AMQP交换机和队列

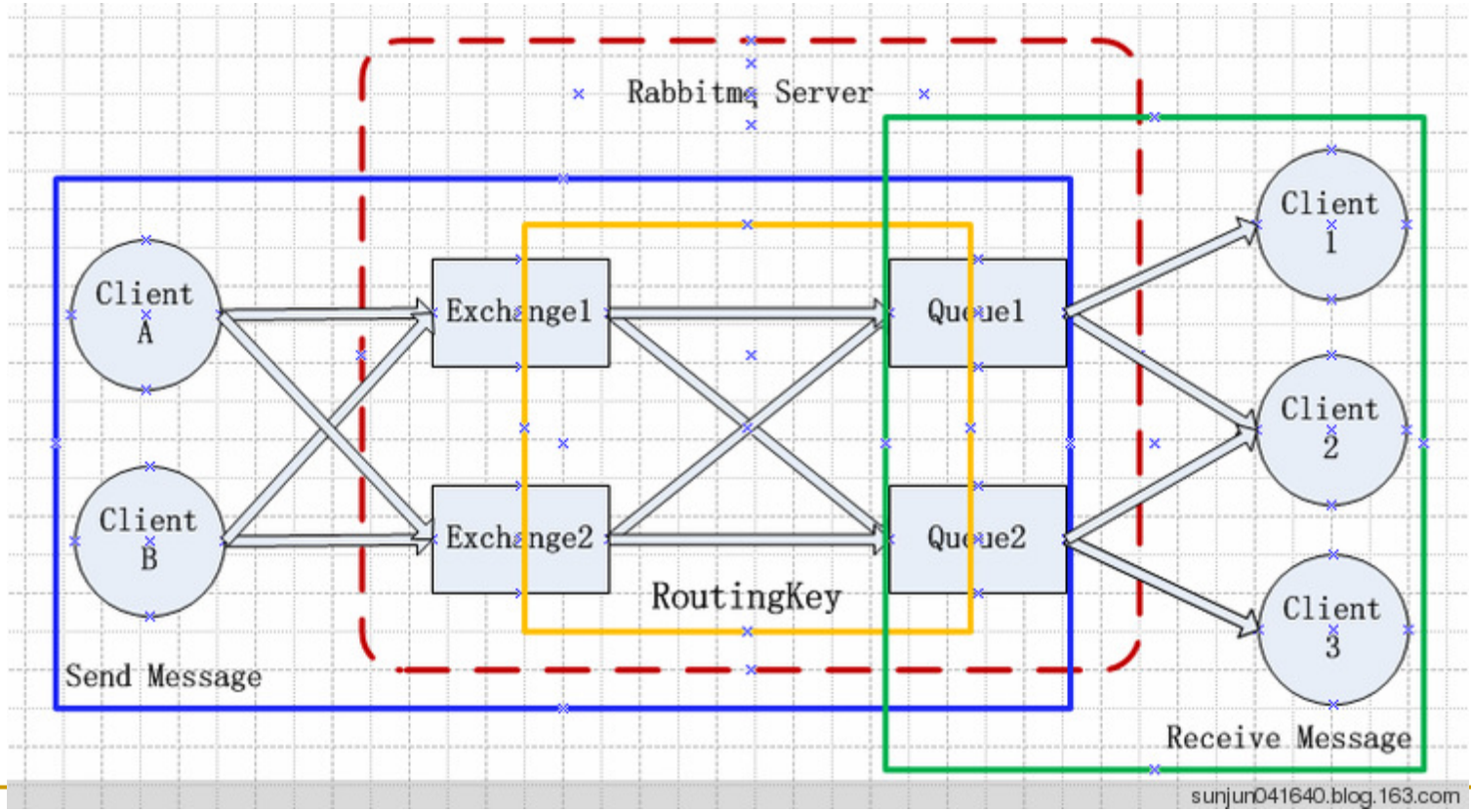
风河 www.nsbeta.info

AMQP协议

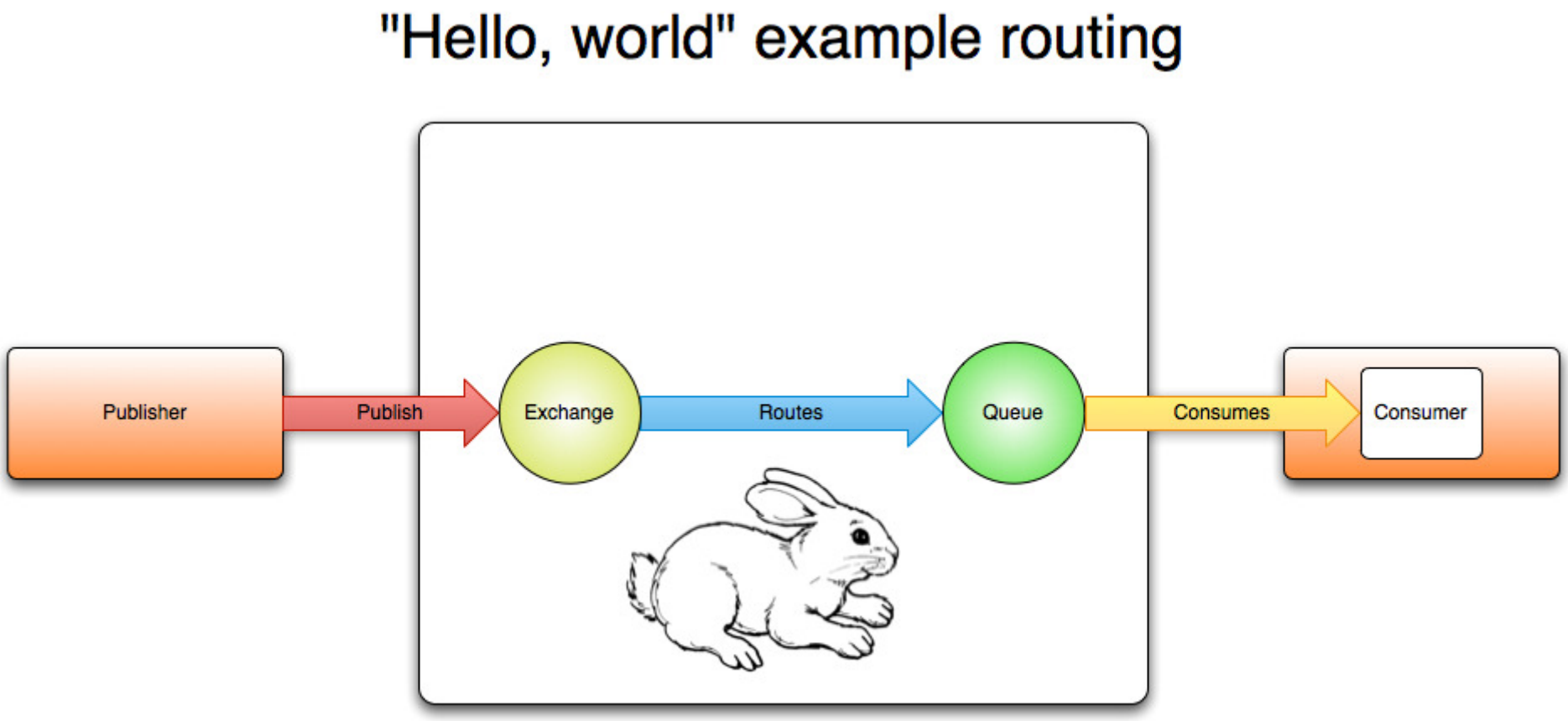
- 高级消息队列协议
- 二进制可编程协议
- 当前1.0版本
- AMQP产品：RabbitMQ、SwiftMQ、Apache Qpid、StormMQ

几个概念

- Broker、Exchange、Queue、Binding、Routing Key、vhost、channel、producer、comsumer



消息路由

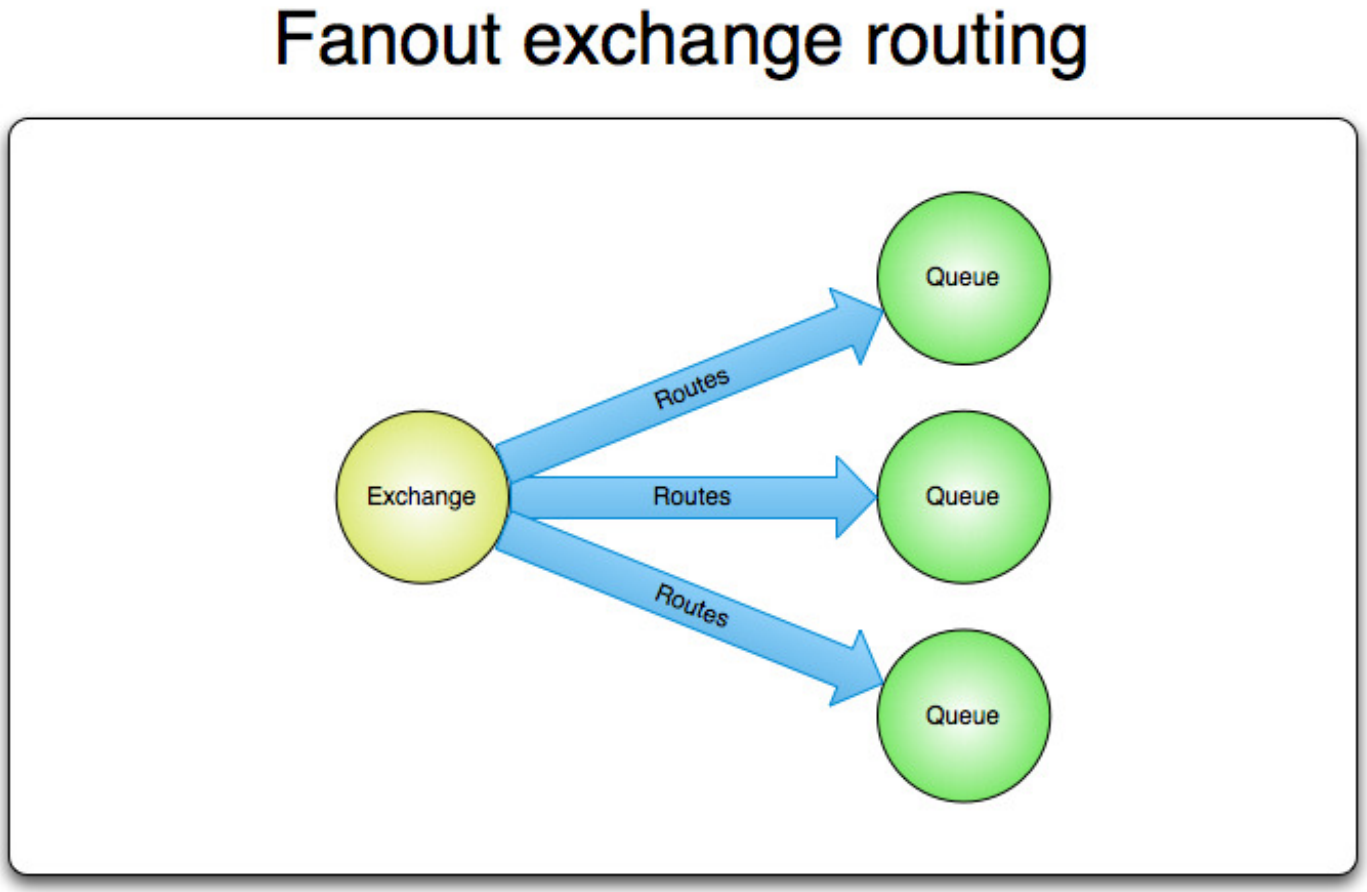

"Hello, world" example routing
Publisher
Publish
Exchange
Routes
Queue
Consumes
Consumer

Exchange类型

- Fanout
- Direct
- Topic
- Headers
- Custom

Fanout Exchange

- 将消息路由到所有绑定的队列，广播模式

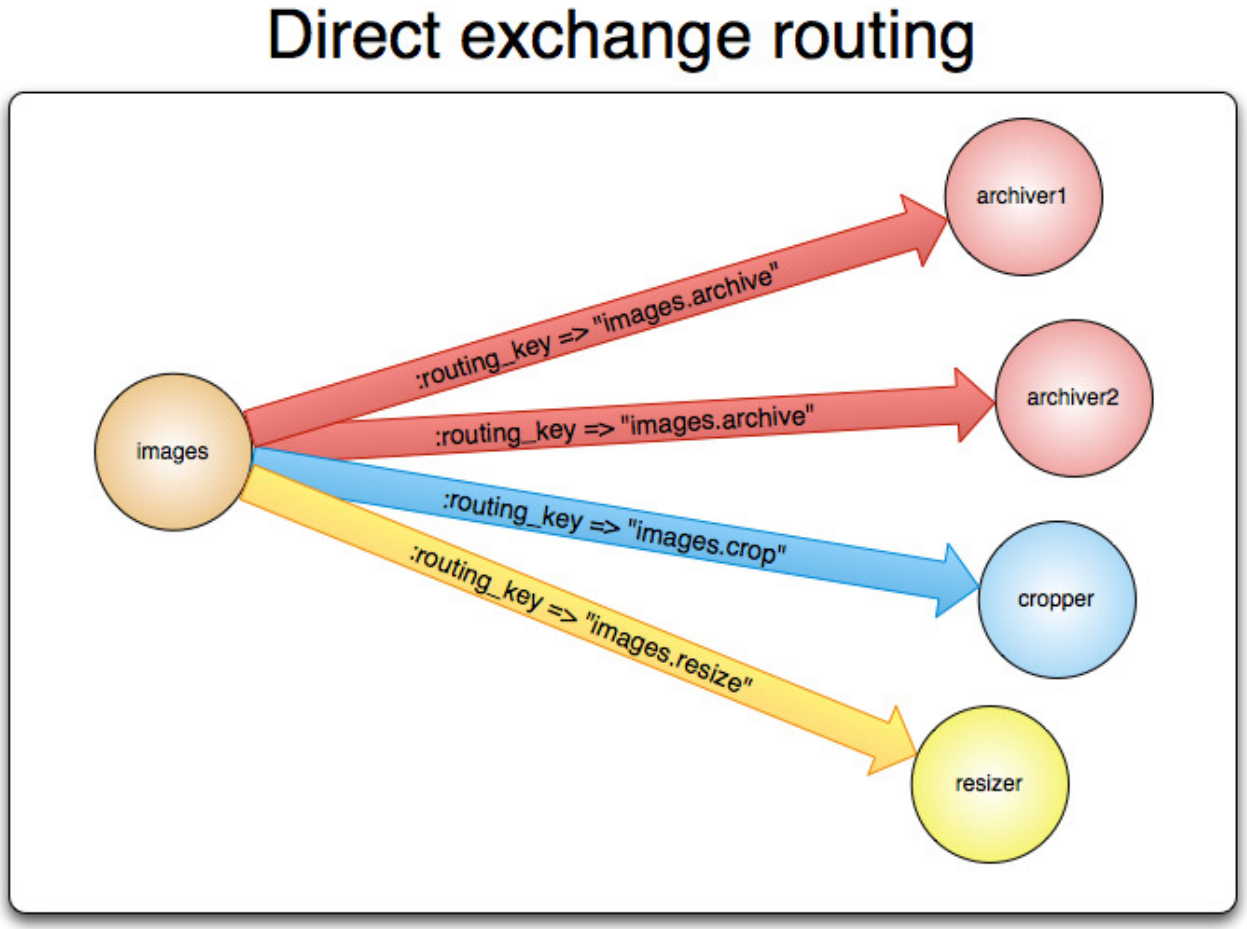


Fanout Exchange

- 用途：即时移动新闻、分布式系统、即时通讯里的群聊
- Code: fanout.rb

Direct Exchange

- 根据routing key来路由消息，单播模式



Direct Exchange

- 用途：地理位置通知服务、分布式系统里的相同计算节点、工作流数据传递
- Code: direct.rb

Default Exchange

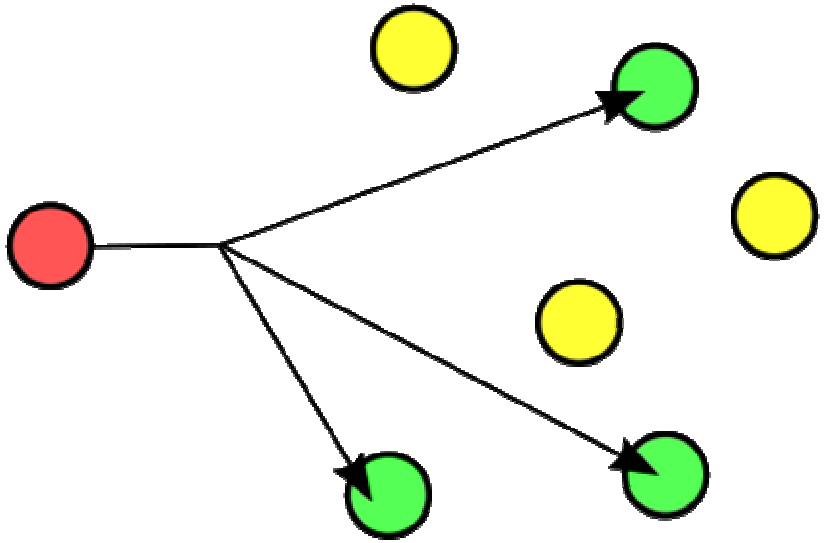
- 特殊的direct交换机，声明时名字为空串
- 默认交换机的队列无需绑定(?)
- 应用简单，客户端直接将消息投递到队列(?)

- Code: default_ex.rb

- *队列自动绑定到默认交换机*
- *队列的routing key就是队列名*

Topic Exchange

- 采用模式匹配来路由消息，多播模式



- Code: topic.rb

Topic Exchange

- 匹配规则：
- 用.分割多个字符单元，例如：

  usa.nasdaq.aapl
  tasks.search.indexing.accounts

- #匹配0或多个字符单元，例如americas.south.#匹配：

  americas.south
  americas.south.brazil
  americas.south.brazil.saopaolo

- *匹配一个字符单元，例如americas.south.*匹配：

  americas.south.brazil
- 但不匹配：

  americas.south
  americas.south.chile.santiago

发布消息 - 元数据

- :routing_key
- :persistent
- :immediate
- :mandatory
- :content_type
- :content_encoding
- :priority
- :message_id
- :correlation_id
- :reply_to
- :type
- :user_id
- :app_id
- :timestamp
- :expiration

Persistent immediate mandatory

- persistent: 发布持久性消息
- immediate: 不能立刻投递到消费者，返回失败
- Code: immediate.rb
- mandatory: 不能路由到队列，返回失败
- Code: mandatory.rb

队列声明

- 持久性队列：一直存在，不管有没有消费者，可以被多个消费者共享。
- AMQP::Queue.new(channel, "my.qname", :durable => true)

- 临时队列：只绑定到一个消费者的私有队列，该消费者断开，队列删除。
- AMQP::Queue.new(channel, "", :auto_delete => true, :exclusive => true)

队列绑定

- 队列需要绑定到交换机
- 同一交换机可以绑定多个队列

- Code: binding.rb

消息订阅

- Code: subscribe.rb

- consume、subscribe、get的区别？

  subscribe等同于：comsumer.consume.on_delivery

  consume和subscribe调用PUSH API，通知broker主动发送数据

  get调用GET API，由客户端被动的取数据

消息确认

- 自动确认模式：消息发送完，broker就从队列里将它移除
- 显式确认模式：由消费者决定何时发送ack消息，接受到ack消息后，broker才从队列里移除消息。如果发送ack之前，客户端断开了，那么broker将消息投递到同一队列的其他消费者。

消息确认

```
queue.subscribe(:ack => true) do |metadata, payload|
    # message handling logic...
    channel.acknowledge(metadata.delivery_tag, false)
end
```

*It takes two arguments: message **delivery tag** and a flag that indicates whether or not we want to acknowledge multiple messages at once. Delivery tag is simply a channel-specific increasing number that the server uses to identify deliveries*

消息拒绝

- 应用可以发送拒绝消息，指示broker消息处理失败，应用可以请求broker丢弃或重传该消息。

消息拒绝

```
queue.bind(exchange).subscribe do |metadata, payload|
    # 拒绝后丢弃
    channel.reject(metadata.delivery_tag)
end

queue.bind(exchange).subscribe do |metadata, payload|
    # 拒绝后重新排队
    channel.reject(metadata.delivery_tag, true)
end
```

其他队列方法

- GET消息(PULL API): queue.pop
- 取消订阅: queue.unsubscribe
- 取消绑定: queue.unbind(exchange)
- purge队列: queue.purge 删除队列里所有消息
- 删除队列: queue.delete

参考

- AMQP Official: http://amqp.org/
- Ruby AMQP: http://rubyamqp.info/
- RabbitMQ: http://www.rabbitmq.com/
- 风河博客: http://www.nsbeta.info/