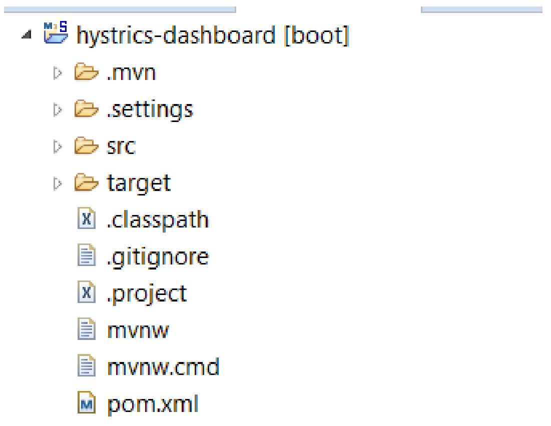
Spring **Cloud**

Enabling the Hystrix Dashboard

Step-1: create a separate application with the name “hystrics-dashboard” as shown below:



```
hystrics-dashboard [boot]
  .mvn
  .settings
  src
  target
  .classpath
  .gitignore
  .project
  mvnw
  mvnw.cmd
  pom.xml
```

With the startes below:

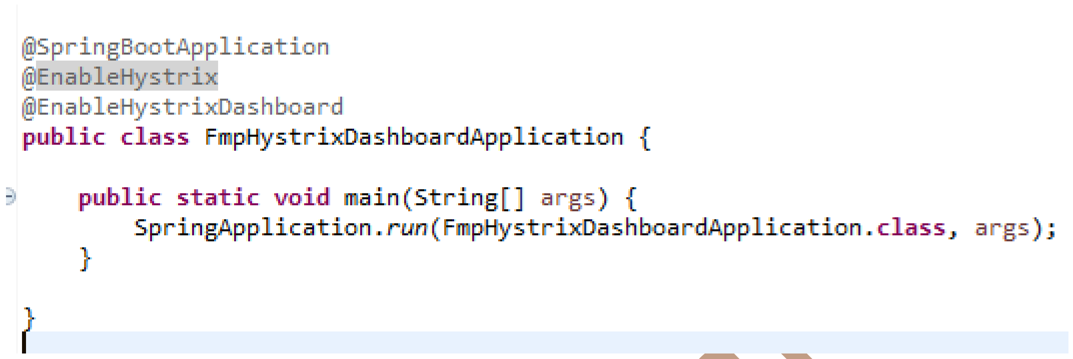
```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-hystrix-dashboard</artifactId>
</dependency>
```

Note:

Hystrix dashboard will use freemarker template engine to render its dashboard. Hence it will down load the latest version of free marker dependency in to our maven local repository. If it is not downloadable by maven , the explicitly pass it as maven dependency in pom .xml file as shown below

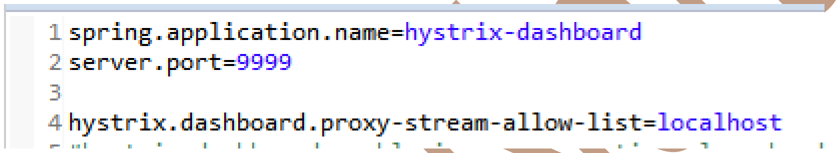
```
<!-- https://mvnrepository.com/artifact/org.freemarker/freemarker -->

        <dependency>

            <groupId>org.freemarker</groupId>

            <artifactId>freemarker</artifactId>
            <version>2.3.28</version>

        </dependency>
```

Step-3: add @**EnableHystrixDashboard** annotation on application java class as shown below.



```
@SpringBootApplication
@EnableHystrix
@EnableHystrixDashboard
public class FmpHystrixDashboardApplication {

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

}
```

Step-4: configure the application name and port as shown below. Hystrix dashboard will start up on the default port 9999.



```
spring.application.name=hystrix-dashboard
server.port=9999

hystrix.dashboard.proxy-stream-allow-list=localhost
```

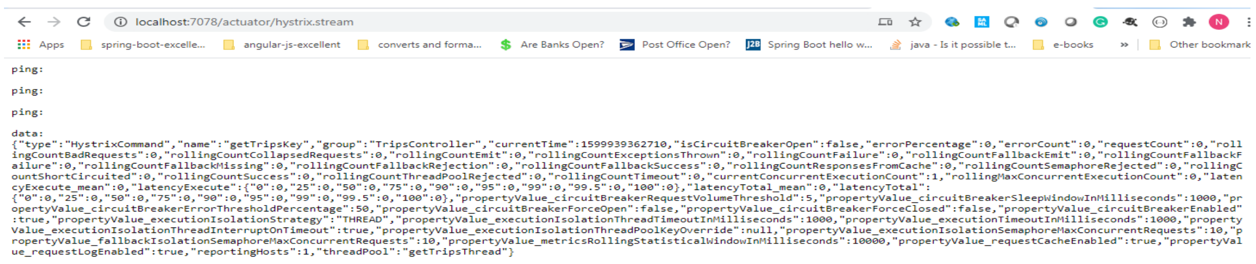
Step-5:

Hystrix Dashboard

As we have added hystrix dashboard dependency, hystrix has provided a nice Dashboard and a Hystrix Stream in the bellow URLS:

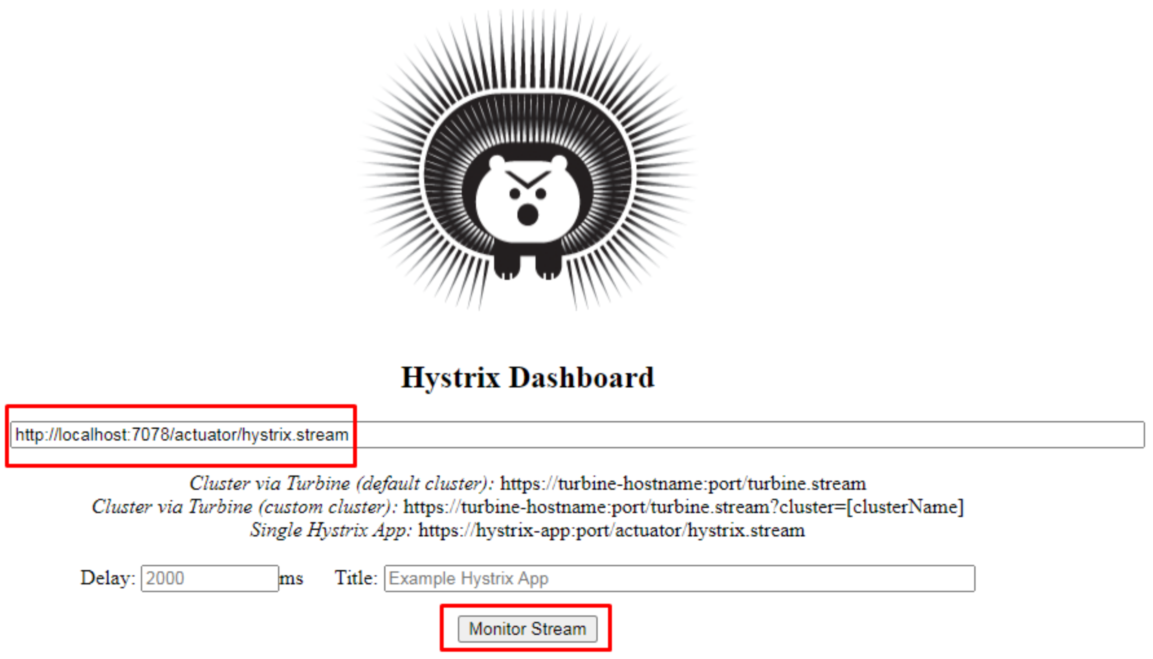
http://localhost:7075/hystrix.stream –Here as doctor-service is enabled with Circuit Breaker, a continuous stream Hystrix will generates on it to check health and being monitored by Hystrix.

Spring **Cloud**



```
ping:

ping:

ping:

data:
{"type":"HystrixCommand","name":"getTripsKey","group":"TripsController","currentTime":1599939362710,"isCircuitBreakerOpen":false,"errorPercentage":0,"errorCount":0,"requestCount":0,"roll
ingCountBadRequests":0,"rollingCountCollapsedRequests":0,"rollingCountEmit":0,"rollingCountExceptionsThrown":0,"rollingCountFailure":0,"rollingCountFallbackEmit":0,"rollingCountFallbackF
ailure":0,"rollingCountFallbackMissing":0,"rollingCountFallbackRejection":0,"rollingCountFallbackSuccess":0,"rollingCountResponsesFromCache":0,"rollingCountSemaphoreRejected":0,"rollingC
ountShortCircuited":0,"rollingCountSuccess":0,"rollingCountThreadPoolRejected":0,"rollingCountTimeout":0,"currentConcurrentExecutionCount":1,"rollingMaxConcurrentExecutionCount":0,"laten
cyExecute_mean":0,"latencyExecute":{"0":0,"25":0,"50":0,"75":0,"90":0,"95":0,"99":0,"99.5":0,"100":0},"latencyTotal_mean":0,"latencyTotal":
{"0":0,"25":0,"50":0,"75":0,"90":0,"95":0,"99":0,"99.5":0,"100":0},"propertyValue_circuitBreakerRequestVolumeThreshold":5,"propertyValue_circuitBreakerSleepWindowInMilliseconds":1000,"pr
opertyValue_circuitBreakerErrorThresholdPercentage":50,"propertyValue_circuitBreakerForceOpen":false,"propertyValue_circuitBreakerForceClosed":false,"propertyValue_circuitBreakerEnabled"
:true,"propertyValue_executionIsolationStrategy":"THREAD","propertyValue_executionIsolationThreadTimeoutInMilliseconds":1000,"propertyValue_executionTimeoutInMilliseconds":1000,"property
Value_executionIsolationThreadInterruptOnTimeout":true,"propertyValue_executionIsolationThreadPoolKeyOverride":null,"propertyValue_executionIsolationSemaphoreMaxConcurrentRequests":10,"p
ropertyValue_fallbackIsolationSemaphoreMaxConcurrentRequests":10,"propertyValue_metricsRollingStatisticalWindowInMilliseconds":10000,"propertyValue_requestCacheEnabled":true,"propertyVal
ue_requestLogEnabled":true,"reportingHosts":1,"threadPool":"getTripsThread"}
```

run the service and open the url http://localhost:9999/hystrix. the following dash board will be opened.



Hystrix Dashboard

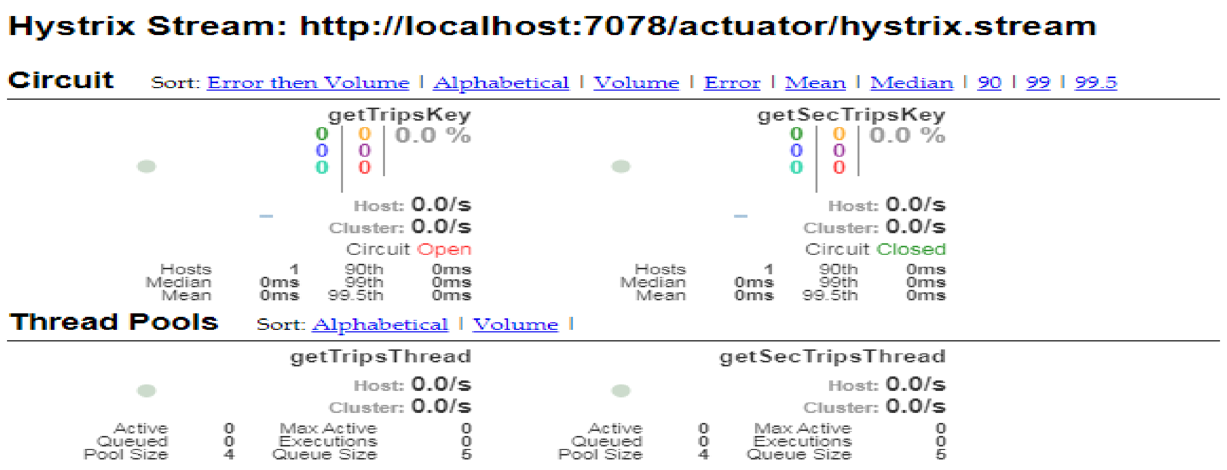
http://localhost:7078/actuator/hystrix.stream

Cluster via Turbine (default cluster): https://turbine-hostname:port/turbine.stream
Cluster via Turbine (custom cluster): https://turbine-hostname:port/turbine.stream?cluster=[clusterName]
Single Hystrix App: https://hystrix-app:port/actuator/hystrix.stream

Delay: 2000 ms Title: Example Hystrix App

Here in dash board to see the stream of trips-service I.e. failure services etc, give the trips-service stream url I.e. **http://localhost:7075/hystrix.stream** in the bar above and give the application-name I.e.trips-service as Title and click on Monitor stream. Now we should see the stream as below:

Spring **Cloud**



```
Hystrix Stream: http://localhost:7078/actuator/hystrix.stream

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

                     getTripsKey                                  getSecTripsKey
                  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 Open                                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 |

                  getTripsThread                              getSecTripsThread
                    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   4   Queue Size   5             Pool Size   4   Queue Size   5
```

If there are continuous failures, then the circuit status will be changed to open. This can be done by hitting the preceding URL a number of times. In the open state, the original service will no longer be checked. The Hystrix Dashboard will show the status of the circuit as **Open** and **Closed.** Once a circuit is opened, periodically, the system will check for the original service status for recovery. When the original service is back, the circuit breaker will fall back to the original service and the status will be set to **Closed.**