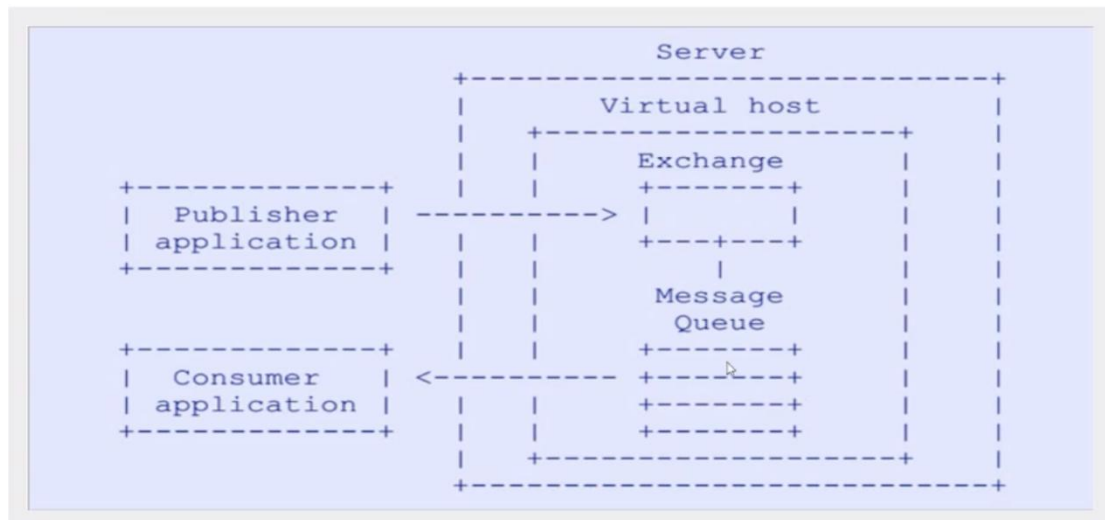


## 1、什么是 AMQP?

### 概叙

AMQP 全称: Advanced Message Queuing Protocol(高级消息队列协议)。

是具有现代特性得二进制协议, 是一个提供统一消息服务的应用层标准高级消息队列协议, 是应用协议的一个开发标准, 为面向消息的中间件设计。



## 2、核心概念

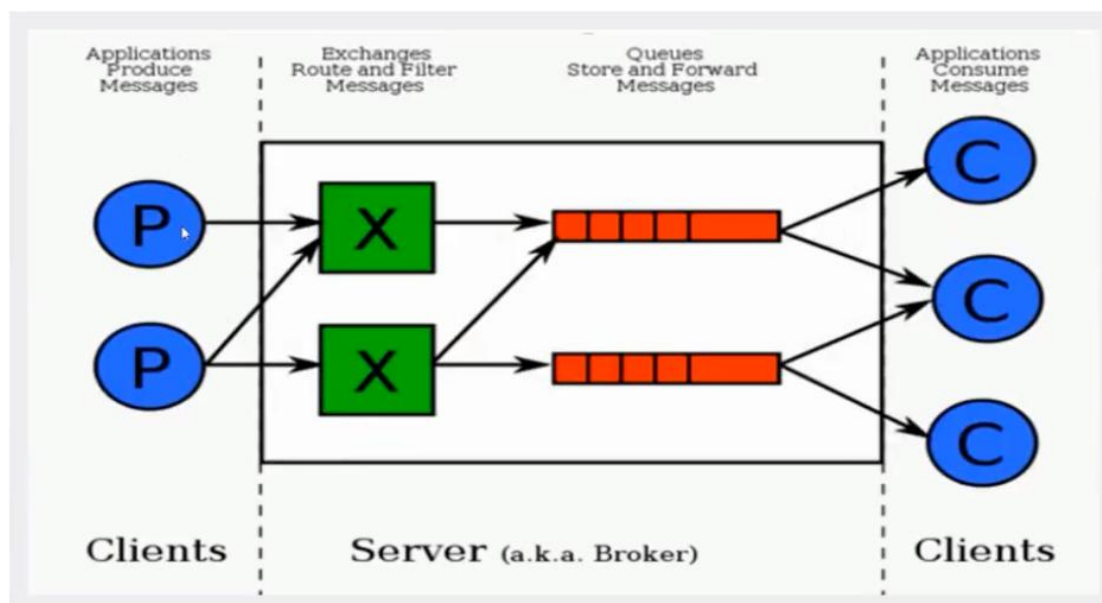
- 1、**Server:** 又称 Broker, 接受客户端的连接, 实现 AMQP 实体服务。
- 2、**Connection:** 连接, 应用程序于 Broker 的网络连接 TCP/IP 三次握手和四次挥手。
  - **重点:** 他不是开连接去执行, 而是开线程去执行, 建立连接是建立长连接然后开启多个线程去执行任务
- 3、**Channel:** 网络信道, 几乎所有的操作都在 channel 中进行, channel 是提供消息读写的通道, 客户端可以建立对各 channel, 每个 channel 代表的一个会话任务。
- 4、**Message:** 消息, 服务与应用之间传送的数据, 由 properties 和 body 组成, properties 是对消息进行修饰, 比如说消息的优先级, 延迟等高级特性, body 则是消息主体的内容。
- 5、**Virtual Host:** 虚拟地址, 用于进行逻辑隔离, 最上层的消息路由, 一个虚拟主机可以有若干个 exchange 和 queue。同一个虚拟主机不能有相同名字的 exchange 和 queue。
  - **实际场景:** 用于隔离不同业务场景, 比如是用户服务、订单服务、商品服务等等。
- 6、**Exchange:** 交换机只做消息中转, 根据路由键发送到绑定的队列。Direct ActiveMq Kafka。
- 7、**Bindings:** exchange 与 queue 之间的虚拟连接 bindings 保护多个 routing key。
- 8、**Routing key:** 是一个路由规则, 虚拟机用他来确定如何路由一个特定消息。
- 9、**Queue 队列:** 也称为 message queue(消息队列), 保存消息并将他们转发给消费者。

### 题外话:

AMQP 也是使用的 TCP 连接(因为 TCP 是安全性的), 但是操作系统对 TCP 连接数有限制, Linux 下 tcp 连接数最多达到 6553 几。

**RabbitMQ:** 是使用的长连接技术, --- 使用线程去执行(因为线程可以做到**即开即关**)

### 3、RabbitMQ 整体架构图



### 4、为什么使用 RabbitMQ?

#### 4.1、分析

RabbitMQ 是一个开源的消息队列服务器，用来通过普通协议在完全不同的应用之间进行数据共享，RabbitMQ 是使用 Erlang 语言来编写的，并且 RabbitMQ 是基于 AMQP 协议的跨平台开源消息中间件。

消息中间件是分布式系统中重要的组件，主要解决应用：**解耦、异步消息、流量削峰、高可用、可伸缩和最终一致性架构**，目前使用较多的消息队列有：ActiveMQ、RabbitMQ、ZeroMQ、Kafka、MeatMQ、RocketMQ 等等。

#### 4.2、什么情况下使用 RabbitMQ?

1. 在写多读少的情况使用。(读多写少，用缓存、写多读少。用队列)。
2. 解耦，系统 A 在代码中直接调用系统 B 和系统 C 的代码，如果未来系统 D 需要接入，系统 A 还需要修改代码，过于麻烦。
3. 异步，将消息写入队列。非必要的业务逻辑采用异步的方式进行，加快响应速度。
4. 削峰，并发量大的时候，所有的请求都丢给数据库，造成数据库处理异常。
5. 达到数据得最终得一致性

#### 4.3、什么是串行，并行?

// 注入线程池 方便异步处理使用

```
ExecutorService executorService = Executors.newCachedThreadPool();
```

\* **TODO: 串行处理 伪代码案例**

\* 这就是《串行》处理，我们平时写的 Java 代码其实大部分都是串行

\* 也就是 从上 往下 执行下去，但是一般都会使用事务 @Transactional 提交

\* 假如说：下单成功，但是发送信息失败。因为做了事务导致全部回滚了也就错失了一个订单

\* 这样子就得不偿失了 这些在我们一些大电商平台 某东 某宝 某多 是万万不能存在。

```

@Transactional
public void makeOrder(long userId, long productId, int buyNumber) {
    // 创建订单
    orderService.save(userId, productId, buyNumber);
    // 扣除库存
    skuService.subStock(productId, buyNumber);

    // 发送消息
    messageService.send();
    // 发送短信
    smsService.send();
}

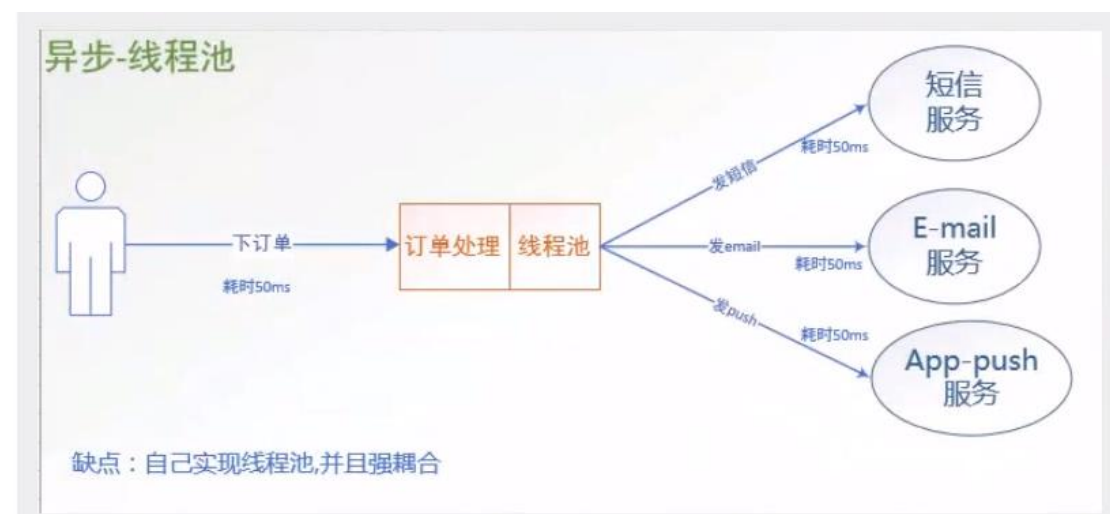
```

### 异步处理业务逻辑

```

/**
 * TODO: 异步处理 伪代码案例
 * 因此 我们诞生出其他一种处理方案，也就是使用线程池异步处理得方式去处理
 * 这里但凡是主线程 创建订单成功了，都会返回，其他子线程不会干扰主线程业务
 */
@Transactional
public void makeOrder(long userId, long productId, int buyNumber) {
    // 创建订单
    orderService.save(userId, productId, buyNumber);
    executorService.submit(new Runnable() {
        @Override
        public void run() {
            // 扣除库存 这里图方便 省略开启多个线程 submit();了
            skuService.subStock(productId, buyNumber);
            // 发送消息
            messageService.send();
        }
    });
}

```

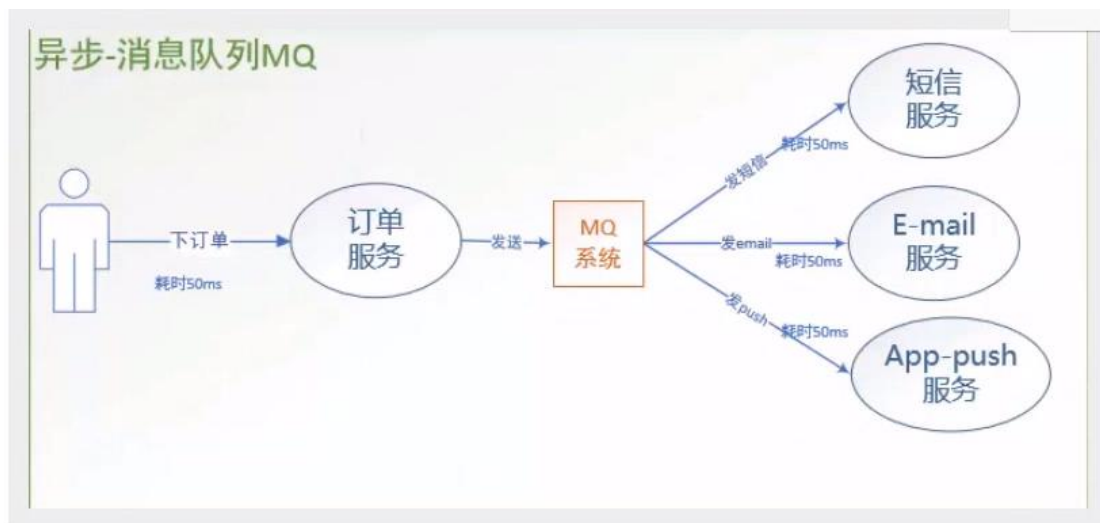


#### 4.4、消息队列

从上面那个案例分析得出：利用多线程线程池也可以达到，异步、削峰等操作，大大提高了系统得效率。但是为什么还需要消息队列嘞？

- 1、由于异步处理会开启大量得线程，而我们又又在同一个 JVM 里面处理这些逻辑，会导致我们 CPU 100%暴增从而产生很多问题。
- 2、虽然可以实现效率问题，但是其实代码得耦合度还是非常之高，由此 demo 可以看出我们如果后续还要增加更多操作得话，又得开启无数个线程去做。

这时候就要使用我们得 RabbitMQ 了，我能将线程出来丢出去独立出来，单独使用 MQ 去处理，这样就大大减少了我们系统中 CPU 瓶颈得问题,减少了资源开销,解开了代码得耦合度。



#### 5、Linux 安装 docker.

博客：<https://mr-zhou.blog.csdn.net/article/details/108357653>

环境：Centos7.X 以上版本+互联网

检查 Centos 版本：

```
[root@org.xing /]# uname -r
```

输出：3.10.0-514.26.2.el7.x86\_64

##### 1、更新 yum 包到最新

```
[root@org.xing /]# yum ypdata
```

##### 2、安装需要软件包 yum-utils 提供 yum-config-manager 功能,另外两个是 devicemapper 驱动

```
[root@org.xing /]# yum install -y yum-utils device-mapper-persistent-data lvm2
```

##### 3、设置 yum 源为阿里云

```
[root@org.xing/]#
```

```
yum-config-manager --add-repo
```

```
http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
```

##### 4、安装 docker

```
[root@org.xing/]# yum install docker-ce -y
```

##### 5、检查版本

```
[root@org.xing/]# docker -v
```

```
[root@i2bp13y6uttlis3dwqna6c12 /]# docker -v
Docker version 19.03.13, build 4484c46d9d
```

## 6、配置阿里云镜像加速服务

注意：建议去阿里云网站复制 <https://cr.console.aliyun.com/cn-hangzhou/instances/mirrors>

## 7、查看是否配置成功

```
[root@org.xing/]# cat etc/docker/daemon.json
```

```
{
  "registry-mirrors": ["https://t6341mld.mirror.aliyuncs.com"]
}
```

## 6、Linux Docker 卸载

博客：<https://www.cnblogs.com/kingsonfu/p/11582495.html>

### 1、查看 docker 运行状态

```
[root@org.xing/]# systemctl status docker
```

```
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Sat 2020-10-10 16:50:16 CST; 1 months 15 days ago
     Docs: https://docs.docker.com
   Main PID: 7326 (dockerd)
   CGroup: /system.slice/docker.service
           └─7326 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

### 2、如果运行中 则停止 docker

```
[root@org.xing/]# systemctl stop docker
```

### 3、查看 docker 安装得 docker 包

```
[root@org.xing/]# yum list installed |grep docker
```

```
docker-ce.x86_64          3:19.03.13-3.el7          @docker-ce-edge
docker-ce-cli.x86_64      1:19.03.13-3.el7          @docker-ce-edge
```

### 4、删除安装包

```
[root@org.xing/]# yum -y remove docker.x86_64
```

### 5、删除容器

```
[root@org.xing/]# rm -rf /var/lib/docker
```

## 7、Docker 安装 RabbitMQ

### 1、查询 rabbitMQ 镜像

```
[root@org.xing/]# docker search rabbitmq:management
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
macintoshplus/rabbitmq-management	Based on rabbitmq:management whit python and...	6		[OK]
xiaochunping/rabbitmq	xiaochunping/rabbitmq:management 2018-06-30	4		
transmits/rabbitmq-sharded	Fork of rabbitmq:management with sharded_exc...	0		
yunyan2140/rabbitmq	docker pull rabbitmq:management	0		

### 2、拉取镜像

```
[root@org.xing/]# docker pull rabbitmq:management
```

```
management: Pulling from library/rabbitmq
171857c49d8f: Downloading [=====>] 18.64MB/26.7MB
419640447d26: Download complete
61e52f862619: Download complete
856781f94405: Download complete
992c175617ad: Downloading [=====>] 7.781MB/39.89MB
3aa8ecec680f: Download complete
af535143b306: Downloading [=====>] 5.565MB/15.67MB
43d585d75063: Waiting
1b39cd7f935c: Waiting
a54ee918187e: Waiting
61b1c2eaeaa4: Waiting
5244adde3eb1: Waiting
```

### 3、查看镜像列表

```
[root@org.xing/]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
rabbitmq	management	87505dc99f21	4 days ago	203MB
duck-provider	1.0	5c6c3d3ac886	5 weeks ago	707MB
java	8	d23bdf5b1b1b	3 years ago	643MB

#### 4、启动

启动方式一：

```
docker run -d -p 5672:5672 -p 15672:15672 --name rabbitmq rabbitmq:management
```

```
-d 后台运行
-d 后台运行
-p 隐射端口
--name 指定 rabbitMQ 名称
```

启动方式二：

```
docker run -d -p 15672:15672 -p 5672:5672 -e RABBITMQ_DEFAULT_USER=xing
-e RABBITMQ_DEFAULT_PASS=xing --name rabbitmq --hostname=rabbitmqhostone
rabbitmq:management
```

```
-d 后台运行
-p 隐射端口
--name 指定 rabbitMQ 名称
RABBITMQ_DEFAULT_USER 指定用户账号
RABBITMQ_DEFAULT_PASS 指定账号密码
```

5、访问：<http://localhost:15672> 【记得开放端口】

The screenshot shows the RabbitMQ management UI. At the top, it says 'RabbitMQ 3.8.9 Erlang 23.1.4'. The 'Overview' tab is active, displaying various metrics like 'Queued messages', 'Message rates', and 'Global counts'. Below these, there's a 'Nodes' section showing the status of the 'rabbit@rabbitmqhostone' node. At the bottom, the 'Ports and contexts' section contains a table of listening ports, which is highlighted with a red box:

Protocol	Bound to	Port
amqp	::	5672
clustering	::	25672
http	::	15672
http/prometheus	::	15692

#### 6、上图红框端口说明

AMQP	5672	协议端口
Clustering	25672	后期集群使用端口
http	15672	也就是我们 web 端口访问 RabbitMQ 控制台需要端口号
http/prometheus	15692	RabbitMQ 插件端口号



## 8、Docker 下 RabbitMQ 安装延迟消息队列插件。

----- 说来踩了个大坑，妈蛋延迟队列竟然要额外安装插件的，研究了半天报错最后打断点调试源码才发现延迟消息交换机创建失败。

### 1、下载插件


1.1、地址：<https://www.rabbitmq.com/community-plugins.html>

#### Routing

<b>rabbitmq_lvc_exchange</b> The last value exchange acts like a direct exchange (binding keys are compared for equality with routing keys); but it also keeps track of the last value that was published with each routing key, and when a queue is bound, it automatically enqueues the last value for the binding key. <ul style="list-style-type: none"><li>• Download for <a href="#">3.7.x and 3.8.x</a></li><li>• Maintainer: <b>Team RabbitMQ</b></li><li>• GitHub: <a href="#">rabbitmq/rabbitmq-lvc-exchange</a></li></ul>
<b>rabbitmq_rtopic_exchange</b> Adds a reverse topic exchange which lets you provide routing patterns at publishing time, instead of at binding time. <ul style="list-style-type: none"><li>• Download for <a href="#">3.7.x and 3.8.x</a></li><li>• Author: <b>Alvaro Videla</b></li><li>• Maintainer: <b>Team RabbitMQ</b></li><li>• GitHub: <a href="#">rabbitmq/rabbitmq-rtopic-exchange</a></li></ul>
<b>rabbitmq_delayed_message_exchange</b> A plugin that adds delayed-messaging (or scheduled-messaging) to RabbitMQ. <ul style="list-style-type: none"><li>• Download for <a href="#">3.7.x and 3.8.x</a></li><li>• Author: <b>Alvaro Videla</b></li><li>• GitHub: <a href="#">rabbitmq/rabbitmq-delayed-message-exchange</a></li></ul>
<b>rabbitmq_routing_node_stamp</b> A plugin that stamps a message with the node who first received it. <ul style="list-style-type: none"><li>• Download for <a href="#">3.7.x and 3.8.x</a></li></ul>

### 1.2、跳转到 GitHub 下载选择

## rabbitmq-delayed-message-exchange v3.8.x


 lukebakken released this on 5 Oct 2019 · 34 commits to master since this release


This release has been superseded by [v3.8.9](#)


`rabbitmq-delayed-message-exchange` build that is compatible with these RabbitMQ versions:

- 3.8.x up to 3.8.4
- 3.7.x

Assets 3

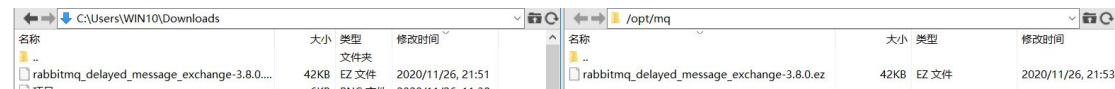
 **rabbitmq\_delayed\_message\_exchange-3.8.0.ez** 42.4 KB

 [Source code \(zip\)](#)

 [Source code \(tar.gz\)](#)

## 2、安装方式 - (听说有两种)一种是 docker file 方式。

### 2.1、上传插件到服务器



### 2.2、查看上传成功与否

```
[root@izbp13y6uttfs3dwqha6cfz mq]# ls
rabbitmq_delayed_message_exchange-3.8.0.ez
```

### 2.3、新开一个窗口，查看 docker 镜像 RabbitMQ ID

```
[root@org.xing mq]# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
da57cc0f9e73	rabbitmq:management	"docker-entrypoint.s...	30 hours ago	Up 30 hours

4505/tcp, 5671/tcp, 0.0.0.0:5672->5672/tcp, 15671/tcp, 15691-15692/tcp, 25672/tcp, 0.0.0.0:15672->15672/tcp

-- 进入 rabbitMQ 工作目录

```
[root@org.xing mq]# docker exec -it da57cc0f9e73 bash
```

```
root@rabbitmqhostone:/# ls
```

-- 进入 rabbitMQ 插件目录

```
root@rabbitmqhostone:/# cd plugins
```

```
root@rabbitmqhostone:/plugins# ls
```

### 2.4、再去之前上传窗口讲插件复制至 plugins 中

```
[root@org.xing mq]# 注意 da57cc0f9e73 是你 docker 中 rabbitmq 的镜像 ID
docker cp rabbitmq_delayed_message_exchange-3.8.0.ez da57cc0f9e73:/plugins
```

### 2.5、再回来 rabbitMQ 窗口

#### 2.5.1、查询是否存在插件

```
root@rabbitmqhostone:/plugins# ls |grep delay
rabbitmq_delayed_message_exchange-3.8.0.ez
root@rabbitmqhostone:/plugins# rabbitmq-plugins enable rabbitmq_delayed_message_exchange
Enabling plugins on node rabbit@rabbitmqhostone:
rabbitmq_delayed_message_exchange
The following plugins have been configured:
  rabbitmq_delayed_message_exchange
  rabbitmq_management
  rabbitmq_management_agent
  rabbitmq_prometheus
  rabbitmq_web_dispatch
Applying plugin configuration to rabbit@rabbitmqhostone...
The following plugins have been enabled:
  rabbitmq_delayed_message_exchange
started 1 plugins.
```

#### 2.5.2、启动插件哦

```
root@rabbitmqhostone:/plugins#
rabbitmq-plugins enable rabbitmq_delayed_message_exchange
```

#### 2.5.3、退出插件启动台

```
root@rabbitmqhostone:/plugins# exit
```

#### 2.5.4、重启:[root@izbp13y6uttfs3dwqha6cfz mq]# docker restart da57cc0f9e73



### 3、重新访问 - 出现 x-delayed-message 表示成功

The screenshot shows the RabbitMQ Management interface. At the top, it says 'RabbitMQ 3.8.9 Erlang 23.1.4'. The 'Exchanges' tab is selected. Below the navigation bar, there's a table of existing exchanges:

Virtual host	Name	Type	Features	Message rate in	Message rate out
/	(AMQP default)	direct	D		
/	amq.direct	direct	D		
/	amq.fanout	fanout	D		
/	amq.headers	headers	D		
/	amq.match	headers	D		
/	amq.rabbitmq.trace	topic	D, I		
/	amq.topic	topic	D		
/	mq.exchange.test	topic	D		

Below the table, there's a section 'Add a new exchange'. The 'Type' dropdown menu is open, showing options: direct, fanout, headers, topic, and x-delayed-message. A red arrow points to 'x-delayed-message'.

Other fields in the form include 'Virtual host' (set to /), 'Name' (empty), 'Durability' (set to direct), 'Auto delete' (checked), 'Internal' (checked), and 'Arguments' (empty). There is also an 'Alternate exchange' field.