Antipattern: Infrastructure Ignorance

Author(s): Andreas, Theo, Christian

Ignoring the target environment, e.g. hardware performance, network topology and quality, power limitations during (initial) design or/and development. This lead to systems which do not meet the requirements and that will be hard or even impossible to fix later.

Description

Developers tend to focus too much on the business logic. Infrastructure is an important part of the whole application, even more today than before.

- Ignoring the network latency and physical layout could result in bad performance.
- Security considerations should be respected from the beginning. It is often not possible reengineer that later.
- Legal considerations should be taken from start, for example might it prove difficult to remove personal data everytime from a Kafka stream, to follow local law (GDPR).
- Just blindly using cloud technology might result in security and/or performance problems. For example, encryption of all communication will improve the security but negatively impact the performance and latency.

What are some examples?

- Developing a kiosk system only on the developer laptop but not on the real hardware.
- Using blindly all cloud infrastructure like istio.io
- Trying to use highly distributed services on a slow intranet.

Why does this happen?

- Focussing too much on the developing code, ignoring the target environment
- Blindly trusting in technology standards like cloud technology
- Start with something, get off fast, in the hope to fix theses other problems later
- Addressing the target environment too late

How can we avoid getting into the situation in the first place?

- Define (the important) quality goals and concrete quality scenarios
- Test/Deploy early and regularly on target hardware (walking skeleton)
- Check every component of the system if it can follow the legal restrictions, for example to delete personal data

- Plan the deployment from the beginning, for example define/test the communication paths
- Ask for help from external experts
- Tactics and actions to reach quality goals
    - Load, performance and stress testing
    - Threat modeling
    - Chaos Engineering

What are suggestions to get out of the situation if we ended up in it?

- Review, analyse current state
- See above (Define quality goals, tactics and actions)
- (Reconsidering the modularization of the overall system under aspects of the infrastructure, then possibly refactoring the components. For example merging microservices that are too dependent, so that the execution of one function is always involving the other service.)