Pub/Sub & Queue/Job Server With PostgreSql

Daniel Valencia
Machine Learning Engineer
https://**loxo.co**

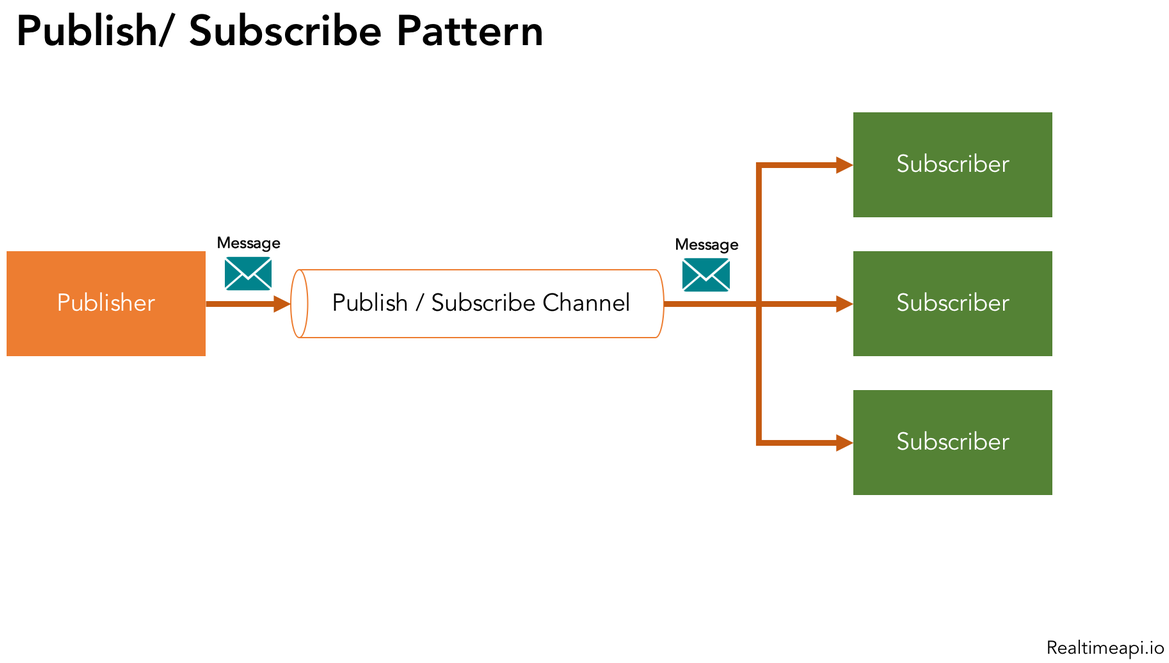
Topics

1. What is Pub/Sub?

2. Why Pub/Sub & Queue Server with Postgresql

3. Python Demo

What is Pub/Sub?



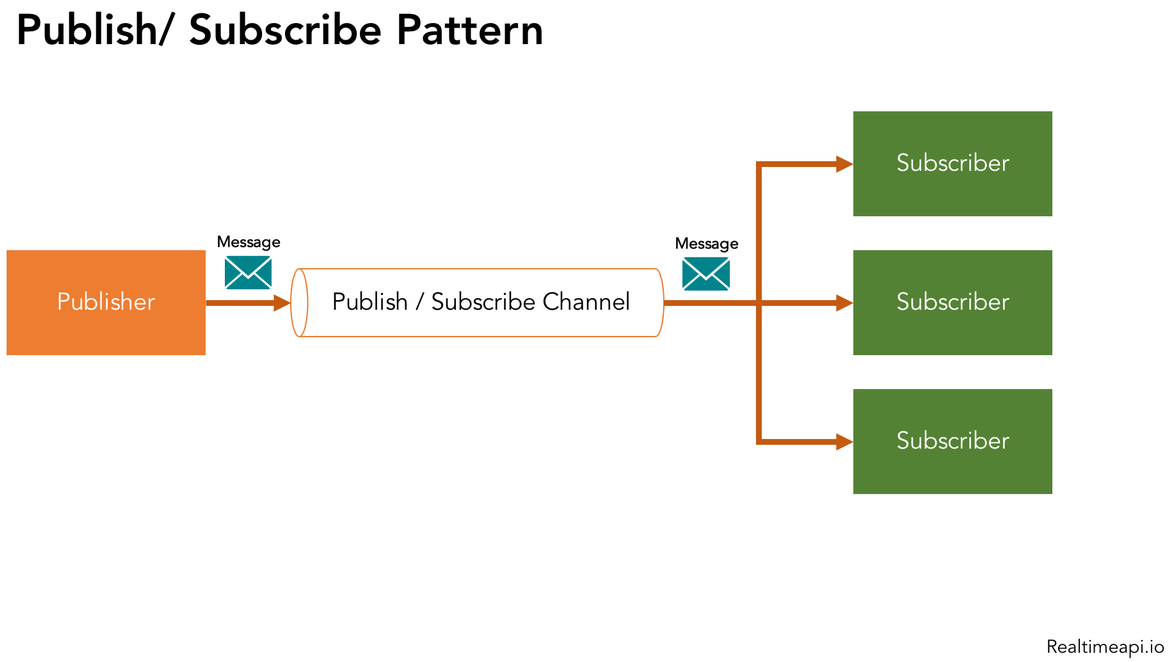
If you're making any project of sufficient complexity, you'll need a publish/subscribe server to process events.

> *In software architecture, **publish–subscribe** is a messaging pattern where senders of messages, called **publishers**, do not program the messages to be sent directly to specific **receivers, called subscribers**, but instead categorize published messages into classes without knowledge of which subscribers, if any, there may be.*
>
> *Similarly, subscribers receive messages that are of interest, without knowledge of which publishers, if any, there are.*
>
> *Publish–subscribe is a sibling of the message queue paradigm, and is typically one part of a larger message-oriented middleware system.*

In a nutshell

So the publisher, register messages somewhere, then a subscriber *listen* messages from somewhere regardless who send it.

This is basically telling us, we can have any kind of channel to send and receive messages.
In our context, a *message* is basically a job or a task (and all its context required to complete it) and that task could take several time to complete and we need to know if that task was completed successfully or not and how to recover in case it failed and make it *try again*



Why Pub/Sub With Postgresql?

- **Many SQL users (and companies) already have Postgres installed** for use as a database, so there are no extra setup costs in time and effort for using it for pub/sub

- **Many developers already *KNOW* PostgreSql** so investing a little bit of more time learning how to get the most of it will return a lot of value instead of begin with a *new* framework but without the chance on getting to know it in detail.

- **Postgres has very good persistence guarantees** - It's easy to query "dead" jobs with, e.g., `SELECT * FROM ci_jobs WHERE status='initializing' AND NOW() - status_change_time > '1 hour'::interval` to handle workers crashing or hanging.

- Since **jobs are defined in SQL, it's easy to generate Graphql or ORM (ActiveRecord/SQLALchemy) representations of them** (i.e., to provide APIs that checks the run status.)

- **It's easy to have multiple subscribers.** For instance, a script running in background listening status changes. It could be thousands of small nodes with scripts listening status changes ... that's called, scaling!

- **Postgres has very good language support**, with bindings for most popular languages. This is a stark difference from most other pub/sub servers. So, no need to learn a new Framework/Toolkit, etc.

But, in my company we handle **Billions** of messages...

- Really?. Think about it slowly, is your company *killing a fly with a nuke*?
- Perhaps this doesn't look hyped (or at least not artificially hyped) but it works.
- Remember, we are using tables to register messages (jobs/tasks).
  - You can add or improve indices in a table.
  - You can read from a db replica to consume the messages. No impact in the production db.
  - You can build a pg cluster if the "problem" is really big.
  - You can tune pg config to improve pg performance.
  - There are so many ways you can make this run faster. It's a relational db. Well known, well documented, well tested.

Python Demo

Find the code here: https://github.com/dafevara/pg_pubsub_py

Clone it!, Fork it!, Use it!

Thanks

dafevara@gmail.com
twitter.com/dafevara
github.com/dafevara

*“inspiration”: https://layerci.com/blog/postgres-is-the-answer/