Zero-Downtime-Development Being King in your microservice realm

Jürgen Albert
Data In Motion Consulting GmbH

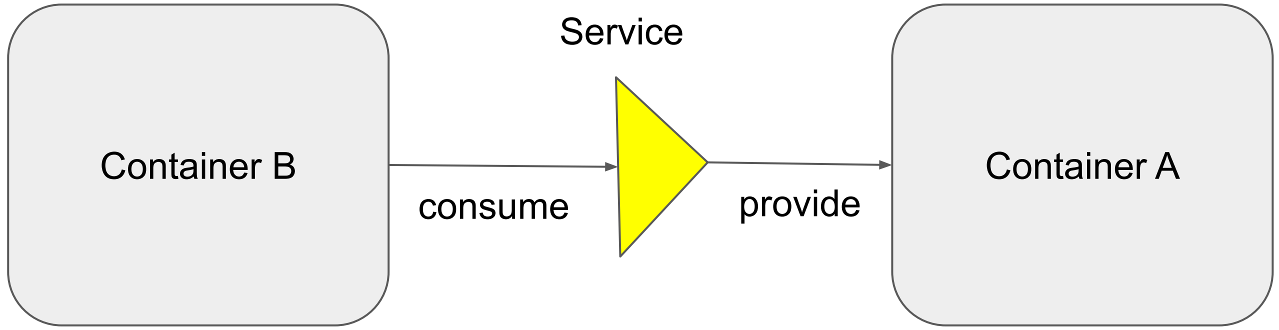
About Us

- Founded in 2010
- Located in Jena/Thuringia - Germany
- Consulting, Independen RnD, Development, Training
- Assisted Development on Complex and Distributed Systems
- Wide Range of Industries
  - Medical
  - Transportation
  - Traffic Control
  - Public Sector
  - Smart City
  - IoT

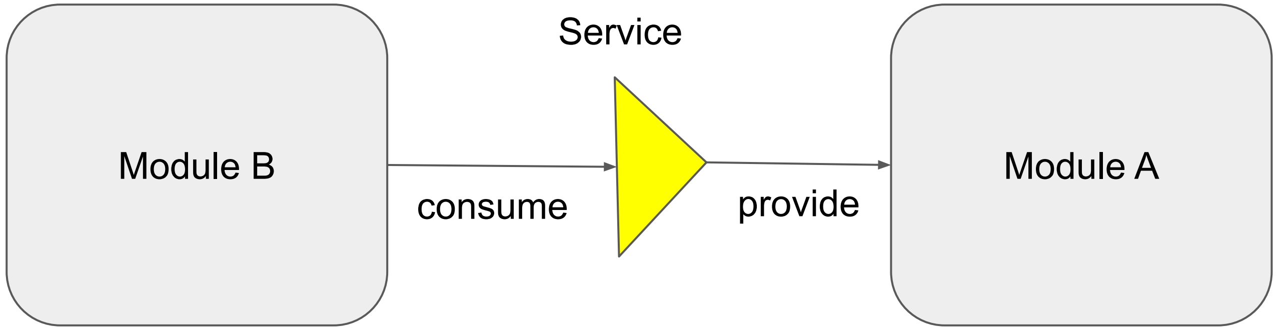
Microservices - A short History

- Are around for 20 years
- Made popular by James Lewis and Martin Fowler around 2014
- Basically they described a Container (Docker) running a small piece of Business logic, providing Interfaces via REST
- Idea behind this: Compartmentalize and decouple your Code
- Teams can concentrate on the Parts they are responsible for
- Use whatever technology seems appropriate for the job

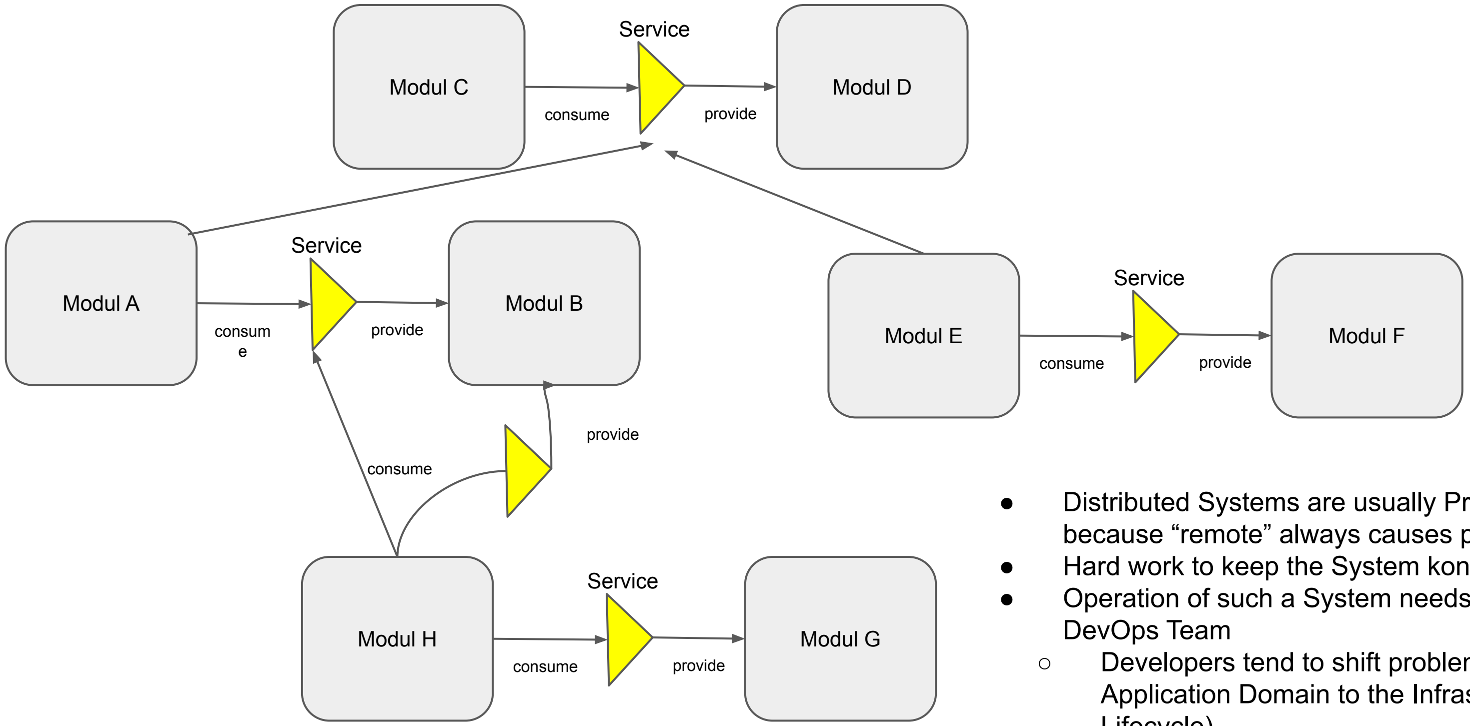
Microservice - In a Nutshell



Microservice - Modularity



Modular System



- Distributed Systems are usually Problematic, because “remote” always causes problems
- Hard work to keep the System konsistent
- Operation of such a System needs a very senior DevOps Team
  - Developers tend to shift problems of the Application Domain to the Infrastructure(e.g. Lifecycle)

Improvements over the Last Years

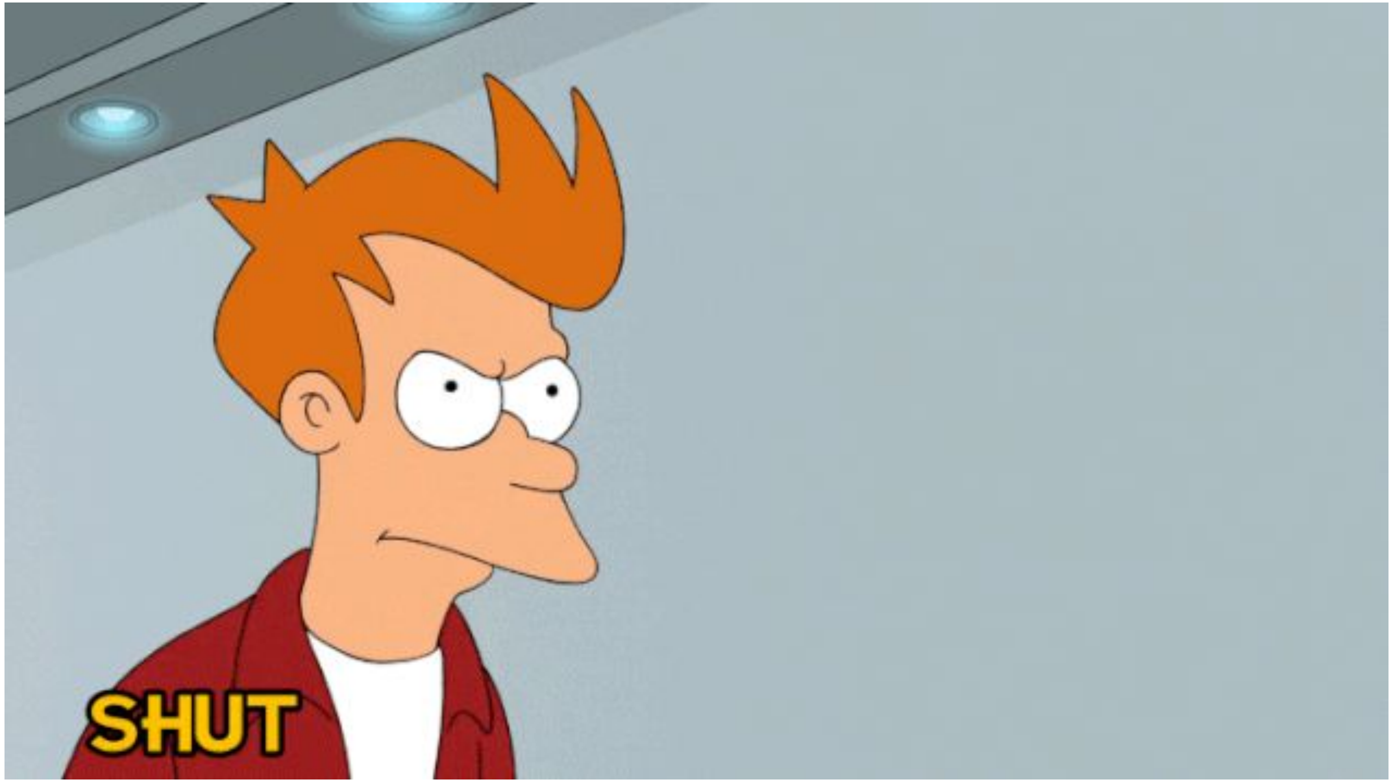
- Automated Builds became CI, CT and CD
- Everybody wants to achieve automated micro releases
- Side effect: Everything needs to run in Docker. Thus Tools became easier to Configure (Apache, Jenkins etc.)

DATA IN MOTION

Álvaro Sánchez-Mariscal
@alvaro_sanchez

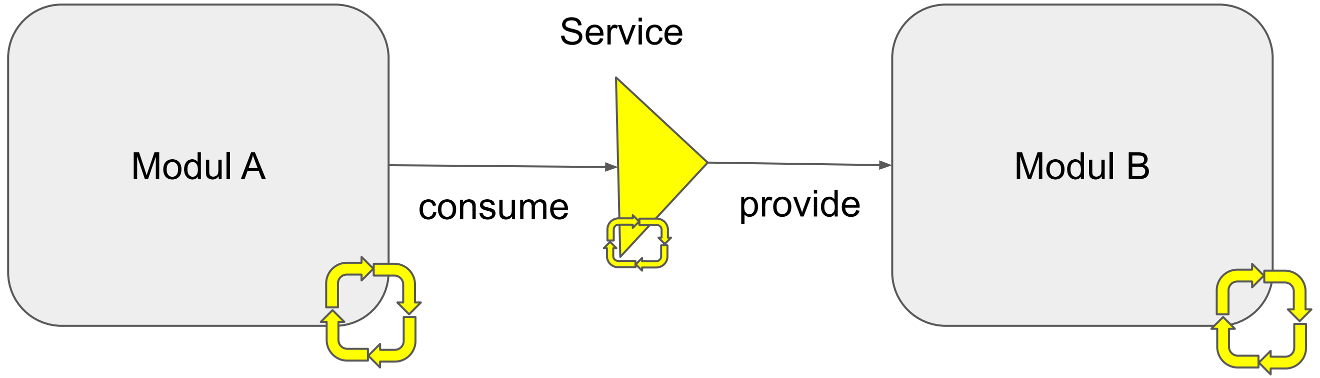
Alright folks, this is serious stuff: how many of you would buy this? RT if you would #microservices

Tweet übersetzen

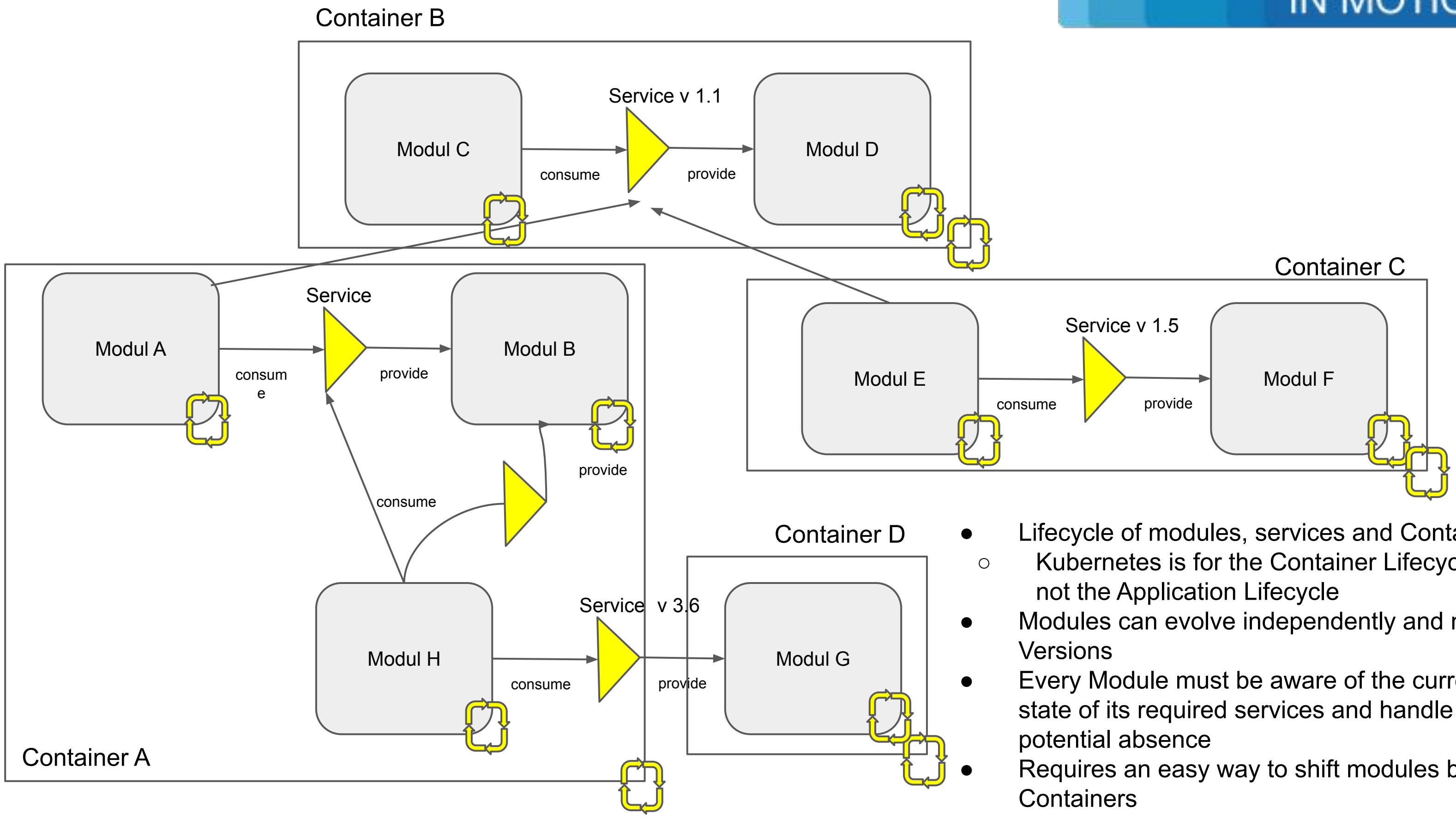

SHUT

- Writing Code became much more convenient
  - Annotations
  - Better Tooling
  - Improved Build Tools
  - Better Test Suites
  - The actual running Code became smaller
- Developers tend to think and develop statically in a dynamic environment
- One big Monolith gets replaced with many small ones
- Writing and Running your Code from the IDE is still oldschool
  - Code -> Run/Deploy -> Debugger until hot code replacement fails -> restart

Modularity in Reality - Lifecycle



Modularity in Reality - Lifecycle and Versions



- Lifecycle of modules, services and Containers
    - Kubernetes is for the Container Lifecycle but not the Application Lifecycle
- Modules can evolve independently and need Versions
- Every Module must be aware of the current state of its required services and handle a potential absence
- Requires an easy way to shift modules between Containers

Modularity is a Mindset and must be learned

“If you want to teach people a new way of thinking, don't bother trying to teach them. Instead, give them a tool, the use of which will lead to new ways of thinking.”

— R. Buckminster Fuller

Requirements for such a tooling

- Developers need to see the impact and dynamic of their actions right away
- One Goal of Microservices is Zero-Downtime and micro releases. This must be the same at Development time.
- Keep track of versions and help with the assembly of what you need
- Allow for a Modulelith or Microservice Monolith
- Allow for easy Moving of Modules and distributing you services

MO'NIQUE IN CHICAGO

STED LEADERSHIP
for a
ONGER AMERICA
JEB2016.COM

Conclusion

- OSGi addresses problems that usually arise in complex systems early on (Versions, Lifecycle, Dependencies)
- OSGi makes bad design harder, but not impossible
- OSGi is uniquely suitable for Microservice Environments, because the same concepts that apply for the big picture, also apply in the JVM and at development time
- OSGi provides open industrial standards without being vendor locked

Thanks for listening!

Resources:

Web:	https://www.datainmotion.de
	https://enroute.osgi.org/
	https://osgi.org/specification/osgi.core/7.0.0/ch01.html
	https://osgi.org/specification/osgi.cmpn/7.0.0/introduction.html
Blog:	https://www.datainmotionblog.de/blog/
Git:	https://gitlab.com/gecko.io/talks/2019_08_22_jug_zero-downtime-development

DATA
IN MOTION

Questions?