Dealing with system failure

HYSTRIX, NETFLIX OSS, SPRING CLOUD AND OTHER GEMS

-STUART INGRAM

@ARETHOSECLAMS

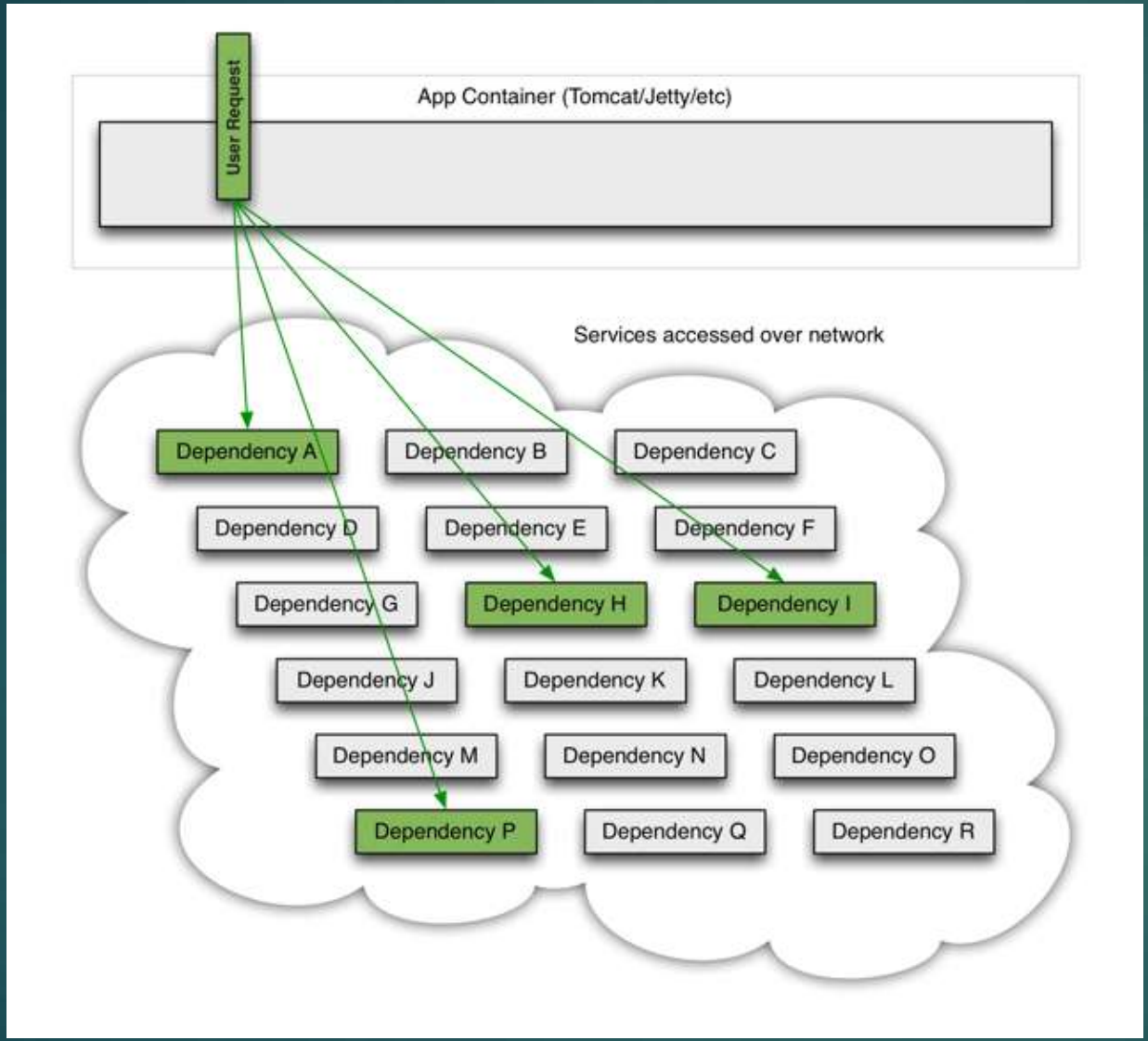
Expectations

- Problem Overview
- Circuit breaker pattern
- Concrete example - Hystrix
- Live demo
- Mystery bag

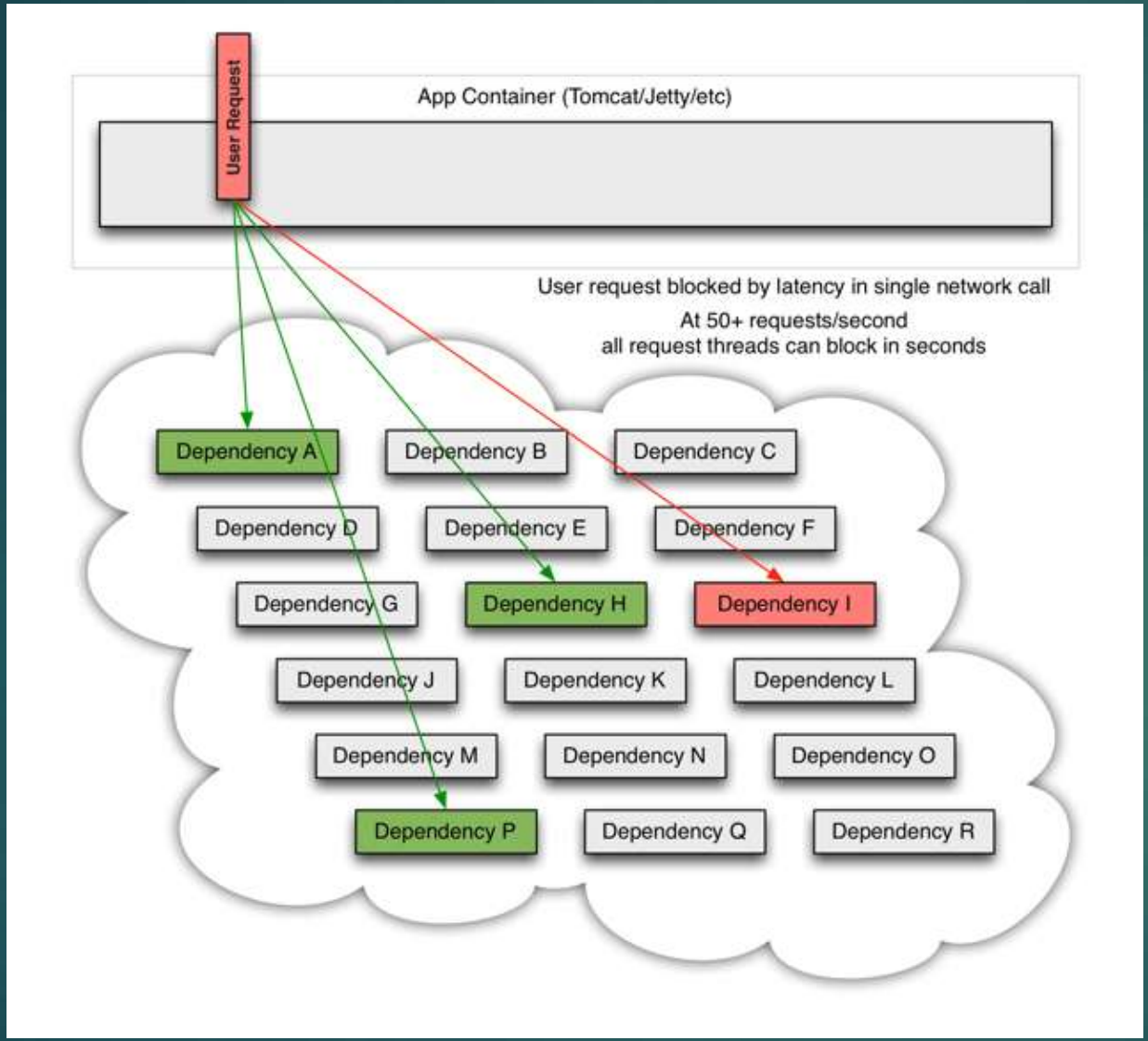
Failure

1 a : omission of occurrence or performance; *specifically* : a failing to perform a duty or expected action <*failure to pay the rent on time*>
b (1) : a state of inability to perform a normal function <*kidney failure*>
b *(2)* : an abrupt cessation of normal functioning <*a power failure*>
c : a fracturing or giving way under stress <*structural failure*>

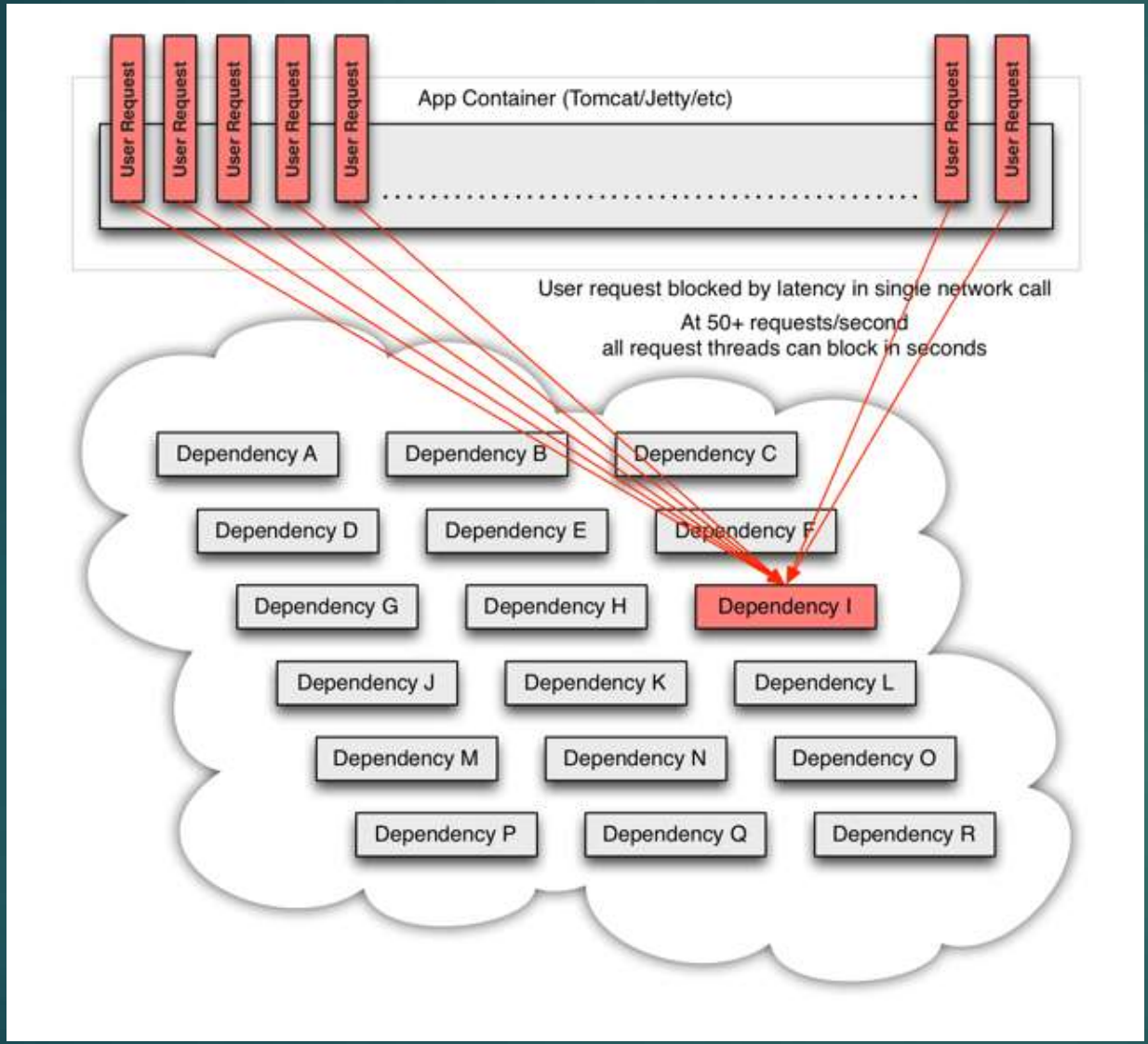
- Merriam-Webster



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



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



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

“It's not what happens to you, but how you react to it that matters.”

EPICTETUS – PHILOSOPHER ~ 55AD

Resilience

1 : the capability of a strained body to recover its size and shape after deformation caused especially by compressive stress

2 : an ability to recover from or adjust easily to misfortune or change

- Merriam-Webster

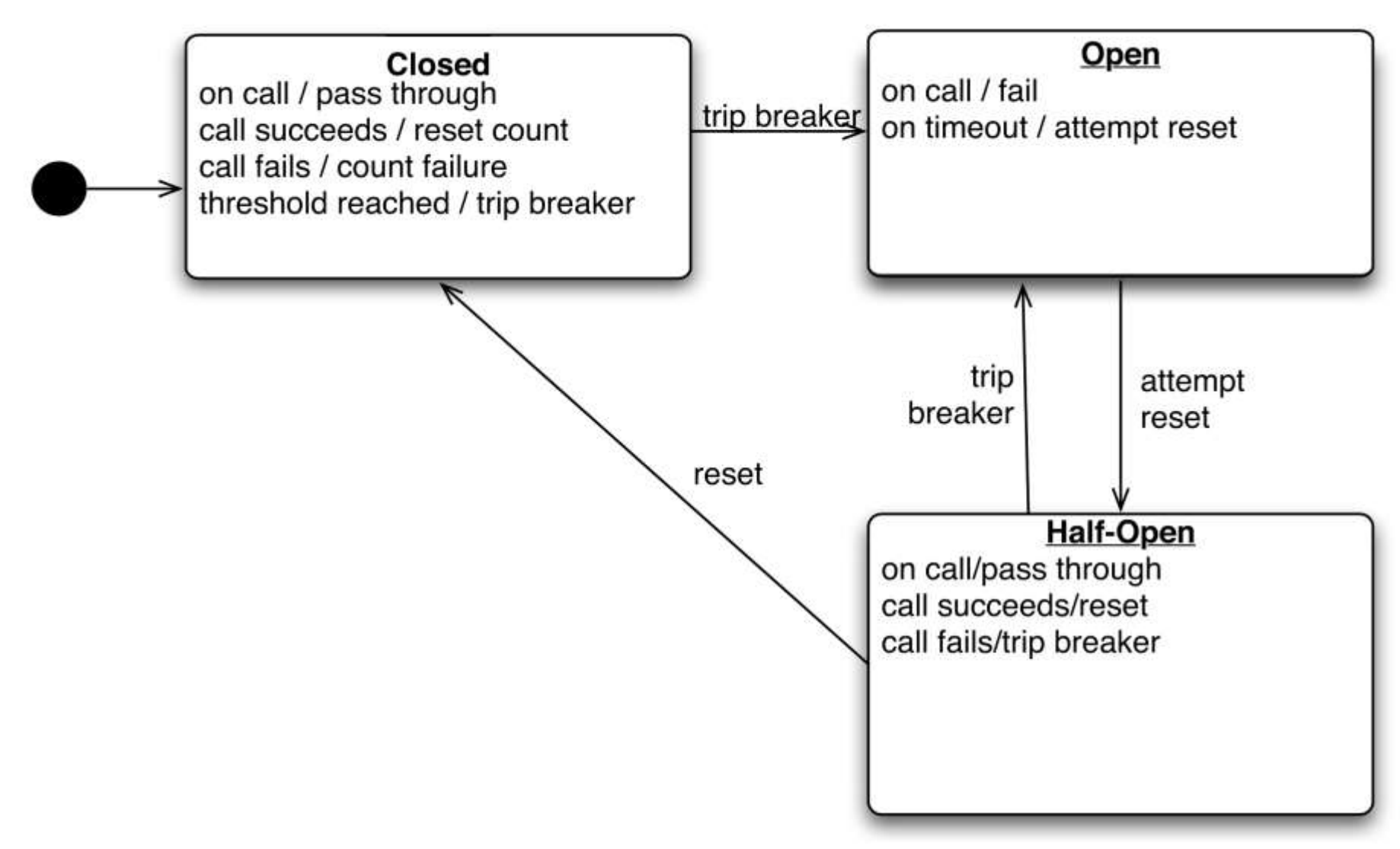
Circuit Breaker – Design Pattern

"A **circuit breaker** is an automatically operated electrical switch designed to protect an electrical **circuit** from damage caused by Overcurrent/overload or short **circuit**. Its basic function is to interrupt current flow after Protective relays detect faults condition."

Wikipedia

"Release It! Design and Deploy Production-Ready Software" (2007)

Michael Nygard


Closed
on call / pass through
call succeeds / reset count
call fails / count failure
threshold reached / trip breaker
trip breaker
Open
on call / fail
on timeout / attempt reset
trip
breaker
attempt
reset
reset
Half-Open
on call/pass through
call succeeds/reset
call fails/trip breaker

Basic Properties

- Pros
  - Fail fast/ (cascading) fallback
- Cons
  - Overhead
- Trip properties
  - Latency
  - Absolute failure count
  - Percentage threshold
- Recovery
  - Retry window

Netflix OSS

- 1998 US DVD, 2007 US streaming, 2010 Canada & beyond.
- 75m subscribers. 190 countries. 125m hrs daily.
- Accounts for 36% of all downstream traffic in North America at primetime.
- Completed 8 year transition to AWS with strong micro service arch
  - Cloud architecture focuses heavily on failure & adaptability
  - Strong proponent of OSS with numerous contributions (203 on github)
    - Chaos Monkey (Simian Army – Chaos Kong)

Spring Cloud

- Spring (1.0.0 released 2002)
  - IoC
  - Source of all things XML
- Spring Boot (1.0.0 April 2014)
  - Convention over configuration
  - Annotation heavy
- Spring Cloud Netflix (1.0.0 March 2015)
  - Simple integration of the Netflix OSS Microservice stack
  - Swappable components

Hystrix – Netflix's Circuit Breaker

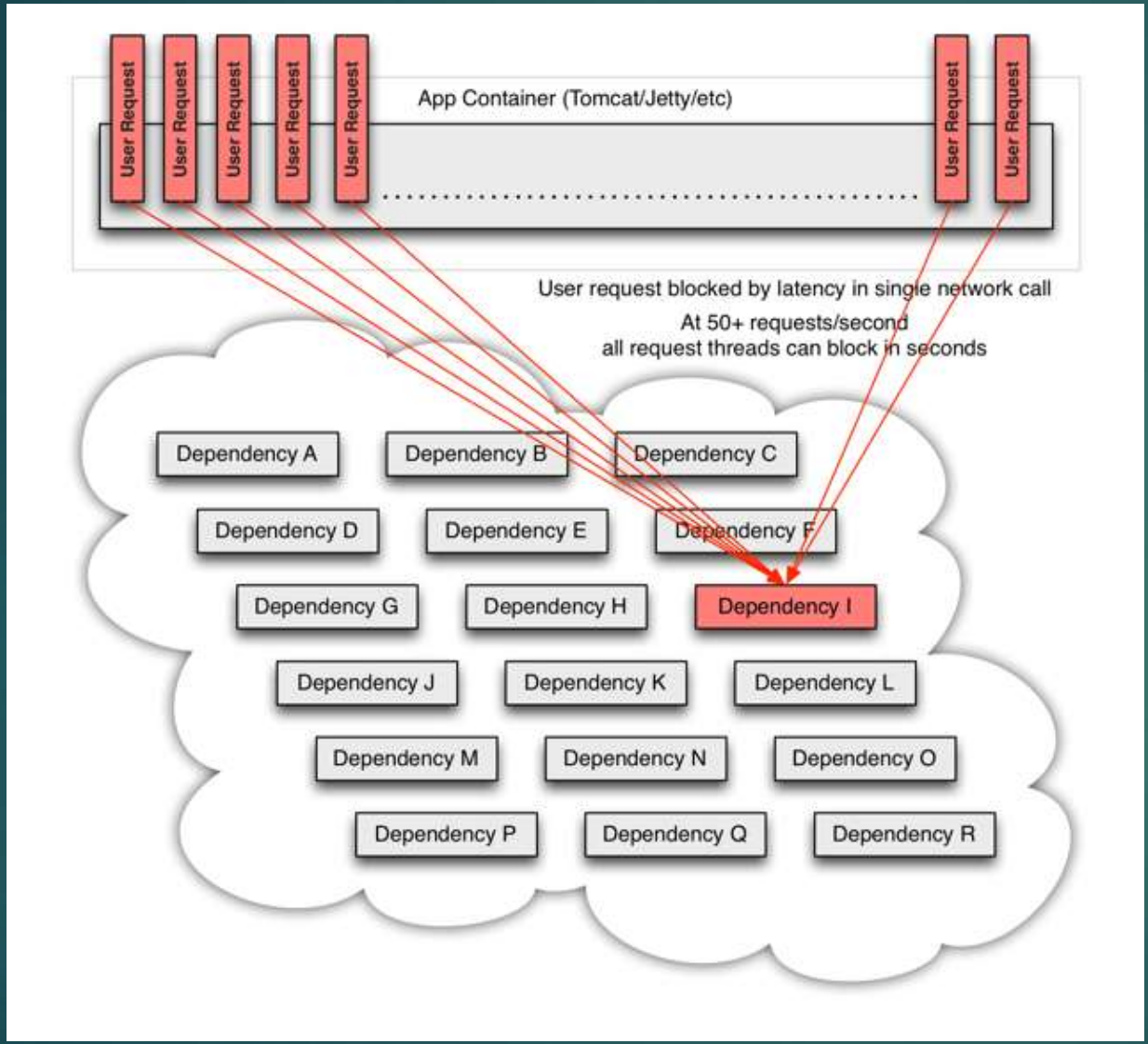
- Announced 2011 http://techblog.netflix.com/2011/12/making-netflix-api-more-resilient.html
- Library facilitating control of the interactions between these distributed services by **adding latency tolerance, fault tolerance logic** and ***near realtime operational insights***.
- Tens of billions of thread-isolated calls are executed via Hystrix every day at Netflix.
- Hystrix library, Javanica, Spring Cloud Hystrix Starter

Configuration

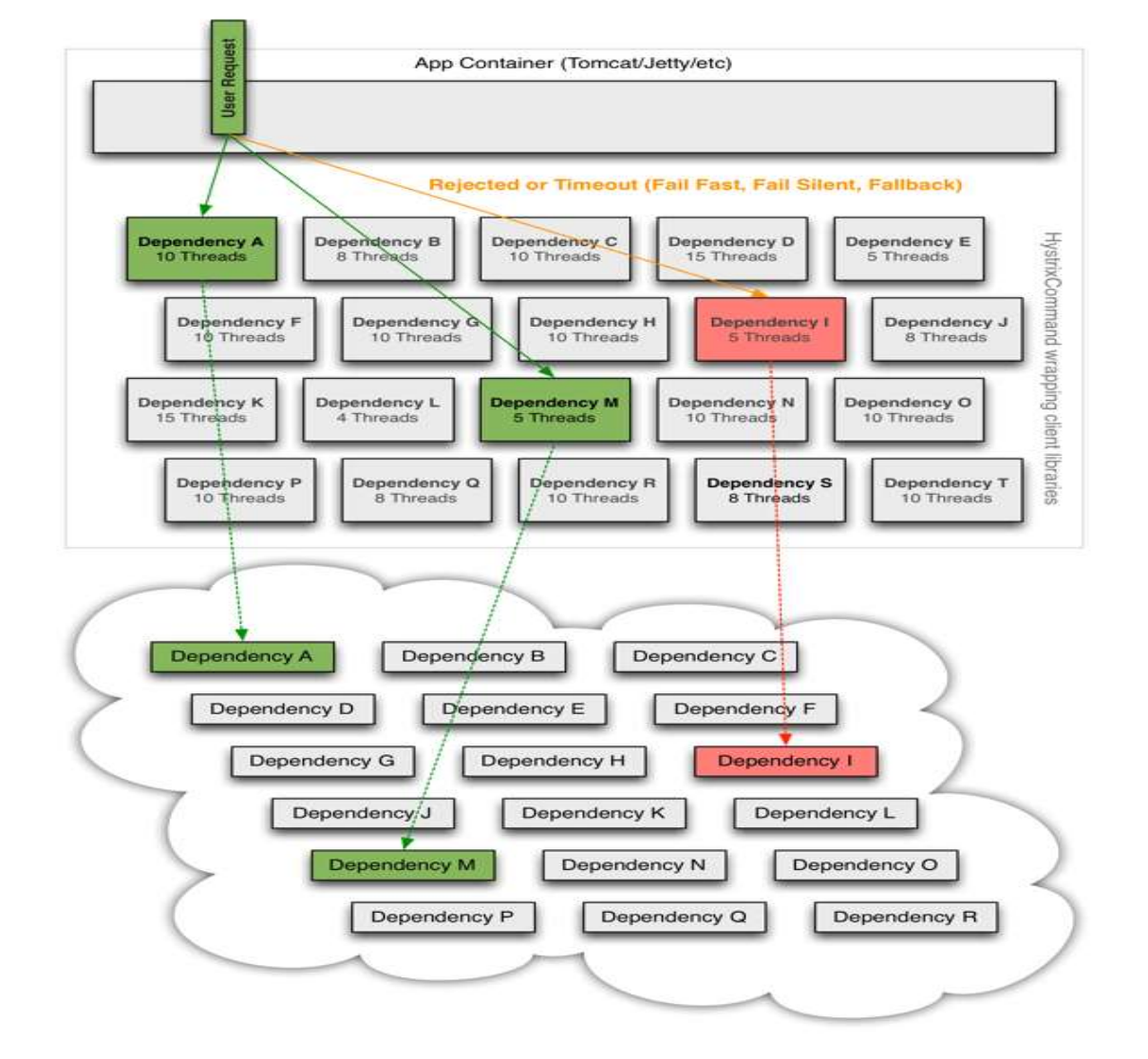
- Circuit breaker (interaction)
  - Timeout
  - Fallback
  - Absolute error threshold
  - Percentage error threshold
  - Retry window
- Global/ per interaction
- Static / Dynamic

Configuration (cont.)

- Metrics (& Collapser)
  - Rolling window size
- Thread pool (external resource)
  - Size (core and queue)



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



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

Thread pools (Bulkheads)

- Shared by multiple commands/actions
  - Sensible defaults
    - GroupKey -> Class name (e.g. PaymentServiceGateway)
    - CommandKey -> Method name (e.g. depositCash)
  - Not configurable on a per-command basis without separate pool
- Resource limiting
- Isolation strategies

Usage

```
@HystrixCommand

@HystrixCommand(groupKey="UserGroup", commandKey =
"GetUserByIdCommand")

@HystrixCommand(fallback='planBMethod')
```

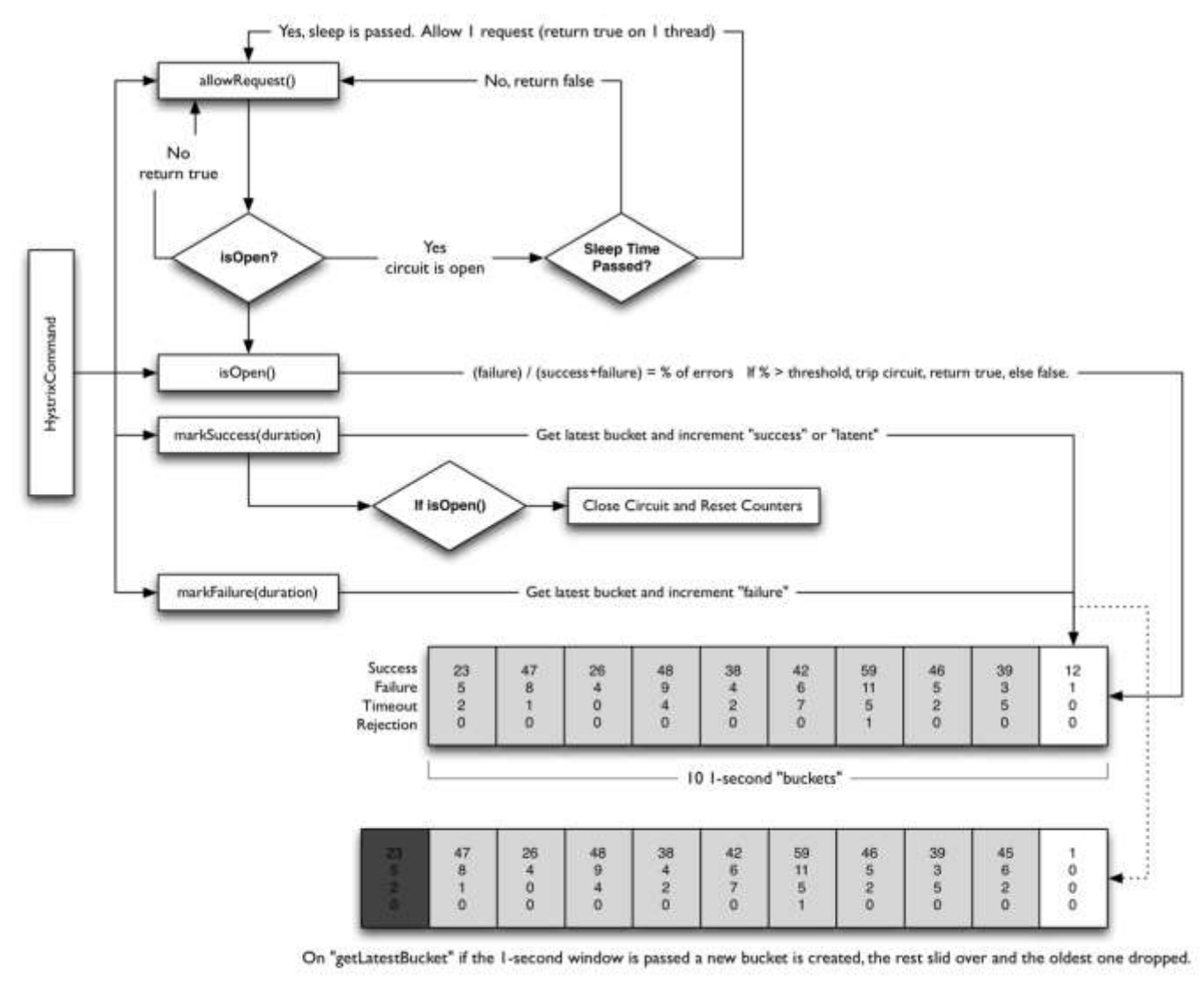
- Synchronous
- Asychronous (Futures)
- Reactive (Observable)

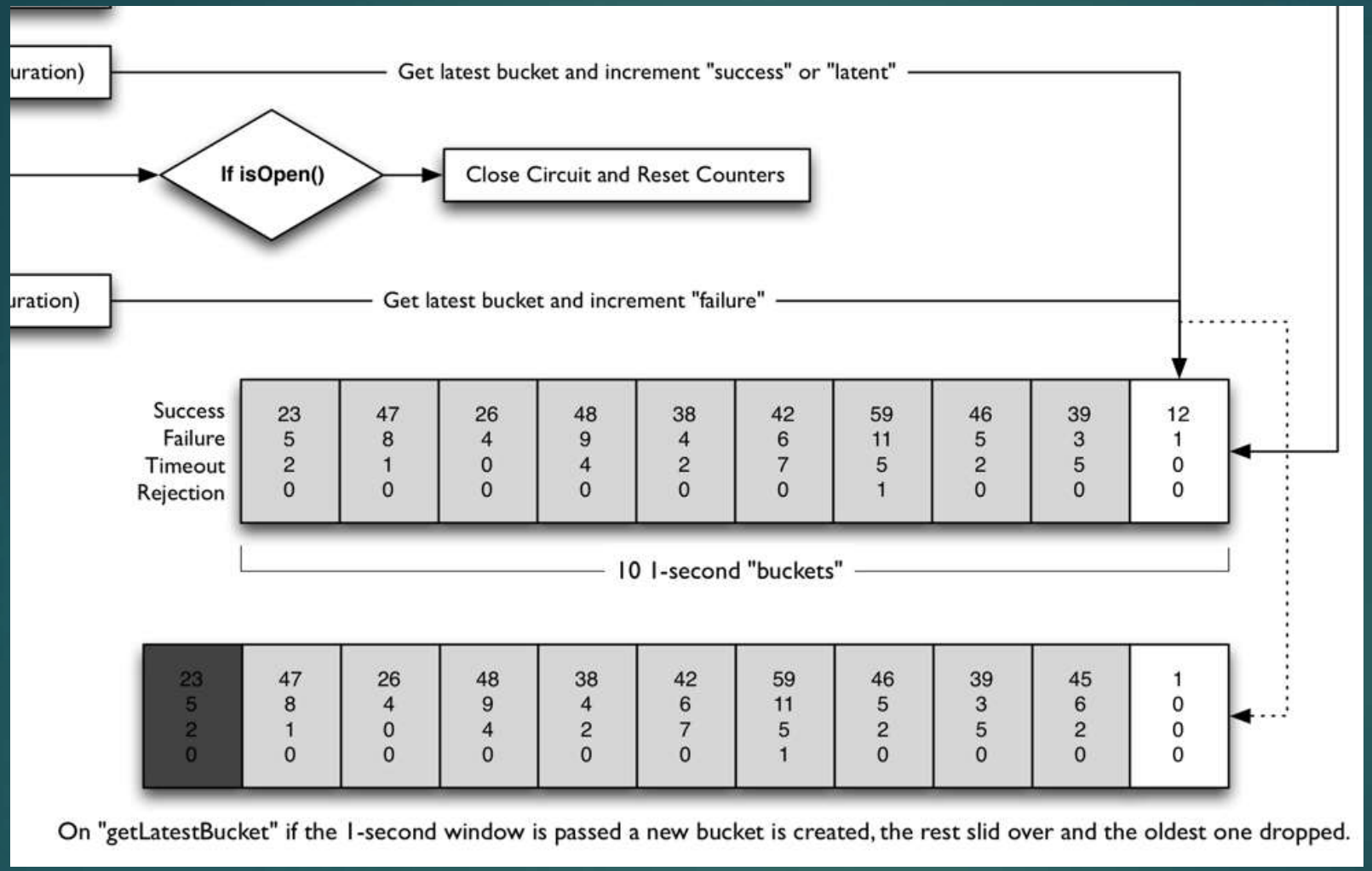
Hystrix Instrumentation

- /health

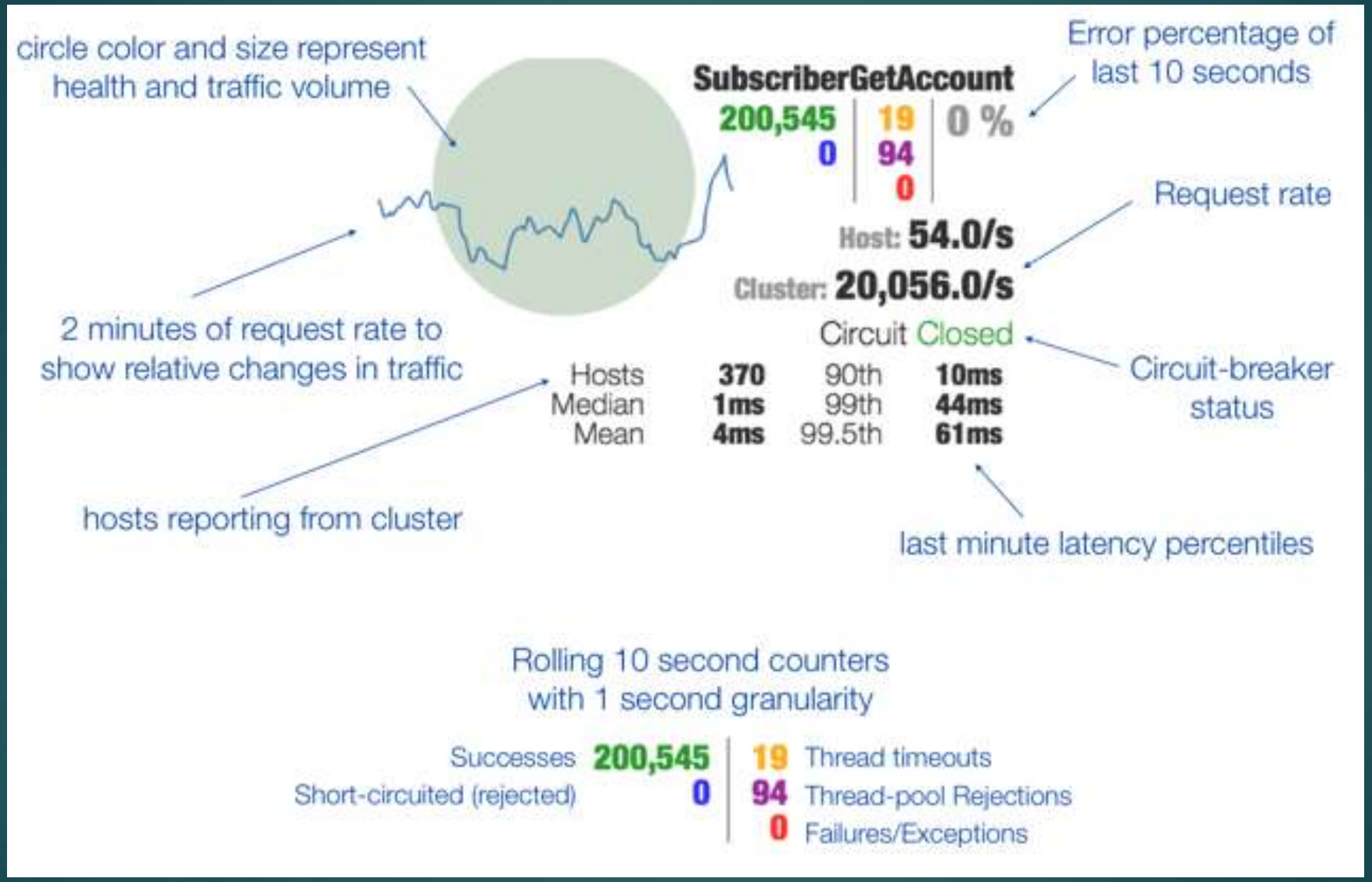
```
{
    "hystrix": {
        "openCircuitBreakers": [
            "StoreServiceGateway::getStoresByZipCode"
        ],
        "status": "CIRCUIT_OPEN"
    },
    "status": "UP"
}
```

- /hystrix.stream
- /metrics


Yes, sleep is passed. Allow 1 request (return true on 1 thread)
allowRequest()
No, return false
No
return true
isOpen?
Yes
circuit is open
Sleep Time
Passed?
HystrixCommand
isOpen()
(failure) / (success+failure) = % of errors If % > threshold, trip circuit, return true, else false.
markSuccess(duration)
Get latest bucket and increment "success" or "latent"
If isOpen()
Close Circuit and Reset Counters
markFailure(duration)
Get latest bucket and increment "failure"
Success 23 47 26 48 38 42 59 46 39 12
Failure 5 8 4 9 4 6 11 5 3 1
Timeout 2 1 0 4 2 7 5 2 5 0
Rejection 0 0 0 0 0 0 1 0 0 0
10 1-second "buckets"
23 47 26 48 38 42 59 46 39 45 1
5 8 4 9 4 6 11 5 3 6 0
2 1 0 4 2 7 5 2 5 2 0
0 0 0 0 0 0 1 0 0 0 0
On "getLatestBucket" if the 1-second window is passed a new bucket is created, the rest slid over and the oldest one dropped.



On "getLatestBucket" if the 1-second window is passed a new bucket is created, the rest slid over and the oldest one dropped.


circle color and size represent health and traffic volume
SubscriberGetAccount
200,545
0
19
94
0
0 %
Error percentage of last 10 seconds
Request rate
Host: 54.0/s
Cluster: 20,056.0/s
Circuit Closed
Circuit-breaker status
2 minutes of request rate to show relative changes in traffic
Hosts 370
Median 1ms
Mean 4ms
90th 10ms
99th 44ms
99.5th 61ms
hosts reporting from cluster
last minute latency percentiles
Rolling 10 second counters with 1 second granularity
Successes 200,545
Short-circuited (rejected) 0
19 Thread timeouts
94 Thread-pool Rejections
0 Failures/Exceptions

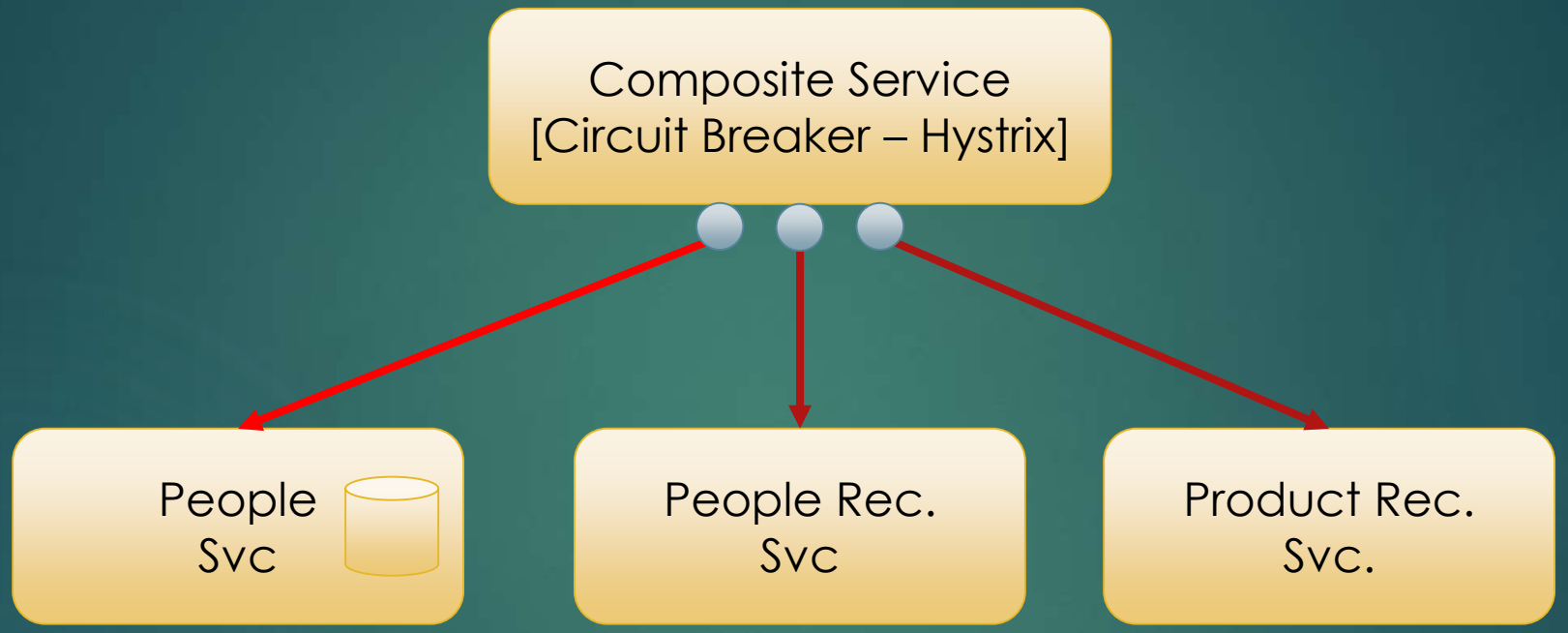
Instrumentation

"Measure what is measurable, and make measurable what is not so".

— Galileo Galilei

"You can't control what you can't measure".

— Tom DeMarco
Controlling Software Projects, Management Measurement & Estimation, (1982)



- Circuit breaker with fallback
- Containerized Spring Boot app with REST API

```
@Service
public class PersonRecommendationServiceGateway {

  @Autowired
  private PersonRecommendationService personRecommendationService;



  public ResponseEntity<Recommendation[]> getPersonRecommendations(int personId) {
    return personRecommendationService.getRecommendations(personId);
  }





}
```

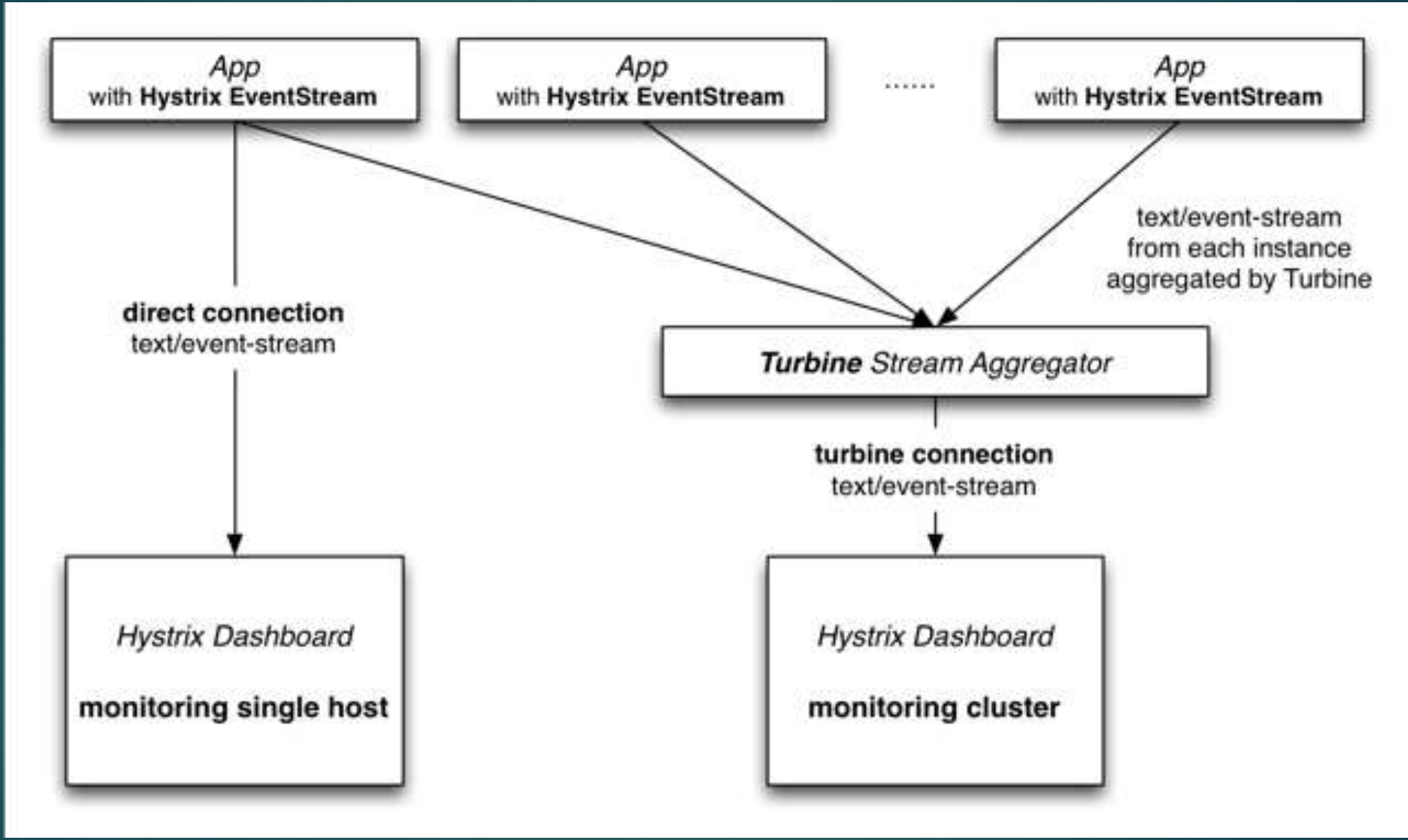
```
@Service
public class PersonRecommendationServiceGateway {

  @Autowired
  private PersonRecommendationService personRecommendationService;

  @HystrixCommand
  public ResponseEntity<Recommendation[]> getPersonRecommendations(int personId) {
    return personRecommendationService.getRecommendations(personId);
  }









}
```

```
@Service
public class PersonRecommendationServiceGateway {

  @Autowired
  private PersonRecommendationService personRecommendationService;

  @HystrixCommand(fallbackMethod = "defaultPersonRecommendations")
  public ResponseEntity<Recommendation[]> getPersonRecommendations(int personId) {
    return personRecommendationService.getRecommendations(personId);
  }

  public ResponseEntity<Recommendation[]> defaultPersonRecommendations(int persontId) {
    Recommendation[] emptyArray = {};
    return new ResponseEntity<Recommendation[]>(emptyArray, HttpStatus.BAD_GATEWAY);
  }
}
```

Demo time!

Recap

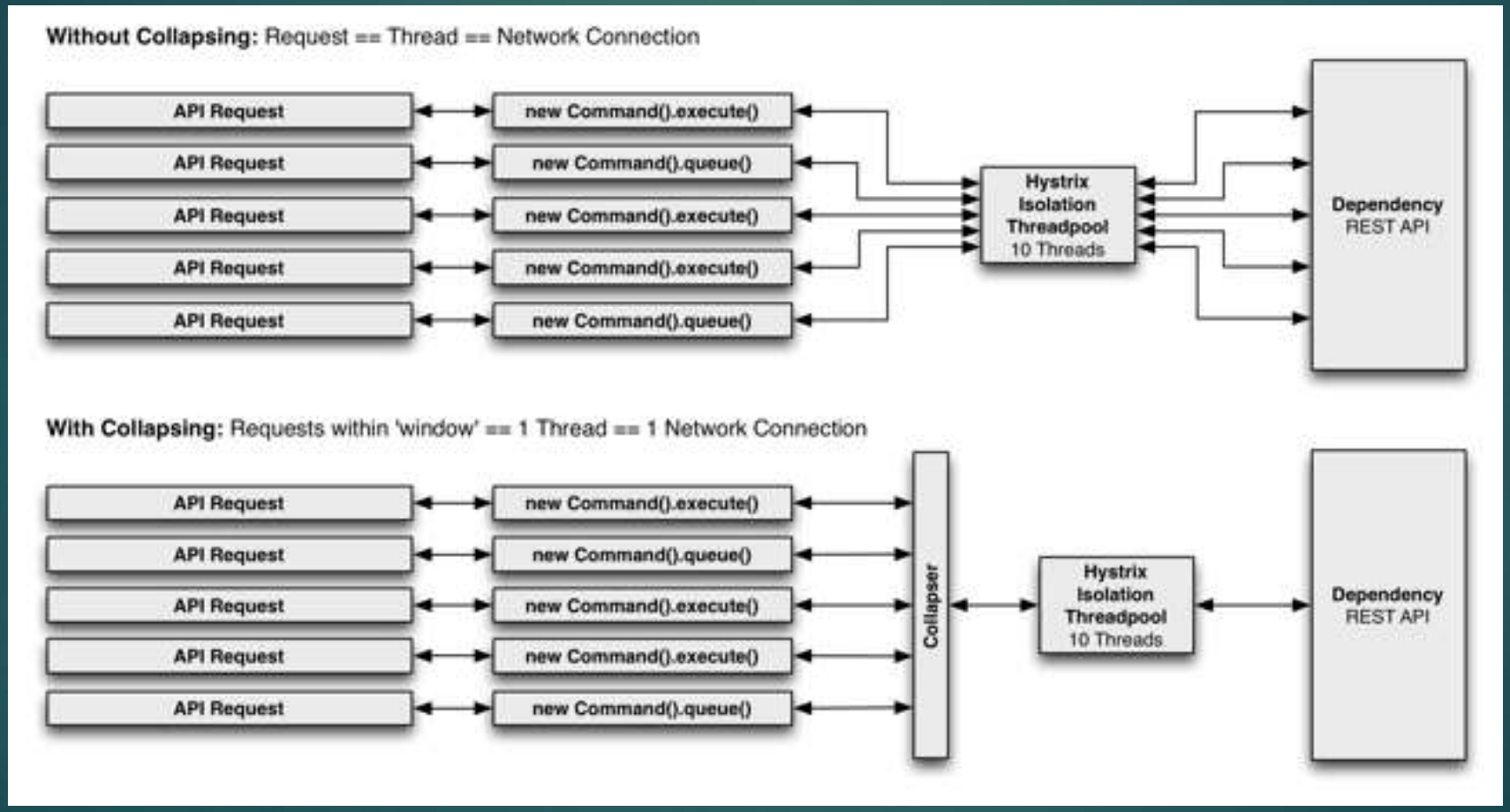
- Why do I care?
  - Not using cloud!
  - Not using Spring!
  - Not using Java!


App
with Hystrix EventStream
App
with Hystrix EventStream
.....
App
with Hystrix EventStream
text/event-stream
from each instance
aggregated by Turbine
direct connection
text/event-stream
Turbine Stream Aggregator
turbine connection
text/event-stream
Hystrix Dashboard
monitoring single host
Hystrix Dashboard
monitoring cluster

But wait … there's more

- Plugin architecture
  - Event notification
  - Metrics publication
  - Property strategy
  - Concurrency strategy
  - Command hook execution
- Additional supported patterns
  - Request caching
  - Rate limiting
  - Request collapsing

Request Collapsing



Request Collapsing (cont…)

```
@HystrixCollapser(batchMethod = "getUserByIds")
public Future<User> getUserById(String id) {
    return null;
}

@HystrixCommand
public List<User> getUserByIds(List<String> ids) {
    List<User> users = new ArrayList<User>();
    for (String id : ids) {
        users.add(new User(id, "name: " + id));
    }
    return users;
}

Future<User> f1 = userService.getUserById("1");
Future<User> f2 = userService.getUserById("2");
Future<User> f3 = userService.getUserById("3");
Future<User> f4 = userService.getUserById("4");
Future<User> f5 = userService.getUserById("5");
```

Local container efficiencies

- Docker image size
  - Java:8 642.4 MB
  - Java:openjdk-8-jre 310.5 MB
  - anapsix/alpine-java:jre8 122 MB
    (x8 + layers adds up… 4gb not including containers)
- JVM defaults
  - -Xmx32m -Xss256k (Memory pool max & thread stack max)
- Monitoring
  - Docker stats
  - cadvisor/ scope

References

- Hystrix - https://github.com/Netflix/Hystrix/wiki
- Hystrix-Javanica - https://github.com/Netflix/Hystrix/tree/master/hystrix-contrib/hystrix-javanica
- Spring Cloud Netflix - http://cloud.spring.io/spring-cloud-netflix/spring-cloud-netflix.html
- Code – https://github.com/singram/spring-cloud-microservices
- 200 + Circuit breakers on github https://github.com/search?utf8=%E2%9C%93&q=circuit+breaker&type=Repositories&ref=searchresults

Thank You!

@arethoseclams
Stuartingram.com
https://github.com/singram/spring-cloud-microservices

Questions?