

Microservice Architecture

Code & Context, WS 2024 / 25

Prof. Dr.-Ing. Stefan Bente

Warum DevOps?

Technology
Arts Sciences
TH Köln

Was, wenn es **viele** kleine, kreative, **unabhängige** Entwicklungsteams wie dieses hier gibt?



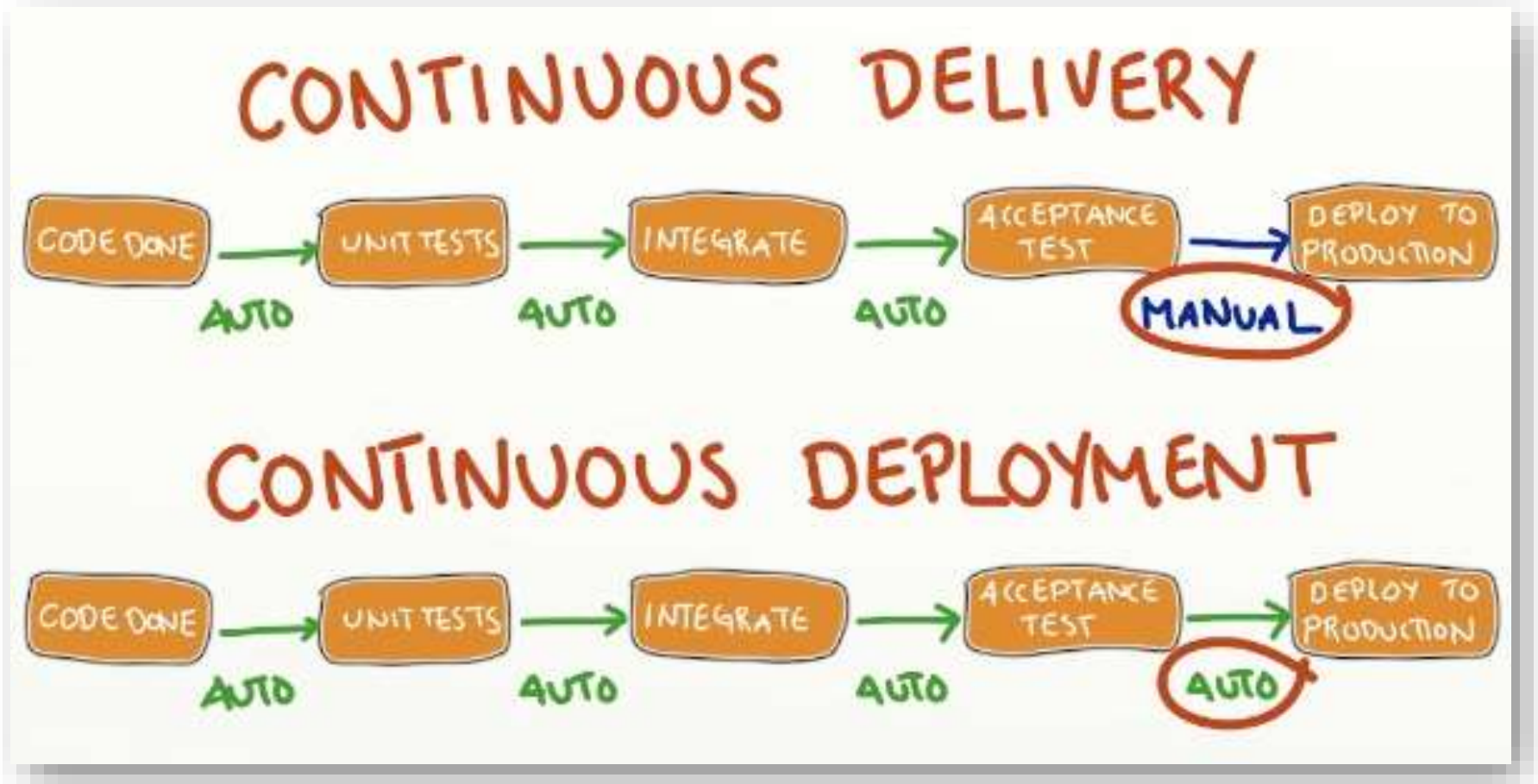
Prinzip 1: Autarke, entscheidungsbefugte Entwicklungsteams

- Für jeden Microservices ist jeweils ein **unabhängiges** Team zuständig
 - Team hat (fast) völlige Entscheidungsfreiheit über Technologie
 - Einige starke Architekturprinzipien werden zentral überwacht
- Kompromisse bei ...
 - Konzeptueller Integrität
 - Datenhaltung konsistent und redundanzfrei
- Entwicklungsansatz: i.d.R. **agil**



Bild: "Daily sprint meeting" by Klean Denmark. Licensed under CC BY-SA 2.0 via Wikimedia Commons.

Prinzip 2: DevOps und unabhängige Deploybarkeit



Sundman, Y. (2013): Continuous Delivery vs Continuous Deployment. Blog post, <http://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment>. Retrieved 08-Sep-15

Prinzip 2: DevOps und unabhängige Deploybarkeit

- DevOps = Entwicklung (Dev) und Betrieb (Ops) eng verzahnt
 - Bei Microservice sogar: "You build it – you run it"
- Entwicklungsteam ist auch für den Betrieb zuständig
- Radikaler Paradigmenwechsel zu herkömmlichen IT-Organisationen
 - Dort: meist strikte Trennung von Entwicklung und Betrieb
- Häufiges Produktivsetzen ist üblich (z.B. wöchentlich)
 - Voraussetzung: Continuous Delivery / Deployment

Wie ist das im MS Dungeon umgesetzt?



`git commit`
`git push`

In der **CI/CD Pipeline**

- Getriggert durch `git push`
- Definiert durch `.gitlab-ci.yml`
- Ausgeführt auf speziellem Runner, siehe https://gitlab.com/msd-players/msd-workflows/-/blob/main/README.md?ref_type=heads#gitlab-ci.yml
- Kann Tests ausführen (in unserem Fall nicht ☹)
- Erzeugt Docker-Image & legt es in Gitlab-Registry
- Das Image wird dann (def. durch `helm chart`) auf das Kubernetes-Cluster gespielt