Pub/Sub and Kafka

Troy Raen
University of Pittsburgh | Pitt-Google Broker

LSST Broker Technical Workshop | virtual | November 8, 2021

Outline

1. Problem setup:

   Requirements for
   Ingesting and redistributing live alert streams at LSST scale

2. Comparison of solutions:

   Apache Kafka and Google Cloud Pub/Sub

3. Solutions talking to each other:

   Kafka ⟷ Pub/Sub connector

Preview Slide 10

You are invited to add your Broker’s info.

Problem Statement:

Brokers need to handle high-volume alert streams

1. Consume alert streams from LSST's Kafka broker (required)
   - **10,000,000 alerts/night**
   - 300 alerts/sec average in 10 hr night
   - (What max rates should we expect on a regular basis?)
2. Publish alert streams
   - Apache Kafka is the traditional choice for producing alert streams in astronomy
   - Pub/Sub is an alternative offering different benefits
   - Brokers can use both
3. Facilitate users receiving and processing the streams

Problem Statement:

Brokers need to handle high-volume alert streams

1. Consume alert streams from LSST's Kafka b
   - 10,000,000 alerts/night
   - 300 alerts/sec average in 10 hr night
   - (What max rates should we expect on a regular bas
2. Publish alert streams
   - **Apache Kafka** is the tr ditional choice for producin
   - **Pub/Sub** is an alternat e offering different benefits
   - Brokers can use both
3. Facilitate users receiving and processing the streams

Requirements for a
message streaming platform/service:

Consume and produce streams
- **at these rates,**
- with minimal lag,
- at reasonable cost,
- **in a format useful to**
  - **internal processing**
  - **end-user astronomers**

Outline

1. Problem setup:

   Requirements for
   Ingesting and redistributing live alert streams at LSST scale

2. Comparison of solutions:

   Apache Kafka and Google Cloud Pub/Sub

3. Solutions talking to each other:

   Kafka ⟷ Pub/Sub connector

Comparing Kafka and Pub/Sub

(Alt title: Intro to Pub/Sub via comparison with Kafka)

	Kafka	Pub/Sub
Function	**Produce and consume live message streams**	**Produce and consume live message streams**
What is it?		
Infrastructure + Software req'd		
Scalability		
Features		
Message Anatomy		
Workflow		
Extendability		
End-user APIs		
Cost		

explored in the following slides

Detailed comparison at cloud.google.com/architecture/migrating-from-kafka-to-pubsub

Comparing Kafka and Pub/Sub: **What is it?**

Kafka

kafka.apache.org

"open-source distributed event streaming **platform**"

"used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications"

Pub/Sub

cloud.google.com/pubsub/docs/overview

"asynchronous messaging **service**"

"used for streaming analytics and data integration pipelines to ingest and distribute data... equally effective as messaging-oriented middleware for service integration or as a queue to parallelize tasks."

Comparing Kafka and Pub/Sub: **Infrastructure + Software req'd**

Kafka

Data storage, delivery, auth managed by user producing the stream.

Publishing at scale requires

Minimum(?): **Server + Confluent Platform**

Common(?): **Cluster + Confluent Platform + managed solution**

Zookeeper, Confluent Cloud, Kubernetes

+ network bandwidth to distribute to all consumers

Pub/Sub

Data storage, delivery, auth managed by Pub/Sub.

Publishing at scale requires

Google Cloud project & credentials + **API**

e.g., `pip install google-cloud-pubsub`

+ network bandwidth to publish once, to Pub/Sub

Who's using what?

	Ingest LSST/ZTF streams	Communicate Internally	Produce streams	Manage
ALeRCE	[Kafka]	[Kafka]	[Kafka]	Zookeeper MirrorMaker
AMPEL	[Kafka]	?		None? (single container)
ANTARES	[Kafka]	Kafka-ElasticSearch connector? + others?	[Kafka]	Zookeeper
Babamul	Fritz (->Kowalski->`confluent-kafka-python`->`librdkafka`)	(Fritz)		(Fritz)
Fink	[Kafka]	Spark-Kafka connector (+ others?)	[Kafka]	Zookeeper
Lasair	[Kafka]	[Kafka] + others	[Kafka]	MirrorMaker
Pitt-Google	Confluent Platform + Kafka-Pub/Sub connector	Pub/Sub push subscriptions + python client	Pub/Sub python client (->gRPC)	none
SNAPS				
POI/Variables				

Comparing Kafka and Pub/Sub: **Authentication**

Kafka

Obtain an account + credentials from the Broker

SSL, SASL, OAuth, …

Producer choice

Pub/Sub

Obtain account + credentials from Google Cloud

OAuth, service account tokens, …

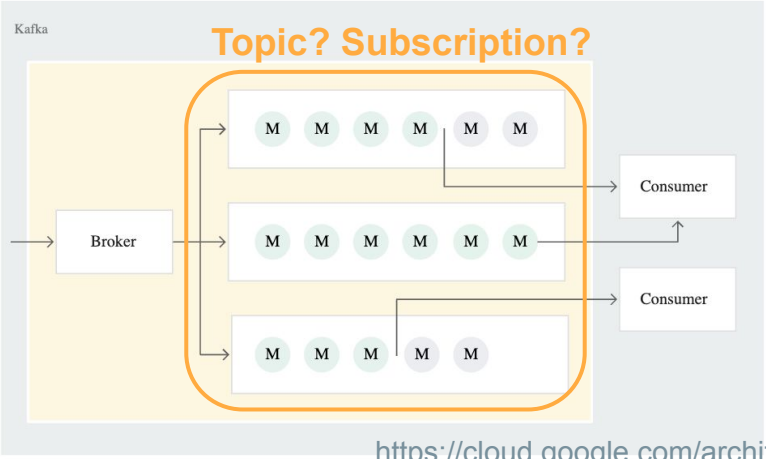
Consumer choice

Comparing Kafka and Pub/Sub: **Scalability**

Kafka

Horizontally scalable.

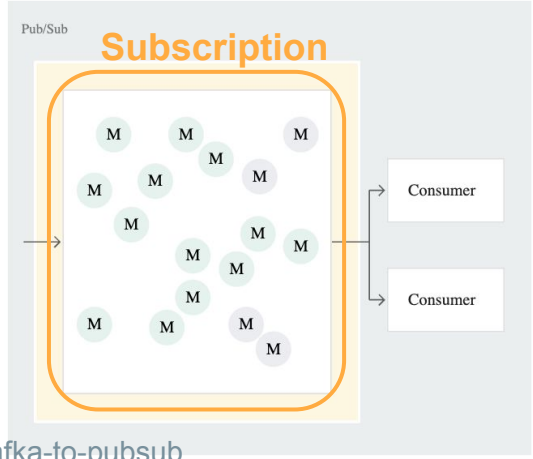
Increase the number of partitions (and machines, if necessary) to handle increased demand.



Pub/Sub

Horizontally scalable.

Happens automatically in response to demand, **handled by Pub/Sub**. *Producers are independent from consumers.*



https://cloud.google.com/architecture/migrating-from-kafka-to-pubsub

Comparing Kafka and Pub/Sub: **Features**

	Kafka	Pub/Sub
Message ordering	Yes within partitions	Yes within topics
Message deduplication	Yes	Yes using Dataflow
Push subscriptions	No	Yes
Unprocessed message queue	As of version 2.0	Yes
Transactions	Yes	No
Message storage	Limited only by available machine storage	7 days
Message replay	Yes	Yes
Locality	Local cluster can replicate using MirrorMaker	Global distributed service with configurable message storage locations
Logging and monitoring	Self-managed	Automated with Cloud Logging and Cloud Monitoring

https://cloud.google.com/architecture/migrating-from-kafka-to-pubsub#comparing_features

Comparing Kafka and Pub/Sub: **Features**

	Kafka	Pub/Sub
Message ordering	Yes within partitions	Yes within topics
Message deduplication	Yes	Yes using Dataflow
Push subscriptions	No	Yes
Unprocessed message queue	As of version 2.0	Yes
Transactions	Yes	No
Message storage	Limited only by available machine storage	7 days
Message replay	Yes	Yes
Locality	Local cluster can replicate using MirrorMaker	Global distributed service with configurable message storage locations
Logging and monitoring	Self-managed	Automated with Cloud Logging and Cloud Monitoring

Not Pub/Sub defaults. Does anyone care?

https://cloud.google.com/architecture/migrating-from-kafka-to-pubsub#comparing_features

Comparing Kafka and Pub/Sub: **Features**

	Kafka	Pub/Sub
Message ordering	Yes within partitions	Yes within topics
Push subscriptions	No	Yes **Push to HTTP endpoints. Trigger event-based processing.** Google Cloud Run and other services.
Transactions	Yes	No
Message storage	Limited only by available machine storage	7 days
Message replay	Yes	Yes
Locality	Local cluster can replicate using MirrorMaker	Global distributed service with configurable message storage locations
Logging and monitoring	Self-managed	Automated with Cloud Logging and Cloud Monitoring

https://cloud.google.com/architecture/migrating-from-kafka-to-pubsub#comparing_features

Comparing Kafka and Pub/Sub: **Message Anatomy**

Kafka

Message data (bytes, always?)

Message metadata

- timestamp, topic, partition, offset, key
- more?
- custom?

reference?

Pub/Sub

Message data (bytes)

Message metadata

- publish time, message id
- custom attributes (can be used for filtering)

https://cloud.google.com/pubsub/docs/reference/rpc/google.pubsub.v1#google.pubsub.v1.PubsubMessage

Comparing Kafka and Pub/Sub: **Workflow**

Kafka

Consumer **polls** the broker.

Broker sends a batch of messages from a topic partition, in order, tracking message offsets.

Consumers periodically commit offsets
(not for every message)

Consumers can rewind/fast forward to specific offset.

Pub/Sub

Subscriber **pulls** messages,
-or- subscription **pushes** messages to endpoint.

Pub/Sub sends messages from a subscription, tracking subscriber acknowledgements.

Subscriber processes the message, then sends an acknowledgement back to Pub/Sub.
(messages are redelivered if no acknowledgement is received)

Subscribers can seek backward/forward in time.
(but acknowledged messages not retained by default; costs extra)

Comparing Kafka and Pub/Sub: **Extendability/Connectors**

Kafka

Many **plugins** that connect Kafka to other services. confluent.io/hub.

- Apache Cassandra,
- MongoDB,
- AWS Lambda & S3,
- Google Pub/Sub & Cloud Functions & Cloud Storage

KafkaUtils API connects to Spark Streaming

Others?

Pub/Sub

Push to any HTTP endpoint for event-driven processing/storage.

- Google Cloud Functions
- Dataflow
- Google Cloud Run
- AWS Lambda?

Pull from anywhere with network access (API clients, REST, RPC, CLI).

Comparing Kafka and Pub/Sub: **End-user APIs**

Kafka

Custom APIs:

- `antares_client.StreamingClient`
- `fink-client`
- `hop-client`
- TOM Toolkit (ANTARES, Fink, Lasair, ALeRCE, SCIMMA, TNS, etc.)
- others?

Python client:

`pip install --no-binary :all: confluent-kafka`

(plus librdkafka and dependencies, for SASL Kerberos/GSSAPI support)

Pub/Sub

APIs:

- REST, RPC
- Client libraries (Python, Java, C++, etc.)
  - Pitt-Google Python wrappers
  - TOM Toolkit (Pitt-Google)
- CLI
- Console

Python client:

`pip install google-cloud-pubsub`

Comparing Kafka and Pub/Sub: **Cost**

Kafka

?

(machines, network, maintenance)

Pub/Sub

Throughput:

$40 per TiB of data transmitted

(first 10 GiB/month is free)

Egress:

$0.045 - $0.23 per GiB delivered

(applies to messages crossing a Google Cloud region)

Who pays?

Producers pay to publish
Subscribers pay to consume

cloud.google.com/pubsub/pricing

Outline

1. Problem setup:

   Requirements for
   Ingesting and redistributing live alert streams at LSST scale

2. Comparison of solutions:

   Apache Kafka and Google Cloud Pub/Sub

3. Solutions talking to each other:

   Kafka ⟷ Pub/Sub connector

Kafka ⟷ Pub/Sub Connector

Kafka Connect plugin to Confluent Platform
https://github.com/GoogleCloudPlatform/pubsub/tree/master/kafka-connector

- **Hands off implementation:**
  - call a bin file, pass in configs
  - it manages the connection, polls the broker, and publishes the messages (Kafka -> Pub/Sub)
- Standalone and distributed modes
- Data conversion tools available
- Built and supported by Pub/Sub developers (not Confluent)

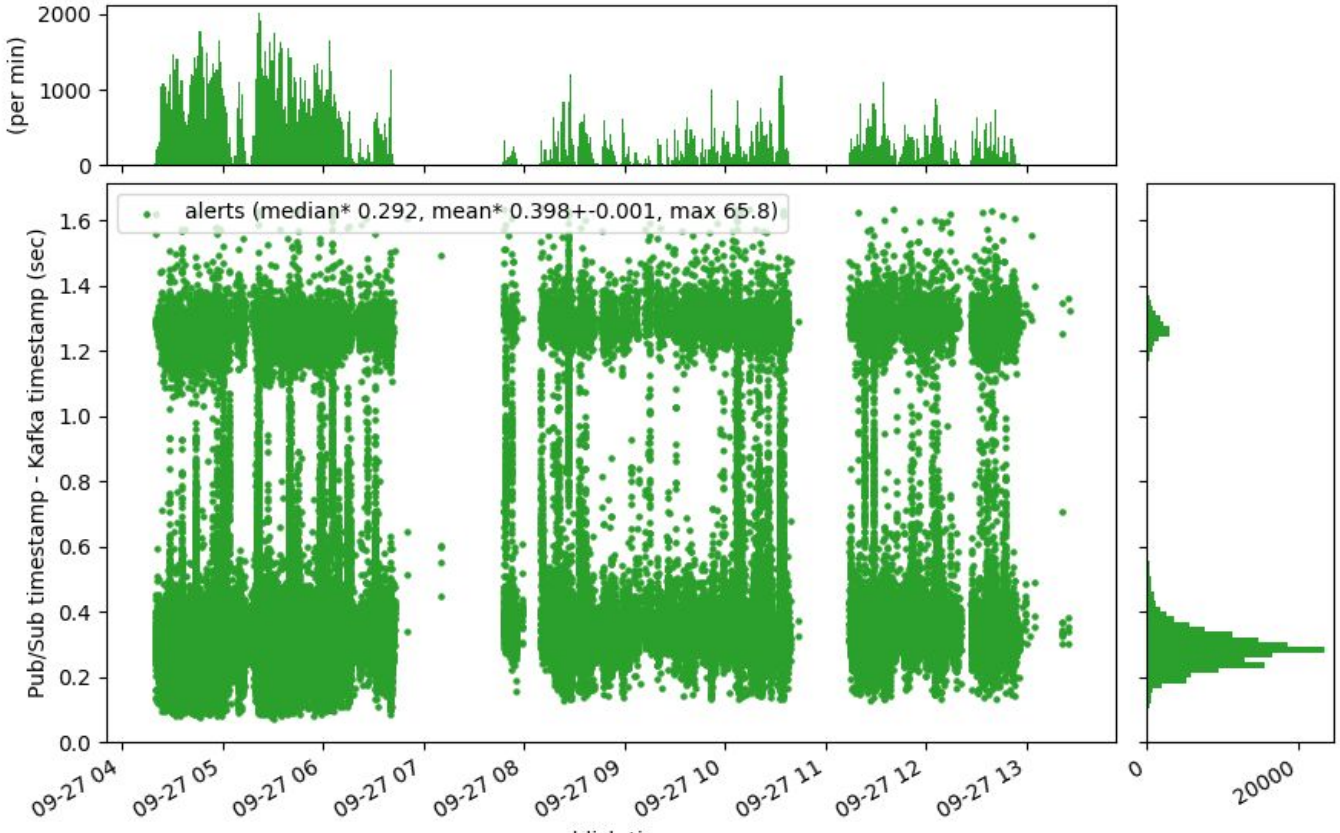
Kafka  Pub/Sub Connector

https://github.com/GoogleCloudPlatform/pubsub/tree/master/kafka-connector

Pitt-Google uses this to ingest ZTF's streams (Kafka -> Pub/Sub)

- We pass message bytes straight through (no decoding), plus metadata
- Single "g1-small" VM (standalone mode)
  - 0.5 vCPU
  - 1.70 GB memory
  - ~$8/month
- <0.5 sec mean delay between Kafka and Pub/Sub timestamps
  - Preliminary tests indicate it will handle LSST loads similarly

Kafka ⟷ Pub/Sub Connector



Kafka  Pub/Sub Connector

https://github.com/GoogleCloudPlatform/pubsub/tree/master/kafka-connector

- Should try in reverse, Pub/Sub -> Kafka (hack day?)
- (Avro formats to facilitate exchange)

Summary

Kafka

Type: Platform

Benefits: More familiarity among astronomers.
Many astronomy tools integrated.

Drawbacks: Managing software, servers, data.
Installation and configuration.

Extendability: Connectors to many applications.

Pub/Sub

Type: Service

Benefits: Ease of use. Lightweight.
Low barrier to entry.

Drawbacks: Cost?
Less familiarity within astronomy.

Extendability: Push subscriptions.
Simple API access.

Talking to each other

Converting Kafka -> Pub/Sub doable at-scale with minimal resources (Pitt-Google demonstrating).
Let's try converting Pub/Sub -> Kafka