CLOUD FOUNDRY
SUMMIT

RUNNING AT SCALE

APRIL 18-20, 2018 | BOSTON

CLOUD FOUNDRY SUMMIT

The Java Ecosystem Collision: What is the future of Cloud Native?

Erin Schnabel

Senior Technical Staff Member, IBM

What is a Cloud Native Application?

Must use **Cloud-only services like Object Storage**

Must be built using **agile, devops approaches** with **developers on pager duty**, must expect failures in prod, …

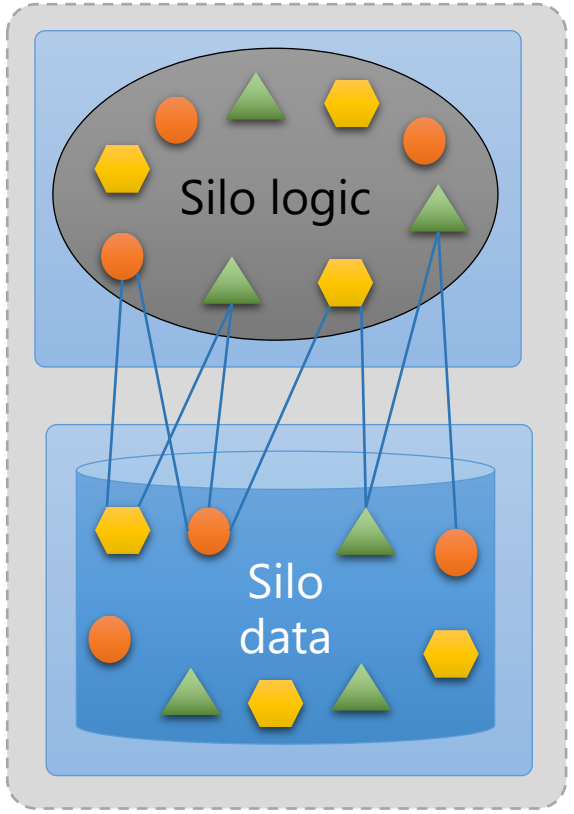
Cloud Native: A definition

- Microservice oriented
  - Loosely coupled
  - Declared external dependencies

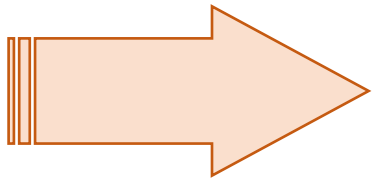
- Container packaged
  - Resource isolation
  - Simplified operations

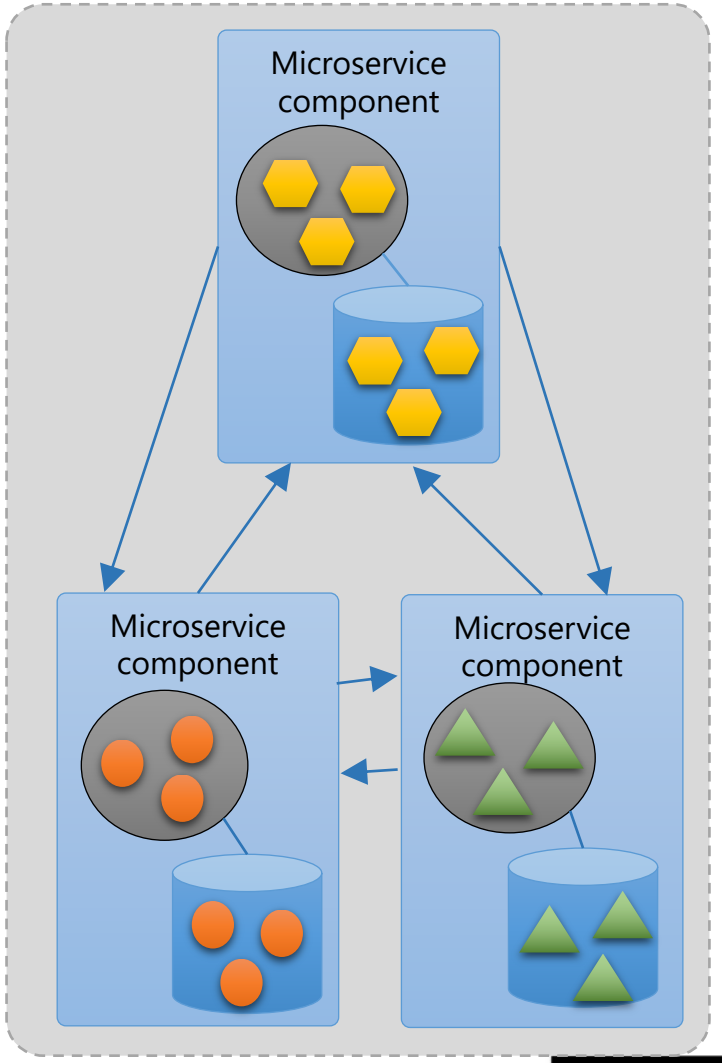
- Dynamically managed / orchestrated
  - Disposable instances → Elastic scaling

Isolate all the things!



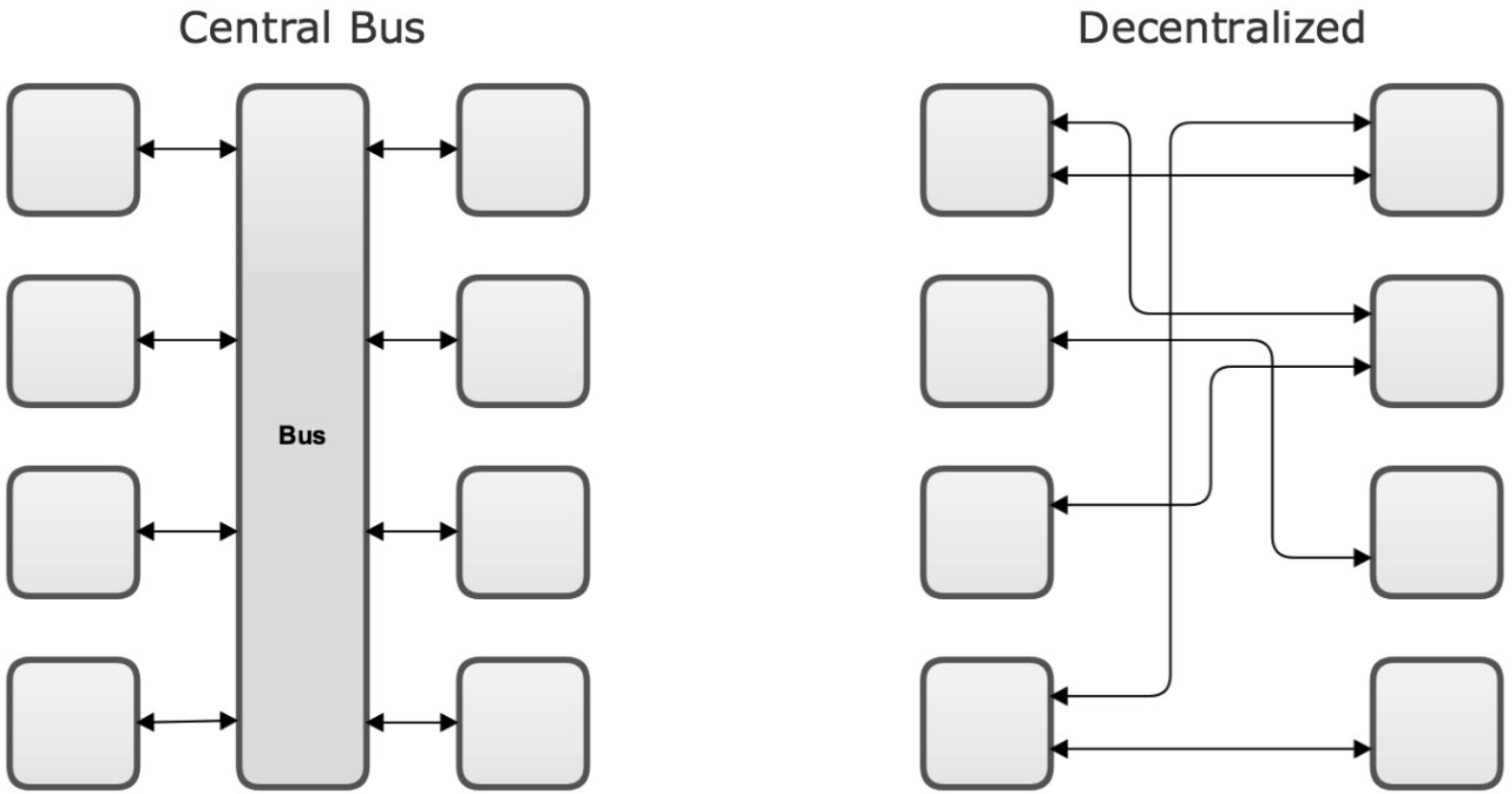
Monolithic application





Microservices application

Smart Endpoints, Dumb pipes



https://medium.com/@nathankpeck/microservice-principles-smart-endpoints-and-dumb-pipes-5691d410700f

"Twitter microservices map looks just like the Netflix one. "We called this the 'Death Star' diagram"

– Adrian Cockcroft via twitter

https://twitter.com/adrianco/status/441883572618

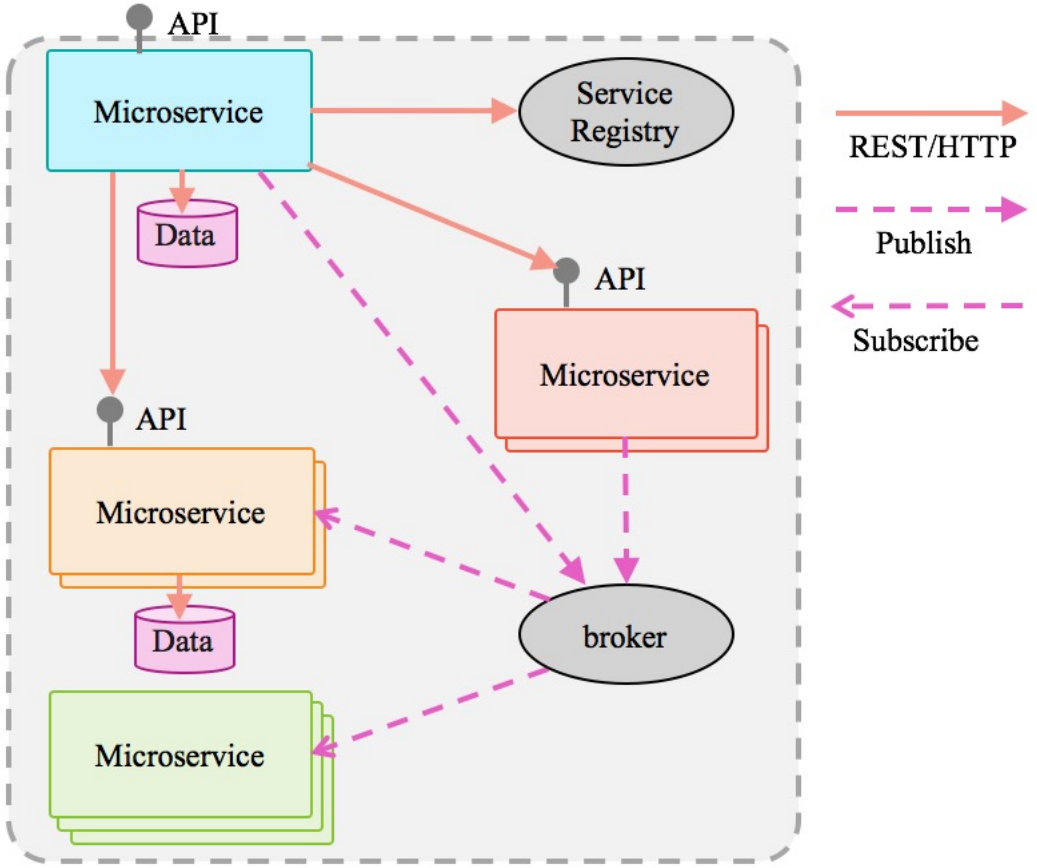
Infrastructure Automation / Orchestration

- Provisioning/Deployment
  - Zero-downtime upgrades
  - Scaling
  - Health Management

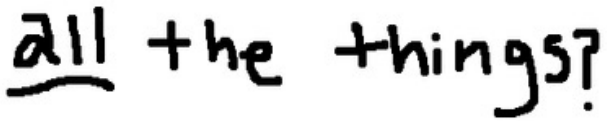
- Real-time monitoring of transient processing
  - Logging: ELK (LogStash), EFK (Fluentd)
  - Metrics: Graphana
  - Distributed tracing: OpenTracing

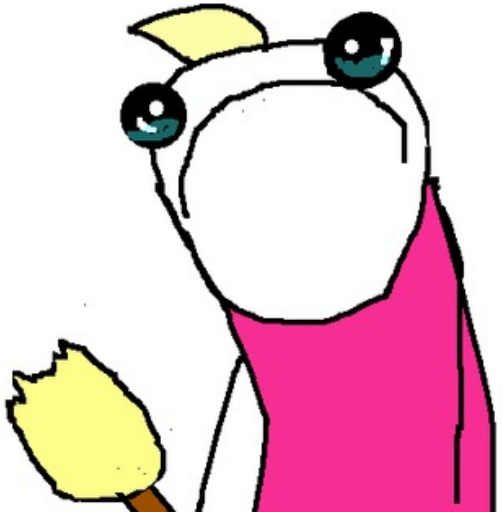
Microservices must be…

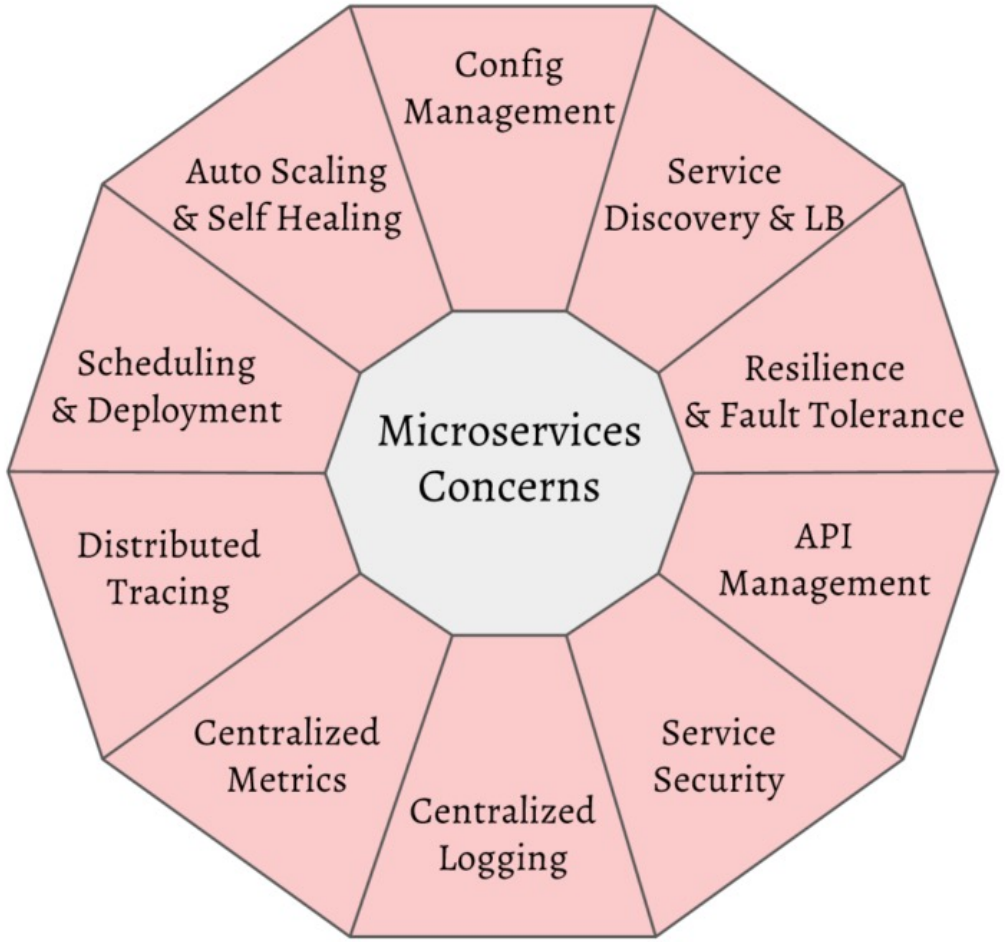
- Stateless (ephemeral)
- Robust
  - **Expect rubbish**
- Fault tolerant:
  - No cascading failures
  - **Fail fast, gracefully**
  - Circuit breakers / bulkheads / timeouts
  - **Fallback**: retry vs. cached data



This is not easy

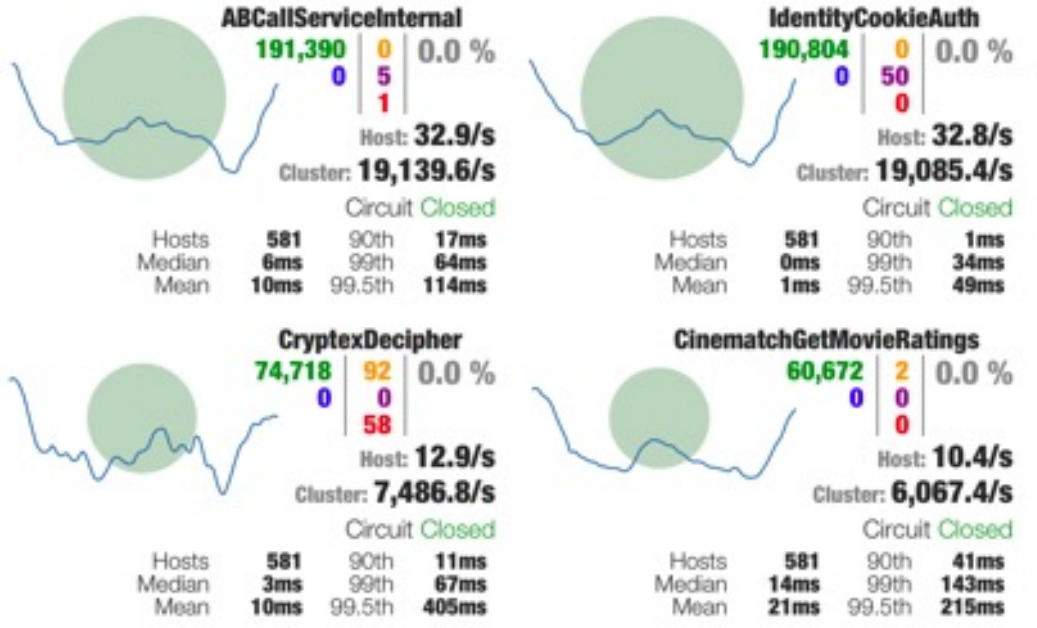






Netflix: (Java) Libraries for the hard stuff

- Eureka: Service discovery
- Zuul: API Gateway, Health-aware routing, Pre/post/route filters
- Ribbon: Client side load balancing
- Hystrix & Turbine:
  - Fault tolerance:
    - circuit breaker, bulkhead, fallbacks, batching
  - Real-time client-side telemetry
- Metrics: Atlas & Spectator
- Archaius: Configuration



Spring Cloud Netflix

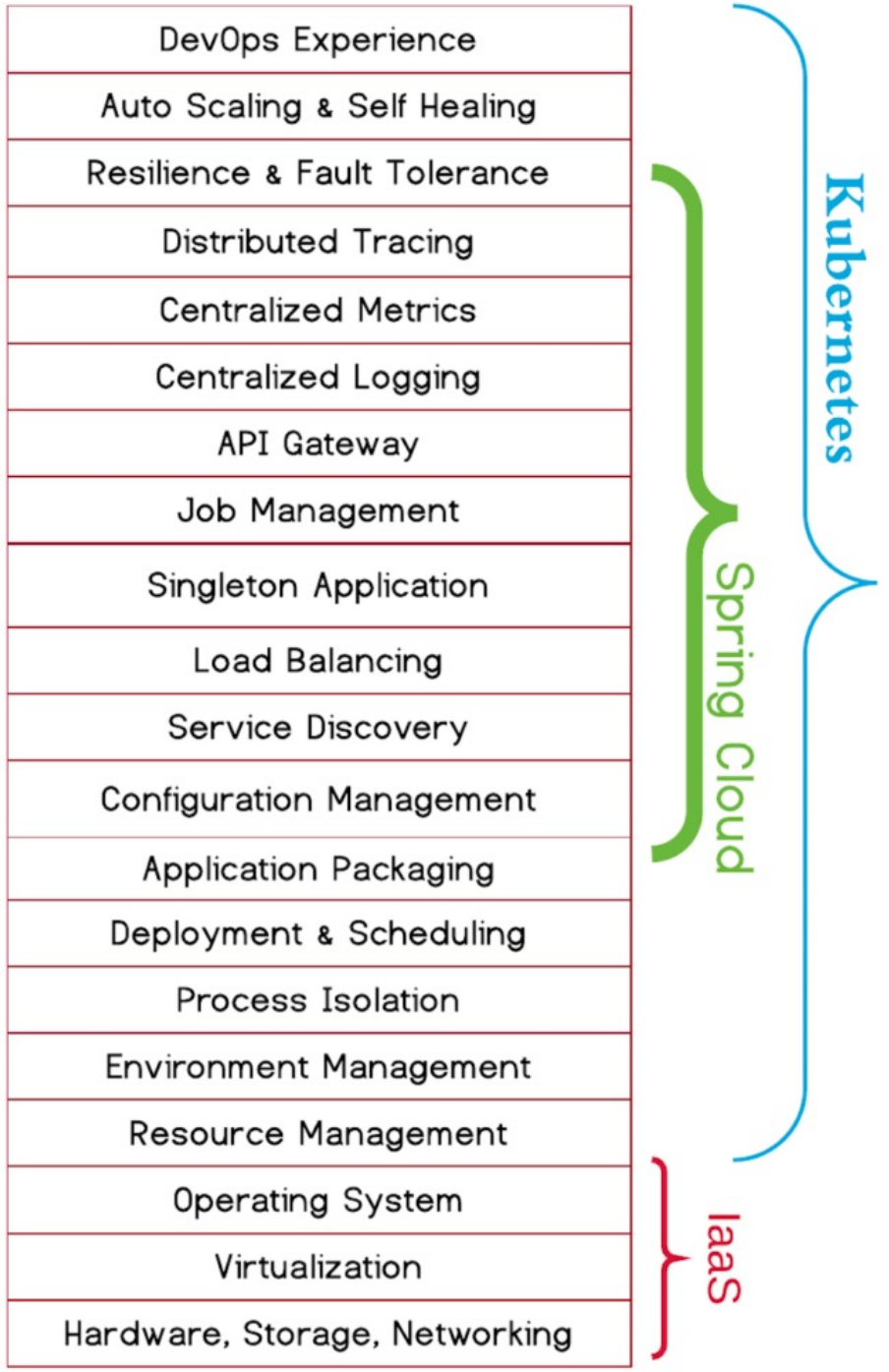
Microservices Concern	Spring Cloud & Netflix OSS
Configuration Management	Config Server, Consul, Netflix Archaius
Service Discovery	Netflix Eureka, Hashicorp Consul
Load Balancing	Netflix Ribbon
API Gateway	Netflix Zuul
Service Security	Spring Cloud Security
Centralized Logging	ELK Stack (LogStash)
Centralized Metrics	Netflix Spectator & Atlas
Distributed Tracing	Spring Cloud Sleuth, Zipkin
Resilience & Fault Tolerance	Netflix Hystrix, Turbine & Ribbon
Auto Scaling & Self Healing	-
Packaging, Deployment & Scheduling	Spring Boot
Job Management	Spring Batch
Singleton Application	Spring Cloud Cluster

https://developers.redhat.com/blog/2016/12/09/spring-cloud-for-microservices-compared-to-kubernetes/

Spring Cloud Kubernetes

- Kubernetes ops/automation
  - Group and scale pods of containers
  - Bring your own container
  - Stateful persistence layer
- Spring programming model niceness
  - *@DiscoveryClient*
  - *@RibbonClient*
  - Property sources for k8s ConfigMap

https://github.com/spring-cloud-incubator/spring-cloud-kubernetes



https://developers.redhat.com/blog/2016/12/09/spring-cloud-for-microservices-compared-to-kubernetes/

This is still too hard ...

- Too much infrastructure awareness in the app!
  - Netflix starters
    - *@EnableEurekaClient*
    - *@EnableAtlas*
  - Client-side load balancing
    - *@EnableRibbonClient*
    - *@LoadBalanced*
    - Retry policies defined per application
    - Zone awareness/avoidance per application

Istio & Envoy: Sidecar for the hard stuff

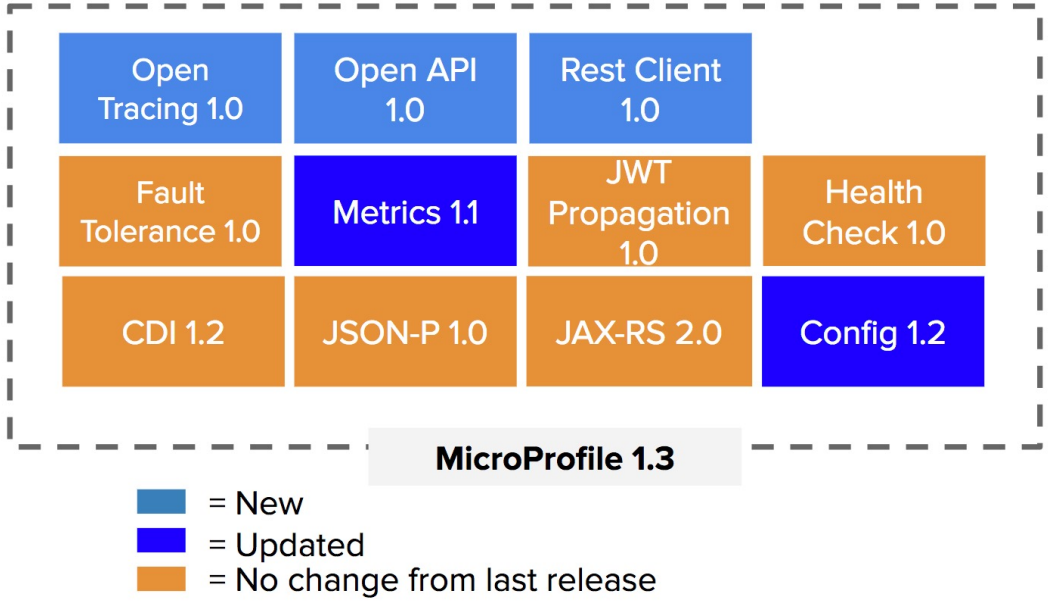
- Language agnostic – out of process
- Content aware
- Common set of capabilities
  - Load balancing
    - Circuit breakers, Automatic retries, Rate Limiting
    - Request shadowing
  - Wire-level tracing
    - OpenTracing (distributed tracing)
  - Secure TLS transport
  - Policy enforcement

What about Fault tolerance?

- Circuit breakers
- Bulkheads
- Retries
- Fallbacks

Do I have to use Spring?

- OpenLiberty, Wildfly, TomEE, Payara
- Cloud Native with fewer dependencies



Serverless: App framework-less?

- AWS Lambda example: Jackson.

- Amazon Java SDK core:
  - Jackson, Commons Logging, Apache httpclient, joda-time

REST/HTTP vs. gRPC

- REST
  - OpenAPI / Swagger
  - Verbose
  - Debuggable/human readable

- gRPC is compact, binary format
  - Proto3 has canonical mapping to JSON
  - gRPC API can also specify the v2 REST-JSON API
    - Care with field renaming: changes in gRPC might break JSON API
    - Envoy: https://blog.envoyproxy.io/evolving-a-protocol-buffer-canonical-api-e1b2c2ca0dec

REST vs. Reactive

- REST/HTTP is inherently synchronous
  - Async libraries can mitigate
  - Does every message need an ack?

- Observer pattern:
  - One event publishes
  - 1..n subscribers/observers to act on events

- Events --> Reactive Streams

Synchronous calls considered harmful

Any time you have a number of synchronous calls between services you will encounter the multiplicative effect of downtime. Simply, this is when the downtime of your system becomes the product of the downtimes of the individual components. You face a choice, making your calls asynchronous or managing the downtime. At www.guardian.co.uk they have implemented a simple rule on the new platform - one synchronous call per user request while at Netflix, their platform API redesign has built asynchronicity into the API fabric.

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

The End

- Today:

2:35pm	Istio Platform vs Spring and MicroProfile Frameworks - Ozzy Osborne, IBM UK
3:15pm	Building Responsive Systems with Serverless, Event-driven Java - Richard Seroter, Pivotal & Asir Vedamuthu Selvasingh, Microsoft
3:55pm	Designing, Implementing, and Using Reactive APIs - Ben Hale & Paul Harris, Pivotal

- Tomorrow:

3:45pm	How Cloud Foundry Compares with Kubernetes for Deployment of Cloud Native Java Applications - Surya V Duggirala, IBM

CLOUD FOUNDRY SUMMIT