**** microservices

Radan Skorić

When is it smart to break you application into microservices?

Drumroll

It depends!

On your project …

Thank you!

Questions?

Pros/Cons

They are closely related!

Complexity

- Smaller code size per service makes it easier to understand.
- Smaller more focused test suite.
- Agility of the small application.

- Harder to develop large features.
- High interdependency gets hidden in complex communication.
- Must reduce the chatter.
- Harder to create integration tests.

Deployment

- Can deploy each service independently.
- Can scale each service independently.
- Can better handle total failure of one service (if correctly designed).

- Need to setup deployment of each service independently.
- Harder to maintain full system uptime (0.99^6 = 0.94).
- Need to build handling for communication failures.

Flexibility

- Polyglot development.
- Can easily rewrite entire service if needed.

- Harder to move developers between services.

Culture

- Team has higher sense of ownership.
- Logical business boundaries are more clearly reflected in code.
- Easier for stakeholder to build a relationship with the team.

- Teams might put functionality in their service just because it is easiest to do.
- Very hard to rework the boundaries.
- Conway’s law.

Things look better when you look at one service

Things look worse when you look at the entire system

Mostly! You know what I mean, don’t troll me. :)

Getting boundaries right is very very important

- You will probably underestimate the surface area.
- There is a hefty price attached to fixing it.
- Unless you have trivial boundaries and an experienced team, start with a monolith and chip away the services when it becomes painful.
- Be ready to refactor across services.

Thank you

this time for real