## **Pub/Sub- Pubsub is used to transmit data in real time and scale automatically**

I recommend understanding the difference between push and pull and what you need to implement a push solution.

This service is glued with other Cloud components so there were some questions related to Pub/Sub / Dataflow / BigQuery implementation.

* Streaming and how to implement this solution with Dataflow
* Globally Unique Identifier (GUID)
* Handle subscriber code errors
* How to connect Kafka to Pub/Sub
* How to know when your topic is currently not working well. This is mostly related to Stackdriver Monitoring.
* Process of moving from the Apache Kafka to pub/sub workflow
* IAM (Identity and Access Management) controls on different levels, such as the fact that the publisher level has no IAM controls
* Learn how the process of message flow works, such as why delays in sending messages might occur

Google Cloud **Pub/Sub** provides a globally durable message ingestion service. By creating topics for streams or channels, you can enable different components of your application to subscribe to specific streams of data without needing to construct subscriber-specific channels on each device. Cloud Pub/Sub also natively connects to other Cloud Platform services, helping you to connect ingestion, data pipelines, and storage systems. <https://cloud.google.com/pubsub/docs/overview>

Cloud Pub/Sub is a service to ingest event streams at any scale. It is scalable and reliable for stream analytics and event-driven computing systems. So, it is the most reliable Google product for such a scenario.

1. The ability to seek a particular offset in a topic, possibly back to the start of all data ever captured  
2. Support for publish/subscribe semantics on hundreds of topics  
3. Retain per-key ordering  
Which system should you choose?

Kafka has all the above functionality. These are the functionalities which are currently lagging/not available with Pub/Sub.

Cloud Pub/Sub is a service to ingest event streams at any scale. It is scalable and reliable for stream analytics and event-driven computing systems.  
Pub/sub is a good product to de-couple a system’s components, so they communicate with each other asymmetrically. From the scenario shown here, instead of directly calling the API to export required report which puts great loads on the API and hence the timeouts faced by users. Instead, the platform can “publish” messages to a “topic” related to exporting activity log reports sending the required parameters such as user ID and custom settings such as date range and what data to export. The API can be switched to be a “subscriber” which receives the messages sent and processes each message asymmetrically to generate the report, then sends the download link to the user’s mailbox when ready.

Cloud Storage upload events can push Cloud Pub/Sub to trigger a Cloud Function to ingest and process the image.

Cloud Pub/Sub Notifications sends information about changes to objects in your buckets to Cloud Pub/Sub, where the information is added to a Cloud Pub/Sub topic of your choice in the form of messages. For example, you can track objects that are created and deleted in your bucket. Each notification contains information describing both the event that triggered it and the object that changed.  
Cloud Pub/Sub Notifications are the recommended way to track changes to objects in your Cloud Storage buckets because they’re faster, more flexible, easier to set up, and more cost-effective.

Dataflow cannot be configured as Push endpoint with Cloud Pub/Sub. -- only pull mechanism

Cloud Pub/Sub can help handle the streaming data. However, Cloud Pub/Sub **does not handle the ordering, which can be done using Dataflow** and **adding watermarks** to the messages from the source.

How do you assign an order to messages published from different publishers? Either the publishers themselves have to coordinate, or the message delivery service itself has to attach a notion of order to every incoming message. Each message would need to include the ordering information. The order information could be a timestamp (though it has to be a timestamp that all servers get from the same source in order to avoid issues of clock drift), or a sequence number (acquired from a single source with ACID guarantees). Other messaging systems that guarantee ordering of messages require settings that effectively limit the system to multiple publishers sending messages through a single server to a single subscriber.

Typically, Cloud Pub/Sub delivers each message once and in the order in which it was published. However, messages may sometimes be delivered out of order or more than once. In general, accommodating more-than-once delivery requires your subscriber to be idempotent when processing messages. You can achieve exactly once processing of Cloud Pub/Sub message streams using Cloud Dataflow PubsubIO. PubsubIO de-duplicates messages on custom message identifiers or those assigned by Cloud Pub/Sub. You can also achieve ordered processing with Cloud Dataflow by using the standard sorting APIs of the service. Alternatively, to achieve ordering, the publisher of the topic to which you subscribe can include a sequence token in the message.

Cloud Pub/Sub does not maintain message order and Dataflow can be used to order the messages and as well as calculate **average using Sliding Time window.**

*pull mode allows new event data to be pulled for processing on demand when the previous data is processed. Pub/Sub will absorb and retain new events in the interim without losing them.*