Standalone Mac

Install the Server

This build of RabbitMQ includes an Erlang runtime. That means you don't need to install Erlang to run RabbitMQ. The build is known to work with OS X versions 10.6.4 and above. If you are curious about how this works you can read more **here**.

Download `rabbitmq-server-mac-standalone-3.5.0.tar.gz` from the link above.

Contained in the tarball is a directory named `rabbitmq_server-3.5.0`. You should extract this into somewhere appropriate for application binaries on your system. The `sbin` directory will be found in this directory.

Run RabbitMQ Server

Start the Server

Invoke the `sbin/rabbitmq-server` shell script. This displays a short banner message, concluding with the message "completed with *[n]* plugins.", indicating that the RabbitMQ broker has been started successfully.

You can also start the server in "detached" mode with `rabbitmq-server -detached`, in which case the server process runs in the background.

Configure the Server

You can **customise the RabbitMQ environment** by setting environment variables in`$RABBITMQ_HOME/etc/rabbitmq/rabbitmq-env.conf`. Server components may be configured, too, in the **RabbitMQ configuration file** located at `$RABBITMQ_HOME/etc/rabbitmq/rabbitmq.config`. Neither of these files exist after installation.

File Locations

The Generic Unix tarball is designed as far as possible to run without requiring configuration changes or special permissions. The directories and files used by default are all held under the installation directory`rabbitmq_server-3.5.0` which is in the $RABBITMQ_HOME variable in the scripts.

Should you wish to install RabbitMQ Server such that it uses the conventional system directories for configuration, database, log files, plugins etc, it is possible to do this.

Find the line:

```
SYS_PREFIX=${RABBITMQ_HOME}
```

in the `sbin/rabbitmq-defaults` script and change this line to:

```
SYS_PREFIX=
```

but do not modify any other line in this script.

Note: After this modification the default directory locations may require different permissions. In

particularRABBITMQ_MNESIA_BASE and RABBITMQ_LOG_BASE may need to be created (the server will attempt to create them at startup), and the RABBITMQ_ENABLED_PLUGINS_FILE will need to be writable (for`rabbitmq-plugins`). The configuration files will be looked for in `/etc/rabbitmq/`.

Default user access

The broker creates a user `guest` with password `guest`. Unconfigured clients will in general use these credentials. By default, these credentials can only be used when connecting to the broker as localhost so you will need to take action before connecting fromn any other machine.

See the documentation on **access control** for information on how to create more users, delete the `guest`user, or allow remote access to the `guest` user.

Managing the Broker

To stop the server or check its status, etc., you can invoke `sbin/rabbitmqctl` (as the user running`rabbitmq-server`). All `rabbitmqctl` commands will report the node absence if no broker is running.

Invoke `rabbitmqctl stop` to stop the server.

Invoke `rabbitmqctl status` to check whether it is running.

More **info on rabbitmqctl**.

Controlling System Limits on OS X

RabbitMQ installations running production workloads may need system limits and kernel parameters tuning in order to handle a decent number of concurrent connections and queues. The main setting that needs adjustment is the max number of open files, also known as `ulimit -n`. The default value on many operating systems is too low for a messaging broker (eg. 1024 on several Linux distributions). We recommend allowing for at least 65536 file descriptors for user `rabbitmq` in production environments. 4096 should be sufficient for most development workloads.

There are two limits in play: the maximum number of open files the OS kernel allows (`kern.maxfilesperproc`) and the per-user limit (`ulimit -n`). The former must be higher than the latter.

To adjust the per-user limit for RabbitMQ, there are several options:

Invoke `ulimit -S -n 4096` before starting RabbitMQ in foreground.

Edit the **rabbitmq-env.conf** to invoke `ulimit` before the service is started.

Configure max open files limit via `launchctl limit /etc/launchd.conf`.

Note that limits cannot be changed for running OS processes.

For more information about controlling `kern.maxfilesperproc` with `sysctl`, please refer to **sysctl(8)** page in Apple documentation.

Verifying the Limit

RabbitMQ management UI displays the number of file descriptors available for it to use on the Overview tab.

```
rabbitmqctl status
```

includes the same value.

The following command

```
launchctl limit
```

can be used to display effective limits for the current user.