Microservices by try&error

Story-telling Guide

Jan Porbincak

Ján Porhinčák

Python Dev
Dark Side: Software architecture apprentice

Programator senior at **Seznam.cz**

Twitter @kenaco666

Agenda

- From Monoliths to Microservices
- Logging
- Monitoring
- DevOps, SRE & Conways Law
- Distributed Monolith
- When system fails
- One nice example
- Conclusion

From Monoliths to Microservices
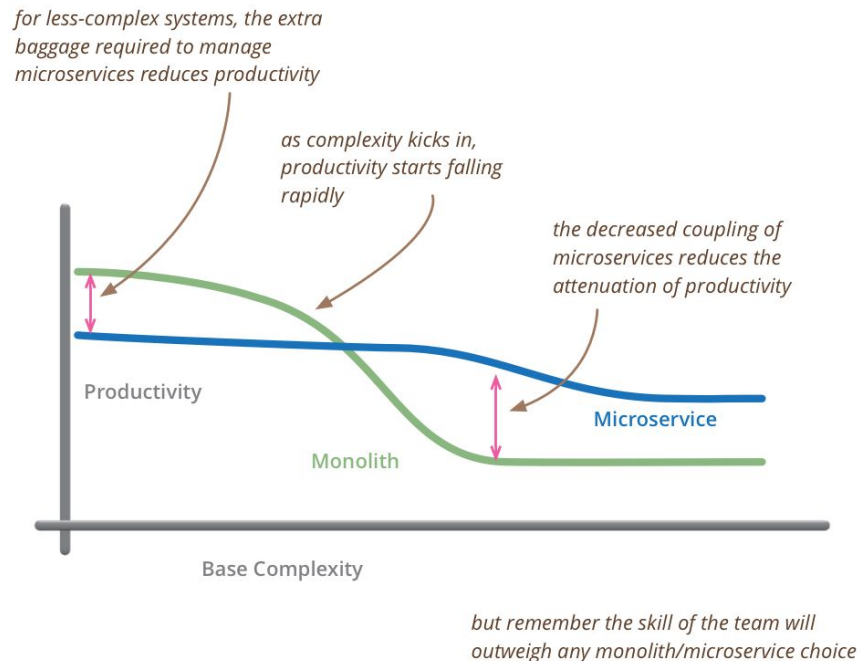
From Monoliths to Microservices

Microservice benefits:

- Technology heterogeneity
- Resilience
- Scaling
- Ease of deployment
- Organisational alignment
- Composability
- Optimizing for replaceability
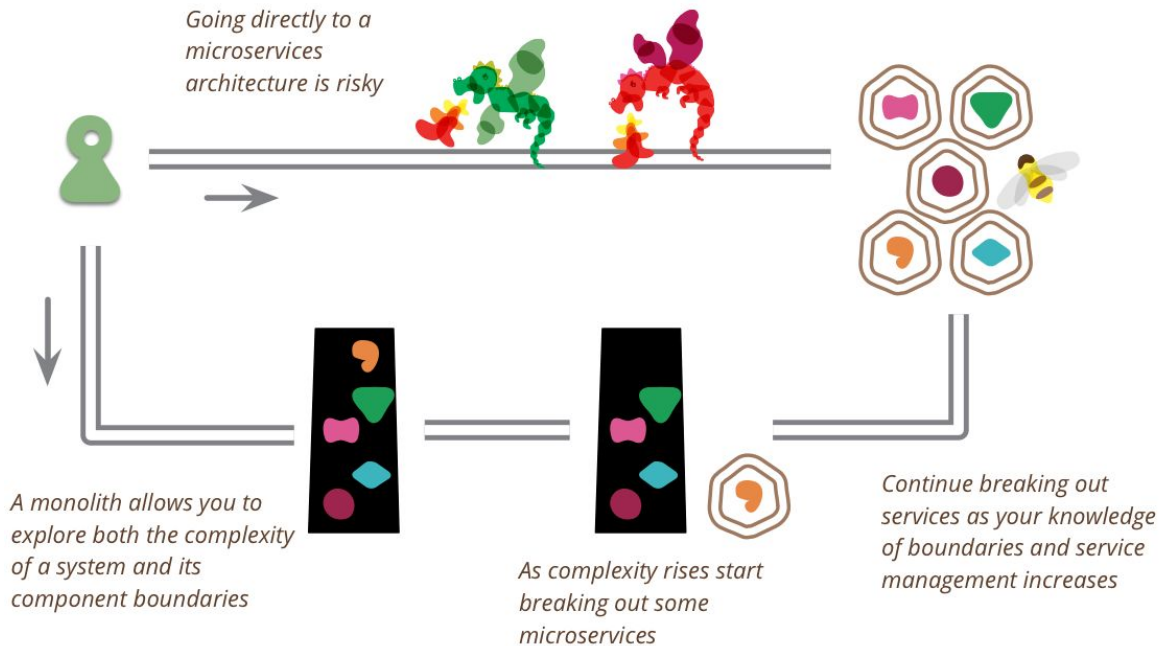
From Monoliths to Microservices

Monolith benefits:

- Technology homogeneity
- Ease of deployment (one artefact)
- No **Microservice Premium** needed
- Optimizing for faster development cycle

for less-complex systems, the extra baggage required to manage microservices reduces productivity

as complexity kicks in, productivity starts falling rapidly

the decreased coupling of microservices reduces the attenuation of productivity

Productivity

Microservice

Monolith

Base Complexity

but remember the skill of the team will outweigh any monolith/microservice choice

https://martinfowler.com/bliki/MicroservicePremium.html

From Monoliths to Microservices

MonolithFirst

Going directly to a microservices architecture is risky

A monolith allows you to explore both the complexity of a system and its component boundaries

As complexity rises start breaking out some microservices

Continue breaking out services as your knowledge of boundaries and service management increases

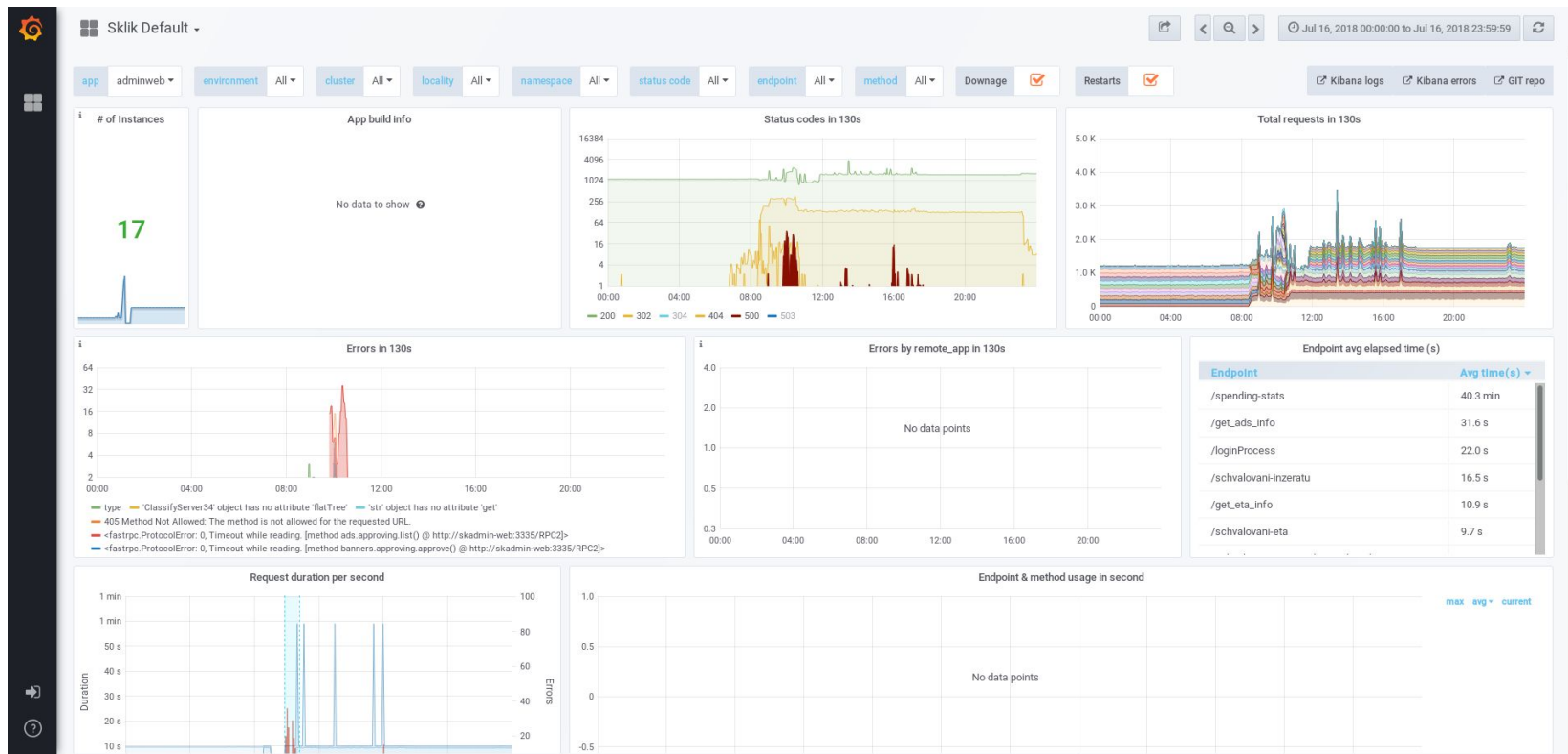https://martinfowler.com/bliki/MonolithFirst.html

Logging

Logging

- Unreliable
- web-scale
- Log what's important to reproduce problem
- Track ID

Monitoring

Monitoring

Monitoring

DevOps, SRE & Conways Law

DevOps, SRE & Conways Law

"Any organization that designs a system … will inevitably produce a design whose structure is a copy of the organization's communication structure."

-- Melvin Conway, 1968

Distributed Monolith

Distributed Monolith

Distributed Monolith

Alberto Brandolini @ziobrando · 19 Mar 2016
OH: "We're going on a microservices architecture. On a shared database." :-0

10   65   73
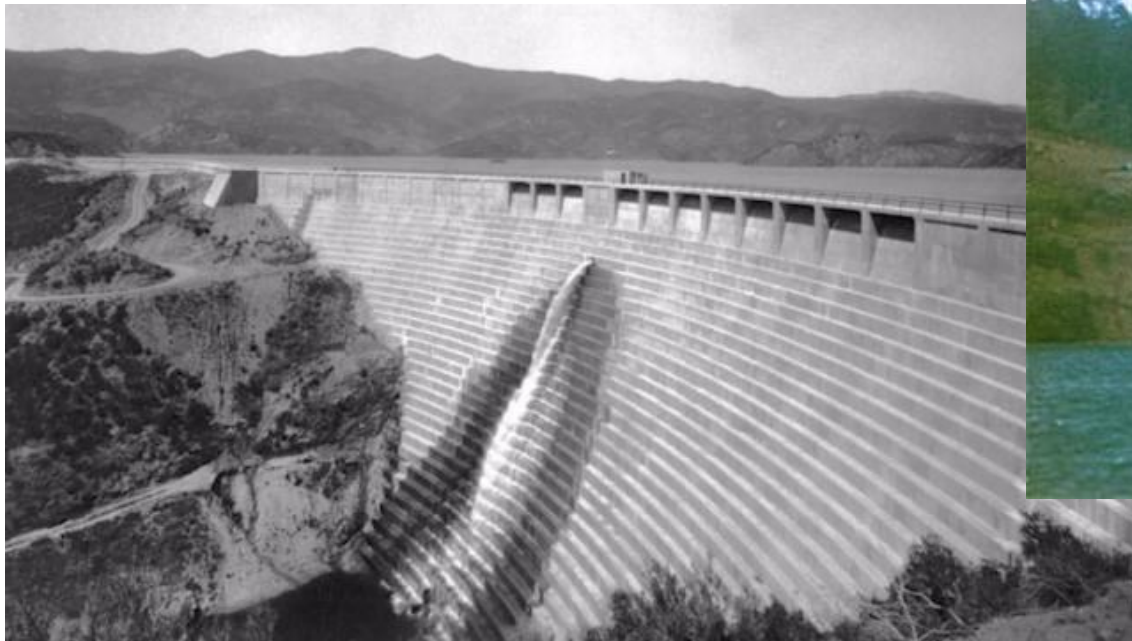
Mathias Verraes
@mathiasverraes   Follow

Replying to @ziobrando

. @ziobrando Here's a diagram of two microservices and their shared database.
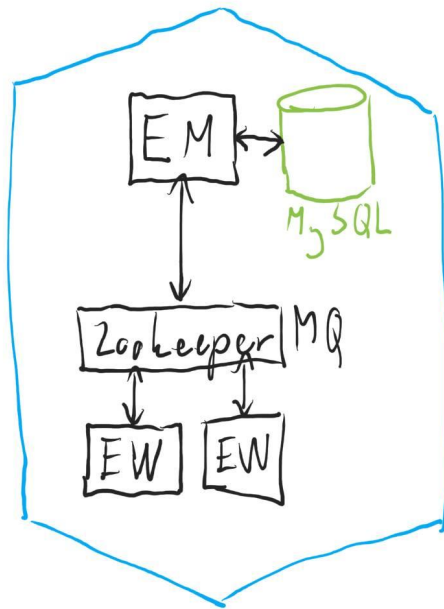
When system fails

When system fails

- Slow/cascade fail
- /readiness /liveness != happiness
- Dependencies

One nice example

Exporter:

- Small enough to fulfill business needs
- Decoupled
- Fail recognition and recovery
- Scalable
- Log-friendly
- Reproducible
- Loved by Users, Devs and Ops

Conclusion

- Monolith or Microservices? It depends..
- Dev part is easy, OPS part is super hard
- [Inverse] Conway Law