Moduldokumentation

Modul Application Performance Management (apm)

Simon Wächter

2018

Inhalt

[1 Einleitung 3](#_Toc508201832)

[1.1 Einleitung 3](#_Toc508201833)

[1.2 Lernziele 3](#_Toc508201834)

[1.3 Prüfungen 3](#_Toc508201835)

[2 Woche 1 4](#_Toc508201836)

[2.1 Allgemeines 4](#_Toc508201837)

[2.1.1 Über dem Dozenten 4](#_Toc508201838)

[2.1.2 Der normale Betrieb 4](#_Toc508201839)

[2.1.3 Was dürfen sie erwarten 5](#_Toc508201840)

[2.2 Spass mit Netzwerken 6](#_Toc508201841)

[2.2.1 Ein anderes «Normdokument» 6](#_Toc508201842)

[2.3 Einführung 6](#_Toc508201843)

[2.3.1 Was ist eine Cloud 6](#_Toc508201844)

[2.3.2 Die Servicemodelle einer Cloud 6](#_Toc508201845)

[2.3.3 Der typische Aufbau einer Virtualisierungsinfrastruktur 7](#_Toc508201846)

[2.3.4 Die Skalierungsarten 7](#_Toc508201847)

[2.3.5 Ein typischer Loadbalancer 8](#_Toc508201848)

[2.3.6 Aufgaben und Übungen 9](#_Toc508201849)

[3 Woche 2 10](#_Toc508201850)

[3.1 Inhalt 10](#_Toc508201851)

[3.2 Repetition von Loadbalancer 10](#_Toc508201852)

[3.3 Spass mit Netzwerken (Datamining in Clustern) 10](#_Toc508201853)

[3.4 Ein Cluster 11](#_Toc508201854)

[3.4.1 Begrifflichkeiten in einem Cluster 11](#_Toc508201855)

[3.4.2 Typische Eigenschaften eines Clusters 11](#_Toc508201856)

[3.4.3 Typische Speichersysteme eines Clusters 12](#_Toc508201857)

[3.4.4 Klassifizierung von Hypervisoren 12](#_Toc508201858)

[3.4.5 Auswirkungen der Virtualisierung auf Applikationen 13](#_Toc508201859)

[3.5 Aufgaben und Übungen 13](#_Toc508201860)

[4 Woche 3 14](#_Toc508201861)

[4.1 Repetition 14](#_Toc508201862)

[4.2 Spass mit Netzwerken 14](#_Toc508201863)

[4.3 Reproduzierbares Testen 14](#_Toc508201864)

[4.4 Testen mit Apache JMeter 15](#_Toc508201865)

[4.4.1 Der Hauptbildschirm 15](#_Toc508201866)

[4.4.2 Überlegungen zum Testinhalt 15](#_Toc508201867)

[4.4.3 Überlegungen zur Testumgebung 16](#_Toc508201868)

[4.5 Eine Klassenübung 16](#_Toc508201869)

[4.6 Aufgaben und Übungen 16](#_Toc508201870)

[5 Woche 4 17](#_Toc508201871)

# Einleitung

## Einleitung

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

## Lernziele

Das Modul beinhaltet folgende Lernziele:

* Studierende dieses Moduls können für eine Applikation Performanceziele aus nicht funktionalen Anforderungen ableiten. Sie sind in der Lage die nötigen Performancemessungen zu planen und durchzuführen.
* Studierende dieses Moduls können Messresultate richtig interpretieren und die daraus notwendigen Massnahmen am Deployment, an der Konfiguration sowie an der Implementation der Applikation vorschlagen bzw. durchführen.
* Studierende dieses Moduls kennen verschiedene Massnahmen zur Performanceoptimierung und können diese auch effektiv einsetzen.
* Studierende dieses Moduls verstehen die Anforderungen von Betreibern von Data Centers. Sie sind in der Lage sinnvolle Quality of Service-Parameter auszuhandeln.

## Prüfungen

Die Modulnote setzt sich zu 100% aus zwei Semesterprüfungen zu je 50% zusammen.

# Woche 1

## Allgemeines

### Über dem Dozenten



### Der normale Betrieb



Jeder Wochenblock richtet sich nach folgendem Muster:

* Unterrichtsblock (2 Lektionen Theorie): Dieser Unterricht wird in Referatsform gehalten. Sie können ihn attraktiver gestalten, wenn
  + Sie den zu behandelnden Stoff bereits aufgearbeitet haben.
  + Fragen zum Stoff stellen.
* Übungsblockblock (1 Lektion Einzelarbeit): In diesem Block erhalten Sie jeweils eine Aufgabe, die sie bearbeiten werden. Sie benötigen hierfür einen Computer mit Internetanbindung und eine virtuelle Maschine mit einem Linux.
* Die Hausaufgaben: Die Hausaufgaben bauen üblicherweise auf den Übungsaufgaben auf. Sie sind eine Vertiefung und Vorbereitung auf die Prüfung. Wenn sie die Aufgaben gut lösen, sollten sie wesentlich bessere Noten im Assessment haben.

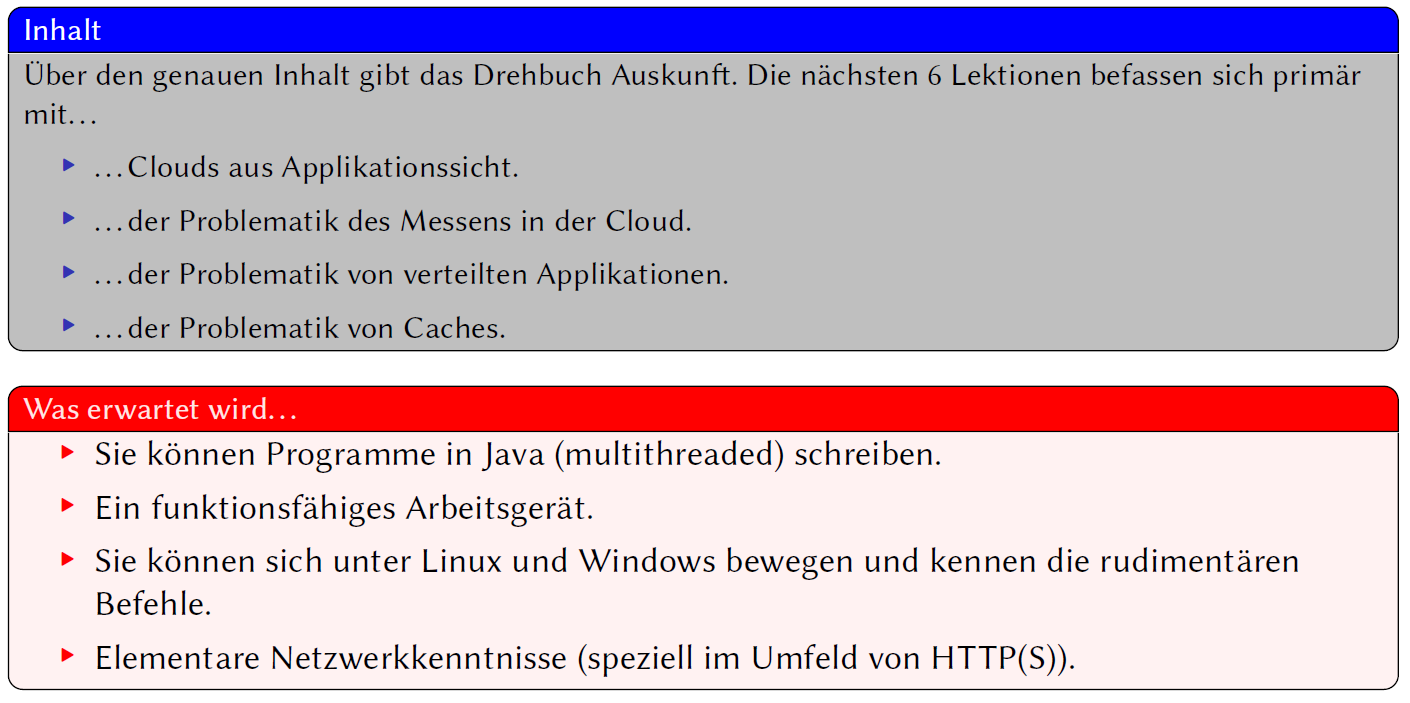
Sie finden an folgenden Orten die Informationen über dieses Modul:

* Im AD
* Drehbuch
* Wenn Sie weitere Quellen benötigen, sind sie in den abgegebenen Unterlagen verlinkt oder aber sie sind direkt in das PDF-Dokument eingebettet.

Die Slides befinden sich jeweils mindestens 3 Arbeitstage vor der Lektion in der Ablage (im “AD”). Die Aufgaben gliedern sich immer gleich:

* Sie arbeiten den Stoff jeweils vor.
* Sie lösen die Übung (Wenn sie in der Lektion nicht bis zum Ende kommen).
* Sie lösen die Aufgaben.
* Das erste Assessment ist ohne Hilfsmittel.

### Was dürfen sie erwarten

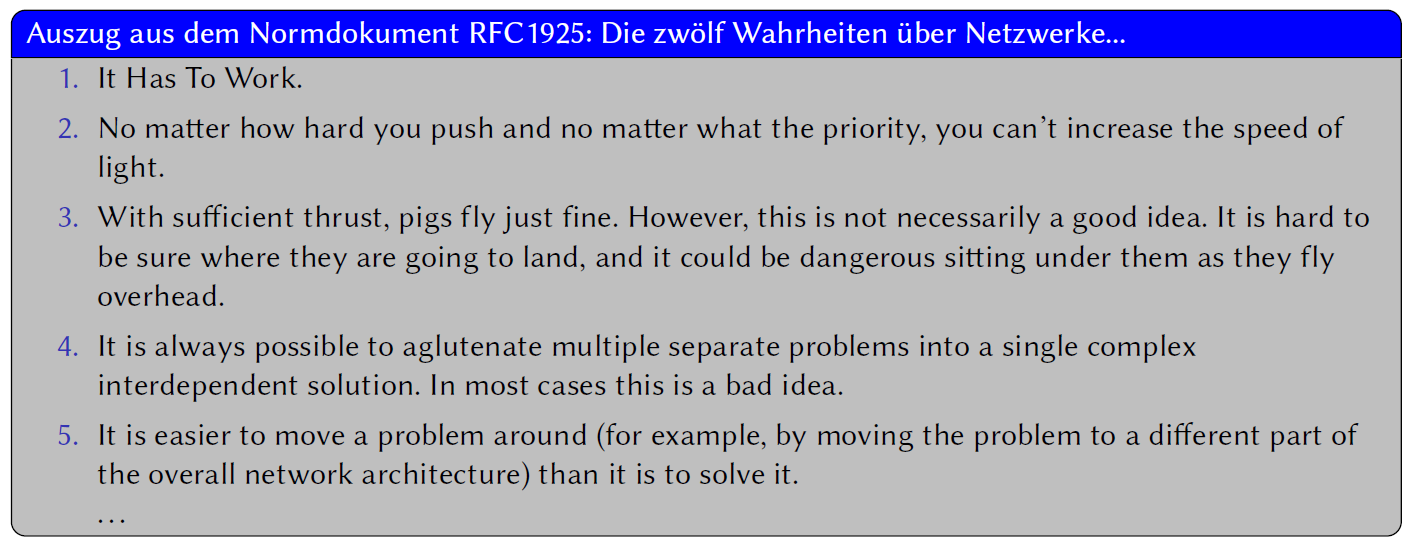


Gewisse Dinge sind für die folgenden Lektionen vorausgesetzt:

* Sie können Programme in Java (Multithreaded) schreiben. Dies bedeutet, Sie kennen die Grundlagen von Threads, Semaphoren, Monitoren, Synchronisation und Interprozess-Kommunikation.
* Ein funktionsfähiges Arbeitsgerät. Es funktionieren alle gängigen Betriebssysteme wie Mac, Windows oder Linux. Es ist gut 8GB RAM zu haben (die Übungen laufen auch mit 4 aber holpriger)
* Sie können sich unter Linux und Windows bewegen und kennen die rudimentären Befehle.
* Elementare Netzwerkkenntnisse (speziell im Umfeld von HTTP(S)).

## Spass mit Netzwerken

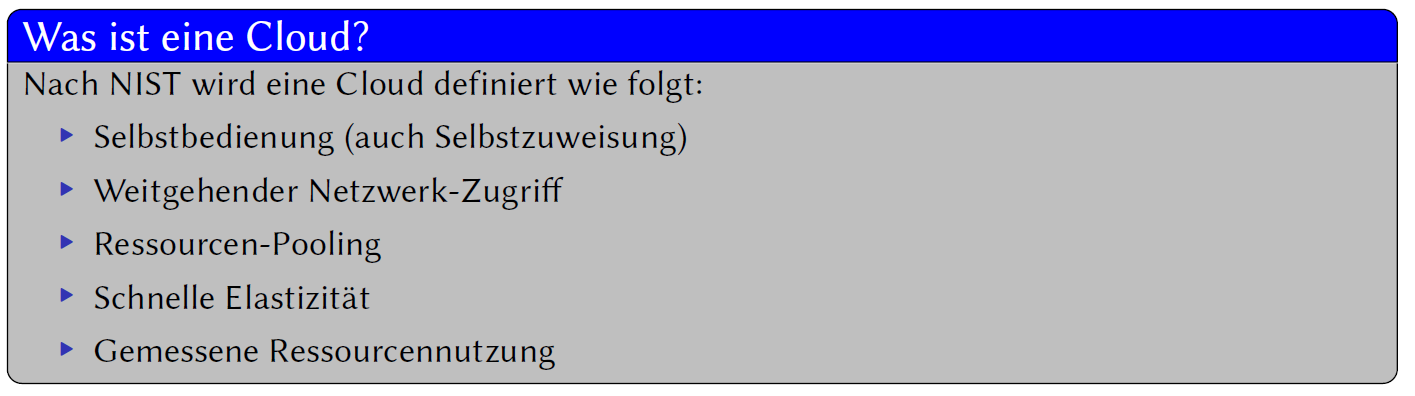
### Ein anderes «Normdokument»



Der vorliegende Auszug stammt aus dem Normen-Dokument RFC1925. Nicht alle Normen sind gleich ernst gemeint und es ist sicher von Bedeutung zu wissen, dass obiges Dokument am 1. April eingereicht worden ist.

## Einführung

### Was ist eine Cloud



Nach NIST wird eine Cloud definiert wie folgt:

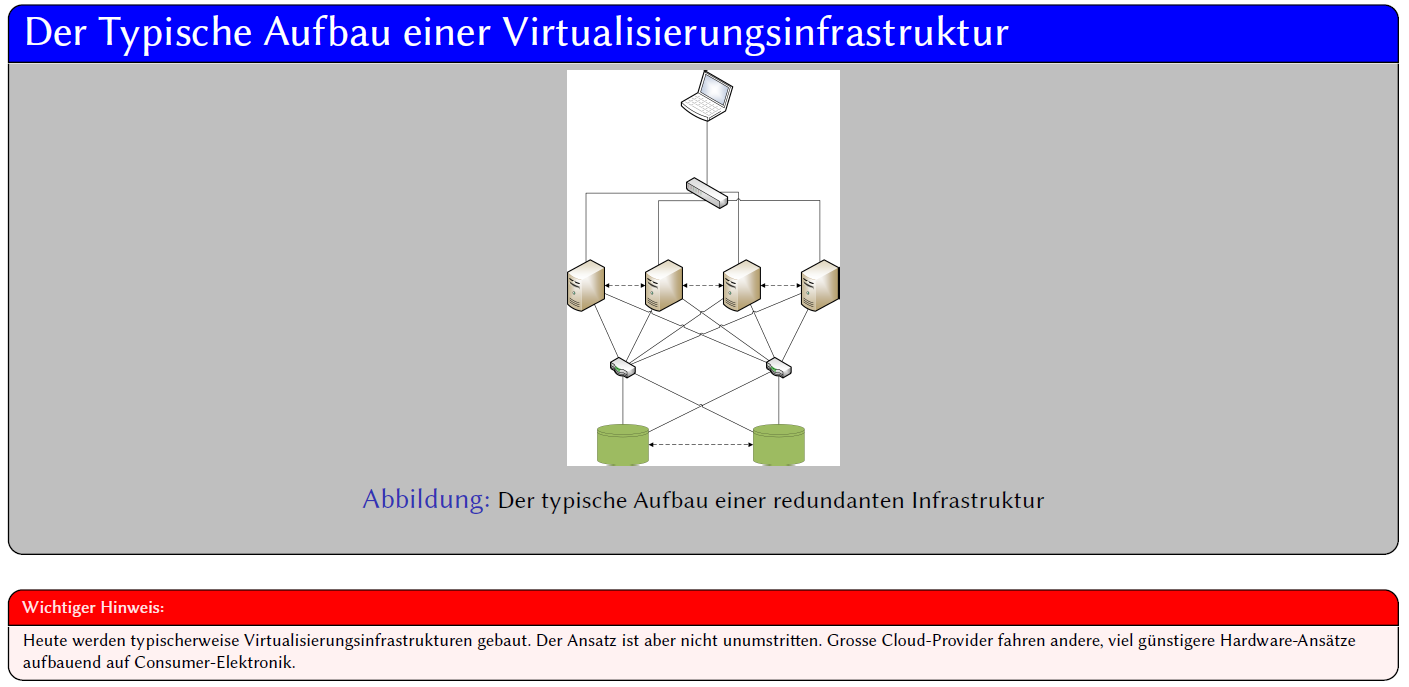
* Selbstbedienung (auch Selbstzuweisung): Das Bereitstellen Ressourcen (z.B. CPUs, Disk-Kapazität, Arbeitsspeicher) läuft automatisch ohne Interaktion mit dem Service Provider ab.
* Weitgehender Netzwerk-Zugriff: Die Services sind mit Standard-Mechanismen über das Netz verfügbar
* Ressourcen-Pooling: Die Ressourcen werden vom Anbieter gebündelt in Pools. Die Benützer können Ressourcen aus diesen Pools beziehen ohne dass sie wissen, wo genau sich die Ressourcen Befinden.
* Schnelle Elastizität: Services können schnell bezogen und auch wieder zurückgegeben werden.
* Gemessene Ressourcennutzung: Die Ressourcennutzung wird gemessen und überwacht.

### Die Servicemodelle einer Cloud



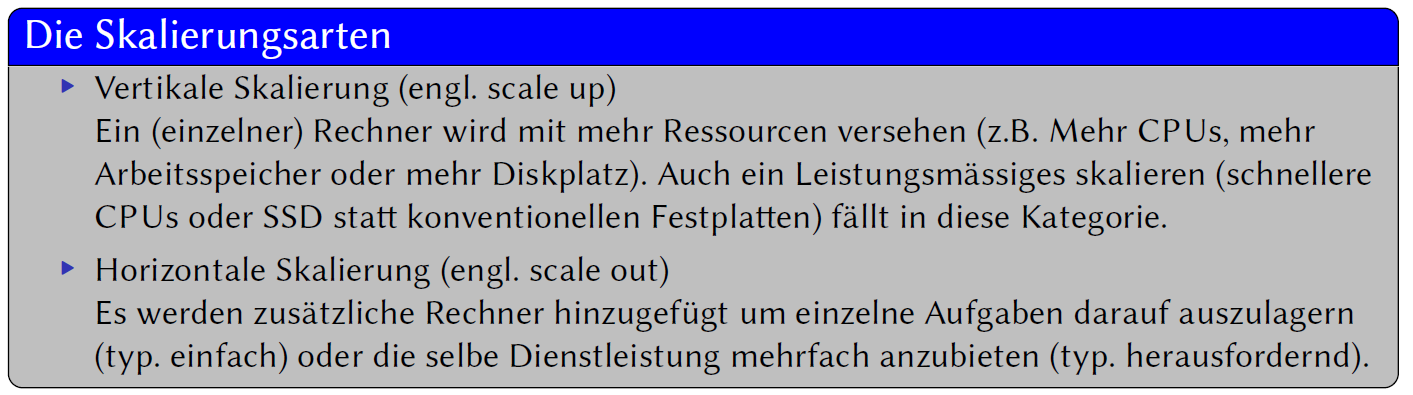
* Infrastructure as a Service (IaaS): Es werden nur die Hardware-Ressourcen zur Verfügung gestellt (CPU, RAM, Disk). Für Betriebssysteme oder Software ist der Kunde zuständig. Manchmal wird auch das Bereitstellen eines Betriebssystems hinzugerechnet.
* Platform as a Service (PaaS): Es wird eine Infrastruktur (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.

### Der typische Aufbau einer Virtualisierungsinfrastruktur



Die typische Virtualisierungsinfrastruktur wird (nach Lehrbuch) aufgebaut wie auf der Grafik gezeigt. Das ist aber aus heutiger Sicht nicht mehr zeitgemäss. Heute werden möglichst günstige Server/Computer verwendet. Der Interne Datenspeicher wird für die virtuellen Maschinen (VMs) verwendet. Eine komplizierte Replikation sorgt dafür, dass Daten zeitnah auch auf anderen Hosts zur Verfügung stehen.

### Die Skalierungsarten



Die typischen Skalierungsarten aus Softwaresicht sind:

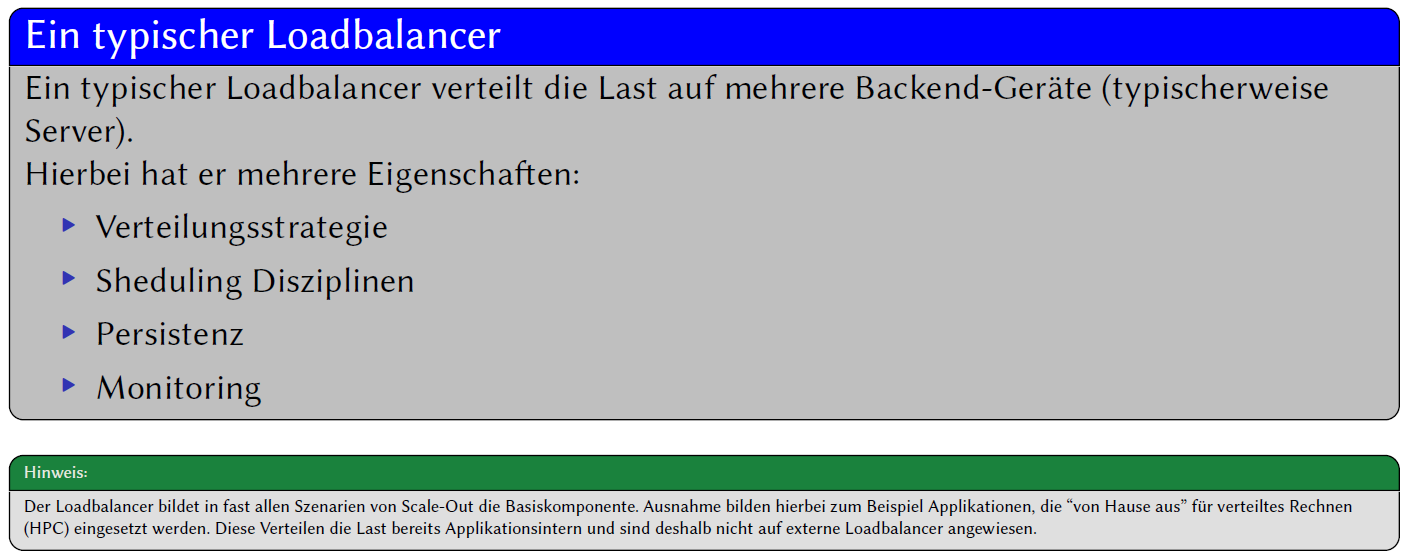
* Vertikale Skalierung (engl scale up): Ein (einzelner) Rechner wird mit mehr Ressourcen versehen (z.B. Mehr CPUs, mehr Arbeitsspeicher oder mehr Diskplatz). Auch ein Leistungsmässiges skalieren (schnellere CPUs oder SSD statt konventionellen Festplatten) fällt in diese Kategorie.
* Horizontale Skalierung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

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 eines Wochenendes (dann werden typischerweise die Full-Backups gemacht) gesichert werden können.

Virtuelle Maschinen sind typischerweise klein (aus Effizienzgründen).

### Ein typischer Loadbalancer



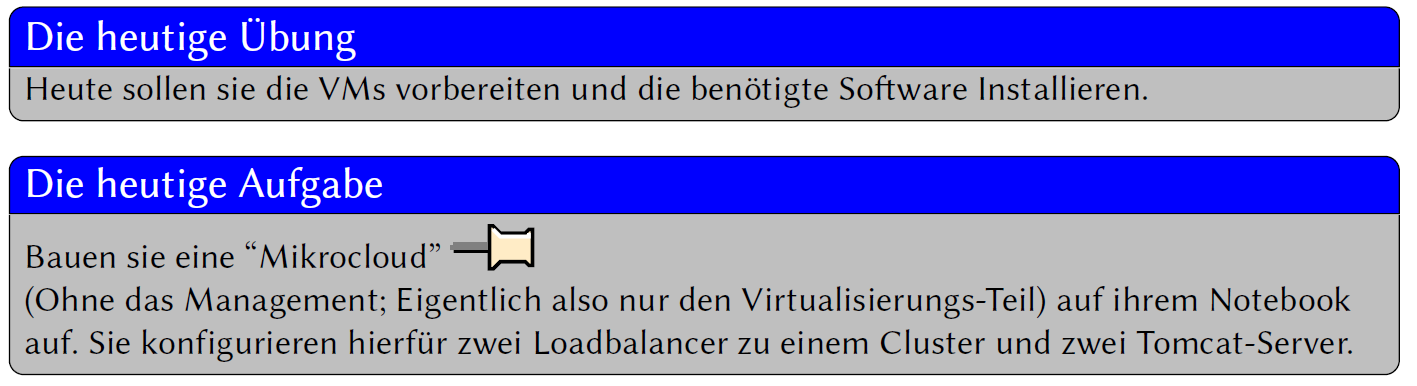
Ein typischer Loadbalancer verteilt die Last auf mehrere Backend-Geräte (typischerweise Server). Hierbei hat er mehrere Eigenschaften:

* Verteilungsstrategie: Nach welcher Strategie werden neu ankommende Verbindungen verteilt. Typische Strategien sind “Round robin” (via DNS oder direkt) oder “load distribution”
* Sheduling Disziplinen: Nach welchem Grundsatz werden Backends zugeteilt. Typische Strategien sind hier beispielsweise FIFO (First In First Out) oder Earliest Deadline First.
* Persistenz: Wie werden eingehende Verbindungen von bekannten Clients identifiziert, damit sie immer an das Selbe Backend geleitet werden. Das kann nach Protokollkriterien erfolgen (z.B. Senderadresse) oder nach Inhalt (z.B. ein spezielles Cookie ist gesetzt).
* Monitoring: Wie werden die Backends überwacht. Über einen simplen Ping, nimmt er TCP-Verbindungen auf einem bestimmten Port an oder über eine vollständige Prüfung respektive Statusabfrage.

Das sind aber nur die grundlegenden Eigenschaften. Fortschrittlichere Geräte können auch arbeiten wie intelligentes Routing oder Authentisierung übernehmen. Das sind aber beides streng genommen Eigenschaften eines sogenannten Reverse-Proxies.

Wir legen den Fokus auf sogenannte Serverside-Loadbalancer. Die Clientside-Lodabalancer spielen nur eine extrem untergeordnete Rolle in der Praxis.

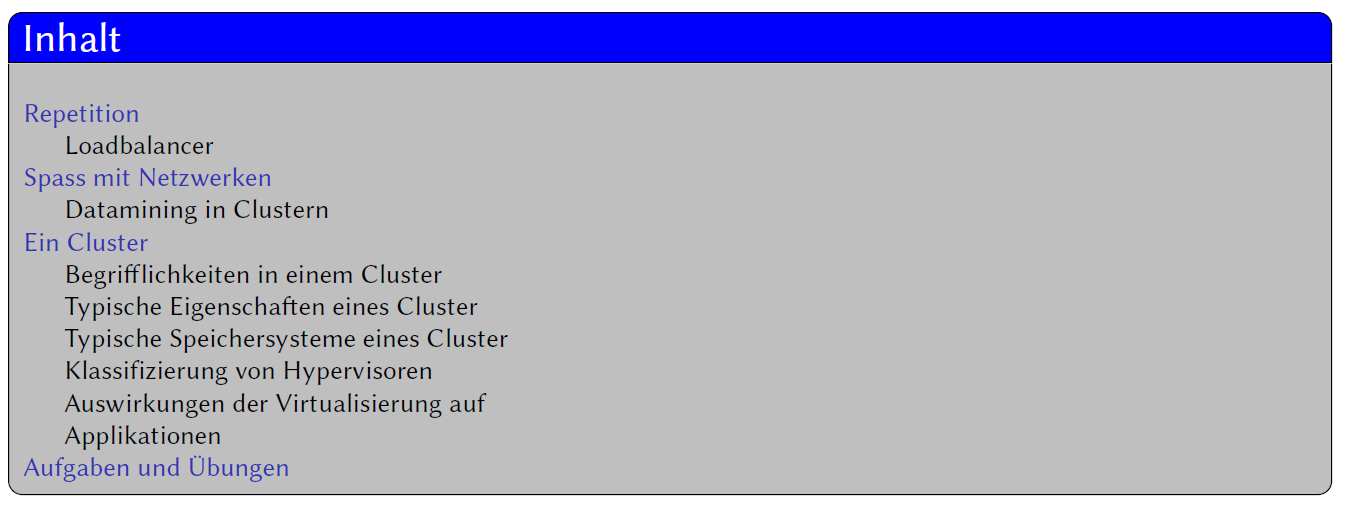
### Aufgaben und Übungen



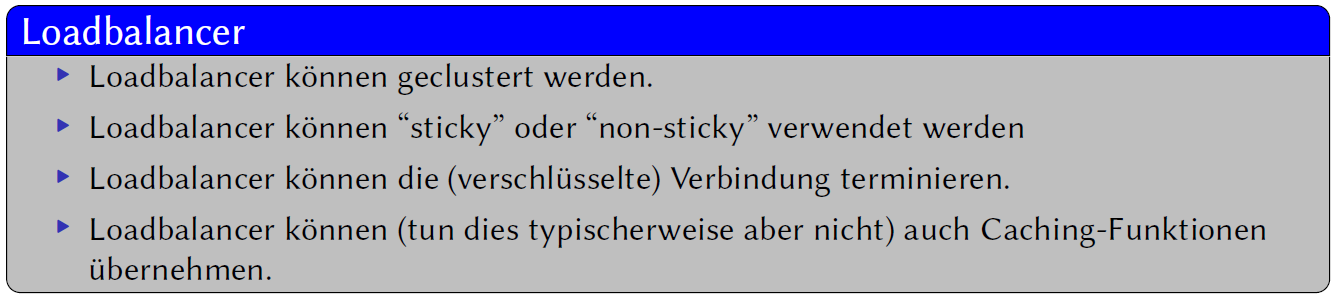
Sorgen Sie dafür, dass Ihre Arbeitsmittel nächstes Mal einsatzbereit sind.

# Woche 2

## Inhalt



## Repetition von Loadbalancer



* Loadbalancer können geclustert werden.
* Loadbalancer können “sticky” oder “non-sticky” verwendet werden
* Loadbalancer können die (verschlüsselte) Verbindung terminieren.
* Loadbalancer können (tun dies typischerweise aber nicht) auch Caching-Funktionen übernehmen.

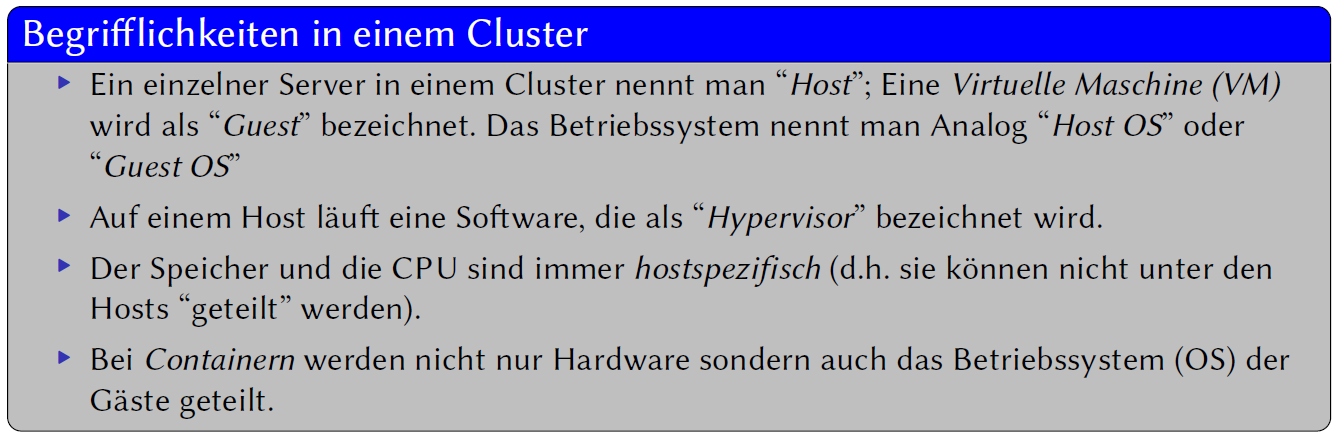
## Spass mit Netzwerken (Datamining in Clustern)



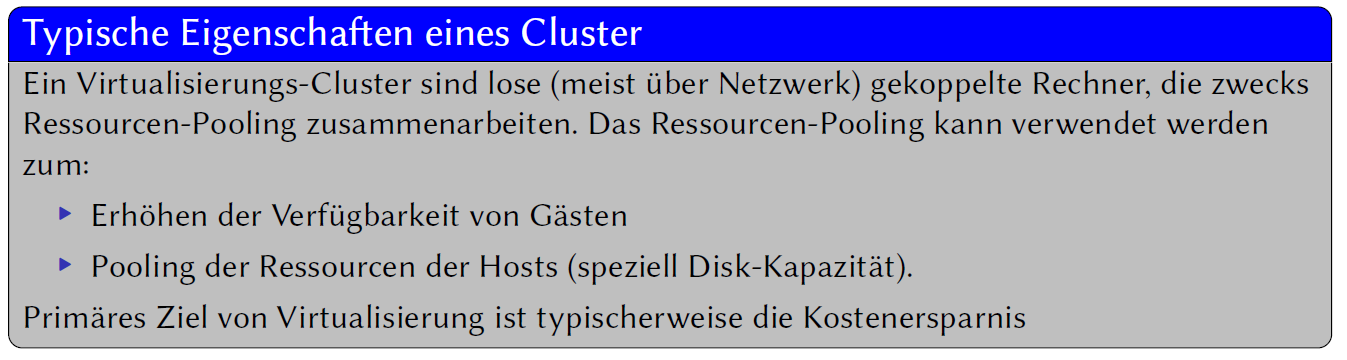
Beim Datamining in BigData werden fast schon “magische Zusammenhänge” ersichtlich. Diese Zusammenhänge können falsch oder auch richtig sein. Auf jeden Fall bieten sie derzeit noch Stoff für viele absurde Geschichten.

## Ein Cluster

### Begrifflichkeiten in einem Cluster



### Typische Eigenschaften eines Clusters



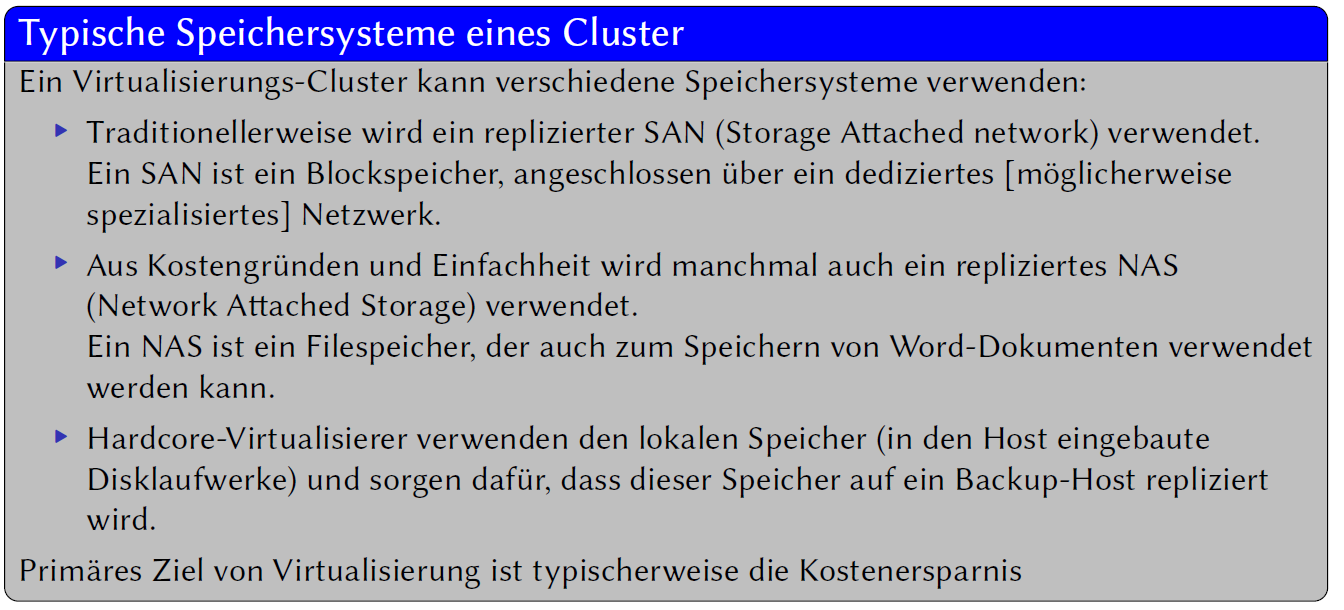
Die Ursprünge der Virtualisierung liegen in den alten Mainframe-Zeiten. Häufig wird IBM mit der Virtual Machine Facility/370 (1972) als Urvater der heutigen Virtualisierung bezeichnet.

Ein Virtualisierungs-Cluster sind lose (meist über Netzwerk) gekoppelte Rechner, die zwecks Ressourcen-Pooling zusammenarbeiten. Das Ressourcen-Pooling kann verwendet werden zum:

* Erhöhen der Verfügbarkeit von Gästen. Hierbei wird unterschieden zwischen:
  + Fault Tolerance (Ein Guest läuft zweifach auf zwei unabhängigen Hosts synchron)
  + High Availability (Beim Ausfall eines Hosts werden seine Virtuellen Maschinen auf einem anderen Host gestartet)
* Pooling der Ressourcen der Hosts (speziell Disk-Kapazität). Die Diskkapazität ist nicht die einzige Form des Poolings. Da viele Betriebssysteme identische Speicherblöcke haben, werden auch identische Speicherbereiche (Disk oder RAM) von verschiedenen Virtuellen Maschinen zusammengelegt. Man spricht hierbei von “Deduplizierung”. Die Deduplizierung hat in der Vergangenheit auch dafür gesorgt, dass Übergriffe auf andere virtuelle Maschinen möglich wurden.

Primäres Ziel von Virtualisierung ist typischerweise die Kostenersparnis

### Typische Speichersysteme eines Clusters



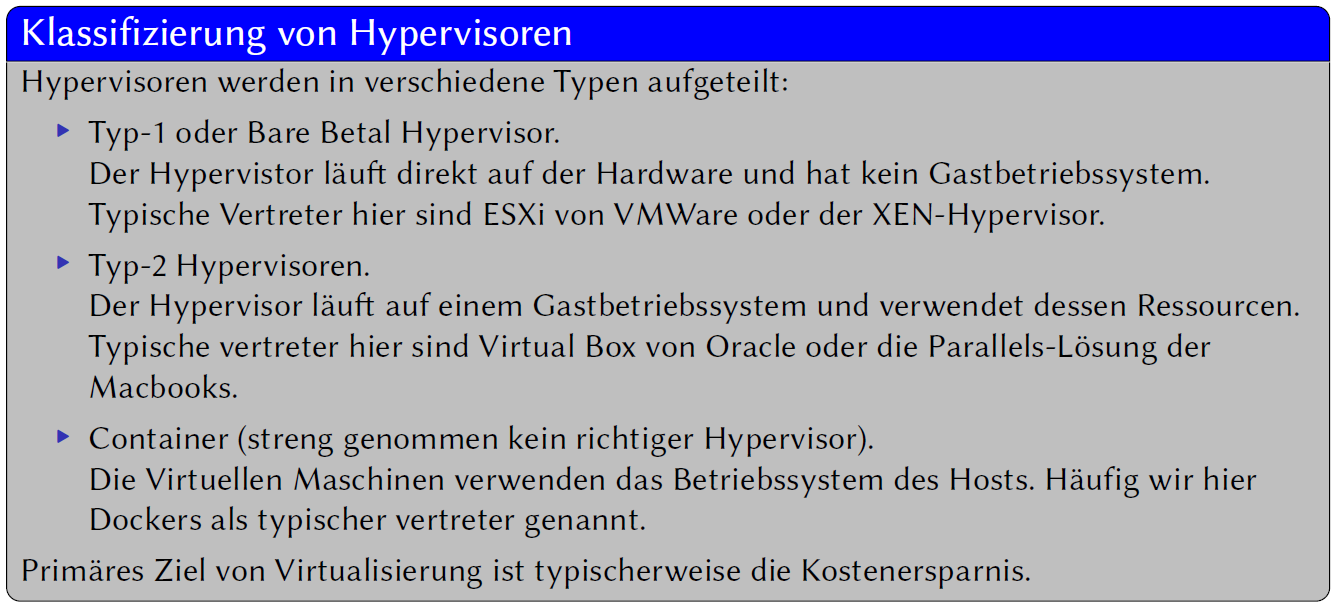
Ein Virtualisierungs-Cluster kann verschiedene Speichersysteme verwenden:

* Traditionellerweise wird ein replizierter SAN (Storage Attached network) verwendet. Ein SAN ist ein Blockspeicher, angeschlossen über ein dediziertes [möglicherweise spezialisiertes] Netzwerk.
* Aus Kostengründen und Einfachheit wird manchmal auch ein repliziertes NAS (Network Attached Storage) verwendet. Ein NAS ist ein Filespeicher, der auch zum Speichern von Word-Dokumenten verwendet werden kann.
* Hardcore-Virtualisierer verwenden den lokalen Speicher (in den Host eingebaute Disklaufwerke) und sorgen dafür, dass dieser Speicher auf einen Backup-Host repliziert wird.

Primäres Ziel von Virtualisierung ist typischerweise die Kostenersparnis

Die Disksubsysteme haben manchmal auch unterschiedliche Qualitäten. So wird beispielsweise auf dem Host möglicherweise der Zugriff auf die Disk durch einen Cache-Speicher unterstützt. Die Disksysteme können schnelle SSDs, Server-Festplatten oder langsame Consumer-Festplatten sein.

### Klassifizierung von Hypervisoren

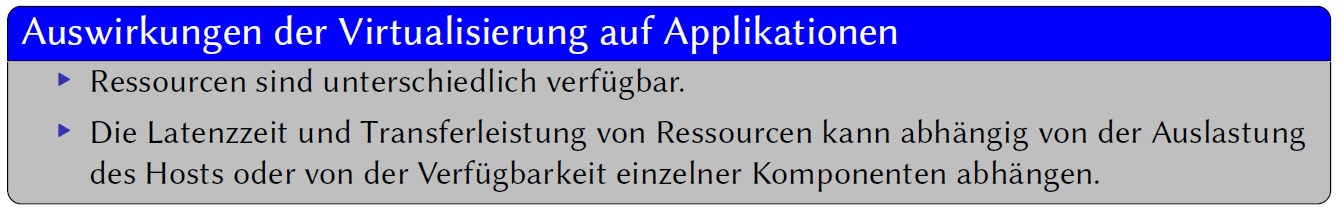


Hypervisoren werden in verschiedene Typen aufgeteilt:

* Typ-1 oder Bare Betal Hypervisor: Der Hypervistor läuft direkt auf der Hardware und hat kein Gastbetriebssystem. Typische Vertreter hier sind ESXi von VMware oder der XEN-Hypervisor.
* Typ-2 Hypervisoren: Der Hypervisor läuft auf einem Gastbetriebssystem und verwendet dessen Ressourcen. Typische Vertreter hier sind Virtual Box von Oracle oder die Parallels-Lösung der Macbooks.
* Container (streng genommen kein richtiger Hypervisor): Die Virtuellen Maschinen verwenden das Betriebssystem des Hosts. Häufig wir hier Dockers als typischer Vertreter genannt.

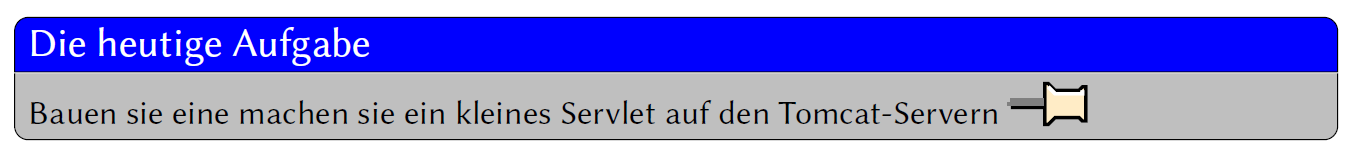
Primäres Ziel von Virtualisierung ist typischerweise die Kostenersparnis.

### Auswirkungen der Virtualisierung auf Applikationen



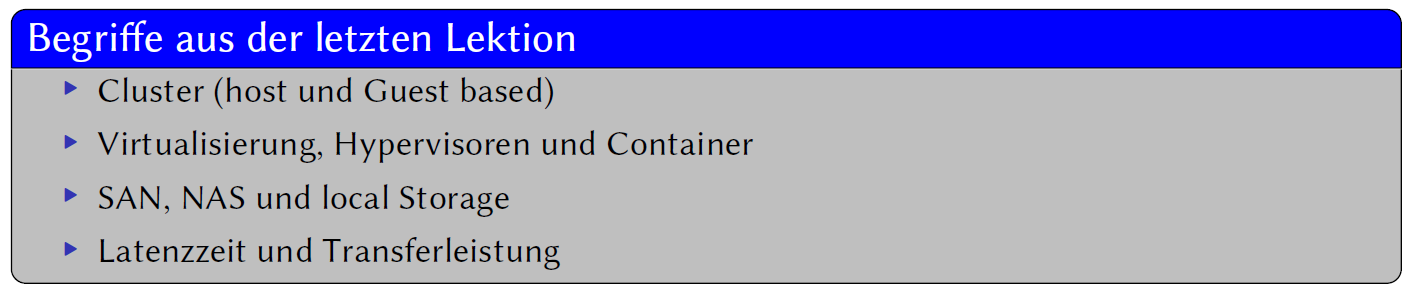
* Ressourcen sind unterschiedlich verfügbar.
* Die Latenzzeit und Transferleistung von Ressourcen kann abhängig von der Auslastung des Hosts oder von der Verfügbarkeit einzelner Komponenten abhängen

## Aufgaben und Übungen



# Woche 3

## Repetition

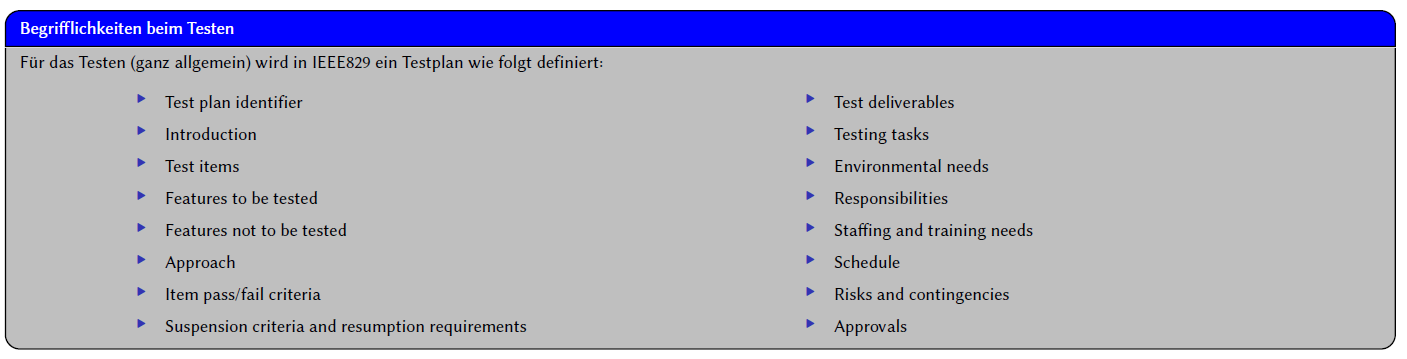


## Spass mit Netzwerken



Beim “Performance Testing” kommt es vor allem auf eines an: Parallelität. Ein Oktopus wäre mit seinen 3 Herzen und unabhängig agierenden Armen tatsächlich prädestiniert für einen solchen Job.

## Reproduzierbares Testen



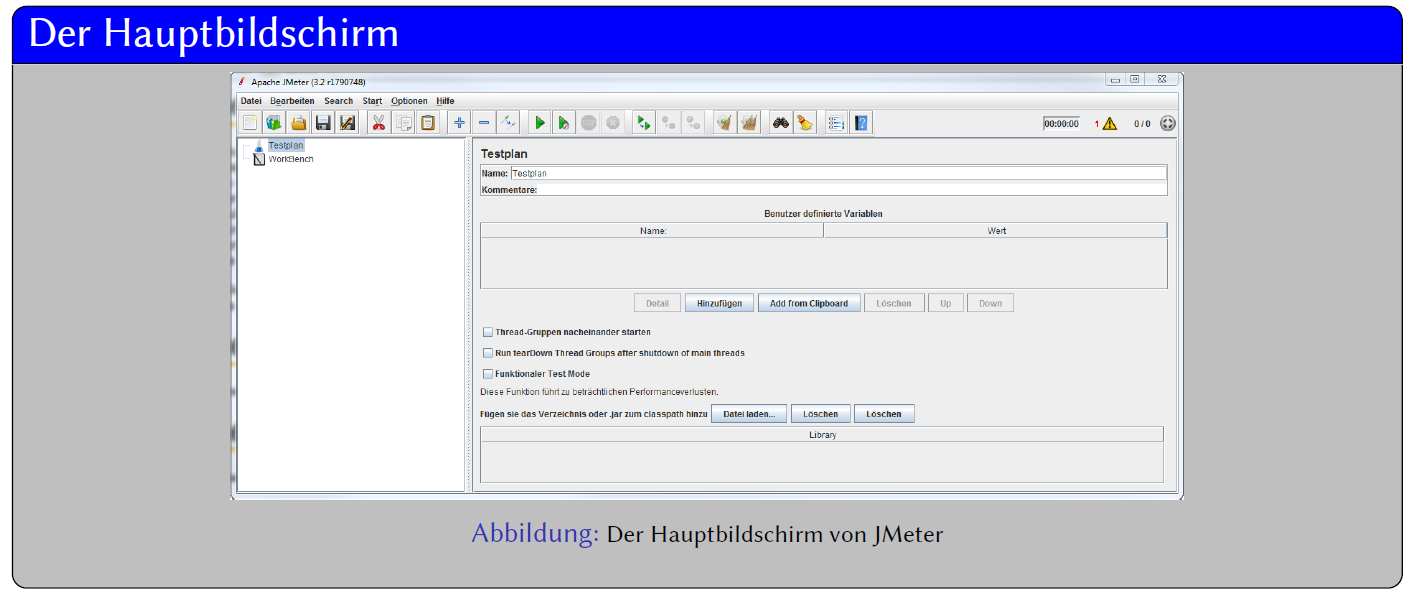
Für das Testen (ganz allgemein) wird in IEEE829 ein Testplan wie folgt definiert:

* Test plan identifier
* Introduction
* Test items
* Features to be tested
* Features not to be tested
* Approach
* Item pass/fail criteria
* Suspension criteria and resumption requirements
* Test deliverables
* Testing tasks
* Environmental needs
* Responsibilities
* Staffing and training needs
* Schedule
* Risks and contingencies
* Approvals

Meistens werden sie aufgeteilt in einen Master Test Plan (MTP) und mehrere Level Test Plans (LTP). Das ist der detaillierte Plan, wie der Test durchzuführen ist. Die Testresultate werden festgehalten in einem Level Test Log (LTL). Möglicherweise werden anschliessend ein Level Test Report (LTR) und/oder ein Master Test Report (MTR) geschrieben aber diese werden meist einfach in einem Testjournal zusammengefasst. Wichtig ist, dass Evidenz für das Ausführen der Tests präsentiert wird und dass genau festgehalten wird, ob und wie die Tests nach Testplan ausgeführt werden konnten und was das Resultat war.

## Testen mit Apache JMeter

### Der Hauptbildschirm

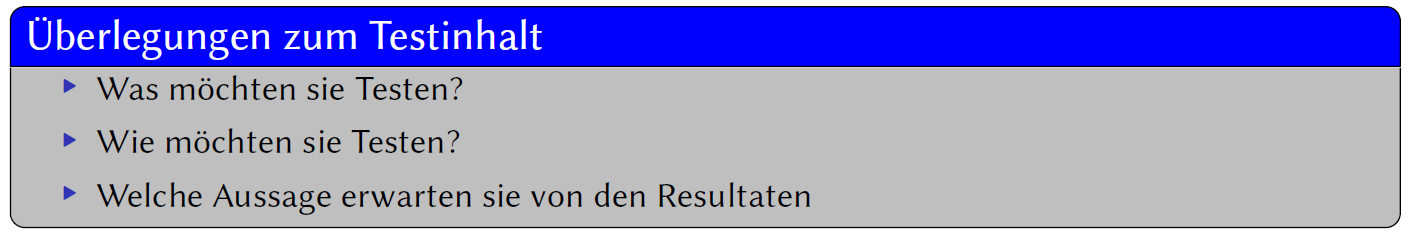


JMeter teilt sich in zwei Bereiche ein:

* Einen Test Plan: In ihm wird der Testvorgang festgehalten.
* Den Workbench: In ihm werden die Testresultate festgehalten

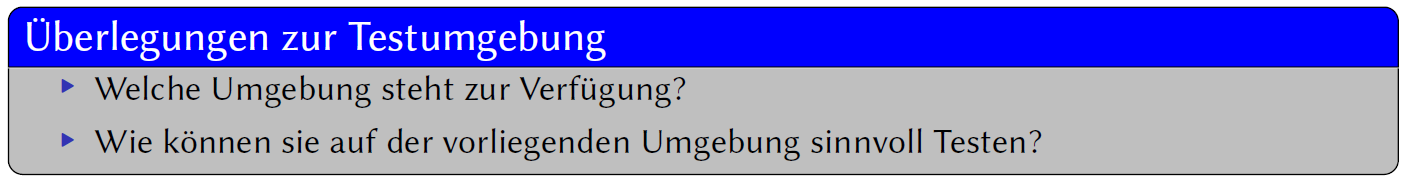
Typicherweise haben sie zuerst eine ThreadGroup im Testplan. Für das HTTP-Testing Empfiehlt sich als Elemente diverse Timer und HTTP-Requests oder einen Recording Controller. Damit der Recording- Controller funktioniert muss ein lokaler Browser darauf konfiguriert werden, dass er den Lokalen Proxy auf localhost:8888 verwendet.

### Überlegungen zum Testinhalt



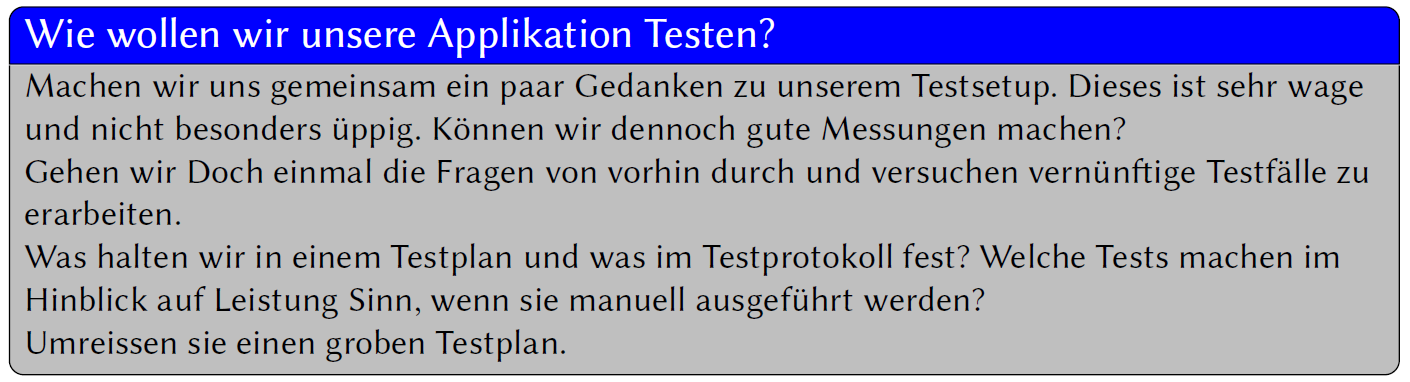
* Was möchten sie Testen? In unserem Fall zielen wir nicht auf Funktionen, sondern auf Leistung ab. Welche Daten sind aussagekräftig.
* Wie möchten sie Testen? Welche Vorbedingungen brauchen sie? Sind am Anfang einer Messung Cache- oder sonstige Effekte zu beobachten? Sind diese für sie interessant oder nicht?
* Welche Aussage erwarten sie von den Resultaten? Welche Leistungsdaten möchten sie ausgewiesen haben? Latenzzeiten? Maximale Benutzerzahl? Anzahl der Anfragen pro Sekunde?

### Überlegungen zur Testumgebung



* Welche Umgebung steht zur Verfügung? Ist die Testumgebung überhaupt geeignet um Tests durchzuführen? Wie müsste eine ideale Umgebung aussehen und welche Falschaussagen der Tests erwarten sie aufgrund des Realen Setups?
* Wie können sie auf der vorliegenden Umgebung sinnvoll Testen? Wie müssen sie Testfälle und Kennzahlen wählen, damit sinnvolle Resultate erzielt werden.

## Eine Klassenübung



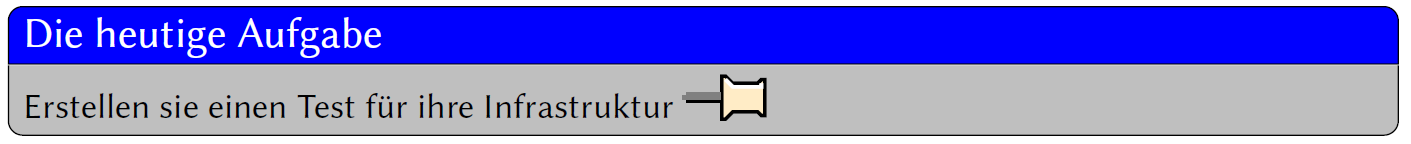
Machen wir uns gemeinsam ein paar Gedanken zu unserem Testsetup. Dieses ist sehr wage und nicht besonders üppig. Können wir dennoch gute Messungen machen?

Gehen wir Doch einmal die Fragen von vorhin durch und versuchen vernünftige Testfälle zu erarbeiten.

Was halten wir in einem Testplan und was im Testprotokoll fest? Welche Tests machen im Hinblick auf Leistung Sinn, wenn sie manuell ausgeführt werden?

Umreissen sie einen groben Testplan.

## Aufgaben und Übungen



# Woche 4