Inhaltsverzeichnis

Cloud-init

Sie kennen die Grundelemente von YAML und erkennen diese in Cloud-Init-Dateien.

YAML ist eine einfache Textdatei mit einer definierten Syntax, die einfach zu lesen ist. Zu den Grundelementen von YAML gehören *Collections (Darstellung einer Liste)* und *Scalars (Darstellung von einzelnen Werten/Daten)*, die unterschiedlich gemappt werden (z.B. key-value-pairs oder Aufzählungen):

2.1. Collections

YAML's block collections use indentation for scope and begin each entry on its own line. Block sequences indicate each entry with a dash and space (“`- `”). Mappings use a colon and space (“`: `”) to mark each key/value pair. Comments begin with an octothorpe (also called a “hash”, “sharp”, “pound” or “number sign” - “`#`”).

Example 2.1 Sequence of Scalars (ball players)

```
- Mark McGwire
- Sammy Sosa
- Ken Griffey
```

Example 2.2 Mapping Scalars to Scalars (player statistics)

```
hr:  65    # Home runs
avg: 0.278 # Batting average
rbi: 147   # Runs Batted In
```

Example 2.3 Mapping Scalars to Sequences (ball clubs in each league)

```
american:
- Boston Red Sox
- Detroit Tigers
- New York Yankees
national:
- New York Mets
- Chicago Cubs
- Atlanta Braves
```

Example 2.4 Sequence of Mappings (players' statistics)

```
-
  name: Mark McGwire
  hr:   65
  avg:  0.278
-
  name: Sammy Sosa
  hr:   63
  avg:  0.288
```

Sie können die Vorteile von Cloud-init erläutern.

Cloud-init konfiguriert das Betriebssystem während der Installation und automatisiert die Konfiguration von Software und Paketen. Statt jeden Server einzeln zu installieren oder ein Skript auf alle Server zu laden und auszuführen, kann man eine Konfigurationsdatei bei der Installation der Instanz mitgeben und Cloud-init erledigt den Rest. Dies ist ein grosser Vorteil, da die Automatisierung der Konfiguration viel Aufwand spart.

Weitere Vorteile von Cloud-init:

- **Plattformübergreifend**: Cloud-init kann auf verschiedenen Cloud-Plattformen wie AWS, GCP und Azure verwendet werden.

- **Flexibilität:** Cloud-init bietet viele Konfigurationsoptionen wie Paketinstallation, Benutzer- und Gruppenkonfiguration, Netzwerkkonfiguration und benutzerdefinierte Skripts.

- **Effizienz**: Cloud-init automatisiert die Konfiguration während der Installation, was Zeit und Aufwand bei der manuellen Konfiguration spart.

- **Wiederholbarkeit**: Mit Cloud-init kann man die gleiche Konfiguration auf mehrere Instanzen anwenden.

- **Skalierbarkeit**: Cloud-init ermöglicht es, Konfigurationen auf eine große Anzahl von Instanzen anzuwenden, was ideal ist, wenn man eine grosse Anzahl von Servern verwalten muss.

- **Sicherheit**: Cloud-init kann durch die automatisierte Konfiguration von Benutzerkonten und Netzwerkeinstellungen dazu beitragen, Sicherheitsrisiken zu minimieren.

- **Einfache Verwaltung**: Cloud-init bietet eine einfache Möglichkeit, Instanzen zu konfigurieren und zu verwalten, indem alle Konfigurationsoptionen in einer einzigen Datei gespeichert werden.

Sie kennen die wichtigsten Elemente von Cloud-init und können diese auch anwenden.

Zu den wichtigsten Elemente von Cloud-init gehören folgende:

- **Cloud-Config**: YAML-Datei mit Konfigurationsoptionen für die Instanz (Benutzer- und Gruppenkonfiguration, Netzwerkkonfiguration und Paketinstallation)

- **DataSource:** Die Quelle, von der Cloud-init Konfigurationsdaten bezieht (z.B. AWS EC2-Metadaten). Dazu gehören Metadaten wie die Instanz-IP-Adresse oder die Region, in der sich die Instanz befindet.

- **User-Data:** Möglichkeit, benutzerdefinierte Skripts und Konfigurationsdaten an die Instanz zu übergeben

- **Modules**: Funktionen, die von Cloud-init zur Ausführung von Konfigurationsaufgaben verwendet werden, wie z.B. runcmd, um Shell-Befehle auszuführen, oder write-files, um Dateien auf der Instanz zu schreiben

---

Sie finden sich mit der offiziellen Referenz zurecht und können ein Element nachschlagen oder neue Elemente entdecken und anwenden.

Offizielle Referenzen:
YAML: https://yaml.org/spec/1.2.2/

Cloud-Init: https://cloudinit.readthedocs.io/en/latest/ (Reference: https://cloudinit.readthedocs.io/en/latest/reference/index.html)

Cloud-Config: https://cloudinit.readthedocs.io/en/latest/reference/examples.html

---

Sie können eine virtuelle Instanz mit Cloud-init installieren.

AWS Instanz

Sie kennen den Begriff "Infrastructure As Code" und können ihn erklären.

Erklärung von GitLab M346:

Infrastruktur als Code ist ein Paradigma (grundsätzliche Denkweise) zur Infrastruktur-Automation.

Es basiert auf konsistenten und wiederholbaren Definitionen (Code) für die Bereitstellung von Systemen und deren Konfiguration.

Produkte sind u.a. Cloud-init, Vagrant, TerraForm, CLIs etc.

Definition von Infrastruktur als Code:

- Infrastruktur als Code ist ein Ansatz zur Automatisierung der Infrastruktur, der auf Praktiken aus der Softwareentwicklung basiert.
- Dabei werden konsistente, wiederholbare Prozesse für die Bereitstellung und Änderung von Systemen und deren Konfiguration verwendet.
- Änderungen werden an Deklarationen (Code) vorgenommen und dann durch automatisierte Prozesse auf Systeme übertragen.
- Infrastruktur als Code hat sich in den anspruchsvollsten Umgebungen bewährt. Für Unternehmen wie Amazon, Netflix, Google und Facebook sind IT-Systeme nicht nur geschäftskritisch. Sie sind das Geschäft!

Ziele von Infrastruktur als Code:

- Die IT-Infrastruktur unterstützt und ermöglicht Veränderungen, statt ein Hindernis oder eine Einschränkung zu sein.
- Änderungen am System sind Routine, ohne Drama oder Stress für Benutzer oder IT-Mitarbeiter.
- IT-Mitarbeiter verbringen ihre Zeit mit wertvollen Dingen, die ihre Fähigkeiten einbeziehen, und nicht mit sich wiederholenden Routineaufgaben.

- Benutzer können die benötigten Ressourcen definieren, bereitstellen und verwalten, ohne dass IT-Mitarbeiter dies für sie tun müssen.
- Teams können sich einfach und schnell von Fehlern erholen, anstatt davon auszugehen, dass Fehler vollständig verhindert werden können.
- Verbesserungen werden kontinuierlich vorgenommen und nicht durch teure und riskante „Urknall“ -Projekte.
- Lösungen für Probleme werden durch Implementierung, Test und Messung bewiesen, anstatt sie in Besprechungen und Dokumenten zu diskutieren.

Kürzere Erklärung:

Infrastructure as Code (IaC) ist ein Ansatz zur Automatisierung der IT-Infrastruktur, der auf Softwareentwicklungstechniken basiert. Dabei werden konsistente und wiederholbare Prozesse eingesetzt, um Systeme und deren Konfigurationen bereitzustellen und zu ändern. Das Ziel ist es, die Infrastruktur flexibler, effizienter und zuverlässiger zu machen, damit Änderungen an der Infrastruktur einfacher und stressfrei durchgeführt werden können. Produkte wie Cloud-init, Vagrant und Terraform unterstützen diesen Ansatz.

Sie können Produkte nennen, die "Infrastructure As Code" anbieten (mit Hilfe des Internet).

- Cloud-init
- Vagrant
- Terraform
- AWS CloudFormation
- Chef
- Puppet
- Ansible
  etc.

Virtualisierung

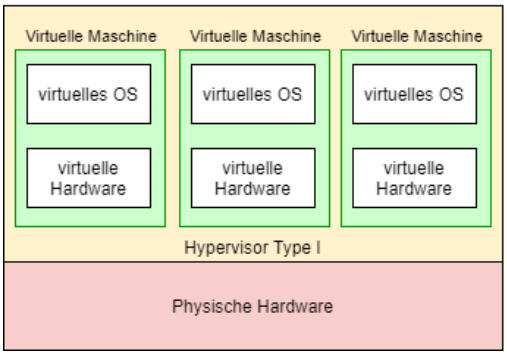
Sie können den Begriff der "Virtualisierung" erläutern.

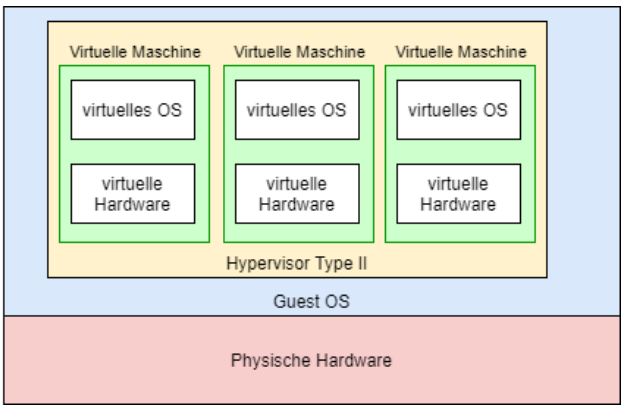
Erklärung GitLab M346:

"A hypervisor is a process that separates a computer's operating system and applications from the underlying physical hardware. Usually done as software although embedded hypervisors can be created for things like mobile devices.

The hypervisor drives the concept of virtualization by allowing the physical host machine to operate multiple virtual machines as guests to help maximize the effective use of computing resources such as memory, network bandwidth and CPU cycles." (networkworld.com)

Grafisch darstellen kann man dies wie folgt.





Sie sehen hier, dass es zwei Typen von Hypervisors gibt:

- Typ 1 (native): Läuft direkt auf der Hardware des Hostsystems.
- Typ 2 (hosted): Läuft als Applikation auf dem Betriebssystem.

Kürzere Erklärung:

Die Virtualisierung ermöglicht es, eine oder mehrere virtuelle Versionen eines Computers zu erstellen und auszuführen, die sich wie separate physische Geräte verhalten (VMs, Container). Durch die Virtualisierung können Ressourcen wie Computerprozessoren, Speicher, Netzwerke und andere Systeme auf eine effizientere und flexiblere Weise genutzt werden. Es gibt verschiedene Arten von Virtualisierung, wie zum Beispiel Hardware-Virtualisierung, Betriebssystem-Virtualisierung oder Anwendungs-Virtualisierung.

Sie können grob erklären, welche Rolle ein "Hypervisor" in der Virtualisierung spielt.

Ein Hypervisor ist eine Software, die es ermöglicht, mehrere virtuelle Maschinen auf einer physischen Maschine auszuführen. Sie stellt eine Abstraktionsschicht zwischen der Hardware und den virtuellen Maschinen dar.

Der Hypervisor teilt die physischen Ressourcen der Hardware wie Prozessor, Arbeitsspeicher, Festplatten und Netzwerkadapter in mehrere virtuelle Maschinen auf und verwaltet den Zugriff darauf. Er stellt sicher, dass jede virtuelle Maschine ihren eigenen isolierten Speicherbereich und ihre eigenen virtuellen Ressourcen hat, damit sie unabhängig voneinander ausgeführt werden können, ohne sich gegenseitig zu beeinträchtigen.

Es gibt zwei Arten von Hypervisoren: Typ 1 (native) und Typ 2 (hosted). Typ 1 Hypervisoren werden direkt auf der Hardware installiert und laufen als primäre Schicht auf der Maschine. Typ 2 Hypervisoren hingegen laufen auf einem normalen Host-Betriebssystem.

Sie kennen den Unterschied zwischen Typ 1 und Typ 2 Hypervisors und können den auch erklären.

Ein Typ 1 Hypervisor wird direkt auf der physischen Hardware des Hostcomputers installiert. Er ist das erste System, das auf dem Computer ausgeführt wird und teilt die Hardware-Ressourcen des Computers in mehrere virtuelle Maschinen auf. Typ 1 Hypervisoren werden direkt auf der Hardware installiert und laufen als primäre Schicht auf der Maschine.

Ein Typ 2 Hypervisor läuft auf einem Host-Betriebssystem. Der Host-Betriebssystem-Kernel teilt die Hardware-Ressourcen in virtuelle Maschinen auf und der Typ 2 Hypervisor wird innerhalb des Host-Betriebssystems ausgeführt. Typ 2 Hypervisoren laufen auf einem Host-Betriebssystem.

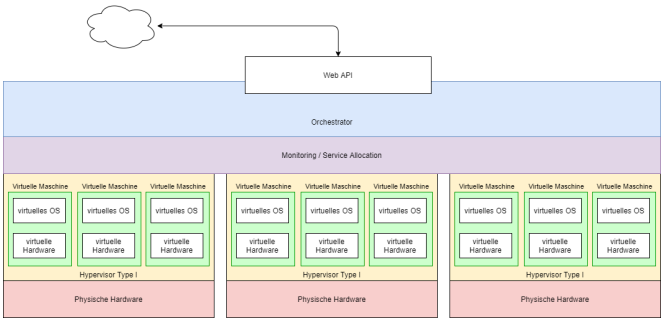
Die wichtigste Unterscheidung zwischen den beiden Typen besteht darin, dass ein Typ 1 Hypervisor direkt auf der Hardware installiert wird und somit den Host-Betriebssystem-Kernel und dessen Overhead umgeht. Dadurch wird eine höhere Leistung und eine bessere Skalierbarkeit erreicht. Ein Typ 2 Hypervisor hingegen läuft auf einem normalen Betriebssystem und teilt die Ressourcen mit anderen Anwendungen, was möglicherweise zu einer schwachen Leistung führen kann.

Sie können den Begriff "Hyperscaler" erklären im Zusammenhang mit Cloud-Systemen.

Erklärung GitLab M346:
"Hyperscalers are large cloud service providers that can provide services such as computing and storage at enterprise scale." (redhat.com)

"Hyperscalers get their name from hyperscale computing, a method of processing data that allows for software architecture to scale and grow as increased demand is added to the system." (redhat.com)

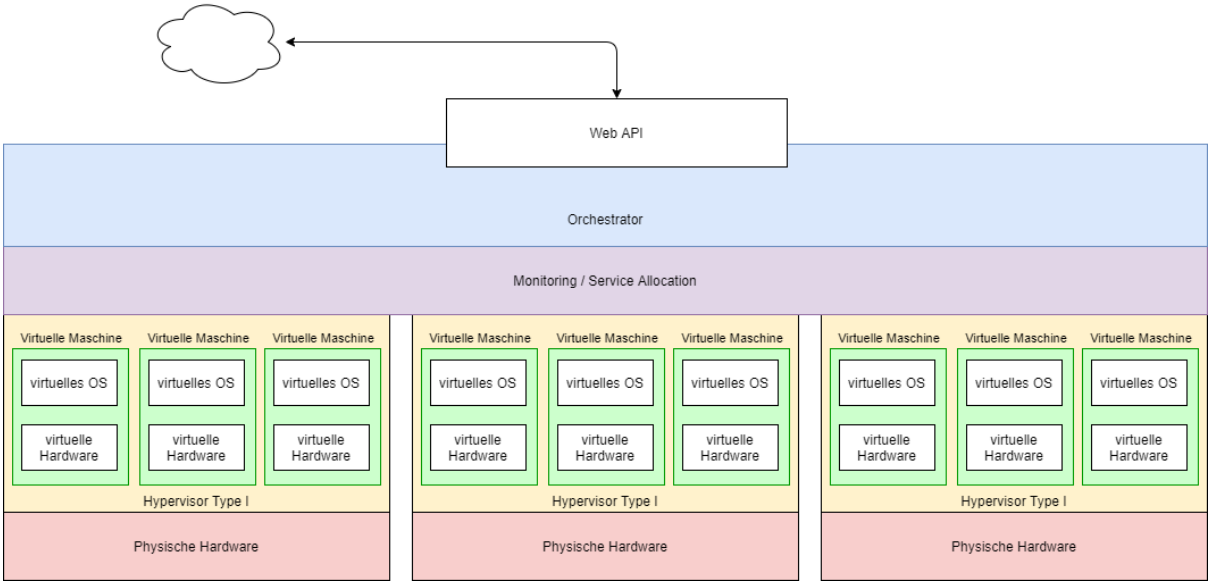
Vorstellen kann man sich dies gut, wenn man die Technologie des Hypervisors verwendet und diese vervielfacht. Natürlich werden hier nur Typ 1 Hypervisors verwendet, wegen der Performance.



Das Grundprinzip ist bei alle grossen Anbietern das gleiche. Die Details können sich natürlich unterscheiden.

Kürzere Erklärung:
Der Begriff "Hyperscaler" bezieht sich auf Unternehmen, die über eine extrem große und skalierbare Cloud-Infrastruktur verfügen und Services wie Computing, Speicherung und Datenverarbeitung im Internet bereitstellen. Diese Unternehmen betreiben Rechenzentren, die oft weltweit verteilt sind und in der Lage sind, enorme Mengen an Daten zu verarbeiten und zu speichern. Zu den bekanntesten Hyperscalern gehören Unternehmen wie Amazon Web Services (AWS), Microsoft Azure, Google Cloud und IBM Cloud.

Sie kennen Schichten von Cloud-Systemen.



Betriebsmodelle

Sie kennen die drei Betriebsmodelle und können Sie erklären.

On Premise: Alle Ihre Hard- und Software werden von Ihnen betrieben in eigenen oder gemieteten Datenzentren.

Hybrid Cloud: Sie haben ihre Server und Applikationen teilweise On Premise und teilweise in der Cloud. Dies kann beispielsweise bedeuten, dass kritische Daten und Anwendungen On Premise gehostet werden, während weniger sensible Ressourcen in der öffentlichen Cloud abgelegt werden.

Heutzutage trifft man dieses Modell am Öftesten an. Sobald eine Firma Office 365 Lizenzen betreibt, sind zumindest Teile der Infrastruktur in der Cloud, auch wenn proprietäre Software noch auf eigenen Servern betrieben wird.

Cloud-Native: In diesem Modell ist die komplette Infrastruktur und alle Applikationen in der Cloud.

Sie kennen die Unterteilung zwischen Public & Private Cloud und können Sie erklären.

Public Cloud: Dienste und Ressourcen werden von einem Drittanbieter gehostet und über das Internet bereitgestellt. Sie werden in der Regel auf der Grundlage eines Abonnementmodells bezahlt und bieten Nutzern Skalierbarkeit und Flexibilität.

Private Cloud: Eine Cloud-Infrastruktur, die von einer einzelnen Organisation gehostet wird und in der Regel innerhalb ihrer eigenen Firewall läuft. Private Clouds bieten mehr Kontrolle und Sicherheit, erfordern jedoch in der Regel auch höhere Kosten und Ressourcen.

Sie können Beispiele zu den Betriebsmodellen nennen.

On-Premise: Viele Unternehmen nutzen On-Premise-Infrastrukturen, um ihre eigenen Datenzentren zu betreiben und Anwendungen lokal zu hosten. Beispiele dafür sind Banken, Regierungsbehörden und Krankenhäuser.

Hybrid Cloud: Viele Unternehmen nutzen eine Kombination aus öffentlicher und privater Cloud, um ihre Daten und Anwendungen zu verwalten. Beispiele dafür sind Einzelhändler, Energieunternehmen und Fertigungsunternehmen.

Cloud Native: Unternehmen, die von Anfang an in der Cloud entwickelt wurden, wie z.B. Netflix, Spotify und Airbnb, nutzen Cloud-native Infrastrukturen, um ihre Anwendungen zu hosten und zu skalieren.

Public Cloud: Viele Unternehmen nutzen öffentliche Cloud-Infrastrukturen, wie z.B. Amazon Web Services (AWS), Microsoft Azure oder Google Cloud Platform, um ihre Anwendungen und Daten zu hosten. Beispiele dafür sind Start-ups, digitale Agenturen und E-Commerce-Unternehmen.

Private Cloud: Viele große Unternehmen und Organisationen, wie z.B. Banken, Versicherungen und Regierungsbehörden nutzen private Cloud-Infrastrukturen, um ihre Daten und Anwendungen zu hosten.

Servicemodelle

Sie kennen die vier Servicemodelle und können Sie erklären.

Infrastruktur as a Service (**IaaS**): Dieses Servicemodell umfasst die Bereitstellung der Hardware (z.B. virtuelle Maschinen oder Netzwerk) in der Cloud. Sie mieten Rechenleistung, Speicher, etc und installieren auf diesen Instanzen ihre Software. Beispiele dazu sind

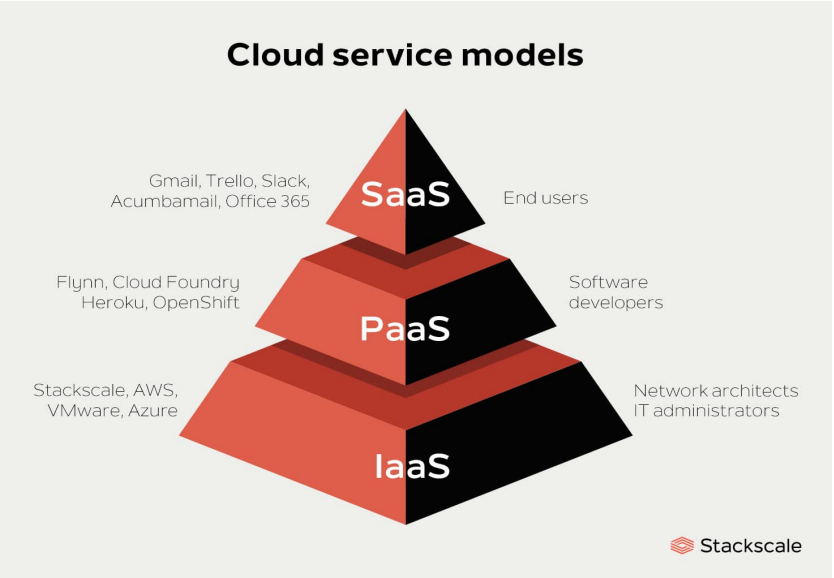
Microsoft Azure, Google Compute Engine (GCE), Amazon Web Services (AWS), Rackspace, Cisco Metapod

Platform as a Service (**PaaS**): In diesem Modell wird eine Plattform in der Cloud zur Verfügung gestellt, auf der Sie ihre Software, Service, etc. anbieten können. Sie verwenden dabei die Infrastruktur und SDKs der Hersteller und konzentrieren sich auf die Entwicklung der Software und nicht auf die Infrastruktur dahinter. Beispiele dafür sind AWS Elastic Beanstalk, Windows Azure, Heroku, Force.com, Google App Engine, Apache Stratos, OpenShift.

Software as a Service (**SaaS**): In diesem Modell wird die Software als Service in der Cloud gehostet. Sie verwenden die Software ohne, dass Sie sie lokal installieren müssen. Jede Webseite auf der Sie sich einloggen dient als Beispiel für SaaS (z.B. Google Workspace, Dropbox, Salesforce, Microsoft 365, etc)

Function as a Service (**FaaS**)/**Serverless**: Ist ähnlich wie PaaS. Während bei PaaS Infrastruktur für Sie bereitgestellt wird, wird in FaaS die Infrastruktur für Sie auch verwaltet. Der große Vorteil bei FaaS sind die reduzierten Kosten. Nur wenn ihre *Funktion* auch ausgeführt wird, fallen Kosten an.

Sie wissen, welches Servicemodell auf einem anderen aufbaut (z.B. PaaS auf IaaS).



Sie können Beispielprodukte für die Servicemodelle nennen (mit Hilfe des Internets)

Infrastructure-as-a-Service (IaaS): Amazon Web Services (AWS) EC2, Google Compute Engine, Microsoft Azure Virtual Machines.

Platform-as-a-Service (PaaS): Heroku, Google App Engine, Microsoft Azure App Service.

Software-as-a-Service (SaaS): Google Workspace, Microsoft Office 365, Salesforce, Dropbox.

Function-as-a-Service (FaaS): AWS Lambda, Google Cloud Functions, Microsoft Azure Functions, IBM Cloud Functions (ehemals OpenWhisk), Alibaba Cloud Function Compute

---

(Cloud-) Migrationsmodelle

Sie kennen die Begriffe der Migrationsmodellen.

Rehosting (Lift & Shift): Dabei wird eine bestehende On-Premise Applikation genommen und mit minimalen Anpassungen in der Cloud wieder in Betrieb genommen. Dabei wird auf das Servicemodell *IaaS* aufgesetzt. Die Applikation wird auf virtuellen Servern installiert.

Replatforming (Lift & Reshape): Ihre On-Premise Applikation wird soweit angepasst, dass Sie hinsichtlich der Infrastruktur optimal gehostet werden kann. z.B. kann eine Applikation mit ein paar Änderungen mit dem *PaaS* Servicemodell gehostet werden.

Refactoring / Replace: Dabei schreiben Sie Ihre Applikation so um, dass Sie optimal mit den Cloud Services betrieben werden kann. Normalerweise wechseln Sie hier auf *Microservices* betrieben mit *Kubernetes* oder auf das *FaaS* Modell. Natürlich sind Mischformen möglich.

Repurchasing: Möglicherweise evaluieren Sie die Marktlage und lösen Ihre bestehende Applikation mit einer bestehenden *SaaS* Lösung ab.

Sie kennen die Zielplattform der Migrationsmodellen.

Rehosting: Die Zielplattform ist in der Regel eine virtuelle Maschine (VM) oder eine Container-basierte Infrastruktur, die auf einer ähnlichen Hardware- oder Cloud-Plattform wie die Quellumgebung läuft.

Replatforming: Die Zielplattform ist eine verbesserte Version der aktuellen Plattform, die jedoch immer noch kompatibel mit der vorhandenen Anwendungsarchitektur ist. Beispielsweise kann eine Anwendung, die auf einer älteren Version eines Datenbanksystems läuft, auf eine neuere Version des gleichen Datenbanksystems migriert werden.

Refactoring: Die Zielplattform ist eine Cloud-native Plattform oder eine Mikrodienstarchitektur, die speziell für die Skalierung und Ausführung von Cloud-Anwendungen entwickelt wurde.

Repurchasing: Die Zielplattform ist eine vollständig neue Anwendungsplattform, die durch den Kauf einer Cloud-Lösung oder eines Software-as-a-Service (SaaS)-Anbieters bereitgestellt wird.

Bei einem gegebenen Szenario können Sie für ein Migrationsmodell argumentieren.

Rehosting (Lift & Shift): Dieses Modell ist in Situationen geeignet, in denen die Infrastruktur einer Organisation veraltet ist oder nicht mehr unterstützt wird und ein schneller Umzug in die Cloud erforderlich ist.

Replatforming (Lift & Reshape): Dieses Modell beinhaltet geringfügige Anpassungen an die Anwendung, um sie besser an die Cloud-Infrastruktur anzupassen. Dies kann bedeuten, dass Anwendungen in Container verpackt oder auf Serverless-Funktionen migriert werden. Dieses Modell ist in Situationen geeignet, in denen Anwendungen in der Cloud skalierbarer und kosteneffizienter gemacht werden sollen.

Refactoring / Replace: Dieses Modell beinhaltet eine umfassende Überarbeitung der Anwendung, um sie besser an die Cloud-Infrastruktur anzupassen und moderner zu gestalten. Dies kann bedeuten, dass Anwendungen in Microservices aufgeteilt oder mit neuen Technologien neu geschrieben werden. Dieses Modell ist in Situationen geeignet, in denen Anwendungen vollständig modernisiert werden sollen, um die Vorteile der Cloud voll auszuschöpfen.

Repurchasing: Dieses Modell beinhaltet den Ersatz der bestehenden Anwendung durch eine neue Cloud-native Anwendung, die bereits auf der Cloud-Infrastruktur ausgeführt wird. Dies kann bedeuten, dass vorhandene Anwendungen durch SaaS-Lösungen oder andere Cloud-native Anwendungen ersetzt werden. Dieses Modell ist in Situationen geeignet, in denen bestehende Anwendungen nicht mehr den Anforderungen entsprechen und eine vollständige Überarbeitung nicht praktikabel ist.

Speichermodelle

Sie kennen die vier Speichermodelle

Hot Storage: Bezeichnet Speicher, welcher häufig aufgerufen wird und nur sehr kleine Verzögerungen erlaubt.

Warm Storage: Bezeichnet Speicher, welcher regelmässig aufgerufen wird. Eine andere Bezeichnung ist z. B. "Cool Storage".

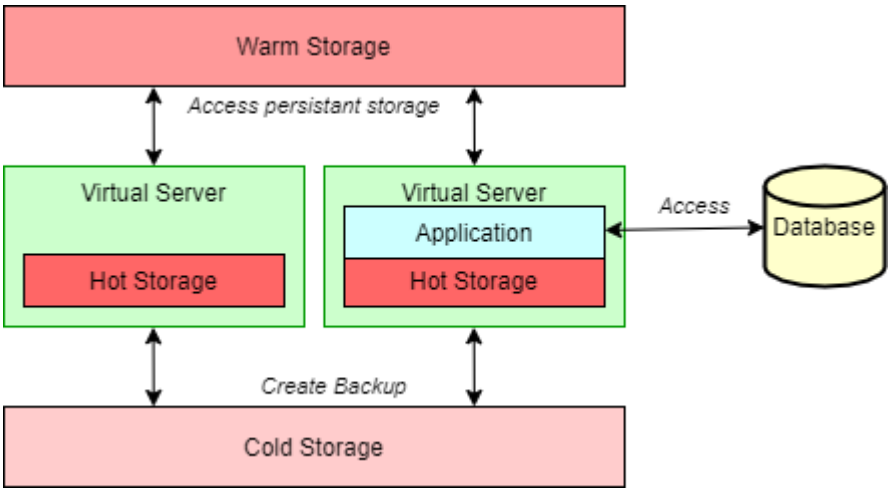
Cold Storage: Bezeichnet Speicher, welcher sehr selten aufgerufen wird. Eine andere Bezeichnung ist z.B. "Archive Storage".

Datenbank: Datenbanken eben!

Sie wissen für welchen Einsatz die Speichermodelle sinnvoll Verwendung finden.

Während Betriebssysteme und Applikationen *Hot Storage* benötigen für die Laufzeit, werden Daten meistens in Datenbanken ausgelagert. *Warm Storage* kommt, z. B. für temporäre Dateien in Frage oder Videos und Audios, die gelegentlich abgespielt werden. *Cold Storage* kann gut für Backups verwendet werden.

Schematisch kann man die Verwendung wie folgt abbilden.



Sie können Beispiele zur Benutzung der Speichermodelle geben.

Hot Storage: Daten, auf die ständig zugegriffen wird (z.B. aktuelle Kundendatenbanken, Echtzeit-Logdaten, Streaming-Daten). System-Beispiele: Oracle DB, Microsoft SQL Server, MongoDB

Warm Storage: Daten, auf die manchmal/gelegentlich zugegriffen wird (z.B. ältere Kundendatenbanken, Archivdaten, Sicherungsdaten, Daten für Analysen, Transaktionsdaten). System-Beispiele: HDFS, Amazon S3 Standard-Infrequent Access, Google Cloud Storage Nearline

Cold Storage: Daten, auf die selten bis gar nicht zugegriffen wird (z.B. Backups, Archivdaten). System-Beispiele: Amazon Glacier, Google Cloud Storage Coldline, Microsoft Azure Archive Storage

---

Sie kennen den Unterschied zwischen persistenten und flüchtigem Speicher und können diesen erklären, speziell im Zusammenhang mit virtuellen Instanzen.

Flüchtiger Speicher, auch bekannt als RAM, ist ein temporärer Speicher, der nur während des Betriebs der virtuellen Instanz besteht. Sobald die Instanz heruntergefahren wird, gehen alle im RAM gespeicherten Daten verloren.
Persistenter Speicher hingegen ist ein dauerhafter Speicher, der auch nach dem Herunterfahren der virtuellen Instanz erhalten bleibt. Daten können darauf gespeichert und bei Bedarf wiederhergestellt werden.

In virtuellen Instanzen wird flüchtiger Speicher zur temporären Speicherung von Daten während der Verarbeitung verwendet, während persistenter Speicher zur dauerhaften Speicherung von Daten und zur Datensicherung verwendet wird.

Sie kennen den Unterschied der Lese- und Schreibgeschwindigkeiten der Speichermodelle.

Hot Storage, der schnellste Typ, bietet in der Regel die höchste Lese- und Schreibgeschwindigkeit. Warm Storage bietet eine moderate Lese- und Schreibgeschwindigkeit, während Cold Storage aufgrund der Archivierung von Daten in der Regel eine längere Zugriffszeit hat.

Summary of access tier options

The following table summarizes the features of the hot, cool, and archive access tiers.

	Hot tier	Cool tier	Archive tier
Availability	99.9%	99%	Offline
Availability (RA-GRS reads)	99.99%	99.9%	Offline
Usage charges	Higher storage costs, but lower access and transaction costs	Lower storage costs, but higher access and transaction costs	Lowest storage costs, but highest access, and transaction costs
Minimum recommended data retention period	N/A	30 days[1]	180 days
Latency (Time to first byte)	Milliseconds	Milliseconds	Hours[2]
Supported redundancy configurations	All	All	LRS, GRS, and RA-GRS[3] only