THE DARK SIDE OF MICRO-SERVICES

@NICOLAS_FRANKEL

ME, MYSELF AND I

- **Developer/Software -/ Solution Architect**
  - Java
  - As consultant

MICRO-SERVICES ON PAPER

QUOTE OF THE DAY

"[…] if people can't build monoliths properly, microservices won't help"

- Simon Brown

https://twitter.com/simonbrown/status/573072777147777024

https://genehughson.wordpress.com/2015/03/08/microservice-architectures-arent-for-everyone/

QUOTE OF THE DAY

"I see you have a poorly structured monolith. Would you like me to convert it into a poorly structured set of microservices?"

- Architect Clippy

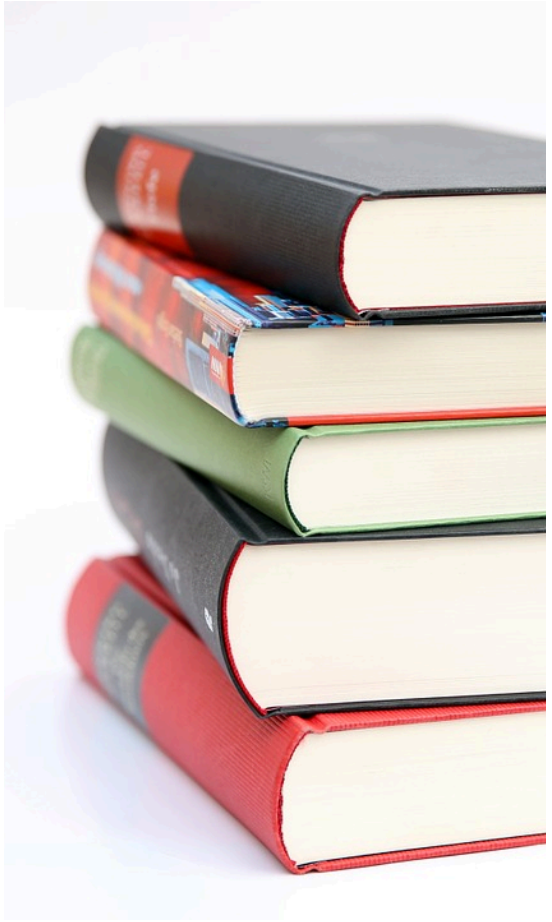
https://twitter.com/architectclippy/status/570025079825764352

https://genehughson.wordpress.com/2015/03/08/microservice-architectures-arent-for-everyone/

MICRO-SERVICES IN REAL LIFE

SUMMARY

- What are micro-services?
- What benefits to expect?
- What requirements?
- What's in it for you now?



A TENTATIVE DEFINITION

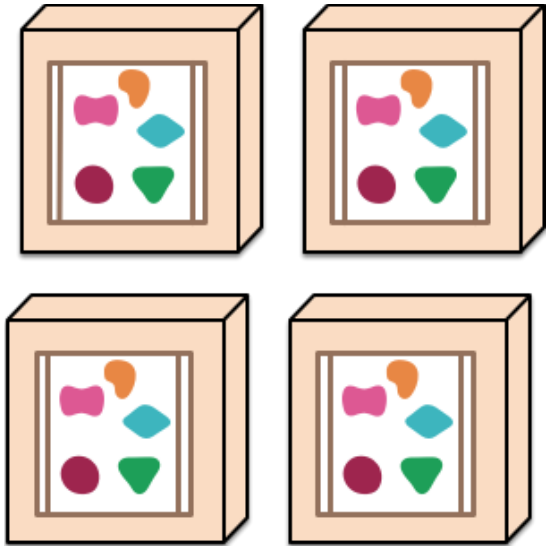
THE MONOLITH

- **"A monolithic application puts all its functionality into a single process…"**

http://martinfowler.com/articles/microservices.html

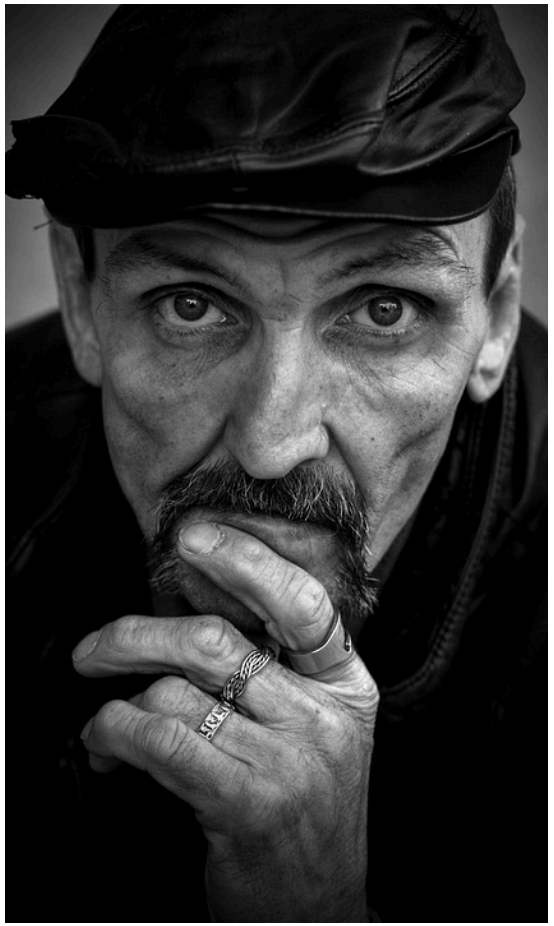
SCALING A MONOLITH

➢ **"… and scales by replicating the monolith on different servers"**



ASSUMPTION

- **All functionalities of your application have the same load**
  - Really?

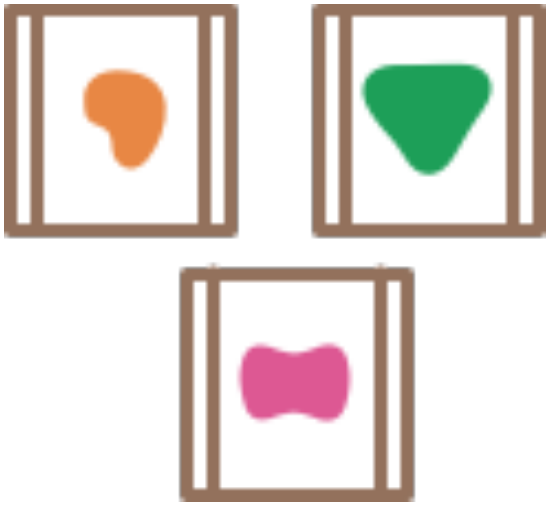


EXAMPLE: E-COMMERCE SHOP

- Catalog
- Cart
- Login
- Payment
- etc.

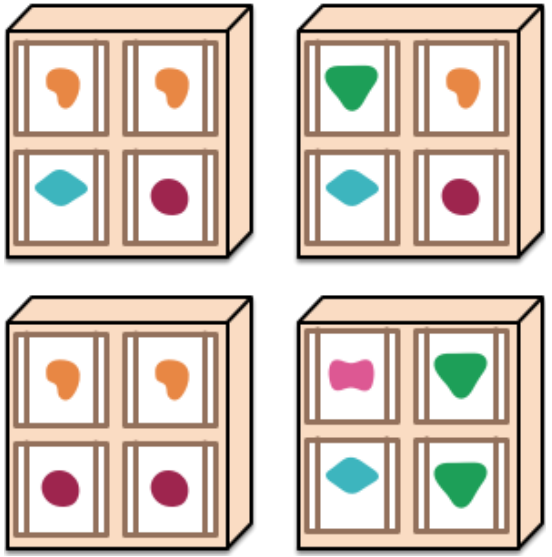
MICRO-SERVICES

- **"A microservices architecture puts each element of functionality into a separate service…"**



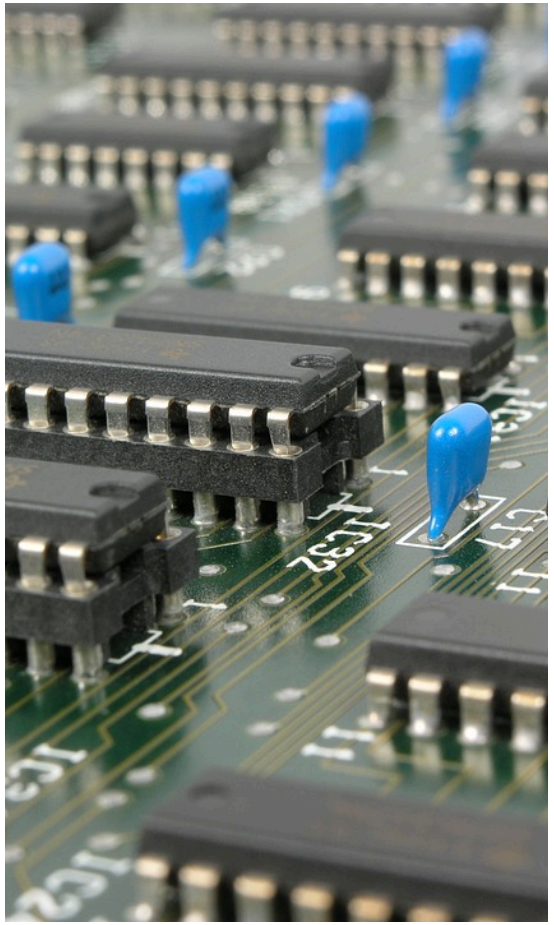
SCALING MICRO-SERVICES

- "… and scales by distributing these services across servers, scaling as needed"



SINGLE RESPONSIBILITY PRINCIPLE

- **Component-based**
- **Wrapping a single business capability**
  - Fine-grained
- **Decoupled from one another**
- **Communicating via simple channels**



BENEFITS

DIFFERENTIAL SCALING

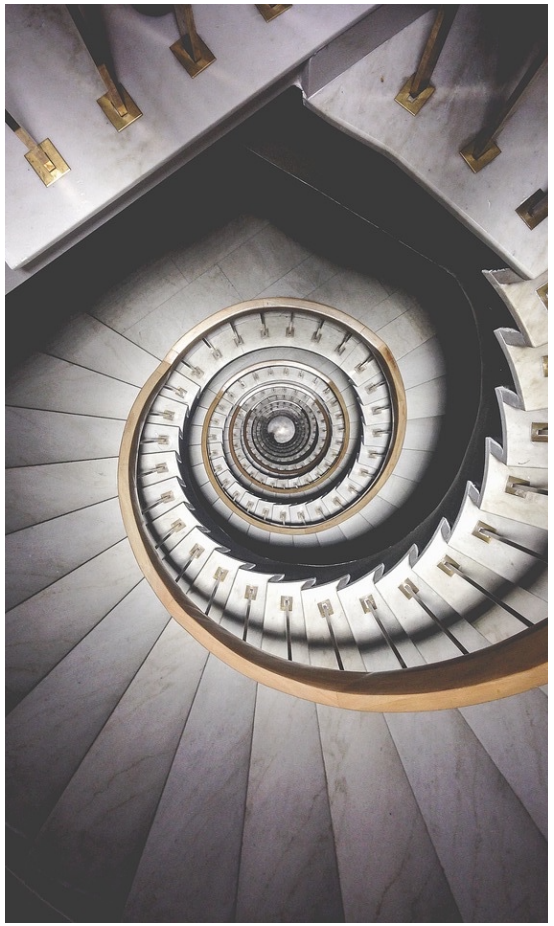
- Optimizing resources

DECOUPLED LIFECYCLES

- **Enable continuous delivery**
  - No more release trains!

CONTINUOUS/EVOLUTIONARY DESIGN

- **No necessary fixed upfront design**
  - Allows for continuous changes

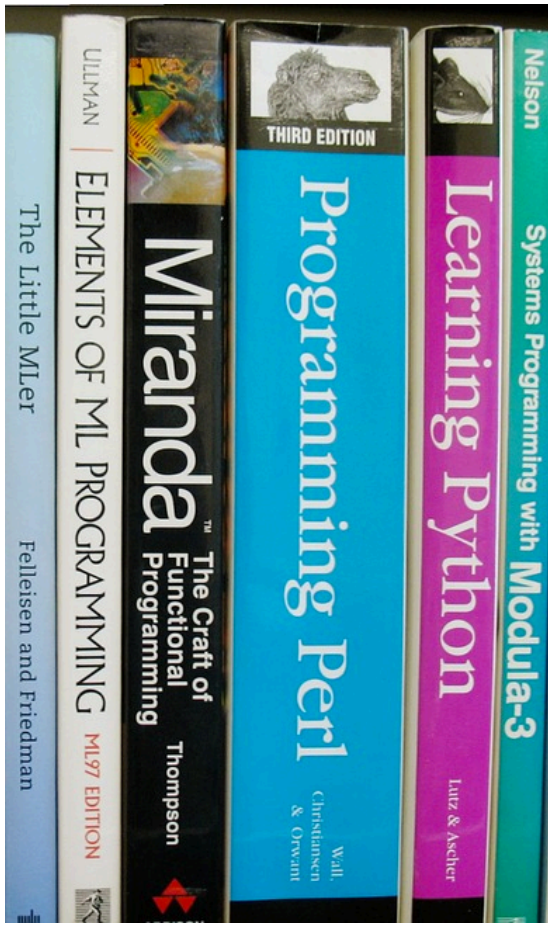


POLYGLOT PERSISTENCE

- **Choose the best persistence tier for the job**

POLYGLOT

- Choose the best language for the job



REQUIREMENTS

INFRASTRUCTURE AUTOMATION

- **To deliver continuously**

FORMALIZED DOCUMENTATION

- To help service users

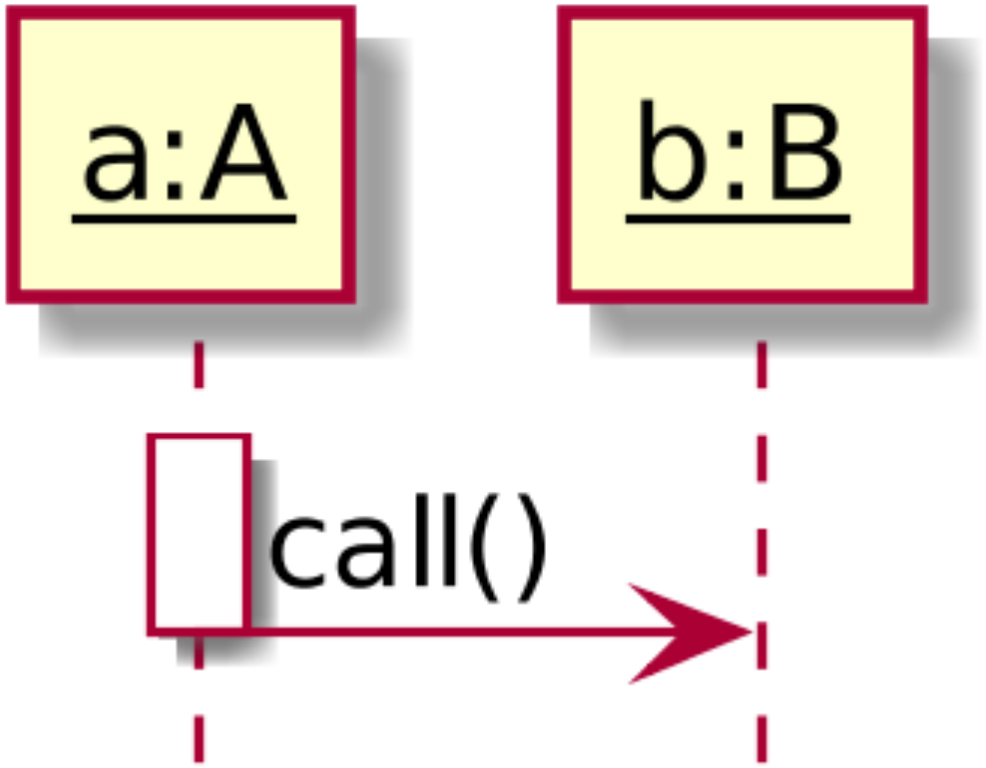
MONITORING

- Cross-components

12-FACTOR APPS

1. **Codebase**
   - One codebase tracked in revision control, many deploys
2. **Dependencies**
   - Explicitly declare and isolate dependencies
3. **Config**
   - Store config in the environment
4. **Backing Services**
   - Treat backing services as attached resources
5. **Build, release, run**
   - Strictly separate build and run stages
6. **Processes**
   - Execute the app as one or more stateless processes
7. **Port binding**
   - Export services via port binding
8. **Concurrency**
   - Scale out via the process model
9. **Disposability**
   - Maximize robustness with fast startup and graceful shutdown
10. **Dev/prod parity**
    - Keep development, staging, and production as similar as possible
11. **Logs**
    - Treat logs as event streams
12. **Admin processes**
    - Run admin/management tasks as one-off processes

CORE ISSUES


a:A
b:B
call()

THE NETWORK!

- **From a simple API call to a network call…**

FALLACIES OF DISTRIBUTED COMPUTING

1. The network is reliable.
2. Latency is zero.
3. Bandwidth is infinite.
4. The network is secure.
5. Topology doesn't change.
6. There is one administrator.
7. Transport cost is zero.
8. The network is homogeneous.

NEW PROBLEMS ARISE

- **Latency**
  - Asynchronicity
  - Performance drop

NEW PROBLEMS ARISE

- **Handling failures**
  - Different handling
  - Unavailable service

NEW PROBLEMS ARISE

- **Serialization/deserialization**
  - Performance drop

STRIKE A BALANCE

- **Between scalability and performance**

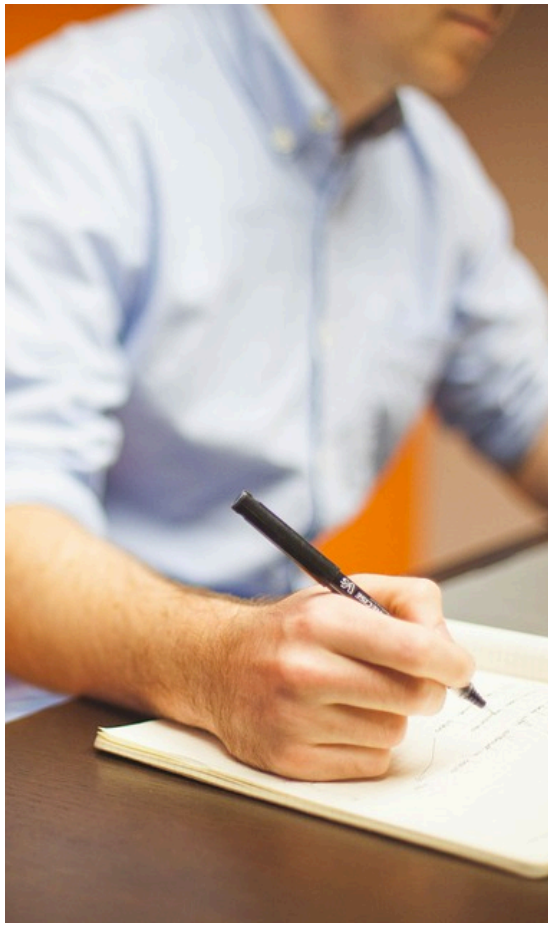
NOT ONLY TECHNICAL ISSUES

AGILE?

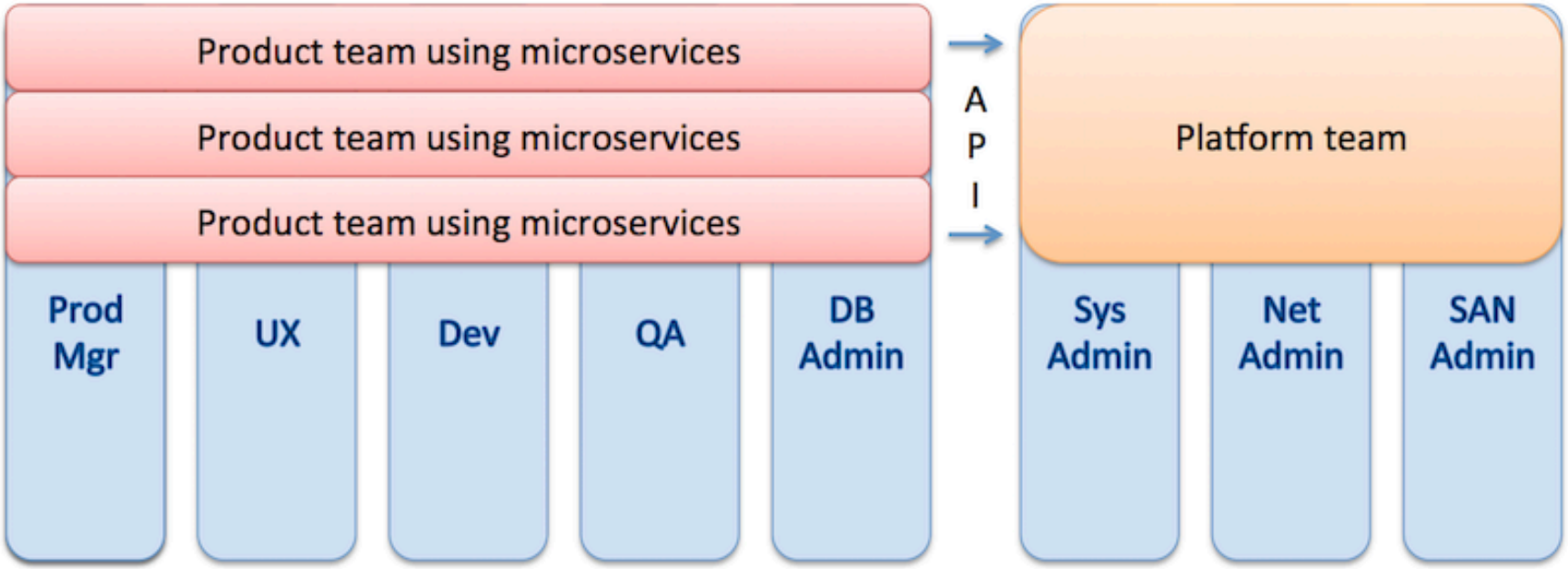
- Trying to do agile
- Doing agile
- Being agile

ORGANIZING TEAMS

- **Limited size**
  - Fed by 2 pizzas
- **Autonomous**
- **Self-organizing**



ORGANIZATION AT NETFLIX



THE BANE OF MIDDLE MANAGEMENT

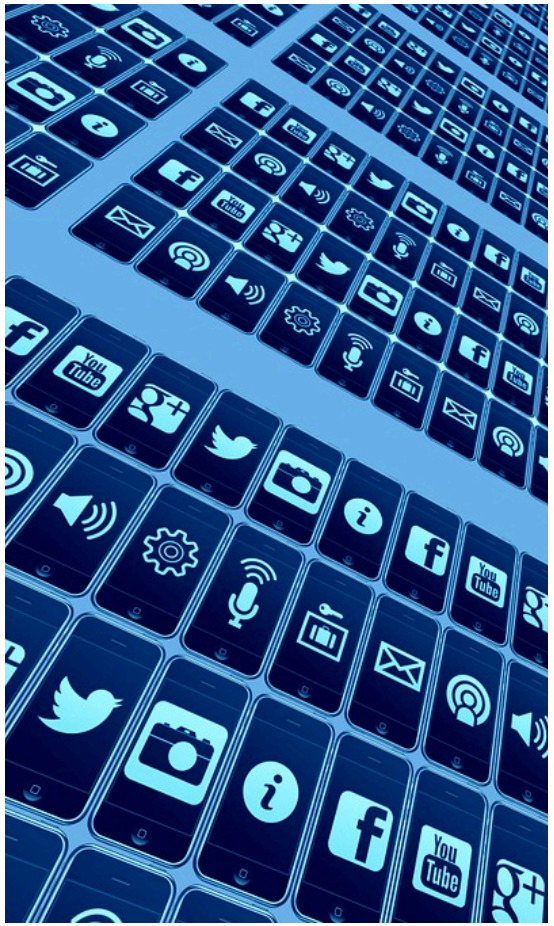
- **Most middle managers are useless**
  - But they want to keep their jobs
- **Better suited to**
  - Small companies
  - Rather flat organizations

THE LIGHT SIDE

AND YET…

THERE ARE MICRO-SERVICES…

- **… and web-services**
  - RESTful



MICRO-SERVICES ENABLERS

- Hystrix
- Ribbon
- Eureka
- Feign

HYSTRIX

DEFEND YOUR APP

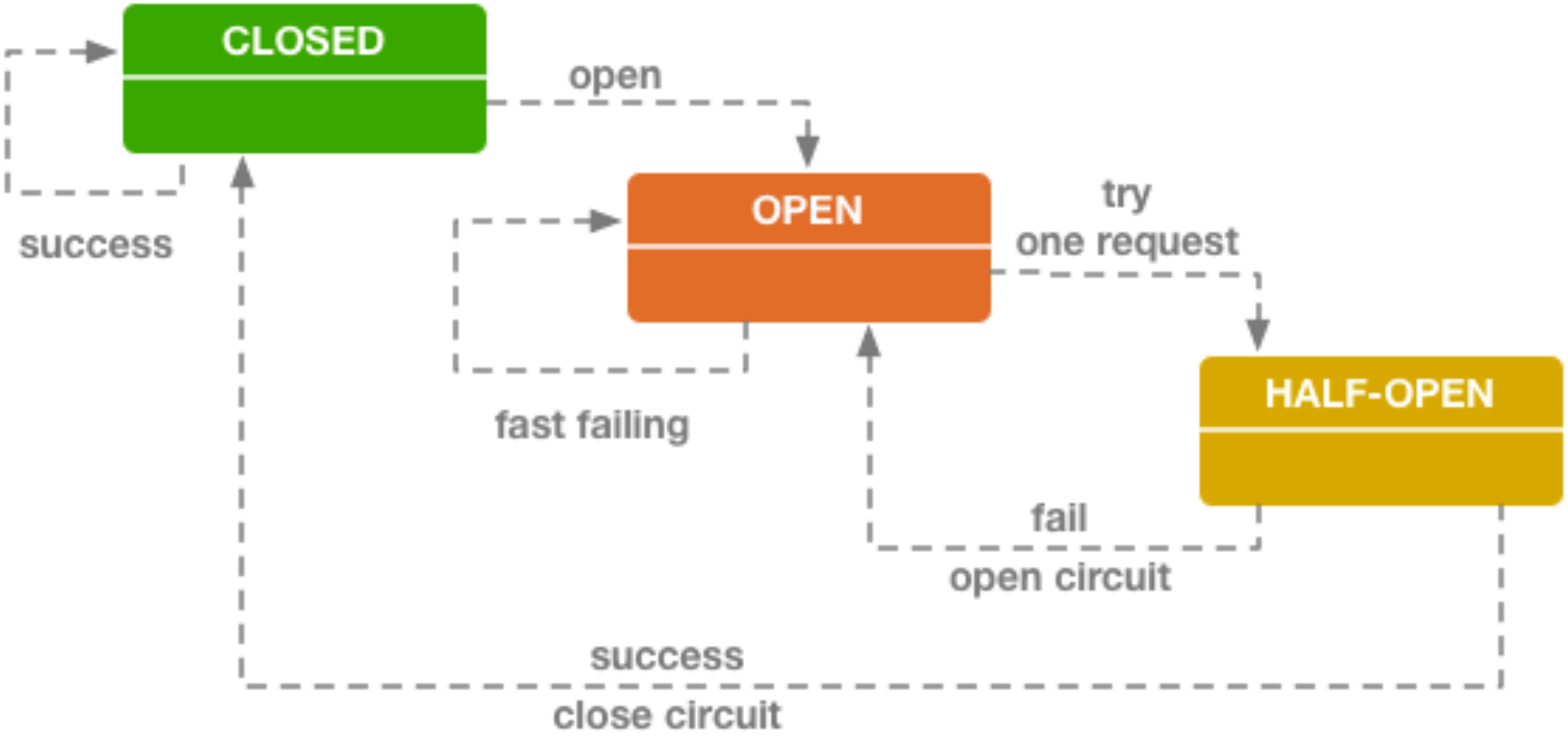
FEATURES

- **Circuit breaker**
  - Fail-fast
- **Fallback**
- **Cache**
- **Collapse**
- **Dashboard**
- **And many more…**

https://github.com/Netflix/Hystrix/wiki/How-To-Use

CIRCUIT-BREAKER

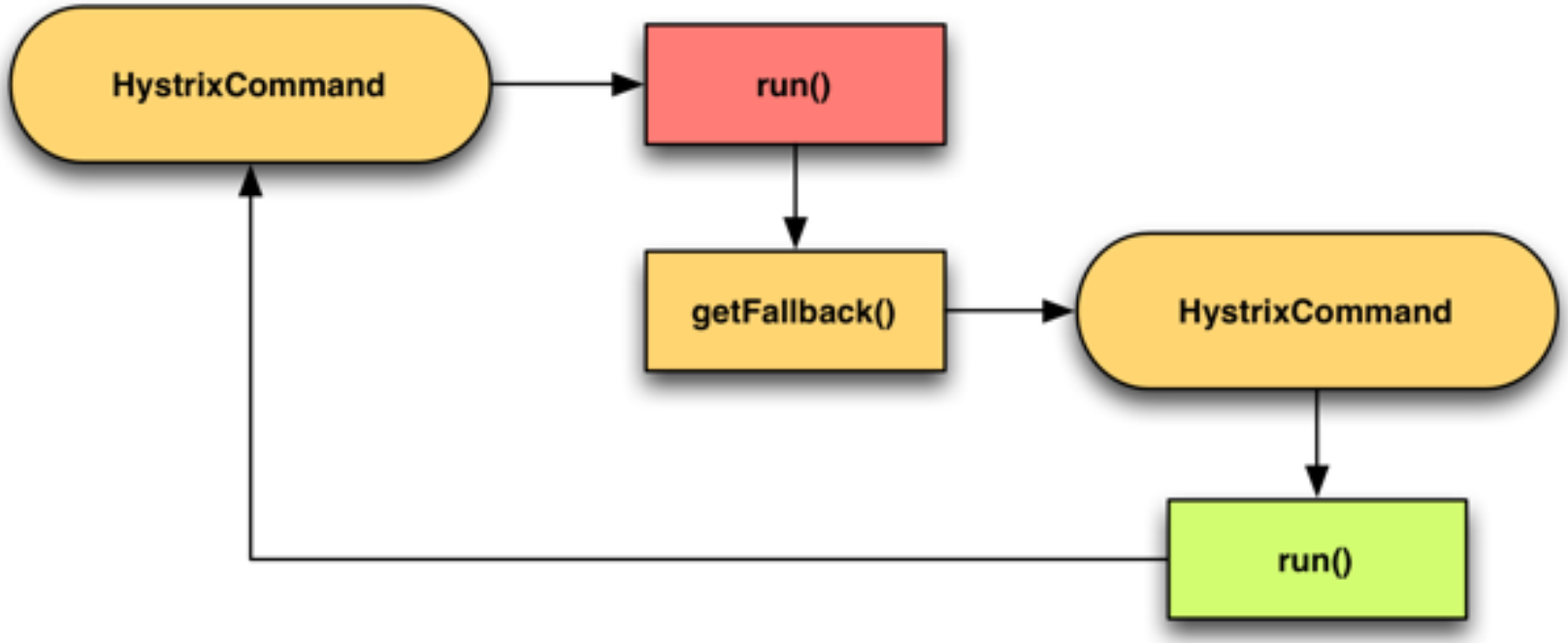
HYSTRIX
DEFEND YOUR APP



Circuit Breaker State Diagram

FALLBACK

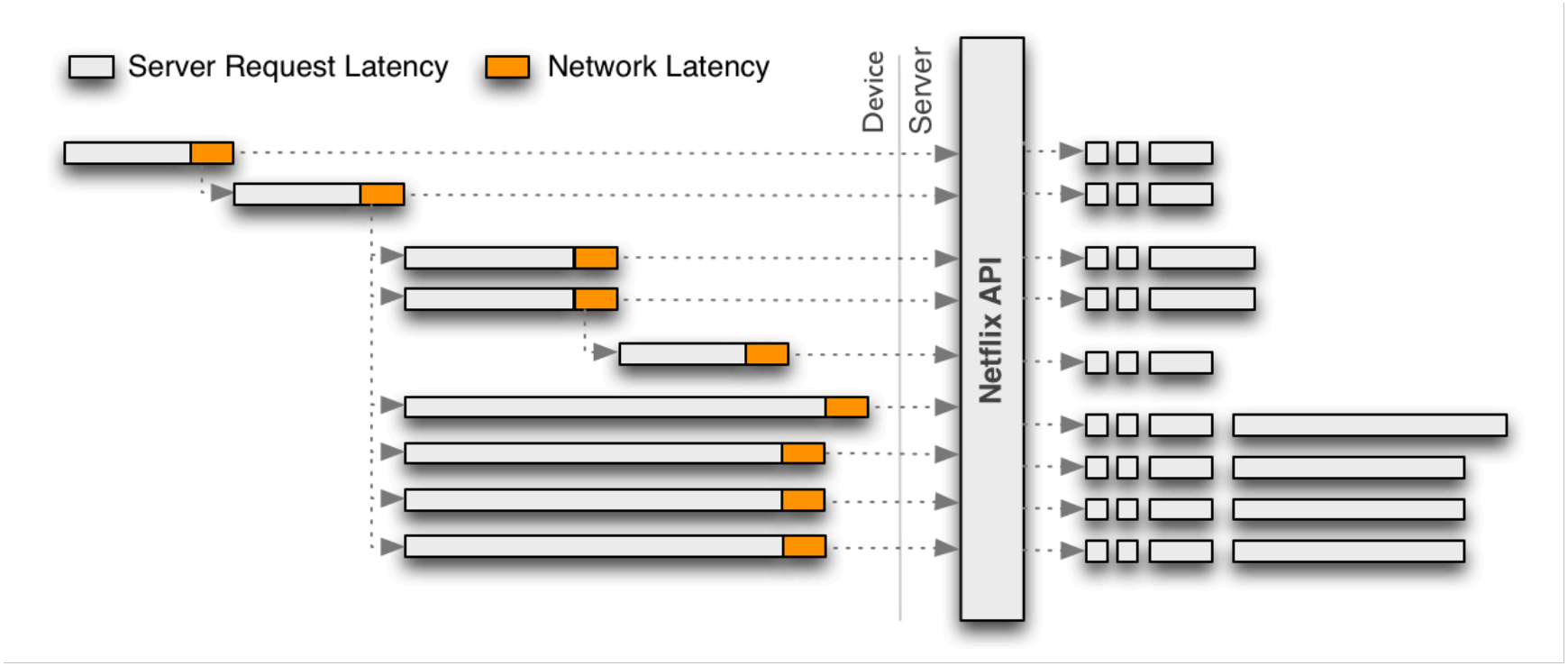
HYSTRIX
DEFEND YOUR APP



REQUEST COLLAPSE

HYSTRIX

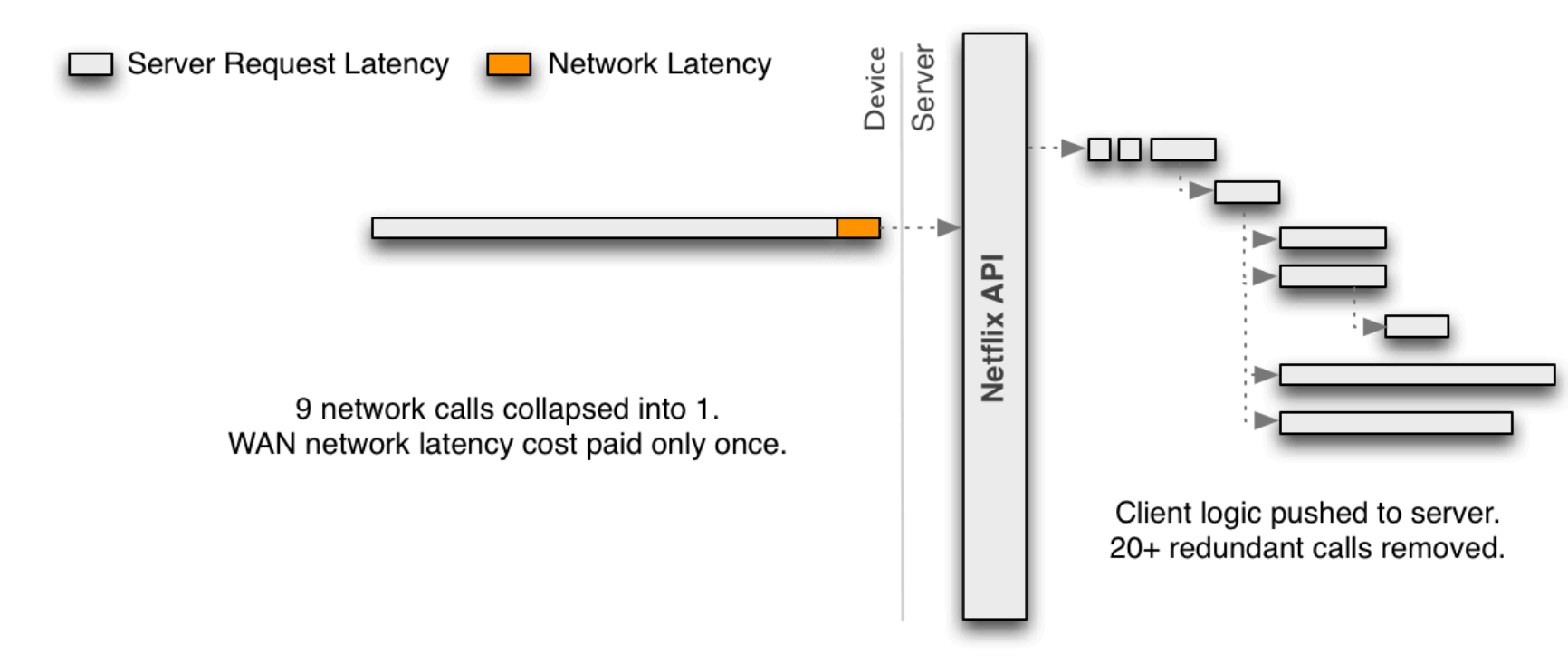
DEFEND YOUR APP



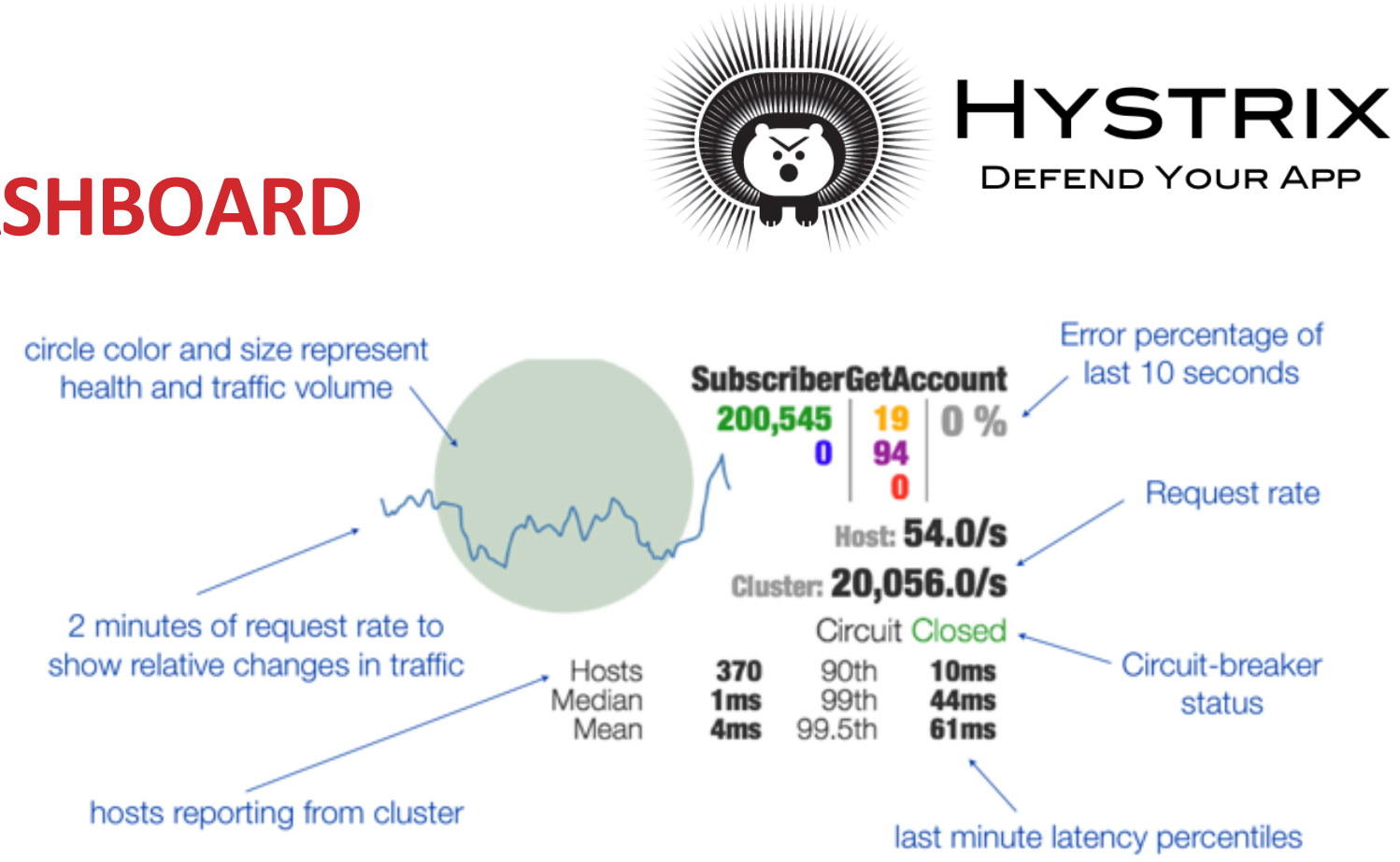
REQUEST COLLAPSE

HYSTRIX

DEFEND YOUR APP



DASHBOARD


HYSTRIX
DEFEND YOUR APP
circle color and size represent health and traffic volume
Error percentage of last 10 seconds
SubscriberGetAccount
200,545
19
0 %
0
94
0
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

HYSTRIX

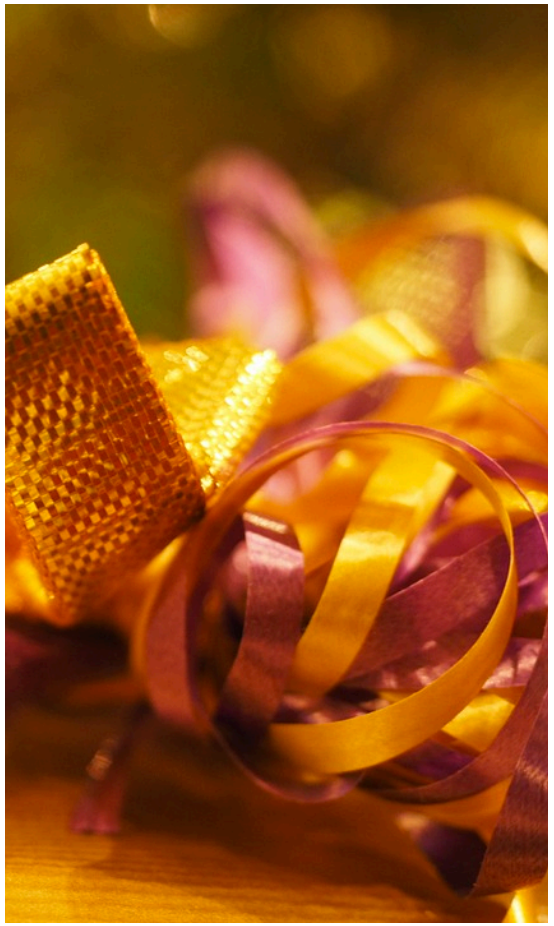
DEFEND YOUR APP

API FLAVORS

- Synchronous
- Asynchronous
- Reactive

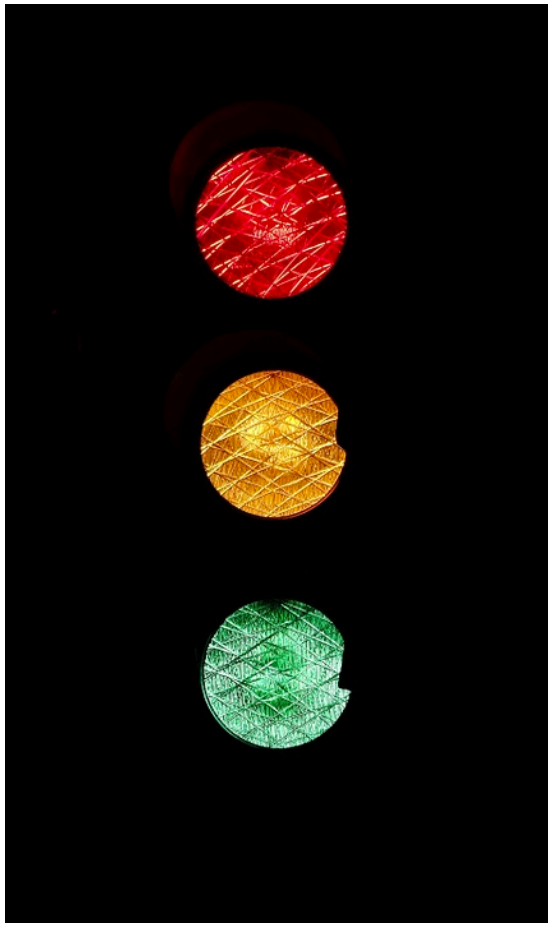
RIBBON

- Client-side load-balancer
- Pluggable rules



COMMONS RULES

- Round robbin
- Availability filtering
- Weighted response time
- <insert your own there>

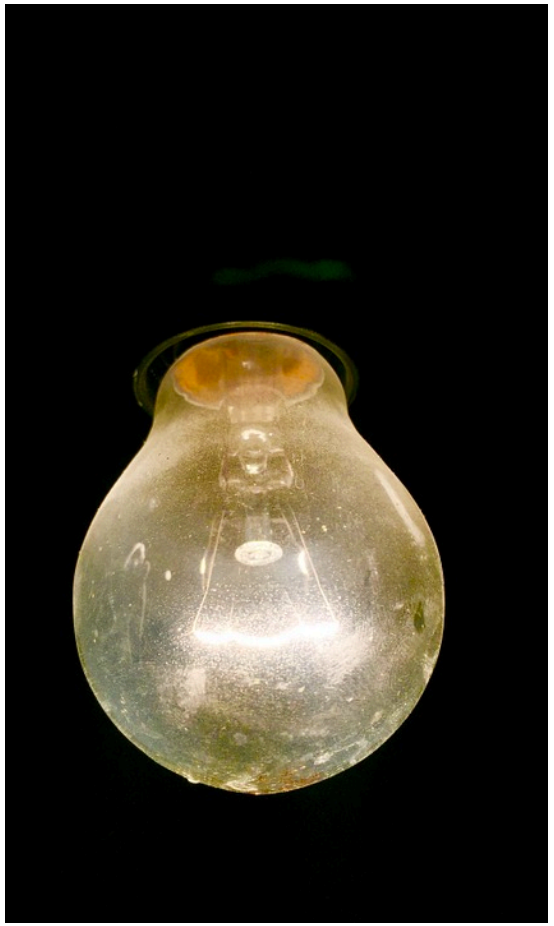


SERVER LIST

- **Ad hoc**
- **Configuration-based**
- **Discovery enabled**
  - Through Eureka

EUREKA

- **Registry for middle tier load balancing**
  - Server
  - Client



FEIGN

- Annotation based REST client

SPRING CLOUD*

SUMMARY

ADOPT AN ENGINEER MINDSET

- **Evaluate benefits**
- **Evaluate costs**
- **Evaluate feasibility**
  - Technical
  - Organizational

Q&A

- http://blog.frankel.ch/
- @nicolas_frankel
- http://frankel.in/