Event-Driven Change Data Capture Pattern using Apache Pulsar

Mary Grygleski
Streaming Developer Advocate @ DataStax
@mgrygles

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 Streaming 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.

AGENDA

Why Change Data Capture (CDC) ?

- Serving Databases / Data Lakes / Data Warehouses
- What was there before CDC?
  - ETL and its drawbacks

Components of a CDC system

- Log-based CDC

Requirements for a Modern CDC system

- Cloud-Native
- Event-Driven

Introduction of Apache Pulsar

- CDC Support for Astra DB (Managed Cassandra)
- Quick Demo

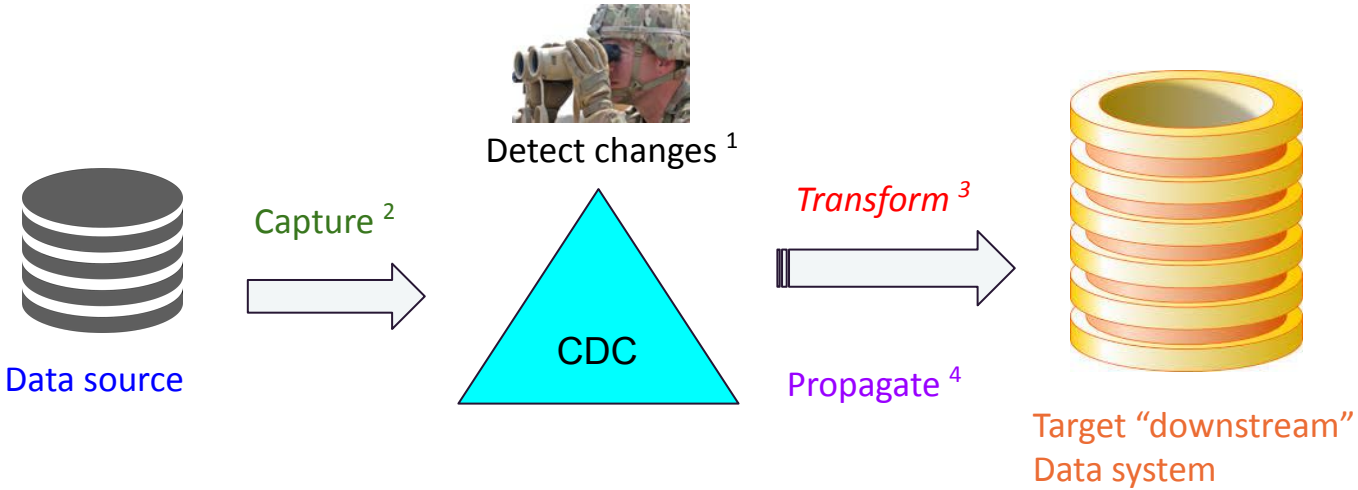
Why Change Data Capture ?

What is CDC for ?

- To serve Data Sources such as:
  - Databases
  - Data Lakes
  - Data Warehouses

- Detecting, capturing, transforming, and propagating changes in data sources to “downstream” data systems.

A simplified illustration of CDC



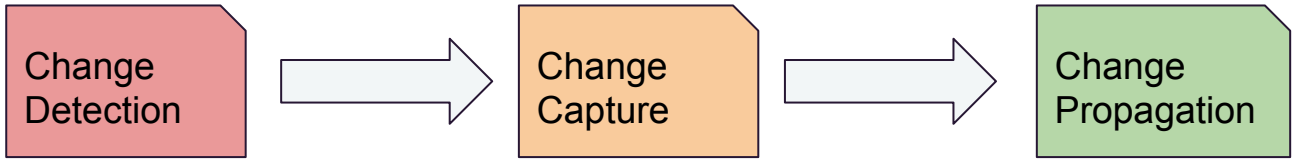
What was there before CDC ?

Disadvantages

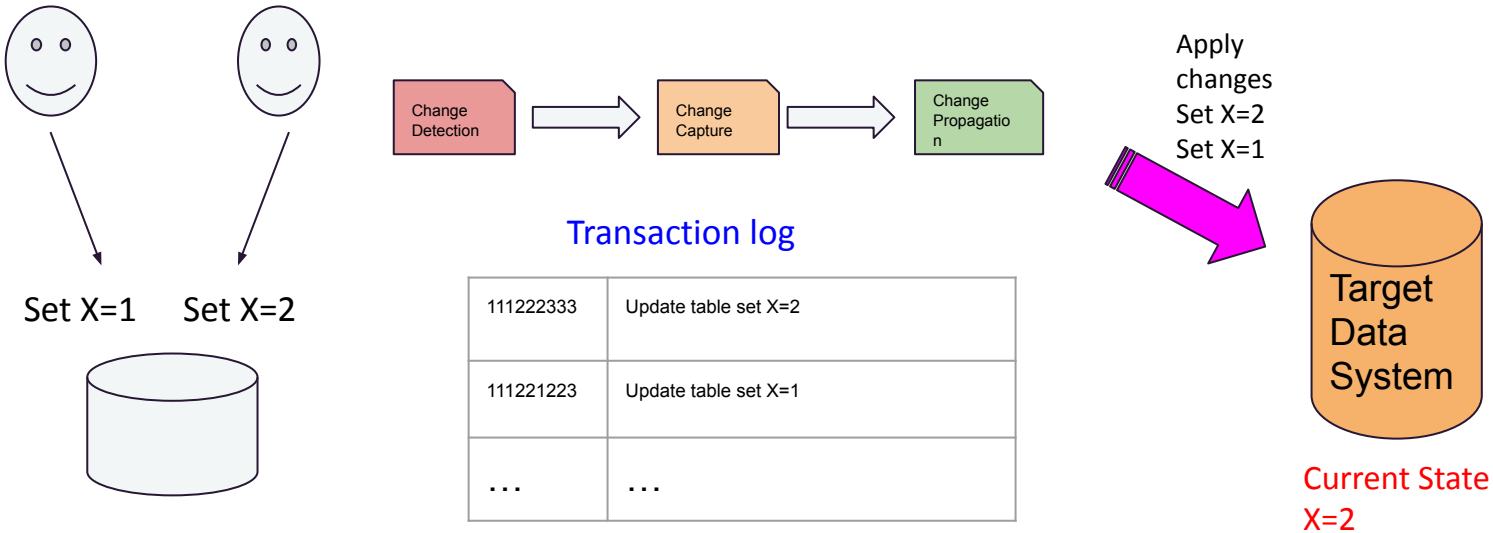
- Slow - “synchronous processing” - processing in batches
- Network bandwidth requirements for large sets of data
- Process setup tends to be heavy
- ETL tools are quite expensive
- Not able to keep up with the times

Components of a CDC System

What make up a CDC system?



Log-based CDC



Requirements for a Modern CDC System

Definition of Modern System

- Cloud-Native
- Responsive
- Scalable
- Resilient

Transmission of Data as Messages

- Reliability and resiliency (QoS, guaranteed delivery, etc)
- Responsive (Asynchronous, lightweight, loosely-coupled, etc)
- Scalability (Pub/Sub)
- Message ordering

These all require a paradigm shift!

How about taking the Event-Driven approach?

Introducing 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)

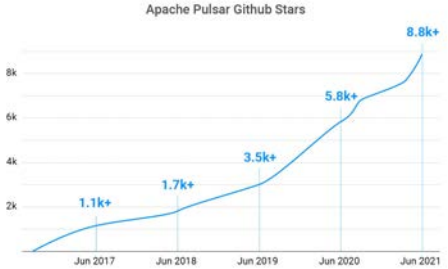
Tiered storage offloads

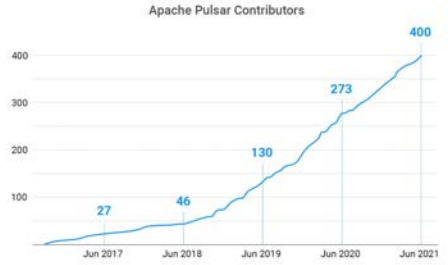
Offload data from hot/warm storage to cold/long-term storage when the data is aging out

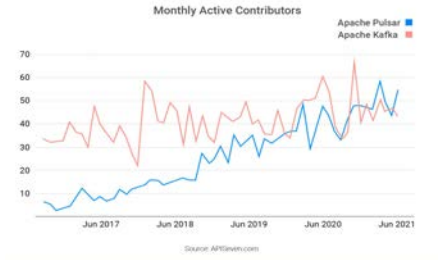
What is Apache Pulsar

- 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]

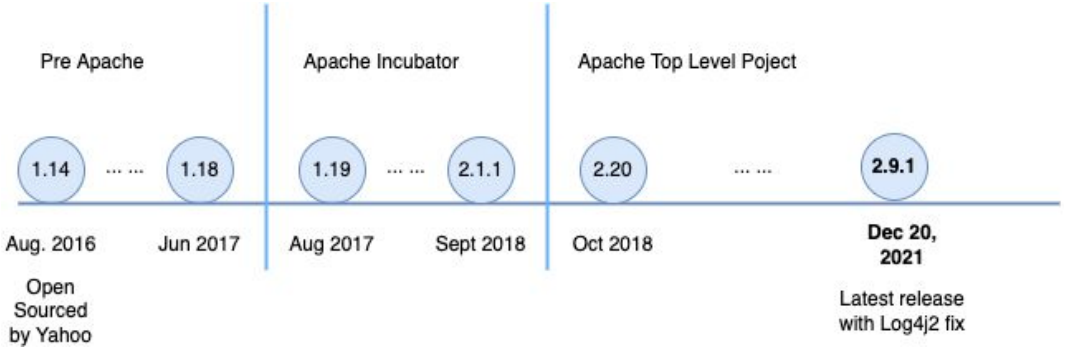


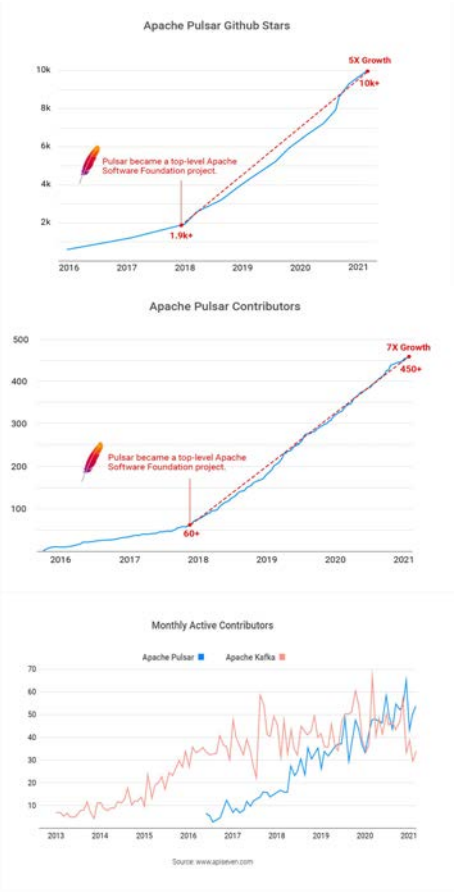




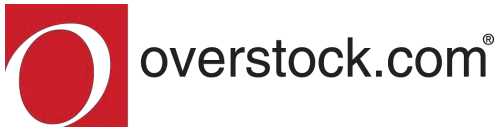
Brief History of Apache Pulsar

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





Who else is using Pulsar?



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

Pulsar Components

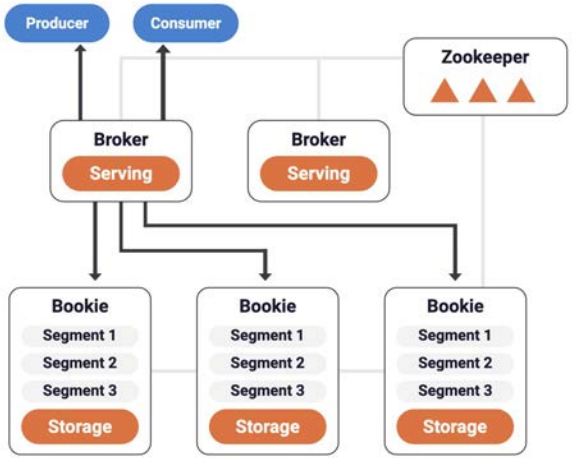
Producer
Client application sending messages to topic managed by Broker

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

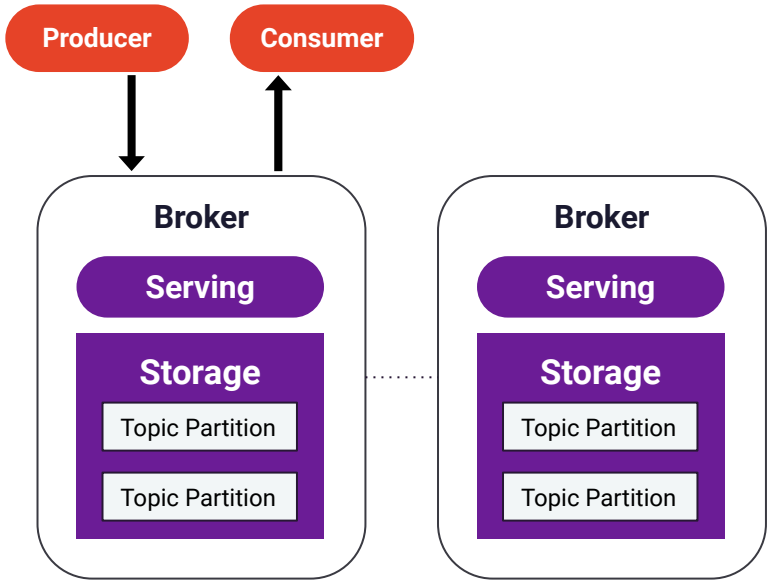
Partitioned topic abstraction masks complexity from consumers

Common Challenges

Scaling requires partition rebalancing

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

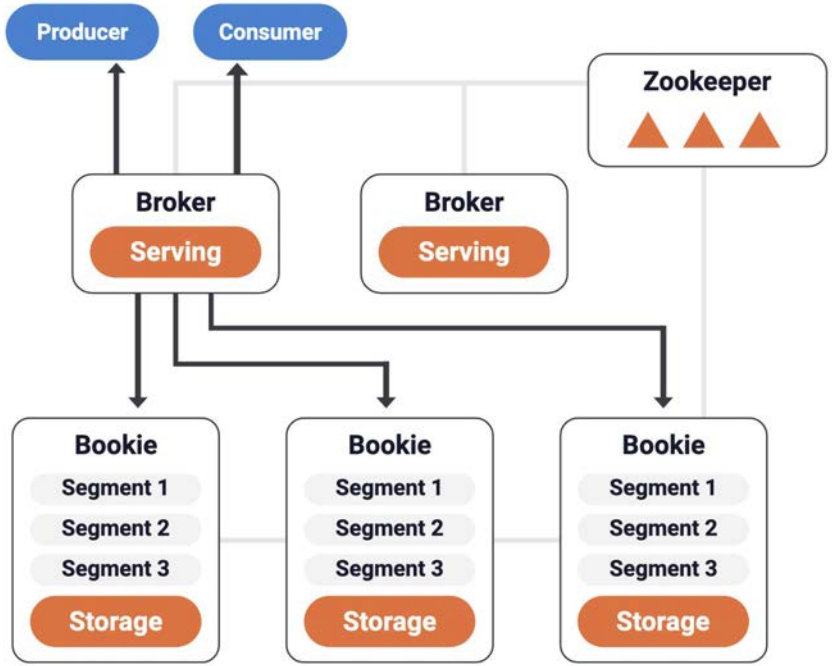
Trade offs to support partitioned topics came at the expense of messaging semantics needed for use cases such as queuing.



❯ 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

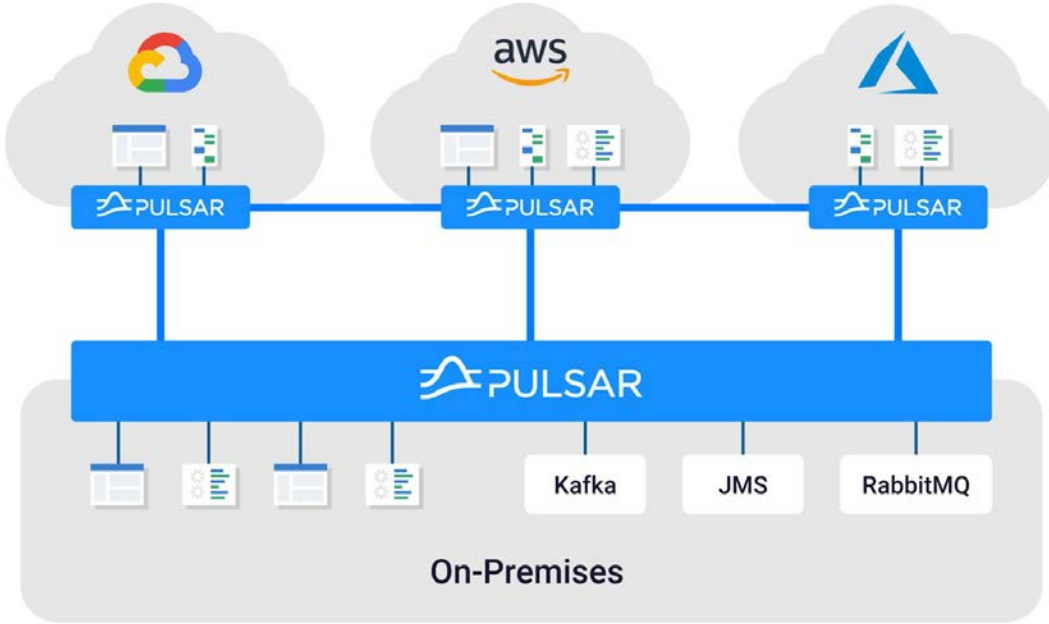
Apache Pulsar represents the **Next Generation of Enterprise Messaging**

Unified Solution for

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

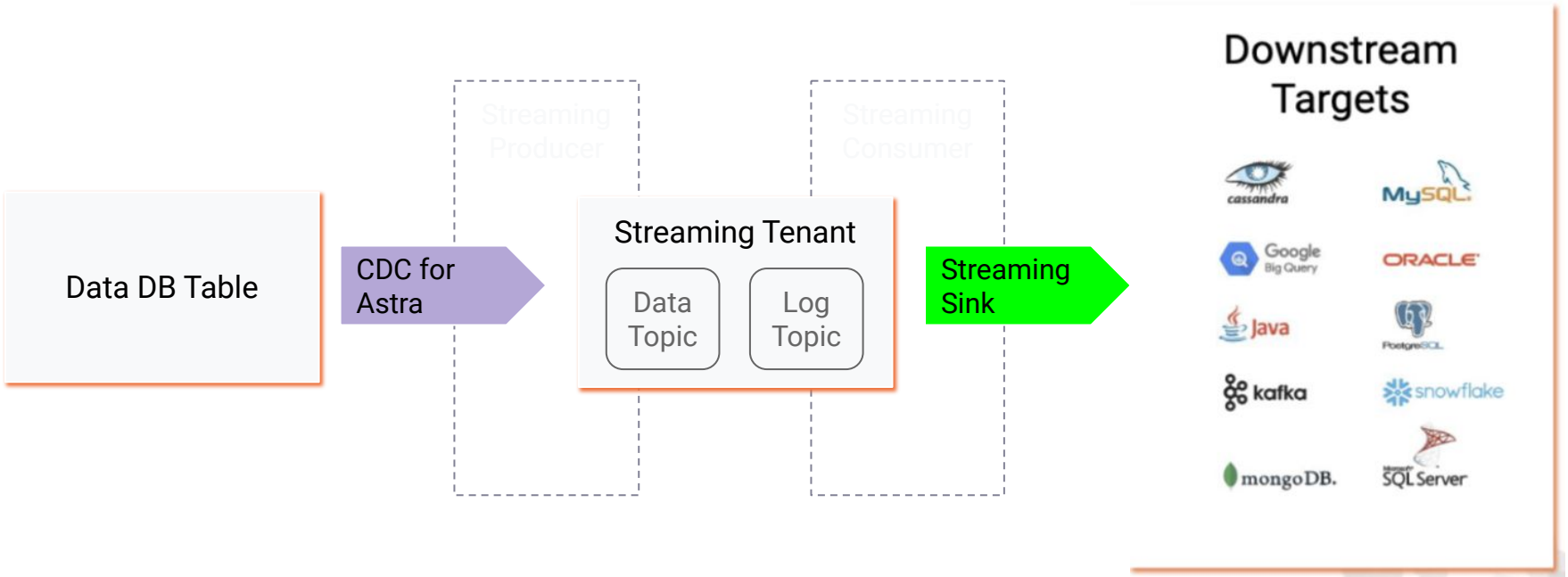
Out of the Box Capabilities Include

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



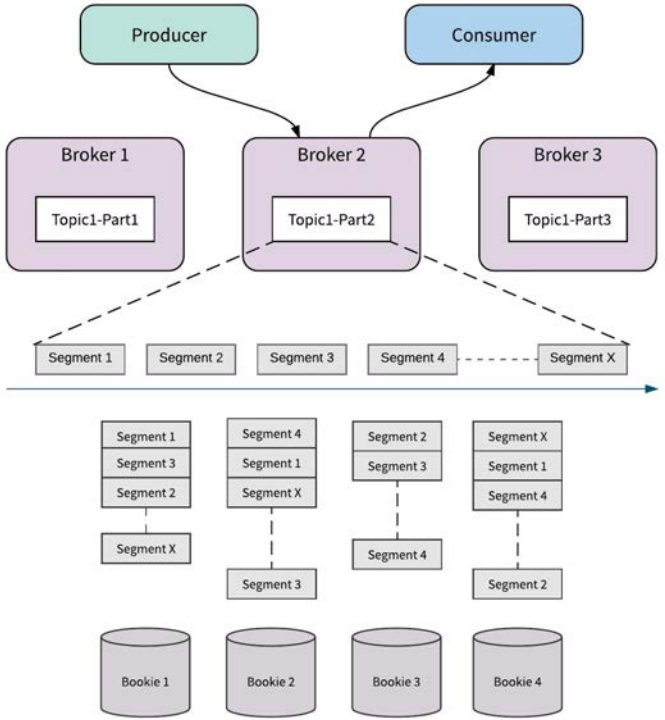
Enabling CDC for Astra DB

Change Data Capture (CDC) for DATASTAX ASTRA



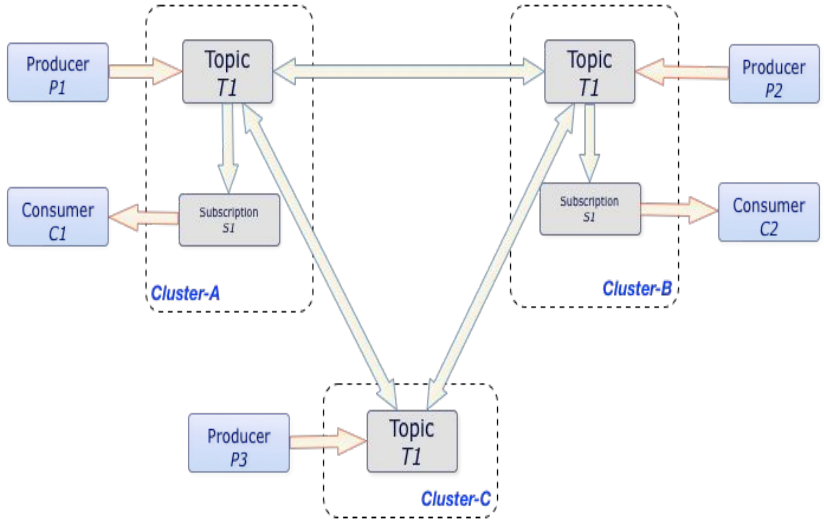
Key Differentiator 1: Separation between Compute and Storage

- 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**



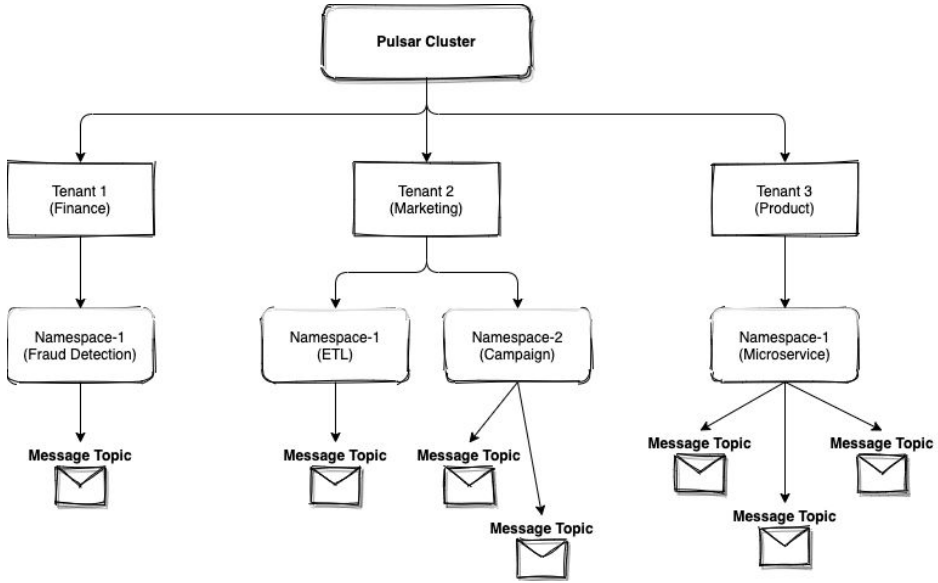
Key Differentiator 2: Native Geo-Replication

- 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



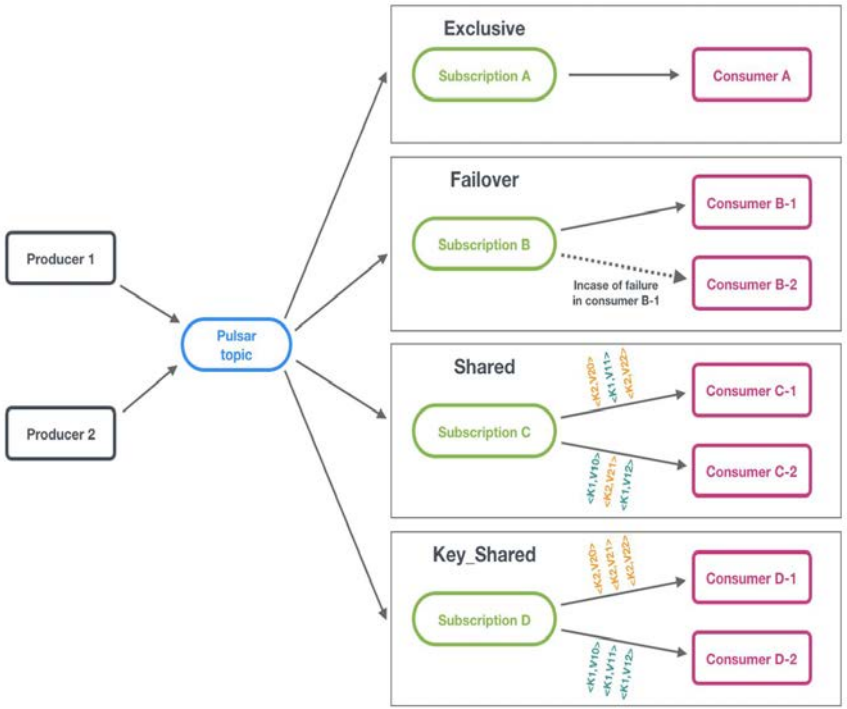
Key Differentiator 3 : Multi-Tenancy

- 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



Pulsar meets you where you are

Astra Streaming

Managed Pulsar

Explain the major selling point to Astra Streaming

Luna Streaming

Enterprise Support Pulsar

Explain Luna Streaming

Open Source

Community Driven Pulsar

Explain the open source project

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

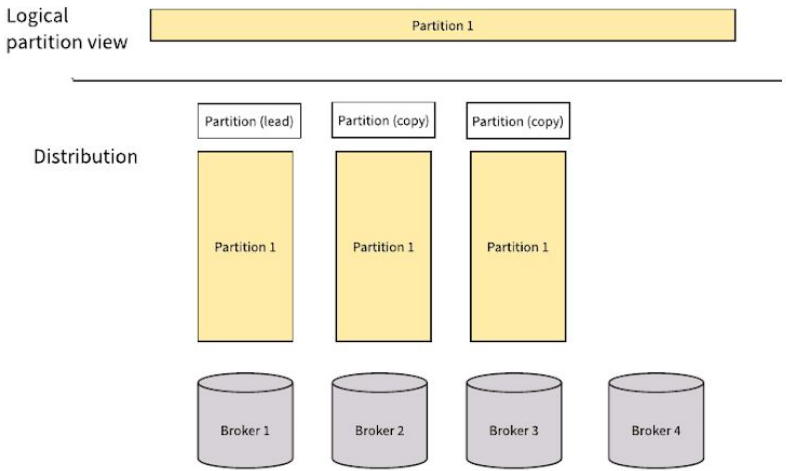
Apache Pulsar as a Service, fully cloud offering that is part of Astra.

CDC for Astra DB

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

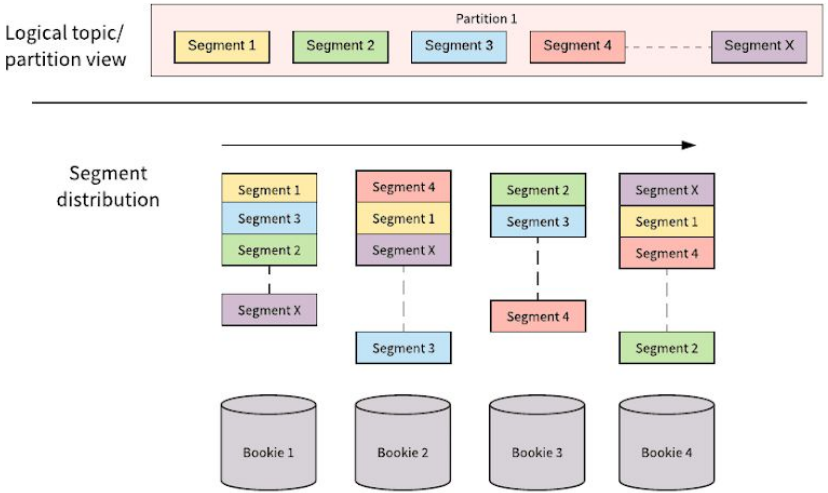
Partition-Centric vs. Segment-Centric

Apache Kafka



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

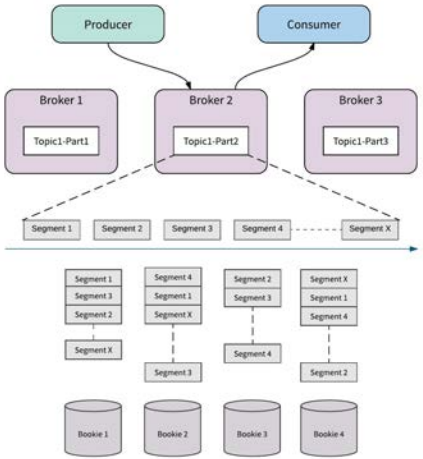
Apache Pulsar/BookKeeper



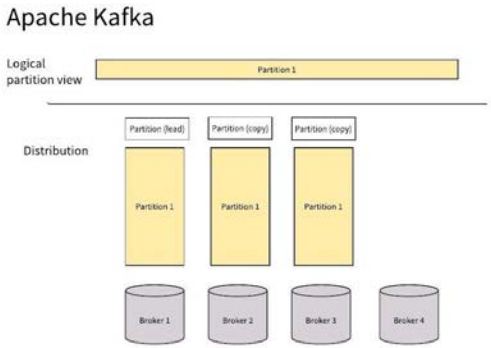
Pulsar/BookKeeper Stream — All log segment are replicated to a configurable number of bookies (replication = 3 here) across N possible bookies (N = 4 here). Log segments are evenly distributed to achive horizontal scalability with no rebalancing.

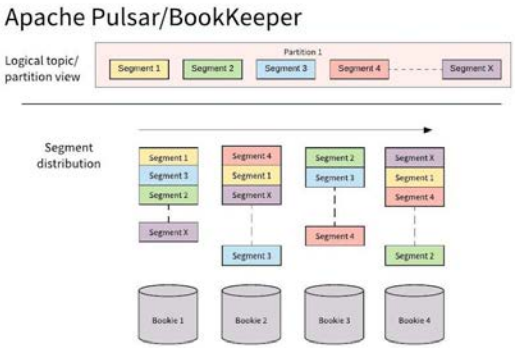
Architecture Advantage of Pulsar

- 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 - Apache Pulsar and Astra from DataStax

PULSAR https://pulsar.apache.org/

 https://bookkeeper.apache.org/

https://zookeeper.apache.org

DATASTAX ASTRA DB https://astra.datastax.com

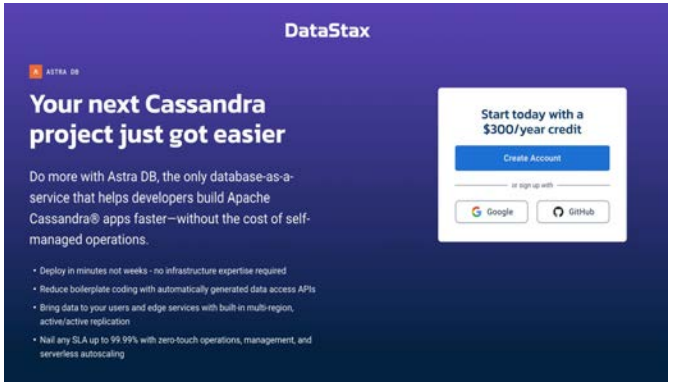
DATASTAX ASTRA STREAMING https://www.datastax.com/products/astra-streaming

DATASTAX LUNA STREAMING https://www.datastax.com/products/luna-streaming

DATASTAX ASTRA

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

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

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

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

JAKARTA EE Jakarta JMS 2.0 speficifications - https://jakarta.ee/specifications/messaging/2.0/

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

THANK YOU

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

@mgrygles

https://discord.gg/RMU4Juw