链路追踪监控使用文档

# 1 前言

一个复杂的业务流程通常会被拆分多个微服务系统来完成，微服务间通过Feign来通信。当业务流程足够复杂时，一个完整的HTTP请求调用链一般会经过多个微服务系统，要通过日志来跟踪一整个调用链变得不再那么简单。我们可以通过Spring Cloud Sleuth来解决这个问题。

微服务中我们常使用Spring Cloud Feign来远程访问受保护的资源，那么如何通过Spring Cloud Sleuth来追踪这个过程？我们借助Zipkin以图形化界面的方式展示整个链路的追踪过程。

# 2 整合Spring Cloud Sleuth

在微服务系统模块的pom里引入sleuth的依赖：

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-sleuth</artifactId>

</dependency>

然后在微服务的测试模块打印hello方法的日志信息

@Slf4j

@RestController

public class TestController {

@Autowired

private IHelloService helloService;

@GetMapping("hello")

public String hello(String name) {

log.info("Feign调用wm-server-system的/hello服务");

return this.helloService.hello(name);

}

......

}

使用PostMan获取令牌后，访问 [localhost:8301/test/hello?name=夏天](localhost:8301/test/hello?name=%E5%A4%8F%E5%A4%A9)：

这时候观察打印的日志：

2020-07-23 hh:mm:ss INFO : Feign调用wm-server-test的/hello服务 [wm-server-system,72bb0469bee07104,72bb0469bee07104,false]

[wm-server-system,72bb0469bee07104, 43597a6edded6f2e,false]信息，这些信息由Spring Cloud Sleuth生成，用于跟踪微服务请求链路。这些信息包含了4个部分的值，它们的含义如下：

wm-server-system微服务的名称，与spring.application.name对应；

72bb0469bee07104称为Trace ID，在一条完整的请求链路中，这个值是固定的。观察上面的日志即可证实这一点；

43597a6edded6f2e称为Span ID，它表示一个基本的工作单元；

false表示是否要将该信息输出到Zipkin等服务中来收集和展示，这里我们还没有集成Zipkin，所以为false。

要跟踪整条请求链路，就可以通过traceId来完成。但是，从海量日志里捞取traceId并追踪也不是一件轻松的事情，不过可以借助zipkin实现使用图形化界面的方式追踪请求链路。

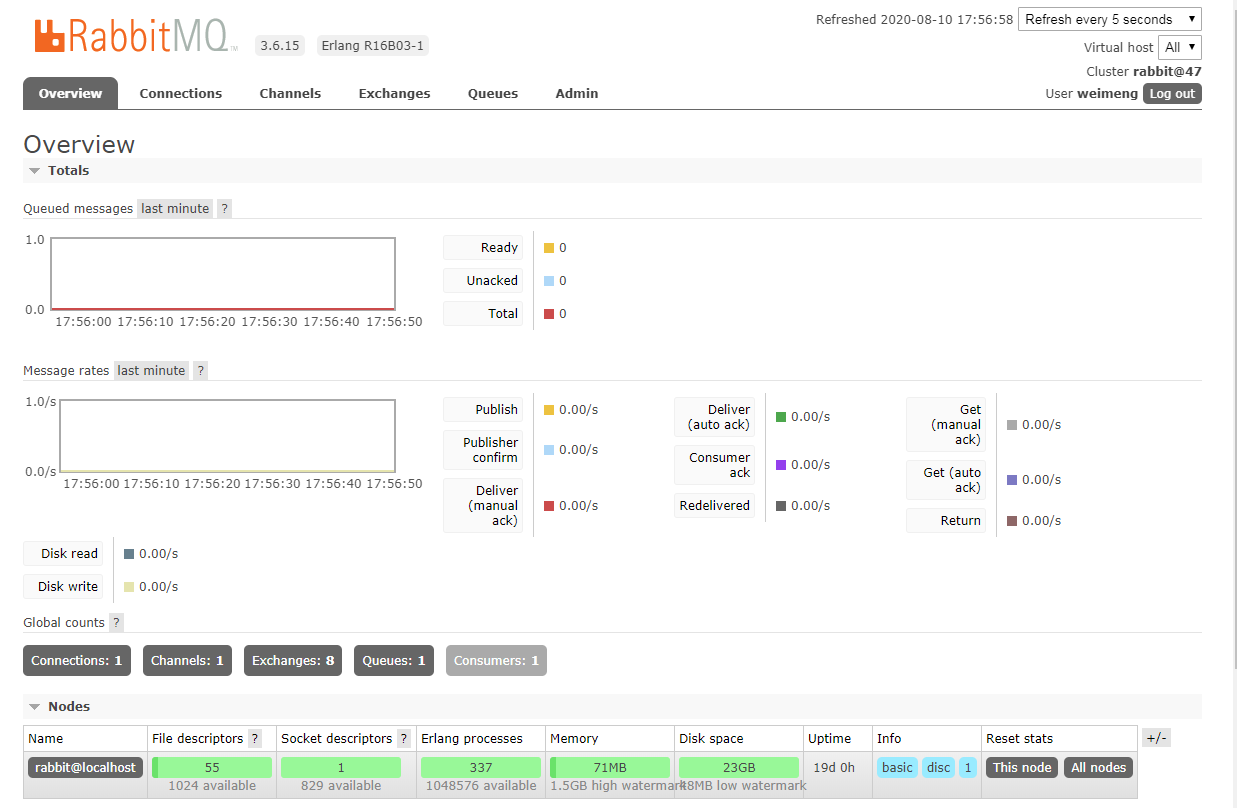
# 3 整合Zipkin

在整合Zipkin之前，需要先搭建RabbitMQ。RabbitMQ用于收集Sleuth提供的追踪信息，然后Zipkin Server从RabbitMQ里获取，这样可提升性能。

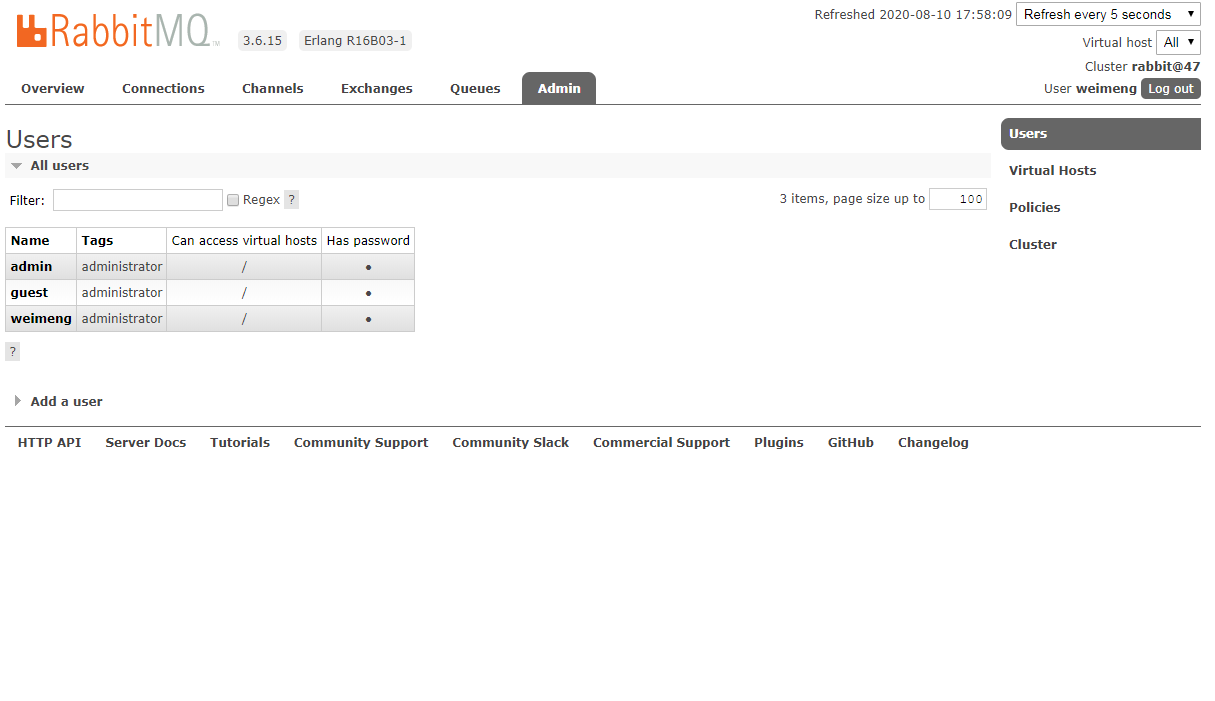
在安装RabbitMQ之前，需要先安装Erlang/OTP，下载地址为：<http://www.erlang.org/downloads/>安装完毕后，下载RabbitMQ，下载地址为 <http://www.rabbitmq.com/install-windows.html>，安装完RabbitMQ之后，我们到RabbitMQ安装目录的sbin下执行如下命令：

rabbitmq-plugins enable rabbitmq\_management

然后在浏览器中输入[http://localhost:15672](http://localhost:15672/)，用户名和密码都是guest，登录后可看到：



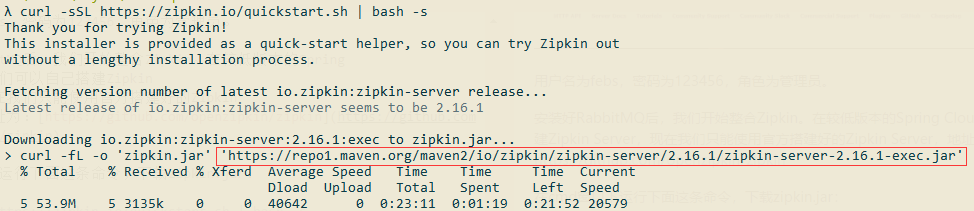
点击Admin Tab页面，新增一个用户：



安装好RabbitMQ后，开始整合Zipkin。在较低版本的Spring Cloud中，可以自己搭建Zipkin Server，目前只能使用官方搭建好的Zipkin Server，地址为：<https://github.com/openzipkin/zipkin>

在cmd窗口下运行下面这条命令（windows下没有curl环境的话，可以在git bash中运行这条命令），下载zipkin.jar：

curl -sSL https://zipkin.io/quickstart.sh | bash -s



zipkin支持将追踪信息保存到MySQL数据库，所以在运行zipkin.jar之前，先准备好相关库表，SQL脚本地址为：<https://github.com/openzipkin/zipkin/blob/master/zipkin-storage/mysql-v1/src/main/resources/mysql.sql>

运行这些SQL后，数据库创建了三张新的表：

库表准备好后，运行下面这条命令启动zipkin.jar：

java -jar zipkin.jar --server.port=8402 --zipkin.storage.type=mysql --zipkin.storage.mysql.db=febs\_cloud\_base --zipkin.storage.mysql.username=root --zipkin.storage.mysql.password=123456 --zipkin.storage.mysql.host=localhost --zipkin.storage.mysql.port=3306 --zipkin.collector.rabbitmq.addresses=localhost:5672 --zipkin.collector.rabbitmq.username=username --zipkin.collector.rabbitmq.password=123456

上面命令指定了数据库链接和RabbitMQ链接信息。更多可选配置可以解压zipkin.jar，查看zipkin\BOOT-INF\classes路径下的zipkin-server-shared.yml配置类源码。

启动好zipkin.jar后，在微服务的模块的pom里引入如下依赖：

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-zipkin</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.amqp</groupId>

<artifactId>spring-rabbit</artifactId>

</dependency>

接着在微服务的模块的配置文件application.yml里添加如下配置:

spring:

zipkin:

sender:

type: rabbit

sleuth:

sampler:

probability: 1

rabbitmq:

host: localhost

port: 5672

username: username

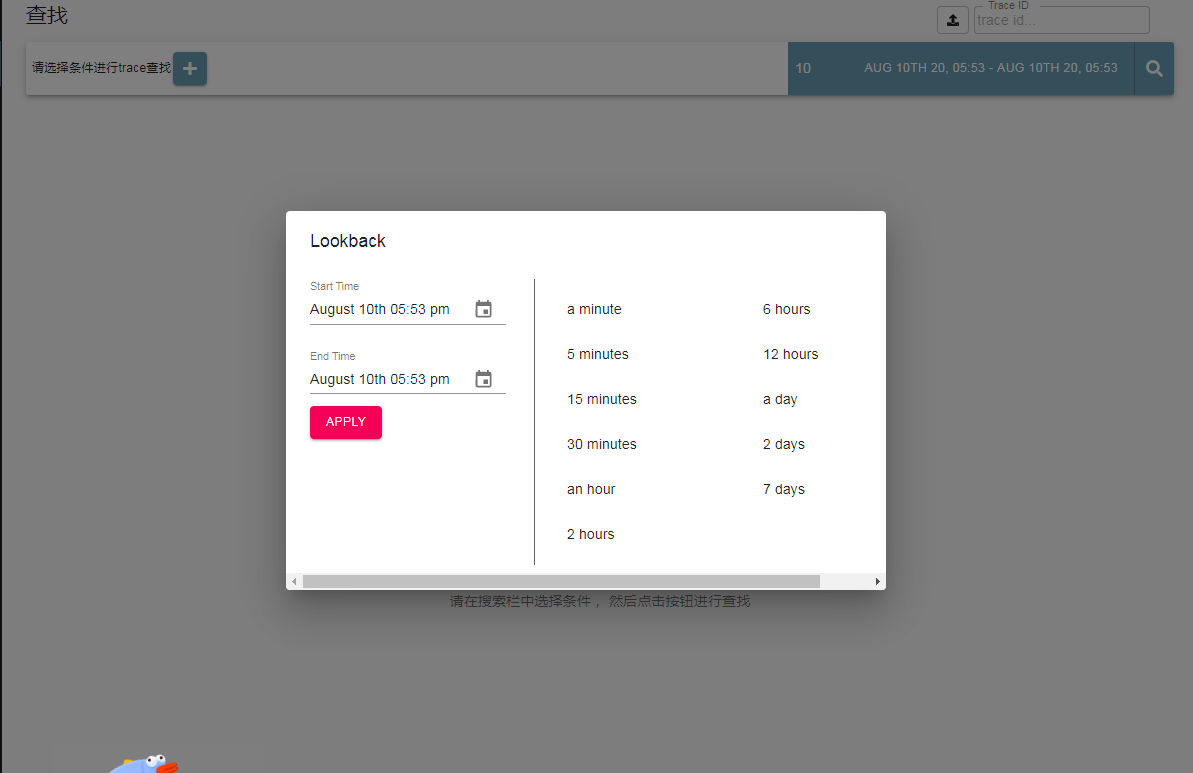
password: 123456

spring.zipkin.sender.type指定了使用RabbitMQ收集追踪信息；

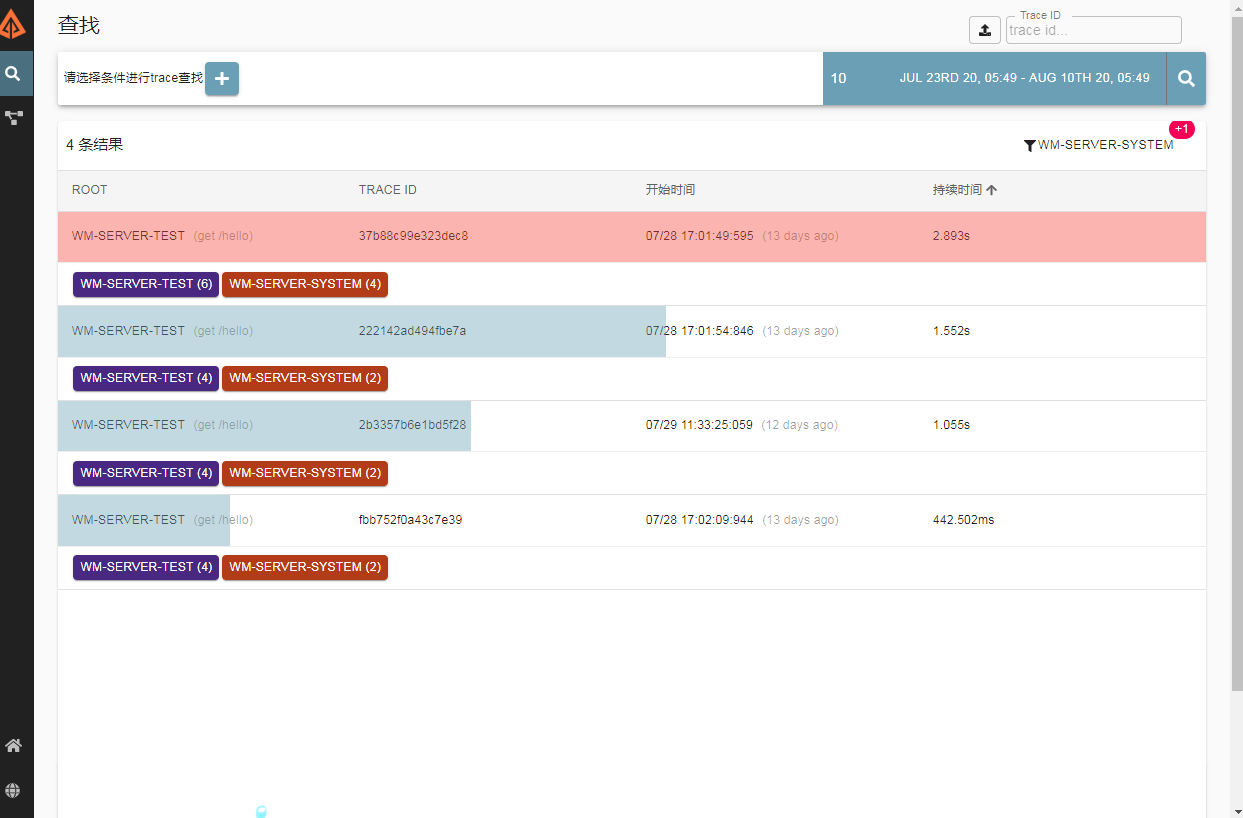
spring.sleuth.sampler.probability默认值为0.1，即采样率才1/10，发送10笔请求只有一笔会被采集。为了测试方便，我们可以将它设置为1，即100%采样；

spring.rabbitmq用于配置RabbitMQ连接信息。

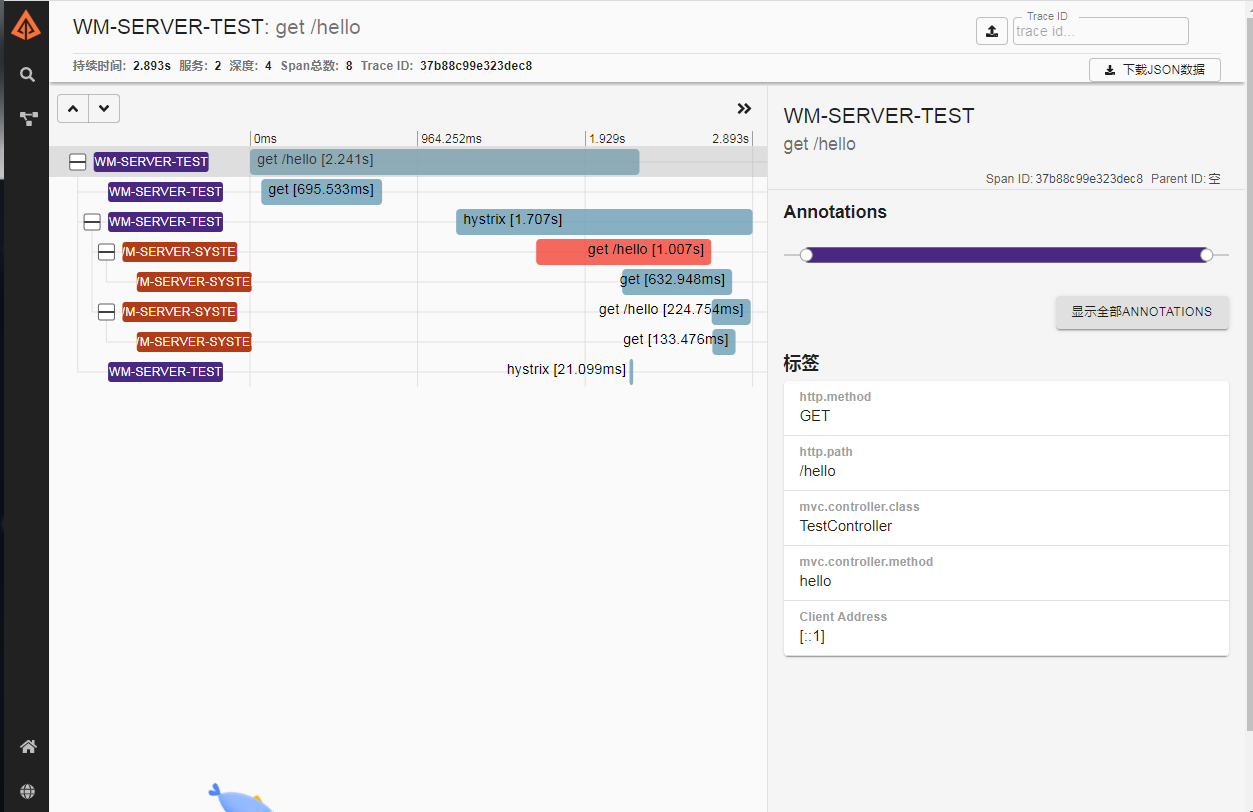
添加好配置后，重启微服务模块，发送一笔[localhost:8301/test/hello?name=夏天](localhost:8301/test/hello?name=%E5%A4%8F%E5%A4%A9)请求后，使用浏览器访问<http://localhost:8402/zipkin/>链接（个人习惯使用Lens UI）：



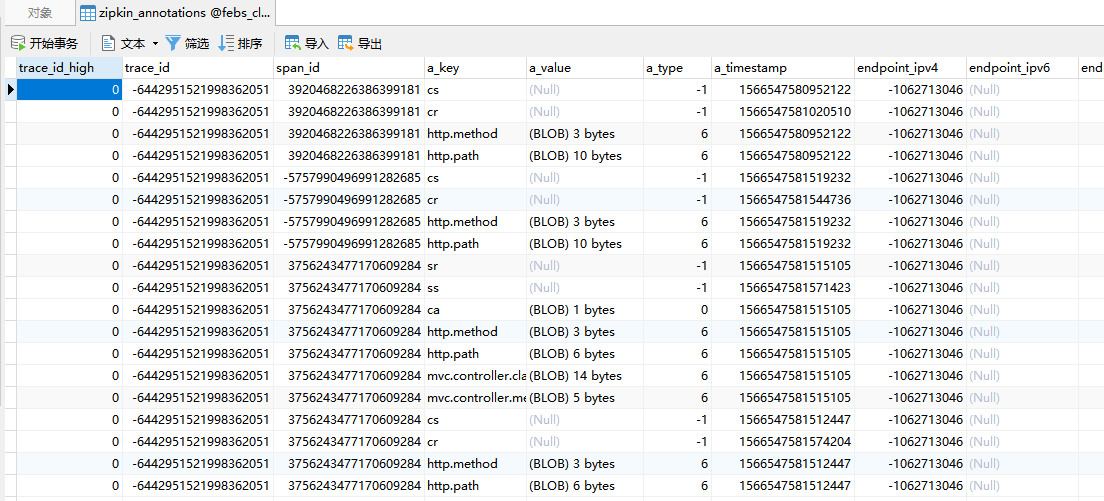
选择查找的条件



查看依赖关系



查看表数据是存储了链路追踪信息：



# 4 RabbitMQ 的安装

Zipkin依赖的消息中间件。

## 4.1 下载erlang

下载地址：<http://www.erlang.org/downloads/19.3>

因为RabbitMq的运行依赖erlang，所以先下载erlang（erlang是什么可自行百度）。

**erlang的版本会影响到rabbitmq的安装**，两者有个版本对照，查看对照的地址：<http://www.rabbitmq.com/which-erlang.html>

## 4.2 下载rbbitmq

官网地址：<http://www.rabbitmq.com/>

具体的安装包的3.7.5：<https://github.com/rabbitmq/rabbitmq-server/releases/tag/v3.7.5>

## 4.3 安装步骤

（1）安装erlang

tar -zxvf  otp\_src\_19.3.tar.gz 解压

mv otp\_src\_19.3 /opt/modules/  解压完成之后，移动到指定目录（不是必须的，根据习惯，比如习惯建一个目录：/opt/modules，然后把解压的移动到这个文件夹里）

[root@localhost ~]# cd /opt/modules/otp\_src\_19.3

[root@localhost otp\_src\_19.3]# ./configure --prefix=/opt/erlang

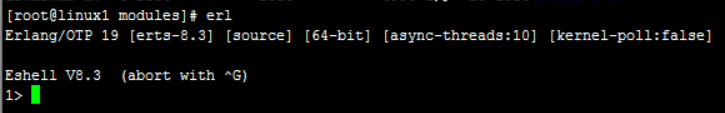
**配置环境变量：**

ERLANG\_HOME=/opt/erlang  
export PATH=$PATH:$ERLANG\_HOME/bin  
export ERLANG\_HOME

source /etc/profile 使其生效

测试时否安装成功：

[root@localhost ~]# erl

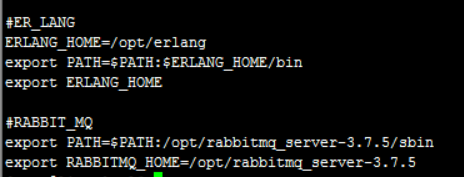


（2）安装 rabbitmq

[root@localhost ~]#  tar -xvf rabbitmq-server-generic-unix-3.7.5.tar

[root@localhost ~]#  mv rabbitmq\_server-3.7**.5 /opt/modules**

**配置环境变量并使其生效**



**测试是否安装成功：**

rabbitmq-server -detached

正常情况下，此时应该能够正常启动了，然额：

Warning: PID file not written; -detached was passed.

查看日志：

cat [/opt/rabbitmq\_server-3.7.5/var/log/rabbitmq/rabbit@linux1.log](mailto:/opt/rabbitmq_server-3.7.5/var/log/rabbitmq/rabbit@linux1.log)

Error description:

init:do\_boot/3

init:start\_em/1

rabbit:start\_it/1 line 446

rabbit:broker\_start/0 line 320

rabbit\_plugins:prepare\_plugins/1 line 289

rabbit\_plugins:ensure\_dependencies/1 line 263

throw:{error,{missing\_dependencies,[crypto,ssl],

[amqp10\_client,cowboy,cowlib,rabbitmq\_aws,

rabbitmq\_management,

rabbitmq\_management\_agent,rabbitmq\_shovel,

rabbitmq\_trust\_store]}}

Log file(s) (may contain more information):

/opt/rabbitmq\_server-3.7.5/var/log/rabbitmq/rabbit@linux1.log

/opt/rabbitmq\_server-3.7.5/var/log/rabbitmq/rabbit@linux1\_upgrade.log

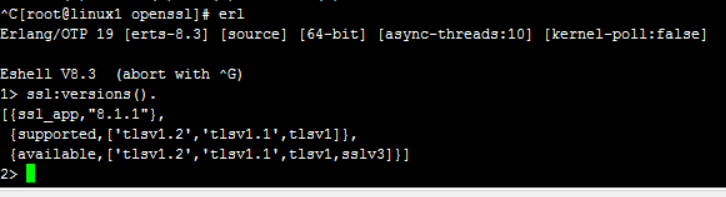
解决：

yum install openssl

yum install openssl-devel

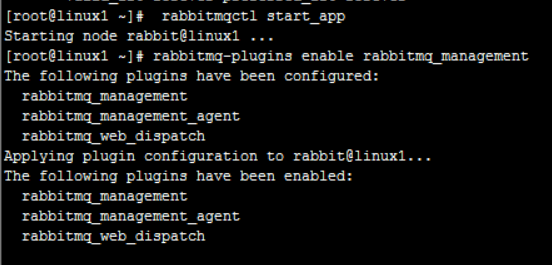
/configure --prefix=/usr/local/erlang --with-ssl --enable-threads --enable-smp-support --enable-kernel-poll --enable-hipe --without-javac

make && install



一切都正常了。启动mq(/sbin/service rabbitmq-server start)，并查看状态：

rabbitmqctl status



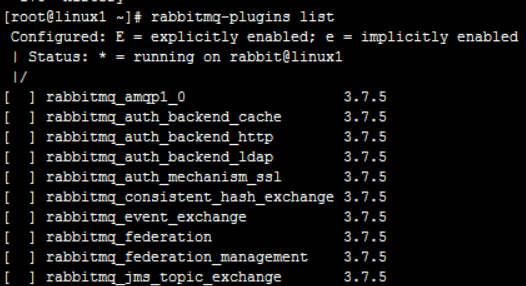
（3）安装管理界面

依次执行下列指令：

开启rabbitmq：[root@localhost ~]# rabbitmqctl start\_app

开启管理插件：[root@localhost ~]# rabbitmq-plugins enable rabbitmq\_management

查看插件集合：rabbitmq-plugins list



如果上面都顺利，重启rabbitmq服务，则可以打开web管理界面：

