# RabbitMQ镜像搭建

在2017/11/24 12:40上被[亮 刘](http://er.msqltec.com:8080/xwiki/bin/view/XWiki/liuliang)修改

# 网易地址：

<https://c.163.com/hub#/m/repository/?repoId=3101>

# 镜像地址：

docker pull hub.c.163.com/library/rabbitmq:latest

公司环境执行的命令

docker run -d --hostname dev-rabbit --name dev-rabbit-mana -e RABBITMQ\_ERLANG\_COOKIE='devcookie' -e RABBITMQ\_DEFAULT\_USER=devmq -e RABBITMQ\_DEFAULT\_PASS=devmq -p 7001:15672 -p 5672:5672  rabbitmq:3.6.11-management

# Supported tags and respective Dockerfile links

* [3.6.11, 3.6, 3, latest (3.6/debian/Dockerfile)](https://github.com/docker-library/rabbitmq/blob/7d76799247764423a69438f72961ef0581788e07/3.6/debian/Dockerfile)
* [3.6.11-management, 3.6-management, 3-management, management (3.6/debian/management/Dockerfile)](https://github.com/docker-library/rabbitmq/blob/79277042564875d55e4b05a60c65b6eb46651a94/3.6/debian/management/Dockerfile)
* [3.6.11-alpine, 3.6-alpine, 3-alpine, alpine (3.6/alpine/Dockerfile)](https://github.com/docker-library/rabbitmq/blob/8047bd34685f27bea47dcd646d3b6332b0ec22d3/3.6/alpine/Dockerfile)
* [3.6.11-management-alpine, 3.6-management-alpine, 3-management-alpine, management-alpine (3.6/alpine/management/Dockerfile)](https://github.com/docker-library/rabbitmq/blob/79277042564875d55e4b05a60c65b6eb46651a94/3.6/alpine/management/Dockerfile)

# How to use this image

## Running the daemon

One of the important things to note about RabbitMQ is that it stores data based on what it calls the \"Node Name\", which defaults to the hostname. What this means for usage in Docker is that we should specify -h/--hostname explicitly for each daemon so that we don't get a random hostname and can keep track of our data:

### $ docker run -d --hostname my-rabbit --name some-rabbit rabbitmq:3

If you give that a minute, then do docker logs some-rabbit, you'll see in the output a block similar to:

=INFO REPORT==== 6-Jul-2015::20:47:02 ===

node : rabbit@my-rabbit

home dir : /var/lib/rabbitmq

config file(s) : /etc/rabbitmq/rabbitmq.config

cookie hash : UoNOcDhfxW9uoZ92wh6BjA==

log : tty

sasl log : tty

database dir : /var/lib/rabbitmq/mnesia/rabbit@my-rabbit

Note the database dir there, especially that it has my \"Node Name\" appended to the end for the file storage. This image makes all of /var/lib/rabbitmq a volume by default.

### Memory Limits

RabbitMQ contains functionality which explicitly tracks and manages memory usage, and thus needs to be made aware of cgroup-imposed limits.

The upstream configuration setting for this is vm\_memory\_high\_watermark, and it is described under [\"Memory Alarms\"](https://www.rabbitmq.com/memory.html) in the documentation.

In this image, this value is set via RABBITMQ\_VM\_MEMORY\_HIGH\_WATERMARK. The value of this environment variable is interpreted as follows:

* 0.49 is treated as 49%, just like upstream ({ vm\_memory\_high\_watermark, 0.49 })
* 56% is treated as 56% (0.56; { vm\_memory\_high\_watermark, 0.56 })
* 1073741824 is treated as an absolute number of bytes ({ vm\_memory\_high\_watermark, { absolute, 1073741824 } })
* 1024MiB is treated as an absolute number of bytes with a unit ({ vm\_memory\_high\_watermark, { absolute, \"1024MiB\" } })

The main behavioral difference is in how percentages are handled. If the current container has a memory limit (--memory/-m), a percentage value will be calculated to an absolute byte value based on the memory limit, rather than being passed to RabbitMQ as-is. For example, a container run with --memory 2048m (and the implied upstream-default RABBITMQ\_VM\_MEMORY\_HIGH\_WATERMARK of 40%) will set the effective limit to 819MB (which is 40% of 2048MB).

### Erlang Cookie

See the [RabbitMQ \"Clustering Guide\"](https://www.rabbitmq.com/clustering.html#erlang-cookie) for more information about cookies and why they're necessary.

For setting a consistent cookie (especially useful for clustering but also for remote/cross-container administration via rabbitmqctl), use RABBITMQ\_ERLANG\_COOKIE:

### $ docker run -d --hostname my-rabbit --name some-rabbit -e RABBITMQ\_ERLANG\_COOKIE='secret cookie here' rabbitmq:3

This can then be used from a separate instance to connect:

### $ docker run -it --rm --link some-rabbit:my-rabbit -e RABBITMQ\_ERLANG\_COOKIE='secret cookie here' rabbitmq:3 bash root@f2a2d3d27c75:/# rabbitmqctl -n rabbit@my-rabbit list\_users Listing users ... guest   [administrator]

Alternatively, one can also use RABBITMQ\_NODENAME to make repeated rabbitmqctl invocations simpler:

### $ docker run -it --rm --link some-rabbit:my-rabbit -e RABBITMQ\_ERLANG\_COOKIE='secret cookie here' -e RABBITMQ\_NODENAME=rabbit@my-rabbit rabbitmq:3 bash root@f2a2d3d27c75:/# rabbitmqctl list\_users Listing users ... guest   [administrator]

### Management Plugin

There is a second set of tags provided with the [management plugin](https://www.rabbitmq.com/management.html) installed and enabled by default, which is available on the standard management port of 15672, with the default username and password of guest / guest:

### $ docker run -d --hostname my-rabbit --name some-rabbit rabbitmq:3-management

You can access it by visiting http://container-ip:15672 in a browser or, if you need access outside the host, on port 8080:

### $ docker run -d --hostname my-rabbit --name some-rabbit -p 8080:15672 rabbitmq:3-management

You can then go to http://localhost:8080 or http://host-ip:8080 in a browser.

## Setting default user and password

If you wish to change the default username and password of guest / guest, you can do so with the RABBITMQ\_DEFAULT\_USER and RABBITMQ\_DEFAULT\_PASS environmental variables:

### $ docker run -d --hostname my-rabbit --name some-rabbit -e RABBITMQ\_DEFAULT\_USER=user -e RABBITMQ\_DEFAULT\_PASS=password rabbitmq:3-management

You can then go to http://localhost:8080 or http://host-ip:8080 in a browser and use user/password to gain access to the management console

## Setting default vhost

If you wish to change the default vhost, you can do so with the RABBITMQ\_DEFAULT\_VHOST environmental variables:

### $ docker run -d --hostname my-rabbit --name some-rabbit -e RABBITMQ\_DEFAULT\_VHOST=my\_vhost rabbitmq:3-management

## Enabling HiPE

**Warning:** if you're using the Alpine variant, there is currently [an outstanding bug (Alpine Linux bug #5700) with the erlang-hipe package](https://bugs.alpinelinux.org/issues/5700) which prevents HiPE from working in Alpine Linux. See [docker-library/rabbitmq#151](https://github.com/docker-library/rabbitmq/issues/151) for more discussion.

See the [RabbitMQ \"Configuration\"](http://www.rabbitmq.com/configure.html#config-items) for more information about various configuration options.

For enabling the HiPE compiler on startup use RABBITMQ\_HIPE\_COMPILE set to 1. Accroding to the official documentation:

Set to true to precompile parts of RabbitMQ with HiPE, a just-in-time compiler for Erlang. This will increase server throughput at the cost of increased startup time. You might see 20-50% better performance at the cost of a few minutes delay at startup.

It is therefore important to take that startup delay into consideration when configuring health checks, automated clustering etc.

## Connecting to the daemon

$ docker run --name some-app --link some-rabbit:rabbit -d application-that-uses-rabbitmq

# Image Variants

The rabbitmq images come in many flavors, each designed for a specific use case.

## rabbitmq:<version>

This is the defacto image. If you are unsure about what your needs are, you probably want to use this one. It is designed to be used both as a throw away container (mount your source code and start the container to start your app), as well as the base to build other images off of.

## rabbitmq:alpine

This image is based on the popular [Alpine Linux project](http://alpinelinux.org/), available in [the alpine official image](https://hub.docker.com/_/alpine). Alpine Linux is much smaller than most distribution base images (~5MB), and thus leads to much slimmer images in general.

This variant is highly recommended when final image size being as small as possible is desired. The main caveat to note is that it does use [musl libc](http://www.musl-libc.org/) instead of [glibc and friends](http://www.etalabs.net/compare_libcs.html), so certain software might run into issues depending on the depth of their libc requirements. However, most software doesn't have an issue with this, so this variant is usually a very safe choice. See [this Hacker News comment thread](https://news.ycombinator.com/item?id=10782897) for more discussion of the issues that might arise and some pro/con comparisons of using Alpine-based images.

To minimize image size, it's uncommon for additional related tools (such as git or bash) to be included in Alpine-based images. Using this image as a base, add the things you need in your own Dockerfile (see the [alpine image description](https://hub.docker.com/_/alpine/) for examples of how to install packages if you are unfamiliar).