ADD-01

RabbitMQ: Messaging in the Cloud

Matthew Sackman
matthew@rabbitmq.com

RabbitMQ

Coming up in the next hour. . .

RabbitMQ

Coming up in the next hour. . .

- Messaging, messaging, messaging: What does it all mean?

Coming up in the next hour. . .

- Messaging, messaging, messaging: What does it all mean?
- What is this angular mutant orange Rabbit thing, and why is it dropping AMQP all over the carpet?

Coming up in the next hour. . .

- Messaging, messaging, messaging: What does it all mean?
- What is this angular mutant orange Rabbit thing, and why is it dropping AMQP all over the carpet?
- How many balloons does it take to keep a Rabbit in a cloud?

RabbitMQ

Coming up in the next hour. . .

- Messaging, messaging, messaging: What does it all mean?
- What is this angular mutant orange Rabbit thing, and why is it dropping AMQP all over the carpet?
- How many balloons does it take to keep a Rabbit in a cloud?
- Prizes! Prizes! Prizes!

MESSAGING IS BECOMING EVER MORE IMPORTANT BECAUSE:

- Scalability issues demand greater flexibility from application developers
- Traditional synchronous programming models fair poorly at large scale
- Cloud computing permits greater dynamic scaling than ever seen before, but applications need to be written well to take advantage of this
- Messaging enables scaling by decoupling components and adding flexibility

RabbitMQ in the Cloud

RabbitMQ

Turns out Rabbits quite enjoy Helium

- Several cloud infrastructures have been built *using* RabbitMQ as the nervous system of the cloud
- RabbitMQ is easily installed and used by clients on existing clouds such as Amazon EC2
- RabbitMQ is available as an additional component on Heroku
- RabbitMQ is going to become available in more clouds in the future, e.g. Social, Nebula, VMforce
- RabbitMQ is just as easy to use to solve problems in clouds as it is to solve problems on the ground

What is Messaging?

What is Messaging?

What is a Banana?

What can I use Messaging for?

What can I use Messaging for?

What can I use a Banana for?

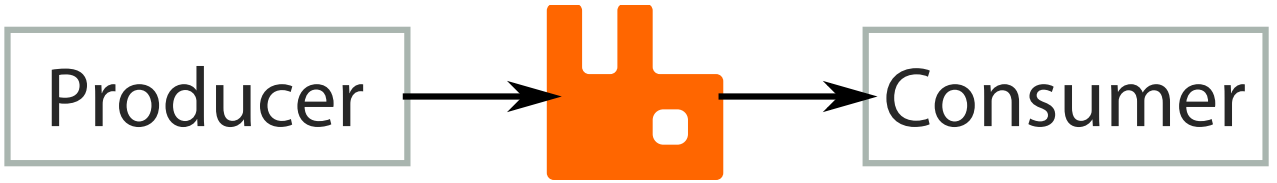
What can I use a Messaging protocol for?

Use cases for Messaging Protocols

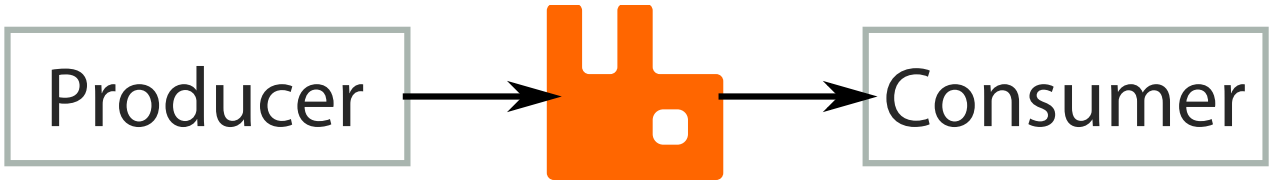
RabbitMQ

It's got to be good for something, right?

Decoupling



Decoupling



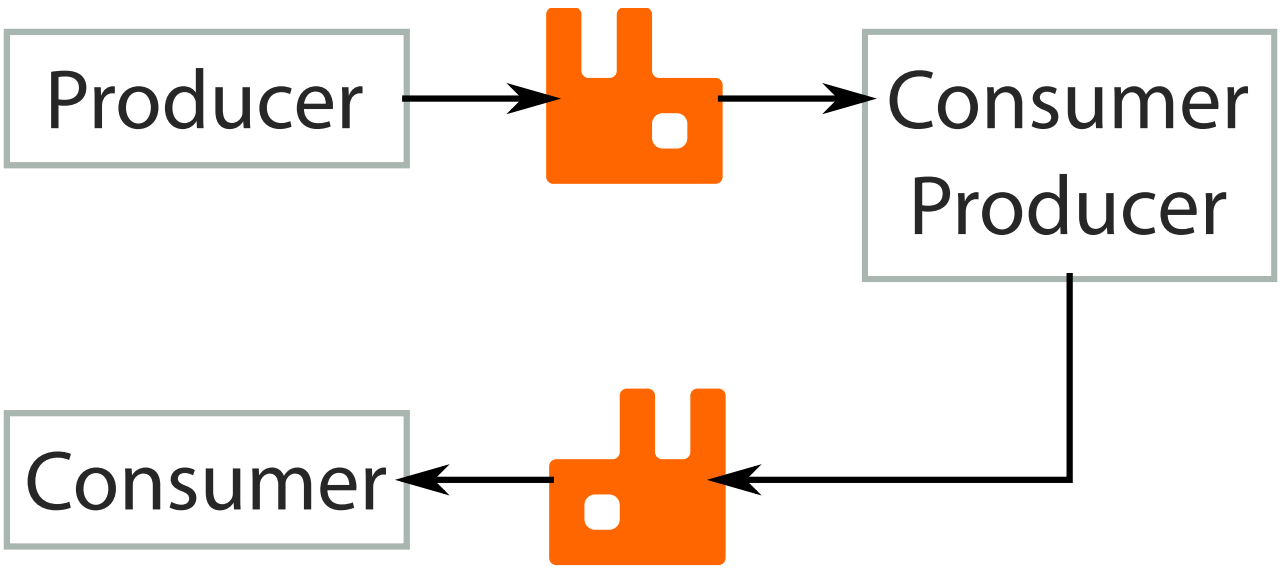
E.g. website passing orders to a credit-card charging engine

Bidirectional Decoupling

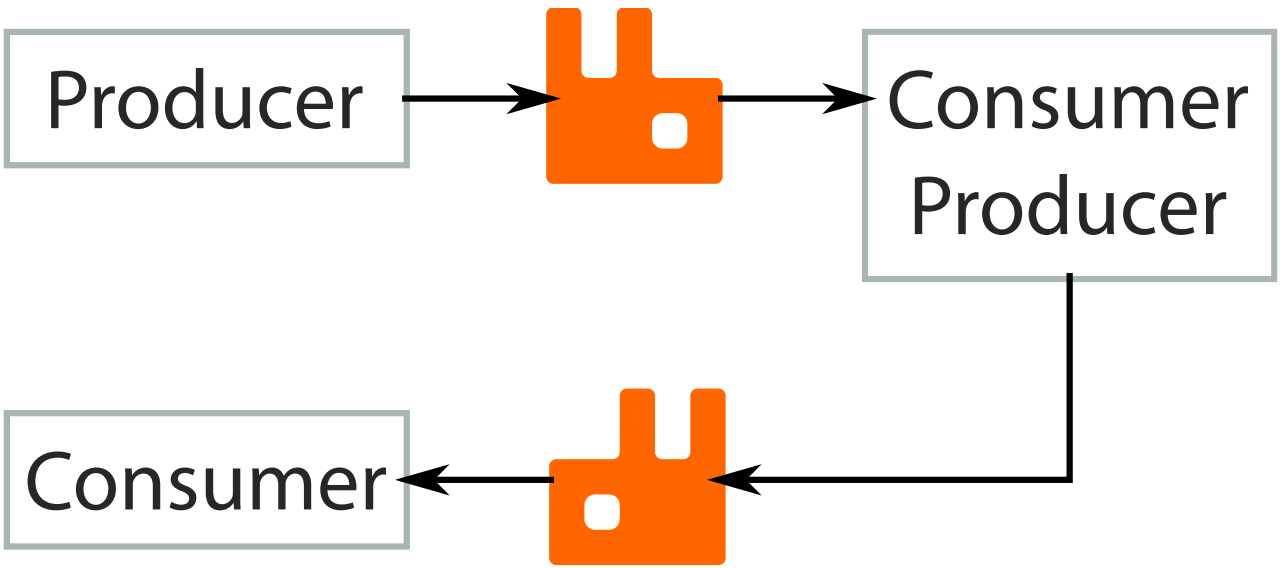
Bidirectional Decoupling

E.g. remote procedure call

Pipelining and Decoupling

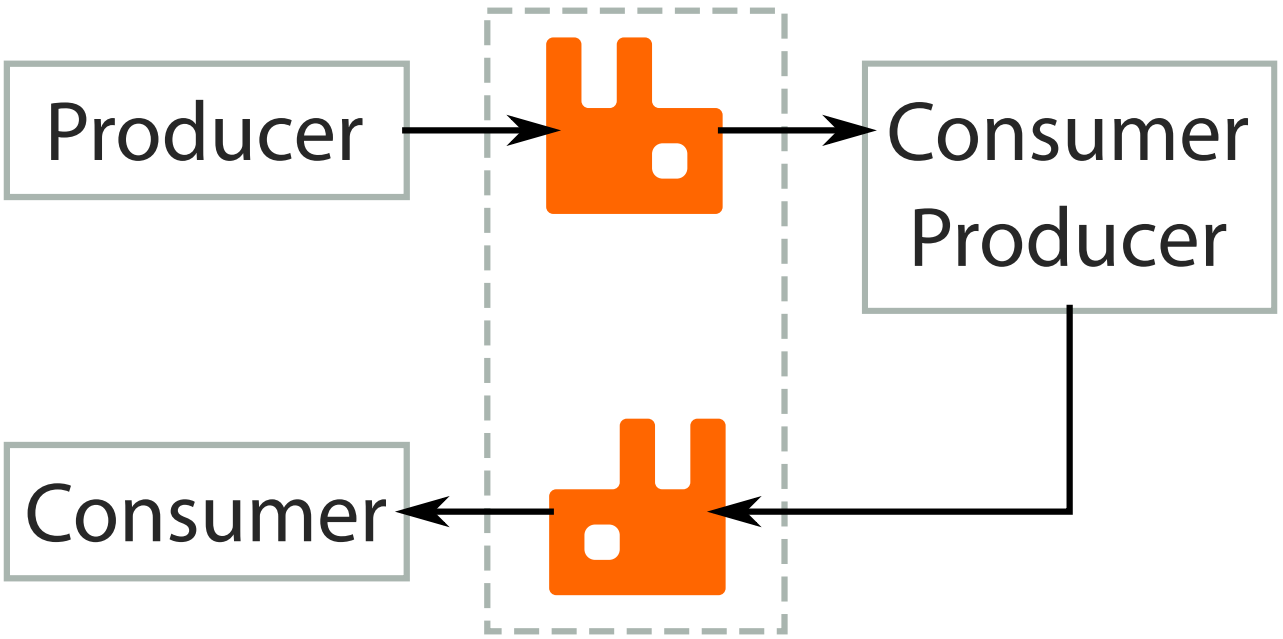


Pipelining and Decoupling



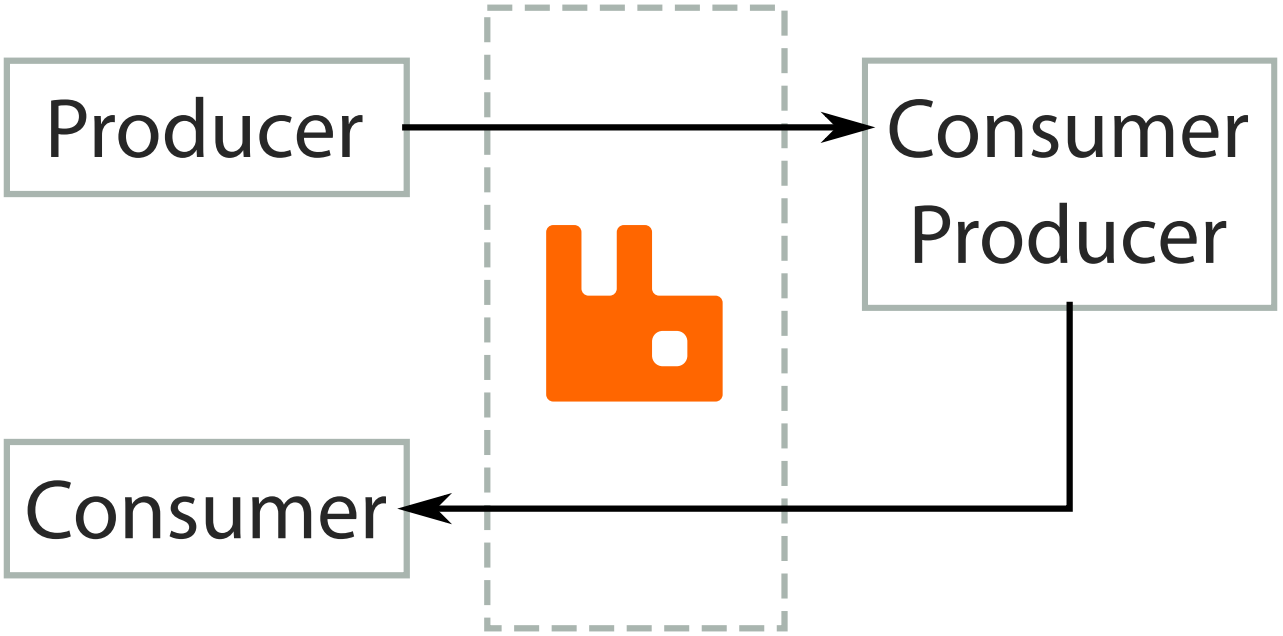
E.g. combining messages with records in a database

Pipelining and Decoupling



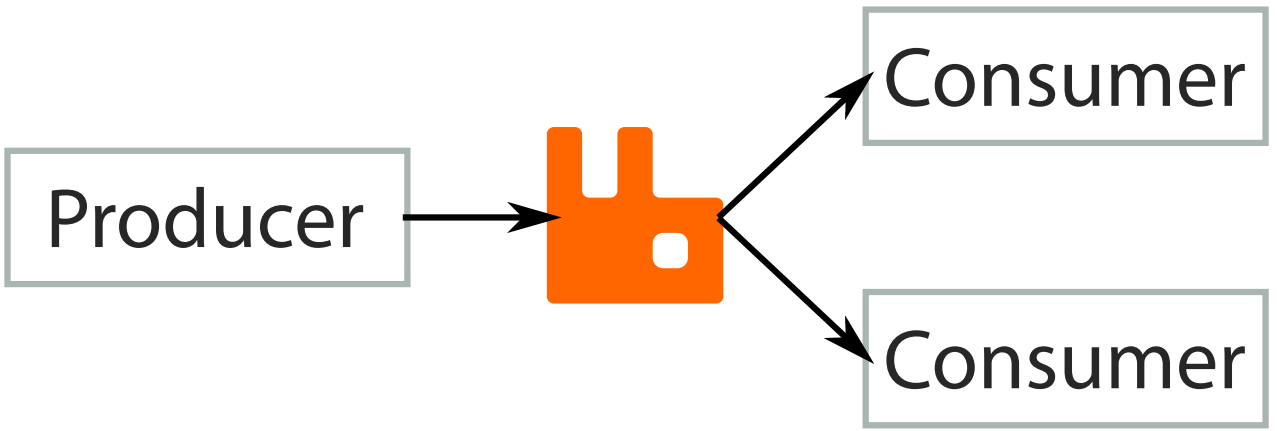
E.g. combining messages with records in a database

Pipelining and Decoupling

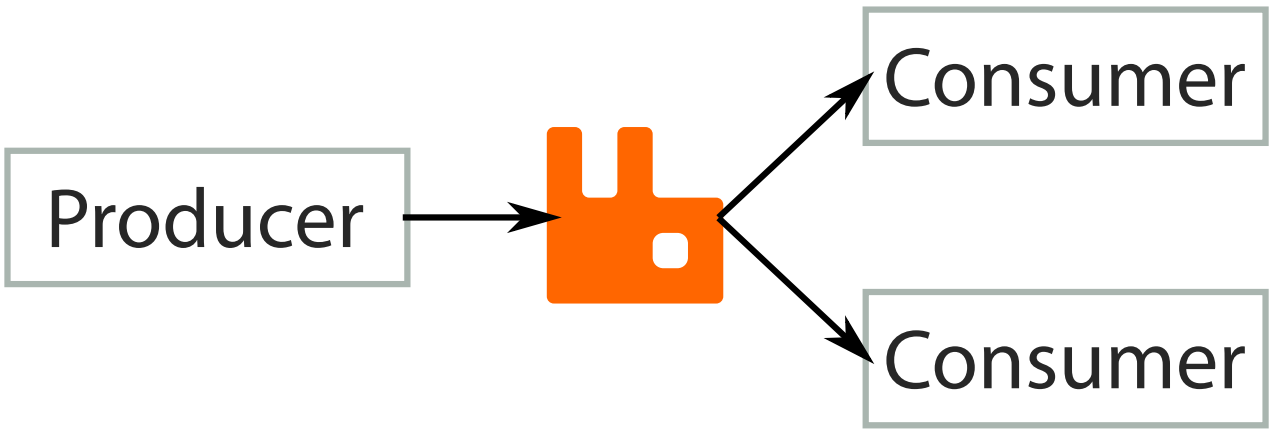


E.g. combining messages with records in a database

Work-distribution and Decoupling

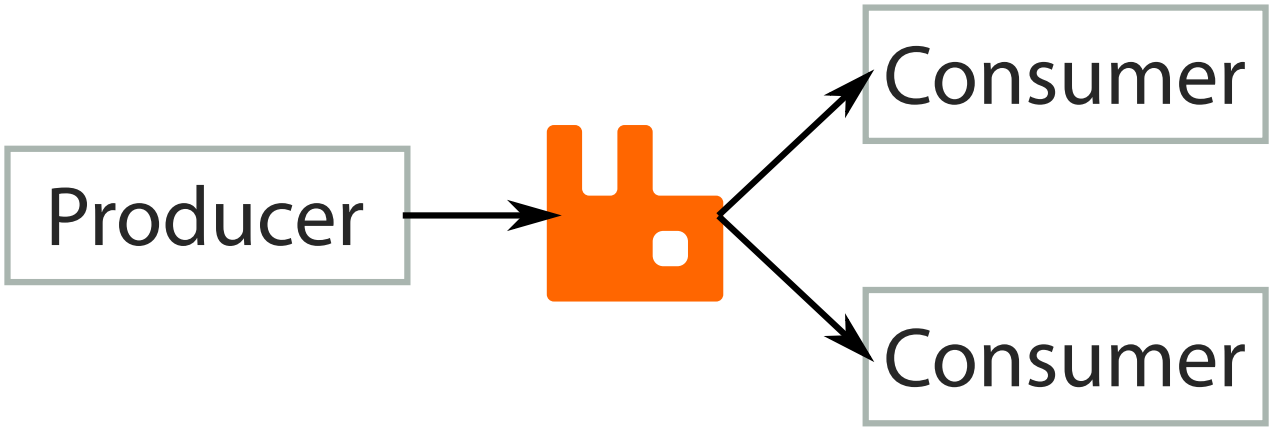


Work-distribution and Decoupling



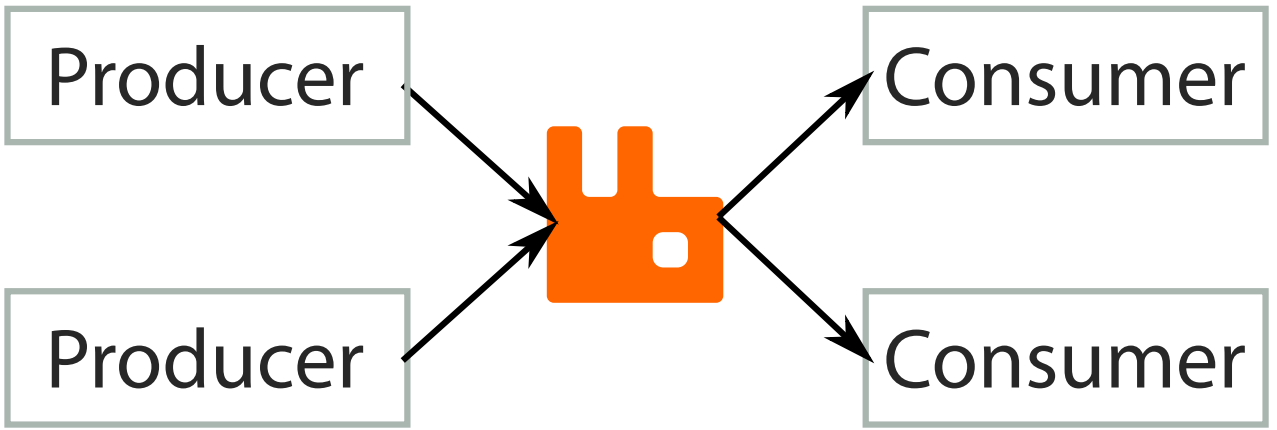
Both duplication and round-robin.

Work-distribution and Decoupling



Both duplication and round-robin.
Duplication can be used for logging messages at certain points in the system.
Round-robin can be used for horizontal scaling.

Work aggregation, distribution and Decoupling



Other things message brokers can typically do

- Store-and-forward of messages
- Absorbing spikes of activity (again, decoupling)
- Routing to different consumers based on message properties

Other things message brokers can typically do

- Store-and-forward of messages
- Absorbing spikes of activity (again, decoupling)
- Routing to different consumers based on message properties

General conclusion

- Messaging protocols as a means of decoupling events is an extremely common and pervasive task
- There is an enormous range of applications and problems to which using a messaging protocol is an effective solution

AMQP:
Advanced Message Queueing Protocol

Why AMQP?

RabbitMQ

It's called *Advanced*, so it must be good!

AMQP is especially nice because...

- All resources are dynamically created and destroyed by clients as they need them – no static preconfiguration
- A clean and simple model: just three key nouns to learn
- Open standard, developed by the AMQP Working Group (we're members)
- Lots of client libraries available in many languages, for free

Why AMQP?

RabbitMQ

It's called *Advanced*, so it must be good!

AMQP is especially nice because...

- All resources are dynamically created and destroyed by clients as they need them – no static preconfiguration
- A clean and simple model: just three key nouns to learn
- Open standard, developed by the AMQP Working Group (we're members)
- Lots of client libraries available in many languages, for free
- An excellent, freely available, open source broker implementation, called RabbitMQ…

Create an exchange,…

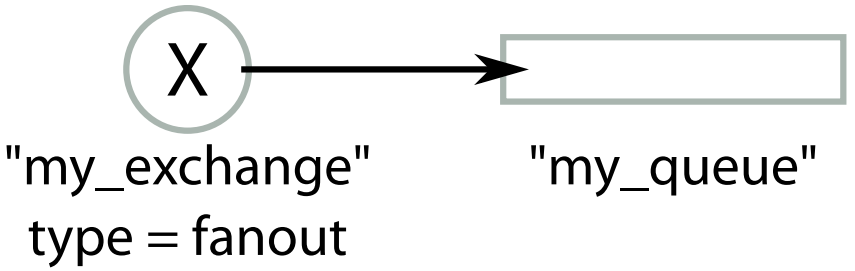
"my_exchange"
type = fanout

…create a queue,…

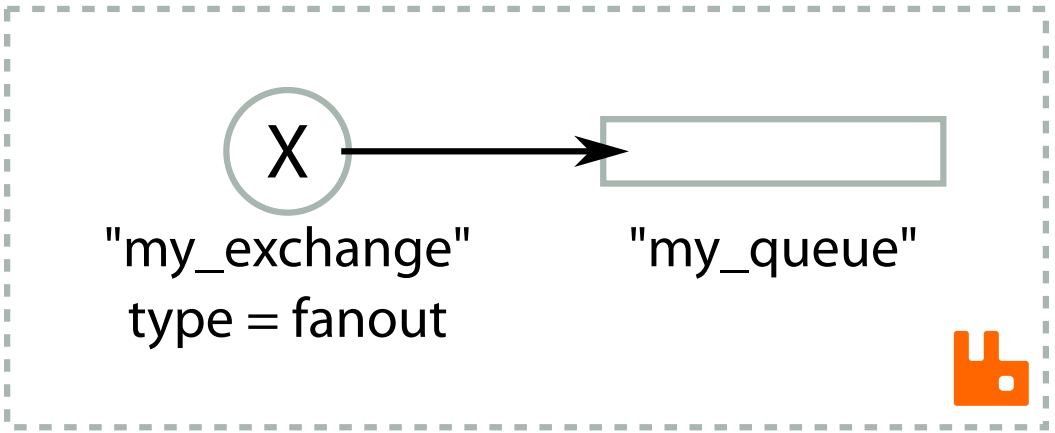
"my_exchange"
type = fanout

"my_queue"

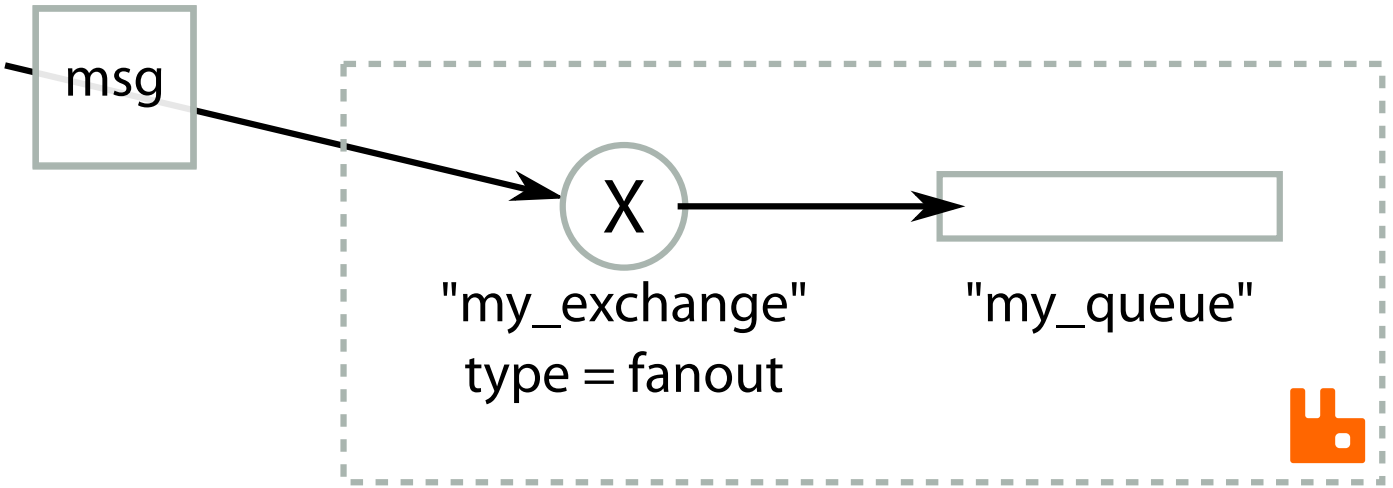
…add a binding,…



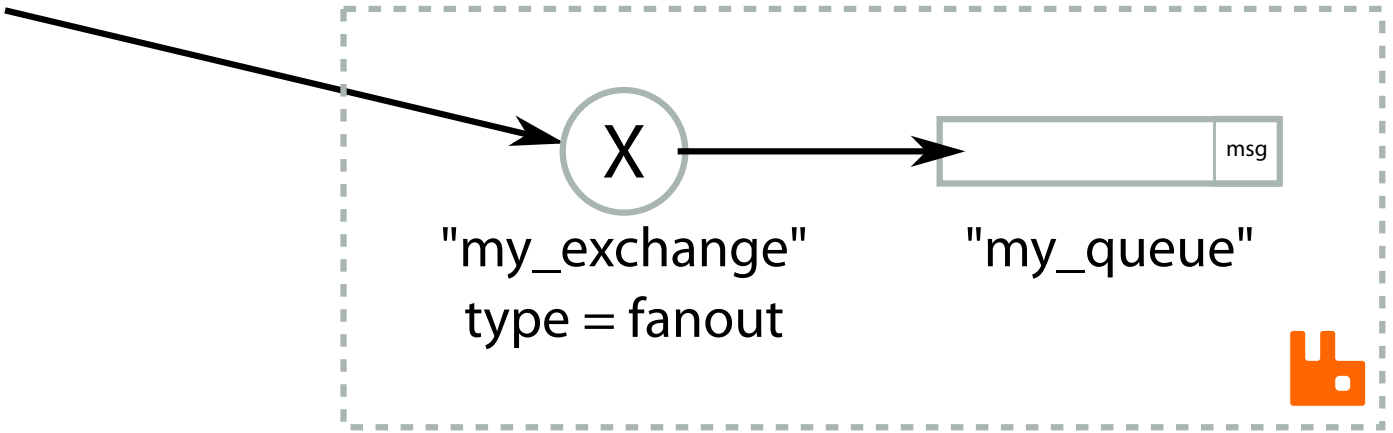
…all inside a broker.



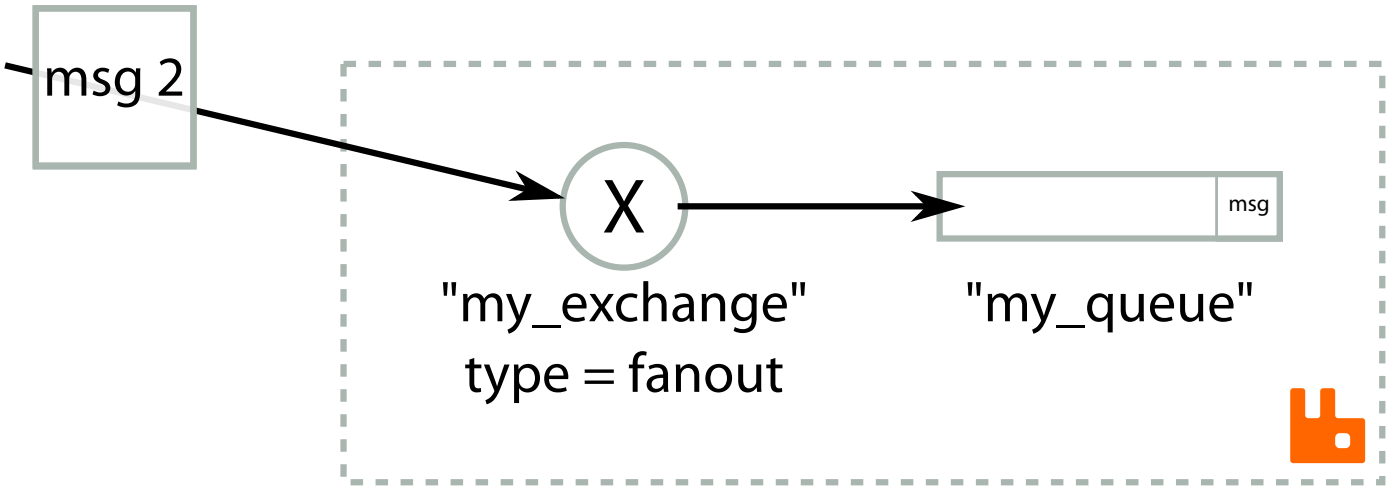
Publish a message, …



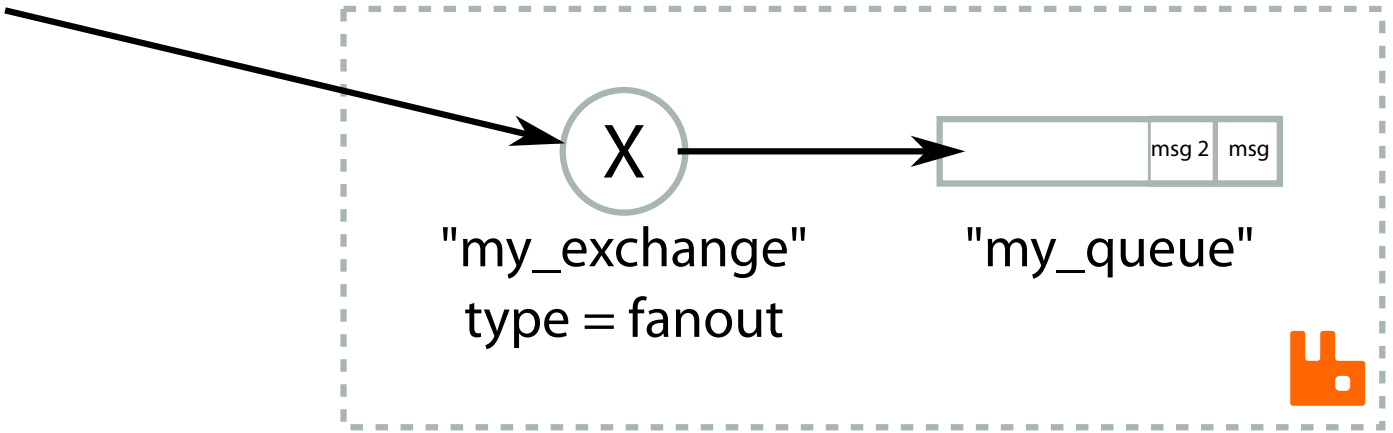
…it sits in a queue.



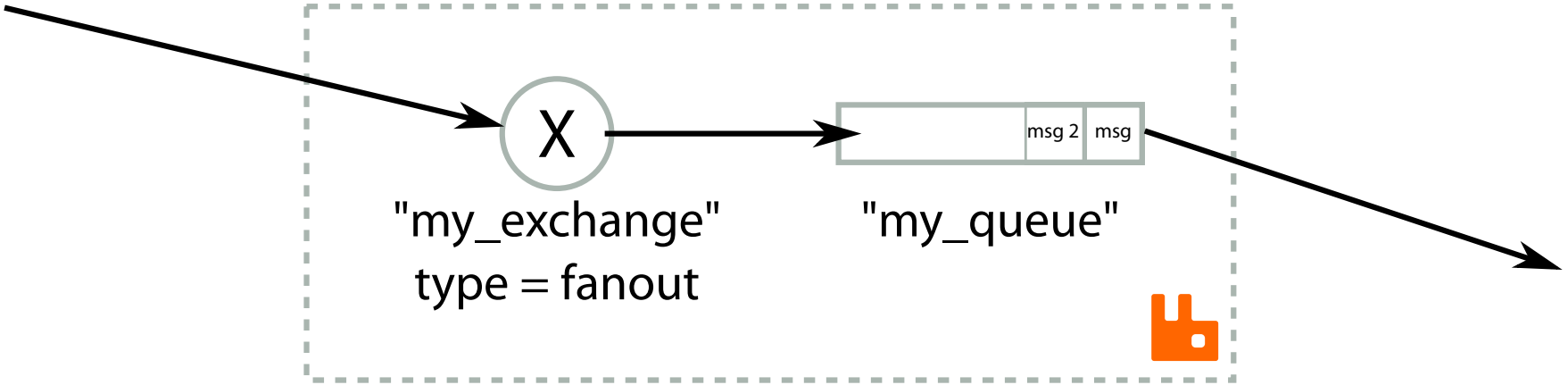
Publish another message, …



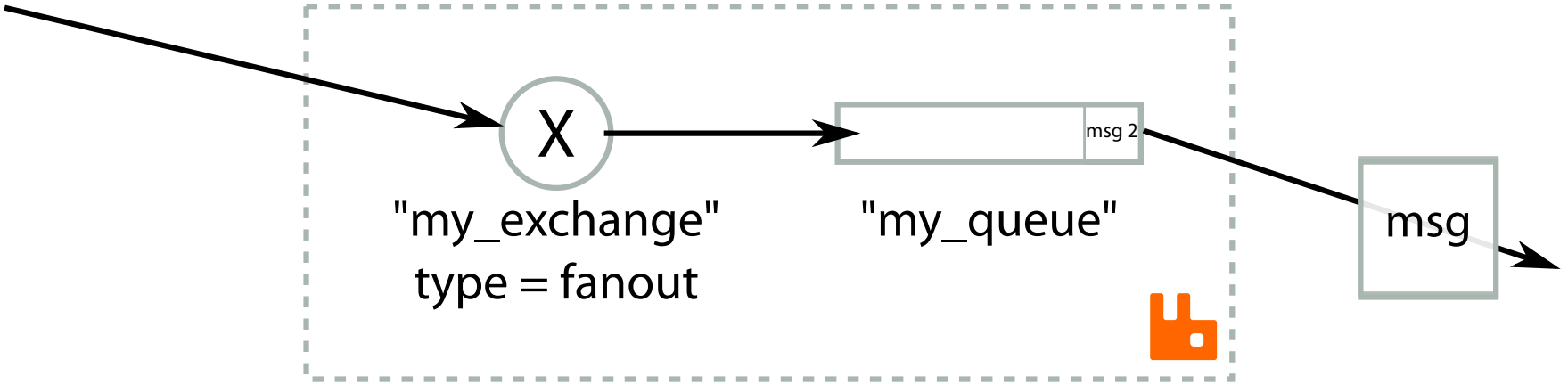
…it also goes to the queue.



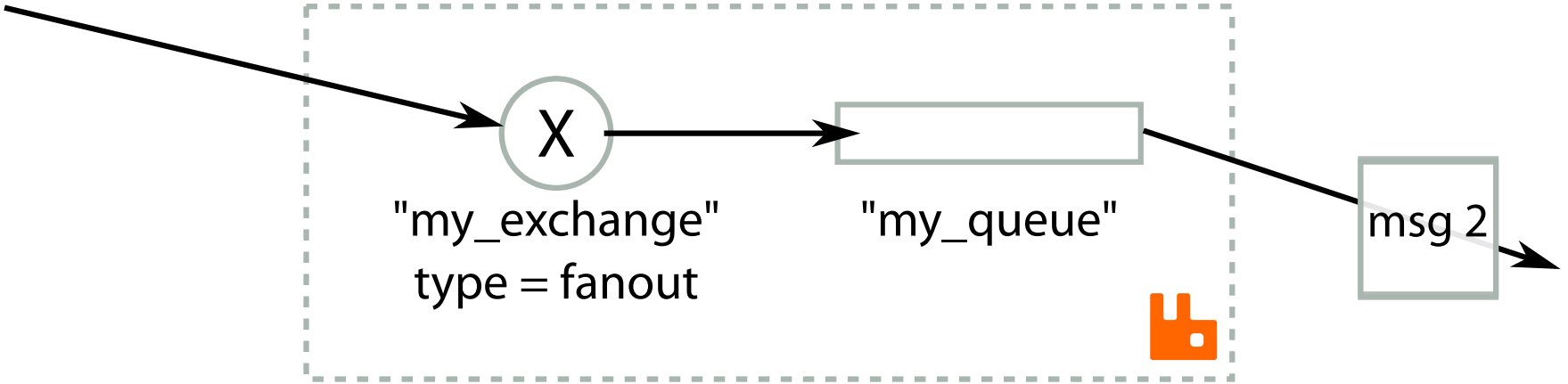
Consuming from the queue retrieves the messages in order.



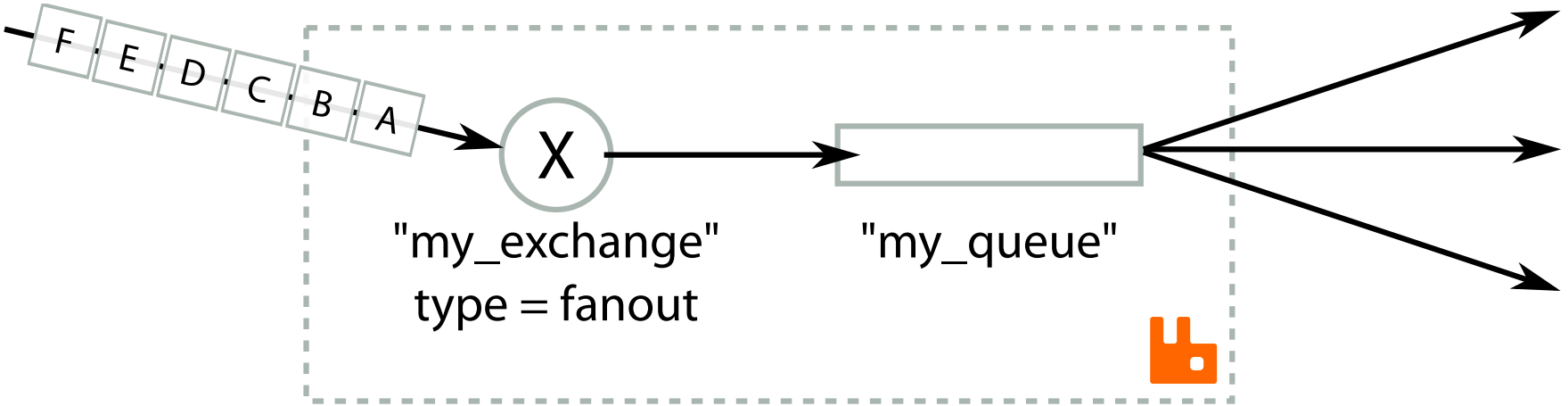
Consuming from the queue retrieves the messages in order.



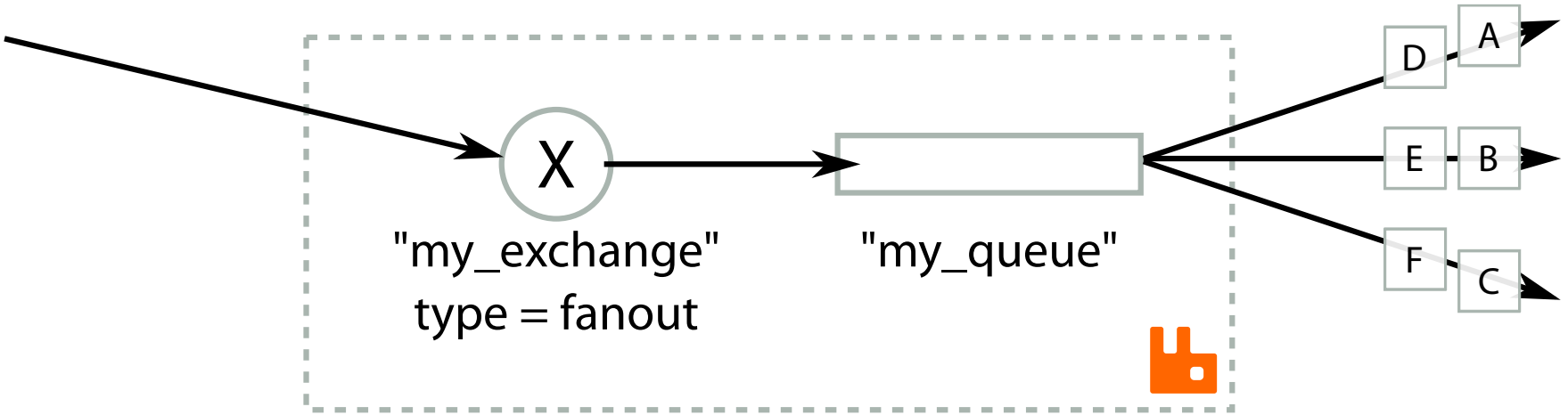
Consuming from the queue retrieves the messages in order.



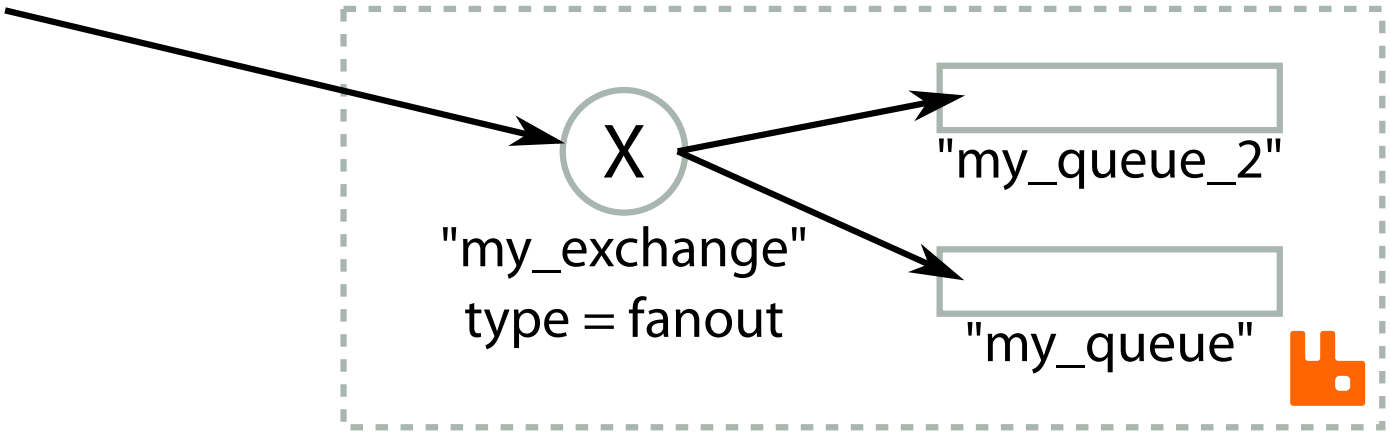
Publish many messages,…



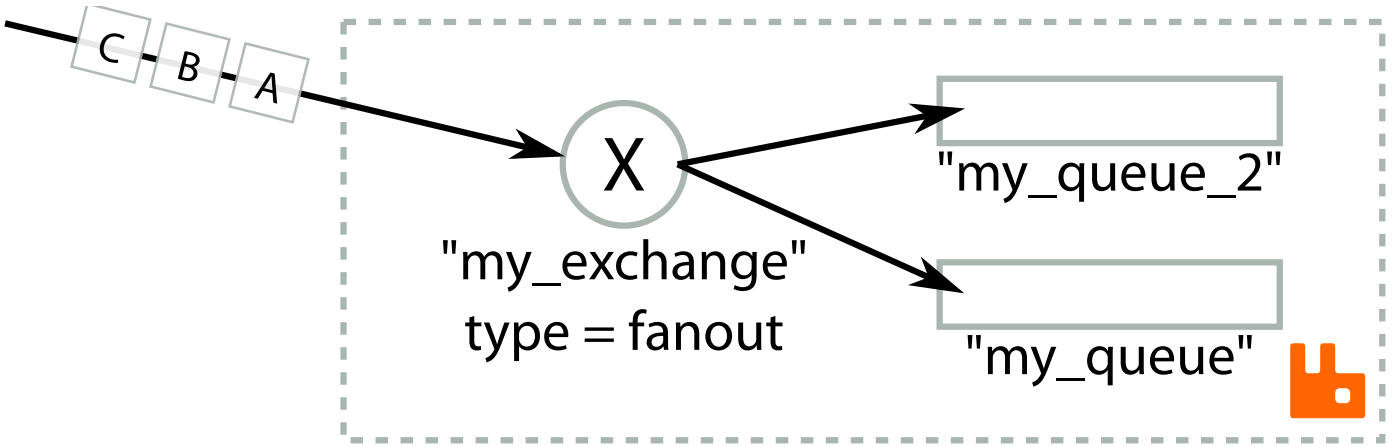
…and they are distributed amongst several consumers on the same queue.



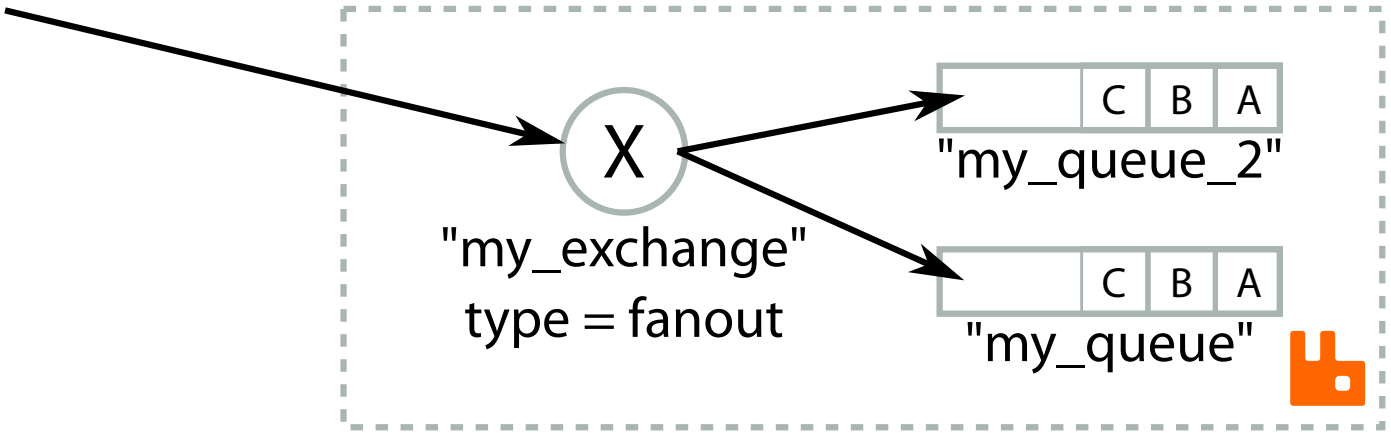
We can create a second queue and bind it to the same exchange.



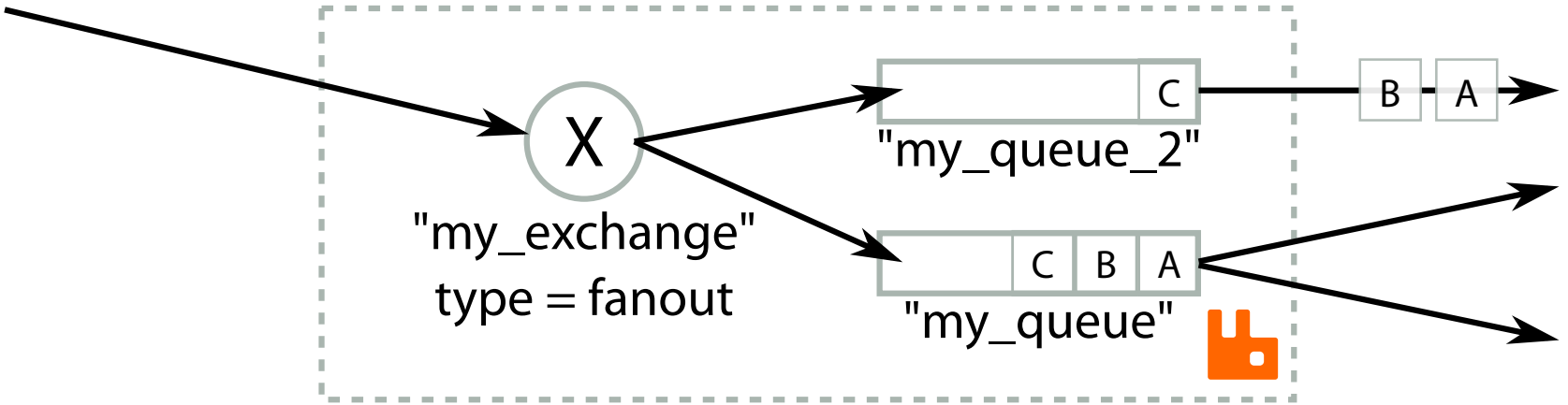
Messages go to *every* queue bound to a *fanout* exchange …



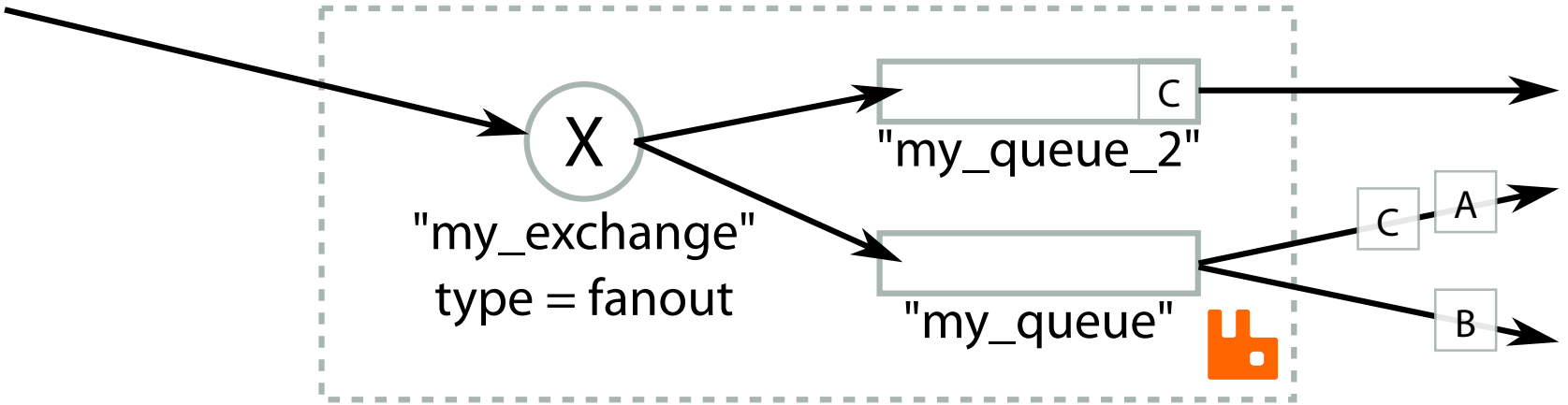
Messages go to *every* queue bound to a *fanout* exchange …



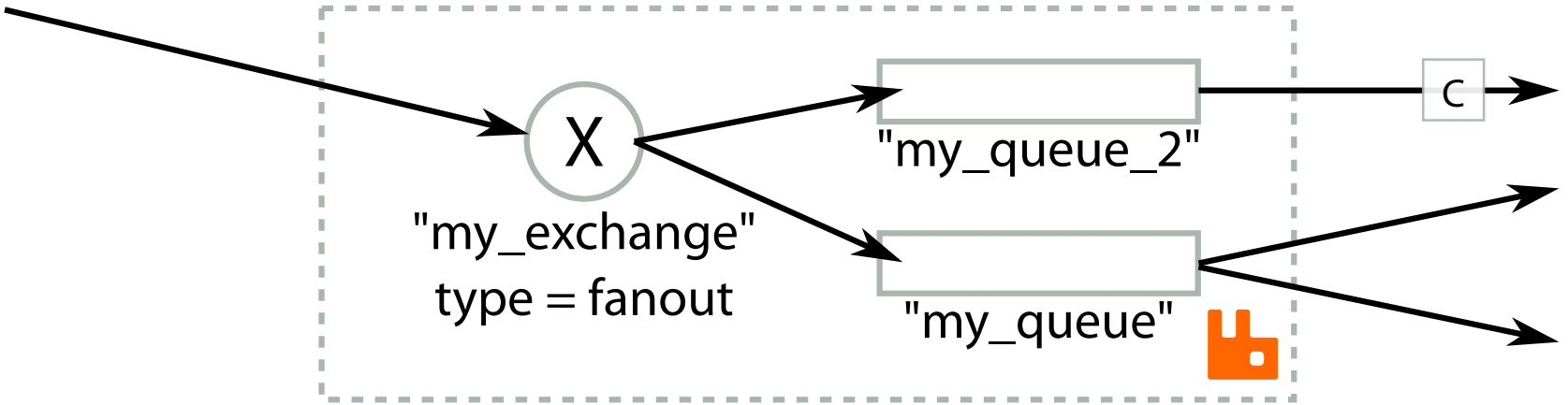
…and the queues can be consumed from at different rates.



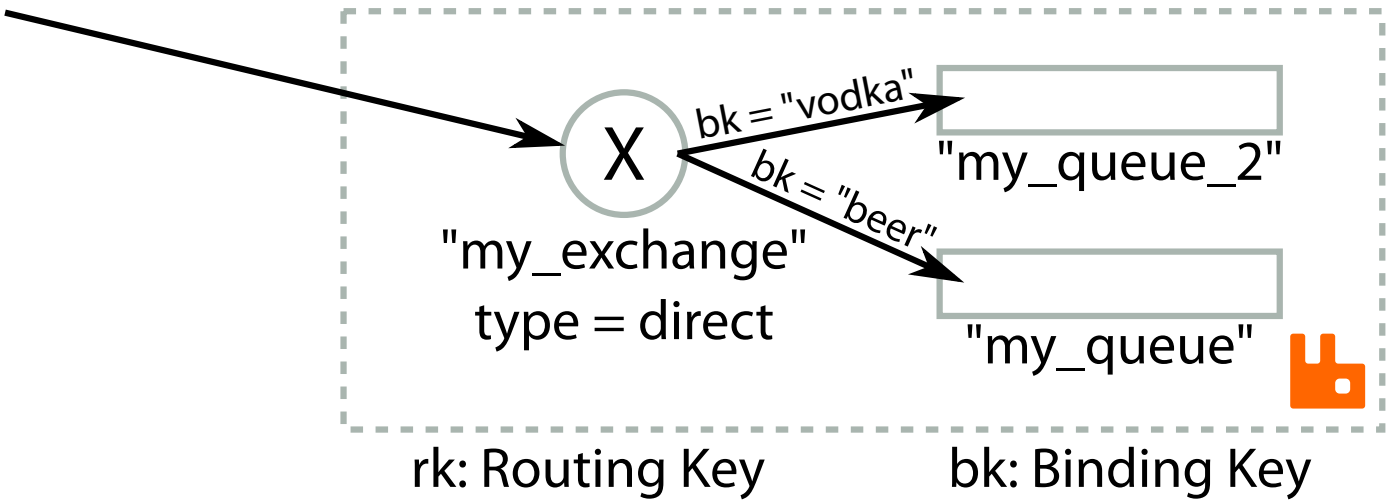
…and the queues can be consumed from at different rates.



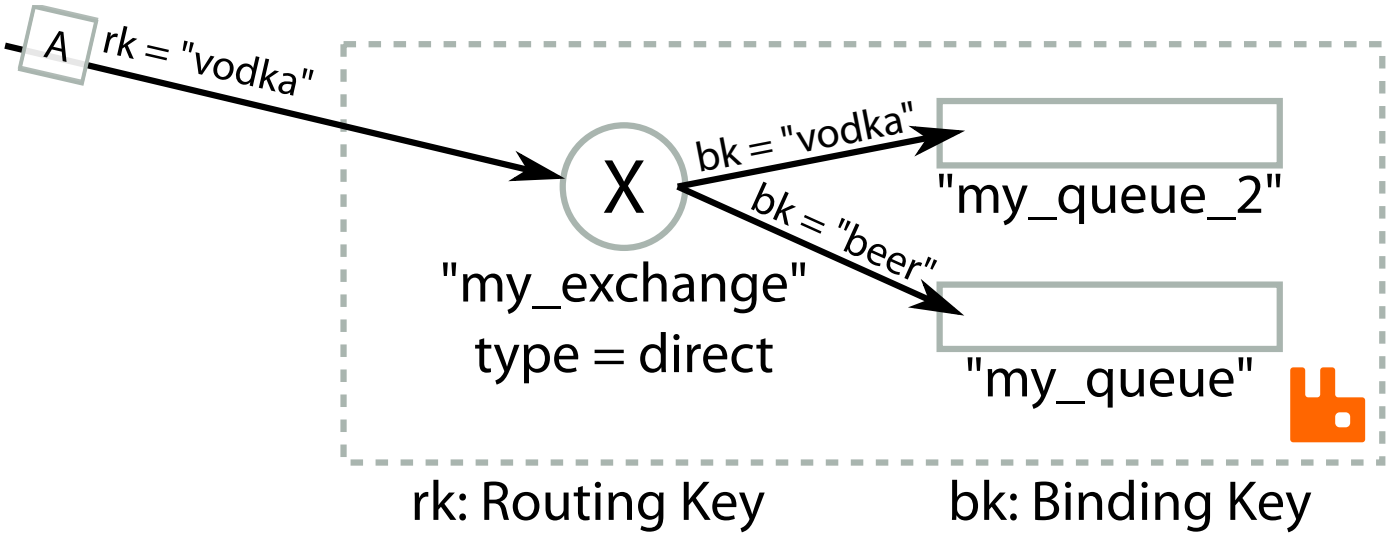
…and the queues can be consumed from at different rates.



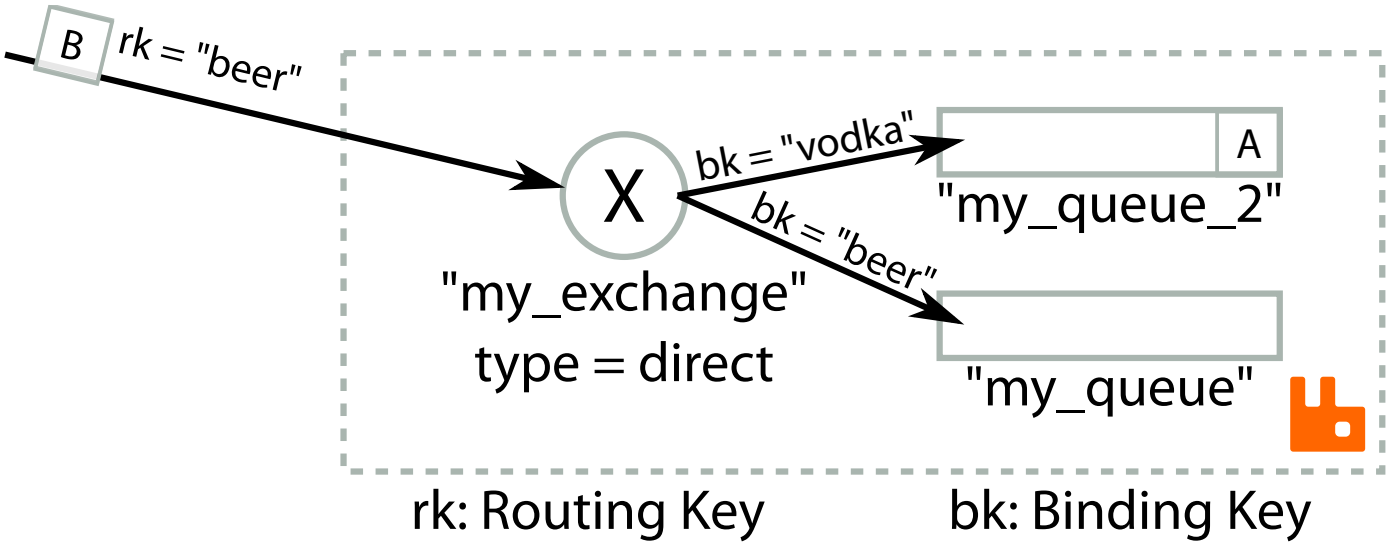
We replace “my_exchange” with a *direct* exchange.



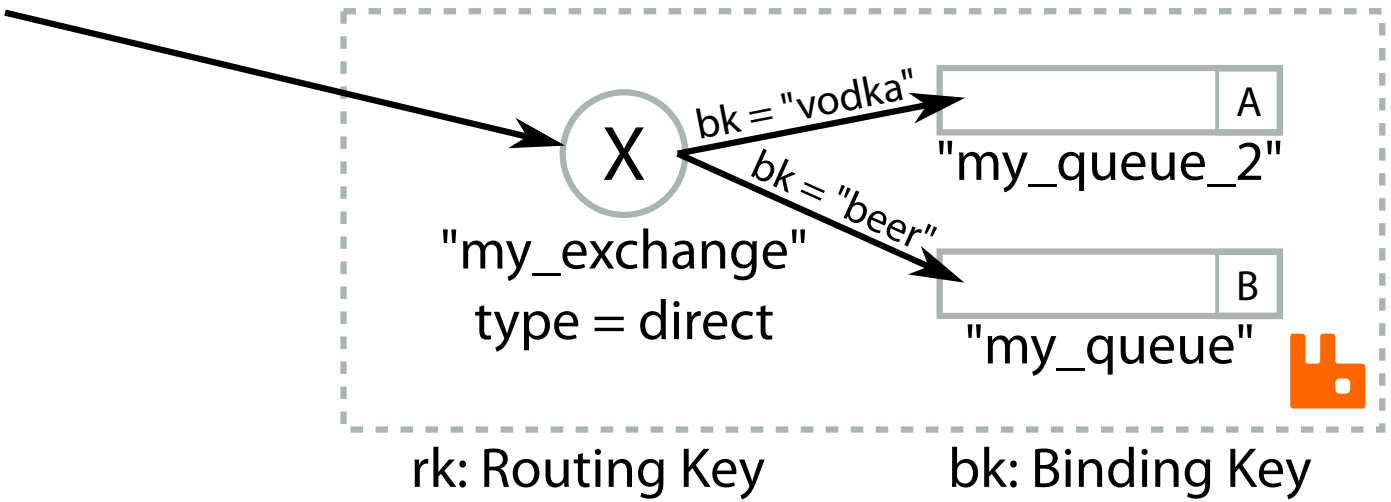
The routing key of a published message selects which queues the message goes to.



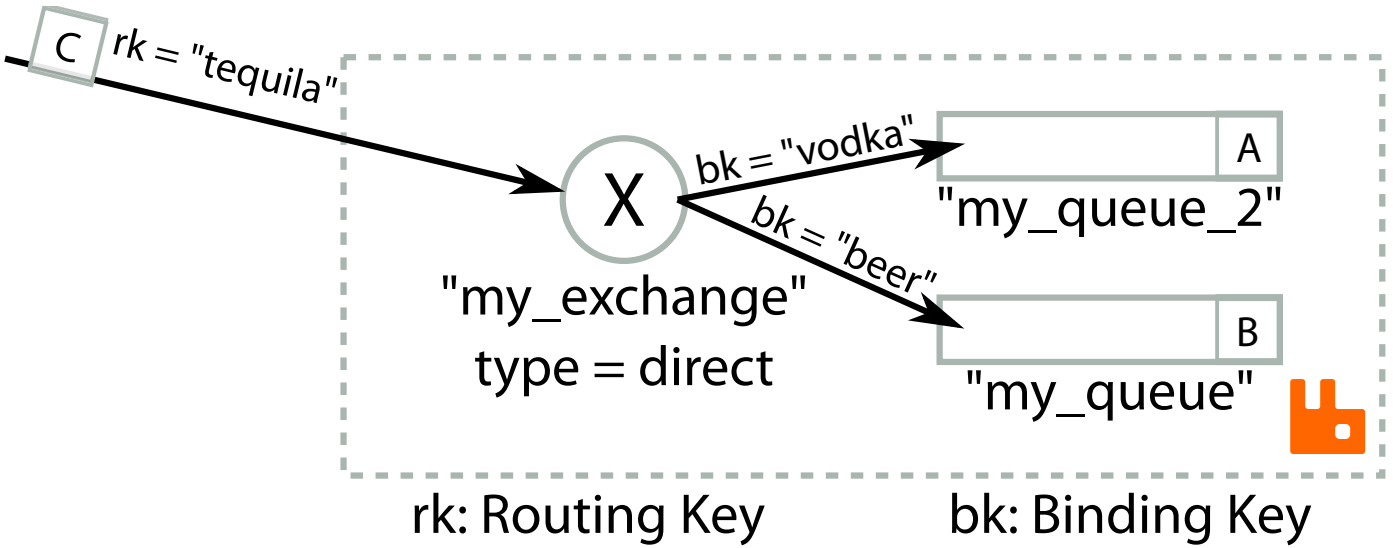
The routing key of a published message selects which queues the message goes to.



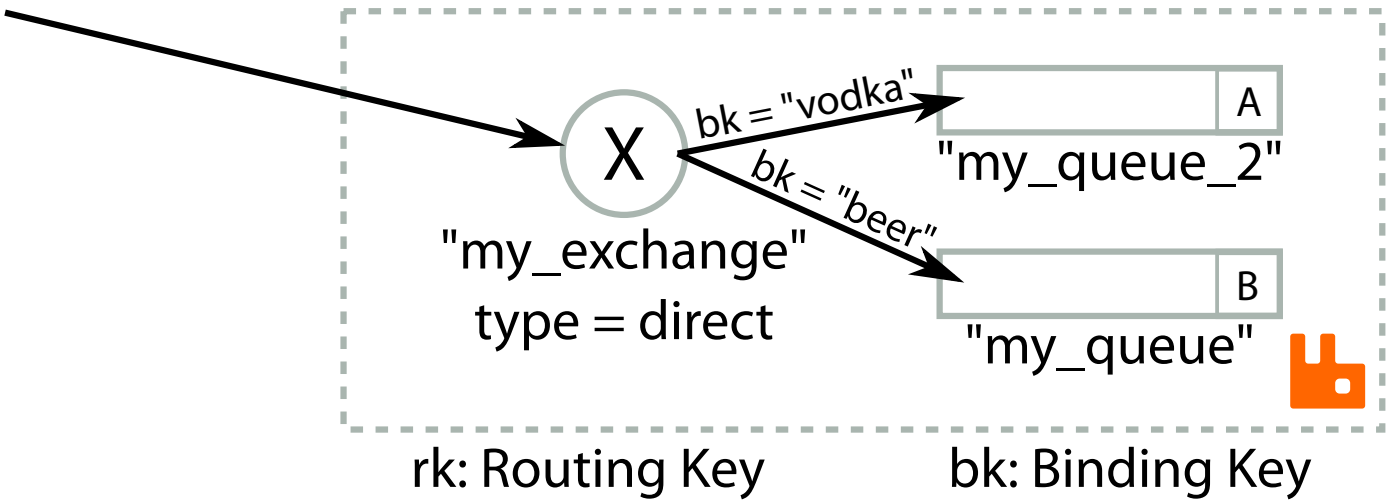
The routing key of a published message selects which queues the message goes to.



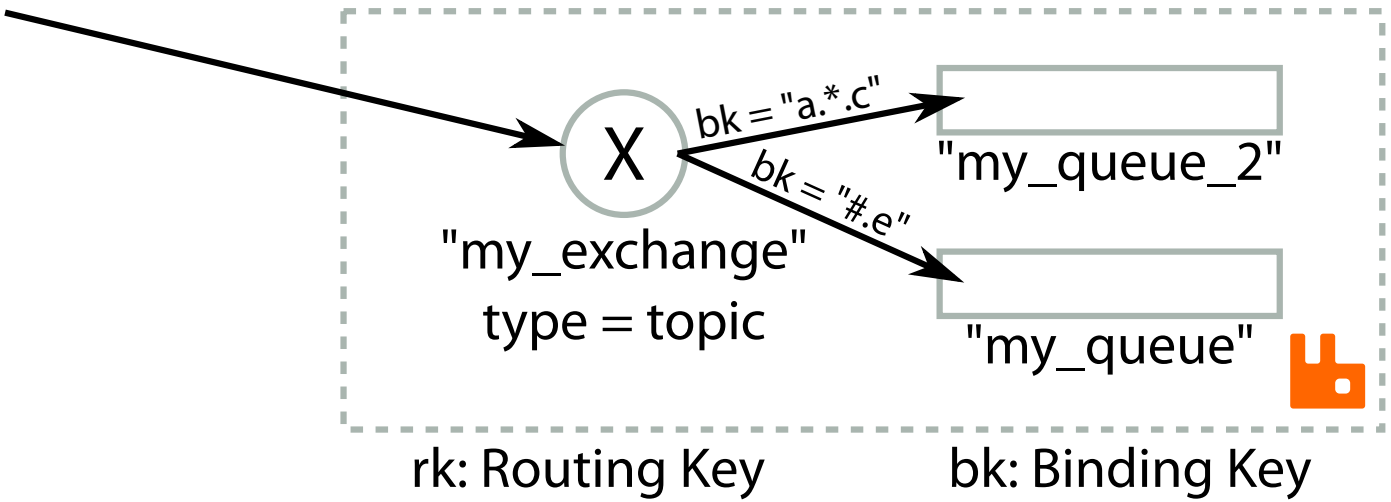
Messages with no matching bindings are discarded.



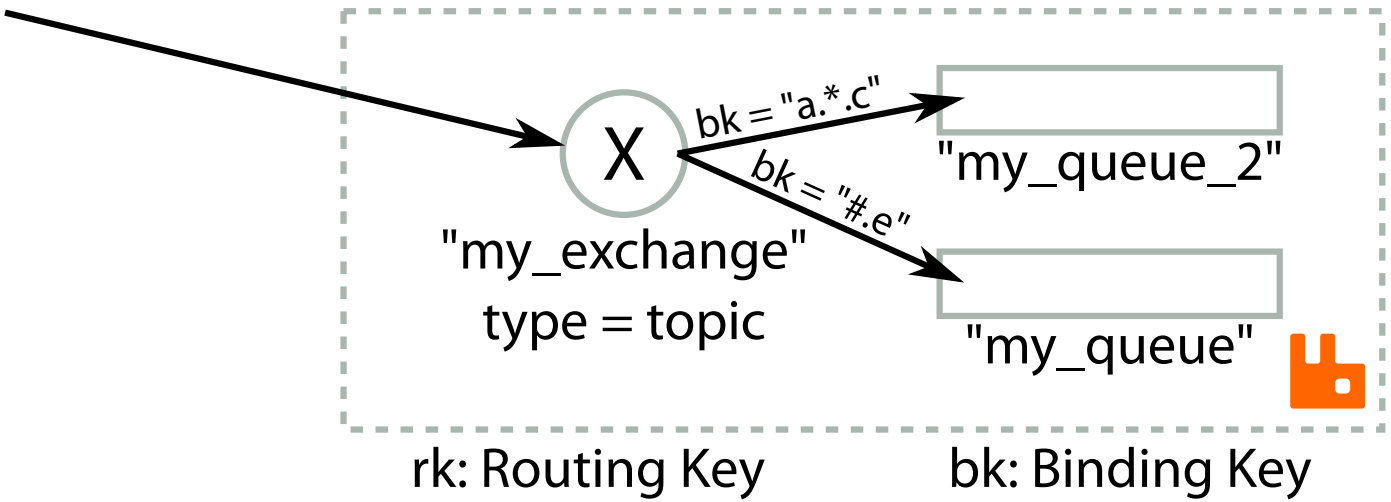
Messages with no matching bindings are discarded.



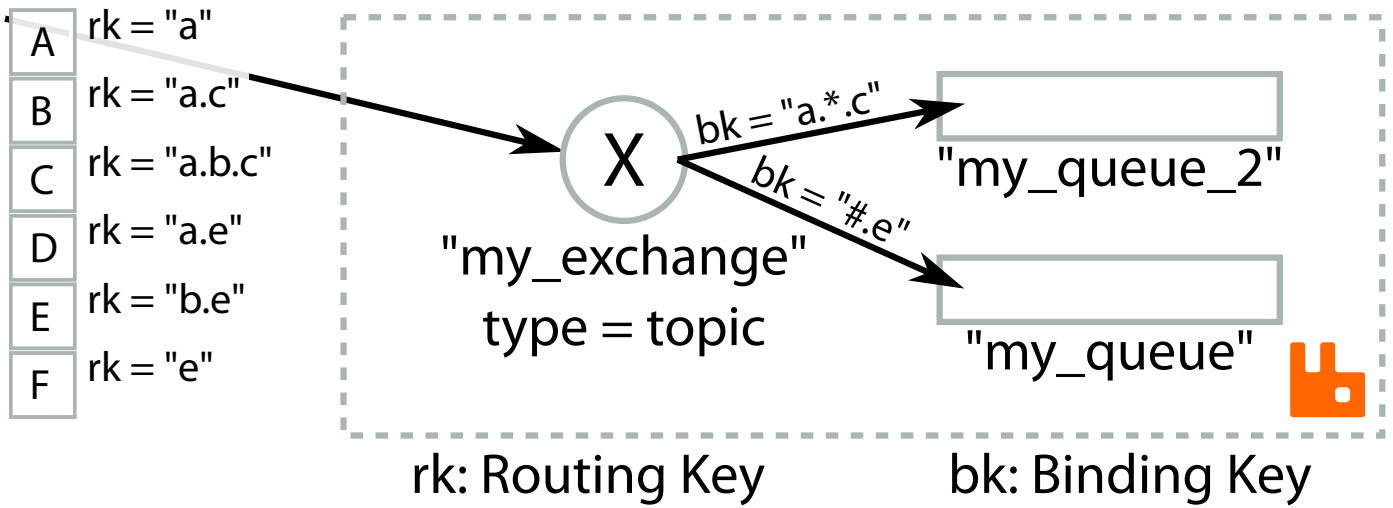
Topic exchanges permit wildcards in the binding key.



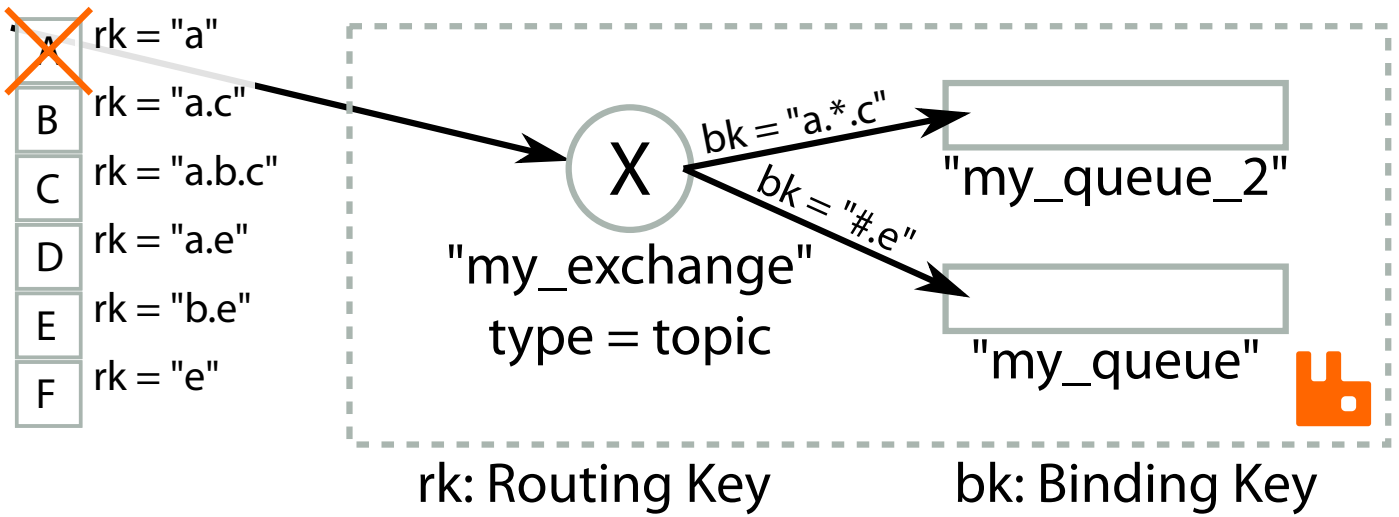
Topic exchanges permit wildcards in the binding key.
Keys are .-separated lists, e.g. “stocks.nyse.vmw”
* matches any 1 element.
# matches zero or more elements.



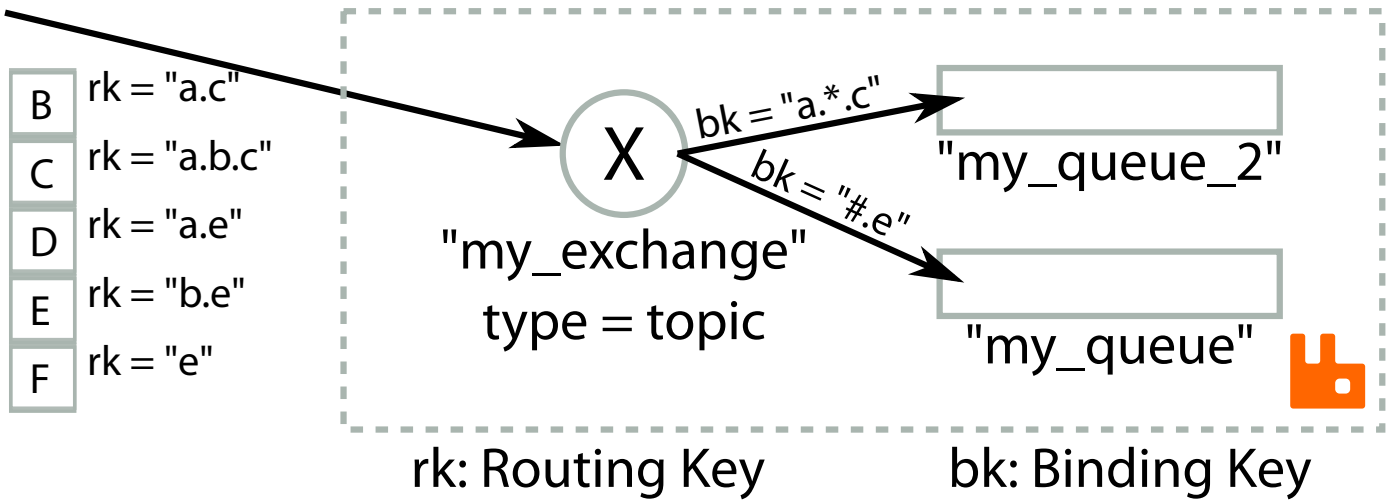
Topic exchanges permit wildcards in the binding key.
Keys are .-separated lists, e.g. “stocks.nyse.vmw”
* matches any 1 element.
# matches zero or more elements.



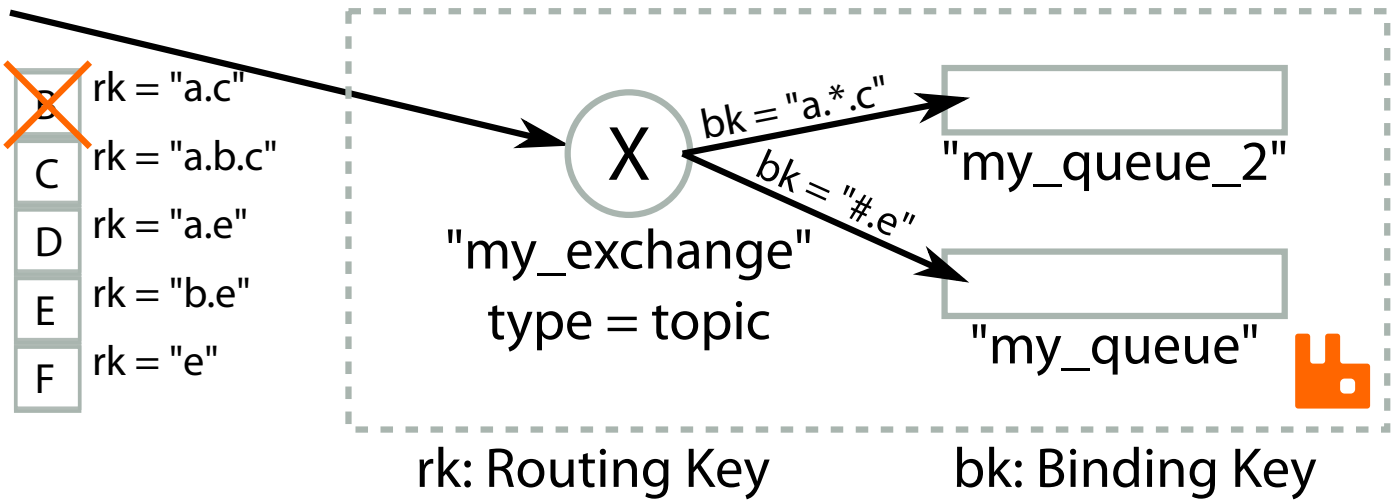
Topic exchanges permit wildcards in the binding key.
Keys are .-separated lists, e.g. “stocks.nyse.vmw”
* matches any 1 element.
# matches zero or more elements.



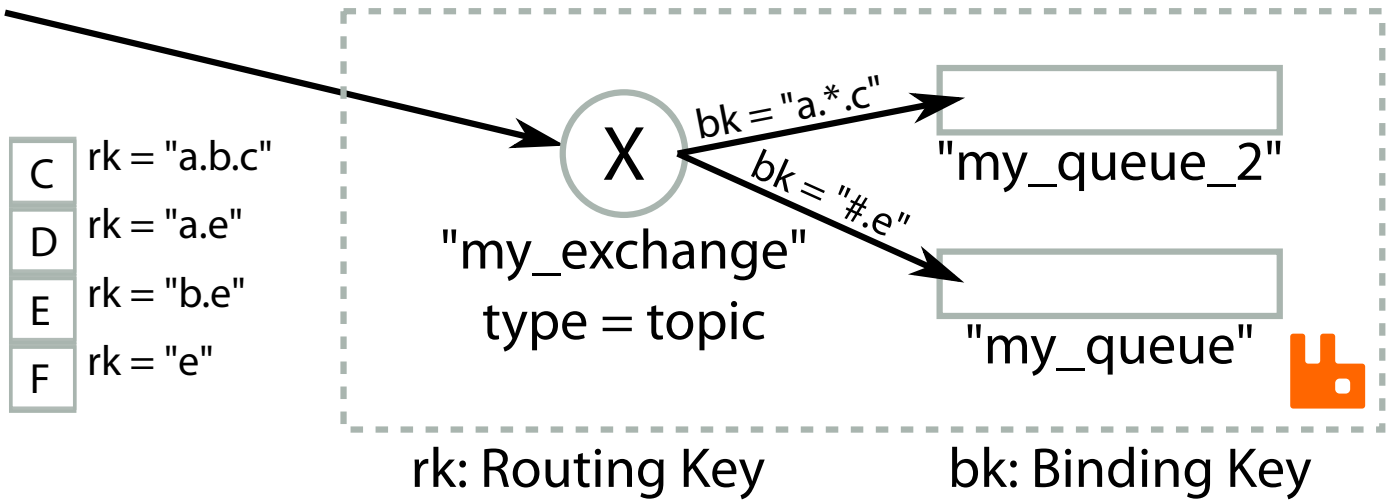
Topic exchanges permit wildcards in the binding key.
Keys are .-separated lists, e.g. “stocks.nyse.vmw”
* matches any 1 element.
# matches zero or more elements.



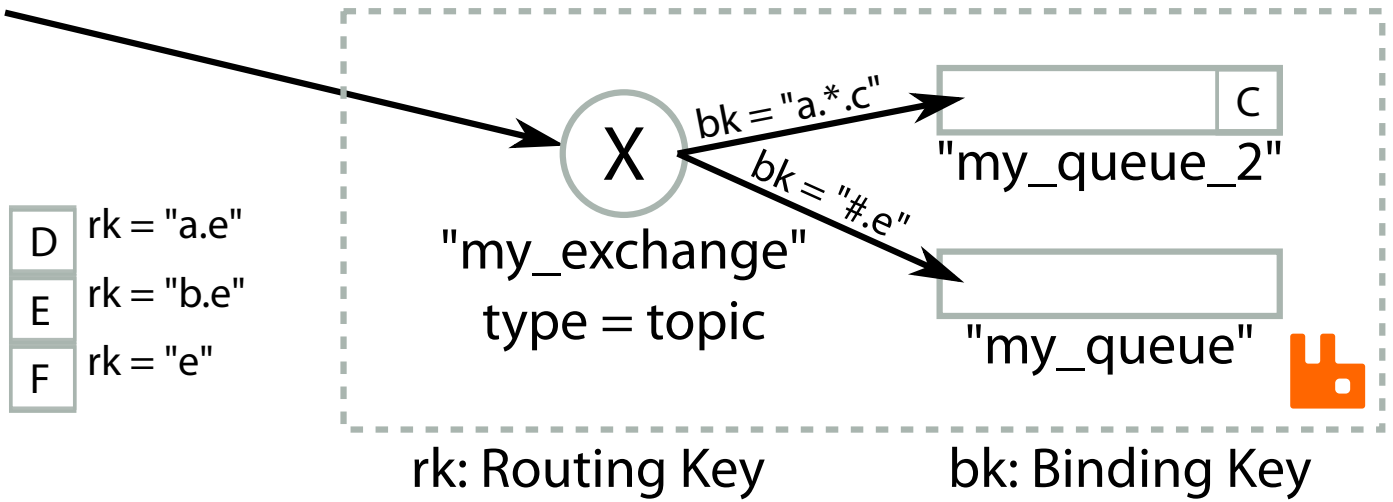
Topic exchanges permit wildcards in the binding key.
Keys are .-separated lists, e.g. “stocks.nyse.vmw”
* matches any 1 element.
# matches zero or more elements.



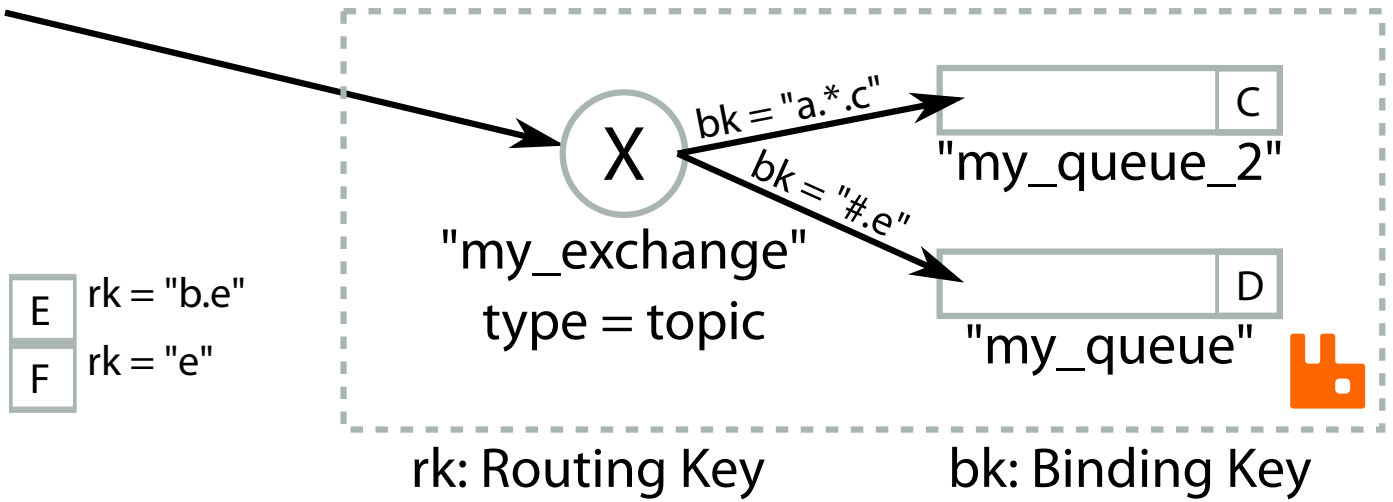
Topic exchanges permit wildcards in the binding key.
Keys are .-separated lists, e.g. “stocks.nyse.vmw”
* matches any 1 element.
# matches zero or more elements.



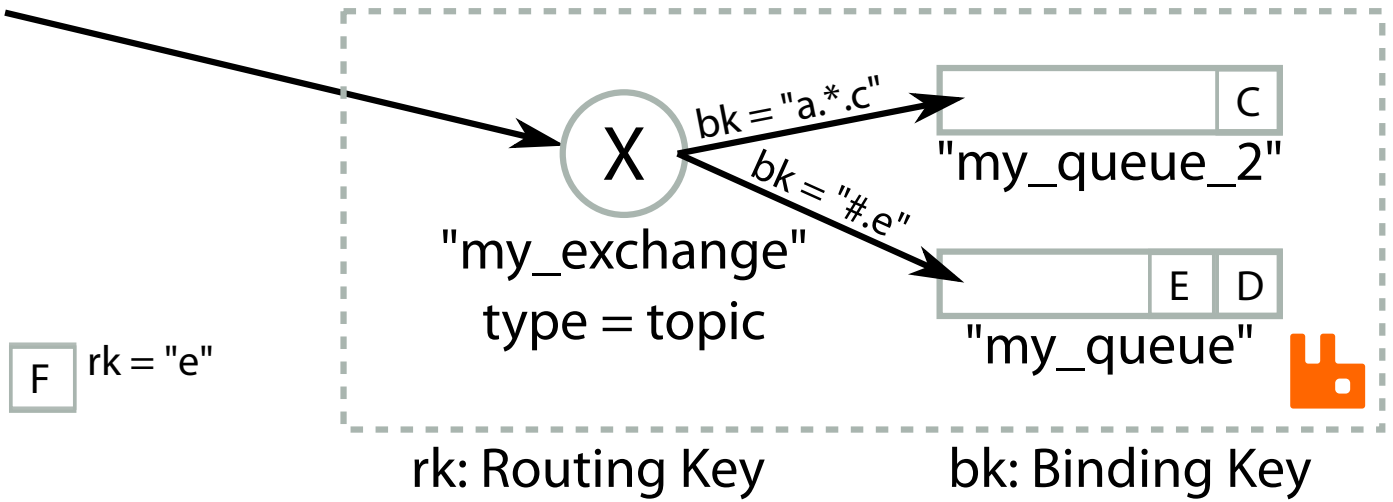
Topic exchanges permit wildcards in the binding key.
Keys are .-separated lists, e.g. “stocks.nyse.vmw”
* matches any 1 element.
# matches zero or more elements.



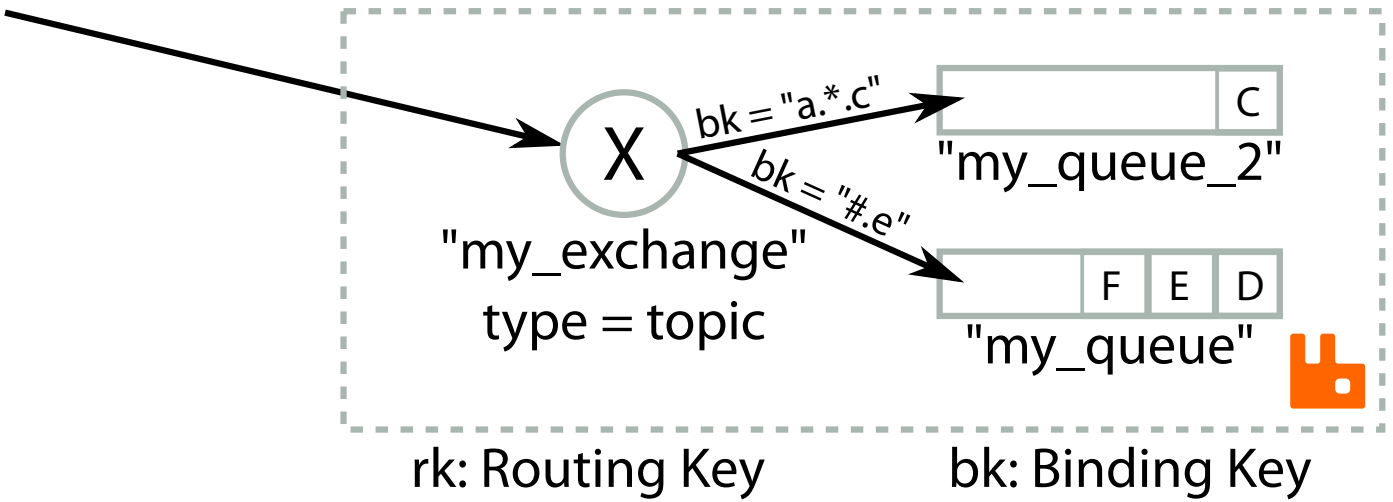
Topic exchanges permit wildcards in the binding key.
Keys are .-separated lists, e.g. “stocks.nyse.vmw”
* matches any 1 element.
# matches zero or more elements.



Topic exchanges permit wildcards in the binding key.
Keys are .-separated lists, e.g. “stocks.nyse.vmw”
* matches any 1 element.
# matches zero or more elements.



Topic exchanges permit wildcards in the binding key.
Keys are .-separated lists, e.g. “stocks.nyse.vmw”
* matches any 1 element.
# matches zero or more elements.



Other key elements of AMQP

RabbitMQ

Yes, it does all the standard boring stuff too

- Errors can be raised for messages that do not get routed to any queues
- Messages can be consumed so that the broker does not forget about the message until the client explicitly acknowledges the message
- Messages can be published with a property indicating whether the message should be written to disk
- Transactions: making publication and acknowledgement of several messages atomic
- Flow control: e.g. used to stop publishers from overwhelming the broker in extreme situations

Conclusion: Why AMQP?

RabbitMQ

It's called *Advanced*, so it must be good!

An Open *Protocol* is a good thing because it...

- Makes it easier to have multiple implementations that interoperate at the wire-level
- Avoids vendor lock-in: easy to rip out and replace individual components with alternative implementations
- Allows third-parties to write client libraries for other languages
- Decouples flag-day upgrades for both client libraries and broker
- Allows third-party traffic analysis tools to inspect and decode interactions between the clients and brokers
- Promotes similarities in APIs presented by different client libraries

RabbitMQ

RabbitMQ, AMQP & Messaging

- RabbitMQ consists of the *broker* (server) and several clients (Java, .Net, plus others)
- They speak the AMQP *protocol* to each other
- Anyone can write (and many have) other clients which also speak AMQP and work with RabbitMQ

RabbitMQ, AMQP & Messaging

- RabbitMQ consists of the *broker* (server) and several clients (Java, .Net, plus others)
- They speak the AMQP *protocol* to each other
- Anyone can write (and many have) other clients which also speak AMQP and work with RabbitMQ
- RabbitMQ is freely available and open source, licensed under the Mozilla Public License v1.1
- It's already in many popular Linux distributions (Ubuntu, Debian, Fedora, Gentoo) and can be easily installed on OS X both through MacPorts and Homebrew

Introducing RabbitMQ

RabbitMQ

Not actually a rabbit

- The broker is written in Erlang: an excellent programming language and platform for the task
- A mere 17k lines of code in the broker
- Supports clustering for increased scalability
- Supports several extension points via plugins which are frequently used to extend RabbitMQ, e.g. STOMP, XMPP adaptors; The Shovel; additional exchange types…
- Can work with Pacemaker and associated tools to provide various forms of High Availability

RabbitMQ Features

RabbitMQ

- A single queue can run up around 30,000 messages per second, depending on payload, clients, and properties of the messages
- But several queues together can achieve much higher throughput, and still keep the queues empty
- RabbitMQ 2.0 gains the ability to send messages to disk to free up memory, thus allowing queue depths to grow, bounded only by disk space, not RAM. Queues of 10s of millions of items are easily accommodated
- Adding extension to RabbitMQ based on feedback from our users, e.g. queue expiry, queue-message TTL, publisher acknowledgements

- RabbitMQ is a leading implementation of AMQP, and has wide adoption from a large community across a large number of languages and problem domains
- Website and downloads available at `http://www.rabbitmq.com/`
- On Ubuntu and Debian, just an `apt-get install rabbitmq-server` away. Similarly easy for Fedora and many other Linux distributions
- For OS X, it's in MacPorts
- Full easy-to-install Windows bundles available from the website

And finally...

RabbitMQ

Can I have a Pony too?

"All our messages are incredibly important and must never ever be lost under any circumstances".

And finally...

RabbitMQ

Can I have a Pony too?

"All our messages are incredibly important and must never ever be lost under any circumstances".

Fact: There are *always* moments at which the message is at a single point of failure.

And finally...

RabbitMQ

Can I have a Pony too?

"We need guaranteed exactly-once delivery – I want to send 1 message, and know it gets delivered to exactly one consumer, once".

And finally...

RabbitMQ

Can I have a Pony too?

"We need guaranteed exactly-once delivery – I want to send 1 message, and know it gets delivered to exactly one consumer, once".

Fact: Provably impossible.
Depends on definition of *guarantee*: you can achieve a high probability of no duplicates and no message loss.

Thank you

Questions?