Real Time Messaging with Pub/Sub

Target Complete Date	@December 2, 2022
Status	Completed
Reviewed	To review
Time to complete (Hours)	1.5
Type	Cloud Guru

Pub/Sub concepts

Message Bus

💡 A common communication platform which can be used to send and receive messages.

Benefits

- Unified communication platform & protocol
- Communication can be controlled
- Prioritization
- Single interface for communication
    - One network connection is enough

Drawbacks

- Bottleneck / single point of failure

    - If every message is transferred over the message bus then it can be overwhelmed
- If anyone can connect to the message bus, then all messages could be read
- Forced communication protocol

Messaging middleware

💡 The primary value of a messaging middleware or a message bus is that it acts as a messaging layer between components. The services are loosely coupled and resilient. It can be thought of as a shock absorber. The current architecture of messaging middleware we are using is Pub/Sub.

✏️ **Data resilience** is the ability to ensure business continuity in the face of unexpected disruptions.

✏️ **Dependency**: Components relying on the functions of other components to work in a certain order that cannot be guaranteed. If one component crashes then the whole system could collapse.

Cloud Pub/Sub

- Global messaging and event ingestion
- Serverless and fully-managed
- Multiple publisher and subscriber patterns
- At-least-once delivery
- Real-time or batch
- Integrates with Cloud Dataflow

Use cases

- Distributing workloads
    - Each instance can grab a task from it’s subscription
- Asynchronous workflows
- Distributing event notifications
- Distributed logging
- Device data streaming
    - Stream data to be consumed on demand

💡 Acts as the glue that adds logic and joins many services together.

Pub/Sub basics

Pub/Sub patterns

1 to 1 pattern

- Publisher sends messages to topic in pub sub where they are queued
- Subscriber accesses messages in a topic via a subscription

1 to N pattern

- There can be subscriptions on a topic

Many to many

- Similar to 1 to 1 but with several topics

Publishing messages

- Create a message containing your data
    - 10 mb JSON file

- Send a request to the pub sub api
- Specify topic

Receiving messages

- Create a subscription to a topic
- Pull is the default delivery method
    - Messages must be acknowledged
        - If not it will remain at top of the queue
- Push will send messages to an endpoint
    - Must be HTTPS with a valid SSL cert

Integrations

- Client libraries
- Cloud dataflow
- Cloud functions
- Cloud run
- Cloud IoT core

Develop for Pub Sub

- Local pub/sub emulator

Pub/Sub demo

⚙️ Create topic and subscriptions and publish messages and receive them.

Pub/Sub advanced

- Each message is delivered at least once for every subscription

- Undelievered message are deleted after the message retention duration
- Messages published before a subscription is created will not be delivered to that subscription

Subscription lifecycle

- Subscriptions expire after 31 days of inactivity
- New subscriptions with the same name have no relationship to the previous subscription

Standard model limitations

- Acknowledged messages are no longer available to subscribers
- Every message must be processed by a subscription

Seeking

💡 Pub sub can retain acknowledged messages. They can be retained for a maximum of 7 days. You can tell Pub Sub to seek from a specific timestamp, where they will be reverted back to a point in the past.

Snapshots

💡 A snapshot can be useful when deploying new code.

Ordering messages

- Due to the high scalability and high availability pub sub cannot guarantee ordering
- Subscribers might receive messages in a different order than they were published in
- Use timestamps when final order matters
  - Consider alternatives for transactional ordering

Resource locations

- Messages are stored in the nearest region
- Message storage policies allow you to control this
- Can lead to additional egress fees

Monitoring

- Total util in bytes
- Subscription util in bytes
- Undelivered messages belong to a subscription
- the message yet to be retrieve by a subscription
    - If both go up there is an issue with the subscriber
- Messages pending delivery to a push subscription

Access control

- Use service accounts for authorization
- Grant per topic or per subscription permissions
- Grant limited access to publish or consume messages

> ⚠️ Security principle of least privilege.

Lab: Loosely-Coupled services with Cloud Pub/Sub

> ⚙️ This lab focuses on showing the value of adding a buffer between services on GCP

- [ ] What is the value of loosely coupling services with Pub/Sub

Lab: Stream data through cloud Pub/Sub to BigQuery

- Create simulated data
- Publish data to Cloud pub sub
- Process data through cloud Dataflow
- View data in BigQuery

⚙️ You can stream data with Pub Sub, using DataFlow and pushing to BigQuery

Exam tips

- Pub sub is a good choice where pub sub would be a good fit to decouple components that would normally send data directly to each other.
    - Can act as a shock absorber, receiving data globally and allowing it to be consumed by other components at their own pace.
- Spot where Pub Sub can add event logic to a stack. Can pass events from one system to another.
- Limitations: message data must be < 10 mb. And expires messages
- If Apache Kafka comes up in the exam, pub sub would be a great service.
    - Look into Cloud IoT as another solution
    - Google cloud tasks
-

☐ Look into the smart analytics reference architectures. See how pub sub works with other GCP products and services