**RabbitMQ**

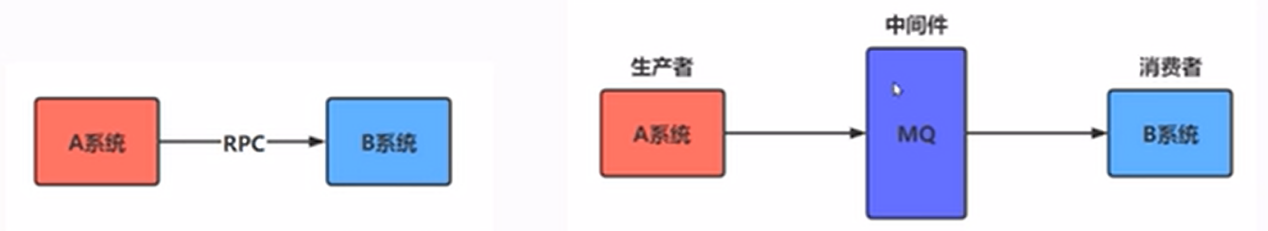
**从0到大牛**

# 1.为什么要学？

解决高并发，达到高可用

# 2.消息队列概念

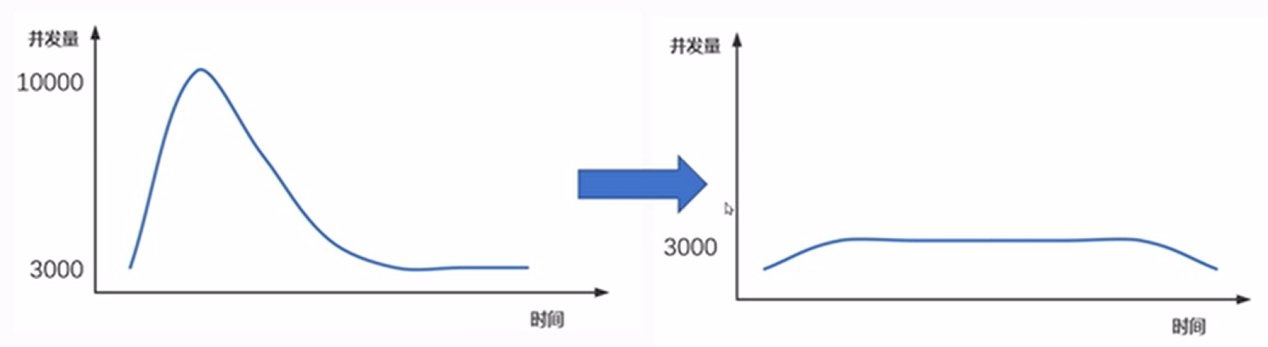
消息队列:MQ(Messge Queue),是在消息的传输过程中保存消息的容器。多用于分布式系统之间的通信



# 3.消息队列的应用场景

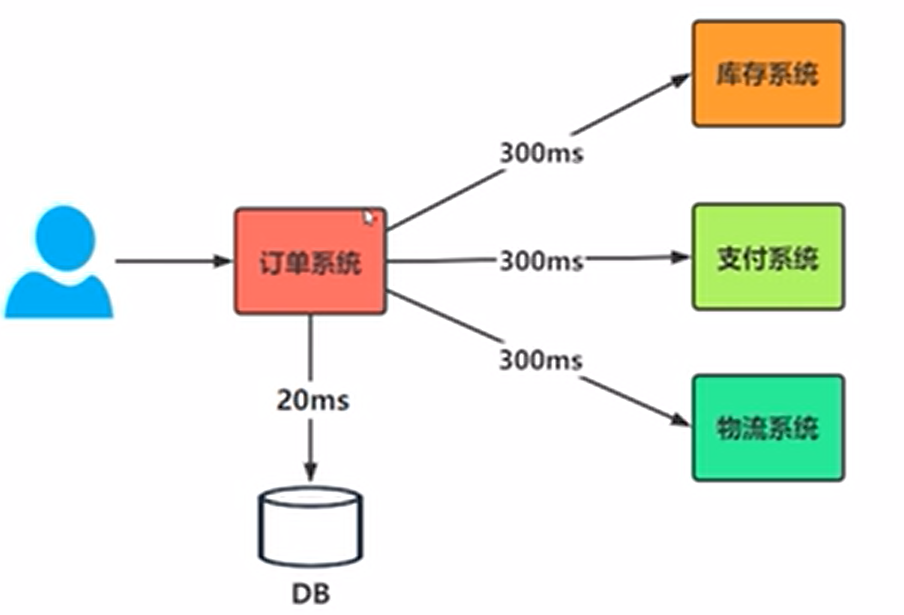
## 3.1削峰填谷

不稳定

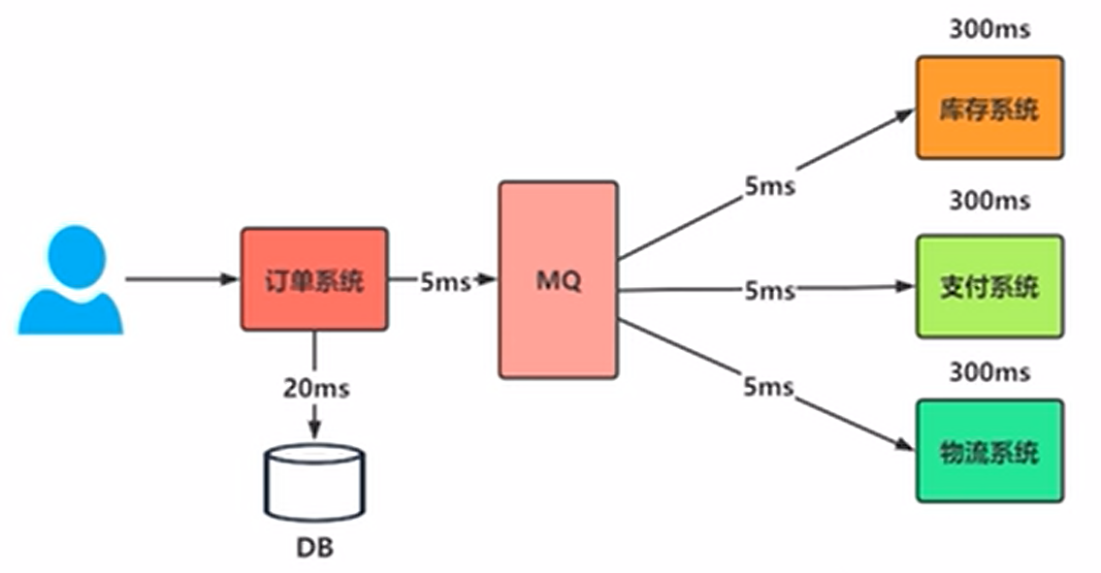


## 3.2应用解耦

问题:耦合度高、容错性低、可维护性差、用户体验差



解决:耦合度低、容错性高、可维护性好、用户体验好

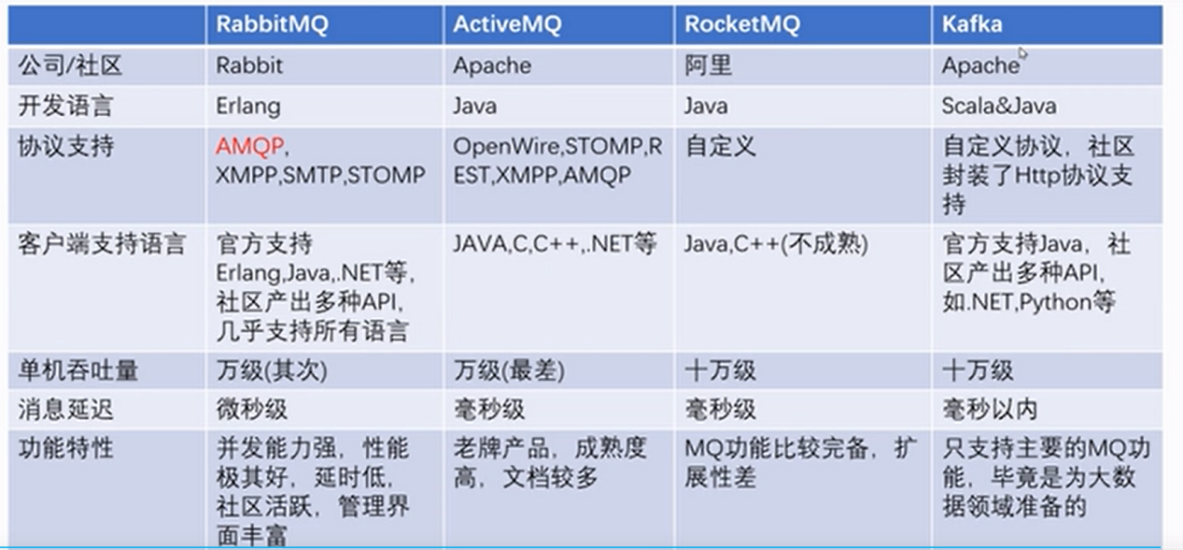


## 3.3异步处理

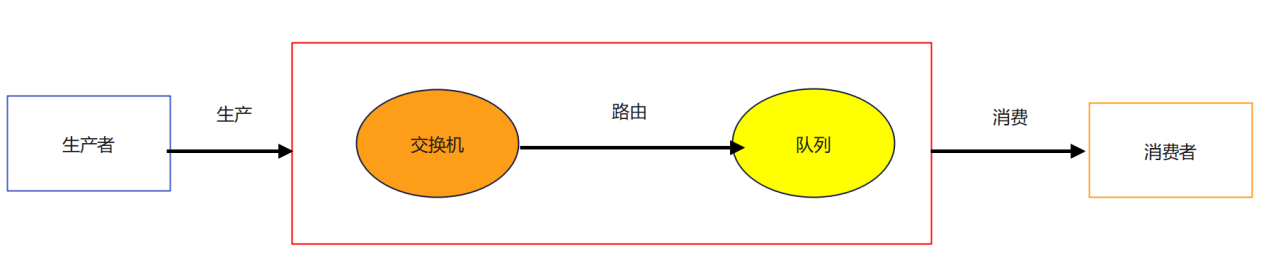
## 3.4消息驱动

前一阶段消息的处理结果放到后一阶段的消息队列

# 4.常用的MQ对比



# 5.RabbitMQ原理



# 6.RabbitMQ安装

## 6.1 Windows环境安装 Docker容器 Images：镜像 container：容器

## 6.2 安装RabbitMQ WSL:window subsystem linux

docker run -it -d --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:3.13-management

# 7.RabbitMQ五种模式

## 7.1基本消息模型/点对点

## 7.2工作队列消息模型

## 7.3发布/订阅(广播、扇形)模型

## 7.4路由模型

## 7.5主题模型

# RabbitMQ的持久化

## 8.1持久化概念

1、内存到磁盘。保证:消息确认签收之前，不能丢。

2、要做到消息持久化，必须：交换机、队列、消息都要持久化。

3、持久化的交换机可以绑定没有持久化的队列；持久化的队列里可以放非持久化消息

## 8.2非持久化概念

能持久化，但是一般不做。内存快耗尽的时候进行持久化

# 9.RabbitMQ的死信队列

## 9.1概念

死信，顾名思义就是无法被消费的消息，字面意思可以这样理解，一般来说，producer将消息投递到 broker或者直接到queue里了，consumer 从 queue取出消息进行消费，但某些时候由于特定的原因导致queue中的某些消息无法被消费，这样的消息如果没有后续的处理，就变成了死信，有死信自然就有了死信队列。

应用场景:为了保证订单业务的消息数据不丢失，需要使用到RabbitMQ的死信队列机制，当消息消费发生异常时，将消息投入死信队列中。还有比如说:用户在商城下单成功并点击去支付后在指定时间未支付时自动失效。

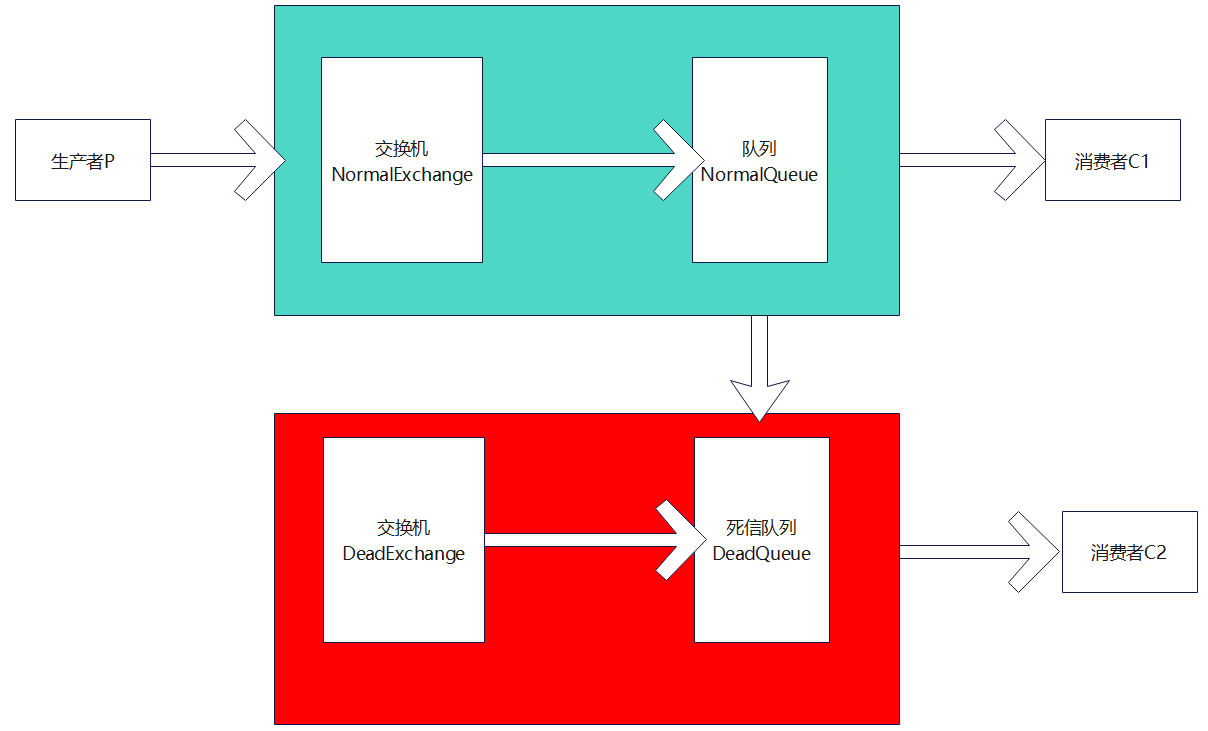
## 9.2来源

1、消息TTL过期(Time To Live 存活时长)

2、队列达到最大长度(队列满了，无法再添加数据到mq中)

3、消息被拒绝(basic.reject或 basic.nack)并且requeue=false(不放回队列)

## 9.3实现流程



# 10.RabbitMQ的延时队列

## 10.1概念

[延迟队列](https://so.csdn.net/so/search?q=%E5%BB%B6%E8%BF%9F%E9%98%9F%E5%88%97&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_35511685/article/details/_blank)存储的对象是对应的延迟消息，所谓“延迟消息”是指当消息被发送以后，并不想让消费者立刻拿到消息，而是等待特定时间后，消费者才能拿到这个消息进行消费。

## 10.2应用场景

1、订单处理：在电商网站中，订单处理是一个常见的业务流程。如果订单需要立即处理，可以使用RabbitMQ的延时队列来实现延迟处理。例如，可以将订单发送到一个延时队列中，并设置一个延迟时间(例如30分钟),然后在延迟时间到达后，将订单从队列中取出并进行处理。

2、消息推送：在移动应用或Web应用程序中，可以使用RabbitMQ的延时队列来实现消息推送。例如，可以将用户订阅的消息发送到一个延时队列中，并设置一个延迟时间(例如1小时),然后在延迟时间到达后，将消息从队列中取出并推送给用户。

3、定时任务：在分布式系统中，可以使用RabbitMQ的延时队列来实现定时任务。例如，可以将需要定期执行的任务发送到一个延时队列中，并设置一个延迟时间(例如每天),然后在延迟时间到达后，将任务从队列中取出并执行。

4、数据备份：在数据库中，可以使用RabbitMQ的延时队列来实现数据备份。例如，可以将需要备份的数据发送到一个延时队列中，并设置一个延迟时间(例如每天),然后在延迟时间到达后，将数据从队列中取出并进行备份。

5、优惠券发放：您可以设置一个延时队列，将优惠券发放任务添加到队列中，设置一定的延时时间，以保证优惠券在特定时间后才能被消费。

## 10.3 安装rabbitmq延时插件

第一步:将插件文件复制到docker容器

docker cp E:\rabbitmq\_delayed\_message\_exchange-3.13.0.ez d16d6c5d7780:/opt/rabbitmq/plugins/rabbitmq\_delayed\_message\_exchange-3.13.0.ez

第二步:进入docker容器

docker exec -it rabbitmq bash

第三步:启用插件

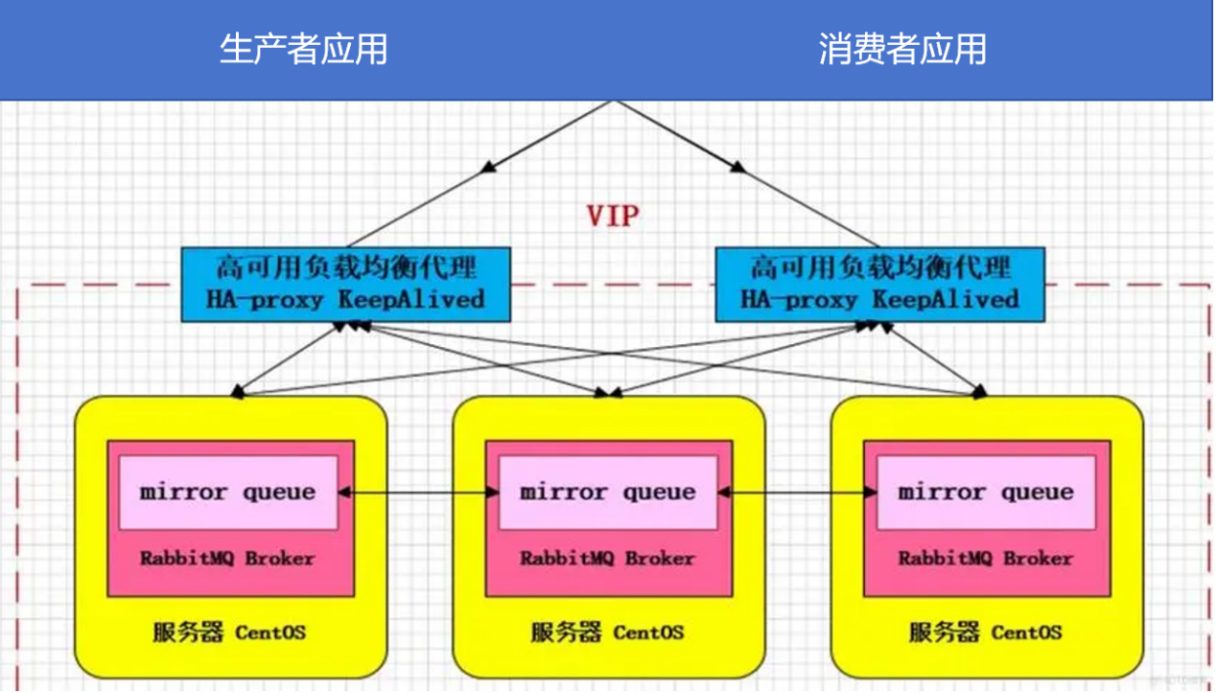
rabbitmq-plugins enable rabbitmq\_delayed\_message\_exchange

# 11.RabbitMQ集群

## 11.1为什么用集群

解决高并发，达到高可用

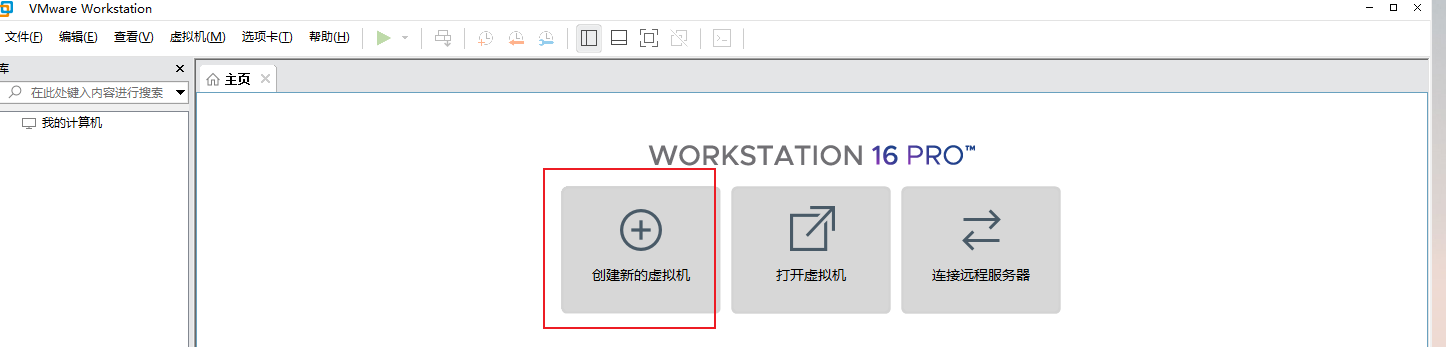
## 11.2集群部署图



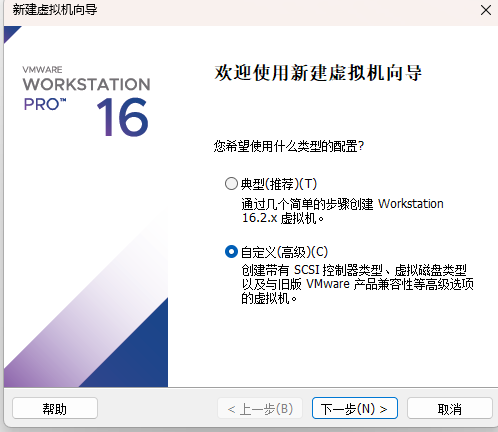
## 11.3安装虚拟机软件

## 11.4虚拟机安装Linux

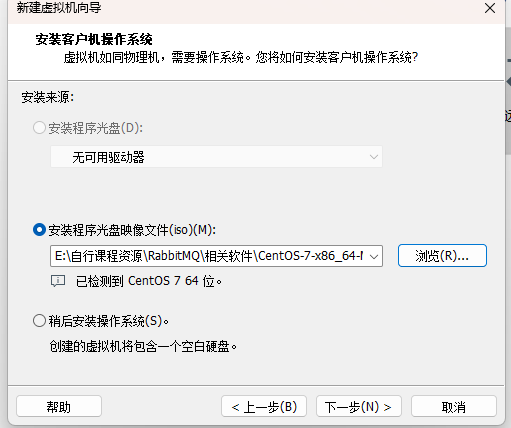
【1】主页-创建新虚拟机



【2】选择 自定义（高级）



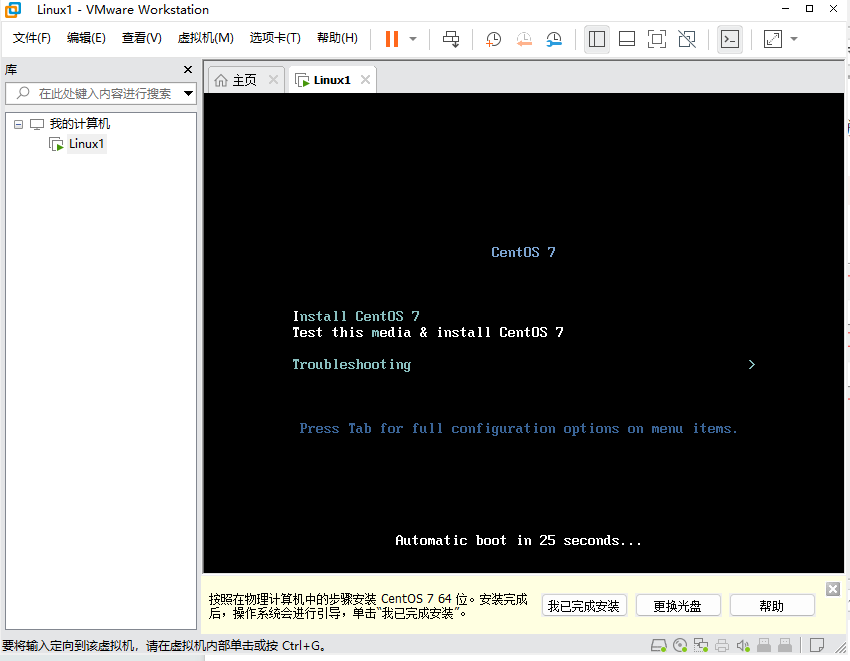
【3】一直下一步，直到如下界面。选择 安装程序光盘映像文件(iso)(M)



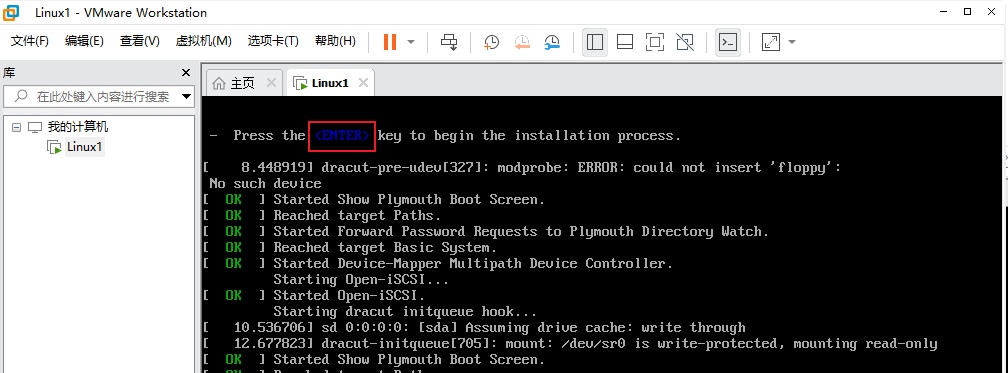
【4】随意设置虚拟机名称。虚拟机位置最好不要放到windows系统盘



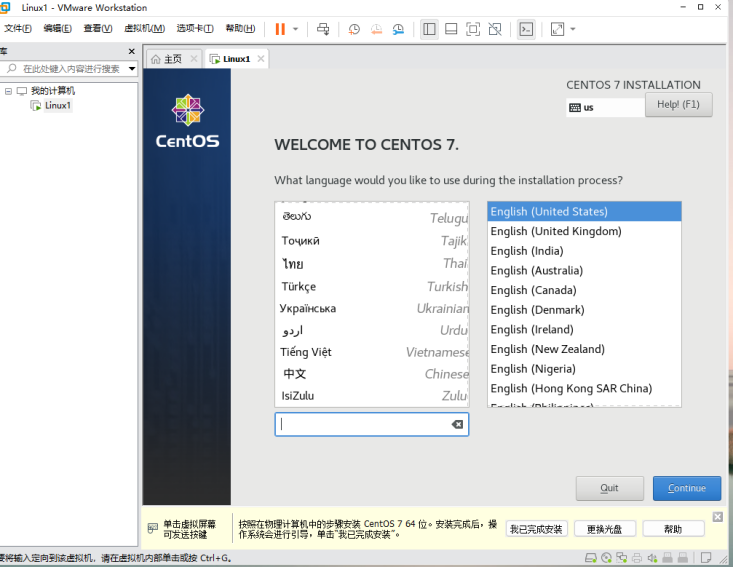
【5】一直下一步到如下界面(中间遇到弹框点确定或是)，等着。



【6】看到下面这个界面按Enter，否则等启动安装，浪费时间



【7】选择语言，中文拉到最后



【8】必须点击安装位置，否则无法点击开始安装



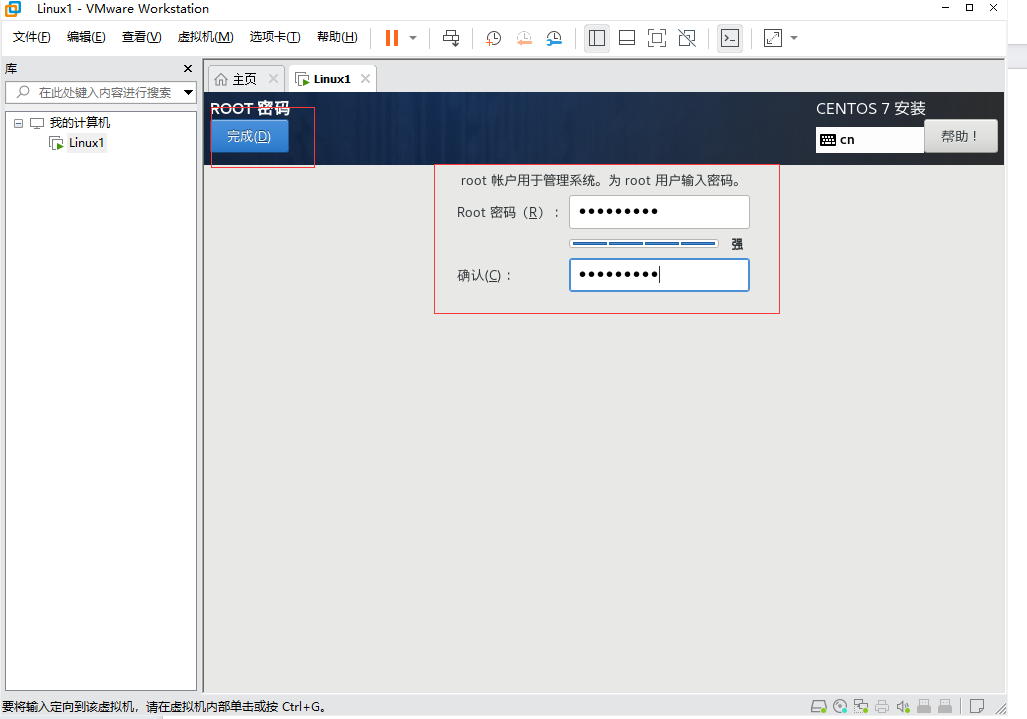




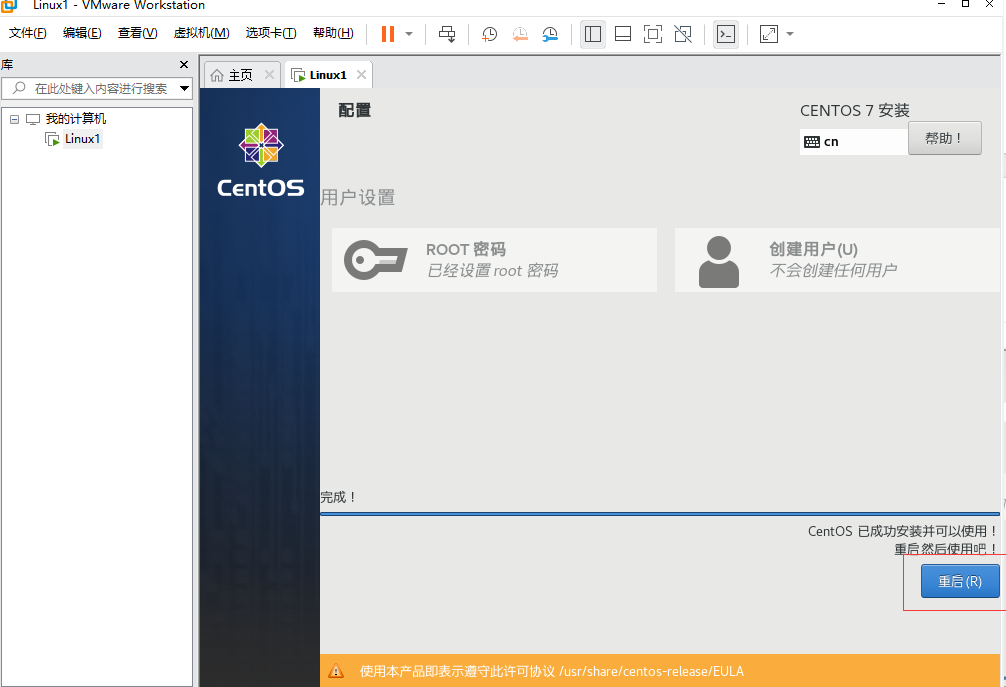
【9】点击进去 ROOT密码，为linux默认账号root设置密码



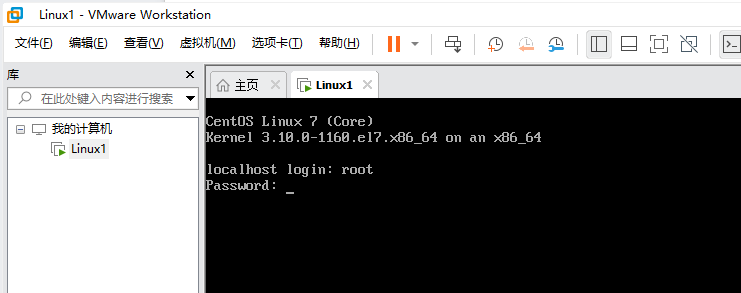
【10】设置的ROOT密码不要忘了，点击完成等着安装



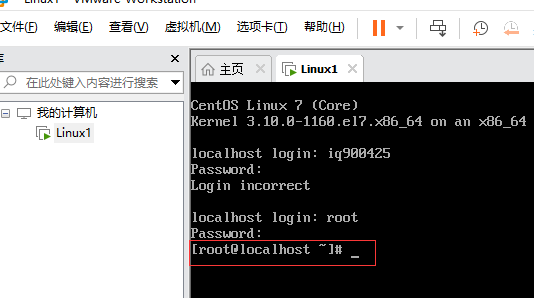
【11】点击重启



【12】账号root，输入密码(输入的密码不显示，输入行) 。每项输入了，按Enter。不能输入命令时，鼠标在黑色区域点击一下。

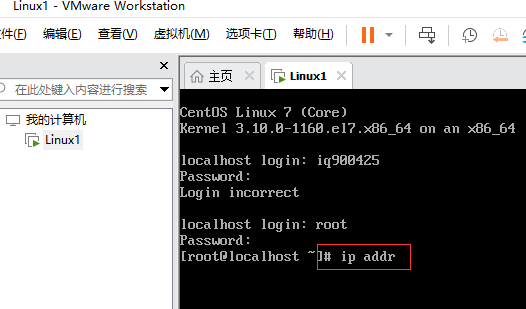


【13】如下，出现红色部分，则登入系统成功

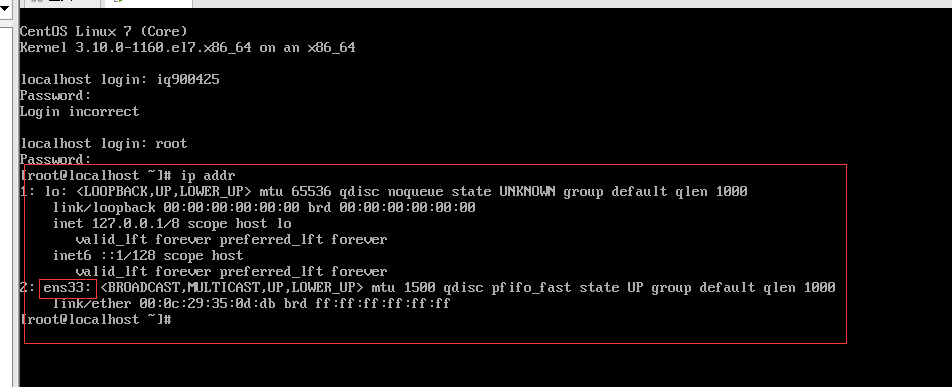


【14】配置网络 输入如图 (输入命令，按Enter，以后相同，不再说明)

命令:ip addr



【15】以下情况说明网络没通，需要配置。ens33为网卡编号

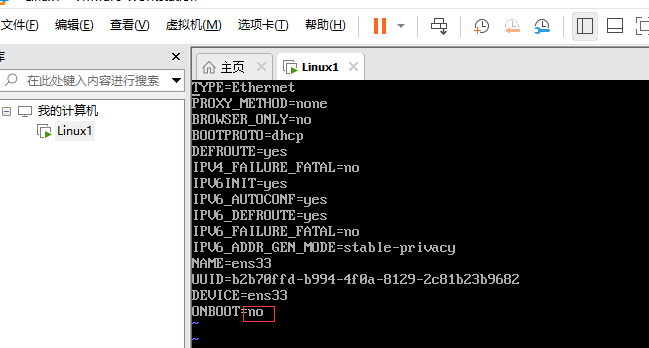


【16】以下情况说明网络没通，需要配置。ens33为网卡编号

以下vi命令表示编辑文件 ，该命令很常用哦。虚拟机可能复制粘贴不太好用，一个一个输入也得把命令敲进去。网络通了，我们不用虚拟机远程操作了。

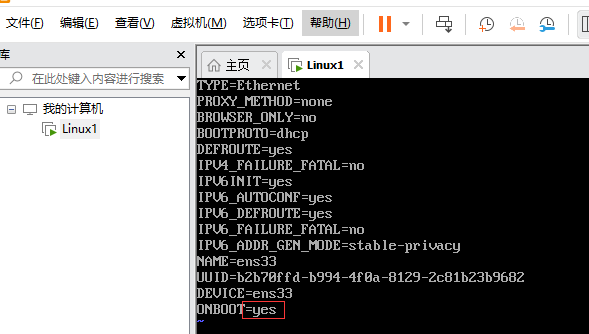
命令:vi /etc/sysconfig/network-scripts/ifcfg-ens33

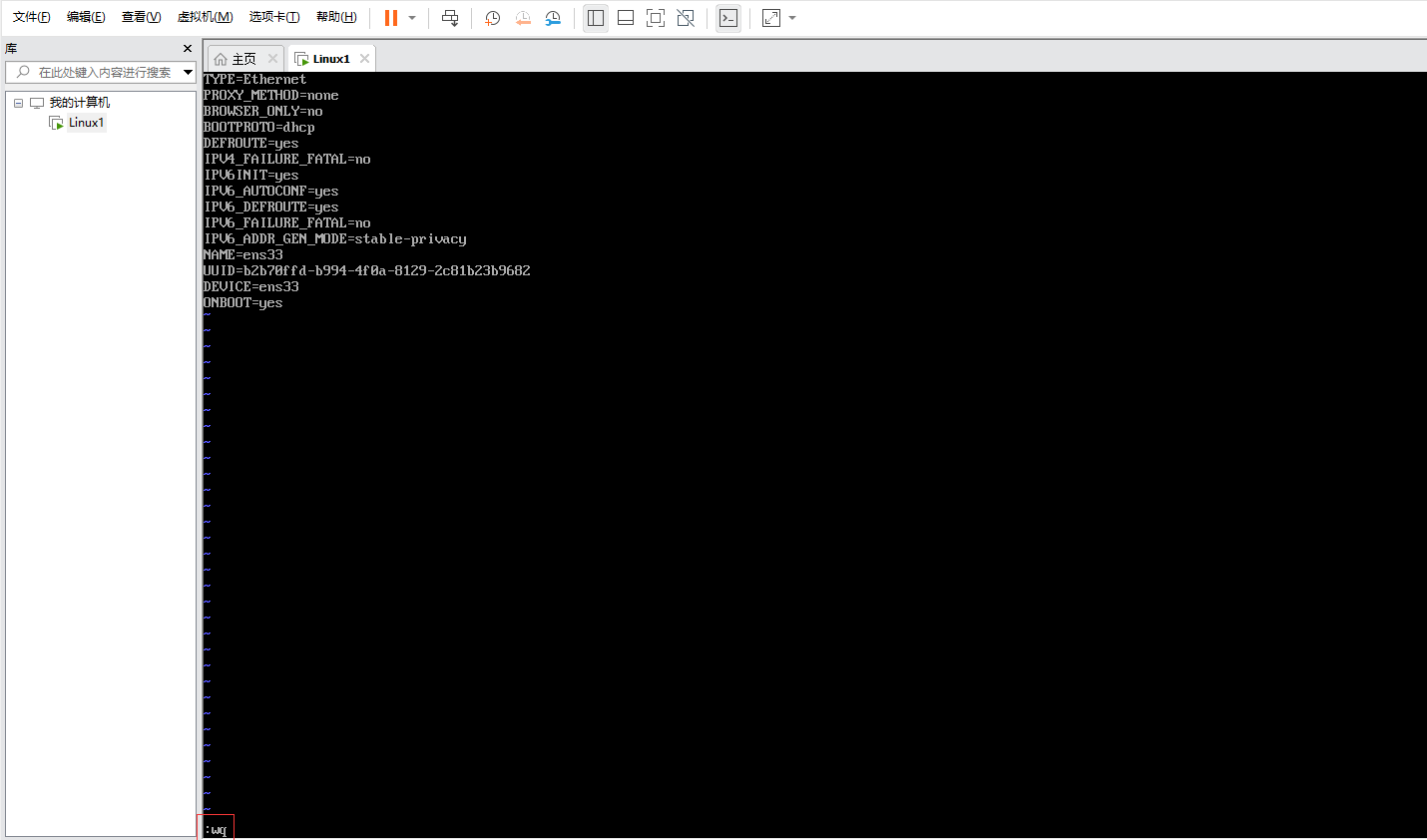
【17】进入到下面界面后，点击字母i，进入编辑状态。 no改成yes



【18】改成yes后，按Esc键，然后在输入:wq 看第二张图左下方。再enter

解释：wq表示保存写入的信息并退出(write quit)

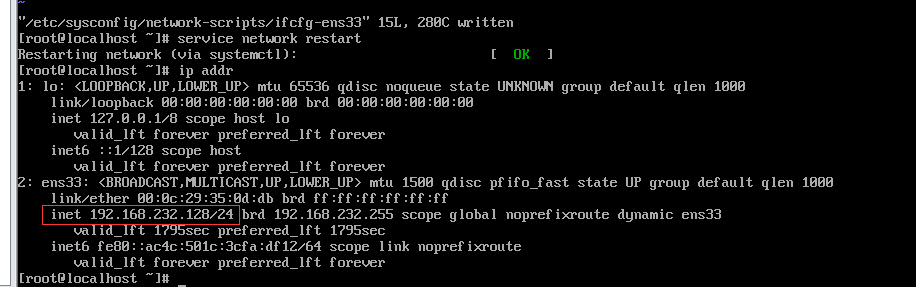




【19】重启网络

命令:service network restart

【20】确认网络是否通了 在输入ip addr。以下结果表示网通了，ip为192.168.232.128



## 11.5打开远程访问软件

## 11.6 单个节点安装RabbitMQ

### 11.6.1设置机器别名hostname和修改hosts文件

(1)hostnamectl set-hostname 机器别名 --static

(2)vi /etc/hosts

例子:

192.168.3.215 rabbit1

192.168.3.216 rabbit2

192.168.3.217 rabbit3

### **11.6.2安装Erlang**

(1)curl -s https://packagecloud.io/install/repositories/rabbitmq/erlang/script.rpm.sh | sudo bash

(2)yum install erlang

(3)erl

### **11.6.3安装rabbitmq**

(1)导入两个key

rpm --import https://packagecloud.io/rabbitmq/rabbitmq-server/gpgkey

rpm --import <https://packagecloud.io/gpg.key>

(2)

curl -s https://packagecloud.io/install/repositories/rabbitmq/rabbitmq-server/script.rpm.sh | sudo bash

(3)

wget <https://github.com/rabbitmq/rabbitmq-server/releases/download/v3.8.5/rabbitmq-server-3.8.5-1.el8.noarch.rpm>

重要说明:如果提示没有wget命令，先安装wget命令。 sudo yum install wget

(4)rpm --import <https://www.rabbitmq.com/rabbitmq-release-signing-key.asc>

(5)yum -y install epel-release

yum -y install socat

(6)rpm -ivh rabbitmq-server-3.8.5-1.el8.noarch.rpm

(7)启用管理平台插件，启用插件后，可以可视化管理RabbitMQ

rabbitmq-plugins enable rabbitmq\_management

(8)启动应用

systemctl start rabbitmq-server

(9)关闭防火墙

systemctl stop firewalld

systemctl disable firewalld

(10)关闭SELINUX

  getenforce

  setenforce 0

### **11.6.4添加rabbitmq用户并设置权限**

(1)创建新账户

rabbitmqctl add\_user longma longma

(2)设置新账户为管理员

rabbitmqctl set\_user\_tags longma administrator

(3)授权远程访问

rabbitmqctl set\_permissions -p / longma ".\*" ".\*" ".\*"

(4)重启服务

systemctl restart rabbitmq-server

## 11.7配置普通集群

### **11.7.1 修改cookie**

所有rabbitMQ服务器cookie改成一样

vi /var/lib/rabbitmq/.erlang.cookie

提示：该cookie文件只读的，保存需要 :wq!

重启rabbitmq服务器，先停再启

sudo service rabbitmq-server stop

sudo service rabbitmq-server start

提示：执行后，记得用命令添加新账号并设置为管理员。命令如下：

(1)创建新账户

rabbitmqctl add\_user longma longma

(2)设置新账户为管理员

rabbitmqctl set\_user\_tags longma administrator

(3)授权远程访问

rabbitmqctl set\_permissions -p / longma ".\*" ".\*" ".\*"

### **11.7.2 将集群的每个节点的hosts设置**

如何设置，查看11.6.1

### **11.7.3集群搭建**

该实例将rabbit2、rabbit3加入到rabbit1中，因此rabbit2、rabbit3都要完成以下配置。

(1)停止服务

rabbitmqctl stop\_app

(2)重置状态

rabbitmqctl reset

(3)节点加入

rabbitmqctl join\_cluster --ram rabbit@rabbit1

(4)启动服务

rabbitmqctl start\_app

第3步说明:join\_cluster命令有一个可选的参数--ram，该参数代表新加入的节点是内存节点，默认是磁盘节点。如果是内存节点，则所有的队列、交换器、绑定关系、用户、访问权限和vhost的元数据都将存储在内存中，如果是磁盘节点，则存储在磁盘中。内存节点可以有更高的性能，但其重启后所有配置信息都会丢失，因此RabbitMQ 要求在集群中至少有一个磁盘节点，其他节点可以是内存节点。当内存节点离开集群时，它可以将变更通知到至少一个磁盘节点；然后在其重启时，再连接到磁盘节点上获取元数据信息。除非是将 RabbitMQ用于RPC 这种需要超低延迟的场景，否则在大多数情况下，RabbitMQ 的性能都是够用的，可以采用默认的磁盘节点的形式。另外，如果节点以磁盘节点的形式加入，则需要先使用reset命令进行重置，然后才能加入现有群集，重置节点会删除该节点上存在的所有的历史资源和数据。采用内存节点的形式加入时可以略过reset这一步，因为内存上的数据本身就不是持久化的。

## 11.8配置镜像队列

1、如果为所有队列开启镜像配置，其语法如下：

rabbitmqctl set\_policy ha-all "^" '{"ha-mode":"all"}'

2、在上面指定了ha-mode 的值为 all ，代表消息会被同步到所有节点的相同队列中。这里我们之所以这样配置，因为我们本身只有三个节点，因此复制操作的性能开销比较小。如果集群有很多节点，那么此时复制的性能开销就比较大，此时需要选择合适的复制系数。通常可以遵循过半写原则，即对于一个节点数为 n 的集群，只需要同步到 n/2+1 个节点上即可。此时需要同时修改镜像策略为 exactly，并指定复制系数 ha-params，示例命令如下：

rabbitmqctl set\_policy ha-two "^" '{"ha-mode":"exactly","ha-params":2,"ha-sync-mode":"automatic"}'

3、除此之外，RabbitMQ 还支持使用正则表达式来过滤需要进行镜像操作的队列，示例如下：

rabbitmqctl set\_policy ha-all "^ha\." '{"ha-mode":"all"}'

## 11.9节点下线

以上介绍的集群搭建的过程就是服务扩容的过程，如果想要进行服务缩容，即想要把某个节点剔除集群，有两种可选方式：

第一种：可以先使用rabbitmqctl stop停止该节点上的服务，然后在其他任意一个节点上执行 forget\_cluster\_node 命令。这里以剔除rabbit2为例，此时可以在 rabbit1或rabbit3上执行下面的命令：

rabbitmqctl forget\_cluster\_node rabbit@rabbit2

第二种方式：先使用 rabbitmqctl stop 停止该节点上的服务，然后再执行 rabbitmqctl reset 这会清空该节点上所有历史数据，并主动通知集群中其它节点它将要离开集群。

## 11.10 集群的关闭与重启

没有一个直接的命令可以关闭整个集群，需要逐一进行关闭。但是需要保证在重启时，最后关闭的节点最先被启动。如果第一个启动的不是最后关闭的节点，那么这个节点会等待最后关闭的那个节点启动，默认进行 10 次连接尝试，超时时间为 30 秒，如果依然没有等到，则该节点启动失败。

这带来的一个问题是，假设在一个三节点的集群当中，关闭的顺序为 rabbit1，rabbit2，rabbit。 如果rabbit1因为故障暂时没法恢复，此时rabbit2 和rabbit3 就无法启动。想要解决这个问题，可以先将rabbit1 节点进行剔除，命令如下：

rabbitmqctl forget\_cluster\_node rabbit@rabbit1 -offline

此时需要加上 -offline 参数，它允许节点在自身没有启动的情况下将其他节点剔除。

## 11.11 修改节点数据的类型

1、rabbitmqctl stop\_app

2、rabbitmqctl reset # 如果是磁盘节点需要重置内容

3、rabbitmqctl join\_cluster --ram rabbit@rabbit1

4、rabbitmqctl start\_app

## 11.12负载均衡(haproxy)

首先配置负载均衡和rabbitmq的每个节点网络都必须打通。通过配置hostname和设置hosts文件。[参考11.6.1](#_11.6.1设置机器别名hostname和修改hosts文件)

### **11.12.1下载**

HAProxy 官方下载地址为：[https://www.haproxy.org/#down](https://www.haproxy.org/" \l "down" \t "E:/RabitMQ/B0144/%E8%AF%BE%E4%BB%B6/_blank)，如果这个网站无法访问，也可以从[https://src.fedoraproject.org/repo/pkgs/haproxy/](https://src.fedoraproject.org/repo/pkgs/haproxy/" \t "E:/RabitMQ/B0144/%E8%AF%BE%E4%BB%B6/_blank)上进行下载。我是从第二个地址下载的是 2.3.10 的版本，直接放在root目录。

### **11.12.2解压**

tar xf haproxy-2.3.10.tar.gz

### **11.12.3 编译**

执行下面的编译命令：

cd haproxy-2.3.10 进入解压后的目录

yum install gcc-c++ -y 该命令为安装c++依赖

make TARGET=linux-glibc  PREFIX=/usr/app/haproxy-2.3.10

make install PREFIX=/usr/app/haproxy-2.3.10

### **11.12.4配置环境变量**

(1)编辑配置环境变量文件

vi /etc/profile

(2)文件最后加上如下两行，保存退出

export HAPROXY\_HOME=/usr/app/haproxy-2.3.10

export PATH=$PATH:$HAPROXY\_HOME/sbin

(3)使得配置的环境变量立即生效：

source /etc/profile

### **11.12.5 负载均衡配置**

新建配置文件

mkdir /etc/haproxy 创建文件夹haproxy

touch /etc/haproxy/haproxy.cfg 创建haproxy.cfg文件

vi /etc/haproxy/haproxy.cfg 编辑haproxy.cfg文件。编辑内容如下附件。



### **11.12.6 启动服务**

搭建完成使用以下命令启动服务：

haproxy -f /etc/haproxy/haproxy.cfg

关闭防火墙

systemctl stop firewalld

systemctl disable firewalld

## 11.13监控心跳(keepalived)

我这里把haproxy1当做主(MASTER)、把haproxy2当做备份(BACKUP)

### **11.13.1 安装Keepalived**

yum install -y keepalived

### **11.13.2 配置 HAProxy 检查**

先在一台haproxy上 /etc/keepalived/keepalived.conf 配置文件进行修改，另外一台一样(只需修改state和priority )。

vi /etc/keepalived/keepalived.conf 编辑内容如下附件:



### **11.13.3 监控脚本**

touch /etc/keepalived/haproxy\_check.sh 创建shell文件

vi /etc/keepalived/haproxy\_check.sh 编辑shell文件。编辑内容如下附件:



### **11.13.3启动服务**

chmod +x /etc/keepalived/haproxy\_check.sh 给shell文件设置执行权限

分别在haproxy1 和 haproxy2上启动 KeepAlived 服务，命令如下：

systemctl start keepalived

### **11.13.4 验证故障转移**

这里我们验证一下故障转移，因为按照我们上面的检测脚本，如果 haproxy1 已经停止且无法重启时，KeepAlived服务就会停止，这里我们直接使用以下命令停止 Keepalived 服务：systemctl stop keepalived

此时再次使用ip addr分别查看，可以发现haproxy1上的 VIP 已经漂移到haproxy2上。此时对外服务的 VIP 依然可用，代表已经成功地进行了故障转移。至此集群已经搭建成功，任何需要发送或者接受消息的客户端服务只需要连接到该 VIP 即可，示例如下：

ConnectionFactory factory = new ConnectionFactory();

factory.setHost("192.168.232.200");