Part 3: RabbitMQ for beginners - The management interface

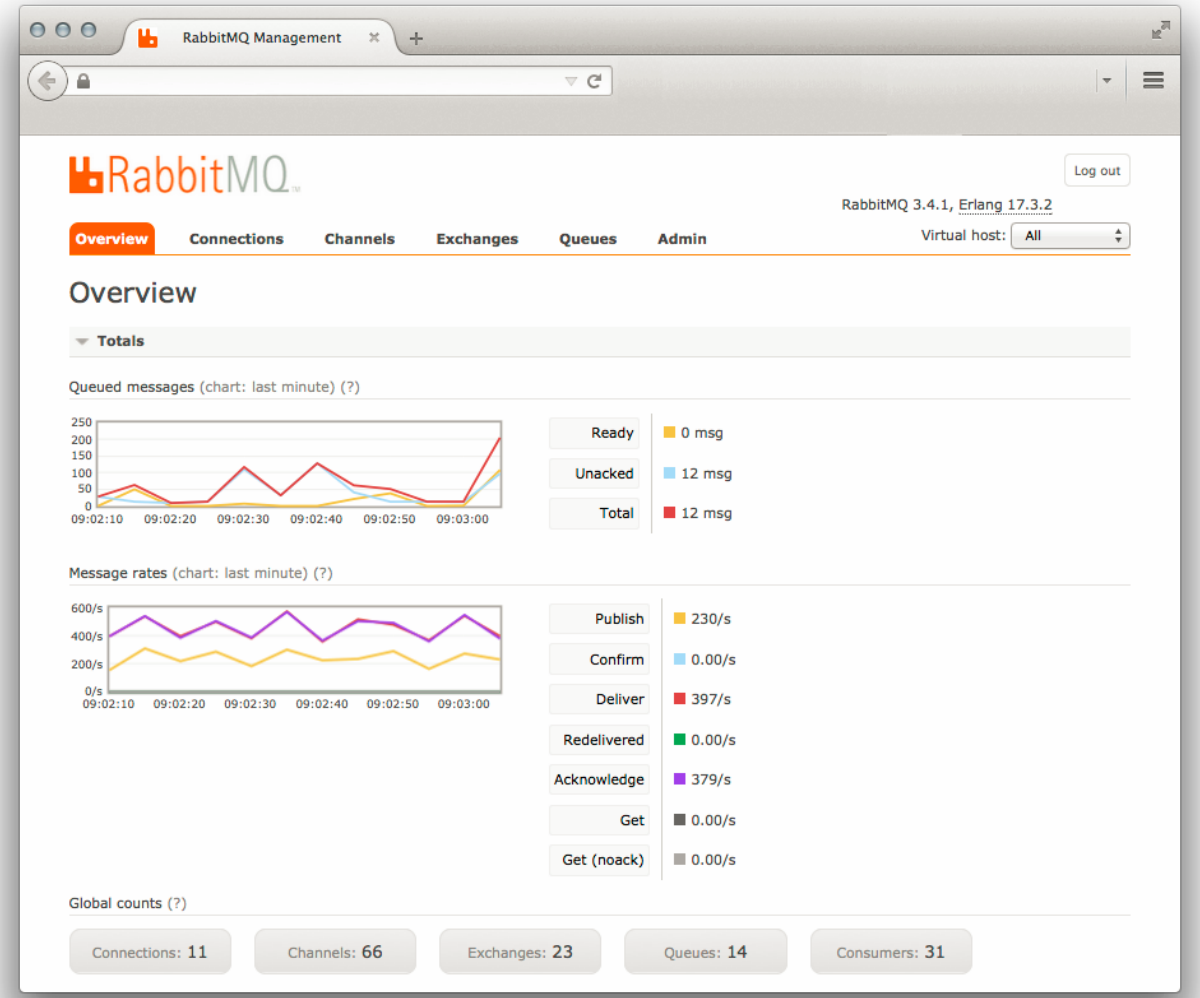
BY LOVISA JOHANSSON (MAILTO:LOVISA@CLOUDAMQP.COM) POSTED 2015-05-27

The RabbitMQ Management is a user-friendly interface that let you monitor and handle your RabbitMQ server from a web browser. Among other things queues, connections, channels, exchanges, users and user permissions can be handled - created, deleted and listed in the browser. You can monitor message rates and send/receive messages manually. This post give information about the different views that you can find in the RabbitMQ Management.

RabbitMQ Management is a plugin that can be enable for RabbitMQ. It gives a single static HTML page that makes background queries to the HTTP API for RabbitMQ. Information from the management interface can be useful when you are debugging your applications or when you need an overview of the whole system. If you see that the number of unacked messages starts to get high, it could mean that your consumers are getting slow. If you need to check if an exchange is working, you can try to send a test message.

A link to the RabbitMQ management interface can be found on the details page for your hosted RabbitMQ solution, your CloudAMQP instance. If you have RabbitMQ installed on localhost, go to `http://localhost:15672/` to find the management page.

All the tabs from the menu are explained in this post. Screenshots from the views are shown for: Overview, Connections and channels, Exchanges, Queues and Admin - users and permissions. A simple example will also show how to set up a queue an exchange and add a binding between them.



Concepts

In the prior posts Part 1: RabbitMQ for beginners - What is RabbitMQ?, the default virtual host, user and permissions (/blog/2015-05-18-part1-rabbitmq-for-beginners-what-is-rabbitmq.html) were used in the examples. In this post different views in the management interface are shown and all the examples are still only working with the default values - still there are some important concept that are good to be familiar with.

- **Users:** Users can be added from the management interface and every user can be assigned permissions such as rights to read, write and configure privileges. Users can also be assigned permissions to specific virtual hosts.

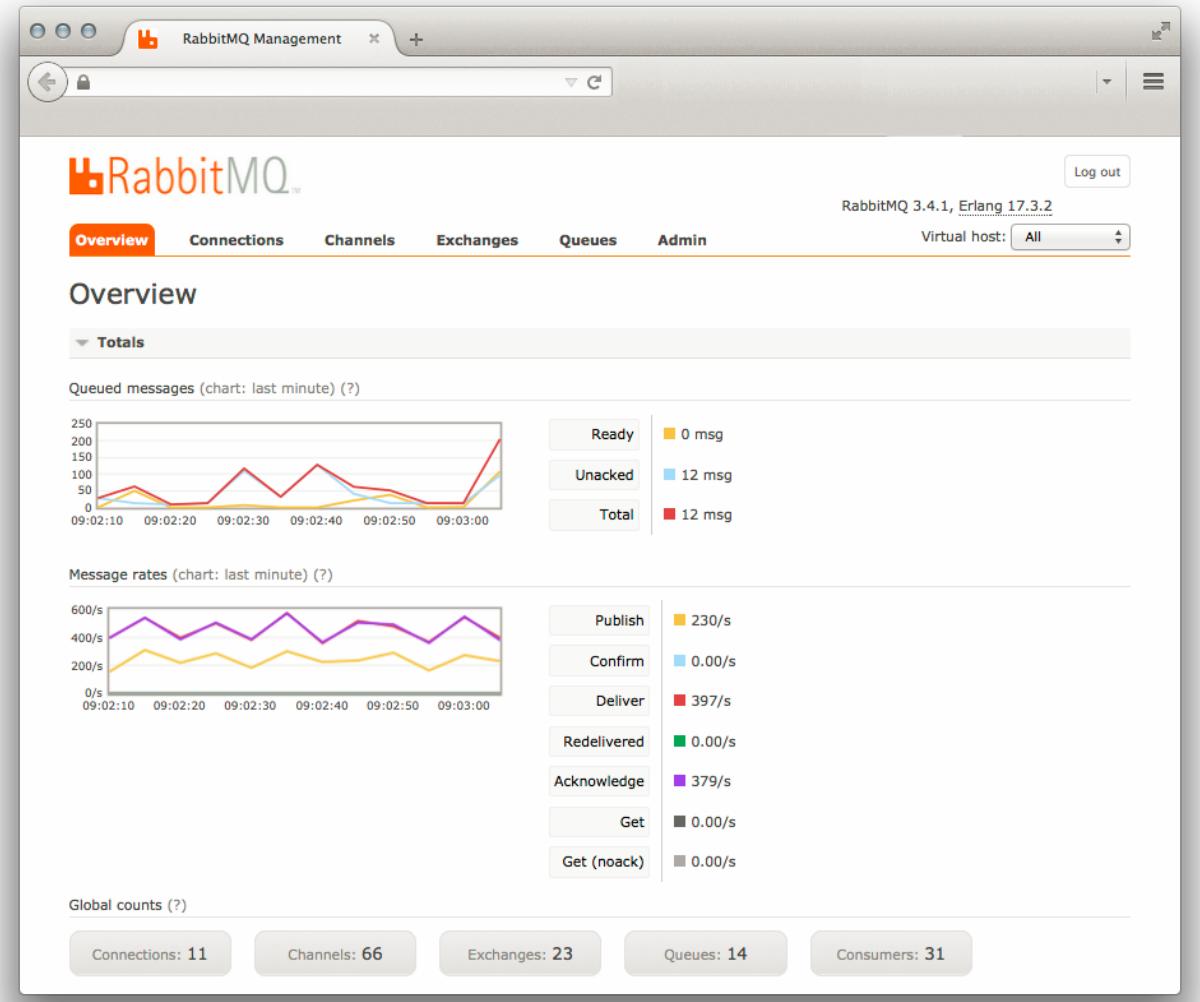
- **Vhost, virtual host:** Virtual hosts provide a way to segregate applications using the same RabbitMQ instance. Different users can have different access privileges to different vhost and queues and exchanges can be created so they only exists in one vhost.

- **Cluster:** A cluster consists of a set of connected computers that work together. If the RabbitMQ instance consisting of more than one node - it is called a RabbitMQ cluster. A cluster is a group of nodes i.e., a group of computers.

- **Node:** A node is a single computer the RabbitMQ cluster.

Overview

The overview shows two charts, one for queued messages and one with the message rate. You can change the time interval shown in the chart by pressing the text *(chart: last minute)* above the charts. Information about all different statuses for messages can be found by pressing (?).



Queued messages

A chart of the total number of queued messages for all your queues. **Ready** show the number of messages that are available to be delivered. **Unacked** are the number of messages for which the server is waiting for acknowledgement.

Messages rate

A chart with the rate of how the messages are handled. **Publish** show the rate at which messages are

entering the server and **Confirm** show a rate at which the server is confirming.

Global Count

The total number of connections, channels, exchanges, queues and consumers for ALL virtual hosts the current user has access to.

Nodes

Nodes show information about the different nodes in the RabbitMQ cluster (a cluster is a group of nodes i.e, a group of computers), or information about one single node if just one node is used. Here can information about server memory, number of erlang processes per node and other node-specific information be found. **Info** show i.e. further information about the node and enabled plugins.

Node

Node: rabbit@localhost **(More about this node)**

File descriptors (?)	Socket descriptors (?)	Erlang processes	Memory	Disk space	Info	+/-
	6 138 available	539 1048576 available	49MB 1.6GB high watermark	22GB 48MB low watermark	Disc 9 Stats	

Port and contexts

Listening ports for different protocols can be found here. More information about the protocols will be found in a later part of RabbitMQ for beginners.

Ports and contexts

Listening ports

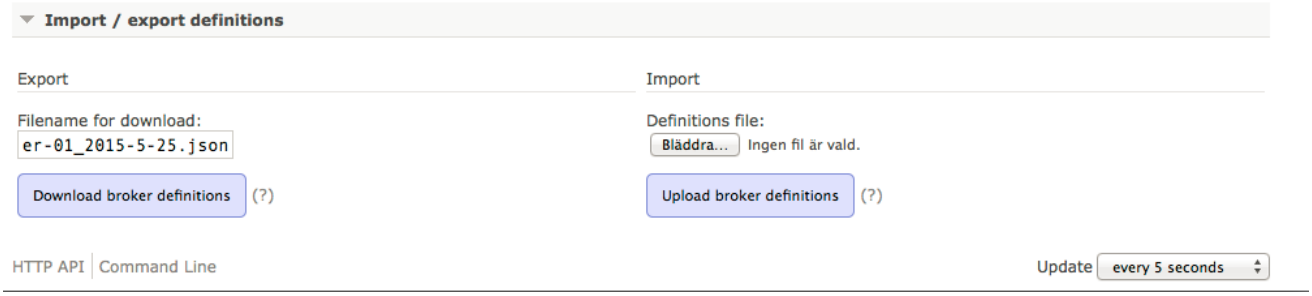
Protocol	Bound to	Port
amqp	0.0.0.0	5672
amqp/ssl	0.0.0.0	5671
clustering	::	25672
mqtt	0.0.0.0	1883
mqtt/ssl	0.0.0.0	8883
stomp	0.0.0.0	61613
stomp/ssl	0.0.0.0	61614

Web contexts

Context	Bound to	Port	SSL	Path
RabbitMQ Management	127.0.0.1	15672	○	/

Import export definitions

It is possible to import and export configuration definitions. When you download the definitions, you get a JSON representation of your broker (your RabbitMQ settings). This can be used to restore exchanges, queues, virtual hosts, policies and users. This feature can be used as a backup. Every time you make a change in the config, you can keep the old settings just in case.



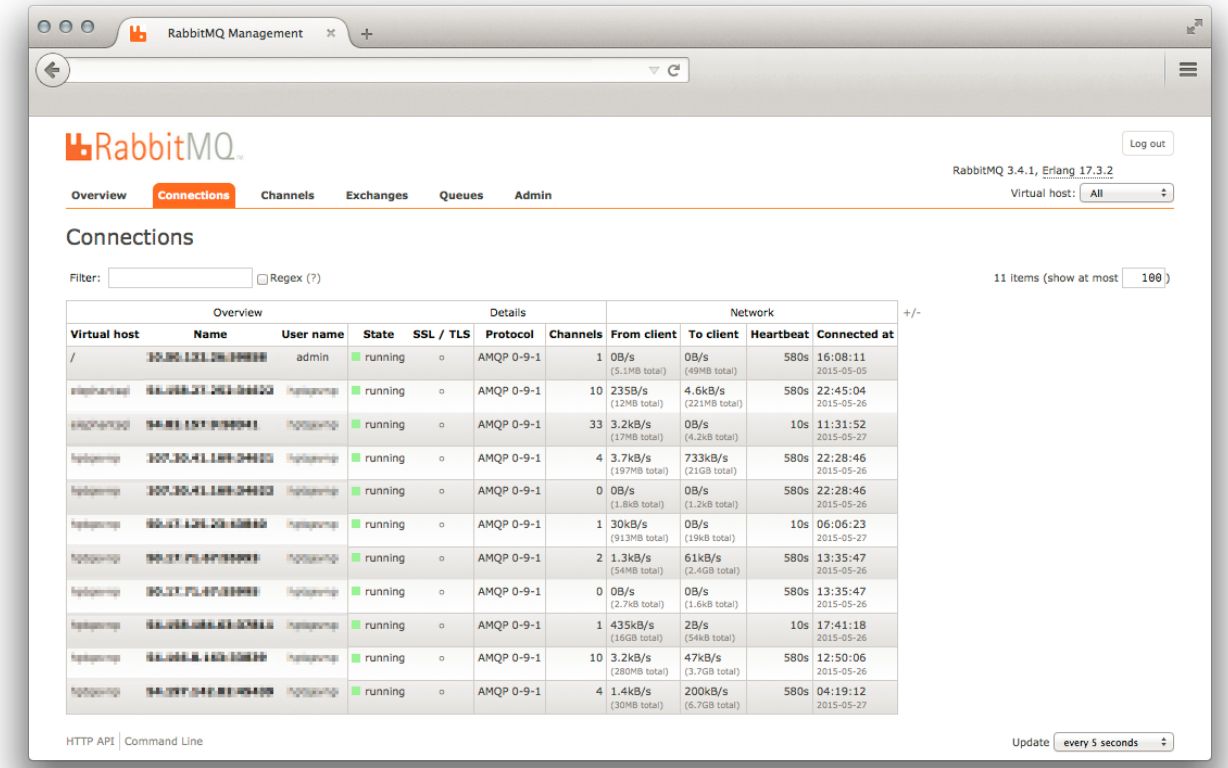
Connections and channels

A connection is a TCP connection between your application and the RabbitMQ broker. A channel is a virtual connection inside a connection.

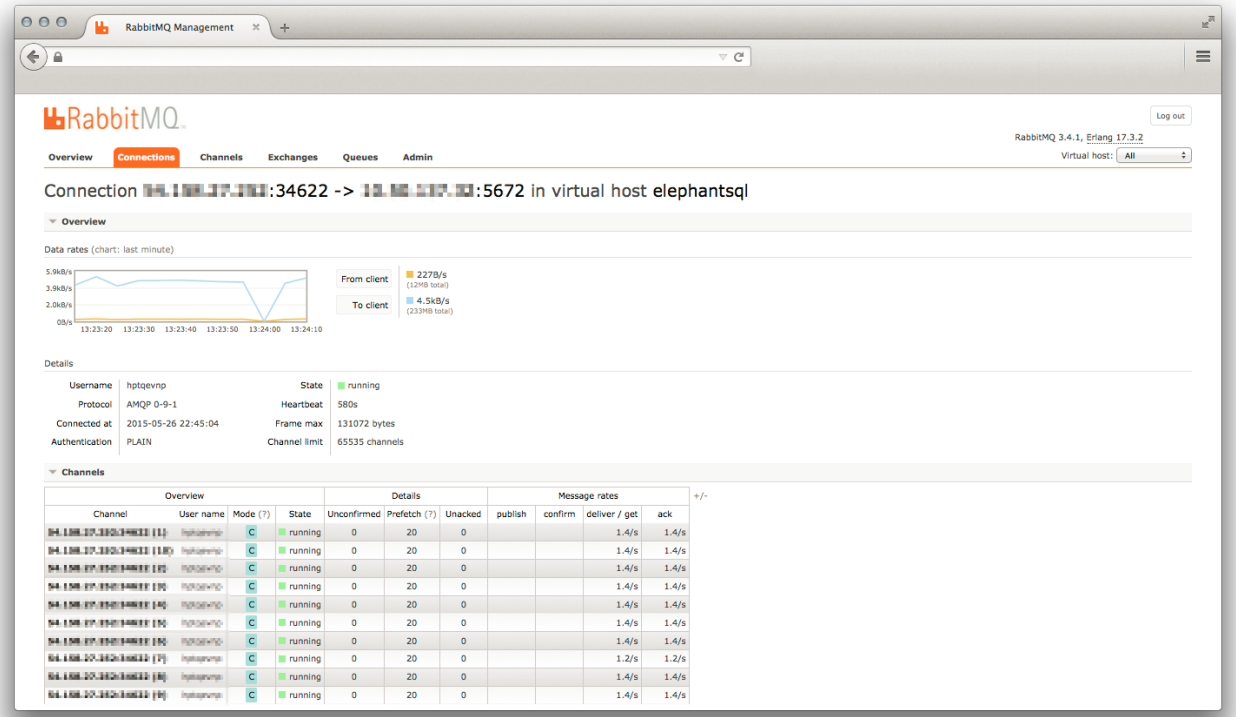
RabbitMQ connections and channels can be in different **states;** starting, tuning, opening, running, flow, blocking, blocked, closing, closed. If a connection enters flow-control this often means that the client is being rate-limited in some way; A good article to read when that is happening can be found here. (http://www.rabbitmq.com/blog/2014/04/14/finding-bottlenecks-with-rabbitmq-3-3/)

Connections

The connection tab shows the connections established to the RabbitMQ server. **vhost** shows in which vhost the connection operates, the **username** the user associated with the connection. **Channels** tell the number of channels using the connection. **SSL/TLS** indicate whether the connection is secured with SSL.



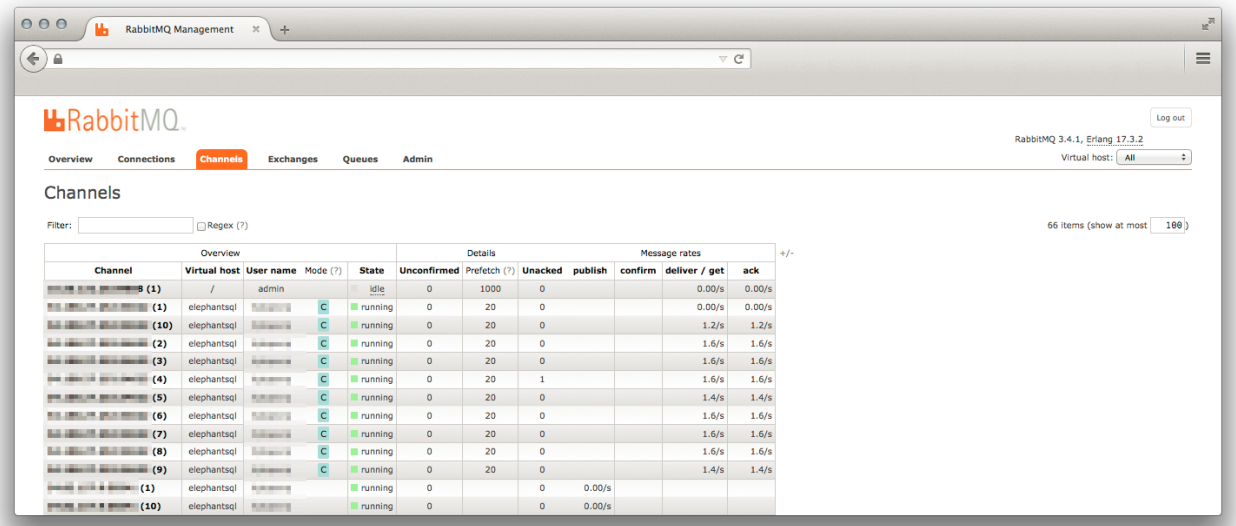
If you click on one of the connections, you get an overview of that specific connection. You can view channels in the connection and data rates. You can see client properties and you can close the connection.



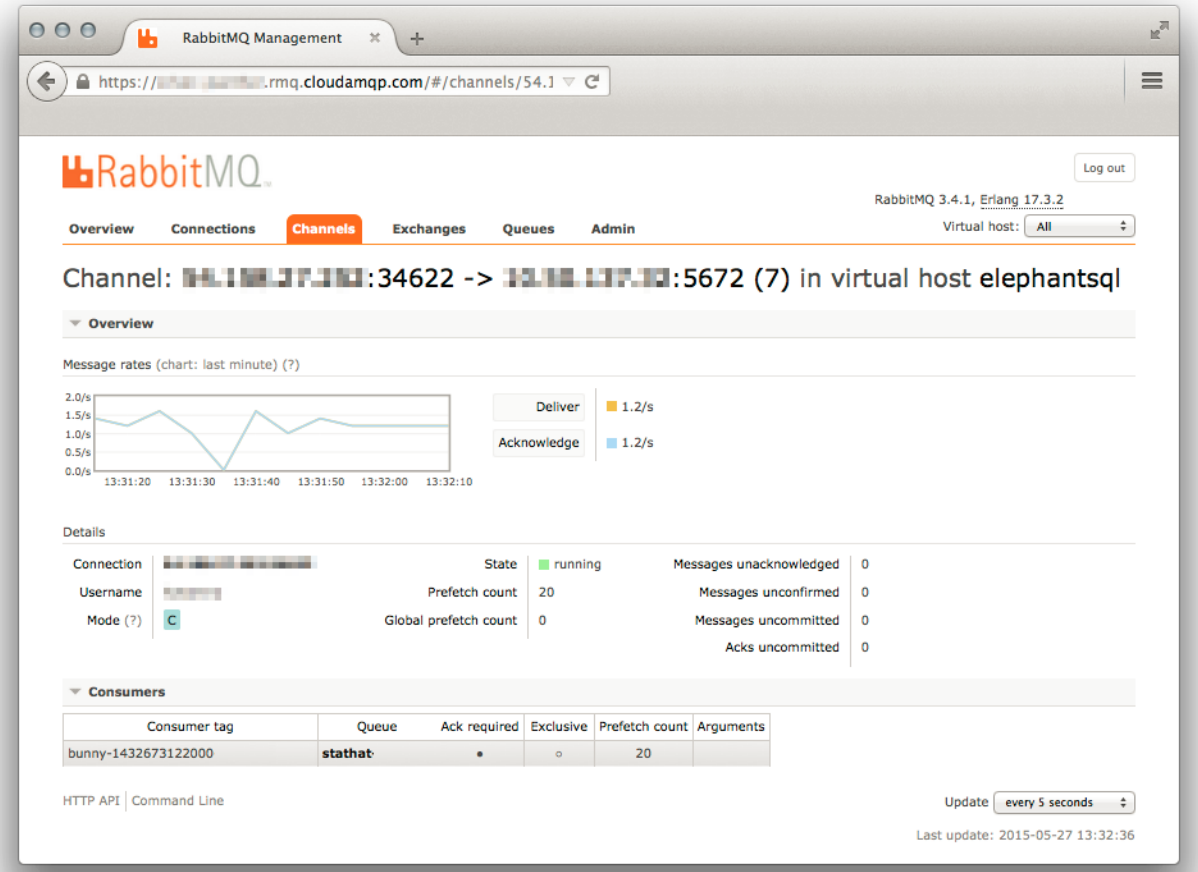
More information about the attributes associated with a connection can be found here (https://www.rabbitmq.com/man/rabbitmqctl.1.man.html#list_connections) in the manual page for rabbitmqctl, the command line tool for managing a RabbitMQ broker.

Channels

The channel tab show information about all current channels. The **vhost** shows in which vhost the channel operates, the **username** the user associated with the channel. The **mode** tells the channel guarantee mode. It can be in confirm or transactional mode. When a channel is in confirm mode, both the broker and the client count messages. The broker then confirms messages as it handles them. Confirm mode is activated once the `confirm.select` method is used on a channel.



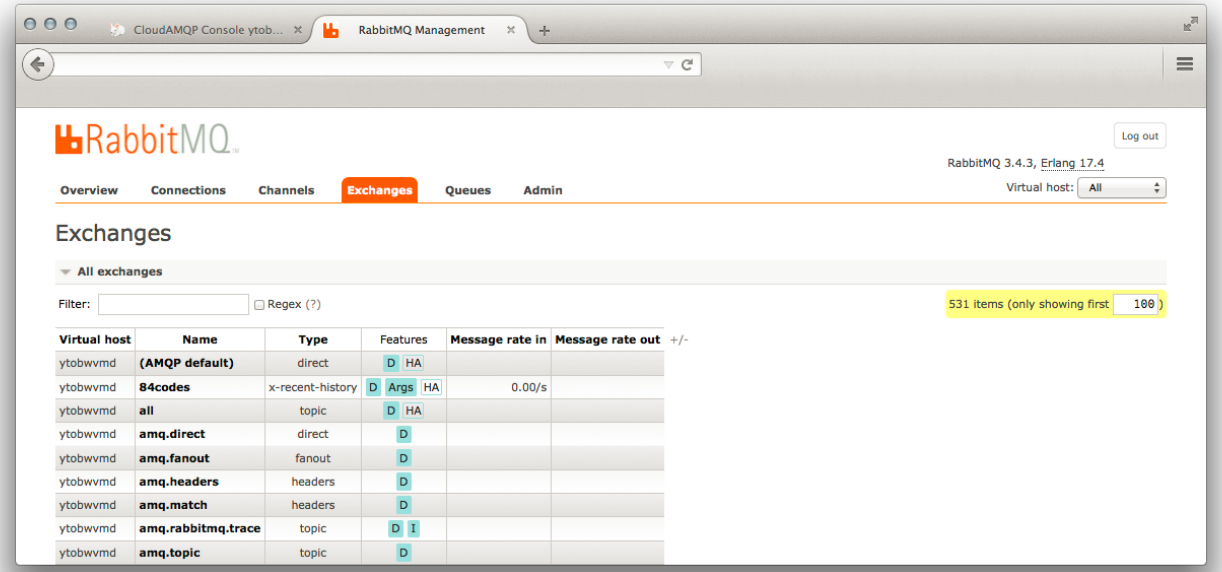
If you click on one of the channels, you get a detailed overview of that specific channel. From here you can see the message rate on the number of logical consumers retrieving messages via the channel.



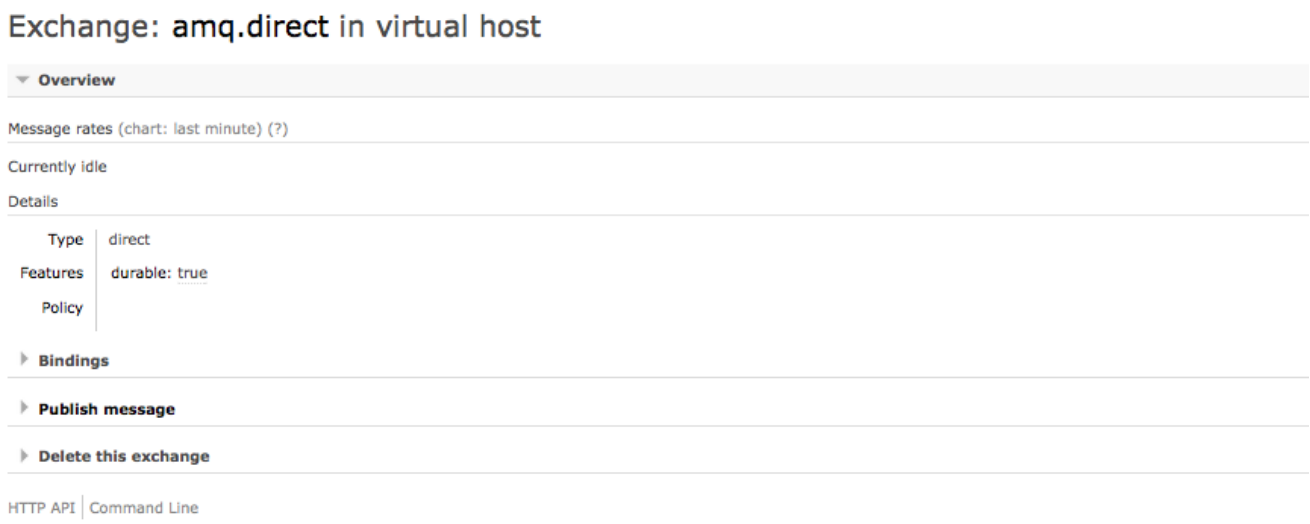
More information about the attributes associated with a channel can be found here (https://www.rabbitmq.com/man/rabbitmqctl.1.man.html#list_channels) in the manual page for rabbitmqctl, the command line tool for managing a RabbitMQ broker.

Exchanges

An exchange receives messages from producers and pushes them to queues. The exchange must know exactly what to do with a message it receives. All exchanges can be listed from the exchange tab. **Virtual host** shows the vhost for the exchange, **type** is the exchange type such as direct, topic, headers, fanout. **Features** show the parameters for the exchange (e.g. D stand for durable, and AD for auto-delete). Features and types can be specified when the exchange is created. In this list there are some amq.* exchanges and the default (unnamed) exchange. These are created by default.

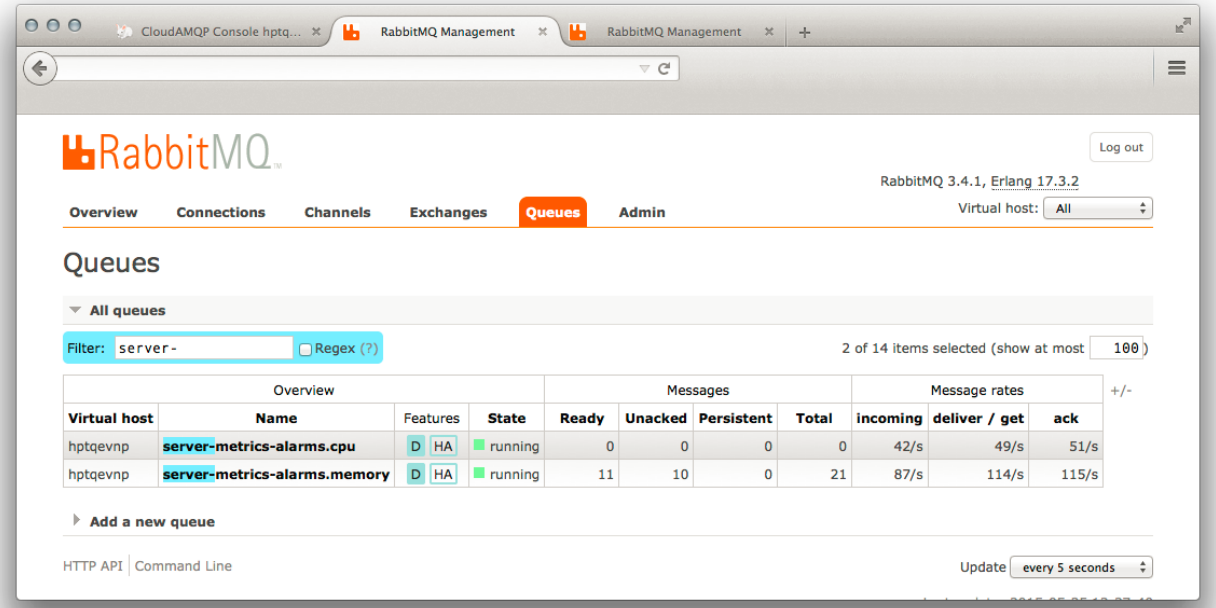


By clicking on the exchange name, a detailed page about the exchange are shown. You can see and add bindings to the exchange. You can also publish a message to the exchange or delete the exchange.



Queues

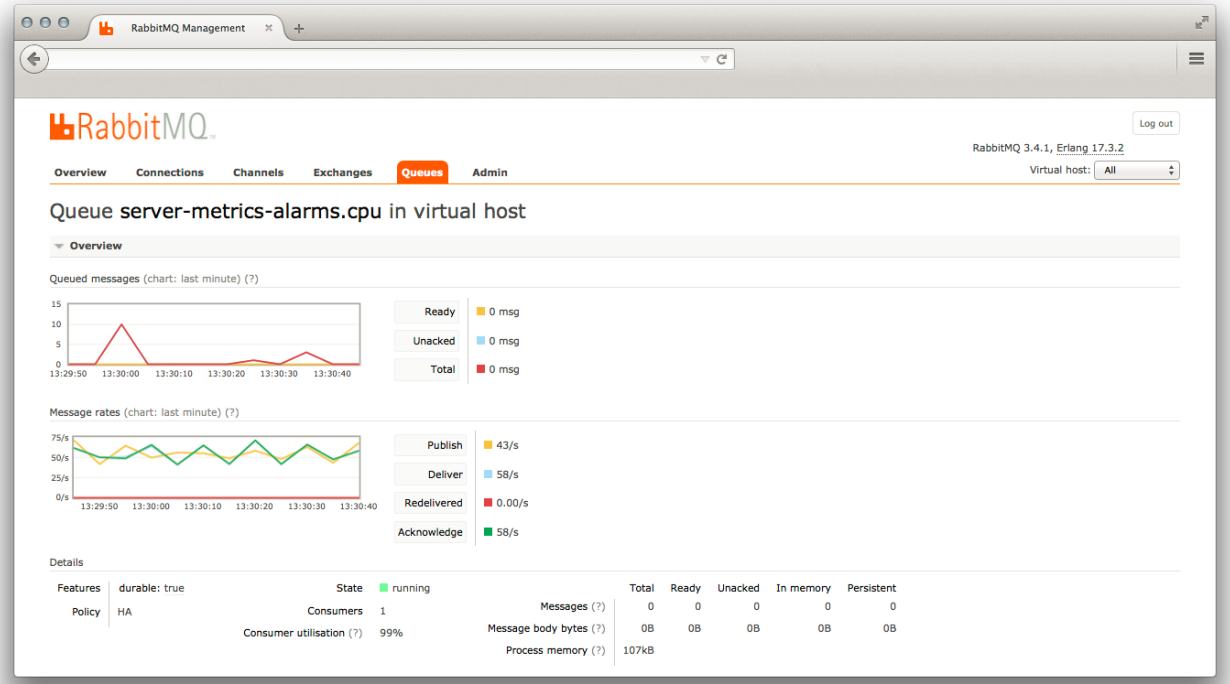
The queue tab show the queues for all or one selected vhost.



Queues have different parameters and arguments depending on how they were created. The *features* column show the parameters that belong to the queue. It could be features like *Durable queue* (which ensure that RabbitMQ will never lose the queue), *Message TTL* (which tells how long a message published to a queue can live before it is discarded), *Auto expire* (which tells how long a queue can be unused for before it is automatically deleted), *Max length* (which tells how many (ready) messages a queue can contain before it starts to drop them) and *Max length bytes (which tells the total body size for ready messages a queue can contain before it starts to drop them).*

You can also create a queue from this view.

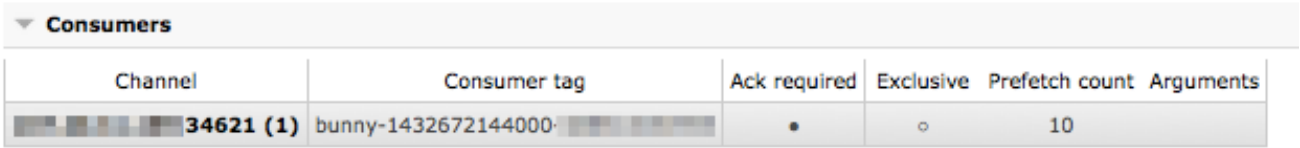
If you press on any chosen queue from the list of queues, all information about the queue are shown like in the pictures that follow below.



The first two charts include the same information as the overview, but it just shows the number of queued messages and the message rates for that specific queue.

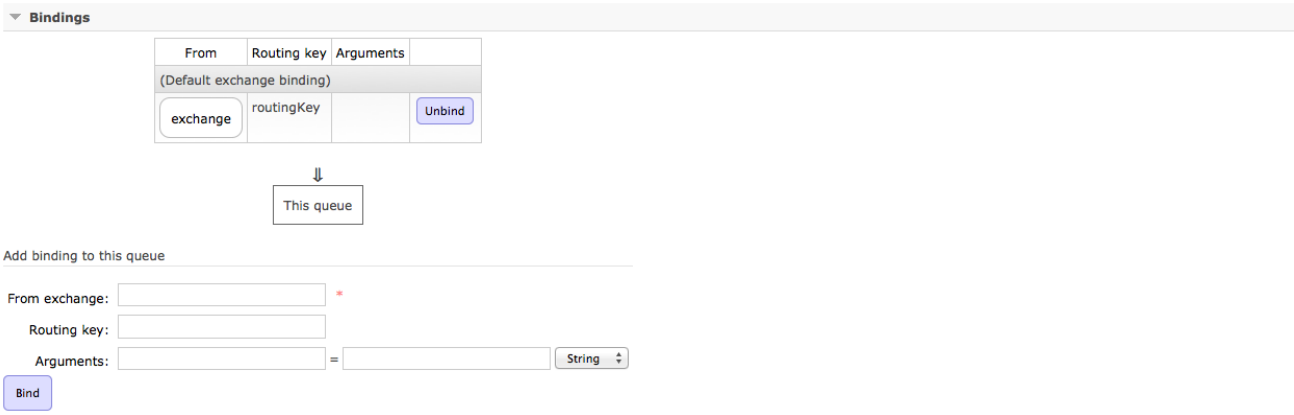
Consumers

Consumers show the consumers/channels that are connected to the queue.



Bindings

A binding can be created between an exchange and a queue. All active bindings to the queue are shown under bindings. You can also create a new binding to a queue from here or unbind a queue from an exchange.

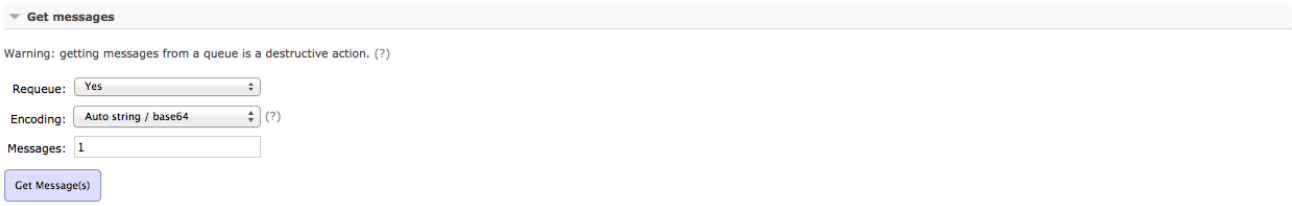


Publish message

It is possible to manually publish a message to the queue from "publish message". The message will be published to the default exchange with the queue name as given routing key - meaning that the message will be sent to the queue. It is also possible to publish a message to an exchange from the exchange view.

Get message

It is possible to manually inspect the message in the queue. "Get message" get the message to you and if you mark it as "requeue", RabbitMQ puts it back to the queue in the same order.

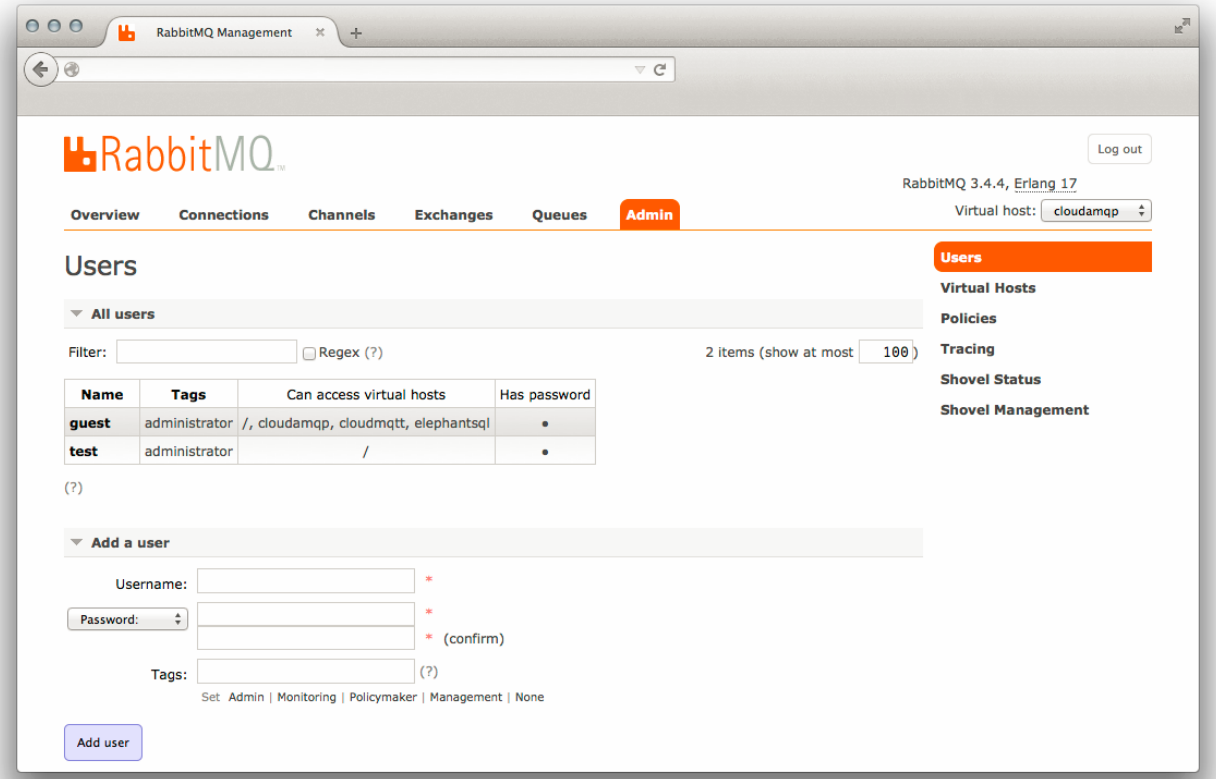


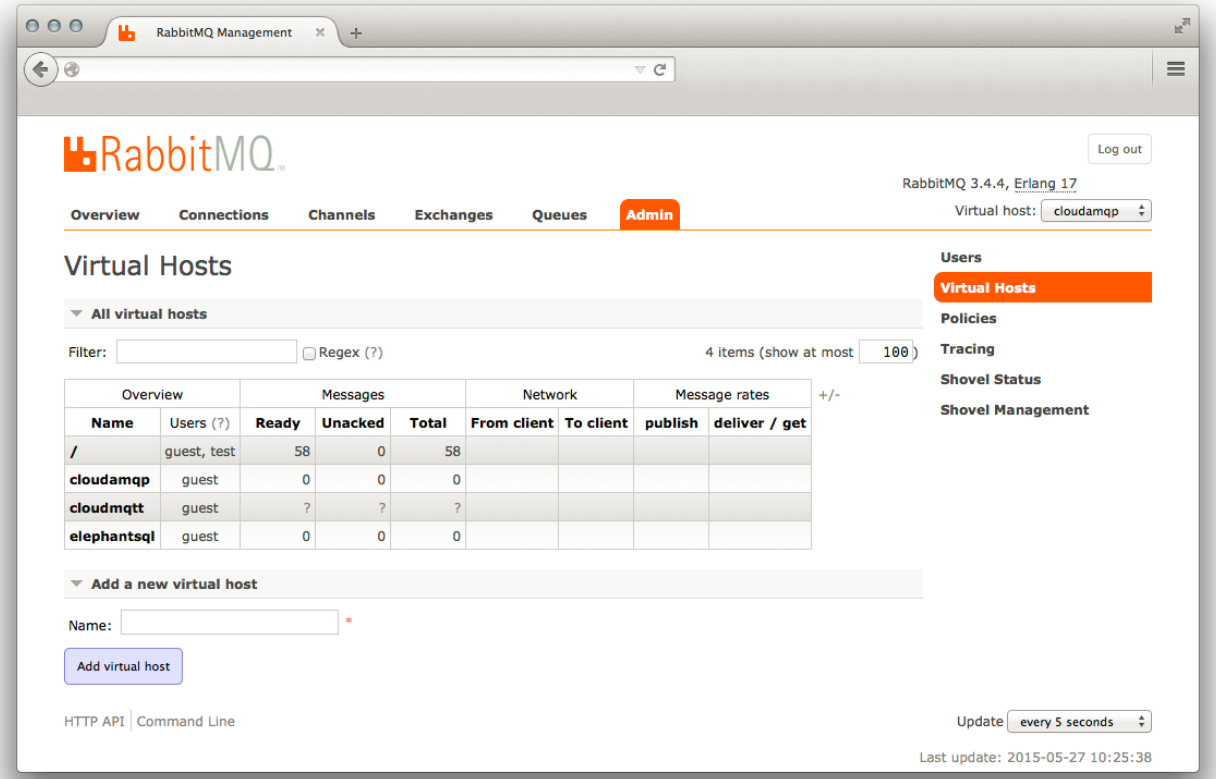
Delete or Purge queue

A queue can be deleted by the delete button, and you can empty the queue by pressing purge.

Admin

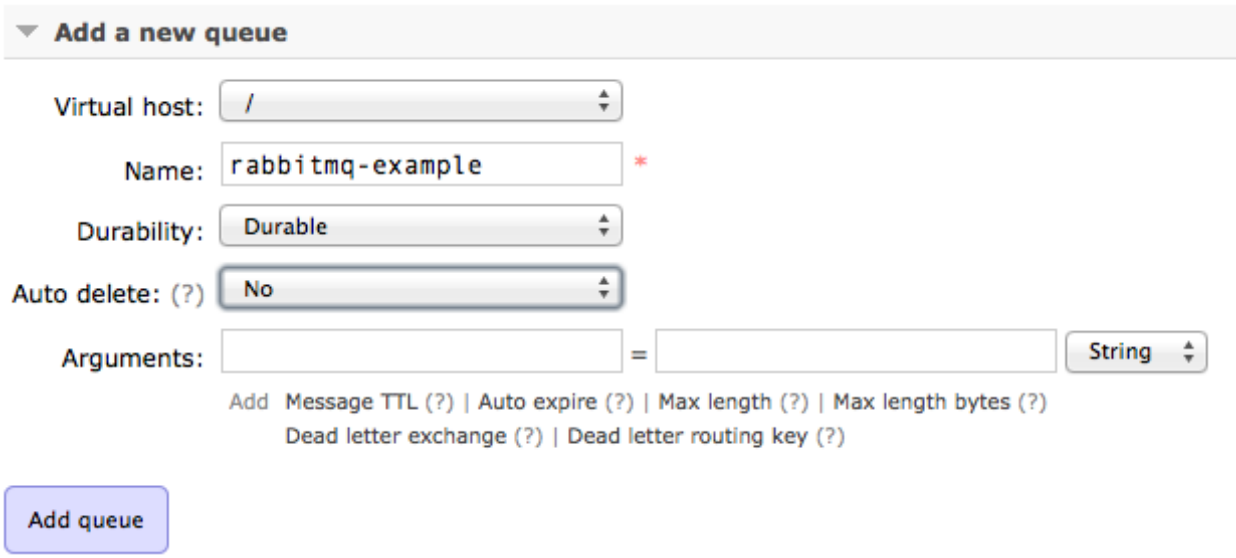
From the Admin view it is possible to add users and change user permissions. You can set up vhosts, policies, federation and shovels. Information about shovels can be found here: https://www.rabbitmq.com/shovel.html (https://www.rabbitmq.com/shovel.html) and http://www.cloudamqp.com/docs/shovel.html. (/docs/shovel.html) Information about federation can be found here: https://www.cloudamqp.com/blog/2015-03-24-rabbitmq-federation.html (https://www.cloudamqp.com/blog/2015-03-24-rabbitmq-federation.html)


RabbitMQ Management
RabbitMQ
Log out
RabbitMQ 3.4.4, Erlang 17
Virtual host: cloudamqp
Overview
Connections
Channels
Exchanges
Queues
Admin
Users
All users
Filter:
Regex (?)
2 items (show at most 100)
Name
Tags
Can access virtual hosts
Has password
guest
administrator
/, cloudamqp, cloudmqtt, elephantsql
test
administrator
/
(?)
Add a user
Username:
Password:
(confirm)
Tags:
(?)
Set Admin | Monitoring | Policymaker | Management | None
Add user
Users
Virtual Hosts
Policies
Tracing
Shovel Status
Shovel Management

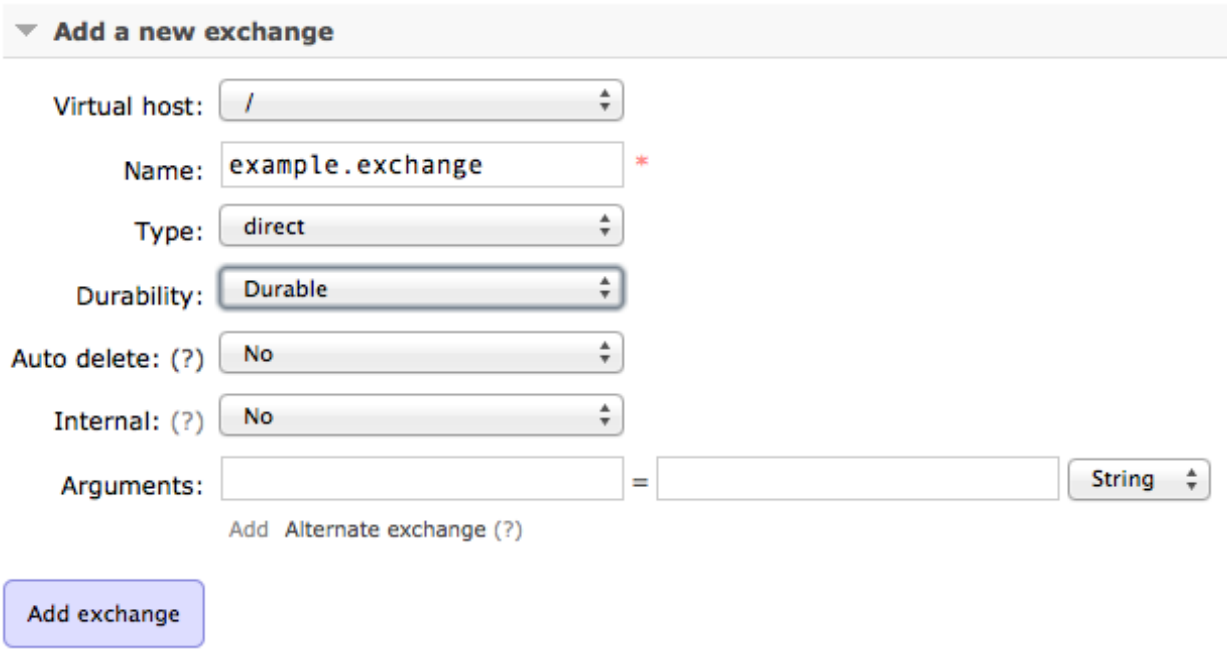

RabbitMQ Management
RabbitMQ
Log out
RabbitMQ 3.4.4, Erlang 17
Virtual host: cloudamqp
Overview
Connections
Channels
Exchanges
Queues
Admin
Virtual Hosts
All virtual hosts
Filter:
Regex (?)
4 items (show at most 100)
Overview
Messages
Network
Message rates
+/-
Name
Users (?)
Ready
Unacked
Total
From client
To client
publish
deliver / get
/
guest, test
58
0
58
cloudamqp
guest
0
0
0
cloudmqtt
guest
?
?
?
elephantsql
guest
0
0
0
Add a new virtual host
Name:
Add virtual host
HTTP API
Command Line
Update
every 5 seconds
Last update: 2015-05-27 10:25:38
Users
Virtual Hosts
Policies
Tracing
Shovel Status
Shovel Management

Example

This example show how you can create a queue "example-queue" and an exchange called example.exchange.



Queue view: Add queue



Exchange view: Add exchange

The exchange and the queue are connected by a binding called "pdfprocess". Messages published to the exchange with the routing key "pdfprocess" will end up in the queue.

Add binding from this exchange

To queue : rabbitmq-example *

Routing key: pdfprocess

Arguments: = String

Bind

Press on the exchange or on the queue go to "Add binding from this exchange" or "Add binding to this queue"

Publish message

Routing key: pdfprocess

Delivery mode: 1 – Non-persistent

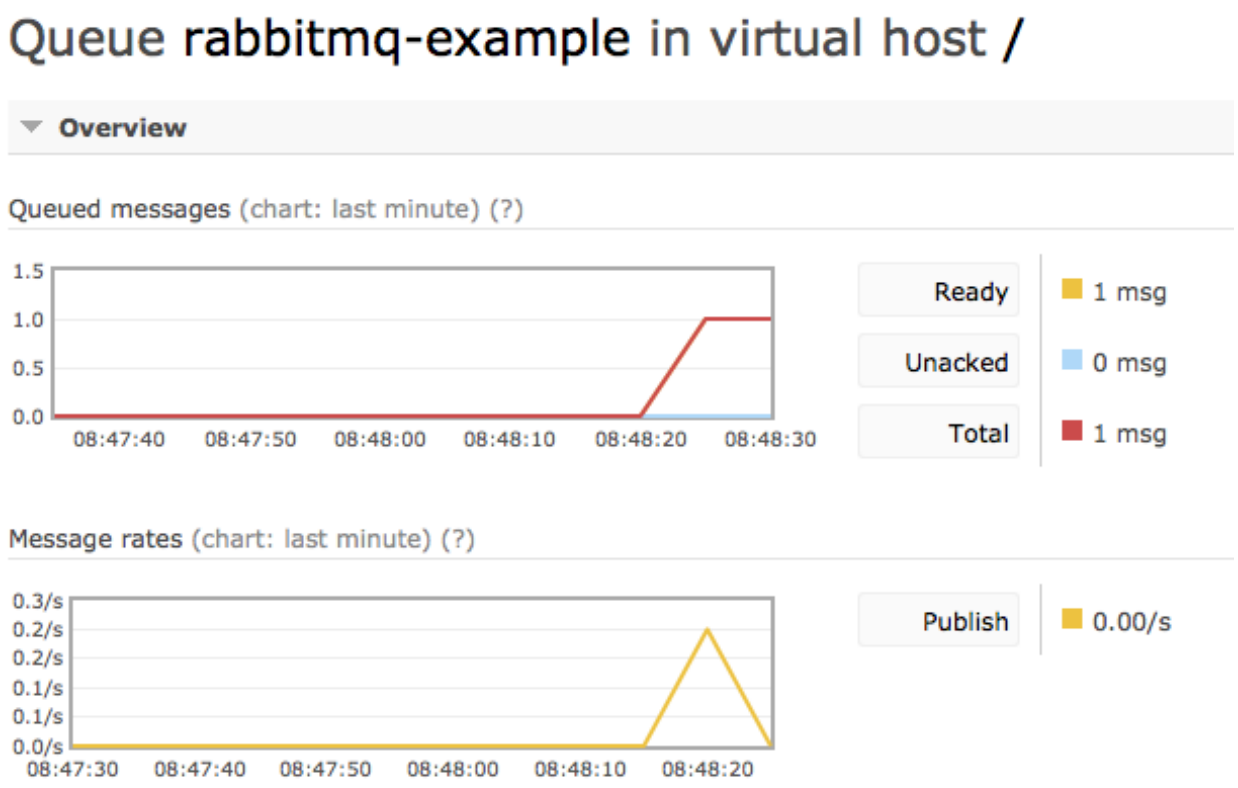
Headers: (?) = String

Properties: (?) =

Payload: Hello CloudAMQP

Publish message

Publish a message to the exchange with the routing key "pdfprocess"



Queue overview for example-queue when a message is published.

A lot of things can be viewed and handled from the management interface and it will give you a good overview of your system. By looking into the management interface, you will get a good understanding about RabbitMQ and how everything is related.

Please email us at contact@cloudamqp.com (mailto:contact@cloudamqp.com) if you have any suggestions or feedback.

‹ Part 2: Get started with RabbitMQ - Sample code (/blog/part2-rabbitmq-for-beginners_example-and-sample-code.html)

Part 4: Exchanges, routing keys and bindings ›