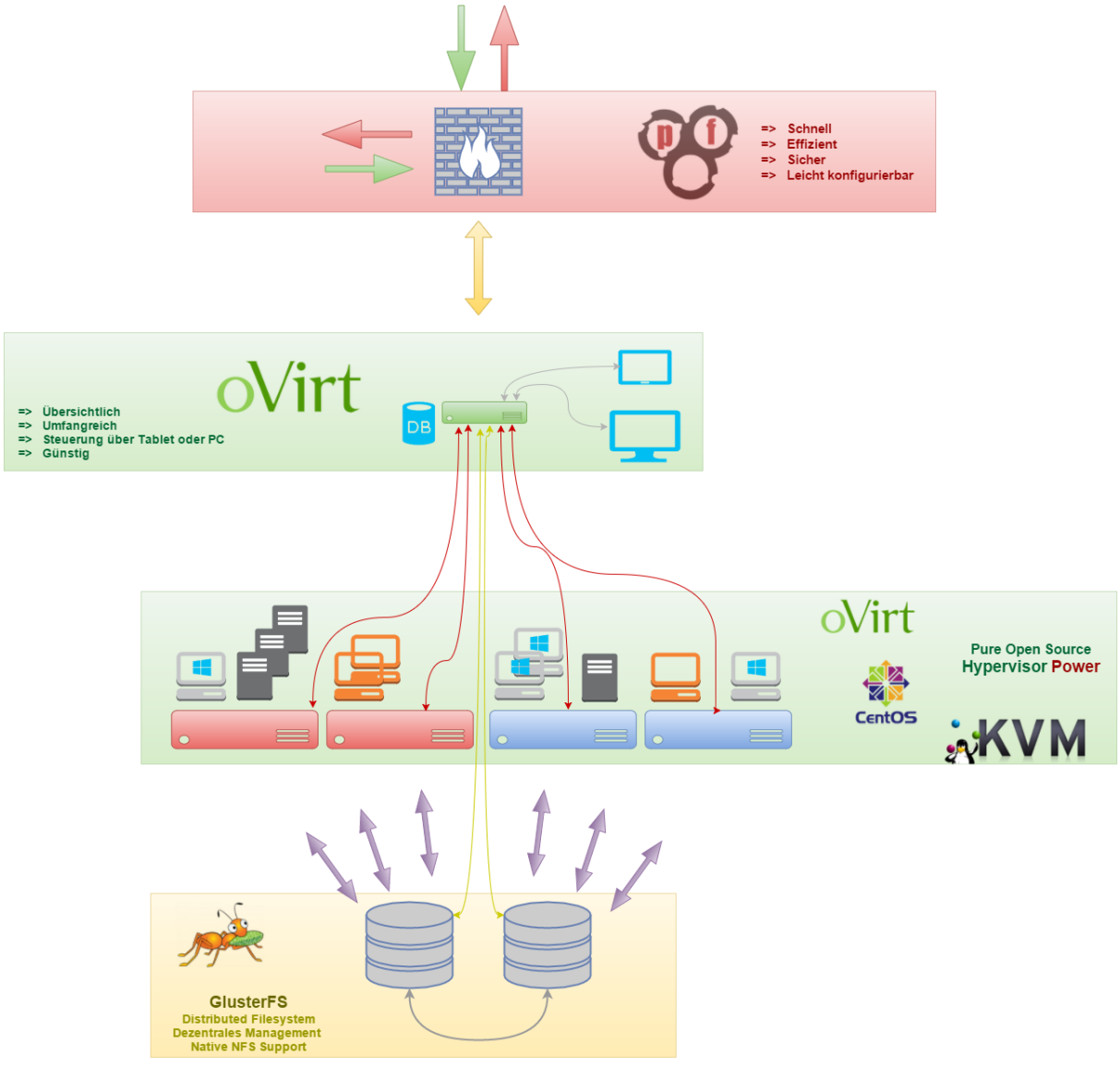
# FullOpenSourceVirtualization

## Eine Virtualisierungsinfrastruktur auf Basis von pfSense, oVirt und GlusterFS





**Eine Diplomarbeit von Bogdanovic Theodor für die Höhere Fachschule Uster, in der Fachrichtung Telekommunikation**


**8. März 2016**

## Impressum



## Danksagungen

Der Prüfung dieser Arbeit auf korrekte Rechtschreibung, Grammatik und Stilistik soll an dieser Stelle den nachfolgend genanten Korrekturlesern herzlichst für ihre hervorragende und professionelle Arbeit, in den jeweiligen Kapiteln gedankt werden:

*Hugo Brändle, für die Korrekturlesung der Kapitel 1 bis 7*

*Claudia und Michael Gosswiler, für die Korrekturlesung des umfangreichen Kapitel 8*

*Bouillet Alexandre, für die Korrekturlesung der Kapitel 9 bis zum Ende*

Bogdanovic Theodor
Aabächliweg 1
8854 Siebnen

**Selbstständigkeitserklärung des Autor für dieses Werk und die damit verbundene Realisation**

Der oben namentlich genante Autor dieser Dokumentation und allen damit verbundenen Arbeiten, hinschlich der Realisierung des gesamten Projektes, erklärt an dieser Stelle die absolute Selbstständigkeit der Umsetzung aller Schritte, die notwendig waren um diese Diplomarbeit für die Höhere Fachschule Uster, nach den Anforderungen des Auftrages realisiert zu haben. Ausnahmen bilden hier die obligatorische Korrekturlesung der Arbeit, zum alleinigen Zwecke der Prüfung auf Rechtschreibung, Grammatik und Stilistik der Rohfassung. Änderungen des Kontextes durch die Korrekturlesung innerhalb der Rohfassung wurden weder zugelassen noch umgesetzt. Eine weitere Ausnahme bildet an dieser Stelle die Montagehilfe der Hardwarekomponenten, hier speziell das Konstruieren der vom Standard abweichenden Halterungen des Rack, durch Bogdanovic Gavrilo welcher an dieser Arbeit beteiligt war.

Bogdanovic Theodor, Siebnen den 8. März 2016

---

# 1 Inhaltsverzeichnis

# 2 Zusammenfassung/ Management – Summary

## 2.1 Management – Summary

Um zu demonstrieren, dass es möglich ist einen Virtualisierungs – Cluster samt eigenständigen Storage, mit allen Extras die heute üblich sind auf reiner open source Basis zu errichten, wurde die Idee zu dieser Diplomarbeit aufgegriffen. Das Ziel dieser Arbeit ist die Erstellung eines Clusters, welcher eine bequeme und zentralisierte Bedienoberfläche bietet, mit deren Hilfe folgende Punkte leicht und schnell realisiert werden können:

- Schnelles, vorlagenbasiertes Erstellen von virtuellen Maschinen
- Einfaches und schnelles Netzwerk – Management über GUI – basierte Werkzeuge
- Die Möglichkeit Ressourcen – Pools zu bilden, um eine höhere Anzahl an virtuellen Maschinen innerhalb der Hosts zu realisieren. Dies sollte vorzugsweise ebenfalls schnell und leicht möglich sein.
- Die Bereitstellung eines zentralen und autonomen Storages, welcher eine grösstmögliche Ausnutzung des Speichervolumens bietet, aber dennoch hohe Geschwindigkeiten bei der Verwendung von Standard – Hardware – Komponenten ermöglicht.
- Die sich aus einem zentralisierten Storage bietenden Möglichkeiten der Virtualisierung wie etwa Live – Migration oder High Availability sollen so ermöglicht werden.

Für die oben genannten Kern – Features gilt aber ein technischer Grundsatz. Es sollen nur Standard – Komponenten verwendet werden, wie sie auch im üblichen Desktop Computer Bereich anzutreffen sind. Dies resultiert im klaren Verzicht auf teure Serverkomponenten wie Hardware basierte RAID – Controller, Glasfaser – Komponenten im Netzwerkbereich, die Nutzung von teuren gemanagten Switches oder eine sonstige Verwendung von Serverperipherie wie etwa SAS Festplatten.

Die verwendeten Kerntechnologien, welche aus dem heute grosszügigen open source Angebot hierfür genutzt werden sind:

- pfSense: zur zentralisierten Anbindung des Netzwerks an die Aussenwelt. Wobei der FreeBSD – Unterbau von pfSense an dieser Stelle nicht nur die typische Firewall Rolle, sondern auch die Switching – Aufgaben eines gemanagten Switches übernimmt.
- oVirt: Als dreh und Angelpunkt sämtlicher Konfigurationsmöglichkeiten der robusten Kernelbasierten Virtualisierung die Linux heute anbietet.
- GlusterFS: Diese Variante der Verteilten Dateisysteme wird heute von RedHat propagiert und dies aus gutem Grund. Die Möglichkeiten an Kombinationen hinsichtlich Ausfallsicherheit und beschleunigtem IO – Speed bilden das heutige RAID – System von Festplattenverbünden beinahe 1:1 nach. Jedoch mit der Konkurrenz weit überlegenen Einfachheit bezüglich Konfiguration und Bedienung.

Beim Lesen des Realisierungkonzeptes könnte vielleicht der Eindruck entstehen, dass manche Ansätze gar nicht so viel billiger sind als eine „echte“ Lösung. Dies könnte bei einigen Punkten tatsächlich zutreffen. Jedoch folgt der Autor dieser Arbeit dem Grundsatz „Alt ist nicht gleich schlecht“ was sich in der Verwendung von Occasion – Hardware, höheren Leistungsumfangs widerspiegelt die beim Lieblingshändler des Autors, „benno-shop.com“ erworben wurden. So kann das Konzept leistungsstark zu günstigen Preisen dennoch eingehalten werden.

### 2.1.1 Projektidee

Die reine Demonstration der Möglichkeit nur mit günstiger Standard Hardware einen Virtualisierungs-Cluster zu erstellen ist nicht die einzige Motivation des Autors. Eine zweite treibende Kraft ist die Notwendigkeit. Dieses Projekt ist keine Neuentwicklung für den persönlichen Produktiveinsatz, sondern eine seit langer Zeit geplante Ablösung eines bis dato bestehenden Xenserver 6.2 Clusters, welcher hinsichtlich mangelnder Unterstützung von diversen Betriebssystem – Varianten, schlechter Bedienbarkeit und der teilweise nicht vorhandenen Funktionen nicht mehr tragbar ist. Somit soll dieses Projekt ein leicht bedienbarer und robuster Ersatz des bestehenden Produktes werden, welches dem Autor dieser Arbeit eine solide und selten zu wartende Plattform bietet für das Bereitstellen diverser Dienste.

### 2.1.2 Erwähnenswerte Schwierigkeiten und unerwartete Probleme

In einer ersten Testphase, welche noch vor Einreichen des Antrages für diese Diplomarbeit durchgeführt wurde, konnte die Installation von oVirt 3.5 einfach vollzogen werden. Da diese Produktivversion aber in kleinen Segmenten weiterentwickelt wurde und Teile der erst heute produktiven Version 3.6 als Standard integriert wurden, änderte sich diese eigentlich simple Installation in eine Odyssee. Diese führte zum zeitraubenden Suchen in teilweise veralteten Tutorials und Foren. Dennoch konnten, mit zugegeben grösserem Zeitverlust, die meisten Probleme ermittelt und behoben werden.

Ein schwerwiegenderes Problem war das  High Availability, den dieses setzt nicht wie bspw. VMware rein auf der Überwachung vom Management – Center auf, sondern erfordert das Vorhandensein eines Lights Out Management. Da dies aber nicht auf allen Hosts vorhanden ist und auch aus Sicht des Autors ungern in Produktivsystemen genutzt wird, ist eine 100%-ige Implementierung  des vollautomatischen HA nicht möglich. Eine „billigere“ semi – automatische Lösung wurde hier gewählt, welche im entsprechenden Punkt „8 Realisierung“ näher erläutert wird.

### 2.1.3 Lösungsansatz grob skizziert

Aufbauend auf der Vorplanung, welche aus den Vorstudien und den Konzeptstudien hervorgegangen ist, aber auch gestützt auf die zur Verfügung stehenden Mittel sieht der Lösungsweg wie folgt aus:

- Sequenz 1: Installation und Inbetriebnahme des Netzwerksegments. Dies beinhaltet die Installation des pfSense – Systems als Firewall/Switch. Hinzu kommt die Planung und teilweise Umsetzung der Verkabelung.
- Sequenz 2: Vorbereitung des Storage – System. Dies kann unabhängig im Vorfeld vollzogen werden. Die Installation beinhaltet auch die Sicherung der bis dato noch laufenden Xenserver – Umgebung bzw. der noch benötigten virtuellen Maschinen. Der hier grösste Aufwand ist die Extraktion der Festplatten aus dem bestehenden System sowie die Integration in den zukünftigen RAID 5 Verbund. Eine kleine zeitliche Reserve, welche aber nirgends spezifisch erfasst wurde, ist hier für „nicht ganz ernst  gemeinte“ persönlichen Tests mit einer für den Autor neuen Technologie reserviert.
- Sequenz 3: Vorbereiten der Virtualisierungsumgebung. Dies beinhaltet die Installation des Management Teils, die Installation der einzelnen Nodes sowie die Zusammenführung der entsprechenden Komponenten.
- Sequenz 4: Beinhaltet die Zusammenführung sämtlicher Komponenten zu einem aus einem Guss wirkenden Komponente.

Dies ist eine kurze und grobe Beschreibung des allgemeingültigen Vorgehens bei der Realisierung eines solchen Systems. Hier explizit ausgeschlossen ist die handwerkliche Arbeit welche zum Umbau des Racks notwendig war. Auch nicht berücksichtigt wurde die Stilllegung des alten Xenserver – Systems, welche teilweise schon vor Beginn dieser Arbeit initialisiert wurde.

### 2.1.4 Erreichen der Zielsetzung

Aufbauend auf den Definitionen im Pflichtenheft, der gesetzten zeitlichen Rahmenfrist sowie des im Vergleich zur Vorgängerlösung besseren Produktes, kann guten Gewissens die Erreichung der Zielsetzung bestätigt werden.

### 2.1.5 Erfahrungen/ Erkenntnisse aus diesem Projekt

Aufgrund der doch erheblichen strukturellen Änderungen von oVirt 3.5 im Vergleich zur ersten Testinstallation, welche noch vor Abgabe des Antrages dieses Projektes vorgenommen wurden, muss ein erhöhter zeitlicher und nervlicher Mehraufwand zugegeben werden. Dies kann open source Produkte mit reinem Community – Support in ein schlechteres Licht rücken als Produkte mit kommerziellem Support. Doch macht dies aus Sicht des Autors dieser Arbeit genau den besonderen Reiz aus, welcher das Erlernen neuer Technologien so besonders macht. Denn erst wenn man eine Menge Arbeit und vor allem Nerven in eine Arbeit investiert hat, kann man sich über das Ergebniss richtig freuen.

### 2.1.6 Zeitaufwand (Soll- / Ist – Vergleich)

Da der Autor dieser Arbeit stets neben dem realen Zeitplan eine Negativreserve definiert, um auf diese Weise sicherzustellen, dass manchmal sinnloser Zeitaufwand schneller kompensiert werden kann, fallen die hohen zeitlichen Investitionen zur Problemlösung an gewissen Stellen des Projektes nicht schwer ins Gewicht. Im Gesamtüberblick kann die geschätzte und entsprechend gesetzte zeitliche Rahmenfrist als akzeptabel definiert werden.

### 2.1.7 Ausblick

Diese Dokumentation beinhaltet neben dem gesamten strukturellen Ablauf der Realisierung des Projektes auch zahlreiche Illustrationen, welche helfen sollen das komplexe Gebilde, das einen solchen Cluster ausmacht, besser zu verdeutlichen. Zusätzlich bietet es neben dem eigentlichen Ablauf auch einige speziell markierte Bereiche, welche auf zusätzliche Infos, Kurztipps oder Alternativlösungen hinweisen. Diese „Extras“ sind für die Realisierung dieser Arbeit nicht zwingend notwendig oder sie weichen von den Vorstellungen des Autors stark ab. Jedoch sind es akzeptable und bereits evaluierte Alternativen, die womöglich dem einen oder anderen helfen können, gewisse Stolpersteine im Vorfeld zu erkennen. Somit kann diese Projektspezifische Dokumentation auch als allgemeine Installations- und Bedienungsanleitung angesehen werden.

# 3 Einleitung

## 3.1 Motivation zu diesem Projekt

Die Motivation zu diesem Projekt entsprang aus zwei Hauptargumenten. Das erste und vermutlich wichtigste Argument ist die Tatsache, dass die bis dato bestehende Lösung (Xenserver 6.2) als nicht aktuell galt und dringend eines Updates bedurfte. Zusätzlich zum Alter der bestehenden Lösung kam die Tatsache hinzu, dass nicht alle Bedürfnisse hinsichtlich virtualisierbarer Betriebssysteme abgedeckt werden konnten. In diesem Fall sei hier speziell auf Solaris bzw. den open source Fork Openindiana verwiesen. Ein weiterer schwerwiegender Nachteil in der bestehenden Lösung war die schlechte Kompatibilität zwischen den unterschiedlichen Prozessor – Familien. So ist es nur mit erheblichem Aufwand möglich zwei Systeme mit unterschiedlichen Prozessor – Generationen in einen Pool zu vereinen, um so Live – Migration zu ermöglichen. Da wie bereits erwähnt ein Upgrade nötig wurde und auch die Möglichkeit seitens Xenserver mit Version 6.5 da war, musste man sich schon die Frage stellen ob dies angesichts der oben aufgezählten Mängel seitens Xenserver lohnenswert wäre. Zwar ist der Autor ein grosser Bewunderer des Xen – Verfahrens als Hypervisor und teilt die Meinung, dass dies der einzig wahre Typ 1 Hypervisor auf dem Markt ist. Jedoch ist die Entwicklung rund um Xen und speziell um Xenserver als Gesamtprodukt in den letzten Jahren etwas ins Stocken geraten. Zusätzlich muss man aber auch sagen, dass KVM (Kernel-basierte Virtualisierung, Linux) hinsichtlich der Tatsache dass es sich um ein Zwischending von Typ – 1 und Typ – 2 Hypervisor handelt, nur insoweit sicher ist, als dass die Mandatory Access Control (MAC), welche über SELinux (Security-Enhanced Linux) realisiert ist, auch funktioniert. Ob die SELinux Rules aber auch eine 100%-ige Isolation gewährleisten, kann heute nur schwer nachvollzogen werden, da die Regeln eine so hohe Komplexität erreicht haben, dass von manuellen Eingriffen dringendst abzuraten ist. Was aber heute in der Virtualisierung oberste Priorität geniessen tut und auch der ausschlaggebende Punkt des Autors war um dennoch auf KVM zu setzen, ist die hohe Anzahl an erstklassigen Management – Systemen welche dem Hypervisor erst seine wahre Macht verleihen. Unter alle Produkten sticht oVirt aus zwei gründen besonders hervor:

- Es ist die Entwicklungsplattform von RedHat, somit ist ein hoher Standard zu erwarten, wobei aufgrund der langen Produktlebenszyklen von RedHat Produkten auch mit einem langen LTS (Long term support) seitens Community zu rechnen ist.
- Es ist ein Rundumpaket. Man bekommt alle Funktionen so, wie sie auch das kommerzielle Produkt enthalten sind.

Zu Punkt zwei könnte man nun gegen- argumentiert , dass dies auch bei anderen Herstellern der Fall ist und man wäre auch im Recht mit dieser Aussage. Jedoch versucht RedHat ähnlich wie SuSE die Installation, wie auch die Konfiguration möglichst leicht zu halten und entwickelt daher eine Vielzahl an Assistenten und GUI's welche einem die Arbeit erleichtern oder gar teilweise abnehmen. All diese Werkzeuge stehen einem dann auch zur Verfügung. Hersteller wie Proxmox mit Virtual Enviroment liefern zwar gute Produkte, jedoch sind nicht sämtliche Funktionen bequem per GUI handlebar, da die Werkzeuge schlicht fehlen oder nur im kommerziellen Produkt erhältlich sind. Einfach gesprochen, oVirt beherrscht alles out of the box und ermöglicht dem Entwickler mit seiner breiten Unterstützung an möglichen Konfigurationsmöglichkeiten die freie Wahl. Zusätzlich ist es die momentan einzige Implementierung, welche standardmässig auch GlusterFS als Speicherdomäne sauber integriert. An dieser Stelle sei noch gesagt, dass der Autor dieses Werkes ein grosser Fan dieser Technologie geworden ist und es wahrscheinlich auch bleiben wird.

### 3.1.1 Wozu der ganze aufwand?

Nun könnte man sich fragen wozu den der ganze Aufwand gut sein soll, wenn man doch heute für wenig Geld ganze Infrastrukturen beim Hoster seiner Wahl haben kann. Diese Frage mag sicherlich zutreffen wenn man sich die hohen Kosten bezüglich Hardware, Stromrechnung, Zeitaufwand oder die zu investierenden Nerven, wenn es mal wieder Ärger gibt, in Betracht zieht. Sieht man sich aber die Freiheiten die man mit einer eigenen Infrastruktur hat, dann sieht die Situation wieder anders aus. Betrachten wir mal folgende Punkte:

- Man kann selber so viele VM's haben wie man will (oder die Infrastruktur hergibt)
- Man bezahlt seinen eigenen Internetanschluss; Stichwort: „unlimitiert“ Traffic
- Die Infrastruktur kann so aussehen wie man es selber möchte und nicht so wie sie einem aufgezwungen wird; Stichworte: Distribution, Versionen, Interpreter diverse Kombinationen der genannten Punkte
- Ressourcen – Limitierung (CPU, RAM, Network, etc.) wie man es selber für richtig hält.
- In Verbindung mit dem oberen Punkt, kein „garantiert so und so viel von dem oder dem“, sondern klare Verhältnisse, man weiss wo das Maximum liegt und ob es auch wirklich so ist.
- Man kann das nutzen was man will. Möchte ich ein FreeBSD anstelle einer Linux Distribution um bspw. Eine Ruby on Rail Infrastruktur zu betreiben, dann installiere ich und suche nicht erst lange nach einem Hoster der auch FreeBSD unsterstützt.
- Und wohl der wichtigste Punkt von allen. Ich weiss stets wo meine Sachen gehostet sind und muss mich nicht fragen wie genau es der Hoster mit nimmt; Stichwort: NSA Affäre

Wenn man sich diese stark abgekürzte Liste an Vorteilen einmal ansieht, dann ist der Autor dieser Arbeit der Meinung, solltn alle Nachteile einem wie nichts vorkommen.

Ein weiterer Vorteil, aber auch nur wenn man genügend Kapazitäten zur Virtualisierung besitzt, ist die Tatsache, dass man immer eine VM zur Verfügung hat. So kam es auch des öfteren vor, dass man von Freunden oder seinen Kommilitone angesprochen wurde, ob man doch noch eine VM schnell zum testen oder für eine Arbeit haben könnte.

## 3.2 Hinweise zur Dokumentation

Der besseren Lesbarkeit halber, aber auch um den Ansprüchen einer Anleitung gerecht zu werden, nutzt diese Dokumentation folgende Darstellungsoptionen um die unterschiedlichen Kontexte besser zu differenzieren:

```
# Hier drin stehen Befehle oder Konfigurationen die an einem bestimmten Punkt in die
  Kommandozeile eingegeben werden müssen.

# Beginnen die Zeilen wie diese mit einem „#“ dann muss man als root eingeloggt sein.

$ Beginnen sie jedoch mit einem „$“ dan genügt ein regulärer User.
```

**TIPP**

**In diesen Felder stehen hilfreiche Tipps, welche nicht vorschreiben wie es gemacht werden muss, aber meistens macht es Sinn ihnen zu folgen um schnell, an nutzbare Ziel zu gelangen.**

**ALTERNATIVE**

**In diesen Feldern wird auf mögliche Alternativ – Konfigurationen verwiesen, welche kurz getestet wurden aber für den restlichen Verlauf dieser Dokumentation nicht mehr berücksichtigt werden können. Achtung: Es wird wirklich nicht mehr darauf eingegangen, wer diese Option wählt muss danach selber schauen wie es weitergehen könnte.**

**ACHTUNG!**

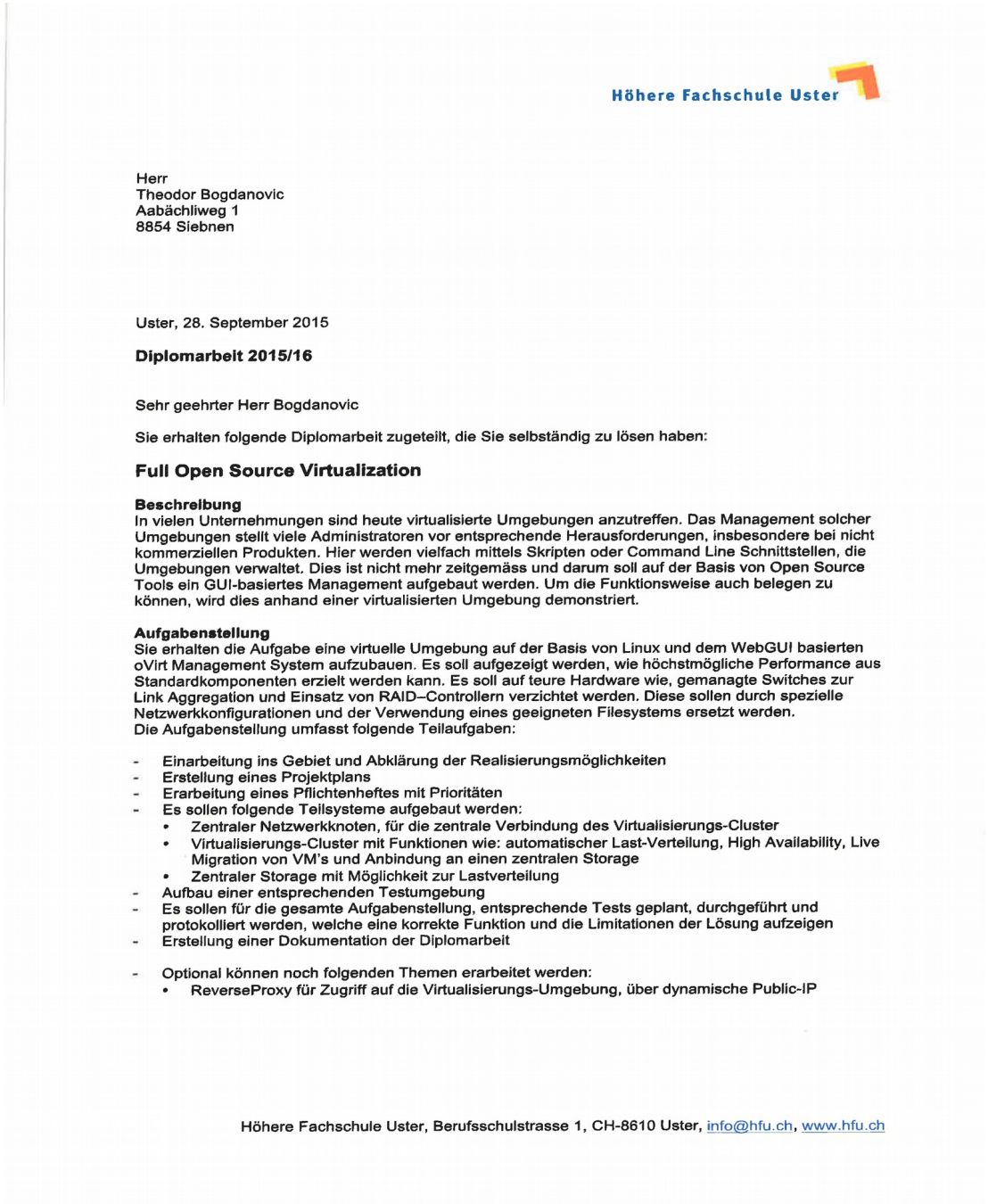
**Was hier drinnen steht wurde meist durch den Autor dieser Arbeit einmal angewendet und endete im Chaos, geistiger Verwirrtheit oder einer langen Suche über google um den Fehler zu finden. Den Anweisungen in diesen Feldern, sollte auf jeden Fall gefolgt werden!**

Kurzer Hinweis bezüglich strukturellem Aufbau dieser Dokumentation: Diese Dokumentation ist in drei Teilbereich gegliedert und zwar in ein Netzwerksegment, ein Virtualisierungssegment und ein Storagesegment. Diese Reihenfolge stellt einen logischen Konstruktionsweg dar, der sich auch in der Dokumentation widerspiegeln sollte. Da aber ein solcher Virtualisierungscluster als ein Element funktionieren soll, kann es vorkommen, dass einige Bereiche hinsichtlich dieser logischen Trennung dennoch miteinander verschmelzen.

## 3.3 Aufgabenstellung – Der eigentliche Auftrag

Der eigentliche Auftrag, wie er dem Studenten seitens der HFU gestellt wurde ist nachfolgend als Abbildung aufgeführt. Die Darstellung als Abbildung dient der Sicherheit des Auftrages, um Verfälschungen des Inhaltes auszuschliessen.

Seite 1:



Höhere Fachschule Uster

Herr
Theodor Bogdanovic
Aabächliweg 1
8854 Siebnen

Uster, 28. September 2015

**Diplomarbeit 2015/16**

Sehr geehrter Herr Bogdanovic

Sie erhalten folgende Diplomarbeit zugeteilt, die Sie selbständig zu lösen haben:

**Full Open Source Virtualization**

**Beschreibung**
In vielen Unternehmungen sind heute virtualisierte Umgebungen anzutreffen. Das Management solcher Umgebungen stellt viele Administratoren vor entsprechende Herausforderungen, insbesondere bei nicht kommerziellen Produkten. Hier werden vielfach mittels Skripten oder Command Line Schnittstellen, die Umgebungen verwaltet. Dies ist nicht mehr zeitgemäss und darum soll auf der Basis von Open Source Tools ein GUI-basiertes Management aufgebaut werden. Um die Funktionsweise auch belegen zu können, wird dies anhand einer virtualisierten Umgebung demonstriert.

**Aufgabenstellung**
Sie erhalten die Aufgabe eine virtuelle Umgebung auf der Basis von Linux und dem WebGUI basierten oVirt Management System aufzubauen. Es soll aufgezeigt werden, wie höchstmögliche Performance aus Standardkomponenten erzielt werden kann. Es soll auf teure Hardware wie, gemanagte Switches zur Link Aggregation und Einsatz von RAID–Controllern verzichtet werden. Diese sollen durch spezielle Netzwerkkonfigurationen und der Verwendung eines geeigneten Filesystems ersetzt werden.
Die Aufgabenstellung umfasst folgende Teilaufgaben:

- Einarbeitung ins Gebiet und Abklärung der Realisierungsmöglichkeiten
- Erstellung eines Projektplans
- Erarbeitung eines Pflichtenheftes mit Prioritäten
- Es sollen folgende Teilsysteme aufgebaut werden:
  - Zentraler Netzwerkknoten, für die zentrale Verbindung des Virtualisierungs-Cluster
  - Virtualisierungs-Cluster mit Funktionen wie: automatischer Last-Verteilung, High Availability, Live Migration von VM's und Anbindung an einen zentralen Storage
  - Zentraler Storage mit Möglichkeit zur Lastverteilung
- Aufbau einer entsprechenden Testumgebung
- Es sollen für die gesamte Aufgabenstellung, entsprechende Tests geplant, durchgeführt und protokolliert werden, welche eine korrekte Funktion und die Limitationen der Lösung aufzeigen
- Erstellung einer Dokumentation der Diplomarbeit

- Optional können noch folgenden Themen erarbeitet werden:
  - ReverseProxy für Zugriff auf die Virtualisierungs-Umgebung, über dynamische Public-IP

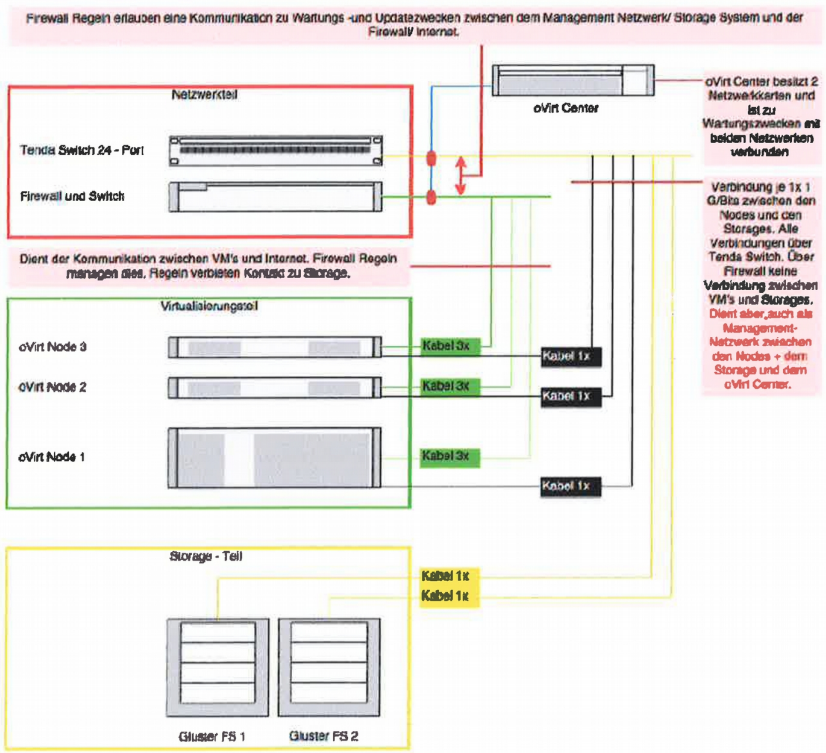
Höhere Fachschule Uster, Berufsschulstrasse 1, CH-8610 Uster, info@hfu.ch, www.hfu.ch

*Abbildung 1: Kopie des Auftrages Seite 1*

Seite 2:

Höhere Fachschule Uster

Das Schema dient zum besseren Verständnis der Aufgabe:




**Austeilung der Aufgabenstellung** **2. Oktober 2015**
**Abgabe der 2 Dokumentationen** **08. März 2016**

Freundliche Grüsse

Ruedi Kubli

Höhere Fachschule Uster, Berufsschulstrasse 1, CH-8610 Uster, info@hfu.ch, www.hfu.ch

*Abbildung 2: Kopie des Auftrages Seite 2*

## 3.4 Einführung ins Projekt

Der Aufbau eines Virtualisierungsclusters ist technisch gesehen der Aufbau einer konventionellen IT – Infrastruktur. Es müssen alle Ressourcen analysiert werden. Die entsprechenden Komponenten müssen so dimensioniert werden, dass sie der erwarteten Belastung standhalten und es muss natürlich auf die hardwareseitigen Ressourcenzuteilungen geachtet werden. Bei Virtualisierungs-Infrastrukturen ist noch zusätzlich auf das gleiche innerhalb der virtuellen Infrastruktur zu achten, wie bspw. virtuelle Netzwerke oder die Nutzung der Rechenressourcen wie CPU und RAM. Dieses Projekt soll ausschliesslich mit open source Produkten und kostengünstigen Standard Hardwarekomponenten realisiert werden. Dies bedeutet bspw. den Verzicht auf teure hardwaregestützte RAID – Controller oder die Verkablung mit 10 Gbit – Komponenten (Glas, Kupfer). Sollten Komponenten notwendig werden die nicht als „Standard“ Computer – Komponenten gelten, wie bspw. 4-Port 1Gbit Netzwerkkarten, so werden diese als kostengünstige Gebrauchtware organisiert.

Aufbauend auf den oben genannten Forderungen werden für die Kernkomponenten folgende drei Hersteller/ Produkte gewählt:

### 3.4.1 Das Netzwerksegment

**PfSense** soll an dieser Stelle die zentrale Rolle des Netzwerkkonten übernehmen, wobei hier auch Aufgaben der Bereiche Switching und Firewall abgedeckt werden können. Diese Aufgabe übernimmt ein altes IBM System, welches mit vier kostengünstigen 4-Port Netzwerkkarten (gebraucht, Preis pro Komponente 60 CHF.) bestückt ist und so mit den beiden On-board Ports auf ein Total von 18 Steckplätzen kommt. Hier kann also auf den Kauf eines teuren gemanagten Switches verzichtet werden, da sich die Ports in eine Bridge binden lassen und somit Switching - Funktionen replizierbar sind. Diese Konfiguration mag auf den ersten Blick nicht als vollwertiger Switch – Ersatz wirken, jedoch sind folgende Möglichkeiten damit realisierbar und werden auch so angewendet:

- Bildung von Link Aggregation (Protokoll LACP IEEE 802.3ad) um mehrere Links zu den grossen Maschinen zu generieren.
- Bildung einer Bridge, welche die autonomen Links (Pseudo- Links) zu einem „Gerät“ zusammenschliesst um nach aussen wie ein Switch zu wirken.
- pfSense erlaubt auch nach der Bildung von Pseudo- Devices wie LAGG – Interfaces (LACP) die Anwendung von eigenständigen Firewall – Rules für solche Interfaces.
- Bilden von VLAN's oder Tunk's ist ebenfalls auf jedes Interface anwendbar, belanglos ob das Device in einer Bridge eingebunden ist oder nicht. Die Konfigurationen bleiben allgemein autonom anwendbar. Jedoch wird hier auf die Segmentierung mittels VLAN verzichtet.
- Mittels des pfSense eigenen RRD Graphen – Systems ist auch ein Monitoring der Datenmengen möglich wie es bspw. auch bei Cisco – Geräten möglich wäre.

Betrachtet man alle Möglichkeiten die einem mit dieser Lösung zur Verfügung stehen, so kann man behaupten, dass dies ein vollwertiger Switch – Ersatz ist, welcher auch noch den Komfort bietet sämtliche Konfigurationsmöglichkeiten (Firewall und Switch) an einem Punkt zu vereinen.

Dieser pfSense „Switch“ dient nur der Kommunikation der VM's mit der Aussenwelt. Firewall – Regeln

werden an dieser Stelle die Kommunikation zwischen den virtualisierten Teilen zur Cluster – internen Kommunikation unterbinden.

Eine zusätzliche Komponente dieses Segmentes ist ein ungemanagter Switch (Tenda 24-port, 1Gbit) aus altem privatem Lagerbestand. Er ist der zentrale Knotenpunkt für die interne Kommunikation des Clusters. Dies beinhaltet die Storage- sowie die Management – Kommunikation und soll so den Cluster vom Public – Teil der Vm's isolieren. Firewall – Regeln verhindern die Kombinationen der Virtualisierungs- Nodes und der Storage – Nodes zur Aussenwelt. Nur zu Update – Zwecken wird diese spezielle „Update – Rule“ deaktiviert. Ansonsten soll sie die interne Kombinationen zur Aussenwelt strickt unterbinden.

### 3.4.2 Das Virtualisierungssegment

**oVirt 3.5:** Um allen Ansprüchen eines heute modernen Virtualisierungs- Systems gerecht zu werden, wurde oVirt in der Version 3.5 (zu Projektbeginn die aktuellste Version) gewählt. Es bietet alle Werkzeuge die heute in dieser Branche notwendig sind und bündelt sie in eine grösstenteils einfach zu bedienende Weboberfläche, welche auch optisch gut aussieht. Was auch noch positiv zu Buche schlägt ist die Tatsache, dass es die Grundlage des RedHat Virtualization Enterprise Servers ist. Man kann also von einer stabilen und Produktreifen Software sprechen.

oVirt ist ähnlich wie Vmware's  Lösung auf ein Management – Center (genannt oVirt – Engine) und eine unbestimmte Anzahl an Virtualisierungs- Konten ausgelegt. Abweichend von der heute üblichen Vorgehensweise, soll die oVirt – Engine nicht als VM innerhalb des Clusterters arbeiten, sondern als eigenständige Instanz auf einem Barebone System (Supermicro) installiert sein.

**ALTERNATIVE**

**Einst als testing markiert, wurde die Funktion Hosted-Engine-HA rückportiert von Version 3.6 auf 3.5 und erlaubt so die Engine als VM innerhalb des Clusters zu betreiben. Hierbei wird aber der Active – Zustand der Engine von den Nodes selber überwacht und im Fehlerfal eines Nodes die Engine einfach auf einem neuen Node neu gestartet.**

Für die Hosts wird das vom oVirt Projekt für jede Version erzeugte oVirt – Node Image verwendet. Diese Image basiert auf CentOS und ist eine stark abgespeckte Version einer ohnehin minimalen CentOS – Minimal – Instllation. Sie bietet den Komfort eines bereits fertigen und leicht installierbaren Systems. Sowohl das fertige Image wie auch eine CentOS Minimal – Installation bieten hinsichtlich Sicherheit die gleichen Features, wie aktives und vorkonfiguriertes SELinux oder eine per Default aktive Firewall. Jedoch ist das Image klar zu bevorzugen, da es noch stärker abgespeckt ist und bspw. auf eine Vielzahl an Userland Tool wie **adduser** (neu User unter Linux anlegen) verzichtet. Dies macht das Image einerseits etwas sicherer und reduziert den notwendigen Diskspace auf das absolute Minimum.

**ACHTUNG!**

**Das fertige Image schränkt der Sicherheit halber den Zugang zur reinen Command Line absolut ein, nur die Konfigurationskonsole ist erreichbar. Möchte man zusätzliche Software installieren wie bspw. einen Zabbix – Agent, so ist dies nicht möglich!**

Der virtualisierungs- spezifische Hardware – Teil hat dabei folgende Konfiguration:

- **Subermicro Barebone System:** Hierauf ist die oVirt – Engine 3.5 installiert. Der Server ist mit einem 1Gbit – Link an den ungemanagten Switch angeschlossen, wobei hier keine netzwerkseitige Ausfallsicherheit als notwendig angesehen wird. Link Aggregation würde auch keinen Sinn machen, da ein 1Gbit – Link für die Management – Funktionen völlig ausreicht.
- **Dell PowerEdge „Occasion Gerät“:** Auf diesem Gerät ist ein fertiges oVirt – Node -Image installiert (Version 3.5). Zusätzlich zu vier On-board Etherports wurde noch eine Vierport – Netzwerkkarte installiert. Diese Maschine zählt zu den grossen Einheiten des Clusters und wurde mit drei Ports (LACP) an pfSense angebunden um den VM's Netzzugang zu ermöglichen. Zwei Etherports sind in eine Link Aggregation gebunden, nutzen aber den Linux softwareseitigen Bonding Mechanismus Mode 2 (Balanced XOR), um die interne Cluster – Kommunikation zu ermöglichen. Diese Konfiguration ist notwendig, da die Links auf einen ungemanagten Switch zeigen, welcher Link Aggregation nicht unterstützt.
- **Subermicro Barebone System:** Dieser Server zählt auch zu den grossen Maschinen und hat eine identische Konfiguration wie der Dell PowerEdge.
- **2x Fujitsu Primergy:** Diese beiden Server sind mit je einem Intel Xeon E3 Prozessor und je 8 GB RAM als mittelstarke Maschinen einzustufen. Ihre Konfiguration hinsichtlich Cluster – Kommunikation ist wie bei allen Node – Konfiguration identisch. Da sie aber vom Leistungsumfang eher in die Sparte bescheiden einzustufen sind, ist mit keiner grossen VM Anzahl zu rechnen. Aus diesem Grund wird auf eine Link Aggregation mit je drei Links im LACP – Verbund verzichtet. Sie besitzen daher nur je einen 1Gbin – Link zu pfSense.

Die oVirt spezifischen Konfigurationen wie Hostnamen, interne Strukturen wie Datacenter oder die verwendeten Cluster sind zu umfangreich für diesen Bereich. Sie werden unter Punkt 8 „Realisierung“ genauer betrachtet.

### 3.4.3 Das Storagesegment

Das Storage System ist die aus Sicht des Autors dieser Arbeit, wohl das wichtigste Kernstück. Dies beruht auf der Tatsache, dass die abgelöste Xenserver Lösung eine zu dieser Zeit aus finanziellen Gründen „günstigere“ Lösung darstellte. Die Festplatten, die nun in der aktuellen Lösung arbeiten, waren zuvor in einem Virtualisierungs- Node untergebracht und mussten mühsam von Hand integriert werden. Dies erschwerte auch das Updaten der Vorgängerlösung, was zur Folge hatte, dass auch keine gemacht wurde.  Im neuen Design ist das Storage System ein autonomer Teil, bestehend aus folgenden Komponenten:

- **2x HP Proliant Microservern der 8-ten Generation:** Diese Geräte bieten sich gerade zu an als NAS oder eine sonstige Storage Variante zu arbeiten. Denn sie sind extrem preisgünstig (je nach Ausstattung), sie bieten bis zu vier Festplatten – Slots (leider nicht Hot-Swap fähig) an und es sind echte Serversysteme. In diesem Projekt werden zwei Microserver in einen Verbund gespannt, welcher mittels der open source Software GlusterFS (Background Daemon) ihre Zielverzeichnisse als einen gemeinsamen Gluster – Mountpoint anbieten. Die Zielverzeichnisse befinden sich auf einem Software – RAID – Verbund des Typs 5 mit je vier Disks. Dieser RAID – Level liefert das beste Ausfall – zu – Speichervolumen Verhältnis, welches mit vier Disks erreicht werden kann. Jedoch ist der Volumenverlust basierend auf dem RAID 5 erheblich. Bei **4** Disks pro Server mit einer Speicherkapazität von **1,8 TB** pro Disk, stehen nach dem bilden des RAID gerade einmal 5,5 TB zur Verfügung.

Da GlusterFS mittels Replizierung der Volumes auf mehrere Nodes nahezu jeden klassischen RAID Level nachbilden kann, wäre es sinnvoll diese Technik der Ausfallsicherheit auch zu nutzten. Da aber nur zwei Nodes zur Verfügung stehen und der Diskspace durch den eingesetzten RAID – Level massiv beschnitten wird, wurde auf eine Ausfallsicherheit zu Gunsten eines höheren Diskspaces verzichtet. Auf diese Weise kann zwar nicht der Ausfall eines Nodes kompensiert werden, jedoch wenigstens noch der Ausfall einer Disk pro Node.

Um den Diskspace zu maximieren und die doch etwas leistungsarmen Microserver zu schonen, wird ein klassischer GlusterFS Mountpoint des Typs Distributed gewählt. Dies bedeutet, dass die Disk – Images welche von den virtualisierungs- Nodes für die VM's erzeugt werden, als ein einziges File nach dem Zufallsprinz stets auf den einen oder den anderen Storage – Node schreiben werden. Durch diese Konstellation konnte ein Mountpoint erschaffen werden, welcher eine ungefähre Speicherkapazität von 11 TB besitzt.

Ein zweiter GlusterFS Mountpoint, welcher aber nach aussen hin als NFS – Mountpoint ersichtlich ist, dient dem ganzen Cluster als ISO_Storage_Domäne. Diese spezielle Gluster – zu – NFS Kombination wird dadurch ermöglicht , dass GlusterFS native NFS Protokoll – Unterstützung implementiert hat. So nutzen zwar Master Storage und ISO Domäne den gleichen RAID 5 – Verbund, jedoch ist dies ein absolut tragbarer zustand, da bei 11 TB genügend Speicherplatz für Images und ISO's vorhanden ist.

### 3.4.4 Allgemeine schematische Ansicht

Des besseren Verständnisses halber und um einen ersten Eindruck bezüglich Aussehen und Struktur des Aufbaus zu ermöglichen, folgen nun zwei Schemen mit den primären Sichtweisen des Projektes.

**Anbindung des Management- und Storage – Teils:**

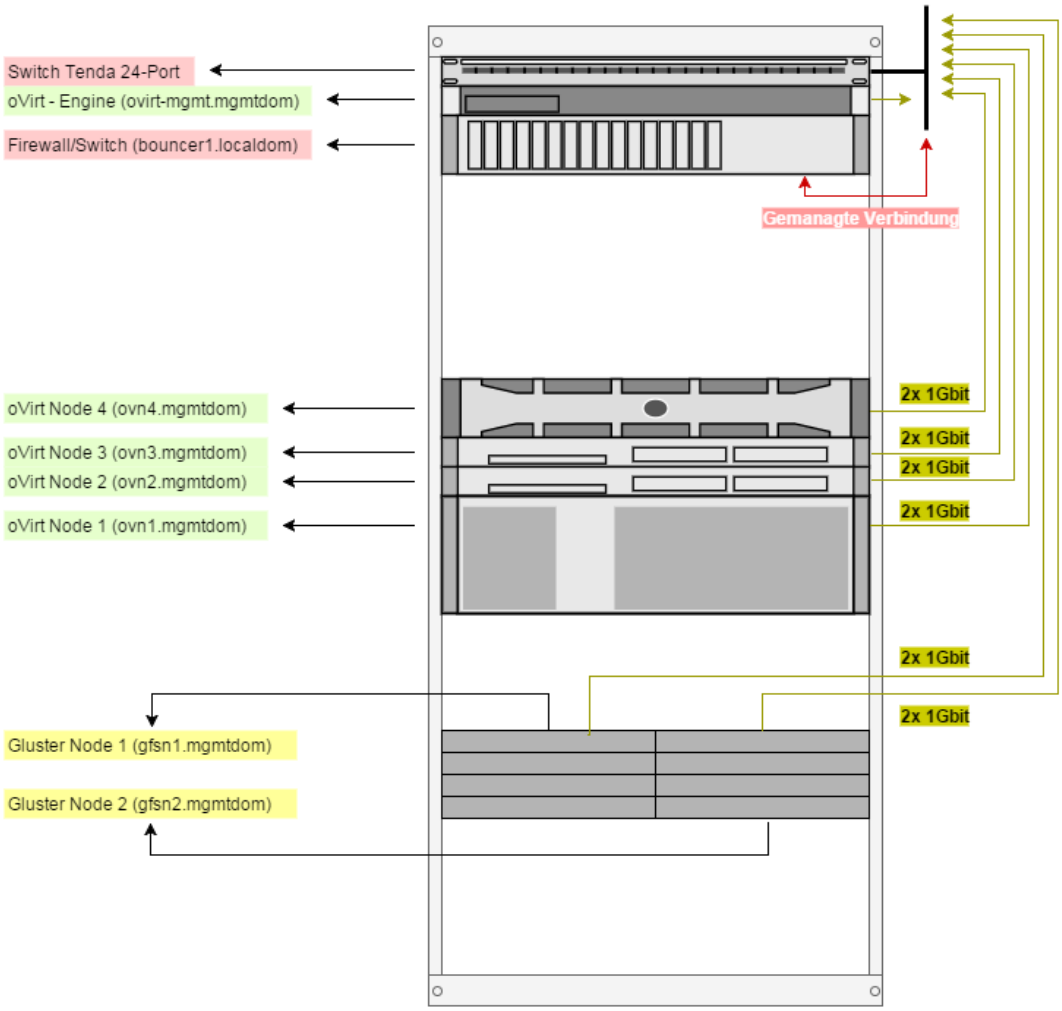



*Abbildung 3: Schematischer Hardware – Aufbau aus Sicht des Management- und Storage – Teils*

In dieser Abbildung ist deutlich zu erkennen wie die gesamte Verkablung für den Management- sowie für den Storage – Teil isoliert auf den Tenda 24-Port Switch geführt wird. Dabei wird stets eine Link Aggregation nach Balanced XOR verwendet. Zweimal pro Gerät resultiert hierbei aus der kleinsten zur Verfügung stehenden Menge, welche die beiden Microserver definieren mit je nur zwei Anschlussmöglichkeiten. Hier rüber kann der gesamte Management- und Storagezugriff ausfallsicher abgewickelt und zugleich die Gesamt – Bandbreite um den Faktor zwei erhöht werden. Der rot markierte Bereich „Gemanagte Verbindung“ ermöglicht der oVirt – Engine den bidirektionalen Zugriff ins Internet, wobei Regelsätze in der Firewall verhindern dass die Restliche Infrastruktur nach aussen kommunizieren kann. Hier sieht man auch deutlich die dezentrale Arbeitsweise von GlusterFS. Die Nodes arbeiten zwar als gemeinsamer Verbund, jedoch ist keine spezielle Verkablung untereinander

notwendig, sie verhalten sich nach aussen wie zwei autonome Hosts.

**Anbindung des Virtual Enviroments:**

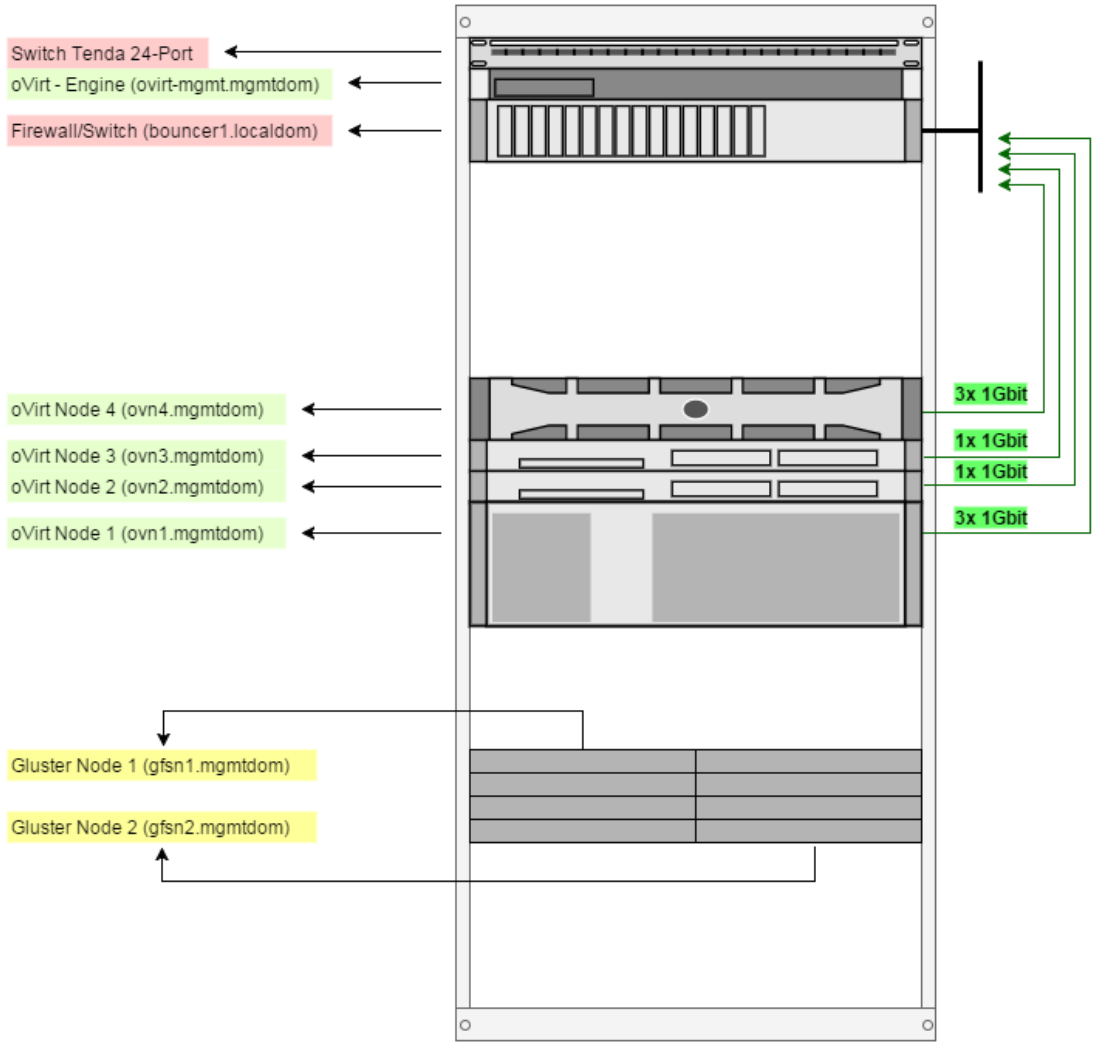



*Abbildung 4: Schematischer Hardware – Aufbau aus Sicht des Virtual Enviroments*

In dieser Ansicht ist zu erkennen, dass keine physische Verbindung zwischen den virtuellen Maschinen und dem Cluster – Netzwerk besteht. Ein Routing durch pfSense wäre technisch denkbar, jedoch gibt es kein vorstellbares Szenario wo dies Sinn machen würde. Daher können simple Regeln in pfSense definiert werden, welche bidirektional „**deny all**“ lauten.

## 3.5 Das Vorgehen in groben Zügen

An dieser Stelle folgt eine grobe Zusammenfassung eines möglichen Projektablaufes. Diese Zusammenfassung ist der besseren Darstellung halber in Phasen aufgeteilt, welche versuchen den Projektablauf in seinem zeitlichen Ablauf zu widerspiegeln. Da an diesem Projekt in jeweils kurzen Zeiteinheiten, verteilt über einen langen Zeitraum gearbeitet werden soll, kann es vorkommen dass einige Phasen in ihrer Reihenfolge schwer nachvollziehbar sind.

Einige Teile wie bspw. der Umbau des Racks oder die Sicherung der VM's des alten Clusters werden hier aus Platzgründen nicht spezifisch erwähnt.

### 3.5.1 PHASE: Initialisierung

Nach Erhalt der Auftragsbestätigung seitens der HFU, wurden alle vorhandenen und notwendigen Parameter nochmals überprüft. Nach positiver Überprüfung der Parameter und Bedingungen, wurde basierend auf dem Aufgabenplan des Antrages versucht einen ersten Grobentwurf für das Zeitmanagement zu erstellen. Hierbei wurde versucht die notwendige Zeit, welche für die Realisierung des Projektes veranschlagt wurde, in Einklang mit der verfügbaren Zeit zu bringen.

### 3.5.2 PHASE: Kurzer Umbau

Noch vor Beginn der eigentlichen Planung, welche sich auf die Vor- und Hauptstudien stützt, gab es noch kleinere definitive Arbeiten die noch zu erledigen waren. Diese Arbeiten beziehen sich hauptsächlich auf das Umbauen des Racks, um einerseits die fehlenden Träger herzustellen und zu montieren und andrerseits um einen minimal notwendigen Teil der Verkablung neu zu erstellen und zu verlegen.

### 3.5.3 PHASE: Evaluierung des Grundaufbaus (gestützt auf Vorstudie)

Die hier vorgenommen Überlegungen beziehen sich stark auf den eigentlichen Grundaufbau der Hardware oder der elementaren Infrastruktur. Hier ohne genau durchdachten Plan vorzugehen, kann bei möglichen nachträglichen Design – Änderungen eine fatale Down – Time des Systems zur Folge haben. Aufgrund der relativ klaren Vorstellungen des Endergebnisses, aber auch der klaren Vorgaben seitens der zur Verfügung stehenden Mittel, bedarf es nur der Evaluierung zweier ungewisser Punkte. Diese beiden nachfolgend erklärten Punkte werden entweder mit einer simplen Gegenüberstellung oder mittels einer Nutzwertanalyse auf je ein Ergebnis minimiert:

- **pfSense oder opnSense:** Als langjähriger User von pfSense tendiert der Autor klar zu pfSense. Somit kommt im Grunde keine Linux - basierte Firewall – Distribution in frage, da die Performance eines FreeBSD Network Stack ohnehin nicht erreicht werden kann. Bei opnSense sieht dieser aber anders aus, da es nicht nur eine FreeBSD Distribution ist, sonder auch ein Fork von pfSense. Zwar ist opnSense noch nicht so ausgereift wie pfSesnse, jedoch sprechen zwei Punkte klar für heute bereits notwendige Überlegungen:
    - Erstens die in den letzten Versionen von pfSense immer häufiger auftretenden und überaus nervigen Versionsdiskrepanzen der Pakete.
    - Und zweitens, die in kurzen Zeitintervallen immer grösser werdende Community von opnSense, was im open source Bereich als klarer Indikator für eine stabile Zukunftslösung

ist. Denn selbst der Entwickler der seit kurzem aufgegebenen m0n0wall Firewall rät klar zu opnSense.

- **RAID 5 oder 10:** Die Möglichkeit einer Ausfallsicherheit auf Host – Ebene mittels der GlusterFS eigenen Replizierungs- Möglichkeiten wäre wohl die beste Lösungen, jedoch fehlen die Rechner und natürlich die finanziellen Mittel schlichtweg dafür. So muss sich der Autor dieser Arbeit mit einer Disk- basierten Ausfallsicherheit zufrieden geben. Die Frage ist jedoch, welcher RAID – Level wäre der geeignetste hierfür. Soll auf absolute Sicherheit gesetzt werden mit dem Nachteil eines relativ hohen Diskspace Verlustes oder soll das Ausfallrisiko zu Gunsten eines höheren Diskspaces auf eine Disk beschränkt werden.

Die Klärung dieser beiden Fragen ist unter Punkt 6 „Vorabklärungen und Analysen“ ersichtlich.

### 3.5.4 PHASE: Evaluierung des Produktivsystems (gestützt auf Hauptstudie)

Auch in der Hauptstudie sollen elementare Fragen geklärt werden, welche das fertige Produkt so formen, dass es nach den Kriterien des Auftrages funktioniert. Ähnlich wie in der Vorstudie sollen auch hier die Fragen geklärt werden, die eine Konfiguration nur in eine Richtung ermöglichen und wo nachträgliche Änderungen mit erheblichem Aufwand, gekoppelt mit langen Down – Timephasen verbunden sind. Die unten aufgeführten Fragen welche einer Evaluierung bedurften sind stark Software- oder Konfigurationslastig. Bei einer so umfangreichen Lösung wie oVirt gibt es eine Vielzahl an möglichen Konfigurationsüberlegungen und Design – Entscheidungen die eine Nutzwertanalyse erfordern würden. Jedoch sind die meisten dieser Überlegungen bzw. Möglichkeiten dynamischer Natur und können selbst im Betrieb ohne weiteres geändert werden. Die unten Aufgeführten beiden Punkte bedurften aber einer Abklärung noch vor Inbetriebnahme:

- **oVirt Version 3.5 oder 3.6:** Open source Produkte die durch eine Community geleitet werden und lange Planungsphasen vertragen sich meist nicht gut. So geschah es auch bei diesem Projekt, dass nach langer Vorbereitung eine neue Version von oVirt als Produktiv freigeschaltet wurde. Da dies auch noch während der Planungsphase geschah, sah sich der Autor dieser Arbeit mit der gossen Frage konfrontiert, welche Version er nun wählen sollte.
- **Ein grosser oder doch separierte Cluster:** oVirt zählt zu den Engines, welche nicht einen virtuellen Pseudoprozessor zur Verfügung stellen, sondern den realen Prozessor mit Hilfe der Ring -1 Erweiterung (Hardware basierte Virtualisierung) durchreichen. Da aber im Cluster drei unterschiedliche Intel Xeon Prozessor – Familien vorhanden sind, muss einem Cluster klar gesagt werden, welche Familie er den VM's durchreichen soll. Man kann dies umgehen indem man einfach die kleinste gemeinsame Prozessor – Familie als Standard für alle Maschinen definiert. Dies bedeutet aber den zur Verfügung stehenden Prozessor – Befehlssatz unnötig für leistungsstärkere Prozessoren zu verringern. Das Resultat wäre ein hoher Overhead seitens des Prozessors.

Die Klärung dieser beiden Fragen ist unter Punkt 7 „Vorabklärungen und Analysen“ ersichtlich.

Bis hier hin wird es sicherlich einige Meilensteine geben, jedoch soll dieser Punkt besonderes hervorgehoben werden. Denn ab hier sind sämtliche Fragen geklärt und es kann mit der Realisierung begonnen werden.

### 3.5.5 PHASE: Mögliche Umsetzung der Realisierung

Nachfolgend sind in einer stark verkürzten Version die wichtigsten Schritte der Realisierung unter eigenständigen Nummerierungen aufgeführt.

#### 3.5.5.1 Installation und Konfiguration von pfSense

Ein Teil dieser Arbeit wurde noch vor dem offiziellen Auftrag erledigt, der zentrale Netzwerkknoten des Clusters welcher zugleich der des privaten Segmentes des Autors ist. Diese Zeit wird auch direkt im Projektplan als Effektivzeit erfasst.

Im ersten Schritt wurde der IBM Server entstaubt und mit vier Intel – Netzwerkkarten mit je vier Ports (alles günstige Occasions Hardware) bestückt. Anschliessend wurde die aktuellste pfSense Version installiert. Dann wurden die entsprechenden Interfaces in die jeweiligen Link Aggregation zusammengefasst und eine Bridge über alles gebildet. Kurze undokumentierte Tests wurden an dieser Stelle ebenfalls vollzogen. Feinkonfigurationen wie das definieren von Regeln werden im Verlauf der Arbeit kontinuierlich ergänzt.

#### 3.5.5.2 Komponenten platzieren und verkabeln

Der ungemanagte Switch wird positioniert und die meisten Netzwerkkabel werden verlegt und entsprechend mit RJ45 – Steckern bestückt. Dies ist möglich, da die Position der Server im Rack fix ist und aufbauend auf der Vorplanung auch klar ist wie sie verkabelt werden müssen.

#### 3.5.5.3 Einrichten des Storage (GlusterFS)

Die beiden HP Proliant Microserver werden mit den entsprechenden Festplatten bestückt und mit der Installation der zu diesem Zeitpunkt aktuellsten Version von CentOS 7 wird begonnen. Installation und Konfiguration der notwendigen Software wie Gluster oder VDSM. Erste kleine Funktionstest mit etwas Dokumentations- unabhängiger Spielerei.

#### 3.5.5.4 Installation der oVirt – Node – Images

Das fertige oVirt – Node – Image wird auf alle vier Virtualisierungsnodes installiert. Entsprechend wird auch eine Engine- unabhängige Minimalkonfiguration vorgenommen wie bspw. ein Netzwerk – Interface konfiguriert. So können bequem über eine SSH Verbindung erste Eindrücke gewonnen werden.

#### 3.5.5.5 Installation der oVirt Management Engine

Auf einem bis dato ausser Betrieb stehenden kleinen Supermicro Barebone System wird eine CentOS 7 Minimalinstallation aufgespielt und die entsprechende Version 3.5 der oVirt Engine aufgesetzt. Dies kann aufgrund oVirt- seitiger Rückportierungen einiger Funktionen von Version 3.6 zu 3.5 mit erheblichen Schwierigkeiten verbunden sein.

#### 3.5.5.6 Vorsichtige Integration Teil 1

Wenn alle Komponenten in Einzelarbeit konfiguriert sind, kommt die Zeit der Zusammenführung. In einem ersten Schritt soll der Storage in die oVirt – Engine Integriert werden. Zu diesem Zeitpunkt noch mit Single – Interface Konfiguration. Dieser Punkt der Vereinigung zweier für den Autor noch unbekannter Komponenten ist absolutes Neuland, es wird aber mit wenigen Problemen gerechnet.

#### 3.5.5.7 Vorsichtige Integration Teil 2

An dieser Stelle ist es an der Zeit die restlichen Komponenten in die Engine zu integrieren. Die vier Virtualisierungsnodes werden ebenfalls mit nur einem Management Interface eingebunden. Das fertige oVirt – Node – Image sollte sich ohne weitere Schwierigkeiten einbinden lassen. Ein Unterschied zwischen dem fertigen Image und den CentOS – Installationen ist technisch gesehen auf den Unterbau bezogen nicht feststellbar, jedoch wird resultierend aus kleineren Vortests mit minimalen Schwierigkeiten bezüglich der Einbindung gerechnet, da hier Versionsdiskrepanzen bezüglich der VDSM – Abstraktionsschicht vorhanden sind.

#### 3.5.5.8 Konfiguration der oVirt spezifischen Infrastruktur

Wenn alles zusammengeführt ist und einige kleinere Tests persönlicher Natur stattgefunden haben, kann die Infrastruktur konfiguriert werden. Dies umfasst folgende Punkte, welche aus den Vor- und Hauptstudien hervorgingen:

- Konfiguration der Link Aggregation, Management -und Virtualisierungsseitig.
- Bilden des Datacenters
- Bilden der Cluster – Gruppen und definieren der zu nutzenden Prozessor – Familie
- Zuteilen der Speicher- und ISO – Domänen
- Bilden und verteilen der Datacenter weiten Netzwerke (VMnet1)

Ab diesem Punkt wird ein zweiter erwähnenswerter Meilenstein dieses Abschnittes erreicht, den ab diesem Punkt kann virtualisiert werden. Diese hier stattfindenden Virtualisierungstest sind aber persönlicher Natur und werden nicht dokumentiert.

#### 3.5.5.9 Beginn Testreihe 1

Ab diesem Zeitpunkt ist die Konfiguration zwar noch nicht fertig, jedoch können erste Tests vollzogen werden. Nachfolgend einige Punkte die Teil der Testreihe sein könnten:

- Ausfall von Netzwerk Interfaces
- Ausfall von Hosts aus der Virtualisierungsebene
- Ausfall der Management – Engine
- Verifizierung der Link Aggregation des Virtual Enviroments
- Verifizierung der Link Aggregation des Storage und Management Teils

Diese Tests sollten an dieser Stelle stattfinden, da sie zu möglichen Schäden am bestehenden Aufbau führen können und eine Reimplementierung des Aufbaus zur Folge hätten. Sollte dieser Fall eintreten, so kann der Neuerstellungsaufwand auf ein technisches Minimum reduziert werden.

#### 3.5.5.10 Beginn des Fine – Tuning des Clusters

Nach erfolgreichen Grobtests kann an dieser Stelle mit den Feinkonfigurationen begonnen werden. Nachfolgend ein kleiner Auszug aus theoretisch notwendigen Konfigurationen:

- Einspielen von weiteren ISO – Images mit diversen Betriebssystemen.
- Erstellung der ersten Vorlagen aufbauend auf einer Debian 8.2 Rohinstallation mit grafischem LXDE Desktop.
- Erstellung von QoS für Speicher, Netztwerk und CPU Nutzung (Datacenter weit)
- Bildung von erster Affinitätsgruppe
- Erweitern der vorhandenen Instanzentypen

#### 3.5.5.11 Beginn Testreihe 2

Hier können nun die oben genanten Konfigurationen des Fine – Tuning getestet werden. Da dies rein softwareseitige Tests sind, muss mit keinen Schäden hinsichtlich der Infrastruktur des Clusters gerechnet werden. Die unter Punkt 3.5.5.9 und 3.5.5.11 erwähnten Tests sind unter Punkt 9 „Tests“ detailliert dokumentiert.

Dies war eine kurze Zusammenfassung des Vorgehens bei der Realisierung des Projektes von den ersten Studien bis zu den Tests. Die hier nicht spezifisch erwähnten Punkte wie bspw. kontinuierliche Erstellung der Dokumentation etc. wurden der Kompaktheit halber dieses Abschnittes weggelassen.

# 4 Projektplan

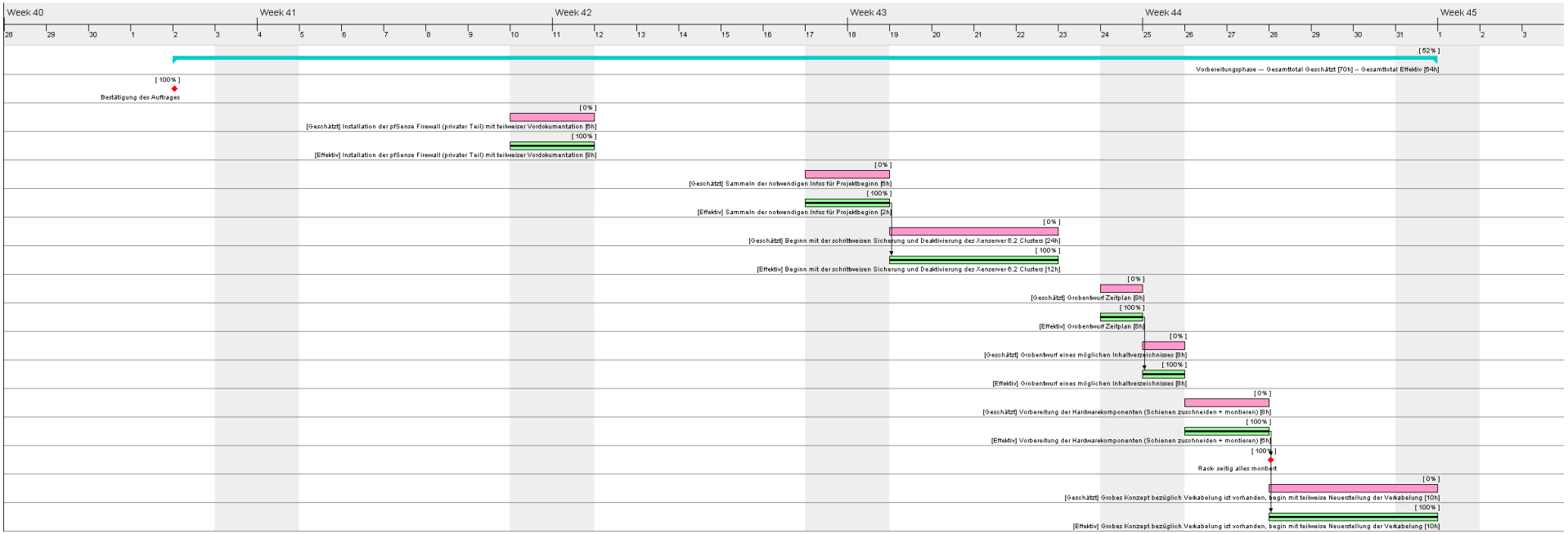
Auf den nachfolgenden Seiten ist der Projektplan mit direktem Vergleich zwischen der geschätzten und der effektiven Zeit aufgeführt. Das Gantt – Diagramm welches für die Darstellung verwendet wurde zeigt jeweils die Tage in denen gearbeitet wurde an. Am Ende des Beschreibungstextes ist jeweils in eckigen Klammern „**[ ]**“ der effektive Stundensatz an gearbeiteter Zeit erfasst. Leichte Abweichungen innerhalb der gearbeiteten Tage sind möglich, jedoch lässt sich die Stundenzahl meist symmetrisch durch die Anzahl eingetragener Tage teilen, um die effektiven Tagesstunden zu erhalten.

Der besseren Darstellung halber wurde der Projektplan in fünf Monate unterteilt, welche teilweise bei einer grösseren Menge an Einträgen in einzelne Teile gesplittet wurden. Die Verteilung sieht dabei wie folgt aus:

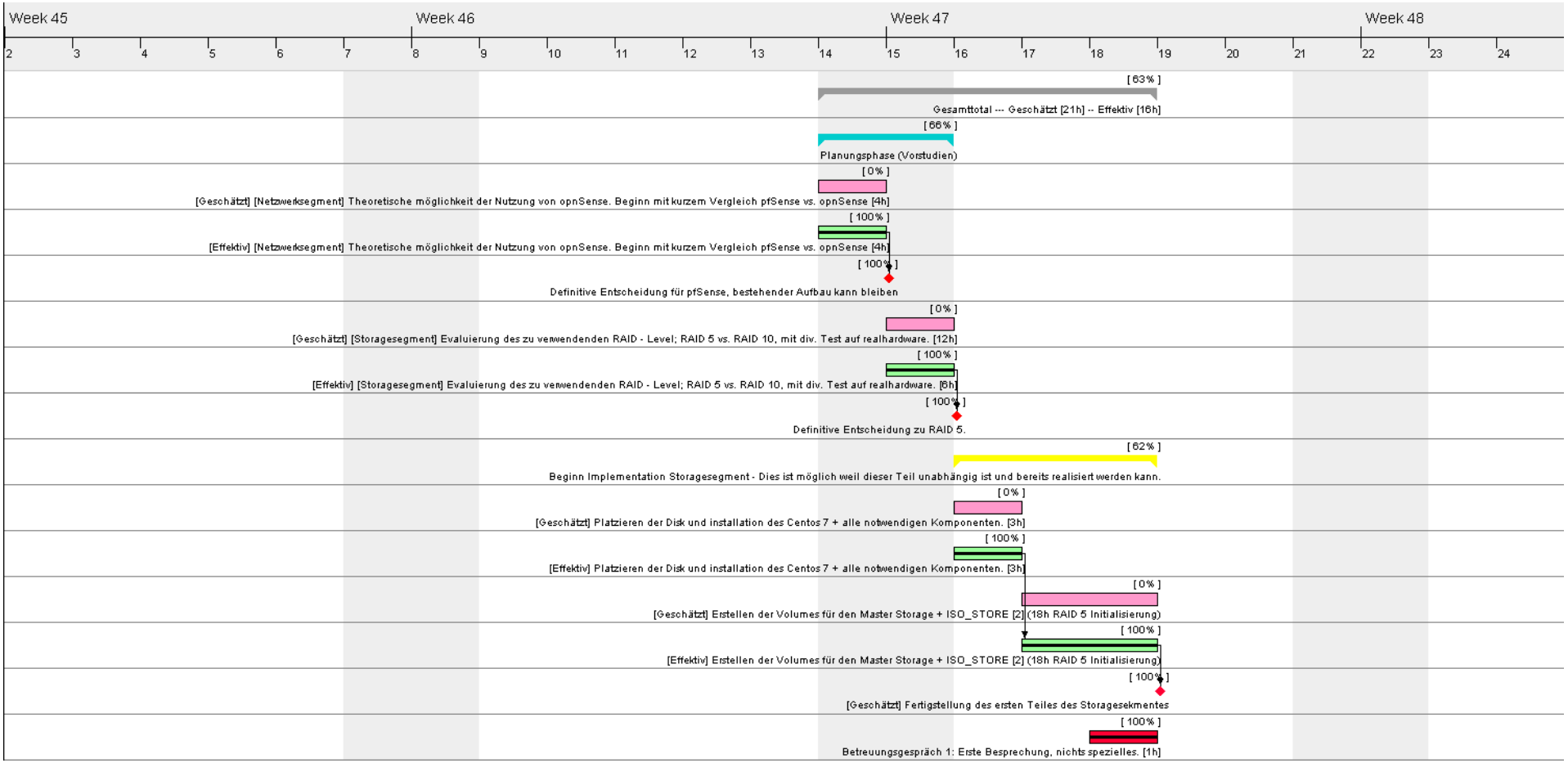
- Oktober
- November
  - Teil 1
  - Teil 2
- Dezember
- Januar
  - Teil 1
  - Teil 2
- Februar
  - Teil 1
  - Teil 2

Hierbei wurde eine Effektivzeit von **251,5 Stunden** benötigt um das Projekt zu realisieren und zu dokumentieren. Die geschätzte Zeit betrug **278,5 Stunden**, was auf die teilweise grosszügigen Reserven zurückzuführen ist, welche Standardmässig seitens des Autors bei Projekten ähnlicher Grösse hinzugefügt werden. Diese Reserven beziehen sich auf Punkte, welche seitens des Autors neu waren und wo eine saubere und schnelle Umsetzung nicht sicher war.

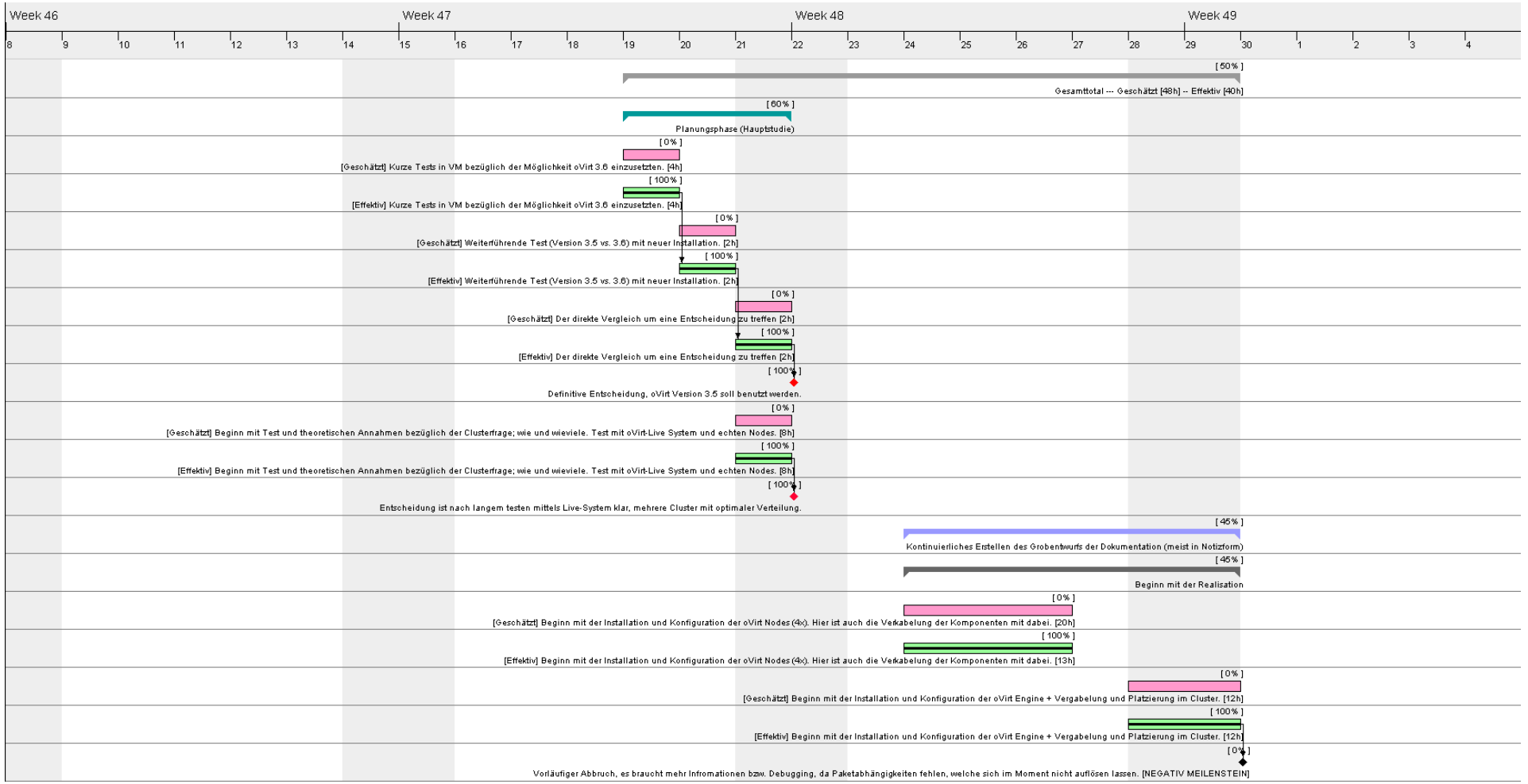
## 4.1 Projektplan des Monats Oktober



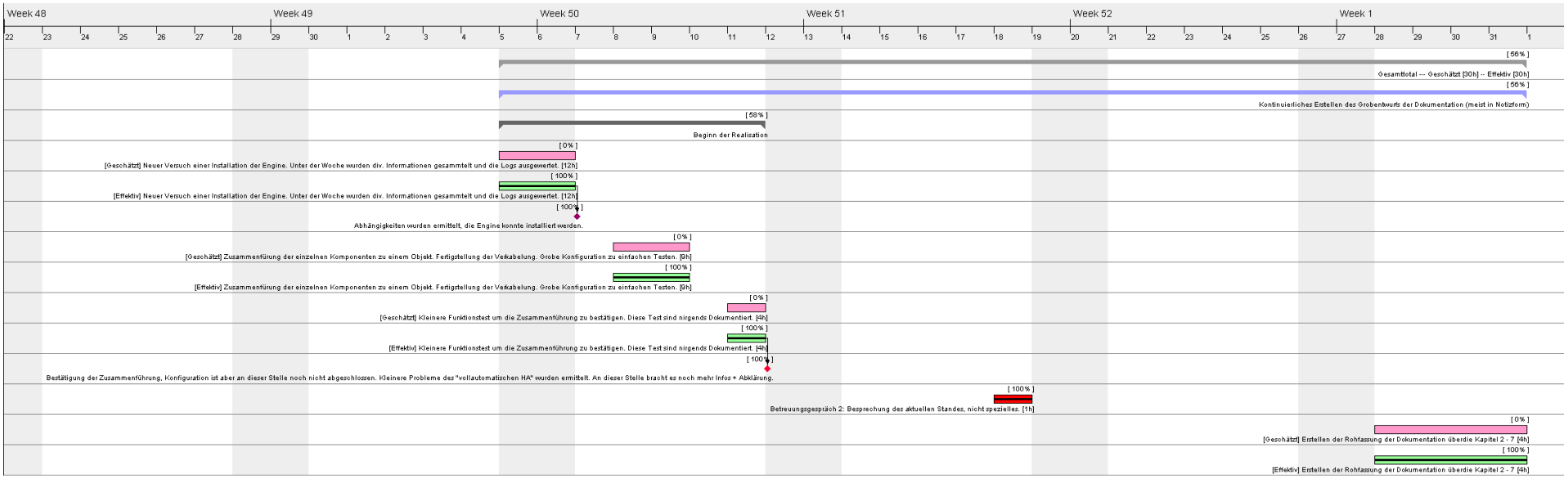
## 4.2 Projektplan des Monats November Teil 1



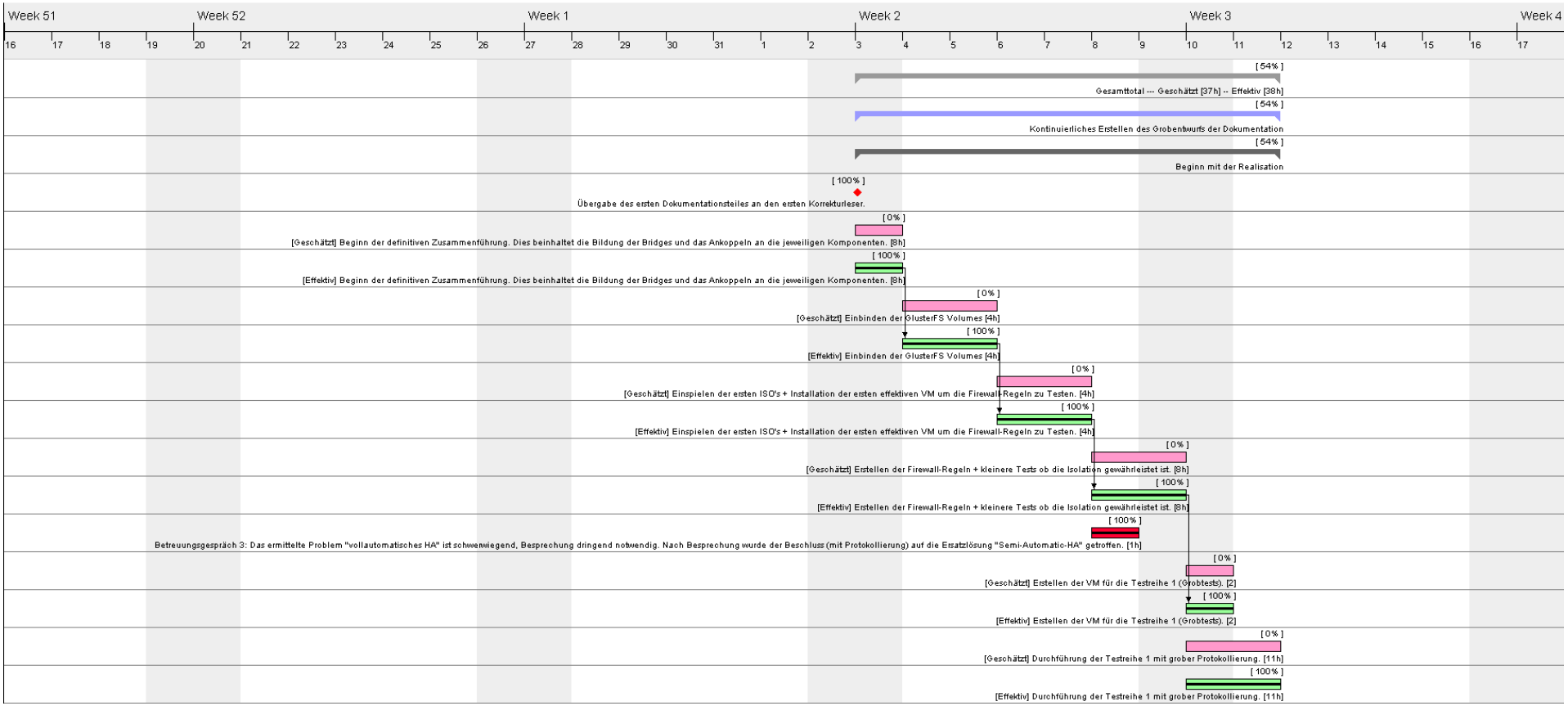
## 4.3 Projektplan des Monats November Teil 2



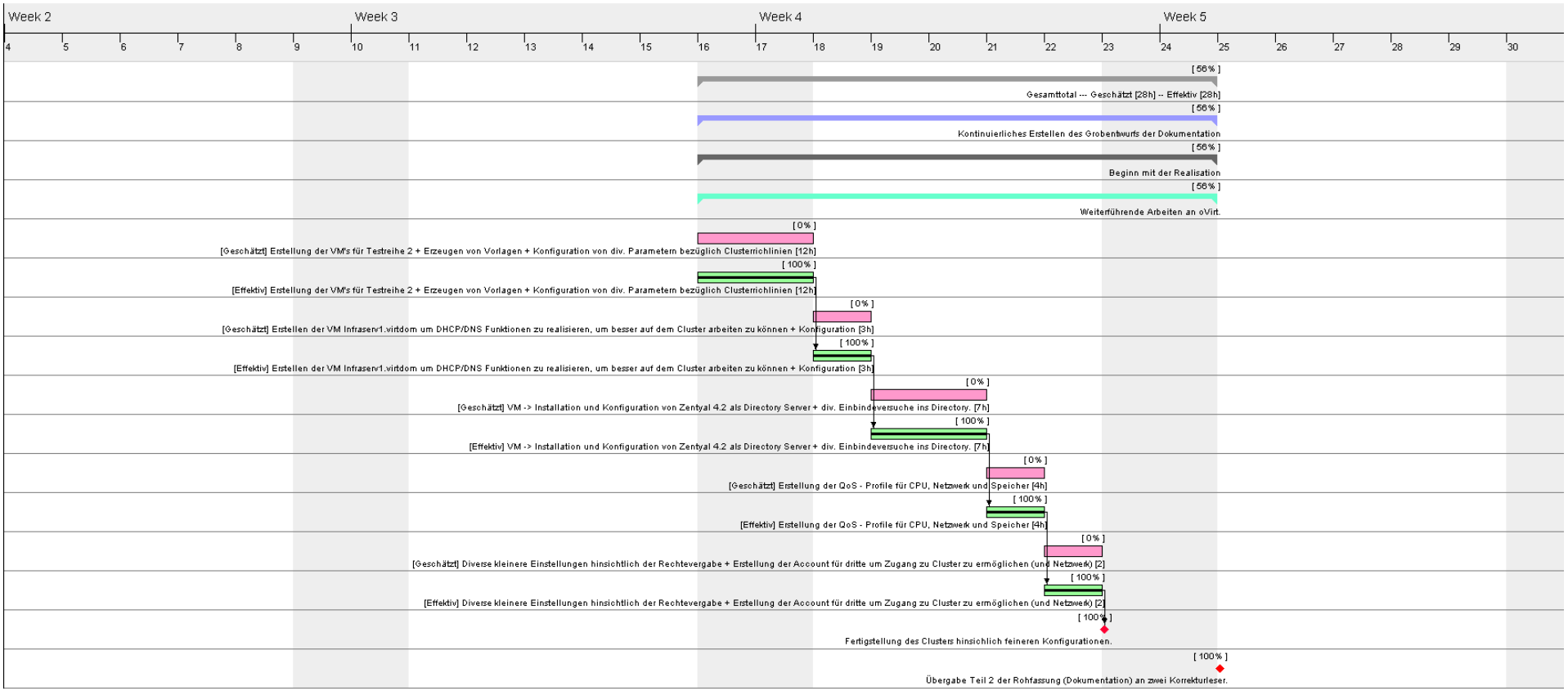
## 4.4 Projektplan des Monats Dezember



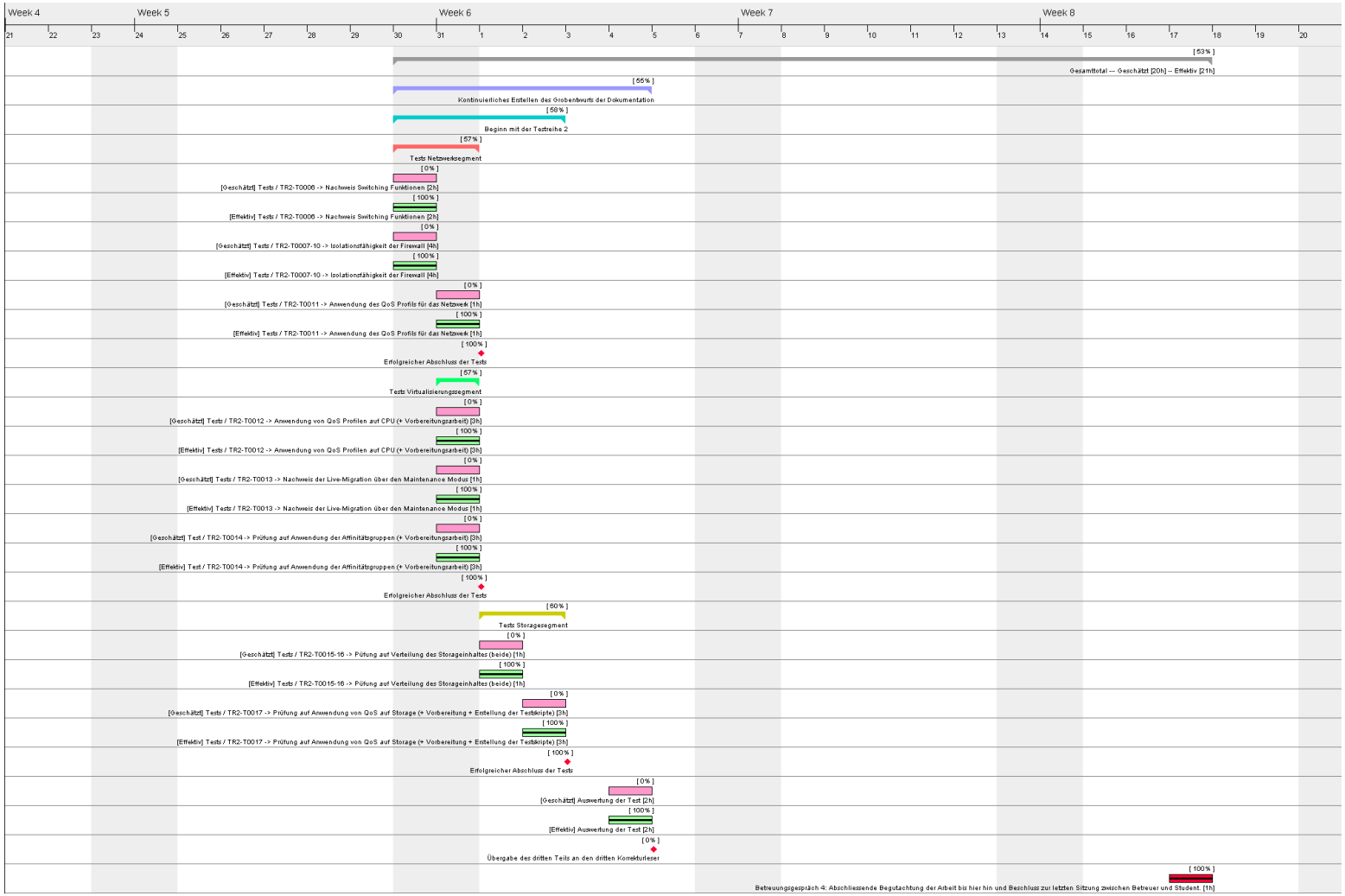
## 4.5 Projektplan des Monats Januar Teil 1



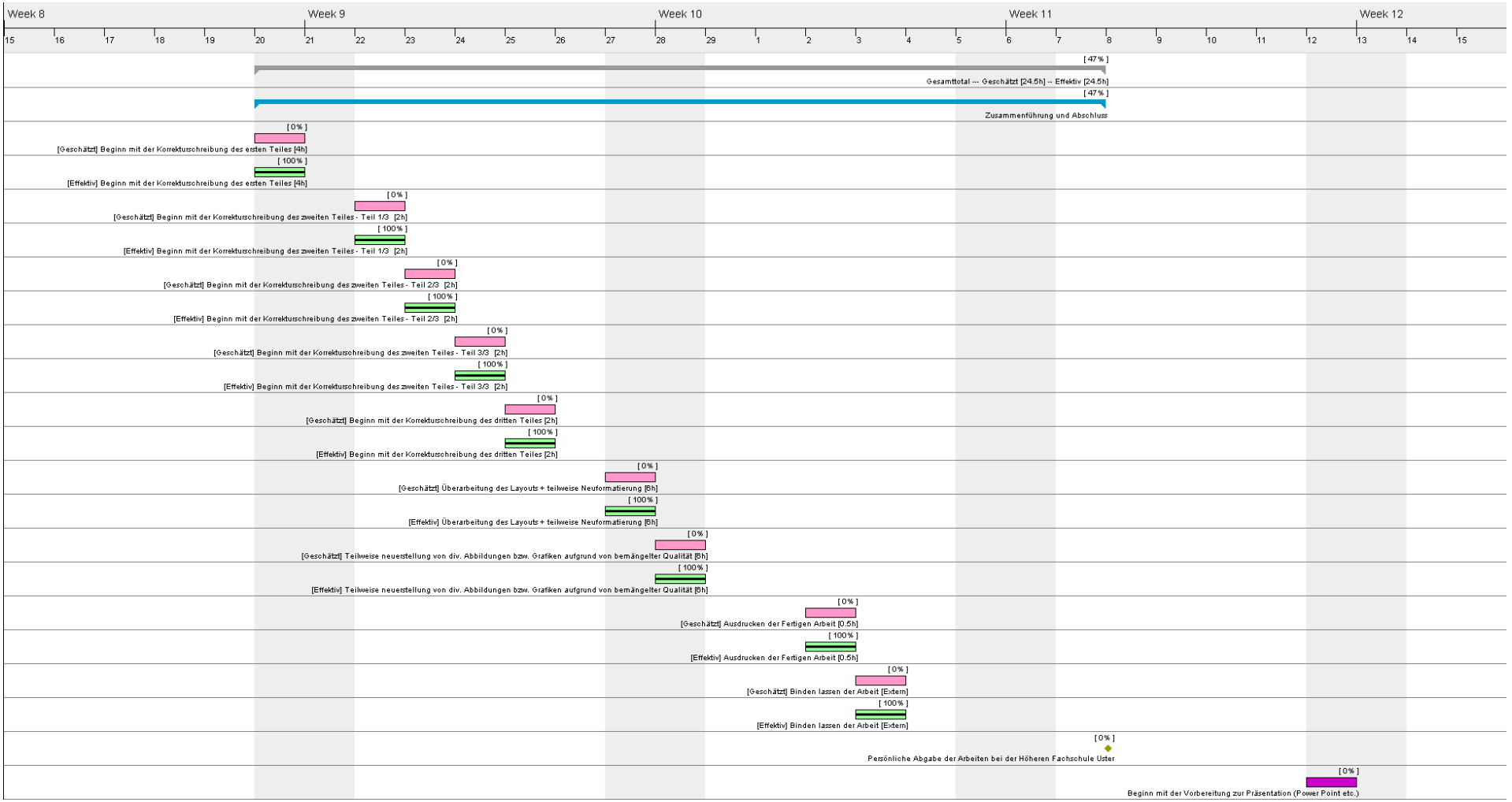
## 4.6 Projektplan des Monats Januar Teil 2



## 4.7 Projektplan des Monats Februar Teil 1



## 4.8 Projektplan des Monats Februar Teil 2



# 5 Pflichten

## 5.1 Allgemeine Definition

Die allgemein zu erfüllenden Spezifikationen dieser Arbeit sind unter Punkt 3.3 „Aufgabenstellung“ definiert. Die dort grob umschriebenen Funktionen werden nachfolgend genauer beschrieben.

### 5.1.1 Sinn und Zweck

Der primäre Zweck dieser Arbeit ist die Erstellung eines Virtualisierungsclusters nach dem Vorbild eines Red Hat Virtualization Systems, unter Verwendung der hierfür zur Verfügung stehenden open source Komponenten, welche Red Hat quelloffen zur Verfügung stellt. Es soll mit neueren Technologien wie bspw. GlusterFS gezeigt werden, dass es möglich ist mit günstiger leistungsschwacher Hardware einen vollwertigen Virtualisierungscluster zu erstellen. Dieser soll neben den heute üblichen Features wie Live Migration, Ressourcenzuteilung oder das Erstellen von vorkonfigurierten Vorlagen, auch eine funktional einfach zu bedienende und auch ansprechende Management – Oberfläche bereitstellen. Es soll möglichst einfache Standard – Hardware verwendet werden. Dies bedeutet den Verzicht auf spezialisierte Hardware – Komponenten wie RAID Controller, gemanagte Switches oder teure Glasfaser – Anbindungen. Wo dennoch spezielle Hardware notwendig werden sollte, wird auf Gebrauchtware ausgewichen.

Diese Arbeit soll nicht nur den Aufbau eines leistungsstarken Virtualisierungsclusters mit Standard – Hardware demonstrieren, sonder auch zugleich die neue Hosting Basis des Autors bereitstellen. Somit ist dies kein rein theoretisches Projekt, sondern es soll ein vollwertiges Produktivsystem darstellen, welches der Autor dieser Arbeit aktiv nutzen kann.

### 5.1.2 Geltungsbereich

Die hier in dieser Arbeit erwähnten und eingesetzten Komponenten unterstehen den jeweiligen Lizenzen und Namensrechten des Herstellers / Anbieters. Innerhalb dieser Arbeit können Namen oder Bezeichnungen vorkommen welche dem Urheberrecht des Herstellers / Anbieters unterstehen, aber nicht mit den notwendigen Rechtsschutz – Zeichen markiert sind. Hier gilt das Urheberrecht/ Namensrecht des Herstellers / Anbieters auch ohne spezifische Zeichen.

Diese Arbeit untersteht bis auf die extern hinzugezogenen Grafiken der Urheberschaft des Verfassers dieser Arbeit. Diese Urheberschaft mit dem dazugehörigen Copyright endet mit der Bekanntgabe der Bewertung dieser Arbeit, oder automatisch am Tagesende des 2. April 2016. Ab diesem Zeitpunkt kann diese Arbeit lizenzlos als Anleitung / Leitfaden genutzt oder ggf. verändert werden.

### 5.1.3 Referenzierte Dokumente und Anleitungen

Die in den meisten Fällen zu Rate gezogene Hilfsquelle ist wie in der heutigen Zeit üblich eine Google – Suche. So soll an dieser Stelle die Definition getroffen werden, dass eine Referenzierung auf eine Online Dokumentation stets mit „→ Google – Anfrage“ beschrieben wird. Dies ist eine globale Definition, welche die hohe Anzahl an Online – Referenzierungen innerhalb dieser Arbeit vereinfachen soll. Bücher oder spezifische Anleitungen in schriftlicher Form werden in den Quellen erwähnt.

## 5.2 Zielsetzung

Das definierte Gesamtziel dieses Projektes ist die Erstellung eines Virtualisierungsclusters auf Basis des open source Framworks oVirt, unter Verwendung von günstiger Standard – Hardware. Es soll mittels der Distributed Filesystem – Lösung GlusterFS, welche seitens Red Hat klar propagiert aber nicht als Standard gilt, ein zentraler Storage erstellt werden. Dieser soll aus leistungsschwacher Hardware das Maximum an Geschwindigkeit und Speichervolumen herausholen. Ebenfalls soll mittels eins älteren IBM Systems gezeigt werden, dass Switch- ähnliche Funktionen auf einer Firewall Distribution nahezu gänzlich abgedeckt werden können. Im Groben gesprochen soll mit wenig das Maximum an Performance herausgeholt werden.

Die genauen Spezifikationen, welche zwingend zu erfüllen sind, werden in der nachfolgenden Tabelle genannt:

| Funktion / Bedingung | Beschreibung | Art der Anforderung |
|---|---|---|
| **Netzwerksegment** | | |
| **Link Aggregation** | In den Bereichen VM – Network und internes Kommunikations-Netzwerk (Management/ Storage) sollen Link Aggregationen so konfiguriert werden, dass Lastteilung und Ausfallsicherheit so gut wie möglich gewährleistet sind. | **Muss Ziel** |
| **Vereinigung in Bridge (Switch Nachbildung)** | Der VM – Network Teil soll so konfiguriert werden, dass er wie ein konventioneller Switch funktioniert. Dieser Punkt soll den Nachweis erbringen, dass auch eine Firewall – Distribution als vollwertiger Switch genutzt werden kann. Monitoring Funktionen wie sie in einem Switch vorkommen können, sind nicht expliziter Bestandteil dieser Arbeit, wo es aber Vergleichsmöglichkeiten gibt wird versucht auf diesen einzugehen. | **Muss Ziel** |
| **Firewall Regelsätze** | Die Firewall soll auch so konfiguriert werden, dass eine annähernd 100%-ige Isolation zwischen dem VM- und dem internen Teil erreicht wird. Dieser Teil ist allgemein und bedarf Konfigurationen an diversen Stellen. | **Muss Ziel** |
| **Firewall Regelsätze (Bridge spezifisch)** | Dieser Teil ähnelt dem oben beschriebenen Punkt. Er wird speziell hervorgehoben, da bei pfSense die Interface – Konfiguration (Firewall) auch nach einem Zusammenschluss in eine Bridge weiterhin möglich ist. So muss eine Einzelkonfiguration der Schnittstellen durchgeführt werden um tatsächliche Isolation zu gewährleisten. | **Muss Ziel** |

*Tabelle 1: Muss Ziele: Netzwerksegment*

| Funktion / Bedingung | Beschreibung | Art der Anforderung |
|---|---|---|
| **Virtualisierungssegment** | | |
| **High Availability (Nachtrag nach Korrektur)** | Es soll so gut wie möglich eine hohe Verfügbarkeit der VM's realisiert werden. Aufgrund technischer Abhängigkeiten ist vollautomatisches HA nicht möglich, die genaue Beschreibung des Problems erfolgt unter Punkt 8 „Realisierung“ sowie eine Korrekturerklärung unter Punkt 5.4 „Abweichungen und Korrekturen“. Eine Alternative, genannt Semi-Automatic-HA, welche seitens des Autors definiert wurde, kommt hier als Ersatz des vollautomatischen HA zum Einsatz.<br>Die Entscheidung hierzu wurde in der dritten Betreuer-Sitzung besprochen und genehmigt. Das Formelle hierzu ist auf dem beiliegenden Datenträger zu finden. | **Muss Ziel** |
| **Last – Verteilung (Nachtrag nach Korrektur)** | Es soll möglich sein die Last so effizient wie möglich auf die zur Verfügung stehenden Nodes zu verteilen. Hierzu wird mit Affinitäts- Gruppen gearbeitet, da die vollautomatische Lastverteilung, welche es der Engine ermöglichen würde die VM Verteilung der momentanen Systemlast der Nodes vollautomatisch vorzunehmen, ebenfalls den unter Punkt „High Availability“ beschriebenen technischen Problemen unterliegt. | **Muss Ziel** |
| **Affinitäts- Gruppen** | Teilweise aufbauend auf den oberen Punkt, soll es möglich sein Affinitäts- Gruppen zu bilden. So kann mit Positiv- oder Negativwerten deklariert werden ob VM's welche in einer gemeinsamen Gruppe sind zwingend auf einem Host zusammenbleiben müssen, oder in der Negativansicht zwingend voneinander getrennt werden müssen. | **Muss Ziel** |
| **Live Migration** | Es soll möglich sein eine virtuelle Maschine von einem Host zu einem anderen zu migrieren. Dies unter der Bedingung, dass die Maschine nicht heruntergefahren werden muss. Hierfür sollen die Cluster so konfiguriert werden, dass sich die Prozessor – Familien überschneiden und kein Pseudo- Prozessor (qemu-kvm) zum Einsatz kommt. | **Muss Ziel** |
| **Ressourcenbegrenzung (oVirt - QoS)** | Für das Datacenter (Cluster) soll ein schmaler Satz (höchstens 3) an Regeln zur Ressourcenbegrenzung erzeugt werden. Eine stark begrenzende Regel (MIN) für „unwichtige“ Maschinen, eine Standard Regel (STD) für den Normalbetrieb und eine hoch priorisierte Regel (HIGH) welche „unbegrenzt“ als Standardwert definiert hat. Diese drei Kürzel werden jeweils im VM – Namen vorkommen, um den QoS – Zustand der VM's direkt herauslesen zu können. | **Muss Ziel** |

*Tabelle 2: Muss Ziele: Virtualisierungssegment*

| Funktion / Bedingung | Beschreibung | Art der Anforderung |
|---|---|---|
| **Storagesegment** | | |
| **Zentralisierung** | Es soll aus zwei Komponenten ein einziger Storage gebildet werden, welcher auch nur über einen Einbindepunkt erreichbar ist. Hierfür wird GlusterFS genutzt um ein Distributed Filesystem zu erzeugen. | **Muss Ziel** |
| **Lastverteilung** | Eine Lastverteilung der Lese- und Schreibzugriffe seitens der Virtualisierungsnodes soll möglich sein. Hierfür wird das Standard – Distributed – Verfahren von GlusterFS verwendet. Eine effizientere Lösung mittels Stripping kann momentan nicht realisiert werden, da dies aus technischen Gründen (Locking Files) nicht möglich ist. | **Muss Ziel** |
| **Leistungsoptimierung (Festplatten und Filesystem)** | Es wird versucht mittels einer optimalen Blocksize – Grösse die optimale Performance aus dem RAID Verbund herauszuholen. Dies resultiert aus der Tatsache, dass eine nicht optimale Festplattenanzahl für einen RAID 5 – Verbund gewählt wurde. | **Muss Ziel** |
| **Ausfallsicherheit** | Die Ausfallsicherheit seitens ganzer Storage – Nodes ist aufgrund der Anzahl der verfügbaren Maschinen nicht möglich. Hier beschränkt sich die Ausfallsicherheit auf die Festplatten, welche mittels eines RAID Levels gesichert sind. | **Muss Ziel** |

*Tabelle 3: Muss Ziele: Storagesegment*

## 5.3 Wunschziele

Nachfolgend eine Liste mit den Wunschzielen, welche technisch gesehen zu Performance – Steigerungen oder allgemein zu einer Verbesserung des Endergebnisses führen. Diese sind aber klar abgegrenzt zum eigentlichen Auftrag und werden nur bei Bedarf oder Zeit implementiert.

| Funktion / Bedingung | Beschreibung | Art der Anforderung |
|---|---|---|
| **DNS System** | Es kann ein primärer DNS Server innerhalb der Virtualisierung aufgebaut werden, welcher einen sekundären DNS mit den eigenen Werten (dynamische IP; Privatanschluss) versorgt. Ein Script innerhalb des privaten Netzwerks würde die dynamisch wechselnde IP erfassen und diese über einen Cronjob unter den CNAME's updaten. So würde eine statische IP bei einem Hoster (externe VM) existieren und die eigentlichen Domain – Einträge könnten lokal erledigt werden. Momentan lässt sich aber kein dem Budget entsprechender Hoster finden. | **Wunschziel** |
| **Reverse Proxy** | Über eine Reverse Proxy Implementierung könnten die innerhalb der Virtualisierung laufenden Webservices über eine IP und zwei Ports angesprochen werden, da ja die URL's aus dem HTTP – Header herauslesbar wären. Dieser Punkt steht aber in Abhängigkeit mit dem oberen und kann daher erst realisiert werden wenn eine Lösung für das „DNS – System“ gefunden wurde. | **Wunschziel** |
| **Zabbix Überwachung** | Ein Monitoring der VM's ist ohnehin allgemein notwendig. Der Autor war sich aber beim Schreiben dieser Zeilen nicht im Klaren, welche Lösung eingesetzt werden soll. Dies bedarf einer genaueren Abklärung, da diese Lösung ein Langzeitsystem darstellt und sich nur unter grossem Aufwand ändern lässt. | **Wunschziel** |

*Tabelle 4: Wunschziele: Allgemeine Ziele*

## 5.4 Abweichungen und Korrekturen

Da während dem Aufbau des Clusters immer wieder mit Schwierigkeiten zu kämpfen war, resultierten folglich auch einige minimale Änderungen daraus. Diese Änderungen haben aber am Hauptdesign des Auftrages nie etwas geändert. Eine grosse Schwierigkeit erforderte aber eine Betreuersitzung, in welcher ein Korrekturbeschluss protokolliert werden musste. Dies betrifft wie im Protokoll des Betreuungsgespräches Nummer 3 herauszulesen ist die Hochverfügbarkeit. An dieser Stelle muss zugegeben werden, dass die Vortests vor einreichen des Antrages zu wenig genau durchgeführt wurden. Die Annahme, dass sich das Ermitteln des Ausfalles eines Virtualisierungsnodes nur durch die Management – Engine ermitteln lässt war nicht korrekt. Für eine saubere Erkennung eines toten Hosts ist auf jeder Maschine das Vorhandensein eines LOM – Interfaces notwendig. Dieses Interface wird nicht direkt von der Engine kontaktiert, sondern die Engine fordert die restlichen noch lauffähigen Hosts auf, jeweils nacheinander dem toten Host eine Steuernachricht zu schicken. Da aber einer der Hosts aus dem High – Level Cluster kein Server Mainboard, sondern ein Workstation Board enthält, ist somit auch kein LOM – Interface vorhanden. Somit ist eine der elementarsten Komponenten im Cluster nicht fähig HA durchzuführen. Zugleich behindert das Fehlen dieses High – Level Nodes auch die Stuktur des geforderten High Availability – Auftrages (inkl. Automatischer Lastverteilung der VM's).

Als Korrektur dieser Fehleinschätzung wurde eine Alternativlösung namens Semi-Automatic-HA beschlossen und entsprechend im Protokoll festgehalten. Diese Lösung sieht die Installation eines Monitoring – Systems wie Zabbix oder einer Nagios – Variante vor, welche über die Überwachung der einzelnen VM's den Ausfall eines Virtualisierungsnodes erkennen soll. Anhand dieser externen Monitoring – Warnung kann nachfolgend das manuell Fencing eingeleitet werden, welches die Engine gewissermassen „gewaltsam“ anweist den Host als tot anzuerkennen und mit dem Neustarten der VM's auf einem anderen System zu beginnen. Diese Lösung wird aber aus Zeitgründen nicht spezifisch umgesetzt, sondern lediglich in der Theorie im betreffenden Abschnitt unter Punkt 8 „Realisierung“ in einer mögliche Umsetzungsvariante erläutert.

# 6 Vorabklärungen und Analysen

## 6.1 Zweck und Umfang der Vorstudie / Analysen

Diese Vorstudie soll sich mit den Bereichen Netzwersegment und Storagesegment befassen. Genauer mit den beiden nachfolgenden Fragen:

- Welches System soll für den Netzwerkknoten verwendet werden. Hier muss geklärt werden ob das klassische und bewährte pfSense oder das neuere, unerprobtere aber dennoch innovativere opnSense zum Einsatz kommt.
- Ebenso soll die elementare Frage bezüglich des Storage geklärt werden. Soll auf volles Risiko gesetzt werden und eine Lösungen mit möglichst hoher Speicherplatz – Ausbeute gewählt werden, oder soll eine Lösung mit absoluter Verfügbarkeit präferiert werden. Die Entscheidungsfindung dreht sich hierbei um die Frage RAID 5 oder RAID 10.

Diese beiden Fragen müssen mittels einer genauen Analyse im Vorfeld geklärt werden, da nachträgliche Änderungen im Bereich dieser beiden Kernkomponenten im Nachhinein nicht ohne grossen Aufwand möglich sind.

Die meisten Design – Entscheidungen sind durch den Auftrag sowie durch die physischen Gegebenheiten klar definiert. Diese Vorstudie, welche sich mit den beiden oben genannten Fragen beschäftigt, soll sich hauptsächlich mit dem hardwareseitigen Kernaufbau beschäftigen.

## 6.2 Zielsetzung

Das klar definierte Ziel dieser Vorstudie ist die Abklärung der beiden elementaren Fragen bezüglich pfSense / opnSense sowie der Wahl des zu verwendenden RAID – Levels. Hierzu sollen allgemein gesprochen folgende Parameter mittels diverser Analyseverfahren geklärt werden:

- In Bezug auf das Netzwerksegment:
  - Die Lösung soll stabil lauffähig sein.
  - Sie soll auf jeden Fall eine leichte Bedienung ermöglichen.
  - Das Produkt muss ständig auf einem hohen Release – Stand sein (Sicherheitslücken/-Updates)
  - Nicht zwingend notwendig aber von Vorteil wäre eine modulare Erweiterbarkeit der softwareseitigen Komponenten. Alternativ kann auch eine Lösung gewählt werden welche nicht erweiterbar ist, aber eine Fülle an Funktionen von Haus aus mitbringt.
  - Die im Pflichtenheft (Netzwerksegment) geforderten Anforderungen sollten enthalten sein.
- Storagesegment:
  - Performante Anbindung an das Virtualisierungssegment. Dieser Punkt ist von höchster Priorität, da ansonsten die Ausführungsgeschwindigkeit der VM's massiv darunter leiden könnte. Das RAID soll in diesem Fall die notwendige Performance liefern können.
  - Die Ausfallsicherheit sollte bis zu einem gewissen Masse gewährleistet sein.

- Das Auswechseln von Festplatten sollte noch möglich sein, bzw. das Wiederherstellen des gewählten RAID – Level sollte noch leicht möglich sein.

Die Klärung dieser Fragen sollte zu einem Netzwerksegement und einem Storagesegment führen, welches die nach Pflichtenheft zu erfüllenden Anforderungen vollumfänglich unterstützt, aber so viele Extras wie möglich bietet.

## 6.3 Zusätzliches

An dieser Stelle sei noch ein kleiner Testversuch bezüglich Storagesegment erwähnt. Eine erste Überlegung, welche nach Pflichtenheft nicht genau spezifiziert wurde, aber zu beginn als beste Lösung klassifiziert wurde, war die Verwendung des „stripe“ - Verfahrens bei GlusterFS. Hierbei wäre eine RAID – Level 0 Nachbildung, verteilt über zwei GlusterFS Nodes möglich gewesen. Diese hätte zwar die Wiederherstellungsfähigkeiten massiv verschlechtert, aber die Lese- und Schreibgeschwindigkeiten bis zur Hardwaregrenze maximiert. Dieses Verfahren ist in simplen Storage – Verfahren wie NAS – Systemen voll funktionsfähig. Die Verwendung in oVirt ist aber heute aus technischen Gründen nicht realisierbar, da das Locking – File welches von den Virtualisierungnodes direkt in die Partition der Storage – Nodes geschrieben werden muss, nicht mit der versplitteten Arbeitsweise von GlusterFS Distributed Striping zurecht kommt.

## 6.4 Klärung der Netzwerksegment – Frage

In den nachfolgend Unterkapiteln soll versucht werden die Frage nach der zu verwendenden „xxSense“ Distribution zu klären.

### 6.4.1 Einleitung

Die Verwendung eines FreeBSD basierten Derivats ist beim erzielen einer hohen Netzwerk – Performance unverzichtbar. Daher war für den Autor dieser Arbeit von Anfang an klar, dass ein FreeBSD System zum Einsatz kommen muss. Ein leicht bedienbares GUI für schnelle Eingriffe ist beim Management eines so komplexen Systems schon beinahe Pflicht. Daher war die Wahl von pfSense schon vorprogrammiert. Jedoch zeigte das pfSense Projekt in den letzten Versionen eine etwas unschöne Problematik mit der Kompatibilität diverser modularer Erweiterungen untereinander. Das opnSense Projekt versucht dies mittels eines Gesamtkonzeptes zu beheben, indem man nicht zulässt dass modulare Erweiterungen von Wildfremden ins Projekt fliessen, sondern alle Funktionen fixer Bestandteil von opnSense sind. So kann das gesamte Projekt, welches wie aus einem Guss wirkt kontinuierlich weiterentwickelt werden. Jedoch muss klar gesagt werden, dass das opnSense Projekt noch in einer späten Anfangsphase steckt und nicht alle Funktionalitäten aus pfSense repliziert worden sind.

### 6.4.2 Allgemeine Vorabklärungen

Vorabklärungen im grossen Still sind nicht notwendig, da der Autor dieser Arbeit pfSense seit Jahren im privaten Umfeld sowie in diversen fremden Umgebungen nutzt. Das Aufkommen von opnSense ging auch nicht unbemerkt am Autor vorbei. Seit dem ersten Fork wurden vergleiche zwischen den xxSenses gezogen und mögliche Migrationsszenarien geplant. Daher wird an dieser Stelle auf Online – Recherchen verzichtet und aus dem eigenen Wissensstand heraus analysiert.

### 6.4.3 Benötigte Funktionalitäten

Zur Realisierung und somit als klar definierter Anforderungskatalog sind nachfolgende Funktionalitäten zwingend notwendig:

- Möglichkeit **Link Aggregationen** zu bilden.
- Möglichkeit **Bridges** zu bilden, um Switch – Funktionen nachzubilden.
- **Umfangreiche Firewall – Funktionen** welche das Sperren von ganzen Netzwerken bis hin zu einzelnen IP – Adressen ermöglichen.
- **Aufzeichnung von Traffic**. Dies im Sinne von Logs zum Zwecke der Sicherheit und das Aufzeichnen von Traffic zur Erstellung von internen Statistiken.
- **Funktionsumfang / Erweiterbarkeit;** Dies ist nicht zwingend notwendig für die Erfüllung dieser Arbeit. Aufgrund der Nutzung der Firewall im privaten Umfeld, soll diese Option für spätere Ausbaumöglichkeiten offen bleiben.
- **Erfahrung / Stabilität;** Dieser Punkt stützt sich auf das Alter der Projekte und somit der daraus resultierenden, unter Produktivsystemen erbrachten Erfahrung. Es wird aber versucht die langjährige Erfahrung seitens pfSense in direktem Vergleich mit der Stabilität von beiden Produkten zu setzen. So soll auch berücksichtigt werden, dass pfSense heute leichte betriebstechnische Instabilitäten aufweist und das opnSense nach Angaben seiner Entwickler in naher Zukunft einen höheren Grad an Stabilität erreichen soll als pfSense.
- **Zeitfaktor;** Zwar wird versucht eine Lösung mit bestmöglichen Zukunftsaussichten zu wählen, jedoch müssen gewisse Faktoren in kürzerer Zeit erfüllbar sein. So kann bspw. die zukünftige hohe Stabilität von opnSense klar als ein Pluspunkt gewertet werden. Dauert dies aber länger als die produktive Inbetriebnahme des Clusters oder ist gar die Rede von Jahren, so kann dies keine Option sein. Dieser Punkt kann diverse Faktoren wie Funktionen, Extrafunktionen, besseres Aussehen etc. beinhalten.
- **Die Bedienung;** Da opnSense pfSense als Basis hat, ist die Bedienung theoretisch identisch. Die GUI's sind aber in einigen Punkten vom Aufbau und Design her unterschiedlich. Dieser Punkt stellt einen Vergleichswert dar mit dem Schwerpunkt, welches GUI ist das angenehmere und natürlich auch das optisch ansehnlichere.

### 6.4.4 Der direkte Vergleich

In nachfolgender Tabelle soll aufbauend auf den Punkten der unter Punkt 6.4.3 „Benötigte Funktionalitäten“ genannten Funktionalitäten / Kriterien, ein direkter Vergleich der beiden Lösungen vollzogen werden. Hierbei wird eine Bewertungsskala mit folgenden Wertigkeiten gewählt:

- **1** → **Niedrig;** Funktionalität / Kriterium ist sehr schlecht bis kaum vorhanden.
- **2** → **Ausreichend;** Funktionalität / Kriterium könnte genügen, aber es sind massive Abstriche notwendige.
- **3** → **Genügend;** Funktionalität / Kriterium ist erfüllt, jedoch gibt es eine bessere Alternative und dem Autor dieser Arbeit gefällt diese Option gefühlsmässig nicht.
- **4** → **Gut;** Funktionalität / Kriterium ist absolut erfüllt.
- **5** → **Sehr Gut;** Funktionalität / Kriterium ist erfüllt. Die Lösung bietet kleine Extras die nicht zwingend notwendig sind, aber dem Autor besonders gefallen.

| Funktionalität / Kriterium | PfSense | OpnSense | Bemerkung/ Begründung |
|---|---|---|---|
| **Link Aggregationen** | 5 | 5 | Ist auf beiden GUI's vom Konfigurationsablauf her identisch. In lokalen Tests (VMware Workstation) zeigten beide Lösungen die gleiche Performance und das gleiche Verhalten. |
| **Bridges** | 5 | 5 | Dito wie Punkt „Link Aggregation“ |
| **Umfangreiche Firewall – Funktionen** | 5 | 4 | Bezogen auf die reinen Firewall Funktionen wie Sperren, Erlauben, Weiterleiten oder nach spezifischen Punkten Filtern sind beide identisch. Jedoch hat opnSense zur besseren Übersicht ihr GUI so modifiziert, dass man die Hilfe – Texte und erweiterte Optionen per Default ausgeblendet hat. Hier verstecken sich aber manchmal wichtige Optionen wie etwa die manuell anzulegende NAT – Regel für Proxy – Funktionen. Solche Situationen können verwirrend sein wenn man sich erst seit kurzem mit opnSense beschäftigt. |
| **Aufzeichnung von Traffic** | 5 | 5 | Diese Funktion ist praktisch 1:1 von pfSense übernommen (mit GUI Design). |
| **Funktionsumfang/ Erweiterbarkeit** | 5 | 3 | PfSense ist per Packages erweiterbar, was viele Zusatzfunktionen ermöglicht. OpnSense hingegen versucht alles in einen Default – Guss zu verpacken. So sollen alle Funktionen die pfSense heute bietet in einem einheitlichen GUI vereint werden. **Es sind aber vom Stand „heute“ gesehen, nicht alle Funktionen implementiert.** |
| **Erfahrung/ Stabilität** | 4 | 4 | Das pfSense Projekt hat sicherlich mehr Erfahrung in dieser Branche und bietet auch mehr Erweiterungsmöglichkeiten. Doch genau diese Möglichkeiten führen heute zu massiven Package – Diskrepanzen untereinander. OpnSense bietet heute nicht alle Erweiterungen an, jedoch ist das was vorhanden ist ziemlich gut. Beiden Produkten wird ein gefühlter Wert „GUT“ gegeben. |
| **Zeitfaktor** | 5 | 3 | Die Zukunftsaussichten von opnSense sehen sehr gut aus. Doch einige Funktionen wie Antiviren – Scanner müssen aus Sicht des Autors bereits heute vorhanden sein. Die Implementierung scheint aber noch in weiter Ferne zu liegen. |
| **Die Bedienung** | 4 | 4 | Schaut man sich jedes GUI einzeln an, so würde man beiden eine „5“ geben. Weiss man aber durch direkte Vergleichstests was das eine Produkt hat, was dem anderen vielleicht auch gut stehen würde, so sinkt die subjektive Wahrnehmung und es reicht gerade mal für ein „GUT“. |
| **Total des Vergleiches** | 38 | 33 | **Sieger nach direktem Vergleich pfSense.** |

*Tabelle 5: Direkter Vergleich der beiden Lösungen pfSense und opnSense*

### 6.4.5 Auswertung des Resultates

Wie in Tabelle 5 zu erkennen ist, kann pfSense nach Punkten klar gewinnen. Würde man die notwendigen Funktionen auf das absolute Minimum reduzieren, könnte auch opnSense als gleichwertiger Ersatz dienen. Hier war aber der Einfluss seitens der Projekt unabhängigen Funktionen wie bspw. Antivirus – Filter oder das weit bessere SquidGurad (Content Filtering) ausschlaggebend. Diese Funktionen sind seitens der privaten Nutzung der Firewall zwingend notwendig.

### 6.4.6 Reelle Umsetzung

Somit wird pfSense auf dem IBM Serversystem installiert und die Netzwerksegmentkonfiguration, welche nach Auftrag und Pflichtenheft erforderlich ist, wird auf einem roten pfSense WebGUI anstelle eines gelben opnSense aufgebaut.

### 6.4.7 Anmerkung zur Analyse

Diese Analyse wurde nach einem vorgängig bereits vorinstallierten pfSense durchgeführt. Dies aus dem Grund, da eine interne Umstrukturierung der eigenen Infrastruktur vor Beginn der Diplomarbeit notwendig wurde. An dieser Stelle wäre eine eigenständige Analyse möglich gewesen. Diese wurde aber aufgeschoben, da sie ohnehin hätte vollzogen werden müssen. So wurde bei der Vorinstallation nur eine Minimalkonfiguration angewendet, um im Falle eines anderen Ergebnisses dieser Analyse eine einfachere Migration vornehmen zu können.

## 6.5 Klärung der Storagesegment – Frage

In den nachfolgenden Unterkapiteln soll versucht werden die Frage nach dem zu verwendenden RAID – Level, RAID 5 vs. RAID 10, zu klären. Aufgrund der schwerwiegenden Fragestellung, mehr Diskspace oder mehr Sicherheit, ist die Klärung äusserst komplex. Aus diesem Grund soll eine Nutzwertanalyse angewendet werden, welche alle Fakten nüchtern vergleicht und rein mathematisch zu einem Ergebnis führt.

### 6.5.1 Einleitung

Die Absicherung vor Unterbrüchen und Datenverlusten muss lokal auf den beiden Storage – Nodes (HP Microserver) realisiert werden. Zwar stellen beide Nodes zusammen einen einzigen Share – Mountpoint dar, jedoch hat GlusterFS hier kein Verständnis für die darunterliegende Hardware oder die Partitionen des Backends. Die Sicherung der Daten muss also über einen RAID – Level Verbund realisiert werden. Hier stellt sich nun die Frage, soll die Sicherheit an erster Stelle stehen und somit ein RAID 10 – Verbund kreiert werden, oder soll so viel Diskspace pro Node wie möglich mittels RAID 5 herausgeholt werden.

#### 6.5.1.1 Wie könnte ein RAID 10 aussehen

Nachfolgende Definitionen orientieren sich an einem Node (standalone):

Ein RAID 10 Verbund würde je zwei Disks in ein RAID 1 Subarray binden, über welches ein primärer RAID 0 gebildet wird, welcher die Schreib- und Lesegeschwindigkeit erhöhen würde. Siehe nachfolgendes Schema:

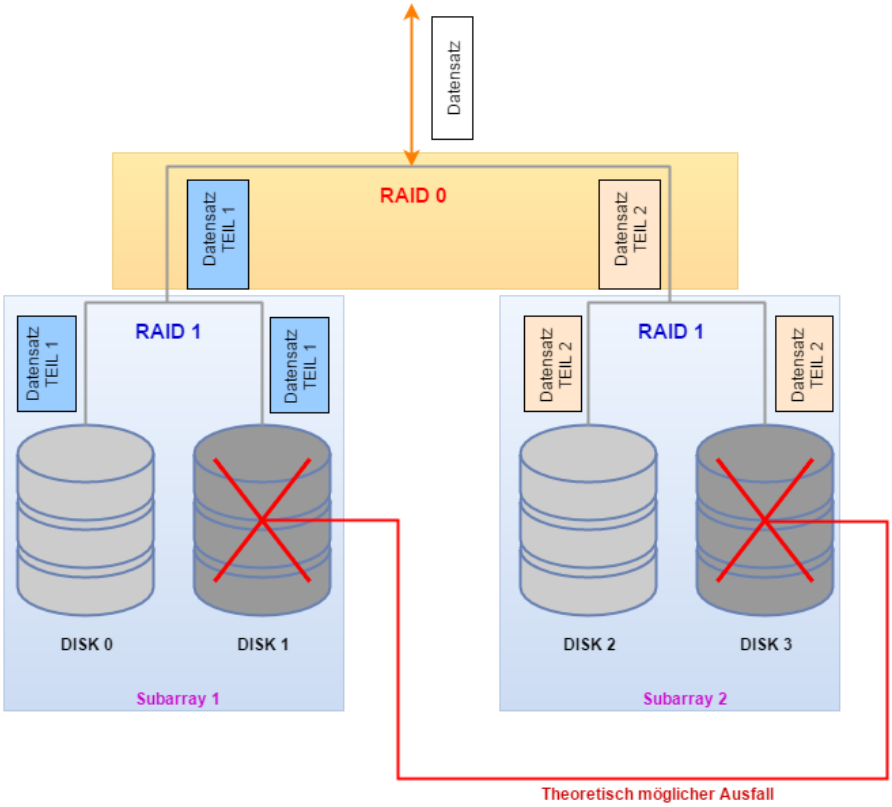



*Abbildung 5: Schema eines RAID 10 Verbundes auf einen Node bezogen*

Dieses Gebilde würde Sicherheit dahingehend liefern, dass „theoretisch“ zwei Festplatten ausfallen könnten. Die Genaue Spezifikation wäre:

Subarray 1 (RAID1(Disk0, Disk1))

Subarray 2 (RAID1(Disk2, Disk3))

RAID10 = RAID0 (RAID_Subarray 1, RAID_Subarray 2)

Jedoch muss klar gesagt werden, dass der Ausfall sich auf zwei Festplatten aus zwei unterschiedlichen Subarrays beziehen würde. Sollten zwei Disks aus demselben Subarray ausfallen, so funktioniert das RAID 0 Primary RAID nicht mehr. Die Wahrscheinlichkeit, dass Disks aus dem gleichen Subarray ausfallen ist theoretisch so gering, dass das Risiko aus Sicht des Autors tragbar ist.

**Die aus diesem Gebilde resultierenden Vorteile wären:**

- Hohe Schreib- und Lesegeschwindigkeit über RAID 0
- Theoretisch eine hohe Ausfallsicherheit basierend auf der Tatsache, dass zwei Disks pro Node ausfallen können.
- Disk Anzahl wäre in dieser Konstellation optimal.

**Die aus diesem Gebilde resultierenden Nachteile wären:**

- Der grösste Nachteil ist der relativ hohe Diskspace – Verlust. Rein mit Rohdaten gerechnet, würden pro Subarray 1,8 TB (gespiegelt) zur Verfügung stehen. Über die RAID 0 Primary – Verbindung wären das 2x 1,8 TB = 3,6 TB an verfügbarem Diskspace je Node. Praktisch gesprochen kann hier nur die Kapazität von zwei Festplatten genutzt werden.
- Das angesprochene Risiko, dass nie zwei Disks aus dem gleichen Subarray ausfallen dürfen, wird als äusserst gering eingeschätzt. Da aber die alten Disks aus dem Vorgänger – Cluster zum Zuge kommen, welche auch schon 3 – 4 Jahre alt sind, kann das Risiko nicht zu 100% ausgeschlossen werden.

### 6.5.1.2 Wie könnte ein RAID 5 aussehen

Nachfolgende Definitionen orientieren sich an einem Node (standalone):

Ein RAID 5 – Verbund würde aus vier Disks bestehen, wobei keine separate Disk für die Parität – Daten verwendet würde. Die Paritätsinfomationen werden per Default auf alle vier Disks verteilt. Siehe nachfolgendes Schema:

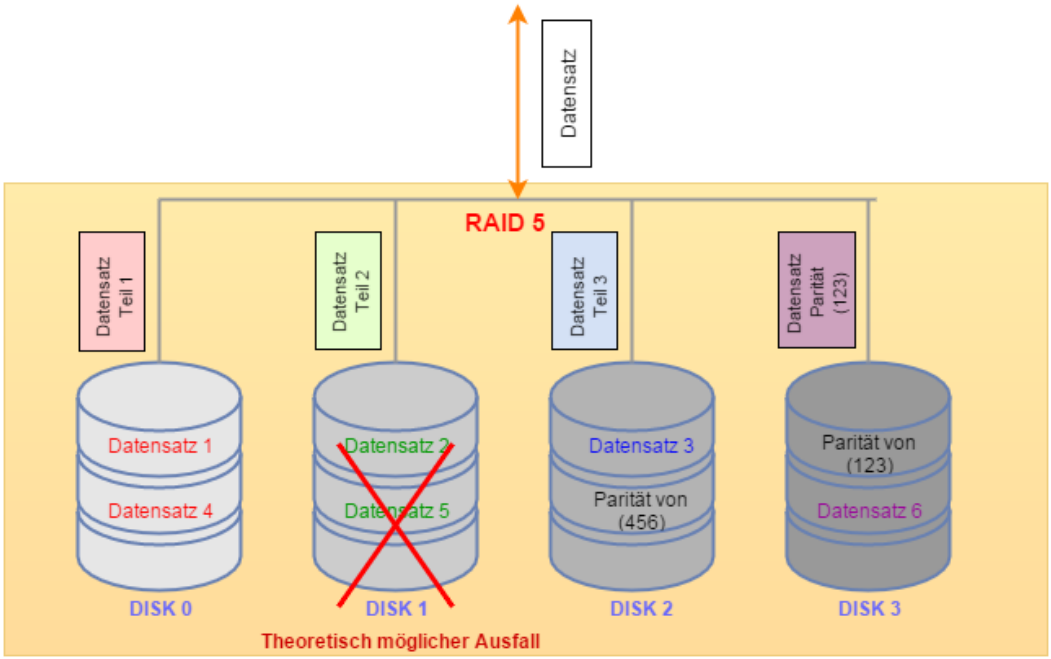



*Abbildung 6: Schema eines RAID 5 Verbundes auf einen Node bezogen*

Dieses Gebilde würde mit einer Ausfallmöglichkeit von nur einer Disk sicherlich das Betriebsrisiko massiv erhöhen. Jedoch wäre der Diskspace – Gewinn bei dieser Konstellation der theoretisch grösste bei Beibehaltung von einer gewissen Ausfallsicherheit. Die Genaue Spezifikation wäre:

RAID5 = (Disk0, Disk1, Disk2, Disk3)

Diese Konstellation würde bei momentaner Nichtbeachtung der Chunk – Size einen gewissen Verlust des Diskspaces beim bilden des RAID 5 nach sich ziehen. Im Schnitt spricht man hier von der Grösse

einer ganzen Festplatte. Dies würde bedeuten, dass bei 3x 1,8 TB (-1 für RAID 5) eine Gesamtkapazität von 5,4 TB zur Verfügung stehen würde. Beim bilden eines GlusterFS Mountpoints würde dies eine für den Virtualisierungscluster sichtbare Gesamtkapazität von ca. 10,8 TB bedeuten. Im Vergleich zum RAID 10 würde dies einen Gewinn von 3,6 TB bedeuten.

**Die aus diesem Gebilde resultierenden Vorteile wären:**

- Lesegeschwindigkeit ist technisch gesehen höher, da von drei Platten gelesen werden kann.
- Bei den zur Verfügung stehenden Komponenten ist diese Konfiguration sicherlich die mit dem grössten Diskspace – Gewinn der überhaupt möglich ist.
- Wiederherstellung der Daten beim Ausfall einer Platte geht einfacher, da sich das RAID selbst um die Neuberechnung der Daten über die Patritätsdaten kümmert.

**Die aus diesem Gebilde resultierenden Nachteile wären:**

- Die Schreibgeschwindigkeit ist bedeutend langsamer, da die Position der einzelnen Teil – Datensätze auf den Festplatten samt Paritätsdaten erst vorberechnet werden muss.
- Die Wiederherstellung der Daten einer neuen Platte wird zwar teilweise automatisch gemanagt, aber dies kann bei einer 1.8 TB Platte relativ lange dauern und frisst eine Menge an Systemressourcen. Dies kann sich bei einer intensiven Nutzung des Storages seitens des Virtualisierungscluster zu spürbaren Performanceeinbussen führen.
- Die Anzahl Festplatten ist bei einem RAID 5 von entscheidender Bedeutung für die maximale Ausbeute an Kapazitätsausnutzung. In der Regel verwendet man Platten in den Konstellationen 3, 6, und 9. Diese Zahlen garantieren eine optimale Verteilung der Paritätsdaten anhand des beim RAID 5 zum Einsatz kommenden Algorithmus. Dabei sinkt der Verlust, welcher durch das Schreiben der Paritätsdaten entsteht bei steigender Zahl der Festplatten, da sie bei einer höheren Anzahl effizienter verteilt werden können. Die Konstellation von vier Platten innerhalb dieses Projektes gilt als ineffizient. Dem Verlust an Kapazitätsausnutzung kann hier nicht entgegengewirkt werden, jedoch ist es möglich die hierbei entstehende Schreib- und Lesegeschwindigkeit mittels einer optimalen Chunk – Size wieder in den Optimalwert von drei Platten zu bringen.

### 6.5.2 Welcher RAID – Level? (Nutzwertanalyse)

In der nachfolgenden Nutzwertanalyse sollen anhand von klar definierten Kriterien, die entsprechenden Gewichtungsfaktoren und die dazugehörigen Priorisierungen der Kriterien so gewählt werden, dass ein performanter und Kapazitätsreicher Storage Cluster entsteht, welcher den hohen Anforderungen einer durch eine Vielzahl an VM's erzeugten Belastung stand hält.

#### 6.5.2.1 Die notwendigen Auswahlkriterien

| Kriterien | Deren Begründung bzw. Notwendigkeit |
|---|---|
| **Mehrfach Ausfall** | Aufgrund des Alters der sich im Moment in Betrieb befindlichen Disks, wäre es bis zum nächsten Hardware – Upgrade von Vorteil, wenn das Risiko des Mehrfachausfalles von Disks besser geschützt wäre.<br>**Dieser Punkt orientiert sich stark an der RAID 10 Lösung.** |
| **Lesegeschwindigkeit** | Die Lesegeschwindigkeit ist bei einem externen Storage von grösster Wichtigkeit. Besonders die Bootzeiten der VM's können bei zu langen Wartezeiten zu Problemen führen. Da der Grossteil der zukünftig in Betrieb stehenden Maschinen Serversysteme sein werden (Webservices), ist schnelles Lesen vom Storage zwingend notwendig.<br>**Dieser Punkt orientiert sich stark an der RAID 5 Lösung.** |
| **Schreibgeschwindigkeit** | Genau wie die Lesegeschwindigkeit, ist auch die Schreibgeschwindigkeit von grösster Bedeutung. Zu lange Wartezeiten beim Schreiben, können eine negative, subjektive Wahrnehmung beim Desktop – User erzeugen. Desktop Systeme werden aber selten zum Einsatz kommen. Was aber neben den stark lesenden Webservices auch noch eine gewisse Bedeutung geniesst, sind Datenbanken als Backend – Systeme. Hier wäre es von Vorteil wenn die Schreiboperationen schnell umgesetzt werden können.<br>**Dieser Punkt orientiert sich stark an der RAID 10 Lösung.** |
| **Diskspace** | Bedenkt man die hohen Ressourcen welche von den beiden High – End – Maschinen zur Verfügung gestellt werden, so ist eine hohe Anzahl an VM's möglich. Diese werden zwangsläufig auch einen hohen Anteil an Storage benötigen. Es wäre ratsam hier auf grösstmögliches Speicherplatz – Volumen zu setzen.<br>**Dieser Punkt orientiert sich stark an der RAID 5 Lösung.** |
| **Wiederherstellung (Einfachheit)** | Im Falle eines Ausfalles gibt es bei Software – RAID 10 nur die Möglichkeit die noch lauffähige Disk des Subarray (RAID 1) auf die neue zu Klonen. Dies ist meist mit einer gewissen Down – Time verbunden. Bei RAID 5 muss ja ohnehin der Inhalt der neuen Disk aus den Paritätsdaten der noch lauffähigen errechnet werden. So kann dies im Laufenden Betrieb vollzogen werden.<br>**Dieser Punkt orientiert sich stark an der RAID 5 Lösung.** |
| **Kostenfaktor (bezogen auf Wiederherstellung)** | Die Anschaffungskosten sind bei einer Disk ohnehin die selben. Jedoch ist es heute üblich auch die Nachbardisk eines RAID 1 – Verbundes anschliessend auszuwechseln. Im Falle eines Disk – Neukaufes ist es auch wahrscheinlich, dass Hardwarediskrepanzen bezüglich Caches und aktiver Lese- / Schreibgeschwindigkeit bestehen können. Um Performance – Einbussen beider Disks zu vermeiden, bleibt das Auswechseln beider Disks unvermeidbar. Bei einem RAID 5 besteht prinzipiell das gleiche Problem, jedoch wäre hier ein Auswechseln aller Disks massiv teurer und würde zu einer langen Downtime führen. An dieser Stelle muss die Problematik einfach akzeptiert werden.<br>**Dieser Punkt orientiert sich stark an der RAID 5 Lösung.** |
| **Risiko Faktor** | Dieser Faktor liegt rein im subjektiven Empfinden des Autors dieser Arbeit und beschreibt primär das Risiko, welches eingegangen werden muss im Falle einer Wahl des RAID 5 Verbundes. Praktisch gesprochen ist dies ein reiner Zahlenfaktor, welcher helfen soll innerhalb der Gewichtungsbestimmung der Kriterien in der nachfolgenden Tabelle, einen gewissen subjektiven, „emotionalen“ Einfluss nehmen zu können.<br>**Dieser Punkt kann einen elementaren Einfluss auf beide Lösungen haben.** |

*Tabelle 6: Kriterien für die Nutzwertanalyse zur Wahl eines geeigneten RAID - Level*

### 6.5.2.2 Ermittlung des Gewichtungsfaktors für die einzelnen Kriterien (GWF)

| Im Vergleich zu | Mehrfach Ausfall | Lesegeschwindigkeit | Schreibgeschwindigkeit | Diskspace | Wiederherstellung | Kostenfaktor | Risiko Faktor | Summe der Bewertungs-kriterien | Gewichtungs-faktor |
|---|---|---|---|---|---|---|---|---|---|
| **Mehrfach Ausfall** | | 50% | 50% | 30% | 20% | 30% | 60% | 240% | 11,32% |
| **Lesegeschwindigkeit** | 50% | | 70% | 50% | 50% | 60% | 60% | 360% | 16,98% |
| **Schreibgeschwindigkeit** | 50% | 30% | | 50% | 50% | 60% | 70% | 310% | 14,63% |
| **Diskspace** | 70% | 50% | 50% | | 50% | 50% | 80% | 350% | 16,51% |
| **Wiederherstellung** | 80% | 50% | 50% | 50% | | 70% | 50% | 350% | 16,51% |
| **Kostenfaktor** | 70% | 40% | 40% | 50% | 30% | | 60% | 290% | 13,68% |
| **Risiko Faktor** | 40% | 40% | 30% | 20% | 50% | 40% | | 220% | 10,38% |
| **Der sich hieraus ergebende Umrechnungsfaktor → 2120 / 100 = 21,20** | | | | | | | | 2120% | 100.01% |

*Tabelle 7: Gewichtungsberechnung zur Wahl eines geeigneten RAID – Levels*

In der Tabelle zur Gewichtungsberechnung der Kriterien wurden die Kriterien aus den senkrechten Spalten mit denen in den waagerechten Zellen verglichen. Hierbei sind die hellgrau markierten Werte die jeweiligen Restwerte zu den Rot markierten. Die Bewertung stellte immer die Frage, was ist wichtiger aus der Perspektive vom waagerechten Kriterium zum senkrechten Kriterium. Je höher der Wert des waagerechten Kriteriums in der jeweiligen Zelle ist (% - Prozentual), desto wichtiger ist der Punkt im direkten Vergleich.

### 6.5.2.3 Bestimmung des Zielerreichungsfaktors (ZEF)

Zur Bestimmung des Zielerreichungsfaktors wird ein Bewertungs – Range von 1 – 5 angewendet:

- **1** → Das Kriterium lässt sich durch den gewählten RAID – Level nicht erfüllen.
- **2** → Das Kriterium lässt sich durch den gewählten RAID – Level nur mit erheblichen Schwierigkeiten erfüllen.
- **3** → Das Kriterium lässt sich durch den gewählten RAID – Level erfüllen. Es ist aber nicht die optimale Lösung.
- **4** → Das Kriterium lässt sich durch den gewählten RAID – Level erfüllen.
- **5** → Das Kriterium lässt sich durch den gewählten RAID – Level erfüllt alles mit optimalen Werten.

| Bewertungskriterien | RAID 5 | RAID 10 |
|---|---|---|
| **Mehrfach Ausfall** | **1** | **5** |
| **Lesegeschwindigkeit** | **5** | **5** |
| **Schreibgeschwindigkeit** | **4** | **3** |
| **Diskspace** | **5** | **1** |
| **Wiederherstellung (Einfachheit)** | **4** | **2** |
| **Kostenfaktor (bezogen auf Wiederherstellung)** | **4** | **1** |
| **Risiko Faktor** | **4** | **5** |

*Tabelle 8: Bestimmung des Zielerreichungsfaktors für die Nutzwertanalyse zur Wahl eines geeigneten RAID – Levels*

Die Zielerreichungsfaktoren in der Tabelle 8 wurden nach dem Wissensstand und der Erfahrung des Autors definiert. Hier flossen auch Annahmen bezüglich der Erfahrung und der zur Verfügung stehenden Hardware ein. Solche Annahmen sind bspw. die Schreibgeschwindigkeit vom RAID 10, der sicherlich eine enorme Performance durch den RAID 0 besitzt. Jedoch kann diese Performance nur durch entsprechend schnelle Hardware realisiert werden. Die beiden HP Microserver sind mit ihren leistungsarmen Intel Celeron und den 2 GB RAM nach Annahme und Erfahrung nicht dazu geeigneten einen Software RAID 1 und 0 praktisch zeitnahe so zu berechnen, dass sie nicht die Prozessoren an ihr Limit treiben und somit die Gesamtperformance über längere Zeit darunter leidet.

#### 6.5.2.4 Ermittlung des Endergebnisses anhand der vorliegenden Fakten

| Bewertungskriterien | GWF | RAID 5 | | RAID 10 | |
|---|---|---|---|---|---|
| | | ZEF | Ermitt. Wert | ZEF | Ermitt. Wert |
| Mehrfach Ausfall | 11,32% | 1 | 11,32% | 5 | 56,60% |
| Lesegeschwindigkeit | 16,98% | 5 | 84,90% | 5 | 84,90% |
| Schreibgeschwindigkeit | 14,63% | 4 | 58,52% | 3 | 43,89% |
| Diskspace | 16,51% | 5 | 82,55% | 1 | 16,51% |
| Wiederherstellung (Einfachheit) | 16,51% | 4 | 66,04% | 2 | 33,02% |
| Kostenfaktor (bezogen auf Wiederherstellung) | 13,68% | 4 | 54,72% | 1 | 13,68% |
| Risiko Faktor | 10,38% | 4 | 41,52% | 5 | 51,90% |
| Gesamtnutzwert | | | 399,57% | | 300,50% |

*Tabelle 9: Ermittlung eines Siegers der Nutzwertanalyse zur Wahl eines geeigneten RAID – Levels*

### 6.5.3 Auswertung des Resultats

Wie aus Tabelle 9 ersichtlich ist, wird ein RAID – Level 5 für die Storage – Systeme gewählt. Diese Lösung erfüllt die geforderten Kriterien am besten und ist auch bei einer nüchternen, etwas entfernteren Betrachtung des Gesamtergebnisses im Verhältnis zu den Kosten auch die beste Lösung. Zwar geht diese Lösung in den heute üblichen Trend auf volles Risiko zu setzen, jedoch wäre die Ideallösung mittels absoluter Sicherheit (RAID 10) mit Unmengen von Kosten verbunden gewesen.

### 6.5.4 Reelle Umsetzung

Zur Umsetzung des Ergebnisses werden pro HP – Microserver alle vier HD – Slots mit den alten Festplatten des alten Clusters bestückt. Anschliessend wird ein RAID 5 über alle vier Platten gebildet, wobei die Paritätsdaten verteilt auf allen Platten abgelegt werden. Hier eine separate Platte für die Paritätsdaten zu nutzen würde klar an Verschwendung grenzen und den Sinn dieser Nutzwertanalyse aufs äusserste in Frage stellen. Die genaue Umsetzung des RAID – Levels wird unter dem entsprechenden Punkt 8 „Realisierung“ erklärt.

### 6.5.4.1 Kurzer Exkurs in Richtung Chunk Size

Aufgrund von Recherchen die bereits an dieser Stelle gemacht wurden, soll auch gleich die Erklärung der gewählten Chunk – Size erfolgen. Zwar geht es bei der Klärung der Storge – Frage grösstenteils um Redundanz, jedoch soll an dieser Stelle auf analysebezoge Redundanz verzichtet werden. Eine → Google – Anfrage führte zu nachfolgendem Weblink:

http://louwrentius.com/linux-raid-level-and-chunk-size-the-benchmarks.html

Hier sind, wie in nachfolgender Grafik zu erkennen ist, bereits diverse Benchmarks mit unterschiedlichen Szenarien durchgeführt worden. Darunter ist auch ein Benchmark, welcher die ermittelte Lösung und die damit verbundene 4 – Disk Problematik beschreibt. Die Hardwarespezifikationen stimmen natürlich nicht überein, jedoch trifft das Grundprinzip vom Aufbau her zu. Wie aus der Grafik zu entnehmen ist, lässt sich die höchste Performance mit einer Chunk – Size von 32 Kilobyte erzielen. Somit ist eine 8x grössere Blockgrösse der Software – RAID – Tabelle notwendig als es der Normalfall wäre, um die Performance zu optimieren. Diese Werte klingen logisch, da bei einer grösseren Blockgrösse auch weniger Paritätsdaten berechnet werden müssen und somit die Prozessorlast auch rapide sinkt. Dies hält mehr Ressourcen frei, welche dann dem Software – RAID zur Verfügung gestellt werden.

Aufgrund der als seriös wirkenden Tests auf besagter Seite, wurden die Werte als akzeptabel und fremd getestet  eingestuft und somit übernommen.

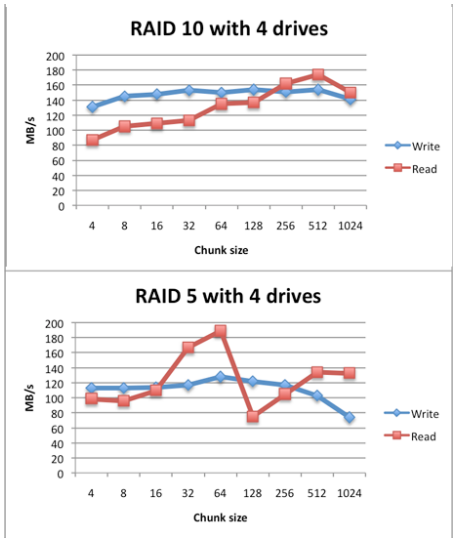



*Abbildung 7: Vergleich von RAID 5 und 10 mi 4 Disks @Quelle: http://louwrentius.com/linux-raid-level-and-chunk-size-the-benchmarks.html*

Achtung: Dieses Bild wurde zurechtgeschnitten um die wichtigsten Vergleiche zu zeigen. Das Original ist auf der Website und dem beiliegenden Datenträger zu Finden.

## 6.6 Recherche und Marktanalyse

Sucht man spezifisch nach Unternehmen die oVirt einsetzten, so kommt man direkt auf die oVirt Seite namens „oVirt Case Studies“ auf welcher die grössten Nutzer zu finden sind. Dies sind ein paar Universitäten und einige grössere Hoster (insgesamt 9). Nun könnte man ja das Produkt aufgrund der geringen Nutzerzahl in Frage stellen. Doch darf man nicht vergessen, dass oVirt die Entwicklerplattform von RedHat ist und somit der gesamte entwickelte Code von oVirt in seiner Stable – Form zurück zum RedHat Virtualization Cluster fliesst.

Betrachtet man die doch etwas sicherere Konfiguration dieses Projektes und ist bereit an einigen Stellen Abstriche zu machen, so kann seitens des Autors dieser Arbeit in Bezug auf die private Nutzung von oVirt eine gute Marktlage vorhergesagt werden. Der Autor ist auch der Meinung, dass wenn man mehr in das Marketing und die geschäftsbezogene Werbung investierten würde, man auch kleinere Unternehmen (hier speziell KMU's) dazu anreizen könnte selbst ihre Services zu betreiben, anstelle all ihrer sensiblen Daten out zu sourcen. Wie gesagt, minimale Abstriche in Bezug auf ein solches Referenzprojekt wie dieses und die Verwendung von bestehender Hardware in Kombination mit einem kostenlosen Projekt wie oVirt, würde nur noch Support- und Unterhaltskosten bedeuten.

# 7 Hauptstudie mit Konzeptvarianten

## 7.1 Zweck und Umfang dieser Hauptstudie

Wie die Vorstudie orientiert sich auch die Hauptstudie den Gegebenheiten des bestehenden Environment und dem Auftrag. Hier sollen die softwareseitigen Faktoren geklärt werden, welche vor Beginn der Realisierung definiert werden müssen. Auch hier gilt; Änderungen im nach hinein sind nur schwer oder überhaupt nicht mehr möglich. Innerhalb dieses Kapitels sollen zwei elementare Fragen geklärt werden, wobei jede Frage für sich eine Umsetzungsvariante darstellt. Die zu klärenden Fragen betreffen hier das Herzstück des gesamten Clusters und zwar oVirt selbst.

Die beiden zu klärenden Punkte, welche sich in jeweils zwei mögliche Umsetzungsvarianten splitten sind:

- **Entscheidungsfindung 1:** Welche Version von oVirt soll integriert werden?
  - oVirt 3.5; alt aber erprobt
  - oVirt 3.6; neu und umfangreicher
- **Entscheidungsfindung 2:** Separierung der Cluster oder doch einfach einen grossen?
  - Sollen sich die Cluster den unterschiedlichen Prozessor – Familien anpassen
  - … oder reicht ein grosser Cluster der die Familien vereint?

Zur Klärung dieser beiden Fragen, welche dem Autor dieser Arbeit doch einiges an Kopfzerbrechen bereitet haben, soll wieder auf das bewährte und nüchterne Verfahren einer Nutzwertanalyse gesetzt werden. Dies garantiert bei bedachter und zukunftsorientierter Wahl der Kriterien, eine auf mathematische Entscheidungsfindung beschränkte und von Emotionen eher befreite Lösung.

## 7.2 Entscheidungsfindung 1

In den nachfolgenden Subpunkten soll die Frage geklärt werden, welches Major – Release von oVirt implementiert erden soll.

### 7.2.1 Einleitung

Nun bedenkt man, dass die Vortests welche als Grundlage zum Einreichen des Auftrages mit Version 3.5 gedient haben, auch auf dieser Version stattgefunden haben, so könnte man den Sinn dieser Fragestellung an dieser Stelle zunächst auch in Frage stellen. Dies vor allem darum da die Versionssprünge bei oVirt meistens mit starken Veränderungen im GUI verbunden sind. Doch kann hier eine Ausnahme gemacht werde, da die neuen Features von Version 3.6 und somit die Prozeduren wie bspw. Installation (hier speziell HA – Installation), seitens oVirt als so innovativ angesehen wurden, dass sie bei diesem speziellen Versionsverhältnis wieder nach 3.5 zurückportiert wurden. Somit sind die Grundlagen der Installation praktisch bei beiden Versionen identisch. Bezogen auf die Grundfunktionen wie bspw. das User – Management oder die GlusterFS – Integration gab es in den letzten Versionen einige relativ starke Veränderungen, doch sind diese in den Versionen 3.5 – 6 im Grundaufbau praktisch identisch. Somit spielt der Grundaufbau von oVirt (inkl. Installation) an dieser Stelle keine grosse Rolle. Was aber ausschlaggebend ist und somit dringend einer genaueren

Untersuchung bedarf, sind die neuen Features die Version 3.6 mit sich bringt. Die neuen Features die sich in Version 3.6 finden lassen sind teilweise noch nicht ganz ausgereift, weswegen darauf zu achten ist, die Auswertung der neuen Features im direkten Vergleich mit der „alten“ Version in einem klaren Verhältnis zu Kriterien wie Stabilität, Produktivität und einer guten Dokumentation gegenüberzustellen.

Hieraus resultieren folgende beiden Varianten der Umsetzung:

- Variante mit Version 3.5 (nachfolgend nur noch mit **Konzept 3.5** betitelt)

  Konzept 3.5 gilt allgemein hin als erprobt und ist bei diversen Stellen auch bereits in Betrieb. Die meisten Dokumentationen die sich durch eine → Google – Anfrage ermitteln lassen sind zwar auch älterer Natur, sie wurden aber so überarbeitet, dass sie sicher auf Version 3.5 anwendbar sind. Dies gilt grösstenteils auch für die frei zugänglichen RedHat – Dokumentation, welche sich in gewissen Punkten noch auf die Version 3.4 beziehen, aber dennoch auf Version 3.5 anwendbar sind. Die Komponenten und die daraus resultierenden Konfigurationseigenschaften richten sich stark nach den Möglichkeiten des KVM Environments, was auch den Wünschen des Autors entspricht.

- Variante mit Version 3.6 (nachfolgend nur noch mit **Konzept 3.6** betitelt)

  Konzept 3.6 kam als Finale – Release gerade noch so knapp vor einem möglichen Realisierungs- Beginn dieser Arbeit heraus. Es bringt einige kleinere Verbesserungen gegenüber 3.5 mit. Die beiden wichtigsten Features welche dem Autor gleich ins Auge gestochen sind, waren die teilweise Verbesserung der VMware ESXi Konvertierungs-Unterstützung und das Hardware – Passthroughing. „Hardware“ - Passthrough deswegen, weil mittlerweile PCI, SCSI und usb_devices bequem und nur über das GUI konfigurierbar sind. Die ESXi – Konvertierung nutzt nun die Erweiterung v2v (VMware2oVirt) um über einen „External Provider“ sich mit einem ESXi – Host zu verbinden und die Virtual – Disk in den oVirt Storage zu holen. Dies ist zwar mit ein wenig  Command – Line – Arbeit verbunden, aber es läuft nicht schlecht. Der grosse Vorteil hier wäre das zukünftig möglicherweise auftretende Migrieren von ESXi Disk – Images von dritten (Freunden) in den eigenen Cluster zwecks Übernahme des Hostings.

#### 7.2.1.1 Die notwendigen Auswahlkriterien

| Kriterien | Deren Begründung bzw. Notwendigkeit |
|---|---|
| **Stabilität/Erfahrung** | Dieses Kriterium stützt sich auf die im Einsatz befindlichen Versionen und wie das Feedback der Nutzer ist. Hierbei werden als Indikatoren die Kommentare der User in den unterschiedlichen Foren betrachtet und in welche Richtung ihre Reaktionen in welcher Anzahl zu welcher Version tendieren. **Dieses Kriterium hat beide Lösungen beeinflusst .** |
| **„LTS“** | Es gibt einige zur Zeit im Einsatz befindliche Lösungen. Die Modifikationen die aber in den hier analysierten Versionen vorkommen, weichen stark von den älteren Versionen ab. Es ist anzunehmen, dass das Design von 3.5 als Standard für eine lange Zeit dienen wird. Da es aber auch Modifikationen in 3.6 gab die in 3.5 nicht mehr vorkommen oder anders umgesetzt sind (v2v und Passthroughing ausgeschlossen), soll mit reinem Bauchgefühl versucht werden diese Unterschiede zu vergleichen. Daraus kann abgeschätzt werden, ob es notwendig wird zwingend den neuesten Versionen zu folgen (gleich beginnend mit 3.6), oder sich zurückzulehnen und mit der Version 3.5 bis zum Release – Ende (und darüber hinaus) zu arbeiten.<br>**Dieses Kriterium kann beide Lösungen beeinflusst.** |
| **Dokumentation** | Bei für den Autor neuen Produkten ist eine saubere Dokumentation manchmal von grösster Wichtigkeit. Version 3.6 ist zwar erst seit kurzem verfügbar, sie ist aber als „Produktiv“ gekennzeichnet. oVirt ist in solchen Situationen relativ schnell was das Thema Doku anbelangt. Dennoch ist für diese Arbeit entscheidend, dass einige Themen jetzt schon erklärbar sind und nicht erst in 2 – 3 Monaten nach der Abgabe dieser Dokumentation. Hier soll eine ungefähre Abschätzung getroffen werden, wie die Situation mit der 3.6 – Dokumentation im Moment und bei Abgabe dieser Arbeit sein wird.<br>**Dieses Kriterium kann beide Lösungen beeinflusst.** |
| **Konvertierung (v2v)** | Die Konvertierung von VMware Disks zu oVirt- kompatiblen ist sicherlich ein nützliches Feature. Hier stellt sich einfach die Frage ob man sich Verblenden lässt aufgrund einer neuen Technologie, wo einfach mal wieder die Augen grösser sind als der reale Nutzen, oder ob es in Zukunft absolut notwendig ist und auch massiv zum Einsatz kommen soll. Dieser Punkt stellt eine ungefähre Einschätzung des Autors da, wie gross eine mögliche Anzahl an Kunden sein wird, wo der Einsatz v2v notwendig wird. Hier soll noch erwähnt werden, dass v2v nicht nur für VMware, sondern auch für reine XEN (nicht Xenserver) Hypervisor einsetzbar ist.<br>**Dieses Kriterium beeinflusst stark das Konzept 3.6** |
| **Passthroughing** | Im allgemeinen muss auch hier die Auge zu Nutzen – Rechnung genau betrachtet werden. Zwar ist in der Server – Virtualisierung dies ein mehr als nutzloses Feature wenn man bedenkt, dass niemand ausser dem Admin realen Zugang zum Cluster hat, jedoch würde es dem Autor mehr als nur gefallen, wenn er ganze Grafikkarten zum Zwecke des Gamens einer Windows VM zuteilen könnte.<br>**Dieses Kriterium beeinflusst stark das Konzept 3.6** |
| **Risikofaktor** | Dies ist ein rein emotionaler Faktor, den der Autor dieser Arbeit ins Spiel bringt um eine Abschätzung bezüglich der Risiken folgender beider Punkte besser abwägen zu können:<br>• Besteht ein Risiko in den Versionen stecken zu bleiben bei der Verwendung mit Version 3.5 (gilt immerhin als alt).<br>• Besteht ein Risiko möglicherweise eine als produktiv eingestufte, aber in Wirklichkeit noch nicht fertige Version 3.6 zu wählen. Denn zum Zeitpunkt dieser Analyse war in diversen Foren bereits die Rede von Version 3.6.1, was bei oVirt mehr als ungewöhnlich ist.<br>**Dieses Kriterium kann beide Lösungen beeinflusst.** |

*Tabelle 10: Kriterien für die Nutzwertanalyse Betreff Konzept 3.5 oder Konzept 3.6*

### 7.2.1.2 Ermittlung des Gewichtungsfaktors für die einzelnen Kriterien (GWF)

| Im Vergleich zu | Stabilität/Erfahrung | „LTS“ | Dokumentation | Konvertierung (v2v) | Passthroughing | Risikofaktor | Summe der Bewertungs-kriterien | Gewichtungs-faktor |
|---|---|---|---|---|---|---|---|---|
| **Stabilität/Erfahrung** | | **50%** | **50%** | **60%** | **60%** | **40%** | **260%** | **17,33%** |
| **„LTS“** | 50% | | **50%** | **70%** | **70%** | **50%** | **290%** | **19,33%** |
| **Dokumentation** | 50% | 50% | | **60%** | **60%** | **30%** | **250%** | **16,66%** |
| **Konvertierung (v2v)** | 40% | 30% | 40% | | **20%** | **50%** | **180%** | **12,00%** |
| **Passthroughing** | 40% | 30% | 40% | 80% | | **40%** | **230%** | **15.33%** |
| **Risikofaktor** | 60% | 50% | 70% | 50% | 60% | | **290%** | **19,33%** |
| **Der sich hieraus ergebende Umrechnungsfaktor → 1500 /100 = 15** | | | | | | | **1500%** | **99.98%** |

*Tabelle 11: Gewichtungsberechnung zur Wahl eines geeigneten Konzeptes (3.5 vs. 3.6)*

In der Tabelle zur Gewichtungsberechnung der Kriterien wurden diese aus den senkrechten Spalten mit denen in den waagerechten Zellen verglichen. Hierbei sind die hellgrau markierten Werte die jeweiligen Restwerte zu den rot markierten. Die Bewertung stellte immer die Frage, was ist wichtiger aus der Perspektive vom waagerechten Kriterium zum senkrechten Kriterium. Je höher der Wert des waagerechten Kriteriums in der jeweiligen Zelle ist (% - prozentual), desto wichtiger ist der Punkt im direkten Vergleich.

#### 7.2.1.3 Bestimmung des Zielerreichungsfaktors (ZEF)

Zur Bestimmung des Zielerreichungsfaktors wird ein Bewertungs- Range von 1 – 5 angewendet:

- **1** → Das Kriterium lässt sich mit dem Konzept nicht erfüllen.
- **2** → Das Kriterium lässt sich mit dem Konzept nur mit erheblichen Schwierigkeiten erfüllen.
- **3** → Das Kriterium lässt sich mit dem Konzept erfüllen. Es ist aber nicht die optimale Lösung.
- **4** → Das Kriterium lässt sich mit dem Konzept erfüllen.
- **5** → Das Kriterium lässt sich mit dem Konzept erfüllen, es ist auch die beste Wahl überhaupt.

| Bewertungskriterien | Konzept 3.5 | Konzept 3.6 |
|---|---|---|
| **Stabilität/Erfahrung** | **5** | **3** |
| **„LTS“** | **5** | **4** |
| **Dokumentation** | **5** | **3** |
| **Konvertierung (v2v)** | **2** | **4** |
| **Passthroughing** | **2** | **5** |
| **Risikofaktor** | **4** | **2** |

*Tabelle 12: Bestimmung des Zielerreichungsfaktors für die Nutzwertanalyse zur Wahl eines geeigneten Konzeptes (3.5 vs. 3.6)*

Wie bereits in Tabelle 10 erwähnt, sind hier einige Faktoren wie bspw. möglicher Anzahl zu migrierender VMware – Images oder der effektive Einsatz von Passthroughing als reine Bauchgefühlsschätzung seitens des Autors hineingeflossen. Genaue Zahlen sind an dieser Stelle nicht oder noch nicht möglich. Daher wird dieser Faktor hier nur erwähnt, aber nicht genauer spezifiziert.

#### 7.2.1.4 Ermittlung des Endergebnisses anhand der vorliegenden Fakten

| Bewertungskriterien | GWF | Konzept 3.5 | | Konzept 3.6 | |
|---|---|---|---|---|---|
| | | ZEF | Ermitt. Wert | ZEF | Ermitt. Wert |
| Stabilität/Erfahrung | 17,33% | 5 | 86,65% | 3 | 51,99% |
| „LTS“ | 19,33% | 5 | 96,65% | 4 | 77,32% |
| Dokumentation | 16,66% | 5 | 83,30% | 3 | 49,98% |
| Konvertierung (v2v) | 12,00% | 2 | 24,00% | 4 | 48,00% |
| Passthroughing | 15,33% | 2 | 30,66% | 5 | 76,65% |
| Risikofaktor | 19,33% | 4 | 77.32% | 2 | 38,66% |
| Gesamtnutzwert | | | 398,58% | | 342,60% |

*Tabelle 13: Ermittlung eines Siegers der Nutzwertanalyse zur Wahl eines Konzeptes (3.5 vs. 3.6)*

### 7.2.2 Auswertung des Resultats

Wie in Tabelle 13 zu sehen ist, hat das altbewährte Konzept knapp gesiegt. Die Kriterien und deren Gewichtung waren zwar so ausgelegt, dass auch ein kleiner „touch“ an neueren Funktionen dabei war, jedoch lag der Fokus klar auf langer Betriebszeit. Um ehrlich zu sein passt das Ergebnis dem Autor dieser Arbeit auch gut, denn bei einer solch wichtigen Arbeit auf volles Risiko zu setzen und mit einer relativ neuen Version zu arbeiten, wäre ohne Netz und doppelten Boden doch sehr riskant gewesen. In so komplexen Cluster – Umgebungen sollte ohnehin mit stabilem Langzeitlösungen gerechnet werden, welche sich mindestens bei jemand anderem im Produktivbetrieb bewährt haben und dies ist bei 3.5 klar der Fall.

## 7.3 Entscheidungsfindung 2

In den nachfolgenden Subpunkten soll die Frage bezüglich der Cluster – Design Frage thematisiert werden.

### 7.3.1 Einleitung

Die Frage nach dem Cluster – Design Schema ist in grossen Hosting – Umgebungen sicherlich von grösster Bedeutung und bedarf der Zuwendung eines Network- oder Cluster – Engineers. Hier könnte man sich die Frage stellen, warum bei vier Nodes den so ein Theater. Der Grund hierfür ist, dass oVirt, glücklicherweise muss man auch sagen, nicht einfach einen Pseudoprozessor (qemu-kvm) generiert wie es KVM von sich aus per Default machen würde, sondern es wird versucht einen Prozessor aus einer Liste mit vordefinierten Prozessoren direkt an die VM weiterzureichen. Hierbei wird der für die Virtualisierung maximale Prozessorbefehlssatz (Ring -1) des jeweiligen Prozessors über das WebGUI der Engine definiert und weitergereicht. Kann die VM von sich aus einen hoch privilegierten Befehl an den Prozessor senden, so muss der Hypervisor nicht ein Befehlsmapping vornehmen, um stellvertretend im Namen der VM diese Aktion im Prozessor auszulösen. Nun ist es so, dass unter den vier Nodes zwei Prozessor – Familien existieren. Die beiden Fujitsu und der Subermicro nutzen Prozessoren aus der Intel SandyBridge – Familie und der leistungsstärkere Dell bewegt sich in der Westmere Familie. Nun könnte man ja sagen, der kleinste gemeinsame Nenner ist die Westmere Familie und entsprechend alle in diese packen. Dann müssten aber alle moderneren SandyBridges die Befehle, welche nicht im Befehlsumfang des Westmere wären, mühsam mappen. Dies bedeutet Overhead in grossen mengen über drei Maschinen. Die Frage ist, soll dieser Overhead bei einem Cluster dieser geringen Grösse akzeptiert werden, oder lohnt sich eine Separierung welche auch die Anzahl Maschinen pro Cluster reduzieren würde.

Hieraus ergeben sich folgende beiden Varianten:

- Variante mit nur einem grossen Cluster (nachfolgend nur noch als Konzept **OneFamily** betitelt)

  Hier würden alle vier Nodes in die kleinste gemeinsame Familie Intel Westmere gepackt. Dies würde eine massiv vereinfachte Live – Migration bedeuten, da ja alle im gleichen Cluster wären. Zusätzlich würde es die Regelsätze bezüglich Affinitätsgruppen und späterer userbezogener Verwaltungsrechte vereinfachen. Ein gewisser Overhead wird aber immer bleiben und praktisch gesprochen; es wird Wärme für nichts erzeugt. Was bei der momentanen Lage der Maschinen (bei mir irgendwo in einer Wohnungsecke) ohnehin etwas problematisch ist.

- Variante mit optimaler Verteilung der Nodes innerhalb des Clusters (nachfolgend nur noch als Konzept **ToBig** betitelt)

  Bei dieser Variante müsste man die Nodes so verteilen, dass sich die Familien so gut wie möglich untereinander einschliessen. Hier müssten aber auch die Leistungsdimensionen berücksichtigt werden, was im Klartext bedeutet, die leistungsschwachen Fujitsus welche nicht in nächster Zeit einer Hardware – Aufstockung unterzogen werden, müssten in einen eigenen Cluster. So bleiben die beiden grossen Maschinen in einem eigenen Cluster. Dies bedeutet zwar für den Supermicro, dass er Overhead erzeugen muss, da sein Prozessor aus der SandyBridge – Familie in die tiefere Westmere rutscht. Jedoch gibt es keine andere Alternative, um den Overhead der bei einer solch asymmetrischen CPU – Durchmischung entsteht auf ein Minimum zu reduzieren.

#### 7.3.1.1 Die notwendigen Auswahlkriterien

| Kriterien | Deren Begründung bzw. Notwendigkeit |
|---|---|
| **Einfachheit** | Die Lösung soll zwar das Optimum herausholen, jedoch soll das Design nicht zu komplex sein, da ja nur ein Admin (der Autor) das Management übernehmen wird. **Dieses Kriterium beeinflusst das Konzept OneFamily.** |
| **Optimale Performance** | Overhead aufgrund von zu tiefer CPU – Klassifizierung kann sicherlich bis zu einem gewissen Grad hingenommen werden. Im Idealfall sollte es in der heutigen Green – IT Gesellschaft nicht vorkommen. **Dieses Kriterium kann theoretisch beide betreffen, je nach Betrachtung.** |
| **Spätere Erweiterbarkeit** | Im Falle einer späteren Erweiterung des Virtualisierungsteils, könnte man ja beim Konzept OneFamaliy einfach einbinden ohne gross zu überlegen. Sollten aber Maschinen einer noch neueren CPU – Generation ins Spiel kommen, so müsste man sich schon Gedanken über den immer grösser werdenden Overhead machen. Möglicherweise wäre eine ToBig Strategie ratsamer für die Zukunft. **Dieses Kriterium kann theoretisch beide betreffen, je nach Betrachtung.** |
| **Anschluss – Thematik (Ethernet Ports)** | Die Fujitsu haben eine Minimalausstattung an Ethernet Ports (4x je Node). Bei einem einzigen Cluster würde jede zusätzliche VM, welche auf einen Fujitsu geschoben wird, auch der Network Traffic steigen. Dies würde Hardware – Aufstockungen zur Folge haben, welche so eigentlich nicht vorgesehen sind. **Dieses Kriterium beeinflusst das Konzept ToBig.** |
| **Systemressourcen** | Dieser Punkt betrifft auch die beiden Fujitsus. Wie bereits erwähnt ist eine Hardware – Aufstockung momentan nicht vorgesehen. Nun stellt sich die Frage, ob es Sinn machen würde die beiden Ressourcen einzubinden, wenn die Anzahl an zusätzlichen VM's pro Node gerade einmal 4 – 6 (je nach Dimension) betragen würde. **Dieses Kriterium beeinflusst das Konzept ToBig.** |

*Tabelle 14: Kriterien für die Nutzwertanalyse betreffs Konzepte OneFamily und ToBig*

### 7.3.1.2 Ermittlung des Gewichtungsfaktors für die einzelnen Kriterien (GWF)

| Im Vergleich zu | Einfachheit | Optimale Performance | Spätere Erweiterbarkeit | Anschluss – Thematik (Ethernet Ports) | Systemressourcen | Summe der Bewertungs-kriterien | Gewichtungs-faktor |
|---|---|---|---|---|---|---|---|
| **Einfachheit** | | **40%** | **50%** | **30%** | **30%** | **150%** | **15,00%** |
| **Optimale Performance** | 60% | | **50%** | **50%** | **80%** | **240%** | **24,00%** |
| **Spätere Erweiterbarkeit** | 50% | 50% | | **70%** | **50%** | **220%** | **22,00%** |
| **Anschluss – Thematik** | 70% | 50% | 30% | | **60%** | **210%** | **21,00%** |
| **Systemressourcen** | 70% | 20% | 50% | 40% | | **180%** | **18,00%** |
| **Der sich hieraus ergebende Umrechnungsfaktor → 1000 / 100 = 10** | | | | | | **1000%** | **100%** |

*Tabelle 15: Gewichtungsberechnung zur Wahl eines geeigneten Konzeptes (OneFamily vs. ToBig)*

In der Tabelle zur Gewichtungsberechnung der Kriterien wurden diese aus den senkrechten Spalten mit denen in den waagerechten Zellen verglichen. Hierbei sind die hellgrau markierten Werte die jeweiligen Restwerte zu den rot markierten. Die Bewertung stellte immer die Frage, was ist wichtiger aus der Perspektive vom waagerechten Kriterium zum senkrechten Kriterium. Je höher der Wert des waagerechten Kriteriums in der jeweiligen Zelle ist (% - prozentual), desto wichtiger ist der Punkt im direkten Vergleich.

#### 7.3.1.3 Bestimmung des Zielerreichungsfaktors (ZEF)

Zur Bestimmung des Zielerreichungsfaktors wird ein Bewertungs- Range von 1 – 5 angewendet:

- **1** → Das Kriterium lässt sich mit dem Konzept nicht erfüllen.
- **2** → Das Kriterium lässt sich mit dem Konzept nur mit erheblichen Schwierigkeiten erfüllen.
- **3** → Das Kriterium lässt sich mit dem Konzept erfüllen. Es ist aber nicht die optimale Lösung.
- **4** → Das Kriterium lässt sich mit dem Konzept erfüllen.
- **5** → Das Kriterium lässt sich mit dem Konzept erfüllen, es ist auch die beste Wahl überhaupt.

| Bewertungskriterien | Konzept OneFamily | Konzept ToBig |
|---|---|---|
| **Einfachheit** | **5** | **2** |
| **Optimale Performance** | **3** | **5** |
| **Spätere Erweiterbarkeit** | **4** | **4** |
| **Anschluss – Thematik (Ethernet Ports)** | **3** | **5** |
| **Systemressourcen** | **3** | **4** |

*Tabelle 16: Bestimmung des Zielerreichungsfaktors für die Nutzwertanalyse zur Wahl eines geeigneten Konzeptes (OneFamily vs. ToBig)*

An dieser Stelle gibt es keine weiteren Bemerkungen. Der Grossteil der Parameter ist in den Kriterien erklärt und wurde 1:1 zur Bestimmung der Zielereichungsfaktoren übernommen.

#### 7.3.1.4 Ermittlung des Endergebnisses anhand der vorliegenden Fakten

| Bewertungskriterien | GWF | Konzept OneFamily | | Konzept ToBig | |
|---|---|---|---|---|---|
| | | ZEF | Ermitt. Wert | ZEF | Ermitt. Wert |
| Einfachheit | 15,00% | 5 | 75,00% | 2 | 30,00% |
| Optimale Performance | 24,00% | 3 | 72,00% | 5 | 120,00% |
| Spätere Erweiterbarkeit | 22,00% | 4 | 88,00% | 4 | 88,00% |
| Anschluss – Thematik (Ethernet Ports) | 21,00% | 3 | 63,00% | 5 | 105,00% |
| Systemressourcen | 18,00% | 3 | 54,00% | 4 | 72,00% |
| Gesamtnutzwert | | | 352,00% | | 415,% |

*Tabelle 17: Ermittlung eines Siegers der Nutzwertanalyse zur Wahl eines Konzeptes (OneFamily vs. ToBig)*

### 7.3.2 Auswertung des Resultats

Wie aus Tabelle 17 herauszulesen ist, gewinnt das Konzept ToBig. Dies resultiert aus der starken Tendenz, den Cluster so performancereich wie nur möglich zu gestalten, bei zeitgleicher Aufrechterhaltung der späteren Erweiterbarkeit. Zugegeben wäre dem Autor an dieser Stelle das Konzept OneFamily lieber gewesen, da es die Anzahl an Fine – Tuning – Konfigurationen erheblich erleichtert hätte, doch ist diese Lösung performancetechnisch die bessere. Zugleich erleichtert sie eine spätere Erweiterbarkeit mit möglicher modernerer Hardware (zusätzliche Nodes), da sie eine breitere Anzahl an Integrationsmöglichkeiten gewährt.

## 7.4 Schlusswort zur Umsetzung

Anhand der Analysen der beiden noch offenen Fragestellungen mittels Nutzwertanalysen, ist klar das ein Gesamtkonzept angestrebt werden muss, dass auf eine solide und erprobte Basis setzt sowie die maximale Performance herausholen kann. Aus diesem Grund wird bei der Engine klar auf das in Produktivumgebungen getestete und bekannte oVirt 3.5 gesetzt. Um die geforderte Performance herauszuholen, heisst keinen grösseren Overhead zu erzeugen, soll eine Lösung gewählt werden, welche mit der zur Verfügung stehenden Hardware eine sinnvolle Separierung ermöglicht. Dazu werden die beiden Fujitsu Nodes in einen eigenen Cluster gekapselt. Die beiden Power – Maschinen werden zum Zwecke der geforderten Live – Migration und des HA ebenfalls in eigene Cluster gekapselt. Der bei den grossen Maschinen auftretende Overhead, welcher aus den unterschiedlichen CPU – Familien resultiert, kann und muss aufgrund der Asymmetrie hingenommen werden. Diese Asymmetrie ist gegeben und die hier gewählte Lösung ist die geeignetste um sie auf ein Minimum zu reduzieren.

Die genauen Spezifikationen des Lösungsweges mit sämtlichen Parametern folgen im anschliessenden Kapitel.

# 8 Realisierung / Aufbau eines virtualisierungs- Clusters

## 8.1 Zweck und Umfang der Realisation

Dieser Abschnitt soll den gesamten Prozess der Realisierung so detailreich wie möglich dokumentieren. Zugleich wird versucht, diese Dokumentation so zu designen, dass sie auch als Einrichtungsanleitung für Dritte genutzt werden kann, welche diese Lösung als Referenzimplementierung nutzen wollen. Infoboxen werden an diversen Stellen wichtige Hinweise liefern, aber auch Verzweigungen auf Alternativlösungen bieten.

Die Dokumentation ist prinzipiell in drei Sektionen gegliedert. Die eigentliche Realisierung ist aber aufgrund der unterschiedlichen Zwischenschritte in fünf Sektionen gegliedert. Die Genaue Gliederung sieht dabei wie folgt aus:

- Einrichtung und Konfiguration des Netzwerksegmentes.
- Einrichtung und Konfiguration des Storagesegmentes.
- Einrichtung und Konfiguration des Virtualisierungssegmentes.
- Zusammenführung der einzelnen Komponenten zu einem als Cluster operierenden Objekt.
- Fine – Tuning; Konfiguration der für das Layout nötigen Vorlagen, Ressourcen ggf. Benutzer etc.

Aufgrund der starken Verflechtungen der einzelnen Konfigurationsschritte kann es vorkommen, dass einige Punkte dennoch in fremden Segmenten vollzogen werden müssen. Diese Spezialfälle werden bei Bedarf hervorgehoben.

## 8.2 Allgemeine Vordefinition

Der Prozess der Verkabelung wird an dieser Stelle nicht spezifisch beschrieben, sondern als erledigt angesehen.

Die verschiedenen Installationen der Betriebssysteme werden auch nicht ausführlich erklärt. Es wird jeweils nur ein Link zur ersten deutschsprachigen Installationsanleitung hinterlegt, welche mittels →Google – Anfrage gefunden wurde. Sollten spezielle Schritte bei einer OS – Installation notwendig sein, so werden diese genauer beschrieben.

## 8.3 Spezielles in Kürze

An dieser Stelle soll ein kurzer Exkurs in Richtung eher unbedeutender, aber dennoch erwähnenswerter Punkte der realen Umsetzung erfolgen. Dies betrifft folgende beiden Punkte:

- Erstellung eigener Trägerschienen für das Rack.
- Beschriftungskonzept der Verkabelung

Die Erstellung der Trägerschienen für das Rack Marke, Eigenbau ist zwar ein simpler und kurzer Zuschneideprozess der Schienen und kleinerer Teile der Finger, ist aber dennoch Teil dieser Arbeit und soll daher erwähnt werden. Hierbei handelt es sich um einfache Winkelprofilschienen von 4x4 cm, welche auf die Länge des Racks zugeschnitten wurden. Aus den Resten der Schienen wurden kleine Winkel zugeschnitten, die als Halterungen an den Ende mit Schrauben fixiert wurden. In nachfolgender Abbildung ist eine solche Montageschiene in montiertem Zustand ersichtlich.

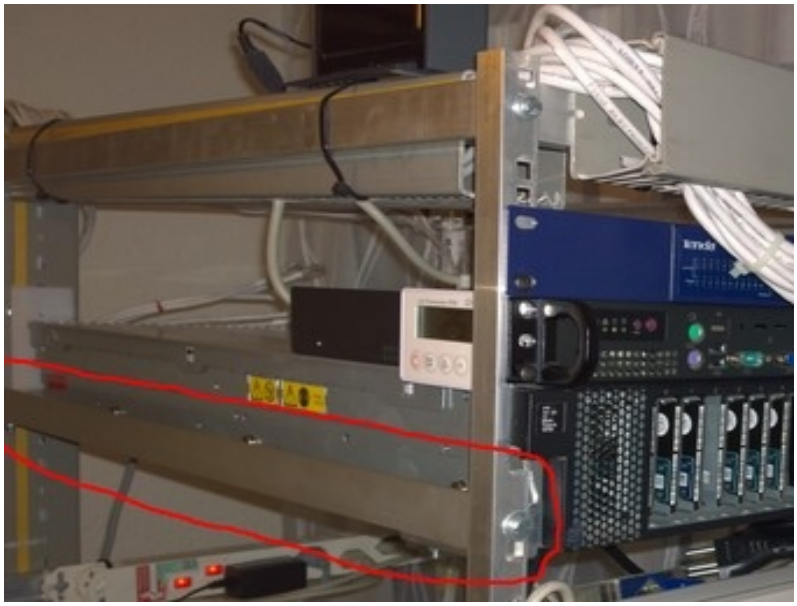


*Abbildung 8: Selber erstellte Montageschiene für das Rack*

Zur Beschriftung der Kabel wurde ein positions- und funktionsabhängiges Verfahren gewählt. Diese Beschriftung ist jeweils auf beiden Seiten gleich und orientiert sich an den Anschlüssen seitens der Computer – Anschlüsse. In nachfolgender Abbildung ist ein Beispiel ersichtlich:

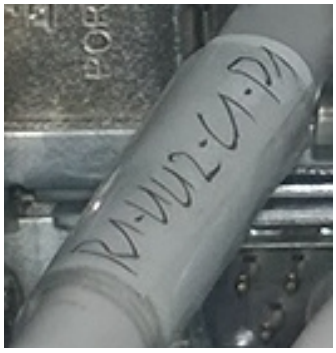



*Abbildung 9: Beispiel einer Kabelbeschriftung mit dem angewendeten Beschriftungskonzept.*

Wie im oberen Beispiel zu sehen ist, umfasst die Beschriftung vier Segmente.

R1-UU2-C1-P1

Hierbei arbeitet sich die logische Reihenfolge von links nach recht, äquivalent zum eigentlichen Objekt durch. Dies bedeutet:

- R1 → Steht für das Rack, in dem sich die Maschine befindet.
- UU2 → Sagt aus, dass es sich um eine Universal Unit handelt, welche die Position 2 im Rack von unter her gezählt handelt.
    - UU → Universal Unit ist meistens ein Hypervisor – Host, kann aber auch eine andere Funktion haben, welche sich nicht mit einer der anderen Hauptfunktionen des Clusters überschneidet.
    - SU → Storage Unit ist die reguläre Bezeichnung für eine Clusterkomponente, welche NUR Cluster Storage zur Verfügung stellt.
- C1 → Card 1 beschreibt die Position des Kabels in Bezug auf die Netzwerkkarte. Hier ist dies die erste Karte (PCI Steckplatz 1). Die Zahl steigt äqivalent mit der Position der Karte in Bezug auf die PCI – Steckposition.
    - CO → Card Onboard beschreibt die sich auf dem Mainboard befindlichen Ports als gesamtes.
- P1 → Port 1 beschreibt die Position des Kabels innerhalb der Netzwerkkarte selbst, also den Ethernet Port. Dabei gilt; senkrecht stehende Netzwerkkarten werden von unten nach oben gezählt, waagrecht stehende Karten werden von hinten betrachtet immer von links nach rechts gezählt.

Diese Beschriftung erlaubt aufgrund ihrer Positionsabhängigkeit das schnelle finden der einzelnen Komponenten von der Firewall oder dem Switch aus. Zusätzlich bleibt das Auswechseln der Komponenten möglich, da eine allfällige neue Komponente den Platz der alten einnimmt und somit die Beschriftung erhalten bleibt.

## 8.4 Beginn mit der Realisation des Netzwerksegmentes

In diesem Abschnitt sollen alle notwendigen Schritte im Top – Down Verfahren aufgezeigt werden, welche notwendig sind, um das Netzwerksegment in einer Einzelbetrachtung aufzusetzen und zu konfigurieren. Dieser Teil umfasst folgende beiden Kernpunkte:

- pfSense; Installation, Konfiguration, Bündelung der Interfaces in Link Aggregation, Bildung einer Bridge und Erstellen erster Firewall Regeln.
- Switch; Hier speziell nur die Montage und die Verkabelung.

### 8.4.1 Die verwendete Hardware

Zum Realisieren der Firewall mittels pfSense wird ein alter und gebrauchter Server mit folgenden Spezifikationen verwendet:

- IBM System x3650 M2
  - Dual CPU, Intel Xeon E5520, 2,27GHz, 16 Cores Total
  - RAM → 24 GB
  - 4x Intel Server Netzwerkkarte – 4x Port je Karte → Total 16 + 2 (Onboard) = 18 Ports

Für das Management – Network, welches die Nodes (inkl. Storage) miteinander verbindet, wird ein simpler, ungemanagter Switch verwendet. Die Trennung von der Cluster – Kommunikation zum restlichen Netzwerk, aber speziell vom Netzwerk der VM's, soll auf diese Weise eine 100%-ige Isolierung ermöglichen. Der hierfür verwendete Switch hat folgende Spezifikationen:

- Tenda TEG1024D
  - 24-Port Gigabit Switch

Zur Verkablung aller Komponenten wird ein Standard CAT 6 Patchkabel mit RJ45 Steckern verwendet. Die Kabel konnten mit optimaler Länge verlegt werden, da die Montage des RJ45 durch den Autor selbst Vorgenommen wurde.

### 8.4.2 Beginn mit der Installation und Konfiguration von pfSense

Die Installation von pfSense selbst ist eine der einfachsten Sachen überhaupt. Der Start von der Installations- CD führt einen direkt in ein „ncurses“ gestaltetes Menü, welches einen neben ersten elementaren Fragen bezüglich Tastatur – Layout (deutsches Layout nicht vorhanden) gleich in ein wichtiges Menü führt. Hier kann einfach „Quick/Easy Install“ gewählt werden. Man muss nur noch bestätigen, dass der Installer automatisch partitioniert soll und dabei alle bestehenden Daten auf der Festplatte löschen kann. Die nächste relevante Frage seitens Installer ist, ob ein Standard – Kernel oder ein Embedded – Kernel gewählt werden soll. Da hier ein Standard x86_64 Prozessor verwendet wird, muss Standard – Kernel gewählt werden. An dieser Stelle muss nur noch auf das Installation – Ende gewartet werden. Nach einem Reboot, startet pfSense in die typische Auswahlkonsole, welche einige Konfigurationsmöglichkeiten bezüglich Interfaces, Adresseingaben und einige WebGUI bezogene Einstellungen ermöglicht. In der Auswahlkonsole wählt man nun die Option 1 „Assign Interfaces“. Hier fragt der Assistent zuerst nach den Parametern der WAN Schnittstelle und bietet auch gleich an, das

LAN Interface zu konfigurieren. Die Konfiguration des LAN Interfaces muss an dieser Stelle klar abgelehnt werden. Der Grund ist, dass pfSense beim Erkennen einer LAN – Schnittstelle auch gleich die Anti – Lockout Rule ins LAN schreibt und das WAN per Default auf Port 80/443 sperrt. Der Autor dieser Arbeit hatte aber bei älteren Versionen von pfSense schlechte Erfahrungen in Bezug auf die LAN Konfiguration aus der Auswahlkonsole gemacht und nutzt lieber das GUI. Auf diese Art kann das WAN Interface zum Login und zur anschliessenden Konfiguration genutzt werden. Loggt man sich mit einem aktiven Interface ein (WAN), so kommt man direkt zum Wizard. Dieser stellt einige elementare Fragen zu den Parametern wie IP – Adresse, NTP – Server, DNS – Server etc. Hier ist Hauptsächlich die IP – Adresse (oder DHCP) von entscheidender Bedeutung, alles andere kann theoretisch auf den jeweiligen Standardwerten belassen oder leer gelassen werden. Nach Beendigung des Wizards kommt man zum Steuer – GUI.

ACHTUNG!

**Es empfiehlt sich bei dieser Methode eine Anti-Lockout Rule auf das WAN zu legen. Dem Autor dieser Arbeit geschieht es auch heute noch, dass er das LAN einbindet und aktiviert, aber vergisst, eine IP zu vergeben. So sperrt man sich auf peinlichste Art komplett aus.**

Nach Erstellung der WAN bezogenen Anti – Lockout Rule kann das LAN Interface alloziert und eine geeignete IP inkl. Subnetz vergeben werden. Dieser Teil ist nicht relevant für die Arbeit. Er gefährdet die Sicherheit der eigenen privaten Infrastruktur, weswegen die LAN Konfigurationserläuterung an dieser Stelle abgebrochen wird.

Auf die Erklärung der restlichen Parameter wird an dieser Stelle verzichtet, da sie für den weiteren Verlauf dieser Arbeit nicht relevant sind.

### 8.4.2.1 Konfiguration der Link Aggregationen

#### 8.4.2.1.1 Erstellen der Link Aggregation

Als Arstes müssen die Link Aggregationen erstellt werden, welche den Virtualisierungsnodes bzw. den VM's die Netzwerkkonnektivität garantieren. Hierzu sollen die beiden grossen Maschinen mit je drei Ports nach IEEE 802.3ad gebündelt werden. Dies ist unter pfSense bzw. über das WebGUI mehr als einfach. Um in das entsprechende Konfigurationsmenü zu gelangen, klickt man sich wie nachfolgend beschrieben durch:

```
Interfaces → (assign) → LAGG
```

*Befehl 1: Realisierung des Netzwerksegmentes (Klicken): Erstellung einer Link Aggregation mit LACP als Algorithmus.*

Hier wählt man das Hinzufügen – Symbol aus. In der Liste **Parent Interface** klickt man auf die Interface Namen, welche man zur neuen Aggregation hinzufügen möchte. Anschliessend sucht man sich in der Liste **Lag proto** das Verfahren aus, nach dem die Last verteilt werden soll. In diesem Fall LACP, da wir eine intelligente Lastverteilung wollen, die sich kontinuierlich mit den Nachbarn

abspricht. Ein kleiner Kommentar unter **Description** ist immer zu empfehlen. Mit einem Klick auf **Save** ist die ganze Arbeit eigentlich schon getan. Dieses Verfahren nach IEEE 802.3ad verspricht die grösste Performance, da beide Seiten untereinander stets die Last optimal aufteilen. Das exakte Verfahren, nach dem diese Berechnungen stattfinden, ist nicht bekannt. Dies spielt aber ohnehin keine Rolle, da es auf keiner Seite beeinflussbar ist. Nachfolgend, des besseren Verständnisses halber, ein Screenshot des Konfigurationsmenüs:

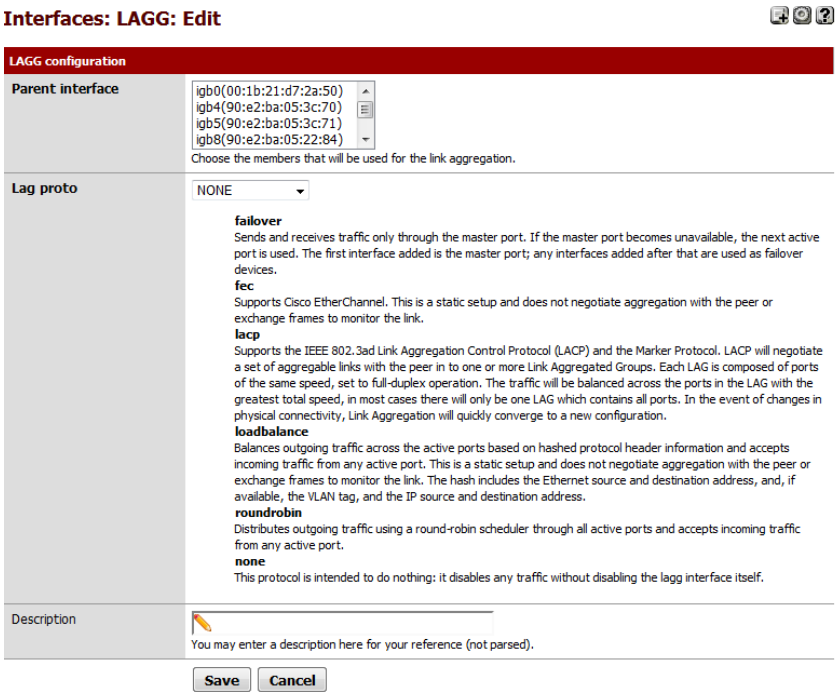



*Abbildung 10: Realisierung des Netzwerksegmentes: Erstellung einer Link Aggregation auf pfSense*

#### 8.4.2.1.2 Einbinden der Link Aggregation

Bei pfSense ist es so, dass das Erstellen eines Objektes nicht bedeutet, dass es auch aktiv ist. Man muss es stets in einem zweiten Schritt dem System (hier speziell der pfSense – Engine) auch bekanntgeben. Hier der Klick – Weg zum Konfigurationsmenü:

```
Interfaces → (assign) → Interface assignments
```

*Befehl 2: Realisierung des Netzwerksegmentes (Klicken): Einbinden der Link Aggregation in System*

Auf dieser Seite findet man eine Liste mit den bereits integrierten Interfaces, darunter auch WAN und LAN. Hier findet man am Ende eine Auswahlliste unter **Available network port,** wo man die LAGG Interfaces (Description ist auch ersichtlich) auswählen kann. Mit einem Klick auf das kleine Hinzufügen Symbol wird das neue Interface erstellt. Nun muss man den neuen Interface Namen anklicken um auf die Konfigurationsseite zu gelangen. Hier müssen folgende Einstellungen vorgenommen werden:

- **Enable** → Das neue Interface muss eingeschaltet werden.
- **Description** → Hier unbedingt einen aussagekräftigen Namen vergeben. Dieser wird später auch überall als Auswahlname ersichtlich sein.
- **IPv4 Configuration** → Muss auf **None** stehen. Da dieses Interface später Teil einer Bridge sein wird, braucht es keine IP.
- **IPv6 Configuration** → Sollte auch dringendst auf **None** stehen, falls kein IPv6 verwendet wird (Sicherheitslücke).

Dies war es schon, die beiden Link Aggregationen sind nun aktiv und im Dashboard bereits als „nicht angeschlossen“ ersichtlich.

Nachfolgend ein Screenshot einer Referenzkonfiguration:

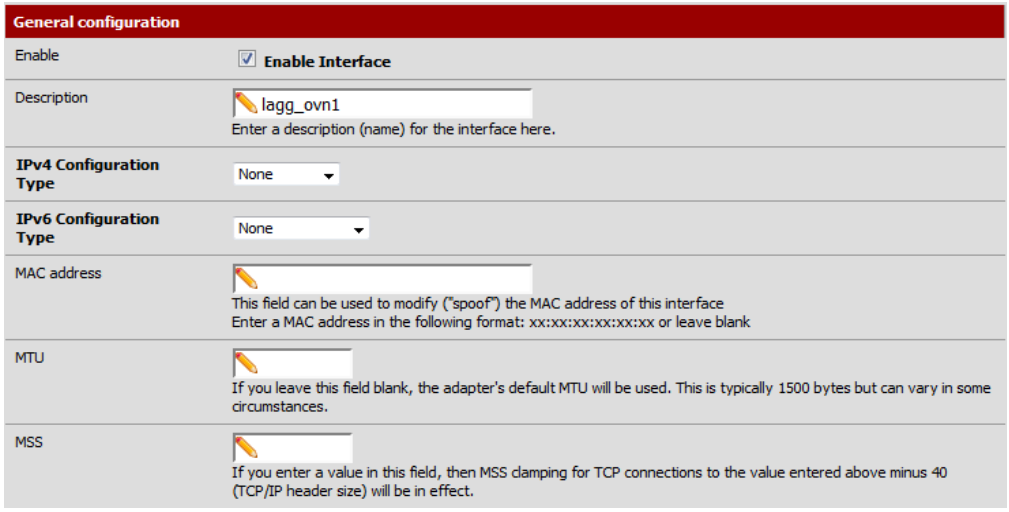



*Abbildung 11: Realisierung des Netzwerksegmentes: Einbinden der Link Aggregation ins System (aktivieren)*

Warum aber drei gebündelte Ports? Da beide Maschinen dieser Grössenklasse je acht Ethernet Ports besitzen und mindestens zwei für die interne Kommunikation wegfallen, bleiben nach Erstellung der LACP Link Aggregation noch drei übrig. Sollten noch spezifische Separierungen (netzwerkseitige Isolierung) von bspw. hoch sicheren Maschinen in der Zukunft notwendig werden, würden immer noch drei Ports zur Verfügung stehen. Es kann aber auch sein, dass die Endleistung (Traffic) nicht den gefühlten Vorstellungen entsprechen würde, dann wäre, wie schon gesagt, immer noch Spielraum nach oben.

### 8.4.2.2 Konfiguration der Einzelinterfaces

Als Nächstes müssen noch die pfSense spezifischen Interfaces der lastungsarmen Fujitsu Maschinen konfiguriert werden, welche Bestandteil des grossen „Switches“ werden sollen. Hier genügt je ein Interface pro Node. In der nachfolgenden Box ist der Klick – Weg zum Konfigurationsmenü ersichtlich:

```
Interfaces → (assign) → Interface assignments
```

*Befehl 3: Realisierung des Netzwerksegmentes (Klicken): Einbinden einzelner Interfaces ins System.*

Da hier keine speziellen Pseudointerfaces im Vorfeld erstellt werden müssen, genügt die Bekanntgabe der neuen Interfaces in Richtung pfSense. Dieser Schritt ist äquivalent zum vorhergegangenen Punkt 8.4.2.1.2 „Einbinden der Link Aggregation“. Hier kann man einfach das gewünschte **reale** Interface bspw. **igb6** in der Auswahlliste wählen und das in pfSense manchmal schlecht ersichtliche Hinzufügen Symbol anklicken. Im nun erscheinenden Konfigurationsmenü, welches wie jenes in Abbildung 11 aussieht, muss man wieder einen aussagekräftigen Namen wählen. Eine IP – Adresse ist auch hier nicht notwendig.

Dies war es auch schon mit der Einrichtung der Interfaces, welche später den Virtualisierungsnodes bzw. den VM's als Netzwerk / Internet – Zugang dienen sollen. Technisch gesehen sind die vier neuen Interfaces auch im Independent – Zustand nutzbar. Hierfür müssten nur noch die Layer 4 Parameter gesetzt werden. Diese Konfiguration wäre aber für spätere Firewall – Regeln sehr umständlich und auch bei grösseren Regelsätzen kaum mehr überschaubar. Ideal wäre eine Adresse mit einem Subnetz, das sich später, in Hinblick auf den Gesamttraffic, besser isolieren lässt. Diese Ideallösung wird im nachfolgenden Subpunkt geklärt.

### 8.4.2.3 Konfiguration des „Soft“- Switches

Um die vier Independent – Interfaces (LAGG0, LAGG1, single1_xnode2 und single2_xnode3) nicht mühsam in Einzelarbeit konfigurieren zu müssen, soll an dieser Stelle eine Bridge erstellt werden. Diese Bridge vereint die vier Interfaces zu einem logischen Verbund von Interfaces, wo das Übermitteln der Pakete zwischen zwei VM's auf zwei unterschiedlichen Nodes mittels ARP entschieden werden kann. Zusätzlich kann der Bridge hier das erste Mal eine IP – Adresse mit den dazugehörigen Parametern übergeben werden, welche den späteren VM's, aber auch allen externen Aufrufen, einen eindeutigen Zielpunkt liefert. So kann bspw. eine Verbindungsanfrage vom privaten Segment aus dem LAN, welche eine IP – Adresse einer VM als Ziel hat, die sich im Subnetz der Bridge befindet, auch geroutet werden. Von der Gegenseite wird eine VM immer die Bridge – IP als Default Gateway und somit als primären Ansprechpartner haben.

Diese Konfiguration bietet zwei elementare Vorteile. Erstens ist die Isolierung des VM – Networks gegenüber allem, was auf der Firewall an Netzwerken vorhanden ist, besonders einfach umsetzbar. Zweitens bleiben die vier Interfaces der Bridge weiterhin im Zustand „Standalone“, was spezifische Regelsätze ermöglicht für VM's, die bspw. statisch sind und stets auf dem gleichen Node arbeiten.

#### 8.4.2.3.1 Erstellen der Bridge

Das Erstellen der Bridge ähnelt dem Verfahren der Link Aggregation – Erstellung. Man klickt folgendermassen:

```
Interfaces → (assign) → Bridges
```

*Befehl 4: Realisierung des Netzwerksegmentes (Klicken): Erstellen der Bridge mit Angabe der dazugehörigen Interfaces.*

Hier wählt man wieder das Hinzufügen – Symbol und landet anschliessend im Konfigurationsfenster.

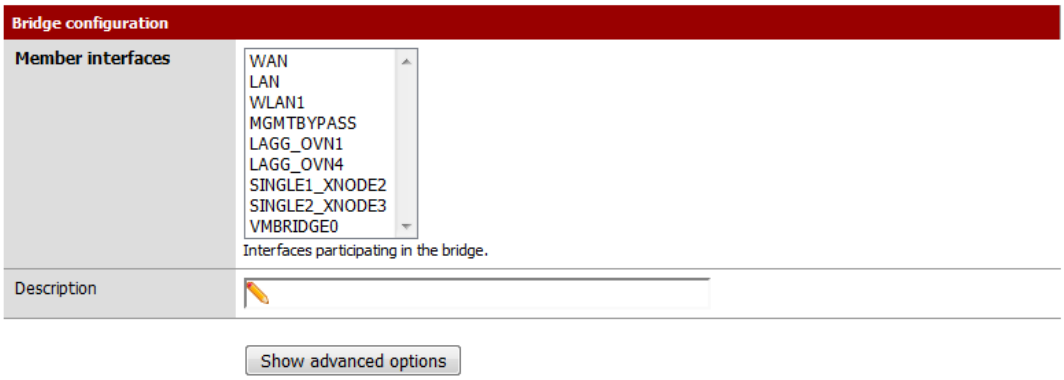



*Abbildung 12: Realisierung des Netzwerksegmentes: Einbinden der Bridge ins System (aktivieren)*

Hier kann man in der Auswahlliste die Interfaces auswählen, welche später in der Bridge vereint werden sollen. Auch hier empfiehlt sich wieder, einen aussagekräftigen Name unter **Description** einzugeben. Hinter dem Button **Show advanced options** verbirgt sich ein gutes Beispiel für die hohe Macht von pfSense. Hier können neben einigen allgemeinen Optionen, auch eine grosse Anzahl an Spanning – Tree Optionen gesetzt werden. Man sieht also, dass pfSense ohne weiteres als vollwertiger Switch verwendet werden kann.

Nach dem bestätigen mit **Save** ist die neue Bridge im System angelegt. Wie aber bei pfSesne üblich, genügt dies meist nicht. Die Bridge muss noch dem System als aktive Komponente bekanntgegeben werden.

#### 8.4.2.3.2 Einbinden der Bridge

Dies Prozess wird wieder über **Interface assignments** abgewickelt. Hierzu klickt man wieder in Richtung:

```
Interfaces → (assign) → Interface assignments
```

*Befehl 5: Realisierung des Netzwerksegmentes (Klicken): Einbinden der Bridge ins System.*

Hier findet man nun in der Auswahlliste die neu angelegte Bridge, welche entsprechend ausgewählt und mit **Hinzufüge** angemeldet wird. Auch hier geht man mittels Klick auf den Name in das Konfigurationsmenü des neuen Interfaces. Wie allgemein bekannt, kann an dieser Stelle ein Interface

Name vergeben und das Interface mit Häkchen auf **Enable Interface** in Betrieb genommen werden. An dieser Stelle wird das erste Mal eine IP – Adresse vergeben, hier 10.100.100.1/22. Warum gerade diese wird nachfolgend erklärt.

Dies war schon die gesamte Konfiguration. Von nun an hat man einen Switch, welcher von aussen vier Ports hat, wobei zwei Ports auf je drei Reale balanciert werden. Im nachfolgenden Screenshot ist ein Konsolen – Ausschnitt ersichtlich, welcher die gesamte Konfiguration der Bridge / des Switches zeigt. Die dort ersichtlichen Punkte wie **root ID**, **priority** oder **hellotime** sind bei FreeBSD Systemen per Default ersichtlich, haben aber keinen Einfluss, solange **Spanning Tree** nicht spezifisch aktiviert wurde.

```
[2.2.5-RELEASE][admin@bouncer1.localdom]/root: ifconfig bridge0
bridge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 02:f0:17:be:09:00
        inet 10.100.100.1 netmask 0xfffffc00 broadcast 10.100.103.255
        nd6 options=1<PERFORMNUD>
        id 00:00:00:00:00:00 priority 32768 hellotime 2 fwddelay 15
        maxage 20 holdcnt 6 proto rstp maxaddr 2000 timeout 1200
        root id 00:00:00:00:00:00 priority 32768 ifcost 0 port 0
        member: lagg0 flags=143<LEARNING,DISCOVER,AUTOEDGE,AUTOPTP>
                ifmaxaddr 0 port 23 priority 128 path cost 55
        member: igb6 flags=143<LEARNING,DISCOVER,AUTOEDGE,AUTOPTP>
                ifmaxaddr 0 port 9 priority 128 path cost 2000000
        member: igb7 flags=143<LEARNING,DISCOVER,AUTOEDGE,AUTOPTP>
                ifmaxaddr 0 port 10 priority 128 path cost 2000000
        member: lagg1 flags=143<LEARNING,DISCOVER,AUTOEDGE,AUTOPTP>
                ifmaxaddr 0 port 24 priority 128 path cost 55
```

*Abbildung 13: Realisierung des Netzwerksegmentes: Screenshot der fertigen Bridge*

### 8.4.3 Das Adress – Design

Das IP – Adressschema wurde nicht spezifisch in einer Analyse auf bestmögliche Ausnutzung ermittelt. Es folgt einer Standard Deklaration, welche der Autor im Allgemeinen verwendet. Hierbei werden immer interne Adressen aus der privaten A – Klasse verwendet. Dabei wird stets mit kleinen Nummern begonnen und bei logischen Segmentübergängen, welche von sich aus eine Isolierung benötigen, ein entsprechend grosser Sprung vorgenommen. Hier ein Beispiel:

- **LAN ←→ WLAN1** haben eine direkte logische Verbindung bezüglich Kommunikation: Dies bedeutet eine hohe Regelzahl zum Filtern, was sich in kleinen Nummerierungssprüngen widerspiegelt → LAN 10.0.10.0/24, WLAN1 10.0.30.0/28
- **LAN ←→ VMBRIDGE0** (bridge0) haben theoretisch keine logische Verbindung im Alltag (höchstens Wartung): In diesem Fall soll eine oder eine kleinere Anzahl an Regelsätzen den Verkehr massiv einschränken oder sogar absolut stoppen. Dies führt zu grossen Nummerierungssprüngen wie bspw. → LAN 10.0.10.0/24, bridge0 10.100.100.0/22
- Eine spezielle Ausnahme sind Konstellationen, welche zwar eine hohe Isolierung benötigen aber dennoch mehr Zugriffe erzeugen als eine simple Wartung. Ein solches Beispiel wäre die Verbindung von LAN zum Management – Teil des Clusters. Dies widerspiegelt sich auch im zweiten Block der IP, wie es im oberen Beispiel ersichtlich ist. Jedoch ist hier die Nummer näher am privaten Teil → LAN 10.0.10.0/24, MGMTBYPASS 10.10.10.1

Der dritte IP – Adressblock dient der weiteren Separierung des jeweiligen Segmentes. Hier wird meist nach Augenmass gearbeitet . Die Nummern werden so gewählt, dass im Falle eines DNS Ausfalles ein einfaches Erinnern möglich ist.

Hier die Adressschema Tabelle:

| Interface Name | Netz – ID | Subnetzmaske | Domainname | Max. Hosts | Bemerkung |
|---|---|---|---|---|---|
| Bouncer1.localdom | 10.0.10.1 - Main | Multi | Multi | Multi | Diese Firewall |
| LAN | 10.0.10.0 | 255.255.255.0 | localdom | 254 | Privater Teil: An dieser Stelle erfolgen aus Sicherheitsgründen keine weiteren Angaben. |
| WLAN1 | 10.0.30.0 | 255.255.255.248 | localdom | 14 | Privater Teil: An dieser Stelle erfolgen aus Sicherheitsgründen keine weiteren Angaben. |
| MGMTBYPASS | 10.10.10.0 | 255.255.255.0 | mgmtdom | 254 | Dieser Teil verbindet alle Komponenten des Clusters, das heisst, dass oVirt Management Netzwerk und die Storage Komponenten. |
| VMRIDGE0 | 10.100.100.0 | 255.255.252.0 | vmdom | 1022 | In diesem breiten Range, werden später die IP's der VM's vergeben. Eine zusätzliche Segmentierung mittels VLAN ist theoretisch möglich, es wird aber darauf verzichtet. |

*Tabelle 18: Realisierung des Netzwerksegmentes: IP - Adressschema*

### 8.4.4 Erster Blick auf die Sicherheit

In dieser Sektion der Realisierung können die ersten Regeln bezüglich der Isolierung des Management teils und des Virtualisierungs- Teils in Bezug auf den privaten Teil des Gesamtnetzwerkes gemacht werden. Hierfür sollen drei Regelsätze definiert werden, welche wie in nachfolgender Tabelle ersichtlich, den Datenverkehr stark begrenzen oder gänzlich unterbinden.

**Anmerkung:** Der Einfachheit halber werden die Interfaces LAN und WLAN1 innerhalb der Tabelle in das real nicht existierende Pseudointerface PRIVATE zusammengefasst.

| Nr.: | Interface Beziehung | Protokoll | Source | Sour. Port | Destination | Dest. Port | Gesetzt auf Interface | AKTION |
|---|---|---|---|---|---|---|---|---|
| 1 | VMBRIDGE0 → PRIVATE | IPv4 | ANY | ANY | ANY | ANY | VMBRIDGE0 | BLOCK |
| 2 | VMBRIDGE0 →MGMTBYPASS | IPv4 | ANY | ANY | ANY | ANY | VMBRIDGE0 | BLOCK |
| 3 | MGMTBYPASS→PRIVATE | IPv4 | ANY | ANY | ANY | ANY | MGMTBYPASS | BLOCK |

*Tabelle 19: Realisierung des Netzwerksegmentes: Regelsatzschema zu den notwendigen Regeln*

Diese drei Regeln sollen angewendet werden, um eine Isolation zwischen dem VM – Netzwerk und Management – Netzwerk gegenüber dem PRIVATE Netzwerk zu ermöglichen. Zusätzlich soll auch eine klare Isolation zwischen dem VM – Netzwerk und dem Management – Netzwerk erzeugt werden.

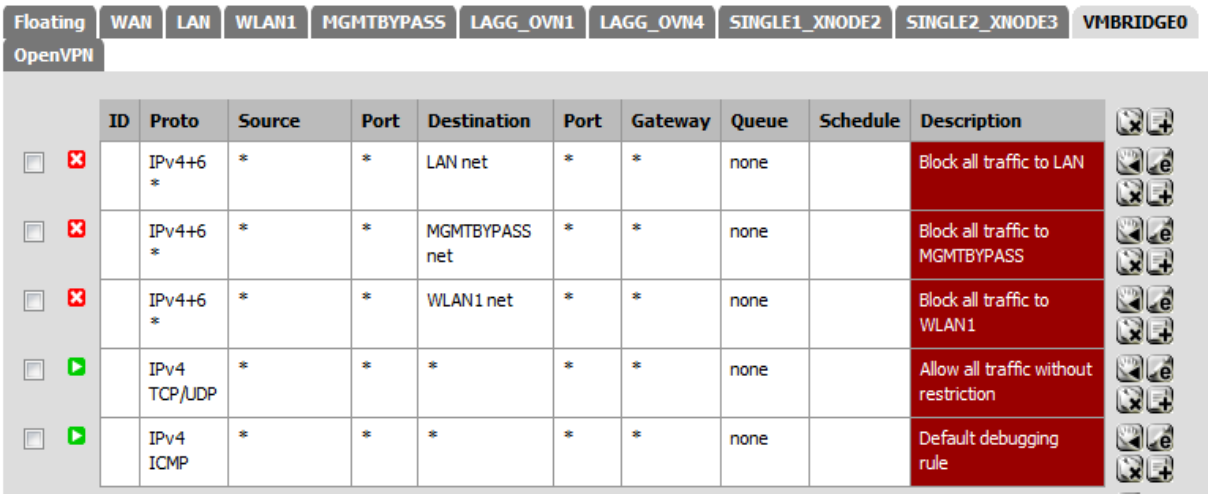
Nun ist die Regelkonfiguration aufgrund der bei pfSense per Default aktiven **Stateful Packet Inspection** nicht ganz einfach nachzuvollziehen. Aus diesem Grund analysieren wir die drei Regeln anhand der Nummern innerhalb der Tabelle 19.

Die nachfolgenden Konfigurationen werden im Firewall – Konfigurationsmenü vorgenommen. Hierzu klicken wir wie folgt:

```
Firewall → Rules → Wahlweise   - VMBRIDGE0
                               - MGMTBYPASS
```

*Befehl 6: Realisierung des Netzwerksegmentes (Klicken): Der Weg zum Menü der Firewall - Regelkonfiguration.*

**Regeln 1 nach Tabelle 19:** Kein Zugriff von VMBRIDGE0 zum privaten Teil des Netzwerkes. Umgekehrt soll es aber weiterhin möglich sein:




| ID | Proto | Source | Port | Destination | Port | Gateway | Queue | Schedule | Description |
|---|---|---|---|---|---|---|---|---|---|
| | IPv4+6 * | * | * | LAN net | * | * | none | | Block all traffic to LAN |
| | IPv4+6 * | * | * | MGMTBYPASS net | * | * | none | | Block all traffic to MGMTBYPASS |
| | IPv4+6 * | * | * | WLAN1 net | * | * | none | | Block all traffic to WLAN1 |
| | IPv4 TCP/UDP | * | * | * | * | * | none | | Allow all traffic without restriction |
| | IPv4 ICMP | * | * | * | * | * | none | | Default debugging rule |

*Abbildung 14: Realisierung des Netzwerksegmentes: Konfiguration von VMBRIDGE0*

Die Firewall – Tabellen sind beim Erstellen von Interfaces bzw. beim Einbinden stets leer. Um die Bedingungen von Regel 1 nach Tabelle 19 zu erfüllen, sind die oben ersichtlichen Regelsätze notwendig. Hierfür muss man wissen, dass pfSense eine pro Interface stets nicht ersichtliche Regel besitzt, welche den outgoing Verkehr komplett sperrt (**deny any to any**). Prinzipiell kann man die Liste leer lassen und somit die innerhalb dieses Interfaces eingeschlossenen Hosts komplett isolieren. Dennoch ist es weiterhin möglich, sich von aussen auf die sich darin befindlichen Host zu connecten. Nun stellt sich die Frage, wieso der outgoing Verkehr zurück zum externen Abfragepartner gelangen kann. Dies funktioniert deswegen, weil die Stateful Inspection den outgoing Verkehr mit dem Flag **Established** markiert. So weiss die Firewall, dass der Verkehr ausnahmsweise das Interface verlassen kann, da hier vorgängig eine zulässige Verbindung von aussen aufgebaut wurde. Im Grunde muss man bei pfSense stets auf dem Interface, von welchem aus man eine Verbindung Richtung outgoing herstellen möchte, Regeln setzen, welche sagen, ob dies geht oder nicht. Setzt man eine **Allow any to any**, so wird die letzte unsichtbare Allgemeinsperre bei dieser First – Match Firewall ausgehebelt.

Daher ist es ratsam, diese Allow – Direktive immer zuletzt, oder an der wirklich geeigneten Stelle, zu setzen. Ansonsten kann es sein, dass einige Regeln umgangen werden.

Beginnen wir mit der Analyse der Regeln in Abbildung 14 (Top – Down):

- **Die erste Regel** sagt der Firewall, dass jeglicher Verkehr über alle Protokolle und alle Ports nicht in Richtung LAN gehen darf.
- **Die zwei Regel** entspricht der vorgängigen, jedoch auf das Management – Netzwerk bezogen.
- **Die drei Regel** schliesst sich dem Konzept der beiden vorgängigen an.
- **Die vier Regel** erlaubt sämtlichen Verkehr nach aussen über das Interface VMBRIDGE0. Da aber nach dem First – Match Verfahren die oberen Regeln zum Zuge gekommen wären, kann an dieser Stelle nur das „Any" Interface genutzt werden. Dies bedeutet in der pfSense Welt, der Verkehr darf das WAN bzw. in diesem Fall, das NAT nutzen, um ins Internet zu gelangen.
- **Die fünf Regel** gibt die Erlaubnis für ICMP ins Internet. Dies ist ein Standard – Vorgehen des Autors, um Debuggen leichter zu gestalten. Da der ICMP Verkehr aber weder in den isolierten Bereichen noch vom Internet her ohne Einwilligung des NAT fliessen kann, ist diese Regel sicherheitstechnisch unbedenklich.

**Regeln 2 nach Tabelle 19:** Kein Zugriff von VMBRIDGE0 zum MGMTBYPASS.

Um absolute Sicherheit zu gewährleisten, hat Kommunikation zwischen dem Virtualisierungsteil und dem Management – Teil nicht statt zu finden, sie ist vollkommen unnötig.

Hier kann auf Regel 1 referenziert werden, da sie die Kommunikation in Richtung MGMTBYPASS ebenfalls unter den gleichen Bedingungen sperrt wie für das LAN.

**Regeln 3 nach Tabelle 19:** Kein Zugriff von MGMTBYPASS zum privaten Teil des Netzwerkes. Umgekehrt soll es aber weiterhin möglich sein.

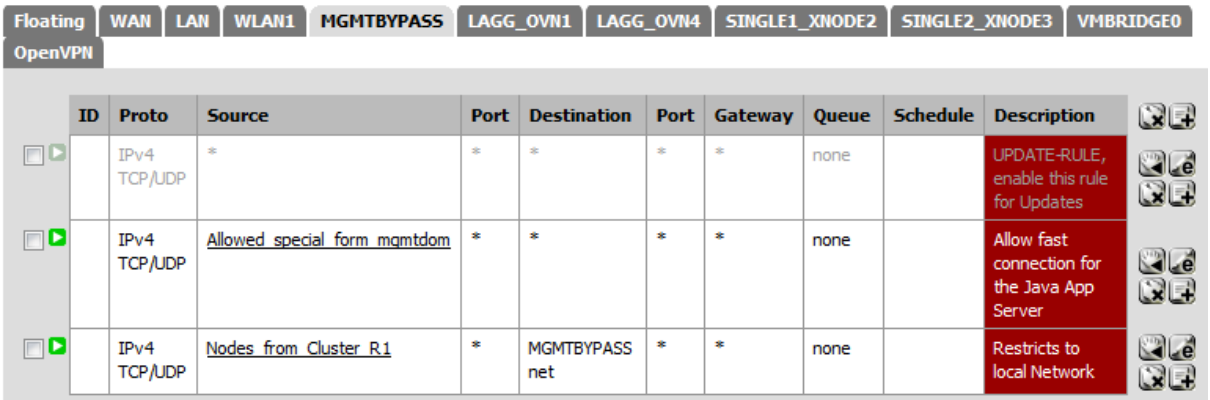



*Abbildung 15: Realisierung des Netzwerksegmentes: Konfiguration von MGMTBYPASS*

Die Situation hier ist etwas heikler als beim VMBRIDE0 Interface. Die Maschinen innerhalb des Clusters sollen von sich aus keine Verbindung zum privaten Teil haben. Sie sollten auch nicht unbedingt frei mit dem Internet kommunizieren können. Wenn überhaupt, soll nur der Admin

kontrolliert mit den einzelnen Maschinen zwecks Updaten ins Internet können. Die Maschinen sollen aber eine gewisse Bequemlichkeit bezüglich Namensauflösung und anderer Dienste aufweisen, wie etwa das Abgleichen der Zeit mit dem zentralen NTP – Server auf der Firewall. Es soll also maximale Sicherheit trotz einer Vielzahl an Ausnahmen realisiert werden. Die in Abbildung 15 erstellten Regeln werden den genannten Forderungen gerecht.

Sehen wir uns nun die genaue Analyse der Regelsätze anhand der Regeln in Abbildung 15 an (Top – Down):

**Die erste Regel** dient dem Updaten der Maschinen und ermöglicht erst bei Aktivierung den Zugang ins Internet. Da pfSense nach dem First – Match Verfahren arbeitet, hebelt diese Regel alle andern aus, was einen massiven Sicherheitsverstoss darstellt. Aus diesem Grund soll die Regel auch nur zu Updatezwecken genutzt werden.

**Die zweite Regel** ist in dieser Form ganz speziell und wäre im Normalfall nicht notwendig. Für sie wurde unter:

**Firewall → Aliases → IP**

*Befehl 7: Realisierung des Netzwerksegmentes (Klicken): Erstellung eines IP - Adressalias*

ein Alias Verweis erstellt, welcher den Namen **ALLowed_special_from_mgmtdom** trägt. Hier ist die IP – Adresse der oVirt – Engine hinterlegt, welche so unter einer einheitlichen Namenskategorie bequem nutzbar ist und bei Bedarf mit noch weiteren Hosts aufgefüllt werden kann. Dieser Alias erlaubt der oVirt – Engine den vollen Internetzugang. Dies wurde notwendig da die Engine bzw. der Java Applicationserver Schwierigkeiten bei der Isolation des Interfaces zeigte. Hier wird vermutet, dass der Applicationserver mehrere Verbindungen in Richtung Client aufbaut und die State Table von pfSense deswegen nicht schnell genug mit den Statusmeldungen umgehen konnte. Dies führte zu Aufbauzeiten des WebGUI's von ca. 40 – 60 Sekunden, was als untragbar eingestuft wurde. Deswegen wird diese Any to Any Regel, welche als effizienteste getestet und bestätigt wurde, so hingenommen (trotz der theoretischen Sicherheitslücke, die daraus resultiert.)

**Die dritte Regel** entspricht dem, was auch bezüglich Sicherheit angestrebt werden sollte. Auch für diese Regel wurde ein Alias angelegt namens **Nodes_form_Datacenter_R1**, welcher alle Nodes des Clusters abzüglich der Engine enthält. Diese Regel besagt, dass NUR diese Maschinen direkten Zugriff auf das Interface MGMTBYPASS haben und somit die Dienste, welche auf der IP des Interfaces horchen, nutzen kann. Ein Host, der irgendwie in das Interface eindringen könnte, ist komplett isoliert. Hier muss noch gesagt werden, dass bei pfSense ein Zugriff auf das Interface und somit auf den Default Gateway nicht bedeutet, dass die Maschine auch ins Internet darf. Denn ein allfälliger Aufruf auf eine Public IP aus diesem Interface, bedeutet laut Regel nur, dass der Verkehr bis zum Interface darf, nicht aber darüber hinaus. Nur eine Any to Any Regel besagt, dass der Verkehr auch über das Interface geroutet bzw. geNATet werden darf.

### 8.4.4.1 Die Members der Bridge

Die Interfaces **LAGG_OVN1, LAGG_OVN4, SINGLE1_XNODE2 und SINGLE2_XNODE3** sind zwar Menbers der BRIDGE0, können und müssen teilweise sogar eigenständig konfiguriert werden. Für eine Reibungslose Kommunikation der VM's untereinander, ist es notwendig den Verkehr entsprechend zu erlauben. Hier muss aber nicht jedes Interface bzw. Bündelung einzeln auf die bestimmten Ziele konfiguriert werden. Primär muss jede Datenverbindung nur in Richtung der Bridge zeigen, die Bridge beinhaltet die Regeln welche Gültigkeit besitzen. Im Klartext bedeutet dies, dass wenn eine VM aus einem Fujitsu Node mit einer anderen kommunizieren will, welche bspw. auf dem Dell arbeitet, so muss der Fujitsu Node immer in Richtung Bridge adressieren. Die Bridge entscheidend anhand ihrer Regelsätze ob dies erlaubt ist und biegt den Datenverkehr über einen internen Link zum eigentlichen Interface hin. Somit müssen folgende Regelen auf allen oben genannten Independent – Interfaces angewendet werde, um die Switching – Funktionen eines Switches zu ermöglichen.

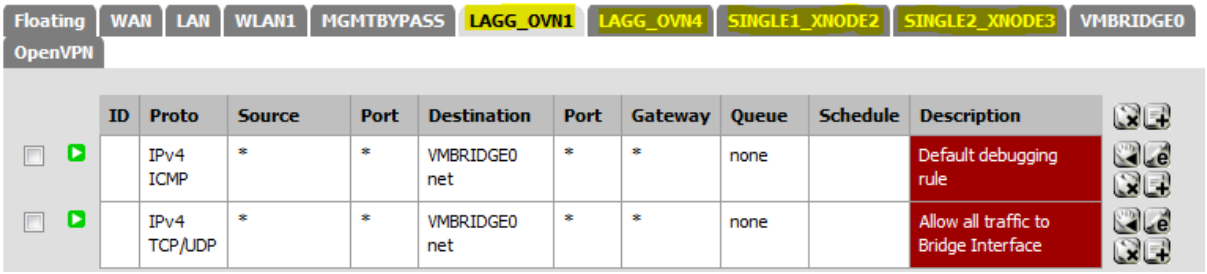



*Abbildung 16: Realisierung des Netzwerksegmentes: Notwendige Regeln um switching zu ermöglichen (independent Interfaces)*

Die obere Regel ist die typische Debugging – Regel, welche das Pingen im Netzwerk ermöglichen soll. Die zweite Regel ist die entscheidende, denn sie erlaubt es den VM's sich über die physischen grenzen der einzelnen Maschinen hinweg zu bewegen. Eine VM die bspw. in den privaten Bereich routen will, wird an dieser Stelle aufgehalten, da sie lediglich zum pseudo- Interface BRIDGE0 verbinden darf. Ob die Bridge den Verkehr in den privaten Sektor erlaubt, muss unter dem Interface VMBRIDE0 definiert werden. Auf diese Weise können sämtliche Isolationsregeln bequem an einem Ort definiert werden, was die Sicherheit bezüglich Sichtbarkeit der Regelsätze massiv vereinfacht.

### 8.4.5 Ein kleiner Abgleich zur Realität

#### 8.4.5.1 Die Firewall/ der gemanagte Switch

Um ein klareres Verständnis für die hardwareseitige Konfiguration zu bekommen, soll die nachfolgende Abbildung einen optischen Einblick gewähren:

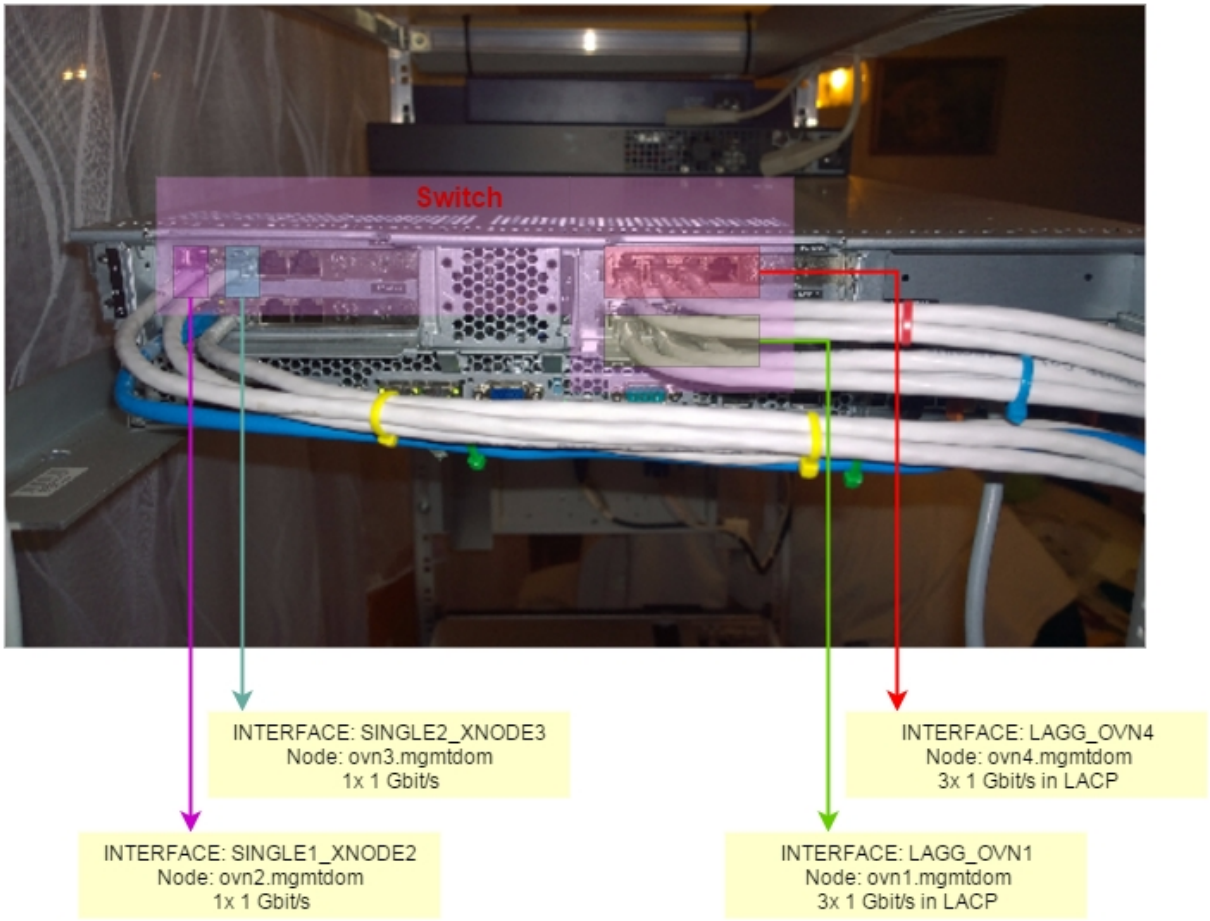



*Abbildung 17: Realisierung des Netzwerksegmentes: Optischer Vergleich zwischen Theorie und Praxis (Firewall/ gemanagter Switch)*

Wie in der Abbildung zu sehen ist, befinden sich die Link Aggregationen auf der rechten Seite und sind jeweils auf ein NIC aufgeteilt. Auf der linken Seite sind die beiden einzelnen Fujitsu Systeme mit je einem Port auf dem gleichen NIC angeschlossen. Das gesamte Gebilde stellt den durch pfSense gemanagten Switch dar. Im Notfall oder bei späteren Erweiterungen stehen noch 6 Ports zur Verfügung, um daraus noch weitere Link Aggregationen zu bilden oder einzeln anzuschliessen.

### 8.4.5.2 Der konventionelle ungemanagte Switch

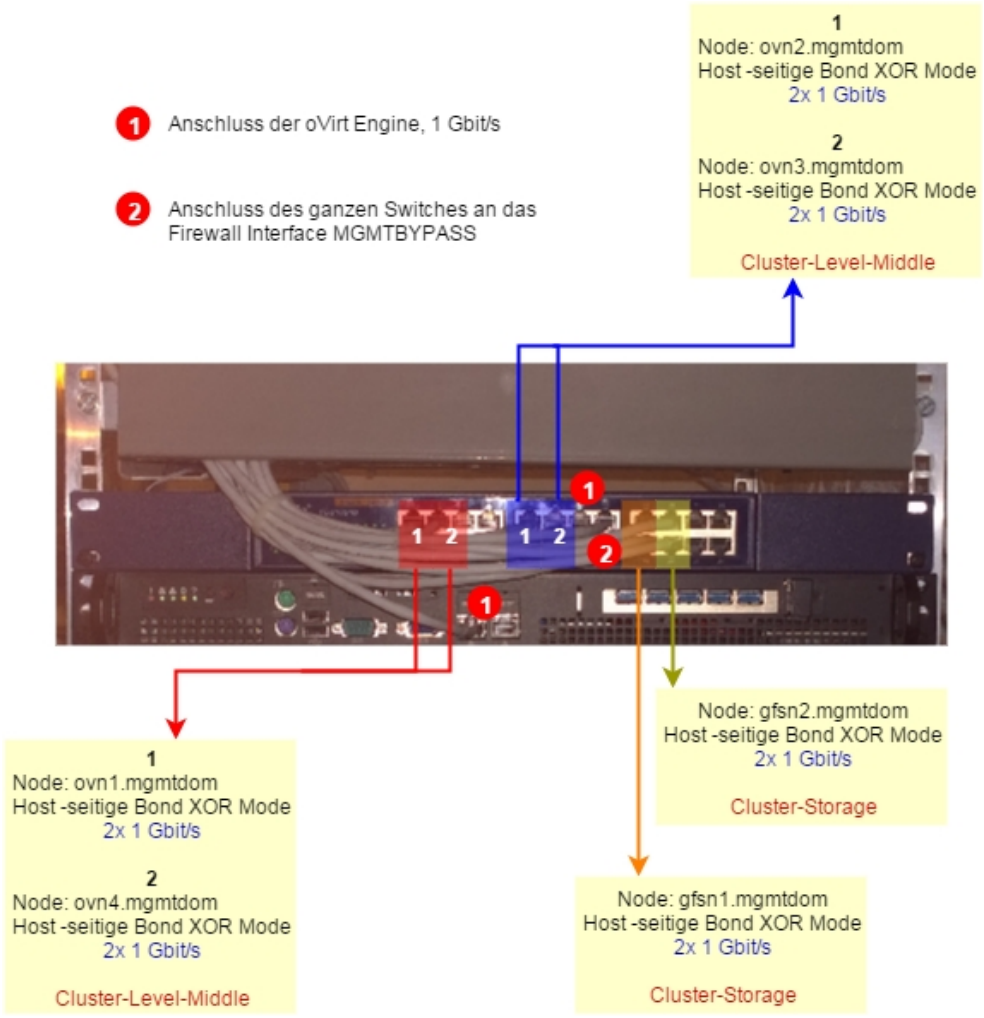



*Abbildung 18: Realisierung des Netzwerksegmentes: Optischer Vergleich zwischen Theorie und Praxis (Switch)*

Wie in der Abbildung 17 zu erkennen ist, sind alle Komponente, die Teil der Clusterinternen Kommunikation sind, am ungemanagten Switch angeschlossen. Das Bonding Verfahren, welches hier gewählt wurde, wird im entsprechenden Abschnitt genauer thematisiert. Prinzipiell ist es hier egal, wo was angeschlossen ist, jedoch wurde versucht, eine gewisse Ordnung zu halten und in Kategoriegruppen zu gliedern. Die Ausfallsicherheit der Engine und der Firewall – Anbindung wurden an dieser Stelle als nicht notwendig eingestuft.

Dies war der elementarste Teil der Konfiguration des Netzwerksegmentes. Weitere Feinkonfigurationen können in den weiteren Abschnitten folgen. Nach Beendigung dieses vorbereitenden Teiles kann mit der weiteren Realisierung fortgefahren werden.

## 8.5 Beginn mit der Realisation des Storagesegmentes

In diesem Abschnitt sollen alle notwendigen Schritte im Top – Down Verfahren aufgezeigt werden, welche notwendig sind, um das Storagesegment in einer Einzelbetrachtung aufzusetzen und zu konfigurieren. Dieser Teil umfasst folgende Konfigurationsschritte:

- Installation der CentOS 7 Minimalinstallationen
- Bilden des RAID5 mit entsprechendem Partitionieren mittels XFS Filesystem
- Installation der notwendigen Komponenten, hier speziell GlusterFS und VDSM
- Bilden der GlusterFS Mountpoints mit minimalen Konfigurationen

### 8.5.1 Die verwendete Hardware

Zur Realisierung des zentralen Storages sollen zwei preisgünstige HP MicroServer Gen8 der ProLiant Serie verwendet werden. Diese Maschinen eignen sich hervorragend für diesen Zweck. Sie haben mit einem Single – CPU System und einem Intel Celeron Prozessor mit zwei physischen Kernen gerade genug Leistung für einen Cluster dieser Grösse. Sie eignen sich auch besonders gut für Storage – Lösungen, da sie vier Festplatten – Slots besitzen, bei denen die Schubladen mit inbegriffen sind. Die genauen Spezifikation sind:

- CPU → Intel Celeron mit 2,3 GHz
- RAM → 2 GB ECC
- HD → 4 Slot – Plätze
- Netzwerk → Zwei Onboard 1 Gbit/s Ports

Um möglichst viel Platz aus den Disks zu holen, wird das Betriebssystem auf eine USB – Disk mit 1 TB Kapazität installiert und in das Gehäuse verbaut bzw. gequetscht. So können die vier HD – Slots ausschliesslich für den Storage verwendet werden.

### 8.5.2 Kurze Definition bezüglich des gewählten GlusterFS Designs

Zur Umsetzung wurde das Standard Verfahren von Gluster gewählt, welches sich Distributed nennt. Hier wird nach einem im Code von Guster integrierten Algorithmus versucht herauszufinden, welche Maschine gerade nichts zu tun hat und somit Daten zum Schreiben empfangen kann. Dabei wird immer ein ganzer, zusammenhängender Datensatz (also File, Verzeichnis oder in diesem Fall ein Image – File), an einen der beiden Nodes zum Speichern übergeben. Ein Design mit Replicated Funktionen, welche die Daten redundant sichern, wäre sicherlich schön gewesen, musste jedoch aus Kosten- und Platzgründen aufgegeben werden. Das gewählte Design ist jedoch auch nicht schlecht, denn beim Ausfall eines ganzen Node kann theoretisch jeder beliebige Rechner mit vier HD – Plätzen verwendet werden.

### 8.5.3 Installation des CentOS 7

Um einen GlusterFS Node in oVirt integrieren zu können, ist im Moment noch ein RedHat, bzw. ein Abkömmling davon, notwendig. Warum dies so ist, wird im Abschnitt „Realisation des Virtualisierungssegmentes“ genau thematisiert. An dieser Stelle wird nur die dafür notwendige Software installiert.

Die Installation von CentOS 7 ist aufgrund der Übernahme des neuen Anaconda – Installers von Fedoara mehr als einfach. Der Installer ermittelt praktisch alle Systemeinstellungen selbstständig und man wird zum Hauptmenüpunkt geführt. Hier muss man nun folgende Punkte angeben:

- **Partitionierung:** Hier kann man ohne weiteres die Standard – Konfiguration nutzen. Seitens des Autors wird aber empfohlen, die Partition **/boot** von 500 MB auf 1000 MB zu erhöhen, da sich hier mit der Zeit die alten Kernel nach dem Updaten ansammeln können.
- **Installationsart:** Ideal ist eine reine Minimalinstallation. Wer aber gerne debugging Tools wie bspw. nslookup nutzen möchte, sich aber die nachträgliche Installation der einzelnen Tools ersparen will, der sollte hier Standard- bzw. Serverinstallation (Fedora) wählen.
- **Netzwerkkonfiguration:** Diese kann auch nachträglich erledigt werden, jedoch ist es an dieser Stelle von Vorteil, um nach der Installation gleich mit SSH fortzufahren.
    - Das zu nutzende Adress- und Namensschema, welches für die beiden GlusterFS Nodes definiert wurde, ist in nachfolgender Tabelle ersichtlich:

| Node Name | IP- Adresse | Subnetzmaske | Hostname | Domainname |
|---|---|---|---|---|
| GlusterFS 1 | 10.10.10.21 | 255.255.255.0 | gfsn1 | mgmtdom |
| GlusterFS 2 | 10.10.10.22 | 255.255.255.0 | gfsn2 | mgmtdom |

*Tabelle 20: Realisierung des Storagesegmentes: Das Adress- und Namensschema der Gluster Nodes*

Für diese Installation wurde die fertige Installations- DVD genutzt. Ein genauer Einblick in die Installation ist aufgrund der Einfachheit nicht notwendig. An dieser Stelle sei auf die hervorgegangen Installationsanleitung im Internet verwiesen, wobei folgender Link besonders zu empfehlen ist:

http://www.tecmint.com/centos-7-installation/

#### 8.5.3.1 Installation der notwendigen Softwarepakete

Für die Nutzung der beiden Nodes als GlusterFS Komponente innerhalb eines oVirt Clusters sind folgende Softwarepakete notwendig:

- **mdadm:** um Software RAID's bilden zu können.
- **GlusterFS: glusterfs** (Client); **glusterfs-fuse**; Gluster arbeitet mit dem Filesystem in Userspace Framework, um auf diese Weise auch Usern das Einbinden zu ermöglichen und um eine entwicklertechnisch unabhängige Basis zu schaffen; und der eigentliche **glusterfs-server**, welcher notwendig ist, um Gluster überhaupt betreiben zu können.
- **VDSM:** Diese Software ist für die Integration in oVirt notwendig.

Mit dem nachfolgenden Befehl, können sämtliche Pakete bequem installiert werden.

```
# yum install mdadm glusterfs glusterfs-fuse glusterfs-server

# yum install centos-release-ovirt35.noarch
# yum innstall vdsm-*

### Alle Befehle mit „y“ → Enter bestätigen!
```

*Befehl 8: Realisierung des Storagesegmentes (Befehl): Installation der notwendigen Softwarepakete*

Mdadm und GlusterFS können ohne weiteres installiert werden, da sie Bestandteil der regulären Paketquellen sind. VDSM welches im nächsten Segment beschrieben wird, ist nicht Bestandteil der regulären Quellen. Deswegen muss an dieser Stelle mit **yum install centos-release-ovirt35.noarch** zuerst die oVirt Paketquelle installiert werden, danach kann vdsm installiert werden. Da bei einigen Versuchen festgestellt wurde, dass die Abhängigkeiten nicht korrekt berechnet werden, müssen alle Pakete in einem Zug installiert werden, weswegen der allgemeine Ausdruck vdsm-* (* → Alles was ähnlich ist) verwendet wird.

### 8.5.3.2 Einrichten des RAID 5

#### 8.5.3.2.1 Erzeugen des Software RAID

Wie in der Hauptstudie definiert, wird an dieser Stelle ein RAID 5 Verbund eingerichtet. Hierfür werden die Festplatten in ihre Trägervorrichtungen montiert und in die Server eingeführt.

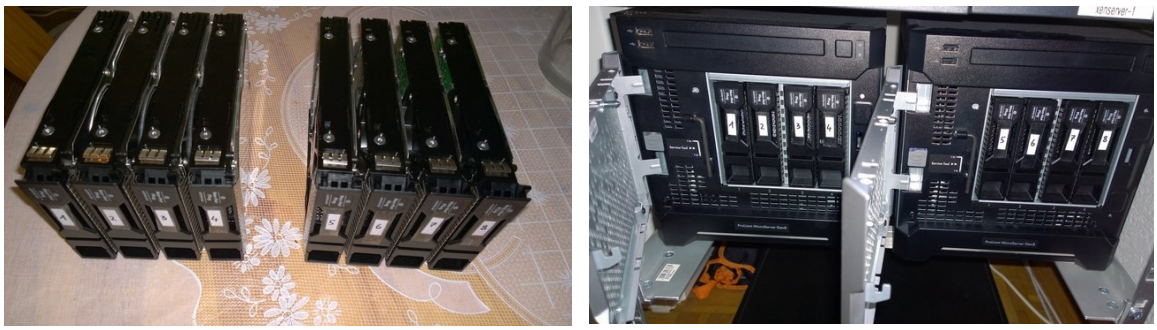


*Abbildung 19: Realisierung des Storagesegmentes: linke Abb.: Fesplattenmontage, rechte Abb.: Einbau*

Mittels eines Partitionierungs- Tools, müssen zuerst die Festplatten mit einer Partitionstabelle überzogen werden. Hierfür wird **fdisk** verwendet, mit dessen Hilfe jede Festplatte mit der Linux spezifischen Partitionstabelle **Linux RAID** (0xFD in der Typen Tabelle) formatiert wird.

**ACHTUNG!**

**Aufgrund des Sortieralgorithmus von uDev, ist die USB – Disk nicht an erster Stelle. Sie findet sich meist tiefer in der Hierarchie. In Diesem Fall liegt sie unter /dev/sdd**

**TIPP**

**Wer sich mit den Command line Tools fdisk und cfdisk nicht auskennt, kann Live – Systeme wie „system rescue cd“ verwenden. Diese bringen das grafische Tool gparted mit. So lassen sich Partitionierungen leicht unter einem GUI vornehmen.**

Nach dem Partitionieren und Markieren als RAID – Disk, kann mit dem Erstellen des Software – RAID begonnen werden.

Das Werkzeug für diesen Zweck ist **mdadm**, mit dessen Hilfe der Software – RAID initialisiert wird. Der für diese Arbeit verwendete Befehlsaufruf lautet wie folgt:

```
# mdadm --create --metadata 1.2 --verbose /dev/md0 --chunk=32 --level=5 --raid-devices=4 /dev/sda1 /dev/sdb1 /dev/sdc1 /dev/sde1
```

*Befehl 9: Realisierung des Storagesegmentes (Befehl): Erstellen des Software - RAID 5*

Doch was bedeuten all diese Werte in dieser langen Befehlskette? Nachfolgend eine genaue Erläuterung:

- **mdadm**, **create** und **verbose** dienen dem Aufruf, der Instruktion und einer Ausgabe für Debugging – Zwecke.
- **Metadata;** Linux Software – RAID speichert alle relevanten Daten zu ein RAID Array an gewissen Positionen im Speicherblock, dem sogenannten Superblock. Die Version, in diesem Fall 1.2, besagt laut Normierung, dass die Metadaten und damit der Superblock 4 KiB nach Beginn des Devices liegen muss.
- **/dev/md0;** ist der Hardware bezogene Name des RAID Device. Dieser kann frei zwischen den Nummern 0 – 127 gewählt werden.
- **Level=5;** ist der RAID Level, welcher erzeugt werden soll.
- **Chunk=32;** ist die Chunk Size, welche genutzt werden soll, um die Blockgrösse einer Speichereinheit zu definieren.
- **Raid-devices;** sagt mdadm, wie viele Disks effektiv genutzt werden sollen. Dieser Wert wird vordefiniert, um allfällige Fehleingaben beim nächsten Schritt zu kompensieren.
- **/dev/sda1-c1+e1;** sind die Disks bzw. die Partitionen, welche dem RAID Verbund beim kreieren zur Verfügung stehen.

**ALTERNATIVE**

**Manche behaupten heute, dass es effizienter sei, die Disks direkt dem Software – RAID zu übergeben und nicht erst eine Partitionstabelle zu generieren. Dies ist mit mdadm heute möglich, ob es auch effizienter ist, muss jeder selbst ermitteln und entscheiden. Der Autor folgt hier der alten Tradition.**

Dies war es mit dem ersten Teil der RAID – Bildung. Dieser Prozess des RAID 5 Kreierens dauert bei

einer Konfiguration wie dieser ca. 16 – 19 Stunden, je nach Chunk Size. In diesem Fall hat er 19 Stunden gedauert. Es lohnt sich, einen solchen Prozess am Abend zu initialisieren, da ab hier nicht viel getan werden kann.

Früher war es noch notwendig, das konfigurierte Array noch in das File /dev/mdadm.conf einzutragen. Hier half einem der **mdadm** Befehl mit den Optionsargumenten **–detail –scan**, welche das / die Arrays ermittelten. Diese Ausgabe konnte mittels „**>**“ in die Datei mdadm.conf umgeleitet werden. Mit den heute modernen Initialsystemen wie bspw. Systemd ist dies nicht mehr notwendig. Sie erkennen, falls mdadm installiert ist beim Systemstart, ob es RAID – Verbünde hat und speichern sie dauerhaft mit einer UUID in das Verzeichnis /dev/md/<Nummer>. Man kann mdadm.conf noch nutzen, da die dort eingetragenen Angaben immer noch eine Gewichtung haben.

```
md0 : active raid5 dm-7[1] dm-4[0] dm-5[2] dm-6[4]
      5860147392 blocks super 1.2 level 5, 32k chunk, algorithm 2 [4/4] [UUUU]
      bitmap: 0/15 pages [0KB], 65536KB chunk
```

*Abbildung 20: Realisierung des Storagesegmentes: Auszug aus /proc/mdstat von Gluster Node 1*

#### 8.5.3.2.2 Erzeugen des Filesystems

Nach Beendigung des hardwarebezogenen Teiles kann das Filesystem erzeugt werden. Hier wurde XFS gewählt. Es bietet einige interessante Features wie bspw. das Erzeugen von Snapshots auf der Partitionensebene. Der elementarste Vorteil von XFS ist die Möglichkeit, Optimierungen beim Einrichten von Partiotionen auf RAID – Verbünden vorzunehmen. So kann XFS klar mitgeteilt werden wie gross die Chunk Size des Filesystems sein soll und auf wie vielen Disks sich das Filesystem in der Realität befindet. So kann die Partition optimal auf die Chunk Size des RAID 5 synchronisiert werden. Hierzu gibt es eine Vielzahl an Erklärungen im Internet, welche aber in hohe mathematische Sphären aufsteigen. Eine verständliche und schnell anwendbare Erklärung findet sich unter folgendem Link:

http://linuxsnippets.net/en/snippet/xfs-how-calculate-correct-sunitswidth-values-optimal-performance

In dieser Erklärung sieht man, dass lediglich zwei Optionen bei XFS notwendig sind, um die entsprechenden Anpassungen für diesen RAID Verbund vorzunehmen.

- **su:** Entspricht der Stripe Size bzw. der Chunk Size
- **sw:** Stripe Width, was bei dieser vereinfachten Anwendungsweise der Anzahl an aktiv genutzten Data Disks entspricht.

Zwar richtet sich das Beispiel an RAID 6 – Verbünde, kann jedoch leicht an diesen Fall angepasst werden. In diesem Fall einspricht der su – Wert = 32KiB und der sw – Wert = 4 Disks. Hier können vier Disks angegeben werden , da die Paritätsdaten auf alle vier Disks aufgeteilt werden und somit auch vier Data Disks zu Verfügung stehen. Nachfolgend der notwendige Befehl zum Erstellen der Partition.

```
# mkfs.xfs -d su=32k,sw=4  /dev/md0
```

*Befehl 10: Realisierung des Storagesegmentes (Befehl): Erstellen des XFS Filesystems*

Nun müssen noch je zwei Einbindepunkte für beide Nodes definiert werden. Hierzu wird auf beiden

Nodes ein Verzeichnis namens **/storage** angelegt. Hier wird die XFS Partition, welche über den RAID – Verbund gezogen wurde, gemountet. Um diese Aktion auch rebootfest zu machen, wird ein permanenter Eintrag in **/etc/fstab** notwendig. Hier wählt der Autor normalerweise den UUID der Partition als Identifikationspunkt, jedoch zeigt CentOS 7 in dieser Hinsicht kleine Schwierigkeiten, was als Backup – Lösung die klassische Schreibweise zur Folge hatte. Nachfolgend ein Beispiel anhand des Node 1:

```
#
# /etc/fstab
# Created by anaconda on Tue Dec  8 16:04:27 2015
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=3945c3b1-72d5-4f2a-b943-162ec64a483d /                       xfs     defaults        0 0
UUID=be45d179-db52-4e07-9c47-8f2cf652dd8c /boot                   xfs     defaults        0 0
UUID=248a5f4c-c1ab-4f62-b088-ba85268b8618 /home                   xfs     defaults        0 0
UUID=a5e21a49-26ff-4d1f-9ca4-413035a590fa swap                    swap    defaults        0 0
/dev/md0                                  /storage                xfs     defaults        0 0
```

*Abbildung 21: Realisierung des Storagesegmentes: Auszug aus /etc/fstab von Node 1 (gfsn1.mgmtdom)*

Wie in Abbildung 20 zu sehen ist, wurde das RAID – Device permanent in **/storage** gemountet. Wie bereits erwähnt ist nicht bekannt, warum der UUID bei diesem Eintrag nicht funktionierte. Dieser Prozess wird auf Node 2 identisch wiederholt.

#### 8.5.3.2.3 Bilden der GlusterFS Volumes

Der nächste Schritt ist die Erstellung der beiden Basisordner in /storage. Hierzu wird Brick1/2 auf beiden Nodes für den Main – Storage von GlusterFS auf Node1/2 erstellt sowie ISO_Store1/2 für die späteren NFS Freigaben, in welchen sich die ISO's befinden werden. Siehe hierzu folgendes Beispiel von Node1:

```
# mkdir /storage/brick1
# mkdir /storage/ISO_STORE1
```

*Befehl 11: Realisierung des Storagesegmentes (Befehl): Anlegen der beiden Gluster Ordner*

Dies geschieht auf beiden Nodes mit den jeweiligen Node – Nummern als Verzeichnisnummer. Dieser Prozess wird mit Root – Rechten vollzogen, die Anpassung der Rechte erfolgt später auf bequeme Art und Weise mit einer in Gluster eigens dafür entwickelten Funktion.

Kommen wir nun zum Erstellungsprozess des Gluster – Volumes, das später den Main – Storage bereitstellen wird. Hierfür geht man per SSH zum Node1 (gfsn1.mgmtdom) und gibt folgendes ein:

```
# gluster peer probe gfsn2.mgmtdom
```

*Befehl 12: Realisierung des Storagesegmentes (Befehl): Kontaktaufnahme mit Node2 (GlusterFS)*

Dieser Befehl pusht Node1 zur Verbindungsaufnahme mit Node2. Dieser Schritt wird auf Node2 mit dem entsprechend angepassten Befehl (gfsn1.mgmtdom) wiederholt. Dies bindet beide Nodes in eine Kommunikationsverbindung, so dass sie sich untereinander wahrnehmen. Momentan geschieht aber nichts, dies ändert sich aber mit nachfolgendem Befehl:

```
# gluster volume create oVirtStorage1 transport tcp gfsn1.mgmtdom:/storage/brick1       \
gfsn2.mgmtdom:/storage/brick2
```

*Befehl 13: Realisierung des Storagesegmentes (Befehl): Erstellung des GlusterFS Volumes oVirtStorage1*

Dieser Befehl weist Gluster an, ein Volume namens **oVirtStorage1** zu createn, welches als Transportprotokoll **tcp** nutzt. Dabei werden nun die Nodes angegeben, gefolgt vom realen Pfad der beiden Verzeichnisse auf dem Filesystem (/storage/brick1/2). Dies funktioniert nur, wenn die Nodes mit entsprechendem Befehl in **Befehl 12** in eine logische Verbindung gebracht wurden. Das war es schon! Der Befehl in **Befehl 13** kann von irgend einem beliebigen Host im logischen Verbund abgesetzt werden, dies spielt bei Gluster absolut keine Rolle. Hier gibt es eine Menge an Zusatzoptionen, um diverse Replika – Verbünde zu kreieren. Gibt man jedoch keinen dieser Parameter an, so wird ein **Standard Distributed** Verbund erzeugt, in diesem kontaktiert der Client einen der Nodes und handelt mit ihm aus, wer sein direkter Ansprechpartner beim Speichern des jeweils aktuellen Datensatzes ist. Hier wäre ein Striping Verfahren die effizienteste Methode bei der Virtualisierung. Derzeit ist dies jedoch aus technischen Gründen nicht möglich, da die Locking Files vorerst direkt vom Client ins VM – Storage – Verzeichnis geschrieben werden. Das geht aber nicht, wenn das Verzeichnis auf mehrere reale Platten verteilt ist.

Anschliessend erzeugen wir den NFS – Mountpoint für den ISO Storage. Hierfür nutzen wir folgenden Befehl:

```
# gluster volume create ISO_STORE gfsn1.mgmtdom:/storage/ISO_STORE1       \
gfsn2.mgmtdom:/storage/ISO_STORE2
```

*Befehl 14: Realisierung des Storagesegmentes (Befehl): Erstellung des GlusterFS Volumes ISO_STORE1*

Dieser Befehl ist absolut identisch mit dem vorhergehenden, mit der Ausnahme, dass hier der Name **ISO_STORE1** verwendet wird. Technisch gesehen ist hier genau das selbe geschehen wie beim Anlegen von oVirtStorage1. Es wurde auch der gesamte Diskspace verwendet, so dass bei beiden der gleiche Maximalwert erscheint. Quotas an dieser Stelle zu setzen würde keinen Sinn ergeben, da die Menge der ISO's nie einen so grossen Diskspace füllen könnte.

Nun gehören die Volumes nach dem Erstellen aber dem Superuser Root, was nicht gerade vorteilhaft für die spätere Nutzung durch die Virtualisierungsnodes ist. Um an dieser Stelle etwas vorzugreifen, sollen die Volumes nun der oVirt spezifischen Gruppe übergeben werden. Diese Gruppe ist im Standard CentOS (RedHat, Fedora) mit der Gruppen – ID 36 (KVM, Mitglieder qemu und sanlock) definiert. Nun könnte man mühsam die Rechte mit dem altbewährten Werkzeug **chown** setzen, oder man benutzt die in Gluster integrierten Werkzeuge dafür.

```
# gluster set oVirtStorage1 storage.owner-uid=36
# gluster set ISO_STORE1 storage.owner-uid=36
```

*Befehl 15: Realisierung des Storagesegmentes (Befehl): Setzen der richtigen Rechte (GlusterFS)*

Diese beiden Befehle setzen die Besitzer der Volumes und passen auch gleich die Schreibrechte an. Diese Methode ist allgemein zu bevorzugen, da so mit Sicherheit alles automatisch richtig gesetzt wird.

Nachfolgend kann mit gluster volume info das fertige Ergebnis betrachtet werden.

```
Volume Name: ISO_STORE
Type: Distribute
Volume ID: 80103a4d-70e0-4b74-a448-49a469b9042d
Status: Started
Number of Bricks: 2
Transport-type: tcp
Bricks:
Brick1: gfsn1.mgmtdom:/storage/ISO_STORE1
Brick2: gfsn2.mgmtdom:/storage/ISO_STORE2
Options Reconfigured:
performance.readdir-ahead: on
storage.owner-uid: 36
storage.owner-gid: 36
cluster.quorum-type: auto
cluster.server-quorum-ratio: 51%
```

```
Volume Name: oVirtStorage1
Type: Distribute
Volume ID: 8dab7452-f032-4900-ae3c-88856c728e17
Status: Started
Number of Bricks: 2
Transport-type: tcp
Bricks:
Brick1: gfsn1.mgmtdom:/storage/brick1
Brick2: gfsn2.mgmtdom:/storage/brick2
Options Reconfigured:
performance.readdir-ahead: on
storage.owner-uid: 36
storage.owner-gid: 36
server.allow-insecure: on
network.ping-timeout: 10
cluster.quorum-type: auto
cluster.server-quorum-ratio: 51%
```

*Abbildung 22: Realisierung des Storagesegmentes (Befehl): Betrachtung der fertigen Gluster Volumes*

Die meisten hier ersichtlichen Optionen sind Defaultwerte. Einige wurden jedoch noch nachträglich hinzugefügt, da sie notwendig waren. Die meisten jedoch sind nicht notwendig und wurden bei der Splitt – Brain Problematik zu Debugging – Zwecken eingeführt und so stehen gelassen.

Dazu folgende Erklärungen:

**cluster.quorum-type: auto →** (Wurde eingeführt); Er definiert wie sich der Cluster verhalten soll, wenn die entsprechende Menge an möglichen Ausfällen eintritt. Hier automatisch die Arbeit einstellen.

**Cluster.server-quorum-ratio: 51% →** (Wurde eingeführt); Definiert die maximale Anzahl (prozentual) an möglichen Ausfällen. Der Wert ist sicherlich fragwürdig bei nur zwei Nodes, jedoch empfiehlt RedHat diesen Wert und es lässt sich ohnehin kein anderer definieren.

**Network-ping-timeout: 10 →** (Wurde eingeführt); Dies zu Debugging – Zwecken bei der Splitt – brain Problematik. Er definiert den Timeout, wie lange ein Gluster – Server auf eine möglicherweise unterbrochene Verbindung mit einem Client warten soll, bis er seine Verbindung als tot klassifisiert.

Dies war es eigentlich mit der Konfiguration der GlusterFS Volumes. Ein weiterführender Teil wird in einem späteren Abschnitt folgen, wo noch einige kleine Tricks beim Einbinden gezeigt werden. Jedoch muss seitens des Autors an dieser Stelle betont werden, dass das Erstellen von Mountpoints über verteilte Dateisysteme noch nie so einfach von der Hand ging wie mit diesem Top Produkt, welches

RedHat als simple Antwort auf CephFS einst einführte.  Es ist bei Gluster vollkommen egal, von welchem Standort aus man die Volumes bildet oder zu welchem Node man sich Zwecks Sharing connected.

**ALTERNATIVE**

**Hier wurden zwei externe Storages für die Bildung des Mountpoint verwendet. Es ist aber auch möglich die einzelnen Storages in den Virtualisierungsnodes zu betreiben. So spart man Rechner und Peripheriekomponenten wie Kabel.**

**ALTERNATIVE**

**Hier wurde als Transportprotokoll TCP verwendet. UDP ist eine massiv schnellere Alternative, wobei man einen möglichen Datenverlust seitens unvollständiger Schreibprozesse beachten muss.**

## 8.6 Beginn mit der Realisation des Virtualisierungssegmentes

In diesem Abschnitt soll ausschliesslich die Installation und Konfiguration der oVirt spezifischen Komponenten thematisiert werden. Dies beinhaltet die Installation der oVirt Komponenten mit ein wenig Konfigurationsaufwand. Die Konfigurationen wie Netzwerkinterfaces oder Storage Angelegenheiten werden im anschliessenden Kapitel thematisiert. Dieser Teil gliedert sich in zwei Abschnitte, die Installation des fertigen Node – Images auf den virtualisierungs- Maschinen und die Installation des oVirt Management Centers auf dem Supermicro Server.

### 8.6.1 Die Komponenten

In diesem Abschnitt sind nachfolgende Komponenten involviert:

#### 8.6.1.1 Die eigentlichen Virtualisierungsnodes

##### 8.6.1.1.1 Dell PowerEdge R810

Dieser Host ist der leistungsstärkste und wird Mitglied des leistungsstärkeren Cluster. Seine Spezifikationen sehen dabei wie folgt aus:

| Komponente | Wert |
|---|---|
| **CPU** | 2x Intel Xeon E7-4870 @ 2.40 GHz, 10 Reale Cores @ 2 Threads = 40 Cores |
| **RAM** | 257691 MB |
| **Swap** | 5999 MB |
| **NIC** | 8x Intel 1Gbit/s, davon 2x Interne Kommunikation, 3x VM Network |

*Tabelle 21: Realisierung des Virtualisierungssegmentes: Spezifikationen von Dell PowerEdge*

*Abbildung 23: Realisierung des Virtualisierungssegmentes: Dell PowerEdge Frontansicht*

Wie bei allen Hosts, dienen auch hier die lokalen Disks nur dem Aufnehmen der oVirt Installation. Dies hat den Vorteil, dass spätere Festplattenwechsel nicht die Integrität der VM Images gefährden. Um an dieser Stelle vorzugreifen, sei hier noch erwähnt, dass es keine passenden Festplattentreiber für die Dell Disks gibt. Hier musste ein lokaler RAID 0 auf Hardwareebene erzeugt werden, um eine Installationsmöglichkeit für das oVirt – Image überhaupt zu schaffen.

#### 8.6.1.1.2 Supermicro (Superworkstation) Eigenbau aus Barebone

Der Supermicro hat zwar knapp einen Viertel der Leistung des Dell, gehört aber ebenfalls zu den leistungsstärkeren Maschinen und wird sich den Platz mit dem PowerEdge im selben Cluster teilen. Seine Spezifikationen sehen wie folgt aus:

| Komponente | Wert |
|---|---|
| **CPU** | 2x Intel Xeon E5-2620 0 @ 2.00 GHz, 6 Reale Cores @ 2 Threads = 12 Cores |
| **RAM** | 64147 MB |
| **Swap** | 40267 MB |
| **NIC** | 8x Intel 1Gbit/s, davon 2x Interne Kommunikation, 3x VM Network |

*Tabelle 22: Realisierung des Virtualisierungssegmentes: Spezifikationen von Supermicro Barebone*

*Abbildung 24: Realisierung des Virtualisierungssegmentes: Supermicro Barebone Frontansicht*

In dieser Maschine ist 250 GB Festplatte verbaut, welche dem oVirt – Node als Installationziel dient. Trotz seines Aussehens aufgrund der später dazugekauften Rack – Montageschinen ist dies eine Workstation. Das Fehlen eines LOM – Interfaces bei diesem Node führte zur Einführung des Semi-Automatic-HA.

#### 8.6.1.1.3 Die baugleichen Fujitsu Primergy RX100 S7p

Diese beiden Maschinen sind eher niedrigerer Leistungsklasse und werden später in einem eigens dafür gebildeten Cluster arbeiten. Ihre primäre Funktion wird voraussichtlich die Bereitstellung von Infrastrukturdiensten sein. So können die internen Dienste wie bspw. DHCP, DNS und ähnliche auch auf Hardwareebene sauber getrennt werden. Die Spezifikation eines Node sieht dabei wie folgt aus:

| Komponente | Wert |
|---|---|
| **CPU** | 1x Intel Xeon E3-1230 V2 @ 3.30 GHz, 4 Reale Cores @ 2 Threads = 8 Cores |
| **RAM** | 7696 MB |
| **Swap** | 7943 MB |
| **NIC** | 4x Intel 1Gbit/s, davon 2x Interne Kommunikation, 1x VM Network |

*Tabelle 23: Realisierung des Virtualisierungssegmentes: Spezifikationen eines Fujitsu Primergy Servers*


*Abbildung 25: Realisierung des Virtualisierungssegmentes: 2x Fujitsu Primergy Frontansicht*

#### 8.6.1.1.4 Das Management Center, Supermicro Barebone 1HE

Dieser Rechner diente in der Vergangenheit schon vielen Zwecken, vom SmartOS Node bis zum netzweiten DHCP – DNS Server. Eine kurze Zeitspanne war er ausser Betrieb, bis er mit Beginn dieser Arbeit wieder ins Leben gerufen wurde. Seine Spezifikationen sind genau richtig für den Betrieb als Standalone Maschine und reichen vollkommen für das Aufrechterhalten eines Java Applicationservers. Seine Spezifikationen sind:

| Komponente | Wert |
|---|---|
| **CPU** | 1x Intel Atom D510 @ 1.66 GHz, 2 Reale Cores @ 2 Threads = 4 Cores |
| **RAM** | 1830 MB |
| **Swap** | 2048 MB |
| **NIC** | 2x Intel 1Gbit/s, davon 1x Interne Kommunikation |

*Tabelle 24: Realisierung des Virtualisierungssegmentes: Spezifikationen von Supermicro Barebone 1HE*

*Abbildung 26: Realisierung des Virtualisierungssegmentes: Supermicro Barebone 1HE Frontansicht*

Dies sind die fünf Hauptdarsteller diese Abschnittes. Bis auf den Supermicro Barebone 1HE waren alle einst Mitglieder eines Xenserver 6.2 Clusters, was die Verkablung und Montage der einzelnen Komponenten massiv vereinfachte.

### 8.6.2 Installation der oVirt Nodes

Die Installation der oVirt Nodes ist eine simple Angelegenheit. Das fertige Installations- ISO lässt sich auf der oVirt Website für ältere aber auch für neuere Preview Versionen herunterladen und auf einen Datenträger brennen. Die hier verwendete Version ist 3.5 – 0.999.20150428093 und entsprach bei Beginn dieses Projektes der aktuellsten. Wie bereits mehrfach erwähnt, ist dies die einfachste aber auch zugleich sicherste Variante der Installation. Denn diese Installation ist seitens oVirt Projekt speziell optimiert und abgehärtet worden. So fehlen einige elementare Tools des Userlands wie bspw adduser, welches zum Anlegen neuer User notwendig ist. Diese Massnahme soll eine komplett in sich gekapselte Installation darstellen, welche sich nur von der oVirt Engine steuern lässt. Installationen von Zusatzsoftware wie etwa ein Zabbix Agent sind erst gar nicht möglich, da man keine Möglichkeit hat Repositories einzubinden. Wer gerne ein System mit vollem Zugriff möchte, dem sei an dieser Stelle wärmstens die reguläre Verwendung einer CentOS Version empfohlen.

Bootet man die Rechner mit eingelegter CD, landet man direkt beim Installer. Dieser stellt wenige elementare Fragen wie:

- Wohin soll installiert werden?
- Die Partitionierung der Basisinstallation ist vorgegeben. Man muss nur noch bestätigen oder ggf. noch die Grössen anpassen. An dieser Stelle ist auch die Partitionierung möglicher weiterer Disks für die Verwendung als Local Storage möglich. Jedoch ist eine solche Standalone Konfiguration bei einem Produkt wie oVirt sinnlos.
- Die Swap – Partition kann auch definiert werden. Hier scheint der Installer grosszügig mit den Ressourcen umzugehen, da er mehr als das gewöhnliche Drittel des Arbeitsspeichers wählt. Bei grossen Arbeitsspeichern kann es vorkommen, dass er zu viel Diskspace wählt und nicht genügend für die Installations- Partition übrig bleibt (so beim Dell geschehen).
- Die Netzwerkkonfiguration kann bei diesem Schritt noch mit einem NIC vollzogen werden. Die Betitlung der Interface Namen ist etwas kompliziert, da hier das Full Path – Verfahren angewendet wird. So ergeben sich Namen wie enp2s0f1, welche sich nur schwer einem bestimmten Port zuweisen lassen. Geht man aber auf das Interface, so besteht die Möglichkeit, den Port für ca. fünf Sekunden aufblinken zu lassen. Man muss sich hier nicht bemühen, schnellstmöglich nach dem Port zu suchen. Schliesst man lediglich ein Netzwerkkabel an, so kann man alle Ports nacheinander durchblinken lassen. Der angeschlossene wird in der Statusleiste beim Aktivieren der Bilk – Funktion automatisch in den **active** Modus wechseln. Anschliessend kann man die Basisparameter wie Default Gateway, DHCP, DNS und natürlich die IP – Konfiguration vornehmen.
- Im Anschluss muss man nur noch das Passwort für den einzig einsetzbaren User **admin** eingeben.

Die Installation ist intuitiv und man wird auf Fehler hingewiesen. Bei Systemen ohne hardwareseitige Virtualisierungs- Unterstützung, oder in virtuellen Umgebungen, wird man mehrfach darauf hingewiesen, dass diese fehle. Eine Installation ohne hardwareseitige Virtualisierungs- Unterstützung ist technisch gesehen möglich, macht aber keinen Sinn. Das installierte System wird dann auch beim ersten Booten hängen bleiben und in einen Emergency Mode fallen. Dies war es eigentlich auch schon mit der Installation der oVirt vordefinierten Images, sie ist wie gesagt simpel und in wenigen Minuten erledigt.

Nachfolgend die Definitionen der Namensräume und deren statischen IP's:

| Node Name | IP- Adresse | Subnetzmaske | Hostname | Domainname |
|---|---|---|---|---|
| oVirt Node 1 | 10.10.10.41 | 255.255.255.0 | ovn1 | mgmtdom |
| oVirt Node 2 | 10.10.10.42 | 255.255.255.0 | ovn2 | mgmtdom |
| oVirt Node 3 | 10.10.10.43 | 255.255.255.0 | ovn3 | mgmtdom |
| oVirt Node 4 | 10.10.10.44 | 255.255.255.0 | ovn4 | mgmtdom |

*Tabelle 25: Realisierung des Virtualisierungssegmentes: Das Adress- und Namensschema der oVirt Virtualisierungsnodes.*

### 8.6.3 Installation der oVirt Engine (Management Center)

Die Installation der Engine ist technisch gesehen einfach. Jedoch überschnitt sich die Realisierung des offiziellen Teiles dieser Arbeit mit der Rückportierung des kürzlich in Produktivbetrieb gegangenen oVirt 3.6. Dabei wurden einige Neuerungen von oVirt 3.6, speziell die Installationsverfahren und die damit abhängigen Repositories, nach 3.5 portiert. Dies führte dazu, dass die bei den Vortests ermittelten Installationswege nicht mehr ganz gleich waren. Zusätzlich kam der open source Faktor hinzu, welcher zwar immer aktuelle Topprodukte garantiert, aber diese nur spärlich dokumentiert. Hier ging einiges an Zeit und Nerven verloren, da die Fehler- bzw. die Logfileanalyse einiges an Zeit kostete.

Dieser Abschnitt soll sich mit der Installation einer oVirt Engine 3.5 beschäftigen und dabei die Einrichtung auf einer Standalone Maschine thematisieren. Hier wird speziell auf die Eigenheiten der heute bereits als veraltet  geltenden Installation der Version 3.5 eingegangen, für welche es heute kaum korrekte Anleitungen mehr im Internet gibt.

#### 8.6.3.1 Installation des Grundsystems (CentOS 7)

Als Betriebssystem wird und muss an dieser Stelle CentOS oder ein anderer RedHat Abkömmling genutzt werden. Auf die Installation der Minimalinstallation soll hier nicht weiter eingegangen sondern auf die äquivalente Anleitung unter Punkt 8.5.3 „Installation des CentOS 7“ hingewiesen werden.

Hier wird folgendes Namensraumschema verwendet:

| Node Name | IP- Adresse | Subnetzmaske | Hostname | Domainname |
|---|---|---|---|---|
| Ovirt Engine | 10.10.10.5 | 255.255.255.0 | Ovirt-mgmt | mgmtdom |

*Tabelle 26: Realisierung des Virtualisierungssegmentes: Das Adress- und Namensschema der oVirt Engine*

#### 8.6.3.2 Installation der notwendigen Software

##### 8.6.3.2.1 Installation und Konfiguration von VDSM

Nach erfolgreicher Installation müssen nun die notwendigen Pakete installiert werden. Hier wies das

Internet einst auf den simplen Installations- Befehl **yum install ovirt-engine-setup** hin. Dies ist aber seit ca. November 2015 nicht mehr möglich und yum antwortet stets mit der Antwort; **Fehler: Nichts zu tun**. Die Lösung schien zunächst einfach, die oVirt Pakete sind nicht mehr Bestandteil der Standard Repositories und müssen von Hand eingebunden werden. Eine kurze → Google – Anfrage ergab, dass man die Paketquelle einfach mit dem Befehl **yum install centos-release-ovirt35.noarch** einrichten lassen kann. Nach diesem Schritt ist das CentOS eigene Repository eingebunden und man kann einige Tools, welche für den Betrieb eines CentOS Systems als Hypervisor notwendig sind, nachinstallieren. Jedoch noch immer nicht das zwingend benötigte **ovirt-engine-setup**.

An dieser Stelle verlassen wir die **ovirt-engine-setup** Problematik und widmen uns der Installation der Komponenten die wenigstens nach dem Hinzufügen des oVirt eigenen Repositories möglich sind. Dieser Punkt wurde im Abschnitt der Storage Node Installation angesprochen und soll hier für die oVirt Engine und die beiden Storage Nodes präziser thematisiert werden.

Der VDSM Daemon (Virtual Desktop Server Manager) ist ein Hintergrunddienst, welcher von oVirt entwickelt wird. Er stellt eine Abstraktionsschicht dar, welche einerseits mit der oVirt Engine und anderseits mit den Nodes bzw. dem darunterliegenden libvirt kommuniziert. Libvirt selbst stellt eine mächtige API zur Verfügung, mit welcher es möglich ist, KVM über die Kommandozeile (virsh) und über grafische Tools wie dem virt-manager zu managen. Jedoch wären die Befehlsaufrufe zur Steuerung der VM's unendlich lang und die Einbindung in ein Framework wie oVirt massiv komplizierter. Aus diesem Grund soll der VDSM Dienst als Vermittler zwischen der Engine und libvirt dienen. Zusätzlich kann VDSM auch Funktionen übernehmen, die libvrit selbst nicht beherrscht, wie etwa die HA Bereitstellung, das Monitoring der Hosts selbst oder das gesamte Storage – Management über GlusterFS. Nachfolgend eine grafische Abbildung von oVirt selbst über das Gesamtkonzept der Virtualisierungs- Technik:

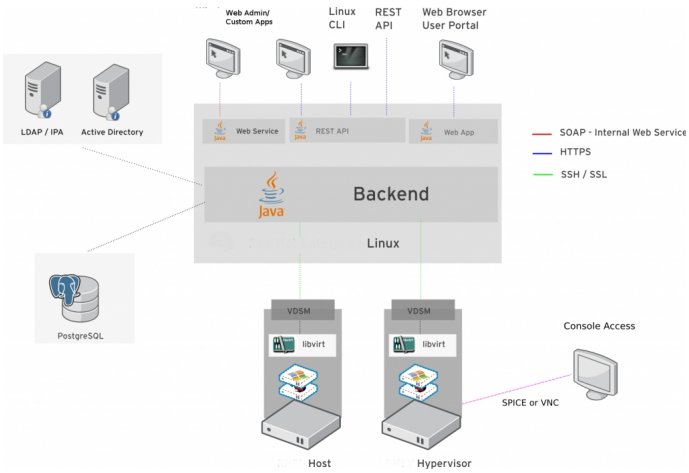



*Abbildung 27: Realisierung des Virtualisierungssegmentes: Gesamtschema einer oVirt Umgebung @ http://www.ovirt.org/Architecture*

Die Installation von VDSM auf den beiden Storages und der Engine ist nicht genau beschrieben und führt beim simplen Aufruf von **yum install vdsm** zu einer nicht kompletten Installation. Der Grund ist, dass die Installation dynamisch ist und bei Bedarf auf ein Minimum reduziert wird. Man kann also

bspw. die Basis installieren und zusätzliche Komponenten wie etwa vdsm – Gluster als Option wählen. Da das Herauspicken der einzelnen Komponenten eine mühsame Arbeit darstellt, kann hier über eine simple Abkürzung einfach alles installiert werden. Diese Methode umfasst ca. 240 MB an benötigtem Diskspace und ist in dieser Form hinnehmbar. Der exakte Installationsaufruf hier lautet:

```
# yum install vdsm-*
```

*Befehl 16: Realisierung des Virtualisierungssegmentes (Befehl): Installation von VDSM*

Dies genügt, um eine komplette VDSM – Installation zu bekommen. Nach der Installation muss der VDSM Daemon noch eingerichtet werden. Dies geschieht mit nachfolgendem Befehl:

```
# vdsm-tool --configure force
```

*Befehl 17: Realisierung des Virtualisierungssegmentes (Befehl): Einrichten vom VDSM Daemon auf den beiden Storage Nodes und der oVirt Engine*

Dieser Befehl gehört zu den wenigen, welche noch sauber in den oVirt Dokumentationen erklärt sind. Seine Funktion ist die vollkommen automatische Einrichtung der für oVirt notwendigen Dienste wie sanlock Daemon oder die Firewall Konfiguration. Die wohl wichtigste Aufgabe dieses Befehls ist das Erstellen und Einrichten der ovirtmgmt Netzwerkschnittstelle, welche als Bridge auf den Nodes die Kommunikation der Hypervisor mit der Engine ermöglicht. Ohne diese Schnittstelle verweigert die Engine bei einem späteren Aufnahmegesuch seitens des Nodes die Integration ins Datacenter. Weshalb das Argument **force** notwendig ist, wird in der oVirt Dokumentation nicht erwähnt, es funktioniert aber nicht ohne.

Dies war eigentlich die gesamte notwendige Konfiguration von VDSM. Die bis hierhin erklärten Schritte können ggf. auf jeder CentOS Minimalinstallation ausgeführt werden, um jeden beliebigen Host als Hypervisor zu konfigurieren. Dies ist der Alternativweg für alle, die nicht das fertige oVirt – Node Image verwenden möchten. Hier muss auch klar gesagt werden, dass die automatische Konfigurations- Routine des oben genannten Befehls jede CentOS Maschine so umkonfiguriert, als ob es ein fertiges Node – Image wäre. Man muss hier aber vorsichtig sein, da das Repository von CentOS stammt und somit die Frage nach dem Updaten der Komponenten und somit die Kompatibilität zum original Projekt nicht definitiv sicher ist.

Die oVirt Engine selbst braucht im Grunde keinen VDSM Daemon um die Nodes zu managen, diese Aufgabe kann der Java Applicationserver auch selbstständig übernehmen. Hier war der Grundgedanke des Autors, die Engine selbst in den Cluster als Node einzubinden und seinen lokalen Storage als NFS Share für ISO – Images einzubinden. Da VDSM aber vor Beginn der Konfiguration prüft, ob eine hardwareseitige Virtualisierungs- Unterstützung vorhanden ist, bricht er bei Nichtvorhandensein ab. Ohne ist aber die Einbindung des Storages (Fehlen der Netzwerkschnittstelle ovirtmgmt) aber nicht möglich. Aus diesem Grund wurde die VDSM Installation wieder vom Host entfernt.

#### 8.6.3.2.2 Installation des Management – Centers

Nach dieser kleinen nervlichen Ablenkung können wir uns wieder dem Installationsproblem mit der eigentlichen Engine Installation widmen. Hier war nach einer Suche auf der der Download – Seite des oVirt Projektes schnell klar, dass möglicherweise eine manuelle Einbindung auf dem hier verfügbaren Repository notwendig sein könnte. So wurde in **/etc/yum.repos.d/** ein bestehendes File als Pattern kopiert und entsprechend angepasst. Hierbei wurde das Zielverzeichnis auf dem Download Server unter **mirrorlist** eingetragen. Die Verwendung des Keyrings (gpgcheck), um eine Repository Verifizierung zu ermöglichen, scheiterte, woraufhin entschieden wurde, der Quelle ausnahmslos zu vertrauen und auf die Verwendung der Sicherheitsprüfung zu verzichten. Das neu angelegte Repository – File trägt den Namen **CentOS-Engine-Setup_oVirt35.repo** und hat folgende Einstellungen:

```
# CentOS-Engine-Setup.repo
#
# Please see http://wiki.centos.org/SpecialInterestGroup/Virtualization for more
# information

[ovirt35-engine-setup]
name=oVirt 3.5- Engine-Setup
baseurl=http://resources.ovirt.org/pub/ovirt-3.5/rpm/el7Server
gpgcheck=0
enabled=1
```

*Abbildung 28: Realisierung des Virtualisierungssegmentes: Neu angelegtes Repository - File CentOS-Engine-Setup_ovirt35.repo*

Nach der Erstellung dieses Files kann mittels yum update die Quelle initialisiert werden. Nun kann wieder versucht werden, **ovirt-engine-setup** zu installieren, da es ja nun bei einer Suche mit **yum search ovirt-engine-setup** als aktives Paket gefunden wurde. Jedoch scheitert auch dieser Versuch mit der Fehlermeldung, dass zwei PHP Module und das Tool novnc in keiner Paketquelle gefunden wurden. Hierauf wurde versucht nach den vermissten Paketen im Internet zu suchen, wobei novnc einen Treffer ergab. Es ist Bestandteil der nicht Standard Pakete von Fedora und gehört auch nicht zu den Standard Werkzeugen der RedHat Infrastruktur Philosophie. Somit war auch dem Autor nach langer vorgängiger Suche im Internet klar: Es muss sich in dem Repository **epel** verstecken. Hierbei handelt es sich um die Fedora Extra Packages for Enterprise Linux, welches die Fedora Community für Neuentwicklungen im eigenen Serverumfeld nutzt, die aber aus Sicht von RedHat nicht notwendig sind für ihr Produkt Portfolio. Glücklicherweise kann diese Quelle bequem über die Paketverwaltung mit nachfolgendem Befehl nachinstalliert werden.

```
# vdsm-tool --configure force
```

*Befehl 18: Realisierung des Virtualisierungssegmentes (Befehl): Installieren der Paketquelle epel*

Voller Hoffnung wurde nun wieder versucht, **ovirt-engine-setup** zu installieren, was aber wieder scheiterte. An dieser Stelle verweigerte wieder das Fehlen zweier ominöser PHP Module das Installieren der Engine.

Nun wurde wieder eine lange Google – Suchaktion gestartet, welche den Autor auf eine Seite brachte, wo es um das Webinterface von Cockpit ging. Bei Cockpit handelt es sich um das WebGUI des unter

Fedora entwickelten Management Tools, welches für die Konfiguration (auch Docker Container) verwendet wird. Hier bemerkte der Autor auch gleich die Ähnlichkeiten zwischen den WebGUI's der beiden Projekte, woraufhin klar war, dass hier wahrscheinlich eine HTML und JavaScript Vorlage zur Erstellung dienten. Nun, nachdem klar war, wie die Sucheingaben bei Google lauten mussten, wurde auch gleich die Seite **https://www.patternfly.org/** entdeckt. Auf der Seite gibt es auch RPM Pakete und ein Verweis auf den Repository Server. Nun konnte auch dieses Repository – File manuell angelegt werden, wobei hier der Keyring Server auch tatsächlich funktionierte. Nachfolgend der Auszug aus dem neu angelegten Repository – File mit Namen **CentOS-Patterfly_oVirt35.repo**:

```
# CentOS-Patternfly.repo
#
# Please see http://wiki.centos.org/SpecialInterestGroup/Virtualization for more
# information

[ovirt35-patternfly]
name=Copr repo for patternfly1 owned by patternfly
baseurl=https://copr-be.cloud.fedoraproject.org/results/patternfly/patternfly1/epel-7-$basearch/
skip_if_unavailable=True
gpgcheck=1
gpgkey=https://copr-be.cloud.fedoraproject.org/results/patternfly/patternfly1/pubkey.gpg
enabled=1
enabled_metadata=1
```

*Abbildung 29: Realisierung des Virtualisierungssegmentes: Neu angelegtes Repository - File CentOS-Patternfly_oVirt35.repo*

Nun kann wieder versucht werden, die Engine zu installieren und tatsächlich: Alle Abhängigkeiten sind vorhanden und der Engine – Installer wurde erfolgreich auf das System gebracht! Nachfolgend der lang ersehnte Befehl zur erfolgreichen **ovirt-engine-setup** Installation:

```
# yum install ovirt-engine-setup
```

*Befehl 19: Realisierung des Virtualisierungssegmentes (Befehl): Installations von ovirt-engine-setup*

Nach Installation des Engine – Installers ist der gröbste Teil abgeschlossen und man kann sich der eigentlichen Installation des Applicationservers widmen.

### 8.6.3.3 Einrichten der oVirt Engine (ovirt-engine-setup)

Die Einrichtung der Engine wird mittels des unter Befehl 13 aufgeführten Befehls initialisiert. Man wird zu einem Dialoggeführten Interface geführt, welches die elementarsten Fragen zur Einrichtung abfragt. Hierbei orientiert sich der Assistent an folgenden Fragekategorien:

```
# engine-setup
```

*Befehl 20: Realisierung des Virtualisierungssegmentes (Befehl): Initialisierung der oVirt Installation*

#### 8.6.3.3.1 Network Configuration

In diesem Abschnitt werden zwei Fragen gestellt. Die erste betrifft die Aktivierung und automatische Konfiguration der Firewall (firewalld). Hier kann, um einen hohen Sicherheitsstandard zu erreichen mit Yes geantwortet werden. Die zweite Frage will den FQDN (Fully qualified DNS name) des Systems wissen. Sollte kein DNS Server zur Verfügung stehen, welcher die Auflösung übernimmt, muss ein Eintrag in /etc/hosts mit der Syntax <IP> <Host.Domainname> <Hostname> eingetragen werden, da ansonsten der Assistent nicht fortsetzt.

```
--== NETWORK CONFIGURATION ==--

Setup can automatically configure the firewall on this system.
Note: automatic configuration of the firewall may overwrite current settings.
Do you want Setup to configure the firewall? (Yes, No) [Yes]:
iptables will be configured as firewall manager.
Host fully qualified DNS name of this server [livecd.localdomain]: ovirt-mgmt.mgmtdom
```

*Abbildung 30: Realisierung des Virtualisierungssegmentes: Einrichten der Engine; Network Configuration - Screenshot*

#### 8.6.3.3.2 Database Configuration

An dieser Stelle möchte der Assistent wissen, was er mit der Datenbank machen muss. Hier wurde eine lokale Installation gewählt, da zur Zeit der Realisation keine externe DB zur Verfügung stand.

```
--== DATABASE CONFIGURATION ==--

Where is the Engine database located? (Local, Remote) [Local]:
Setup can configure the local postgresql server automatically for the engine to run. This may conflict with existing applications.
Would you like Setup to automatically configure postgresql and create Engine database, or prefer to perform that manually? (Automatic, Manual) [Automatic]:
```

*Abbildung 31: Realisierung des Virtualisierungssegmentes: Einrichten der Engine; Database Configuration - Screenshot*

#### 8.6.3.3.3 oVirt Engine Configuration

Dieser Abschnitt ist simpel im Aufbau. Er möchte nur wissen, wie das lokal in die Postgres SQL gespeicherte Passort lautet. Hier gibt es eine Längenprüfung, welche die Komplexität des Passwortes in Frage stellt. Es empfiehlt sich an dieser Stelle, ein sicheres Passwort zu wählen, da dieser Account die vollen Rechte des späteren Management – Centers besitzen wird und er sich weder deaktivieren noch löschen lässt.

Zusätzlich wird man gefragt, ob die Virtualisierungs- Subsysteme (VDSM) und GlusterFS installiert und konfiguriert werden sollen. Beim Supermicro 1HE macht im Grunde beides keinen Sinn, da er weder virtualisieren kann noch genügend Diskspace für ein Gluster Volume besitzt. Man muss hier aber eine Entscheidung treffen und kann ohne weiteres **Both** wählen. Es braucht später keines von beidem und der Assistent ist zufrieden und kann ohne weitere Beschwerden weiter machen.

```
--== OVIRT ENGINE CONFIGURATION ==--

Engine admin password:
Confirm engine admin password:
Application mode (Virt, Gluster, Both) [Both]:
```

*Abbildung 32: Realisierung des Virtualisierungssegmentes: Einrichten der Engine; oVirt Engine Configuration - Screenshot*

#### 8.6.3.3.4 PKI Configuration

In dieser Sektion wird automatisch ein Zertifikat (Public Key Infrastruktur) generiert, welches später https für das Management – Center ermöglicht. Hier ist aus Sicherheitsgründen auch nur https möglich, simples http wird per Redirecting stets zum SSL verschlüsselten URL umgeleitet. Hier muss nur ein sinnvoller bzw. bei Unternehmen ein zulässiger Name vergeben werden.

```
--== PKI CONFIGURATION ==--

Organization name for certificate [mgmtdom]: TB-Network
```

*Abbildung 33: Realisierung des Virtualisierungssegmentes: Einrichten der Engine; PKI Configuration - Screenshot*

#### 8.6.3.3.5 Apache Configuration

An dieser Stelle fragt der Assistent zuerst, ob man die oVirt Frontpage als Standardseite konfiguriert haben möchte: Dies macht bei der Funktion dieser Standalone Maschine absolut Sinn und soll auch so übernommen werden.

Die zweite Frage richtet sich an das im Schritt „PKI Configuration“ erzeugte Zertifikat und die damit verbundene CA (Certificate Authority) welche ebenfalls in diesem Schritt lokal generiert wurde. Diese Frage kann mit **Automatic** bestätigt werden, was zu einer automatischen SSL Signierung führt. Unternehmen können hier eigne und gültige Zertifikate einspielen, um die lästige Browser – Warnung zu umgehen.

```
--== APACHE CONFIGURATION ==--

Setup can configure the default page of the web server to present the application home page. This may conflict with existing applications.
Do you wish to set the application as the default page of the web server? (Yes, No) [Yes]:
Setup can configure apache to use SSL using a certificate issued from the internal CA.
Do you wish Setup to configure that, or prefer to perform that manually? (Automatic, Manual) [Automatic]:
```

*Abbildung 34: Realisierung des Virtualisierungssegmentes: Einrichten der Engine; Apache Configuration - Screenshot*

#### 8.6.3.3.6 System Configuration

An dieser Stelle kann ein ISO Storage mittels NFS Share eingerichtet werden. Dies macht aber keinen Sinn, da VDSM zwar als einzurichten angegeben wurde, es aber nicht auf dem Supermicro 1HE lauffähig ist. Somit ist die Einbindung der ISO Domäne später ohnehin nicht möglich.

```
--== SYSTEM CONFIGURATION ==--

Configure an NFS share on this server to be used as an ISO Domain? (Yes, No) [Yes]: no
```

*Abbildung 35: Realisierung des Virtualisierungssegmentes: Einrichten der Engine; System Configuration - Screenshot*

#### 8.6.3.3.7 Misc Configuration

Dieser Abschnitt blieb beim eigentlichen Einrichten und bei diversen Vortests stets leer. Welche Funktion er hat, kann nicht mit Sicherheit gesagt werden. Es wird vermutet, dass dies ein Script – Überbleibsel einer älteren oVirt Version ist.

#### 8.6.3.3.8 Abschlussbestätigung

Nach all diesen Fragen präsentiert der Assistent eine Zusammenfassung der Konfigurationsparameter, welche noch bestätigt werden müssen. Nach Bestätigung beginnt der Assistent die Konfiguration, was je nach System eine Weile dauern kann. Nach Beendigung der Konfigurationsarbeiten seitens des Assistenten, startet er auch gleich die Engine und trägt den Service ins Startverfahren von Systemd ein, was die Installation auch rebootsicher macht.

```
--== CONFIGURATION PREVIEW ==--

Application mode                        : both
Firewall manager                        : iptables
Update Firewall                         : True
Host FQDN                               : ovirt-mgmt.mgmtdom
Engine database name                    : engine
Engine database secured connection      : False
Engine database host                    : localhost
Engine database user name               : engine
Engine database host name validation    : False
Engine database port                    : 5432
Engine installation                     : True
PKI organization                        : TB-Network
Configure VDSM on this host             : False
Configure local Engine database         : True
Set application as default page         : True
Configure Apache SSL                    : True
Configure WebSocket Proxy               : True
Engine Host FQDN                        : ovirt-mgmt.mgmtdom

Please confirm installation settings (OK, Cancel) [OK]:
```

*Abbildung 36: Realisierung des Virtualisierungssegmentes: Einrichten der Engine; Abschlussbestätigung - Screenshot*

#### 8.6.3.3.9 Abschliessende Bemerkung

Die hier erzeugten Screenshots wurden in einer VMware Workstation mit einem Live – System erzeugt. Das verwendete Live – System ist von der Version her identisch mit dem eigentlichen oVirt (3.5 Subversion 6), welches in dieser Arbeit fix auf einem CentOS 7 installiert wurde. Die eigentliche Installation auf dem Supermicro 1HE wurde in einer Konsole direkt am Gerät vollzogen, was das schiessen von Screenshots verunmöglichte. Die im Live – System verwendete CentOS Version entspricht nicht der gleichen wie die der fix installierten, was aber technisch gesehen egal ist, da die oVirt Version und somit das Verhalten des Assistenten identisch sind.

### 8.6.4 Abschluss der freischwebenden Einrichtung

Dies war auch schon die gesamte Einrichtung des Virtualisierungs- Environments. Ab diesem Punkt sind alle Nodes inklusive des Management – Centers installiert und auf einer tiefen Ebene konfiguriert. Wie in den wenigen Schritten zu sehen war, ist die Einrichtung mit Hilfe der fertigen oVirt – Node Images relativ einfach. Was bei diesem Arbeitsschritt am meisten Zeit gekostet hat, war das Debuggen der Engine – Installation. Hier zeigte sich die Schwäche von open source Communities, Neuentwicklungen vielleicht nicht zeitnahe aber wenigsten 1 – 2 Monate nach Freigabe des Finale Releases sauber dokumentieren zu können. So waren aufgrund der teilweisen Rückportierung von in Version 3.6 als stabil eingestuften Features nach Version 3.5 der Grund, warum viele in den Vortests bekannte Websites massiv nach hinten rutschten in der Google Suche.

**ALTERNATIVE**

**Mit Version 3.6 wurde die Hosted – Engine als quasi Standard freigegeben und nach 3.5 migriert. Mit ihr ist der Betrieb der Engine auch innerhalb der Virtualisierung möglich. So können Kosten für Standalone Maschinen und deren Komponenten eingespart werden.**

**TIPP**

**Wer sich bei der Hosted Engine Sorgen um die Hochverfügbarkeit macht, sei an dieser Stelle beruhigt. Mit Version 3.6 wurde auch die Hosted Engine HA zurückportiert. Hierbei kann die Engine über einen NFS Share innerhalb der Virtualisierung installiert werden. Dabei ist den Virtualisierungs- Nodes die Existenz der Engine bewusst und eine in VDSM integrierte Funktion sorgt dafür, dass die Nodes sich im Notfall selbst um den Neustart der Engine auf einem anderen Node kümmern. Diese Funktion ist datacenterweit gültig.**
**STICHWORT: ovirt-hosted-engine-setup / hosted-engine --deploy**

## 8.7 Viele Einzelteile ein Cluster

Bis hier hin haben wir drei Einzelsegmente installiert und konfiguriert, nun wird es Zeit, aus vielem eins zu machen. In diesem Abschnitt befassen wir uns mit der Zusammenführung der Teile Netzwerk, Storage und Virtualisierung und bilden einen Cluster aus einem Guss. Hierbei wird die tiefste Grundkonfigurationsebene behandelt. Aufgrund des angewendeten Konfigurationsverfahrens bis hier hin kann alles vom oVirt Management – Center aus gesteuert werden. Dieser Teil befasst sich mit folgenden Grundkonfigurationen:

- Erstellung des Data – Centers
- Erstellung der Cluster
- Anmelden/ Registrieren der Nodes
- Einbinden des GlusterFS Volumes als Master Storage
- Einbinden des ISO_STORE's per NFS
- Erstellen der Bondings für das Management – Netzwerk
- Erstellen des Netzwerks für die VM's

Nachfolgend werden die Schritte nach dem Verfahren „dokumentieren – beim – konfigurieren“ aufgebaut. Dies bedeutet, alle Konfigurationsschritte werden nach dem Anwenden auch gleich niedergeschrieben.

### 8.7.1 Erstellung des Data – Centers

Der erste Schritt in Virtualisierungs- Clustern ist stets die Erstellung mindestens eines Data – Centers. Hierzu klicken wir als **admin** eingeloggt folgenden Weg:

```
System (Seitenleiste) ausrollen → Data – Center
```

*Befehl 21: Realisierung des Virtualisierungssegmentes (Klicken): oVirt - Engine; Erstellung eines Data - Centers*

Im sich nun öffnenden Fenster sieht man bereits das standardmässig angelegte Data – Center **Default**. Es kann gelöscht werden, man kann es aber in diesem rohen Zustand auch einfach umbenennen. Hierfür klickt man sich ins Kontextmenü und wählt **Bearbeiten**. Nun kann man den Namen ändern und eine sinnvolle Beschreibung einfügen. Der **Speichertyp** wird auf gemischt gesetzt, da wir später GlusterFS und NFS nutzen möchten. Das Feld **Kompatibilitätsversion** sollte auf 3.5 belassen werden, ausser man hat bereits einen bestehenden Cluster von einer älteren Version, dann muss diese Version angegeben werden. Die Kategorie **Kontingentmodus** konnte nicht genau definiert werden. Es finden sich auch keine genauen Beschreibungen zu dieser Kategorie und der verwendeten Version 3.5. Hier muss angenommen werden, dass es aus einer älteren Version stammt und einst den Zweck der Globalen Ressourcen Zuteilung hatte. Hier wird **Deaktiviert** gewählt. Der letzte Parameter ist die Eingabe eines Kommentars, welcher zusätzliche und von der Länge her unabhängige Zusatzinformationen bietet.

Nachfolgend die getätigten Eingaben als Screenshot:

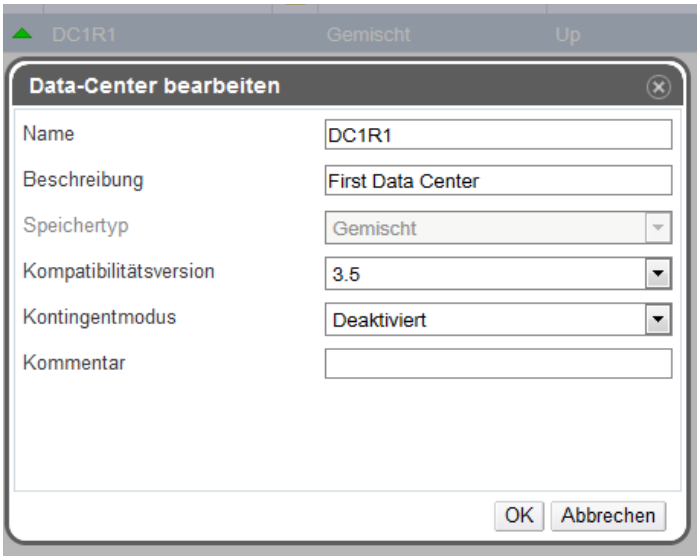



*Abbildung 37: Realisierung des Virtualisierungssegmentes: Erstellung eines Data - Centers*

Der hier gewählte Name **DC1R1** ist logischen Ursprungs und besagt, dass es sich hier um das Data – Center 1 mit den Maschinen aus Rack 1 handelt. Dies kann mit **OK** bestätigt werden und das war es.

### 8.7.2 Erstellung der Cluster

Der nächste logische Schritt in der Cluster Bildung ist auch gleich die Erstellung eines, oder wie in diesem Fall gleich drei, Clustern. Hierfür geht man in der Seitenleiste folgenden Weg:

System (Seitenleiste) ausrollen → Cluster

*Klicken 22: Realisierung des Virtualisierungssegmentes (Klicken): oVirt - Engine; Erstellung der Cluster*

Hier kann man auf Cluster klicken und findet sich auf einer leeren Seite wieder. Mittels Klick auf **Neu**, öffnet sich ein Konfigurationsfenster, in welchem folgende sechs Kategorien vorhanden sind, welche die gesamte Konfiguration des Cluster verlangen. Hier soll exemplarisch eine Konfiguration am Storage – Cluster vorgenommen werden, welche von der Dokumentation her gesehen auf alle weiteren Cluster übertragbar ist. In den nun nachfolgenden Schritten sollen alle Konfigurationskategorien einzeln durchgearbeitet werden. Die Verweise auf die anderen beiden Cluster werden dabei immer in Klammern gesetzt. Hier gilt folgender Farbcode (Wo keine Farbcode – Angabe sind gilt die Einstellung für alle gleich)

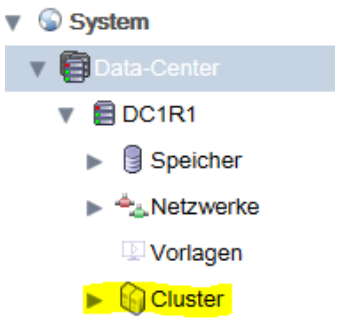



*Abbildung 38: Realisierung des Virtualisierungssegmentes: Der Klick - Weg zum Cluster erstellen*

- Cluster-Storage → Bleibt exemplarisch Scharz
- Cluster-Level-High → ROT
- Cluster-Level-Middle → BLAU

**Allgemein:**

- **Data – Center**: Hier kann das zu verwendende Data – Center ausgewählt werden (DC1R1)
- Name: Der neue Name des Clusters. Er sollte aussagekräftig sein → Cluster-Storage, Cluster-Level-high, Cluster-Level-Middle
- **Beschreibung**: Hier empfiehlt es sich einen Bezugspunkt zu wählen, der auch auf die logische Trennung der Cluster hinweist. An dieser Stelle wurde die CPU Familie als logischer Kontext gewählt. Somit gilt → Storage for all Nodes, Cluster-SandyBridge, Cluster-Westmare
- Der **Kommentar** ist frei wählbar und soll hier nicht thematisiert werden.
- Der **CPU-Typ** ist hier von elementarer Bedeutung. Er wurde schon in der Hauptstudie thematisiert und führte zu folgender Aufteilung:
  - Cluster-Storage → Intel Nehalem Family
  - Cluster-Level-M → Intel SandyBridge Family
  - Cluster-Level-High → Intel Westmere Family
- Die **Kompatibilitätsversion** kann hier auf die aktuelle Version gesetzt werden.
- Die Optionen des **Zufallsgenerators** sind nicht zwingend notwendig. Sie beschreiben, woher der Cluster bzw. Maschinen, die Zufallszahlen für die UUID holen soll, welche bei oVirt zur Generierung der Verzeichnisnamen dienen. Bei einem so kleinen Cluster kann man sich auf die Software selbst stützen und muss nicht /dev/random bemühen.

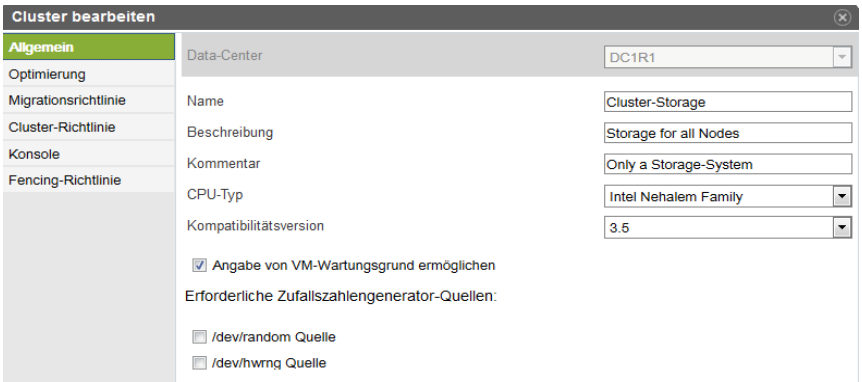



*Abbildung 39: Realisierung des Virtualisierungssegmentes: Erstellen des Cluster-Storage; Allgemein*

**Optimierung:**

- **Arbeitsspeicheroptimierung**: Hier kann angegeben werden, ob der reale Arbeitsspeicher überschritten werden darf und ob eine Optimierung, basierend auf einer Server- oder Desktop – Einstellung, genutzt werden soll. Da dies dem Autor aufgrund der stärkeren Wärmeentwicklung zu riskant ist, wurde hier keine Überschreitung gewählt.
- **CPU-Threds**: Definiert, ob die reale Kernanzahl, oder ob zur Maximierung der möglichen VM – Anzahl das reguläre Model Kernanzahl x 2 genutzt werden soll. Hier wird, um eine höhere VM Zahl zu generieren, das reguläre Modell gewählt.
- **Memory-Balloon-Optimierung aktivieren**: Hierbei versucht der Hypervisor, das Ballooning des dynamisch variierenden Arbeitsspeichers zu optimieren, um das Maximum herauszuholen. Auf diese Oprimierung kann bei einem Cluster dieser Grösse im privaten Umfeld verzichtet werden.
- **KSM – Steuerung**: Hierbei geht es um die Optimierung der NUMA (Non-Uniform Memory Access), welche die RAM – Bausteine, die zu einem Kern gehören, so über den Hypervisor fixieren kann, dass eine VM bspw. sich nur in dieser Bausteingruppe aufhält. oVirt bietet über die KSM Funktion die Möglichkeit, dies sauber und zentral zu steuern und stets auf eine korrekte Verteilung der VM's zu achten. Diese Funktion und die spätere Nutzung der NUMA Fuktion wird seitens des Autors strickt abgelehnt, da sie das Live-Migrieren der VM's verunmöglicht.

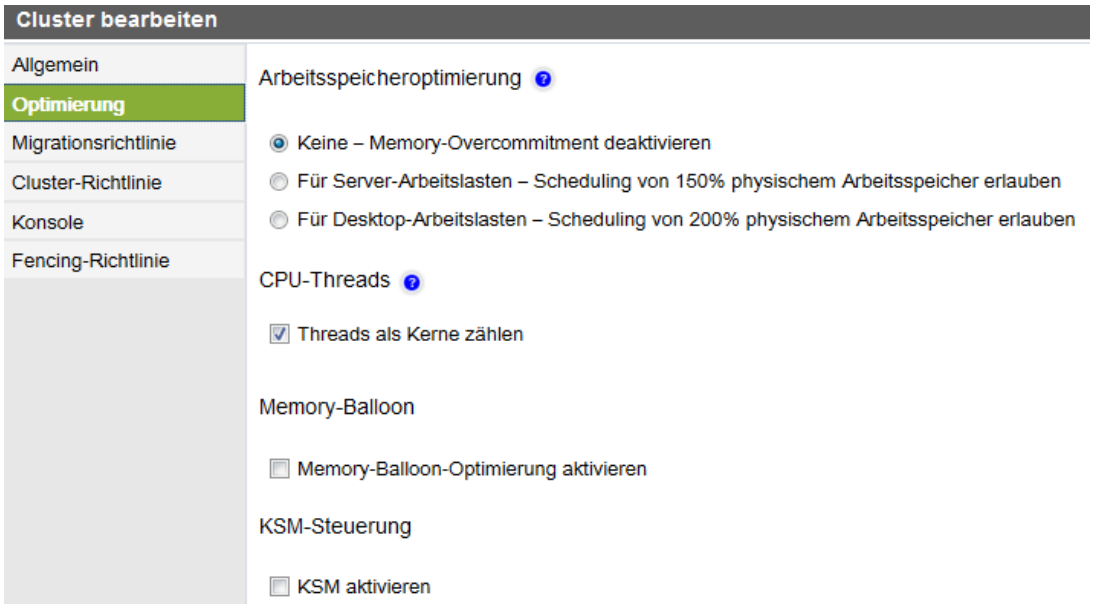



*Abbildung 40: Realisierung des Virtualisierungssegmentes: Erstellen des Cluster-Storage; Optimierung*

**Migrationsrichtlinien:**

- Hier kann angegeben werden, ob Live – Migration für den Cluster möglich sein soll oder nicht. Dabei kann noch einmal granuliert werden, ob bei Live – Migration dies allgemein möglich sein soll oder nur für hoch verfügbare VM's gelten soll. Da der Referenzcluster Cluster-Storage hier weder die Leistung (CPU) noch den Arbeitsspeicher hierfür hat, ist er explizit gesperrt für diese Funktion. Cluster-Level-high und Cluster-Level-Middle dürfen und sollen VM's in jedem Fall und ohne HA Einschränkung migrieren dürfen.

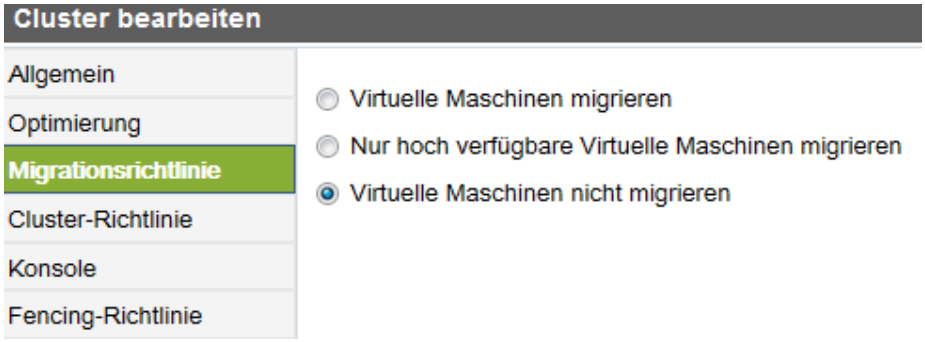



*Abbildung 41: Realisierung des Virtualisierungssegmentes: Erstellen des Cluster-Storage; Migrationsrichtlinien*

**Cluster – Richtlinien:**

- **Eigenschaften:** Hier gibt es einige mögliche Einstellungen. Die meisten beziehen sich auf das Verhalten bei aktivem HA und Ressourcenzuteilungen. Zum Beispiel kann hier definiert werden wie viele VM's laufen dürfen, bis die Anzahl als zu hoch eingestuft wird und eine Verteilung stattfinden muss. Alle Optionen an dieser Stelle durchzugehen, würde den Rahmen dieses Kapitels sprengen. Hier wird nichts konfiguriert und die systemeigenen Default Werte bleiben wie sie sind. Es würde ohnehin keinen Sinn machen hier Parameter zu übergeben, da hierfür das Power – Management über LOM aktiv sein müsste.
- **Optimierung für Auslastung:** Definiert, ob der Cluster für Auslastung, also eine hohe Anzahl an VM's, oder für eine hohe Geschwindigkeit, also eine kleinere Anzahl an VM's, konfiguriert werden soll. Hier soll so viel wie möglich an VM's herausgeholt werden, weswegen **Optimierung für Auslastung** gewählt wird.
- **HA-Reservierung aktivieren:** Diese Option aktiviert die hoch Verfügbarkeit des Clusters, damit die VM's eines ausgefallenen Nodes von einem anderen übernommen werden. Diese Option ist beim Cluster-Storage deaktiviert, da er dies nicht können muss. Bei Cluster-Level-high und Cluster-Level-Middle nützt dies ohne Power – Management zwar nichts (Semi-Automatic-HA), es wird aber präventiv aktiv gelassen, da die Möglichkeit einer späteren Realisierung noch offen bleibt.
- Die Option **Vertrauenswürdigen Dienst aktivieren** konnte nicht exakt geklärt werden, weswegen an dieser Stelle auf eine Definition verzichtet wird.
- **Benutzerdefinierte Seriennummern-Richtlinie angeben:** Hier kann die Vergabe der

Seriennummer an die VM's definiert werden. Diese Option ist für grosse Hoster sicherlich von Bedeutung, jedoch wird hier darauf verzichtet und die Vergabe oVirt überlassen.

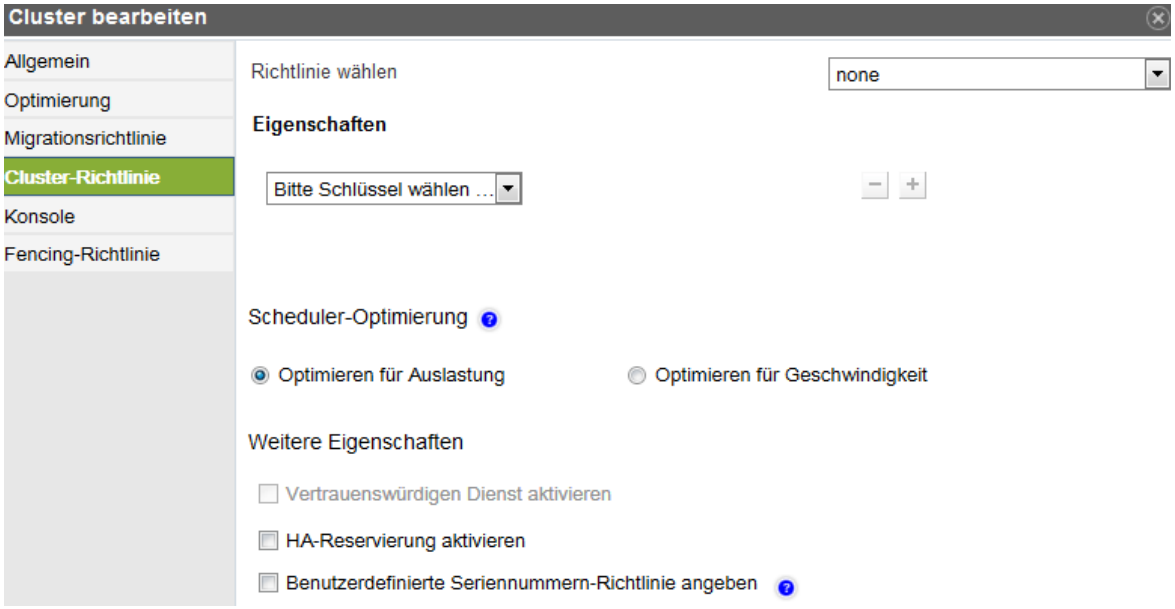



*Abbildung 42: Realisierung des Virtualisierungssegmentes: Erstellen des Cluster-Storage; Cluster-Richlinien*

**Konsole:**

- oVirt unterstützt zwei arten von Video – Konsolen, einmal das heute als veraltet geltende VNC und das von RedHat entwickelte und weit modernere SPICE Protokoll. SPICE bietet heute die Möglichkeit von extrem hoher Video Auflösung und der Weiterleitung von Audio und USB an den Client. Standardmässig sollte SPICE heute in jedem Fall verwendet werden. An dieser Stelle kann ein SPICE – Proxy vergeben werden, falls man aus Sicherheitsgründen keine direkte Verbindung zu Hypervisor wünscht. Aufgrund der Grösse des Clusters und der Tatsache, dass der Autor auch die Kontrolle darüber hat, wer Zugang hat, kann hier auf einen Proxy verzichtet werden.


*Abbildung 43: Realisierung des Virtualisierungssegmentes: Erstellen des Cluster-Storage; Konsole*

**Fencing aktivieren:**

- Hier wird das Verhalten von HA definiert, welches abhängig vom Power – Management die Initialisierung des Neustarts der VM's auf einem anderen Cluster – Node in die Wege leitet. Hier kann **Fencing aktivieren** aktiv gelassen werden, sodass oVirt die Nodes ständig auf Verfügbarkeit prüft und im Falle eines Node Ausfalles versucht, ihn über das LOM Interface neu zu starten oder herunterzufahren. Dabei kann hier angegeben werden, ob das Fencing übersprungen wird, wenn Verbindungsprobleme (ausgefallene Hosts) in den Prozentual – Bereichen 25, 50, 75 und merkwürdigerweise 100% liegen. Die zweite Option „Fencing überspringen, wenn der Host aktive Lease zum Speicher hat " konnte nicht exakt geklärt werden. Weder die Aussagen auf deutsch noch auf englisch ergeben Sinn und es konnte keine saubere Dokumentation im Internet gefunden werden. Lediglich ein Bugreport brachte etwas Weniges an Erkenntnis, woraus die **Schlussfolgerung** gezogen wurde, dass das Fencing abgelehnt wird, wenn der Host die Speicherdomäne als Gast (external) nutzt. Jedoch wurde diese Einstellungsmöglichkeit an Storage – Anbindung nie so in dieser Form unter oVirt 3.5 gefunden. Das Fencing wurde für die Cluster Cluster-Level-high und Cluster-Level-Middle aktiviert, auch wenn diese Möglichkeit momentan aus technischen Gründen gar nicht möglich ist. Sie stört den regulären Betrieb in keinster Weise, ausser, dass sie eine Unmenge an Warnungen produziert.

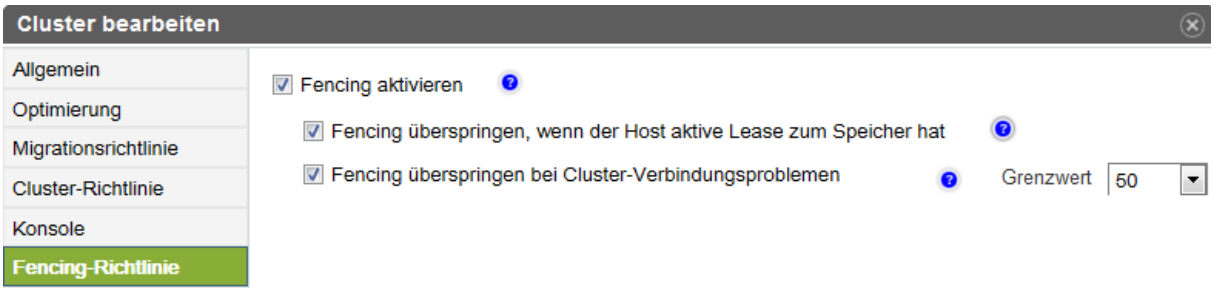



*Abbildung 44: Realisierung des Virtualisierungssegmentes: Erstellen des Cluster-Storage; Fencing-Richtlinien*

### 8.7.3 Kurzer Exkurs zu SSH Soft Fencing

Technisch ist es möglich, bei aktivem Kernel Dumps (kdump) die light Version von Fencing zu nutzen. Diese ist rein softwarebasiert und benötigt kein Power – Management in Kombination mit einem LOM – Interface, da sie den Host per SSH direkt anspricht. Dabei wird versucht, den VDSM Daemon wieder zu starten. Jedoch übernimmt diese Aufgabe heute der Watchdog Daemon, was diese Lösung im Allgemeinen immer weiter in die Vergessenheit drängt. Zusätzlich muss man sich die Frage stellen, was dies denn überhaupt für einen Sinn hat, wenn der Host selbst nicht mehr ansprechbar ist. Er kann ja in eine Kernel Panic fallen und somit einfach hängen bleiben, dann macht Zugriff per SSH wenig Sinn. Aus diesem Grund wurde auf diese Funktion verzichtet, um die unnötige Belastung durch kdump zu vermeiden.

### 8.7.4 Einbinden / Registrieren der einzelnen Nodes

Nun haben wir im vorhergehenden Schritt das für diesen physischen Cluster notwendige Data – Center angelegt und die logische Aufteilung nach Vorstudie an internen Clustern erzeugt. Jetzt wird es Zeit alle Nodes in die dafür vorgesehenen Cluster zu verteilen. Hierbei werden zwei von den drei möglichen Einbindevarianten aufgezeigt, welche sich bei den beiden eingesetzten Installationsvarianten ergeben. In diesem ersten Schritt wird auf die Netzwerkkonfiguration (Bonding) vorläufig verzichtet, da es aufgrund des stabilen Stacks von oVirt immer und zu jeder beliebigen Zeit möglich ist.

#### 8.7.4.1 Einbindeprozess anhand der CentOS Minimalinstallation

Wie bereits erwähnt, bietet die VDSM – Abstraktionsschicht die Möglichkeit der konsolenseitigen Einbindung. Diese ist jedoch mit langen Kommandoeingaben verbunden und eindeutig nicht zu empfehlen. Hier ist die Einbindung aus dem oVirt Center der Einfachste und zugleich sicherste Weg. Dabei wird hier der Host gfsn1.mgmtdom (Storage Node 1) als Referenz konfiguriert, wobei sich die Konfiguration auch auf den zweiten Storage Node übertragen lässt. Hierfür klickt man wie folgt:

```
Hosts (Registerkarte) → Neu (innerhalb des neuen Fensters)
```

*Befehl 23: Realisierung des Virtualisierungssegmentes (Klicken): oVirt - Engine; Hosts einbinden mit Assistenten*

Hierbei öffnet sich ein kleines Konfigurationsfenster, welches sich in vier Konfigurationskategorien aufteilt. Auch hier werden wir die Kategorien Schritt für Schritt durchgehen, beginnend mit der bei oVirt stets ersten und wichtigsten:

**Allgemein:**

- **Data-Center**: Hier wird das Data – Center ausgewählt, in unserem Fall gibt es ja nur eins.
- **Host-Cluster**: Hier kann der Host einem internen Cluster zugewiesen werden. Da es sich um einen Storage Node handelt, soll hier auch der Cluster **Cluster-Storage** aus der Auswahlliste gewählt werden.
- **Name**: Hier kann dem neuen Host ein oVirt- interner Name zugewiesen werden, jedoch empfiehlt es sich hier nicht vom eigentlichen Hostname abzuweichen, da bei einer grossen Host Anzahl ein kaum durchschaubares Chaos entstehen könnte. Der Name in dieser Referenzkonfiguration ist gleich dem eigentlichen Hostnamen, also **gfsn1.mgmtdom**.
- **Kommentar**: Bezüglich der Dokumentation der Arbeit an dieser Stelle, ist hier mittlerweile ein Kommentar mehr notwendig.
- **SSH – Port**: Innerhalb dieses Projektes und der starken Isolation des Cluster durch die Firewall, sind Abweichungen der Port nicht notwendig. Aus Diesem Grund kann hier der vorgeschlagene Default Port 22 auch so belassen werden.
- **Authentifizierung/ Benutzername**: Dieser wird seitens oVirt per Default auf **root** gesetzt und kann auch so belassen werden.
- **Authentifizierung/ Passwort**: Dieser Punkt spricht für sich selbst.

- **Authentifizierung/ Öffentlicher SSH-Schlüssel**: Diese Option erlaubt anstelle eines Passwortes die Authentifizierung mittels eines Public – Keys, welcher auf dem Host hinterlegt werden muss. Dieser Key wurde bereits erzeugt und ist ersichtlich beim Auswählen der Option. Er müsste von der Liste kopiert und dem Host unter **/root/.ssh/authorized_keys** manuell hinzugefügt werden. Auf diesen Mehraufwand kann aber an dieser Stelle verzichtet werden, da dies ohnehin automatisch beim Einbinden des Hosts im Hintergrund geschieht und zugleich die Standard- Kommunikationsart von oVirt ist.
- **Erweiterte Parameter**: Hier können zwei Parameter aktiviert bzw. deaktiviert werden.
  - **Automatisches Konfigurieren der Host-Firewall**; aktiviert den CentOS eigenen firewall Daemon und konfiguriert ihn zugleich. Diese Konfiguration wurde so per Default aktiv gelassen und später wieder manuell auf den Host deaktiviert, da sich hier ein Konfigurationsfehler seitens oVirt in Bezug auf die GlusterFS Ports zeigte. Auf manuelle Eingriffe in die Host – Configs wurde verzichtet, da aufgrund der restriktiven Regelsätze seitens des Netzwerksegmentes diese Firewall ohnehin überflüssig wird und nur Ressourcen verschwenden würde.
  - **JSON-Protokoll verwenden**; wurde einmal mit und einmal ohne getestet. Es konnte weder ein Vor- noch Nachteil erkannt noch eine klar definierende Dokumentation gefunden werden. Hier wird einfach der Defaultwert (aktiv) belassen.

**Energieverwaltung:**

- Dieser Punkt stellt das Power – Management dar und wird hier nicht konfiguriert, da ein Host dies nicht unterstützt und somit die Konfiguration sinnlos wäre. Hier kommt das Konzept Semi-Automatic-HA zum Zuge.

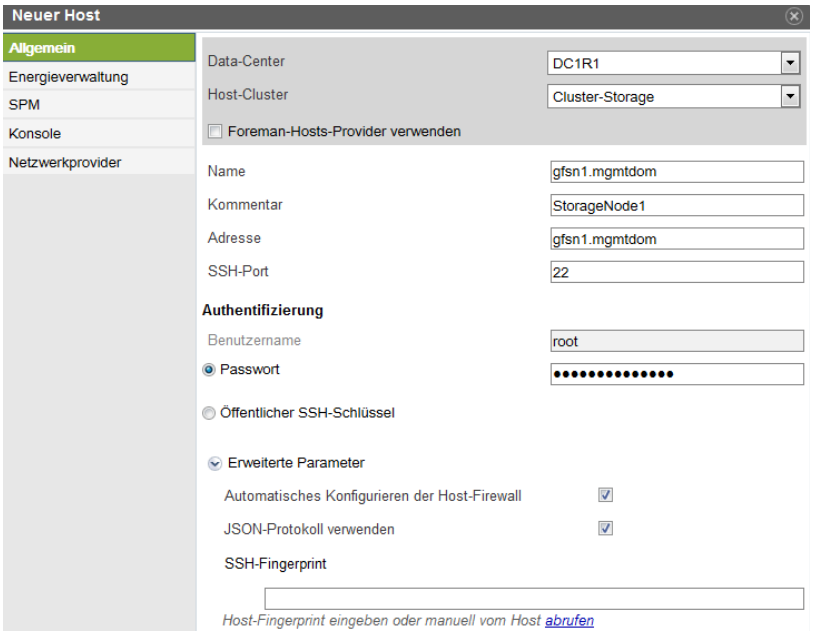



*Abbildung 45: Realisierung des Virtualisierungssegmentes: Assistent zu einbinden der Nodes; Allgemein*

**SPM:**

- Die Option **Storage Pool Manager** definiert, welcher Node für das Sharen der GlusterFS und NFS Informationen innerhalb des **Data – Center** zuständig ist. Zwar wird die Konfiguration im oVirt Center erledigt, aber das eigentliche Verteilen der Informationen übernehmen die Nodes untereinander selbst. Der Grund ist das Aufrechterhalten der Shares beim Ausfall der oVirt – Engine. Hier muss angegeben werden, welcher Node welche Priorität besitzt, wobei der erste ausgewählte Node, belanglos, welche Priorität ihm zugewiesen wird, die Rolle des Masters übernimmt. Wurden aber zu Anfang Prioritäten tiefer als **Hoch** vergeben, so wird der erste Host, dem die Priorität **Hoch** verliehen wird zum neuen Master. An dieser Stelle macht es Sinn die Storage Nodes mit der Priorität **Hoch** zu klassifizieren, da sie nicht virtualisieren und primär für die Storages zuständig sind. Storage Node 1 wurde als erster mit **Hoch** klassifiziert und ist somit der Master, was sich später in in der Host – Liste daran erkennen lässt, dass alle Hosts bis auf den Master die Priorisierungs- Angabe aufgeführt haben, der Master aber trägt das Siegel SMP. Sollte der Master ausfallen, so übernimmt der nächste Host mit der höchsten Priorisierung. Fällt dieser auch aus, so übernimmt der nächste mit der entsprechenden Priorisierung, usw.

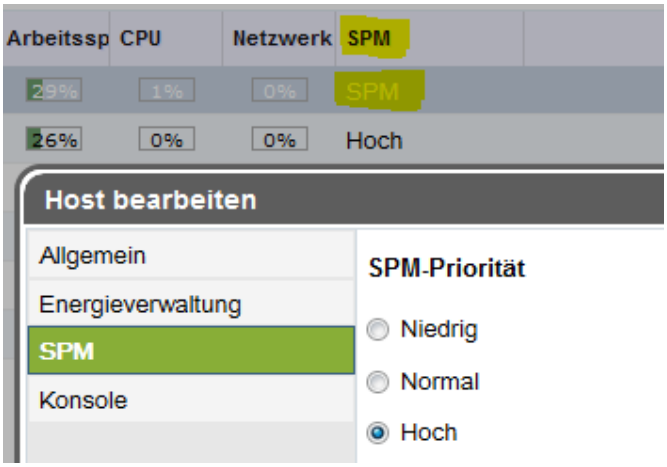



*Abbildung 46: Realisierung des Virtualisierungssegmentes: Assistent zu einbinden der Nodes; SPM, Master ist GlusterFS Node 1, der zweite sichtbare Eintrag ist Node 2*

**Konsole:**

- Hier kann auf Host Ebenen die SPICE – Proxy Weiterleitung spezifiziert werden. Auch an diesem Punkt kann die Konfiguration, ähnlich der Cluster – Konfiguration, deaktiviert belassen werden.


*Abbildung 47: Realisierung des Virtualisierungssegmentes: Assistent zu Einbinden der Nodes; Konsole, Proxy Weiterleitung für die Darstellungskonsole*

### 8.7.4.2 Einbindeprozess anhand eines oVirt – Node Images

Im vorhergehenden Beispiel haben wir gesehen, wie es mit einer CentOS Rohinstallation funktioniert. Technisch gesehen könnten wir dies auch mit den Node – Images machen, jedoch soll hier auch die Alternative kurz angeschnitten werden. Hierbei muss man über einen Monitor und eine Tastatur direkt Kontakt zum Node haben. Im Control – Menü des Nodes muss man auf die Registerkarte in der Seitenleiste rechts auf oVirt Engine gehen. Hier kann man nun zuoberstdie IP – Adresse bzw. den FQDN des Management Centers eingeben. Der Default Port 443 kann so belassen werden. Allfällige Zertifikatsfragen können im Allgemeinen ignoriert werden. Optional kann hier auch das Passwort für eine Art Zweitverbindung angegeben werden. Dieses Passwort wird zusätzlich zum bestehenden genutzt und soll SSH weiterhin zum Host per Konsole ermöglichen. Diese Angabe scheint aber ein Überbleibsel einer älteren Version zu sein, da trotz Angabe der Host so konfiguriert wird, dass ein Passwort – Login nicht möglich ist. Er benötigt trotzdem weiterhin eine Key – Authentifizierung, was aber nicht möglich ist, da man nicht zum Useraccount des Hosts gelangen kann, um ihn zu hinterlegen. Man kann sich an dieser Stelle die Eingabe eines Passwortes sparen und mit **Save & Register** bestätigen. Wechselt man nun in das oVirt Management Center, so wird der Host unter **System (Seitenleiste) → Hosts (Registerkarte)** als noch nicht alloziert aufgeführt. Mittels Rechtsklick (Kontextmenü) und „Hinzufügen“, gelangt man zum Konfigurationsassistenten, welcher vom Ablauf her identisch mit dem Vorgehen unter Punkt 8.7.3.1 ist. An dieser Stelle werden die restlichen Nodes äquivalent zum vorhergehenden Einbinden der Storage Nodes ins System registriert. Hierbei gelten aber folgende Abweichungen:

- Die Fujitsu Maschinen wurden unter **Allgemein** in den Cluster **Cluster-Level-Middle** integriert.
- Die beiden Power Maschinen wurden unter **Allgemein** in den **Cluster Cluster-Level-High** eingebunden.
- Alle Nodes wurden unter **SPM** mit der Priorität **Niedrig** eingebunden. Mehr Priorität ist an dieser Stelle für die Virtualisierungs- Nodes nicht notwendig.

Dieses Vorgehen wurde für die Nodes ovn1 – 4.mgmtdom angewendet und ist vom Konfigurationsablauf her gesehen bis auf die oben genannten drei Punkte identisch mit der Referenzkonfiguration unter Punkt 8.7.3.1 „Einbindeprozess anhand der CentOS Minimalinstallation“. Hier sollte nur gezeigt werden, dass es von den Nodes aus auch eine Anmelde – Alternative gibt. Zeitsparender ist mit Sicherheit die unter Punkt 8.7.3.1 genannte Methode, welche in jedem Fall zu bevorzugen ist.

### 8.7.5 Einbinden des GlusterFS Volumes als Master Storage

Nun, da alle Maschinen in den für sie vorgesehenen Clustern untergebracht sind, sollen wir noch den Master Storage einbinden, um überhaupt etwas speichern zu können. Wie wir uns erinnern, haben wir alles Notwendige bereits vorbereitet. So ist es auch hier möglich, die komplette Einbindung des vorbereiteten GlusterFS Volumes oVirtStorage1 aus der Engine aus zu vollziehen. Hierfür klicken wir folgenden Weg:

```
Speicher (Registerkarte) → Neue Domäne
```

*Befehl 24: Realisierung des Virtualisierungssegmentes (Klicken): oVirt - Engine; Einbinden des Master Storage*

Nun öffnet sich ein kleines Konfigurationsfenster, in welchem wir nachfolgende Parameter definieren müssen:

- **Name**: Hier kann ein unabhängiger und aussagekräftiger Storagename vergeben werden. In unserem Fall nennen wir in schlicht und einfach Storage-R1, woraus wir schliessen können, dass der Storage im Rack 1 beheimatet ist.
- **Data-Center**: Storages, welche in oVirt definiert werden, sind stets in einem Data – Center aktiv. Dies macht auch Sinn, da sich ja auch die einzelnen Cluster in einem Data – Center befinden müssen.
- **Beschreibung und Kommentar**: Diese Punkte sind selbsterklärend.
- **Dämonenfunktion/Speichertyp**: Hier kann die Art definiert werden, wie und mit welcher Anbindeart der Storage in die Domäne geholt werden soll. Für den Master Storage wählen wir **Data/GlusterFS**, da wir eine allgemeine Speicherquelle (Data) wollen, welche GlusterFS als Verfahren nutzt.
- **Format**: Dieser Punkt kann bei GlusterFS nicht geändert werden und muss V3 sein, da Gluster erst ab Major – Release 3.x einsetzbar ist.
- **Host verwenden**: Jeder Host, welcher eine Funktion im Cluster hat, muss auch mittels VDSM Daemon eingebunden werden. Eine neutrale Speicherquelle ohne VDSM ist nicht möglich. Dies sorgte für etwas Unmut bei anderen Distributionen, was RedHat dazu veranlasste, hier der restlichen open source Community etwas entgegenzukommen. So ist in Version 3.6 von oVirt nun auch eine Portierung von VDSM auf Debian vorhanden. An dieser Stelle geben wir nun einen der Gluster Storage Nodes an, welcher ist vollkommen egal. Der Ordnung halber benutzen wir hier den ersten, also **gfsn1.mgmtdom**.
- **Pfad**: Der Pfad zur Speicherquelle folgt der Syntax <NodeName>**:**<GlusterVolume> und lautet somit **gfsn1.mgmtdom:oVirtStorage1**.
- **VFS-Type** und Einhängeoptionen: Sind hier vorgegeben und man muss und kann nicht wählen.
- **Quorum**: Hier ist ein Infotext, welcher darauf hinweist, dass Quorum client- und serverseitig aktiv sein sollte. Wie wir uns erinnern wurde diese Option auch so beim Erstellen des Volumes

im Nachhinein definiert. Getestet wurde aber in einer virtualisierten Umgebung mit und ohne. Ein Unterschied konnte nicht festgestellt werden.

Aufgrund der Vorbereitungen im Vorfeld war dies auch schon der grösste Teil des Einbindeprozesses.

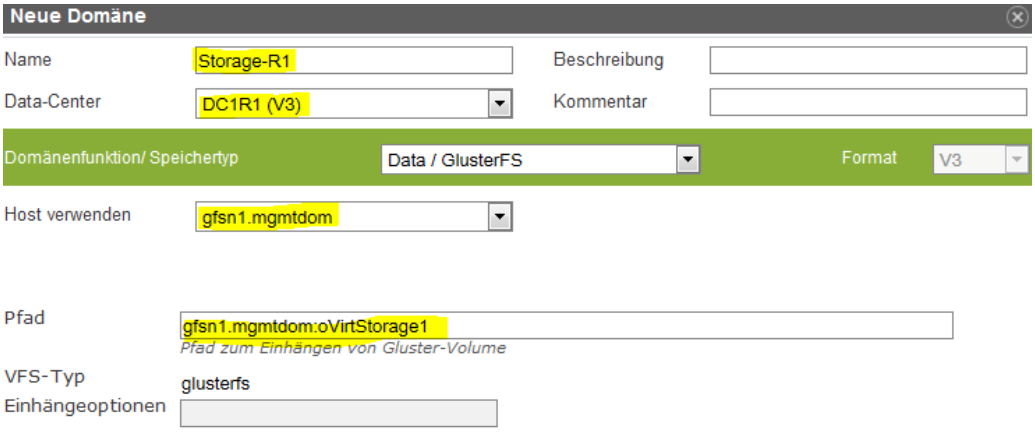



*Abbildung 48: Realisierung des Virtualisierungssegmentes: Einbinden von GlusterFS Storage mittels Assistenten*

Dies war die gesamte Prozedur. Die Storage Domäne wurde unseren angaben zufolge automatisch ins Data – Center eingebunden. Es ist auch möglich unter Data – Center „none“ anzugeben. Dies ist sinnvoll, wenn man mehrere Data – Centers hat und den Storage temporär plazieren möchte, um anschliessend das endgültige Ziel zu bestimmen. In diesem Fall kann einfach der Storage angeklickt werden und im unteren Bereich des Browsers öffnen sich die „Optionen“. Hier kann unter Data – Center → Zuordnen bestimmt werden, wem der Storage gehören soll.

### 8.7.6 Einbinden des ISO_Store's per NFS

Das Einbinden der ISO Domäne ist vom Ablauf her identisch mit dem vorhergehenden. Was sich hier explizit ändert, abgesehen vom Namen und den Kommentaren, ist die Domänenfunktion/Speichertyp. Hier muss **ISO/NFS** angegeben werden. Ob man hier bei NFS den ersten oder den zweiten Gluster Storage Node wählt ist ebenfalls belanglos. Hier ist ein Punkt jedoch von grösster Wichtigkeit, welcher unter **Erweiterte Optionen** definiert werden muss. Wie wir uns erinnern, haben wir beim Bilden der Gluster Volumes (beide) TCP als primäres Übertragungsprotokoll gewählt. Nun ist aber der zentrale Einbindemechanismus bei NFS auf UDP ausgelegt. Unter **Erweiterte Optionen** haben wir die Möglichkeit unter **Zusätzliche Einhängeoptionen** den Wert **mountproto=tcp** einzutragen. Somit kann der GlusterFS Server auf den Nodes als nativer NFS Server genutzt werden, es muss also kein zusätzliches nfs-server Paket nachinstalliert zu werden. Die Verteilung der Daten auf beide Nodes kann hier natürlich nicht über den Gluster Client gesteuert werden, da es sich ja um einen NFS Mountpoint handelt. Das ist aber absolut kein Problem, da sich der GlusterFS – Server auf dem Einbinde- Node gfsn1.mgmtdom selbst um das Verteilen der Daten kümmert. Somit entscheidet an dieser Stelle nicht der Client, sondern der Server ,was wohin soll und kopiert alles, was für gfsn2.mgmtdom bestimmt ist direkt nach Empfang zu ihm. Hier muss auch noch erwähnt werden, dass bei oVirt nur eine einzige ISO

Domäne definiert werden kann.

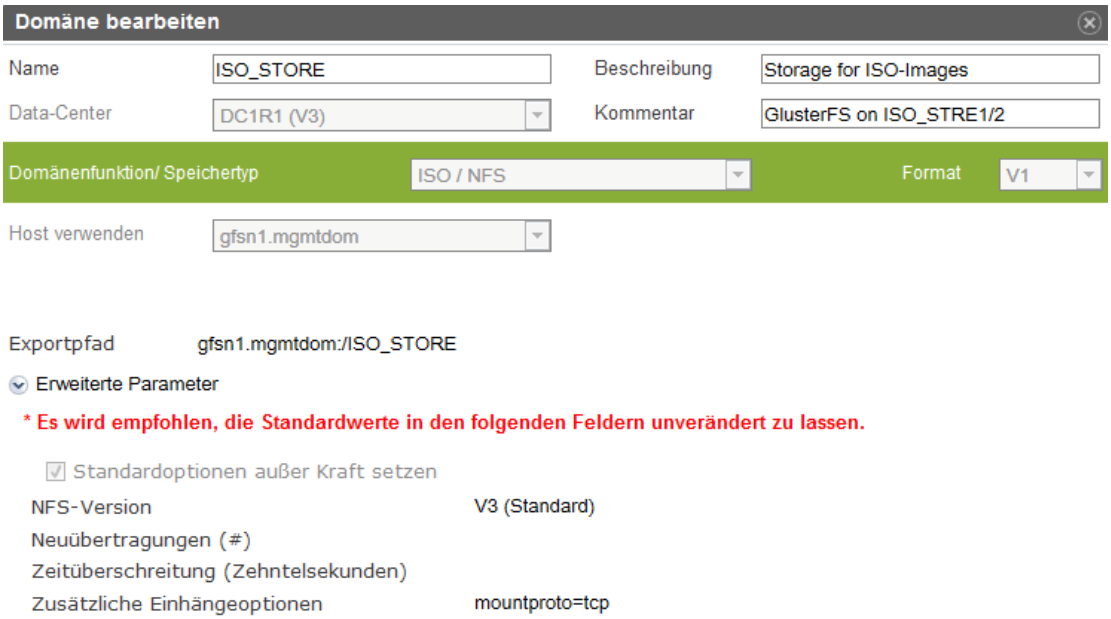



*Abbildung 49: Realisierung des Virtualisierungssegmentes: Einbinden von GlusterFS Storage als ISO Domäne per NFS (hier die Fertige Konfiguration)*

Dies war es im Allgemeinen mit der Storage Einbindung. Nun könnte unser Virtualisierungs- Cluster sogar betrieben werden, da ja jetzt auch Speichermöglichkeiten vorhanden wären.

Bezüglich des Kopierens der ISO's auf den ISO_STORE gibt es eine Menge an Möglichkeiten. Der Autor nutzt hier den GlusterFS Node selbst für diese Prozedur. Hierfür kann man mit nachfolgendem Befehl auf gfsn1.mgmtdom arbeiten.

```
# mount.nfs -o mountproto=tcp gfsn1.mgmtdom:/ISO_STORE /mnt
# cd /mnt/8f62618b-7910-4987-a6a2-052b8b083bfa/ 11111111-1111-1111-1111-111111111111
```

*Befehl 25: Realisierung des Virtualisierungssegmentes (Befehl): oVirt - Engine; Einbinden des Master Storage*

Mit diesem Befehl bindet man man den NFS Mountpoint lokal ins **/mnt**, wobei bei anderen Installationen natürlich die **UUID** abweichen wird. Was aber bei allen gleich sein wir, ist das Verzeichnis **11111111-1111-1111-1111-111111111111**. Dieses ist allgemeingültig und steht oVirt- Intern für ISO Domäne. Nun kann das gewünschte ISO auf der Konsole mittels wget heruntergeladen werden und der Mountpoint mittels **umount** wieder gelöst werden. Diese Methode wirkt umständlich, ist aber eine beliebte Praktik des Autors.

### 8.7.7 Erstellen der Bondings für das Management – Netzwerk

Wie bereits erwähnt, könnten wir an dieser Stelle bereits los virtualisieren. Um aber etwas mehr Performance herauszuholen, werden wir uns in diesem Abschnitt mit dem Bonding der Netzwerkschnittstellen des Managementteils beschäftigen. Prinzipiell gilt bei den meisten Virtualisierungs- Umgebungen der Grundsatz, zuerst das Netzwerk definieren und dann entsprechend den Hosts zuteilen. Aus diesem Grund gehen wir in folgendes Menü unter oVirt:

```
Netzwerk (Seitenleiste) →
```

*Befehl 26: Realisierung des Virtualisierungssegmentes (Klicken): oVirt - Engine; Einbinden des Master Storage*

Auf dieser Seite treffen wir das erste Mal keine leere Konfiguration an. Hier ist das ovirtmgmt Netzwerk bereits per Default von VDSM eingerichtet worden. Die Einstellungen innerhalb dieses Netzwerkes sind simpel, sie beinhalten nur das MV-Netzwerk, welches standardmässig aktiv ist. Diese Einstellung belassen wir auch so und gehen zum ersten Host. An dieser Stelle werden wir stets oVirt Node 4 (ovn4.mgmtdom) als Referenzkonfiguration arbeiten. Sämtliche hier getätigten Einstellungen sind entweder eins zu eins übertragbar oder es wird spezifisch erwähnt, wo Differenzen zu den anderen Nodes vorhanden sind. Die Interface – Bezeichnungen variieren natürlich von Host zu Host, jedoch wird hier nicht spezifisch darauf eingegangen.

Bevor wir aber zu oVirt Node 4 gehen, bleiben wir noch auf der Seite für die Netzwerkkonfigurationen, wo wir etwas vorgreifen können und gleich das zweite benötigte Netzwerk anlegen. Auf dieser Seite wählen wir **Neu**, woraufhin ein Konfigurationsfenster aufgeht. Das Design ist ähnlich, wie wir es schon kennengelernt haben. Dieses Fenster unterteilt sich in drei Kategorien:

**Allgemein:**

- **Data - Center**: Hier kann wieder nur das eine bestehende ausgewählt werden.
- **Name**: Ein aussagekräftiger Name für das Netzwerk, hier VMnet1
- **Kommentar**: kommentarlos
- **Exportieren**: Mit dieser Option kann ein Netzwerk zu einem fremden Provider (bspw. Data – Center) angegeben werden. Es ist nützlich falls man ein Transitnetzwerk benötigt für eine Migration von bestehenden VM's zu einem Provider bzw. eine Migration in Planung hat. So können die VM's noch im aktuellen Data – Center betrieben werden, nutzen aber die Netzwerkkonfiguration eines anderen Data – Centers. Diese Option benötigen wir nicht.
- **Netwerk-Parameter**:
  - **Netzwerkbezeichnung**: Kann an dieser Stelle nicht geändert werden, jedoch später an einer anderen Stelle. Diese Option ermöglicht die abweichende Bezeichnung des Netzwerkes zu Standard.
  - **VLAN-Tagging aktivieren**: Hier kann einem Netzwerk eine VLAN Tag mitgegeben werden (NUR eine Nummer), falls man auf diese Weise separieren möchte. Diese Option wird hier

nicht benötigt.

- **VM-Netzwerk**: Ist per Default immer aktiv und soll es auch sein.
- **MTU**: Hier empfiehlt es sich, im Allgemeinen nichts zu ändern falls man mit VPN arbeitet, da bei Tunneln standardmässig Don't Fragment aktiv ist.

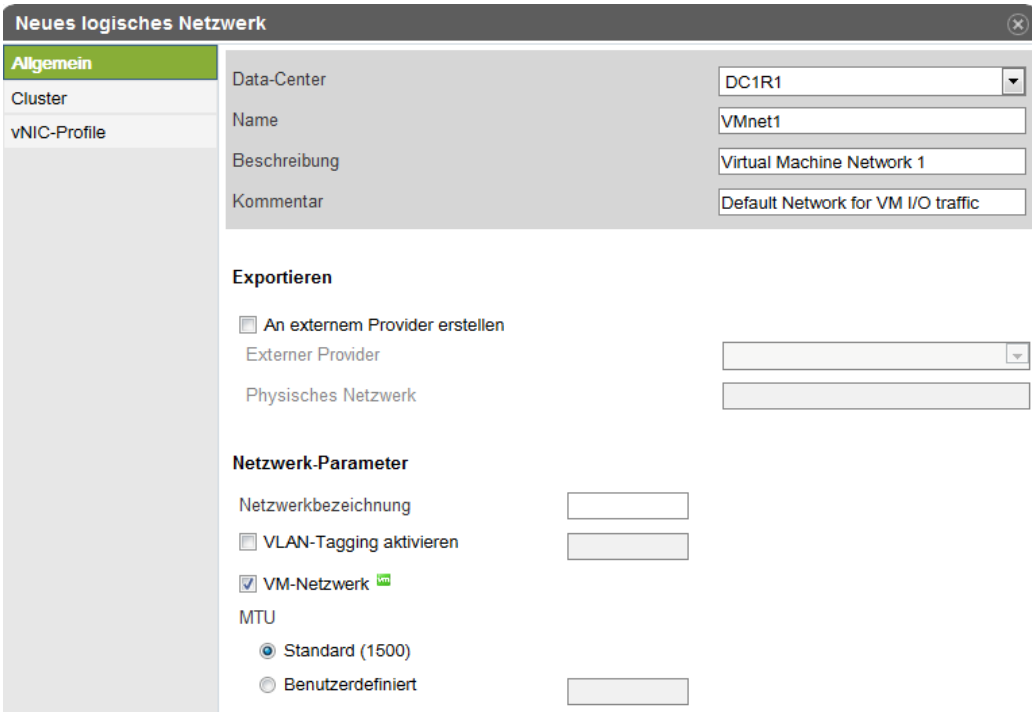



*Abbildung 50: Realisierung des Virtualisierungssegmentes: Erstellen eines neuen Netzwerkes über den Assistenten; Allgemein.*

### Cluster:

Hier kann angegeben werden, welche Cluster dieses Netzwerk nutzen dürfen. Dabei wird differenziert zwischen **Zuordnen**, also soll das Netzwerk unter den Nodes sichtbar sein, und **Erforderlich**, wobei hier die Nodes in den angegeben Cluster das Netzwerk als zwingend notwendig einstufen und es nicht mehr aus der Auswahlliste entfernbar ist.


*Abbildung 51: Realisierung des Virtualisierungssegmentes: Realisierung des Virtualisierungssegmentes: Erstellen eines neuen Netzwerkes über den Assistenten; Cluster wählen.*

**VNIC-Profile:**

Hier können bereits erste Limitierungen bezüglich Durchsatzbegrenzung (QoS) angewendet werden. Doch hierzu später mehr im nachfolgenden Kapitel.

Wenn wir alles richtig gemacht haben, sollte in der Netzwerkansicht folgendes Bild anzutreffen sein.

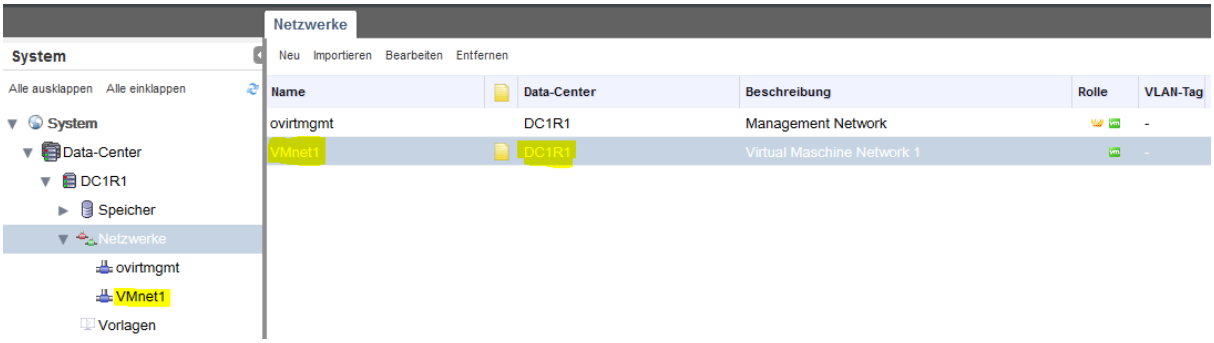



*Abbildung 52: Realisierung des Virtualisierungssegmentes: Erstellen eines neuen Netzwerkes über den Assistenten; Gesamtübersicht über alle Netzwerke.*

Nach diesem kurzen Vorgreifen kehren wir wieder zur eigentlichen Bonding Konfiguration des Management Netzwerkes zurück. Wir gehen dazu auf die Konfigurationsseite des Virtualisierungs-Node 4:

Auf Seitenleiste → Cluster (Ausrollen) → Cluster-Level-high (Ausrollen) → Hosts (Ausrollen) → ovn4.mgmtdom → Hosts (Registerkarte) → Netzwerkschnittstellen (untere Registerkarte)

*Befehl 27: Realisierung des Virtualisierungssegmentes (Klicken): Zum Konfigurationsmenü des Netzwerks von ovn4.mgmtdom*

Hier sehen wir eine Liste mit sämtlichen verfügbaren Netzwerkschnittstellen. Bei diesem Host wird glücklicherweise die verkürzte Schreibform der Namen angewendet. Nicht in einen running – Status versetzte Schnittstellen sind an dieser Stelle als rotes, nach unten blickendes Dreieck ersichtlich. Dies macht die Identifikation der Schnittstellen schwierig. Wenn alle Netzwerkkarten vom gleichen Typ sind, wird die Angelegenheit noch schwieriger. Glücklicherweise sind bei dieser Maschine die Onboard Schnittstellen und die PCI Karte von unterschiedlichen Herstellern, was die Konfiguration hier etwas einfacher gestaltet. Da das Default Netzwerk **ovirtmgmt** bereits einer Schnittstelle zugewiesen ist (p6p1), lässt sich ohne langes Herumprobieren feststellen, dass p6p2 der nächste aktive Nachbar sein muss, da ja nur diese zwei an der PCI Karte angeschlossen sind. Hat man seine Ports identifiziert, so kann man in dieser Subsektion den oberhalb liegenden Button **Hostnetzwerk einrichten** wählen. Siehe dazu nachfolgende Abbildung:


*Abbildung 53: Realisierung des Virtualisierungssegmentes: Host-Netzwerk mit Bonding erstellen Teil 1 (ovirtmgmt)*

Anschliessend öffnet sich ein Konfigurationsfenster, in welchem wir bequem per Drag and Drop arbeiten können. Hier sehen wir sämtliche Interfaces in eigenständigen Kästchen, wo wir einfach unser **p6p2** per Maus packen und auf **p6p1 ziehen** können. Bei dieser Aktion erkennt der Assistent gleich, dass wir irgend eine Art von Aggregation wünschen und öffnet uns ein kleines Fenster in welchem die genauen Parameter definiert werden können.

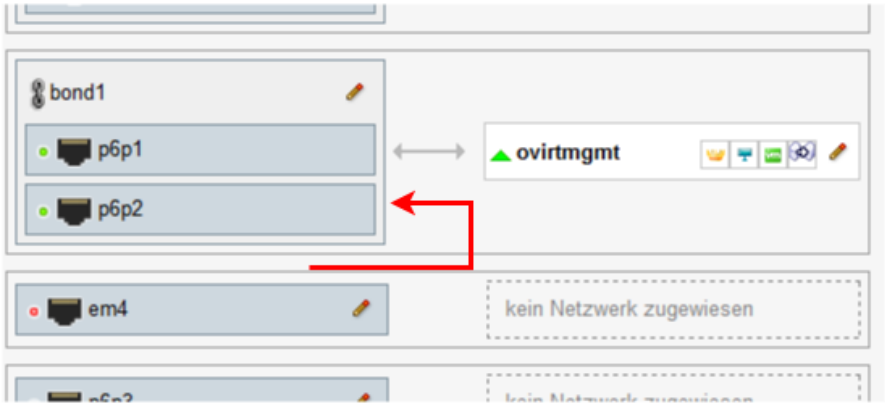



*Abbildung 54: Realisierung des Virtualisierungssegmentes: Host-Netzwerk mit Bonding erstellen Teil 2 (ovirtmgmt)*

Im Konfigurationsfenster geben wir einen Namen, hier Bond1, ein und wählen einen softwareseitigen Bond-Modus aus. Wir wählen hier aus der Auswahlliste den **Mode 2** aus (balance-xor), da dieser lastverteilend und ausfallsicher ist. Zugleich verteilt er die Verbindungen nach einem IP- Hash basierten Algorithmus auf die einzelnen Gegenstellen, wobei diese kontinuierlich für einen gewissen Zeitraum auch fixiert werden. Auf eine Bezeichnung kann an dieser Stelle verzichtet werden, da der eigentliche Schnittstellenname genug aussagekräftig ist. Sind die beiden Hauptparameter gesetzt, kann mit **OK** bestätigt werden und man landet auf der neu erstellten Übersicht.

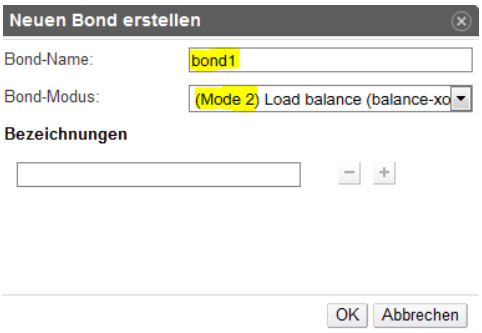



*Abbildung 55: Realisierung des Virtualisierungssegmentes: Host-Netzwerk mit Bonding erstellen Teil 3*

Ist der optische Eindruck auf der Übersichtsseite in Ordnung, kann man auch hier mit **OK** bestätigen. Ist man sich bei einer Konfiguration aber nicht sicher, besteht noch die Möglichkeit, das Optionskästchen im unteren Bereich des Konfigurationsmenüs, **Netzwerkkonfiguration speichern**, auszuschalten. In diesem Fall wird die Konfiguration zwar angewendet, jedoch nicht gespeichert. Sollte irgendetwas schiefgehen, oder die Konfiguration schneidet einen vom Host ab, so genügt nur ein Neustart des Hosts und alles ist wie vor der Konfiguration. Das Optionskästchen namens **Verbindung zwischen Host und Engine überprüfen,** sollte in jedem Fall bei jeder Konfiguration aktiv gelassen werden, auch wenn man sich zu 100% sicher ist mit dem, was man tut. Hier wird die Engine die gesamte Logik der Konfiguration prüfen und bei schwerwiegenden Fehlern, welche die Verbindung zu Host trennen können, unterbinden.

Diese Konfiguration muss nun auf jedem Host (Virtualisierung und Storage) angewendet werden. Sie ist bei wenig Aufwand schnell getan und lohnt sich in jedem Fall, da man mit Bond – Mode 2 einen höheren Datendurchsatz in Kombination mit Ausfallsicherheit erhält.

### 8.7.8 Erstellung des Netzwerkes für die VM's

An dieser Stelle wollen wir das Netzwerk für die virtuellen Maschinen einbinden. Wie wir uns erinnern haben wir im vorhergehenden Schritt bereits das VMnet1 erstellt. Hier müssen wir nur noch das Netzwerk den Hosts bekannt geben und die Nutzung erlauben. Wir werden an dieser Stelle den beiden Power Maschinen des Clusters **Cluster-Level-High** je drei Links und den beiden Fujitsu Maschinen je einen Link entreissen. Hierfür nutzen wir das gleiche Vorgehen wie im letzten Abschnitt und gehen nach der Beschreibung von Befehl 20 zur Host internen Netzwerkkonfiguration. Hier soll ovn4.mgmtdom wieder als Referenzbeispiel für seinen Partner ovn1.mgmtdom dienen. Auch hier orientieren wir uns am Beispiel von Abbildung 54 und dropen uns die drei Links zusammen. An dieser Stelle wählen wir aber Mode 4 (Dynamic Link Aggregation) als Bonding Methode aus, da wir die ja so unter pfSense vorkonfiguriert haben. Als Namen verwenden wir hier **Bond0.** Dies hat keine spezielle Bedeutung, sondern ist ein Standard Vorgehen des Autors bezüglich Public – Links bzw. DMZ Betitelung.


*Abbildung 56: Realisierung des Virtualisierungssegmentes: Host-Netzwerk mit Bonding erstellen Teil 1 (VMnet1)*

Nach Erstellung der Link Aggregation landet man wieder in der Gesamtübersicht, wo nun der neue Link – Verbund **bond0** sichtbar ist. Dieser nützt uns im Moment nicht viel, da ihm noch kein Netzwerk zugewiesen ist. Auf der rechten Seite des Konfigurationsmenüs sieht man aber unter der Kategorie **Erforderlich** unser erstelltes Netzwerk VMnet1. Dieses kann nun per **Drag and Drop** der Aggregation zugewiesen werden und das war es schon. Mit einem Klick auf **OK** wird die Einstellung übernommen und das Netzwerk geht über die Aggregation online.


*Abbildung 57: Realisierung des Virtualisierungssegmentes: Host-Netzwerk mit Bonding erstellen Teil 2 (VMnet1)*

Für die beiden Fujitsu Maschinen entfällt das Bilden einer Aggregation, hier kann referenziert auf Abbildung 57 einfach das Netzwerk auf seinen Platz geschoben werden.

### 8.7.8.1 Kurzer Exkurs auf diverse Erklärungen

#### 8.7.8.1.1 Nicht erforderliche Netzwerke

Wie wir in Abbildung 57 erkennen können, wird zwischen **Erforderlichen** und **Nicht erforderlichen** Netzwerken differenziert. Bei der Erstellung von VMnet1 haben wir den Default Wert auf **Erforderlich** belassen und somit oVirt mitgeteilt, dass das Netzwerk, wenn es einem Host zugeteilt worden ist, als aktive Abhängigkeit gilt. Dies bedeutet, dass oVirt vor der Migration einer VM prüft, ob das Netzwerk auch auf der Gegenseite verfügbar ist. Sollte dies nicht der Fall sein so wird die Migration abgelehnt. Man hat also die Sicherheit, dass eine Migration vor der Ausführung stets geprüft wird. Zusätzlich erlaubt oVirt auch nicht die Bildung von HA unter Nodes, welche nicht sämtliche **Erforderlichen** Netzwerke einem Interface zugewiesen haben. Bei **Nicht Erforderlichen** Netzwerken ist die Bildung von HA möglich und oVirt akzeptiert jede Konstellation. Das Migrieren einer Maschine in ein **nicht erforderliches** Netzwerk wird aber dennoch untersagt. Diese Konfiguration soll die freie Netzwerkbildung ermöglichen und oVirt dennoch die Möglichkeit bieten zwischen möglichen Szenarien zu differenzieren.

#### 8.7.8.1.2 Netzwerk- seitiger Vergleich mit anderen Lösungen

An dieser Stelle seien alle an oVirt interessierten Nutzer gewarnt. Die unter VMware oder Xenserver bekannte Bildung von isolierten Netzwerken, welche nicht direkt einem physischen Link unterstellt sind ist hier nicht möglich. Man muss jedes erstellte Netzwerk auch einem Link zuweisen, da es ansonsten nicht genutzt werden kann und Down bleibt. Es ist zwar möglich mehrere Netzwerke einem Link oder einer Link Aggregation zuzuweisen. Diese können dann isoliert voneinander über die gleiche

physische Verbindung betrieben werden. Techniken wie vSwitch von VMware, virtuelle Netzwerke von Xenserver oder die Bildung von vSwitches bei Solaris (Crossbow) sind jedoch nicht möglich. Distributed Switches sind bei dieser Technik des klassischen Linux bridging auch nicht möglich.

## 8.8 Abschluss des groben Teiles

Ab hier haben wir einen voll funktionsfähigen Cirtualisierungscluster, welcher bereits produktiv genutzt werden könnte. Der Betrieb wäre ab hier doch etwas umständlich und mit viel Einzelkonfigurationsarbeit verbunden, da Vorlagen fehlen, Beschränkungen nicht gesetzt sind und vieles mehr. Dies alles folgt im finalen Abschlussteil dieses Hauptkapitels. An dieser Stelle wird mit der Dokumentation zu Gunsten von ersten Grobtests abgebrochen. Da die gröberen Tests, welche in Kapitel 9 dokumentiert sind, müssen vorgezogen werden, um bei allfälligen Schäden innerhalb der Testreihen ein einfacheres Recovery zu ermöglichen. So soll im Ernstfall das im nachfolgenden Kapitel beschriebene Fine – Tuning nicht beschädigt werden und keinen Zeitlichen Mehraufwand erzeugen.

## 8.9 Die Finale Feinkonfiguration

Nach Abschluss der Grobtests können wir uns dem Fine – Tuning des Clusters widmen. Um den Überblick noch zu behalten, ist nachfolgend ein kleines Übersichtsschema zu sehen, welches die wichtigsten Highlights nochmals hervorhebt.

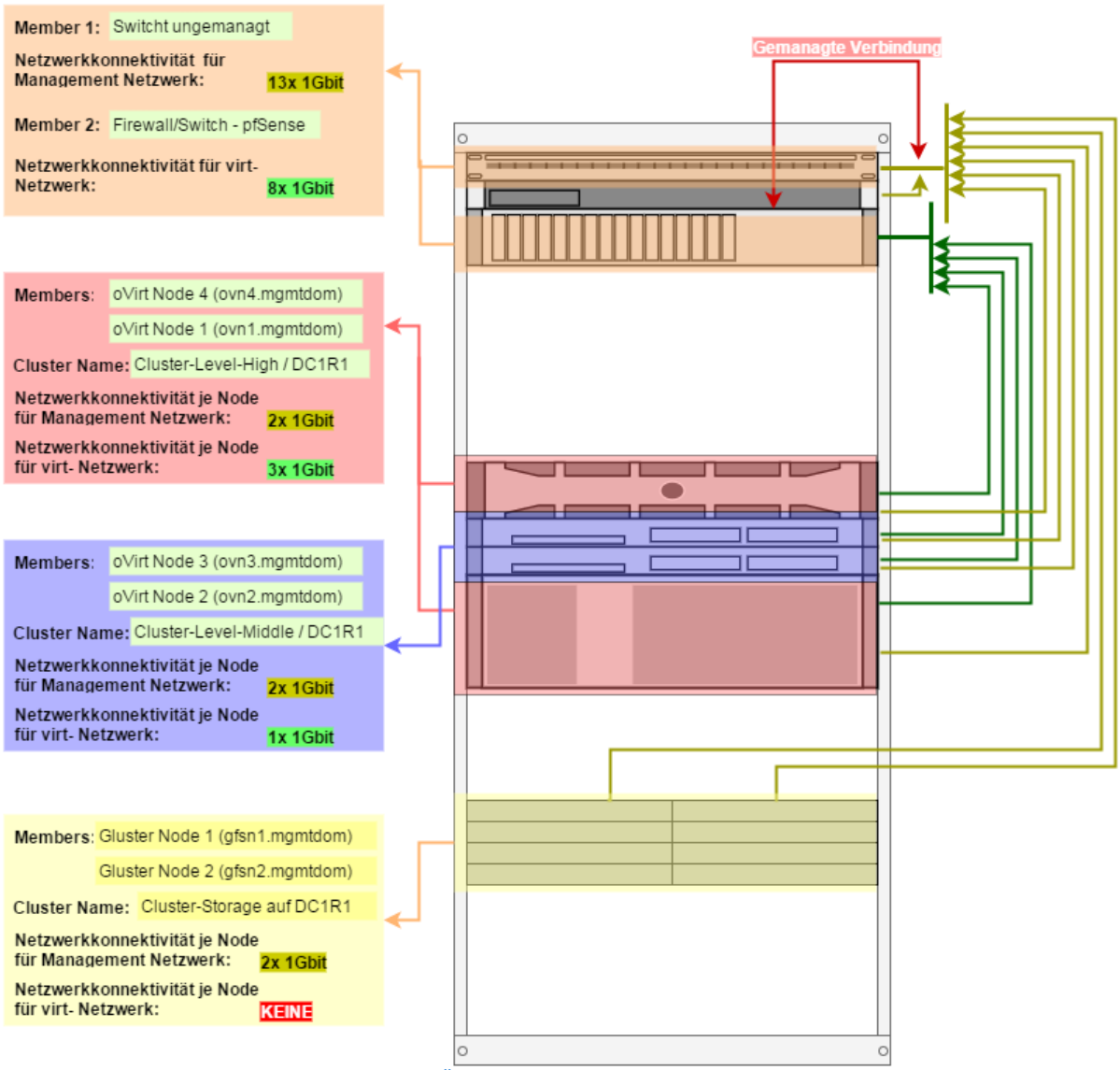



*Abbildung 58: Beginn mit dem Fine - Tuning: Übersichtsschema des Rohaufbaus*

Dieser Abschnitt behandelt die extra Funktionen, welche es uns ermöglichen sollen, Ressourcen zu begrenzen, VM's schneller aus Vorlagen auszurollen oder die Konfiguration des Semi-Automatic-HA anzuwenden. Es werden auch kleinere Exkurse zu Definitionen enthalten sein, welche relevant für diese Arbeit sind aber auch solche, welche nicht explizit im Auftrag enthalten sind, jedoch die

Integrität dieser Arbeit als neutrale Installations- und Konfigurationsanleitung wahren sollten. In diesem Abschnitt werden an dieser Stelle keine Vordefinitionen aufgelistet, es wird viel mehr nach der Methodik „documenting-by-doing“, Schritt für Schritt konfiguriert und gleichzeitig erklärt. Die Logik des Vorgehens ist zwar grösstenteils von sich aus vorgegeben, jedoch ist an gewissen Stellen mindestens die Reihenfolge variabel. Hier wird nach bestem Wissen und Gewissen des Autors vorgegangen, wobei mögliche Alternativwege spezifisch mit den bekannten Infoboxen aufgezeigt werden. Beginnen wir nun mit der freien Konfiguration.

### 8.9.1 Die Ressourcenbegrenzung

Als erstes werden wir die im Pflichtenheft definierten drei Ressourcenbegrenzungen _MIN, _STD und _HIGH definieren, sodass wir später nicht nachrüsten müssen. Zu diesem Zweck gehen wir folgenden Klick – Weg:

```
System (Seitenleiste) → Data-Center (Registerkarte) → QoS (untere Registerkarte)
```

*Befehl 28: Realisierung des Virtualisierungssegmentes (Klicken): Zum Konfigurationsmenü des Netzwerks von ovn4.mgmtdom*

Hier sehen wir die drei Hauptkonfigurationskategorien **Speicher**, **Netzwerk** und **CPU**. Unter Data – Center können die Parameter global gesetzt werden und gelten dann auch Data – Center weit. Das Hinzufügen der globalen Werte muss anschliessend manuell an diversen Orten erfolgen.

#### 8.9.1.1 Setzen der globalen Werte

In den nachfolgenden Unterabschnitten werden wir die drei Ressourcen Kategorien einzeln durch konfigurieren. Wobei die Ressourcenzuteilung dabei folgenden Richtwerten entspricht:

- **_MIN**: Entspricht dabei stets ca. 1/3 des absoluten Maximums der Kategorie.
- **_STD**: Wird innerhalb dieser Arbeit als Default Wert eingestuft. Der Grund für diesen tiefen Wert als Standardwert ist die höhere Ausnutzung an möglichen VM's. Technisch gesehen sollte der Standardwert niemals am Maximalwert liegen, da man vielleicht später noch nachkorrigieren möchte.
- **_HIGH**: Ist die absolute Obergrenze und ist per Default stets auf unbegrenzt gesetzt. Diese Option ist für VM's höchster Priorität reserviert.

##### 8.9.1.1.1 Setzen des globalen QoS für den Speicherzugriff (Storage)

Hierfür klicken wir auf **Speicher** und anschliessend auf **Neu,** woraufhin sich ein kleines Konfigurationsfenster öffnet. In diesem Fenster können wir nun zuoberst den **QoS-Namen** vergeben. Dieser soll aussagekräftig QoS_Storage_MIN (bzw. _STD, _HIGH) lauten. Der hier vergebene Name ist später nicht bei der VM Erstellung sichtbar, er muss also noch keine logischen Zuordnungen bezüglich Cluster beinhalten. Die Beschreibung kann frei gewählt werden, sollte aber mindestens einen Hinweis auf gewisse Parameter enthalten. Hier wurden in die Beschreibungen die Prozentualwerte mit hinein gefügt. Nachfolgend müssen die Grenzwerte definiert werden. Ovirt bietet hier die Möglichkeit, den Durchsatz als Zahl in Mega Byte anzugeben und zusätzlich die Input – Output Aktionen pro Sekunden.

Da die Berechnung der I/O pro Sekunde zu kompliziert wäre, genügt es, hier die Einschränkung als MB – Wert unter **Durchsatz** anzugeben. An dieser Stelle ist es möglich einen Gesamtwert anzugeben. Ovirt berechnet selbstständig die Idealwerte für Lesen und Schreiben, oder man gibt diese Wert manuell ein. Wir werden die Werte manuell eingeben und eine leichte Asymmetrie erzeugen, indem wir dem Leseprozess etwas mehr Leistung geben. Zur Berechnung der Werte wird nachfolgende Formel verwendet:

**Maximaler Speicherzugriff**: Dieser wird über eine 1 Gbit/s Leitung zum Storage definiert. Hier wird statisch mit dem Maximalwert von 1 Gbit/s pro Host gerechnet, die Belastung der Leitung durch mehrere Host – Zugriffe wird hier ignoriert.

Hieraus resultiert 1000 Mbit/s / 8 bit pro Byte = 125 MB/s

125 MB/s * 0.3 (30%) = 37.5 MB/s → **Abgerundet auf 35 MB/s**

Somit ist der priorisierte Lesezugriff auf 35 MB/s zu setzen und der langsamere Schreibzugriff wird der Asymmetrie halber auf **30 MB/s** gesetzt. Diese Einstellung kann nun mit **OK** übernommen werden, woraufhin sie in der Liste erscheint.

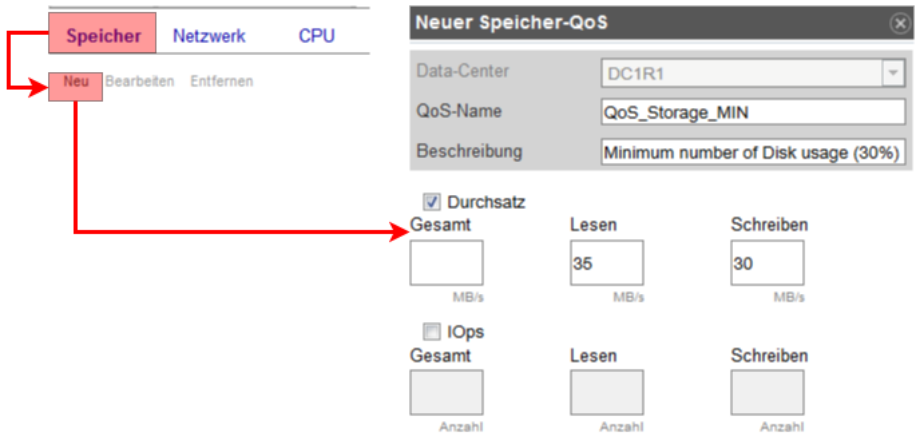



*Abbildung 59: Beginn mit dem Fine - Tuning: QoS - Speicher; erstellen der globalen Profile*

Diese Schritte können nun für die Profile **_STD** und **_HIGH** wiederholt werden, wobei natürlich die Namen und Beschreibungen anzupassen sind. Die Zahlenwerte für die beiden Profile sind folgende:

- **_STD** (60% des Maximums) → Lesen **65 MB/s**, Schreiben **60 MB/s**
- **_HIGH** (100% bzw. unbegrenzt) → Lesen **NICHTS**, Schreiben **NICHTS**

So sollte die Konfiguration schlussendlich aussehen:

| QoS-Name | Beschreibung | Gesamt-Durchsat | Lese-Durchsatz | Schreib-Durchsatz | Gesamt-IOps | Lese-IOps | Schreib-IOps |
|---|---|---|---|---|---|---|---|
| QoS_Storage_MIN | Minimum number of Disk usage (30%) | Unbegrenzt | 35 | 30 | Unbegrenzt | Unbegrenzt | Unbegrenzt |
| QoS_Storage_STD | Default number of Disk usage (60%) | Unbegrenzt | 65 | 60 | Unbegrenzt | Unbegrenzt | Unbegrenzt |
| QoS_Storage_HIGH | Maximum number of Disk usage (100%) | Unbegrenzt | Unbegrenzt | Unbegrenzt | Unbegrenzt | Unbegrenzt | Unbegrenzt |

*Abbildung 60: Beginn mit dem Fine - Tuning: QoS - Speicher; Das fertige Ergebnis*

#### 8.9.1.1.2 Setzen des globalen QoS für den Netzwerkzugriff

An der gleichen Stelle können wir nun auf den Button **Netzwerk** und anschliessend auf **Neu** klicken und wir landen wieder beim Konfigurationsmenü. Hier kann aus einem unerfindlichen Grund nur der Name aber keine Beschreibung angegeben werden. Wir geben hier wieder den Namen nach unserem definierten Standard ein → QoS_Network_MIN. Hier bietet uns oVirt die Möglichkeit, den eingehenden wie auch den ausgehenden Datenverkehr zu manipulieren. Asymmetrische Werte sind hier vor allem bei Webservern sinnvoll, wo der ausgehende Verkehr um einiges höher sein muss als der eingehende. Wir werden auch hier unsere Zerteilung in **30%**, **60%** und **unbeschränkt** anwenden, wobei hier folgende Formal zum Zuge kommt:

Maximaldurchsatz → 1000 Mbit/s * 0.3 (30%) = **300 Mbit/s**

Somit nutzen wir an dieser Stelle für _MIN den Durchschnittswert **300** und multiplizieren dies mit **2** um den Peak zu erhalten. Die Verdoppelung des Spitzenwertes ist ein reiner Gefühlswert des Autors und absolut akzeptabel in Kombination mit dem dritten Wert **100 MB** für Burst. Der Burst Wert besagt nämlich, dass der Peak nur für die Menge von 100 MB pro auftretendem Schub zulässig ist.

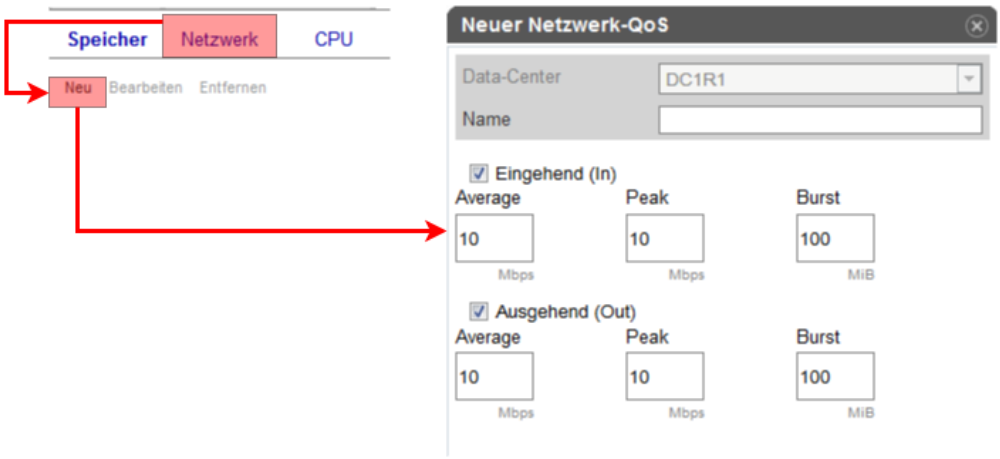



*Abbildung 61: Beginn mit dem Fine - Tuning: QoS - Netzwerk; Erstellen der globalen Profile*

Wir wenden diese Werte für den eingehenden sowie für den ausgehenden Datenverkehr an, da hier nicht speziell auf die Infrastruktur (Webserver) geachtet werden muss.

Für die Profile _STD und _HIGH sehen die Parameter nach unserer prozentualen Aufteilungs- Skala wie folgt aus:

- _STD (bei 60%) → Average = 600, Peak = 1000 (1200 geht ja nicht), Burst = 200
- _HIGH (bei 100%) → Average = NICHTS, Peak = NICHTS, Burst = NICHTS

Die fertige Konfiguration sollte so aussehen:

| Name | In – Average | In – Peak | In – Burst | Out – Average | Out – Peak | Out – Burst |
|---|---|---|---|---|---|---|
| QoS_Network_MIN | 300 | 600 | 200 | 400 | 600 | 200 |
| QoS_Network_STD | 600 | 1000 | 200 | 600 | 1000 | 200 |
| QoS_Network_HIGH | Unbegrenzt | Unbegrenzt | Unbegrenzt | Unbegrenzt | Unbegrenzt | Unbegrenzt |

*Abbildung 62: Beginn mit dem Fine - Tuning: QoS - Netzwerk; Das fertige Ergebnis*

#### 8.9.1.1.3 Setzen des globalen QoS für den CPU – Zugriff

Kommen wir zum letzten Abschnitt der globalen Konfiguration, der CPU Limitierung. Hier können blanke Zahlenwerte gesetzt werden. Diese mussten zuerst mittels → Google – Anfrage ermittelt werden, um überhaupt zu wissen, von welcher Art Zahlenwert die Rede ist. Es handelt sich um Prozentangaben, welche sich stets auf die gesamte virtuelle CPU beziehen. Hier ist das Vorgehen gleich wie unter Abbildung 59 und 61, jedoch wählen wir an dieser Stelle den Button **CPU**. Dieser öffnet uns nachfolgendes Fenster, wo wieder ein Name und eine Beschreibung definiert werden müssen. Die Beschränkung der CPU Nutzung wird hier auf simple Art mit einem prozentualen Zahlenwert angegeben. In unserem Referenzbeispiel geben wir für **_MIN** den Begrenzungswert 30 ein, womit wir die Nutzung der virtuellen CPU (Core – Anzahl belanglos) auf 30% des verfügbaren Maximums herabstufen. Für _STD und _HIGH setzen wir die Werte 60 bzw. 100.

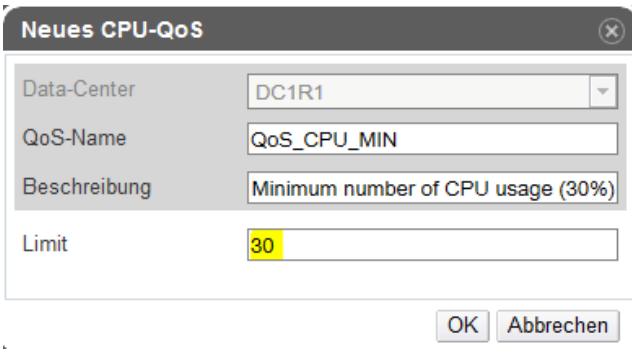



*Abbildung 63: Beginn mit dem Fine - Tuning: QoS - CPU; Das Konfigurationsfenster*

Das fertige Ergebnis sollte wie folgt aussehen:

| QoS-Name | Beschreibung | Limit |
|---|---|---|
| QoS_CPU_MIN | Minimum number of CPU usage (30%) | 30 |
| QoS_CPU_STD | Default number of CPU usage (60%) | 60 |
| QoS_CPU_HIGH | Maximum number of CPU usage (100%) | 100 |

*Abbildung 64: Beginn mit dem Fine - Tuning: QoS - CPU; Das fertige Ergebnis*

### 8.9.1.2 Verteilen der globalen QoS Profile

Leider ist die Verteilung der Profile nicht so luxuriös gelöst wie bei VMware. Hier müssen die Profile noch manuell in den drei Destinationen eingebunden werden. Hierzu müssen wir zu den Registerkarten **Cluster**, **Netzwerk** und **Speicher** gehen, wo die Einbindepunkte teilweise etwas versteckt liegen.

#### 8.9.1.2.1 Einbinden des QoS – Profils für die CPU – Limitierung

Um zum Einbindepunkt zu gelangen nutzen wir folgenden Klick – Weg:

```
System (Seitenleiste) → Cluster (Registerkarte) → Cluster-Level-High (in der Auswahlliste) CPU-Profile (untere Registerkarte)
```

*Befehl 29: Realisierung des Virtualisierungssegmentes (Klicken): Beginn mit dem Fine - Tuning: QoS - CPU; der Weg zum Konfigurationspunkt des Einbindens.*

Dort treffen wir auf eine Liste mit einem bestehenden Eintrag und dem bekannten Button **Neu**, welchen wir nun anklicken. Im sich nun öffnenden Konfigurationsmenü haben wir die Möglichkeit einen Namen und eine Beschreibung zu setzen. Der dritte Parameter mit dem Namen **QoS** ist eine Auswahlliste, in welcher die im letzten Abschnitt erstellten QoS Profile aufgeführt sind. Nun vergeben wir als erstes einen sinnvollen Namen, welcher eine klare Referenz zu unserem Ressourcen – Zuleitungsschema hat wie bspw. **QoS_CPU_MIN**. Die Namensüberlappung an dieser Stelle ist nicht gewollt, ergibt sich aber als einzige logische Variante. In der Beschreibung geben wir an, dass es sich um einen Link zum QoS Profil handelt. In der Auswahlliste suchen wir das QoS Profil, welches unseren Verweis auf _MIN besitzt. Das war es schon! Mit einem Klick auf **OK** werden die Einstellungen übernommen. Dies wiederholen wir nun auch für die Einstellungen **_STD** und **_HIGH,** wobei wir hier natürlich die entsprechenden Profile auswählen.

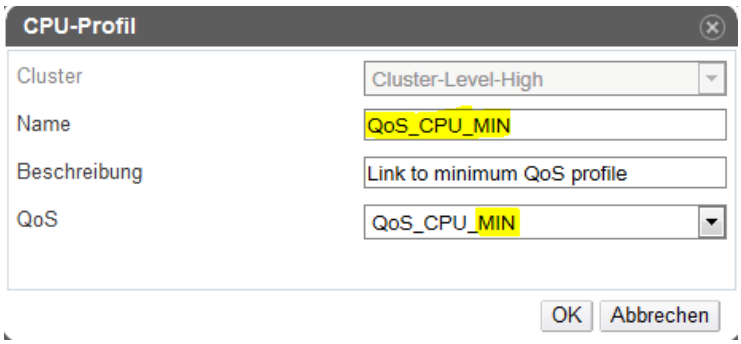



*Abbildung 65: Beginn mit dem Fine - Tuning: QoS - CPU; Einbinden des CPU - QoS Profiles für Cluster-Level-High _MIN*

An dieser Stelle haben wir nun die drei QoS Profile für Cluster-Level-High an drei Auswahlparameter gebunden, welche wir später bei der VM – Erstellung nutzen können. Diese drei Schritte können nun äquivalent für Cluster-Level-Middle angewendet werden. Das fertige Ergebnis sollte nun wie folgt aussehen:

| Name | Beschreibung | QoS-Name |
|---|---|---|
| Cluster-Level-High | | Unbegrenzt |
| QoS_CPU_MIN | Link to minimum QoS profile | QoS_CPU_MIN |
| QoS_CPU_STD | Link to default QoS profile | QoS_CPU_STD |
| QoS_CPU_HIGH | Link to high QoS profile | QoS_CPU_HIGH |

*Abbildung 66: Beginn mit dem Fine - Tuning: QoS - CPU; Einbinden des CPU - QoS Prfiles; das fertige Ergebnis für Cluster-Level-High*

| Name | Beschreibung | QoS-Name |
|---|---|---|
| Cluster-Level-Middle | | Unbegrenzt |
| QoS_CPU_MIN | Link to QoS minimum profile | QoS_CPU_MIN |
| QoS_CPU_STD | Link to QoS default profile | QoS_CPU_STD |
| QoS_CPU_HIGH | Link to QoS high profile | QoS_CPU_HIGH |

*Abbildung 67: Beginn mit dem Fine - Tuning: QoS - CPU; Einbinden des CPU - QoS Prfiles; das fertige Ergebnis für Cluster-Level-Middle*

Wie in den beiden Abbildungen oben zu erkennen ist, gibt es jeweils zuoberst eine Einstellung mit dem Flag **Unbegrenzt**. Hierbei handelt es sich um die Default Konfiguration, welche von der Engine bei der Installation angelegt wird. Sie wird so stehen gelassen, da sie nicht stört und die Auswirkungen bei oVirt, was das Löschen von Systemeinstellungen angeht, etwas heikel sein kann.

#### 8.9.1.2.2 Einbindung des QoS – Profils für die Netzwerklimitierung

Die Einbindung der Netzwerkbeschränkungen ist vom Vorgehen her fast gleich wie im vorhergegangenen Abschnitt. Hier gehen wir wie nachfolgend beschrieben zum Netzwerkkonfigurations- Tab:

```
System (Seitenleiste) → Netzwerk (Registerkarte) → vNIC-Profile (untere Registerkarte)
```

*Befehl 30: Realisierung des Virtualisierungssegmentes (Klicken): Beginn mit dem Fine - Tuning: QoS - Netzwerk; der Weg zum Konfigurationspunkt des Einbindens.*

Auch hier sehen wir den System – Standarteintrag und den altbekannten Button **Neu**. Mittels Klick auf **Neu** öffnet sich ein Konfigurationsmenü. Als erstes sehen wir den Namen. Hier wählen wir **VMnet1_MIN** als sinnvollen Namen. Die Beschreibung soll uns auch an dieser Stelle einen Verweis auf das QoS Profil geben. Wie im oberen Abschnitt beschrieben, können wir auch hier aus der Auswahlliste das zu referenzierende QoS Profil mit der Endung **_MIN** auswählen. Zusätzlich steht hier noch die Option **Port-Mirroing** zur Auswahl, welche zu Debugging – Zwecken aktiviert werden kann, falls man Traffic mitschneiden möchte. Die anschliessende Auswahlliste bietet eine Option, bei der es sich um das Bilden von SecurityGroups dreht. Diese Option wird hier nicht genutzt und soll auch nicht weiter

thematisiert werden. Am Ende des Konfigurationsmenüs kann man noch angeben, ob die neu erstellte Limitierung allen Benutzern zugänglich gemacht werden soll. Dies macht beim späteren Hinzufügen von weiteren Benutzern auch Sinn und sollte aktiv gelassen werden.

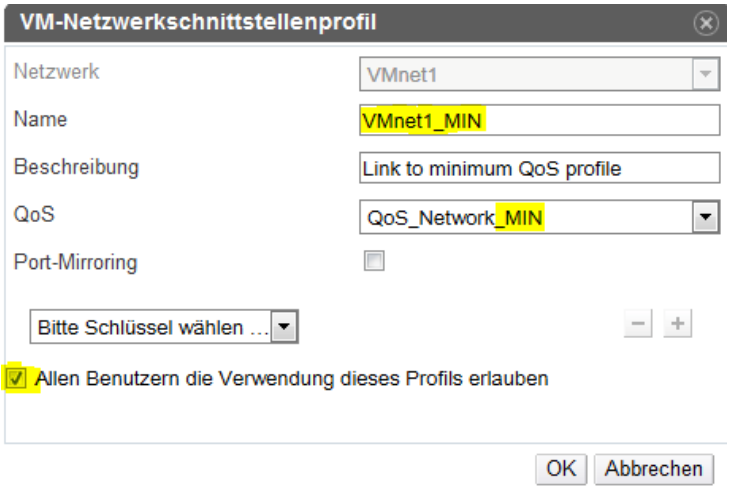



*Abbildung 68: Beginn mit dem Fine - Tuning: QoS - Netzwerk; Einbinden des Netzwerks - QoS Profils für das Netzwerk (Data - Center weit)*

Diese Konfigurationen wiederholen wir nun für die restlichen beiden Limitierungen _STD und _HIGH. Da die allgemeine Netzwerkkonfiguration Data – Center- weit gültig ist, sind somit auch alle unsere Einstellungen entsprechend gleich weit gültig, womit eine Cluster – Separierung an dieser Stelle wegfällt.

Das fertige Ergebnis sollte in etwa wie folgt aussehen:

| Name | Netzwerk | Data-Center | Kompatibilitätsversion | QoS-Name | Port-Mirror | Beschreibung |
|---|---|---|---|---|---|---|
| VMnet1 | VMnet1 | DC1R1 | 3.5 | | | |
| VMnet1_HIGH | VMnet1 | DC1R1 | 3.5 | QoS_Network_HIGH | | Link to high QoS profile |
| VMnet1_MIN | VMnet1 | DC1R1 | 3.5 | QoS_Network_MIN | | Link to minimum QoS profile |
| VMnet1_STD | VMnet1 | DC1R1 | 3.5 | QoS_Network_STD | | Link to default QoS profile |

*Abbildung 69: Beginn mit dem Fine - Tuning: QoS - Netzwerk; Einbinden des Netzwerk - QoS Prfiles; das fertige Ergebnis*

#### 8.9.1.2.3 Einbindung des QoS – Profils für die Storagelimitierung

Die Einbindung der Storagelimitierungen folgt dem gleichen Muster wie bisher und spielt sich unter folgendem Punkt ab:

```
System (Seitenleiste) → Speicher (Registerkarte) → Storage-R1 (Storage Name) → Disk-Profile
```

*Befehl 31: Realisierung des Virtualisierungssegmentes (Klicken): Beginn mit dem Fine - Tuning: QoS - Speicher; der Weg zum Konfigurationspunkt des Einbindens.*

An dieser Stelle werden wir nur den Master Storage „Storage-R1“ konfigurieren, da eine Limitierung der ISO Speicher allgemein mehr als sinnlos wäre. Auch an dieser Stelle gehen wir auf **Neu** und treffen das altbekannte Konfigurationsmenü wieder vor. Hier ist der zu konfigurierenden Aufwand wieder geringer und es genügen Name, Beschreibung und die korrekte Wahl aus der Auswahlliste. Als Namen vergeben wir hier wieder einen aussagekräftigen, welcher uns auch gleich auf den Storage selbst verweist, in diesem Fall **Storage-R1_MIN**. Die Beschreibung ist wieder ein Link – Verweis auf die eigentlichen QoS Profile und die korrekte Wahl aus der Auswahlliste ist wieder der Eintrag, welcher die Begrenzung im Namen trägt, also **_MIN**. Auch hier wiederholen wir alle Schritte, um die noch offenen Limitierungen für _STD und _HIGH zu erzeugen.

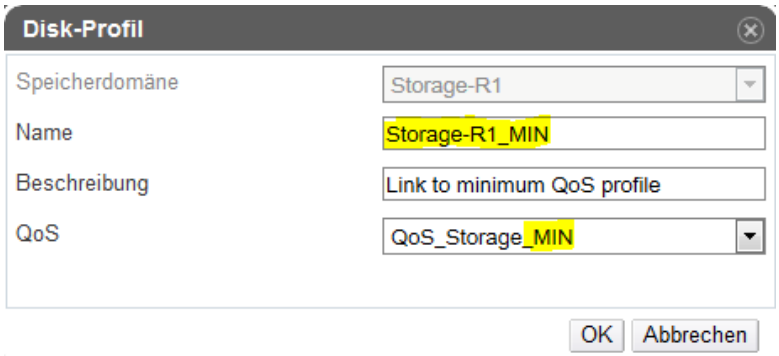



*Abbildung 70: Beginn mit dem Fine - Tuning: QoS - Storage; Einbinden des Netzwerks - QoS Profils für den Storage Storage-R1 - Master (Data - Center weit)*

Wenn wir alles richtig gemacht haben, sollte das Schlussergebnis wie in nachfolgender Abbildung aussehen:

| Name | Beschreibung | QoS-Name |
|---|---|---|
| Storage-R1 | | Unbegrenzt |
| Storage-R1_HIGH | Link to high QoS profile | QoS_Storage_HIGH |
| Storage-R1_MIN | Link to minimum QoS profile | QoS_Storage_MIN |
| Storage-R1_STD | Link to default QoS profile | QoS_Storage_STD |

*Abbildung 71: Beginn mit dem Fine - Tuning: QoS - Storage; Einbinden des Storage - QoS Prfiles; das fertige Ergebnis*

Dies war auch schon die komplette Ressourcen – Konfiguration. Sie ist wie bereits erwähnt nicht so komfortabel wie bei VMware, aber sie erfüllte ihren funktionalen Zweck vollkommen. Ab diesem Punkt haben wir einen fertigen Virtualisierungscluster, welcher nicht verschwenderisch mit seinen Ressourcen umgeht und uns so eine effizientere und höhere Anzahl an virtuellen Maschinen ermöglicht.

### 8.9.2 Die erste VM

Da wir nun einen voll funktionsfähigen Cluster mit der Möglichkeit der Ressourcenbegrenzung haben, wird es Zeit, die erste produktive virtuelle Maschine einzurichten. Hierfür nehmen wir ein openSUSE, welches zugleich eine wichtige Komponente im VM – Netzwerksegment erfüllen soll, nämlich der DHCP / DNS Server. Dazu geben wir nachfolgenden Befehl auf **gfsn1.mgmtdom** ein, um den **ISO_STORE** als NFS Mountpoint lokal ins **/mnt** einzubinden. Dort können wir anschliessend per Kommandozeilenbefehl **wget** das ISO – File von opensuse.org holen.

```
#  mount.nfs -o mountproto=tcp gfsn1.mgmtdom:/ISO_STORE /mnt
```

*Befehl 32: Realisierung des Virtualisierungssegmentes (Klicken): Herunterladen des openSUSE ISO's*

Wenn das ISO auf dem Storage ist, können wir uns zu einem VM – Erstellungspunkt begeben. Hierfür stehen uns mehrere Wege zur Verfügung, wobei in jedem Fall das gleiche Konfigurationsmenü geöffnet wird. Wir nehmen den einfachsten Weg und machen alles über die globale Ansicht, welche wir auf mit folgender Klick – Kombination erreichen:

```
System (Seitenleiste) → Virtuelle Maschine (Registerkarte)
```

*Befehl 33: Realisierung des Virtualisierungssegmentes (Klicken): Herunterladen des openSUSE ISO's*

Am oberen Rand der Listenansicht sehen wir den Button **Neue VM** welchen wir nun anklicken. Beim sich öffnenden Konfigurationsassistenten sehen wir neben den bekannten Parametern auch die möglichen Konfigurationskategorien, welche an dieser Stelle aber auf zwei beschränkt sind. Technisch gesehen würden diese beiden Kategorien für den Normalbetrieb auch ausreichen, da der Grossteil aus fix vordefinierten Default – Parametern besteht. Wir nehmen uns aber die Zeit und konfigurieren diese Maschine mit einigen Extraoptionen, weswegen wir im Konfigurationsassistent zu unterst den Button **Erweiterte Optionen anzeigen** wählen. Nun sehen wir die acht zusätzlichen Parameter, welche wir nachfolgend Schritt für Schritt durchgehen werden.

#### 8.9.2.1 Die einzelnen Konfigurationsschritte der VM – Erstellung

In den nachfolgenden Unterkapiteln werden wir die wichtigsten Parameter durcharbeiten, welche für die Realisierung dieser Arbeit notwendig sind. Einige Punkte, welche nicht zwingend notwendig sind werden an dieser Stelle nur leicht tangiert. Sie werden nicht ausführlich thematisiert, da dies sonst den Rahmen dieses Kapitels sprengen würde.

##### 8.9.2.1.1 Kategorie → Allgemein

In dieser Kategorie sehen wir einen hellgrau markierten Bereich, welcher für die Primär – Konfiguration gedacht ist, welche sich nie verändert und bei jedem Kategoriewechsel stets die gleichen Informationen enthält. Dieser Bereich wird für die nachfolgende Dokumentation der Schritte mit Basiskonfiguration betitelt. Dabei enthält dieser Bereich folgende Konfigurationspunkte:

- **Cluster:** Hier kann der zu nutzende Cluster angegeben werden. Wir werden unsern DHCP / DNS Server, an dieser Stelle nur noch als **Infraserv1** bezeichnet, in den Cluster-Level-Middle platzieren.
- **Basierend auf Vorlage:** Hier können bereits erstellte VM's, welche als Vorlage definiert worden sind, angegeben werden. Momentan haben wir ja aber keine Vorlage, deshalb belassen wir dieses Feld auf **Blank**.
- **Vorlage-Subversion:** Es besteht bei oVirt die Möglichkeit aus VM's welche als Vorlage dienen auch Subversionen zu generieren, welche Abweichungen zur ursprünglichen Version besitzen. Bei dieser Arbeit wird auf dieses Feature verzichtet.
- **Betriebssystem:** Hier kann die OS Variante angegeben werden, wobei auch noch eine Unterteilung in die unterschiedlichen Versionen möglich ist. Diese Option konfiguriert die darunterliegenden XML – Files, welche KVM als Konfigurationsparameter nutzt, entsprechend den Bedürfnissen des OS, also bspw. mehr CPU – Befehlssätze oder die Vordefinition für die Nutzung der KVM eigenen Libvirt – Libraries zur teilweisen Paravirtualisierung. Da unsere openSUSE 42.1 Version nicht in der Liste anzutreffen ist, nutzen wir einfach die Option **Linux**. Dies ist kein Problem, da heute die meisten Distributionen praktisch identisch sind.
- **Instanztyp:** Möchte man sich nicht selbst die Mühe machen und Leistungsparameter wie CPU (Core – Anzahl) oder RAM definieren, so kann man hier unter fünf Leistungsstufen (Small – XLarge) wählen. XLarge ist hier die höchste Stufe und verwendet 4 CPU's bei ca. 16 GB RAM. Wir werden hier **Benutzerdefiniert** wählen und unsere Parameter nachfolgend selbst definieren.
- **Optimiert für:** Hier kann zwischen Server und Desktop gewählt werden. Die beiden Parameter definieren, wie die restliche Konfiguration bezüglich grafischem Subsystem aussehen soll. Im Falle der Wahl Desktop, wird der SPICE Server auf dem Host so konfiguriert, dass die maximale grafische Performance zur Verfügung steht. Bei unserem openSUSE werden wir zwar ein minimales X11 nutzen, jedoch ist hier der Parameter **Server** die ideale Wahl.

Dies war das Ende der Basiskonfiguration, welche sich nun bei jedem Kategoriewechsel nicht mehr ändern wird und auch in den nachfolgenden Schritten nicht mehr thematisiert wird.

- Name: Hier kann ein VM – Name vergeben werden, in unserem Fall wird folgende Namenskonvention definiert und kontinuierlich für alle VM's als Design – Vorlage genutzt.
  - **DYN_HIGH_InfraServ1**
    - DYN: Sagt aus, dass die VM Live migriert werden darf.
    - _HIGH: Definiert die im vorgängigen Schritt konfigurierte Ressourcen – Limitierung.
    - _<NAME>: An dieser Stelle steht der eigentliche Name der VM. Hier gibt es die Möglichkeit der Namensraumerweiterung, welche später mit einem Bindestrich definiert wir.
- Beschreibung und Kommentar: Die Begrifflichkeiten sind an dieser Stelle selbsterklärend.

- **Die möglichen Zustandsoptionen:**
  - **Zustanslos:** Hier konnte keine aktuelle Dokumentation seitens oVirt geliefert werden, da die meisten → Google – Anfragen zwar Ergebnisse lieferten, diese jedoch in einem „Not found :(“ auf dem oVirt Webserver landeten. Zwar zeigt oVirt den Status **Stateless** im Icon der VM an, jedoch haben diverse Versuche gezeigt, dass dies weder Einfluss auf Migration, Vorlagenerstellung oder den regulären Betrieb hat. Es kann somit nicht eindeutig identifiziert werden, ob diese Funktion überhaupt noch in Betrieb ist oder einfach nur ein Überbleibsel einer älteren Version.
  - **Im Pausenmodus starten:** Der Sinn dieser Funktion entzieht sich vollkommen dem Verständnis des Autors. Hier wird die VM beim starten automatisch in einen Pausenmodus versetzt, aus welchem sie mit einem erneuten Startversuch befreit werden kann. Dies macht weder aus Ressourcen- noch aus Debugging – Gründen Sinn. Der genaue Nutzen dieser Funktion ist nicht exakt ersichtlich.
  - **Löschschutz:** Der Einsatz dieses Parameters ist stets zu empfehlen, wenn mehrere Administratoren am System arbeiten. Denn solange jeder Admin einen eigenen Account hat, kann kein anderer (auch Admin) diese VM ohne Einwilligung des Besitzers löschen.
- **Virtueller Netzwerkadapter:** An dieser Stelle kann aus einer Auswahlliste der erste NIC an ein Netzwerk gebunden werden. Hier genügt vorläufig ein NIC, welcher an das „limitierte“ QoS – Netzwerk **VMnet1_HIGH** gebunden und mit dem „+“ - Button hinzugefügt werden kann.

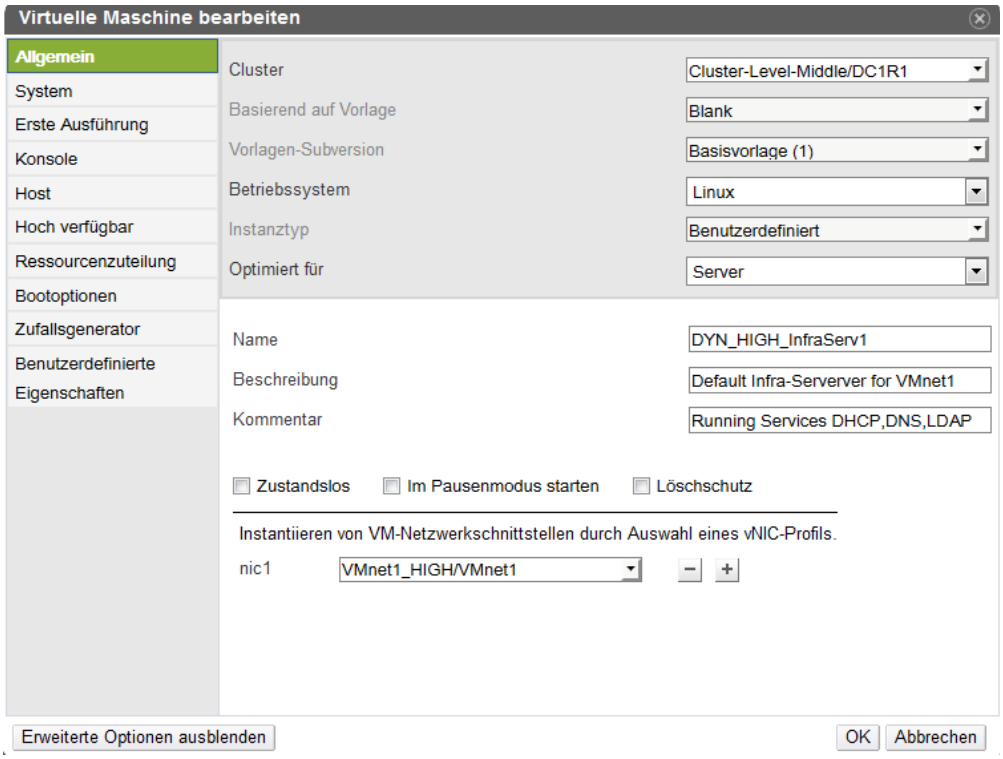



*Abbildung 72: Beginn mit dem Fine - Tuning: Vm - Erstellung; Kategorie Allgemein – Beispiel InfraServ1*

#### 8.9.2.1.2 Kategorie → System

In dieser Konfigurationskategorie können die elementarsten und der Hardware nahestehendsten Einstellungen vollzogen werden, welche die Art der CPU und die Menge an RAM definieren.

- **Arbeitsspeichergrösse:** Hier kann die Grösse des Arbeitsspeichers angegeben werden, was wir auch gleich mit 1024 MB RAM für unseren InfraServ1 tun. Da wir bei der Erstellung der Cluster gesagt haben, dass wir keine RAM Überschreitung zulassen wollen, können hier auch nicht mehr RAM pro VM vergeben werden als der Host im allgemeinen zur Verfügung hat. Zwar akzeptiert oVirt eine massive Überschreitung der Eingabe, jedoch lässt sich die VM so nicht starten.
- **Virtuelle CPU's insgesamt:** Hier kan die Anzahl an CPU defniert werden, welche der VM zugeteilt werden sollen.
- **Erweiterte Parameter:** Hier können Parameter gesetzt werden, die das Layout des Prozessors welcher zur VM gereicht wird, modifizieren. Dabei stehen folgende beiden Optionen zur auswahl:
  - **Kerne pro virtuellem Socket:** Die Anzahl Kerne je Socket welche der VM sichtbar gemacht werden. Hier wird nicht wie bei anderen Lösungen Simultaneous Multithreading angeboten, sondern es wird nur der virtuelle Kern dargeboten.
  - **Virtuelle Sockets:** Gibt die Anzahl an virtualisierten „realen“ CPU – Steckplätzen an.

  Somit kann man angeben, wie viele Prozessorsockets das virtuelle System haben soll und wie viele Kerne je Socket (CPU) vorhanden sein sollen. Hier Simultaneous Multithreading (SMT) anzubieten wäre nur eine sinnlose Verschwendung von Ressourcen, da sie der Hypervisor nur mühsam virtualisieren müsste. Die Anzahl an virtuellen CPU's gibt die Vorgabe, nach welcher oVirt bei diesen beiden Erweiterungsparametern automatisch alle möglichen Lösungen berechnet.
- Die restlichen Optionen sind von geringer Bedeutung und werden hier nicht weiter thematisiert.

InfraServ1 soll hier etwas stärker dimensioniert werden, da die Möglichkeit besteht neben der Hauptfunktion als DHCP / DNS Server auch andere Rollen zu einem späteren Zeitpunkt zu übernehmen.

Die fertige Konfiguration dieser Kategorie könnte wie folgt aussehen:

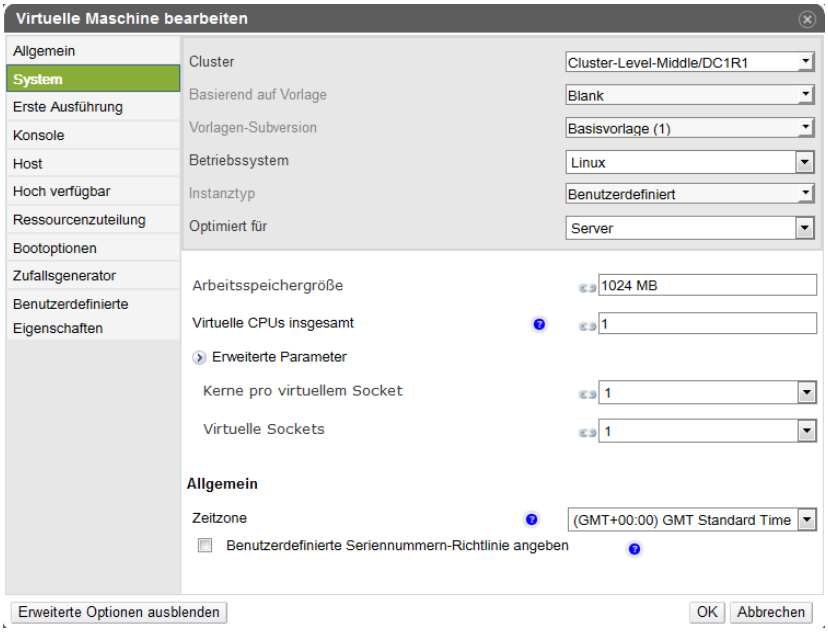



*Abbildung 73: Beginn mit dem Fine - Tuning: Vm - Erstellung; Kategorie System – Beispiel InfraServ1*

#### 8.9.2.1.3 Kategorie → Erste Ausführung

Diese Kategorie soll nicht spezifisch thematisiert werden, da es hier um die Optionen geht welche eine Vordefinition von Parametern wie Benutzername, Passwort oder SSH Schlüssel, ermöglichen sollen. Dies widerspricht der Grundeinstellung des Autors und ist primär für den Einsatz bei externen Providern wie OpenStack gedacht, was ebenfalls nicht den Vorstellungen des Autors entspricht. Somit kann die einzige Auswahloption an dieser Stelle deaktiviert bleiben.

#### 8.9.2.1.4 Kategorie Konsole

An dieser Stelle können diverse Parameter bezüglich der visuellen Darstellung der SPICE – Verbindung definiert werden. Dabei kann zwischen dem weit moderneren SPICE – Protokoll, welches auch Audio- und USB – Support anbietet und dem veralteten VNC – Protokoll gewählt werden. Hier empfiehlt sich auf jeden Fall die SPICE – Nutzung. Der Punkt USB – Unterstützung kann auf „Systemeigen“ belassen werden, um später von einem SPICE – Client aus einem eingesteckten USB – Flashdrive weiterreichen zu können. Doch an dieser Stelle muss gesagt werden, dass diese Funktion nur mit Fedora 23 und somit mit der aktuellsten SPICE – Version einwandfrei funktioniert.

**ACHTUNG!**

**Für den Windows SPICE – Client kann die USB – Unterstützung nachinstalliert werden. Die Installation des Treibers bildet eine Abstraktionsschicht, welche USB – Flashdrives abfangen soll um sie zum Hypervisor zu leiten. Die Software ist veraltet und scheint trotz Support nicht sauber zu funktionieren. Im schlimmsten Fall funktioniert danach keine USB – Schnittstelle mehr.**

Beim nächsten Punkt zeigt sich die Performance von SPICE, denn es ist möglich, bis zu vier Monitore auf der VM zu virtualisieren.

Der Punkt **Single-Sign-On-Methode** definiert ob ein Konsolen – Login überhaupt möglich sein soll und falls ja, wie denn die Parameter dazu aussehen. Die Auswahloption in der Basiskonfiguration, ob es sich um eine Server- oder Desktop – Maschine handelt, definieren hier die Parameter automatisch. Bei einer Server Konfiguration werden hier nur die nötigsten Parameter wie VirtIO – Konsolengerät und SPICE – Zwischenablage aktiviert. Beim Desktop sind alle Parameter automatisch gesetzt um Funktionen wie Soundkarten – Weiterleitung zu ermöglichen. Unter dem Menüpunkt **Erweiterte Parameter** ist noch der Parameter **Strenge Benutzerkontrolle deaktivieren** per Default aktiv. Es wird seitens oVirt mit einer Warnmeldung darauf hingewiesen, diesen Parameter nicht zu verändern. Entsprechende Versuche haben keinerlei unterschiedliches Verhalten beim Ändern dieses Parameters gezeigt.

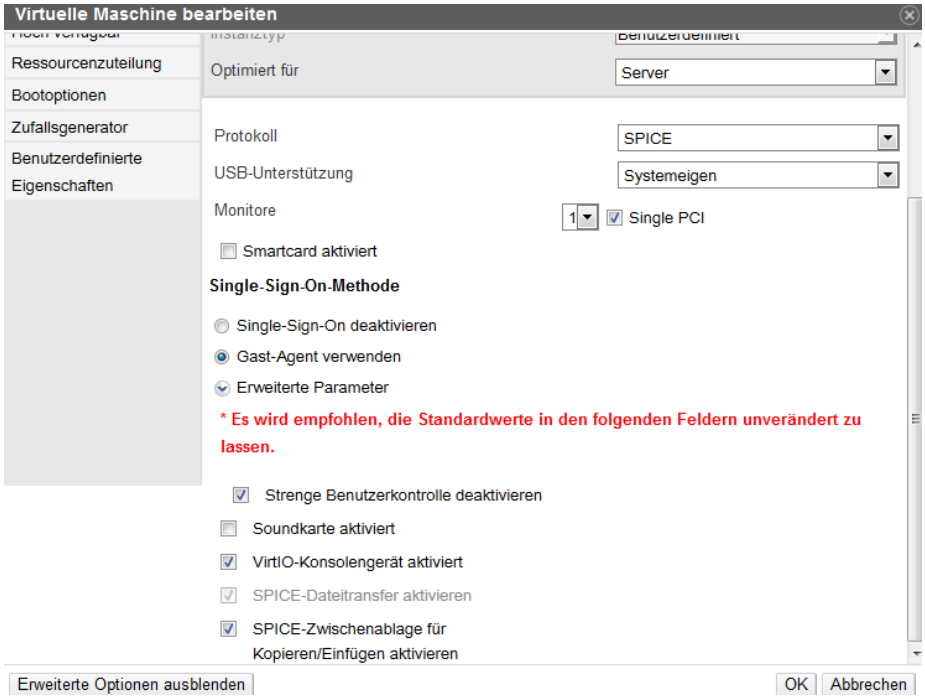



*Abbildung 74: Beginn mit dem Fine - Tuning: Vm - Erstellung; Kategorie Konsole – Beispiel InfraServ1*

#### 8.9.2.1.5 Kategorie → Host

Hier können die Position der VM bezüglich Host sowie die Migrationsrichtlinien definiert werden. Mittels **Ausführung starten auf** kann statisch ein Host innerhalb eines Clusters für den allgemeinen Start der VM definiert werden. Dies betrifft aber nur die Startsequenz, späteres automatisches Migrieren ist mit Funktionen wie **Affinitätsgruppen** weiterhin erlaubt. Es empfiehlt sich hier die Wahl oVirt selbst zu überlassen, da es selbst am besten beurteilen kann, wo der optimale Platz für diese Maschine im Produktivbetrieb ist. Der Punkt **Migrationsoptionen** definiert, ob diese Maschine überhaupt migriert werden kann und falls ja, unter welchen Umständen. Man kann hier zwischen nur manueller Migration und der kombinierten Migration, manuell und automatisch, wählen. Soll diese VM später HA unterstützen, so muss hier zwingend **Manuelle und automatische Migration erlauben**

gewählt werden. Es besteht auch noch die Möglichkeit, die Migrationsausfallzeit manuell einzutragen. Bei oVirt ist dies per Default deaktiviert und die Engine bestimmt diese selbstständig. Die reguläre Ausfallzeit beträgt bei KVM ca. 30 Millisekunden und muss auch von oVirt so übernommen werden. Falls man hier eingreifen will, so sollte man wissen, dass eine zu tiefe Zeit zu einem Migrationsabbruch im Fehlerfall führen kann, was wiederum zum Absturz der VM führt. Eine zu hohe Zeit macht bei ausgelasteten Systemen zwar auf den ersten Blick Sinn. Man muss jedoch bedenken, dass das installierte OS nicht virtuell ist. Zu langen Unterbruchszeiten können möglicherweise Startschwierigkeiten auf dem neuen Host verursachen, was ebenfalls zu einem Systemabsturz führen kann.

Die ganze NUMA – Thematik, wo es um das Fixieren von zusammengehörigen RAM Bausteinen zu einem CPU – Core geht, mag zwar die Fähigkeit besitzen, eine VM massiv zu beschleunigen, jedoch verunmöglicht sie die Migration der VM im Allgemeinen. Da dieser Punkt für sich selbst den Rahmen dieses Abschnittes und auch des ganzen Kapitels sprengen würde, wird hier auf eine weitere Vertiefung verzichtet.

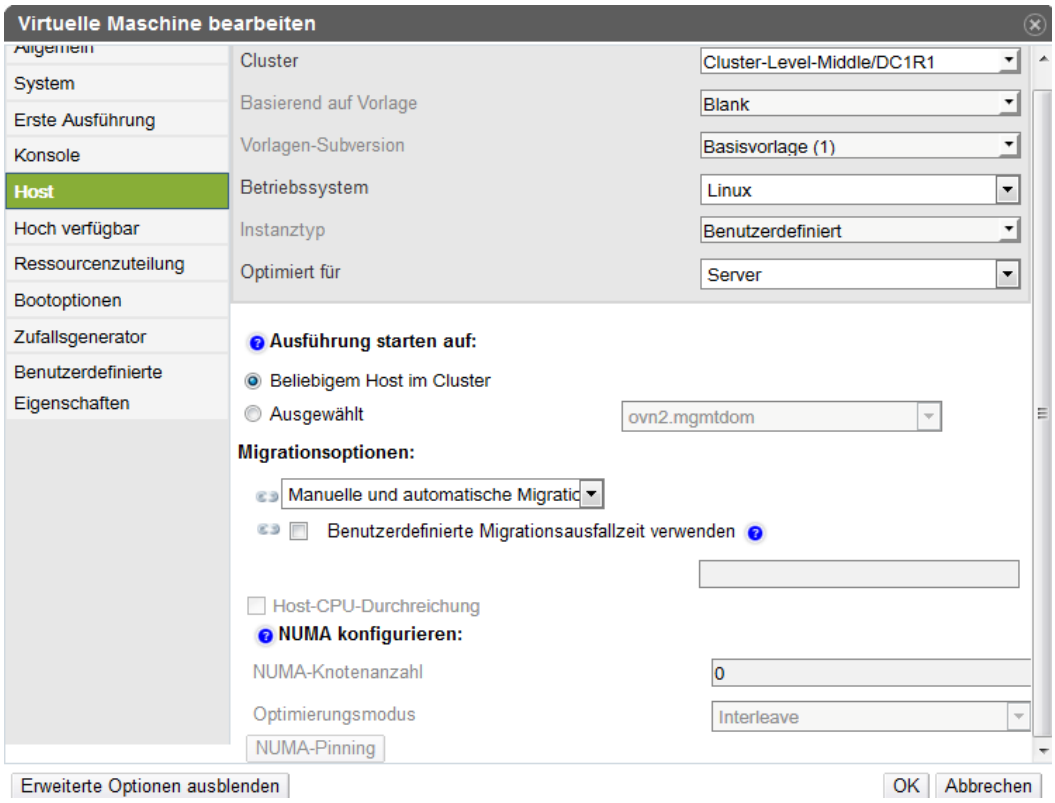



*Abbildung 75: Beginn mit dem Fine - Tuning: Vm - Erstellung; Kategorie Host – Beispiel InfraServ1*

#### 8.9.2.1.6 Kategorie → Hoch verfügbar

Hier kann definiert werden, ob eine VM das Anrecht besitzt, die Systemressource HA überhaupt zu nutzen oder ob sie im Notfall einfach fallen gelassen wird. Dazu genügt technisch gesehen nur das Aktivieren des Feldes **Hoch Verfügbar**. Es ist auch möglich zu definieren, mit welcher Priorität die Migration im allgemeinen aber auch die Migration im Notfall vollzogen werden soll. Hierfür stehen die drei Auswahlkriterien **Niedrig**, **Mittel** und **Hoch** zur Verfügung. Dies war auch schon die gesamte Konfiguration des HA für unsere VM InfraServ1.

Ovirt bietet auch an dieser Stelle die Möglichkeit, die VM selbst zu überwachen. Dazu muss lediglich in der Auswahlliste **Watchdog-Modell** unter der Parameterkategorie **Watchdog** ein verfügbares Modell ausgewählt werden. Unter **Watchdog-Aktion** kann die auszuführende Aktion **Zurücksetzten**, **Ausschalten**, **dump** oder **pausieren,** gewählt werden. Ist diese Option aktiv, so versucht der Host, welcher die Maschine gestartet hat, mit gelegentlichen Keepalive – Tests auf vorhandenen Traffic seitens CPU, RAM, Disk oder Netzwerk zu prüfen. Antwortet die VM nicht, aber der Zustand des Hypervisors ist als OK eingestuft, weiss er, dass etwas mit der VM nicht in Ordnung ist und führt eine der definierten Aktionen aus. Auf diese Funktion wird aber an dieser Stelle zwecks einer anderen Lösung, welche nachfolgend beschrieben wird, verzichtet.

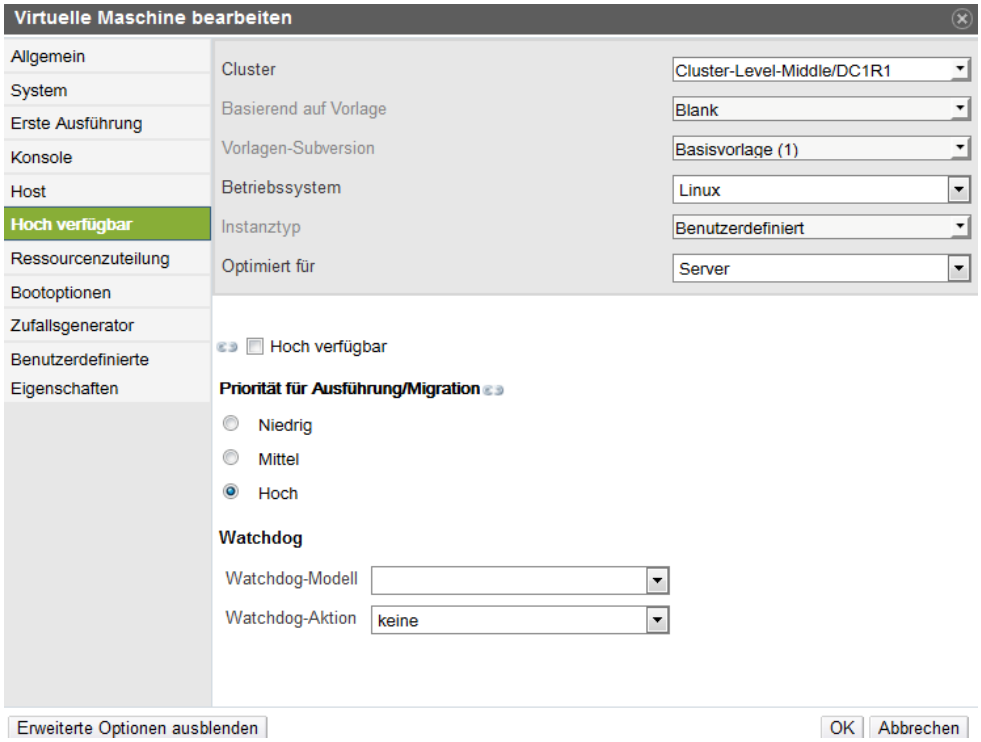



*Abbildung 76: Beginn mit dem Fine - Tuning: Vm - Erstellung; Kategorie Hoch verfügbar – Beispiel InfraServ1*

#### 8.9.2.1.7 Unterbruch zwecks eines kurzen Exkurses bezüglich Semi-Automatic-HA

Wie bereits erwähnt, ist die Realisierung des nativen HA mit der gegebenen Hardware nicht umsetzbar. Weswegen die Alternativlösung Semi-Automatic-HA defniert und entsprechend als Änderung im Betreuungsprotokoll 3 schriftlich festgehalten wurde. Diese Lösung sieht vor, dass ein Zabbix – Server alle VM's im Cluster überwacht. Auf diese Art und Weise kann erstens der Ausfall einer VM unverzüglich ermittelt werden, was die Nutzung des Watchdog überflüssig macht, und zweitens kann bei einem grossflächigen Ausfall davon ausgegangen werden, dass ein kompletter Node offline gegangen ist. Im Falle eines Node Ausfalles kann dann manuell eingegriffen werden, indem der sichtbar als **non-responsible** klassifizierte Node über das **Kontextmenü → Manuelles Fencing** als tot klassifiziert wird und der Cluster mit der Migration beginnt. Hier muss aber klar gesagt werden, dass die Konfigurationsparameter unter der Cluster – Konfiguration → Fencing-Richlinien nicht mehr gültig sind. Hat man also definiert, dass kein Fencing beim Ausfall von 50% des Clusters stattfinden soll, so wird dies ignoriert und die Engine versucht alles, was von der Priorität her möglich ist, auf die restlichen Maschinen desselben Clusters zu verteilen. Dies ist zwar keine Ideallösung, aber die technisch einzige ohne massive Investitionen in die vorhandene Hardware. Sie muss somit als akzeptable Lösung übernommen werden.

An dieser Stelle muss klar erwähnt werden, dass die Realisation und Dokumentation hier nicht stattfinden wird, sondern dies eine rein theoretische Definition einer möglichen Umsetzungsvariante darstellt.

Wie könnte aber eine mögliche Umsetzung aussehen? Ideal wäre eine externe Hardware, welche selbst nicht unter einem möglichen Node – Ausfall leiden würde. Eine mögliche und auch zugleich günstige Variante wäre der Einsatz eines Raspberry Pi 2 als Zabbix – Server. Dies ist technisch realisierbar und sollte von der Leistung vom Modell 2 her gesehen auch realistisch umsetzbar sein. Das Raspberry würde einen eigenen Netzwerkanschluss innerhalb des Virtualisierungsteils bekommen, wofür die pfSense Bridge um einen Port erweitert werden müsste. Auf jedem Client innerhalb des Clusters würde stets ein Zabbix – Client mitinstalliert, welcher für die Realisation dieses Projektes lediglich die Zabbix Ping Funktion (Zabbix Alive) nutzen müsste. Fällt eine VM oder ein Verbund an VM's innerhalb eines Nodes aus, so würde dies der Zabbix – Server registrieren und eine E – Mail an eine vordefinierte Adresse mit der Trigger – Meldung des betroffenen Items verschicken. Somit ist das sofortige Eingreifen des Administrators in jedem Fall gewährleistet und die Ausfallzeit lediglich um ein kleines Minimum grösser als es bei nativem HA der Fall wäre.

Eine absolut perfekte Alternativlösung wäre, den Zabbix – Server nicht eine Nachricht absetzen zu lassen, sondern gleich ein internes Skript auszuführen, welches die RESTful API der Engine anspricht und somit das Fencing manuell auslöst. Hierfür wäre aber eine Menge an Konfiguration- und Entwicklungsaufwand notwendig.

Zur besseren Verdeutlichung des Beschriebenen sollen die beiden nachfolgenden Abbildungen das Szenario mittels zwei Abläufen grafisch darstellen.

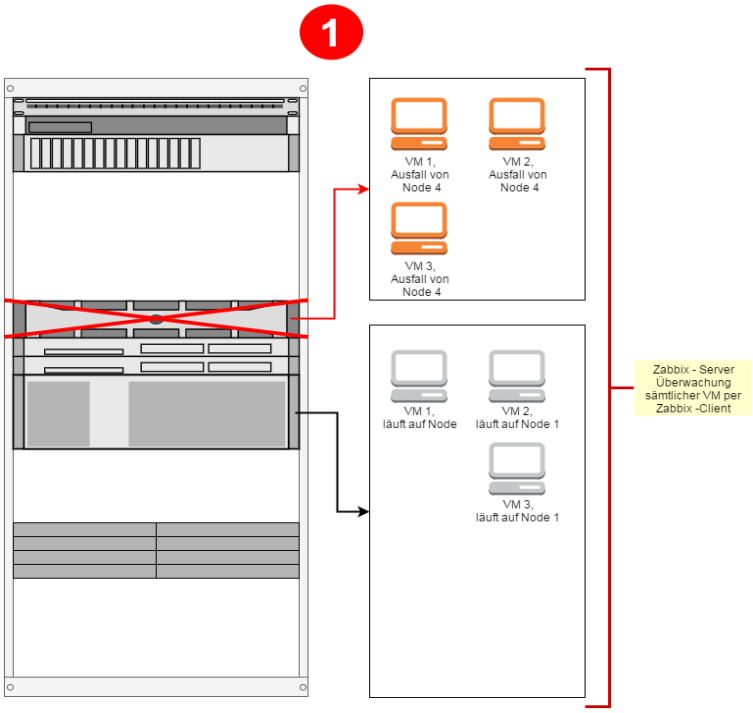



*Abbildung 77: Kurzer Exkurs nach Kapitel 8.9.2.1.7 - Semi-Automatic-HA Teil1*

Fehler wurde durch Zabbix – Server erkannt und gemeldet. Anschliessend wurde das manuelle Fancing durch den Admin gepusht. Die VM's wurden migriert und die Fehlersuche auf Node 4 kann beginnen.

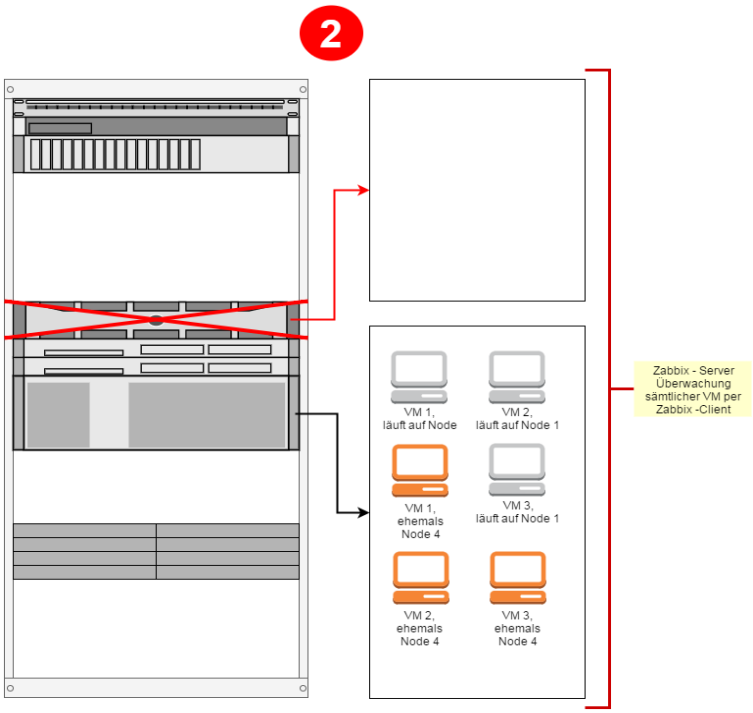



*Abbildung 78: Kurzer Exkurs nach Kapitel 8.9.2.1.7 - Semi-Automatic-HA Teil2*

#### 8.9.2.1.8 Kategorie → Ressourcenzuteilung

An dieser Stelle können die QoS – Limitierungen die wir erzeugt haben, gesetzt werden. Hierzu nutzen wir die Auswahlliste **CPU-Profil** und suchen uns das Profil mit dem Flag **_HIGH** im Namen aus. Es macht Sinn, diese hohe Priorität einem Infrastruktur – Server zu vergeben, da er permanent auf Broadcast – Nachrichten horchen muss und somit eine zu tiefe Priorität zum Überspringen der ankommenden Anfrage führen kann.

Die Auswahlliste namens **CPU-Anteile** definiert die möglichen Anteile pro Zeiteinheit die der virtuelle Prozessor pro Zyklus nutzen darf. Ein Prozessor kann auf einen prozentualen Anteil an zu nutzendem Maximum begrenzt werden. Es ist jedoch auch möglich, die Anteile an Recheneinheiten, die er in diesem Zeitraum nutzt, zu reduzieren, um auf diese Weise noch mehr VM's auf ein System zu bringen. Hier wird darauf verzichtet, da der Cluster nicht an seine thermische Obergrenze bezüglich Abluft gebracht werden soll.

Bei der **Arbeitsspeicherzuteilung** kann, sofern das Kästchen darunter (Memory-Balloon-Gerät aktiviert) aktiviert ist, ein tieferer Wert als der zugeteilte maximal RAM gesetzt werden. Unterstützt der Host ebenfalls das Ballooning, so kann der Hypervisor der Maschine RAM abzweigen um ihn einer anderen Maschine zuzuteilen. Das definierte Minimum wird dem Host aber auf jeden Fall garantiert.

Der Parameter **Disk-Zuteilung** sollte auf jeden Fall aktiv sein, da so ein paravirtualisierter Zugriff auf die virtualisierte Festplatte seitens der VM möglich ist und somit die Durchsatzrate erheblich steigt. Die Libvirt Treiber gibt es mittlerweile für eine Vielzahl an Betriebssystemen, einschliesslich Windows bis zur momentan aktuellsten Version.

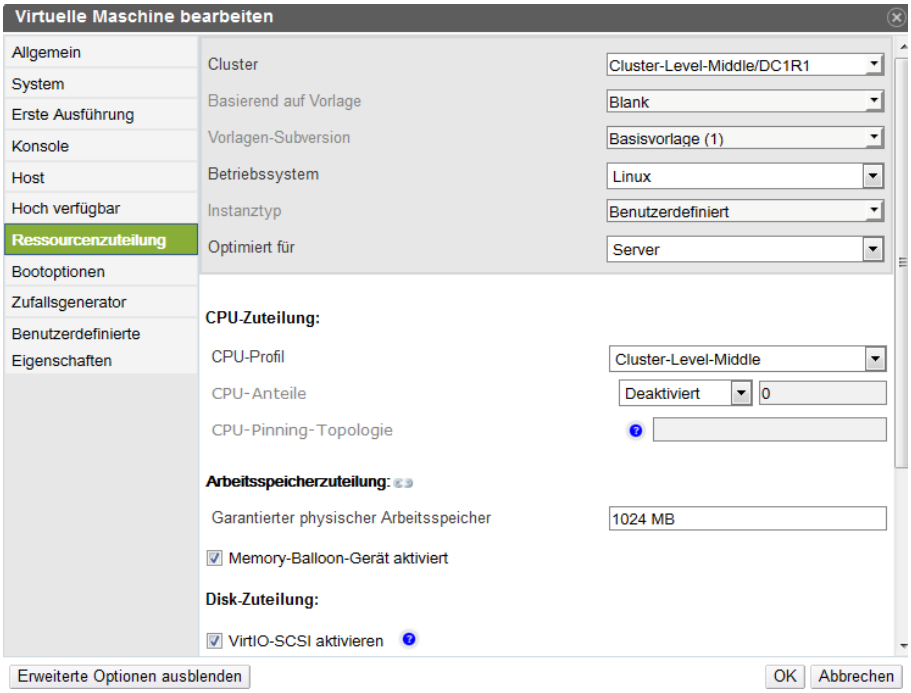



*Abbildung 79: Beginn mit dem Fine - Tuning: Vm - Erstellung; Kategorie Ressourcenzuteilung – Beispiel InfraServ1*

#### 8.9.2.1.9 Kategorie → Bootoptionen

Unter den **Bootoptionen** kann die Boot – Reihenfolge auf zwei Devices begrenzt eingestellt werden. Dabei folgt der Autor der Praxis, stets als erstes Device das CD – ROM Laufwerk zu setzen und erst anschliessend die eigentliche Festplatte. Auf diese Weise kann das Optionskästchen **Bootmenü aktivieren** deaktiviert bleiben. Somit hat man ein System, das stets von CD bootet und das ist bei der ersten Ausführung ja auch so gewollt. Denn unter **CD zuordnen** kann man nun das ISO – Image des openSUSE Netinstall auswählen und kann später das Optionskästchen einfach wieder deaktivieren, wodurch die CD – Ausführung beim booten unterbunden wird.

Die Möglichkeit des Kernel – Booting aus einem konstanten Pfad heraus für alle Maschinen (Linux-Bootoptionen) soll hier nicht thematisiert werden, da der Autor niemals ein Freund dieser Methode war und auch nicht vor hat sie für diese Arbeit zu nutzen.

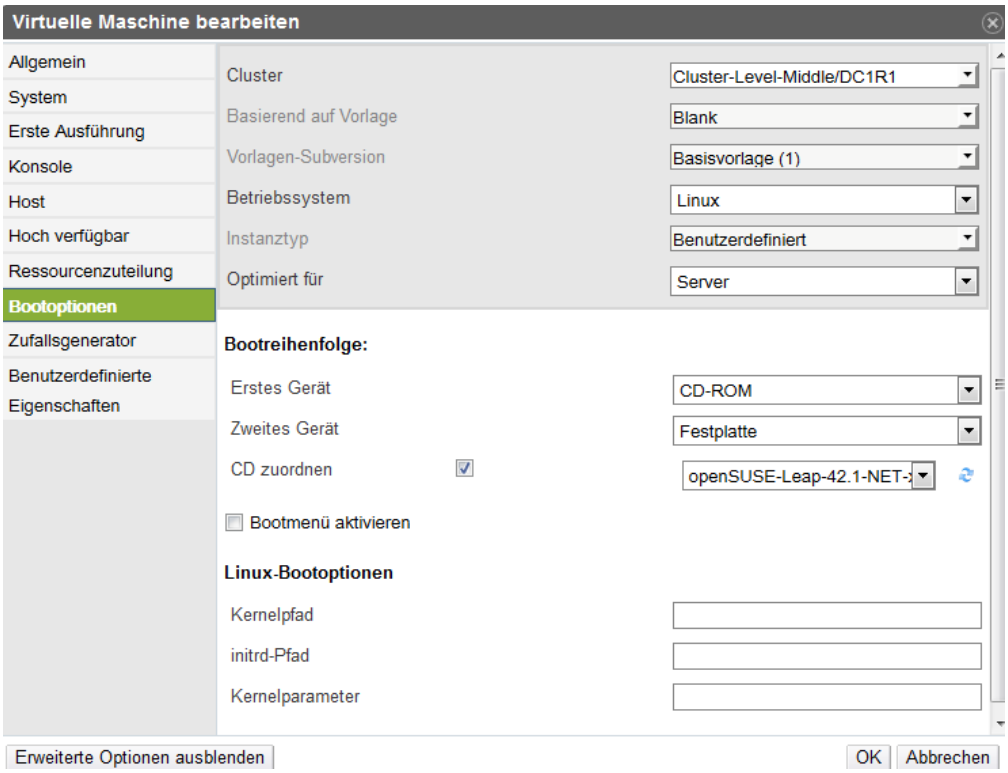



*Abbildung 80: Beginn mit dem Fine - Tuning: Vm - Erstellung; Kategorie Bootoptionen – Beispiel InfraServ1*

#### 8.9.2.1.10 Kategorie → Zufallsgenerator

Wie wir uns erinnern, haben wir den Zufallsgenerator innerhalb der Cluster deaktiviert, um nicht unnötig allen VM's diesen zur Verfügung zu stellen. An dieser Stelle können wir dies für einzelne VM's aber nachholen, falls es die Infrastruktur erfordert. Technisch gesehen müssten wir einen Berechnungszeitraum in Millisekunden und die Bytes pro Zeitraum definieren. Dies kann man sich aber sparen, da Libvirt alternativ seinen eigenen Standardwert bei einer Leereingabe verwendet. Diese sind absolut akzeptabel und die Berechnung für jeden Einzelfall wäre auch zu umständlich. Die Weiterleitung der Random – Werte des Host Devices **/dev/random** ist besonders bei der Verwendung

von Verschlüsslungssystemen auf Webservern zu empfehlen, da viele Prüfseiten wie bspw. ssllabs.com eine zu lange Latenzzeit als schlechte Key Exchange – Werte klassifizieren. Diese hohe Latenz entsteht, wenn die VM ohne Hilfe des Hosts selber die Berechnungen durchführen muss. Denn so muss sie jedes Mal den Host um Weiterleitung der Random - Werte von der CPU aus bitten.

Wir werden für unseren Infraserv1 keinen Zufallsgenerator benötigen. Die nachfolgende Abbildung zeigt jedoch, wie es im Falle einer Verwendung aussehen würden.

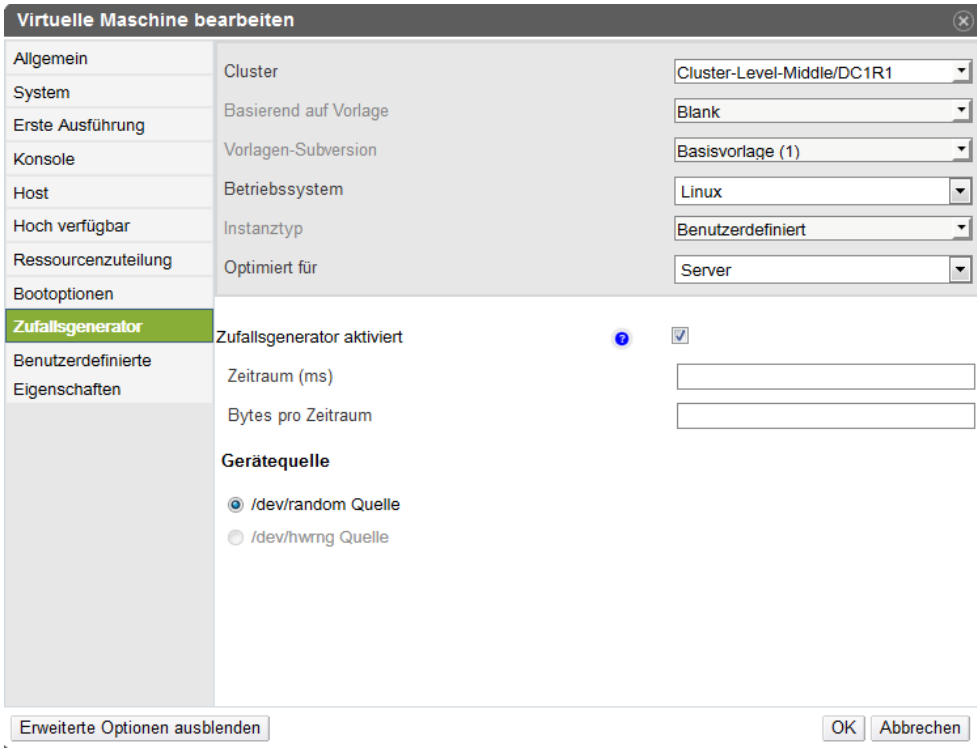



*Abbildung 81: Beginn mit dem Fine - Tuning: Vm - Erstellung; Kategorie Zufallsgenerator – Beispiel InfraServ1*

#### 8.9.2.1.11 Kategorie → Benutzerdefinierte Eigenschaften

Es handelt sich hier um einige wenige Parameter wie etwa die Verwendung des Disk – Caches, was ohnehin nicht möglich ist, da auch auf diese Weise die Live – Migration ebenfalls unterbrochen wird. Da diese Kategorie keine relevanten Parameter für diese Arbeit enthält, wird sie auch nicht weiter thematisiert.

Nach Erreichen dieser letzten Kategorie kann mit **OK** bestätigt werden und wir gelangen zum letzten Punkt der VM – Erstellung.

#### 8.9.2.1.12 VM – Erstellung, die virtuelle Disk

Nach Bestätigung der VM - Erstellung wird diese auch gleich erzeugt. Die Konfiguration ist hier aber noch nicht beendet. Gleich nach dem Bestätigen mittels **OK** im vorhergegangenen Schritt, werden wir zum Disk – Erstellungsassistenten weitergeleitet, wo wir noch eine, später vielleicht auch noch mehrere, Disks erstellen können.

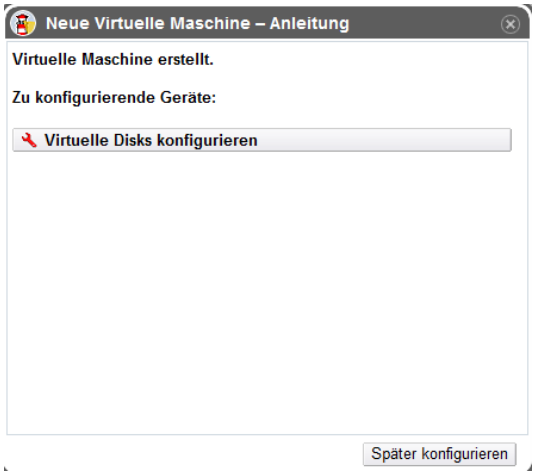



*Abbildung 82: Beginn mit dem Fine - Tuning: Vm - Erstellung; Erstellen einer virtuellen Disk – Beispiel InfraServ1*

Wenn wir hier auf **Virtuelle Disks konfigurieren** klicken, öffnet sich ein Fenster, wo wir als erstes die Grösse in GB setzen können. Um genügend Speicherplatz für spätere Nachinstallationen zu haben, aber auch weil openSUSE selbst in der Minimalinstallation stets etwas um 5 GB an Daten installiert, vergeben wir hier grosszügige **20 GB**. **Alias** und **Beschreibung** sind hier von geringer Relevanz, da oVirt im Hintergrund immer fix **UUID** als Namen speichert. Bei Linux, aber auch immer mehr bei Windows Installationen, lohnt es sich, hier als **Schnittstelle** den paravirtualisierten Treiber **VirtIO-SCSI** zu verwenden, da er die beste Performance bietet. Die **Zuteilungsrichtlinie** definiert, ob der allozierte Diskspace gleich 1:1 reserviert und beschrieben werden soll (Preallocated), oder ob nur Reservierung stattfinden soll (Thin Provision). Data – Center und Speicherdomäne sind ohnehin je nur einzeln vorhanden und per Default vordefiniert. Unter **Disk – Profil** kommt nun endlich unser QoS – Limitierungsprofil zum Einsatz, wo wir uns wieder für jenes mit dem **_HIGH** Flag entscheiden. Die **Auswahlkästchen** auf der rechten Seite sind bereits ideal gesetzt. Sie sollen dafür sorgen, dass erstens unsere neue Disk überhaupt aktiviert wird und zweitens, dass oVirt überhaupt versteht, dass sie als unser primäres Boot – Device dient, denn schliesslich könnten wir ja mehrere Disks haben.

Die zwei Auswahlkästchen **Gemeinsame Nutzung** und **Schreibgeschützt** sind im Produktivbetrieb nicht wirklich von grossem Nutzen. Wer gerne auf Nummer sicher geht, der kann aber das Kästchen **Bereinigen nach Löschen** aktivieren. Dies sorgt für eine sichere Löschung sämtlicher Daten des Images, um eine spätere Wiederherstellung zu verunmöglichen.

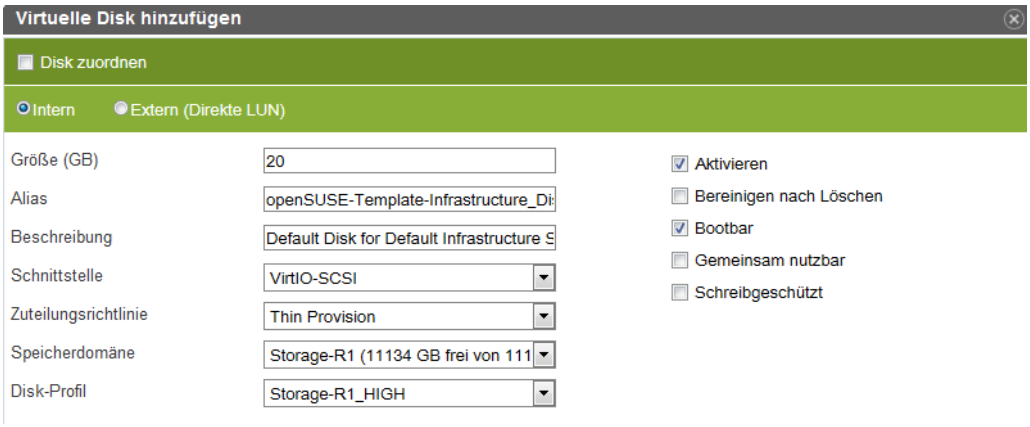



*Abbildung 83: Beginn mit dem Fine - Tuning: Vm - Erstellung; Erstellen einer virtuellen Disk; Assistent – Beispiel InfraServ1*

### 8.9.2.2 Die erste fertige VM

Dies waren auch schon alle Schritte der VM – Erstellung. In der Liste sollten wir nun unsere VM namens DYN_HIGH_InfraServ1 sehen und können sie auch gleich starten. Dies machen wir am einfachsten, indem wir die VM anklicken und im oberen Rand des Fensters auf das grüne Dreieck, welches nach oben zeigt, nochmals klicken. Die VM startet nun, was wir am grünen Dreieck neben dem VM – Namen sehen. Auf der gleichen Höhe wie das Startdreieck haben wir ein kleines Monitorsymbol welches uns ein *.vv File zum Herunterladen anbietet. Wenn wir dieses File herunterladen und öffnen, sollte unser bereits installierter SPICE – Client mit denen im File enthaltenen Informationen eine verschlüsselte grafische Verbindung zur VM erstellen können. Das File mit den Verbindungs-Informationen ist übrigens temporärer Natur und lediglich 120 Sekunden lang gültig.

Um die Art der Konsolenverbindung zu ändern, kann man über das **Kontextmenü → Konsolenoptionen** auch ein SPICE – Browser – Plugin oder einen SPICE-HTML5 Browser-Client (Technologievorschau) nutzen. Beide funktionieren aber nur unter Fedora 23 einigermassen gut.

Für Windows Freunde kann auf gleiche Art wie oben beschrieben noch auf Remote Desktop (RDP) umgestellt werden, wobei dann anstelle eines *.vv ein *.rdp File heruntergeladen werden muss.

Nun verbinden wir uns grafisch mit der VM und installieren ein openSUSE 42.1 Leap.

Bis später im nächsten Abschnitt.

### 8.9.3 Eine Vorlage erstellen

Nach Beendigung der Installation von DYN_HIGH_InfraServ1, wollen wir uns der Thematik der Vorlagen widmen. Eine VM bei Bedarf zu klonen, ist eine durchaus einfache, aber sicherlich Zeit und nervenaufreibende Angelegenheit. Besser ist es, gleich von seiner Lieblingsdirtribution eine Vorlage zu erstellen, welche dann immer wieder verwendet werden kann. Glücklicherweise ist dies bei oVirt eine simple Angelegenheit, die vollautomatisch im Hintergrund gemanagt wird. Hierzu muss die Maschine heruntergefahren werden. Danach kann man ins Kontextmenü der VM gehen und wählt dort den Punkt **Vorlage erstellen**. Im sich nun öffnenden Fenster kann man einen Namen vergeben. Hierfür definieren wir die folgende Namenskonvention:

TEMPLATE_<Name der Distribution + Version>_<Name welcher die Funktion beschreibt>

In unserem Fall heisst die Vorlage **TEMPLATE_openSUSE-42.1_Infrastruktur.** Das Infrastruktur am Ende definiert, dass diese Vorlage der internen Infrastruktur dient und für diverse Zwecke wie DHCP, DNS oder Directory Service dienen könnte. Die Beschreibung sollte an dieser Stelle etwas aussagekräftiger sein, da bei einer hohen Anzahl an Vorlagen schnell der Überblick verloren gehen könnte. Merkwürdigerweise muss nun ein Cluster definiert werden, für welchen die Vorlage gelten soll. Dies macht jedoch keinen Sinn. Die Vorlage Data – Center ist weit gültig und Versuche haben gezeigt, dass man die Vorlage von überall her erreichen kann. Wir geben trotzdem den Cluster-Level-Middle an und definieren anschliessend, welches CPU – Limitierungsprofil gelten soll. Auch hier wählen wir das höchste aus: QoS_CPU_HIGH. Wenn wir mehrere Vorlagen mit unterschiedlicher Software – Ausstattung hätten, so könnten wir hier das Optionskästchen **Als Vorlage-Subversion erstellen** anklicken und eine Unterversion erstellen die bspw. nur für Samba – Freigaben dienen würde. Wir bleiben aber bei unserer Basisvorlage und sehen unten eine Tabelle, in welcher die Disk bereits ermittelt wurde. Auch hier können wir den Diskzugriff limitieren, indem wir unter Disk-Profil **Storage-R1_HIGH** wählen, was wir natürlich auch tun. Zuunterst steht uns mit einem Optionskästchen noch die Möglichkeit offen, die Vorlage allen anderen Benutzern des Data – Centers zur Verfügung zu stellen. Die Option **VM – Berechtigungen kopieren** sollte man nicht nutzen. Dies sollte immer einzeln konfiguriert werden.

Dies war es eigentlich schon. Die Vorlage wird nun ca. während 5 Minuten erstellt und am Ende ist sie unter folgendem Weg ersichtlich:

```
System (Seitenleiste) → Vorlagen (Registerkarte)
```

*Befehl 34: Realisierung des Virtualisierungssegmentes (Klicken): Der weg zur ersten Vorlage*

**TIPP**

**Vorsicht bei den Limitierungen der Vorlagen. Man kann noch einige Werte wie RAM oder Kernanzahl der CPU bearbeiten, doch die Limitierungen sind fix beim Erstellen der Vorlage integriert worden.**

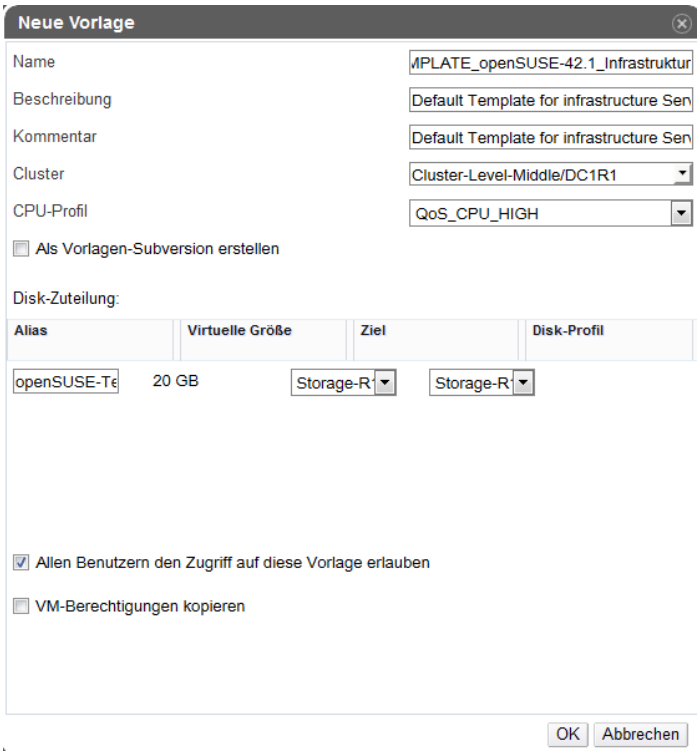



*Abbildung 84: Beginn mit dem Fine - Tuning: Vorlage - Erstellung aus InfraServ1*

Möchte man nun eine VM aus der Vorlage erstellen, so kann man den Prozess wie unter 8.9.2.1 „ Die einzelnen Konfigurationsschritte der VM – Erstellung“ initialisieren und muss unter **Basierend auf Vorlage** lediglich die gewünschte Vorlage aus der Auswahlliste auswählen und einen neuen Namen samt Beschreibung vergeben. Den Rest erledigt oVirt selbstständig, indem es alle relevanten Parameter aus der Vorlage liest. Diese Konfiguration kann man 1:1 übernehmen, um eine identische VM zu erhalten, oder man passt die Parameter seinen Wünschen an.

Das war es eigentlich auch schon mit den Vorlagen. Nun erstellen wir einige neu VM's für den nächsten Teil und auch gleich für die späteren Tests.

### 8.9.4 Positive und negative Affinitätsgruppen

Die Affinitätsgruppen sind ein ideales Werkzeug, wenn man gewisse VM's, welche vielleicht eine logische Verbindung eingehen, untereinander verankern will. So macht es Sinn, dass wenn man bspw. einen MariaDB Cluster mittels Galera bilden möchte, nicht gerade alle Maschinen auf dem gleichen Node arbeiten. Zu diesem Zweck werden wir mit den neu erstellten Testmaschinen **DYN_HIGH_openSUSE-42.1_DA-VM-(1 – 3)** und dem bestehenden **Infraserv1** versuchen, je zwei VM's auf Cluster-Level-Middle voneinander zu trennen und auf dem grossen Cluster je zwei auf einen Node zu bringen. Dann werden wir in diesem Abschnitt eine neue Cluster – Richtlinie erzeugen, welche wir auf die beiden Cluster **-Middle** und **-High** sharen werden. Anschliessend werden wir die eigentlichen Affinitätsgruppen bilden, welche dann aufbauend auf unsere neue Cluster – Richtlinie, als Abhängigkeit verweisen werden.

#### 8.9.4.1 Erzeugen der neuen Cluster – Richtlinie

Als erstes müssen wir eine neue Cluster – Richtlinie erzeugen. Dabei muss man wissen, dass dies erst ab dieser Version so ist. Bis Version 3.4 mussten Scheduling – Richtlinien erzeugt werden, welche von oVirt mühsam alle 20 – 60 Sekunden ausgeführt werden mussten. Dies hat sich ab dieser Version geändert und es gibt nur noch einen oVirt internen Scheduler, welcher nur noch von Trigger – Werten angestossen wird. Solche Trigger – Werte sind heute von Filtern und deren Gewichtungen abhängig, welche in Richtlinien definiert werden. Somit ist es an dieser Stelle notwendig, eine neue Cluster – Richtlinie zu erzeugen, was wir auch mit nachfolgendem Klick – Weg einleiten wollen:

```
Konfigurieren (Systemleiste zu oberst am rechten Browser – Rand) → Cluster – Richtlinien (Kategorie)
```

*Befehl 35: Realisierung des Virtualisierungssegmentes (Klicken): Zum Konfigurationsmenü der Cluster – Richtlinien*

Wenn wir dies getan haben, sehen wir eine Liste mit in dieser Arbeit noch nicht genannten Konfigurationsnamen. Was aber nicht ganz stimmt, denn einen Namen davon haben wir bei der Erstellung der Cluster schon einmal gesehen und zwar die Richtlinie **none**, welche wir als Defaultwert belassen haben. Technisch gesehen müssen wir lediglich einen Parameter in dieser Richtlinie ändern. oVirt eigene Konfigurationsparameter lassen sich jedoch prinzipiell weder ändern noch löschen, was der Grund für eine Neuerstellung ist. Glücklicherweise können wir aber **none** mittels Klick auf **Kopieren** in der darüber liegenden Button – Leiste einfach klonen und nachträglich bearbeiten. Wenn wir diesen Prozess initialisieren, öffnet sich das Konfigurationsfenster. Dort werden sämtliche Parameter von „none“ angezeigt. Als erstes vergeben wir einen Namen und eine Beschreibung . In diesem Fall folgende Parameter:

Name: **DC1R1**

Beschreibung: **Default Cluster-Polices for this Data-Center**

Analysieren wir nun die vorhandenen Abschnitte innerhalb des Konfigurationsfensters Schritt für Schritt:

- **Filtermodule:** Hier können die logischen Filter gesetzt werden, welche jede VM beim Starten als Prüfabhängigkeiten durchlaufen muss, um zu ermitteln, ob gewisse Prüfparameter gesetzt

sind. Dabei gibt es Prüfpunkte wie bspw. CPU und RAM, welche von oVirt selbst abhängig sind und wo oVirt bei fehlerhaften Parametern gleich selbst eingreift, ohne dabei auf andere logische Abhängigkeiten zu prüfen. Dann gibt es Punkte, welche technisch gesehen nicht notwendig sind und oVirt auch nicht weiss, ob es diese mit in die Prüfung nehmen muss. Ein solcher Punkt ist die Prüfung auf vorhandene Affinitätsgruppen. Man muss also einen zusätzlichen Filter in den Prüfalgorithmus mitaufnehmen, um oVirt darauf hinzuweisen, dass dieser auch eine Abhängigkeit darstellt. Unser Filter an dieser Stelle trägt den Namen **VmAffinityGroups** und ist als Defualt – Wert in der geklonten Richtlinie bereits enthalten. Die Flags **Erster** und **Letzter** innerhalb der Profilauswahl haben nichts mit der Prüfreihenfolge zu tun, womit hier ein Nachsortieren wegfällt.

- **Gewichtungsmodule:** Hier werden die Prüfmodule zu den einzelnen Filtern gesetzt. Die Reihenfolge dieser ist für die spätere Logik von grösster Bedeutung, d.h wie oVirt prüfen soll und was Vorrang haben soll. Unser Modul zum Filter trägt den gleichen Namen und ist zu unterst in der Liste. Dies macht im Normalfall auch Sinn, da die HA Module technisch gesehen immer Vorrang haben sollten. Da wir ja aber kein vollautomatisches HA besitzen, können wir hier die Reihenfolge auch ändern. Hierfür kann man die Nummerierung auf der linken Seite neben dem Modulnamen nutzen. Etwas umständlicher: Alle Module nach rechts versieben (Deaktivierte Gewichtungen) und sie in der richtigen Reihenfolge wieder in den aktiven Bereich ziehen. Hier gilt der Grundsatz: Haben alle die gleiche Nummer als Ausführungspriorität, so entscheidet die Reihenfolge.
- **Lastverteiler**: Hier kann definiert werden ob eine Lastverteilung innerhalb der Cluster stattfinden soll. Falls ja: Soll sie allgemein gültig sein, soll sie nur für Gäste (externe Provider) gelten oder soll, sogar stromsparend mittels shutdown gearbeitet werden. Wir werden hier den allgemeinen Modus wählen und wollen somit, dass Maschinen migriert werden, falls ein gewisser Grenzwert erreicht wird. Dieser Punkt ist die eigentliche Änderung, welche hier angestrebt wurde, da ansonsten alle hier vorgenommenen Arbeiten nur für die simple Prüfung des VM – Starts anwendbar wären.
- **Eigenschaften:** Hier können noch eigene Parameter übergeben werden, welche sich auf die Lastteilung auswirken. Wir werden hier zwei Grundparameter setzen, welche sich später innerhalb eines Clusters im Notall noch überschreiben lassen. Die Parameter, die wir hier ergänzen, sind folgende:
  - **HighUtilization** → Besagt für den angewendeten Cluster, was die Grenzwerte seiner Hosts sind und wann der Trigger für den interen Scheduler greifen soll. Hier definieren wir den Parameter 70 und teilen dem Cluster mit, dass er bei 70% Auslastung eines Hosts reagieren muss, ausser der nachfolgende Wert trifft zu.
  - **CpuOverCommitDurationMinutes** → Dieser Wert ist auf 3 gesetzt. Er erlaubt ein Überschreiten des für den Host gesetzten Grenzwertes für 3 Minuten. Dies ist absolut in Ordnung, da in 3 Min. keine grösseren Temperaturerhöhungen bei 70% zu erwarten sind.

Haben wir alles eingetragen, so können wir das Fenster mit **OK** schliessen.

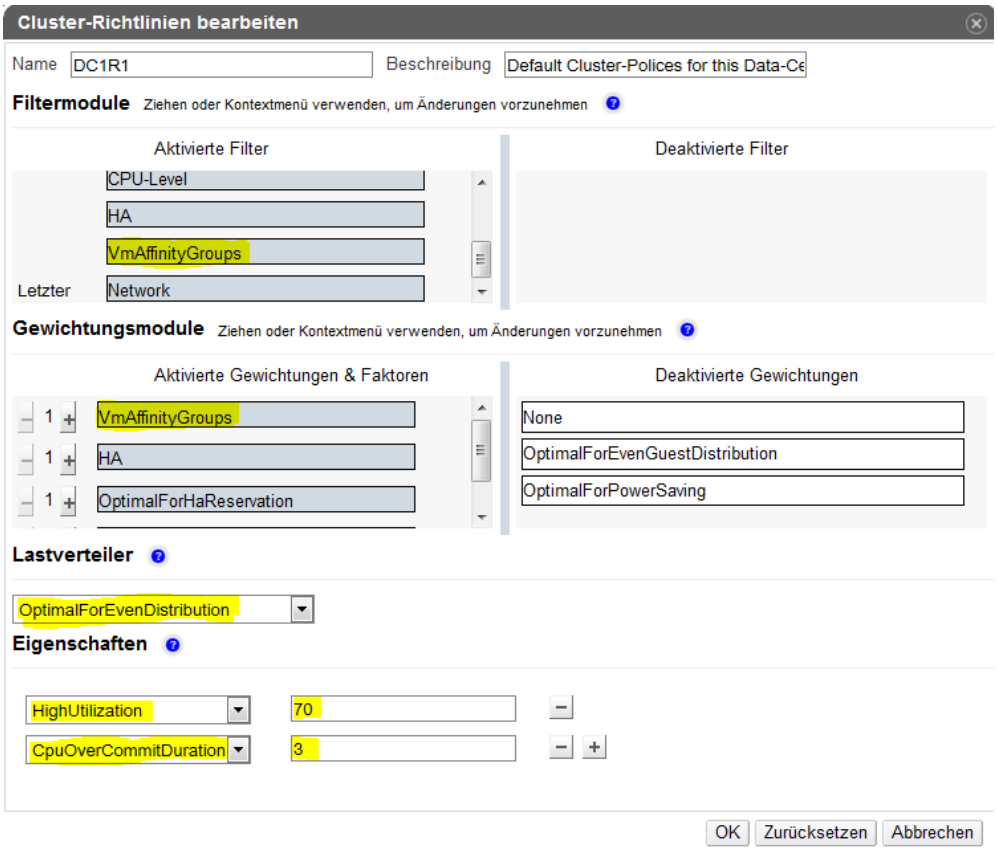



*Abbildung 85: Beginn mit dem Fine - Tuning: Erstellen einer Cluster - Richtlinie (DC1R1)*

Nun haben wir eine neue Cluster – Richtlinie und müssen diese noch auf den Clustern verteilen. Hierfür gehen wir auf Cluster-Level-High als Referenzbeispiel:

System (Seitenleiste) → Cluster (Registerkarte) → Cluster-Level-High (Kontextmenü) → Bearbeiten → Cluster-Richtlinien (Kategorie)

*Befehl 36: Realisierung des Virtualisierungssegmentes (Klicken): Einrichten der Cluster – Richtlinie auf Cluster-Level-High.*

Hier sehen wir zu oberst die Auswahlliste **Richtlinien wählen**, wo wir unsere neue Richtlinie DC1R1 auswählen und mit OK bestätigen. Das war es auch schon.

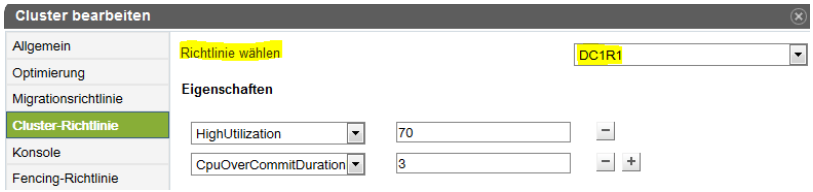



*Abbildung 86: Beginn mit dem Fine - Tuning: Verteilen der neuen Cluster - Richtlinie*

Diesen Schritt wiederholen wir nun auch für Cluster-Level-Middle.

Technisch gesehen haben wir hier alles für die Einrichtung der Affinitätsgruppen vorbereitet. Aufgrund der schrittweisen Verschmelzung der einzelnen Funktionen in oVirt, haben wir mit nur drei kleineren Paramteren auch gleich das gesamte Ressourcenmanagement für das Data – Center mit konfiguriert. Somit haben wir nun ein Gebilde, das pro Cluster selbstständig die CPU – Auslastung beobachtet und bei Überschreiten der zulässigen 70% auf 3 Minuten mit der Live – Migration der Maschinen beginnt. Dabei achten die Cluster speziell auf die Affinitätsgruppen und versuchen, dies so gut wie möglich in positiver oder negativer Polaritätsform in Abhängigkeit zu bringen.

Doch an dieser Stelle weiss oVirt gar nicht, nach welchen Gruppenrichtlinien es überhaupt die Maschinen behandeln muss. Also konfigurieren wir die Gruppenabhängigkeiten nun im nächsten Schritt.

### 8.9.4.2 Erzeugen der Affinitätsgruppen

Das Bilden von Affinitätsgruppen ist technisch gesehen nur das Erzeugen von Listen, in welchen definiert wird, welche Maschinen dort Mitglieder sind und wie ihre Polarität zueinander sein soll. Zu diesem Zweck gehen wir zum folgenden Konfigurationspunkt und werden eine Referenzkonfiguration an Cluster-Level-Middle erzeugen:

```
System (Seitenleiste) → Cluster (Registerkarte) → Cluster-Level-Middle (Anklicken) → Affinitätsgruppen (untere Registerkarte)
```

*Befehl 37: Realisierung des Virtualisierungssegmentes (Klicken): Bilden der ersten Affinitätsgruppe.*

In der leeren Liste treffen wir den altbewährten Button **Neu** an, den wir anklicken und uns dem Konfigurationsassistenten widmen. Wie üblich vergeben wir einen Namen und eine Beschreibung nach folgendem Schema:

**Name →** AG_N_DA; wobei hier das „N“ für negative Polarität steht. Wir wollen also, dass die sich in dieser Liste befindlichen Maschinen negativ zueinander verhalten, was im Klartext bedeutet, dass sie sich vorläufig nicht auf dem gleichen Host aufhalten dürfen.

**Beschreibung →** Ist frei wählbar, muss aber bei dieser Komplexität doch noch irgendeinen Sinn ergeben.

Das erste Optionskästchen definiert, ob es sich hier um eine positive Affinität handelt, d.h. ob die Maschinen zusammen auf einem Host gehalten werden sollen oder, ob sie wie in unserem Fall negativ ist. Das Optionskästchen **Erzwingen** differenziert intern zwischen **Hard** und **Soft**. Ist es aktiv (Hard), so bedeutet dies, dass diese Affinität auf jeden Fall eingehalten werden **muss.** Es ist belanglos, ob nun HA aktiv wird, um einen Notfall abzuarbeiten oder der Admin eine manuelle Migration von Hand oder per Wartungsmodus initiiert. Deaktiviert man aber das Erzwingen, so kann man manuell ohne weiteres migrieren, auch wenn dies zu einem Affinitätsbruch führt. HA würde ebenfalls die Regeln etwas biegen können, um im Notfall einen einigermassen konsistenten Zustand zu erreichen, jedoch gilt dies speziell für diese Arbeit nicht. Wie wir uns erinnern, haben wir bei der Erstellung der neuen Cluster – Richtlinie die Gewichtung für das Modul VMAffinityGroups zu oberst platziert und somit das HA ausgehebelt. Dies fällt aber nicht weiter ins Gewicht, da das vollautomatische HA hier ohnehin

nicht funktioniert und das manuelle Fencing solche Richtlinien im allgemeinen ignoriert. Lediglich die Gewichtungen von Modulen wie CPU und RAM behalten ihre Wertigkeit und Dringlichkeit bis zum Schluss.

Im darunterliegenden Auswahlfeld kann nun die erste Maschine ausgewählt werden und dann die nächste mittels „+“ usw. Die Liste umfasst dabei alle im Cluster befindlichen Maschinen und wird auch mit jeder getroffenen Auswahl entsprechend kleiner. In unserm Fall wählen wir DYN_HIGH_InfraServ1 und **DYN_HIGH_openSUSE-42.1_DA-VM-1** aus. Wir wollen ja in diesem Testbeispiel nicht, dass unser DHCP / DNS Server durch eine andere Maschine gestört wird.

Dies war es auch schon. Wir haben den Clustern mit der neuen Cluster – Richtlinie gesagt, wie sie sich verhalten sollen und jetzt mit der Affinitätsgruppe, wie es aussehen soll. Sollten noch Maschinen in einem gebrochenen Zustand bezüglich ihrer Affinität sein, so müssen diese jetzt neu gestartet werden, um sich der neuen Affinität anzupassen. Prinzipiell lohnt sich eine Affinitätskonfiguration immer vor dem Starten spezieller Maschinen.

Für den **Cluster-Level-High** definieren wir an dieser Stelle gleich im Voraus die Affinitätsgruppe, welche wir später für die Tests benötigen werden. Dabei wird nach Ablauf der gerade beschriebenen Referenzkonfiguration folgendes an Parametern übergeben:

**AG_P_DA:**

- Positiv → aktiv
- Erzwungen → aktiv
- Host 1 → **DYN_HIGH_openSUSE-42.1_DA-VM-2**
- Host 1 → **DYN_HIGH_openSUSE-42.1_DA-VM-3**

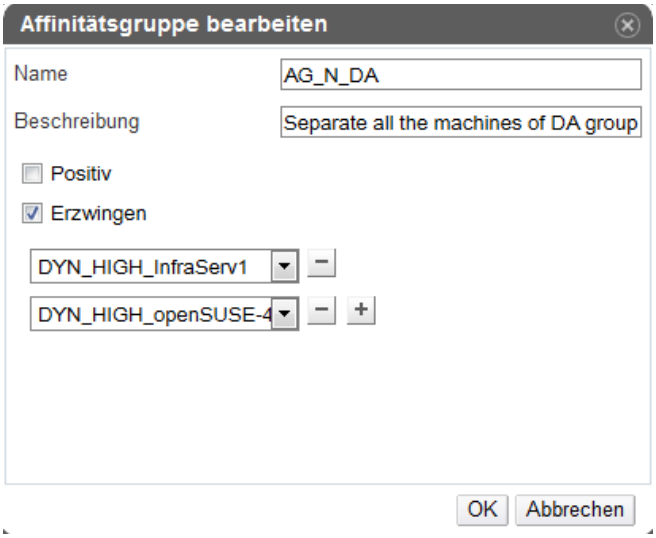



*Abbildung 87: Beginn mit dem Fine - Tuning: Erstellung einer Affinitätsgruppe für Cluster-Level-High als Referenzbeispiel.*

### 8.9.5 Exkurs zur Asymmetrie des Aufbaus

Nun haben wir bis hierhin viel über Lastverteilung und halbautomatisches HA gehört, jedoch noch nie konkretisiert, wie es genau umgesetzt werden kann. Bei einem korrekten Aufbau wären hier vier Hosts mit der gleichen ressourcentechnischen Ausstattung, was die Berechnung des HA Designs vereinfachen würde. Betrachten wir hierzu folgende Grafik:

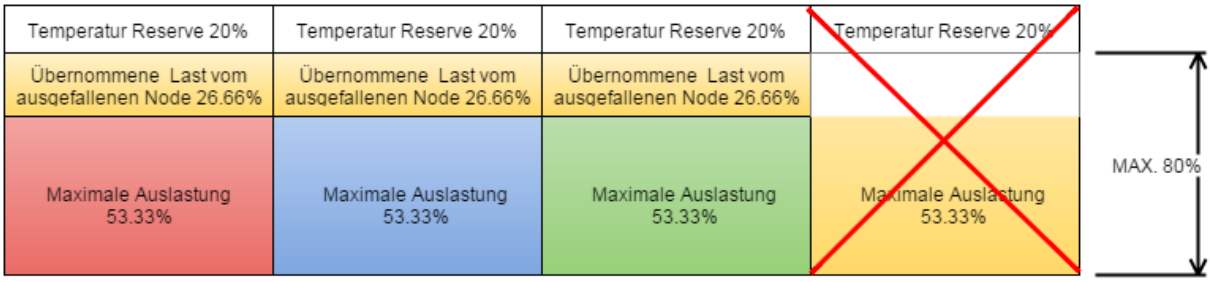



*Abbildung 88: Exkurs zur Asymmetrie des Clusters: Ein perfekt symmetrischer Cluster.*

Unter der Annahme, dass nur ein Node ausfallen dürfte, sehen wir, dass bei einem perfekt symmetrischen Cluster nur eine Betriebslast von 53,33% möglich wäre. Dies resultiert aus der Tatsache, dass eine Temperatur – Reserve von 20% mit eingerechnet werden muss, um den Prozessor vor Überhitzung zu schützen. Somit ergibt sich ein Maximum von 80% je Node, was durch drei geteilt werden muss im Falle eines Node Ausfalles. Hieraus resultiert die Last von 26,66%, welche von jedem Node im Notfall getragen werden müsste ohne selbst über seine 80% - Grenze zu gelangen. Somit kann bei grosszügiger Abrundung gesagt werden, dass die durchschnittliche Last je Node gerade einmal die Hälfte seiner Kapazität darstellt. Wie gesagt, dies wäre die optimale Betriebsumgebung. Sehen wir uns dies einmal mit den beiden Nodes des Clusters Cluster-Level-High an, welcher klar asymmetrisch aufgebaut ist.

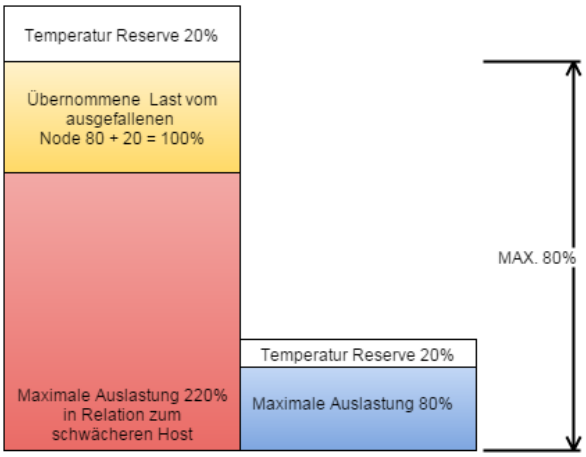



*Abbildung 89: Exkurs zur Asymmetrie des Clusters: Die einfachste Lösung*

Dies wäre die einfachste Lösung, da mathematisch gesehen der Dell den gesamten Sumpermicro 4HE inklusive seiner Temperatur – Reserve aufnehmen könnte. Dabei entspricht sein Betriebsmaximum in

Relation zur vierfachen Leistung zum Supermicro 4HE, 220%. Mit dieser Variante könnten theoretisch 80% der VM's über Semi-Automatic-HA gerettet werden bis der Supermicro 4HE wieder betriebsbereit wäre. Was aber wenn der Dell ausfällt? Aufgrund der Leistungsdiskrepanz ist eine hundertprozentige Rettung der VM's nicht möglich. Definiert man aber eine fixe Anzahl an VM's mit hoher Priorität und verteilt diese auf beide Hosts, so hat man mit Beschränkung auf diese fixe Maximalanzahl dennoch die Möglichkeit, 80% der VM's sicher zu betreiben. Dies würde ungefähr wie in nachfolgender Abbildung aussehen:

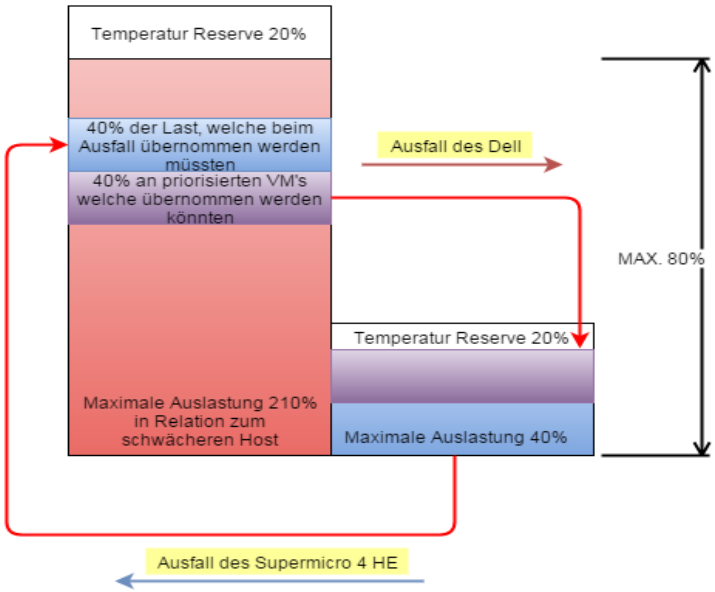



*Abbildung 90: Exkurs zur Asymmetrie des Clusters: Die etwas kompliziertere Variante*

Bei dieser Variante könnte der Dell mit 210% unprivilegierten VM's arbeiten und müsste 40% der VM's vom Supermicro halten, was nach dem umrechnen ca. 10% für den Dell bedeuten. Damit hätte der Dell, vom Supermicro 4HE aus gesehen, 70% seiner regulären Leistung und könnte 10% (40% für den Supermicro 4HE) für ihn halten. Bei einem Ausfall des Dell wären somit seine umgerechneten 40% + die 40% des Supermicro 4HE wieder 80% an möglichen VM's die HA fähig wären. Reduziert man aber die Leistung des Dell um weitere 10% (40% vom Supermicro 4 HE gesehen), kann man eine zusätzliche Reserve schaffen, welche den Ausfall des Supermicro 4 HE wiederum abfangen könnte. Somit würde der Dell nach Umrechnung mit 60% seiner maximalen Leistung arbeiten und der Supermicro 4 HE bei realer Betrachtung mit 40%, womit technisch gesehen eine hohe Leistungsausbeute erzielt wird, aber dennoch ein 80%-er Teil an VM's da wäre, welcher HA fähig bleiben könnte. Dies ist augenscheinlich die beste Lösung und wird mit 90% Wahrscheinlichkeit auch umgesetzt.

Die der Cluster Cluster-Level-Middle wird an dieser Stelle nicht thematisiert. Er ist symmetrisch und kann beim Abzug der 20% für die Temperatur – Reserve ohnehin nur noch 40% je Node halten. Dies ist bei 8 GB RAM zu wenig, weswegen hier vermutlich auf HA verzichtet wird.

Bezüglich der Realwerte an möglichen VM's wurde hier bewusst auf Zahlen verzichtet. Erst im

Produktivbetrieb kann die Zahl an VM's schrittweise definiert und somit genannt werden. Der Beginn des Produktivbetriebs ist aber erst nach der Fertigstellung und der Präsentation dieser Arbeit. Jedoch werden auch dann die Prozentual – Werte ausschlaggebend sein für die exakte Berechnung der VM – Zahlenwerte, womit die Einheit Prozent hier ohnehin der ausschlaggebende Faktor ist.

## 8.10 User und Rechtemanagement

Prinzipiell wird für den Produktivbetrieb seitens des Autors keine User- und Rechteverwaltung benötigt, da ja der einzige Admin der Autor selbst ist. Um aber die Integrität dieser Arbeit als spätere neutrale Bedienungsanleitung zu gewährleisten, soll an dieser Stelle eine kurze Einleitung stattfinden, welche die wichtigsten Schritte behandelt. Vorab muss aber gesagt werden, dass oVirt in der Version 3.5 die vier Authentifizierungsdienste **ad** (Microsoft Avtive Directory), **oldap** (openLDAP), **ipa** (RedHat's eigene Kreation freeIPA) und **rhds** (389 Directory Server) unterstützt. Es besteht noch die Möglichkeit **itds** (IBM Security Directory Server) einzubinden, doch ist diese kommerzielle Variante innerhalb dieser stark open source- lastigen Arbeit keine Alternative. Ebenfalls wichtig zu erwähnen ist die Tatsache, dass für alle Varianten ein Kerberos Server notwendig ist. Hier musste aber mit Schwierigkeiten bezüglich Kerberos gekämpft werden, da keine manuelle Konfiguration mit der Engine (Java Code) zusammenarbeiten wollte, selbst eine eins zu eins übernommene Referenzimplementierung von FreeBSD brachte keinen Erfolg. Lediglich die Directory Implementation von Microsoft konnte hier problemlos angebunden werden. Da hier aber keine Microsoft Produkte verwendet werden sollen, ist als alternative eine Zentyal 4.2 Installation zum Einsatz gekommen, welche mittels Samba 4 einen Microsoft AD emuliert. Diese Wahl ist als Temporärlösung für diese Arbeit ideal, da sie schnell installiert ist und eine bequeme Möglichkeit der GUI – basierten Konfiguration bietet.

In diesem Teil soll in zwei Abschnitten erklärt werden wie eine Domäne in die Engine eingebunden wird und wie die User nachfolgend ins System integriert werden bzw. auf welche Weise ihnen welche Rechte erteilt werden.

### 8.10.1 Domäne in oVirt einbinden

Wir verwenden an dieser Stelle einen „Active Directory“ Authentifizierungsserver, dessen Spezifikationen an dieser Stelle nicht näher thematisiert werden. Entscheidend sind dabei lediglich folgende Parameter:

- Domain → **localdom.lan**
- Domain Admin → **ovirtadm**
- Server → **dc1.localdom.lan**

An dieser Stelle ist es notwendig, sich per SSH mit der Engine zu verbinden, da nachfolgender Befehl auf der Konsole eingegeben werden muss:

```
# engine-manage-domains add --domain=localdom.lan --provider=ad --ldap-servers=dc1.localdom.lan
--user=ovirtadm
```

*Befehl 38: User und Rechtemanagement (Klicken): Einbinden einer Domäne zur Authentifizierung per Active Directory.*

Dieser Befehl umfasst dabei folgende Logik:

- **--domain** → Hier wird die Domäne eingetragen, welche der AD betreut.
- **--provider** → Ist die Art des Authentifizierungsdienstes, welcher verwendet werden soll.
- **--ldap-servers** → Ist hier speziell notwendig, da der AD im Netzwerk virtdom steht. Ein Aufruf des Befehls ohne diesen Parameter bewirkt eine Suche nur innerhalb der Broadcast – Domäne.
- **--user** → Ist der Domain Administrator Account, welcher benötigt wird, um das Directory auszulesen.

Ist dieser Befehl abgesetzt muss anschliessend in interaktiver Form noch das Admin – Passwort eingegeben werden. Nach einer gefühlten Ewigkeit meldet der Konfigurationsbefehl, dass die Domäne hinzugefügt wurde, aber die User keine Rechte besitzen würden. Diese müsse man mit dem Parameter --add-premissions noch nachholen. Dies ist aber absolut nicht notwendig und kann später über das GUI auf bequeme Art nachgeholt werden. Als letzten Schritt müssen wir nur noch den Application – Server neu starten. Dies geschieht mit nachfolgendem Befehl:

```
# service ovirt-engine restart
```

*Befehl 39: User und Rechtemanagement (Klicken): Neustart des Application – Server zur Domänen – Einbindung.*

Dies was es auch schon mit der Domänen – Integration. Sie ist nun eingebunden, kann aber noch nicht genutzt werden. Kommen wir nun zum zweiten Schritt und somit zur eigentlichen Einbindung der einzelnen AD – User.

### 8.10.2 Einbinden der einzelnen User aus dem AD

Nun werden wir in einem ersten Schritt den User ovirtadm zu Demonstrationszwecken den ins Management – System der Engine holen. Dafür klicken wir uns auf folgende Weise zum Einbindemenü.

```
System (Seitenleiste) → Benutzer (Registerkarte)
```

*Befehl 40: User und Rechtemanagement (Klicken): Der Weg zum Einbinden eines neuen Users aus der Domäne.*

An dieser Stelle sehen wir den per Default von oVirt bei der Installation erzeugten Administrations – Account admin. An dieser Stelle ist noch der Account Everyone anzutreffen, dessen Funktion nicht ganz klar ist und auch nicht über die oVirt Dokumentation ermittelt werden konnte. Wir klicken nun am oberen Rand der User – Liste auf Hinzufügen und gelangen zum Konfigurationsfenster. Hier können wir nun in der ersten Auswahlliste **Suchen** unsere Domäne localdom.lan auswählen. Das Feld **Namensraum** beinhaltet nur ein Sternchen und hat keine Auswahloptionen, da wir nur eine Ebene innerhalb unsers AD's haben. In der **Textbox** daneben können wir unsere Suche einschränken, indem wir den Usernamen eingeben. Dies lohnt sich, da das Standard – Directory von Zentyal doch einige Einträge beinhaltet. Nachdem wir alles entsprechend ausgewählt und eingegeben haben, können wir auf **LOS** klicken und sehen nun unseren User als einzigen Eintrag in der Liste. Nun müssen wir nur den

User auswählen und mittels OK bestätigen, was ihn als neutralen SystemUser in die Engine einbindet.


*Abbildung 91: User und Rechtemanagement: Auswählen von ovirtadm aus der Domäne.*

Der User erscheint nun in der Liste mit einem neutralen Icon und einer Beschreibung seiner Domänen – Zugehörigkeit. So nützt uns der User aber im Moment nicht viel, da er keinerlei Rechte besitzt und seine Rolle gegenüber oVirt nicht genau definiert ist. Um dies zu ändern wählen wir folgendes:

Konfiguration in der rechten Seite oben am Browserrand.

*Befehl 41: User und Rechtemanagement (Klicken): Beginn mit der Rollen- und Rechtevergabe.*

Hier klicken wir als erstes auf die Kategorie **Systemberechtigungen** und sehen eine Liste mit den beiden Usern admin@internal (Rolle → **SuperUser**) und admin@internal (Rolle → **PowerUser**). Dies bringt uns auch gleich zum wichtigsten Punkt in oVirt, nämlich der Trennung von Administratoren und Usern. Fügen wir einen User als Administratorrolle ein, so kann er beim Login sowohl ins Administrationsportal als auch ins Benutzerportal eintreten. Fügen wir jedoch einen User als Benutzerrolle ein, so ist ihm nur der Login ins Benutzerportal möglich. Dies ist von Vorteil wenn man eine grössere Menge an Kunden besitzt, welche Virtualisierungsdienste in Anspruch nehmen. Um unseren User ovirtadm eine Systemberechtigung zu vergeben, klicken wir am oberen Rand des Fensters auf **Hinzufügen.** Es öffnet sich hier ein Fenster wie in Abbildung 91, wo wir wieder gleich vorgehen wie oben beschrieben. Dieses Fenster ist aber nicht absolut identisch mit dem in Abbildung 91. Im unteren Bereich ist eine Auswahlliste namens **Zuzuweisende Rolle**, wo wir uns für die Rolle **SuperUser** entscheiden. Auf diese Weise können wir einen zweiten Administrations – Account eröffnen und somit den internen Admin – Account als reinen Backup zurückstufen, wofür er theoretisch auch vorgesehen ist. Als wir aber in der Liste nach **SuperUser** suchten, haben wir gesehen dass eine Vielzahl an Rollen vorhanden war. Was sind das alles für Rollen? Man findet sie in der Nachbarkategorie **Rollen**, wo sie alle aufgelistet sind. Hier finden sich Rollenname wie Cluster-, Host- oder NetworkAdmin, welche im Grunde genau die Funktion haben, welche auch im Namen vorkommt. Man erkennt ebenfalls auf den ersten Blick, dass es blaue und grüne Symbole gibt. Auf diese Farben kann man sich auch verlassen, da sie einer oVirt internen Konvention folgen, wobei blaue Administrations- und grüne Benutzerrollen darstellen. Möchte man selbst neue Rollen erstellen, so genügt ein Klick auf **Neu.** Man landet in einem neuen Konfigurationsfenster, wo man definieren kann, ob die neue Rolle einen Admin- oder einen Benutzerstatus haben soll. Danach kann man je nach vorheriger Vorwahl aus einer breiten Liste an Optionen wie System, Vorlagen, Disk oder VM, mit unzähligen Unterkategorien wählen. An dieser Stelle soll nicht weiter auf diesen Bereich eingegangen werden, da es weit mehr Optionen gibt als Platz innerhalb dieser Arbeit.

**TIPP**

**Das Erstellen von zusätzlichen Rollen ist meistens nicht notwendig, da die Systemvorlagen bereits alle erdenklichen Szenarien eines Clusterbetriebes abdecken. Bei neuen Rollen besteht zudem auch das Risiko, dass man einen Teil des grossen Ganzen übersieht und die eigene Rolle mehr zulässt, als man eigentlich wollte.**

Unser Beispiel mit dem Account **ovirtadm** auf der Rolle **SuperUser** zeigt einen der grössten Vorteile innerhalb von oVirt, nämlich das automatische Setzen der Berechtigungen der VM's beim Erstellen. Technisch gesehen müsste eine Rolle nur die elementarsten Funktionen wie Schreibrechte auf Storage oder allgemein das Recht auf VM – Erstellung haben. Meldet man sich nämlich als User am Administrationsportal an und erstellt eine VM, so hat man automatisch die vollen Berechtigungen und das Eigentum an besagter VM. Dies gilt aber nur für domänenabhängige Accounts wie unser **ovirtadm** aus **localdom.lan**. Der interne Administrator ist, wie bereits erwähnt, nur ein Backup – Account, welcher im Notfall, wie bspw. dem Verbindungsverlust zum AD, zum Einsatz kommen sollte.

Dies war es eigentlich auch schon mit der User – Einbindung inklusive Rechtevergabe. Wir haben an dieser Stelle ein simples Beispiel mit einer hoch privilegierten Superuser – Rolle gesehen. Theoretisch gibt es eine Vielzahl an möglichen Variationen wo man mit Admin- und Benutzerrollen spielen kann. Dies hängt stark mit dem Einsatzzweck und Einsatzgebiet von oVirt zusammen An dieser Stelle kann nicht mehr gezeigt werden, als das absolute Minimum für einen durchschnittlichen Einsatzzweck.

## 8.11 Vorläufiges Fazit zur Realisation

Ab diesem Punkt haben wir einen funktionierenden Virtualisierungscluster welcher die wichtigsten, aber auch die im Pflichtenheft definierten Funktionen erfüllt. Es wäre theoretisch eine einfache Realisation gewesen, jedoch kam es wegen der schlechten Dokumentation seitens des oVirt Projektes immer wieder zu kleineren Problemen, welche die Realisierung in die Länge zogen. Zum Zeitpunkt des Schreibens dieser Zeilen kam noch erschwerend hinzu, dass das oVirt Projekt seine Webseite umstrukturierte und sie dem aktuellen Release 3.6 angepasst hat. Somit ist die Suche nach Konfigurationshilfen ab Version 3.3 bis 3.5 heute noch schwieriger als noch zu Beginn der ersten Vortests. Dennoch wurde aus Sicht des Autors ein akzeptables Ergebnis erzeugt, welches für den späteren Produktivbetrieb absolut geeignet ist.

# 9 Verifizierung der Funktionsfähigkeit

An dieser Stelle soll mit Tests die Funktionsfähigkeit wie beschrieben und in Pflichtenheft definiert, erwiesen werden. Hierzu wird wie bereits im Abschnitt Realisation angedeutet, in zwei Testsektionen hervorgehoben. In der ersten Testsektion wurden primär gröbere Tests durchgeführt, welche sich mit dem Trennen von Kommunikationsverbindungen untereinander befassen. Hier wird klar auf die teilweise Ausfallsicherheit von Komponenten wie bspw. die netzwerkseitige Trennung der Storages oder das reale Vorhandensein von LACP im VM – Netzwerkbereich getestet. Der zweite Teil befasst sich mit Parametertests wie bspw. Ressourcenbegrenzung, Ermittlung des Verteilens der Daten über den GlusterFS Storage oder die Berechtigungen des Usermanagements.

Zur besseren Übersicht werden die Messprotokolle in drei Sektionen unterteilt. Sie werden stets mit einer tabellarischen Beschreibung des Testszenarios beginnen, gefolgt vom Mittelteil welcher den grafischen Nachweis des Tests enthält und mit einem ebenfalls tabellarisch strukturierten Abschlussbericht, welcher den Status des Tests enthält enden.

## 9.1 Testumgebung

Die Testumgebung entspricht dem fertigen Aufbau des Virtualisierungsclusters, welcher innerhalb dieser Arbeit realisiert wurde. Siehe dazu nachfolgende Abbildung.

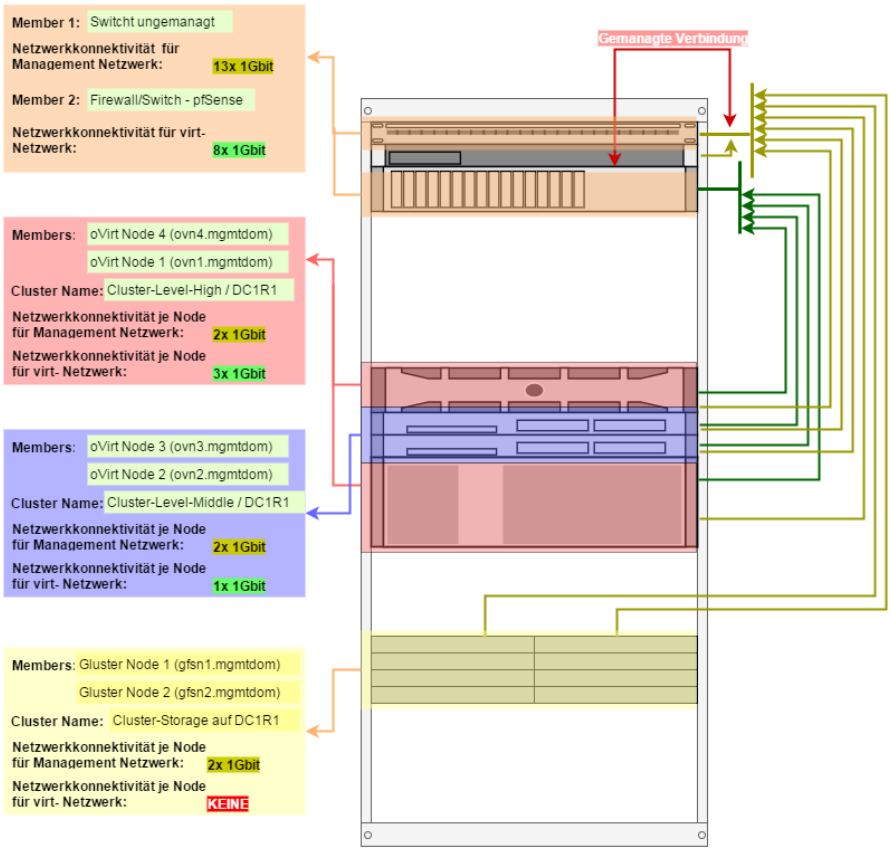



*Abbildung 92: Tests: Die Testumgebung in grafischer Form.*

## 9.2 Testreihe 1 Definitionen

Nachfolgend sind die Parameter und der eigentliche Testablauf der Testreihe 1 beschrieben. Die hier definierten Punkte sind notwendig für die Umsetzung der Tests. Kleinere Änderungen welche innerhalb der Tests als notwendig erachtet werden, um eine bessere grafische Darstellung der Ergebnisse zu ermöglichen sind denkbar. Es wird aber darauf geachtet, dass diese Abweichungen zu den unten definierten Punkten keinen Einfluss auf das reale Ergebnis haben. Sie werden bei einem möglichen auftreten dennoch spezifisch innerhalb der Testprotokolle deklariert.

### 9.2.1 Parameter für die Testreihe 1

Innerhalb der Grobtests werden folgende Parameter an VM's und Software verwendet:

- **STATIC_debian_Testsys_1**: Keine Ressourcenbegrenzung für CPU, Netzwerk und Storage
- **STATIC_debian_Testsys_2**: Keine Ressourcenbegrenzung für CPU, Netzwerk und Storage
- **STATIC_debian_Testsys_3**: Keine Ressourcenbegrenzung für CPU, Netzwerk und Storage

Diese VM's wurden vor Beginn des Fine – Tunings  erzeugt und besitzen keinerlei Begrenzungen im Ressourcenbereich. Sie haben lediglich den simplen Zweck virtuelle System für diverse Testszenarien bereitzustellen.

Innerhalb dieser Testreihe wird hauptsächlich **tcpdump** zum Einsatz kommen, welches die Netzwerkkonnektivität im Unterbruchfall nachweisen soll. Bei gewissen Test werden auch Kommandozeilen – Programme zum Einsatz kommen, welche diverse Zustände grafisch nachweisen sollen. Diese Tools sind aber von herkömmlicher Natur und werden nur bei speziellem Bedarf genauer erklärt. Der Grossteil der grafischen Nachweisbarkeit soll aber über die Statusmeldungen von oVirt selbst erfolgen, da es auch im Realbetrieb die erste Anlaufstelle bei Störungen sein wird.

### 9.2.2 Notwendige Tests für die Testreihe 1

Innerhalb dieser Testreihe sollen die nachfolgend genannten Test durchgeführt werden. Sie sollen die Ausfallsicherheit der Netzwerkinfrastruktur, das Verhalten bei Node (Engine) Ausfällen sowie die Initialisierung des HA mittels manuellem Fencing prüfen und bestätigen.

- Verhalten beim Ausfall eines Bonding – Ports bei diversen Nodes (Management Netzwerk).
- Verhalten beim Ausfall mehrerer Bonding – Ports im VM – Netzwerk.
- Verhalten beim Initialisieren des manuellen Fencing in Bezug auf Semi-Automaitc-HA.
- Neustart der Engine um das Verhalten des Clusters zu ermitteln.

## 9.3 Testreihe 2 Definitionen

An dieser Stelle gelten die gleichen Rahmenbedingungen und Pflichten wie unter Punkt 9.2.

### 9.3.1 Parameter für die Testreihe 2

Für die Tests aus der Reihe 2 wurden die VM's bereits innerhalb der Realisation unter Kapitel 8 bereits erzeugt. Diese werden für Testreihe 2 teilweise umkonfiguriert, um gewisse Test durchführen zu können. Hierbei handelt es sich hauptsächlich um die Änderung der zu nutzenden Ressourcen, welche zu folgendem Bild führen:

- **DYN_HIGH_openSUSE-42.1_DA-VM-1:**
  - CPU → Level _HIGH
  - Netzwerk → Level _HIGH
  - Speicher → Level _HIGH
- **DYN_HIGH_openSUSE-42.1_DA-VM-2:**
  - CPU → Level _HIGH
  - Netzwerk → Level _HIGH
  - Speicher → Level _HIGH
- **DYN_HIGH_openSUSE-42.1_DA-VM-3:**
  - CPU → Level _MIN
  - Netzwerk → Level _MIN
  - Speicher → Level _MIN

Die zu verwendenden Testtools sind umfangreich und werden in den entsprechenden Testprotokollen genauer definiert. Auch an dieser Stelle stellt das oVirt GUI einen elementaren Teil des grafischen Nachweises an erwarteter Funktionalität dar.

### 9.3.2 Notwendige Tests für die Testreihe 2

Hier sollen mit Schwerpunkt Virtualisierung die Fähigkeiten von oVirt als Management Engine getestet werden. Innerhalb dieser Testreihe sollen die Fähigkeiten von oVirt in Bezug auf Sicherheit, die Fähigkeit auf Fehlererkennung und Ressourcenbegrenzung getestet und protokolliert werden. Zusätzlich soll innerhalb dieser Testreihe auch die Isolationsfähigkeit der Regelsätzte von pfSense getestet werden.

Die Strukturierung des Prüfungsablaufs orientiert sich an dieser Stelle wieder dem allgemeinen Grundaufbau dieser Arbeit. Somit werden die Tests der Reihenfolge Netzwerk, Virtualisierung und Storage folgen. Aufgrund der Verschmelzung der einzelnen Komponenten zu einem Cluster ist jedoch ein minimales Abweichen innerhalb der Reihenfolge, zwecks Erhalt der logischen Zusammenhänge durchaus möglich. Diese Testreihe soll an dieser Stelle folgende Prüfungen enthalten:

- Netzwerk:
  - Nachweis der Switching – Funktionen der Bridge.
  - Prüfen der Isolation der einzelnen Netzwerke untereinander.
  - Prüfung auf Anwendung von netzwerkseitigem QoS seitens oVirt.
- Virtualisierung:
  - Prüfung auf Anwendung von CPU- seitigem QoS seitens oVirt.
  - Minimaler Nachweis der Live – Migration inklusive des Verhaltens bei Umschaltung auf Maintenance Modus.
  - Prüfung auf Anwendung der definierten Affinitätsgruppen seitens oVirt.
- Storage:
  - Erbringung des Nachweises auf tatsächliche Verteilung der virtuellen Disks auf beide Nodes. Dies Schliesst die ISO – Files mit ein.
  - Prüfung auf Anwendung von Storage- seitigem QoS seitens oVirt.

## 9.4 Expliziter Ausschluss von möglichen Tests

Einige Test sollen an dieser Stelle klar von den Testreihen ausgeschlossen werden. Das prüfen dieser Punkte ist technisch nicht notwendig, da sie meistens in Abhängigkeit mit anderen zu prüfenden Punkten steht oder die Prüfung ein hohes Risiko mit sich bringt einen schwerwiegenden und irreparablen Schaden zu erzeugen. Die wichtigsten hier explizit ausgeschlossenen Tests sind:

- Test auf Ausfallsicherheit der beiden RAID 5 – Verbünde auf den GlusterFS Nodes; (Risiko eines nicht wiederherstellbaren Schadens ist zu hoch.)
- Tests mit der kompletten Trennung eines GlusterFS Node vom Cluster sind nicht notwendig, da zwischen den Nodes keine Ausfallsicherheit besteht; (Risiko eines irreparablen Schadens ist zu hoch.)
- Auslastungstests mit einer überdimensionierten CPU Anzahl einer VM im allgemeinen, aber auch beim Migrieren sind nur schwer realsierbar, da KVM an dieser Stelle eine hohe virtualisierungsanzahl an virtuellen CPU's unterstützt. Hier genügen RAM bezogene Tests, da das Überschreiten des maximalen RAM's deaktiviert wurde.
- Test der möglichen Maximalauslastung der einzelnen Nodes währen denkbar jedoch nicht repräsentativ, da dies von einem klaren Design der maximalen Ressourcenzuteilung der VM's abhängt, hierfür aber noch keine Design – Entscheidung getroffen wurde und auch nicht expliziter Bestandteil dieser Arbeit ist.

## 9.5 Beginn der Testreihe 1

Diese Testreihe orientiert sich stark am Ausfallmanagement der Bereiche Netzwerk und Virtualisierungsnodes. Dabei werden die Ausfälle teils mit netzwerkseitigem Trennen der Ethernet Anschlüsse und teils mit direktem „harten“ Ausschalten der Nodes simuliert. Die nachfolgende Grafik soll mögliche Trenn- bzw. Ausschaltpunkte aufzeigen.

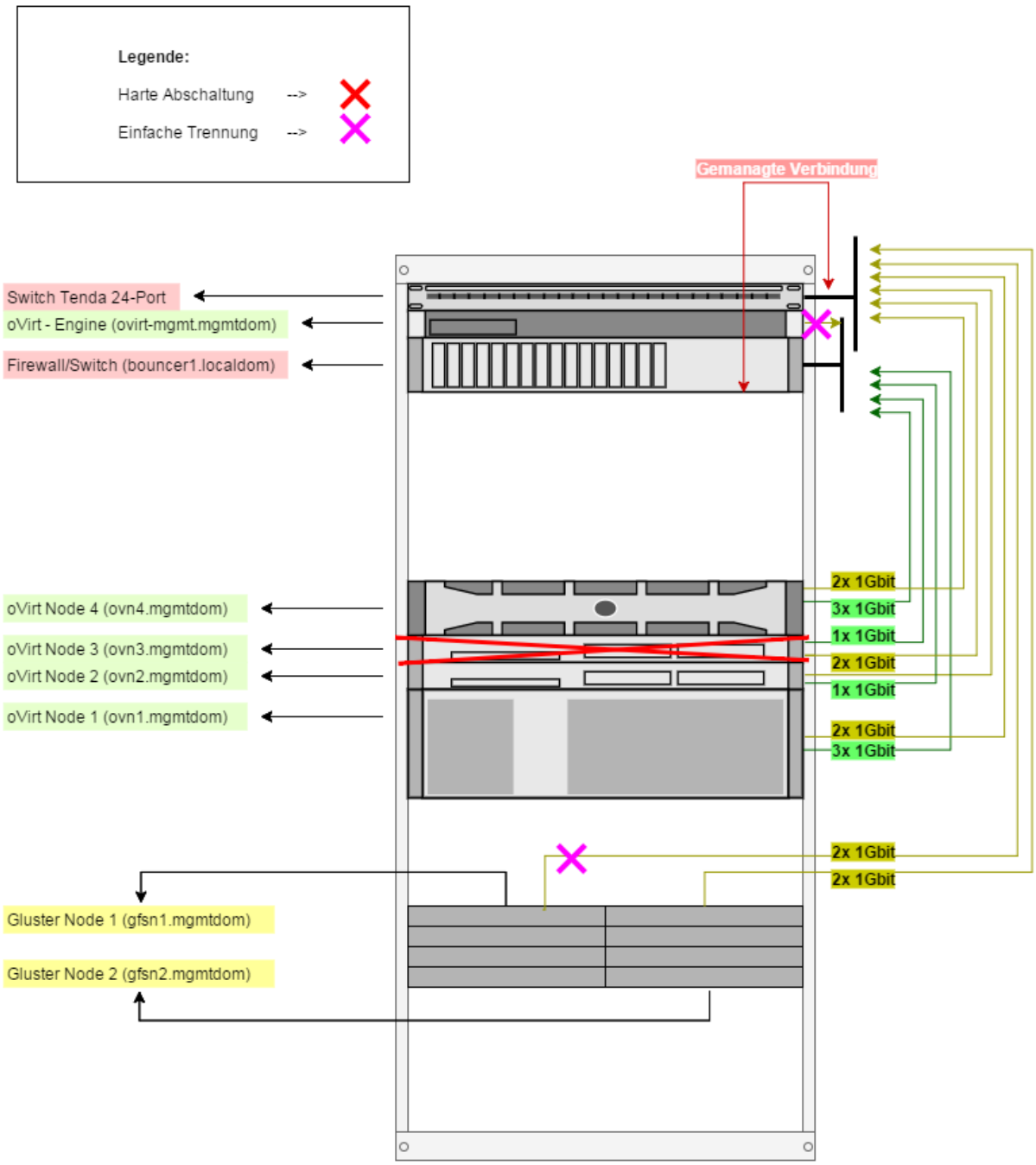



*Abbildung 93: Tests, Testreihe 1: Das Trenn- bzw. Abschaltschema*

## 9.5.1 Testprotokoll: Trennen von Ethernet – Port (XOR – Mgmt – Net)

### 9.5.1.1 Testprotokoll: Trennung von XOR – Bonding Port 1 bei gfsn1.mgmtdom

| | |
|---|---|
| **Testname:** TR1-T0001 | **Abhängigkeiten:** Keine speziellen, oVirt + Node GlusterFS 1 |
| **Parameter:** Cluster noch im Leerlauf, keine VM's aktiv. | **Verwendete Tools:** Keine speziellen, oVirt selbst |
| **Ablauf des Tests:** Der Cluster ist in Betrieb, es wird der Ethernet – Port 1 des Storage – Nodes gfsn1.mgmtdom entfernt. | |
| **Messpunkte / Standort der Messung:** Die Verifikation kann von jedem beliebigen Standpunkt im Netzwerk realisiert werden, wo ein Rechner mit Browser vorhanden ist. | |
| **Verkettung / Ähnlich zu:** ----- | |
| **Erwartetes Ergebnis:** Kein Unterbruch zum Node, der zweite Ethernet – Port arbeitet weiter. | **Modifikationen Testablauf:** ----- |
| **Spezielles:** ----- | |

*Abbildung 94: Tests / TR1-T0001-1: Der Ethernet - Port 1 wurde im laufenden Betrieb von gfsn1.mgmtdom ausgestreckt.*

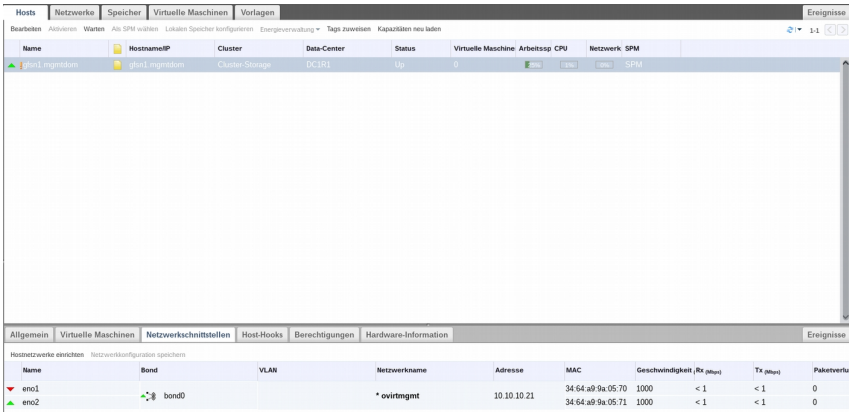



*Abbildung 95: Tests / TR1-T0001-2: oVirt hat den Unterbruch sofort erkannt und ihn in der Statusleiste gemeldet. Grafisch wurde auch der aktuelle Zustand abgebildet.*

| | |
|---|---|
| **Resultat des Tests:** OK | **Bemerkungen:** Der Node erkannte den Fehler und meldete ihn. Der reguläre Betrieb war weiterhin möglich, da ein Ethernet – Port noch vorhanden war und das entsprechende Kernel – Modul das Balancing auf einen Port reduzierte. Im Realbetrieb wäre der Traffic der vorher über Port 1 lief von Port 2 übernommen worden. Der Node ist weiterhin im UP – Zustand. |
| **Mögliche Konsequenzen / Änderungen:** ----- | |
| **Spezielles:** ----- | |

### 9.5.1.2 Testprotokoll: Trennung von XOR – Bonding Port 1 bei ovn3.mgmtdom

| | |
|---|---|
| **Testname:** TR1-T0002 | **Abhängigkeiten:** Keine speziellen, oVirt + Node OVN 3 |
| **Parameter:** Cluster noch im Leerlauf, keine VM's aktiv. | **Verwendete Tools:** Keine speziellen, oVirt selbst |
| **Ablauf des Tests:** Der Cluster ist in Betrieb, es wird der Ethernet – Port 1 des Virtualisierungsnodes ovn3.mgmtdom entfernt. | |
| **Messpunkte / Standort der Messung:** Die Verifikation kann von jedem beliebigen Standpunkt im Netzwerk realisiert werden, wo ein Rechner mit Browser vorhanden ist. | |
| **Verkettung / Ähnlich zu:** TR1-T0001 | |
| **Erwartetes Ergebnis:** Kein Unterbruch zum Node, der zweite Ethernet – Port arbeitet weiter | **Modifikationen Testablauf:** ----- |
| **Spezielles:** ----- | |

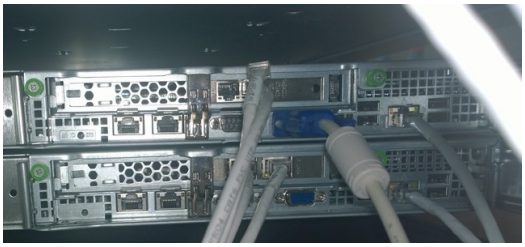


*Abbildung 96: Tests / TR1-T0002-1: Der Ethernet - Port 1 wurde im laufenden Betrieb von ovn3.mgmtdom ausgestreckt.*

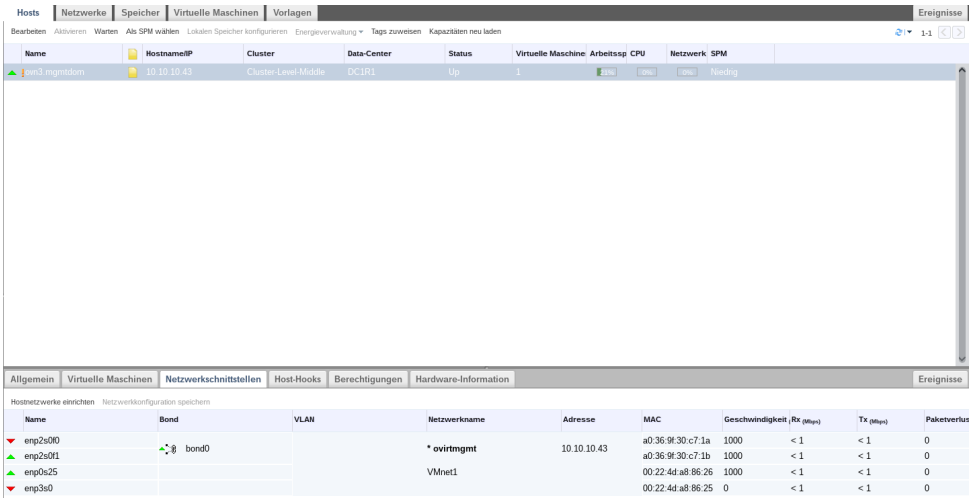



*Abbildung 97: Tests / TR1-T0002-2: oVirt hat den Unterbruch sofort erkannt und ihn in der Statusleiste gemeldet. Grafisch wurde auch der aktuelle Zustand abgebildet.*

| | |
|---|---|
| **Resultat des Tests:** OK | **Bemerkungen:** Der Node erkannte den Fehler und meldete ihn. Der reguläre Betrieb war weiterhin möglich, da ein Ethernet – Prot noch vorhanden war und das entsprechende Kernel – Modul das Balancing auf einen Port reduzierte. Im Realbetrieb wäre der Traffic der vorher über Port 1 lief von Port 2 übernommen worden. Der Node ist weiterhin im UP – Zustand. |
| **Mögliche Konsequenzen / Änderungen:** ----- | |
| **Spezielles:** ----- | |

### 9.5.2 Testprotokoll: Trennen von VM – Netzwerk Port (LACP)

| | |
|---|---|
| **Testname:** TR1-T0003 | **Abhängigkeiten:** Lauffähige VM's mit Netwerkanbindung:<br>• STATIC_debian_Testsys_1<br>• STATIC_debian_Testsys_1<br>• STATIC_debian_Testsys_1<br>Alle drei Maschinen Laufen auf ovn4.mgmtdom. |
| **Parameter:** Cluster aktiv, drei VM's sind in Betrieb. | **Verwendete Tools:** tcpdump auf pfSense Firewall über SSH – Verbindung. Ping auf VM's. |

**Ablauf des Tests:** Die drei VM's werden über das VM – Netzwerk zu je drei unterschiedlich Zielen von Node OVN 4 Pingen:

- STATIC_debian_Testsys_1 → google.ch
- STATIC_debian_Testsys_2 → hbu.ch
- STATIC_debian_Testsys_3 → distrowatch.com

Es wird ein Ping – Auszug aus dem für jede VM zuständen Interface von pfSense gezogen. Am Ende werden alle Interfaces bis auf eines von pfSense im laufenden Ping – Betrieb getrennt.

| | |
|---|---|
| **Messpunkte / Standort der Messung:** Home – PC über SSH zu pfSense. | |
| **Verkettung / Ähnlich zu:** ----- | |
| **Erwartetes Ergebnis:** Kein Unterbruch, das Pingen sollte über ein Interface noch möglich sein, die anderen werden keine Daten mehr liefern. | **Modifikationen Testablauf:** ----- |

**Spezielles:** Die hier aufgeführten Auszüge sind lediglich stark verkürzte Zusammenfassungen, die Originale sind auf dem beiliegenden Datenträger unter → **...DATA/Tests/Testreihe1/TR1-T0003 zu finden.**

```
[2.2.5-RELEASE][admin@bouncer1.localdom]/root: tcpdump -vv -i igb9 icmp
tcpdump: WARNING: igb9: no IPv4 address assigned
tcpdump: listening on igb9, link-type EN10MB (Ethernet), capture size 65535 bytes
17:38:09.280594 IP (tos 0x0, ttl 57, id 28551, offset 0, flags [none], proto ICMP (1), length 84)
    zrh04s05-in-f15.1e100.net > 10.100.100.140: ICMP echo reply, id 1571, seq 152, length 64
17:38:10.283842 IP (tos 0x0, ttl 57, id 29359, offset 0, flags [none], proto ICMP (1), length 84)
    zrh04s05-in-f15.1e100.net > 10.100.100.140: ICMP echo reply, id 1571, seq 153, length 64
17:38:11.284087 IP (tos 0x0, ttl 57, id 29918, offset 0, flags [none], proto ICMP (1), length 84)
    zrh04s05-in-f15.1e100.net > 10.100.100.140: ICMP echo reply, id 1571, seq 157, length 64
17:38:15.290690 IP (tos 0x0, ttl 57, id 31487, offset 0, flags [none], proto ICMP (1), length 84)
    zrh04s05-in-f15.1e100.net > 10.100.100.140: ICMP echo reply, id 1571, seq 158, length 64
^C
7 packets captured
9 packets received by filter
0 packets dropped by kernel
```

*Abbildung 98: Tests / TR1-T0003-1: STATIC_debian_Testsys_1 Ping in Richtung --> google.ch über pfSense Iface igb9, alle Ifaces noch aktiv.*

```
[2.2.5-RELEASE][admin@bouncer1.localdom]/root: tcpdump -vv -i igb11 icmp
tcpdump: WARNING: igb11: no IPv4 address assigned
tcpdump: listening on igb11, link-type EN10MB (Ethernet), capture size 65535 bytes
17:42:01.822186 IP (tos 0x0, ttl 58, id 33764, offset 0, flags [none], proto ICMP (1), length 84)
    3.inware.ch > 10.100.100.141: ICMP echo reply, id 2339, seq 9, length 64
17:42:02.822226 IP (tos 0x0, ttl 58, id 33765, offset 0, flags [none], proto ICMP (1), length 84)
    3.inware.ch > 10.100.100.141: ICMP echo reply, id 2339, seq 10, length 64
17:42:03.826493 IP (tos 0x0, ttl 58, id 33766, offset 0, flags [none], proto ICMP (1), length 84)
    3.inware.ch > 10.100.100.141: ICMP echo reply, id 2339, seq 15, length 64
17:42:08.832211 IP (tos 0x0, ttl 58, id 33771, offset 0, flags [none], proto ICMP (1), length 84)
    3.inware.ch > 10.100.100.141: ICMP echo reply, id 2339, seq 16, length 64

^C
9 packets captured
10 packets received by filter
0 packets dropped by kernel
```

*Abbildung 99: Tests / TR1-T0003-1: STATIC_debian_Testsys_2 Ping in Richtung --> hbu.ch über pfSense Iface igb11, alle Ifaces noch aktiv.*

```
[2.2.5-RELEASE][admin@bouncer1.localdom]/root: tcpdump -vv -i igb10 icmp
tcpdump: WARNING: igb10: no IPv4 address assigned
tcpdump: listening on igb10, link-type EN10MB (Ethernet), capture size 65535 bytes
17:45:47.117390 IP (tos 0x0, ttl 64, id 20947, offset 0, flags [DF], proto ICMP (1), length 84)
    10.100.100.142 > e82-103-136-226s.easyspeedy.com: ICMP echo request, id 1645, seq 39, length 64
17:45:48.119087 IP (tos 0x0, ttl 64, id 21083, offset 0, flags [DF], proto ICMP (1), length 84)
    10.100.100.142 > e82-103-136-226s.easyspeedy.com: ICMP echo request, id 1645, seq 40, length 64
17:45:49.121072 IP (tos 0x0, ttl 64, id 21121, offset 0, flags [DF], proto ICMP (1), length 84)
    10.100.100.142 > e82-103-136-226s.easyspeedy.com: ICMP echo request, id 1645, seq 41, length 64
17:45:50.122703 IP (tos 0x0, ttl 64, id 21307, offset 0, flags [DF], proto ICMP (1), length 84)
    10.100.100.142 > e82-103-136-226s.easyspeedy.com: ICMP echo request, id 1645, seq 42, length 64
^C
10 packets captured
12 packets received by filter
0 packets dropped by kernel
```

*Abbildung 100: Tests / TR1-T0003-1: STATIC_debian_Testsys_3 Ping in Richtung --> distrowatch.com über pfSense Iface igb10, alle Ifaces noch aktiv.*

*Abbildung 101: Tests / TR1-T0003-2: Trennen der Ifaces igb9 und igb 10 von der pfSense Firewall in laufendem Betrieb. Iface igb11 bleibt Online.*

```
[2.2.5-RELEASE][admin@bouncer1.localdom]/root: tcpdump -vv -i igb10 icmp
tcpdump: WARNING: igb10: no IPv4 address assigned
tcpdump: listening on igb10, link-type EN10MB (Ethernet), capture size 65535 bytes
^C
0 packets captured
0 packets received by filter
0 packets dropped by kernel
[2.2.5-RELEASE][admin@bouncer1.localdom]/root: tcpdump -vv -i igb9 icmp
tcpdump: WARNING: igb9: no IPv4 address assigned
tcpdump: listening on igb9, link-type EN10MB (Ethernet), capture size 65535 bytes
^C
0 packets captured
0 packets received by filter
0 packets dropped by kernel
```

*Abbildung 102: Tests / TR1-T0003-2: Start tcpdump auf igb10 und igb9 nach Trennung der Ifaces.*

```
[2.2.5-RELEASE][admin@bouncer1.localdom]/root: tcpdump -vv -i igb11 icmp
tcpdump: WARNING: igb11: no IPv4 address assigned
tcpdump: listening on igb11, link-type EN10MB (Ethernet), capture size 65535 bytes
17:53:14.131874 IP (tos 0x0, ttl 64, id 21084, offset 0, flags [DF], proto ICMP (1), length 84)
    10.100.100.142 > e82-103-136-226s.easyspeedy.com: ICMP echo request, id 1648, seq 259, length 64
17:53:14.166667 IP (tos 0x0, ttl 51, id 54835, offset 0, flags [none], proto ICMP (1), length 84)
    e82-103-136-226s.easyspeedy.com > 10.100.100.142: ICMP echo reply, id 1648, seq 259, length 64
17:53:14.604761 IP (tos 0x0, ttl 64, id 34284, offset 0, flags [DF], proto ICMP (1), length 84)
    10.100.100.140 > zrh04s05-in-f24.1e100.net: ICMP echo request, id 1734, seq 281, length 64
17:53:14.614236 IP (tos 0x0, ttl 57, id 65427, offset 0, flags [none], proto ICMP (1), length 84)
    zrh04s05-in-f24.1e100.net > 10.100.100.140: ICMP echo reply, id 1734, seq 281, length 64
17:53:14.817254 IP (tos 0x0, ttl 64, id 15233, offset 0, flags [DF], proto ICMP (1), length 84)
    10.100.100.141 > 3.inware.ch: ICMP echo request, id 2343, seq 266, length 64
17:53:14.825187 IP (tos 0x0, ttl 58, id 34115, offset 0, flags [none], proto ICMP (1), length 84)
    3.inware.ch > 10.100.100.141: ICMP echo reply, id 2343, seq 266, length 64
17:53:15.132985 IP (tos 0x0, ttl 64, id 21227, offset 0, flags [DF], proto ICMP (1), length 84)
^C
102 packets captured
108 packets received by filter
0 packets dropped by kernel
```

*Abbildung 103: Tests / TR1-T0003-3: Start tcpdump auf igb11, alle drei ICMP Pings laufen nun über das eine noch aktive Iface igb11.*

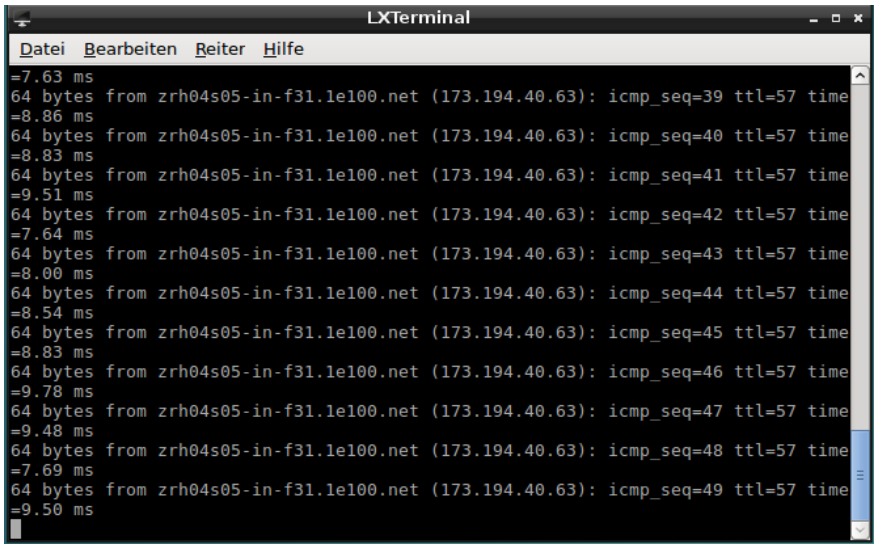



*Abbildung 104: Tests / TR1-T0003-3: Dieser Screenshot wurde von STATIC_debian_Testsys_1 während dem "schnellen" Trennen der Ifaces 9 und 10 erstellt. Die Aufzeichnung zeigt den Zeitpunkt kurz vor und nach dem Trennen. Wie zu sehen ist, fand kein Unterbruch statt.*

| | |
|---|---|
| **Resultat des Tests:** OK | **Bemerkungen:** Wie in dieser Screenshot Serie zu sehen ist, verteilt LACP den Traffic auf alle drei Interfaces. Beim Ausfall von zwei Ifaces fand kein Unterbruch statt, der Traffic wurde von pfSense und dem Node neu ausgehandelt und über igb11 geleitet. |
| **Mögliche Konsequenzen / Änderungen: -----** | |
| **Spezielles:** Die Screenshots zeigen je nur ein Iface pro VM. Der Nachweis auf effizienteres LACP wird später mit Hilfe von diesen Auszügen erbracht. | |

### 9.5.3 Testprotokoll: Manuelles Fencing auf der theoretischen Basis von Semi-Automatic-HA

| | |
|---|---|
| **Testname:** TR1-T0004 | **Abhängigkeiten:** Keine speziellen, oVirt + Node OVN 3 |
| **Parameter:** Cluster-Level-Middle mit den beiden Nodes OVN 2 / 3. Hier wurde eine temporäre VM namens DA_Testsys_1 zum testen erzeugt, um bei schwerwiegenden Schäden nicht eine der bestehenden STATIC_ Maschinen zu verlieren. Hierbei handelt es sich um eine weitere VM aus der gleichen Vorlage wie die anderen ohne irgendwelche Limitierungen. | **Verwendete Tools:** Keine speziellen, oVirt selbst |

**Ablauf des Tests:** DA_Testsys_1 wird auf ovn3.mgmtdom in betrieb genommen. Und mit HA Privilegien ausgestattet. Danach wird der Node ovn3.mgmtdom per Power – Button gewaltsam ausgeschaltet. Hierauf sollte oVirt mit der Neuklassifizierung des Node beginnen und ihm in den Status **Non Operational** setzen. Nachfolgend, wenn alle Versuche ihn neu zu starten gescheitert sind, kann er per Hand in den Staus **Non Responsive** versetzt. Ab jetzt geschieht nichts bis das manuelle Fencing initialisiert wird. Wird es initialisiert, so sollte die VM auf den Cluster – Partner ovn2.mgmtdom neu gestartet werden.

**Messpunkte / Standort der Messung:** Die Verifikation kann von jedem beliebigen Standpunkt im Netzwerk realisiert werden, wo ein Rechner mit Browser vorhanden ist.

**Verkettung / Ähnlich zu:** -----

| | |
|---|---|
| **Erwartetes Ergebnis:** VM DA_Testsys_1 sollte nach dem manuellen Fencing sauber auf ovn2.mgmtdom neu gestartet werden. | **Modifikationen Testablauf:** ----- |

**Spezielles: -----**

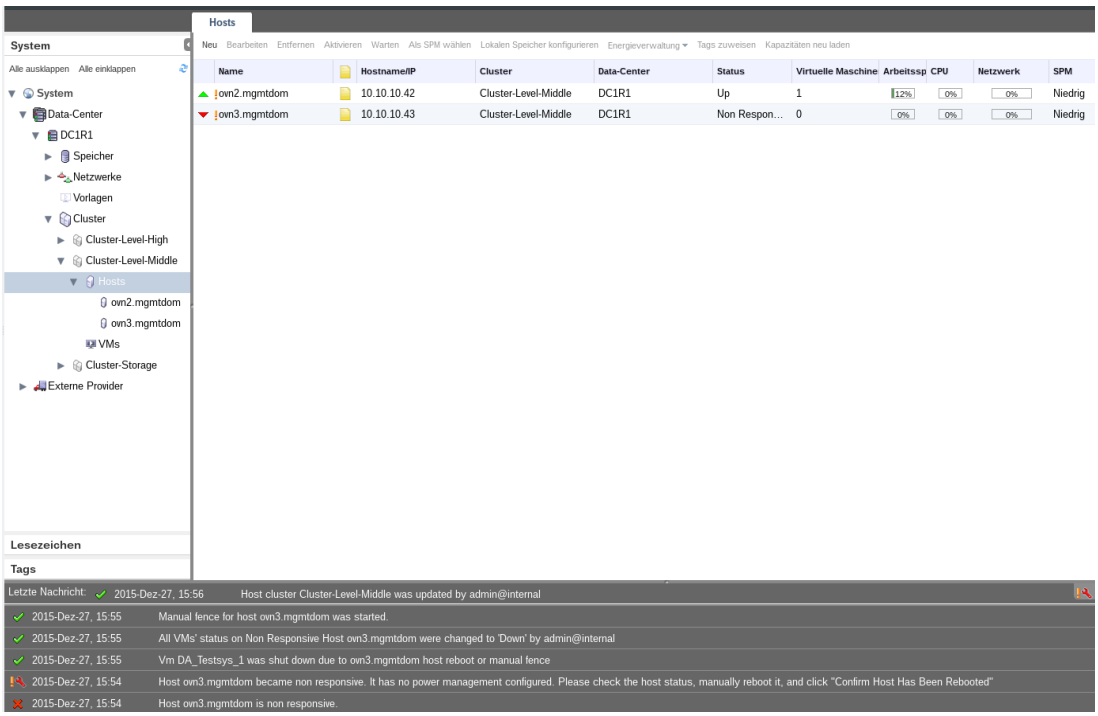



*Abbildung 105: Tests / TR1-T0004-1: Host ovn3.mgmtdom wurde gewaltsam ausgeschaltet und ist per Hand in den Modus Non Responsive gesetzt worden. Nachfolgend wurde über das Kontextmenü das manuelle Fencing initialisiert. Nun beginnt oVirt mit der Kalkulieren des HA für diesen Cluster. Alle Schritte sind in der grauen Statusleiste der Abbildung als Ereignis zu erkennen.*

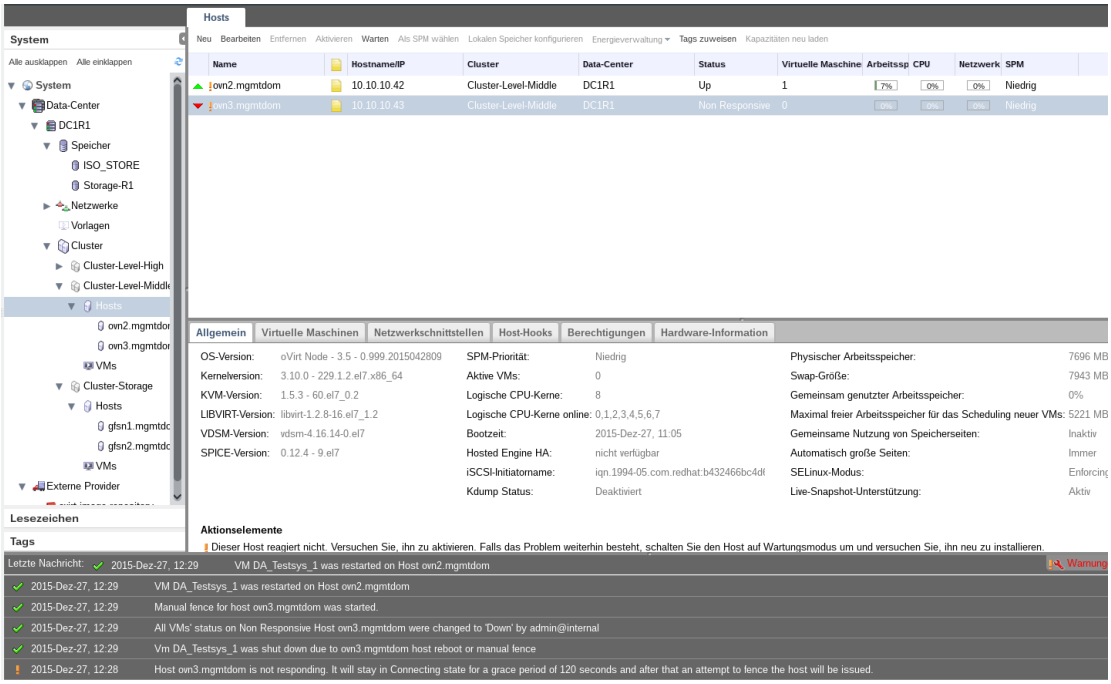



*Abbildung 106: Tests / TR1-T0004-2: Nach initialisieren des manuellen Fencing, wurde die VM neu gestartet und läuft nun auf ovn2.mgmtdom.*

| | |
|---|---|
| **Resultat des Tests:** OK | **Bemerkungen:** Das manuelle Fencing verlief wie gewünscht, die VM wurde nach initialisieren des manuellen Fencing ordnungsgemäss auf ovn2.mgmtdom neu gestartet. Nach dem Neustart von ovn3.mgmtdom tauchte die VM zwar als nicht nutzbarer Zombie in der Liste wieder auf, da vermutlich das Locking von ovn3.mgmtdom nicht vom Storage entfernt wurde, jedoch erkannte oVirt dies ohne Weiteres und löschte sie wieder aus der Liste. Die VM funktionierte nach dem Neustart ohne Probleme wie gewünscht. |
| **Mögliche Konsequenzen / Änderungen: -----** | |
| **Spezielles:** Ein gewaltsam ausgeschalteter Host hat etwas Schwierigkeiten beim erneuten Einbinden in den Cluster. Erst beim zweiten Aktivierungsversuch liess er sich wieder einbinden. Ansonsten verlief der Eingriff problemlos. | |

### 9.5.4 Testprotokoll: Trennen der Netzwerkverbindung der Engine im Betrieb

| | |
|---|---|
| **Testname:** TR1-T0005 | **Abhängigkeiten:** Keine speziellen, oVirt selbst |
| **Parameter:** Alle Cluster sind Online und alle Test – VM's sind in Betrieb. | **Verwendete Tools:** oVirt selbst + netstat + Ping |
| **Ablauf des Tests:** Im Laufenden Betrieb soll die Netzwerkverbindung der Engine getrennt werden. Dies geschieht durch herausziehen des Netzwerkkabels an der Engine selbst. Die Trenndauer soll hier 5 Minuten betragen. | |
| **Messpunkte / Standort der Messung:** Die Verifikation kann von jedem beliebigen Standpunkt im Netzwerk realisiert werden, wo ein Rechner mit Browser und Konsole (SSH) vorhanden ist. | |
| **Verkettung / Ähnlich zu:** ----- | |
| **Erwartetes Ergebnis:** Die Engine selbst kann nach dieser Aktion nicht mehr Kontaktiert werden um die Ereignisse auf dem GUI zu analysieren. Es wird aber erwartet, dass erstens alles innerhalb des Clusters weiter läuft (bis auf grafische Konsole; SPICE) wie vor der Trennung und zweitens, dass sich die Engine wieder ohne Schwierigkeiten selbst einbindet beim Wiederherstellen der Verbindung. | **Modifikationen Testablauf:** ----- |
| **Spezielles:** ----- | |

```
[root@gfsn1 ~]# netstat
Aktive Internetverbindungen (ohne Server)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 gfsn1.mgmtdom:24007     gfsn1.mgmtdom:65533     VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:985       gfsn1.mgmtdom:nfs       VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:49153     gfsn2.mgmtdom:65533     VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:24007     ovn3.mgmtdom:exp2       VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:65534     gfsn2.mgmtdom:24007     VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:65524     gfsn2.mgmtdom:49154     VERBUNDEN
tcp        0      0 localhost:65532         localhost:24007         VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:65520     gfsn2.mgmtdom:49153     VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:49153     gfsn1.mgmtdom:65521     VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:24007     gfsn2.mgmtdom:65530     VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:49153     gfsn1.mgmtdom:65532     VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:24007     ovn2.mgmtdom:exp2       VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:49153     gfsn2.mgmtdom:65522     VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:nfs       ovn2.mgmtdom:964        VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:65532     gfsn1.mgmtdom:49153     VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:nfs       gfsn1.mgmtdom:985       VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:65523     gfsn1.mgmtdom:49154     VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:nfs       gfsn2.mgmtdom:739       VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:65527     gfsn2.mgmtdom:49153     VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:65521     gfsn1.mgmtdom:49153     VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:ssh       rhw-oss-1.localdo:41067 VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:65531     gfsn1.mgmtdom:24007     VERBUNDEN
tcp        0      0 localhost:24007         localhost:65532         VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:65533     gfsn1.mgmtdom:24007     VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:24007     gfsn1.mgmtdom:65530     VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:24007     gfsn2.mgmtdom:65534     VERBUNDEN
tcp        0   1147 gfsn1.mgmtdom:54321     ovirt-mgmt.mgmtdo:41389 VERBUNDEN
tcp        0   1147 gfsn1.mgmtdom:54321     ovirt-mgmt.mgmtdo:41391 VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:49154     gfsn1.mgmtdom:65523     VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:24007     gfsn1.mgmtdom:65531     VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:49153     ovn3.mgmtdom:exp1       VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:49154     gfsn2.mgmtdom:65525     VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:nfs       ovn3.mgmtdom:971        VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:65530     gfsn1.mgmtdom:24007     VERBUNDEN
tcp        0      0 gfsn1.mgmtdom:49153     ovn2.mgmtdom:1017       VERBUNDEN
Aktive Sockets in der UNIX Domäne (ohne Server)
```

*Abbildung 107: Tests / TR1-T0005-1: Auf gfsn1.mgmtdom wurde per SSH verbunden und ein netstat aufgerufen. Dieser zeigt klar, dass sämtliche Storageverbindungen noch bestehen. oVirt selbst ist ebenfalls in der Liste, dies scheint aber an der hohen Timeout - Phase von netstat zu liegen.*

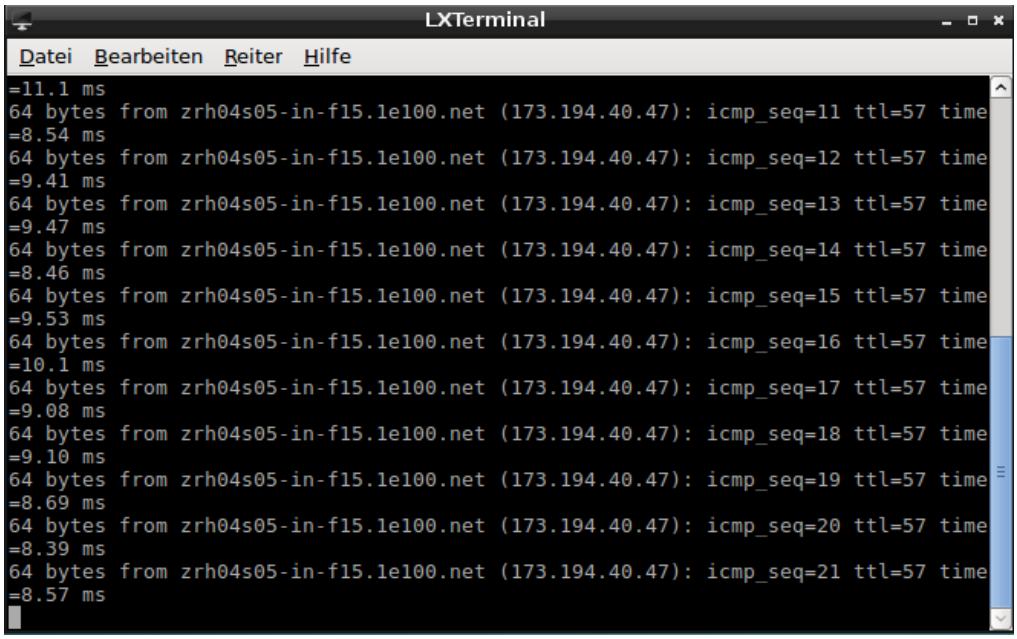



*Abbildung 108: Tests / TR1-T0005-2: Eine SPICE - Konsole wurde mit Absicht offen gelassen um von der VM aus in Richtung google.ch zu pingen. Somit ist auch der Erhalt der Netzwerkverbindung gewährleiste.*

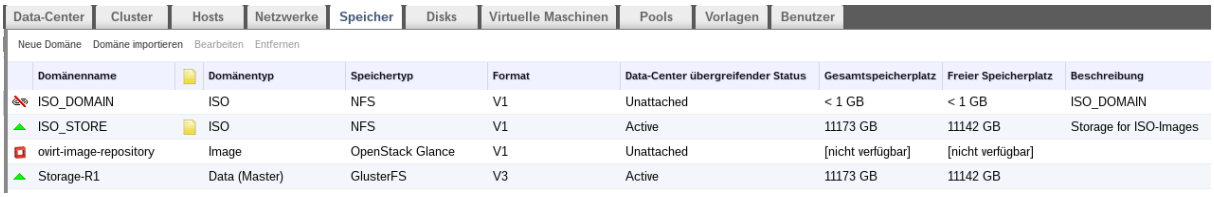



*Abbildung 109: Tests / TR1-T0005-3: Ein Screenshot der Storage – Ansicht nach dem Einbinden der Engine (5 minütiger Unterbruch).*

| | |
|---|---|
| **Resultat des Tests:** OK | **Bemerkungen:** Wie zu erwarten war, funktioniert der Cluster auch ohne die Engine so weiter wie zum Zeitpunkt der Trennung. Dies soll auch so sein, da der Clusterbetrieb in den elementarsten Funktionen unabhängig ist von der Engine. Informationen wie Storage – Zugehörigkeiten werden ohnehin von der Engine an die Nodes verteilt und vom Storage Pool Master gemanagt. Lediglich der Verbindungsaufbau zu den SPICE – Konsolen und das HA funktionieren ohne die Engine nicht. |
| **Mögliche Konsequenzen / Änderungen: -----** | |
| **Spezielles:** Die Engine brauchte nach der Wiederherstellung der Verbindung eine gute Minute um in einem ewig laufendem Aktualisierungsprozess sämtliche Informationen zu aktualisieren. Innerhal dieser Zeit war zwar die Oberfäche ereichbar, aber keine Ressource konnte direkt angesprochen werden. | |

## 9.6 Beginn der Testreihe 2

Diese Testreihe soll sich stark mit den internen Eigenschaften des Cluster – Konstruktes befassen. Hier wird ausschliesslich mit den Softwarekomponenten gearbeitet, womit dass manuelle Manipulieren der Hardware komplett wegfällt. Auch werden hier die neuen VM's welche innerhalb des Kapitels 8 „Realisation“ erstellt worden sind zum Zuge kommen. Es wird versuch die allgemeine Struktur Netzwerk, Virtualisierung und Storage an dieser Stelle so gut wie möglich aufrechtzuerhalten.

## 9.6.1 Testprotokolle des Bereiches Netzwerk

### 9.6.1.1 Testprotokoll: Funktionskontrolle des Switch (Bridge pfSense)

| | |
|---|---|
| **Testname:** TR2-T0006 | **Abhängigkeiten:** Minimum 2 VM's + pfSense |
| **Parameter:** Funktionierender Cluster mit 2 VM's + funktionierende Bridge. | **Verwendete Tools:** ssh |

**Ablauf des Tests:** Es werden zwei VM's in zwei unterschiedlichen Clustern benötigt. Hierfür nehmen wir DYN_HIGH_openSUSE-42.1_DA-VM-1 → Cluster-Level-Middle und DYN_HIGH_openSUSE-42.1_DA-VM-2 → Cluster-Level-High. Beide befinden sich in unterschiedlichen Clustern, wobei mindestens ein Übergang über die Link Aggregation vorhanden ist. Somit ist auch die Funktionsfähigkeit von LACP im Prüfbereich enthalten.

**Messpunkte / Standort der Messung:** Die Verifikation kann von jedem beliebigen Standpunkt im Netzwerk realisiert werden, wo ein Rechner mit Browser und Konsole (SSH) vorhanden ist.

**Verkettung / Ähnlich zu:** -----

| | |
|---|---|
| **Erwartetes Ergebnis:** Es soll versucht werden sich mittels SSH auf die jeweiligen VM's untereinander zu verbinden. Hierbei muss beide Male die Link Aggregation überwunden werden und dies noch mit einem sensiblen und hoch sicheren SSH – Verfahren. | **Modifikationen Testablauf:** ----- |

**Spezielles:** -----

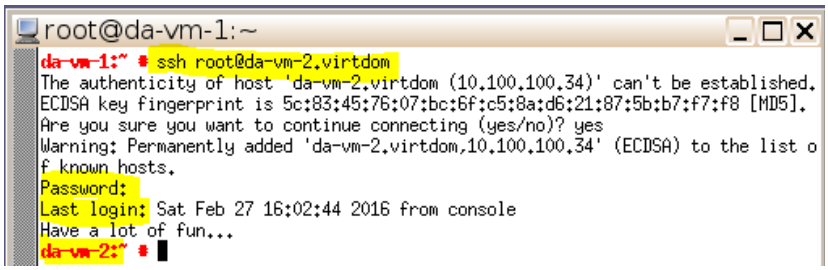



*Abbildung 110: Tests / TR2-T0006-1: Funktionierende Athentifizierung und Anmeldung an da-vm-2*

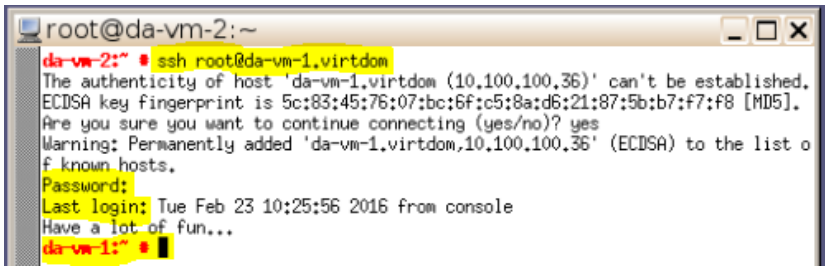



*Abbildung 111: Tests / TR2-T0006-2: Funktionierende Athentifizierung und Anmeldung an da-vm-1*

| | |
|---|---|
| **Resultat des Tests:** OK | **Bemerkungen:** Verbindungen per SSH in beide Richtungen durch die Link Aggregation waren ohne weiteres möglich. |

**Mögliche Konsequenzen / Änderungen:** -----

**Spezielles:** -----

## 9.6.1.2 Testprotokolle: Isolationsfähigkeit der Regelsätze

### 9.6.1.2.1 Testprotokoll: Kein Zugriff auf privaten Sektor (LAN) von virtdom aus

| | |
|---|---|
| **Testname:** TR2-T0007 | **Abhängigkeiten:** Eine VM erforderlich |
| **Parameter:** Funktionierender Cluster mit 1 VM + Aktive Sperrregeln. | **Verwendete Tools:** ssh |
| **Ablauf des Tests:** Von DYN_HIGH_openSUSE-42.1_DA-VM-2 → Cluster-Level-High soll versucht werden eine Verbindung zu openelec.localdom (IPTV des Autors) herzustellen. Die Regelsätze für die restlichen Nodes sind identisch, womit dieses Testergebnis auf alle adaptiert werden kann. | |
| **Messpunkte / Standort der Messung:** Die Verifikation kann von jedem beliebigen Standpunkt im Netzwerk realisiert werden, wo ein Rechner mit Browser und Konsole (SSH) vorhanden ist. | |
| **Verkettung / Ähnlich zu:** ----- | |
| **Erwartetes Ergebnis:** Die Verbindung soll nicht zustande kommen und muss nach 1 Minute manuell abgebrochen werden, falls bis dahin kein Timeout erfolgt. | **Modifikationen Testablauf:** ----- |
| **Spezielles:** ----- | |

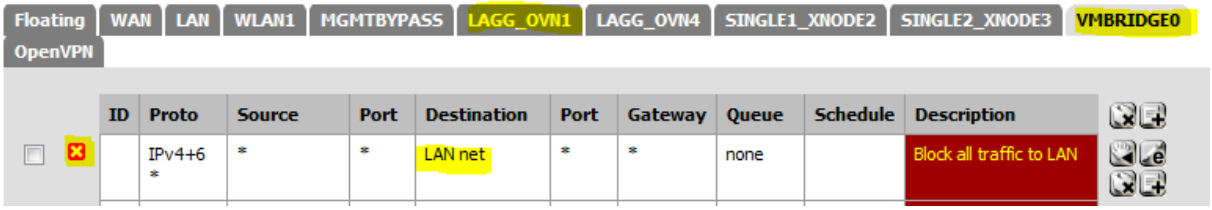



*Abbildung 112: Tests / TR2-T0007-1: Die allgemeine Regel, welche allen VM's den Zugang zum privaten Teil des Netzwerkes verweigert (gilt auch für WLAN1).*

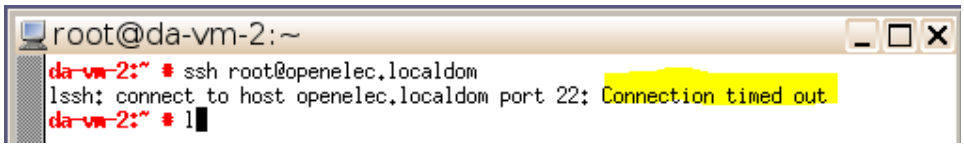



*Abbildung 113: Tests / TR2-T0007-2: Wie zu erwarten lief die Verbindung in ein Timeout*

| | |
|---|---|
| **Resultat des Tests:** OK | **Bemerkungen:** ----- |
| **Mögliche Konsequenzen / Änderungen:** ----- | |
| **Spezielles:** ----- | |

### 9.6.1.2.2 Testprotokoll: Kein Zugriff auf mgmtdom von virtdom aus

| | |
|---|---|
| **Testname:** TR2-T0008 | **Abhängigkeiten:** Eine VM erforderlich |
| **Parameter:** Funktionierender Cluster mit 1 VM + Aktive Sperrregeln. | **Verwendete Tools:** ssh |

**Ablauf des Tests:** Von DYN_HIGH_openSUSE-42.1_DA-VM-2 → Cluster-Level-High soll versucht werden eine Verbindung zu ovirt-mgmtdom (Engine) herzustellen. Die Regelsätze für die restlichen Nodes sind identisch, womit dieses Testergebnis auf alle adaptiert werden kann.

**Messpunkte / Standort der Messung:** Die Verifikation kann von jedem beliebigen Standpunkt im Netzwerk realisiert werden, wo ein Rechner mit Browser und Konsole (SSH) vorhanden ist.

**Verkettung / Ähnlich zu:** -----

| | |
|---|---|
| **Erwartetes Ergebnis:** Die Verbindung soll nicht zustande kommen und muss nach 1 Minute manuell abgebrochen werden, falls bis dahin kein Timeout erfolgt. | **Modifikationen Testablauf:** ----- |

**Spezielles:** DNS – Auflösung in Richtung ovirt-mgmt.mgmtdom ist möglich.

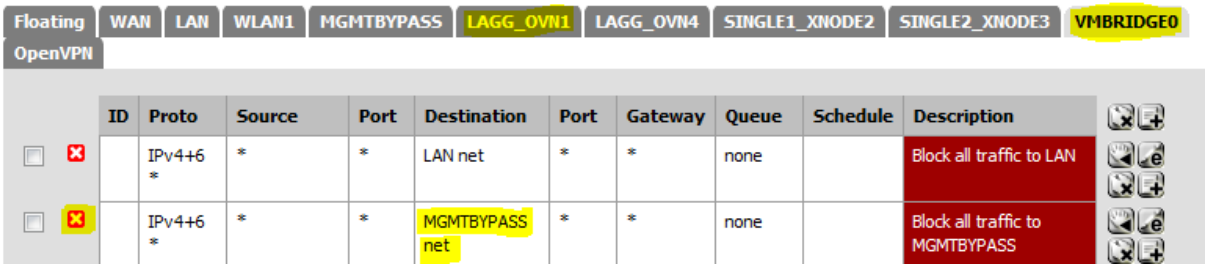



*Abbildung 114: Tests / TR2-T0008-1: Die allgemeine Regel, welche allen VM's den Zugang zu mgmtdom verweigert.*

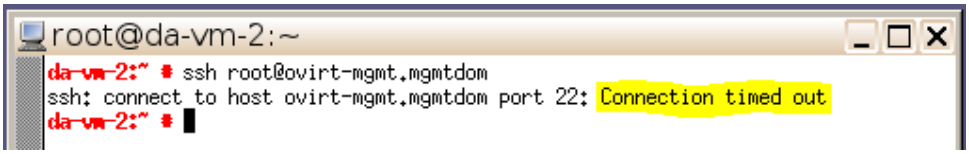



*Abbildung 115: Tests / TR2-T0008-2: Wie zu erwarten lief die Verbindung in ein Timeout*

| | |
|---|---|
| **Resultat des Tests:** OK | **Bemerkungen:** ----- |

**Mögliche Konsequenzen / Änderungen:** -----

**Spezielles:** -----

### 9.6.1.2.3 Testprotokoll: Kein Zugriff auf privaten Sektor (LAN) von mgmtdom aus

| | |
|---|---|
| **Testname:** TR2-T0009 | **Abhängigkeiten:** Eine VM erforderlich |
| **Parameter:** Funktionierender Cluster mit 1 VM + Aktive Sperrregeln. | **Verwendete Tools:** ssh |

**Ablauf des Tests:** Von gfsn1.mgmtdom soll versucht werden eine Verbindung zu openelec.localdom herzustellen. Die Regelsätze für die restlichen Nodes sind identisch, womit dieses Testergebnis auf alle adaptiert werden kann.

**Messpunkte / Standort der Messung:** Die Verifikation kann von jedem beliebigen Standpunkt im Netzwerk realisiert werden, wo ein Rechner mit Browser und Konsole (SSH) vorhanden ist.

**Verkettung / Ähnlich zu:** -----

| | |
|---|---|
| **Erwartetes Ergebnis:** Die Verbindung soll nicht zustande kommen und muss nach 1 Minute manuell abgebrochen werden, falls bis dahin kein Timeout erfolgt. | **Modifikationen Testablauf:** ----- |

**Spezielles:** DNS – Auflösung in Richtung openelec.localdom ist möglich.

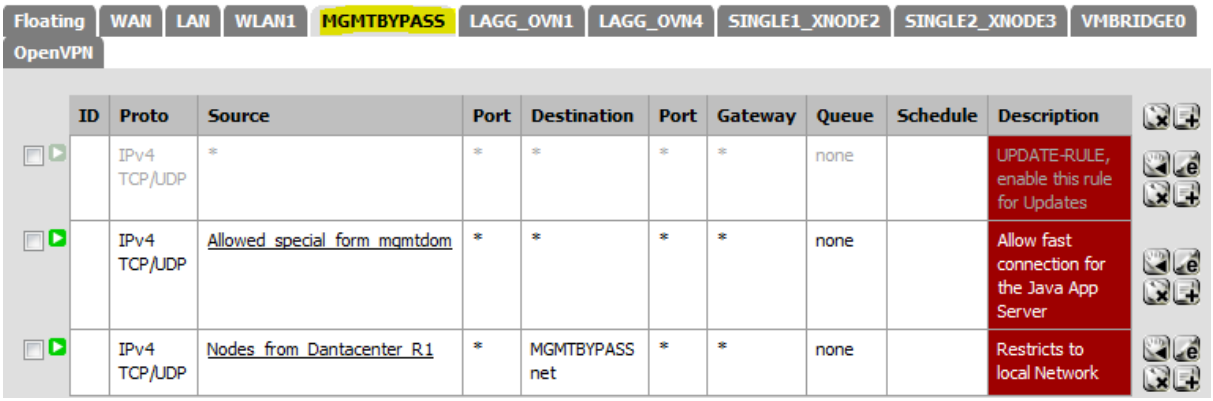



*Abbildung 116: Tests / TR2-T0009-1: Die allgemeine Regel, welche allen Nodes den Zugang zum privaten Sektor verweigert.*

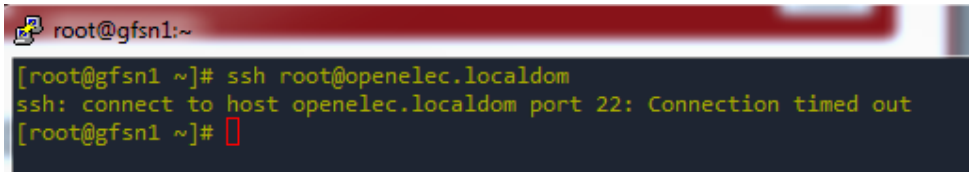



*Abbildung 117: Tests / TR2-T0009-2: Wie zu erwarten lief die Verbindung in ein Timeout*

| | |
|---|---|
| **Resultat des Tests:** OK | **Bemerkungen:** ----- |

**Mögliche Konsequenzen / Änderungen:** -----

**Spezielles:** -----

#### 9.6.1.2.4 Testprotokoll: Kein Zugriff auf virtdom von mgmtdom aus

| | |
|---|---|
| **Testname:** TR2-T0010 | **Abhängigkeiten:** Eine VM erforderlich |
| **Parameter:** Funktionierender Cluster mit 1 VM + Aktive Sperrregeln. | **Verwendete Tools:** ssh |
| **Ablauf des Tests:** Von gfsn1.mgmtdom soll versucht werden eine Verbindung zu Infraserv1.virtdom herzustellen. Die Regelsätze für die restlichen Nodes sind identisch, womit dieses Testergebnis auf alle adaptiert werden kann. | |
| **Messpunkte / Standort der Messung:** Die Verifikation kann von jedem beliebigen Standpunkt im Netzwerk realisiert werden, wo ein Rechner mit Browser und Konsole (SSH) vorhanden ist. | |
| **Verkettung / Ähnlich zu:** Ist ähnlich wie TR2-T0009. Regeln sind identisch, weswegen auf Abbildung 116 referenziert wird. | |
| **Erwartetes Ergebnis:** Die Verbindung soll nicht zustande kommen und muss nach 1 Minute manuell abgebrochen werden, falls bis dahin kein Timeout erfolgt. | **Modifikationen Testablauf:** ----- |
| **Spezielles:** DNS – Auflösung in Richtung Infraserv1.virtdom ist **NICHT** möglich. | |

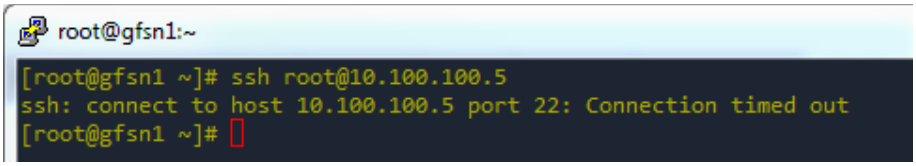



*Abbildung 118: Tests / TR2-T0010-2: Wie zu erwarten lief die Verbindung in ein Timeout*

| | |
|---|---|
| **Resultat des Tests:** OK | **Bemerkungen:** ----- |
| **Mögliche Konsequenzen / Änderungen:** ----- | |
| **Spezielles:** ----- | |

### 9.6.1.3 Testprotokoll: Prüfung auf Anwendung von QoS auf das VM – Netzwerk

| | |
|---|---|
| **Testname:** TR2-T0011 | **Abhängigkeiten:** Zwei VM's erforderlich |
| **Parameter:** Funktionierender Cluster mit 2 VM mit unterschiedlichen QoS Profilen. Die Genauen Spezifikationen des QoS sind unter Punkt 8.9.1.1.2 nachlesbar. | **Verwendete Tools:** ftp |

**Ablauf des Tests:** Mit folgenden beiden VM's; DYN_HIGH_openSUSE-42.1_DA-VM-1 → Cluster-Level-Middle und DYN_HIGH_openSUSE-42.1_DA-VM-3 → Cluster-Level-High soll das gleiche ISO – File Heruntergeladen werden. Beim File handelt es sich um das Net – Installationsimage von openSUSE 42.1 auf der Adresse → **ftp://mirror.switch.ch/mirror/opensuse/opensuse/distribution/leap/42.1/iso/openSUSE-Leap-42.1-NET-x86_64.iso**

**Messpunkte / Standort der Messung:** Die Verifikation kann von jedem beliebigen Standpunkt im Netzwerk realisiert werden, wo eine Konsole (SSH) vorhanden ist.

**Verkettung / Ähnlich zu:** -----

| | |
|---|---|
| **Erwartetes Ergebnis:** Die Downloadzeiten sollten sich markant voneinander unterscheiden. | **Modifikationen Testablauf:** ----- |

**Spezielles:** Der Disk QoS auf DYN_HIGH_openSUSE-42.1_DA-VM-3 wird temporär auf **_HIGH** gesetzt.

```
ftp> get openSUSE-Leap-42.1-NET-x86_64.iso
local: openSUSE-Leap-42.1-NET-x86_64.iso remote: openSUSE-Leap-42.1-NET-x86_64.iso
229 Entering Extended Passive Mode (|||56852|).
150 Opening BINARY mode data connection for openSUSE-Leap-42.1-NET-x86_64.iso (8912
100% |*****************************************************************************
226 Transfer complete.
89128960 bytes received in 00:43 (1.94 MiB/s)
ftp> bye
221 Goodbye.
da-vm-1:~ #
```

*Abbildung 119: Tests / TR2-T0011-1: Hier wurde der Download ohne Limitierung des Netzwerks durchgeführt.*

```
ftp> get openSUSE-Leap-42.1-NET-x86_64.iso
local: openSUSE-Leap-42.1-NET-x86_64.iso remote: openSUSE-Leap-42.1-NET-x86_64.iso
229 Entering Extended Passive Mode (|||58915|).
150 Opening BINARY mode data connection for openSUSE-Leap-42.1-NET-x86_64.iso (8912
100% |*****************************************************************************|
226 Transfer complete.
89128960 bytes received in 01:16 (1.10 MiB/s)
ftp> bye
221 Goodbye.
da-vm-3:~ #
```

*Abbildung 120: Tests / TR2-T0011-2: Hier wurde der Download mit spezieller Limitierung des Netzwerks durchgeführt.*

| | |
|---|---|
| **Resultat des Tests:** OK | **Bemerkungen:** Die Limitierungen unter Punkt 8.9.1.1.2 wurden für das interne Netzwerk designt. Hier wurde aber aus dem Internet heruntergeladen, wo die tiefste QoS – Limitierung immer noch doppelt so hoch ist wie die max. Download – Rate. Der Download aus dem Internet wurde gewählt, da er weit stabiler ist als die momentane Netzauslastung im privaten Sektor. Um dennoch ein sichtbares Ergebnis zu erhalten, wurde für diesen Test die _MIN – Limitierung temporär auf Eingehend → 10 Mbit/s allgemein und Burst auf 5 MB gesetzt. Der Test ist weiterhin gültig, da das Anwenden von QoS erwiesen ist und die Adaption auf das interne QoS – Design gleich funktioniert, einfach mit höheren Datenraten. |

**Mögliche Konsequenzen / Änderungen:** -----

**Spezielles:** -----

## 9.6.2 Testprotokolle des Bereiches Virtualisierung

### 9.6.2.1 Testprotokoll: Prüfung auf Anwendung des CPU – QoS

| | |
|---|---|
| **Testname:** TR2-T0012 | **Abhängigkeiten:** Zwei VM's erforderlich |
| **Parameter:** Funktionierender Cluster mit 2 VM's mit unterschiedlichen QoS Profilen. Die Genauen Spezifikationen des QoS sind unter Punkt 8.9.1.1.3 nachlesbar. | **Verwendete Tools:** SSH, Script stress.sh |

**Ablauf des Tests:** Mit diesen beiden VM's; DYN_HIGH_openSUSE-42.1_DA-VM-2 und DYN_HIGH_openSUSE-42.1_DA-VM-3 soll das eigens hierfür erstellte Stresstest Skript **stress.sh** ausgeführt werden. Dabei gelten folgende Limitierungen für die VM's:

- DYN_HIGH_openSUSE-42.1_DA-VM-2 → läuft auf ovn1.mgmtdom und ist komplett unlimitiert.
- DYN_HIGH_openSUSE-42.1_DA-VM-3 → läuft auf ovn1.mgmtdom und ist CPU- seitig auf 30% limitiert.

Beide Test werden nacheinander durchgeführt, um die gleichen Host – CPU Bedingungen zu erzeugen.

**Messpunkte / Standort der Messung:** Die Verifikation kann von jedem beliebigen Standpunkt im Netzwerk realisiert werden, wo eine Konsole (SSH) vorhanden ist.

**Verkettung / Ähnlich zu:** -----

| | |
|---|---|
| **Erwartetes Ergebnis:** Das Script startet 4 Prozesse gleichzeitig für 30 Sekunden, welche aus **/dev/random** lesen und auf die Disk schreiben. Dabei vergleicht es die Grösse der erzeugten vier Files und addiert sie zusammen. Da dieser Vorgang sehr CPU intensiv ist, sollte bei einer limitierten CPU das Erzeugen des Random länger dauern, was sich bei konstanter Zeit auf die Grösse des Ergebnisses auswirkt. Da die Nodes aber zur Zeit kaum ausgelastet sind, wird hier mit einer minimalen Diskrepanz gerechnet. | **Modifikationen Testablauf:** ----- |

**Spezielles:** Keine der beiden oben genanten VM's nutzt den Zufallsgenerator des Host. Beide vCPU's müssen hier selber arbeiten.


*Abbildung 121: Tests / TR2-T0012-1: Die unlimitierte VM konnte in 30s 56MB an Random - Daten generieren.*

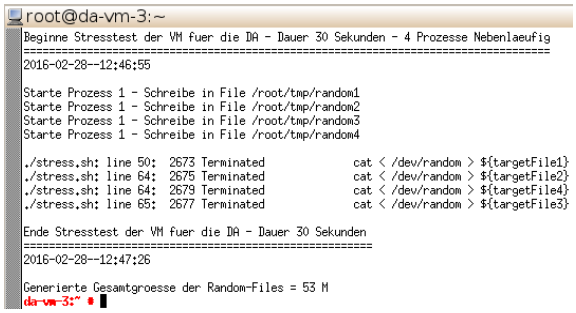



*Abbildung 122: Tests / TR2-T0012-2: Die limitierte VM konnte in 30s 53MB an Random - Daten generieren.*

| | |
|---|---|
| **Resultat des Tests:** OK | **Bemerkungen:** Für beide VM's wurde der Stresstest 3x Mal wiederholt. Dabei konnten praktisch drei identische Ergebnisse erzielt werden. Somit hat die Limitierung durchaus einen Einfluss auf die zu nutzenden CPU – Ressourcen. Jedoch müsste für stärkere Diskrepanzen ein komplexeres Testverfahren genutzt werden und die Testbedingungen auf dem Node müssten für einen realistischen Test so modifiziert werden, dass weit mehr Auslastung herrscht. |

**Mögliche Konsequenzen / Änderungen: -----**

**Spezielles:** Das Script stress**.sh** ist auf dem Datenträger unter **.../DATA/Tests/Testreihe2/TR2-T0012** zu finden.

### 9.6.2.2 Testprotokoll: Nachweis der Live – Migration mit Miteinbezug des Maintenance Mode

| | |
|---|---|
| **Testname:** TR2-T0013 | **Abhängigkeiten:** Eine VM erforderlich |
| **Parameter:** Funktionierender Cluster mit 1 VM mit unterschiedlichen QoS Profilen. Die Genauen Spezifikationen des QoS sind unter Punkt 8.9.1.1.3 nachlesbar. | **Verwendete Tools:** oVirt selbst |

**Ablauf des Tests:** Zu diesem Zeitpunkt wird mindestens eine VM auf Node ovn1.mgmtdom in Betrieb sein, während der Node in den Maintenance – Modus versetzt wird.

**Messpunkte / Standort der Messung:** Die Verifikation kann von jedem beliebigen Standpunkt im Netzwerk realisiert werden, wo ein Browser vorhanden ist.

**Verkettung / Ähnlich zu:** Dieser Test erweist zugleich die allgemeine Migrationsfähigkeit des Aufbaus.

| | |
|---|---|
| **Erwartetes Ergebnis:** Es wird erwartet, dass nach umschalten des Node ovn1.mgmtdom in den Maintenance – Modus, die VM auf den Nachbarnode innerhalb des Clusters verschoben wird. | **Modifikationen Testablauf:** ----- |

**Spezielles:** -----

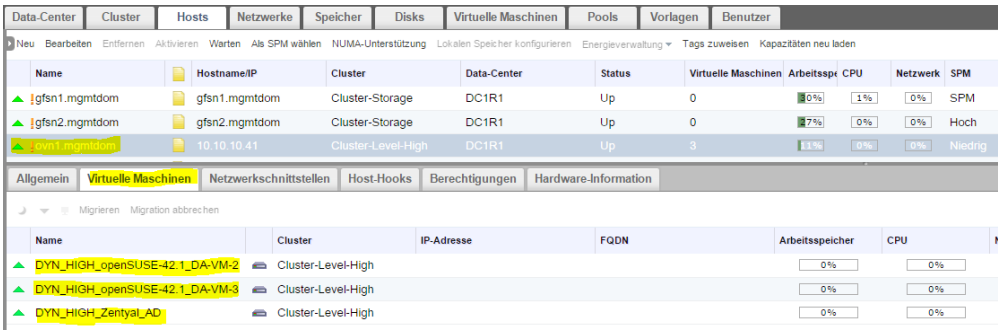



*Abbildung 123: Tests / TR2-T0013-1: Der Zustand auf ovn1.mgmtdom vor der Migration / des Übergangs in den Wartungsmodus.*


*Abbildung 124: Tests / TR2-T0013-2: Der eigentliche Migrationsvorgang nach einleiten des Wartungsmodus auf ovn1.mgmtdom.*

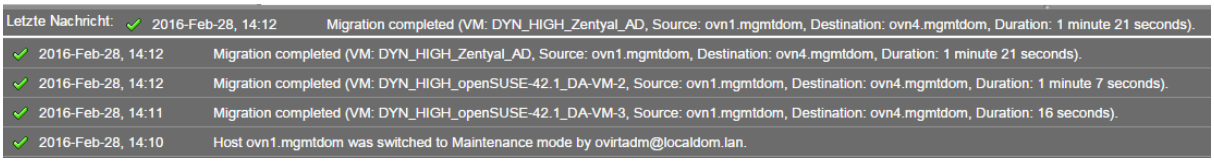



*Abbildung 125: Tests / TR2-T0013-3: Auszug aus dem Ereignisprotokoll von oVirt, wo die einzelnen Schritte ersichtlich sind.*

| | |
|---|---|
| **Resultat des Tests:** OK | **Bemerkungen:** ----- |

**Mögliche Konsequenzen / Änderungen:** -----

**Spezielles:** -----

### 9.6.2.3 Testprotokoll: Prüfung auf Anwendung der Affinitätsgruppen

| | |
|---|---|
| **Testname:** TR2-T0014 | **Abhängigkeiten:** Zwei VM's erforderlich |
| **Parameter:** Funktionierender Cluster mit 2 VM's mit unterschiedlichen QoS Profilen. | **Verwendete Tools:** oVirt selbst |

**Ablauf des Tests:** Um diesen Test etwas kompakter zu halten, wir hier lediglich versuch zwei VM's welche sich in einer positiven Affinitätsgruppe befinden, voneinander zu trennen. Das Funktionsprinzip ist hier in umgekehrter Betrachtung dasselbe wie beim Starten der VM's. Somit kann dieses Testergebnis auf beide Seiten äquivalent angewendet werden. Für diesen Test wird versucht folgende VM aus der Affinität weg zu migrieren:

- DYN_HIGH_openSUSE-42.1_DA-VM-3 weg von ovn4.mgmtdom zu ovn1.mgmtdom

**Messpunkte / Standort der Messung:** Die Verifikation kann von jedem beliebigen Standpunkt im Netzwerk realisiert werden, wo ein Browser vorhanden ist.

**Verkettung / Ähnlich zu:** -----

| | |
|---|---|
| **Erwartetes Ergebnis:** Da beide VM's in einer positiven Bindung zueinander stehen, sollte oVirt die Migration einer von ihnen mit einer Fehlermeldung ablehnen. | **Modifikationen Testablauf:** ----- |

**Spezielles:** -----

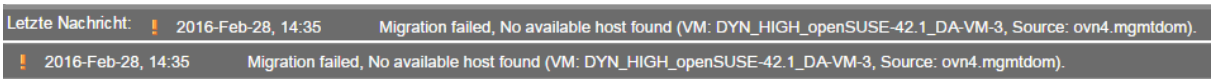



*Abbildung 126: Tests / TR2-T0014-1: Ereignisanzeige von oVirt; die Migration wurde abgelehnt, da sie zu einem Affinitätsbruch führt.*

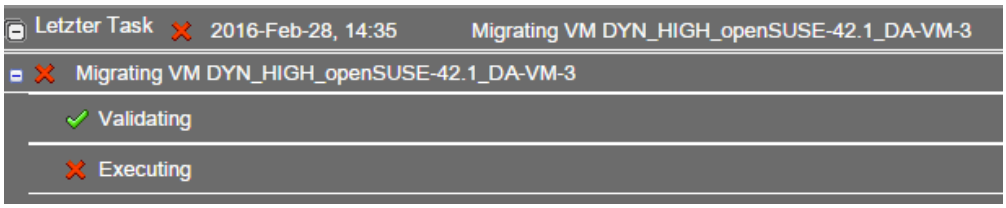



*Abbildung 127: Tests / TR2-T0014-1: Taskanzeige von oVirt; Der Migrationsprozess wurde zuerst als i.O. klassifiziert und bei der nachfolgenden Richtlinienprüfung als ungültig abgelehnt.*

| | |
|---|---|
| **Resultat des Tests:** OK | **Bemerkungen:** Zugegeben, die Meldungen sind nicht wirklich aussagekräftig und könnten einen oVirt – Neuling im ersten Moment verwirren, da sie nicht spezifisch auf die Affinitätsgruppen als ablehnende Quelle hindeuten. Jedoch funktioniert die Sicherung hervorragen und ein Affinitätsbruch ist nicht erzwingbar solange die Regel auf **Hard** steht. |

**Mögliche Konsequenzen / Änderungen:** -----

**Spezielles:** -----

## 9.6.3 Testprotokolle des Bereiches Storage

### 9.6.3.1 Testprotokolle zur Prüfung auf Verteilung der Daten innerhalb von GlusterFS

#### 9.6.3.1.1 Testprotokoll: Prüfung auf Verteilung der Daten des Master – Storage

| | |
|---|---|
| **Testname:** TR2-T0015 | **Abhängigkeiten:** GlusterFS Nodes |
| **Parameter:** Lauffähiger GlusterFS – Verbund. | **Verwendete Tools:** du |

**Ablauf des Tests:** Es wird auf die beiden GlusterFS Nodes per SSH zugegriffen und mittels des Kommandozeilen – Tools **du** ein Auszug aus den Bricks gezogen, welcher von den Verzeichnissen her identisch sein sollte. Jedoch sollten Grössenunterschiede innerhalb der gleichnamigen Verzeichnisse vorhanden sein, da ja die effektiven Rohdaten auf unterschiedlichen Maschinen gespeichert werden.

**Messpunkte / Standort der Messung:** Die Verifikation kann von jedem beliebigen Standpunkt im Netzwerk realisiert werden, wo ein SSH Konsolenclient vorhanden ist.

**Verkettung / Ähnlich zu:** -----

| | |
|---|---|
| **Erwartetes Ergebnis:** Es sollten unterschiedliche Werte hinsichtlich der Speicherplatzbelegung der gleichnamigen Verzeichnisse erschlich sein. | **Modifikationen Testablauf:** ----- |

**Spezielles:** Verzeichnis lokal auf Nodes: **/storage/brick(1 bzw. 2)/20d7808c-90f9-49bd-90c4-0aaee13d5fed/images**

```
[root@gfsn1 images]# du -h
12K     ./afa2de92-a133-436f-b97b-bbaf175f9d2f
1.1M    ./340af683-efe2-4813-965e-e8c454fdc147
1.2M    ./699e8266-4966-4c72-b6ff-cd7f348dd174
1.1M    ./9d48e285-e95e-4e75-819a-0a04a58f7422
11G     ./b297d815-32f7-4531-be6b-acfb52d85ef4
3.6G    ./6aa4723f-4c29-4f7f-a7af-b11420bc5eed
1.1M    ./1d6f23c8-817c-404b-a7d3-191dbaeba52d
0       ./f4898c70-d9a1-4a8a-8618-c1e258f7fff4
4.0G    ./a266bf22-e1a7-4da4-af38-686802b3c6fe
1.2G    ./0ffd4f8a-59e4-4423-8469-aed53450c34b
1.1M    ./ad274c0f-c12f-4d78-bda4-7b38435ca17e
4.3G    ./81f0505b-0981-4577-b9bd-63f52af41823
5.4G    ./df31fc4c-7962-42c8-ae24-a5ead2c1fd8a
4.6G    ./94107c81-340f-4c9f-b0fd-8920916fc5da
1.1M    ./de5afc2a-53d7-4642-8806-f568da1a97ae
34G     .
[root@gfsn1 images]#
```

```
[root@gfsn2 images]# du -h
1.1M    ./afa2de92-a133-436f-b97b-bbaf175f9d2f
0       ./340af683-efe2-4813-965e-e8c454fdc147
4.0K    ./699e8266-4966-4c72-b6ff-cd7f348dd174
208K    ./9d48e285-e95e-4e75-819a-0a04a58f7422
432M    ./b297d815-32f7-4531-be6b-acfb52d85ef4
4.0K    ./6aa4723f-4c29-4f7f-a7af-b11420bc5eed
3.6G    ./1d6f23c8-817c-404b-a7d3-191dbaeba52d
4.1G    ./f4898c70-d9a1-4a8a-8618-c1e258f7fff4
0       ./a266bf22-e1a7-4da4-af38-686802b3c6fe
5.0G    ./0ffd4f8a-59e4-4423-8469-aed53450c34b
1.2G    ./ad274c0f-c12f-4d78-bda4-7b38435ca17e
1.1M    ./81f0505b-0981-4577-b9bd-63f52af41823
1.1M    ./df31fc4c-7962-42c8-ae24-a5ead2c1fd8a
1.0M    ./94107c81-340f-4c9f-b0fd-8920916fc5da
3.8G    ./de5afc2a-53d7-4642-8806-f568da1a97ae
18G     .
[root@gfsn2 images]#
```

*Abbildung 128: Tests / TR2-T0015-1: Der effektive vergleich der beiden Verzeichnisstrukturen auf den GlusterFS Nodes.*

| | |
|---|---|
| **Resultat des Tests:** OK | **Bemerkungen:** Wie zu sehen ist, werden die Daten auf beiden Nodes verteilt. Die leichten Diskrepanzen welche sich bei einigen gleichnamigen Verzeichnissen zeigen, sind Snapshot von VM's welche durch GlusterFS ausbalanciert wurden.Weswegen der Node 1 beim Speichern so stark bevorzugt wird, konnte nicht ermittelt werden. |

**Mögliche Konsequenzen / Änderungen:** -----

**Spezielles:** -----

### 9.6.3.1.2 Testprotokoll: Prüfung auf Verteilung der ISO's auf beide Nodes

| | |
|---|---|
| **Testname:** TR2-T0016 | **Abhängigkeiten:** GlusterFS Nodes |
| **Parameter:** Lauffähiger GlusterFS Verbund. | **Verwendete Tools:** du, ls |
| **Ablauf des Tests:** Der Ablauf des Tests ist identisch mit TR2-0015. Hier wir NFS als Kontaktpunkt zur Datenübermittlung genutzt, somit ist für alle Nodes nur ein Schnittstelle sichtbar. Jedoch kümmert sich GlusterFS eigenständig um das Balancieren der ISO's. Hier wird noch zusätzlich mit dem Befehl **ls -l** der besseren Übersicht halber gearbeitet. | |
| **Messpunkte / Standort der Messung:** Die Verifikation kann von jedem beliebigen Standpunkt im Netzwerk realisiert werden, wo ein SSH Konsolenclient vorhanden ist. | |
| **Verkettung / Ähnlich zu:** TR2-T0015 | |
| **Erwartetes Ergebnis:** Es sollten unterschiedliche Werte hinsichtlich der Speicherplatzbelegung der gleichnamigen Verzeichnisse erschlich sein. | **Modifikationen Testablauf:** ----- |
| **Spezielles:** Verzeichnis lokal auf Nodes: **/storage/ISO_STORE(1 bzw. 2)/8f62618b-7910-4987-a6a2-052b8b083bfa/images/11111111-1111-1111-1111-111111111111** | |

```
[root@gfsn1 11111111-1111-1111-1111-111111111111]# ls -l
insgesamt 851976
-rw-r--r-- 2 root root  87031808 29. Nov 14:14 CorePlus-current.iso
-rw-r--r-- 2 root root 685621248 10. Dez 14:29 FreeBSD-10.1-RELEASE-amd64-disc1.iso
-rw-r--r-- 2 root root  99768320 21. Dez 21:28 oVirt-toolsSetup-3.5_8.fc22.iso
[root@gfsn1 11111111-1111-1111-1111-111111111111]# du -h
833M    .
```

*Abbildung 129: Tests / TR2-T0016-1: Auszug der ISO's aus ISO_STORE1 auf gfsn1.mgmtdom.*

```
[root@gfsn2 11111111-1111-1111-1111-111111111111]# ls -l
insgesamt 6413300
-rw-r--r-- 2 root root  657457152 14. Dez 21:33 debian-8.2.0-amd64-CD-1.iso
-rw-r--r-- 2 root root 4592762880 21. Dez 20:19 de_windows_server_2012_r2_with_update_x64_dvd_4048497.iso
-rw-r--r-- 2 root root   89128960 10. Dez 14:18 openSUSE-Leap-42.1-NET-x86_64.iso
-rw-r--r-- 2 root root  106803200 21. Dez 21:28 ovirt-guest-tools-3.5_5.iso
-rw-r--r-- 2 root root  258998272 22. Feb 22:50 turnkey-domain-controller-14.0-jessie-amd64.iso
-rw-r--r-- 2 root root  232783872 22. Feb 19:09 turnkey-openldap-14.0-jessie-amd64.iso
-rw-r--r-- 2 root root  629284864 22. Okt 12:22 zentyal-4.2-development-amd64.iso
[root@gfsn2 11111111-1111-1111-1111-111111111111]# du -h
6.2G    .
```

*Abbildung 130: Tests / TR2-T0016-2: Auszug der ISO's aus ISO_STORE2 auf gfsn2.mgmtdom.*

| | |
|---|---|
| **Resultat des Tests:** OK | **Bemerkungen:** ----- |
| **Mögliche Konsequenzen / Änderungen:** ----- | |
| **Spezielles:** ----- | |

### 9.6.3.2 Testprotokoll: Prüfung auf Anwendung von QoS auf Storage

| | |
|---|---|
| **Testname:** TR2-T0017 | **Abhängigkeiten:** Zwei VM's erforderlich |
| **Parameter:** Funktionierender Cluster mit 2 VM's mit unterschiedlichen QoS Profilen. Die Genauen Spezifikationen des QoS sind unter Punkt 8.9.1.1.3 nachlesbar. | **Verwendete Tools:** SSH, Script disktest.sh |

**Ablauf des Tests:** Mit diesen beiden VM's; DYN_HIGH_openSUSE-42.1_DA-VM-2 und DYN_HIGH_openSUSE-42.1_DA-VM-3 soll das eigens hierfür erstellte Stresstest Skript **disktest.sh** ausgeführt werden. Dabei gelten folgende Limitierungen für die VM's:

- DYN_HIGH_openSUSE-42.1_DA-VM-2 → läuft auf ovn1.mgmtdom und ist komplett unlimitiert.
- DYN_HIGH_openSUSE-42.1_DA-VM-3 → läuft auf ovn1.mgmtdom und ist Storag- seitig auf 30% limitiert.

Beide Test werden nacheinander durchgeführt, um die gleichen Host – Bedingungen zu erzeugen.

**Messpunkte / Standort der Messung:** Die Verifikation kann von jedem beliebigen Standpunkt im Netzwerk realisiert werden, wo eine Konsole (SSH) vorhanden ist.

**Verkettung / Ähnlich zu:** -----

| | |
|---|---|
| **Erwartetes Ergebnis:** Das Script startet 8 Prozesse gleichzeitig für 30 Sekunden, welche aus **/dev/zero** lesen und auf die Disk schreiben. Dabei vergleicht es die Grösse der erzeugten acht Files und addiert sie zusammen. Das lesen aus /dev/zero ist ein relativ schneller Prozess, welcher auch schnell Daten zum Schreiben liefert. Innerhalb des gleichen Zeitraumes sollte die limitierte VM wesentlich weniger Daten durch das QoS- limitierte Storageportal liefern können, da das Script per KILL – Befehl beendet wird. | **Modifikationen Testablauf:** ----- |

**Spezielles:** Die Messung des Storagezugriffs über die Datenmenge in MB mag an dieser Stelle ungewöhnlich wirken, jedoch kann dies hier angewendet werden, da das Script seine Workerprozesse per Kill beendet. Da weniger Daten durch die reduzierte Leitung gehen können, werden diese im virtuellen Disk – Cache längerfristig gespeichert. Wenn das Script dann abrupt abbricht, ist die Diskrepanz auf eine konstante Zeit gerechnet, der Wert welcher durch das QoS ausgebremst wurde. So kann nicht direkt die Leitungsbegrenzung auf 30% ermittelt werden, jedoch das Vorhandensein einer Diskrepanz welche auf eine Limitierung hindeutet.

```
Beginne Disktest der VM fuer die DA - Dauer 30 Sekunden - 8 Prozesse Nebenlaeufig
================================================================================
2016-02-28--17:19:40

Starte Prozess 1 - Schreibe in File /root/tmp/zero1
Starte Prozess 2 - Schreibe in File /root/tmp/zero2
Starte Prozess 3 - Schreibe in File /root/tmp/zero3
Starte Prozess 4 - Schreibe in File /root/tmp/zero4
Starte Prozess 5 - Schreibe in File /root/tmp/zero5
Starte Prozess 6 - Schreibe in File /root/tmp/zero6
Starte Prozess 7 - Schreibe in File /root/tmp/zero7
Starte Prozess 8 - Schreibe in File /root/tmp/zero8

./disktest.sh: line 100:  1531 Terminated              cat < /dev/zero > ${targetFile4}
./disktest.sh: line 110:  1525 Terminated              cat < /dev/zero > ${targetFile1}
./disktest.sh: line 110:  1527 Terminated              cat < /dev/zero > ${targetFile2}
./disktest.sh: line 110:  1529 Terminated              cat < /dev/zero > ${targetFile3}
./disktest.sh: line 110:  1533 Terminated              cat < /dev/zero > ${targetFile5}
./disktest.sh: line 110:  1535 Terminated              cat < /dev/zero > ${targetFile6}
./disktest.sh: line 110:  1537 Terminated              cat < /dev/zero > ${targetFile7}
./disktest.sh: line 110:  1539 Terminated              cat < /dev/zero > ${targetFile8}

Ende Disktest der VM fuer die DA - Dauer 30 Sekunden
====================================================
2016-02-28--17:20:28

Generierte Gesamtgroesse der Random-Files = 835 M
da-vm-2:~ #
```

*Abbildung 131: Tests / TR2-T0017-1: Die unlimitierte VM konnte 835MB an Daten in 30s zum Storage Transportieren.*

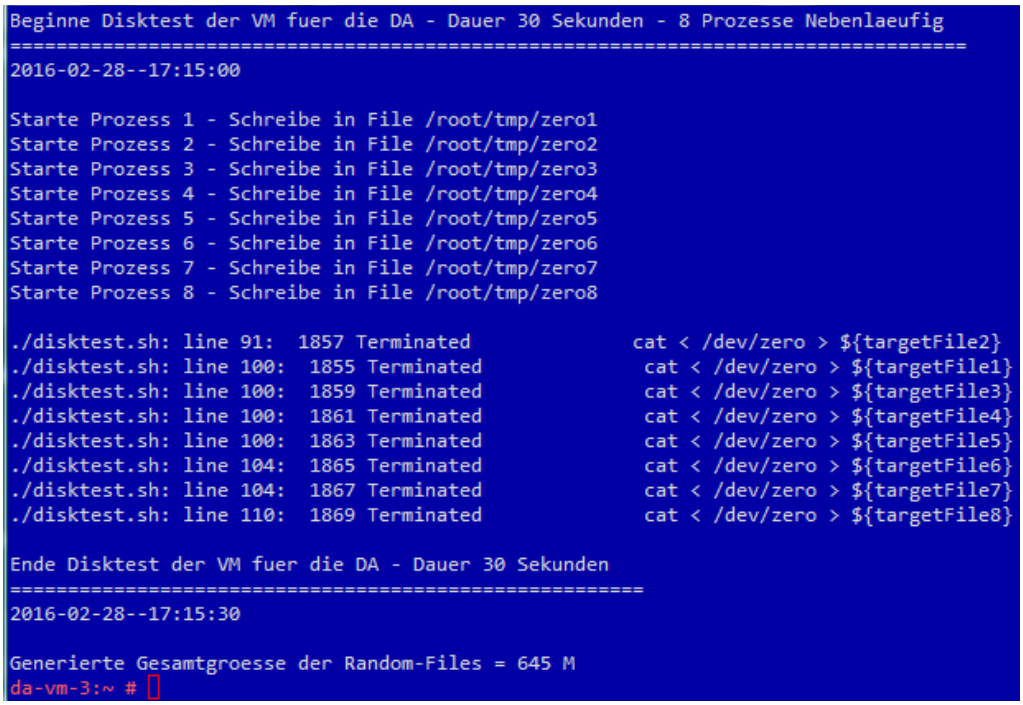



*Abbildung 132: Tests / TR2-T0017-1: Die limitierte VM konnte 645MB an Daten in 30s zum Storage Transportieren.*

| | |
|---|---|
| **Resultat des Tests:** OK | **Bemerkungen:** Diese Test wurden drei mal auf den VM's ausgeführt und brachten auch drei Mal das annähernd gleiche Ergebnis an transportierbaren Daten in MB. |
| **Mögliche Konsequenzen / Änderungen:** ----- | |
| **Spezielles:** Das Script **disktest.sh** ist auf dem Datenträger unter **.../DATA/Tests/Testreihe2/TR2-T0017** zu finden. | |

## 9.7 Auswertung der Testresultate im gesamten

An dieser Stelle soll ein Vergleich mit den im Pflichtenheft definierten Punkten stattfinden. Hier wird explizit das Pflichtenheft als Vergleichspunkt herangezogen, da es sämtliche Anforderungen der Aufgabenstellung im Detail umfasst. Somit soll nachfolgend in tabellarischer Form auf das Pflichtenheft referenziert werden, um sämtliche Punkte mit den vorhandenen Tests zu vergleichen. Es kann Punkte geben welche nicht expliziert getestet, aber vom rein logischen Standpunkt aus implementiert sein müssen, da ansonsten der Aufbau als solches nicht funktionieren würde.

| Funktion / Bedingung | Art der Anforderung | Wurde das Ziel Erreicht | Durch welchen Test | Bemerkungen |
|---|---|---|---|---|
| **Netzwerksegment** | | | | |
| **Link Aggregation** | **Muss Ziel** | **JA** | **TR1-T0003** | Kein direkter Test, aber der durchgeführte Test erzwingt das logische Vorhandensein der Anforderung als zwingende Abhängigkeit. |
| **Vereinigung in Bridge (Switch Nachbildung)** | **Muss Ziel** | **JA** | **TR1-T0003** | Wurde erfolgreich getestet. |
| **Firewall Regelsätze** | **Muss Ziel** | **JA** | **TR2-T0007 TR2-T0008 TR2-T0009 TR2-T0010** | Wurde erfolgreich getestet. |
| **Firewall Regelsätze (Bridge spezifisch)** | **Muss Ziel** | **JA** | **TR2-T0007** | Kein direkter Test, aber der durchgeführte Test erzwingt das logische Vorhandensein der Anforderung als zwingende Abhängigkeit. |

*Tabelle 27: Test: Der direkte Vergleich mit den Anforderungen im Pflichtenheft - Netzwerksegment.*

| Funktion / Bedingung | Art der Anforderung | Wurde das Ziel Erreicht | Durch welchen Test | Bemerkungen |
|---|---|---|---|---|
| **Virtualisierungssegment** | | | | |
| **High Availability (Nachtrag nach Korrektur)** | **Muss Ziel** | **JA** | **TR1-T0004** | Wurde durch Semi-Automatic-HA ersetzt und auch entsprechend getestet. |
| **Last – Verteilung (Nachtrag nach Korrektur)** | **Muss Ziel** | **JA** | **TR2-T0014** | Kein direkter Test, aber der durchgeführte Test erzwingt das logische Vorhandensein der Anforderung als zwingende Abhängigkeit. Hier wird speziell durch Vordefinition der Affinitätsgruppen die Lastverteilung vorgenommen (nicht automatisch im Betrieb). |
| **Affinitäts- Gruppen** | **Muss Ziel** | **JA** | **TR2-T0014** | Wurde erfolgreich getestet. |
| **Live Migration** | **Muss Ziel** | **JA** | **TR2-T0013** | Wurde erfolgreich getestet, in Kombination mit dem Maintenance – Modus |
| **Ressourcenbegrenzung (oVirt - QoS)** | **Muss Ziel** | **JA** | **TR2-T0011 TR2-T0012** | Wurde erfolgreich getestet. |

*Tabelle 28: Test: Der direkte Vergleich mit den Anforderungen im Pflichtenheft - Virtualisierungssegment.*

| Funktion / Bedingung | Art der Anforderung | Wurde das Ziel Erreicht | Durch welchen Test | Bemerkungen |
|---|---|---|---|---|
| **Storagesegment** | | | | |
| **Zentralisierung** | **Muss Ziel** | **JA** | ----- | Kein Test, durch Endergebnis logisch gewährleistet. In Dokumentation beschrieben. |
| **Lastverteilung** | **Muss Ziel** | **JA** | **TR2-T0015 TR2-T0016** | Wurde erfolgreich getestet. |
| **Leistungsoptimierung (Festplatten und Filesystem)** | **Muss Ziel** | **JA** | ----- | Dieser Punkt kann nicht spezifisch getestet werden. Die Erfüllung ist in der Dokumentation als elementarer Teil der Hauptstudie und als Teil der Realisation „Storagesegment“ erklärt. |
| **Ausfallsicherheit** | **Muss Ziel** | **JA** | ----- | Dieser Test ist explizit ausgeschlossen (Punkt 9.4). Seine Erfüllung kann nur theoretisch angenommen werden, ein Realtest kann nur in der Praxis im Ernstfall bestätigt werden. |

*Tabelle 29: Test: Der direkte Vergleich mit den Anforderungen im Pflichtenheft - Storagesegment.*

## 9.8 Beurteilung der Test in Relation mit dem fertigen Produkt

Die hier durchgeführten Tests bestätigen die Funktionsweise des Virtualisierungsclusters und der kompletten Arbeit. Das Endergebnis ist ein funktionierender, robuster und hoch skalierbarer Cluster, welcher alle, seitens des Autors für den privaten Betrieb erforderlichen Kriterien, erfüllt. Zwar lässt sich erst im Produktivbetrieb unter Vollast ein genaueres Urteil erstellen, jedoch sind die bis jetzt mit den Testmaschinen gelieferten Ergebnisse höchst zufriedenstellend. Die Breite an möglichen OS – Varianten im Vergleich zum Vorgängercluster ist weit höher und der Bedienkomfort wesentlich höher. Die Tatsache, dass HA nur über eine halbautomatische Lösung implementiert werden konnte ist zwar höchstens als zufriedenstellend einzustufen, jedoch kann diese Lösung als brauchbar klassifiziert werden.

Abschliessend kann im direkten Vergleich mit der fertigen Lösung und den hier erzielten Testresultaten, aus Sicht des Autors dieser Arbeit das fertige Produkt als getestet und bereit für den Produktivbetrieb klassifiziert werden.

# 10 Dokumentationen und Anleitungen

In den nachfolgenden Punkten soll eine kurze Erklärung bezüglich dieser Dokumentation und ihrem Status als Anleitung allgemein, aber auch spezifisch für Version 3.5 von oVirt folgen.

## 10.1 Diese Dokumentation und ihr Status als Anleitung

Die Arbeit wurde spezifisch als Dokumentation für den offiziellen Auftrag seitens der Höheren Fachschule User erstellt (HFU). Sie beschreibt alle notwendigen Schritte, welche im Pflichtenheft definiert wurden, um ein Projekt nach den Vorstellungen des Autors unter Berücksichtigung der Anforderungen, welche seitens der HFU gestellt wurden, zu realisieren. Mit dieser Arbeit wurde eine Referenzimplementierung eines möglichen Aufbaus beschrieben, welche aber in reeller Betrachtung doch stark vom heute üblichen Standard abweicht. Aus diesem Grund wurde versucht mit einigen hilfreichen Tipps (Infoboxen), eine leichte Annäherung an das heute übliche Design zu erzeugen. Somit ist diese Dokumentation bis zu einem gewissen Grad als vollwertige Installations- und Konfigurationsanleitung zu betrachten, welche helfen soll einen oVirt Version 3.5 Virtualisierungscluster zu erstellen und zu betreiben.

## 10.2 Die externen Dokumentationen

Die Realisierung dieses Projektes traf eine Umbruchphase innerhalb des oVirt eigenen Wiki – Projektes, was dazu führte, dass praktisch alle Suchanfragen über Google und ähnliche Suchmaschinen zwar Treffer lieferten, diese jedoch stets auf eine 404 – Meldung des Webservers liefen. Es war sozusagen nur der Main – Teil des Projektes, welcher sich um die Version 3.6 drehte korrekt erreichbar. Wenn man aber seine Suchanfrage geschickt stellte und die Diskrepanzen zwischen den Versionen 3.5 und 3.6 kannte, konnte man durchaus nützliche Tipps finden. Jedoch zeigt sich ab Mitte Februar eine leichte Besserung, welche darauf hindeutet, dass oVirt seine älteren Versionen doch nicht im Stich lässt. Somit sei mit viel Hoffnung, dass sich dieser zustand noch weiter verbessern wird, auf die oVirt eigene Dokumentation auf **http://www.ovirt.org/documentation/** als weiter Hilfequelle verwiesen.

# 11 Fazits und Abschlussbemerkungen

## 11.1 Risikoeinschätzung zu den Versionen 3.5 und 3.6

Bis Version 3.5 ging oVirt bzw. der federführende Entwickler RedHat einen etwas eigenwilligen Weg in der Entwicklung, welcher das Produkt spezifisch an seine eigenen Bedürfnisse binden sollte. Dies schreckte viele davon ab sich mit oVirt bzw. dem RedHat Virtualization Server zu befassen. Technisch gesehen stellte dies aber nie ein Problem dar, da die Lösung zwar in einigen Punkten von den Konkurrenten abwich, dennoch aber ein stabiles und hoch skalierbars Produkt darstellte. Mit der Version 3.6 versucht RedHat diese Differenz zu den Konkurrenzprodukten zu minimieren und nähert sich den Standard teilweise an. So ist Version 3.6 heute auch offiziell auf Debian lauffähig und es finden sich viele Komponenten, welche eine bessere Integration in die heute üblichen quasi Standards VMware und openStack ermöglichen sollen. Hinzu kommt noch der seitens oVirt zwar nicht offiziell vorhanden, aber dennoch angewendete Long Term Support, welcher auch vom Standpunkt „heute“ betrachtet, bis zu Version 3.2 zurückreicht. Dennoch wird die Verwendung von Version 3.6 empfohlen, da mit ihr ein spürbares umdenken seitens oVirt bzw. RedHat zu erkennen ist, welches stark in Richtung Anpassung des Mainstream geht. Dies ist aus Sicht des Autors die zukünftig sicherere Wahl und erspart einem, den mühsamen Übergang welchen oVirt mit Version 3.6 zu vollziehen zu versucht.

## 11.2 Erhalt des open source Geistes

Um den Geist dieser Arbeit hinsichtlich des im Titel erwähnten open source aufrecht zu erhalten, wurde sie grösstenteils mit Hilfe von open source Tools erstellt. Dabei sind sicherlich die wichtigsten und heute auch als ausgereift zu betitelnden Tools LibreOffice, GanttProject und das online zugängliche draw.io, welches auch als eigenständiges und lokal auszuführendes Chrome Add-On erhältlich ist, verwendet worden.

## 11.3 Abschliessendes Fazit zur Diplomarbeit

Zum Schluss der Arbeit kann der Autor klar sagen, dass es sicherlich eine etwas anstrengendere Arbeit war, was durch die hohe Seitenzahl wahrscheinlich zu erkennen ist. Jedoch ist das Gesamtergebnis bei Betrachtung des fertigen Produktes mehr als zufriedenstellend, da ein akzeptabler Ersatz zur Vorgängerlösung geschaffen werden konnte. Klar ist das Fehlen des vollautomatischen High Availability ein kleiner Wermutstropfen in der Gesamtbetrachtung, jedoch ist es mit der hier gewählten Ersatzlösung weit mehr, als es mit der Vorgängerversion jemals hätte realisiert werden können. Bei Betrachtung des Gesamtergebnisses aus einem etwas entfernteren Blickwinkel, kann das fertige Produkt als ein hoch effizienter, stark skalierbarer und am wichtigsten, funktionierender Virtualisierungscluster eingestuft werden, wo über die vielen Stunden des Suchens nach nicht dokumentierten Lösungen hinweggesehen werden kann. Der Autor kann an dieser Stelle klar sagen, dass er mit dem Ergebnis vollumfänglich zufrieden ist.

# 12 Glossar und Verzeichnisse

## 12.1 Glossar

| Begriff | Erklärung |
|---|---|
| API | Application Programming Interface, ist eine Schnittstelle mit welcher ein Programm, anderen Programen die Anbindung an ein System ermöglicht. |
| Chunk / Chunk Size | In der Elektronikwelt, ein Datenblock in Bezug auf das speichern auf ein Speichermedium. |
| Daemon | Ein Hintergrunddienst in der Unix – Welt, welcher meist Serverfunktionen übernimmt. |
| GUI / WebGUI | Graphical User Interface, ist die Bezeichnung für eine grafische Benutzerschnittstelle, mit der ein User bequem per Menüs Funktionen steuern kann. |
| LOM / LOM - Interface | Lights Out Management, stellt eine hardwarenahe Implementierung dar, welche es dem Benutzer erlaubt, auf ein System (Server) zuzugreifen, auch wenn der Rechner ausgeschaltet ist oder auch kein Betriebssystem auf dem Server installiert ist. |
| Maschine | In der IT umgangssprachlich für Computer/ Server. In dieser Arbeit wird der Begriff häufig für den Verweis auf einen der Server benutzt. Manschmal kann der Begriff auch innerhalb des Kontextes eines Satzes, ausnahmsweise für eine virtuelle Maschine verwendet werden. |
| ncurses | Hierbei handelt es sich um eine frei Bibliothek, welche das Zeichnen von einfachen Menüs innerhalb der Kommandozeile ermöglicht. |
| NIC | Network Interface Card, ist englisch für Netzwerkkarte. |
| Node | Ein Rechnerknoten, welcher Teil eines Verbundes ist und eine spezifische Aufgabe übernimmt. In dieser Arbeit wird der Begriff häufig für einen der Netzwerk-, Virtualisierungs- oder Storagerechner verwendet. |
| RAID | Ein Verbund von physischen Festplatten, welche zum Zweck der Redundanz oder der Geschwindigkeitssteigerung, in diversen Konstellationen miteinander verbunden werden können. |
| RPM | Ein Paketformat für RedHat- basierte Linux Distributionen, um darin enthaltene Software (Binärformat) zu installieren. |
| SPICE | Ein von RedHat entwickeltes Protokoll, welches grafische Weiterleitung von Desktops ermöglicht. Hier kann aber neben dem hochauflösenden Desktop, auch das Audiosignal und eine belibige Anzahl an |
| Systemd | Ein Startinitialisierungsprogramm, welches heute von den meisten Linux Distributionen, zum Starten der einzelnen Dienste, während dem Booten, genutzt wird. |
| Node | Ein Rechnerknoten, welcher Teil eines Verbundes ist und eine spezifische Aufgabe übernimmt. In dieser Arbeit wird der Begriff häufig für einen der Netzwerk-, Virtualisierungs- oder Storagerechner verwendet. |
| VM | Ein Akronym für virtuelle Maschine. |

## 12.2 Abbildungsverzeichnis

## 12.3 Tabellenverzeichnis



## 12.4 Befehls- und Klickverzeichnis

## 12.5 Quellenverzeichnis

Die meistgenutzte Quelle für Suchanfragen war innerhalb dieser Arbeit Google. Somit ist eine genaue Angabe kaum möglich, da es eine Vielzahl an besuchten Seiten gab. Die Seiten, von welchen auch effektiv Inhalt extrahiert wurde, sind auch an den entsprechenden Positionen innerhalb der Arbeit gekennzeichnet. Hier sei als meistgenutzte Quelle die oVirt – Seite unter **http://www.ovirt.org** selbst genannt.

## 12.6 Bücherverzeichnis

Es wurden keine Bücher innerhalb dieser Arbeit als effektive Hilfsmittel genutzt.

# 13 Beilagen

| Beilage | Titel | Inhalt / Bemerkung |
|---|---|---|
| 1 | Datenträger mit nachfolgendem Inhalt | |
| 1.1 | Dokumentationen | In PDF und ODT Format. DOCX ist konvertiert worden, saubere Darstellung in MS Word ist nicht garantiert. Die PDF – Variante wurde modifiziert, indem Blindseiten entfernt wurden. Dies ist ein interner Fehler von LibreOffice und hat keinen Einfluss auf die Integrität dieser Arbeit. Im ODT sind die Blindseiten noch sichtbar. |
| 1.2 | Formelles | Hier ist in PDF Form zu finden, der Antrag des Studenten, der Auftrag seitens HFU und die Betreuungsprotokolle. |
| 1.3 | Tests | Hier sind in div. Unterverzeichnissen die Screenshots der Test (auch Scripts) zu finden, im Falle, dass die Qualität innerhalb der Ausgedruckten Dokumentation nicht ausreicht. |
| 1.4 | addons | |
| 1.4.1 | Experte | Hier finden Sie in div. Unterverzeichnissen die nötige Software und die Zertifikat, welche notwendig sind, um sich mit dem Cluster per OpenVPN zu verbinden |
| 1.4.2 | Betreuer | Hier finden Sie in div. Unterverzeichnissen die nötige Software und die Zertifikat, welche notwendig sind, um sich mit dem Cluster per OpenVPN zu verbinden |
| 1.5 | Fotoalbum | Hier sind einige der Bilder, welche während des Aufbaus des Clusters gemacht wurden. |
| 1.6 | Video | Hier ist ein Video des in Betrieb stehenden Clusters. |
| 1.7 | Gantt | Hier sind in mehreren Unterverzeichnissen, welche den Namen der jeweiligen Monate tragen, jeweils die originalen Guntt – Files und ihre dazugehörigen Exports im PDF und PNG Formaten. |
| 1.8 | generalInsetrs | Hier sind, falls noch im Nachhinein noch notwendig, div. Kleinigkeiten, welche nicht explizit erwähnt worden sind oder allgemein nicht notwendig. Sozusagen, nachträgliche Beilagen im sinne von nicht notwendigen Anhängen. |