**MQTT**

EMQ X (Erlang/Enterprise/Elastic MQTT Broker) 是基于 Erlang/OTP 平台开发的开源物联网 MQTT 消息服务器。Erlang/OTP 是出色的软实时(Soft-Realtime)、低延时(Low-Latency)、分布式(Distributed) 的语言平台。MQTT 是轻量的(Lightweight)、发布订阅模式(PubSub) 的物联网消息协议。

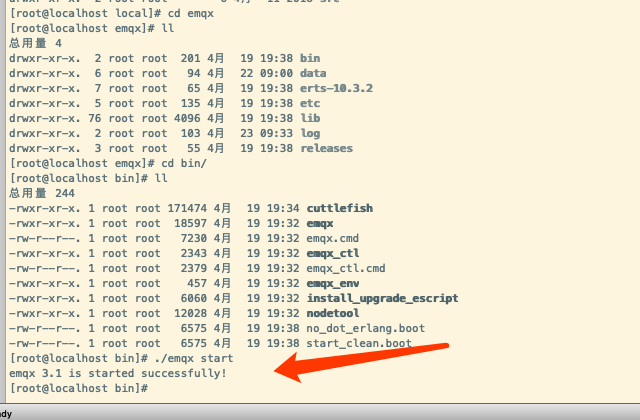
MQTT 是基于 发布(Publish)/订阅(Subscribe) 模式来进行通信及数据交换的，与 HTTP 的 请求(Request)/应答(Response) 的模式有本质的不同。

# 安装

下载地址： <https://www.emqx.io/downloads/broker?osType=Linux>

解压后即可使用

进入bin目录进行启动即可



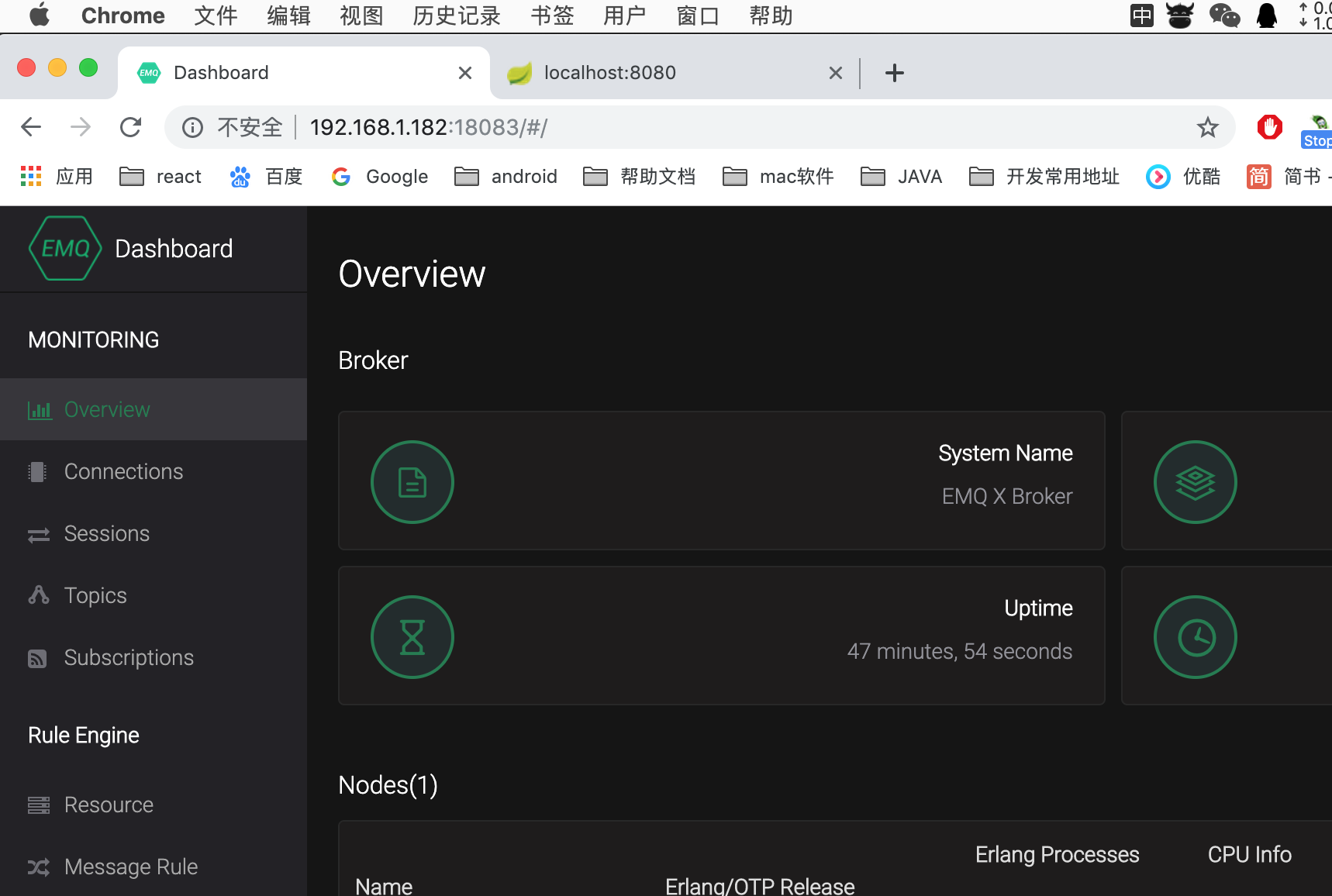
表示启动成功。

其他命令：



# 查看管理界面

控制台地址: [http://127.0.0.1:18083](http://127.0.0.1:18083/)，默认用户名: admin，密码：public



进入及表示成功。

# 订阅

需要先进行订阅，然后发布的消息才能收到。

加入依赖

|  |
| --- |
| <**dependency**>  <**groupId**>org.fusesource.mqtt-client</**groupId**>  <**artifactId**>mqtt-client</**artifactId**>  <**version**>1.12</**version**> </**dependency**>  <**dependency**>  <**groupId**>com.hivemq</**groupId**>  <**artifactId**>hivemq-mqtt-client</**artifactId**>  <**version**>1.0.1</**version**> </**dependency**>  <**dependency**>  <**groupId**>org.eclipse.paho</**groupId**>  <**artifactId**>org.eclipse.paho.client.mqttv3</**artifactId**>  <**version**>1.2.1</**version**> </**dependency**>  <**dependency**>  <**groupId**>org.apache.activemq</**groupId**>  <**artifactId**>activemq-all</**artifactId**>  <**version**>5.15.9</**version**> </**dependency**> |

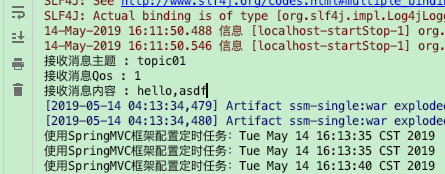
忘记是哪个依赖了，好像是倒数第二个依赖。

|  |
| --- |
| **package** com.zengqiang.utils;  **import** org.eclipse.paho.client.mqttv3.MqttClient; **import** org.eclipse.paho.client.mqttv3.MqttConnectOptions; **import** org.eclipse.paho.client.mqttv3.MqttTopic; **import** org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;  **import** javax.servlet.ServletContextEvent; **import** javax.servlet.ServletContextListener; **import** java.util.concurrent.ScheduledExecutorService;  */\*\*  \** ***@Author:*** *iszengziqiang@163.com  \** ***@Date:*** *2019-04-23 11:08  \** ***@Version:*** *1.0  \** ***@Desc: TODO*** *\*\*/* **public class** SubscribeListener **implements** ServletContextListener {   **public static final** String ***HOST*** = **"tcp://192.168.1.182:1883"**;  **public static final** String ***TOPIC*** = **"topic01"**;  **private static final** String ***clientid*** = **"client01"**;  **private** MqttClient **client**;  **private** MqttConnectOptions **options**;  **private** String **userName** = **""**;  **private** String **passWord** = **""**;   **private** ScheduledExecutorService **scheduler**;   @Override  **public void** contextInitialized(ServletContextEvent sce) {  **try** {  *// host为主机名，clientid即连接MQTT的客户端ID，一般以唯一标识符表示，MemoryPersistence设置clientid的保存形式，默认为以内存保存* **client** = **new** MqttClient(***HOST***, ***clientid***, **new** MemoryPersistence());  *// MQTT的连接设置* **options** = **new** MqttConnectOptions();  *// 设置是否清空session,这里如果设置为false表示服务器会保留客户端的连接记录，这里设置为true表示每次连接到服务器都以新的身份连接* **options**.setCleanSession(**true**);  *// 设置连接的用户名* **options**.setUserName(**userName**);  *// 设置连接的密码* **options**.setPassword(**passWord**.toCharArray());  *// 设置超时时间 单位为秒* **options**.setConnectionTimeout(10);  *// 设置会话心跳时间 单位为秒 服务器会每隔1.5\*20秒的时间向客户端发送个消息判断客户端是否在线，但这个方法并没有重连的机制* **options**.setKeepAliveInterval(20);  *// 设置回调* **client**.setCallback(**new** PushCallback());  MqttTopic topic = **client**.getTopic(***TOPIC***);  *//setWill方法，如果项目中需要知道客户端是否掉线可以调用该方法。设置最终端口的通知消息* **options**.setWill(topic, **"close"**.getBytes(), 2, **true**);   **client**.connect(**options**);  *//订阅消息* **int**[] Qos = {1};  String[] topic1 = {***TOPIC***};  **client**.subscribe(topic1, Qos);   } **catch** (Exception e) {  e.printStackTrace();  }  }   @Override  **public void** contextDestroyed(ServletContextEvent sce) {   } } |

订阅模块代码

|  |
| --- |
| **package** com.zengqiang.utils;  **import** org.apache.log4j.Logger; **import** org.eclipse.paho.client.mqttv3.IMqttDeliveryToken; **import** org.eclipse.paho.client.mqttv3.MqttCallback; **import** org.eclipse.paho.client.mqttv3.MqttMessage;  */\*\*  \** ***@Author:*** *iszengziqiang@163.com  \** ***@Date:*** *2019-04-23 10:39  \** ***@Version:*** *1.0  \** ***@Desc: TODO*** *\*\*/* **public class** PushCallback **implements** MqttCallback {   Logger **logger** = Logger.*getLogger*(PushCallback.**class**);   **public void** connectionLost(Throwable cause) {  *// 连接丢失后，一般在这里面进行重连* **logger**.info(**"连接出错："**+cause.getMessage());  System.***out***.println(**"连接断开，可以做重连"**);  }   **public void** deliveryComplete(IMqttDeliveryToken token) {  System.***out***.println(**"deliveryComplete---------"** + token.isComplete());  }   **public void** messageArrived(String topic, MqttMessage message) **throws** Exception {  *// subscribe后得到的消息会执行到这里面* **logger**.info(**"接收的主题："** + topic + **"====消息内容："** + **new** String(message.getPayload()));  System.***out***.println(**"接收消息主题 : "** + topic);  System.***out***.println(**"接收消息Qos : "** + message.getQos());  System.***out***.println(**"接收消息内容 : "** + **new** String(message.getPayload()));  }  } |
| <**listener**>  <**listener-class**>com.zengqiang.utils.SubscribeListener</**listener-class**> </**listener**> |

好了，现在可以启动订阅模块了



启动后会去服务器查看是否有未消费的消息。

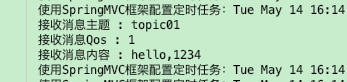
现在来进行发布模块编写，编写完成即可进行发布

# 发布

|  |
| --- |
| **package** com.zengqiang.utils;  **import** org.eclipse.paho.client.mqttv3.\*; **import** org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;  */\*\*  \** ***@Author:*** *iszengziqiang@163.com  \** ***@Date:*** *2019-04-23 09:31  \** ***@Version:*** *1.0  \** ***@Desc: TODO*** *\*\*/* **public class** PubMain {   *//tcp://MQTT安装的服务器地址:MQTT定义的端口号* **public static final** String ***HOST*** = **"tcp://192.168.1.182:1883"**;  *//定义一个主题* **public static final** String ***TOPIC*** = **"topic01"**;  *//定义MQTT的ID，可以在MQTT服务配置中指定* **private static final** String ***clientid*** = **"server01"**;   **private** MqttClient **client**;  **private** MqttTopic **topic11**;  **private** String **userName** = **""**;  **private** String **passWord** = **""**;   **private** MqttMessage **message**;   */\*\*  \* 构造函数  \*  \** ***@throws*** *MqttException  \*/* **public** PubMain() **throws** MqttException {  *// MemoryPersistence设置clientid的保存形式，默认为以内存保存* **client** = **new** MqttClient(***HOST***, ***clientid***, **new** MemoryPersistence());  connect();  }   */\*\*  \* 用来连接服务器  \*/* **private void** connect() {  MqttConnectOptions options = **new** MqttConnectOptions();  options.setCleanSession(**false**);  options.setUserName(**userName**);  options.setPassword(**passWord**.toCharArray());  *// 设置超时时间* options.setConnectionTimeout(10);  *// 设置会话心跳时间* options.setKeepAliveInterval(20);  **try** {  **client**.setCallback(**new** PushCallback());  **client**.connect(options);   **topic11** = **client**.getTopic(***TOPIC***);  } **catch** (Exception e) {  e.printStackTrace();  }  }   */\*\*  \** ***@param topic*** *\** ***@param message*** *\** ***@throws*** *MqttPersistenceException  \** ***@throws*** *MqttException  \*/* **public void** publish(MqttTopic topic, MqttMessage message) **throws** MqttPersistenceException,  MqttException {  MqttDeliveryToken token = topic.publish(message);  token.waitForCompletion();  System.***out***.println(**"message is published completely! "** + token.isComplete());  }   **public static void** main(String[] args) {  **try** {  PubMain server = **new** PubMain();   server.**message** = **new** MqttMessage();  server.**message**.setQos(1);  server.**message**.setRetained(**true**);  server.**message**.setPayload(**"hello,1234"**.getBytes());  server.publish(server.**topic11**, server.**message**);  System.***out***.println(server.**message**.isRetained() + **"------ratained状态"**);   } **catch** (MqttException e) {  e.printStackTrace();  }  } } |

这里我们发布了一条消息**hello,1234**

进行发布。



我们将看到订阅模块收到了消息。

APP和后台可以双向订阅，即可实现消息推送功能。