Cloud Native Applications

Spring Cloud Netflix – Circuit Breakers and Fault Tolerance

Fault Tolerance

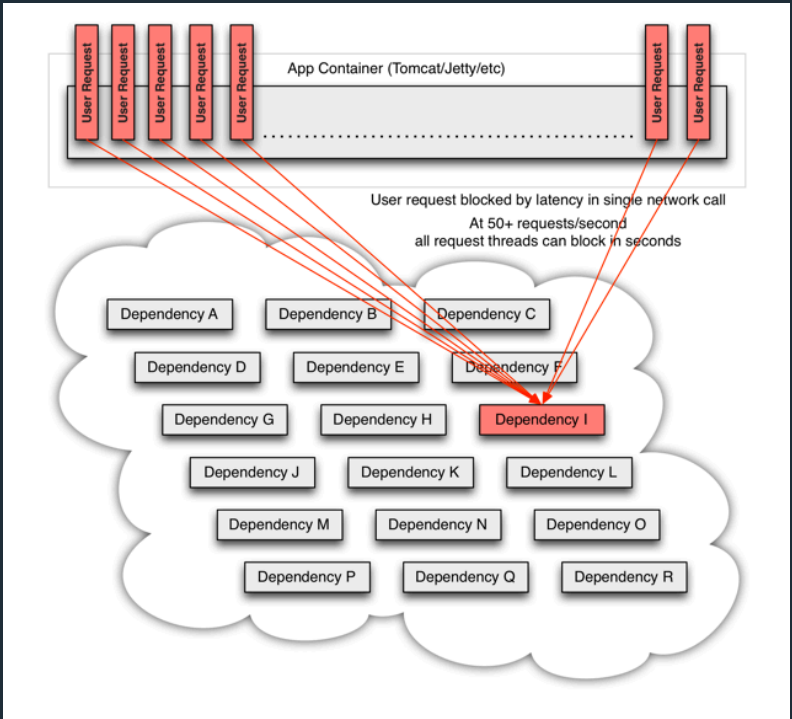
- One failure must not cause a cascading failure across the entire system.
- For example, for an application that depends on 30 services where each service has 99.99% uptime, here is what you can expect:

```
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

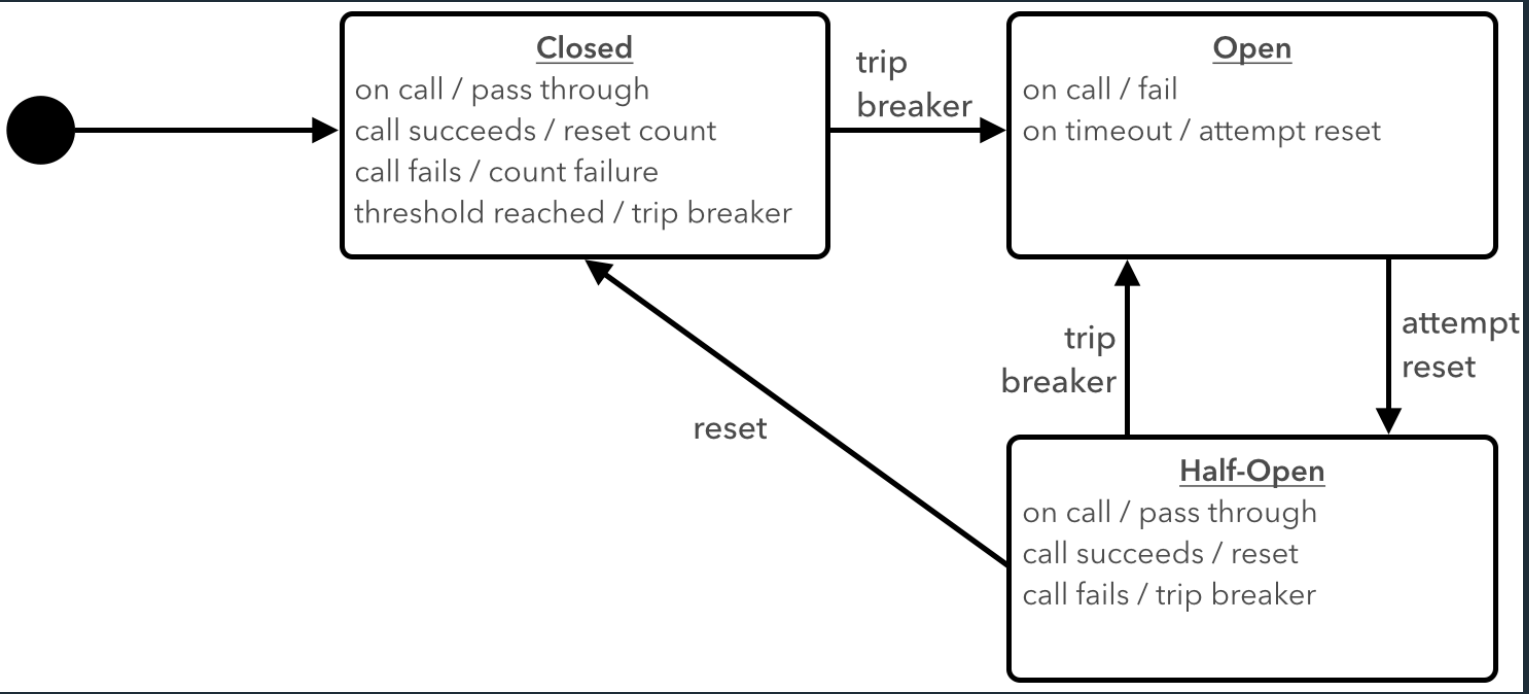
Pivotal

Distributed Systems Failures



Pivotal

Circuit Breaker Pattern



Pivotal

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>
```

Pivotal

@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.";
   }
 }
```

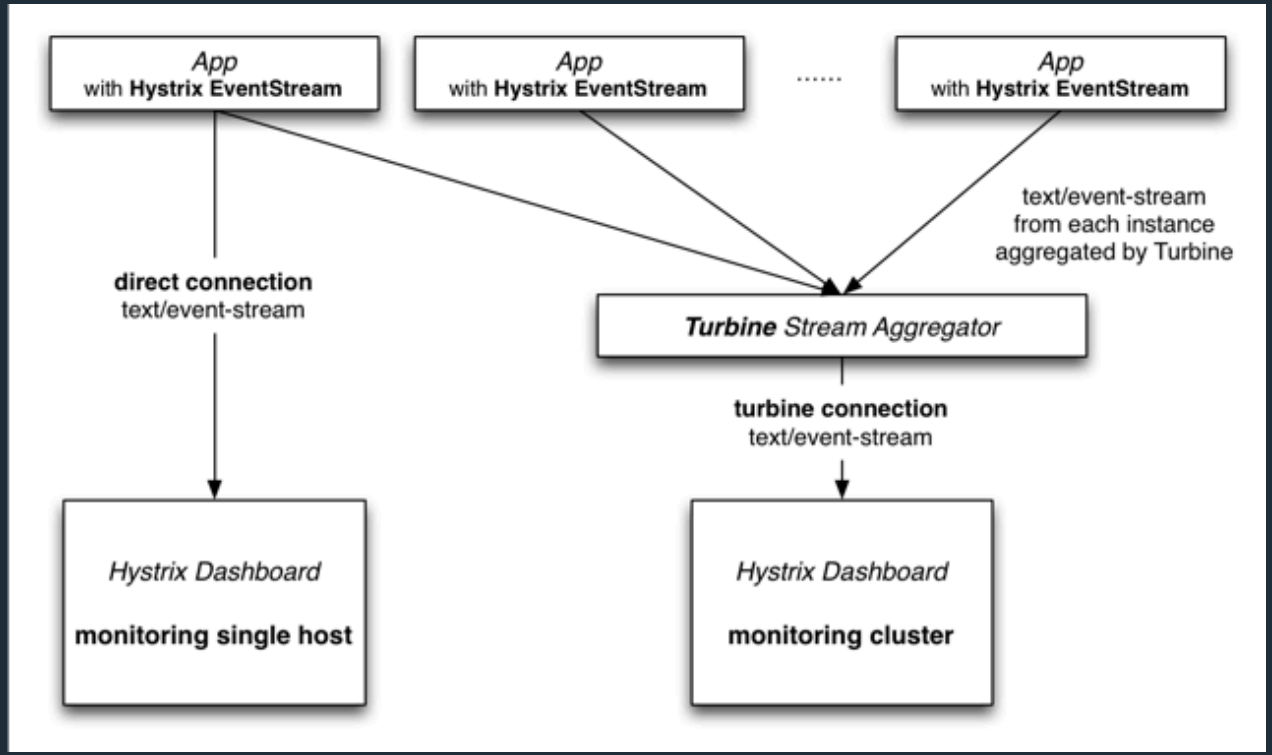
Pivotal

@HystrixCommand Metrics

- **Hystrix publishers 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.

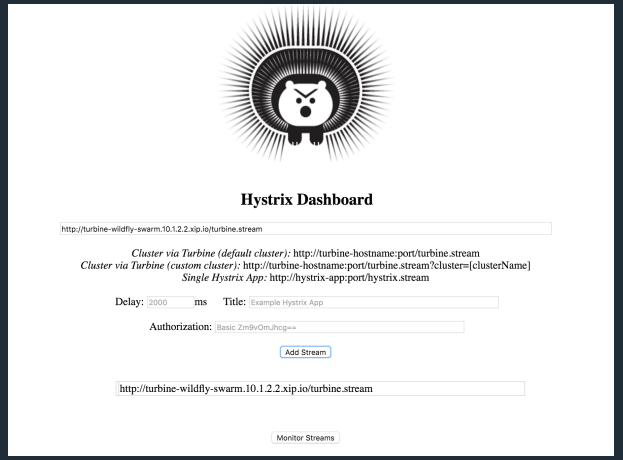
Pivotal

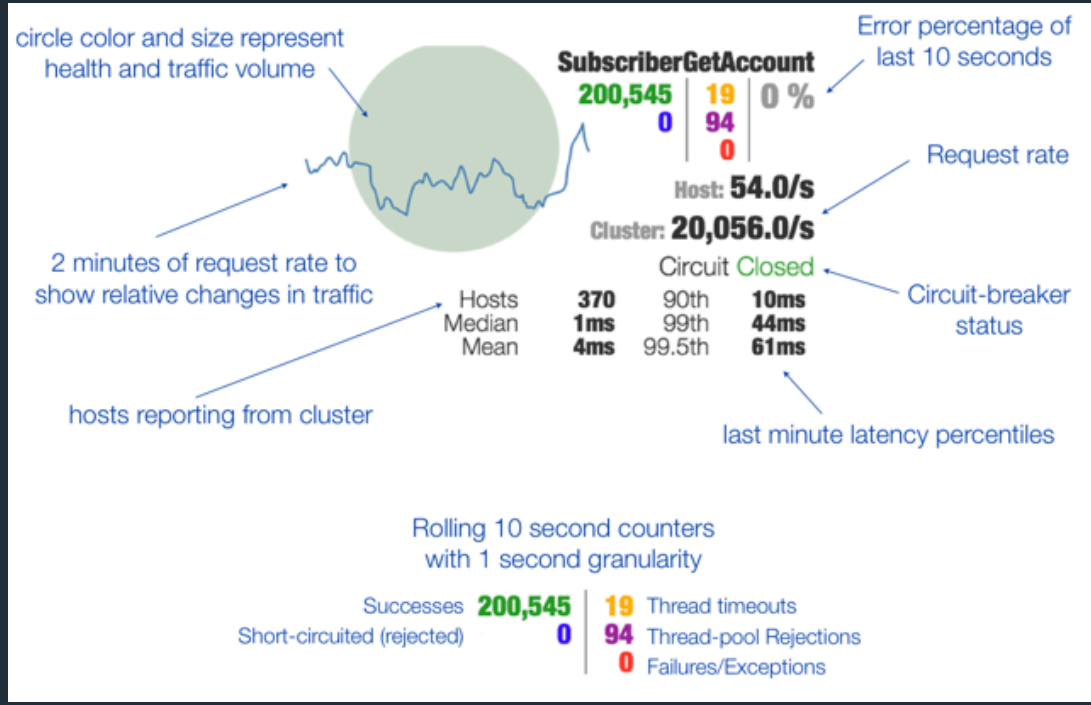
Hystrix Metrics With Turbine



Pivotal

Hystrix Dashboard

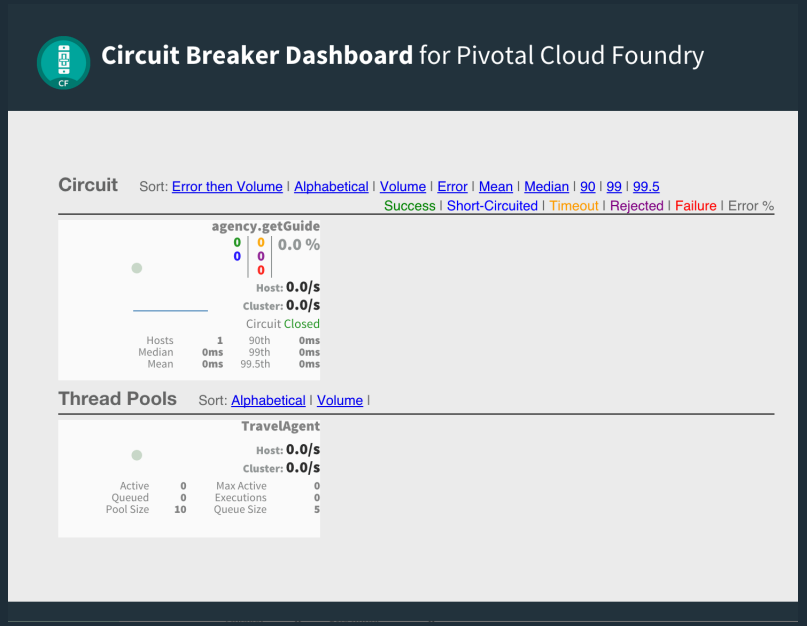




Spring Cloud Services: Hystrix Dashboard

- 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>
```



Pivotal


LAB