CLOUD-NATIVE COMPUTING

Unit 01:

Grundlagen des Cloud Computing

Stand: 27.03.23

KAPITEL 2

Cloud Computing

Nane Kratzke

Cloud-native Computing

Software Engineering von Diensten und Applikationen für die Cloud

284 Seiten. E-Book inside

€ 59,99. ISBN 978-3-446-46228-1

Weitere Informationen unter: **www.hanser-fachbuch.de** HANSER

CLOUD NATIVE COMPUTING

creative commons

CLOUD NATIVE COMPUTING

Urheberrechtshinweise

Diese Folien werden zum Zwecke einer praktikablen und pragmatischen Nutzbarkeit im Rahmen der **CCo 1.0 Lizenz** bereitgestellt.

Sie dürfen die Inhalte also kopieren, verändern, verbreiten, mit eigenen Inhalten mixen, auch zu kommerziellen Zwecken, und ohne um weitere Erlaubnis bitten zu müssen.

Eine Nennung des Autors ist nicht erforderlich (aber natürlich gern gesehen, wenn problemlos möglich).

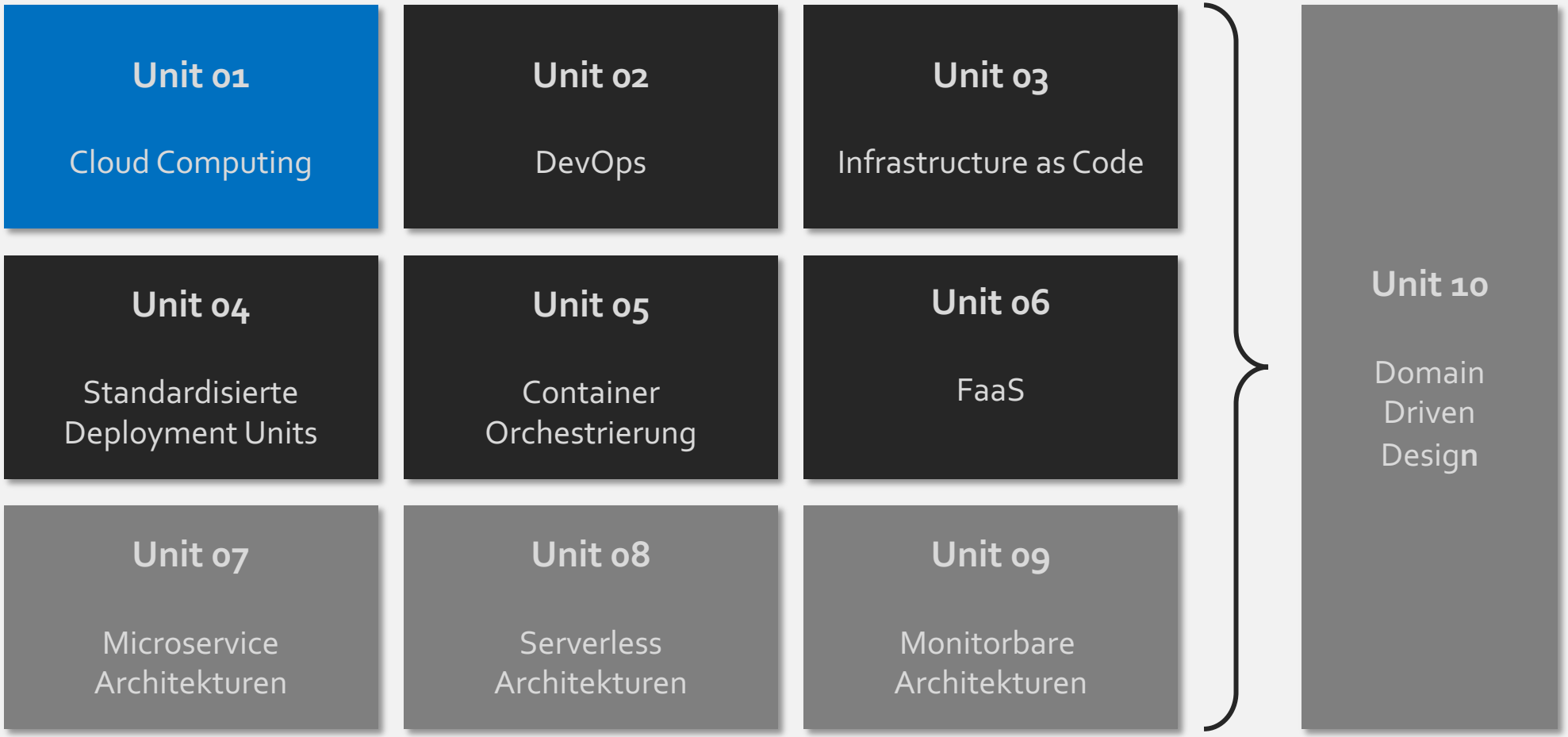
Diese Folien sind insb. für die Lehre an Hochschulen konzipiert und machen daher vom **§51 UrhG (Zitate)** Gebrauch.

Die CCo Lizenz überträgt sich nicht auf zitierte Quellen. Hier sind bei der Nutzung natürlich die Bedingungen der entsprechenden Quellen zu beachten.

Die Quellenangaben finden sich auf den entsprechenden Folien.

INHALTSVERZEICHNIS

Überblick über Units und Themen dieses Moduls



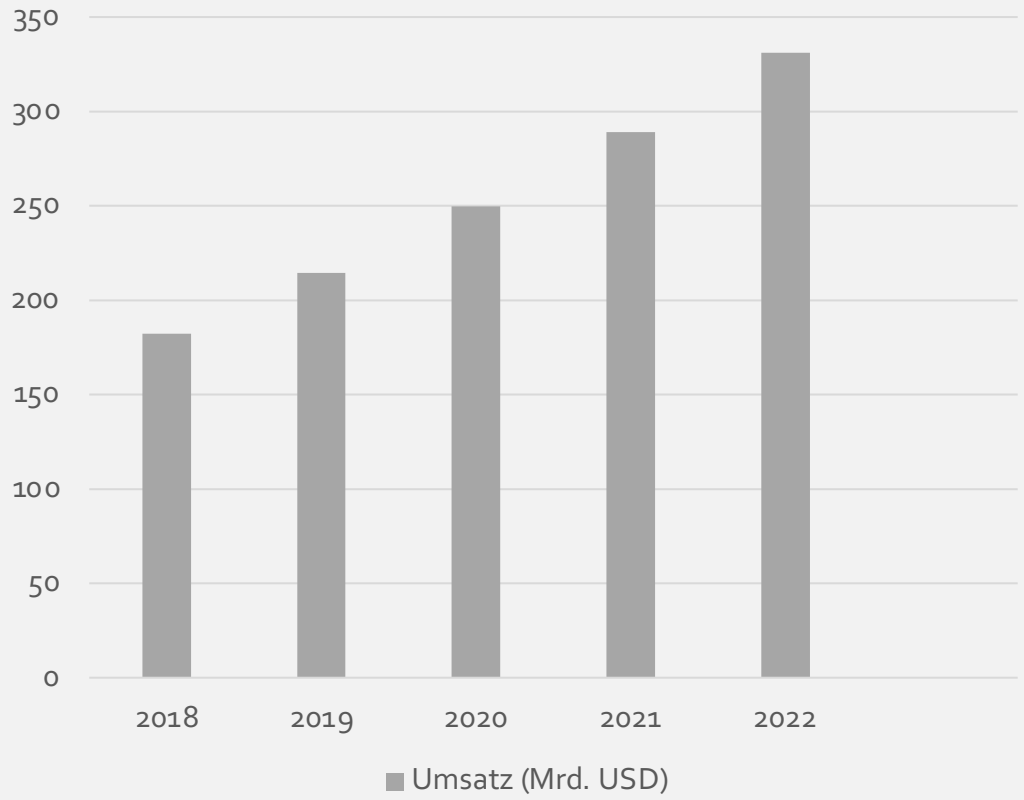
INHALTE

- Cloud Computing
- Cloud-Service Modelle
- Cloud-Ökonomie (Pay-as-you-Go)
- Eine kurze Architekturgeschichte der Cloud
- Cloud-native Anwendungen und Dienste
- Zusammenfassung

WACHSTUMSMARKT CLOUD-COMPUTING

*70% des Marktes teilen sich die sogenannten **Big-Five***

*„Laut Gartner wird der weltweite Markt für Public Cloud Services 2019 um 17,5 Prozent auf insgesamt 214,3 Milliarden Dollar wachsen […]. Das am **schnellsten wachsende Marktsegment** wird Infrastructure as a Service **(IaaS)** sein[…]. Die **zweithöchste Wachstumsrate** von 21,8 Prozent wird durch […] Platform as a Service **(PaaS)** erreicht."*



CLOUD NATIVE COMPUTING

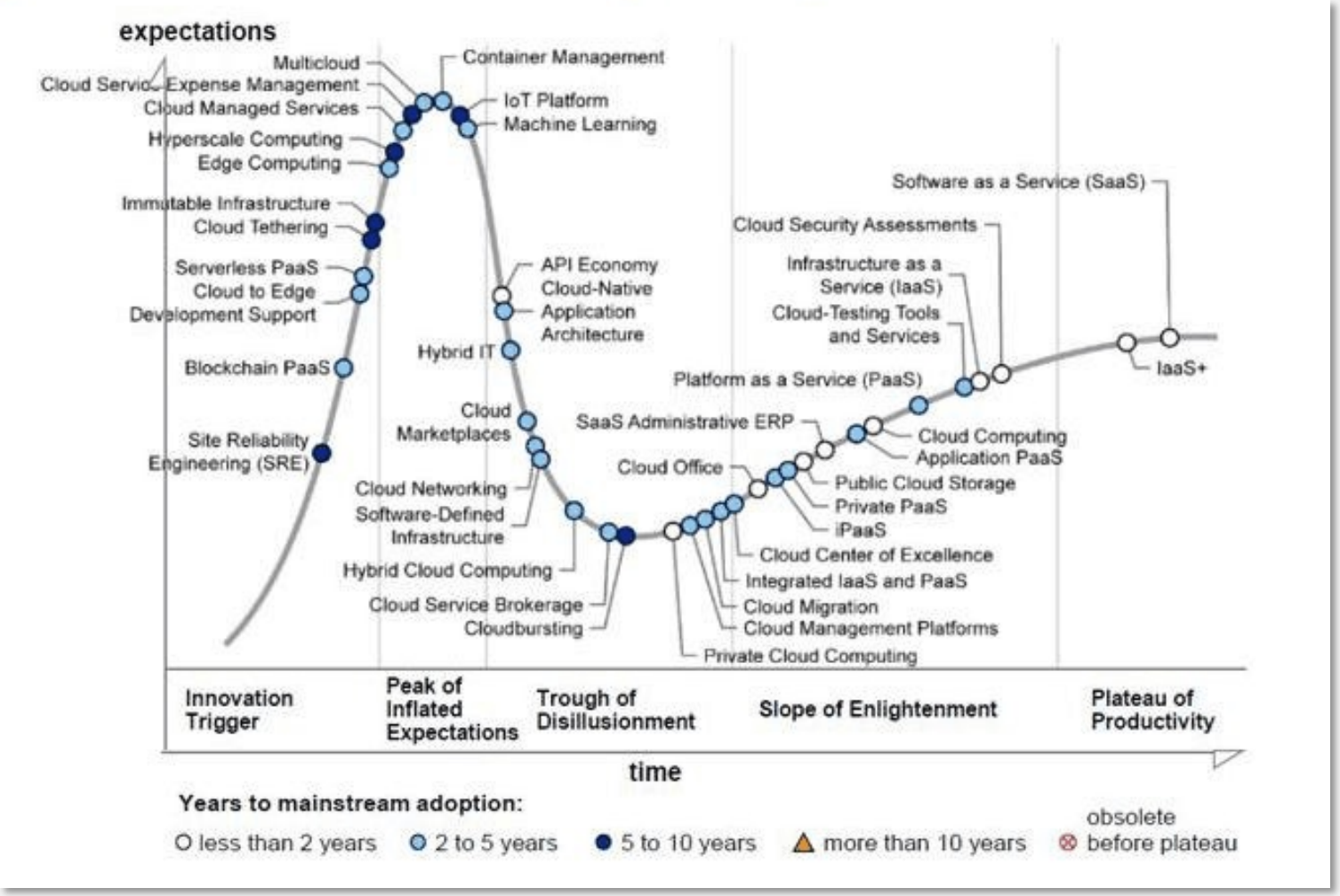
Die Entwicklung ist zwar beeindruckend, aber immer noch linear und nicht exponentiell wie manchmal behauptet wird.

BIG-FIVE:

- Amazon
- Microsoft
- Alibaba
- Google
- IBM

CLOUD COMPUTING HYPE CYCLE

Gartner, 2018

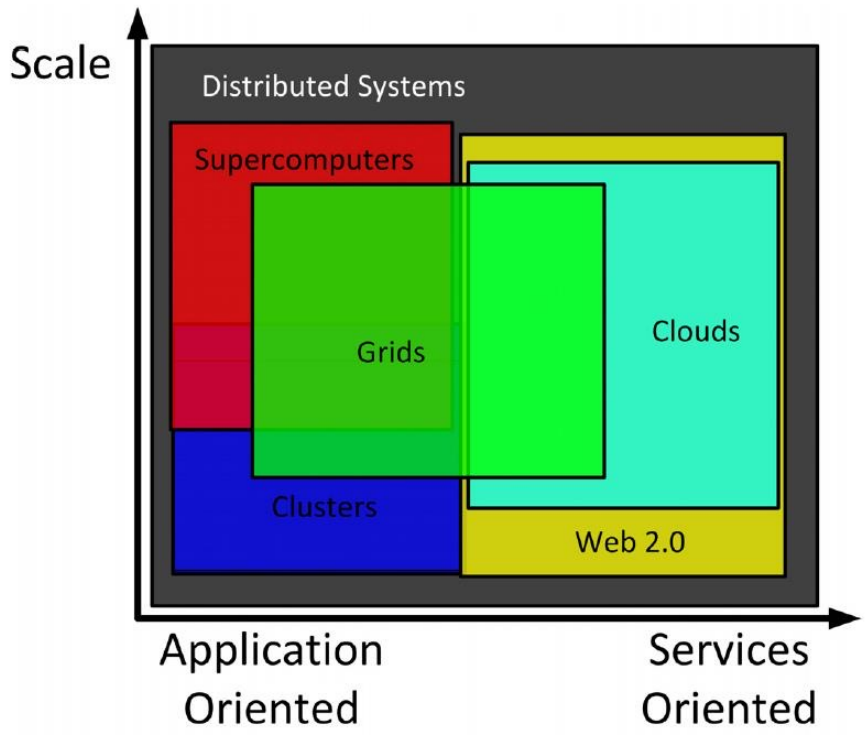


„In Zeiten des digitalen Wandels ist Cloud Computing heute die primäre Option und nicht mehr nur eine von vielen Möglichkeiten"

Gartner-Analyst Gregor Petri

CLOUD COMPUTING

Im Vergleich zu anderen Ansätzen Verteilter Systeme



Die 5 Gebote der Cloud

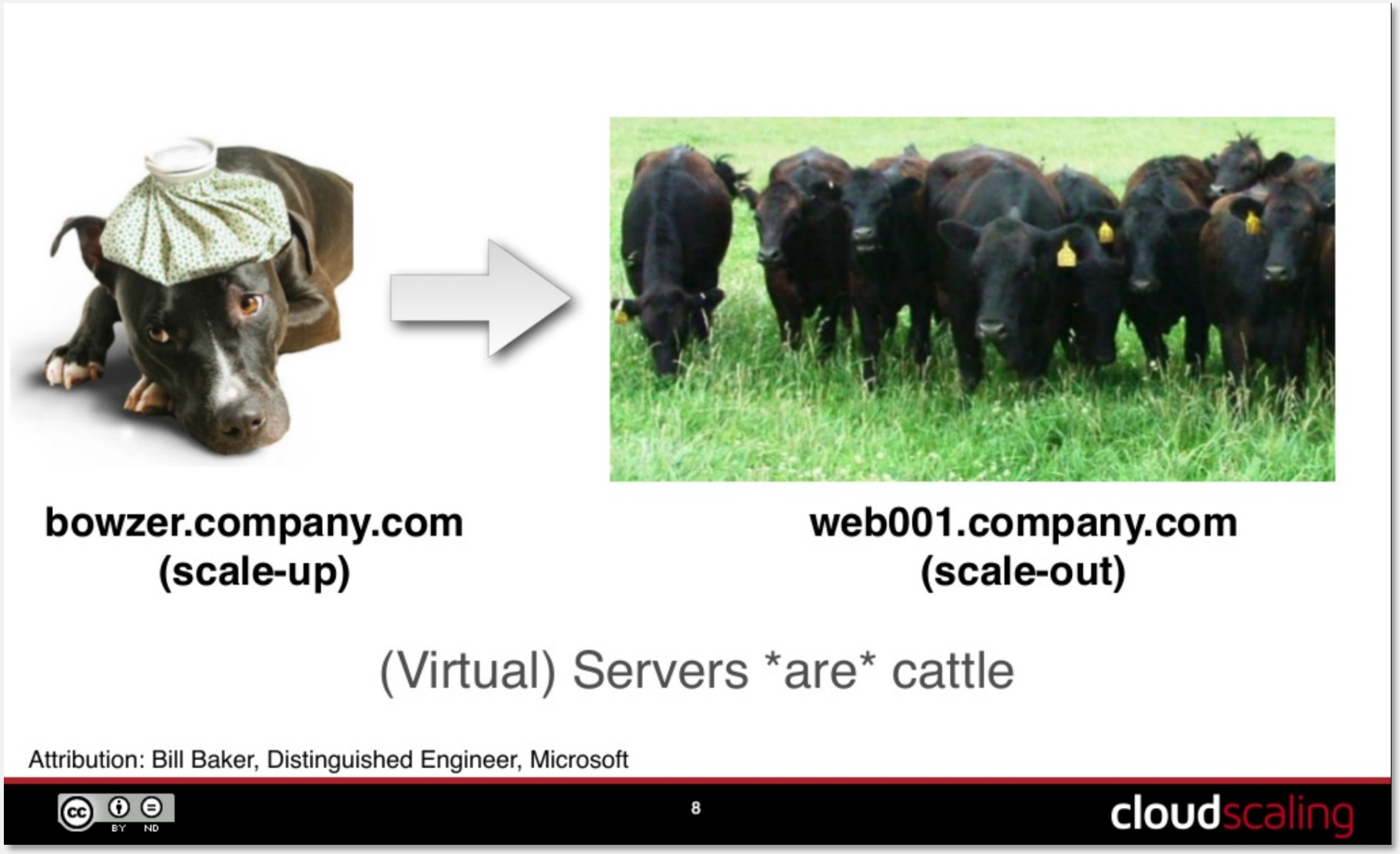
1. Everything fails all the time
2. Focus on MTTR and not on MTTF
3. Respect the Eight Fallacies of Distributed Computing
4. Scale out, not up
5. Treat ressources as cattle, not pets

MTTR = Mean Time to Repair

MTTF = Mean Time to Failure

PETS VS. CATTLE

Cloud Computing is a Mindset Change



CLOUD NATIVE COMPUTING

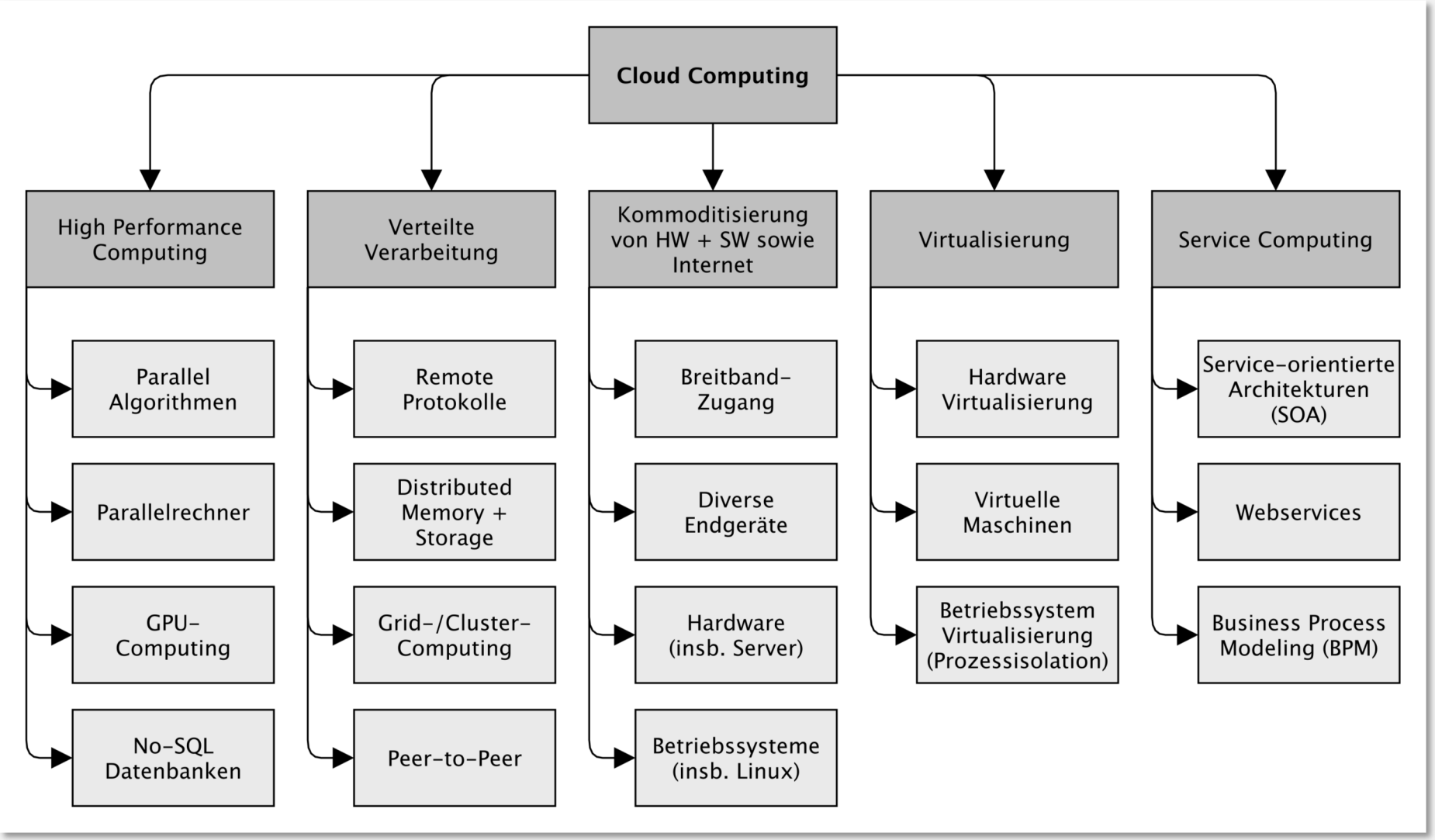
Alte Informatiker-Weisheit:
Ausschalten, Einschalten, geht wieder.

Server sind Wegwerf-Artikel!
Wegschmeißen - neu machen, ist schneller als Problem suchen, Problem beheben.

Mehr Power,
kommt durch mehr Server.

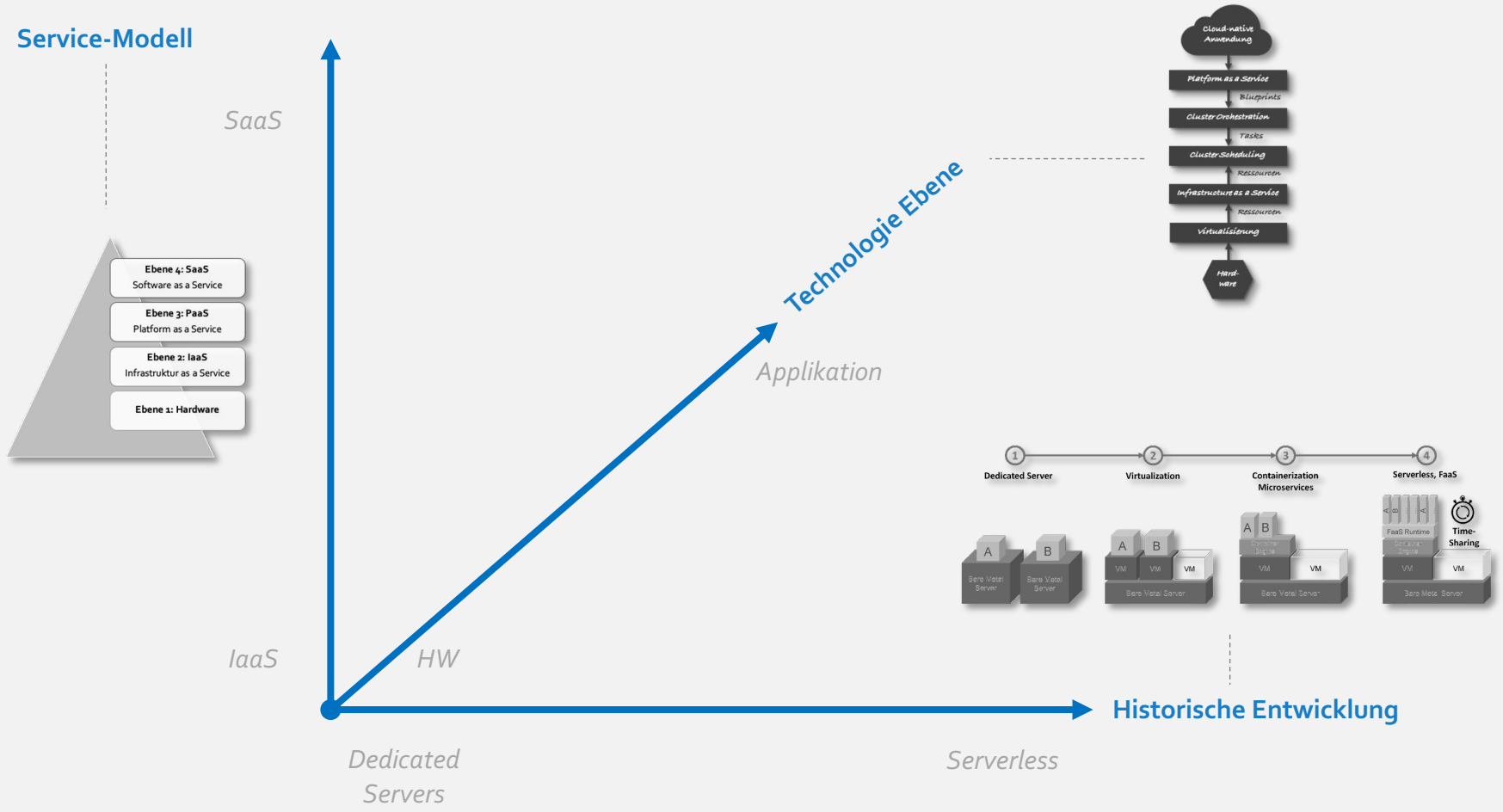
CLOUD COMPUTING

Auf den Schultern von Giganten



ORIENTIERUNG IN DIESEM MODUL

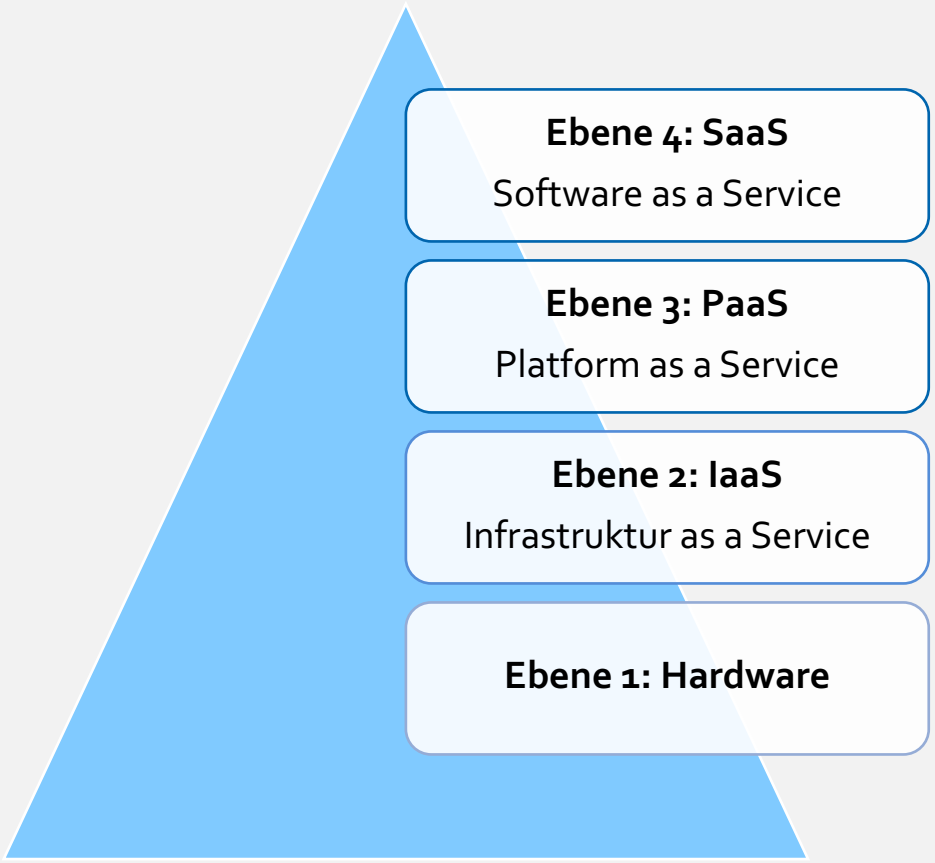
Bezugssystem



Hinweis:
Dazu kommt noch eine architekturelle Ebene. Diese wird aber im Schwerpunkt im Folgemodul „Cloud-native Architekturen" behandelt werden.

DAS SCHICHTENMODELL DES CLOUD COMPUTINGS

Service - Modelle



Kunden, Endnutzer
- Anpassbare Software-Dienste
- XaaS (Everything as a Service)
- Transparente Updates

Entwickler
- Programmierschnittstellen (APIs)
- Plattformdienste
- Abstraktion der technischen Infrastruktur

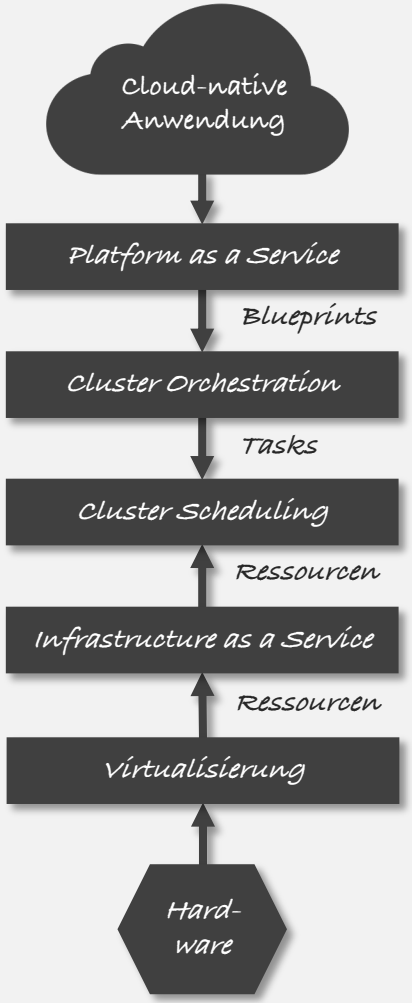
Administratoren
- Elastizität
- Virtuelle Ressourcenpools
- Technische Infrastruktur (VM, Storage, Network)

Rechenzentrum
- Rechner
- Netzwerk
- Storage

Diese Pyramide werden wir uns nach oben durcharbeiten.

TECHNOLOGIE - EBENEN

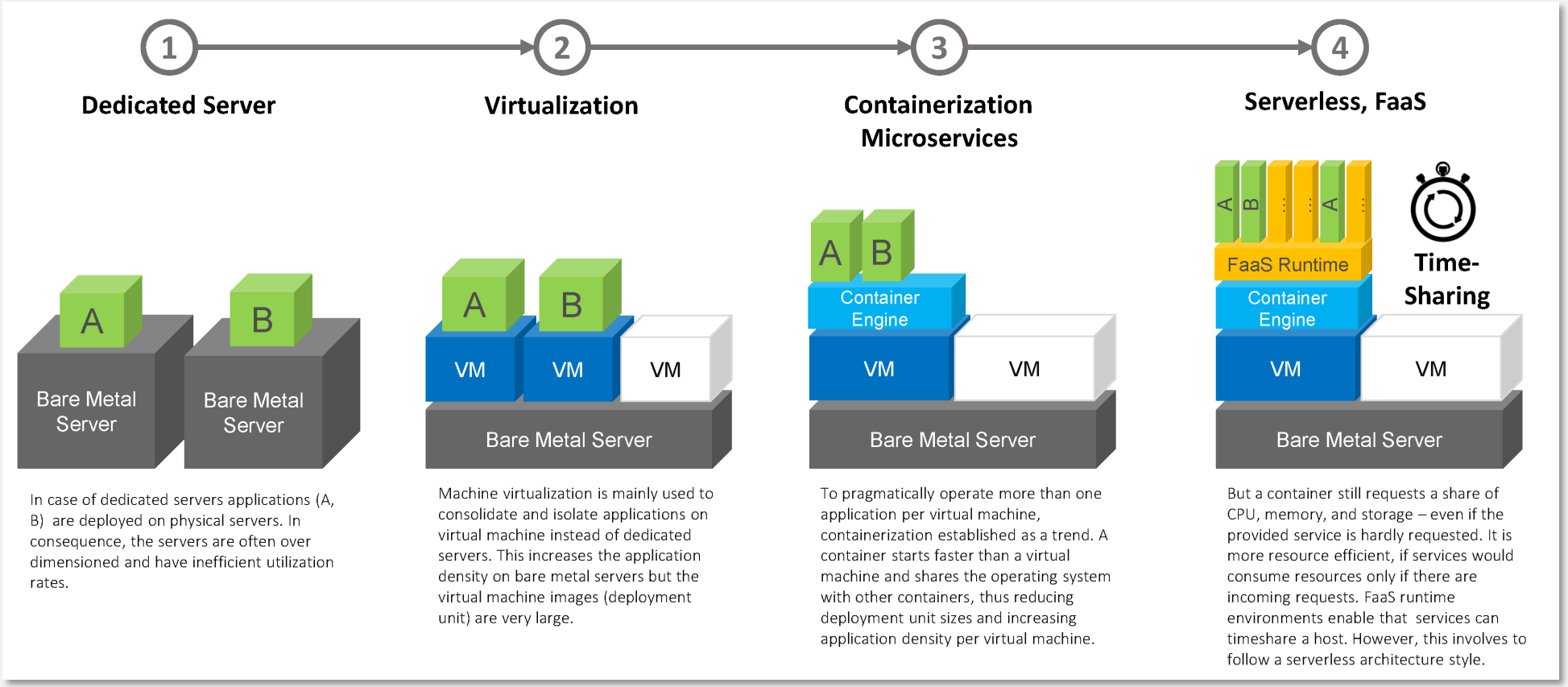
Oder: Wie kommt die Software an das Blech?



Diese Ebenen werden wir aus verschiedenen Blickwinkeln beleuchten.

HISTORISCHE ENTWICKLUNG

Eine kurze Geschichte der Cloud



Quelle: Kratzke, N. A Brief History of Cloud Application Architectures. Appl. Sci. 2018, 8, 1368.

CLOUD NATIVE COMPUTING

Im Verlaufe der Zeit sind die Größen von Deployment Units geschrumpft!

VM -> Container -> Function

Parallel dazu haben sich Servic-orientierte Architekturen (weiter)entwickelt.

Microservices + Serverless Architectures

INHALTE

- Cloud Computing
- Cloud-Service Modelle
- Cloud-Ökonomie (Pay-as-you-Go)
- Eine kurze Architekturgeschichte der Cloud
- Cloud-native Anwendungen und Dienste
- Zusammenfassung

CLOUD COMPUTING

The NIST Definition of Cloud Computing

Cloud computing is a model for enabling

- ubiquitous,
- convenient,
- on-demand network access

to a **shared pool of configurable computing resources** that can be

- rapidly provisioned and released
- with minimal management effort or service provider interaction.

https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

CLOUD NATIVE COMPUTING

This cloud model is composed of

- *five essential characteristics*
- *three service models,*
- *and four deployment models.*

Resources: *e.g., networks, servers, storage, applications, and services*

CLOUD COMPUTING

Essential Characteristics

CLOUD NATIVE COMPUTING

On-demand self-service:

A consumer can provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with the cloud service provider.

Broad network access:

Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous client platforms.

Rapid elasticity:

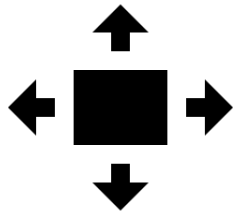
Capabilities can be elastically provisioned and released to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

Measured service:

Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

Resource pooling:

The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. The customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter).



CLOUD COMPUTING

Service Models

CLOUD NATIVE COMPUTING

Software as a Service (SaaS)

The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface.

The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings

Platform as a Service (PaaS)

The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider.

The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

Infrastructure as a Service (IaaS)

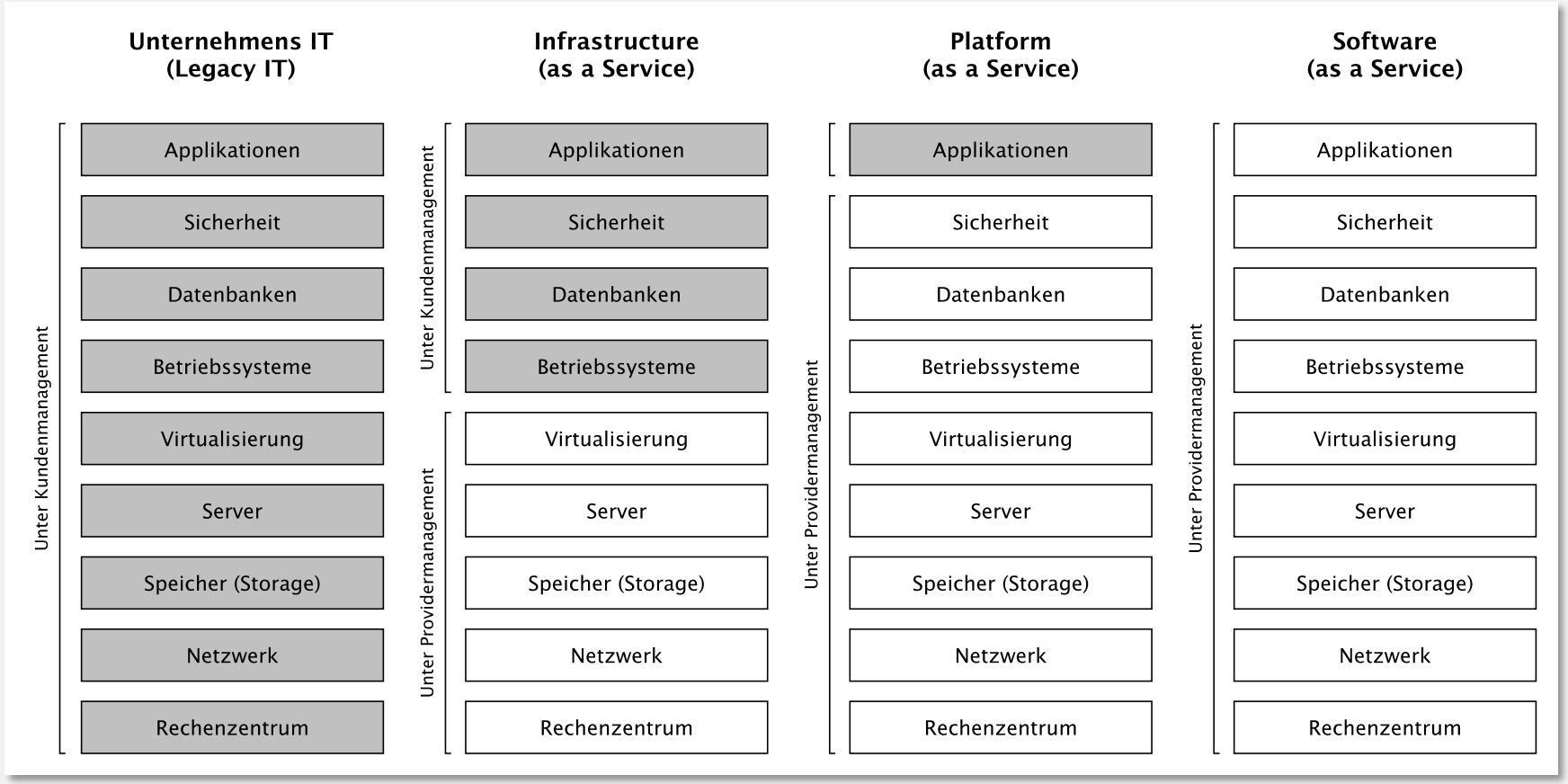
The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications.

The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

CLOUD-SERVICE MODELLE

Customer Managed vs. Provider Managed Services

sinkender Lock-In, steigende Betriebskomplexität



steigender Lock-In, sinkende Betriebskomplexität

CLOUD NATIVE COMPUTING

Mittels Cloud-Computing lassen sich Teile der IT-basierten Wertschöpfung an externe Dienstleister (Cloud-Provider) auslagern.

Von IaaS über PaaS zu SaaS wird dieser ausgelagerte Anteil dabei immer gröber (ebenso wie die Gewinnmargen der Provider).

Alle Service-Modelle folgen dabei denselben wirtschaftlichen Gesetzmäßigkeiten.

CLOUD COMPUTING

Deployment Models

Private cloud

The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

Community cloud

The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

Public cloud

The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

Hybrid cloud

The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

CLOUD NATIVE COMPUTING

Hinweis:

Wenn von Cloud Computing gesprochen wird, wird häufig Public Cloud Computing gemeint, ohne dies explizit zu sagen.

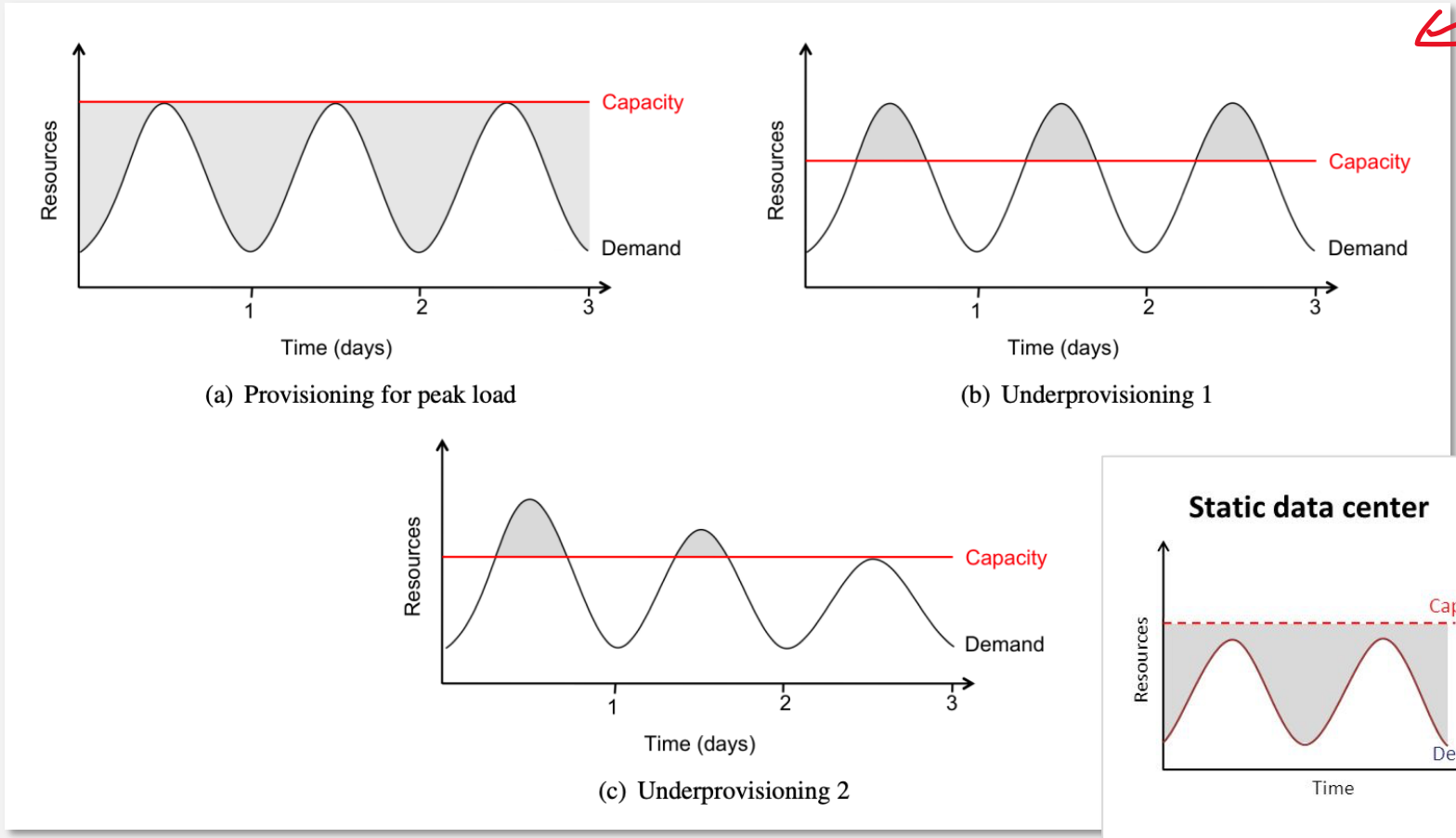
Vendor Lock-In Bedenken werden häufig durch Private Cloud Ansätze beantwortet (auch wenn das wirtschaftlich nicht immer sinnvoll ist).

INHALTE

- Cloud Computing
- Cloud-Service Modelle
- Cloud-Ökonomie (Pay-as-you-Go)
- Eine kurze Architekturgeschichte der Cloud
- Cloud-native Anwendungen und Dienste
- Zusammenfassung

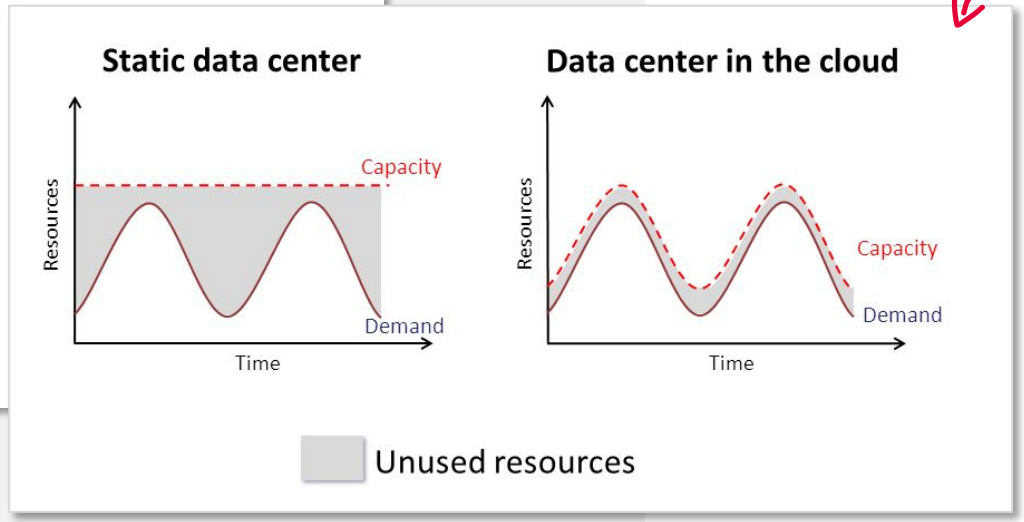
PAY-AS-YOU-GO

Nur für das Zahlen was man benötigt …



Das Dimensionierungs-problem in „klassischen" On-Premise Rechenzentren.

Cloud Computing ermöglicht Lastkurven enger zu folgen (und Ressourcen zu sparen).



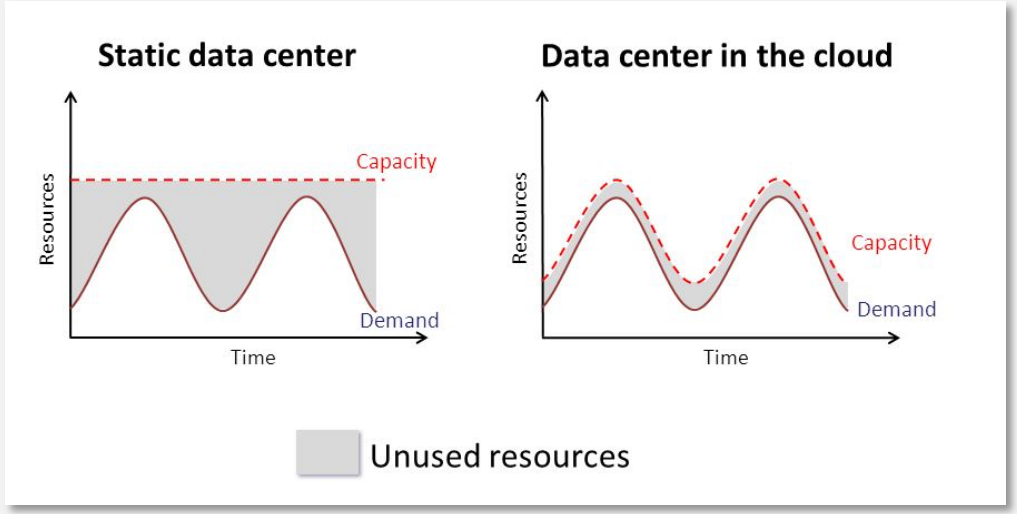
Quelle: Above the Clouds: A Berkley View on Cloud, Computing, Feb. 2009

CRASHKURS IN CLOUD-ÖKONOMIE

Ist Cloud-Computing immer billiger?

Cloud-Ressourcen sind vor allem dann wirtschaftlich, wenn Lastschwankungen in einem Anwendungsfall auftreten.

Die Kosten pro Cloud-Ressource können sogar deutlich höher als die In-house Kosten liegen – solange das Verhältnis von Cloud zu In-house Kosten nicht das Verhältnis von Spitzen- zu Durchschnittslast übersteigt.



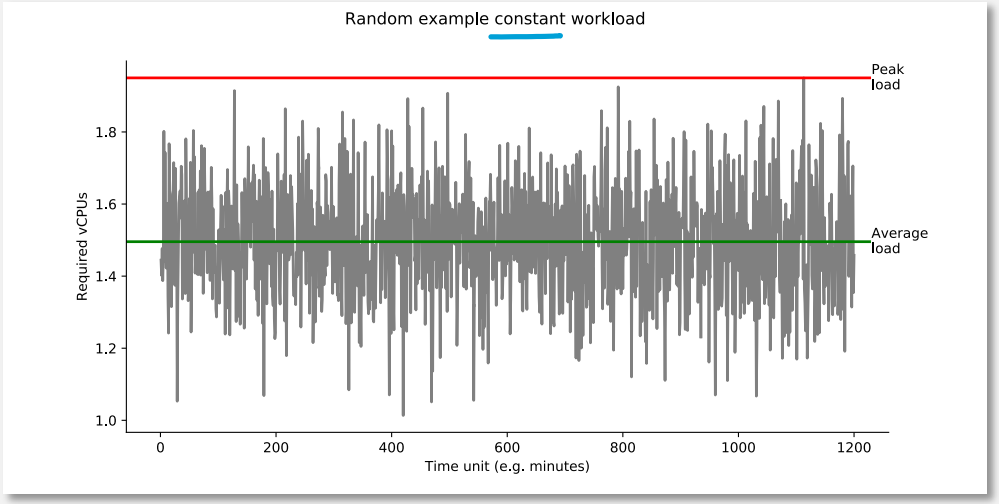
$$\frac{c}{d} < \frac{p}{a} \Leftrightarrow c < d\frac{p}{a}$$

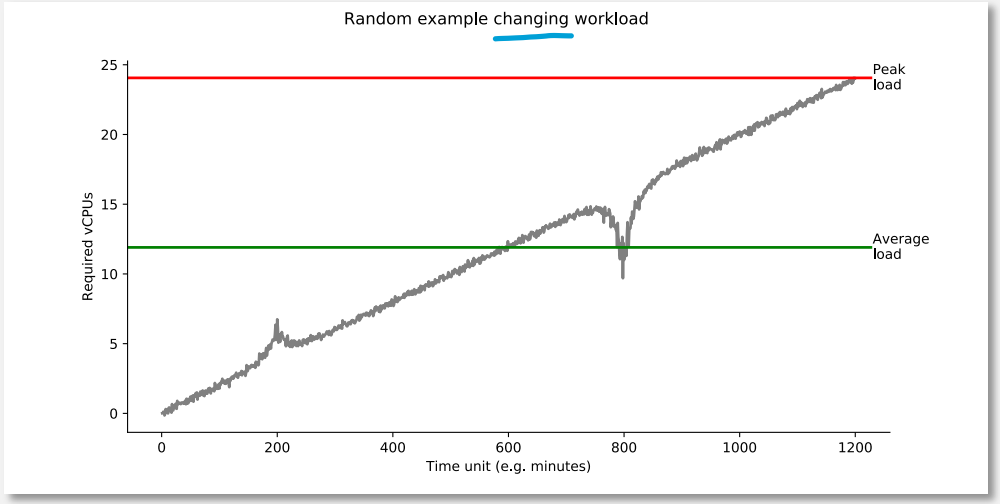
- d In-house Aufwand
- c Cloud-Kosten
- a Durchschnittslast (average)
- p Spitzenlast (peak)

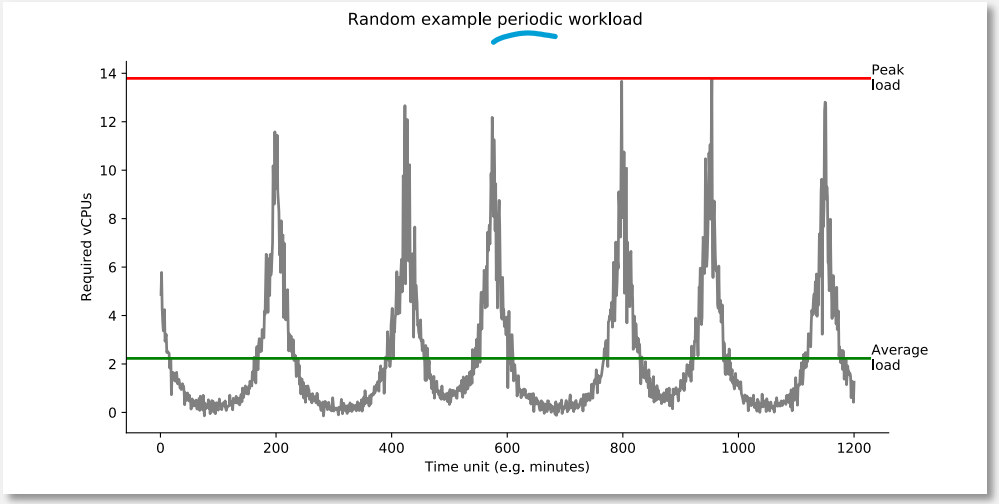
Quelle: J. Weinman, *Mathematical Proof of the Inevitability of Cloud Computing*, Jan. 2011
https://cloud-native-computing.de/materials/Joe_Weinman_Inevitability_Of_Cloud.pdf

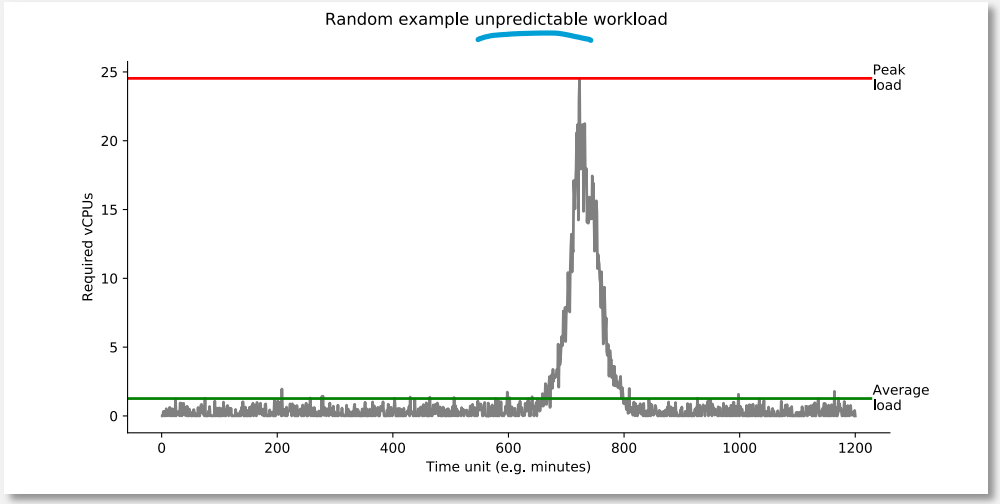
WORKLOAD KATEGORIEN

Constant, Changing, Periodic, Unpredictable (Once-in-a-lifetime)







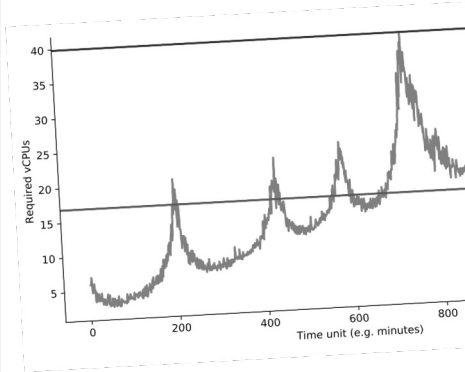


CLOUD NATIVE COMPUTING

Diese Workload Kategorien werden im Cloud Computing normalerweise unterschieden.

Sie treten im „echten Leben" natürlich selten in Reinform auf.

Die meisten Workloads setzen sich aus diesen vier Grundtypen zusammen.



PIZZA-AS-A-SERVICE

Ein Beispiel zur Veranschaulichung der Gesetzmäßigkeiten der Cloud-Ökonomie

Legacy Service	IaaS	PaaS	SaaS
Esstisch	Esstisch	Esstisch	Esstisch
Getränke	Getränke	Getränke	Getränke
Ofen	Ofen	Ofen	Ofen
Belag	Belag	Belag	Belag
Tomatensauce	Tomatensauce	Tomatensauce	Tomatensauce
Pizzateig	Pizzateig	Pizzateig	Pizzateig
Zutaten	Zutaten	Zutaten	Zutaten
Pizza di Mama	Kaufen & Backen	Pizza Service	Pizzeria

CLOUD NATIVE COMPUTING

selbst

fremd

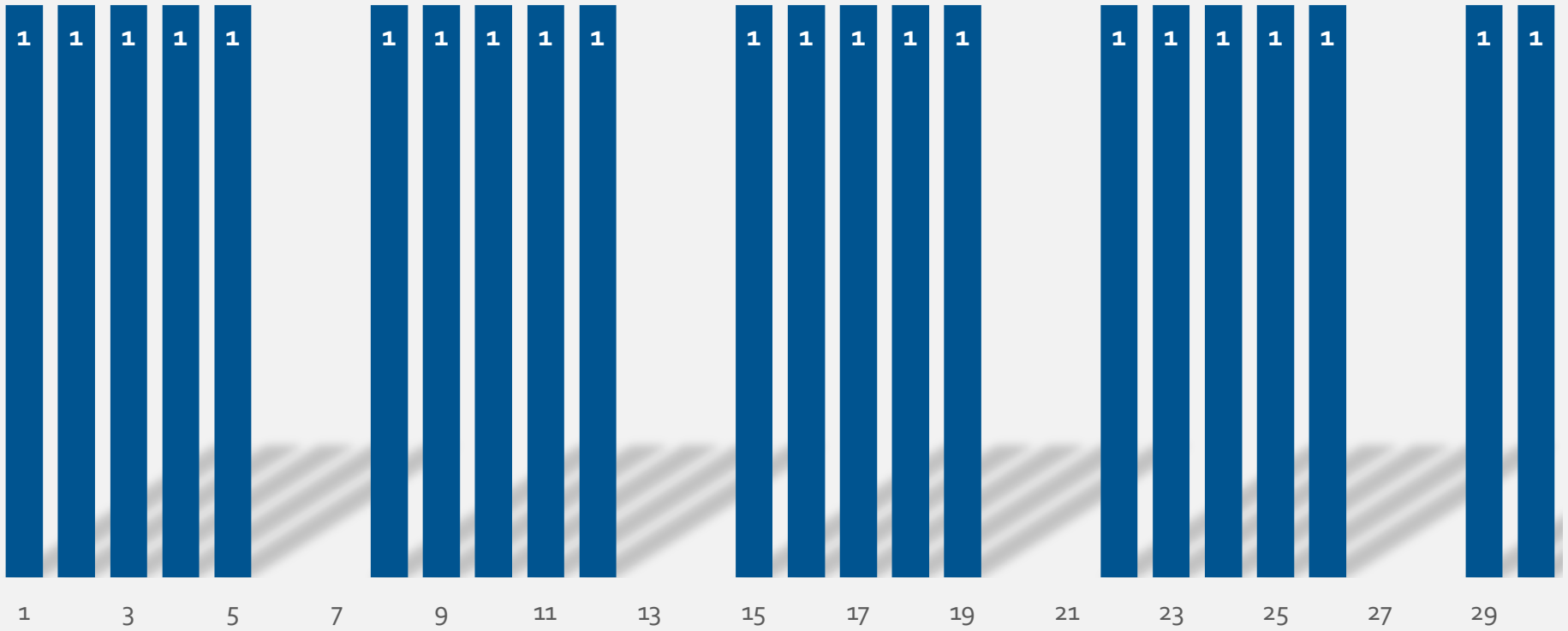
Wir fragen uns nun, ob es Pizzakonsum-Gewohnheiten geben könnte, die Service-Modelle wirtschaftlich vorteilhafter erscheinen lassen könnten.

$$\frac{c}{d} < \frac{p}{a} \Leftrightarrow c < d\frac{p}{a}$$

STATISCHE WORKLOADS

Der regelmäßige „Fastfood" Workload

Der Cloud-Provider kann bis zu 30% teurer sein als Ihre selbstgemachte Pizza.



Sie kaufen sich an jedem Werktag zur Mittagszeit eine Pizza am Stand gegenüber vor Ihrer Arbeitsstelle.

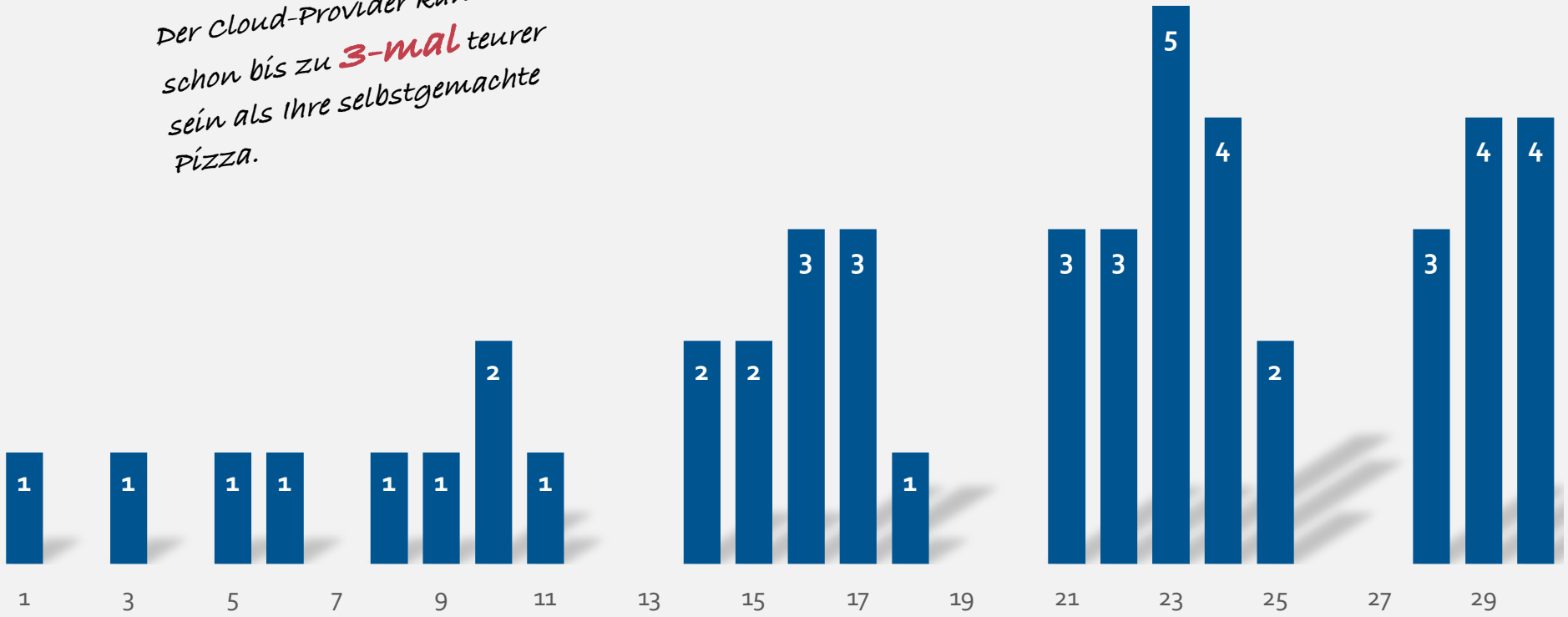
An Wochenenden tun Sie das natürlich nicht.

$$p = 1$$
$$a = {}^{22}/_{30}$$
$$p/a \approx 1.3$$

CONTINUOUSLY-CHANGING WORKLOADS

Der Trendsetter-Workload

Der Cloud-Provider kann nun schon bis zu **3-mal** teurer sein als Ihre selbstgemachte Pizza.



Sie bringen Ihren Kollegen immer was vom Pizzawagen mit.

Das spricht sich rum, und Woche für Woche müssen Sie mehr Pizza besorgen.

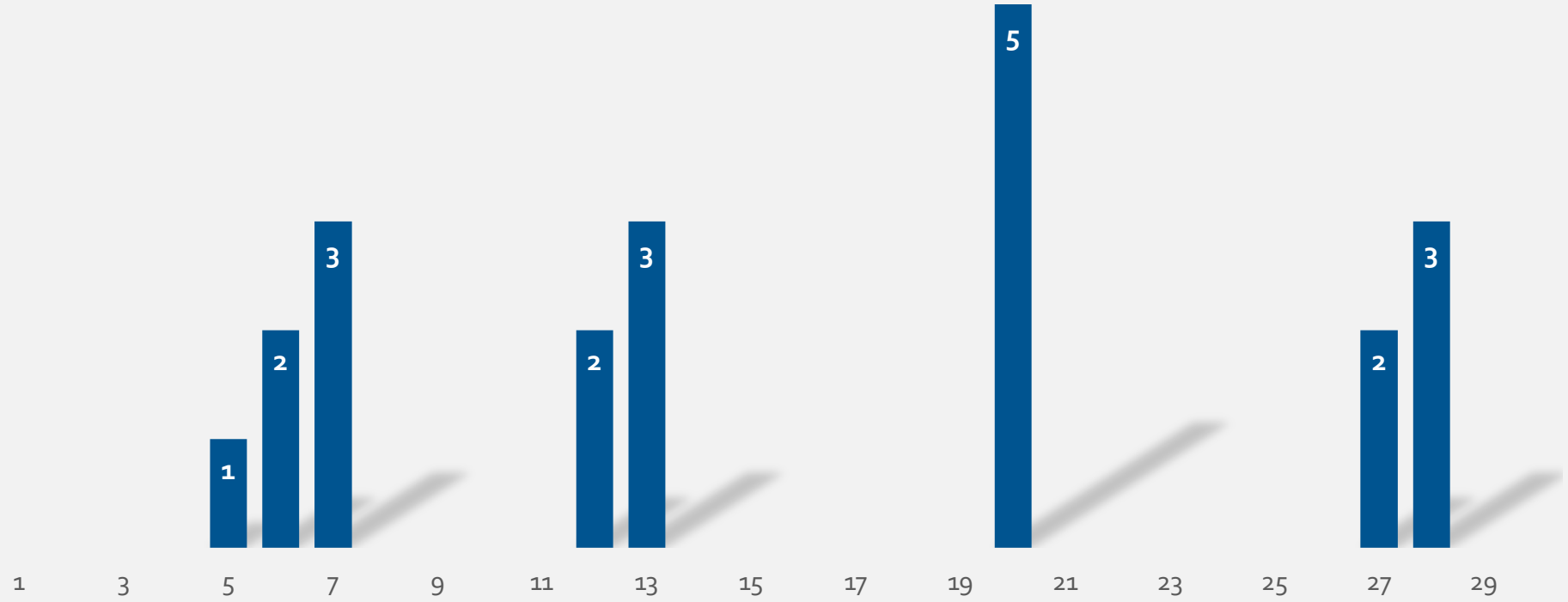
An Wochenenden arbeitet natürlich niemand.

$$p = 5$$
$$a = {}^{48}/_{30}$$
$${}^{p}/_{a} \approx 3.1$$

PERIODISCHE WORKLOADS

Der DVD/Party-Workload

Ihr Bedarf wird seltener und der Cloud-Provider kann nun schon bis zu ~~7~~ mal teurer sein als Ihre selbstgemachte Pizza.



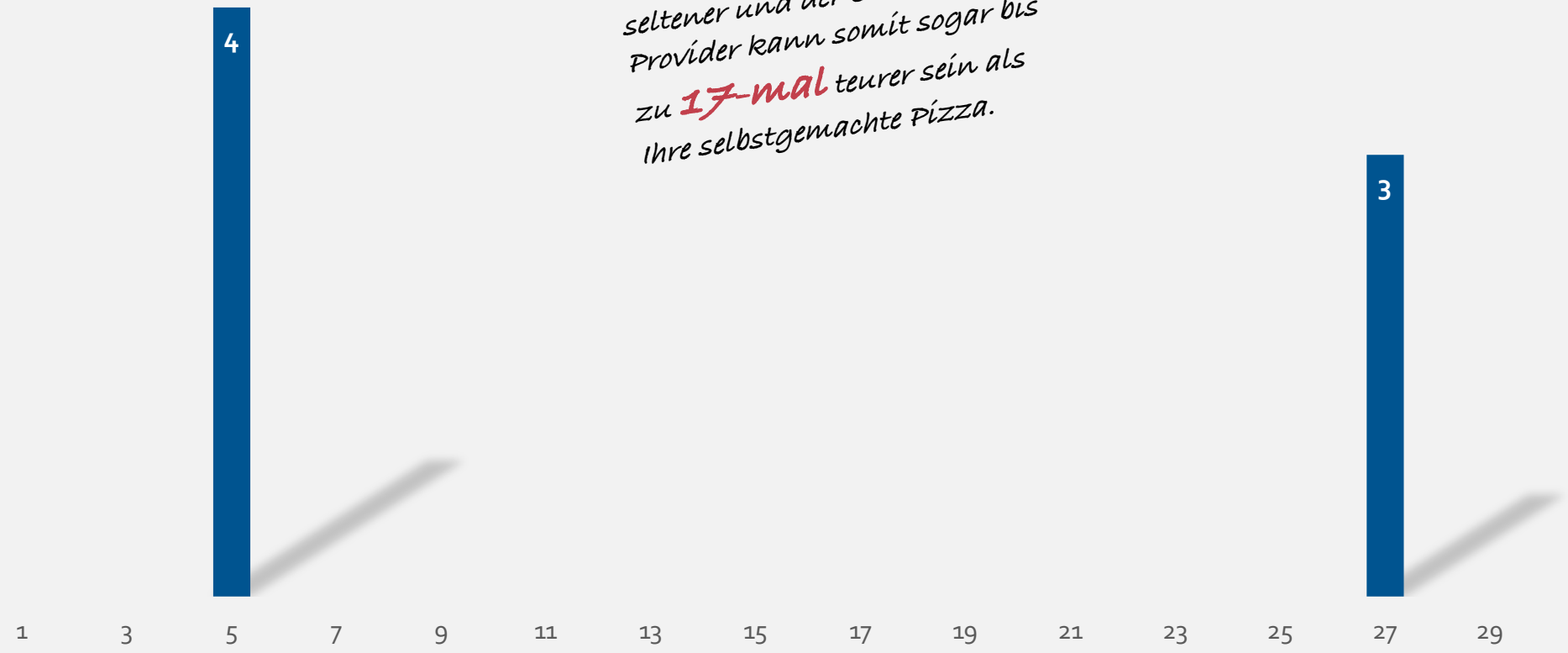
Sie machen mit Familie und Freunden an Wochenenden DVD-Abende und reichen dazu Pizza.

Unter der Woche haben Sie dazu natürlich keine Zeit.

$$p = 5$$
$$a = {}^{21}/_{30}$$
$$p/a \approx 7.1$$

UNVORHERSEHBARE/SELTENE WORKLOADS

Der Pizzeria-Workload



Ihr Bedarf wird nun noch seltener und der Cloud-Provider kann somit sogar bis zu 17-mal teurer sein als Ihre selbstgemachte Pizza.

CLOUD NATIVE COMPUTING

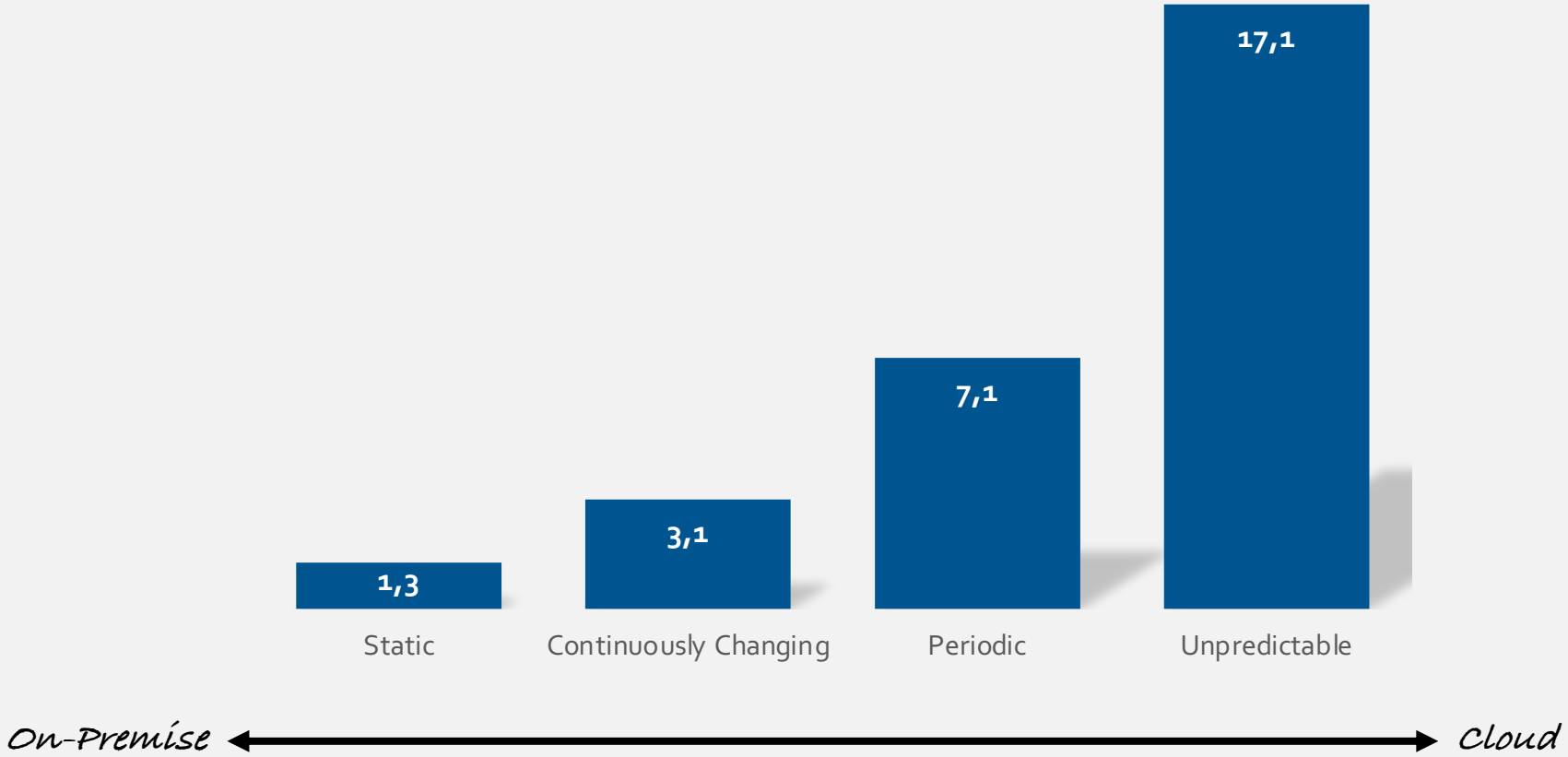
Sie laden Ihre Familie an Wochenenden ab und an in eine Pizzeria ein.

Unter der Woche haben Sie dazu natürlich keine Zeit.

$$p = 4$$
$$a = {}^{7}/_{30}$$
$$p/a \approx 17.1$$

KOSTENVORTEILE

entstehen in erster Linie durch den Workload



CLOUD NATIVE COMPUTING

Kostenvorteile entstehen also in aller Regel durch den Workload und erst in zweiter Linie durch die Kostenstruktur des Dienstes.

Fun Fact:
AWS hat ca. 50% Marktanteil im Cloud Computing verlangt aber häufig bis zu 10% mehr pro Einheit für Services als bspw. Google.

Warum ist das so?

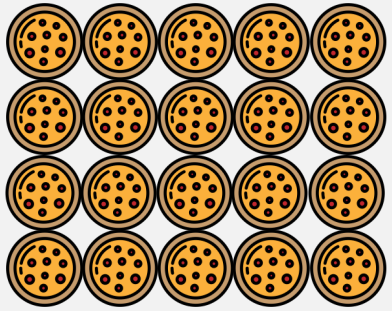
Vielleicht weil bei diesen Kostenvorteilen 10% Preisunterschiede kaum noch messbar sind?

FUN FACT

Es kostet übrigens dasselbe …

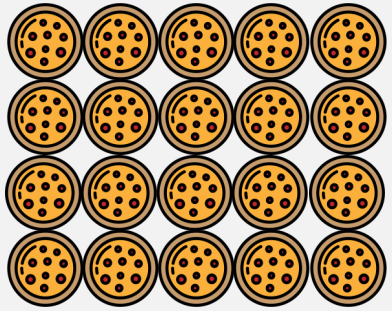
Diese Tatsache nennt sich **Kostenassoziativität.**

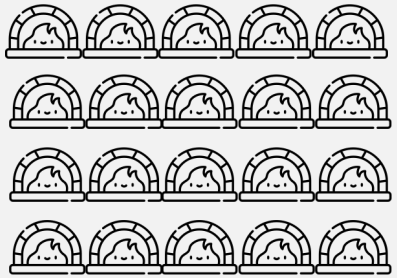
1 Ofen für 20 x 15 min zu mieten



Dauer: 5 Stunden

20 Öfen für 15 min zu mieten





Dauer: 15min

CLOUD NATIVE COMPUTING

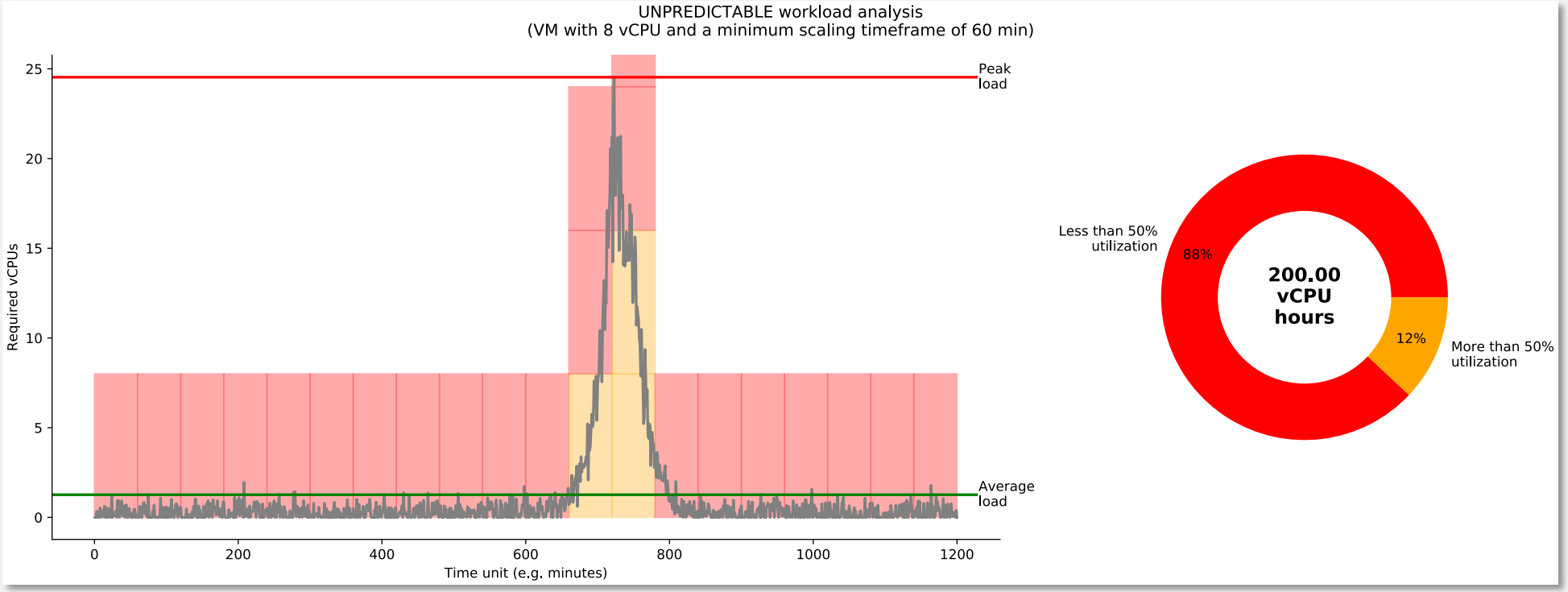
Cloud-Ökonomie kreativ nutzen

Bei welchem Lieferdienst würden Sie 20 Pizzen bestellen?

- Bei dem der Sie in 5 Stunden beliefert und bei dem 19 Pizzen kalt sind?
- Bei dem, der Ihnen nach 15 Minuten 20 warme Pizzen liefern kann?
- Wieviel Aufpreis wäre Ihnen das wert?
- Wieviel Mehraufwand kostet das den Lieferdienst?
- Wie häufig brauchen Sie als Lieferdienst wohl 20 Öfen gleichzeitig?

DER EFFEKT FREINGRANULARER ZUTEILUNGEN

Am Beispiel der Unpredictable Workload Kategorie



Wir sehen in diesem Modell recht ineffizient genutzte VMs. Aber ungenutzte VMs kosten dasselbe wie genutzte VMs. Die Ursache liegt letztlich in zu „großen Kästen".

An welchen Stellschrauben kann man also drehen, um die Ressourcennutzung effizienter zu machen (also die Kästen kleiner zu machen)?

Das ist im Wesentlichen der Kostentreiber!!!

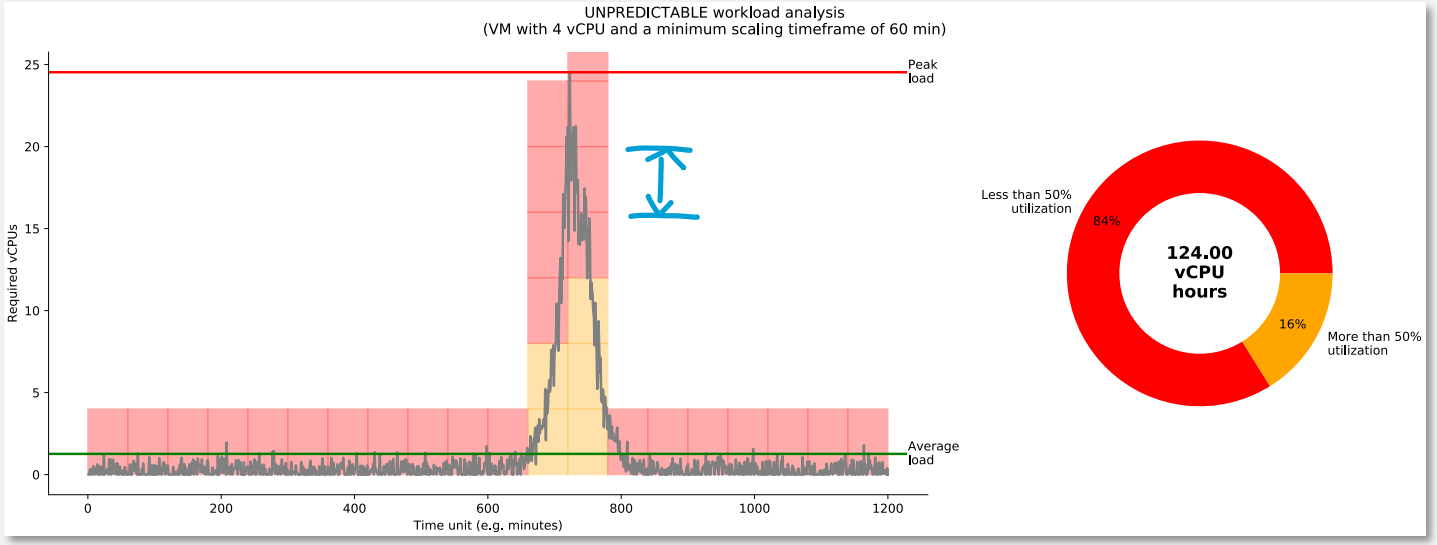
Ausgangssituation:
Nutzung von großen virtuellen Maschinen für „lange" Zeiträume

VM: 8 vCPU
Dauer: 60 min

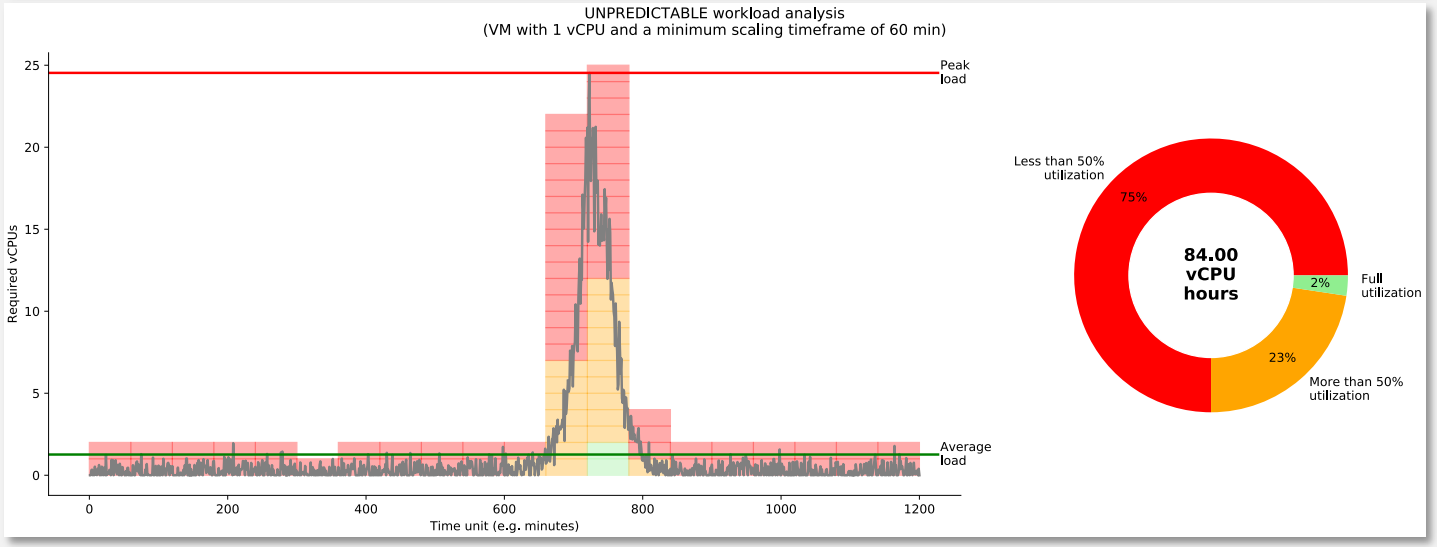
Resultierende Ressourcen-anforderung:
200 CPU-Stunden

DER EFFEKT FREINGRANULARER ZUTEILUNGEN

Was passiert wenn wir die VM Größe reduzieren?



VM: 4 vCPU
Dauer: 60 min
=> 124 CPU-Stunden



VM: 1 vCPU
Dauer: 60 min
=> 84 CPU-Stunden

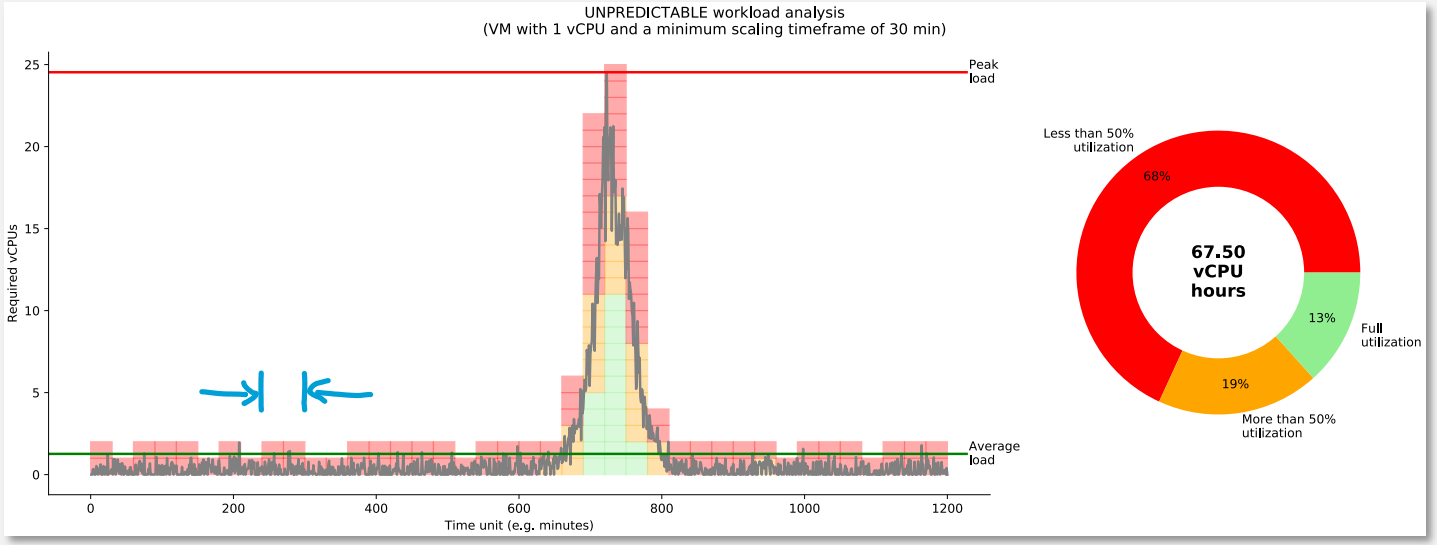
Stellschraube 1:
VM-Größe

Mit kleineren VMs benötigt man zwar mehr VMs, aber fordert dennoch weniger CPU-Stunden insgesamt an.

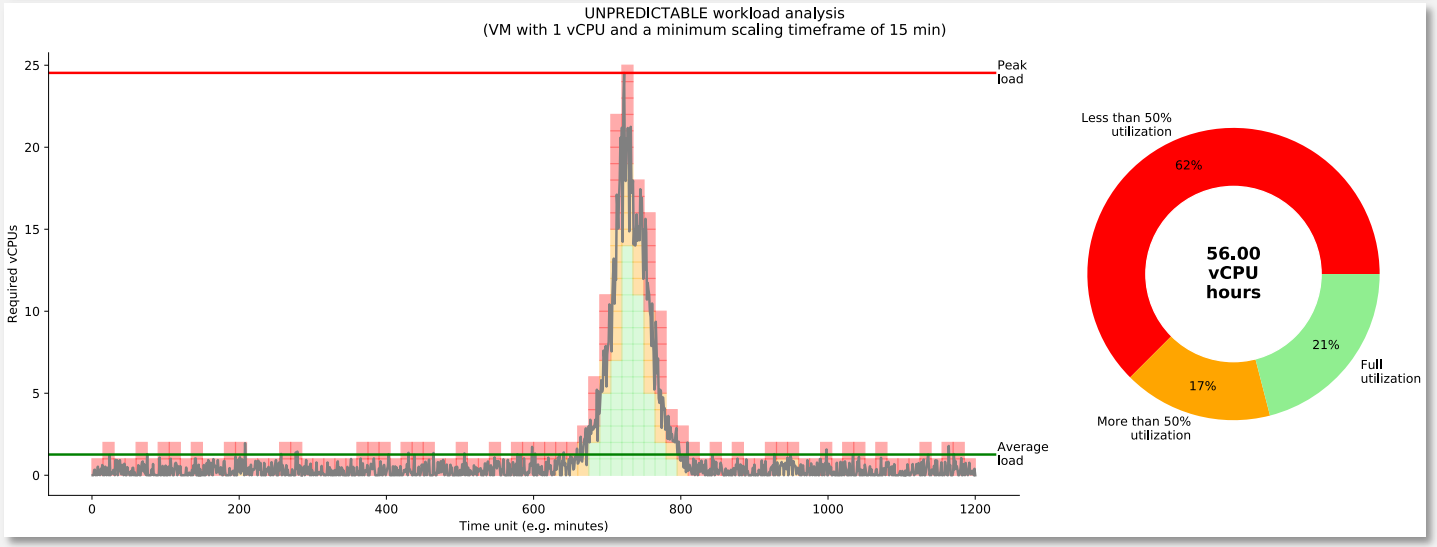
Mit kleineren VMs kann man Lastkurven also **„enger"** folgen und verschwendet **weniger** ungenutzte Ressourcen.

DER EFFEKT FREINGRANULARER ZUTEILUNGEN

Was passiert wenn wir die Zuteilungsdauer reduzieren?



VM: 1 vCPU
Dauer: 30 min
=> 67.5 CPU-Stunden



VM: 1 vCPU
Dauer: 15 min
=> 56 CPU-Stunden

Stellschraube 2:
Zeit
(Abrechnungsintervalle)

*Mit kürzeren Zuteilungsdauern kann man Lastkurven also **„schneller"** folgen und verschwendet **kürzer** ungenutzte Ressourcen.*

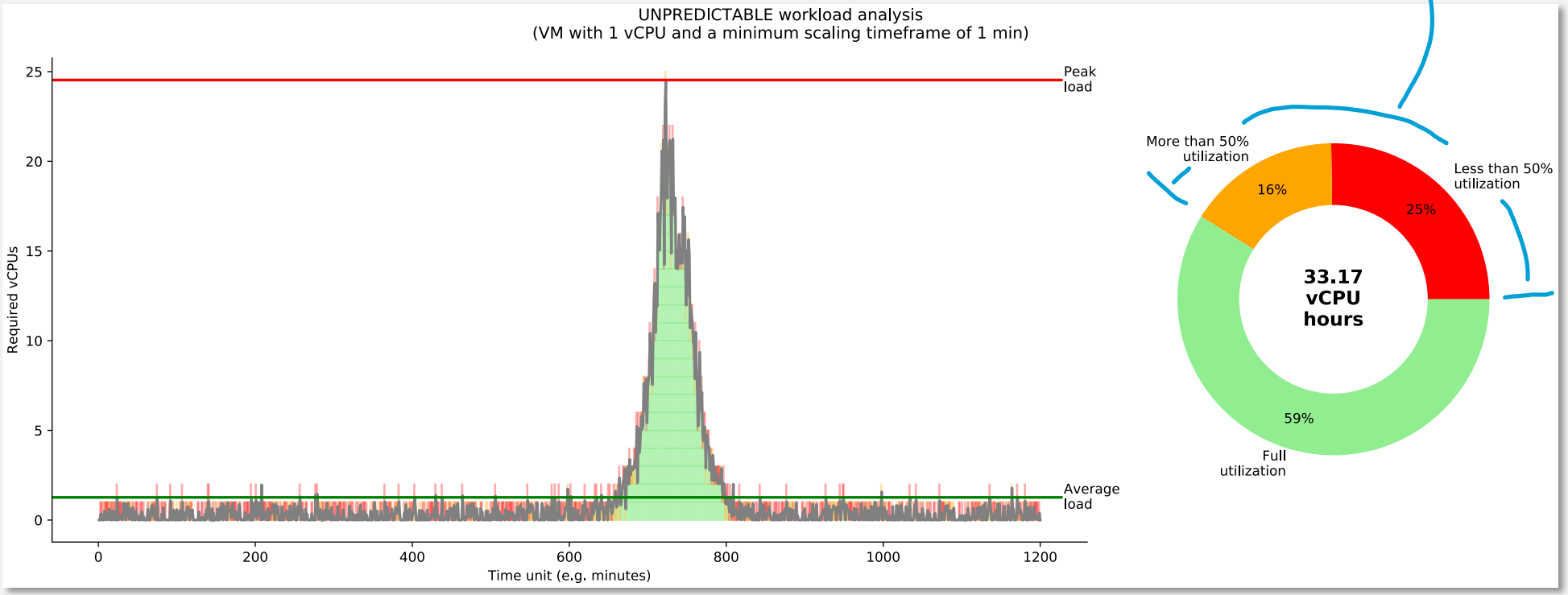
Alle Cloud Provider haben mit einer Stunden-basierten Abrechnung begonnen, und haben dann auf 30min, 15min, 5min, 1minAbrechnungsinter valle umgestellt (einige auch bereits auf Sekunden-basis).

Den maximalen „Effekt" den Sie damit erzielen können, sehen Sie auf der linken Seite!

DER EFFEKT FREINGRANULARER ZUTEILUNGEN

Wo liegen die Grenzen dieses IaaS-fokussierten Ansatzes?

Hier scheint noch Weiteres Optimierungs-potenzial zu liegen.



VM: 1 vCPU
Dauer: 1 min

⇒ **33.17** CPU-Stunden

Um in diesen Bereich vorzudringen müssen wir allerdings noch schneller und noch feingranularer skalieren können.

- Container (Sub Core Scale)
- FaaS (Sub Second Scale)

JUPYTER NOTEBOOK

Nur Versuch macht kluch ...

GitLab

CLOUD NATIVE COMPUTING

Klonen Sie dieses Repository (in Jupyter Lab, https://jupyter.mylab.th-luebeck.de):

```
git clone https://git.mylab.th-luebeck.de/cloud-native/lab-workload-analysis.git
```

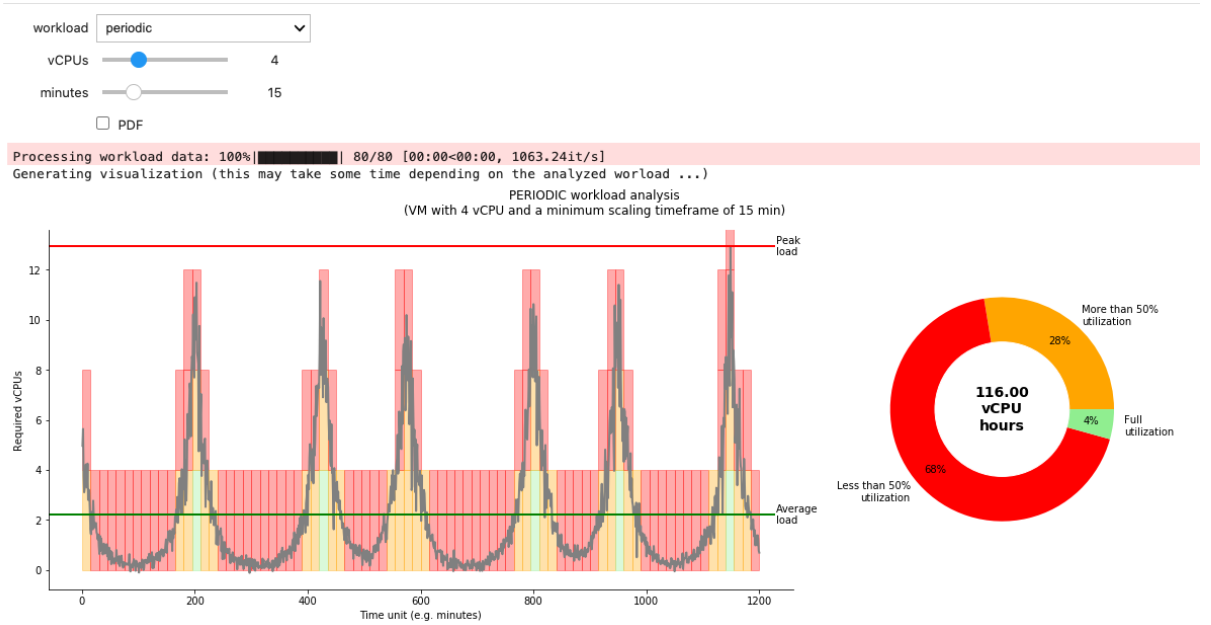
Workload Analyse

Übung 1:
Workload Arten

Übung 2:
Periodische Workloads

Übung 3:
Unpredictable Workloads

Übung 4:
Weitere Workloads

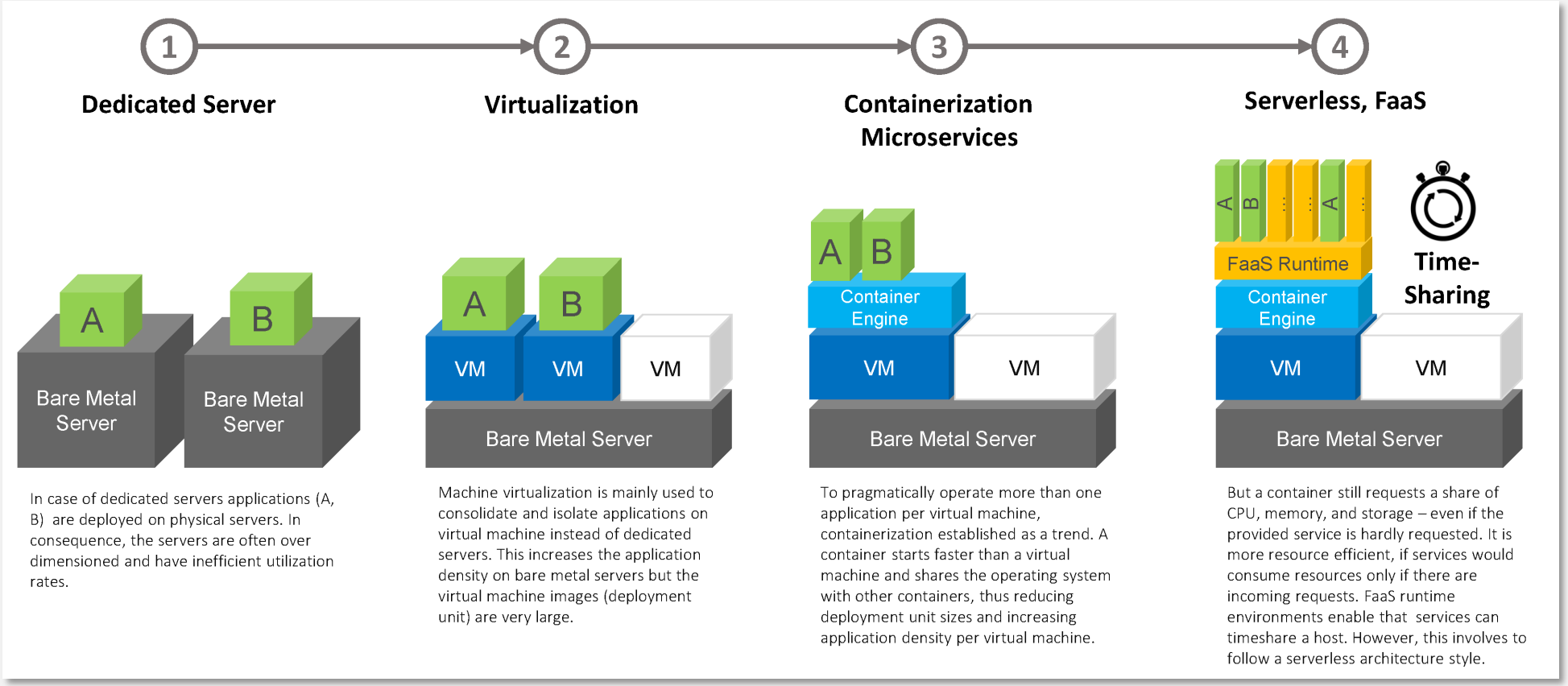


INHALTE

- Cloud Computing
- Cloud-Service Modelle
- Cloud-Ökonomie (Pay-as-you-Go)
- Eine kurze Architekturgeschichte der Cloud
- Cloud-native Anwendungen und Dienste
- Zusammenfassung

EINE KURZE GESCHICHTE DER CLOUD

Die Verkleinerung von Deployment Units im Verlaufe der Zeit



Quelle: Kratzke, N. A Brief History of Cloud Application Architectures. Appl. Sci. 2018, 8, 1368.

CLOUD NATIVE COMPUTING

Im Verlaufe der Zeit sind die Größen von Deployment Units geschrumpft!

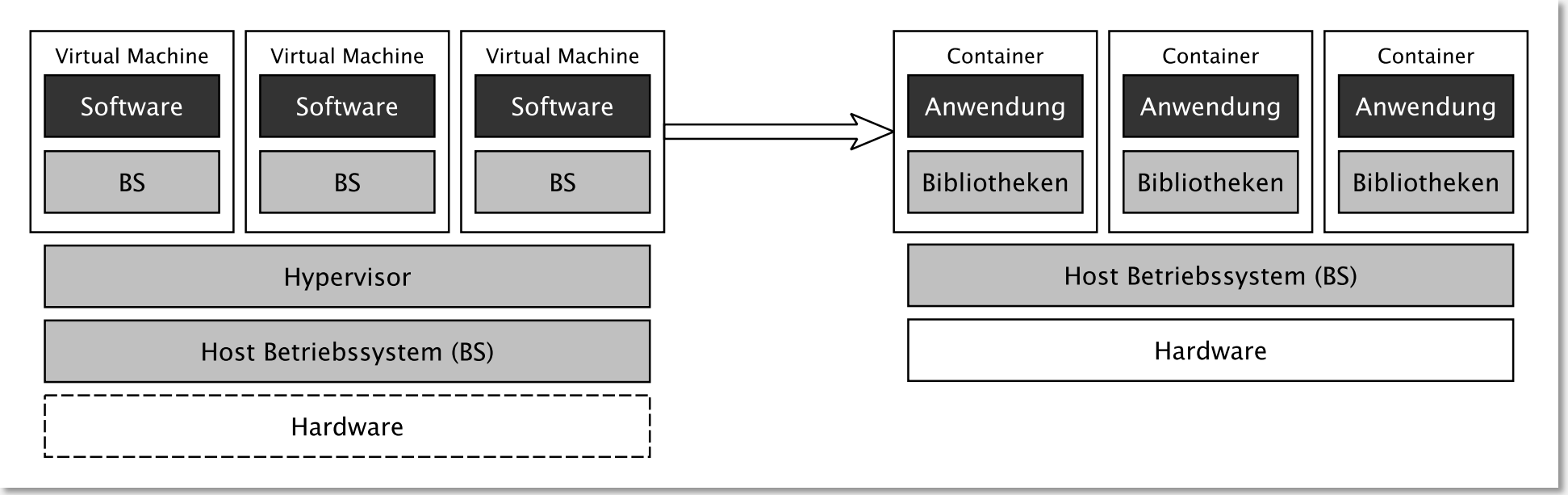
VM -> Container -> Function

Parallel dazu haben sich Servic-orientierte Architekturen (weiter)entwickelt.

Microservices + Serverless Architectures

EINE KURZE GESCHICHTE DER CLOUD

Containerisierung



Überwindung der 1-vCPU-Schwelle!

Container nutzen Betriebssystem-Virtualisierung (Prozessisolation) anstelle von Maschinen-Virtualisierung.

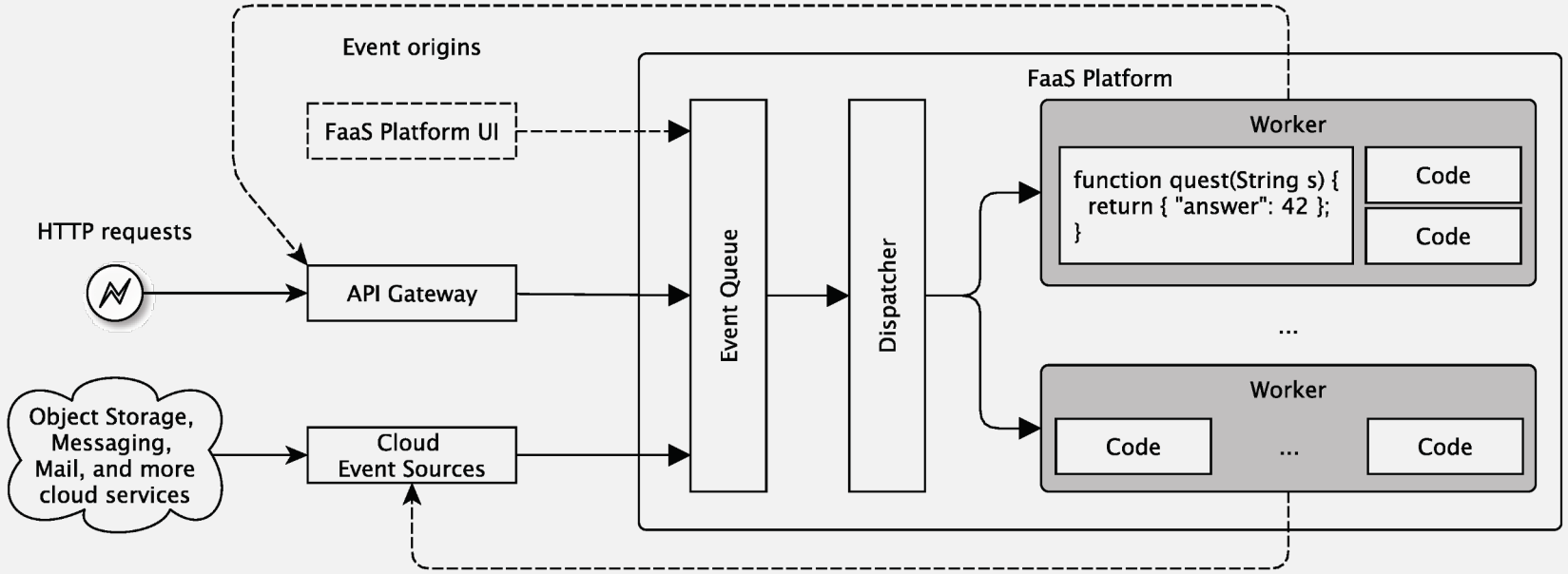
Nachteil:
Container-OS und Host-OS müssen identisch sein (meist Linux).

Vorteil:
Container starten sehr viel schneller und können wesentlich kleiner sein.

EINE KURZE GESCHICHTE DER CLOUD

CLOUD NATIVE COMPUTING

Serverless Computing aka Function as a Service (FaaS)



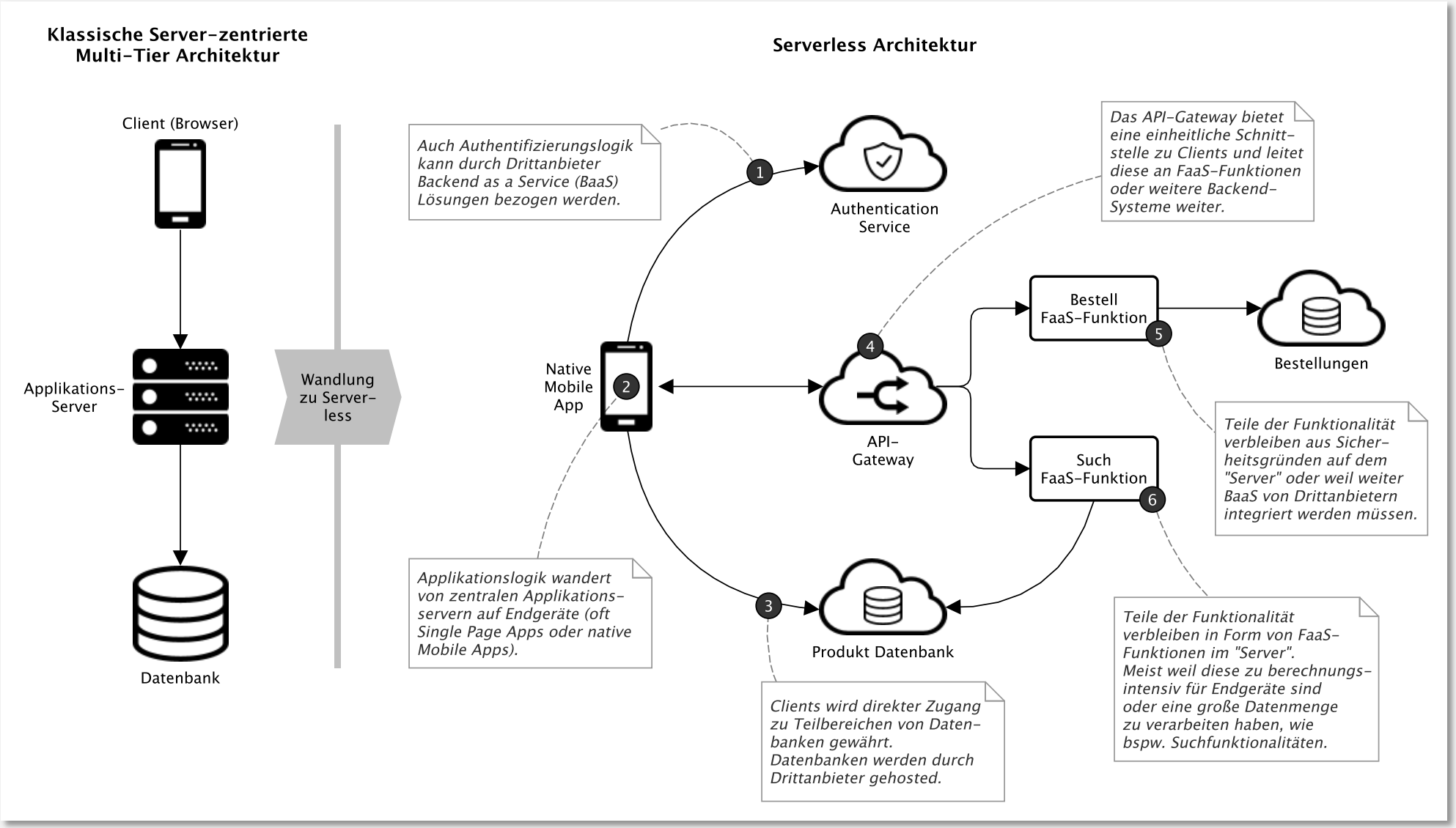
Eine serverlose Plattform ist lediglich ein Ereignisverarbeitungssystem. Ereignisse können z.B. über HTTP gesendet oder von weiteren Ereignisquellen (aus Cloud-Infrastrukturen) empfangen werden. Die Plattform bestimmt dann, welche Funktionen für ein Ereignis registriert sind, sendet das Ereignis an die Funktionsinstanz und wartet auf eine Antwort um diese Antwort an den Requestor weiterzuleiten (im Falle von Request-Response-basiertem Triggering).

Siehe auch Unit 06! FaaS

Quelle: Nane Kratzke: A Brief History of Cloud Application Architectures. In *Applied Sciences*; MDPI, 2018, https://doi.org/10.3390/app8081368

SERVERLESS ARCHITEKTUREN

Containerisierung in Cloud-nativen Anwendungen zu Ende gedacht

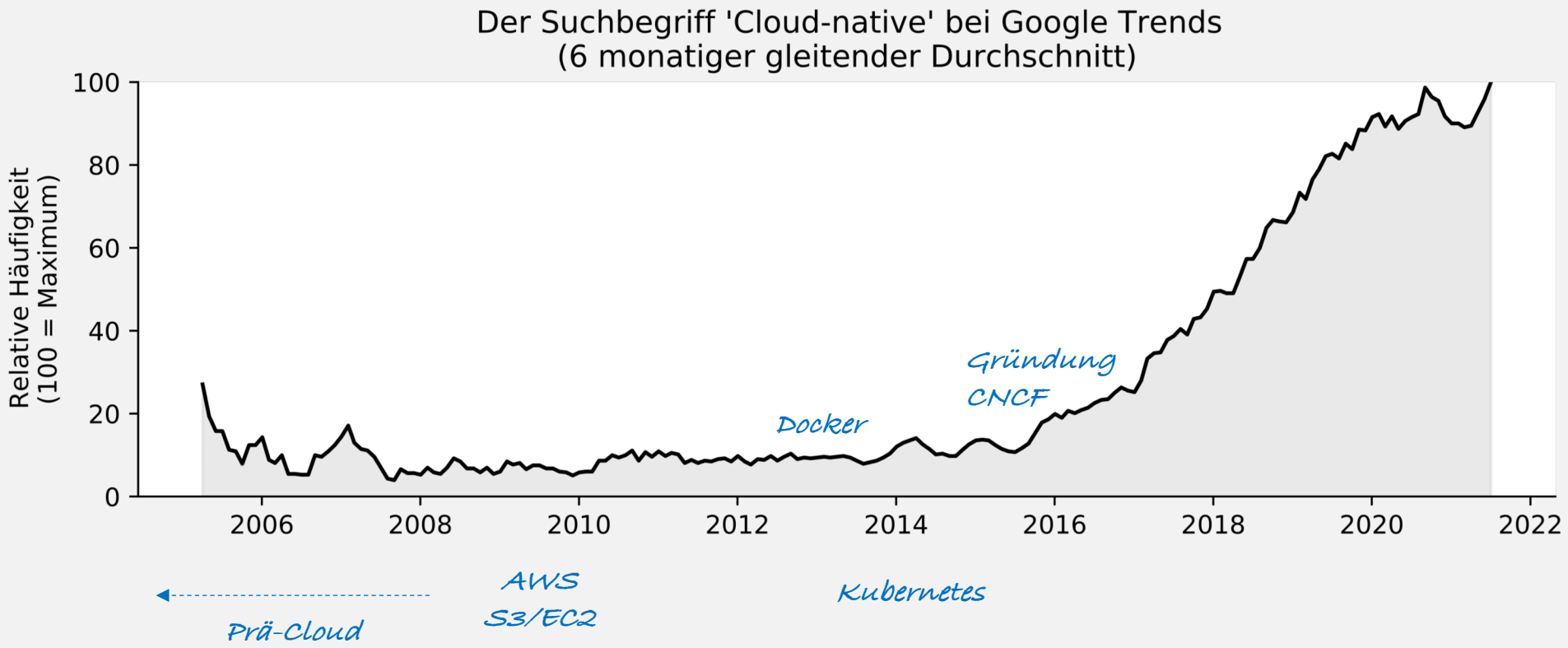


INHALTE

- Cloud Computing
- Cloud-Service Modelle
- Cloud-Ökonomie (Pay-as-you-Go)
- Eine kurze Architekturgeschichte der Cloud
- Cloud-native Anwendungen und Dienste
- Zusammenfassung

DER BEGRIFF „CLOUD-NATIVE“

Im Verlaufe der Zeit



CNCF =
Cloud Native
Computing Foundation

THE EIGHT FALLACIES OF DISTRIBUTED COMPUTING

Bill Joy + Tom Lyon (1 – 4) , *Peter Deutsch (5- 7)*, *James Gosling (8)*

Diese Liste der sogenannten **Irrtümer der verteilten Datenverarbeitung** sind eine Sammlung eigentlich trivialer, aber doch häufiger fehlerhafter Annahmen, die Entwickler (häufig unbewusst) voraussetzen, wenn sie insbesondere das erste Mal eine verteilte Anwendung entwickeln.

Die Liste der Fallacies of Distributed Computing kommt ursprünglich von **Sun Microsystems** und wurde dort von Bill Joy und Tom Lyon mit vier Trugschlüssen eröffnet. Weite Bekanntheit erlangten sie 1994 durch Peter Deutsch, der sie zu sieben Trugschlüsse erweiterte und als "The Seven Fallacies of Distributed Computing" veröffentlichte. James Gosling, ebenfalls von Sun, setzte ungefähr 1997 noch den achten Punkt dazu.

(1) Das Netzwerk ist ausfallsicher

(2) Die Latenzzeit ist gleich null

(3) Der Datendurchsatz ist unbegrenzt

(4) Das Netzwerk ist sicher

(5) Die Netzwerktopologie wird sich nicht ändern

(6) Es gibt immer nur einen Netzwerkadministrator

(7) Die Kosten des Datentransports können mit null angesetzt werden

(8) Das Netzwerk ist homogen

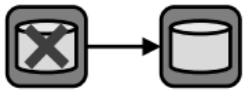
CLOUD NATIVE COMPUTING

SUN Microsystems

ist die Firma, die Java (1996, V1.0) entwickelt hat und 2009/2010 von Oracle übernommen wurde.

CLOUD-NATIVE APPLIKATIONEN

Das IDEAL-Modell (Von Service-orientierten Architekturen inspiriert)

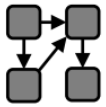


I solated State

A maximimum of cloud application components should be stateless. They do not handle:

Session State:
State of the communication with the application.

Application State:
Business data handled by the application.



D istribution

Cloud applications are split up into multiple components

to utilize multiple cloud resources

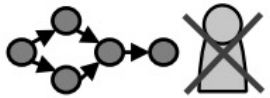
because the cloud itself is a large distributed system.

E lasticity

Cloud applications are scaled by adjusting resource numbers (scaling out) – not by scaling up:

Scale out (horizontally):
Increase performance by adding more resources.

Scale up (vertically):
Increase performance by improving existing resources.

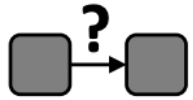


A utomated

Management tasks during runtime have to be handled quickly.

Example: Cost reduction by adjusting pay-per-use resource numbers automatically.

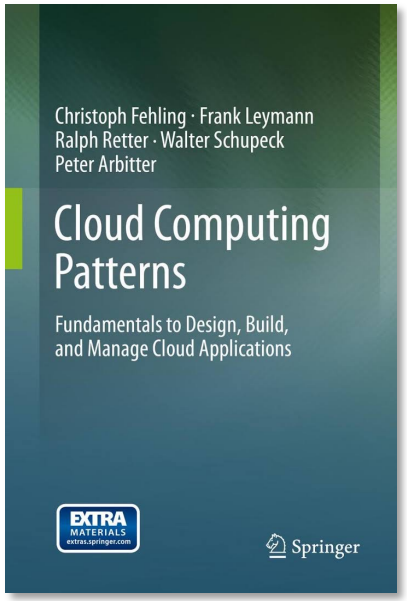
Example: Automatic reaction to resource failures.



L oose Coupling

Cloud-native application components should not influence each other regarding factors such as availability, data format, data exchange rate.

Example:
Failure of one application component does not cause failure of other components.



Cloud Computing Patterns

http://www.cloudcomputingpatterns.org

Native Cloud Applications

Two-Tier Cloud Application (290)
Presentation and business logic is bundled to one stateless tier that is easy to scale. It is separated from the data tier that is harder to scale.

Three-Tier Cloud Application (294)
Presentation, business logic, and data handling are realized in separate tiers to scale them according to their individual requirements.

Content Distribution Network (300)
Applications component instances and data handled by them are globally distributed to meet access performance requirements.

Hybrid Cloud Applications

Hybrid User Interface (304)
Varying workload from a user group interacting asynchronously with an application is handled in an elastic environment.

Hybrid Processing (308)
Processing functionality is hosted in an elastic cloud while the remainder of an application resides in a static environment.

Hybrid Data (311)
Data of varying size is hosted in an elastic cloud while the remainder of an application resides in a static environment.

Hybrid Backup (314)
Data is periodically extracted from an application to be archived in an elastic cloud for disaster recovery purposes.

Hybrid Backend (317)
Backend functionality (data-intensive processing and storage) is experiencing varying workloads and is hosted in an elastic cloud.

Hybrid Application Functions (320)
Some application functionality provided by user interfaces, processing, and data handling is hosted in an elastic cloud.

Hybrid Multimedia Web Application (323)
Website content is mainly served from a static environment. Multimedia files are served from an elastic high-performance environment.

Hybrid Development Environment (326)
A production runtime environment is replicated and mocked in an elastic environment where applications are developed and tested.

Application Workloads

Static Workload (26)
IT resources with an equal utilization over time experience static workload.

Periodic Workload (29)
IT resources with a peaking utilization at reoccurring time intervals experience periodic workload.

Once-in-a-lifetime Workload (33)
IT resources with an equal utilization over time disturbed by a strong peak occurring only once experience once-in-a-lifetime workload.

Unpredictable Workload (36)
IT resources with a random and unforeseeable utilization over time experience unpredictable workload.

Continuously Changing Workload (40)
IT resources with a utilization that grows or shrinks constantly over time experience continuously changing workload.

Cloud Service Models & Cloud Deployment Types

Infrastructure as a Service (IaaS) (45)
Physical and virtual hardware IT resources are shared between customers to enable self-service, rapid elasticity, and pay-per-use pricing.

Platform as a Service (PaaS) (49)
An application hosting environment is shared between customers to enable self-service, rapid elasticity, and pay-per-use pricing.

Software as a Service (SaaS) (55)
Human-usable application software is shared between customers to enable self-service, rapid elasticity, and pay-per-use pricing.

Public Cloud (62)
IT resources are provided as a service to a very large customer group in order to enable elastic use of a static resource pool.

Private Cloud (66)
IT resources are provided as a service exclusively to one customer in order to meet requirements on privacy, security, and trust.

Community Cloud (71)
IT resources are provided as a service to multiple customers trusting each other in order to enable collaborative elastic use of resources.

Hybrid Cloud (75)
Different clouds and static data centers are integrated to form a homogeneous hosting environment.

Fundamental Architecture Styles

Loose Coupling (156)
A broker encapsulates concerns of communication partner location, implementation platform, time of communication, and data format.

Distributed Application (160)
A cloud application divides provided functionality among multiple application components that can be scaled out independently.

Application Components

Stateful Component (168)
Multiple instances of a scaled-out application component synchronize their internal state to provide a unified behavior.

Stateless Component (171)
State is handled external of application components to ease scaling-out and to make the application more tolerant to component failures.

Multi-Component Image (206)
Virtual servers host multiple components that may not be active at all times to reduce provisioning and decommissioning operations.

User Interface Component (175)
Customizable user interfaces are accessed by humans. Application-internal interaction is realized asynchronously to ensure loose coupling.

Processing Component (180)
Processing functionality is handled by elastically-scaled components. Functionality is made configurable to support different requirements.

Batch Processing Component (185)
Requests are delayed until environmental conditions make their processing feasible.

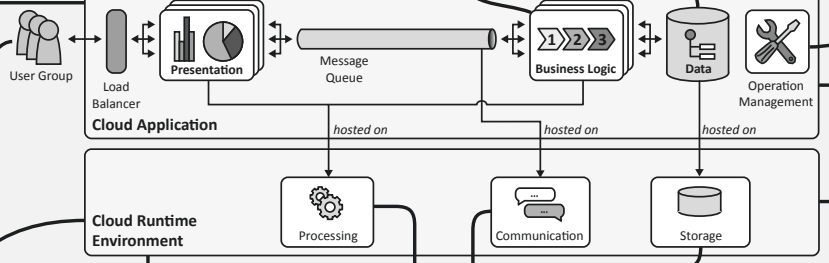
Timeout-based Message Processor (204)
Clients acknowledge message processing to ensure that messages are processed. If a message is not acknowledged it is processed again.

Transaction-based Processor (201)
Components receive messages or read data and process the obtained information under a transactional context to ensure processing.

Idempotent Processor (197)
Application functions detect duplicate messages and inconsistent data or are designed to be immune to these conditions.

Data Access Component (188)
Access to data is handled by components that isolate complexity, enable additional consistency, and ensure adjustability of data elements.

Data Abstractor (194)
Data is altered to inherently support eventually consistent data storage through the use of abstractions and approximations.



Application Management

Provider Adapter (243)
Provider interfaces are encapsulated to separate concerns of interactions with the provider from application functionality.

Managed Configuration (247)
Application components use a centrally stored configuration to provide a unified behavior that can be adjusted simultaneously.

Elasticity Manager (250)
The utilization of IT resources on which an application is hosted is used to adjust the number of required application component instances.

Elastic Load Balancer (254)
The number of synchronous accesses is used to adjust the number of required application component instances.

Elastic Queue (257)
The number of accesses via messaging is used to adjust the number of required application component instances.

Watchdog (260)
Applications cope with failures automatically by monitoring and replacing faulty application component instances.

Elasticity Management Process (267)
Application component instances are added and removed automatically to cope with increasing or decreasing workload.

Feature Flag Management Process (271)
If the cloud cannot provide required resources in time, some application features are degraded in order to keep vital features operational.

Update Transition Process (275)
When a new application component version becomes available, running application components are updated seamlessly.

Standby Pooling Process (279)
Application component instances are kept on standby to increase provisioning speed and utilize billing time-slots efficiently.

Resiliency Management Process (283)
Application components are checked for failures and replaced automatically without human intervention.

Cloud Integration

Restricted Data Access Component (222)
Data provided to clients from different environments is adjusted based on access restrictions.

Message Mover (225)
Messages are moved automatically between different cloud providers to provide unified access to application components using messaging.

Application Component Proxy (228)
An application component is made available in an environment from where it cannot be accessed directly by deploying a proxy.

Compliant Data Replication (231)
Data is replicated among multiple environments. Data is automatically obfuscated and deleted to meet laws and security regulations.

Integration Provider (234)
Integration functionality such as messaging and shared data is hosted by a separate provider to enable integration of hosting environments.

Cloud Environments

Elastic Infrastructure (87)
Hosting of virtual servers, disk storage, and configuration of network connectivity is offered via a self-service interface over a network.

Elastic Platform (91)
Middleware for the execution of applications, their communication, and data storage is offered via a self-service interface over a network.

Node-based Availability (95)
A cloud provider guarantees the availability of individual nodes, such as virtual servers, middleware, or hosted application components.

Environment-based Availability (98)
A cloud provider guarantees the availability of the environment hosting individual nodes, such as virtual servers or application components.

Multi-Tenancy

Shared Component (210)
A component is accessed by multiple tenants to leverage economies of scale.

Tenant-isolated Component (214)
A component avoids influences between tenants regarding assured performance, available storage capacity, and accessibility.

Dedicated Component (218)
Components providing critical functionality are provided exclusively to tenants while still allowing other components to be shared.

Storage Offerings

Block Storage (110)
Centralized storage is integrated into servers as a local hard drive to enable access to this storage via the local file system.

Blob Storage (112)
Data is provided in form of large files that are made available in a file system-like fashion.

Relational Database (115)
Data is structured according to a schema that is enforced during data manipulation and enables expressive queries of handled data.

Key-Value Storage (107)
Semi-structured or unstructured data is stored with limited querying support but high-performance, availability, and flexibility.

Strict Consistency (123)
Data is stored at different locations to improve response time and failure resiliency while consistency of replicas is ensured at all times.

Eventual Consistency (126)
Performance and availability of data are increased by ensuring data consistency eventually and not at all times.

Processing Offerings

Hypervisor (101)
To enable the elasticity of clouds, the time required to provision and decommission servers is reduced through hardware virtualization.

Execution Environment (104)
To avoid duplicate implementation of functionality, applications are deployed to a hosting environment providing common functionality.

Map Reduce (106)
Large data sets to be processed are divided into smaller data chunks and distributed among processing application components.

Communication Offerings

Virtual Networking (132)
Networking resources are virtualized to enable customers to configure networks, firewalls, and remote access using a self-service interface.

Message-oriented Middleware (136)
Asynchronous communication is made robust and flexible by hiding the complexity of addressing, routing, or data formats.

Exactly-once Delivery (141)
The messaging system ensures that each message is delivered exactly once by filtering possible message duplicates automatically.

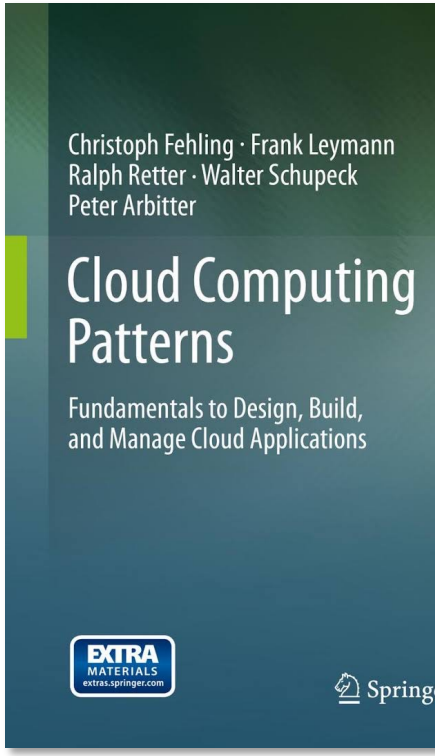
At-least-once Delivery (144)
In case of failures that lead to message loss, messages are retransmitted to assure they are delivered at least once.

Transaction-based Delivery (146)
Clients retrieve messages under a transactional context to ensure that messages are received by a handling component.

Timeout-based Delivery (149)
Clients acknowledge message receptions to ensure that messages are received properly.

CLOUD NATIVE COMPUTING

Das IDEAL-Modell „nur" etwas detaillierter ;-)



CLOUD-NATIVE APPLIKATIONEN

Definition

CNCF Cloud Native Definition v1.0

Cloud native Technologien ermöglichen es Unternehmen, skalierbare Anwendungen in modernen, dynamischen Umgebungen zu implementieren und zu betreiben. Dies können öffentliche, private und Hybrid-Clouds sein. Best-Practices, wie Container, Service-Meshs, Microservices, immutable Infrastruktur und deklarative APIs, unterstützen diesen Ansatz.

Die zugrundeliegenden Techniken ermöglichen die Umsetzung von entkoppelten Systemen, die belastbar, handhabbar und beobachtbar sind. Kombiniert mit einer robusten Automatisierung können Softwareentwickler mit geringem Aufwand flexibel und schnell auf Änderungen reagieren.

Source: Cloud-native Computing Foundation, https://github.com/cncf/toc/blob/master/DEFINITION.md

Industry point of view

Understanding Cloud-native Applications after 10 Years of Cloud Computing

A cloud-native application (CNA) is a distributed, elastic and horizontally scalable system composed of loosely-coupled (micro)services. A CNA isolates state in a minimum of stateful components.

The application and each self-contained deployment unit of that application are designed according to cloud-focused design patterns and operated on a self-service elastic platform.

Source: Kratzke, Nane and Quint, Peter-Christian. Understanding Cloud-native Applications after 10 Years of Cloud Computing - A Systematic Mapping Study, in Journal of Systems and Software, Elsevier, 2017, DOI: 10.1016/j.jss.2017.01.001

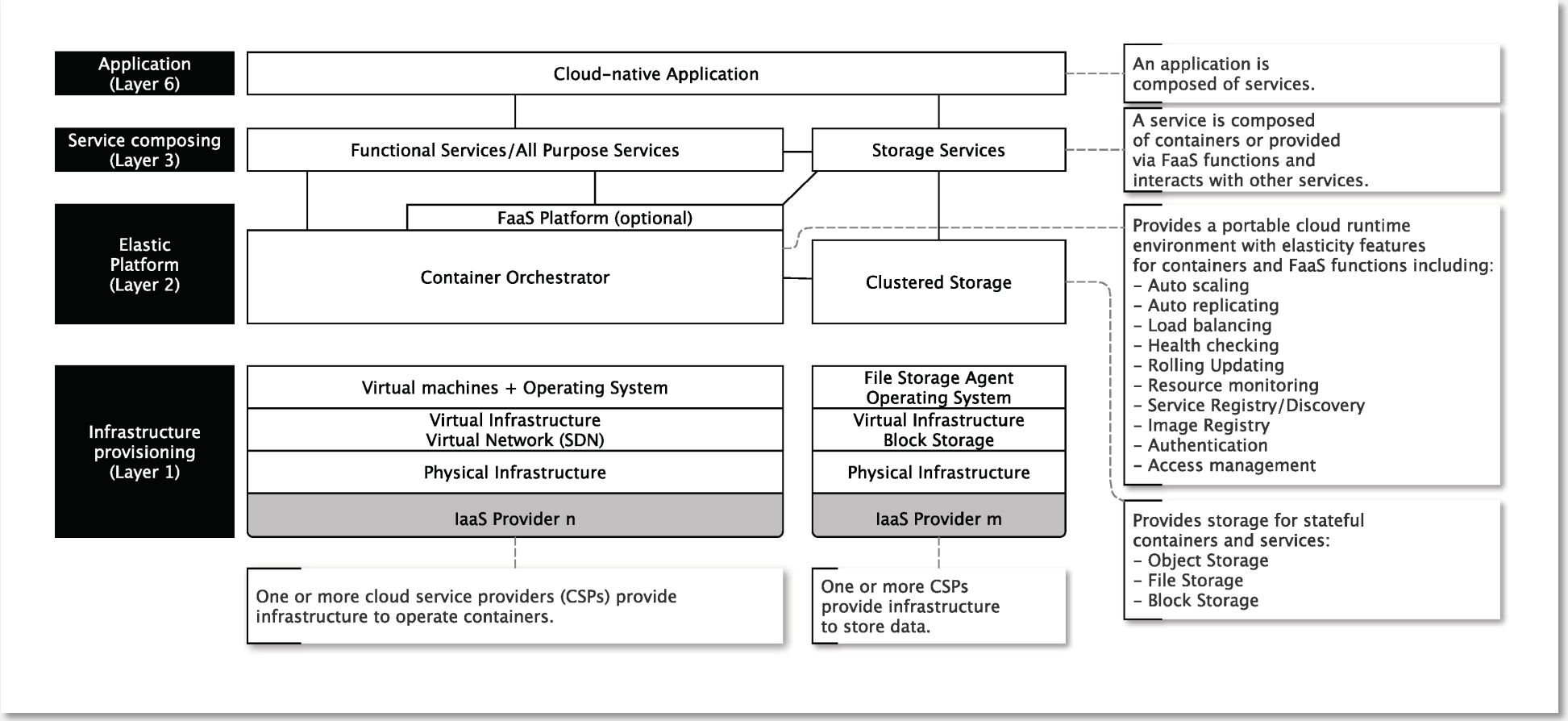
Software engineering research point of view

CLOUD NATIVE COMPUTING

Oft zu hören: Cloud-native Applikationen sind solche, die absichtsvoll für Cloud-Infrastrukturen und -Plattformen entwickelt werden (also bewusst nirgendwo anders laufen sollen).

CLOUD-NATIVE APPLIKATIONEN

Referenzmodell einer Cloud-native Applikation



Quelle: Kratzke, N. A Brief History of Cloud Application Architectures. Appl. Sci. 2018, 8, 1368.

CLOUD NATIVE COMPUTING

CNA folgen häufig diesem Referenzmodell:

- *Service of Services*
- *Isolierter Storage*
- *Container oder FaaS Functions*
- *Microservice oder Serverless Architektur*

CLOUD-NATIVE

Die Begriffsdefinition für dieses Modul

CLOUD NATIVE COMPUTING

Definition

Eine Cloud-native Anwendung (CNA) ist ein **verteiltes**, **beobachtbares**, **elastisches** und auf horizontale **Skalierbarkeit** optimiertes **Service-of-Services**-System, das seinen Zustand in (einem Minimum an) **zustandsbehafteten Komponenten** isoliert.

Die Anwendung und jede in sich geschlossene **Bereitstellungseinheit** dieser Anwendung wird nach Cloud-fokussierten Designmustern entworfen und auf elastischen Self-Service-Plattformen betrieben.

Quelle: Kratzke, N. Cloud-native Computing, 2021, Hanser Verlag.

CLOUD-NATIVE

Die Begriffsdefinition für dieses Modul

CLOUD NATIVE COMPUTING

Beobachtbarkeit

Beobachtbarkeit bei Softwaresystemen bezieht sich typischerweise auf Telemetriedaten, die meist in drei Aspekte unterteilt werden: **Tracing** (verteilte Ablaufverfolgung von Transaktionen) ,**Monitoring** (Metriken, quantitative Informationen zu Prozessen) **Logging** (Protokollierung von Systemereignissen)

Elastizität

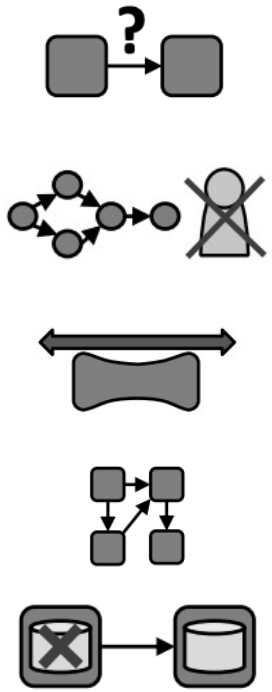
ist der Grad, in dem ein System in der Lage ist, sich an Workload-Änderungen anzupassen, indem es Ressourcen in einer autonomen Art und Weise provisioniert und deprovisioniert, sodass zu jedem Zeitpunkt die verfügbaren Ressourcen so gut wie möglich mit dem aktuellen Bedarf übereinstimmen.

Skalierbarkeit

Strukturelle Skalierbarkeit ist die Fähigkeit eines Systems, sich in einer gewählten Dimension ohne größere Änderungen an seiner Architektur zu erweitern. Unter **Lastskalierbarkeit** ist die Fähigkeit eines Systems zu verstehen, auch steigenden Datenverkehr bewältigen zu können

Service-of-Services

Bei Cloud-nativen Anwendungen werden einzelne Anwendungen als eine Suite kleiner loser gekoppelter Services (**Microservices**) dekomponiert, die jeweils in einem eigenen Prozess laufen. Diese Services können unabhängig voneinander **vollautomatisch aktualisiert** werden.



Standardisierte Bereitstellungseinheiten (Container)

Container können eine Softwarekomponente und alle ihre Abhängigkeiten in einem Format kapseln, das **selbstbeschreibend** und portabel ist, sodass jede konforme Laufzeitumgebung sie **ohne zusätzliche Abhängigkeiten** ausführen kann.

Zustand (Statefulness)

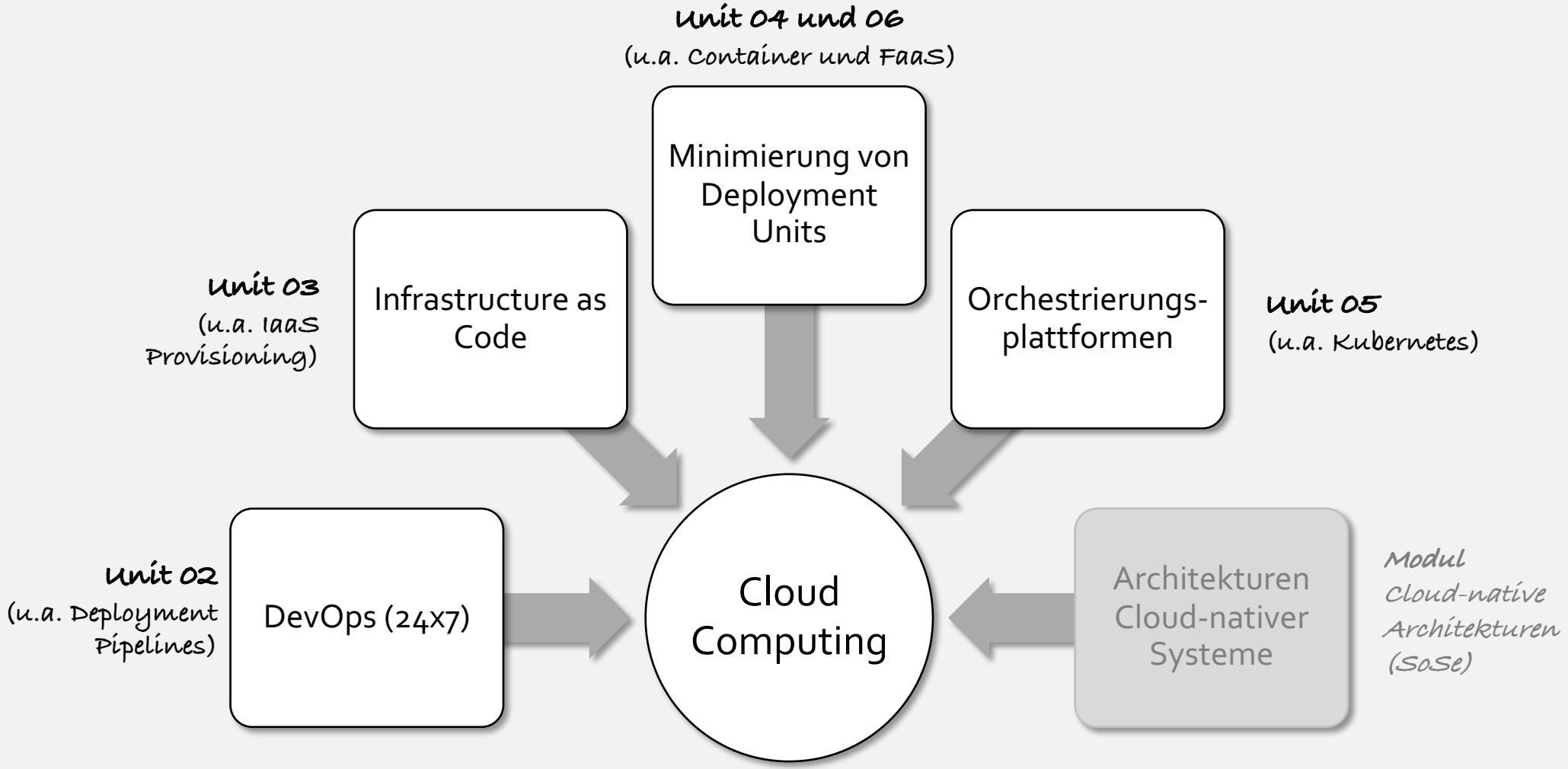
Zustandsbehaftete Komponenten (**Datenbanken**) werden für mehrere Instanzen einer skalierten Anwendungskomponente verwendet, die ihren internen Zustand synchronisieren, um ein einheitliches Verhalten zu bieten. Da die Skalierung zustandsbehafteter Komponenten meist aufwendiger ist als die Skalierung zustandsloser Komponenten, versucht man, Zustände in möglichst wenigen zustandsbehafteten Komponenten zu **isolieren**.

INHALTE

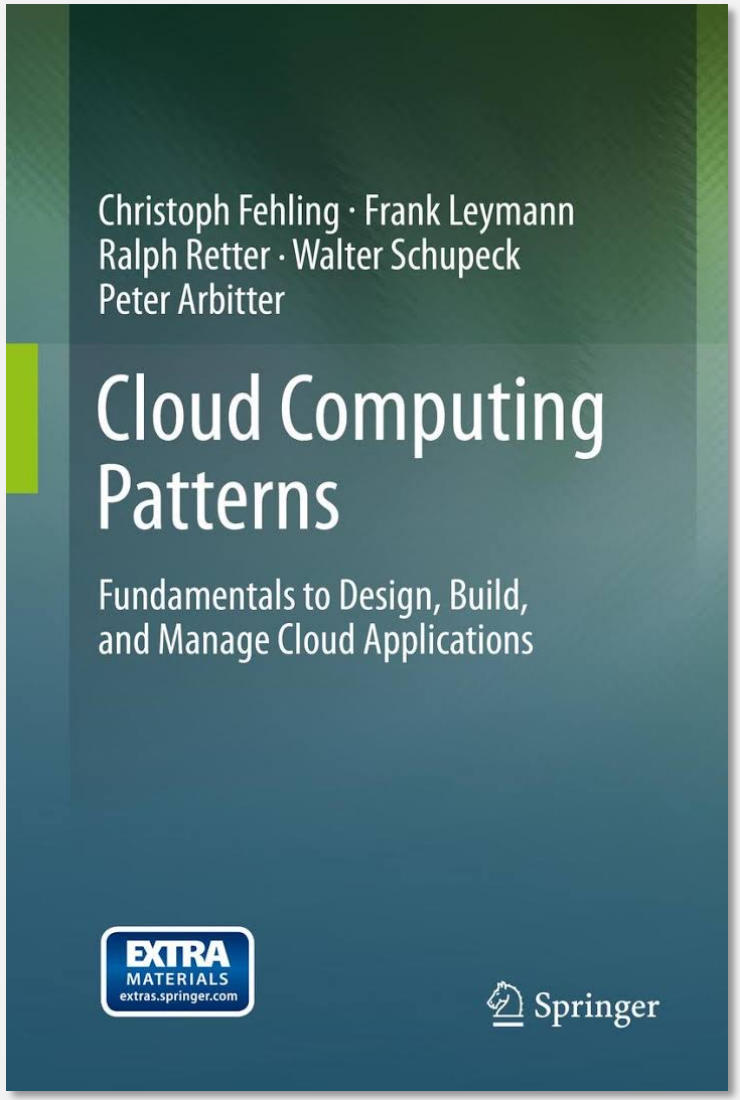
- Cloud Computing
- Cloud-Service Modelle
- Cloud-Ökonomie (Pay-as-you-Go)
- Eine kurze Architekturgeschichte der Cloud
- Cloud-native Anwendungen und Dienste
- Zusammenfassung

ZUSAMMENFASSUNG

und Ausblick auf die Module Cloud-native Programmierung und Cloud-native Architekturen



ZUM NACHLESEN



ZUM NACHLESEN

Paper

- Armbrust et al.; **Above The Clouds – A Berkley View on Cloud Computing (2009)**
  https://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf
- Mell, Grance; **NIST Definition of Cloud Computing (2011)**
  https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf
- Weinman; **Mathematical Proof of the Inevitability of Cloud Computing (2011)**
  http://www.joeweinman.com/resources/joe_weinman_inevitability_of_cloud.pdf
- Kratzke, Quint; **Understanding cloud-native applications after 10 years of cloud computing - A systematic mapping study (2017)**
  https://doi.org/10.1016/j.jss.2017.01.001
- Kratzke; **A Brief History of Cloud Application Architectures (2018)**
  https://www.mdpi.com/2076-3417/8/8/1368

TOOL-CHAIN

Dieses Moduls

Unit 01:

- Python
- Jupyterlab (THL Managed Service)

Unit 02:

- Gitlab CI (.gitlab-ci.yml, THL Managed Service)

Unit 03:

- VirtualBox
- Vagrant
- Google Cloud Platform (Managed Service)
- Terraform

Unit 04:

- Docker

Unit 05:

- Kubernetes
- Google Kubernetes Engine (Managed Service)

Unit 06:

- Kubeless
- Google Cloud Functions (Managed Service)

CLOUD NATIVE COMPUTING

Fast alle Programmierbeispiele basieren in diesem Modul auf Python.

Sie müssen aber kein Python-Experte sein, um diesen folgen zu können.

Komplexe Tool-Chain (typisch für CNA).

In den Praktika erhalten Sie die entsprechenden Quellen, wie diese zu installieren sind.

KONTAKT

Disclaimer

Nane Kratzke

+49 451 300-5549

nane.kratzke@th-luebeck.de

kratzke.mylab.th-luebeck.de