**Pub/sub model**

What is pub/sub messaging?

Publish-subscribe messaging, or pub/sub messaging, is an asynchronous communication model that makes it easy for developers to build highly functional and architecturally complex applications in the cloud. In modern cloud architecture, applications are decoupled into smaller, independent building blocks called *services*. Pub/sub messaging provides instant event notifications for these distributed systems. It supports scalable and reliable communication between independent software modules.

## How does pub/sub messaging work?

The publish-subscribe (pub/sub) system has four key components.

### **Messages**

A message is communication data sent from sender to receiver. Message data types can be anything from strings to complex objects representing text, video, sensor data, audio, or other digital content.

### **Topics**

Every message has a topic associated with it. The topic acts like an intermediary channel between senders and receivers. It maintains a list of receivers who are interested in messages about that topic.

### **Subscribers**

A subscriber is the message recipient. Subscribers have to register (or subscribe) to topics of interest. They can perform different functions or do something different with the message in parallel.

### **Publishers**

The publisher is the component that sends messages. It creates messages about a topic and sends them once only to all subscribers of that topic. This interaction between the publisher and subscribers is a one-to-many relationship. The publisher doesn’t need to know who is using the information it is broadcasting, and the subscribers don’t need to know where the message comes from.

## What are the benefits of pub/sub messaging?

The publish-subscribe (pub/sub) model enables [event-driven architecture](https://aws.amazon.com/event-driven-architecture/), which is required in several modern applications. You can use events to trigger and communicate between decoupled services. An event is a change in state, or an update, like an item being placed in a shopping cart.

Pub/sub messaging provides significant advantages to developers who build applications that rely on real-time events. We outline some of the advantages below.

### **Eliminate polling**

Message topics allow instantaneous, push-based delivery, eliminating the need for message consumers to periodically check, or poll, for new information and updates. This promotes faster response time and reduces the delivery latency that can be particularly problematic in systems where delays cannot be tolerated.

### **Dynamic targeting**

The pub/sub pattern makes the discovery of services easier, more natural, and less error-prone. Instead of maintaining a roster of peers so an application can send messages, a publisher will simply post messages to a topic. Then, any interested party will subscribe its endpoint to the topic and start receiving these messages. Multiple subscribers can change, upgrade, or disappear, and the system adjusts dynamically.

### **Decouple and scale independently**

Pub/sub makes the software more flexible. Publishers and subscribers are decoupled and work independently from each other, which allows you to develop and scale them independently. You can decide to handle orders one way this month and then another the following month. Adding or changing functionality won’t send ripple effects across the system, because pub/sub allows you to flex how everything works together.

### **Simplify communication**

Code for communications and integration is some of the hardest code to write. The publish-subscribe model reduces complexity by removing all the point-to-point connections with a single connection to a message topic. The topic will manage subscriptions to decide what messages should be delivered to which endpoints. Fewer callbacks result in looser coupling, plus code that is easier to maintain and extend.

### **Durability**

Pub/sub messaging services often provide very high durability, and at-least-once delivery, by storing copies of the same message on multiple servers.

### **Security**

Message topics authenticate applications that try to publish content and allow you to use encrypted endpoints to secure messages in transit over the network.

**Reference:**

https://aws.amazon.com/what-is/pub-sub-messaging/