HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Software-Defined Infrastructure

Bachelorarbeit HS 2019

Abteilung Informatik

Hochschule für Technik Rapperswil

Autoren:	Pirmin Wenk | Yannick Zwicker
Betreuer:	Prof. Laurent Metzger
Projektpartner:	Führungsunterstützungsbasis (FUB) der Schweizer Armee
Experte:	Marcel Witmer, Cisco Schweiz
Gegenleser:	Prof. Dr. Farhad D. Mehta

Abstract

Ziel dieser Arbeit ist es, ein Proof of Concept einer Automatisierungslösung für die Führungsunterstützungsbasis der Armee zu schaffen, um die Serverlandschaft und insbesondere die Network Services – wie Firewall, DNS und DHCP Server – eines Aussenstandortes automatisiert zu konfigurieren. Die Network Services sollen pro Standort verfügbar sein, damit ein teilautonomer Betrieb in einem Krisenfall möglich ist, wenn die Kommunikation mit dem Headquarter unterbrochen ist.

Zu Beginn der Analyse wurde ein Musterstandort aufgebaut, um die Umsysteme und involvierten Softwarekomponenten und deren Schnittstellen zu eruieren. Zudem wurde ein skalierbares, redundantes Netzwerkkonzept erarbeitet. Aufgrund dieser Analyse konnte der Deployment Prozess evaluiert und in einzelne zu automatisierende Schritte unterteilt werden.

Aufgrund der Analysephase entstand eine skalierende, erweiterbare und auf Docker Containern basierende Automatisierungslösung. StackStorm ist eine Event-Driven Plattform zur Integration und Automatisierung von bestehenden Services und Plattformen. Die Workflow Engine «Orquesta» von StackStorm bietet die Möglichkeit, die komplexe Businesslogik des Deployment Prozesses zu automatisieren. Der Zugriff auf die Umsysteme und deren Konfiguration wird primär mittels des Automatisierungs-Werkzeugs «Ansible» gemacht. Die einzelnen Ansible Playbooks werden durch StackStorm orchestriert.

Es konnte aufgezeigt werden, dass es möglich ist die Network Function Virtualization Infrastruktur und die einzelnen VNFs (Virtualized Network Functions) automatisiert zu deployen. Es hat sich jedoch auch gezeigt, dass sich noch nicht alle eingesetzten Hard- und Softwarekomponenten gleichermassen gut automatisieren lassen.

Durch den Fokus auf eine erweiterbare Architektur und eine modulare Aufbauweise wird eine gute Ausbaumöglichkeit für weitere Workflows gewährleistet.

Inhaltsverzeichnis

1 Aufgabenstellung **8**

2 Management Summary **10**
2.1 Ausgangslage . . . 10
2.2 Vorgehen . . . 11
2.3 Ergebnisse . . . 11
2.4 Ausblick . . . 12

3 Ausgangslage **13**
3.1 Constraints . . . 13
3.1.1 Hardware . . . 13
3.1.2 Software . . . 13
3.2 Abgrenzungen . . . 13

4 Requirements **14**
4.1 Use Cases . . . 14
4.1.1 Aktoren . . . 15
4.1.2 User Stories . . . 15
4.2 Nichtfunktionale Anforderungen . . . 17
4.2.1 Performance Efficiency . . . 17
4.2.2 Security . . . 17
4.2.3 Maintainability . . . 17
4.3 Anforderungen an den Standort . . . 17
4.3.1 Standortgrössen . . . 17
4.3.2 Netzwerkdesign . . . 17
4.3.3 High Availability . . . 17
4.3.4 Standort Autonomie . . . 17

5 Risikoanalyse **18**
5.1 Definition . . . 18
5.2 Risiken . . . 19
5.3 Version 1 - Meilenstein 1 . . . 20
5.4 Version 2 - Meilenstein 3 . . . 20
5.5 Version 3 - Meilenstein 4 . . . 21

6 Software Architektur und Design **22**
6.1 Building Block View . . . 22
6.1.1 Context View . . . 22
6.1.2 Container View . . . 23
6.1.3 Component View . . . 24
6.2 Runtime View - Component Interaction . . . 27
6.2.1 Automation Engine . . . 27
6.2.2 Workflows . . . 27

6.3 Konzepte . . . 31
6.3.1 IP-Adresskonzept . . . 31
6.3.2 Namenskonzept . . . 31
6.3.3 Netzwerkdesign . . . 31
6.4 Technologieentscheidungen . . . 37
6.4.1 Bare-Metal Betriebssysteminstallation . . . 37
6.4.2 SSH Zugriffe . . . 37
6.4.3 Docker . . . 37
6.4.4 NFV . . . 38
6.4.5 IPAM . . . 38
6.5 Architekturentscheidungen . . . 38
6.5.1 Automation Engine . . . 38
6.5.2 High Availability . . . 38
6.5.3 Check Point Firewall-Konzept . . . 39

7 Software Evaluation 41
7.1 Ansible . . . 42
7.2 StackStorm . . . 42
7.3 AWX . . . 42
7.4 Salt . . . 43
7.5 Lizenzierung . . . 43

8 Umsetzung 44
8.1 Infrastruktur Aufbau . . . 44
8.2 Vorbedingungen . . . 44
8.2.1 Infoblox . . . 44
8.2.2 Netzwerk Connectivity . . . 44
8.2.3 CE Router Konfiguration . . . 45
8.2.4 IP-Connectivity . . . 46
8.2.5 Key-management . . . 46
8.2.6 UCS Vorbereitungen . . . 47
8.3 Bare Metal Deployment . . . 47
8.3.1 PXE . . . 47
8.3.2 CIMC CLI . . . 47
8.3.3 ISO-Packer . . . 48
8.4 OVS RPM Packen . . . 48
8.5 Automatisierung . . . 49
8.5.1 Inventory . . . 49
8.5.2 StackStorm . . . 53
8.5.3 Ansible . . . 54
8.5.4 Weitere Packs . . . 55
8.6 Services . . . 56
8.6.1 Check Point . . . 56

8.6.2 NFV Services . . . 60
8.7 Probleme . . . 63
8.7.1 CIMC Image Download . . . 63
8.7.2 Check Point Gateway . . . 63
8.7.3 CIMC . . . 64
8.8 Nächste Schritte . . . 65
8.8.1 CE Router Konfiguration . . . 65
8.8.2 High Availability . . . 65
8.8.3 Inventory Script . . . 65

9 Anhang 66
9.1 Constraints . . . 66
9.2 Projektplanung . . . 66
9.2.1 Projektorganisation . . . 66
9.2.2 Projektphasenplanung . . . 66
9.2.3 Arbeitspakete . . . 67
9.2.4 Planung der Sprints . . . 67
9.3 Projecttracing . . . 76
9.3.1 Dokumentation . . . 76
9.3.2 Zeitauswertung . . . 76
9.3.3 Zielerreichung . . . 78
9.3.4 Fazit . . . 78
9.4 Weitere Abklärungen . . . 79
9.4.1 StackStorm . . . 79
9.4.2 Check Point . . . 80
9.5 Kickstart Files . . . 81
9.6 Anleitungen . . . 81
9.6.1 Installation der Repositories . . . 81
9.6.2 Check Point Management Installation . . . 82
9.6.3 ISR 4461 einrichten . . . 83
9.6.4 Infoblox vorbereiten . . . 86
9.6.5 PXE . . . 92
9.7 Glossar . . . 96
9.8 Abbildungsverzeichnis . . . 97
9.9 Literaturverzeichnis . . . 98
9.10 Danksagungen . . . 101

1 Aufgabenstellung

Aufgabestellung Software-Defined Infrastructure (SDI)

NFV (Network Function Virtualisation) stellt die Netzwerkfunktionen (Routing, Firewall, DNS, DHCP, usw.) in einer virtualisierten Umgebung (virtuelle Maschinen oder Container) bereit.

Das Ziel dieser Bachelorarbeit ist es eine Lösung zu entwickeln, welche automatisiert die für NFVs benötigte Infrastruktur installiert & konfiguriert und gewisse NFVs deployt werden. Dieses automatisierte Deployment soll Faktoren wie die Grösse eines Standortes, an welchem diese NFV Infrastruktur in Zukunft betrieben wird, und Sonderwünsche bezüglich des Vorhandenseins gewisser Netzwerk Funktionalitäten handhaben können. Mittels eines Self-Service-Portals, sollte ein Netzwerk Administrator die NFVs in einem entfernten Standort automatisiert deployen können.

Das Tool muss die folgenden Merkmale enthalten:

Feature 1: Deployment der Infrastruktur	Gegeben sei an einem bestehenden Campus LAN Standort ein mit zwei Server Blades ausgerüsteter Cisco ISR 4461 Router in Betrieb. Diese ISR Router ist mit dem MPLS CE Router verbunden und ist somit vom globalen Netzwerk-Management LAN aus erreichbar. Von da an, kann ein Netzwerk Engineer mit dem zu entwickelnden Tool eine betriebsbereite virtuelle Infrastruktur umsetzen. Es gibt verschiedene Standortgrössen.
Feature 2: Integration vom IPAM	Die IP Adressen (z.B. für die NFVs) werden automatisch im IP Pool des Standorts vom Infoblox bezogen und reserviert.
Feature 3: Umsetzung der Network Services	Als Beispiel NFVs soll ein DNS und ein DHCP Netzwerk Service aufgezogen werden. Der DHCP Service muss für die in Infoblox reservierten IP Ranges konfiguriert werden und dem lokalen Campus und somit den verschiedenen Security Zonen IP Adressen zuteilen können.
Feature 4: Umsetzung der FW und die FW-Rules	Eine virtuelle FW Appliance wird deployt und die FW-Grundregeln sind entsprechend konfiguriert.
Feature 5: Redundantes Deployment	Die zu entwickelnde Lösung muss erweiterbar sein und ein redundantes Deployment innerhalb eines Standortes muss unterstützt werden können.

Mittwoch, 8. Januar 2020 Seite 1 von 2

Die Spezifikationen des Tools setzen sich aus den folgenden Use Cases zusammen:

Use Case	Beschreibung	Muss/optional-Kriterien
UC01 - Workflow-Details anzeigen	Der SDI Orchestration Admin kann über eine graphische Oberfläche detaillierte Informationen über einen ausgeführten Workflow anzeigen und bei Problemen die Logs ansehen.	Muss
UC02 - Infos über deployten Standort abfragen	Das FUB Serverteam und das Netzwerkteam können über eine graphische Oberfläche die Netzwerkinformationen zu den deployten Services anzeigen lassen.	Optional
UC03 - Remote Standort deployen	Das FUB Netzwerkteam kann unter Angabe eines Standortnamens und Auswahl einer Standortgrösse einen neuen Standort deployen.	Muss
UC04 - Login mit eigenem User durchführen	Standortspezifische Applikationen können durch den User ausgewählt und mitdeployt werden.	Optional

08.01.2020, Rapperswil

i.V. P. Schmid

L. METZGER

Mittwoch, 8. Januar 2020 Seite 2 von 2

2 Management Summary

2.1 Ausgangslage

Das FUB Netzwerkteam ist für die Bereitstellung der Netzwerkinfrastruktur an Aussenstandorten zuständig. Bis anhin wird für alle Services, wie Server, Firewall, Router und Switches, jeweils dedizierte Hardware verwendet. Diese muss von Hand verkabelt und konfiguriert werden.

Um die Bereitstellung und Verwaltung der Infrastruktur zu beschleunigen und zu vereinfachen, wird das sogenannte «Converged (Branch) Infrastructure» Modell eingesetzt. Computing, Storage und Networking werden dadurch in einem einzigen System vereint.

Das Ziel ist es, einen teilautonomen Betrieb sicherstellen zu können, indem alle benötigten Services dezentral an jedem Standort vorhanden sind. Dadurch ist es möglich, dass ein Standort autonom funktioniert, auch wenn die Verbindung zum Headquarter unterbrochen ist.

Um alle Services in einer Hardware unterzubringen, wird Ciscos Converged Branch Infrastructure Hardware eingesetzt, um das klassische Datacenter zu vereinfachen. Network Function Virtualization (NFV) Services und weitere Business Applikationen können dadurch auf einer einzigen Hardware zur Verfügung gestellt werden.

Ziel dieser Bachelorarbeit ist es, ein erweiterbares Automatisierungs-Tool zu erstellen, mit dem in einem ersten Prototyp das Deployment der Converged Branch Infrastructure automatisiert wird. Dazu gehört die Provisionierung der Bare-Metal Server mit einer Hypervisor-Technologie als Betriebssystem und die Bereitstellung von Networking und der NFV Services wie Firewall, Domain Name System (DNS) und Dynamic Host Configuration Protocol (DHCP).

Die Schwierigkeit liegt darin, einerseits ein Konzept für den Standort zu entwerfen, welches überhaupt automatisiert werden kann und dennoch möglichst einfach zu deployen ist. Andererseits sind sehr viele unterschiedliche Betriebssysteme und Softwarekomponenten im Einsatz, die im besten Fall gute Application Programming Interfaces (APIs) anbieten, aber teilweise auch sehr umständlich anzusprechen sind.

Abbildung 1: Cisco ISR 4461 mit zwei UCS E-Series Server Blades

2.2 Vorgehen

Am Anfang der Arbeit wurde ein Musterstandort von Hand aufgebaut, um alle dafür benötigten Komponenten zu identifizieren und den Prozessablauf zu verstehen. Die manuelle Installation zeigte, welche Systeme involviert sind, welche Art von Schnittstellen diese anbieten und welche Informationen zu welchem Zeitpunkt benötigt werden. Anschliessend konnte der Prozess Schritt für Schritt automatisiert werden. Die erstellten Konzepte müssen möglichst generisch gehalten werden, damit die Lösung einfach erweitert werden kann.

Aufgrund der identifizierten Schnittstellen und des Prozessablaufs, musste ein geeignetes Framework für die Automatisierung evaluiert werden. Der Deployment-Prozess wurde in einzelne Sub-Workflows aufgeteilt und anschliessend Schritt für Schritt automatisiert.

Als Single-source-of-truth für alle Informationen zu einem Standort dient das IP Address Management (IPAM) System «Infoblox», welches von FUB verwendet wird und auch für die Bachelorarbeit eingesetzt werden muss. Aus diesem werden alle Daten bezogen, aufbereitet, mit weiteren Daten angereichert und anschliessend in einer maschinenlesbaren Form den Workflows übergeben.

Im speziellen liegt ein Augenmerk darauf, dass die Kommunikation immer über verschlüsselte Verbindungen stattfindet, da neben Konfigurationseinstellungen auch Passwörter übertragen werden.

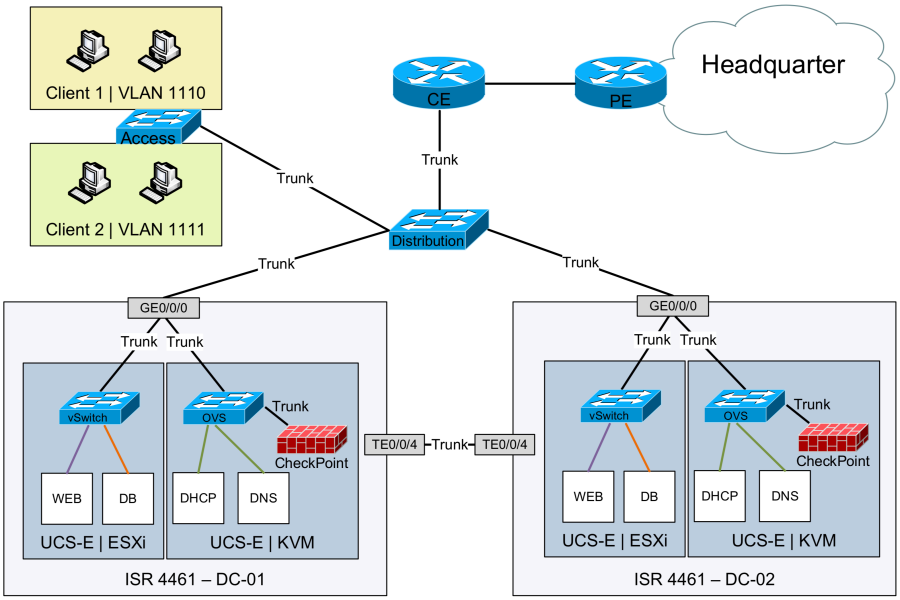


Abbildung 2: Ausgerollter Standort mit redundanter Hardware

2.3 Ergebnisse

Aufgrund der Analysephase haben wir uns für eine auf Workflows basierende Automatisierungslösung entschieden, die aus mehreren Docker Containern besteht und dadurch einfach und schnell zu konfigurieren und deployen ist.

Zur Orchestrierung einzelner Tasks, bis hin zu ganzen Workflows, wird StackStorm eingesetzt. Es bietet die Möglichkeit die Businesslogik in Workflows abzubilden. Da die Workflows in YAML geschrieben werden, sind sie gut lesbar, einfach verständlich und können als Pack in einem Git Repository gespeichert und auch direkt von dort installiert werden. Zudem können weitere Workflows als einzelne YAML-Dateien hinzugefügt werden.

Die Umsysteme werden primär über Ansible Playbooks provisioniert und konfiguriert, da Ansible ohne einen Agent über SSH genutzt werden kann. Die einzelnen Playbooks werden durch

die StackStorm Workflows orchestriert. Dies hat den Vorteil, dass die sehr komplexe Businesslogik im StackStorm abgebildet werden kann und der eigentliche Zugriff auf die Geräte und deren Konfiguration in die Ansible Playbooks ausgelagert ist.

Mit dieser Bachelorarbeit konnten wir aufzeigen, dass die Ressourcen ausreichen, virtuelle Network Function Services, wie eine Check Point Firewall, DNS und DHCP Server auf einem Router-Chassis zu betreiben.

Dafür wurde ein Netzwerkkonzept ausgearbeitet, das zu einem hochverfügbaren Design erweitert werden kann. Es verbindet – intern über den Router – die virtuellen Maschinen der zwei Hypervisor Server.

Neben den Konzepten konnte der gesamte Deployment Prozess für die NFV Services automatisiert werden. Dies beinhaltet das Auslesen und Aufbereiten der benötigten Informationen aus dem IPAM, die Konfiguration des Routers, die Bare-Metal Installation der zwei Hypervisor Server, die Konfiguration des KVM Hypervisor Servers und zum Schluss das Deployment der NFV Services (Check Point Firewall, DNS und DHCP Server) als virtuelle Maschinen.

Es hat sich gezeigt, dass vor allem die Installationsprozesse der Betriebssysteme teilweise noch nicht sehr gut für die Automatisierung ausgelegt sind. Sobald jedoch eine Linux Bash über SSH oder eine Web API zur Verfügung steht, ist vieles direkt einfacher zu automatisieren.

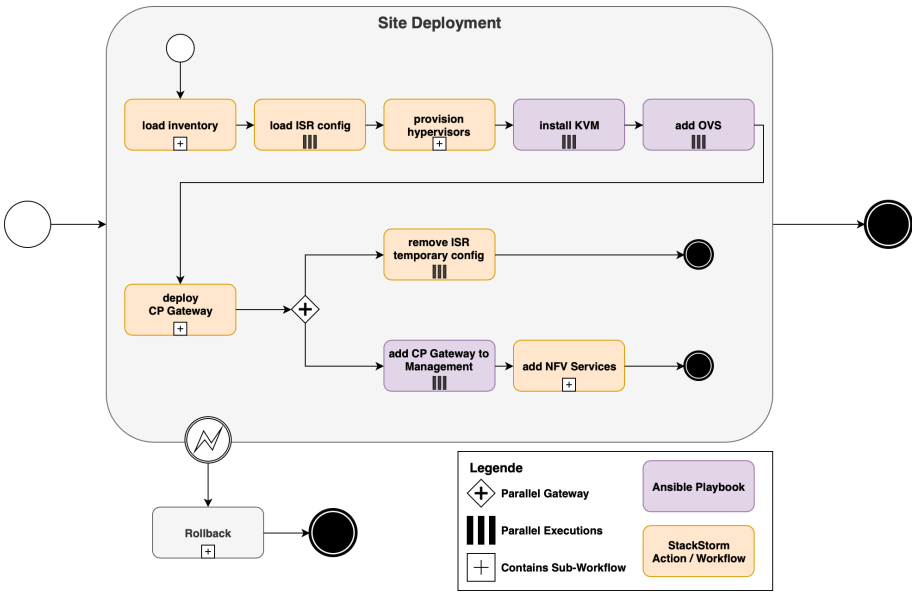


Abbildung 3: Deployment Prozess für einen Aussenstandort

2.4 Ausblick

Der bereits umgesetzte Deployment Prozess für einen kleinen Standort mit nur einem Cisco ISR 4461 ist bereits dafür ausgelegt, mit wenig Änderungen auch einen grösseren Standort mit einer Redundanz ausrollen zu können.

Zudem kann über weitere, in YAML spezifizierte Workflows das Deployment auch für andere Hardware für noch grössere Standorte erweitert werden.

In einem nächsten Schritt und für den produktiven Einsatz müssen die einzelnen Konfigurationen des Workflows, wie z.B. die Konfiguration der konkreten Firewall Regeln, auf die kundenspezifischen Anforderungen angepasst werden.

3 Ausgangslage

Um alle Services in einer Hardware unterzubringen, wird Ciscos Converged Branch Infrastructure Hardware der Serie ISR 4000 eingesetzt. Ein Cisco ISR 4461 dient unter anderem als Chassis für zwei UCS E-Series Blade Server, die dazu eingesetzt werden, um NFV Services und weitere Applikationen an einem Aussenstandort zur Verfügung zu stellen.

3.1 Constraints

Diese Arbeit ist gedacht als PoC (Proof of Concept) und dient dazu, das Szenario möglichst praxisnahe umzusetzen, sodass es später (oder Teile davon) im produktiven Umfeld eingesetzt werden kann.

3.1.1 Hardware

Es stehen zwei Cisco ISR 4461 von der Führungsunterstützungsbasis (FUB) zur Verfügung. Die restliche Hardware für ein Lab-Aufbau wird vom Institute for Networked Solutions der HSR (INS) zur Verfügung gestellt.

3.1.2 Software

IP Address Management

Da die FUB Infoblox als IPAM verwendet, wird dieses auch als Single-source-of-truth genutzt. Ein entsprechendes System wird ebenfalls vom INS zur Verfügung gestellt.

Firewall

Als Firewall wird eine virtuelle Check Point eingesetzt. Die ISO Dateien für das Management und das Gateway wird von Check Point in der Version R80.30 inkl. Evaluationslizenzen zur Verfügung gestellt.

Hypervisor

Das kleinere UCS E-Series Blade wird als Hypervisor für die NFV Services verwendet. Als Hypervisor wird KVM verwendet. Auf dem grösseren Server wird VMware ESXi als Hypervisor verwendet, der in ein bestehendes vCenter im Headquarter eingebunden werden kann.

3.2 Abgrenzungen

IPv6 ist nicht Bestandteil der Bachelorarbeit.

Ein High Availability (HA) Szenario soll in Betracht gezogen und allenfalls schematisch eingeplant werden, jedoch ist die Automatisierung des HA-Szenarios nicht Teil dieser Arbeit.

4 Requirements

4.1 Use Cases

Das Use Case Diagramm zeigt die Aktoren und Umsysteme der Softwareumgebung und die darin enthaltenen Tätigkeiten, die das System gegenüber den Nutzern leisten soll.

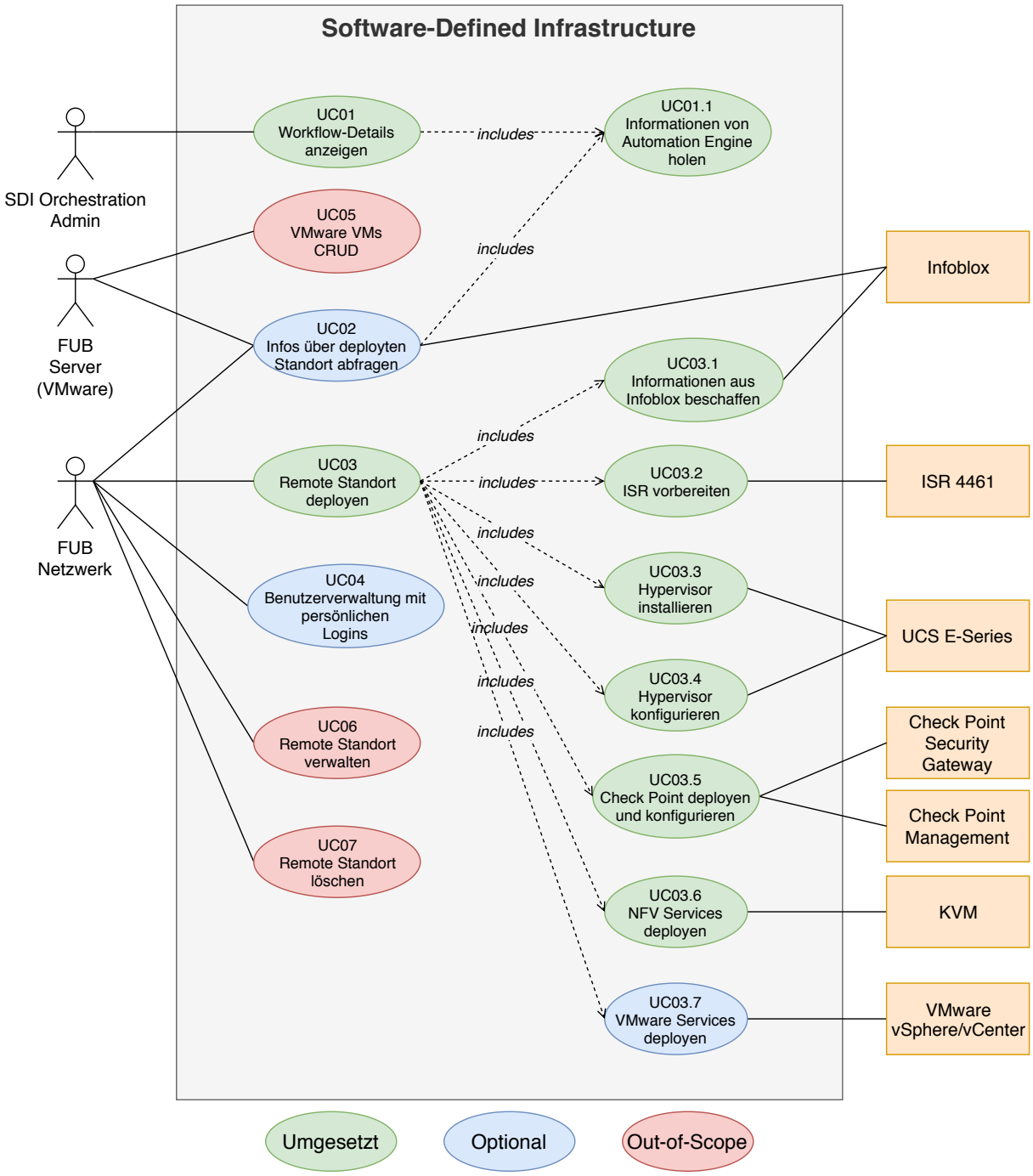


Abbildung 4: Use Case Diagramm

4.1.1 Aktoren

Die folgenden Aktoren interagieren mit dem SDI Orchestration Tool.

FUB (VMware) Server Team Das Server Team ist zuständig für die VMware ESXi Server und die darauf deployten Services.

SDI Orchestration Admin Der Orchestration Admin hat mehr Rechte und verfügt zudem über das Know-how, die Workflows zu Überwachen und bei Problemen erste Informationen zu sammeln.

FUB Netzwerk Team Das Netzwerk Team ist zuständig für die Bereitstellung und Wartung der Netzwerkinfrastruktur an einem neuen Standort.

4.1.2 User Stories

Beim Gespräch mit dem Auftraggeber sind einige Anforderungen an das System zusammengekommen, welche im Folgenden beschrieben sind.

UC01 - Workflow-Details anzeigen

Der SDI Orchestration Admin kann über eine graphische Oberfläche detaillierte Informationen über einen ausgeführten Workflow anzeigen und bei Problemen die Logs ansehen.

UC01.1 - Informationen von der Automation Engine holen

Die Automation Engine stellt die Informationen und Logs eines Workflows über eine API oder direkt als User Interface zur Verfügung.

UC02 - Infos über deployten Standort abfragen

Das FUB Serverteam und das Netzwerkteam können über eine graphische Oberfläche die Netzwerkinformationen zu den deployten Services anzeigen lassen.

UC03 - Remote Standort deployen

Das FUB Netzwerkteam kann unter Angabe eines Standortnamens und Auswahl einer Standortgrösse einen neuen Standort deployen.

UC03.1 - Infoblox Informationen holen

Infoblox dient als IPAM und Inventar. Die benötigten Informationen zu einem Standort werden vom Infoblox bezogen und neue dort gespeichert.

UC03.2 - ISR vorbereiten

Der Router wird nur mit IP Connectivity und einer Grundkonfiguration über einen vorherigen Prozessschritt ausserhalb des SDI vorbereitet. Die komplette individuelle Netzwerkkonfiguration für den Standort wird durchgeführt.

UC03.3 - Hypervisor installieren

Auf den zwei physischen UCS E-Series Blade Server wird jeweils über eine Bare-Metal Installation das Betriebssystem des Hypervisors installiert.

UC03.4 - Hypervisor konfigurieren

Die Hypervisor Server bekommen eine spezifische Netzwerk- und Storage-Konfiguration und werden unter anderem mit Benutzerdaten und Passwörtern konfiguriert. Zusätzlich werden die nötigen Softwarepakete installiert und konfiguriert.

UC03.5 - Firewall deployen und konfigurieren

Das Gateway der Check Point Firewall wird als virtuelle Maschine deployt und konfiguriert, sodass diese nur noch den nötigen Traffic über das Netzwerk zulässt.

UC03.6 - NFV Services deployen

Weitere NFV Services, wie DHCP- und DNS-Server, werden in einer eigenen Firewall-Zone deployt.

UC03.7 - VMware Services deployen - optional

Standortspezifische Applikationen können durch den User ausgewählt werden. Alle ausgewählten Services werden ebenfalls deployt.

Dieser Use Case wurde aufgrund der Zwischenpräsentation und der anschliessenden Sitzung mit dem Industriepartner und dem Betreuer als «optional» definiert.

UC04 - Benutzerverwaltung mit persönlichen Logins - optional

Die User sollen sich mit eigenen Accounts und unterschiedlichen Rechten einloggen können.

UC05 - VMware VMs CRUD – out-of-scope

Es ist denkbar, dass die virtuellen Maschinen für VMware auch über dieselbe Automatisierung deployt und verwaltet werden können. Dies ist aber im Rahmen dieser Arbeit nicht vorgesehen.

UC06 - Remote Standort verwalten – out-of-scope

Änderungen an einem Standort können automatisiert durchgeführt werden. Dies beinhaltet auch die Konfiguration von zusätzlicher oder ersetzter Hardware, die ausgetauscht oder ersetzt werden muss.

UC07 - Remote Standort löschen – out-of-scope

Ein Standort kann wieder gelöscht werden, wenn dieser aufgelöst wird. Dabei werden sämtliche Informationen aufgeräumt und Daten wo nötig vernichtet.

4.2 Nichtfunktionale Anforderungen

4.2.1 Performance Efficiency

- Workflows sind teilweise parallel ausführbar, wodurch Zeit bei der Durchführung eingespart werden kann.

4.2.2 Security

- Sämtliche Kommunikationsschnittstellen müssen eine verschlüsselte Kommunikation unterstützen.
- Ein Sicherheitskonzept (Authentifizierung, Autorisierung) soll miteingeplant und von der Architektur unterstützt werden.

4.2.3 Maintainability

- Die Workflows sind modular aufgebaut und können erweitert werden.
- Die Workflows werden in einem Repository verwaltet, von wo sie auf der Automatisierungsengine installiert werden können.
- Es werden aktuelle Technologien verwendet. Eine Weiterentwicklung der Software soll aktuell und, wo immer möglich, nicht älter als ein Jahr sein.

4.3 Anforderungen an den Standort

4.3.1 Standortgrössen

Die Hardware an einem Standort ist je nach Standortgrösse unterschiedlich. Ein «XS» Standort besteht lediglich aus einem Cisco ISR 4461. Ein «S» Standort wäre z.B. mit zwei redundanten Cisco ISR 4461. An einem noch grösseren Standort kommt z.B. ein Cisco Hyperflex zum Einsatz, wodurch noch mehr Ressourcen zur Verfügung stehen.

Das Deploymenttool muss so ausgelegt sein, dass die Workflows erweitert werden können, um den unterschiedlichen Anforderungen gerecht zu werden.

4.3.2 Netzwerkdesign

Das Netzwerkdesign soll so ausgelegt sein, dass es einen möglichst kleinen Unterschied macht, ob an einem Standort nur ein Cisco ISR 4461 oder zwei zum Einsatz kommen. Das Design soll zudem skalierbar sein, indem z.B. an einem Standort zusätzliche Firewall-Zonen z.B. für weitere Servernetze deployt werden können.

4.3.3 High Availability

Je nach Standortgrösse müssen die Komponenten ausfallsicher betrieben werden.

- Hypervisor Cluster
- Firewall Cluster
- NFV Services (active/standby)

Wichtig ist zudem, dass die Datenpersistenz sichergestellt ist.

4.3.4 Standort Autonomie

Das ganze Design soll möglichst so ausgelegt sein, dass ein Standort autonom funktionieren kann. Wenn ein Standort die Verbindung ins Headquarter verliert, müssen die NFV Services und die Applikationen auf dem ESXi Hypervisor trotzdem noch funktionieren. Das heisst, dass z.B. Netzwerkservices wie DNS und DHCP pro Standort lokal betrieben werden müssen.

5 Risikoanalyse

5.1 Definition

Die Risikoanalyse wurde in mehreren Schritten durchgeführt. Die Schritte umfassen:

Risikoidentifikation Zuerst müssen die Risiken ausfindig gemacht werden.

Risikoanalyse Die Analyse der Risiken ist notwendig, um zu definieren, mit welcher Priorität die einzelnen Risiken behandelt werden sollen. Die Bewertung umfasst die quantitative Eintrittswahrscheinlichkeit (0–100%) und die Auswirkung beim Eintritt eines Risikos (in Arbeitsstunden). Wenn die Eintrittswahrscheinlichkeit nur qualitativ gemacht werden kann, wird das entsprechende Mapping der Auflistung unten genommen, wobei der Mittelwert der Quantitativen Bewertung genommen wird (z.B. 70% für eine grosse Eintrittswahrscheinlichkeit).
Die Eintrittswahrscheinlichkeiten sind so zu lesen:

- 0–20%: sehr tiefe Eintrittswahrscheinlichkeit (0 bis 1 erwartetes Ereignis)
- 20–40%: geringe Eintrittswahrscheinlichkeit (ca. 1 erwartetes Ereignis)
- 40–60%: mittlere Eintrittswahrscheinlichkeit (2–3 erwartete Ereignisse)
- 60–80%: grosse Eintrittswahrscheinlichkeit (4–6 erwartete Ereignisse)
- 80–100%: sehr hohe Eintrittswahrscheinlichkeit (> 6 erwartete Ereignisse)

Risikoplanung Die Risikoplanung umfasst verschiedene Kategorien [3]:

Risikovermeidung Das Risiko soll vollkommen beseitigt werden. Dies ist meist nur zu erreichen, indem ein Teilprojekt oder eine Technologie komplett weggelassen wird.

Risikominderung Dabei sollen Massnahmen ergriffen werden, sodass die Eintrittswahrscheinlichkeit für ein Risiko minimiert wird.

Risikobegrenzung Hier ist das Ziel, den potenziellen Schaden beim Eintritt eines Risikos zu minimieren.

Risikoverlagerung Durch die Verlagerung wird das Risiko komplett auf Dritte abgewälzt. Das Risiko wird bei Risikoverlagerung nicht beseitigt, sondern lediglich die finanziellen Auswirkungen abgeschwächt. Es gibt Risiken, die nicht verlagert werden können (z.B. Imageschaden).

Risikoakzeptanz Das Risiko wird bewusst in Kauf genommen. Dafür können z.B. Puffer geplant werden, die die nötige verlorene Zeit beim Eintritt eines Risikos im Voraus miteinberechnen.

5.2 Risiken

R1 Bei der Automatisierung kann aufgrund einer Inkompatibilität oder Mangel an Funktionalität die Vollautomatisierung nicht komplett durchgeführt werden.

R1.1 Das «unattended» UCS Bare-Metal Deployment ist nicht möglich, weil es das Cisco UCS / CIMC nicht zulässt. Das Deployment des Betriebssystems müsste somit allenfalls manuell vor dem Versand der Hardware erfolgen.
Risikovermeidung / Risikobegrenzung: Durch viele Tests und Versuche wird sich zeigen, was alles möglich ist. Alles was nicht möglich ist, braucht eine alternative Lösung.

R1.2 Die Automatisierungslösung unterstützt nicht alle benötigten Features. Dadurch lassen sich einige Tasks nicht oder nur teilweise automatisieren.
Risikovermeidung / Risikominderung: Durch eine genaue Analyse / Evaluation der Automatisierungssoftware soll ein Eintrittsfall minimiert werden. Allenfalls sollen Module weggelassen oder auf eine alternative Software zurückgegriffen werden. Ansonsten soll nach alternativen Möglichkeiten gesucht werden.

R1.3 Das Firewall Deployment lässt sich nicht vollständig automatisieren, weil diese keine unattended Installation unterstützt.
Risikominderung: Durch eine manuelle Installation muss der Deployment Prozess analysiert werden und aufgrund dessen soll eine geeignete Möglichkeit der Automatisierung erarbeitet werden. Ansonsten muss nach alternativen Möglichkeiten umgeschaut oder mit dem Hersteller eine Lösung gesucht werden.

R2 Die Beschaffung der Software (Betriebssysteme, Lizenzen etc.) kann sich verzögern, wodurch die Arbeiten verschoben werden müssen.
Risikobegrenzung / Risikoakzeptanz: Wenn der Schaden nicht minimiert werden kann, soll ein Teil der nötigen Zeit mit in die Planung eingerechnet werden.

R3 Das Netzwerk kann unter gewissen Bedingungen nicht so umgesetzt werden, wie wir es geplant haben.

R3.1 High Availability ist nur eingeschränkt möglich.
Risikobegrenzung: Ein alternatives Konzept entwerfen.

R3.2 Unser erarbeitetes Design kann nicht oder nur teilweise umgesetzt werden, da es nicht den Vorstellungen des Auftraggebers entspricht.
Risikobegrenzung: Ein alternatives Konzept entwerfen und regelmässig präsentieren.

R4 Der zur Verfügung gestellte Router hat nicht genügend Ressourcen.
Risikoakzeptanz: Versuchen, die benötigten Ressourcen einzugrenzen oder Ersatzhardware in Betracht ziehen.

Das Risiko ist folgend in verschiedenen Versionen vorhanden, welche jeweils eine Analyse zu einem gewissen Meilenstein darstellen.

Die maximale Auswirkung des Ereignisses zeigt das geschätzt höchste Schadenspotential in Stunden. Die Grösse der Blase zeigt den gewichteten Schaden im Verhältnis zu den anderen Risiken.

5.3 Version 1 - Meilenstein 1

Der gewichtete Schaden beläuft sich auf 56 Stunden.

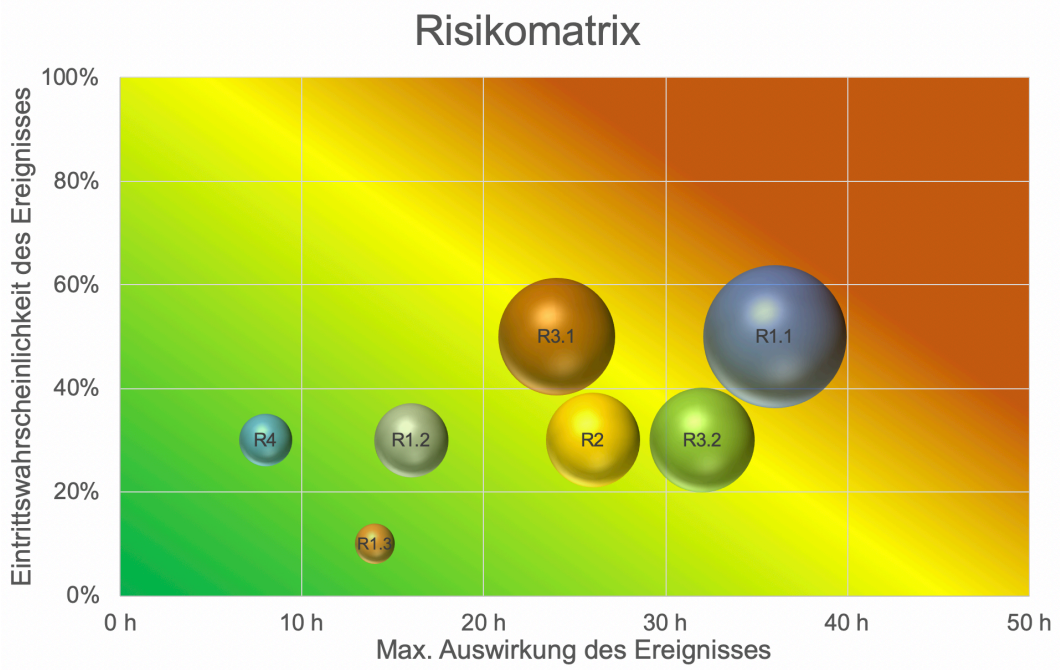


Abbildung 5: Risikomatrix Version 1

5.4 Version 2 - Meilenstein 3

Der gewichtete Schaden beläuft sich noch auf 18 Stunden. Einige Risiken konnten bereits beseitigt werden.

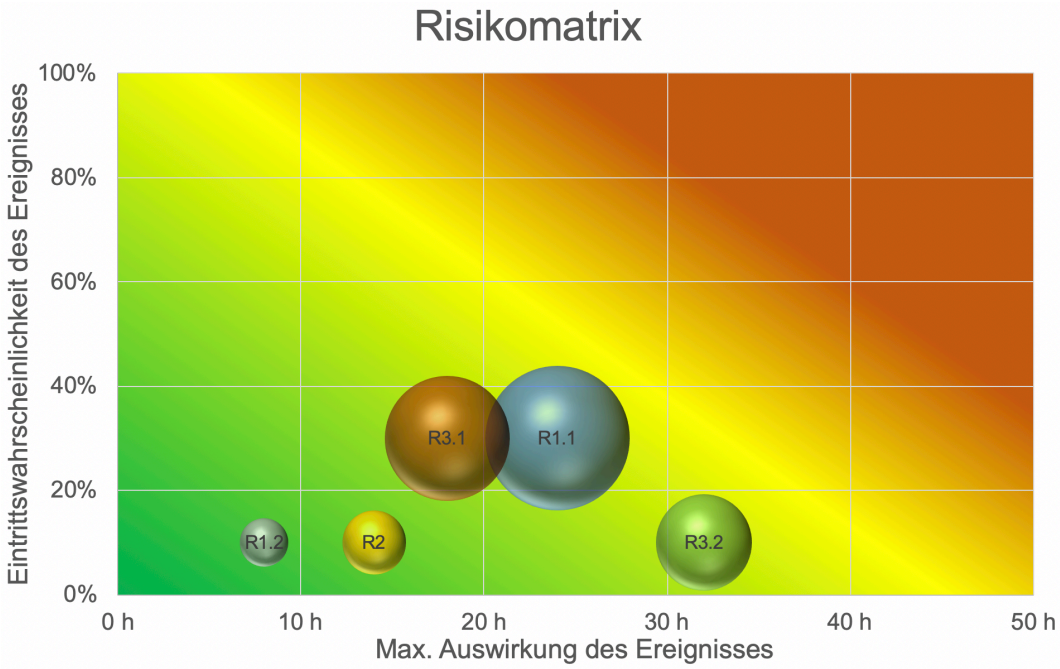


Abbildung 6: Risikomatrix Version 2

5.5 Version 3 - Meilenstein 4

Zum Beginn der Entwicklung (Construction-Phase) sind alle Risiken minimiert und geplant. Das heisst diese sind entweder beseitigt oder bei der Planung mit eingerechnet.

Aus unserer Sicht können folgende Risiken noch eintreten:

R3.1 Unser erarbeitetes theoretisches Konzept wurde konnte nicht vollständig auf der Hardware geprüft werden und es ist deshalb noch nicht sicher, ob es so mit den Cisco UCS-Servern im HA-Betrieb noch nicht voraussehbare Probleme gibt.

R3.2 Wir haben unsere Entwürfe mehrmals mit dem Auftraggeber besprochen, jedoch bleibt das Restrisiko, dass das Konzept unerwünschte Stellen aufweist.

Abklärungen

R1 Bei der Automatisierung kann aufgrund einer Inkompatibilität oder Mangel an Funktionalität die Vollautomatisierung nicht komplett durchgeführt werden.

- **R1.1** Zwei unterschiedliche Lösungskonzepte wurden erarbeitet und getestet (Bare Metal Deployment, Seite 47).
- **R1.2** StackStorm ist einfach erweiterbar mit sogenannten Packs aus der Community (https://exchange.stackstorm.org/) oder mit selbst geschriebenen StackStorm Python Actions (CIMC, Seite 37).
- **R1.3** Wir haben verschiedene Varianten ausprobiert und uns für eine geeignete davon entschieden (Check Point, Seite 56).

R2 Die Software und Hardware konnte mehrheitlich pünktlich, noch während der Elaboration Phase, organisiert und bereitgestellt werden.

R3 Das geplante Netzwerk kann unter gewissen Bedingungen nicht so umgesetzt werden.

- **R3.1** Es wurde ein Netzwerkkonzept erarbeitet, welches auch für ein High Availability Design anwendbar ist.
- **R3.2** Wir haben mehrfach mit dem Auftraggeber die Details und unsere Konzepte besprochen.

R4 Durch das Deployment mehrerer VMs konnte während der Arbeit aufgezeigt werden, dass genügend Ressourcen für die NFV Services zur Verfügung stehen.

6 Software Architektur und Design

6.1 Building Block View

6.1.1 Context View

Die Context View ist eine sehr abstrakte Sichtweise auf die Automatisierungslösung. Die Aktoren (siehe Aktoren, Seite 15) greifen auf die Software zu. Die folgenden Umsysteme werden von der Automatisierungslösung angesprochen.

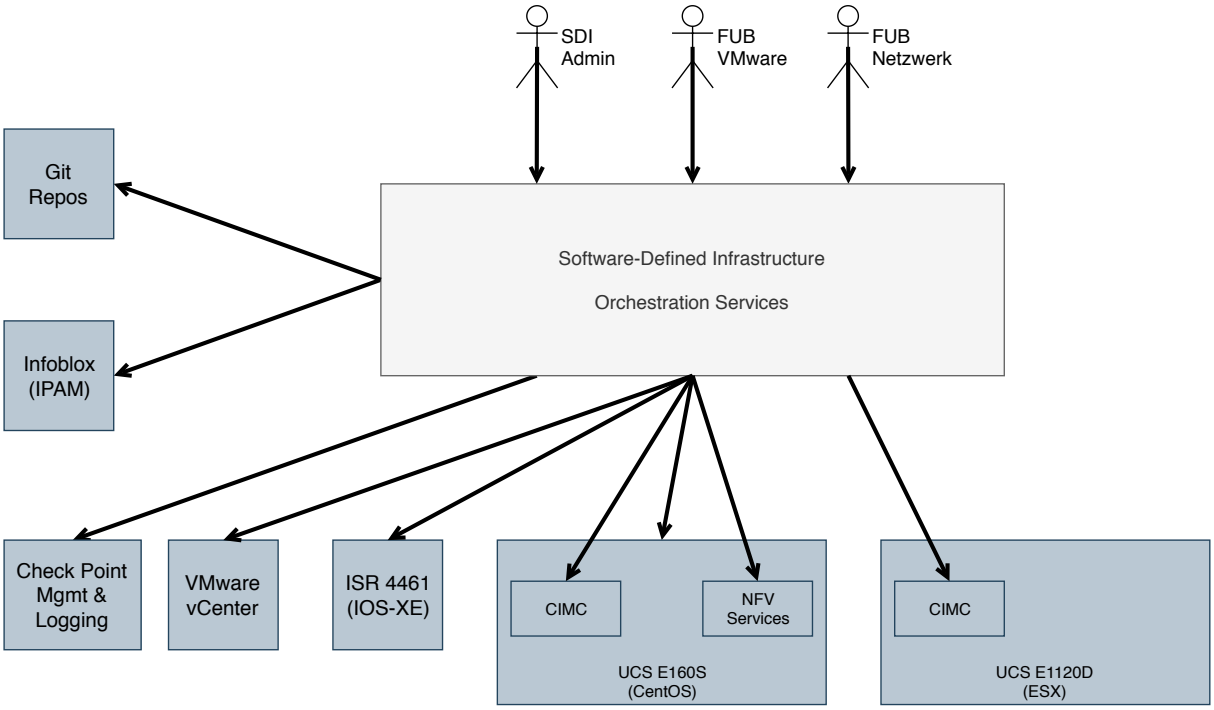


Abbildung 7: Context Diagram

Git Repositories: Die Software, Workflows sowie die auf Docker basierende Infrastruktur befindet sich in Git Repositories.

Infoblox: Im Infoblox werden sämtliche IP-Adressen und dazugehörige Hostnames für einen Standort gespeichert. Das Infoblox dient als «Single Source of Truth».

Check Point Management: Die Check Point Firewall besteht aus einem Management und dedizierten Gateways. Die Gateways werden über das Management verwaltet. Das Management bietet dazu eine Web API an, um dies zu automatisieren.

VMware vCenter: Die ESXi Hypervisor werden durch das VMware vCenter verwaltet. Zudem werden über dieses auch die virtuellen Maschinen verwaltet. Es bietet ebenfalls eine API an.

ISR 4461 (IOS-XE): Der Router ist mit einer Grundkonfiguration versehen. Während des Deployments müssen standortspezifische Einstellungen auf dem IOS-XE Betriebssystem des Routers konfiguriert werden.

CIMC: Über das CIMC können die UCS Server mit einem Betriebssystem provisioniert werden. Zudem können weitere BIOS Einstellungen, wie die Boot-Reihenfolge konfiguriert werden.

CentOS/KVM: Für den KVM Hypervisor bietet ein Linux Betriebssystem die Grundlage. Über dieses werden die NFV Services als virtuelle Maschinen deployt.

NFV Services: Die NFV Services wie Firewall, DNS und DHCP benötigen pro Standort eine individuelle Konfiguration.

6.1.2 Container View

Die Container View entspricht der Deployment View und zeigt, wie die Software physisch verteilt wird und wie die Container untereinander kommunizieren. Ein Container entspricht in unserem Fall direkt auch einem Docker Container.

Im Speziellen wurde darauf geachtet, dass die Gesamtheit an Services über verschlüsselte Verbindungen untereinander kommunizieren.

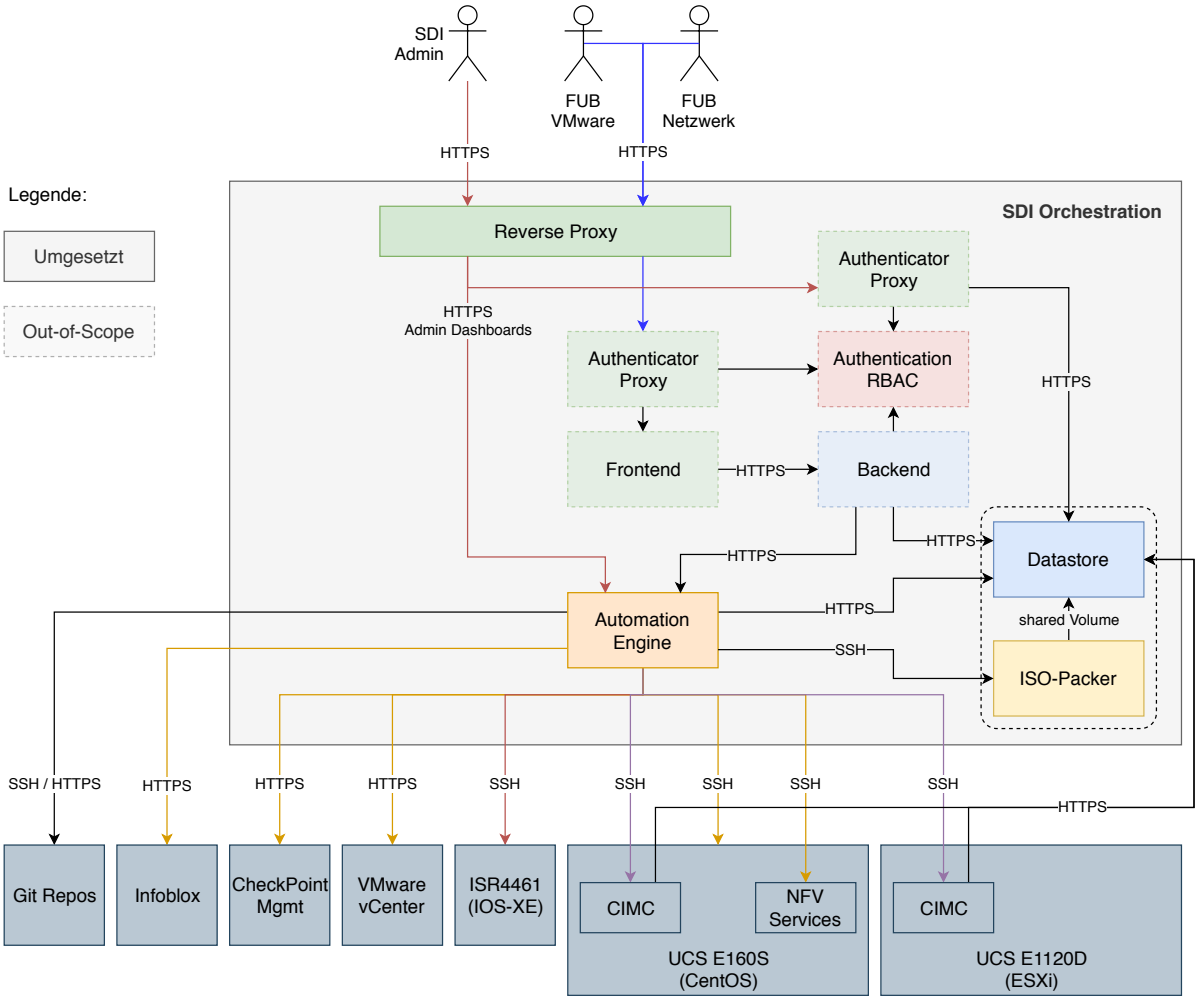


Abbildung 8: Container/Deployment Diagram

6.1.3 Component View

Durch die Component View sieht man in die einzelnen Container, wie sie aufgebaut sind und was für Komponenten enthalten sind.

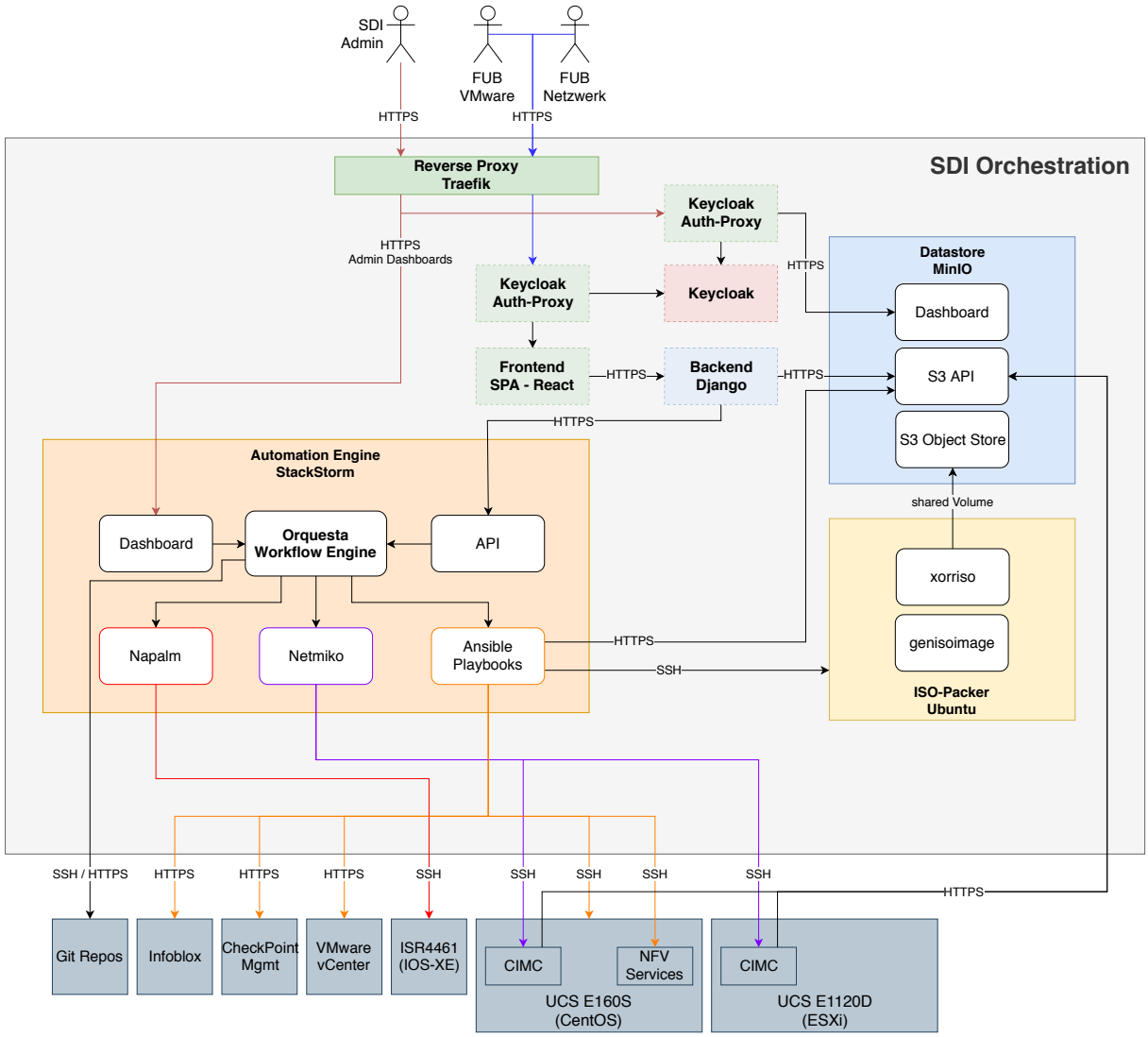


Abbildung 9: Component Diagram

Eine CRC Card (Component-Responsibility-Collaboration Karte) beschreibt die jeweiligen Verantwortlichkeiten (Responsibilities) und die Abhängigkeiten (Collaborators) einer Komponente und zeigt mögliche Umsetzungsvarianten oder bereits bekannte Implementationen auf. Im Folgenden werden für die einzelnen Komponenten eine solche CRC-Card aufgeführt.

Automatisierungstool

Komponente: Automation Engine	
Verantwortlichkeiten: - Orchestriert Workflows und Tasks - Konfiguriert die Services	**Abhängigkeiten:** Schnittstellen von: - Frontend - Direkter Benutzerzugriff via auth Proxy Schnittstellen zu: - Git Repos - Infoblox IPAM - Check Point Management - VMware vCenter - Cisco ISR 4461 - UCS - CIMC - NFV Services - KVM Hypervisor - Datastore API - ISO Packer
Mögliche Implementationen (und bekannte Anwendungsfälle): - StackStorm mit Ansible und weiteren Packs - Ansible AWX mit Ansible-Playbooks - SaltStack	

Datenspeicher

Komponente: Datastore	
Verantwortlichkeiten: - Speichert Betriebssystem Images (ISO) - Speichert Linux Softwarepakete (RPM) - Speichert Betriebssystem Disks (QCOW) - Stellt diese Dateien über eine sichere HTTPS API zur Verfügung	**Abhängigkeiten:** Schnittstellen von: - Direkter Benutzerzugriff via auth Proxy - Backend - Automation Engine - CIMC Image Download Spezielle Schnittstellen: - Shared Volume mit ISO-Packer
Mögliche Implementationen (und bekannte Anwendungsfälle): - MinIO S3 Object Store - Flask	

Authentifizierung

Komponente: Authentication	
Verantwortlichkeiten: - Authentifiziert einen Benutzer - Autorisiert Rechte rollenbasiert - Schützt die APIs vor unbefugtem Zugriff	**Abhängigkeiten:** Schnittstellen von: - Reverse Proxy Schnittstellen zu: - Frontend - Datastore
Mögliche Implementationen (und bekannte Anwendungsfälle): - Keycloak - Implementation im Frontend / Backend	

Benutzerinterface

Komponente: Frontend	
Verantwortlichkeiten: - Bietet grafische Unterstützung beim Standort Deployment - Zeigt die Informationen über einen deployten Standort übersichtlich auf	**Abhängigkeiten:** Schnittstellen von: - Authentication Schnittstellen zu: - Backend
Mögliche Implementationen (und bekannte Anwendungsfälle): - React.js - Vue.js	

Backend

Komponente: Backend	
Verantwortlichkeiten: - Schützt die API-Keys vor dem Frontend und somit vor dem User - Stellt dem Frontend eine API zur Verfügung	**Abhängigkeiten:** Schnittstellen von: - Frontend Schnittstellen zu: - Automation Engine API - Datastore API
Mögliche Implementationen (und bekannte Anwendungsfälle): - Django - Flask - Spring	

6.2 Runtime View - Component Interaction

6.2.1 Automation Engine

Folgende Zugriffsarten werden für die unterschiedlichen Umsysteme verwendet.

Umsystem	Komponente	Protokoll	Bemerkungen
Infoblox	StackStorm Action (Python)	HTTPS	Über Python Infoblox-Client [18]
Check Point Management	Ansible	HTTPS	
Check Point Gateway	Ansible	SSH	
ISR 4461 (IOS-XE)	Napalm	SSH	Spezieller IOS Napalm Treiber
UCS CIMC	Netmiko	SSH	Python Script mit speziellem Error-Handling
Linux (KVM Host)	Ansible	SSH	
Linux (NFV Services)	Ansible	SSH	
MinIO S3 Object Store	Ansible	HTTPS	
VMware vCenter	Ansible	HTTPS	
ISO Packer	Ansible	SSH	
MinIO	Ansible	HTTPS	

6.2.2 Workflows

Die Orchestrierung und die Fehlerbehandlung sind die grossen Stärken des StackStorms. Dafür ist der Zugriff bzw. die Konfiguration der einzelnen Geräte die Stärke von Ansible.

Im Rahmen der Bachelorarbeit wird nur ein Hauptworkflow umgesetzt. Es ist der «Site Deployment» Workflow. Mit diesem kann ein neuer Standort provisioniert werden.

Weitere Workflows werden von diesem einen Hauptworkflow parametrisiert aufgerufen.

Der Vorteil von StackStorm ist, dass sehr einfach weitere Workflows hinzugefügt und in übergeordneten Workflows zusammengefasst werden können. So zum Beispiel «Add NFV Service», «Add Server Firewall-Zone» oder nach einem Hardware Austausch «Redeploy KVM Hypervisor».

Site Deployment Workflow

Der nachfolgende Business Process Model and Notation (BPMN) 2.0 Prozess (Abbildung 10) beschreibt den groben Ablauf der Automatisierung mit den wichtigsten Schritten.

Im Workflow wurden zusammengehörende Schritte in Sub-Workflows gruppiert. Dies wurde gemacht, damit sie einfacher wiederverwendbar sind und als eigenständige Workflows ausgeführt werden können.

Im Weiteren werden die Sub-Workflows detailliert abgebildet (Abbildung 11).

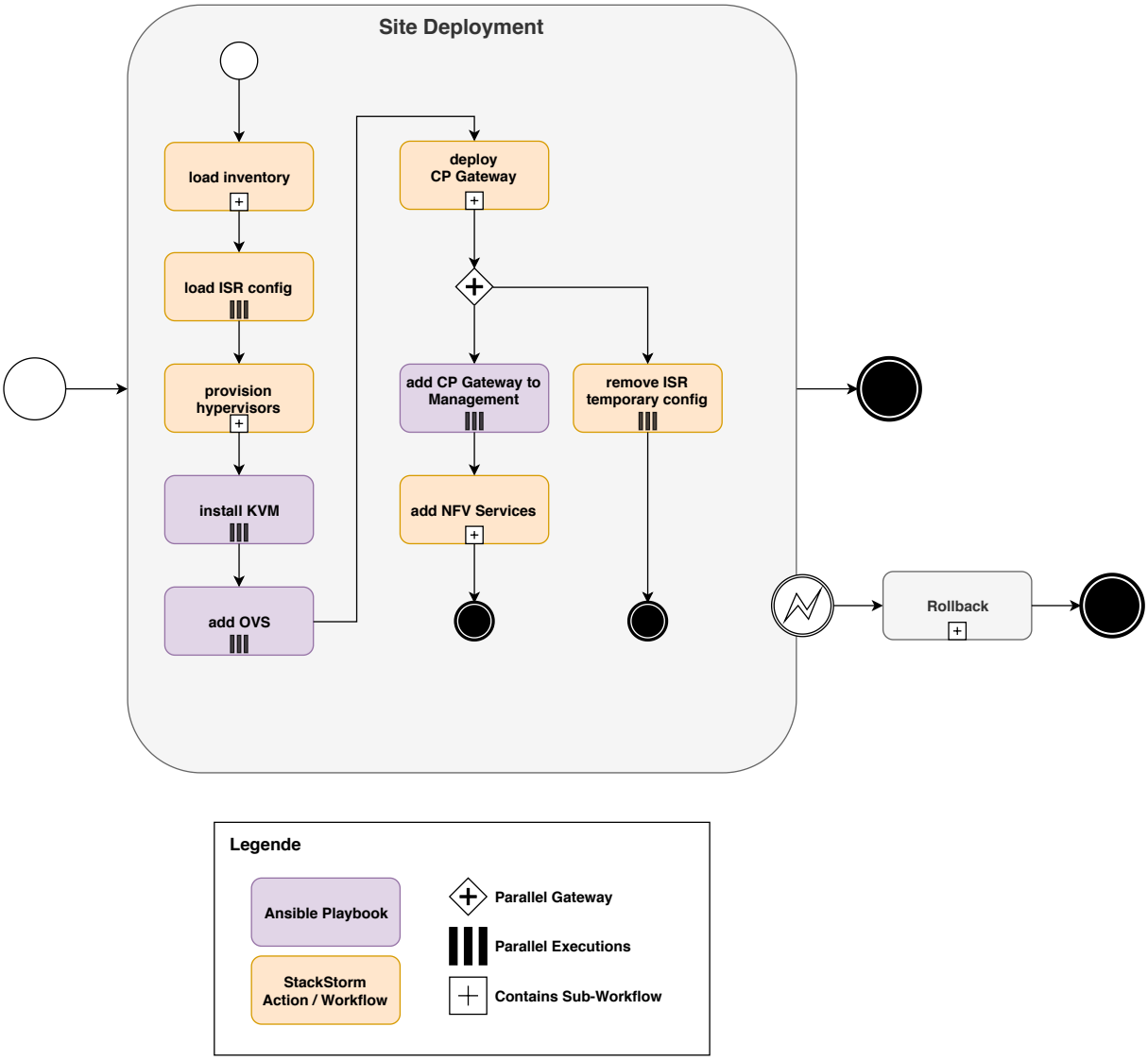


Abbildung 10: BPMN 2.0 – Effektiver Site Deployment Prozess

Die erweiterten Sub-Workflows sind hier im Detail ersichtlich:

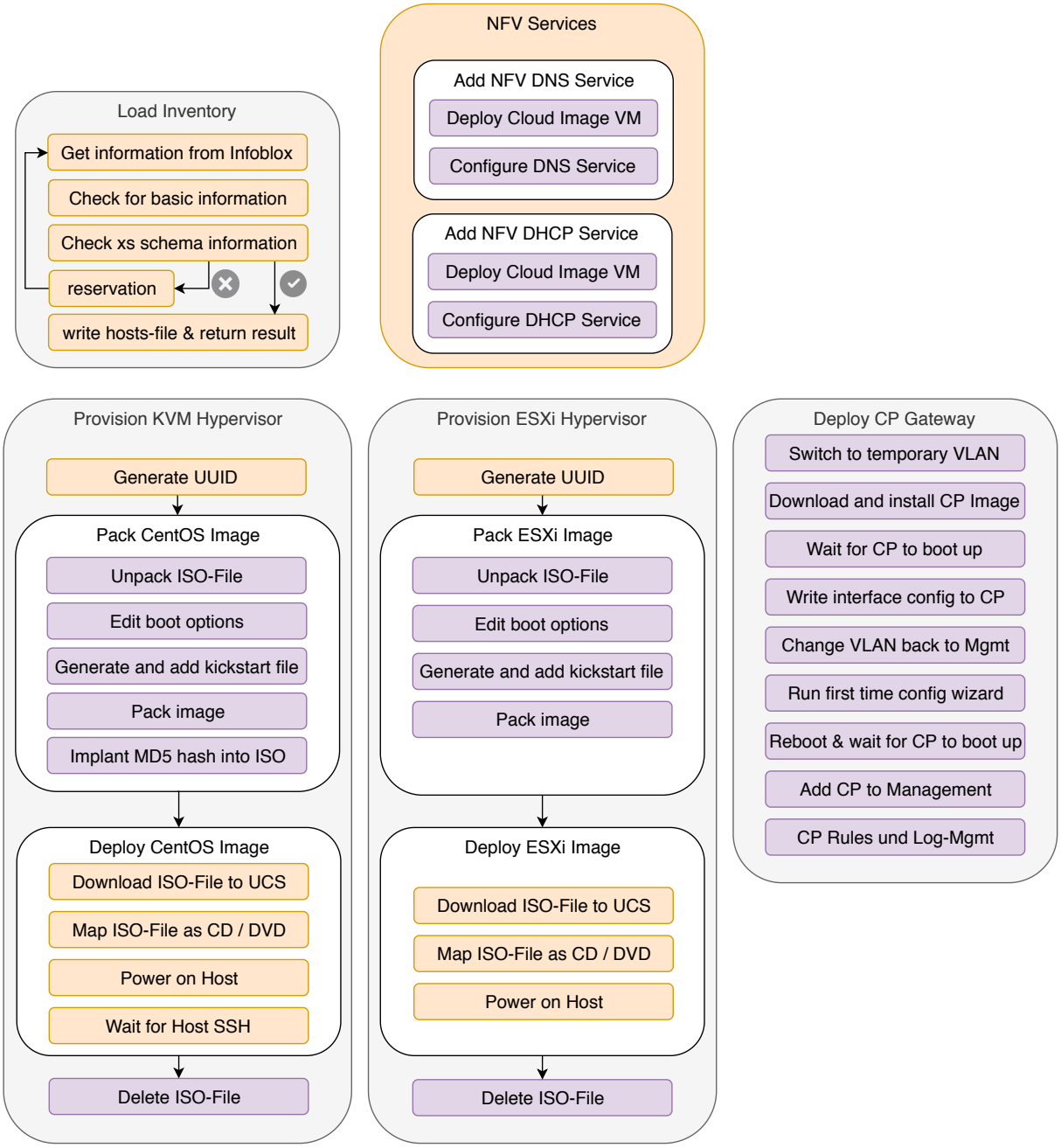


Abbildung 11: Aufgeklappte Sub-Workflows mit jeweils sequenzieller Abfolge

Input Parameter

site_name: Name des zu deployenden Standortes
esxi_iso_image: Optional kann ein anderes ESXi ISO Image mitgegeben werden
kvm_iso_image: Optional kann ein anderes Basis ISO Image für den KVM Host mitgegeben werden
nfv_qcow2_image: Optional kann ein anderes Cloud-Image für die NFV Services mitgegeben werden
server_vlan_zones: Anzahl zu deployende Firewall-Zonen bzw. Netzwerke für die virtuellen Maschinen auf dem VMware ESXi Hypervisor (standardmässig sind zwei definiert)

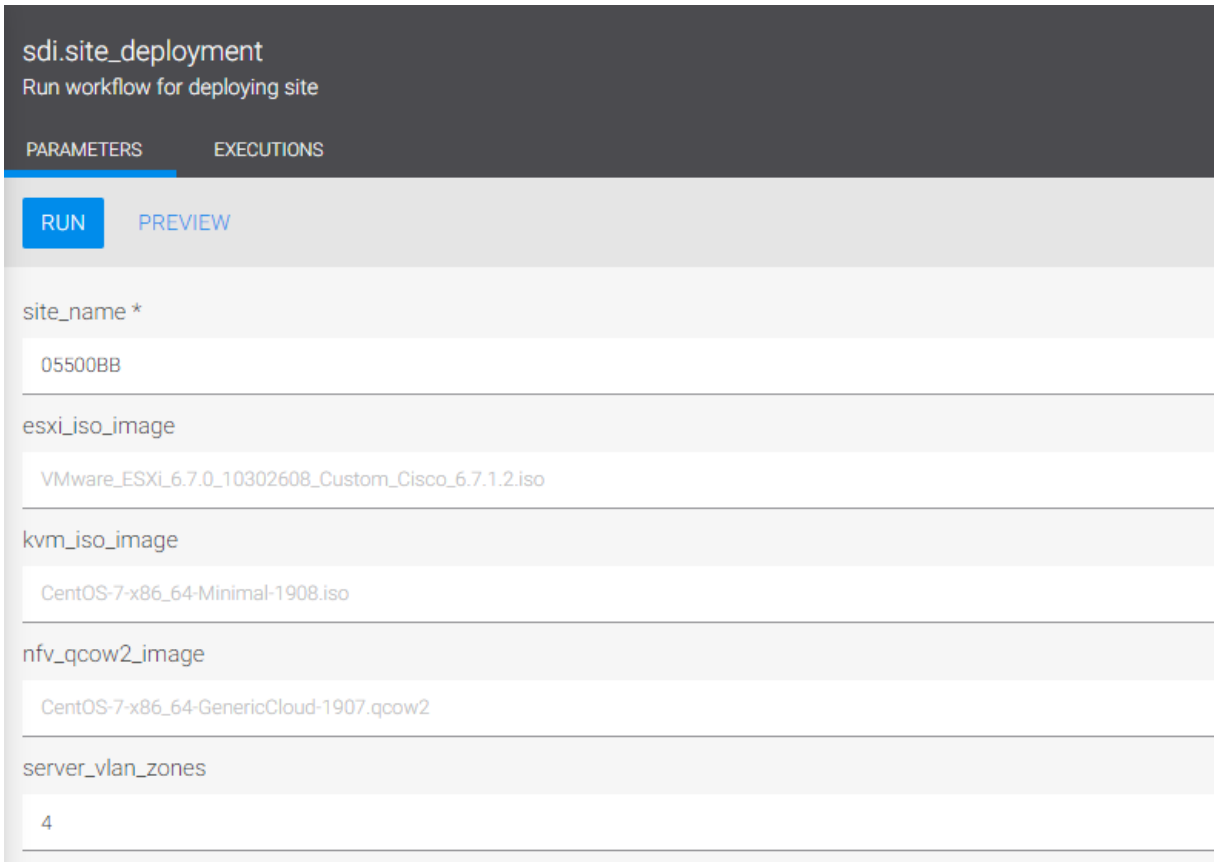


Abbildung 12: StackStorm Site Deployment Input Parameter

Der Task kann über den Button RUN (Abbildung 12) ausgeführt werden. Alternativ lässt sich derselbe Task folgendermassen über die API des StackStorms anstossen:

HTTP POST auf `https://stackstorm.sdi.local/api/v1/executions` [43] mit folgenden Headern gesetzt:

- Content-Type: application/json
- St2-Api-Key: <st2 API Key>

NB: Der API-Key ist via StackStorm Konsole [43] über folgenden Befehl generierbar und sollte gut verwahrt werden:

```
st2 apikey create -k -m '{"used_by": "StackStorm"'}
```

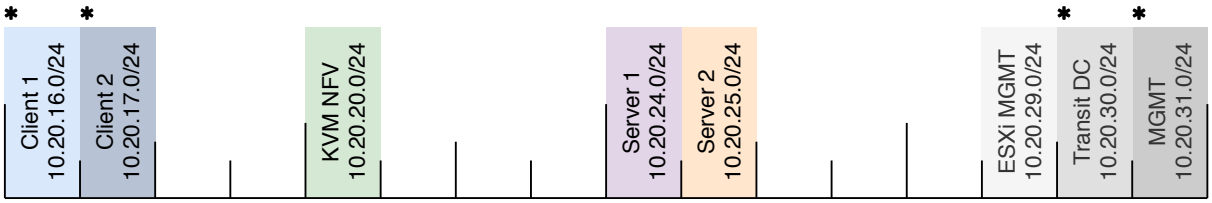
Und mit folgendem Body (dabei ist «sdi» das Pack und «site_deployment» steht für den Workflow oder die Action):

```
{
  "action": "sdi.site_deployment",
  "parameters": {
    "site_name": "05500bb",
    "server_vlan_zones": 4
  }
}
```

6.3 Konzepte

6.3.1 IP-Adresskonzept

Für einen Standort benötigen wir verschiedene Netze für die verschiedenen Services. Unser Konzept sieht es vor, verschiedene /24-Netzwerke aus einem /20-Subnetz zu verwenden:



10.20.16.0/20 (10.20.16.0 - 10.20.31.255)

Abbildung 13: IP-Adresskonzept

Wir empfehlen, mindestens ein /20-Netz zu verwenden. Auf dem Bild ist eine Minimalkonfiguration mit zwei Client- und zwei Servernetzen aufgezeigt. Ein /21-Range wäre somit mit den acht /24-Netzen bereits komplett belegt.

Jeder Standort soll dieselbe Struktur aufweisen. Jene Netze mit einem Stern (*) müssen im Vorfeld im Infoblox IPAM erfasst werden. Die weiteren werden automatisch durch das Deployment im Infoblox reserviert. Details folgen in der Umsetzung.

6.3.2 Namenskonzept

Die Geräte sind nach einem Schema benannt, welches durchgängig verwendet werden muss. Das heisst im Infoblox IPAM muss zwingend dieses Konzept angewendet werden, damit die Automatisierung richtig erkennen kann, welches Gerät für was zuständig ist. Falls das Namenskonzept angepasst werden soll, kann dies in der SDI Pack Konfiguration bewerkstelligt werden (siehe SDI Pack Configuration, Seite 53).

Der Name setzt sich zusammen aus fixen Teilen, Standortnamen, Gerätekürzel und Role-Identifier, damit es ins Namenskonzept der FUB passt:

Beispiel: 05500aa-1adc-01

- Der Standortnamen ist eine fortlaufende Nummer. Diese wird vom FUB bestimmt.
- Das Gerätekürzel ist immer zweistellig. Ein Mapping muss entsprechend erstellt werden (siehe Inventory, Seite 49).
- «01» bedeutet `Primary`, «02» bedeutet `Secondary` (im High Availability verwendet)

6.3.3 Netzwerkdesign

Cisco ISR 4461 Interface Konzept

Jeder UCS Server hat zwei interne Verbindungen zum Cisco ISR. Diese logischen Interfaces haben auf dem UCS und dem ISR jedoch unterschiedliche Bezeichnungen. Die Hypervisor Server haben zudem auch wieder unterschiedliche Bezeichnungen. Da es zwei unterschiedliche Typen von UCS Server sind, kann die Bezeichnung – zumindest, wenn ein Linux installiert ist – auch wieder anders sein.

ISR	UCS CIMC	UCS Hypervisor	Hypervisor
ucse2/0/0	GE0	enp9s0f0	KVM
ucse2/0/1	GE1	enp9s0f1	KVM
ucse3/0/0	GE0	vmnic2	ESXi
ucse3/0/1	GE1	vmnic3	ESXi

Anforderungen

An einem Standort kann es mehrere Client Netze haben. Diese müssen vollständig voneinander getrennt sein, da dies unterschiedliche Kunden sein können.

- Die Virtual Routing and Forwardings (VRFs) auf dem Customer Edge (CE) Router dürfen nicht miteinander kommunizieren können
- Es existiert ein dediziertes Netz für die NFV Services
- Das NFV Netz beinhaltet lokale Services wie DNS und DHCP
- Alle Netze sind isoliert. Eine Kommunikation zwischen den Netzen ist nur über die Firewall möglich
- Das Design soll auch für eine HA Redundanz mit einem zweiten Cisco ISR 4461 funktionieren

Entscheidung

Wir haben uns für das Design mit Bridge Domains Seite 34 entschieden, da es entscheidend weniger Netze zu provisionieren gibt und dadurch auch weniger IP-Adressen verschwendet werden. Es wird kein Routing Protokoll oder eine aufwändige Konfiguration von statischen Routen benötigt. Dadurch wird auf dem ISR ebenfalls kein First Hop Redundancy Protocol (FHRP) benötigt.

Der Nachteil, dass pro Firewall-Zone nur das direkt verbundene (directly connected) Netzwerk verwendet werden kann und keine weiteren Netze hinzugefügt werden können, ist im Moment kein Problem, da es keine Anforderung ist und das ganze Deployment um einiges komplizierter werden würde.

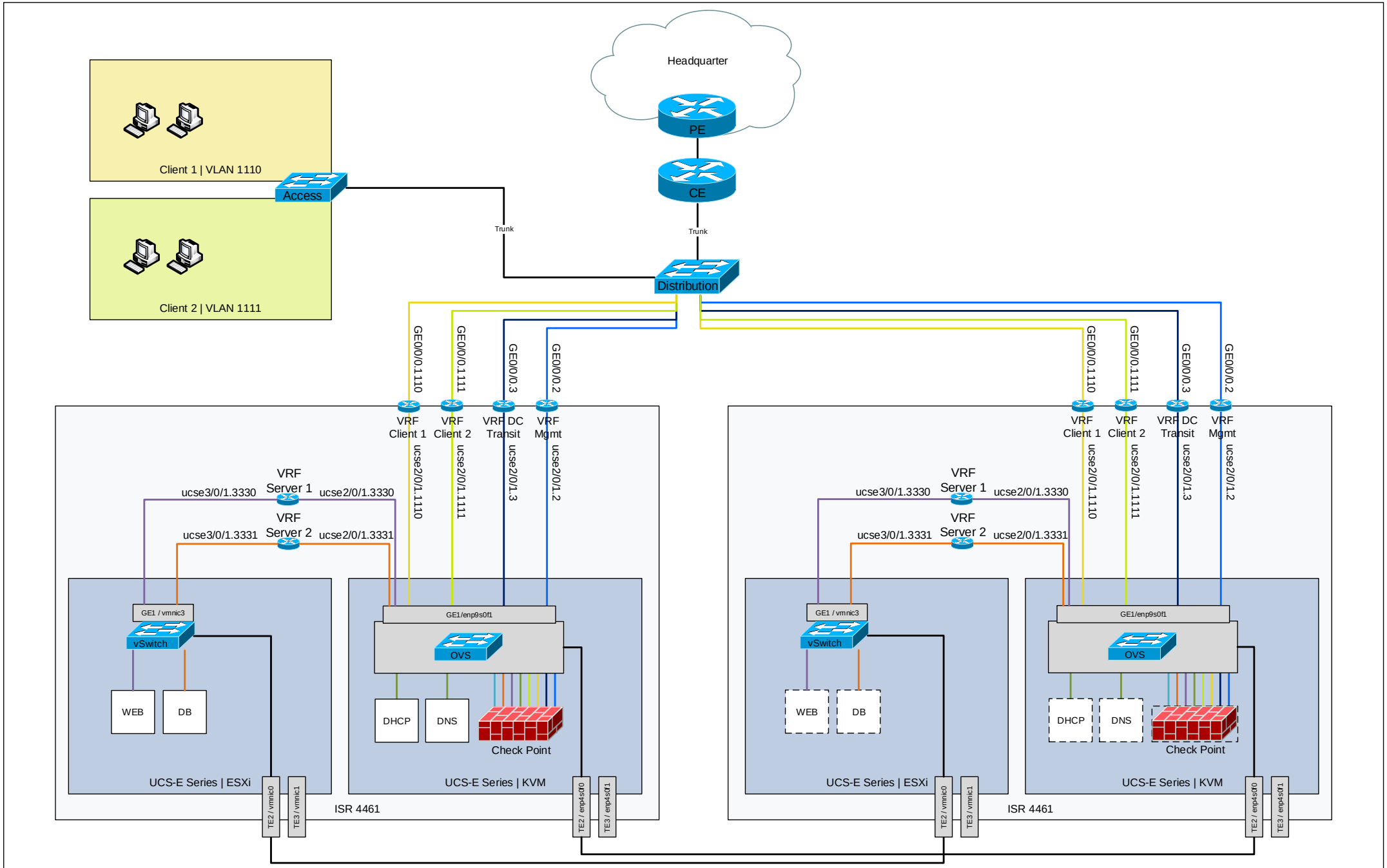
Routed Design mit VRFs

VRFs teilen den Cisco ISR 4461 in verschiedene Router-Instanzen, welche jeweils eine eigene Routingtabelle besitzen. Mittels VRF wird diese strikte Isolation umgesetzt. Jedoch müssen trotzdem beide Client Netze auf die gleichen Datacenter Ressourcen zugreifen können. Diese Zugriffe werden explizit über die Firewall gesteuert und zugelassen.

Alle VRFs auf dem CE Router müssen ebenfalls als VRFs auf dem ISR konfiguriert werden. Für jedes Servernetz muss ebenfalls ein eigenes VRF erstellt werden. Die Firewall hat somit in jede Firewall-Zone – die einem VRF entspricht – ein Transit-Netz.

Damit das Routing sauber funktioniert, muss entweder zwischen dem ISR und dem CE Router eine sehr aufwendige Konfiguration von statischen Routen gemacht werden oder ein dynamisches Routing Protokoll (wie z.B. «OSPF») eingesetzt werden.

Für ein redundantes Setup muss jeweils eine physische Verbindung zwischen den beiden selben UCS Servern gezogen werden. Diese Verbindungen dienen als Trunk. Zwischen den VRFs muss jeweils ein FHRP eingesetzt werden, damit der Default Gateway der virtuellen Maschinen und der Firewall automatisch umschaltet.



HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL

Routed Design - L1/L2

	Datum	Zeichner
Erstellung	01.10.2019	Yannick Zwicker, Pirmin Wenk
Bearb.	09.01.2020	Yannick Zwicker, Pirmin Wenk

BA SDI - Software-Defined Infrastructure

Design mit Bridge Domains

Bridge Domains repräsentieren eine Layer 2 Broadcast Domain [11]. Mittels Service-Instanzen kann die VLAN Encapsulation auf einem Interface hinzugefügt werden. Durch diese Konfiguration können die VLANs vom CE Router über den ISR in den Hypervisor bis in die VMs gezogen werden.

Die IP-Adresse des Cisco ISR wird nicht mehr direkt auf einem physischen Interface, sondern einem Bridge Domain Interface (BDI) konfiguriert. Ein BDI ist ein logisches Interface, welches sehr ähnlich zu einem Switched Virtual Interface (SVI) ist. Das GigabitEthernet0/0/0 vom ISR verhält sich somit wie ein Trunk der IEEE-Norm 802.1Q.

Der Vorteil dieser Bridge Domains ist, dass die VRFs und die Transitnetze komplett wegfallen. Der ISR macht kein Routing mehr. Geroutet wird nur noch auf dem CE Router und der Firewall. Obwohl kein Routing mehr gemacht wird, liegt der Vorteil dieser Hardware darin, dass weiterhin nur ein Gerät eingebaut werden muss, da der Router zwei UCS Server Blades beinhaltet. Da zwei unabhängige Server Blades vorhanden sind, können zwei Hypervisor von unterschiedlichen Herstellern verwendet werden. Der Router als Chassis bietet zudem den Vorteil, dass das ganze Networking direkt intern über reine Konfiguration gemacht werden kann und keine externen Kabel benötigt werden. Des Weiteren ist man bestens vorbereitet, falls es zusätzliche Anforderungen an den Router gibt.

VLANs

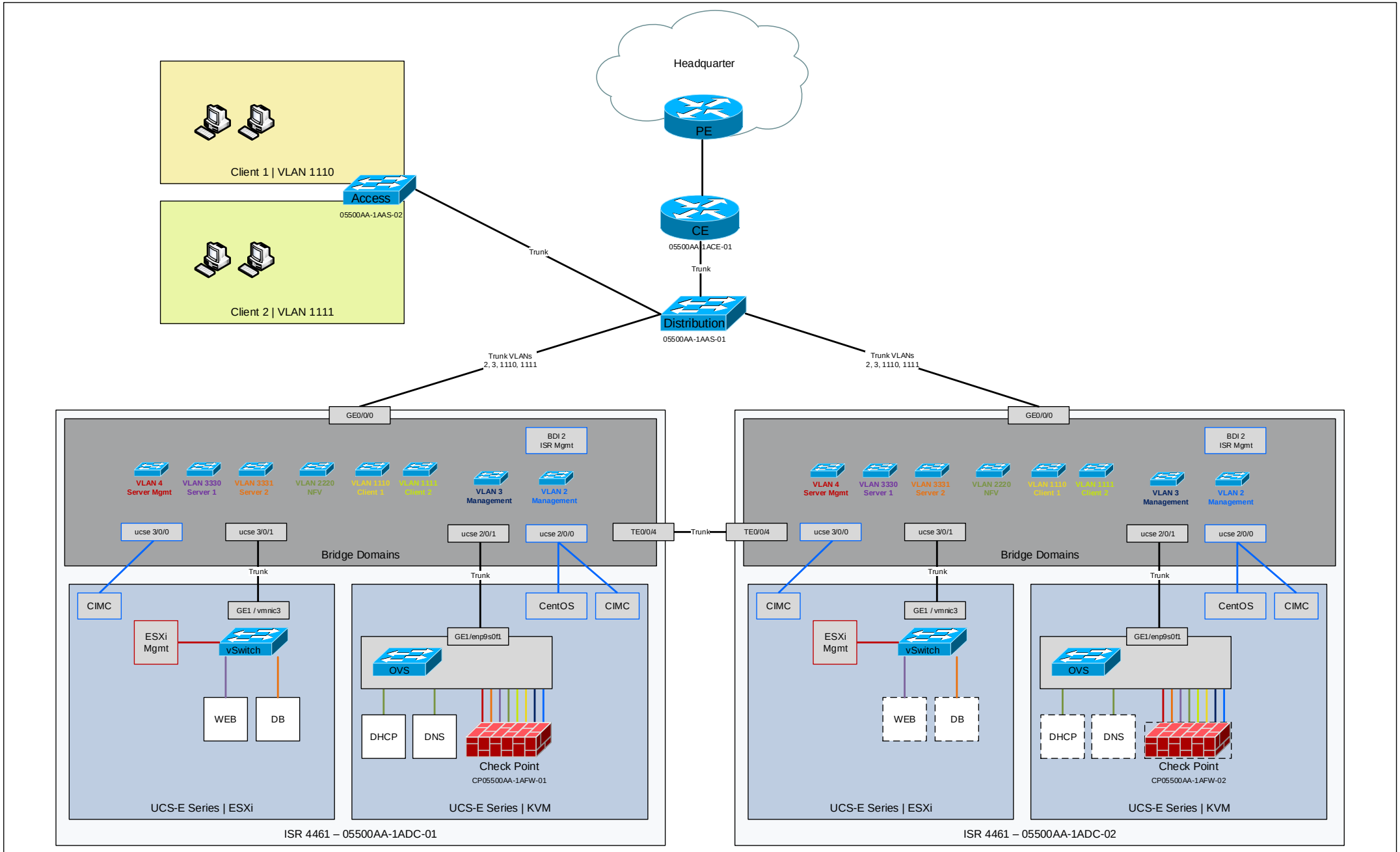
Von der FUB sind folgende VLANs vorgegeben:

1XXX Client Netze
2 Netzwerk Management

Es sind aber noch weitere VLANs notwendig. Das NFV VLAN bekommt immer die VLAN ID 2220. Die Servernetze beginnen bei 3330 und werden inkrementiert bis zur notwendigen Anzahl. Zudem gibt es ein eigenes Netz, um die ESXi Server zu verwalten und eines, für die Serververbindung ins Headquarter.

3 Transfernetz für Verbindungen ins Headquarter Datacenter
4* ESXi Server Management
2220* NFV Services
3330* Servernetz 1
3331* Servernetz 2
333X* Servernetz X

Einige dieser VLANs verlassen nie den Cisco ISR und terminieren direkt auf der Firewall. Dazu gehören die Servernetze, NFV Netz und Server Management, jene mit Stern (*). Andere Netze wie Netzwerk-Management, Transit Headquarter und die Client Netze sind bis auf den CE Router gezogen. Weitere Details sind in den Abbildungen Netzwerkdesign – Switched L1/L2 Seite 35 und Netzwerkdesign – Switched L3 Seite 36 ersichtlich.



HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL

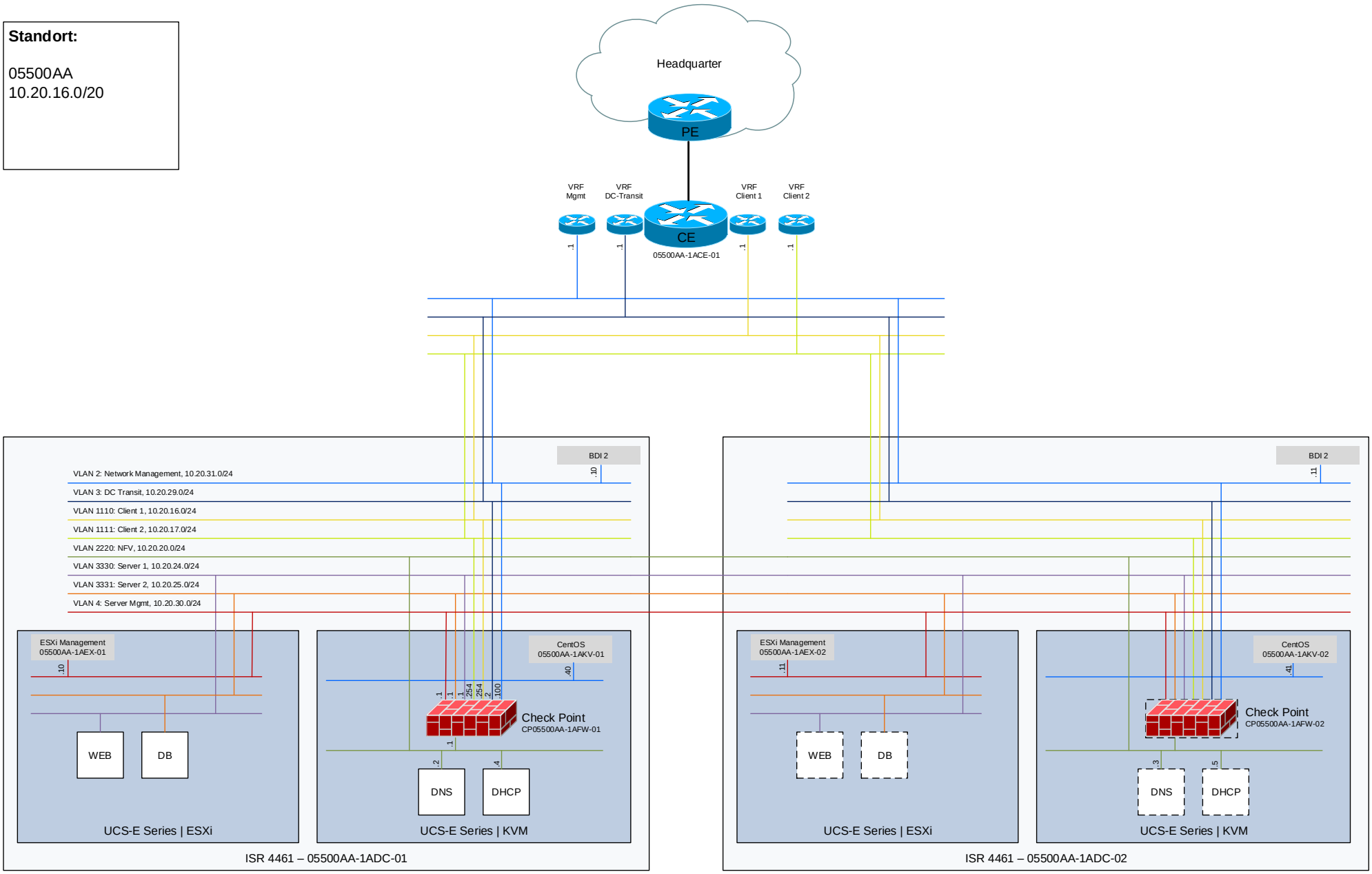
Switched Design - L1/L2

	Datum	Zeichner
Erstellung	01.10.2019	Yannick Zwicker, Pirmin Wenk
Bearb.	09.01.2020	Yannick Zwicker, Pirmin Wenk

BA SDI - Software-Defined Infrastructure

Standort:

05500AA
10.20.16.0/20



HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL	Switched Design - L3		Datum / Zeichner
		Erstellung	01.10.2019 / Yannick Zwicker, Pirmin Wenk
		Bearb.	09.01.2020 / Yannick Zwicker, Pirmin Wenk
	BA SDI - Software-Defined Infrastructure		

6.4 Technologieentscheidungen

6.4.1 Bare-Metal Betriebssysteminstallation

Zu Beginn der Analyse wurden mehrere verschiedene Möglichkeiten in Betracht gezogen, das Betriebssystem auf die «Bare-Metal» UCS Server automatisiert zu deployen. Dazu gehört ein CentOS auf dem einen UCS-Einschub und ein VMware VMware ESXi-Server für den anderen Einschub.

Obwohl das CIMC eine XML API anbieten würde, gelang es uns nicht diese zu verwenden, da die Dokumentation sehr schlecht ist, keine Beispiele vorhanden sind und soweit wir testen konnten die Funktionalität zur Installation eines Betriebssystem nicht über die API angeboten wird.

Wir haben Preboot Execution Environment (PXE) oder eine Installation über die CLI des CIMC genauer ins Auge gefasst. PXE verwendet leider das unsichere TFTP-Protokoll. Deshalb entscheiden wir uns schlussendlich für die zweite Variante. Dort wird ein ISO-Packer benötigt, um die Konfiguration und Boot-Optionen direkt ins ISO hineinzupacken. Diese beiden Möglichkeiten sind in der Umsetzung beschrieben (Siehe Bare Metal Deployment, Seite 47).

6.4.2 SSH Zugriffe

CIMC

Um das Bare-Metal Deployment über das UCS CLI zu automatisieren, muss via SSH auf das CIMC zugegriffen werden, um danach die Befehle in der richtigen Reihenfolge und zum richtigen Zeitpunkt abzusetzen. Eine Automatisierung mit Ansible ist insofern ungeeignet, weil auf dem SSH-Zielsystem Python installiert sein müsste. Die Befehle über StackStorm direkt abzusetzen ist auch nicht sinnvoll, weil über die vorhandene Action beim Zielsystem eine Linux-Shell erwartet wird.

Wir haben uns deshalb dazu entschieden, über die Python-Library «Netmiko» [4] ein eigenes Script zu implementieren, welches zugleich auch den gesamten Prozess überwacht und mögliche Fehler berücksichtigt. Über die live Ausgabe im StackStorm sind die Logs des Scripts immer sofort ersichtlich.

Check Point

Unser eigenes Check Point Image enthält eine Grundkonfiguration mit einer temporären IP-Adresse (siehe Gateway, Seite 57). Um die richtige Konfiguration (IP-Adressen, Default Gateway, Interfaces etc.) zu setzen, müssen per SSH einige Befehle abgesetzt werden.

Unsere Tests haben gezeigt, dass es am sinnvollsten ist, Ansible mit dem Native-Command «raw» [36] zu verwenden und dabei den Python-Interpreter manuell auf `/opt/CPsuite-R80.30/fw1/Python/bin/python` zu setzen, da dort der Python-Interpreter abgelegt ist.

6.4.3 Docker

Docker [15] bietet uns verschiedene Vorteile in der Architektur unserer Orchestrierung. Die Services können dadurch leichtgewichtig in sogenannten Containern verwaltet werden. Der Updateprozess ist ebenfalls sehr elegant gelöst, wobei nur der Container ausgetauscht werden muss. Die Daten bleiben bestehen.

6.4.4 NFV

Für die virtuellen Network Functions, wie DHCP, DNS und Firewall, kommt entweder eine auf Docker Containern oder auf virtuellen Maschinen basierende Lösung in Frage.

Die **Docker Container** bieten den Vorteil, dass die Images sehr klein sein können und auf das Minimum reduziert sind. Dazu sind sie einfach wartbar. Jedoch wird damit eine zusätzliche Technologie eingeführt, da die Check Point Firewall nur als virtuelle Maschine (VM) zur Verfügung steht und somit zwei unterschiedliche Virtualisierungslösungen eingesetzt werden würden. Zudem gestaltet sich die Hochverfügbarkeit über ein Cluster schwieriger, da für Docker Container ein Cluster wie Docker Swarm oder Kubernetes eingesetzt werden müsste.

Virtuelle Maschinen hingegen bieten den Vorteil, dass sie einen eigenen Kernel besitzen und dieser nicht wie bei Docker mit dem Host geteilt wird. Dies erhöht die Sicherheit. Da die Check Point Firewall nur als virtuelle Maschine verfügbar ist, muss nur eine Technologie eingesetzt werden. Ein Nachteil zeigt sich in der Wartbarkeit bei virtuellen Maschinen. Diese stehen oft nur als klassische Images zur Verfügung und müssen installiert und konfiguriert werden.

Aufgrund dessen verwenden wir virtuelle Maschinen mit Cloud Images [33]. Dies sind bereits vorinstallierte Images, die jedoch über Cloud-Init [25] konfiguriert werden können, so wie es auch die grossen Cloud Provider z.B. AWS [1] machen.

6.4.5 IPAM

Als IPAM ist «Infoblox» vorgegeben, weil FUB dieses bereits im Einsatz hat.

Infoblox bietet eine HTTP-REST API [17], die über eine Automatisierungslösung sehr elegant angestossen werden kann. Auch gibt es vorgefertigte Libraries für Python [18] und Ansible [35], die verwendet werden können.

6.5 Architekturentscheidungen

6.5.1 Automation Engine

Als Automatisierungslösung haben wir uns entschieden, StackStorm zu verwenden, weil es uns sehr viel Flexibilität bereitet und wir durch die sogenannten Packs sehr viel Funktionalität hinzufügen können. Eine detaillierte Analyse befindet sich im Kapitel Software Evaluation Seite 41.

6.5.2 High Availability

Grundsätzlich gibt es zwei Ansätze die Services hoch verfügbar zu betreiben. Einerseits kann man einen HA-Cluster auf der Ebene der Hypervisor erstellen, andererseits kann man auch die einzelnen Services – wie z.B. die Check Point Firewall in einem ClusterXL [27] – in einem HA-Cluster Betreiben.

Hypervisor Cluster

Der Vorteil, wenn ein hoch verfügbarer Cluster auf der Hypervisor Ebene erstellt wird, muss die Applikation dies selbst nicht unterstützen.

Der Nachteil ist, dass sowohl bei VMware, als auch bei einer KVM Lösung, dedizierter shared Storage – z.B. Network File System (NFS), virtual Storage Attached Network (vSAN) von VMware o.ä. – benötigt wird, um einen Cluster bilden zu können [52].

Des Weiteren ist es absolut empfehlenswert, einen Cluster aus jeweils mindestens drei Nodes zu betreiben, damit die Datenintegrität sichergestellt ist [51]. «Split-Brain» Situationen können dadurch verhindert werden.

Service Cluster

Wenn der Service, wie z.B. die Check Point Firewall, DNS oder DHCP Server ein Clustering oder active/standby in der Applikation anbieten, muss sich der Hypervisor nicht darum kümmern. Es wird zudem kein shared Storge benötigt.

Jedoch müssen alle NFV Services dies anbieten, wenn der Service hoch verfügbar sein soll.

Fazit

Da es angedacht ist maximal zwei Cisco ISR 4461 an einem Standort zu betreiben, bevorzugen wir die Variante Service Cluster, weil nicht drei Nodes zur Verfügung stehen und sowohl die Check Point Firewall ein HA-Deployment unterstützt, als auch DNS und DHCP Server als active/standby betrieben werden können.

Entscheidung

Wir haben uns zusammen mit Check Point, dem Auftraggeber und der Betreuung für die letzte Variante mit Central Managed Logs entschieden. Es wäre nicht praktikabel, an jedem Aussenstandort ein Management zu verwalten.

Der Nachteil, der in Kauf genommen werden muss, ist bei einem Unterbruch zwischen Gateway und Management. Denn dann können keine Änderungen mehr an der Firewall vorgenommen werden. Diese funktioniert aber autonom wie gewohnt weiter. Sobald der Standort wieder online ist, können auch wieder Änderungen über das Management am Gateway vorgenommen werden.

6.5.3 Check Point Firewall-Konzept

Wir haben verschiedene Varianten ausgearbeitet, wie das Firewall-Setup aussehen könnte (Details siehe Check Point – Desings, Seite 40).

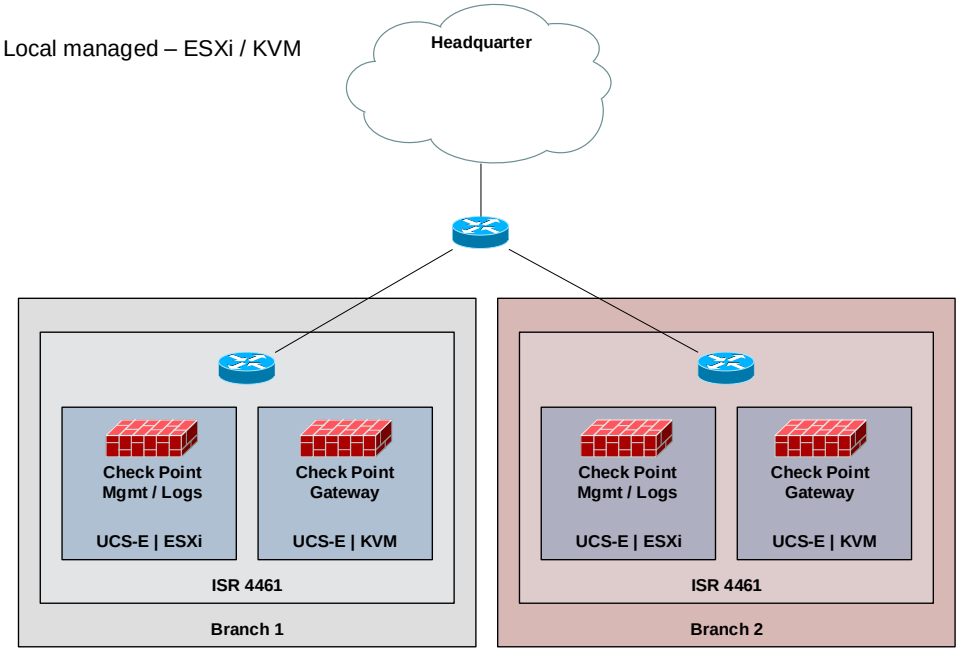
Local managed – ESXi Das Management wird an jedem Standort auf dem VMware ESXi Hypervisor deployt, das Gateway auf dem KVM Host

Local managed – KVM Das Management und das Gateway werden an jedem Standort auf dem KVM Hypervisor deployt

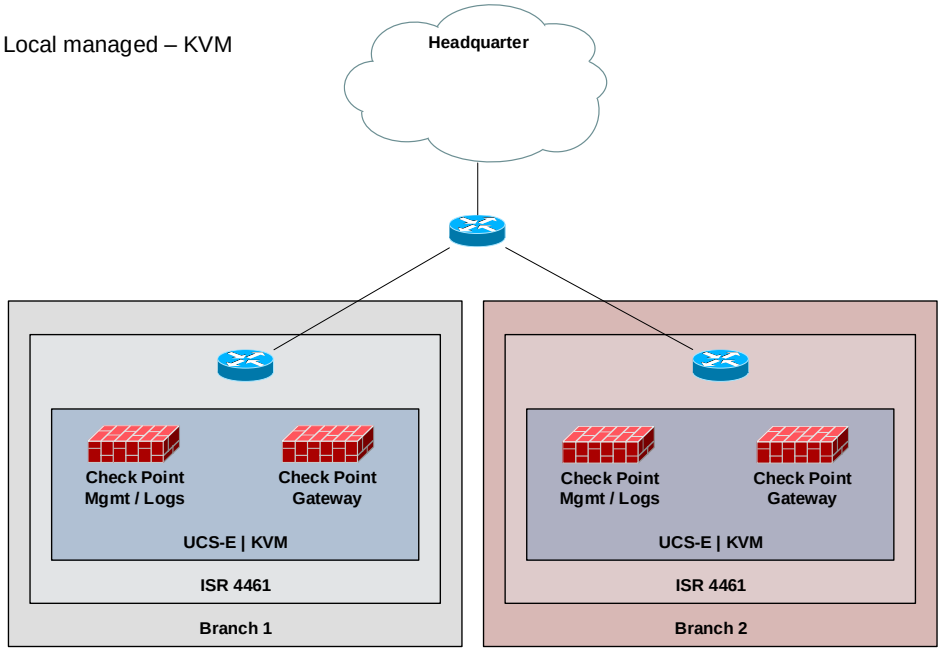
Central managed Das Management wird zentral betrieben. An den Aussenstandorten stehen nur jeweils das Gateway und die Logs

Central managed mit Logs Die Logs werden ebenfalls zentral verwaltet. Nur die Gateways sind verteilt

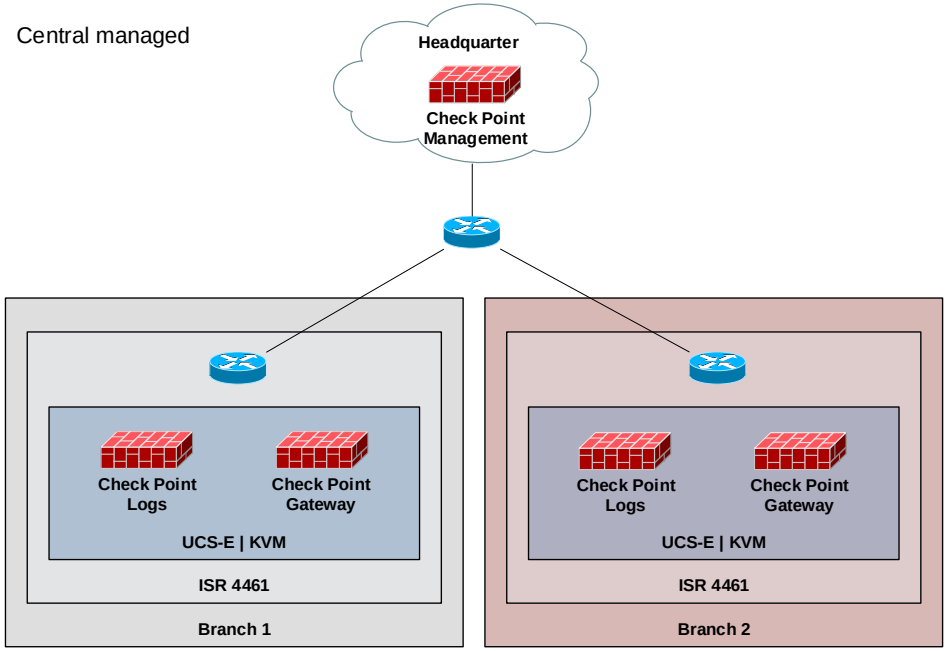
Local managed – ESXi / KVM



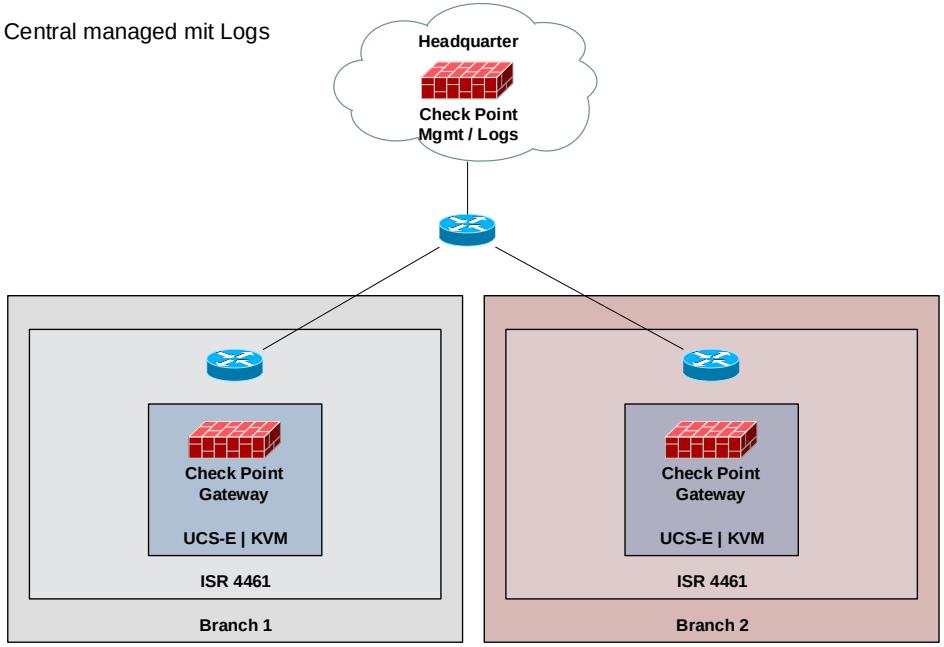
Local managed – KVM



Central managed



Central managed mit Logs



HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL

Check Point Mgmt Designs

	Datum	Zeichner
Erstellung	01.10.2019	Yannick Zwicker, Pirmin Wenk
Bearb.	09.01.2020	Yannick Zwicker, Pirmin Wenk

BA SDI - Software-Defined Infrastructure

7 Software Evaluation

Um das richtige Framework und die richtigen Libraries für die Automatisierung zu finden gibt es ein paar zentrale Fragen zu beantworten.

- Welche Umsysteme sind involviert und müssen unterstützt sein?
- Was für Schnittstellen bieten diese Systeme an?
- Wer sind die User dieses Systems?
- Unterstützen die Umsysteme Agents oder muss es eine Lösung ohne Agents sein (Agent vs. Agentless)? Ist ein Push oder Pull basierter Ansatz besser geeignet?
- Welche APIs bieten die Frameworks und Libraries an?
- Gibt es allenfalls einen Enterprise Support für den Industriepartner?
- Können die Use Cases (Seite 14) umgesetzt werden? Wie einfach kann das System im Nachhinein noch erweitert werden?

In die engere Auswahl haben es die folgenden Produkte geschafft. Um diese miteinander zu vergleichen, mussten unter anderem aus der jeweiligen Dokumentation genaue Informationen und Vergleiche gesucht werden, die uns die nötigen Anhaltspunkte geben [40].

Ansible ist ein IT Automation Tool. Es kann Software deployen, Konfigurationen ändern, aber auch Tasks ausführen, wie einen Server neustarten. Es ist rein CLI basiert. [37]

StackStorm ist eine Event-Driven Plattform zur Integration und Automatisierung von bestehenden Services und Plattformen. Es bietet eine Workflow und Rule Engine zur Orchestrierung auf Basis einer If-this-then-that (IFTTT) Logik. [30]

AWX ist das Upstream Open Source Projekt von Ansible Tower [38]. Es bietet eine Task Engine, ein GUI und eine REST API für Ansible. [12]

Salt ist ein «distributed remote execution system» von der Firma SaltStack, welches stark auf einer Hub and Spoke Architektur basiert. Es kann als Configuration Management Tool genutzt werden, aber auch Event-Driven Tasks ausführen. [16]

Wir haben uns für eine Kombination aus **StackStorm** und **Ansible** entschieden.

StackStorm bietet eine sehr gute und einfach erweiterbare Workflow Engine an, um die komplexe Businesslogik zu implementieren. Zudem können weitere Workflows hinzugefügt, aber wiederum auch in anderen Workflows wiederverwendet werden. Die Workflows können in sogenannten Packs – die einzelnen Git Repositories entsprechen – abgelegt werden, was zusätzliche Vorteile mit sich bringt. Dadurch gibt es eine Versionsverwaltung und über Pull-Request können alle Änderungen von weiteren Stellen überprüft werden.

Ansible dient vor allem als Schnittstelle zu den Umsystemen. Darüber werden die effektiven Konfigurationen durchgeführt. Das Ökosystem an bereits vorhandenen Modulen für den Zugriff oder die Konfiguration von diversen Systemen ist sehr gross.

Da dies aber aus unseren Augen nicht die einzig «richtige» Entscheidung ist und ebenfalls andere Tools für diese Problemstellung in Frage kämen, haben wir die für uns wichtigsten Kriterien für die Entscheidungsbasis festgehalten.

7.1 Ansible

Vorteile:

- ✓ Grosse Community und viele Hersteller, die Module für ihre Software oder Hardware anbieten
- ✓ Viele Module bereits vorhanden
- ✓ Agentless Architektur via SSH und Filetransfer über SCP/SFTP
- ✓ YAML als Sprache
- ✓ Der Soll Zustand kann spezifiziert werden. Falls ein System diesem nicht entspricht, wird es in jenen gebracht. Dadurch können Konfigurationsänderungen einfach erkannt werden

Nachteile:

- ✗ Bietet keine API an bzw. ist nur ein CLI Tool
- ✗ Die Orchestrierung und Abbildung von komplexer Businesslogik ist schwierig zu bewerkstelligen

7.2 StackStorm

Vorteile:

- ✓ Sehr gute Integration von Ansible
- ✓ YAML als Sprache
- ✓ Weitere Packs können aus einer Vielzahl von verfügbaren Modulen hinzuinstalliert werden [44]
- ✓ Starke Workflow und Rule Engine zur Orchestrierung auf Basis einer IFTTT Logik. [30]

Nachteile:

- ✗ Keine leichtgewichtige Umgebung, da in-Memory-Datenbanken und Queues für die Orchestrierung verwendet werden
- ✗ Noch keine Möglichkeit, einen Workflow ab einem bestimmten Task zu starten (diese Option soll in den nächsten Versionen unterstützt sein)

7.3 AWX

Vorteile:

- ✓ Native Unterstützung von Ansible und Ansible-Playbooks
- ✓ Webbasiertes User-Interface mit REST API
- ✓ Einfache Installation über ein Ansible-Playbook

Nachteile:

- ✗ Schwierige Abbildung komplexer Logik
- ✗ Alles muss mit Ansible kompatibel sein. Wenn es noch kein Modul gibt, muss dieses selbst entwickelt werden
- ✗ Bindung an die eine Technologie

7.4 Salt

Vorteile:

- ✓ Ähnlich wie Ansible definiert man den gewünschten State
- ✓ YAML als Sprache
- ✓ Bis zu sehr komplexer Architektur erweiterbar

Nachteile:

- ✗ Beim Einstieg extrem spezielle und komplexe Nomenklatur (schwer zu verstehen) [40]
- ✗ Agent-basierter Zugriff, spezielle Treiber für Netzwerkgeräte
- ✗ Keine Kompatibilität für low-level Befehle gefunden

7.5 Lizenzierung

Dieses Kapitel soll auf einen Blick aufzeigen, welche Lizenzierungsmodelle die wichtigsten Softwarekomponenten aufweisen, damit ersichtlich wird, was bei der Anwendung der Produkte zu beachten ist.

Software	**Lizenzmodell**
Ansible	GNU General Public License v3 [8]
StackStorm	Apache 2.0 [7]
Ansible AWX	Apache 2.0 [7]
SaltStack	Apache 2.0 [7]
VMware ESXi	Kostenpflichtige VSphere-Lizenz [49] (Lizenz kann via vCenter hinterlegt werden)
VMware vCenter	Kostenpflichtige vCenter Server Lizenz [50]
Check Point Firewall Mgmt	Kostenpflichtige Lizenz von Check Point
Check Point Firewall Gateway	Kostenpflichtige Lizenz von Check Point

Unter Apache 2.0 [13] ist es möglich, die Software für kommerzielle Zwecke zu verwenden und anzupassen, sofern diese weiterhin unter derselben Lizenz verwendet wird.

8 Umsetzung

8.1 Infrastruktur Aufbau

Folgende Hardware und Lizenzen stehen uns im Rahmen des Projektes zur Verfügung:

- 2x Cisco ISR 4461 (IOS XE 16.09.02) Router mit je:
  - UCS-E160S Blade (CIMC BMC 3.2)
  - UCS-E1120D Blade (CIMC BMC 3.2)
- Cisco Catalyst 2811 Router (IOS 15.1(4)M12a) -> CE Router
- 2x Cisco Catalyst 3750-X Switch (IOS 12.2(53)SE2) -> Distribution / Access
- Check Point Management Lizenzen (All-in-One Security Bundle Eval - 6 Monate)

8.2 Vorbedingungen

Einige Dinge lassen sich nicht vollständig automatisieren oder sind eine Vorbedingung, damit die Automatisierung reibungslos funktionieren kann.

8.2.1 Infoblox

Aus dem Infoblox IPAM muss ein IP-Bereich reserviert werden. Dieser Bereich muss mit dem «Extensible Attribute» `sdi_site` versehen werden (z.B. «05500bb»).

Name	Comment	Type
10.20.16.0/20		IPv4 Network Container
10.20.16.0/24	client	IPv4 Network
10.20.17.0/24	client	IPv4 Network
10.20.30.0/24	dc_transit	IPv4 Network
10.20.31.0/24	network_mgmt	IPv4 Network
05500aa-1ace-01-gi0-0.1110		Host
05500aa-1ace-01-gi0-0.1111		Host
05500aa-1ace-01-gi0-0.2		Host
05500aa-1ace-01-gi0-0.3		Host
05500aa-1adc-01-bdi2		Host

Abbildung 18: Manuelle Konfiguration im Infoblox

Eine Anleitung, wie diese Einträge genau gemacht werden müssen, findet sich im Anhang (siehe Infoblox vorbereiten, Seite 86).

8.2.2 Netzwerk Connectivity

Zum Aussenstandort braucht es eine entsprechende Netzwerkverbindung. Diese muss inkl. Routing des IP-Range von oben vorhanden sein. Falls eine Firewall dazwischensteht, ist es wichtig, dass folgende Services zugelassen sind:

- DNS – falls das Deployment Domainnames auflösen muss
- SSH – im gesamten Subnetz des Branchoffice
- HTTPS – für den Zugriff des UCS auf den Datastore
- Check Point Firewall spezifische Ports zwischen Gateway und Management [24]

8.2.3 CE Router Konfiguration

Das Netzwerk vor Ort muss mit den entsprechenden VLANs durchkonfiguriert sein (siehe VLANs, Seite 34) und das Routing muss entsprechend eingerichtet sein.

Beispiel Base CE Config

Die Konfiguration des CE Routers könnte dann in etwa so gesetzt sein:

```
vrf definition dc_transit
 address-family ipv4
 exit-address-family
vrf definition client1
 address-family ipv4
 exit-address-family
vrf definition client2
 address-family ipv4
 exit-address-family

interface GigabitEthernet0/0.2
 description isr4461 gi0/0/0
 encapsulation dot1Q 2
 ip address 10.20.47.1 255.255.255.0
interface GigabitEthernet0/0.3
 encapsulation dot1Q 3
 vrf forwarding dc_transit
 ip address 10.20.46.1 255.255.255.0
interface GigabitEthernet0/0.1110
 encapsulation dot1Q 1110
 vrf forwarding client1
 ip address 10.20.32.1 255.255.255.0
interface GigabitEthernet0/0.1111
 encapsulation dot1Q 1111
 vrf forwarding client2
 ip address 10.20.33.1 255.255.255.0
```

Manuelle CE Config

Nach dem Deployment benötigt der CE Router jeweils noch die folgende Konfiguration:

- die statische Route des gesamten Netzes zur Firewall (Netzwerk Management)
- die statischen Routen der Client-Netze zur Firewall (jeweiliges Clientnetz)
- die statische Route des DC Transit-Netzes zur Firewall (DC Transit)
- die IP Helper Adresse pro Client Interface auf den NFV DHCP-Server

Eine Beispielkonfiguration sieht so aus:

```
ip route 10.20.32.0 255.255.240.0 10.20.47.3
ip route vrf client1 10.20.32.0 255.255.240.0 10.20.32.2
ip route vrf client2 10.20.32.0 255.255.240.0 10.20.33.2
ip route vrf dc_transit 10.20.32.0 255.255.240.0 10.20.46.2
interface GigabitEthernet0/0.1110
 ip helper-address 10.20.35.2
interface GigabitEthernet0/0.1111
 ip helper-address 10.20.35.2
```

8.2.4 IP-Connectivity

Über ein anderes Tool wird die Grundkonfiguration auf den Cisco ISR 4461 geschrieben. In dieser Grundkonfiguration muss ein BDI mit IP-Adresse aus dem IP-Range `network_mgmt` vergeben sein und die default Route auf den CE Router muss gesetzt sein.

Listing 1: Beispiel Base Config Cisco ISR 4461

```
hostname 05500bb-1adc-01
no ip domain lookup
ip domain name 05500bb.sdi.local

username ins secret 8 <encrypted secret>
enable secret 8 <encrypted secret>
crypto key generate rsa modulus 4096
line vty 0 4
 login local
 transport input ssh

interface GigabitEthernet0/0/0
 no ip address
 negotiation auto
 no shutdown
 service instance 2 ethernet
  encapsulation dot1q 2
  rewrite ingress tag pop 1 symmetric
  bridge-domain 2

interface BDI2
 ip address 10.20.47.2 255.255.255.0
 no shutdown

ip route 0.0.0.0 0.0.0.0 10.20.47.1
```

8.2.5 Key-management

Grundsätzlich wird im Deployment für folgende Dienste ein Public-Key hinterlegt, damit mit dem zugehörigen Private-Key auf den Dienst per SSH zugegriffen werden kann:

- CentOS KVM Hypervisor
- NFV DHCP Server
- NFV DNS Server

Das Key-Pair kann folgendermassen erstellt werden:

```
$ ssh-keygen -t ecdsa -b 521
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/ins/.ssh/id_ecdsa): /home/ins/.ssh/sdi
Your identification has been saved in /home/ins/.ssh/sdi.
Your public key has been saved in /home/ins/.ssh/sdi.pub.
```

Der Public-Key muss nun in den StackStorm Key-Value-Store (mit dem Namen `ssh_public_key`) abgelegt werden (siehe Secrets, Seite 54), damit dieser während dem Deployment auf die Hosts verteilt wird.

8.2.6 UCS Vorbereitungen

Erster Zugriff und einrichten

Eine Anleitung, wie auf das CIMC des UCS zugegriffen werden kann und die nötigen Einstellungen definiert werden, findet sich im Anhang (siehe ISR 4461 einrichten, Seite 83).

Festplatte initialisieren

Die Festplatte muss komplett geleert sein, damit der Server vom ISO-Image startet (siehe Festplatte initialisieren, Seite 84).

Boot Reihenfolge

Die Boot-Reihenfolge auf den beiden UCS-Blades muss bereits im Vorfeld korrekt sein, da diese nicht über das CLI geändert werden kann (siehe CIMC, Seite 64) und dadurch nicht automatisiert werden kann. Theoretisch müsste es möglich sein, aber in unseren Tests hat es nicht funktioniert.

Default Benutzername und Passwort lauten `admin/password`. Dieses sollte geändert werden. Muss dann aber im StackStorm Key-Value-Store entsprechend hinterlegt werden.

1. Festplatte (muss komplett leer sein -> initialized)
2. CD/DVD

Weitere Details siehe Bootreihenfolge festlegen, Seite 84.

8.3 Bare Metal Deployment

8.3.1 PXE

Zu Beginn wurde viel Zeit in die vermeintlich einzige Lösung PXE investiert. Für eine komplett automatisierte Installation des Betriebssystems müssen die Boot-Options mitgegeben werden, in denen das Kickstart-File [5] (Beschreibung der zu installierenden Parameter wie Netzwerkinterfaces, IP-Adresse, Hostname etc.) und sonstige Installationsquellen beschrieben sind. Dadurch kann jede manuelle Nutzereingabe während der Installation automatisiert werden. Da zu Beginn nur die Installation über PXE funktionierte, um die Boot-Options mitzugeben, wurde damit begonnen. Die Installation via GUI des CIMCs kann nicht automatisiert werden und die API des CIMCs bietet keine Schnittstelle dafür an.

PXE wurde aus Sicherheitsgründen von uns nicht mehr weiterverfolgt und eine alternative Lösung gesucht. Eine Anleitung der erarbeiteten Schritte befindet sich im Anhang (siehe PXE, Seite 92).

8.3.2 CIMC CLI

Weil PXE das unsichere TFTP-Protokoll verwendet, wurde eine alternative Lösung über die CIMC CLI angeschaut. Dabei wird zuerst das Image via HTTPS auf den entsprechenden UCS geladen, dort als CD/DVD virtuell eingehängt und dann davon gestartet.

Als Vorbedingung muss die Boot-Reihenfolge korrekt sein und die Festplattenpartitionen komplett geleert, damit automatisch vom CD-Image gestartet wird.

Weitere Details siehe Bootreihenfolge festlegen, Seite 84.

Über die SSH-Konsole des CIMC kann mit folgenden Befehlen das Image heruntergeladen und als CD eingehängt werden:

```
scope host-image-mapping
download-image https <ip-adresse> <dateipfad>/<dateiname>
map-image <dateiname>
exit
scope chassis
power on
```

Dies erledigt unser Python-Script mit dem Netmiko Framework (siehe Netmiko, Seite 55).

8.3.3 ISO-Packer

Für den Download des Betriebssystems muss das ISO-Image so gepackt werden, dass die gesamte Konfiguration darin enthalten ist und dadurch die Installation «unattended» durchgeführt werden kann. Dies wird in unserem ISO-Packer bewerkstelligt, der folgende Schritte durchläuft.

1. ISO Datei entpacken:
   `xorriso -osirrox on -indev /path/to/file.iso -extract / /temp/path`
2. Bootoptionen anpassen: Dies ist für CentOS [46] und ESXi [47] verschieden
   In den Ansible-Playbooks sind die Optionen ebenfalls ersichtlich
3. Kickstart-Files (siehe Kickstart Files, Seite 81) in das richtige Verzeichnis legen
4. ISO Image packen: Dies ist ebenfalls für CentOS [46] und ESXi [47] verschieden
   Siehe auch Ansible-Playbooks für weitere Details
5. Ordner `/temp/path` wieder löschen

8.4 OVS RPM Packen

Für den KVM-Host muss die Open vSwitch Software von der Source selbst gebuildet und als RPM gepackt werden, da in den offiziellen Betriebssystemrepositories keine aktuellen Versionen vorhanden sind.

Dies kann folgendermassen auf einem CentOS Host gemacht werden [21]:

```
yum install wget openssl-devel gcc make python-devel openssl-devel kernel-devel
    ↪ graphviz kernel-debug-devel autoconf automake rpm-build redhat-rpm-config
    ↪ libtool python-twisted-core python-zope-interface PyQt4 desktop-file-utils
    ↪ libcap-ng-devel groff checkpolicy selinux-policy-devel gcc-c++ python-six
    ↪ unbound unbound-devel -y

useradd ovs
su - ovs

mkdir -p ~/rpmbuild/SOURCES
wget https://www.openvswitch.org/releases/openvswitch-2.12.0.tar.gz
cp openvswitch-2.12.0.tar.gz ~/rpmbuild/SOURCES/
tar xfz openvswitch-2.12.0.tar.gz
rpmbuild -bb --nocheck openvswitch-2.12.0/rhel/openvswitch-fedora.spec
exit
```

Die RPM-Datei liegt nun im Ordner `rpmbuild` ▸ `RPMS` ▸ `x86_64` und heisst `openvswitch-2.12.0-1.el7.x86_64.rpm`.

8.5 Automatisierung

Als Automatisierungslösung wurde StackStorm mit Ansible-Integration evaluiert. Wir nutzen StackStorm, um den gesamten Workflow (siehe Workflows, Seite 27) zu orchestrieren.

8.5.1 Inventory

Das Inventory ist die «Single Source of Truth» für den gesamten Deployment-Prozess. Es wird vor jedem Deployment automatisch aus dem Infoblox generiert. Ein Beispiel eines generierten Inventories liegt in der Abgabe.

Es ist so konzipiert, dass es ein valides Ansible Inventory [34] erzeugt und auch vom StackStorm verarbeitet werden kann.

Funktionsweise

Die selbst entwickelte StackStorm Action `create_inventory` basiert auf einem Python Script. Diese Action verwendet das `infoblox_client` [18] Python Modul, um die benötigten Objekte aus dem Infoblox auszulesen.

Die Basis bildet das Extensible Attribute `sdi_site` (siehe Infoblox, Seite 44). Über dieses Attribut werden die folgenden Objekte zu einem Standort ausgelesen:

- Network Container
- Networks
- IPv4 Host Records

Weitere Konfigurationen des Inventories sind ausgelagert in die StackStorm SDI Pack Configuration `/opt/stackstorm/configs/sdi.yaml` (siehe StackStorm SDI Pack Installation und Konfiguration, Seite 81).

Networks

Die folgenden Informationen werden zu einem Network gespeichert. Alle Networks sind in einem Array als globale Gruppen Variable im Inventory abgelegt.

Parameter	Beschreibung	Beispiel
id	VLAN ID basiert auf dem Extensible Attribute `sdi_vlan`	2
type	VLAN Type basiert auf dem Extensible Attribute `sdi_vlan_type`	network_mgmt
subnet_mask	Subnetz Maske in Dotted Decimal Notation	255.255.255.0
network_id	Netzwerk ID in Dotted Decimal Notation	10.20.31.0
network_cidr	Netzwerk in der CIDR Notation	10.20.31.0/24

VLAN Types

Jedes Netzwerk, welches im Infoblox für einen Standort erfasst wird, braucht das Extensible Attribut `sdi_vlan_type` zugewiesen. Diese dienen als Kategorien, damit z.B. auf dem Trunk zum ESXi nicht einfach alle VLANs erlaubt sind, sondern nur die wirklich benötigten.

VLAN Type	Beschreibung	Anzahl
client	Für alle Client Netze an einem Standort	min. 1
dc_transit	Datacenter Transit Netz	1
network_mgmt	Netzwerk Management Netz	1
server_mgmt	Server Management Netz um die ESXi zu verwalten	1
server	Server Service Netze auf dem ESXi	min. 1
nfv	NFV Service Netz für die NFV Services	1
temporary	Staging VLAN für Firewall	1

Das `temporary` VLAN ist ein spezielles Netz. Dieses wird benötigt, um initial auf die Check Point Firewall zuzugreifen, da die Grundkonfiguration nicht automatisiert werden kann (siehe Gateway, Seite 57).

Dieses VLAN muss in der StackStorm Pack Configuration des SDI Packs (siehe SDI Pack Configuration, Seite 53) erfasst werden, da es nicht im Infoblox ist. Für jeden Standort wird dieses VLAN verwendet. Es ist jedoch nur intern zwischen dem ISR und dem KVM Host verfügbar.

```
vlans:
  - id: 11
    type: temporary
    default_gateway: 192.168.1.1
    subnet_mask: 255.255.255.0
    network_id: 192.168.1.0
    network_cidr: 192.168.1.0/24
```

Gruppen

Das Ansible Inventory basiert stark auf Gruppen von Hosts. Dadurch kann einfach eine Gruppe angegeben werden, auf welcher gewisse Tasks ausgeführt werden sollen.

Für das Deployment werden die folgenden Gruppen verwendet.

Gruppenname	Kürzel (Hostname)	Beschreibung
primary	01	alle Geräte mit einem **01** im Hostname werden dieser Gruppe hinzugefügt z.B. «05500aa-1adc-**01**-bdi2»
secondary	02	wenn es eine Redundanz an einem Standort gibt, werden die Hosts mit einem **02** dieser Gruppe hinzugefügt z.B. «05500aa-1adc-**02**-bdi2»
kvm	kv	Gruppe für die KVM Hypervisor z.B. «05500aa-1a**kv**-01»
esxi	es	Gruppe für die ESXi Hypervisor z.B. «05500aa-1a**es**-01»
routers	dc	Gruppe für die Cisco ISR z.B. «05500aa-1a**dc**-01-bdi2»
dhcp_servers	dh	Gruppe für die DHCP Server z.B. «05500aa-1a**dh**-01»
dns_servers	dn	Gruppe für die DNS Server z.B. «05500aa-1a**dn**-01»
firewalls	fw	Gruppe für die Firewall Gateways z.B. «cp05500aa-1a**fw**-01-eth0»

Special-Groups

Es existieren zwei spezielle Gruppen, die nicht als eigenständige Gruppe im Inventory auftauchen. Die CIMC der UCS Server sind ebenfalls im Infoblox eingetragen. Sie werden jedoch nicht in eigene Gruppen eingeteilt, sondern werden dem jeweiligen dazugehörigen UCS Host als Host Variable `ucs_ip` hinzugefügt.

Damit dies funktioniert, ist ein mapping notwendig und wird wie in folgendem Beispiel konfiguriert. Zum CIMC UCS Server Kürzel `ue` (ESXi) und `uk` (KVM) werden die Gruppennamen bzw. Kürzel erfasst.

```
special_groups:
  ue:
    long: esxi
    short: es
  uk:
    long: kvm
    short: kv
```

Listing 2: Beispiel Inventory Output kvm Gruppe

```
kvm:
  hosts:
    05500aa-1akv-01:
      ansible_host: 10.20.31.3
      interfaces:
      - ipv4: 10.20.31.3
        name: ''
        subnet_mask: 255.255.255.0
        vlan: 2
      ucs_ip: 10.20.31.4
```

Hosts

Grundsätzlich werden alle Infoblox Host Records eines Standortes einer Gruppe zugewiesen. Wenn es keine spezifische Gruppe gibt, wird der Host Record der globalen Gruppe `all` hinzugefügt. Zudem wird jede IP Adresse zusätzlich als Interface zum Host gespeichert.

Wenn ein Host Record im Infoblox das Extensible Attribute `sdi_ansible_host` besitzt, wird diese IP Adresse im Inventory Attribut `ansible_host` erfasst. Über diese IP Adresse wird auf den entsprechenden Host zugegriffen.

Zum Beispiel hat die Firewall mehrere Interfaces und somit auch mehrere Einträge im Infoblox. Im Inventory soll jedoch nur ein Host Objekt existieren. Die IP Adresse des Host Records mit dem Extensible Attribute `sdi_ansible_host` ist bei der Firewall die Management IP Adresse. Alle anderen Host Records der Firewall werden nur als Interfaces aufgenommen (siehe Listing 3).

Listing 3: Beispiel Inventory Output firewalls Gruppe

```
firewalls:
  hosts:
    cp05500aa-1afw-01:
      ansible_host: 10.20.31.2
      interfaces:
      - ipv4: 10.20.31.2
        name: eth0
        subnet_mask: 255.255.255.0
        vlan: 2
      - ipv4: 10.20.30.2
        name: eth1
        subnet_mask: 255.255.255.0
        vlan: 3
...
```

Group Variables

Zudem existieren die folgenden globalen Variablen im Inventory, die zum Teil ebenfalls von der StackStorm SDI Pack Configuration abhängig sind.

Variable	Beschreibung	Beispiel
datastore	Hostname S3 MinIO Object Store (DNS muss vom CIMC bei der Installation aufgelöst werden können)	minio.sdi.local
dns_server	DNS Server im Headquarter	152.96.120.54
site_domain_name	Der Site-Domain-Name setzt sich aus dem Sitename und dem konfigurierten Domainname im SDI Pack zusammen	05500aa.sdi.local
site_network	Das Site-Network ist die Network ID in CIDR Notation des Network Container eines Standorts	10.20.16.0/20
vlans	Dieser Array beinhaltet alle Networks (siehe Networks, Seite 49)	

8.5.2 StackStorm

Unsere Automatisierungs-Orchestrierung verwendet einige Konzepte, die im Weiteren erläutert sind.

SDI Pack Configuration

Weitere Details zur Installation und Konfiguration des StackStorm SDI Packs befinden sich unter StackStorm SDI Pack Installation und Konfiguration Seite 81.

Secrets

StackStorm bietet die Möglichkeit, Werte als Text oder verschlüsselt in einem Key-Value Store abzulegen [42]. Diese Werte können dann im Workflow angezogen und verwendet werden. Folgendermassen werden neue Werte in der CLI von StackStorm gespeichert:

```
st2 key set <key> <value>
st2 key set <key2> <password> --encrypt
```

Dabei ist zu beachten, wenn die Value, also z.B. der Public-Key, ein Leerzeichen beinhaltet, muss diese in Hochkommas geschrieben werden.

Im Workflow (.yaml-File) wird auf die Werte dann per Jinja2-Syntax zugegriffen:

```
"{{ st2kv('system.key') }}"
"{{ st2kv('system.key2', decrypt=true) }}"
```

Damit die Passwörter nicht im Plain-Text in den Logs ersichtlich sind, ist es notwendig, in den Ansible-Playbooks das Logging auszuschalten. Eine entsprechende Warnung macht zusätzlich darauf aufmerksam.

StackStorm Testing

Das Testing ist nicht so einfach, wie in einem Software-Engineering Projekt, denn es müssten Integrationstests geschrieben werden, um die Workflows zu testen. Wir haben uns damit beholfen, in jedem Workflow direkt eine Prüfung einzubauen, ob die nötigen Aufgaben ausgeführt wurden. Wenn dies nicht der Fall ist, wird das Deployment mit einer entsprechenden Fehlermeldung abgebrochen oder der Vorgang wiederholt durchgeführt.

Allerdings haben einen Docker Container erstellt[1], der einen YAML-Linter enthält, den wir im Git-Repository über den Continuous Integration (CI) Mechanismus anstossen. Dieser weist uns beim Einchecken der Dateien ins Repository auf einen allfälligen Fehler in der Syntax hin. Die Pipeline schlägt in diesem Fall fehl.

8.5.3 Ansible

Die meiste Automatisierungsarbeit wird von Ansible-Playbooks übernommen. Diese Playbooks werden durch das StackStorm angestossen und leisten eine erhebliche Arbeit für die Konfiguration der Geräte.

Weil die Playbooks sequenziell abgearbeitet werden und diese nahezu selbsterklärend sind, werden diese hier nicht im Detail aufgegriffen. Einzig das Check Point Management Modul ist erwähnenswert.

[1] https://github.com/HSRNetwork/docker-ansible-lint

Check Point Management

Für die Check Point Management API gibt es eine Ansible Galaxy Collection [6] von Check Point. Diese ist zurzeit noch stark in Entwicklung und wurde erst während der Bachelorarbeit released. Sie ist seit der Ansible Version 2.9 im Ansible Core. Es wird jedoch empfohlen, direkt von Ansible Galaxy die Collection zu installieren, da dort Updates schneller einfliessen.

Listing 4: Zugriff über die Collection

```
- name: Add Gateway to Check Point Management
  check_point.mgmt.cp_mgmt_simple_gateway:
    name: checkpoint_name
    ipv4_address: 10.20.31.100
    one_time_password: aaaa
    firewall: true
    gateway_version: R80.30
```

Ansible Testing

Es wird der gleiche YAML Syntax Check gemacht, wie bei den StackStorm Workflows. Auch hier können die Playbooks nicht besser getestet werden als die Workflows im StackStorm.

8.5.4 Weitere Packs

Netmiko

Netmiko [4] ist eine Python-Library, welche wir in einer selbst geschriebenen StackStorm Action verwenden.

Wir verwenden das Netmiko-Script, um das CIMC des Cisco Routers zu konfigurieren, das Betriebssystem-Image als CD/DVD einzulegen und komplett automatisiert zu installieren. Dies aus dem Grund, weil Napalm nicht in Frage kommt, da es keinen Treiber für das CIMC CLI gibt und Ansible eine Python-Library auf dem Zielsystem benötigt.

Ein weiterer Vorteil liegt darin, dass die Ausgabe auf der stdout- / stderr-Pipeline live im Output von StackStorm mitverfolgt werden kann. Dies vereinfacht die Fehleranalyse.

Napalm

Napalm [2] ist eine Python-Library, die auf Netmiko basiert und speziell für die automatisierte Konfiguration von Netzwerkgeräten entwickelt wurde. Es gibt ein Pack für StackStorm [19], welches auch direkt über das GUI von StackStorm installiert werden kann.

Der Vorteil liegt darin, dass der Cisco Router über einen vorhandenen Treiber direkt angesprochen werden kann und so ein Zusammenführen (Merge) der vorhandenen Konfiguration mit den Änderungen sehr einfach möglich ist. Über den Treiber kann festgelegt werden, was für ein Endgerät dahinter ist. Dadurch kann die gleiche Library für IOS basierte Geräte, aber auch für Geräte anderer Hersteller eingesetzt werden.

8.6 Services

8.6.1 Check Point

Management

Das Setup des Check Point Managements wird im Anhang detailliert beschrieben (siehe Check Point Management Installation, Seite 82).

Danach muss die API aktiviert werden. Dies wird am einfachsten in der Smart-Console erledigt:

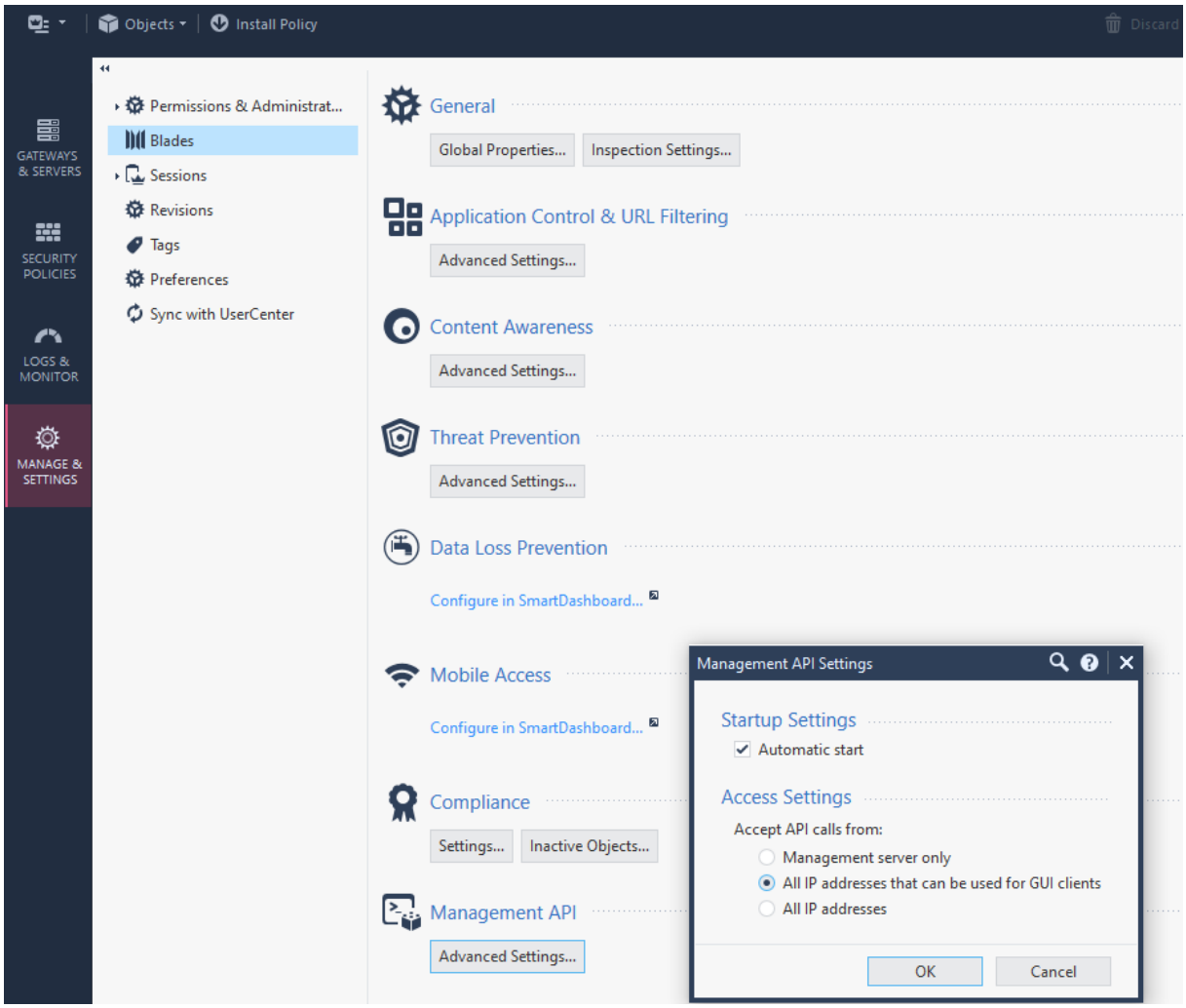


Abbildung 19: Check Point Mgmt API aktivieren - Smart-Console

Oder per CLI:

```
mgmt_cli -r true --domain MDS set api-settings accepted-api-calls-from "All IP
    ↪ addresses"
```

Zum Schluss muss die API noch via CLI aktiviert werden:

```
cp-mgmt> api restart
2019-Oct-21 13:35:42 - Stopping API...
2019-Oct-21 13:35:45 - API stopped successfully.
2019-Oct-21 13:35:45 - Starting API...
```

Gateway

An jedem Standort wird ein eigener Check Point Security Gateway betrieben. Dabei ist zu erwähnen, dass dieser zwar weiterhin funktioniert, wenn die Verbindung zum Management verloren geht, Änderungen an der Konfiguration aber vorübergehend nicht mehr möglich sind.

Für die Installation werden unterschiedliche Möglichkeiten in Betracht gezogen:

Blink Image Die Konfiguration wird dabei aus der Cloud gezogen [26]. Die Automatisierung dieser Variante stellt sich als sehr komplex dar.

vSEC Das vSEC Image ist für Cloud-Deployments geeignet. Da zu Beginn aber nur eine Console-Connection und noch keine IP-Connectivity vorhanden ist, kann dieses Image nicht als mögliche Lösung für das automatisierte Deployment verwendet werden.

ISOMorphic Dies ist ein Tool von Check Point, um einen bootable USB-Stick zu erstellen [32]. Diese Variante scheint aber nicht möglich zu automatisieren.

eigenes Image Ein eigens erstelltes KVM-Image mit einer Grundkonfiguration.

Weil es keine Kickstart-Konfiguration oder eine ähnliche «unattended» Installations-Methode gibt, haben wir uns entschieden, ein eigenes KVM-Image zu erstellen, welches eine fixe Grundkonfiguration beinhaltet. Dieses Image kann vor Ort kopiert und als VM gestartet werden, wobei folgende Konfiguration im Image enthalten ist (an jedem Standort):

- Partitionierung
- Management Interface: eth0
- Management IP-Adresse: 192.168.1.10
- Subnetzmaske: 255.255.255.0
- Default-Gateway: 192.168.1.1

Beim ersten Verbinden mit der Check Point Konsole gilt es einiges zu beachten.

Nachdem die Virtuelle Maschine definiert ist und das ISO-Image eingelegt wurde, muss die VM gestartet werden.

```
sudo qemu-img create -f qcow2 /opt/sdi/images/base_gateway_raw.qcow2 100G

sudo virt-install \
   --virt-type kvm \
   --name base_gateway \
   --ram 4096 \
   --vcpus 2 \
   --disk path=/opt/sdi/base_gateway_raw.qcow2,device=disk,format=raw \
   --network bridge=ovsbr0,model=virtio,virtualport_type=openvswitch \
   --os-type=linux \
   --os-variant=generic \
   --noautoconsole \
   -c /opt/sdi/Check_Point_R80.30_T200_Security_Gateway.iso

sudo virsh console base_gateway
```

Nach dem Start bleiben 60 Sekunden Zeit, auf die Konsole zuzugreifen und mit den Pfeiltasten «Install GAIA on this System» auszuwählen. Leider ist dieser Text auf der Console hinter den anderen Befehlen etwas versteckt. Mit den Pfeiltasten navigiert man ans richtige Ort:

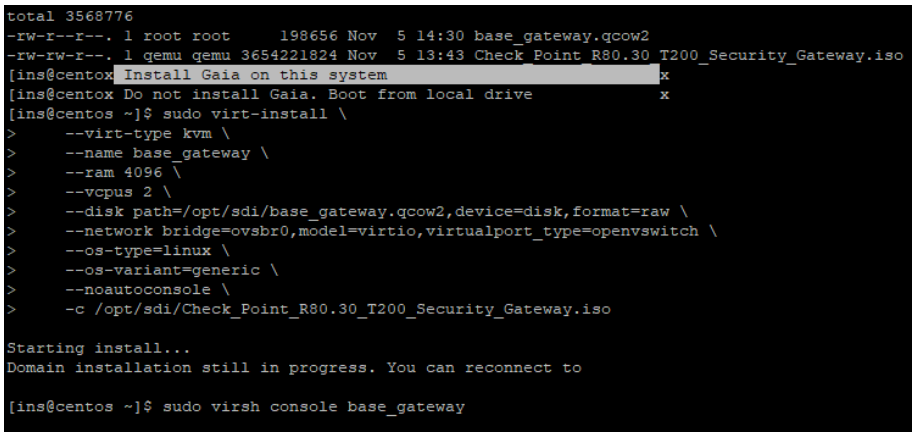


Abbildung 20: Check Point initial setup – Installationsprozess in der Konsole starten

Partitionierung der Check Point Festplatte [31]:

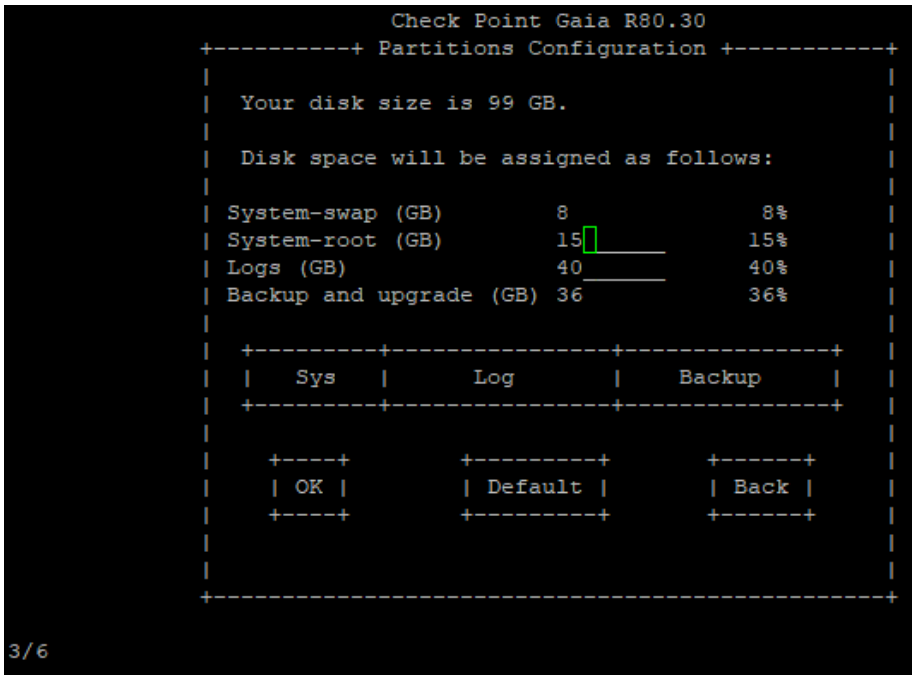


Abbildung 21: Check Point initial setup – Partitionierung

IP Konfiguration:

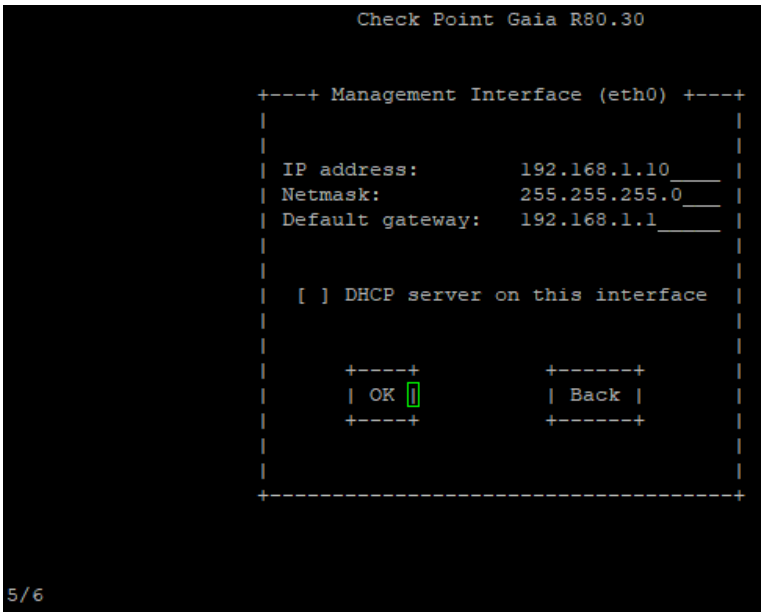


Abbildung 22: Check Point initial setup – Netzwerk Einstellungen

Danach muss die VM heruntergefahren und das `.qcow2`-File mit folgendem Befehl komprimiert werden, wodurch etwa 50% der Festplattengrösse eingespart werden kann:

```
qemu-img convert -f qcow2 -O qcow2 -c base_gateway_raw.qcow2 compressed.qcow2
```

Konfiguration

Die Konfiguration wird automatisiert mittels Checkpoint First Time Configuration Wizard (Checkpoint FTW) [29] durchgeführt. Dabei werden folgende Parameter mitgegeben (Die Argumente sind in den Ansible Playbooks parametrisiert und somit pro Deployment spezifisch für den Standort).

```
config_system --config-string "hostname=checkpoint_gateway&domainname=05500AA.sdi.
    ↪ local&timezone='Europe/Zurich'&install_security_gw=true&gateway_daip=false&
    ↪ install_security_managment=false&primary=152.96.120.54&download_info=true&
    ↪ upload_info=true&ntp_primary=pool.ntp.org&ntp_primary_version=3&ftw_sic_key=
    ↪ aaaa&reboot_if_required=false"
```

Erklärungen:

primary Primärer DNS-Server
ftw_sic_key One-Time-Passwort für die erste Verknüpfung mit dem Check Point Management Server
reboot_if_required Der Gateway wird vom Deployment Prozess neugestartet, deshalb ist der Wert hier auf false

NAT

Um den Checkpoint FTW durchzuführen, muss ein Network Address Translation (NAT) Port Forwarding für ein temporäres BDI auf dem ISR eingerichtet werden. Das NAT übersetzt die vom Headquarter erreichbare Outside-IP-Adresse des ISR (Port 2222) auf die initiale, temporäre IP-Adresse des Check Point Gateways (192.168.1.10, Port 22).

```
interface BDI11
 ip address 192.168.1.1 255.255.255.0
 ip nat inside
 no shut

interface ucse2/0/1
 service instance 11 ethernet
  encapsulation dot1q 11
  rewrite ingress tag pop 1 symmetric
  bridge-domain 11
  no shut

interface BDI2
 ip nat outside
 exit

ip nat inside source static tcp 192.168.1.10 22 10.20.31.10 2222 extendable
```

Logs

Die Logs werden grundsätzlich auf dem Management-Server im Headquarter gespeichert [28]. Dort kann definiert werden, wie lange die Logs verwahrt werden sollen. Auch gibt es die Möglichkeit, die Logs in ein externes Tool zu exportieren (z.B. PRTG, Splunk oder Elasticsearch).

Auch werden die Logs bis zu einem bestimmten Punkt lokal auf dem Gateway gespeichert. Im Falle eines Unterbruchs der Leitung ins Headquarter werden die Logs lokal auf dem Gateway zwischengespeichert.

8.6.2 NFV Services

Die NFV-Services werden über ein Cloud-Image [33] deployt. Das bedeutet, dass die Installation komplett unattended vonstattengehen kann. Dies wird folgendermassen gemacht [39]:

Drei Konfigurationsdateien werden geschrieben und zu einem Image zusammengepackt. Dieses Image wird dann als zusätzliches Laufwerk eingehängt.

Beim Start der VM läuft das – im Cloud-Image vorinstallierte – cloud-init Programm los und konfiguriert das System mit den nötigen Parametern.

meta-data

```
dsmode: local
local-hostname: centos_hostname
```

user-data

Die Datei muss zwingend die erste Zeile `#cloud-config` aufweisen.

```
#cloud-config
hostname: centos.05500aa.sdi.local
user: <username>
password: <password>
chpasswd: { expire: False }
ssh_pwauth: True
ssh_authorized_keys:
  - <kompletter public key>
```

network-config

Die Datei muss zwingend die erste Zeile `#cloud-config` aufweisen.

```
#cloud-config
version: 1
config:
  - type: physical
    name: enp0s3
    subnets:
      - type: static
        address: 10.20.31.150/24
        gateway: 10.20.31.1
  - type: nameserver
    address:
      - 152.96.120.54
    search:
      - 05500aa.sdi.local
```

Image packen

```
virt-make-fs --type=vfat --label=cidata /path/to/source /path/to/destination/cidata.
    ↪ vfat
```

Diese `.vfat`-Datei wird nun als Disk in die VM eingehängt. Das geschieht beim Definieren der VM im XML:

```
[...]
<disk type='file' device='disk'>
  <driver name='qemu' type='raw'/>
  <source file='/path/to/destination/cidata.vfat'/>
  <target dev='vdb' bus='virtio'/>
</disk>
[...]
```

Nach dem Boot-Vorgang ist die Konfiguration abgeschlossen und es muss ein erneutes Anlaufen des cloud-init Programmes verhindert werden [14]:

```
touch /etc/cloud/cloud-init.disabled
```

DHCP Server

Zum Einsatz kommt ein «dhcpd»-Dienst des Internet Systems Consortium (ISC). Die Konfiguration [20] ist verhältnismässig sehr einfach und die Funktionsweise hat sich über die Jahre etabliert.

In der Konfigurationsdatei `etc ▸ dhcp ▸ dhcpd.conf` braucht es für den eigenen IP-Range und für jedes Client-Netz eine Subnet-Definition.

```
subnet 10.20.35.0 netmask 255.255.255.0 {
   option routers 10.20.35.1;
   option subnet-mask 255.255.255.0;
}
# example client network
subnet 10.20.32.0 netmask 255.255.255.0 {
   option routers 10.20.32.1;
   option subnet-mask 255.255.255.0;
   range 10.20.32.20 10.20.32.200;
}
```

Details zum Setup kann dem Konfigurations-Link oben oder dem Ansible Playbook direkt entnommen werden.

DNS Server

Als DNS-Server verwenden wir den «bind»-Dienst, der ebenfalls vom ISC zur Verfügung gestellt wird. Konfigurieren [41] lässt er sich über eine einfache Datei, welche unter `etc ▸ named.conf` konfiguriert werden kann.

Der DNS-Server kann in einem redundanten Setup betrieben werden, wobei der Backup-DNS Server die Konfiguration vom Primary automatisch übernimmt. Wichtig dabei ist, dass bei jeder Änderung die `Serial` geändert wird, da sonst keine Konfigurationsänderung erkannt wird. Es ist üblich, diese mit dem aktuellen Datum und einer fortlaufenden, zweistelligen Nummer `xx` zu beschreiben: `yyyymmddxx`

Beispiel:

```
$TTL 86400
@           IN  SOA       ns1.05500bb.sdi.local. root.05500bb.sdi.local. (
            2019112701    ;Serial                 <---------
            3600          ;Refresh
            1800          ;Retry
            604800        ;Expire
            86400         ;Minimum TTL
)
@           IN  NS        ns1.05500bb.sdi.local.
@           IN  A         10.20.35.3              ; IP DNS Server
ns1         IN  A         10.20.35.3              ; IP DNS Server
server_xy   IN  A         10.20.35.2
```

Details zum Setup kann dem Konfigurations-Link oben oder dem Ansible Playbook direkt entnommen werden.

8.7 Probleme

Einige Probleme sind noch offen und nicht vollständig geklärt. Diese werden im Folgenden genauer aufgeführt.

8.7.1 CIMC Image Download

Aus unerklärlichen Gründen ist der ISO-Datei Download des CIMCs extrem langsam, wenn der Download über eine gesicherte HTTPS-Verbindung heruntergeladen wird. Es wird vermutet, dass es mit dem Lab-Aufbau oder einem Bug der CIMC-Firmware zu tun hat.

Wird die gleiche Datei in einer VM auf dem Hypervisor heruntergeladen, funktioniert der Download einwandfrei und sehr schnell.

Als Workaround wurde der Download über eine ungesicherte HTTP-Verbindung bewerkstelligt.

8.7.2 Check Point Gateway

Netzwerkadapter

Lange Zeit verursachte das automatisierte Deployment des Checkpoint Gateways Probleme. Nach dem Konfigurieren der VM gibt diese plötzlich keine Antwort mehr. Das Problem tritt jedoch nicht bei jedem Deployment auf. Sobald die temporäre `192.168.1.10` IP-Adresse (siehe NAT, Seite 60) des `eth0` Interfaces auf die korrekte Management IP-Adresse gewechselt wird, konnte man auf die neue IP-Adresse nicht zugreifen. Zudem war es auch nicht möglich von der Check Point selbst nach draussen zu pingen (von dieser IP-Adresse). Von einer anderen IP-Adresse der Firewall funktionierte es weiterhin.

Wenn man das Interface `eth0` deaktivierte und wieder aktivierte, zeigte es `NO-CARRIER` an.

```
[Expert@cp05500aa-1afw-01:0]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
   link/loopback 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
   inet 127.0.0.1/8 brd 127.255.255.255 scope host lo
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue qlen 1000
   link/ether 52:54:00:61:c5:f3 brd ff:ff:ff:ff:ff:ff
   inet 10.20.31.2/24 brd 10.20.31.255 scope global eth0
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue qlen 1000
   link/ether 52:54:00:f2:b3:f1 brd ff:ff:ff:ff:ff:ff
   inet 10.20.30.2/24 brd 10.20.30.255 scope global eth1
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue qlen 1000
   link/ether 52:54:00:8a:fa:f8 brd ff:ff:ff:ff:ff:ff
   inet 10.20.19.1/24 brd 10.20.19.255 scope global eth2
5: eth3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue qlen 1000
   link/ether 52:54:00:8f:b8:b0 brd ff:ff:ff:ff:ff:ff
   inet 10.20.16.2/24 brd 10.20.16.255 scope global eth3
6: eth4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue qlen 1000
   link/ether 52:54:00:da:9b:01 brd ff:ff:ff:ff:ff:ff
   inet 10.20.17.2/24 brd 10.20.17.255 scope global eth4
7: eth5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue qlen 1000
   link/ether 52:54:00:c7:04:b6 brd ff:ff:ff:ff:ff:ff
   inet 10.20.18.1/24 brd 10.20.18.255 scope global eth5
8: eth6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue qlen 1000
   link/ether 52:54:00:79:18:50 brd ff:ff:ff:ff:ff:ff
   inet 10.20.20.1/24 brd 10.20.20.255 scope global eth6
```

```
[Expert@cp05500aa-1afw-01:0]# ping 10.20.19.2
PING 10.20.19.2 (10.20.19.2) 56(84) bytes of data.
64 bytes from 10.20.19.2: icmp_seq=1 ttl=64 time=2.68 ms
64 bytes from 10.20.19.2: icmp_seq=2 ttl=64 time=0.501 ms
64 bytes from 10.20.19.2: icmp_seq=3 ttl=64 time=0.598 ms
64 bytes from 10.20.19.2: icmp_seq=4 ttl=64 time=0.557 ms

--- 10.20.19.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.501/1.084/2.681/0.922 ms
```

Schlussendlich haben wir herausgefunden, dass es allenfalls an der Netzwerkadapterkonfiguration der KVM VM liegt. Anstatt einen `E1000` Netzwerkadapter zu verwenden, konnten wir das Problem lösen, indem wir auf einen `virtio` Netzwerkadapter wechselten [22].

8.7.3 CIMC

Wie bereits beschrieben (siehe Boot Reihenfolge, Seite 47), kann die Boot-Reihenfolge des UCS-Servers nicht über die CLI des CIMC verändert werden, obwohl dies gemäss vorhandenen Parametern und Dokumentation (siehe [9]) funktionieren sollte. Dies könnte ein Software-Bug im CIMC sein.

8.8 Nächste Schritte

8.8.1 CE Router Konfiguration

Gewisse statische Routen und IP-Helper Adressen für den DHCP Service müssen nach dem Deployment zurzeit noch von Hand hinzugefügt werden (siehe Manuelle CE Config, Seite 45).

Dies müsste definiert werden, ob dies bereits zur Grundkonfiguration des CE Routers gehört, oder ob dies am Ende des Deployment Workflows noch konfiguriert wird.

8.8.2 High Availability

Das Automatisierungs-Tool ist ausgelegt dafür, dass ein Standort mit zwei Routern provisioniert werden kann.

In der Version `R80.30` unterstützt die API des Check Point Management nur ein Deployment eines «Simple Gateways». Ein ClusterXL kann noch nicht konfiguriert werden über die API. Zurzeit wäre es möglich dies über die CLI zu automatisieren, konnte jedoch aufgrund von Zeitmangel nicht mehr umgesetzt werden.

Zudem ist angekündigt[2], dass in der Version `R80.40` das Cluster Deployment vom Management unterstützt wird.

Als weiteren Schritt muss das HA Konzept noch mit der physischen Hardware getestet werden und der Workflow um die spezifischen Konfigurationen (Unterschiede in Active und Standby Konfiguration) erweitert werden.

8.8.3 Inventory Script

Schema Checker parametrisieren

Der Schema Checker ist ein Python-Script, welches eine YAML-Struktur gegenüber einem JSON-Schema testet (die Schemas liegen im StackStorm SDI-Pack unter `actions` ▸ `lib` ▸ `schemas`).

Er prüft das Inventory sehr statisch auf seinen Inhalt. Zum Beispiel wird momentan geprüft, ob ein XS-Standort genau sechs Geräte enthält. Dies kann mit Jinja2-Syntax stark parametrisiert werden, wobei sehr dynamisch auf Veränderungen im Deployment reagiert werden könnte.

[2] https://checkpoint.engineer/ea/r80-40-ea-program/

9 Anhang

9.1 Constraints

Für die Bachelorarbeit sind pro Person 360 Arbeitsstunden (12 ECTS Credits à 30h) vorgesehen. Die Arbeit beinhaltet neben der Analyse in diesem Dokument auch die beschriebene praktische Umsetzung.

9.2 Projektplanung

9.2.1 Projektorganisation

Dozent: Laurent Metzger
Technischer Assistent: Philip Schmid
Externer Auftraggeber: Laurent Billas und Serge Pidoux (FUB)
Projektleitung: Pirmin Wenk und Yannick Zwicker

9.2.2 Projektphasenplanung

Die Projektphasen werden nach dem Rational Unified Process (RUP) eingeteilt. Dabei werden vier Phasen verwendet, die jeweils agil gestaltet sind.

Geplant ist eine Fertigstellung und Abgabe der Arbeit in der Woche 14 (KW51), vor dem Jahreswechsel. Nach den zweiwöchigen Ferien werden dann noch die Präsentation und die Einreichung aller nötigen Dokumente erfolgen.

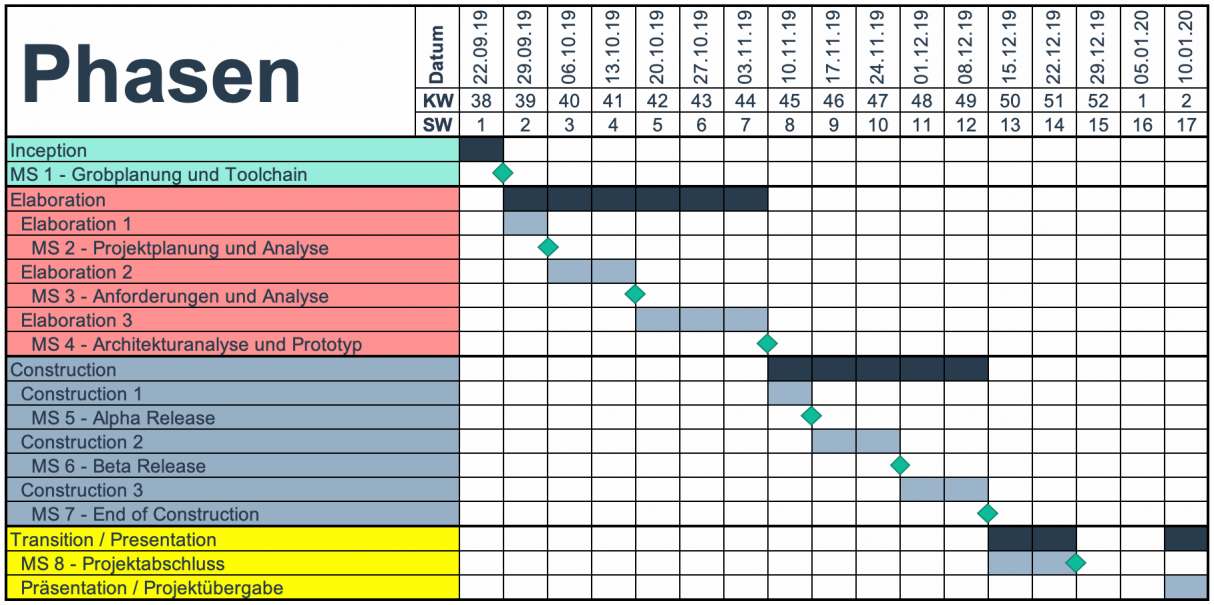


Abbildung 23: Projektphasenplanung Gantt

Beschreibung der einzelnen Phasen

Inception

Die Inception dauert eine Woche und beinhaltet den ersten Meilenstein.

Darin enthalten sind das Kick-off, sowie das Aufsetzen der Toolchain und eine Grobplanung der Arbeit. Beim Erreichen des Meilensteins soll alles soweit bereit sein, dass mit der Planung und der Analyse begonnen werden kann.

Elaboration

Die Elaboration-Phase dauert sechs Wochen und beinhaltet drei Meilensteine.

Darin enthalten sind die detaillierte Projektplanung, die Analyse der Software und ein entsprechender Prototyp.

Die End of Elaboration stellt den Übergang von Testing / Prototyping zur Umsetzung dar. Zudem steht die Tool-Chain für das produktive Arbeiten zur Verfügung. Die Risiken für die Construction Phase sollen, wenn möglich, alle behoben oder miteingeplant sein. Eine Grobplanung der Construction-Phase ist hier bereits erstellt.

Construction

Die Construction-Phase dauert fünf Wochen und beinhaltet drei Meilensteine.

Hier wird das Produkt entwickelt und getestet.

Eine detaillierte Planung wird in einem separaten Kapitel behandelt (siehe Planung der Sprints, Seite 67).

Transition

Die Transition dauert insgesamt drei Wochen und beinhaltet den letzten Meilenstein.

Diese Zeit wird dazu genutzt, die Präsentation, Dokumentation, Abstract und das Poster fertigzustellen und abzuschliessen und die weiteren Schritte im Vorgehen nach der Arbeit zu besprechen.

Diese Phase endet mit der Ab- bzw. Übergabe der Arbeit an die Betreuenden.

9.2.3 Arbeitspakete

Die Verwaltung der Arbeitspakete wird aktiv auf GitLab abgewickelt. Diese Pakete können dort direkt während des Verlaufs der Arbeit eingesehen werden[3].

Die Arbeitspakete werden auf GitLab erfasst und die dafür benötigte Zeit wird dort geschätzt. Zusätzlich werden diese den Meilensteinen und einem Label zugeordnet. Das Label zeigt die Kategorie der Arbeit (Analyse, Diskussion, Dokumentation, Entwicklung oder Infrastruktur) und den aktuellen Fortschritt (Wartend, In Arbeit oder Erledigt). In der Construction-Phase werden die priorisierten Tasks in eine Iteration aufgenommen und abgearbeitet. Eine Iteration während der Construction-Phase ist jeweils ein Sprint, in welchem der Backlog geschätzt und priorisiert wird. Im Issue-Board ist dann die Übersicht, welche Pakete welchen Status besitzen, und wem die verschiedenen Arbeitspakete zugewiesen sind. Nebenbei werden die benötigten Zeiten direkt auf die Arbeitskategorien verbucht, welche dann auch für die Zeitauswertung verwendet werden.

9.2.4 Planung der Sprints

Die Planung der Sprints erfolgt in GitLab. Dabei werden die Arbeitspakete jeweils zu Beginn des jeweiligen Sprints definiert, geschätzt und nach Priorität sortiert. Dies ist im Folgenden für alle Meilensteine ersichtlich.

[3] https://gitlab.com/groups/software-defined-infrastructure/-/boards (privates Repo)

Inception

Sprint 1

ISR einbauen und in Betrieb nehmen
management#15 · opened 6 days ago by Yannick Zwicker MS1 - Grobplanung und Toolchain infrastructure

ISR KVM Modul Analyse
management#16 · opened 5 days ago by Yannick Zwicker MS1 - Grobplanung und Toolchain analysis

Konzept Entwurf - Analyse Aufgabenstellung
management#19 · opened 5 days ago by Yannick Zwicker MS1 - Grobplanung und Toolchain discussion

Toolchain einrichten
management#17 · opened 5 days ago by Yannick Zwicker MS1 - Grobplanung und Toolchain infrastructure

Grobplanung erstellen
management#18 · opened 5 days ago by Yannick Zwicker MS1 - Grobplanung und Toolchain documentation

Dokumentation einrichten
management#20 · opened 5 days ago by Yannick Zwicker MS1 - Grobplanung und Toolchain documentation

Meeting W1
management#1 · opened 6 days ago by Yannick Zwicker MS1 - Grobplanung und Toolchain discussion

Elaboration

Sprint 2

Risikos definieren
management#30 · opened 1 day ago by Yannick Zwicker MS2 - Projektplanung und Analyse documentation

CIMC API Test
management#29 · opened 1 day ago by Yannick Zwicker MS2 - Projektplanung und Analyse analysis

Unattended Hypervisor Installation
management#28 · opened 1 day ago by Yannick Zwicker MS2 - Projektplanung und Analyse analysis

Infoblox Installation und Setup
management#27 · opened 1 day ago by Yannick Zwicker MS2 - Projektplanung und Analyse infrastructure

Analyse Basic Network Setup ISR/UCS
management#26 · opened 1 day ago by Yannick Zwicker MS2 - Projektplanung und Analyse analysis

Sample KVM VM manual Deployment
management#25 · opened 1 day ago by Yannick Zwicker MS2 - Projektplanung und Analyse analysis

UCS-E Server Installation ESX
management#24 · opened 1 day ago by Yannick Zwicker MS2 - Projektplanung und Analyse infrastructure

UCS-E Server Installation KVM
management#23 · opened 1 day ago by Yannick Zwicker MS2 - Projektplanung und Analyse infrastructure

Projektplanung mit Doku
management#22 · opened 1 day ago by Yannick Zwicker MS2 - Projektplanung und Analyse documentation

Images für UCS beschaffen
management#21 · opened 1 day ago by Yannick Zwicker MS2 - Projektplanung und Analyse infrastructure

Meeting W2
management#2 · opened 1 week ago by Yannick Zwicker MS2 - Projektplanung und Analyse discussion

Sprint 3

Meeting W3
management#3 · opened 2 weeks ago by Yannick Zwicker MS3 - Anforderungen und Analyse discussion

CIMC API Test
management#29 · opened 1 week ago by Yannick Zwicker MS3 - Anforderungen und Analyse analysis

Unattended Hypervisor Installation
management#28 · opened 1 week ago by Yannick Zwicker MS3 - Anforderungen und Analyse analysis

HA Konzept entwerfen
management#34 · opened 2 days ago by Pirmin Wenk MS3 - Anforderungen und Analyse analysis

Checkpoint einarbeiten
management#31 · opened 2 days ago by Pirmin Wenk MS3 - Anforderungen und Analyse analysis

Checkpoint Konzept
management#32 · opened 2 days ago by Pirmin Wenk MS3 - Anforderungen und Analyse discussion

Checkpoint Setup
management#33 · opened 2 days ago by Pirmin Wenk MS3 - Anforderungen und Analyse infrastructure

Projektplanung mit Doku
management#22 · opened 1 week ago by Yannick Zwicker MS3 - Anforderungen und Analyse documentation

Infoblox Installation und Setup
management#27 · opened 1 week ago by Yannick Zwicker MS3 - Anforderungen und Analyse infrastructure

Software-Architekturanforderungen definieren
management#38 · opened 2 days ago by Pirmin Wenk MS3 - Anforderungen und Analyse documentation

Automatisierungssoftware evaluieren
management#35 · opened 2 days ago by Pirmin Wenk MS3 - Anforderungen und Analyse analysis

Automatisierungssoftware Konzeptentwurf
management#39 · opened 2 days ago by Pirmin Wenk MS3 - Anforderungen und Analyse documentation

Komponentendiagramm entwerfen
management#37 · opened 2 days ago by Pirmin Wenk MS3 - Anforderungen und Analyse documentation

Use Cases definieren
management#36 · opened 2 days ago by Pirmin Wenk MS3 - Anforderungen und Analyse documentation

Sprint 4

Meeting W5
management#5 · opened 3 weeks ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp discussion

HA Konzept Hypervisor
management#41 · opened 3 minutes ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp analysis

CheckPoint Ansible Library Test
management#40 · opened 25 minutes ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp analysis

Projektplanung mit Doku
management#22 · opened 2 weeks ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp documentation

Automatisierungssoftware evaluieren
management#35 · opened 1 week ago by Pirmin Wenk MS4 - Architekturanalyse und Prototyp analysis

Software-Architekturanforderungen definieren
management#38 · opened 1 week ago by Pirmin Wenk MS4 - Architekturanalyse und Prototyp documentation

Automatisierungssoftware Konzeptentwurf
management#39 · opened 1 week ago by Pirmin Wenk MS4 - Architekturanalyse und Prototyp documentation

Komponentendiagramm entwerfen
management#37 · opened 1 week ago by Pirmin Wenk MS4 - Architekturanalyse und Prototyp documentation

Use Cases definieren
management#36 · opened 1 week ago by Pirmin Wenk MS4 - Architekturanalyse und Prototyp documentation

Komponentendiagramm entwerfen
management#37 · opened 3 weeks ago by Pirmin Wenk MS4 - Architekturanalyse und Prototyp documentation

Automatisierungssoftware Konzeptentwurf
management#39 · opened 3 weeks ago by Pirmin Wenk MS4 - Architekturanalyse und Prototyp documentation

Software-Architekturanforderungen definieren
management#38 · opened 3 weeks ago by Pirmin Wenk MS4 - Architekturanalyse und Prototyp documentation

Use Cases definieren
management#36 · opened 3 weeks ago by Pirmin Wenk MS4 - Architekturanalyse und Prototyp documentation

Projektplanung mit Doku
management#22 · opened 4 weeks ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp documentation

HA Konzept Hypervisor
management#41 · opened 1 week ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp analysis

Meeting W6
management#6 · opened 1 month ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp discussion

DHCP Server VM
management#43 · opened 1 week ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp analysis

DNS Server VM
management#42 · opened 1 week ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp analysis

Evaluation Docker oder ähnliches für NFV Services
management#44 · opened 1 week ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp documentation

DNS Server VM
management#42 · opened 1 week ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp analysis

DHCP Server VM
management#43 · opened 1 week ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp analysis

Meeting W7
management#7 · opened 1 month ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp discussion

StackStorm - ISR Config via Napalm
stackstorm-sdi#1 · opened 1 hour ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp development

StackStorm - UCS Host Deployment
stackstorm-sdi#2 · opened 1 hour ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp development

StackStorm / Ansible - UCS Host ISO erstellen
stackstorm-sdi#3 · opened 1 hour ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp development

Ansbile - OVS and KVM packages installation
stackstorm-sdi#4 · opened 1 hour ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp development

Ansible - OVS Base Config
stackstorm-sdi#5 · opened 1 hour ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp development

Ansible - Libvirt deploy VM
stackstorm-sdi#6 · opened 1 hour ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp development

StackStorm - Deploy Branch Webhook
stackstorm-sdi#7 · opened 1 hour ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp development

Zwischenpräsentation erstellen
management#45 · opened 1 hour ago by Yannick Zwicker MS4 - Architekturanalyse und Prototyp discussion

Construction

Sprint 5

Meeting W8
management#8 · opened 1 month ago by Yannick Zwicker MS5 - Alpha Release discussion

Ansible Testing/Linting
stackstorm-sdi#11 · opened 15 minutes ago by Pirmin Wenk MS5 - Alpha Release development

Stackstorm Testing
stackstorm-sdi#10 · opened 16 minutes ago by Pirmin Wenk MS5 - Alpha Release development

CentOS ISO Packing Image-Test deaktivieren
stackstorm-sdi#9 · opened 19 minutes ago by Pirmin Wenk MS5 - Alpha Release development

Stackstorm Workflow ISO-Dateien löschen
stackstorm-sdi#8 · opened 29 minutes ago by Pirmin Wenk MS5 - Alpha Release development

LaTeX Dokumentation nachführen
management#48 · opened 3 minutes ago by Pirmin Wenk MS5 - Alpha Release documentation

Zwischenpräsentation erstellen
management#45 · opened 1 week ago by Yannick Zwicker MS5 - Alpha Release documentation

Zwischenpräsentation durchführen
management#47 · opened 6 minutes ago by Pirmin Wenk MS5 - Alpha Release discussion

Netzwerkdesign überarbeiten inkl. Namens- und IP-Konzept
management#46 · opened 1 week ago by Yannick Zwicker MS5 - Alpha Release documentation

Sprint 6

Meeting W9
management#9 · opened 1 month ago by Yannick Zwicker · MS6 - Beta Release · discussion

Zwischenpräsentation erstellen
management#45 · opened 2 weeks ago by Yannick Zwicker · MS6 - Beta Release · documentation

UC03 - unterschiedliche Standortgrössen
stackstorm-sdi#18 · opened 5 hours ago by Yannick Zwicker · MS6 - Beta Release · development

UC03.1 - Gather Information from Infoblox
stackstorm-sdi#12 · opened 5 hours ago by Yannick Zwicker · MS6 - Beta Release · development

UC01/UC02 - Frontend Designentwürfe / Mockup
management#50 · opened 4 hours ago by Pirmin Wenk · MS6 - Beta Release · documentation

Meeting W10
management#10 · opened 1 month ago by Yannick Zwicker · MS6 - Beta Release · discussion

Zwischenpräsentation durchführen
management#47 · opened 1 week ago by Pirmin Wenk · MS6 - Beta Release · discussion

UC03.5 - CheckPoint Gateway im Mgmt hinzufügen
stackstorm-sdi#14 · opened 5 hours ago by Yannick Zwicker · MS6 - Beta Release · development

UC03.5 - CheckPoint Gateway konfigurieren
stackstorm-sdi#15 · opened 5 hours ago by Yannick Zwicker · MS6 - Beta Release · development

StackStorm store secrets securely
stackstorm-sdi#13 · opened 5 hours ago by Yannick Zwicker · MS6 - Beta Release · infrastructure

Sprint 7

Meeting W11
management#11 · opened 2 months ago by Yannick Zwicker · MS7 - End of Construction · discussion

UC03.1 - Gather Information from Infoblox
stackstorm-sdi#12 · opened 2 weeks ago by Yannick Zwicker · MS7 - End of Construction · development

UC03.6 - NFV Services deployen
stackstorm-sdi#16 · opened 2 weeks ago by Yannick Zwicker · MS7 - End of Construction · development

UC03.5 - CheckPoint Gateway konfigurieren
stackstorm-sdi#15 · opened 2 weeks ago by Yannick Zwicker · MS7 - End of Construction · development

UC03.1 - Inventory Python Script
stackstorm-sdi#23 · opened 3 minutes ago by Pirmin Wenk · MS7 - End of Construction · development

FIX - iteritems() to items()
stackstorm-sdi#22 · opened 17 hours ago by Pirmin Wenk · MS7 - End of Construction · development

Meeting W12
management#12 · opened 2 months ago by Yannick Zwicker · MS7 - End of Construction · discussion

UC03 - HA Setup
stackstorm-sdi#19 · opened 2 weeks ago by Pirmin Wenk · MS7 - End of Construction · infrastructure

UC03 - HA Setup
stackstorm-sdi#20 · opened 2 weeks ago by Pirmin Wenk · MS7 - End of Construction · development

Kickstart Predictable Interface Name
stackstorm-sdi#21 · opened 2 weeks ago by Yannick Zwicker · MS7 - End of Construction · development

UC03 - ESXi HA Setup
management#51 · opened 1 week ago by Yannick Zwicker · MS7 - End of Construction · analysis

UC01/UC02 - Frontend/Backend
management#49 · opened 2 weeks ago by Yannick Zwicker · MS7 - End of Construction · development

Transition

Sprint 8

Meeting W13
management#13 · opened 3 months ago by Yannick Zwicker MS8 - Projektabschluss discussion

Code Refactoring
stackstorm-sdi#25 · opened 3 days ago by Pirmin Wenk MS8 - Projektabschluss development

UC03 - ESXi HA Setup
management#51 · opened 4 weeks ago by Yannick Zwicker MS8 - Projektabschluss analysis

UC01/UC02 - Frontend/Backend
management#49 · opened 1 month ago by Yannick Zwicker MS8 - Projektabschluss documentation

Meeting W14
management#14 · opened 3 months ago by Yannick Zwicker MS8 - Projektabschluss discussion

Troubleshooting Check Point
stackstorm-sdi#26 · opened 13 minutes ago by Pirmin Wenk MS8 - Projektabschluss development

Troubleshooting Open vSwitch
stackstorm-sdi#27 · opened 13 minutes ago by Pirmin Wenk MS8 - Projektabschluss development

Dokumentation
management#52 · opened 3 days ago by Pirmin Wenk MS8 - Projektabschluss documentation

Backlog

Die optionalen Anforderungen wurden teilweise mit eingeplant aber nicht mehr vollständig umgesetzt.

UC03 - HA Setup
stackstorm-sdi#19 · opened 1 month ago by Pirmin Wenk infrastructure

UC03 - HA Setup
stackstorm-sdi#20 · opened 1 month ago by Pirmin Wenk development

Kickstart Predictable Interface Name
stackstorm-sdi#21 · opened 1 month ago by Yannick Zwicker development

UC03.7 - vCenter in Betrieb nehmen
infrastructure#1 · opened 1 month ago by Yannick Zwicker infrastructure

UC03.7 - ESXi Services deployen
stackstorm-sdi#17 · opened 1 month ago by Yannick Zwicker development

9.3 Projecttracing

9.3.1 Dokumentation

Die Dokumentation wurde mit dem Schriftsetztool LaTeXverfasst. Das Zusammenarbeiten konnte durch den Online-Editor Overleaf[4] bewerkstelligt werden.

9.3.2 Zeitauswertung

Wie bereits erwähnt, wurde die Arbeitspaketverwaltung direkt im GitLab vorgenommen. Dort wurden auch die Zeiten geschätzt, jedoch nicht die geleisteten Stunden eingetragen.

Der Grund liegt darin, dass wir die Zeitverwaltung nicht aus GitLab exportieren können und somit eine saubere Auswertung (auf Arbeitspakete, Personen und Sprints) nicht möglich wäre.

Wir haben uns deshalb entschieden, die Zeitverwaltung über ein selbst erstelltes Excel-File zu organisieren.

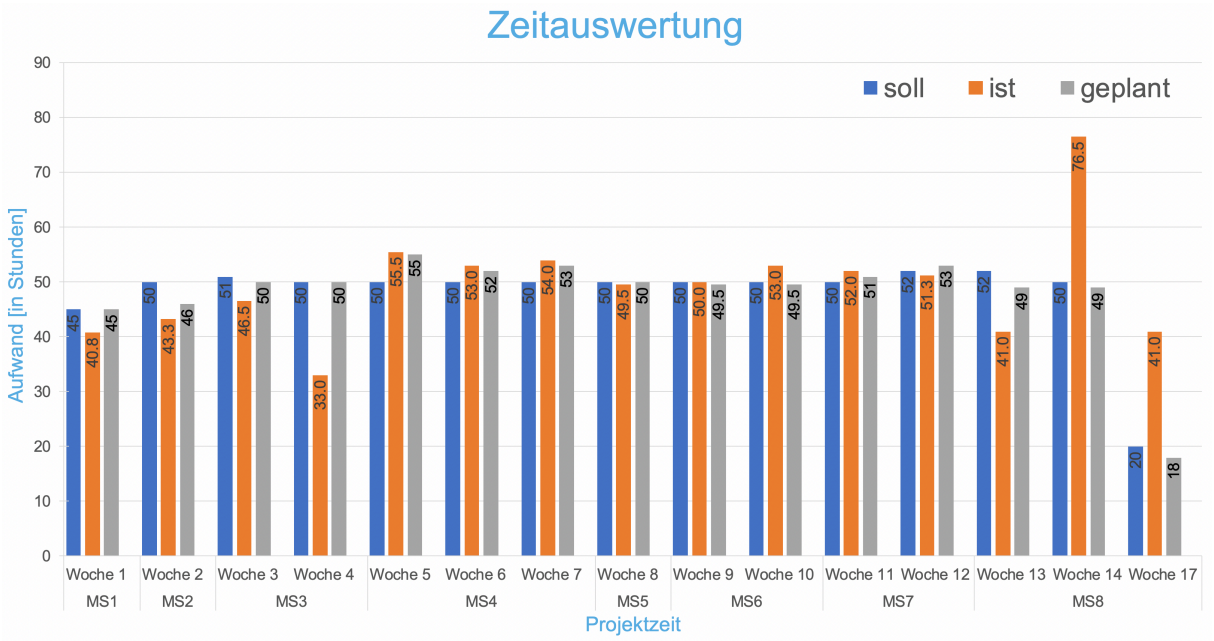


Abbildung 24: Zeitauswertung über die Wochen

[4] https://www.overleaf.com

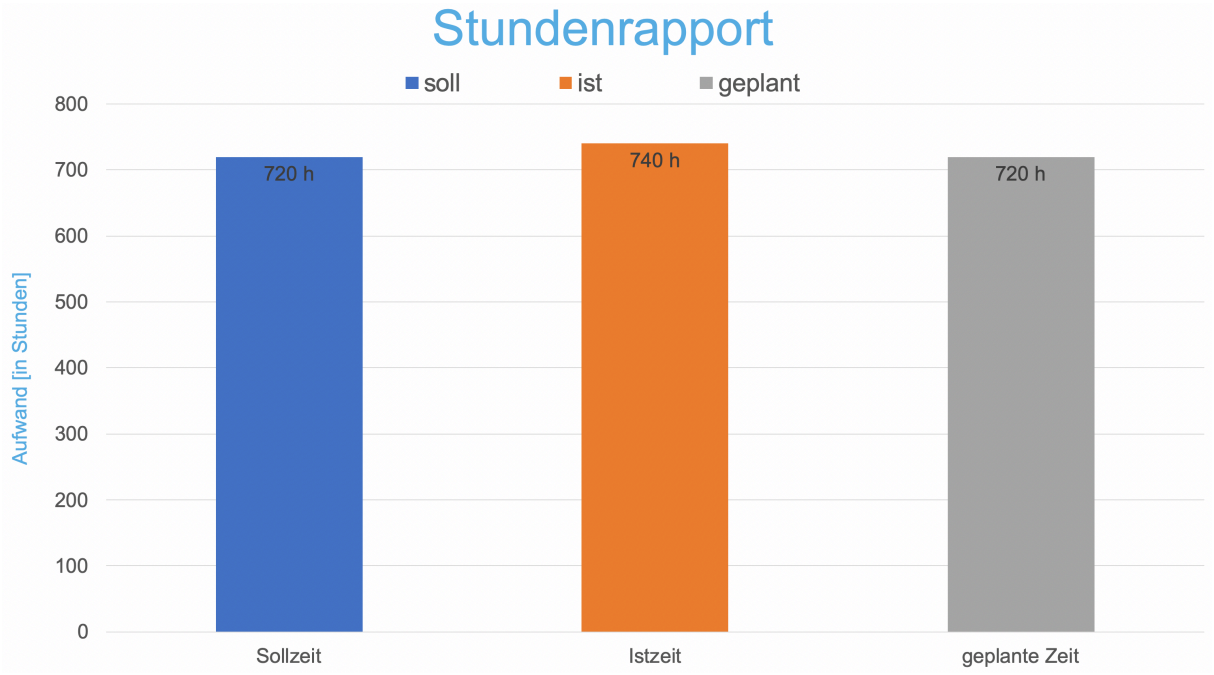


Abbildung 25: Stundenrapport Vergleich – Soll, Ist, Geplant

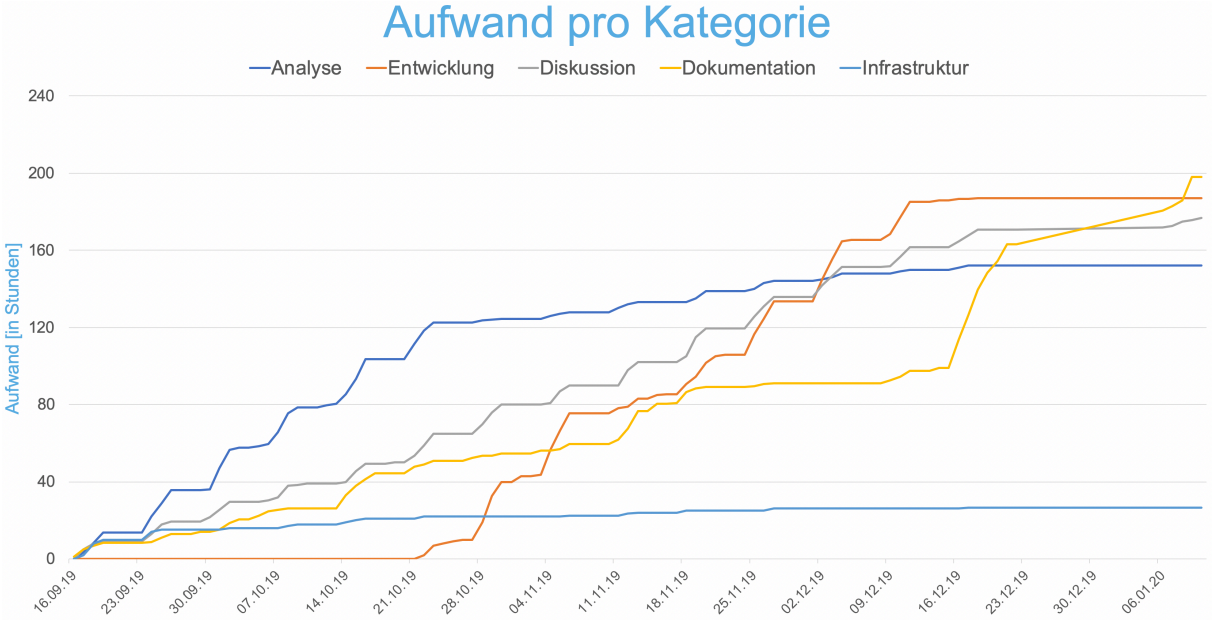


Abbildung 26: Summierter Aufwand pro Kategorie

9.3.3 Zielerreichung

Auch wenn die Meilensteine und die definierten Szenarios umgesetzt werden konnten, gibt es einige Punkte, in denen wir noch viele weitere Ideen miteinbringen wollten. Leider ist uns dies nicht komplett gelungen und wir mussten aus zeitlichen Gründen die eine oder andere Funktionalität etwas eingeschränkt umsetzen oder ganz weglassen.

Nach der Zwischenpräsentation wurde mit dem Industriepartner definiert, dass der Fokus mehr auf die Automatisierung des Workflows gelegt werden soll und deshalb kein eigenes UI gebaut wird (optional) und das Deployment von den VMware Services (optional) erst an zweiter Stelle kommen soll, sofern noch Zeit bleibt. Diese beiden Use Cases konnten nicht umgesetzt werden. Für die Benutzerverwaltung bzw. Login mit persönlichen Accounts wurde ein Konzept bereits ausgearbeitet.

Alle anderen als «Muss» definierten Use Cases konnten umgesetzt werden (siehe Use Cases, Seite 14).

Projekt in Zahlen

Anzahl Docker-Container 8
Anzahl StackStorm Workflows / Actions 12
Anzahl Ansible Playbooks 15
Anzahl Lines of YAML 2154
Anzahl Lines of Python Code 219

9.3.4 Fazit

Lediglich die Bachelorarbeit – ohne weitere Module nebenbei – durchzuführen ist eine neue, aber auch eine gute Erfahrung. Während der drei Tage, die wir an der Bachelorarbeit gearbeitet haben, gab es keine anderen Pendenzen und die Zeitplanung fiel etwas leichter. Die Zeit muss jedoch so effektiv wie möglich genutzt werden. Dies kommt der Arbeit im Geschäft sehr nahe und mindert auch die Stundenanteile in der Nacht und während dem Wochenende.

Wir haben mit einem Kick-off Meeting gut in die Arbeit gestartet. Wir wussten bereits da, dass wir es mit vielen neuen Technologien und Systemen zu tun bekommen. Beim Aufbau des Musterstandorts befassten wir uns – teilweise das erste Mal – mit diesen neuen Systemen. Nebenbei machten wir uns direkt schon Gedanken darüber, wie dieser manuelle Aufbau automatisiert werden könnte.

Die Komplexität der Automatisierung wurde zu Beginn etwas unterschätzt. Wenn man sich das erste Mal mit einem neuen System befasst, es noch gar nicht richtig kennt und bereits die Konfiguration automatisieren möchte, muss viel getestet, die aufgetauchten Fehler recherchiert und behoben werden. Dadurch kamen wir leider auch nicht so schnell vorwärts, wie wir uns das zu Beginn vorgestellt hatten.

Im Endeffekt war es eine sehr spannende, intensive und vor allem auch eine lehrreiche Arbeit.

9.4 Weitere Abklärungen

9.4.1 StackStorm

StackStorm-Ansible

Wir nutzen StackStorm für die Automatisierung. Um Ansible Playbooks laufen zu lassen, wird das StackStorm Pack stackstorm-ansible[5] verwendet.

Da die Ansible Playbooks bzw. gewisse Ansible Module – wie z.B. das virt[6] Module – noch weitere Abhängigkeiten für die Ausführung benötigen, klärten wir in der Community ab, was die Best Practices sind, um diese am sinnvollsten, möglichst automatisiert installieren zu können.

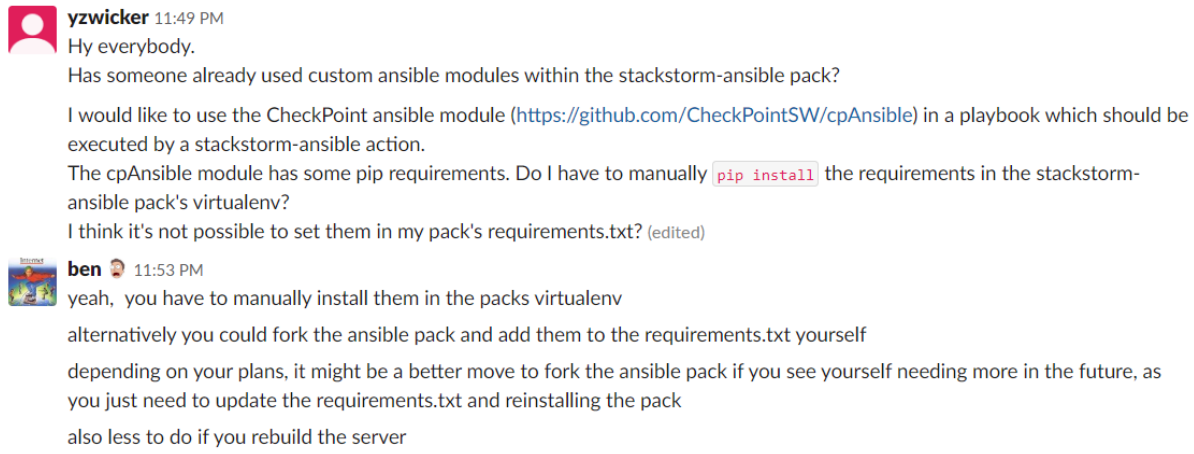


Abbildung 27: StackStorm-Ansible – NetworkToCode – Slack

Die Frage wurde im Slack Workspace «NetworkToCode»[7] im Channel «StackStorm» gestellt. Aus dieser Antwort ist ersichtlich, dass das originale StackStorm-Ansible Paket am besten geforkt wird und dort die nötigen Python Dependencies via requirements.txt mitgegeben werden. Dies haben wir dann sogleich umgesetzt https://github.com/zwiy/stackstorm-ansible.

Das alte Check Point Ansible Modul – nach welchem gefragt wurde – verwenden wir nicht mehr, da es seit Ansible 2.9 im Core ist bzw. wir es via Ansible Galaxy installieren. Jedoch wird diese Lösung für andere Module noch so benötigt.

[5] https://github.com/StackStorm-Exchange/stackstorm-ansible
[6] https://docs.ansible.com/ansible/latest/modules/virt_module.html
[7] http://slack.networktocode.com

9.4.2 Check Point

Check Point Ansible Mgmt Collection

Über das Ansible gibt es eine neue Möglichkeit, die API des Check Point Managements anzusteuern. Die Library dazu findet man unter GitHub[8], bzw. direkt bei Ansible unter https://galaxy.ansible.com/check_point/mgmt.

Leider haben wir einen Fehler im Code entdeckt, durch welchen unser Check Point Deployment nicht vollständig durchlaufen konnte. Wir eröffneten einen Issue[9] auf GitHub am 21.11.2019 und es wurde schnell reagiert:

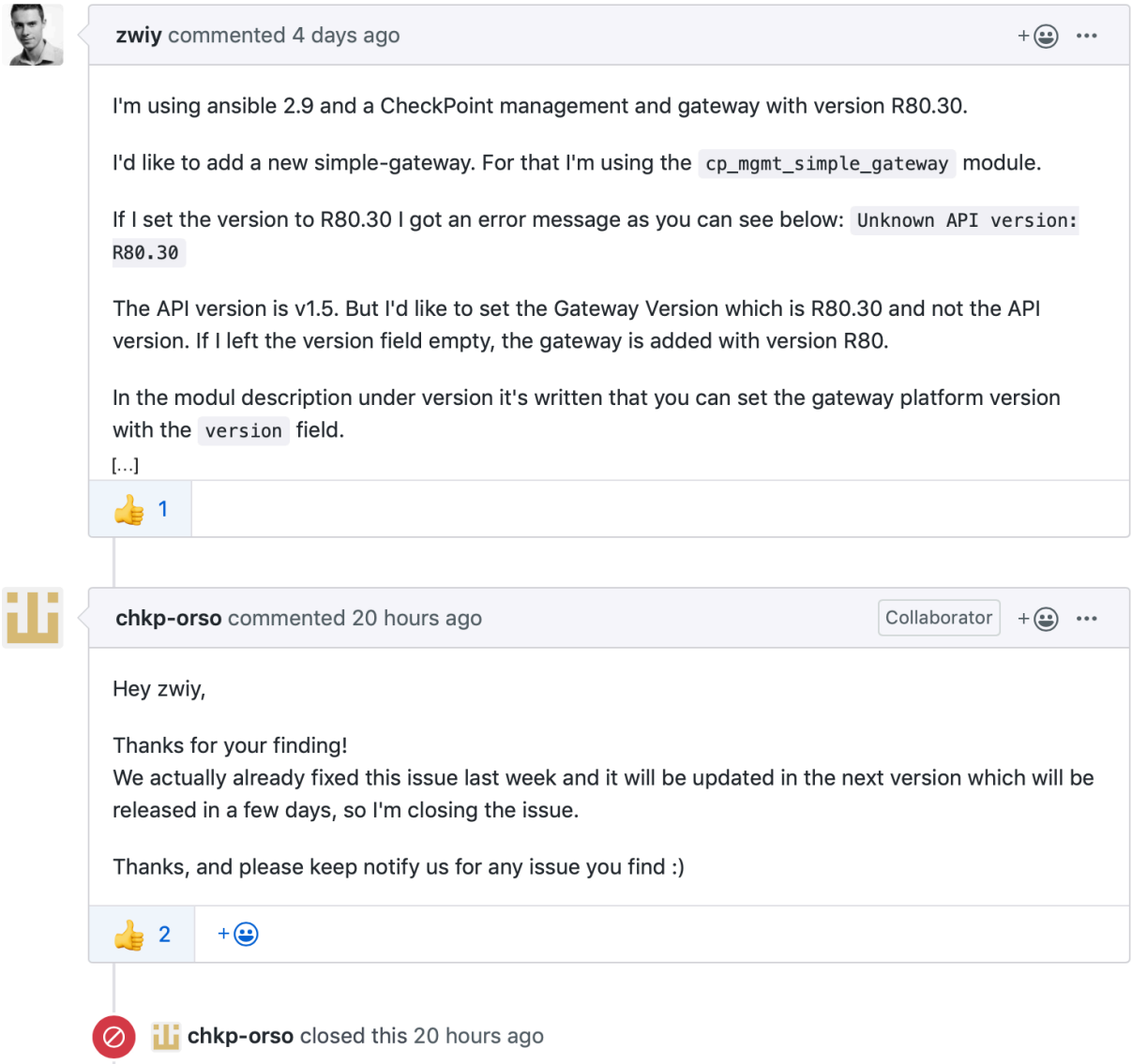


Abbildung 28: Check Point Ansible-Mgmt-Collection – GitHub – Issues

Das Update ist am 8.12.2019, gut zwei Wochen nach dem Eröffnen des Issues, erschienen und hat unser Problem gelöst.

[8] https://github.com/CheckPointSW/CheckPointAnsibleMgmtCollection
[9] https://github.com/CheckPointSW/CheckPointAnsibleMgmtCollection/issues/1

9.5 Kickstart Files

CentOS für PXE

Diese unterscheidet sich nur wenig von der nachfolgenden CentOS für ISO-Installation:

- Anstatt «cdrom» wird die FTP-Source angegeben:
  `url-url="ftp://<ftp-server-ip>/pub/centos_iso_files"`

CentOS für eine ISO-Installation

Ein Beispiel der Kickstart-Datei befindet sich in der Abgabe.

ESXi

Diese ist für PXE und eine ISO-Installation gleich.

Ein Beispiel der Kickstart-Datei befindet sich in der Abgabe.

9.6 Anleitungen

9.6.1 Installation der Repositories

Infrastruktur Setup

Im Infrastruktur-Repository in der Abgabe liegen die Vorlagen für die Inbetriebnahme der nötigen Docker-Container. Dazu gehören:

- Traefik
- MinIO
- ISO-Packer

StackStorm kann nach Anleitung des StackStorm Docker Repository[10] aufgesetzt werden.

StackStorm SDI Pack Installation und Konfiguration

Die Anleitung zur Installation und Konfiguration befindet sich im `stackstorm-sdi` Repository im `README.md` in der Abgabe.

StackStorm Ansible Pack Installation und Konfiguration

Die Anleitung zur Installation und Konfiguration befindet sich im `stackstorm-ansible` Repository im `README.md` [45].

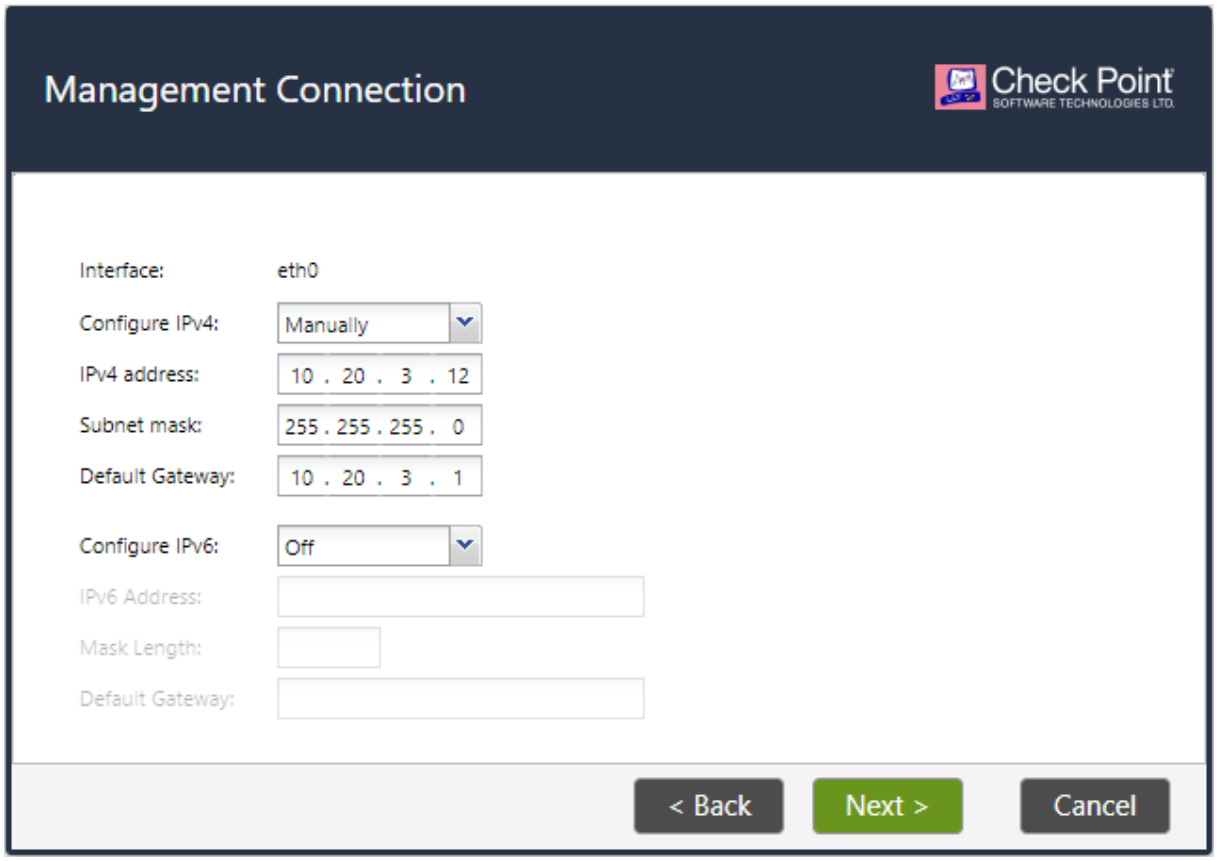
[10] https://github.com/stackstorm/st2-docker

9.6.2 Check Point Management Installation

Als Installationsquelle wird die ISO-Datei für das Check Point Management verwendet.

Nach dem Setup wird auf das Webinterface zugegriffen und dort der First Time Configuration Wizard [23] durchgeführt:

1. «Continue with R80.30 configuration»
2. IP-Konfiguration



3. Geräteinformationen entsprechend ausfüllen
4. Zeit oder NTP-Server festlegen
5. «Security Management»
6. «Define Security Management as: Primary» und Haken setzen
7. «Use Gaia administrator: admin»
8. «Any IP Address»
9. Finish

9.6.3 ISR 4461 einrichten

Grundkonfiguration der UCS

Damit auf das CIMC zugegriffen werden kann, muss auf dem ISR die nötige Grundkonfiguration temporär eingerichtet werden [10]. Dabei ist es wichtig, dass die IP-Adressen so gewählt werden, dass eine Verbindung zustande kommen kann.

NB: Grundsätzlich kann diese Einstellung auch erst dann gemacht werden, wenn der Router bereits am finalen Ort steht. Dies lässt sich vereinfacht bewerkstelligen, indem der Anfang des Workflows inkl. Konfiguration des ISR 4461 separat angestossen wird, vorausgesetzt, der Router besitzt bereits eine IP-Konnektivität.

Beispielkonfiguration für den ISR 4461:

```
interface ucse2/0/0
 no ip address
 negotiation auto
 switchport mode trunk
 no mop enabled
 no mop sysid
 service instance 2 ethernet
  encapsulation untagged
  bridge-domain 2

interface ucse3/0/0
 no ip address
 negotiation auto
 switchport mode trunk
 no mop enabled
 no mop sysid
 service instance 2 ethernet
  encapsulation untagged
  bridge-domain 2

ucse subslot 2/0
 imc access-port shared-lom console
 imc ip address 10.20.47.40 255.255.255.0 default-gateway 10.20.47.1

ucse subslot 3/0
 imc access-port shared-lom console
 imc ip address 10.20.47.50 255.255.255.0 default-gateway 10.20.47.1
```

Nun funktioniert die Verbindung zu den UCS-Blades. Folgende Schritte müssen für alle UCS-Blades separat gemacht werden:

1. Webseite des UCS aufrufen (über die oben gesetzte IP-Adresse)
2. Passwort für den admin-User ändern
3. Host Power einschalten
4. Boot Order festlegen -> im Folgenden beschrieben
5. Festplatten leeren (initialisieren) -> im Folgenden beschrieben

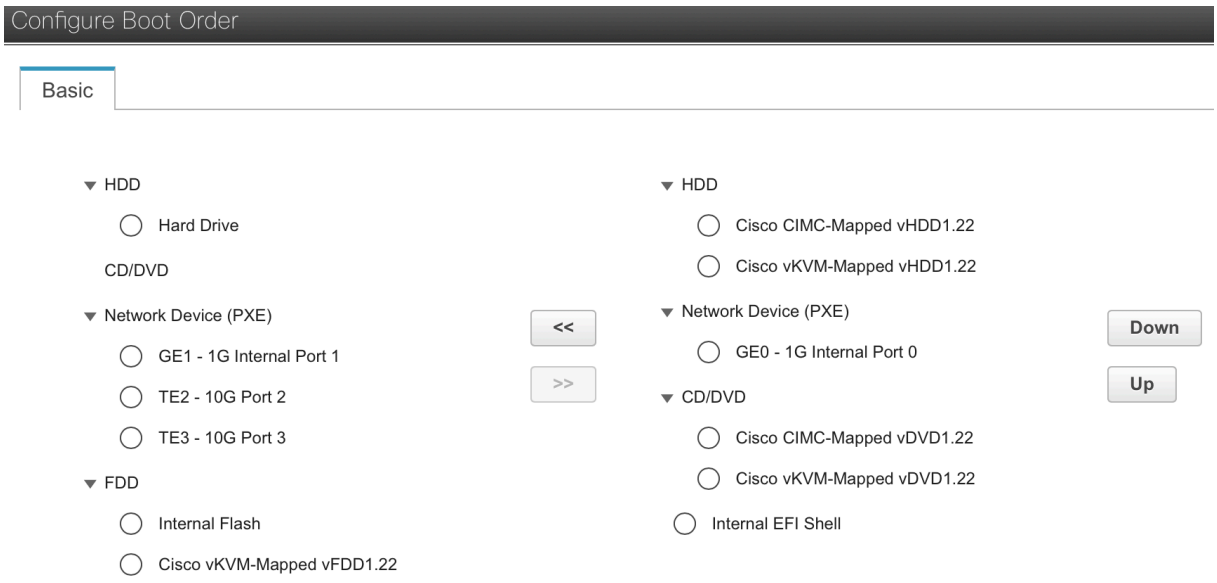
Zum Schluss sollte diese temporäre Konfiguration wieder entfernt werden:

```
no interface ucse2/0/0
no interface ucse3/0/0
no ucse subslot 2/0
no ucse subslot 3/0
```

Bootreihenfolge festlegen

Via CIMC sieht das folgendermassen aus:

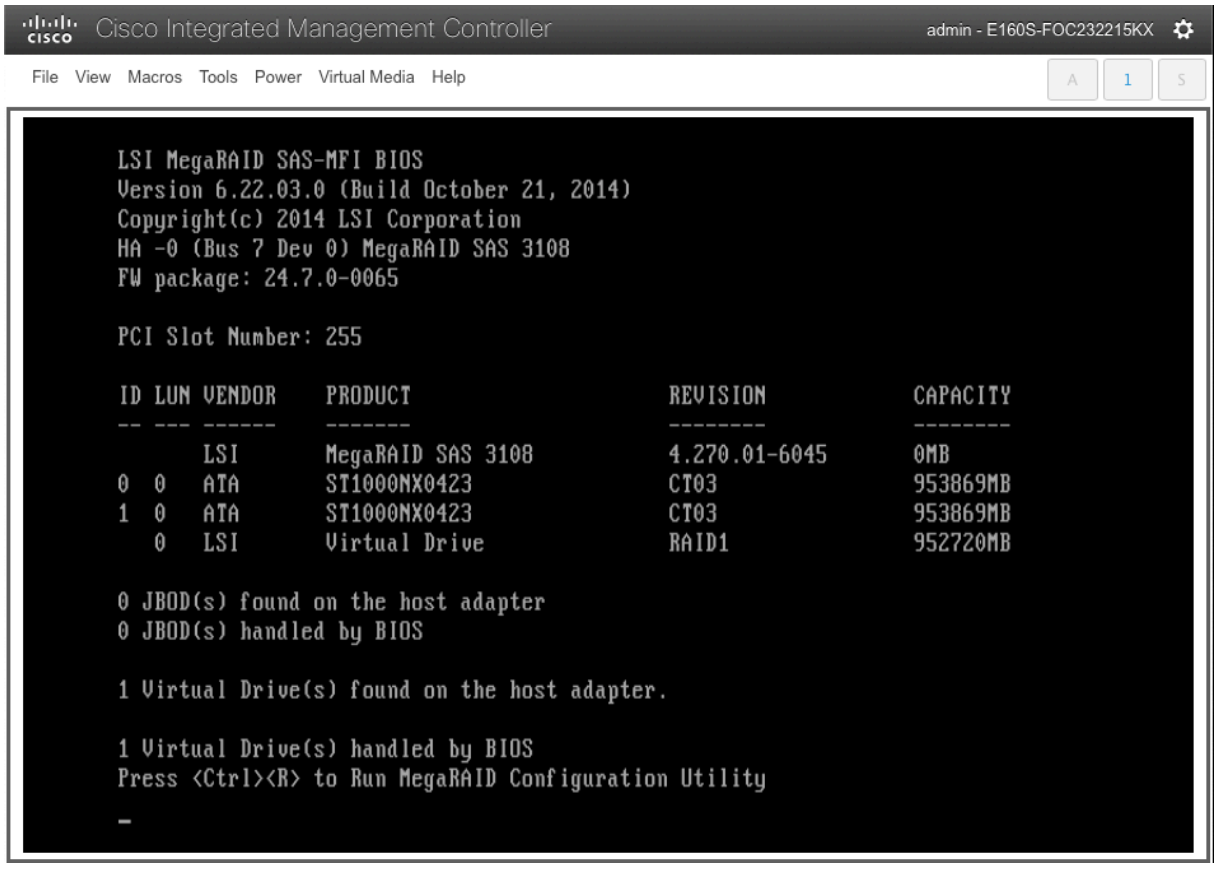
Unter Compute > BIOS > Configure Boot Order



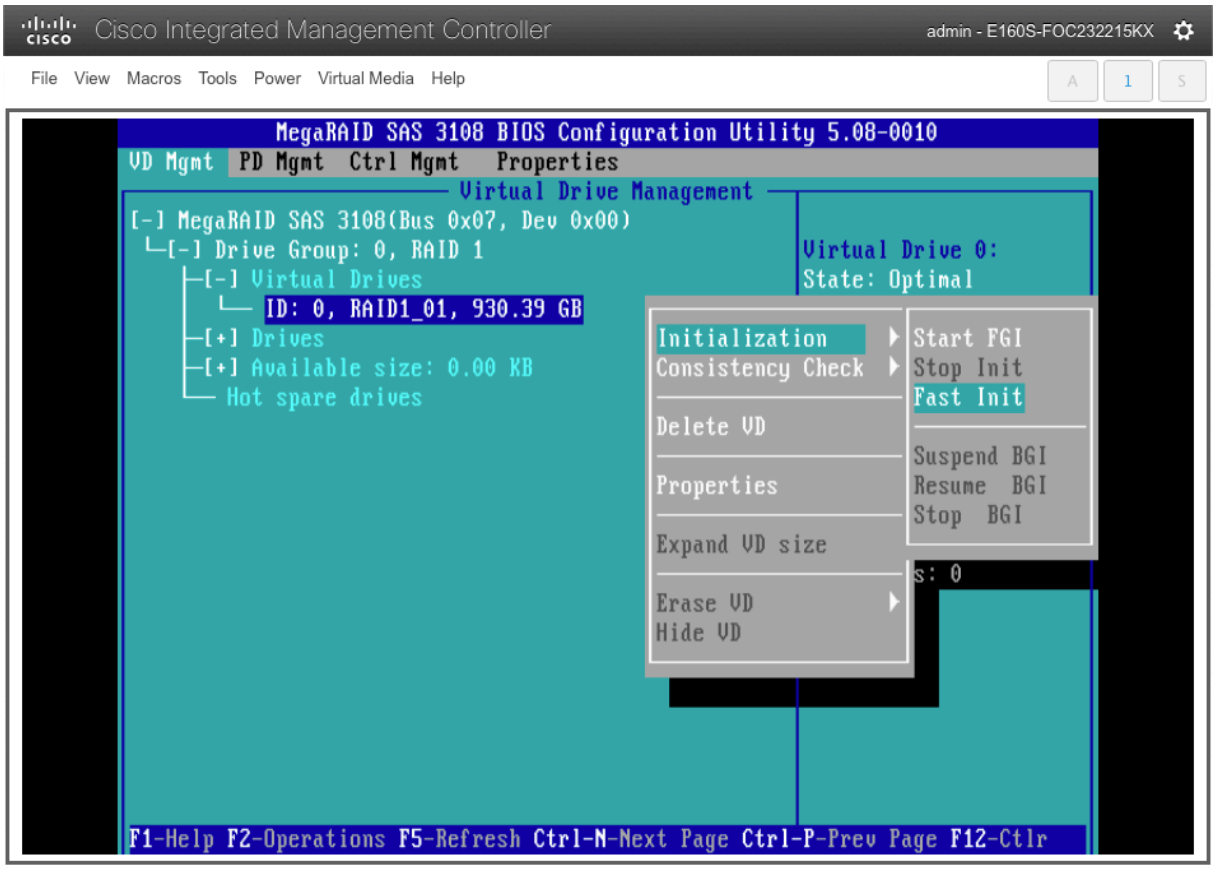
Festplatte initialisieren

Beim Startvorgang des UCS muss eine Verbindung auf die Konsole hergestellt werden. Dies kann im CIMC über Launch KVM > HTML Based KVM erfolgen.

Ca. 45s nach dem Power ON erscheint die RAID-Meldung. Hier Ctrl + R gleichzeitig drücken:



Dann kann über F2 → ↓ Enter «Fast Init» ausgewählt werden:



Das dauert bis zu 30s.

9.6.4 Infoblox vorbereiten

Network View

Unter Administration > Network View in der Infoblox Web-Administration kann einmalig eine neue View erstellt oder eine vorhandene verwendet werden. Sie muss im StackStorm für den Reservations Workflow (z.B. `infoblox_reservation_xs.yaml`) angegeben werden. In Zukunft sollten diese Informationen auch in einer Config Datei ausgelagert sein.

Extensible Attribute mit Vererbung

Als erstes schalten wir dann die Vererbung der Attribute richtig ein. Das machen wir unter Administration > Extensible attributes

Name	Type	Comment	Required	Restricted to Objects	Inheritance Enabled
sdi_vlan_type	String		No	IPv4 Network,IPv6 Network,Host	Yes
sdi_vlan	Integer		No	IPv4 Network,IPv6 Network,Host	Yes
sdi_site	String		No		Yes
sdi_ansible_host	String	If set then always true	No		No

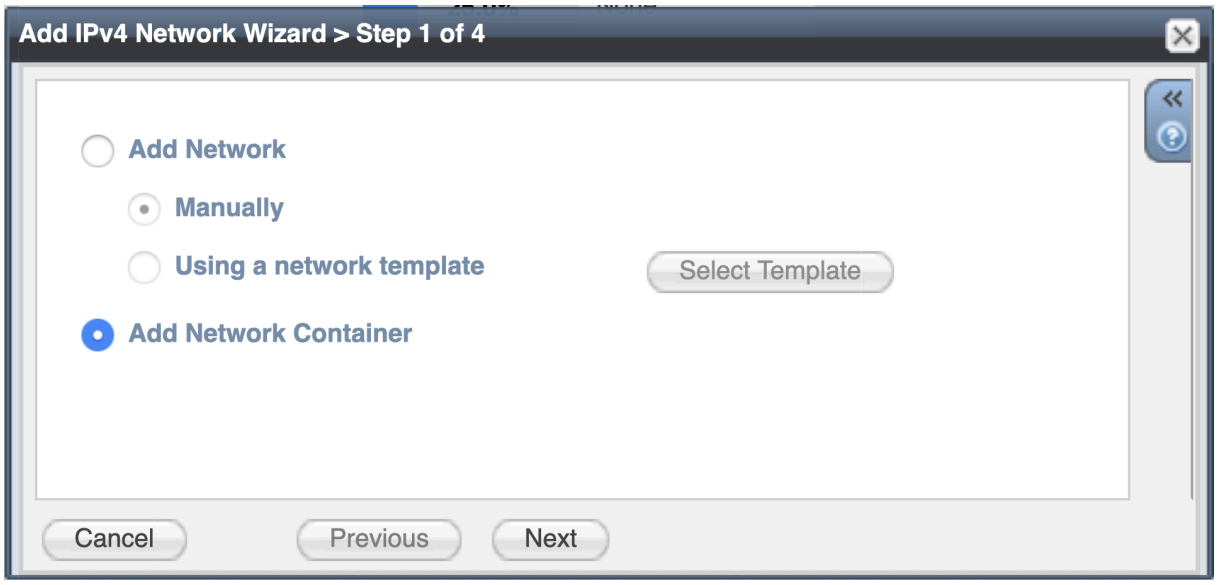
Pro Standort

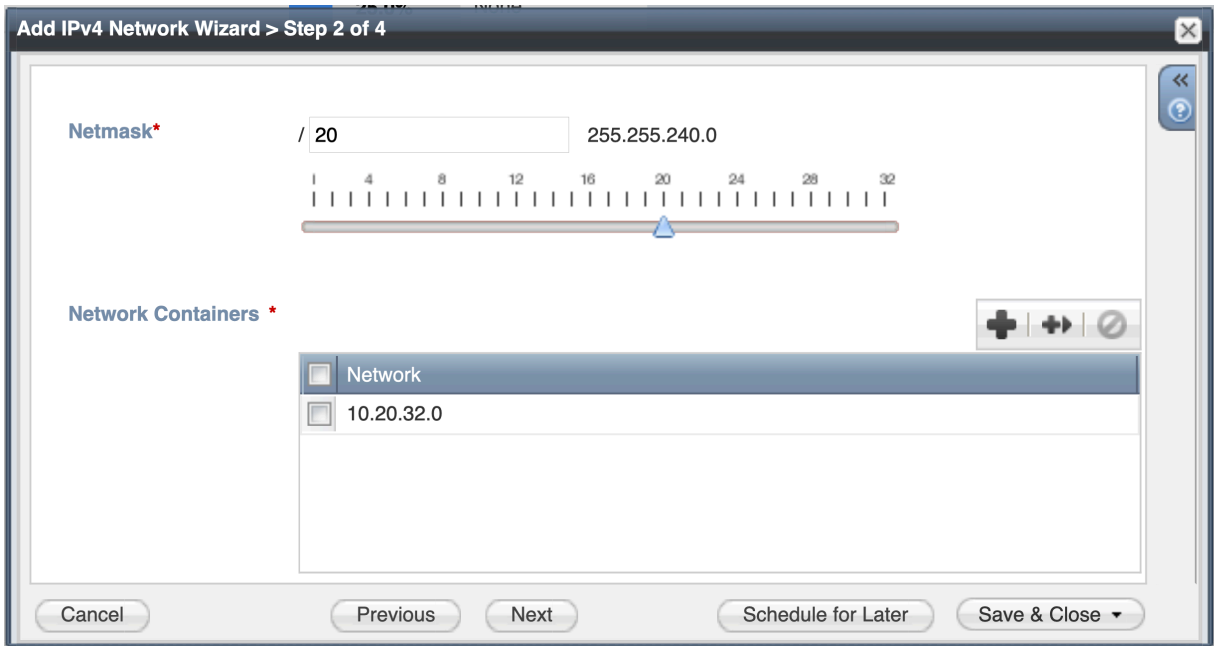
Auf dem Infoblox muss für jeden Standort eine gewisse Grundkonfiguration gemacht werden. Folgende Assets müssen gepflegt sein:

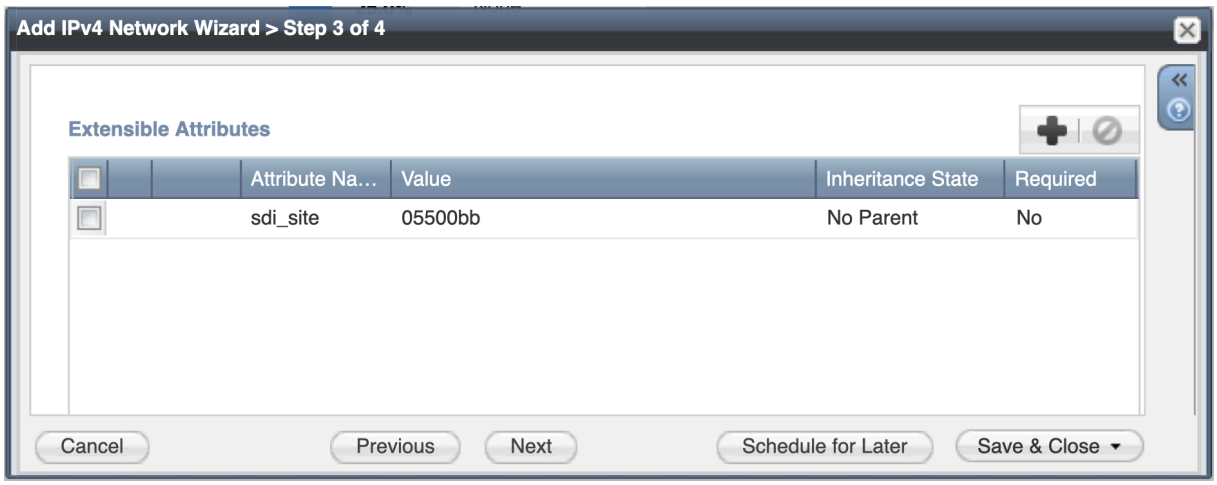
- Netzwerkcontainer (zum Beispiel /20-Netz)
- Netzwerke (zum Beispiel /24-Netze) für `client`, `dc_transit` und `network_mgmt`
- CE Router, Client1 Netz: `05500bb-1ace-01-gi0-0.1110`
- CE Router, Client2 Netz: `05500bb-1ace-01-gi0-0.1111`
- CE Router, dc_transit Netz: `05500bb-1ace-01-gi0-0.3`
- CE Router, network_mgmt Netz: `05500bb-1ace-01-gi0-0.2`
- Data Center (DC) Router (ISR 4461), network_mgmt Netz: `05500bb-1adc-01-bdi2`

Network Container anlegen

Oben links im Data Management kann über das Dropdown in die Network-View gewechselt werden. Darin werden die Netzwerk-Container angezeigt. Über + erstellen wir einen Container neuen und setzen das Extensible Attribute `sdi_site` auf den Site Namen:

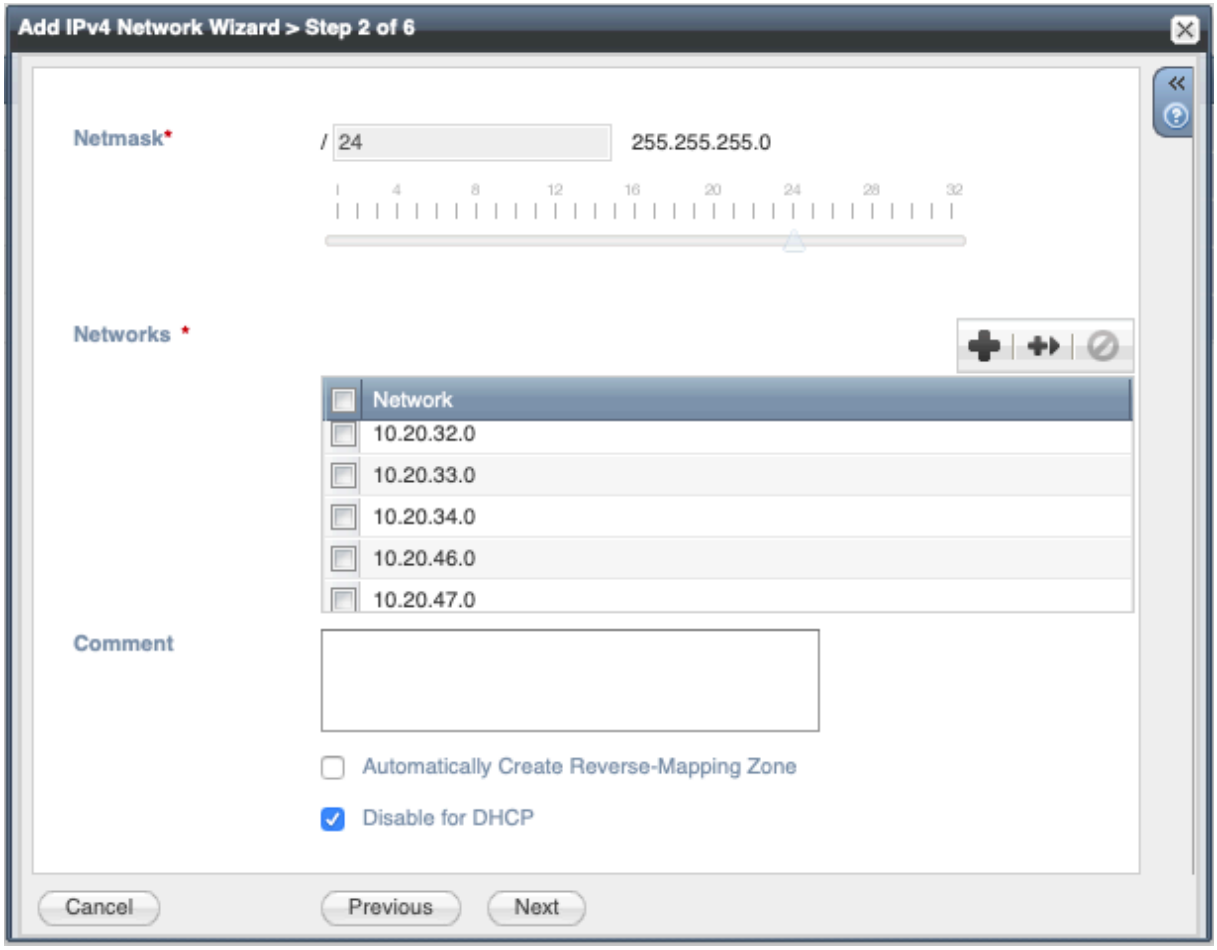

Add IPv4 Network Wizard > Step 1 of 4
Add Network
Manually
Using a network template
Select Template
Add Network Container
Cancel
Previous
Next


Add IPv4 Network Wizard > Step 2 of 4
Netmask*
/ 20
255.255.240.0
1 4 8 12 16 20 24 28 32
Network Containers *
Network
10.20.32.0
Cancel
Previous
Next
Schedule for Later
Save & Close


Add IPv4 Network Wizard > Step 3 of 4
Extensible Attributes
Attribute Na...
Value
Inheritance State
Required
sdi_site
05500bb
No Parent
No
Cancel
Previous
Next
Schedule for Later
Save & Close

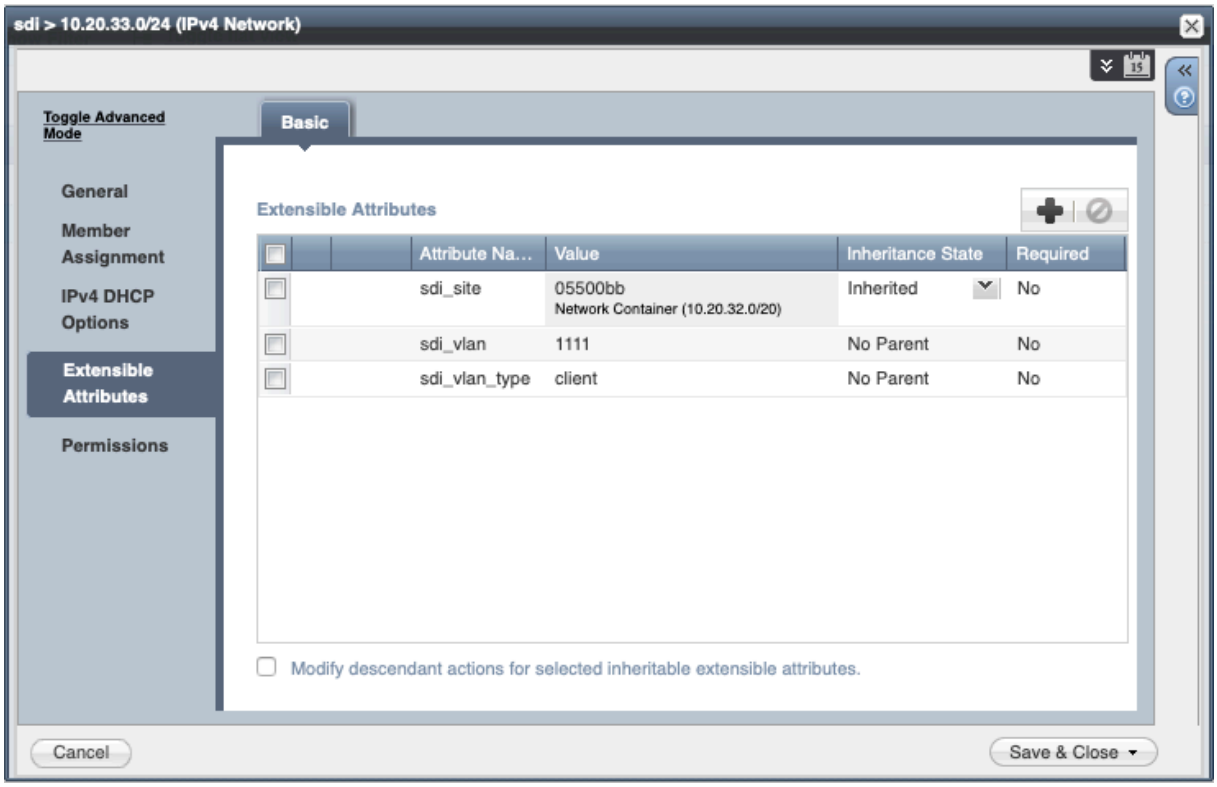
Netzwerke anlegen

Im Container drin können wir über + die nötigen Netzwerke hinzufügen. Diese erhalten die Extensible Attributes vom Container vererbt:



Zusätzlich müssen auf jedem Netzwerk die folgenden Extensible Attributes gepflegt werden:

- `sdi_vlan` Der VLAN Tag des Netzwerks
- `sdi_vlan_type` Der Typ, wie er im Inventory-Script steht (siehe Inventory, Seite 49)



Hosts anlegen

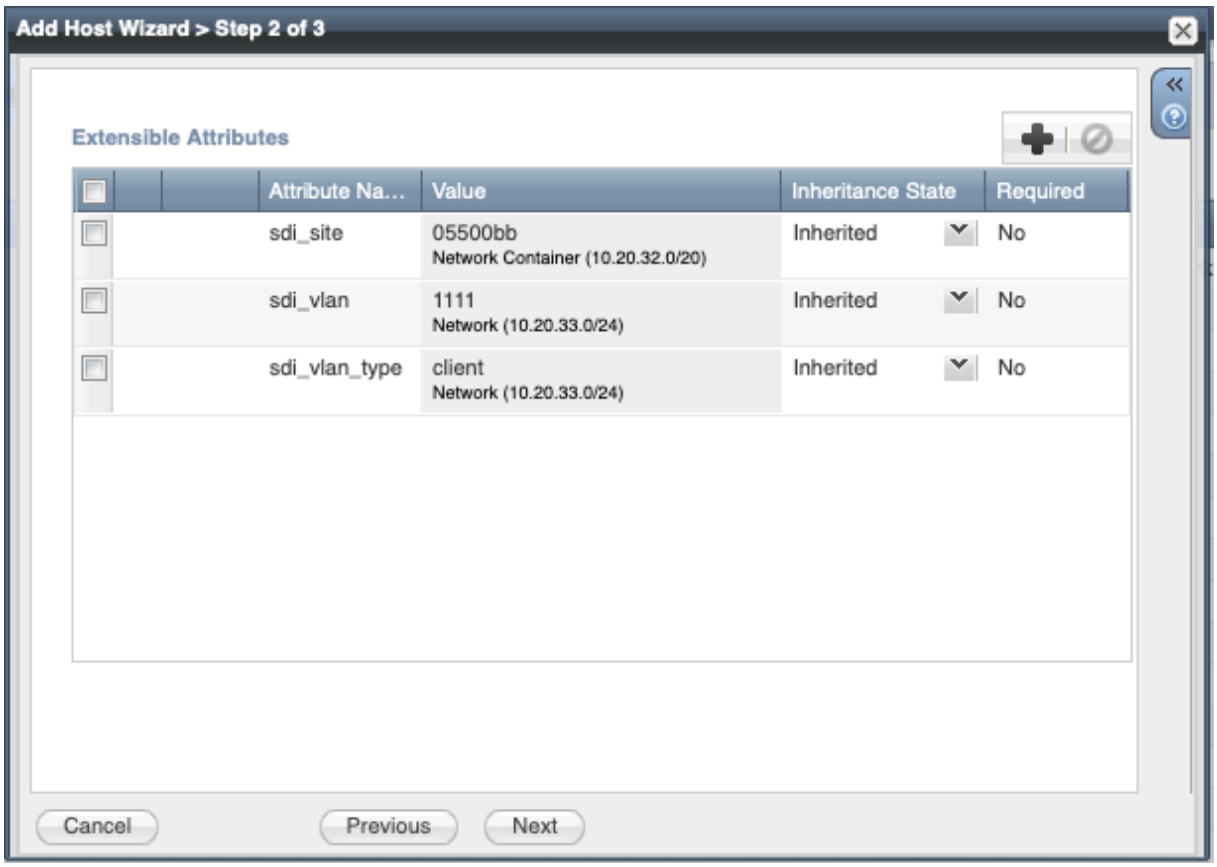
Ein Host mit Automatisierung muss zwingend das Extensible Attribute `sdi_ansible_host` aufweisen. Das Gerät erhält dann das Attribut `ansible_host` im Inventory, welches für die Automatisierung benötigt wird. Wenn ein Gerät, wie z.B. die Firewall, mehrere Interfaces besitzt, darf das Extensible Attribute nur auf einem Host Record gesetzt werden. Es muss das Management Interface sein, oder das Interface, über welches sich das Automatisierungstool einloggen kann.

Dies ist zum Beispiel auch für den DC Router gewollt.

Host ohne Automatisierung

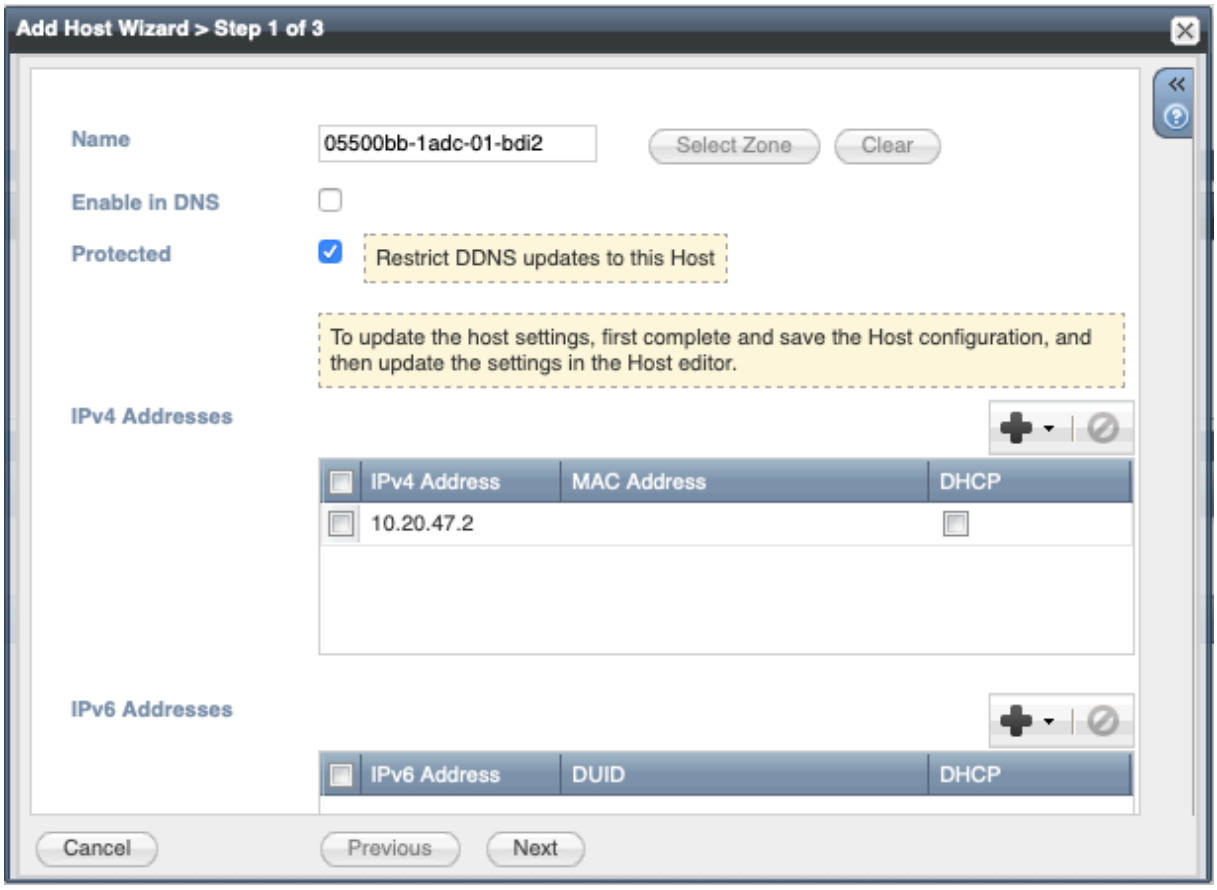
`sdi_ansible_host` wird zum Beispiel für den CE Router **nicht** benötigt, der nicht über das Automatisierungstool angesprochen wird.

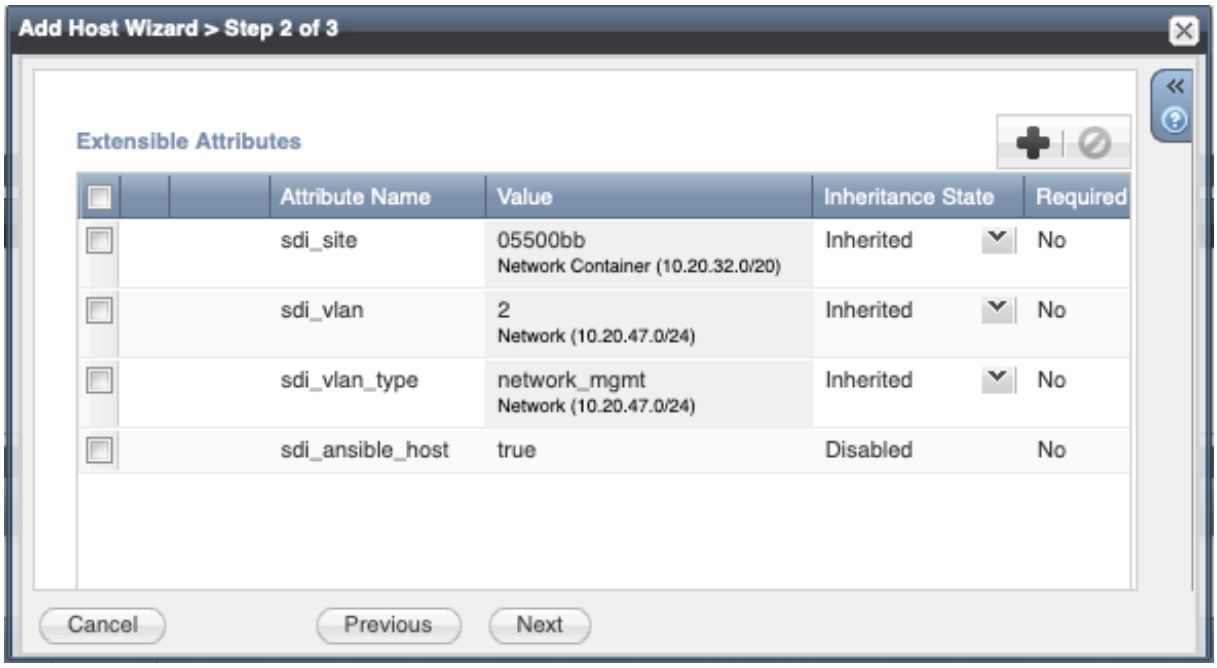
Hier ist keine Anpassung erforderlich:



Host mit Automatisierung

Dies wird zum Beispiel für den DC Router (ISR 4461) benötigt, da dieser über die Automatisierung konfiguriert wird.





9.6.5 PXE

PXE läuft in mehreren Schritten ab. Als Vorbedingung muss die Boot-Reihenfolge korrekt sein und die Festplattenpartitionen komplett geleert, damit automatisch von dem Netzwerk gestartet wird. Leider ist es uns nicht möglich, die Boot-Reihenfolge über unsere Automatisierung zu ändern.

1. Festplatte (muss komplett leer sein -> initialized)
2. Netzwerk
3. Optional CD/DVD
4. Optional Cisco EFI Boot (Cisco UCS EFI-Verwaltungstool)

Weitere Details siehe ISR 4461 einrichten, Seite 83.

Danach wird vom Netzwerk gestartet, wobei vom DHCP-Server, welcher auf dem ISR konfiguriert wurde, neben der IP-Adressvergabe die nötigen Optionen 66 (IP-Adresse des PXE-Servers) und 67 (Bootfile-Name auf dem PXE-Server, z.B. pxelinux.0) gezogen werden. Somit sind alle nötigen Informationen für das Booten vorhanden.

Konfiguration DHCP-Server auf dem Cisco ISR:

```
ip dhcp excluded-address vrf outside 10.20.4.33 10.20.4.34 10.20.4.35

ip dhcp pool pxe
 network 10.20.4.32 255.255.255.248
 bootfile pxelinux.0
 default-router 10.20.4.33
 option 66 ip 10.18.2.25
 option 67 ascii pxelinux.0
```

Auf dem PXE-Server muss entschieden werden, welches Image gezogen werden soll (ESXi oder CentOS). Dies ist über mehrere verschiedene Varianten in folgender Reihenfolge möglich:

MAC-Adresse des PXE-Clients Die MAC-Adresse (geführt von «01-») ist insofern nicht brauchbar, weil es jene virtuelle Adresse des UCS ist, welche der internen Kommunikation zwischen UCS und ISR dient. Es ist das Interface GE0. Diese MAC-Adresse lässt sich über den ISR nicht automatisiert auslesen und müsste daher manuell gepflegt werden.

IP-Adresse in hexadezimaler Schreibweise Dies ist eine gute Möglichkeit für uns. Wenn man weiss, wie gross der DHCP-Bereich ist, kann diese Adresse in hexadezimal umgerechnet werden. Beispiel: Die IP-Adresse 10.20.4.36 wird zu 0x0A140424. Es wird fortlaufend von rechts eine Ziffer abgeschnitten und somit der IP-Range erhöht. Dadurch ist die spezifischere Konfiguration zuerst.
Hier zur Veranschaulichung ein Beispiel, bei dem die «default»-Config gezogen wurde:

```
sent /var/lib/tftpboot/pxelinux.0 to 10.20.4.36
file /var/lib/tftpboot/pxelinux.cfg/01-00-5d-73-05-24-c4 not found
file /var/lib/tftpboot/pxelinux.cfg/0A140424 not found
file /var/lib/tftpboot/pxelinux.cfg/0A14042 not found
[...]
file /var/lib/tftpboot/pxelinux.cfg/0A not found
file /var/lib/tftpboot/pxelinux.cfg/0 not found
sent /var/lib/tftpboot/pxelinux.cfg/default to 10.20.4.36
```

default Trifft von oben nichts zu, wird die Konfiguration namens «default» verwendet. Darin kann ebenfalls ein «DEFAULT»-Eintrag vorhanden sein. Ist dies nicht der Fall, braucht es vom User zwingend eine Interaktion.

Eine solche Konfigurationsdatei (0A140424 im tftpboot-Verzeichnis) sieht z.B. so aus:

```
DEFAULT vmware
  SAY Now booting in the PXE Boot MENU...
LABEL vmware
  MENU LABEL vmware
  KERNEL /vmware/mboot.c32
  APPEND -c /vmware/boot.cfg
[...]
```

Defaultmässig wird hier das Image mit dem Label «vmware» gezogen. Beim Label werden das Boot-File und die Konfigurationsdatei mit den Boot-Optionen mitgegeben. Danach ist das Menu-Layout und das Timeout der Benutzerwahl definiert.

Für CentOS sieht das dann so aus:

```
DEFAULT centos
  SAY Now booting in the PXE Boot MENU...
LABEL centos
  MENU LABEL centos
  KERNEL /centos/vmlinuz
  APPEND initrd=/centos/initrd.img inst.ks=https://<https-server-ip>/anaconda-ks.cfg
      ↪ inst.repo=ftp://<ftp-server-ip>/pub ip=dhcp
[...]
```

Neben der Kernel-Datei, die es für den Boot-Vorgang benötigt, werden hier unter Anderem das Kickstart-File und das FTP-Repository, wo die CentOS-Dateien liegen, angegeben.

Im Folgenden ist dieser Ablauf nochmals grafisch beschrieben, am Beispiel von VMware:

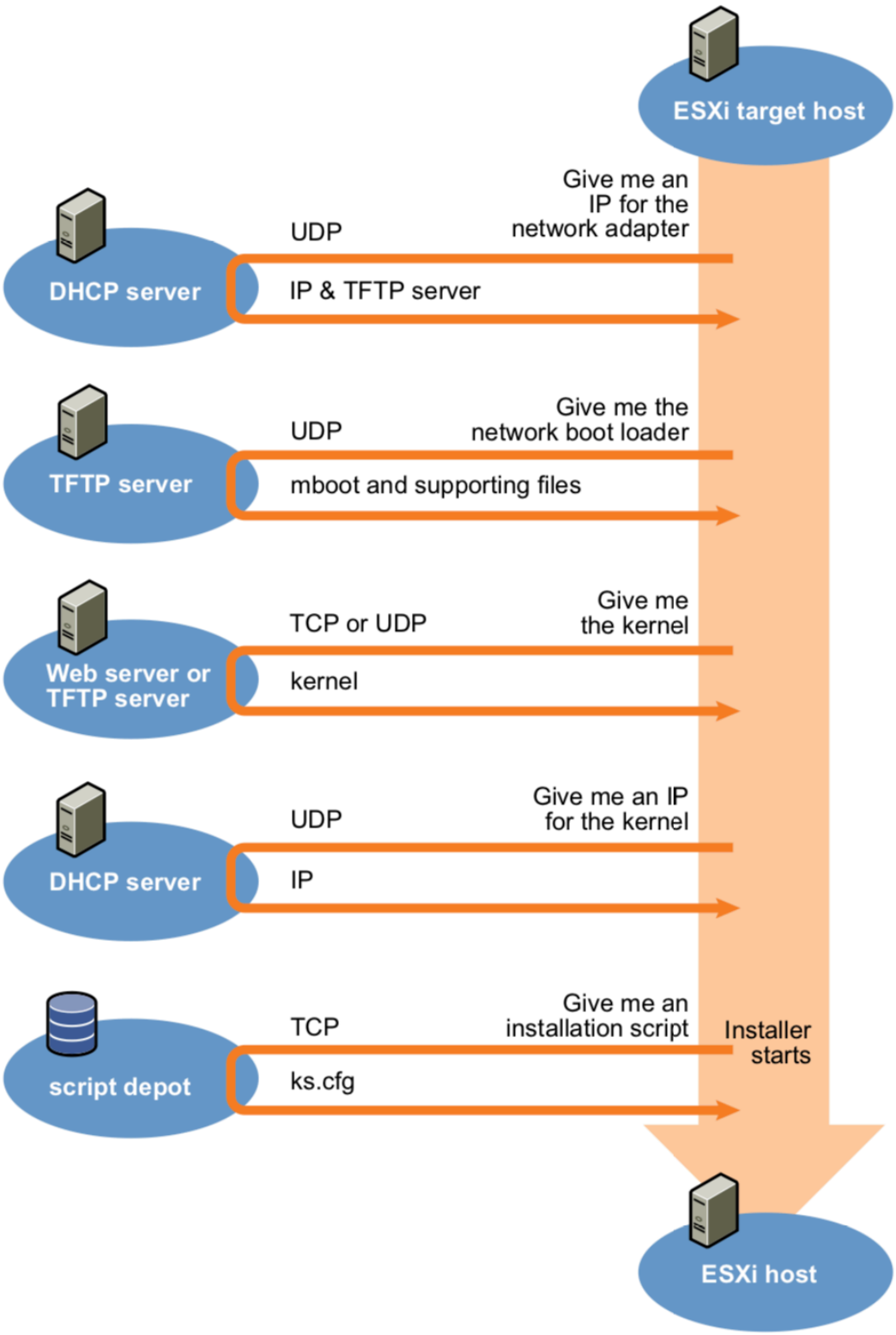


Abbildung 29: PXE Ablauf VMware [53]

Als PXE-Server wurde ein Docker-Container erstellt. Hier sind die Details ersichtlich: Dockerfile

```
FROM alpine:3.10.2

# Install the necessary packages
RUN apk add --update dnsmasq wget && rm -rf /var/cache/apk/*

ENV SYSLINUX_VERSION 3.86
ENV TEMP_SYSLINUX_PATH /tmp/syslinux-"$SYSLINUX_VERSION"
WORKDIR /tmp
RUN \
  mkdir -p "$TEMP_SYSLINUX_PATH" \
  && wget -q https://www.kernel.org/pub/linux/utils/boot/syslinux/3.xx/syslinux-"\
      ↪ $SYSLINUX_VERSION".tar.gz \
  && tar -xzf syslinux-"$SYSLINUX_VERSION".tar.gz \
  && mkdir -p /var/lib/tftpboot \
  && cp "$TEMP_SYSLINUX_PATH"/core/pxelinux.0 /var/lib/tftpboot/ \
  && cp "$TEMP_SYSLINUX_PATH"/com32/menu/menu.c32 /var/lib/tftpboot/ \
  && rm -rf "$TEMP_SYSLINUX_PATH" \
  && rm /tmp/syslinux-"$SYSLINUX_VERSION".tar.gz

# Configure DNSMASQ
COPY etc/ /etc
```

Im Dockerfile werden die nötigen Pakete in ein Alpine-Linux installiert. Das benötigte Syslinux muss für VMware in der Version 3.86 installiert werden [48], ansonsten funktioniert das Setup nicht.

docker-compose.yaml

```
version: '3.7'

services:

  pxe:
    build:
      context: ./
      dockerfile: Dockerfile
    network_mode: host
    volumes:
      - ./tftpboot/pxelinux.cfg:/var/lib/tftpboot/pxelinux.cfg
      - ./tftpboot/centos:/var/lib/tftpboot/centos
      - ./tftpboot/vmware:/var/lib/tftpboot/vmware
```

`network_mode:host` sagt, dass alle Ports gemappt werden.

Als Volumes werden die Konfigurationsdateien von oben gemappt und zusätzlich die Dateien für die Betriebssysteme.

Weil in CentOS das Betriebssystem nicht via CD eingelegt wird, müssen die nötigen Ressourcen von einem herkömmlichen FTP-Server zur Verfügung gestellt werden. Im Kickstart-file wird auf diese Source verwiesen (siehe Kickstart Files, Seite 81).

9.7 Glossar

API Application Programming Interface. 10, 30

BDI Bridge Domain Interface. 34, 46, 60
BPMN Business Process Model and Notation. 28
Bridge Domain Eine Bridge Domain repräsentiert eine Layer 2 Broadcast Domain. 34

CE Customer Edge. 32, 34, 44, 46, 86, 90
CentOS Betriebssystem von RedHat. 37
Checkpoint FTW Checkpoint First Time Configuration Wizard. 59, 60
CIMC Cisco Integrated Management Controller zur Administration und Fernwartung eines Cisco UCS. 37, 47, 48, 55, 63, 83

DC Data Center. 86, 89, 91
DHCP Dynamic Host Configuration Protocol. 10, 12
DNS Domain Name System. 10, 12
Docker Docker ist eine containerbasierte Virtualisierungslösung, die zur Isolierung von Anwendungen dient und dabei einen einzigen Betriebssystem-Kernel teilt. 11, 22, 23, 37

FHRP First Hop Redundancy Protocol. 32
FUB Führungsunterstützungsbasis. 13

Hyperflex Cisco Hyperflex ist eine «hyperkonvergente» Infrastruktur, die Rechenpower, Storage und Netzwerk in einem Server vereint. 17

IFTTT if-this-then-that. 41, 42
INS Institute for Networked Solutions der HSR. 13
IP Connectivity IP Connectivity beschreibt den Zustand eines IP-basierten Netzwerkes, wenn ein Gerät über eine IP Adresse angesprochen werden kann. 15
IP Helper Adresse Eine IP-Helper Adresse dient dazu, die Broadcast-Pakete vom Client-Netzwerk per Unicast zum DHCP-Server zu leiten, falls dieser sich in einem anderen Layer 2 befindet. 45
IPAM IP Address Management. 11–13, 31, 38, 44
ISC Internet Systems Consortium. 62
ISR Cisco Integrated Services Router ist ein Router von Cisco. 83, 92

Kickstart Kickstart beschreibt eine Möglichkeit, eine Installation des Betriebssystems teilweise oder komplett ohne Benutzerinteraktionen durchzuführen. 47, 57
KVM Die Kernel-based Virtual Machine ist eine im Linux-Kern enthaltene Technik, virtuelle Maschinen auf dem Host laufen zu lassen. 57

NAT Network Address Translation. 60
NFS Network File System. 38
NFV Network Function Virtualization. 10, 12, 13, 16, 32, 34, 39, 50

PXE Preboot Execution Environment. 37, 47, 92

RUP Rational Unified Process. 66

SVI Switched Virtual Interface. 34

UCS Cisco Unified Computing System ist die Rechenpower in einem Cisco Gerät. 37, 47, 92

VM virtuelle Maschine. 38, 57, 63
VMware ESXi Betriebssystem von VMware. 37
VRF Virtual Routing and Forwarding. 32, 34
vSAN virtual Storage Attached Network. 38

9.8 Abbildungsverzeichnis

1 Cisco ISR 4461 mit zwei UCS E-Series Server Blades . . . 10
2 Ausgerollter Standort mit redundanter Hardware . . . 11
3 Deployment Prozess für einen Aussenstandort . . . 12
4 Use Case Diagramm . . . 14
5 Risikomatrix Version 1 . . . 20
6 Risikomatrix Version 2 . . . 20
7 Context Diagram . . . 22
8 Container/Deployment Diagram . . . 23
9 Component Diagram . . . 24
10 BPMN 2.0 – Effektiver Site Deployment Prozess . . . 28
11 Aufgeklappte Sub-Workflows mit jeweils sequenzieller Abfolge . . . 29
12 StackStorm Site Deployment Input Parameter . . . 30
13 IP-Adresskonzept . . . 31
14 Netzwerkdesign – Routed L1/L2 . . . 33
15 Netzwerkdesign – Switched L1/L2 . . . 35
16 Netzwerkdesign – Switched L3 . . . 36
17 Check Point – Desings . . . 40
18 Manuelle Konfiguration im Infoblox . . . 44
19 Check Point Mgmt API aktivieren - Smart-Console . . . 56
20 Check Point initial setup – Installationsprozess in der Konsole starten . . . 58
21 Check Point initial setup – Partitionierung . . . 58
22 Check Point initial setup – Netzwerk Einstellungen . . . 59
23 Projektphasenplanung Gantt . . . 66
24 Zeitauswertung über die Wochen . . . 76
25 Stundenrapport Vergleich – Soll, Ist, Geplant . . . 77
26 Summierter Aufwand pro Kategorie . . . 77
27 StackStorm-Ansible – NetworkToCode – Slack . . . 79
28 Check Point Ansible-Mgmt-Collection – GitHub – Issues . . . 80
29 PXE Ablauf VMware [53] . . . 94

9.9 Literaturverzeichnis

[1] Inc. Amazon Web Services. Amazon Linux (Cloud-Init). URL: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/amazon-linux-ami-basics.html#amazon-linux-cloud-init. (accessed: 2019-12-19).

[2] David Barroso. NAPALM's documentation. URL: https://napalm.readthedocs.io/en/latest/. (accessed: 2019-12-09).

[3] Joachim Burkhardt. Risikomanagement beim Softwareprojektmanagement. Studiengang MBA - München, 2004.

[4] Kirk Byers. Netmiko. URL: https://github.com/ktbyers/netmiko. (accessed: 2019-11-11).

[5] CentOS. Kickstart Installations. URL: https://docs.centos.org/en-US/centos/install-guide/Kickstart2/. (accessed: 2020-01-06).

[6] check_point. Check Point Ansible Mgmt Collection. URL: https://galaxy.ansible.com/check_point/mgmt. (accessed: 2019-12-26).

[7] ChooseALicense. Apache License 2.0. URL: https://choosealicense.com/licenses/apache-2.0/. (accessed: 2019-10-27).

[8] ChooseALicense. GNU General Public License v3.0. URL: https://choosealicense.com/licenses/gpl-3.0/. (accessed: 2019-10-27).

[9] Cisco. CLI Configuration Guide for Cisco UCS E-Series Servers. URL: https://www.cisco.com/c/en/us/td/docs/unified_computing/ucs/e/3-2-x/sw/cli/config/b_3_2_x_CLI_Config_Guide/b_3_2_1_CLI_Config_Guide_chapter_011.html#task_332CDED69FD34FF9A4933F4C0A4BEBF5. (accessed: 2020-01-06).

[10] Cisco. Getting Started Guide for Cisco UCS E-Series Servers. URL: https://www.cisco.com/c/en/us/td/docs/unified_computing/ucs/e/gs/guide/b_Getting_Started_Guide.html. (accessed: 2019-12-27).

[11] Inc. Cisco Systems. Understanding Bridge Domain Interface (BDI). URL: https://www.cisco.com/c/en/us/support/docs/lan-switching/integrated-routing-bridging-irb/200650-Understanding-Bridge-Virtual-Interface.html#anc5. (accessed: 2019-10-10).

[12] Community. AWX. URL: https://github.com/ansible/awx. (accessed: 2019-12-26).

[13] The Apache Software Foundation. APACHE LICENSE, VERSION 2.0. URL: https://www.apache.org/licenses/LICENSE-2.0. (accessed: 2019-10-27).

[14] Paul Gorman. Virtual Laboratory with Linux, KVM, etc. URL: https://paulgorman.org/technical/virtual-laboratory.txt.html. (accessed: 2019-12-17).

[15] Docker Inc. Docker. URL: https://www.docker.com. (accessed: 2019-12-18).

[16] SaltStack Inc. SaltStack Documentation. URL: https://docs.saltstack.com/en/latest/. (accessed: 2019-12-26).

[17] Infoblox. Infoblox REST API. URL: https://www.infoblox.com/wp-content/uploads/infoblox-deployment-infoblox-rest-api.pdf. (accessed: 2019-10-27).

[18] InfobloxOpen. Infoblox Client. URL: https://github.com/infobloxopen/infoblox-client. (accessed: 2019-12-19).

[19] Kami. NAPALM. URL: https://github.com/StackStorm-Exchange/stackstorm-napalm. (accessed: 2019-12-09).

[20] Aaron Kili. How to Setup DHCP Server and Client on CentOS and Ubuntu. URL: https://www.tecmint.com/install-dhcp-server-client-on-centos-ubuntu/. (accessed: 2019-12-18).

[21] Pradeep Kumar. How to Install and use Open vSwitch with KVM on CentOS 7. URL: https://www.linuxtechi.com/install-use-openvswitch-kvm-centos-7-rhel-7/. (accessed: 2019-12-16).

[22] libvirt. Setting the NIC model. URL: https://libvirt.org/formatdomain.html#elementsNICSModel. (accessed: 2019-10-27).

[23] Check Point Software Technologies Ltd. Installing One Security Management Server. URL: https://sc1.checkpoint.com/documents/R80.30/WebAdminGuides/EN/CP_R80.30_Installation_and_Upgrade_Guide/206105.htm. (accessed: 2019-12-27).

[24] Check Point Software Technologies Ltd. R80.x Ports Used for Communication. URL: https://community.checkpoint.com/t5/Enterprise-Appliances-and-Gaia/R80-x-Ports-Used-for-Communication-by-Various-Check-Point/td-p/38153. (accessed: 2019-12-27).

[25] Canonical Ltd. The standard for customising cloud instances. URL: https://cloud-init.io/. (accessed: 2019-12-17).

[26] Check Point Ltd. Blink - Gaia Fast Deployment. URL: https://supportcenter.checkpoint.com/supportcenter/portal?eventSubmit_doGoviewsolutiondetails=&solutionid=sk120193. (accessed: 2019-12-09).

[27] Check Point Ltd. ClusterXL Administration Guide. URL: https://sc1.checkpoint.com/documents/R80.30/WebAdminGuides/EN/CP_R80.30_ClusterXL_AdminGuide/html_frameset.htm. (accessed: 2019-11-18).

[28] Check Point Ltd. Logging and Monitoring R80.30 Administration Guide. URL: https://sc1.checkpoint.com/documents/R80.30/WebAdminGuides/EN/CP_R80.30_LoggingAndMonitoring_AdminGuide/html_frameset.htm?topic=documents/R80.30/WebAdminGuides/EN/CP_R80.30_LoggingAndMonitoring_AdminGuide/120829. (accessed: 2019-12-09).

[29] Check Point Ltd. Running the First Time Configuration Wizard in CLI Expert mode. URL: https://sc1.checkpoint.com/documents/R80.30/WebAdminGuides/EN/CP_R80.30_Installation_and_Upgrade_Guide/html_frameset.htm?topic=documents/R80.30/WebAdminGuides/EN/CP_R80.30_Installation_and_Upgrade_Guide/214696&anchor=o198185. (accessed: 2019-10-21).

[30] Extreme Networks. StackStorm. URL: https://stackstorm.com/. (accessed: 2019-12-26).

[31] Check Point. How to configure partition sizes during Gaia installation. URL: https://supportcenter.checkpoint.com/supportcenter/portal?eventSubmit_doGoviewsolutiondetails=&solutionid=sk92303. (accessed: 2019-11-11).

[32] Check Point. How to install Gaia from USB device on Open Servers using ISOmorphic Tool. URL: https://supportcenter.checkpoint.com/supportcenter/portal?eventSubmit_doGoviewsolutiondetails=&solutionid=sk65205. (accessed: 2019-12-23).

[33] The CentOS Project. CentOS Cloud Images Download. URL: https://cloud.centos.org/centos/7/images/. (accessed: 2019-12-17).

[34] Inc. Red Hat. How to build your inventory. URL: https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html. (accessed: 2019-12-17).

[35] Inc. Red Hat. Infoblox Guide. URL: https://docs.ansible.com/ansible/latest/scenario_guides/guide_infoblox.html. (accessed: 2019-12-17).

[36] Inc. Red Hat. raw – Executes a low-down and dirty command. URL: https://docs.ansible.com/ansible/latest/modules/raw_module.html. (accessed: 2019-12-18).

[37] Inc. Red Hat. Red Hat Ansible. URL: https://www.ansible.com/. (accessed: 2019-12-26).

[38] Inc. Red Hat. Red Hat Ansible Tower. URL: https://www.ansible.com/products/tower. (accessed: 2019-12-26).

[39] Richard. Cloud Images with cloud-init Demystified. URL: https://packetpushers.net/cloud-init-demystified/. (accessed: 2019-12-17).

[40] Anthony Shaw. Ansible v.s. Salt (SaltStack) v.s. StackStorm. URL: https://medium.com/@anthonypjshaw/ansible-v-s-salt-saltstack-v-s-stackstorm-3d8f57149368. (accessed: 2019-10-27).

[41] SK. Setting Up DNS Server On CentOS 7. URL: https://www.unixmen.com/setting-dns-server-centos-7/. (accessed: 2019-12-18).

[42] StackStorm. Datastore. URL: https://docs.stackstorm.com/datastore.html. (accessed: 2019-12-09).

[43] StackStorm. StackStorm API Reference. URL: https://api.stackstorm.com. (accessed: 2019-12-23).

[44] StackStorm. StackStorm Exchange. URL: https://exchange.stackstorm.org. (accessed: 2019-12-27).

[45] StackStorm-Exchange. StackStorm Ansible. URL: https://github.com/zwiy/stackstorm-ansible. (accessed: 2019-12-23).

[46] U880D. How to create a custom ISO image in CentOS. URL: https://serverfault.com/a/521672. (accessed: 2019-11-03).

[47] Inc VMware. Create an Installer ISO Image with a Custom Script. URL: https://docs.vmware.com/en/VMware-vSphere/6.7/com.vmware.esxi.upgrade.doc/GUID-C03EADEA-A192-4AB4-9B71-9256A9CB1F9C.html. (accessed: 2019-11-03).

[48] Inc VMware. Installing ESXi Using PXE. URL: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/vsphere-esxi-vcenter-server-60-pxe-boot-esxi.pdf. (accessed: 2019-11-06).

[49] Inc VMware. Lizenzierung für ESXi-Hosts. URL: https://docs.vmware.com/de/VMware-vSphere/6.5/com.vmware.vsphere.vcenterhost.doc/GUID-7AFCC64B-7D94-48A0-86CF-8E7EF55DF68F.html. (accessed: 2019-10-27).

[50] Inc VMware. Lizenzierung für vCenter Server. URL: https://docs.vmware.com/de/VMware-vSphere/6.5/com.vmware.vsphere.vcenterhost.doc/GUID-F2B7886A-A4A6-4D1B-BF6F-BD0EEE573602.html. (accessed: 2019-10-27).

[51] Inc VMware. vCenter Architecture Overview. URL: https://docs.vmware.com/en/VMware-vSphere/6.7/com.vmware.vsphere.avail.doc/GUID-2D395533-4C7C-45A6-9B5C-0A78C3E3A5C8.html. (accessed: 2019-11-18).

[52] Inc VMware. vCenter HA Hardware and Software Requirements. URL: https://docs.vmware.com/en/VMware-vSphere/6.7/com.vmware.vsphere.avail.doc/GUID-8FD87389-8CC9-4298-8B08-A1526FB44524.html. (accessed: 2019-11-18).

[53] Inc. VMware. Installing ESXi Using PXE. URL: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/vsphere-esxi-vcenter-server-60-pxe-boot-esxi.pdf. (accessed: 2019-10-21).

9.10 Danksagungen

Wir möchten uns bei folgenden Personen ganz herzlich für Ihre Unterstützung während der Bachelorarbeit bedanken:

- Prof. Laurent Metzger (INS) als Examinator für die Betreuung unserer Arbeit
- Philip Schmid (INS) als Betreuer für die Betreuung und Unterstützung bei unserer Arbeit
- Laurent Billas und Serge Pidoux (FUB) als Industriepartner für die Aufgabenstellung und die zur Verfügung gestellte Hardware
- Marcel Witmer (Cisco) als Experte für die Bewertung der Arbeit

Des Weiteren danken wir auch folgenden Personen für ihre geleistete Unterstützung:

- Farhad Mehta (HSR) als Gegenleser unserer Arbeit
- Christian Spielmann (INS) für die Einrichtung der Studienräume und die prompte Bereitstellung der VMs und DNS-Einträge
- Patrik Honegger und Sandor Renz (Check Point) für die Bereitstellung der Check Point Software für die Firewall