# rabbitmqctl 指南

# 名称

rabbitmqctl — 用于管理中间件的命令行工具

# 参照来源url

https: //blog.csdn.net/wulex/article/details/64127224:

# 语法

rabbitmqctl [-n node] [-t timeout] [-q] {command} [command options…]

# 描述

RabbitMQ是AMQP的实现, 后者是高性能企业消息通信的新兴标准. RabbitMQ server是AMQP 中间件健壮的，可扩展的实现.rabbitmqctl 用来管理RabbitMQ中间件的命令行工具.它通过连接中间件节点来执行所有操作。如果中间件没有运行，将会显示诊断信息, 不能到达，或因不匹配Erlang cookie而拒绝连接.

# 选项

[-n node]默认节点是”rabbit@server”,此处的server是本地主机. 在一个名为”server.example.com”的主机上, RabbitMQ Erlang node 的节点名称通常是rabbit@server (除非RABBITMQ\_NODENAME在启动时设置了非默认值). hostname -s 的输出通常是”@” 标志后的东西.查看rabbitmq-server(1)来了解配置RabbitMQ broker的细节.

[-q] 使用-q标志来启用宁静(quiet)模式，这会一致消息输出.

[-t timeout] 操作超时时间（秒为单位）. 只适用于"list" 命令. 默认是无穷 大.

# 1. 应用管理

# 1.1 stop [pid\_file] (用于停止运行RabbitMQ的Erlang node)

如果指定了pid\_file，还将等待指定的过程结束。例如：rabbitmqctl stop此命令会终止RabbitMQ node的运行

# 1.2 stop\_app (停止RabbitMQ application,但Erlang node会继续运行）

此命令主要用于优先执行其它管理操作（这些管理操作需要先停止RabbitMQ application），如. reset.例如：rabbitmqctl stop\_app

stop与stop\_app区别:stop是彻底停止包括rabbitmq node和application, stop\_app只是停止了application，应用停止了但服务还存在

# 1.3 start\_app (启动RabbitMQ application)

此命令典型用于在执行了其它管理操作之后，重新启动停止的RabbitMQ application。如reset.例如：rabbitmqctl start\_app此命令来指导RabbitMQ node来启动RabbitMQ application。

# 1.4 wait {pid\_file} (等待RabbitMQ application启动)

此命令用来等待RabbitMQ application来启动node。它会等待创建pid文件，然后等待pid文件中的特定pid过程启动，最后等待RabbitMQ application 来启动node。 pid\_file是通过rabbitmq-server 脚本来创建的.默认情况下，它存放于Mnesia目录中. 修改RABBITMQ\_PID\_FILE 环境变量可以改变此位置 rabbitmqctl wait /var/run/rabbitmq/pid 此命令会在RabbitMQ node启动后返回。返回服务的pid进程编号

# 1.5 reset (将RabbitMQ node还原到最初状态)

包括从所在群集中删除此node,从管理数据库中删除所有配置数据，如已配置的用户和虚拟主机，以及删除所有持久化消息.执行reset和force\_reset之前，必须停止RabbitMQ application ,如使用stop\_app.

示例:rabbitmqctl reset此命令会重设RabbitMQ　node.

# 1.6 force\_reset（强制RabbitMQ node还原到最初状态）

不同于reset ， force\_reset 命令会无条件地重设node,不论当前管理数据库的状态和集群配置是什么. 它只能在数据库或集群配置已损坏的情况下才可使用。

reset,force\_rest都是将rabbitmq node还原到初始状态，使用前都必须先停止RabbitMQ application stop\_app

# 1.7 rotate\_logs {suffix} （指示RabbitMQ node循环日志文件）

RabbitMQ 中间件会将原来日志文件中的内容追加到原始名称和后辍的日志文件中，然后再将原始日志文件内容复制到新创建的日志上。实际上，当前日志内容会移到以此后辍结尾的文件上。当目标文件不存在时，将会进行创建。如果不指定后辍，则不会发生循环，日志文件只是重新打开。

示例：rabbitmqctl rotate\_logs .1

此命令指示RabbitMQ node将日志文件的内容追加到新日志文件（文件名由原日志文件名和.1后辍构成）中。如. rabbit@mymachine.log.1 和 rabbit@mymachine-sasl.log.1. 最后, 日志会在原始位置恢复到新文件中.

# 2集群管理

# 2.1 join\_cluster {clusternode} [–ram]（指导节点成为集群中的一员）

在加入集群之前，节点会重置,因此在使用此命令时，必须小心. 这个命令要成功,RabbitMQ应用程序必须先停止,如stop\_app,特别注意是停止当前服务，不是已存在的集群中的服务。集群节点可以是两种类型: 磁盘节点(Disc Node) 或 内存节点(RAM Node).磁盘节点会在RAM和磁盘中复制数据, 通过冗余可以防止节点失效事件，并可从断电这种全局事件中进行恢复. RAM节点只在RAM中复制数据(除了队列的内容外，还依赖于队列是否是持久化的或者内容对于内存来说是否过大) ，并主要用于可伸缩性. RAM节点只有当管理资源(如，增加/删除队列,交换机,或绑定)的时候才具有更高的性能.一个集群必须至少有一个磁盘节点，通常来说还不止一个.

默认情况下，节点是磁盘节点.如果你想要创建内存节点,需要提供--ram 标志.

在执行cluster命令之后, 无论何时，当前节点上启动的RabbitMQ 应用程序在节点宕机的情况下，会尝试连接集群中的其它节点。

要脱离集群, 必须重设(reset)节点. 你也可以通过forget\_cluster\_node 命令来远程删除节点

例如: rabbitmqctl join\_cluster rabbitmq@01 --ram

此命令用于指示RabbitMQ node以ram节点的形式将 rabbitmq@01加入集群

# 2.2 cluster\_status（按节点类型来分组展示集群中的所有节点，包括当前运行的节点）

例如:rabbitmqctl cluster\_status 此命令会显示集群中的所有节点.

# 2.3 change\_cluster\_node\_type {disc | ram}（修改集群节点的类型）

要成功执行此操作，必须首先停止节点，要将节点转换为RAM节点，则此节点不能是集群中的唯一disc节点。

例如: rabbitmqctl change\_cluster\_node\_type disc

此命令会将一个RAM节点转换为disc节点.

# 2.4 forget\_cluster\_node [–offline]（允许节点从脱机节点中删除）

这只在所有节点都脱机且最后一个掉线节点不能再上线的情况下有用，从而防止整个集群从启动。它不能使用在其它情况下，因为这会导致不一致．

远程删除一个集群节点.要删除的节点必须是脱机的, 而在删除节点期间节点必须是在线的，除非使用了--offline 标志.

当使用--offline 标志时，rabbitmqctl不会尝试正常连接节点;相反，它会临时改变节点以作修改.如果节点不能正常启动的话，这是非常有用的.在这种情况下，节点将变成集群元数据的规范源（例如，队列的存在），即使它不是以前的。因此，如果有可能，你应该在最新的节点上使用这个命令来关闭。

例如:rabbitmqctl -n rabbitmq@01 forget\_cluster\_node rabbitmq@02

此命令会从节点rabbitmq@01中删除rabbitmq@02节点.

# 2.5 rename\_cluster\_node {oldnode1} {newnode1} [oldnode2] [newnode2 …] (支持在本地数据库中重命名集群节点)

此子命令会促使rabbitmqctl临时改变节点以作出修改. 因此本地集群必须是停止的，其它节点可以是在线或离线的。这个子命令接偶数个参数，成对表示节点的旧名称和新名称.你必须指定节点的旧名称和新名称，因为其它停止的节点也可能在同一时间重命名。同时停止所有节点来重命名也是可以的(在这种情况下，每个节点都必须给出旧名称和新名称)或一次停止一个节点来重命名(在这种情况下，每个节点只需要被告知其名句是如何变化的)。

eg: rabbitmqctl rename\_cluster\_node rabbit@01 rabbit@02

此命令来将节点名称rabbit@01 重命名为rabbit@02

# 2.6 update\_cluster\_nodes {clusternode}

clusternode :用于咨询具有最新消息的节点.

指示已集群的节点醒来时联系clusternode.这不同于join\_cluster ，因为它不会加入任何集群 - 它会检查节点已经以clusternode的形式存在于集群中了．

需要这个命令的动机是当节点离线时，集群可以变化.考虑这样的情况，节点Ａ和节点Ｂ都在集群里边，这里节点Ａ掉线了，Ｃ又和Ｂ集群了，然后Ｂ又离开了集群．当Ａ醒来的时候，它会尝试联系Ｂ，但这会失败，因为Ｂ已经不在集群中了.update\_cluster\_nodes -n A C 可解决这种场景．

# 2.7 force\_boot

确保节点将在下一次启动，即使它不是最后一个关闭的。通常情况下，当你关闭整个RabbitMQ 集群时，你重启的第一个节点应该是最后一个下线的节点，因为它可以看到其它节点所看不到的事情. 但有时这是不可能的:例如，如果整个集群是失去了电力而所有节点都在想它不是最后一个关闭的。

在这种节点掉线情况下，你可以调用rabbitmqctl force\_boot ．这就告诉节点下一次无条件的启动节点.在此节点关闭后，集群的任何变化，它都会丢失．

如果最后一个掉线的节点永久丢失了，那么你需要优先使用rabbitmqctl forget\_cluster\_node --offline, 因为它可以确保在丢失的节点上掌握的镜像队列得到提升。

eg: rabbitmqctl force\_boot

这可以强制节点下次启动时不用等待其它节点

# 2.8 set\_cluster\_name {name} (设置集群名称)

集群名称在client连接时，会通报给client,也可用于federation和shovel插件记录消息的来源地. 群集名称默认是来自在群集中的第一个节点的主机名，但可以改变。

eg：rabbitmqctl set\_cluster\_name rabbitmq001

设置集群名称为"rabbitmq001".

# 3 队列管理

# 3.1 sync\_queue [-p vhost] {queue} (同步queue队列)

指示未同步slaves上的镜像队列自行同步.同步发生时，队列会阻塞(所有出入队列的发布者和消费者都会阻塞).此命令成功执行后，队列必须是镜像的．

注意，未同步队列中的消息被耗尽后，最终也会变成同步. 此命令主要用于未耗尽的队列。

# 3.2 cancel\_sync\_queue[-p vhost]{queue} (取消同步queue队列)

# 3.3 purge\_queue [-p vhost] {queue} (清除queue队列(删除其中的所有消息))

# 4 用户管理

注意rabbitmqctl 管理RabbitMQ 内部用户数据库. 任何来自其它认证后端的用户对于rabbitmqctl来说是不可见的

# 4.1 add\_user {username} {password}(添加一个非管理员权限的用户)

eg: rabbitmqctl add\_user admin admin123

此命令用于指示RabbitMQ broker 创建一个拥有非管理权限的用户，其名称为admin， 初始密码为admin123.

# 4.2 delete\_user {username}(删除的用户)

eg：rabbitmqctl delete\_user admin

此命令用于指示RabbitMQ broker删除名为admin的用户

# 4.3 change\_password {username} {newpassword} (修改用户密码)

eg: rabbitmqctl change\_password admin newAdmin123

此命令用于指定RabbitMQ broker将admin用户的密码修改为newAdmin123.

# 4.4 clear\_password {username} (清除用户密码)

eg: rabbitmqctl clear\_password admin

此命令会指示RabbitMQ broker清除名为admin的用户密码.现在，此用户不能使用密码登录(但可以通过SASL EXTERNAL登录，如果配置了的话).

# 4.5 authenticate\_user {username} {password}

rabbitmqctl authenticate\_user admin admin

此命令会指示RabbitMQ broker以名称为admin， 密码为admin来进行验证.

# 4.6 set\_user\_tags {username} {tag …}（设置用户角色）

eg：rabbitmqctl set\_user\_tags admin administrator

此命令指示RabbitMQ broker用于确保admin 是administrator.当通过AMQP来登录时，这没有什么效果,但用户通过其它的途经来登录时，它可用来管理用户,虚拟主机和权限(如使用管理插件).rabbitmqctl set\_user\_tags admin

此命令会指示RabbitMQ broker删除admin上的任何现有的tag.

# 4.7 list\_users（列出用户. 每个结果行都包含用户名，其后紧跟用户的tags）

eg： rabbitmqctl list\_users

此命令指示RabbitMQ broker列出所有用户.

# 5 访问权限管理

注意rabbitmqctl 会管理RabbitMQ的内部用户数据库. 无权限的用户将不能使用rabbitmqctl.

# 5.1 add\_vhost {vhost}（创建一个虚拟主机）

eg: rabbitmqctl add\_vhost test

此命令指示RabbitMQ broker来创建一个新的名为test的虚拟主机.

# 5.2 delete\_vhost {vhost}（创建一个虚拟主机）

eg: rabbitmqctl delete\_vhost test

此命令指示RabbitMQ broker删除名为test的虚拟主机, 删除一个虚拟主机，同时也会删除所有交换机，队列，绑定，用户权限，参数和策略.

# 5.3 list\_vhosts [vhostinfoitem …]（列出所有虚拟主机）

vhostinfoitem 参数用于标识哪些虚拟主机应该包含在结果集中.结果集中的列顺序会匹配参数的顺序.vhostinfoitem 可接受下面的值：

name :虚拟主机的名称.

tracing :是否对虚拟主机启用追踪.

如果没有指定vhostinfoitem 参数，那么会显示虚拟主机名称.

例如：rabbitmqctl list\_vhosts name tracing

此命令用于指示RabbitMQ broker显示所有虚拟主机.

# 5.4 set\_permissions [-p vhost] {user} {conf} {write} {read}（配置用户虚拟主机权限）

vhost：授予用户可访问的虚拟机名称，默认是/.

user ：可访问指定虚拟主机的用户名称.

conf ：一个用于匹配用户在哪些资源名称上拥有配置权限的正则表达式

write：一个用于匹配用户在哪些资源名称上拥有写权限的正则表达式.

read ：一个用于匹配用户在哪些资源名称上拥有读权限的正则表达式.

设置用户权限.

例如:

rabbitmqctl set\_permissions -p /myvhost tonyg "^tonyg-.\*" ".\*" ".\*"

此命令表示RabbitMQ broker授予tonyg 用户可访问 /myvhost虚拟主机,并在资源名称以"tonyg-"开头的所有资源上都具有配置权限，并在所有资源上都拥有读写权限。

# 5.5 clear\_permissions [-p vhost] {username}（清除用户虚拟主机权限）

vhost ：用于设置禁止用户访问的虚拟主机名称,默认为/.

username ：禁止访问特定虚拟主机的用户名称.

设置用户权限.

例如:rabbitmqctl clear\_permissions -p /myvhost tonyg

此命令用于指示RabbitMQ broker禁止tonyg 用户访问/myvhost虚拟主机.

# 5.6 list\_permissions [-p vhost]

vhost:用于指定虚拟主机名称，将会列出所有可访问此虚拟主机的所有用户名称和权限.默认为/.显示虚拟机上权限.

例如:rabbitmqctl list\_permissions -p /myvhost

此命令指示RabbitMQ broker列出所有已授权访问/myvhost 虚拟主机的用户，同时也会列出这些用户能在虚拟主机资源可操作的权限.注意，空字符串表示没有任何授予的权限。

# 5.7 list\_user\_permissions {username}

username:要显示权限的用户名称.

列出用户权限.

例如:rabbitmqctl list\_user\_permissions tonyg

此命令指示RabbitMQ broker列出tonyg可授权访问的所有虚拟主机名称，以及在这些虚拟主机上的操作.

# 6 参数管理

RabbitMQ的某些特性(如联合插件)是动态控制的. 每个参数都是与特定虚拟主机相关的组件名称, name和value构成的. 组件名称和name都是字符串,值是Erlang term. 参数可被设置，清除和显示.通常你可以参考文档来了解如何设置参数.

# 6.1 set\_parameter [-p vhost] {component\_name} {name} {value} （设置一个参数）

component\_name ： 要设置的组件名称.

name： 要设置的参数名称.

value ：要设置的参数值，作不JSON项。在多数shells中，你更喜欢将其引起来.

例如:rabbitmqctl set\_parameter federation local\_username '"guest"'

此命令用于在默认虚拟主机上设置federation 组件的local\_username 参数值"guest".

# 6.2 clear\_parameter [-p vhost] {component\_name} {key}（清除参数）

component\_name 要清除参数的组件名称.

name要清除的参数名称.

例如:rabbitmqctl clear\_parameter federation local\_username

此命令用于清除默认虚拟主机上的federation 组件的local\_username 参数值.

# 6.3 list\_parameters [-p vhost] （列出虚拟主机上的所有参数）

eg:rabbitmqctl list\_parameters

此命令用于列出默认虚拟主机上的所有参数.

# 7 策略管理

策略用于在集群范围的基础上用于控制和修改队列和交换机的行为. 策略应用于虚拟主机,由name, pattern, definition或可选的priority组成. 策略可被设置，清除和列举.

# 7.1 set\_policy [-p vhost] [–priority priority] [–apply-to apply-to] {name} {pattern} {definition} （设置策略）

name策略名称.

pattern正则表达式, 匹配要应用的资源

definition 策略的定义,JSON形式．在大多数shells中，你很可能需要引用这个

priority策略的整数优先级. 数字越高则优先级越高.默认是０.

apply-to 策略适用的对象类型，其值可为 "queues", "exchanges" 或 "all".默认是"all".

eg:

rabbitmqctl set\_policy federate-me "^amq." '{"federation-upstream-set":"all"}'

此命令在默认虚拟主机上设置策略为federate-me，这样内建的交换器将进行联合.

# 7.2 clear\_policy [-p vhost] {name} （清除策略）

name要清除的策略名称.

例如:rabbitmqctl clear\_policy federate-me

此命令来清除默认虚拟主机上的federate-me 策略.

# 7.3 list\_policies [-p vhost] （显示虚拟主机上的所有策略）

例如:rabbitmqctl list\_policies

此命令会显示默认虚拟主机上的所有策略.

# 8 服务器状态

服务器状态查询查询服务器返回一个结果以制表符分隔的列表. 某些查询(list\_queues, list\_exchanges, list\_bindings, 和 list\_consumers) 接受一个可选的vhost 参数. 如果这个参数出现了，那么它必须指定在查询的后面.

list\_queues, list\_exchanges and list\_bindings 命令接受一个可选的虚拟主机参数以显示其结果．默认值为”/”.

# 8.1 list\_queues [-p vhost] [queueinfoitem …] （返回队列的详细信息）

如果无-p标志，将显示／虚拟主机上的队列详情．"-p" 标志可用来覆盖此默认值．

queueinfoitem 参数用于指示哪些队列信息项会包含在结果集中.结果集的列顺序将匹配参数的顺序.queueinfoitem 可以是下面列表中的任何值:

name 非ASCII字符的队列名称.

durable服务器重启后，队列是否能幸存.

auto\_delete不再使用时，是否需要自动删除队列.

arguments队列参数.

policy 应用到队列上的策略名称.

pid关联队列的Erlang进程ID.

owner\_pid表示队列专用所有者的代表连接的Erlang进程ID.如果队列是非专用的，此值将为空．

exclusive True:如果队列是专用的(即有owner\_pid), 反之false

exclusive\_consumer\_pid 表示此channel的专用消费者订阅到此队列的Erlang进程Id. 如果没有专用消费者，则为空．

exclusive\_consumer\_tag 专用消费者订阅到此队列的Consumer tag.如果没有专用消费者，则为空.

messages\_ready 准备分发给客户端的消息数目.

messages\_unacknowledged 分发到客户端但尚未应答的消息数目.

messages 准备分发和未应答消息的总和(队列深度).

messages\_ready\_ram 驻留在ram中messages\_ready的消息数目.

messages\_unacknowledged\_ram 驻留在ram中messages\_unacknowledged的消息数目.

messages\_ram 驻留在ram中的消息总数.

messages\_persistent 队列中持久化消息的数目(对于瞬时队列总是0).

message\_bytes 队列中所有消息体的大小总和.这不包括消息属性(包括headers) 或任何开销(overhead)。

message\_bytes\_ready 类似于message\_bytes ，但只统计准备投递给clients的那些消息．

message\_bytes\_unacknowledged 类似于message\_bytes ，但只统计那些已经投递给clients但还未应答的消息

message\_bytes\_ram 类似于message\_bytes ，但只统计那些在RAM中的消息

message\_bytes\_persistent 类似于message\_bytes ，但只统计那些持久化的消息

head\_message\_timestamp 如果存在，只显示队列中第１个消息的timestamp属性. 消息的时间戳只出现在分页情况下．

disk\_reads 从队列启动开如，已从磁盘上读取该队列的消息总次数.

disk\_writes 从队列启动开始，已向磁盘队列写消息总次数.

consumers 消费者数目.

consumer\_utilisation 时间分数(0.0与1.0之间)，队列可立即向消费者投递消息. 它可以小于1.0，如果消费者受限于网络堵塞或预提取数量.

memory 与队列相关的Erlang进程消耗的内存字节数,包括栈,堆以及内部结构.

slave\_pids 如果队列是镜像的，这里给出的是当前slaves的IDs.

synchronised\_slave\_pids 如果队列是镜像的，当前slaves的IDs是master同步的- 即它们可在无消息丢失的情况下，接管master.

state 队列状态.正常情况下是'running', 但如果队列正在同步也可能是"{syncing, MsgCount}". 处于集群下的节点如果掉线了，队列状态交显示'down' (大多数queueinfoitems 将不可用).

如果没有指定queueinfoitems，那么将显示队列名称和队列深度．

eg: rabbitmqctl list\_queues -p /myvhost messages consumers

此命令显示了/myvhost虚拟主机中每个队列的深度和消费者数目.

# 8.2 list\_exchanges [-p vhost] [exchangeinfoitem …] （返回交换器细节）

如果没有指定"-p"选项，将返回 / 虚拟主机的细节. "-p" 选项可用来覆盖默认虚拟主机.

exchangeinfoitem 参数用来表示哪些交换器信息要包含在结果中. 结果集中列的顺序将与参数顺序保持一致. exchangeinfoitem 可接受下面的列表中任何值：

name 交换器名称.

type 交换器类型(如[direct, topic, headers, fanout]).

durable 当服务器重启时，交换器是否能复活.

auto\_delete 当不再使用时，交换器是否需要自动删除.

internal 交换器是否是内部的，即不能由client直接发布.

arguments 交换器参数

policy应用到交换器上的策略名称.

如果没有指定exchangeinfoitems，那么将会显示交换器类型和类型

eg: rabbitmqctl list\_exchanges -p /myvhost name type.

此命令会显示/myvhost中每个交换器的名称和类型.

# 8.3 list\_bindings [-p vhost] [bindinginfoitem …]（返回绑定细节）

默认情况下返回的是 / 虚拟主机上的绑定详情.可使用"-p" 标记来覆盖默认虚拟主机.

bindinginfoitem 参数用来表示结果中包含哪些绑定信息. 结果集中列的顺序将匹配参数的顺序.bindinginfoitem可接受下面列表的任意值:

source\_name 绑定中消息来源的名称. C中非ASCII转义字符.

source\_kind 绑定中消息来源的类别.当前总是exchange. C中非ASCII转义字符.

destination\_name 绑定中消息目的地名称.C中非ASCII转义字符.

destination\_kind 绑定中消息目的地的种类. C中非ASCII转义字符.

routing\_key 绑定的路由键,C中非ASCII转义字符.

arguments 绑定参数.

如果没有指定bindinginfoitems，将会显示所有上述条目．

例如:

rabbitmqctl list\_bindings -p /myvhost exchange\_name queue\_name

此命令来显示/myvhost虚拟主机上绑定的交换器名称和队列名称.

# 8.4 list\_connections [connectioninfoitem …]（返回TCP/IP连接统计）

connectioninfoitem 参数用来表示在结果中包含哪些连接信息. 结果集中列的顺序将匹配参数的顺序.

connectioninfoitem可接受下面列表的任意值:

pid 与连接相关的Erlang进程ID.

name 连接的可读名称.

port 服务器端口.

host 返回反向DNS获取的服务器主机名称，或 IP地址(反向DNS解析失败) 或者未启用．

peer\_port Peer 端口.

peer\_host　返回反向DNS获取的Peer主机名称，或 IP地址(反向DNS解析失败) 或者未启用．

ssl 用Boolean来表示连接是否是SSL的.

ssl\_protocol SSL 协议(如. tlsv1)

ssl\_key\_exchange SSL key exchange 算法 (如 rsa)

ssl\_cipher SSL cipher 算法 (如aes\_256\_cbc)

ssl\_hash SSL hash 函数 (如 sha)

peer\_cert\_subject peer的 SSL 安全证书的主体, RFC4514形式．

peer\_cert\_issuer peer的 SSL安全证书的发行者, RFC4514 形式.

peer\_cert\_validity peer的SSL安全证书的有效期.

state 连接状态(可为[starting, tuning, opening, running, flow, blocking, blocked, closing, closed]其中一个).

channels 使用连接的channel数。

protocol 使用的AMQP协议版本(当前是{0,9,1} 或{0,8,0}). 注意，如果client请求的是AMQP 0-9 连接, 我们会视为AMQP 0-9-1.

auth\_mechanism 使用的SASL认证机制,如PLAIN.

user 与连接相关的用户名

vhost 虚拟主机名称，Ｃ中非ASCII转义字符.

timeout 连接超时／协商的心跳间隔,秒为单位.

frame\_max 最大 frame 大小(字节).

channel\_max 此连接上channel的最大数目.

client\_properties 连接建立期间由client发送的信息属性.

recv\_oct Octets已收到.

recv\_cnt Packets 已收到.

send\_oct Octets 发送.

send\_cnt Packets 发送.

send\_pend 发送队列大小.

connected\_at 连接建立的日期和时间,当作timestamp.

如果没有connectioninfoitems， 那么会显示user, peer host, peer port,流量控制和内存块状态的时间

例如:rabbitmqctl list\_connections send\_pend port

此命令会显示发送队列的大小以及第个连接的服务器端口．

# 8.5 list\_channels [channelinfoitem …]（返回所有当前channel上的信息）

逻辑容器执行大部分 AMQP命令.这将包含最初AMQP连接的部分，以及不同插件和其它扩展创建的channels.

channelinfoitem 参数用来表示在结果集中包含哪些channel信息．结果集中列的顺序将匹配参数的顺序. channelinfoitem 可接受下面列表中的任何一个参数：

pid 与连接相关的Erlang进程ID.

connection channel所属的连接Erlang进程ID.

name channel的可读名称.

number channel的数目，在一个连接中，它有唯一的标识符．

user 与channel相关的用户名称.

vhost channel操作的虚拟主机.

transactional True：如果channel处于事务模式,其它情况为false.

confirm True：如果channel是确认模式，其它情况为false.

consumer\_count 在channel中接收消息的逻辑AMQP消费者数目．

messages\_unacknowledged 在channel中消息已投递但还未应答的消息数目．

messages\_uncommitted 在channel中已收到消息但还没有提交事务的消息个数．

acks\_uncommitted 确认收到一个还未提交的事务数。

messages\_unconfirmed 尚未确认已发布消息的数目。在通道不在确认模式下时，这将是0。

prefetch\_count 新消费者QoS预提取限制, 0表示无上限.

global\_prefetch\_count 整个channel　QoS预提取限制, 0表示无上限.

如果没有指定channelinfoitems，那么将显示pid, user, consumer\_count,messages\_unacknowledged．

例如:rabbitmqctl list\_channels connection messages\_unacknowledged

此命令会显示每个channel中连接进程和未应答消息的数目.

# 8.6 list\_consumers [-p vhost]（列举消费者, 即订阅队列的消息流）

每行将打印出由制表符分隔的已订阅队列的名称，创建并管理订阅的channel进程的标识,channel中订阅的consumer tag唯一标识符, boolean值表示投递到此消费者的消息是否需要应答,整数值表示表示预提取限制(为０表示无限制), 以及关于此消费者的任何其它参数．

# 8.7 status（列举消费者, 即订阅队列的消息流）

显示 broker 状态信息，如当前Erlang节点上运行的应用程序, RabbitMQ 和 Erlang 的版本信息, OS 名称, 内存和文件描述符统计信息. (查看cluster\_status 命令来找出那些节点是集群化的以及正在运行的.)

例如:rabbitmqctl status

此命令显示了RabbitMQ broker的相关信息.

# 8.8 environment（显示每个运行程序环境中每个变量的名称和值）

# 8.9 report

为所有服务器状态生成一个服务器状态报告，输出应该重定向到一个文件.

例如:rabbitmqctl report > server\_report.txt

此命令创建了一个服务器报告，可将它附着在支持请求的电子邮件中．

# 9 其他杂项

# 9.1 eval {expr} (执行任意Erlang表达式)

例如:rabbitmqctl eval 'node().'

此命令用于返回rabbitmqctl连接的节点名称

# 9.2 close\_connection {connectionpid} {explanation}

connectionpid 要关闭的与连接相关的Erlang进程ID.

explanation 解释字符串.

指示broker关闭与Erlang进程id相关的连接(可通过list\_connections 命令查看), 通过为连接客户端传递解释字符串(作为AMQP连接关闭协议的一部分).

例如:

rabbitmqctl close\_connection "<rabbit@tanto.4262.0>" "go away"

此命令指示RabbitMQ broker关闭与Erlang 进程id<rabbit@tanto.4262.0>相关联的连接, 同时向客户端传递go away字符串.

# 9.3 trace\_on [-p vhost]

vhost 要开启追踪的虚拟主机的名称.

开启追踪.注意，追踪状态不是持久化的; 如果服务器重启，追踪状态将会丢失.

# 9.4 trace\_off [-p vhost]

vhost 要停止追踪的虚拟主机名称

停止追踪

# 9.5 set\_vm\_memory\_high\_watermark {fraction}

fraction

当一个浮点数大于或等于0时，会触发流量控制新内存阈值部分。

# 9.6 set\_vm\_memory\_high\_watermark absolute {memory\_limit}

memory\_limit

流程控制触发的新内存限制, 以字节来表示大于或等于０的整数或以字符串和内存单位来表示(如 512M或1G). 可用的单位是: k, kiB: kibibytes (2^10 bytes) M, MiB: mebibytes (2^20) G, GiB: gibibytes (2^30) kB: kilobytes (10^3) MB: megabytes (10^6) GB: gigabytes (10^9)

# 9.7 set\_disk\_free\_limit {disk\_limit}

disk\_limit 以整数或字符串单位的可用磁盘下限限制(查看vm\_memory\_high\_watermark), 如 512M or 1G. 一旦可用磁盘空间达到这个限制，就会设置磁盘报警.

# 9.8 set\_disk\_free\_limit mem\_relative {fraction}

fraction 相对于整个可用内存的限制，其值为非负浮点数. 当值小于１.０时是很危险的，应该谨慎使用．