LION[5]

Introduction to the Cloud for IT Managers

Outline

1. **What is the Cloud?**
2. Cloud Service Models
3. Why move / move not into the Cloud?
4. Software Development for the Cloud

Cloud Computing Data Centers

Src: Microsoft

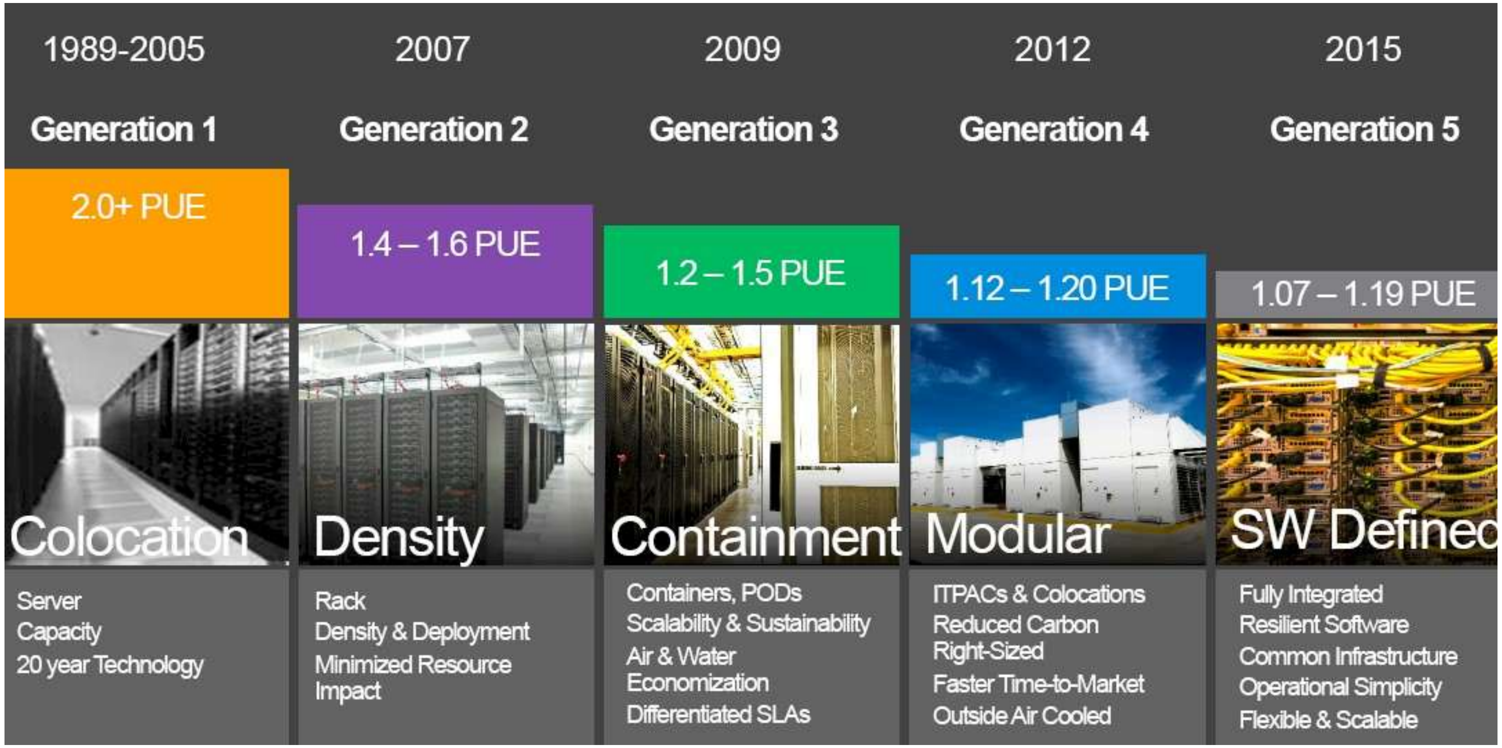
Cloud Computing Data Centers

- Cloud business is **economies of scale** business
- Big **data centers** usually housing 50,000 to 100,000 servers
- Where and when to build data centers
  - Power Usage Effectiveness (**PUE**)
  - Legal constraints, e.g., data privacy
  - Technical factors, e.g., latency
  - Predictions on hardware cost
  - Marketing factors: *“Green data centers are run in Finland / Canada”*

Is that true?

Cloud Computing Data Centers

Microsoft Data Center Evolution



1989-2005	2007	2009	2012	2015
Generation 1	**Generation 2**	**Generation 3**	**Generation 4**	**Generation 5**
2.0+ PUE	1.4 – 1.6 PUE	1.2 – 1.5 PUE	1.12 – 1.20 PUE	1.07 – 1.19 PUE
Colocation	Density	Containment	Modular	SW Definec
Server Capacity 20 year Technology	Rack Density & Deployment Minimized Resource Impact	Containers, PODs Scalability & Sustainability Air & Water Economization Differentiated SLAs	ITPACs & Colocations Reduced Carbon Right-Sized Faster Time-to-Market Outside Air Cooled	Fully Integrated Resilient Software Common Infrastructure Operational Simplicity Flexible & Scalable

Src: Microsoft

Structure of „the Cloud“

- Concept of Cloud Provider

contains 1..n

- **Region**
  Geographic location of data centers. Determines available Services, Resources, and Costs.

contains 1..n

- ***Availability Zone (AZ)***
  one or more isolated and independent data centers in a region with **low latency** to other AZs in same region. Contains computing resources

contains 0..n

- Infrastructure used by Client

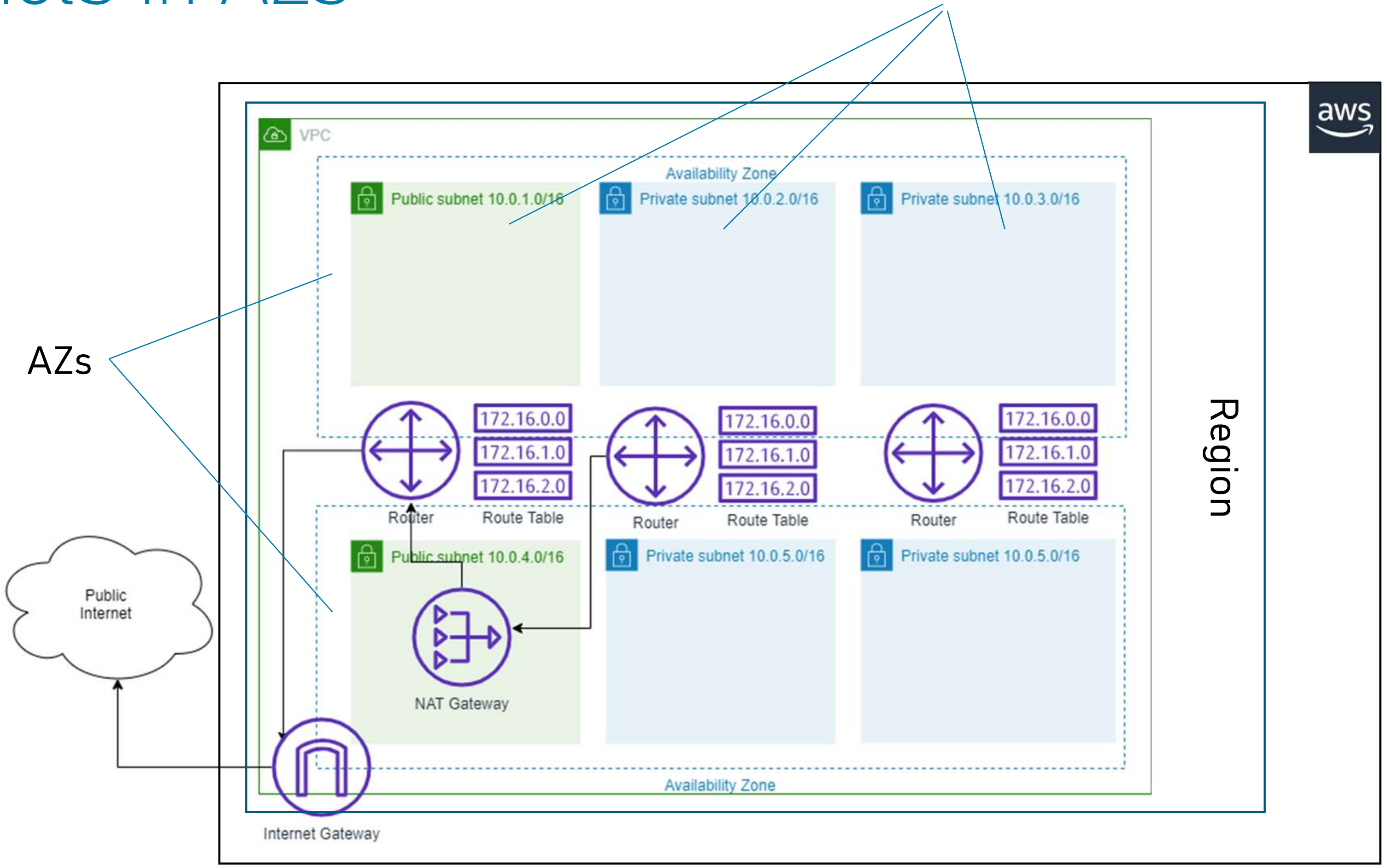
- **Virtual Private Cloud (VPC)** / Network
  Range of (private) Network-Addresses for Cloud Ressources

contains 1..n

- ***Subnet*** Range of Network-Addresses contained in VPC and routed to/within data centers in AZ for computing resources

VPC in Region with Subnets in AZs

Subnets in AZ



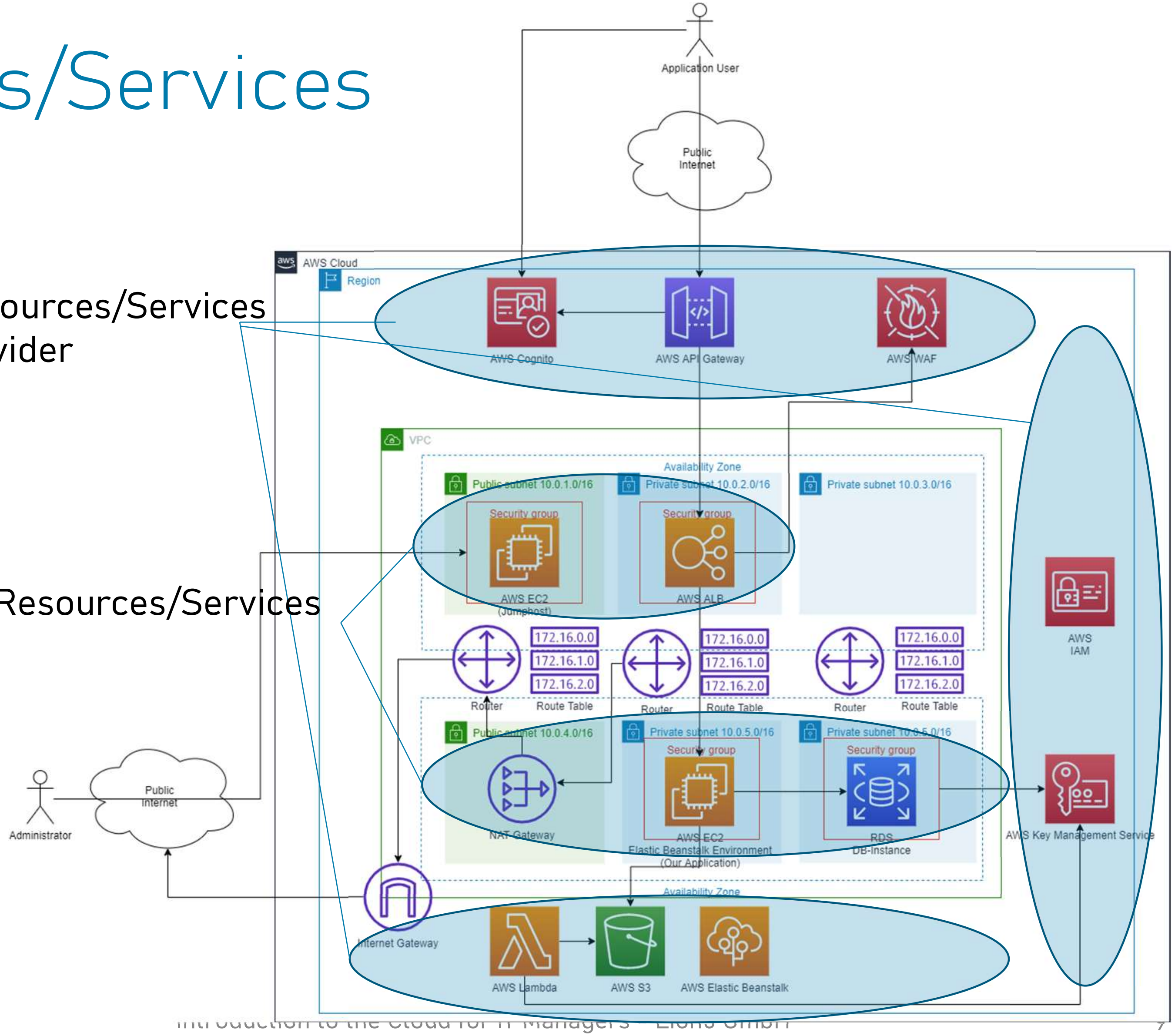
AZs

Region

Resources/Services

Regional Resources/Services of Cloud Provider

Computation Resources/Services in AZs



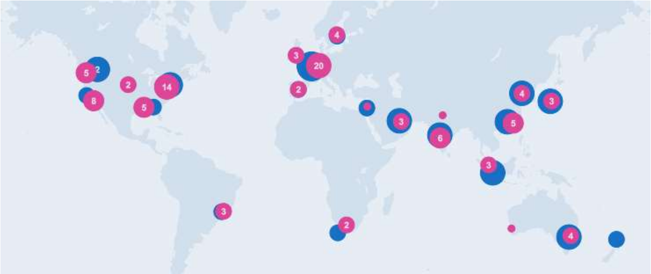
Considerations for Regions and AZs

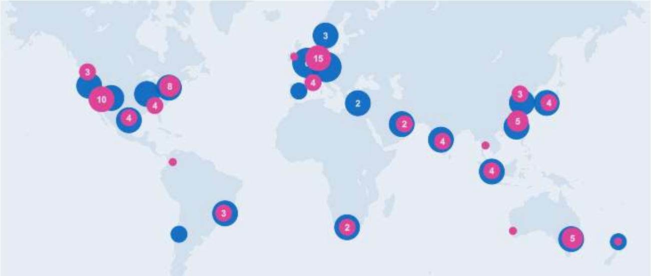
- Availability
  - Use several (independent) AZs to increase availability (read SLAs, availability defined per AZ)
  - Very high availability requirements may require Applications to run in multiple Regions
  - For regional resources of Cloud Provider read SLAs carefully
- Geographic Locations / Regions
  - Latency (bring service close to customer)
  - Costs (use region with low prices)
  - Regulations (e.g. GDPR)
- Elasticity/Scalability
  - Design VPC/Subnets and Region/AZs to allow adaptation to demand

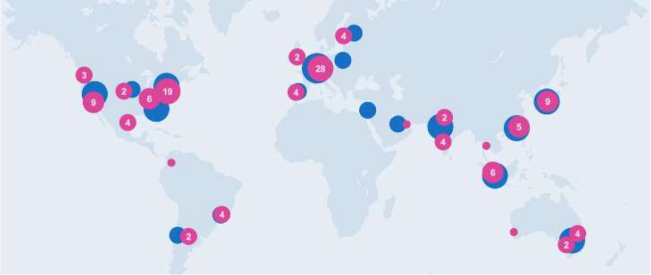
Regions and AZs of diffferent Providers

Src: https://www.cloudinfrastructuremap.com/#/cloud-service-provider/microsoft-azure

- Amazon Web Service
  - 25 geographic regions
  - 81 availability zones
  - 218+ edge locations
  - 12 Regional Edge Caches
- Microsoft Azure
  - 60+ regions with
  - minimum of three availability zones in each region
  - 116 edge locations (Points of Presence)
- Google Cloud Platform
  - 27 cloud regions
  - 82 zones
  - 146 edge locations







Demo

Platform Continuum – Is the Cloud New?

Src: Krishna Kumar, Azure Master Deck

On-Premises Servers

Hosted Servers

Cloud Platform

- Bring your own machines, connectivity, software, etc.
- Complete control
- Complete responsibility
- Static capabilities
- Upfront capital costs for the infrastructure

- Renting machines, connectivity, software
- Less control
- Fewer responsibilities
- Lower capital costs
- More flexible
- Pay for fixed capacity, even if idle

- Shared, multi-tenant infrastructure
- Virtualized & dynamic
- Scalable & available
- Abstracted from the infrastructure
- Higher-level services
- Pay as you go

Excercise 1 – Describe a Cloud Service

- Take two minutes to think about Cloud Usage in your department / business unit
- From your daily work
  - find a cloud service example
  - describe why you think this is a „cloud“ service
  - describe why you think it is not a classically hosted service
- Propose and discuss within the group

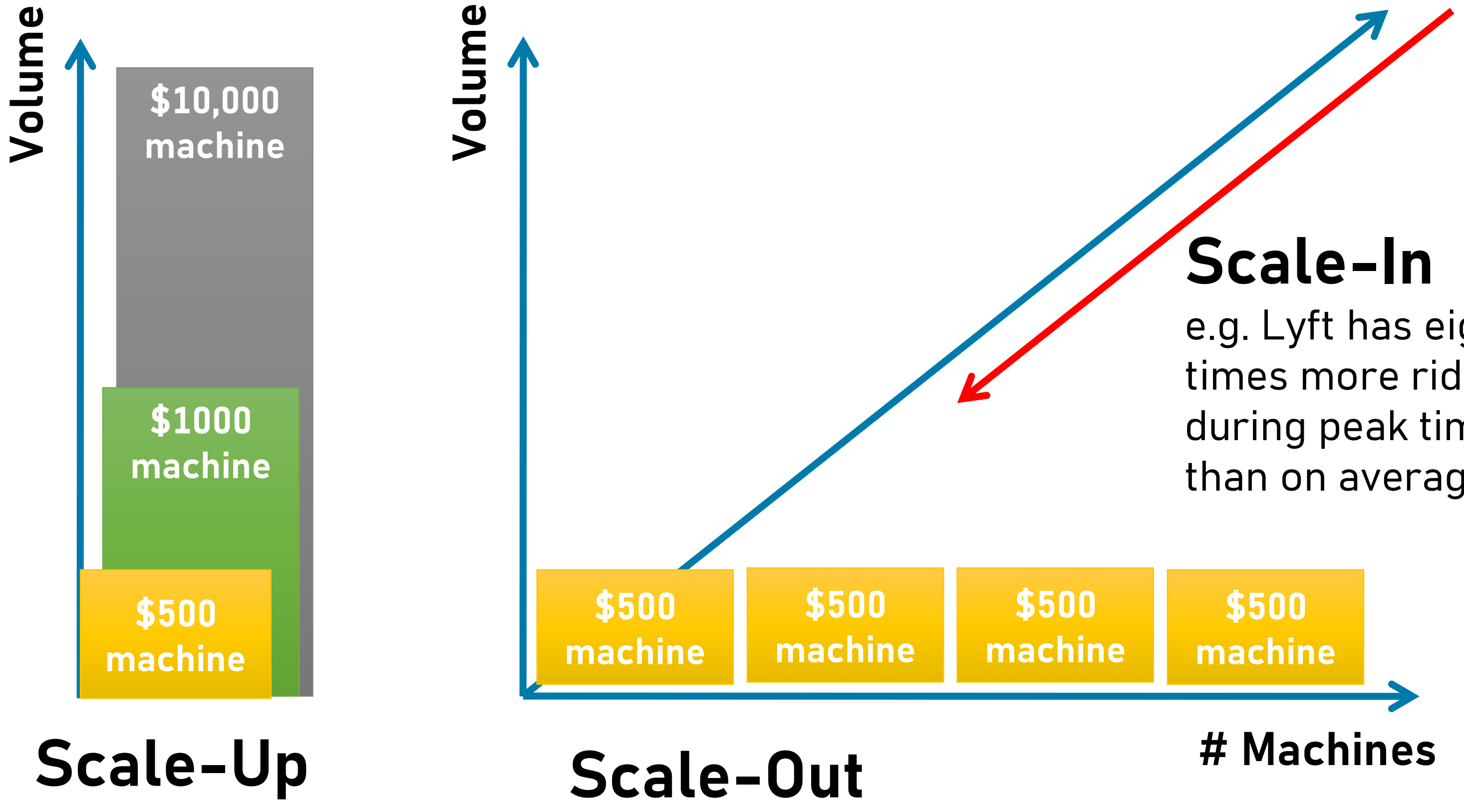
Cloud Characteristics

Cloud Characteristics: On-Demand

- Self-service
- Provisioning of computing resources and cloud services
  - Web user interface
  - API
  - Dynamic/automatic provisioning of resources
    - Scaling
    - Elasticity
- Generally real-time or near real-time availability of demanded resources
- Usually no human involvement on provider side

Cloud Characteristics: Scaling

Scale-out/Scale-in instead of Scale-up



e.g. Lyft has eight times more riders during peak times than on average

Src: Krishna Kumar, Azure Master Deck

Cloud Characteristics: Pay-as-you-Go

- Related to the on-demand provisioning of resources
- Key idea: Only pay for the resources that you really use
- Various manifestations of pay-as-you-go depending on service level
    - Compounded pricing
        - Machine based price (includes vCPUs)
        - Consumed storage capacity
        - Network traffic
    - Partially real pay-per-use
        - Pay for CPU time, 200 milliseconds billing unit
    - Extra cost for special cloud technologies like content delivery and AI services

Cloud Characteristics: Cloud-Specific Technologies

- Services for trending technologies
  - Machine Learning/AI (e.g. Natural Language Processing, Image Recognition, Sentiment Analysis, …)
  - Augmented Reality, Virtual Reality
  - Internet of Things (IoT)
  - Data (Blockchain, NoSQL databases)
- Vendor-Defined Services
  - Vendor-defined typically means vendor lock-in
  - Evolving cloud product portfolio
    - Discontinuation of services, APIs and libraries possible
    - Deprecation and removal of formerly supported SDK versions

  → The customer has to adapt (quickly)
  - Vendor-specific restrictions

Excercise 2 – Check your Cloud Service

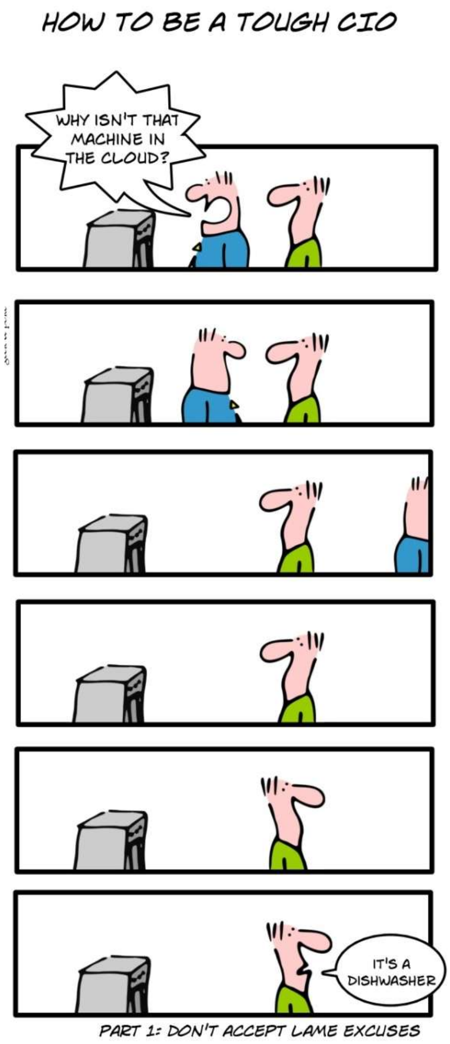
- Take again 2 minutes to apply the last learnings
- Revisit the example „cloud“ service of exercise 1
  (or choose a different example)
- Propose in how far your example „cloud“ service meets the defined cloud characteristics

Outline

1. What is the Cloud?
2. **Cloud Service Models**
3. Why move / move not into the Cloud?
4. Software Development for the Cloud

Cloud Provider Tasks

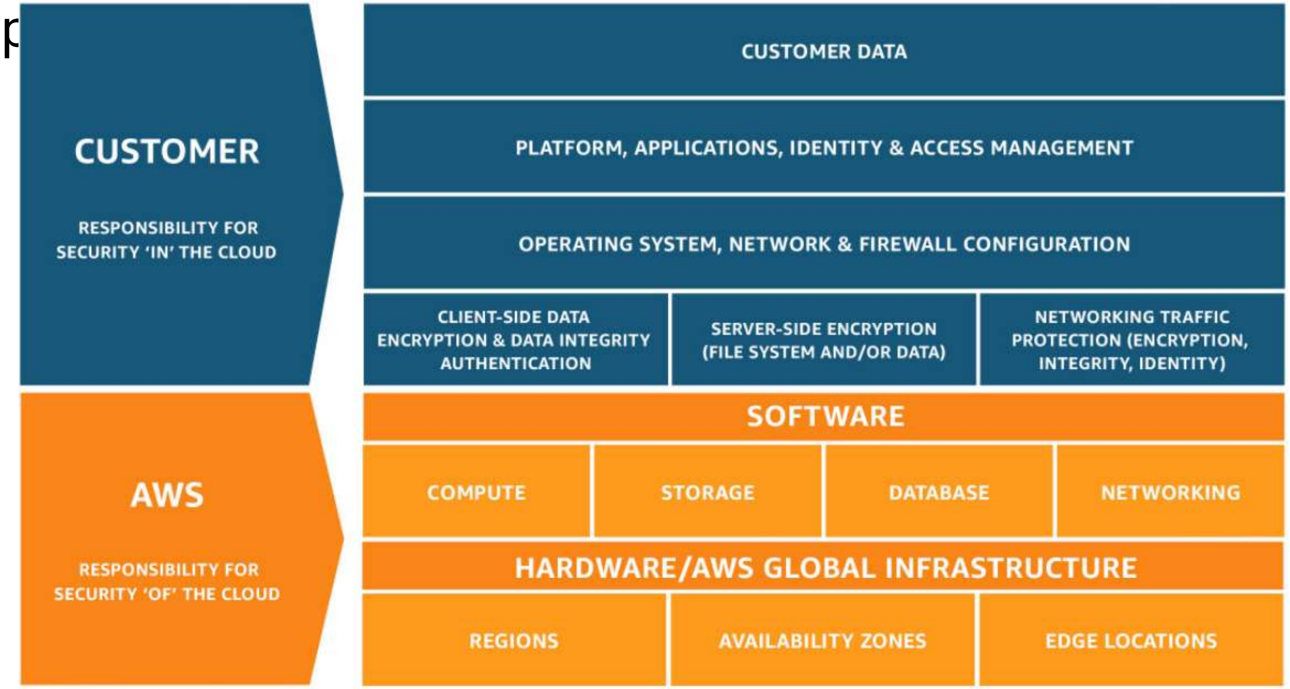
- Service Management
  - Capacity and Change Management
  - Customer Support
  - Incident Management
- Software Lifecycle Management
  - Updates and Hotfixes
  - Monitoring and Incident Management
  - Monitoring and System Provisioning
- Infrastructure Management
  - Hardware Monitoring and Management
  - OS Configuration and Virtualization


HOW TO BE A TOUGH CIO
WHY ISN'T THAT MACHINE IN THE CLOUD?
IT'S A DISHWASHER
PART 1: DON'T ACCEPT LAME EXCUSES

http://geekandpoke.typepad.com/

AWS Shared Responsibility Model

- Security ’of’ vs security ‘in’ the cloud
  - Varies according to **service model** of used cloud p



CUSTOMER

RESPONSIBILITY FOR SECURITY ‘IN’ THE CLOUD

CUSTOMER DATA

PLATFORM, APPLICATIONS, IDENTITY & ACCESS MANAGEMENT

OPERATING SYSTEM, NETWORK & FIREWALL CONFIGURATION

CLIENT-SIDE DATA ENCRYPTION & DATA INTEGRITY AUTHENTICATION

SERVER-SIDE ENCRYPTION (FILE SYSTEM AND/OR DATA)

NETWORKING TRAFFIC PROTECTION (ENCRYPTION, INTEGRITY, IDENTITY)

AWS

RESPONSIBILITY FOR SECURITY ‘OF’ THE CLOUD

SOFTWARE

COMPUTE

STORAGE

DATABASE

NETWORKING

HARDWARE/AWS GLOBAL INFRASTRUCTURE

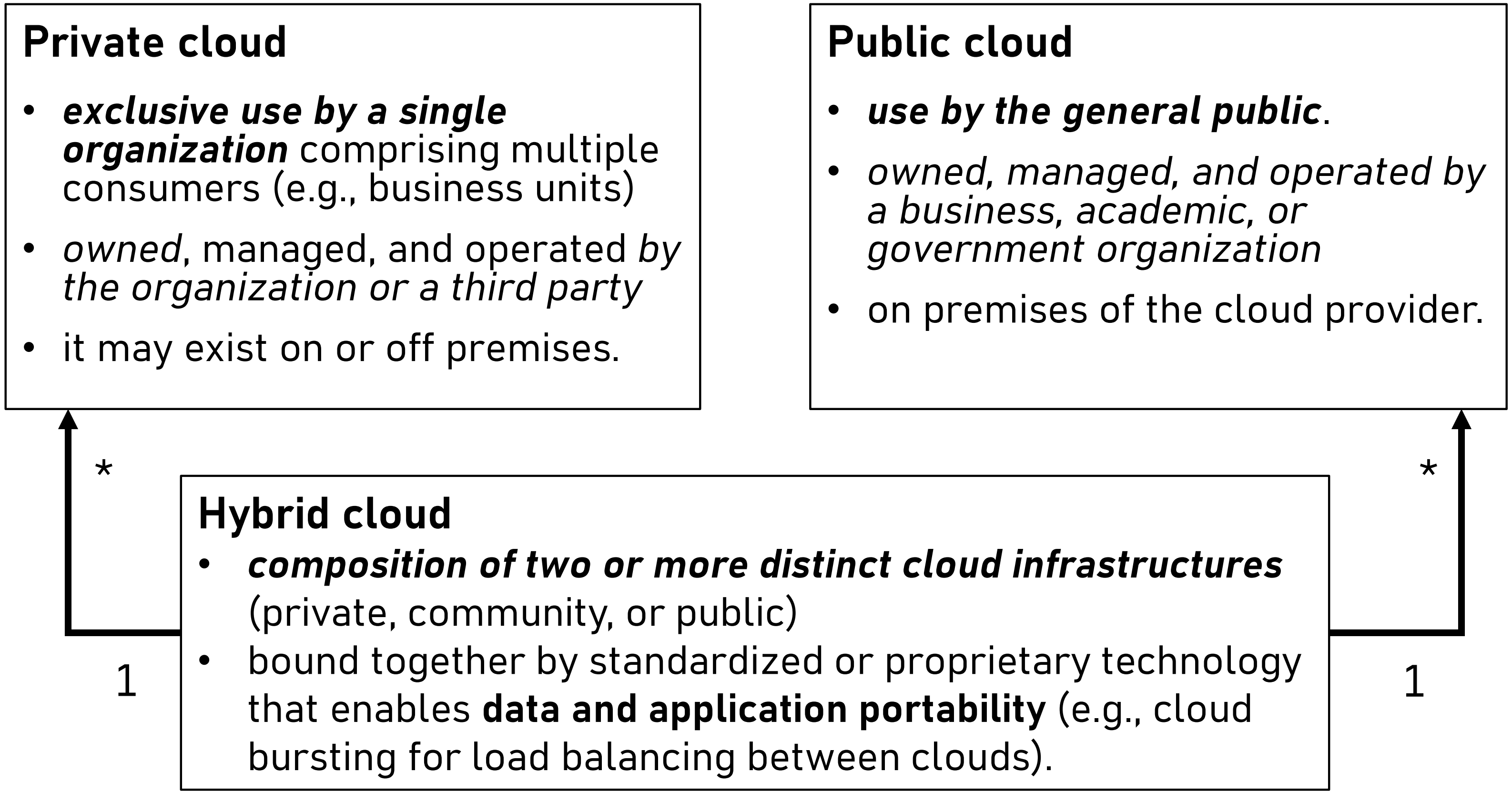
REGIONS

AVAILABILITY ZONES

EDGE LOCATIONS

Src: https://aws.amazon.com/de/compliance/shared-responsibility-model/

Cloud Tenancy Models



Private cloud

- ***exclusive use by a single organization*** comprising multiple consumers (e.g., business units)
- *owned*, managed, and operated *by the organization or a third party*
- it may exist on or off premises.

Public cloud

- ***use by the general public***.
- *owned, managed, and operated by a business, academic, or government organization*
- on premises of the cloud provider.

Hybrid cloud

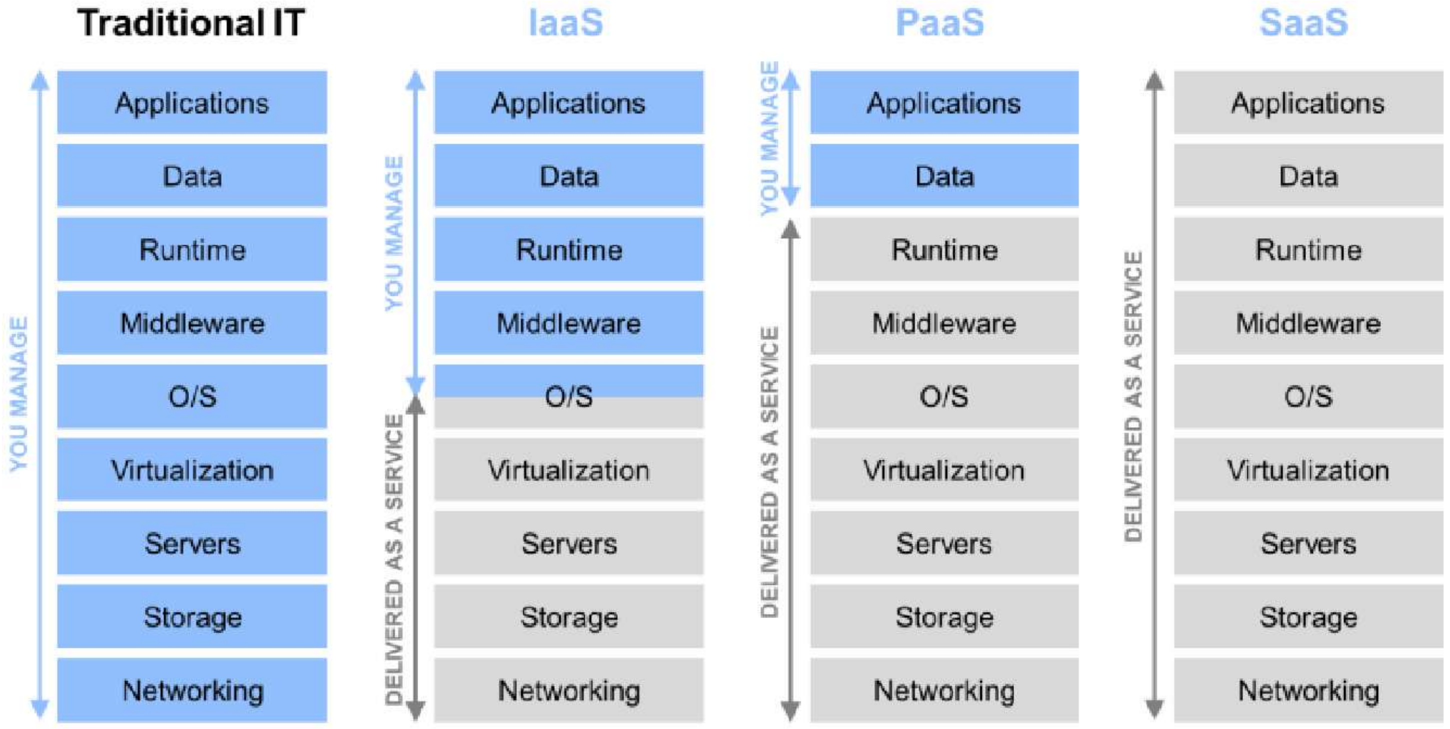
- ***composition of two or more distinct cloud infrastructures*** (private, community, or public)
- bound together by standardized or proprietary technology that enables **data and application portability** (e.g., cloud bursting for load balancing between clouds).

Src: NIST Special Publication 800-145 The NIST Definitionof Cloud Computing

Traditional Cloud Service Models

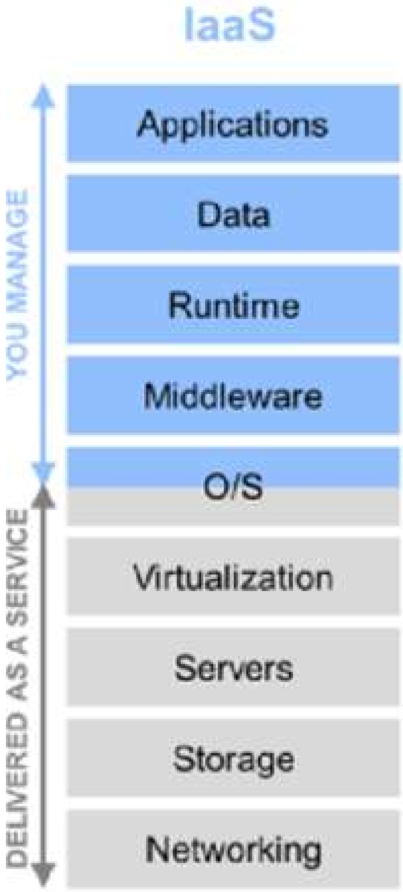
Source:
Microsoft Cloud Computing Whitepapers,
“The Economics of the Cloud”, 2010

Central criterion: How much control/responsibility does the customer have?



Traditional IT	IaaS	PaaS	SaaS
Applications	Applications	Applications	Applications
Data	Data	Data	Data
Runtime	Runtime	Runtime	Runtime
Middleware	Middleware	Middleware	Middleware
O/S	O/S	O/S	O/S
Virtualization	Virtualization	Virtualization	Virtualization
Servers	Servers	Servers	Servers
Storage	Storage	Storage	Storage
Networking	Networking	Networking	Networking

Infrastructure as a Service (IaaS)



- Provision of **processing, storage, networks** and other **fundamental computing resources**
- The consumer controls **operating systems**, **storage** and **deployed applications**
- Limited configuration of networking components possible (e.g. firewall rules)

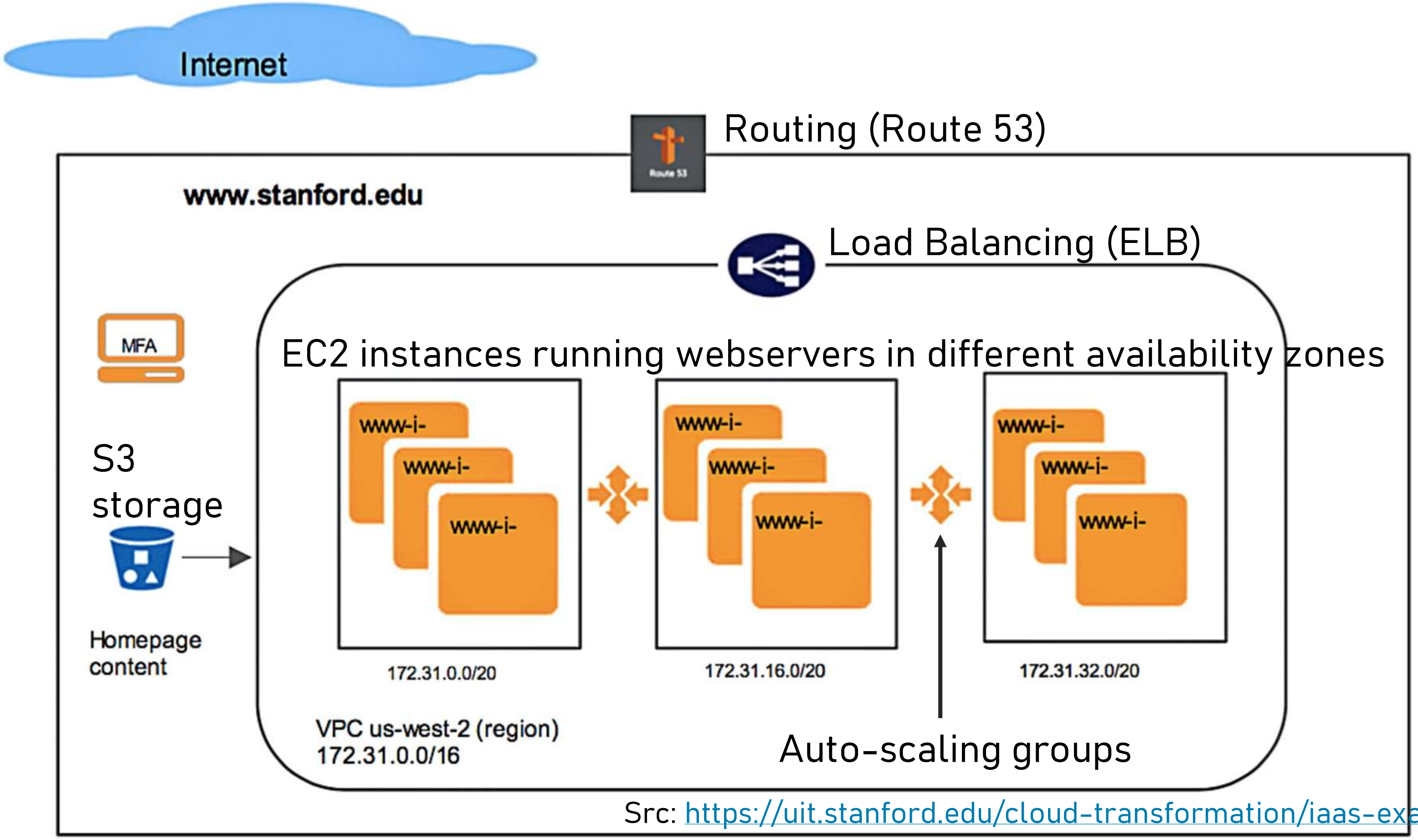
→Think of Virtual Machines

→Typical product: AWS EC2

→High Engineering Effort vs. Full Control

Src: Peter Mell and Tim Grance ,
The NIST Definition of Cloud Computing

Use Case - Stanford Homepage on AWS



Routing (Route 53)

Load Balancing (ELB)

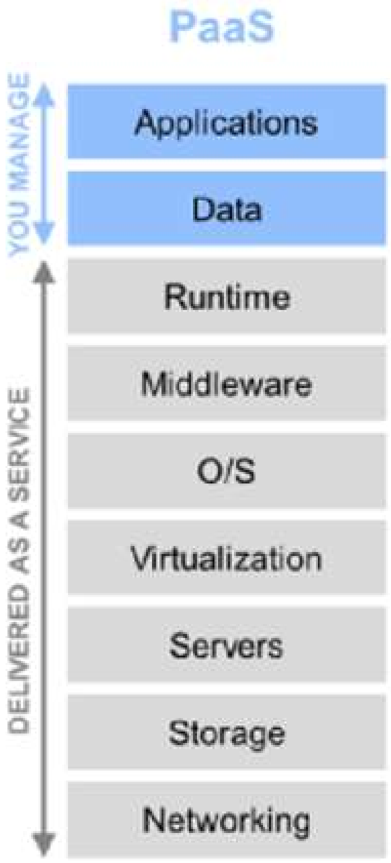
EC2 instances running webservers in different availability zones

S3 storage

Auto-scaling groups

Src: https://uit.stanford.edu/cloud-transformation/iaas-example

Platform as a Service (PaaS)



- Allows the **deployment** of costumer-created or acquired **applications**
- Applications are **restricted** to the **programming languages, libraries** and **services** supported by the provider
- The customer has control over deployed applications and possibly **configuration settings**.

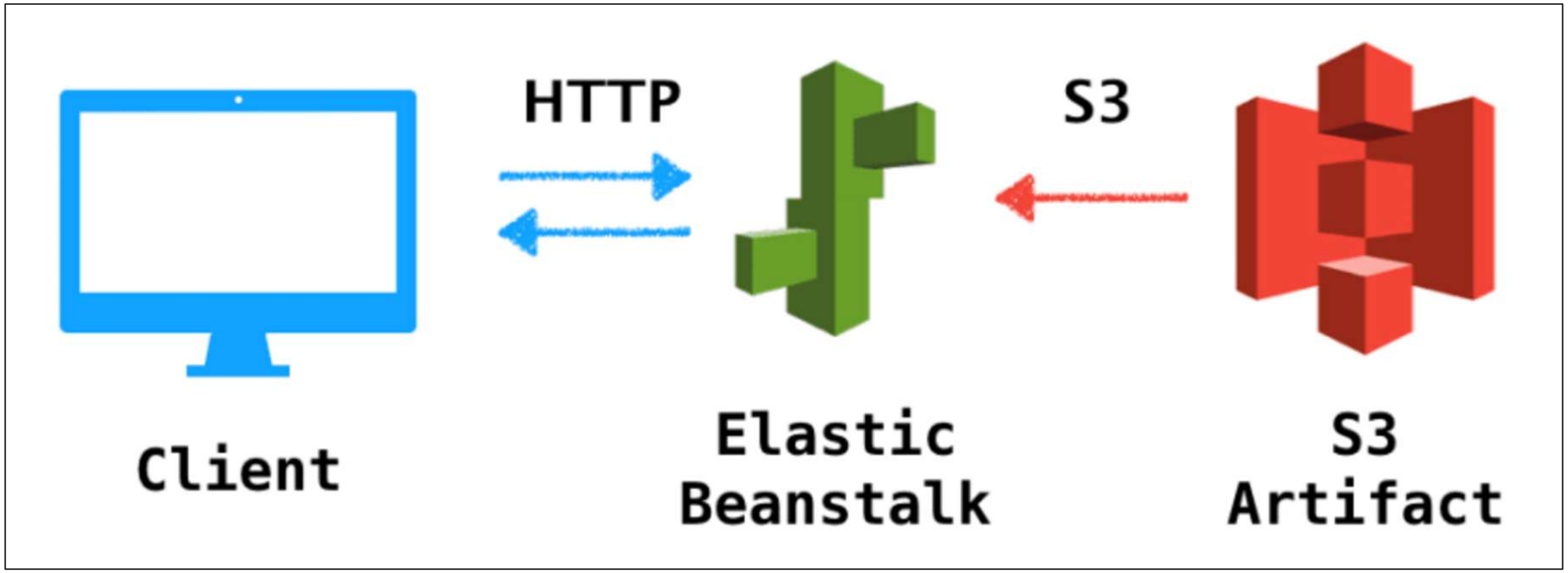
→Think of Virtual Machines

→Low Engineering Effort vs. Vendor Lockin

Src: Peter Mell and Tim Grance ,
The NIST Definition of Cloud Computing

Use Case – HTTP API for Backends

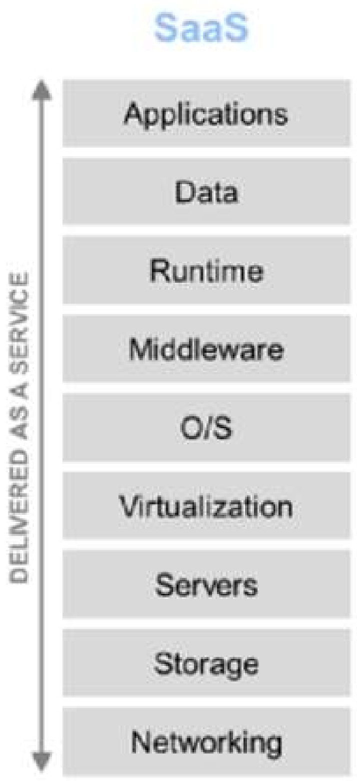
- AWS Elastic Beanstalk
  - S3 bucket needed to provide the application artifact



Src: https://dev.to/frosnerd/deploying-an-http-api-on-aws-using-elastic-beanstalk-5dh7

Software as a Service (SaaS)

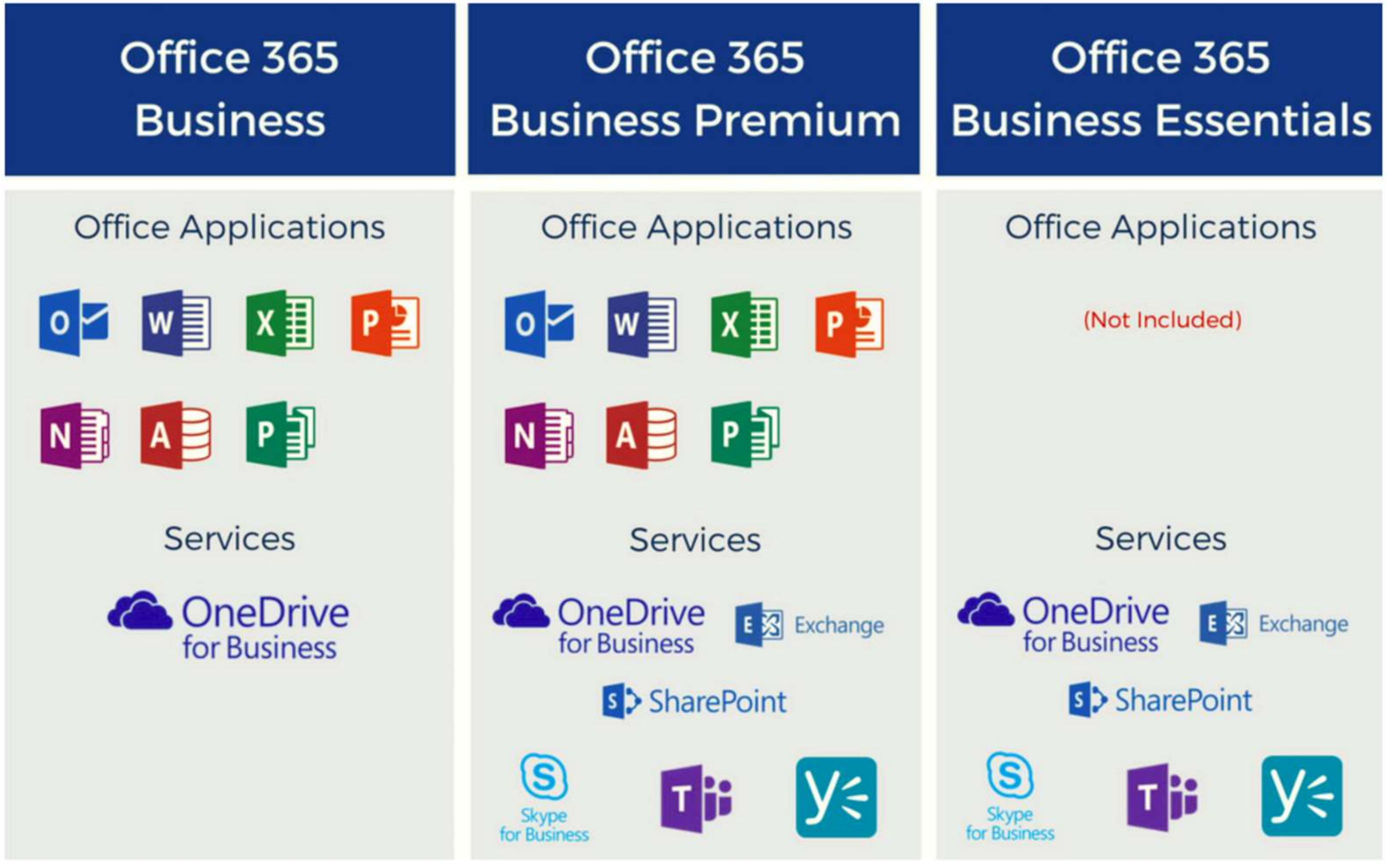
Src: Peter Mell and Tim Grance ,
The NIST Definition of Cloud
Computingw



- Applications that are running on a cloud infrastructure
- Customers can **use the applications**, but have **no control** over the cloud resources:
  - network, servers, operating systems, storage technologies or individual applications
- Possibility of **limited user specific application configuration**

IT companies may consume IaaS/PaaS services to provide SaaS services to their customers

SaaS Example – Microsoft Office 365



Office 365 Business	Office 365 Business Premium	Office 365 Business Essentials
Office Applications	Office Applications	Office Applications
O W X P N A P	O W X P N A P	(Not Included)
Services	Services	Services
OneDrive for Business	OneDrive for Business Exchange SharePoint Skype for Business T y	OneDrive for Business Exchange SharePoint Skype for Business T y

Src: https://www.sherweb.com/blog/office-365/office-365-plan-for-your-business/

SaaS – Benefits/Challenges

- Lower investment risk (no or low up-front cost)
- Predictable cost
- Faster rollout
- Mobility
  - often desktop, web and mobile apps available
- Automatic updates
- Automatic backups
- Focus on business tasks

- Vendor lock-in
- Possibly lower data transfer rates compared to on-premise-infrastructure
- No or less adaptability
- Lower security and potential data protection issues (physical location of data stores, certifications)
- Shallow in-house expertise

Can we do better than IaaS and PaaS?

(Not so Recent) Cloud Computing Trends

- **Function as a Service (FaaS)** service model
- Instance of **Serverless** computing
  - *"Serverless denotes a cloud computing style where the user does **not care about concrete server instances**"*
  - Of course for providing services there have to be servers
  - Examples for well-established Serverless Cloud Services:
    - Storage, Messaging
    - Google Cloud Functions, AWS Lambda, Azure Functions
    - Existing PaaS services, e.g.,
      Google App Engine (however, some argue)

Function as a Service (FaaS) [Eyk2017]

- FaaS is an **event-driven computing model**
- FaaS is a cloud service model, where a provider
  - provides a managed execution environment
  - abstracts (nearly) all operational tasks (gain of control)
  - facilitates **auto-scaling** of **short-lived**, **context-unaware** cloud functions
  - facilitates **scaling to zero** (no idle running functions, unique for FaaS)
  - enables a **pay-per-use** cost model
    (most granular billing option in present cloud technologies)
- FaaS is a cloud service model, where a user
  - writes **single-threaded**, performant, **stateless** cloud functions
  - specifies only a **few configuration parameters** (e.g. RAM, timeout)

FaaS: Cloud Function Execution

- Example: Photo upload in a S3 bucket, event processing:
    - Resize to standardized dimensions, convert to JPG (quality 90%), store processed images in a new bucket, keeping the original
- Functions are executed in **lightweight containers**
- **Cold starts** sometimes a problem for FaaS

Src: https://www.youtube.com/watch?v=eOBq__h4OJ4

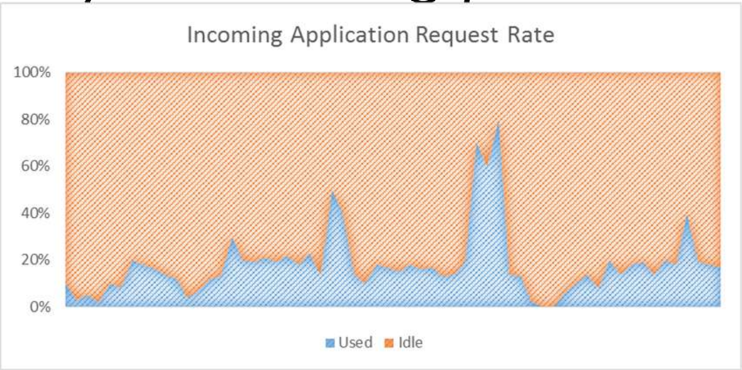
- Typical triggers
    - Http based triggers (e.g. API Gateway)
    - CRON triggers (e.g. Cloud Watch)
    - Queuing/ Notification triggers (e.g. SNS, SQS)
    - Database triggers (e.g. DynamoDb, S3)

FaaS: Powerful Use Cases

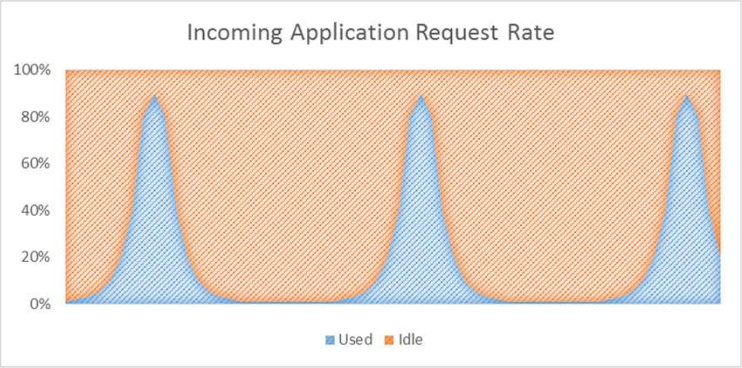
Solving especially under-/overprovisioning problem

You can do this with IaaS as well.
FaaS makes it harder to get it wrong ;-)

- Bursty Workloads



- Periodic Workloads

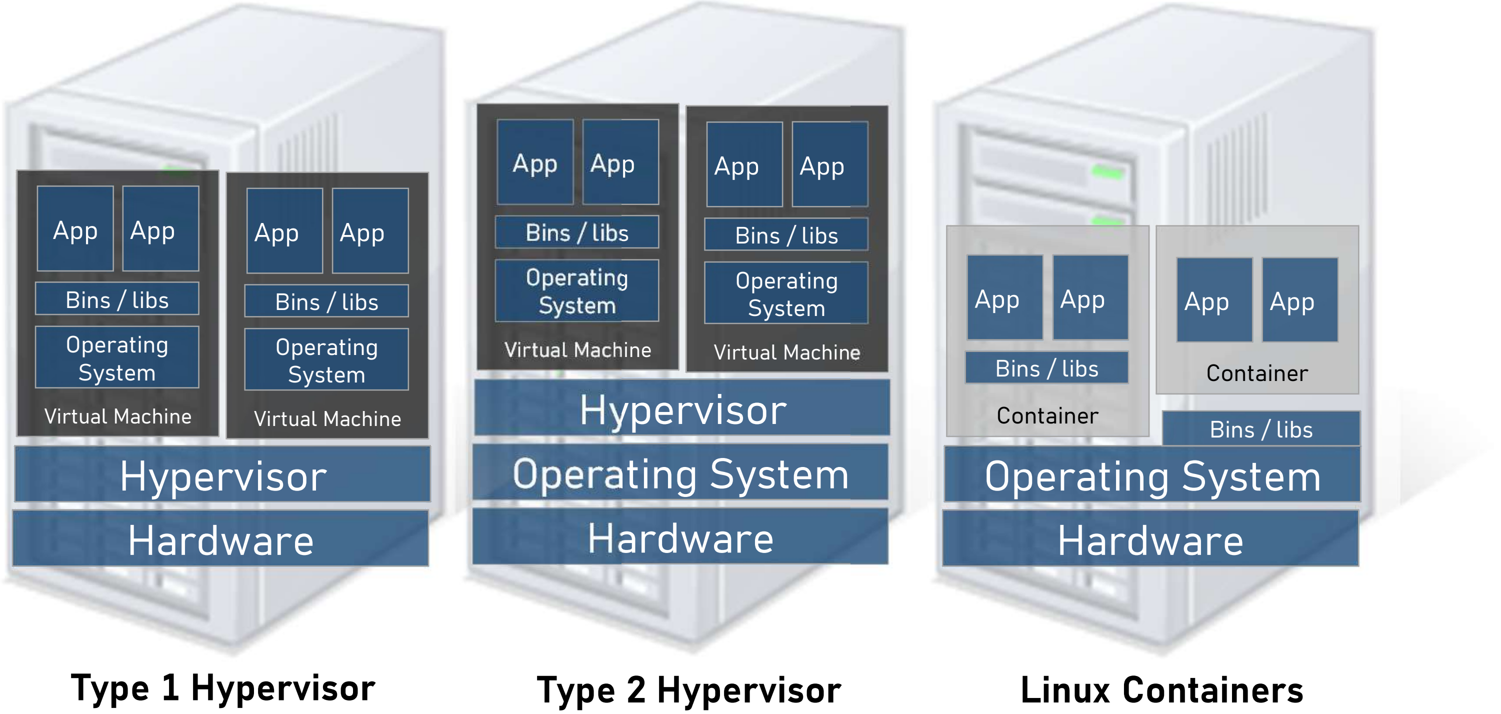


(Not so Recent) Cloud Computing Trends

- **Container as a Service (CaaS)** service model
- Containers are a **lightweight** alternative to traditional virtualization
- Run many applications in a **fast**, **portable** and **isolated** way in many different environments!
- **Containerization** in Linux is not a new concept, but employing it used to be rather cumbersome

Hypervisors vs. Linux Containers

Containers share the OS kernel of the host and thus are lightweight. However, each container must have the same OS kernel.

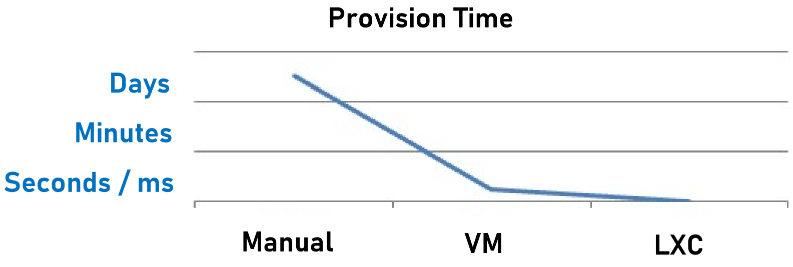


Containers are isolated, but share OS and, where appropriate, libs / bins.

Src.:Boden Russell, IBM

Why Linux Containers

- Provision in seconds / milliseconds
- Near bare metal runtime performance
- VM-like agility – it's still "virtualization"
- Flexibility
  - Containerize a "system"
  - Containerize "application(s)"
- **Lightweight**
  - Just enough Operating System
  - **Minimal per container penalty**
- Open source – free – lower TCO
- Supported with OOTB modern Linux kernel



Src.:Boden Russell, IBM

Docker

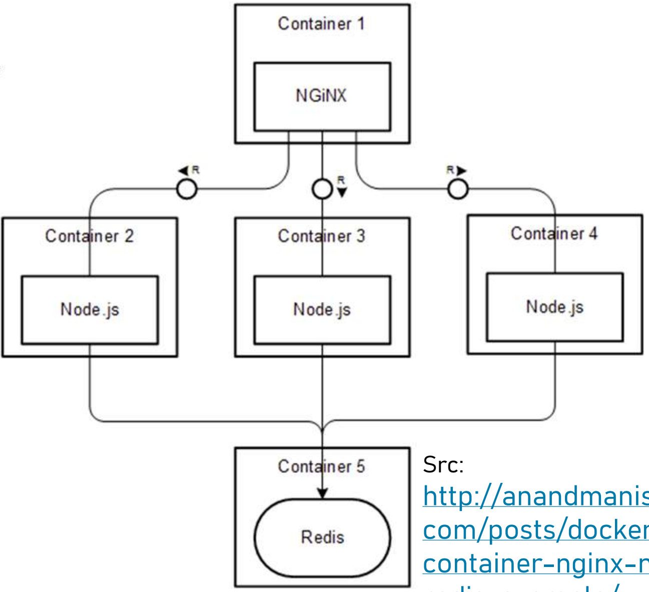
- In 2013 **Docker** was released, which made container technology very accessible
- What Docker does:
  - "Docker allows you to package an application with all of its dependencies into a **standardized** unit for software development." → Docker image
  - "Docker containers wrap up a piece of software in a complete filesystem that **contains everything** it needs to run.
  - This guarantees that it will (almost) **always run the same**, regardless of the environment it is running in."

Key promise: **Build, ship and run everywhere!**

Image src: https://commons.wikimedia.org/wiki/File:Docker_(container_engine)_logo.svg

Setting up a Containerized System

- Problem with running productive Docker applications
  - Best practice: One service per image/container
  - Applications typically consist of multiple, isolated service units
  - Manual orchestration (startir containers is a hassle
    → use Docker compose
    → Use Kubernetes



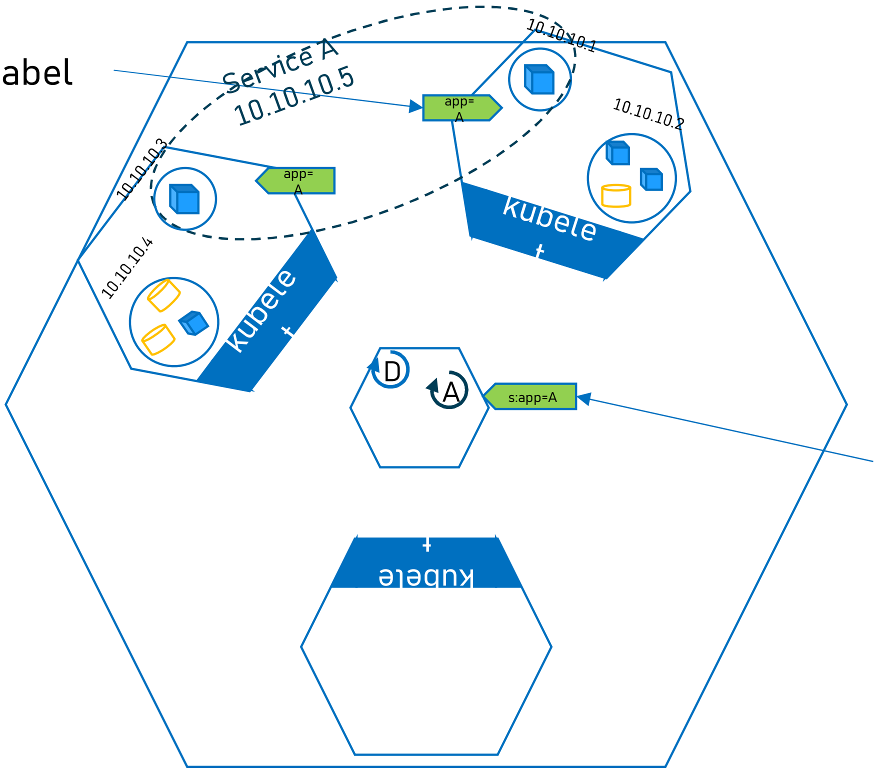
Src: http://anandmanisankar.com/posts/docker-container-nginx-node-redis-example/

Kubernetes

- Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of **containerized applications**
- Declarative configuration
- Comparison of desired state with observed state
  - Self-healing
- Multiple physical or virtual machines **Nodes** are joined to a **Kubernetes Cluster** which is controlled by a **Master**

Kubernetes – Core Concepts

- A **pod** contains containers + volumes (= app instance)
- A **kubelet** controls pods on a node (= a machine)
- A **master** controls a cluster
- **Services** a realized by multiple pods and identified by **labels**



Label

Label Selector

Kubernetes Cluster

Adapted from https://kubernetes.io/docs/tutorials/kubernetes-basics/

Containers in the Cloud

- Two basic ways of software deployment with containers in the cloud
- Using **Kubernetes Clusters**
  - More control, cluster management and container
  - Clouds provide Kubernetes clusters as a service
  - Deployment configuration must be provided by the customer
    → Effort for realizing sensible load balancing, scaling…

e.g. AWS EKS
Elastic Kubernetes Service

- Using **Docker Container** Services
  - Control of containers, but not cluster management
  - Essentially, managing containers with more vendor-defined settings

e.g. AWS ECS
Elastic Container Service

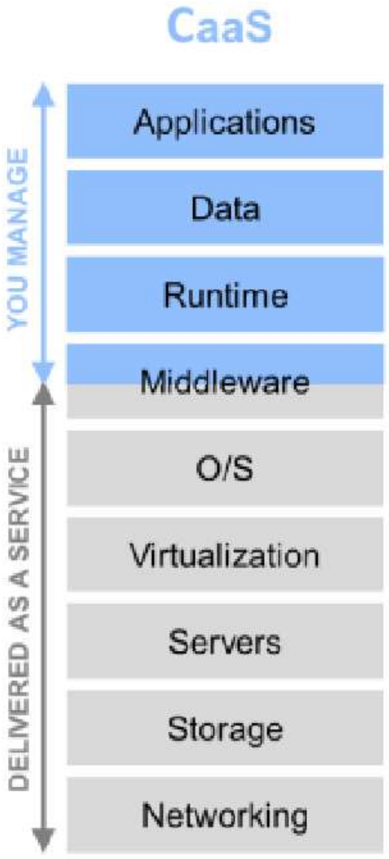
- Do not confuse with PaaS: Some CSPs allow Docker containers to be deployed to extend their PaaS products

Containers in the Cloud

Option 1: Using Kubernetes Clusters

- Closer to IaaS
- More control
  - Container
  - Cluster management
- Kubernetes cluster is provided by the cloud
- Deployment configuration must be provided by the customer
- Effort for configuring load balancing, scaling, networking, …

e.g. AWS EKS
Elastic Kubernetes Service

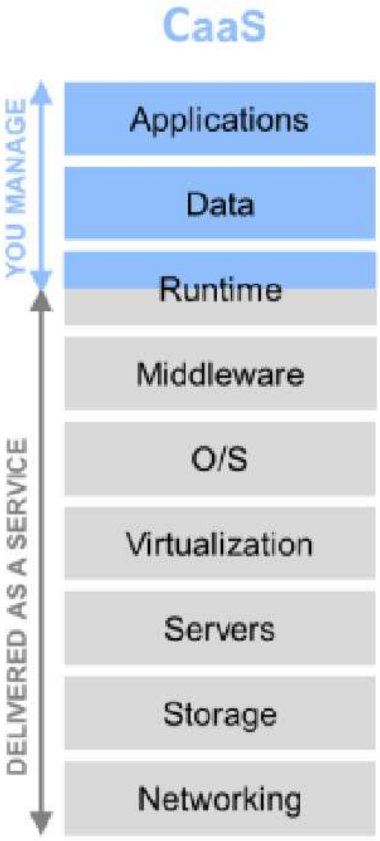


Containers in the Cloud

Option 2: Using Docker Containers

- Less control
  - Just Containers
  - No cluster management
- Upload of Docker images to the cloud
- Configuration is done via user interface or API
  - E.g. set minimum, maximum number of containers, port forwarding
- Load balancing, scaling and networking is handled by the service

e.g. AWS ECS Elastic Container Service



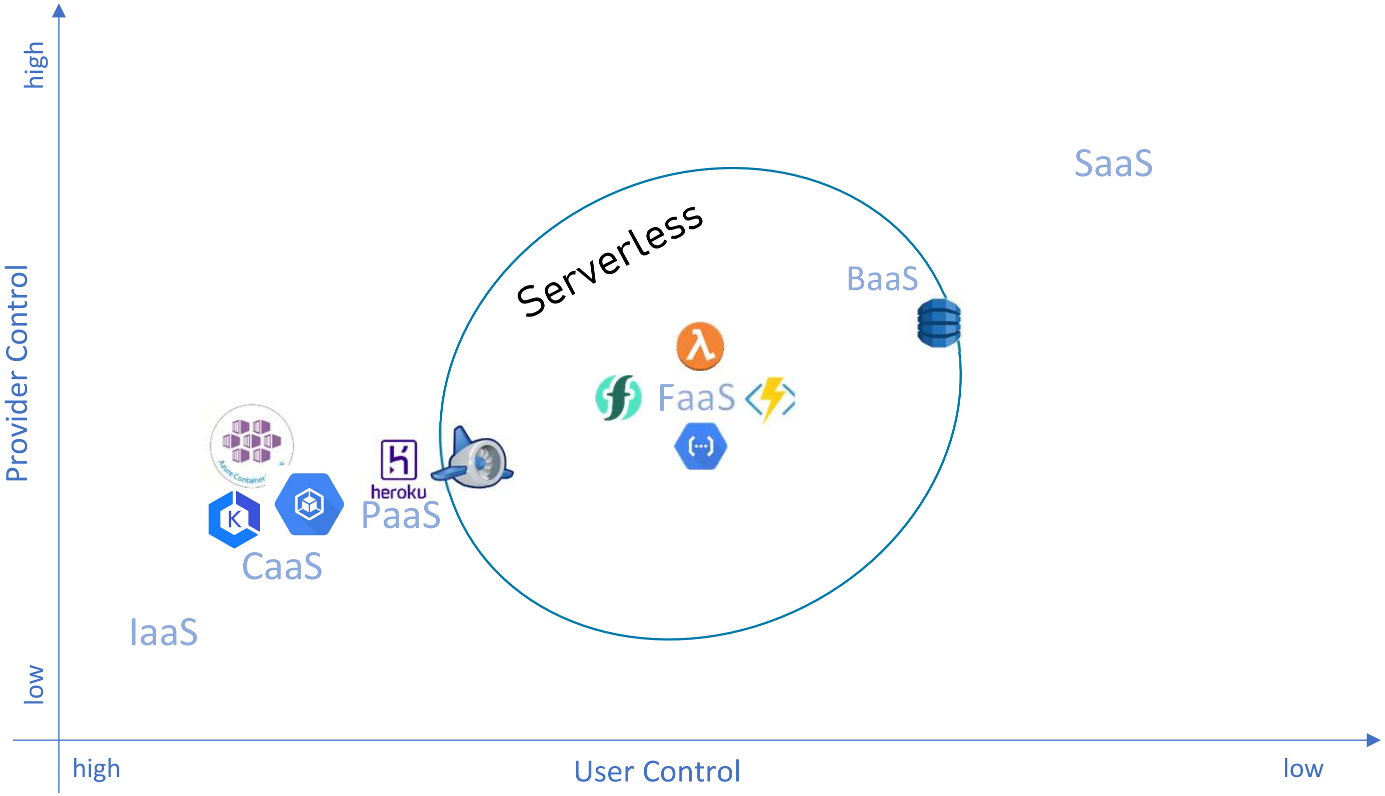
CaaS Considerations

- Docker and Kubernetes are **field-tested technologies** that can be run **on-premise for developing and staging purposes**
- Danger of **vendor lock-in** is somewhat **mitigated** by relying on open source technologies. However, cloud applications will probably still use cloud-specific services
- Deploying applications with Kubernetes is not trivial
- Some PaaS services also allow uploading a Docker container
  - Not CaaS in a strict sense because Docker is merely used to enable more flexible runtime environments

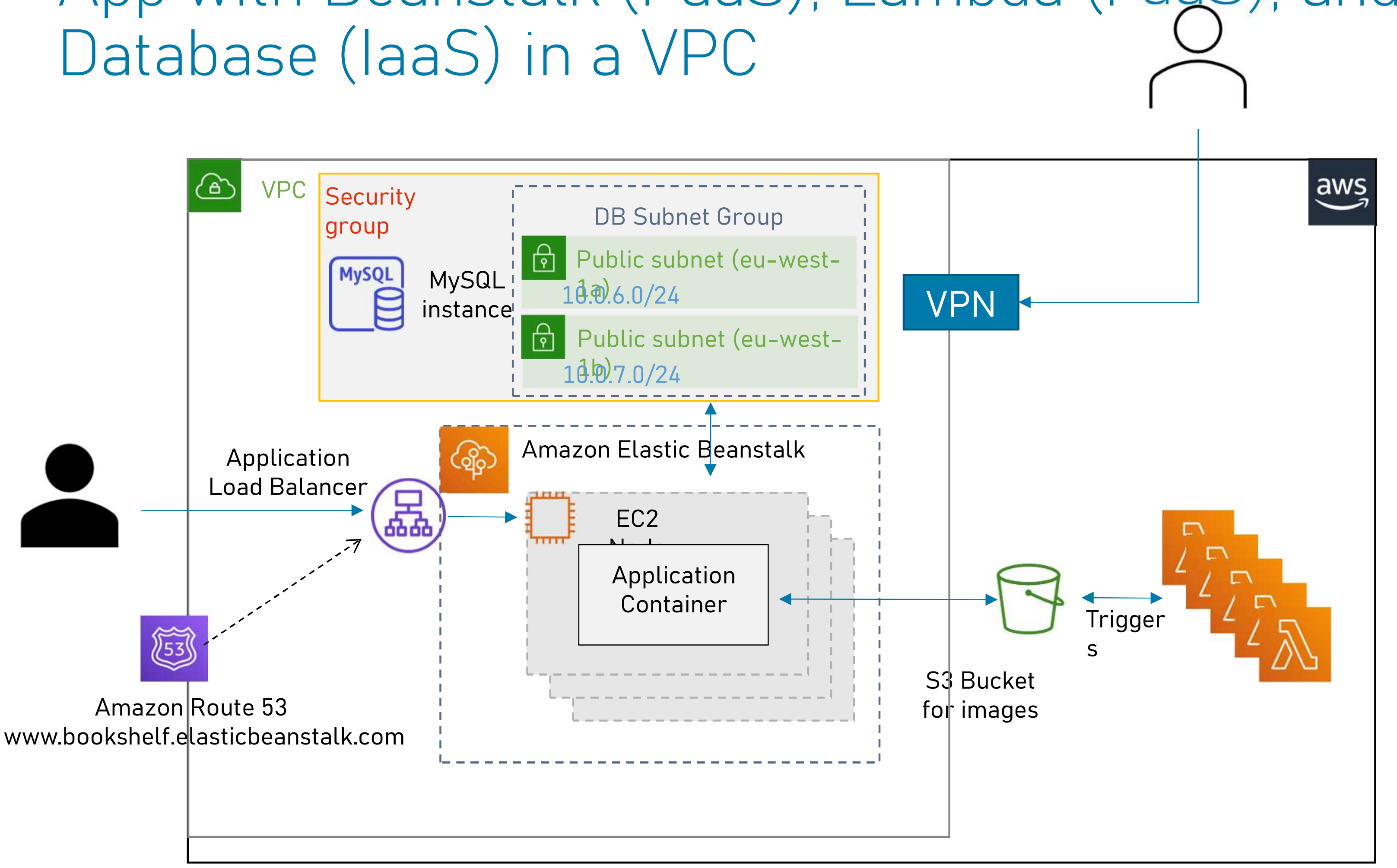
Backend as a Service (BaaS)

- Provides **backend cloud storage** and **standardized solutions** to common tasks to developers of web and mobile apps
- Features usually include
  - Authentication
  - Analytics
  - Push notifications
  - Storage
  - Hosting
  - Serverless Functions
- AWS Amplify, Google Firebase

… as a Service - Summary



Let's talk about Cost
App with Beanstalk (PaaS), Lambda (FaaS), and Database (IaaS) in a VPC



Cost estimation

https://calculator.aws/#/estimate?id=67533da036ab59ffad9abc80a940a40f2a9d4a73

1 VPN Connection to VPC for administrative tasks

~432000 images processed per month (~ 100 KB / image)

All required quantities for cost calculation are rough estimates!

-> Know the profile of your Application for each service of Cloud provider (monitor and evaluate continuously!)

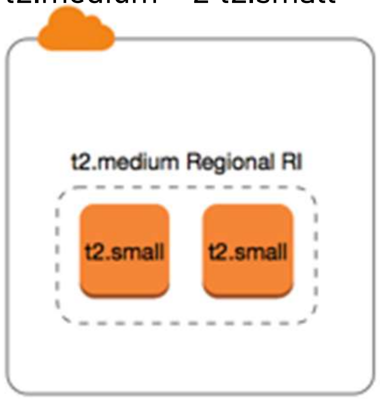
AWS Pricing - Reserved Instances – EC2 IaaS

- Regional
  - no reserved capacity
  - discount for any EC2 instance of applicable instance family (e.g. t2) and OS in any AZ
  - instance size flexibility
  - Limit: 20 per Region per month
- Zonal
  - no AZ flexibility
  - no instance size flexibility
  - guaranteed capacity
  - Limit 20 per AZ per month

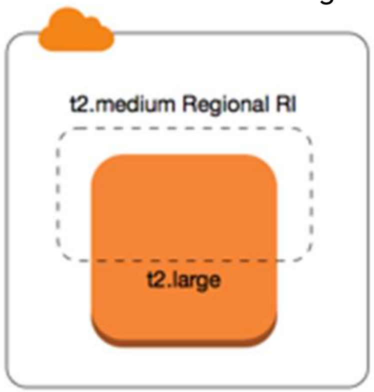
- Size flexibility

Instance size	Normalization factor
nano	0.25
micro	0.5
small	1
medium	**2 (reserved)**
large	4
xlarge	8

t2.medium = 2 t2.small



t2.medium = 0.5 t2.large



AWS Pricing – EC2 Offering Classes

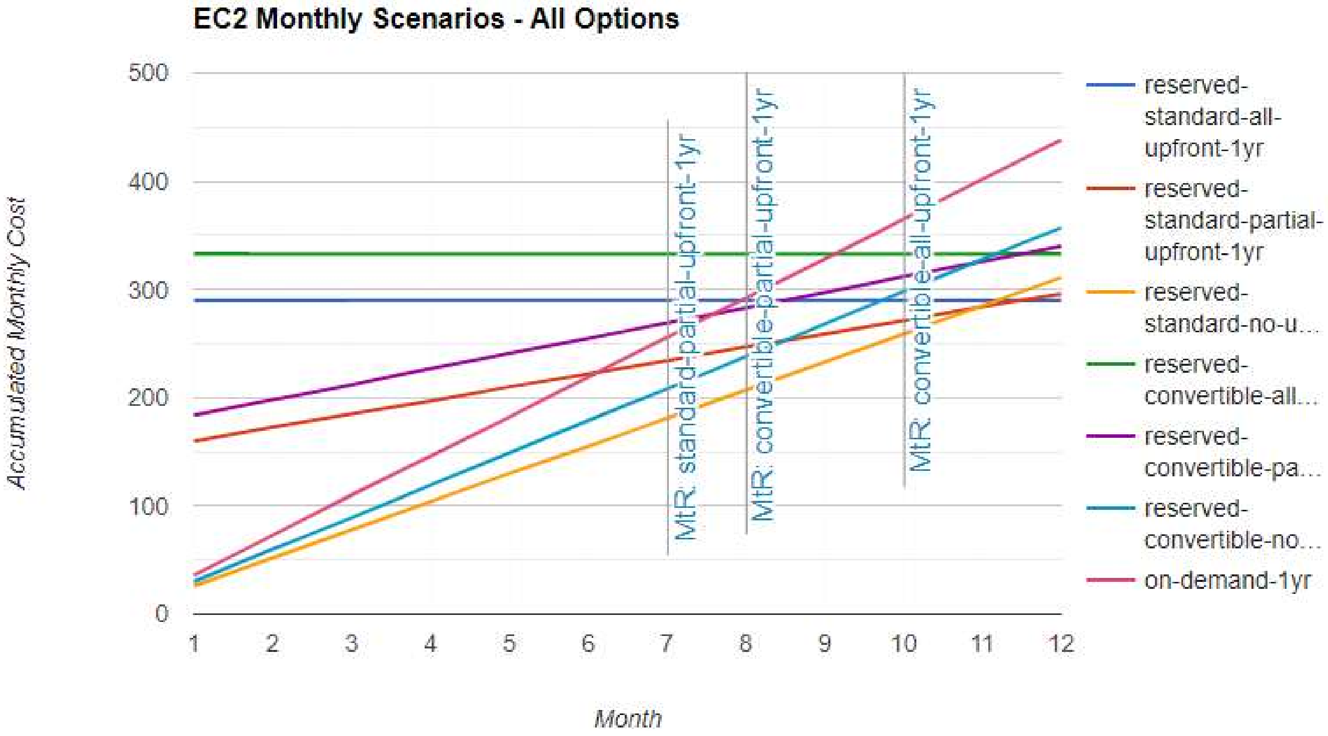
- On Demand (pay as you go) – most expensive and most flexible
- Standard
  - Fixed instance family, but changeable size
  - Sellable on Marketplace (bank with US address required)
- Convertible
  - Change instance family
  - more expensive than Standard
  - Not sellable on Marketplace

- EC2 Instance Saving Plans
  - Cost commitment per hour
  - Fixed instance family, but changeable size
- Compute Saving Plans
  - Cost commitment per hour
  - Flexibility: all instance families and sizes

- Reserved Pricing
  - Duration: 1 year or 3 years
  - Payment:
    - All Upfront
    - Partial Upfront
    - No Upfront (pay every month)

AWS Pricing

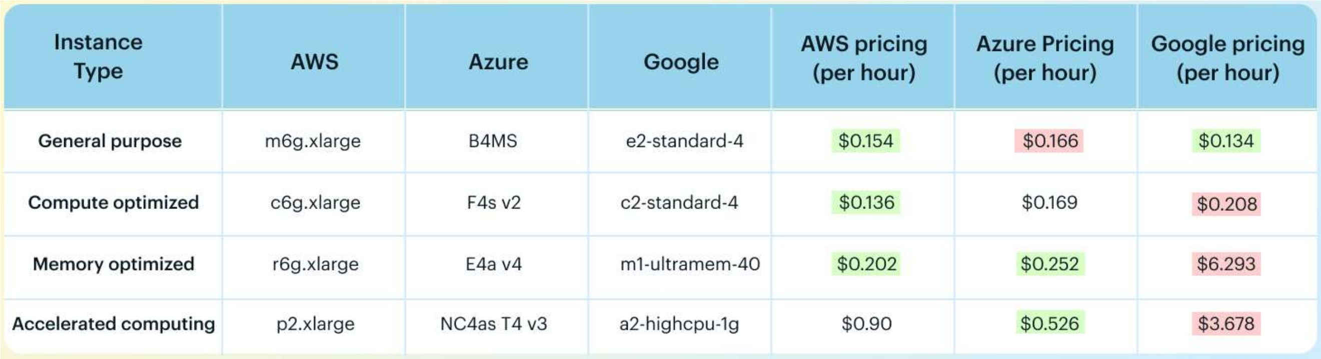
- t2.medium – eu-west-1 – 1 year



Calculated using AWS Price List API (2021-05-26) and Concurrency Labs' AWS Pricing Tools
https://github.com/concurrencylabs/aws-pricing-tools

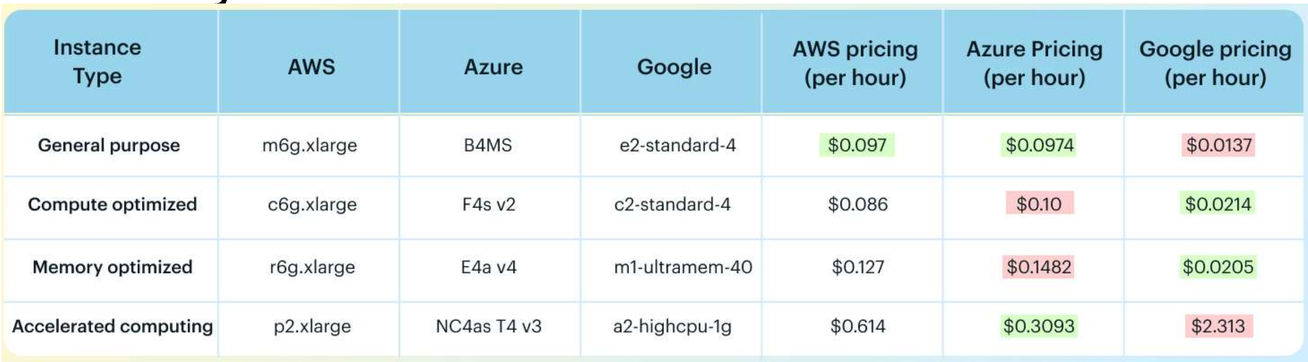
Pricing – Compute (IaaS)

- On Demand



Instance Type	AWS	Azure	Google	AWS pricing (per hour)	Azure Pricing (per hour)	Google pricing (per hour)
General purpose	m6g.xlarge	B4MS	e2-standard-4	$0.154	$0.166	$0.134
Compute optimized	c6g.xlarge	F4s v2	c2-standard-4	$0.136	$0.169	$0.208
Memory optimized	r6g.xlarge	E4a v4	m1-ultramem-40	$0.202	$0.252	$6.293
Accelerated computing	p2.xlarge	NC4as T4 v3	a2-highcpu-1g	$0.90	$0.526	$3.678

- Discount – 1 year commitment



Instance Type	AWS	Azure	Google	AWS pricing (per hour)	Azure Pricing (per hour)	Google pricing (per hour)
General purpose	m6g.xlarge	B4MS	e2-standard-4	$0.097	$0.0974	$0.0137
Compute optimized	c6g.xlarge	F4s v2	c2-standard-4	$0.086	$0.10	$0.0214
Memory optimized	r6g.xlarge	E4a v4	m1-ultramem-40	$0.127	$0.1482	$0.0205
Accelerated computing	p2.xlarge	NC4as T4 v3	a2-highcpu-1g	$0.614	$0.3093	$2.313

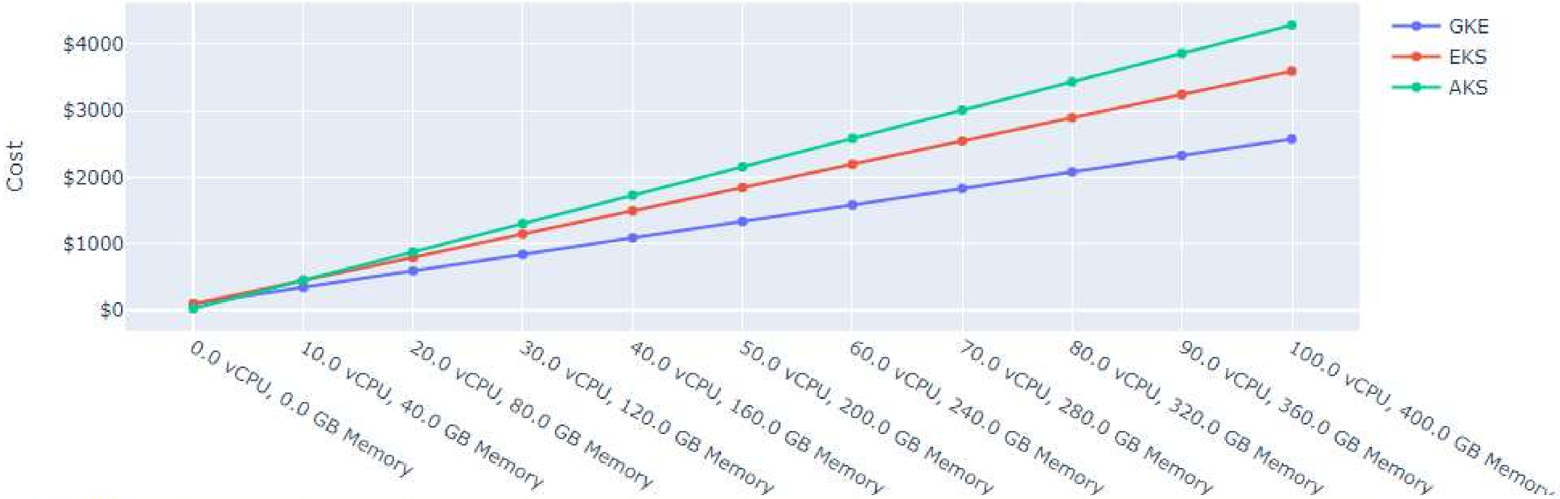
Pricing - FaaS

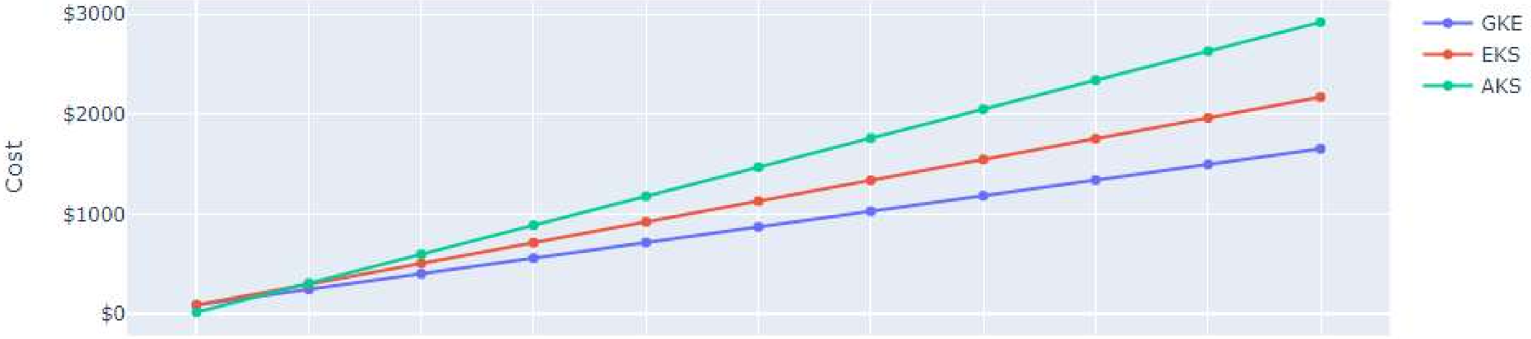
Provider	Free Monthly Duration (GB-seconds)	Free Monthly Requests	Cost of Each Additional 1 Million Requests	Cost of Each Additional 1 GB-second	Duration is Rounded to the Nearest
AWS	400,000	1 Million	$0.20	$0.000016	1ms
Azure	400,000	1 Million	$0.20	$0.000016	1ms
GCP	400,000	2 Million	$0.40	$0.0000125	100ms

Pricing – CaaS (Kubernetes)

- On Demand monthly cost

Adapted from https://github.com/sidpalas/managed-kubernetes-pricing/





- 1Year commitment monthly cost

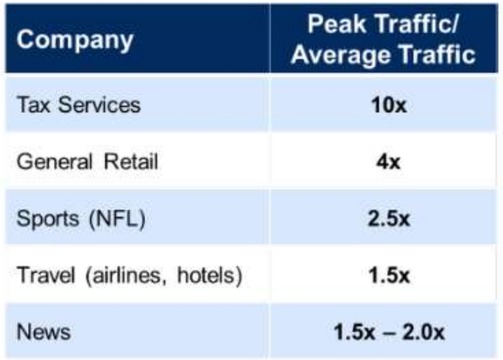
Outline

1. What is the Cloud?
2. Cloud Service Models
3. **Why move / move not into the Cloud?**
4. Software Development for the Cloud

Reasons to Move into the Cloud

- Cost/CPU hour, Cost/storage may be more expensive in the cloud, but…
  - Easy 'in', (easy 'out' ?),
  - Time to Market (TTO)
  - Consider Total Cost of Ownership (TCO)
    - Personnel, Skillset
    - Housing, electricity etc.
    - Elasticity (be able to handle peak loads), risk
  - New types of applications
  - Cloud Providers can leverage **economies of scale**
    - Very large data centers 10s of 1000s / 10000s
    - Reuse the same hardware for different customers

FIG. 12: INDUSTRY VARIABILITY



Company	Peak Traffic/ Average Traffic
Tax Services	10x
General Retail	4x
Sports (NFL)	2.5x
Travel (airlines, hotels)	1.5x
News	1.5x – 2.0x

Source:
Microsoft Cloud Computing Whitepapers,
"The Economics of the Cloud", 2010

Triggers to Move into the Cloud

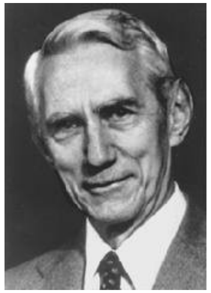
- Data center contract renewals
- Acquisitions
- Increased capacity requirements
  - e.g. rapid growth of userbase
- Software and hardware refresh cycles
- Security threats
- Compliance needs
- Product development benefits
- End-of-life events

Src: https://cloud.google.com/blog/products/storage-data-transfer/8-common-reasons-why-enterprises-migrate-to-the-cloud

Reasons to Stay Off the Cloud

Src: Michael Armbrust et al., "Above the Clouds: A Berkeley View of Cloud Computing", UCB Tech. Reports, 2009:

- Availability of Service
- Data Lock-In
- Data Confidentiality and Auditability
- Data Transfer Bottlenecks
- Performance Unpredictability
- Scalable Storage
- Bugs in Large-Scale Distributed Systems
- Scaling Quickly
- Reputation Fate Sharing
- Software Licensing
- Compounded pricing



Claude Shannon, creator of modern information theory;

Security is a question of attacker's computing power, key safety and information leakage per message.

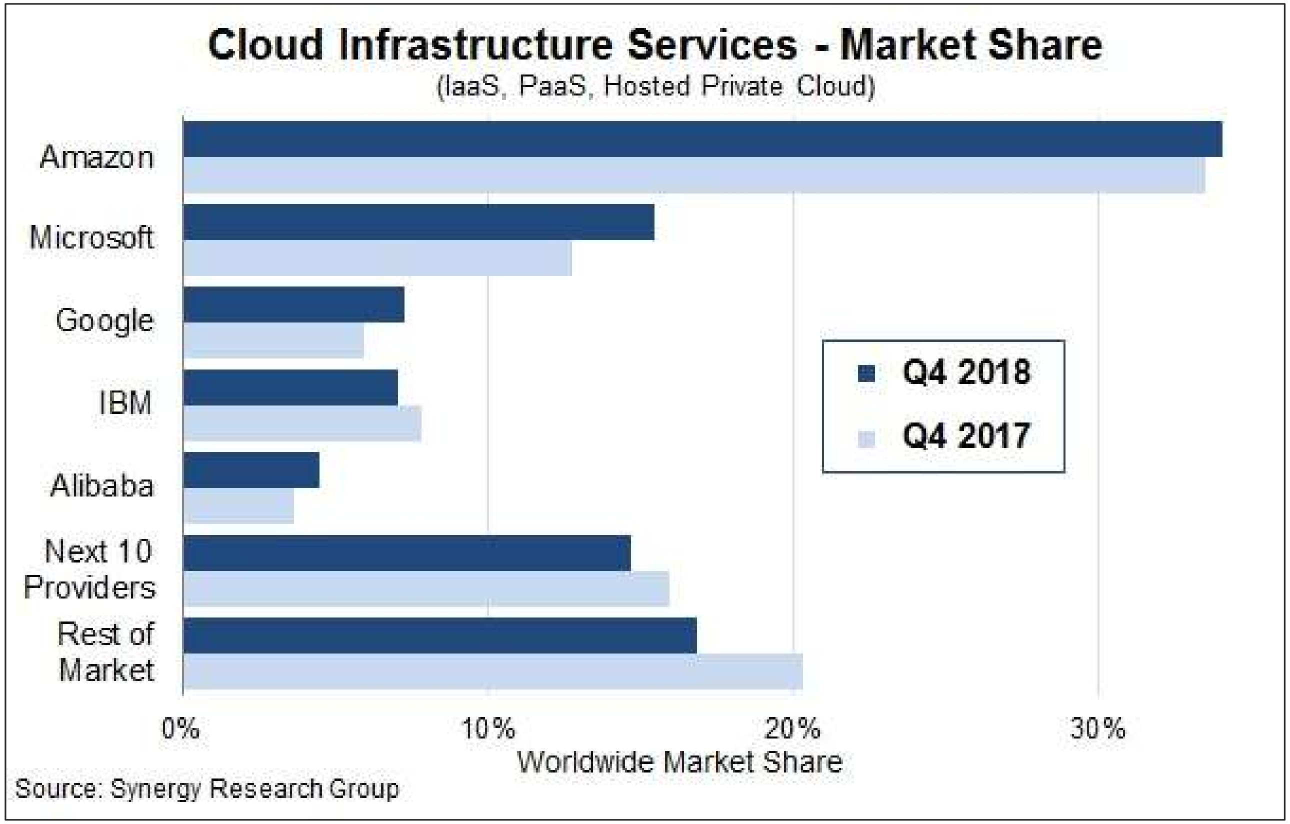
The cloud changes two important factors negatively:

- **attacker power**
- **attacker incentive**

Why is Everybody so Keen on Choosing one Single Cloud Provider?

Cloud Provider Selection – I

- Q4 2018



Source: Synergy Research Group

Src: https://www.srgresearch.com/articles/fourth-quarter-growth-cloud-services-tops-banner-year-cloud-providers

Cloud Provider Selection – II

- Certifications & Standards
- Technologies & Service Roadmap
- Data Security, Data Governance and Business policies
- Service Dependencies & Partnerships
- Contracts, Commercials & SLAs
- Reliability & Performance
- Migration Support, Vendor Lock in & Exit Planning
- Business health & Company profile

Src: https://www.cloudindustryforum.org/content/8-criteria-ensure-you-select-right-cloud-service-provider#sdep

Cloud Provider Selection – III

- BSI: Anforderungskatalog Cloud Computing (C5)
  - **C**loud **C**omputing **C**ompliance **C**ontrols **C**atalogue
  - **Environmental parameters**
    - System description
    - Jurisdiction and locations of data storage, processing and backup
    - Disclosure and investigatory powers
    - Certifications
- ISO/IEC 27017: Guidelines specially for cloud computing, complementary to:
  - ISO/IEC 27001 and ISO/IEC 27002 information security standards
- Big cloud providers usually have them all (also Alibaba)

Exemplary Use Cases

Cloud-Only Services - I

Example: „digital Service-Mgt.“

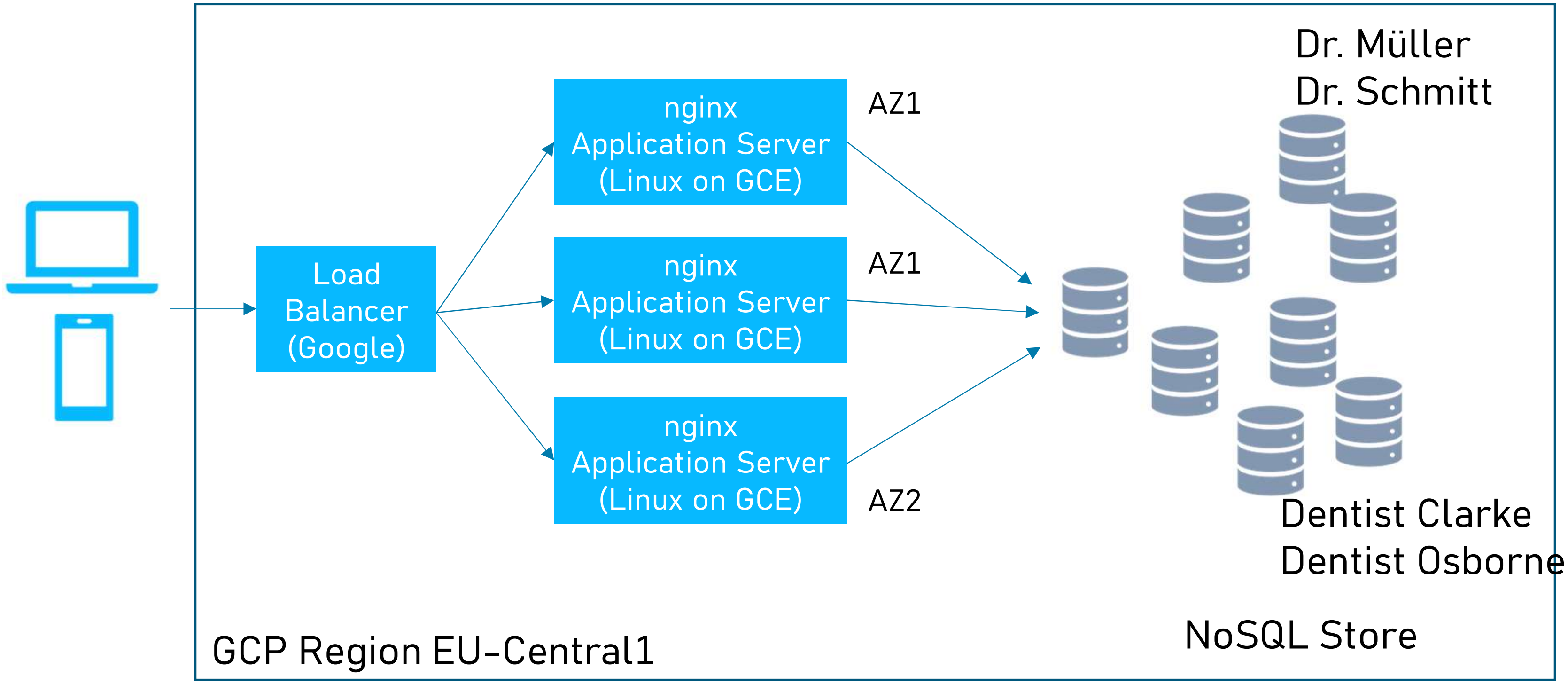
- List of bought products
- Value-Added Services
- Personal contact data
- Online inbox functionality
- Maintenance information

Praxis Dr. Müller

Praxis Dr. Schmitt

Image Src.
Jason Taix, Pixabay
StockSnap, Pixabay
Michal Jarmoluk, Pixabay

Cloud-Only Services – II



Load Balancer (Google)
nginx Application Server (Linux on GCE)
AZ1
nginx Application Server (Linux on GCE)
AZ1
nginx Application Server (Linux on GCE)
AZ2
Dr. Müller
Dr. Schmitt
Dentist Clarke
Dentist Osborne
NoSQL Store
GCP Region EU-Central1

Essentially, you provide an account management application where you may allow the dentist to manage devices / users himself.

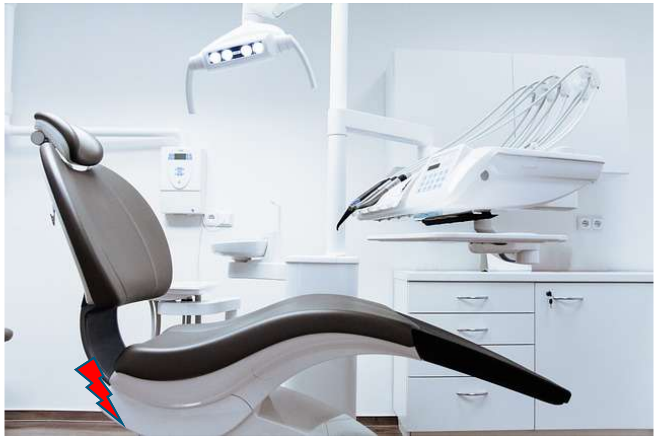
Disclaimer: There are a lot more technical options!

App Backend – I

Example: „Digital claim management“

- Scan your device identifier
- Take a photo of the issue

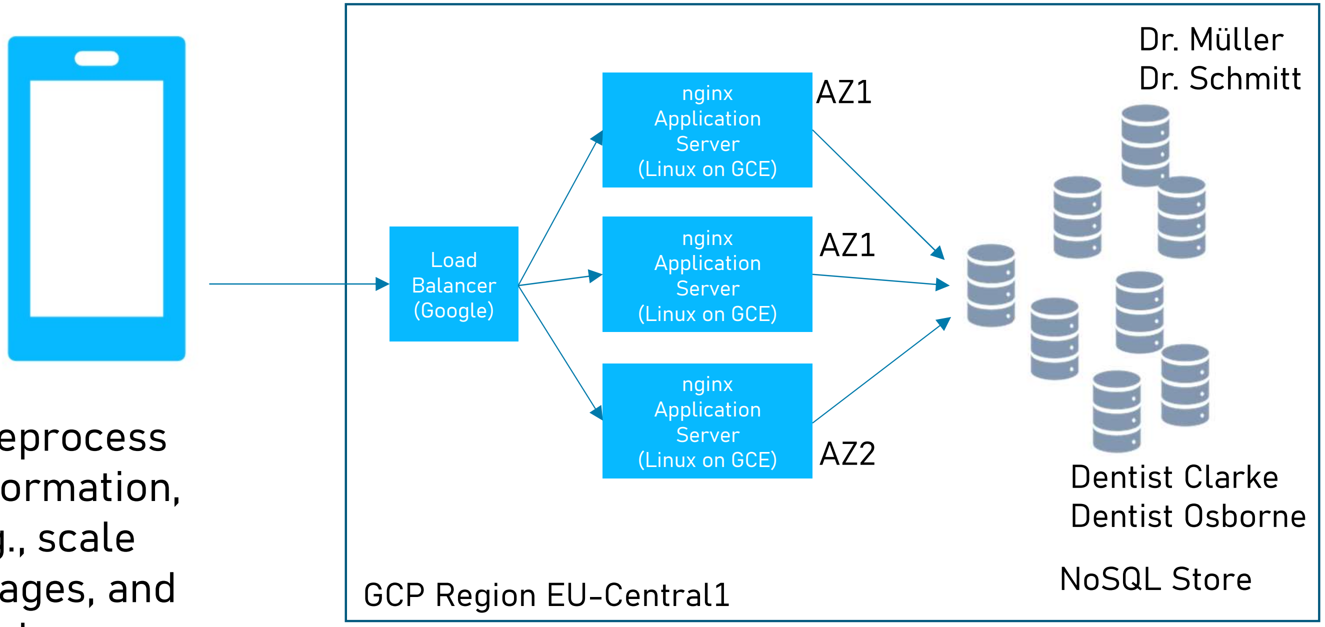
- Push issue to your Account Manager



⚡ Drive is broken

Image Src.
StockSnap, Pixabay

App Backend – II



Preprocess information, e.g., scale images, and push information to user

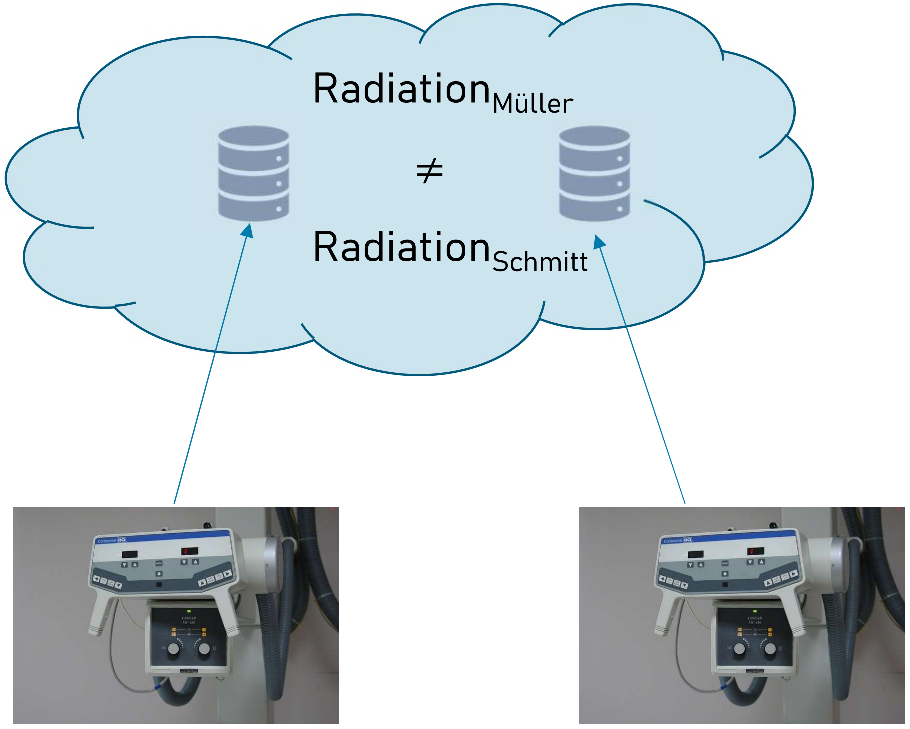
The architecture is not so different from the cloud only service. In this case, you offer additional convenience functionality on the mobile device.

Disclaimer: There are a lot more technical options!

IoT-Application – I

Example:
„Dose compare“

- Track dose of your imaging devices locally
- Collect dose data in cloud
- Analyze dose data for dentist feedback, e.g., overdose compared to others



$\text{Radiation}_{\text{Müller}}$

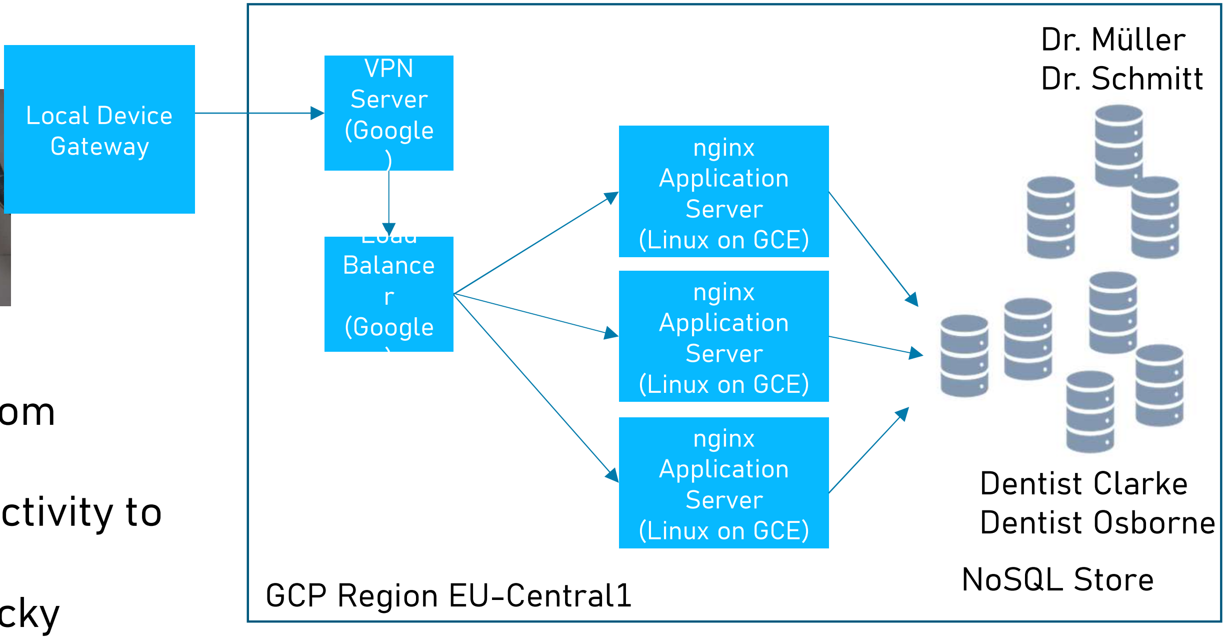
$\neq$

$\text{Radiation}_{\text{Schmitt}}$

Praxis Dr. Müller

Praxis Dr. Schmitt

IoT Application – II



Collect data from device;
manage connectivity to cloud;
Separate IT-lacky functions from safety functions

There is some similarity in architecture to connecting smartphones or tablets, however connectivity management for remote devices, a common platform for local gateways and data protection are the real challenges.

Disclaimer: There are a lot more technical options!

Exemplary Cloud Transformation

Cloud Transformation

- The following slides look at specific Cloud Transformation Topics
  → not a general IT transformation treatment
- Instead of presenting a theoretic transformation framework, we look at what others have done to bring across a gut feeling for cloud transformation

Greenfield

Greenfield Project: Dropbox

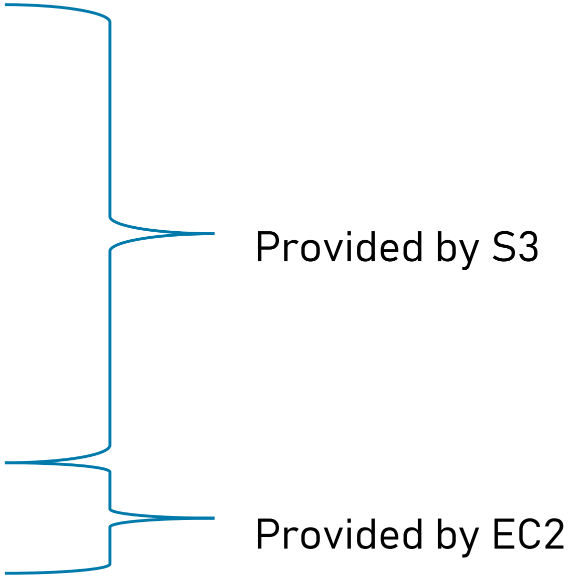
- Freemium file hosting service, started in 2007
- Customer Requirements
  - Sufficient amount of space for files
  - World-wide access
  - Fast file transfers
  - File sharing
  - Restoration of deleted files
  - Available on all customer devices

Reasons for Dropbox to Use the Cloud

- Business considerations
  - Rollout as **global service**
  - **Large upfront investment** for building data centers or depending on colocation for **storage servers** needed
    - **Cost control**
    - **Staff** needed to operate and maintain storage servers (hard drive disks)
    - Comparatively low cost for operating **metadata servers**
  - Does the company have the **knowhow** to efficiently operate large-scale file storage while being able to ensure data security?
  - Fast **time to market**
  - **Elasticity** of resources, especially for file storage and encryption
    - **Unknown number of users** and unknown growth rate
    - **On-demand** and **scalable** file storage capacity required
  - Concentrate on **core competencies**

Reasons for Dropbox to Use the Cloud

- Technical considerations
  - Constant **availability**
  - **Scalability**
  - Data security
  - **Backups** and **versioning**
  - Handle unpredictable **load peaks**
  - **Disaster recovery**
  - **Monitoring**
  - **CPU-intensive file encryption**
  - Clients for all established operating systems, web and mobile apps

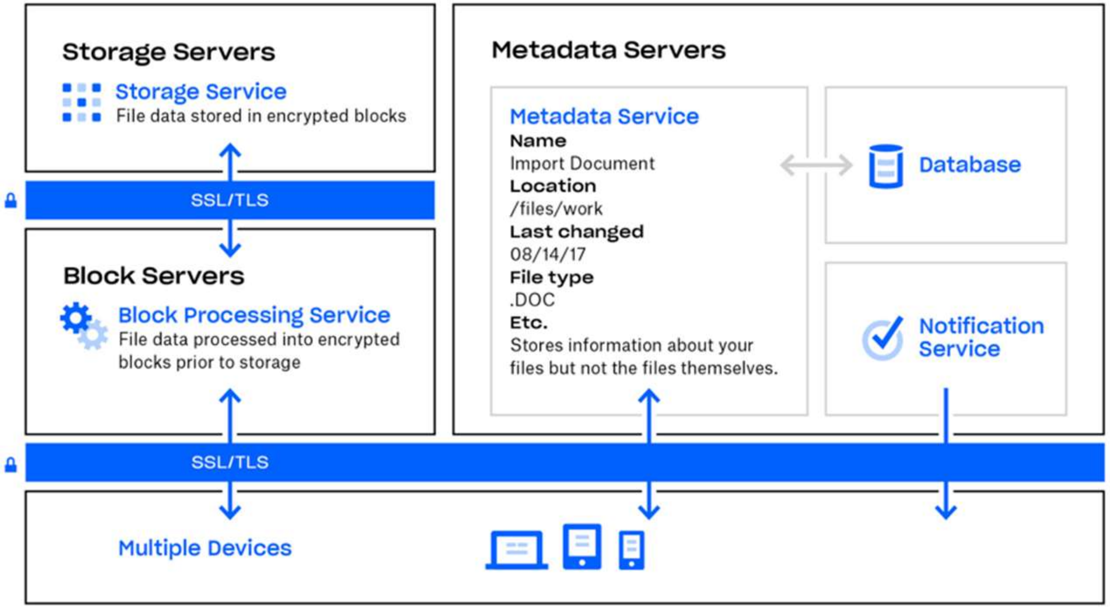


Provided by S3

Provided by EC2

Greenfield Project: Dropbox

- File storage: AWS S3 object stores
- Metadata: On-premise servers
- Focus on core competencies: **Hybrid Cloud approach**



Src: https://www.dropbox.com/business/trust/security/architecture

Dropbox – Private Cloud Transformation

Src: https://www.hpe.com/us/en/insights/articles/cloud-or-on-premises-for-dropbox-the-answer-is-yes-1702.html

- 2016: Migration of a large portion of file storage (**500 petabytes**) into private cloud
  - Avoid over-dependence on AWS, high cost-structure
  - Amazon beginning to offer similar services (WorkDocs)
  - Data Centers: Third-party subservice organization and managed service providers located in the US
- **Stayed a hybrid cloud** approach
  - S3 data storage in Germany for european business customers
- Positive financial effects
  - **$92.5 million** less direct billing to AWS in the first year
  - **$38.5 million** reduced cost of revenue in two years (reduction of 9,45%), revenue increase 83%

Src: https://www.bmc.com/blogs/dropbox-aws/

Brownfield

Brownfield Projects - Migration Strategies I

I NEED TO KNOW WHY MOVING OUR APP TO THE CLOUD DIDN'T AUTOMATICALLY SOLVE ALL OUR PROBLEMS.
YOU WOULDN'T LET ME RE-ARCHITECT THE APP TO BE CLOUD-NATIVE.
JUST PUT IT IN CONTAINERS.
YOU CAN'T SOLVE A PROBLEM JUST BY SAYING TECHY THINGS.
KUBERNETES.
Dilbert.com
@ScottAdamsSays

- **Lift & Shift**: might be possible using IaaS or CaaS services
  - Required infrastructure is rebuilt using appropriate CSP services →application is transitioned as-is with no modifications.
  - Benefits of cloud environment limited
  - Can be the starting point for cloud migration: Think of moving VMs into the cloud and replacing your current on-premise database by an automatically scaling one

Brownfield Projects - Migration Strategies II

- **Replatform**: Exchange parts of the application with appropriate counterparts from CSP
- **Refactor**: Applications designed to operate on customized infrastructure are modified to leverage available cloud services before migrating.
- **Rebuild**: Applications that are still required by the organization but cannot be modified to use available cloud-based services. These applications are redesigned and rebuilt before transitioning the process to the cloud.
  - Which **backing services** does your application need, which can be provided by cloud services?
    - Replace self-developed services by commodity services
    - Routing, load balancing, access management, databases …
    - Easiest technologies for peak loads: FaaS and PaaS
  - PaaS allows for simple scaling, but code rewrites will probably be unavoidable
  - Start by adding new cloud-based functionality to existing projects (**hybrid**)
  - Start migrating old projects by splitting between on-premise and cloud operation

Brownfield Projects - Migration Strategies III

- **Retire**: Applications that are no longer operationally or economically viable to the organizations.
- **Repurchase**:
  Associated processes are either eliminated or replaced with available SaaS.
- **Retain**: Application or processes and data are kept on premise as they meet their requirements or are not appropriate for cloud migration due to legal/organizational reasons

Use Case: Multiview SaaS-ification

Src: https://multiviewcorp.com

Multiview Financials ERP

- Starting point
  - Server application has to be operated on customer side
  - Only Windows client software available
  - Grown software: Company and product exist for nearly three decades
- Customer Requirements
  - High demand for a SaaS **cloud version**
    - Usage without operating an on-premise server
  - High demand for **web-based access**
    - Usage on all established operating systems and mobile devices
  - Desirable extras:
    - Fast access to data
    - Same application version for all users
    - Remove software rollout issues
    - World-wide access

Src: https://aws.amazon.com/solutions/case-studies/multiview/

Reasons for Multiview to Use the Cloud

Business considerations:

- Development of a web-native application
  - Estimated **costs of multiple millions of dollars**
  - Estimated **time to market of four years**
  - Significant effort for securing the service
  - Company has/had its **core competencies** in developing Windows applications
  - **Parallel development** of two applications necessary
- SaaS model implies further questions
  - Where and how to operate application servers?
  - Total cost of ownership?
  - How many existing customers want to transfer to the SaaS model?

→ **Rebuilding** the software as web application with SaaS service model **unfeasible**

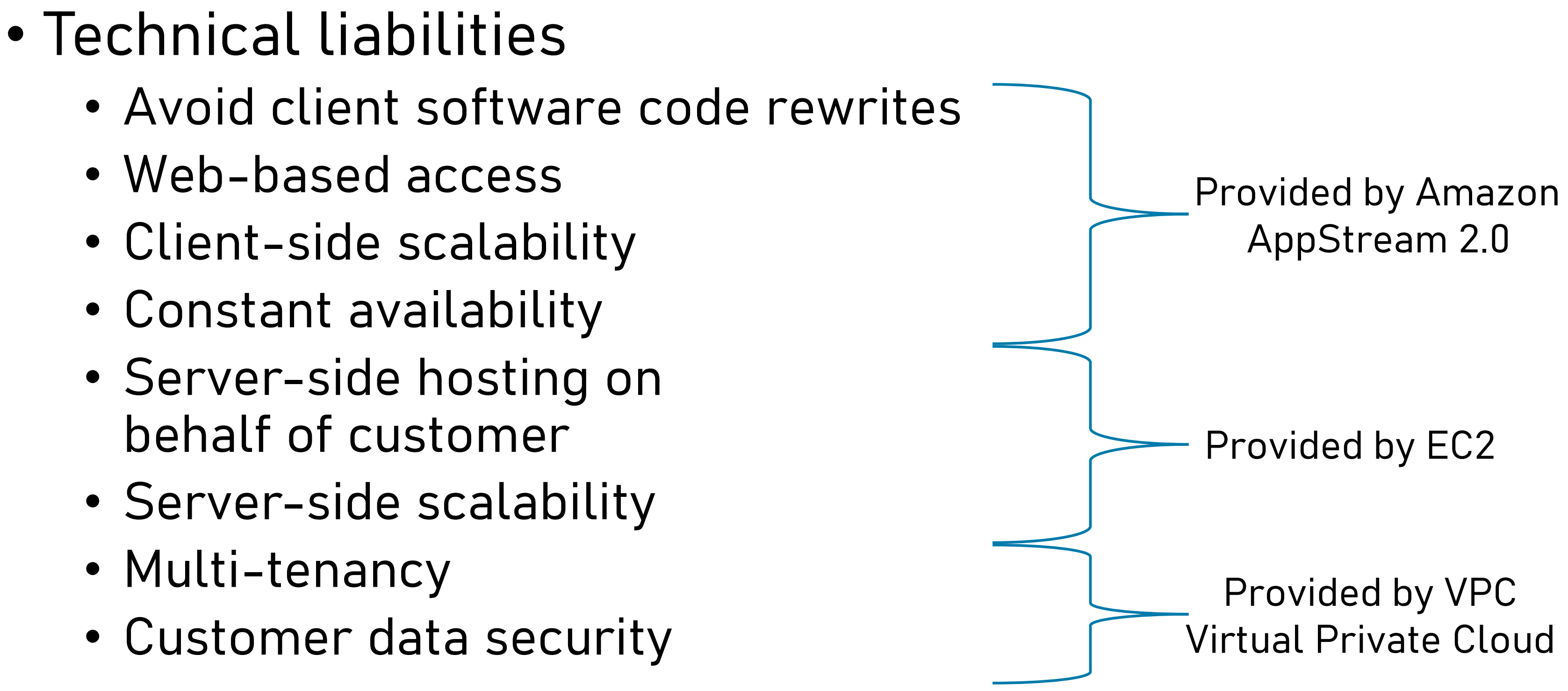
Src: https://aws.amazon.com/solutions/case-studies/multiview/

Amazon AppStream 2.0

Fully managed **streaming application server**

- Can stream **any Windows application**
- Functionality
  - **Central definition** of an **application stack** using the **Image Builder**
  - Definition of a **fleet** (VM instances that all run the same application stack)
    - **On-Demand** (1-2 minute startup time, lower instance fees while not streaming)
    - **Always-on** (no startup time, constant instance fees)
  - **Adaptable** to special workloads (e.g. instances with GPUs)
  - For each active user one fleet instance is required
  - **Limited scaling**: Maximum number of instances must be set in advance
- Running in a **Virtual Private Cloud** (AWS VPC)
- **Optional user home directories**, provided by S3 file storage
- Users access applications via **Windows desktop client** or **browser**

Rough Technology Mapping



- Technical liabilities
  - Avoid client software code rewrites
  - Web-based access
  - Client-side scalability
  - Constant availability
  - Server-side hosting on behalf of customer
  - Server-side scalability
  - Multi-tenancy
  - Customer data security

Provided by Amazon AppStream 2.0

Provided by EC2

Provided by VPC
Virtual Private Cloud

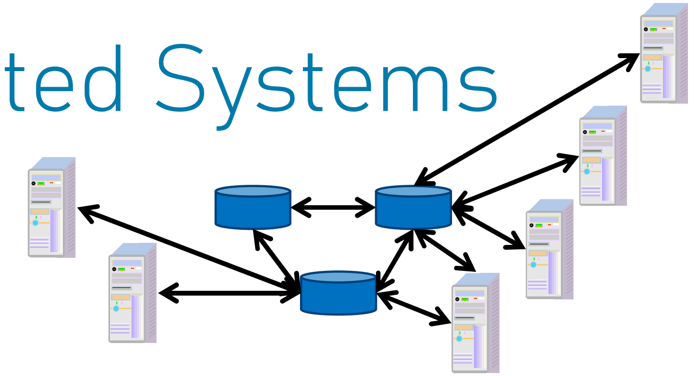
Outline

1. What is the Cloud?
2. Cloud Service Models
3. Why move / move not into the Cloud?
4. **Software Development for the Cloud**

“A distributed system is one
in which the failure of a computer you didn't even know existed
can render your own computer unusable.”

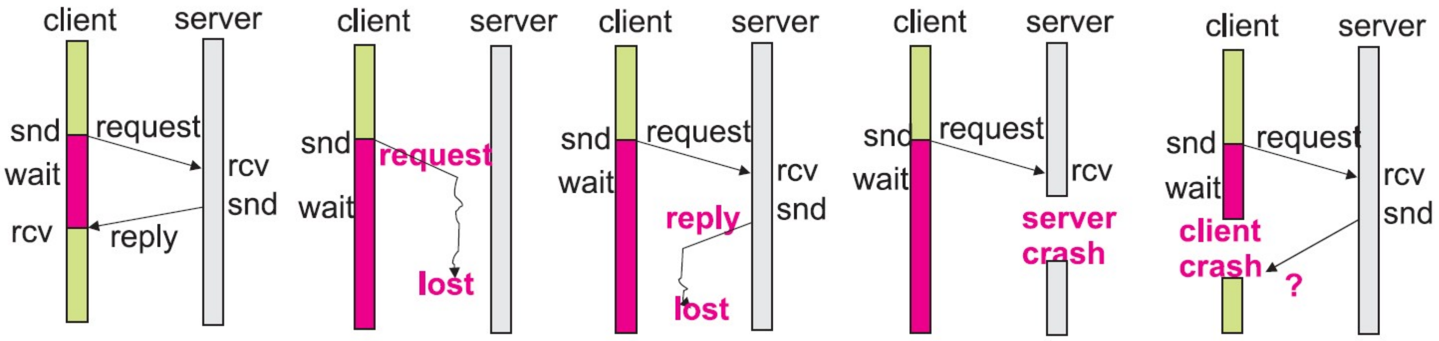
Leslie Lamport

Cloud Systems are Distributed Systems



- Definition:
  "A distributed system is a collection of independent computers that appears to its users as a single coherent system."
  (Tanenbaum, *Distributed Systems*, Prentice Hall, 2016)

  "We define a distributed system as one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages."
  (Coulouris, Dollimore, Kindberg, *Distributed Systems*, Addison Wesley, 2011)



Implications for Architecture

Loose Coupling

Prepare for independent changes and sudden failure.

Service Orientation

Derive proper service cut with clear interfaces.

Hide internal implementation.

Web-Based

Support multiple client platforms as needed.

Take care of security.

Implications for Development

Agility

Prepare for constant change, most notably automate integration and test.

Everything as Code

Automate infrastructure setup.

Cattle not pets!

OpEx

Capital Expenditure largely replaced by Operational Expenditure.

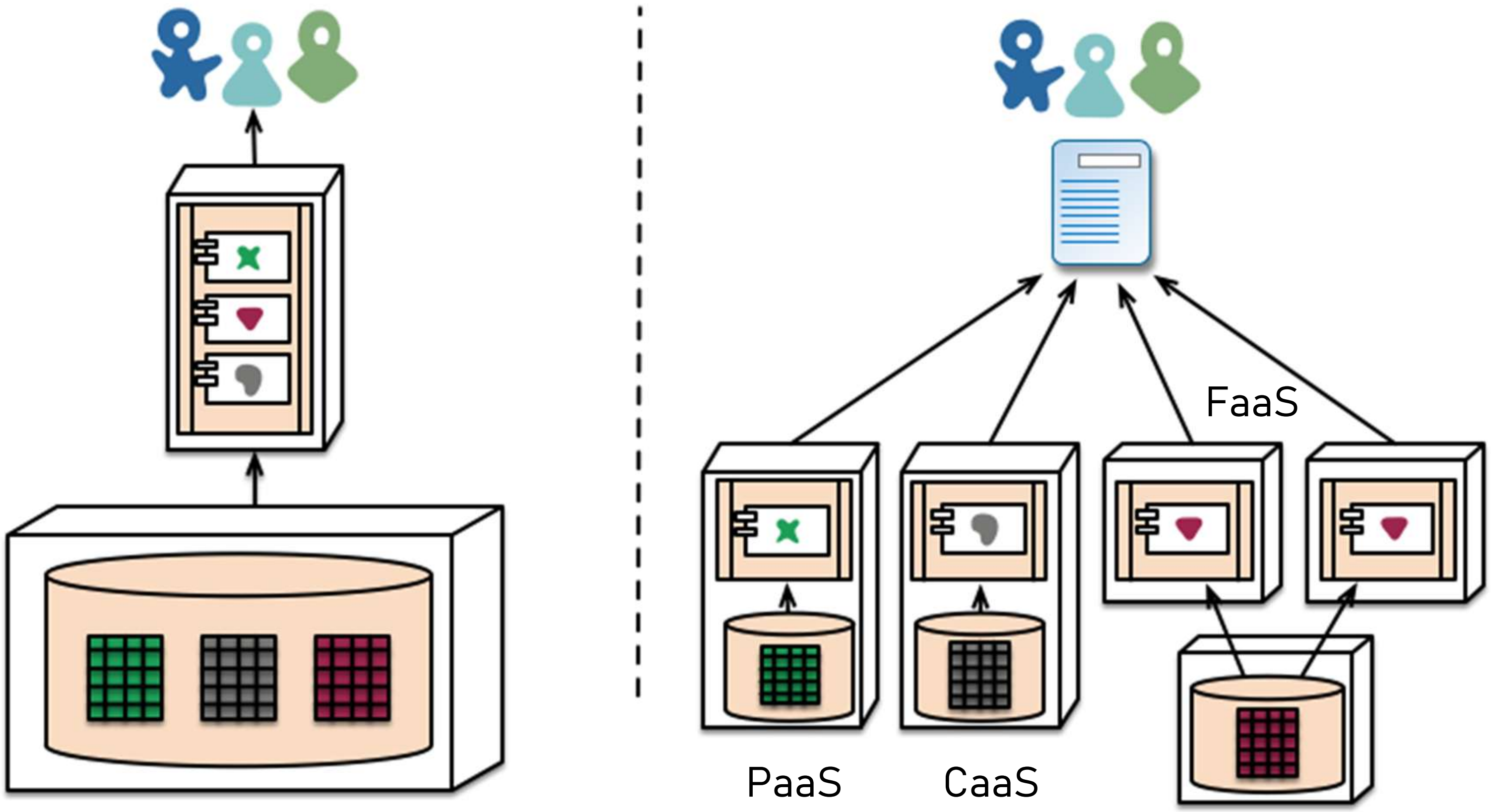
Microservices

Microservices are an architectural **paradigm** that pays tribute to modern (cloud-based) application development

- Core Characteristics of Microservices
  - **Componentization** via **decentralized services**, organized around **Business Capabilities**
    - Created by **cross-functional teams**
  - **Products**, not projects
    - **Permanent** involvement of developers in application operation
  - **Infrastructure Automation**
  - **Evolutionary Design**
  - **Design for failure**
    - Smart endpoints and dumb pipes
    - Self-healing capabilities
  - … can be written in different **programming languages**
  - … persist their **business data decentrally** using **suitable data stores** as appropriate

Src: https://martinfowler.com/articles/microservices.html

Microservices – Decentralized Data

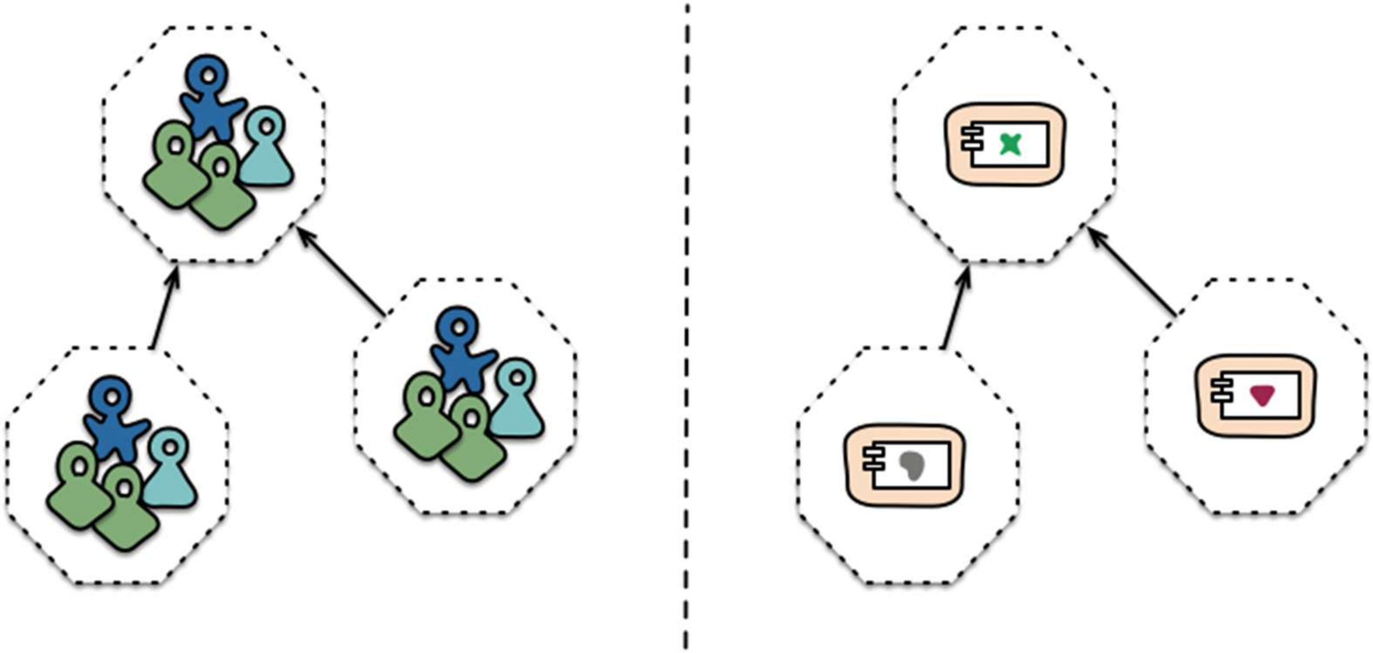


monolith – single database

microservices – application databases

Src:
https://martinfowler.com/articles/microservices.html

Microservices –
Organized Around Business Capabilities



Cross-functional teams…

… organized around capabilities
Because Conway’s Law

Src:
https://martinfowler.com/articles/microservices.html

Microservices - Products, not Projects

Project Model:
Development team develops application that is later operated by a different team

Product Model:
“You build it, you run it” (Werner Vogels)

Amazon reports that assigning responsibility of the software in production to development teams greatly improves quality and increases understanding of the domain
➔ Clearly calls for the product model, nowadays DevOps
(May be hard to realize in embedded / automation settings!)

Src:
https://martinfowler.com/articles/microservices.html

A Mental Model for Industrial Cloud Apps

Industrial Systems Development

Edge Connectivity

Cloud Development

Continuous Integration & Delivery

Thank you!

www.lion5.io