Dissecting the Rabbit: RabbitMQ Internal Architecture

Alvaro Videla - RabbitMQ

Alvaro Videla

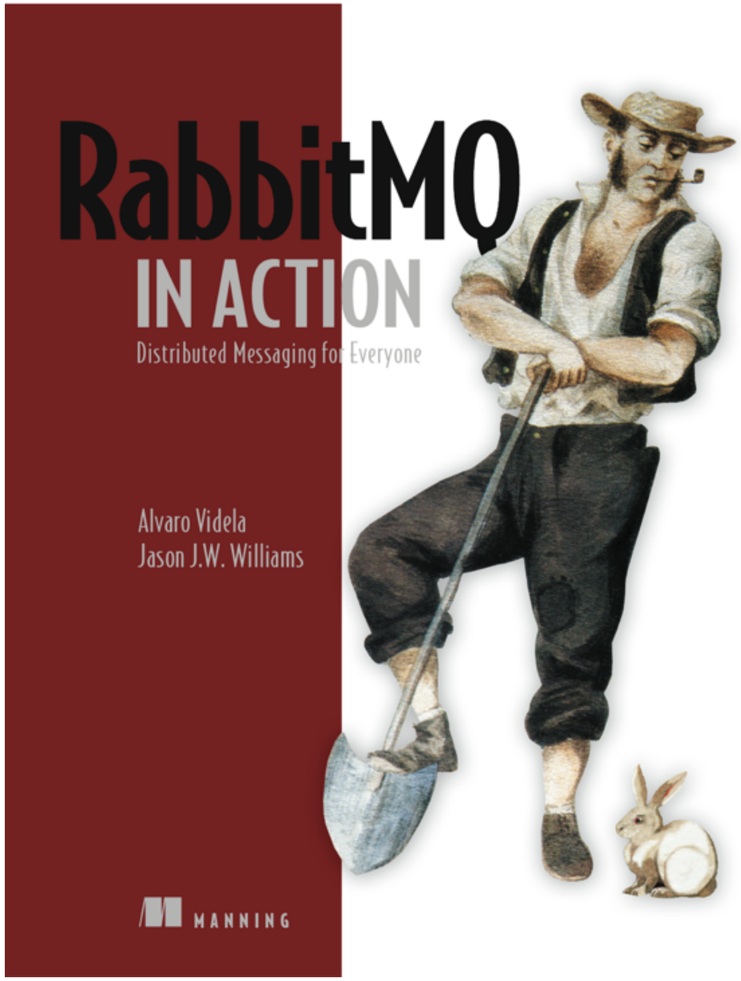
- Developer at RabbitMQ
- Co-Author of RabbitMQ in Action
- Creator of the RabbitMQ Simulator
- Blogs about RabbitMQ Internals: http://videlalvaro.github.io/internals.html
- @old_sound I alvaro@rabbitmq.com
  github.com/videlalvaro

About Me

Co-authored

RabbitMQ in Action

http://bit.ly/rabbitmq



We are hiring!

https://groups.google.com/forum/#!topic/rabbitmq-users/sCLl9eDbQoc

alvaro@rabbitmq.com


clochure.org
Clochure - A Better Clojure
Clochure
A Better Clojure
Clochure is an educated attempt to solve Clojure's number one problem and first obstacle that puts away newcomers:
parentheses
Tweet 424
Like 129
Get it now on Github

Agenda

- Intro to RabbitMQ
- Dive into RabbitMQ Internals
- A day in the life of a message
- RabbitMQ message store
- RabbitMQ behaviours and extensibility

What is RabbitMQ

RabbitMQ

RabbitMQ

RabbitMQ

- Multi Protocol Messaging Server

RabbitMQ

- Multi Protocol Messaging Server
- Open Source (MPL)

RabbitMQ

- Multi Protocol Messaging Server
- Open Source (MPL)
- Polyglot

RabbitMQ

- Multi Protocol Messaging Server
- Open Source (MPL)
- Polyglot
- Written in Erlang/OTP

Multi Protocol

http://bit.ly/rmq-protocols

Polyglot

Polyglot

Polyglot

- Java

Polyglot

- Java
- node.js

Polyglot

- Java
- node.js
- Erlang

Polyglot

- Java
- node.js
- Erlang
- PHP

Polyglot

- Java
- node.js
- Erlang
- PHP
- Ruby

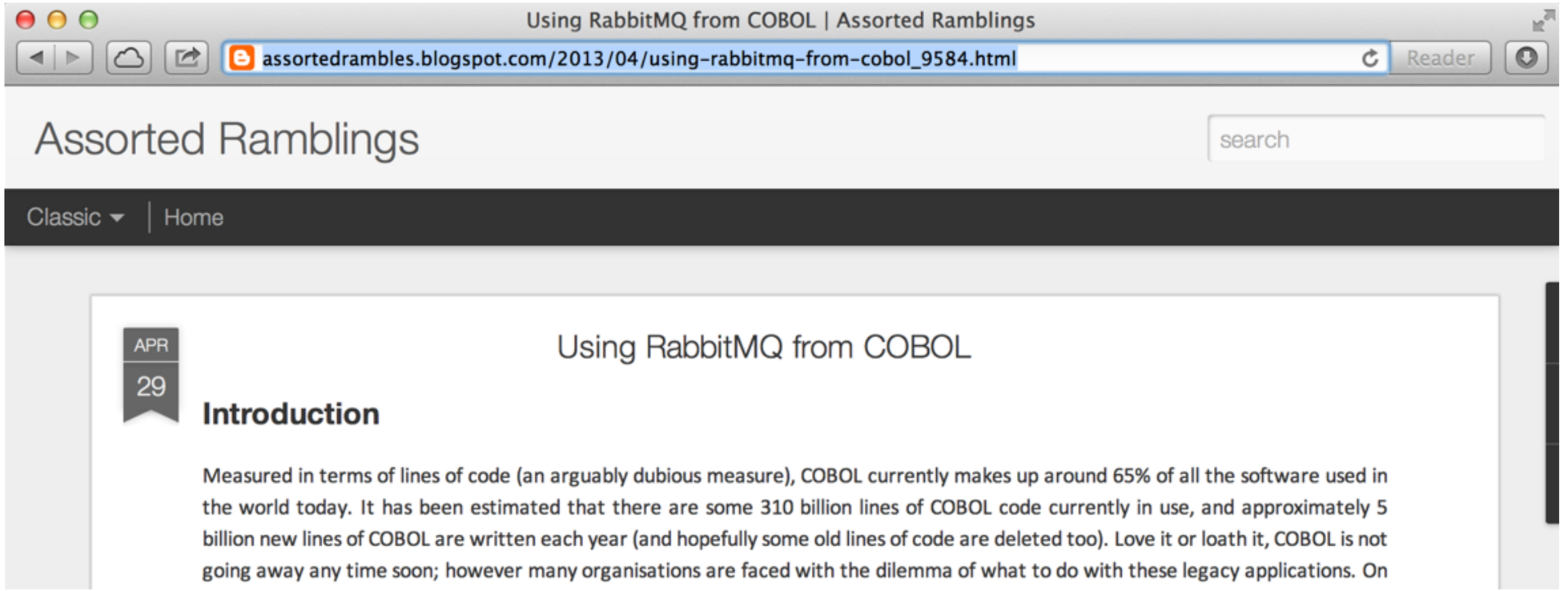
Polyglot

- Java
- node.js
- Erlang
- PHP
- Ruby
- .Net

Polyglot

- Java
- node.js
- Erlang
- PHP
- Ruby
- .Net
- Haskell

Polyglot



Even COBOL!!!11

Some users of RabbitMQ

Some users of RabbitMQ

- Instagram

Some users of RabbitMQ

- Instagram
- Indeed.com

Some users of RabbitMQ

- Instagram
- Indeed.com
- MailboxApp

Some users of RabbitMQ

- Instagram
- Indeed.com
- MailboxApp
- Mercado Libre

Some users of RabbitMQ

- Instagram
- Indeed.com
- MailboxApp
- Mercado Libre
- NHS

Some users of RabbitMQ

- Instagram
- Indeed.com
- MailboxApp
- Mercado Libre
- NHS
- Mozilla

http://www.rabbitmq.com/download.html

Unix - Mac - Windows

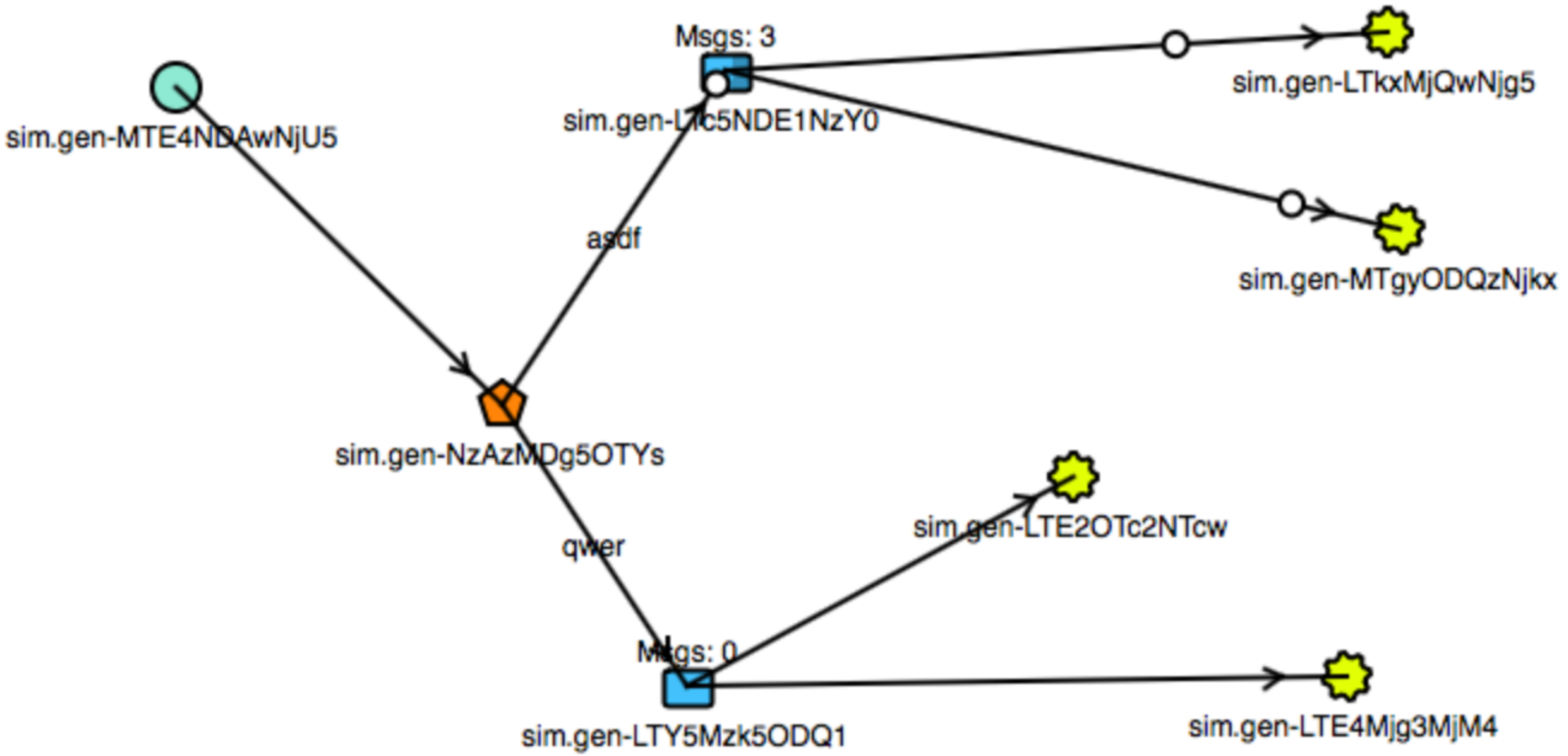
Messaging with RabbitMQ

A demo with the RabbitMQ Simulator

https://github.com/RabbitMQSimulator/RabbitMQSimulator

http://tryrabbitmq.com

RabbitMQ Simulator



RabbitMQ Internals

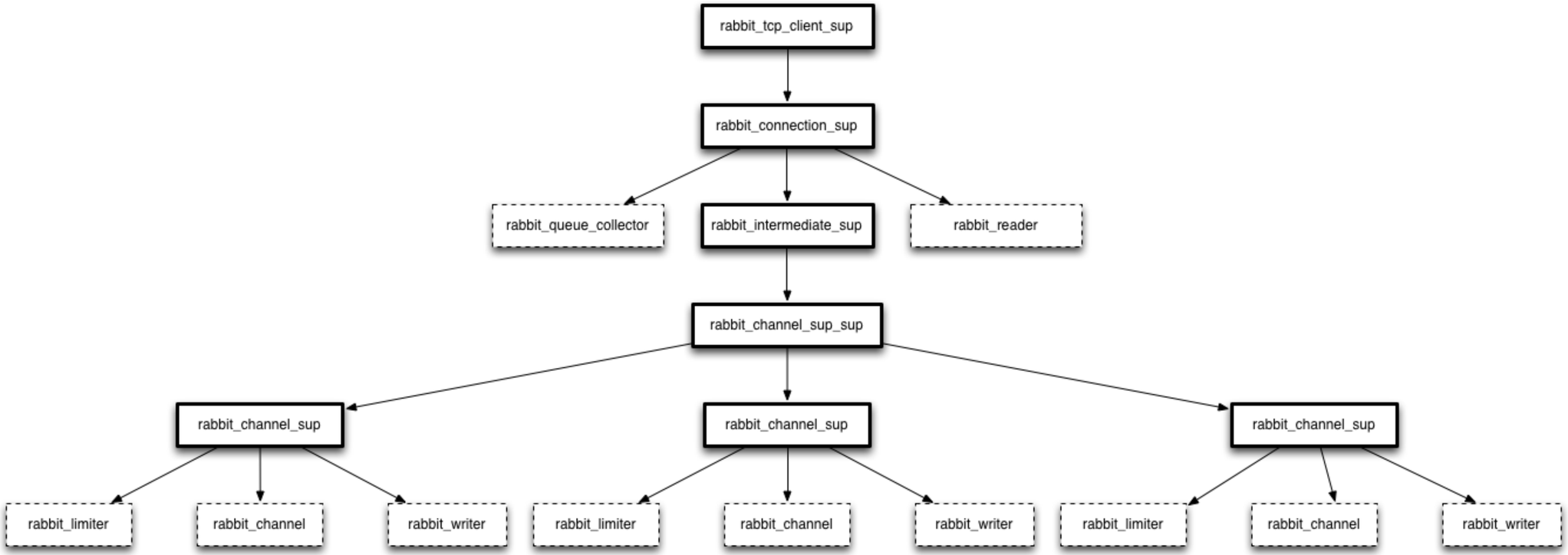
Opening an AMQP Channel

```
{ok, Connection} =
        amqp_connection:start(
           #amqp_params_network{host = “localhost"}),

{ok, Channel} = amqp_connection:open_channel(Connection).
```

What happens here?

```
{ok, Connection} =
        amqp_connection:start(
           #amqp_params_network{host = “localhost"}),

{ok, Channel} = amqp_connection:open_channel(Connection).
```

What happens here?



Erlang

Erlang App

Erlang App

- Processes (probably thousands)

Erlang App

- Processes (probably thousands)
- They communicate sending messages

Erlang App

- Processes (probably thousands)
- They communicate sending messages
- Each process has a message queue (don’t confuse with RabbitMQ queues)

Erlang App

- Processes (probably thousands)
- They communicate sending messages
- Each process has a message queue (don’t confuse with RabbitMQ queues)
- Virtual Machine has preemptive scheduler

Erlang Scheduler
Read More Here:

http://jlouisramblings.blogspot.ru/2013/01/how-erlang-does-scheduling.html

Erlang code structure

- Modules
- Functions
- Function Arity
- Arguments

```
M, F, A = Module, Function, Arguments
```

rabbit_client_sup.erl

```
-module(rabbit_client_sup).

-behaviour(supervisor2).

-export([start_link/1, start_link/2, start_link_worker/2]).

-export([init/1]).

-include("rabbit.hrl").
```

rabbit_client_sup.erl

```
start_link(Callback) ->
    supervisor2:start_link(?MODULE, Callback).

start_link(SupName, Callback) ->
    supervisor2:start_link(SupName, ?MODULE, Callback).

start_link_worker(SupName, Callback) ->
    supervisor2:start_link(SupName, ?MODULE, {Callback, worker}).

init({M,F,A}) ->
    {ok, {{simple_one_for_one, 0, 1},
          [{client, {M,F,A}, temporary, infinity, supervisor, [M]}]}};
init({{M,F,A}, worker}) ->
    {ok, {{simple_one_for_one, 0, 1},
          [{client, {M,F,A}, temporary, ?MAX_WAIT, worker, [M]}]}}.
```

Creating Processes

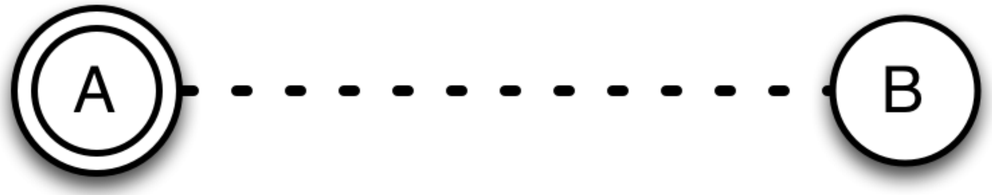
Creating Processes

```
Pid = spawn(Fun)
Pid = spawn(Node, Fun)
Pid = spawn(Module, Function, Args)
```

Fault Tolerance

Process Links

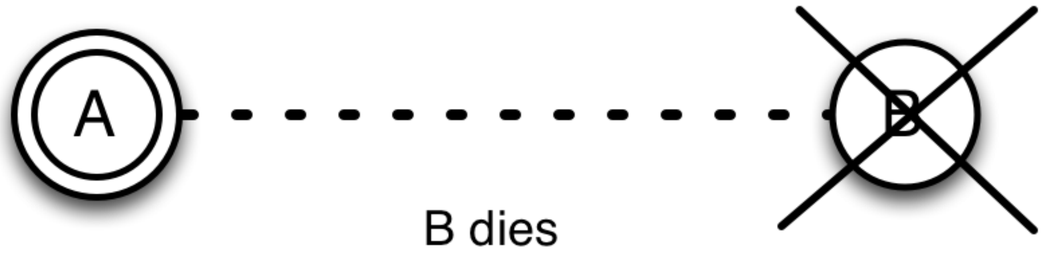
Process Links



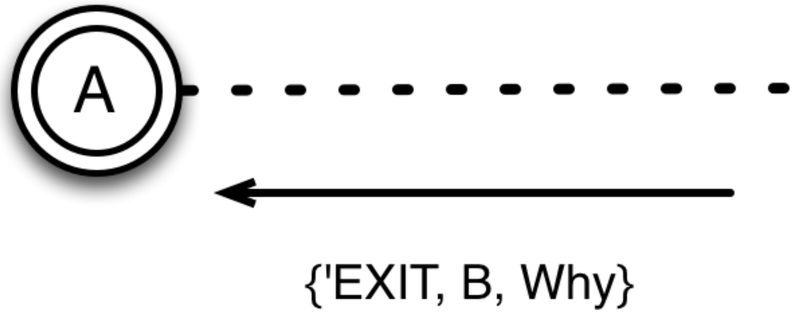
A traps exits

A is linked to B

Process Links



Process Links



Programming Idioms

```
Pid = spawn(fun () -> … end).
```

I don’t care if a process I create crashes

Programming Idioms

```
Pid = spawn_link(fun () -> … end).
```

I want to die if a process I create crashes

Programming Idioms

```
process_flag(trap_exit, true),
Pid = spawn_link(fun () -> … end).
```

I want to handle errors if a process I create crashes

Kinds of Processes

- normal processes

Kinds of Processes

- normal processes
- system processes

Kinds of Processes

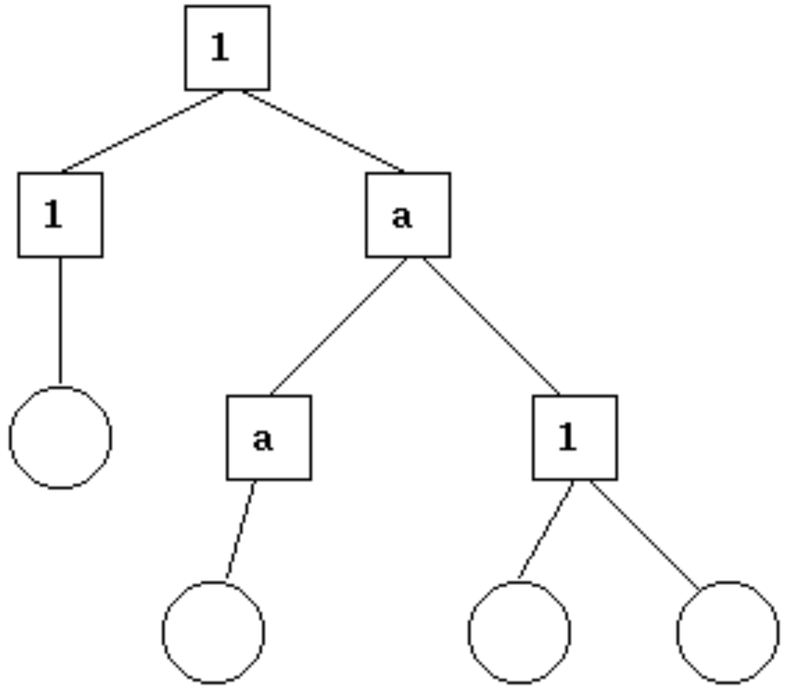
- normal processes (worker)
- system processes

Kinds of Processes

- normal processes (worker)
- system processes (supervisor)

Supervision Trees

Supervision tree



http://www.erlang.org/documentation/doc-4.9.1/doc/design_principles/sup_princ.html

Supervision tree

Supervision tree

- Worker Processes

Supervision tree

- Worker Processes
- Supervisor Processes

Supervision tree

- Worker Processes
- Supervisor Processes
- Supervision tree as a **hierarchical** arrangement of processes

Supervision tree

- Worker Processes
- Supervisor Processes
- Supervision tree as a **hierarchical** arrangement of processes

http://www.erlang.org/doc/man/supervisor.html

rabbit_client_sup

```
init({M,F,A}) ->
    {ok, {{simple_one_for_one, 0, 1},
          [{client, {M,F,A}, temporary,
            infinity, supervisor, [M]}]}};
```

rabbit_client_sup

```
init({M,F,A}) ->
    {ok, {{simple_one_for_one, 0, 1},
          [{client, {M,F,A}, temporary,
            infinity, supervisor, [M]}]}};
```

Child Spec - restart strategies

- one_for_one: only restart failing process
- one_for_all: restart failing process and all siblings
- simple_one_for_one: simplified version of one_for_one
- MaxR: maximum restarts allowed
- MaxT: in MaxT seconds

rabbit_client_sup

```
init({M,F,A}) ->
     {ok, {{simple_one_for_one, 0, 1},
           [{client, {M,F,A}, temporary,
             infinity, supervisor, [M]}]}};
```

Child Specification

```
child_spec() = {Id,StartFunc,Restart,Shutdown,Type,Modules}
```

Child Specification

```
child_spec() = {Id,StartFunc,Restart,Shutdown,Type,Modules}
 Id = term()
```

Child Specification

```
child_spec() = {Id,StartFunc,Restart,Shutdown,Type,Modules}
 Id = term()
 StartFunc = {M,F,A}
  M = F = atom()
  A = [term()]
```

Child Specification

```
child_spec() = {Id,StartFunc,Restart,Shutdown,Type,Modules}
 Id = term()
 StartFunc = {M,F,A}
  M = F = atom()
  A = [term()]
 Restart = permanent | transient | temporary
```

Child Specification

```
child_spec() = {Id,StartFunc,Restart,Shutdown,Type,Modules}
 Id = term()
 StartFunc = {M,F,A}
  M = F = atom()
  A = [term()]
 Restart = permanent | transient | temporary
 Shutdown = brutal_kill | int()>0 | infinity
```

Child Specification

```
child_spec() = {Id,StartFunc,Restart,Shutdown,Type,Modules}
 Id = term()
 StartFunc = {M,F,A}
  M = F = atom()
  A = [term()]
 Restart = permanent | transient | temporary
 Shutdown = brutal_kill | int()>0 | infinity
 Type = worker | supervisor
```

Child Specification

```
child_spec() = {Id,StartFunc,Restart,Shutdown,Type,Modules}
 Id = term()
 StartFunc = {M,F,A}
  M = F = atom()
  A = [term()]
 Restart = permanent | transient | temporary
 Shutdown = brutal_kill | int()>0 | infinity
 Type = worker | supervisor
 Modules = [Module] | dynamic
  Module = atom()
```

rabbit_client_sup

```
init({M,F,A}) ->
    {ok, {{simple_one_for_one, 0, 1},
          [{client, {M,F,A}, temporary,
            infinity, supervisor, [M]}]}};
```

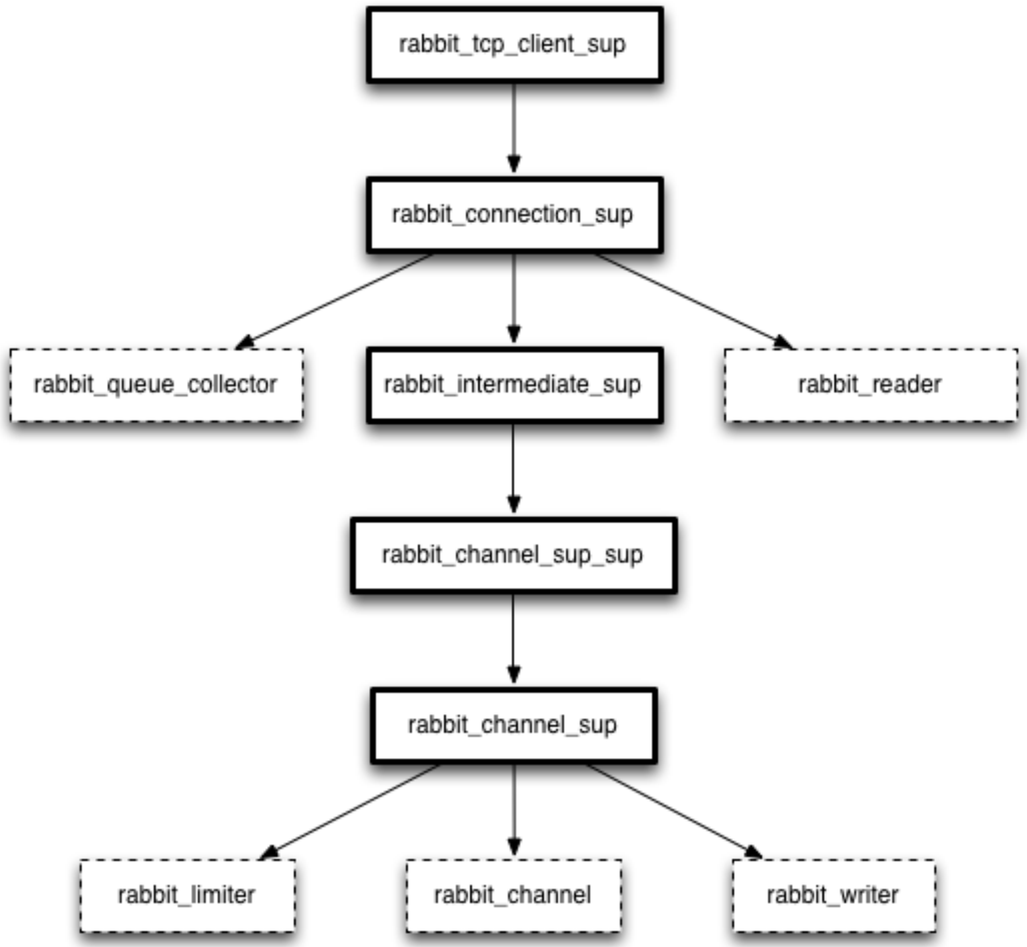
Child Spec - restart

- **permanent**: should always be restarted
- **temporary**: should never be restarted
- **transient**: should only be restarted if terminated abnormally

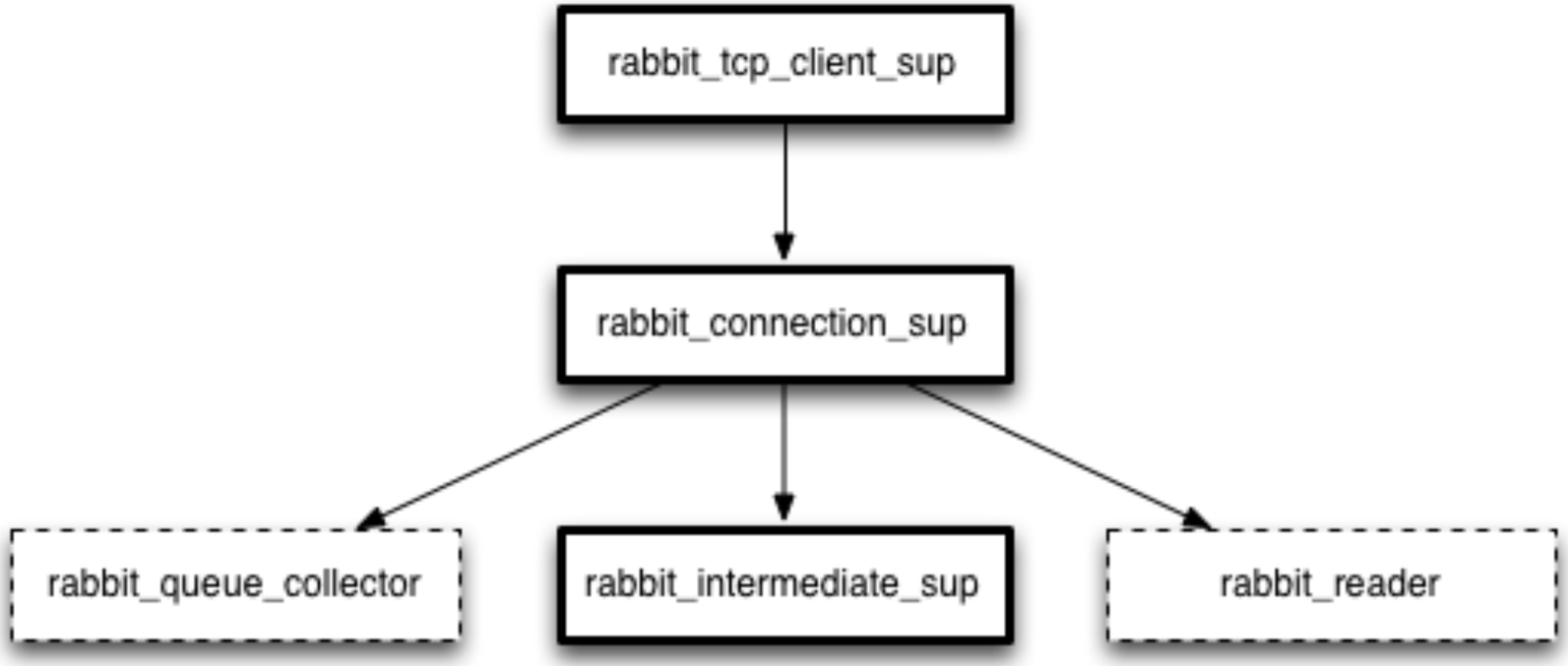
Child Spec - shutdown

- **brutal_kill**: child is terminated immediately.
- **timeout in seconds**: supervisor waits for timeout before terminating children.
- **infinity**: give enough timeout to children to shutdown its own supervision tree.

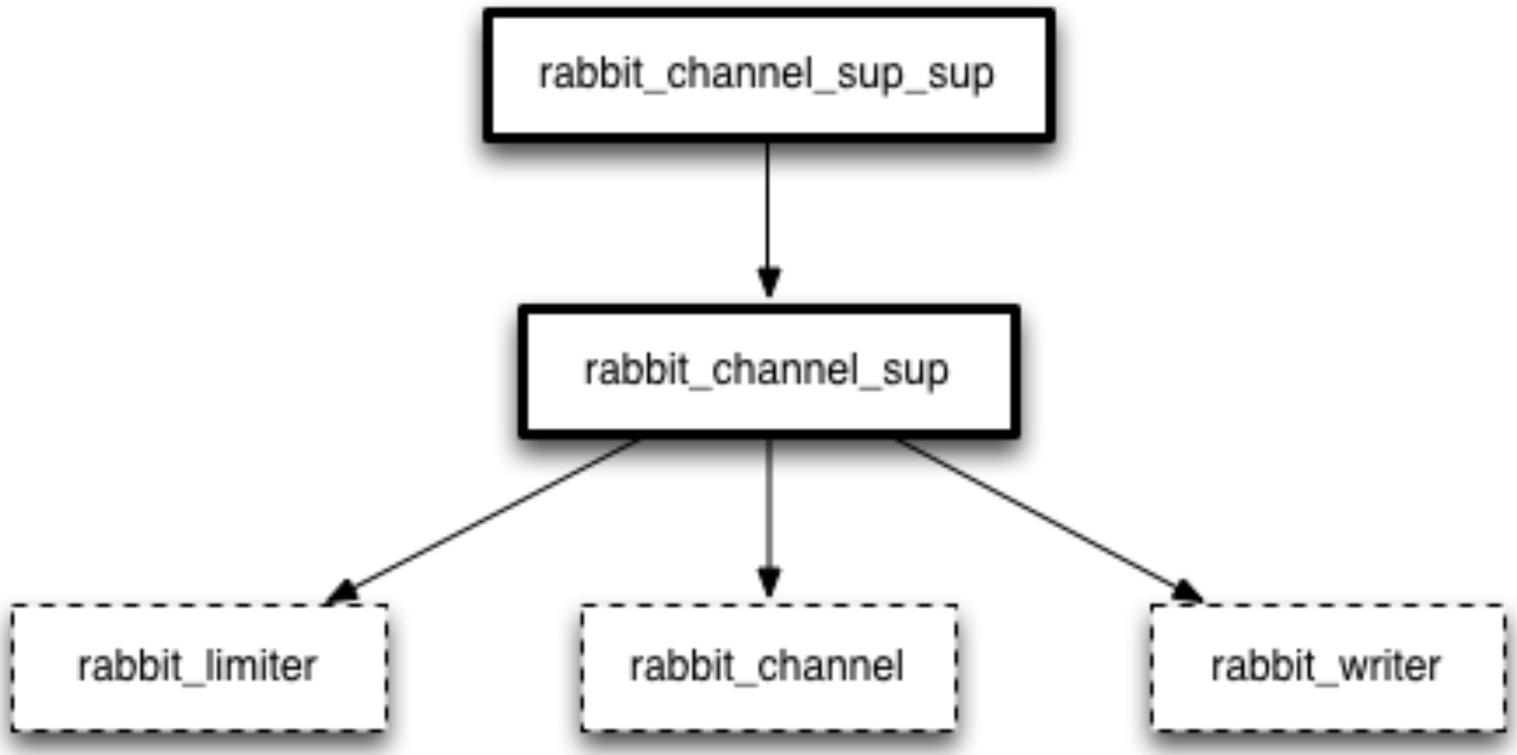
Connection Supervision Tree



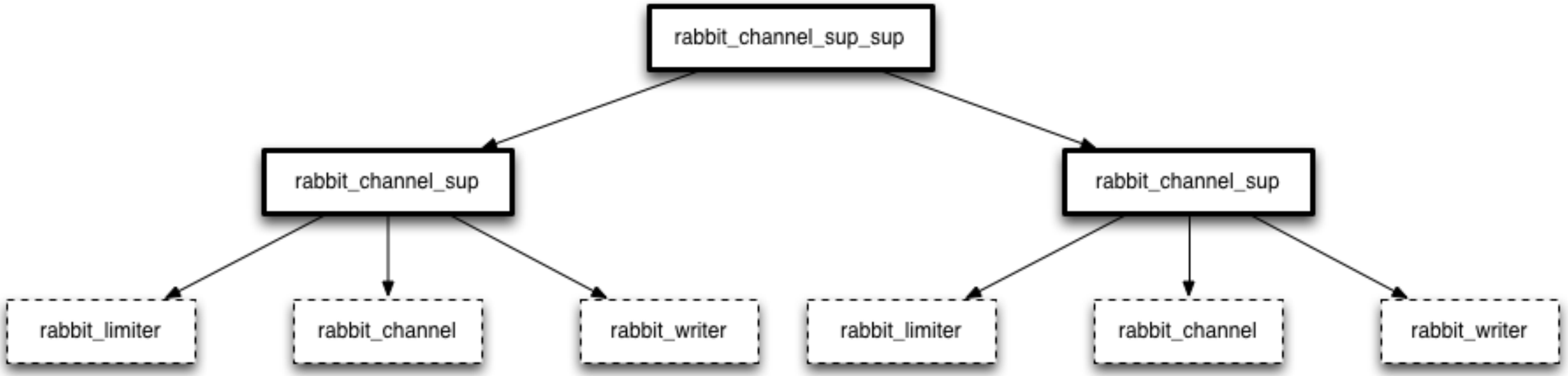
Connection Supervision Tree



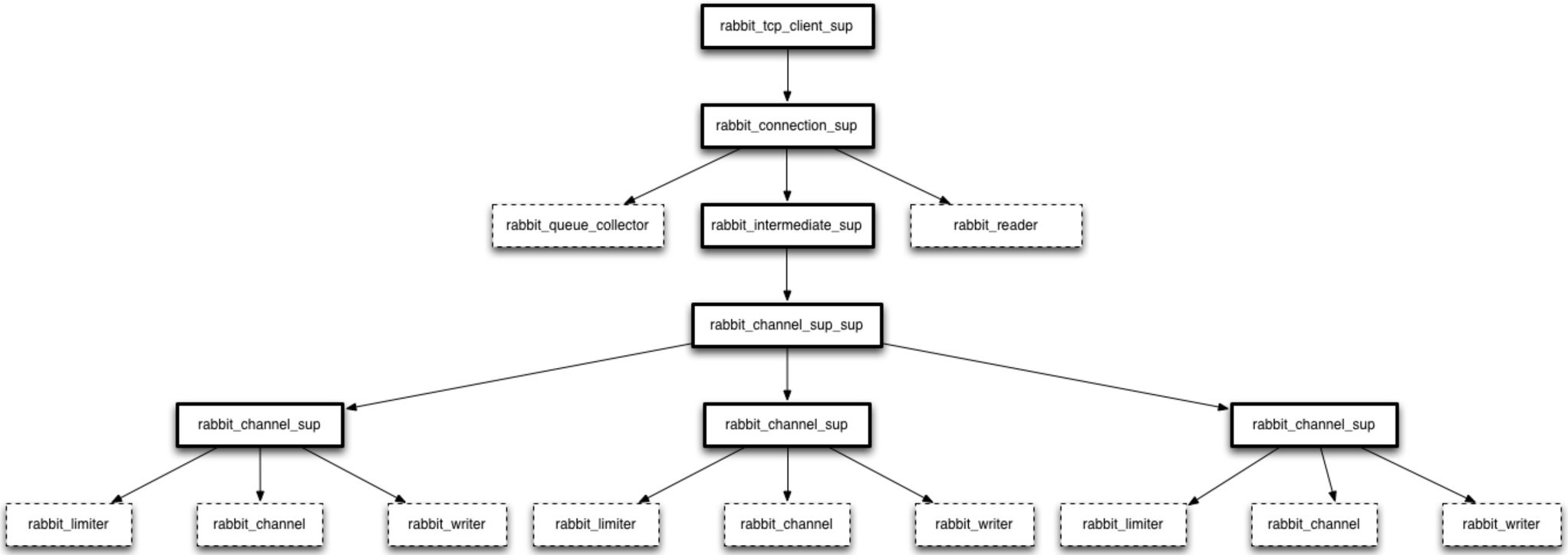
Connection Supervision Tree



Connection Supervision Tree



Connection Supervision Tree



RabbitMQ Behaviours

RabbitMQ Behaviours

- Add a common interface for different components

RabbitMQ Behaviours

- Add a common interface for different components
  - Exchanges

RabbitMQ Behaviours

- Add a common interface for different components
  - Exchanges
  - Queues

RabbitMQ Behaviours

- Add a common interface for different components
  - Exchanges
  - Queues
  - Decorators

RabbitMQ Behaviours

- Add a common interface for different components
  - Exchanges
  - Queues
  - Decorators
  - Authentication methods

RabbitMQ Behaviours

- Add a common interface for different components
  - Exchanges
  - Queues
  - Decorators
  - Authentication methods
- Add extensibility

rabbit_exchange_type

```
-module(rabbit_exchange_type).

-callback description() -> [proplists:property()].

-callback serialise_events() -> boolean().

-callback route(rabbit_types:exchange(), rabbit_types:delivery()) ->
    rabbit_router:match_result().

-callback validate(rabbit_types:exchange()) -> 'ok'.

-callback validate_binding(rabbit_types:exchange(), rabbit_types:binding()) ->
    rabbit_types:ok_or_error({'binding_invalid', string(), [any()]}).
```

rabbit_exchange_type

- You can add your own exchange type via plugins
  - consistent hash exchange
  - random exchange
  - recent history exchange
  - riak exchange

RabbitMQ Behaviours

- rabbit_auth_backend (config)
- rabbit_msg_store_index (config)
- rabbit_backing_queue (config)

RabbitMQ Behaviours

- rabbit_auth_backend (config)
- rabbit_msg_store_index (config)
- rabbit_backing_queue (config)

```
[{rabbit,
  [{auth_backends, [rabbit_auth_backend_http,
                    rabbit_auth_backend_internal]}]
 }].
```

RabbitMQ Behaviours

- rabbit_auth_mechanism (registry)
- rabbit_exchange_decorator (registry)
- rabbit_exchange_type (registry)
- rabbit_mirror_queue_mode (registry)
- rabbit_policy_validator (registry)
- rabbit_queue_decorator (registry)
- rabbit_runtime_parameter (registry)

RabbitMQ Behaviours

- rabbit_auth_mechanism (registry)
- rabbit_exchange_decorator (registry)
- rabbit_exchange_type (registry)
- rabbit_mirror_queue_mode (registry)
- rabbit_policy_validator (registry)
- rabbit_queue_decorator (registry)
- rabbit_runtime_parameter (registry)

```
-rabbit_boot_step({?MODULE,
                   [{description, "exchange type direct"},
                    {mfa,         {rabbit_registry, register,
                                   [exchange, <<"direct">>, ?MODULE]}},
                    {requires,    rabbit_registry},
                    {enables,     kernel_ready}]}).
```

RabbitMQ Federation Plugin

- Message replication across WANs
- Uses an exchange decorator
- Uses a queue decorator
- Uses parameters
- Uses policies

RabbitMQ Federation Plugin

```
rabbitmqctl set_parameter federation-upstream my-upstream \
'{"uri":"amqp://server-name","expires":3600000}'
```

RabbitMQ Federation Plugin

```
rabbitmqctl set_parameter federation-upstream my-upstream \
'{"uri":"amqp://server-name","expires":3600000}'
```

```
rabbitmqctl set_policy federate-me "^amq\." '{"federation-upstream-
set":"all"}'
```

RabbitMQ Federation Plugin

```
-module(rabbit_federation_queue).

-rabbit_boot_step({?MODULE,
                   [{description, "federation queue decorator"},
                    {mfa, {rabbit_registry, register,
                           [queue_decorator, <<"federation">>, ?MODULE]}},
                    {requires, rabbit_registry},
                    {enables, recovery}]}).

-behaviour(rabbit_queue_decorator).
```

RabbitMQ Federation Plugin

```
-module(rabbit_federation_exchange).

-rabbit_boot_step({?MODULE,
                   [{description, "federation exchange decorator"},
                    {mfa, {rabbit_registry, register,
                           [exchange_decorator, <<"federation">>, ?MODULE]}},
                    {requires, rabbit_registry},
                    {enables, recovery}]}).

-behaviour(rabbit_exchange_decorator).
```

Read More Here:

http://www.rabbitmq.com/federation.html

Read More Here:

Making sure the user provided the right behaviour:

http://videlalvaro.github.io/2013/09/rabbitmq-internals-validating-erlang-behaviours.html

How RabbitMQ prevents arbitrary code execution:

http://videlalvaro.github.io/2013/09/rabbitmq-sanitzing-user-input-in-erlang.html

RabbitMQ Boot Step System:

https://github.com/videlalvaro/rabbit-internals/blob/master/rabbit_boot_process.md

RabbitMQ uses Erlang's features to be robust, fault tolerant and very extensible

Go grab the source code!

https://github.com/rabbitmq/rabbitmq-server/

Questions?

Thanks

Alvaro Videla - @old_sound