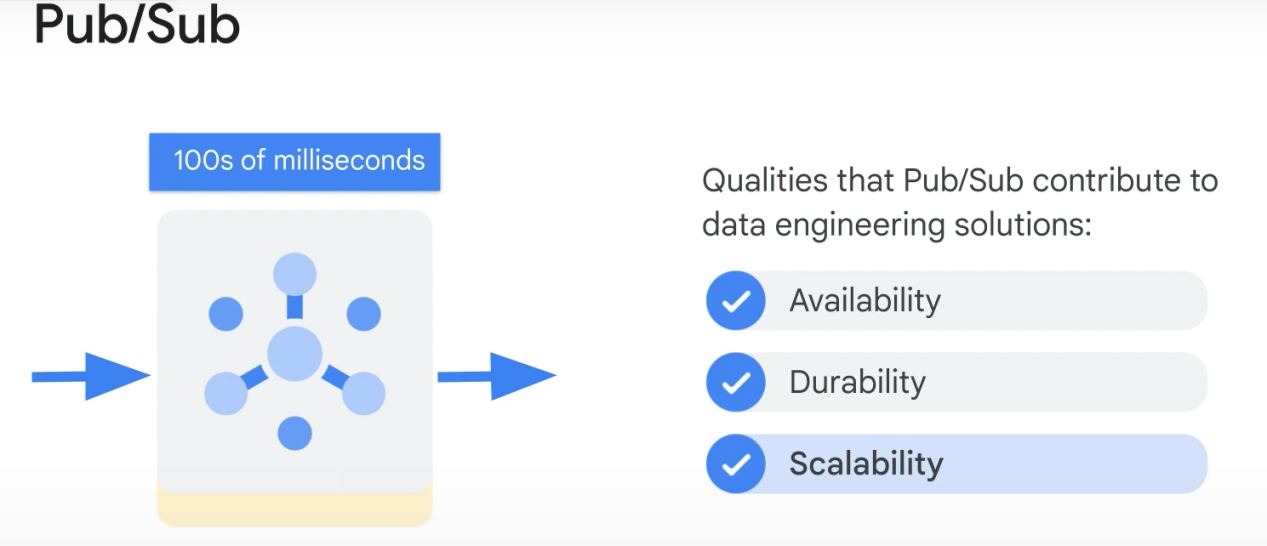
\* No need for the apps to be online all the time

\* The apps don’t need to know how to communicate with each other. They only need to know how to communicate with Pub/Sub

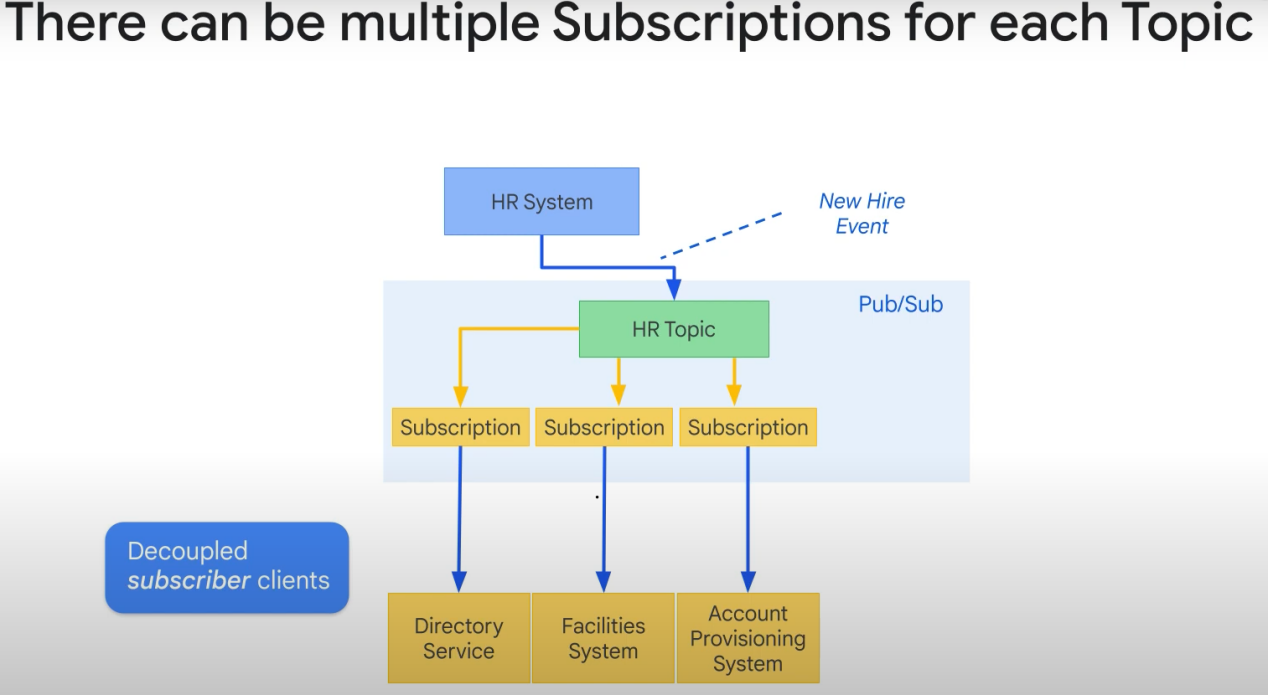
\* offers APIs on C#, Java, Python, Go, NodeJS, Ruby

\* Pub/Sub will by default save your messages for **7 days i**f your systems are down and not able to process them

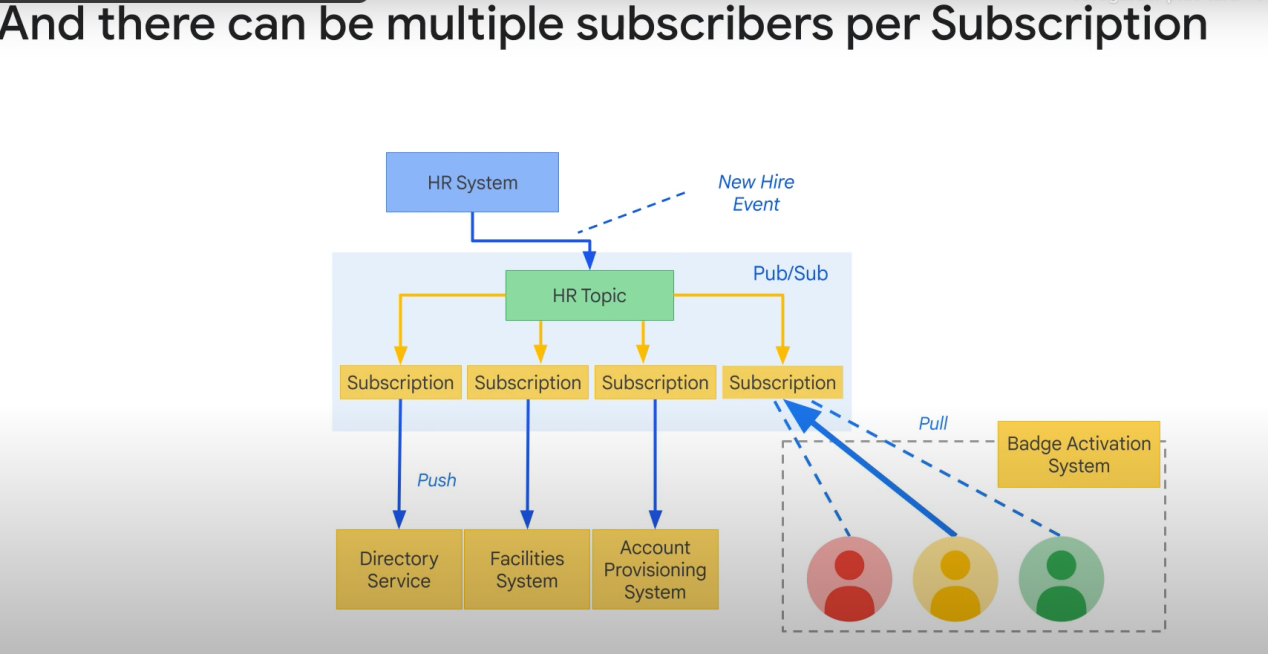
\* Pub/Sub is highly scalable (Google uses it and gets 100 Million/msgs per sec!)



End-to-end encryption. Messages are encrypted during transportation and during rest.

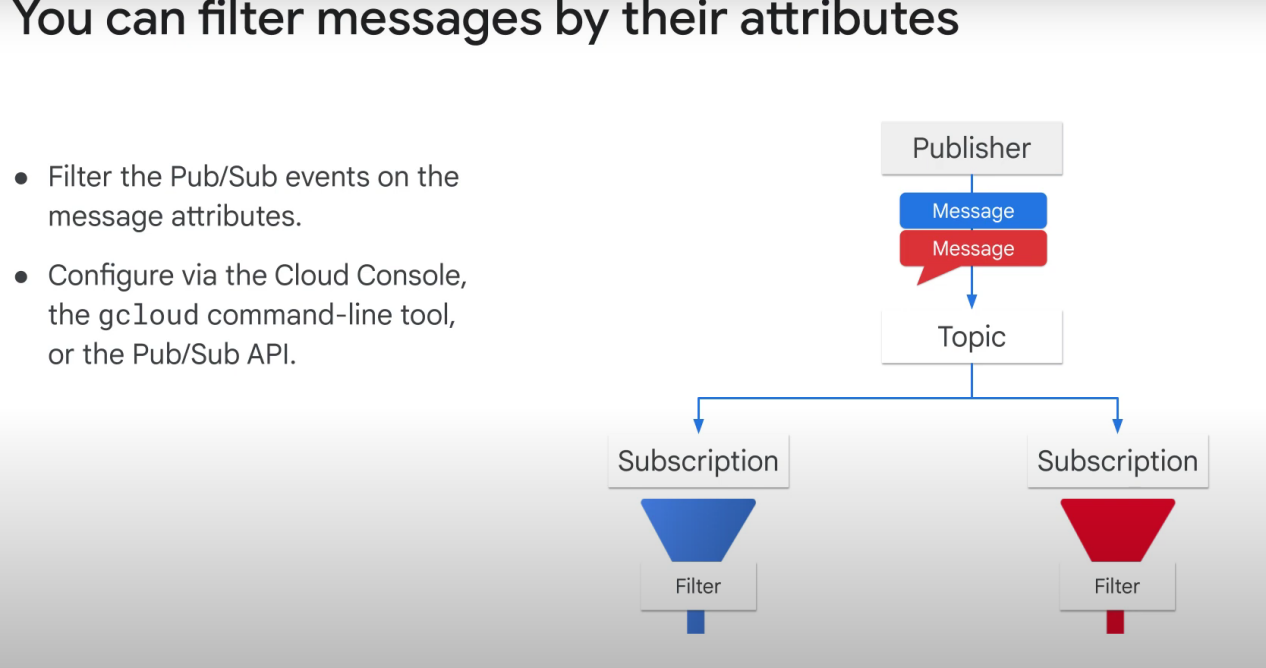


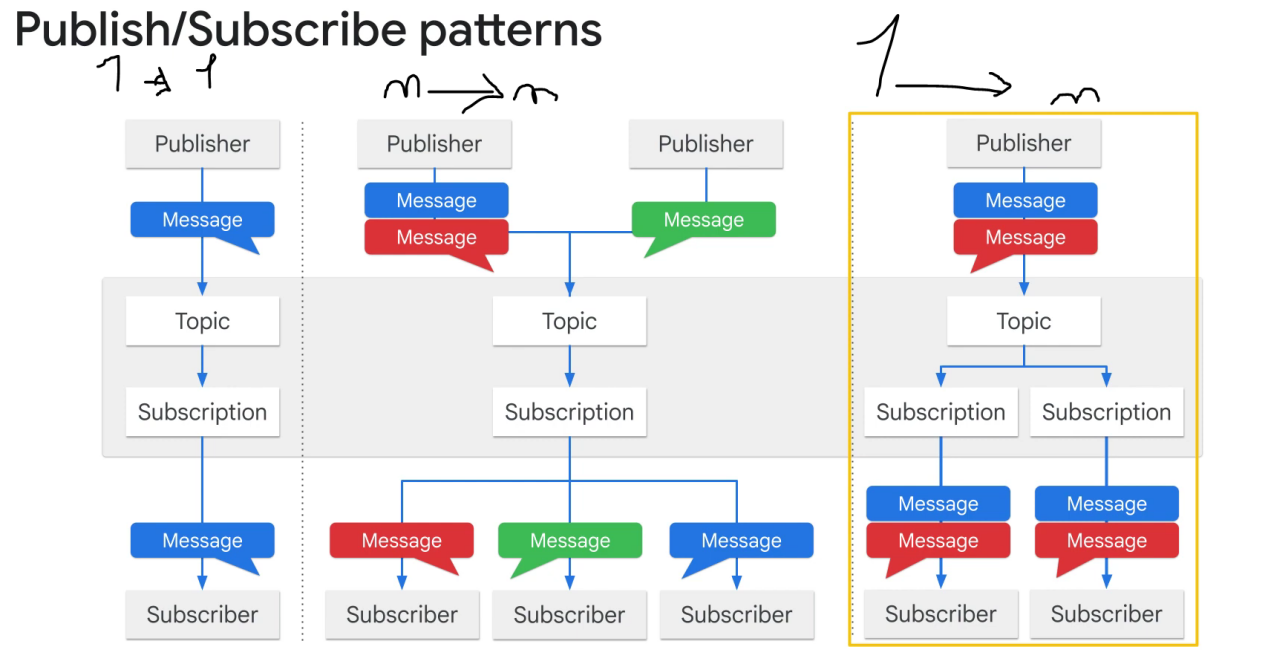
(you can also have many subscribers for 1 subscription)



The badge activation system require a human being to activate the badge. The msg will be available to all the subscribers. One of them NEEDS TO PULL THE MESSAGE.

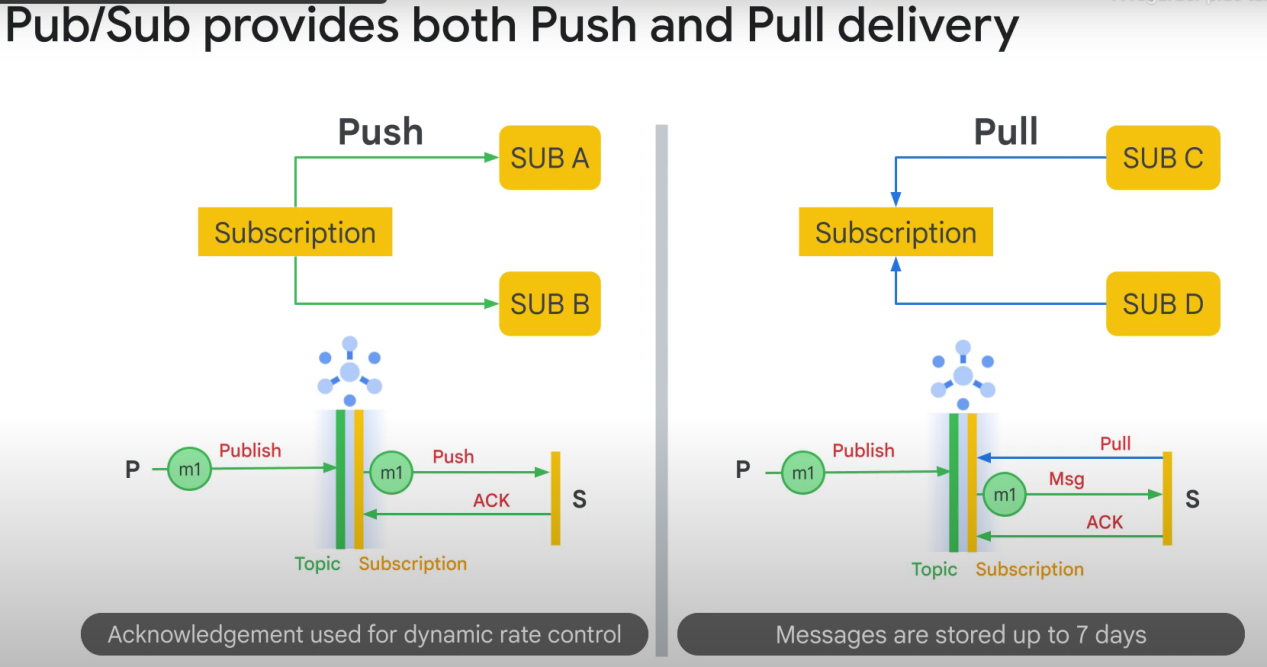
The other subscriptions here are «Push» subscriptions.

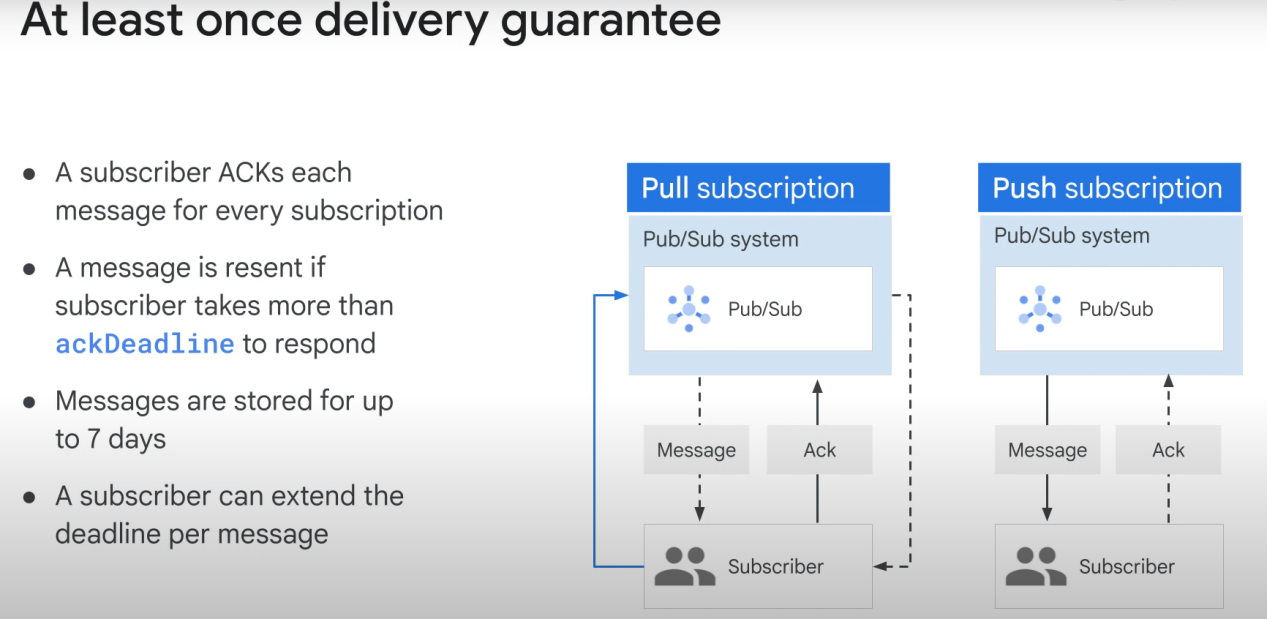




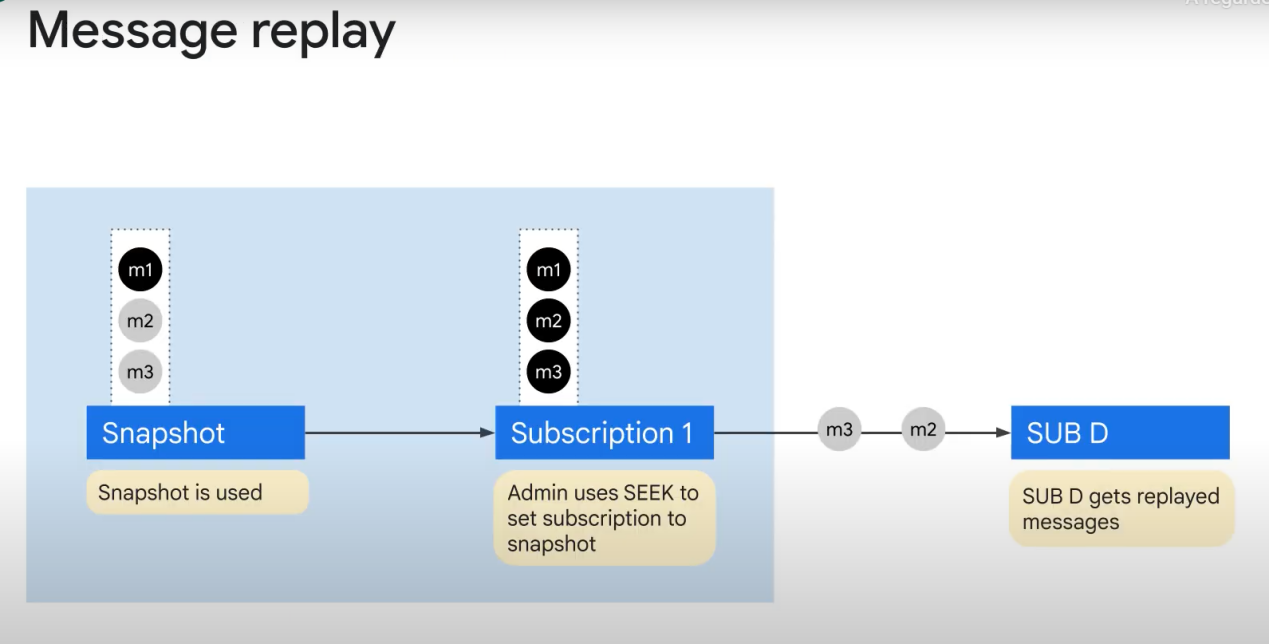
1. n Fan-in load balancing
2. N Fan-out (many use cases for the same data)

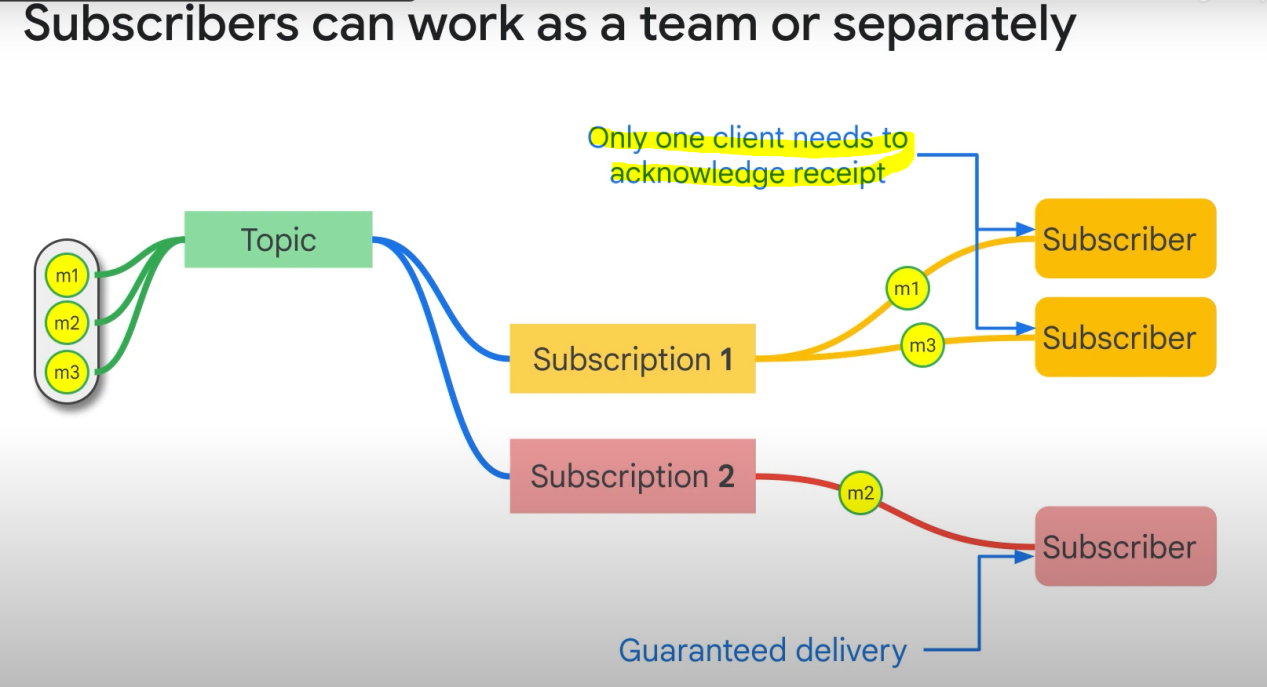
Note: In a pub/sub model, any message published to a topic is immediately received by all of the subscribers to the topic (kind of like fanout?)





Configuring a **topic** with **message retention** gives you more flexibility: it **allows any subscription** attached to the topic **to go back in time** and read old messages.

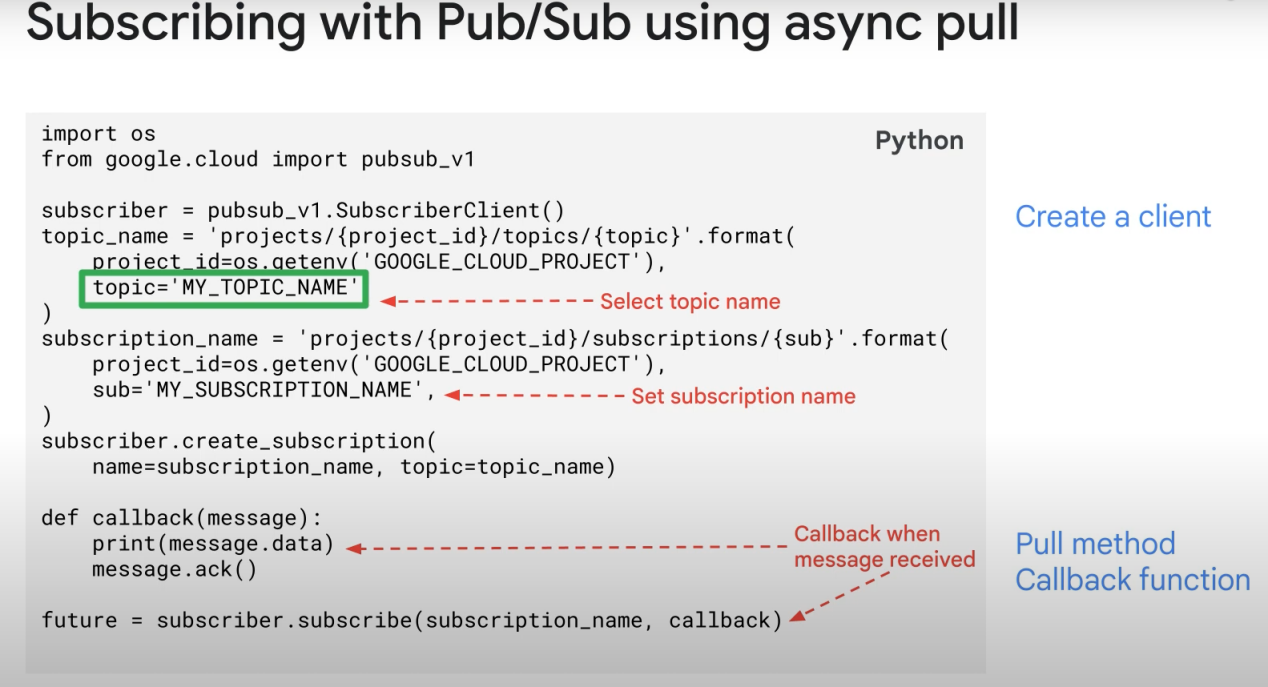


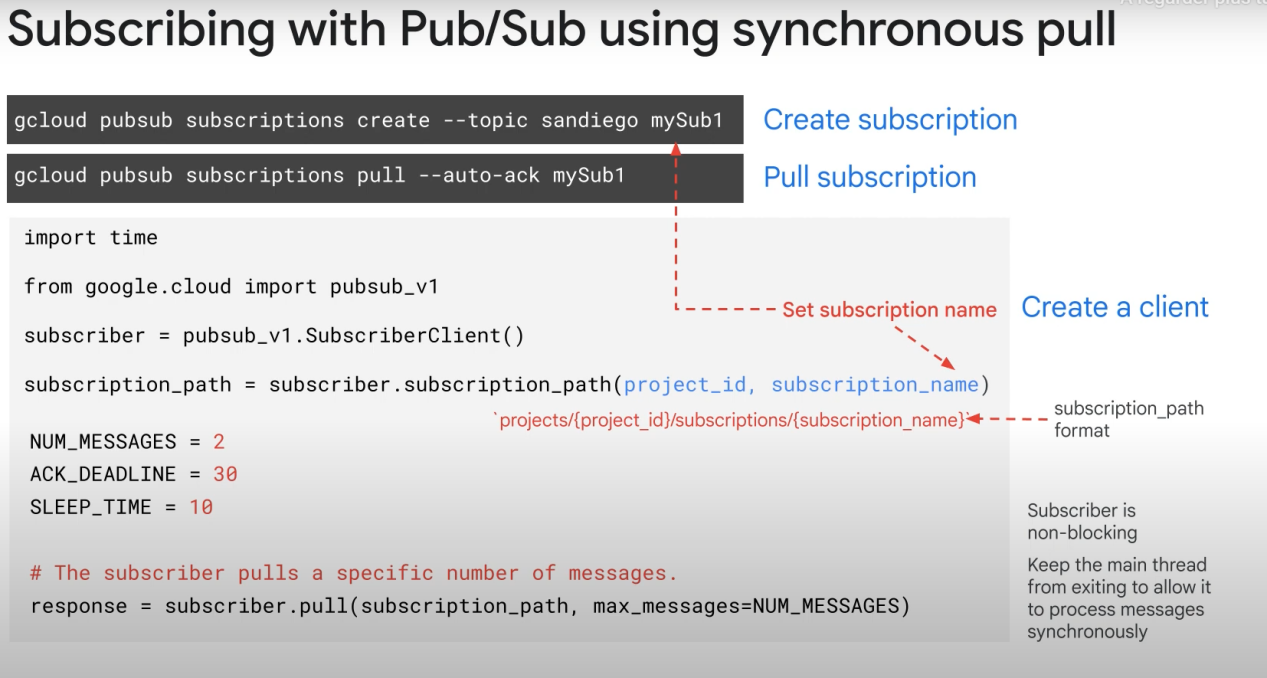


Push sub: 1 endpoint, and 1 subscriber (typically)

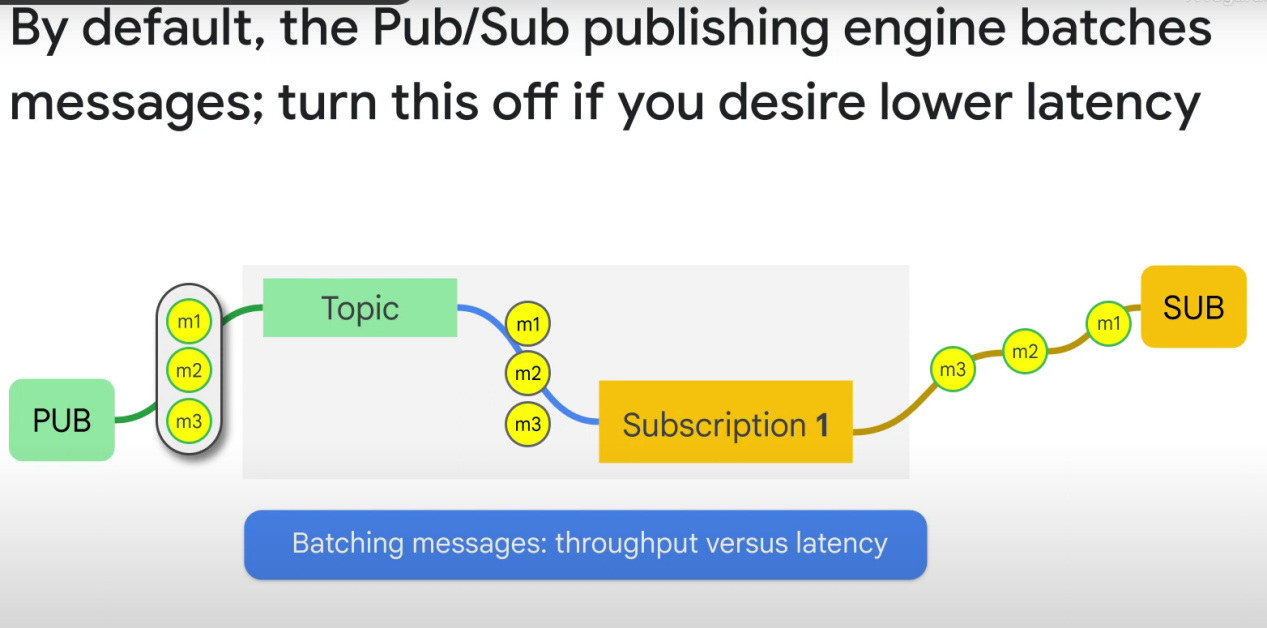


(you can push/publish either using gcloud console or code)

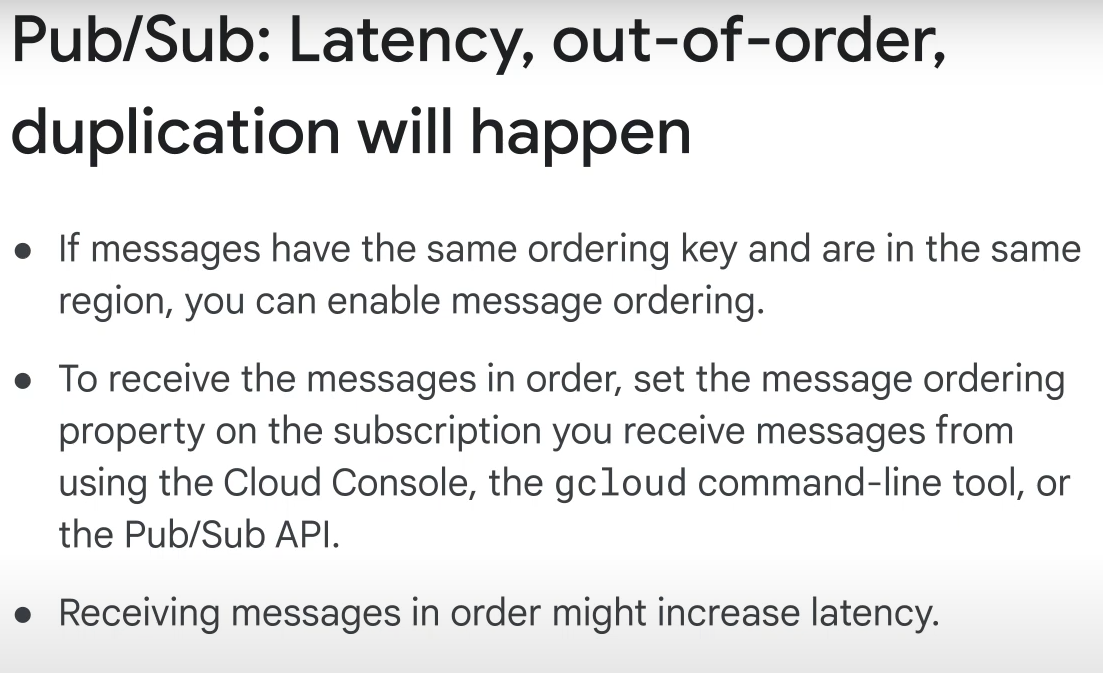




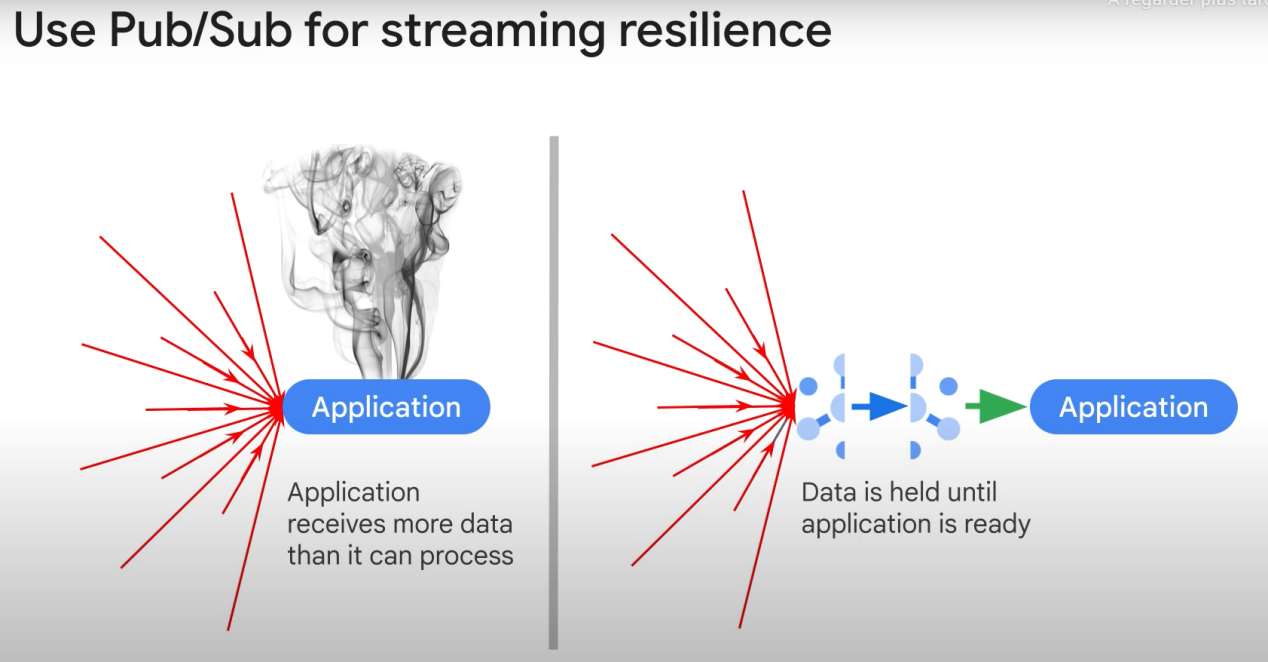
By default, it will only pull the last message. But you can modify it with the «\_\_LIMIT» parameter

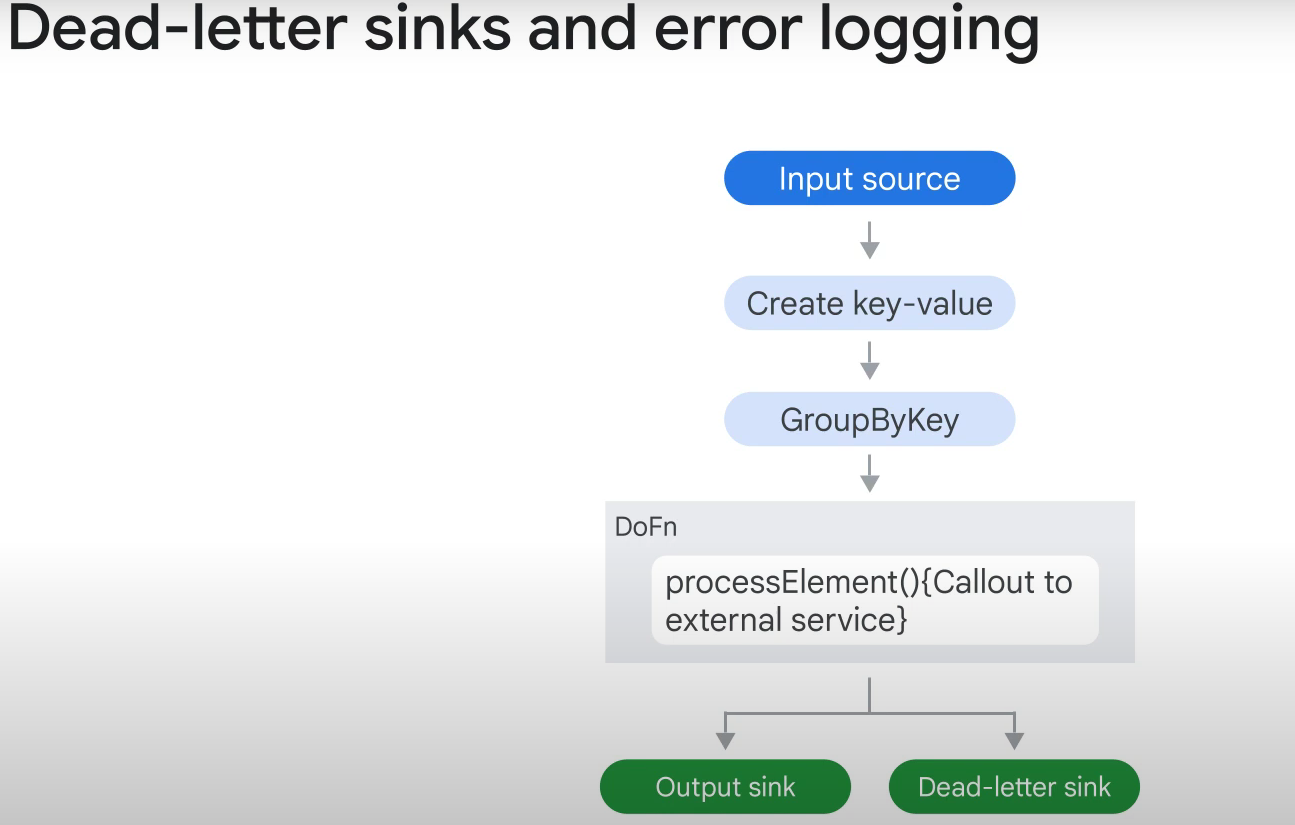


**Batch publish**: Send N messages together, but they will be delivered one after the other



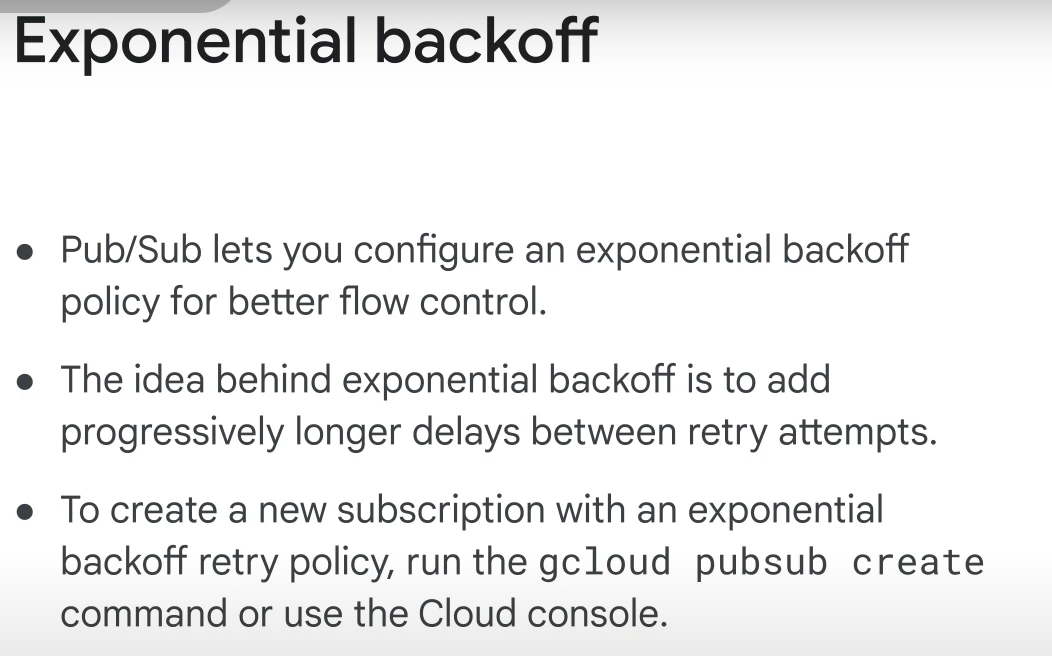
By default, messages don’t necessarly always arrive on order unless the message ordering property is activated. This can cause some latency.



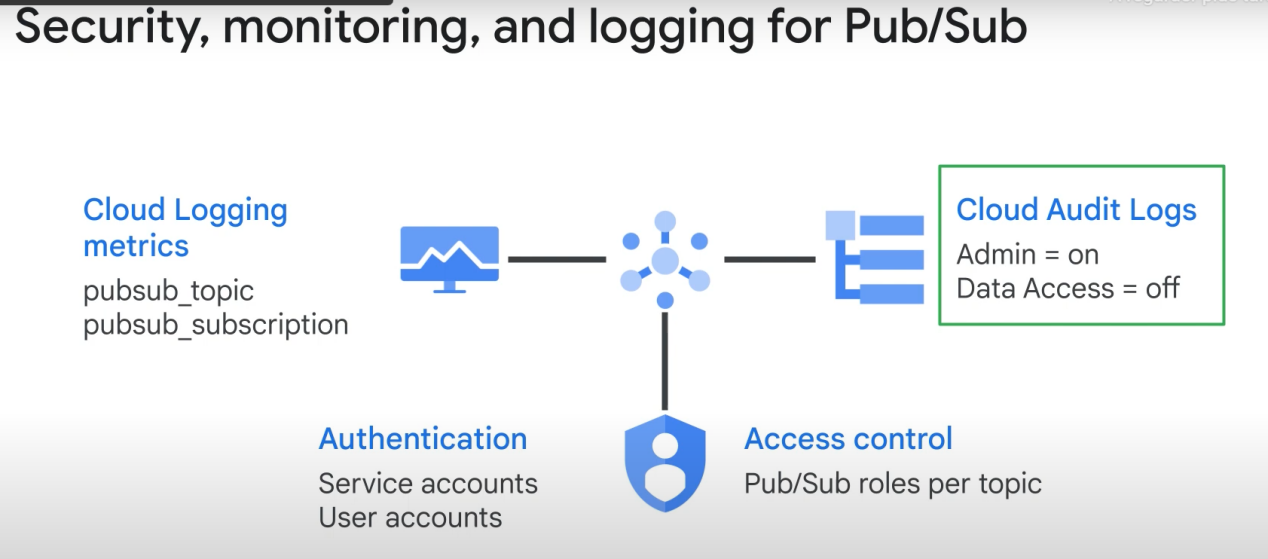


More on the Dead-letter topic

Maximum number of delivery attempts: **A numeric value that signifies the number of delivery attempts that Pub/Sub makes for a specific message**. If the subscriber client cannot acknowledge the message within the configured number of delivery attempts, **the message is forwarded to a dead-letter topic.**



Backoff: delay between retrial



Admin logs (change of settings , etc..) are always On and you can’t disable them.

You need to **enable data access audits**. (logs of data update/delete, etc..)

**Note: Duplication could happen with Pub/Sub but the doublon messages will have the same id == you can filter les doublons with dataflow (or, with limited capabilities, bigquery)**