

The background of the slide is a photograph of the Golden Gate Bridge in San Francisco, taken from a low angle looking down the length of the bridge towards the foggy horizon. The bridge's iconic orange-red color is muted by a dark blue overlay. The suspension cables and towers are visible, creating a strong sense of perspective.

Pivotal

# Cloud Native Applications

Spring Cloud Netflix – Circuit Breakers and Fault Tolerance

# 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:

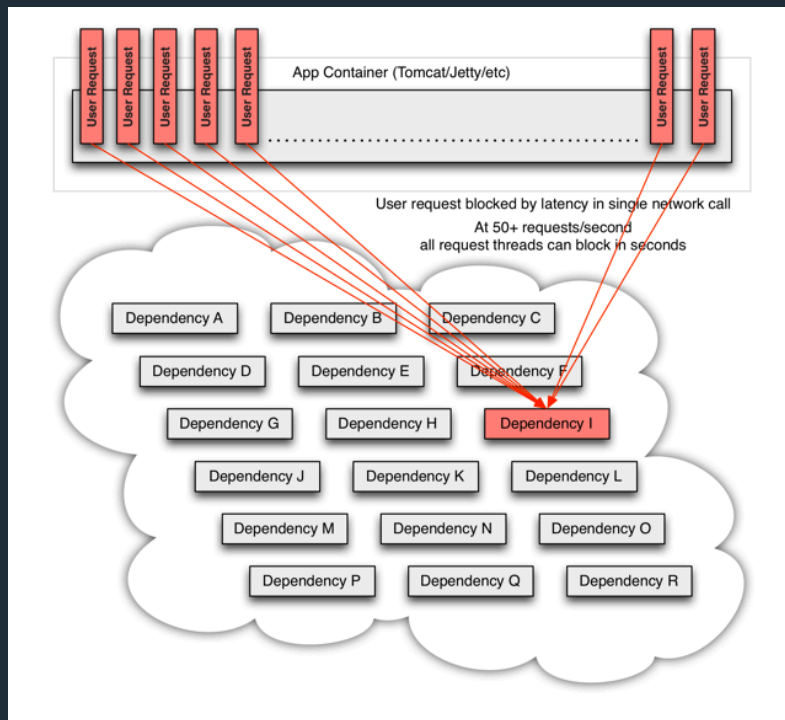
`99.99^30 = 99.7% uptime`

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

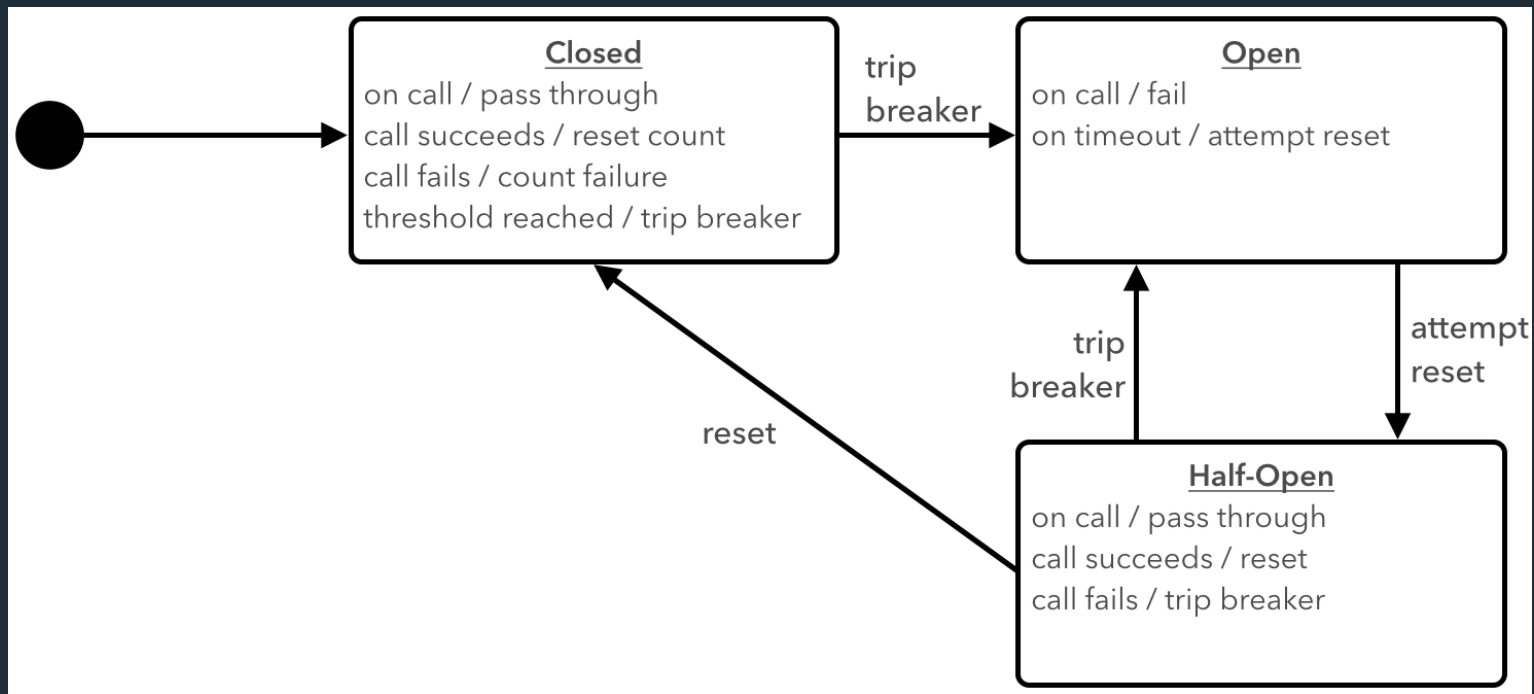
`2+ hours downtime/month if all dependencies have 99.99%`

- Reality is *generally* worse.
- Source: <https://github.com/Netflix/Hystrix/wiki>

# Distributed Systems Failures



# Circuit Breaker Pattern



# Implementing Circuit Breakers

```
@SpringBootApplication
@EnableDiscoveryClient
@EnableCircuitBreaker ← MAGIC!!
public class MyClientApp{
    public static void main(String[] args) {
        SpringApplication.run(MyClientApp.class, args);
    }
}
```

```
<dependency>
  <groupId>io.pivotal.spring.cloud</groupId>
  <artifactId>spring-cloud-starter-hystrix</artifactId>
</dependency>
```

# @HystrixCommand

```
@Service
public class FortuneService {

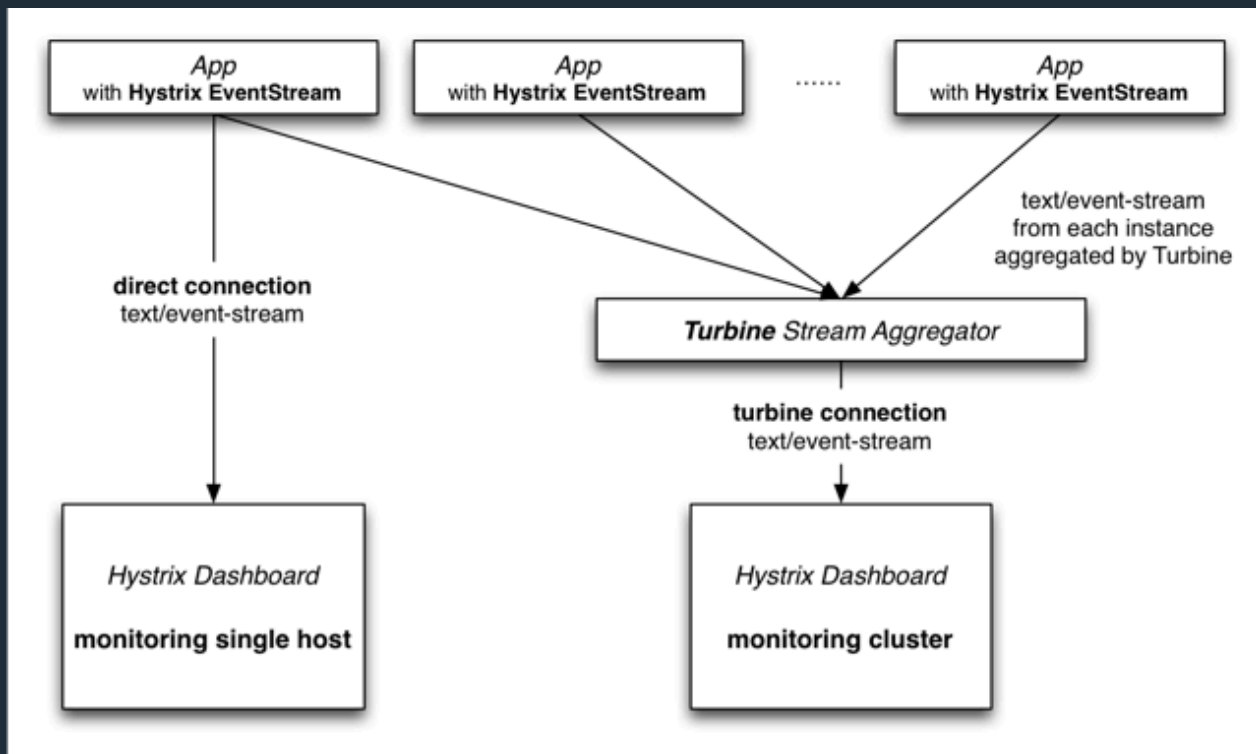
    @HystrixCommand(fallbackMethod = "defaultFortune",
        commandProperties = {
            @HystrixProperty(name="execution.isolation.thread.timeoutInMilliseconds", value="500")
        })
    public String getFortune() {
        return restTemplate.getForObject("http://fortune-service", String.class);
    }

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

# @HystrixCommand Metrics

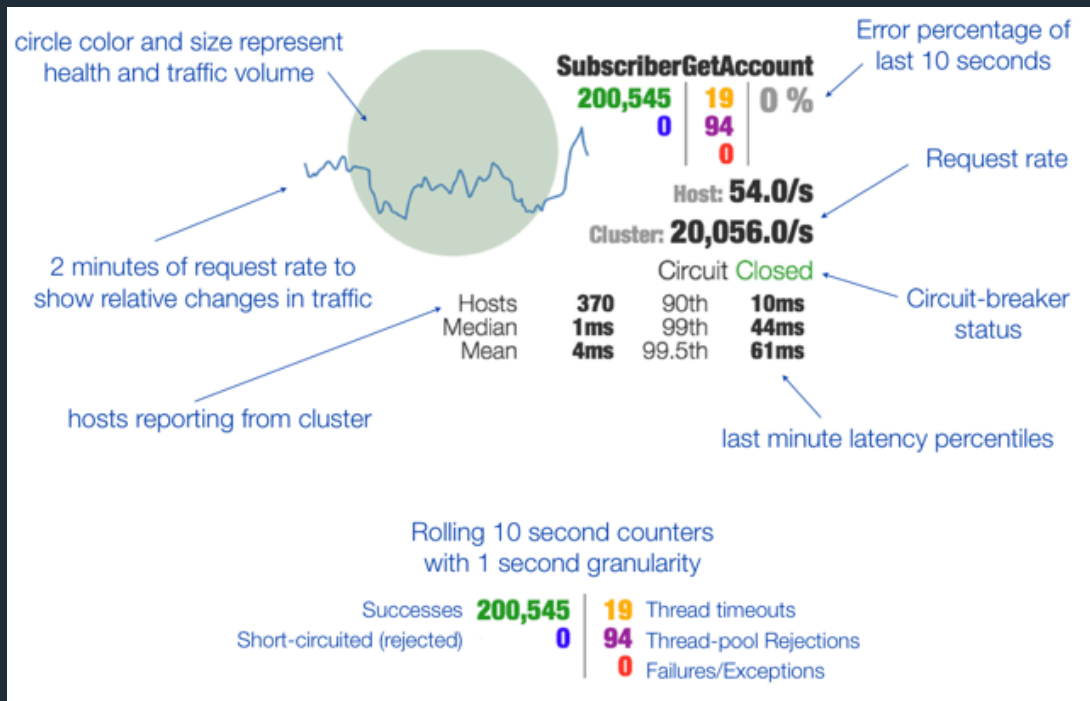
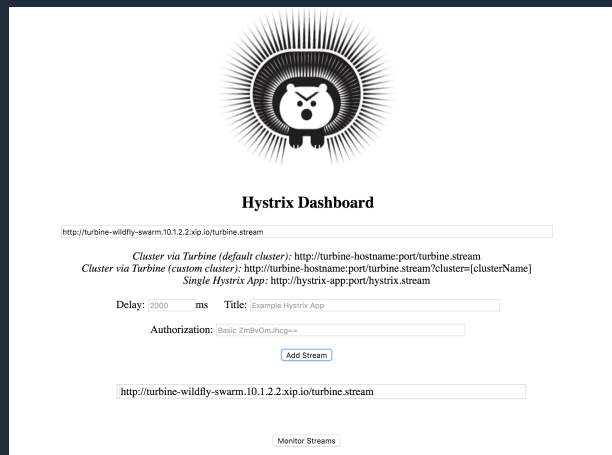
- **Hystrix publishes real-time metrics for each @HystrixCommand**
  - Informational and Status (isCircuitOpen)
  - Cumulative and Rolling Event Counts (countExceptionsThrown & rollingCountExceptionsThrown)
  - Latency Percentiles (latencyExecute\_percentile\_995)
  - Latency Percentiles: End-to-End Execution ( latencyTotal\_percentile\_5)
  - Property Values (propertyValue\_circuitBreakerRequestVolumeThreshold)
- Published to /hystrix.stream endpoint & boot actuator metrics
- Individual /hystrix.streams aggregated via Turbine and published via / turbine.stream or AMQP.

# Hystrix Metrics With Turbine





# Hystrix Dashboard



# Spring Cloud Services: Hystrix Dashboard

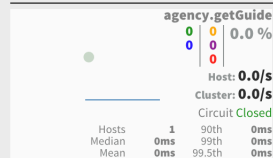
- Automated deployment dashboard + Turbine + RabbitMQ
- Bind service into app
- Include starter dependency in app

```
<dependency>
  <groupId>io.pivotal.spring.cloud</groupId>
  <artifactId>spring-cloud-services-starter-circuit-
    breaker</artifactId>
</dependency>
```

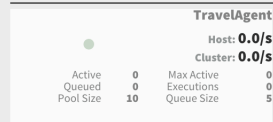


## Circuit Breaker Dashboard for Pivotal Cloud Foundry

Circuit Sort: [Error then Volume](#) | [Alphabetical](#) | [Volume](#) | [Error](#) | [Mean](#) | [Median](#) | [90](#) | [99](#) | [99.5](#)  
[Success](#) | [Short-Circuited](#) | [Timeout](#) | [Rejected](#) | [Failure](#) | Error %



Thread Pools Sort: [Alphabetical](#) | [Volume](#) |



A dark, high-contrast photograph of a modern interior space, likely a transit hub or a large office building. The scene is filled with the silhouettes of many people walking or standing. In the background, a large wall of windows allows light to filter through, creating a grid-like pattern. A large clock is visible in the upper left corner. The floor is highly reflective, mirroring the silhouettes and the light from the windows. The overall atmosphere is one of movement and modern architecture.

LAB