Spring Cloud Netflix: Circuit Breakers with Hystrix

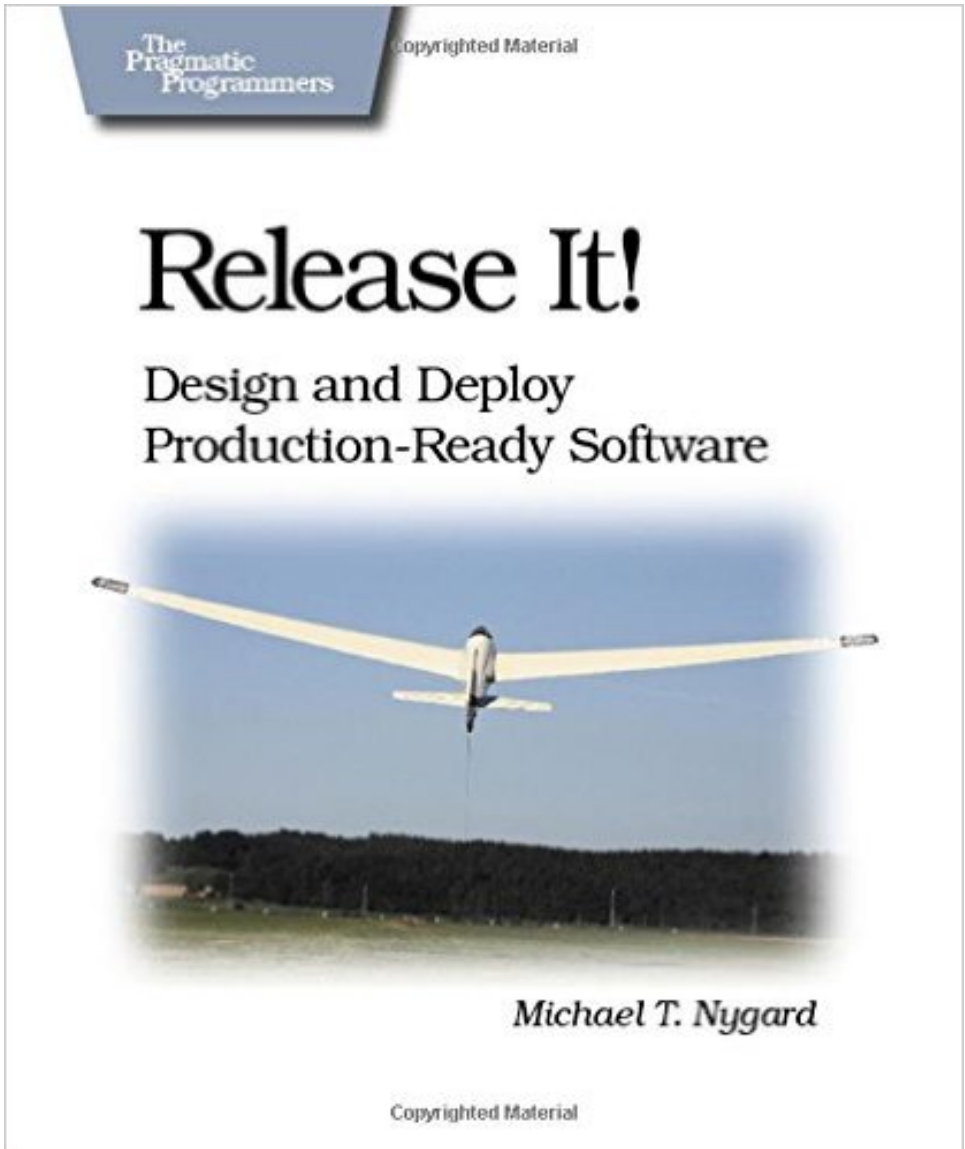
Spring Cloud Netflix: Circuit Breakers with Hystrix

- Review Fault Tolerance
- Circuit Breakers
- Hystrix Dashboard

Fault Tolerance

Fault Tolerance Patterns like the *Circuit Breaker* stop cascading failures.

As described by, Michael T. Nygard in *Release It!*



Fault Tolerance

One failure must not cause a cascading failure across the entire system.

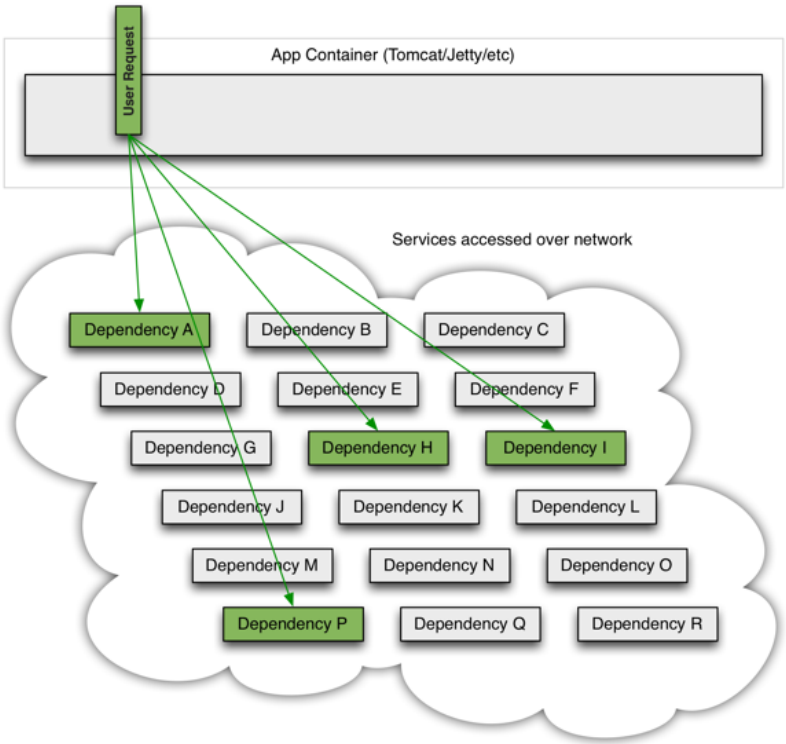
For example, for an application that depends on 30 services where each service has 99.99% uptime, here is what you can expect:

```
99.99^30 = 99.7% uptime

0.3% of 1 billion requests = 3,000,000 failures

2+ hours downtime/month even if all dependencies have excellent uptime.
```

Reality is *generally* worse.

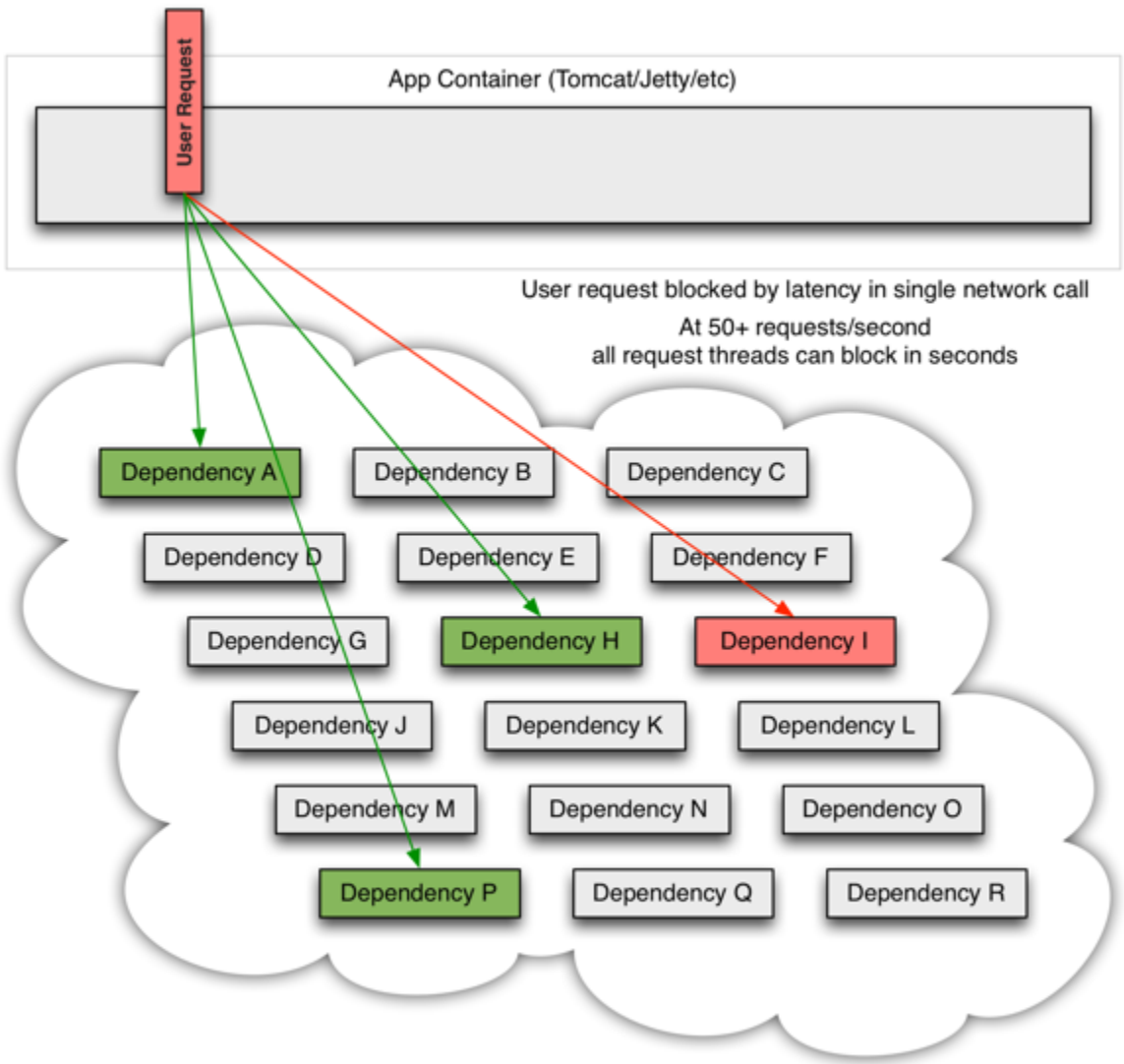
Source: https://github.com/Netflix/Hystrix/wiki

Healthy Request Flow

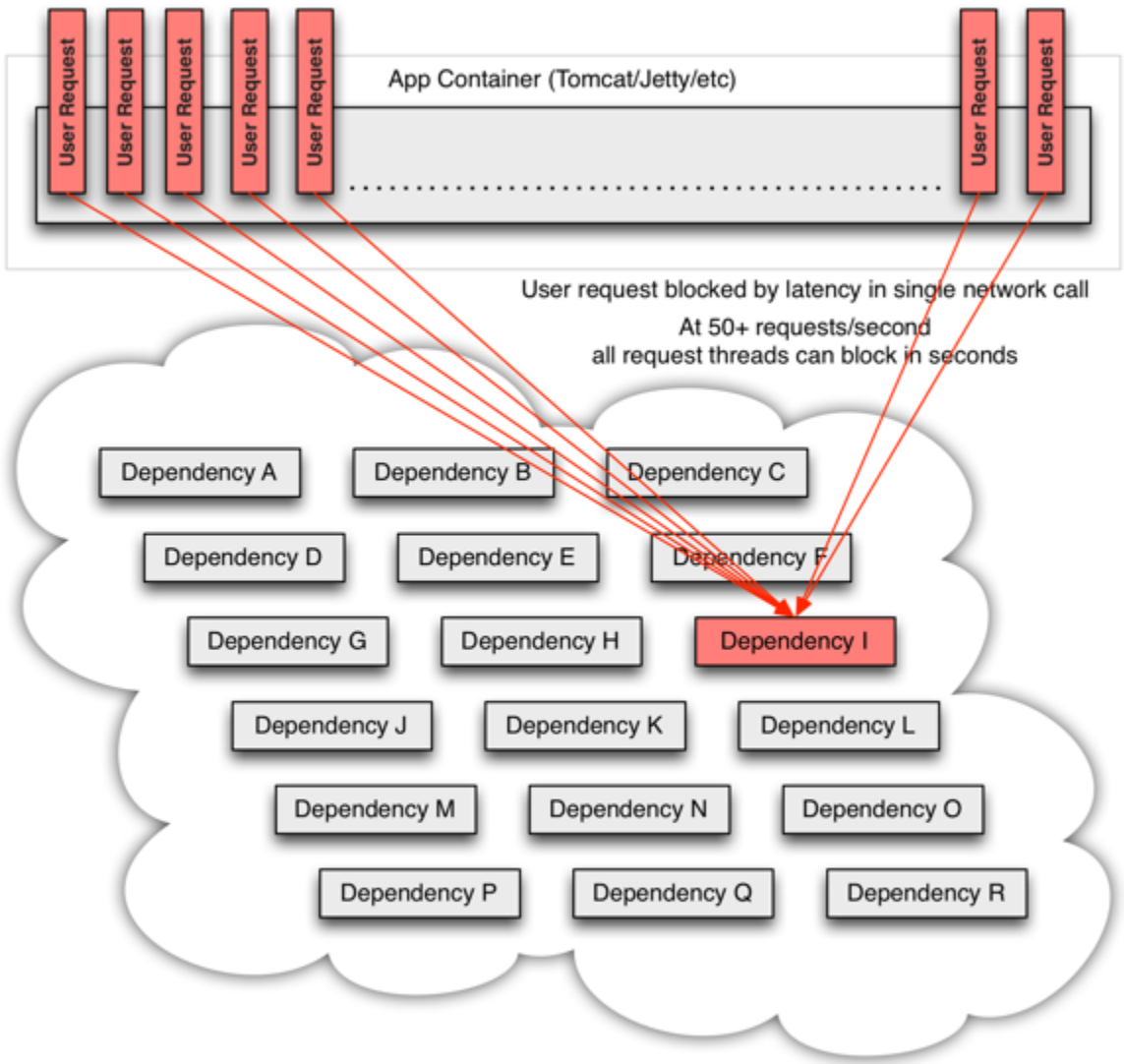


Source: https://github.com/Netflix/Hystrix/wiki

Dependency Down



Cascading Failure



Spring Cloud Netflix: Circuit Breakers with Hystrix

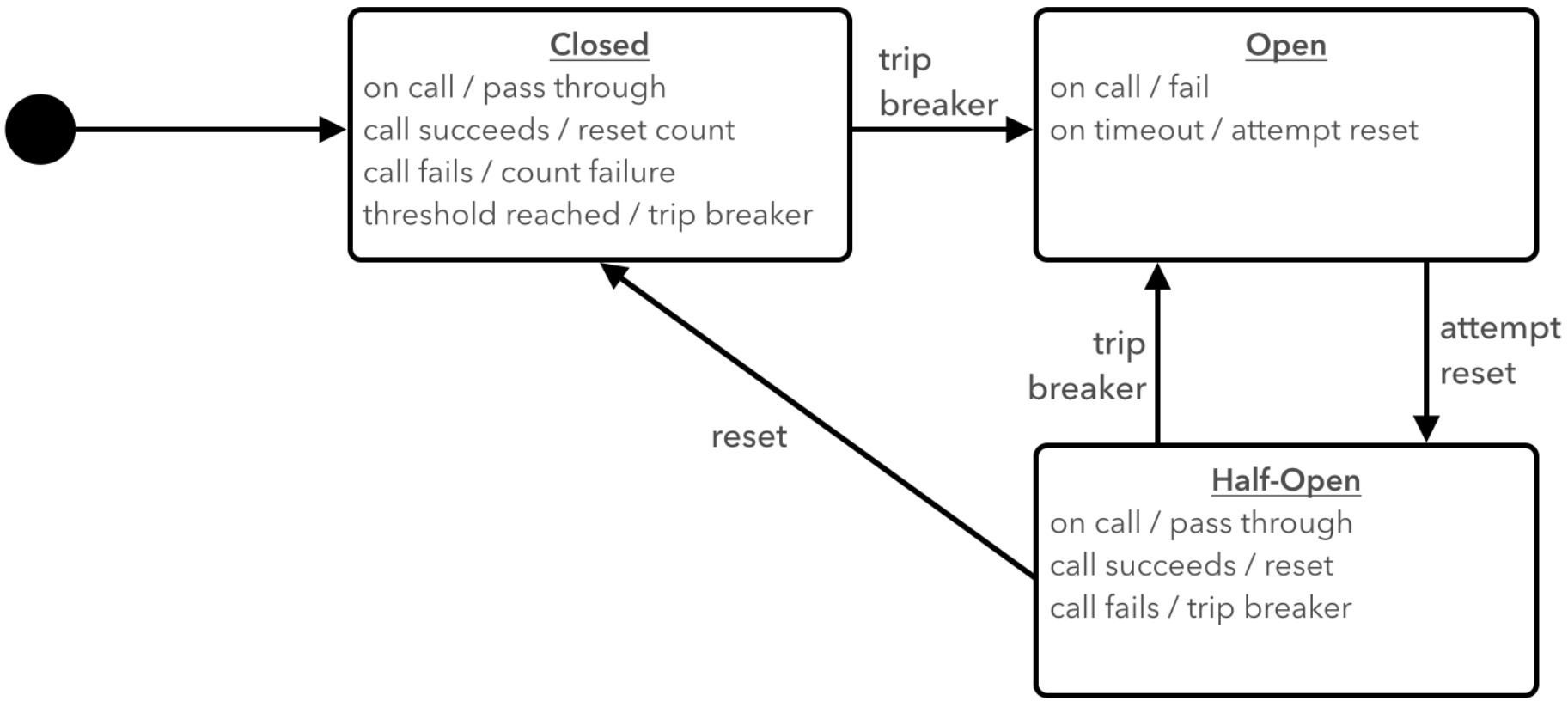
- Review Fault Tolerance
- Circuit Breakers
- Hystrix Dashboard

Hystrix - Defend Your App

Hystrix is designed to do the following:

- Give protection from and control over latency and failure from dependencies accessed (typically over the network) via third-party client libraries.
- Stop cascading failures in a complex distributed system.
- Fail fast and rapidly recover.
- Fallback and gracefully degrade when possible.
- Enable near real-time monitoring, alerting, and operational control.

Circuit Breaker Diagram



Include Dependency

pom.xml

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

`@EnableCircuitBreaker` Annotation

```
@SpringBootApplication
@EnableDiscoveryClient
@EnableCircuitBreaker
public class GreetingHystrixApplication {


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

}
```

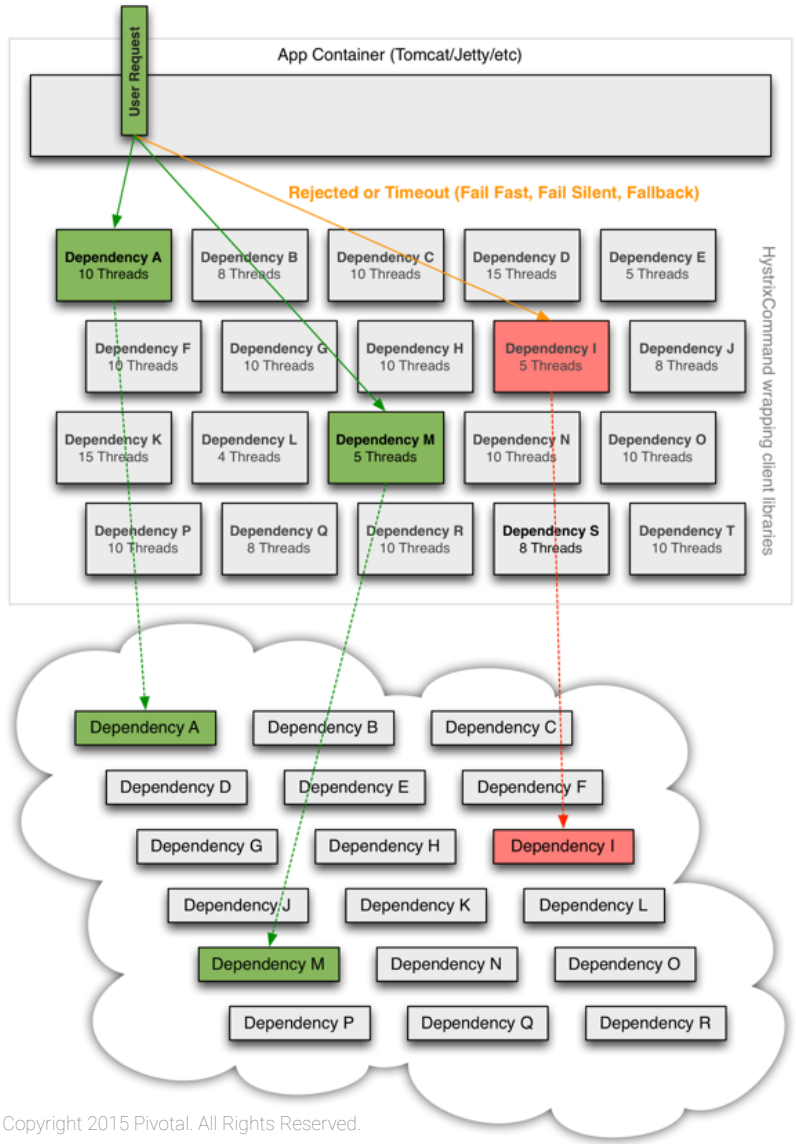
`@HystrixCommand` annotation

```
@Service
public class FortuneService {
    ...

    @HystrixCommand(fallbackMethod = "defaultFortune")
    public String getFortune() {

      return restTemplate.getForObject
        ("http://fortune-service", String.class);
    }

    public String defaultFortune(){
        logger.debug("Default fortune used.");
        return "This fortune is no good. Try another.";
    }
}
```

Customize Behavior with `@HystrixProperty`

```
@Service
public class FortuneService {
    ...

  @HystrixCommand
  (fallbackMethod = "defaultFortune",
  commandProperties = {@HystrixProperty
  (name="execution.isolation.thread.timeoutInMilliseconds",
  value="500")})
    public String getFortune() {

      return restTemplate.getForObject
        ("http://fortune-service", String.class);
    }
}
```

Wrapping Calls With Hystrix



Metrics

Hystrix publishes real-time metrics for each **`@HystrixCommand`**:

- Informational and Status (**`isCircuitOpen`**)
- Cumulative and Rolling Event Counts (**`countExceptionsThrown`** & **`rollingCountExceptionsThrown`**)
- Latency Percentiles (**`latencyExecute_percentile_995`**)
- Latency Percentiles: End-to-End Execution (**`latencyTotal_percentile_5`**)
- Property Values (**`propertyValue_circuitBreakerRequestVolumeThreshold`**)

** Thread Pool metrics are also available

Event stream

To expose the **`/hystrix.stream`** as a management endpoint include the **`actuator`** dependency.

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

Spring Cloud Netflix: Circuit Breakers with Hystrix

- Review Fault Tolerance
- Circuit Breakers
- Hystrix Dashboard

Include Dependency

pom.xml

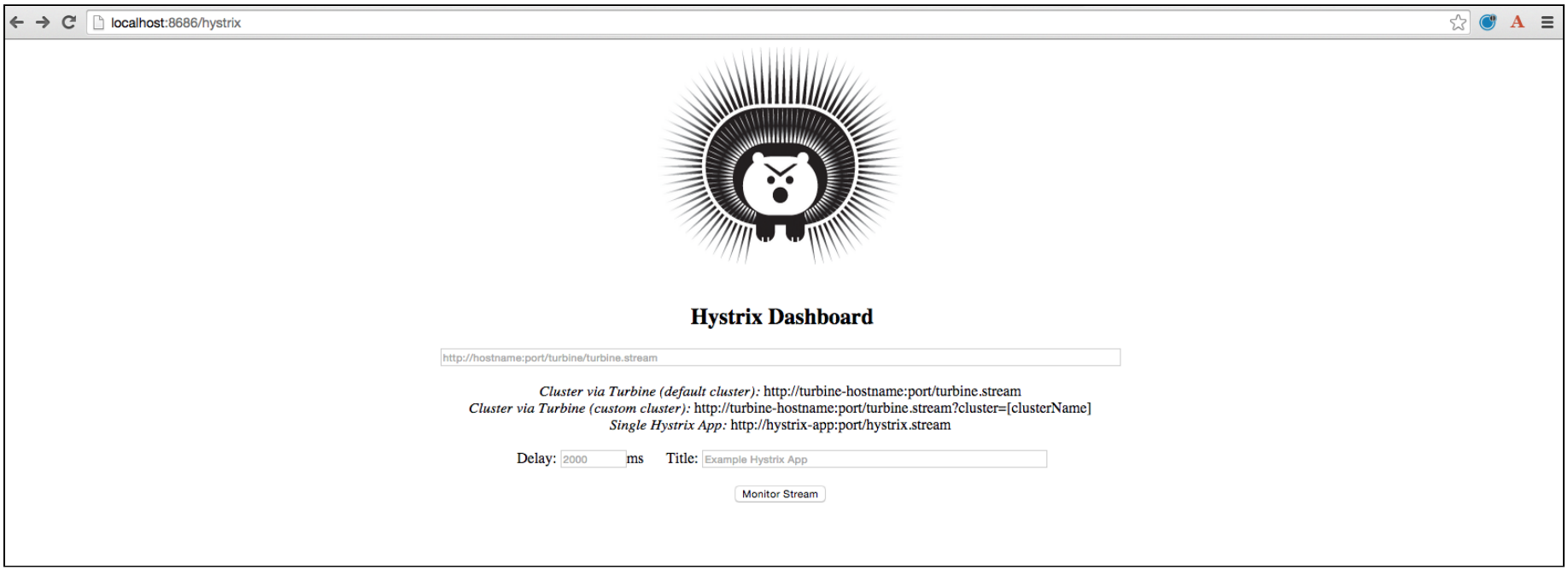
```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-hystrix-dashboard</artifactId>
</dependency>
```

`@EnableHystrixDashboard` Annotation

```
@SpringBootApplication
@EnableHystrixDashboard
public class HystrixDashboardApplication {

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

Configure the Dashboard

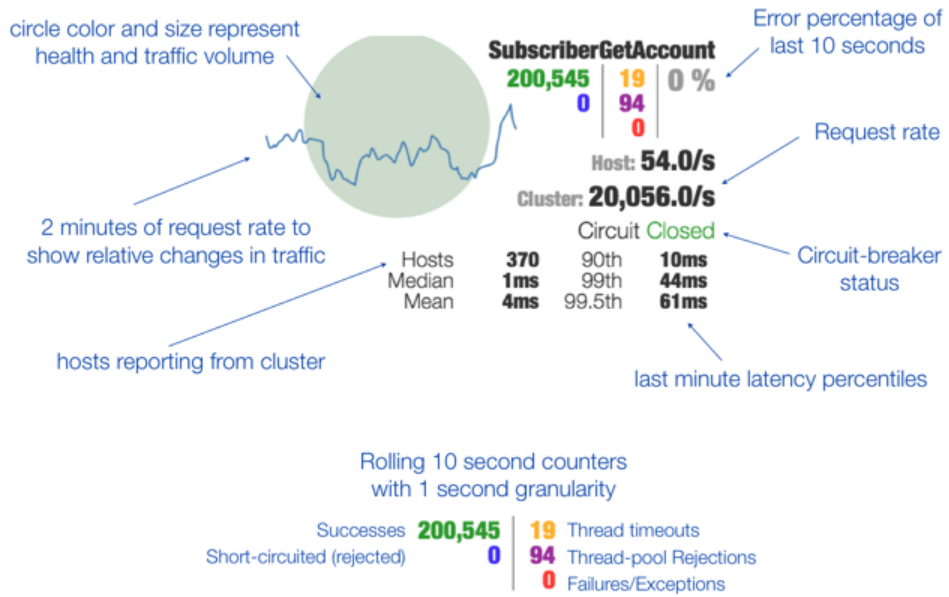
Enter the the **`/hystrix.stream`** endpoint.



Hystrix Dashboard

When Netflix began to use this dashboard, their operations improved by reducing the time needed to discover and recover from operational events.

The duration of most production incidents (already less frequent due to Hystrix) became far shorter, with diminished impact, due to the real-time insights into system behavior provided by the Hystrix Dashboard.



Source: https://github.com/Netflix/Hystrix/wiki/Dashboard