Pub Sub Synopsis

Overview
Google Cloud Pub/Sub is a fully managed, real-time messaging service that enables asynchronous communication between independent applications. It follows a **publisher–subscriber model**, allowing producers to send messages to topics, and consumers to receive those messages through subscriptions. This decouples services from each other, ensuring scalability, reliability, and real-time data flow.

Problem Statement
In distributed systems, direct communication between applications creates tight coupling, scaling limitations, and message delivery failures. There is a need for a robust messaging layer to enable seamless and reliable communication across microservices.

Objectives

- Enable asynchronous message processing between distributed components
- Improve system reliability through decoupling
- Ensure real-time delivery with minimal latency
- Support horizontal scaling of services

Working Principle

1. **Publisher** sends messages to a **Topic**
2. **Topic** routes messages to **Subscriptions**
3. **Subscriber** applications pull/push messages from these subscriptions
4. Pub/Sub ensures:
    - **At-least-once delivery**
    - **Message persistence**
    - **Fault tolerance**

Key Features

- Scalable event streaming system
- Automatic load balancing for subscribers
- Encryption of messages in transit and at rest

- Integration with Cloud Functions, Dataflow, BigQuery, Cloud Run, etc.

Use Cases

- Real-time event streaming
- Microservices communication
- Log processing and analytics pipelines
- Triggering serverless functions

Pub/Sub – gcloud Commands Practical Flow

1️⃣ Enable Pub/Sub API

```
gcloud services enable pubsub.googleapis.com
```

Enables Pub/Sub service in the current GCP project.

2️⃣ Verify Pub/Sub is Enabled

```
gcloud services list --enabled | grep pubsub
```

Checks if the Pub/Sub API is active.

3️⃣ Create Topic

```
gcloud pubsub topics create user-events-arun
```

Creates a topic named **user-events-arun** for publishing events.

4️⃣ List All Topics

```
gcloud pubsub topics list
```

Displays all topics in the project.

5️⃣ Create Subscription

```
gcloud pubsub subscriptions create user-event-sub-arun --topic=user-events-arun
```

Subscription **user-event-sub-arun** listens to messages from the topic.

6️⃣ List All Subscriptions

```
gcloud pubsub subscriptions list
```

Shows all subscriptions configured in the project.

7️⃣ Publish a Message

```
gcloud pubsub topics publish user-events-arun --message='{"event":"User_Created","user":"Arun"}'
```

Sends a single JSON message to the topic.

8️⃣ Pull Messages

```
gcloud pubsub subscriptions pull user-event-sub-arun --limit=5 --auto-ack
```

Reads up to 5 messages and auto-acknowledges them.

Trigger Cloud Function – Practical Flow

Used for **automatic execution** when Pub/Sub receives a message.

1️⃣ Deploy Cloud Function

```
gcloud functions deploy consume_pubsub_arun \
--gen2 \
--region=asia-south2 \
--runtime=python310 \
--source=. \
--entry-point=consume_pubsub_arun \
--trigger-topic=user-events-arun
```

Deploys a Cloud Function named **consume_pubsub_arun**
triggered whenever a message is published to **user-events-arun**.

2️⃣ Publish Events (Trigger Cloud Function)

➡ For single user event:

```
gcloud pubsub topics publish user-events-arun --message="{\"event\":\"User_Created\",\"user\":\"arun\"}"
```

➡ For multiple users (batch processing):

```
gcloud pubsub topics publish user-events-arun \
--message="{\"event\":\"User_Created\",\"users\":[\"arun\",\"pooja\",\"rahul\",\"akhil\"]}"
```

This triggers the Cloud Function **only once**, allowing it to process multiple records in a single execution.

📌 Result

✓ Topic created
✓ Messages published
✓ Logs visible in Cloud Function → **“PUB/SUB MESSAGE RECEIVED”**
✓ Subscription received the message successfully

Conclusion
Google Cloud Pub/Sub provides a powerful and highly scalable messaging backbone for event-driven applications. By decoupling components and ensuring reliable real-time message delivery, it significantly enhances system performance, flexibility, and maintainability in both cloud-native and hybrid deployments.