RabbitMQ CLI Message broker admin

START HERE

`rabbitmqctl` **`status`**
Is the node healthy?

`rabbitmqctl` **`add_user`** *`<user> <password>`*
Create a user

`rabbitmqctl` **`cluster_status`**
Who is in the cluster?

`rabbitmq-plugins enable rabbitmq_management`
Turn on the web UI

`rabbitmqctl` **`list_queues name messages consumers`**
Queue depth at a glance

`rabbitmqctl` **`list_exchanges`**
What exchanges exist?

`rabbitmqctl` **`list_bindings`**
How is traffic routed?

`rabbitmqctl` **`list_connections`**
Who is connected?

Publisher sends messages → `routing key` → **Exchange** routes by type → `binding` → **Queue** holds messages → `delivery + ack` → **Consumer** processes them

CONCEPTS

Run rabbitmqctl as the same OS user as the broker: it authenticates with the Erlang cookie, and a wrong cookie means a refused connection.

Everything lives in a virtual host (default /). Users, permissions, queues and policies are all scoped per vhost; pass -p <vhost> to look at another one.

NODE AND STATUS

`rabbitmqctl` **`status`**
Node health, versions, memory, listeners

`rabbitmqctl` **`ping`**
Check the node is reachable

`rabbitmqctl` **`await_startup`**
Wait until the node has booted
Useful in scripts right after starting the broker.

`rabbitmqctl` **`environment`**
Show effective application environment

`rabbitmqctl` **`-n`** *`<node> <command>`*
Target a specific node
Default is rabbit@<short hostname>; add -l (--longnames) for FQDN node names.

`rabbitmqctl` **`-q`** *`<command>`*
Quieter output

NODE AND STATUS (cont.)

`rabbitmqctl` **`-s`** *`<command>`*
Silent: no headers, easy to parse

`rabbitmqctl` **`stop_app`**
Stop RabbitMQ, keep the Erlang VM

`rabbitmqctl` **`start_app`**
Start RabbitMQ again

QUEUES

`rabbitmqctl` **`list_queues`**
List queues with name and depth

`rabbitmqctl` **`list_queues -p`** *`<vhost>`*
List queues in one vhost

`rabbitmqctl` **`list_queues name messages_ready messages_unacknowledged`**
Ready vs unacked counts

`rabbitmqctl` **`list_queues name type state`**
Queue type and state

`rabbitmqctl` **`purge_queue`** *`<queue>`*
Drop all ready messages

`rabbitmqctl` **`delete_queue`** *`<queue>`*
Delete a queue

`rabbitmqctl` **`delete_queue`** *`<queue>`* **`--if-empty`**
Delete only when empty

QUEUES (cont.)

`rabbitmqctl` **`delete_queue`** *`<queue>`* **`--if-unused`**
Delete only when no consumers

`rabbitmqctl` **`list_consumers`**
Consumers per queue, with ack mode

EXCHANGES AND BINDINGS

`rabbitmqctl` **`list_exchanges`**
List exchanges with name and type

`rabbitmqctl` **`list_exchanges name type durable`**
Choose the columns

`rabbitmqctl` **`list_bindings`**
List all bindings

`rabbitmqctl` **`list_bindings source_name destination_name routing_key`**
Source, target and key only

CONNECTIONS AND CHANNELS

`rabbitmqctl` **`list_connections`**
List client connections

`rabbitmqctl` **`list_connections name user vhost state`**
Who, where and in what state

CONNECTIONS AND CHANNELS (cont.)

`rabbitmqctl` **`list_channels`**
List channels

`rabbitmqctl` **`list_channels connection name consumer_count messages_unacknowledged`**
Find stuck unacked messages

USERS, VHOSTS, PERMISSIONS

`rabbitmqctl` **`list_users`**
List users and their tags

`rabbitmqctl` **`add_user`** *`<user> <password>`*
Create a user

`rabbitmqctl` **`change_password`** *`<user> <new-password>`*
Change a password

`rabbitmqctl` **`delete_user`** *`<user>`*
Remove a user

`rabbitmqctl` **`set_user_tags`** *`<user>`* **`administrator`**
Grant management UI admin

`rabbitmqctl` **`add_vhost`** *`<vhost>`*
Create a virtual host

`rabbitmqctl` **`delete_vhost`** *`<vhost>`*
Delete a vhost and its contents

USERS, VHOSTS, PERMISSIONS (cont.)

`rabbitmqctl` **`set_permissions -p`** *`<vhost> <user>`* **`".*" ".*" ".*"`**
Grant configure, write, read
The three values are regexes: configure, write, read. Use "" to deny.

POLICIES

`rabbitmqctl` **`list_policies -p`** *`<vhost>`*
List policies in a vhost

`rabbitmqctl` **`set_policy`** *`<name> "<pattern>" '<json>'`*
Apply settings to matching queues
The definition is a JSON object, for example {"max-length":10000}.

CLUSTERING

`rabbitmqctl` **`cluster_status`**
Members, alarms, partitions

`rabbitmqctl` **`stop_app && rabbitmqctl reset`**
Wipe a node back to blank
Deletes the node's data and removes it from its cluster; stop_app must come first.

`rabbitmqctl` **`join_cluster`** *`<seed-node>`*
Join this node to a cluster
The app must be stopped; the node is reset first.

CLUSTERING (cont.)

`rabbitmqctl` **`forget_cluster_node`** *`<node>`*
Remove a dead node from the cluster

PLUGINS AND RABBITMQADMIN

`rabbitmq-plugins list`
Show plugins and their state

`rabbitmq-plugins list -E`
Only explicitly enabled plugins

`rabbitmq-plugins enable` *`<plugin>`*
Enable a plugin

`rabbitmq-plugins disable` *`<plugin>`*
Disable a plugin

`rabbitmqadmin queues list`
List queues over the HTTP API

`rabbitmqadmin --vhost` *`<vhost>`* **`queues declare --name`** *`<queue>`* **`--type quorum --durable true`**
Create a durable quorum queue

rabbitmqadmin v2 talks to the management HTTP API, so it needs the rabbitmq_management plugin. Global flags like --vhost go before the command group.