RabbitMQ / Bunny

RubyFTW
Tuesday, September 23, 2014

Dan Swain   @dantswain   http://dantswain.com   dan.t.swain@gmail.com
http://github.com/dantswain/rabbitmq_presentation

Real Time Bidding Lead
simpli.fi

About Simpli.fi

- Simpli.fi is a real-time bidding (RTB) platform

  - ~500,000 queries per second

  - < 50 msec latency

  - C++ bidder, Ruby infrastructure, Erlang services

- We use RabbitMQ at Simpli.fi

  - Relatively new to us

  - Distribute thousands of messages per second across multiple data centers

Why use RabbitMQ?

- Message-based architecture is magical

  - Not the same way that Rails is "magical"

- Asynchronous request processing with built-in routing and queueing

- Trivially share data across multiple applications (pub/sub)

  - Potentially written in multiple languages

- Trivially scale up the number of workers

- Fault tolerance!

- No polling

Outline

- RabbitMQ concepts

- Ruby Bunny

- Example

- Advanced Message Queue Protocol

  - JP Morgan Chase, 2003

- Implementations of AMQP 1.0

  - SwiftMQ - Java (JMS)

  - Windows Azure Service Bus

  - Apache - Qpid, ActiveMQ, Apollo

  - IBM MQ Light

  - **RabbitMQ**

RabbitMQ™

- Rabbit Technologies, 2007 -> VMWare, 2010 -> Pivotal, 2013

- Open Source, written in Erlang

- Current - 3.3.5

- Also supports ZeroMQ

- Scalable by clustering*

- Failover support

- RabbitMQ In Action - Videla, Williams (Manning)
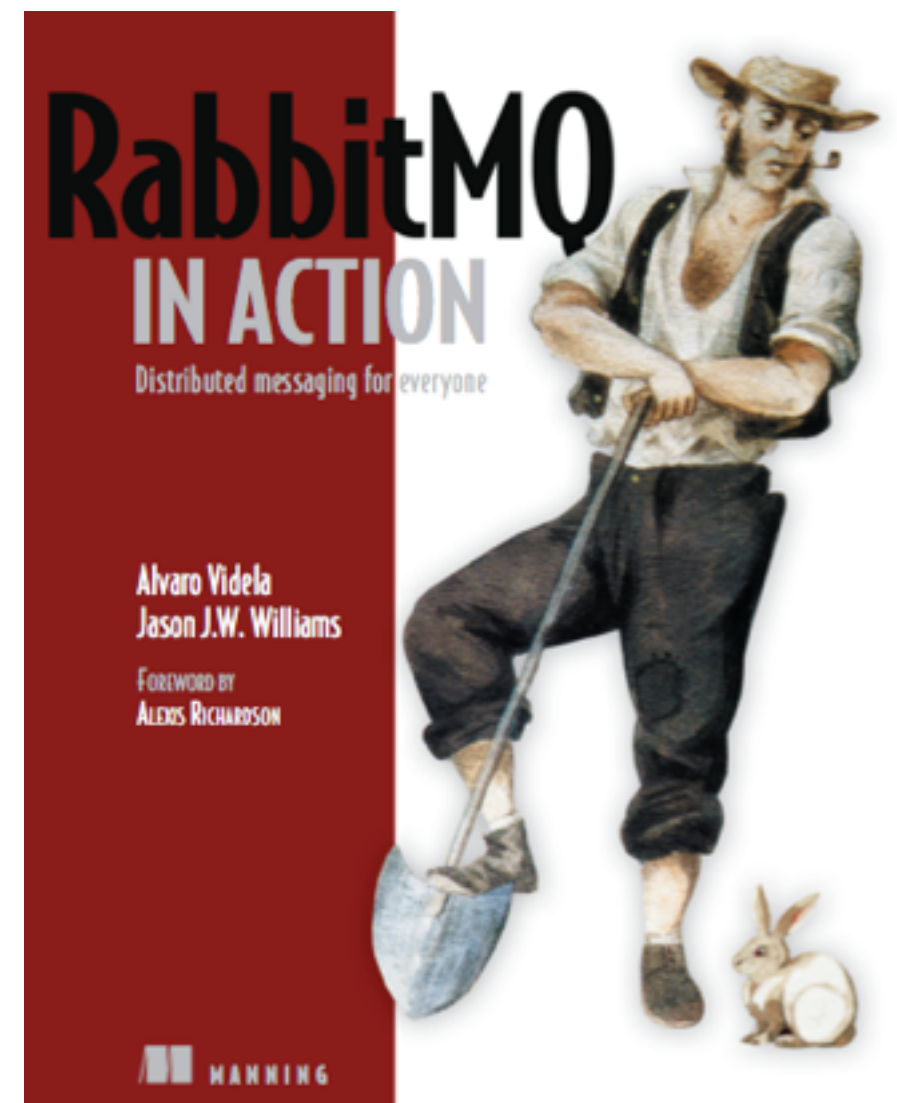
RabbitMQ™

- brew install rabbitmq / apt-get install rabbitmq-server

- rabbitmq-server -detached / rabbitmqctl stop

- Management Plugin - HTTP / REST - VERY useful -

  - rabbitmq-plugins enable rabbitmq_management

  - http://localhost:15672 - guest/guest

- rabbitmqadmin - Python script controls via REST

- Default username/password is guest/guest

  - Only works via 127.0.0.1
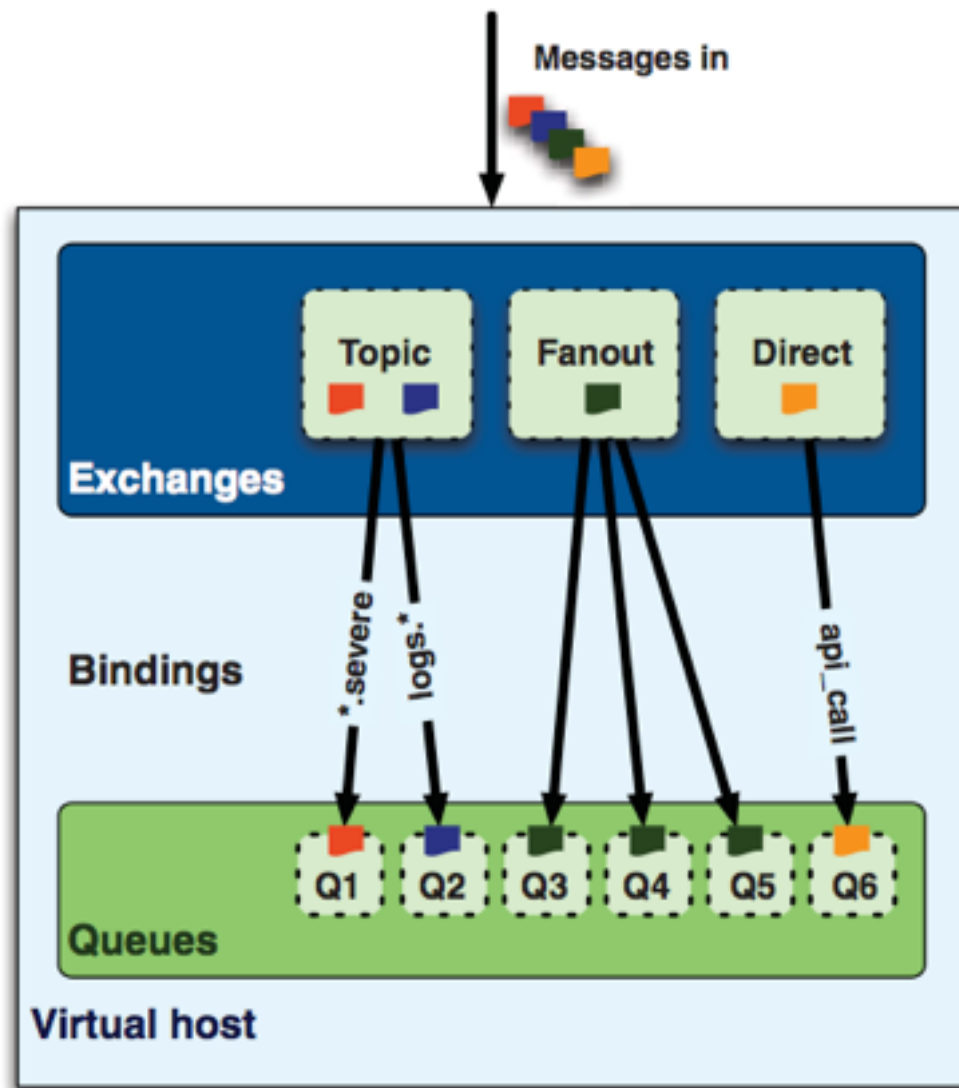
- Exchanges

  - Publish to exchanges

  - Routing logic

- Queues

  - Subscribe to queues

  - **Bind** queues to Exchanges

- Exchange Types: Direct, Fanout, Topic

- Queues and messages can be durable

- Virtual hosts - separate concerns

amqp uri
amqp://user:password@host/vhost
amqp://user:password@host <- default vhost = "/"

Distributing the rabbits

- Clustering intended for same-data center usage

- Two options: Shovels and Federation

- Shovels

    - More manual

    - More flexible

- Federation

    - More automated

    - Exchange has the appearance of existing on each broker

- github.com:dantswain/rabbitmq-federation-example

Rubying the rabbits: Bunny

- gem install bunny

- http://rubybunny.info - https://github.com/ruby-amqp/bunny

- rubybunny.info - "Guides"

- "Real" API doc site: http://reference.rubybunny.info/

- Official RabbitMQ tutorials using Bunny:
  http://www.rabbitmq.com/tutorials/tutorial-one-ruby.html

Design patterns

Pedagogical

1 "Hello World!"

The simplest thing that does *something*

Python | Java | Ruby | PHP | C#

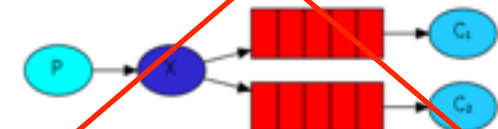(Use Topics)

2 Work queues

Distributing tasks among workers

Python | Java | Ruby | PHP | C#

Use Topics

3 Publish/Subscribe

Sending messages to many consumers at once
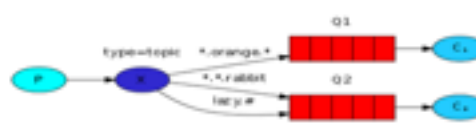
Python | Java | Ruby | PHP | C#

4 Routing

Receiving messages selectively

Python | Java | Ruby | PHP | C#

5 Topics

Receiving messages based on a pattern

Python | Java | Ruby | PHP | C#

6 RPC

Remote procedure call implementation

Python | Java | Ruby | PHP | C#

Use Topics

Good luck…

Lessons so far

- Everything looks like topics. It's OK.

- Let consumers ensure the queue/binding exist.

- Federation was tricky to figure out, but works great.

  - Caveat: If a node goes down, prepare for big queues.

- "Redeliver" doesn't necessarily mean what you think it means. Re-publishing always works.

- When queues get big, RabbitMQ gets slower, and can take the box with it.

- Give yourself an easy way to spawn extra workers.

- Use monit (or god, bluepill, whatever) to make sure workers come back up without intervention, but check that they are ACTUALLY coming back up (Groundhog Day suicide effect).