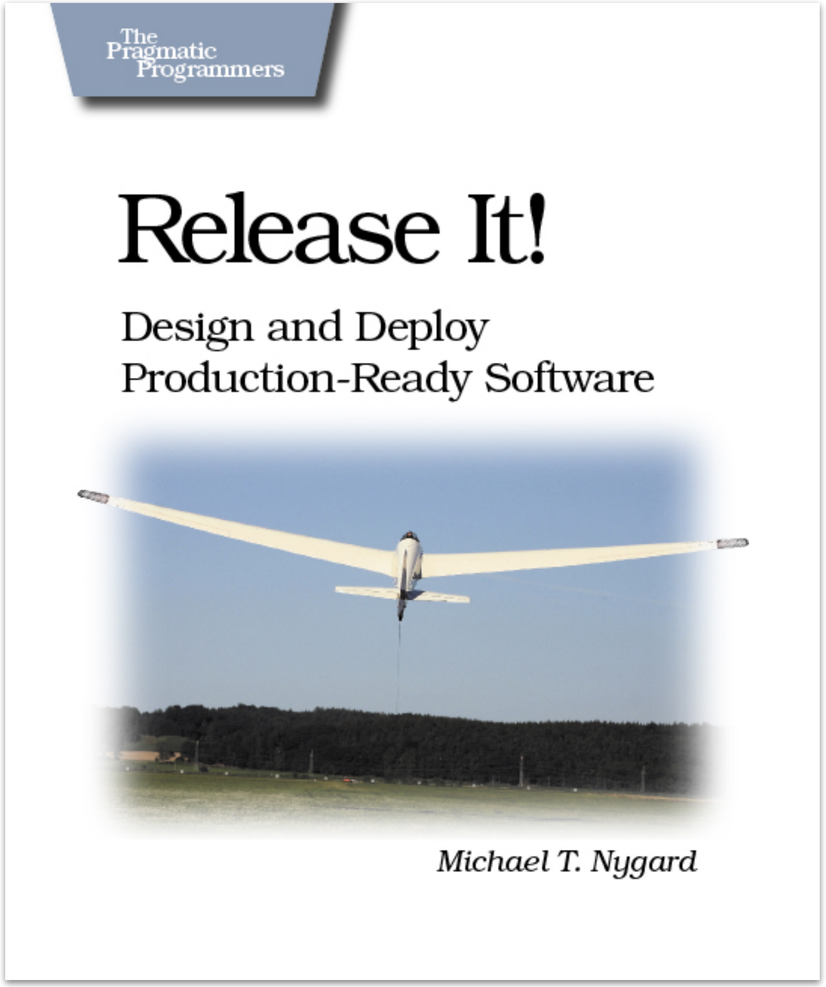
MicroServices Patterns

- Michael Nygard, 2007

- Describes fault tolerance patterns that became the inspiration for many of the implementations now in use with MicroServices

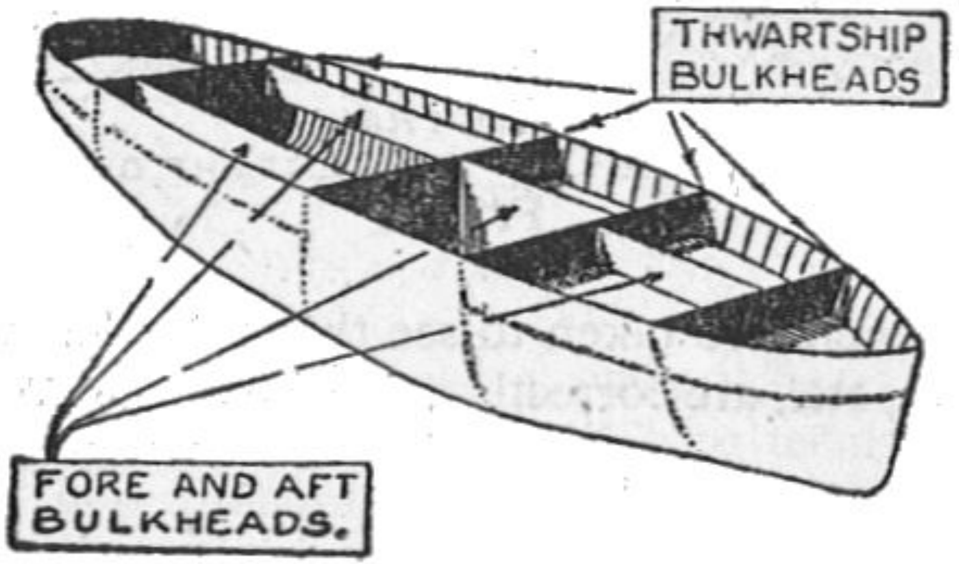


Circuit Breakers

- Objective is to make a distributed system of microservices less vulnerable to failures
- Prevents failures in a microservice from impacting calling applications, and cascading upstream to calling microservices
- Limit scope (or “blast radius”) of a failure, degrade gracefully is possible, and automate recovery when failing service comes back online
- Implementation: *Netflix Hystrix*

Bulkheads

Design comes from the ship construction:



Applications in software:

Compartmentalizing thread pools prevents a single misbehaving service from taking down a client application instance

Service Registries

- A microservice architecture consists of many collaborating service instances that much know each others' address
- A cloud environment implies application instances that come and go, that are dynamically scaled
- Service registries provide dynamic application instance lookup capabilities
- Pattern prevalent in distributed systems: Service Locators, Membership Coordinators
- Examples: HashiCorp Consul, Apache ZooKeeper, *Netflix Eureka*

Load Balancing

- Load balancing traditionally available as an independent appliance (F5) or software component (HAProxy)
- Move load balancing capability directly into the consumer microservice
- *Netflix Ribbon*: a client-side load-balancing library

Distributed Tracing

- In a monolith, the majority of the call graph is contained in a single process. A stack trace can easily help us see the context of handling an http request. Not so with microservices.

- Zipkin is an implementation of a distributed tracing system

- Spring Cloud Sleuth (zipkin compatible) can easily instrument a Spring Boot application to trace calls across microservice process boundaries.

Configuration as a Service

- Idea of separating application configuration from codebase, serving configuration over REST endpoints

- Centralizes configuration for multiples applications and multiple environments (staging, QA, prod, etc..) in a single repository

- Benefits include ability to alter and reload configuration without restarting application, altering logging levels, turning on/off feature toggles

- Spring Cloud Config Server is an open source implementation, backed by a git repository naturally supports retaining configuration history and audit trail, supports encryption of sensitive configuration properties, and designed to integrate with Spring Boot applications.

Proxies and Edge Services

- Request interception modeled in a microservice architecture by routing requests to back-end services via a proxy or edge service

- Has many uses: proxies can be used as part of a strangler pattern strategy, can be use for rate limiting, authentication, request normalization and enrichment, to avoid duplicating common request processing logic into multiple back-end services..

- Examples: *Netflix Zuul*, Pivotal Cloud Foundry's *Route Services*