gen_rmq

Simpler RabbitMQ consumers and publishers in Elixir



Elixir Berlin Meetup 08.11.2018, Berlin Mateusz Korszun @mkorszun

(O) Meltwater

How Meltwater uses RabbitMQ

Meltwater

- Number of clusters across the whole company
- Main data exchange mechanism

Meltwater API

- ~15 microservices
- Half of them publishes / consumes messages to / from RabbitMQ
- 2 RabbitMQ clusters: self managed and CloudAMQP



RabbitMQ + Elixir @ Meltwater API

- amqp elixir client
- Our typical consumer setup:
 - Long lived consumers
 - Creating connection / channel and keeping it in a state
 - Creating and binding a queue
 - Deadletter setup
 - Handling reconnections / consumer cancellations



Consequences

- A lot of duplication and boilerplate
- "Standards" diverging across our different projects

```
196 lines (159 slee) | 6-32 KB
                 (sek, conn) --
Process, monitor(conn.pid)
                       gueue - setup_rabbit(chan, config)
                 Setup_deadletter(chan, exchange_error, queue_error)
                # Asks the server to redeliver all unasknowledged messages on the 
# consumer's channel 
def handle_sall((recover, requeue), _From, %(in: channel) = state) do 
(!reply, Besis-recover(channel, requeue) requeue) ateto)
             defp Bugsned_metadata(%ficesage[
   attributes: %cdetivery_lag; tag, routing_key: routing_key),
   myleast_invicest_myleas
             Odoc "Acknowledge a message to the rabbiling Broker"
dar ank(Message(states %(in: Sharnet), attributes %(delivery_nag. vag))) de
5x15. arcmount("Inseque, varenage, varenage)
```

Simplified Consumer

```
defmodule Consumer do
  @behaviour GenRMQ.Consumer
  def init() do
      queue: "gen rmq in queue",
      exchange: "gen rmq exchange",
      routing key: "#",
      prefetch count: "10",
      uri: "amqp://guest:guest@localhost:5672"
  end
  def consumer tag() do
    "test tag"
  end
  def handle message(%GenRMQ.Message{} = message) do
    . . .
    GenRMQ.Consumer.ack (message)
  end
end
. . .
GenRMQ.Consumer.start link(Consumer, name: Consumer)
```

Simplified Publisher

```
defmodule Publisher do
  @behaviour GenRMQ.Publisher
  def init() do
      exchange: "gen rmq exchange",
      uri: "amqp://guest:guest@localhost:5672"
  end
end
GenRMQ.Publisher.start link(Publisher, name: Publisher)
GenRMQ.Publisher.publish(Publisher, "msg")
```

Simplified Tests

<u>GenRMQ.RabbitCase</u> - test helpers:

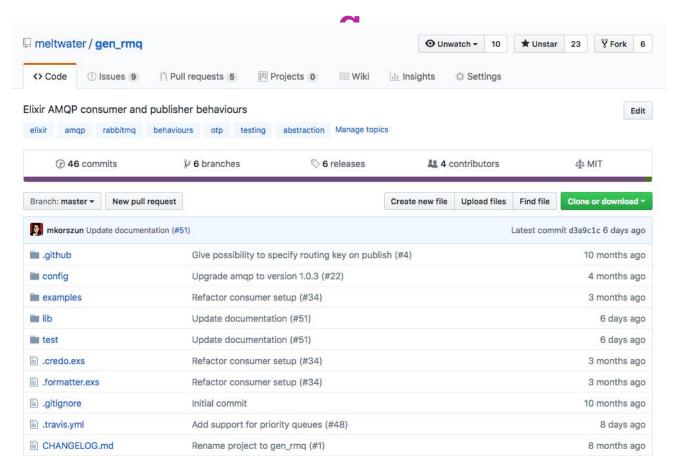
- Test queue setup
- Publish / consume test messages
- Queue count / cleanup

```
use GenRMQ.RabbitCase

setup_all do
   {:ok, conn} = rmq_open(@uri)
   :ok = setup_out_queue(conn, @out_queue, @exchange)
   {:ok, rabbit_conn: conn, out_queue: @out_queue}
end

setup do
   purge_queues(@uri, [@out_queue])
end
```

https://github.com/meltwater/gen_rm





Thank You! Q&A

Looking for Elixir jobs in Berlin? We hire Juniors too!

<u>Influencer Content Team</u> | <u>API Team</u>

