目录

[1、 安装rabbitmq 1](#_Toc522213979)

[1.1 docker 容器安装（推荐，因为简单方便） 1](#_Toc522213980)

[1.1.1查找镜像 1](#_Toc522213981)

[1.1.2 将镜像pull下来 2](#_Toc522213982)

[1.1.3列出镜像： 2](#_Toc522213983)

[1.1.4 获取该镜像的详细信息 2](#_Toc522213984)

[1.1.5创建并启动容器 2](#_Toc522213985)

[1.1.6 查看容器 2](#_Toc522213986)

[1.1.7访问*web* 管理端 3](#_Toc522213987)

[1.2下载安装包，安装（比较传统） 3](#_Toc522213988)

[2 、springboot整合rabbitmq的一个简单例子 4](#_Toc522213989)

[2.1 引入依赖 4](#_Toc522213990)

[2.2配置rabbitmq的安装地址、端口以及账户信息 4](#_Toc522213991)

[2.3队列配置 4](#_Toc522213992)

[2.4发送者 5](#_Toc522213993)

[2.5消费者 6](#_Toc522213994)

[2.6 测试（controller层） 6](#_Toc522213995)

[2.6.1打开WEB GUI 7](#_Toc522213996)

[3.RabbitMQ 使用参考 7](#_Toc522213997)

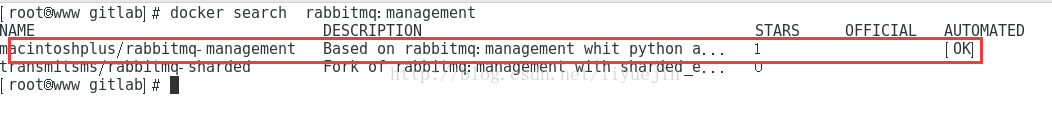
[3.1、通常的操作流程是: 7](#_Toc522213998)

# 安装rabbitmq

## 1.1 docker 容器安装（推荐，因为简单方便）

### 1.1.1查找镜像

docker search rabbitmq:management



ps:rabbitmq 后面不带management，启动rabbitmq后是无法打开管理界面的，所以我们要下载带management插件的rabbitmq.

### 1.1.2 将镜像pull下来

docker pull rabbitmq:management

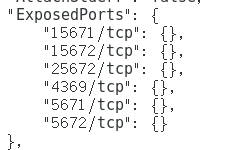
### 1.1.3列出镜像：

docker images

### 1.1.4 获取该镜像的详细信息

使用docker inspect 镜像名：tag或者id

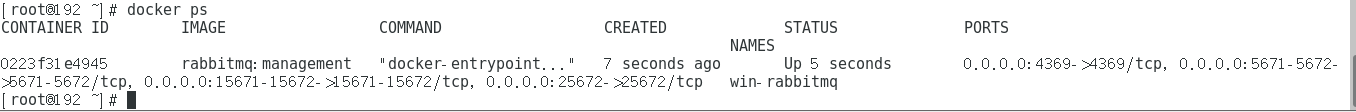
看到其默认需暴露端口



### 1.1.5创建并启动容器

docker run -d --name win-rabbitmq -p 5671:5671 -p 5672:5672 -p 4369:4369 –p 25672:25672 -p 15671:15671 -p15672:15672 rabbitmq:management

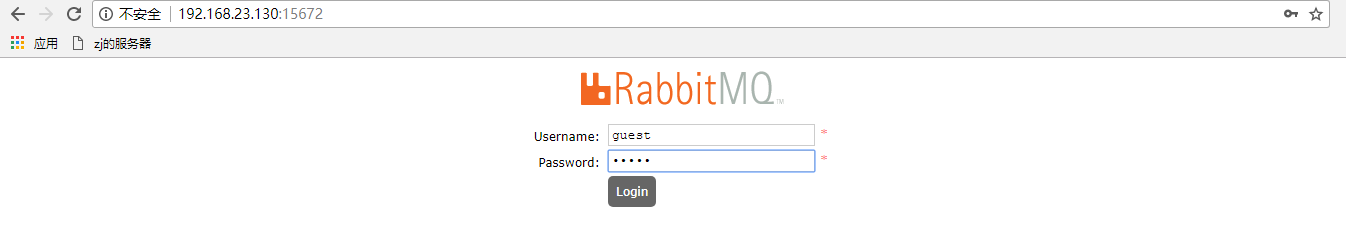
### 1.1.6 查看容器



运行正常

### 1.1.7访问*web* 管理端

容器启动之后就可以访问*web* 管理端了 http://宿主机IP:15672，默认创建了一个 *guest* 用户，密码也是 *guest*。

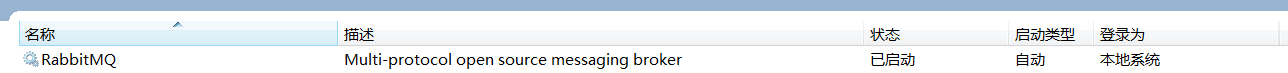


参考：<https://blog.csdn.net/liyuejin/article/details/78410586>

## 1.2下载安装包，安装（比较传统）

#### Windows安装

1. 安装Erland，通过官方下载页面http://www.erlang.org/downloads获取exe安装包，直接打开并完成安装。
2. 安装RabbitMQ，通过官方下载页面https://www.rabbitmq.com/download.html获取exe安装包。
3. 下载完成后，直接运行安装程序。
4. RabbitMQ Server安装完成之后，会自动的注册为服务，并以默认配置启动起来。

[](http://blog.didispace.com/assets/5-1.png)

# 2 、springboot整合rabbitmq的一个简单例子

## 2.1 引入依赖

|  |
| --- |
| <!--引入AMQP模块支持RabbitMQ--> <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-amqp</artifactId> </dependency> |

## 2.2配置rabbitmq的安装地址、端口以及账户信息

|  |
| --- |
| #配置rabbitmq的安装地址、端口以及账户信息 spring.application.name=win-rabbitmq  spring.rabbitmq.host=192.168.23.130 spring.rabbitmq.port=5672 spring.rabbitmq.username=guest spring.rabbitmq.password=guest |

## 2.3队列配置

|  |
| --- |
| package jit.hf.agriculture.config;  import org.springframework.amqp.core.Queue; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration;  */\*\*  \** ***@Auther:*** *zj  \** ***@Date:*** *2018/8/9 16:34  \** ***@Description:*** *创建RabbitMQ的配置类RabbitConfig，  \* 用来配置队列、交换器、路由等高级信。  \* 这里我们以入门为主，先以最小化的配置来定义，以完成一个基本的生产和消费过程。  \*/* @Configuration public class RabbitConfig {   @Bean  public Queue helloQueue() {  return new Queue("hello");  }  } |

## 2.4发送者

|  |
| --- |
| package jit.hf.agriculture.Component;  import org.springframework.amqp.core.AmqpTemplate; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Component;  import java.util.Date;  */\*\*  \** ***@Auther:*** *zj  \** ***@Date:*** *2018/8/9 16:26  \** ***@Description:*** *创建消息生产者Sender(RabbitMq test)  \* 通过注入AmqpTemplate接口的实例来实现消息的发送，AmqpTemplate接口定义了一套针对AMQP协议的基础操作。  \* 在Spring Boot中会根据配置来注入其具体实现。  \* 在该生产者，我们会产生一个字符串，并发送到名为hello的队列中。  \*/* @Component public class Sender {   @Autowired  private AmqpTemplate rabbitTemplate;   public void send() {  String context = "hello " + new Date();  System.*out*.println("Sender : " + context);  this.rabbitTemplate.convertAndSend("hello", context);  }  } |

## 2.5消费者

|  |
| --- |
| package jit.hf.agriculture.Component;  import org.springframework.amqp.rabbit.annotation.RabbitHandler; import org.springframework.amqp.rabbit.annotation.RabbitListener; import org.springframework.stereotype.Component;  */\*\*  \** ***@Auther:*** *zj  \** ***@Date:*** *2018/8/9 16:30  \** ***@Description:*** *创建消息消费者Receiver(RabbitMq test)  \* 通过@RabbitListener注解定义该类对hello队列的监听，并用@RabbitHandler注解来指定对消息的处理方法。  \* 所以，该消费者实现了对hello队列的消费，消费操作为输出消息的字符串内容。  \*/* @Component @RabbitListener(queues = "hello") public class Receiver {   @RabbitHandler  public void process(String hello) {  System.*out*.println("Receiver : " + hello);  }  } |

## 2.6 测试（controller层）

|  |
| --- |
| @Autowired private Sender sender;  */\*\*  \* 简单的rabbitmq测试  \*/* @GetMapping("/test/rabbit/hello") public void TestRabbitHello() {  sender.send(); } |

结果：

C:\Users\zj\Desktop\2345截图20180810142947.png

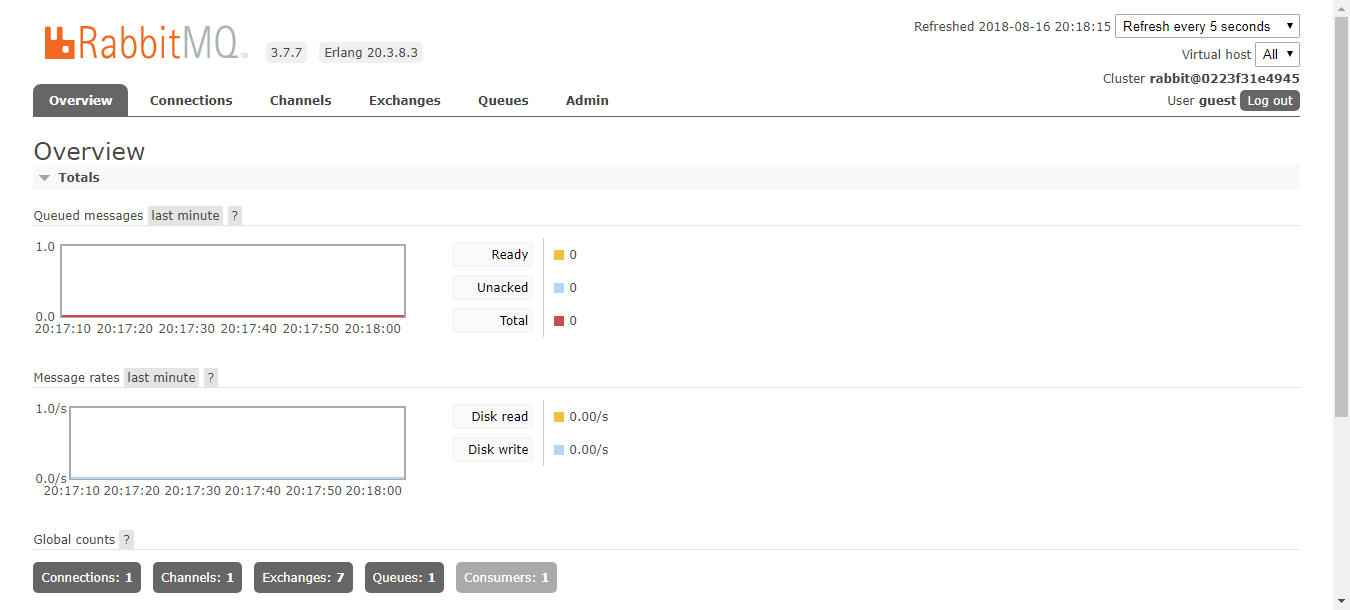
### 2.6.1打开WEB GUI

  >>主页地址: <http://服务ip>地址:15672/ (默认用户名guest,密码guest)

  >>API文档: <http://服务ip>地址:15672/api/

  >>下载 [**rabbitmqadmin**](http://www.rabbitmq.com/management-cli.html): http://server-name:15672/cli/

打开后:



参考：<https://blog.csdn.net/VirBird/article/details/20706159>

# 3.RabbitMQ 使用参考

RabbitMQ：Spring 集成 RabbitMQ 与其概念，消息持久化，ACK机制等 · Issue #2 · 401Studio/WeekLearn

<https://github.com/401Studio/WeekLearn/issues/2>

<https://www.zouyesheng.com/rabbitmq.html#toc1>

## 3.1、通常的操作流程是:

(1) 消费者: 创建信息通道。   
(2) 消费者: 定义消息队列。   
(3) 消费者: 定义特定类型的交换机。   
(4) 消费者: 设定绑定规则 (包括交换机名称、队列名称以及路由键)。   
(5) 消费者: 等待消息。   
(6) 生产者: 创建消息。   
(7) 生产者: 将消息投递给信息通道 (注明接收交换机名称和路由键)。   
(8) 交换机: 获取消息，依据交换机类型决定是否匹配路由规则 (如需匹配，则对比消息路由键和绑定路由键)。   
(9) 消费者: 获取并处理消息，发送反馈。   
(10) 结束: 关闭通道和连接。

@Service用于标注业务层组件，@Controller用于标注控制层组件（如struts中的action）,@Repository用于标注数据访问组件，即DAO组件，而@Component泛指组件，当组件不好归类的时候，我们可以使用这个注解进行标注。

Spring Boot中使用RabbitMQ | 程序猿DD

<http://blog.didispace.com/spring-boot-rabbitmq/>

springboot(八)：RabbitMQ详解 - 纯洁的微笑博客

<http://www.ityouknow.com/springboot/2016/11/30/spring-boot-rabbitMQ.html>

Spring Boot 揭秘与实战（六） 消息队列篇 - RabbitMQ | 梁桂钊的博客

http://blog.720ui.com/2017/springboot\_06\_mq\_rabbitmq/#Spring-Boot-整合-RabbitMQ

spring boot集成RabbitMQ - 简书

<https://www.jianshu.com/p/e1258c004314>

SpringBoot 系列 | 第十篇：使用RabbitMQ收发消息 - CSDN博客

<https://blog.csdn.net/u012943767/article/details/79424592>

第四十六章：SpringBoot & RabbitMQ完成消息延迟消费 - 简书

<https://www.jianshu.com/p/b74a14c7f31d>