链滴

# springcloud(六)：熔断监控 Turbine

作者：911708498

原文链接：https://ld246.com/article/1543460802697

来源网站：链滴

在复杂的分布式系统中，相同服务的节点经常需要部署上百甚至上千个，很多时候，运维人员希望能把相同服务的节点状态以一个整体集群的形式展现出来，这样可以更好的把握整个系统的状态。 为此Netflix提供了一个开源项目（Turbine）来提供把多个hystrix.stream的内容聚合为一个数据源供Dasboard展示。

1、添加依赖

```
<dependencies>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-turbine</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-netflix-turbine</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-actuator</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-hystrix-dashboard</artifactId>
    </dependency>
</dependencies>
```

2、配置文件

```
spring.application.name=hystrix-dashboard-turbine
server.port=8001
turbine.appConfig=node01,node02
turbine.aggregator.clusterConfig= default
turbine.clusterNameExpression= new String("default")
 eureka.client.serviceUrl.defaultZone=http://localhost:8000/eureka/
```

turbine.appConfig ：配置Eureka中的serviceId列表，表明监控哪些服务

turbine.aggregator.clusterConfig ：指定聚合哪些集群，多个使用”,”分割，默认为default。可用http://.../turbine.stream?cluster={clusterConfig之一}访问

turbine.clusterNameExpression ： 1. clusterNameExpression指定集群名称，默认表达式appNam
；此时：turbine.aggregator.clusterConfig需要配置想要监控的应用名称；2. 当clusterNameExpres
ion: default时，turbine.aggregator.clusterConfig可以不写，因为默认就是default；3. 当clusterN
meExpression: metadata[‘cluster’]时，假设想要监控的应用配置了eureka.instance.metadata-
ap.cluster: ABC，则需要配置，同时turbine.aggregator.clusterConfig: ABC

3、启动类

启动类添加@EnableTurbine，激活对Turbine的支持

```
@SpringBootApplication
@EnableHystrixDashboard
@EnableTurbine
public class DashboardApplication {

    public static void main(String[] args) {
        SpringApplication.run(DashboardApplication.class, args);
    }

}
```

到此Turbine（hystrix-dashboard-turbine）配置完成

4、测试

在示例项目spring-cloud-consumer-hystrix基础上修改为两个服务的调用者spring-cloud-consume-node1和spring-cloud-consumer-node2

spring-cloud-consumer-node1项目改动如下： application.properties文件内容

```
spring.application.name=node01
server.port=9001
feign.hystrix.enabled=true

eureka.client.serviceUrl.defaultZone=http://localhost:8000/eureka/
```

spring-cloud-consumer-node2项目改动如下： application.properties文件内容

```
spring.application.name=node02
server.port=9002
feign.hystrix.enabled=true

eureka.client.serviceUrl.defaultZone=http://localhost:8000/eureka/
```

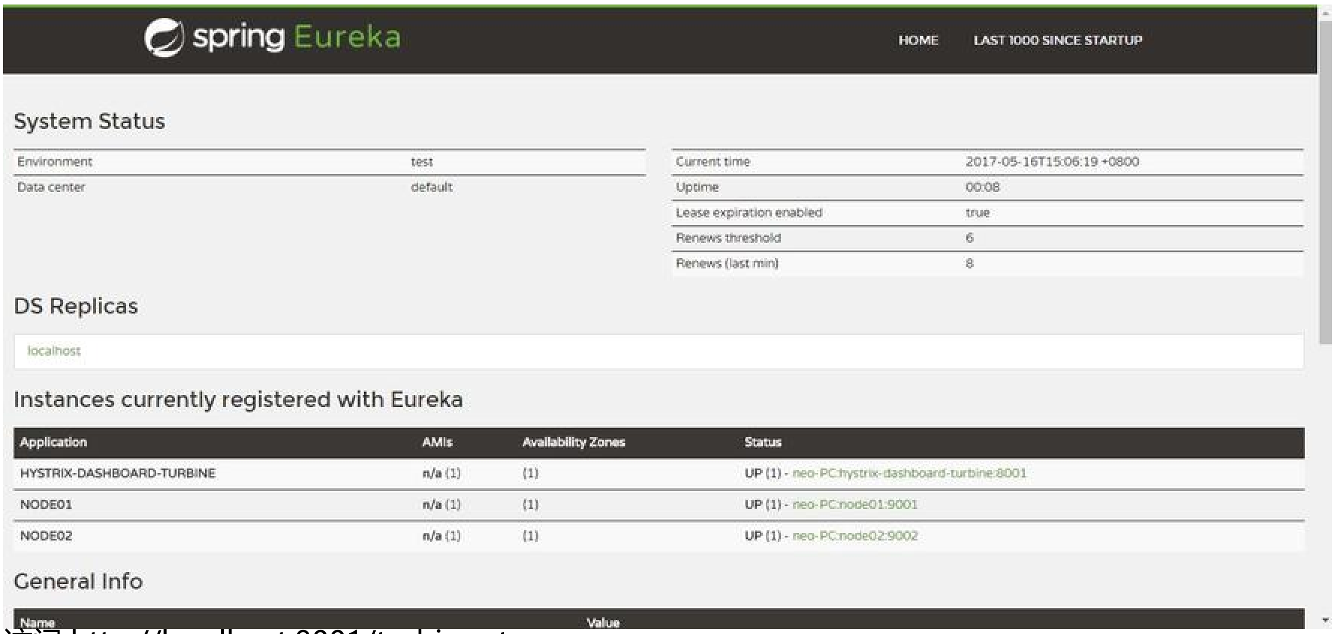
HelloRemote类修改：

```
@FeignClient(name= "spring-cloud-producer2", fallback = HelloRemoteHystrix.class)
public interface HelloRemote {

    @RequestMapping(value = "/hello")
    public String hello2(@RequestParam(value = "name") String name);

}
```

对应的HelloRemoteHystrix和ConsumerController类跟随修改，具体查看代码

修改完毕后，依次启动spring-cloud-eureka、spring-cloud-consumer-node1、spring-cloud-conumer-node1、hystrix-dashboard-turbine（Turbine）

打开eureka后台可以看到注册了三个服务：



System Status

| | |
|---|---|
| Environment | test |
| Data center | default |

| | |
|---|---|
| Current time | 2017-05-16T15:06:19 +0800 |
| Uptime | 00:08 |
| Lease expiration enabled | true |
| Renews threshold | 6 |
| Renews (last min) | 8 |

DS Replicas

localhost

Instances currently registered with Eureka

| Application | AMIs | Availability Zones | Status |
|---|---|---|---|
| HYSTRIX-DASHBOARD-TURBINE | n/a (1) | (1) | UP (1) - neo-PC:hystrix-dashboard-turbine:8001 |
| NODE01 | n/a (1) | (1) | UP (1) - neo-PC:node01:9001 |
| NODE02 | n/a (1) | (1) | UP (1) - neo-PC:node02:9002 |

General Info

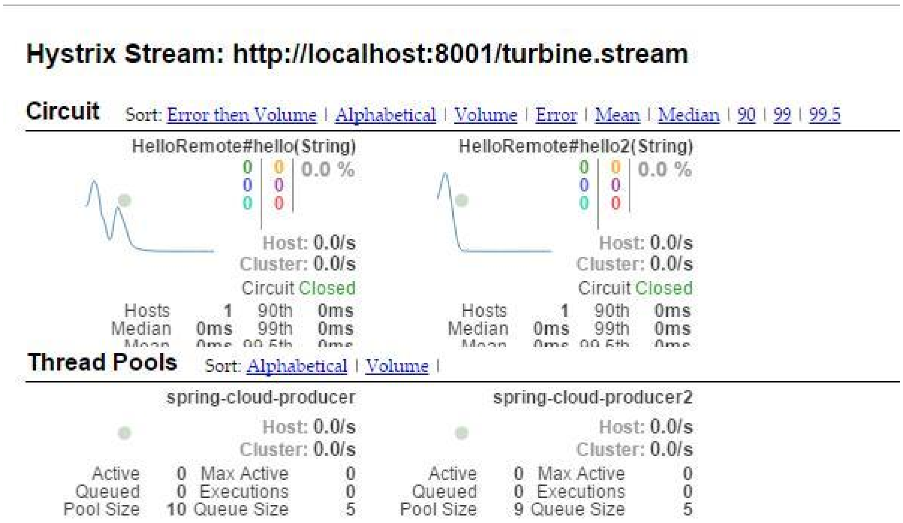
| Name | Value |
|---|---|

访问 http://localhost:8001/turbine.stream

返回：

```
: ping
data: {"reportingHostsLast10Seconds":1,"name":"meta","type":"meta","timestamp":149492198839}
```

并且会不断刷新以获取实时的监控数据，说明和单个的监控类似，返回监控项目的信息。进行图形化控查看，输入：http ://localhost:8001/hystrix，返回酷酷的小熊界面，输入： http ://localhost:801/turbine.stream，然后点击 Monitor Stream ,可以看到出现了俩个监控列表



Hystrix Stream: http://localhost:8001/turbine.stream

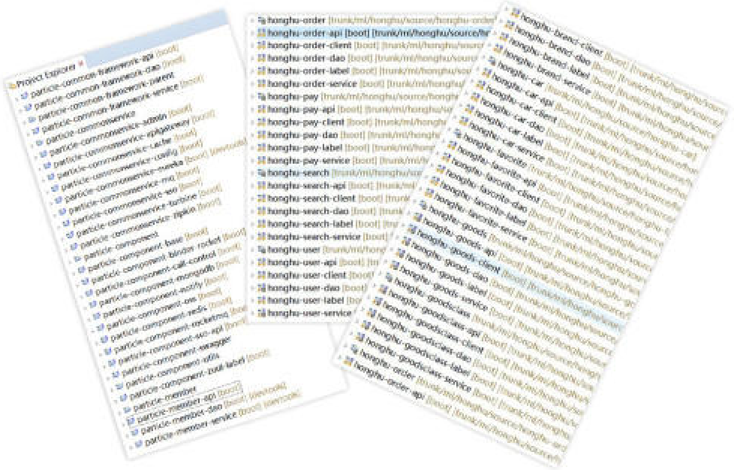
Circuit Sort: Error then Volume | Alphabetical | Volume | Error | Mean | Median | 90 | 99 | 99.5

```
HelloRemote#hello(String)              HelloRemote#hello2(String)
        0 | 0 | 0.0 %                          0 | 0 | 0.0 %
        0 | 0                                  0 | 0
        0 | 0                                  0 | 0
        Host: 0.0/s                            Host: 0.0/s
        Cluster: 0.0/s                         Cluster: 0.0/s
        Circuit Closed                         Circuit Closed
Hosts     1   90th   0ms               Hosts     1   90th   0ms
Median  0ms   99th   0ms               Median  0ms   99th   0ms
Mean    0ms 99.5th   0ms               Mean    0ms 99.5th   0ms
```

Thread Pools Sort: Alphabetical | Volume |

```
        spring-cloud-producer                  spring-cloud-producer2
        Host: 0.0/s                            Host: 0.0/s
        Cluster: 0.0/s                         Cluster: 0.0/s
Active      0  Max Active    0         Active      0  Max Active    0
Queued      0  Executions    0         Queued      0  Executions    0
Pool Size  10  Queue Size    5         Pool Size   9  Queue Size    5
```

整体代码结构如下：



```
Project Explorer
particle-common-framework-api [boot]
particle-common-framework-dao [boot]
particle-common-framework-parent
particle-common-framework-service [boot]
particle-commonservice
particle-commonservice-admin [boot]
particle-commonservice-apigateway [boot]
particle-commonservice-cache [boot]
particle-commonservice-config [boot] [devtools]
particle-commonservice-eureka [boot]
particle-commonservice-mq [boot]
particle-commonservice-sso [boot]
particle-commonservice-turbine [boot]
particle-commonservice-zipkin [boot]
particle-component
particle-component-base [boot]
particle-component-binder-rocket [boot]
particle-component-call-control [boot]
particle-component-mongodb [boot]
particle-component-notify [boot]
particle-component-oss [boot]
particle-component-redis [boot]
particle-component-rocketmq [boot]
particle-component-sso-api [boot]
particle-component-swagger
particle-component-utils
particle-component-zuul-label [boot]
particle-member
particle-member-api [boot]
particle-member-dao [boot] [devtools]
particle-member-service [boot] [devtools]
```

```
honghu-order [trunk/ml/honghu/source/honghu-order
honghu-order-api [boot] [trunk/ml/honghu/source/ho
honghu-order-client [boot] [trunk/ml/honghu/sourc
honghu-order-dao [boot] [trunk/ml/honghu/sourc
honghu-order-label [boot] [trunk/ml/honghu/sou
honghu-order-service [boot] [trunk/ml/honghu/
honghu-pay [trunk/ml/honghu/source/hongh
honghu-pay-api [boot] [trunk/ml/honghu/so
honghu-pay-client [boot] [trunk/ml/hongh
honghu-pay-dao [boot] [trunk/ml/hongh
honghu-pay-label [boot] [trunk/ml/hong
honghu-pay-service [boot] [trunk/ml/h
honghu-search [trunk/ml/honghu/so
honghu-search-api [boot] [trunk/ml
honghu-search-client [boot] [trunk
honghu-search-dao [boot] [trunk
honghu-search-label [boot] [tru
honghu-search-service [boot]
honghu-user [trunk/ml/hong
honghu-user-api [boot] [tr
honghu-user-client [boot]
honghu-user-dao [boot]
honghu-user-label [bo
honghu-user-service
```

```
honghu-brand-client [boot] [trunk/ml/honghu/sour
honghu-brand-dao [boot] [trunk/ml/honghu/source/
honghu-brand-label [boot] [trunk/ml/honghu/source
honghu-brand-service [boot] [trunk/ml/honghu/source/
honghu-car [trunk/ml/honghu/source/honghu-car]
honghu-car-api [boot] [trunk/ml/honghu/source/
honghu-car-client [boot] [trunk/ml/honghu/source
honghu-car-dao [boot] [trunk/ml/honghu/source/
honghu-car-label [boot] [trunk/ml/honghu/source
honghu-car-service [boot] [trunk/ml/honghu/source/
honghu-favorite [trunk/ml/honghu/source/honghu-
honghu-favorite-api [boot] [trunk/ml/honghu/sour
honghu-favorite-client [boot] [trunk/ml/honghu/
honghu-favorite-dao [boot] [trunk/ml/honghu/sou
honghu-favorite-label [boot] [trunk/ml/honghu/
honghu-favorite-service [boot] [trunk/ml/hongh
honghu-goods [trunk/ml/honghu/source/honghu-
honghu-goods-api [boot] [trunk/ml/honghu/sou
honghu-goods-client [boot] [trunk/ml/honghu/
honghu-goods-dao [boot] [trunk/ml/honghu/sou
honghu-goods-label [boot] [trunk/ml/honghu/
honghu-goods-service [boot] [trunk/ml/hongh
honghu-goodsclass [trunk/ml/honghu/source/
honghu-goodsclass-api [boot] [trunk/ml/hon
honghu-goodsclass-client [boot] [trunk/ml/
honghu-goodsclass-dao [boot] [trunk/ml/ho
honghu-goodsclass-label [boot] [trunk/ml/
honghu-goodsclass-service [boot] [trunk/
honghu-order [trunk/ml/honghu/source/hongh
honghu-order-api [boot] [trunk/ml/honghu/source/
```

原文链接：springcloud(六)：熔断监控 Turbine