在13 服务容错保护（D版 Hystrix断路器）.note的介绍中，我们提到断路器是根据一段时间窗内的请求情况来判断并操作断路器的打开和关闭状态的。而这些请求情况的指标信息都是HystrixCommand和HystrixObservableCommand实例在执行过程中记录的重要度量信息，它们除了Hystrix断路器实现中使用之外，对于系统运维也有非常大的帮助。这些指标信息会以“滚动时间窗”与“桶”结合的方式进行汇总，并在内存中驻留一段时间，以供内部或外部进行查询使用，Hystrix Dashboard就是这些指标内容的消费者之一。

下面我们基于之前的示例来结合Hystrix Dashboard实现Hystrix指标数据的可视化面板，这里我们将用到下之前实现的几个应用，包括：

- eureka-server：服务注册中心
- eureka-client：服务提供者
- eureka-consumer-ribbon-hystrix：使用ribbon和hystrix实现的服务消费者

由于eureka-consumer-ribbon-hystrix项目中的/consumer接口实现使用了@HystrixCommand修饰，所以这个接口的调用情况会被Hystrix记录下来，以用来给断路器和Hystrix Dashboard使用。断路器我们在上一篇中已经介绍过了，下面我们来具体说说Hystrix Dashboard的构建。

动手试一试

在Spring Cloud中构建一个Hystrix Dashboard非常简单，只需要下面四步：

- 创建一个标准的Spring Boot工程，命名为：hystrix-dashboard。
- 编辑pom.xml，具体依赖内容如下：

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

- 为应用主类加上@EnableHystrixDashboard，启用Hystrix Dashboard功能：

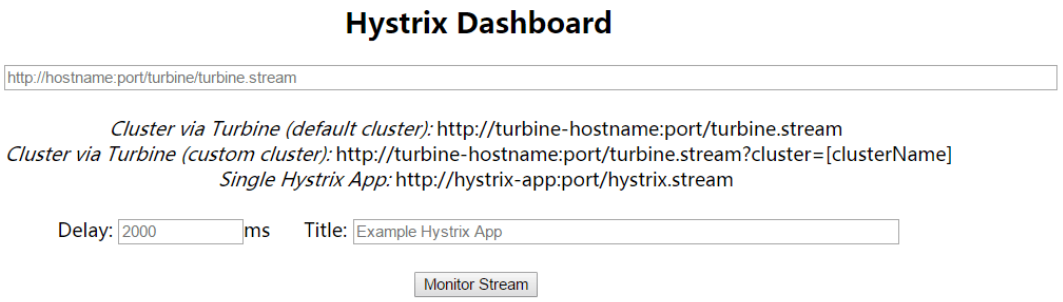
```
@EnableHystrixDashboard
@SpringCloudApplication
public class HystrixDashboardApplication {
    public static void main(String[] args) {
```

```
        SpringApplication.run(HystrixDashboardApplication.class, args);
    }
}
```

- 根据实际情况修改application.properties配置文件，比如：选择一个未被占用的端口等，此步非必须。

```
spring.application.name=hystrix-dashboard
server.port=2310
```

到这里我们已经完成了基本配置，接下来我们可以启动该应用，并访问：http://localhost:2310/hystrix，我们可以看到如下页面：



这是Hystrix Dashboard的监控首页，该页面中并没有具体的监控信息。从页面的文字内容中我们可以知道，Hystrix Dashboard共支持三种不同的监控方式，依次为：

- 默认的集群监控：通过URLhttp://turbine-hostname:port/turbine.stream开启，实现对默认集群的监控。
- 指定的集群监控：通过URLhttp://turbine-hostname:port/turbine.stream?cluster=[clusterName]开启，实现对clusterName集群的监控。
- 单体应用的监控：通过URLhttp://hystrix-app:port/hystrix.stream开启，实现对具体某个服务实例的监控。

前两者都对集群的监控，需要整合Turbine才能实现，这部分内容我们在下一节中做详细介绍。在本节中，我们主要实现对单个服务实例的监控，所以这里我们先来实现单个服务实例的监控。

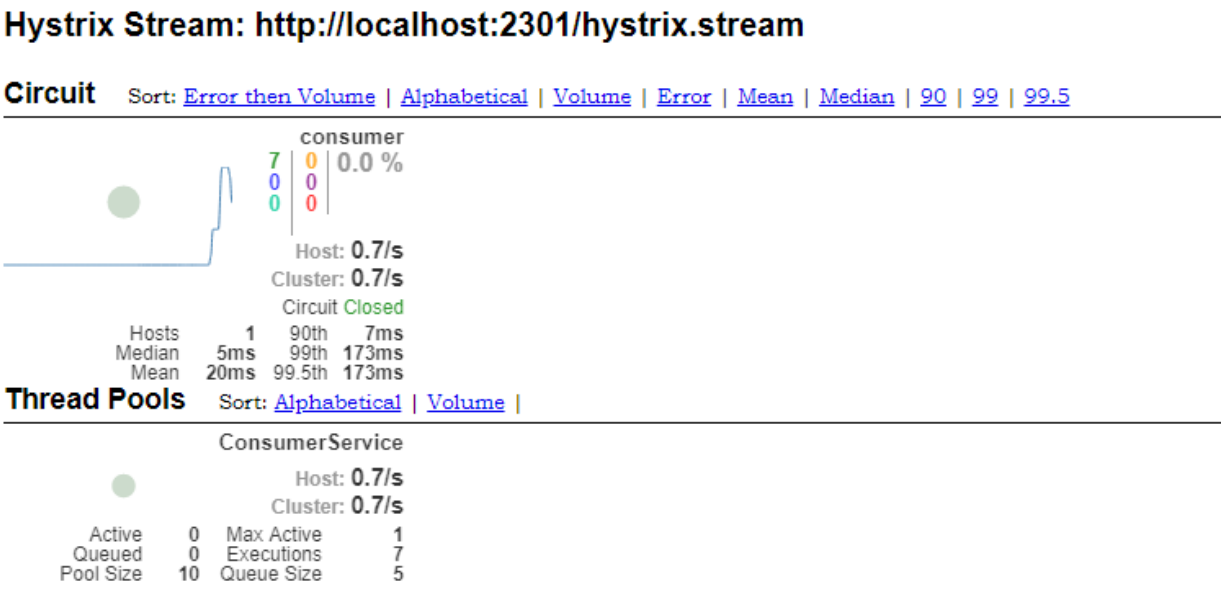
既然Hystrix Dashboard监控单实例节点需要通过访问实例的/hystrix.stream接口来实现，自然我们需要为服务实例添加这个端点，而添加该功能的步骤也同样简单，只需要下面两步：

- 在服务实例pom.xml中的dependencies节点中新增spring-boot-starter-actuator监控模块以开启监控相关的端点，并确保已经引入断路器的依赖spring-cloud-starter-hystrix：

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

- 确保在服务实例的主类中已经使用@EnableCircuitBreaker或@EnableHystrix注解，开启了断路器功能。

到这里已经完成了所有的配置，我们可以在Hystrix Dashboard的首页输入http://localhost:2301/hystrix.stream，已启动对“eureka-consumer-ribbon-hystrix”的监控，点击“Monitor Stream”按钮，此时我们可以看到如下页面：

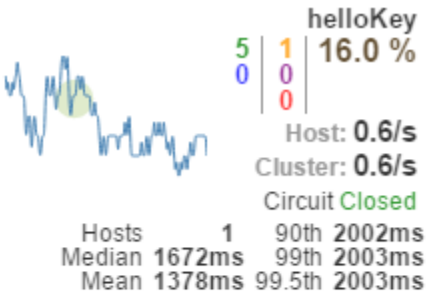


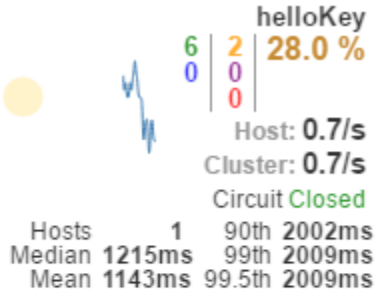
在对该页面介绍前，我们先看看在首页中我们还没有介绍的两外两个参数：

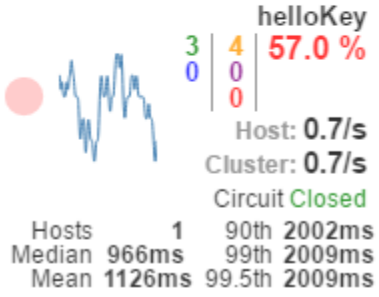
- Delay：该参数用来控制服务器上轮询监控信息的延迟时间，默认为2000毫秒，我们可以通过配置该属性来降低客户端的网络和CPU消耗。
- Title：该参数对应了上图头部标题Hystrix Stream之后的内容，默认会使用具体监控实例的URL，我们可以通过配置该信息来展示更合适的标题。

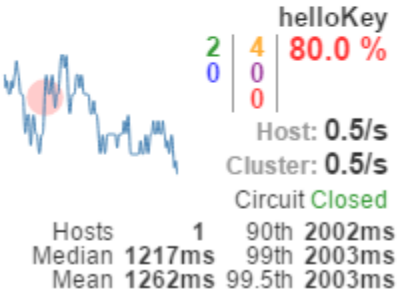
回到监控页面，我们来详细说说其中各元素的具体含义：

- 我们可以在监控信息的左上部分找到两个重要的图形信息：一个实心圆和一条曲线。
    - 实心圆：共有两种含义。它通过颜色的变化代表了实例的健康程度，如下图所示，它的健康度从绿色、黄色、橙色、红色递减。该实心圆除了颜色的变化之外，它的大小也会根据实例的请求流量发生变化，流量越大该实心圆就越大。所以通过该实心圆的展示，我们就可以在大量的实例中快速的发现故障实例和高压力实例。









- 曲线：用来记录2分钟内流量的相对变化，我们可以通过它来观察到流量的上升和下降趋势。

- 其他一些数量指标如下图所示：