I intend to develop a topic based pub/sub message broker like RabbitMQ or Kafka.

Pub/sub, an abbreviation for publish/subscribe, is an asynchronous messaging architecture in which messages are transferred between entities without the sender or the recipient knowing the identity of the other.

There are three main components that make up the pub/sub architecture, publishers, the event bus or broker, and the subscribers. Let's define these.

Publishers

Publishers are nodes that generate messages that are subsequently sent across the system using the event bus/broker.

Event bus/broker

The event bus/broker nodes serve as middlemen, facilitating the flow of messages from publishers to subscribers. Brokers further strengthen the decoupling between system nodes because subscribers deal with the broker rather than the entire system.

Subscribers

Subscribers essentially listen for communication about the topics and categories in which they are interested, doing so without knowing who the senders of these communications are.

Subscribers typically indicate their interest in receiving specific messages, acting as a filtering mechanism. Because of the fine-grained topic control, it's simple to ensure that the various event buses are sending the correct message. We can achieve this type of filtering using either topic-based filtering or content-based filtering.

Topic-based filtering

Topic based filtering requires the messages to be disseminated into logical channels. The subscribers only get messages from logic channels to which they have subscribed.

Pub/sub messaging brokers

A message broker is a computer program module that enables message validation, communication, and routing between applications, systems, and services. Message brokers serve as middlemen between nodes, thus facilitating the exchange of messages between publishers and subscribers.

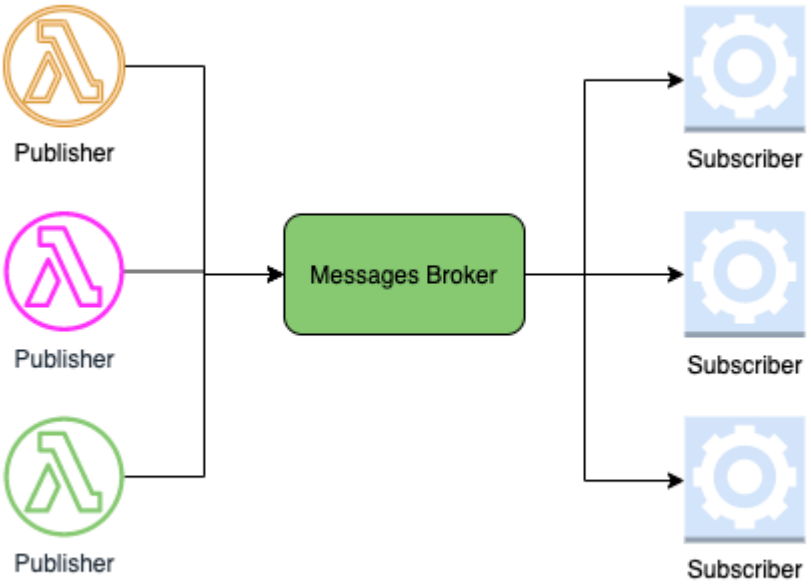
The primary purpose of a broker is to take incoming messages from applications and perform some action on them. Message brokers effectively implement decoupling by minimizing the mutual awareness between nodes.

An exchange is a very simple thing. On one side it receives messages from producers and

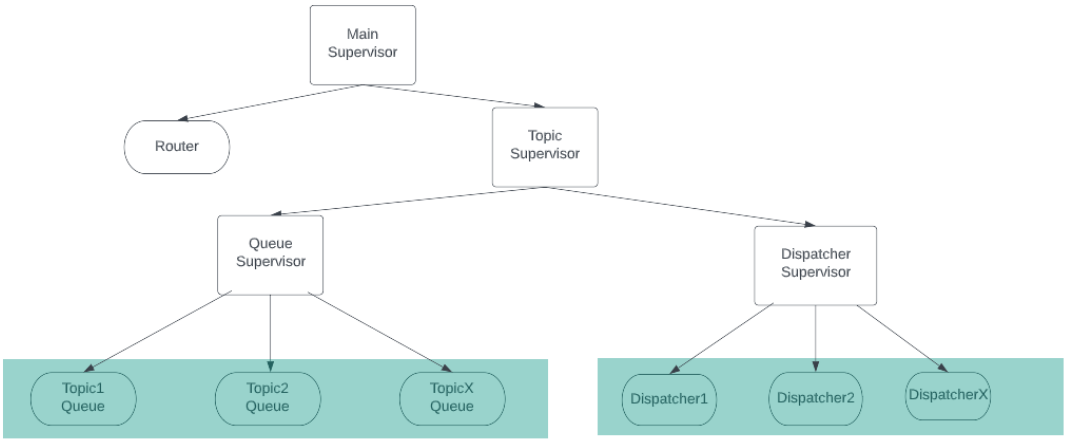
the other side it pushes them to queues. The exchange must know exactly what to do with a message it receives. Should it be appended to a particular queue? Should it be appended to many queues? Or should it get discarded. The rules for that are defined by the *exchange type*.
Our exchange type is by **topic**.

Stack:
– Akka
– borer - for JSON serialization
https://github.com/sirthias/borer
– alpakka: akka + SSE + HTTP - for reading streams

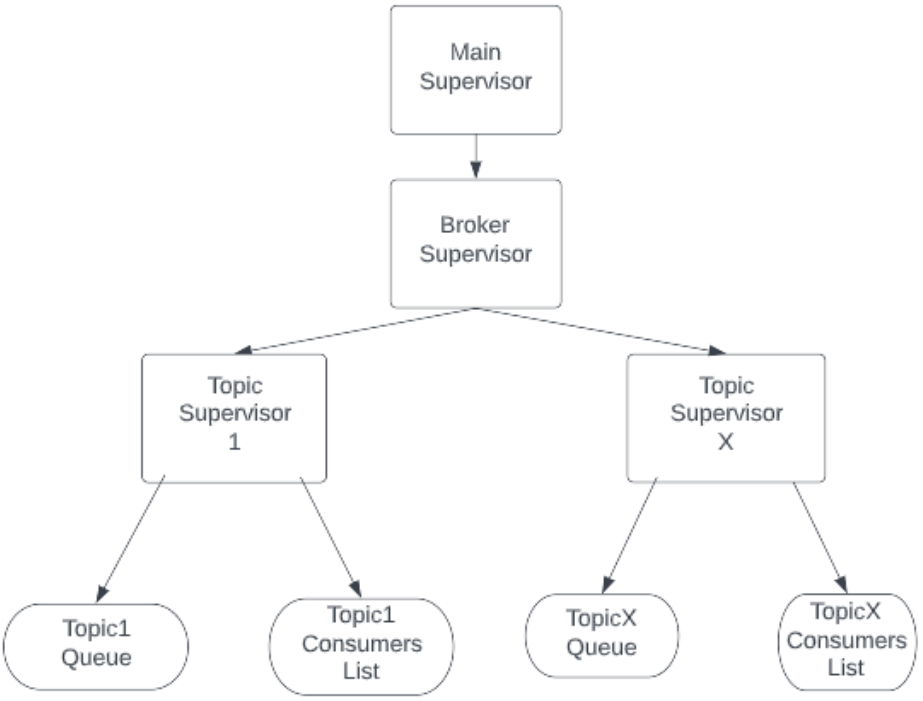
General Architecture



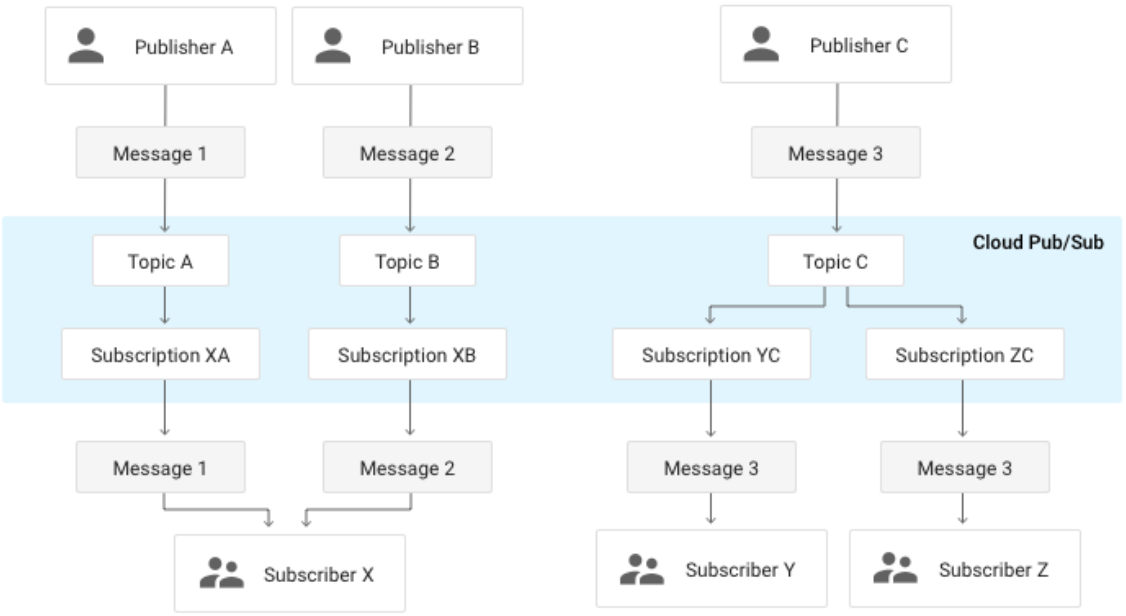
Supervision Tree:



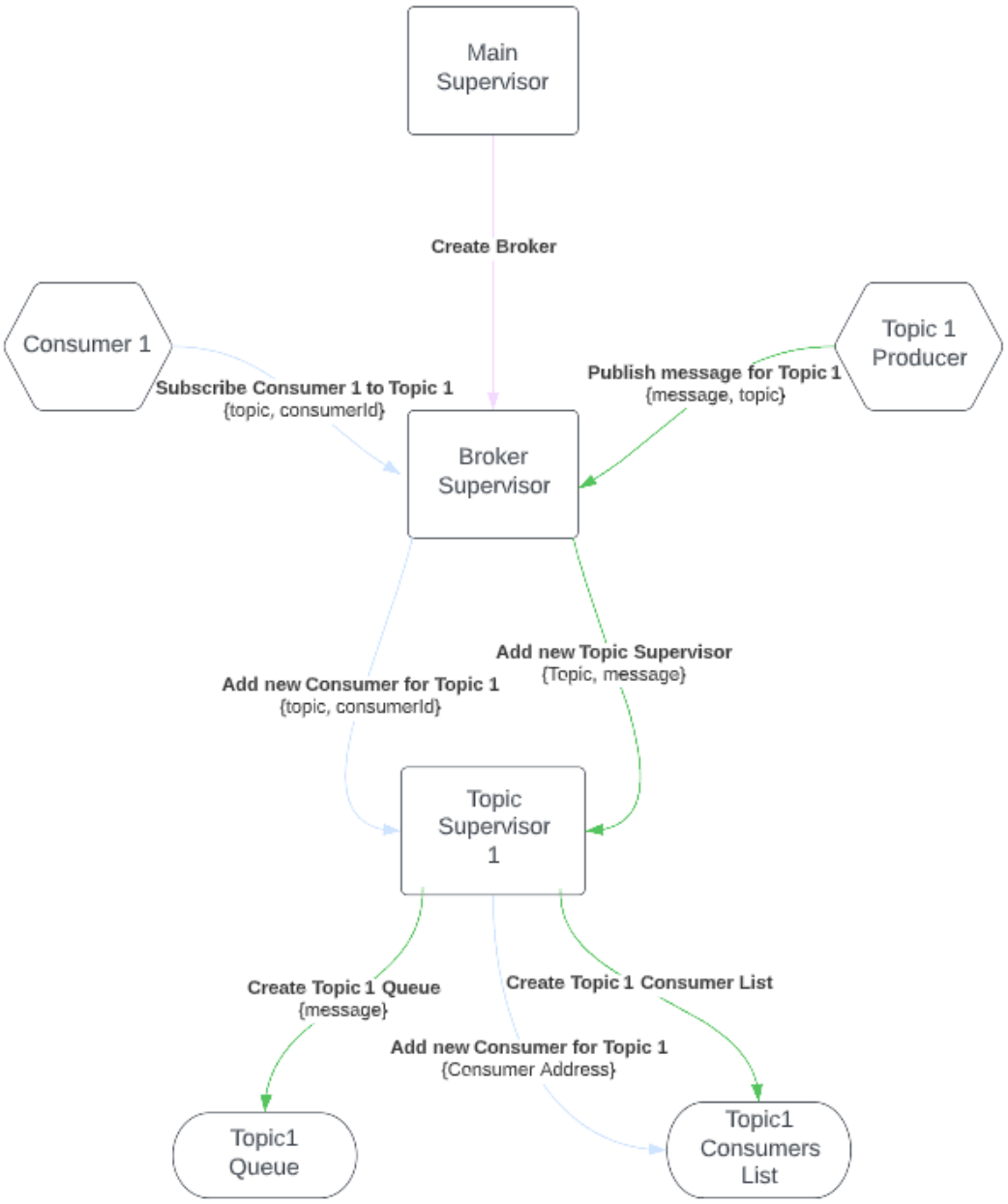
Due to the fact that Dispatchers after some research and development seemed redundant, the Supervision Tree was simplified to:



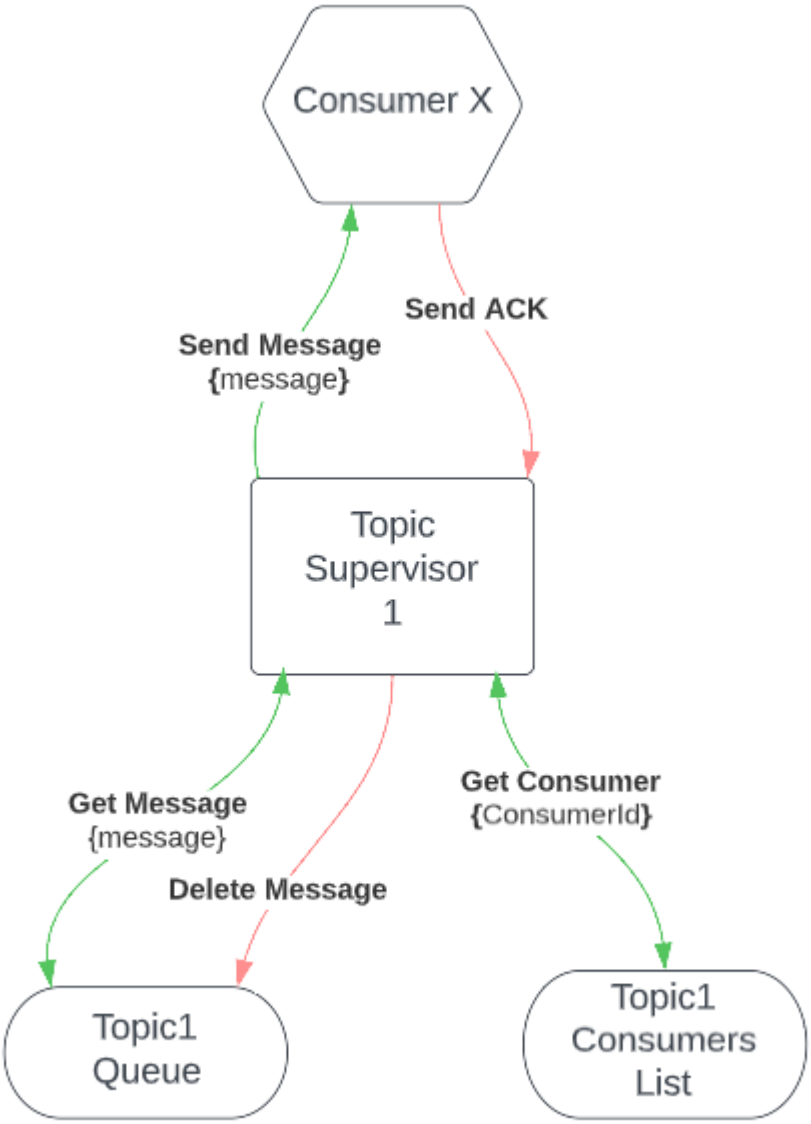
Pub-Sub Tree Diagram


Publisher A
Publisher B
Publisher C
Message 1
Message 2
Message 3
Cloud Pub/Sub
Topic A
Topic B
Topic C
Subscription XA
Subscription XB
Subscription YC
Subscription ZC
Message 1
Message 2
Message 3
Message 3
Subscriber X
Subscriber Y
Subscriber Z

Message Flow Diagram



The pink one is message broker start. The Green flow is the publication of Producer 1 to Topic 1.
The green one is subscription of Consumer 1 to Topic 1.



Here is what is happening after pub and sub. The green flow is the transfer of the message to the Consumer. The pink one is the flow when the acknowledgement arrived and the message is deleted if no other consumers need it.