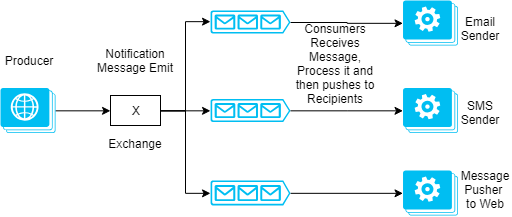
Notification System

High level design:



Above diagram depicts high level design of the notification system. Basically, it consists of 3 basic components.

1. Notification Message Producer & Exchange
2. Message Brokers
3. Notification Message Processor/Consumer

Components of a scalable notification system:

**Notification Message Producer & Exchange**

The role of this component is to trigger send notification request message to exchange. The producer never sends any messages directly to a queue. Quite often the producer doesn’t even know if a message will be delivered to any queue at all. Instead, the producer can only send messages to an exchange. 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 tech stack for producer can be anything. I will restrict this entity in above diagram to any tech stack.

**Storage and message brokers**

First, we will need to pick a message broker for fanout on write. My recommendation is **RabbitMQ for mid-size projects**. If you have more time available **Kafka** is a great option. It scales much better than RabbitMQ.

**Notification Message Processor**

For processing Feeds stored I would recommend **Dotnet Core/Nodejs**.

**Dotnet core**: As it gives you advantage of easy implementation non-blocking IO operations and great performance for cpu intensive operations.

**Nodejs**: In case, feed processing doesn’t involve much processing then my choice would be nodejs for this problem as javascript naturally gives non-blocking IO operations. So, we can easily ensure, there is no possibility of blocking for external services.

**Realtime**

Faye is a great open source project. In terms of hosted solutions PubNub and Pusher are awesome options. Firebase can be used for mobile notifications.