

CentOS 7下安装配置RabbitMQ详细教程

[日期：2018-01-27]

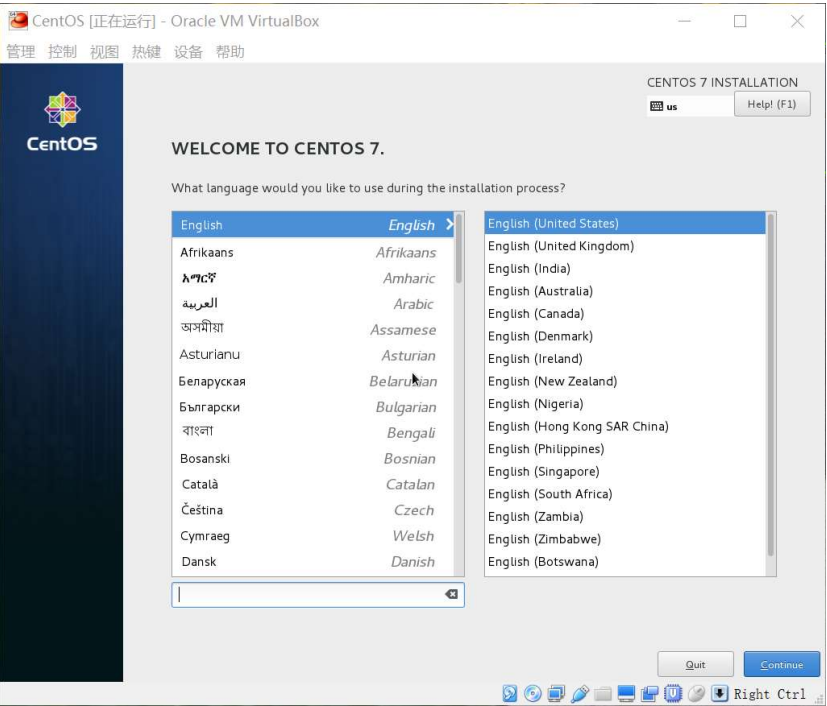
来源：Linux社区 作者：冷豪

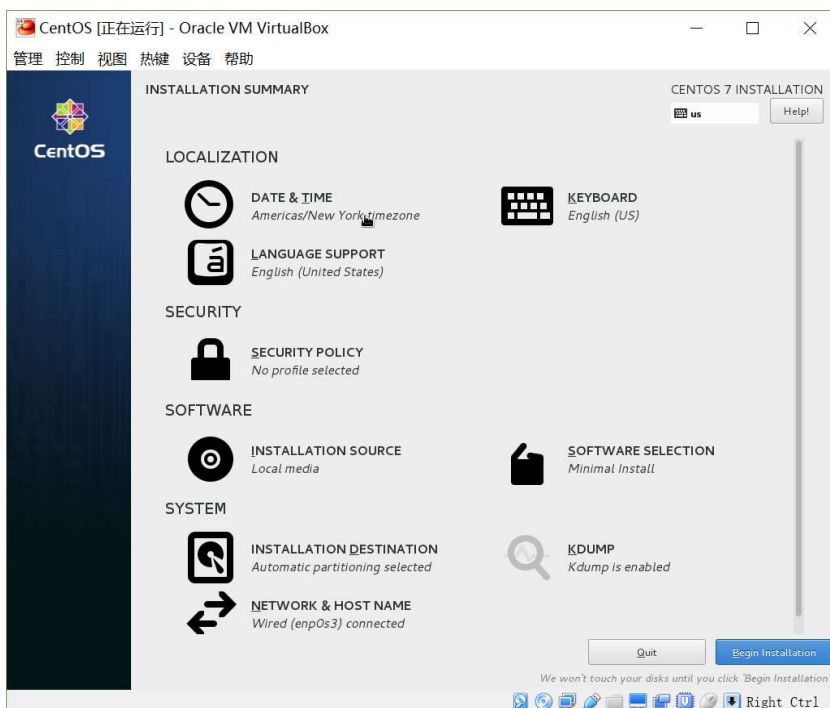
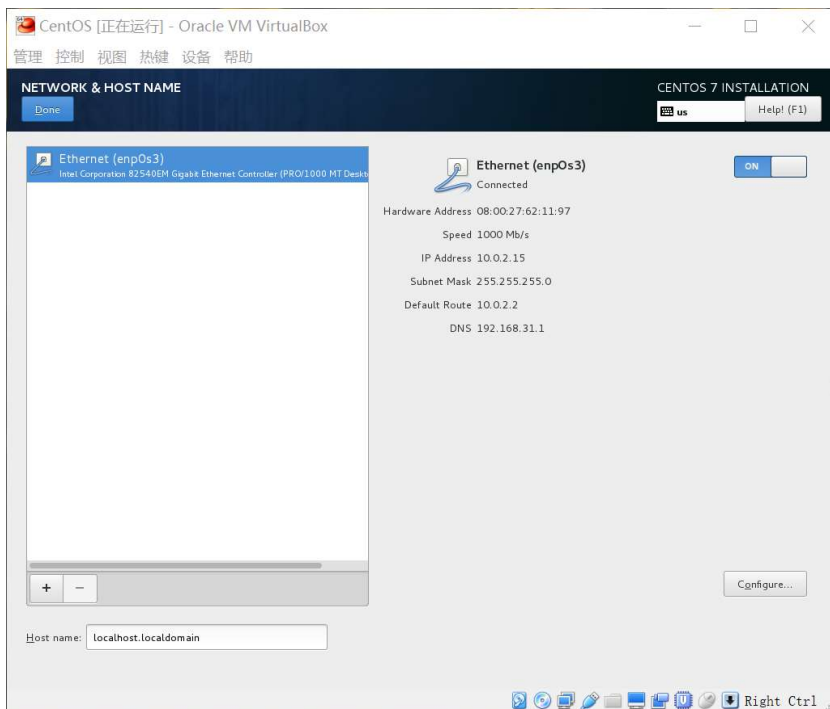
[字体：大 中 小]

最近在学习RabbitMQ，在网上找了不少资料发现都特高端。动辄集群部署，分布式架构什么的，对于一个初学者实在不够友好。心想求人不如求自己，为什么不自己整理一套资料呢？于是《CentOS 7下安装配置RabbitMQ详细教程》诞生。

一、准备工作

据说RabbitMQ是可以部署到Windows环境的，不过作为一个专业级的开发人员怎么能够让这样的事情发生呢？自然我们的准备工作从Linux开始。首先在虚拟机中安装CentOS 7，选择英文，最小安装，默认开启网络以及创建一个root用户：





完成以后进入系统，由于最小安装有一些基本的命令无法使用，因此在进入下一步之前先将ifconfig、vim以及基本的编译环境准备好：

<!-- 安装ifconfig -->

```
yum install net-tools
```

<!-- 安装vim -->

```
yum install vim
```

<!-- 准备基础编译环境 -->

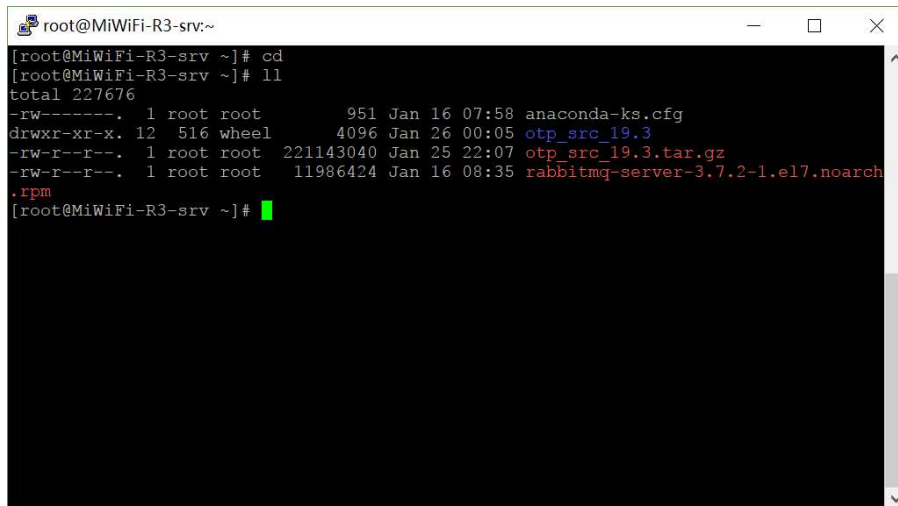
```
yum install gcc glibc-devel make ncurses-devel openssl-devel xz-devel
```

接下来我们从官网下载安装包rabbitmq-server-3.7.2-1.el7.noarch.rpm和otp_src_19.3.tar.gz（千万别问我Erlang是什么，我也是第一次知道这门语言）。上传到虚拟机后执行命令：

<!-- 解压包 -->

```
[root@MiWiFi-R3-srv ~]# tar -xvf otp_src_19.3.tar.gz
```

```
[root@MiWiFi-R3-srv ~]# ll
```

A terminal window titled 'root@MiWiFi-R3-srv:~' showing the output of the 'll' command. The output lists files in the current directory: 'anaconda-ks.cfg' (951 bytes, Jan 16 07:58), 'otp_src_19.3' (4096 bytes, Jan 26 00:05), 'otp_src_19.3.tar.gz' (221143040 bytes, Jan 25 22:07), and 'rabbitmq-server-3.7.2-1.el7.noarch.rpm' (11986424 bytes, Jan 16 08:35). The permissions for each file are also shown: '-rw-----' for anaconda-ks.cfg, 'drwxr-xr-x' for otp_src_19.3, '-rw-r--r--' for otp_src_19.3.tar.gz, and '-rw-r--r--' for rabbitmq-server-3.7.2-1.el7.noarch.rpm. The total size of the files is 227676 bytes. The prompt is '[root@MiWiFi-R3-srv ~]# ' with a green cursor.

```
[root@MiWiFi-R3-srv ~]# mkdir /usr/local/erlang
```

```
[root@MiWiFi-R3-srv ~]# cd otp_src_19.3
```

```
[root@MiWiFi-R3-srv otp_src_19.3]# ./configure --prefix=/usr/local/erlang --without-javac
```

```
[root@MiWiFi-R3-srv otp_src_19.3]# make && make install
```

编译&安装完成以后配置Erlang环境变量：

```
[root@MiWiFi-R3-srv otp_src_19.3]# vim /etc/profile
```

#追加环境变量到文件末尾

```
ERL_HOME=/usr/local/erlang
PATH=$ERL_HOME/bin:$PATH
export ERL_HOME PATH
```

```
[root@MiWiFi-R3-srv otp_src_19.3]# source /etc/profile
```

接下来可以正式安装RabbitMQ：

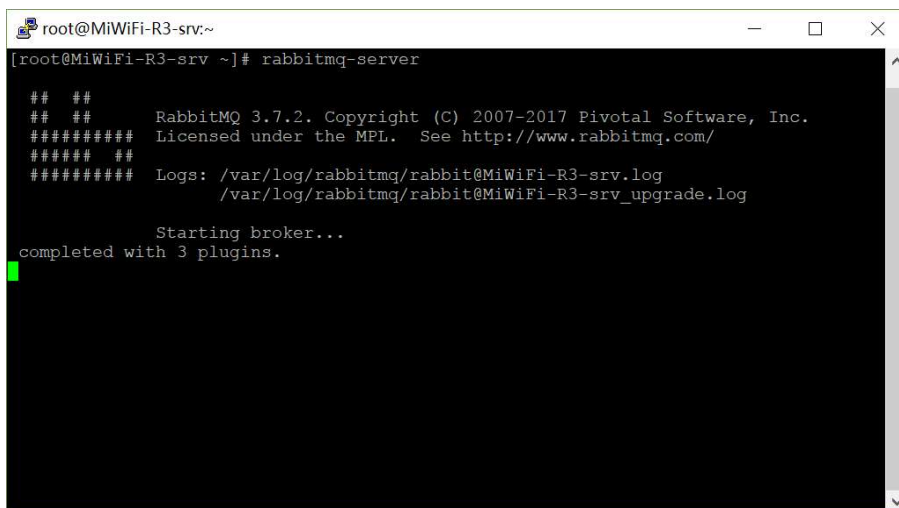
```
[root@MiWiFi-R3-srv otp_src_19.3]# cd ..
[root@MiWiFi-R3-srv ~]# rpm -ivh --nodeps rabbitmq-server-3.7.2-1.el7.noarch.rpm
```

运行RabbitMQ需要首先开放15672和5672端口:

```
[root@MiWiFi-R3-srv ~]# firewall-cmd --zone=public --add-port=15672/tcp --permanent
[root@MiWiFi-R3-srv ~]# firewall-cmd --zone=public --add-port=5672/tcp --permanent
[root@MiWiFi-R3-srv ~]# firewall-cmd --reload
```

正常情况下RabbitMQ已经安装完成, 最后测试一下:

```
[root@MiWiFi-R3-srv ~]# rabbitmq-plugins enable rabbitmq_management
[root@MiWiFi-R3-srv ~]# rabbitmq-server
```



```
root@MiWiFi-R3-srv:~
[root@MiWiFi-R3-srv ~]# rabbitmq-server

## ##
## ##   RabbitMQ 3.7.2. Copyright (C) 2007-2017 Pivotal Software, Inc.
#####   Licensed under the MPL.  See http://www.rabbitmq.com/
#####   ##
#####   Logs:  /var/log/rabbitmq/rabbit@MiWiFi-R3-srv.log
                /var/log/rabbitmq/rabbit@MiWiFi-R3-srv_upgrade.log

        Starting broker...
completed with 3 plugins.
█
```

正常启动以后, 我们可以在本地使用浏览器中访问管理页面: <http://<虚拟机IP>:15672/>

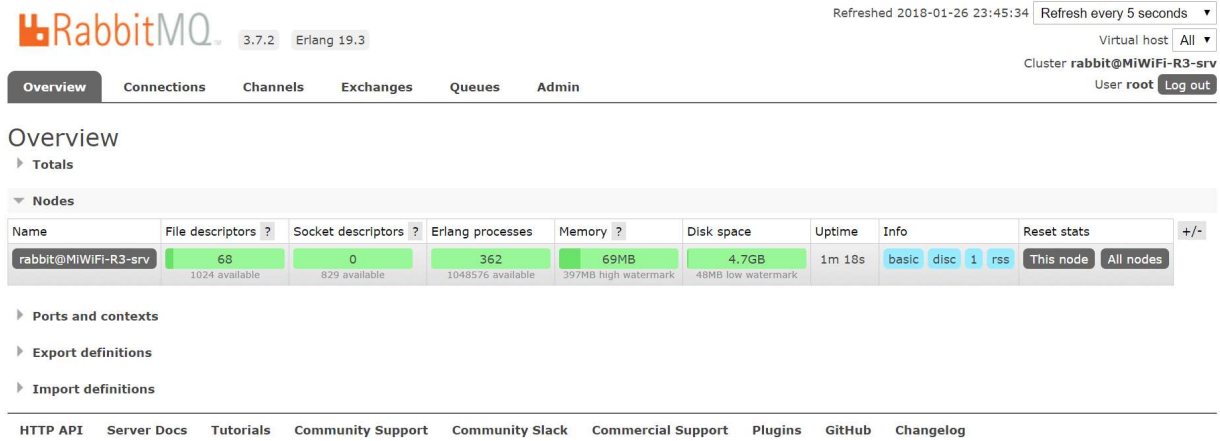


大功告成...

至此, 我们的准备工作已经完成了80%。接下来我们需要为RabbitMQ创建用户并赋权。

```
[root@MiWiFi-R3-srv ~]# rabbitmqctl add_user root root
[root@MiWiFi-R3-srv ~]# rabbitmqctl set_user_tags root administrator
[root@MiWiFi-R3-srv ~]# rabbitmqctl set_permissions -p / root '.*' '.*' '.*'
<!-- 后台启动 -->
[root@MiWiFi-R3-srv ~]# rabbitmq-server -detached
```

重新通过在本机浏览器访问管理页面, 输入用户名和密码。



二、简单开发指南

深入的开发案例网上很多，我就不在这里重复的发明轮子了。作为一个指南，这里主要介绍两种开发方式，更加具体的用例我可能会在以后的文章中专门介绍。

1.单独使用——一个简单的消息生产者

通过maven引入依赖

```
<dependency>
  <groupId>com.rabbitmq</groupId>
  <artifactId>amqp-client</artifactId>
  <version>5.1.1</version>
</dependency>
```

创建生产者：

```
package com.learnhow.rabbitmq;
```

```
import java.io.IOException;
import java.util.concurrent.TimeoutException;
```

```
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;
```

```
public class App {
    // 队列名称
    public final static String QUEUE_NAME = "Hello.rabbitMQ";

    public static void main(String[] args) throws IOException, TimeoutException {
        // 连接工厂
        ConnectionFactory factory = new ConnectionFactory();
        // 配置连接属性
```

```

factory.setHost("192.168.31.244"); // 虚拟机地址
factory.setPort(5672); // 端口号
factory.setUsername("root"); // 用户名
factory.setPassword("root"); // 密码

// 得到连接，创建通道
Connection connection = factory.newConnection();
Channel channel = connection.createChannel();

// 声明一个叫Hello.rabbitMQ的队列
channel.queueDeclare(QUEUE_NAME, false, false, false, null);

String message = "Hello RabbitMQ";

// 发送消息
channel.basicPublish("", QUEUE_NAME, null, message.getBytes("UTF-8"));

// 关闭通道和连接
channel.close();
connection.close();
}
}

```

执行完成以后切换到浏览器的管理页面：

RabbitMQ 3.7.2 Erlang 19.3

Refreshed 2018-01-27 00:02:41 Refresh every 5 seconds

Virtual host All

Cluster rabbit@MiWiFi-R3-srv

User root Log out

Overview Connections Channels Exchanges **Queues** Admin

Queues

▶ All queues (1)

Overview			Messages			Message rates			
Name	Features	State	Ready	Unacked	Total	incoming	deliver	get	ack
Hello.rabbitMQ		idle	1	0	1	0.00/s			

▶ Add a new queue

HTTP API Server Docs Tutorials Community Support Community Slack Commercial Support Plugins GitHub Changelog

我们发现刚才在main函数中声明的QUEUE NAME已经出现了。

2.Spring AMQP——与Spring Boot集成

Maven依赖：

```

<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.5.9.RELEASE</version>
</parent>
<dependencies>
  <dependency>

```

```
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-amqp</artifactId>
</dependency>
</dependencies>
```

Spring Boot启动项，声明测试队列：

```
package org.dispatcher;
```

```
import org.springframework.amqp.core.Queue;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
```

```
@SpringBootApplication
public class DispatcherApplication {
    @Bean
    public Queue helloQueue() {
        return new Queue("helloQueue");
    }

    public static void main(String[] args) throws Exception {
        SpringApplication.run(DispatcherApplication.class, args);
    }
}
```

消息生产者：

```
package org.dispatcher.controller;
```

```
import java.util.Date;
```

```
import org.springframework.amqp.core.AmqpTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
```

```
@Component
public class Producer {
    @Autowired
    private AmqpTemplate rabbitTemplate;

    public void send() {
        String sendMsg = "Hi~ " + new Date();
        this.rabbitTemplate.convertAndSend("helloQueue", sendMsg);
    }
}
```

消息接收者：

```
package org.dispatcher.controller;

import org.springframework.amqp.rabbit.annotation.RabbitHandler;
import org.springframework.amqp.rabbit.annotation.RabbitListener;
import org.springframework.stereotype.Component;

@Component
@RabbitListener(queues = "helloQueue")
public class Receiver {
    @RabbitHandler
    public void process(String msg) {
        System.out.println("Receiver: " + msg);
    }
}
```

Restful接口：

```
package org.dispatcher.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/rabbitmq")
```



```

public class RabbitMqController {
    private static final String SUCCESS = "SUCCESS";
    private static final String FAILURE = "FAILURE";

    @Autowired
    private Producer producer;

    @GetMapping("/push")
    public String push() {
        producer.send();
        return SUCCESS;
    }
}

```

配置文件application.yml:

```

server:
  port: 8081
spring:
  rabbitmq:
    host: 192.168.31.244
    port: 5672
    username: root
    password: root
    virtual-host: /
    publisher-confirms: true

```

启动Spring Boot，访问：<http://localhost:8081/rabbitmq/push>，再切换到管理页面：

Refreshed 2018-01-27 00:27:37 Refresh every 5 seconds Virtual host All Cluster rabbit@MiWiFi-R3-srv User root Log out

Overview Connections Channels Exchanges **Queues** Admin

Queues

▼ All queues (2)

Pagination

Page 1 of 1 - Filter: ☐ Regex ? Displaying 2 items , page size up to: 100

Overview			Messages			Message rates			
Name	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	
Hello.rabbitMQ		Idle	1	0	1	0.00/s			
helloQueue	D	Idle	0	0	0	0.00/s	0.00/s	0.00/s	

► Add a new queue

HTTP API Server Docs Tutorials Community Support Community Slack Commercial Support Plugins GitHub Changelog

新的QUEUE NAME证明Spring Boot与RabbitMQ整合成功。

CentOS 7.2 下 RabbitMQ 集群搭建 <http://www.linuxidc.com/Linux/2016-12/137812.htm>

CentOS7环境安装使用专业的消息队列产品RabbitMQ <http://www.linuxidc.com/Linux/2016-11/13673.htm>

RabbitMQ入门教程 <http://www.linuxidc.com/Linux/2015-02/113983.htm>

在CentOS7上安装RabbitMQ 详解 <http://www.linuxidc.com/Linux/2017-05/143765.htm>

NServiceBus 结合 RabbitMQ 使用教程 <http://www.linuxidc.com/Linux/2017-05/143787.htm>

CentOS 7下RabbitMQ集群安装配置 <http://www.linuxidc.com/Linux/2017-10/147707.htm>

RabbitMQ实战：高效部署分布式消息队列 中文PDF扫描版 <http://www.linuxidc.com/Linux/2017-10/147592.htm>

CentOS7上RabbitMQ安装详述 <http://www.linuxidc.com/Linux/2017-12/149202.htm>

RabbitMQ分布式集群架构和高可用性（HA） <http://www.linuxidc.com/Linux/2017-12/149466.htm>

RabbitMQ 的详细介绍： [请点击这里](#)

RabbitMQ 的下载地址： [请点击这里](#)

本文永久更新链接地址： <http://www.linuxidc.com/Linux/2018-01/150600.htm>