Cloud Pub/Sub Overview

Next

What is Cloud Pub/Sub?

- Global-scale messaging buffer/coupler
- NoOps, global availability, auto-scaling
- Decouples senders and receivers
- Streaming data ingest:
    - Also connects other data pipeline services
- Equivalent to Apache Kafka (open source)
- Guaranteed at-least-once delivery

No ops $\left\{\begin{array}{l}\text{No second figure}\\\text{No cluster to provision}\end{array}\right.$

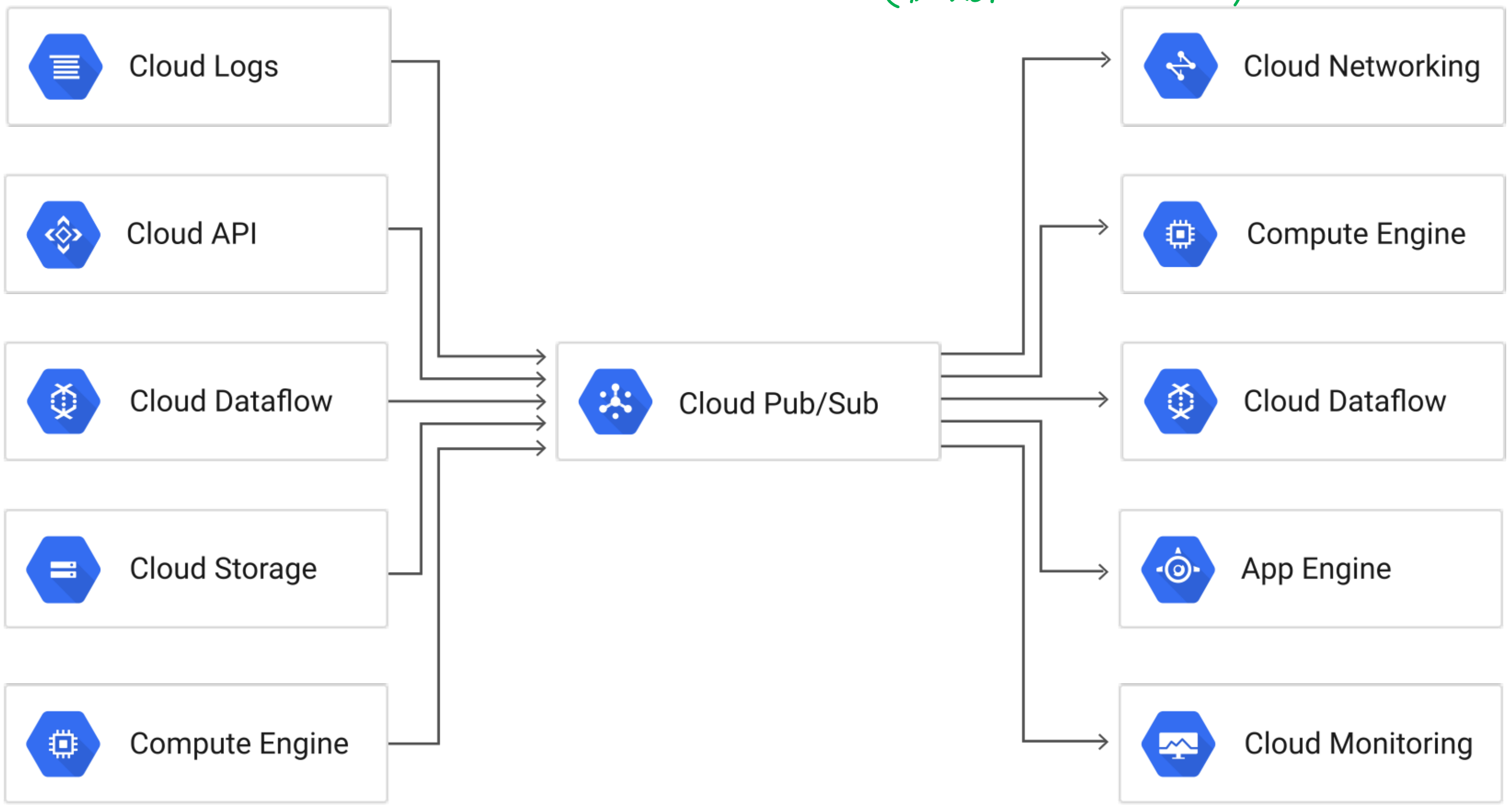
exam question

Terminology

1 to 1
many to many
7 to 8, 6 to 2, doesn't matter

Asynchronous messaging - many to many (or any other combination)

(Sender to reciever)

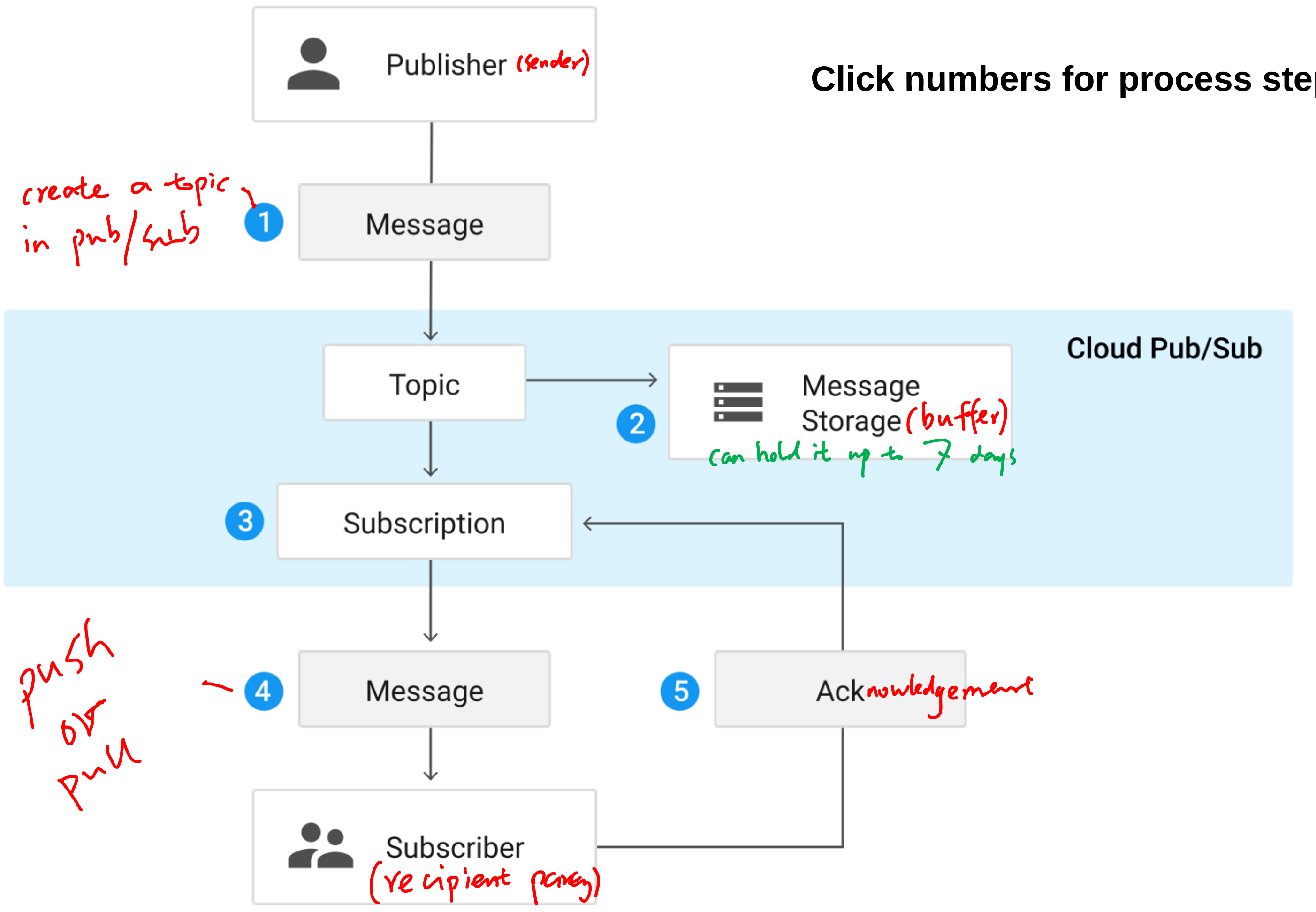


Cloud Pub/Sub Overview

Previous
Next

How It Works: Terminology

- Topics, Messages, Publishers, Subscribers, Message Store



Click numbers for process steps

Cloud Pub/Sub Overview

Previous

Next

Push and Pull (maybe in exam)

- Pub/Sub can either *push* messages to subscribers, or subscribers can *pull* messages from Pub/Sub.
- Push = lower latency, more real-time.
- Push subscribers must be Webhook endpoints that accept POST over HTTPS.
- Pull is ideal for large volume of messages, and uses batch delivery

pull is the default option

IAM (exam)

- Allows for controlling access at project, topic, or subscription level
- Admin, Editor, Publisher, Subscriber
- Service accounts are best practice (especially for built-ints an application)

Pricing

- Data volume used per month (per GB)

Monthly data	Price Per GB
First 10 GB	$0.00
Next 50 TB	$0.06
Next 100 TB	$0.05
Beyond 150 TB	$0.04

Out of order messaging

- Messages may arrive from multiple sources out of order.
- Pub/Sub does not care about message ordering.
- Dataflow is where out of order messages are processed/resolved.
- It's possible to add message attributes to help with ordering.

(Pub/Sub doesn't)

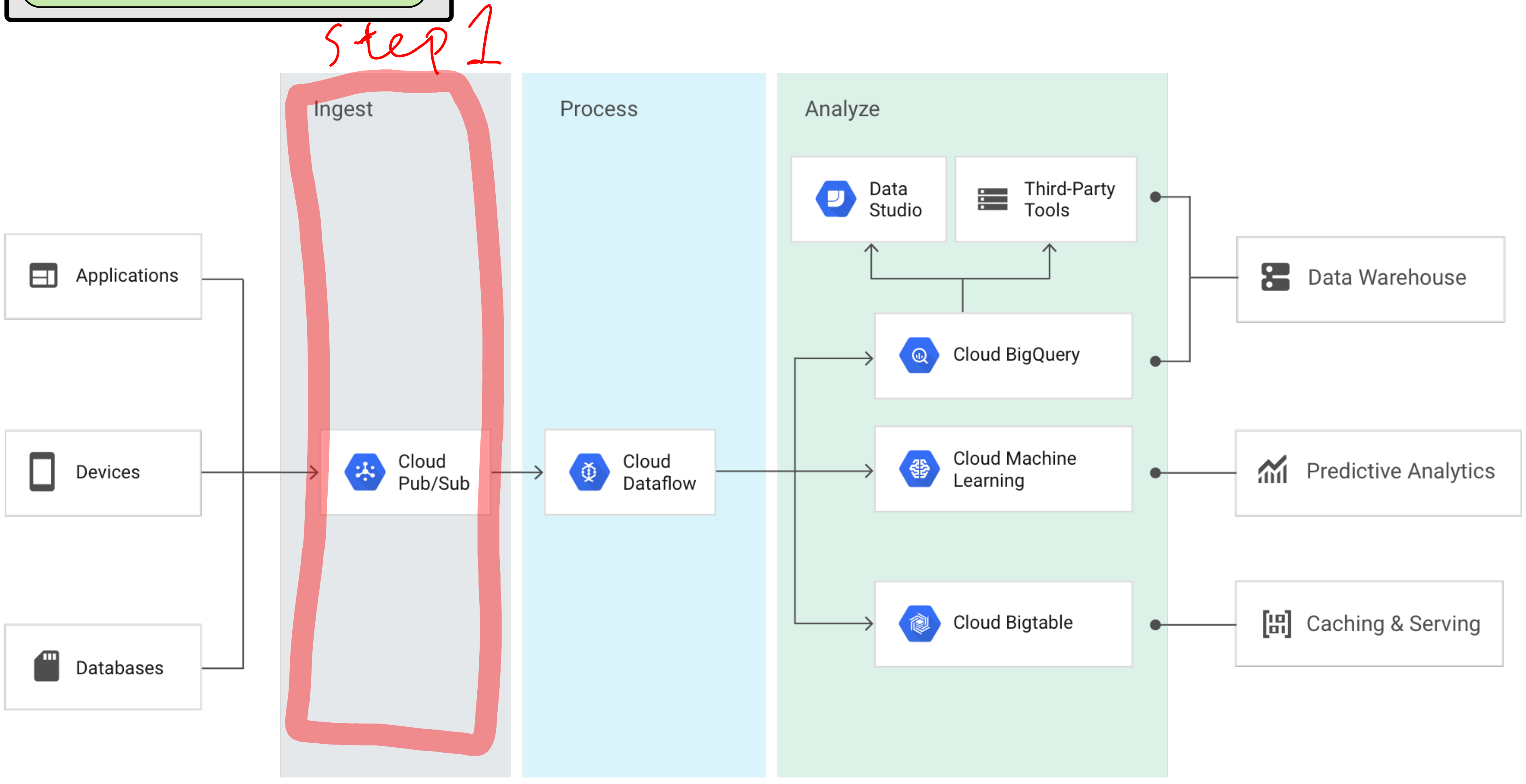
exam

Return to Table of Contents

Cloud Pub/Sub Overview

Previous

Big Picture: Data Lifecycle for Streaming Data Ingest



Return to Table of Contents

Pub/Sub Hands On

Next

The Steps

- Create a topic
- Create a subscription
- Publish messages
- Retrieve messages

Simple topic/subscription/publish via `gcloud`

Create a topic called *my-topic*:

- `gcloud pubsub topics create my-topic`

Create subscription to topic *my-topic*:

- `gcloud pubsub subscriptions create --topic my-topic mySub1`

Publish a message to your topic:

- `gcloud pubsub topics publish my-topic --message "hello"`

Retrieve message with your subscription, acknowledge receipt, and remove message from queue:

- `gcloud pubsub subscriptions pull --auto-ack mySub1`

Cancel subscription:

- `gcloud pubsub subscriptions delete mySub1`

Return to Table of Contents

Pub/Sub Hands On

Previous

Traffic Data Exercise

- **Clone GitHub**
- **Copy data points**
- **Simulate traffic data**
- **Pull messages**

Clone GitHub data to Cloud Shell (or other SDK environment), and browse to publish folder:

```
cd ~
git clone https://github.com/linuxacademy/googledataengineer
cd ~/googledataengineer/courses/streaming/publish
```

Create a topic called *sandiego*:

```
gcloud pubsub topics create sandiego
```

Create subscription to topic *sandiego*:

```
gcloud pubsub subscriptions create --topic sandiego mySub1
```

Run script to download sensor data:

```
./download_data.sh
```

May need to authenticate shell to ensure we have the right permissions:

```
gcloud auth application-default login
```

View script info:

```
vim ./send_sensor_data.py or use viewer of your choice
```

Run python script to simulate one hour of data per minute:

```
./send_sensor_data.py --speedFactor=60 \
--project=YOUR-PROJECT-ID
```

If you receive error: *google.cloud.pubsub can not be found* or an *ImportError: No module named iterator*, run this `pip` command to install components, then try again:

```
sudo pip install -U google-cloud-pubsub
```

Open new Cloud Shell tab (using + symbol)

Pull message using subscription *mySub1*:

```
gcloud pubsub subscriptions pull --auto-ack mySub1
```

Create a new subscription and pull messages with it:

```
gcloud pubsub subscriptions create --topic sandiego mySub2
gcloud pubsub subscriptions pull --auto-ack mySub2
```

Connecting Kafka to GCP

Does Pub/Sub Replace Kafka?

Next

- Not always
- Hybrid workloads:
  - Interact with existing tools and frameworks
  - Don't need global/scaling capabilities with pub/sub
- Can use *both*: Kafka for on-premises and pub/sub for GCP in same data pipeline

How do we connect Kafka to GCP?

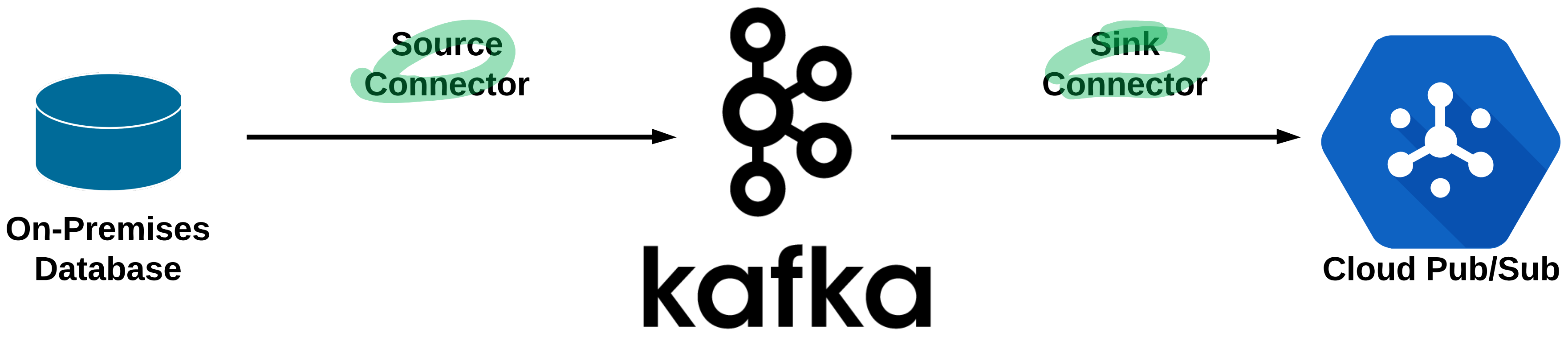
Overview on Connectors:

- Open-source plugins that connect Kafka to GCP
- Kafka Connect: One optional "connector service"
- Exist to connect Kafka directly to pub/sub, Dataflow, and BigQuery (among others)

Additional Terms

exam related terminology

- **Source connector:** An upstream connector:
  - Streams *from* something ~~to~~ → Kafka
- **Sink connector:** A downstream connector:
  - Streams *from* Kafka ~~to~~ → something else



Monitoring Subscriber Health

Trouble shooting

In a Perfect World...

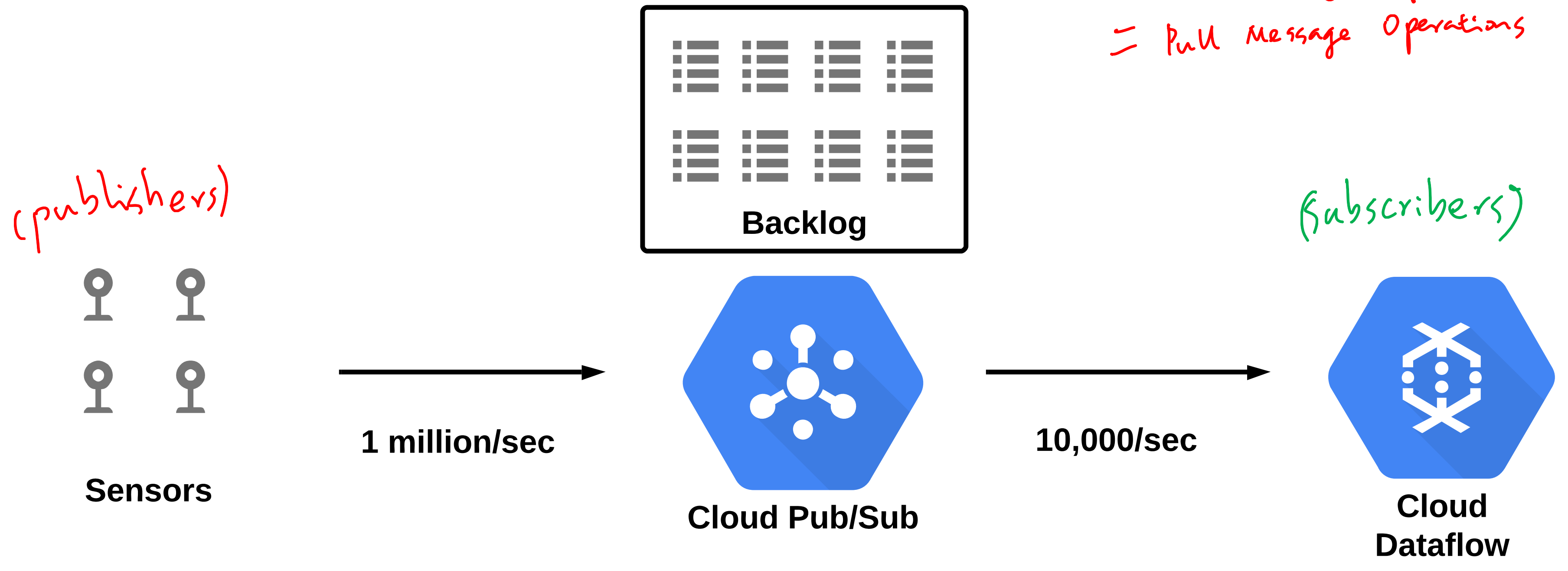
- Subscribers and Publishers work in perfect harmony:
  - Example:
    - 1 million messages/second published
    - 1 million messages/second successfully pulled/pushed
    - Result: No backlog in Pub/Sub queue
- But we don't live in a perfect world...
  - Subscriber cannot keep up with publish rate
  - Result: Backlog in Pub/Sub queue

Troubleshooting Subscriber Health (Backlog)

4 trouble shootings

- ① Create alerts for (x) backlog threshold
- ② Subscriber not able to keep up:
  - two reasons { Under-provisioned
  - Code not optimized
- ③ Not acknowledging message receipt:
  - Pub/Sub doesn't know it's delivered, and keeps trying
  - Subscriber code not properly acknowledging pulled messages
- ④ Check publishers for excessive re-transmits

acknowledging Requests = Pull Message Operations


Backlog
(publishers)
Sensors
1 million/sec
Cloud Pub/Sub
10,000/sec
(subscribers)
Cloud Dataflow