Moduldokumentation

Modul Cloud Computing (cloud)

Simon Wächter

2019

Inhalt

[1 Einleitung 4](#_Toc21379144)

[1.1 Einleitung 4](#_Toc21379145)

[1.2 Lernziele 4](#_Toc21379146)

[1.3 Prüfungen 4](#_Toc21379147)

[2 Woche 1 5](#_Toc21379148)

[2.1 Inhalt 5](#_Toc21379149)

[2.2 Allgemeines 5](#_Toc21379150)

[2.2.1 Vorbereitung 5](#_Toc21379151)

[2.2.2 Aufbau der Lektionen 5](#_Toc21379152)

[2.2.3 Pausen und Administratives 6](#_Toc21379153)

[2.2.4 Erfahrungsnote 6](#_Toc21379154)

[2.2.5 Modulschlussprüfung 6](#_Toc21379155)

[2.2.6 Noten 7](#_Toc21379156)

[2.3 Spass mit Sicherheit 7](#_Toc21379157)

[2.3.1 Eine "Cloud-Engine"... 7](#_Toc21379158)

[2.4 Die Grundlagen der Cloud 7](#_Toc21379159)

[2.4.1 Die NIST-Kriterien 7](#_Toc21379160)

[2.4.2 Die Virtualisierungstypen 8](#_Toc21379161)

[2.4.3 Die Cloudarten 8](#_Toc21379162)

[2.4.4 Die Providermodelle 9](#_Toc21379163)

[2.4.5 Die Hypervisoren-Typen 9](#_Toc21379164)

[2.4.6 Begrifflichkeiten in einem Cluster 10](#_Toc21379165)

[2.5 Implementationen von Technologien 11](#_Toc21379166)

[2.5.1 Hypervisoren 11](#_Toc21379167)

[2.5.2 Cloud Frameworks und Abstraktionen 12](#_Toc21379168)

[2.5.3 Container 12](#_Toc21379169)

[2.6 Repetition und Aufgabe 12](#_Toc21379170)

[2.7 Termine 12](#_Toc21379171)

[3 Woche 2 13](#_Toc21379172)

[4 Woche 3 17](#_Toc21379173)

[4.1 Inhalt 17](#_Toc21379174)

[4.2 Allgemeines 17](#_Toc21379175)

[4.2.1 Vorbereitung 17](#_Toc21379176)

[4.3 Spass mit Cloud 17](#_Toc21379177)

[4.3.1 Was “Hardware Virtualisierung” wirklich ist... 17](#_Toc21379178)

[4.4 Aufbau von Virtualisierungen 18](#_Toc21379179)

[4.4.1 Containmenttypen 18](#_Toc21379180)

[4.4.2 Die Skalierungsarten 19](#_Toc21379181)

[4.4.3 Typische Hardware-Architekturen in der Cloud 19](#_Toc21379182)

[4.4.4 Die Typen von Compute-HW 21](#_Toc21379183)

[4.4.5 Eine typische Netzwerk-Verkabelung 21](#_Toc21379184)

[4.4.6 Die Typen von Storages 22](#_Toc21379185)

[4.4.7 Block und Objektspeicher 23](#_Toc21379186)

[4.4.8 Die Replikationsarten 23](#_Toc21379187)

[4.5 Repetition und Aufgabe 24](#_Toc21379188)

[4.6 Termine 24](#_Toc21379189)

[5 Woche 4 25](#_Toc21379190)

[5.1 Allgemeines 25](#_Toc21379191)

[5.1.1 Vorbereitung 25](#_Toc21379192)

[5.2 Spass mit Cloud 25](#_Toc21379193)

[5.2.1 It’s Magic… 25](#_Toc21379194)

[5.3 Storage Theorie 25](#_Toc21379195)

[5.3.1 Allgemeine Grundlagen 25](#_Toc21379196)

[5.3.2 Weitere relevante Features für Storagesysteme 26](#_Toc21379197)

[5.3.3 Thin und Thick Provisioning 27](#_Toc21379198)

[5.3.4 Die verschiedenen Fileformate für Disklaufwerke 28](#_Toc21379199)

[5.3.5 Die Verwendung von Snapshots 28](#_Toc21379200)

[5.4 Das CEPH-Projekt 29](#_Toc21379201)

[5.4.1 Ein Fallbeispiel von CERN 29](#_Toc21379202)

[5.4.2 Die Komponenten von CEPH 29](#_Toc21379203)

[5.4.3 Installation von CEPH 30](#_Toc21379204)

[5.4.4 Installation: Die Vorbereitung 30](#_Toc21379205)

[5.4.5 Installation: Cluster Aufbauen 31](#_Toc21379206)

[5.5 Repetition und Aufgabe 32](#_Toc21379207)

[5.6 Termine 32](#_Toc21379208)

[6 Woche 5 33](#_Toc21379209)

[6.1 Inhalt 33](#_Toc21379210)

[6.2 Allgemeines 33](#_Toc21379211)

[6.2.1 Vorbereitung 33](#_Toc21379212)

[6.2.2 Das Labor 33](#_Toc21379213)

[6.3 Spass mit Cloud 33](#_Toc21379214)

[6.3.1 Your next exam... 33](#_Toc21379215)

[6.4 Einführung in OpenStack 34](#_Toc21379216)

[6.4.1 Warum Openstack 34](#_Toc21379217)

[6.4.2 Der Aufbau von OpenStack 34](#_Toc21379218)

[6.4.3 Das Architekturkonzept 35](#_Toc21379219)

[6.4.4 Eine Übersicht über die typischen Nodes 35](#_Toc21379220)

[6.4.5 Die Komponentenübersicht pro Node 36](#_Toc21379221)

[6.5 Repetition und Aufgabe 36](#_Toc21379222)

[6.6 Termine 36](#_Toc21379223)

[7 Woche 6 37](#_Toc21379224)

# Einleitung

## Einleitung

Dieses Dokument stellt die Moduldokumentation für das Modul cloud dar. Allfällige Unterlagen sind im Modulordner zu finden.

## Lernziele

Das Modul beinhaltet folgende Lernziele:

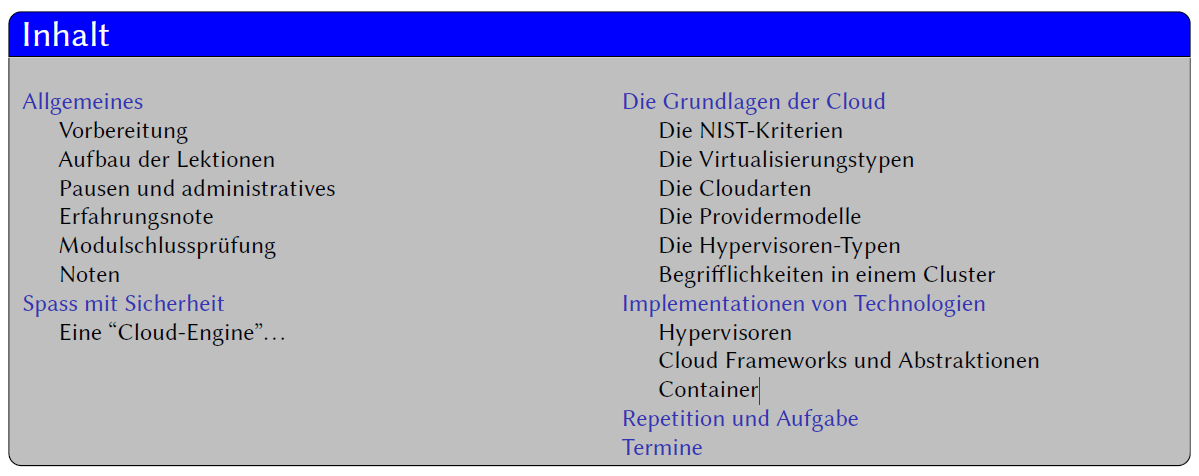
* Die Teilnehmer kennen die grundlegenden Konzepte und Standards einer Cloud und können diese benennen und einordnen.
* Die Teilnehmer kennen und verstehen die gängigen Typen von Virtualisierungstechnologien (Container sowie Hypervisoren) und können diese klassifizieren.
* Die Teilnehmer können eine einfache, vollwertige Cloud nach NIST aufbauen.
* Die Teilnehmer verstehen cloudgestützte Continuous Deployment Modelle.
* Die Teilnehmer kennen cloudspezifische Sicherheitsanforderungen und können einfache Aspekte umsetzen.

## Prüfungen

Die Modulnote setzt sich zu 100% aus einer Erfahrungsnote und einer mündlichen Modulschlussprüfung zu je 50% zusammen. Mit den Frequent Tests + Laborübungen kann das Mittel der Erfahrungsnote verbessert werden.

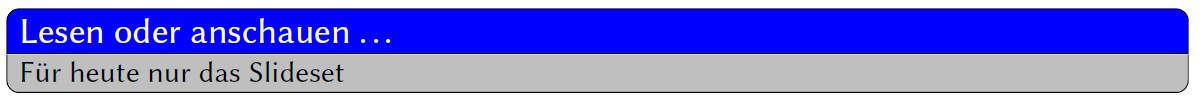
# Woche 1

## Inhalt



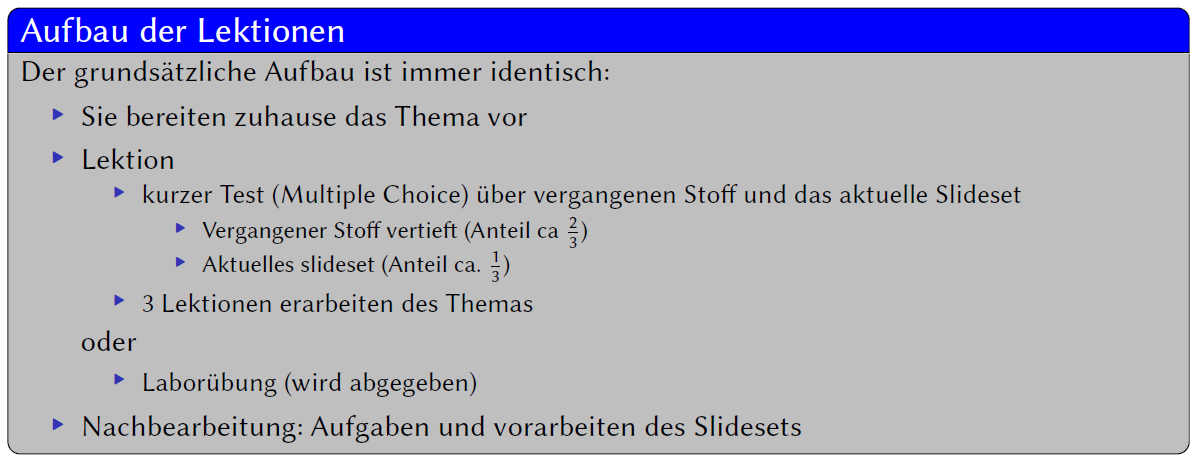
## Allgemeines

### Vorbereitung



Lesen sie jeweils vor der Lektion das Slideset durch. Es enthält den roten Faden, den wir in der Lektion haben werden. Je mehr Leute vorbereitet in die Lektionen kommen, desto interessanter wird die Lektion.

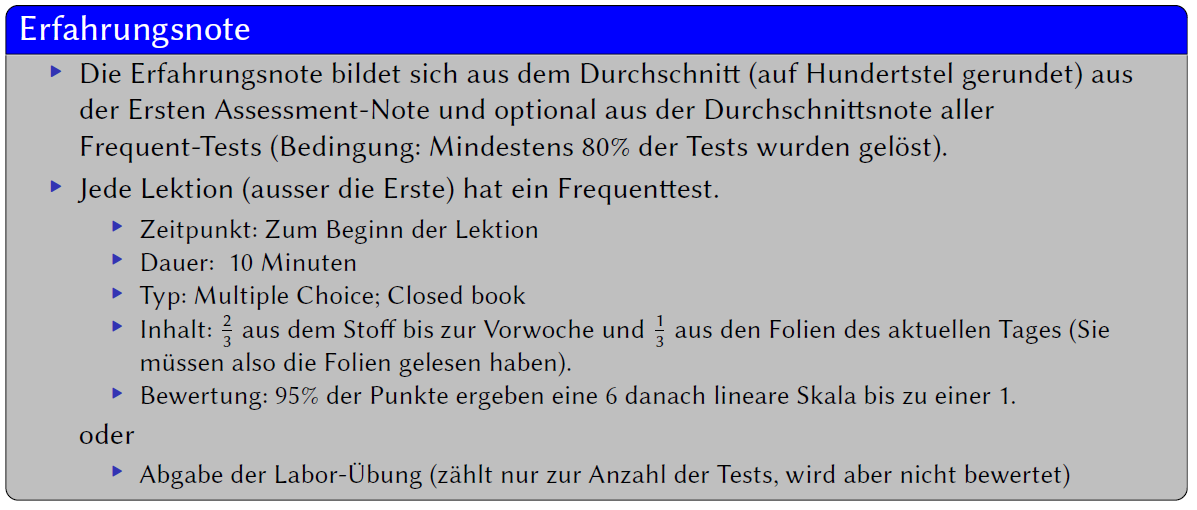
### Aufbau der Lektionen



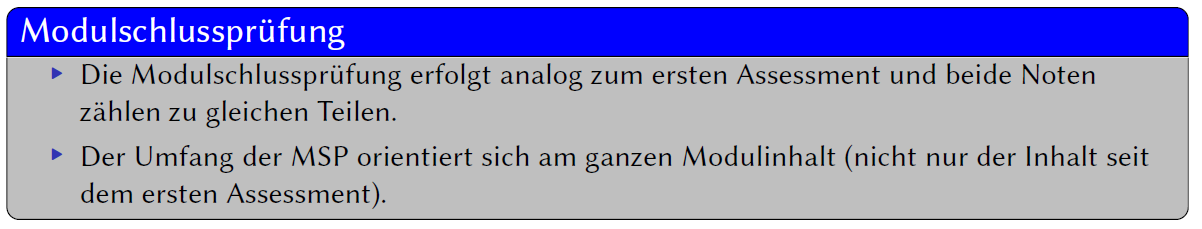
### Pausen und Administratives



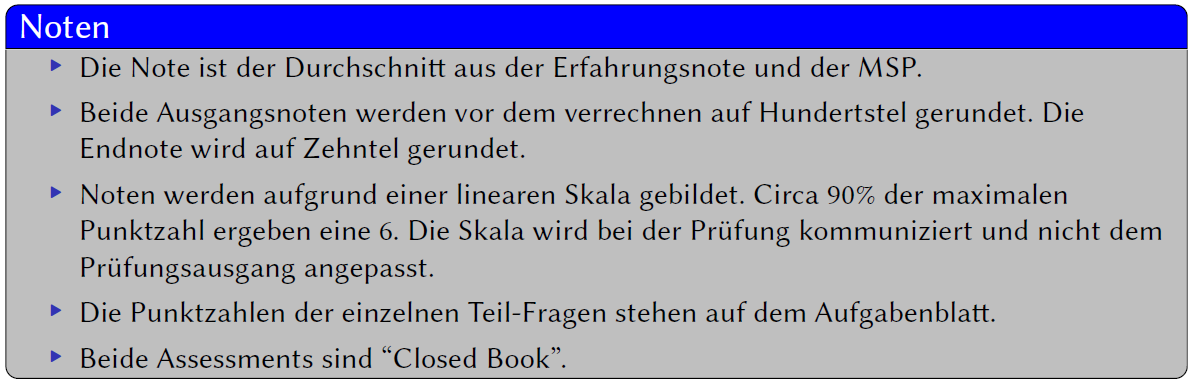
### Erfahrungsnote



### Modulschlussprüfung

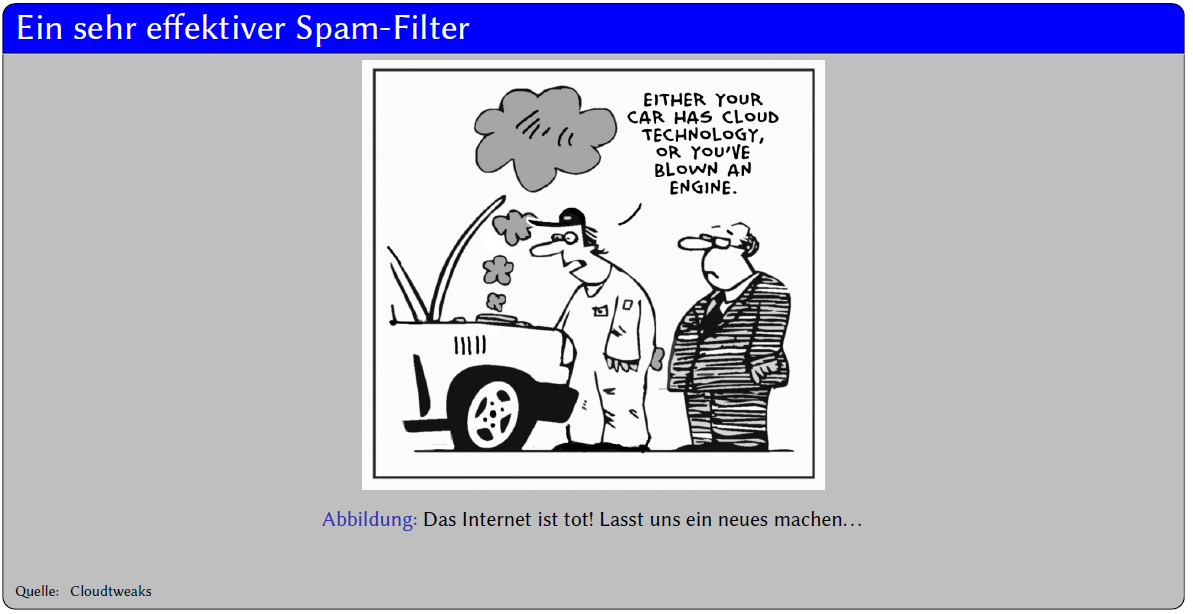


### Noten



## Spass mit Sicherheit

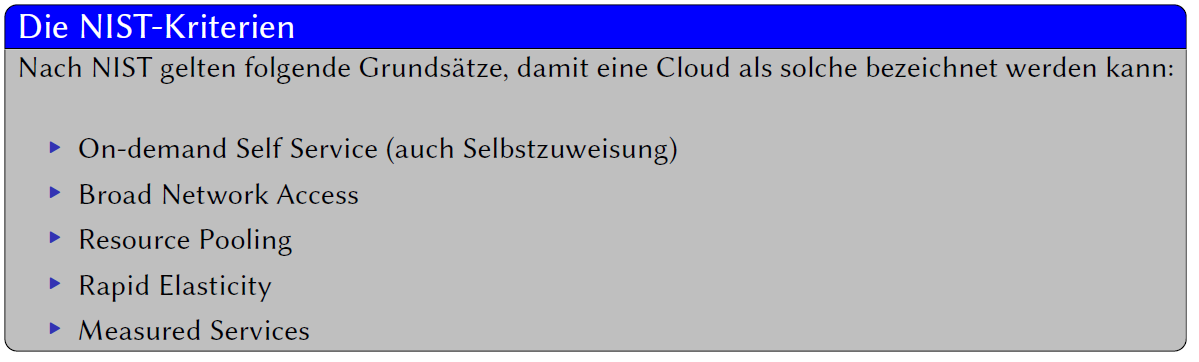
### Eine "Cloud-Engine"...



Heute haben alle das Gefühl, dass eine Applikation oder eine Webseite eine Cloud sei (oder “Cloud Ready” sei) sobald irgendetwas mit einer Cloud involviert ist (zum Beispiel eine Applikation ist auf Amazon Web Services installiert). Dem ist aber nicht so!

## Die Grundlagen der Cloud

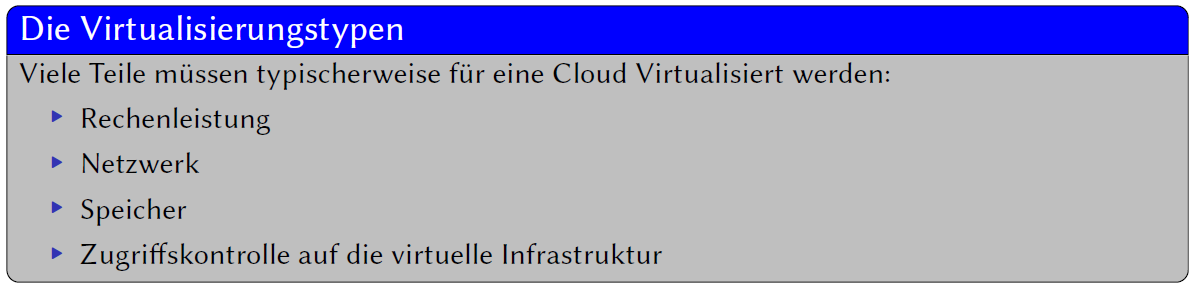
### Die NIST-Kriterien



Nach NIST gelten folgende Grundsätze, damit eine Cloud ihren Namen verdient:

* On-demand Self Service (auch Selbstzuweisung): Die Ressourcen (z. B. Rechenleistung, Storage) werden automatisch und ohne Interaktion eines Menschen bereitgestellt.
* Broad Network Access: Die Services sind über ein Netz verfügbar und nicht an einen bestimmten Client gebunden.
* Resource Pooling: Die Ressourcen des Anbieters liegen in einem Pool vor, aus dem sich viele Anwender bedienen können (Multi-Tenant Modell).
* Rapid Elasticity: Die Services können schnell und elastisch (eigentlich vergrössern und verkleinern; wobei manche Parameter nur sehr umständlich in der Praxis geändert werden können. Dies gilt insbesondere für das Verkleinern) zur Verfügung gestellt werden. Entweder ein Kunde kann Sie dynamisch hinzufügen oder gewisse Modelle erlauben es sogar automatisch (z.B. auslastungsgesteuert) Dienste/Produkte hoch- und runterfahren respektive zu skalieren.
* Measured Services: Die Ressourcennutzung kann gemessen und entsprechend auch den Cloud-Anwendern zur Verfügung gestellt werden.

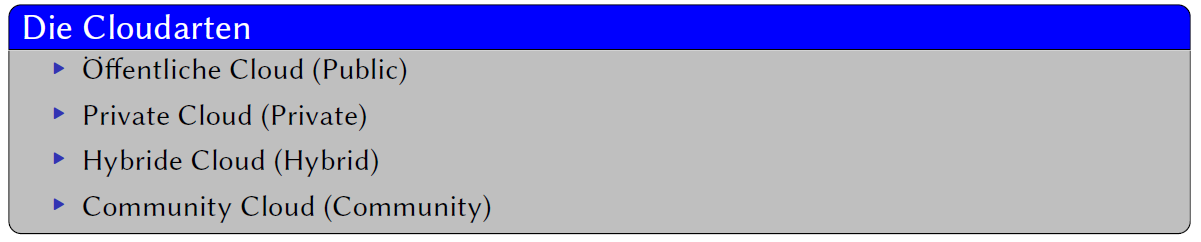
### Die Virtualisierungstypen



Viele Teile müssen typischerweise für eine Cloud virtualisiert werden:

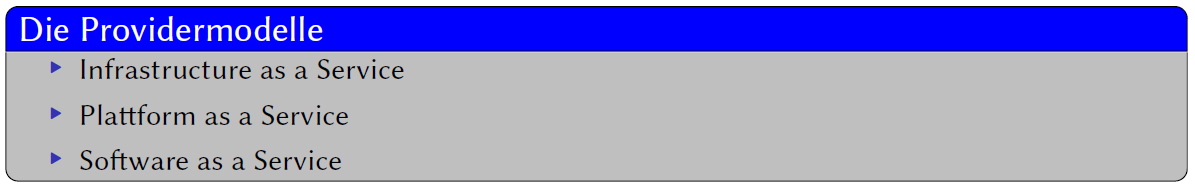
* Rechenleistung: Die Rechenleistung einer Hostmaschine muss virtualisiert werden, damit sich mehrere Virtuellen Maschinen (oder auch Container) die Rechenleistung teilen können.
* Netzwerk: Das Netzwerk eines Hostrechners muss ebenfalls virtualisiert werden. Einfachste Formen der Virtualisierung verwenden einfach Das Hostnetzwerk als Ressource. Um komplexe Strukturen virtualisieren zu können ist es jedoch notwendig, dass die Hosts Untereinander (beinahe) beliebig viele Netzwerke Sharen können. Das wird häufig über einen sogenannten “Trunked Port” erreicht. Jedes VLAN auf diesem Port ist dann stellvertretend für eine Broadcastdomäne in der virtuellen Welt.
* Speicher: Speicher ist auf modernen Rechnerplatformen bereits virtualisiert.
* Zugriffskontrolle auf die virtuelle Infrastruktur: Das ist wohl einer der Knackpunkte, die gerne vergessen werden. Typische “Bare Metal Server” (Traditionellle Server in einer Blechbox, also nicht virtualisiert), stehen in einem Rechenzentrum und unauthorisiertes Personal kommt werder an die Physischen Ports (z.B. Konsole USB usw.) an und kann auch nicht den Server oder Teile davon mitnehmen. In der virtuellen Welt müssen alle diese Zugänge (erneut) separat gesichert werden.

### Die Cloudarten



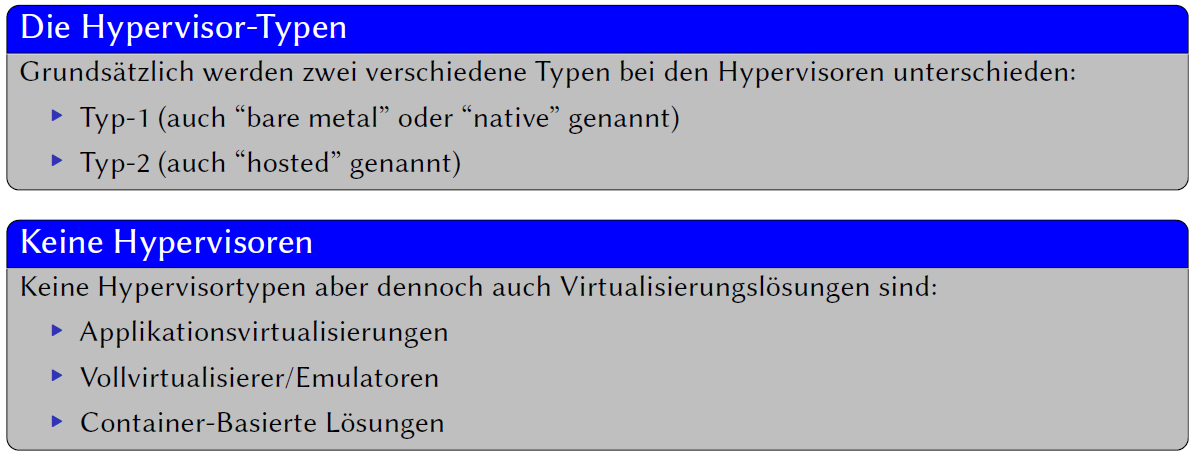
* Öffentliche Cloud (Public): Ein Provider, der Clouddienstleistungen öffentlich zur Verfügung stellt (wie Amazon, Google, Microsoft oder IBM) betreibt eine öffentliche Cloud.
* Private Cloud (Private): Eine Firma, die eine Cloud nur für sich und seine Mitarbeiter betreibt ist, verfügt über eine private Cloud. Bitte beachten Sie, dass aufgrund der NIST-Kriterien die Server nicht zwingend aus dem Internet erreichbar sein müssen.
* Hybride Cloud (Hybrid): Eine Hybride Cloud ist ein Konstrukt aus öffentlicher und privater Cloud, zwischen denen orchestriert wird. Das heisst, dass jede Maschine (oder zumindest bestimmte Klassen von Maschinen) auf beiden Clouds liegen können und, dass zwischen den Clouds eine Orchestrierung besteht, die es erlaubt die Maschinen zwischen den Clouds zu verschieben. Diese Clouds werden typischerweise verwendet um spitzen in der Ressourcennutzung zu decken und damit den Hardwareanteil der privaten Cloud möglichst gering zu halten.
* Community Cloud (Community): Eine Commuity Cloud ist eine öffentliche Cloud, die nur für bestimmte Benutzergruppen zugänglich ist (zum Beispiel für Bund, Kantonale Behörden und Gemeinden).

### Die Providermodelle



* Infrastructure as a Service (IaaS): Es werden nur die Hardware-Ressourcen zur Verfügung gestellt (CPU, RAM, Disk). Für Betriebssysteme oder Software ist der Kunde zuständig. Manchmal wird auch das Bereitstellen eines Betriebssystems hinzugerechnet.
* Platform as a Service (PaaS): Es wird eine Infrastruktur (siehe IaaS) und mindestens das Betriebssystem bereitgestellt. Typischerweise können auch generische Dienste wie ein SQL-Server oder ein Webserver vorinstalliert sein. Umstritten ist, ob auch die Lizenzen dazu gehören (NIST schweigt sich hierüber aus)
* Software as a Service (SaaS): Weitergehendes bereitstellen einer fertig vorkonfigurierten Software.

### Die Hypervisoren-Typen



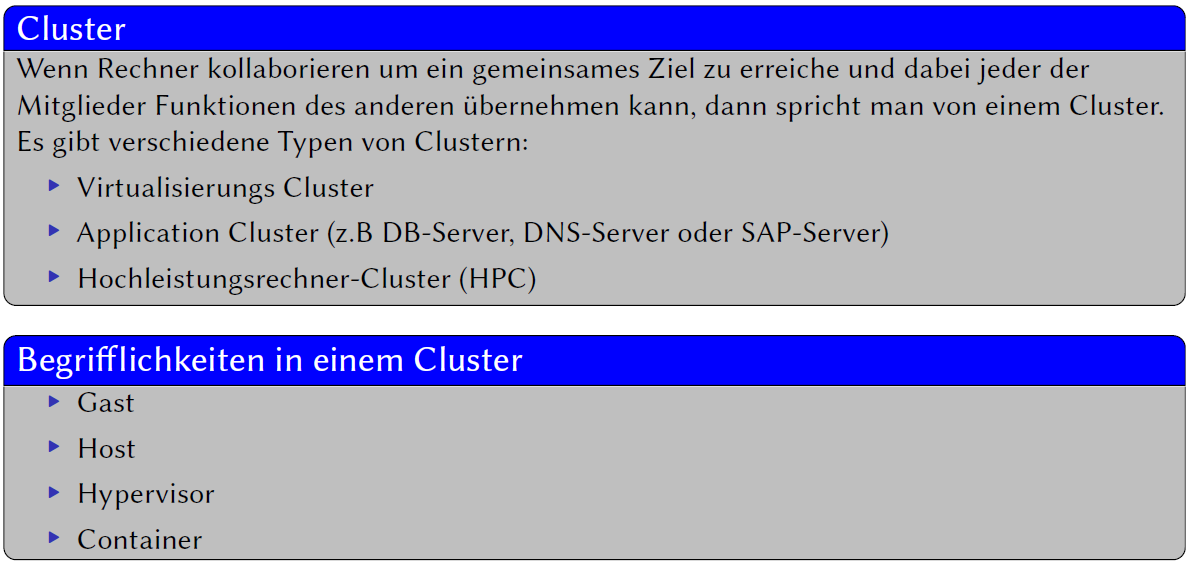
Grundsätzlich werden zwei verschiedene Typen bei den Hypervisoren unterschieden:

* Typ-1: Diese Hypervisoren werden auch “bare metal”- oder “native”-Hypervisoren genannt. Sie benötigen kein Betriebssystem um auf der Maschine zu laufen. Typische Hypervisoren dieser Klasse sind KVM,VMWare ESXi oder XEN. Hyper-V wird normalerweise auch zu diesen Hypervisoren gezählt, obwohl man darüber zweigeteilter Meinung sein kann.
* Typ-2: Diese Hypervisoren werden auch “hosted”-Hypervisoren genannt. Sie greifen auf die Funktionen eines Betriebssystems zurück um eine Virtualisierung zu realisieren. Typische Hypervisoren dieser Klasse sin Oracle Virtualbox oder VMWare Workstation.

Keine eigenen Hypervisortypen aber dennoch auch Virtualisierungslösungen sind:

* Applikationsvirtualisierungen: Virtualisierungslösungen dieser Kategorie sind VMWare ThinApp oder Citrix XenApp.
* Vollvirtualisierer/Emulatoren: Emulatoren sind in der Lage auch unabhängig von der darunterliegenden Hardware eine Maschine laufen zu lassen. Auf diesen können also auch völlig Artfremde Programme von anderen Prozessoren (z.B. Commodore 64 oder Commodore Amiga) laufen gelassen werden. Typische Vertreter dieser Klasse sind QEMU (hat auch Eigenschaften eines Typ-2 Hypervisors) und Bochs. Aufgrund der fehlenden Hardwareunterstützung sind Emulatoren zwar vielseitiger aber wesentlich ineffizienter als Hypervisoren.
* Container-Basierte Lösungen: Container basierte Lösungen verwenden direkt die Ressourcen des Hostsystems. Beispielsweise werden bei Linux-Containern der Kernel vom Gastsystem verwendet. Es kann also nicht auf einen anderen Kernel zurückgegriffen werden. Typische Lösungen sind hier Docker, LXC oder OpenVZ.

### Begrifflichkeiten in einem Cluster



Wenn Rechner kollaborieren um ein gemeinsames Ziel zu erreiche und dabei jeder der Mitglieder Funktionen des anderen übernehmen kann, dann spricht man von einem Cluster. Es gibt verschiedene Typen von Clustern:

* Virtualisierungs Cluster
* Application Cluster (z.B DB-Server, DNS-Server oder SAP-Server)
* Hochleistungsrechner-Cluster (HPC)

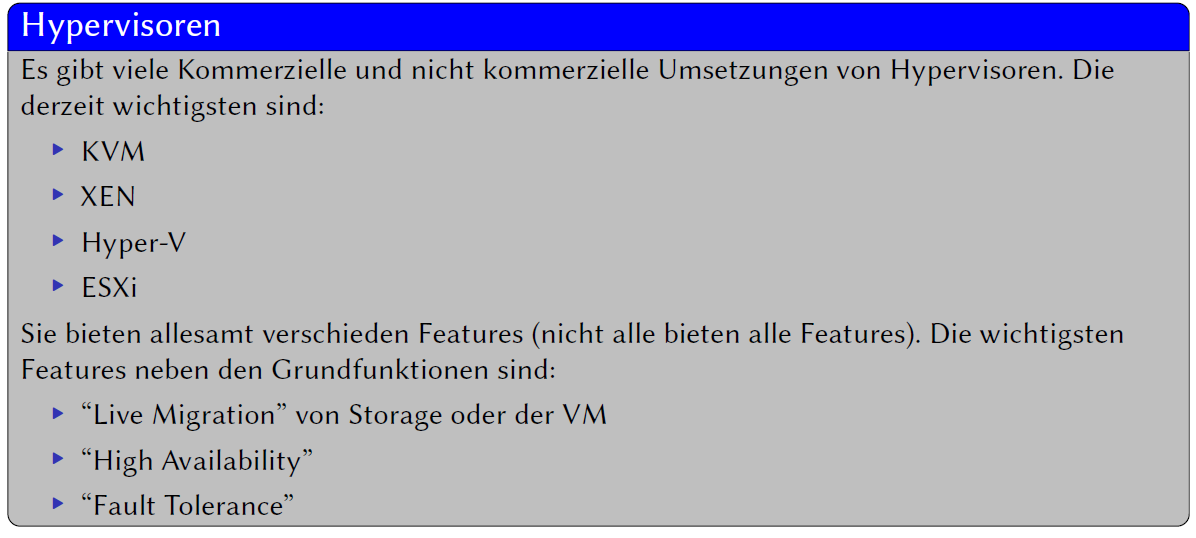
Folgende Begrifflichkeiten sind wichtig:

* Gast und Host: Ein einzelner Server in einem Cluster nennt man “Host”; Eine Virtuelle Maschine (VM) wird als “Guest” bezeichnet. Das Betriebssystem nennt man Analog “Host OS” oder “Guest OS”
* Hypervisor: Auf einem Host läuft eine Software, die als “Hypervisor” bezeichnet wird. Sie abstrahiert die Maschinenfunktionen, so dass mehrere Virtuelle Maschinen auf diese zugreifen können.
* Container: Bei Containernwerden nicht nur Hardware, sondern auch das Betriebssystem (OS) der Gäste geteilt. Container sind dadurch, dass Sie aufeinander aufbauen können extrem leichtgewichtig. Sie können im Idealfall (setzt Technologien wie Caches und Copy-on-write voraus) innert Sekunden erstellt, gestartet, abgestellt und gelöscht werden. Vernünftige Container existieren verbreitet derzeit nur auf Unix-Basierten Systemen.

Der Speicher und die CPU sind immer hostspezifisch (d.h. sie können nicht unter den Hosts “geteilt” werden).

## Implementationen von Technologien

### Hypervisoren



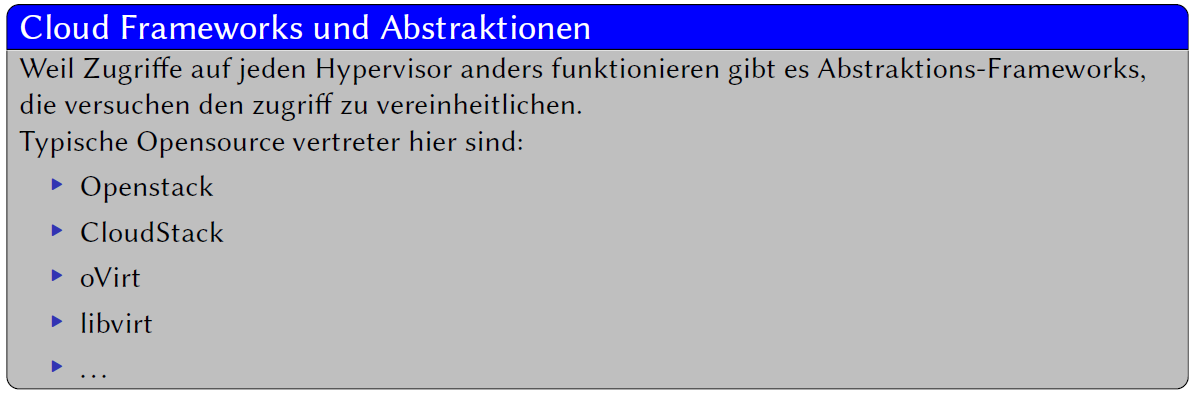
Es gibt viele Kommerzielle und nicht kommerzielle Umsetzungen von Hypervisoren. Die derzeit wichtigsten sind:

* KVM
* XEN
* Hyper-V
* ESXi

Sie bieten allesamt verschieden Features (nicht alle bieten alle Features). Die wichtigsten Features neben den Grundfunktionen sind:

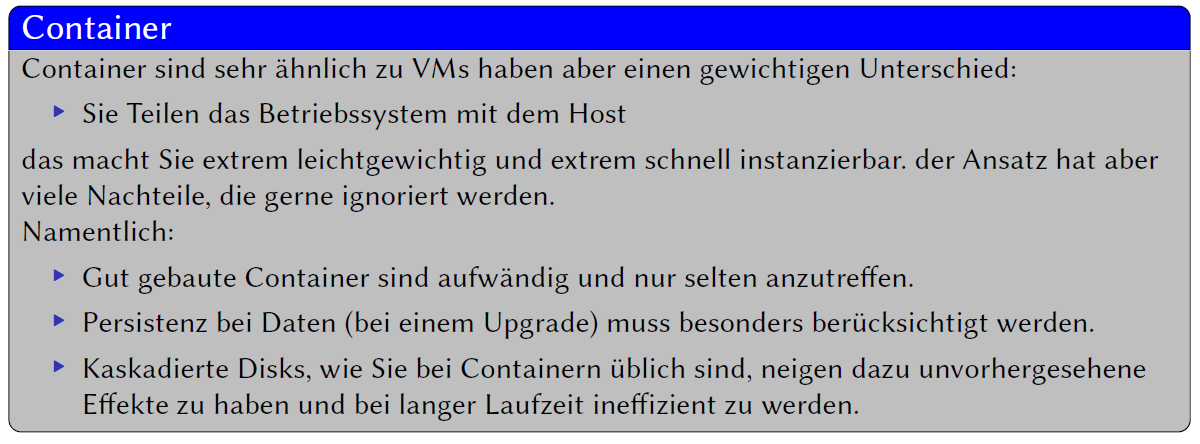
* “Live Migration” von Storage oder der VM: Das heisst, dass die VM entweder von einem Host zu einem anderen Host in laufendem Zustand geschoben werden kann, oder dass das Disk-file einer VM, während sie läuft, von einem Speicher zu einem anderen geschoben werden kann.
* “High Availability”: Das heisst, dass eine VM, die unvorhergesehen beendet wurde (z.B. weil der Host ausgefallen ist) automatisch auf einem anderen Host wieder gestartet wird.
* “Fault Tolerance”: Das heisst, dass eine VM parallel auf zwei Hosts läuft und wenn eine Ausfällt ein allfällig zugreifender Client kein Ausfall bemerkt.

### Cloud Frameworks und Abstraktionen



Viele dieser Frameworks bieten nur eine vereinheitlichte API aber keine eigene Funktionalität.

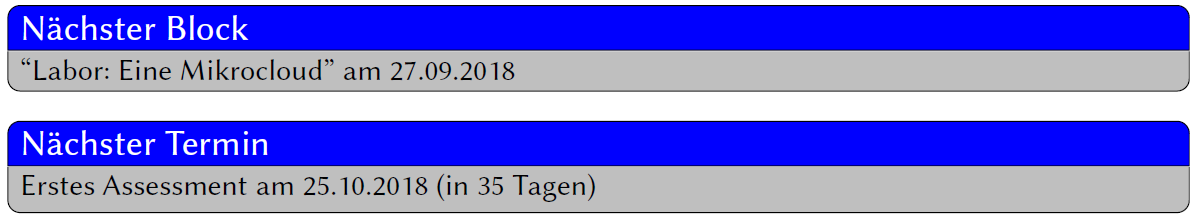
### Container



## Repetition und Aufgabe

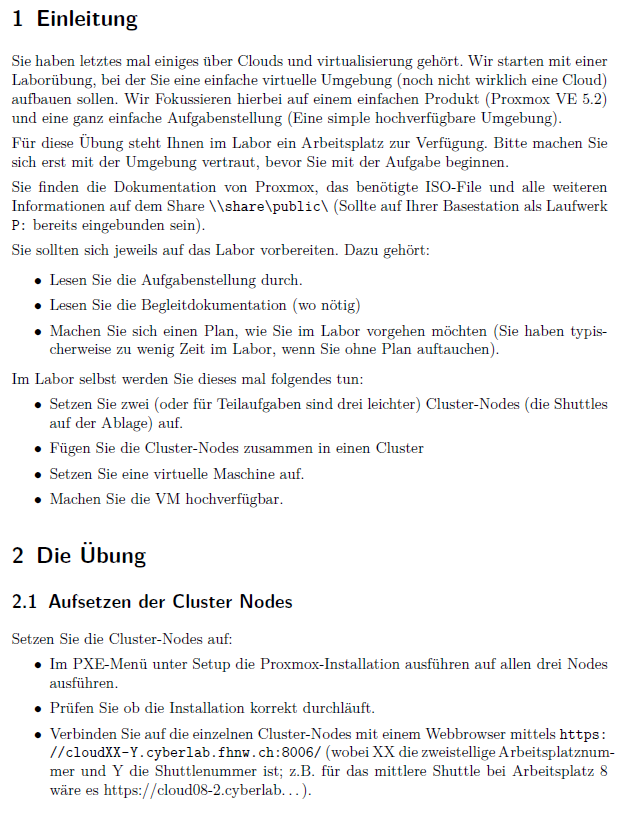


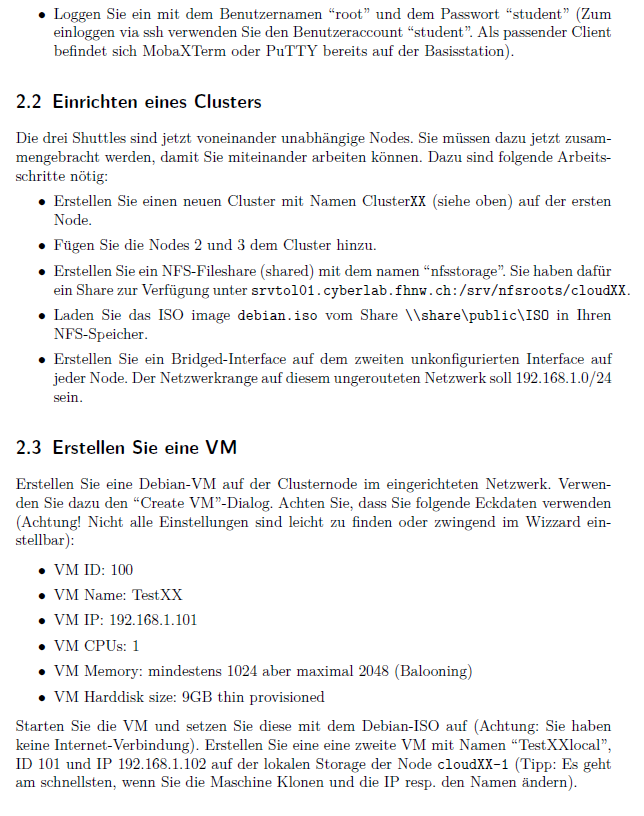
## Termine

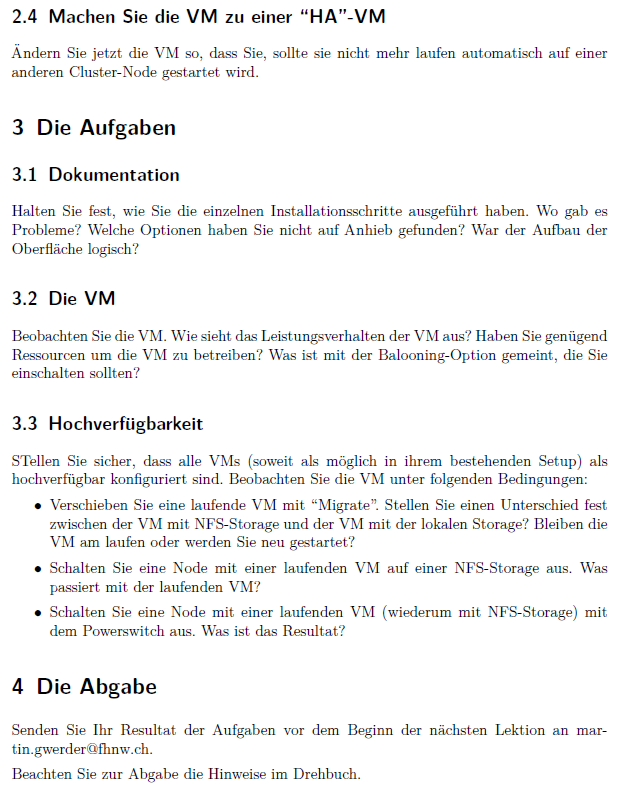


# Woche 2



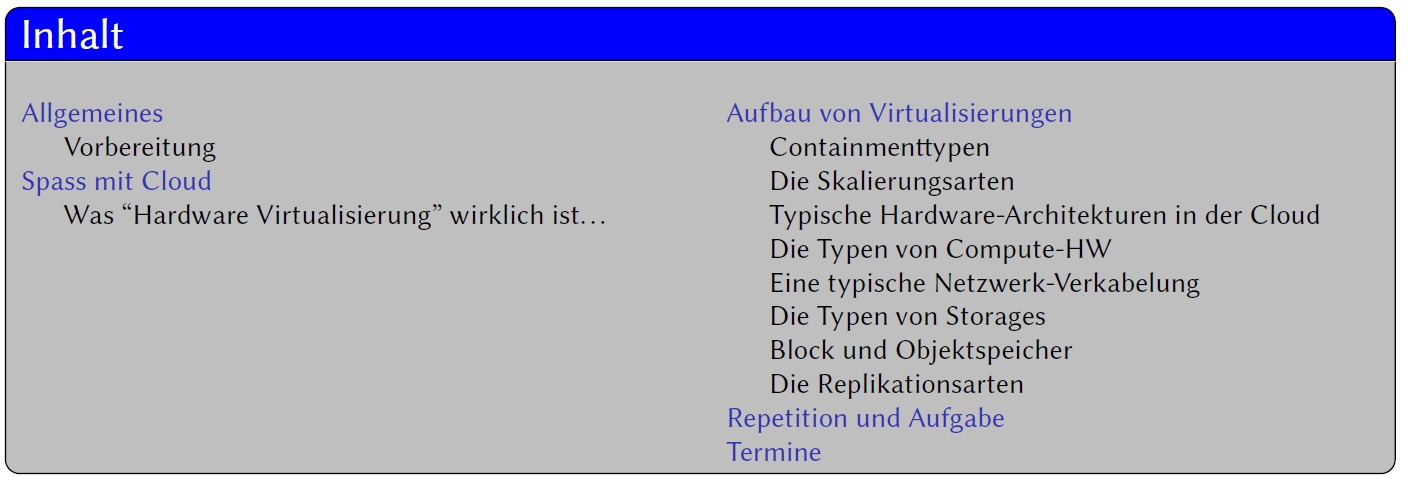






# Woche 3

## Inhalt



## Allgemeines

### Vorbereitung



## Spass mit Cloud

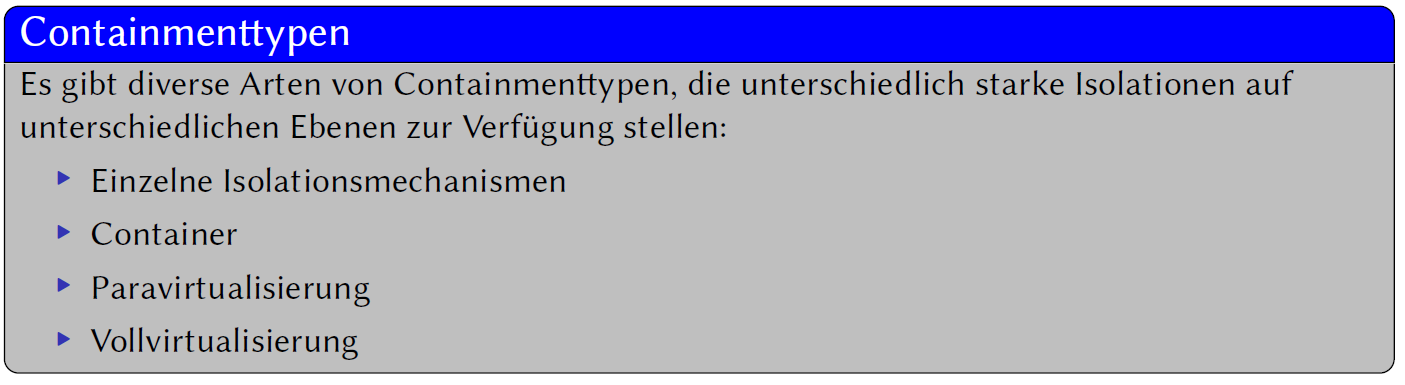
### Was “Hardware Virtualisierung” wirklich ist...



Die Cloud ist nicht ein fernes Märchenland, sondern der Computer eines anderen. Dieser verdient seinen Lebensunterhalt damit. Die Cloud kostet mich also immer (die Hardware-Kosten, den Support und den Gewinn). Zwar kann über die Masse einiges tatsächlich eingespart werden, aber viele CIOs (oder CEOs) würde die Farbe aus deren Gesicht weichen, wenn sie sich bewusst wären auf was für einfache, gestrickter Hardware ihre geschäftskritischen Prozesse dort laufen.

## Aufbau von Virtualisierungen

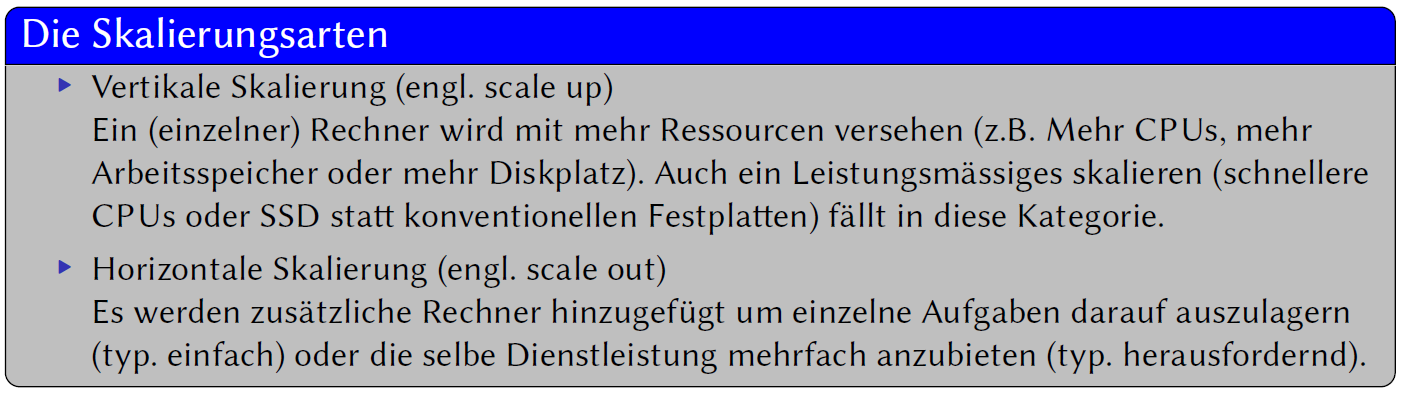
### Containmenttypen



Es gibt diverse Arten von Containmenttypen, die unterschiedlich starke Isolationen auf unterschiedlichen Ebenen zur Verfügung stellen:

* Einzelne Isolationsmechanismen: Zu dieser Kategorie gehören die Verwendung unprivilegierter Benutzer oder die Verwendung einer Sandbox in der Form einer isolierten API zum OS oder die Isolation auf dem Betriebssystem via chroot. Ebenfalls in diese Kategorie gehören die Namespaces und cgroups unter Linux.
* Container: Bei sogenannten Containern werden einzelne Isolationsmechanismen zusammengefasst. Die Features oder Wortverwendung ist nicht einheitlich geregelt. Container zeichnen sich typisch durch folgende Merkmale aus:
  + Sie teilen den Kernel mit ihrem Host.
  + Sie teilen möglicherweise das OS mit ihrem Gast – Einzelne Container können nicht aufeinander zugreifen (weder auf Prozess-, noch auf Diskoder RAM-Level).
  + Es gibt eine Vielzahl von Implementierungen für Container. Die gängigsten freien Produkte sind: Docker,OpenVZ, LXC, BSD Jails, Solaris Zones
* Paravirtualisierung: Hierbei handelt es sich um eine “wirkliche” Virtualisierung. Die Hardware wird aber (möglicherweise teilweise) über eine Vereinfachte API, statt über Direktzugriff zur Verfügung gestellt. Eine Paravirtualisierung setzt typischerweise entsprechende Treiber im Gast-Betriebssystem voraus. Bei einer Paravirtualisierung “weiss” deshalb das Gastsystem, dass es auf einer virtualisierten Umgebung läuft.
* Vollvirtualisierung: Bei der Vollvirtualisierung kümmert sich ein VM-Manager (VMM) darum, dass Hardware individuell zugeteilt (oder auch dessen Zustand gesichert) wird. Heutzutage bietet eine Vielzahl von Hardware diese Funktionen native an, um den VMM zu entlasten. Beispiele dafür sind:
  + Prozessoren: Bei AMD die AMD-V respektive bei Intel die VT-x Unterstützung.
  + Interrupcontroller: AMD AVIC repektive Intel APICv
  + GPUs: Intel GVT-g
  + Memory Management: AMD Vi oder Intel VT-d
  + Netzwerk Virtualisierung: Intel VT-c

### Die Skalierungsarten



Die typischen Skalierungsarten aus Softwaresicht sind:

* Vertikale Skalierung (engl scale up): Ein (einzelner) Rechner wird mit mehr Ressourcen versehen (z.B. Mehr CPUs, mehr Arbeitsspeicher oder mehr Diskplatz). Auch ein Leistungsmässiges skalieren (schnellere CPUs oder SSD statt konventionellen Festplatten) fällt in diese Kategorie.
* Horizontale Skalierung (engl. scale out): Es werden zusätzliche Rechner hinzugefügt um einzelne Aufgaben darauf auszulagern (typ. einfach) oder dieselbe Dienstleistung mehrfach anzubieten (typ. herausfordernd).

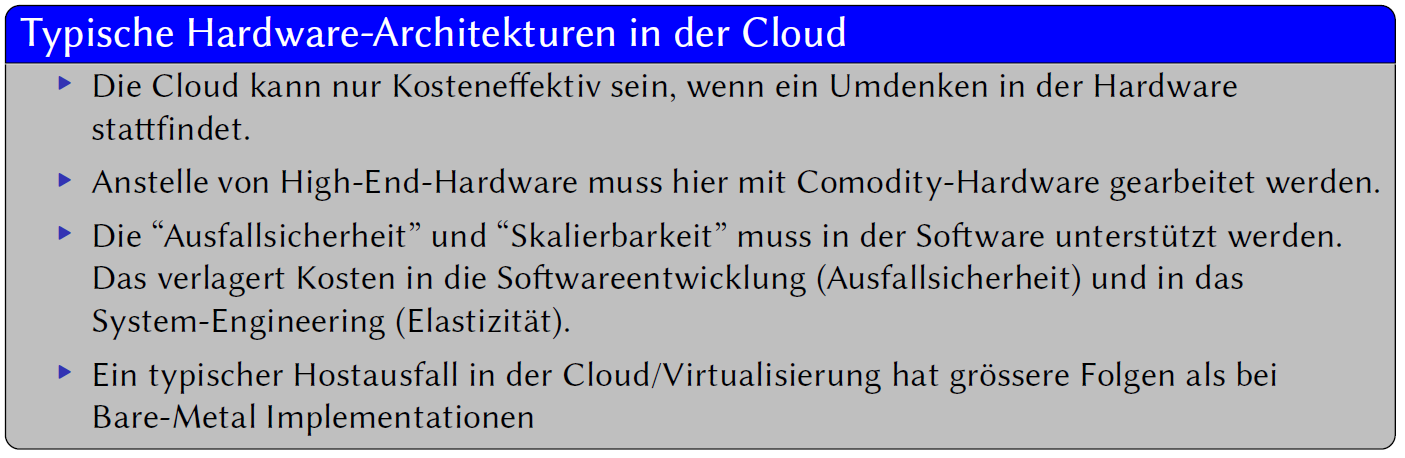
Die vertikale Skalierung ist einfacher aus Applikationssicht. Für die Virtualisierung bietet Sie aber Probleme. Eine Virtuelle Maschine sollte nicht mehr als ein Sechstel der CPU-Cores des Hostsystems umfassen, damit das Scheduling noch effizient stattfinden kann. Hingegen kann das Overcommitment bei den CPU-Cores schnell einmal auf 10 vCPUS pro pCPU ansteigen (oder gar noch höher).

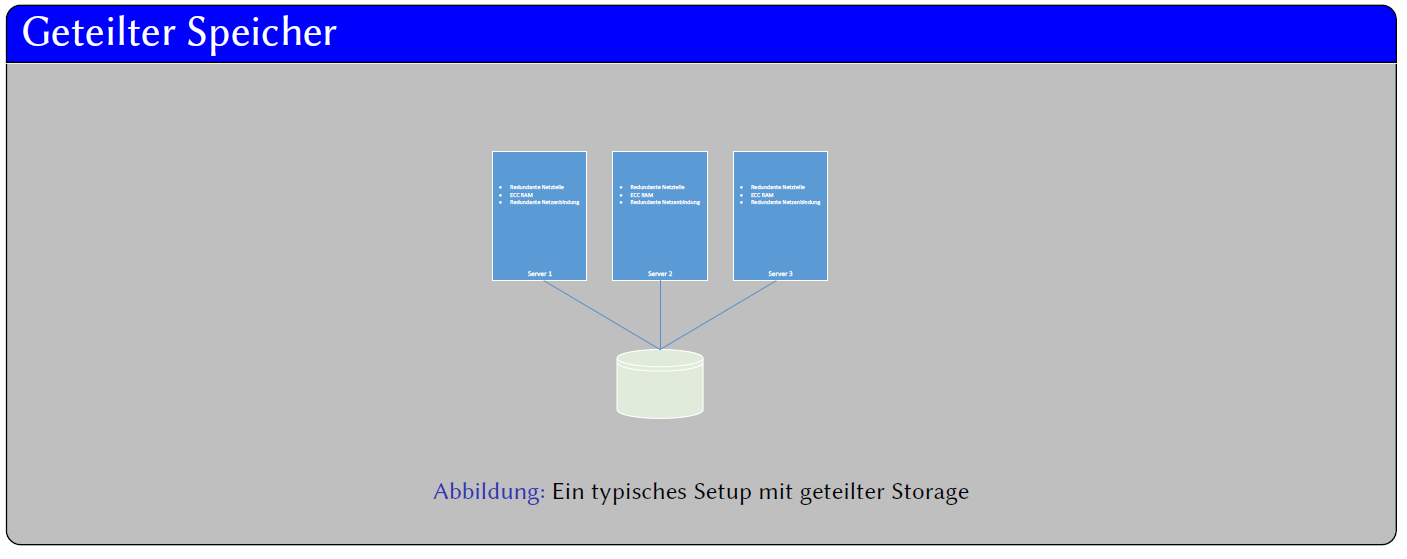
Eine vertikale Skalierung gerät normalerweise schnell an seine Grenzen. Server können nur mit einer begrenzten Anzahl CPUs (wegen der Steckplätze), RAM (auch wegen der Steckplätze) und Festplatten ausgerüstet werden (wobei es am einfachsten ist grosse Festplatten anzuschliessen).

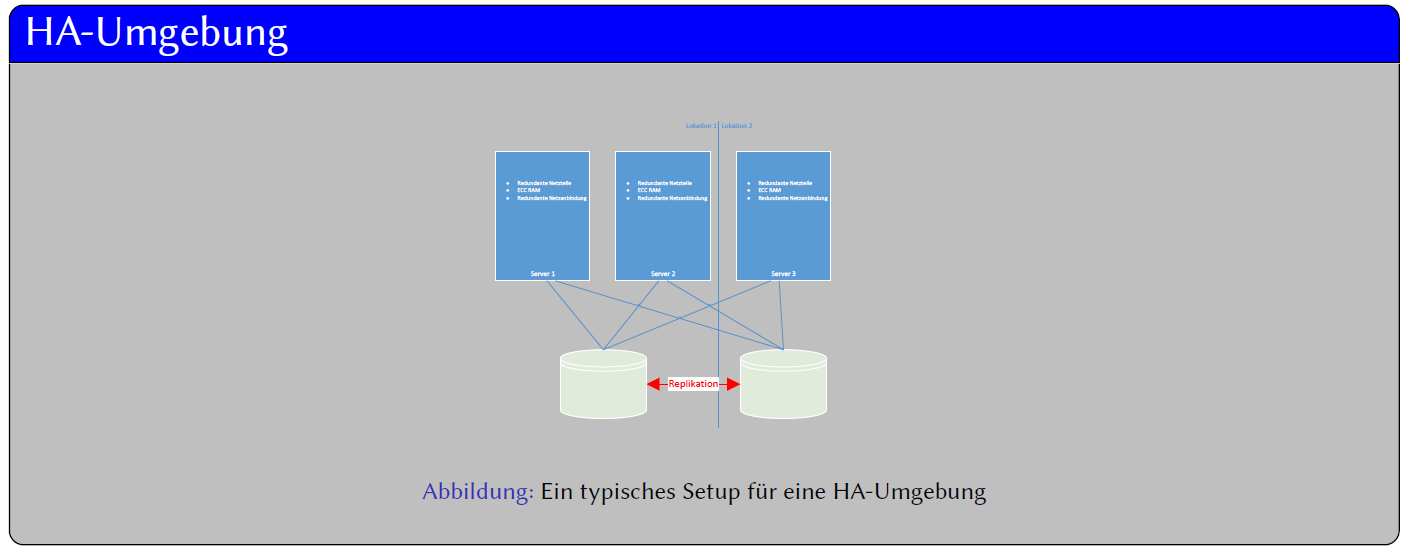
Eine grosse Disk bietet grosse Probleme, die gemeinhin nicht erwartet werden. Eine Festplatte von 1TB Grösse dauert bei einer vollständigen Übertragung über ein 1Gb Netzwerk (z.B. bei einem vollständigen Backup) rund 2.5h (Idealannahme). Stellen sie sich die Datenmengen vor, die während einesWochenendes (dann werden typischerweise die Full-Backups gemacht) gesichert werden können.

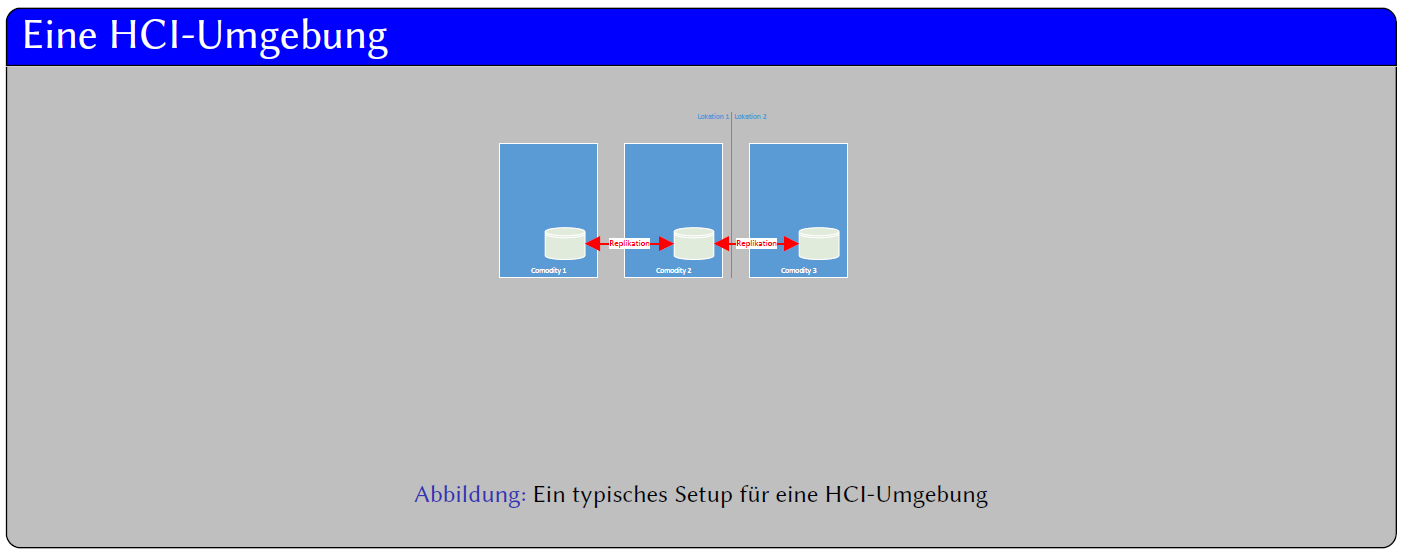
Virtuelle Maschinen sind typischerweise klein (aus Effizienzgründen).

### Typische Hardware-Architekturen in der Cloud





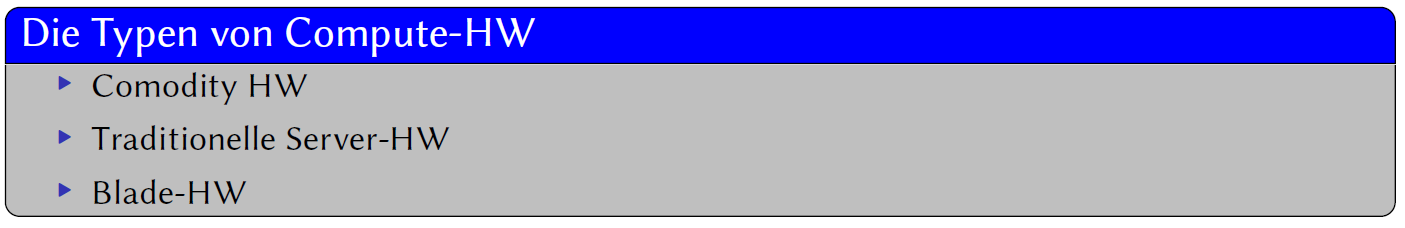




Damit eine gute Virtualisierung stattfinden kann (und nicht ein Hostausfall zum Desaster führt), muss die Umgebung über eine Storage verfügen, die von allen Hosts aus verwendet werden kann (oder zumindest von mehreren). Es gibt verschiedene Möglichkeiten dies zu erreichen. Traditionell werden hierfür externe Massenspeicher (idealerweise mit Replizierung eingesetzt, weil ja auch die Storage ausfallen kann). Der Ausfall eines Hosts ist dann vom Gesamtsystem kompensierbar.

Solche Storagesysteme sind aber typischerweise unverhältnismässig kostspielig, weswegen im Cloud-Umfeld aus Kostengründen HCI-Ansätze üblich sind. Beim HCI-Ansatz wird die lokale Storage (oder Teile davon) ganz oder teilweise auf weitere Nodes repliziert, damit diese im Falle eines Host-Ausfalls noch verfügbar sind.

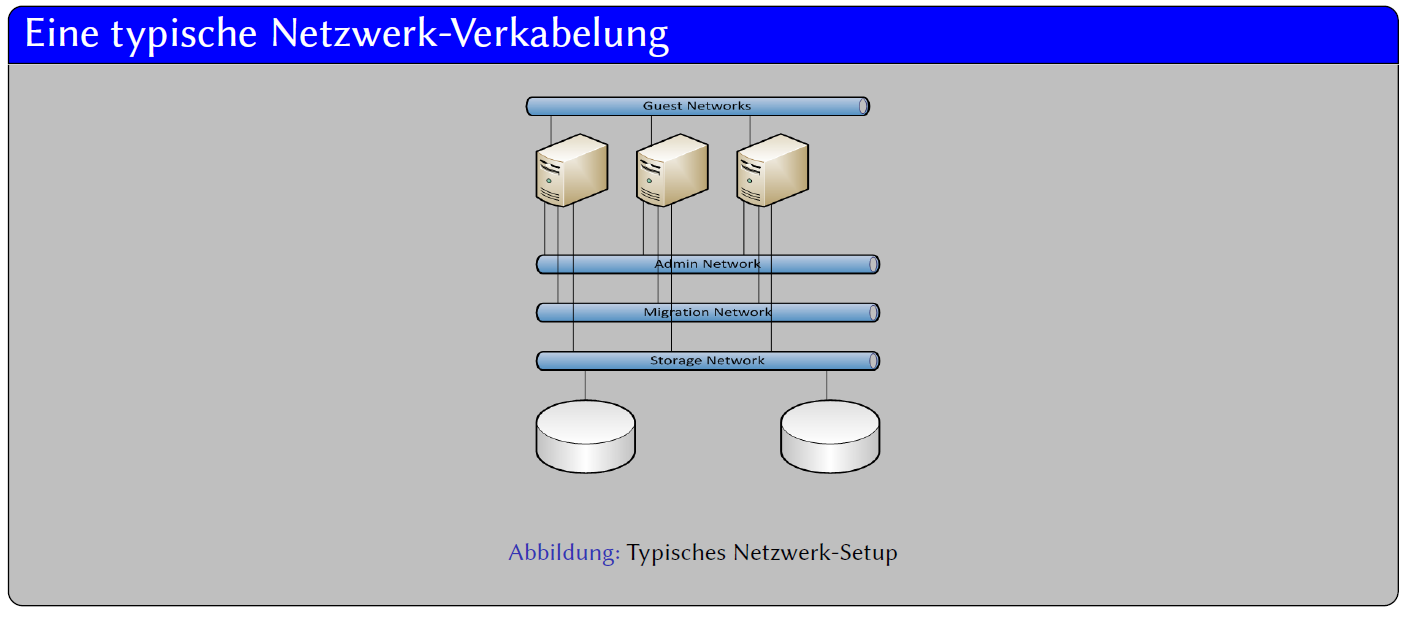
### Die Typen von Compute-HW

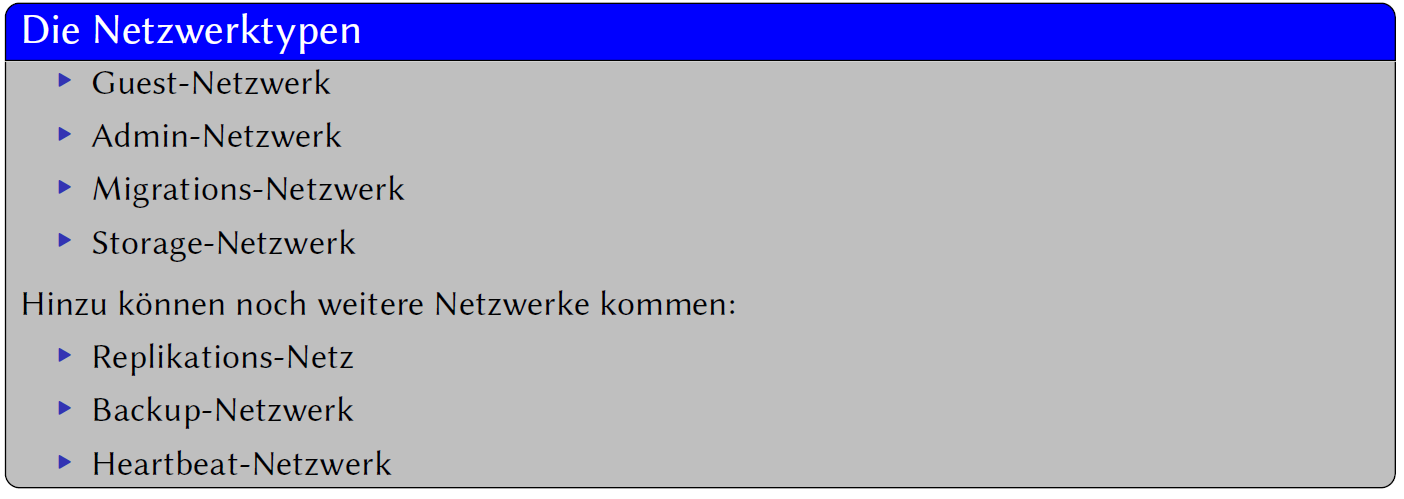


* Comodity HW: ComodityHWist günstig und bietet dieselbe Leistungsklasse zu einem viel tieferen Preis als Server HW. Die Ausfallrate dieser HWist aber aufgrund des Designs wesentlich höher als der von Server HW.
* Traditionelle Server-HW: Die traditionelle Server-HW ist darauf ausgelegt im 24c7-Betrieb zu laufen. Kleinere Ausfälle wie Bitausfälle im RAM, einzelne HD-Ausfälle oder Stromausfällewerden typischerweise durch Interne Redundanzen kompensiert.
* Blade-HW: Bei Blade-Servern wird die Basisinfrastruktur (Netzteile, Netzwerkanbindung, Lüftung u.Ä.) durch eine “Enclosure” mit typischerweise 8-16 Slots zur Verfügung gestellt (können auch wesentlich mehr sein). Die Netzwerkanbindung erfolgt generisch und kann durch die Software gesteuert werden. In die Slots werden dann die spezialisierten “Server” eingesetzt wie Schubladen (genannt “Blades”). Die Stecksysteme am hinteren Teil der Einschübe machen eine separate Verkabelung unnötig.

Blade HW ist typischerweise, wegen der hohen Investition für die Enclosure, erst kosteneeffizient ab ca. 80% Befüllungsgrad der Enclosures.

### Eine typische Netzwerk-Verkabelung



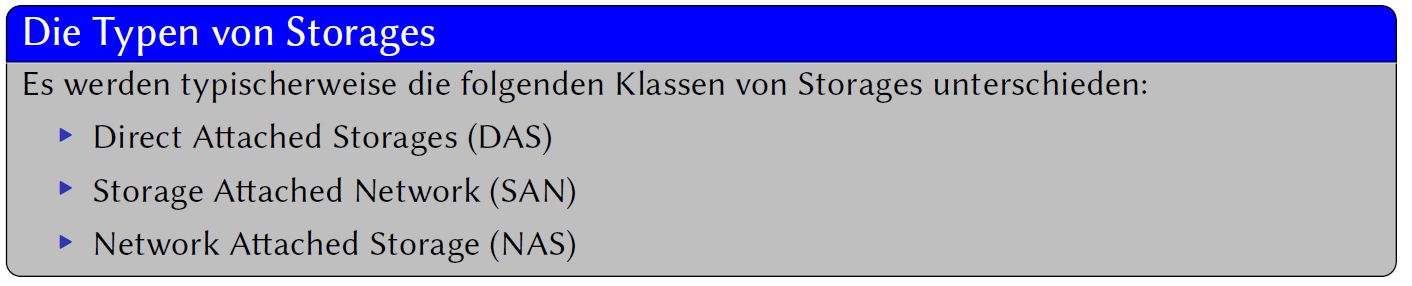


* Guest-Netzwerk: Dieses Netzwerk wird für den eigentlichen Verkehr zwischen den VMs verwendet. Es ist typischerweise ein Netzwerk-Trunk, damit mehrere Gästenetze ohne Änderung an der HW oder Verkabelung verwendet werden können.
* Admin-Netzwerk: Dieses Netzwerk wird für die Kontrolle der Hypervisoren auf den einzelnen Hosts verwendet. Es ist typischerweise von allen anderen Netzen physisch isoliert, damit Übergriffe auf dieses hochsensible Netzwerk vermindert werden können.
* Migrations-Netzwerk: Über dieses Netzwerk werden die Speicher-Abbilder oder Disk-Abbilder der VMs verschoben. Aufgrund des grossen Bandbreitenbedarfs solcher Operationen wird dieses Netzwerk ebenfalls typischerweise separat gehalten.
* Storage-Netzwerk: Das Storage-Netzwerk ist typischerweise ein Hochgeschwindigkeitsnetzwerk mit niederer Latenzzeit, um die Anbindung der Diskimages möglichst effizient zu gewährleisten.

Hinzu können noch weitere Netzwerke kommen:

* Replikations-Netz: Dieses Netzwerk wird für die Replikation der Diskinhalte verwendet. Die Bandbreitenbedürfnisse eines solchen Netzwerkes sind typischerweise sehr hoch.
* Backup-Netzwerk: Das Backup-Netzwerk wird zum Sichern der Disk-Abbilder auf ein Massenmedium verwendet. Aufgrund des hohen Durchsatzes wird auch hier gerne ein isoliertes Netzwerk verwendet. Da Backups normalerweise als Abbilder gemacht werden, ist ein Restore einzelner Files typischerweise extrem aufwändig.
* Heartbeat-Netzwerk: Über das Heartbeat-Netzwerk wird die Verfügbarkeit der einzelnen Nodes überwacht. Dieses Netzwerk wird häufig auch aus Kostengründen mit dem Admin-Netzwerk zusammengelegt.

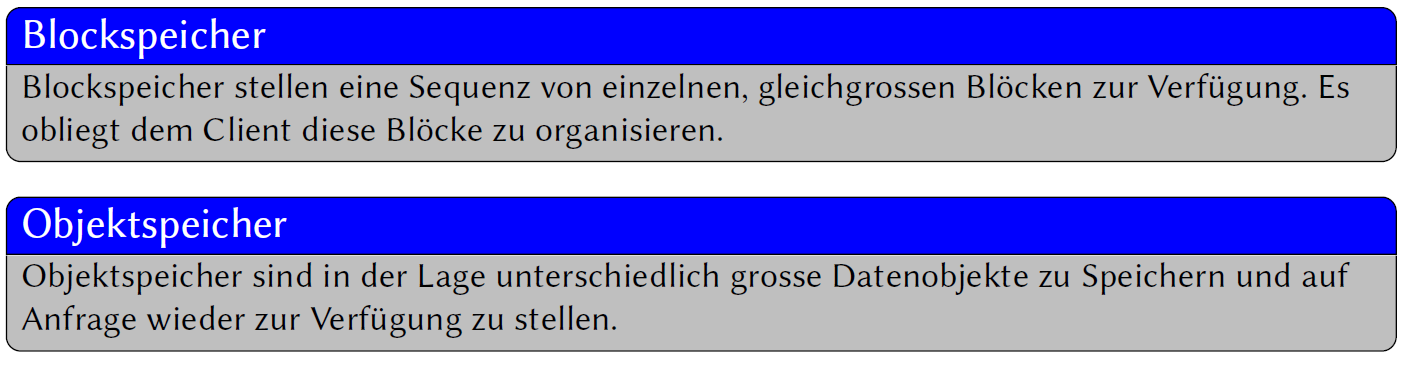
### Die Typen von Storages



Es werden typischerweise die folgenden Klassen von Storages unterschieden:

* Direct Attached Storages (DAS): Diese Laufwerke werden direkt an ein Hostsystem angeschlossen. Als Bussystem kommen traditionellerweise die typischen lokalen Bussysteme wie (e)SATA, SAS, SCSI, USB oder ähnliche zum Einsatz. Die Disksysteme unterscheiden sich von internen Laufwerken normalerweise nur durch die Tatsache, dass sie über ein externes Gehäuse verfügen und dass sie möglicherweise über einen BUS-Konverter verfügen (z.B. USB nach SATA).
* Storage Attached Network (SAN): Diese Storage funktioniert grundsätzlich gleich wie ein DAS. Der Unterschied besteht darin, dass ein SAN über ein Netzwerkprotokoll direkt als Blockspeicher angeschlossen wird. Typische Protokolle sind hier iSCSI oder Fibre Channel Protokoll. Sie sind Blockspeicher.
* Network Attached Storage (NAS): Diese Netzwerkstorages sind eigentliche Fileserver und stellen ganze Dateidienste zur Verfügung. Sie gehören deshalb auch zu den Objektstorages. Protokolle, die hier typischerweise zum Einsatz kommen sind NFS, CIFS, DAV oder FTP.

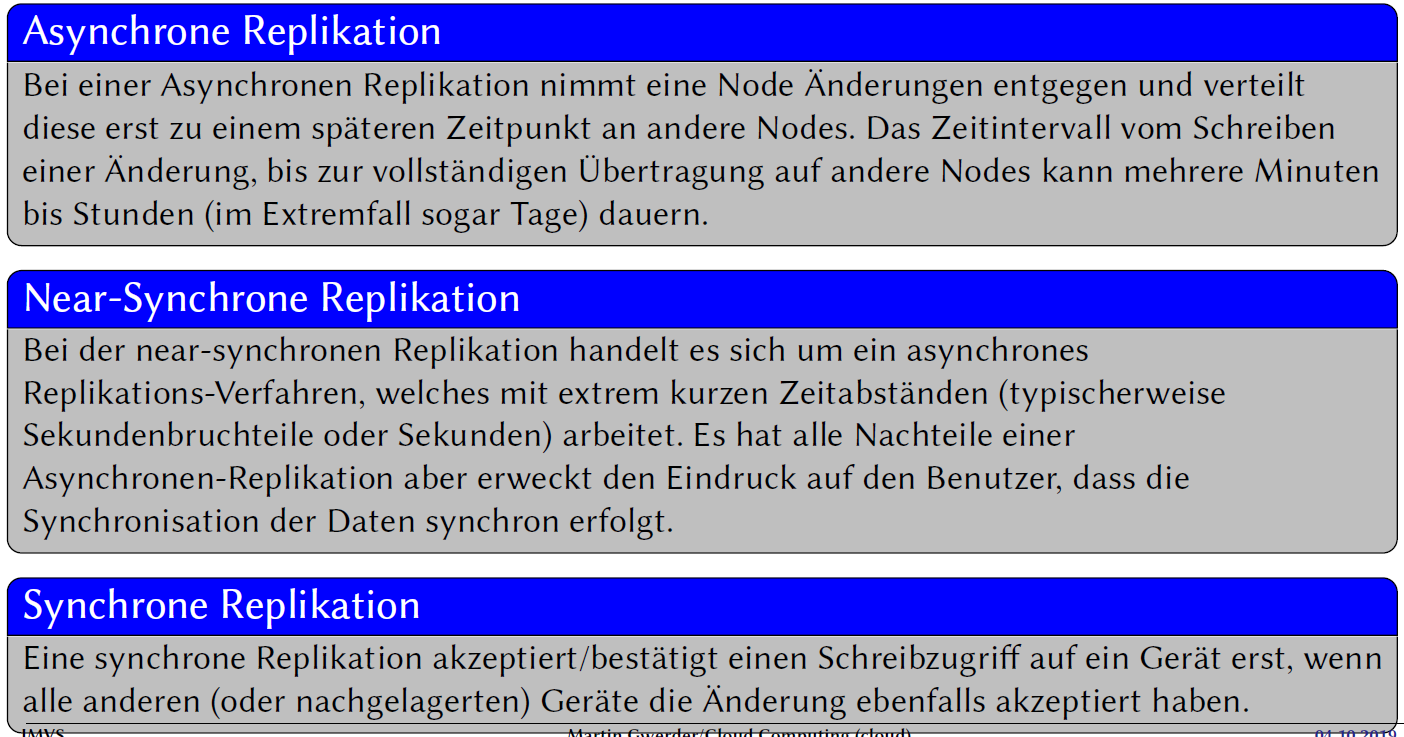
### Block und Objektspeicher



Blockspeicher stellen eine Sequenz von einzelnen, gleichgrossen Blöcken zur Verfügung. Es obliegt dem Client diese Blöcke zu organisieren. Typischerweise erfolgt dies mittels eines Dateisystems (z.B. XFS, BTRFS, FAT, NTFS usw.). Wenn mehrere Geräte/Clients gleichzeitig auf einen Blockspeicher zugreifen obliegt es Ihnen dafür zu sorgen, dass die Daten immer in einem konsistenten Zustand gelesen werden.

Objektspeicher sind in der Lage unterschiedlich grosse Datenobjekte zu Speichern und auf Anfrage wieder zur Verfügung zu stellen. Der partielle Zugriff (z.B. Zugriff ab Byte 100) kann, muss aber nicht unterstützt werden. Die Objekte werden häufig als Liste oder hierarchisch strukturiert. Ebenfalls ist typisch, dass Objekte neben den “Nutzdaten” auch Metadaten enthalten kann. Typische Beispiele von Metadaten sind Zugriffsrechte gesicherte Schlüssel zur Entschlüsselung oder Informationen wie Ersteller.

### Die Replikationsarten



Man Unterscheidet typischerweise zwei bis drei Arten von Replikation.

* Synchrone Replikation
* Asynchrone Replikation
* Near-Synchrone Replikation

Eine synchrone Replikation akzeptiert/bestätigt einen Schreibzugriff auf ein Gerät erst, wenn alle anderen (oder nachgelagerten) Geräte die Änderung ebenfalls akzeptiert haben. Dies wird üblicherweise mit 3-Phasen-Commit-Protokollen und Quoras (im Multimasterbetrieb) gelöst. Synchrone Replikation hat eine schlechte Schreibperformance, die durch Netzwerke mit hoher Latenzzeit noch verschlimmert wird. Aus diesem Grund werden synchrone Replikationen nur sparsam und meistens lokal eingesetzt.

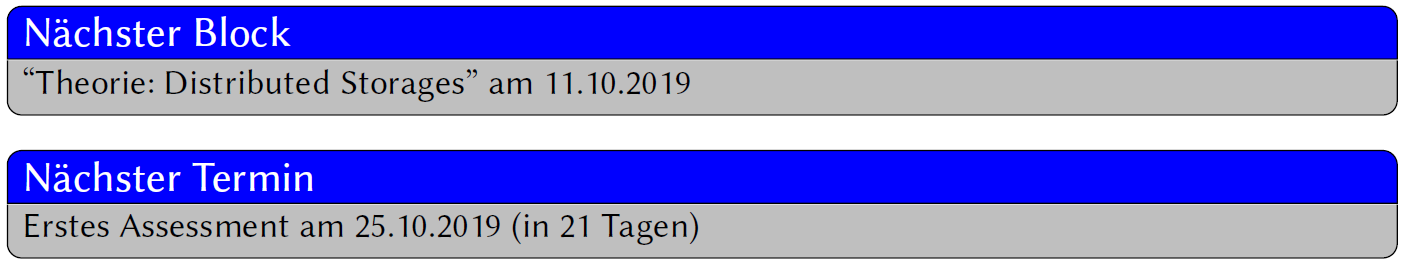
Bei einer Asynchronen Replikation nimmt eine Node Änderungen entgegen und verteilt diese erst zu einem späteren Zeitpunkt an andere Nodes. Je nach Strategie (Master/Slave oder Multimaster) kann dies zu verschiedenen Problemen führen (z.B. mehrere simultane Änderungen auf zwei Mastern). Das Zeitintervall vom Schreiben einer Änderung, bis zur vollständigen Übertragung auf andere Nodes kann mehrere Minuten bis Stunden (im Extremfall sogar Tage) dauern.

Bei der near-synchronen Replikation handelt es sich um ein asynchrones Replikations-Verfahren, welches mit extrem kurzen Zeitabständen (typischerweise Sekundenbruchteile oder Sekunden) arbeitet. Es hat alle Nachteile einer asynchronen Replikation aber erweckt den Eindruck auf den Benutzer, dass die Synchronisation der Daten synchron erfolgt.

## Repetition und Aufgabe



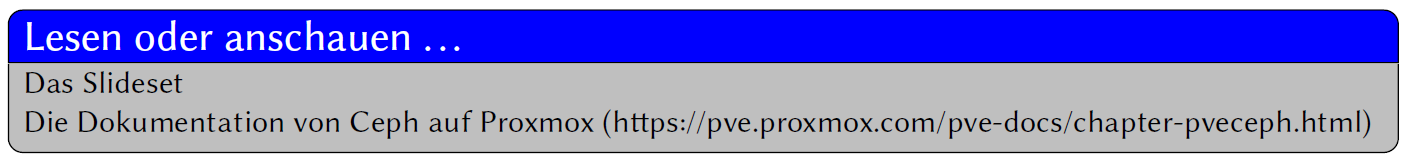
## Termine



# Woche 4

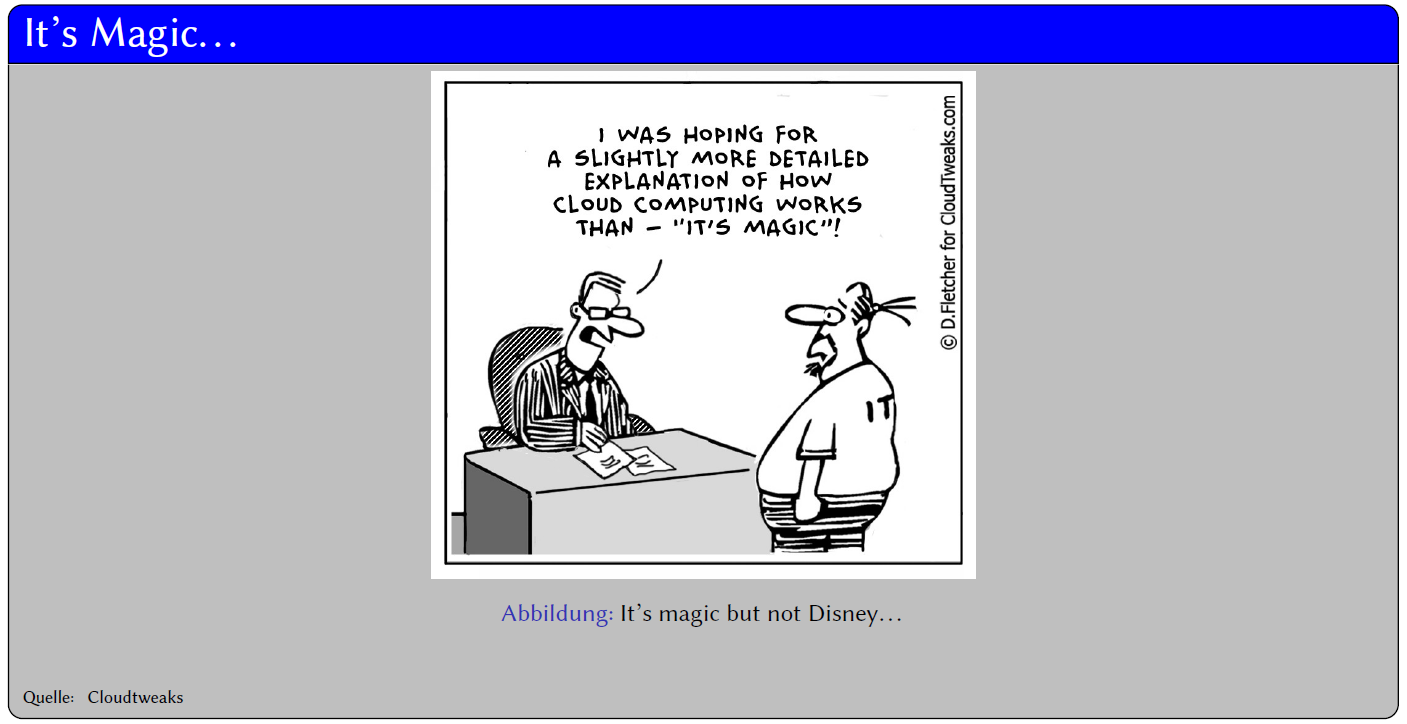
## Allgemeines

### Vorbereitung



## Spass mit Cloud

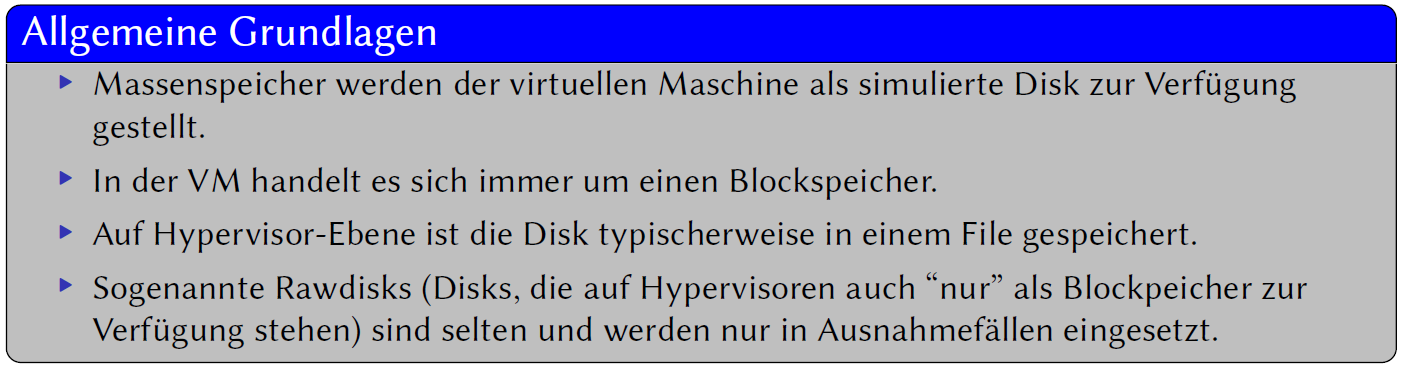
### It’s Magic…



Cloud ist eine sehr fortgeschrittene und zerbrechliche Technologie. Die wenigsten Leute verstehen Sie noch. Machen Sie es besser. Sie kriegen derzeit das Rüstzeug für diese Technologie und müssen versuchen diese zu Meistern. Das funktioniert aber nur, wenn Sie diese auch kritisch hinterfragen!

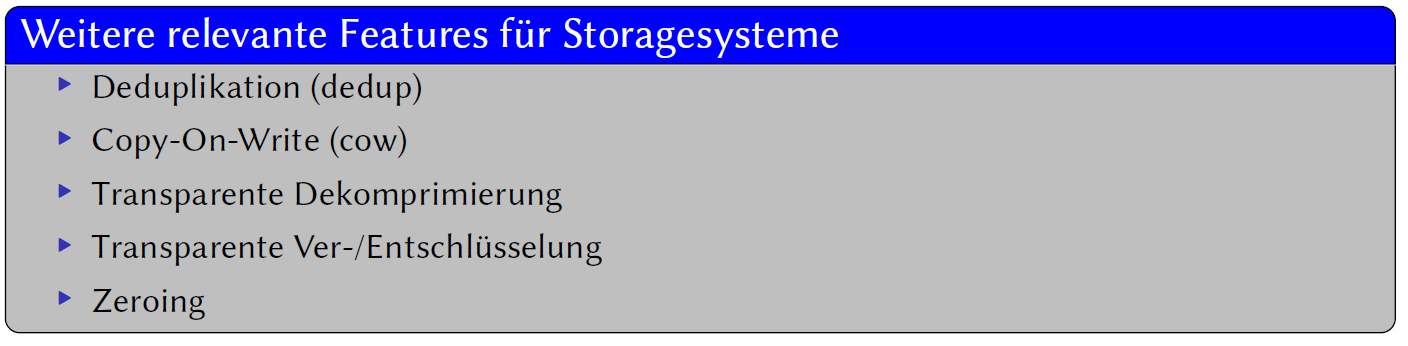
## Storage Theorie

### Allgemeine Grundlagen



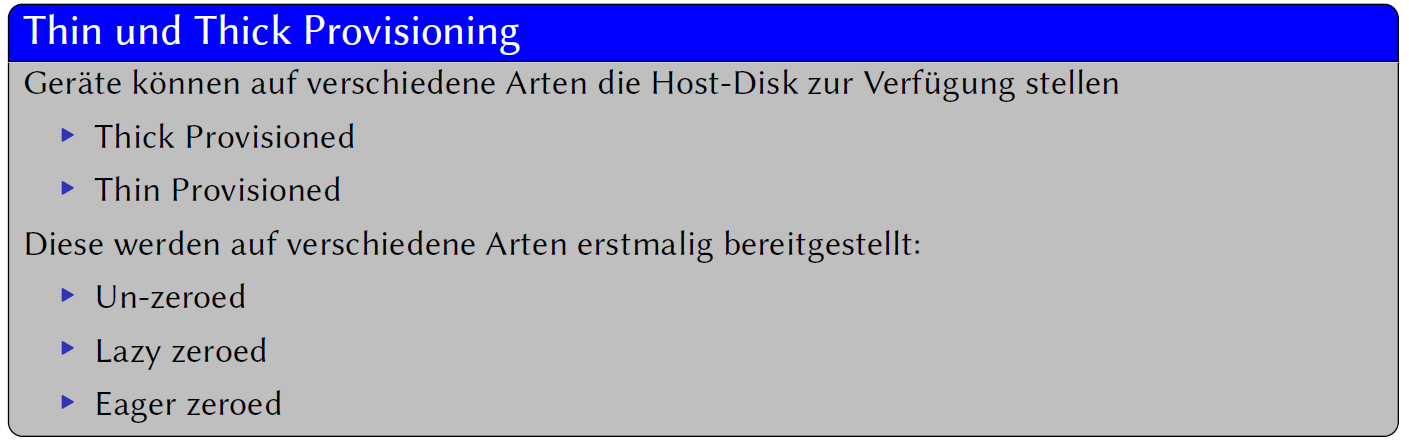
* Massenspeicher werden der virtuellen Maschine als simulierte Disk zur Verfügung gestellt. Das muss zwingend so sein, weil der Hypervisor ja keine Möglichkeit hat herauszufinden, wie ein Betriebssystem die Festplatte verwenden möchte. Alle Möglichkeiten sind demzufolge identisch mit einer physischen Festplatte.
* In der VM handelt es sich immer um einen Blockspeicher. Der Blockspeicher wird zumeist als generisches SCSI-Device zur Verfügung gestellt (heisst: Angeschlossen an einem virtuellen Kontroller können mindestens 8 (aber bis zu 32) Geräte angeschlossen werden).
* Auf Hypervisor-Ebene ist die Disk typischerweise in einem File gespeichert. Dies ist zweckmässig, weil eine Datei unabhängig von der Physis angelegt werden kann. Im Gegensatz dazu verfügen Blockspeicher nicht über ein standardisiertes Interface zum Anlegen neuer virtueller Geräte.
* Sogenannte Rawdisks (Disks, die auf Hypervisoren auch “nur” als Blockpeicher zur Verfügung stehen) sind selten und werden nur in Ausnahmefällen eingesetzt. Sie kommen Beispielsweise zum Einsatz, wenn zwei Gäste sich eine Blockdevice teilen sollen (zum Beispiel bei einem Quorums-Device; siehe später). Es ist dann allerdings Aufgabe der Gäste dafür zu sorgen, dass der wechselseitige Ausschluss beim Zugriff auf die Geräte, wo notwendig, gewährleistet ist.

### Weitere relevante Features für Storagesysteme



* Deduplikation (dedup): Als Deduplikation wird die Eigenschaft von Speichersystemen bezeichnet, gleichen Content nur einmal auf dem Hostsystem abzulegen und folgedessen Platz zu sparen. Der Begriff Deduplikation kann sowohl auf den Arbeitsspeicher als auch auf das Disksystem angewendet werden, bedeutet aber in beiden Fällen das Gleiche.
* Copy-On-Write (cow): Copy-on-write kann als eine reduzierte Form von dedup gesehen werden. Bei cow wird ein File, wenn es kopiert wird, nicht physisch kopiert, sondern als Deltafile zum Ursprungsfile angelegt.
* Transparente Dekomprimierung: Bei der transparenten Dekomprimierung werden Daten vom Host-Systemm gelesen und dekomprimiert, bevor sie an das Gastsystem weitergegeben werden. Dies ist speziell attraktiv, wenn auf gut komprimierbare aber grosse, feste Datenbestandteile bei einer Appliance zurückgegriffen werden.
* Transparente Verschlüsselung: Bei der Transparenten Verschlüsselung erfolgt eine Verschlüsselung des Diskfiles bereits auf Hostbasis. Normalerweise ist der Sicherheitsgewinn einer solchen Verschlüsselung berechtigterweise umstritten, es sei denn es handelt sich beim Storagesystem um ein nicht-vertrauenswürdiges System oder nicht-vertrauenswürdige Systeme haben ebenfalls Zugriff auf die verwendete Storage.
* Zeroing: Siehe Nachfolgende Folien...

### Thin und Thick Provisioning



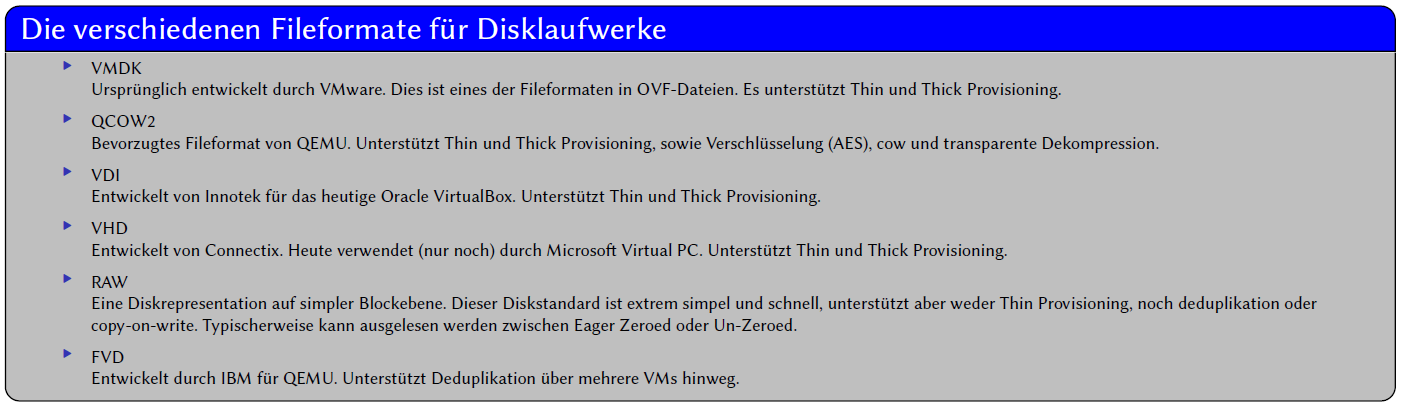
Geräte können auf verschiedene Arten die Host-Disk zur Verfügung stellen

* Thick Provisioned: Bei diesem Provisionierungs-Modell werden alle Blocks beim Anlegen der Disk automatisch alloziert. Das heisst, dass eine solche Disk von Anfang an dieselbe Grösse hat, wie im Endzustand (alles belegt durch das Dateisystem). Ein unbestreitbarer Vorteil von Thick Provisioned Disks ist, dass wenn das Basisvolume auf dem die Disks angelegt werden voll ist die Dsik weiterhin lauffähig bleibt (natürlich ausgenommen, wenn aktive Snapshots auf der Festplatte sind). Eine RAW-Disk ist der Trivialfall einer Thick provisioned Disk.
* Thin Provisioned: Bei diesem Provisionierungsmodell werden Blöcke (oder Cluster von Blöcken) angelegt, sobald diese das erste Mal durch das Gastsystem verwendet werden. Das führt dazu, dass Disks anfangs nur klein sind. Über die Zeit hinweg entwickeln sie sich allerdings zur vollen Grösse. Virtuelle Appliances (vorkonfigurierte VM-Templates) können durch diese Technologieweiter reduziertwerden für den Transport durch das Netzwerk.

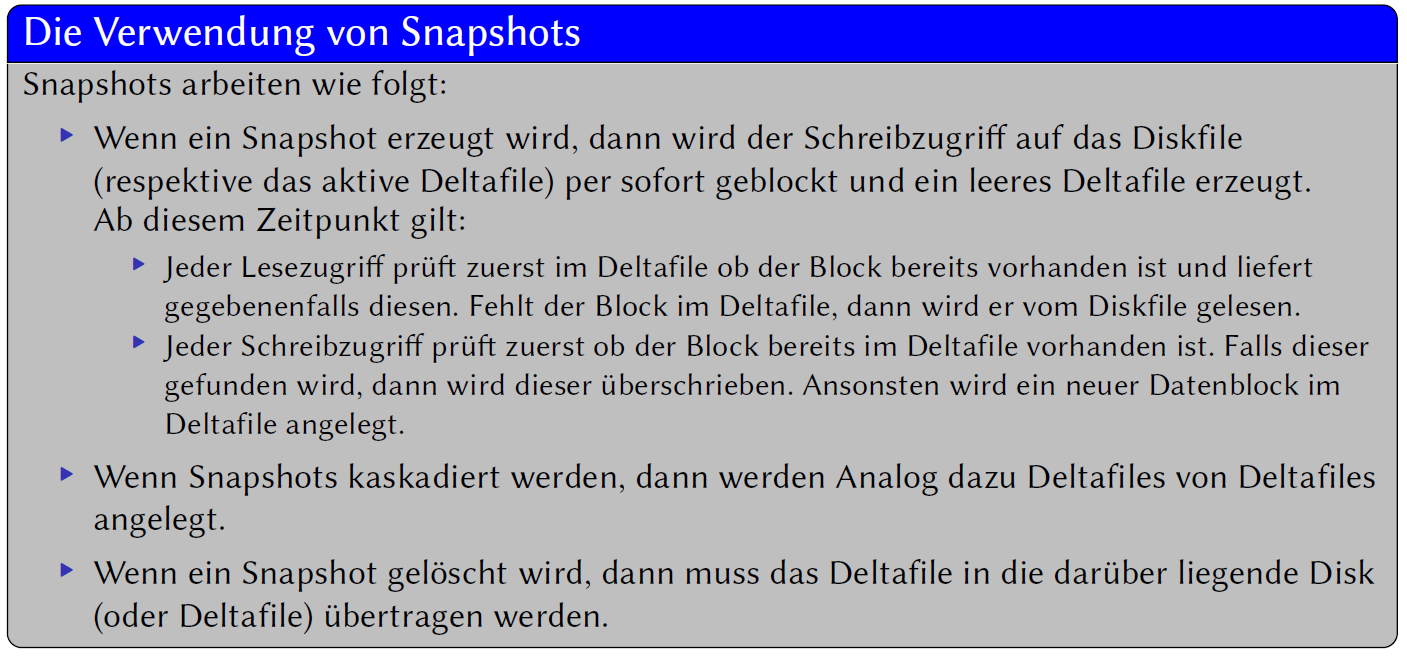
Die Blocks einer Host-Disk werden erstmalig folgendermassen bereitgestellt:

* Un-zeroed: Eine un-zeroed Disk alloziert den Diskplatz für die Disk auf dem Hostsystem und gibt diese eins zu eins an das Gastsystem weiter. Das heisst, dass in den Blöcken, auf die erstmalig zugegriffen wird Fragmente von ehemaligen Dateien auf dem Hostsystem enthalten sein können. Dies stellt
* ein Sicherheitsrisiko dar und un-zeroed Disks sollten folgedessen nicht eingesetzt werden.
* Lazy zeroed: Bei lazy zeroed Disks wird ein Block, wenn er noch nicht beschrieben wurde automatisch auf null gesetzt, wenn er gelesen wird. Auf diese Weise wird verhindert, dass ehemalige Dateiinhalte des Hostsystems zum Gast “geleaked” werden. Bei Lazy zeroing wird entweder alle Blöcke, die alloziert werden automatisch mit Nullen überschrieben (bei Thin Provisioned Disks) oder aber es muss vom Diskstandard Buch geführt werden, welche Inhalte bereits mit Nullen überschrieben wurden.
* Eager zeroed: Bei Eager Zeroed Disks werden alle Blöcke vor der Erstverwendung der Disk mit Nullen physisch überschrieben. Disks dieses Typs sind theoretisch am schnellsten. Viel mehr als das Zeroing fällt allerding die Zugriffszeit auf das Disksubsystem ins Gewicht.

### Die verschiedenen Fileformate für Disklaufwerke



### Die Verwendung von Snapshots

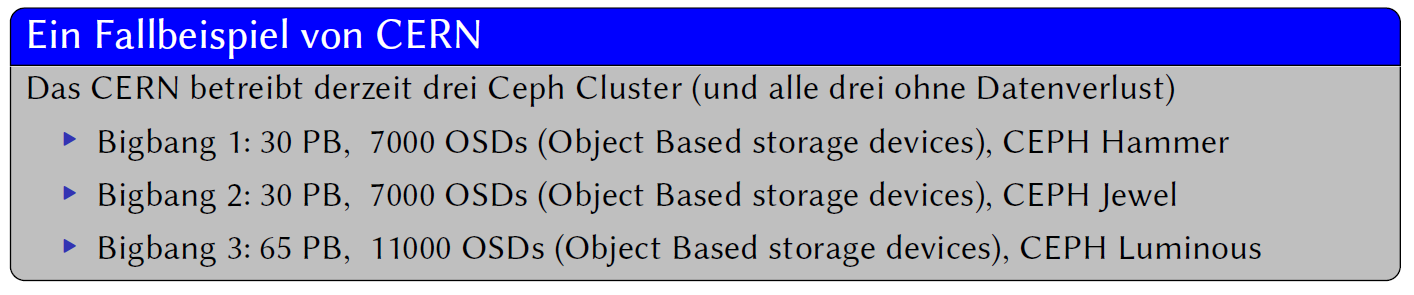


Snapshots arbeiten wie folgt:

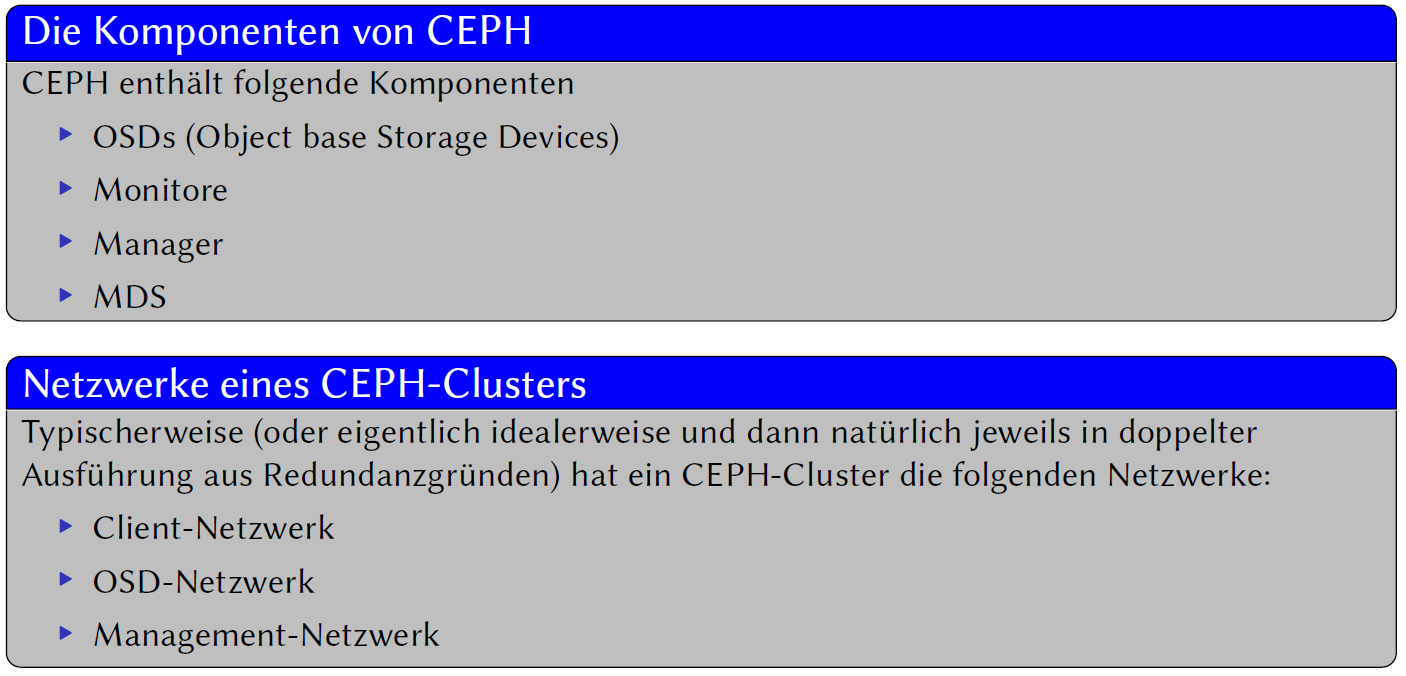
* Wenn ein Snapshot erzeugt wird, dann wird der Schreibzugriff auf das Diskfile (respektive dasaktive Deltafile) per sofort geblockt und ein leeres Deltafile erzeugt.Ab diesem Zeitpunkt gilt:
  + Jeder Lesezugriff prüft zuerst im Deltafile ob der Block bereits vorhanden ist und liefertgegebenenfalls diesen. Fehlt der Block im Deltafile, dann wird er vom Diskfile gelesen.
  + Jeder Schreibzugriff prüft zuerst ob der Block bereits im Deltafile vorhanden ist. Falls dieser gefunden wird, dann wird dieser überschrieben. Ansonsten wird ein neuer Datenblock im Deltafile angelegt.
* Wenn Snapshots kaskadiert werden, dann werden analog dazu Deltafiles von Deltafiles angelegt. Die Lesezugriffe kaskadieren sich natürlich dann auch. Schreibzugriffe bleiben vergleichbar effizient.
* Wenn ein Snapshot gelöscht wird, dann muss das Deltafile in die darüber liegende Disk (oder Deltafile) übertragen werden. Daraus folgt die Tatsache, dass ein Snapshot anlegen sehr schnell erfolgt. Ein Snapshot löschen ist allerdings Zeitintensiv, weil die Summe der Transaktionen im darunterliegenden System nachgeführt werden muss.

## Das CEPH-Projekt

### Ein Fallbeispiel von CERN



### Die Komponenten von CEPH



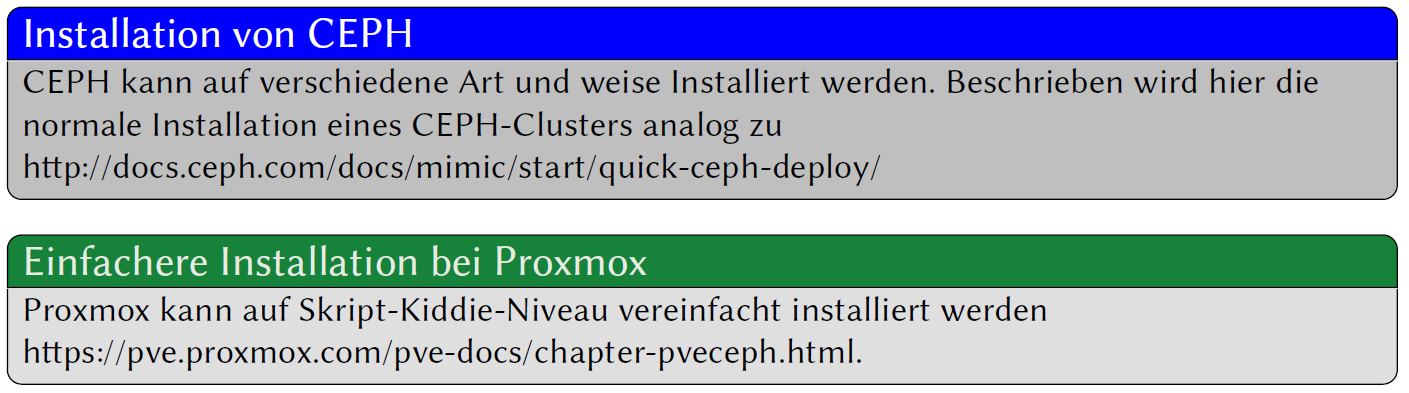
CEPH enthält folgende Komponenten

* OSDs (Object base Storage Devices): Sie bilden die “Speichermedien des Clusters”.
* Monitore: Monitore überwachen bei ceph den Status des Clusters und erstellen eine Karte über aktive (oder standby) Komponenten des Clusters. Sie sind des Weiteren zuständig für Authentifizierung von Clients. Es sollte in einem Setup mindestens drei Monitore (oder eine höhere, idealerweise ungerade Anzahl) geben, damit diese jederzeit ein sauberes Quorum bilden und “Split-Brain” Situationen effektiv handeln können.
* Manager: Die Manager stellen eine API für die Clusterdaten zur Verfügung, damit externe Dienste diesen effizient abfragen und überwachen können.
* MDS: Der Metadatenserver ist eigentlich eine Cache-Struktur, die es erlaubt effizient Metadaten zu enumerieren, ohne dass aufwändige Anfragen an den Cluster gestellt werden müssen.

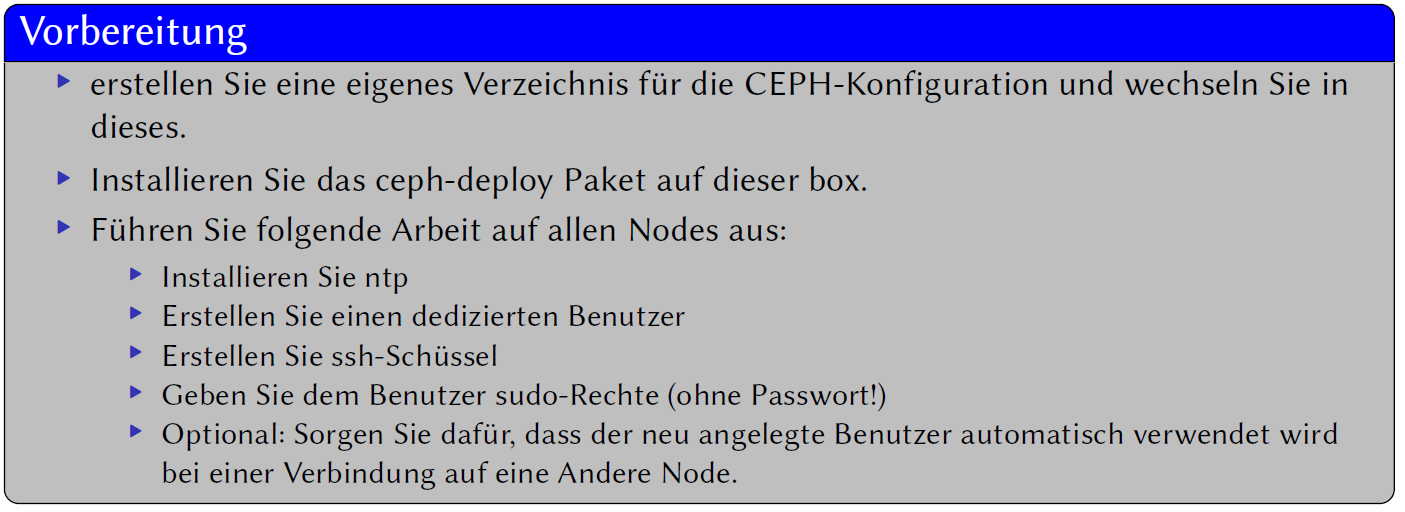
Typischerweise (oder eigentlich idealerweise und dann natürlich jeweils in doppelter Ausführung aus Redundanzgründen) hat ein CEPH-Cluster die folgenden Netzwerke:

* Client-Netzwerk: Zum Abfragen der Clusterinhalte.
* OSD-Netzwerk: Für die Replikation zwischen den OSDs.
* Management-Netzwerk: Für den Verkehr von Monitoren, MDS und Managern.

### Installation von CEPH

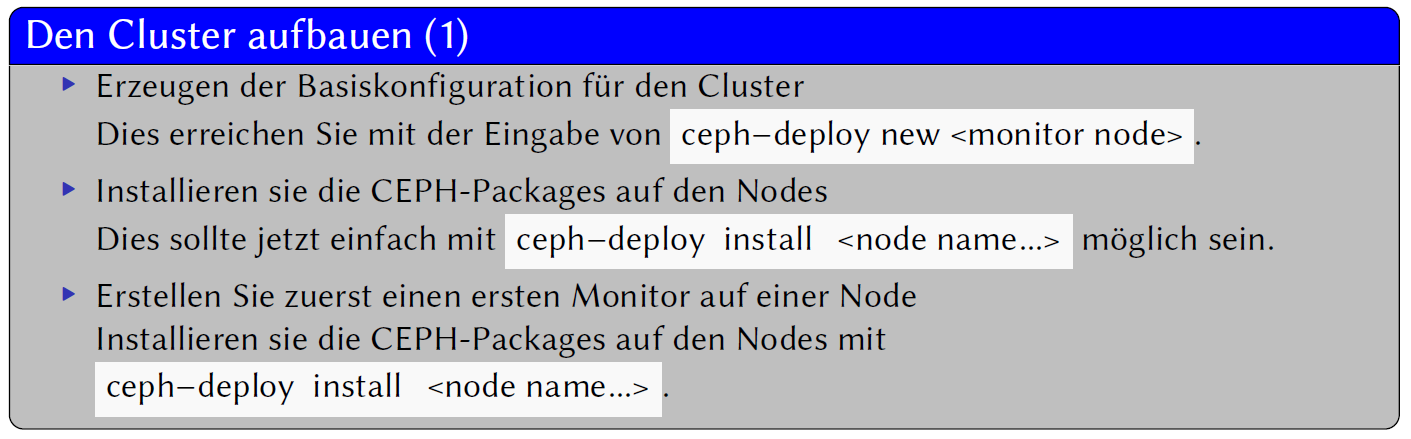


### Installation: Die Vorbereitung

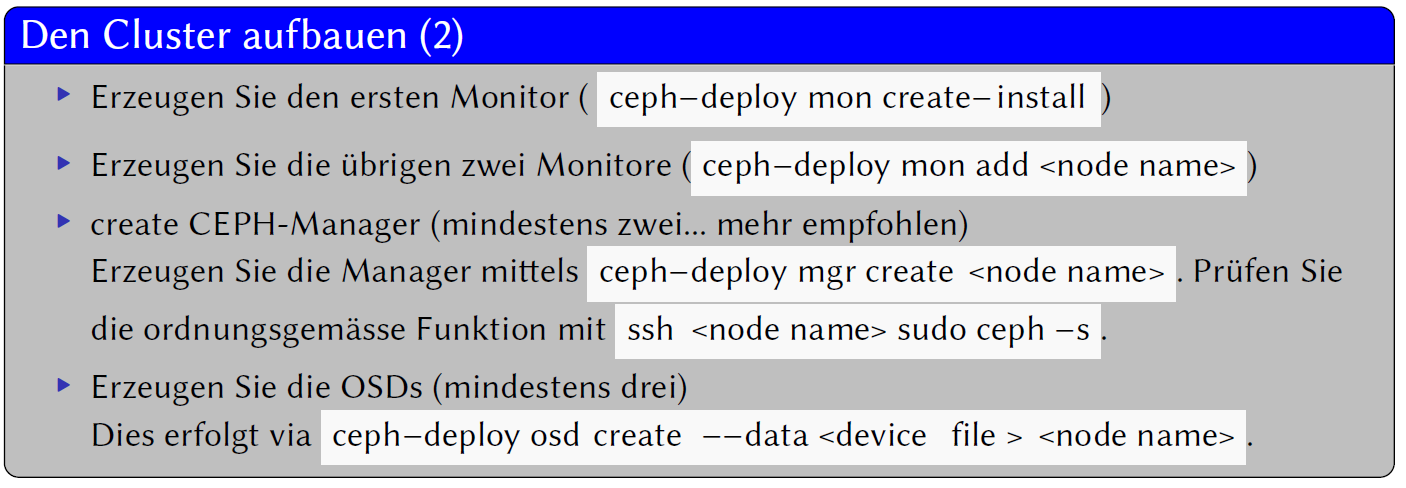


* Erstellen Sie ein eigenes Verzeichnis für die CEPH-Konfiguration und wechseln Sie in dieses.
* Installieren Sie das ceph-deploy Paket auf dieser box. Sie können dies mit apt get ceph−deploy erledigen.
* Führen Sie folgende Arbeit auf allen Nodes aus:
  + Installieren Sie ntp ( apt install ntp ): Achten Sie im Lab darauf, dass der einzige erreichbare Server ntp1.cyberlab.fhnw.ch ist. Prüfen Sie, ob die Zeit wirklich auch synchron auf den Nodes läuft.
  + Erstellen Sie einen dedizierten Benutzer: Sorgen Sie dafür, dass der Benutzer ein ordentliches Homeverzeichnis hat und ein Passwort gesetzt.
  + Erstellen Sie ssh-Schüssel: Erstellen Sie zuerst in ihrem Arbeitsverzeichnis mit ssh-keygen einen Schlüssel und verteilen sie diesen anschliessend auf alle Nodes, so dass Sie passwortlos einloggen können.
  + Geben Sie dem Benutzer sudo-Rechte (ohne Passwort!)
  + Optional: Sorgen Sie dafür, dass der neu angelegte Benutzer automatisch verwendet wird bei einer Verbindung auf eine Andere Node. Dies erreichen Sie über Hosteinträge im via /<userhome>/.ssh/config.

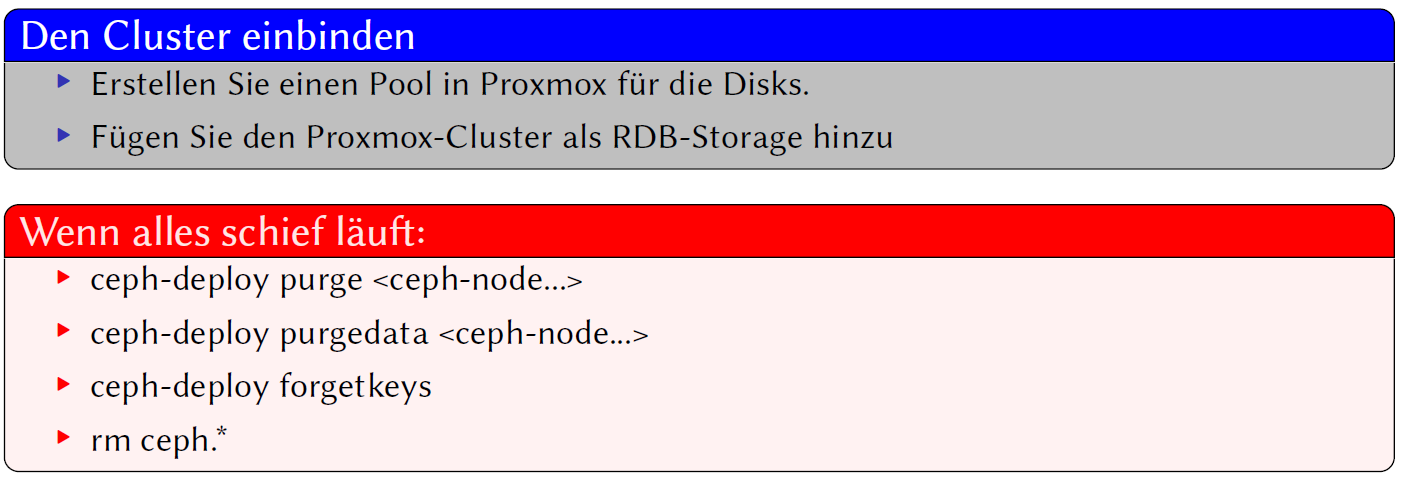
### Installation: Cluster Aufbauen



* Erzeugen der Basiskonfiguration für den Cluster: Dies erreichen Sie mit der Eingabe von ceph−deploy new <monitor node>. Im neu erzeugten ceph.conf sollten Sie noch unter “public network” die CIDR-Adresse des Hostnetzwerkes angeben über das kommuniziert wird.
* Installieren sie die CEPH-Packages auf den Nodes: Dies sollte jetzt einfach mit ceph−deploy install <node name...> möglich sein.
* Erstellen Sie zuerst einen ersten Monitor auf einer Node: Installieren sie die CEPH-Packages auf den Nodes mit ceph−deploy install <node name...>.



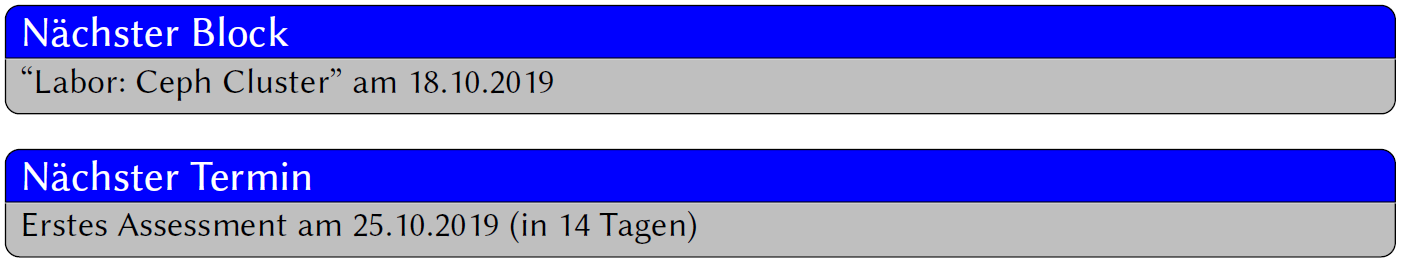
* Erzeugen Sie den ersten Monitor (ceph−deploy mon create−install)
* Erzeugen Sie die übrigen zwei monitore (ceph−deploy mon add <node name>)
* Erzeugen Sie einen CEPH-Manager (mindestens zwei... mehr empfohlen): Erzeugen Sie die Manager mittels ceph−deploy mgr create <node name>. Prüfen Sie die ordnungsgemässe Funktion mit ssh <node name> sudo ceph −s.
* Erzeugen Sie die OSDs (mindestens drei): Dies erfolgt via ceph−deploy osd create −−data <device file > <node name>.



## Repetition und Aufgabe

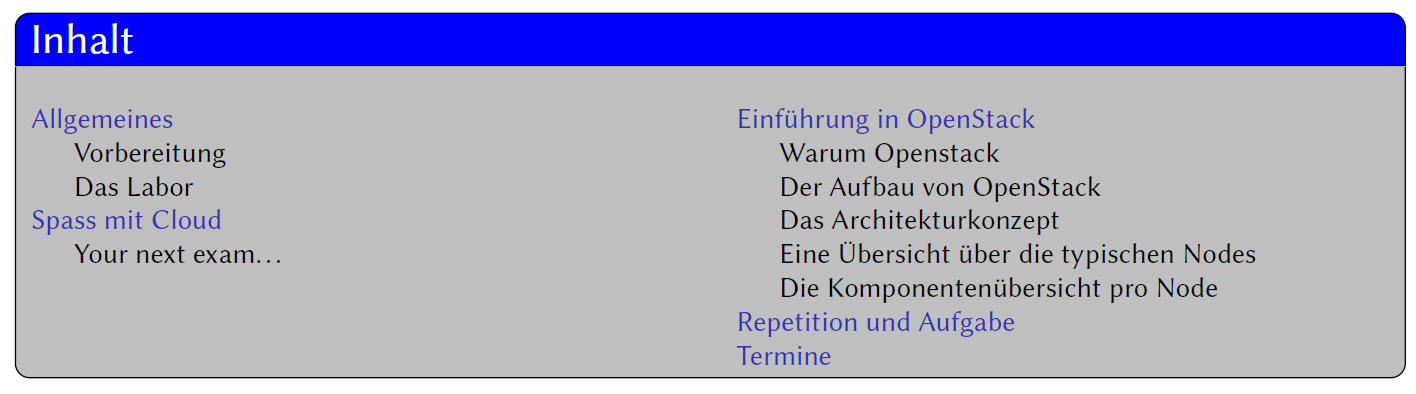


## Termine



# Woche 5

## Inhalt



## Allgemeines

### Vorbereitung

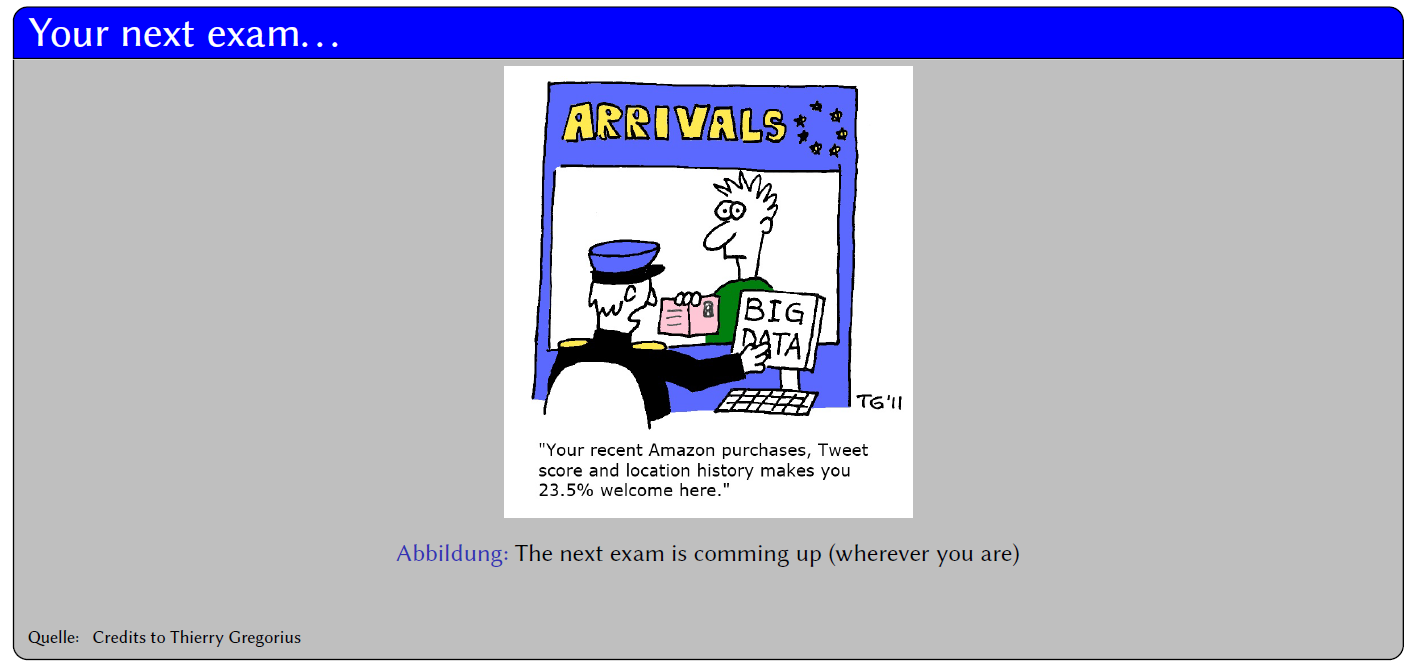


### Das Labor



## Spass mit Cloud

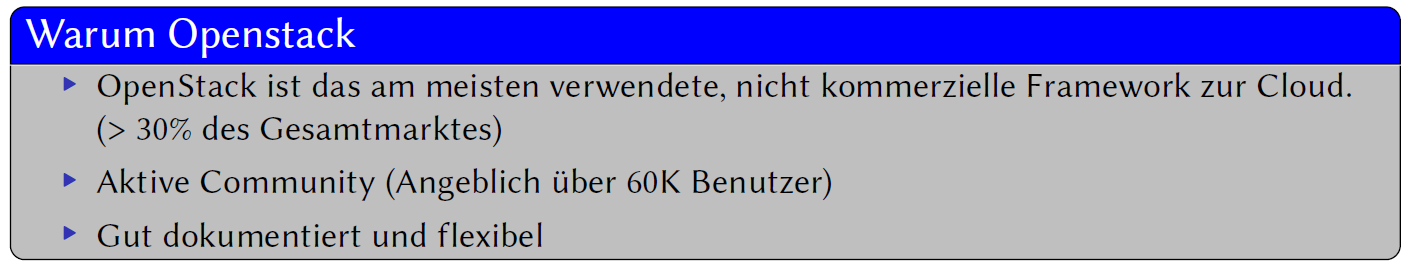
### Your next exam...



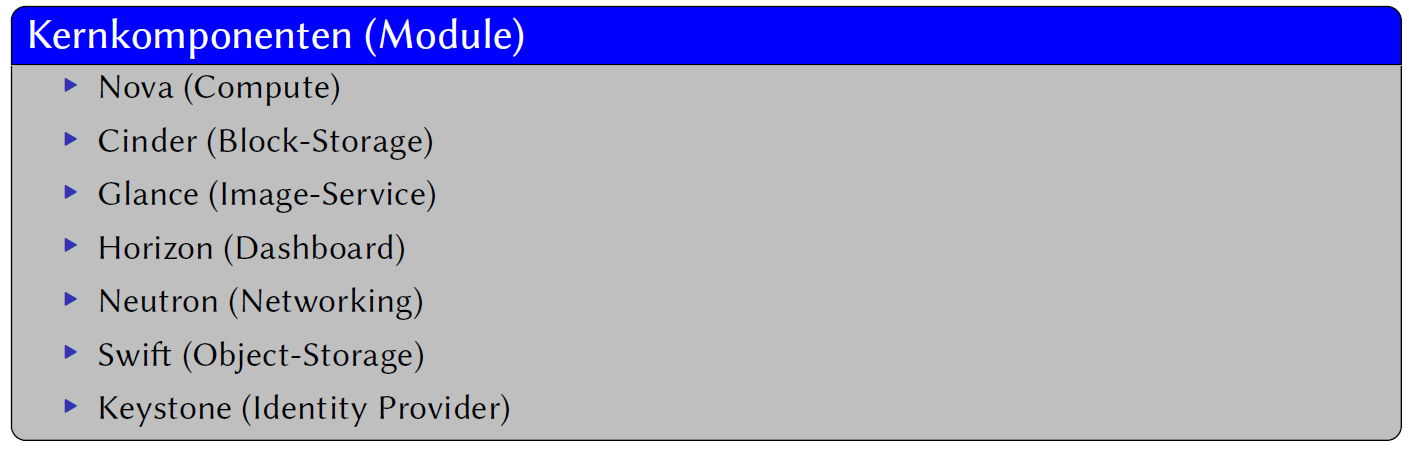
Soviel zu Examen und der Versuch, ihnen zu entkommen.

## Einführung in OpenStack

### Warum Openstack



### Der Aufbau von OpenStack



OpenStack besteht grundsätzlich aus sehr vielen verschiedenen Komponenten (40+). Anbei nur gerade die sieben wichtigsten Hauptkomponenten:

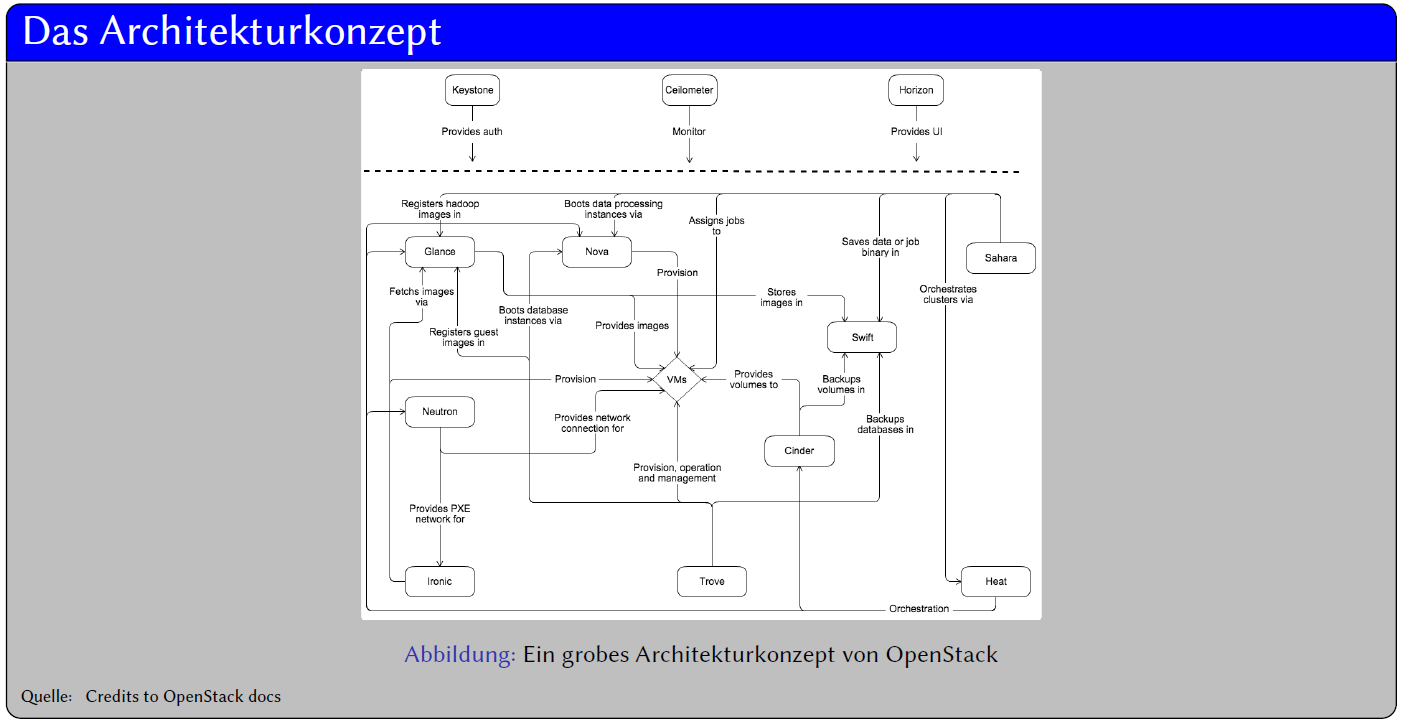
* Nova: Nova ist das Core Module, dass das Computing eines OpenStack-Clusters zur Verfügung stellt
* Cinder: Cinder stellt die Block-Storage als Basisstorage zur Verfügung.
* Glance: Mit Glance wird dem Framework ein Imageservice für Templates und Medien zur Verfügung gestellt. Dieses Modul baut auf dem Objekt-Store von Swift auf.
* Horizon: Horizon ist das Dashboard von OpenStack und bietet viele Möglichkeiten das Interface anzupassen.
* Neutron: Neutron stellt den Networking-Teil für das Framework zur Verfügung. Im Idealfall ist das Management der Netzwerkinterfaces einer Compute-Node vollständig abstrahiert durch dieses Modul.
* Swift: Dieses Modul stellt einen Object-Storage für das Projekt zur Verfügung.
* Keystone: Keystone ist ein Identity-Provider. Er kümmert sich um die Vereinheitlichung des Dienstzugriffes.

Neben diesen Hauptkomponenten gibt es noch viele weiter für:

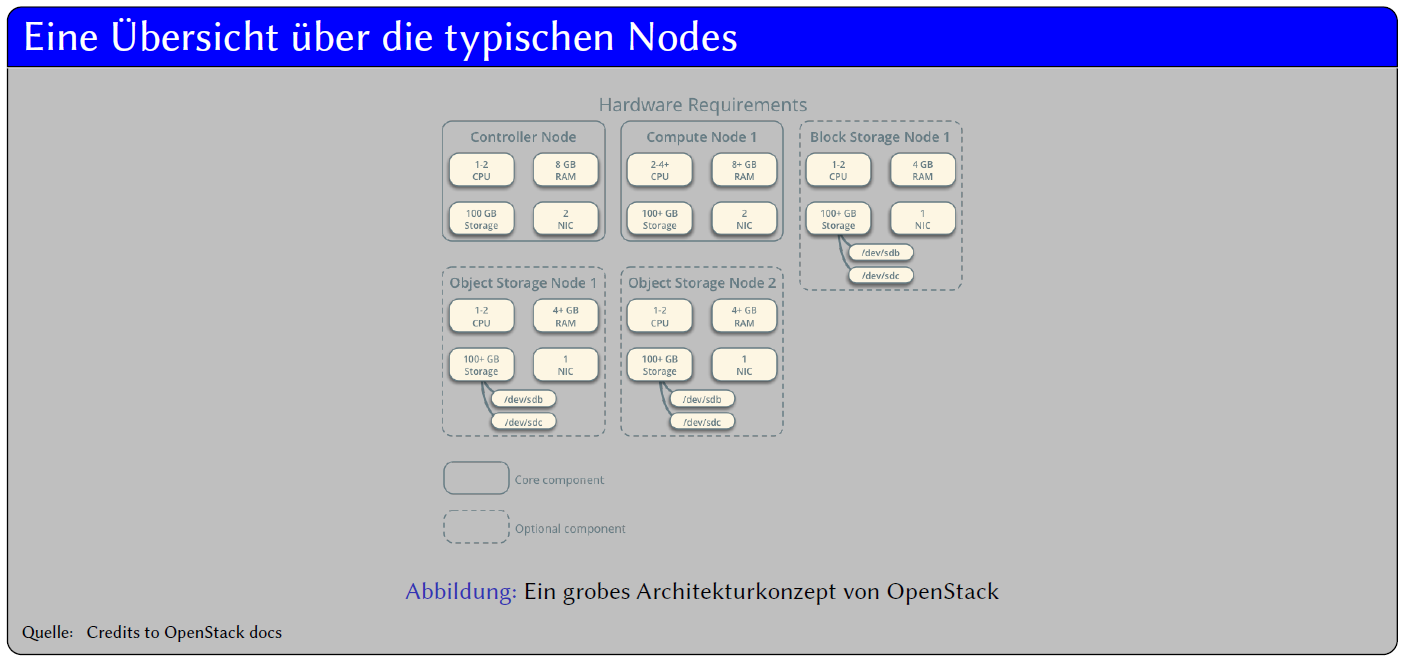
* Container Management
* Verrechnung
* Loadbalancer und DNS
* Schlüsselverwaltung (kann als eine funktionale Erweiterung zu Keystone gesehen werden)
* Indexierung von Inhalten (z.B. für eine Suche)
* Workflow Management
* Und viele mehr...

Eine Übersicht über die einzelnen Komponenten und Ihre Möglichkeiten finden Sie unter https://www.openstack.org/

### Das Architekturkonzept



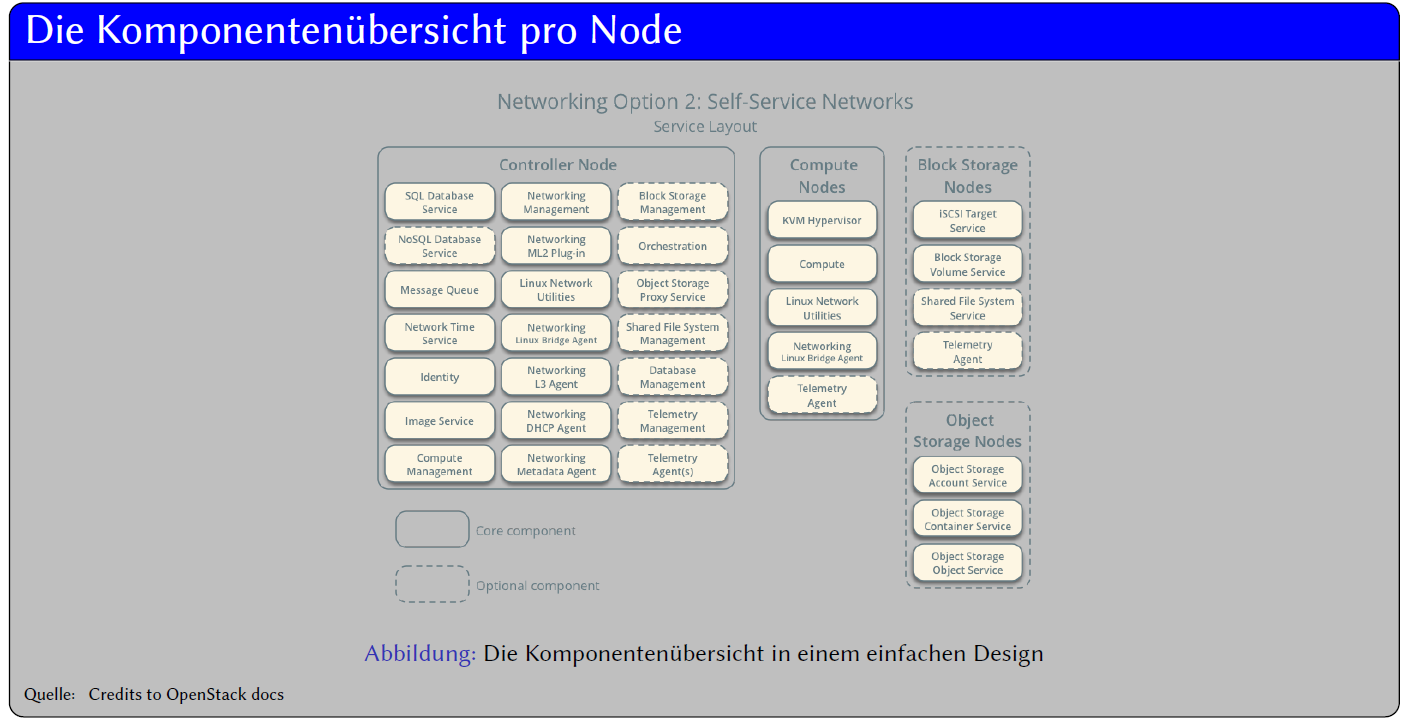
### Eine Übersicht über die typischen Nodes



Diese Architektur ist natürlich absolut minimalistisch und ohne jegliche Redundanz zu verstehen. Die Controller-Node umfasst sehr viele einzelne Dienste. Diese Node müsste natürlich im Minimum zweifach ausgeführt werden. Auch Compute-Nodes sind typischerweise natürlich mehrfach vorhanden um überhaupt zu skalieren.

Die Storage-Nodes sind traditionell aufgebaut. Der HCI-Ansatz ist hier ganz klar nicht ersichtlich. Storages werden bei diesem Ansatz typischerweise von Compute getrennt. Es ist zwar möglich OpenStack als HCI zu betreiben (Das macht beispielsweise das kommerzielle Produkt “Nutanix”, das auf OpenStack aufbaut), wird aber nur selten gemacht, weil die Kernarchitektur eigentlich anders ausgelegt und verwendet wird.

### Die Komponentenübersicht pro Node



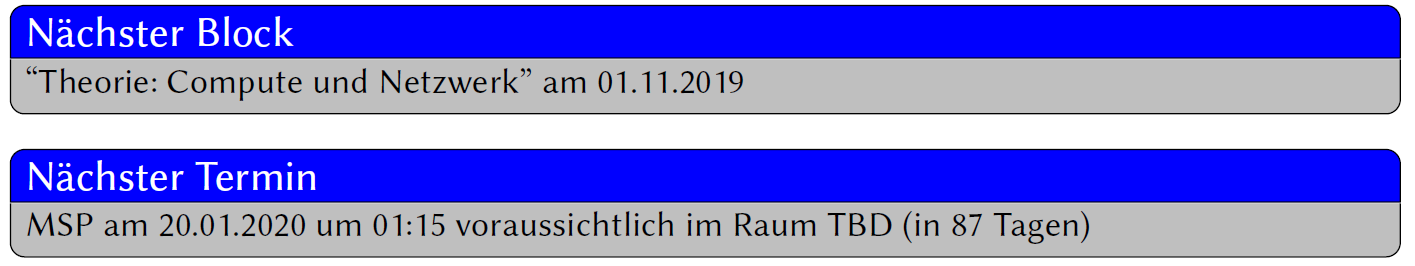
Bei diesem Design übernimmt de Contoller-Node den Löwenanteil der Koordinations-Arbeit. Sie reguliert alle Funktionen von OpenStack. Die Storages und Compute-Nodes agieren als zentral kontrollierte Slaves.

Die Controller-Node stellt die Core-Services als zentrales API (REST) zur Verfügung. Es gilt zu beachten, dass nicht alle Komponenten zwingend erforderlich sind (auch nicht die Core-Komponenten). So ist Beispielsweise der L3 Networking Agent nur erforderlich, wenn Selfservice bei L3-Netzwerkkomponenten (z.B. Loadbalancer oder Firewall) erforderlich ist.

## Repetition und Aufgabe



## Termine



# Woche 6