

VIRTUELLES NETZWERK

- entsteht durch die Anwendung von Netzwerkvirtualisierung(-techniken)
- es werden logische / nicht physikalische Netzwerke gebildet
- wired oder wireless

MOTIVATION

- Welchen Nutzen bringt ein Virt. Netzwerk:
- Isolation: private network in der Cloud -> Security
- Performance: Direkte Kommunikation zwischen Netzwerk Komponenten (innerhalb des VNets)
- Organisation: Backend Servers (DBs, Storage etc.) vs Internet Facing Servers (Web Apps)
- Kontrolle: security policies, DNS routing, FW-Regeln, Subnets

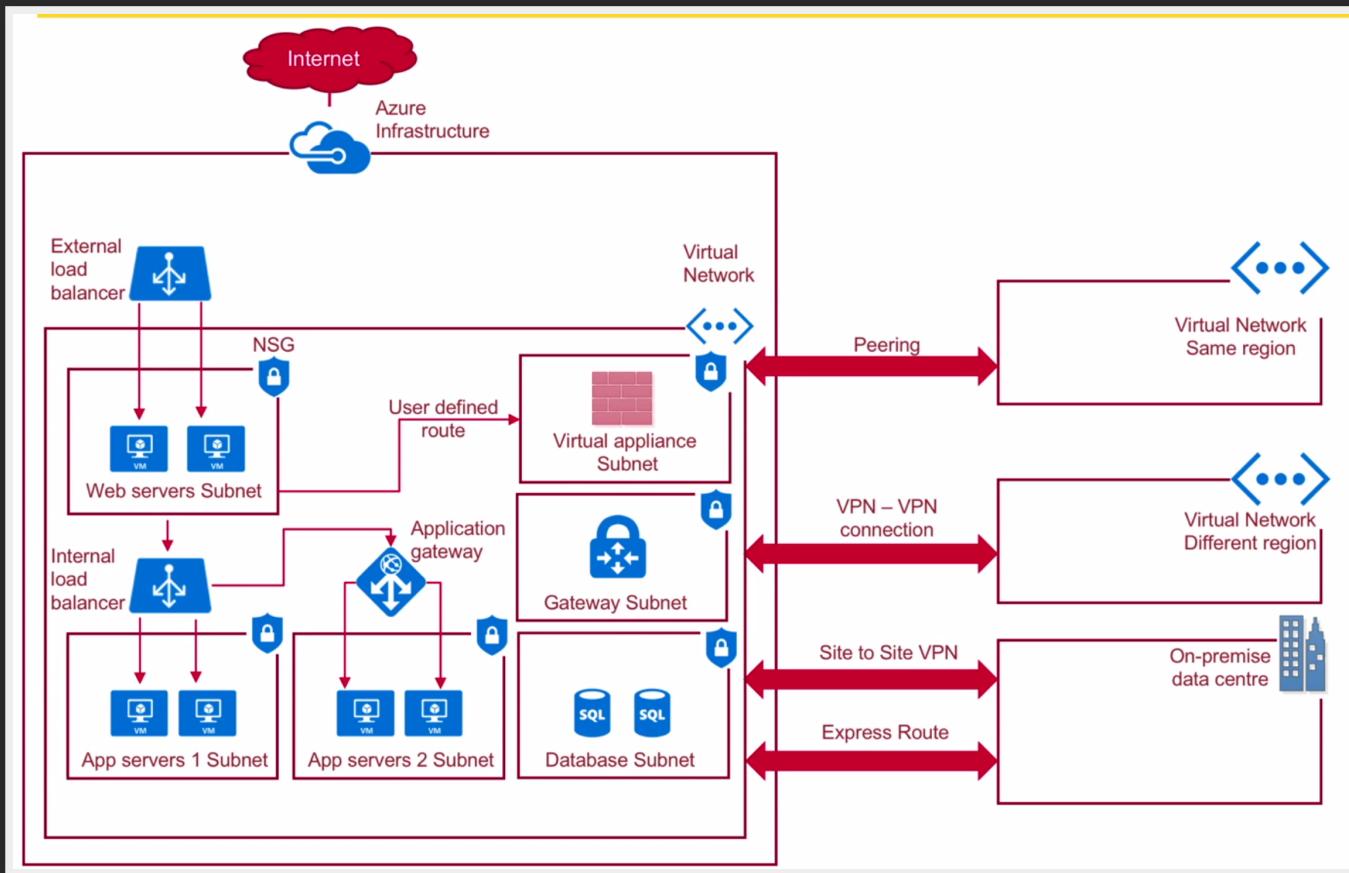
AZURE VNET

Features:

- Isolation & Segmentation
- Outbound/Internet Communication
- Network Security & Routing
- Azure Managed Services Integration
- On-Premises Communication (VPN)
- VNet Peering

Azure VNet Overview

AZURE VNET



AZURE VNET PRICING

- VNet ist kostenfrei
- pro Subscription bis zu 50 VNets über alle Regionen
- Kostenpflichtig: Public IP Adressen, VPN Gateway, API Gateway, VMs, VNet Peering

Azure VNet Pricing

AZURE VNET

Network Security Groups (NSG)

- definiert Netzwerk Routing Regeln für Netzwerk Interfaces / Subnets (Firewall)
- Inbound Regeln und Outbound Regeln
- Regel besteht aus Name, Priorität, Source, Destination, Port, Action

AZURE VNET

NSG

Inbound security rules							
PRIORITY	NAME	PORT	PROTOCOL	SOURCE	DESTINATION	ACTION	...
100	allow_kube_tls	443-443	TCP	Any	Any	✓ Allow	...
101	⚠ allow_ssh	22-22	TCP	Any	Any	✓ Allow	...
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	✓ Allow	...
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalanc...	Any	✓ Allow	...
65500	DenyAllInBound	Any	Any	Any	Any	✗ Deny	...
Outbound security rules							
PRIORITY	NAME	PORT	PROTOCOL	SOURCE	DESTINATION	ACTION	...
65000	AllowVnetOutBound	Any	Any	VirtualNetwork	VirtualNetwork	✓ Allow	...
65001	AllowInternetOutBound	Any	Any	Any	Internet	✓ Allow	...
65500	DenyAllOutBound	Any	Any	Any	Any	✗ Deny	...

VNET SUBNETS

- Was ist Subnetting ?
- Segmentierung des Adressraums in Teilträume
- man verwendet häufig die Classless Inter-Domain Routing (CIDR-) Notation
- Nutzen: Organisation (Sicherheit), Performance, Vereinfachung von Routing

AZURE VNET - PEERING

- VNet Peering
- Wie können 2 VMs innerhalb 2 unterschiedlicher VNets miteinander kommunizieren ? (ohne über das Internet zu gehen)

AZURE VNET - PEERING

- VMs können direkt über ihre private IP miteinander kommunizieren
- Subnetze dürfen sich nicht überlappen
(Eindeutigkeit der privaten IPs)
- Routing geschieht über die Azure Infrastruktur
- local VNet Peering: innerhalb einer Azure Region
- global Vnet Peering: zwischen Azure Regionen

VNet Peering

AZURE VNET - PEERING

- Vorteile:
- Sicherheit - Traffic bleibt im Microsoft Backbone Network
- Performance - Traffic wird nicht indirekt über das Internet geroutet
- Datentransfer zwischen Subscriptions / Azure Services / Regionen werden ermöglicht

AZURE IP ADRESSEN

- public IPs - essentiell für Outbound Traffic
- private IPs - für traffic innerhalb eines VNets

AZURE PUBLIC IP ADRESSEN

- können folgenden Resourcen zugeteilt werden:
- VM Network Interfaces, Internet-Loadbalancers, VPN Gateways, Application Gateways (API Gateway)
- statische public IPs - fest zugeteilt, bspw. nützlich um eigene DNS Namen zu binden
- dynamische public IP Adressen (ändern sich bei Restarts der zugeordneten Resource)
- IPv4 und IPv6 (nur bei Loadbalancer)

VNET DEMO MIT AZURE CONTAINER SERVICES (AKS)

FRAGE ZUM ÜBERBRÜCKEN DES DEPLOYMENTS: WAS SIND DIE UNTERSCHIEDE ZWISCHEN HUB/SWITCH/ROUTER ?

CLOUD NETWORKING - LOADBALANCER

- Was ist ein Loadbalancer ?

LOADBALANCER

- Typen: Hardware, Software Loadbalancer
- L4, L7 Loadbalancer
- Loadbalancer algorithmen: DNS round robin, gewichtetes round robin, least connection, affinity etc.

AZURE LOADBALANCER

- features:
- Loadbalancer algorithmen: source IP-hash basierte Verteilung, affinity basierte Verteilung

BONUSFRAGE

- Was ist das OSI Modell ?

OSI Modell

ZUSAMMENFASSUNG AZURE VNETS

2. CLOUD COMPUTING

F. MODERNE SOFTWARE ENTWICKLUNG



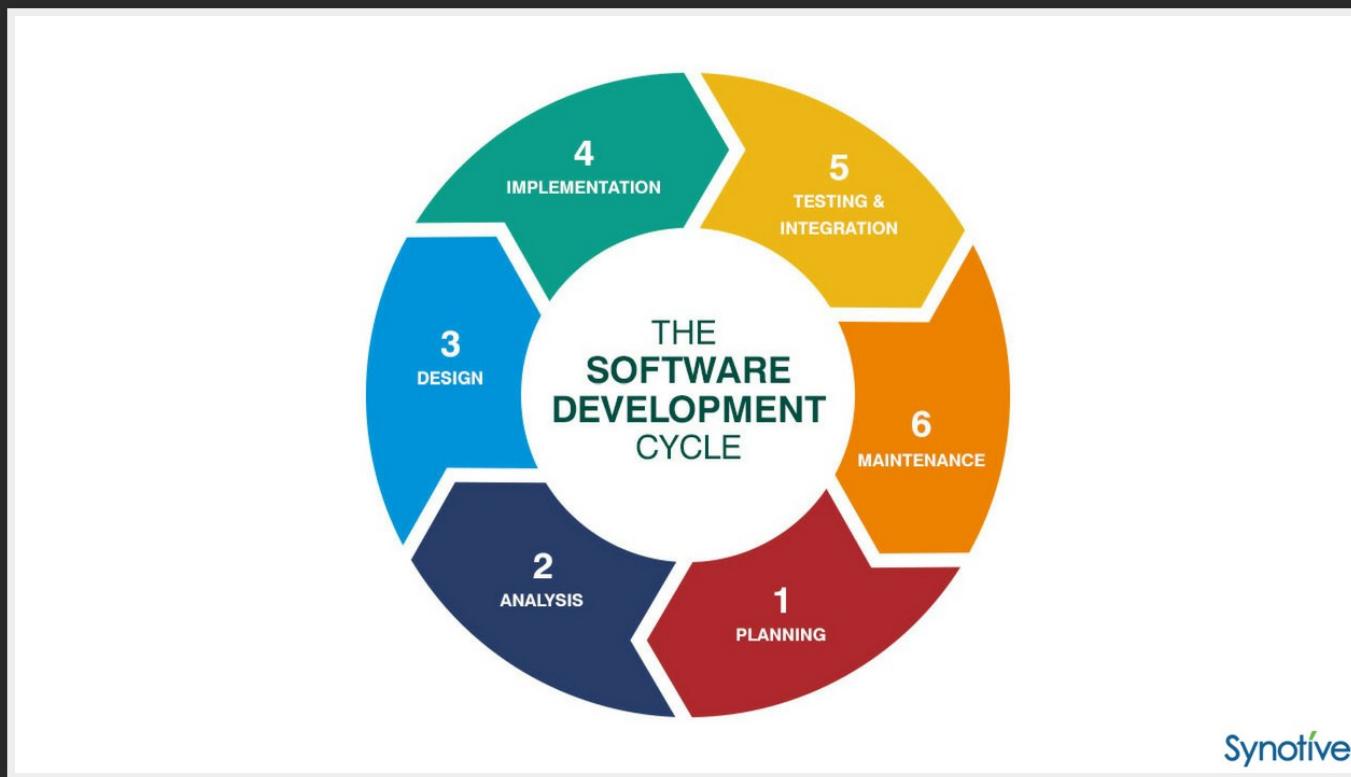
SOFTWARE DEVELOPMENT PROCESS

Software Development Lifecycle



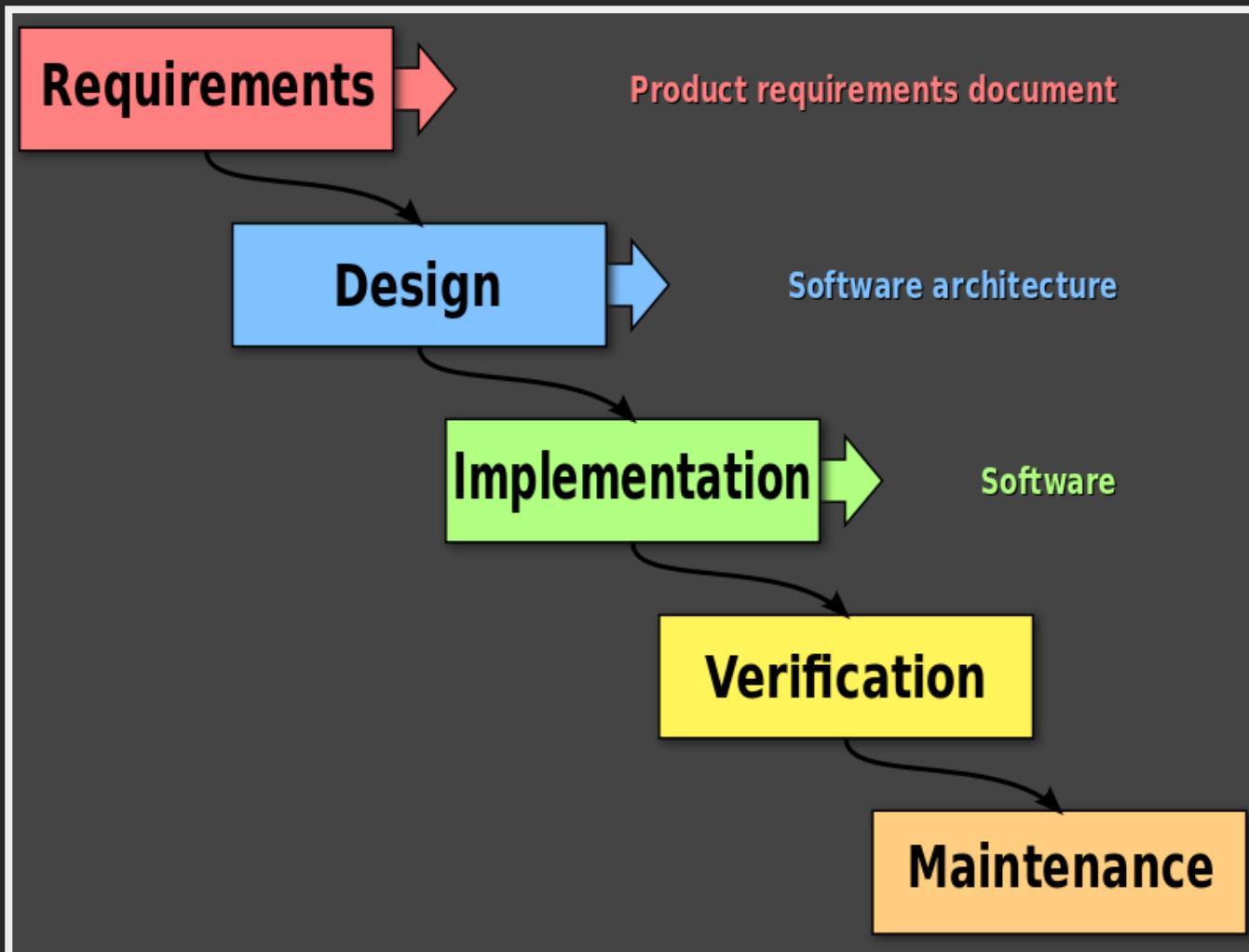
SOFTWARE DEVELOPMENT PROCESS

Software Development Lifecycle



SOFTWARE DEVELOPMENT PROCESS

Waterfall

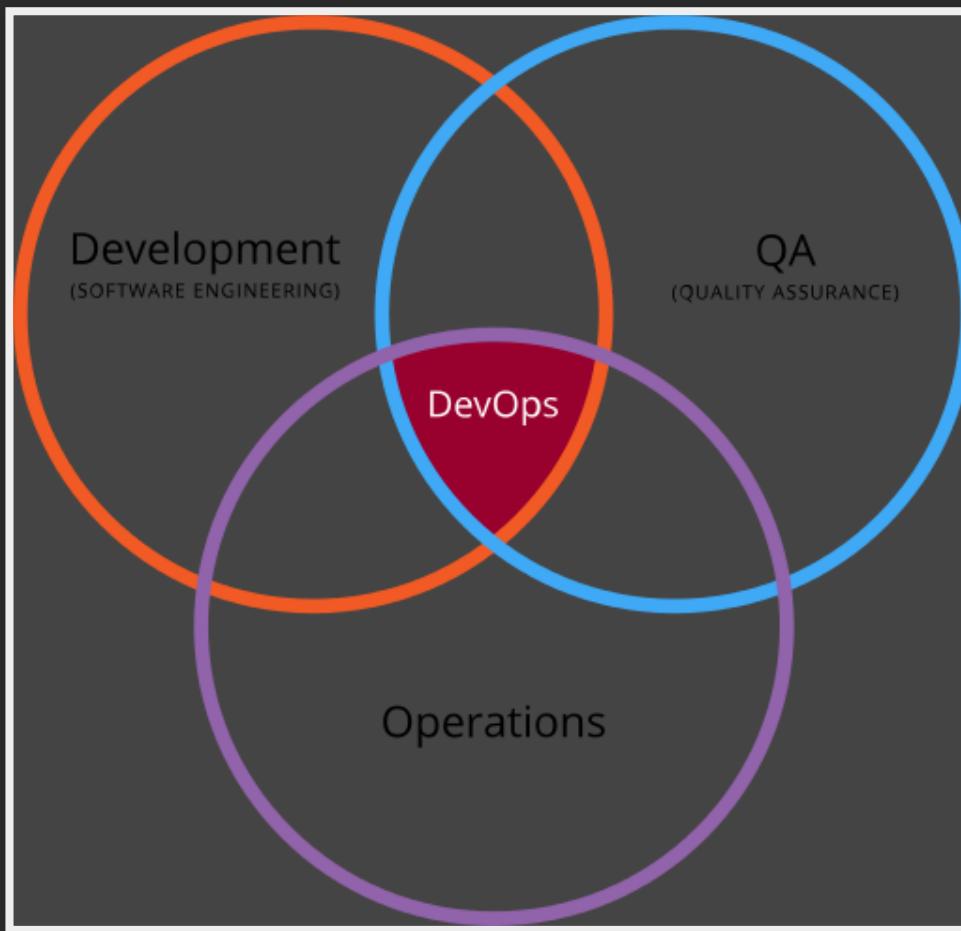


SOFTWARE DEVELOPMENT PROCESS

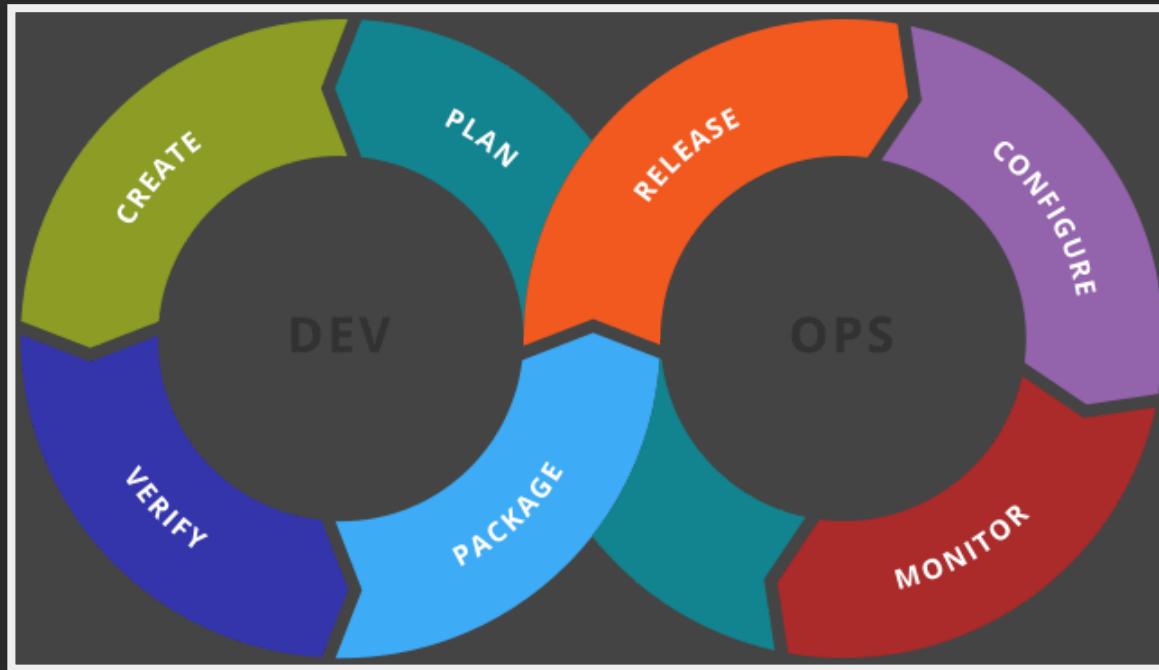
V-Modell

SOFTWARE DEVELOPMENT METHODEN

DevOps



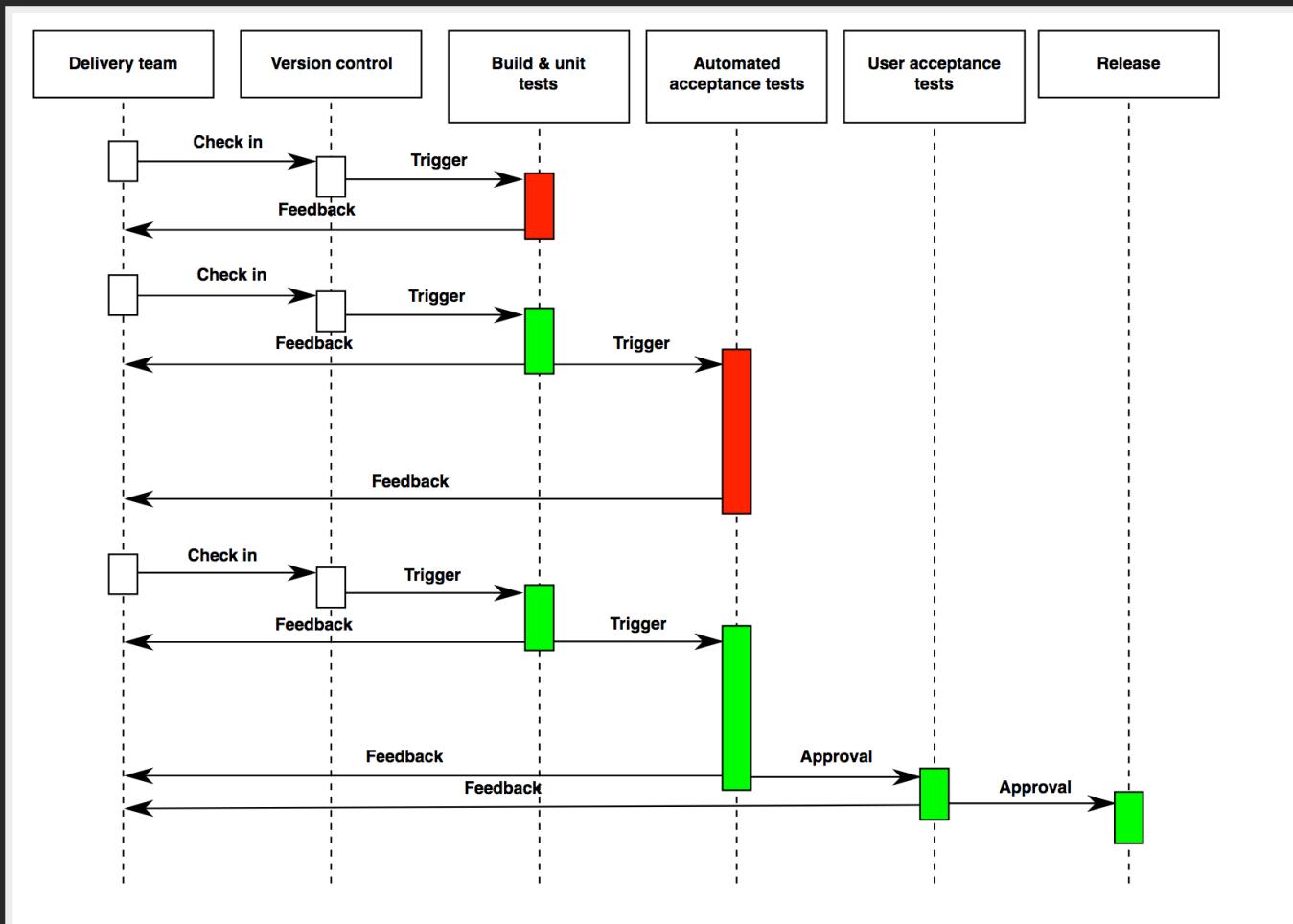
DevOps



- Aufgabe:
- welche Tasks kennt ihr aus der Praxis ?
- mit welchen Tools werden diese Tasks unterstützt ?

SOFTWARE DEVELOPMENT METHODEN

CI/CD



SOFTWARE DEVELOPMENT METHODEN

CI/CD

- Gitlab Übung:
- erstellt ein Projekt auf Gitlab.com
- entwickelt eine Gitlab CI/CD Pipeline
- CI: Docker Push nach Dockerhub
- CD: Deployment nach AKS

SOFTWARE DEVELOPMENT METHODEN

Frage:

- Was ist Infrastructure-as-Code
- Was ist Test Driven Development

MICROSERVICES

- modernes / populäres Architekturmuster in der Cloud
- eignen sich für "App-Containerisierung"
- eignen sich für die horizontale Skalierung in der Cloud
- eignen sich für "Continuous Delivery"

PHILOSOPHIE

- Entkopplung von Funktionalitäten in kleine eigenständig lebende Funktionen
- Dekomposition von Applikationen in abgegrenzte, isolierte Module
- Überschaubarkeit, Einfachheit von Komponenten
- Offene Schnittstellen, keine proprietäre Protokolle, meistens Http + JSON/XML
- Programmiersprachenunabhängigkeit von Komponenten
- Entkopplung von Entwicklungsteams

VORTEILE

- unabhängige Skalierbarkeit von Microservices
- bessere Wartbarkeit, einfache Neuimplementierung
- versteckte Abhängigkeiten werden über die API erkennbar
- Sprachenunabhängig, Teams können nach ihren Wünschen den Technologiestack auswählen
- unabhängige Entwicklungszyklen, Parallelisierung der Entwicklung
- bei Überlastung (DDos) können sekundäre Services zugunsten von primären Services abgeschaltet werden

NACHTEILE

- Performance Overhead durch Netzwerk-Kommunikation
- Erhöhte Komplexität von Tests: Unit/Komponenten-Tests werden Integrationstests
- Erhöhte Komplexität von Monitoring: zentralisiertes Monitoring, Splunk, AppDynamics etc.
- Deployment Prozess muss ggf. zwischen Teams abgestimmt werden (API Breaking Changes)
- Einführung von generellen Problemen von verteilten Anwendungen: z.B. Lastverteilung, Zeitsynchronisation, distributed Locking

<https://www.youtube.com/embed/CKL3fV5UR8w>

12-FACTOR APPS

The Twelve Factors

#	Factor	Description
I	Codebase	There should be exactly one codebase for a deployed service with the codebase being used for many deployments.
II	Dependencies	All dependencies should be declared, with no implicit reliance on system tools or libraries.
III	Config	Configuration that varies between deployments should be stored in the environment.
IV	Backing services	All backing services are treated as attached resources and attached and detached by the execution environment.
V	Build, release, run	The delivery pipeline should strictly consist of build, release, run.
VI	Processes	Applications should be deployed as one or more stateless processes with persisted data stored on a backing service.
VII	Port binding	Self-contained services should make themselves available to other services by specified ports.
VIII	Concurrency	Concurrency is advocated by scaling individual processes.
IX	Disposability	Fast startup and shutdown are advocated for a more robust and resilient system.
X	Dev/Prod parity	All environments should be as similar as possible.
XI	Logs	Applications should produce logs as event streams and leave the execution environment to aggregate.
XII	Admin Processes	Any needed admin tasks should be kept in source control and packaged with the application.

- Wikipedia

DISKUSSION

Was ist der Sinn und Zweck von 12 Factor Apps

- 15 min.

2. CLOUD COMPUTING

G. WORKSHOP SESSION



WORKSHOP SESSION: MICROSERVICE MESH



ZIEL

- Erstellung eines Microservice Backends in der Cloud
- Realisierung eines simplen Connected Mobility Use Cases

WERKZEUGE

- Microsoft Azure
- Kubernetes
- NodeJS

AUFGABE 1

Erstellt für einen simplen Connected Mobility Use Case
das Backend System in der Cloud

Use-Case Beispiel:

- Auto Lokalisationsdienst
- Flottenmanagement (bspw. für Taxi Unternehmen)