☁️ Cloud Pub/Sub and Cloud Scheduler — Lab Summary

🔍 Overview

This lab focuses on **Google Cloud Pub/Sub**, a fully managed, asynchronous messaging service that allows independent applications to communicate using messages in real time.
You'll also interact with **Cloud Scheduler** for scheduled message publishing (cron jobs), **Pub/Sub Snapshots** for message state capture, and **Pub/Sub Lite** for cost-efficient, region-based messaging.

🧠 Concepts You Should Know

1. Topic

- A *topic* is a named resource to which publishers send messages.
- Topics act as communication channels.
- Example: gcloud-pubsub-topic

2. Subscription

- A *subscription* represents a stream of messages from a topic to be delivered to subscribers.
- Subscriptions can either **pull** messages (on request) or **push** them (HTTP endpoint).
- Example: pubsub-subscription-message

3. Publisher

- A *publisher* sends messages to a topic.
- Example message: "Hello World"

4. Subscriber

- A *subscriber* receives messages from a subscription.
- Messages can be acknowledged (ACK) once processed.

5. Snapshot

- A *snapshot* captures the state of a subscription at a given point in time.
- Used to replay or recover messages.

6. Cloud Scheduler

- A *fully managed cron job service* that can trigger Pub/Sub topics at specific intervals.

7. Pub/Sub Lite

- A *lower-cost version* of Pub/Sub optimized for predictable throughput and regional use cases.

⚙️ Lab Tasks & Commands

Task 1: Publish a Message to the Topic

Steps:

1. **Create a subscription** for the pre-created topic:
2. gcloud pubsub subscriptions create pubsub-subscription-message \
3. --topic=gcloud-pubsub-topic
4. **Publish a message** to the topic:
5. gcloud pubsub topics publish gcloud-pubsub-topic --message="Hello World"

✅ **Expected Result:**

The message “Hello World” is successfully published to the topic.

Task 2: View the Published Message

Steps:

1. **Pull messages** from your subscription:
2. gcloud pubsub subscriptions pull pubsub-subscription-message --limit=5
3. (Optional) Automatically acknowledge messages:
4. gcloud pubsub subscriptions pull pubsub-subscription-message \
5. --limit=5 --auto-ack

✅ **Expected Result:**

The output displays the message data:

DATA: Hello World

Task 3: Create a Pub/Sub Snapshot

Purpose:

Snapshots are used to **preserve the message delivery state** of a subscription.
You can later *seek* the subscription back to this snapshot if needed.

Step:

gcloud pubsub snapshots create pubsub-snapshot \

--subscription=gcloud-pubsub-subscription

✅ **Expected Result:**

A snapshot named pubsub-snapshot is created successfully.

🧩 **Additional Concepts (for Full Understanding)**

🔸 **Cloud Scheduler + Pub/Sub Integration**

- You can automate publishing messages by scheduling them with Cloud Scheduler.
- Example:
- gcloud scheduler jobs create pubsub send-message-job \
- --schedule="* * * * *" \
- --topic=gcloud-pubsub-topic \
- --message-body="Automated message"

🔸 **Pub/Sub Lite Setup**

- Pub/Sub Lite is ideal for regional systems where you control partitioning and throughput.
- Key differences from standard Pub/Sub:
    - Lower cost.
    - Requires you to specify capacity (storage, throughput).
    - Not globally replicated.

📘 **Summary of Key Takeaways**

Concept	Description	Example Command
Topic	Message channel for publishers and subscribers	gcloud pubsub topics create my-topic
Subscription	Pulls/pushes messages from a topic	gcloud pubsub subscriptions create my-sub --topic=my-topic
Publish Message	Sends a message to a topic	gcloud pubsub topics publish my-topic --message="Hello"
Pull Message	Views messages from a subscription	gcloud pubsub subscriptions pull my-sub --limit=5
Snapshot	Captures subscription message state	gcloud pubsub snapshots create snap --subscription=my-sub
Scheduler Job	Triggers Pub/Sub at intervals	gcloud scheduler jobs create pubsub my-job --schedule="* * * * *" --topic=my-topic
Pub/Sub Lite	Cost-efficient regional messaging service	Set up via console or gcloud pubsub lite-* commands

---

🗞️ **Real-World Use Cases**

- **Microservices communication** – e.g., one service publishes an event, others consume it.
- **IoT pipelines** – sensors send continuous data streams.
- **Event-driven systems** – trigger workflows when new data arrives.
- **Data ingestion pipelines** – Pub/Sub used as buffer before storing data in BigQuery or Dataflow.
- **Scheduled notifications** – using Cloud Scheduler to automate publishing.

✅ **End of Lab Notes**

What you achieved:

- Understood core concepts of Google Cloud Pub/Sub.
- Created subscriptions and published messages.
- Viewed and verified message delivery.
- Created a snapshot for message state recovery.
- Learned integration with Cloud Scheduler and Pub/Sub Lite.