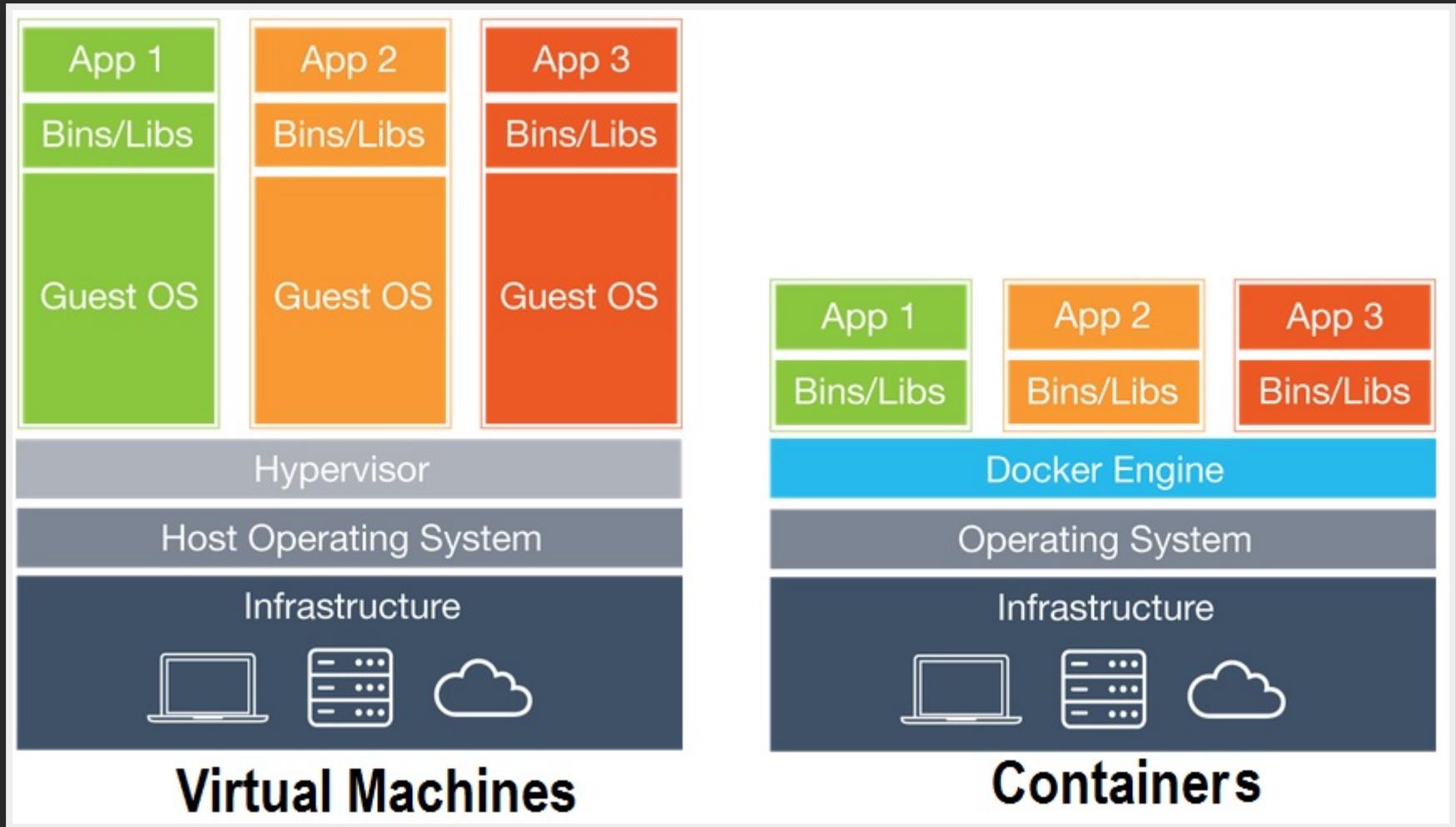


# BETRIEBSSYSTEM-VIRTUALISIERUNG

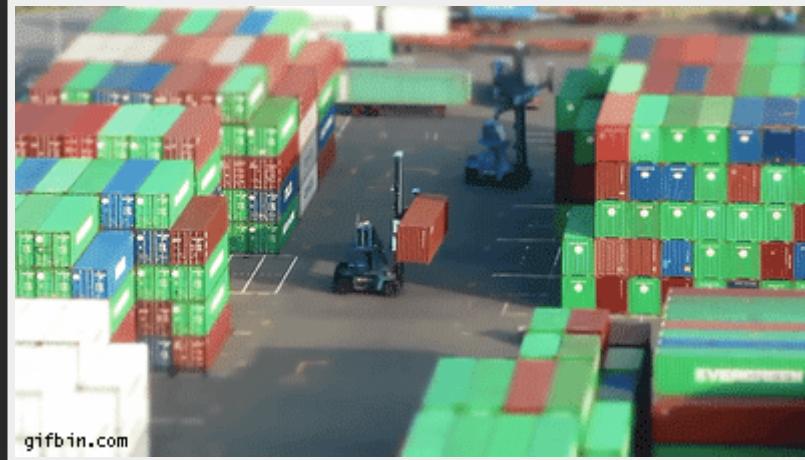
# BETRIEBSSYSTEM-VIRTUALISIERUNG VS SYSTEMVIRTUALISIERUNG



# FRAGE: WIE NENNT MAN FOLGENDER ART VON SOFTWARE?



# CONTAINER ORCHESTRATION



# CONTAINER ORCHESTRIERUNG

- Framework verantwortlich für Container Life-cycle, Scaling, Networking, Loadbalancing, Virtual Storage
- Container kommunizieren über ein privates Netzwerk miteinander
- Kommunikation nach extern über externes Netzwerk
- "Bring-your-own-Container"-Prinzip: die Plattform startet beliebige Container-Images
- Container Images werden in Image-Repositories hochgeladen (public/private)

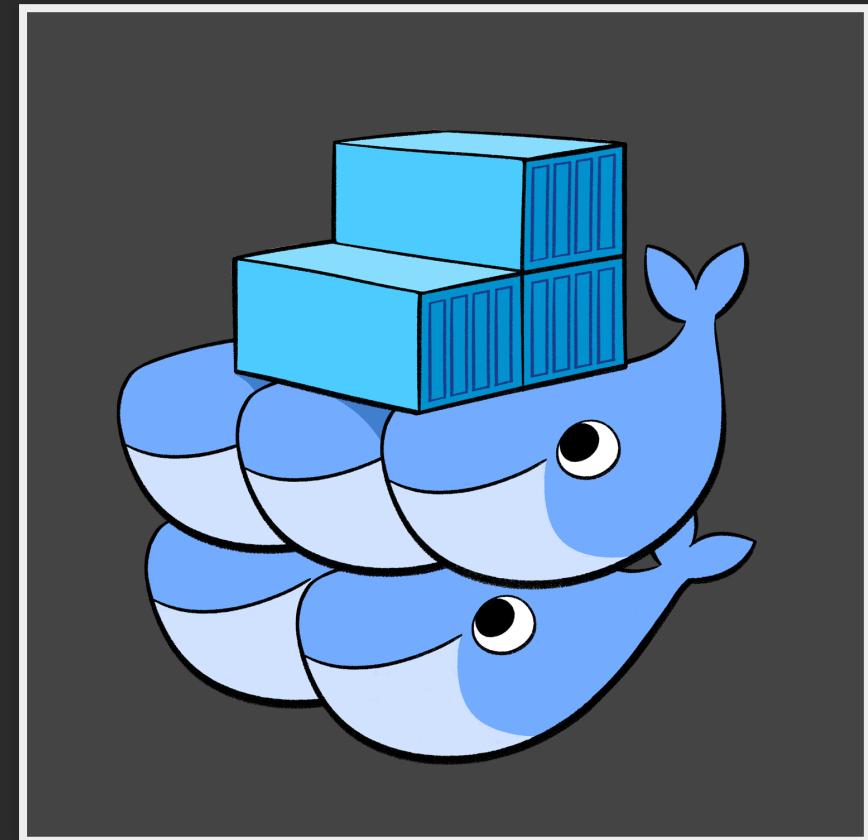
# Bedeutung für Cloud Computing

- Basis für Container-as-a-Service (CaaS)
  - flexibler als PaaS, einfacher als IaaS
  - CaaS als Alternative zur PaaS und IaaS
  - DevOps-Orientiert
- Kein Vendor Lock-in
  - Kubernetes ist open-source, integrierbar in private / public cloud

Beispiel:

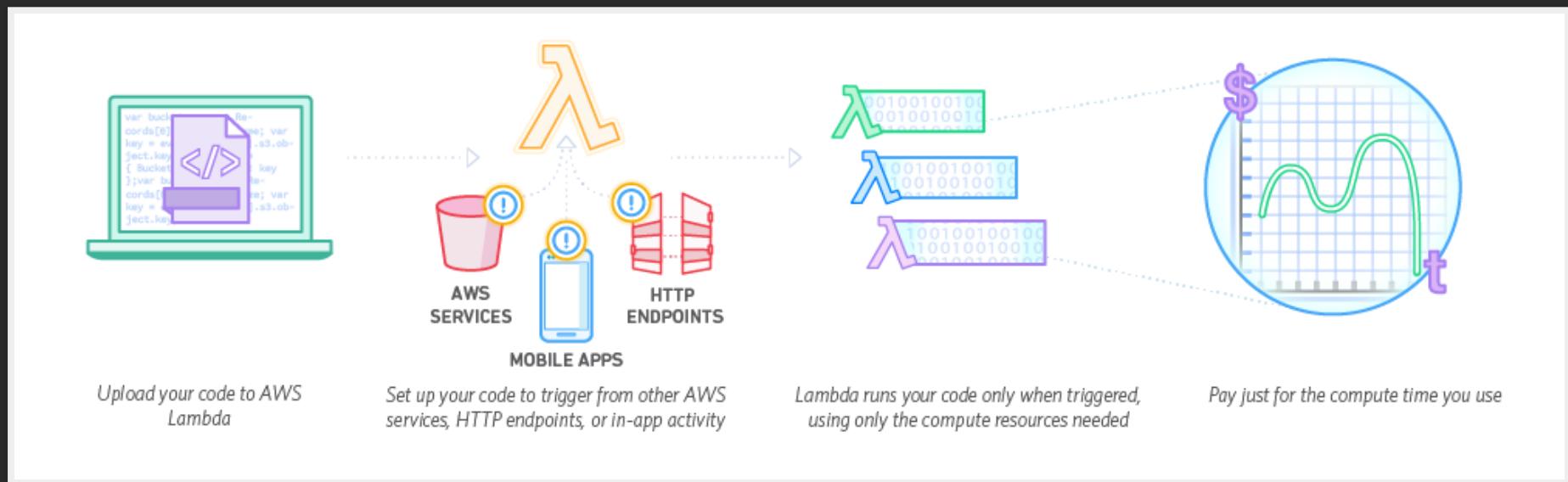


Kubernetes



Docker Swarm

# SERVERLESS



- Serverless != Ohne Server
- kurzlebige Funktionen (stateless)
- In Serverless-Funktionen können On-Demand Rechenoperationen implementiert werden, die nicht ständig laufen müssen
- Lambda-Funktionen werden in der Cloud On-Demand erstellt, ausgeführt und beendet
- Die Kosten werden im Pay-per-Use Prinzip in Millisekunden abgerechnet

# Bedeutung für Cloud Computing

- Rechenkosten können genauer abgerechnet werden (Pay-per-Use auf Millisekunden - Basis)
- Rechenkosten entstehen wirklich nur bei Bedarf
- Leichtgewichtige Funktionen können schnell entwickelt und in den Betrieb überführt werden
- Management und Betrieb von Funktionen liegen in der Verantwortung des Cloud-Providers

Beispiele:

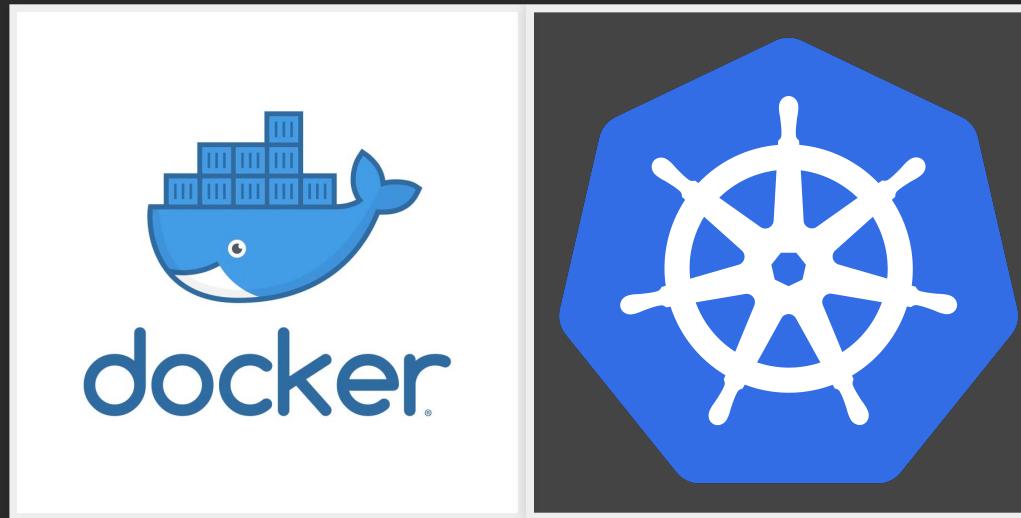
AWS Lambda, Apache OpenWhisk, Google Cloud Functions



# 2. CLOUD COMPUTING

## C. WORKSHOP SESSION

# WORKSHOP SESSION: DOCKER & KUBERNETES



<https://www.docker.com>

<https://kubernetes.io/>

# ZIEL

- Erstellung eigener Docker Container
- Betrieb von Docker Container in der Cloud

# WERKZEUGE

- Docker (docker)
- Azure CLI (az)
- Kubernetes CLI (kubectl)

## AUFGABE 1

Erstellt ein Docker Image für die Hello-World App:

- `docker run --name hello world -p 80:80 -it hello-world-app`

## AUFGABE 2

Betreibt die Hello-World App im MS Azure AKS Cluster:

- die App soll über das Internet zugreifbar sein
- RESTful API soll 'CarData' in MongoDB abspeichern

templates:

`deployment.yml`

`service.yml`

## AUFGABE 3

Betreibt die Hello-World App als MS Azure Function:

- die App soll über das Internet zugreifbar sein
- alle bisherigen RESTful API functions sollen implementiert werden





# 2. CLOUD COMPUTING

## D. SKALIERUNG

# AUTO-SCALING

- Essentieller Mechanismus im Cloud Computing um auf den Bedarf von Ressourcen zu reagieren
- dynamische Bereitstellung von VMs, Container, Speicher, Storage
- bidirektional: Scale-Up, Scale-Down
- Nutzen:
  - Elastizität: bessere Auslastung, Kostenoptimierung (Kosteneinsparung)
  - Verfügbarkeit und Zuverlässigkeit: durch Sicherstellung von min. Instanzen

## AUTOSCALING STRATEGIEN

- Reaktiv: Cloud Umgebung beobachtet Kernmetriken (CPU, Memory, Traffic) und reagiert auf Last
- Proaktiv:
  - Scheduling (scheduled scaling): Zeitlich geplante Skalierung auf Basis von Erfahrungswerten
  - Prädiktiv (predictive scaling): durch Vorhersage auf Basis von prädiktiver Analyse

## AUTO-SCALING BEI AWS (IAAS)

- CPU, Speicherauslastung, Netzwerk Traffic wird durch AWS EC2 überwacht
- alle Logs/Stdout werden an AWS Cloudwatch übermittelt (Logging/Monitoring-Service)

<http://docs.aws.amazon.com/autoscaling/latest/userguide/WhatIsAutoScaling.html>

- AMIs/VMs können zu Auto-Scaling Groups hinzugefügt werden, Konfiguration von Scaling-Plans:
  - Scaling Policies: reaktiv (standard), scheduled-scaling, dynamisch (auf Basis von Metriken aus Amazon SQS, Cloudwatch alarms)
  - min./max. und gewünschte Anzahl an Instanzen

<https://docs.aws.amazon.com/autoscaling/latest/userguide/as-using-sqs-queue.html>

## AUTO-SCALING BEI AWS (IAAS)

- Ausführung: auf Basis von Scaling Plans werden AMI Instanzen hinzugefügt / terminiert
- hinzufügen von Instanzen (auf Basis von Launch-Configuration):
  - Instanziierung aus AMI
  - boot-up Phase, Initialisierung, Konfigurierung per Skript (siehe AMI Design Strategien)
  - Anbindung an Cloud-Storage
  - Registrierung ins Virtual Private Cloud (Netzwerk)
  - Registrierung beim Elastic Load-Balancer

## AUTO-SCALING BEI AWS (IAAS)

- Status Überwachung: Health-Checking
  - über HTTP-Endpunkt: z.B. /heathcheck
  - healthy flag

## AUTO-SCALING BEI KUBERNETES (CAAS)

- Monitoring / Überwachung von Basis-Metriken: CPU, Memory-Auslastung
- Skalierung auf Pod-Ebene (Kubernetes Konzept: Zusammenfassung von Containern)
- Skalierung über die Definition einer Auto-Scaling Konfiguration
- Konfiguration wird über Selektoren auf Kubernetes Bausteine (Pod, ReplicaSets, Deployments) angewendet
- Auto-Scaling ist reaktiv

## AUTO-SCALING BEI KUBERNETES (CAAS)

- Ausführung: auf Basis der Auto-Scaler Konfiguration werden Container-Instanzen hinzugefügt / terminiert
- hinzufügen von Instanzen:
  - Pod Instanz wird erstellt, alle Container Definitionen werden aus dem konfiguriertem Image instanziert
  - Einbinden der Pod/Container Instanz ins private virtual network
  - Container Prozess wird gestartet (run-Befehl)
  - Registrierung beim Service und Load-Balancer

## AUTO-SCALING BEI KUBERNETES (CAAS)

- Status Überwachung: Health-Checking
  - über HTTP-Endpunkt: z.B. /heathcheck, (http-response-Code: 200 bedeutet "OK", ansonsten "NOK")
  - Überwachung der Pod/Container Prozesse: terminiert der Pod/Container Prozess, stoppt der Pod/Container, es wird ein neuer Pod/Container gestartet

# 2. CLOUD COMPUTING

## E. CLOUD COMPUTING NETWORKING



# **RECAP: NAT (LETZTE VORLESUNG)**

Frage: Wie werden Computer innerhalb eines Heimnetzwerkes vor Zugriffen von außerhalb durch NAT "geschützt" ?

# NETWORK ADDRESS TRANSLATION (NAT)

# VIRTUELLE NETZWERKE (AM BEISPIEL AZURE VNET)

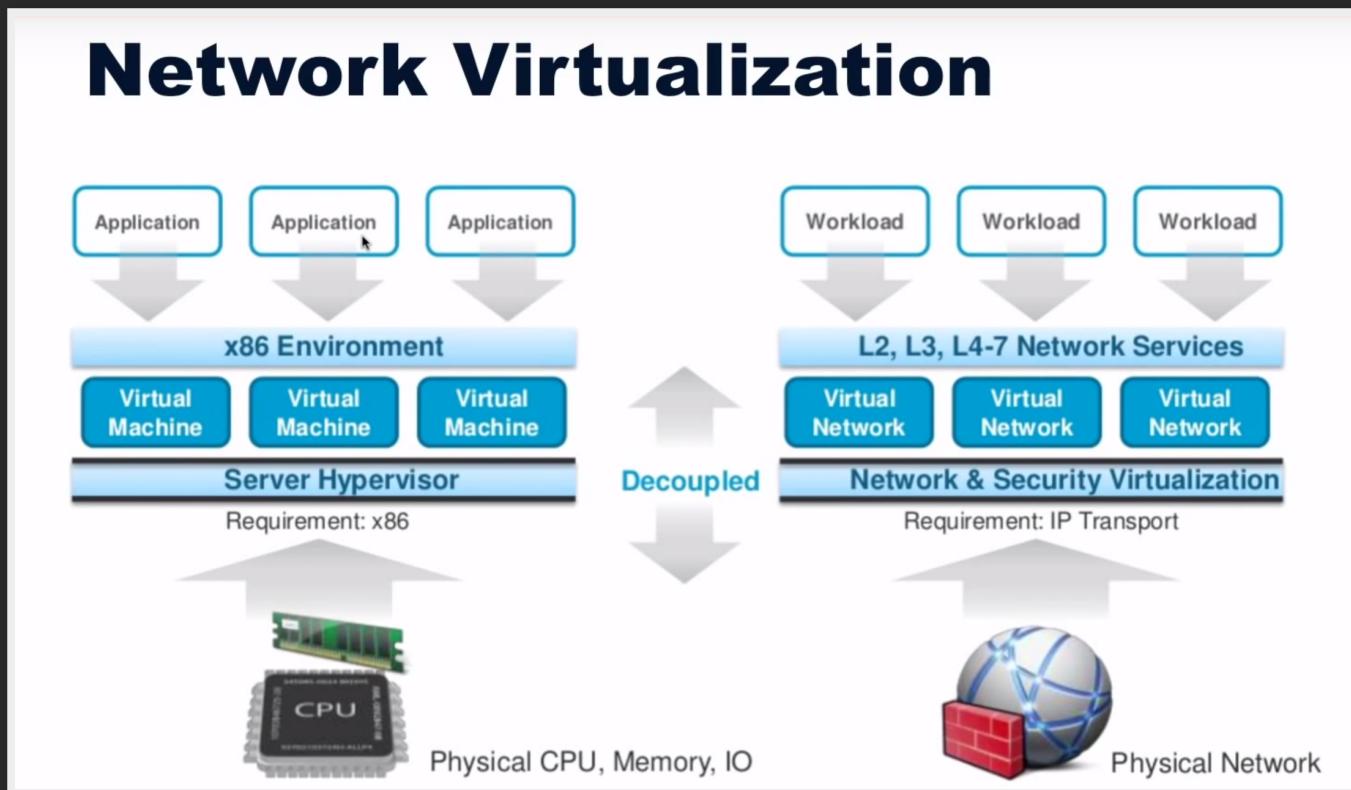
- Was ist ein Virtuelles Netzwerk ?
- Was ist Netzwerkvirtualisierung

# VIRTUELLE NETZWERKE

- Woraus besteht ein Computer Netzwerk ?
- Firewall
- Router, Switch
- Loadbalancer
- Network Interface Cards
- etc.

# NETZWERKVIRTUALISIERUNG

## Network Virtualization



# NETZWERKVIRTUALISIERUNG

- Abstrahierung von Netzwerk-Resourcen von Hardware-Komponenten
- Netzwerkkomponenten und Netzwerkfunktionen werden in Software repliziert
- die physikalischen Resourcen / Komponenten dienen zum Daten Transport
- die virtualisierten Komponenten steuern und definieren die Transportwege

# NFV & SDN

- Techniken: Network Function Virtualization
- Entkopplung von Hardware-Resourcen von Funktionen
- Ziel: Optimierung/Effizienzsteigerung von Netzwerkfunktionen durch bessere Auslastung der Infrastruktur
- Network Functions: Firewall, Loadbalancing (L4-L7)

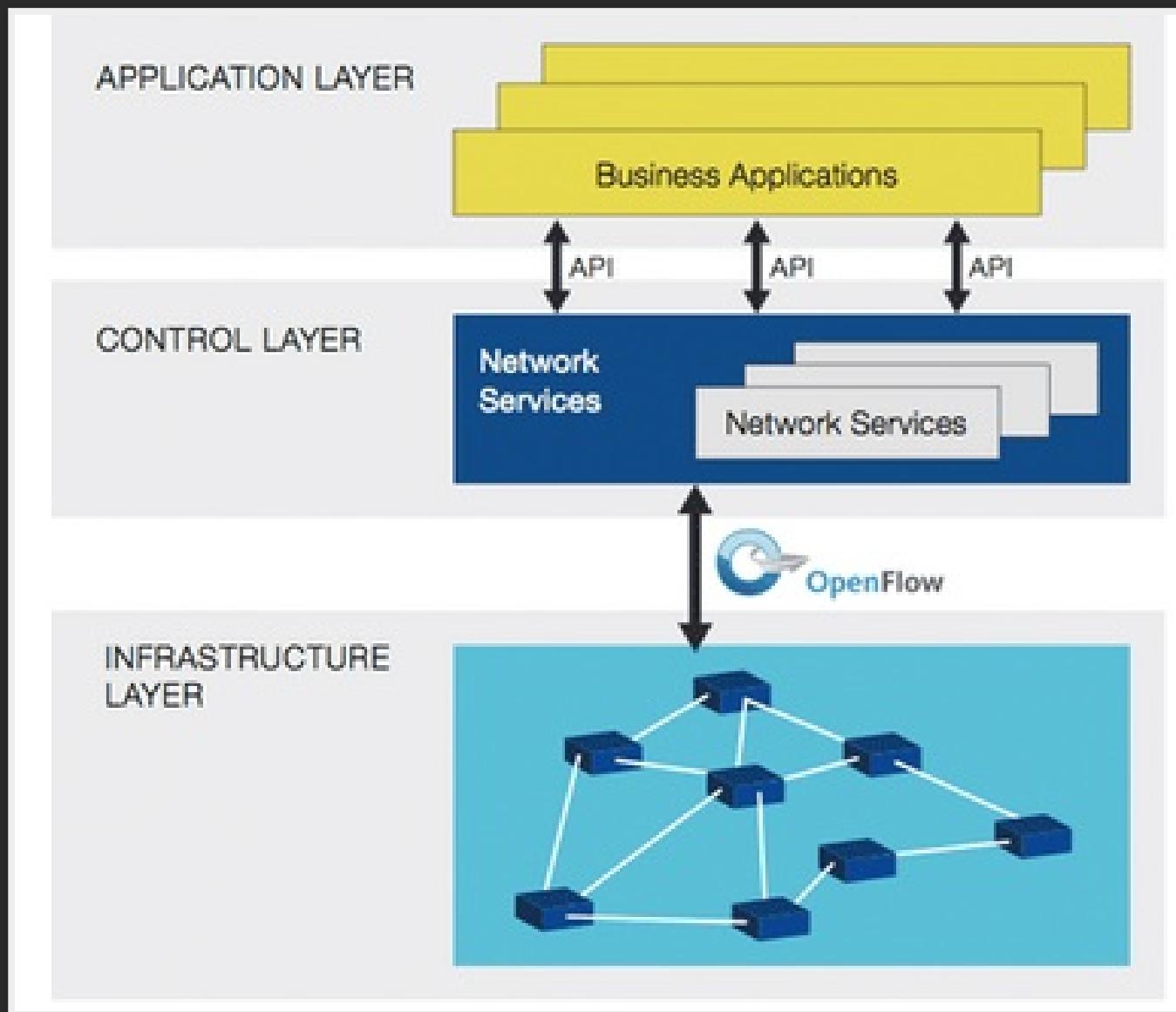
# NETWORK FUNCTION VIRTUALIZATION

# SDN

- Techniken: Software Defined Networking
- komplementär zu Network Function Virtualization
- dient dem Netzwerkmanagement: zentralisiert die Steuerung von Switches und Routern über APIs
- Einsatz: Cloud Computing zum steuern und bereitstellen von virtuellen Netzwerken
- Entkopplung von "Data-plane" und "Control-plane"

# SDN

# SDN



# VIRTUELLES NETZWERK

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