MESSAGING WITH RABBITMQ

Marcus Kröger | IRIAN

@subft1

Bernd Bohmann | IRIAN

@bbohmann

RabbitMQ
Messaging that just works

AGENDA

Agenda

- Basics
- Exchanges
- Extensions
- Policy
- Flow Control
- Clustering
- Plugins
- Experiences

BASICS

Messaging

- Decoupling
- Validation
- Transformation
- Routing

Enterprise Integration Patterns

RabbitMQ

- Mozilla Public License
- Advanced Message Queuing Protocol(AMQP)
- Erlang/OTP
- since 2007
- Pivotal

AMQP

- Transport
- Functional

AMQP

- Transport
  - Binary encoding
  - asynchronous
  - Channel
  - Frame
- Functional
  - Exchange
  - Queue
  - Binding

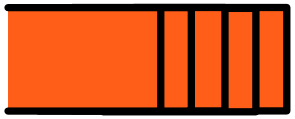
Message

- Properties
- Header
- Body

Message Properties

- Content type
- Content encoding
- Routing key
- Delivery mode (persistent/not)
- Priority
- Correlation id
- Reply to
- Expiration period
- Id
- Publishing timestamp
- Type
- User id
- Application id

Queue



- FIFO
- Persistence
- Durable
- Exclusive
- Auto delete
- *Message TTL*
- *Expire*
- *Length*
- *Dead letter exchange*

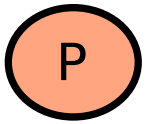
Exchange

- Direct
- Fanout
- Topic
- Headers
- Custom
- Options
  - Durable
  - *Auto delete*
  - *Internal*
  - *Alternate Exchange*

Binding

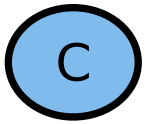
- Exchange
- Queue
- Routing Key

Producer



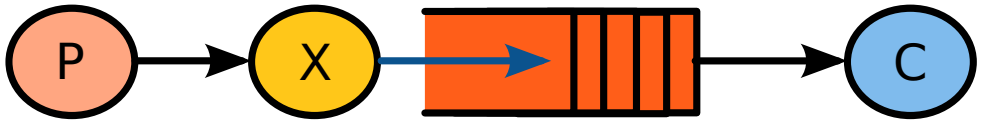
- basic.publish

Consumer



- `basic.consume`
- `basic.get`
- `basic.ack`
- `basic.reject`
- *`basic.nack`*

Summary



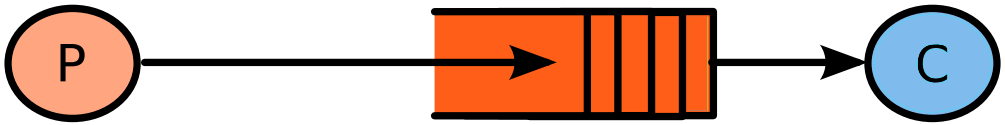
Summary Code

```
channel.queueDeclare(QUEUE_NAME,
       false, // durable
       false, // exclusive
       false, // auto delete
       null // properties);
channel.exchangeDeclare(EXCHANGE_NAME, "fanout");
channel.queueBind(QUEUE_NAME, EXCHANGE_NAME, ROUTING_KEY);
channel.basicPublish(EXCHANGE_NAME,
       QUEUE_NAME, // routing key
       null, // properties
       "Hello World!".getBytes());
```

```
QueueingConsumer consumer = new QueueingConsumer(channel);
channel.basicConsume(QUEUE_NAME,
       true,//
       consumer);
while (true) {
    QueueingConsumer.Delivery delivery = consumer.nextDelivery();
}
```

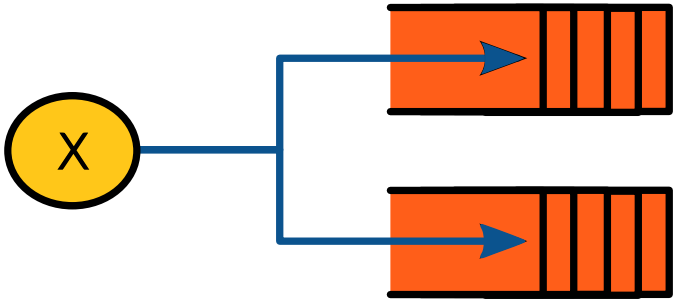
EXCHANGES

Default



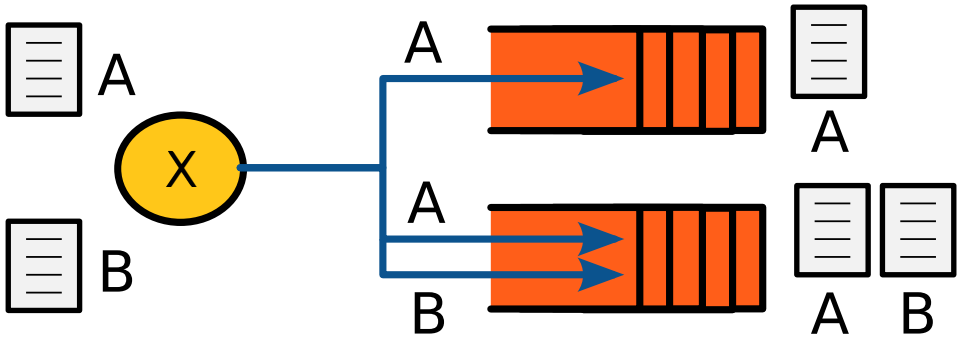

```
channel.queueDeclare(QUEUE_NAME, false, false, false, null);
channel.basicPublish("", // default exchange
                     QUEUE_NAME, // routing key
                     null, // properties
                     "Hello World!".getBytes());
```

Fanout




```
channel.exchangeDeclare(EXCHANGE_NAME, "fanout");
channel.basicPublish(EXCHANGE_NAME, // exchange
                     "", // routing key
                     null, // properties
                     "Hello World!".getBytes());
```

Direct

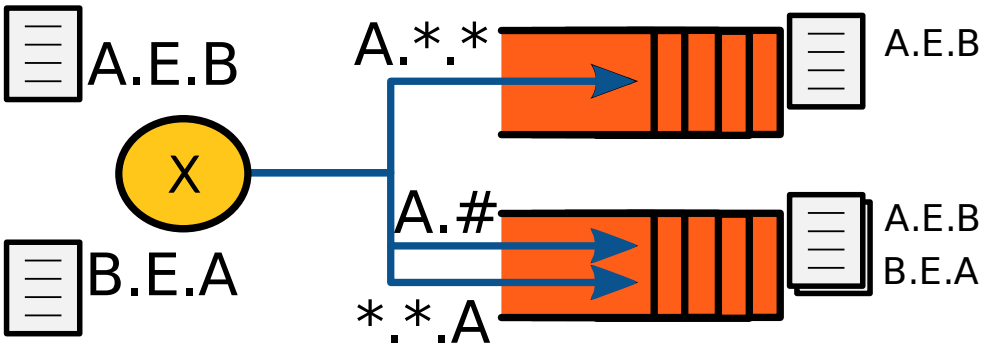


- exactly matches

```
channel.exchangeDeclare(EXCHANGE_NAME, "direct");
String queueName = channel.queueDeclare().getQueue();
channel.queueBind(queueName, EXCHANGE_NAME, "A");
channel.queueBind(queueName, EXCHANGE_NAME, "B");
```

```
channel.basicPublish(EXCHANGE_NAME, ROUTING_KEY,
                     null, MESSAGE.getBytes());
```

Topic



- words, delimited by dots
- * (star) can substitute for exactly one word
- # (hash) can substitute for zero or more words.

Header

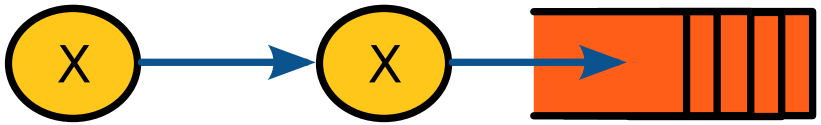
- instead of routing key on any header
- `"x-match","any"` or `"x-match","all"`

EXTENSIONS

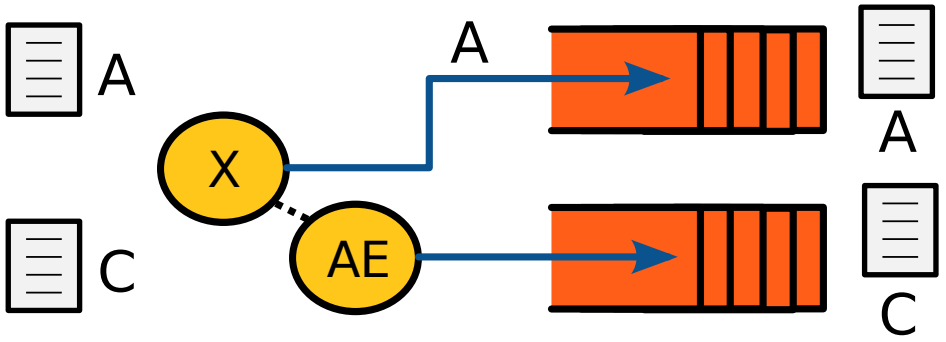
RabbitMQ Extensions

- Confirms
- Blocked Connection Notifications
- Consumer Cancellation Notifications
- basic.nack
- Consumer Priorities
- Exchange Exchange Binding
- Alternate Exchange
- Sender-selected Distribution
- TTL
- Queue length
- Dead Letter Exchange

Exchange Exchange Binding



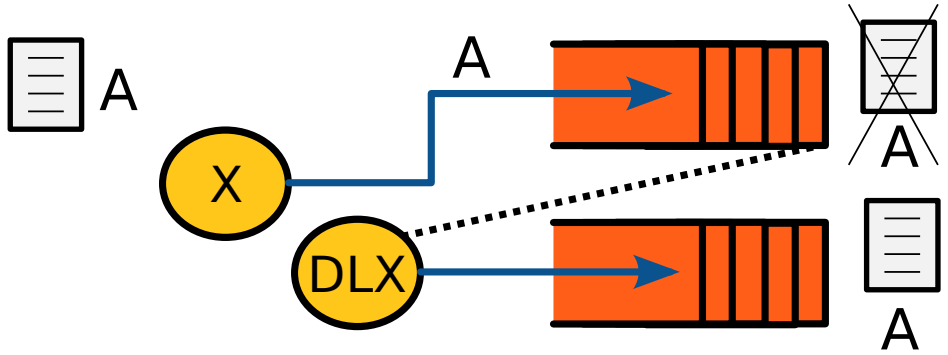
Alternate Exchange



- No matching routing keys
- No matching headers
- No queue bound
- vs mandatory publish

```
Map<~,~> args = new HashMap<~,~>();
args.put("alternate-exchange", "my-ae");
channel.exchangeDeclare("my-direct", "direct", false, false, args);
```

Dead Letter Exchange



- `basic.reject` || `basic.nack` with `requeue=false`
- TTL
- Queue length

```
Map<~,~> args = new HashMap<~,~>();
args.put("x-dead-letter-exchange", "some.exchange.name");
channel.queueDeclare("myqueue", false, false, false, args);
```

POLICY

Policy

- Feature
  - Federation
  - Mirrored queue
  - Alternate exchange
  - Dead lettering
  - Queue TTL
  - Queue length
- Feels more natural than configuring by arguments
- Not everything is configurable via policy
- Still work in progress...

Example Policy

```
$ rabbitmqctl set_policy ha-all '^(?!amq\.).*' '{"ha-mode":"all"}'
```

FLOW CONTROL

Flow Control

- Per-Connection
- Memory-Based
- Disk-Based

CLUSTERING

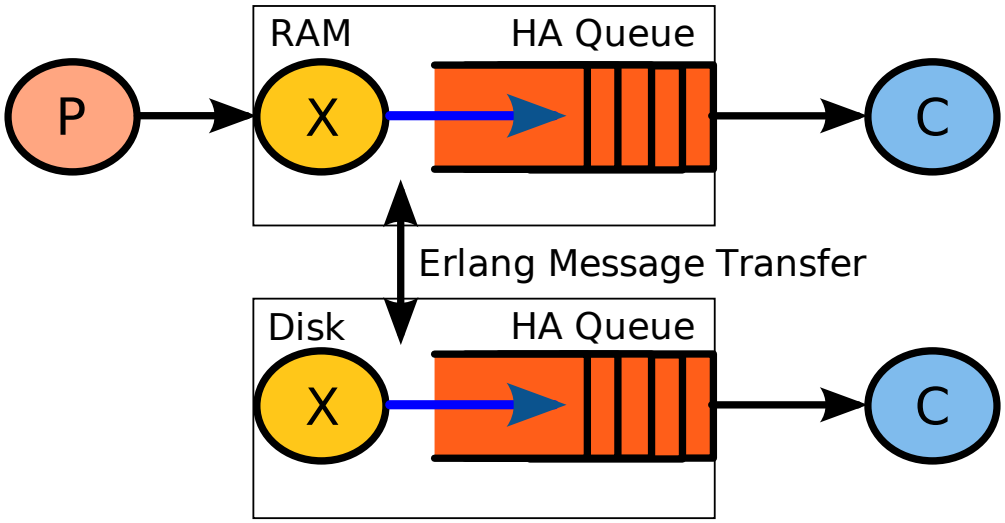
Nodes

- Disk
  - State in memory and on disk
- RAM
- Both persist messages in queues if necessary

Cluster

- Does not tolerate network partitions well
- At least one disk node
- Must have the same erlang cookie
- Consumer should be able to reconnect
- Configure Mirrored Queues
- No specific Cluster configuration required for Exchanges and Bindings

Overview



Cluster Setup

Go to Cluster Guide

PLUGINS

Plugins

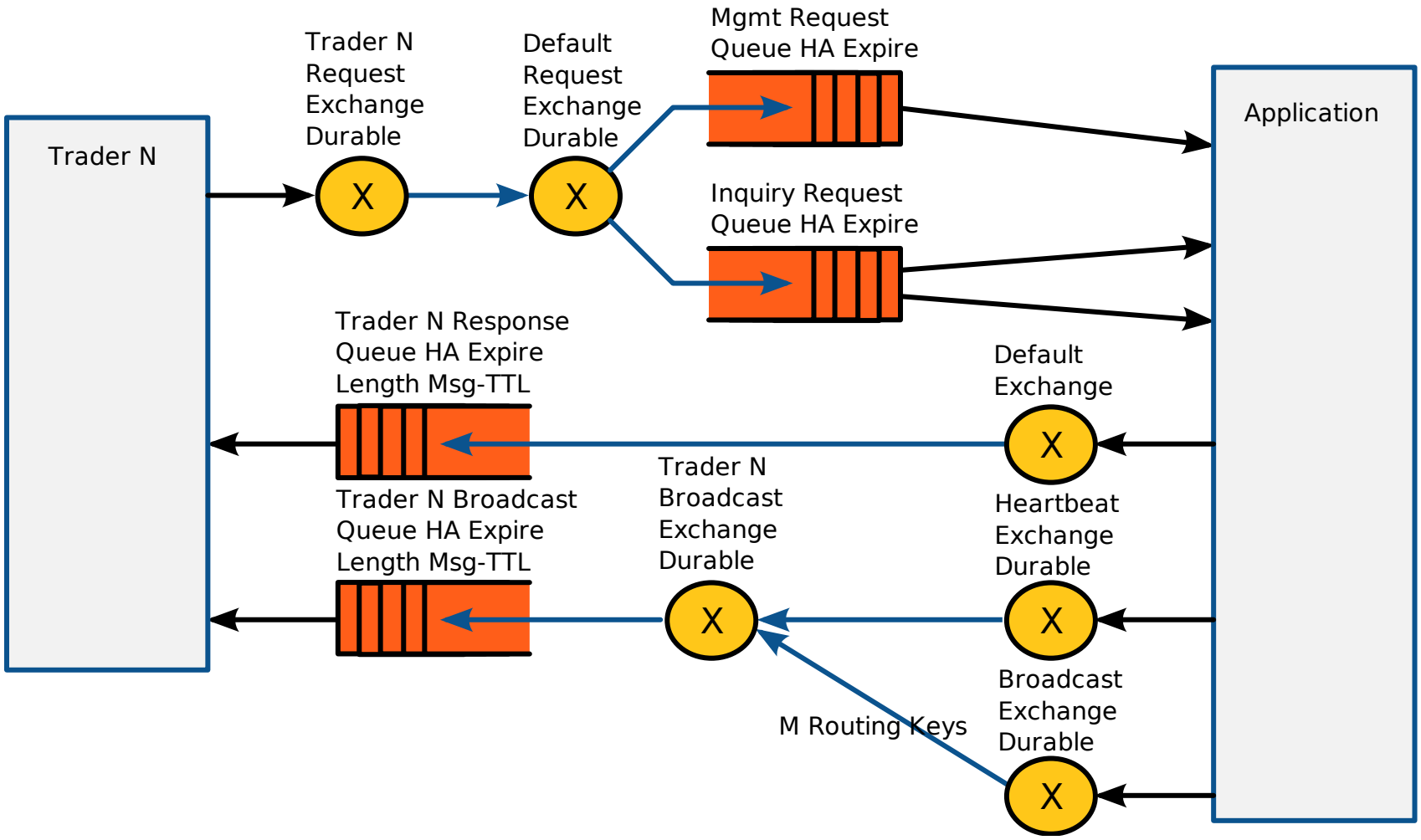
- Management Plugin
- Ldap
- SSL
- Shovel
- Federation
- Stomp
- rabbitmq_tracing

EXPERIENCES

Our Experiences

- Connections Channel Synchronisation
- Message Size matters
- Queues without Consumer
- Message TTL
- Mandatory Flag
- Private Broadcasts Queues
- Exchange Exchange Binding
- Routing Key Definition
- Topic Exchange with Wildcards
- X-expire Policy Declaration
- 3 Nodes Cluster (split brain)
- Nack Messages stay in Queue

Current Architecture



THE END