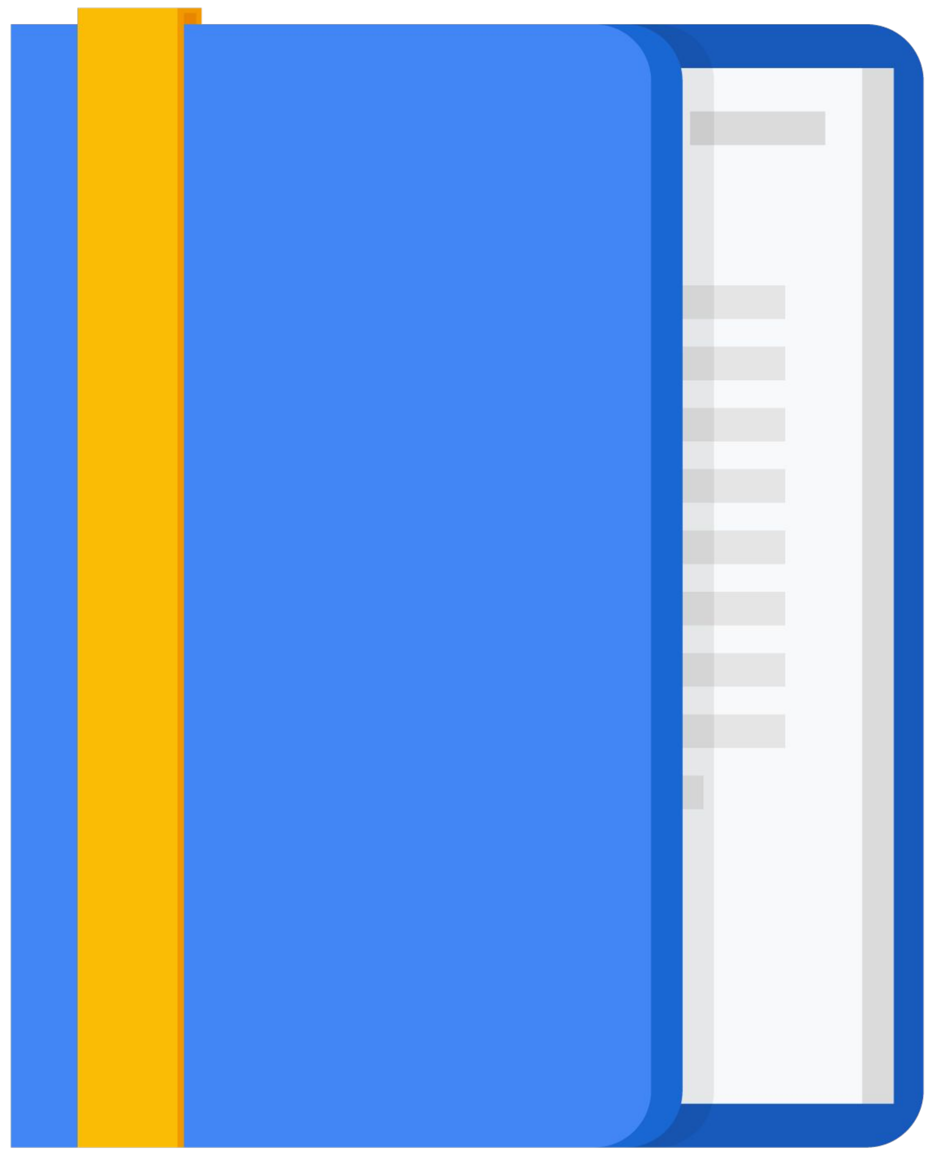
Google Cloud

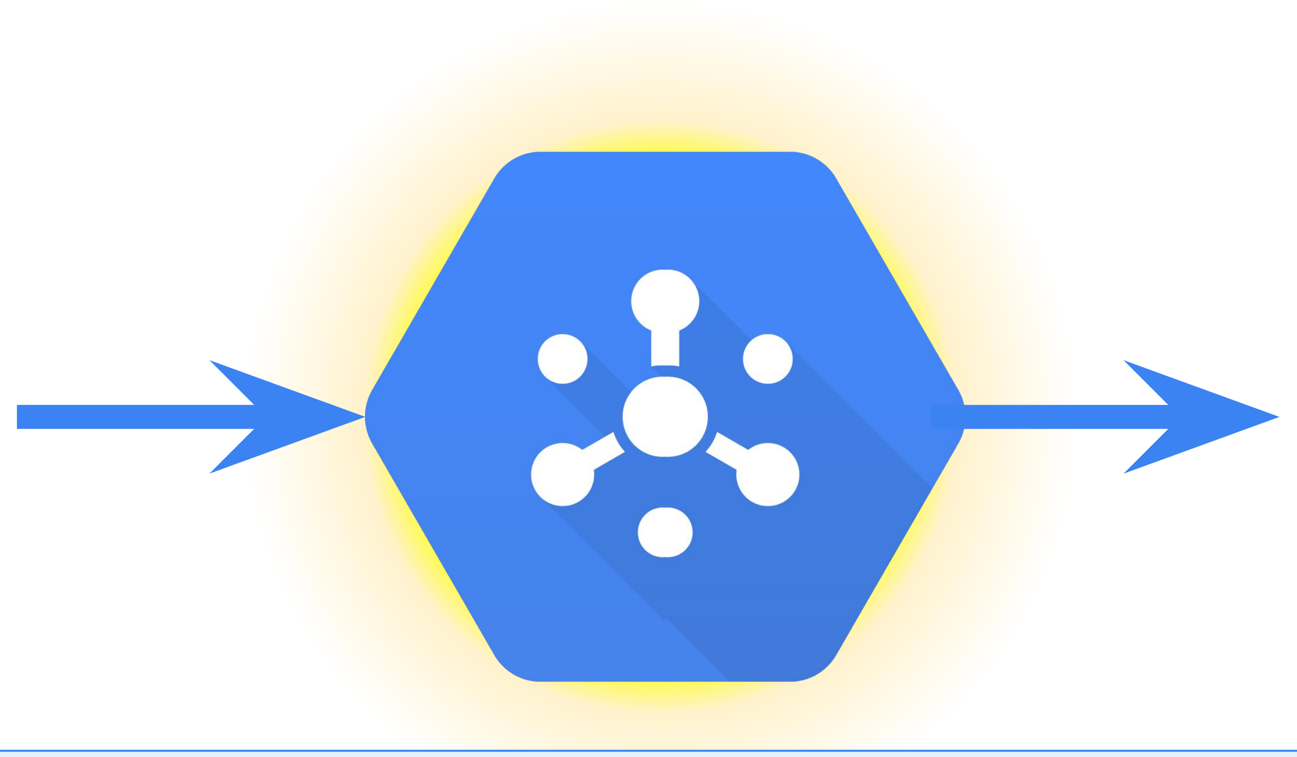
Serverless Messaging with Pub/Sub

Agenda

- Processing Streaming Data
- **Cloud Pub/Sub**
- Cloud Dataflow Streaming Features
- BigQuery and Bigtable Streaming Features
- Advanced BigQuery Functionality



Cloud Pub/Sub



Cloud
Pub/Sub

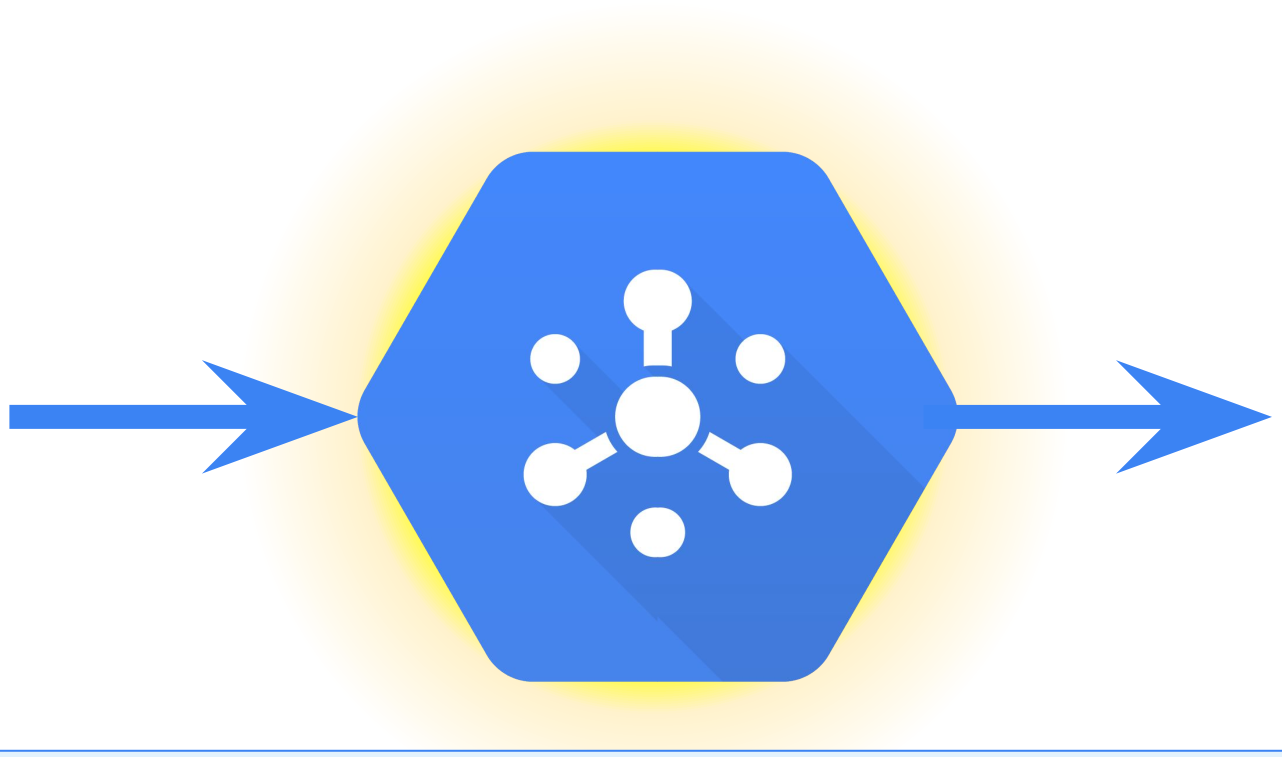
Qualities that Cloud Pub/Sub contribute to Data Engineering solutions:

- Availability
- Durability
- Scalability

100s of milliseconds

Google Cloud

Cloud Pub/Sub



Cloud
Pub/Sub

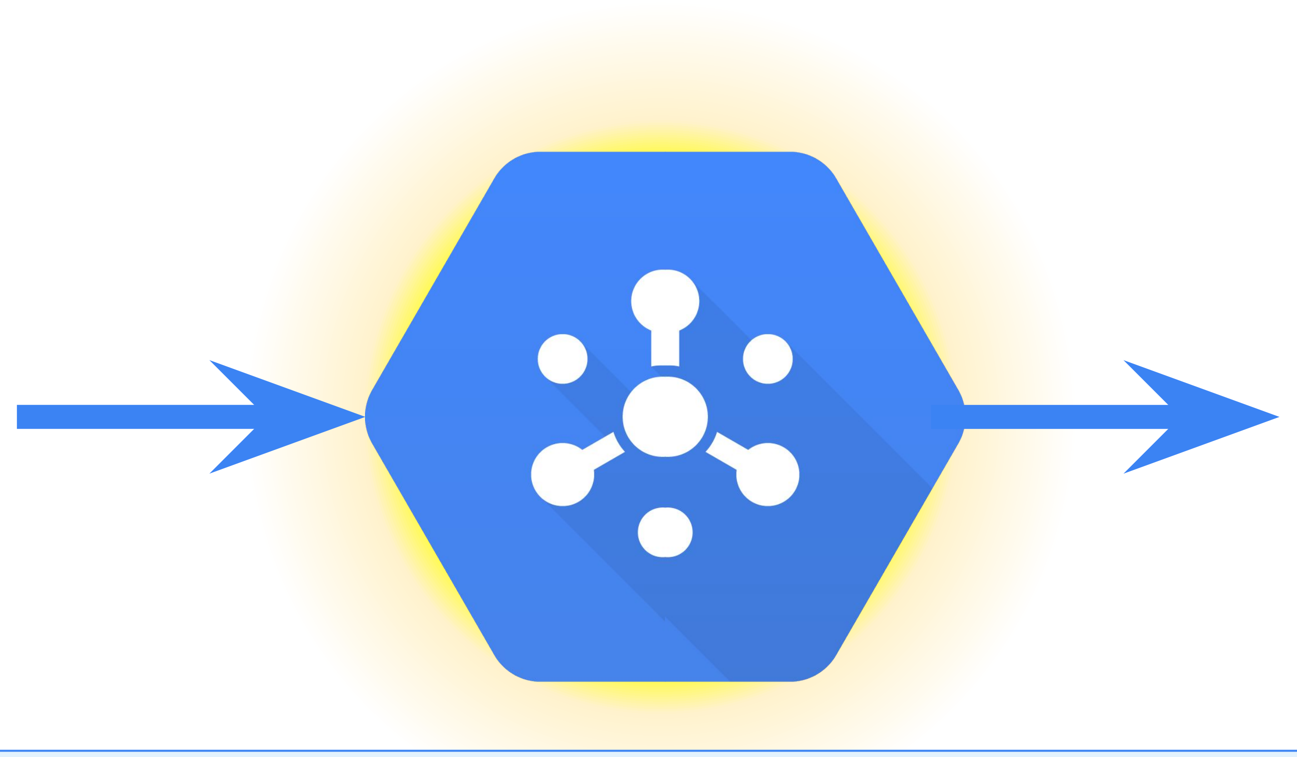
Qualities that Cloud Pub/Sub contribute to Data Engineering solutions:

Availability
Durability
Scalability

100s of milliseconds

Google Cloud

Cloud Pub/Sub



Cloud
Pub/Sub

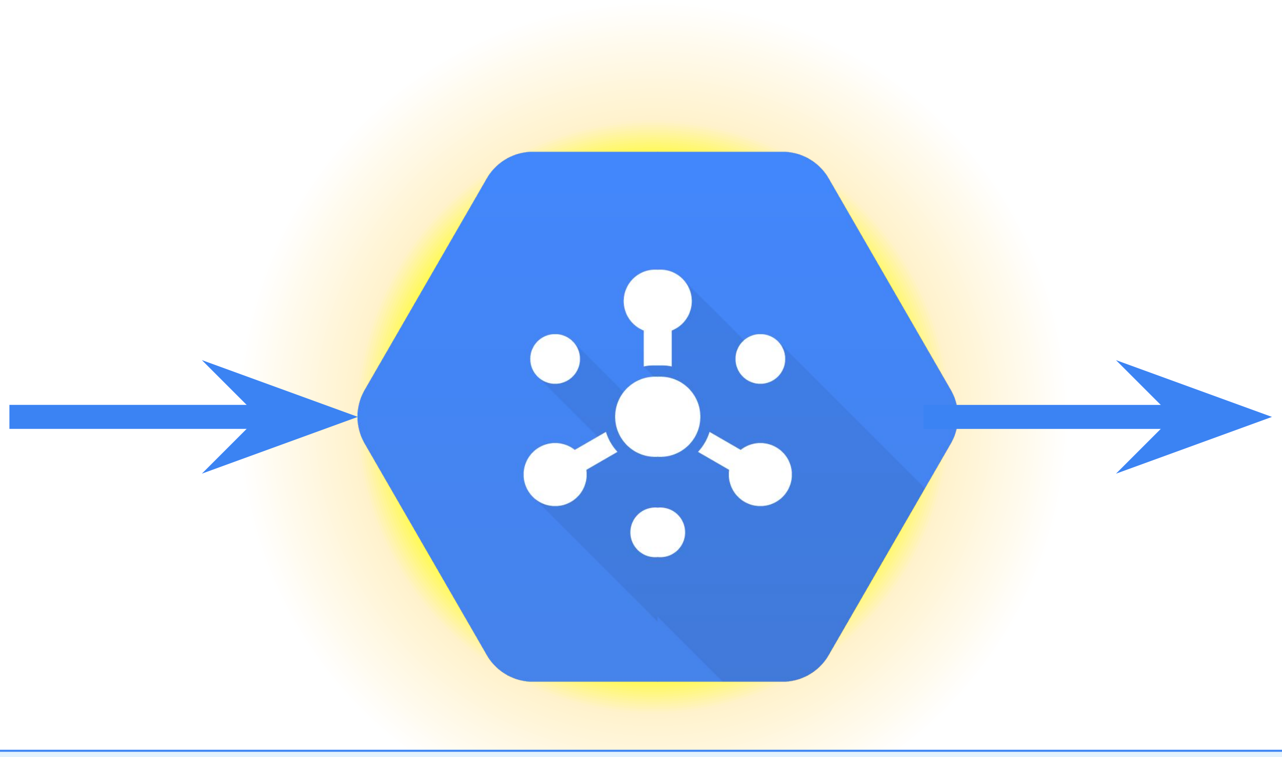
Qualities that Cloud Pub/Sub contribute to Data Engineering solutions:

Availability
Durability
Scalability

100s of milliseconds

Google Cloud

Cloud Pub/Sub



Qualities that Cloud Pub/Sub contribute to Data Engineering solutions:

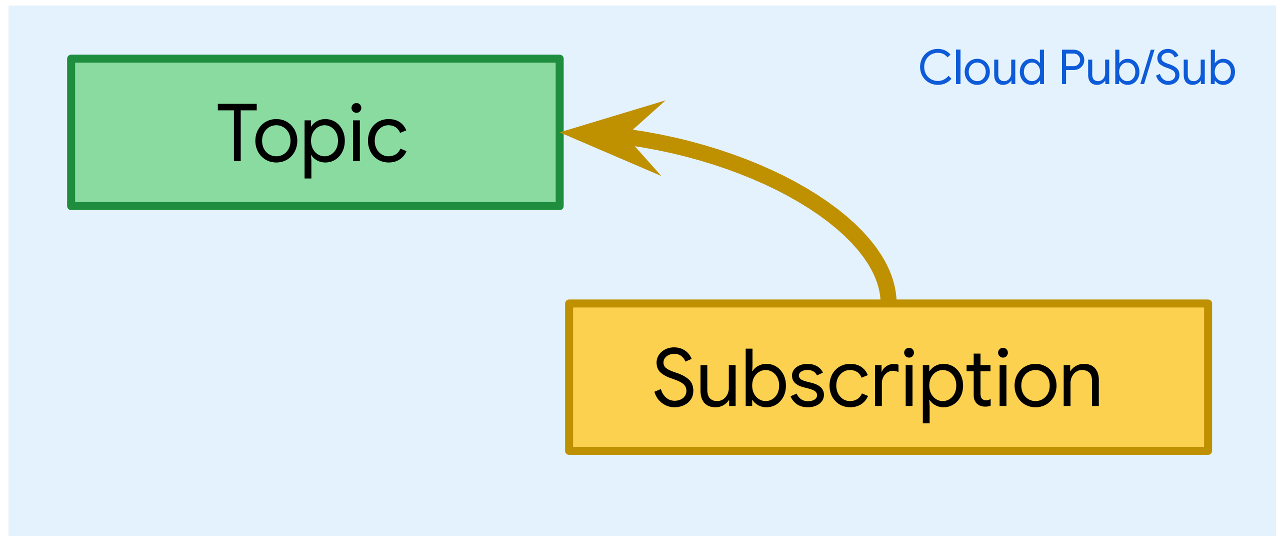
Availability
Durability
Scalability

Cloud
Pub/Sub

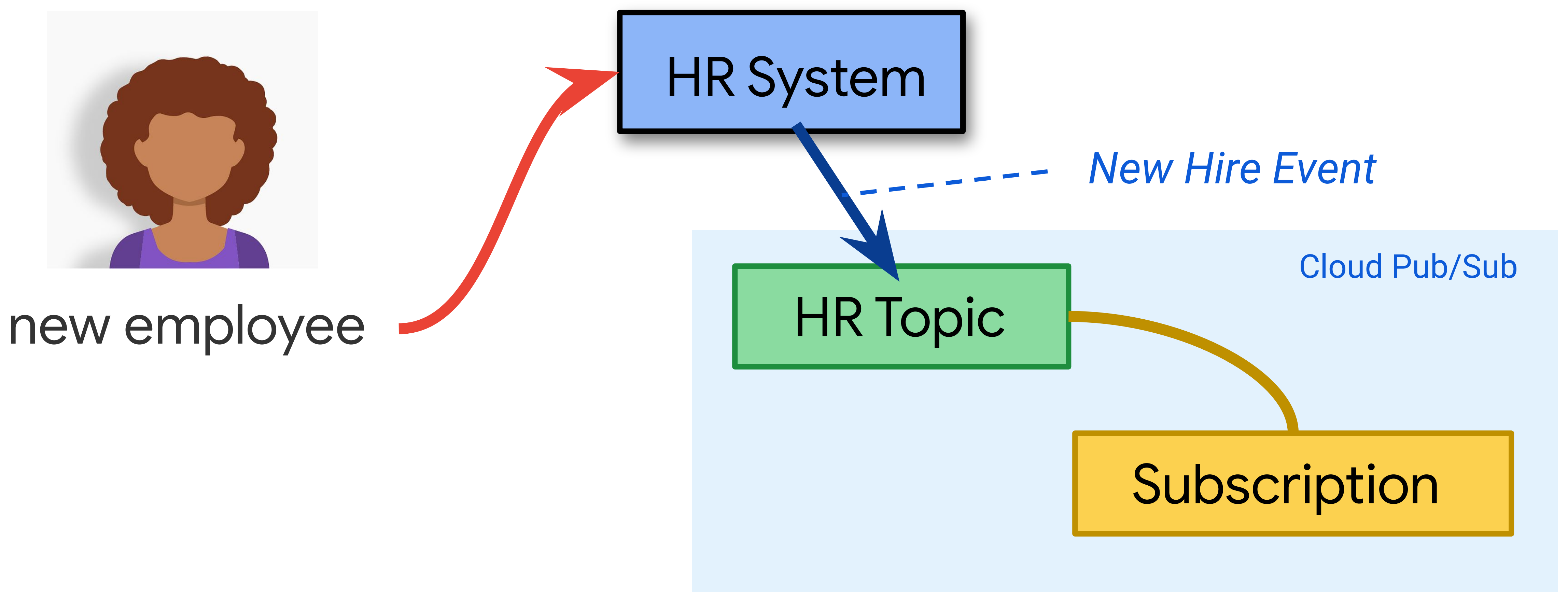
100s of
milliseconds

Google Cloud

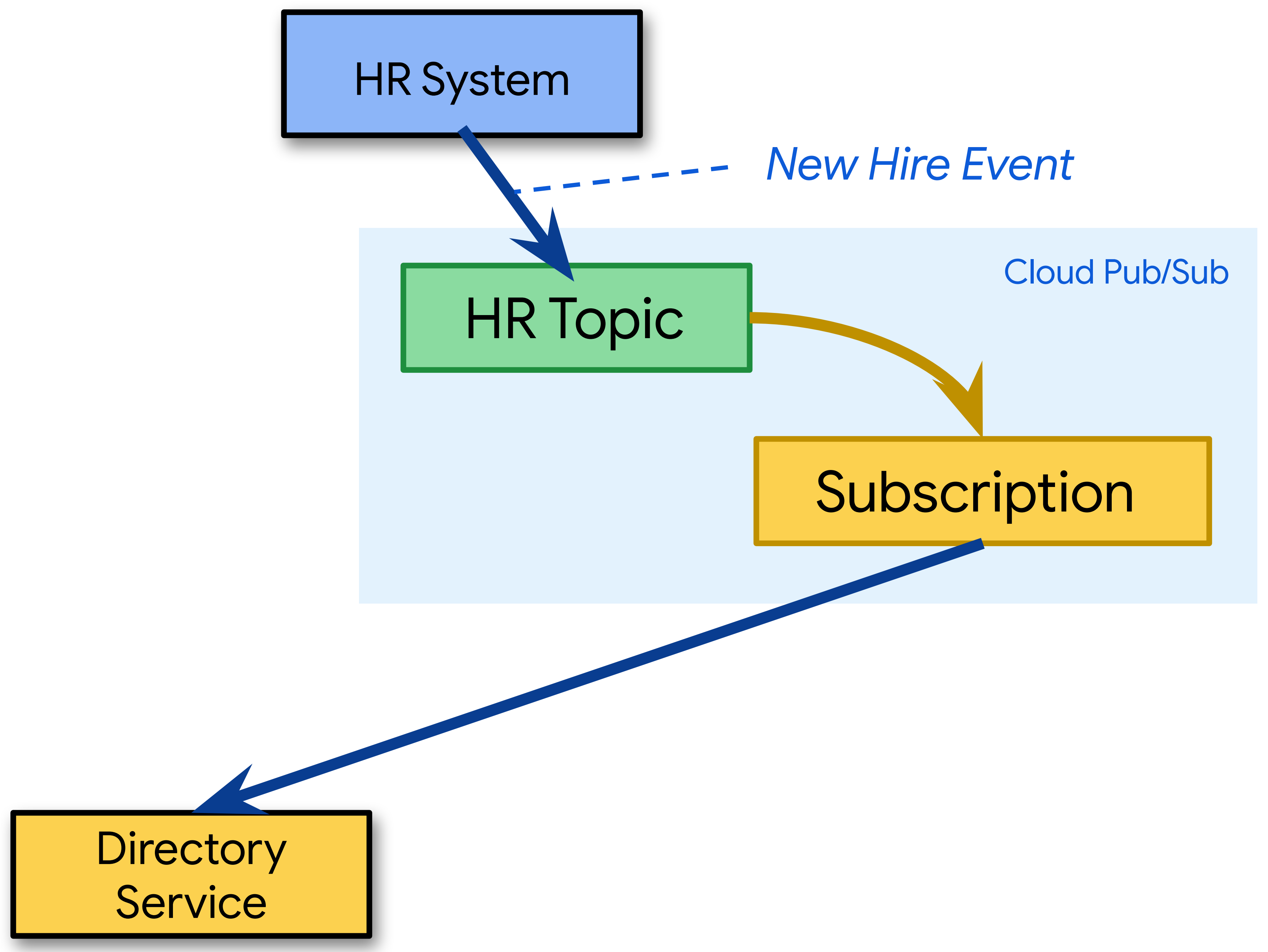
Example of a Cloud Pub/Sub application



A new employees arrives causing a new hire event.

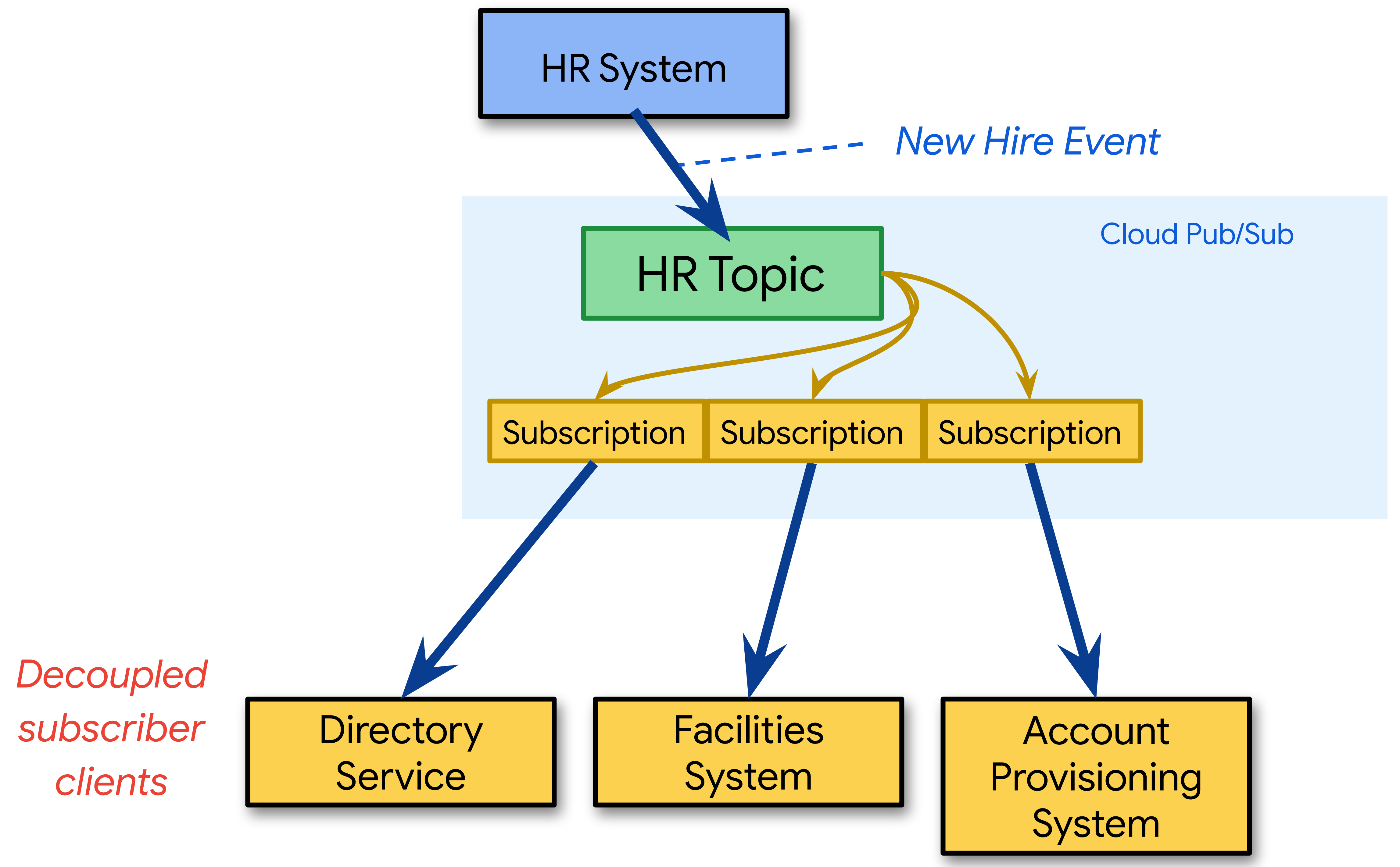


The message is sent from Topic to Subscription

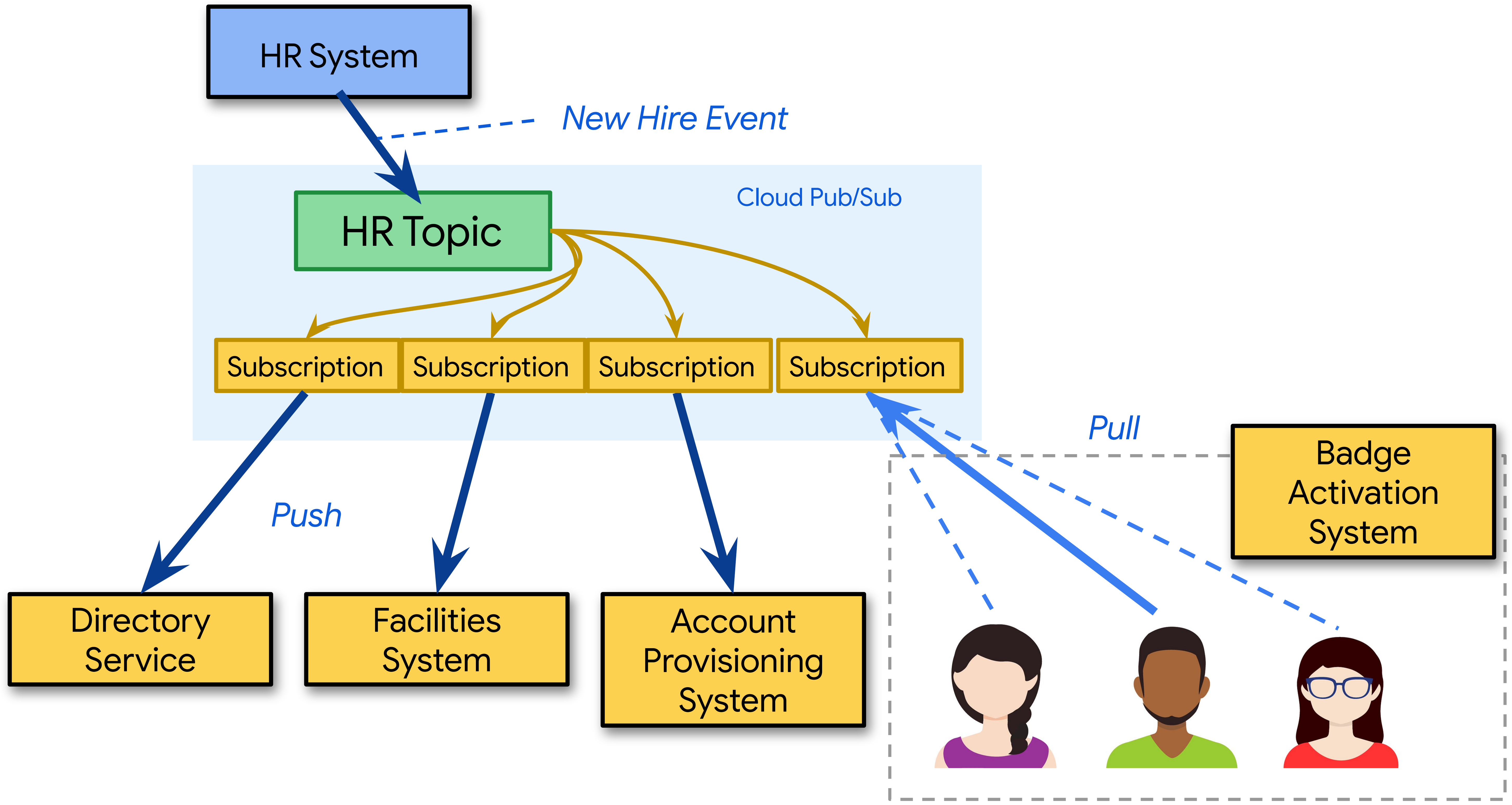


Google Cloud

There can be multiple Subscriptions for each Topic

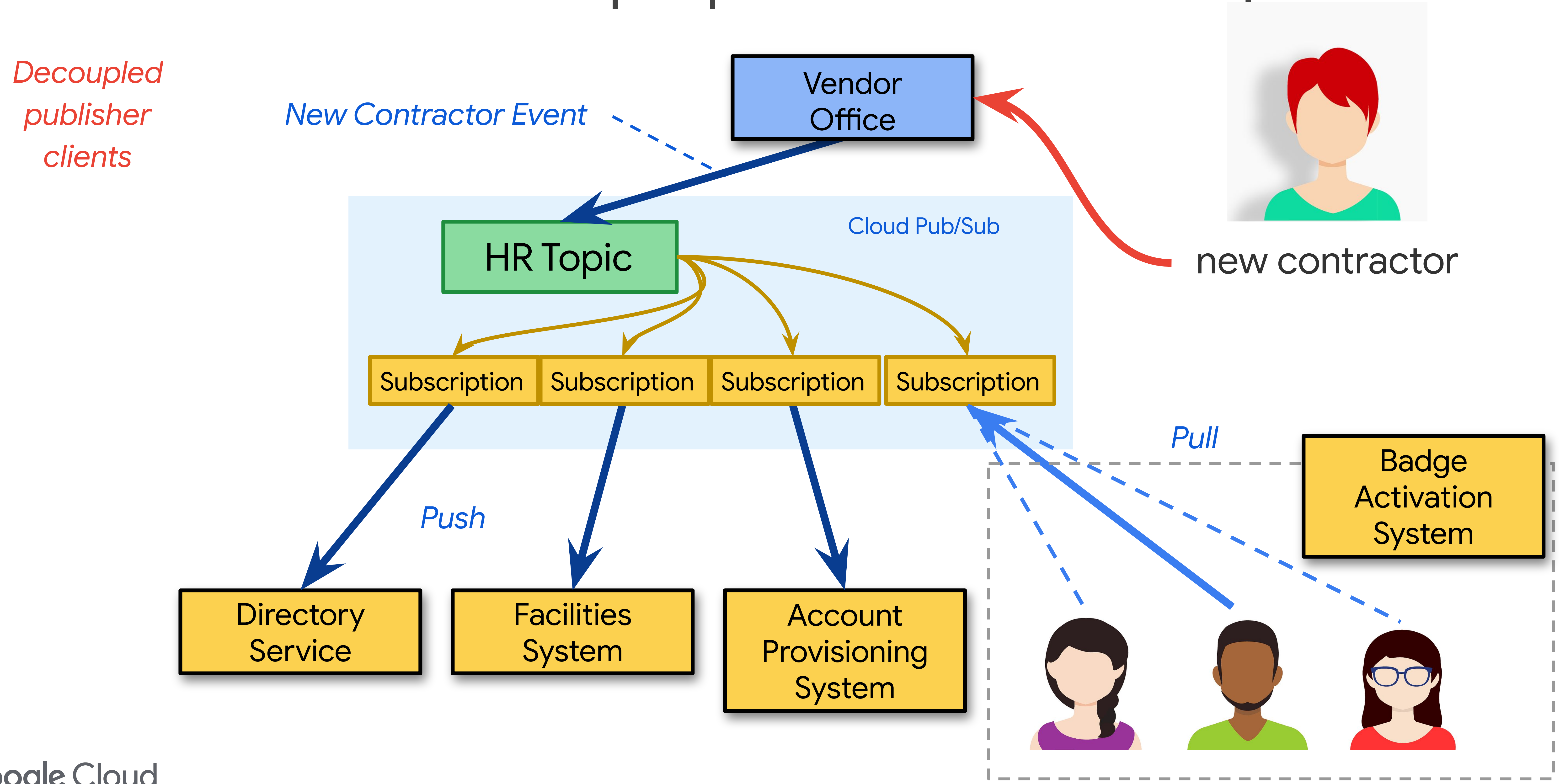


And there can be multiple subscribers per Subscription



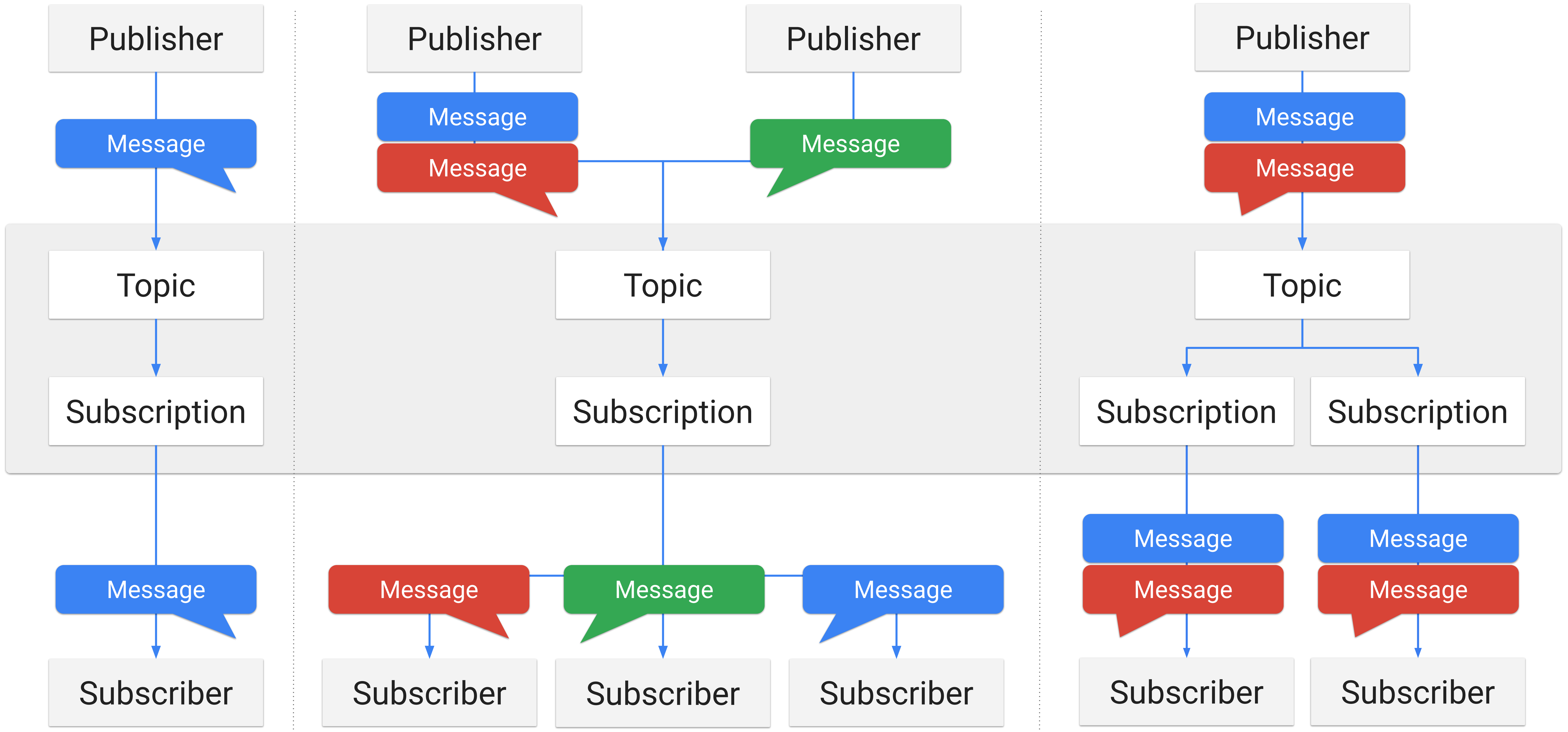
Google Cloud

And there can be multiple publishers to the Topic



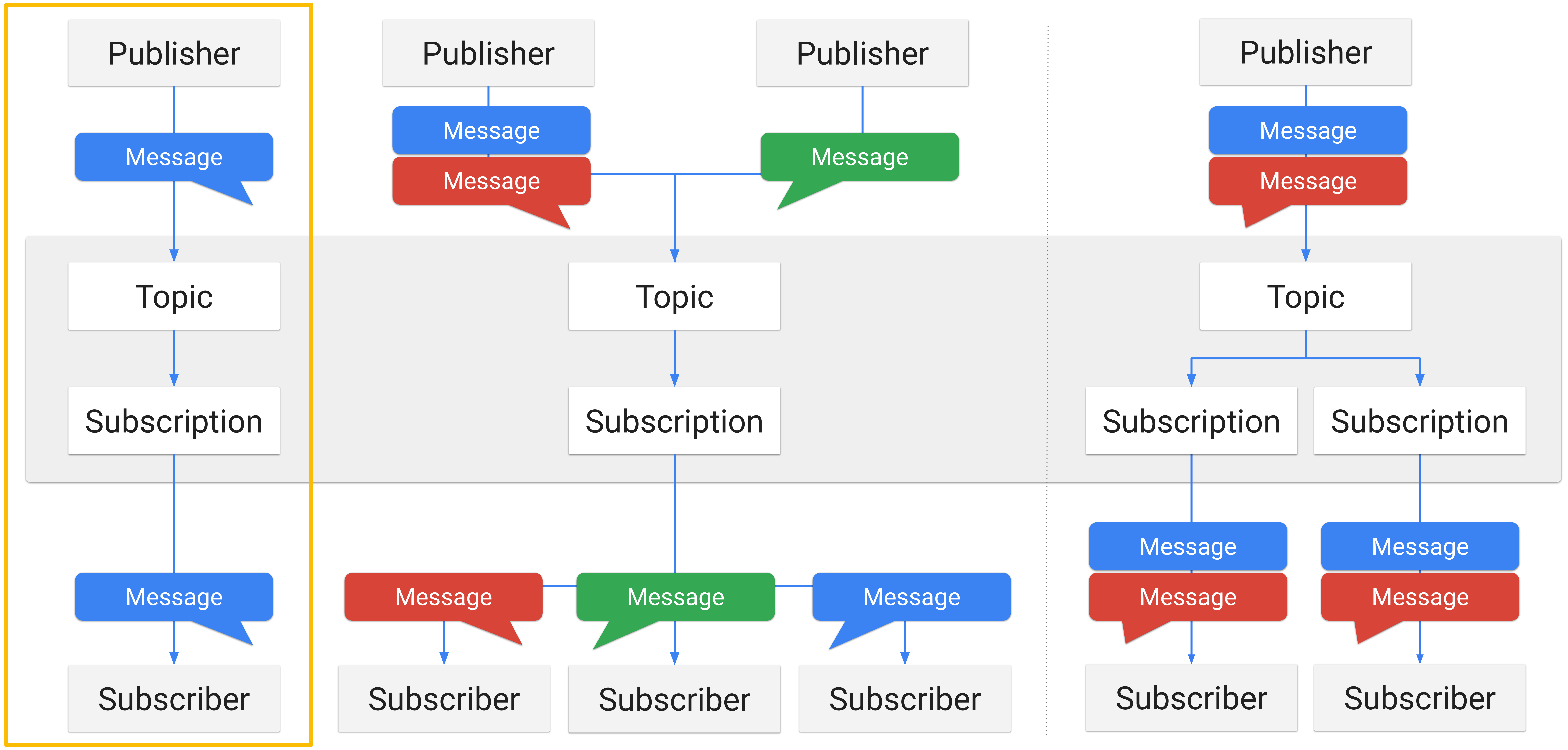
Google Cloud

Publish/Subscribe patterns



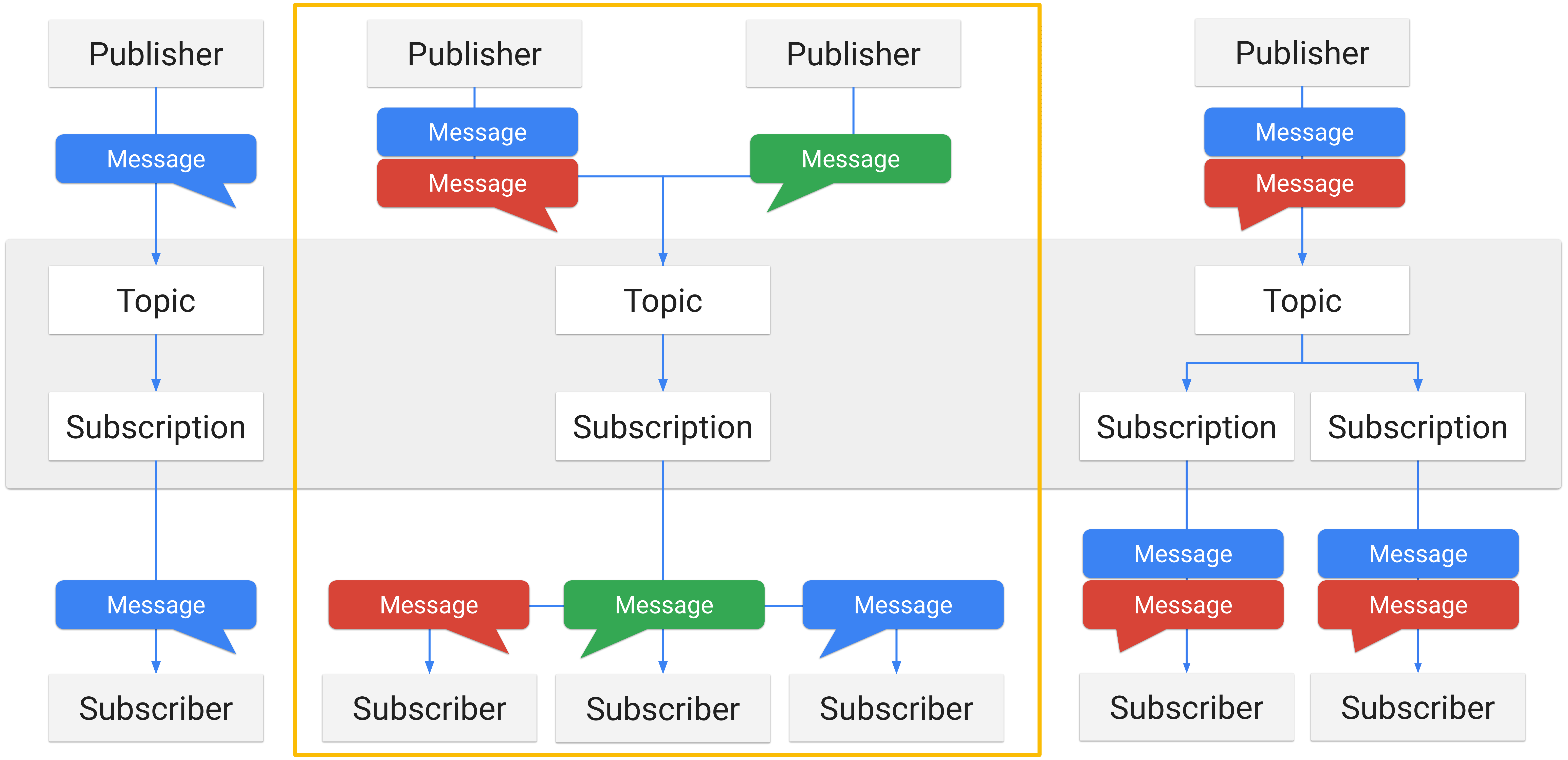
Google Cloud

Publish/Subscribe patterns



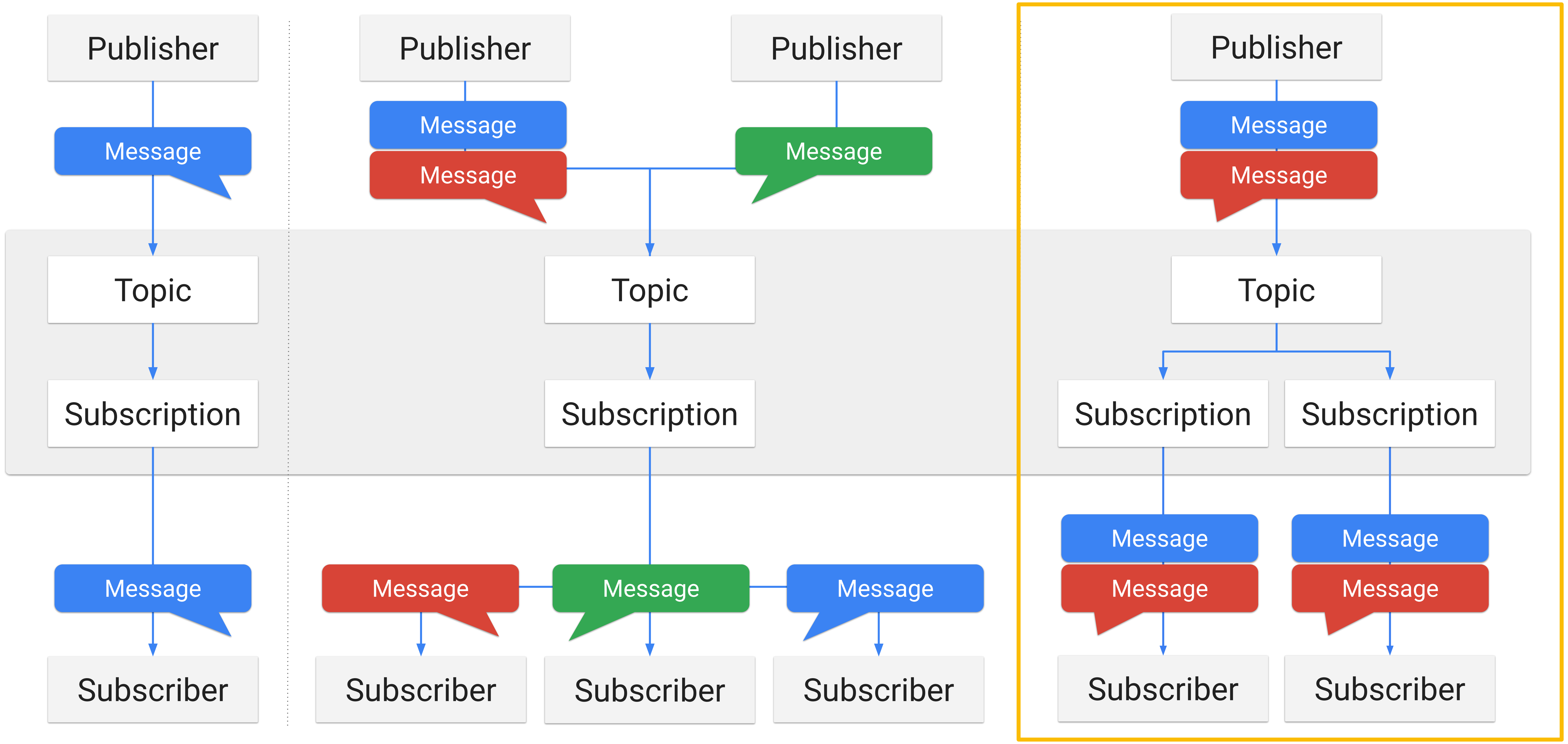
Google Cloud

Publish/Subscribe patterns



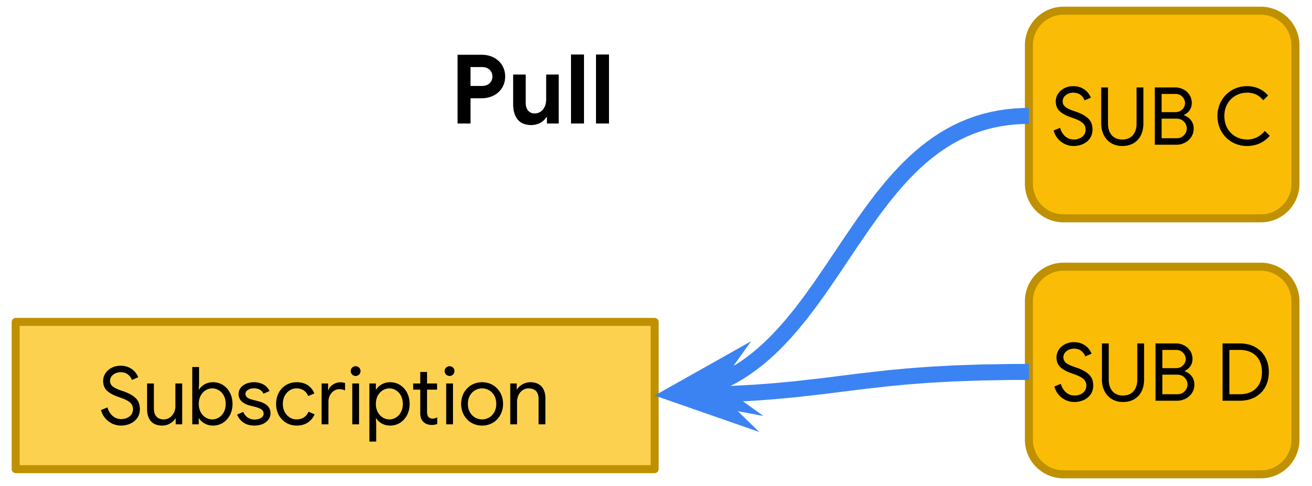
Google Cloud

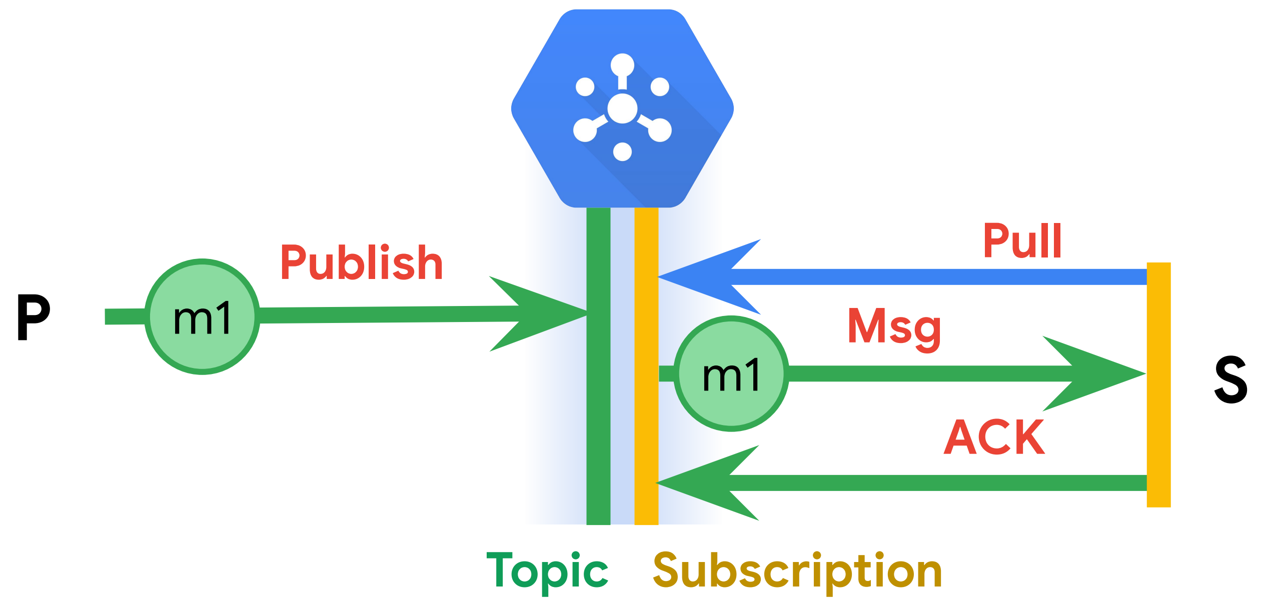
Publish/Subscribe patterns



Google Cloud

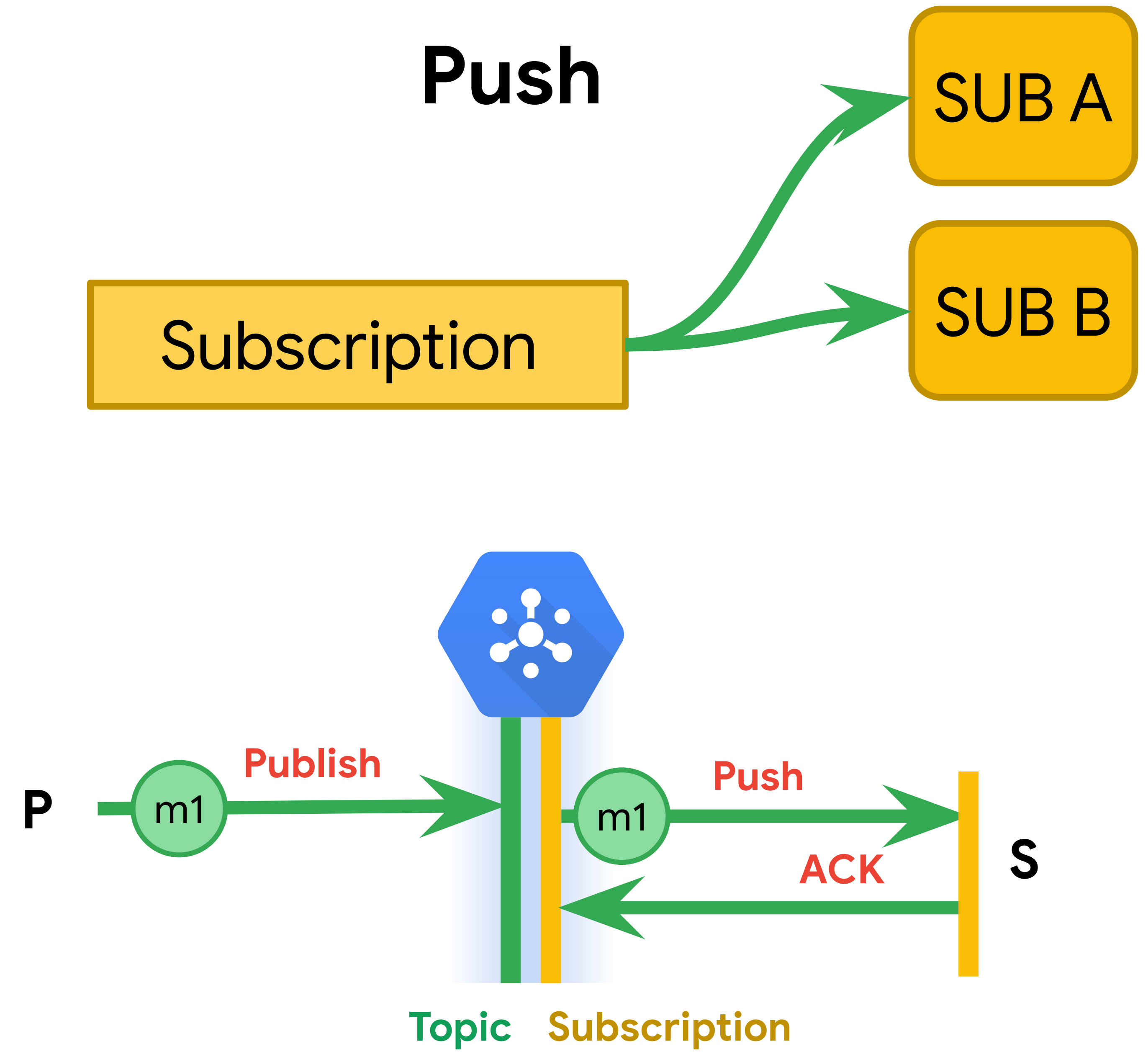
Cloud Pub/Sub provides both Push and Pull delivery



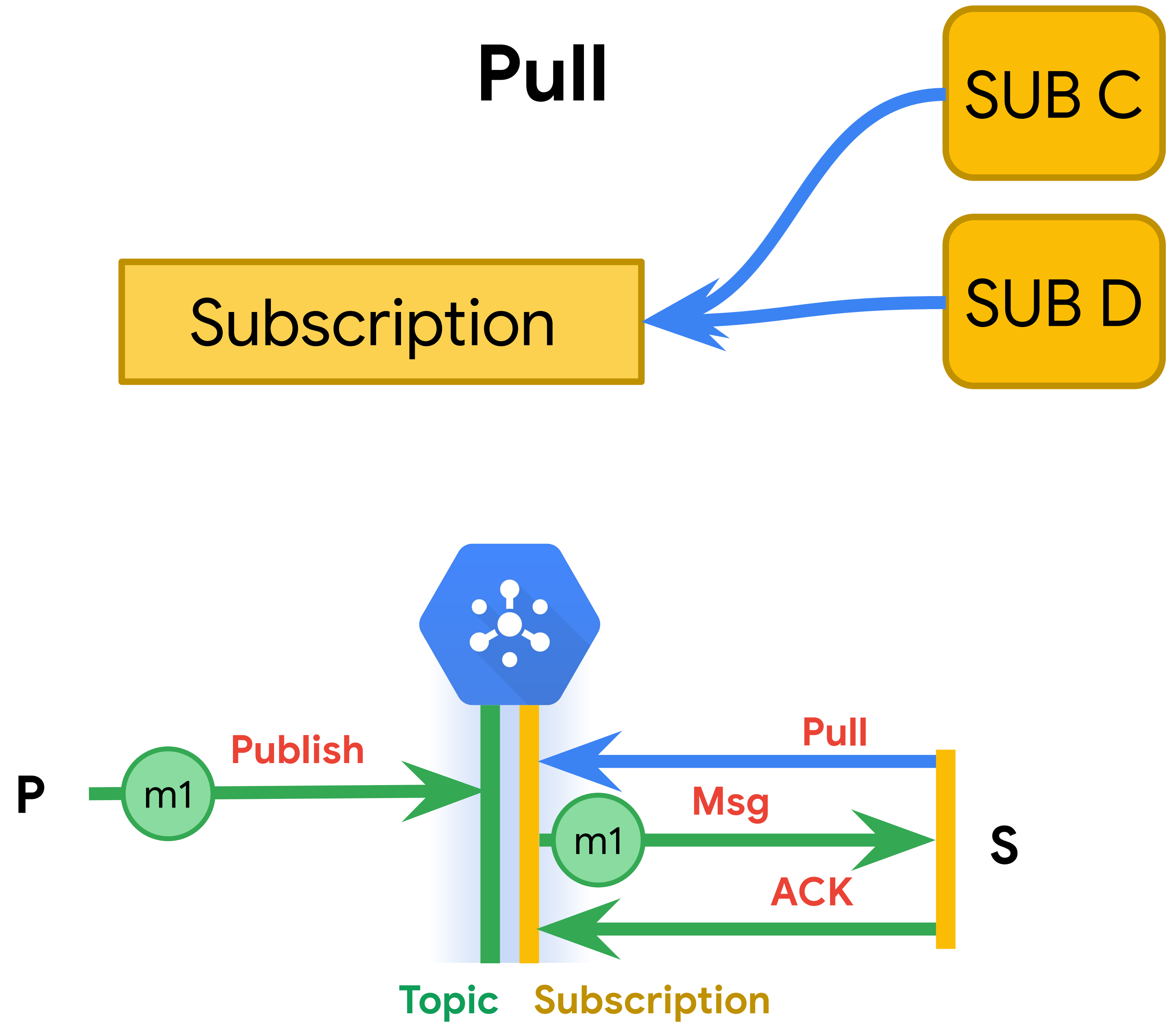


Messages are stored up to 7 days

Cloud Pub/Sub provides both Push and Pull delivery



Acknowledgement used for dynamic rate control

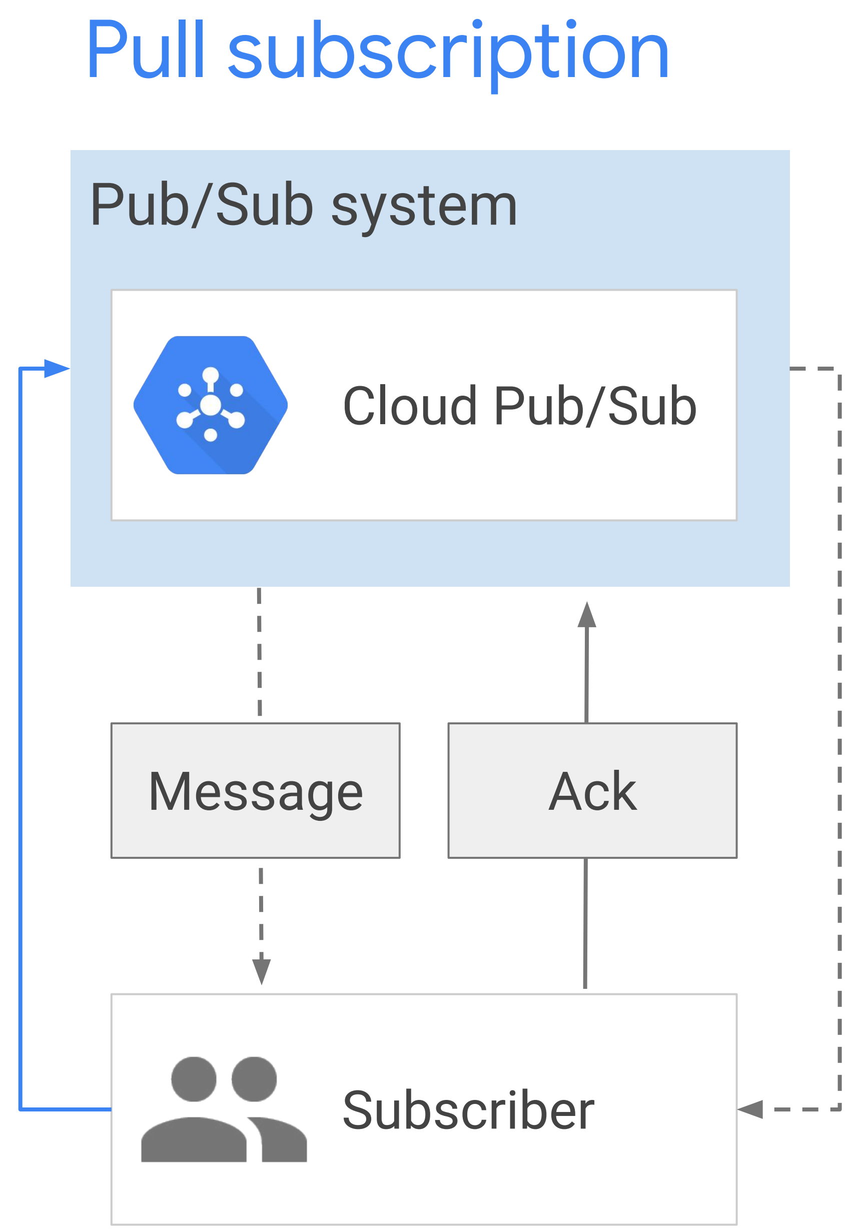


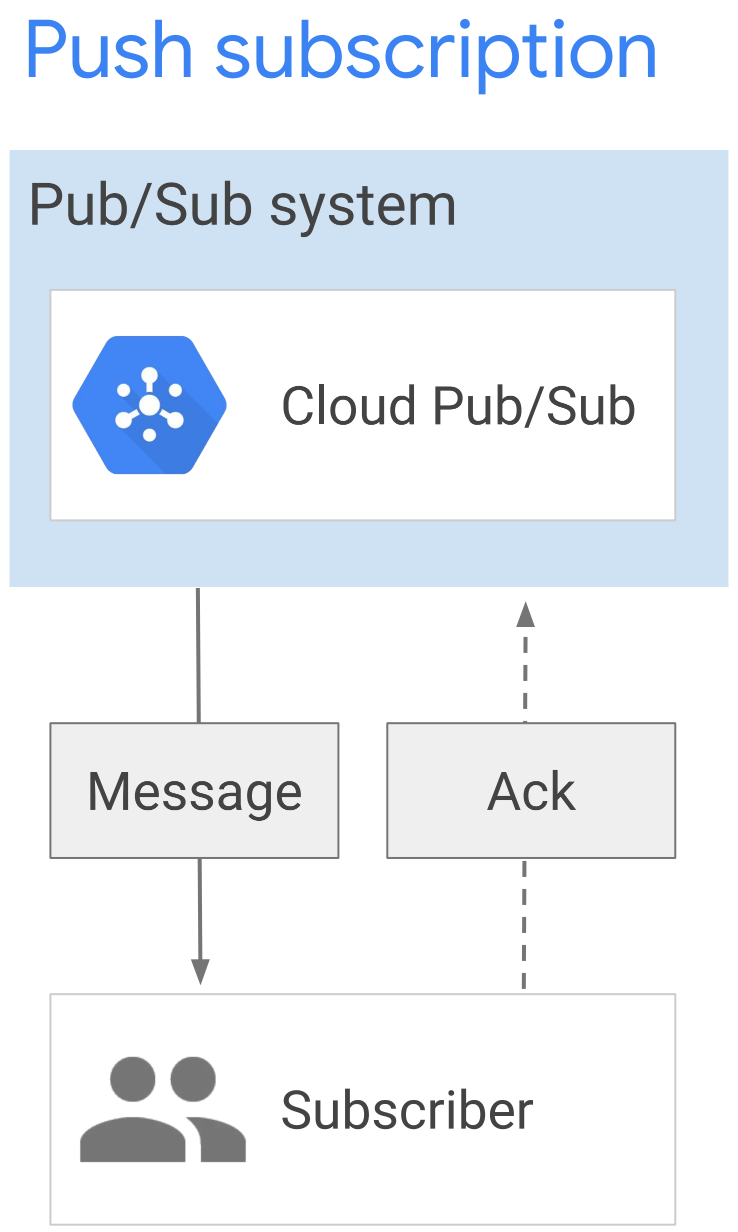
Messages are stored up to 7 days

Google Cloud

At least once delivery guarantee

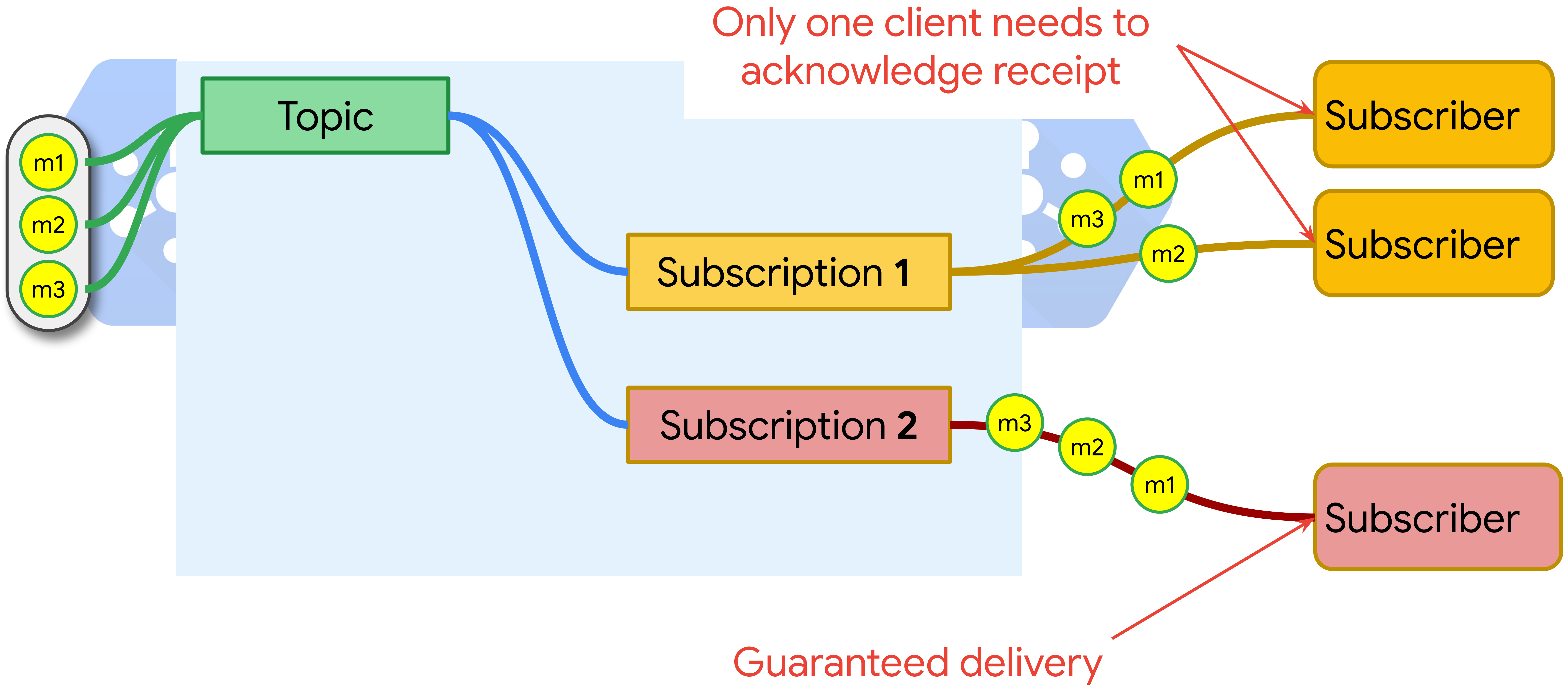
- A subscriber ACKs each message for every subscription
- A message is resent if subscriber takes more than `ackDeadline` to respond
- Messages are stored for up to 7 days
- A subscriber can extend the deadline per message





Google Cloud

Subscribers can work as a team or separately



Google Cloud

Publishing with Cloud Pub/Sub

```
gcloud pubsub topics create sandiego
```

Create topic

Publishing with Cloud Pub/Sub

```
gcloud pubsub topics create sandiego
```

Create topic

```
gcloud pubsub topics publish sandiego --message
"hello"
```

Publish to topic

Publishing with Cloud Pub/Sub

```
gcloud pubsub topics create sandiego
```

Create topic

```
gcloud pubsub topics publish sandiego --message
"hello"
```

Publish to topic

Python

```
import os
from google.cloud import pubsub_v1

publisher = pubsub_v1.PublisherClient()

topic_name = 'projects/{project_id}/topics/{topic}'.format(
    project_id=os.getenv('GOOGLE_CLOUD_PROJECT'),
    topic='MY_TOPIC_NAME',
)

publisher.create_topic(topic_name)
publisher.publish(topic_name, b'My first message!', author='dylan')
```

Create a client

Set topic name

Message

Send attribute

Google Cloud

Publishing with Cloud Pub/Sub

```
gcloud pubsub topics create sandiego
```

Create topic

```
gcloud pubsub topics publish sandiego --message
"hello"
```

Publish to topic

Python

```
import os
from google.cloud import pubsub_v1

publisher = pubsub_v1.PublisherClient()

topic_name = 'projects/{project_id}/topics/{topic}'.format(
    project_id=os.getenv('GOOGLE_CLOUD_PROJECT'),
    topic='MY_TOPIC_NAME',
)

publisher.create_topic(topic_name)
publisher.publish(topic_name, b'My first message!', author='dylan')
```

Create a client

Set topic name

Message

Send attribute

Google Cloud

Publishing with Cloud Pub/Sub

```
gcloud pubsub topics create sandiego
```

Create topic

```
gcloud pubsub topics publish sandiego --message
"hello"
```

Publish to topic

Python

```
import os
from google.cloud import pubsub_v1

publisher = pubsub_v1.PublisherClient()

topic_name = 'projects/{project_id}/topics/{topic}'.format(
    project_id=os.getenv('GOOGLE_CLOUD_PROJECT'),
    topic='MY_TOPIC_NAME',
)

publisher.create_topic(topic_name)
publisher.publish(topic_name, b'My first message!', author='dylan')
```

Create a client

Set topic name

Message

Send attribute

Google Cloud

Subscribing with Cloud Pub/Sub using async pull

Python

```python
import os
from google.cloud import pubsub_v1

subscriber = pubsub_v1.SubscriberClient()
topic_name = 'projects/{project_id}/topics/{topic}'.format(
    project_id=os.getenv('GOOGLE_CLOUD_PROJECT'),
    topic='MY_TOPIC_NAME',
)
subscription_name = 'projects/{project_id}/subscriptions/{sub}'.format(
    project_id=os.getenv('GOOGLE_CLOUD_PROJECT'),
    sub='MY_SUBSCRIPTION_NAME',
)
subscriber.create_subscription(
    name=subscription_name, topic=topic_name)

def callback(message):
    print(message.data)
    message.ack()

future = subscriber.subscribe(subscription_name, callback)
```

Select topic name

Set subscription name

callback when message received

Create a client

Push method
Callback function

Google Cloud

Subscribing with Cloud Pub/Sub using async pull

Python

```
import os
from google.cloud import pubsub_v1

subscriber = pubsub_v1.SubscriberClient()
topic_name = 'projects/{project_id}/topics/{topic}'.format(
    project_id=os.getenv('GOOGLE_CLOUD_PROJECT'),
    topic='MY_TOPIC_NAME',
)
subscription_name = 'projects/{project_id}/subscriptions/{sub}'.format(
    project_id=os.getenv('GOOGLE_CLOUD_PROJECT'),
    sub='MY_SUBSCRIPTION_NAME',
)
subscriber.create_subscription(
    name=subscription_name, topic=topic_name)

def callback(message):
    print(message.data)
    message.ack()

future = subscriber.subscribe(subscription_name, callback)
```

Select topic name

Set subscription name

callback when message received

Create a client

Push method
Callback function

Google Cloud

Subscribing with Cloud Pub/Sub using async pull

Python

```
import os
from google.cloud import pubsub_v1

subscriber = pubsub_v1.SubscriberClient()
topic_name = 'projects/{project_id}/topics/{topic}'.format(
    project_id=os.getenv('GOOGLE_CLOUD_PROJECT'),
    topic='MY_TOPIC_NAME',
)
subscription_name = 'projects/{project_id}/subscriptions/{sub}'.format(
    project_id=os.getenv('GOOGLE_CLOUD_PROJECT'),
    sub='MY_SUBSCRIPTION_NAME',
)
subscriber.create_subscription(
    name=subscription_name, topic=topic_name)

def callback(message):
    print(message.data)
    message.ack()

future = subscriber.subscribe(subscription_name, callback)
```

Select topic name

Set subscription name

callback when message received

Create a client

Push method
Callback function

Google Cloud

Subscribing with Cloud Pub/Sub using synchronous pull

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

Create subscription

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

Pull subscription

Set subscription name

Create a client

```
import time

from google.cloud import pubsub_v1

subscriber = pubsub_v1.SubscriberClient()

subscription_path = subscriber.subscription_path(project_id, subscription_name)
```

`projects/{project_id}/subscriptions/{subscription_name}`

subscription_path format

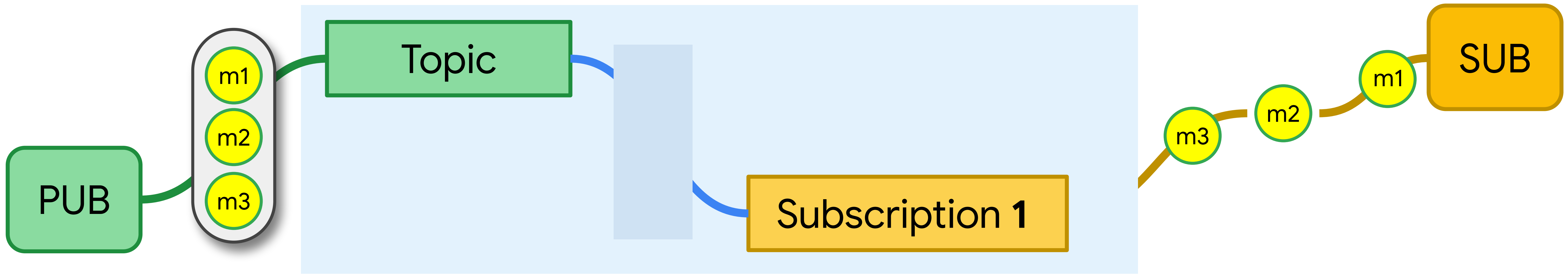
```
 NUM_MESSAGES = 2
 ACK_DEADLINE = 30
 SLEEP_TIME = 10


 # The subscriber pulls a specific number of messages.
 response = subscriber.pull(subscription_path, max_messages=NUM_MESSAGES)
```

Subscriber is non-blocking

Keep the main thread from exiting to allow it to process messages synchronously

Google Cloud

By default, the Publisher batches messages; turn this off if you desire lower latency



Batching messages: throughput versus latency

Google Cloud

Changing the batch settings in Cloud Pub/Sub

Python

```
from google.cloud import pubsub
from google.cloud.pubsub import types

client = pubsub.PublisherClient(
    batch_settings=BatchSettings(max_messages=500),
)
```

Change batch setting

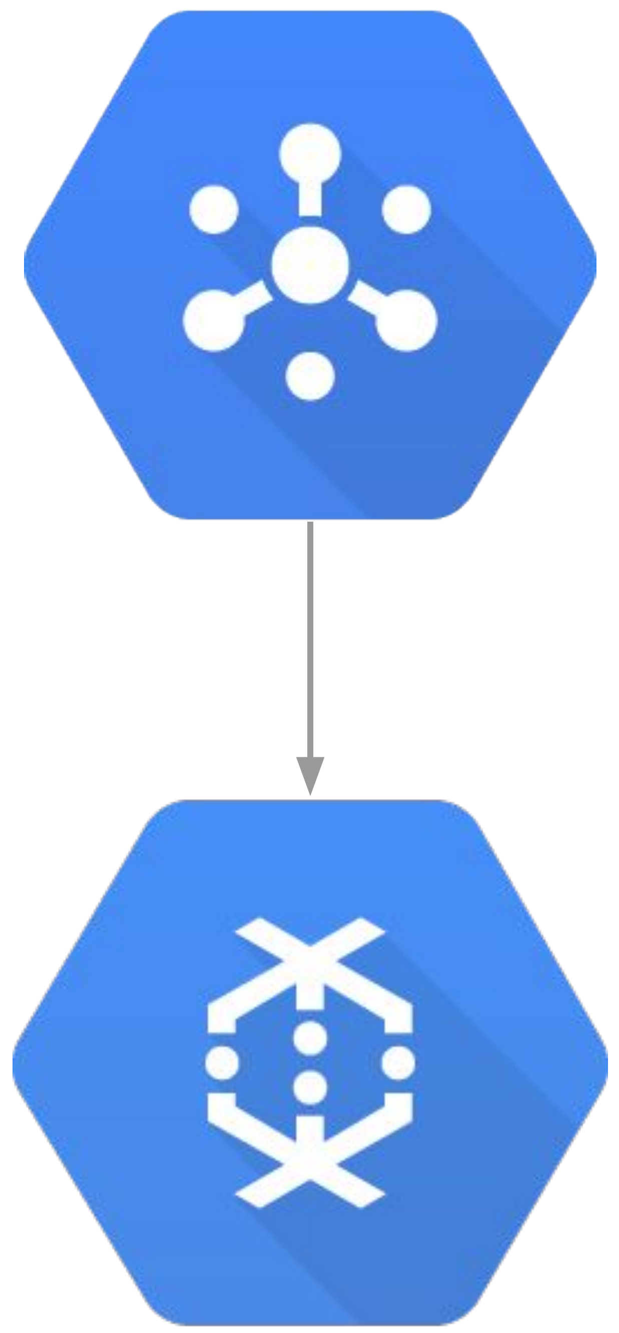
Google Cloud

Pub/Sub: latency, out-of-order, duplication will happen

- Latency -- no guarantees
- Messages can be delivered in any order, especially with large backlog
- Duplication may happen

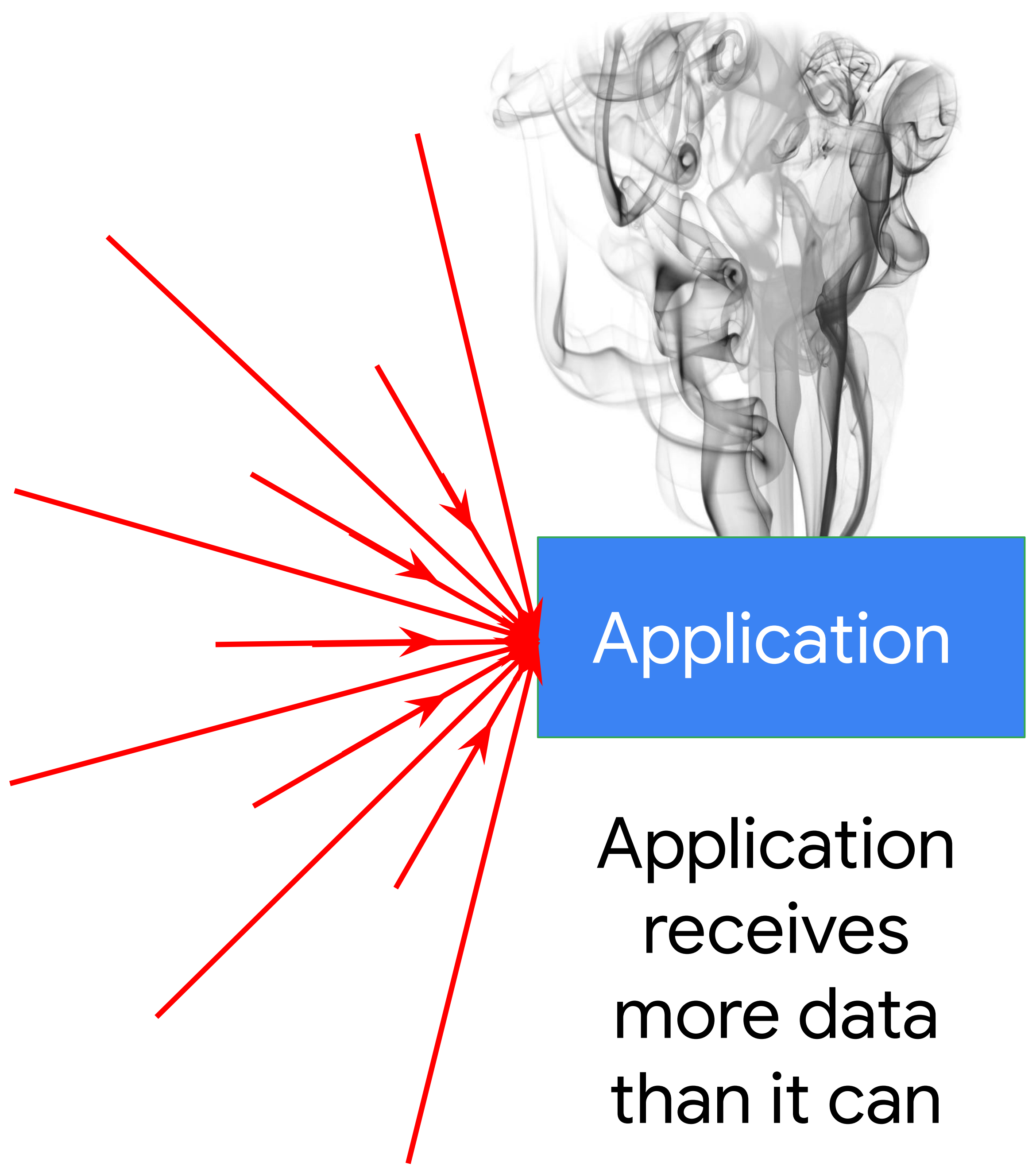
Cloud Pub/sub with Dataflow: Exactly once, ordered processing

- ✓ Cloud Pub/Sub delivers at least once
- ✓ Cloud Dataflow: Deduplicate, order, and window
- ✓ Separation of concerns ⟶ scale



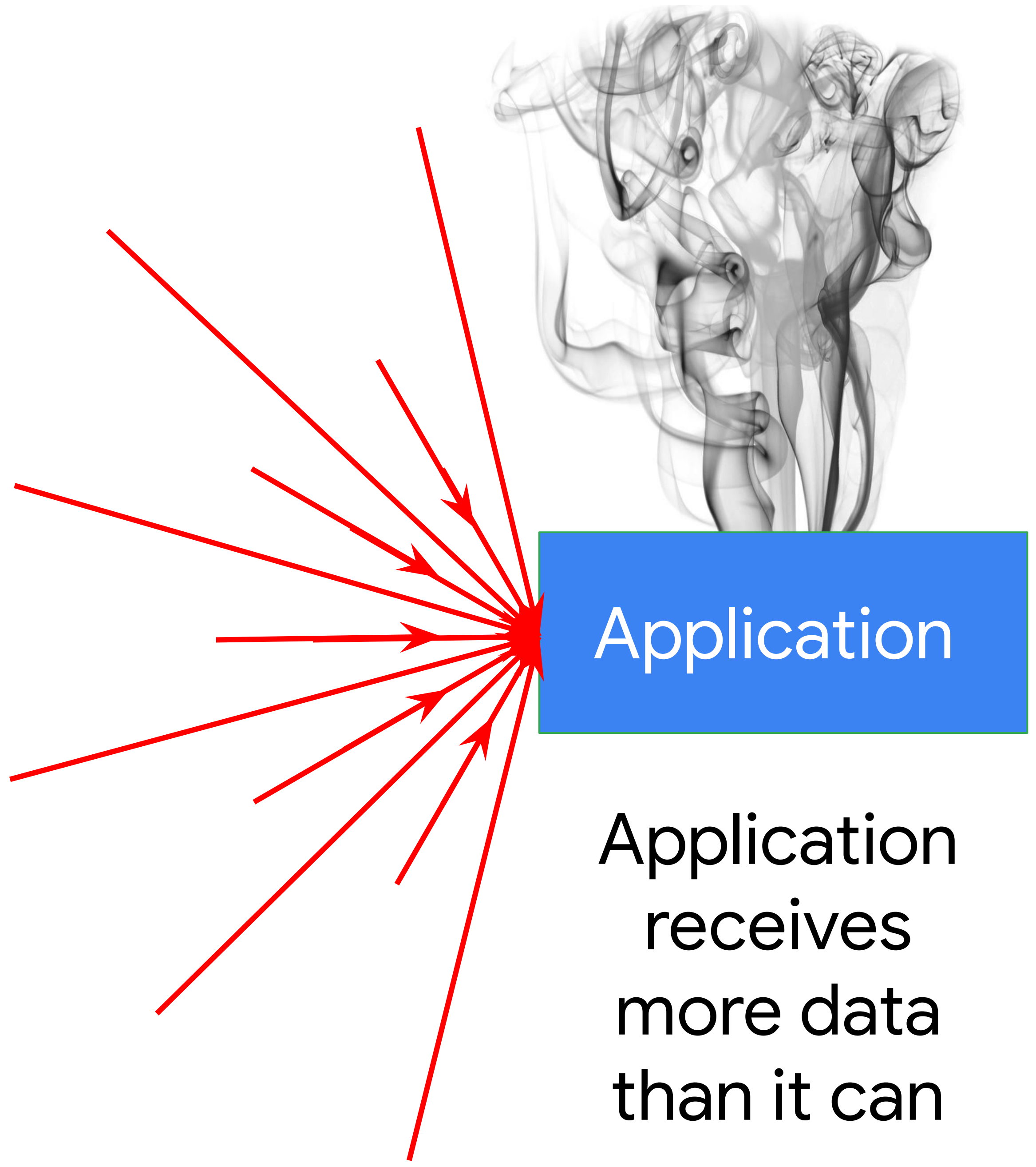
Google Cloud

Use Cloud Pub/Sub for streaming resilience

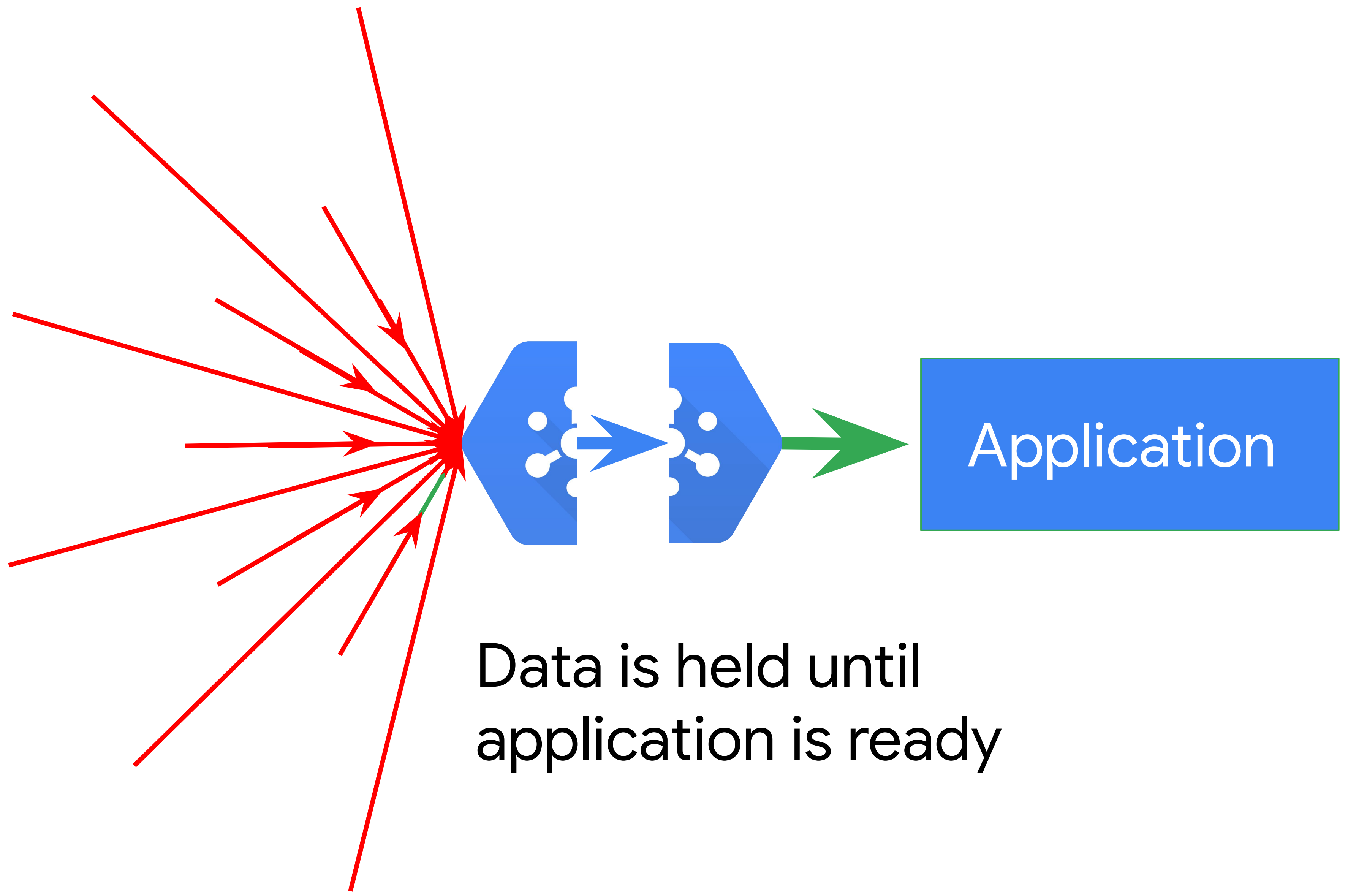


Application receives more data than it can process

Use Cloud Pub/Sub for streaming resilience

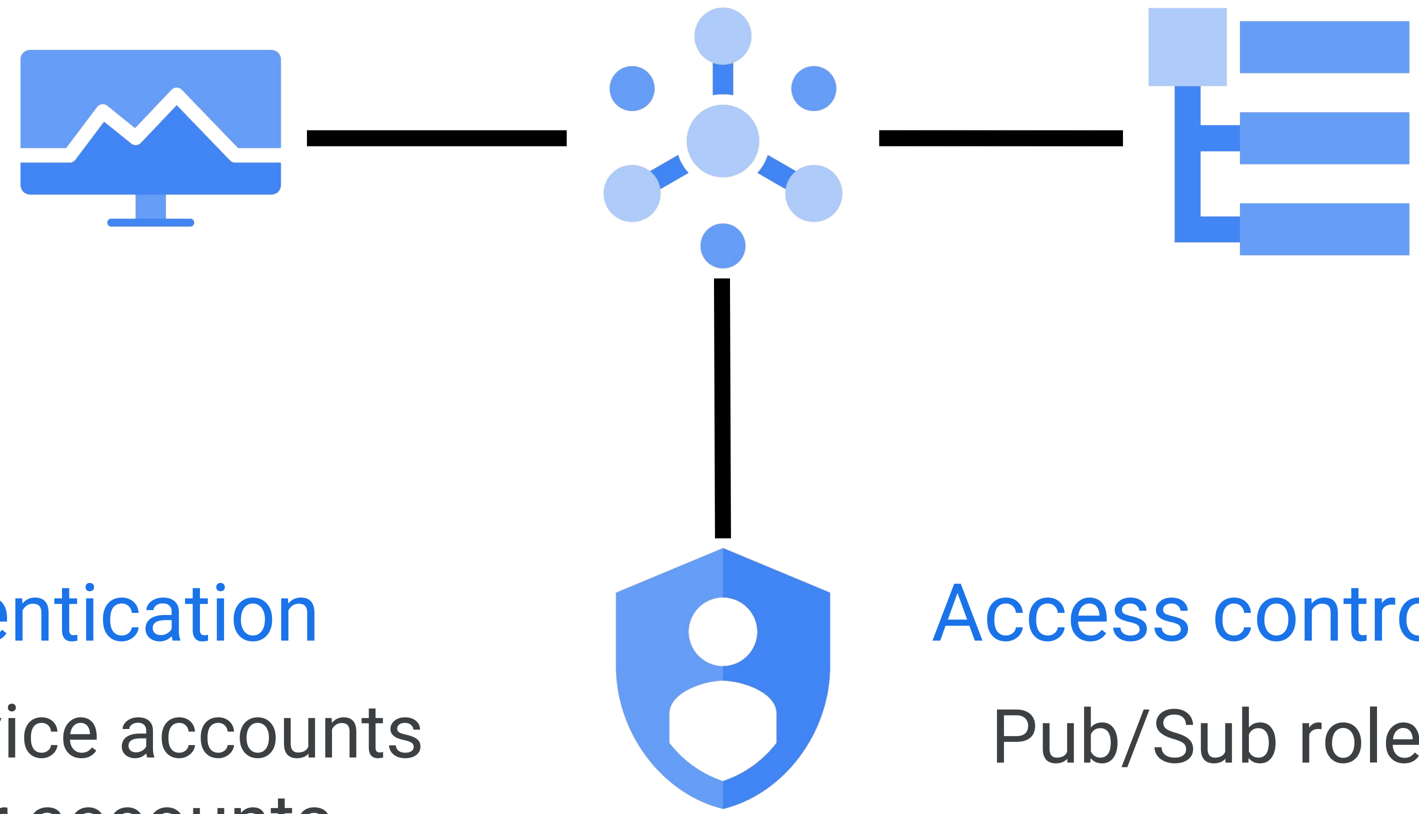


Application receives more data than it can process



Data is held until application is ready

Security, monitoring, and logging for Pub/Sub



Cloud Logging metrics

pubsub_topic
pubsub_subscription

Cloud Audit Logs

Admin = on
Data Access = off

Authentication

Service accounts
User accounts

Access control

Pub/Sub roles per topic

Lab

Publish Streaming Data into Pub/Sub

Lab Objectives

Create a Pub/Sub topic and subscription

Simulate your traffic sensor data into Pub/Sub