Developing RabbitMQ plugins in Elixir

Matteo Cafasso
Lead Software Engineer @ F-Secure

GitHub noxdafox

Twitter @nox_da_fox

2018-11-12

$ whoami

Software Engineer @ F-Secure Corporation

Backend development

Distributed systems enthusiast.

Malware analysis automation

Behavioural/dynamic malware analysis.

RabbitMQ Summit 2018

Topic

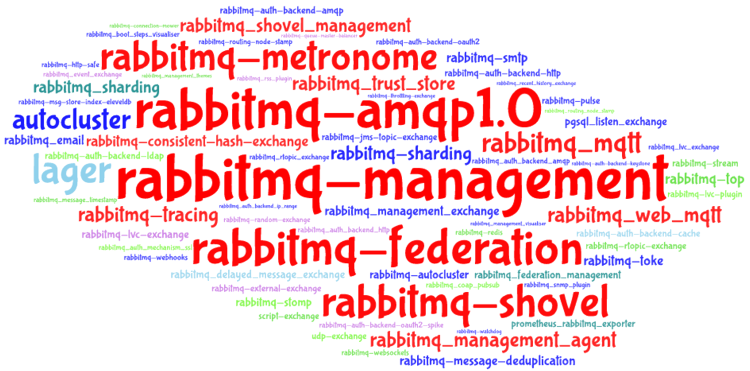
RabbitMQ plugins

Erlang/Elixir Design Principles

Hello World plugin

Case study and Conclusions

RabbitMQ plugins


rabbitmq-auth-backend-amqp
rabbitmq-connection-mower
rabbitmq_shovel_management
rabbitmq_boot_steps_visualiser
rabbitmq-queue-master-balancer
rabbitmq-routing-node-stamp
rabbitmq-auth-backend-oauth2
rabbitmq-http-safe
rabbitmq-metronome
rabbitmq-smtp
rabbitmq_event_exchange
rabbitmq_management_themes
rabbitmq_rss_plugin
rabbitmq-auth-backend-http
rabbitmq_sharding
rabbitmq_trust_store
rabbitmq_recent_history_exchange
rabbitmq-throttling-exchange
rabbitmq-msg-store-index-eleveldb
rabbitmq-pulse
autocluster
rabbitmq-amqp1.0
rabbitmq_routing_node_stamp
pgsql_listen_exchange
rabbitmq_email
rabbitmq-consistent-hash-exchange
rabbitmq-jms-topic-exchange
rabbitmq_mqtt
rabbitmq_lvc_exchange
rabbitmq-auth-backend-ldap
rabbitmq_rtopic_exchange
rabbitmq-sharding
rabbitmq_auth_backend_amqp
rabbitmq-auth-backend-keystone
rabbitmq-stream
lager
rabbitmq-management
rabbitmq-top
rabbitmq_message_timestamp
rabbitmq-lvc-plugin
rabbitmq_auth_backend_ip_range
rabbitmq-tracing
rabbitmq_management_exchange
rabbitmq_web_mqtt
rabbitmq-random-exchange
rabbitmq_management_visualiser
rabbitmq-lvc-exchange
rabbitmq_auth_backend_http
rabbitmq-redis
rabbitmq-auth-backend-cache
rabbitmq_auth_mechanism_ssl
rabbitmq-federation
rabbitmq-rtopic-exchange
rabbitmq-webhooks
rabbitmq-toke
rabbitmq_delayed_message_exchange
rabbitmq-autocluster
rabbitmq_federation_management
rabbitmq-external-exchange
rabbitmq_coap_pubsub
rabbitmq_snmp_plugin
rabbitmq-stomp
rabbitmq-shovel
script-exchange
rabbitmq-auth-backend-oauth2-spike
rabbitmq-watchdog
prometheus_rabbitmq_exporter
udp-exchange
rabbitmq_management_agent
rabbitmq-websockets
rabbitmq-message-deduplication

RabbitMQ Summit 2018

Installing RabbitMQ plugins

RabbitMQ plugins are shipped as Erlang `'.ez'` archive files. To install a plugin is enough to copy its archive file in the RabbitMQ plugins directory.
The `rabbitmq-plugins` tool can be used to load a plugin.

```
rabbitmq-plugins list                      # list the available plugins

rabbitmq-plugins enable <plugin-name>      # enable a plugin

rabbitmq-plugins disable <plugin-name>     # disable a plugin
```

RabbitMQ Summit 2018

Topic

RabbitMQ plugins

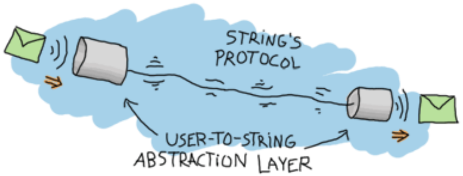
Erlang/Elixir Design Principles

Hello World plugin

Case study and Conclusions

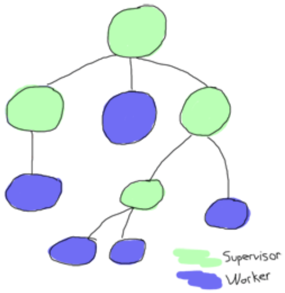
Process

A lightweight, isolated and independent unit of execution.
Processes rely on messages to communicate between each other.



Supervisor Process

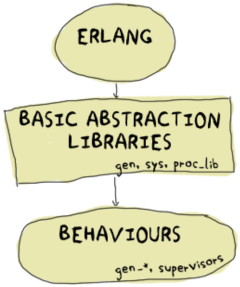
A process monitoring the behaviour of other processes. Supervisors can restart supervised processes if anomalies are detected.
Processes are often organised hierarchically in what is known as Supervisor Trees.



RabbitMQ Summit 2018

Behaviour

The formalisation of a common programming pattern.
Behaviours are contracts describing the expected callback functions to be implemented by the User.



Application

A component implementing some specific functionality, that can be started and stopped as a unit, and that can be reused in other systems.
Applications can be program structures consisting of processes or libraries organised as modules.

RabbitMQ plugin

Erlang Application consisting in one or more supervised processes interfacing with the broker via:

- message passing
- modules implementing specific behaviours

Topic

RabbitMQ plugins

Erlang/Elixir Design Principles

Hello World plugin

Case study and Conclusions

Requirements

- Erlang
- Elixir
- Git
- Make
- Zip

Makefile

```
PROJECT = my_first_rabbitmq_plugin

DEPS = rabbit_common rabbit
DEP_PLUGINS = rabbit_common/mk/rabbitmq-plugin.mk

# FIXME: Use erlang.mk patched for RabbitMQ, while waiting for PRs to be
# reviewed and merged.
ERLANG_MK_REPO = https://github.com/rabbitmq/erlang.mk.git
ERLANG_MK_COMMIT = rabbitmq-tmp

# https://github.com/rabbitmq/rabbitmq-common
include rabbitmq-components.mk
# https://github.com/ninenines/erlang.mk
include erlang.mk
```

Building the plugin

```
make
```

Fetches the dependencies and builds the source.

```
make run-broker
```

Builds and starts the server altogether with the plugin under development.

```
make tests
```

Runs the tests.

```
make dist
```

Builds the plugin and packages it in a `'.ez'` archive.

Elixir mix.exs

```elixir
def project do
  [
    app: :my_first_rabbitmq_plugin,
    version: "0.0.1",
    deps: deps()
  ]
end

# Do not compile RabbitMQ dependencies
defp deps() do
  [
    {
      :rabbit,
      path: "deps/rabbit",      # specify local dependencies folder location
      compile: ":",             # give a "noop" as compiling command
      override: true
    },
    ...
  ]
end
```

Makefile with Mix rule

```
PROJECT = my_first_rabbitmq_plugin

DEPS = rabbit_common rabbit
DEP_PLUGINS = rabbit_common/mk/rabbitmq-plugin.mk

# Add make app rule for Elixir plugin
elixir_srcs := mix.exs
app:: $(elixir_srcs) deps
	$(MIX) deps.get
	$(MIX) deps.compile
	$(MIX) compile

ERLANG_MK_REPO = https://github.com/rabbitmq/erlang.mk.git
ERLANG_MK_COMMIT = rabbitmq-tmp

include rabbitmq-components.mk
include erlang.mk
```

Hello World

```elixir
defmodule RabbitMQ.HelloWorldPlugin do
  Module.register_attribute __MODULE__,
    :rabbit_boot_step,
    accumulate: true, persist: true

  @rabbit_boot_step {__MODULE__,
                     [description: "hello world rabbitmq plugin",
                      mfa: {__MODULE__, :hello_world, []},
                      requires: :notify_cluster]}

  def hello_world() do
    IO.puts("Hello World!")
  end
end
```

Where to go next

Read the source!

Most of the interfaces and data structures can be found in the `'rabbitmq-commons'` repository.

Behaviours

- rabbit_exchange_type
- rabbit_backing_queue
- rabbit_authz_backend

Topic

RabbitMQ plugins

Erlang/Elixir Design Principles

Hello World plugin

Case study and Conclusions

RabbitMQ message deduplication plugin

Plugin for filtering duplicate messages.

Duplicates are identified based on the `'x-deduplication-header'` header content.
Deduplication can be applied on exchanges and on queues.

- `https://github.com/noxdafox/rabbitmq-message-deduplication`

Conclusions

Getting started is easy

RabbitMQ plugins tend to be simple

Two supported languages: Erlang and Elixir

Excellent learning experience

References

RabbitMQ Plugins and development guides

`https://www.rabbitmq.com/plugins.html`
`https://www.rabbitmq.com/plugin-development.html`

RabbitMQ Plugins in Elixir

`https://binarin.ru/post/rabbitmq-plugins-in-elixir/`

Slides and examples

`https://github.com/noxdafox/rabbitmqsummit2018`

More references

Rabbit Internals

https://github.com/videlalvaro/rabbit-internals

Learn You Some Erlang for Great Good!

https://learnyousomeerlang.com/

Elixir documentation

https://elixir-lang.org/getting-started/introduction.html