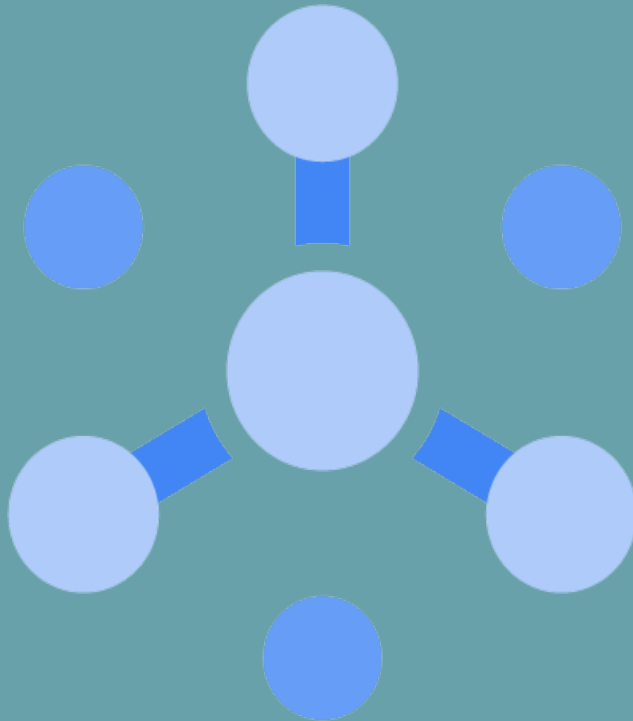


CLOUD PUB/SUB ON GCP: THE WHAT, WHYS AND HOWS

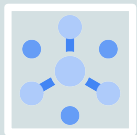
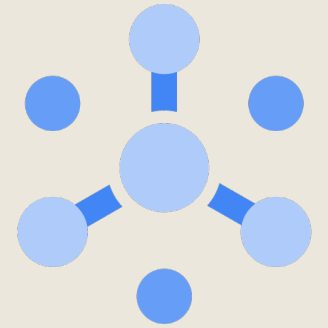
Karen Reeves

Connect Forward

December 2020



Let's talk Pub/Sub



What is Pub/Sub Messaging

Definitions

Use Cases



Cloud Pub/Sub on GCP

Benefits

Publishers

Subscribers

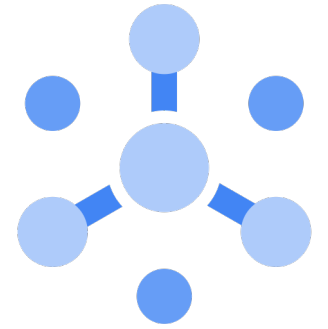
What happens when things go wrong



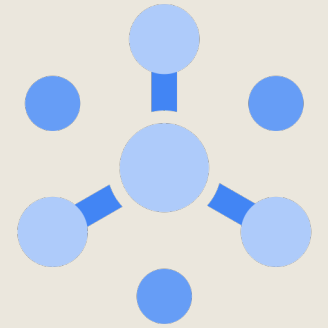
Next Steps

Pub/Sub Messaging

- Asynchronous messaging pattern that decouples applications to increase reliability, performance, and scalability
- Any message sent by a publisher is immediately received by any subscribers to the topic.
- Used for Event driven microservice or serverless architectures



Pub/Sub Terms



MESSAGE

THE UNIT OF DATA BEING SENT BY THE PUBLISHER



TOPIC

AN INTERMEDIARY CHANNEL THAT THE PUBLISHER POSTS TO. THE TOPIC HAS A LIST OF SUBSCRIPTIONS THAT IT PUBLISHES MESSAGES TO.



PUBLISHER

AN APPLICATION THAT SENDS MESSAGES TO TOPIC



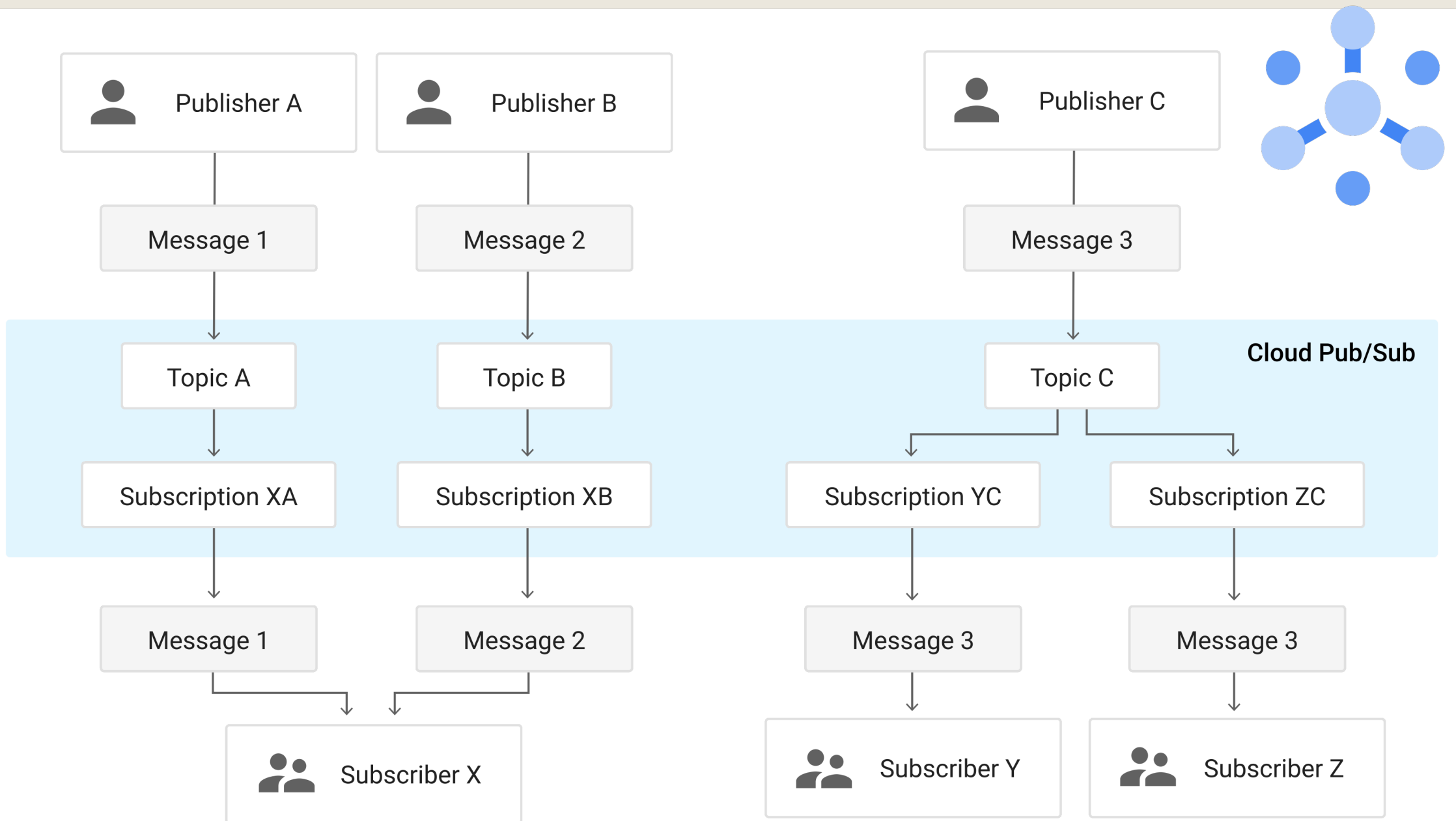
SUBSCRIPTION

A QUEUE FROM EACH TOPIC. A TOPIC MAY HAVE MULTIPLE SUBSCRIPTIONS AND EVERY MESSAGE WILL BE SENT SIMULTANEOUSLY TO ALL SUBSCRIPTIONS.



SUBSCRIBER - AN

APPLICATION THAT SUBSCRIBES TO A TOPIC VIA A SUBSCRIPTION TO RECEIVE MESSAGES PUBLISHED BY ANOTHER APPLICATION





Balancing workloads in network clusters



Implementing asynchronous workflows



Distributing event notifications



Refreshing distributed caches



Logging to multiple systems



Data streaming from various processes or devices

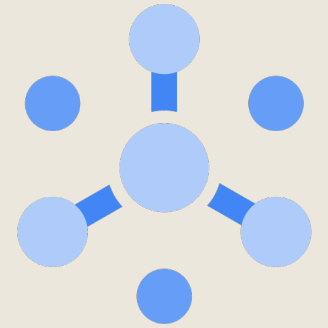


Reliability improvement

Common use cases

<https://cloud.google.com/pubsub/docs/overview>

Google Cloud Pub/Sub



Guaranteed at
least once
delivery

Best effort
ordering

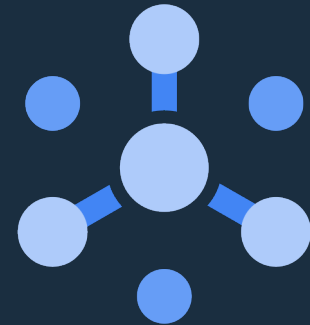
Global, Load
Balanced

Messages
persisted until
they expire or are
acknowledged

Configurable
message
expiration

Setting to persist
acknowledged
messages for
specified period

Built in
monitoring



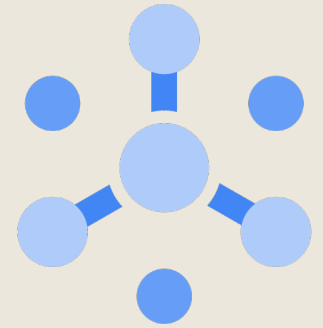
Publishing to a Topic

Publisher creates message

- Content
- Attributes
- Ordering Key

Send request to pub/sub server

Pub/Sub publishes to topic



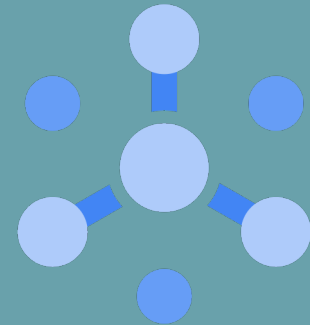
Cloud Pub/Sub Subscriptions

Pull

- Subscriber app sends request to Pub/Sub
- Pub/Sub sends message
- Subscriber acknowledges message
- Un-acknowledged messages will be resent until the configured acknowledge expiration is reached
- Pub/Sub removes message from queue when acknowledged or expired
- Allows flow control
- Good when throughput critical and large number of messages
- Good when Public HTTPS endpoint with non-self signed cert not feasible

Push

- Pub/Sub sends via webhook
- Subscriber responds with a HTTP response code.
- Any Non-Success response code will cause the message to be resent
- If Pub/Sub gets a success message it removes the message from the queue
- Good in environments where Google Credentials are not feasible to set up



WHAT
HAPPENS
WHEN
THINGS GO
WRONG?

Like...

Message that
can't be
processed for
some reason

Issue in the
application so it
isn't processing
messages

Messages sent
that don't need
to be processed

Seek



Seek to Timestamp

*Automatically Acknowledges all messages
before timestamp set*

*Unacknowledges all messages after the
timestamp*

*Requires setting to keep acknowledged
messages to be on*



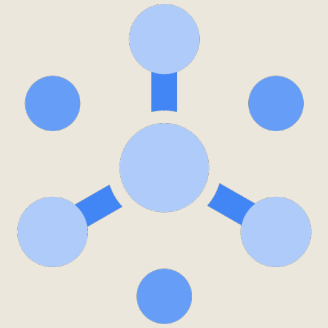
Seek To Snapshot

Create a snapshot

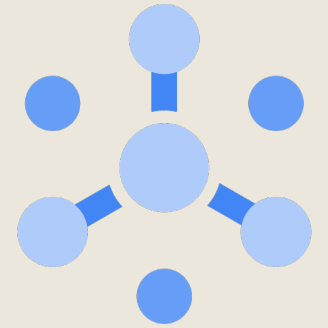
Will resend everything from the snapshot after

*Does not require keeping acknowledged
messages*

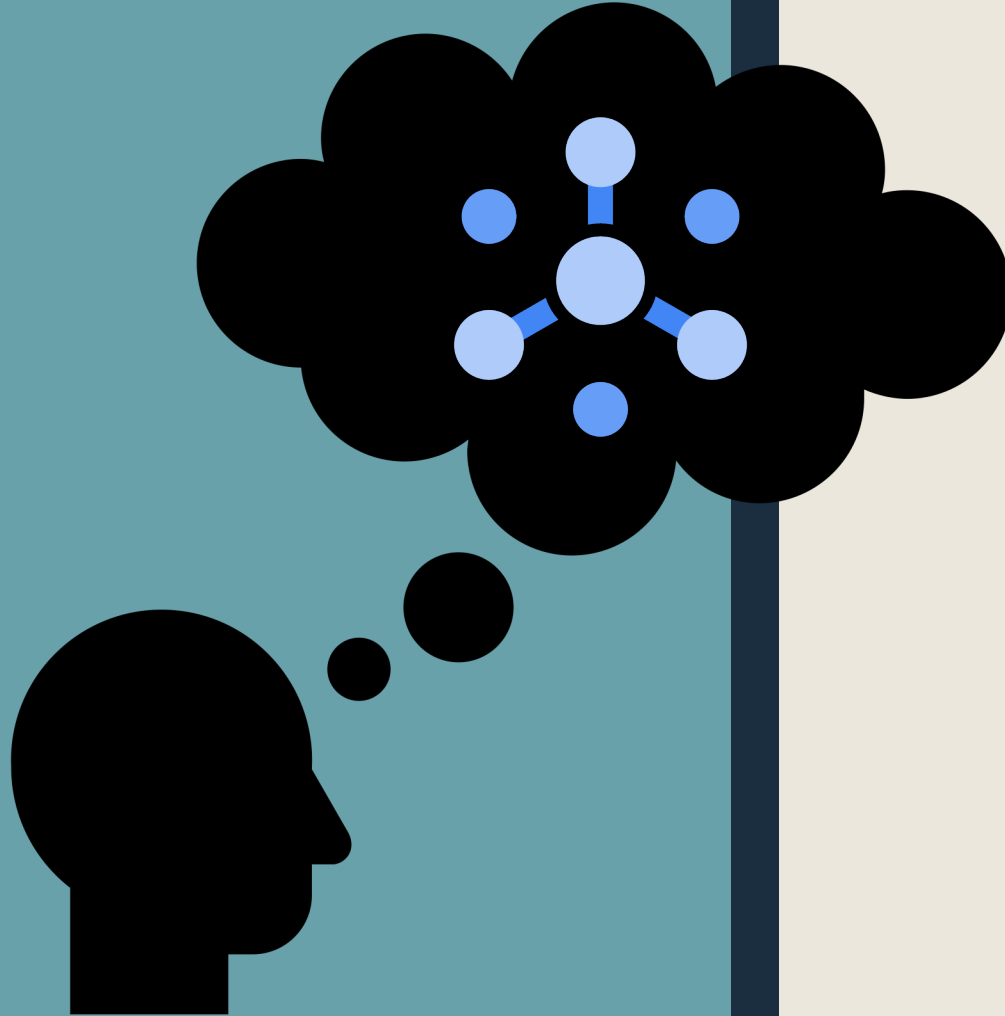
Sometimes used after deployments



Dead Letter Queue



- Newer feature of pub/sub
- Configure delivery retry a message for a set number of times
- If message not acknowledged after retries sent to Dead Letter Queue
- Message removed from Subscription
- Stops using resources for messages that can't be processed
- Allows processing by another method



What Now?

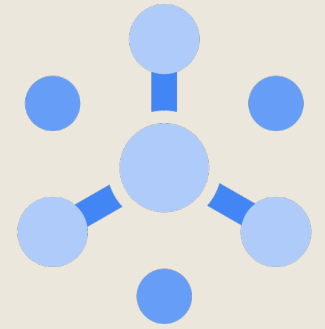
Check out Google's Docs

<https://cloud.google.com/pubsub/docs/overview>

<https://cloud.google.com/pubsub/architecture>

Create a GCP account and play

Pluralsight and Qwiklabs have some resources if you have access to them.

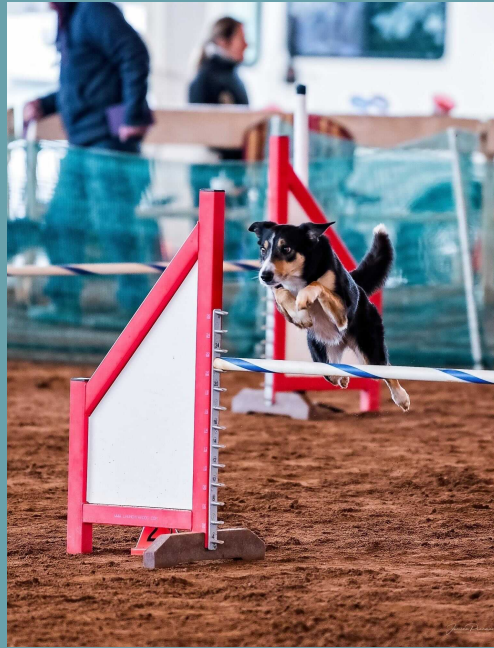


About Me

Karen Reeves
Home Depot
Senior Software Engineer

- [Email: karen_m_reeves@homedepot.com](mailto:karen_m_reeves@homedepot.com)
- [LinkedIn: karenmreeves](#)
- Twitter: @quiltndogs
- Slide Deck:
<https://kmr0018.github.io/ConnectForward-PubSubOnGCP/>

Karen is a software engineer at Home Depot working on GCP, Java Microservices and React. She has 22 years experience in Java and Web Development. In her spare time she plays dog sports with her dogs and quilts, which explains her twitter and IG handle.



Resources

<https://thenewstack.io/publish-subscribe-introduction-to-scalable-messaging/>

<https://cloud.google.com/pubsub/docs/overview>

<https://cloud.google.com/pubsub/architecture>

<https://aws.amazon.com/pub-sub-messaging/>

<https://blog.stackpath.com/pub-sub/>