Die neue Welt der Microservices und Container

Lernziele

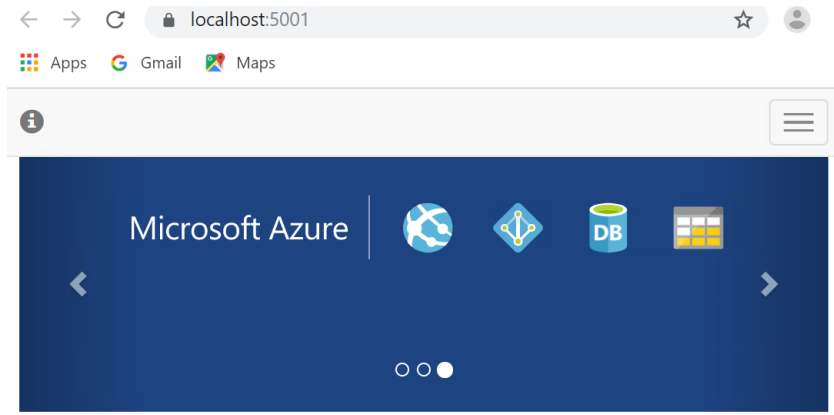
- ★ Sie haben einen ersten Überblick über die neue Welt der Microservices und Container.
- ★ Sie haben eine erste Docker und Kubernetes Umgebung zum laufen gebracht.

Zeitlicher Ablauf

- ★ Die Sicht des Entwicklers
- ★ Microservices und Container
- ★ Wie bestehende Anwendungen einbinden?
- ★ Perspektive der CEOs/Entscheider
- ★ Übung
- ★ Reflexion
- ★ Lernzielkontrolle

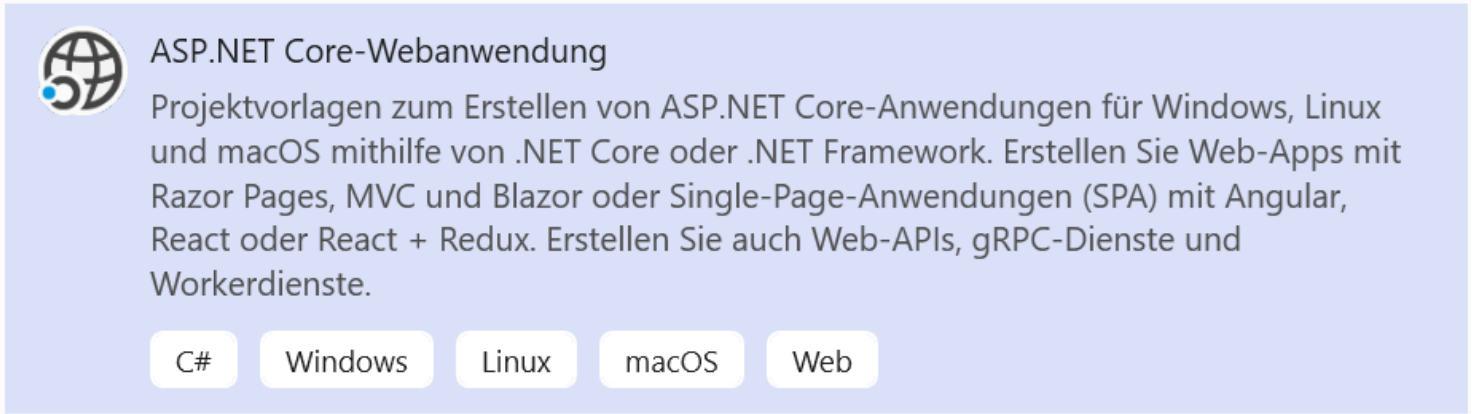
Die Sicht der Entwickler

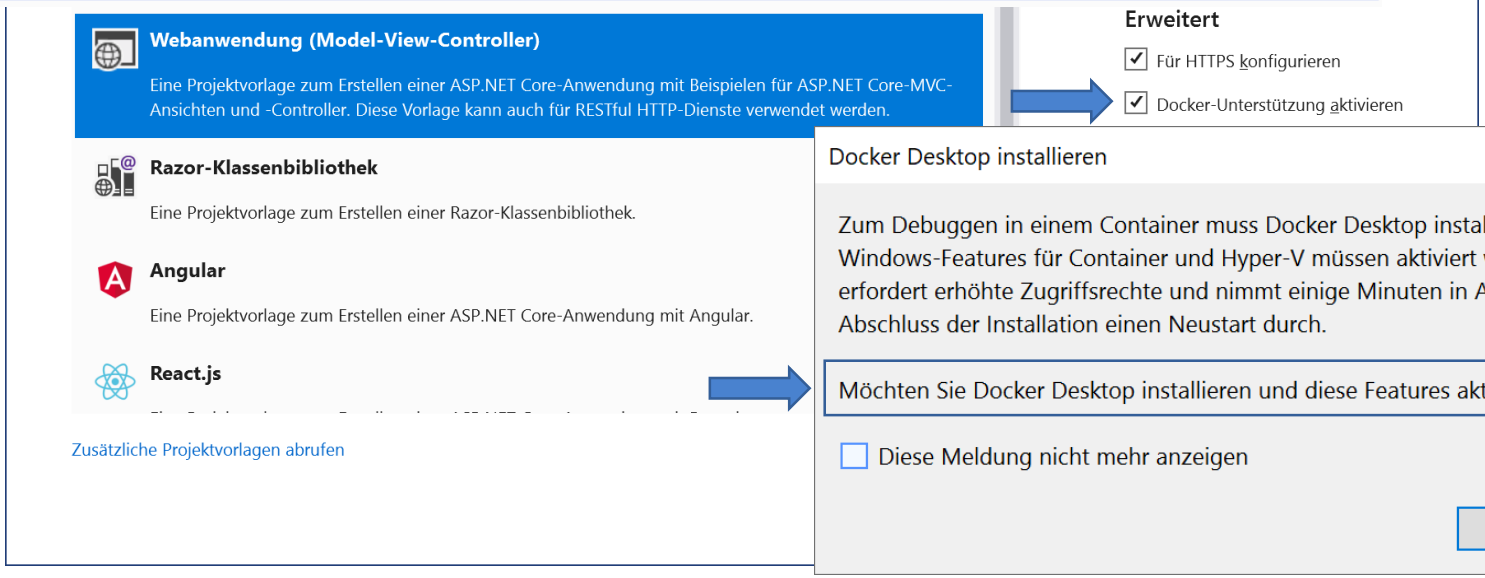
Webanwendung (Ausgabe)

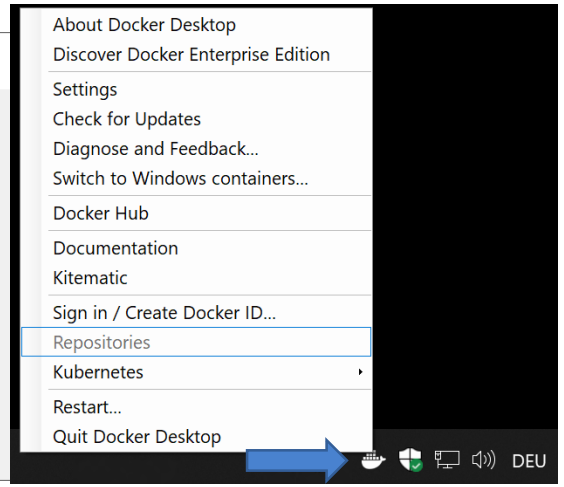


Application uses

- Sample pages using ASP.NET Core MVC
- Theming using Bootstrap

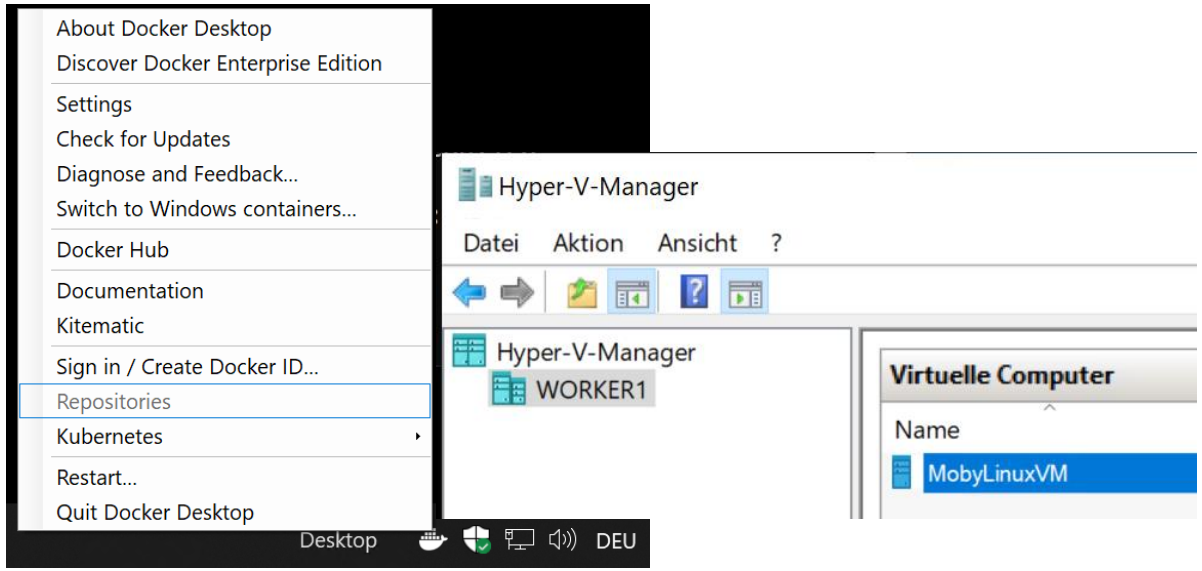




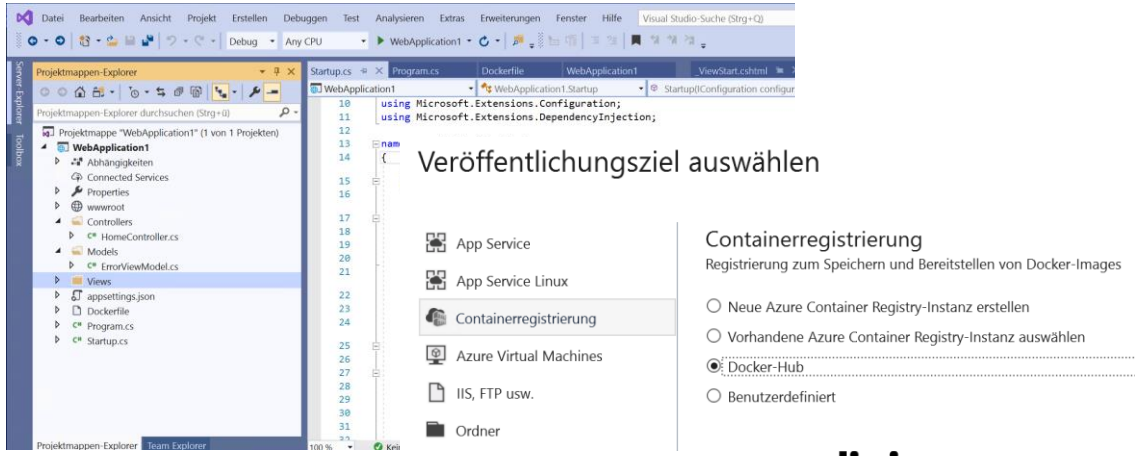


Was ist passiert?

- ★ Docker Desktop (Applikation und Service) wurde installiert (= Container Umgebung)
- ★ Hyper-V (Virtualisierung) wurde aktiviert
- ★ Eine Linux VM wurde erstellt



- ★ Eine C# Webanwendung wurde erstellt
- ★ Kompiliert in einem Container und mit allen Abhängigkeiten, als Container Image verpackt
- ★ Das Container Image wurde in einem Container gestartet
- ★ Optional wurde das Container Image auf https://hub.docker.com veröffentlicht

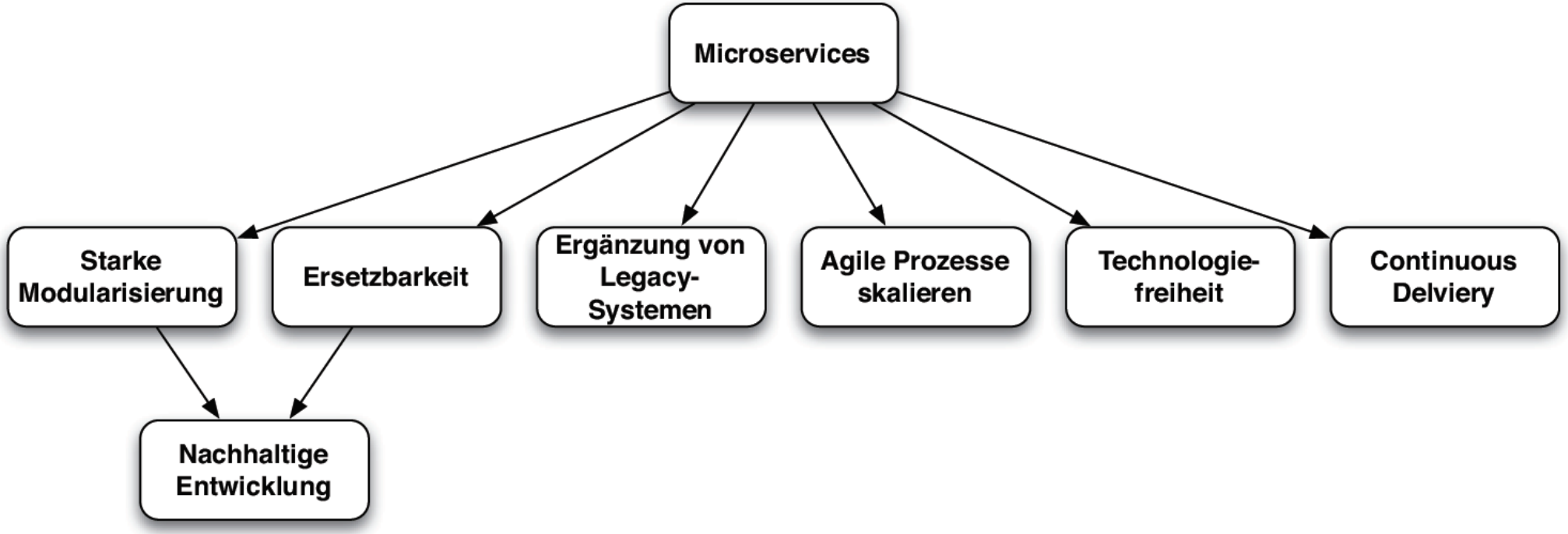


Was hat der Entwickler gemacht?

- **Der Entwickler hat das Architekturmuster Microservices verwendet um eine Webanwendung zu erstellen.**

- Was sind Microservices?
  - Microservices – ein Ansatz zur Modularisierung von Software.
  - **Das Neue**: Microservices nutzen als Module einzelne Programme, die als eigene Prozesse laufen. Der Ansatz basiert auf der UNIX-Philosophie. Sie lässt sich auf drei Aspekte reduzieren:
    - Ein Programm soll nur eine Aufgabe erledigen, und das soll es gut machen.
    - Programme sollen zusammenarbeiten können.
    - Nutze eine universelle Schnittstelle. In UNIX sind das Textströme. Bei Microservices das Internet (REST)

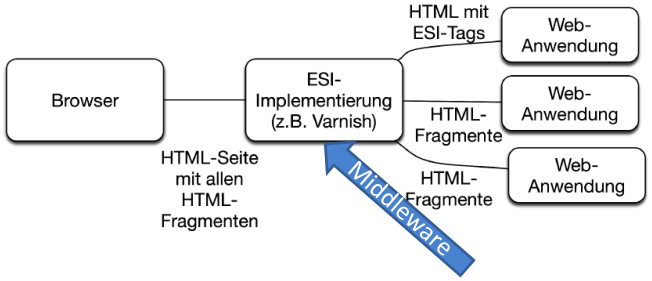
Einsatzszenarien vom Microservices



Quelle: Buch Microservices, Grundlagen flexibler Softwarearchitekturen

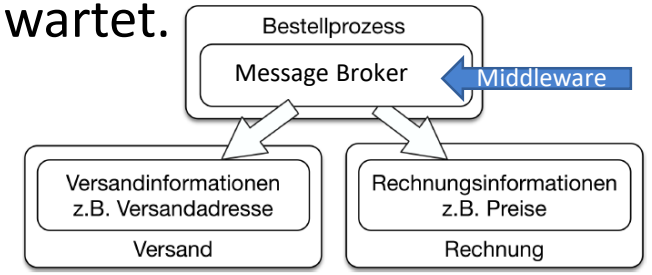
Konzepte für Microservices (nicht abschliessend)

★ Frontend Integration
- Verschiedene Microservices werden, z.B. mittels Edge Side Includes (ESI), in einem Frontend integriert.

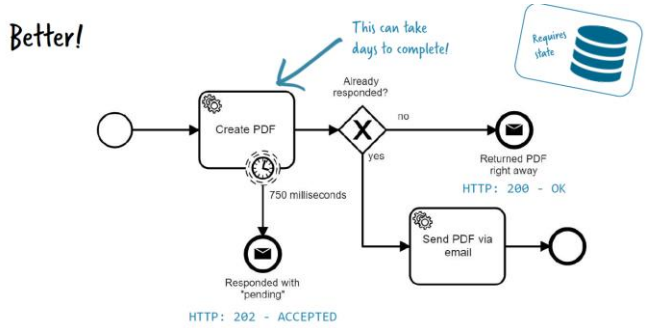


★ Synchrone Microservices (Klassisch)
- Ein Microservice ist synchron, wenn er bei der Bearbeitung von Requests selber einen Request an andere Microservices stellt und auf das Ergebnis wartet.

★ Asynchrone Microservices
- Der Microservice schickt einem anderen Microservice (Event Bus, Message Broker) einen Request, wartet aber nicht auf eine Antwort.

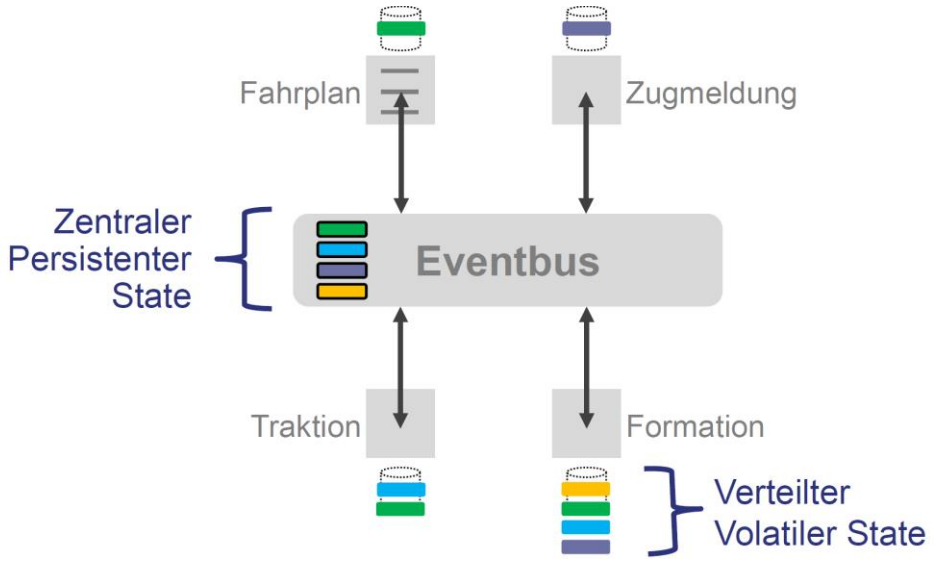


★ Orchestriert
- Eine Workflow Engine, z.B. BPMN Engine, übernimmt die Steuerung der Microservices



Unser Unternehmen und Microservices

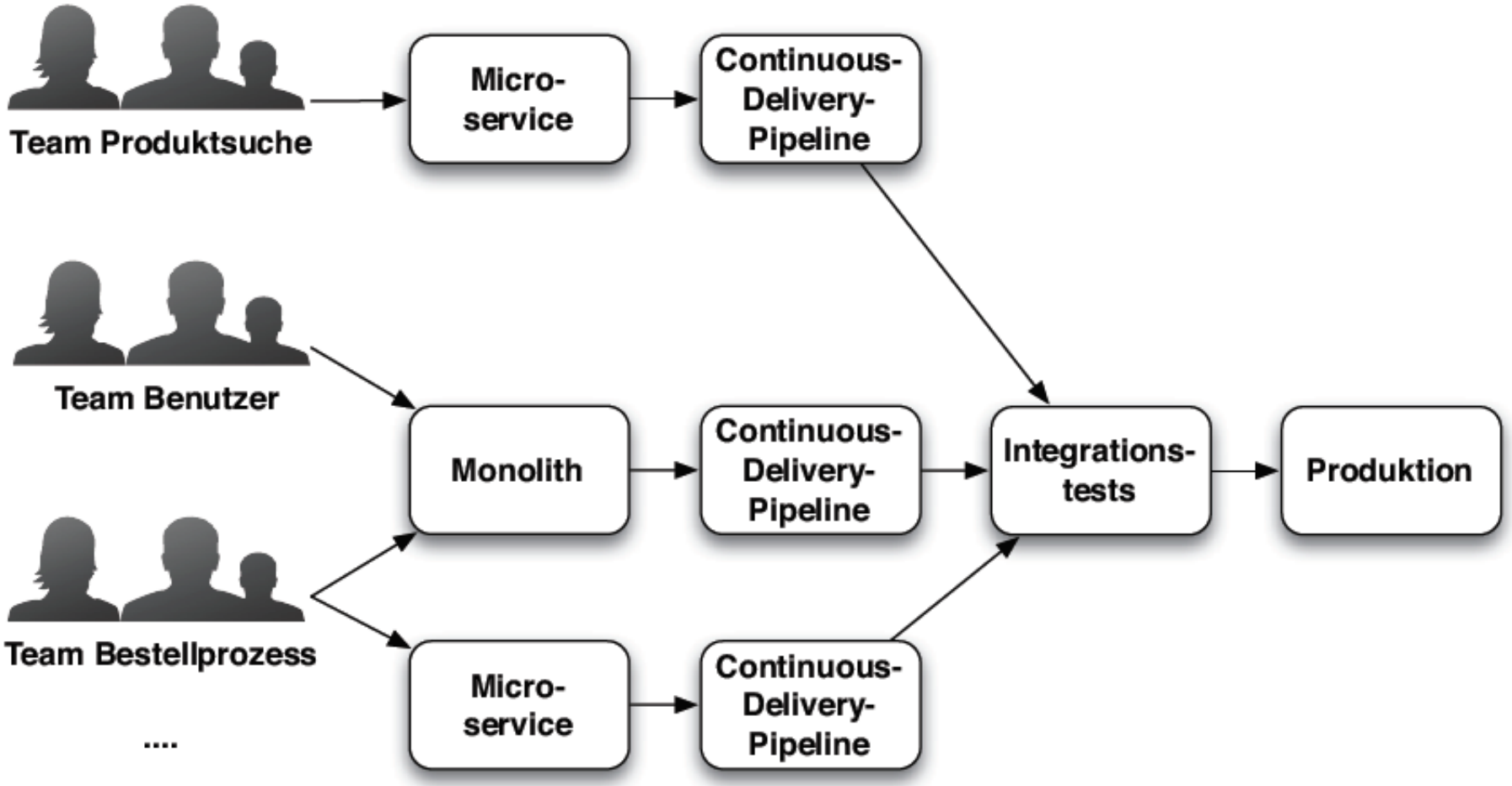
- ★ **Was passiert im Hintergrund?**
  - **Statt monolithischer Anwendungen – entkoppeln der Dienste, und einzeln als ein Microservice, in je einem Container betrieben und vernetzt.**
  - Dies ermöglicht eine massive Skalierbarkeit, Verfügbarkeit und Portierbarkeit.
  - Bietet die Möglichkeit durch Modularisierung und sukzessive Aktualisierung *Big Bang Releases* zu entschärfen.
- ★ **Aber es setzt voraus:**
  - Orchestrierung der Microservices untereinander
  - Container Images für Microservices
  - Ein hoher Grad an Automation
  - Mehr Middleware (Web Cache, Message Broker für Eventbus, Workflow Systeme etc.)



Quelle: https://www.jug.ch/html/events/2019/architektur_mit_apache_kafka_bs.html

Quelle: Buch Skalierbare Container-Infrastrukturen, Kapitel 2.2

Wie bestehende Anwendungen einbinden?



Quelle: Buch Microservices, Grundlagen flexibler Softwarearchitekturen

Perspektive der CEOs/Entscheider

★ Produkt/die Marke
  - Hohe Qualität
  - Umsetzung Geschäftsanforderungen in Technologie
  - **Neue Variationen und Prototypen in kürzester Zeit ausrollen** (Time-to-Market).

Quelle: Buch Skalierbare Container-Infrastrukturen, Kapitel 2.2.3

Übung: Edge Side Includes (ESI) (1)

★ Microservice: Frontend Integration besteht aus drei Microservices:

- scs-demo-esi-common stellt allgemeine Kopf- und Fusszeilen für Services bereit.
- scs-demo-esi-order erledigt die Auftragsbearbeitung
- Varnish interpretiert die ESI-Befehle (Edge Side Includes)

★ Aufgabe

- Startet die Microservices und vergleicht die, mittels ESI zusammengestellte, mit der Order Standalone Variante.

Übung: Edge Side Includes (ESI) (2)

★ Starten der Git/Bash Shell, Clonen lernkube Repository und Starten VM (wenn noch nicht erfolgt)
- `git clone https://github.com/mc-b/lernkube`
- `cd lernkube`
- `cp templates/DUK.yaml config.yaml`
- `vagrant plugin install vagrant-disksize`
- `vagrant up`
- `exit`

★ Wechsel auf dem Desktop in das lernkube Verzeichnis und starten der CLI durch Ausführen von
- `kubeps.bat` oder Start Git/Bash, Wechsel nach `cd lernkube`, Umgebungsvariablen setzen mittels `source kubeenv`

★ und Starten der ersten Microservices
- `kubectl apply -f misegr/ewolff/`

★ http://localhost:32080/ und http://localhost:32090/ anwählen und Unterschiede suchen.

★ Weave Oberfläche (im CLI) mittels `weave` Starten und Verbindungen Analysieren.

Reflexion

- ★ Container und Microservices verändern die IT Landschaft Technologisch und Organisatorisch (DevOps).
- ★ Container und Microservices Beschleunigen den Time-to-Market von neuen Produkten.
- ★ Container und Microservices brauchen eine effiziente Continuous Delivery / Continuous Integration (CI/CD) Pipeline.
- ★ In der Entwicklung reden wir von Microservices meinen aber modulare Anwendungen in Container Images verpackt.

Lernzielkontrolle

- ★ Sie haben einen ersten Überblick über die neue Welt der Container und Microservices.
- ★ Sie haben eine erste Docker und Kubernetes Umgebung zum laufen gebracht.