# 创建RabbitMQ集群

## 北京短信模型集群

## RabbitMQ

MQ（messages quene）消息队列提供了一个异步通信协议，消息的发送者不用一直等待消息是否被成功处理，而是发完消息立即返回。消息被暂存于队列之中，对消息感兴趣的消费者会订阅消息并处理它们。

消息队列使用场景：

1 ：需要将同一条消息发送给多个感兴趣的系统， 这时用消息队列不用把同一条消息分别发送给它们

2：系统的同步处理方式严重影响性能及吞吐量时，如用户订单处理、消息通知等修改为异步处理方式

RabbitMQ是实现了高级消息队列协议（AMQP）的开源消息代理软件（亦称面向消息的中间件）。RabbitMQ服务器是用Erlang语言编写的，而群集和故障转移是构建在开放电信平台框架上的。所有主要的编程语言均有与代理接口通讯的客户端库。易用性、扩展性、可靠性和高可用性等方面表现卓越。

RabbitMQ 是信息传输的中间者。本质上，他从生产者（producers）接收消息，转发这些消息给消费者（consumers）.换句话说，他能够按根据你指定的规则进行消息转发、缓冲、和持久化。

RabbitMQ消息传递过程：生产者创建消息，然后发布到RabbitMQ服务器。消息的内容包含两部分内容，消息标签和消息体。消息体就是你想要传递的数据。它可以是任何格式的数据，例如JSON数据。消息标签用来描述消息体，RabbitMQ服务器通过它决定由谁获得消息体的内容。消费者连接到RabbitMQ,并订阅到队列（Quenue）上。

RabbitMQ的使用流程：

（1）客户端连接到消息队列服务器，打开一个channel。　　（2）客户端声明一个exchange，并设置相关属性。　　（3）客户端声明一个queue，并设置相关属性。　　（4）客户端使用routing key，在exchange和queue之间建立好绑定关系。　　（5）客户端投递消息到exchange。

RabbitMQ的基本要素：Broker：简单来说就是消息队列服务器实体。　　Exchange：消息交换机，它指定消息按什么规则，路由到哪个队列。　　Queue：消息队列载体，每个消息都会被投入到一个或多个队列。　　Binding：绑定，它的作用就是把exchange和queue按照路由规则绑定起来。　　Routing Key：路由关键字，exchange根据这个关键字进行消息投递。　　vhost：虚拟主机，一个broker里可以开设多个vhost，用作不同用户的权限分离。　　producer：消息生产者，就是投递消息的程序。　　consumer：消息消费者，就是接受消息的程序。　　channel：消息通道，在客户端的每个连接里，可建立多个channel，每个channel代表一个会话任务。

1. docker安装及RabbitMQ镜像下载

rabbitMQ镜像可在阿里云上搜索并下载：<https://dev.aliyun.com/detail.html?spm=5176.1971733.2.16.6mF2ln&repoId=1256>



注意使用后缀为"-management"的镜像版本，是包含网页控制台的

安装docker（mirrors.aliyun.com）

若有外网可直接用yum -y install docker 安装docker

若没外网，可提前下载好docker的依赖包：

(1/28): container-selinux-2.107-1.el7\_6.noarch.rpm

(2/28): container-storage-setup-0.11.0-2.git5eaf76c.el7.noarch.rpm

(3/28): PyYAML-3.10-11.el7.x86\_64.rpm

(4/28): atomic-registries-1.22.1-26.gitb507039.el7.centos.x86\_64.rpm

(5/28): checkpolicy-2.5-8.el7.x86\_64.rpm

(6/28): containers-common-0.1.37-1.el7.centos.x86\_64.rpm

(7/28): audit-libs-python-2.8.4-4.el7.x86\_64.rpm

(8/28): docker-client-1.13.1-102.git7f2769b.el7.centos.x86\_64.rpm

(9/28): docker-common-1.13.1-102.git7f2769b.el7.centos.x86\_64.rpm

(10/28): libseccomp-2.3.1-3.el7.x86\_64.rpm

(11/28): libcgroup-0.41-20.el7.x86\_64.rpm

(12/28): libsemanage-python-2.5-14.el7.x86\_64.rpm

(13/28): libyaml-0.1.4-11.el7\_0.x86\_64.rpm

(14/28): oci-register-machine-0-6.git2b44233.el7.x86\_64.rpm

(15/28): oci-systemd-hook-0.2.0-1.git05e6923.el7\_6.x86\_64.rpm

(16/28): oci-umount-2.5-1.el7\_6.x86\_64.rpm

(17/28): python-IPy-0.75-6.el7.noarch.rpm

(18/28): python-backports-ssl\_match\_hostname-3.5.0.1-1.el7.noarch.rpm

(19/28): python-backports-1.0-8.el7.x86\_64.rpm

(20/28): python-pytoml-0.1.14-1.git7dea353.el7.noarch.rpm

(21/28): python-ipaddress-1.0.16-2.el7.noarch.rpm

(22/28): policycoreutils-python-2.5-29.el7\_6.1.x86\_64.rpm

(23/28): setools-libs-3.3.8-4.el7.x86\_64.rpm

(24/28): yajl-2.0.4-4.el7.x86\_64.rpm

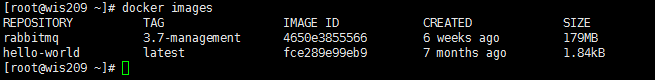
(25/28): policycoreutils-2.5-29.el7\_6.1.x86\_64.rpm

(26/28): subscription-manager-rhsm-certificates-1.21.10-3.el7.centos.x86\_64.rpm

(27/28): docker-1.13.1-102.git7f2769b.el7.centos.x86\_64.rpm

(28/28): python-setuptools-0.9.8-7.el7.noarch.rpm

1. Docker容器常用命令
2. 启动docker服务：systemctl start docker (开机自启：systemctl enable docker)
3. 查看docker本地镜像：docker images



1. 导入镜像：docker load -i /root/rabbitmq.tar 或者 docker load < /dev/rabbitmq.tar
2. 以本地镜像导出镜像：docker save rabbitmq:3.7-management -o /root/rabbitbak.tar

IMG_256

1. 简单启动容器：docker run -it rabbitmq:3.7-management /bin/bash （启动时直接进入容器，exit推出时会关闭容器，后面的/bin/bash是以什么解释器运行，可省略）

docker run -itd rabbitmq:3.7-management（后台运行容器，不会进入容器）

运行时挂载本地目录到容器中：

Docker run -v /var/webroot:/var/www/html -itd dockername /bin/bash

本地目录:容器目录

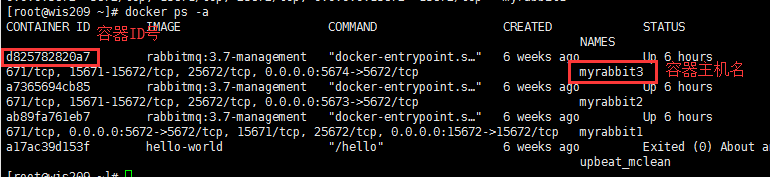
运行时绑定本地端口号给容器：

Docker run -p 8080:8080 -itd dockername /bin/bash

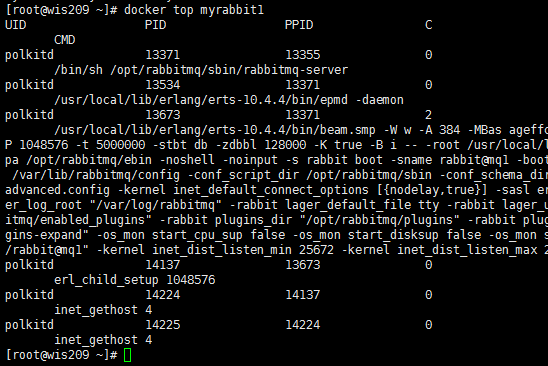
本地端口：容器端口（注意端口不能重复占用）

1. 查看正在运行的容器：docker ps

docker ps -a (连未运行的容器一起看)



1. 查看容器里运行的服务：docker top myrabbit1 (dockername)



1. 进入容器：

docker attach -it 容器ID号 /bin/bash (用exit推出时，容器会一起关闭)

docker exec -it 容器ID号 /bin/bash (用exit推出时，容器不会关闭，常用)

1. 启动、关闭和重启容器

docker start/stop/restart dockername（容器名或ID号）

1. 查看容器的具体信息

docker inspect dockername(容器或ID号)

1. 更改镜像名和标签

docker tag dockernameold dockernamenew

12、容器复制文件到物理机：docker cp 容器名称:容器目录 物理机目录  
物理机复制文件到容器：docker cp 物理机目录 容器名称:容器目录

1. docker运行rabbitmq集群

目前rabbitMQ部署在209上；

模型服务部署在205和206上；

205和206模型服务python代码存储于：

/home/kevin/src

用root账号进入/home/kevin/目录；

1、

source py2env/bin/activate

2、

nohup python -u consumer.py > ./logs/11.out 2>&1 &

nohup python -u consumer.py > ./logs/12.out 2>&1 &

nohup python -u consumer.py > ./logs/13.out 2>&1 &

nohup python -u consumer.py > ./logs/14.out 2>&1 &

开4个独立进程。

一般在205开4个独立进程,206再开4个。

202上传数据/data/weidong/test

注意使用后缀为"-management"的镜像版本，是包含网页控制台的

docker run -d --hostname mq1 --name myrabbit1 -e RABBITMQ\_ERLANG\_COOKIE='rabbitcookie' -p 15672:15672 -p 5672:5672 rabbitmq:3.7-management

docker run -d --hostname mq2 --name myrabbit2 --link myrabbit1:mq1 -e RABBITMQ\_ERLANG\_COOKIE='rabbitcookie' -p 5673:5672 rabbitmq:3.7-management

docker run -d --hostname mq3 --name myrabbit3 --link myrabbit1:mq1 --link myrabbit2:mq2 -e RABBITMQ\_ERLANG\_COOKIE='rabbitcookie' -p 5674:5672 rabbitmq:3.7-management

参数说明：

* -d 后台进程运行
* hostname RabbitMQ主机名称
* name 容器名称
* -p port:port 本地端口:容器端口
* -p 15672:15672 http访问端口
* -p 5672:5672 amqp访问端口

启动完成之后，使用：docker ps 查看程序运行情况。

使用：http://宿主ip:15672 访问，用户名密码使用默认：guest/guest.

**注意点：**

1. 自容器之间互相访问通信的问题，就用到了docker run --link选项，多个容器之间使用“--link”连接，此属性不能少；
2. Erlang Cookie值必须相同，也就是RABBITMQ\_ERLANG\_COOKIE参数的值必须相同，

RabbitMQ是用Erlang实现的，Erlang Cookie相当于不同节点之间相互通讯的秘钥，Erlang节点通过交换Erlang Cookie获得认证。cookie路：/var/lib/rabbitmq/.erlang.cookie

# **docker.service启动失败：Unit not found的原因及解决办法**

解决办法

删除/usr/lib/systemd/system/docker.service的[UNIT]里包含的docker.socket，然后systemctl daemon-reload，最后systemctl start docker.service，发现启动成功了。

类似情况

如果是类似的情况，缺少docker.socket，但是新版本需要docker.socket。有两种方法可以解决该问题:

1. 可以卸载docker，再重新安装，即可出现docker.socket。或者
2. 创建一个/usr/lib/systemd/system/docker.socket文件，然后systemctl  
   daemon-reload，最后systemctl start docker.service，即可启动成功。