

RabbitMQ部署指南

1.单机部署

我们在Centos7虚拟机中使用Docker来安装。

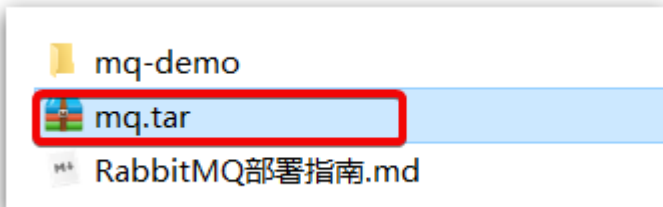
1.1.下载镜像

方式一：在线拉取

```
docker pull rabbitmq:3.8-management
```

方式二：从本地加载

在课前资料已经提供了镜像包：



上传到虚拟机中后，使用命令加载镜像即可：

```
docker load -i mq.tar
```

1.2.安装MQ

执行下面的命令来运行MQ容器：

```
docker run \
-e RABBITMQ_DEFAULT_USER=itcast \
-e RABBITMQ_DEFAULT_PASS=123321 \
-v mq-plugins:/plugins \
--name mq \
--hostname mq1 \
-p 15672:15672 \
-p 5672:5672 \
-d \
rabbitmq:3.8-management
```

2.安装DelayExchange插件

官方的安装指南地址为：<https://blog.rabbitmq.com/posts/2015/04/scheduling-messages-with-rabbitmq>

上述文档是基于linux原生安装RabbitMQ，然后安装插件。

因为我们之前是基于Docker安装RabbitMQ，所以下面我们会讲解基于Docker来安装RabbitMQ插件。

2.1. 下载插件

RabbitMQ有一个官方的插件社区，地址为：<https://www.rabbitmq.com/community-plugins.html>

其中包含各种各样的插件，包括我们要使用的DelayExchange插件：

rabbitmq_delayed_message_exchange

A plugin that adds delayed-messaging (or scheduled-messaging) to RabbitMQ.

- Download for [3.7.x and 3.8.x](#)
- Author: **Alvaro Videla**
- GitHub: [rabbitmq/rabbitmq-delayed-message-exchange](https://github.com/rabbitmq/rabbitmq-delayed-message-exchange)

大家可以去对应的GitHub页面下载3.8.9版本的插件，地址为<https://github.com/rabbitmq/rabbitmq-delayed-message-exchange/releases/tag/3.8.9>这个对应RabbitMQ的3.8.5以上版本。

课前资料也提供了下载好的插件：

assets

mq-advanced-demo

rabbitmq_delayed_message_exchange-3.8.9-0199d11c.ez

RabbitMQ部署指南.md

2.2. 上传插件

因为我们是基于Docker安装，所以需要先查看RabbitMQ的插件目录对应的数据卷。如果不是基于Docker的同学，请参考第一章部分，重新创建Docker容器。

我们之前设定的RabbitMQ的数据卷名称为mq-plugins，所以我们使用下面命令查看数据卷：

```
docker volume inspect mq-plugins
```

可以得到下面结果：

```
[root@localhost ~]# docker volume inspect mq-plugins
[
  {
    "CreatedAt": "2021-07-12T21:44:26+08:00",
    "Driver": "local",
    "Labels": null,
    "Mountpoint": "/var/lib/docker/volumes/mq-plugins/_data",
    "Name": "mq-plugins",
    "Options": null,
    "Scope": "local"
  }
]
```

接下来，将插件上传到这个目录即可：

/var/lib/docker/volumes/mq-plugins/_data/

Name	Size (KB)
rabbitmq_aws-3.8.5.ez	60
rabbitmq_consistent_hash_exchange-3.8.5.ez	34
rabbitmq_delayed_message_exchange-3.8.9-0199d11c.ez	49

2.3. 安装插件

最后就是安装了，需要进入MQ容器内部来执行安装。我的容器名为mq，所以执行下面命令：

```
docker exec -it mq bash
```

执行时，请将其中的 `-it` 后面的mq替换为你自己的容器名。

进入容器内部后，执行下面命令开启插件：

```
rabbitmq-plugins enable rabbitmq_delayed_message_exchange
```

结果如下：

```
root@mq1:/# rabbitmq-plugins enable rabbitmq_delayed_message_exchange
Enabling plugins on node rabbit@mq1:
rabbitmq_delayed_message_exchange
The following plugins have been configured:
  rabbitmq_delayed_message_exchange
  rabbitmq_management
  rabbitmq_management_agent
  rabbitmq_web_dispatch
Applying plugin configuration to rabbit@mq1...
The following plugins have been enabled:
  rabbitmq_delayed_message_exchange

started 1 plugins.
```

3. 集群部署

接下来，我们看看如何安装RabbitMQ的集群。

2.1. 集群分类

在RabbitMQ的官方文档中，讲述了两中集群的配置方式：

- 普通模式：普通模式集群不进行数据同步，每个MQ都有自己的队列、数据信息（其它元数据信息如交换机等会同步）。例如我们有2个MQ：mq1，和mq2，如果你的消息在mq1，而你连接到了mq2，那么mq2会去mq1拉取消息，然后返回给你。如果mq1宕机，消息就会丢失。
- 镜像模式：与普通模式不同，队列会在各个mq的镜像节点之间同步，因此你连接到任何一个镜像节点，均可获取到消息。而且如果一个节点宕机，并不会导致数据丢失。不过，这种方式增加了数据同步的带宽消耗。

我们先来看普通模式集群，我们的计划部署3节点的mq集群：

```
| 主机名 | 控制台端口 | amqp通信端口 | | ----- | ----- | ----- | | mq1 | 8081 ---> 15672 | 8071 ---> 5672 | | mq2 | 8082 ---> 15672 | 8072 ---> 5672 | | mq3 | 8083 ---> 15672 | 8073 ---> 5672 |
```

集群中的节点标示默认都是： `rabbit@[hostname]`，因此以上三个节点的名称分别为：

- rabbit@mq1
- rabbit@mq2
- rabbit@mq3

2.2. 获取cookie

RabbitMQ底层依赖于Erlang，而Erlang虚拟机就是一个面向分布式的语言，默认就支持集群模式。集群模式中的每个RabbitMQ 节点使用 cookie 来确定它们是否被允许相互通信。

要使两个节点能够通信，它们必须具有相同的共享秘密，称为**Erlang cookie**。cookie 只是一串最多 255 个字符的字母数字字符。

每个集群节点必须具有**相同的 cookie**。实例之间也需要它来相互通信。

我们先在之前启动的mq容器中获取一个cookie值，作为集群的cookie。执行下面的命令：

```
docker exec -it mq cat /var/lib/rabbitmq/.erlang.cookie
```

可以看到cookie值如下：

```
FXZMCVGLBIXZCDEMMVZQ
```

接下来，停止并删除当前的mq容器，我们重新搭建集群。

```
docker rm -f mq
```

```
[root@localhost ~]# docker start mq
mq
[root@localhost ~]# docker exec -it mq cat /var/lib/rabbitmq/.erlang.cookie
FXZMCVGLBIXZCDEMMVZQ[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# docker rm -f mq
mq
```

2.3.准备集群配置

在/tmp目录新建一个配置文件 rabbitmq.conf：

```
cd /tmp
# 创建文件
touch rabbitmq.conf
```

文件内容如下：

```
loopback_users.guest = false
listeners.tcp.default = 5672
cluster_formation.peer_discovery_backend = rabbit_peer_discovery_classic_config
cluster_formation.classic_config.nodes.1 = rabbit@mq1
cluster_formation.classic_config.nodes.2 = rabbit@mq2
cluster_formation.classic_config.nodes.3 = rabbit@mq3
```

再创建一个文件，记录cookie

```
cd /tmp
# 创建cookie文件
touch .erlang.cookie
# 写入cookie
echo "FXZMCVGLBIXZCDEMMVZQ" > .erlang.cookie
# 修改cookie文件的权限
chmod 600 .erlang.cookie
```

准备三个目录,mq1、mq2、mq3：

```
cd /tmp
# 创建目录
mkdir mq1 mq2 mq3
```

然后拷贝rabbitmq.conf、cookie文件到mq1、mq2、mq3:

```
# 进入/tmp
cd /tmp
# 拷贝
cp rabbitmq.conf mq1
cp rabbitmq.conf mq2
cp rabbitmq.conf mq3
cp .erlang.cookie mq1
cp .erlang.cookie mq2
cp .erlang.cookie mq3
```

2.4.启动集群

创建一个网络:

```
docker network create mq-net
```

docker volume create

运行命令

```
docker run -d --net mq-net \
-v ${PWD}/mq1/rabbitmq.conf:/etc/rabbitmq/rabbitmq.conf \
-v ${PWD}/.erlang.cookie:/var/lib/rabbitmq/.erlang.cookie \
-e RABBITMQ_DEFAULT_USER=itcast \
-e RABBITMQ_DEFAULT_PASS=123321 \
--name mq1 \
--hostname mq1 \
-p 8071:5672 \
-p 8081:15672 \
rabbitmq:3.8-management
```

```
docker run -d --net mq-net \
-v ${PWD}/mq2/rabbitmq.conf:/etc/rabbitmq/rabbitmq.conf \
-v ${PWD}/.erlang.cookie:/var/lib/rabbitmq/.erlang.cookie \
-e RABBITMQ_DEFAULT_USER=itcast \
-e RABBITMQ_DEFAULT_PASS=123321 \
--name mq2 \
--hostname mq2 \
-p 8072:5672 \
-p 8082:15672 \
rabbitmq:3.8-management
```

```
docker run -d --net mq-net \
-v ${PWD}/mq3/rabbitmq.conf:/etc/rabbitmq/rabbitmq.conf \
-v ${PWD}/.erlang.cookie:/var/lib/rabbitmq/.erlang.cookie \
```

```
-e RABBITMQ_DEFAULT_USER=itcast \  
-e RABBITMQ_DEFAULT_PASS=123321 \  
--name mq3 \  
--hostname mq3 \  
-p 8073:5672 \  
-p 8083:15672 \  
rabbitmq:3.8-management
```

2.5.测试

在mq1这个节点上添加一个队列：

Overview

Connections

Channels

Exchanges

Queues

Admin

Queues

▶ All queues (0)

... no queues ...

▼ Add a new queue

Type: Classic

Name: simple.queue

Durability: Durable

Node: rabbit@mq1

Auto delete: No

Arguments: = String

Add Message TTL ? | Auto expire ? | Max length ? | Max length bytes ? | Overflow behaviour ? | Dead letter exchange ? | Dead letter routing key ? | Single active consumer ? | Maximum priority ? | Lazy mode ? | Master locator ?

Add queue

队列名称

队列所在街道，是rabbit@mq1

如图，在mq2和mq3两个控制台也都能看到：

Queues

▼ All queues (1)

Pagination

Page 1 of 1 - Filter: ☐ Regex

Overview					Messages			Message rat
Name	Node	Type	Features	State	Ready	Unacked	Total	incoming
simple.queue	rabbit@mq1	classic	D Args	idle	0	0	0	

2.5.1.数据共享测试

点击这个队列，进入管理页面：

Queues

▼ All queues (1)

Pagination

Page 1 of 1 - Filter: ☐ Regex ?

Overview					Messages		
Name	Node	Type	Features	State	Ready	Unacked	Total
simple.queue	rabbit@mq1	classic	D Args	idle	0	0	0

然后利用控制台发送一条消息到这个队列：

Queue simple.queue

► Overview

► Consumers

► Bindings

▼ Publish message

Message will be published to the default exchange with routing key **simple.queue**, routing it to this queue.

Delivery mode: 2 - Persistent

Headers: ? = String


Properties: ? =

Payload:

Publish message

结果在mq2、mq3上都能看到这条消息：

← → ↺ ⌂ ⚠ 不安全 | 192.168.150.101:8083/#/queues

 **RabbitMQ**™ RabbitMQ 3.8.19 Erlang 24.0.3

Overview

Connections

Channels

Exchanges

Queues

Admin

Queues

► All queues (1)

Overview					Messages			Message rates		
Name	Node	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack
simple.queue	rabbit@mq1	classic	D Args	idle	1	0	1	0.00/s	0.00/s	0.00/s


2.5.2.可用性测试

我们让其中一台节点mq1宕机：

```
docker stop mq1
```

然后登录mq2或mq3的控制台，发现simple.queue也不可用了：

← → ↺ ⌂ ▲ 不安全 | 192.168.150.101:8083/#/queues

 RabbitMQ™

RabbitMQ 3.8.19

Erlang 24.0.3

OverviewConnectionsChannelsExchangesQueuesAdmin

Queues

▶ All queues (1)

Overview				Messages			Message rates			+/-	
Name	Node	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	
simple.queue	rabbit@mq1	classic	D Args	down	NaN	NaN	NaN				

说明数据并没有拷贝到mq2和mq3。