A Microservice Story

Architectural transition with ease powered by Quarkus and OpenShift

PUZZLE ITC
changing IT for the better

Nice to meet you

Raffael Hertle
Senior Software Engineer
hertle@puzzle.ch
@g1raffi

One Team

09.02.2021

Agenda

- Story time: The great refactoring
- Microservices in general
- Technological changes and experiences

Mentimeter

09.02.2021

Story time

Base scenario

- Energy law reform
- Build **s**elf-**c**onsumption **c**ommunities (SCC)
- Households could trade energy
- Tons of paperwork

Base scenario

- Use case found
- Smart billing and management of SCC
- Read meter data, store, bill – easy

Base scenario

- Innovation project started
- Blockchain technology
- IoT approach
- Technical buzzwords

Prototype

- Built prototype
- Monolithic approach
- Spring boot back end, Angular front end
- Containerized on OpenShift

Prototype

- Built prototype
- Monolithic approach
- Spring boot backend, Angular frontend
- Containerized on OpenShift

Prototype

- Customers happy
- Reliable, stable system
- Switch to real product – but how?

Prototype

- Discussions about migration path
- Application was already there
- Prototype => Prod-o-type
- Reuse codebase

Product

- Userbase and reading sources multiply
- Codebase and feature landscape grew
- In transition refactoring to smaller domains
- Microservice like architecture



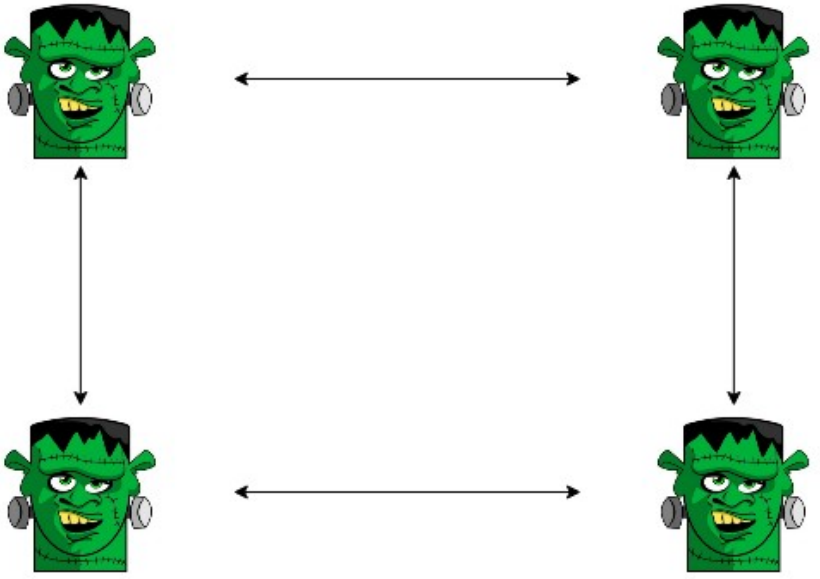

Micro-monoliths

- 4 smaller domains



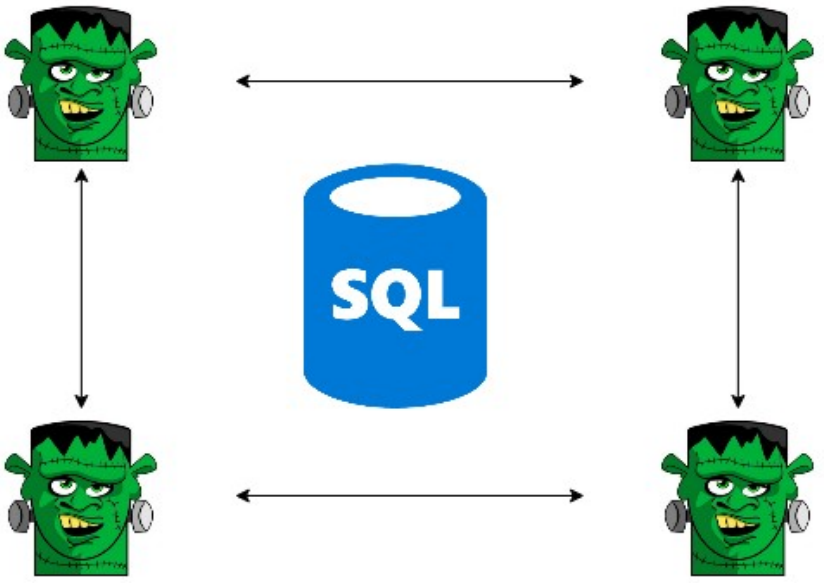

Micro-monoliths

- 4 smaller domains
- REST calls



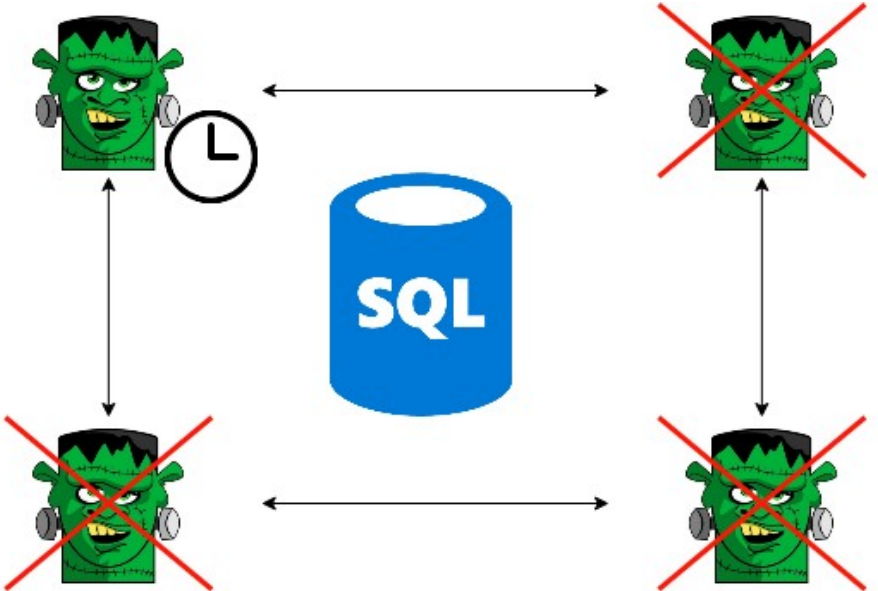
Micro-monoliths

- 4 smaller domains
- REST calls
- Shared database



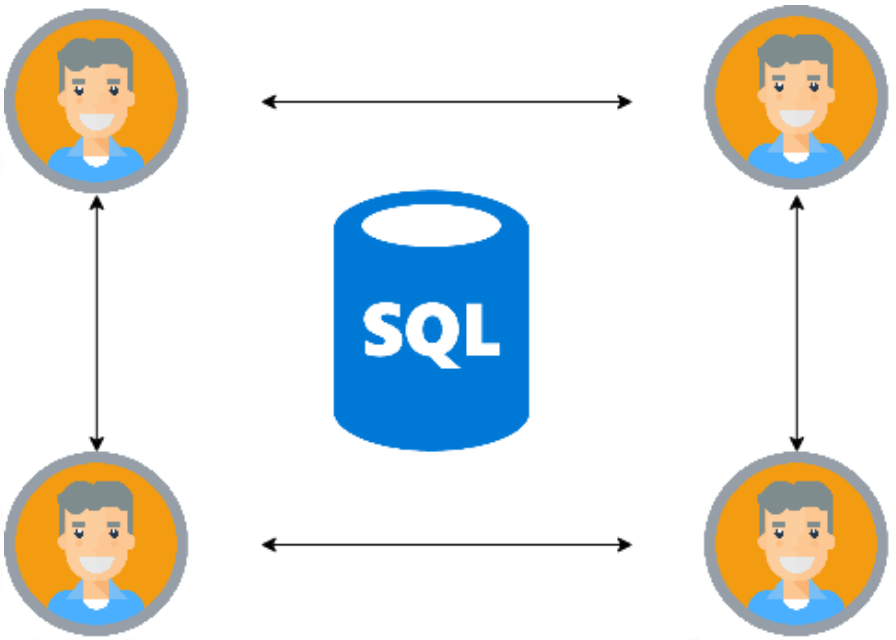
Micro-monoliths

- 4 smaller domains
- REST calls
- Shared database
- Startup times



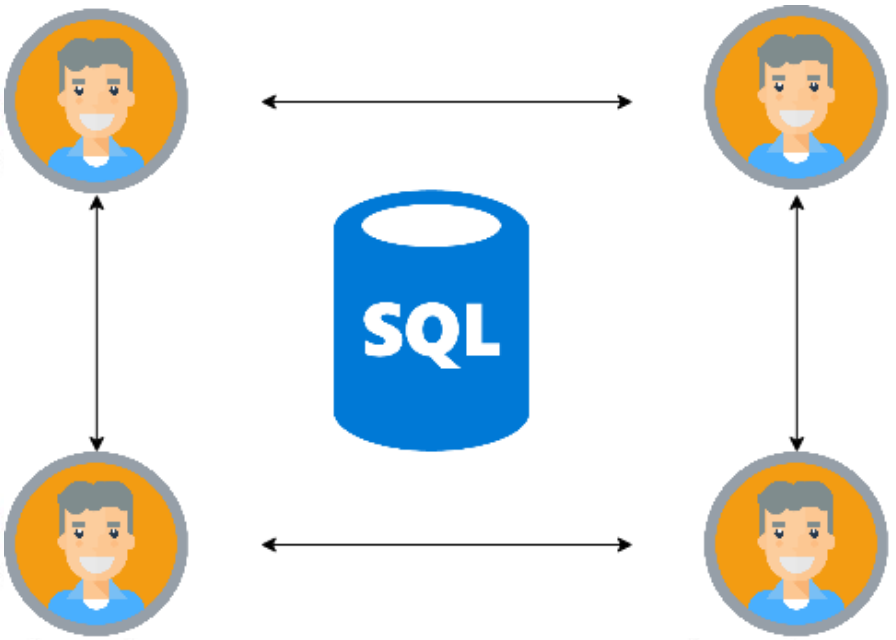
Micro-monoliths

- 4 smaller domains
- REST calls
- Shared database
- Startup times
- System stable



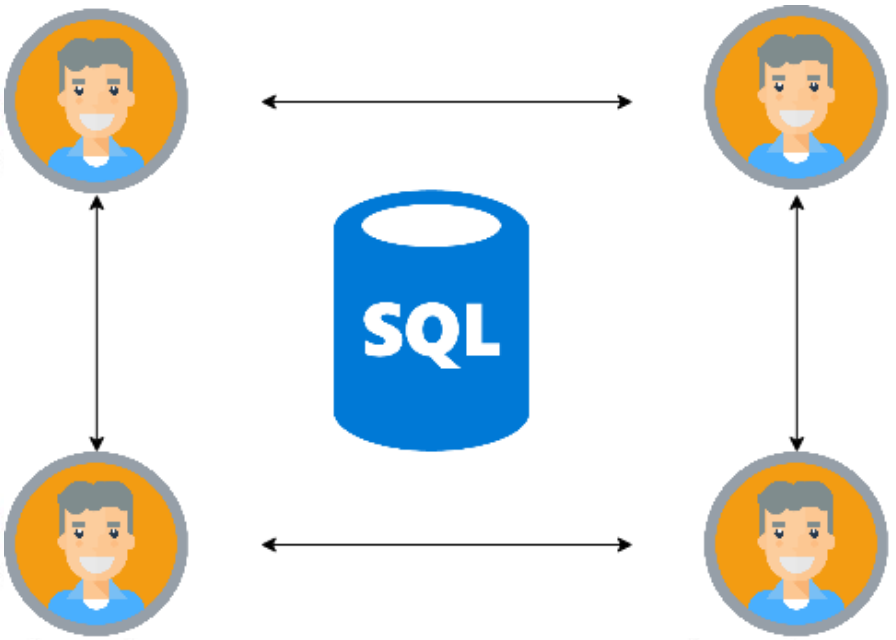
Micro-monoliths

- Happy customers
- More meters
- Blocking operations
- Bottlenecks



Refactoring #2

- Get smaller, more reactive
- True microservices
- Loosen coupling
- Create robustness



Discovery of Quarkus

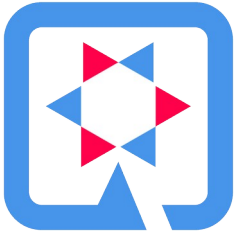
- In parallel project Quarkus came up
- Promising technology
- Fast startup times, low memory footprint
- Standards

Discovery of Quarkus

Discovery of Quarkus

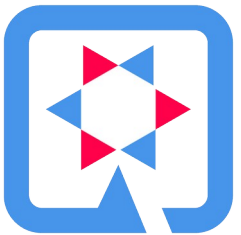
Discovery of Quarkus

- Personal PoC
- Microservices with Quarkus
- Messaging instead of REST
- Independent services, stateless



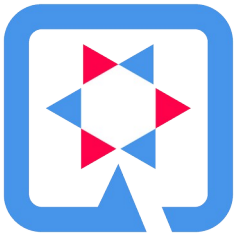
Discovery of Quarkus

- PoC successful
- Transition to true microservices
- Parallel development
- Transition with Strangler approach



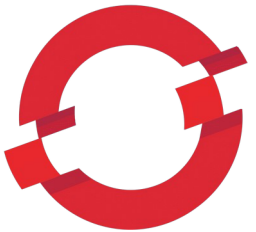
Refactoring with Quarkus

- True microservices
- Landscape from 4 to 14 microservices
- Non-blocking workflows with messaging
- More robust, more reliable, more cloud-native



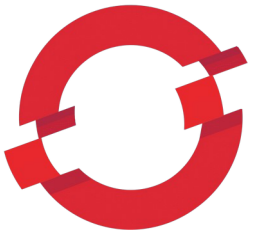
Infrastructure with OpenShift

- 14 services, 14+ infrastructure elements
- Infrastructure as code
- Fast set-up of new environments
- Self-healing system

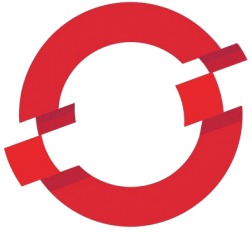


Infrastructure with OpenShift

- Transparent landscape
- Points of failure fast identified
- Observability

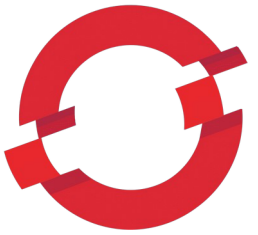


Infrastructure with OpenShift



Recap

- Transition to microservices
- Requirements awareness
- Fail fast
- Adapt to situation



09.02.2021

Microservices

- What are microservices
- Why and when to use them
- Advantages / Disadvantages
- Approach to migrate

09.02.2021

Monolithic Architecture

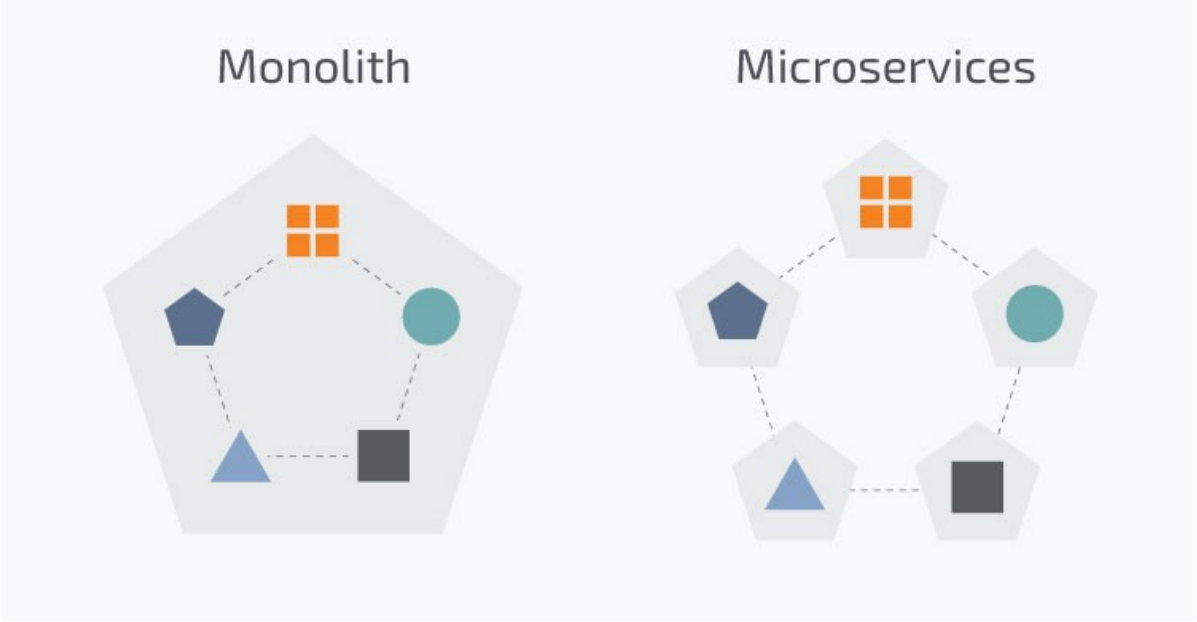
09.02.2021

Monolithic Architecture

- Single code-base
- Single unit deployable
- Independent from other applications
- All domains or business processes in one application

09.02.2021

Microservices Architecture



09.02.2021

Microservices Architecture

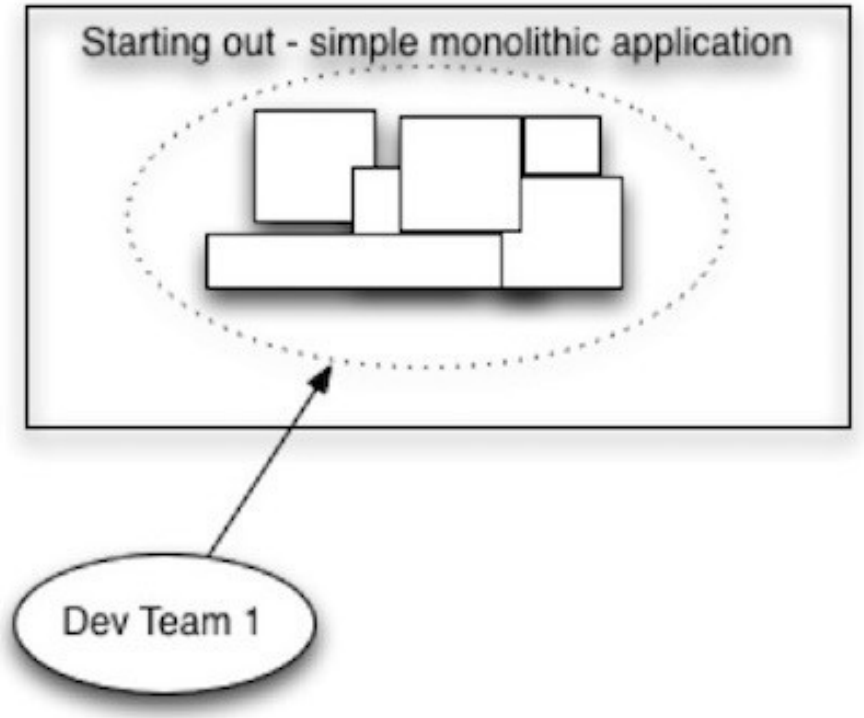
- Small autonomous applications
- Independent life cycles
- Microservice for single responsibility / domain
- Loosely coupled
- Code base per domain / business process

09.02.2021

Microservices Architecture

- When to choose which architecture?

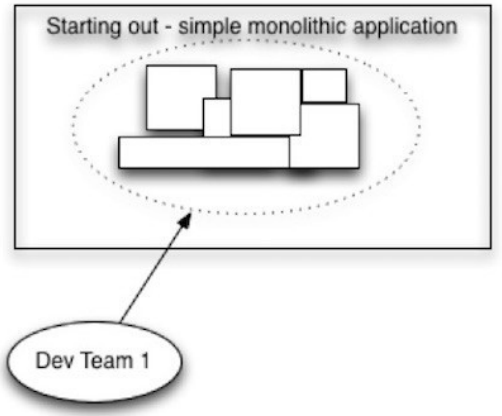
New Application



Advantages of monoliths

Simple architecture

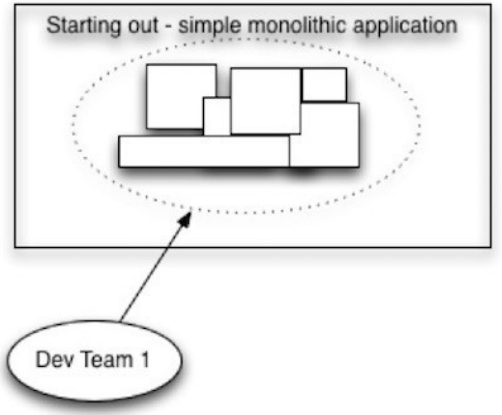
- Everything is local
- High productivity
- Limited attack vectors
- Easy testing
- Performance matches requirements



Advantages of monoliths

Team

- Dedicated team
- Independent releasing
- Features can be released fast
- No dependencies to other teams
- Devs have strong application knowledge



But then...

- Application is a big success
- Users increase
- Traffic increases dramatically
- New features
- Dev team grows

Application and complexity grows

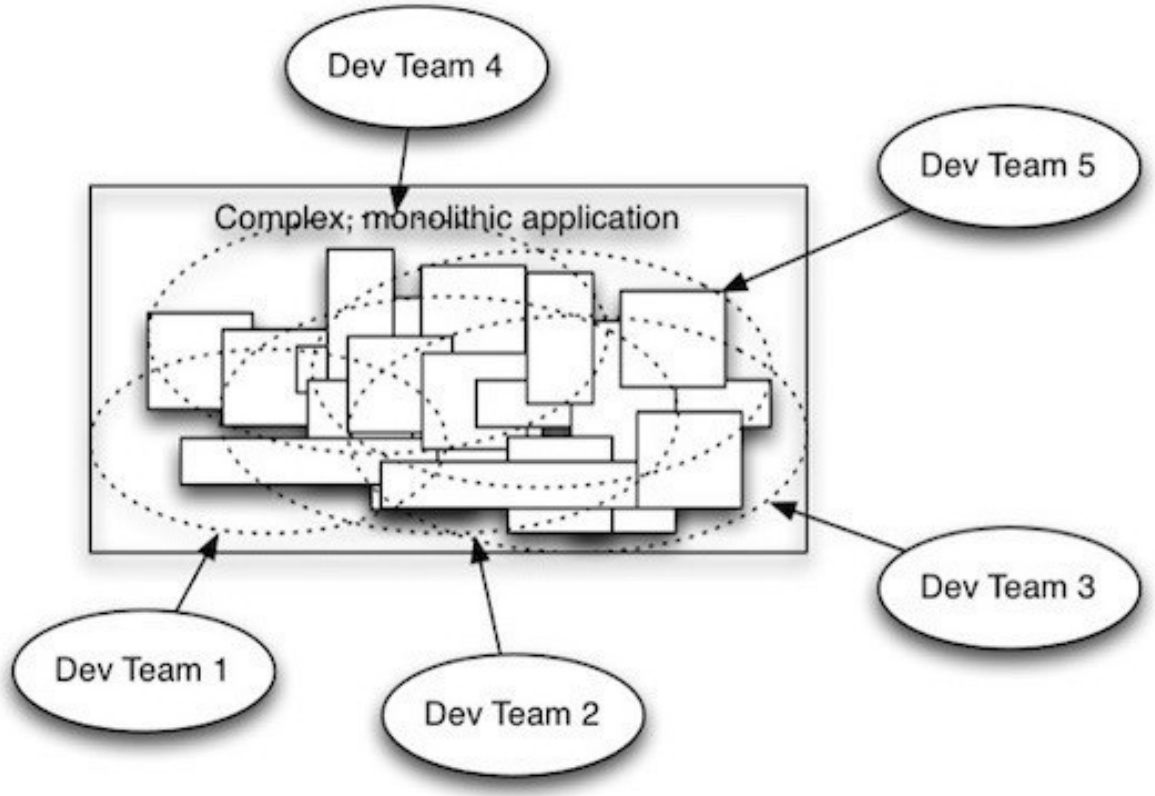
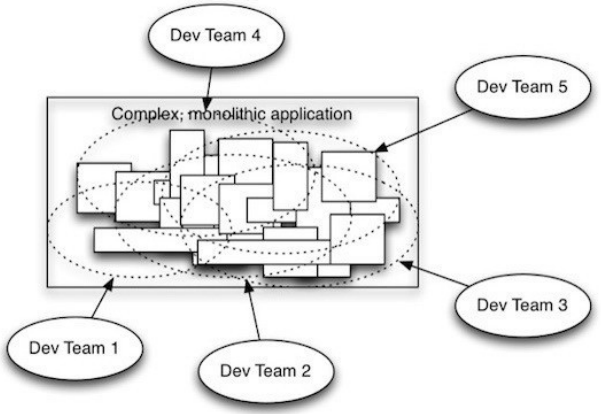


Image source: linkedin.com

Disadvantages of monoliths

Complex architecture

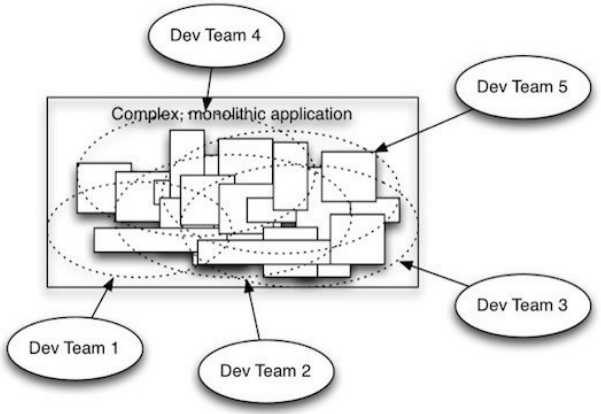
- Architectural changes are difficult
- Impact of code change are hard to estimate
- Keeping up code quality needs extra effort
- Newer technologies are hard to pickup



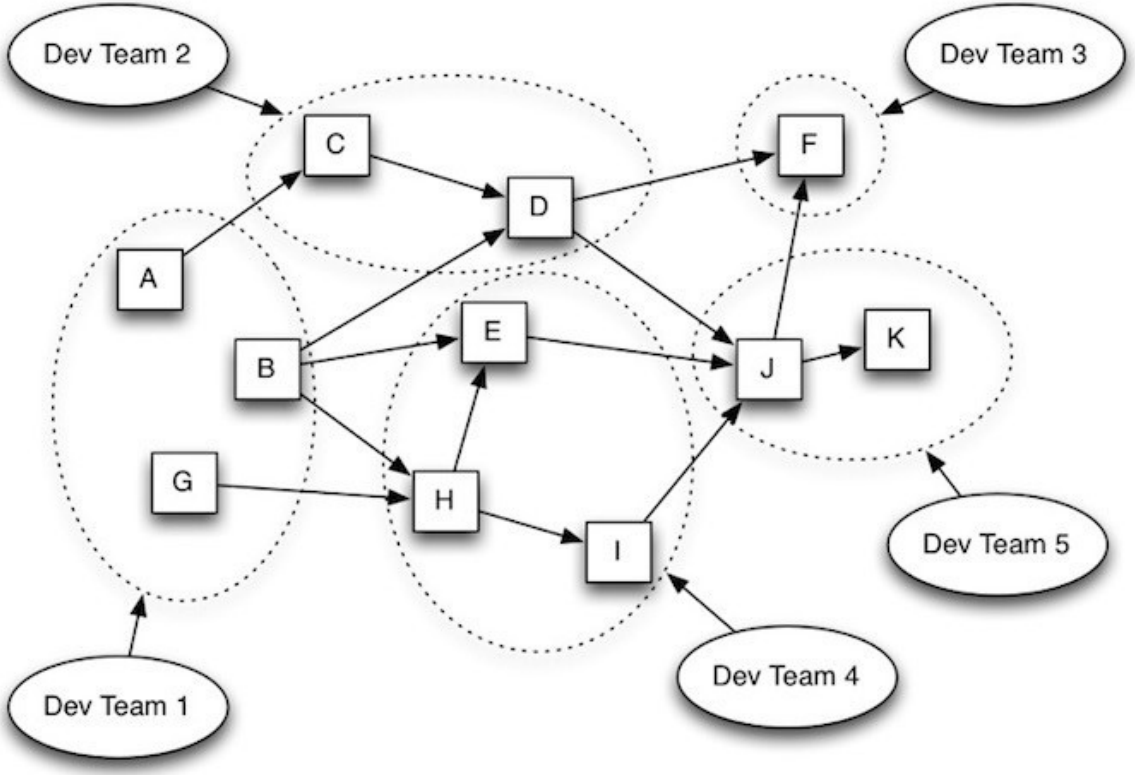
Disadvantages of monoliths (cont.)

Team

- Teams need to be coordinated
- Code changes may collide
- Release planning required
- Feature freeze and test cycles
- Devs have limited knowledge
- Productivity drops



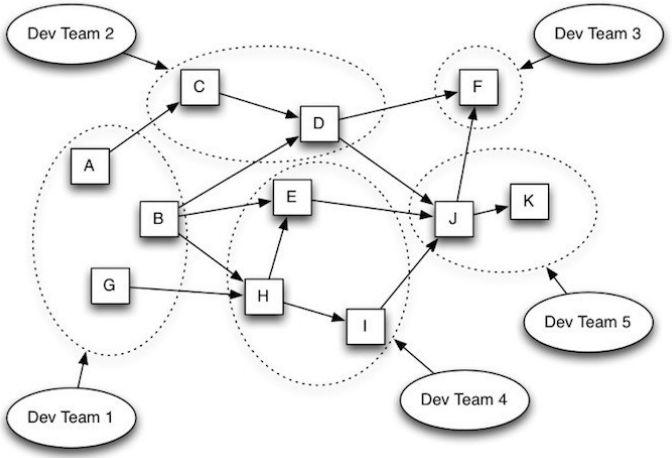
Ok, now what?


Dev Team 2
Dev Team 3
C
F
D
A
E
K
B
J
G
H
I
Dev Team 5
Dev Team 1
Dev Team 4

Advantages of microservices

Architecture

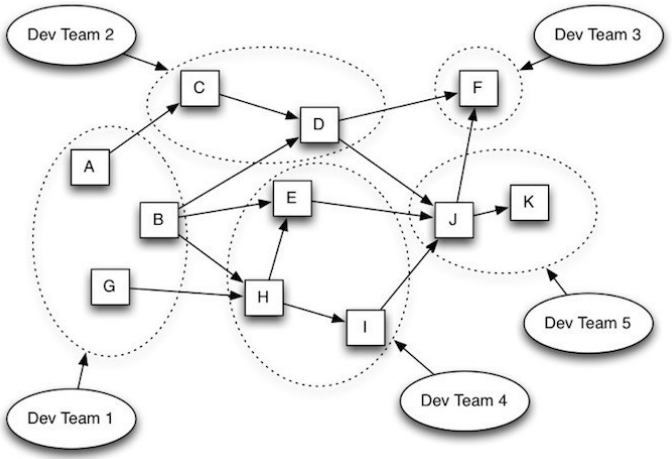
- Independent modules
- Defined boundaries (APIs, Events)
- Loosely coupled (if done right)
- Polyglot (what best fits the task)



Advantages of microservices

Team

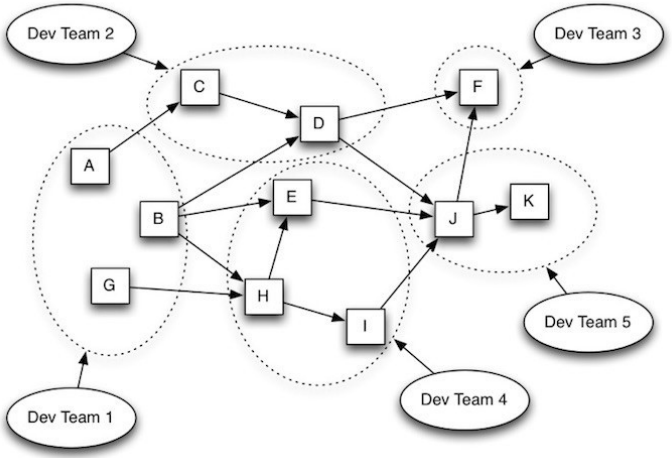
- Teams work independently
- Need to agree to defined boundaries
- Independent within their microservice
- Easier onboarding due limited scope



Advantages of microservices

Deployment and Runtime

- Deploy independently
- Easier scaling of single components
- Bugs may be local only



But ...

Image source: turnoff.us

Disadvantages of microservices

Architecture

- Everything is local does not hold anymore
- Data is distributed, no foreign keys across boundaries
- Keeping data consistent needs extra effort
- Communication and error handling needs extra effort
- Changing the agreed boundaries may be hard

Disadvantages of microservices (cont.)

Deployment and Runtime

- Harder troubleshooting with multiple instances
- Root cause detection can be hard
- More attack vectors

Short Recap

Microservices

- Lead to modularity
- Developers are enforced to respect boundaries
- Enable teams to work and release independently
- Can be replaced as long as boundary is untouched

But they introduce technical complexity

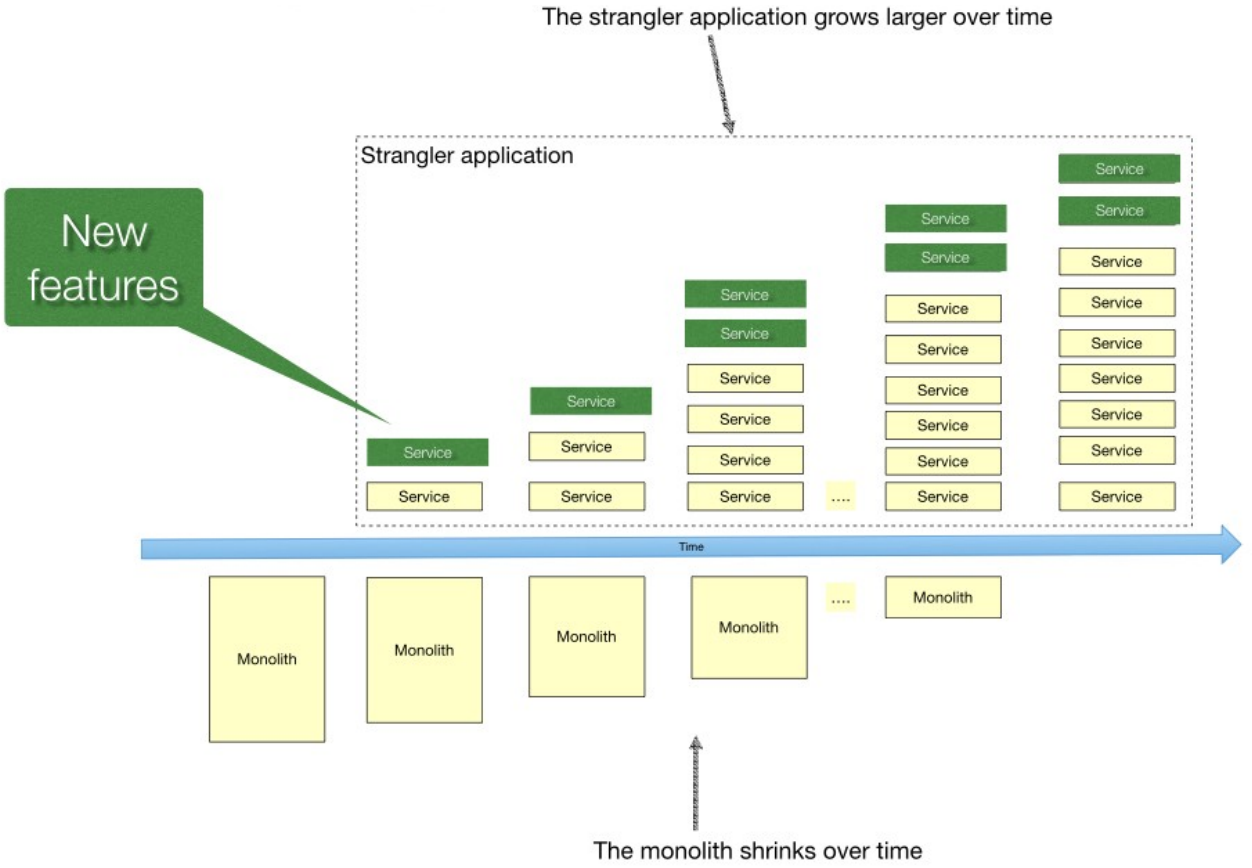
09.02.2021

Microservices Architecture

- How to migrate from monolithic appilcation?

Strangler pattern

Strangler pattern

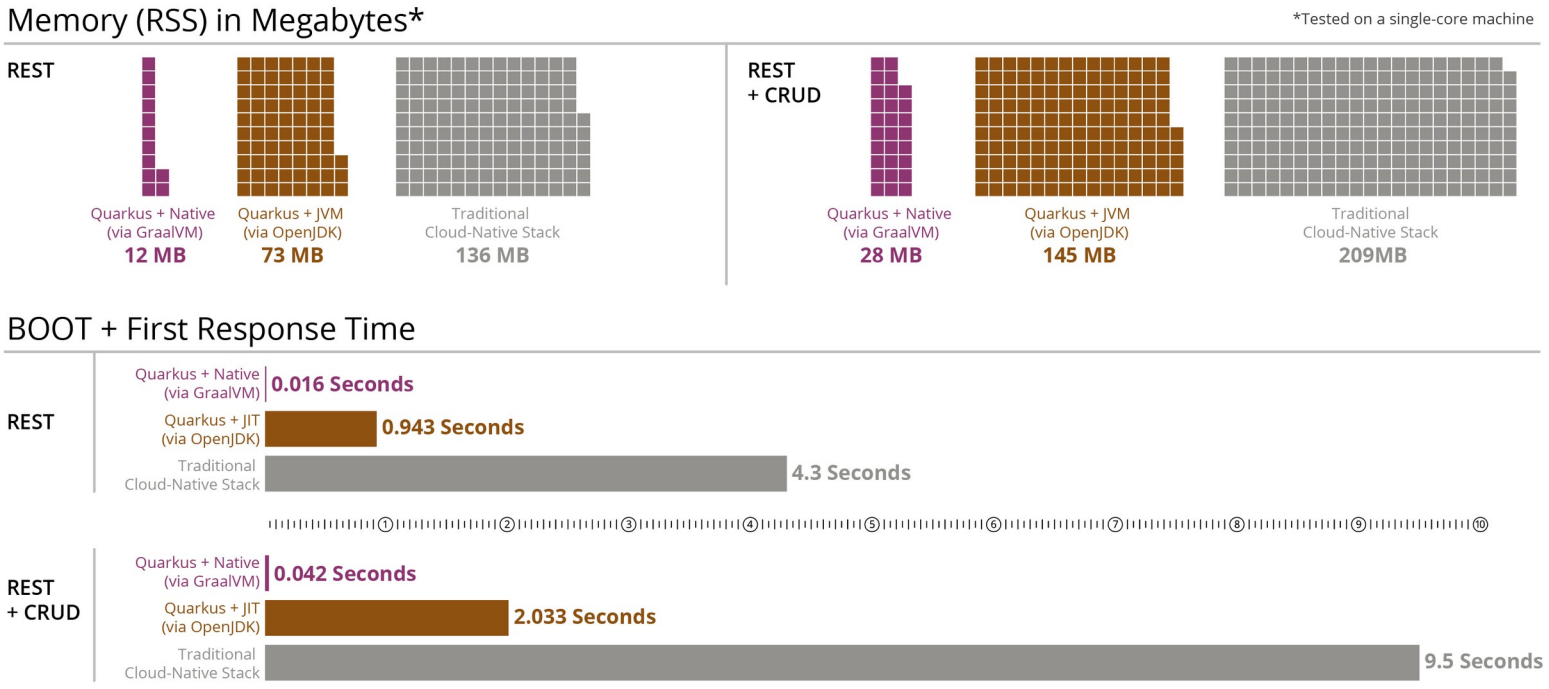


09.02.2021

Technological changes

- Quarkus’ and resource greed
- Asynchronous is the way to go
- Dynamic infrastructure
- Automate all the things

Quarkus' and resource greed

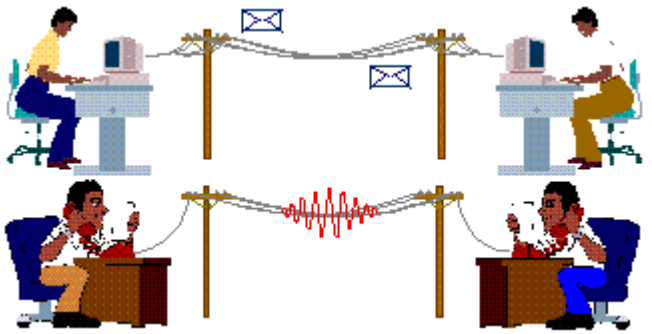


Quarkus’ and resource greed

- Quarkus made us think smaller
- Thinking smaller brought awareness
- Awareness became literal resource greed

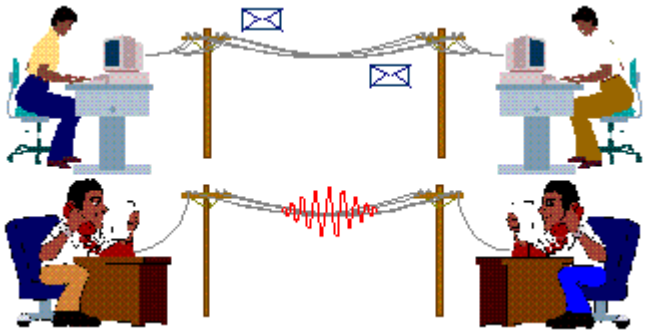
Asynchronous communication

- Synchronous – Telegraph
- Asynchronous - Email



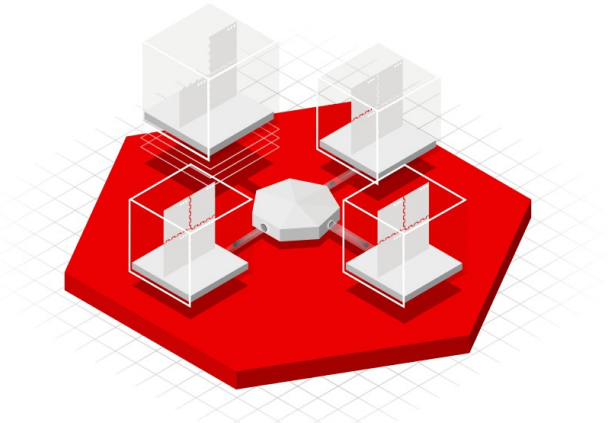
Asynchronous communication

- Reduced the coupling
- Increased system robustness
- Buffer for incoming floods



Dynamic infrastructure

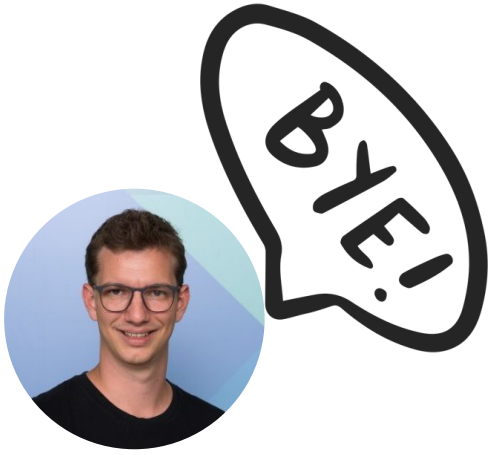
- Infrastructure as Code (IaC)
- Fast provisioning
- Enhances Dev know-how
- Simplifies communication internally



Automate

- Engineers should be lazy
- Automate manual tasks
- Build, test, deploy
- Set up infrastructure

Thanks for listening



Raffael Hertle
Senior Software Engineer
hertle@puzzle.ch
@g1raffi

Thanks