# VMware Tanzu RabbitMQ

## An Overview

Dan Buchko, VMware Tanzu Solutions Engineer

July 16, 2020

# Introduction

# What is RabbitMQ?

- A long-running open-source project written in Erlang backed by VMware
- A general-purpose message broker designed with a smart broker / passive consumer model
- Enables cross-language messaging by implementing Advanced Message Queueing Protocol specification
- Support for range of additional protocols (STOMP, MQTT, AMQP, etc.)
- A tool with wide and extendable support (many client libraries, many plugins)

# Why RabbitMQ? [1/2]

- Most widely used & deployed open-source messaging tool with a proven track record
- Popular with a strong community that creates resources to extend and support its capabilities (e.g., community plug-ins)
- Built off of Erlang for high availability
- Pluggable authentication support (LDAP)
- Empowers developers with developer APIs for all actions (incl config)

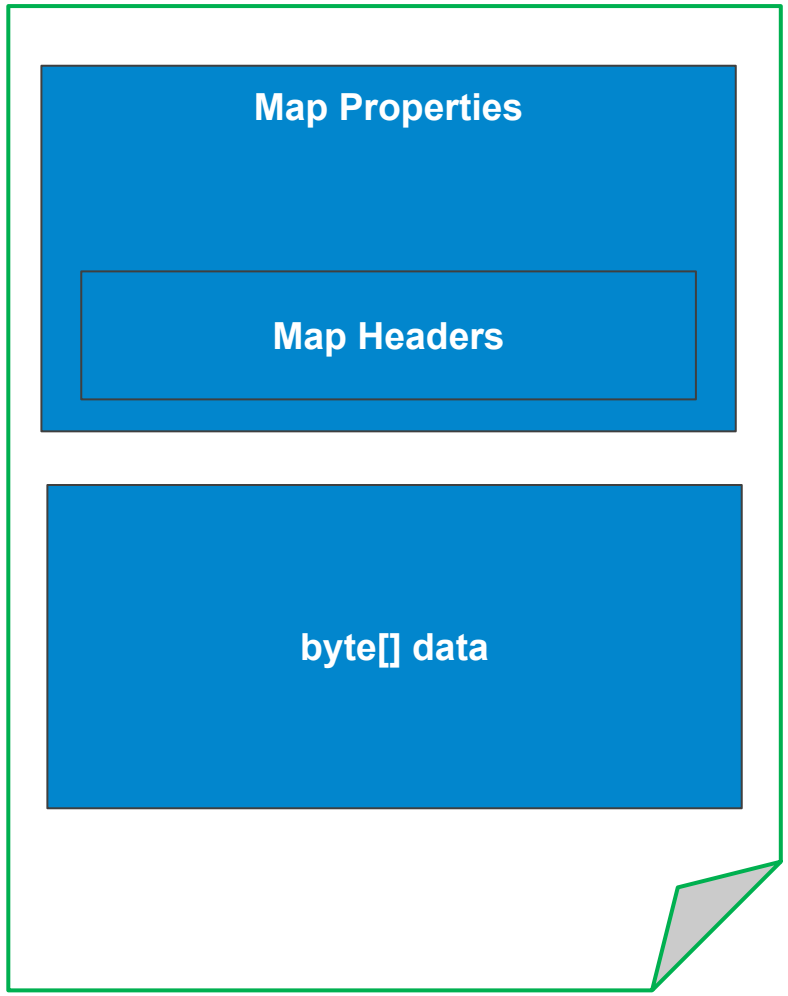
# Why RabbitMQ? [2/2]

- Routing from smart broker unlocks powerful messaging capabilities
- Flexibility for almost any language to share messages across operating systems and environments
- Get up & running in <20 minutes
- Extendibility in plug-ins and protocols means it can be customized for wide range of needs

# RabbitMQ Concepts

# RabbitMQ / AMQP Terminology

## Messages




Properties/headers are just Key-Value pairs

Immutable properties: message-id, user-id …
- routing-key
- delivery-mode (2 = persistent)
- content-type
- reply-to
- correlation-id

Application-specific properties

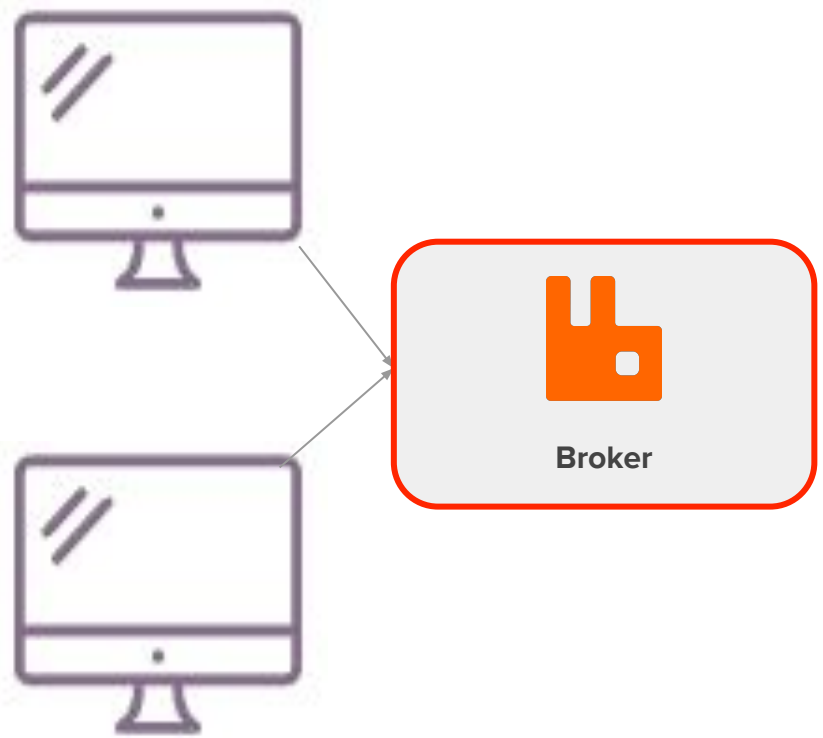
Data is opaque (just an array of bytes)

-> AMQP is language agnostic

# RabbitMQ / AMQP Terminology

## Producer / Publisher

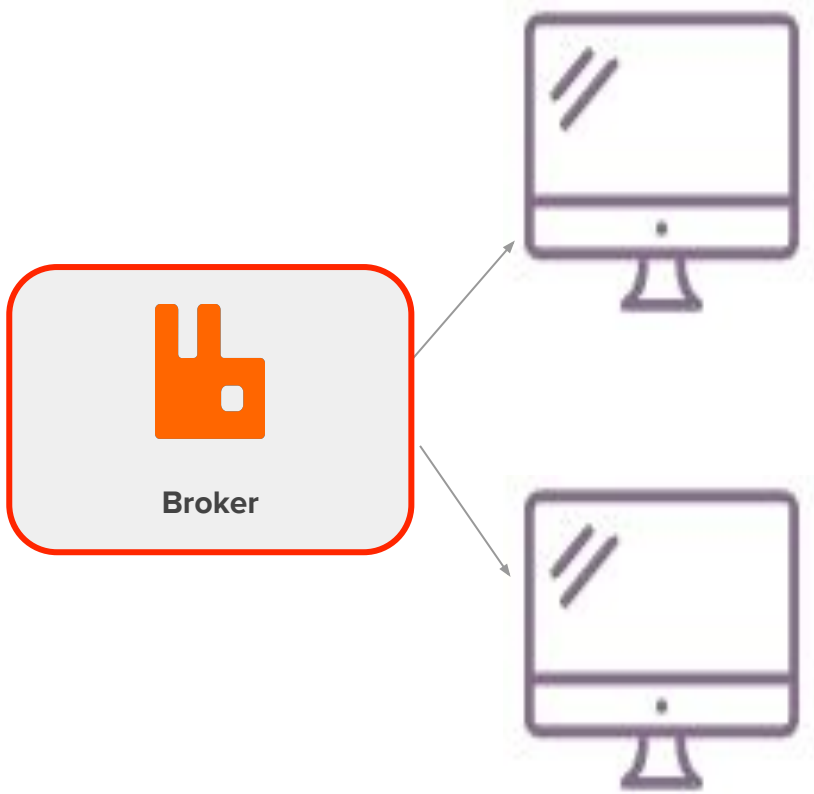
Applications that create messages and deliver them to RabbitMQ Broker.



# RabbitMQ / AMQP Terminology

## Consumer / Subscriber

Applications that receive messages from the broker



# RabbitMQ / AMQP Terminology

## Exchanges

Exchanges reside on the broker

Can be:

- Durable (persisted to disk)
- Transient (in memory)

Publishers always publish to an exchange

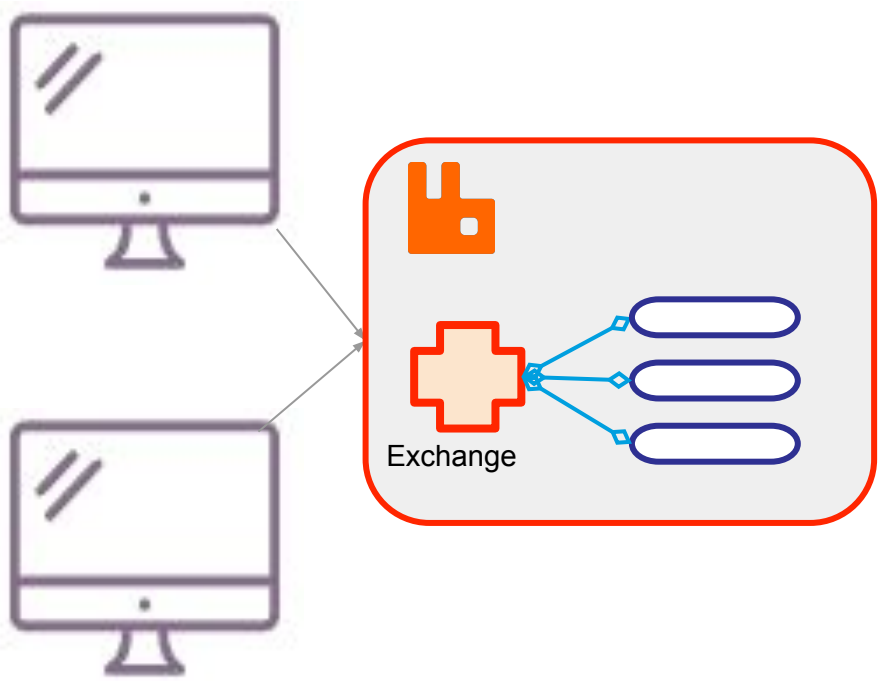
Responsible for routing messages based on type and message content

Stateless

Bound to queues

4 main types:

- Direct
- Fanout
- Topic
- Headers

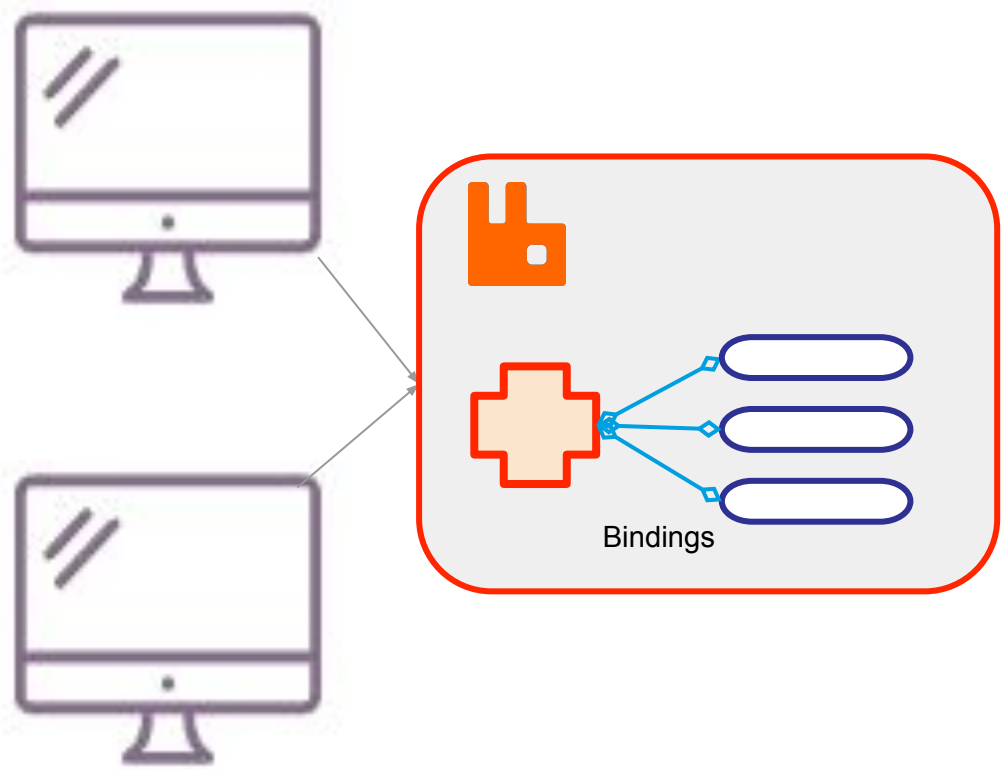


# RabbitMQ / AMQP Terminology

## Bindings

Define associations between resources, eg

- Exchanges bound to queues
- Exchanges bound to other exchanges



# RabbitMQ / AMQP Terminology

## Queues

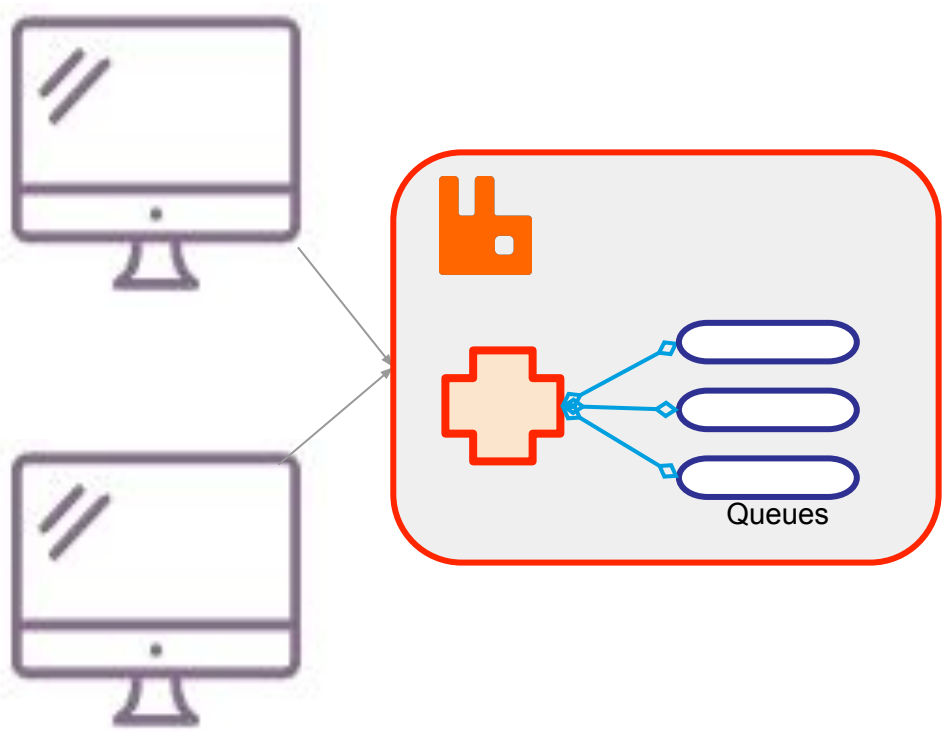
Store messages, i.e. stateful

Consumers always consume from queues

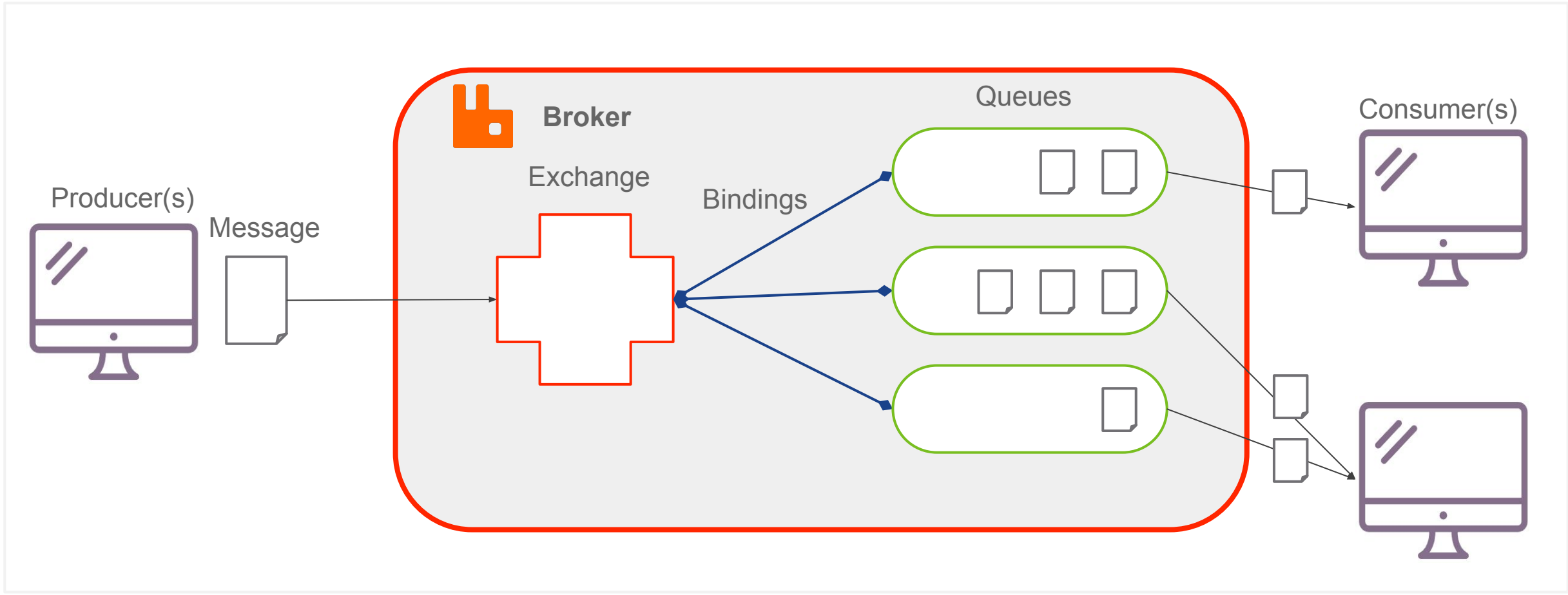
Can be configured to have range of behavior:

- Durable (persisted to disk) / transient
- Expiration
- Lazy
- HA
- Quorum
- etc.

HA and Quorum queues give control over reliability and performance.



# Putting it All Together



# Clustering

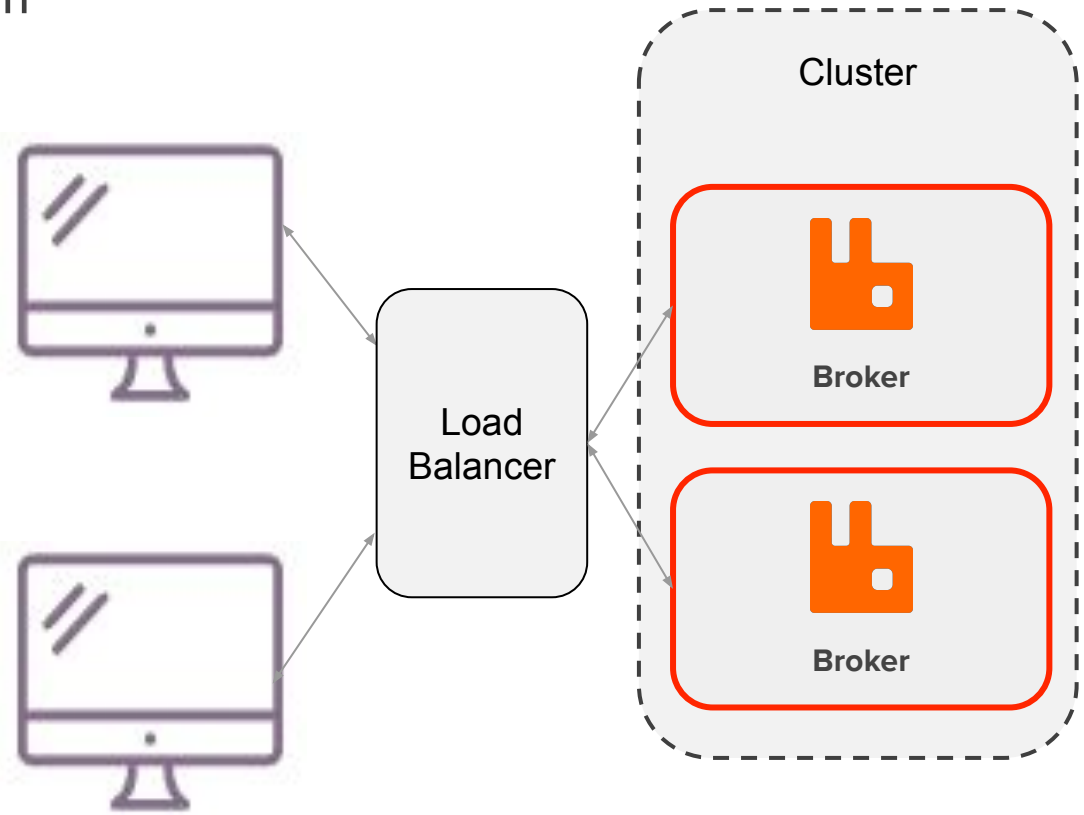
# Clustering

## Introduction

A Cluster is two or more nodes working together to form one logical broker

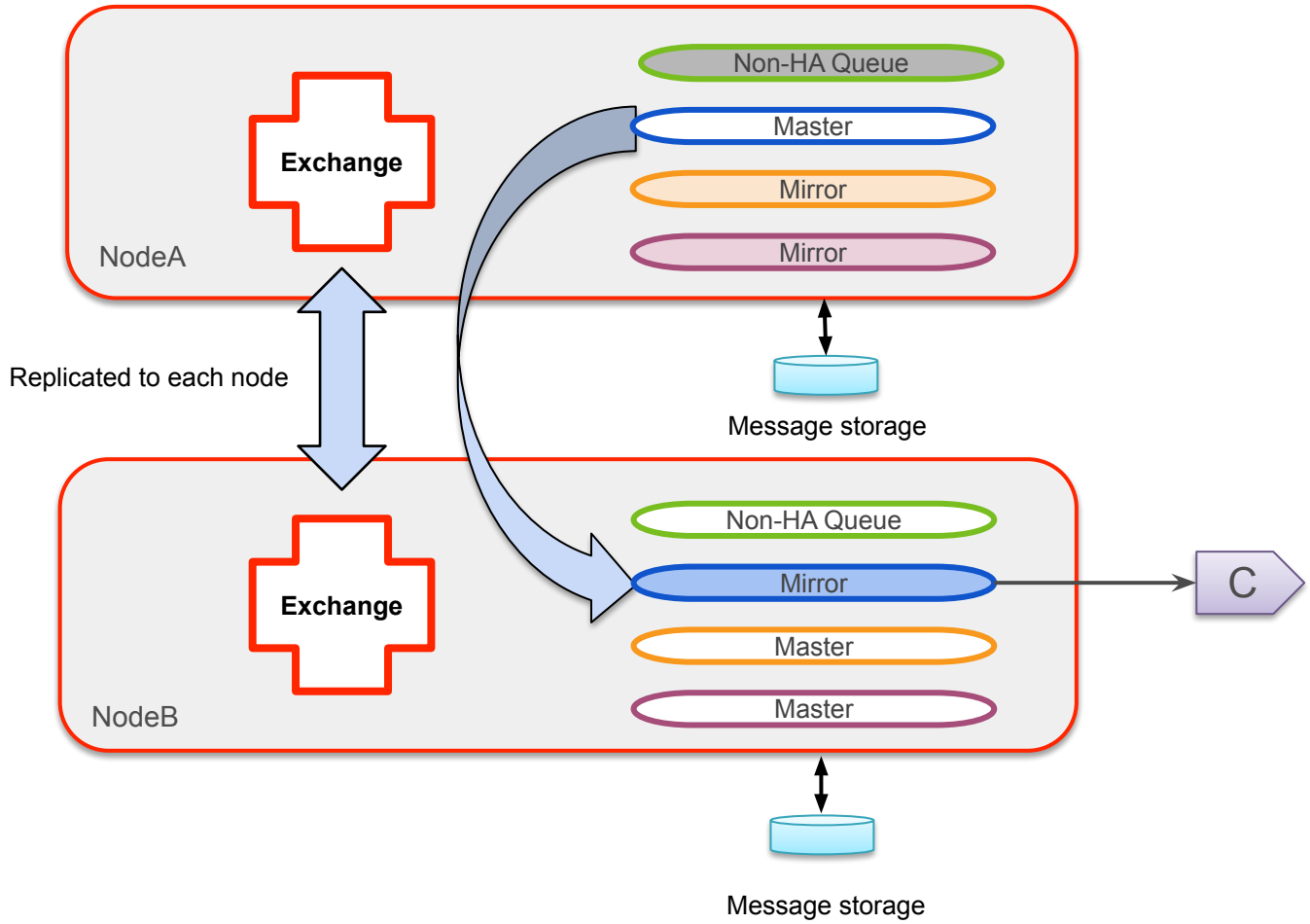
Clustering brings a number of benefits:

Improved performance (* if used wisely)
Multiple queues across multiple machines
Higher availability
Higher reliability
Mirrored queues



# A 2-Node Cluster

## HA vs Non-HA Queues




**Non-HA Queue**

- All messages are held on the node where it was created
- Other nodes act as proxies for clients which connect to them

**HA-Queue**

- Client connects to a node and creates a queue. The “master” for this queue is now on this node
- Mirror queues are created on other nodes in the cluster - how many is configurable by policy
- Mirrors hold copies of all messages
- Distribution of messages to consumers is managed by the master node - so there is always network traffic back to the master
- If the node holding the master queue fails, remaining mirrors hold an election and the winner becomes the new master

# High Availability

## What Happens If A Node Fails?

What happens if one node in a cluster fails?

- All messages in queues which reside on that node are lost unless messages are persistent
- Persistent messages can only be recovered if the failed node can be recovered

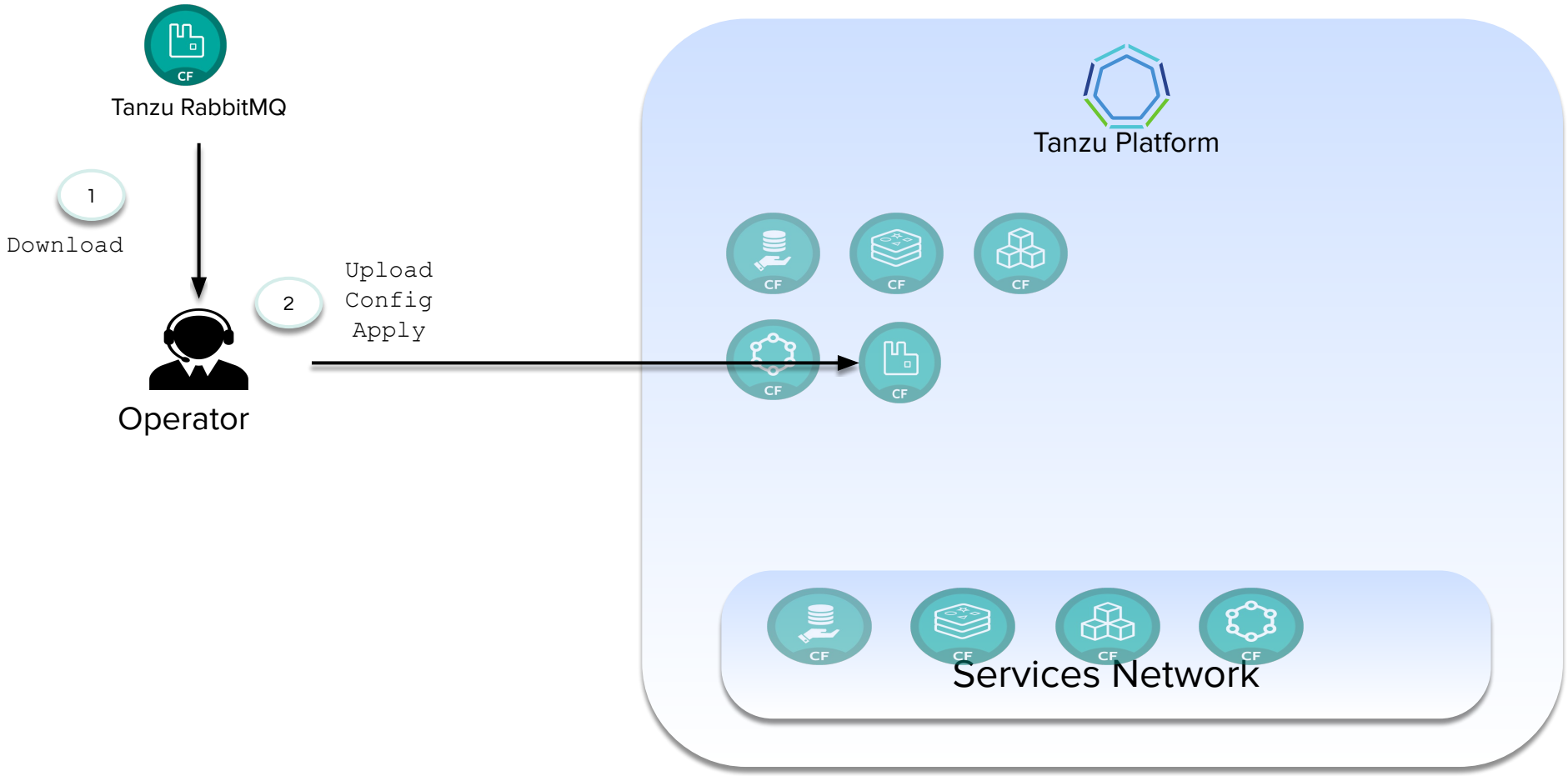
Solution – HA

- Replicates messages across other nodes in the cluster
- If a node dies, other nodes have copies of all messages
- Clients reconnect to the cluster and continue where they left off
- Configured by policy on a per-queue basis (`ha-mode={all,exactly,nodes}` )
- Transactions and Publisher Confirms span all mirrors

# Tanzu RabbitMQ (RabbitMQ on PCF)

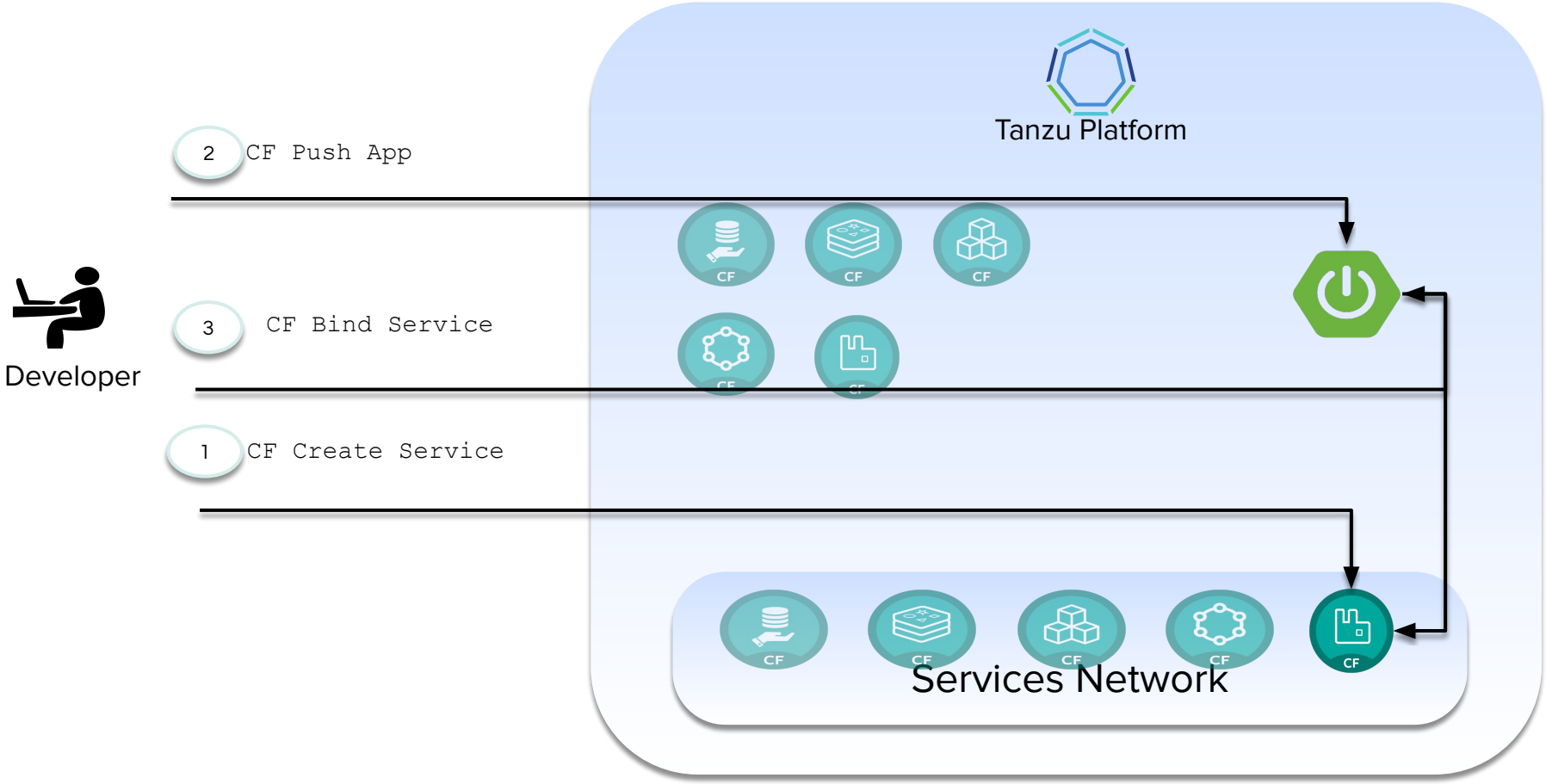
# RabbitMQ on Tanzu Platform

## Operator Flow



# RabbitMQ on Tanzu Platform

## Developer Flow

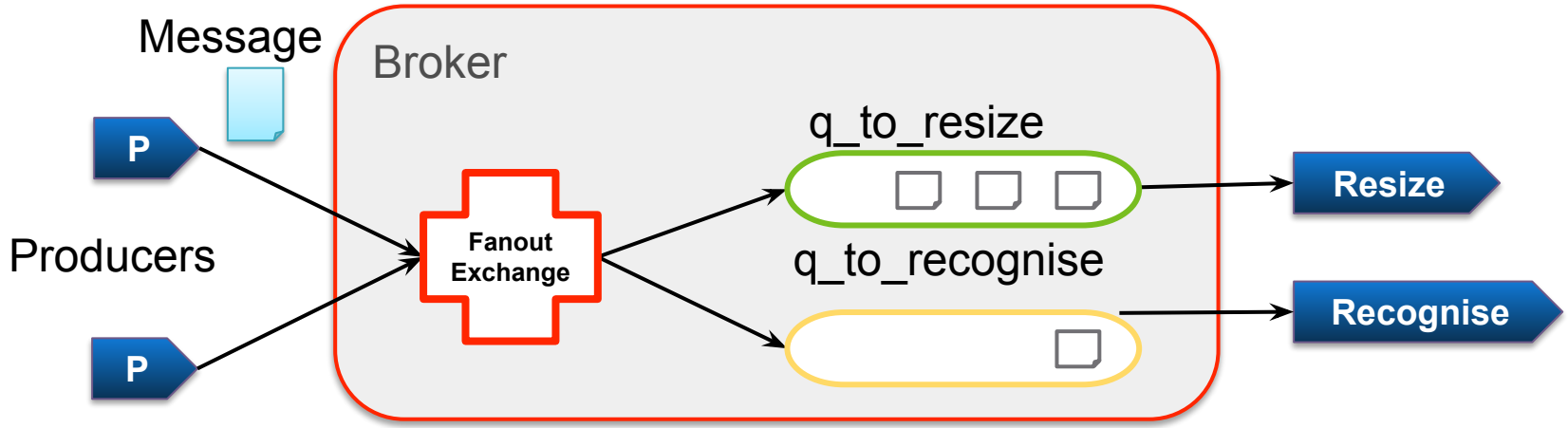


# Thank You!

# Appendix: Exchange Types

# Fanout Exchange

## Publish/Subscribe



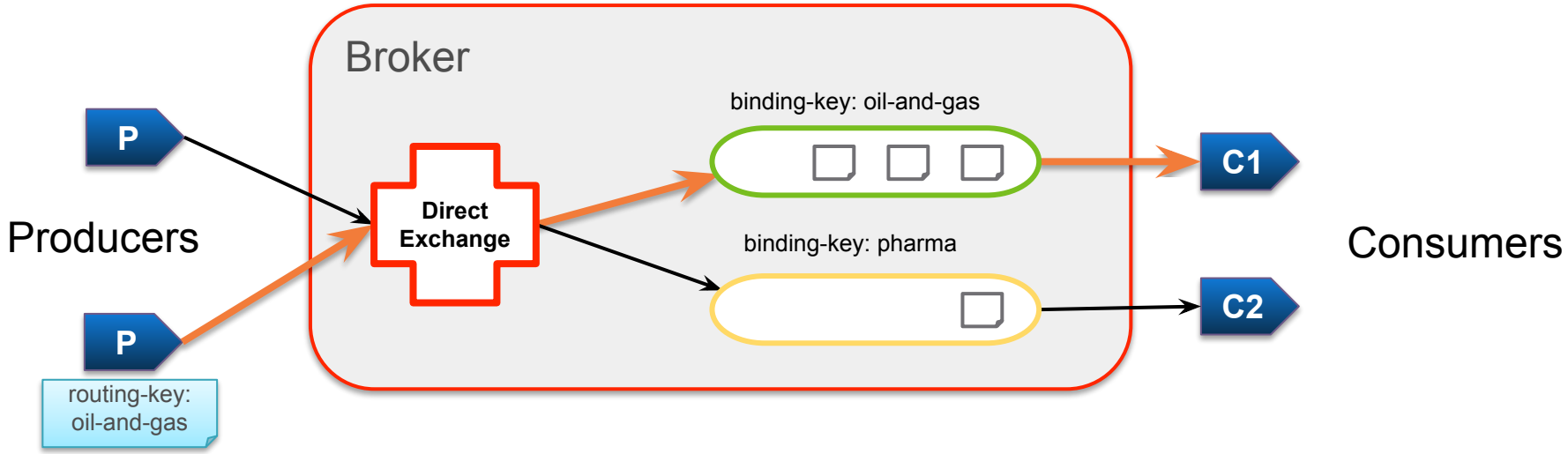

Every queue gets a copy of every message

**Upload a photo to Facebook.**

**Consumers resizing, doing face recognition…**

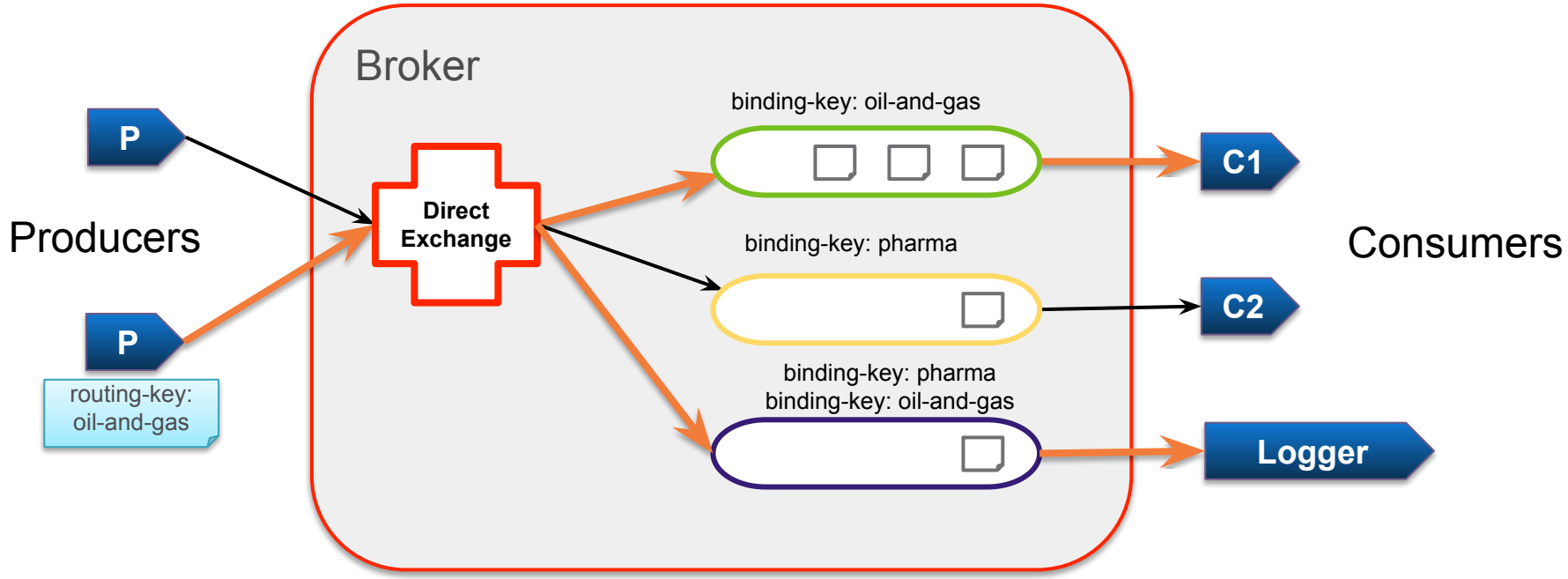
# Direct Exchange

## Routing – The Direct Exchange




- Queues ‘bind’ to the exchange
- A queue can have more than one binding-key
- A message is routed to each queue whose ‘binding-key’ matches the message’s ‘routing-key’

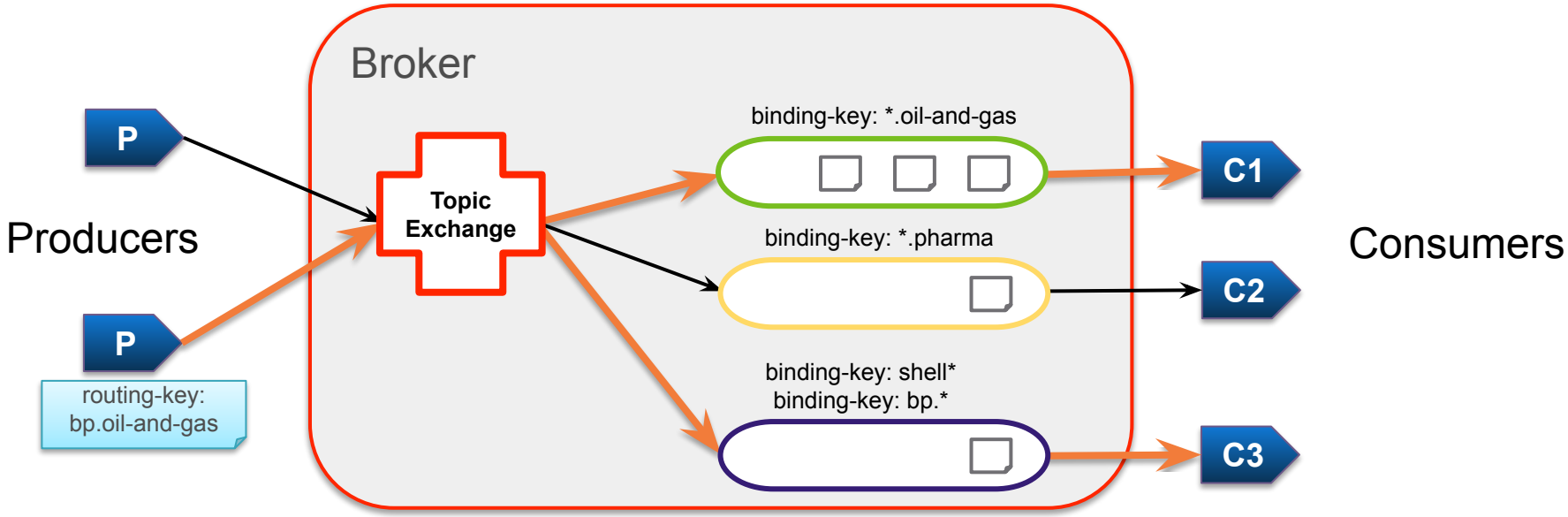
# Direct Exchange - Multiple Bindings




Different queues can use the same binding key

# Topic Exchange

## Pattern Matching The Routing-Key

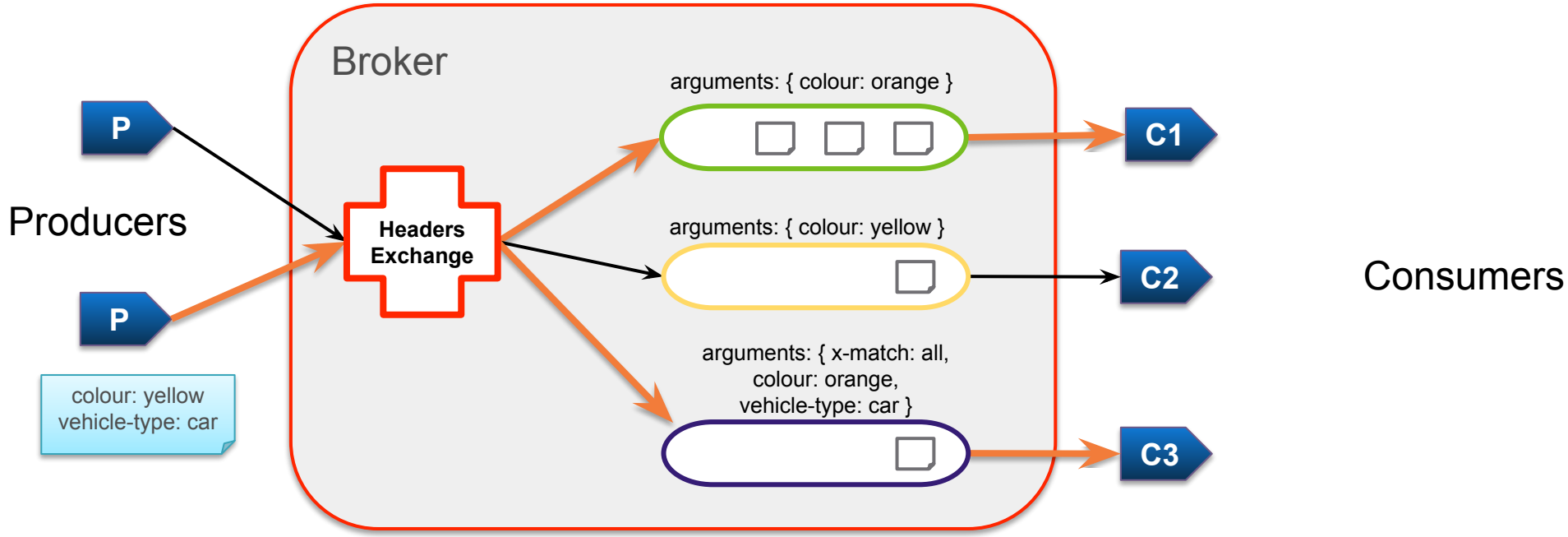



Bindings match word patterns
Words are separated by dots '.'

# - zero or more words
* - exactly one word

# Headers Exchange

## Not Just The Routing Key




Binding arguments specify a list of headers to match exactly
Optional x-match argument: ‘all’ or ‘any’ gives AND/OR