

# Google Cloud Messaging Service

## 1. Google Cloud Messaging (GCM) – *Legacy Service*

Google Cloud Messaging was a service provided by Google that enabled developers to send messages and notifications to Android devices. However, **GCM has been deprecated** and replaced by **Firestore Cloud Messaging (FCM)**. It's important to know this because if you're building messaging or push notification systems today, you should be using FCM instead of GCM.

- **Purpose:** It was primarily used to deliver push notifications to apps on Android or Chrome, and messages from servers to clients efficiently.

## 2. Google Cloud Pub/Sub – *Core Messaging Service*

Google Cloud Pub/Sub is a **messaging and event ingestion service** for real-time analytics and event-driven architectures. It follows a **publish-subscribe model**, allowing services to **exchange messages asynchronously** and decouples senders (publishers) from receivers (subscribers).

### Use Cases:

- **Real-time analytics:** Collecting and delivering logs, telemetry, or event data for processing in systems like BigQuery or Dataflow.
- **Microservices communication:** Enabling services to exchange data without being tightly coupled.
- **IoT data ingestion:** Collecting and processing sensor data from large-scale IoT systems.
- **Database replication or change data capture:** Distributing changes from a database to multiple services.
- **Multicloud or hybrid-cloud architectures:** Bridging applications across environments.

### Benefits:

- **Scalability:** Can handle millions of messages per second and scales automatically.
- **Durability:** Ensures message persistence with high availability and fault tolerance.
- **Global delivery:** Messages can be routed to subscribers anywhere, across multiple regions.
- **Decoupling services:** Reduces system complexity by avoiding direct dependencies.
- **At-least-once delivery:** Ensures messages are delivered at least once unless acknowledged.

### Key Features:

- **Push and Pull subscriptions:** Subscribers can either pull messages or receive them via HTTP push.
- **Dead-letter topics:** Allows routing undelivered messages to another topic for inspection.
- **Message filtering:** Subscribers can receive only messages matching certain attributes.
- **Ordering guarantees:** Optional message ordering per key, useful when message sequence matters.
- **Integration with Google services:** Works seamlessly with Dataflow, Cloud Functions, Cloud Run, and others.

### 3. Pub/Sub Lite – *Low-Cost Alternative*

Pub/Sub Lite is a **lightweight, lower-cost version** of Pub/Sub designed for use cases that are **cost-sensitive** but still need scalable message ingestion.

#### Use Cases:

- **Batch processing pipelines:** Where high throughput is needed but latency is less critical.
- **Log aggregation for infrequent querying:** Collecting application logs or metrics that are analyzed periodically.
- **Offline data ingestion:** Systems that don't need real-time guarantees.
- **Startups or small businesses:** Who want event streaming capabilities without the full cost of standard Pub/Sub.

#### Benefits:

- **Lower cost:** Pricing is based on throughput and storage rather than requests or bandwidth, making it economical for predictable workloads.
- **High throughput:** Supports large message volumes with predictable patterns.
- **Durable storage:** Retains messages for a configurable period even if there are no subscribers.

#### Key Features:

- **Partition-based model:** Users must pre-provision partitions and throughput, unlike standard Pub/Sub which auto-scales.
- **Message retention:** Customizable retention settings for managing cost and performance.
- **Zonal availability:** Pub/Sub Lite is designed for use within a single zone or region (not global).
- **Client library-based delivery:** Unlike standard Pub/Sub, Lite clients must explicitly connect and consume data.

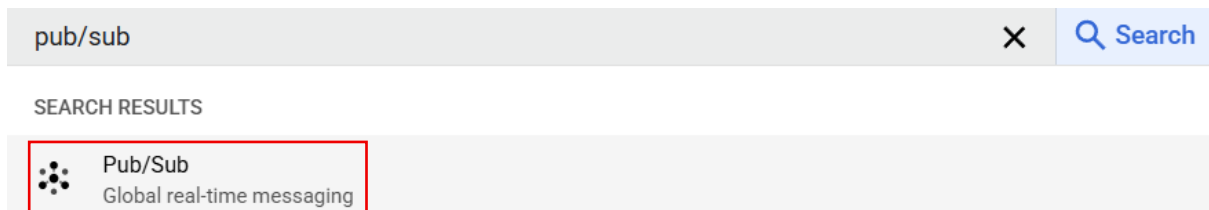
- **Integration with other GCP services:** Works with Dataflow, though with more manual setup than standard Pub/Sub.

## Summary

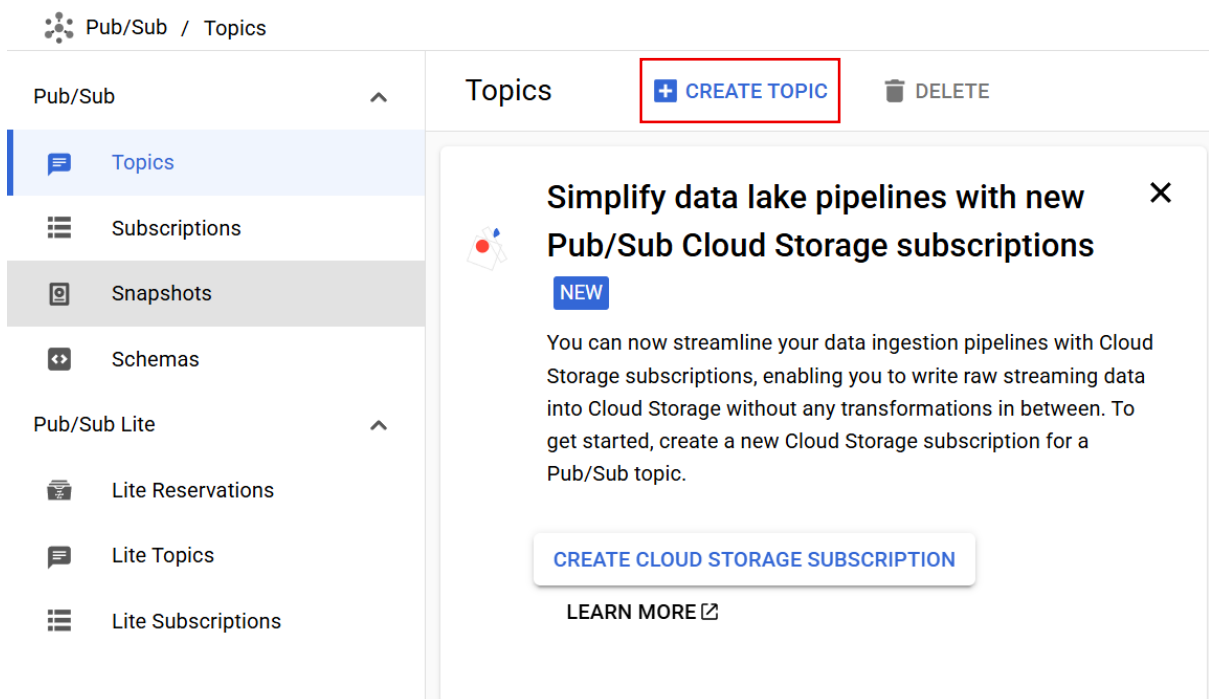
- **GCM** was a legacy push messaging service, now replaced by **Firestore Cloud Messaging**.
- **Pub/Sub** is a robust, scalable, real-time messaging service for complex systems and global apps.
- **Pub/Sub Lite** is the budget-friendly sibling of Pub/Sub, optimized for high-throughput, zonal, and cost-sensitive scenarios.

## To begin with the Lab

1. From your GCP console, search for **pub/sub** and navigate to the service accordingly.



2. Now, to work with pub/sub, we need to create a Topic. So, click on Create topic.



3. We are going to keep things simple, just give your topic a name and leave all the other options to default. Click on Create.

Pub/Sub / Topics / Create topic

Pub/Sub
Topics
Subscriptions
Snapshots
Schemas
Pub/Sub Lite
Lite Reservations
Lite Topics
Lite Subscriptions
Release Notes

## Create topic

Topic ID \*  
first-topic

Topic name: projects/still-kit-459403-e2/topics/first-topic

☒ Add a default subscription
☐ Use a schema
☐ Enable ingestion
☐ Enable message retention
☐ Export message data to BigQuery
☐ Backup message data to Cloud Storage

### Encryption

☒ Google-managed encryption key  
Keys owned by Google
☐ Cloud KMS key  
Keys owned by customers

CREATE

- With the topic, a subscription is also created, and it will be the thing that will distribute the messages from the topic.

Pub/Sub / Topics / Topic: first-topic

Pub/Sub
Topics
Subscriptions
Snapshots
Schemas
Pub/Sub Lite
Lite Reservations
Lite Topics
Lite Subscriptions
Release Notes

first-topic
EDIT
TRIGGER CLOUD RUN FUNCTION
IMPORT
SHOW INFO PANEL

Export to BigQuery

Export data to a BigQuery table.

EXPORT DATA

Export to Cloud Storage

Export data to a text or Avro file in Cloud Storage.

EXPORT DATA

SUBSCRIPTIONS
SNAPSHOTS
METRICS
DETAILS
MESSAGES

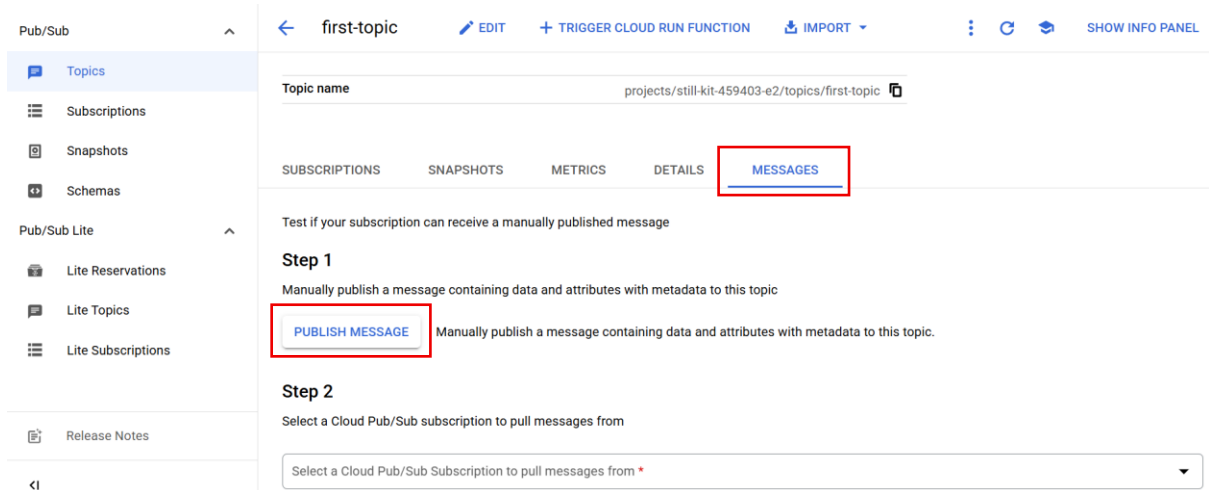
Only subscriptions attached to this topic are displayed. A subscription captures the stream of messages published to a given topic. You can also stream messages to BigQuery or Cloud Storage by creating a subscription from a Cloud Dataflow job. [Learn more](#)

CREATE SUBSCRIPTION EXPORT

Filter Filter subscriptions

Subscription ID	Subscription name	Project
<a href="#">first-topic-sub</a>	projects/still-kit-459403-e2/subscriptions/first-topic-sub	still-kit-459403-e2

- Let's test out the topic and subscription for that, click on messages, and then choose to publish a message using step 1.



6. To publish a message, we just need to type a message in the **Message body** and then click on Publish.

### Message body

The message you want to publish to this topic. Either message or attribute will be required to publish.

**Message \***

This is just a test message. |

Message size should not exceed 10MB.

### Message attributes

**PUBLISH** **CANCEL**

7. After publishing your message, we need to pull it from step 2, choose your subscription.
8. Then click on enable ack messages, and in the end just click on Pull.

#### Step 2

Select a Cloud Pub/Sub subscription to pull messages from

Select a Cloud Pub/Sub Subscription to pull messages from \*  
projects/still-kit-459403-e2/subscriptions/first-topic-sub

Click Pull to view messages and temporarily delay message delivery to other subscribers. Select Enable ACK messages and then click ACK next to the message to permanently prevent message delivery to other subscribers.

**PULL** ☒ Enable ack messages

9. It will start pulling your messages, and in no time, you can see your message.

PULL ☒ Enable ack messages

Filter Filter messages				?	☰
Publish time	Attribute keys	Message body		Ack	↑
May 22, 2025, 10:03:34 PM	—	This is just a test message.			ACK

10. Now we are going to publish the message given below and then pull it.

```
{
  "orderId" : "54",
  "orderDate" : "01/09/2020 17:34:02",
  "total" : 55,
  "priority" : 2,
  "items": [
    {
      "name" : "Harry Potter",
      "id" : "443",
      "price" : 55
    }
  ]
}
```

11. In the message body, paste your message and click on publish.

#### Message body

The message you want to publish to this topic. Either message or attribute will be required to publish.

Message \*

```
{
  "orderId" : "54",
  "orderDate" : "01/09/2020 17:34:02",
  "total" : 55,
  "priority" : 2,
  "items": [
    {
      "name" : "Harry Potter",
      "id" : "443",
      "price" : 55
    }
  ]
}
```

Message size should not exceed 10MB.

#### Message attributes

At most 100 attributes per message, attribute key should not start with 'goog' and should not exceed 256 bytes, attribute value should not exceed 1024 bytes.


ADD ATTRIBUTE

PUBLISH

CANCEL

12. Below you can see your message, you can also scroll it to right to view its more properties.

 **Filter** Filter messages

Publish time 	Attribute keys	Message body	Body JSON keys
May 22, 2025, 10:08:58 PM	—	{ "orderId" : "54", "orderDate" : "01/09/2020 17:34:02", "total" : 55, "priority" : 2, "items": [ {"name" : "Harry Potter", "id" : "443", "price" : 55 } ] }	<div>orderId</div> <div>orderDate</div> <div>total</div> <div>priority</div> <div>items[0].name</div> <div>items[0].id</div> <div>items[0].price</div>