



Vorlesung „Service and Cloud Computing“

8. Cloud Computing - Erweiterte Konzepte

Dr.-Ing. Iris Braun

Cloud Computing – Erweiterte Konzepte

- Monitoring, Überwachungstools der Cloudplattformen
- Service Level Agreements
- LoadBalancing
- Autoscaling

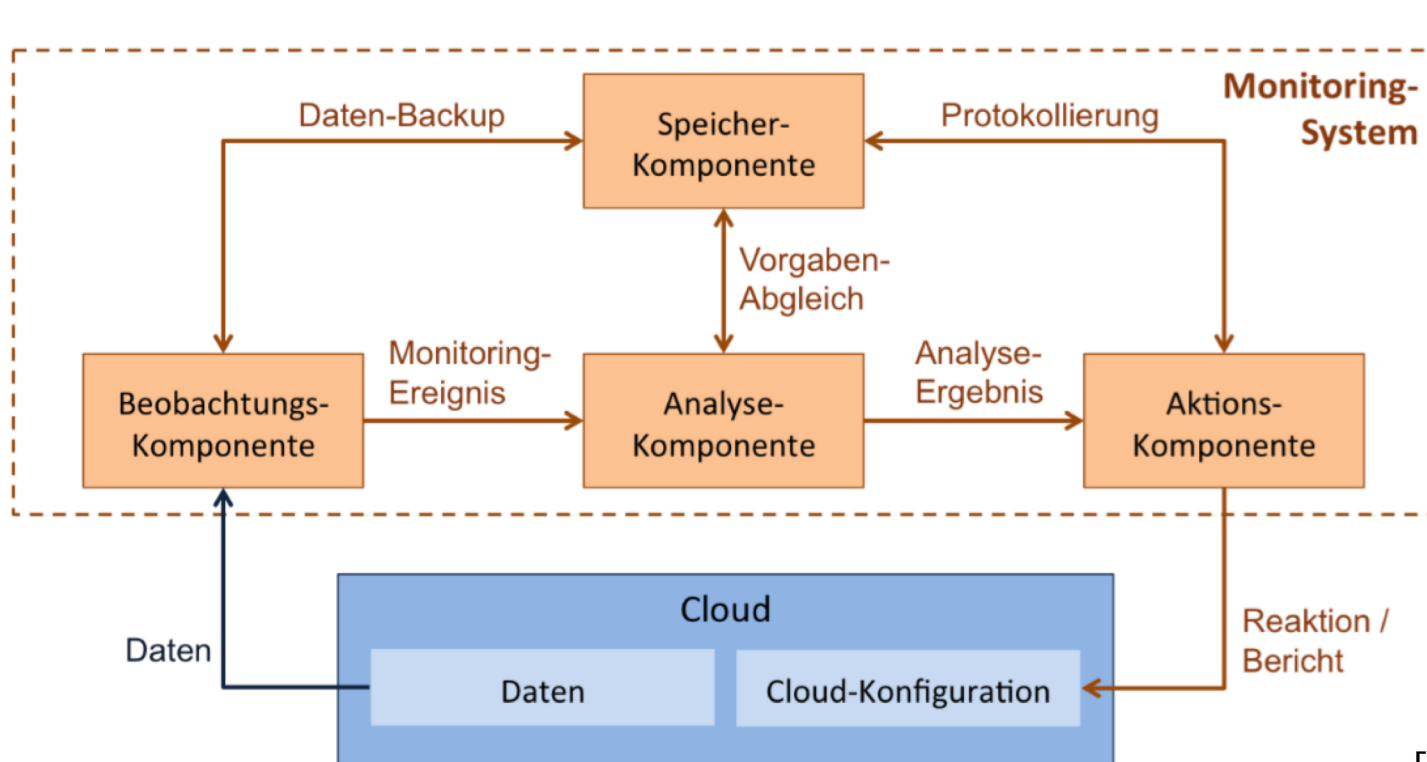
- Multicloudansteuerungstools – Terraform, Cloudify, Sparkleformation
- Konfigurationsmanagement + Provisioning – Ansible, Chef, Puppet, etc.
- DevOps, Docker

Skalierbarkeit

Stellen Sie sich vor, der von Ihnen entwickelte Service kommt an die Grenzen seiner Leistungsfähigkeit, weil plötzlich Millionen Nutzer darauf zugreifen. Was würden Sie (als erstes) tun, um die Performanz wieder zu verbessern?

- Die Leistungsfähigkeit meines Servers erhöhen durch mehr RAM und bessere Prozessoren.
- Meinen Service auf mehreren Servern/virtuellen Maschinen installieren und diese parallel betreiben.
- Meine Anwendungslogik anpassen oder eine performantere Programmiersprache oder DB nutzen.
- Meine Implementierung in mehrere Ebenen teilen - WebApp, Business-Logik, Persistenz-Schicht und die auf verschiedenen Servern laufen lassen.
- Meine Anwendung in viele kleine Services zerlegen und diese auf verschiedene Server verteilen.

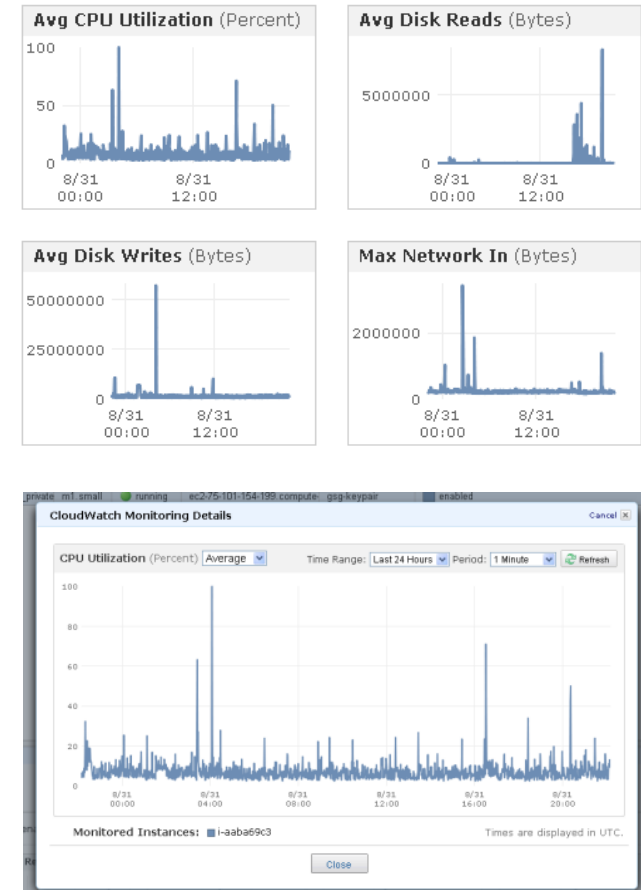
- „Monitor“** beschreibt ein System, „welches das Verhalten eines anderen Systems beobachtet und entscheidet, ob dieses Verhalten mit den gegebenen Vorgaben übereinstimmt“ [1]



[2,3]

AWS CloudWatch

- überwacht automatisch die Elastic Load Balancer für Metriken wie die eingehenden Requests und Latenz
- Überwachung von Schreib-/Leselatenzen, Datenbankinstanzen und verfügbaren Speicherplatz, Warteschlangen
- Definition von Alarmen, welche Benachrichtigungen auslösen oder automatische Aktionen durchführen, wenn die Metrik einen bestimmten Schwellenwert überschritten hat
- benutzerdefinierte System-, Anwendungs- und Protokolldateien in Amazon CloudWatch Logs



[4]

Google Stackdriver Monitoring

- Dashboard - umfangreiches Interface mit Diagrammen zur Performance, Auslastung, Gesundheit etc. der Anwendungen
- benutzerdefinierte Metriken auf Business- und Applikationsebene
- auch kompatibel mit AWS-API
- kompatibel mit bekannten Software-Lösungen wie Cassandra, Nginx, Apache Web Server, Elasticsearch
- Push-Benachrichtigungen und Alarme via Slack, PagerButy, HipChat, Campfire

Only \$300.00 and 59 days remain in your free trial. DISMISS [UPGRADE](#)

Google Cloud Platform ComputeTest 📧 🔔 🔍 👤

Home [CUSTOMIZE](#)

Dashboard

Project: ComputeTest

ID: computetest-1201 (#836304524219)

Resources

Compute


- Compute Engine
1 instance

Explore other services

- Tour the console
- API Enable APIs and get credentials like keys
- Deploy a prebuilt solution
- Deploy a Hello World app
- Create a Cloud Storage bucket

Compute Engine

CPU (%)

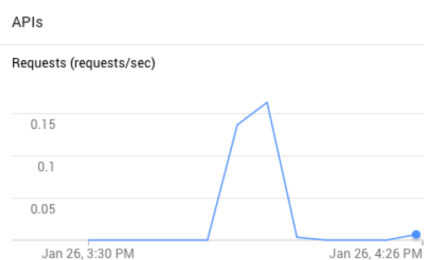


Jan 26, 3:30 PM Jan 26, 4:26 PM

CPU: 2.302

APIs

Requests (requests/sec)



Jan 26, 3:30 PM Jan 26, 4:26 PM

Requests: 0.0067

Google Cloud Platform status

🟢 All services normal

Billing

\$0.00
Approximate charges so far this month

News

- Google and Red Hat integrate OpenShift Dedicated and Google Cloud Platform to make adopting containers easier
Google Cloud Platform blog · 5 days ago
- Dataflow and open source - proposal to join the Apache Incubator
Google Cloud Platform blog · 6 days ago
- Build a mobile gaming analytics platform
Google Cloud Platform blog · 12 days ago

Free Trial Support

📧 Need help getting started with your Free Trial? Send us a question

Documentation

- Google Cloud Platform documentation
- Cloud Platform solutions
- Cloud Platform tutorials

Klassische Quality of Service Parameter (QoS)

- Verfügbarkeit, Service-Zeit
- Ausfallsicherheit
- Netzwerk-Anbindung/Bandbreite
- Antwortzeit (Response Time) einzelner Services (SaaS)
- Support-Zeiten

- Definition von Strafzahlungen/Rabatten bei Nichteinhaltung
- Monitoring durch Provider
- Meist keine formale Definition, sondern textuell
- Keine Standardisierung

Service Level Auswahl

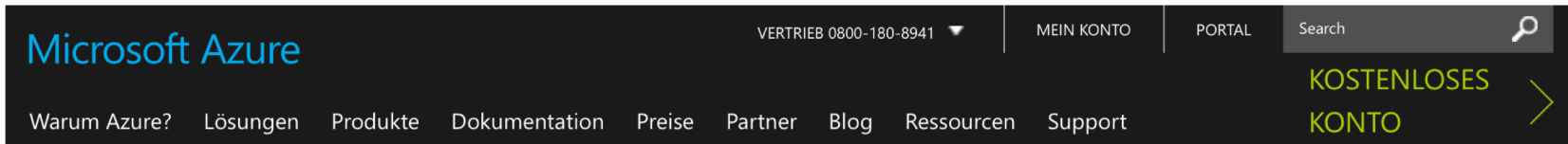


Definition	Service Level		
Service-Betrieb	7 x 24		
Bediente Service-Zeit	Mo-Fr 8.00 - 17.00 MEZ		
	Allround	Advanced	Superior
Verfügbarkeit der Infrastruktur im Rechenzentrum von Fujitsu, Messung der Verfügbarkeit ist die Internet-Schicht (auf Basis von Internet-Schicht-Wartungszeiten)			99.5 %
Einrichten neuer Kunden	10 Arbeitstage	10 Arbeitstage	10 Arbeitstage
Bereitstellung virtueller Server (~30Min.)	5 Arbeitstage	3 Arbeitstage	3 Arbeitstage
Bereitstellung dedizierter Server	10 Arbeitstage	5 Arbeitstage	5 Arbeitstage

Optionale Services:

- ✓ Administration Support
- ✓ Erstellen VMDK (VM Disk Format)
- ✓ Active Directory Integration
- ✓ System zurücksetzen (Reset)
- ✓ Wiederherstellung von Daten aus Snapshot

SLA: Viele Business-orientierte Cloud Dienstleister - im Bild ein Beispiel von Fujitsu - verwenden abgestufte Service Levels, so dass unterschiedliche Anforderungen bedient werden.



Microsoft Azure

VERTRIEB 0800-180-8941 | MEIN KONTO | PORTAL | Search

Warum Azure? | Lösungen | Produkte | Dokumentation | Preise | Partner | Blog | Ressourcen | Support

KOSTENLOSES KONTO

Berechnung der Monatlichen Betriebszeit und Servicelevel für Virtuelle Computer

„**Maximal Verfügbare Minuten**“ ist die Gesamtzahl der Minuten während eines Monats der Rechnungsstellung für alle internetseitigen Virtuellen Computer, für die zwei oder mehr Instanzen in der gleichen Verfügbarkeitsgruppe bereitgestellt sind. Die Maximal Verfügbaren Minuten werden ab dem Zeitpunkt gemessen, zu dem mindestens zwei Virtuelle Computer in der gleichen Verfügbarkeitsgruppe als Ergebnis einer vom Kunden initiierten Aktion gestartet sind, und bis zu dem Zeitpunkt, an dem der Kunde eine Aktion eingeleitet hat, die zum Stoppen oder Löschen der Virtuellen Computer führt.

„**Ausfallzeit**“ ist die Gesamtzahl der Minuten unter den Maximal Verfügbaren Minuten, für die keine externe Verbindung besteht.

Der „**Prozentsatz der Monatlichen Betriebszeit**“ für Virtuelle Computer wird für ein bestimmtes Microsoft Azure-Abonnement berechnet als die Maximal Verfügbaren Minuten minus Ausfallzeit geteilt durch die Maximal Verfügbaren Minuten im Monat der Rechnungsstellung. Der „Prozentsatz der monatlichen Betriebszeit“ wird mithilfe der folgenden Formel berechnet:

Prozentsatz der Monatlichen Betriebszeit = $(\text{Maximal verfügbare Minuten} - \text{Ausfallzeit}) / \text{Maximal verfügbare Minuten}$

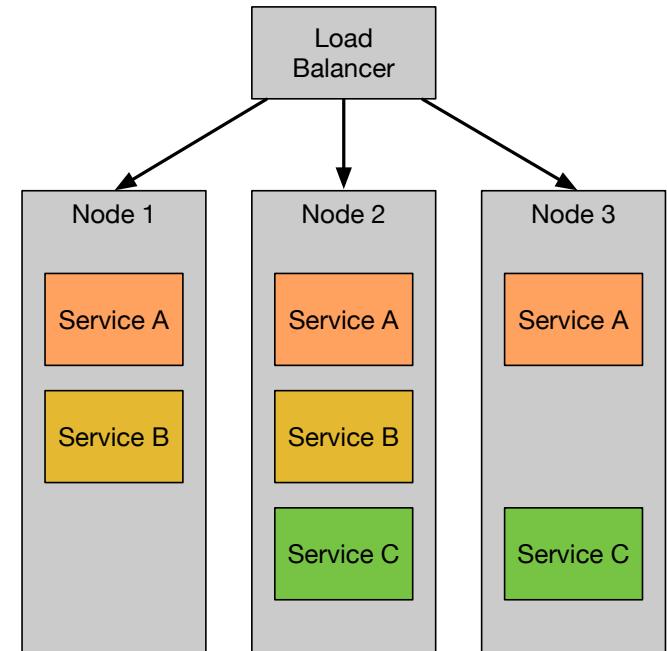
Die folgenden Servicelevel und Dienstgutschriften gelten für die Nutzung von Virtuellen Computern durch den Kunden:

PROZENTSATZ DER MONATLICHEN BETRIEBSZEIT	DIENSTGUTSCHRIFT
< 99.95%	10%
< 99%	25%

- Bedarfsorientierte Verteilung der Last auf mehrere Rechnerknoten
- Erhöhung der Ausfallsicherheit bei Infrastrukturproblemen
- Verteilung der eingehenden Requests auf mehrere virtuelle Maschinen, die redundante Service Deployments ausführen

2 Arten von Load Balancern:

- **Packet-based**
 - Für jedes ankommende Paket wird ein Zielhost ausgewählt
- **Flow-based**
 - Pro Verbindung wird ein Ziel für die Datenübertragung selektiert, das während der Verbindung konstant bleibt



- **Round Robin:** verwaltet alle Server in einer Liste und arbeitet nacheinander die Liste der Requests und der Server ab (FIFO), wurde der letzte Eintrag in der Liste erreicht, fängt er wieder von vorn an.
- **Gewichtetes Round Robin:** Wie Round-Robin, jedoch bekommen durch die Gewichtung gewisse Requests mehr Ressourcen zugeteilt.
- **Random:** Zufällige Zuteilung des Traffics.
- **Geringste Verbindungen:** Der LB zählt die offenen Verbindungen jedes Servers und sendet weiter an den Server mit der geringsten Last.
- **Geringster Traffic:** Der LB überwacht den Traffic des Servers und leitet die Anfrage an den mit dem geringsten ausgehenden Traffic weiter.
- **Geringste Latenz:** LB baut einen kurzen HTTP OPTIONS Request zum Backend Server auf und sendet die Anfrage zum Ersten weiter, der antwortet.
- **Source IP Hash:** Verbindungen sind basierend auf ihrer Source-IP auf Backend-Server aufgeteilt. Fällt ein Server aus wird eine neue IP zugeordnet. Anderenfalls werden Anfragen eines zugewiesenen Clients immer an den gleichen Webserver weitergeleitet.
- **URL Hash:** Ähnlich wie bei dem vorherigen Verfahren, nur das eine URL immer auf den gleichen Webserver zeigt.

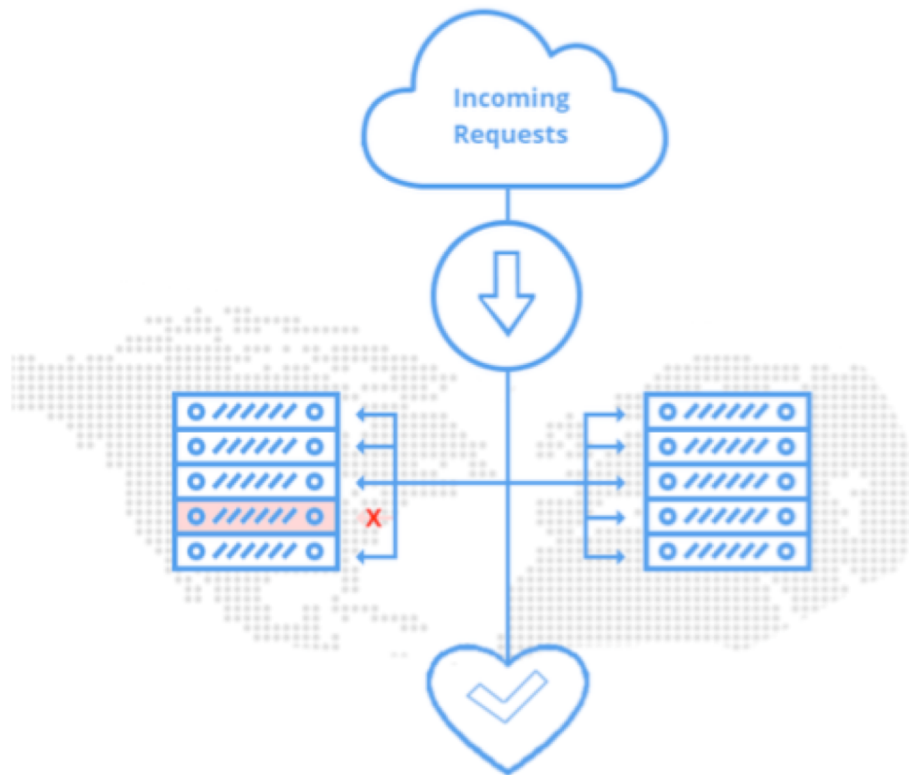
- Domain Name System (DNS) unterstützt mehrere Resource Records für eine einzelne Klasse:

```
www.example.com. 2400 IN A 141.76.40.2
```

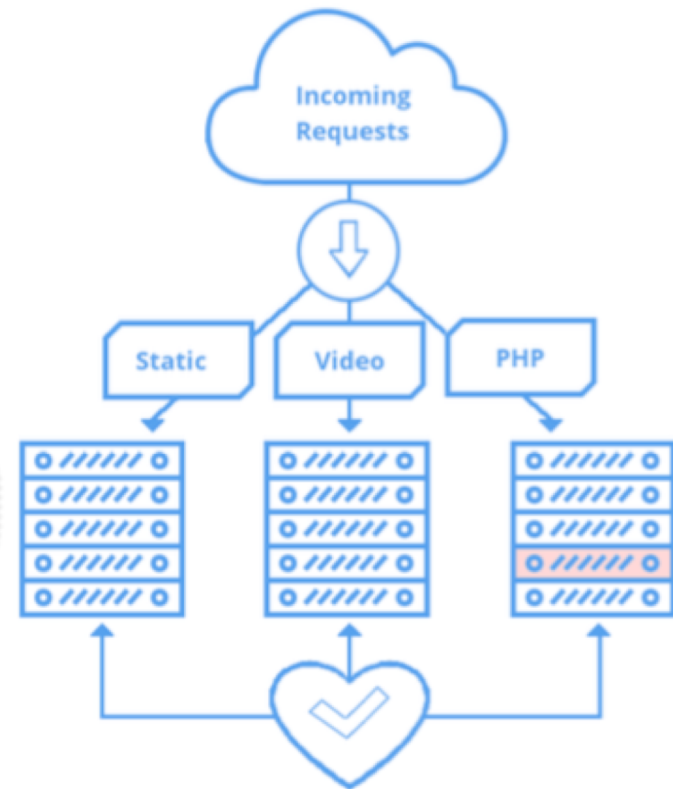
```
www.example.com. 2400 IN A 141.76.40.8
```

...

- DNS Server antwortet mit der Liste der Ergebnisse in unterschiedlicher Reihenfolge
- Sortierung der Ergebnisliste kann konfiguriert werden:
 - **Fixed:** vorgegebene Sortierung in Zonefile
 - **Random:** zufällig sortierte Liste
 - **Cyclic:** Round-Robin (Default-Einstellung)
- Einschränkungen:
 - Server-Status und -Auslastung werden nicht berücksichtigt
 - Ansatz wird durch Caching ausgehebelt



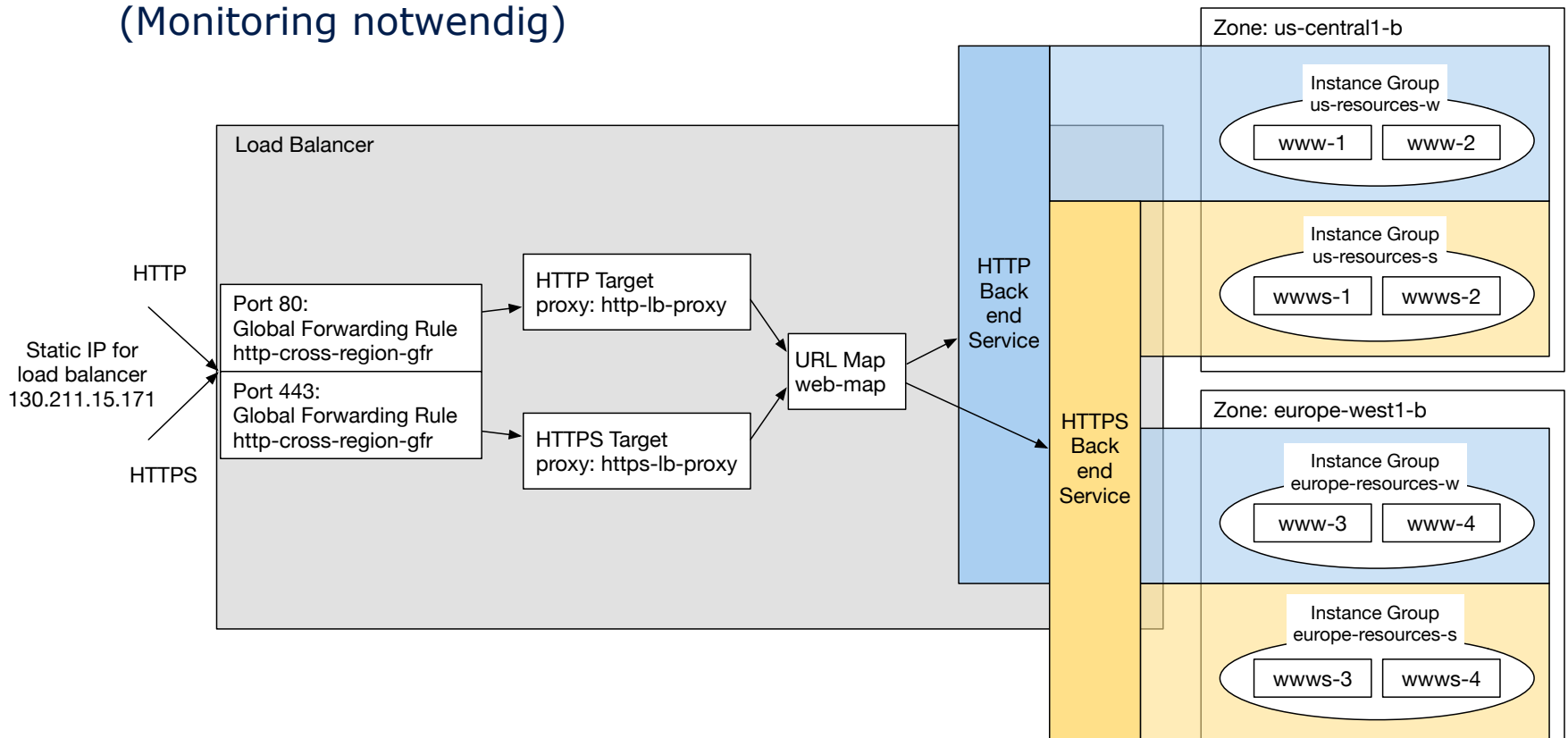
Cross-region LB



Content-based LB

Cross-Region Load Balancing

- Load Balancer mit statischer IP-Adresse
- Weiterleitung an den Server, der am nächsten beim Anfrager ist
- Load Balancer kann auch die Auslastung der Server berücksichtigen (Monitoring notwendig)

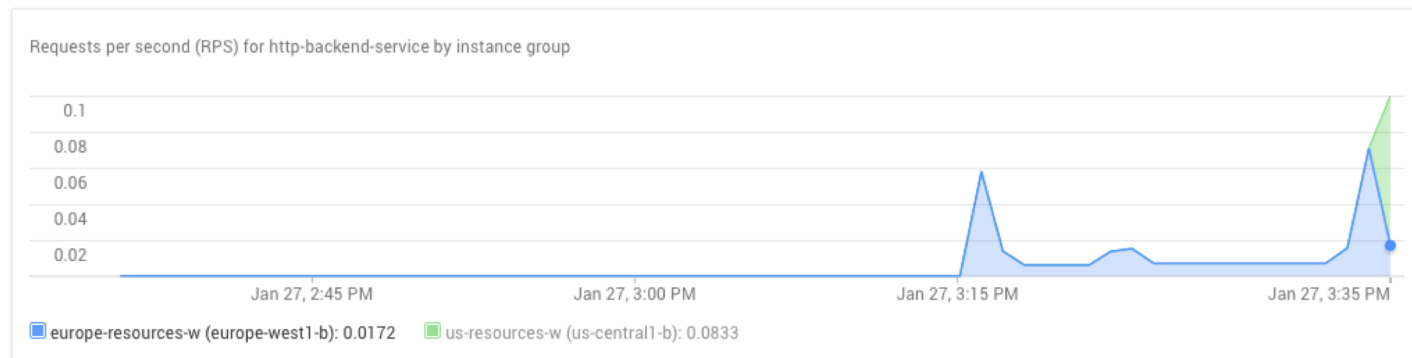


Forwarding rules Target proxies **Backend services** Certificates

http-backend-service

Activity for the last hour

1 hour 6h 12h 1 day 2d 4d 7d 14d 30d



Frontend Location
(Total inbound traffic)

Backend
(Just now)

Frontend Location	Backend	Health	CPU Utilization	Rate
North America 0.08 RPS	us-resources-w us-central1-b	2 of 2 instances healthy	23.1%	0.08 RPS
Europe 0.02 RPS	europe-resources-w europe-west1-b	2 of 2 instances healthy	0%	0.02 RPS

General properties



Protocol
HTTP



In use by
[cross-region-web-map](#)



➤ Requests werden in die Zone weitergeleitet, die geografisch am nächsten ist


- Innerhalb der Zone werden die Requests entsprechend der Auslastung verteilt
- Balancing Mode
Definiert, ob der Scheduler abhängig von der CPU-Last oder der Anzahl der eingehenden Requests pro Sekunde entscheidet
- Balancing mode = Utilization
Festlegung einer maximalen CPU-Auslastung
- Balancing mode = Rate
Maximale Anzahl von Requests pro Minute


Backends



Edit Backend  


Instance group 
europe-resources-w (europe-west1-b) 

Port numbers  
80

Balancing mode 
 Utilization
 Rate

Maximum utilization 
80 %

Maximum rate (Optional) 
Max total RPS. Leave blank for unlimited RPS per instance 

Capacity 
100 %

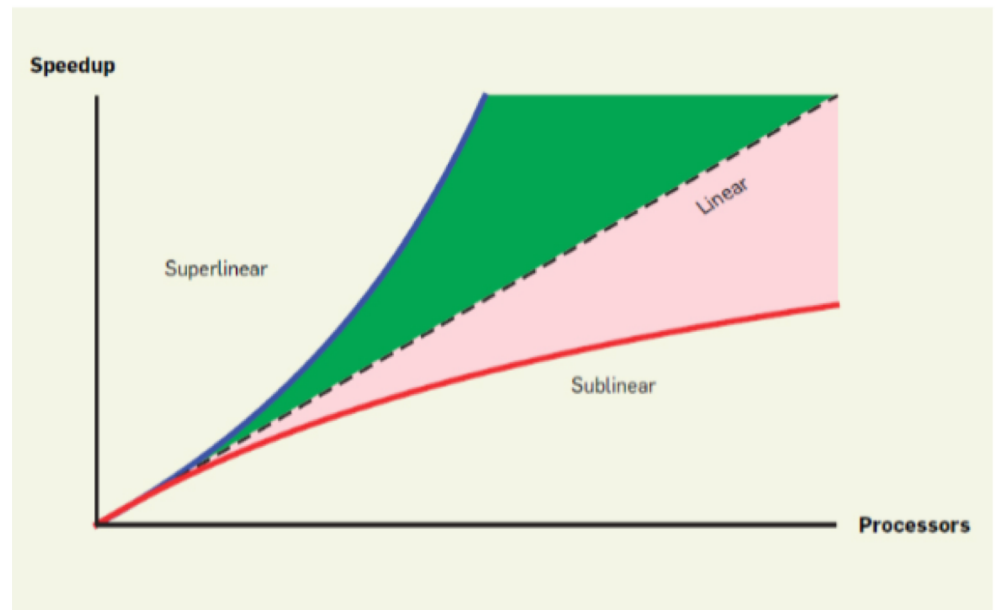
[+ Add backend](#)

Skalierbarkeit ist die Fähigkeit eines Systems, sich wachsenden Ansprüchen an die Leistungsfähigkeit anzupassen.

Der Skalierbarkeitsfaktor (*Speedup*) ist das Verhältnis von zusätzlicher Leistung zu zusätzlich eingesetzten Ressourcen.

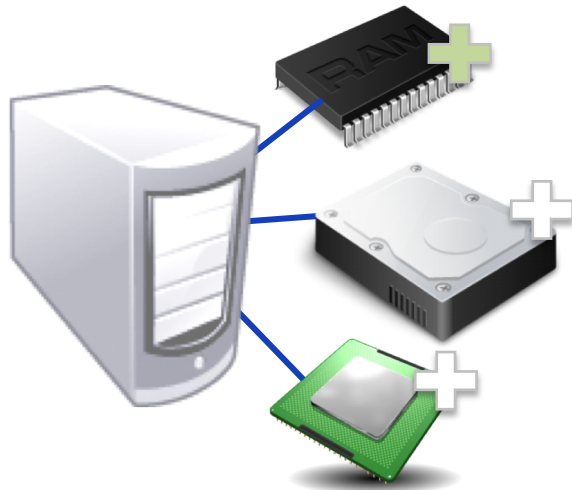
Universal Scalability Law
(Neil J. Gunther 1993 [9])

- Superlinear
- Linear
- Sublinear
- Negative



Vertikale Skalierung

- Ressourcen zu vorhandenen Knoten (Hosts) hinzufügen
- + Einfach umzusetzen
- Limitiert durch die Grenzen der eingesetzten Hardware
- Kostenintensiv



Horizontale Skalierung

- weitere Knoten (Hosts) hinzufügen
- + Grenzenlos erweiterbar
- + Kosten entstehen nur bei Bedarf
- Virtualisierung der Hardware-Ressourcen erforderlich
- Replikation der Rechnerknoten und Daten muss unterstützt werden



Autoscaling: automatisch neue Instanzen hinzufügen, wenn die Last steigt oder löschen, wenn der Bedarf nachlässt

- Erhöhung der Elastizität – Abfangen von Lastspitzen
- Kostenreduktion, wenn Last sinkt (Pay-per-Use)
- Autoscaling Policy definiert die Regeln für das An-/Abschalten
 - Verschiedene Metriken sind konfigurierbar
 - Z.B.: wenn die durchschnittliche CPU Nutzung aller Instanzen $\geq 60\%$, dann erstelle neue Instanz

Autohealing: automatisches Neustarten von VMs, wenn diese nicht mehr reagieren

Google Compute Engine

Autoscale based on [?]

For best results read [Configuring autos](#)

CPU usage

Target CPU usage [?]

Scaling dynamically creates or deletes

60 %

Minimum number of instances [?]

1

Maximum number of instances [?]

10

Cool-down period [?]

60 seconds

Autohealing

VMs in the group are recreated as needed. You can use a health check to recreate a VM if the health check finds the VM unresponsive. If you do not select a health check, VMs are recreated only when stopped. [Learn more](#)

Health check

http-basic-check

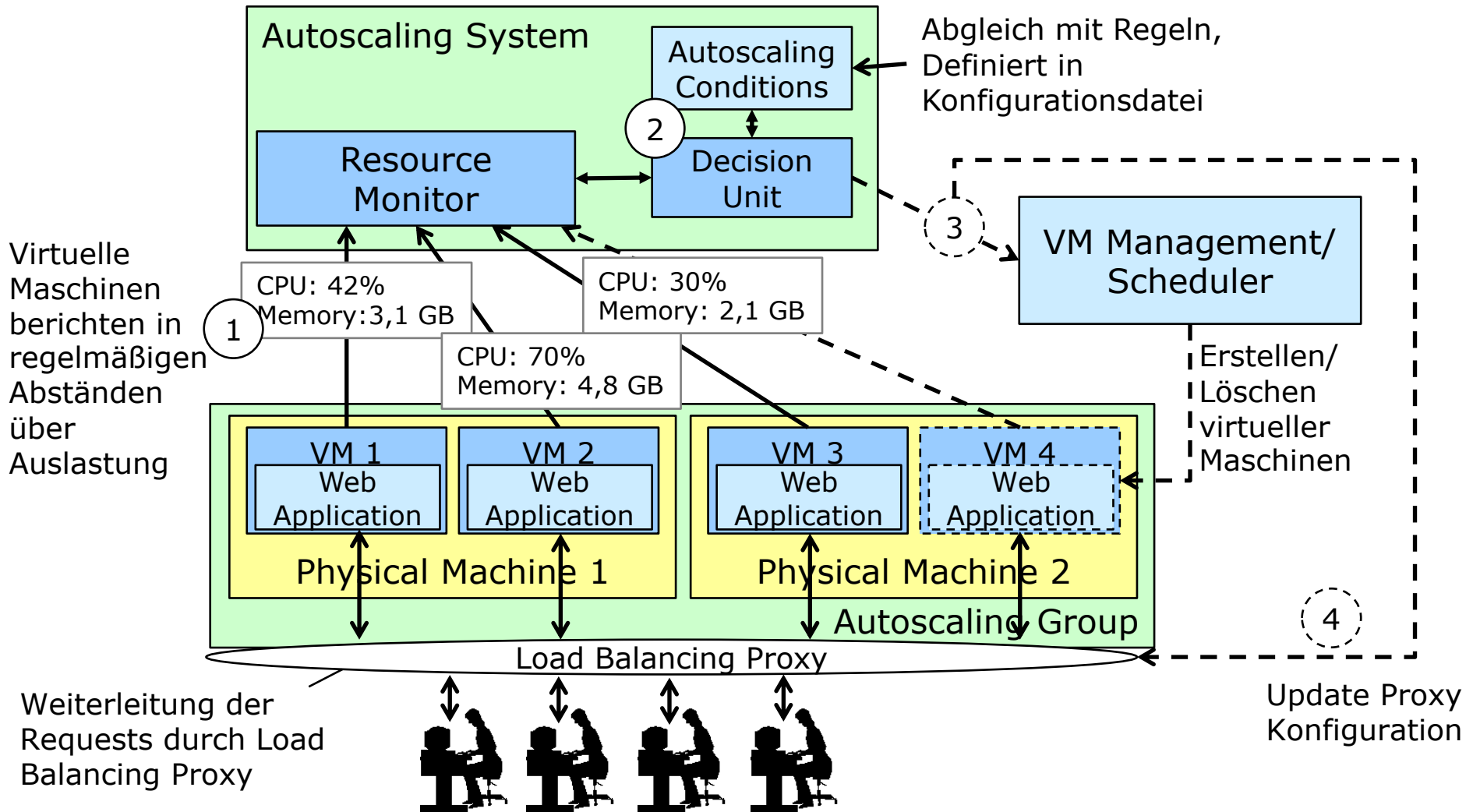
port: 80, timeout: 5s, check interval: 5s, unhealthy threshold: 2 attempts

Initial delay [?]

300 seconds

Create Cancel

CPU usage
HTTP load balancing usage
Monitoring metric
Multiple metrics



Health Checks

- Periodische Tests, ob die Instanzen noch korrekt arbeiten
- Als Ergebnis Werte HEALTHY oder UNHEALTHY

z. B. *http-basic-check*

- Sendet periodisch HTTP-Anfragen an Webserver auf virtueller Maschine
- Definition von Timeout, Testintervall und Grenzwert
- Wenn HTTP-Response 200 OK, dann Instanz „gesund“

← Create a health check

Autohealing instance groups and load balancing use health checks to detect when an instance is unresponsive. [Learn more](#)

Name [?]

Description (Optional)

Protocol

Port [?]

Request path [?]

∨ More

Health criteria

Define how health is determined: how often to check, how long to wait for a response, and how many successful or failed attempts are decisive.

Check interval [?]

 seconds

Timeout [?]

 seconds

Healthy threshold [?]

 consecutive successes

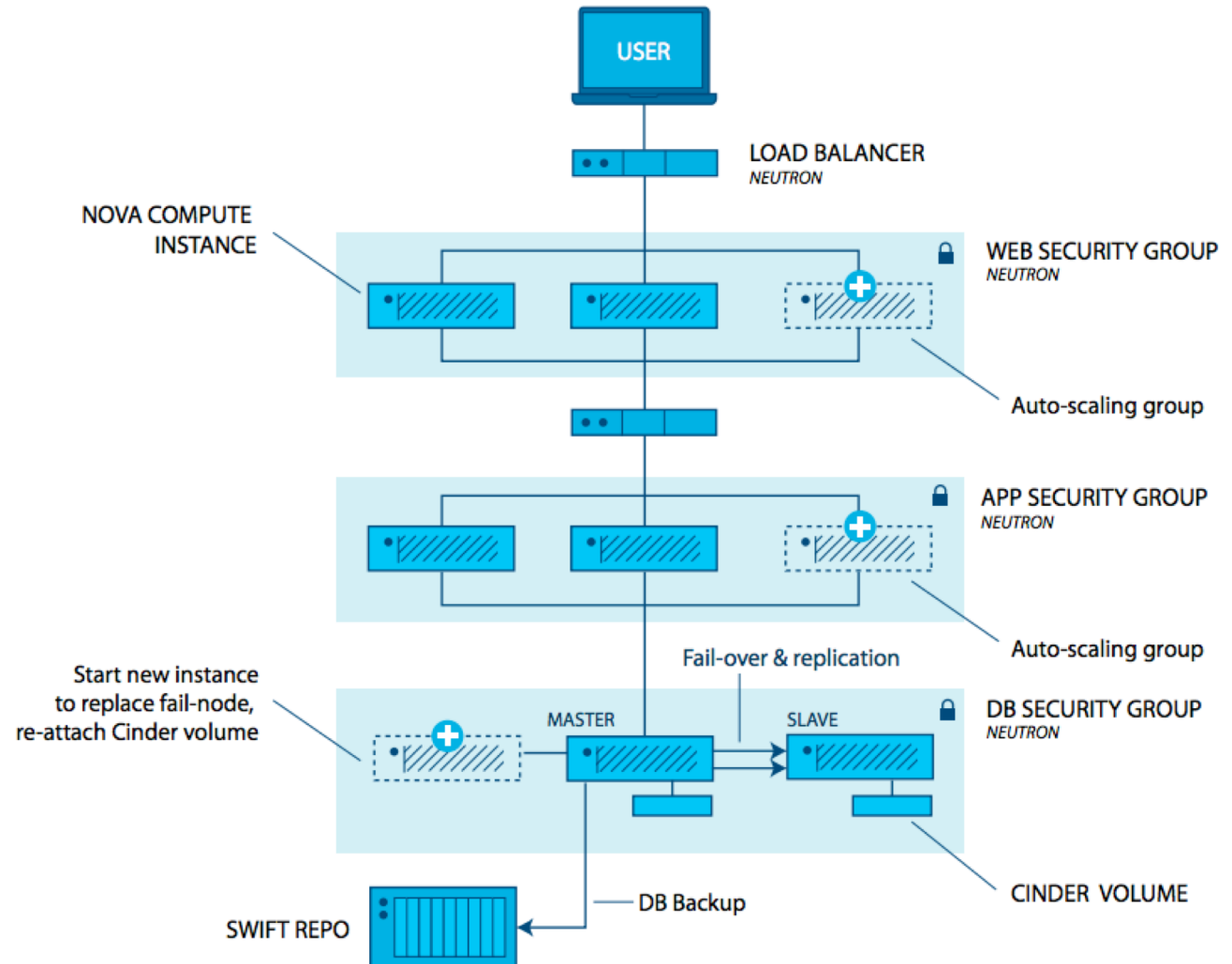
Unhealthy threshold [?]

 consecutive failures

Web
Application

Business
Logic

Database/
Persistence
Layer



- **Autoscaling und Load Balancing** setzen eine Automatisierung der Bereitstellung von Cloud-Instanzen voraus
- Automatisierung der Erstellung und Konfiguration von Cloud-Ressourcen (z.B. mit Terraform, sparkleformation, cloudify)
- Multicloud-Ansteuerung, bedienen APIs vieler Cloud-Anbieter (Google, AWS, Azure, OpenStack ...)
- **Konfigurations-Management**
 - Verwaltung von Systemkonfigurationen und Änderungen derselben zur Erhaltung von Integrität und Konsistenz der verwalteten Systeme
 - Management von einzelnen Instanzen und Clustern
 - Werkzeuge, Regelwerke, Prozeduren und verschiedene Techniken zur Verwaltung und Evaluierung von Änderungen, Durchgeführte Änderungen sind über die Zeit nachvollziehbar und zurücksetzbar

Konfiguration einer AWS- Instanz mit Terraform

```
PS C:\Automation\AWSExample1> terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but
will not be persisted to local or remote state storage.

The Terraform execution plan has been generated and is shown below.
Resources are shown in alphabetical order for quick scanning. Green resources
will be created (or destroyed and then created if an existing resource
exists), yellow resources are being changed in-place, and red resources
will be destroyed. Cyan entries are data sources to be read.

Note: You didn't specify an "-out" parameter to save this plan, so when
"apply" is called, Terraform can't guarantee this is what will execute.

+ aws_instance.ninjaxample1
  ami: "ami-13be557e"
  availability_zone: "<computed>"
  ebs_block_device.#: "<computed>"
  ephemeral_block_device.#: "<computed>"
  instance_state: "<computed>"
  instance_type: "t2.micro"
  key_name: "<computed>"
  network_interface_id: "<computed>"
  placement_group: "<computed>"
  private_dns: "<computed>"
  private_ip: "<computed>"
  public_dns: "<computed>"
  public_ip: "<computed>"
  root_block_device.#: "<computed>"
  security_groups.#: "<computed>"
  source_dest_check: "true"
  subnet_id: "<computed>"
  tenancy: "<computed>"
  vpc_security_group_ids.#: "<computed>"

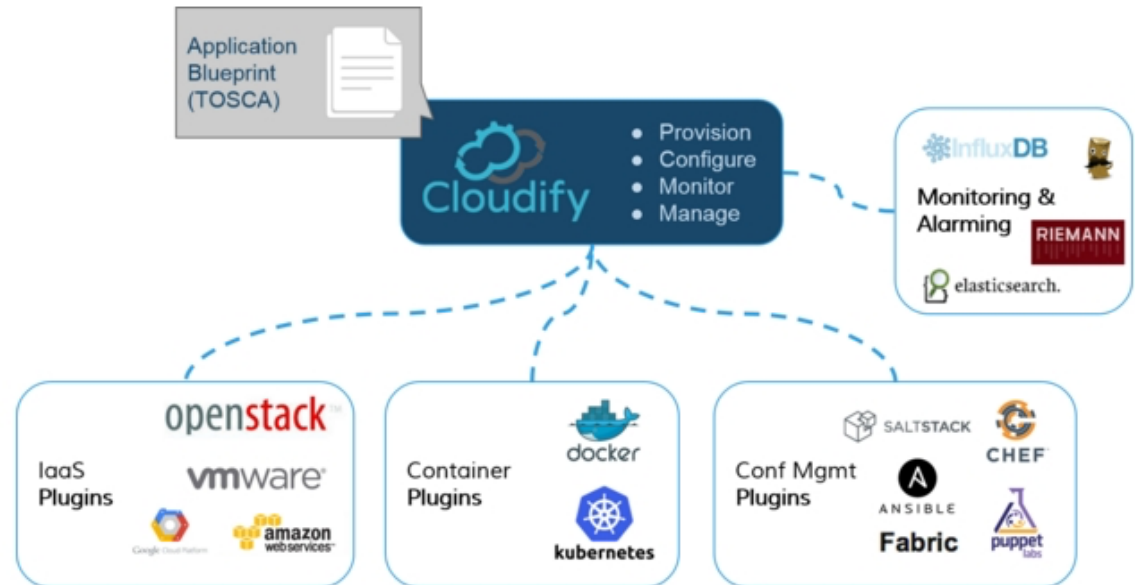
Plan: 1 to add, 0 to change, 0 to destroy.
PS C:\Automation\AWSExample1>
```

- **Provisioning**
 - Automatisiertes Deployment von Applikationen (SaaS)
 - beinhaltet Systemkonfiguration, Softwareinstallationen und die Übertragung aller notwendigen Daten für den Betrieb einer Applikation
 - Wichtig für Test-Automatisierung, Continuous Delivery and Integration
 - DevOps - "You build it, you run it" - Entwicklung und Betrieb der Anwendungen/Services durch ein Team
- **Tools**
 - Ansible, Chef, Puppet, Cloudify

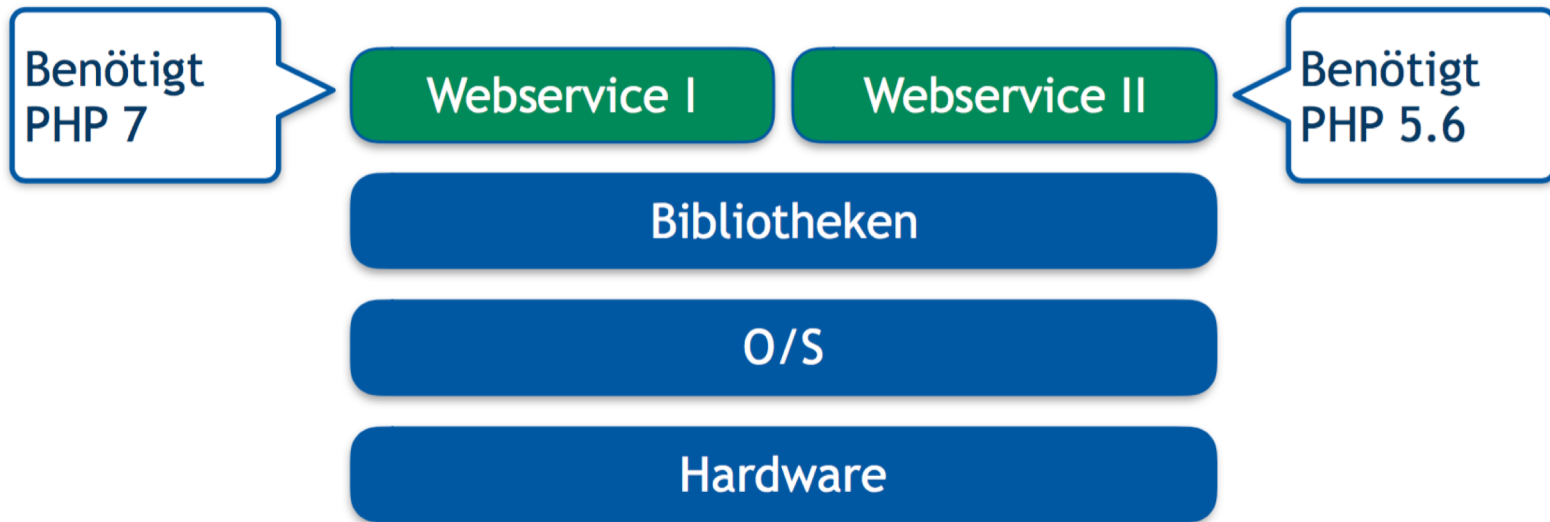
- **Chef-Konfiguration** besteht aus *Cookbooks*, welche einzelne *Recipes* enthalten.
- **Cookbook** bezeichnet eine vollständige Konfiguration für einen oder mehrere Nodes, welche via Chef-Server auf entsprechende Instanzen deployed werden kann.
- **Recipes** sind Unterkonfigurationen, beispielsweise für eine Datenbank, Web-Server oder andere Konfigurationen.
- **Chef-Server** ist die zentrale Verwaltungseinheit in der Architektur. Verwaltet Repository für die erstellten Cookbooks und enthält alle notwendigen Informationen über die zu verwaltenden Nodes und provisioniert diese.
- **Workstation** ist ein Rechner, von dem aus die gesamte Administration der Cookbooks und Netzwerke stattfindet. Hier werden die Konfigurationen für einzelne Nodes erstellt, getestet und auf den Chef-Server abgespeichert.
- **Nodes** sind physische Server oder Virtual Machines, die von Chef verwaltet werden.

Orchestrierung verschiedener Cloud-Konfigurations- und Provisioning-Tools zur "voll"-automatisierten Bereitstellung von Multi-Thier-Applikationen in der Cloud

- Erstellung der Instanzen und Cluster in Multicloud-Umgebungen
- Provisioning der Plattformdienste und Applikationen
- Monitoring, Self-Healing, Auto-Scaling, Disaster Recovery Automation

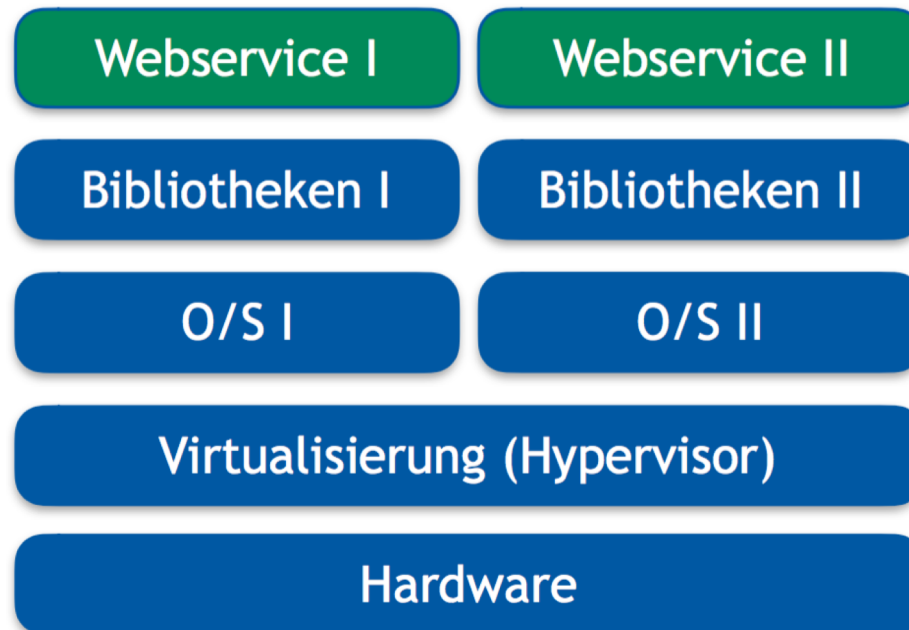


Klassisches Deployment von Webservices



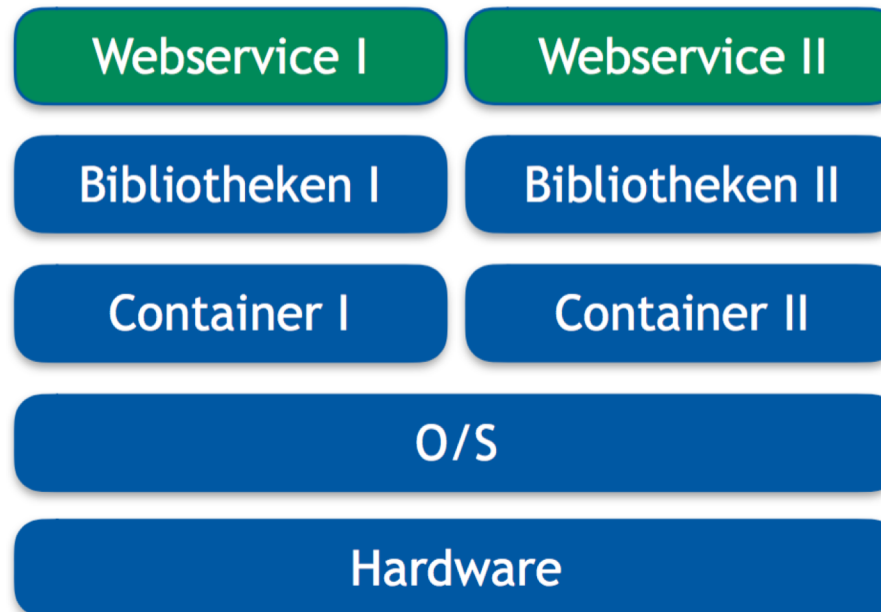
- Kollision, da unterschiedliche Versionen nötig

Lösung durch Virtualisierung

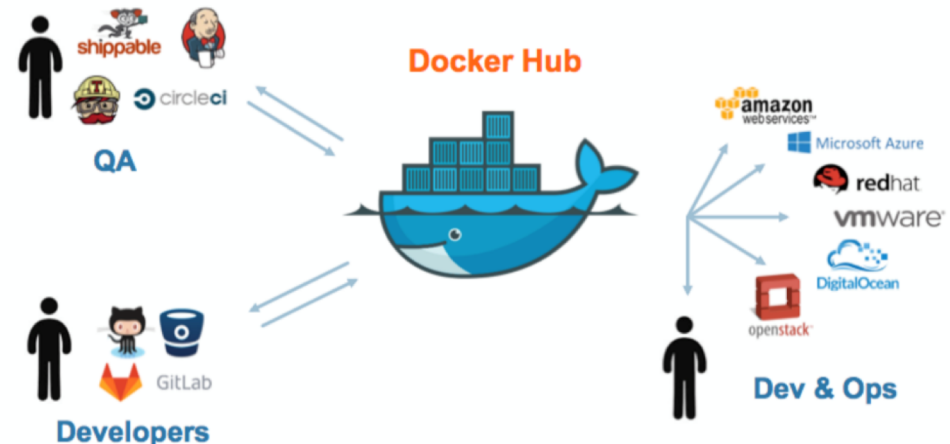


- Admin muss mehrere Betriebssysteme aktualisieren, absichern, etc.
- Overhead

Lösung durch Docker-Container



- Keine Virtualisierung des Betriebssystems notwendig
- Anwendung und alle benötigten Bibliotheken als Image in Container ausführbar



3-Phasen-Konzept

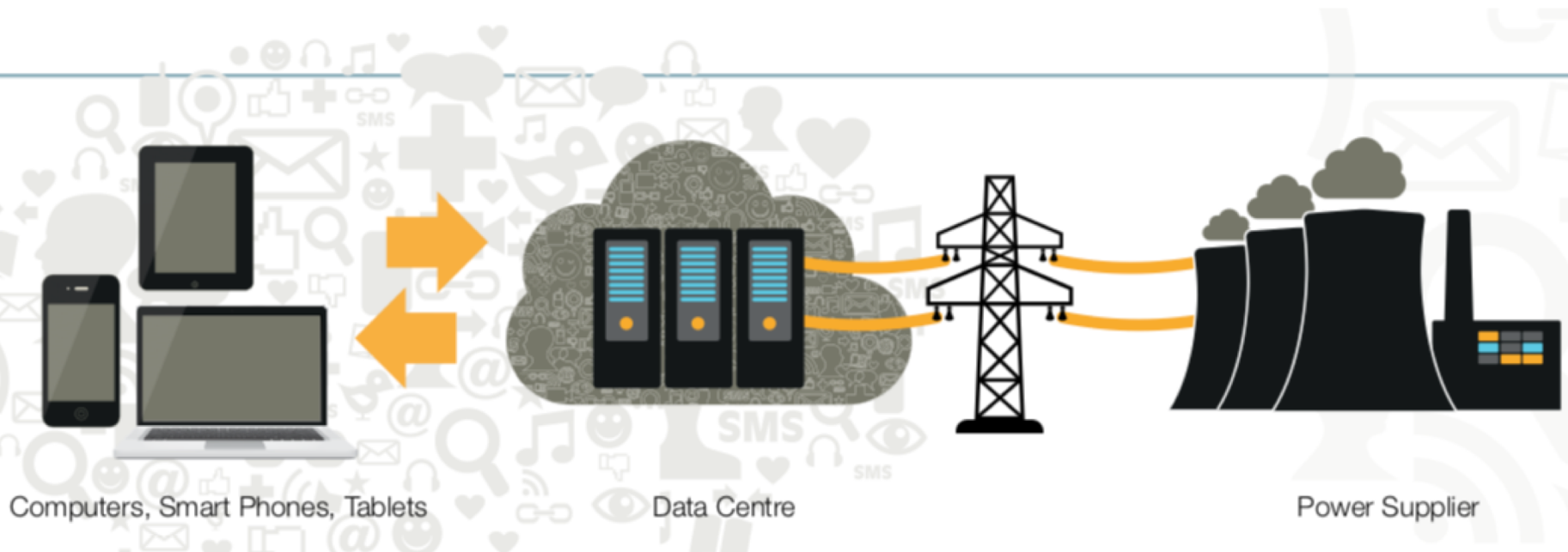
- **Build**: Erstellung von Images
- **Ship**: Verteilung der Images mit Docker Hub
- **Run**: Ausführung der Images mit der Docker Engine

Container (Instanz von Image) werden ohne Virtualisierung ausgeführt

- Nutzung von Linux-Technologien
- Namespaces, Control groups, Union file system
- Auf Windows + MAC Virtualisierung notwendig

- **Ziel:** Bedarfsgerechte Bereitstellung von Cloud-Ressourcen
 - Setzt Monitoring voraus, Health checks
 - Horizontale Skalierung und Load Balancing zur Vermeidung von Bottlenecks
 - Elastizität durch Autoscaling bei Lastspitzen
 - Ausfallsicherheit durch AutoHealing
- Voraussetzung ist Automatisierung der Cloud-Konfiguration und des Plattform- und Applikations-Provisioning
 - Terraform, sparkleformation, cloudify – Multicloud-Ansteuerung, Cloud-Konfiguration
 - Chef, Puppet, Ansible – Cloud-Konfigurationsmanagement und Provisioning
 - Docker – Deployment von Applikationen und benötigten Plattformdiensten in Containern

Wie kann Cloud Computing zum Klimaschutz beitragen?



Wie kann Cloud Computing zum Klimaschutz beitragen?

ein paar Zahlen/Fakten/Argumente:

- Strombedarf eines einzelnen Rechenzentrums für den Rechner-Betrieb, für Kühlsysteme und die sonstige Infrastruktur entspricht dem einer ganzen Kleinstadt. Fällt die Versorgung aus, springen Akkumulatoren und Dieselgeneratoren ein.
- Cloud-Rechenzentren verschwenden auch im Leerlauf enorme Energiemengen
- Nicht jeder, der Cloud-Dienste nutzt, würde ohne sie ein eigenes Rechenzentrum unterhalten.
- + Unternehmen im geschäftlichen Anwendungsbereich könnten je nach Größe zwischen 30 und 90 Prozent an Energiekosten einsparen. (Untersuchung im Auftrag von Microsoft)
- + Google, Yahoo und Facebook priorisieren die Nutzung von erneuerbaren Energien für die Versorgung Ihrer Rechenzentren
- + Erhöhung der Power Usage Effectiveness durch effizientere Kühlmethoden
- + Nutzung der Abwärme der Rechenzentren zum Heizen – Cloud&Heat

<https://irights.info/artikel/cloud-computing-wie-steht-es-um-die-okobilanz>
<https://storage.googleapis.com/planet4-international-stateless/2012/04/e7c8ff21-howcleanisyourcloud.pdf>
<https://www.golem.de/news/rechenzentrum-facebook-kuehlt-seine-daten-in-schweden-1306-99792.html>

1. Peters, D.K.: Automated Testing of Real-Time Systems. S.3. 1999.
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.43.4833&rep=rep1&type=pdf>
2. Ward, J. S., Barker, A. Observing the clouds: a survey and taxonomy of cloud computing. In: Journal of Cloud Computing: Advances, Systems and Applications. 2014.
<http://link.springer.com/article/10.1186/s13677-014-0024-2>
3. Lampesberger, H., Rady, M. Monitoring of Client-Cloud Interaction. S. 2ff. 2015.
http://link.springer.com/chapter/10.1007/978-3-319-17112-8_6#page-1
4. Amazon CloudWatch. https://aws.amazon.com/de/?nc2=h_lg
5. Google Google Compute Engine Documentation,
<https://cloud.google.com/compute/docs/>
6. Microsoft Azure. <https://azure.microsoft.com/de-de/>
7. Chef, Tutorials - Learn Chef, <https://learn.chef.io/tutorials/>
8. Cloudify http://getcloudify.org/configuration_management_ssh_cloud_orchestration_puppet_ansible_chef_open_source.html
9. Gunther, N. J.: How to Quantify Scalability - The Universal Scalability Law (USL). <http://www.perfdynamics.com/Manifesto/USLscalability.html>

Learn how to scale your applications with Google Compute Engine

<https://www.youtube.com/watch?v=TfbEwfYjKI4>

Load Balancing with Cloud Computing

https://www.youtube.com/watch?v=_xH5POea0Jc

From Zero to Docker - Tutorial for Beginners

<https://www.youtube.com/watch?v=JprTjTViaEA>

Learn Docker in 20 Minutes

<https://www.youtube.com/watch?v=wCTTHhehJbU>