OFFICIAL REPOSITORY

rabbitmq (/r/_/rabbitmq/) ☆

Last pushed: a month ago

Repo Info (/_/rabbitmq/)

Short Description

RabbitMQ is an open source multi-protocol messaging broker.

Full Description

Supported tags and respective `Dockerfile` links

- `3.7.8` , `3.7` , `3` , `latest` (*3.7/debian/Dockerfile*) (https://github.com/docker-library/rabbitmq/blob/1a8fd1c6ee027baafb7144c24ad3c995ba5e0d24/3.7/debian/Dockerfile)
- `3.7.8-management` , `3.7-management` , `3-management` , `management` (*3.7/debian/management/Dockerfile*) (https://github.com/docker-library/rabbitmq/blob/4b2b11c59ee65c2a09616b163d4572559a86bb7b/3.7/debian/management/Dockerfile)
- `3.7.8-alpine` , `3.7-alpine` , `3-alpine` , `alpine` (*3.7/alpine/Dockerfile*) (https://github.com/docker-library/rabbitmq/blob/1a8fd1c6ee027baafb7144c24ad3c995ba5e0d24/3.7/alpine/Dockerfile)
- `3.7.8-management-alpine` , `3.7-management-alpine` , `3-management-alpine` , `management-alpine` (*3.7/alpine/management/Dockerfile*) (https://github.com/docker-library/rabbitmq/blob/4b2b11c59ee65c2a09616b163d4572559a86bb7b/3.7/alpine/management/Dockerfile)
- `3.6.16` , `3.6` (*3.6/debian/Dockerfile*) (https://github.com/docker-library/rabbitmq/blob/1a8fd1c6ee027baafb7144c24ad3c995ba5e0d24/3.6/debian/Dockerfile)
- `3.6.16-management` , `3.6-management` (*3.6/debian/management/Dockerfile*) (https://github.com/docker-library/rabbitmq/blob/b9eda3e4665c24db70a9a290fddf33bc5c567b10/3.6/debian/management/Dockerfile)
- `3.6.16-alpine` , `3.6-alpine` (*3.6/alpine/Dockerfile*) (https://github.com/docker-library/rabbitmq/blob/1a8fd1c6ee027baafb7144c24ad3c995ba5e0d24/3.6/alpine/Dockerfile)
- `3.6.16-management-alpine` , `3.6-management-alpine` (*3.6/alpine/management/Dockerfile*) (https://github.com/docker-library/rabbitmq/blob/b9eda3e4665c24db70a9a290fddf33bc5c567b10/3.6/alpine/management/Dockerfile)

Quick reference

- **Where to get help**:
  the Docker Community Forums (https://forums.docker.com/), the Docker Community Slack (https://blog.docker.com/2016/11/introducing-docker-community-directory-docker-community-

slack/), or Stack Overflow (https://stackoverflow.com/search?tab=newest&q=docker)

- **Where to file issues**:
  https://github.com/docker-library/rabbitmq/issues (https://github.com/docker-library/rabbitmq/issues)

- **Maintained by**:
  the Docker Community (https://github.com/docker-library/rabbitmq)

- **Supported architectures**: (more info (https://github.com/docker-library/official-images#architectures-other-than-amd64))
  `amd64` (https://hub.docker.com/r/amd64/rabbitmq/), `arm32v5` (https://hub.docker.com/r/arm32v5/rabbitmq/), `arm32v6` (https://hub.docker.com/r/arm32v6/rabbitmq/), `arm32v7` (https://hub.docker.com/r/arm32v7/rabbitmq/), `arm64v8` (https://hub.docker.com/r/arm64v8/rabbitmq/), `i386` (https://hub.docker.com/r/i386/rabbitmq/), `ppc64le` (https://hub.docker.com/r/ppc64le/rabbitmq/), `s390x` (https://hub.docker.com/r/s390x/rabbitmq/)

- **Published image artifact details**:
  repo-info repo's `repos/rabbitmq/` directory (https://github.com/docker-library/repo-info/blob/master/repos/rabbitmq) (history (https://github.com/docker-library/repo-info/commits/master/repos/rabbitmq))
  (image metadata, transfer size, etc)

- **Image updates**:
  official-images PRs with label `library/rabbitmq` (https://github.com/docker-library/official-images/pulls?q=label%3Alibrary%2Frabbitmq)
  official-images repo's `library/rabbitmq` file (https://github.com/docker-library/official-images/blob/master/library/rabbitmq) (history (https://github.com/docker-library/official-images/commits/master/library/rabbitmq))

- **Source of this description**:
  docs repo's `rabbitmq/` directory (https://github.com/docker-library/docs/tree/master/rabbitmq) (history (https://github.com/docker-library/docs/commits/master/rabbitmq))

- **Supported Docker versions**:
  the latest release (https://github.com/docker/docker-ce/releases/latest) (down to 1.6 on a best-effort basis)

What is RabbitMQ?

RabbitMQ is open source message broker software (sometimes called message-oriented middleware) that implements the Advanced Message Queuing Protocol (AMQP). The RabbitMQ server is written in the Erlang programming language and is built on the Open Telecom Platform framework for clustering and failover. Client libraries to interface with the broker are available for all major programming languages.

> wikipedia.org/wiki/RabbitMQ (https://en.wikipedia.org/wiki/RabbitMQ)

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
```

This will start a RabbitMQ container listening on the default port of 5672. 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_
```

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

```
$ docker run -it --rm --link some-rabbit:my-rabbit -e RABBITMQ_ERLANG_COOKI
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_COOKI
root@f2a2d3d27c75:/# rabbitmqctl list_users
Listing users ...
guest   [administrator]
```

If you wish to provide the cookie via a file (such as with Docker Secrets (https://docs.docker.com/engine/swarm/secrets/)), it needs to be mounted at `/var/lib/rabbitmq/.erlang.cookie`:

```
docker service create ... --secret source=my-erlang-cookie,target=/var/lib/
```

(Note that it will likely also be necessary to specify `uid=XXX,gid=XXX,mode=0600` in order for Erlang in the container to be able to read the cookie file properly. See Docker's `--secret` documentation for more details (https://docs.docker.com/engine/reference/commandline/service_create/#create-a-service-with-secrets).)

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-manageme
```

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 rabbi
```

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

Environment Variables

A small selection of the possible environment variables are defined in the Dockerfile to be passed through the docker engine (listed below). For a list of environment variables supported by RabbitMQ itself, see: https://www.rabbitmq.com/configure.html (https://www.rabbitmq.com/configure.html)

For SSL configuration without the management plugin:

```
 RABBITMQ_SSL_CACERTFILE
RABBITMQ_SSL_CERTFILE
RABBITMQ_SSL_DEPTH
RABBITMQ_SSL_FAIL_IF_NO_PEER_CERT
RABBITMQ_SSL_KEYFILE
RABBITMQ_SSL_VERIFY
```

For SSL configuration using the management plugin:

```
 RABBITMQ_MANAGEMENT_SSL_CACERTFILE
RABBITMQ_MANAGEMENT_SSL_CERTFILE
RABBITMQ_MANAGEMENT_SSL_DEPTH
RABBITMQ_MANAGEMENT_SSL_FAIL_IF_NO_PEER_CERT
RABBITMQ_MANAGEMENT_SSL_KEYFILE
RABBITMQ_MANAGEMENT_SSL_VERIFY
```

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
```

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

To source the username and password from files instead of environment variables, add a `_FILE` suffix to the environment variable names (for example, `RABBITMQ_DEFAULT_USER_FILE=/run/secrets/xxx` to use Docker Secrets (https://docs.docker.com/engine/swarm/secrets/)).

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
```

Enabling HiPE

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`. Acording 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.

Enabling Plugins

Creating a Dockerfile will have them enabled at runtime. To see the full list of plugins present on the image `rabbitmq-plugins list`

```
 FROM rabbitmq:3.7-management
RUN rabbitmq-plugins enable --offline rabbitmq_mqtt rabbitmq_federation_man
```

You can also mount a file at `/etc/rabbitmq/enabled_plugins` with contents as an erlang list of atoms ending with a period.

Example `enabled_plugins`

```
[rabbitmq_federation_management,rabbitmq_management,rabbitmq_mqtt,rabbitmq_
```

Additional Configuration

If additional configuration is required, it is recommended to supply an appropriate `/etc/rabbitmq/rabbitmq.conf` file (see the "Configuration File(s)" section of the RabbitMQ documentation for more details (https://www.rabbitmq.com/configure.html#configuration-files)), for example via bind-mount, Docker Configs (https://docs.docker.com/engine/swarm/configs/), or a short `Dockerfile` with a `COPY` instruction.

Alternatively, it is possible to use the `RABBITMQ_SERVER_ADDITIONAL_ERL_ARGS` environment variable, whose syntax is described in section 7.8 ("Configuring an Application") of the Erlang OTP Design Principles User's Guide (http://erlang.org/doc/design_principles/applications.html#id81887) (the appropriate value for `-ApplName` is `-rabbit`), this method requires a slightly different reproduction of its equivalent entry in `rabbitmq.conf`. For example, configuring `channel_max` (https://www.rabbitmq.com/configure.html#config-items) would look something like `-e RABBITMQ_SERVER_ADDITIONAL_ERL_ARGS="-rabbit channel_max 4007"`. Where the space between the variable `channel_max` and its value `4007` correctly becomes a comma when translated in the environment.

Additional configuration keys would be specified as a list. For example, configuring both `channel_max` (https://www.rabbitmq.com/configure.html#config-items) and `auth_backends` (https://www.rabbitmq.com/ldap.html#overview) would look something like `-e RABBITMQ_SERVER_ADDITIONAL_ERL_ARGS="-rabbit channel_max 4007 auth_backends` `[rabbit_auth_backend_ldap,rabbit_auth_backend_internal]"` . Note that some variables such as for `auth_backends` require their value(s) to be enclosed in brackets, and for multiple values explicitly including the comma as a delimiter.

Connecting to the daemon

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

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:<version>-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).

License

View license information (https://www.rabbitmq.com/mpl.html) for the software contained in this image.

As with all Docker images, these likely also contain other software which may be under other licenses (such as Bash, etc from the base distribution, along with any direct or indirect dependencies of the primary software being contained).

Some additional license information which was able to be auto-detected might be found in the `repo-info` repository's `rabbitmq/` directory (https://github.com/docker-library/repo-info/tree/master/repos/rabbitmq).

As for any pre-built image usage, it is the image user's responsibility to ensure that any use of this image complies with any relevant licenses for all software contained within.

Docker Pull Command

```
docker pull rabbitmq
```