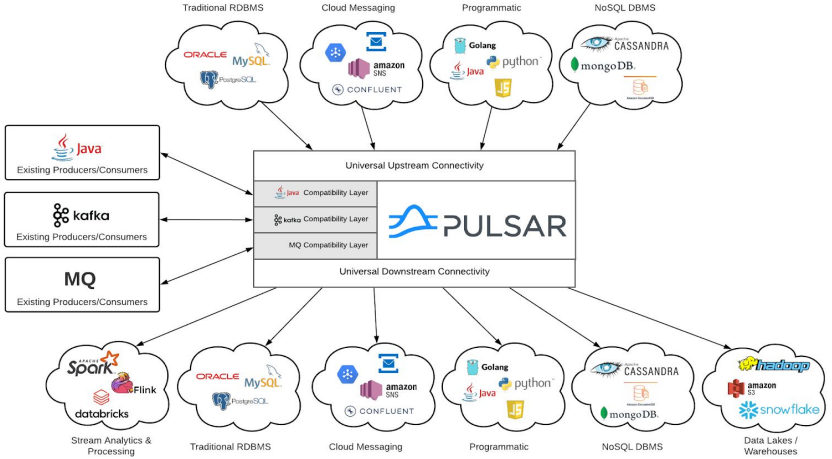


30

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 Puslar Different?

Pulsar's next generation architecture provides

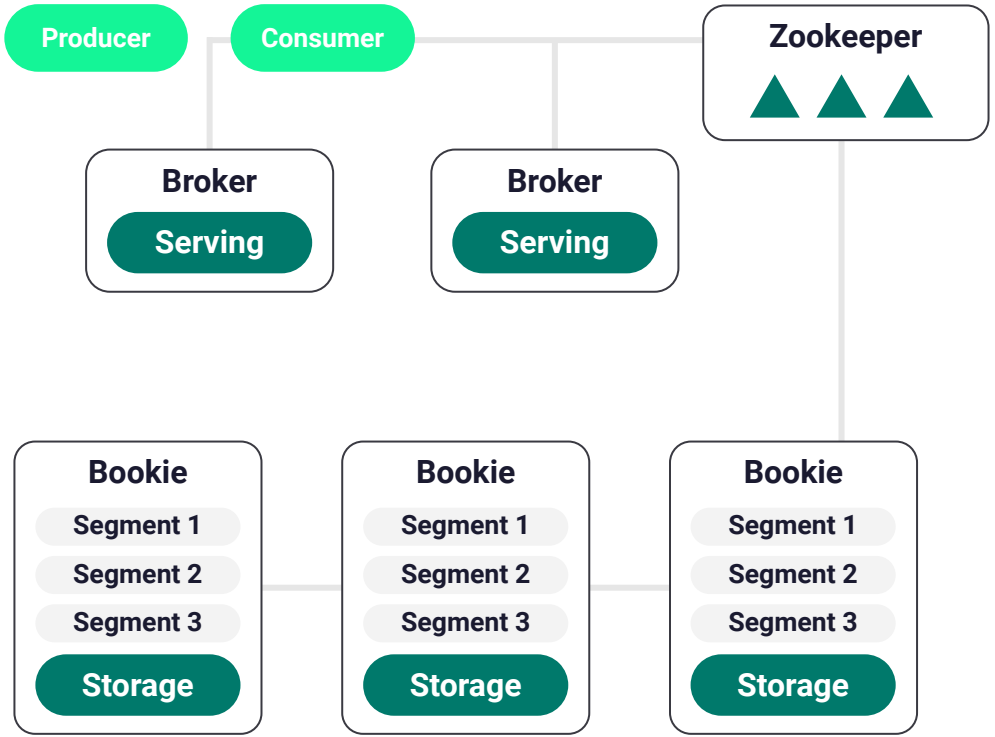
Distributed, tiered architecture

Separates compute from storage

Zookeeper holds metadata for the cluster

Stateless Broker handles producers and consumers

Storage is handled by Apache Bookkeeper



Rich ecosystem of connectors and clients

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

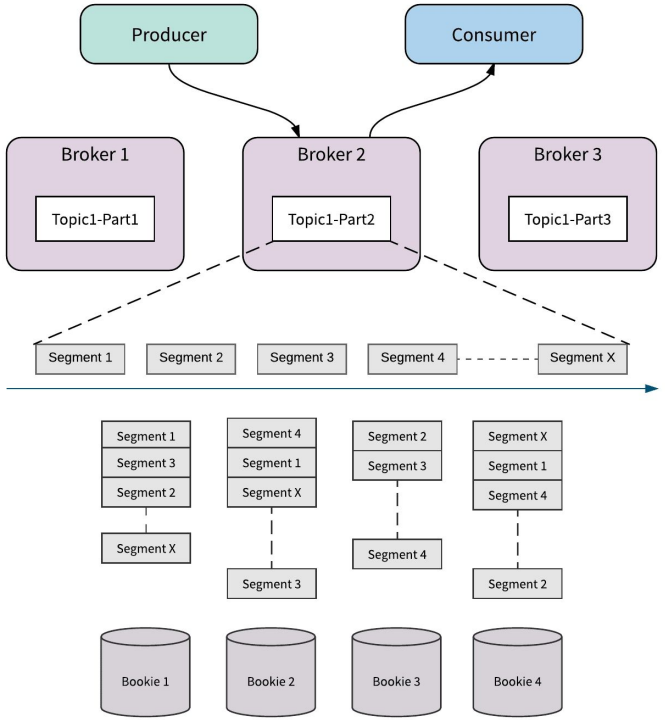
Role-based authorization providing control at the cluster, tenant, message producer and consumer level.

End-to-End Encryption

In transit TLS encryption and application managed content encryption.

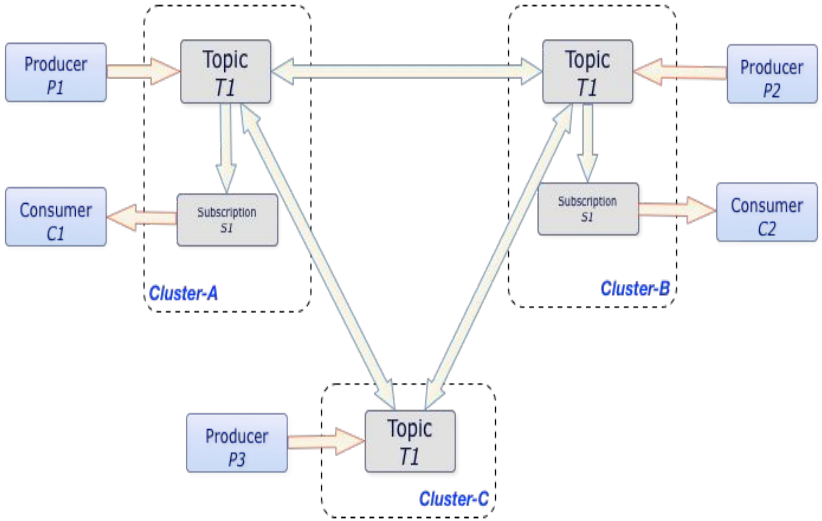
Key Differentiator 1: Separation between Compute and Storage

- 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**



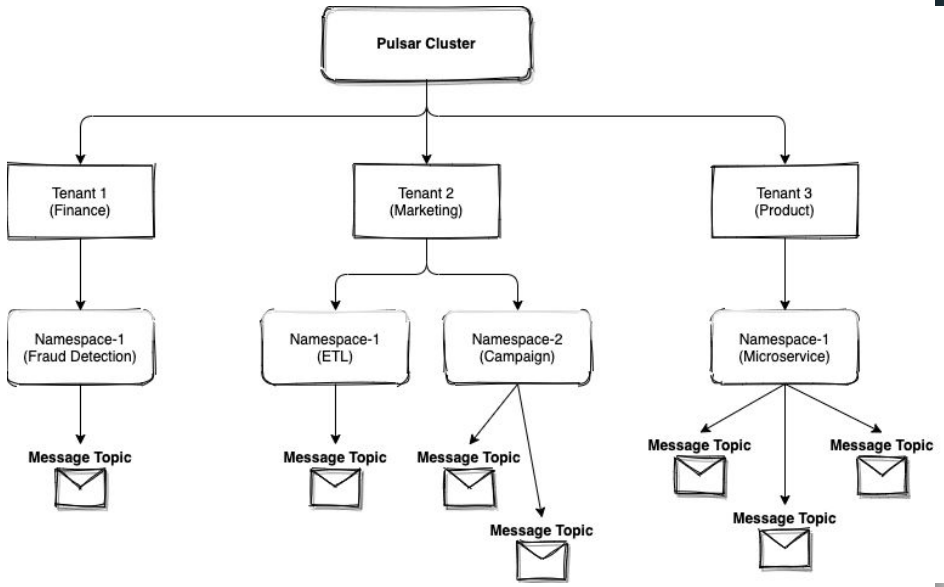
Key Differentiator 2: Native Geo-Replication

- 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



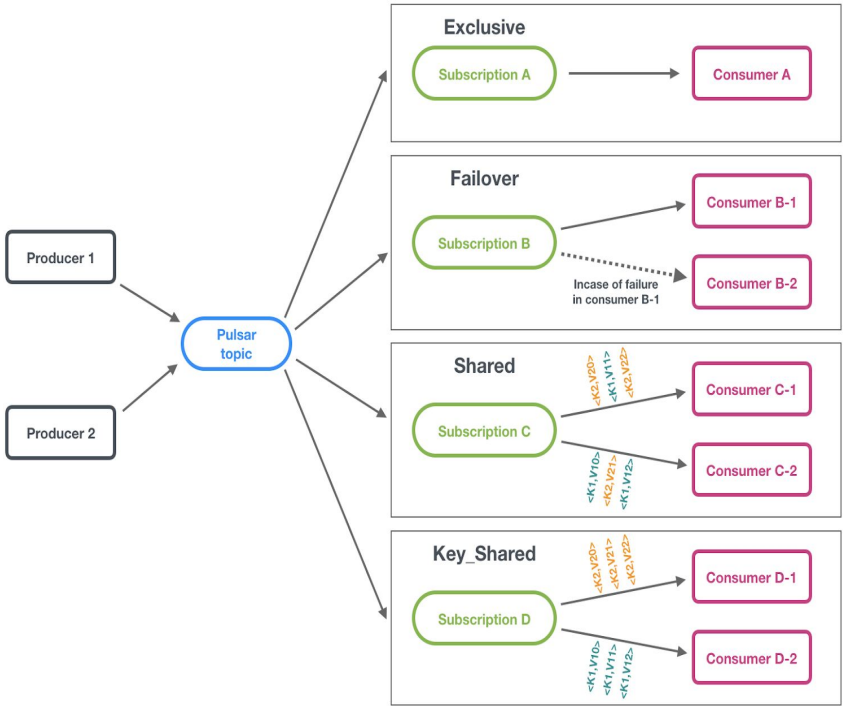
Key Differentiator 3 : Multi-Tenancy

- 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



So why Pulsar?

Pulsar is very flexible; it can act as a distributed log like Kafka or a pure messaging system like RabbitMQ. It has multiple types of subscriptions, several delivery guarantees, retention policies and several ways to deal with schema evolution.

When you should consider Pulsar

You need both queues like RabbitMQ and stream processing like Kafka.

You need easy geo-replication.

Multi tenancy is a must have and you want to secure the access for each of your teams.

You need to persist all your messages for a long time and you don’t want to off load them to another storage.

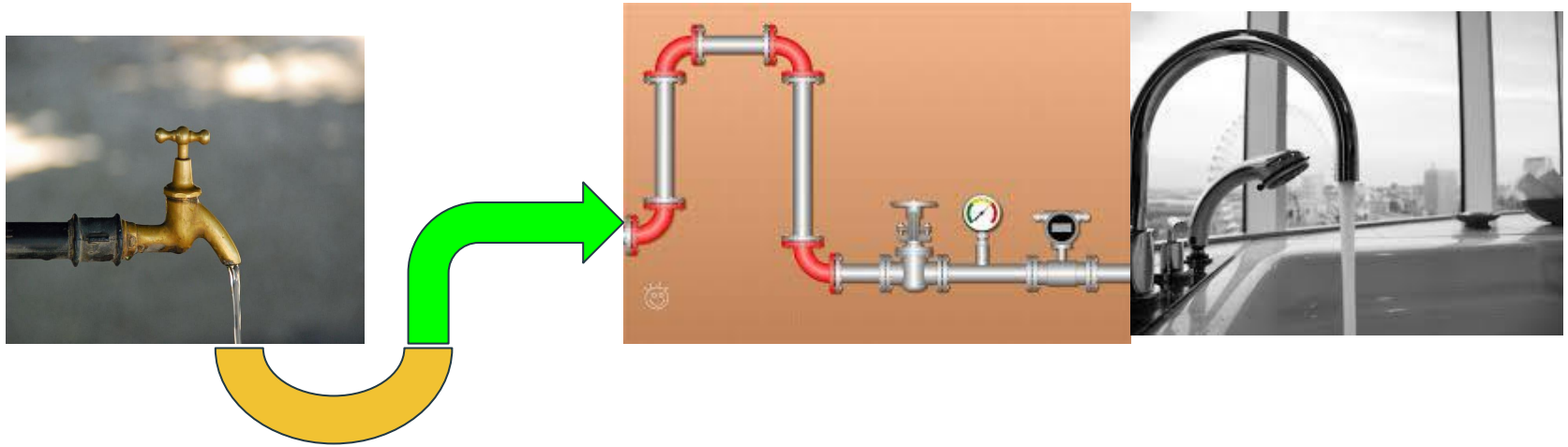
Performance is critical for you and your benchmarks have shown that Pulsar provides lower latency and higher throughput.

You run on-prem and you don’t have experience setting up Kafka but you have Hadoop experience.

Highlighting a few Pulsar Developer Features

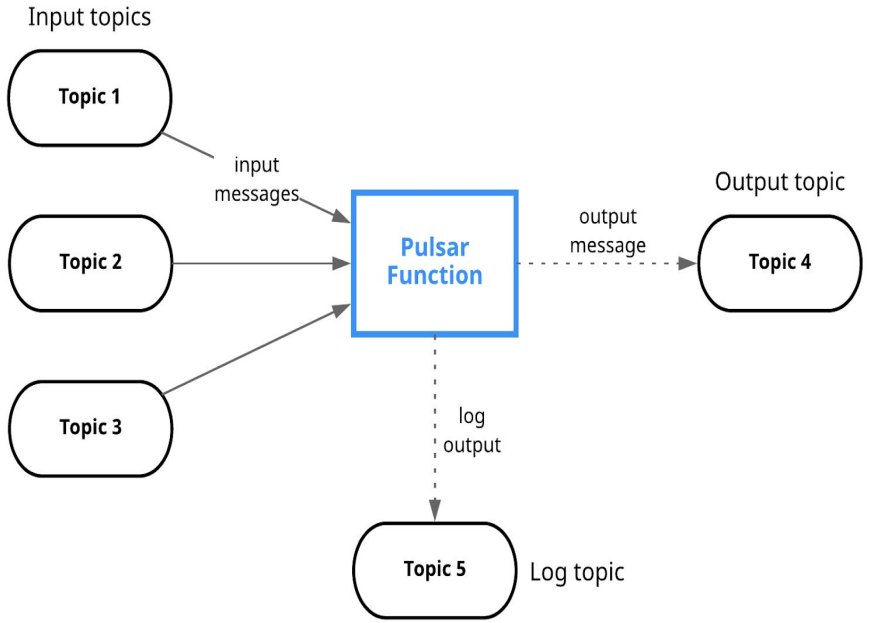
The Data Pipeline

“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

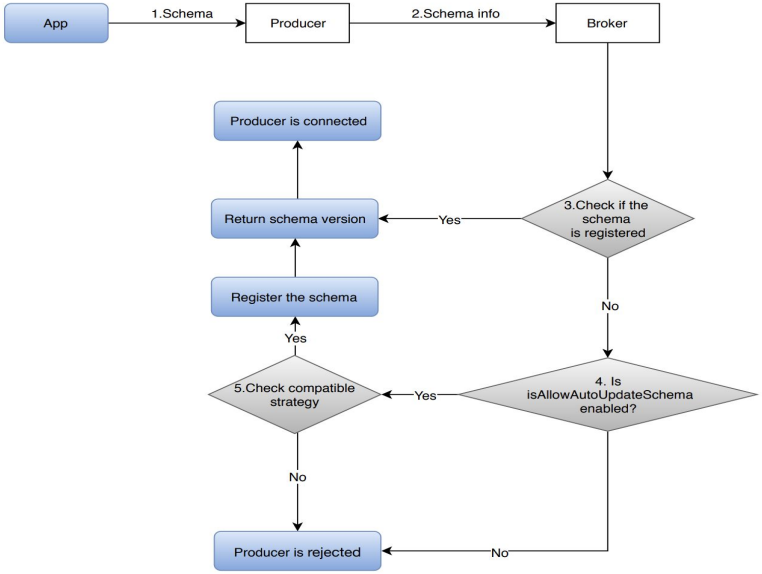


Pulsar Schema

- 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

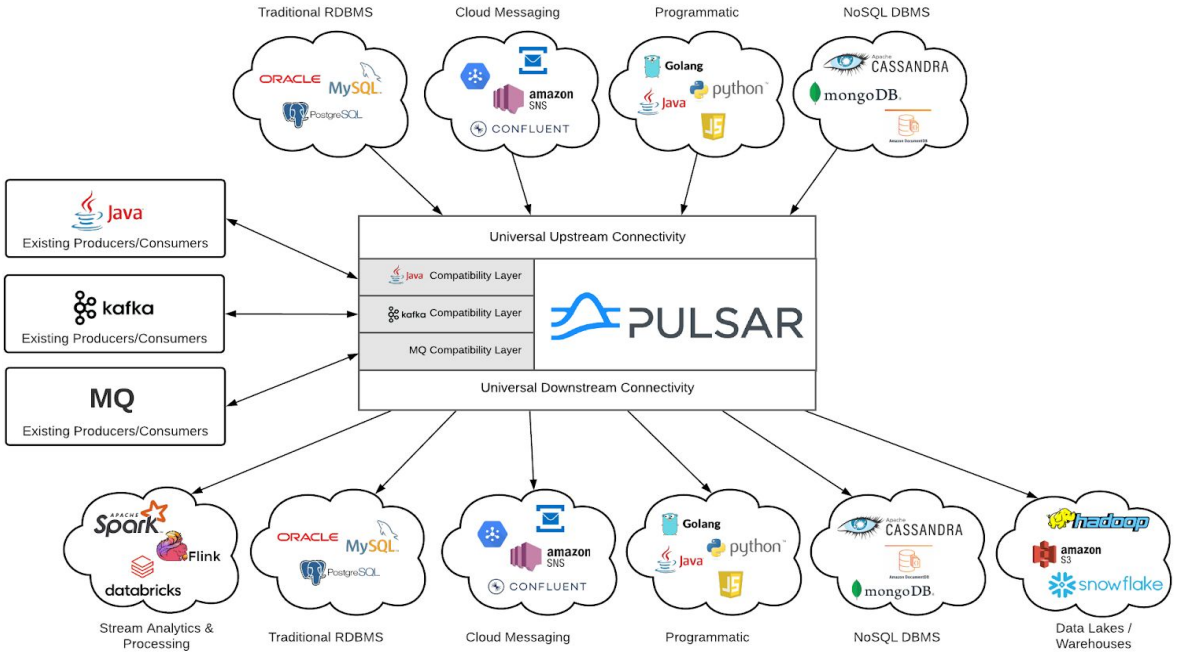
Build your own connectors

DataStax Flavors of Pulsar

DataStax is taking Pulsar 10x farther

A unified enterprise messaging backbone:

- Binary compatibility with JMS, MQ and Apache Kafka
- Built for real time data pipelines
- Natively multi-cloud
- Fully open source
- Built-in multi tenancy
- Elastic scalability
- Customer managed or fully hosted
- Self-service developer experience



** Forward looking product vision*

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

Cloud Offerings

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.

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.**

DataStax Streaming Offerings

Luna STREAMING	Astra Streaming
Customer Hosted	Fully Hosted SaaS
Open Source Distribution	Open Source Foundation
Optional Commercial Support	Fully Supported Offering
Free to Use	Serverless Consumption Pricing
Available Now	**Available Now!**

How is Pulsar different from another messaging platform?

Things to be aware of with Kafka…

Scaling Kafka is tricky, this is due to the coupled architecture where brokers also store data. Spinning off another broker means it has to replicate the topic partitions and replicas, which is time consuming.

No native multi-tenancy with complete isolation of tenants .

Storage can become quite expensive, and although you can store data for a long period of time, it is rarely used because of the cost implications.

It is possible to lose messages in case replicas are out of sync.

You must plan and calculate number of brokers, topics, partitions and replicas ahead of time (that fits planned future usage growth) to avoid scaling problems, this is extremely difficult.

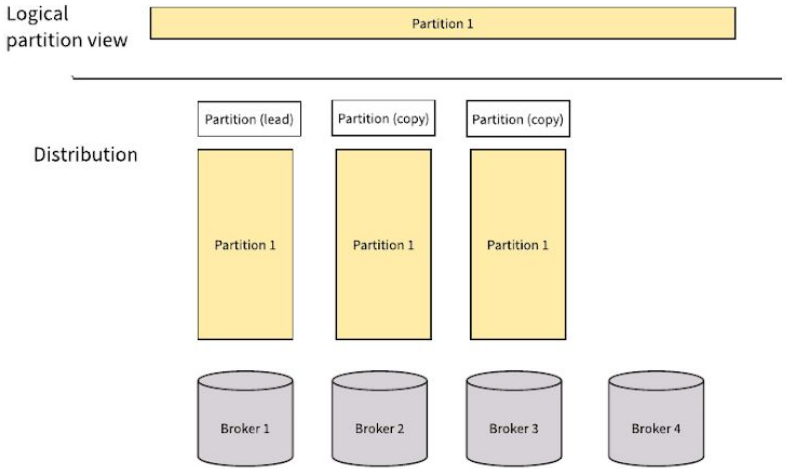
Working with offsets could be complicated if you just need a messaging system.

Cluster re-balancing can impact the performance of connected producers and consumers.

MirrorMaker Geo replication mechanism is problematic. Companies such Uber have created their own solution to overcome these issues.

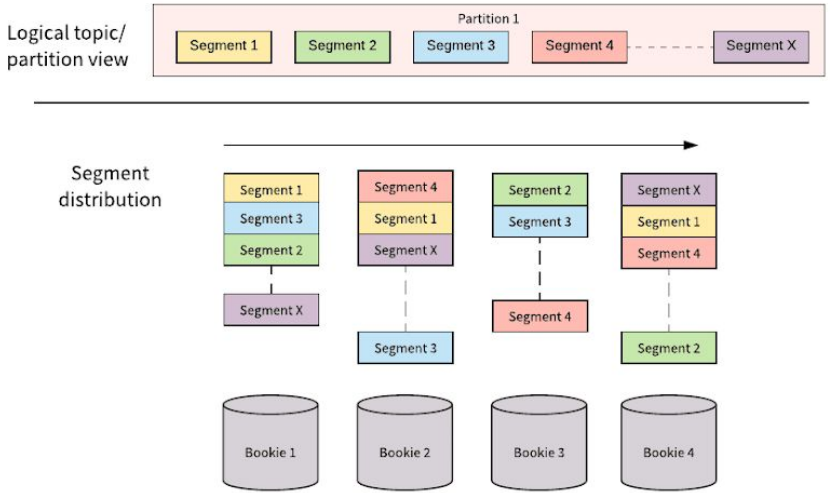
Partition-Centric vs. Segment-Centric

Apache Kafka



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

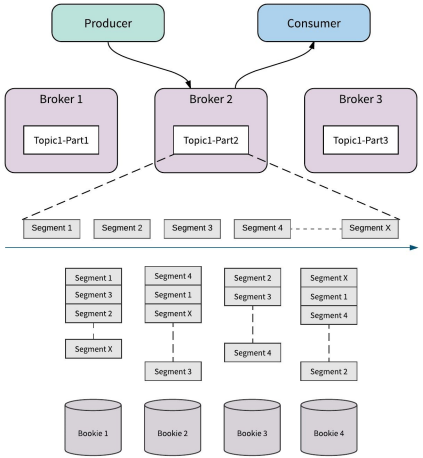
Apache Pulsar/BookKeeper



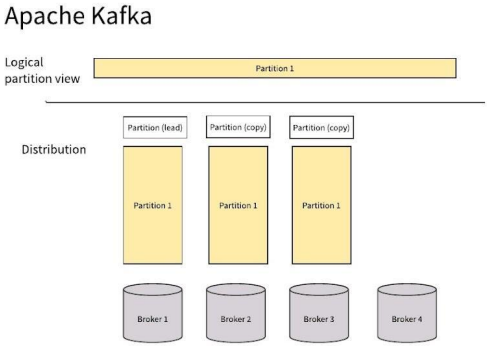
Pulsar/BookKeeper Stream — All log segment are replicated to a configurable number of bookies (replication = 3 here) across N possible bookies (N = 4 here). Log segments are evenly distributed to achive horizontal scalability with no rebalancing.

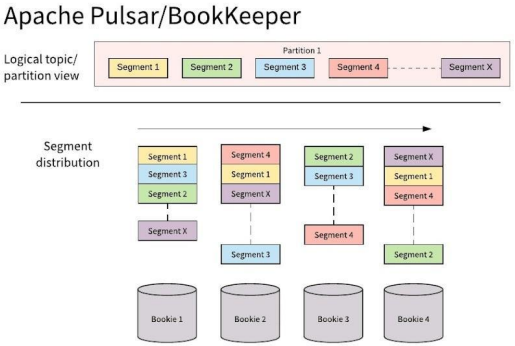
Architecture Advantage of Pulsar

- 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





Quick Demo

Where to go from here
and
Let’s keep in touch!

Resources

https://pulsar.apache.org/

https://bookkeeper.apache.org/

https://zookeeper.apache.org

Astra DB: **https://astra.datastax.com**

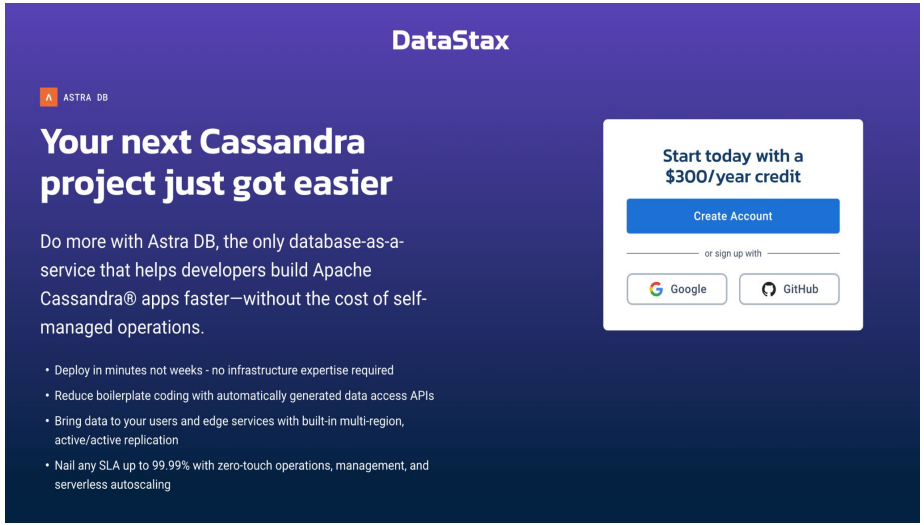
Astra Streaming:
https://www.datastax.com/products/astra-streaming

Luna Streaming:
https://www.datastax.com/products/luna-streaming

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

Astra DB

Managed Cassandra in the Cloud



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

Join us at the ‘hood

https://www.meetup.com/pro/apache-pulsar-neighborhood

Apache Pulsar
Neighborhood

https://pulsar-neighborhood.github.io/

THANK YOU

Mary Grygleski

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

@mgrygles

https://discord.gg/RMU4Juw

David Dieruf