# **ActiveMq**

# 入门概述

## 1.1前言

## 1.2生活案例

系统之间接口耦合比较严重

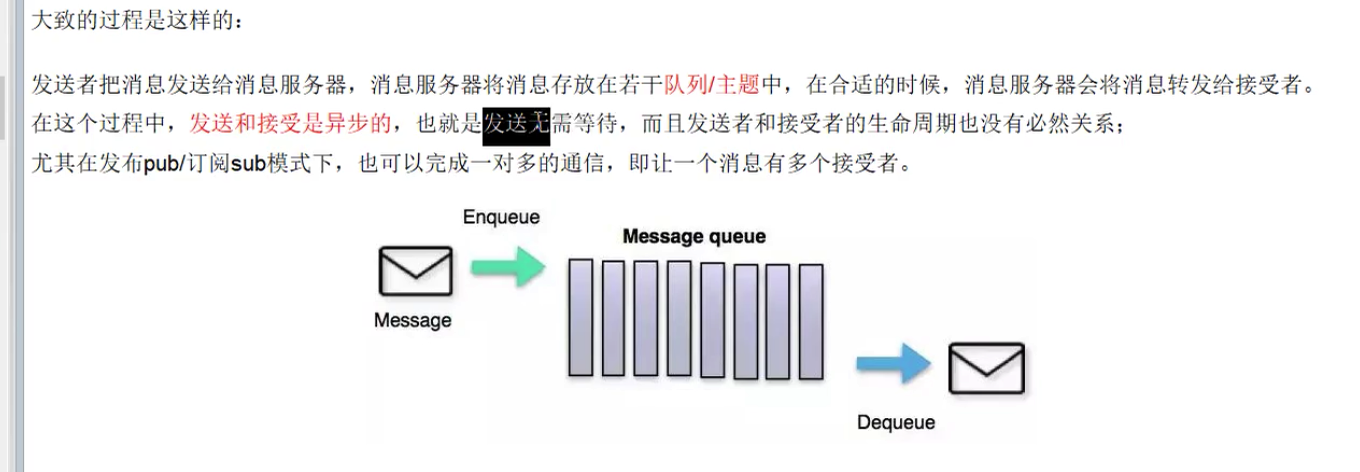
面对大流量并发时，容易被冲垮

等待同步存在性能问题

## 1.3是什么

定义：面向消息的中间件

对列/主题



## 1.4能干嘛

解耦，削峰，异步

mq解决的问题？

1. 解约耦合调用
2. 异步模型
3. 抵御洪峰流量，达到保护主业务，削峰

## 1.5 去哪下载

<http://activemq.apache.org/components/classic/download/>

## 1.6 怎么玩

1.最主要的功能：实现高可用、高性能、高伸缩、易用和安全的企业级面向消息服务的系统

2.异步消息的消费和处理

3.控制消息的消费顺序

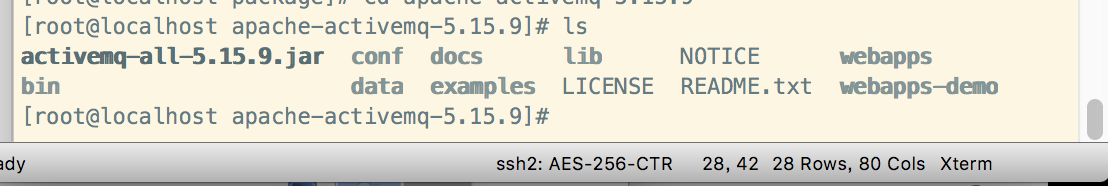
4.可以和spring/springboot整合简化编码

5.配置集群容错的MQ集群

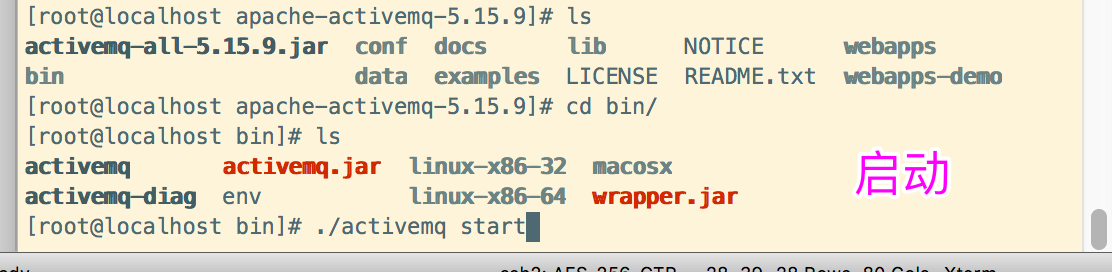
# ActiveMQ安装和控制台

首先要安装jdk

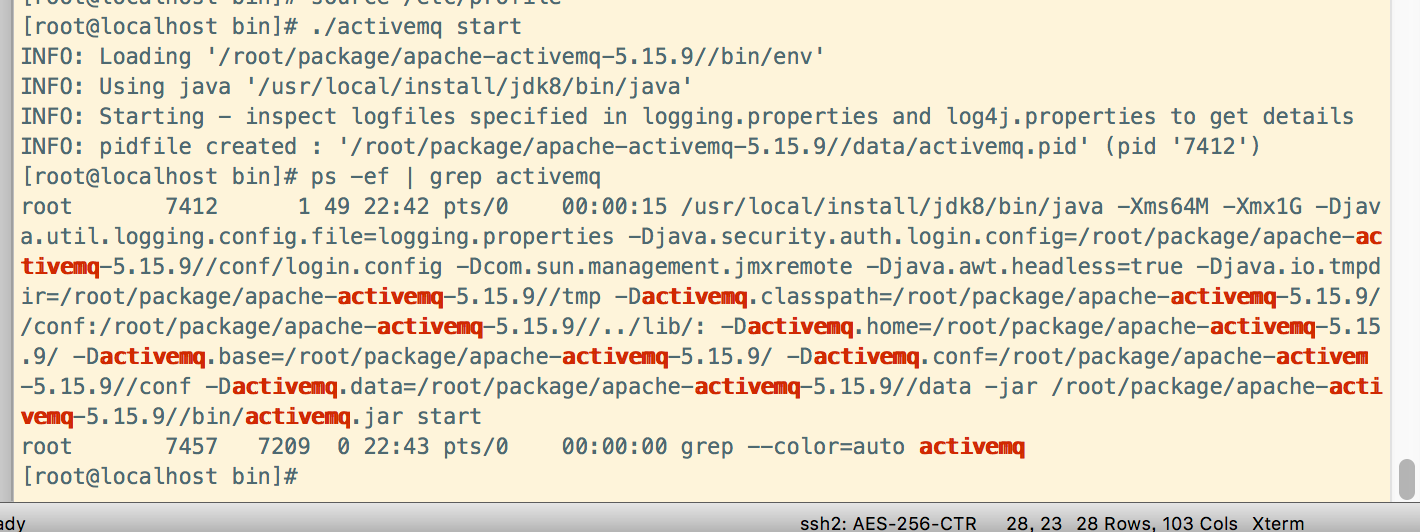
1. 下载apache-activemq-5.15.9-bin.tar.gz
2. Tar -vxf apache-activemq-5.15.9-bin.tar.gz



1. 启动



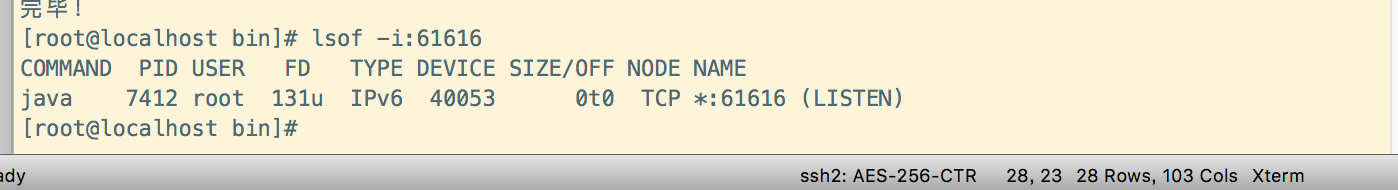
Ps -ef|grep activemq|grep -v grep



Netstat -anp|grep 61616

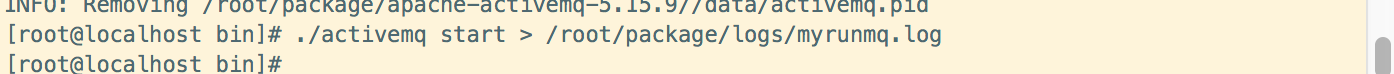


Lsof -i:61616

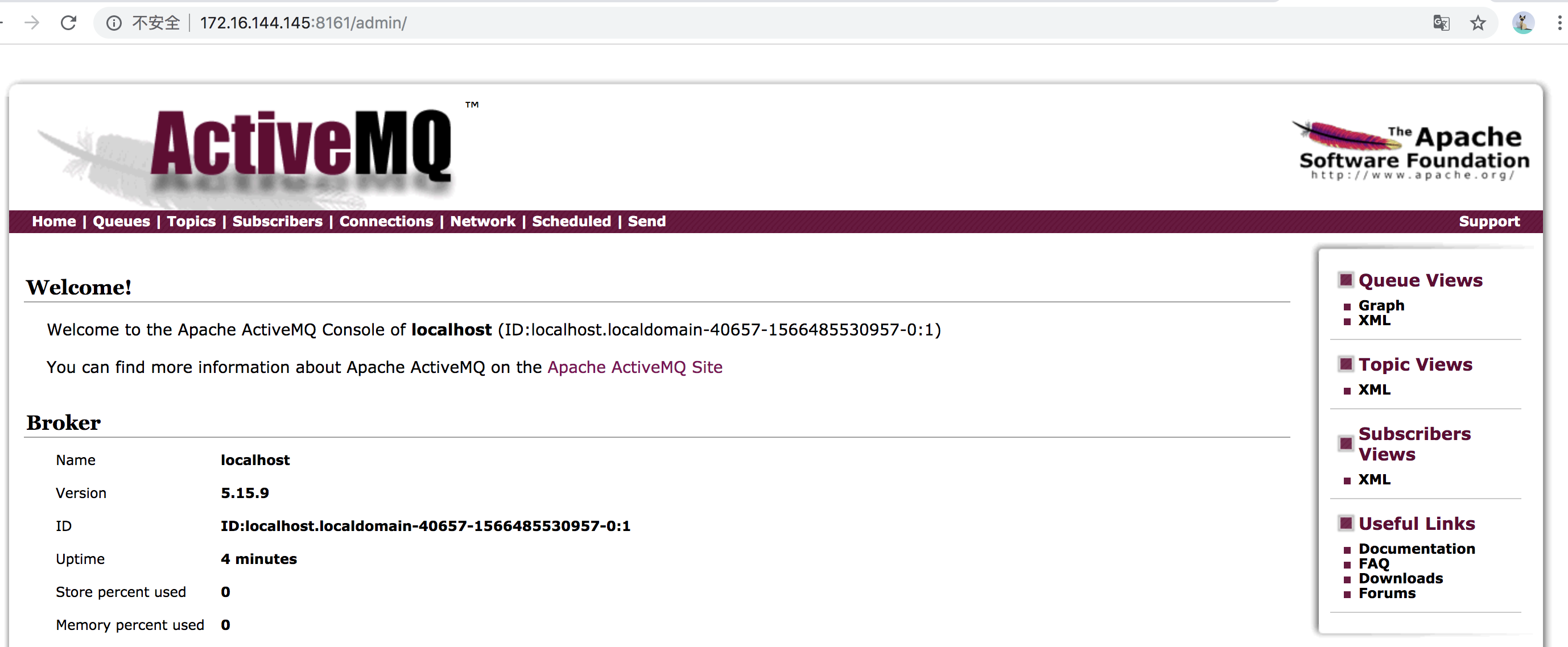


启动输出日志

./activemq start > /root/package/logs/myrunmq.log



<http://172.16.144.145:8161/admin/> admin/admin 访问客户端



# Java编码实现ActiveMQ通讯

## 3.1新建工程

|  |
| --- |
| 工程结构    Parent工程 Pom.xml  <?xml version="1.0" encoding="UTF-8"?> <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>   <groupId>com.gardenia.activemq</groupId>  <artifactId>activemq-parent</artifactId>  <version>1.0-SNAPSHOT</version>   <dependencyManagement>  <dependencies>  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-api</artifactId>  <version>1.7.26</version>  </dependency>  <dependency>  <groupId>ch.qos.logback</groupId>  <artifactId>logback-classic</artifactId>  <version>1.2.3</version>  </dependency>  <dependency>  <groupId>org.projectlombok</groupId>  <artifactId>lombok</artifactId>  <version>1.18.2</version>  </dependency>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.12</version>  </dependency>   <dependency>  <groupId>org.apache.activemq</groupId>  <artifactId>activemq-all</artifactId>  <version>5.15.9</version>  </dependency>  <dependency>  <groupId>org.apache.xbean</groupId>  <artifactId>xbean-spring</artifactId>  <version>3.16</version>  </dependency>  </dependencies>  </dependencyManagement>  </project>  Demo01 的pom文件和parent目前一样的 |

## 3.2消息发送（生产消息）

|  |
| --- |
| 1. 生产消息   package com.gardenia.activemq.study;  import org.apache.activemq.ActiveMQConnectionFactory;  import javax.jms.\*;  public class JmsProduce {   public static final String *ACTIVEMQ\_URL* = "tcp://172.16.144.145:61616";  public static final String *QUEUE\_NAME* = "queue01";    public static void main(String[] args) throws JMSException {   // 1.创建连接工厂，按照给定的url地址，采用默认用户名和密码  ActiveMQConnectionFactory activeMQConnectionFactory = new ActiveMQConnectionFactory(*ACTIVEMQ\_URL*);  // 2.通过连接工厂，获得连接connection并启动  Connection connection = activeMQConnectionFactory.createConnection();  connection.start();  // 3.创建会话session  // 两个参数，第一个叫事务/第二个叫签收  Session session = connection.createSession(false, Session.*AUTO\_ACKNOWLEDGE*);  // 4.创建目的地（具体是对列是主题topic）  Queue queue = session.createQueue(*QUEUE\_NAME*);//  // 5.创建消息的生产者  MessageProducer messageProducer = session.createProducer(queue);  // 6.循环创建多条消息  for (int i = 0; i < 3; i++) {  // 7.创建消息  TextMessage textMessage = session.createTextMessage("msg----" + i);  // 8.通过messageProducer发送Mq  messageProducer.send(textMessage);   }   messageProducer.close();  session.close();  connection.close();     }  }  发送成功后，在后台可以看到生产的待消费的消息 |

## 3.3消息接收（消费消息）

|  |
| --- |
| 1. 接收消息   package com.gardenia.activemq.study;  import org.apache.activemq.ActiveMQConnectionFactory;  import javax.jms.\*;  public class JmsConsumer {   public static final String *ACTIVEMQ\_URL* = "tcp://172.16.144.145:61616";  public static final String *QUEUE\_NAME* = "queue01";    public static void main(String[] args) throws JMSException {   // 1.创建连接工厂，按照给定的url地址，采用默认用户名和密码  ActiveMQConnectionFactory activeMQConnectionFactory = new ActiveMQConnectionFactory(*ACTIVEMQ\_URL*);  // 2.通过连接工厂，获得连接connection并启动  Connection connection = activeMQConnectionFactory.createConnection();  connection.start();  // 3.创建会话session  // 两个参数，第一个叫事务/第二个叫签收  Session session = connection.createSession(false, Session.*AUTO\_ACKNOWLEDGE*);  // 4.创建目的地（具体是对列是主题topic）  Queue queue = session.createQueue(*QUEUE\_NAME*);//  // 5.创建消息的消费者  MessageConsumer consumer = session.createConsumer(queue);  // 6.循环创建多条消息  while (true) {  TextMessage textMessage = (TextMessage) consumer.receive();  if (null != textMessage) {  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*消息：" + textMessage.getText());  } else {  break;  }   }  consumer.close();  session.close();  connection.close();    }  }  2. |

# JMS规范和落地产品

# ActiveMQ的Broker