AARHUS UNIVERSITET

Microservices and DevOps

DevOps and Container Technology

Microservices – the short story…

Henrik Bærbak Christensen

AARHUS UNIVERSITET

Microservices

- The next step? A successful step?
- Wikipedia (2018)

> A microservice is a software development technique—a variant of the service-oriented architecture (SOA) architectural style that structures an application as a collection of loosely coupled services. In a microservices architecture, services are fine-grained and the protocols are lightweight. The benefit of decomposing an application into different smaller services is that it improves modularity. This makes the application easier to understand, develop, test, and become more resilient to architecture erosion.[1] It parallelizes development by enabling small autonomous teams to develop, deploy and scale their respective services independently.[2] It also allows the architecture of an individual service to emerge through continuous refactoring.[3] Microservices-based architectures enable continuous delivery and deployment.[1][4]

- Keywords
  - *Loosely coupled, fine-grained, lightweight protocols, autonomous teams, independent deployment and scaling, continuous delivery.*

Definition

- Newman's *groundbreaking* and *highly precise* definition.

Microservices are small,
autonomous services that work together.

Definition: **Object-orientation (Responsibility)**
An object-oriented program is structured as a community of interacting agents called objects. Each object has a role to play. Each object provides a service or performs an action that is used by other members of the community.

Budd (2002)

AARHUS UNIVERSITET

Defining Characteristics

- **Small, focused on doing one thing well**
  - Service boundaries are business boundaries
  - Explicit boundaries (out-of-process communication)
  - *Small enough and no smaller*

- **Autonomous**
  - Separate entity
  - Communication is network calls (avoid tight coupling) (hm...)
  - Expose API (technology-agnostic)
  - *Decoupling: can I change this service without changing any other?*

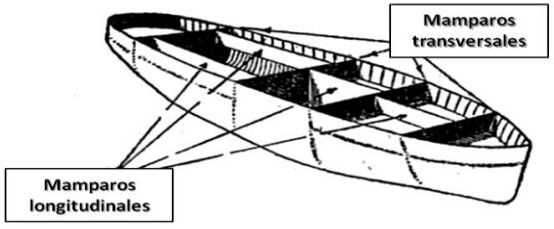
Key Benefits

- Technological Heterogenity
  - Each service may use its own technology stack
    - *Pick the right tool for each job*
  - May choose data storage techology independently
  - Quick technology adaption
    - Lower risk by selecting new technology for given service
  - *Counterpoint*
    - *Overhead in maintaining many technologies*
    - Company ‘Technology Decisions’ may restrict that
      - NetFlix and Twitter: Only JVM based systems

AARHUS UNIVERSITET

Key Benefits

- Resilience
  - Nygard (2017) pattern: **Bulkhead**
  - *Bulkhead:* Partitioning a system so failures in one part does not lead to system failure



  - Handle failure of services and degrade functionality accordingly
  - *Counterpoint*
    - Highly distributed systems have a *lot* of failure modes that needs to be addressed
    - *Will return to Nygard and techniques in next course!*

Key Benefits

- Scaling
  - Just scale the microservice that needs scaling
    - Opposite monolith system: all things scale together
  - Utilize on-demand provisioning of VMs to scale automatically
- Ease of Deployment
  - A one-line bug fix in one service only means one service to redeploy
    - And rollback is also much easier
    - Opposite monolith system: full redeployment of monolith
      - Fear of breaking stuff => changes accumulate

Key Benefits

- Organization Alignment
  - Small teams on small service – align organization and architecture
    - Smaller teams on smaller code bases are more efficient
- Composability
  - Functionality consumed in different ways for different purposes
- Optimizing for Replacability
  - Out of date services are easier to replace because its small size
    - Opposite: That monster COBOL system everybody is afraid of...

AARHUS UNIVERSITET

Discussion

- Actually, I find Fowler has a more precise defintion ☺
- Stay tuned – we *will* return in the second course !!
- And the failure mode handling…