Battle of the Circuit Breakers

Istio vs. Hystrix/Resilience4J

@nicolas_frankel

Me, myself and I

- Developer Advocate
  - Developer/Architect for 17 years
- DevOps and Cloud curious

@nicolas_frankel

hazelcast

Hazelcast

HAZELCAST IMDG is an **operational, in-memory**, distributed computing platform that manages data using in-memory storage, and performs parallel execution for breakthrough application speed and scale.

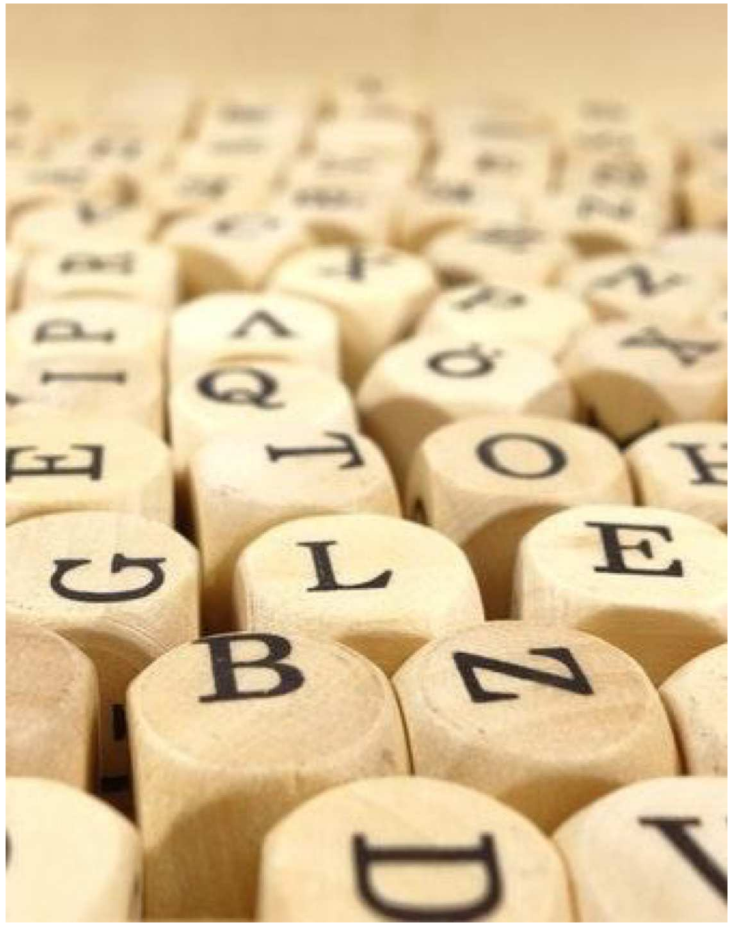
HAZELCAST JET is the ultra fast, application embeddable, 3rd generation stream processing engine for low latency batch and stream processing.

Agenda

- Some introduction
- The problem
- The circuit-breaker pattern
- Istio implementation
- Hystrix implementation
- Demo

µservice: a tentative definition

- Componentization via Services
- Smart endpoints and dumb pipes
- Decentralized Governance
- Decentralized Data Management
- Infrastructure Automation
- Design for failure
- Evolutionary Design
- Organized around Business Capabilities
- Products not Projects



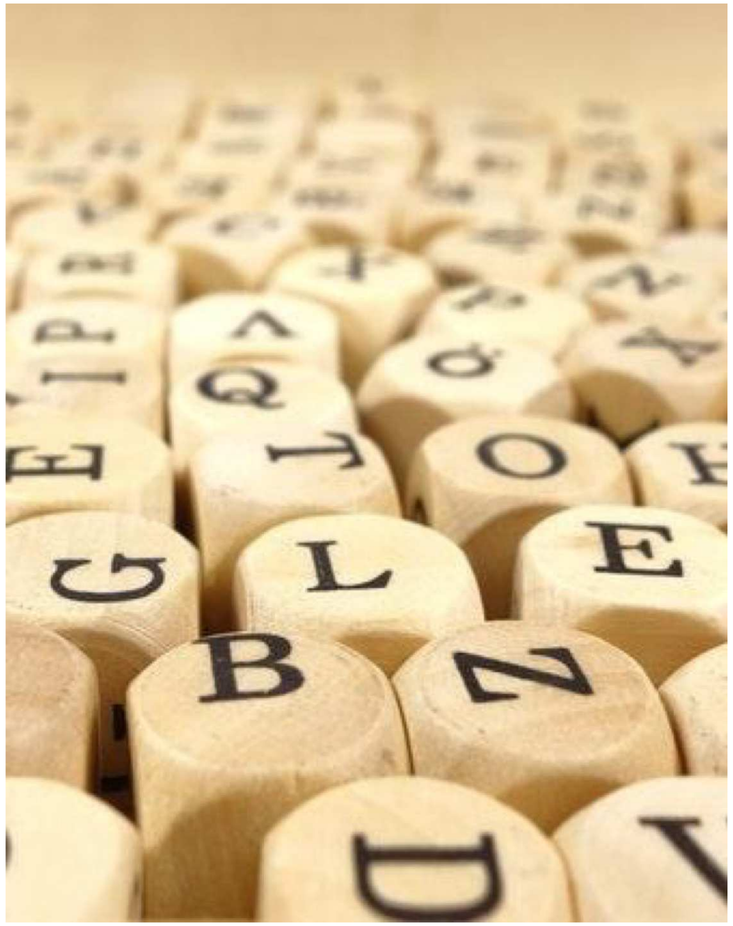
https://martinfowler.com/articles/microservices.html

@nicolas_frankel

hazelcast

µservice: a tentative definition

- Componentization via Services
- Smart endpoints and dumb pipes
- Decentralized Governance
- Decentralized Data Management
- Infrastructure Automation
- Design for failure
- Evolutionary Design
- Organized around Business Capabilities
- Products not Projects



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

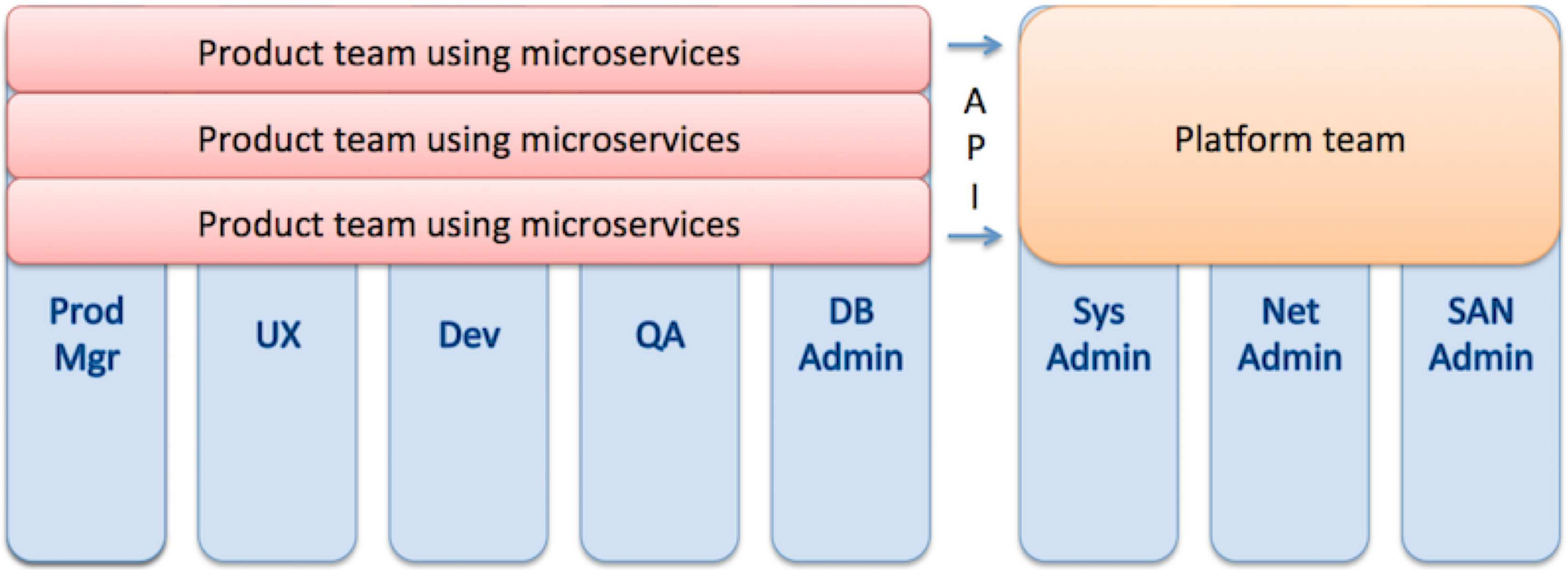
Word of warning

- Microservices are an organizational solution to an organizational problem
- They are ill-adapted to most orgs

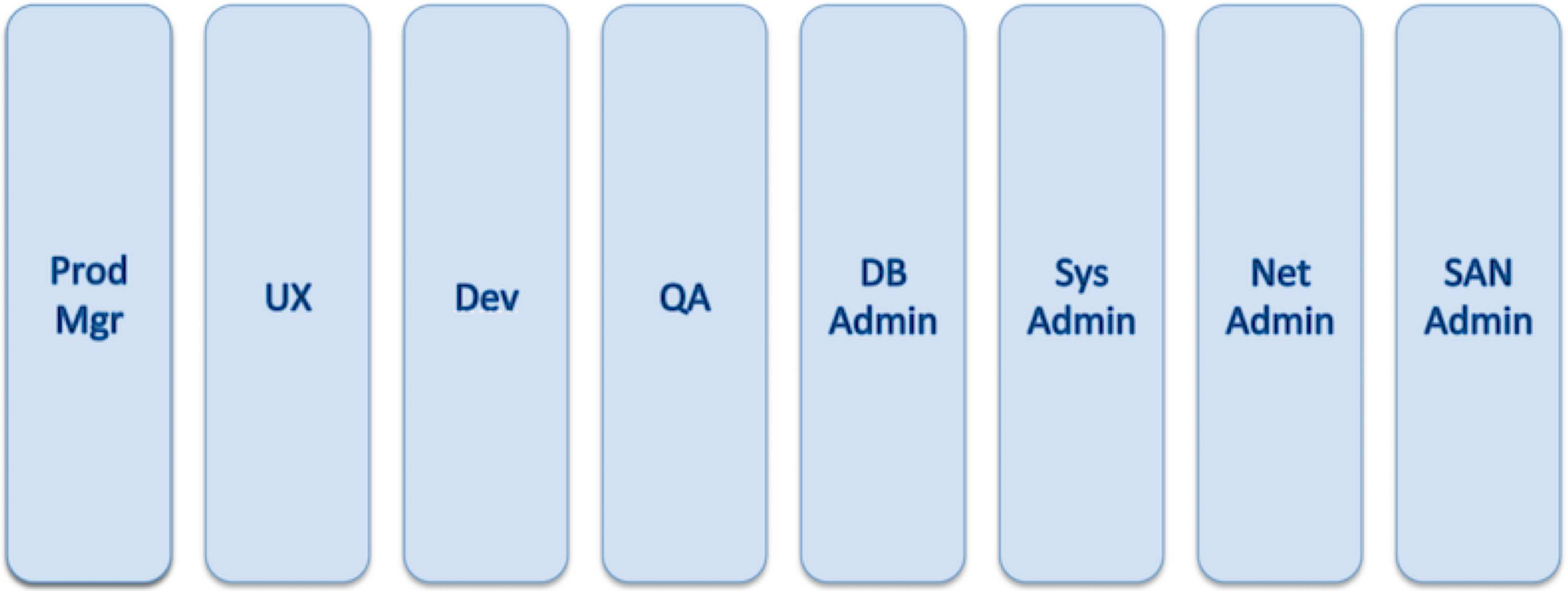
https://martinfowler.com/bliki/MicroservicePrerequisites.html

Conway’s Law

"organizations which design systems … are constrained to produce designs which are copies of the communication structures of these organizations."


Product team using microservices
Product team using microservices
Product team using microservices
A
P
I
Platform team
Prod Mgr
UX
Dev
QA
DB Admin
Sys Admin
Net Admin
SAN Admin

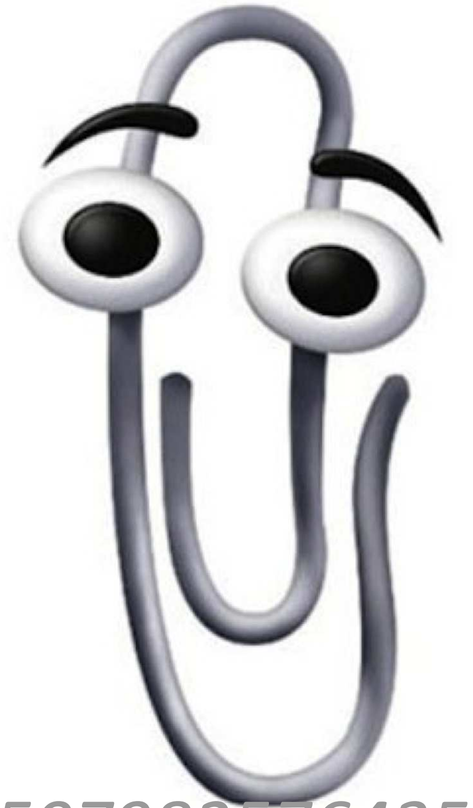
https://www.nginx.com/blog/adopting-microservices-at-netflix-lessons-for-team-and-process-design/



https://www.nginx.com/blog/adopting-microservices-at-netflix-lessons-for-team-and-process-design/

Rant of the day

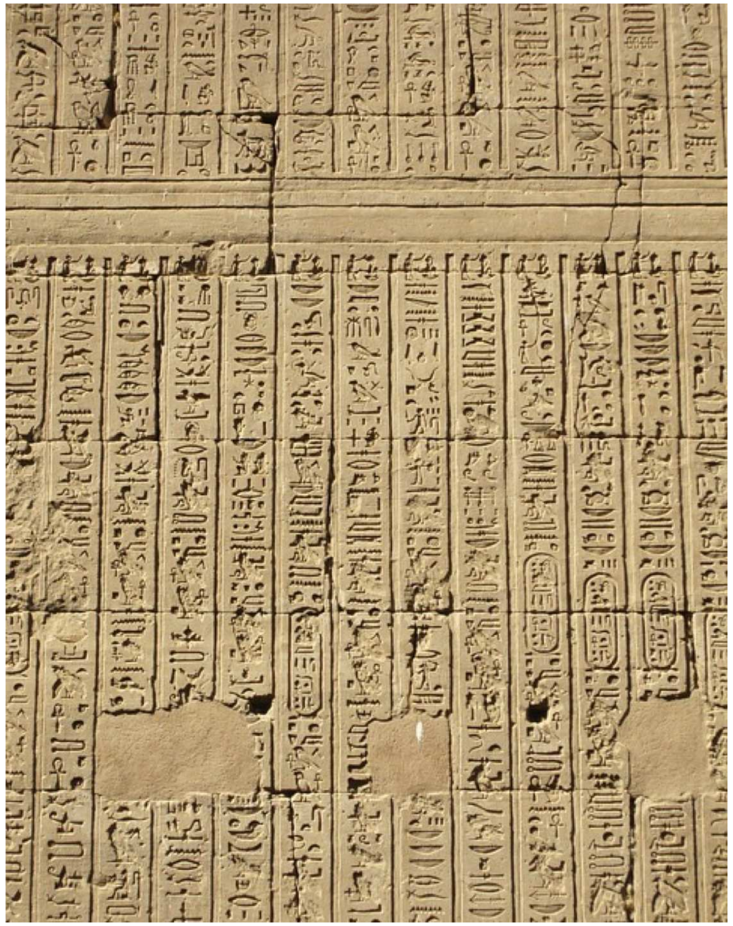
"I see you have a poorly structured monolith. Would you like me to convert it into a poorly structured set of microservices?"



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

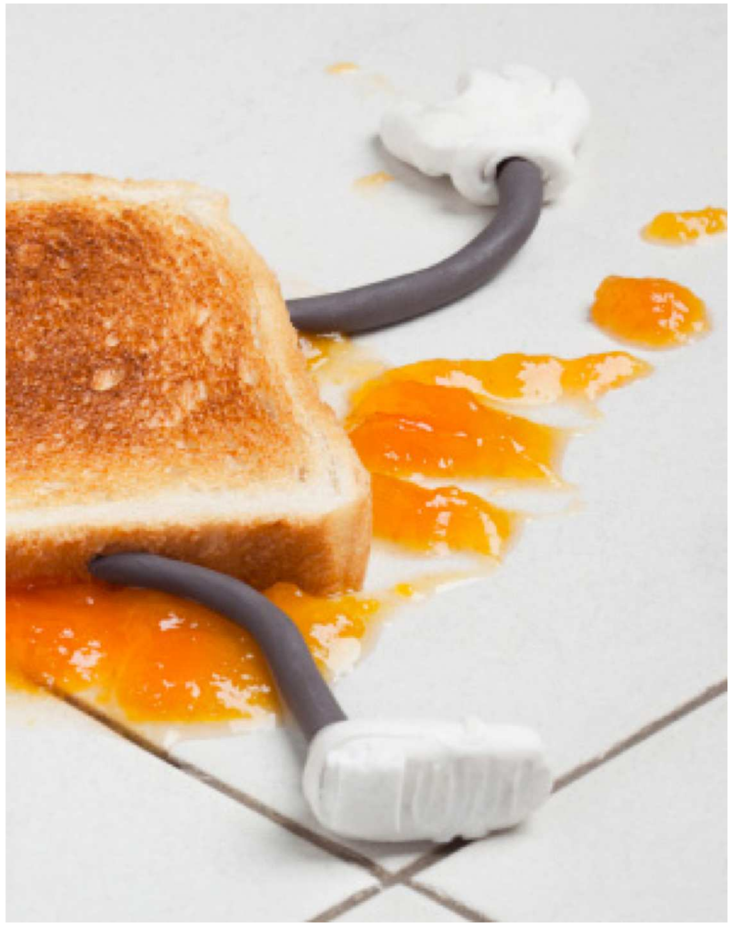
Semantics!

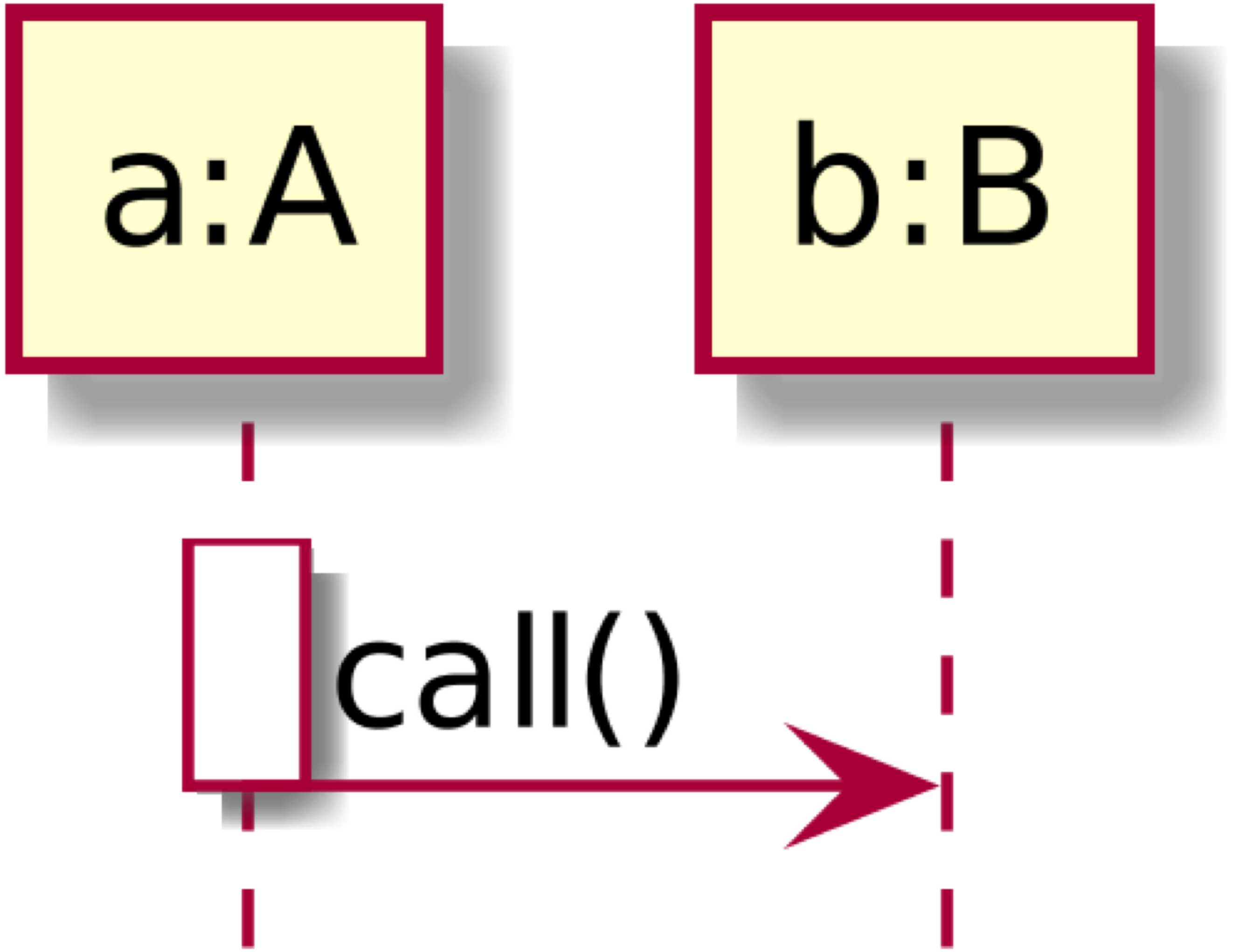
Webservice, not microservice



Reminder: Murphys's law

- "Anything that can go wrong will go wrong"
- Apply that to webservices architecture




a:A
b:B
call()

Reminder: Fallacies of distributed computing

- The network is reliable
- Latency is zero
- Bandwidth is infinite
- The network is secure
- Topology doesn't change
- There is one administrator
- Transport cost is zero
- The network is homogeneous

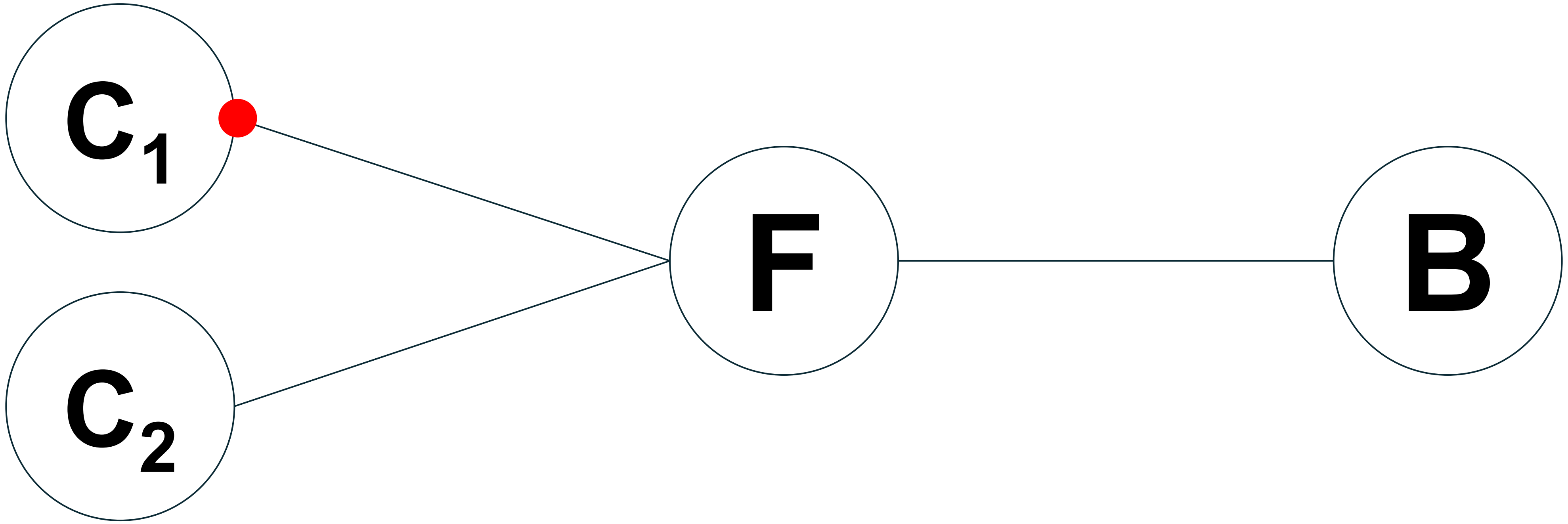
https://yourlogicalfallacyis.com/

Reminder: Fallacies of distributed computing

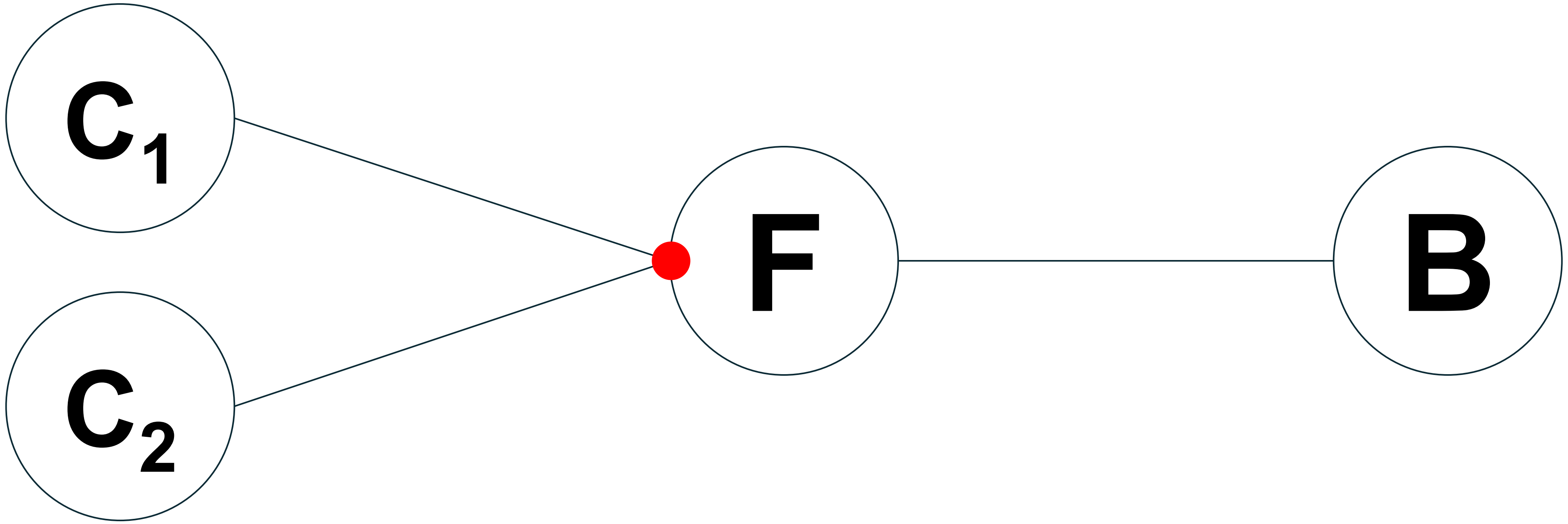
- The network is reliable
- Latency is zero
- Bandwidth is infinite
- The network is secure
- Topology doesn't change
- There is one administrator
- Transport cost is zero
- The network is homogeneous

https://yourlogicalfallacyis.com/

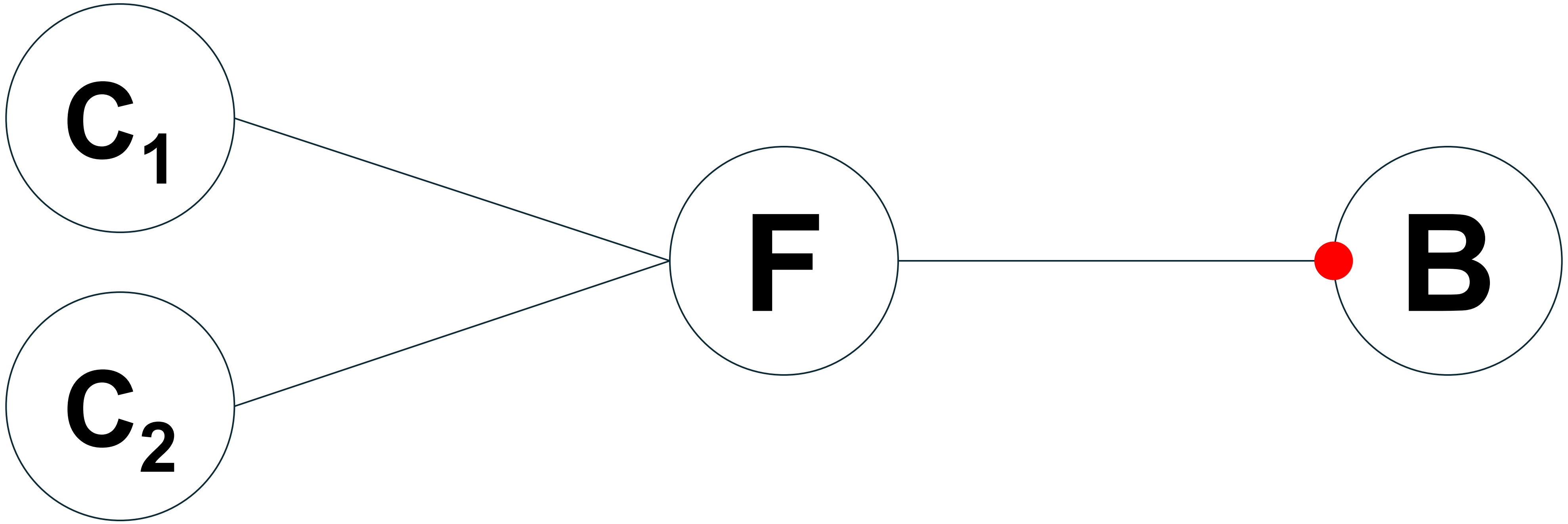
A sample webservice architecture



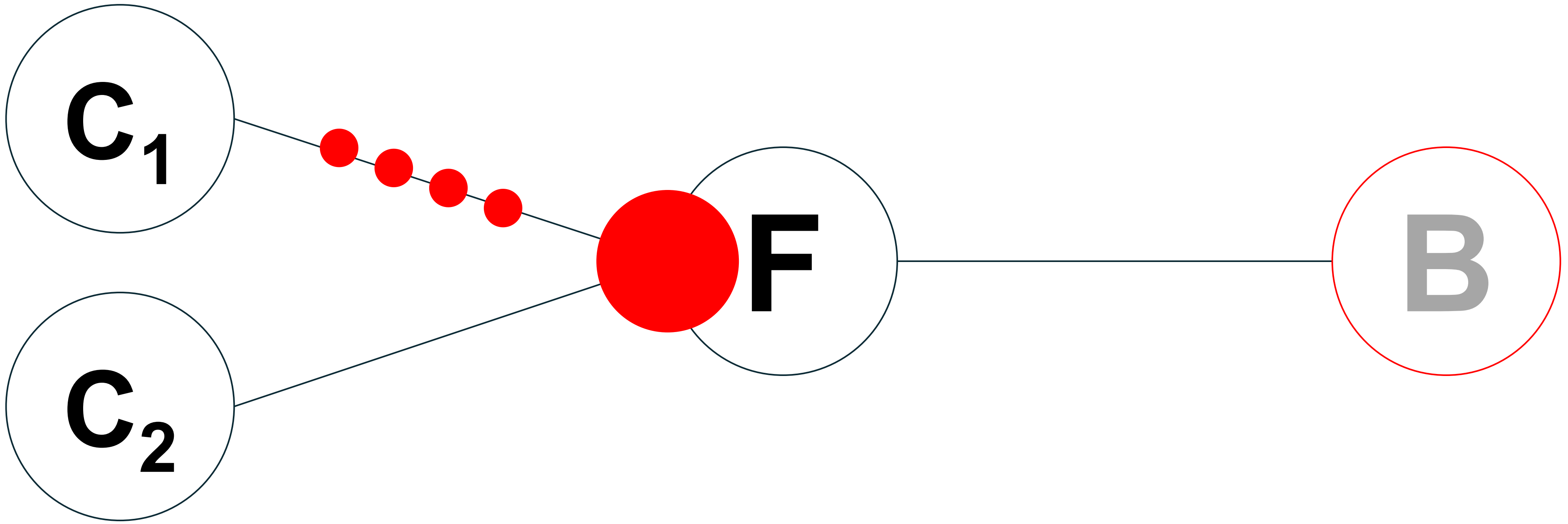
A sample webservice architecture



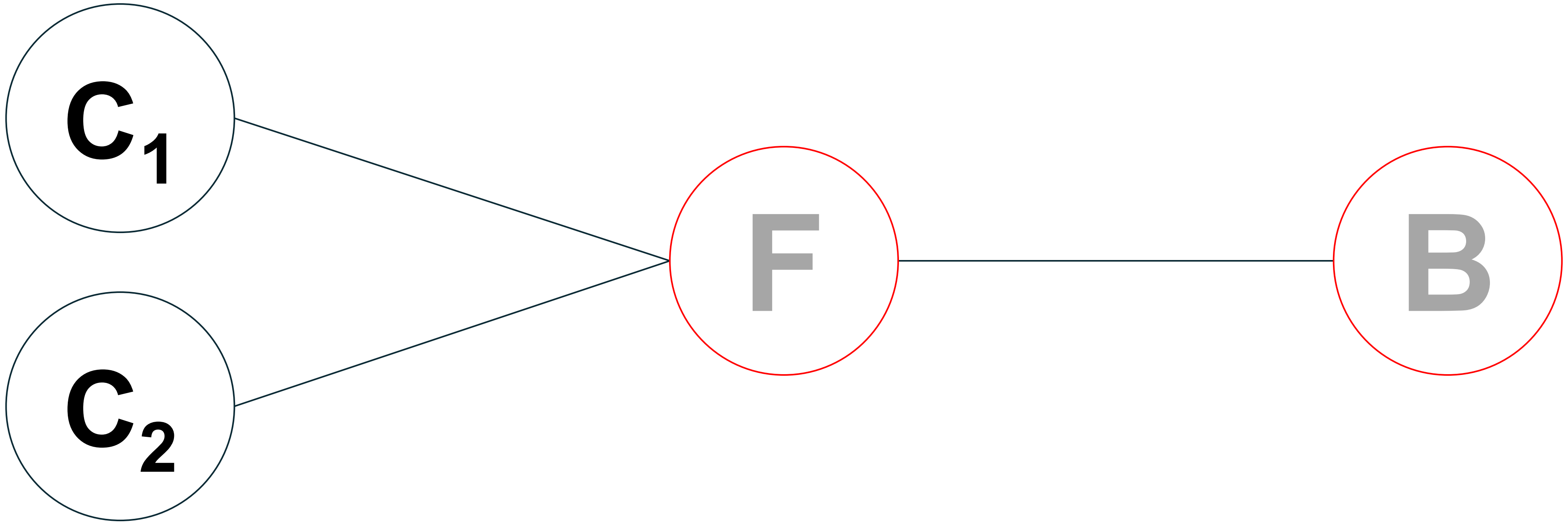
A sample webservice architecture



A sample webservice architecture



A sample webservice architecture

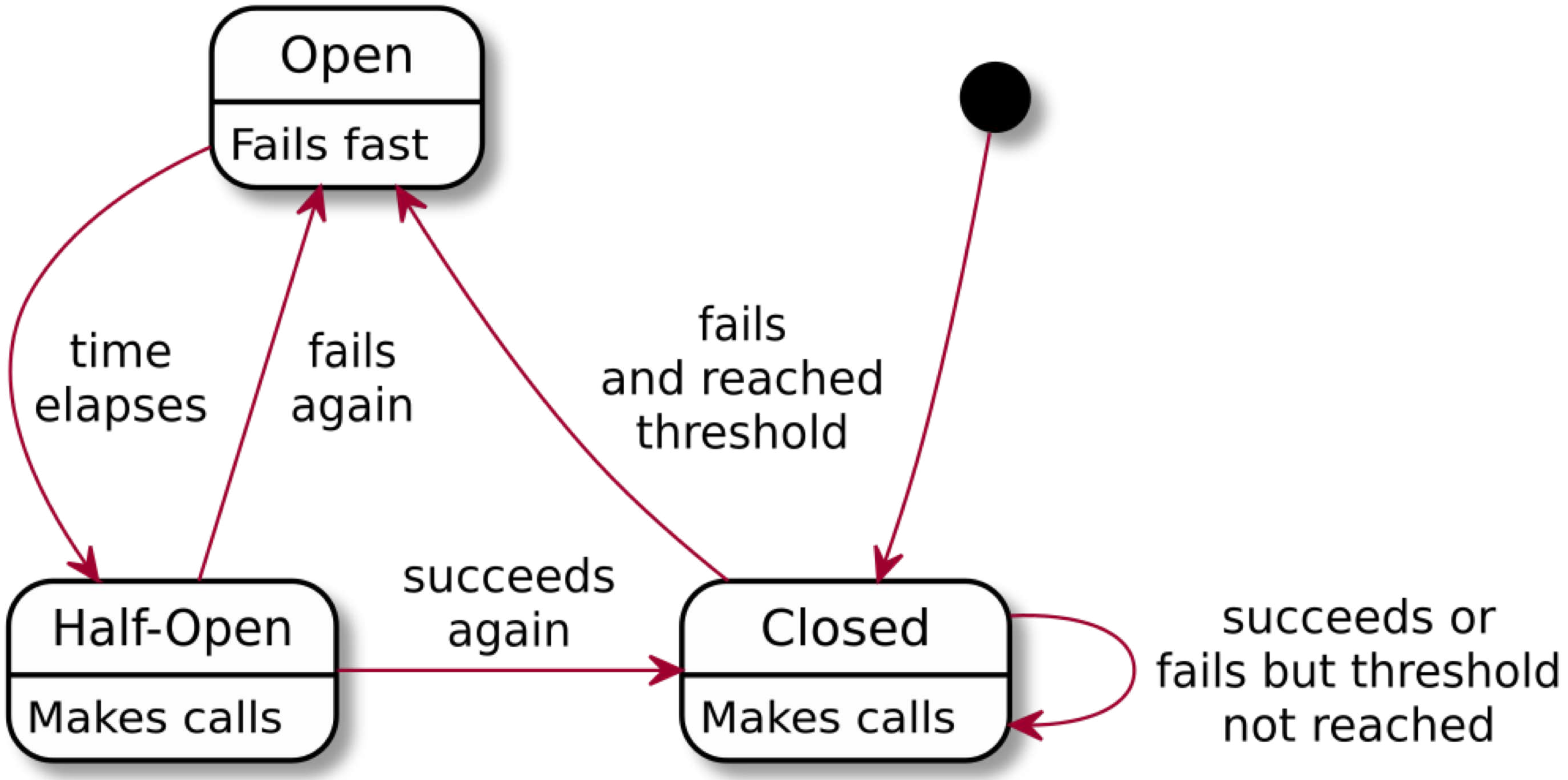


Enter the Circuit Breaker pattern

"A service client should invoke a remote service via a proxy that functions in a similar fashion to an electrical circuit breaker."

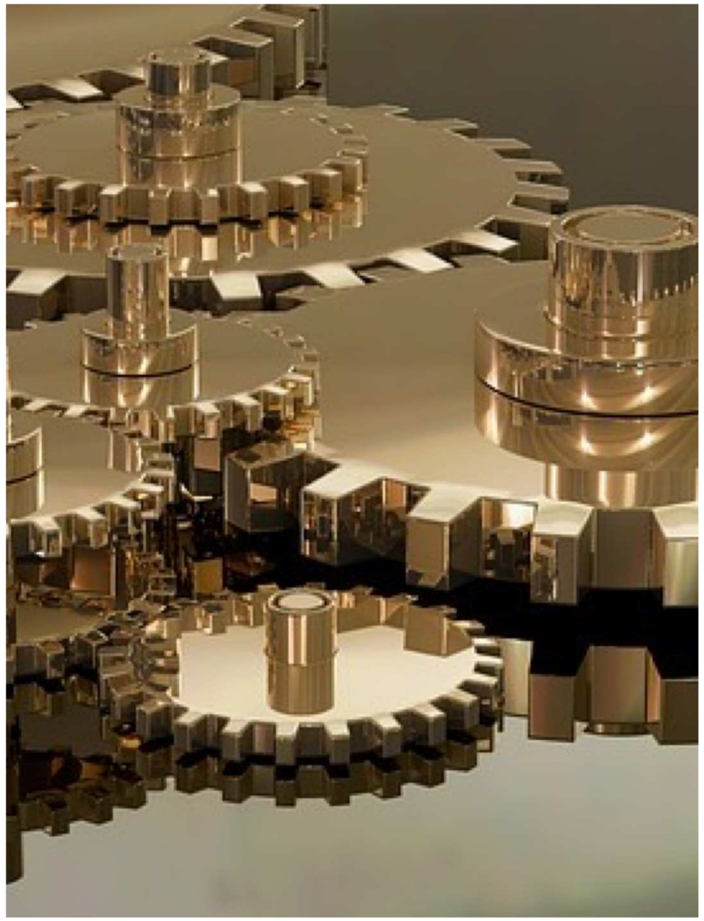
https://microservices.io/patterns/reliability/circuit-breaker.html

Circuit Breaker state machine



Configuration options

- Number of failed calls
- Elapsed time strategy:
  - Fixed
  - Doubling
  - Something else
- Number of successful calls



The most important configuration option

What to do in the case of timeout?

Use-case: e-commerce webshop

1. Recommendation webservice
   - “People also bought xyz”
2. Pricing webservice
3. Payment webservice
4. Logging webservice

Logging

- Fire-and-forget
- Asynchronous calls

Recommendation

- Synchronous req/response
- Optional
- Fallback options
  - Display no recommendations
  - Static recommendations set

Pricing

- Synchronous req/response
- Required
  - But better sell at a slightly outdated price!
- Fallback options
  - Accept outdated data from another source
  - In-memory cache

Payment

- Synchronous req/response
- Required
- Fallback options
  - Accept potentially bad payments 🤔

Available strategies

Strategy	Implementations	Fits
Black Box	• Proxies • Service meshes	Fail fast
White Box	Libraries • Hystrix • Resilience4J	Fallbacks relying on business logic

Service mesh

"A service mesh is a configurable infrastructure layer for a microservices application. It makes communication between service instances flexible, reliable, and fast. The mesh provides service discovery, load balancing, encryption, authentication and authorization, support for the circuit breaker pattern, and other capabilities."

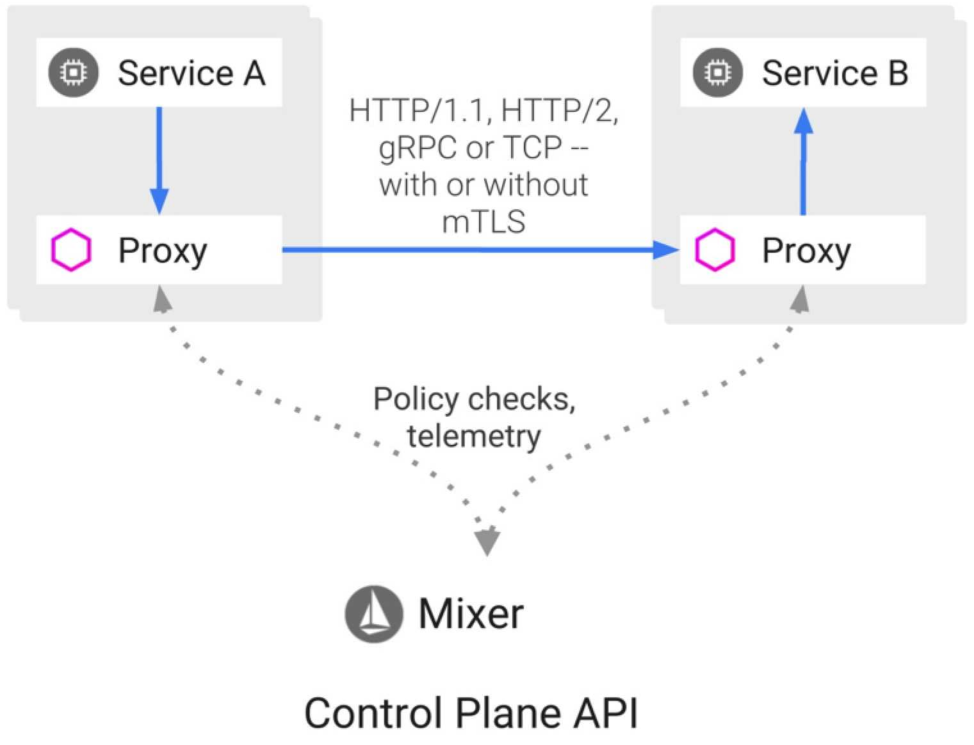
https://www.nginx.com/blog/what-is-a-service-mesh/

Istio

- Open Source service mesh
- Leverages Kubernetes
- Implements the sidecar pattern
- Uses the Envoy proxy under the hood

Sidecar pattern

Istio from a birds-eye view



https://istio.io/docs/concepts/what-is-istio/

Circuit-breaker configuration in Istio

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
 name: foo
spec:
 host: foo
 trafficPolicy:
   outlierDetection
     consecutiveErr
     interval: 10s
     baseEjectionTime: 1m
     maxEjectionPercent: 80
```

Number of consecutive errors that open the circuit breaker

Interval between two checks

Dur

Percentage of evicted instances

@nicolas_frankel

hazelcast

Cons of Istio

- No fallback

A talk in which you're the hero!

Go to slide 39

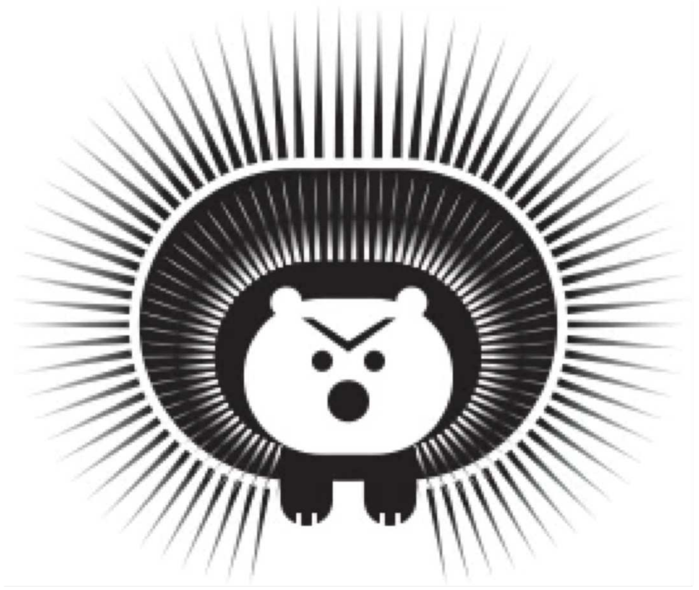
Go to slide 44

Hystrix

“Hystrix is a latency and fault tolerance library designed to isolate points of access to remote systems, services and 3rd party libraries, stop cascading failure and enable resilience in complex distributed systems where failure is inevitable.”

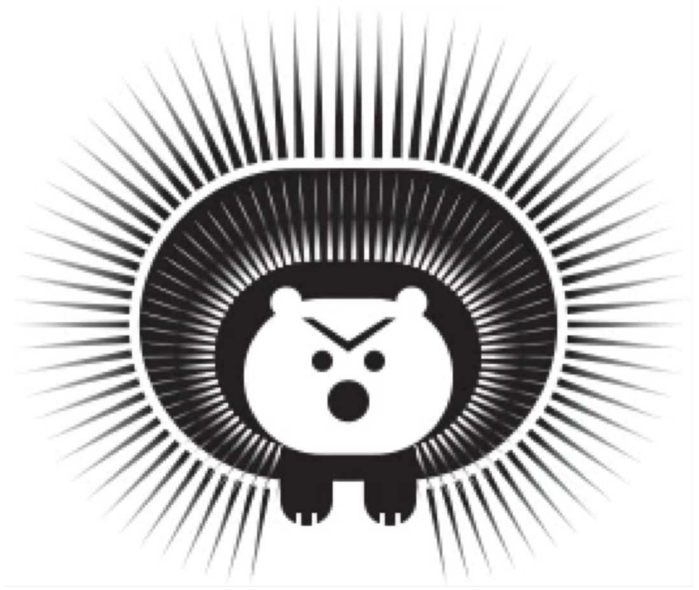
Hystrix

- Provided by Netflix
- Currently in maintenance mode ⚠️
- Superseded by Resilience4J
  - But not equivalent



Hystrix features

- Wraps calls into “commands”
- Run commands asynchronously from a thread pool
- Measure success/failures
- Circuit-breaker implementation
- Fallback logic



Cons of Hystrix

- A lot of configuration options
  - Hard to fine-tune
- No big picture

› Spring Cloud Netflix

- Easy Hystrix integration
- Also:
  - Service discovery: Eureka
  - Declarative REST client: Feign
  - Client-side LB: Ribbon
  - etc.

Resilience4J

"Resilience4j is a lightweight fault tolerance library inspired by Netflix Hystrix, but designed for Java 8 and functional programming."

Resilience4J's features

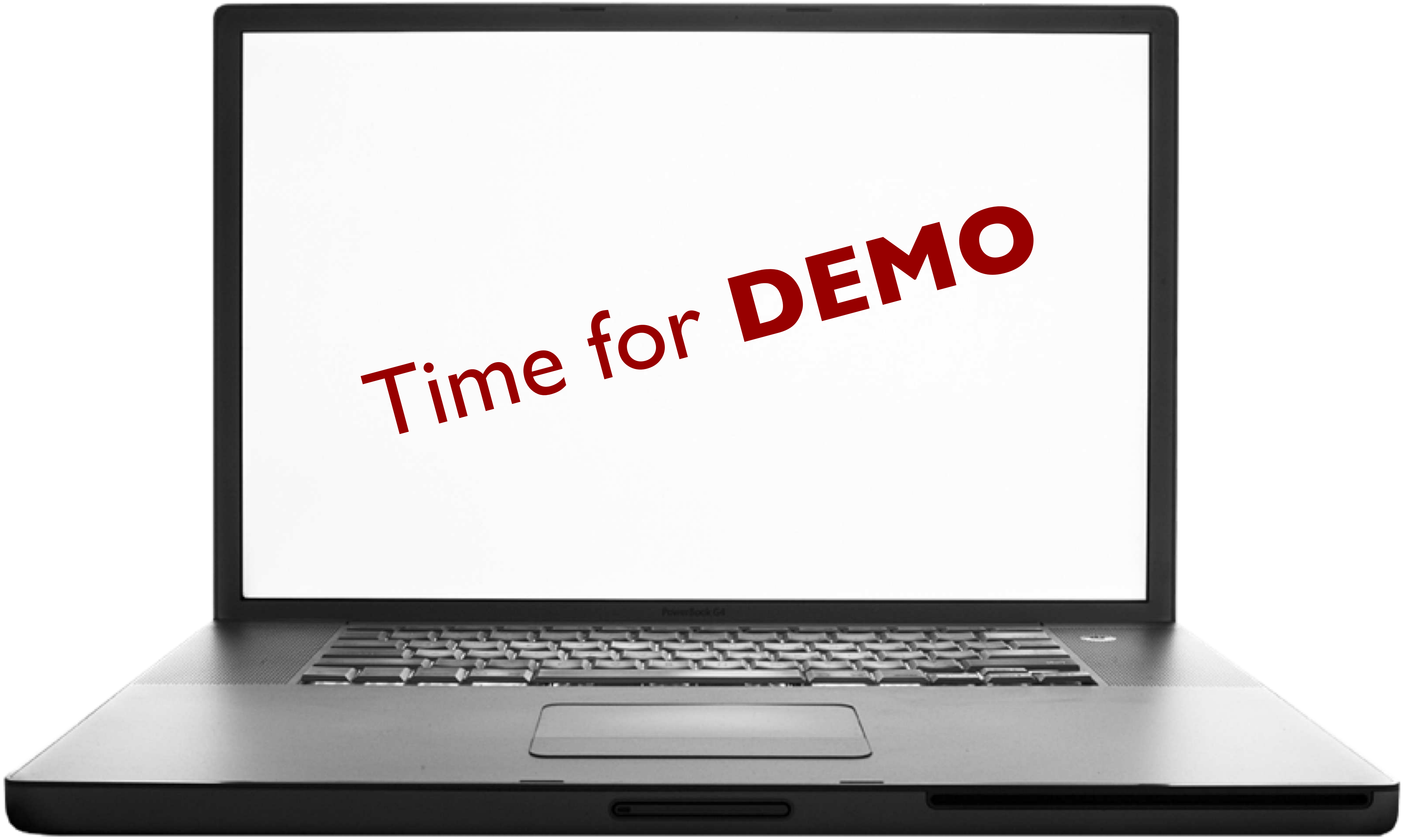
- Circuit Breaker
- Rate Limiter
- Retry
- Cache
- etc.

Resilience4J's design principles

- Each feature is designed as a function
- Uses Java 8 functional interfaces
  - *e.g.* Supplier
- Based on function composition
- Based on Vavr
  - Functional Programming in Java

Cons of Resilience4J

- Need to be very familiar with Functional Programming
- No big picture



Time for **DEMO**

Thanks

- https://blog.frankel.ch/
- @nicolas_frankel
- https://git.io/JenH9