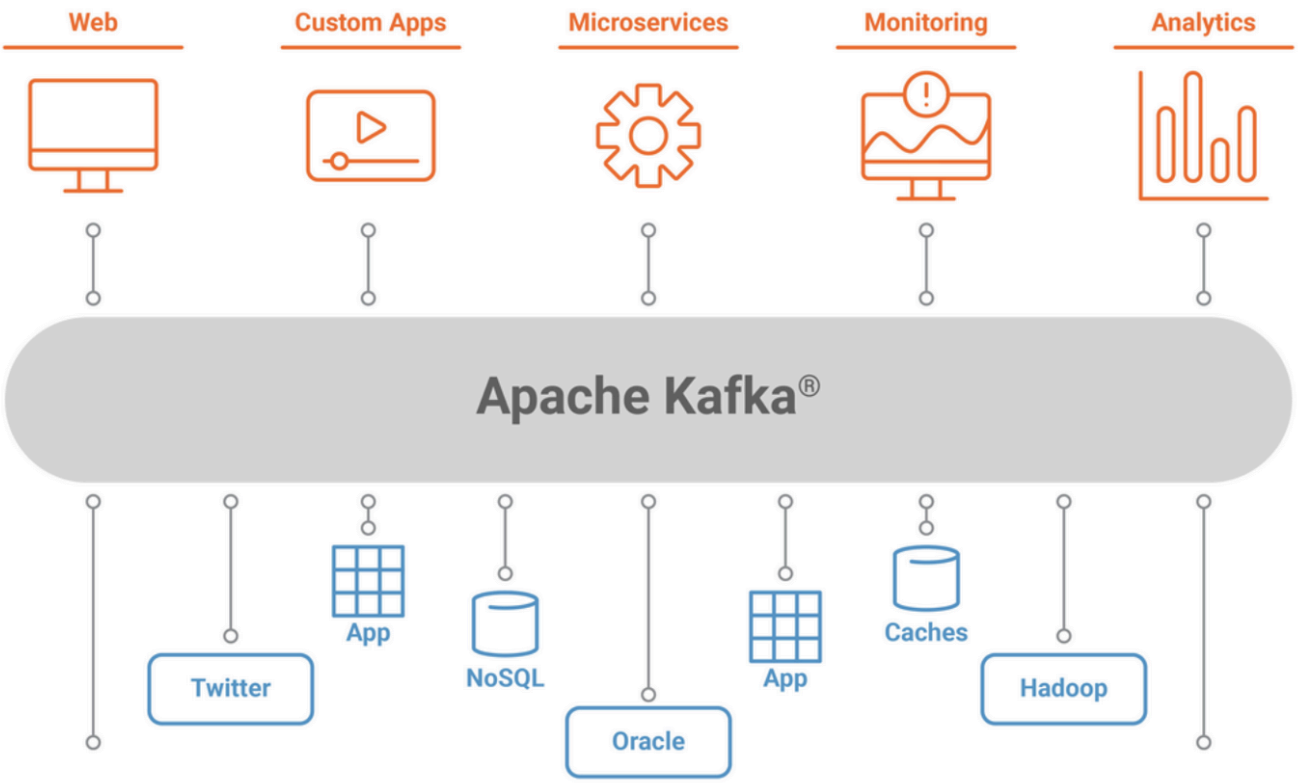
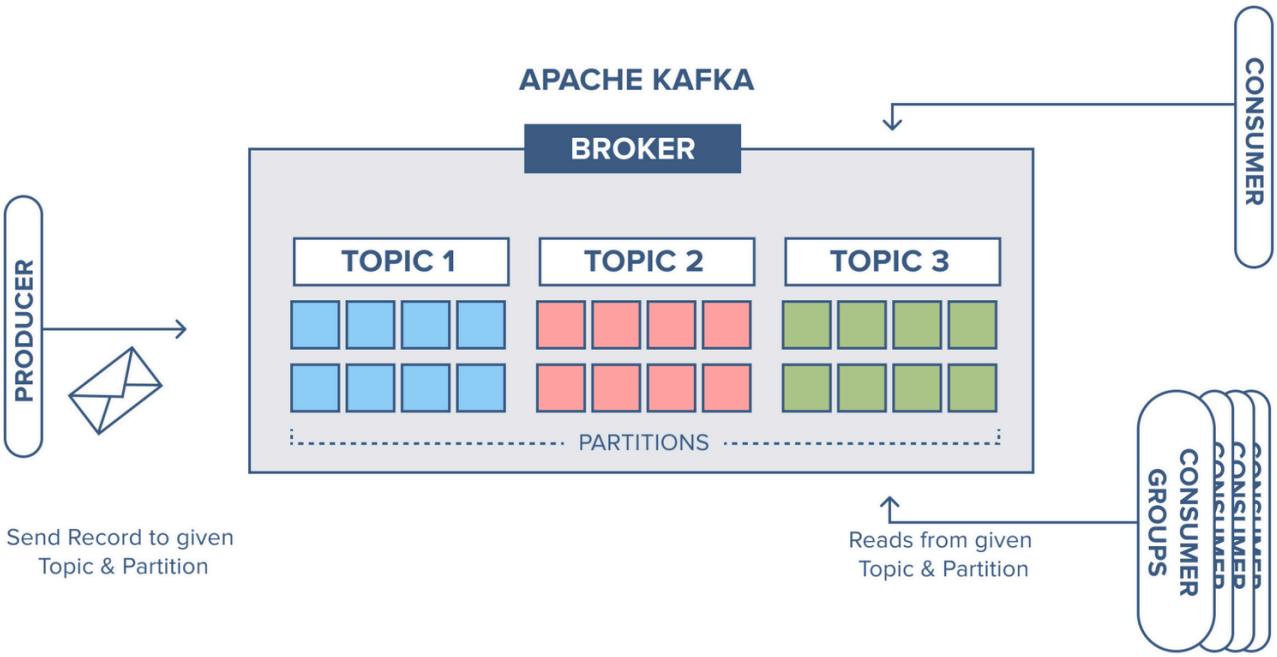
01 PubSub -KirkYagami

KAFKA

- Apache Kafka is a popular event streaming platform used to collect, process, and store streaming event data or data that has no discrete beginning or end.
- Kafka makes possible a new generation of distributed applications capable of scaling to handle billions of streamed events per minute.
- Kafka takes streaming data and records exactly what happened and when.
- This record is called an immutable commit log.
- It is immutable because it can be appended to, but not otherwise changed.
- From there, you can subscribe to the log (access the data) and you can also publish to it (add more data) from any number of streaming real-time applications, as well as other systems.





- Kafka Core Functions:
  1. Publish
  2. Consume
  3. Process
  4. Connect
  5. Store

1. **Publish**
   A data source can publish or place a stream of data events into one or more Kafka topics, or groupings of similar data events. For example, you can take data streaming from an IoT device and publish it to an application that does predictive forecasting.

2. **Consume**

- An application can subscribe to, or take data from, one or more Kafka topics and process the resulting stream of data.
- For example, an application can take data from multiple social media streams and analyze it to determine the tenor of online conversations about a brand.

3. **Process**

- Kafka Streams API can act as a stream processor, consuming incoming data streams from one or more topics and producing an outgoing data stream to one or more topics.

4. **Connect**

- You can also build reusable producer or consumer connections that link Kafka topics to existing applications.
- There are hundreds of existing connectors already available, including connectors to key services like Dataproc, BigQuery, and more.

4. **Store**

- Apache Kafka provides durable storage. Kafka can act as a "source of truth," being able to distribute data across multiple nodes for a highly available deployment within a single data center or across multiple availability zones.

> GCP's Pub/Sub service also builds on top of Apache Pulsar, which is also a publisher-subscriber model that can dynamically scale up or down depending on load demand.

Apache Pulsar Features:

- Low Latency
- Multi-tenancy
- Persistent Storage
- Horizontal Scalability

PubSub Overview

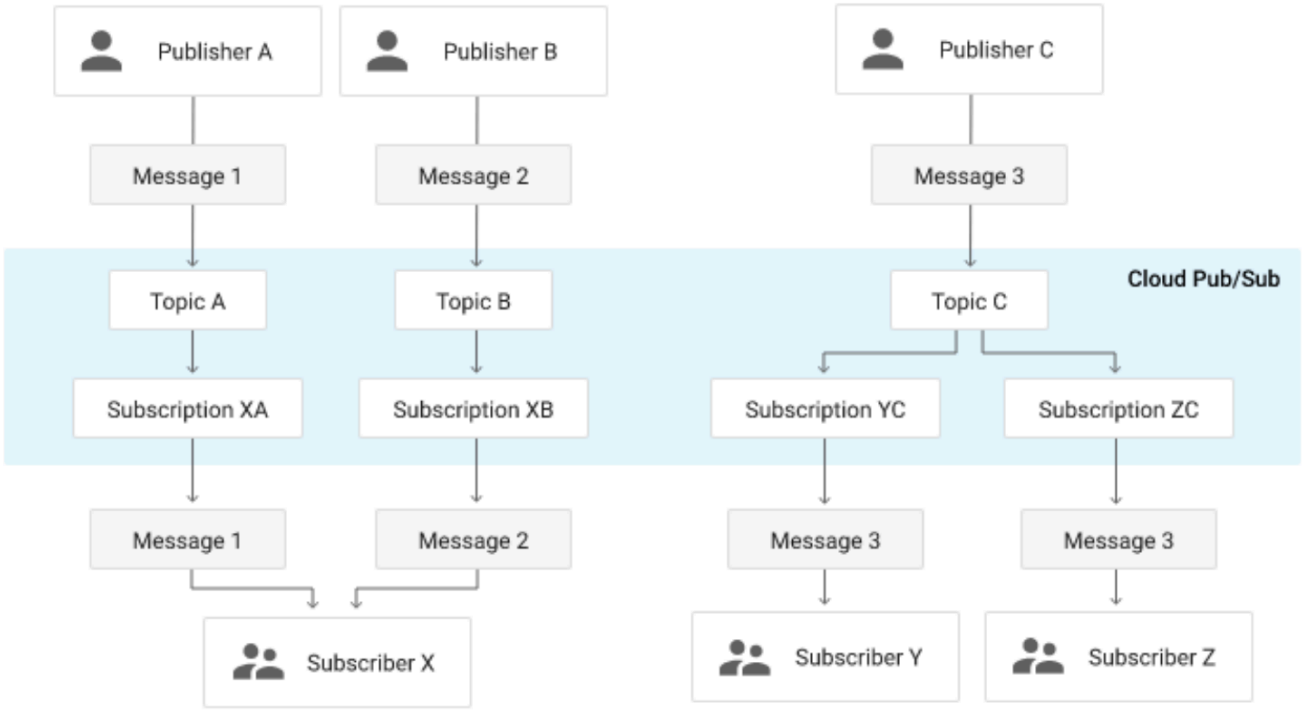
Pub/Sub allows services to communicate asynchronously, with latencies on the order of 100 milliseconds.

- Pub/Sub is used for streaming analytics and data integration pipelines to ingest and distribute data.
- It is equally effective as a messaging- oriented middleware for service integration or as a queue to parallelize tasks.
- Pub/Sub enables you to create systems of event producers and consumers, called publishers and subscribers.
- Publishers communicate with subscribers asynchronously by broadcasting events.
- Publishers send events to the Pub/Sub service, without regard to how or when these events are to be processed.
- Pub/Sub then delivers events to all services that need to react to them.

Pub/Sub Use Cases:

1. **Ingestion user interaction and server events.**
   - To make use of user interaction events from apps, you may forward them to Pub/Sub and then use a stream processing tool (such as Dataflow) which delivers them to BigQuery.
2. **Real-time event distribution**
   - Events, raw or processed, may be made available to multiple applications across your team and organization for real time processing.
3. **Replicating data among databases**
   - Pub/Sub is commonly used to distribute change events from databases.

4. **Enterprise event bus**
   - You can create an enterprise-wide real-time data sharing bus, distributing business events, database updates, and analytics events across your organization.
5. **Refreshing distributed caches**
   - For example, an application can publish invalidation events to update the IDs of objects that have changed.

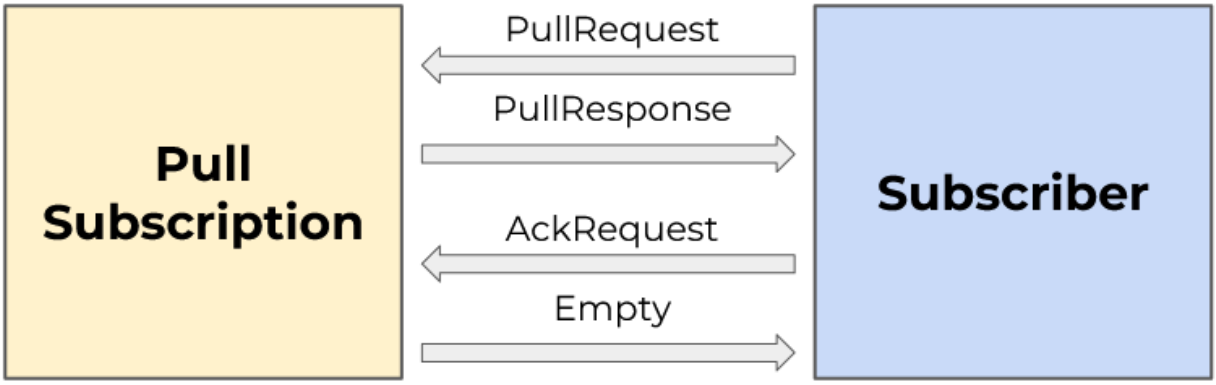


PubSub Architecture

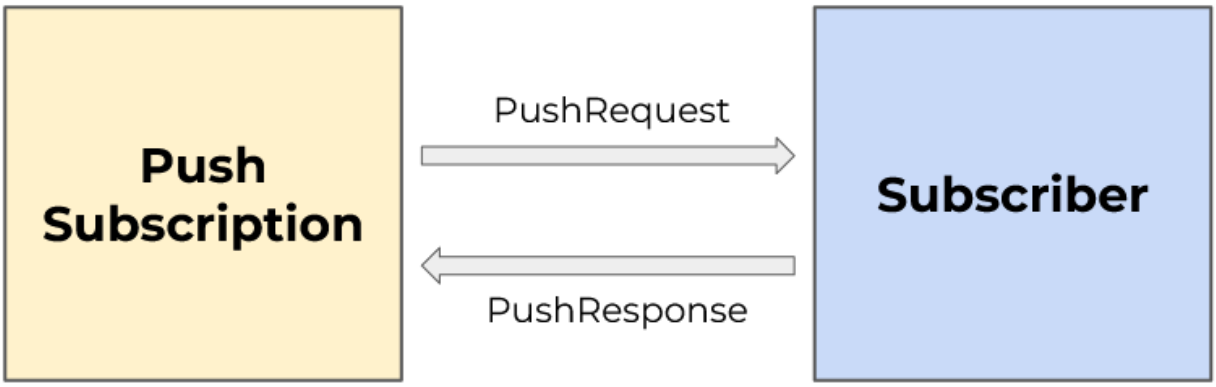
Subscriptions in PubSub

- **Pull Subscription**
  - For a pull subscription, your subscriber client initiates requests to a Pub/Sub server to retrieve messages using the REST Pull API, RPC PullRequest API, the REST StreamingPullRequest API, or the RPC StreamingPullRequest API.
  - Most subscriber clients do not make these requests directly and instead rely on the Google Cloud-provided high-level client library that performs streaming pull requests internally and delivers messages asynchronously.

- workflow:



- **Push Subscription**
  - In a push subscription, a Pub/Sub server initiates a request to your subscriber client to deliver messages.
  - Workflow:



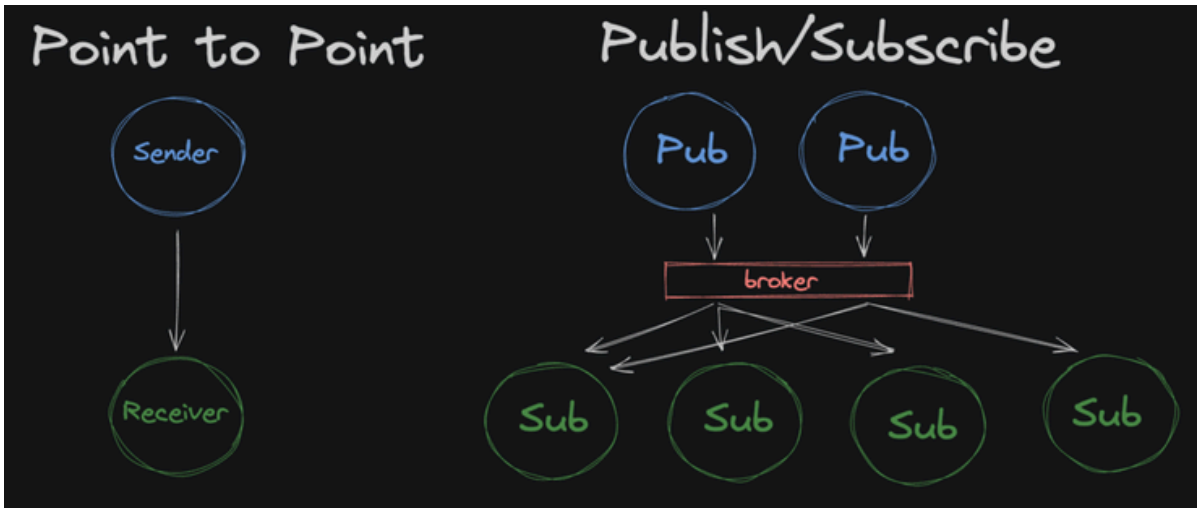
	Pull	Push
Use case	Large volume of messages (many more than 1 per second).	Multiple topics that must be processed by the same webhook.
	Efficiency and throughput of message processing is critical.	App Engine Standard and Cloud Functions subscribers.
	Environments where a public HTTPS endpoint with a non-self-signed SSL certificate is not feasible to set up.	Environments where Google Cloud dependencies (such as credentials and the client library) are not feasible to set up.

- By default, Pub/Sub offers at-least-once delivery with no ordering guarantees on all subscription types.
- Alternatively, if messages have the same ordering key and are in the same region, you can enable message ordering.
- After the message ordering property is set, the Pub/Sub service delivers messages with the same ordering key in the order that the Pub/Sub service receives the messages.

Pub/Sub (publish/subscribe) is a pattern software systems can use to communicate. It enables architectures that can be more:

- Scalable
- Reliable
- Maintainable

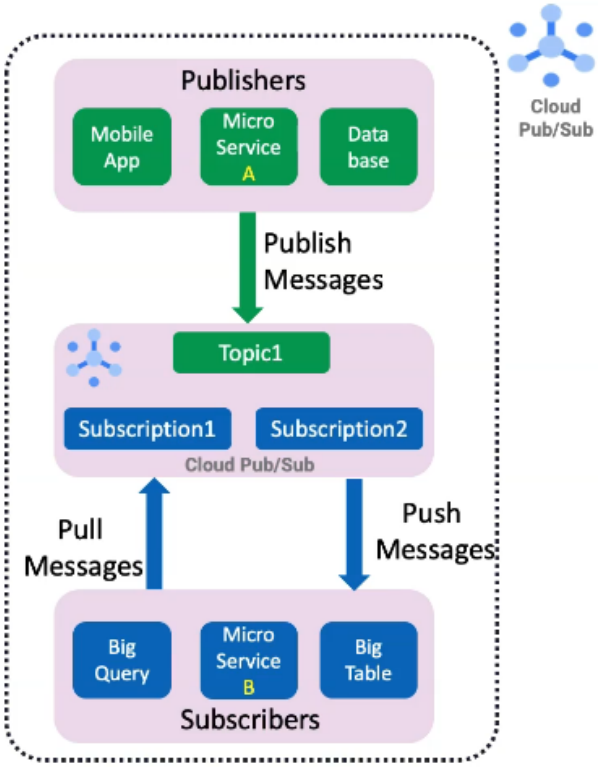
Pub/Sub systems are the opposite of point-to-point systems. In point-to-point systems, the sender is painfully aware of the receiver. For example, an HTTP request is point-to-point. The client sends a request *directly* to the server, and the server sends a response *directly* back to the client.
Email - point-to-point
YouTube: pub/sub system



Twitter (X) works like a Pub/Sub system. When you tweet, you don't individually list all of the followers you want to send your tweet to. Instead, you just tweet, and Twitter's servers deliver your tweet to all of your followers.

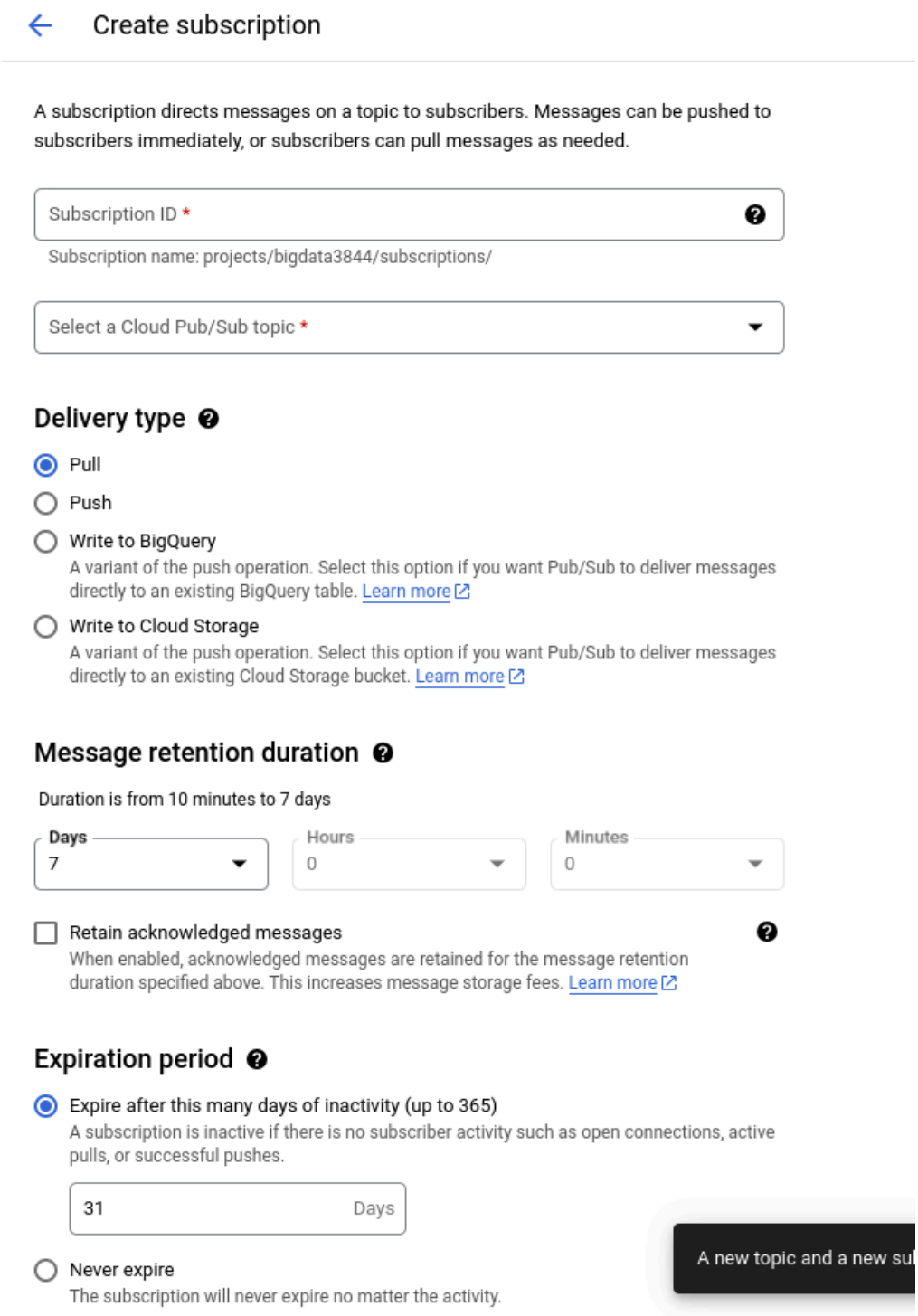
Google Cloud Pub/Sub

- Cloud Pub/Sub: Pub/Sub is a fully-managed asynchronous messaging service designed to be highly reliable and scalable.
- Google products, such as Ads, Search, and Gmail, send 500 million messages per second using this infrastructure, totaling over 1TB/s of data.
- Primarily used for real-time data streaming and event-driven systems.
- Stream Analytics (real-time data streaming)
  Very powerful feature
  Ingest analytic events of our applications and stream them to BigQuery, with Dataflow.



- **Publishers**
  - Services that produce messages
  - Publishers send events to Pub/Sub Topics, without worrying about when or how these events will be handled.

- **Subscribers**
  - Services that process those messages
  - Subscribers subscribe to a Pub/Sub subscription
  - Subscriptions have the following delivery types:
    - **Pull**: Subscriber needs to pull the messages
    - **Push**,
    - **Write to BigQuery**,
    - **Write to Cloud Storage**
  - As soon as a message arrives at a Pub/Sub topic, the subscription will push them to registered subscribers.



- **Global Service**
  - Pub/Sub is a global service
  - Topics and subscriptions are not region-specific
  - Messages flow within the Pub/Sub service between regions when needed
  - When using the global endpoint (`pubsub.googleapis.com`), publishers and subscribers connect to the nearest network region where Pub/Sub runs.
  - When using the locational endpoints (`us-central1-pubsub.googleapis.com`), publishers and subscribers connect to Pub/Sub in the specified region.
- **Autoscaling**

- Designed to scale horizontally
- No provisioning, not visible, everything happens in the background
- Auto-everything

Life of a message

- A publisher sends a message to Pub/Sub.
- The message is written to Pub/Sub storage.
- Pub/Sub sends an acknowledgement to the publisher that it has received the message and guarantees its delivery to all attached subscriptions.
- At the same time as writing the message to storage, Pub/Sub delivers it to subscribers.
- Subscribers send an acknowledgement to Pub/Sub that they have processed the message.
- Once at least one subscriber for each subscription has acknowledged the message, Pub/Sub deletes the message from storage.
- **Compliance and Security**
  - HIPAA-compliant service
  - End-to-end encryption
  - Fine-grained access control
- **Google Cloud-native Integrations**
  - Cloud Functions for serverless event-driven computing
  - Dataflow (super powerful service in entire Google Cloud) for Stream Analytics
  - Cloud Logging
- **Message Filtering**
  - Subscribers will only receive messages that match the filter
  - Helps reduce delivery volume to subscribers

Google Cloud Pub/Sub

- **Dead Letter Topics**
  - To enable, need to create a Dead Letter Topic
  - Messages unable to be processed are published to Dead Letter Topic for later review and troubleshooting
  - This ensures that other messages aren't held up while issues are addressed.
- **Exactly Once Delivery**
  - Messages sent to the subscription are guaranteed not to be resent before the message's acknowledgement deadline expires

- Acknowledged messages will not be resent to the subscription