Hochverfügbarkeit

Hackenberger Christoph, Malik Patrick – 5BHIT

Inhaltsverzeichnis

[1. Hochverfügbarkeit – Allgemein 4](#_Toc447031618)

[2. Gekoppelte Systeme 5](#_Toc447031619)

[2.1. Serielle Kopplung 5](#_Toc447031620)

[2.2. Parallele Kopplung 5](#_Toc447031621)

[3. Hochverfügbare Architekturen 7](#_Toc447031622)

[3.1. Cold Standby 7](#_Toc447031623)

[3.2. Hot Standby 7](#_Toc447031624)

[3.3. Cluster 7](#_Toc447031625)

[3.3.1. Load balanced Cluster 7](#_Toc447031626)

[3.3.2. Failover Cluster 8](#_Toc447031627)

[4. Chaos Engineering 8](#_Toc447031628)

[4.1. Deterministisches Chaos 8](#_Toc447031629)

[4.2. Chaos Engineering in Verteilten Systemen 9](#_Toc447031630)

[4.3. Erweitere Prinzipien von Chaos Engineering 9](#_Toc447031631)

[4.3.1. Aufstellen einer Hypothese um das „Steady State“-Verhalten 9](#_Toc447031632)

[4.3.2. Verschiedene Ereignisse verwenden 9](#_Toc447031633)

[4.3.3. Experimente in Produktionsumgebung durchführen 9](#_Toc447031634)

[4.3.4. Automatisierung der Experimente 10](#_Toc447031635)

[4.4. Chaos Engineering bei Netflix 10](#_Toc447031636)

[4.4.1. Chaos Monkey 10](#_Toc447031637)

[4.4.2. Chaos Gorilla 10](#_Toc447031638)

[4.4.3. Chaos Kong 10](#_Toc447031639)

[5. Hochverfügbarkeit bei Oracle 12c 11](#_Toc447031640)

[5.1. Oracle-DB Allgemeines 11](#_Toc447031641)

[5.1.1. Physical Database Structures 11](#_Toc447031642)

[5.1.2. Speicherstrukturen 12](#_Toc447031643)

[5.1.3. OCW 12](#_Toc447031644)

[5.2. RAC 12](#_Toc447031645)

[5.2.1. Installation 13](#_Toc447031646)

[5.2.2. Planned Downtime 13](#_Toc447031647)

[5.3. Data Guard 13](#_Toc447031648)

[5.3.1. Architektur 13](#_Toc447031649)

[5.3.2. Physical Standby 13](#_Toc447031650)

[5.3.3. Logical Standby 14](#_Toc447031651)

[6. Amazon Web Services (AWS) 14](#_Toc447031652)

[6.1. Zusammenfassung 14](#_Toc447031653)

[6.1.1. Was ist AWS 14](#_Toc447031654)

[6.1.2. Wann sollte man AWS verwenden? 14](#_Toc447031655)

[6.1.3. Grundlegendes 15](#_Toc447031656)

[6.2. Sicherheit 16](#_Toc447031657)

[6.2.1. Virtual Private Cloud 16](#_Toc447031658)

[6.2.2. Amazon Inspector 18](#_Toc447031659)

[6.2.3. Amazon WAF 18](#_Toc447031660)

[6.3. High Availability 18](#_Toc447031661)

[6.3.1. Auto Scaling 18](#_Toc447031662)

[7. Couchbase 19](#_Toc447031663)

[7.1. High Availability bei Couchbase 19](#_Toc447031664)

[7.1.1. Rack Zone Awareness 19](#_Toc447031665)

[7.1.2. XDCR 19](#_Toc447031666)

[Literaturverzeichnis 19](#_Toc447031667)

[Abbildungsverzeichnis 22](#_Toc447031668)

# Hochverfügbarkeit – Allgemein

Hochverfügbarkeit beschreibt im allgemeinen die Fähigkeit von Systemen, trotz auftretender Fehler z.B. Ausfall eines Servers, den Betrieb mit einer großen Wahrscheinlichkeit, meist 99,99% und mehr, zu ermöglichen. [1]

Zuerst muss allerdings geklärt werden welche Ausfallzeiten die angeben Verfügbarkeitswahrscheinlichkeiten eigentlich in Realität bedeuten. Nimmt man z.B. eine Verfügbarkeit von 99,5% pro Jahr, hört sich das auf den ersten Blick womöglich nach einem guten Wert an. Berechnet man nun allerdings die tatsächliche Zeit in der das System nicht verfügbar ist, kommt man auf knapp 44 Stunden pro Jahr. Unser System wäre demnach also pro Jahr fast 2 Tage nicht betriebsfähig. [1]

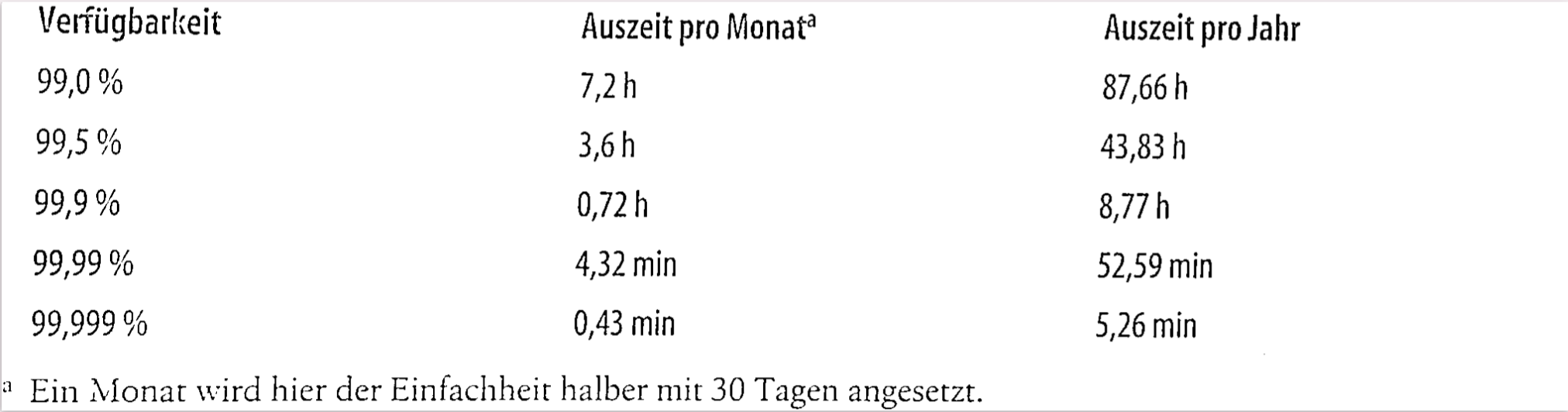


Abbildung 1: Verfügbarkeit und Auszeit pro Monat bzw. Jahr [1]

Die Zeit, welche benötigt wird, um einen aufgetretene Störung wieder zu beheben wird Mean Time to Repair (MTTR) genannt und wird im Normalfall in der Servicevereinbarung (engl. Service Level Agreement, SLA) festgelegt. Die MTTR ist indirekt proportional zu den Kosten, welche für den Betrieb der Services investiert werden müssen. [1]

Eine weitere wichtige Größe im Bereich der Hochverfügbarkeit ist die sogenannte Mean Time Between Failures (MTBF). Diese gibt die mittlere Zeit zwischen zwei Ausfällen des Systems an. Üblicherweise wird die MTBF beim Kauf von Hardwarekomponenten angeben. [1]

Aus den beiden oben angeführten Größen lässt sich nun die Verfügbarkeit berechnen:

A = [1]

Aus dieser Verfügbarkeit lässt sich dann die Auszeit berechnen:

Auszeit = (1 – A) \* Basiszeitraum [1]

Basiszeitraum ist die vereinbarte Zeit, meist ein Jahr oder ein Monat.

Zusätzlich zur Berechnung der Verfügbarkeit und Ausfallzeiten gibt es eine Klassifizierung durch die Harvard Research Group (HRG), die sogenannte Availability Environment Classification (AEC) ist eine textuelle Einteilung in sechs verschiedene AEC-Verfügbarkeitsklassen:

|  |  |  |
| --- | --- | --- |
| Klasse | Bezeichnung | Beschreibung |
| AEC-0 | Conventional | Funktion kann unterbrochen werden, Datenintegrität ist nicht essentiell |
| AEC-1 | Highly Reliable | Funktion kann unterbrochen werden, Datenintegrität muss jedoch gewährleistet sein |
| AEC-2 | High Availability | Funktion darf nur innerhalb festgelegter Zeiten oder zur Hauptbetriebszeit minimal unterbrochen werden |
| AEC-3 | Fault Resilient | Funktion muss innerhalb festgelegter Zeiten oder während der Hauptbetriebszeit ununterbrochen aufrechterhalten werden |
| AEC-4 | Fault Tolerant | Funktion muss ununterbrochen aufrechterhalten werden, 24/7-Betrieb (24 Stunden, 7 Tage die Woche) muss gewährleistet sein |
| AEC-5 | Disaster Tolerant | Funktion muss unter allen Umständen verfügbar sein |

[2]

# Gekoppelte Systeme

Die Funktionsfähigkeit eines Systems ist Abhängig vom Funktionieren vieler einzelner Komponenten innerhalb des Systems. Um z.B. eine Applikation auf einem Server zu betreiben müssen die Stromversorgung, die Hardware (Prozessoren, RAM) und die Software (Betriebssystem, Applikation) funktionieren. Wie kann man also die Verfügbarkeit von solchen komplexen Systemen berechnen? Dazu muss man wissen, das die Komponenten jedes noch so komplexen Systems nur auf zwei Arten verschaltet werden können. [1]

## Serielle Kopplung

Betrachten wir ein einfaches System welches aus zwei Komponenten besteht.

Damit das Gesamtsystem funktioniert müssen beide Komponenten korrekt arbeiten. Die Verfügbarkeit Aser berechnet sich wie folgt:

Aser = A1 \* A2 [1]

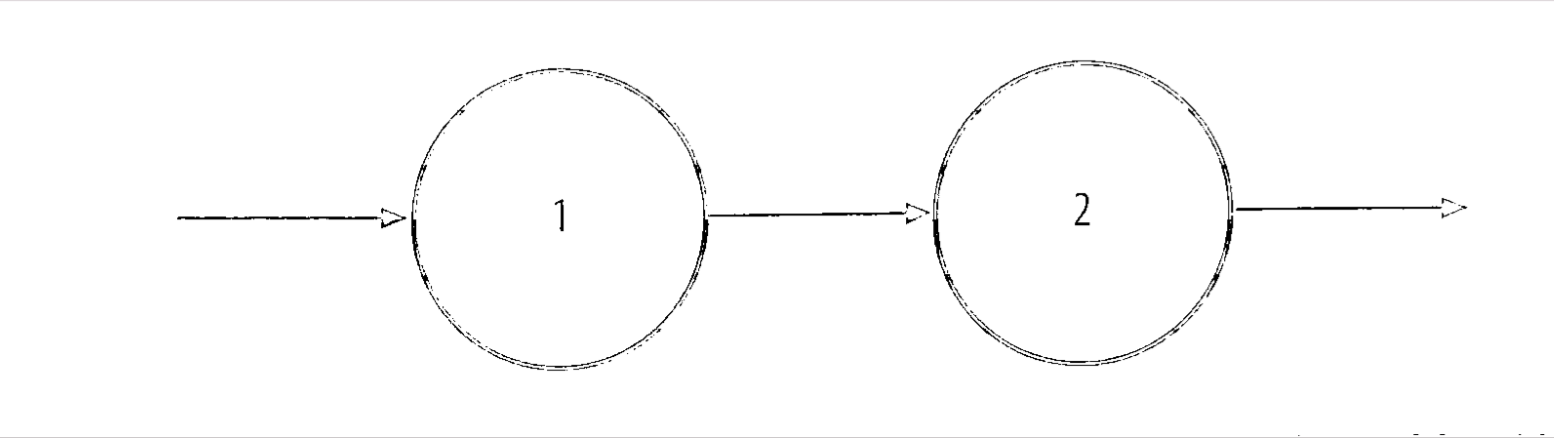


Abbildung 2: Serielle Kopplung zweier Komponenten [1]

Die Verfügbarkeit des Gesamtsystems ist also immer kleiner als die der einzelnen Komponenten. Will man die Verfügbarkeit des Systems verbessern, so muss immer zuerst die Verfügbarkeit der schlechtesten Komponente verbessert werden, da diese die größte Auswirkung auf die Gesamtverfügbarkeit hat. [1]

## Parallele Kopplung

Die andere Möglichkeit Komponenten zu verschalten ist die parallele Kopplung. Hier erfüllen zwei Komponenten den selben Dienst. Es wird also Redundanz geschaffen. [1]

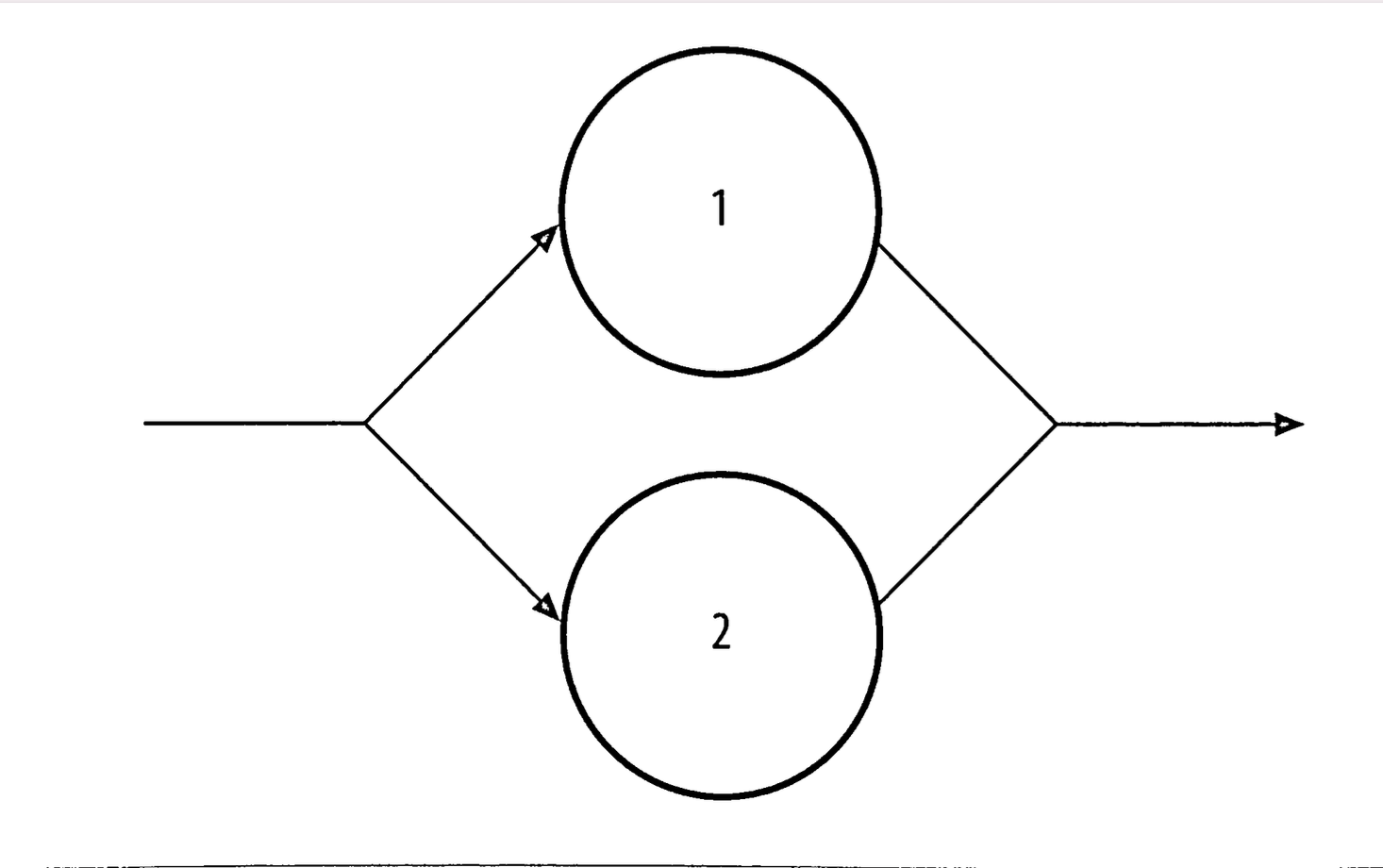


Abbildung 3: Parallele Kopplung zweier Komponenten [1]

Das Gesamtsystem besteht in diesem Fall auch wieder aus zwei Komponenten. Das Funktionieren des Systems ist allerdings nicht von beiden Komponenten abhängig, es muss nur einer der beiden Komponenten funktionieren. [1]

Die Verfügbarkeit Apar lässt sich wie folgt berechnen:

Apar = 1 – (1 – A1) \* (1 – A2) [1]

Die Verfügbarkeit des Gesamtsystems ist also immer größer als die Verfügbarkeit der einzelnen Komponenten.

Ein Beispiel mit jeweils einem anderen Grad von Redundanz bei jeweils gleicher Verfügbarkeit der einzelnen Komponenten von 99% ist in Abbildung 4 zusehen.



Abbildung 4: Verfügbarkeit bei redundanter Auslegung mit identischen Komponenten [1]

Die Grundlage aller hochverfügbaren Systemen ist daher die sogenannte 3R-Regel:

„Redundanz, Redundanz und noch einmal Redundanz“ [1]

Durch den Einsatz von redundanten Komponenten sollen sogenannte Singe Points of Failure (SPoF) verhindert werden. Ein Single Point of Failure ist daher eine Komponente von der die Funktion des gesamten Systems abhängt. [1]

# Hochverfügbare Architekturen

## Cold Standby

Bei Cold-Standby wird neben den eigentlichen Produktivsystem ein zweites identisches System aufgebaut. Dieses ist allerdings solange das Produktivsystem funktioniert nicht im Einsatz und bleibt deaktiviert. Sollte das Produktivsystem ausfallen, muss das Ersatzsystem manuell hochgefahren werden. [3]

Vorteile [3]:

* Keine Komplexitätserhöhung
* Geringe Kosten

Nachteile [3]:

* Ersatzsystem muss ständig aktuell gehalten werden
* Ersatzsystem muss manuell aktiviert werden
* Daten müssen ggf. migriert werden

## Hot Standby

Beim Hot Standby steht ebenfalls neben dem Produktivsystem ein Ersatzsystem zur Verfügung. Das Ersatzsystem wird allerdings neben dem Produktivsystem parallel betrieben. Beim Ausfall des Produktivsystems wird das Ersatzsystem aktiviert. Der Wechsel auf das Ersatzsystem kann ggf. auch automatisch stattfinden, dafür sind allerdings zusätzliche Funktionen, wie z.B. die automatische Erkennung von Ausfällen, notwendig. [3]

Vorteile:

* Ausfallzeiten geringer als bei Cold-Standby
* Relativ Kostengünstig
* Ersatzsystem in Betrieb, daher Datenreplikation möglich

Nachteile:

* Ersatzsystem muss ständig aktuell gehalten werden
* Ggf. manuelle Aktivierung notwendig

## Cluster

Durch die Gruppierung von mehreren Rechnern zu einem Cluster, welche parallel arbeiten, führt zu einer Steigerung der Verfügbarkeit sowie der Performance. Die Applikation kann dabei aktiv auf einem Rechner oder parallel verteilt auf mehre Rechner ausgeführt werden. [3]

Man unterscheidet dabei je nach Funktionsart zwischen Load balanced Cluster und Failover Cluster.

### Load balanced Cluster

Beim Load balanced Cluster wird die Applikation bzw. die Instanzen einer Applikation auf mehrere Server verteilt. Die Verteilung basiert meist auf der Auslastung der einzelnen Server. Durch diese Lastverteilung wird eine Performance-Steigerung erzielt. Außerdem kann durch die geschaffene Redundanz ein Ausfall eines einzelnen Servers kompensiert werden. [3]

Vorteile [3]:

* Steigerung der Verfügbarkeit und Performance
* Alle Ressourcen werden dauerhaft genutzt
* Skalierbar
* Komplexität geringer als Failover Cluster

Nachteile [3]:

* Nicht für alle Applikation einsetzbar

### Failover Cluster

Bei einem Failover-Cluster, wird beim Ausfall eines der Cluster-Systeme, ein automatischer Wechsel des Betriebes auf einen anderen Teil des Clusters durchgeführt. Diese automatische Übernahme wird Failover genannt. Für diese Failover-Funktionalität, wird eine sogenannte „heartbeat“ Verbindung zwischen den Cluster-Servern benötigt, welche für die Kommunikation innerhalb des Clusters verwendet wird. [3]

Vorteile [3]:

* Erhebliche Steigerung der Verfügbarkeit
* Voll Automatisiert

Nachteile [3]:

* Hoch komplex
* Nicht gut skalierbar
* Nur teile der Ressourcen werden genutzt
* Hohe Kosten

# Chaos Engineering

## Deterministisches Chaos

Deterministisches Chaos beschreibt das unvorhersehbare Verhalten von Systemen bei bekannten, vorherbestimmten Abläufen. Durch eine kleine Änderung bzw. Störung der Anfangsbedingungen, das Endergebnis über den Lauf der Zeit stark verändert werden kann. Die Veränderung des Ergebnisses wird allerdings immer innerhalb eines Wertebereichs bleiben. Die Kombination dieser beiden Aspekte führt zu einem unregelmäßigen und unvorhersehbaren Ergebnis. [4]

Chaos Engineering gilt als Allgemeine Studie über den Einsatz von deterministischen Chaos in sowohl theoretischen bzw. mathematischen oder auch technologischen Bereichen. So erstreckt sich das Einsatzgebiet von Chaos Engineering von der Bewegung der Spülkopfs im Geschirrspüler bis hin zur Darstellung von Neuronen in einem Neuronalem Netzwerk. [5]

Im Bereich des Chaos Engineerings wird zwischen zwei Arten unterschieden

* „Synthetic Chaos“ – Hier wird künstlich erzeugtes Chaos verwendet, um aus dem chaotischem Verhalten neue Lösungsmöglichkeiten zu finden. Beispiel: das Verlassen eines lokalen Minimums um besser Optimierungslösungen zu finden. [5]
* „Analytic Chaos“ – Hier werden natürliche oder künstliche komplexe Systeme analysiert, um auf deterministische Regeln zu schließen. Beispiel: Um zukünftig notwendige Wartungsarbeiten an Maschinen vorauszusagen. [5]

## Chaos Engineering in Verteilten Systemen

Im Bereich von Verteilten Systemen befasst sich Chaos Engineering mit mit dem Testen von Systemkapazitäten während ungewöhnlichen Situationen. Denn selbst wenn alle Systeme einzelnen funktionieren, kann die Interaktion der verschiedenen Systeme zu unvorhersehbaren Ergebnissen führen. Ein weiteres Problem sind sogenannte störende reale Ereignisse, wie z.B. ein Hardwareausfall. [6]

Das Ziel von Chaos Engineering ist schwächen im System zu finden, damit diese verbessert werden können, bevor sie zu einem systemweiten fehlerhaften Verhalten führen. Um dieses Ziel zu erreichen wird während eines kontrollierten Experiments, das Verhalten des Systems überwacht um so potentielle Schwachstellen zu erkennen. [6]

Die Umsetzung eines solchen Experiments erfolgt in vier Schritten [6]:

1. Man definiert einen sogenannten „Steady State“, also ein messbares Ergebnis eines Systems welcher als normales Verhalten betrachtet wird.
2. Man nimmt nun an, dass dieser „Steady State“ sowohl in der Experimentalgruppe sowie der Kontrollgruppe beibehalten wird.
3. Nun wird in die Experimentalgruppe Chaos gebracht, sprich man simuliert z.B. einen Ausfall eines Servers.
4. Nun vergleicht man das Ergebnis der Experimentalgruppe mit jenem der Kontrollgruppe und versucht Unterschiede zwischen dem „Steady State“ zu finden.

Je schwieriger es ist, den „Steady State“ zu stören, desto sicherer kann man sich auf das korrekte Verhalten des Systems verlassen.

## Erweitere Prinzipien von Chaos Engineering

### Aufstellen einer Hypothese um das „Steady State“-Verhalten

Beim festlegen des „Steady State“ sollte man sich auf messbare Ergebnisse des Gesamtsystems stützen, anstatt nur auf interne Attribute. Dafür würde sich z.B. die Fehlerrate des Gesamtsystems oder die Latenz eignen. Durch das kontrollieren von systematischen Verhaltensmustern, erhält man eine Kontrolle, dass das System funktioniert anstatt wie es funktioniert. [6]

### Verschiedene Ereignisse verwenden

Das Chaos in den Experimenten entspricht realen Ereignissen die während der Laufzeit unseres Systems eintreten können. Deshalb sollten die simulierten Ereignisse nach deren Auftrittshäufigkeit und deren Auswirkung auf das System priorisiert werden. Dabei sollten allerdings die Typen der Ereignisse gemischt werden, so können z.B. Hardwarefehler, Softwarefehler, Traffic Spitzen, ... als störendes Ereignis verwendet werden. [6]

### Experimente in Produktionsumgebung durchführen

Verteilte Systeme verhalten sich je nach Umgebung und derzeitiger Auslastung anders. Daher sollten beim Chaos Engineering die Experimente immer auf den Produktivsystemen mit echtem Traffic durchgeführt werden, umso zu das echte Verhalten des Systems analysieren zu können. [6]

### Automatisierung der Experimente

Die Durchführung der Experimente sollte automatisiert und kontinuierlich erfolgen. Chaos Engineering setzt sich außerdem als Ziel sowohl die Orchestration, also die Kombination verschiedener Dienste, sowie die Analyse der Experimente zu automatisieren. [6]

## Chaos Engineering bei Netflix

Netflix ist ein Vorreiter im Bereich Chaos Engineering bei Verteilten Systemen. Auch die vorher beschriebenen Prinzipien zu Chaos Engineering wurden von Netflix in einem offenen Dokument [6], zur Bearbeitung von Anderen, definiert.

Netflix betreibt ihr System mithilfe von Amazon Web Services (siehe Kapitel 6) und hat um dieses zu testen und zu überwachen einige Tools entwickelt und dieser unter dem Namen SimianArmy [7] veröffentlicht. [8]

Zu den drei wichtigsten Tools zählen:

### Chaos Monkey

Chaos Monkey bietet verschiedene Strategien um eine EC2 Instanz innerhalb einer Auto Scaling Group „anzugreifen“. Unter anderem bietet Chaos Monkey die Möglichkeit Instanzen zu terminieren, den kompletten Netzwerk-Traffic zu blockieren, zufällig Prozesse auf einer Instanz zu beenden, ... [8]

Netflix bietet zum Einsatz von Chaos Monkey auch einen Quick Start Guide [9] an.

### Chaos Gorilla

Chaos Gorilla funktioniert ähnlich wie Chaos Monkey, allerdings simuliert Chaos Gorilla eines Ausfall einer ganzen Availability Zone. Dadurch möchte Netflix überprüfen ob ihr System automatisch und ohne, dass es ihre Kunden mitbekommen die anderen verbleibenden Availability Zones den Service übernehmen können. [10]

### Chaos Kong

Und schließlich der sozusagen der Worst-Case, nämlich der Ausfall einer kompletten Region von Amazon Web Services, welcher mithilfe des Tools Chaos Kong simuliert wird. Der große Unterschied zu Chaos Monkey und Gorilla liegt hierbei, dass der Ausfall der Region hier wirklich nur simuliert und nicht wirklich, wie bei den beiden anderen Tools, Instanzen terminiert werden. [11] Ein Verlauf eines solchen Experiments mittels Chaos Kong ist in Abbildung 5 zu sehen.

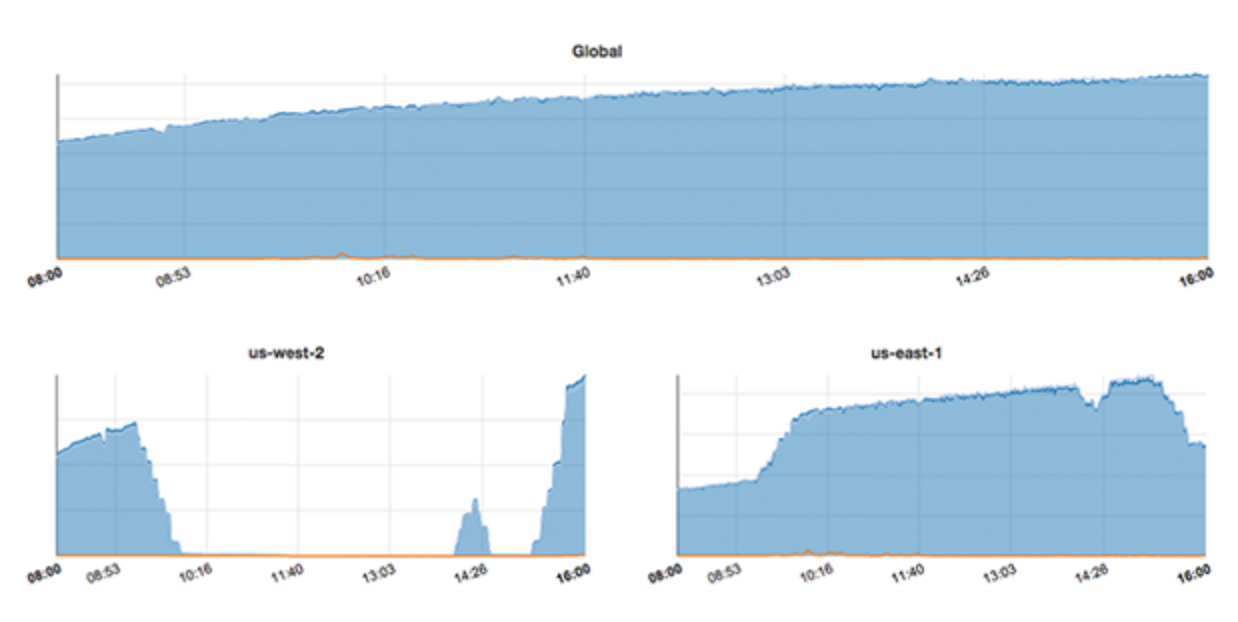


Abbildung 5: Verlauf eines Chaos Kong Experiments [11]

# Hochverfügbarkeit bei Oracle 12c

## Oracle-DB Allgemeines

In diesem Teil wird das notwendige Hintergrundwissen für die kommenden Erklärungen bereitgestellt. Alle notwendigen Begriffe bzw. Funktionen und Strukturen die für das Verständnis essentiell sind, werden hier kurz erklärt.

### Physical Database Structures

Hier werden lediglich die 3, für das Folgende benötigten, Grundstrukturen erklärt.

#### Datafiles

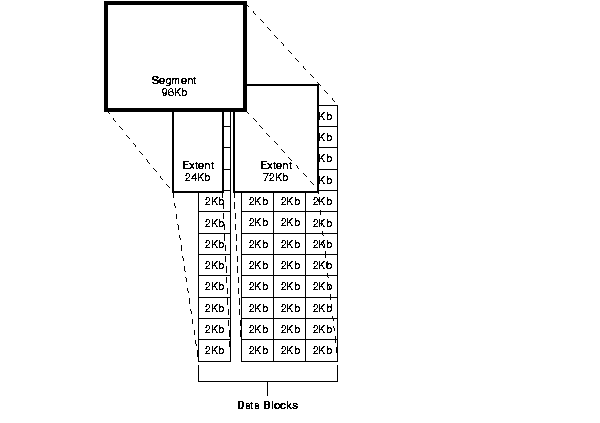
Hierbei handelt es sich um die physikalische Abbildung von Daten. Jede Datenbank besteht aus einem oder mehreren Tablespaces, wobei jeder Tablespace aus einem oder mehreren Datafiles besteht. Ein Datafile kann jedoch lediglich zu einer Datenbank und einem Tablespace gehören. [13]

Abbildung 6: Data Blocks, Extents und Segments [13]

#### Controlfiles

Jede Datenbank besitzt ein Controlfile, welches für die Funktionalität dieser verantwortlich ist. Ist das Controlfile beschädigt oder nicht vorhanden, kann die Datenbank nicht ordnungsgemäß verwendet werden. Ein solches Controlfile beinhaltet Daten wie den Datenbanknamen und weitere Informationen über die Datenbank bzw. die zugehörigen Tablespaces. [13]

#### Redo Log Files

Jede Datenbank hat zumindest ein Redo Log File (sie können repliziert und auf verschiedene Platten gelegt werden), in welchem alle Änderungen der Datenbank festgehalten werden. Diese Redo Files werden zur Wiederherstellung nach Problemen verwendet und unter Anderem auch bei der Wiederherstellung mittels Data Guard. [14]

### Speicherstrukturen

#### Data Blocks

Ein Datablock ist die kleinste Einheit in der Oracle Daten der Datenbank speichert. Jeder Block hat eine bestimmte Größe, die für gewöhnlich ein mehrfaches der der Blockgröße des Betriebssystems ist. [15]

#### Extents

Ein Extent besteht aus mehreren Data Blocks und hat keine festgelegte Größe. Wenn Extents alloziiert werden, wird in den zum Tablespace gehörigen Datafiles nach dem verlangten Platz gesucht, ist der Platz nicht vorhanden, wird ein weiteres Datafile überprüft. [15]

#### Segments

Es gibt 3 Arten von Segmenten: Data Segments, Index Segments und Temporary Segments. Ein Segment besteht aus Extents und sobald ein Segment voll ist, werden neue Extents alloziiert.

Ein Data Segment beinhaltet alle Daten eines nicht partitionierten oder geclusterten Tables, einer Partition eines partiionierten Tables oder einem Cluster von Tables. Wenn eine Tabelle oder ein Cluster erstellt wird, wird unter Anderem das zugehörige Data Segment erstellt.

Ein Index Segment beinhaltet alle Daten zu einem nicht partitionierten Index. Ansonsten besitzt jede Partition ein eigenes Index Segment. Dieses Segment wird erstellt, wenn „CREATE INDEX“ aufgerufen wird.

Temporary Segments werden von Oracle verwendet, wenn SQL-Statements zusätzlichen „Arbeitsplatz“ benötigen (bspw. beim Sortieren). Wenn die Sortierung über Indizes oder in memory durchgeführt werden kann, wird dieses Segment nicht erstellt.  
Befehle wie „CREATE INDEX“ oder „SELECT DISTINCT …“ benötigen diesen zusätzlichen Workspace auch. [15]

#### Tablespaces

Bei einem Tablespace handelt es sich um eine logische Einheit welche Bestandteil einer Datenbank ist. Jede Datenbank hat zumindest 2 Tablespaces, namentlich SYSTEM und SYSAUX. Tablespaces haben ein oder mehrere Datafiles. Zudem ist es möglich einen Tablespace als „online“ oder „offline“ zu markieren (std.: online), auf als offline markierte Tablespaces ist es nicht möglich zuzugreifen. [14]

### OCW

Oracle Clusterware ist für die Verwaltung der Nodes verantwortlich, sie macht aus einzelnen Servern Teilnehmer eines Clusters und wurde mit Oracle 10g eingeführt. Neben der Verwendung in RAC kann OCW auch für andere Applikationen verwendet werden, welche einen Cluster benötigen. [12]

## RAC

Bei Real Application Clusters von Oracle, handelt es sich um eine Option des Oracle DBMS. RAC dient dazu, ein Datenbanksystem auf einem Cluster zu verteilen und dadurch hohe Verfügbarkeit sowie Skalierbarkeit zu erreichen. Die Daten in einem RAC-Cluster sind so lange verfügbar, so lange mindestens 1 Node dieses Clusters verfügbar ist. [16]

### Installation

Um RAC auf einer Maschine verwenden zu können, muss Oracle Clusterware installiert sein.  
Die Installation ist insofern vereinfach, dass Oracle DB erkennt wenn es auf einem Cluster installiert wird und dadurch die RAC-Option mitinstalliert, nicht nur auf die eigene Maschine sondern auf allen Knoten des Clusters.

Zudem besteht die Option zu Klonen. Diese wird empfohlen wenn der Cluster schnell um einige gleiche (gleich in Hard- und Software) Nodes erweitert werden soll.[17]

### Planned Downtime

RAC bietet sich auch für planned downtime-Szenarios an. Ist bspw. ein Update notwendig, ist es kein Problem eine der verwendeten Nodes aus dem Cluster zu nehmen, upzugraden und nach Beendigung wieder online zu stellen. [18]

## Data Guard

Beim Oracle Data Guard handelt es sich um ein System zur Disaster-Recovery. Neben der Problembehandlung kann Data Guard auch für die Wartung von Datenbanken verwendet werden. Es kann problemlos auf eine der Stanbydatenbanken gewechselt werden, um auf dem Produktivsystem bspw. Updates zu installieren.

### Architektur

Es können bis zu 9 Standby-Datenbanken (n physical + m logical = 9) verwaltet werden, welche eine real-time-Kopie der Produktivdatenbank sind. Bei Fehlern im Produktivsystem kann auf eine Standby-DB gewechselt werden. Die Datenbanken können sowohl über das LAN als auch das WAN verbunden sein. [16]

### Physical Standby

Bei dieser Art von Standby, müssen alle physikalischen Datenbanken der Produktivdatenbank übernommen werden. Physical Standbys müssen auf einer „Block-for-Block“ Basis ident mit dem Produktivsystem sein, da die Wiederherstellung „Block-for-Block“ mittels ROWID ausgeführt wird.

Jede Row in Oracle DB besitzt eine ROWID. Bei löschen oder neuinfügen einer row, ist jedoch nicht gesagt, dass diese row die gleiche ID erneut bekommt, daher sollte die ROWID weder als Primary Key verwendet werden, noch kann sie verlässlich bei nicht identen Datenbanken zur Wiederherstellung verwendet werden. (Deshalb ist es bei einem Physical Standby notwendig eine exakte Kopie des Produktivsystems zu haben und nicht möglich bei einem Logical Standby-System die Wiederherstellung mittels der ROWID durchzuführen) [41]

Für die Wiederherstellung werden die Daten aus dem Redo-File der Produktivdatenbank verwendet.  
Eine Physical Standby-DB kann entweder die Konsistenz erhalten (Kopieren der Daten des Produktivsystems) oder Read-only geöffnet werden, wobei die Daten des Redo-Logs des Produktivsystems weiterhin an die Datenbank geliefert und appliziert werden. Sie kann nicht beschrieben werden, da die ROWIDs geändert werden würden und somit die weitere Wiederherstellung nach Öffnen der DB unmöglich wäre. Sie können unter anderem verwendet werden um dem Produktivsystem Abfragen abzunehmen.[19] [16]

### Logical Standby

Hierbei handelt es sich um eine Stanby-Datenbank die lediglich logisch konsistent sein muss. Aus dem Redo-Log des Produktivsystems werden SQL-Befehle generiert, welche auf der Standbymaschine ausgeführt werden. Die physikalischen Eigenschaften dieser DB müssen nicht mit denen des Produktivsystems übereinstimmen!

Ein Logical Standby-System kann auch beschrieben werden, die Tabellen die jedoch durch die Konsistenzerhaltung betroffen sind, können lediglich gelesen werden während sie upgedated werden. Zudem gibt es Restriktionen bezüglich Datentypen, Tabellentypen, DDL- und DML-Befehlen. (genauer hier: [42]) Ein weiterer Vorteil liegt darin, dass Blockfehler welche ein Physical Standby kopieren würde, die durch HW-Fehler entstanden sind, nicht gemacht werden.  
Außerdem kann direkt auf das Standby-System als Primärsystem gewechselt werden, wodurch die Downtime durch Updates oder Patches enorm verringert wird. [19] [16]

Vergleich der Vorteile der Stanby-Optionen:

|  |  |
| --- | --- |
| **Physical** | **Logical** |
| Keine Einschränkungen bzgl Datentypen, usw.. | Auch beschreibbar |
| Sehr Performant | Vor Blockfehlern durch HW-Fehler geschützt |
| Lastverteilung | Keine Downtime durch Updates |
|  | Lastverteilung |

# Amazon Web Services (AWS)

## Zusammenfassung

### Was ist AWS

Amazon Web Services ist eine Cloud-Computing Plattform.   
Mit der „Elastic Compute Cloud“ (EC2) hat man die Möglichkeit die benötigte Rechenkapazität in Minuten zu ändern. Da diese Veränderung mittels einer Web service API kontrolliert wird, können sich Applikationen je nach verlangen, selbst mehr oder weniger Kapazität zuordnen. EC2 bietet einem außerdem Root-Zugriff auf die Maschine.  
AWS lässt die Konfiguration der einzelnen Server komplett frei wählen, sowohl Software als auch Hardware ist auswählbar (sofern verfügbar).

EC2 bietet eine Verfügbarkeit in allen Regionen von 99.95%.

Zudem bietet AWS Optionen wie Loadbalancing für den Eingehenden Traffic auf verschiedene EC2 Instanzen. Amazon EC2 Container Service bietet einem die Möglichkeit Docker-Container auf einem EC2-Cluster laufen zu lassen wobei sich hier nicht um das Cluster-Management gekümmert werden muss. Außerdem hat man die Möglichkeit VMs zu importieren sowie zu exportieren.[20]

Viele weitere Funktionen werden noch Angeboten, die meiner Meinung nach bedeutendsden wurden jedoch hier gelistet. (für genauere info siehe [20])

### Wann sollte man AWS verwenden?

Ich möchte hier keine definitive Aussage treffen. Im folgenden werde ich die generelle Meinung und die Vor- sowie Nachteile vermitteln. Entscheidungen müssen je nach Fall getroffen werden. Ein guter Ansatz um Hochverfügbarkeit zu gewährleisten ist AWS jedoch allemal.

**Vorteile:**

* Skalierbar in Minuten [21]  
  AWS bietet die Möglichkeit binnen weniger Minuten ein riesiges Ausfgebot an Rechenleistung zu liefern. Es ist möglich äußerst schnell viele EC2 Instanzen zu starten und somit in kürzester Zeit auf bspw. enormen Ansturm zu reagieren.
* Niedrige Initialkosten und es wird nur gezahlt was benötigt wird [22]  
  Sich als Unternehmen zu Beginn Hardware anzuschaffen kann teuer kommen, Instanzen bei AWS werden nach Stunden gezahlt und kosten nur so lange, wie sie verwendet werden.
* Schnelle und Down-Time freie Updates [22]  
  Während großen Updates ist es einfach möglich zusätzliche Hardware zu bestellen welche für die Zeit des Updates verwendet wird. Solche Hardware ist in diesem Zeitraum günstiger und schneller/einfacher anzuschaffen als in einem anderen Umfeld.
* Amazon ist für die Sicherheit der Infrastruktur verantwortlich [22]
* Speicher in großen Mengen sehr leicht verfügbar [21]

**Nachteile:**

* Schlechte Kommunikation [22]  
  Es wurden Erfahrungen gemacht, dass EC2-Instanzen ohne langer Vorwarnung (oder mit zuspäter) „retired“ wurden, was bedeutet, dass diese plötzlich nicht mehr erreichbar waren.
* Verteilung auf Zonen [22]  
  Der Ausfall einer einzelnen Instanz ist äußert unwahrscheinlich, ein Ausfall einer gesamten Availability Zone ist jedoch, sollte tatsächlich etwas Ausfallen, wahrscheinlicher, daher sollten bspw. Backup-Server in anderen Zonen oder sogar Regionen beheimatet sein. (Jedoch sei erneut gesagt, dass eine Verfügbarkeit von 99.95% erreicht wird)

### Grundlegendes

#### EC2

Die Elastic Compute Cloud von Amazon ist eine Instanz in der Cloud von AWS. Die Instanzen verfügen über verschiedene Konfigurationen, bspw. ausgelegt auf DBs oder Rechenintensive Applikationen. Diese Instanzen werden pro Stunde bezahlt und lassen sich binnen Minuten hochfahren um in kürzester Zeit Applikationen verteilen zu können. Diese EC2 Instanzen lassen sich auch programmatisch starten und beenden und ermöglichen einer Applikation somit, dass sie sich selbst skaliert. [23]

#### VPC

Die Virtual Private Cloud von Amazon ist ein isolierter Bereich in der AWS-Cloud. Eine solche Cloud gibt einem die Möglichkeit das gesamte Netzwerk selbst zu gestalten. Hier gibt es verschiedene Konfigurationsmöglichkeiten, welche später in dieser Ausarbeitung besprochen werden. Für die Verwendung einer solchen Cloud fallen keine zusätzlichenn Kosten an, lediglich die Instanzen die in der VPC laufen werden wie üblich verrechnet. Eine solche Cloud kann außerdem über Availability Zones gestreckt werden. [24]

#### EBS

Amazon Elastic Block Storage ist ein Service, welcher einem AWS-User die Möglichkeit bietet, seine Daten an einem zentralen Punkt zu verwalten. Diese Daten sind von allen EC2-Instanzen zugänglich und ermöglichen einen einfachen Datenaustausch. Die Daten werden innerhalb einer Availability Zone repliziert. [25]

#### S3

Bei Amazon Simple Storage Service wird lediglich Speicherplatz geboten (ähnlich Dropbox), welcher jedoch auf die Verwendung mit AWS-Services optimiert ist. [26]

#### Availability Zones

Availability Zones sind von Amazon erstellte Zonen, die geografisch voneinander in bestimmten Regionen verteilt sind. Beim erstellen einer EC2-Instanz lässt sich bestimmen in welcher Region diese Instanz starten soll und die Verteilung über Availability Zones ermöglicht einem höhere Ausfallsicherheit, da man bei Ausfällen ganzer Zonen geschützt ist. [27]

#### AMI

Amazon Machine Images sind Images die verwendet werden um EC2-Instanzen zu starten. Es gibt Template-AMIs, prinzipiell Grundimages, welche mit einem Betriebssystem versehen sind, jedoch lassen sich auch eigene erstellen, welche es einem erleichtern idente Instanzen zu starten. [28]

#### Security Group

Security Groups geben einem die Möglichkeit den ein- und ausgehenden Verkehr zu regeln. Einzelne EC2-Instanzen benötigen eigene Security Groups, VPCs benötigen als ganze welche. Auch Loadbalancer und andere Services die AWS bietet benötigen eine solche Security Group. [29]

## Sicherheit

Zur Sicherheit sei gesagt, dass bei AWS die Verwanntwortung geteilt wird. Für die Infrastruktur ist Amazon verantwortlich, bei Sicherheitsproblemen mit Applikationen die die Cloud verwenden oder in der Cloud laufen ist der User jedoch selbst verantwortlich. [30]

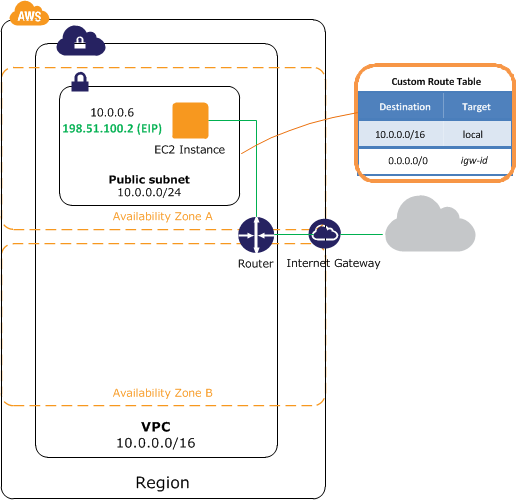
AWS erfüllt eine Reihe von Sicherheitsstandards und verwendet Best Practices, welche alle unter Anderem hier [30] gelistet werden. Zudem werden

AWS verwendet proprietäre Software zum Schutz gegen DDoS-Angriffe. Alle AWS APIs sind über SSL-geschützte Endpoints erreichbar, welche Server-Authentifikation ermöglichen. Alle EC2s generieren beim ersten Boot des AMIs (Amazon Machine Image) ein SSH-Host-Zertifikat welches in die Konsole der Instanz gelogged wird. [30]

Die Anzahl der Access Points zur Cloud ist beabsichtigt gering gehalten um den ein- und ausgehenden Verkehr besser überwachen zu können. [30]

### Virtual Private Cloud

Für mehr Sicherheit stellt Amazon die Virtual Private Cloud (VPC) zur Verfügung. Die VPC lässt ein gesamtes Netzwerk kontrollieren, isoliert von, jedoch in der AWS-Cloud. Außerdem besteht die Möglichkeit ein IPsec VPN zu einem eigenen Data Center aufzubauen.

Es gibt 4 Arten von Architekturen die die Amazon VPC bietet [30]:

* VPC mit einem einzelnen public Subnetz [31]

Die Instanzen befinden sich gesammelt in einem Subnetz, sind jedoch lediglich über ein Gateway mit dem Internet und anderen AWS-Diensten verbunden. Die Instanzen bekommen nicht wie üblich eigene öffentliche IPs (aus dem Amazon EC2 address space) zugeordnet. Zudem ist es möglich den Verkehr strikt zu kontrollieren.

Abbildung 7: einzelnes public [31]

* VPC mit public und private Subnetzen [32]

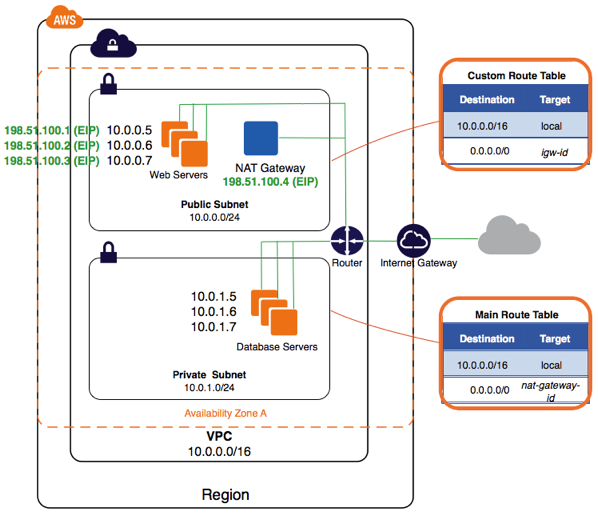
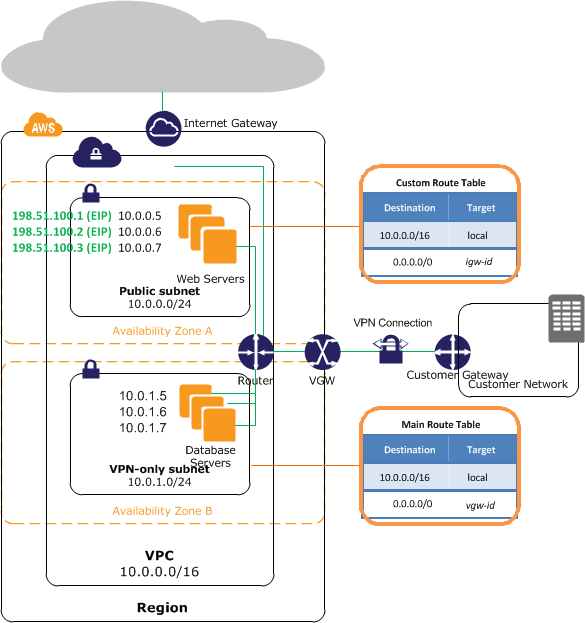
Hier ist sowohl ein public als auch ein private Subnetz vorhanden. Beide Netze können über den Router in der Cloud kommunizieren, die Instanzen im private Netz sind jedoch von außen nicht erreichbar.  
Empfohlen wird diese Konfiguration für bspw. Web-Applikationen, welche das Back-End auch in der Cloud auslagern wollen. Die DB-Server befinden sich im privaten Subnetz und können sich lediglich über das NAT-Gateway im public Subnetz mit dem Internet verbinden (für Updates z.B.), sind jedoch mit den Applikations-Servern im public Subnetz verbunden. Der Zugriff auf diese DB-Server ist jedoch von außen nicht möglich.

Abbildung 8: public und private [32]

* VPC mit sowohl public und private Subnetzen als auch einem HW-VPN [33]

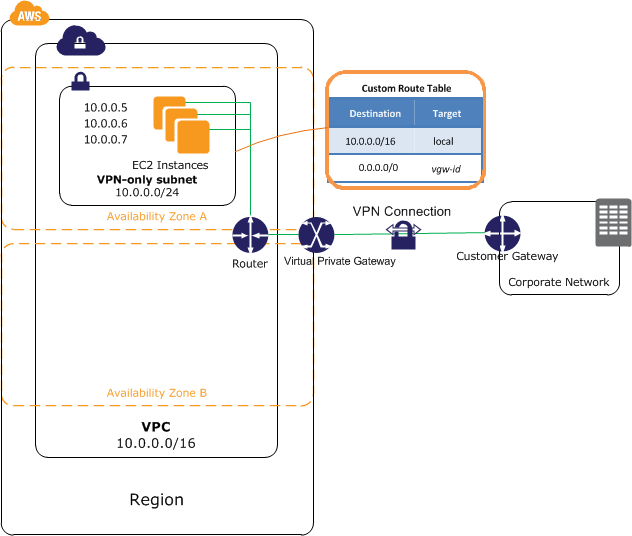
Bei dieser Art ist neben der eben genannten Struktur mit public und private Subnetzen ein HW-VPN inkludiert. Man hat hier ähnliche Möglichkeiten wie bei der vorherigen Lösung, zusätzlich jedoch noch eine sichere Verbindung zum eigenen Data Center. Diese Lösung bietet sich an um das eigene Nezwerk in die Cloud zu erweitern und direkt dort Internetzugang zu haben.

Abbildung 9: public und private mit VPN [33]

Abbildung 10: private mit VPN [34]

* VPC mit einem privaten Subnetz und einem HW-VPN [34]

Hier hat man in der Cloud keinen Internetzugang. Empfohlen wird diese Lösung wenn das Firmennetz in die Cloud erweitert werden soll, die dort vorhandenen Instanzen jedoch keinerlei Internetzugang benötigen.

VPCs haben die Möglichkeit den eingehenden und ausgehenden Verkehr zu kontrollieren. Sogenannte „security groups“ ersetzen hier die Firewall und lassen Regeln erstellen, welche, wie üblich, den Traffic einschränken lassen.

### Amazon Inspector

Amazon Inspector ist ein Tool welches einem hilft Sicherheitprobleme eigener Applikationen aufzudecken. Es verwendet best practices und typische Fehlerfälle um die Applikation zu überprüfen und danach einen Bericht dazu auszugeben. Amazon Inspector kann vor dem Deployment als auch während die Applikation altiv ist in Verwendung sein um ständig vor Sicherheitsrisiken zu warnen. [35]

### Amazon WAF

Bei der Web Application Firewall von Amazon handelt es sich um ebendiese. Sie schützt vor üblichen Angriffen wie SQL-Injections und Cross-Site-Scripting und gibt einem sowohl die Möglichkeit eigene Regeln zu erstellen als auch den Verkehr einzuschränken. [36]

## High Availability

Bei AWS hat man die Möglichkeit seine Infrastruktur über Regionen zu verteilen. Prinzipiell werden die Ressourcen nicht verteilt, es lässt sich jedoch wählen. Alle Regionen sind komplett unabhängig und sog. Availability Zones (isolierte Zonen) sind in einer Region geografisch getrennt, jedoch mit einem „low-latency link“ verbunden. Zusätzlich werden die Zonen mit backup-Generatoren gesichert und die Stromversorgung wird aus verschiedenen Anlagen von verschiedenen Anbietern bezogen um auch dieses Risiko zu entfernen. [20]

AWS ist insofern vor Ausfällen geschützt, als dass sie keine „kalten“ Data Centers haben. Im Falle eines Ausfalls ihrerseits, werden die von den Kunden verwendeten Dienste aus der betroffenen Zone verlagert.

### Auto Scaling

Der Auto Scaling Service von AWS ist ein Tool um seine EC2-Instanzen verfügbar zu halten. Es ermöglicht einem bei größerer Nachfrage mehr Kapazität anzufordern und bei geringerer diese wieder zurückzuschrauben. Dafür interagiert Auto Scaling unter Anderem mit dem Load Balancer um die neuen Instanzen auch verwenden zu können. Auto Scaling geht jedoch davon aus, dass die Server homogen sind, folglich die gleichen Ressourcen zur Auswahl haben. Zur Vervielfachung der Instanz wird das AMI der zu skalierenden EC2-Instanz verwendet.  
Um die Skalierung sinnvoll zu gestalten, werden sogenannte „scaling policies“ erstellt, welche bestimmen wann und um wieviel skaliert wird. Hier wird einem die Möglichkeit geboten entweder nach einem Zeitplan vorzugehen oder auf die metrics der einzelnen Instanzen zu reagieren und bspw. bei ungewöhnlich hoher Prozessorauslastung neue Instanzen zu aktivieren.  
Außerdem ist die Möglichkeit vorhanden, beim Hinzufügen oder Terminieren einer Instanz den User zu informieren. Es kann bspw. eine Mail an den User gesendet werden (hierzu wird Amazon SNS verwendet[43]) [37]

# Couchbase

## High Availability bei Couchbase

Neben der üblichen Option der Replikation kommen bei Couchbase diese Optionen hinzu.

### Rack Zone Awareness

Hierbei handelt es sich um ein Feature, dass es erlaubt, logische Gruppen von Servern, innerhalb eines Clusters zu bilden, wobei jede Gruppe physikalisch zu einem Rack gehört. Hier macht es Sinn bei bspw. 2 Gruppen Gruppe 2 neben den eigenen Buckets die Replikas von Gruppe 1 zu halten und umgekehrt. Es ist möglich in den Gruppen eine verschiedene Anzahl von Servern zu halten, jedoch nicht empfohlen. [38]

### XDCR

Cross Data Center Replication ist eine Technologie von Couchbase die Daten zwischen 2 oder mehr autonomen Couchbase Server Clustern repliziert. In den meisten Fällen befinden sich diese Cluster in verschiedenen Data Centern, geografisch verteilt.

Für Disaster Recovery können diese Cluster als Hot-Standby verwendet werden, jedoch können die Cluster auch als Backup verwendet werden. Zudem kann die geografische Verteilung dem End-User einen Vorteil in Sachen Latenz bringen.

XDCR bietet außerdem die Möglichkeit mittels bi-direktionaler Replikation sowohl Lese- als auch Schreibzugriffe an jedem Endpunkt.

Die Einrichtung von XDCR ist einfach, durch wenige Clicks können Cluster als zusammengehörig markiert werden und die Replikation-Streams hinzugefügt werden. Es ist jedoch nicht notwendig einen ganzen Cluster zu replizieren, für die Replikation werden Buckets ausgewählt die repliziert werden sollen. [39] Zudem gibt es keine Einschränkungen in Topologien. [40]

# **Literaturverzeichnis**

1. Clusterbau: Hochverfügbarkeit mit pacemaker, OpenAIS, heartbeat und LVS (2010); Michael Schwartzkopff; ISBN: 978-3-89721-919-9
2. SunCluster (2010); Rolf Dietze; ISBN: 978-3-540-33805-5
3. Verwendung von Hochverfügbaren Architekturen für Server; IT-Grundschutz – Bundesamt für Sicherheit in der Informationstechnik; online verfügbar unter: <https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/m/m02/m02314.html> [zuletzt abgerufen am 27.3.2016]
4. Deterministisches Chaos (2002); Welt der Physik; online verfügbar unter: <http://www.weltderphysik.de/gebiet/theorie/chaos-und-ordnung/deterministisches-chaos/> [zuletzt abgerufen am 21.3.2016]
5. Chaos Engineering in Japan (Nov. 1995); Kazuyuki Aihara und Ryu Katayama
6. Principles of Chaos Engineering (Sep. 2015); online verfügbar unter: <http://principlesofchaos.org> [zuletzt abgerufen am 21.3.2016]
7. Simian Army; Netflix – Github; online verfügbar unter: <https://github.com/Netflix/SimianArmy> [zuletzt abgerufen am 25.3.2016]
8. Chaos Monkey Released Into The Wild (Jul 2012); Techblog Netflix; online verfügbar unter: <http://techblog.netflix.com/2012/07/chaos-monkey-released-into-wild.html> [zuletzt abgerufen am 21.3.2016]
9. Quick Start Guide; Netflix – Github; online verfügbar unter: <https://github.com/Netflix/SimianArmy/wiki/Quick-Start-Guide> [zuletzt abgerufen am 25.3.2016]
10. The Netflix Simian Army; Techblog Netflix; online verfügbar unter: <http://techblog.netflix.com/2011/07/netflix-simian-army.html> [zuletzt abgerufen am 21.3.2016]
11. Chaos Engineering Upgraded; Techblog Netflix; online verfügbar unter: <http://techblog.netflix.com/2015/09/chaos-engineering-upgraded.html> [zuletzt abgerufen am 21.3.2016]
12. Introduction to Oracle Clusterware; Oracle Documentation, online verfügbar unter: <https://docs.oracle.com/cd/E14795_01/doc/rac.112/e10717/intro.htm> [zuletzt abgerufen am 29.03.2016]
13. Tablespaces, Datafiles, and Control Files; Oracle Documentation; online verfügbar unter: <https://docs.oracle.com/cd/B19306_01/server.102/b14220/physical.htm> [zuletzt abgerufen am 29.03.2016]
14. Introduction to the Oracle Database; Michele Cyran, Paul Lane; online verfügbar unter: <https://web.stanford.edu/dept/itss/docs/oracle/10g/server.101/b10743/intro.htm> [zuletzt abgerufen am 29.03.2016]
15. Data Blocks, Extents, and Segments; Oracle Documentation, online verfügbar unter: <https://docs.oracle.com/cd/B10500_01/server.920/a96524/c03block.htm> [zuletzt abgerufen am 29.03.2016]
16. High Availability; Michele Cyran, Paul Lane; online verfügbar unter: <https://web.stanford.edu/dept/itss/docs/oracle/10g/server.101/b10743/high_av.htm#g38347> [zuletzt abgerufen am 29.03.2016]
17. Introduction to Oracle RAC; Oracle Documentation; online verfügbar unter: <https://docs.oracle.com/cd/E11882_01/rac.112/e41960/admcon.htm#RACAD1111> [zuletzt abgerufen am 29.03.2016]
18. Oracle Database High Availability Solutions for Planned Downtime; Oracle Documentation; online verfügbar unter: <https://docs.oracle.com/cd/E11882_01/server.112/e17157/planned.htm#HAOVW144> [zuletzt abgerufen am 29.03.2016]
19. Getting Started with Data Guard; Oracle Documentation; online verfügbar unter: <https://docs.oracle.com/cd/E11882_01/server.112/e41134/standby.htm#SBYDB00102> [zuletzt abgerufen am 29.03.2016]
20. Overview of Amazon Web Services; Amazon Web Services; online verfügbar unter: <https://d0.awsstatic.com/whitepapers/aws-overview.pdf> [zuletzt abgerufen am 29.03.2016]
21. When to use aws – and when not to; David N. Welton; online verfügbar unter: <http://journal.dedasys.com/2015/01/22/when-to-use-aws-and-when-not-to/> [zuletzt abgerufen am 29.03.2016]
22. AWS: the good, the bad and the ugly; seldo (awe.sm); online verfügbar unter: <http://blog.awe.sm/2012/12/18/aws-the-good-the-bad-and-the-ugly/#~pGFBo9y7wEzKlm> [zuletzt abgerufen am 29.03.2016]
23. Amazon EC2 – Virtual Server Hosting; Amazon Web Services; online verfügbar unter: <https://aws.amazon.com/ec2/> [zuletzt abgerufen am 29.03.2016]
24. Amazon Virtual Private Cloud; Amazon Web Services; online verfügbar unter: <https://aws.amazon.com/vpc/> [zuletzt abgerufen am 29.03.2016]
25. Amazon Elastic Block Store; Amazon Web Services; online verfügbar unter: <https://aws.amazon.com/ebs/> [zuletzt abgerufen am 29.03.2016]
26. Amazon S3 Online Speicher in der Cloud; Amazon Web Services; online verfügbar unter: <https://aws.amazon.com/de/s3/> [zuletzt abgerufen am 29.03.2016]
27. Regions and Availability Zones; Amazon Web Services; online verfügbar unter: <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html> [zuletzt abgerufen am 29.03.2016]
28. Amazon Machine Images (AMI); Amazon Web Services; online verfügbar unter: <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html> [zuletzt abgerufen am 29.03.2016]
29. Amazon EC2 Security Groups for Linux Instances; Amazon Web Services; online verfügbar unter: <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html> [zuletzt abgerufen am 29.03.2016]
30. Amazon Web Services: Overview of Security Processes; Amazon Web Services; online verfügbar unter: <https://d0.awsstatic.com/whitepapers/Security/AWS%20Security%20Whitepaper.pdf> [zuletzt abgerufen am 29.03.2016]
31. Scenario 1: VPC with a single public subnet; Amazon Web Services; online verfügbar unter: <http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Scenario1.html> [zuletzt abgerufen am 29.03.2016]
32. Scenario 2: VPC with Public and Private Subnets (NAT); Amazon Web Services; online verfügbar unter: <http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Scenario2.html> [zuletzt abgerufen am 29.03.2016]
33. Scenario 3: VPC with Public and Private Subnets and Hardware VPN Access; Amazon Web Services; online verfügbar unter: <http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Scenario3.html> [zuletzt abgerufen am 29.03.2016]
34. Scenario 4: VPC with a Private Subnet Only and Hardware VPN Access; Amazon Web Services; online verfügbar unter: <http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Scenario4.html> [zuletzt abgerufen am 29.03.2016]
35. Amazon Inspector (Preview); Amazon Web Services; online verfügbar unter: <https://aws.amazon.com/inspector/> [zuletzt abgerufen am 29.03.2016]
36. AWS WAF – Webanwendungs-Firewall; Amazon Web Services; online verfügbar unter: <https://aws.amazon.com/de/waf/> [zuletzt abgerufen am 29.03.2016]
37. Auto Scaling; Amazon Web Services; online verfügbar unter: <https://aws.amazon.com/autoscaling/> [zuletzt abgerufen am 29.03.2016]
38. Rack Awareness; Couchbase Documentation; online verfügbar unter: <http://docs.couchbase.com/admin/admin/Concepts/concept-rack-awareness.html> [zuletzt abgerufen am 29.03.2016]
39. Enabling HA – DR for Mission Critical Production Systems; Kirk Kirkconnell; online verfügbar unter: <http://connect15.couchbase.com/agenda/enabling-ha-dr-mission-critical-production-systems/> [zuletzt abgerufen am 29.03.2016]
40. High Availability and disaster recovery; Couchbase Documentation; online verfügbar unter: <http://developer.couchbase.com/documentation/server/4.1/ha-dr/ha-dr-intro.html> [zuletzt abgerufen am 29.03.2016]
41. ROWID Pseudocolumn; Oracle Documentation; online verfügbar unter: <https://docs.oracle.com/cd/B19306_01/server.102/b14200/pseudocolumns008.htm> [zuletzt abgerufen am 29.03.2016]
42. Data Type and DDL Support on a Logical Standby Database; Oracle Documentation; online verfügbar unter: <https://docs.oracle.com/cd/E11882_01/server.112/e41134/data_support.htm#SBYDB00305> [zuletzt abgerufen am 29.03.2016]
43. Amazon Simple Notification Service (SNS); Amazon Web Services; online verfügbar unter: <https://aws.amazon.com/sns/> [zuletzt abgerufen am 29.03.2016]

# Abbildungsverzeichnis

Abbildung 1: Verfügbarkeit und Auszeit pro Monat bzw. Jahr [1] 4

Abbildung 2: Serielle Kopplung zweier Komponenten [1] 5

Abbildung 3: Parallele Kopplung zweier Komponenten [1] 6

Abbildung 4: Verfügbarkeit bei redundanter Auslegung mit identischen Komponenten [1] 6

Abbildung 5: Verlauf eines Chaos Kong Experiments [11] 11

Abbildung 6: Data Blocks, Extents und Segments [13] 11

Abbildung 7: einzelnes public [31] 17

Abbildung 8: public und private [32] 17

Abbildung 9: public und private mit VPN [33] 17

Abbildung 10: private mit VPN [34] 18