



Overview



Introduce the RabbitMQ platform

RabbitMQ management portal

AMQP Protocol

Exchanges

Queues, bindings and consumers

Client support

Docker setup

Demo

Introducing RabbitMQ

Reliability

Routing

Clustering and
high availability

Management
web interface

Command
line interface

Introducing RabbitMQ



RabbitMQ management portal

Declare, list and delete RabbitMQ entities

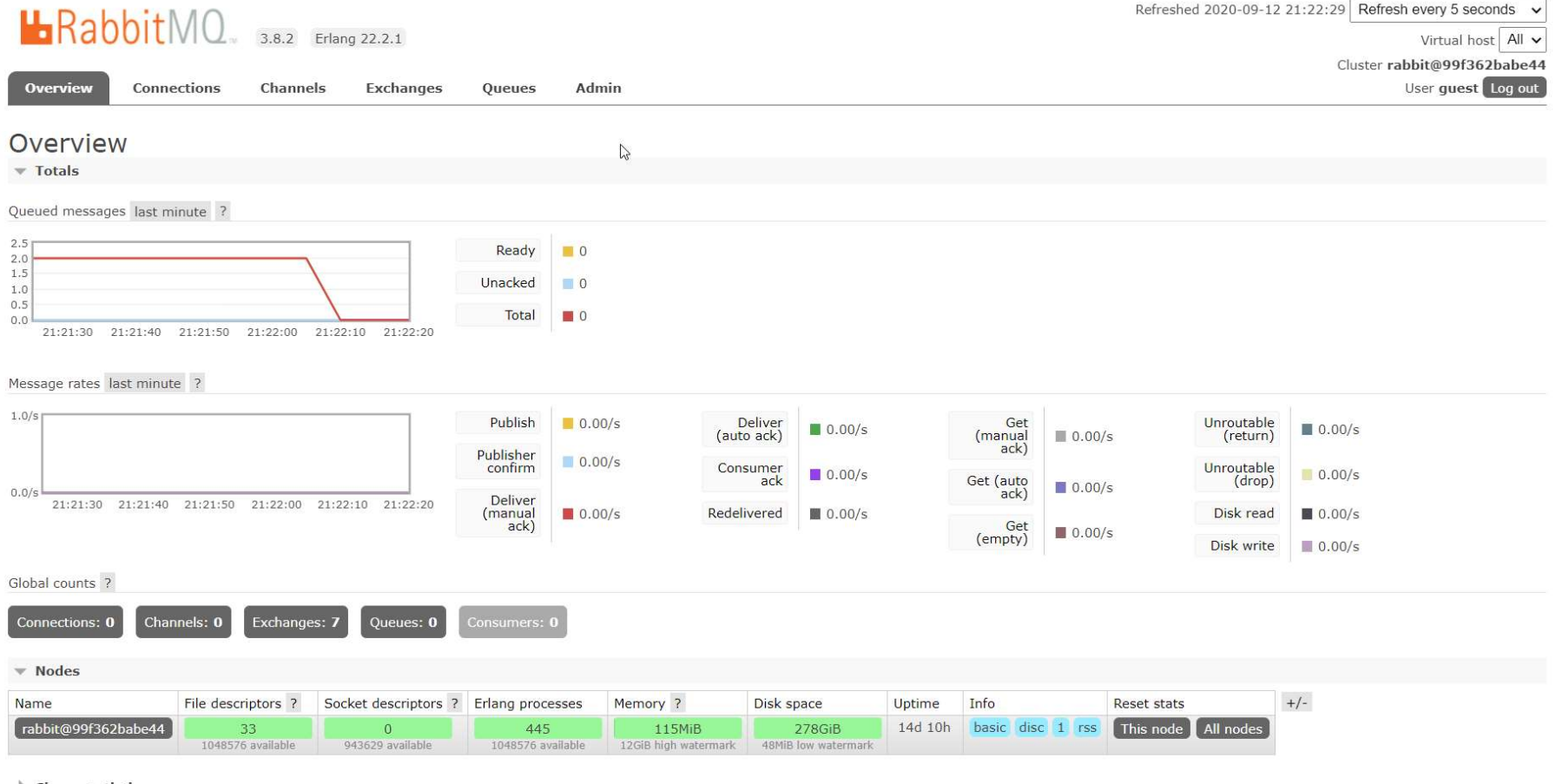
Queue and exchange monitoring

Send and receive messages

Monitor Erlang processes, file descriptors, and memory use

Force close connections, and purge queues

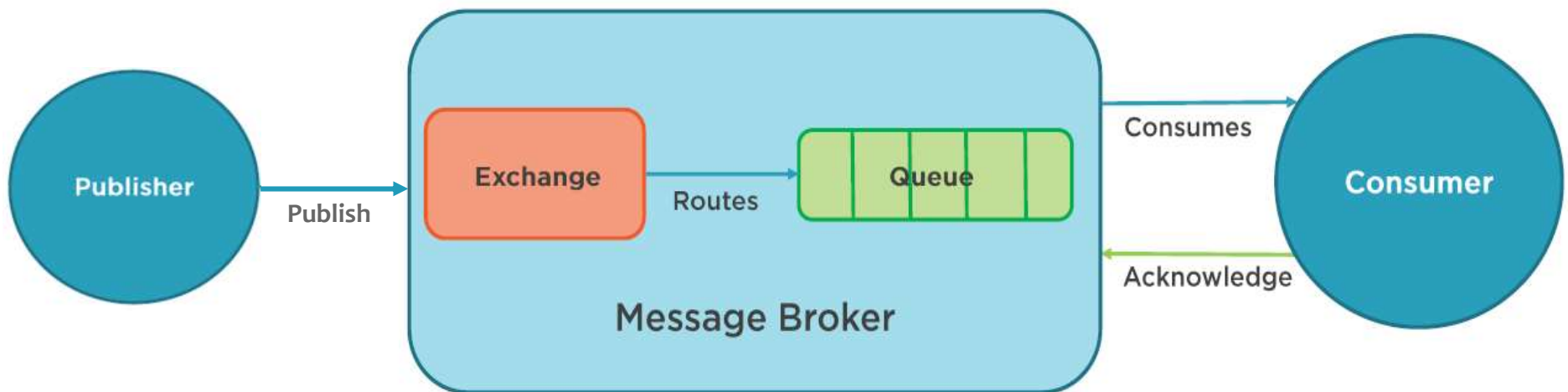
RabbitMQ management portal



AMQP Protocol

Advance Message Queueing Protocol

Supports version 0-9-1



Exchanges

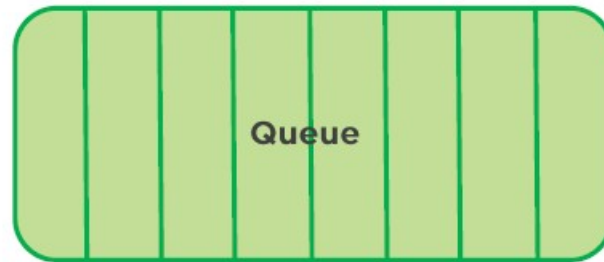
Direct Exchanges

Fanout Exchanges

Topic Exchanges

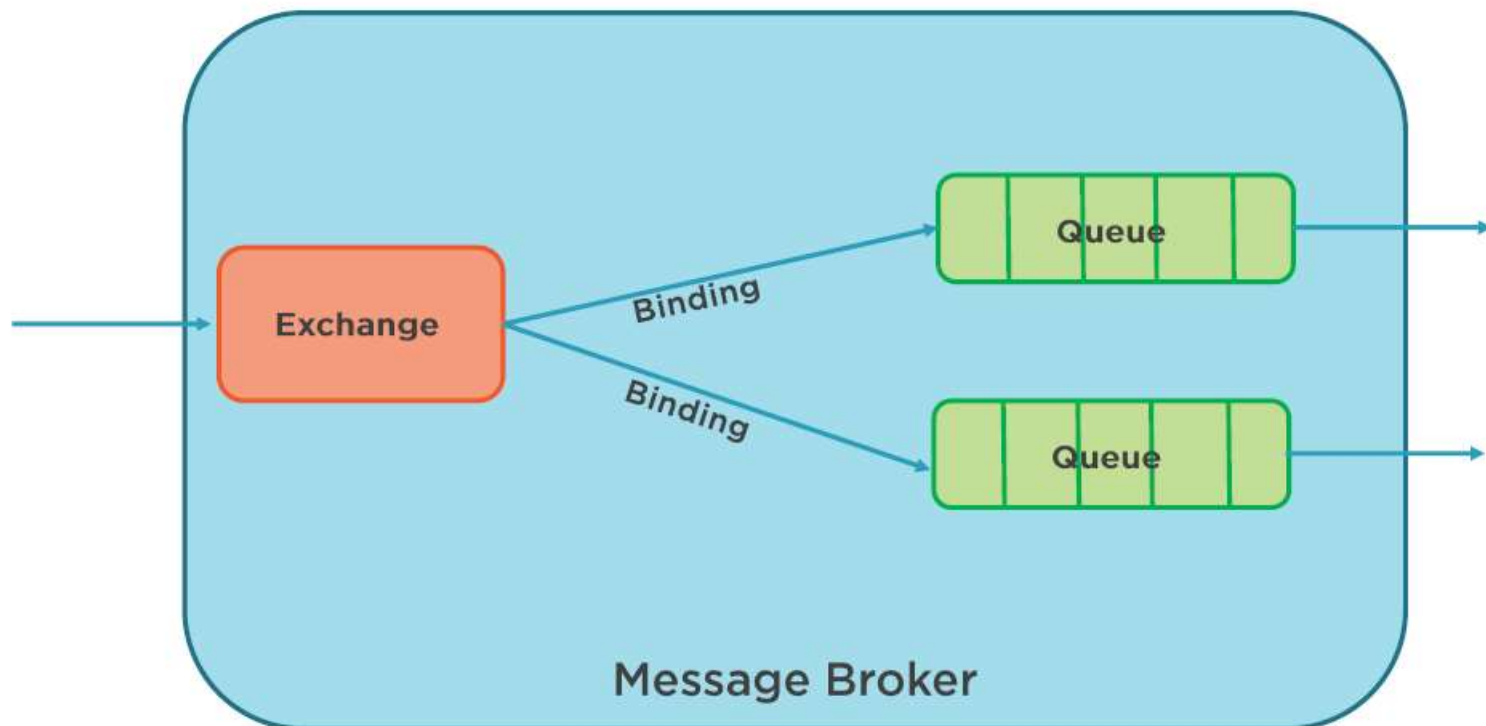
Header Exchanges

Queue

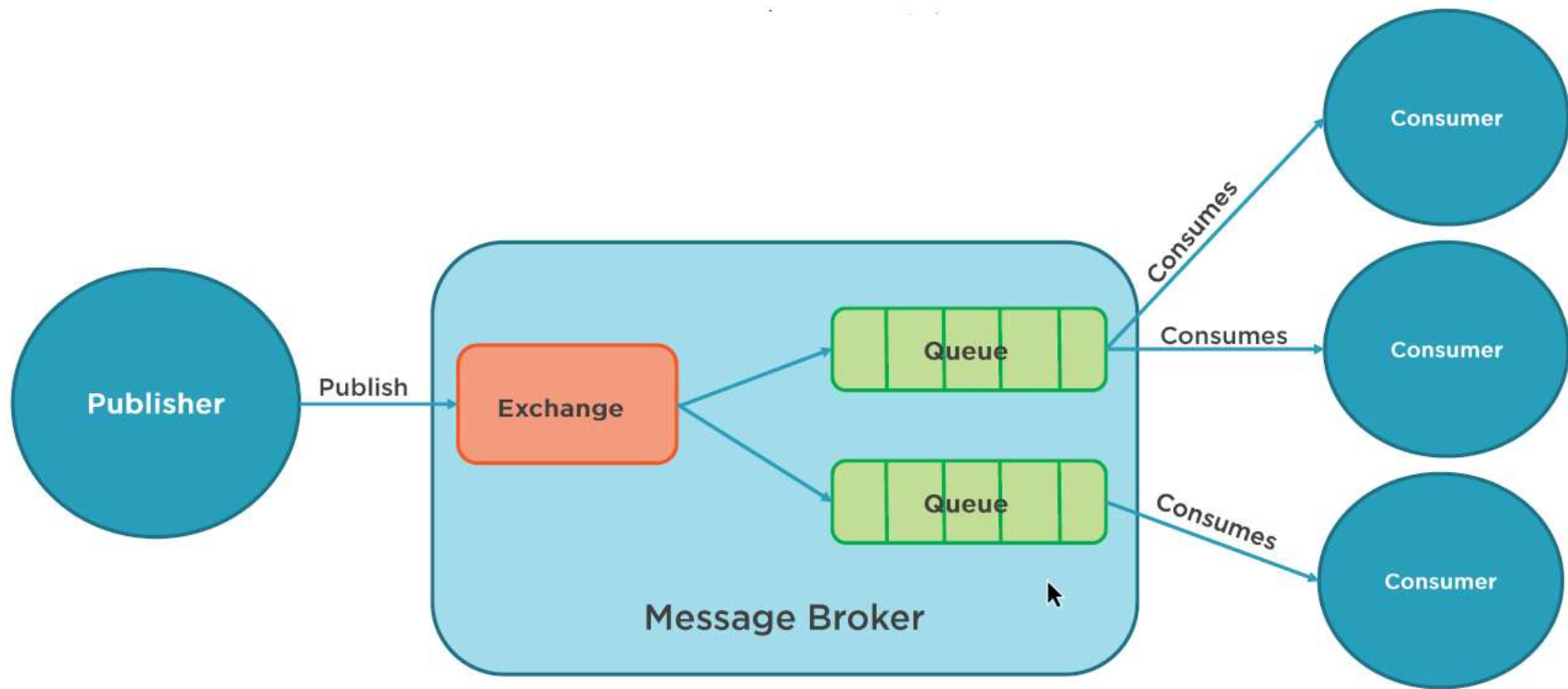


Name	<i>The name of the queue</i>
Durable	<i>Persisting the queue to disk</i>
Exclusive	<i>Delete queue when not needed</i>
Auto Delete	<i>Queue deleted when consumer unsubscribes</i>

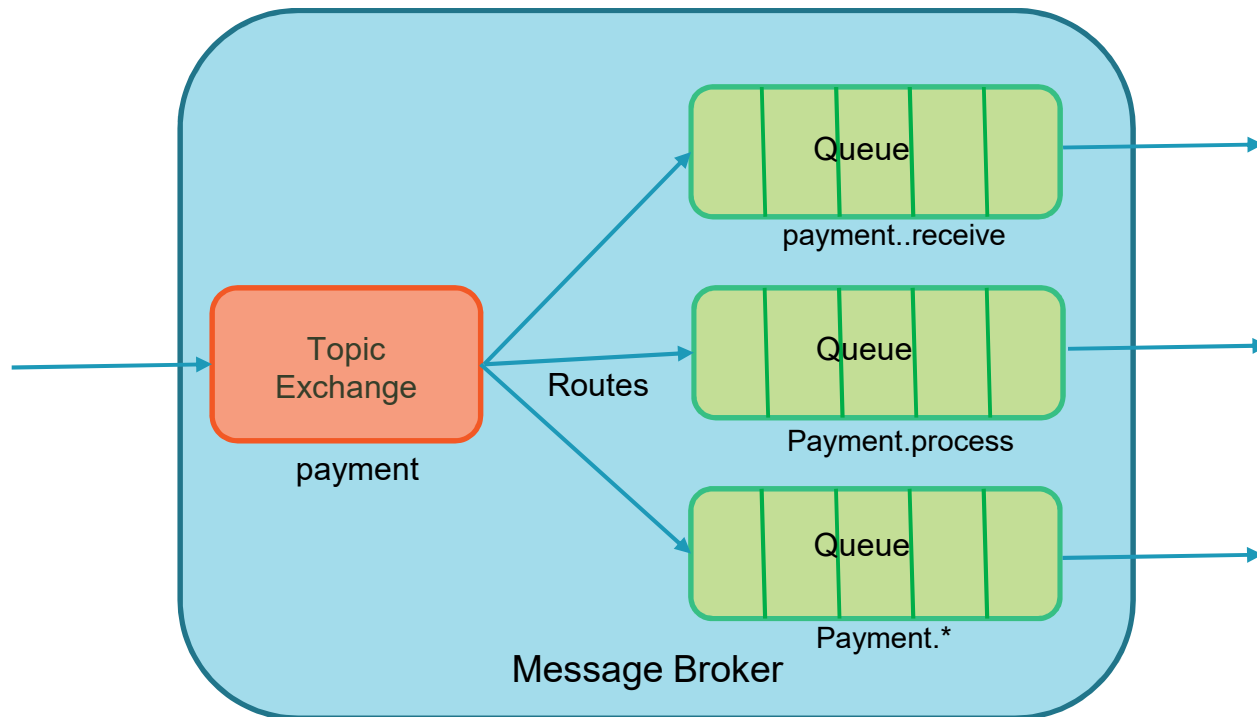
Binding



Consumer



Topic Exchanges



Client support

[Features](#)[Get Started](#)[Support](#)[Community](#)[Docs](#)[Blog](#)

Introduction

RabbitMQ is a message broker: it accepts and forwards messages. You can think about it as a post office: when you put the mail that you want posting in a post box, you can be sure that Mr. or Ms. Mailperson will eventually deliver the mail to your recipient. In this analogy, RabbitMQ is a post box, a post office and a postman.

The major difference between RabbitMQ and the post office is that it doesn't deal with paper, instead it accepts, stores and forwards binary blobs of data – *messages*.

RabbitMQ, and messaging in general, uses some jargon.

- *Producing* means nothing more than sending. A program that sends messages is a *producer*:



- A *queue* is the name for a post box which lives inside RabbitMQ. Although messages flow through RabbitMQ and your applications, they can only be stored inside a *queue*. A *queue* is only bound by the host's memory & disk limits, it's essentially a large message buffer. Many *producers* can send messages that go to one queue, and many *consumers* can try to receive data from one *queue*. This is how we represent a queue:

<https://www.rabbitmq.com/tutorials/tutorial-one-java.html>

Prerequisites

This tutorial assumes RabbitMQ is installed and running on `localhost` on standard port (`5672`). In case you use a different host, port or credentials, connections settings would require adjusting.

Where to get help

If you're having trouble going through this tutorial you can contact us through the mailing list.

1 "Hello World!"

The simplest thing that does *something*

[Python](#)[Java](#)[Ruby](#)[PHP](#)[C#](#)[JavaScript](#)[Go](#)[Elixir](#)[Objective-C](#)[Swift](#)[Spring AMQP](#)

2 Work queues

Distributing tasks among workers (the [competing consumers](#))

RabbitMQ Docker

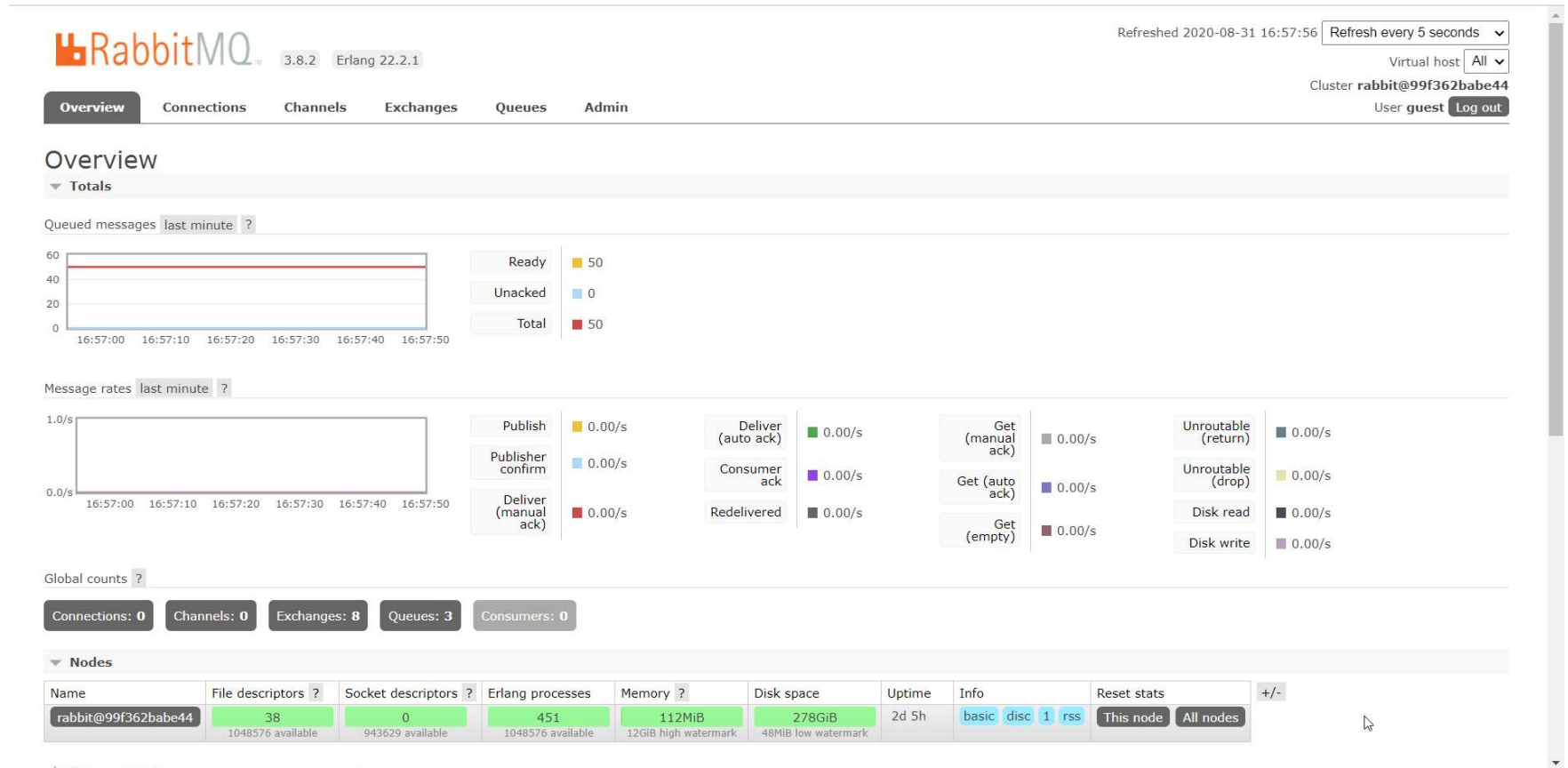
```
$ cat docker-compose.yml
version: "3.3"
services:
  rabbitmq:
    image: rabbitmq:3-management
    container_name: rabbitmq
    networks:
      - cluster
    volumes:
      - /i[REDACTED]/rabbitmq/data:/var/lib/rabbitmq/
      - /[REDACTED]/rabbitmq/logs:/var/log/rabbitmq/
    environment:
      RABBITMQ_DEFAULT_USER: ${RABBITMQ_DEFAULT_USER}
      RABBITMQ_DEFAULT_PASS: ${RABBITMQ_DEFAULT_PASS}
    ports:
      - 5672:5672
      - 15672:15672
networks:
  cluster:
    $
$
$
```

RabbitMQ Docker

- 4369: [epmd](#), a peer discovery service used by RabbitMQ nodes and CLI tools
- 5672, 5671: used by AMQP 0-9-1 and 1.0 clients without and with TLS
- 25672: used for inter-node and CLI tools communication (Erlang distribution server port) and is allocated from a dynamic range (limited to a single port by default, computed as AMQP port + 20000). Unless external connections on these ports are really necessary (e.g. the cluster uses [federation](#) or CLI tools are used on machines outside the subnet), these ports should not be publicly exposed. See [networking guide](#) for details.
- 35672-35682: used by CLI tools (Erlang distribution client ports) for communication with nodes and is allocated from a dynamic range (computed as server distribution port + 10000 through server distribution port + 10010). See [networking guide](#) for details.
- 15672: [HTTP API](#) clients, [management UI](#) and [rabbitmqadmin](#) (only if the [management plugin](#) is enabled)
- 61613, 61614: [STOMP clients](#) without and with TLS (only if the [STOMP plugin](#) is enabled)
- 1883, 8883: [MQTT clients](#) without and with TLS, if the [MQTT plugin](#) is enabled
- 15674: STOMP-over-WebSockets clients (only if the [Web STOMP plugin](#) is enabled)
- 15675: MQTT-over-WebSockets clients (only if the [Web MQTT plugin](#) is enabled)
- 15692: Prometheus metrics (only if the [Prometheus plugin](#) is enabled)

<https://www.rabbitmq.com/networking.html#ports>

RabbitMQ Management Portal



RabbitMQ Management Portal



3.8.2 Erlang 22.2.1

Refreshed 2020-08-31 16:59:24 Refresh every 5 seconds

Virtual host All

Cluster rabbit@99f362babe44

User guest Log out

Overview Connections Channels Exchanges Queues Admin

Exchanges

▼ All exchanges (8)

Pagination

Page 1 of 1 - Filter: ☐ Regex ?

Displaying 8 items , page size up to: 100

Name	Type	Features	Message rate in	Message rate out	+/-
(AMQP default)	direct	D			
amq.direct	direct	D			
amq.fanout	fanout	D			
amq.headers	headers	D			
amq.match	headers	D			
amq.rabbitmq.trace	topic	D I			
amq.topic	topic	D			
shuttle	topic	D	0.00/s	0.00/s	

▼ Add a new exchange

Name:

Type: direct

Durability: Durable

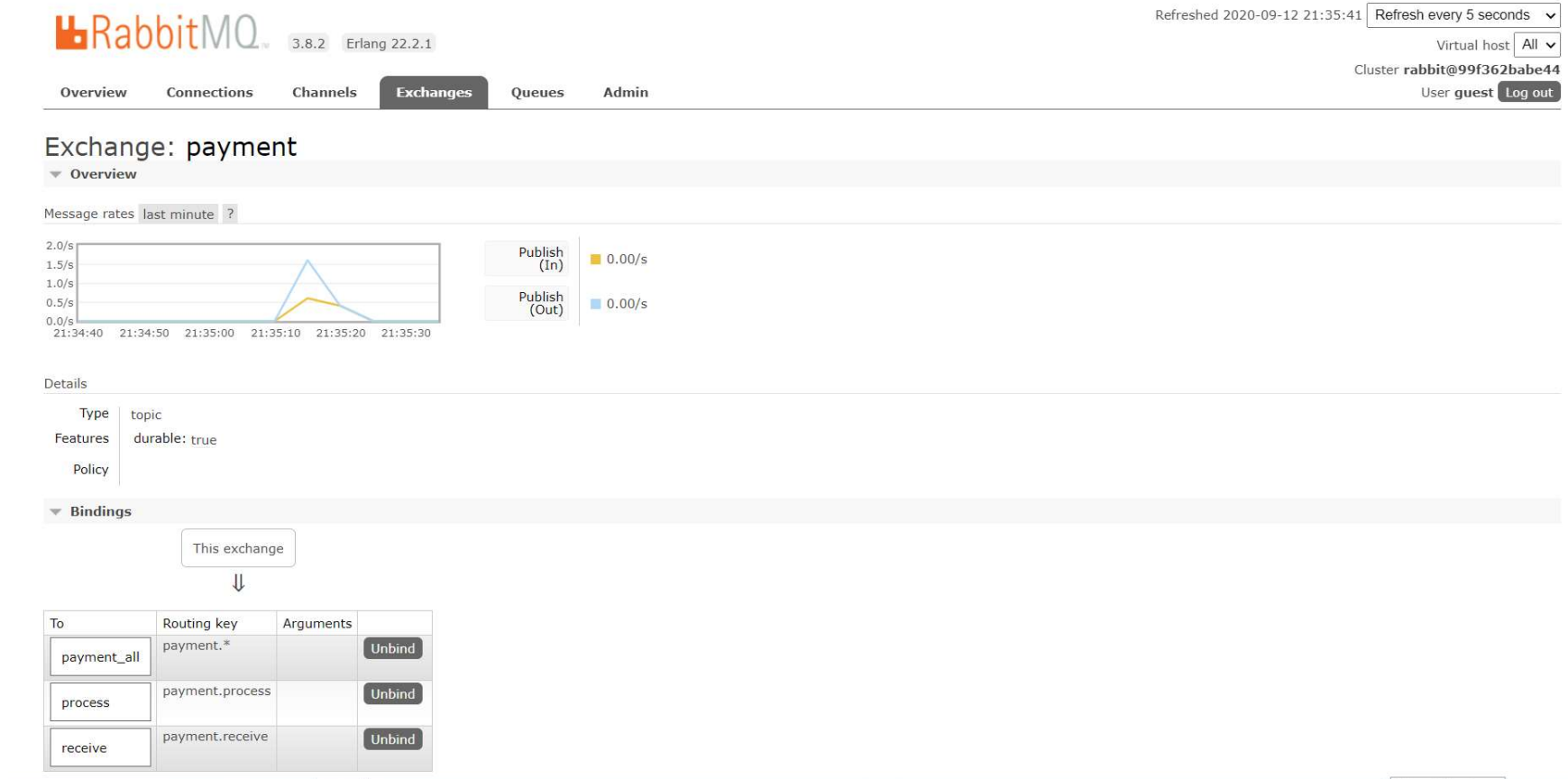
Auto delete: ? No

Internal: ? No

Arguments: = String

Add Alternate exchange ?

RabbitMQ Management Portal



RabbitMQ Management Portal



3.8.2 Erlang 22.2.1

Refreshed 2020-09-12 21:36:48

Refresh every 5 seconds

Virtual host All

Cluster rabbit@99f362babe44

User guest Log out

Overview Connections Channels Exchanges **Queues** Admin

Queues

▼ All queues (3)

Pagination

Page 1 of 1 - Filter: ☐ Regex

Displaying 3 items , page size up to: 100

Overview				Messages			Message rates			+/-
Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	
payment_all	classic		idle	25	0	25	0.00/s			
process	classic		idle	5	0	5	0.00/s			
receive	classic		idle	20	0	20	0.00/s			

▼ Add a new queue

Type: Classic

Name:

Durability: Durable

Auto delete: No

Arguments: = String

Add Message TTL ? | Auto expire ? | Max length ? | Max length bytes ? | Overflow behaviour ?
Dead letter exchange ? | Dead letter routing key ? | Single active consumer ? | Maximum priority ?
Lazy mode ? | Master locator ?

Add queue

RabbitMQ Management Portal



3.8.2 Erlang 22.2.1

Refreshed 2020-09-12 21:37:59 Refresh every 5 seconds

Virtual host All

Cluster rabbit@99f362babe44

User guest Log out

Overview Connections Channels Exchanges Queues Admin

Queue receive

Overview

Queued messages last minute ?



Ready 20
Unacked 0
Total 20

Message rates last minute ?



Publish 0.00/s

Bindings

From	Routing key	Arguments
(Default exchange binding)	payment.receive	

This queue

Add binding to this queue

From exchange: *
Routing key:
Arguments: = String

Bind

Publish message

Message will be published to the default exchange with routing key **receive**, routing it to this queue.

Delivery mode: 1 - Non-persistent
Headers: = String
Properties: =
Payload:

Details

Features	durable: true	State	idle	Messages ?	20	Total	20	Ready	20	Unacked	0	In memory	20	Persistent	20	Transient, Paged Out	0
Policy		Consumers	0	Message body bytes ?	1.4kiB		1.4kiB		1.4kiB		0iB		1.4kiB		1.4kiB		0iB
Operator policy		Consumer utilisation ?	0%	Process memory ?	31kiB												
Effective policy definition																	

Bindings

From	Routing key	Arguments
(Default exchange binding)	payment.receive	

Demo

RabbitMQ management portal

Exchanges

Queues

Publish and subscribe

Administration

Docker container

Summary



Introduce the RabbitMQ platform

RabbitMQ management portal

AMQP Protocol

Exchanges

Queues, bindings and consumers

Client support

Docker setup

Demo

