Building a Distributed Data Ingestion System with RabbitMQ

Alvaro Videla - RabbitMQ

Alvaro Videla

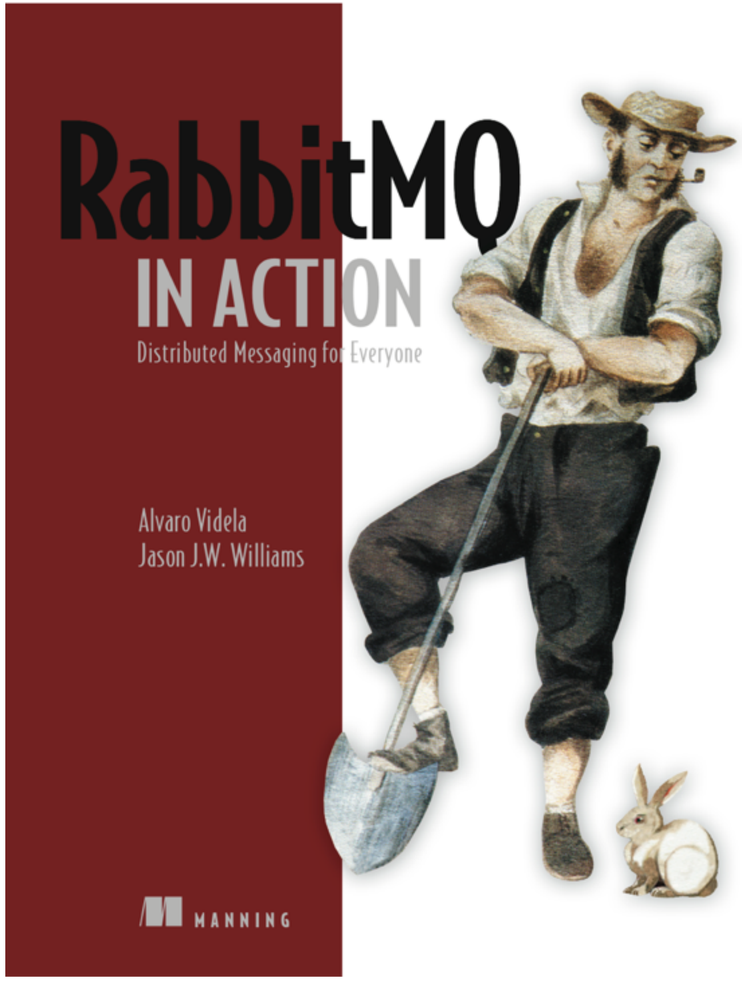
- Developer Advocate at Pivotal / RabbitMQ
- Co-Author of RabbitMQ in Action
- Creator of the RabbitMQ Simulator
- Blogs about RabbitMQ Internals: http://videlalvaro.github.io/internals.html
- @old_sound — alvaro@rabbitmq.com — github.com/videlalvaro

About Me

Co-authored

RabbitMQ in Action

http://bit.ly/rabbitmq



About this Talk

- Exploratory Talk
- A ‘what could be done’ talk instead of ‘this is how you do it’

Agenda

- Intro to RabbitMQ
- The Problem
- Solution Proposal
- Improvements

Intermission

Intermission

History Lessons

The Hungarian Connection

The Hungarian Connection

- I’m from Uruguay

The Hungarian Connection

- I’m from Uruguay
- I live in Switzerland

The Hungarian Connection

- I’m from Uruguay
- I live in Switzerland
- I’m in Hungary right now

WHAT?

The Hungarian Connection

The Hungarian Connection

Switzerland World Cup - 1954

The Hungarian Connection

Switzerland World Cup - 1954

The Hungarian Connection

Switzerland World Cup - 1954

Hungary 4 - Uruguay 2

The Hungarian Connection II

Tivadar Puskás

Telephone Switch Inventor

The Hungarian Connection III

Old Hungarian Alphabet

The Hungarian Connection III

Erlang

```
Pi = fun(P) when pid(P) -> case process_info(P, registered_name) of[] -> case process_info(P, initial_call) of {_, {proc_lib,init_p,5}}-> proc_lib:translate_initial_call(P); {_,
MFA} -> MFA; undefined ->unknown end; {_,Nam} -> Nam; undefined -> unknown end; (P) whenport(P) -> {name,N} = erlang:port_info(P,name), [Hd|_] =string:tokens
(N," "), Tl =lists:reverse(hd(string:tokens(lists:reverse(Hd),"/"))),list_to_atom(Tl); (R) when atom(R) -> R; ({R,Node}) when atom(R),Node == node() -> R; ({R, Node}) when
atom(R), atom(Node) -> {R,Node} end, Ts = fun(Nw) -> {_,{H,M,S}} =calendar:now_to_local_time(Nw), {H,M,S,element(3,Nw)} end, Munge =fun(I) -> case string:str(I,
"Return addr") of 0 -> casestring:str(I, "cp = ") of 0 -> []; _ -> [_, C|_] =string:tokens(I,"()+"), list_to_atom(C) end; _ -> case string:str(I,"erminate process normal") of 0 -> [_,
C|_] =string:tokens(I,"()+"), list_to_atom(C); _ -> [] end end end, Stack= fun(Bin) -> L = string:tokens( binary_to_list( Bin),"\n"),{stack,lists:flatten(lists:map(Munge,L))} end,
Prc = fun(all) ->all; (Pd) when pid(Pd) -> Pd; ({pid,P1,P2}) when integer(P1),integer(P2) -> c:pid(0,P1,P2); (Reg) when atom(Reg) -> case whereis(Reg) of undefined -> exit
({rdbg, no_such_process, Reg}); Pid when pid(Pid) -> Pid end end, MsF = fun(stack, [{Head,Cond,Body}])-> [{Head,Cond,[{message,{process_dump}}|Body]}]; (return,
[{Head,Cond,Body}]) -> [{Head,Cond, [{return_trace}|Body]}]; (Head,[{_,Cond,Body}]) when tuple(Head)-> [{Head,Cond,Body}]; (X,_) ->exit({rdbg,bad_match_spec,X})
end, Ms = fun(Mss) -> lists:foldl(MsF,[{'_',[],[]}], Mss) end, ChkTP = fun({M,F}) when atom(M), atom(F),M/='_', F/='_' -> {{M,F,'_'},[],[global]}; ({M,F,MS}) when atom
(M),atom(F), M/='_', F/='_' -> {{M,F,'_'},Ms(MS),[global]};({M,F,MS,local}) when atom(M), atom(F), M/='_', F/='_' ->{{M,F,'_'},Ms(MS), [local]}; ({M,F,MS,global})
when atom(M), atom(F),M/='_', F/='_' -> {{M,F,'_'},Ms(MS),[global]}; (X) ->exit({rdbg,unrec_trace_pattern,X}) end, ChkTPs = fun(TPs) when list(TPs) -> lists:map(ChkTP,
TPs); (TP) -> [ChkTP(TP)] end, SetTPs =fun({MFA,MS,Fs}) -> erlang:trace_pattern(MFA,MS,Fs) end, DoInitFun =fun(Time) -> erlang:register(rdbg, self()),erlang:
start_timer(Time,self(), {die}),erlang:trace_pattern({'_','_','_'}, false,[local]),erlang:trace_pattern({'_','_','_'},false,[global]) end, InitFun =fun(Time,all,send) -> exit({rdbg,
too_many_processes}); (Time,all,'receive') -> exit({rdbg,too_many_processes}); (Time,P,send) -> DoInitFun(Time), erlang:trace(Prc(P),true,[send,timestamp]); (Time,
P,'receive') ->DoInitFun(Time), erlang:trace(Prc(P),true, ['receive', timestamp]);(Time,P,TPs) -> CTPs = ChkTPs(TPs), DoInitFun(Time),erlang:trace(Prc(P),true, [call,
timestamp]), lists:foreach(SetTPs,CTPs) end, LoopFun = fun(G,N,Out) when N < 1 -> erlang:trace(all, false,[call,send,'receive']), erlang:trace_pattern({'_','_','_'},false,
[local]), erlang:trace_pattern( {'_','_','_'},f alse,[global]), io:fwrite("**rdbg, ~w msgs **~n", [length(Out)]), io:fwrite("~p~n", [lists:reverse(Out)]), io:fwrite
("~p~n", process_info(self(), message_queue_len)]); (G,Cnt,Out) -> case process_info(self(),message_queue_len) of {_,N} when N > 100 -exit({rdbg,msg_queue, N}); _ -> ok
end, receive {timeout,_,{die}} ->G(G,0,Out); {trace_ts,Pid,send,Msg,To,TS} ->G(G,Cnt-1,[{send,Ts(TS), Pi(To),Msg}|Out]); {trace_ts,Pid,'receive',Msg,TS} ->G(G,Cnt-1,
[{'receive',Ts(TS),Msg}|Out]); {trace_ts,Pid,return_from, MFA,V,TS} ->G(G,Cnt-1,[{return,MFA,V}|Out]); {trace_ts,Pid,call,MFA,B,TS} whenbinary(B) -> G(G,Cnt-1,[{Pi
(Pid),Ts(TS), {Stack(B),MFA}}|Out]);{trace_ts,Pid,call,MFA,TS} -> G(G,Cnt-1,[{Pi(Pid),Ts(TS),MFA}|Out])end end, Rdbg = fun(Time,Msgs,Proc, Trc) when integer
(Time),integer(Msgs) -> Start = fun() ->InitFun(Time,Proc,Trc), LoopFun(LoopFun,Msgs,[]) end, erlang:spawn_link(Start) end.
```

The Hungarian Connection III

As Neumann János Lajos said

The Hungarian Connection III

As Neumann János Lajos said

Use Erlang

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

Community Plugins

http://www.rabbitmq.com/community-plugins.html

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

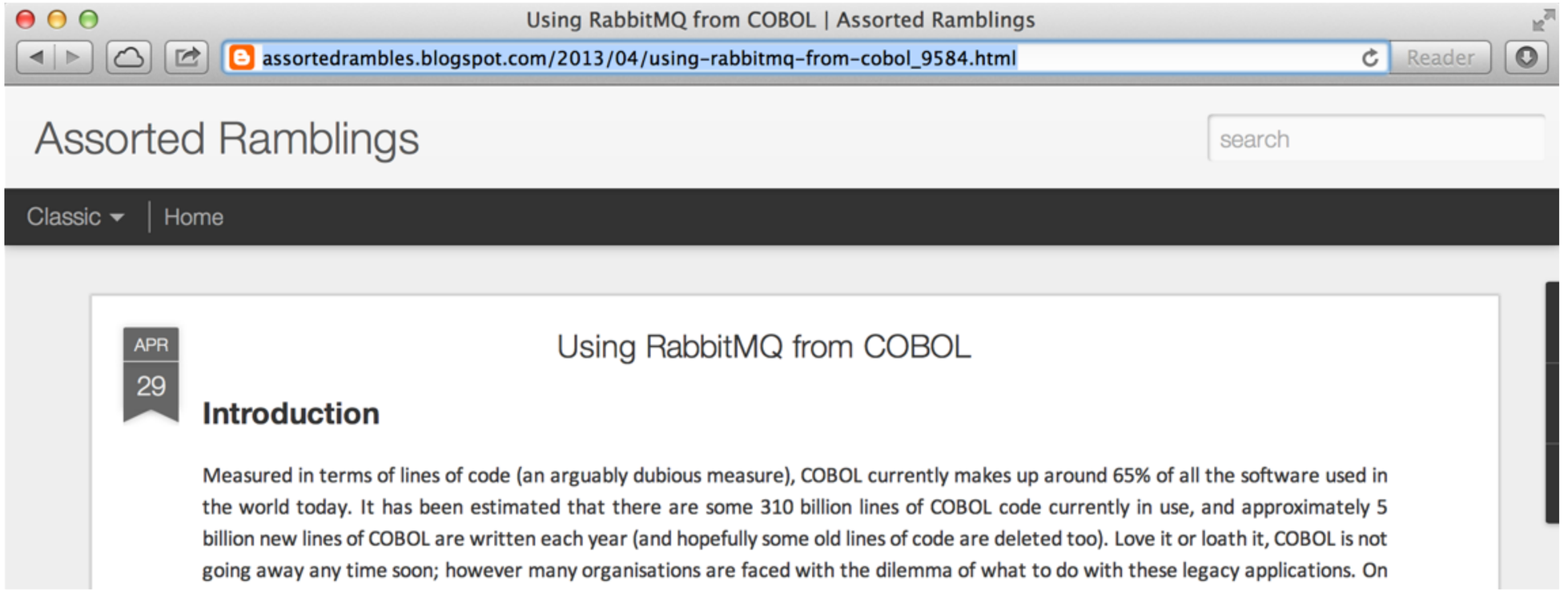
Polyglot

- 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
- Telefonica

Some users of RabbitMQ

- Instagram
- Indeed.com
- Telefonica
- Mercado Libre

Some users of RabbitMQ

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

Some users of RabbitMQ

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

The New York Times on RabbitMQ

This architecture - Fabrik - has dozens of RabbitMQ instances spread across 6 AWS zones in Oregon and Dublin.

Upon launch today, the system autoscaled to ~500,000 users. Connection times remained flat at ~200ms.

http://lists.rabbitmq.com/pipermail/rabbitmq-discuss/2014-January/032943.html

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

Unix - Mac - Windows

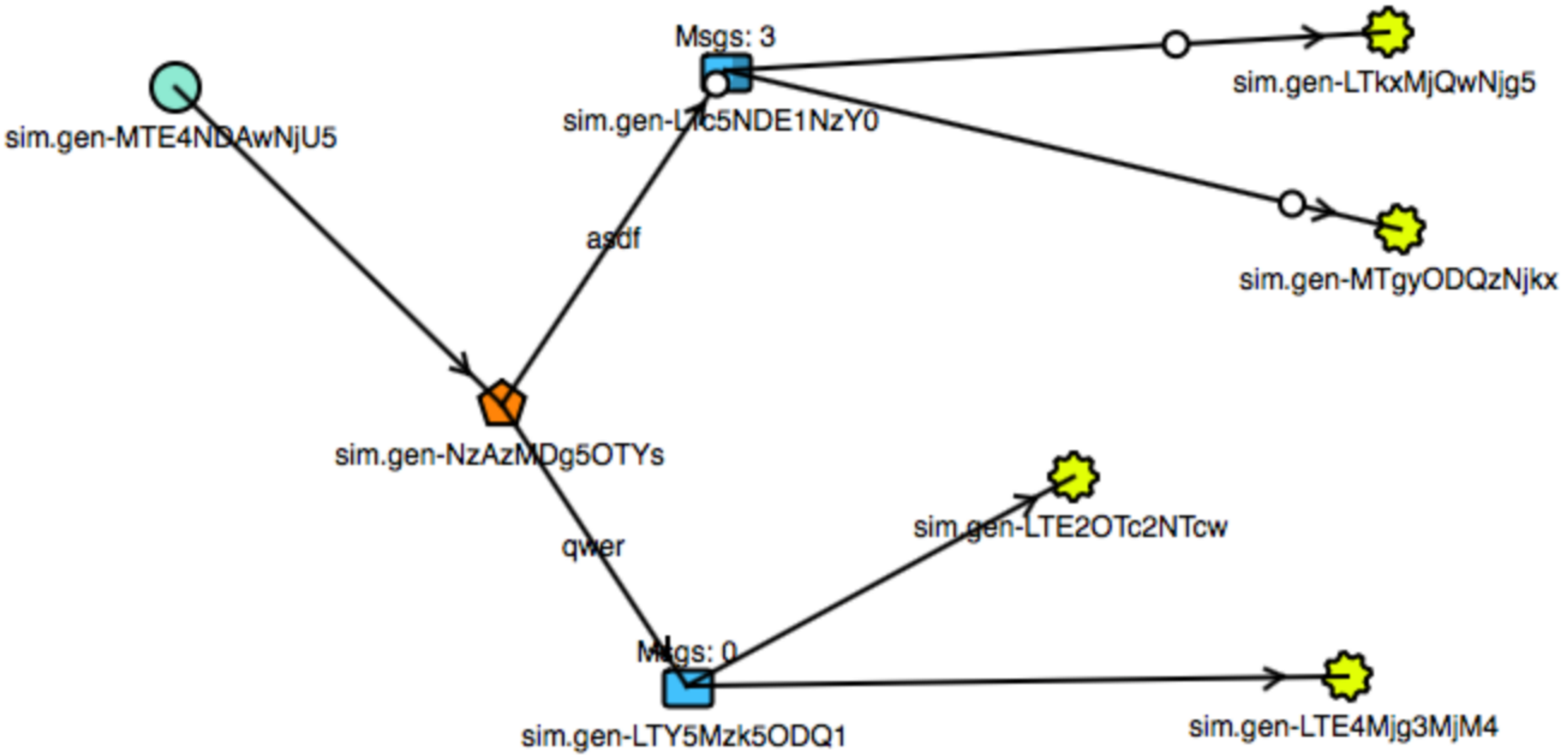
Messaging with RabbitMQ

A demo with the RabbitMQ Simulator

https://github.com/RabbitMQSimulator/RabbitMQSimulator

http://tryrabbitmq.com

RabbitMQ Simulator



The Problem

Distributed Application

Distributed Application

Data Producer

Obtain a Channel

```
ConnectionFactory factory = new ConnectionFactory();
factory.setHost("localhost");

Connection connection = factory.newConnection();

Channel channel = connection.createChannel();
```

Data Producer

Declare an Exchange

```
channel.exchangeDeclare(EXCHANGE_NAME, "direct", true);
```

Data Producer

Publish a message

```
String message = "Hello Federation!";
channel.basicPublish(EXCHANGE_NAME, "", null,
                                message.getBytes());
```

Data Consumer

Obtain a Channel

```
ConnectionFactory factory = new ConnectionFactory();
factory.setHost("localhost");

Connection connection = factory.newConnection();

Channel channel = connection.createChannel();
```

Data Consumer

Declare Queue and bind it

```
channel.queueDeclare(QUEUE_NAME, true, false, false,
                     null);
channel.exchangeDeclare(EXCHANGE_NAME, "direct", true);
channel.queueBind(QUEUE_NAME, EXCHANGE_NAME, "");
```

Data Consumer

Start a consumer

```
QueueingConsumer consumer = new
QueueingConsumer(channel);
channel.basicConsume(QUEUE_NAME, false, consumer);
```

Data Consumer

Process messages

```
while (true) {
   QueueingConsumer.Delivery delivery =
      consumer.nextDelivery();

   String message = new String(delivery.getBody());
   System.out.println("Received '" + message + "'");

   channel.basicAck(
      delivery.getEnvelope().
      getDeliveryTag(), false);
}
```

Ad-hoc solution

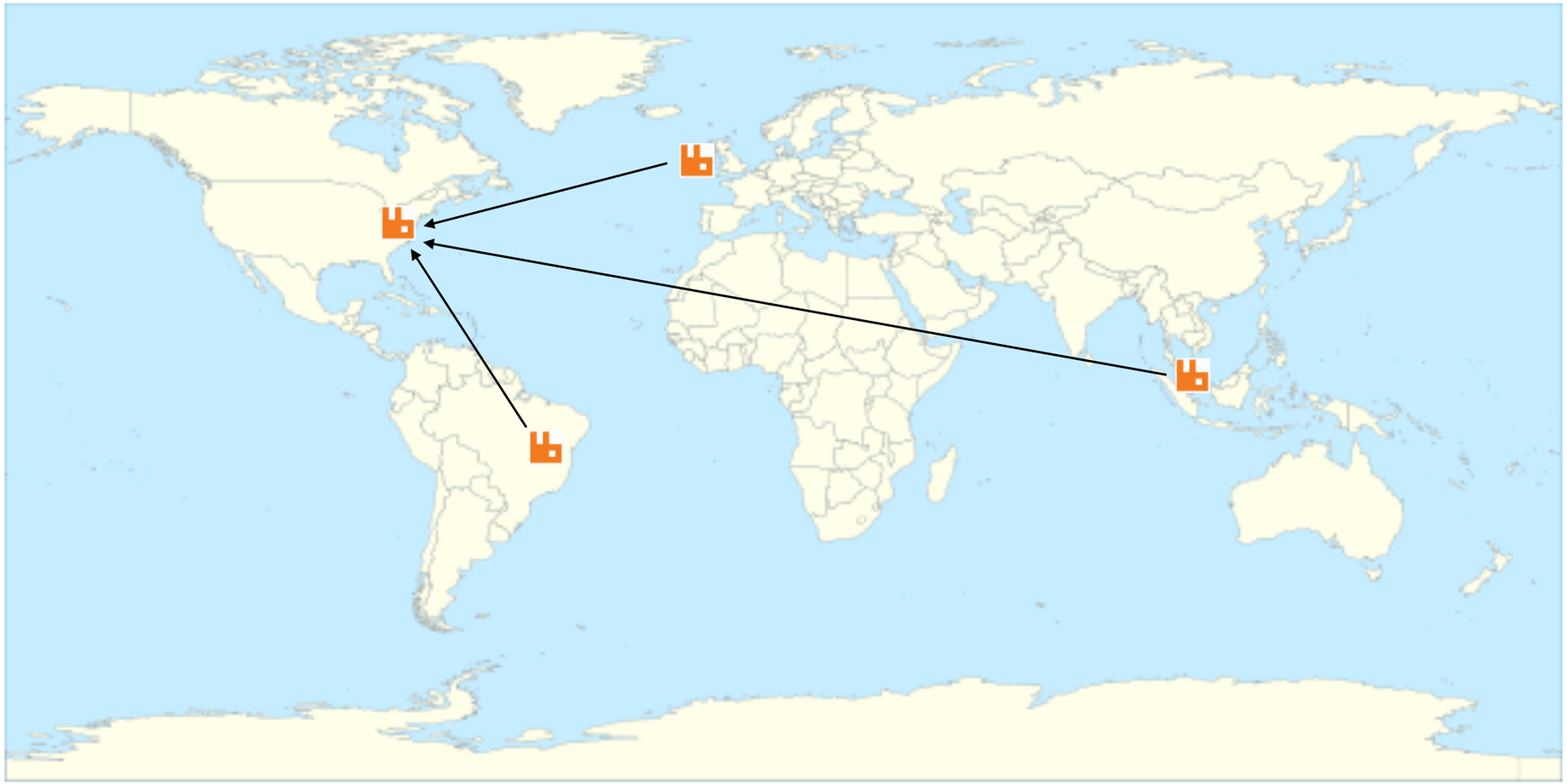
A process that replicates data to the remote server

Possible issues

- Remote server is offline
  - Prevent unbounded local buffers
  - Prevent message loss
- Prevent unnecessary message replication
  - No need for those messages on remote server
  - Messages that became stale

Can we do better?

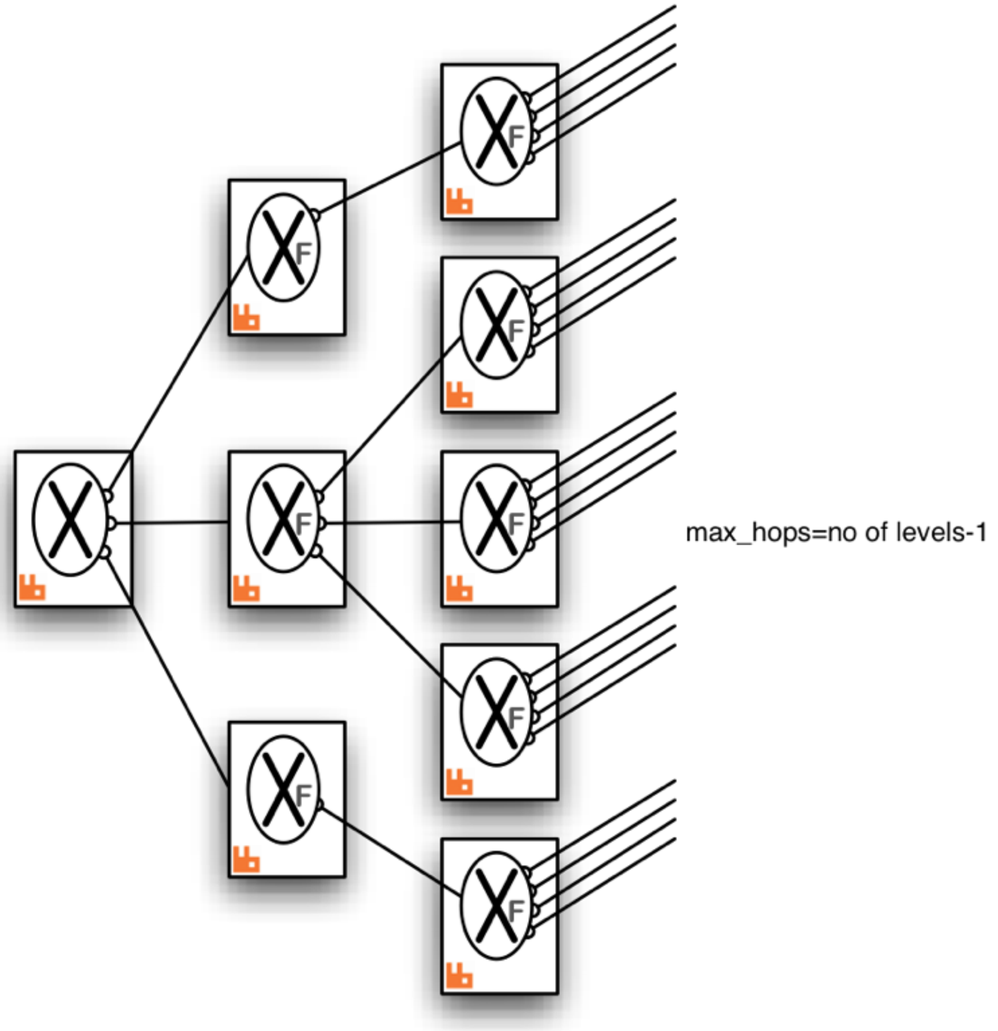
RabbitMQ Federation



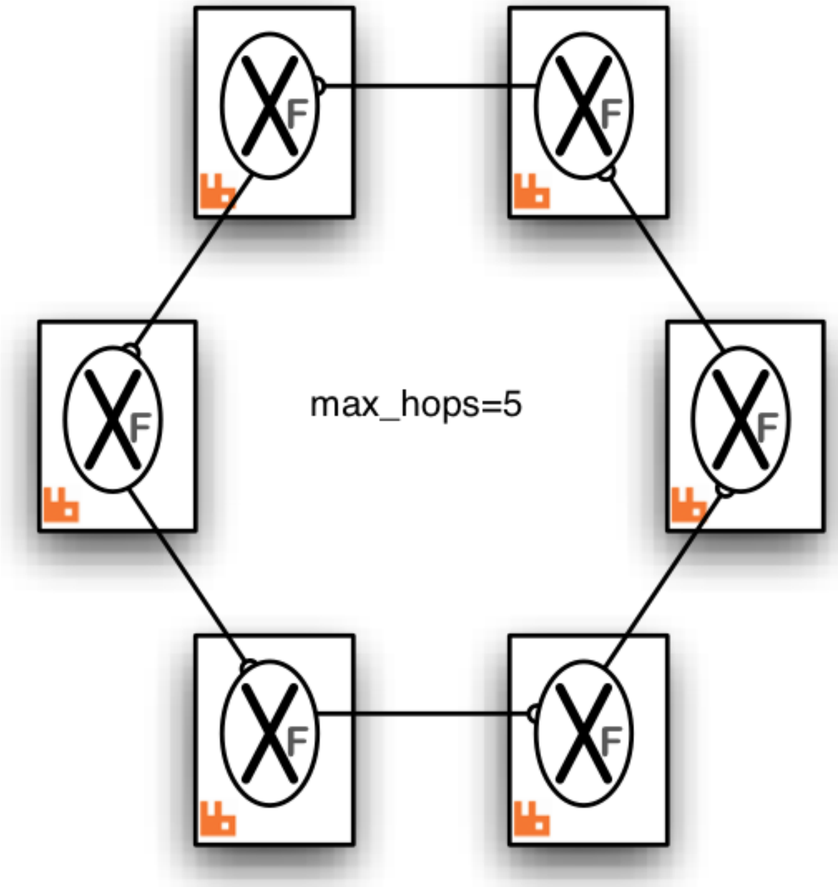
RabbitMQ Federation

- Supports replication across different administrative domains
- Supports mix of Erlang and RabbitMQ versions
- Supports Network Partitions
- Specificity - not everything has to be federated

RabbitMQ Federation



RabbitMQ Federation



RabbitMQ Federation

RabbitMQ Federation

- It’s a RabbitMQ **Plugin**

RabbitMQ Federation

- It’s a RabbitMQ **Plugin**
- Internally uses **Queues** and **Exchanges Decorators**

RabbitMQ Federation

- It’s a RabbitMQ **Plugin**
- Internally uses **Queues** and **Exchanges Decorators**
- Managed using **Parameters** and **Policies**

Enabling the Plugin

```
rabbitmq-plugins enable rabbitmq_federation
```

Enabling the Plugin

```
rabbitmq-plugins enable rabbitmq_federation

rabbitmq-plugins enable rabbitmq_federation_management
```

Federating an Exchange

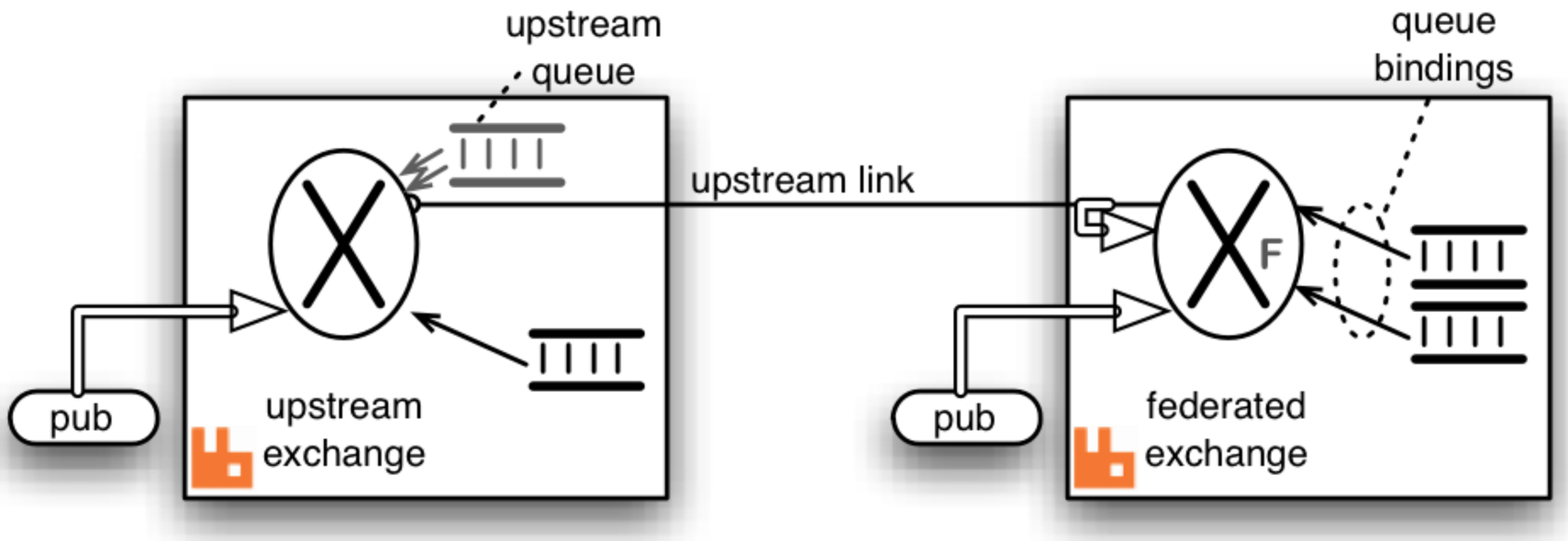
```
rabbitmqctl set_parameter federation-upstream my-upstream \
'{“uri":"amqp://server-name","expires":3600000}'
```

Federating an Exchange

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

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

Federating an Exchange



Configuring Federation

Config Options

```
rabbitmqctl set_parameter federation-upstream \
name ‘json-object’
```

Config Options

```
rabbitmqctl set_parameter federation-upstream \
name ‘json-object’

json-object: {
   ‘uri’: ‘amqp://server-name/’,
   ‘prefetch-count’: 1000,
   ‘reconnect-delay’: 1,
   ‘ack-mode’: on-confirm
}
```

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

Prevent unbound buffers

```
expires: N // ms.
message-ttl: N // ms.
```

Prevent message forwarding

```
max-hops: N
```

Speed vs No Message Loss

```
ack-mode: on-confirm
ack-mode: on-publish
  ack-mode: no-ack
```

AMQP URI:

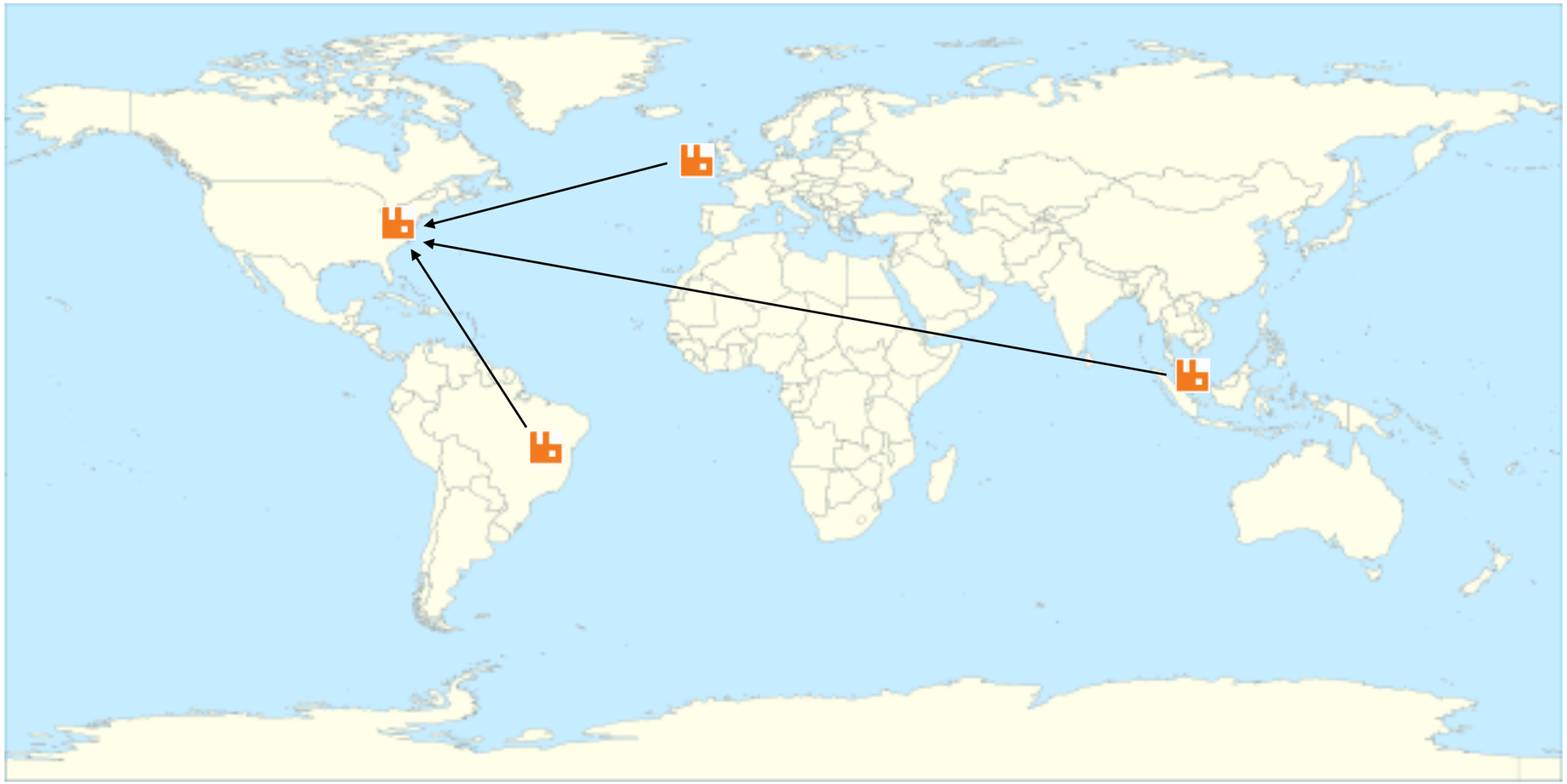
```
amqp://user:pass@host:10000/vhost
```

http://www.rabbitmq.com/uri-spec.html

Config can be applied via

- CLI using **`rabbitmqctl`**
- HTTP API
- RabbitMQ Management Interface

RabbitMQ Federation



Scaling the Setup

The Problem

The Problem

- Queues contents live in the node where the Queue was declared

The Problem

- Queues contents live in the node where the Queue was declared
- A cluster can access the queue from every connected node

The Problem

- Queues contents live in the node where the Queue was declared
- A cluster can access the queue from every connected node
- Queues are an Erlang process (tied to one core)

The Problem

- Queues contents live in the node where the Queue was declared
- A cluster can access the queue from every connected node
- Queues are an Erlang process (tied to one core)
- Adding more nodes doesn’t really help

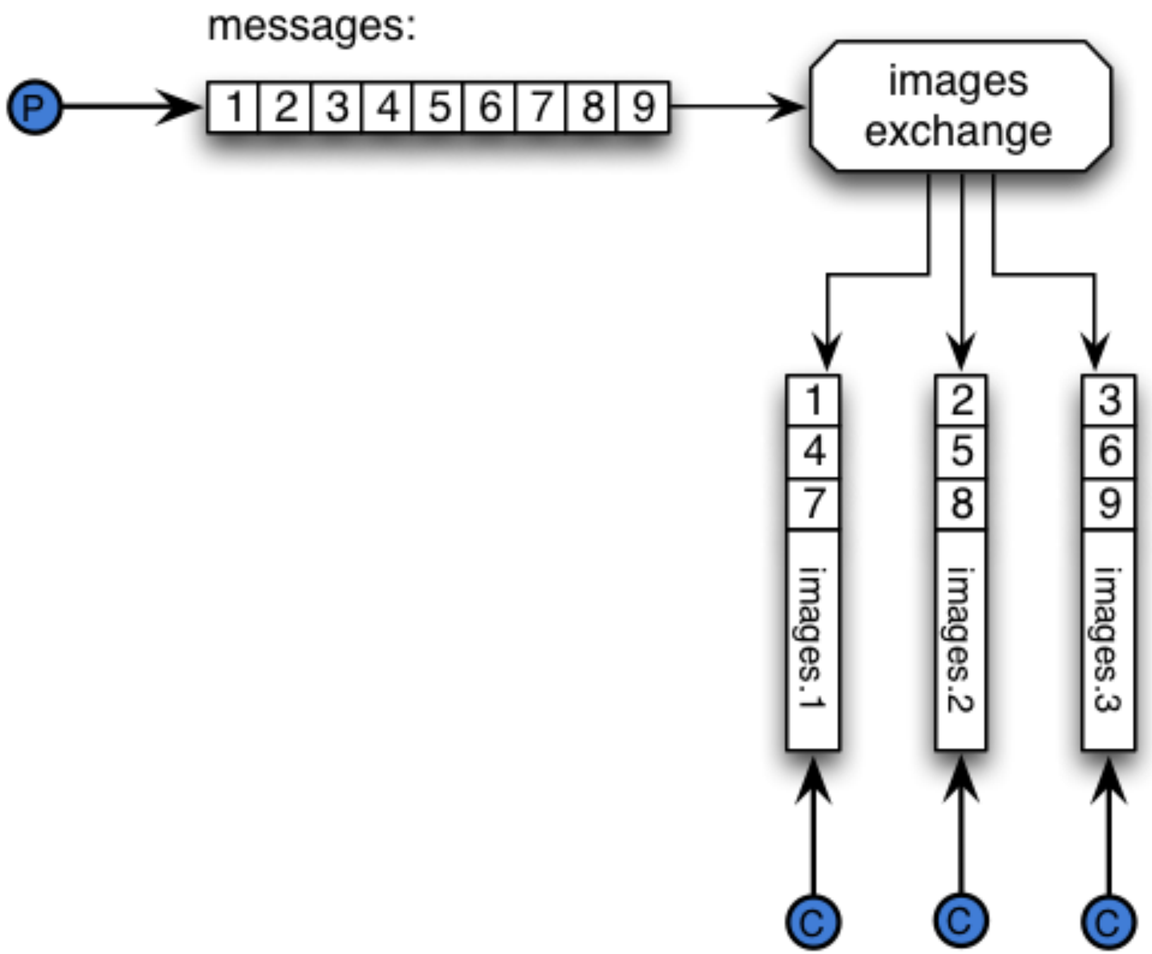
Enter Sharded Queues

Enter Sharded Queues

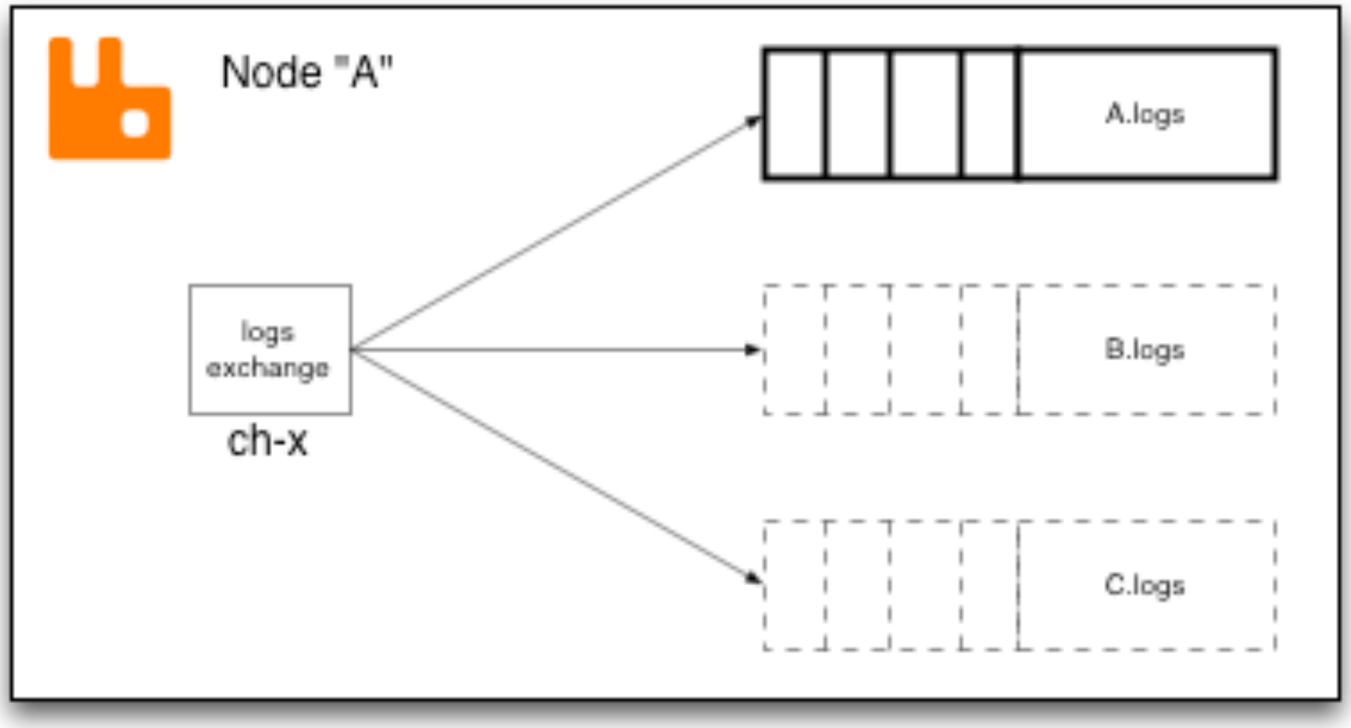
Pieces of the Puzzle

- modulo hash exchange (consistent hash works as well)
- good ol’ queues

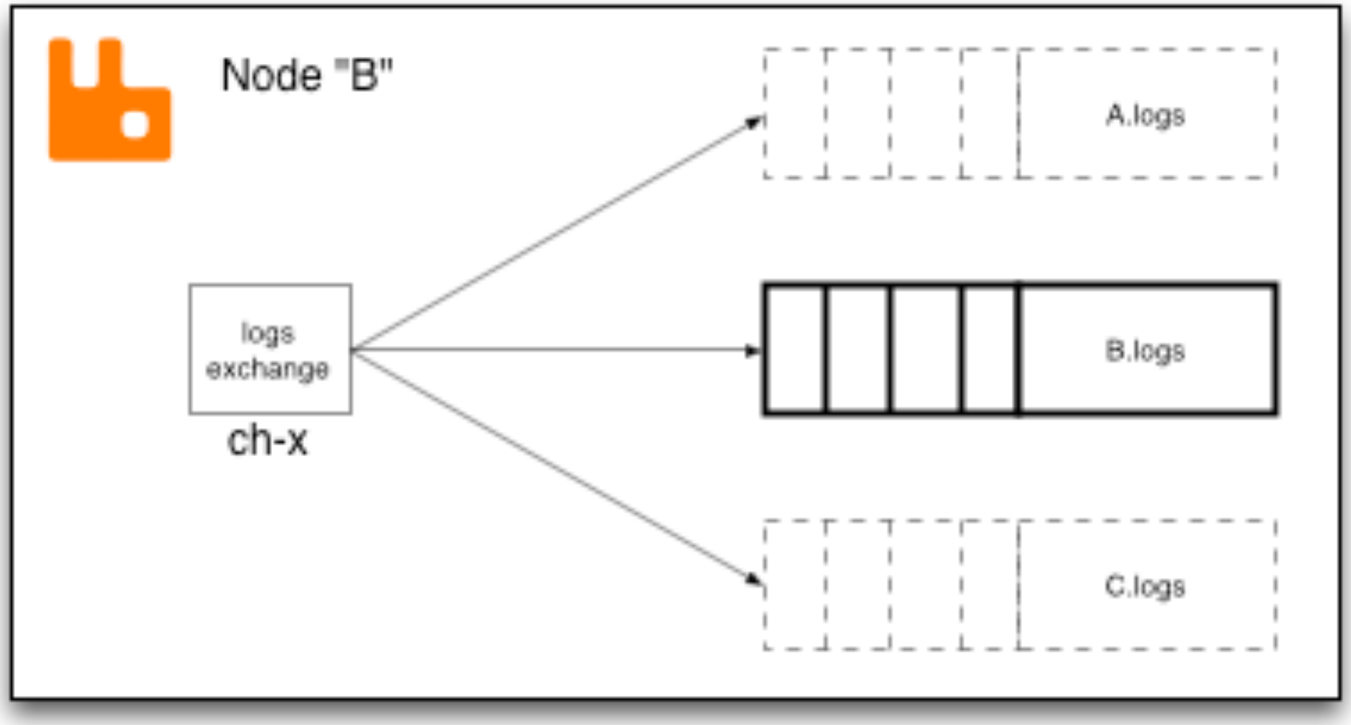
Sharded Queues



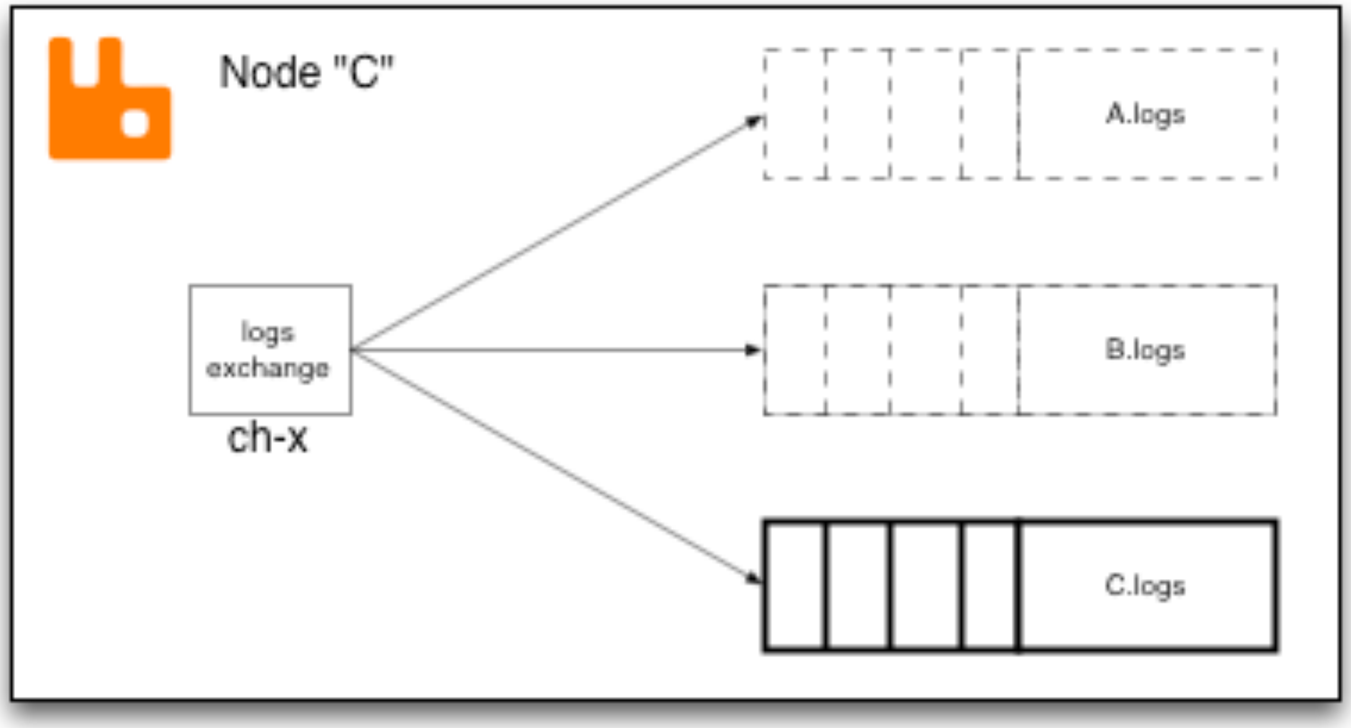
Sharded Queues



Sharded Queues



Sharded Queues



Sharded Queues

- Declare Queues with name: `nodename.queuename.index`

Sharded Queues

- Declare Queues with name: `nodename.queuename.index`
- Bind the queues to a consistent hash exchange

Sharded Queues

- Declare Queues with name: `nodename.queuename.index`
- Bind the queues to a partitioner exchange
- Transparent to the consumer (virtual queue name)

We need more scale!

Federated Queues

Federated Queues

- Load-balance messages across federated queues
- Only moves messages when needed

Federating a Queue

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

Federating a Queue

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

rabbitmqctl set_policy --apply-to queues federate-me "^images\." \
'{"federation-upstream-set":"all"}'
```

With RabbitMQ we can

With RabbitMQ we can

- Ingest data using various protocols: AMQP, MQTT and STOMP

With RabbitMQ we can

- Ingest data using various protocols: AMQP, MQTT and STOMP
- Distribute that data globally using Federation

With RabbitMQ we can

- Ingest data using various protocols: AMQP, MQTT and STOMP
- Distribute that data globally using Federation
- Scale up using Sharding

With RabbitMQ we can

- Ingest data using various protocols: AMQP, MQTT and STOMP
- Distribute that data globally using Federation
- Scale up using Sharding
- Load balance consumers with Federated Queues

Credits

world map: wikipedia.org

federation diagrams: rabbitmq.com

Questions?

Thanks

Alvaro Videla - @old_sound