Microservice communication with RabbitMQ

Tetiana Chupryna
Disclaimer. Language
Татьяна Чуприна
Person of a million names

Татьяна Чуприна
Тетяна Чуприна
Тетяна Чуприна
Тетяна Чуприна
Tetyiiaana Chuprina
nikolay.klsn 😄 11:42 AM
Чупринас

каждый раз

ед

в голове ты в этой шляпе с маракасам

👍 2 🇷🇺 2

nikolay.klsn 😴 1:21 PM
look up in the sky
it's a bird
it's a plain
no, it's cpt. Chuprynas
Tania
Agenda

❖ Inter-service communication problematics
❖ JSON over HTTP option
❖ Message queues option
❖ RabbitMQ
❖ Integration with Ruby
The importance of Metaphors

“Important developments often **arise** out of **analogies**. By comparing a topic you understand poorly to something similar you understand better, you can come up with **insights** that result in a better understanding of the less-familiar topic.”

—Steven McConnell, Code complete
Luggage for dummies
Luggage for dummies
Luggage for dummies

Drop-off
Luggage for dummies

Drop-off

Sorting
Luggage for dummies
Luggage for dummies

Drop-off

Sorting

Sorting
Luggage for dummies

Drop-off

Sorting

carousel
Luggage for dummies

Drop-off

Sorting

carousel
Game changer
YOU
TRAVEL
Event-based system
Scalable
Maintainable
Back to clouds
Microservice communication
Inter-service communication

- New problem which doesn’t exist in monolith
- So many ways to success
- So many pitfalls
JSON over HTTP

Reuse API

at least Basic Auth

100% control
JSON over HTTP
You can’t be 100% sure
500

The server encountered an error processing your request.
Design failure to success
Sidekiq to the rescue

- Handle “other side” failures gracefully
- Retry if needed
Procs

- Easy
- Quick
- Dirty
- Great choice for small apps
Cons

- Bad in scaling
- Pain for maintenance
- Net::HTTP
Inter-service communication
Break it. Make it.

- Decouple service interaction
- Keep messages in temporary storage (mailbox?)
- Make all asynchronous
Message queues

- Message-oriented middleware
- Applicable not only for web-applications
- Async communication protocol
RabbitMQ
Reasons to use

- Written in Erlang
- Fast & Reliable
- Easy to setup
- Client libraries on all major languages (Ruby yippee!)
- Advanced Message Queuing Protocol
In a nutshell
Direct exchange routing
Fanout exchange routing
Topic exchange
Queue properties

- Name
- Durable
- Exclusive
- Auto-delete
- Arguments
Bindings

- Direct: by routing_key
- Multiple bindings
- Mask routing
Security

- TLS (transport layer security)
- SSL
- authentication
- authorization
Setup Rabbit

- Install and run
- No painful dependencies
Bunny

- Ruby client for RabbitMQ
- Lightweight
- Up&go
Bunny usage

class Commute
  def initialize(msg, key)
    @msg = msg
    @key = key
  end

  def publish
    conn = Bunny.new
    conn.start
    ch = conn.create_channel
    x = ch.direct('ruby_hat')
    x.publish(@msg, routing_key: @key)
    puts "[x] Sent #{msg} to #{key}"
    conn.close
  end
end
Sneakers

❖ Background processing framework for Ruby and RabbitMQ
❖ Can be used for high workload
❖ Up&go (as always)
Worker

class RabbitWorker

  include Sneakers::Worker

  from_queue :ruby

  def work(msg)
    Operator::Rabbit.process(msg)
    ack!
  end
end
Procs

- Scalable
- Maintainable
- Lightweight
Cons

- Need additional infrastructure $eq(Money)$
- Overkill for small projects
- Need additional knowledge
Afterwords

- Decoupling message flow from app changes app design and in general makes all easier.
- Decoupling message flow makes programs more safe and scalable.
- Message queues are a save choice for inter-service communication where RabbitMQ is a solid option.
- Ça dépend
Share your stories of inter-service communication
Massive thanks
Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks Thanks