RabbitMQ-集群监控

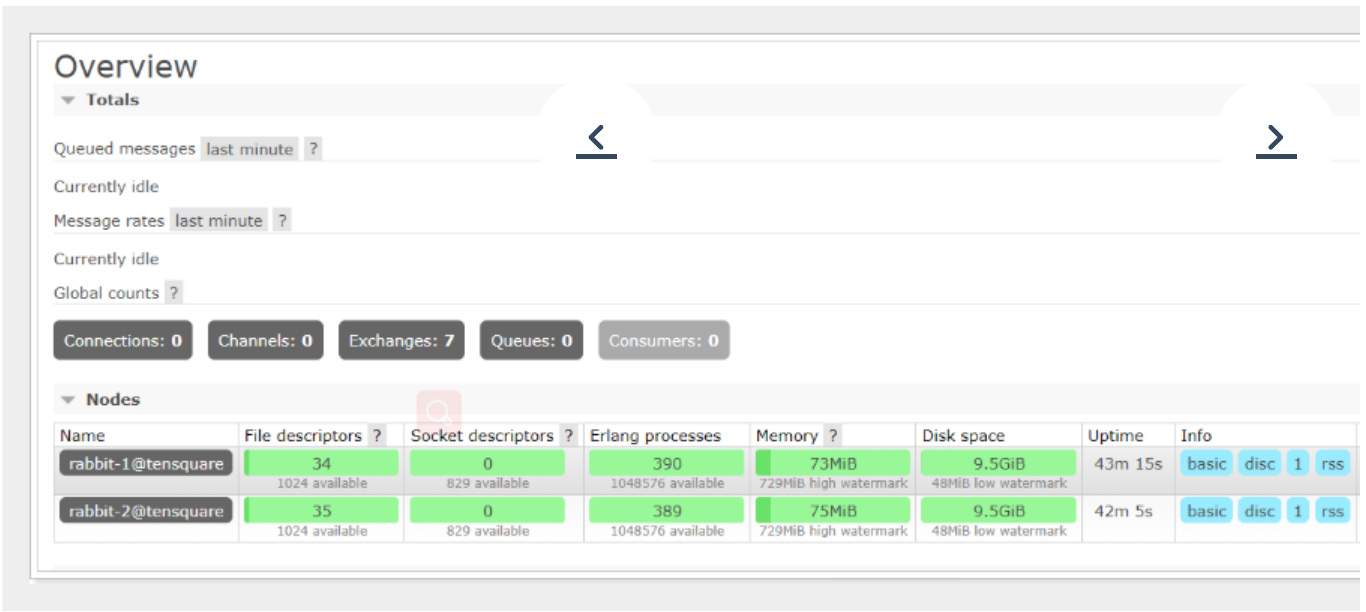
飞哥 VIP 分类：学习笔记 创建时间：2021/03/06 22:00 ☑字体 ☐皮肤

集群监控

在广大的互联网行业中RabbitMQ几乎都会有集群,那么对于集群的监控就成了企业生态中必不可少的一环。接下来
控。

1、管理界面监控

> 管理界面监控需要我们开启对应的插件(rabbitmq-plugins enable rabbitmq_management)
> 然后访问http://ip:15672



> 在管理控制台我们就可以直观的看到集群中的每一个节点是否正常,如果为红色则表示节点挂掉了,同时可以很方
> 存、磁盘等相关的信息，使用起来也是非常方便的。但是遗憾的该功能做的比较简陋,没有告警等一些列的个性化
> 入到公司其他的监控系统统一管理也是很难做到的,所以扩展性不强，一般在小型企业的小集群中使用。

2、tracing日志监控

> 对于企业级的应用开发来讲,我们通常都会比较关注我们的消息,甚至很多的场景把消息的可靠性放在第一位,但是
> 现消息异常丢失或者客户端无法发送消息等异常情况,此时为了帮助开发人员快速的定位问题,我们就可以对消息
> 控,而tracing日志监控插件帮我们很好的实现了该功能
> 消息中心的消息追踪需要使用Trace实现，Trace是Rabbitmq用于记录每一次发送的消息，方便使用Rabbitmq
> 通过插件形式提供可视化界面。Trace启动后会自动创建系统Exchange：amq.rabbitmq.trace ,每个队列会自
> 后发送到队列的消息都会记录到Trace日志。

消息追踪启用与查看

以下是trace的相关命令和使用（要使用需要先rabbitmq启用插件，再打开开关才能使用）：

命令集	描述
rabbitmq-plugins list	查看插件列表

命令集	描述
rabbitmq-plugins enable rabbitmq_tracing	rabbitmq启用trace插件
rabbitmqctl trace_on	打开trace的开关
rabbitmqctl trace_on -p itcast	打开trace的开关(itcast为需要日志追踪的vhost)
rabbitmqctl trace_off	关闭trace的开关
rabbitmq-plugins disable rabbitmq_tracing	rabbitmq关闭Trace插件
rabbitmqctl set_user_tags heima administrator	只有administrator的角色才能查看日志界面

安装插件并开启 trace_on 之后，会发现多个 exchange：amq.rabbitmq.trace ， 类型为：

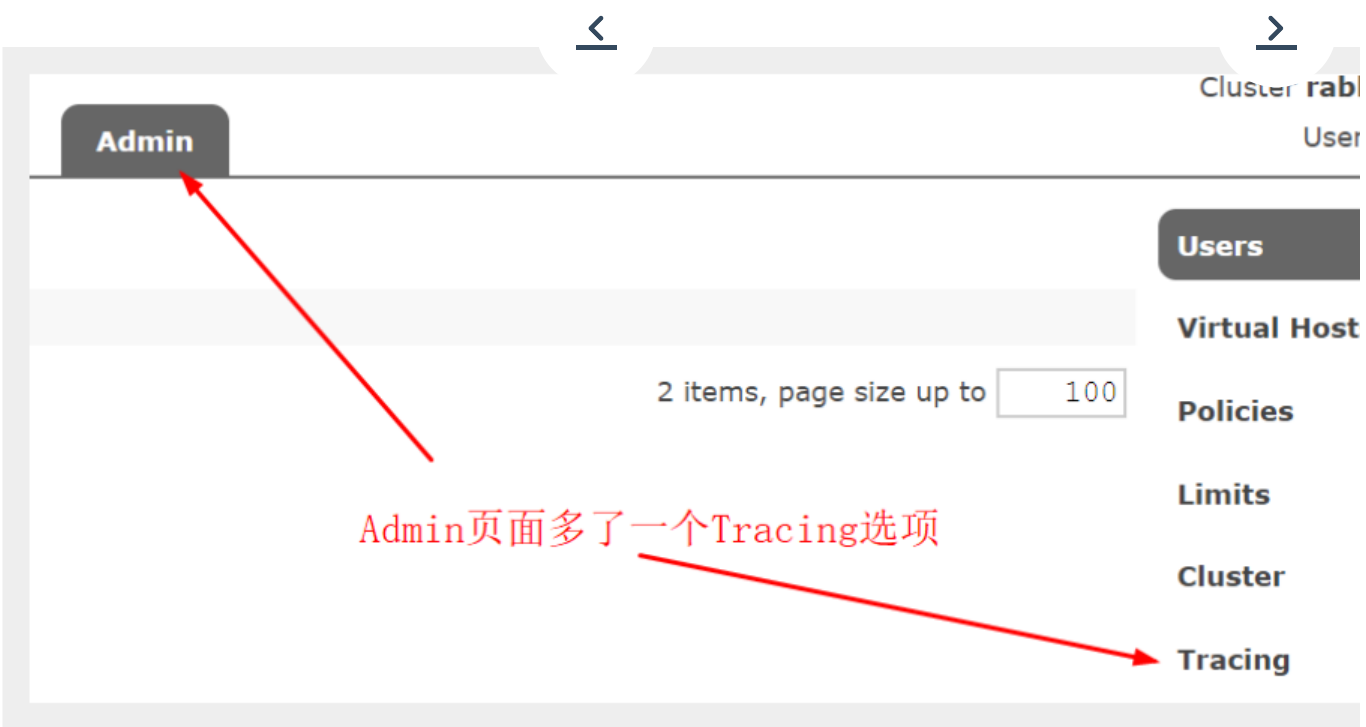
amq.rabbitmq.trace	topic	D I		

2、日志追踪

1、发送消息

```
1. rabbitTemplate.convertAndSend("spring_queue", "只发队列spring_queue的消息--01。");
```

2、查看trace



点击Tracing查看Trace log files

3、点击Tracing查看Trace log files

4、点击xuexiangban-trace.log确认消息轨迹正确性

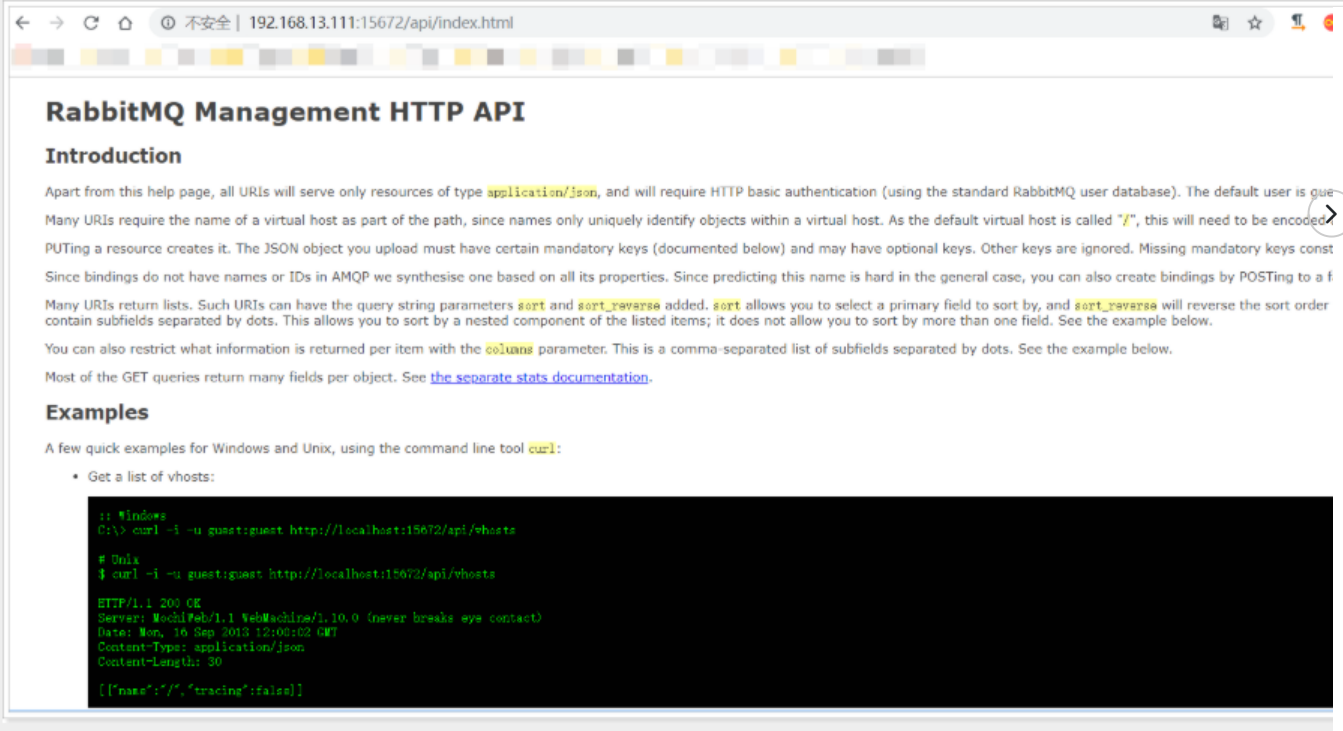
```
Node:         rabbit@Server-node      节点信息
Connection:   127.0.0.1:57384 -> 127.0.0.1:5672
Virtual host: itcast        虚拟机信息
User:         heima         用户信息
Channel:      1
Exchange:
Routing keys: [<<"spring_queue">>]
Routed queues: [<<"spring_queue">>]
Properties:   [{<<"priority">>,signedint,0},        消息属性
               {<<"delivery_mode">>,signedint,2},
               {<<"headers">>,table,[]},
               {<<"content_encoding">>,longstr,<<"UTF-8">>},
               {<<"content_type">>,longstr,<<"text/plain">>}]
Payload:
只发队列spring_queue的消息--01。        消息内容
```


3、定制自己的监控系统

> RabbitMQ提供了很丰富的restful风格的api接口,我们可以通过这些接口得到对应的集群数据,此时我们就可以

HTTP API URL	HTTP 请求类型	接口含义
/api/connections	GET	获取当前RabbitMQ集群下所有打开的
/api/nodes	GET	获取当前RabbitMQ集群下所有节点实
/api/vhosts/{vhost}/connections	GET	获取某一个虚拟机主机下的所有打开的
/api/connections/{name}/channels	GET	获取某一个连接下所有的管道
/api/vhosts/{vhost}/channels	GET	获取某一个虚拟机主机下的管道信息
/api/consumers/{vhost}	GET	获取某一个虚拟机主机下的所有消费者
/api/exchanges/{vhost}	GET	获取某一个虚拟机主机下面的所有交换
/api/queues/{vhost}	GET	获取某一个虚拟机主机下的所有队列信
/api/users	GET	获取集群中所有的用户信息
/api/users/{name}	GET/PUT/DELETE	获取/更新/删除指定用户信息
/api/users/{user}/permissions	GET	获取当前指定用户的所有权限信息
/api/permissions/{vhost}/{user}	GET/PUT/DELETE	获取/更新/删除指定虚拟主机下特定用
/api/exchanges/{vhost}/{name}/publish	POST	在指定的虚拟机主机和交换器上发布一
/api/queues/{vhost}/{name}/get	POST	在指定虚拟机主机和队列名中获取消息 同时该动作会修改队列状态
/api/healthchecks/node/{node}	GET	获取指定节点的健康检查状态

> 更多API的相关信息和描述可以访问http://ip:15672/api/



> 接下来我们使用RabbitMQ Http API接口来获取集群监控数据

HttpClient以及Jackson的相关Jar

```
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.3.6</version>
</dependency>
<dependency>
 <groupId>com.fasterxml.jackson.core</groupId>
 <artifactId>jackson-databind</artifactId>
 <version>2.7.4</version>
</dependency>
<dependency>
 <groupId>com.fasterxml.jackson.core</groupId>
 <artifactId>jackson-annotations</artifactId>
 <version>2.7.4</version>
</dependency>
<dependency>
 <groupId>com.fasterxml.jackson.core</groupId>
 <artifactId>jackson-core</artifactId>
 <version>2.7.4</version>
</dependency>
```

创建MonitorRabbitMQ类实现具体的代码

```java
package com.xuexiangban.rabbitmq;

import com.fasterxml.jackson.databind.DeserializationFeature;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.apache.http.HttpEntity;
import org.apache.http.auth.UsernamePasswordCredentials;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.auth.BasicScheme;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

import java.io.IOException;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;

/**
 * RabbitMQ的监控
 */
public class MonitorRabbitMQ {
    //RabbitMQ的HTTP API——获取集群各个实例的状态信息，ip替换为自己部署相应实例的
    private static String RABBIT_NODES_STATUS_REST_URL =
"http://192.168.13.111:15672/api/nodes";
    //RabbitMQ的HTTP API——获取集群用户信息，ip替换为自己部署相应实例的
    private static String RABBIT_USERS_REST_URL = "http://192.168.13.111:156
    //rabbitmq的用户名
    private static String R    SER_NAME = "guest";
    //rabbitmq的密码
    private static String RABBIT_USER_PWD = "guest";

    public static void main(String[] args) {
        try {
            //step1.获取rabbitmq集群各个节点实例的状态信息
            Map<String, ClusterStatus> clusterMap =
                    fetchRabbtMQClusterStatus(RABBIT_NODES_STATUS_REST_URL, R
RABBIT_USER_PWD);

            //step2.打印输出各个节点实例的状态信息
            for (Map.Entry entry : clusterMap.entrySet()) {
                System.out.println(entry.getKey() + " : " + entry.getValue()
            }

            //step3.获取rabbitmq集群用户信息
            Map<String, User> userMap =
                    fetchRabbtMQUsers(RABBIT_USERS_REST_URL, RABBIT_USER_NAMI

            //step4.打印输出rabbitmq集群用户信息
            for (Map.Entry entry : userMap.entrySet()) {
                System.out.println(entry.getKey() + " : " + entry.getValue()
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static Map<String, ClusterStatus> fetchRabbtMQClusterStatus(Strin
username, String password) throws IOException {
        Map<String, ClusterStatus> clusterStatusMap = new HashMap<String, Cl
        String nodeData = getData(url, username, password);
        JsonNode jsonNode = null;
        try {
            jsonNode = JsonUtil.toJsonNode(nodeData);
        } catch (IOException e) {
            e.printStackTrace();
```

```
66.         }
67.         Iterator<JsonNode> iterator = jsonNode.iterator();
68.         while (iterator.hasNext()) {
69.             JsonNode next = iterator.next();
70.             ClusterStatus status = new ClusterStatus();
71.             status.setDiskFree(next.get("disk_free").asLong());
72.             status.setFdUsed(next.get("fd_used").asLong());
73.             status.setMemoryUsed(next.get("mem_used").asLong());
74.             status.setProcUsed(next.get("proc_used").asLong());
75.             status.setSocketUsed(next.get("sockets_used").asLong());
76.             clusterStatusMap.put(next.get("name").asText(), status);
77.         }
78.         return clusterStatusMap;
79.     }
80.
81.     public static Map<String, User> fetchRabbtMQUsers(String url, String user
    password) throws IOException {
82.         Map<String, User> userMap = new HashMap<String, User>();
83.         String nodeData = getData(url, username, password);
84.         JsonNode jsonNode = null;
85.         try {
86.             jsonNode = JsonUtil.toJsonNode(nodeData);
87.         } catch (IOException e) {
88.             e.printStackTrace();
89.         }
90.         Iterator<JsonNode> iterator = jsonNode.iterator();
91.         while (iterator.hasNext()) {
92.             JsonNode next = iterator.next();
93.             User user = new User();
94.             user.setName(ne      ("name").asText());
95.             user.setTags(ne    "tags").asText());
96.             userMap.put(next.g    ("name").asText(), user);
97.         }
98.         return userMap;
99.     }
100.
101.    public static String getData(String url, String username, String passwor
    IOException {
102.         CloseableHttpClient httpClient = HttpClients.createDefault();
103.         UsernamePasswordCredentials creds = new UsernamePasswordCredentials(
104.         HttpGet httpGet = new HttpGet(url);
105.         httpGet.addHeader(BasicScheme.authenticate(creds, "UTF-8", false));
106.         httpGet.setHeader("Content-Type", "application/json");
107.         CloseableHttpResponse response = httpClient.execute(httpGet);
108.
109.         try {
110.             if (response.getStatusLine().getStatusCode() != 200) {
111.                 System.out.println("call http api to get rabbitmq data retur
    response.getStatusLine().getStatusCode() + ", url: " + url);
112.             }
113.             HttpEntity entity = response.getEntity();
114.             if (entity != null) {
115.                 return EntityUtils.toString(entity);
116.             }
117.         } finally {
118.             response.close();
119.         }
120.
121.         return null;
122.     }
123.
124.     public static class JsonUtil {
125.         private static ObjectMapper objectMapper = new ObjectMapper();
126.
127.         static {
128.             objectMapper.disable(DeserializationFeature.FAIL_ON_UNKNOWN_PROPI
129.             //objectMapper.disable(SerializationFeature.FAIL_ON_EMPTY_BEANS)
130.         }
```

```
131.
132.         public static JsonNode toJsonNode(String jsonString) throws IOException {
133.             return objectMapper.readTree(jsonString);
134.         }
135.     }
136.
137.     public static class User {
138.         private String name;
139.         private String tags;
140.
141.         @Override
142.         public String toString() {
143.             return "User{" +
144.                     "name=" + name +
145.                     ", tags=" + tags +
146.                     '}';
147.         }
148.         //GET/SET方法省略
149.
150.         public String getName() {
151.             return name;
152.         }
153.
154.         public void setName(String name) {
155.             this.name = name;
156.         }
157.
158.         public String getTags() {
159.             return tags;
160.         }
161.
162.         public void setTags(String tags) {
163.             this.tags = tags;
164.         }
165.     }
166.
167.     public static class ClusterStatus {
168.         private long diskFree;
169.         private long diskLimit;
170.         private long fdUsed;
171.         private long fdTotal;
172.         private long socketUsed;
173.         private long socketTotal;
174.         private long memoryUsed;
175.         private long memoryLimit;
176.         private long procUsed;
177.         private long procTotal;
178.
179.         // 此处省略了Getter和Setter方法
180.         public long getDiskFree() {
181.             return diskFree;
182.         }
183.
184.         public void setDiskFree(long diskFree) {
185.             this.diskFree = diskFree;
186.         }
187.
188.         public long getDiskLimit() {
189.             return diskLimit;
190.         }
191.
192.         public void setDiskLimit(long diskLimit) {
193.             this.diskLimit = diskLimit;
194.         }
195.
196.         public long getFdUsed() {
197.             return fdUsed;
198.         }
```

```
199.
200.        public void setFdUsed(long fdUsed) {
201.            this.fdUsed = fdUsed;
202.        }
203.
204.        public long getFdTotal() {
205.            return fdTotal;
206.        }
207.
208.        public void setFdTotal(long fdTotal) {
209.            this.fdTotal = fdTotal;
210.        }
211.
212.        public long getSocketUsed() {
213.            return socketUsed;
214.        }
215.
216.        public void setSocketUsed(long socketUsed) {
217.            this.socketUsed = socketUsed;
218.        }
219.
220.        public long getSocketTotal() {
221.            return socketTotal;
222.        }
223.
224.        public void setSocketTotal(long socketTotal) {
225.            this.socketTotal = socketTotal;
226.        }
227.
228.        public long getMemor      ') {
229.            return memoryUs
230.        }
231.
232.        public void setMemoryUsed(long memoryUsed) {
233.            this.memoryUsed = memoryUsed;
234.        }
235.
236.        public long getMemoryLimit() {
237.            return memoryLimit;
238.        }
239.
240.        public void setMemoryLimit(long memoryLimit) {
241.            this.memoryLimit = memoryLimit;
242.        }
243.
244.        public long getProcUsed() {
245.            return procUsed;
246.        }
247.
248.        public void setProcUsed(long procUsed) {
249.            this.procUsed = procUsed;
250.        }
251.
252.        public long getProcTotal() {
253.            return procTotal;
254.        }
255.
256.        public void setProcTotal(long procTotal) {
257.            this.procTotal = procTotal;
258.        }
259.
260.        @Override
261.        public String toString() {
262.            return "ClusterStatus{" +
263.                    "diskFree=" + diskFree +
264.                    ", diskLimit=" + diskLimit +
265.                    ", fdUsed=" + fdUsed +
266.                    ", fdTotal=" + fdTotal +
```

```
267.                     ", socketUsed=" + socketUsed +
268.                     ", socketTotal=" + socketTotal +
269.                     ", memoryUsed=" + memoryUsed +
270.                     ", memoryLimit=" + memoryLimit +
271.                     ", procUsed=" + procUsed +
272.                     ", procTotal=" + procTotal +
273.                     '}';
274.         }
275.
276.     }
277. }
```


启动测试

4、Zabbix 监控RabbitMQ

Zabbix是一个基于WEB界面提供分布式系统监视以及网络监视功能的企业级开源解决方案,他也可以统,同时提供预警等功能，但是由于其搭建配置较高一般都是由运维人员负责搭建,感兴趣的同学可以 https://www.zabbix.com/ 官网进行了解学习。