Apache Pulsar, a next-generation streaming engine

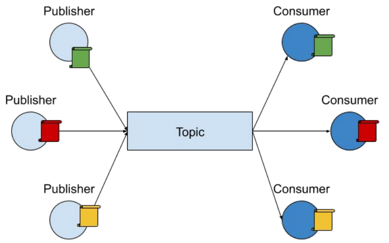
Chris Bartholomew, CEO and Founder Kesque

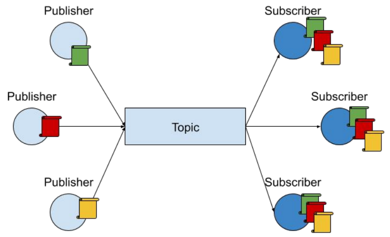
What is Apache Pulsar

- Distributed pub-sub messaging system
  - High throughput, low latency
  - Separates compute from storage
  - Horizontally scalable
  - Streaming and queuing
- Open source
  - Originally developed at Yahoo!
  - Contributed to the Apache Software Foundation (ASF) in 2016
  - Top-level project
  - 6.6K GitHub stars, over 300 contributors

Multiple message exchange patterns

- Pub-sub (fanout)
- High volume streaming (clickstream, logs, metrics)
- Event driven architectures (event sourcing)
- Work queues (competing consumers)
- Message bus for microservices
- Message retention
- Message replay

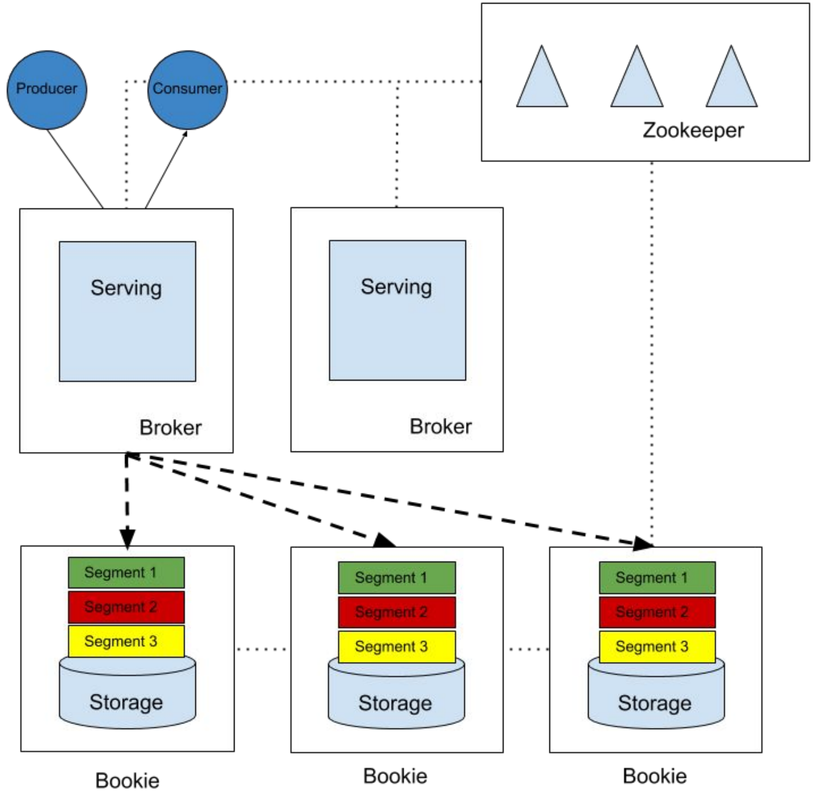




KESQUE

Architecture

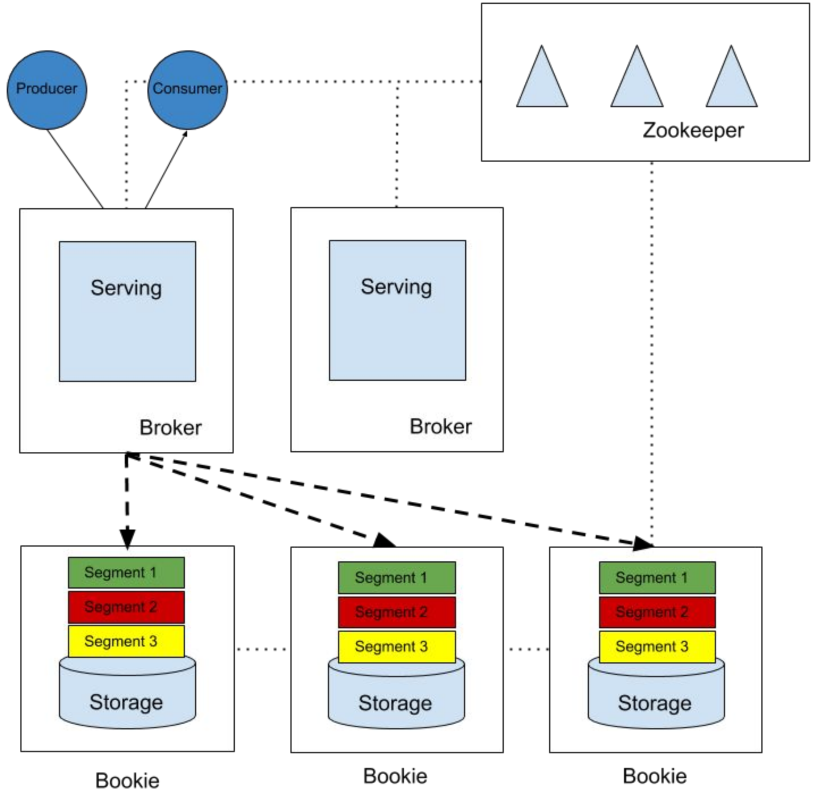
- Distributed, tiered architecture
- Separates compute from storage
- ZooKeeper holds metadata for the cluster
- Stateless Broker handles producers and consumers
- Storage is handled by Apache BookKeeper



KESQUE

Architecture (cont'd)

- BookKeeper distributed, append-only log
- Data is broken into segments written to multiple bookies
- Producer acknowledged when quorum of bookies acknowledge
- No single bookie holds entire log



KESQUE

Cloud native

- Separation of compute and storage suited to cloud
- Brokers keep no state
- Brokers and BookKeeper nodes are horizontally scalable
- Storage layer (BookKeeper) scales independently of the compute layer (Broker)
- Commonly deployed in Kubernetes
    - External access problem solved by Pulsar Proxy

KESQUE

Performance

- Designed for low latency, high throughput
- Data is cached on the broker for “tailing reads”
- Writing and reading is isolated on Bookies using 2 disks (journal and ledgers)
- Optimized for flushing each message to disk for maximum durability

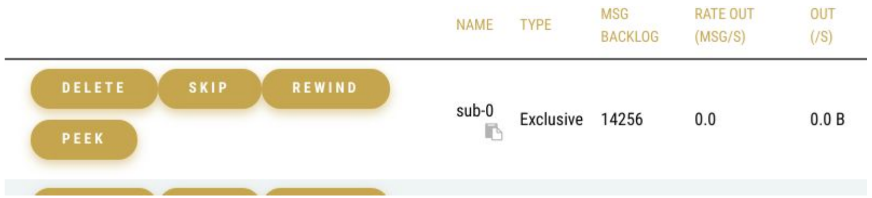
Clients

- Apache clients
  - Java
  - Python
  - C++
  - Go (C++ wrapper)
  - Native Go
  - Node.js
  - C#
  - WebSocket

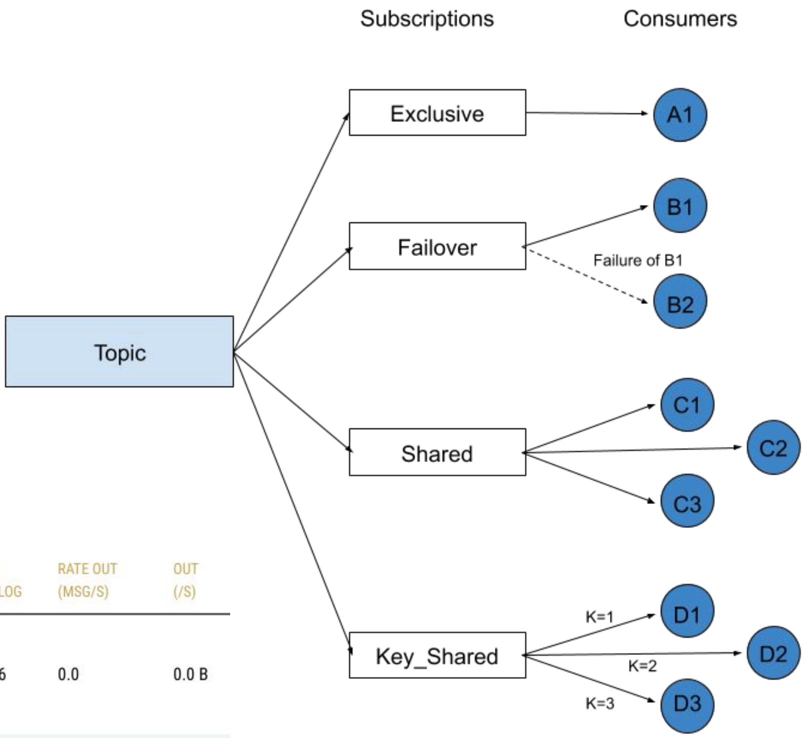
- Community clients
  - .NET
  - Scala
  - Rust
  - HTTP
  - Haskell

Subscriptions

- Durable
- Multiple subscriptions per topic
- Multiple consumers per subscription
- Exclusive, failover, shared, key_shared
- Skip and rewind in subscription



	NAME	TYPE	MSG BACKLOG	RATE OUT (MSG/S)	OUT (/S)
DELETE SKIP REWIND PEEK	sub-0	Exclusive	14256	0.0	0.0 B

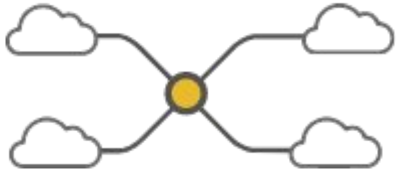


KESQUE

Partitions

- Not necessarily partitions, but partitions if necessary
- Supports partitions for ordering guarantees at partition level
- Partitions are also unit of parallelism so allow for scaling high volume topics
- Partitions hash by key, round robin, or custom
- Can dynamically add partitions to a topic

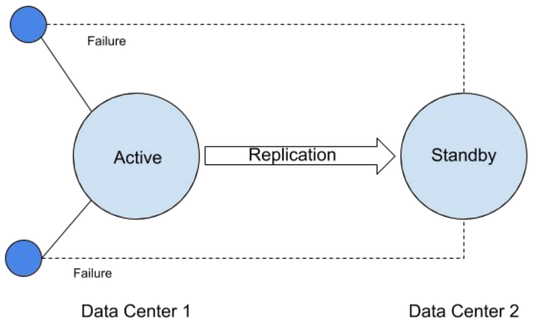
Multi-tenancy

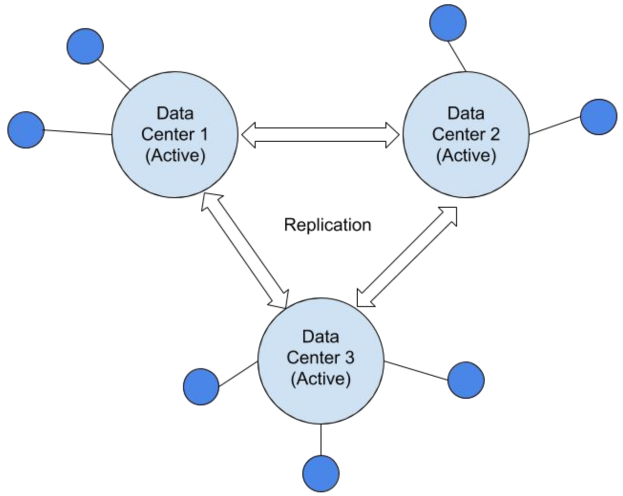


- Support multiple user groups on single cluster
- Built-in tenant with authorization
- Tenants can be further divided into namespaces with authorization
- Authentication using tokens, client certificates, Athenz (Yahoo! Open source)
- Policies defined at namespace level
  - Max producers/consumers
  - Max rate
  - Storage/backlog

Geo-replication

- Built-in geo replication
- Managed through Pulsar CLI or REST API
- Multiple topologies
  - Active/standby
  - Active/active
- Shared configuration
- Replicated subscriptions





KESQUE

Tiered Storage

- Pulsar is tiered: compute, storage
- Further tier in storage
  - Offload older messages
  - S3, Google Cloud Storage, HDFS
- Transparent to client
- Supports event sourcing
- Savings:
  - Cloud storage is significantly cheaper than SSDs
  - Further tiering (infrequent access, glacier)

“We're also looking to Pulsar to solve the problem of a never-ending log of messages for our large-scale data systems where events are expected to persist indefinitely and subscribers are able to start consuming messages retrospectively.”

ThoughtWorks Technology Radar
https://www.thoughtworks.com/radar/platforms/apache-pulsar

KESQUE

Advanced Queue Capabilities

- Negative acknowledgment
    - Temporary failure
    - Put message back in topic
    - Consumed by different client
- Dead letter topic
    - Permanent failure
    - Send to topic to not block processing
- Delayed delivery
    - Temporary failure, retry later
    - Scheduled events

KESQUE

Pulsar SQL

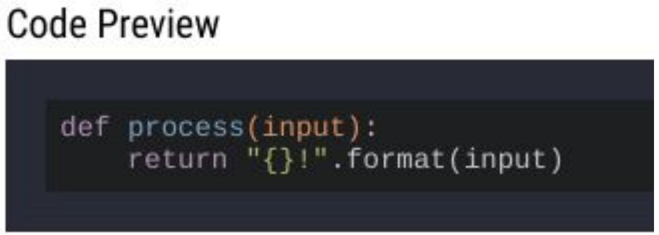
- BookKeeper stores all data
- Integration between Presto and BookKeeper
- Perform SQL queries on messages stored on BookKeeper nodes
- Querying data at rest, not data in motion

Schema Registry

- Type saftey is important when producers/consumers are decoupled
- Built-in schema registry
- Clients/REST API can register a schema for a topic
  - Primative: string, integer
  - Complex: Key/Value, Avro, JSON
- Pulsar stores the current schema, ensures producers and consumers conform
- Supports schema evolution

Functions

- Can create small functions that run per message
- Java, Python, and Go
- Upload to cluster and connect to topics
- Can do per-message routing, cleaning, enrichment
- Simple, Lambda-like
- For more complex tasks can use stream processing tool (ex Apache Flink)


Code Preview

```
def process(input):
    return "{}!".format(input)
```

Sinks/Sources (Connectors)

- Connector framework runs inside Pulsar cluster
- Built-in connectors and custom
- Kafka, RabbitMQ
- JDBC to mysql, postgresql, mongoDB, etc
- Change data capture from mysql, postgresql, mongoDB using debezium

KESQUE

Summary

- Apache open source
- Distributed and horizontally scalable
- Cloud/Kubernetes friendly
- Can replace both Kafka and RabbitMQ
- Optimized for high throughput, low latency
- Tiered storage
- Functions and connectors
- Other feature: multi-tenancy, geo-replication, SQL queries, schema registry

Thanks!

Email: chris.bartholomew@kesque.com

Web page: https://pulsar.apache.org/

GitHub: https://github.com/apache/pulsar

Apache Pulsar Slack: https://apache-pulsar.herokuapp.com/

KESQUE