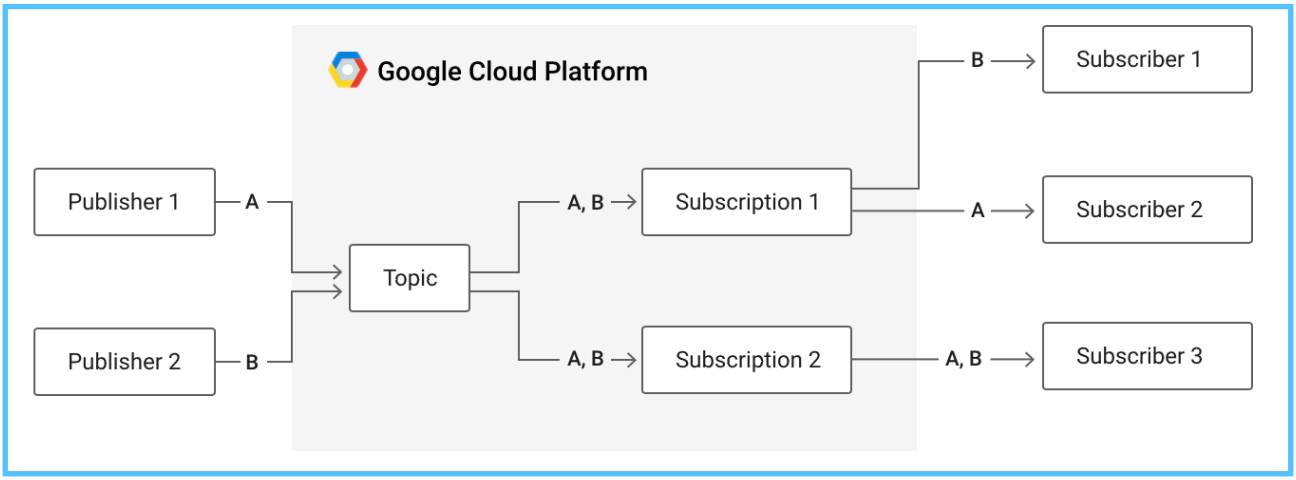
Google Cloud Pub/Sub

- Google Cloud Pub/Sub is a **messaging service** for exchanging event data among applications and services.
- Cloud Pub/Sub handles messaging and event ingestion at a **global level**, It is a **Serverless service** in Google Cloud.
- Cloud Pub/Sub is used for streaming analytics and data integration pipelines to ingest and distribute data.
- Cloud Pub/Sub is an asynchronous and scalable messaging service that decouples services producing messages from services processing those messages with latencies on the order of 100 milliseconds.
- Cloud Pub/Sub adopts from messaging middleware is **per-message parallelism**, rather than partition-based messaging.
- Cloud Pub/Sub works equally well for exchanging messages and data around a distributed system, as well as simple triggers and events, acting on certain actions and triggering other actions to occur. In this way, you can think of Pub/Sub as a kind of **shock absorber** for your systems.
- Cloud Pub/Sub guarantees **at-least-once delivery of every message**.It can scale to billions of message per day.
- Cloud Pub/Sub enables you to create systems of event producers and consumers, called publishers and subscribers. Publishers communicate with subscribers asynchronously by broadcasting events, rather than by synchronous remote procedure calls (RPCs).
- Using the **publisher/subscriber** model it can handle messages or events consistently in multiple regions as a serverless, NoOps fully-managed service.
- Publishers send events to the Cloud Pub/Sub service, we call these as Topics, without regard to how or when these events are to be processed.

- Service can publish a message to a topic or choose to receive a message from a topic. So now information from our users and other apps is published to a topic which can be consumed by other applications.
- Cloud Pub/Sub then delivers events to all the services that react to them. In systems communicating through RPCs, publishers must wait for subscribers to receive the data. However, the asynchronous integration in Pub/Sub increases the **flexibility** and **robustness** of the overall system.
- Pub/Sub has many integrations with other Google Cloud products to create a fully featured messaging system:
    - **Stream processing and data integration -** Supported by Dataflow, including Dataflow templates and SQL, which allow processing and data integration into BigQuery and data lakes on Cloud Storage.
    - **Monitoring, Alerting and Logging -** Supported by Monitoring and Logging products.
    - **Authentication and IAM -** Pub/Sub relies on a standard OAuth authentication used by other Google Cloud products and supports granular IAM, enabling access control for individual resources.
    - **APIs -** Pub/Sub uses standard gRPC and REST service API technologies along with client libraries for several languages.
    - **Triggers, notifications, and webhooks -** Pub/Sub offers push-based delivery of messages as HTTP POST requests to webhooks.
    - **Orchestration -** Pub/Sub can be integrated into multistep serverless Workflows declaratively. Big data and analytic orchestration often done with Cloud Composer, which supports Pub/Sub triggers.
    - **Integration Connectors -** These connectors let you connect to various data sources. With connectors, both Google Cloud services and third-party business applications are exposed to create a Pub/Sub connection for use in your integrations.

Cloud Pub/Sub Architecture

- Cloud Pub/Sub service is built on a core Google infrastructure component that many Google products have relied upon for over a decade.
- Google products including Google Ads, Google Search and Gmail use this infrastructure to send over 500 million messages per second, totalling over 1TB/s of data.



- In the above diagram, there are two publishers publishing messages on a single topic.
- There are two subscriptions to the topic.
  - The first subscription has two subscribers, meaning messages will be load-balanced across them, with each subscriber receiving a subset of the messages.
  - The second subscription has one subscriber that will receive all of the messages.
    - The bold letters represent messages.
      - Message A comes from Publisher 1 and is sent to Subscriber 2 via Subscription 1, and to Subscriber 3 via Subscription 2.
      - Message B comes from Publisher 2 and is sent to Subscriber 1 via Subscription 1 and to Subscriber 3 via Subscription 2.

- Pub/Sub is divided into two primary parts:
    - **the data plane** - which handles moving messages between publishers and subscribers. The servers in the data plane are called forwarders
    - **the control plane** - which handles the assignment of publishers and subscribers to servers on the data plane. The servers in the control plane are called routers.
- A messaging service like Pub/Sub can be judged on its performance in 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 Pub/Sub:
    - Number of topics
    - Number of publishers
    - Number of subscriptions
    - Number of subscribers
    - Number and Size of messages
    - Rate of messages (throughput) published or consumed
    - Size of backlog on any given subscription

Availability

- 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.
- Failures can occur in hardware (e.g., disk drives not working or network connectivity problems), in software, and due to load.

- 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.

Latency

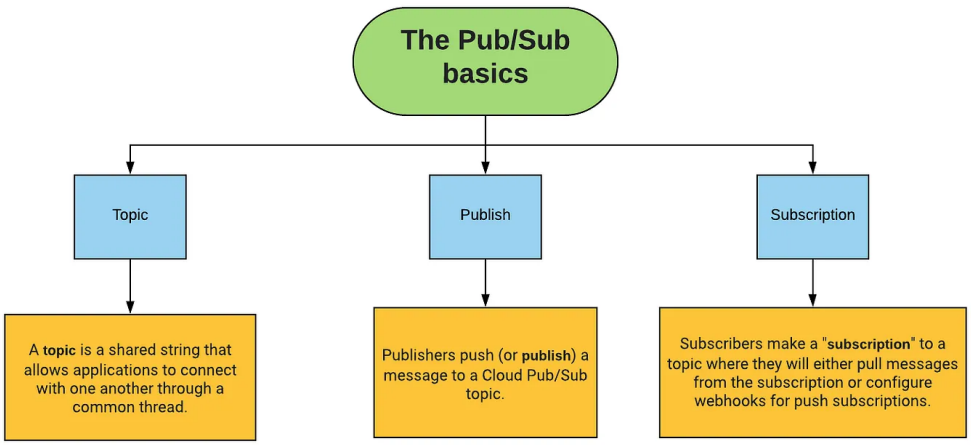
- Latency is a time-based measure of the performance of a system. A service generally wants to minimize latency wherever possible.
- For Pub/Sub, the two most important latency metrics are:
  - The amount of time it takes to acknowledge a published message.
  - The amount of time it takes to deliver a published message to a subscriber.

Basics of a Publish/Subscribe Service

- Pub/Sub is a publish/subscribe (Pub/Sub) service: a messaging service where the senders of messages are decoupled from the receivers of messages.

There are several key concepts in a Pub/Sub service:

- **Topic**
  - A named resource to which messages are sent by publishers.

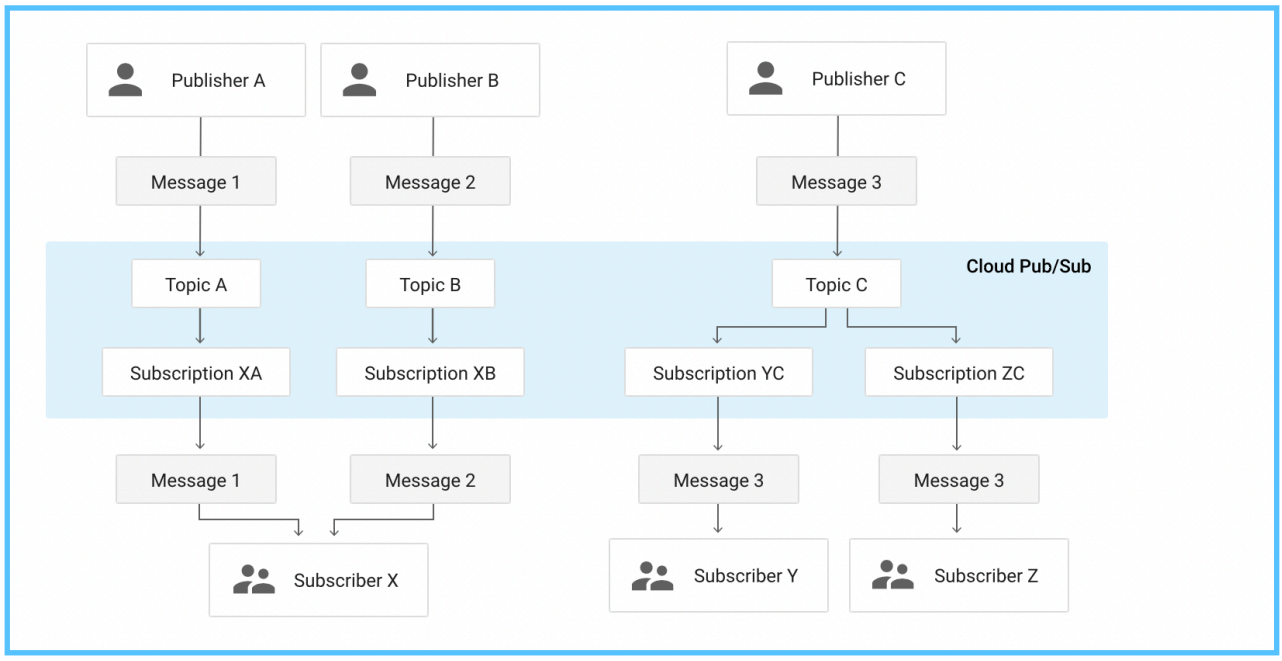


- **Subscription**

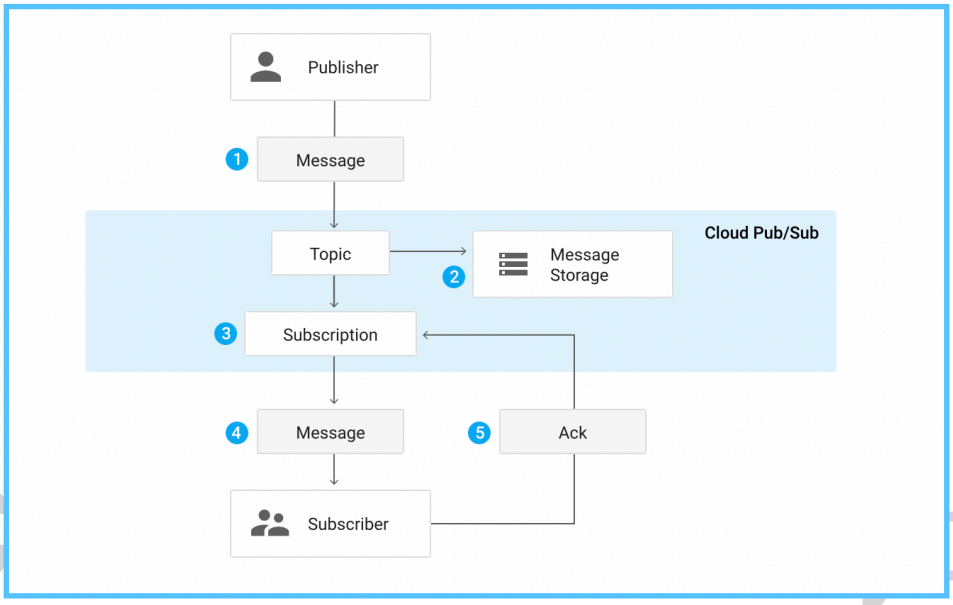
- A named resource representing the stream of messages from a single, specific topic, to be delivered to the subscribing application. For more details about subscriptions and message delivery semantics, see the Subscriber Guide.

- **Message**
  - The combination of data and (optional) attributes that a publisher sends to a topic and is eventually delivered to subscribers.
- **Message attribute**
  - A key-value pair that a publisher can define for a message. For example, key iana.org/language_tag and value en could be added to messages to mark them as readable by an English-speaking subscriber.
- **Publisher**
  - An application that creates and sends messages to a single or multiple topics.
- **Subscriber**
  - An application with a subscription to a single or multiple topics to receive messages from it.
- **Acknowledgment (or “ack")**
  - A signal sent by a subscriber to Pub/Sub after it has received a message successfully. Acknowledged messages are removed from the subscription message queue.
- **Push and pull**
  - The two message delivery methods. A subscriber receives messages either by Pub/Sub pushing them to the subscriber chosen endpoint, or by the subscriber pulling them from the service.
  - Publisher-subscriber relationships can be
    - one-to-many (fan-out)

- many-to-one (fan-in)
- many-to-many, as shown in the following diagram:



- One topic – Multiple Subscriber
- One subscriber – Multiple Topic
- The following diagram illustrates how a message passes from a publisher to a subscriber. For push delivery, the acknowledgment is implicit the response to the push request, while for pull delivery it requires a separate RPC. Publisher send message to topic at pubsub.googleapis.com



Features on Cloud PubSub

At-least-once delivery	• Synchronous, cross-zone message replication and per-message receipt tracking ensures at-least-once delivery at any scale.
Open	• Open APIs and client libraries in seven languages support cross-cloud and hybrid deployments.
Exactly-once processing	• Dataflow supports reliable, expressive, exactly-once processing of Pub/Sub streams.
No provisioning, auto-everything	• Pub/Sub does not have shards or partitions. Just set your quota, publish, and consume.
Compliance and security	• Pub/Sub is a HIPAA-compliant service, offering fine-grained access controls and end-to-end encryption.
Google Cloud–native integrations	• Take advantage of integrations with multiple services, such as Cloud Storage and Gmail update events and Cloud Functions for serverless event-driven computing.

Third-party and OSS integrations	• Pub/Sub provides third-party integrations with Splunk and Datadog for logs along with Striim and Informatica for data integration. • Additionally, OSS integrations are available through Confluent Cloud for Apache Kafka and Knative Eventing for Kubernetes-based serverless workloads.
Seek and replay	• Rewind your backlog to any point in time or a snapshot, giving the ability to reprocess the messages. Fast forward to discard outdated data.
Dead letter topics	• Dead letter topics allow for messages unable to be processed by subscriber applications to be put aside for offline examination and debugging so that other messages can be processed without delay.
Filtering	• Pub/Sub can filter messages based upon attributes in order to reduce delivery volumes to subscribers.

Types Of Pub/Sub Services

It consists of two services.

Pub/Sub service

- This messaging service is the default choice for most users and applications.
- It offers the highest reliability and largest set of integrations, along with automatic capacity management.
- Pub/Sub guarantees synchronous replication of all data to at least two zones and best-effort replication to a third additional zone.

Pub/Sub Lite service

- A separate but similar messaging service built for lower cost.
- It offers lower reliability compared to Pub/Sub. It offers either zonal or regional topic storage.
- Zonal Lite topics are stored in only one zone.Regional Lite topics replicate data to a second zone asynchronously.
- Also, Pub/Sub Lite requires you to pre-provision and manage storage and throughput capacity.
- Consider Pub/Sub Lite only for applications where achieving a low cost justifies some additional operational work and lower reliability.

Cloud PubSub Pricing

- Pub/Sub pricing is calculated based upon monthly data volumes. The first 10 GB of data per month is offered at no charge.
- The cost of Pub/Sub has three components:
  - **Throughput costs** for message publishing and delivery
  - **Egress costs** associated with throughput that crosses a Google Cloud zone or region boundary
  - **Storage costs** for snapshots, messages retained by topics, and acknowledged messages retained by subscriptions
- Pub/Sub service charges are based on usage (the number of published, delivered, or stored bytes).
- Pub/Sub Lite throughput and storage charges, by contrast, are based on reserved capacity.
- Egress charges for both services are based on usage, rather than reserved capacity.

Pros and Cons of Cloud PubSub

Pros

- Decoupled/loosely coupled components
- Greater system-wide visibility
- Real-time communication
- Ease of development. No complex programming is required.
- Increased scalability & reliability
- Testability improvements

Cons

- Testing can be a challenge - Unnecessary complexity in smaller systems
- Requires a well-defined policy for message formatting and message exchange; otherwise, message consumption can become mangled and error-prone.

Popular Use cases

- **Ingestion user interaction and server events**
  - To use user interaction events from end-user apps or server events from your system, you might forward them to Pub/Sub. You can then use a stream processing tool, such as Dataflow, which delivers the events to databases.
  - Examples of such databases are BigQuery, Cloud Bigtable, and Cloud Storage. Pub/Sub lets you gather events from many clients simultaneously.
- **Real-time event distribution**
  - Events, raw or processed, may be made available to multiple applications across your team and organization for real- time processing. Pub/Sub supports an "enterprise event bus" and event-driven application design patterns.
  - Pub/Sub lets you integrate with many Google systems that export events to Pub/Sub.
- **Replicating data among databases**
  - Pub/Sub is commonly used to distribute change events from databases.
  - These events can be used to construct a view of the database state and state history in BigQuery and other data storage systems.
- **Parallel processing and workflows**
  - You can efficiently distribute many tasks among multiple workers by using Pub/Sub messages to connect to Cloud Functions.
  - Examples of such tasks are compressing text files, sending email notifications, evaluating AI models, and reformatting images.

- **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.