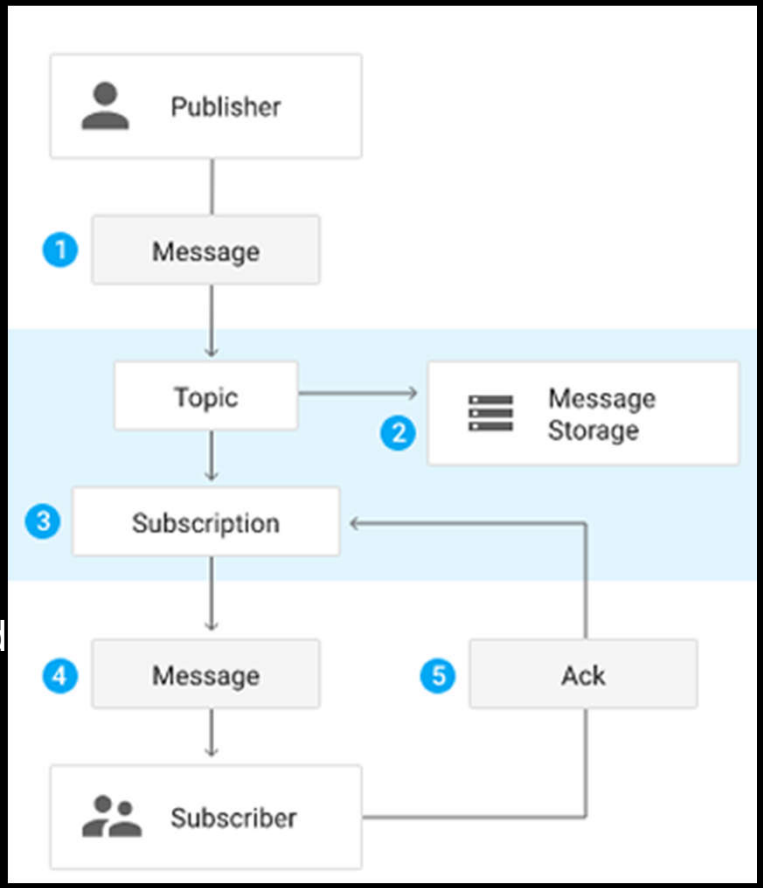

GOOGLE CLOUD PLATFORM
CLOUD PUB/SUB

WHAT IS CLOUD PUB/SUB?

- Cloud Pub/Sub is a scalable, durable, fully managed messaging system that's real-time and guarantees delivery.
- Brings the flexibility and reliability of enterprise message-oriented middleware to the cloud
- Provides many-to-many, asynchronous messaging that decouples senders and receivers – facilitates secure and highly available communication among independently written applications.
- Cloud Pub/Sub delivers low-latency, durable messaging that helps developers quickly integrate systems hosted on the Google Cloud Platform and externally
- Google products including Ads, Search and Gmail use this infrastructure to send over 500 million messages per second, totaling over 1TB/s of data.

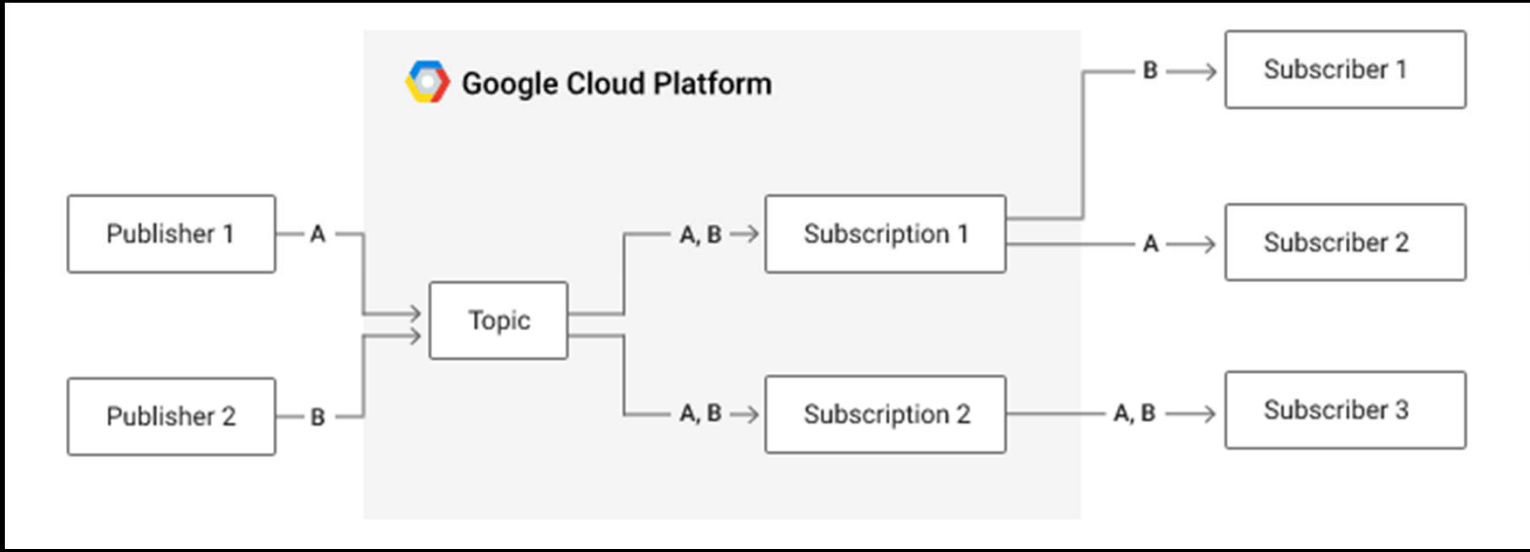
BASICS OF A PUBLISH/SUBSCRIBE SERVICE

- Cloud Pub/Sub is a *publish/subscribe* (*Pub/Sub*) *service*: a messaging service where the senders of messages are decoupled from the receivers of messages.
- Key concepts in a Pub/Sub service:
  - *Message*: the data that moves through the service.
  - *Topic*: a named entity that represents a feed of messages.
  - *Subscription*: a named entity that represents an interest in receiving messages on a particular topic.
  - *Publisher* (also called a producer): creates messages and send (publishes) them to the messaging service on a specified topic.
  - *Subscriber* (also called a consumer): receives messages on a specified subscription.



BASIC FLOW OF MESSAGES

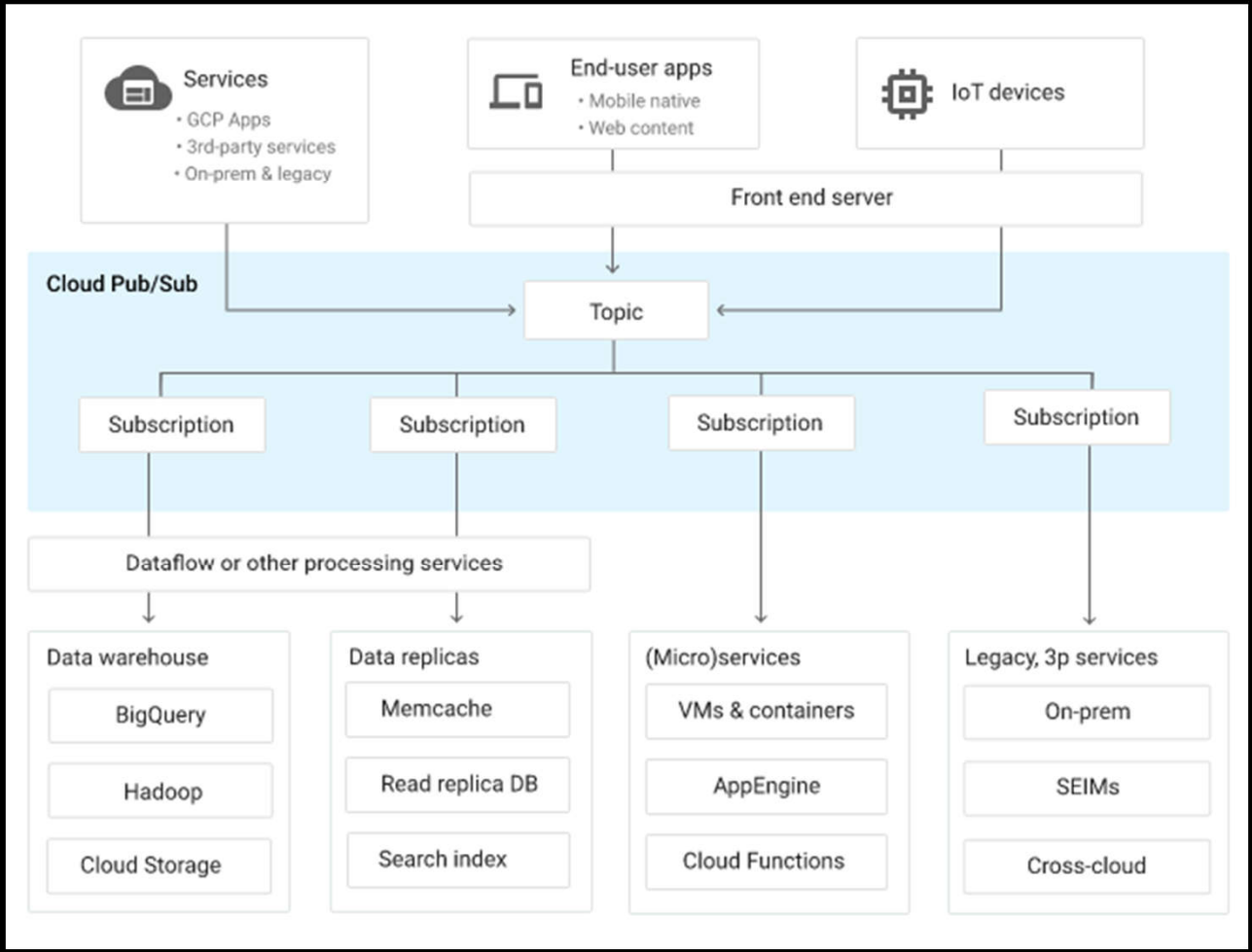
In this scenario, there are two publishers publishing messages on a single topic.



COMMON USE CASES

- **Balancing workloads in network clusters.** For example, a large queue of tasks can be efficiently distributed among multiple workers, such as Google Compute Engine instances.
- **Implementing asynchronous workflows.** For example, an order processing application can place an order on a topic, from which it can be processed by one or more workers.
- **Distributing event notifications.** For example, a service that accepts user signups can send notifications whenever a new user registers, and downstream services can subscribe to receive notifications of the event.
- **Refreshing distributed caches.** For example, an application can publish invalidation events to update the IDs of objects that have changed.
- **Logging to multiple systems.** For example, a Google Compute Engine instance can write logs to the monitoring system, to a database for later querying, and so on.
- **Data streaming from various processes or devices.** For example, a residential sensor can stream data to backend servers hosted in the cloud.
- **Reliability improvement.** For example, a single-zone Compute Engine service can operate in additional zones by subscribing to a common topic, to recover from failures in a zone or region.

PUBLISHER AND SUBSCRIBER ENDPOINTS



- Publishers can be any application that can make HTTPS requests to googleapis.com:
  - App Engine app
  - A web service hosted on Google Compute Engine or any other third-party network
  - An installed app for desktop or mobile device
  - Browser apps

PERFORMANCE OF A MESSAGING SERVICE

- Performance in judged three aspects: scalability, availability, and latency.

- **Scalability:** A scalable service should be able to handle increases in load without noticeable degradation of latency or availability. "Load" can refer to various dimensions of usage in Cloud Pub/Sub:
  - Number of topics
  - Number of publishers
  - Number of subscriptions
  - Number of subscribers
  - Number of messages
  - Size of messages
  - Rate of messages (throughput) published or consumed
  - Size of backlog on any given subscription

PERFORMANCE OF A MESSAGING SERVICE

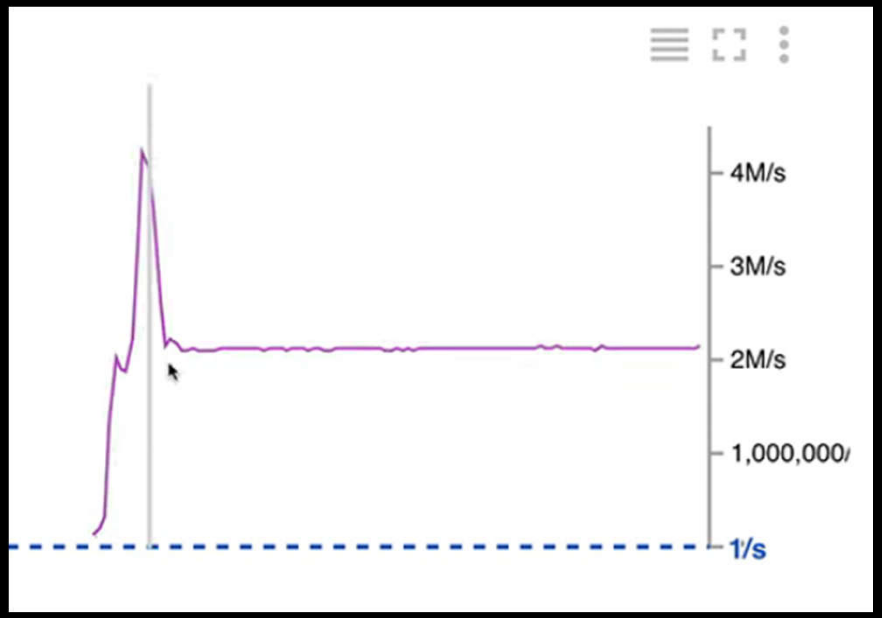
Availability:

- Failure due to load could happen when a sudden increase in traffic in the service (or in other software components running on the same hardware or in software dependencies) results in resource scarcity.
- Availability can also degrade due to human error, where one makes mistakes in building or deploying software or configurations.
- A system's availability is measured on how well it deals with different types of issues, gracefully failing over in a way that is unnoticeable to end users.

Latency: For Cloud Pub/Sub, the two most important latency metrics are:

- The amount of time it takes to acknowledge a published message.
- The amount of time it takes to deliver a published message to a subscriber.

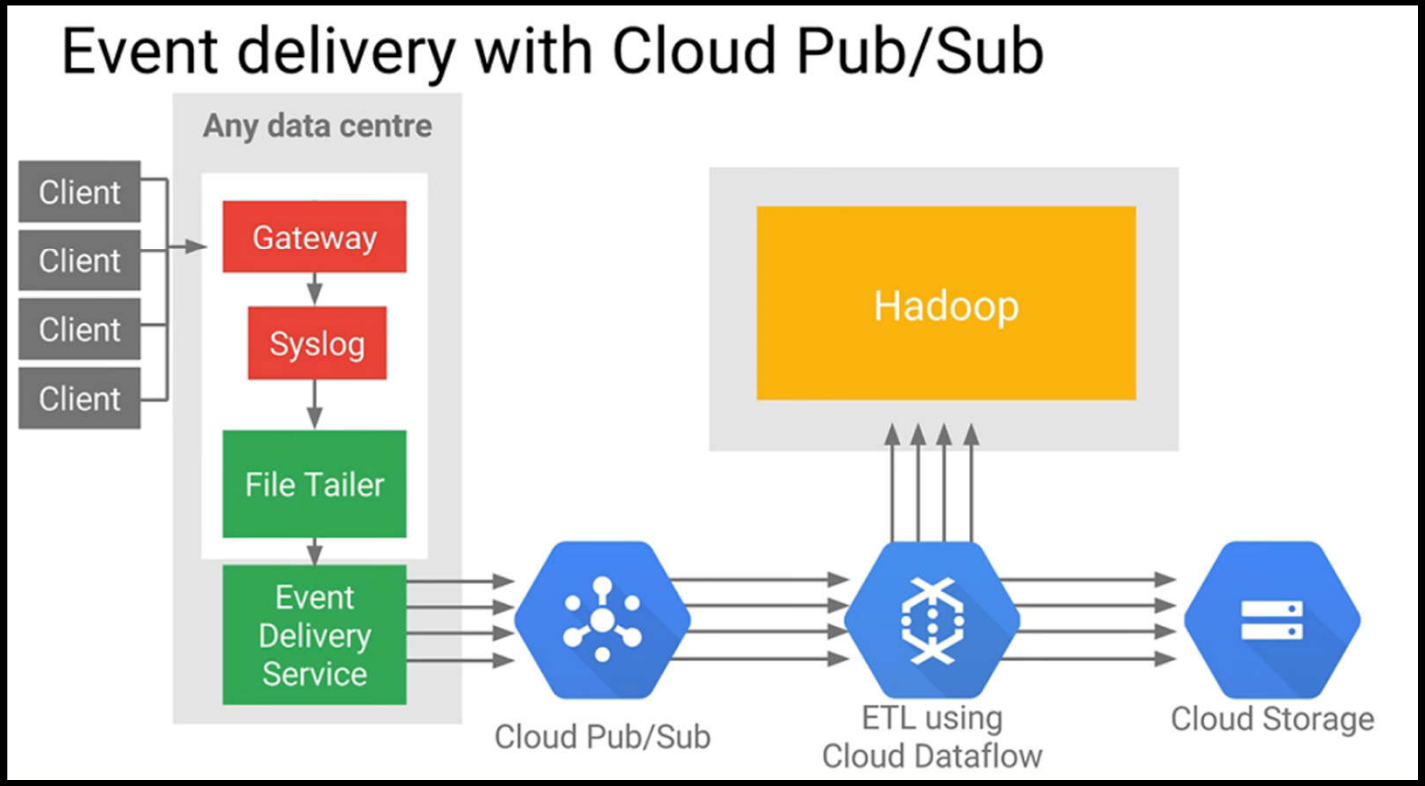
REAL WORLD EXAMPLE: SPOTIFY



Pub/Sub Subscription - Messages Pulled

- 75 Million monthly active users and tripling
- Data was increasing at a rate of 60 billion events per day
- 2 Million messages per second

REAL WORLD EXAMPLE: SPOTIFY



Drop-in replacement for kafca

ETL using Cloud Dataflow SDK

Make use of Google Cloud Storage

PUBLISHING MESSAGES

- A publisher application creates and sends messages to a *topic*.
- Cloud Pub/Sub offers at-least-once message delivery and best-effort ordering to existing subscribers
- The general flow for a publisher application is:
  - Create a message containing your data.
  - Send a request to the Cloud Pub/Sub Server to publish the message to the desired topic.
- **Custom Attributes:** Attributes can be text strings or byte strings
- **Balancing**: Messages can be batched based on request size (in bytes), number of messages, and time.
- **Retrying Requests:** Publishing failures are automatically retried, except for errors that do not warrant retries.
- **Concurrency Control:** **Threading and** support for concurrency is available but varies by language

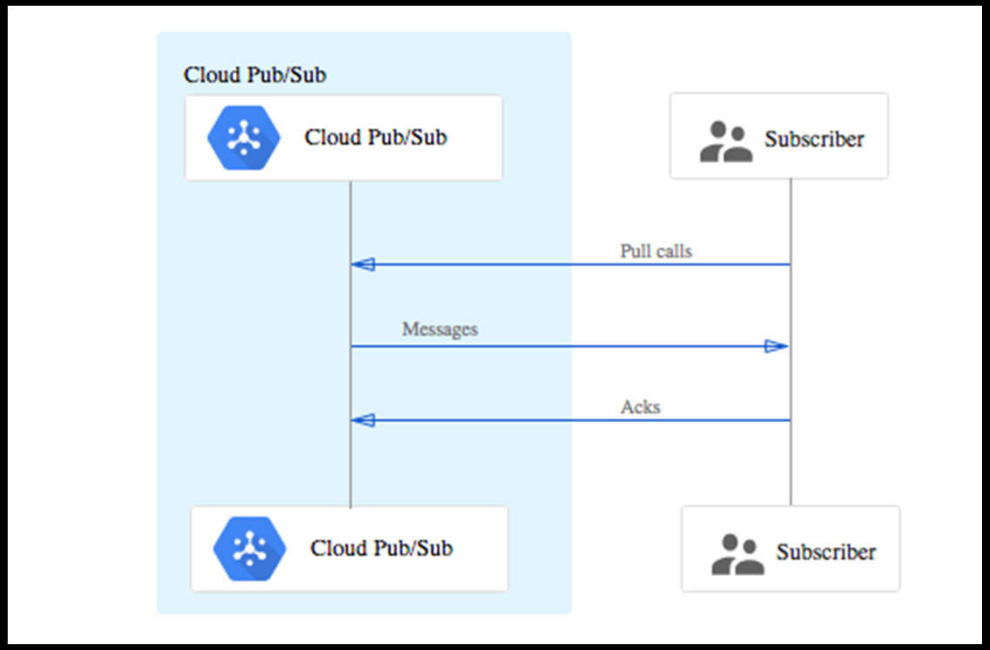
RECEIVING MESSAGES: SUBSCRIBER

- To receive messages published to a topic, you must create a subscription to that topic.
- Only messages published to the topic after the subscription is created are available to subscriber applications.
- The subscription connects the topic to a subscriber application that receives and processes messages published to the topic.
- A topic can have multiple subscriptions, but a given subscription belongs to a single topic.
- **At-Least-Once delivery:**
  - Cloud Pub/Sub delivers each published message at least once for every subscription.
  - By default, a message that cannot be delivered within the maximum retention time of 7 days is deleted and is no longer accessible.
  - A message published before a given subscription was created will usually not be delivered

SUBSCRIPTION: PULL OR PUSH DELIVERY

Pull subscription

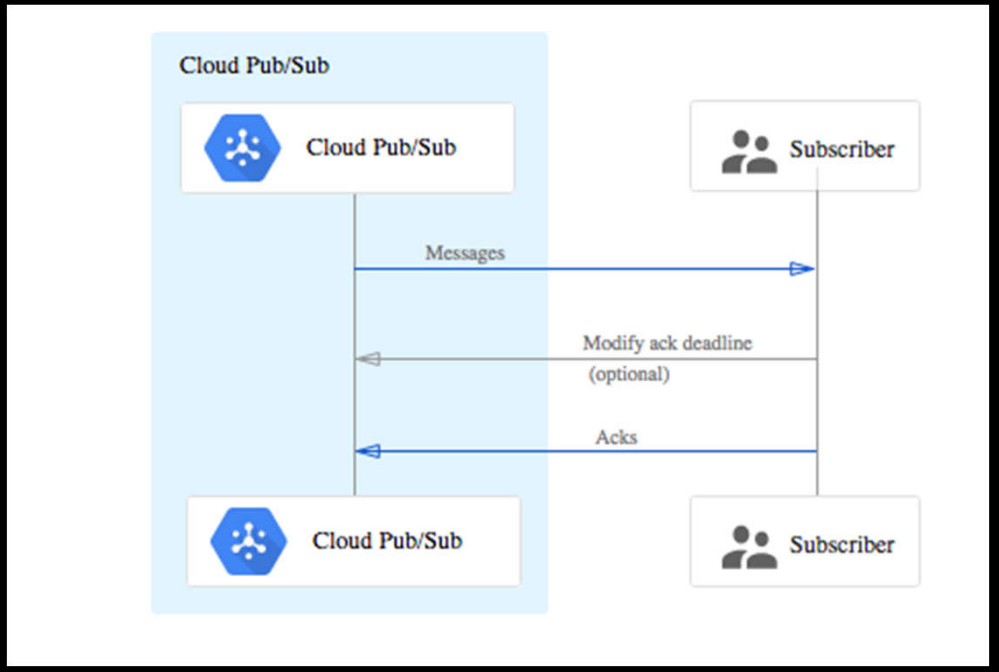
- In pull delivery, your subscriber application initiates requests to the Cloud Pub/Sub server to retrieve messages.
- The subscribing application explicitly calls the pull method, which requests messages for delivery.
- The Cloud Pub/Sub server responds with the message (or an error if the queue is empty) , and an ack ID.
- The subscriber explicitly calls the acknowledge method, using the returned ack ID to acknowledge receipt.



SUBSCRIPTION: PULL OR PUSH DELIVERY

Push subscription

- The Cloud Pub/Sub server sends each message as an HTTPS request to the subscriber application at a pre-configured endpoint.
- The endpoint acknowledges the message by returning an HTTP success status code. A non-success response indicates that the message should be resent.
- Cloud Pub/Sub dynamically adjusts the rate of push requests based on the rate at which it receives success responses.



CHOOSING PUSH VS. PULL

The following table offers some guidance in choosing the appropriate delivery mechanism for your application:

PUSH	PULL
Large volume of messages (many more than 1/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.
Public HTTPS endpoint, with non-self-signed SSL certificate, is not feasible to set up.	Environments where Google Cloud Platform dependencies (such as credentials and the client library) are not feasible to set up.

LIFECYCLE OF A SUBSCRIPTION

- By default, subscriptions expire after 31 days of inactivity
- If Cloud Pub/Sub detects subscriber activity, the subscription deletion clock restarts.
- Using subscription expiration policies (Beta) , you can configure the inactivity duration or make the subscription persistent regardless of activity.
- You can also delete a subscription manually. Although you can create new a subscription with the same name as a deleted one, the new subscription has no relationship to the old one.
- By default, Cloud Pub/Sub ensures that subscriptions retain unacknowledged messages for 7 days from the moment of publication.

MONITORING

- The Cloud Pub/Sub API exports metrics via Stackdriver.
- Stackdriver allows you to create monitoring dashboards and alerts or access the metrics programmatically.
- **Metrics and resource types**
  - To see the usage metrics that Cloud Pub/Sub reports to Stackdriver, you can view the Metrics List.
  - To see the details for the pubsub_topic and pubsub_subscription monitored resource types, you can view Monitored Resource Types in the Stackdriver.
- You can use the APIs and services quotas dashboard to monitor the current utilization for a given topic or subscription.

DEMO

1. Select/create a project
2. Select Cloud pub sub from the navigation menu
3. Enable the API
4. Create your first topic
 - Open Google Cloud Shell
 - $ gcloud pubsub topics create my-topic
5. Add a subscription
 - $ gcloud pubsub subscriptions create my-sub --topic my-topic --ack-deadline=60
6. List topics and subscriptions
 - $ gcloud pubsub topics list
 - $ gcloud pubsub subscriptions list
7. Publish messages to the topic
 Send two messages with the following commands:
 - $ gcloud pubsub topics publish my-topic --message hello
 - $ gcloud pubsub topics publish my-topic --message goodbye
8. Pull messages from the subscription
 - $ gcloud pubsub subscriptions pull --auto-ack --limit=2 my-sub
9. Manual acknowledgement
 - Send a new message $ gcloud pubsub topics publish my-topic --message thanks
 - Pull messages again $ gcloud pubsub subscriptions pull my-sub
 - Acknowledge the message $ gcloud pubsub subscriptions ack my-sub --ack-ids [ACK_ID]
10. View/manage topics & subscriptions from the GUI