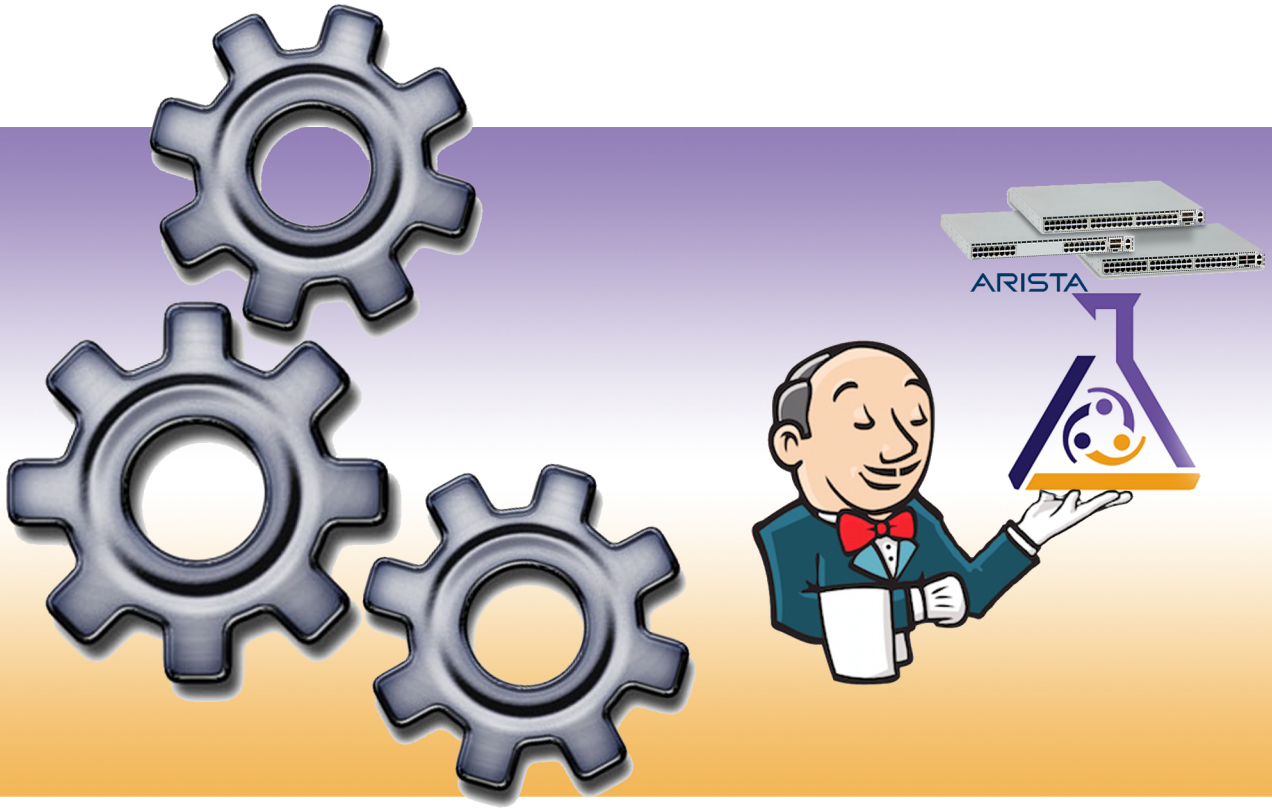
Konfigurationsdaten Automatisierung für Arista Switches

Hochschule Luzern - Technik & Architektur

Industrieprojektarbeit Informatik (PAWI)



Bern, 03.01.2016

Raffaela Wüst

Selbständigkeitserklärung

Die Verfasserinnen und Verfasser bestätigen mit ihrer Unterschrift, dass die vorliegende Arbeit selbstständig, ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt wurde. Die aus fremden Quellen (einschliesslich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form noch nicht vorgelegt worden.

Ort, Datum:

Unterschrift:

Abstract

Swisscom (Schweiz) AG setzt im Rahmen der Cloud-Entwicklung auf eine möglichst hohe Automatisierung der Infrastruktur Komponenten. Zum Einsatz für das automatische Deployment kommt Puppet - ein Systemkonfigurationsmanagement Tool - mit Hiera (ein Key/Value Lookup Tool). Die Hieradaten erweitern die generellen Puppet Roles mit Server oder stackspezifischen Konfigurationswerten. Das Erstellen dieser Hieradaten ist heute ein manueller Prozess, erfordert viel Zeit und ist fehleranfällig. Das Ziel dieser Projektarbeit ist, die Konfigurationsdaten fur Arista Switches auf der Basis einer Datensource automatisch generieren zu lassen und mit wenigen, manuellen Schritten in die Stages DEV (Development), INT (Integration) und PRD (Production) der CI/CD Umgebung auszurollen. Die Arbeit ist in zwei verschiedene Bereiche unterteilt. Der erste Teil beschäftigt sich damit, eine eigenständige CI/CD Umgebung mit virtuellen Arista Switches aufzubauen, der zweite Teil beinhaltet das Entwickeln und Paketieren eines Hieradatengenerators, welcher in Ruby programmiert ist. Als Resultate dieser Arbeit gehen vier Artefakte hervor. Dazu gehören: Die nachfolgenden Dokumente „Konzept Konfigurationsdatengenerator“ und „Projektmanagement“, sowie eine lauffähige CI/CD Umgebung und eine Individualsoftware - der Konfiguraitonsdatengenerator „hieragen“.

Swisscom (Switzerland) Ltd. pursue in the cloud development on a very high automation of the infrastructure components. For the infrastructure automation, Puppet (system configuration management) with Hiera (key-value-lookup tool) is in place. The individual Hieradata extend the general Puppet Role with server or stack-specific configuration values. The process of Hieradata creation is today manual, requires a lot of time and it’s error prone. The goal of this project is, to automatically generate the Hieradata for Arista Switches based on a data source. With a few manual steps, the generated Hieradata should be rollout to the three different stages DEV (development), INT (integration) and PRD (production) of the CI/CD environment. This project contains two different parts. The first part deals with the setup of an independent CI/CD environment including virtual Arista Switches, the second part includes the programming and packaging of the Hieradatagenerator software who is programmed in Ruby. The outcomes of this project are four artefacts. The two following documents „Concept Configurationdatagenerator“ and „Projectmanagement“ as well as a performing CI/CD environment and the individual software - the Hieradatagenerator „hieragen“.

Inhalt

Dokument: Konzept & Implementierung

Dokument: Projektmanagement

CD:

- Sourcecode
- Scripts
- Hieradaten
- RPM's
- Aufgabenstellung

Konfigurationsdaten Automatisierung für Arista Switches

Konzept & Implementierung

28. April 2016

Hochschule Luzern - Technik & Architektur

Industrieprojektarbeit Infromatik (PAWI)

Autorin:
Raffaela Wüst
Swisscom (Schweiz) AG
Ey 10
3063 Ittigen
raffaela.wuest@swisscom.com

Betreuender Dozent:
Prof. Andreas Kurmann
Hochschule Luzern - T&A
Technikumstrasse 21
6048 Horw
andreas.kurmann@hslu.ch

Inhalt

1. Einleitung 1
1.1 Firmenportrait 1
1.2 Ausgangslage und Problemstellung 1
1.3 Ziele und Scope 1
1.4 Resultate 2
1.5 Begriffe und Abkürzungen 3

2. IST-Situation 5
2.1 Technologien und Produkte 5
2.1.1 Puppet 5
2.1.2 Hieradata & YAML 6
2.1.3 Jenkins 6
2.1.4 Gitlab 6
2.1.5 Pulp Server 7
2.1.6 Arista Switches 7
2.1.7 Datensource EnClouder 8
2.1.8 Alternative Datensource 8

3. Requirements 9
3.1 Vision 9
3.2 User Stories 9
3.2.1 CI/CD Umgebung 9
3.2.2 Konfigurationsdatengenerator 10
3.3 Use Cases 10
3.3.1 Use Case 1: Fully Puppet managed CI/CD environments 10
3.3.2 Use Case 2: Artefact installation over RPM's 11
3.3.3 Use Case 3: Three different identical environments (DEV, INT, PRD) 12
3.3.4 Use Case 4: Generate automatically a YAML file on a source base, do a syntax check and push it to a repository 13
3.4 Nicht-Funktionale Anforderungen 13

4. Systemdesign 15
4.1 CI/CD Umgebung 15
4.2 Konfigurationsdatengenerator 16
4.2.1 Funktionale Spezifikation 16
4.2.2 Benutzerinteraktion 18
4.2.3 Schnittstellen 19

5. Implementation 21
5.1 CI/CD Umgebung 21
5.1.1 Vorbereitung Hieradaten erstellen 21
5.1.2 Infrastruktur bereitstellen 22
5.1.3 Connectivity 22
5.1.4 Deployment Management Umgebung 23
5.1.5 Setup Puppet Master 24
5.1.6 Setup Pulp Server 26
5.1.7 Setup Jenkins 28
5.1.8 Arista Switches 29
5.1.9 Konfiguration Arista Switches 32
5.2 Entwicklung Konfigurationsdatengenerator „hieragen“ 34
5.2.1 Klassendiagramm 34
5.2.2 Funktionen des Generators 35
5.2.3 Unit Testing mittels Rspec 38
5.3 Einbindung des Konfigurationsdatengenerators in CI/CD Umgebung 38
5.4 Einsatz des Generators 39
5.4.1 Installation 39
5.4.2 Konfiguration 39
5.4.3 Weitere Funktionen 40
5.4.4 Ausführung / Generierung von Hieradaten 41

6. Testing 42
6.1 Testfälle 42
6.1.1 Testing CI/CD Umgebung 42
6.1.2 Testing Konfigurationsdatengenerator 46
6.1.3 Integrationstests 47
6.2 Testresultate 49

7. Fazit und Lessons Learned 54
7.1 Ausblick 54

Abbildungsverzeichnis 55

Tabellenverzeichnis 55

Literatur 56

Anhang A Serverliste CI/CD Umgebung 57

Anhang B CI/CD und Puppet Konfigurationen 58

Anhang C Testresultate 67

1. Einleitung

Im Bereich Enterprise Customers verfolgt Swisscom (Schweiz) AG eine Cloudstrategie und bietet ihren Kunden verschiedene Cloudlösungen von Infrastructure (IaaS) bis hin zu Software as a Service (SaaS) an. Um die Kundenbedürfnisse an eine Cloud zu erfüllen, steht neben vielen weiteren Anforderungen eine hohe Automatisierung der Infrastruktur und Prozesse im Vordergrund. Die Automatisierung der Infrastrukturkomponenten wurde mit Puppet (Tool für Systemkonfigurationsmanagement) umgesetzt, ist bereits weit fortgeschritten und ermöglicht schon jetzt ein komplett automatisiertes Stack-Deployment innerhalb eines Tages. Puppet definiert und speichert den Zustand der IT Infrastruktur und sorgt dafür, dass sich die Infrastruktur immer automatisch im gewünschten, vordefinierten Zustand befindet. Jeder Server wird einer Puppet Role zugewiesen. Eine Puppet Role ist generisch und könnte beispielsweise alle Konfigurationen für einen Switch beinhalten. In Hiera (Key/Value Lookup Tool für Konfigurationsdaten) werden Server spezifische Werte hinterlegt, die die generische Puppet Role verwendet.

1.1 Firmenportrait

Swisscom (Schweiz) AG ist führender IT Service Provider in der Schweiz. Um im aktuellen Markt konkurrenzfähig zu bleiben, die Kundenwünsche zu erfüllen und dem Kostendruck des Marktes Stand zu halten, setzt Swisscom auf eine Cloudstrategie. Im Bereich Enterprise Customers wird die Swisscom Cloud gebaut und entwickelt. Eines dieser Teams - das Cloud Automation Team - hat zum Ziel eine möglichst hohe Automatisierung für die in der Cloud eingesetzten Produkte zu erreichen. (Wüst 2015)

1.2 Ausgangslage und Problemstellung

Jedes Stack-Deployment bedingt viele Konfigurationsdaten für die verschiedenen Puppet Roles, die vorgängig zusammengestellt werden müssen. Dieser Prozess wird manuell ausgeführt, ist dadurch fehleranfällig und dauert mindestens 2 Tage. Das Ziel ist, möglichst alle Konfigurationsdaten automatisch generieren zu lassen und diese Daten in den Continuous Integration / Continuous Delivery (CI/CD) Workflow einzubinden, um den manuellen Aufwand möglichst klein zu halten.

1.3 Ziele und Scope

Ziel:
Das Ziel des Projektes ist, die Konfigurationsdaten für Arista Switches auf der Basis einer Datensource automatisch generieren zu lassen und per Knopfdruck in die Stages DEV (Development), INT (Integration) und PRD (Production) der CI/CD Umgebung zu deployen.

Scope:
Aufgrund der begrenzten Projektzeit von rund 180 Arbeitsstunden wird im Rahmen dieses Projektes nur die Konfigurationsdatenautomatisierung für Arista Switches umgesetzt. Weitere Automatisierungen für andere Komponenten - wie zum Beispiel Router oder virtuelle Maschinen - sind out of Scope.

1.4 Resultate

Die Projektarbeit umfasst die folgenden vier Resultate:

1. Konzept Konfigurationsdatengenerator
    - Einleitung
        - Firmenportrait
        - Ausgangslage / Problemstellung
        - Ziele und Scope
        - Resultate
        - Begriffe und Abkürzungen
    - Requirements
    - Systemdesign
        - CI/CD Umgebung
        - Konfigurationsdatengenerator
    - Implementation
        - CI/CD Umgebung
        - Entwicklung Konfigurationsdatengenerator
        - Einbindung des Konfigurationsdatengenerators in CI/CD Umgebung
    - Testing
    - Fazit und Lessons Learned
2. CI/CD Umgebung
    - Aufbau der virtuellen Maschinen (zirka 20)
    - Installation / Konfiguration Puppet Master, Pulp Server, Jenkins
    - Jenkins Anbindung an Gitlab
    - Installation vEOS auf virtuellen Maschinen für Switch Simulation
3. Konfigurationsdatengenerator
    - Entwicklung Konfigurationsdatengenerator (Ruby)
    - Anbindung des Generators an die Datensource
    - Einbindung des Generators in CI/CD Umgebung
4. Projektmanagementdokument
    - Ausgangslage / Ziel / Resultat
    - Detaillierte Arbeitspakete während den einzelnen Projektphasen

- Klar definierte und terminierte Meilensteine
- Risikomanagement
- Reporting
- Planung
- Arbeitsjournal

1.5 Begriffe und Abkürzungen

API	Application Programming Interface, Programmierschnittstelle
Arista Networks	amerikanischer Netzwerkkomponenten Hersteller
CI/CD	Continuous Integration / Continuous Delivery
DEV	Development (Umgebung)
DNS	Domain Name System
DOPv	Deployment Orchestrator for Puppet, Inhouse entwickelte Software
DSL	Domain Specific Language ermöglicht Job Beschreibungen basierend auf der Sprache Groovy
EnClouder	Server Provisioning Software
EOS	Extensible Operating System von Arista
ERB	Embedded Ruby
FQDN	Full qualified Domain Name
Gitlab	Versionskontrollsystem
GLI	Git-Like Interface Command Line Parser
Hiera	Key/Value Lookup Tool für Konfigurationsdaten
HSLU	Hochschule Luzern
IaaS	Infrastructrue as a Service
INT	Integration (Umgebung)
IPMI	Intelligent Platform Management Interface, Schnittstelle zur Verwaltung von ausgeschalteten oder neu aufgesetzten Servern ohne Betriebssystem
ISO	Archiv Datei einer optischen Disk
Java	Objektorientierte Programmiersprache
Jenkins	Buildserver
KVM	Kernel-based Virtual Machine
PAWI	Industrieprojektarbeit Infromatik HSLU
PRD	Production (Umgebung)
Pulp	Repository Server
Puppet	Tool für Systemkonfigurationsmanagement
QCOW2	Dateiformat für Disk Images
RHEL	Redhat Enterprise Linux
RHEV	RedHat Enterprise Virtualization, basierend auf dem KVM Hypervisor
RPM	RPM Package Manager, früher Red Hat Package Manager

Ruby	dynamische, objektorientierte Programmiersprache
SaaS	Software as a Service
SPINE Switch	Kern Switch , aufgebaut aus mehreren Layer 3 Switches mit hoher Durchsatzrate und hoher Port Dichte
SSH	Secure Shell, Netzwerkprotokoll für verschlüsselte Verbindungen
TOR Switch	Top of Rack Switch
UML	Unified Modeling Language, Modellierungssprache zur Spezifikation von Software
vEOS	virtual Extensible Operating System von Arista
VMDK	Virtual Machine Disk
YAML	Markup Language
YUM	Yellowdog Updater Modified ist eine Package-Management Utility

2. IST-Situation

Die bereits existierenden CI/CD Umgebungen sind jeweils nach dem gleichen Schema aufgebaut. Dazu stehen verschiedene Produkte im Einsatz. Diese sind im Kapitel 2.1 „Technologien und Produkte“ näher beschrieben und sollen eine Übersicht und ein Verständnis für die einzelnen Produktbezeichnungen liefern. Die CI/CD Umgebungen unterscheiden sich pro Kunde minimal, sind aber grundsätzlich gemäss folgender Systemlandschaft (Abbildung 1) aufgebaut. Übergreifend werden alle CI/CD Umgebungen über eine „Hauptumgebung“ verwaltet.

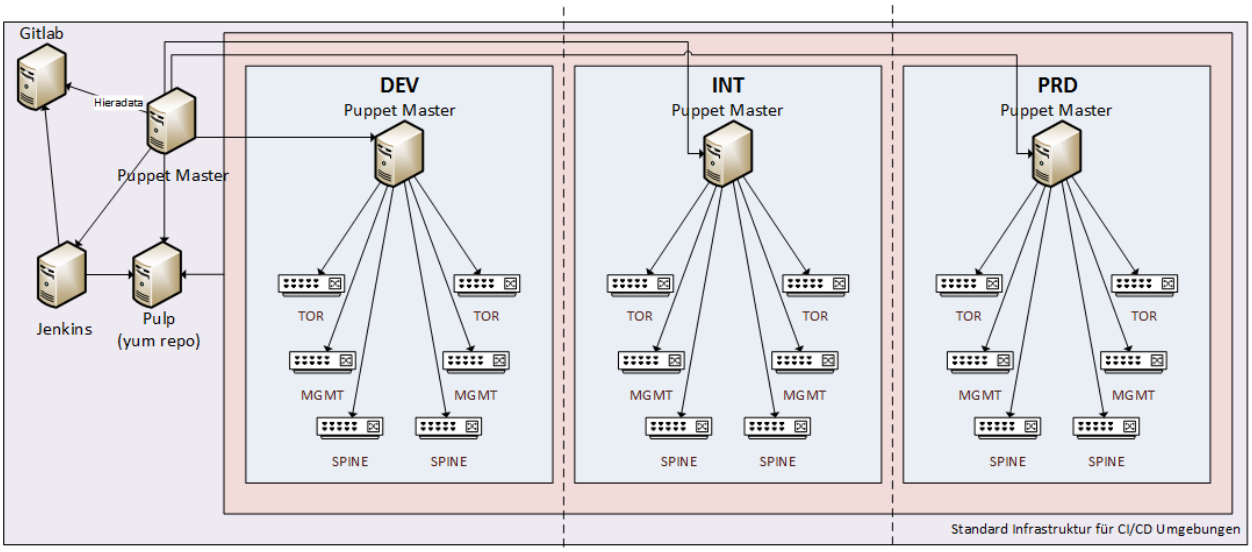


Abbildung 1: Auszug Systemlandschaft Standard CI/CD Umgebungen

2.1 Technologien und Produkte

In der Arbeit werden verschiedene Produkte und Technologien eingesetzt. Um das nötige Produktwissen zu erhalten, werden in den nachfolgenden Unterkapitel diese Produkte und Technologien genauer beschrieben.

2.1.1 Puppet

Puppet ist eine Konfigurationsmanagement Lösung, die den Zustand der IT Infrastruktur definiert. Sie sorgt dafür, dass sich die Infrastruktur immer automatisch im gewünschten, vordefinierten Zustand befindet. Puppet kann jeden Schritt des Software Devlivery Prozesses automatisieren. (Puppet-Labs 2015b) Swisscom setzt als Abstraktionslayer zwischen dem Server und dem Puppet Modul das „Roles und Service Pattern“ (Abbildung 2) ein.

Grundlegend sind Roles und Services reguläre Puppet Module und unterscheiden sich technisch nicht. Es gilt: ein Node hat eine einzige Role. Eine Role besteht aus einem oder mehreren Services, und ein Service kann wieder aus anderen Services bestehen und beinhaltet die benötigten Ressourcen. (z.B.

RPM Packages - RPM steht für RedHat oder RPM Package Manager) Eine Role kann für mehrere Nodes wiederverwendet werden.

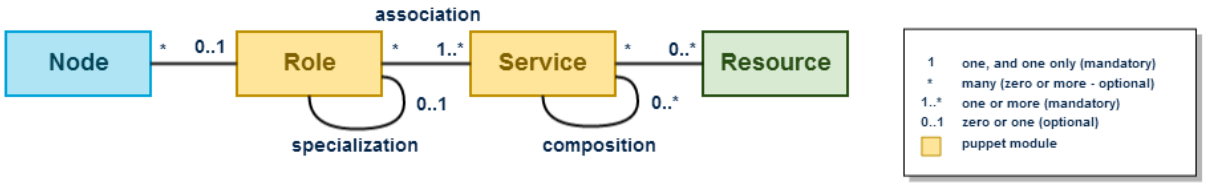


Abbildung 2: Puppet Roles und Service Pattern

2.1.2 Hieradata & YAML

Für die Wiederverwendung der Puppet Roles kommt Hiera zum Einsatz. Hiera ist ein Key/Value Lookup Tool für Konfigurationsdaten. (Puppet-Labs 2015a) In Hiera werden Node spezifische Werte für die Variablen in Puppet hinterlegt, die Puppet beim Run auf dem jeweiligen Server verwenden kann. Hiera verwendet YAML - eine Markup Language zur Speicherung der einzelnen Werte. Um individuelle Konfigurationen auf bestimmten Ebenen in einer Infrastruktur Umgebung zu definieren gibt es eine Hierarchie über die YAML Files. Das default.yaml umfasst alle Konfigurationswerte für die komplette Umgebung - beinhaltet als Beispiel die DNS Server. Die nächst tiefere Stufe ist das stack.yaml. Dieses beinhaltet Informationen für eine bestimmte Gruppe an Server und als Letztes gibt es pro Server eine Konfigurationsdatei, die jeweils mit dem Full qualified Domain Name FQDN.yaml benannt wird. Die Node Konfiguration ist pro Server individuell und kann durch die Angabe des stack.yaml dessen Informationen erben. Die Werte aus dem default.yaml werden automatisch - ohne explizite Angabe - an alle Server vererbt.

2.1.3 Jenkins

Jenkins steht bei Swisscom als Build Server zum Einsatz. Er bezieht über Gitlab (Versionskontrollsystem) den Source Code der eigen entwickelten Software sowie der Puppet Roles und buildet diesen Source Code für Linux Umgebungen zu RPM's. Die gebuildeten RPM's werden anschliessend auf dem Pulp Server (Repository Server) abgelegt. Mehr zum Pulp Server ist im Kapitel 2.1.5 „Pulp Server" beschrieben.

2.1.4 Gitlab

Gitlab wird als Source Code und Versionskontrollsystem verwendet. Die interne Instanz ist über gitlab.internaldomain.ch erreichbar, Projekte sind nur für berechtigte Mitarbeiter einseh- und bearbeitbar. Der komplette Sourcecode für die Puppet Roles, den Konfigurationsdatengenerator und die Hieradaten werden in Gitlab gepflegt. Die Schnittstelle zwischen CI/CD Umgebung und Gitlab ist essentiell. Änderungen der oben erwähnten Komponenten werden stets auf Gitlab gepusht, von wo aus die Weiterverarbeitung der Informationen angestossen wird. (z.B. RPM Build der Puppet Module)

2.1.5 Pulp Server

„Pulp ist eine Plattform um Software Package Repositories zu managen." (Pulp-Project 2015) Auf dem Pulp Server werden die für das Projekt benötigten Puppet Roles, RPM Packages und die Software des Konfigurationsdatengenerators abgelegt. Die Pulp Repository URL's werden auf den unterschiedlichen Linux Server als YUM Repositorys (Package-Management Utility) konfiguriert. Das heisst, der Server kann von dieser Source seine zu installierende Software beziehen. Grundsätzlich verfügen die Server von Swisscom über keinen Zugriff auf das Internet und können daher ihre benötigten Softwarepakete nur von internen, vertrauenswürdigen Sourcen beziehen.

2.1.6 Arista Switches

„Arista Networks ist ein amerikanischer Netzwerkkomponenten Hersteller, der sich auf Software getriebene Netzwerk Lösungen für grosse Datencenter spezialisiert hat. Das Herzstück von Arista Plattformen ist das Extensible Operating System EOS, ein innovatives Netzwerkbetriebssystem mit Single-Image Konsistenz über Hardware Plattformen hinweg." (Arista-Networks 2015a) Im Rahmen dieser Arbeit werden die Switches in einer virtuellen Maschine simuliert. Dafür bietet sich das virtuelle Extensible Operating System (vEOS) von Arista bestens an. (Arista-Networks 2015b)

Switch Architektur
Zum Einsatz kommen drei verschiedene Switchtypen.

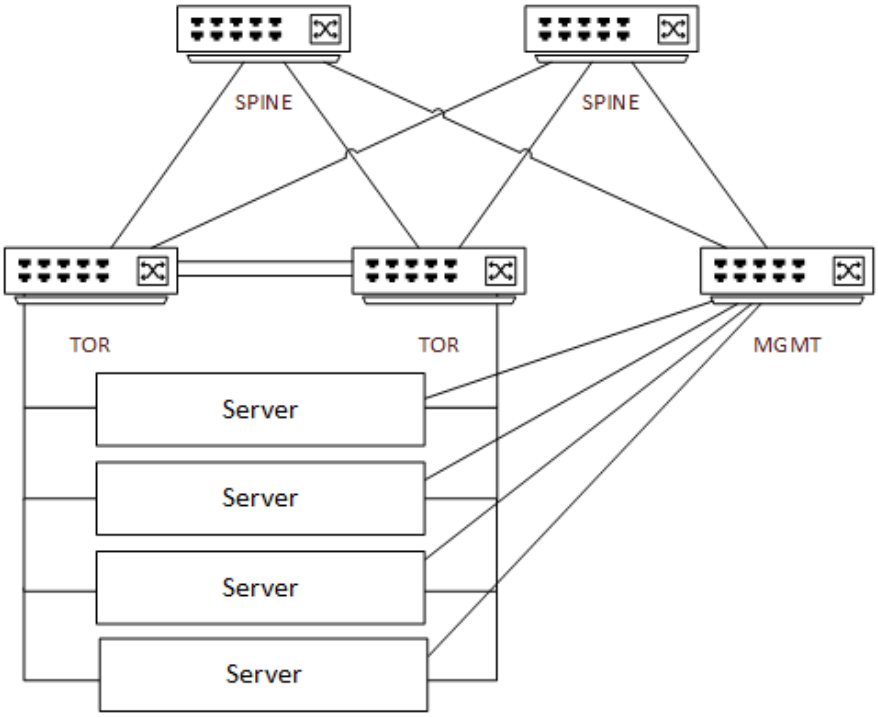


Abbildung 3: Switch Architektur

Die SPINE Switches entsprechen in der Architektur den Core Backbone Switches. Sie verbinden mehrere Racks in einem Rechenzentrum und managen die TOR Switches. Die Top of Rack (TOR) Switches können als Leafes in der leaf-spine Architektur - vgl. (Rouse 2015) - angesehen werden. Wie aus der Bezeichnung Top of Rack hervorgeht, ist ein TOR Switch oben auf den Racks angebracht und vereinfacht dadurch die Verkabelung. (IT-Wissen 2015) Die TOR Switches managen die effektiven Server zur Laufzeit und deren Datentraffic. Für die initiale Serverkonfiguration steht der Management Switch zur Verfügung. Er kann auch als Leaf angesehen werden, stellt jedoch zur Laufzeit lediglich noch das IPMI Interface (Schnittstelle zur Verwaltung von ausgeschalteten oder neu aufgesetzten Servern ohne Betriebssystem) zur Verfügung. Die Abbildung 3 gibt eine Übersicht wie die drei Switch Arten SPINE, TOR und Management zusammenspielen.

2.1.7 Datensource EnClouder

Der EnClouder ist eine Server Provisioning Software, die von der Firma Puzzle ITC GmbH für Swisscom (Schweiz) AG entwickelt wurde. „Der EnClouder stellt Stammdaten für Hardware, Netzwerke und Software in einer Baumstruktur dar. Mit diesen Daten werden Systeme konfiguriert und automatisch installiert.“ (Puzzle-ITC 2015)

Der EnClouder ist nach wie vor in Entwicklung, weshalb dieser für Arista Switches noch nicht über alle nötigen Daten für die Konfigurationsgenerierung verfügt. Möglicherweise kann der EnClouder nicht als Datensource verwendet werden. Dieses Risiko wurde ins Risikomanagement aufgenommen und klassifiziert. Falls die benötigten Daten im EnClouder bis zum 27.11.2015 nicht verfügbar sind muss das Projekt auf einer Mockdatenbasis aufgebaut werden.

2.1.8 Alternative Datensource

Als alternative Datensource zum EnClouder wird eine Textdatei in Betracht gezogen, die alle nötigen Input Daten enthält. Diese Datei wird als Datensource für das Projekt erforderlich, solange der EnClouder noch nicht über alle nötigen Datensätze für Arista Switches verfügt. Am 27.11.2015 wurde die Entscheidung getroffen, die alternative Datensource für das Projekt zu verwenden.

3. Requirements

Die Requirements sind gemäss dem HSLU Modul „Software Architecture, Quality and Testing“ evaluiert und bewusst in Englisch verfasst, um die gelernten Schemata anwenden zu können.

3.1 Vision

For operations and engineers
Who want to fully automatically generate the configurationdata files for Arista switches (Hieradata)
The new configurationdata generator
Is flexible enough
That all parameters for the specific switch Puppet Roles can be automatically generated based on a defined data source.

3.2 User Stories

Eine User Story umfasst eine Anforderung, die aus Sicht eines Users verfasst wird. Sie enthält keine technischen Implementationsdetails.

Die User Stories sind nach dem folgenden Schema aufgebaut: „As a [role] I can [function] so that [rationale].“ Sie wurden in die Unterkapitel CI/CD Umgebung und Konfigurationsdatengenerator unterteilt.

3.2.1 CI/CD Umgebung

1. „As an operator I need to be sure the fully CI/CD environment is Puppet managed so that the servers are at very time in the predefined state.“
Definition of done: Every 30 minutes a Puppet run will be done on every Puppet managed server to restore the defined state.

2. „As an operator I need to be sure only RPM’s are installed on the CI/CD environment so that all installed artefacts are versioned and reproducible. “
Definition of done: All artefacts (Puppet Roles, generator and other software) are built on a Jenkins and uploaded as a RPM to the Pulp repository server. From where the artifacts can be installed via YUM package manager on the servers.

3. „As an operator I need three different environments for CI/CD so that the configurationdata can be generated in a DEV, INT and PRD environment.“
Definition of done: There are three identical environments (DEV, INT and PRD) where it is possible to generate the Hieradata for each environment by click.

3.2.2 Konfigurationsdatengenerator

4. „As an operator I can generate the Hieradata for Arista switches automatically so that I have no manual effort.“
Definition of done: The YAML file for the specific Arista switch is generated with the correct filename and the correct values in it and it is uploaded correctly to the Gitlab.

5. „As an operator I need to be sure the generated Hieradata for Arista switches are syntactically correct so that I do not get an syntax error during Puppet run. “
Definition of done: Before uploading the Hieradata YAML file to Gitlab, it is syntax checked and if there are syntax errors, they would be corrected by the generator or inform the user about the error.

6. „As an operator I need to be sure that all necessary variables for the specific Puppet Role are in the configurationdata file so that I can do a Puppet run and setup my environment without errors.“
Definition of done: The Puppet run on the specific Arista switch will run properly and set the environment up without errors.

7. „As an operator I need to be informed if the configurationdata generator generates some errors so that I can react, fix the issue and restart the generation of configuration data for Arista switches. “
Definition of done: If the configurationdata generator throws errors, the operator get’s an error message and an entry in the log file, what’s exactly is wrong in the process.

8. „As an operator I need an preview of the YAML file before it’s saved to Gitlab. “
Definition of done: The YAML file is shown to the operator in a preview output, and it’s possible to restart the generation if a value is missing or wrong.

9. „As an operator I need the Hieradata saved to a git repository. “
Definition of done: The YAML file with the Hieradata is successfully saved to the internal Gitlab repository.

3.3 Use Cases

Ein Use Case fasst verschiedene User Stories zusammen und bildet einen konkreten technischen Ablauf mit Erfolgsszenarien und Ausnahmen. Die einzelnen Use Cases referenzieren auf die entsprechend zugehörigen User Stories.

3.3.1 Use Case 1: Fully Puppet managed CI/CD environments

Primary Actor: Puppet Master & Engineer
Scope: The CI/CD environment

Reference: User Story 1
Level: System goal
Preconditions: There are three identically setup environments who are completely separated and on each machine Puppet is installed. There is a Puppet Master who's job is to manage all the Puppet Agents. Puppet Roles and the Pulp servers are available and Hieradata (at least a minimum) is setup for the agents.
Trigger: Every 30 minutes a Puppet run will reset the servers to the predefined state.

Main success scenario:

1. Hieradata for the Puppet Master is setup correctly - and includes the correct Puppet Roles who are necessary for the Puppet Agents.
2. via a Puppet run on the Puppet Master all Puppet Roles are installed
3. to get the necessary Hieradata for the agents, the folder /etc/puppet/hieradata exists on the machine and via git clone the Gitlab repository with the Hieradata for the agents is available.
4. auto sign file has to be created, so that all servers would be automatically signed by the Puppet Master during the first run.
5. do a Puppet run on each server with specifying the Puppet Master „puppet agent -t server puppetmaster_fqdn"

Extensions:

1. Puppet Roles are not available - check connectivity to the Pulp server, check if RPM packages are available on the Pulp server
2. Errors during Puppet run - check connectivity to the Puppet Master, check correctness of Hieradata or search for solutions on the internet

3.3.2 Use Case 2: Artefact installation over RPM's

Primary Actor: Puppet Master & Engineer
Scope: Puppet Master, Puppet Agents
Reference: User Story 2
Level: System goal
Preconditions: All necessary RPM's are available on the Pulp server and connectivity between Pulp server and Puppet Master works as expected. Puppet Roles and Hieradata are setup correctly.
Trigger: Puppet run every 30 Minutes

Main success scenario:

1. „puppet agent -t" is done on every server every 30 minutes. The Puppet Master is Puppet managed as well.

2. all RPM's (Puppet Roles and specific software packages) are installed during the Puppet run if they aren't already installed on the machines.
3. Puppet installs the RPM's over YUM
4. Puppet run finishes successfully and with a check „rpm -qa | grep "packagename" " the Engineer knows, if Puppet did his work.

Extensions:

1. especially during upgrade it's possible, that Puppet get some errors, the packages are not available. The solution for this problem is to clean the YUM cache „yum clean all" and restart the Puppet run.

3.3.3 Use Case 3: Three different identical environments (DEV, INT, PRD)

Primary Actor: Engineer
Scope: One Puppet Master, two TOR switches, two management switches and two SPINE switches for each environment
Reference: User Story 3
Level: System goal
Preconditions: The prerequisites / requirements for the needed servers are clear - has to be checked which prerequisites a vEOS needs. A matured system design. Specified the stack, where the project should take place
Trigger: Start of Sprint 3

Main success scenario:

1. depends on the stack where the environment should be built.
   a) If it's an existing already in used stack, the servers have to be ordered over the infrastructure team.
   b) If it's a new or a development stack, the servers could be setup by the project itself.
2. Order or create the servers with the specific requirements for vEOS
3. check if they are up and running
4. setup the connectivity to the Puppet Master and Pulp server
5. check if all works properly

Extensions:

1. Servers not available after staging - take a look with the infrastructure team at the situation.
2. connectivity will not work properly - take a look with the connectivity team at the situation.

3.3.4 Use Case 4: Generate automatically a YAML file on a source base, do a syntax check and push it to a repository

Primary Actor: Configurationdata Generator
Scope: The source software and the configurationdata generator
Reference: User Stories 4,5,9
Level: System goal
Preconditions: The source offers the correct information and the connectivity between source and generator is granted.
Trigger: The operator or engineer wants to deploy a new Arista switch and needs therefore the corresponding Hieradata.

Main success scenario:

1. Starting configurationdata generator in a environment (e.g. DEV)
2. The generator connects to Gitlab to get all needed variables from the Puppet Role
3. The generator buffers this variables
4. The generator connects to the data source (EnClouder) and gets all needed /available values
5. buffers this values as well
6. generates an YAML file and exports the buffered variables and values to the file
7. the YAML file would be syntax checked
8. the output of the YAML file is shown to the user over the console
9. saved and pushed to the Gitlab repository for Hieradata

Extensions:

1. if there is an error in every step, the generator throws an error to the console and in a log file

3.4 Nicht-Funktionale Anforderungen

Drei Nicht-Funktionale Anforderungen an den Hieradatengenerator sind spezifiziert.

Performance
Die Aufbereitung eines YAML-Files für die Hieradaten über den Generator dauert nicht länger als 10 Sekunden. Der Upload aller generierten YAML-Files ins Gitlab dauert nicht länger als 20 Sekunden.

Reproduzierbarkeit

Jeder Stand der Generator Software kann reproduziert werden. Fehler der Software werden in ein Logfile geschrieben und können somit eruiert werden.

Wartbarkeit

Der Konfigurationsdatengenerator muss im laufenden Betrieb in der CI/CD Umgebung für das Betriebsteam wartbar sein. Die Generator Software ist genau dokumentiert und gut programmiert. Als Beispiel sollen die Konfigurationswerte für die Generator Software ausserhalb des Installationspfades auf dem Server abgelegt werden, sodass bei einem Upgrade der Software diese Daten nicht überschrieben werden.

4. Systemdesign

Das Systemdesign teilt sich in zwei Unterkapitel. Das Kapitel CI/CD Umgebung umschreibt die Konzeption der Infrastruktur und das Kapitel Konfigurationsdatengenerator beschreibt das Systemdesign der Individual-Software.

4.1 CI/CD Umgebung

Die CI/CD Umgebung wird in der Labor Umgebung nach folgender Architektur (Abbildung 4) aufgebaut.

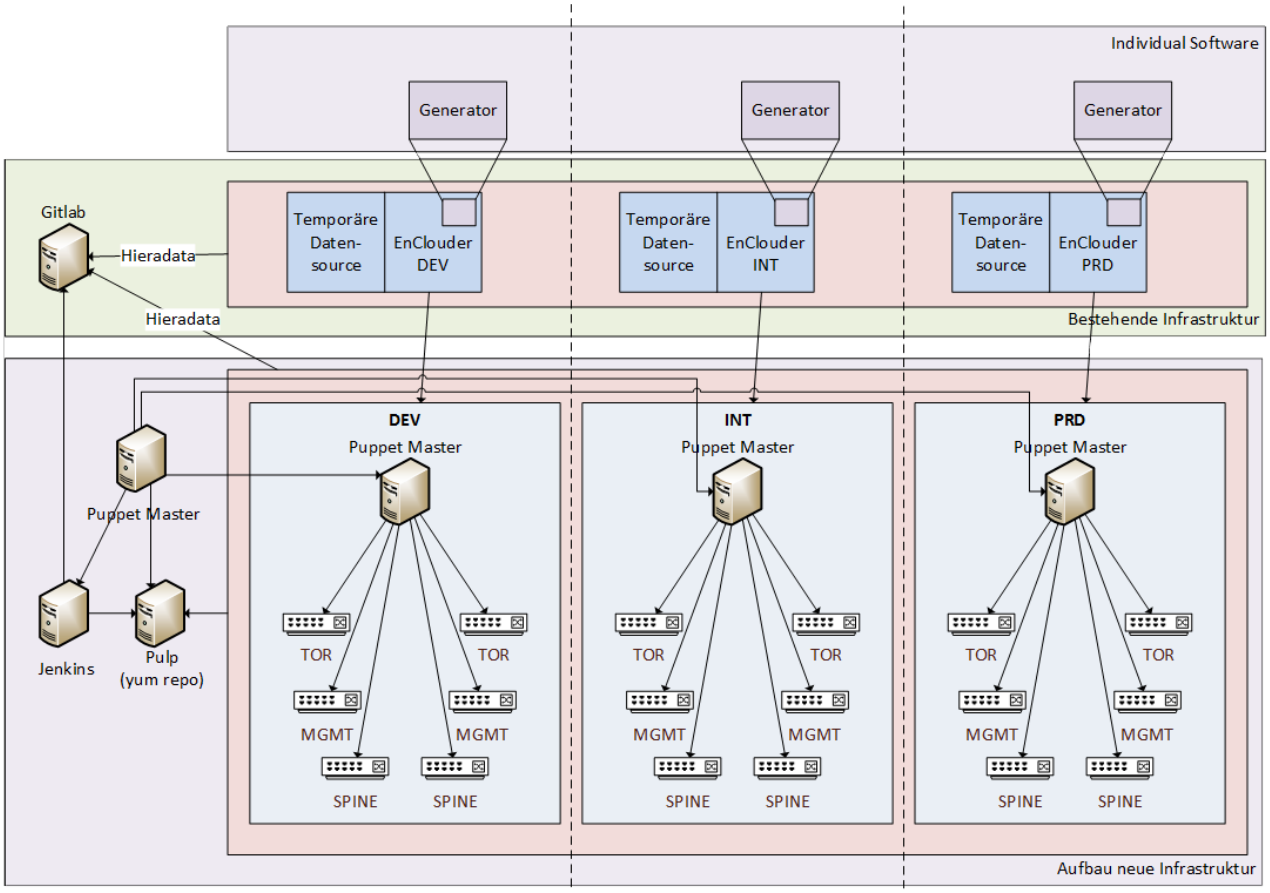


Abbildung 4: Systemlandschaft CI/CD Umgebung

Für die Automatisierung kommt Puppet mit Hiera - Produktebeschreibung im Kapitel 2.1 „Technologien und Produkte“ - zum Einsatz. Die Puppet Infrastruktur besteht aus vier Puppet Master. Diese verwalten die einzelnen virtuellen Switches. Einer der vier Puppet Master, der „Haupt Puppet Master“, verwaltet die Management Umgebung Pulp Server, Jenkins und auch die drei Puppet Master pro Stage (DEV, INT, PRD). Alle Systeme werden Puppet managed aufgesetzt. Das heisst, über die Hieradaten werden die einzelnen Nodes konfiguriert und über die generelle, ihnen zugeordnete Puppet Rolle, werden auf den Server und Switches alle nötigen Packages installiert, Files angelegt und Konfigurationen geschrieben, damit z.B. ein Jenkins Server entsteht. Die Puppet Roles existieren bereits

und können wiederverwendet werden. Folgende Roles kommen zum Einsatz:

Tabelle 2: Einzusetzende Puppet Roles

Komponente	Puppet Role	Puppet Service
Puppet Master	role_puppetmaster	puppet::master, git
Pulp Server	role_pulp	pulp
Jenkins	role_jenkins	jenkins
TOR Switch	role_arista_eos	nc_arista_eos
Mgmt Switch	role_arista_eos	nc_arista_eos
SPINE Switch	role_arista_eos	nc_arista_eos

4.2 Konfigurationsdatengenerator

Der Konfigurationsdatengenerator wird in Ruby geschrieben. Um die Funktionalitäten gemäss den User Storys zu implementieren werden verschiedene Klassen erstellt, die schlussendlich als Modul zusammengefasst werden und als Gesamtes den Konfigurationsdatengenerator ausmachen.

4.2.1 Funktionale Spezifikation

Die funktionale Spezifikation vermittelt ein Verständnis, welche Aufgaben die einzelnen Komponenten wahrnehmen und wie sie zusammenspielen. (Abbildung 5)

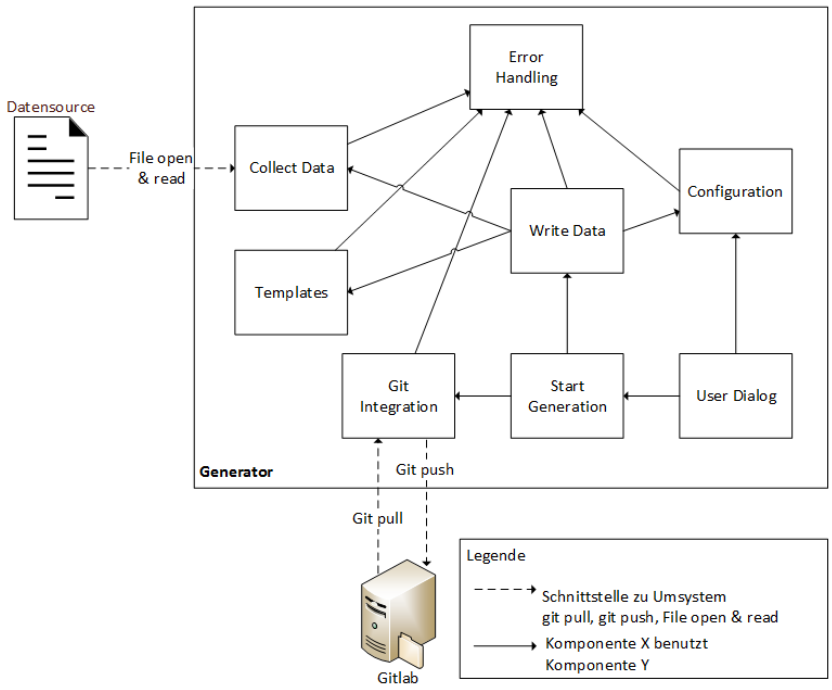


Abbildung 5: Komponenten Diagramm Konfigurationsdatengenerator

Collect Data

Collect Data gehört mit Write Data zum Kernstück des Konfigurationsdatengenerators. Die Klasse holt sich von einer Datensource die Konfigurationsdatenwerte. Die erhaltenen Werte werden in einem Hash zwischengespeichert, um im nächsten Schritt in Write Data weiterverarbeitet zu werden.

Write Data

Write Data hat den Schwerpunkt, ein Hieradaten-YAML-File aus den gewonnenen Werten mit Hilfe eines Embedded-Ruby-Templates (ERB) zusammenzustellen und mit dem korrekten Namen - dem FQDN des Switches - zu benennen. Der im Write Data integrierte Syntax Validator soll das neu erstellte YAML File auf seine Syntaxkorrektheit prüfen. YAML ist anspruchsvoll, was die Korrektheit der Syntax betrifft, bei Fehler im File, kann der Puppet Run später nicht durchlaufen. Um die Daten persistent zu speichern werden die Hieradaten Files mit einer zusätzlichen Git Integrationsklasse in ein Git Repository hochgeladen.

Templates

Die vordefinierten Templates werden in Embedded Ruby (ERB) geschrieben. Sie geben die Struktur des Hieradatenfiles vor und erhalten mit Hilfe von Iterationen über den Input-Value-Hash die benötigten Konfigurationswerte.

User Dialog

Der User Dialog wird als Commandline User Interface verwendet. Der Generator kann damit von einem Systemadministrator konfiguriert und angestossen werden.

Configuration

Die Konfiguration des Generators soll dem Endbenutzer/Systemadministrator die Möglichkeit geben, den Generator individuell verwenden zu können und ihn umgebungsspezifisch konfigurieren zu können. Hierbei können z.B. Repositories für die generierten Hieradaten angegeben werden.

Error Handling

Alle Komponenten verwenden das Modul Error Handling und schreiben ihre Fehler mit einem Zeitstempel in ein Logfile.

4.2.2 Benutzerinteraktion

Die Benutzerinteraktion gibt eine Übersicht, wie der Hieradatengenerator im Zusammenspiel mit dem Benutzer und der Infrastruktur funktioniert. (vgl. Abbildung 6)

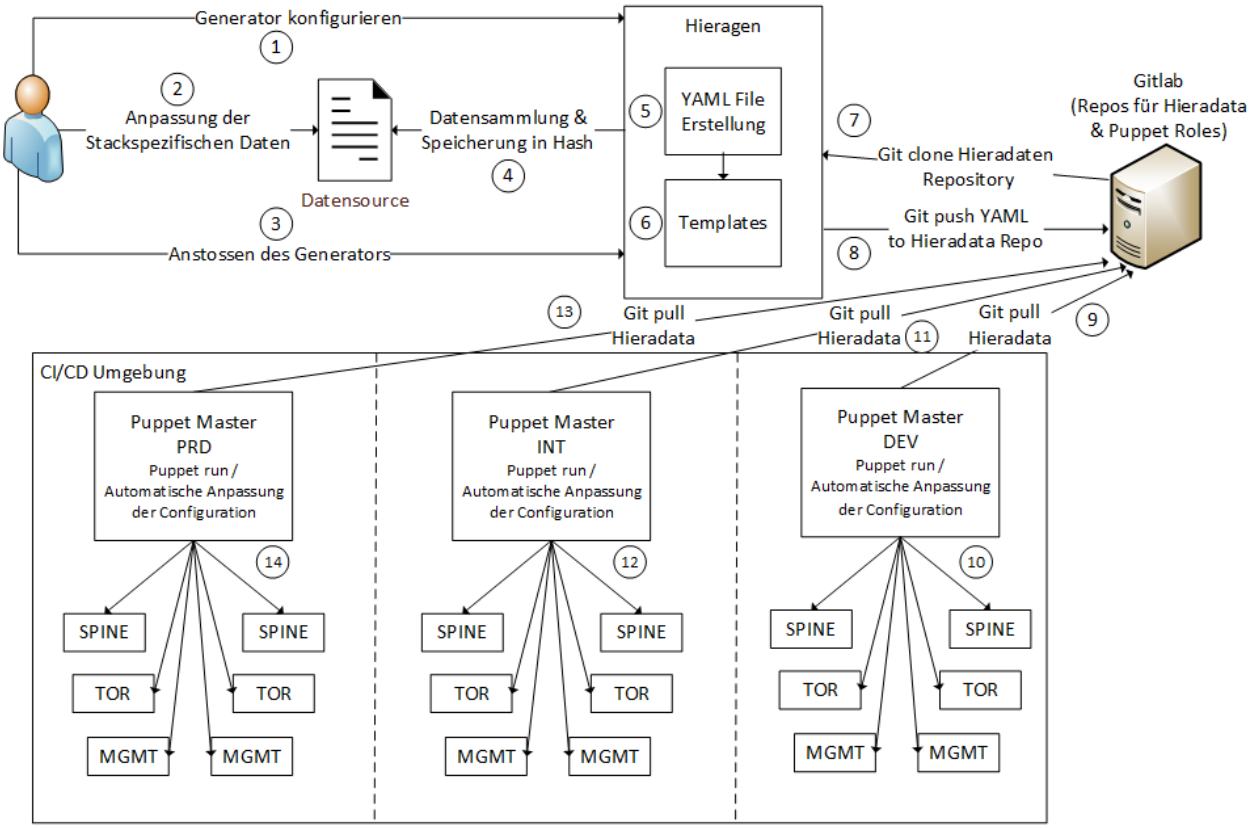


Abbildung 6: Benutzerinteraktion mit dem Konfigurationsdatengenerator

1. In einem ersten Schritt, nach der Installation des Generators auf dem Puppet Master nimmt der Benutzer die Konfiguration des Generators vor. Unter anderem werden dabei Source- und Output-Pfad festgelegt.
2. Der Benutzer kann eine Änderung der Stammdaten in der Datensource vornehmen oder die Datensource komplett neu anlegen.
3. Durch das Anstossen des Generators durch den Benutzer beginnt dieser mit der Generierung der Hieradaten.
4. Die Stammdaten aus der Datensource für Arista Switches werden im Generator in einem Hash zwischengespeichert.
5. Die YAML-File Erstellung beginnt.
6. Die Daten werden mit Hilfe von ERB-Templates zu syntaktisch korrekten YAML Files geschrieben.

7. Mit dem Upload der erstellten Files in das entsprechende Gitrepository werden die Daten persistent gespeichert und dem Puppet Master zur Verfügung gestellt.

8. Die Hieradaten sind nun bereit zum Deployment. Dazu muss vom jeweiligen Puppet Master mit einem „git pull“ das entsprechende Repository ausgecheckt werden. Die Hieradaten werden zuerst auf dem DEV Puppet Master updatet.

9. Die Änderungen werden auf DEV über einen Puppet run (manuell oder alle 30 Minuten automatisch) auf den einzelnen Arista Switches deployed.

10. Der Vorgang wird für INT (Schritte 10 und 11) und anschliessend für PRD (Schritte 12 und 13) wiederholt.

4.2.3 Schnittstellen

Die CI/CD Umgebung sowie der Konfigurationsdatengenerator nutzen Umsysteme über Schnittstellen. Dadurch können Synergien genutzt und Zeit gespart werden. (Abbildung 7)

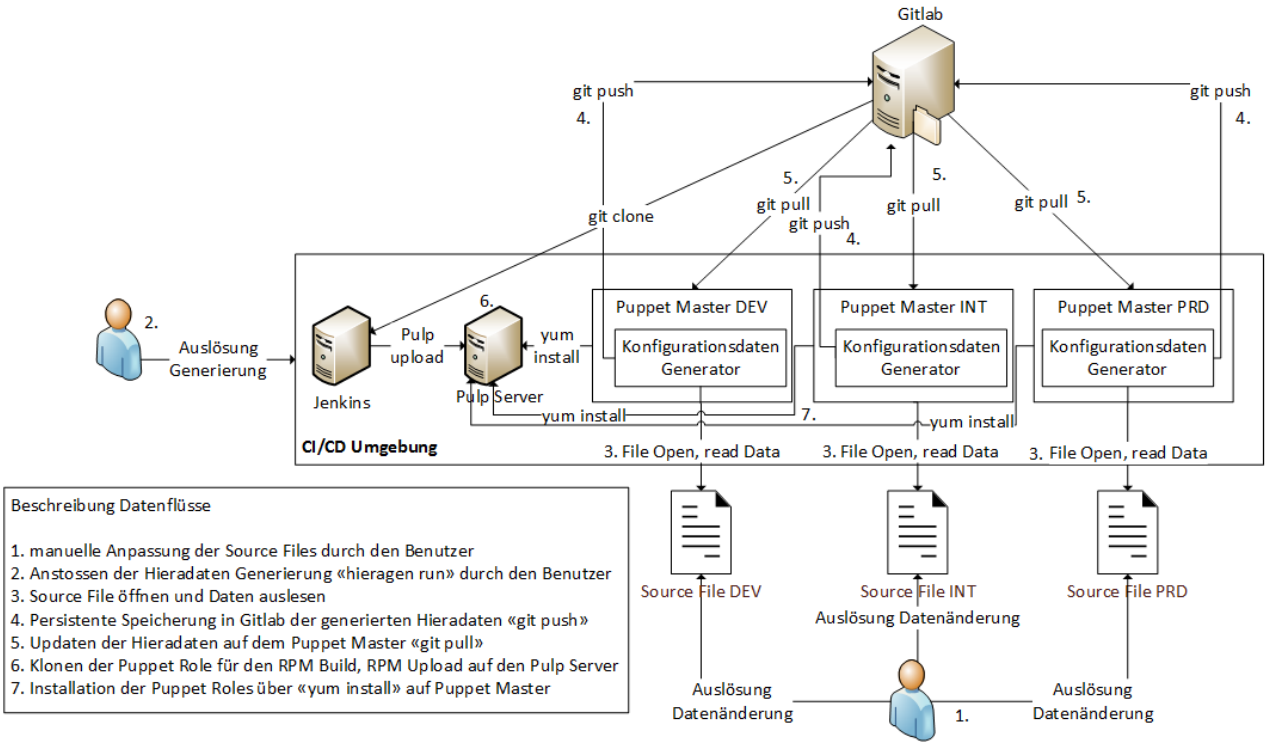


Abbildung 7: Schnittstellen zu Umsystemen

Die wichtigste Schnittstelle für die CI/CD Umgebung und den Generator ist das interne Gitlab. Dieses verwaltet alle Puppet Roles sowie die Hieradaten. Die vom Generator erstellen Hieradaten müssen zur persistenten Speicherung wieder ins Gitlab eingespielt werden. Der Puppet Master bezieht seine Puppet Roles via Jenkinsbuild über Pulp und installiert diese als RPM auf dem System.

Der Konfigurationsdatengenerator bezieht seine Daten vom EnClouder oder einer Textdatei als Fallback, falls Risiko Nummer vier (der EnClouder verfügt nicht über alle nötigen Daten für Arista) eintritt.

Zusätzlich besteht eine Schnittstelle zum Benutzer, der die Datensource verändert und den Generator steuert.

5. Implementation

Die Implementation gibt einen Einblick in die umgesetzten Konzepte und Implementierungen. Sie setzt sich aus den drei Subkapiteln 5.1 „CI/CD Umgebung“, 5.2 „Entwicklung Konfigurationsdatengenerator“ und 5.3 „Einbindung des Konfigurationsdatengenerators in die CI/CD Umgebung“ zusammen.

5.1 CI/CD Umgebung

Das Kapitel CI/CD Umgebung beschreibt die Vorbereitung und Zusammenstellung der Konfigurationsdaten, den Aufbau der einzelnen Server, sowie das Endresultat der eingesetzten Produkte.

5.1.1 Vorbereitung Hieradaten erstellen

Jeder Server bekommt seine eigenen Hieradaten, sodass die Puppet Role individualisiert verwendet werden kann. Die Hieradaten werden in einem YAML File pro Server hinterlegt und pro Puppet Master in ein Gitlab Repository abgelegt. Die nachfolgende Abbildung 8 zeigt, welche virtuelle Maschine / virtueller Switch an welchem Puppet Master hängt und aus welchen Repositories die Hieradaten bezogen werden.
Um einen Einblick in die Hieradaten Implementation zu erhalten, werden im Kapitel 5.1.4 Deployment Management Umgebung in den Unterkapitel der einzelnen Komponenten im Detail gezeigt und erklärt.
Alle, in den Hieradaten verwendeten Variablen, müssen in den einzelnen Puppet Roles oder Puppet Services bereits deklariert sein. Die Variablen werden durch den Hauptklassennamen, getrennt mit einem Doppelpunkt zum Subklassennamen, wieder getrennt durch einen Doppelpunkt und so weiter mit abschlissend dem Variablennamen angesprochen.
Bsp. `puppet::master::autoupdate::hiera_githost: „gitlab.internaldomain.ch“`
Jedes YAML-File startet mit der Role, die der entsprechende Host erhalten soll.

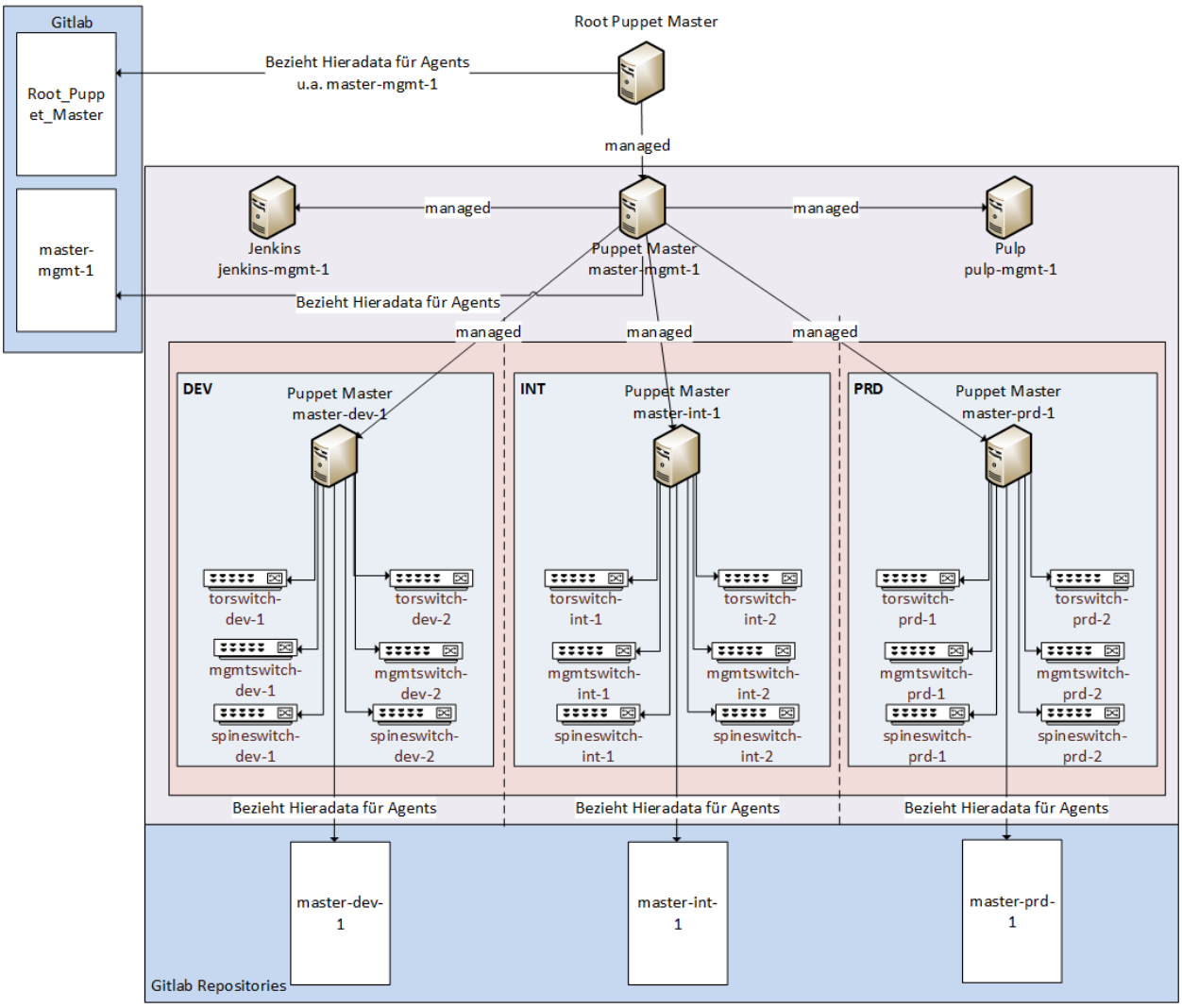


Abbildung 8: Zusammenhänge Puppet Master, Puppet Agents und Hieradatenrepositories

5.1.2 Infrastruktur bereitstellen

Für die Arbeit werden 24 virtuelle Maschinen verwendet. Diese werden in der Laborumgebung, basierend auf Red Hat Enterprise Virtualization (RHEVM), aufgebaut. Die Serverliste im Anhang A dieser Arbeit zeigt die verwendeten Hostnamen, sowie den zugehörigen Puppet Master. Als zusätzliche Information wird das Betriebssystem ausgewiesen.

5.1.3 Connectivity

Die CI/CD Umgebung wird in der existierenden Labor Umgebung für Puppet Testing aufgebaut. Die Connectivity zwischen diesen einzelnen Maschinen ist dadurch bereits gewährleistet. Die DNS Einträge, für die in dem Projekt spezifischen Nodes, wurden zusätzlich implementiert.

5.1.4 Deployment Management Umgebung

Als Basis für die CI/CD Umgebung, wird der „Haupt Puppet Master“ der Management Umgebung (master-mgmt-1.internaldomain.ch) mittels Puppet deployed.
Als zweiter Server wird der Pulp Repository Server in Betrieb genommen. Seine Repositories werden für die weiteren zu deployenden Server benötigt.
Alle Vorbedingungen für die restlichen Server sind erfüllt, das Deployment der drei restlichen Puppet Master für DEV, INT und PRD sowie der Jenkins kann gestartet werden.

Die virtuellen Maschinen für die Management Umgebung werden in RHEV manuell zusammengestellt und beim Puppet Master Mgmt registriert. Die nachfolgende Auflistung zeigt die einzelnen Schritte für das manuelle Setup der Management Umgebung.

1. Neue Virtuelle Maschine erstellen: Der Hostname wird gemäss der Serverliste im Anhang A gesetzt und dazu wird das RHEL 6.6 Standard Template von Swisscom für die Cloud Umgebungen verwendet. In diesem Template ist bereits Puppet, wie auch weitere Software installiert.
2. Als nächstes wird der Username und das Passwort für den Maschinenzugriff definiert.
3. Wichtig, die Netzwerkeinstellungen müssen bereits beim Erstellen der VM (in RHEV) vorgenommen werden, anderenfalls kann über SSH (Secure Shell, Verbindung über Port 22) nicht auf die virtuelle Maschine zugegriffen werden.
4. Vor dem Serverstart werden zusätzliche virtuelle Disks hinzugefügt. Je nach Server wird mehr oder weniger Diskspace gebraucht.

Sobald der Server lauffähig und über SSH erreichbar ist wird die Puppet Konfiguration auf dem Agent durchgeführt. Dabei bezieht dieser seine vordefinierte Puppet Role mit den zugehörigen Hieradaten.

Die Konfiguration der Agents erfordert verschiedene Schritte. Die nachfolgende Auflistung zeigt den Vorgang. Zu beachte ist: jeder Puppet Master ist wiederum ein Puppet Agent von seinem übergeordneten Puppet Master und deshalb muss die initiale Konfiguration auch auf dem Puppet Master vorgenommen werden.

1. Das File /etc/puppet/puppet.conf wird mit dem Puppet Master (Auflistung 1) ergänzt, wobei der Puppet Master nicht für jeden Agent gleich ist. Welcher Agent an welchem Puppet Master registriert ist kann der Spezifikation im Kapitel 4.1 entnommen werden. Das komplette Konfigurations-File ist im Anhang B1

Auflistung 1: Auszug aus der Puppet.conf Datei

```
server=master-mgmt-1.internaldomain.ch
report=true
```

2. Mit vorhergehender Konfiguration kann sich der Agent bei seinem Puppet Master registrieren. mit dem Command „puppet agent -t“ wird der Puppet Run auf dem Agent angestossen. Er löst beim ersten Run einen Zertifikatsrequest beim Puppet Master aus. (Auflistung 2)

Auflistung 2: Erster Puppet Run mit Cert Request

```
[root@jenkins-mgmt-1 ~]# puppet agent -t
Info: Creating a new SSL key for jenkins-mgmt-1.internaldomain.ch
Info: Caching certificate for ca
Info: csr_attributes file loading from /etc/puppet/csr_attributes.yaml
Info: Creating a new SSL certificate request for jenkins-mgmt-1.internaldomain.ch
Info: Certificate Request fingerprint (SHA256):
    1D:0B:C8:EE:53:D4:A1:97:CC:A1:9F:C8:9E:C5:5E:20:97:AA:D9:3A:02:A2:75:C0:6C:7F:BC:4B
Info: Caching certificate for ca
Exiting; no certificate found and waitforcert is disabled
```

3. Als nächstes wird der Zertifikatsrequest des Agents auf dem Puppet Master akzeptiert. Vorgängig kann mit dem Command „puppet cert list“ alle aktuellen Zertifikatsrequests abgefragt werden und mittels „puppet cert sign hostname“ wird der entsprechende Host gesigned. (Auszug in Auflistung 3)

Auflistung 3: Signing Cert Request auf dem Puppet Master

```
[root@master-mgmt-1 nodes]# puppet cert list
  "jenkins-mgmt-1.internaldomain.ch" (SHA256)
      1D:0B:C8:EE:53:D4:A1:97:CC:A1:9F:C8:9E:C5:5E:20:97:AA:D9:3A:02:A2:75:C0:6C:7F:BC:4B

[root@master-mgmt-1 nodes]# puppet cert sign jenkins-mgmt-1.internaldomain.ch
Notice: Signed certificate request for jenkins-mgmt-1.internaldomain.ch
Notice: Removing file Puppet::SSL::CertificateRequest jenkins-mgmt-1.internaldomain.ch at
    '/var/lib/puppet/sslmaster/ca/requests/jenkins-mgmt-1.internaldomain.ch.pem'
```

4. Mittels erneutem „puppet agent -t“ auf dem jeweiligen Agent wird die komplette Konfiguration bezogen, alle nötigen Packages installiert, Dateien erstellt und Services gestartet, sodass als Endresultat des Puppet Runs ein komplett funktionsfähiges System zur Verfügung steht.

5.1.5 Setup Puppet Master

Die erste Zeile der Hieradaten `sc_role: 'puppetmaster'` spezifiziert in jedem YAML-File die jeweilige Puppet Rolle die ein Server erhalten soll. Alle weiteren Hieradaten sind jeweils Role spezifisch und für den Puppet Master in Auflistung 4 abgebildet. Im nachfolgenden Text sind die einzelnen Variablen und deren Werte genauer erklärt. Die kompletten Hieradaten sind auf der CD im Anhang dieser Arbeit zu finden.

Auflistung 4: Hieradaten Puppet Master

```
1  sc_role: 'puppetmaster'
2
3  puppet::master::hiera_top_level_element:
4    'nodes/%%{}{clientcert}'
5
6  puppet::master::autoupdate::modules_repo: puppet-modules
7  puppet::master::autoupdate::modules: true
8  hiera::owner: puppet-hieradata
9
10 osbaseconfig::yum::pulp_repos:
11   puppet-modules:
12     descr: 'Puppet Modules'
13     baseurl: 'http://pulp-lab.internaldomain.ch/pulp/repos/puppet-module-lab/'
14   centos-scl:
15     descr: 'CentOS Software Collection'
16     baseurl: 'http://pulp-lab.jinternaldomain.ch/pulp/repos/centos-scl-6-$basearch-lab/'
17
18 puppet::master::certname: 'puppet-mgmt-1.internaldomain.ch'
19
20 httpd::loglv_size: '1G'
21 nc_mysql::datavg::size_data: '10G'
22 nc_mysql::datavg::size_backup: '15G'
23 puppet::master::datavg::dashboard_size: '5G'
24
25 nc_mysql::root_password: 'login2mysql'
26 nc_postgres::postgres_password: 'login2postgres'
27 puppet::master::puppetdb::postgres_password: 'login2puppetdb'
28 puppet::master::dashboard::mysql_password: 'login2dashboard'
29
30 puppet::master::hiera_deep_merge: true
31
32 # Je nach Puppet Master und den Anzuhängenden Agents können die Roles variiern
33 puppet::master::modules:
34   - role_base-lnx
35   - role_pulp
36   - role_jenkins
37   - role_puppetmaster
```

Die Zeilen 3 bis 8 definieren Variablen in der Puppet Role, die für das Konfigurieren und Auffinden der Hieradaten und den regelmässigen Updates der Puppet Roles zuständig sind. Ab Zeile 10 werden

mit `osbaseconfig:: yum::pulp_repos` die YUM Repositories des Puppet Masters definiert und beim Puppet run unter /etc/yum.repos.d als Repositories „puppet-modules.repo“ und „centos-scl.repo“ angelegt. Über den Package Manager YUM holt sich der Puppet Master (sowie auch die anderen eingesetzten Linux Server) ihre zu installierenden Packages. Die Puppet Role osbaseconfig ist eine, auf jedem Betriebssystem ausgerollte Puppet Role, die grundlegende Einstellungen des Betriebssystems konfiguriert. Mit dem `puppet::master::certname` wird der Alias des Puppet Masters hinterlegt, dieser werden die Agents zukünftig in ihre Konfigurationsdateien aufnehmen. Die Zeilen 20 bis 28 setzen Diskgrössen und Passwörter für den Puppet Master, und schlussendlich werden in der Zeile `puppet::master::modules` definiert, welche Puppet Roles auf dem Puppet Master für seine Agents installiert werden sollen. Für diese Projektarbeit besonders wichtig, die „role_arista_eos“ für die Arista Switches.

Jeder Puppet Master stellt ein Dashboard (vgl. Abbildung 9) zur Verfügung, welches alle registrierten Puppet Agents darstellt und der Status der einzelnen Puppet Runs ausweist. Mit Hilfe dieses Dashboards ist der Zustand der Agents auf einen Blick ersichtlich. Ein Nachteil des Dashboards ist, dass nur die jeweiligen Agents dieses Master angezeigt werden und keine Sub-Agents von untergeordneten Puppet Master ersichtlich sind. Es entstehen vier verschiedene Dashboards, eines der Management Umgebung (Abbildung 9) und jeweils eines pro Stage (DEV, INT, PRD) der CI/CD Umgebung.

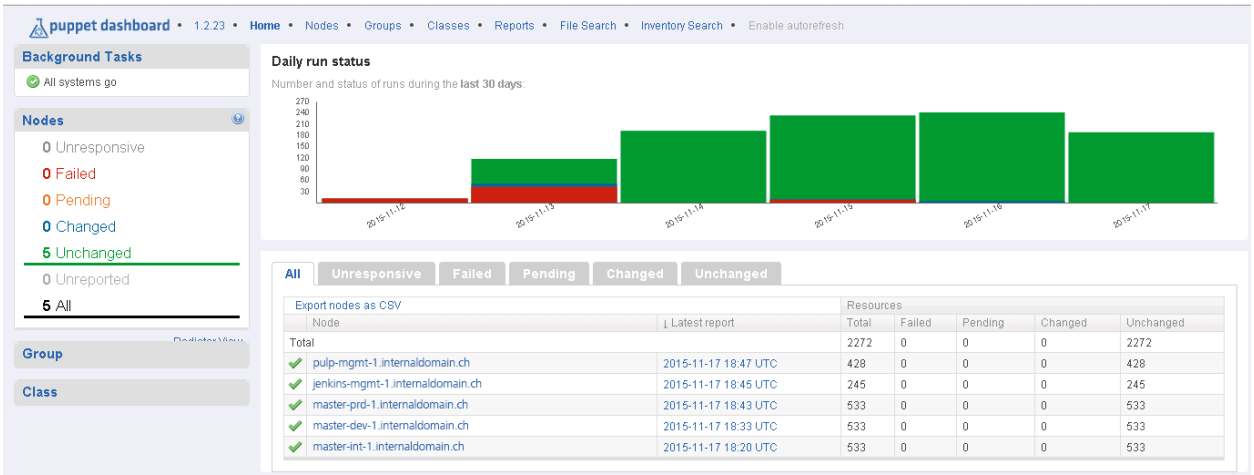


Abbildung 9: Puppet Dashboard Mgmt Umgebung

5.1.6 Setup Pulp Server

Das Setup des Pulp Servers erfordert eine zusätzliche Disk mit genügend Diskspace. Die zu synchronisierenden Repositories werden ihre RPM Pakete auf dem Pulp Server ablegen.

Die Hieradaten (komplette Datei auf der CD im Anhang dieser Arbeit) für den Pulp Server werden nach dem gleichen Schema wie die des Puppet Masters aufgebaut. Die Konfiguration in Auflistung 5 `pulp::merge_repositories:` ist besonders interessant. Hier werden alle Repositories aufgelistet, die auf

dem neuen Pulp Server zur Verfügung gestellt werden sollen. Sie werden von einem übergeordneten „Master-Pulp“ synchronisiert.

Auflistung 5: Hieradaten Pulp Server

```
pulp::merge_repositories:
  pulp-v2-dev:
    feed: 'http://pulp-lab.internaldomain.ch/pulp/repos/pulp-v2'
    interval: P1D
    retain_old_count: 5

  puppetlabs-products-6-x86_64-lab:
    feed: 'http://pulp-lab.internaldomain.ch/pulp/repos/puppetlabs-products-6-x86_64-lab'
    time: '05:30:00'
    retain_old_count: 5
```

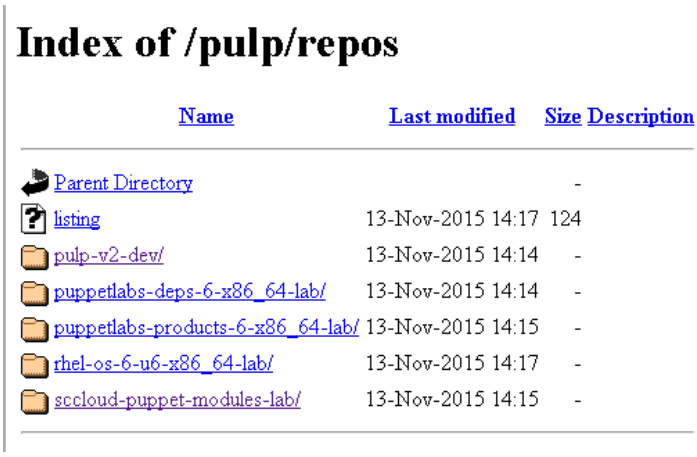
Der Pulp Server benötigt SSL Zertifikate und Keys. Weil die Instanz intern aufgebaut wird, können selfsigned SSL Zertifikate und Keys eingesetzt werden. Der komplette Vorgang, um die Keys und Zertifikate zu erstellen, ist im Anhang B2 genauer beschrieben.
Diese Zertifikate und Privat Keys werden abschliessend am Ende des YAML-Files gemäss Auflistung 6 in den Hieradaten hinterlegt.

Auflistung 6: Hieradaten Pulp Server

```
pulp::server::ssl::ssl_key: |
  -----BEGIN RSA PRIVATE KEY-----
  MIIEowIBAAKCAQEArlMQdcSCAgiN+nxjocdOaUMWktrb+bfuNHpeENJ0GR0Pf9hf
  UjN4EfKpJiD7uDYET8mw1zvEM7tAgCXslr2beaym3yeOywJwbwUMtdu4aVnr0TjE…

  pb7n+XCmO8zKy0Ne0vAdNRb60fdjaabKu9ZYttHingC5/zJ1w3Wf
  -----END RSA PRIVATE KEY-----
```

Der Pulp Server kann nach dem Deployment mit Puppet über den Webbrowser aufgerufen und die Repositories, ähnlich eines FTP Servers, mit den Paketen angezeigt und auch heruntergeladen werden. (Abbiludung 10)



Index of /pulp/repos

Name	Last modified	Size	Description
Parent Directory		-	
listing	13-Nov-2015 14:17	124	
pulp-v2-dev/	13-Nov-2015 14:14	-	
puppetlabs-deps-6-x86_64-lab/	13-Nov-2015 14:14	-	
puppetlabs-products-6-x86_64-lab/	13-Nov-2015 14:15	-	
rhel-os-6-u6-x86_64-lab/	13-Nov-2015 14:17	-	
sccloud-puppet-modules-lab/	13-Nov-2015 14:15	-	

Abbildung 10: Pulp Server Repositories

5.1.7 Setup Jenkins

Die Konfigurationsdaten des Jenkins (Auflistung 7) sind relativ simpel. Wichtig ist die Installation des Flavor „RPM“, sodass der Jenkins in der Lage ist RPM's aus dem Code des Konfigurationsdatengenerators zu builden und diese nach erfolgreichem Build auf den Pulp Server hochzuladen.

Auflistung 7: Hieradaten Jenkins

```
role_jenkins::flavors:
  - rpm
  - puppet

pulp::admin::version: '2.6.4'

jenkins::flavors::rpm::pulp_password: ''verysecret

jenkins::datavg::size: '7G'
jenkins::logrotate: '25'

jenkins::options:
  pulp_pwd: verysecret
```

Sobald der Jenkins erfolgreich deployed ist, kann er über den Webbrowser erreicht (vgl. Abbildung 11) und Jobs können konfiguriert werden. Für den Build des Konfigurationsdatengenerators wird im Kapitel „5.3 Einbindung des Konfigurationsdatengenerators in CI/CD Umgebung “ der zu erstellende

Buildjob genauer erklärt.

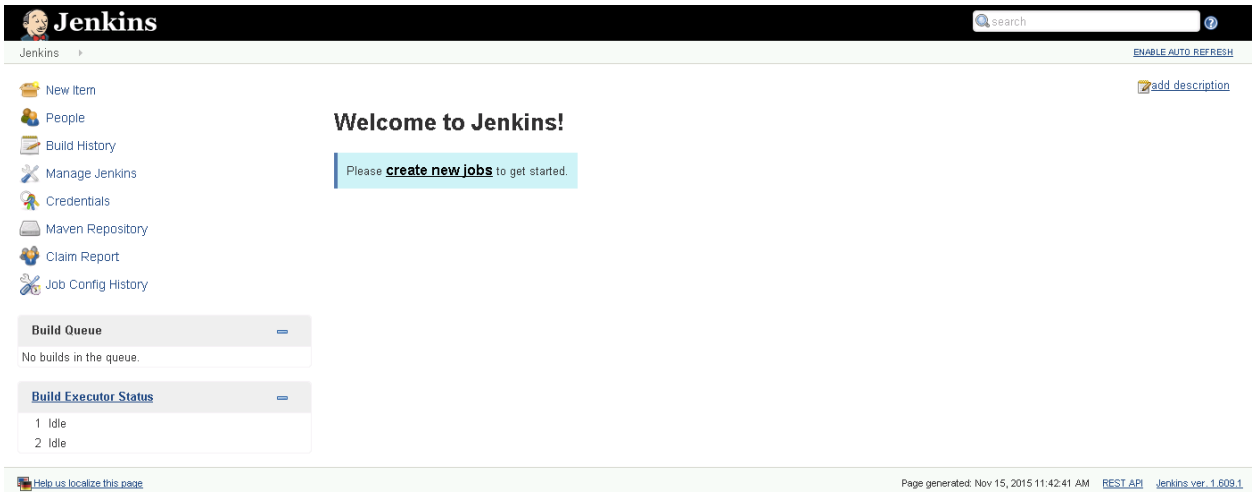


Abbildung 11: Jenkins Weboberfläche

5.1.8 Arista Switches

Die 18 virtuellen Arista Switches werden mit einem Deployment Plan über die API (application programming interface) des RHEV deployed. Dadurch können alle Switches automatisch innerhalb weniger Minuten aufgesetzt werden. Pro Umgebung (DEV, INT und PRD) existiert ein Plan im YAML-Format, welcher mit dem eigen entwickelten Tool DOPv (Deployment Orchestrator for Puppet) pro Puppet Master ausgeführt wird. Nachfolgende Sequenz (Auflistung 8) enthält einen Ausschnitt aus einem der Deployment Plans. Der komplette Plan für die Produktive Umgebung ist im Anhang B3 zu finden. Die Pläne für Development und Integration unterscheiden sich nur in Hostnamen und IP-Adressen von dem der Produktion. Deshalb werden diese Pläne dem Anhang nicht hinzugefügt, sind aber auf der CD am Ende dieser Arbeit zu finden.

Auflistung 8: Auszug Deployment Plan für Arista Switches

```
infrastructures:
  management:
    type: rhev
    endpoint: https://rhevm.internaldomain.ch/api
    credentials:
      username: rhevuser@internaldomain.ch
      password: topsecret

    networks:
      rhevm:
        ip_pool:
          from: 172.20.1.60
```

```
13              to: 172.20.1.80
14            ip_netmask: 255.255.255.0
15            ip_defgw: 172.20.1.1
16            dns:
17              nameserver:
18                - 172.20.1.12
19                - 172.20.1.13
20              domain: internaldomain.ch
21          puppettest_vms:
22            ip_pool:
23              from: 172.20.1.81
24              to: 172.20.1.102
25            ip_netmask: 255.255.255.0
26            ip_defgw: 172.20.1.1
27            dns:
28              nameserver:
29                - 172.20.1.12
30                - 172.20.1.13
31              domain: internaldomain.ch
32
33  nodes:
34    # Arista Switches TOR 1
35    torswitch-prd-1.internaldomain.ch:
36      infrastructure: management
37      infrastructure_properties:
38        datacenter: lab
39        cluster: lab
40      image: vEOS-4.14.2F
41      interfaces:
42        nic1:
43          network: puppettest_vms
44        nic2:
45          network: rhevm
46      cpu: 2
47      memory: 2G
```

Der effektive Run des DOPv Plans wird mittels `dopv deploy -p dop_plan_dev_aristaswitches.yaml` angestossen.

Als Voraussetzung, die Arista Switches als virtuelle Maschinen im Labor zu deployen, erfordert RHEV das Zusammenstellen und Importieren des Betriebssystem Templates. Dazu sind einige Zusatzschritte

notwendig, da Arista in erster Linie auf Virtualbox und VMWare für die Virtualisierung setzt. Arista stellt ein VMDK Disk Image (Virtual Machine Disk) und ein ISO Image (Archiv Datei einer optischen Disk) mit dem Bootloader zur Verfügung. Das Diskfile wird vom Format VMDK nach QCOW2 (Dateiformat für Disk Images) konvertiert und anschliessend mit dem ISO Image in RHEV importiert.
Die Aristaswitch Konfiguration (Auflistung 9) ist im Grundsatz für alle drei Switch-Arten sehr ähnlich und unterschiedet sich nur in kleinen Nuancen. Die Auflistung 9 zeigt einen Ausschnitt aus den Hieradaten beinhaltet aber bei weitem nicht die gesamte Konfiguration. Die kompletten Hieradaten für die Arista Switches sind auf der CD im Anhang dieser Arbeit zu finden.

Auflistung 9: Hieradaten Arista Switches

```
sc_role: arista_eos

# SNMP settings
nc_arista_eos::snmp_location: 'lab/dev/rack3'

# Interface settings
nc_arista_eos::interfaces:
  'Ethernet3':
    description: 'spineswitch-dev-1:Ethernet3'
    enable: true
    speed: 'default'
    lacp_priority: 32768

# Switchport settings
nc_arista_eos::switchports:
  'Ethernet1':
    mode: 'access'
    access_vlan: '4091'
  'Ethernet3':
    ensure: 'absent'

# IP Interface settings
nc_arista_eos::ip_interfaces:
  'Ethernet3':
    ensure: 'present'
    address: ,172.20.1.10/31'
    mtu: 1500

# OSPF Instance
nc_arista_eos::ospf_instances:
  '1':
    ensure: 'present'
    router_id: ,172.20.0.5'
    maximum_paths: '16'
    active_interfaces: ['Ethernet3']
```

```

# OSPF Networks
nc_arista_eos::ospf_networks:
  # Ethernet3,
  172.20.1.10/31':
    instance_id: '1'
    ensure: 'present'
    area: '0.0.0.0'
```

5.1.9 Konfiguration Arista Switches

Die Inbetriebnahme der 18 virtuellen Arista Switches erfordert Einiges an manueller Konfiguration bis diese schlussendlich über den Puppet Master ihre Konfiguration beziehen können.

In einem ersten Schritt erhalten die Switches eine IP-Adresse, sodass ein Login via SSH möglich ist. Zusätzlich wird ein User mit Passwort benötigt. Alle ausgeführten Commands sind im Anhang B4 enthalten.

Auflistung 10: Initiale Konfiguration der Arista Switches

```
#Angabe welches Interface konfiguriert werden soll
interface management 1

#IP Adresse für das Mgmt Interface 1 setzen
ip address 172.20.20.99/24

#IP Route und Gateway für das Mgmt Interface 1 setzen
ip route 0.0.0.0/0 172.20.20.1

#Neues Passwort für den Admin User setzen
username admin secret newpassword
```

Die Konfiguration kann überprüft werden, indem auf dem Switch `show run` ausgeführt wird. Für weiteres Troubleshooting kann `show ip interface brief` einen grossen Nutzen bieten. Das Command zeigt welchen Status das Interface hat.
Kann erfolgreich, z.B. vom Puppet Master über SSH auf den Switch, eingeloggt werden, war der erste Schritt der Konfiguration erfolgreich.

Um Puppet auf Arista Switches einzusetzen braucht es weitere Interaktionen. Der DNS Server, sowie der Hostname des Switches, müssen gesetzt werden, dazu alle Commands im Anhang B5. Die Installation von Puppet erfolgt über sogenannte Extensions (Auflistung 11), die auf dem Switches installiert werden.

Auflistung 11: Installation von Puppet auf den Arista Switches

```
#Kopieren der Ruby, Puppet und EOS Puppet Konfiguration Extension
copy http://ztp.internaldomain.ch/files/extensions/ruby-1.9.3-1.swix
    extension:ruby-1.9.3-1.swix
copy http://ztp.internaldomain.ch/files/extensions/puppet-3.7.1-3-ruby1.swix
    extension:puppet-3.7.1-3-ruby1.swix
copy http://ztp.internaldomain.ch/files/extensions/eos-puppet-conf-0.1-2.eos4.noarch.rpm
    extension:eos-puppet-conf-0.1-2.eos4.noarch.rpm

#Anzeigen der vorhandenen Extensions, diese sind noch nicht installiert
show extensions

#Installation der Extensions
extension ruby-1.9.3-1.swix
extension puppet-3.7.1-3-ruby1.swix
extension eos-puppet-conf-0.1-2.eos4.noarch.rpm

#Persistieren der Installation, sprich, bei jedem Bootvorgang werden die Extensions neu installiert.
copy installed-extensions boot-extensions

#Kontrolle ob Extensions installiert wurden
show boot-extensions
```

Sobald die Extensions installiert sind, folgen die nächsten Schritte der Konfiguration. (Auflistung 12)

Auflistung 12: Installation von Puppet auf den Arista Switches

```
#Installation der eapi.conf
http://ztp.internaldomain.ch:8080/files/eapi.conf flash:eapi.conf

#Eintreten in den Konfigurations Modus
conf terminal

#Puppet Konfiguration erstellen
alias puppet bash sudo puppet agent --server puppet-dev-1.internaldomain.ch --confdir
      /persist/sys/puppet --daemonize --no-onetime
!
event-handler puppet-agent-on-startup
    trigger on-boot
    action bash sudo puppet agent --server puppet-dev-1.internaldomain.ch --confdir
          /persist/sys/puppet --daemonize --no-onetime
    delay 60
```

```
14 !
15 End
```

Zusätzlich bietet Arista für ihre Switches eine Command API, die pro Switch, über den Webbrowser (gemäss Abbildung 12) aufgerufen werden kann.

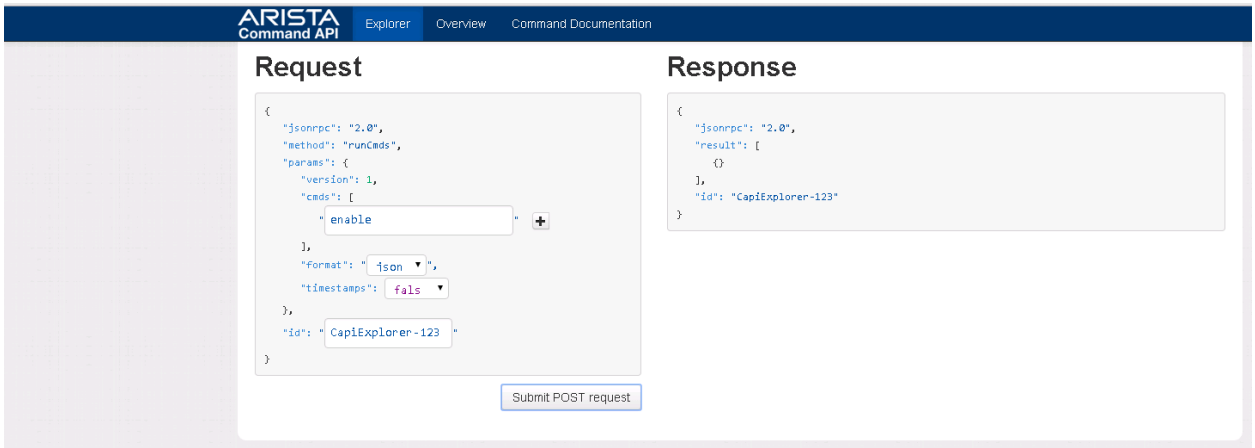


Abbildung 12: Arista Command API

5.2 Entwicklung Konfigurationsdatengenerator „hieragen“

Der Konfigurationsdatengenerator wird in Ruby geschrieben. Ruby ist eine objektorientierte Programmiersprache, "die sich einfach anwenden und produktiv einsetzen lässt."(Ruby-Community 2015) Als Entwicklungsumgebung kommt IntelliJ mit entsprechenden Ruby Plugins zum Einsatz. Das Testingframework für die Unittests heisst Rspec. Die effektive Implementation des Konfigurationsdatengenerator weicht leicht vom Design ab. Insbesondere der Fakt, dass die Datensource EnClouder nicht eingesetzt werden kann, erfordert einige Anpassungen an der Datensource Schnittstelle und demnach auch an den implementierten Methoden.

5.2.1 Klassendiagramm

Das Klassendiagramm (Abbildung 13) - nach der UML (Unified Moddelling Language) Notation erstellt - gibt eine Übersicht, welche Klassen, Methoden und Instanzvariablen in der Software „Hieragen“ existieren und wie deren Beziehungen untereinander sind.

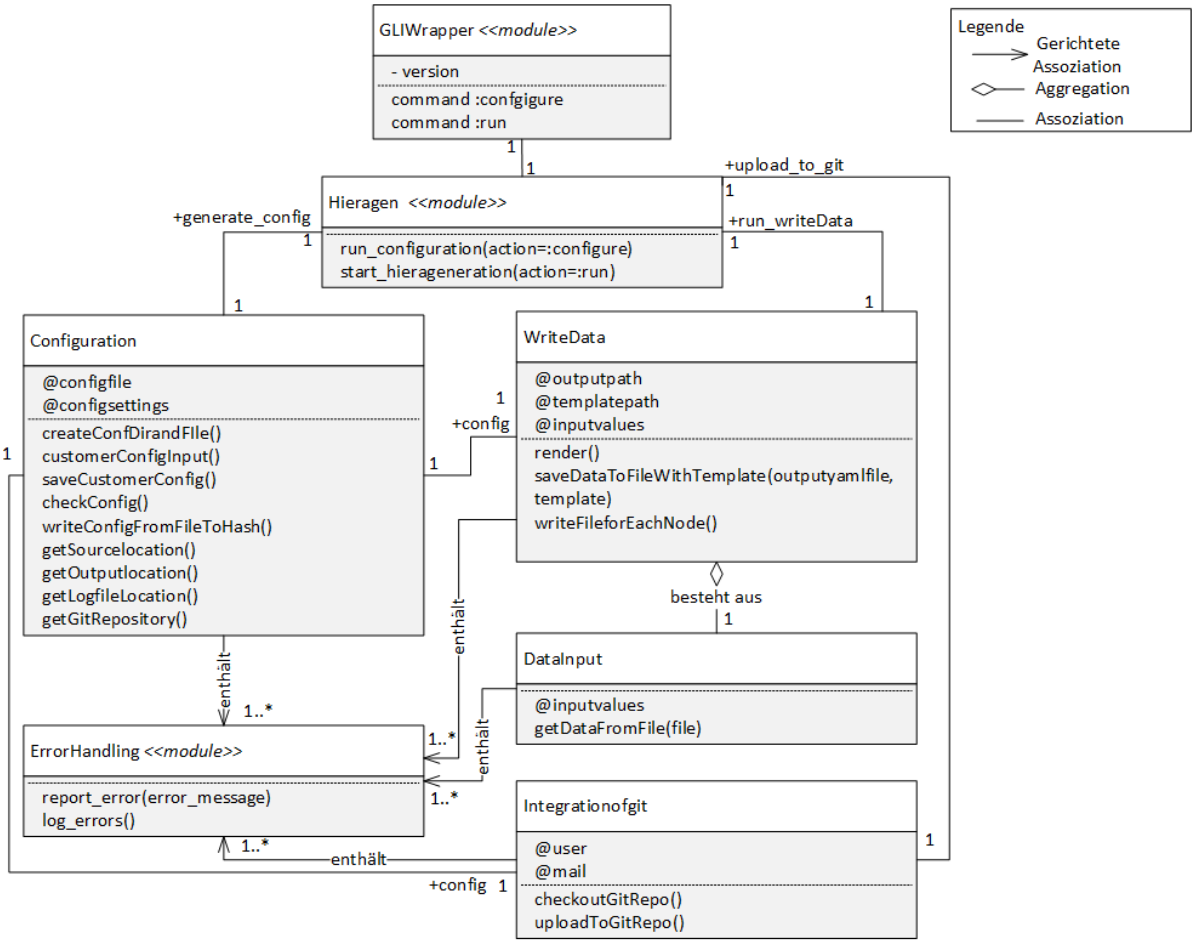


Abbildung 13: Klassendiagramm Hieragen

5.2.2 Funktionen des Generators

Der Generator, als Modul „hieragen“ zusammengefasst, umfasst vier Klassen. Diese Klassen nutzen zwei weitere Module („GLIWrapper“ und „ErrorHandling“) Die Hauptfunktionen des Generators werden im nachfolgenden Absatz genauer erklärt. Der komplette Code des Generators ist auf einer CD am Ende dieser Arbeit beigelegt.

Commandline Interface

Das Commandline Interface ist die Schnittstelle zum Endbenutzer. Mit Hilfe dessen kann der Konfigurationsdatengenerator auf einfache Art und Weise konfiguriert und bedient werden. Um Command Line Interfaces zu erstellen eignet sich das RubyGem (Paketsystem für Ruby) GLI (Git-Like Interface Command Line Parser) bestens. Neben dem Definieren des Programmnamens können verschiedene Commands mit GLI festgelegt werden, die zur Laufzeit verschiedene Methoden des Generators aufrufen. Der Generator „Hieragen“ hat zwei Funktionen (Auflistung 13) - „Configuration“, zur Kon-

figuration des Generators und „Run“, zum Anstossen der Hieradaten Generierung.

Auflistung 13: Auszug GLI Interface Commands

```
desc 'Configure HieraGen and set source and destination path'
command :configure do |c|
  c.action do |global_options,options,args|
    ::Hieragen::run_configuration(:configure)
  end
end

desc 'Run HieraGen, Collect Data and write it to a file'
command :run do |c|
  c.action do |global_options,options,args|
    ::Hieragen::start_hierageneration(:run)
  end
end
```

Zusätzlich bietet GLI dem Benutzer eine Versions und Hilfe Funktion. Screenshots der einzelnen Funktionen sind im Kapitel 5.4 „Einsatz des Generators“ abgelegt.

Configuration

Das Modul Configuration setzt die initiale Konfiguration des Generators vor dem ersten Run. Als Erstes wird beim Instanzieren der Klasse „Configuration“ die Methode checkConfig() aufgerufen, sie prüft, ob bereits eine Konfigurationsdatei existiert. Bei einer existierenden Konfiguration kann der User entscheiden, ob er diese behalten oder neu generieren will. Bei keiner vorhandenen Konfigurationsdatei wird der User direkt zum Input Dialog der Methode customerConfigInput() geführt. Über die Commandline Eingabe durch den User werden Sourcelocation der Inputdaten, der Logfilepfad, die Outputlocation der generierten Hieradaten, sowie das Git Repository gesammelt. Diese Werte werden in der Konfigurationsdatei unter /etc/hieragen/hieragen.conf gespeichert. Mit der Methode writeConfigFromFileToHash() werden die, in der Konfigurationsdatei niedergeschrieben Konfigurationswerte, zu einem späteren Zeitpunkt wieder ausgelesen und in einem Hash zwischengespeichert. Die Getter-Methoden ermöglichen den Zugriff auf diese Werte von anderen Klassen.

DataInput

Die Klasse DataInput ist klein gehalten. Die Instanzmethode initialize() erstellt ein Objekt der Klasse Configuration und speichert mittels „config.writeConfigFromFileToHash()“ die Konfigurationsdaten in einem Hash zwischen. Die Hauptmethode getDataFromFile(file) (Auflistung 14) liest die Sourcedatei (Lokation aus der Konfigurationsdatei bekannt) aus und gibt als Return-Wert alle Daten als Hash @inputvalues zurück.

Auflistung 14: Auszug DataInput Klasse

```
def getDataFromFile(file)
    begin
        if File.exist?(file)
            puts "Starting Data Collection ..."
            inputfile = File.read(file)
            @inputvalues = JSON.parse(inputfile)
            puts "Data Collection successfully finished..."
        else
            message = 'Input File does not exist. Please check the Path in the Config File.'
            ErrorHandling.report_error(message)
        end
        return @inputvalues
    rescue => error
        ErrorHandling.report_error("#{error.class} and #{error.message}")
    end
end
```

WriteData

Über die initialize() Methode werden - wie bereits in der Klasse DataInput - die Konfigurationswerte aufgerufen. Zusätzlich wird in dieser Methode ein Objekt der Klasse DataInput erzeugt und dadurch mit dem Aufruf der Methode getDataFromFile(file) die @inputvalues zurückgeliefert. Im nächsten Schritt werden die verschiedenen YAML-Files mit den korrekten Dateinamen (FQDN.yaml) angelegt und mit Hilfe von ERB-Templates (existieren pro Switch-Art) mit dem richtigen Inhalt befüllt. Auszug aus der Klasse WriteData in Auflistung 15.

Auflistung 15: Auszug WriteData Klasse

```
def saveDataToFileWithTemplate(outputyamlfile, template)
    begin
      @template = File.read(template)
      puts "Starting save Data for #{outputyamlfile}..."

      configyaml = YAML::load_file(outputyamlfile)

      File.open(outputyamlfile, "w") do |f|
        f.write(render()).to_yaml
      end
    rescue => error
      ErrorHandling.report_error("#{error.class} and #{error.message}")
    end
end
```

Templates
Templates - in Embedded Ruby geschrieben - sind keine eigentlichen Klassen, werden aber von WriteData verwendet, um die Hieradaten für die einzelnen Puppet Roles zu schreiben. Mit Hilfe dieser Templates erübrigt sich der Schritt, die Puppet Variabeln aus den einzelnen Roles zusammenzutragen. Bei Anpassungen der Variablen wird das jeweilige Template abgeändert und über den Buildjob (mehr dazu im Kapitel 5.3 „Einbindung des Konfigurationsdatengenerators in CI/CD Umgebung“) ausgerollt.

Integrationofgit
Die Klasse Integrationofgit ist für die komplette Interaktion mit Git verantwortlich. Die Methode checkoutGitRepo() führt ein „git clone http://gitlab.internaldomain.ch/.....git“ des Hieradaten Repositories durch oder bei einem bereits existierenden Repository ein „git pull“, um die Daten zu aktualisieren. Die Methode uploadToGitRepo() speichert nach erfolgreicher Erstellung der Hieradaten diese in einem Git Branch mit dem Namen „generation“ und lädt diese mit „git push“ auf Gitlab.

Errorhandling
Das Errorhandling hat zwei Methoden:
„report_error(error_message)“, sie nimmt die Error Messages der Klassen entgegen und speichert die Error Meldungen in einer Variable zwischen.
„log_errors()“ schreibt schlussendlich die effektive Meldung mit dem aktuellen Zeitstempel in das vordefinierte Errorlogfile. Die Verwendung des Errorhandlings in anderen Klassen ist in Auflistung 14 und 15 ersichtlich.

5.2.3 Unit Testing mittels Rspec

Fürs Unit Testing wird das Framework Rspec eingesetzt. Zu jeder Klasse des Generators existiert eine entsprechende Testklasse. Die Testklassen sind mit „klassenname_spec.rb“ benannt. Die konkrete Implementation der Tests und deren Resultate werden im Kapitel 6.1.2 „Testing Konfigurationsdatengenerator“ beschrieben.

5.3 Einbindung des Konfigurationsdatengenerators in CI/CD Umgebung

Der Konfigurationsdatengenerator „hieragen“ wird auf den Puppet Master DEV, INT und PRD installiert, um von da aus die entsprechenden Hieradaten generieren zu können. Die Software Hieragen muss, um sauber installiert werden zu können, als RPM paketiert und auf den Repository Server Pulp abgelegt werden. Für die Paketierung wird der Buildserver Jenkins eingesetzt. Die Software wird zu zwei verschiedenen RPM’s paketiert. Das eine Paket beinhaltet die effektive Software und das andere alle ERB-Templates.

Die Jenkinsjobs für den Build dieser Artefakte werden mit Hilfe des „Job DSL Plugin“ (DSL = Domain Specific Language) über Scripts automatisch auf dem Jenkins erstellt. Als Erstes wird er sogenannte

„Seed-Job“ mit dem Namen „Auto-Generate-Jenkins-Jobs“ über das Script DSL-rpm.groovy im Jenkins angelegt. Dieser Job hat zur Funktion, bei seinem Run, die Subjobs zu kreieren. Es sollen drei Jobs - „jenkins-rpm-gembundle_hieragen“ für die Hieragen Softwarepaketierung, „jenkins-rpm-rpm-hieragentemplates“ für die RPM Paketierung der Templates und einen Job fürs automatisierte RSpec Testing „hieragen-testing“ - erstellt werden. Abbildung 14 zeigt die automatisch generierten Jobs.

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration
		Auto-Generate-Jenkins-Jobs	13 hr - #42	10 days - #39	5.2 sec
		hieragen-testing	6.9 sec - #20	1 hr 57 min - #13	2.7 sec
		jenkins-rpm-gembundle_hieragen	3 days 11 hr - #39	10 days - #38	1 min 26 sec
		jenkins-rpm-rpm-hieragentemplates	13 hr - #1	N/A	4.4 sec

Abbildung 14: Jenkins Buildjobs

Alle Scripts werden in einem Gitrepository verwaltet. Der Jenkinsjob holt sich von diesem Repository die benötigten Informationen. Es existieren drei generelle Scripts (DSL-gembundle.list, DSL-rpm.list und DSL-testing.list) für die drei Aufgaben, wobei das RPM Script mehrfach verwendet wird. Alle Details für den Build werden in Ordner mit entsprechenden Scripts geführt. Die zwei Buildjobs speichern nach erfolgreichem Build auf dem Jenkins ihre generierten Pakete auf dem Pulp Server. Der Testingjob gibt das Resultat der RSpec Jobs zurück. Die kompletten Scripts für die Buildjobs sind auf der CD am Ende dieser Arbeit zu finden.

5.4 Einsatz des Generators

Die vier Subkapitel 5.4.1 „Installation“, 5.4.2 „Konfiguration“, 5.4.4 „Ausführung / Generierung von Hieradaten“ und 5.4.3 „Weitere Funktionen“ beschreiben den Einsatz der, in dieser Arbeit entwickelten Software „hieragen“.

5.4.1 Installation

Der Generator wird auf den drei umgebungsspezifischen Puppet Master (DEV, INT und PRD) installiert. Die als RPM paketierte Software „hieragen“ lässt sich mit dem YUM Package Manager via „yum install ruby193-hieragen“ installieren.

5.4.2 Konfiguration

Bevor „hieragen“ eingesetzt werden kann, muss die initiale Konfiguration durchgeführt werden. Dazu wird die integrierte Funktion „configure“ genutzt. Dabei wird der User über einen Commandline-Dialog aufgefordert, die Sourcelocation für die Input Daten, der Logfile Pfad, die Outputlocation für

die generierten Daten, sowie das Gitrepository für die persistente Speicherung der erstellten Hieradaten, anzugeben. Die Abbildung 15 zeigt den Ablauf der Konfiguration des Generators. In Gelb sind die Eingaben des Users zu sehen.

```
[root@master-int-1 ~]# hieragen configure
Please do the configuration first
Please enter the sourcelocation:
/etc/hieragen/input/testdata_int.json
Please enter the logfilepath:
/var/log/
Please enter the outputlocation:
/etc/hieragen/output/
Please enter the gitrepository:
https://gitlab.internaldomain.ch/rafaela.wuest/master-int-1.git
Your sourcelocation is /etc/hieragen/input/testdata_int.json
 logfilepath is /var/log/
 outputlocation is /etc/hieragen/output/
 gitrepository is https://gitlab.internaldomain.ch/rafaela.wuest/master-int-1.git


##############################
successfully saved to file.
see /etc/hieragen/hieragen.conf
##############################
```

Abbildung 15: Hieragen Konfiguration

5.4.3 Weitere Funktionen

„Hieragen“ verfügt über zwei weitere Funktionen: zum Einen die Versions-Abfrage und zum Anderen das Hilfemenü.

Die Versions-Abfrage „hieragen –version“ gibt die aktuell installierte Version der Software zurück:

```
[root@master-int-1 ~]# hieragen --version
hieragen version 0.2.2
[root@master-int-1 ~]#
```

Abbildung 16: Hieragen Version

Das Hilfemenü „hieragen –help“ zeigt dem Benutzer, über welche Funktionen „Hieragen“ verfügt:

```
[root@master-int-1 ~]# hieragen --help
NAME
    hieragen - HieraGen command line tool

SYNOPSIS
    hieragen [global options] command [command options] [arguments...]

VERSION
    0.2.2

GLOBAL OPTIONS
    --help    - Show this message
    --version - Display the program version

COMMANDS
    configure - Configure HieraGen and set source and destination path
    help      - Shows a list of commands or help for one command
    run       - Run HieraGen, Collect Data and write it to a file
```

Abbildung 17: Hieragen Help

5.4.4 Ausführung / Generierung von Hieradaten

Sobald die initiale Konfiguration erfolgreich vorgenommen wurde, kann der Generator mittels „hieragen run“ ausgeführt werden. Die Abbildung 18 zeigt einen erfolgreichen Durchlauf der Hieradaten Generierung.

```
[root@master-int-1 ~]# hieragen run
Starting Data Collection ...
Data Collection successfully finished...
Starting YAML File Creation ...
Starting save Data for /etc/hieragen/output/nodes/torswitch-int-1.yaml...
Starting render and load YAML ...
Starting save Data for /etc/hieragen/output/nodes/torswitch-int-2.yaml...
Starting render and load YAML ...
Starting save Data for /etc/hieragen/output/nodes/spineswitch-int-1.yaml...
Starting render and load YAML ...
Starting save Data for /etc/hieragen/output/nodes/spineswitch-int-2.yaml...
Starting render and load YAML ...
Starting save Data for /etc/hieragen/output/nodes/mgmtswitch-int-1.yaml...
Starting render and load YAML ...
Starting save Data for /etc/hieragen/output/nodes/mgmtswitch-int-2.mgmt.yaml...
Starting render and load YAML ...
Process finished...
Successfully Uploaded...
[root@master-int-1 ~]#
```

Abbildung 18: Hieragen Run

6. Testing

Das Testing wird in drei Etappen realisiert. In der ersten Etappe wird die CI/CD Umgebung auf ihre Funktionalität geprüft, die zweite Etappe umfasst das komplette Testing des Konfigurationsdatengenerators und in der finalen, dritten Etappe werden Systemintegrationstests zwischen Generator und CI/CD Umgebung durchgeführt. Das Kapitel gliedert sich in zwei Unterkapitel. Kapitel 6.1 „Testfälle“ beschreibt die einzelnen Test Cases im Detail. Diese basieren hauptsächlich auf denen in Kapitel 3. „Requirements“ spezifizierten User Stories und Use Cases. Das zweite Kapitel 6.2 „Testresultate“ protokolliert alle ausgeführten Tests mit dem erhaltenen Endresultat.

6.1 Testfälle

Die nachfolgenden Testfälle werden möglichst genau spezifiziert, sodass diese auch von einer, nicht im Projekt involvierten Person, durchgeführt werden könnten.

6.1.1 Testing CI/CD Umgebung

Die Tests für die CI/CD Umgebung werden manuell von einer Person ausgeführt. Mit Hilfe der nachfolgen spezifizierten Tests kann eine Aussage gemacht werden, ob das System lauffähig ist. Die Testfälle haben jeweils eine Referenz (Requirements Ref.) auf die zugehörigen Use Cases. Die Test-ID *_CICD referenzieren auf die Testcases für die CI/CD Umgebung.

Tabelle 3: Testfälle CI/CD Umgebung

ID	1_CICD
Titel	Full Puppet Managed Puppet Master in der CI/CD Umgebung
Komponente	CI/CD Umgebung
Requirements Ref.	Use Case 1: Fully Puppet managed CI/CD environments, Use Case 2: Artifact installation over RPM’s
Beschreibung	Die vier Puppet Master (Management, DEV, INT und PRD) sollen komplett mit Puppet gemanagt werden. Das heisst, es sind nach dem ersten Puppet Run keine weiteren Interaktionen nötigt um die Puppet Master zur Lauffähigkeit zu bringen.
Vorgehen	Die Voraussetzungen für den Test müssen berücksichtigt werden. 1. Login auf den Management Puppet Master 2. „puppet agnet -t“ ausführen Diese zwei Schritte werden für alle Puppet Master wiederholt.
Voraussetzung	Installation von Puppet, initiae Konfiguration (Puppet.conf) und Hieradaten gemäss CD im Anhang erstellt. Akzeptierter Zertifikatsrequest auf dem übergeordneten Puppet Master

Erwartetes Resultat	Erfolgreicher Puppet run ohne Fehler, Server (Puppet Master) erscheint im Puppet Dashboard seines Puppet Masters, und das Dashboard des neu erstellten Puppet Masters kann über seine URL aufgerufen werden. (http://master-mgmt-1.internaldomain.net/dashboard, http://master-dev-1.internaldomain.net/dashboard, http://master-int-1.internaldomain.net/dashboard, http://master-prd-1.internaldomain.net/dashboard)

ID	2_CICD
Titel	Full Puppet Managed Pulp Server in der CI/CD Umgebung
Komponente	CI/CD Umgebung
Requirements Ref.	Use Case 1: Fully Puppet managed CI/CD environments, Use Case 2: Artifact installation over RPM's
Beschreibung	Der Pulp Server soll komplett mit Puppet gemanagt werden. Das heisst, es sind nach dem ersten Puppet Run keine weiteren Interaktionen nötigt, um das System zur Lauffähigkeit zu bringen.
Vorgehen	Die Voraussetzungen für den Test müssen berücksichtigt werden. 1. Login auf den Pulp Server 2. „puppet agnet -t" ausführen
Voraussetzung	Installation von Puppet, initiae Konfiguraition (Puppet.conf) und Hieradaten gemäss CD im Anhang erstellt. Akzeptierter Zertifikatsrequest auf dem Puppet Master Management
Erwartetes Resultat	Erfolgreicher Puppet run ohne Fehler, Pulp Server erscheint im Puppet Dashboard des Management Puppet Masters, und seine Repositories können über die Weboberfläche aufgerufen werden. (http://pulp-mgmt-1.internaldomain.net/pulp/repos)

ID	3_CICD
Titel	Full Puppet Managed Jenkins in der CI/CD Umgebung
Komponente	CI/CD Umgebung
Requirements Ref.	Use Case 1: Fully Puppet managed CI/CD environments, Use Case 2: Artifact installation over RPM's
Beschreibung	Der Jenkins soll komplett mit Puppet gemanagt werden. Das heisst, es sind nach dem ersten Puppet Run keine weiteren Interaktionen nötigt um das System zur Lauffähigkeit zu bringen.
Vorgehen	Die Voraussetzungen für den Test müssen berücksichtigt werden. 1. Login auf den Jenkins Server 2. „puppet agnet -t" ausführen

Voraussetzung	Installation von Puppet, initiae Konfiguraition (Puppet.conf) und Hieradaten gemäss CD im Anhang erstellt. Akzeptierter Zertifikatsrequest auf dem Puppet Master Management
Erwartetes Resultat	Erfolgreicher Puppet run ohne Fehler, Jenkins erscheint im Puppet Dashboard des Management Puppet Masters, und die Weboberfläche ist über http://jenkins-mgmt-1.internaldomain.net:8080 erreichbar.

ID	4_CICD
Titel	Full Puppet Managed Arista SPINE Switches in der CI/CD Umgebung
Komponente	CI/CD Umgebung
Requirements Ref.	Use Case 1: Fully Puppet managed CI/CD environments, Use Case 2: Artifact installation over RPM's
Beschreibung	Die Arista SPINE Switches (DEV, INT und PRD) sollen komplett mit Puppet gemanagt werden. Das heisst, es sind nach dem ersten Puppet Run keine weiteren Interaktionen nötigt, um die Switches zur Lauffähigkeit zu bringen.
Vorgehen	Die Voraussetzungen für den Test müssen berücksichtigt werden. 1. Login auf den SPINE Switch 2. „puppet -t“ ausführen Dieser Vorgang wird für alle SPINE Switches pro Umgebung durchgeführt.
Voraussetzung	Initiale Switch Konfiguration (inkl. SSH Login), Installation von Puppet und Hieradaten gemäss CD im Anhang erstellt. Akzeptierter Zertifikatsrequest auf dem Puppet Master DEV / INT / PRD
Erwartetes Resultat	Erfolgreicher Puppet run ohne Fehler, SPINE Switch erscheint im Puppet Dashboard des Puppet Masters (DEV, INT, PRD) und die Switch API kann über den Browser aufgerufen werden. (https://spineswitch-UMGEBUNG-1/2.internaldomain.net)

ID	5_CICD
Titel	Full Puppet Managed Arista TOR Switches in der CI/CD Umgebung
Komponente	CI/CD Umgebung
Requirements Ref.	Use Case 1: Fully Puppet managed CI/CD environments, Use Case 2: Artifact installation over RPM's
Beschreibung	Die Arista TOR Switches (DEV, INT und PRD) sollen komplett mit Puppet gemanagt werden. Das heisst, es sind nach dem ersten Puppet Run keine weiteren Interaktionen nötigt, um die Switches zur Lauffähigkeit zu bringen.
Vorgehen	Die Voraussetzungen für den Test müssen berücksichtigt werden. 1. Login auf den TOR Switch 2. „puppet -t“ ausführen Dieser Vorgang wird für alle TOR Switches pro Umgebung durchgeführt.

Voraussetzung	Initiale Switch Konfiguration (inkl. SSH Login), Installation von Puppet und Hieradaten gemäss CD im Anhang erstellt. Akzeptierter Zertifikatsrequest auf dem Puppet Master DEV / INT / PRD
Erwartetes Resultat	Erfolgreicher Puppet run ohne Fehler, TOR Switch erscheint im Puppet Dashboard des Puppet Masters (DEV, INT, PRD) und die Switch API kann über den Browser aufgerufen werden. (https://torswitch-UMGEBUNG-1/2.internaldomain.net)

ID	6_CICD
Titel	Full Puppet Managed Arista Mgmt Switches in der CI/CD Umgebung
Komponente	CI/CD Umgebung
Requirements Ref.	Use Case 1: Fully Puppet managed CI/CD environments, Use Case 2: Artifact installation over RPM's
Beschreibung	Die Arista Mgmt Switches (DEV, INT und PRD) sollen komplett mit Puppet gemanagt werden. Das heisst, es sind nach dem ersten Puppet Run keine weiteren Interaktionen nötigt, um die Switches zur Lauffähigkeit zu bringen.
Vorgehen	Die Voraussetzungen für den Test müssen berücksichtigt werden. 1. Login auf den Mgmt Switch 2. „puppet -t" ausführen Dieser Vorgang wird für alle Mgmt Switches pro Umgebung durchgeführt.
Voraussetzung	Initiale Switch Konfiguration (inkl. SSH Login), Installation von Puppet und Hieradaten gemäss CD im Anhang erstellt. Akzeptierter Zertifikatsrequest auf dem Puppet Master DEV / INT / PRD
Erwartetes Resultat	Erfolgreicher Puppet run ohne Fehler, Mgmt Switch erscheint im Puppet Dashboard des Puppet Masters (DEV, INT, PRD) und die Switch API kann über den Browser aufgerufen werden. (https://mgmtswitch-UMGEBUNG-1/2.internaldomain.net)

ID	7_CICD
Titel	Drei identische Umgebungen (DEV, INT, PRD)
Komponente	CI/CD Umgebung
Requirements Ref.	Use Case 3: Three different identical environments (DEV, INT, PRD)
Beschreibung	Es existieren drei identische Umgebungen eine DEV, eine INT und eine PROD.

Vorgehen

1. Login aufs Puppet Dashboard der einzelnen Puppet Master Vergleich der existierenden Switches
2. Login in Gitlab um die Hieradaten zu vergleichen
3. Login auf Switch
4. „show configuration“ um die aktuelle Konfiguration anzuzeigen und zu vergleichen.

Voraussetzung Die drei Umgebungen konnten gemäss Testcases 1-6 komplett und ohne manuelle Eingriffe mit Puppet aufgebaut werden. Die Hieradaten für die Umgebungen sind mehrheitlich identisch und unterscheiden sich lediglich in Stack spezifischen Werte, wie z.B. IP Adressen.

Erwartetes Resultat Zum einen sind im Puppet Dashboard der drei Puppet Master (DEV, INT und PRD) genau gleich viele Switches vorhanden. Anhand der Hostnamen und Hieradaten wird erkenntlich, welche Typen von Switches aufgesetzt wurden. Zwei TOR, zwei SPINE und zwei Management Switches werden erwartet.

6.1.2 Testing Konfigurationsdatengenerator

Der Konfigurationsdatengenerator wird hauptsächlich mit Unittests, die mittels Rspec geschrieben wurden, getestet. Diese Tests sind in einem Jenkins Job Integriert und können von einer Person angestossen werden. Die Testresultate beeinflussen das Resultat des Jenkinsjobs: sind alle Tests erfolgreich durchgelaufen, ist der Jenkins Job erfolgreich, gab es Fehler in den Tests, weist der Jenkins Job einen fehlerhaften Zustand aus. In der Übersicht der Jenkinsjobs (Abbildung 19) kann das Testresultat ausgelesen werden.

Abbildung 19: RSpec Testresultate Übersicht

Die Abbildung 20 zeigt die Resultate der einzelnen RSpec Unittests für Hieragen.

Console Output

```
Started by user anonymous
[EnvInject] - Loading node environment variables.
Building in workspace /var/lib/jenkins/jobs/hieragen-testing/workspace
Wiping out workspace first.
Cloning the remote Git repository
Cloning repository
 > git init /var/lib/jenkins/jobs/hieragen-testing/workspace # timeout=10
Fetching upstream changes from https://gitlab.internaldomain.ch/raffaela.wuest/hieragen.git
 > git --version # timeout=10
 > git fetch --tags --progress https://gitlab.internaldomain.ch/raffaela.wuest/hieragen.git
+refs/heads/*:refs/remotes/origin/*
 > git config remote.origin.url https://gitlab.internaldomain.ch/raffaela.wuest/hieragen.git     # timeout=10
 > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git config remote.origin.url https://gitlab.internaldomain.ch/raffaela.wuest/hieragen.git     # timeout=10
Fetching upstream changes from https://gitlab.internaldomain.ch/raffaela.wuest/hieragen.git
 > git fetch --tags --progress https://gitlab.internaldomain.ch/raffaela.wuest/hieragen.git
+refs/heads/*:refs/remotes/origin/*
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
 > git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 0a1deaa1b090c1955008d70a9b7c313b96af4ff2 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 0a1deaa1b090c1955008d70a9b7c313b96af4ff2
 > git rev-list 4907a41e3a318ad34c022d064d1acd3fd68fcfd2 # timeout=10
 > git tag -a -f -m Jenkins Build #15 jenkins-hieragen-testing-15 # timeout=10
[workspace] $ /bin/sh -xe /tmp/hudson2939665965171487542.sh
+ cd .//
+ bundle install
Rubygems 1.8.23 is not threadsafe, so your gems will be installed one at a time. Upgrade to Rubygems 2.1.0 or higher to
enable parallel gem installation.
Resolving dependencies...
Using rake 10.4.2
Using git 1.2.9.1
Using gli 2.13.4
Using json 1.8.3
Using builder 3.2.2
Using bundler 1.11.2
Using diff-lcs 1.2.5
Using rspec-support 3.4.1
Using Hieragen 0.2.2 from source at `.`
Using ci_reporter 2.0.0
Using rspec-core 3.4.1
Using rspec-expectations 3.4.0
Using rspec-mocks 3.4.0
Using rspec 3.4.0
Bundle complete! 5 Gemfile dependencies, 14 gems now installed.
Use `bundle show [gemname]` to see where a bundled gem is installed.
+ bundle exec rspec
............

Finished in 0.02082 seconds (files took 0.15017 seconds to load)
12 examples, 0 failures

+ rm -rf
Finished: SUCCESS
```

Abbildung 20: RSpec Testresultate

6.1.3 Integrationstests

Die Integrationstests umfassen das Zusammenspiel der CI/CD Umgebung mit dem Hieradatengenerator. Das heisst, die kompletten Funktionalitäten dieser Projektarbeit werden mit den nachfolgenden Tests überprüft. Die Test-ID *_INT referenzieren auf die Integrations-Testcases.

Tabelle 5: Testfälle Systemintegrations-Tests

ID	1_INT
Titel	Installation Hieradatengenerator auf Puppet Master
Komponente	Konfigurationsdatengenerator und CI/CD Umgebung

Requirements Ref.	Use Case 2: Artifact installation over RPM
Beschreibung	Der Hieradatengenerator lässt sich mittels YUM auf den Puppet Master (DEV, INT, PRD) installieren, und die installierte Version des Generators kann abgefragt werden.
Vorgehen	1. Login auf Puppet Master (DEV, INT, PRD) 2. ausführen von „yum install ruby193-hieragen“ 3. nach erfolgreicher Installation Version abfragen „hieragen –version“
Voraussetzung	Der Hieradatengenerator wurde erfolgreich mit dem Jenkins zu einem RPM gebuildet und ist im Pulp Repository verfügbar.
Erwartetes Resultat	Hieragen konnte mittels yum erfolgreich installiert werden, und die Abfrage der Version gibt „0.2.2“ zurück

ID	2_INT
Titel	Konfiguration des Hieradatengenerators erstellen
Komponente	Konfigurationsdatengenerator und CI/CD Umgebung
Requirements Ref.	Use Case 4: Generate automatically a YAML File on a source base, do a Syntax check and push it to a repository
Beschreibung	Um die Hieradaten erfolgreich automatisch generieren zu lassen, muss der Generator vorgängig konfiguriert werden, dazu wird die Funktion „hieragen configure“ verwendet.
Vorgehen	1. Login auf Puppet Master (DEV, INT, PRD) 2. ausführen von „hieragen configure“ 3. Eingabe der sourcelocation „/etc/hieragen/inputdata.json“ 4. Eingabe des logfilepath „/var/log/“ 5. Eingabe der outputlocation „/etc/hieragen/output/“ 6. Eingabe des Gitrepositorys für den Upload „https://gitlab.internaldomain.ch/raffaela.wuest/master-dev-1.git“ oder „https://gitlab.internaldomain.ch/raffaela.wuest/master-int-1.git“ oder „https://gitlab.internaldomain.ch/raffaela.wuest/master-prd-1.git“
Voraussetzung	Der Konfigurationsdatengenerator ist auf dem entsprechenden System installiert.
Erwartetes Resultat	Die Meldung „successfully saved to file.“ erscheint auf der Commandline nach erfolgreicher Eingabe.

ID	3_INT
Titel	Automatische Erstellung der Hieradaten mit dem Generator
Komponente	Konfigurationsdatengenerator und CI/CD Umgebung
Requirements Ref.	Use Case 4: Generate automatically a YAML File on a source base, do a Syntax check and push it to a repository
Beschreibung	Die Hieradaten werden anhand der Datensource erstellt und in das entsprechende Gitlab Repository gespeichert.
Vorgehen	auf dem Puppet Master „hieragen run“ ausführen
Voraussetzung	Der Hieradatengenerator ist installiert und konfiguriert.
Erwartetes Resultat	„Successfully Uploaded...“ erscheint auf der Commandline und im Git Repository, welches vorgängig konfiguriert wurde, sind die neu generierten Dateien im Branch „generation“ vorhanden

6.2 Testresultate

Die Testresultate werden in einer überschaubaren Übersicht zusammengetragen, sodass der Leser dieser Arbeit einen Überblick über die aktuelle Testsituation erhält. die detaillierten Testresultate mit den einzelnen Schritten und dem kompletten Resultat ist im Anhang C zu finden. Die ausführlichen Testresultate im Anhang dienen der Reproduzierbarkeit, insbesondere bei fehlgeschlagenen Tests.

Tabelle 7: Testresultate

TestCase Ref. ID	1_CICDa
Titel	Full Puppet Managed Puppet Master in der CI/CD Umgebung
Datum und Zeit	04.12.2015 15:00 Uhr
Involvierte Komponenten	Puppet Master Mgmt
Resultat	Test Status Not OK

TestCase Ref. ID	1_CICDb
Titel	Full Puppet Managed Puppet Master in der CI/CD Umgebung
Datum und Zeit	04.12.2015 15:05 Uhr
Involvierte Komponenten	Puppet Master DEV
Resultat	Test Status OK

TestCase Ref. ID	1_CICDc
Titel	Full Puppet Managed Puppet Master in der CI/CD Umgebung
Datum und Zeit	04.12.2015 15:10 Uhr
Involvierte Komponenten	Puppet Master INT
Resultat	Test Status OK

TestCase Ref. ID	1_CICDd
Titel	Full Puppet Managed Puppet Master in der CI/CD Umgebung
Datum und Zeit	04.12.2015 15:15 Uhr
Involvierte Komponenten	Puppet Master PRD
Resultat	Test Status OK

TestCase Ref. ID	1_CICDa
Titel	Full Puppet Managed Puppet Master in der CI/CD Umgebung
Datum und Zeit	07.12.2015 07:55 Uhr
Involvierte Komponenten	Puppet Master Mgmt
Resultat	Test Status OK

TestCase Ref. ID	1_CICDa
Titel	Full Puppet Managed Puppet Master in der CI/CD Umgebung
Datum und Zeit	04.12.2015 15:30 Uhr
Involvierte Komponenten	Puppet Master Mgmt
Resultat	Test Status OK

TestCase Ref. ID	2_CICD
Titel	Full Puppet Managed Pulp Server in der CI/CD Umgebung
Datum und Zeit	04.12.2015 16:00 Uhr
Involvierte Komponenten	Pulp Server
Resultat	Test Status OK

TestCase Ref. ID	3_CICD
Titel	Full Puppet Managed Jenkins in der CI/CD Umgebung
Datum und Zeit	04.12.2015 16:07 Uhr
Involvierte Komponenten	Jenkins
Resultat	Test Status OK

TestCase Ref. ID	4_CICDa
Titel	Full Puppet Managed Arista SPINE Switches in der CI/CD Umgebung
Datum und Zeit	04.12.2015 16:10 Uhr
Involvierte Komponenten	SPINE Switch DEV
Resultat	Test Status OK

TestCase Ref. ID	4_CICDb
Titel	Full Puppet Managed Arista SPINE Switches in der CI/CD Umgebung
Datum und Zeit	04.12.2015 16:12 Uhr
Involvierte Komponenten	SPINE Switch INT
Resultat	Test Status OK

TestCase Ref. ID	4_CICDc
Titel	Full Puppet Managed Arista SPINE Switches in der CI/CD Umgebung
Datum und Zeit	04.12.2015 16:17 Uhr
Involvierte Komponenten	SPINE Switch PRD
Resultat	Test Status OK

TestCase Ref. ID	5_CICDa
Titel	Full Puppet Managed Arista TOR Switches in der CI/CD Umgebung
Datum und Zeit	04.12.2015 16:26 Uhr
Involvierte Komponenten	TOR Switch DEV
Resultat	Test Status Not OK

TestCase Ref. ID	5_CICDa
Titel	Full Puppet Managed Arista TOR Switches in der CI/CD Umgebung
Datum und Zeit	04.12.2015 16:29 Uhr
Involvierte Komponenten	TOR Switch DEV
Resultat	Test Status OK

TestCase Ref. ID	5_CICDb
Titel	Full Puppet Managed Arista TOR Switches in der CI/CD Umgebung
Datum und Zeit	04.12.2015 16:29 Uhr
Involvierte Komponenten	TOR Switch INT
Resultat	Test Status OK

TestCase Ref. ID	5_CICDc
Titel	Full Puppet Managed Arista TOR Switches in der CI/CD Umgebung
Datum und Zeit	04.12.2015 16:31 Uhr
Involvierte Komponenten	TOR Switch PRD
Resultat	Test Status OK

TestCase Ref. ID	6_CICDa
Titel	Full Puppet Managed Arista Mgmt Switches in der CI/CD Umgebung
Datum und Zeit	04.12.2015 16:38 Uhr
Involvierte Komponenten	Management Switch DEV
Resultat	Test Status OK

TestCase Ref. ID	6_CICDb
Titel	Full Puppet Managed Arista Mgmt Switches in der CI/CD Umgebung
Datum und Zeit	04.12.2015 16:40 Uhr
Involvierte Komponenten	Management Switch INT
Resultat	Test Status OK

TestCase Ref. ID	6_CICDc
Titel	Full Puppet Managed Arista Mgmt Switches in der CI/CD Umgebung
Datum und Zeit	04.12.2015 16:42 Uhr
Involvierte Komponenten	Management Switch PRD
Resultat	Test Status OK

TestCase Ref. ID	7_CICD
Titel	Drei identische Umgebungen (DEV, INT, PRD)
Datum und Zeit	04.12.2015 16:53 Uhr
Involvierte Komponenten	Puppet Master DEV, INT & PRD
Resultat	Test Status OK

TestCase Ref. ID	1_INTa
Titel	Installation Hieradatengenerator auf Puppet Master
Datum und Zeit	13.12.2015 18:05 Uhr
Involvierte Komponenten	Puppet Master DEV, Hieragen
Resultat	Test Status OK

TestCase Ref. ID	1_INTb
Titel	Installation Hieradatengenerator auf Puppet Master
Datum und Zeit	13.12.2015 18:35 Uhr
Involvierte Komponenten	Puppet Master INT, Hieragen
Resultat	Test Status OK

TestCase Ref. ID	1_INTc
Titel	Installation Hieradatengenerator auf Puppet Master
Datum und Zeit	13.12.2015 18:50 Uhr
Involvierte Komponenten	Puppet Master PRD, Hieragen
Resultat	Test Status OK

TestCase Ref. ID	2_INTa
Titel	Konfiguration des Hieradatengenerators erstellen
Datum und Zeit	13.12.2015 19:16 Uhr
Involvierte Komponenten	Puppet Master DEV, Hieragen
Resultat	Test Status OK

TestCase Ref. ID	2_INTb
Titel	Konfiguration des Hieradatengenerators erstellen
Datum und Zeit	13.12.2015 19:30 Uhr
Involvierte Komponenten	Puppet Master INT, Hieragen
Resultat	Test Status OK

TestCase Ref. ID	2_INTc
Titel	Konfiguration des Hieradatengenerators erstellen
Datum und Zeit	13.12.2015 19:30 Uhr
Involvierte Komponenten	Puppet Master PRD, Hieragen
Resultat	Test Status OK

TestCase Ref. ID	3_INTa
Titel	Automatische Erstellung der Hieradaten mit dem Generator
Datum und Zeit	15.12.2015 18:45 Uhr
Involvierte Komponenten	Puppet Master DEV, Hieragen
Resultat	Test Status OK

TestCase Ref. ID	3_INTb
Titel	Automatische Erstellung der Hieradaten mit dem Generator
Datum und Zeit	15.12.2015 19:00 Uhr
Involvierte Komponenten	Puppet Master INT, Hieragen
Resultat	Test Status OK

TestCase Ref. ID	3_INTc
Titel	Automatische Erstellung der Hieradaten mit dem Generator
Datum und Zeit	15.12.2015 19:17 Uhr
Involvierte Komponenten	Puppet Master PRD, Hieragen
Resultat	Test Status OK

Total Tests OK: 27
Total Tests Not OK: 2

7. Fazit und Lessons Learned

Die Informatik Projektarbeit PAWI besteht aus zwei Hauptteilen: dem Aufbau einer CI/CD Umgebung und dem Programmieren eines Hieradatengenerators für Arista Switches. Die CI/CD Umgebung ist in drei gleichwertige Stages (DEV, INT und PRD) unterteilt, mit jeweils einem Puppet Master, zwei virtuelle TOR Switches, zwei virtuelle SPINE Switches und zwei virtuelle Management Switches. Alle drei Stages sind an der Management Umgebung (bestehend aus einem Puppet Master, einem Pulp Server und einem Jenkins) angehängt und beziehen daraus ihre Konfigurationen. Alle Systeme sind mit Puppet gemanagt. Um die Fehlerquote beim Erstellen der Hieradaten (Konfigurationen für Puppet) zu minimieren, sollen zukünftig Generatoren für die Hieradatenerstellung zum Einsatz kommen. Im Rahmen dieser Projektarbeit wurde im zweiten Teil ein Hieradatengenerator, genannt „hieragen“, für Arista Switches erstellt. Die Software ist in Ruby geschrieben und ermöglicht, anhand einer initialen Konfiguration des Generators, Hieradaten für alle drei Switch Typen (TOR, SPINE und MGMT) zu schreiben und auf ein Git Repository zu pushen.

Für das Projekt wurde ein agiles Vorgehen gewählt. Die einzelnen Projektabschnitte wurden als Sprints geplant und mit einer Retrospective nach dem effektiven Sprint abgeschlossen. Die Retrospectiven, auch Rückblicke genannt, sind im Dokument Projektmanagement Kapitel 4 einsehbar. Sie ermöglichen einen tieferen und detaillierteren Einblick in die Probleme und Erfolge während der Projektarbeit. Aus Sicht Projektcontrolling waren sie insbesondere während dem Projekt als Lessons Learned spannend, positive Aspekte wurden in darauf folgenden Sprints weiter gepflegt, negative Aspekte wurden zu vermeiden versucht.
Die Projektarbeit war enorm spannend und brachte viel Abwechslung und grosse Lernmomente. Die Kombination zwischen Softwareprojekt und Infrastrukturprojekt bietet die Vielfältigkeit, welche Projekte wie diese einzigartig machen. Bereits erlangtes Wissen, in Bezug auf Puppet, konnte erweitert und gefestigt werden. Neues Wissen konnte in den Bereichen Software Entwicklung mittels Ruby und dem Builden mit Hilfe von Jenkins DSL-Jobs angeeignet werden.

7.1 Ausblick

Das Projekt „Hieradatengenerierung“ wird bei Swisscom (Schweiz) AG weitergeführt. Die zentrale Datensource EnClouder wird um die fehlenden Informationen für die Infrastruktur Komponenten erweitert und wird alle Sourcedaten in einer zentralen Sourcedatei allen Generatoren zur Verfügung stellen. Für alle eingesetzten Infrastruktur Komponenten (z.B. Puppet Master, Pulp Server, Storage, und viele mehr) sollen Generatoren programmiert werden, um auch in diesem Bereich die Automatisierung voranzutreiben und gegen 100% zu erreichen.

Abbildungsverzeichnis

1 Auszug Systemlandschaft Standard CI/CD Umgebungen 5
2 Puppet Roles und Service Pattern 6
3 Switch Architektur 7
4 Systemlandschaft CI/CD Umgebung 15
5 Komponenten Diagramm Konfigurationsdatengenerator 16
6 Benutzerinteraktion mit dem Konfigurationsdatengenerator 18
7 Schnittstellen zu Umsystemen 19
8 Zusammenhänge Puppet Master, Puppet Agents und Hieradatenrepositories 22
9 Puppet Dashboard Mgmt Umgebung 26
10 Pulp Server Repositories 28
11 Jenkins Weboberfläche 29
12 Arista Command API 34
13 Klassendiagramm Hieragen 35
14 Jenkins Buildjobs 39
15 Hieragen Konfiguration 40
16 Hieragen Version 40
17 Hieragen Help 41
18 Hieragen Run 41
19 RSpec Testresultate Übersicht 46
20 RSpec Testresultate 47

Tabellenverzeichnis

2 Einzusetzende Puppet Roles 16
3 Testfälle CI/CD Umgebung 42
5 Testfälle Systemintegrations-Tests 47
7 Testresultate 49
9 Serverliste mit Puppet Master 57
10 Testresultate 67

Literatur

Arista-Networks (2015a). Coprorate Profile. URL: https://www.arista.com/en/company/company-overview.

– (2015b). vEOS Running EOS in a VM). URL: https://eos.arista.com/veos-running-eos-in-a-vm/.

Pulp-Project (2015). What is Pulp? URL: http://www.pulpproject.org/.

Puppet-Labs (2015a). Hiera 1: Overview. URL: https://puppetlabs.com/puppet/what-is-puppet.

– (2015b). What is Puppet? URL: https://puppetlabs.com/puppet/what-is-puppet.

Puzzle-ITC (2015). EnClouder - Server Provisioning für Swisscom IT Services – Webapplikationen. URL: https://www.puzzle.ch/de/blog/articles/pitc_client/enclouder-server-provisioning-fuer-swisscom-it-services-webapplikation/.

Rouse, Margaret (2015). leaf-spine (leaf-spine architecture) definition). URL: http://searchdatacenter.techtarget.com/definition/Leaf-spine/.

Ruby-Community (2015). Ruby ist... URL: https://www.ruby-lang.org/de/.

IT-Wissen (2015). ToR (top of rack). URL: http://www.itwissen.info/definition/lexikon/ToR-top-of-rack-ToR-Switch.html.

Wüst, Raffaela (2015). Puppet Role ReadTheDocs Applikation. Abschlussbericht Praxismodul der Hochschule Luzern. Hochschule Luzern Technik und Architektur.

Anhang A Serverliste CI/CD Umgebung

Die nachfolgende Tabelle 9 beinhaltet alle in dieser Projektarbeit eingesetzten Server, ihre Hostnamen, sowie das verwendete Betriebssystem.

Tabelle 9: Serverliste mit Puppet Master

Hostname	Puppet Master	OS
master-mgmt-1.internaldomain.net	puppet1-cfmg-dev	RHEL 6
pulp-mgmt-1.internaldomain.net	master-mgmt-1	RHEL 6
jenkins-mgmt-1.internaldomain.net	master-mgmt-1	RHEL 6
master-dev-1.internaldomain.net	master-mgmt-1	RHEL 6
master-int-1.internaldomain.net	master-mgmt-1	RHEL 6
master-prd-1.internaldomain.net	master-mgmt-1	RHEL 6
torswitch-dev-1.internaldomain.net	master-dev-1	vEOS
torswitch-dev-2.internaldomain.net	master-dev-1	vEOS
mgmtswitch-dev-1.internaldomain.net	master-dev-1	vEOS
mgmtswitch-dev-2.internaldomain.net	master-dev-1	vEOS
spineswitch-dev-1.internaldomain.net	master-dev-1	vEOS
spineswitch-dev-2.internaldomain.net	master-dev-1	vEOS
mgmtswitch-int-1.internaldomain.net	master-int-1	vEOS
mgmtswitch-int-2.internaldomain.net	master-int-1	vEOS
spineswitch-int-1.internaldomain.net	master-int-1	vEOS
spineswitch-int-2.internaldomain.net	master-int-1	vEOS
torswitch-int-1.internaldomain.net	master-int-1	vEOS
torswitch-int-2.internaldomain.net	master-int-1	vEOS
mgmtswitch-prd-1.internaldomain.net	master-prd-1	vEOS
mgmtswitch-prd-2.internaldomain.net	master-prd-1	vEOS
spineswitch-prd-1.internaldomain.net	master-prd-1	vEOS
spineswitch-prd-2.internaldomain.net	master-prd-1	vEOS
torswitch-prd-1.internaldomain.net	master-prd-1	vEOS
torswitch-prd-2.internaldomain.net	master-prd-1	vEOS

Anhang B CI/CD und Puppet Konfigurationen

Um Puppet in Betrieb zu nehmen werden verschiedene Konfigurationen durchgeführt. Diese sind nachfolgend zu finden.

Anhang B1 Konfigurationsdatei Puppet.conf

Die Puppet.conf Datei ist auf jedem Agent unter /etc/puppet/puppet.conf zu finden und definiert unter anderem, bei welchem Puppet Master sich der Agent registrieren und seine Daten beziehen muss.

Auflistung 16: Auszug aus der Puppet.conf Datei

```
[main]
    # The Puppet log directory.
    # The default value is '$vardir/log'.
    logdir = /var/log/puppet

    # Where Puppet PID files are kept.
    # The default value is '$vardir/run'.
    rundir = /var/run/puppet

    # Where SSL certificates are kept.
    # The default value is '$confdir/ssl'.
    ssldir = $vardir/ssl

[agent]
    # The file in which puppetd stores a list of the classes
    # associated with the retrieved configuratiion. Can be loaded in
    # the separate ''puppet'' executable using the ''--loadclasses''
    # option.
    # The default value is '$confdir/classes.txt'.
    classfile = $vardir/classes.txt

    # Where puppetd caches the local configuration. An
    # extension indicating the cache format is added automatically.
    # The default value is '$confdir/localconfig'.
    localconfig = $vardir/localconfig
server=master-mgmt-1.internaldomain.ch
report=true
```

Anhang B2 Self Signed SSL Certifiactes

Der Pulp Server ist für seine korrekte Funktionalität auf SSL Zertifikate angewiesen. Im Rahmen des internen Gebrauches des Pulp Servers können selbst signierte Zertifikate - sogenannte self signed Cer-

tificates - erstellt werden. Die nachfolgende Methode in Auflistung ?? dient dazu, dass die Clients die Connection zum Pulp Server verifizieren können, so wie dies mit einer offiziellen trusted Certificate Authority möglich ist.

Als erstes muss die eigene Certificate Authority erstellt werden, sie wird die Server Zertifikate signieren. Dazu sind zwei Schritte nötig, zum einen das Generieren eines privaten Keys siehe ?? und zum anderen das Generieren des Zertifikats siehe ??.

Auflistung 17: Generieren des Private Keys

```
# Generate Key
$ openssl genrsa -out CA.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
```

Auflistung 18: Generieren des Zertifikats

```
$ openssl req -x509 -new -nodes -key CA.key -days 1024 -out CA.crt
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CH
State or Province Name (full name) [Some-State]:Bern
Locality Name (eg, city) []:Bern
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Swisscom PAWI CA
Organizational Unit Name (eg, section) []:Testing only
Common Name (e.g. server FQDN or YOUR name) []:Raffaela Wüst
Email Address []:
```

Im nächsten Schritt wird der SSL Key und der Certificate Signing Request für den Pulp Server erstellt. Dazu wird ein Zertifikat für den Pulp Server ?? und einen Certificate Signing Request (CSR) ?? benötigt.

Auflistung 19: Generieren des Zertifikats für den Pulp Server

```
$ openssl genrsa -out pulp-mgmt-1.internaldomain.net.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
```

Auflistung 20: Zertifikat Signing Request erstellen

```
$ openssl req -new -key pulp-mgmt-1.internaldomain.net.key -out
    pulp-mgmt-1.internaldomain.net.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CH
State or Province Name (full name) [Some-State]:Bern
Locality Name (eg, city) []:Bern
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Swisscom PAWI
Organizational Unit Name (eg, section) []:DEV
Common Name (e.g. server FQDN or YOUR name) []:pulp-mgmt-1.internaldomain.net
Email Address []:
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Der generierte CSR kann nun verwendet werden, um ein Zertifikat zu generieren, welches zum Private Key passt.

Anhang B3 DOPv Deployment Plan für Arista Switches

Für das Deployment der Arista Switches wurde DOPv eingesetzt. Dazu wurde der nachfolgende Deployment Plan erstellt.21

Auflistung 21: DOPv Deployment Plan PRD

```
infrastructures:
  management:
    type: rhev
    endpoint: https://rhevm.internaldomain.net/api
    credentials:
      username: rhev@internaldomain.net
      password: topsecret

    networks:
      rhevm:
        ip_pool:
          from: 192.168.1.15
```

```
13              to: 192.168.1.20
14            ip_netmask: 255.255.255.0
15            ip_defgw: 192.168.1.1
16            dns:
17              nameserver:
18                - 192.168.0.10
19              domain: internaldomain.net
20          puppettest_vms:
21            ip_pool:
22              from: 192.168.1.15
23              to: 192.168.1.20
24            ip_netmask: 255.255.255.0
25            ip_defgw: 192.168.1.1
26            dns:
27              nameserver:
28                - 192.168.0.10
29              domain: internaldomain.net
30
31  nodes:
32    # Arista Switches TOR 1
33    torswitch-dev-1.internaldomain.net:
34      infrastructure: management
35      infrastructure_properties:
36        datacenter: lab
37        cluster: clu-lab
38      image: vEOS-4.14.2F
39      interfaces:
40        nic1:
41          network: puppettest_vms
42        nic2:
43          network: rhevm
44        nic3:
45          network: rhevm
46      cpu: 2
47      memory: 2G
48
49    # Arista Switches TOR 2
50    torswitch-dev-2.internaldomain.net:
51      infrastructure: management
52      infrastructure_properties:
53        datacenter: lab
54        cluster: clu-lab
55      image: vEOS-4.14.2F
56      interfaces:
57        nic1:
58          network: puppettest_vms
```

```
59          nic2:
60            network: rhevm
61          nic3:
62            network: rhevm
63        cpu: 2
64        memory: 2G
65
66      # Arista Switches SPINE 1
67      spineswitch-dev-1.internaldomain.net:
68        infrastructure: management
69        infrastructure_properties:
70          datacenter: lab
71          cluster: clu-lab
72        image: vEOS-4.14.2F
73        interfaces:
74          nic1:
75            network: puppettest_vms
76          nic2:
77            network: rhevm
78          nic3:
79            network: rhevm
80        cpu: 2
81        memory: 2G
82
83      # Arista Switches SPINE 2
84      spineswitch-dev-2.internaldomain.net:
85        infrastructure: management
86        infrastructure_properties:
87          datacenter: lab
88          cluster: clu-lab
89        image: vEOS-4.14.2F
90        interfaces:
91          nic1:
92            network: puppettest_vms
93          nic2:
94            network: rhevm
95          nic3:
96            network: rhevm
97        cpu: 2
98        memory: 2G
99
100     # Arista Switches Mgmt 1
101     mgmtswitch-dev-1.internaldomain.net:
102       infrastructure: management
103       infrastructure_properties:
104         datacenter: lab
```

```
        cluster: clu-lab
    image: vEOS-4.14.2F
    interfaces:
        nic1:
            network: puppettest_vms
        nic2:
            network: rhevm
        nic3:
            network: rhevm
    cpu: 2
    memory: 2G

  # Arista Switches Mgmt 2
  mgmtswitch-dev-2.internaldomain.net:
    infrastructure: management
    infrastructure_properties:
        datacenter: lab
        cluster: clu-lab
    image: vEOS-4.14.2F
    interfaces:
        nic1:
            network: puppettest_vms
        nic2:
            network: rhevm
        nic3:
            network: rhevm
    cpu: 2
    memory: 2G
```

Anhang B4 Arista Switches Initial Konfiguraiton

Damit die Arista Switches via SSH erreicht werden konnten, mussten vorgängig einige Konfigurationsschritte durchgeführt werden. (Auflistung 22)

Auflistung 22: Initiale Konfiguration der Arista Switches

```
#Login mit dem User Admin, vorerst ohne Passwort
admin

#Eintreten in den Priviledged Mode
enable

#Eintreten in den Konfigurations Modus
conf terminal

#Angabe welches Interface konfiguriert werden soll
```

```
interface management 1

#IP Adresse für das Mgmt Interface 1 setzen
ip address 172.20.20.99/24

#IP Route und Gateway für das Mgmt Interface 1 setzen
ip route 0.0.0.0/0 172.20.20.1

#Neues Passwort für den Admin User setzen
username admin secret newpassword

#Aktivieren von Trippe-A
aaa authorization exec default local

#IP Routing aktivieren
ip routing

#Gesamte Konfiguration persistent speichern
write
```

Anhang B5 Arista Switches Puppet Installation

Die Auflistung 23 zeigt die einzelnen Schritte für die Puppet Installation auf den Arista Switches.

Auflistung 23: Installation von Puppet auf den Arista Switches

```
####################################
# DNS und Hostname konfigurieren
####################################

#Login mit dem User Admin, vorerst ohne Passwort
admin

#Eintreten in den Priviledged Mode
enable

#Eintreten in den Konfigurations Modus
conf terminal

#IP-Adresse des DNS Servers setzen
ip name-server vrf default 172.20.1.12

#Domain setzen
ip domain-name internaldomain.ch

#Hostnamen für den Switch setzen
```

```
hostname mgmtswitch-dev-1

#Passwort für Network-Operator setzen
username eapi role network-operator secret 5 $1asdF.sdffsdfafw2234wlehf5jP1

#Zulassen von http-commands
management api http-commands
    no shutdown
    vrf cloud-mgmt
        no shutdown
!

####################################
# Puppet Installation
####################################

#Kopieren der Ruby, Puppet und EOS Puppet Konfiguration Extension
copy http://ztp.internaldomain.ch/files/extensions/ruby-1.9.3-1.swix
    extension:ruby-1.9.3-1.swix
copy http://ztp.internaldomain.ch/files/extensions/puppet-3.7.1-3-ruby1.swix
    extension:puppet-3.7.1-3-ruby1.swix
copy http://ztp.internaldomain.ch/files/extensions/eos-puppet-conf-0.1-2.eos4.noarch.rpm
    extension:eos-puppet-conf-0.1-2.eos4.noarch.rpm

#Anzeigen der vorhandenen Extensions, diese sind noch nicht installiert
show extensions

#Installation der Extensions
extension ruby-1.9.3-1.swix
extension puppet-3.7.1-3-ruby1.swix
extension eos-puppet-conf-0.1-2.eos4.noarch.rpm

#Persistieren der Installation, sprich, bei jedem Bootvorgang werden die Extensions neu installiert.
copy installed-extensions boot-extensions

#Kontrolle ob Extensions installiert wurden
show boot-extensions


####################################
# Puppet Konfiguration
####################################

#Installation der eapi.conf
http://ztp.internaldomain.ch:8080/files/eapi.conf flash:eapi.conf

```

```
#Eintreten in den Konfigurations Modus
conf terminal

#Puppet Konfiguration erstellen
alias puppet bash sudo puppet agent --server puppet-dev-1.internaldomain.ch --confdir /persist/sys/puppet
      --daemonize --no-onetime
!
event-handler puppet-agent-on-startup
    trigger on-boot
    action bash sudo puppet agent --server puppet-dev-1.internaldomain.ch --confdir /persist/sys/puppet
          --daemonize --no-onetime
    delay 60
!
End
```

Anhang C Testresultate

Nachfolgend eine Auflistung der detaillierten Testresultate. Die Testresultate beinhalten die einzelnen Testschritte sowie Datum, Uhrzeit und allfällige Äussere Einflüsse, sodass insbesondere fehlgeschlagene Tests reproduzierbar sind und daher wichtigen Input für die Weiterentwicklung und die Fehlerbehebung liefern.

Tabelle 10: Testresultate

TestCase Ref. ID	1_CICDa
Datum und Zeit	04.12.2015 15:00 Uhr
Involvierte Komponenten	Puppet Master Mgmt
Äussere Einflüsse	-
Genauer Testablauf	1. Login via Linux Jump Host auf Puppet Master Mgmt (master-mgmt-1.internaldomain.net) 2. puppet agent -t
Resultat	Test Status Not OK, Output des Puppet runs Mgmt:

```
Info: Retrieving pluginfacts
Error: /File[/var/lib/puppet/facts.d]: Failed to generate additional
    resources using 'eval_generate': SSL_connect returned=1
    errno=0 state=SSLv3 read server session ticket A: sslv3 alert
    certificate revoked
```

TestCase Ref. ID	1_CICDb
Datum und Zeit	04.12.2015 15:05 Uhr
Involvierte Komponenten	Puppet Master DEV
Äussere Einflüsse	-
Genauer Testablauf	1. Login via Linux Jump Host auf Puppet Master Mgmt (master-dev-1.internaldomain.net) 2. puppet agent -t

Resultat	Test Status OK, Output des Puppet runs DEV:

```
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
Info: Caching catalog for master-dev-1.internaldomain.net
Info: Applying configuration version '1449137719'
Notice: Finished catalog run in 15.52 seconds
```

TestCase Ref. ID	1_CICDc
Datum und Zeit	04.12.2015 15:10 Uhr
Involvierte Komponenten	Puppet Master INT
Äussere Einflüsse	-
Genauer Testablauf	

1. Login via Linux Jump Host auf Puppet Master Mgmt (master-int-1.internaldomain.net)
2. puppet agent -t

Resultat	Test Status OK, Output des Puppet runs INT:

```
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
Info: Caching catalog for master-int-1.internaldomain.net
Info: Applying configuration version '1449137719'
Notice: Finished catalog run in 19.44 seconds
```

TestCase Ref. ID	1_CICDd
Datum und Zeit	04.12.2015 15:15 Uhr
Involvierte Komponenten	Puppet Master PRD
Äussere Einflüsse	-
Genauer Testablauf	

1. Login via Linux Jump Host auf Puppet Master Mgmt (master-prd-1.internaldomain.net)
2. puppet agent -t

Resultat

Test Status OK, Output des Puppet runs PRD:

```
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
Info: Caching catalog for master-prd-1.internaldomain.net
Info: Applying configuration version '1449137719'
Notice: Finished catalog run in 21.57 seconds
```

TestCase Ref. ID	1_CICDa
Datum und Zeit	07.12.2015 07:55 Uhr
Involvierte Komponenten	Puppet Master Mgmt
Äussere Einflüsse	-
Genauer Testablauf	

1. Login via Linux Jump Host auf Puppet Master Mgmt (master-mgmt-1.internaldomain.net)
2. rm -rf /var/lib/puppet/ssl
3. Login via Linux Jump Host auf Puppet Master Lab (master1-cfmg-dev.internaldomain.net)
4. puppet cert clean rh6pt-28.internaldomain.net (Server war mit dem falschen FQDN registriert)
5. Login via Linux Jump auf Puppet Master Mgmt (master-mgmt-1.internaldomain.net)
6. puppet agent -t

Resultat

Test Status OK, Output des Puppet runs Mgmt:

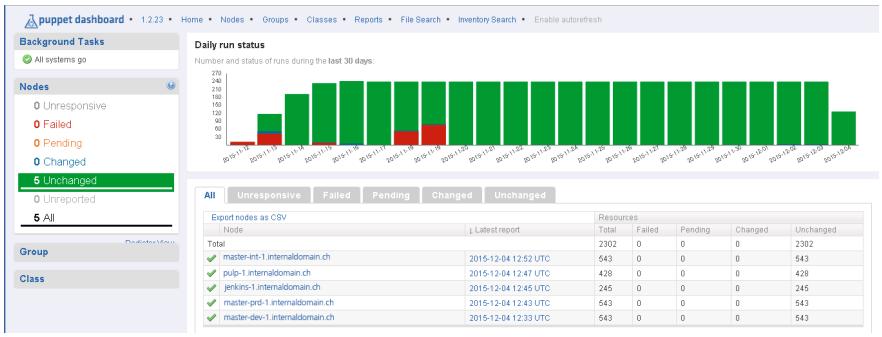
```
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
Info: Caching catalog for master-mgmt-1.internaldomain.net
Info: Applying configuration version '1449450511'
Notice: Finished catalog run in 23.54 seconds
```

TestCase Ref. ID	1_CICDa
Datum und Zeit	04.12.2015 15:30 Uhr

Involvierte Komponenten	Puppet Master Mgmt
Äussere Einflüsse	-
Genauer Testablauf	

1. Login via Windows Jump Host auf Dashboard des Puppet Masters Mgmt (http://master-mgmt-1.internaldomain.net/dashboard)
2. prüfen ob alle Puppet Master (DEV, INT, PRD) „grün“ sind.

Resultat	Test Status OK,



TestCase Ref. ID	2_CICD
Datum und Zeit	04.12.2015 16:00 Uhr
Involvierte Komponenten	Pulp Server
Äussere Einflüsse	-
Genauer Testablauf	

1. Login via Linux Jump Host auf Pulp Server (pulp-mgmt-1.internal-domain.net)
2. puppet agent -t

Resultat	Test Status OK, Output des Puppet runs:

```
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
Info: Caching catalog for pulp-mgmt-1.internaldomain.net
Info: Applying configuration version '1451290915'
Notice: Finished catalog run in 32.78 seconds
```

TestCase Ref. ID	3_CICD
Datum und Zeit	04.12.2015 16:07 Uhr
Involvierte Komponenten	Jenkins
Äussere Einflüsse	-

Genauer Testablauf

1. Login via Linux Jump Host auf Jenkins Server (jenkins-mgmt-1.internaldomain.net)
2. puppet agent -t

Resultat

Test Status OK, Output des Puppet runs:

```
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
Info: Caching catalog for jenkins-mgmt-1.internaldomain.net
Info: Applying configuration version '1451290915'
Notice: Finished catalog run in 13.68 seconds
```

TestCase Ref. ID	4_CICDa
Datum und Zeit	04.12.2015 16:10 Uhr
Involvierte Komponenten	SPINE Switch DEV
Äussere Einflüsse	-

Genauer Testablauf

1. Login via Linux Jump Host auf SPINE Switch DEV (spineswitch-dev-1.internaldomain.net)
2. puppet -t

Resultat

Test Status OK, Output des Puppet runs:

```
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
Info: Caching catalog for spineswitch-dev-1.internaldomain.net
Info: Applying configuration version '1451459503'
Notice: Finished catalog run in 5.91 seconds
```

TestCase Ref. ID	4_CICDb
Datum und Zeit	04.12.2015 16:12 Uhr
Involvierte Komponenten	SPINE Switch INT
Äussere Einflüsse	-

Genauer Testablauf

1. Login via Linux Jump Host auf SPINE Switch INT (spineswitch-int-1.internaldomain.net)
2. puppet -t

Resultat

Test Status OK, Output des Puppet runs:

```
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
Info: Caching catalog for spineswitch-int-1.internaldomain.net
Info: Applying configuration version '1451459503'
Notice: Finished catalog run in 7.32 seconds
```

TestCase Ref. ID	4_CICDc
Datum und Zeit	04.12.2015 16:17 Uhr
Involvierte Komponenten	SPINE Switch PRD
Äussere Einflüsse	-

Genauer Testablauf

1. Login via Linux Jump Host auf SPINE Switch PRD (spineswitch-prd-1.internaldomain.net)
2. puppet -t

Resultat

Test Status OK, Output des Puppet runs:

```
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
Info: Caching catalog for spineswitch-prd-1.internaldomain.net
Info: Applying configuration version '1451459503'
Notice: Finished catalog run in 5.63 seconds
```

TestCase Ref. ID	5_CICDa
Datum und Zeit	04.12.2015 16:26 Uhr
Involvierte Komponenten	TOR Switch DEV
Äussere Einflüsse	-

Genauer Testablauf

1. Login via Linux Jump Host auf TOR Switch DEV (torswitch-dev-1.internaldomain.net)
2. puppet -t

Resultat

Test Status NOT OK, Login failed:

```
ssh: connect to host torswitch-dev-1.internaldomain.net port 22: No route to host
```

TestCase Ref. ID	5_CICDa
Datum und Zeit	04.12.2015 16:29 Uhr
Involvierte Komponenten	TOR Switch DEV
Äussere Einflüsse	-

Genauer Testablauf

1. Login via Linux Jump Host auf TOR Switch DEV (torswitch-dev-1.internaldomain.net)
2. puppet -t

Resultat

Test Status OK mit Änderungen, Output des Puppet runs:

```
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
Info: Caching catalog for torswitch-dev-1.internaldomain.net
Info: Applying configuration version '1451459503'
Notice: /Stage[main]/Nc_arista_eos/Eos_ipinterface[Vlan4094]/address: address changed '0.0.0.0/0' to '1.1.1.1/30'
Notice: /Stage[main]/Nc_arista_eos/Eos_ospf_instance[1]/active_interfaces: active_interfaces changed [] to 'Ethernet3 Ethernet4 Vlan4093'
Notice: /Stage[main]/Nc_arista_eos/Eos_mlag[settings]/peer_address: defined 'peer_address' as '1.1.1.2'
Notice: Finished catalog run in 41.50 seconds
```

TestCase Ref. ID	5_CICDa
Datum und Zeit	04.12.2015 16:29 Uhr

Involvierte Komponenten	TOR Switch DEV
Äussere Einflüsse	-
Genauer Testablauf	

1. Login via Linux Jump Host auf TOR Switch DEV (torswitch-dev-1.internaldomain.net)
2. puppet -t

Resultat: Test Status OK, Output des Puppet runs:

```
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
Info: Caching catalog for torswitch-dev-1.internaldomain.net
Info: Applying configuration version '1451459503'
Notice: Finished catalog run in 23.13 seconds
```

TestCase Ref. ID	5_CICDb
Datum und Zeit	04.12.2015 16:29 Uhr
Involvierte Komponenten	TOR Switch INT
Äussere Einflüsse	-
Genauer Testablauf	

1. Login via Linux Jump Host auf TOR Switch INT (torswitch-int-1.internaldomain.net)
2. puppet -t

Resultat: Test Status OK, Output des Puppet runs:

```
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
Info: Caching catalog for torswitch-int-1.internaldomain.net
Info: Applying configuration version '1451459503'
Notice: Finished catalog run in 15.08 seconds
```

TestCase Ref. ID	5_CICDc
Datum und Zeit	04.12.2015 16:31 Uhr
Involvierte Komponenten	TOR Switch PRD
Äussere Einflüsse	-

Genauer Testablauf

1. Login via Linux Jump Host auf TOR Switch PRD (torswitch-prd-1.internaldomain.net)
2. puppet -t

Resultat

Test Status OK, Output des Puppet runs:

```
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
Info: Caching catalog for torswitch-prd-1.internaldomain.net
Info: Applying configuration version '1451459503'
Notice: Finished catalog run in 24.17 seconds
```

TestCase Ref. ID	6_CICDa
Datum und Zeit	04.12.2015 16:38 Uhr
Involvierte Komponenten	Management Switch DEV
Äussere Einflüsse	-

Genauer Testablauf

1. Login via Linux Jump Host auf MGMT Switch DEV (mgmtswitch-dev-1.internaldomain.net)
2. puppet -t

Resultat

Test Status OK, Output des Puppet runs:

```
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
Info: Caching catalog for mgmtswitch-dev-1.internaldomain.net
Info: Applying configuration version '1451459503'
Notice: Finished catalog run in 20.17 seconds
```

TestCase Ref. ID	6_CICDb
Datum und Zeit	04.12.2015 16:40 Uhr
Involvierte Komponenten	Management Switch INT
Äussere Einflüsse	-

Genauer Testablauf

1. Login via Linux Jump Host auf MGMT Switch INT (mgmtswitch-int-1.internaldomain.net)
2. puppet -t

Resultat

Test Status OK, Output des Puppet runs:

```
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
Info: Caching catalog for mgmtswitch-int-1.internaldomain.net
Info: Applying configuration version '1451459503'
Notice: Finished catalog run in 19.70 seconds
```

TestCase Ref. ID	6_CICDc
Datum und Zeit	04.12.2015 16:42 Uhr
Involvierte Komponenten	Management Switch PRD
Äussere Einflüsse	-

Genauer Testablauf

1. Login via Linux Jump Host auf MGMT Switch PRD (mgmtswitch-prd-1.internaldomain.net)
2. puppet -t

Resultat

Test Status OK, Output des Puppet runs:

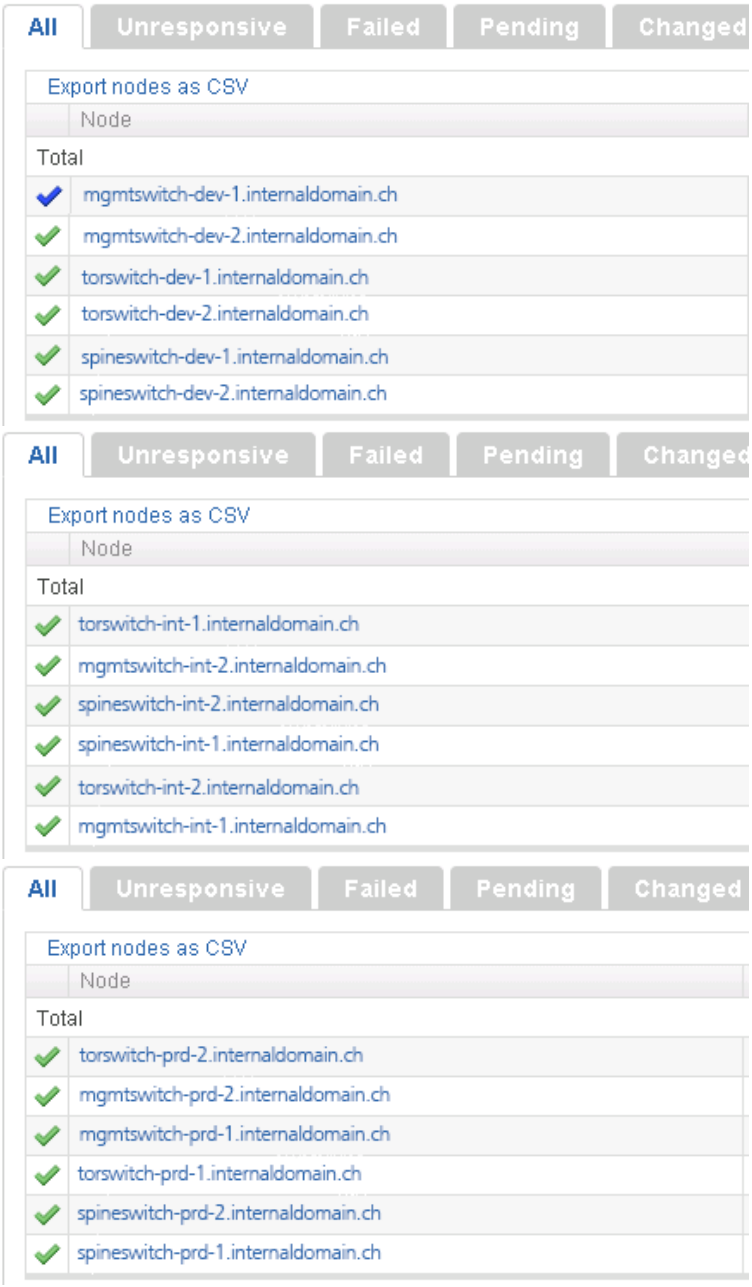
```
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
Info: Caching catalog for mgmtswitch-prd-1.internaldomain.net
Info: Applying configuration version '1451459503'
Notice: Finished catalog run in 16.54 seconds
```

TestCase Ref. ID	7_CICD
Datum und Zeit	04.12.2015 16:53 Uhr
Involvierte Komponenten	Puppet Master DEV, INT & PRD
Äussere Einflüsse	-

Genauer Testablauf

1. Login via Windows Jump Host auf Dashboard Puppet Master DEV (master-dev-1.internaldomain.net)
2. Login via Windows Jump Host auf Dashboard Puppet Master INT (master-int-1.internaldomain.net)
3. Login via Windows Jump Host auf Dashboard Puppet Master PRD (master-prd-1.internaldomain.net)

Resultat Test Status OK:



TestCase Ref. ID	1_INTa
Datum und Zeit	13.12.2015 18:05 Uhr
Involvierte Komponenten	Puppet Master DEV, Hieragen
Äussere Einflüsse	-
Genauer Testablauf	

1. Login via Linux Jump Host auf Puppet Master DEV (master-dev-1.internaldomain.net)
2. yum clean all
3. yum install ruby193-hieragen
4. hieragen –version

Resultat: Test Status OK:

```
[root@master-dev-1 ~]# yum install ruby193-hieragen
...
Updated:
  ruby193-hieragen.x86_64 0:0.2.2-39

Complete!
[root@master-dev-1 ~]# hieragen --version
hieragen version 0.2.2
```

TestCase Ref. ID	1_INTb
Datum und Zeit	13.12.2015 18:35 Uhr
Involvierte Komponenten	Puppet Master INT, Hieragen
Äussere Einflüsse	-
Genauer Testablauf	

1. Login via Linux Jump Host auf Puppet Master INT (master-int-1.internaldomain.net)
2. yum clean all
3. yum install ruby193-hieragen
4. hieragen –version

Resultat Test Status OK:

```
[root@master-int-1 ~]# yum install ruby193-hieragen
...
Updated:
  ruby193-hieragen.x86_64 0:0.2.2-39

Complete!
[root@master-int-1 ~]# hieragen --version
hieragen version 0.2.2
```

TestCase Ref. ID	1_INTc
Datum und Zeit	13.12.2015 18:50 Uhr
Involvierte Komponenten	Puppet Master PRD, Hieragen
Äussere Einflüsse	-
Genauer Testablauf	

1. Login via Linux Jump Host auf Puppet Master PRD (master-prd-1.internaldomain.net)
2. yum clean all
3. yum install ruby193-hieragen
4. hieragen –version

Resultat Test Status OK:

```
[root@master-prd-1 ~]# yum install ruby193-hieragen
...
Updated:
  ruby193-hieragen.x86_64 0:0.2.2-39

Complete!
[root@master-prd-1 ~]# hieragen --version
hieragen version 0.2.2
```

TestCase Ref. ID	2_INTa
Datum und Zeit	13.12.2015 19:16 Uhr
Involvierte Komponenten	Puppet Master DEV, Hieragen
Äussere Einflüsse	-

Genauer Testablauf

1. Login via Linux Jump Host auf Puppet Master DEV (master-dev-1.internaldomain.net)
2. hieragen configure
3. Eingabe /etc/hieragen/input/testdata_dev.json
4. Eingabe /var/log
5. Eingabe /etc/hieragen/output
6. Eingabe https://gitlab.internaldomain.ch/raffaela.wuest/master-dev-1.git

Resultat Test Status OK:

```
Your sourcelocation is /etc/hieragen/input/testdata_dev.json
logfilepath is /var/log/
outputlocation is /etc/hieragen/output/
gitrepository is
    https://gitlab.internaldomain.ch/raffaela.wuest/master-dev-1.git

##############################
successfully saved to file.
see /etc/hieragen/hieragen.conf
##############################
```

TestCase Ref. ID	2_INTb
Datum und Zeit	13.12.2015 19:30 Uhr
Involvierte Komponenten	Puppet Master INT, Hieragen
Äussere Einflüsse	-

Genauer Testablauf

1. Login via Linux Jump Host auf Puppet Master INT (master-int-1.internaldomain.net)
2. hieragen configure
3. Eingabe /etc/hieragen/input/testdata_int.json
4. Eingabe /var/log
5. Eingabe /etc/hieragen/output
6. Eingabe https://gitlab.internaldomain.ch/raffaela.wuest/master-int-1.git

Resultat Test Status OK:

```
Your sourcelocation is /etc/hieragen/input/testdata_int.json
logfilepath is /var/log/
outputlocation is /etc/hieragen/output/
gitrepository is
    https://gitlab.internaldomain.ch/raffaela.wuest/master-int-1.git

##############################
successfully saved to file.
see /etc/hieragen/hieragen.conf
##############################
```

TestCase Ref. ID	2_INTc
Datum und Zeit	13.12.2015 19:30 Uhr
Involvierte Komponenten	Puppet Master PRD, Hieragen
Äussere Einflüsse	-

Genauer Testablauf

1. Login via Linux Jump Host auf Puppet Master PRD (master-prd-1.internaldomain.net)
2. hieragen configure
3. Eingabe /etc/hieragen/input/testdata_prd.json
4. Eingabe /var/log
5. Eingabe /etc/hieragen/output
6. Eingabe https://gitlab.internaldomain.ch/raffaela.wuest/master-prd-1.git

Resultat Test Status OK:

```
Your sourcelocation is /etc/hieragen/input/testdata_prd.json
logfilepath is /var/log/
outputlocation is /etc/hieragen/output/
gitrepository is
    https://gitlab.internaldomain.ch/raffaela.wuest/master-prd-1.git

##############################
successfully saved to file.
see /etc/hieragen/hieragen.conf
##############################
```

TestCase Ref. ID	3_INTa
Datum und Zeit	15.12.2015 18:45 Uhr
Involvierte Komponenten	Puppet Master DEV, Hieragen
Äussere Einflüsse	-

Genauer Testablauf

1. Login via Linux Jump Host auf Puppet Master DEV (master-dev-1.internaldomain.net)
2. hieragen run

Resultat	Test Status OK:

```
[root@master-dev-1 ~]# hieragen run
Starting Data Collection ...
Data Collection successfully finished...
Starting YAML File Creation ...
Starting save Data for /etc/hieragen/output/nodes/
  torswitch-dev-1.internaldomain.net.yaml...
Starting render and load YAML ...
Starting save Data for /etc/hieragen/output/nodes/
  torswitch-dev-2.internaldomain.net.yaml...
Starting render and load YAML ...
Starting save Data for /etc/hieragen/output/nodes/
  spineswitch-dev-1.internaldomain.net.yaml...
Starting render and load YAML ...
Starting save Data for /etc/hieragen/output/nodes/
  spineswitch-dev-2.internaldomain.net.yaml...
Starting render and load YAML ...
Starting save Data for /etc/hieragen/output/nodes/
  mgmtswitch-dev-1.internaldomain.net.yaml...
```

TestCase Ref. ID	3_INTb
Datum und Zeit	15.12.2015 19:00 Uhr
Involvierte Komponenten	Puppet Master INT, Hieragen
Äussere Einflüsse	-
Genauer Testablauf	

1. Login via Linux Jump Host auf Puppet Master INT (master-int-1.internaldomain.net)
2. hieragen run

Resultat	Test Status OK:

```
  mgmtswitch-dev-2.internaldomain.net.yaml...
Starting render and load YAML ...
Process finished...
Successfully Uploaded...

#Hieragen run INT
[root@master-int-1 ~]# hieragen run
Starting Data Collection ...
Data Collection successfully finished...
Starting YAML File Creation ...
Starting save Data for /etc/hieragen/output/nodes/
  torswitch-int-1.internaldomain.net.yaml...
Starting render and load YAML ...
Starting save Data for /etc/hieragen/output/nodes/
  torswitch-int-2.internaldomain.net.yaml...
Starting render and load YAML ...
Starting save Data for /etc/hieragen/output/nodes/
  spineswitch-int-1.internaldomain.net.yaml...
```

TestCase Ref. ID	3_INTc
Datum und Zeit	15.12.2015 19:17 Uhr
Involvierte Komponenten	Puppet Master PRD, Hieragen
Äussere Einflüsse	-
Genauer Testablauf	

1. Login via Linux Jump Host auf Puppet Master PRD (master-prd-1.internaldomain.net)
2. hieragen run

Resultat

Test Status OK:

```
  spineswitch-int-2.internaldomain.net.yaml...
Starting render and load YAML ...
Starting save Data for /etc/hieragen/output/nodes/
  mgmtswitch-int-1.internaldomain.net.yaml...
Starting render and load YAML ...
Starting save Data for /etc/hieragen/output/nodes/
  mgmtswitch-int-2.internaldomain.net.yaml...
Starting render and load YAML ...
Process finished...
Successfully Uploaded...

#Hieragen run PRD
[root@master-prd-1 ~]# hieragen run
Starting Data Collection ...
Data Collection successfully finished...
Starting YAML File Creation ...
Starting save Data for /etc/hieragen/output/nodes/
  torswitch-prd-1.internaldomain.net.yaml...
```

Konfigurationsdaten Automatisierung für Arista Switches

Projektmanagement

3. Januar 2016

Hochschule Luzern - Technik & Architektur

Industrieprojektarbeit Infromatik (PAWI)

Autorin:
Raffaela Wüst
Swisscom (Schweiz) AG
Ey 10
3063 Ittigen
raffaela.wuest@swisscom.com

Betreuender Dozent:
Prof. Andreas Kurmann
Hochschule Luzern - T&A
Technikumstrasse 21
6048 Horw
andreas.kurmann@hslu.ch

Inhalt

1. Einleitung 1

2. Projektplanung 2
2.1 Meilensteine 2
2.2 Sprints 3
2.3 Sprint Planung 6

3. Risikomanagement 9
3.1 Risikobewertung 9
3.2 Risiken 9
3.3 Eingetroffene Risiken 11

4. Retrospective 12
4.1 Sprint 1 - Projektinitialisierung, Planung und Anforderungen 12
4.2 Sprint 2 - Systemdesign und Konzept 12
4.3 Sprint 3 - Implementation CI/CD Umgebung 13
4.4 Sprint 4 - Implementation Konfigurationsdatengenerator 13
4.5 Sprint 5 - Integration und Testing 13
4.6 Sprint 6 - Abschluss Dokumentation & Vorbereitung Präsentation 14

5. Reporting 15

6. Arbeitsjournal 16

Abbildungsverzeichnis 18

Tabellenverzeichnis 18

1. Einleitung

Das vorliegende Dokument umfasst das Projektmanagement für das PAWI Projekt „Konfigurationsdaten Automatisierung für Arista Switches“. In Kapitel zwei wird die agile Projektplanung mit den Arbeitspaketen und Meilensteinen aufgeführt. Diese kann sich während dem Projekt verändern, da vor jedem Sprint eine kurze Planungssession stattfindet. Das dritte Kapitel umfasst das Risikomanagement mit einer Risikobewertung und den möglichen Projektrisiken. Im Kapitel Reporting werden die Soll- und Ist-Stunden für die einzelnen Sprints dokumentiert. Die Retrospective dient als Rückblick auf den letzten Sprint, daraus wird Positives für den nächsten Sprint übernommen und Negatives in den nachfolgenden Sprints zu vermeiden versucht. Im letzten Kapitel Arbeitsjournal werden die effektiv verrichteten Arbeiten mit Datum und Stunden erfasst.
Alle Abkürzungen und Fachbegriffe werden im Dokument „Konzept - Konfigurationsdatengenerator für Arista Switches“ geführt.

2. Projektplanung

Die Projektplanung wird mit app.teamweek.com erstellt und gepflegt. Eine aufbereitete Version der einzelnen Sprint Planungen steht dem Leser in diesem Dokument zur Verfügung.

2.1 Meilensteine

Die Meilensteine (Tabelle 1) dienen als Kontrollpunkte während dem Projekt. Anhand des Erreichens dieser Meilensteine ist ersichtlich, wie weit das Projekt bereits fortgeschritten ist und wo allenfalls Risiken während der Projektdauer entstehen.

Tabelle 1: Meilensteine

Meilenstein	Beschreibung	Datum	Status
1	Abschluss der ersten Version der Anforderungen, diese werden agil während den Sprints gepflegt.	03.10.2015	erfüllt
2	Systemdesign fertiggestellt und bereit für die Implementation. Alle offennen Fragen sind geklärt.	20.10.2015	erfüllt
3	CI/CD Umgebung ist erstellt und funktioniert einwandfrei. Die Arista Switches sind deployed, haben ihre korrekte Konfiguration über Puppet erhalten und beziehen ihre Packages vom Pulp Repository.	20.11.2015	erfüllt
4	Der Konfigurationsdatengenerator ist zu 90% funktionsbereit, kennt seine Datensource und kann Hieradata für Arista Switches schreiben und in ein Git Repository hochladen.	12.12.2015	erfüllt
5	Die Resultate der Systemtests stimmen mit den Anforderungen an das Projekt überein.	16.12.2015	erfüllt
6	Die Schlussdokumentation ist für die Abgabe bereit, das heisst, sie wurde auf Rechtschreibung geprüft und entspricht den Anforderungen für die Abgabe der Arbeit.	04.01.2015	erfüllt

2.2 Sprints

Die Sprint Planung wird im Teamweek (app.teamweek.com) geführt. Nachfolgende Abbildungen (1 bis 7) sind Auszüge aus Teamweek. Im Kapitel 2.3 „Sprint Planung“ werden die Arbeitspakete der einzelnen Sprints genauer beschrieben.

Sprint 1 „Projektinitialisierung, Planung & Anforderungen“ dauert vom 23. September bis 03. Oktober 2015 und umfasst die folgenden wichtigen Termine:

- 23.09.15: Projektstart
- 23.09.15: Besprechung Aufgabenstellung mit Herr Kurmann
- 30.09.15: Coaching Termin HSLU
- 03.10.15: Meilenstein 1: Abschluss Anforderungen & Planung
- 04.10.15: Retrospective Sprint 1
- 04.10.15: Planning Sprint 2

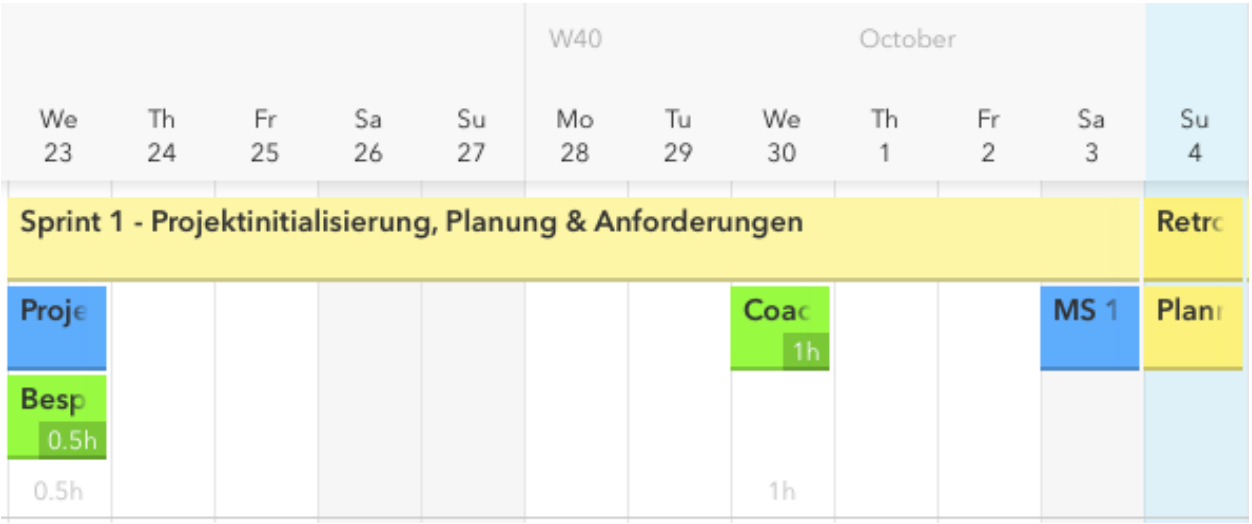


Abbildung 1: Auszug Planung Sprint 1

Sprint 2 „Systemdesign & Konzept“ dauert vom 05. Oktober bis 19. Oktober 2015 und umfasst die folgenden wichtigen Termine:

- 20.10.15: Meilenstein 2: Systemdesign fertiggestellt
- 20.10.15: Retrospective Sprint 2
- 20.10.15: Planning Sprint 3

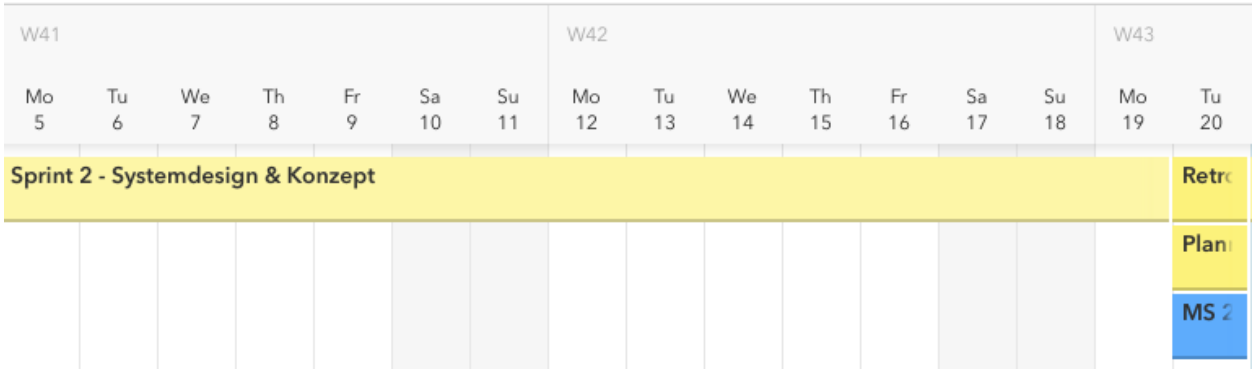


Abbildung 2: Auszug Planung Sprint 2

Sprint 3 „Implementation CI/CD Umgebung“ dauert vom 21. Oktober bis 19. November 2015 - ist somit der längste Sprint - und umfasst die folgenden wichtigen Termine:

- 23.10.15: Besprechung mit Herr Kurmann
- 06.11.15: Besprechung mit Herr Kurmann
- 20.11.15: Meilenstein 3: CI/CD Umgebung fertiggestellt
- 20.11.15: Retrospective Sprint 3
- 20.11.15: Planning Sprint 4
- 20.11.15: Besprechung mit Herr Kurmann

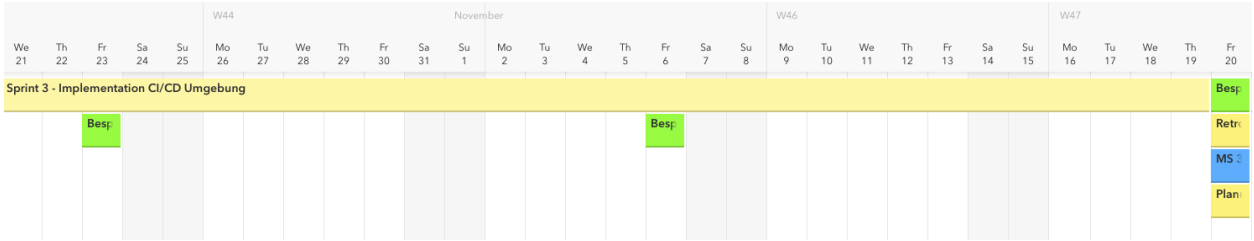


Abbildung 3: Auszug Planung Sprint 3

Sprint 4 „Implementation Konfigurationsdatengenerator“ dauert vom 21. November bis 10. Dezember 2015 und umfasst die folgenden wichtigen Termine:

- 04.12.15: Besprechung mit Herr Kurmann
- 11.12.15: Meilenstein 4: Konfigurationsdatengenerator zu 90% fertiggestellt
- 11.12.15: Retrospective Sprint 4
- 11.12.15: Planning Sprint 5

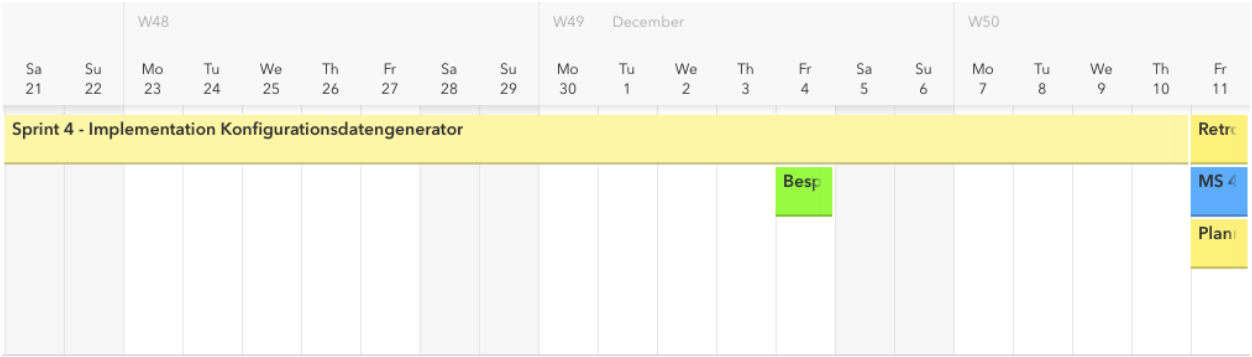


Abbildung 4: Auszug Planung Sprint 4

Sprint 5 „Integration und Testing“ dauert vom 12. Dezember bis 15. Dezember 2015 und umfasst die folgenden wichtigen Termine:

- 16.12.15: Meilenstein 5: Resultate Systemtests
- 16.12.15: Retrospective Sprint 5
- 16.12.15: Planning Sprint 6

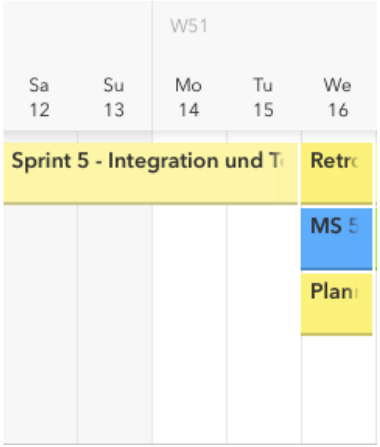


Abbildung 5: Auszug Planung Sprint 5

Sprint 6 „Abschluss Dokumentation & Vorbereitung Präsentation“ dauert vom 17. Dezember bis 27. Dezember 2015 und umfasst die folgenden wichtigen Termine:

- 17.12.15: Besprechung mit Herr Kurmann
- 18.12.15: Retrospective Sprint 6

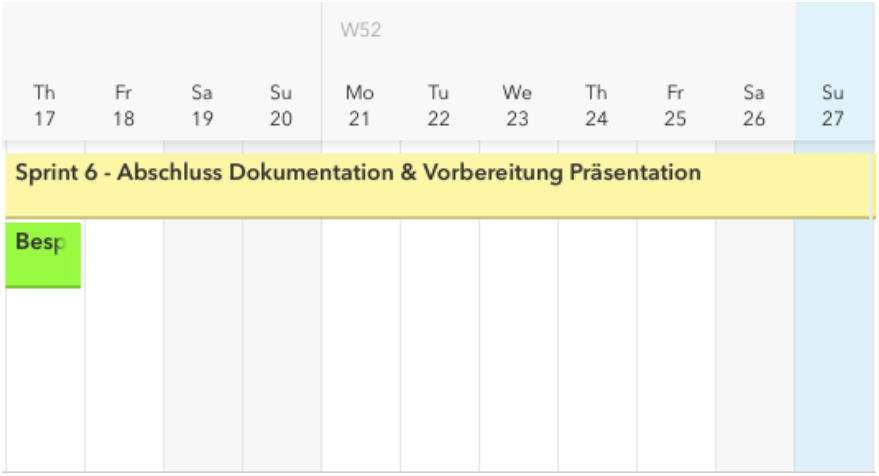


Abbildung 6: Auszug Planung Sprint 6

Der Projektabschluss dauert vom 29. Dezember 2015 bis 02. Januar 2016 und umfasst die folgenden wichtigen Termine:

- 04.01.15: Abgabe Schlussdokumentation

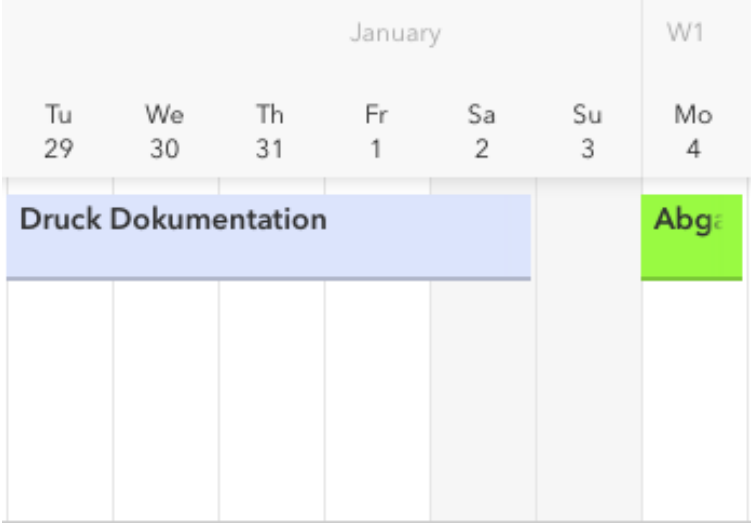


Abbildung 7: Auszug Planung Projekt Abschluss

2.3 Sprint Planung

Die Sprint Planung wird laufend vor dem jeweiligen Sprint ergänzt und erweitert. Die Arbeitspakete fürs Projektmanagement (Retrospective und Planung) werden nicht zu den Soll-Stunden des Sprints gezählt. Total sind 15 Stunden für Projektmanagement über die Gesamtplanung berechnet worden.

2.3.1 Sprint 1 - Arbeitspakete (15h)

1.1	Initialisierung der Projektdokumente	2.5h
1.2	Risikomanagement	2h
1.3	Grobplanung	3h
1.4	Requirements Engineering (User Stories & Use Cases	7.5h
	Projektmanagement (Retrospektive und Detail Planung Sprint 2)	2.5 h

2.3.2 Sprint 2 - Arbeitspakete (15h)

1.2	Risikomanagement	1h
2.1	Grobkonzept fertigstellen	2h
2.2	Systemdesign CI/CD Umgebung	3.5h
2.3	Systemdesign Konfigurationsdatengenerator	6.5h
2.4	Testing (Ausarbeitung, Planung und Vorbereitung)	2h
	Projektmanagement (Retrospektive und Detail Planung Sprint 3)	2.5 h

2.3.3 Sprint 3 - Arbeitspakete (40h)

3.1	Hieradaten erstellen (Management Umgebung mit Puppet Master, Pulp und Jenkins und initiale Hieradaten für Switches)	11h
3.2	Infrastruktur bereitstellen (VM's)	1h
3.3	Connectivity gewährleisten (zwischen VM's und zu Schnittstellensysteme Gitlab und EnClouder)	6h
3.4	Deployment (Management Umgebung und Arista Switches)	12h
3.5	Initiale (manuelle) Konfiguration von Arista Switches	5h
3.6	Testing (Ausarbeitung Testcases und durchführen der Unit Tests)	5h
	Projektmanagement (Retrospektive und Detail Planung Sprint 4)	2.5 h

2.3.4 Sprint 4 - Arbeitspakete (65h)

4.1	Hieradaten Templates für alle Switchtypen mit Embedded Ruby erstellen	15h
4.2	Data Generation (Sammeln und Aufbereiten der Source Daten)	15h
4.3	YAML File Creator (erstellen des YAML Files)	10h
4.4	YAML Syntax Validator (File auf Syntaxerrors überprüfen)	3h
4.5	Uploader (upload des erstellten Files ins Gitlab Hieradata Repo)	2h
4.6	Konfiguration und Errorhandling für Generator	10h
4.7	Commandline User Interface	5h
4.8	Testing (Ausarbeitung Testcases und durchführen der Unit Tests)	5h
	Projektmanagement (Retrospektive und Detail Planung Sprint 5)	2.5 h

2.3.5 Sprint 5 - Arbeitspakete (30h)

5.1	Integration des Konfigurationsdatengenerators in die CI/CD Umgebung	15h
5.2	Testing (durchführen Systemintegrationstests und Dokumentation)	15h
	Projektmanagement (Retrospektive und Detail Planung Sprint 6)	2.5 h

2.3.6 Sprint 6 - Arbeitspakete (40)

6.1	Finalisierung der Dokumentation	30h
6.2	Rechtschreibkorrektur und Druck	10h
	Projektmanagement (Retrospektive)	2.5 h

3. Risikomanagement

Das Risikomanagement setzt sich aus drei Kapiteln - 3.1 Risikobewertung, 3.2 Risiken, 3.3 Eingetroffene Risiken - zusammen.

3.1 Risikobewertung

Alle Risiken im Kapitel 3.2 werden nach Eintrittswahrscheinlichkeit (klein = 1, mittel = 2, gross = 3) und Auswirkung (klein = 1, mittel = 2, gross = 3) bewertet. Die Risikogewichtung (Eintrittswahrscheinlichkeit x Auswirkung) klassifiziert die einzelnen Risiken.

3.2 Risiken

Die Risiken in der folgenden Tabelle werden während der ganzen Projektdauer neu bewertet und falls nötig überarbeitet.

Tabelle 3: Risiken

Nr.	Risiko	Massnahmen	Eintrittsw.	Auswirkung	Risikogewichtung
1.	Missverständnisse bei den Anforderungen. Die Anforderungen werden falsch, ungenau oder zu wenig detailliert aufgenommen. Der Entwickler muss bei der Entwicklung Annahmen treffen die zu Fehlentscheidungen führen können.	die Anforderungen müssen vor jedem Sprint überprüft und mit dem Auftraggeber abgeglichen werden.	2	3	6

2.	Umsetzung nicht in der vorgegebenen Zeit möglich. Alle geplanten Features können während der Projektdauer nicht umgesetzt werden, weil zu wenig Zeit vorhanden ist. Das erfolgreiche Abschliessen der Projekts steht in Frage.	Eine genaue Sprint Planung und eine detaillierte Retrospektive sind wichtig, sodass die Umsetzung der geplanten Arbeitspakete während den Sprints möglich ist. Die frühzeitige Abgrenzung von Minor Features, falls diese den Zeitrahmen sprengen, und insbesondere das hoch priorisieren der Major Features ist wichtig. Bemerkung 20.11.15 die Eintrittswahrscheinlichkeit wurde auf 3 erhöht, der Sprint 3 hat einiges länger gedauert als ursprünglich geplant.	3	2	6
3.	Fehlendes Know How der Studierenden. Das fachliche Know How der Studentin reicht für die Projektarbeit nicht aus, es treten unnötige Fehler auf und die Fehlersuche nimmt sehr viel Zeit in Anspruch.	Die Studentin soll sich frühzeitig bei grösseren Problemen an den Auftraggeber oder einen Spezialisten der Firma wenden.	2	1	2

4.	Zuwenige Informationen aus Datensource für die Hieradaten Der EnClouder als Datasource für Arista Switches kann nicht alle nötigen Informationen liefern, die für das korrekte Setup der Puppet Role benötigt werden.	Der EnClouder kann als Datasource nur teilweise verwendet werden. Die restlichen Daten werden über die Commandline Eingabe, eine weitere Datensource, oder über Mock-Daten zusammengetragen. Bemerkung: Es wurde entschieden,	3	2	6
5.	Fehler im Setup der CI/CD Umgebung Die CI/CD Umgebung wird fehlerhaft aufgebaut, weil die Hieradaten nicht korrekt sind, Packages fehlen oder die Puppet Module Fehler aufweisen.	Die Hieradaten sollen mit bestehenden Systemen abgeglichen werden. Packages müssen auf den vorhanden Pulp Server transferiert und Fehler in Puppet Roles möglichst schnell an die zuständigen Personen gemeldet werden.	1	3	3

3.3 Eingetroffene Risiken

Das Risiko Nummer vier „Zuwenige Informationen aus Datensource für die Hieradaten“ ist während der Projektdauer eingetroffen. Der EnClouder als Datasource für Arista Switches konnte nicht alle nötigen Informationen zur Hieradaten Generierung liefern. An dieser Stelle kam als Zwischenlösung eine einfache Text Datei als Datensource zum Einsatz. In welcher Form der EnClouder zukünftig die Sourcedaten den Generatoren zur Verfügung stellen wird, ist in Abklärung. Die Zeitplanung des Projekts wurde durch das eingetroffene Risiko nicht beeinträchtigt und konnte Termingerecht abgeschlossen werden.

4. Retrospective

In der Retrospective wird ein Rückblick über den jeweiligen Sprint gemacht. Es wird festgehalten, was erledigt werden konnte und welche Anforderungen nicht umgesetzt wurden, was positiv und was negativ war.

4.1 Sprint 1 - Projektinitialisierung, Planung und Anforderungen

Der Sprint 1 umfasste die Projektinitialisierung, die Planung, sowie die Aufnahme der ersten Anforderungen für das Projekt. Während dem Projekt Coaching am 30. September 2015 an der HSLU wurde die Grobplanung und der erste Entwurf des Grobkonzepts mit den Zielen und Resultaten der Projektarbeit diskutiert und von den Dozierenden kritisch hinterfragt. Jeder StudentIn / jede Gruppe musste sich für eine Projektart entscheiden. Die Projektartenwahl für den Konfigurationsdatengenerator fiel auf ein IT-Infrastrukturprojekt mit einem Teil Individual-Software-Entwicklung (Generator). Das Projekt wird nach SODA geplant und durchgeführt.
Die Anforderungen werden als User Stories erfasst und als Use Cases ausgearbeitet. Da das Projekt agil geführt wird, sind die User Stories noch nicht komplett und werden vor jeder Sprint Planung überarbeitet und ergänzt.
Fazit des ersten Sprints, die Projektinitialisierung konnte abgeschlossen werden, die Grobplanung wurde erstellt und im Teamweek (app.teamweek.com) geführt, und die ersten Anforderungen sind erfasst. Das Risikomanagement konnte noch nicht abgeschlossen werden. Es wird im Sprint 2 als erster Task aufgenommen.

4.2 Sprint 2 - Systemdesign und Konzept

Um für die Projektarbeit eine solide Grundlage zu schaffen, wurde im Sprint 2 das Systemdesign und das Konzept ausgearbeitet. Die User Stories wurden verwendet um die Systemkomponenten zu ermitteln und daraus ein entsprechendes Systemdesign zu gestalten. Während dem Sprint konnte das Verständnis für das Projekt vertieft und kritische Fragen aufgeworfen werden. So wurde das Risiko Nummer vier (Zuwenig Informationen aus Datensource für die Hieradaten) zum konkreten Thema. Höchstwahrscheinlich wird bis zur Implementation des Generators der EnClouder nicht über alle nötigen Daten für Arista Switches verfügen. Im Konzept wurde die Auswirkung des Risikos bereits berücksichtigt. Voraussichtlich wird eine minimale Webapplikation oder eine einfache Textdatei als Datensource verwendet. Um den späteren Umstieg auf den EnClouder zu vereinfachen, soll die Datensource zudem über eine REST API angesprochen werden können. Dieser Faktor erweitert das Projekt um einige Stunden. Die genaue Planung für den Mehraufwand wurde noch nicht kalkuliert. Aufgrund hoher Belastung der Studentin durch andere Projekte in der Firma konnte der Sprint nicht plangemäss abgeschlossen werden und wird um vier Tage, bis 21.10.2015 verlängert. Die Planung für die nachfolgenden Sprints wird überarbeitet, wobei der Endtermin nicht gefährdet ist.

4.3 Sprint 3 - Implementation CI/CD Umgebung

Die Aufgabe in diesem Sprint, eine komplette CI/CD Umgebung aufzubauen und in einen lauffähigen Modus zu bekommen stellte ein besonders lehrreiches und interessantes Unterfangen dar. Bisher wurden die einzelnen Komponenten von der Studierenden jeweils genutzt, jedoch nie von Grund auf selbständig aufgebaut. Der erste Knackpunkt - welcher Zeit gekostet hat - war die Beschaffung der IP-Adressen für die 24 einzusetzenden Server. Da der IP Range bereits am Limit war musste vorgängig in der zu nutzenden Labor Umgebung aufgeräumt werden, sodass nicht mehr gebrauchte IP-Adressen wieder freigegeben, und für dieses Projekt benutzt werden konnten. Sobald dieser Punkt geklärt war, konnte zügig mit dem Setup der Management Umgebung gestartet werden. Dieses Setup verlief problemlos. Als Hilfestellung dienten die Hieradaten bereits bestehender Systeme. Beim Setup der Arista Switches war das Vorgehen etwas schwieriger. Da in dieser Arbeit auf virtuelle Arista Switches gesetzt wird - und in der Firma bisher keine Erfahrung mit vEOS vorhanden war - musste experimentiert werden. Beim Setup der virtuellen Switches gab es weitere Herausforderungen. So konnte das Betriebssystem nur erfolgreich mit 2GB Memory booten. Bei der Arista Switch Konfiguration und manueller Installation des Puppet Agents stand ein Mitarbeiter für Fragen und Antworten zur Verfügung. Daher konnte auch dieser Part ohne grosse vEOS Kenntnisse gemeistert werden. Schliesslich steht nun eine komplette CI/CD Umgebung gemäss Spezifikation, die darauf wartet, eingesetzt zu werden.

4.4 Sprint 4 - Implementation Konfigurationsdatengenerator

Das Programmieren des Konfigurationsdatengenerators, in der Programmiersprache Ruby, ermöglichte der Studierenden sich in einem komplett neuen Umfeld Wissen aufzubauen. Da auf diesem Gebiet kein Know-How vorhanden war, gebührte diesem Sprint besonderen Respekt. Die an der HSLU vermittelten Grundlagen der objektorientierten Programmierung in Java, sowie die bereits erlernten Puppet Kenntnisse (Anlehnung an Ruby), ermöglichten jedoch einen schnellen Einstieg und ein guten Verständnis für die benötigten Konzepte von Ruby.
Der Generator entstand in kleinen Schritten und wuchs zu einem einfachen, aber kompletten System heran, um erfolgreich Hieradaten für Arista Switches zu generieren. Als Hilfe und Wegleiter für die Erstellung des Codes dienten Beispiele aus Eigenentwicklungen oder Codeausschnitte vom Internet. Zum Erstaunen der Studentin konnte der Generator termingerecht abgeschlossen und der Meilenstein 4 voll und ganz erfüllt werden.

4.5 Sprint 5 - Integration und Testing

Der Sprint 5 „Integration und Testing“ mit der Integration des Generators in den Buildprozess war eine neue Herausforderung. Diese konnte bewältigt werden, da intern entwickelte Software zu RPM Packages mittels DSL Jobs auf anderen Jenkins gebuildet werden. So konnte auf bereits bestehende Scripts zugegriffen und mit leichten Modifizierungen lauffähig gemacht werden.
Als kurzfristiger Entscheid, wurden die Templates als eigenständiges RPM Package gebuildet. Der Hintergrund zu dieser Entscheidung war die Wiederverwendbarkeit des Generators. Weitere Templates erfordern dadurch keine Neupaketierung der Software. Die Installation auf dem Puppet Master war

ein einfaches Unterfangen. Die zwei gebuildeten Packages standen auf dem Pulp Server zur Verfügung und konnten somit via YUM installiert werden.
Das Testkonzept des Generators wurde überarbeitet. So entstanden während dem Sprint 4 diverse Unit Tests mit dem Ruby Framework Rspec, welche die Funktionalität der einzelnen Klassen und deren Methoden überprüfen. Die Unit Tests können über einen Jenkins Job ausgeführt werden, dieser gibt die Resultate aller Tests zurück.
Zusätzlich wurden Integrationstests durchgeführt. Sie zeigen das Zusammenspiel der CI/CD Umgebung mit dem Generator auf. Diese Tests verliefen wie gewünscht und deshalb kann auch dieser Sprint als erfolgreich betrachtet werden.

4.6 Sprint 6 - Abschluss Dokumentation & Vorbereitung Präsentation

Im letzten Sprint dieser Projektarbeit wurde die komplette Dokumentation überarbeitet und mit den letzten fehlenden Kapiteln ergänzt. Es wurde ein Abstract zur Arbeit erfasst und ein übergeordnetes Titelblatt erstellt. Die Arbeit wurde auf Herz und Nieren geprüft, das heisst, die Rechtschreibung und Satzstellungen wurden überarbeitet, Tabellen und Bildbeschriftungen kontrolliert und die komplette Gestaltung mit Anpassung der Bilder und Absätzen bearbeitet. Als zusätzliche Qualitätskontrolle wurde die Arbeit von Frau Vreni Wüst gegengelesen und auf allfällig weitere Fehler geprüft, bevor sie termingerecht gedruckt werden konnte.
Die Vorbereitungen für die Abschlusspräsentationen konnten nicht komplett abgeschlossen werden.

5. Reporting

Das Reporting (Tabelle 5) ist eine Zusammenfassung und Übersicht aus den geleisteten Arbeiten und Stunden des Arbeitsjournals.

Tabelle 5: Reporting

Sprint	Soll (h)	Ist (h)	Diff
Sprint 1 - Initialisierung, Planung, Anforderungen	15	17	+ 2
Sprint 2 - Systemdesign und Konzept	15	20.5	+ 5.5
Sprint 3 - Implementation CI/CD Umgebung	40	46.5	+ 6.5
Sprint 4 - Implementation Generator	50	71	+ 21
Sprint 5 - Integration & Testing	30	34	+ 4
Sprint 6 - Abschluss und Dokumentation	40	41	+ 1
Planning und Retrospektive	15	12	- 3
Total	205	242	+ 37

6. Arbeitsjournal

Das Arbeitsjournal beinhaltet alle geleisteten Arbeiten, zugeordnet zu den einzelnen Arbeitspaketen. Eine gesamte Projektübersicht über die aufgewendeten Stunden ist im Kapitel Reporting zu finden.

Tabelle 6: Arbeitsjournal

Datum	AP	Tätigkeit	Stunden
Vorbereitungen			
23. September 2015	1.1	Besprechung Aufgabenstellung mit Dozent	0. 5
24. September 2015 - 25. September 2015	1.1	Ausarbeitung Aufgabenstellung	3
30. September 2015	1.1	Coaching	1
Dokumentationen			
25. September 2015	1.1	Dokumentationsstruktur und Templates erstellen	5
12. November 2015 - 19. November 2015	3	Dokumentation der CI/CD Implementation	10
20. November 2015 - 10. Dezember 2015	4	Dokumentation Konfigurationsdatengenerator	9
17. Dezember 2015	6.1	Besprechung mit Herr Kurmann	0.5
20. November 2015 - 10. Dezember 2015	5	Dokumentation Integration und Testing	5
17. Dezember 2015 - 27. Dezember 2015	6.1	Finalisierung Dokumentation	24.5
27. Dezember 2015	6.2	Rechtschreibkorrektur	10
29. Dezember 2015	6.2	Druck & Binden	6
Planung & Projektmanagement			
25. September 2015	1.3	Grobplanung	1
04. Oktober 2015		Retrospektive und Planning Sprint 2	1.5
06. Oktober 2015 - 20. Oktober 2015	1.2	Risikomanagement	1.5
20. Oktober 2015		Retrospektive und Planning Sprint 3	2.5
20. November 2015		Retrospektive und Planning Sprint 4	3
11. Dezember 2015		Retrospektive und Planning Sprint 5	3.5
16. Dezember 2015		Retrospektive und Planning Sprint 6	2.5 h
27. Dezember 2015		Retrospektive	0.5 h
Anforderungen			
26. September 2015 - 03. Oktober 2015	1.4	Requirements Engineering	5
Konzept und Systemdesign			
07. Oktober 2015 - 19. Oktober 2015	2.2	Systemdesign CI/CD	5
09. Oktober 2015 - 10. Oktober 2015	2.2	Produkte und Technologien	5
09. Oktober 2015 - 19. Oktober 2015	2.3	Systemdesign Konfigurationsdatengenerator	10

23. Oktober 2015	2	Besprechung mit Herr Kurmann	0.5
Implementation CI/CD Umgebung			
26. Oktober 2015 - 30. Oktober 2015	3.1	Hieradaten erstellen	8.5
06. November 2015	3	Besprechung mit Herr Kurmann	0.5
06. November 2015	3.2	Deployment Infrastrkutur	1.5
10. November 2015	3.2	Erstellen des Deployment Plans für Arista Switches	2
17. November 2015	3.2	Redeployment Arista Switches	0.5
08. November 2015	3.3	Connectivity Kontrolle	0.5
11. November 2015	3.1	Referenzhieradaten erstellen	4
14. November 2015 - 15. November 2015	3.4	Deployment Management Umgebung	8.5
18. November 2015	3.5	Initial Konfiguration Arista Switches	3
18. November 2015	3.5	Puppet Installation Arista Switches	2
19. November 2015	3.5	Puppet Installation Arista Switches	2
20. November 2015	3	Besprechung mit Herr Kurmann	0.5
14. November 2015 - 20. November 2015	3.6	Testing CI/CD Umgebung	3
Implementation Konfigurationsdatengenerator			
21. November 2015 - 28. November 2015	4.2	Data Generation (Sammeln und Aufarbeiten von der Datensource)	17
28. November 2015 - 29. November 2015	4.3	YAML File Creator (erstellen des YAML Files)	6
30. November 2015 - 04. Dezember 2015	4.1	Hieradaten Templates für alle Switchtypen mit Embedded Ruby erstellen	12
30. November 2015	4.6	Errorhandling für Generator implementieren	1.5
01. Dezember 2015 - 02. Dezember 2015	4.6	Konfiguration für Generator implementieren	3
03. Dezember 2015	4.7	Implementation Commandline User Interface	2
04. Dezember 2015	4.5	Git Uploader Klasse programmieren	6
04. Dezember 2015	4	Besprechung mit Herr Kurmann	0.5
07. Dezember 2015 - 09. Dezember 2015	4	Code Refactoring	7
10. Dezember 2015	4.8	Unit Tests Generator	7
Integration und Testing			
03. Dezember 2015 - 09. Dezember 2015	5.1	Jenkins Buildjob für Generator	7
03. Dezember 2015 - 09. Dezember 2015	5.1	Jenkins Buildjob für Templates	8
12. Dezember 2015	5.1	Installation Generator	0.5
12. Dezember 2015 - 13. Dezember 2015	5.2	Unit Testing	4
13. Dezember 2015 - 15. Dezember 2015	5.2	Integration Testing	9.5
Total			242

Abbildungsverzeichnis

1 Auszug Planung Sprint 1 3
2 Auszug Planung Sprint 2 4
3 Auszug Planung Sprint 3 4
4 Auszug Planung Sprint 4 5
5 Auszug Planung Sprint 5 5
6 Auszug Planung Sprint 6 6
7 Auszug Planung Projekt Abschluss 6

Tabellenverzeichnis

1 Meilensteine 2
3 Risiken 9
5 Reporting 15
6 Arbeitsjournal 16