Projektdokumentation

Einführung eines zentralen Versionskontrollsystems für Serverkonfigurationen und Projekte

von
Florian Baumann
Ausbildungsberuf: Fachinformatiker SI
Berufsschule 1, IFS12b
Bayreuth
florian.baumann@tmt.de

Ausbildungsbetrieb
TMT
Technik.Medien.Telekommunikation.
TMT GmbH & Co. KG
Nürnbergerstraße 42
95448 Bayreuth
tech@tmt.de

1 Projektumfeld

1.1 Firmenvorstellung

Die Firma „TMT GmbH & Co. KG“ (im weiteren Verlauf „TMT“) in Bayreuth deckt 3 wichtige Bereiche moderner Kommunikation ab. TMT bietet somit eine abgestimmte und ganzheitliche Produktpalette in den Bereichen „Webdevelopment und Design“, „Call Center“ und „IT- und Netzwerk-Sicherheit“. Derzeit sind etwa 50 Mitarbeiter beschäftigt. Aufgabe der Abteilung „IT- und Netzwerk-Sicherheit“, der ich während diesem Projekt angehörte, ist die Pflege der mittlerweile mehr als 250 Server (95% davon mit der Betriebssystembasis Linux) und Realisierung von netzwerkbezogenen Kundenaufträgen sowie Einrichtung und Betreuung der hauseigenen Produktlinie TMT-blueHost.

1.2 Beschreibung des Umfeldes

Mit zunehmender Größe der Rechenzentren von TMT wächst auch der Konfigurationsumfang und -aufwand der Server. Immer mehr Mitarbeiter arbeiten parallel an Systemen und bearbeiten Konfigurationen. Die Änderungen betreffen meist Dienste, deren einwandfreie Funktion für Kunden oder auch für TMT von großer Wichtigkeit sind. So werden beispielsweise HTTP-Daemons, Mail-Server, Proxies, Firewalls, Individualsoftware der Kunden oder auch größere hoch redundante Systeme von TMT gepflegt und gewartet.

Minimale Anpassungen an diesen Diensten können große Veränderungen mit sich bringen, die selbst mit stabilster Software und bestem Fachwissen nicht vorhersehbar sind. Um eine lückenlose Dokumentation sowie Funktionstüchtigkeit für die Systeme zu gewährleisten wird die Einrichtung einer zentralen Verwaltungsstelle in Erwägung gezogen.

2 Projektdefinition

Um in der Zukunft getätigte Änderungen an Konfigurationsdateien zu erfassen und nachvollziehbar zu machen, wurde ich in einer internen Besprechung mit der Aufgabe betraut, ein zentrales Versionskontrollsystem (nachfolgend „VCS“) für diesen Zweck zu konzipieren und umzusetzen. Die datei- und änderungsorientierten VC-Systeme bieten mit unter anderem einer Änderungsübersicht und Versionshistorie alle erforderlichen Eigenschaften einer Konfigurationsverwaltung. Mit Hilfe dieses Werkzeugs sollen oben genannte Merkmale professioneller Systembetreuung sichergestellt werden.

Als Teil dieser Aufgabe verstand sich die Anschaffung von Hardware, Auswahl geeigneter Software und Inbetriebnahme des Systems. Das Projektziel stellte die Einrichtung des Versionskontrollsystems für die Pflege von Softwareprojekten und Konfigurationen innerhalb einer Woche dar.

3 Ist-Analyse

Die Rechenzentren der TMT GmbH & Co. KG und Kundenstandorte sind geographisch voneinander getrennt (Siehe Anhang S. 11 TMT Netzwerk). Administration und Pflege werden per SSH (Secure Shell) vorgenommen, wodurch sich gute Wartbarkeit und schnelle Anpassungsmöglichkeiten ergeben. SSH bietet also zugleich den optimalen (bereits vorhandenen) Kommunikationskanal für die Vernetzung durch das Versionskontrollsystem.

Da ebenso Softwareentwicklung zu der Produktpalette von TMT gehört, soll auch die Verwaltung derer ihren Platz im neuen Setup finden. Die, für Pflege und Betreuung von Softwareprojekten ausgelegten, *Source Control Manager* eignen sich hervorragend für eine zentrale Sammelstelle des Software Developments.

4 Soll-Konzept

4.1 Anforderungen

Die Anforderungen an das neue VCS sind in folgenden Punkten definiert.

- Lokale Versionierung der Server-Konfigurationsverzeichnisse /etc/
- Wiederherstellbarkeit vorheriger Konfigurationen
- Dokumentation der Änderungen in Form von Changelogs
- Zusätzlich zentrale Sammelstelle lokaler Versionierungen
- Verwaltungsmöglichkeit zukünftiger (Software-)Projekte
- Benutzerverwaltung mit Authentifizierung
- Verschlüsselte Übertragung der Daten

4.2 Hardware

Die Hardware für das VCS ging aus Beständen von TMT hervor, welche die benötigten Eigenschaften bereitstellte. Ausschlaggebend für die Entscheidung waren große Festplatten mit schnellen Zugriffszeiten und Gigabit Netzwerkkarten des Servers.

Server	Typ	HP ProLiant DL160 G6
	Größe	19 Zoll 1HE
	CPU	Intel Xeon X5677 3.46GHz
	Arbeitsspeicher	4GB
	Festplatten	4x 250GB SAS (RAID5)
	Netzwerkkarten	2x 1000-Base-T

4.3 Software

Die Auswahl der Software erforderte mehr Aufwand. Die Kombination derer musste alle Anforderungen wie Authentifizierung, Sicherheit, Flexibilität und Effizienz erfüllen. Zugleich aber einfach zu bedienen sein um die Arbeit aller Administratoren zu erleichtern und entsprechendes Know-How weiter vermitteln zu können. Die Wahl fiel auf das Versionskontrollsystem *git*, weil das neue aufstrebende VCS von Linus Torvalds sich großer Beliebtheit in Entwicklung und Dokumentation erfreut. Ebenso haben Kollegen und ich selbst durchgehend positive Erfahrungen mit der Software gemacht. Weiterhin kommen folgende Softwarebausteine (siehe Anhang S. 11 VCS Software) zum Einsatz.

Software	Betriebssystem	Debian GNU/Linux 5.0 (Lenny)
	Version Control System	git-core
	Lokale Verwaltung	etckeeper
	Authentifizierung	gitolite & ssh

5 Planung der Projektschritte

5.1 Strukturelle und zeitliche Planung

Abgeleitet aus den geplanten Anforderungen entstand folgende strukturelle und zeitliche Einteilung.

Nr.	Tätigkeit	Std.
1	Ist-Analyse	2
2	Ermittlung des Soll-Zustandes	3
3	Vorbereitung der Hardware	1
4	Installation des Betriebssystems	2
5	Einrichtung und Konfiguration des VCS	8
6	Inbetriebnahme des Servers	2
7	Erstellung des Client-Skripts	7
8	Server in das Versionskontrollsystem hinzufügen	1
9	Test des Systems	2
10	Schulung der Mitarbeiter und Dokumentation	7
	Gesamt	35

5.2 Qualitative Nutzwertanalyse

Um die Wahl der Software zusätzlich zu untermauern diente außerdem noch eine Nutzwertanalyse. Der Vergleich erfolgte mit ähnlichen aktuellen Versionsverwaltungssystemen, aus dem git eindeutig als bevorzugte Software hervorgeht.

	Gewichtung	**subversion**	**git-core**	**bazaar**
Geschwindigkeit	5	1	4	3
Stabilität	3	2	5	4
Bedienbarkeit	4	4	2	2
Dokumentation	5	4	5	3
Speicherbedarf	4	1	5	3
Erlernbarkeit	2	4	2	3
Funktionsumfang	4	3	5	4
Gesamt		**70**	**102**	**84**

6 Projektrealisierung

6.1 Vorbereitung der Hardware

Nachdem alle Planungsarbeiten abgeschlossen waren, konnte mit der Realisierung des Projekts begonnen werden. Die ausgewählte Hardware wurde an die Arbeits- und Installationsumgebung von TMT angeschlossen und betriebsfertig eingerichtet. Nach Einbau der vier 250GB SAS Festplatten in deren Einschübe konnte mit der Erstellung des Raidsystems begonnen werden. Ein RAID5 sollte durch seine Partitätsmengen für zusätzliche Ausfallsicherheit sorgen. Außerdem ermöglicht es durch Splitten der Daten erhöhte Schreib- und Lesegeschwindigkeit. Über den RAID-Controller des Servers ließ sich dieses über die 4 HDDs „bauen“.

6.2 Installation des Betriebssystems

Das Betriebssystem sollte, wie bei den meisten von TMT betreuten Servern, Debian GNU/Linux 5.0 sein. Der Installationsvorgang über eine sogenannte „NetInstall“-CD verlief wie gewohnt durch die Stationen Lokalisierung, Partitionierung, Paketauswahl und Benutzereinrichtung. Alle dieser Stationen wurden dem internen Leitfaden gemäß durchlaufen. Eine zusätzliche größere Partition war allerdings für die Daten der Versionsverwaltung vorgesehen.

/boot	512MB
/	45GB
/srv/git	700GB
swap	4GB

Als geeignete Organisation der Archive auf Filesystemebene bot sich die Sturkturierung nach Abteilungen an. Unter den Aspekten des Soll-Zustandes ergaben sich drei Hauptgruppen die innerhalb des VCS unter */srv/git/repositories/* abgebildet werden mussten (Siehe Anhang 12 VCS Filesystem).

- tech - Technik Softwareprojekte und Skripte
- web - Projekte aus Webdesign und Webentwicklung
- hosts - Konfigurationsverzeichnisse der Server

Für die Realisierung dieses Schemas und deren Zugriffsrechte ist *gitolite* zuständig. Die Orderstruktur wurde wie folgt erstellt.

```
$ mkdir -p /srv/git/repositories/{tech,web,hosts}
```

6.3 Einrichtung und Konfiguration des VCS

6.3.1 git

Grundlegende Pakete der zukünftigen Versionsverwaltung sind in der Debian Paketverwaltung vorhanden. Der Grundbaustein *git-core*, der für das Management der Daten sowohl auf den Hosts als auch auf dem VCS-Server zuständig ist, konnte also mit Hilfe des Paketmanagers *apt* installiert werden.

```
$ aptitude install git-core
```

Git erfordert bei jedem Arbeitsschritt die Identifikation des Benutzers. Deshalb sollten grundlegende Informationen vor Benutzung konfiguriert werden.

```
$ git config --global user.name "TMT Admin"
$ git config --global user.email "vcs@tmt.de"
```

Auch weitere Anpassungen der Config Datei von git sind für Erhöhung der Benutzbarkeit und Leserlichkeit zu empfehlen (Siehe Anhang S. 13 git Konfigurationsdatei).

6.3.2 gitolite

Die Authentifizierung des VCS übernimmt das Zugangskontrollsystem *gitolite*. gitolite benutzt einen dedizierten SSH-Benutzer um Zugang zu den Git Repositories zu schaffen. Meldet sich ein Benutzer mittels seines SSH-Schlüsselpaares an, kontrolliert gitolite die Anmeldung über eine Konfigurationsdatei in der die Rechtevergabe geregelt ist (Siehe Anhang S. 11 VCS Software und S. 12 VCS Topologie). Sie stellt die Schnittstelle zwischen den Clients und dem zentralen Server dar.

Die Einrichtung erfolgte ebenfalls über ein Debian Paket.

```
$ aptitude install gitolite
```

Nach der erfolgreichen Installation folgt die Einrichtung des Authentifizierungssystems. *gitolite* muss zuerst mit einem *SSH Public Key* (vorzugsweise dem des Administrators) initialisiert werden. Diese erste Initialisierung stellt die Möglichkeit der Konfiguration durch den Administrator bereit.

Mit den folgenden Kommandozeilen wurde mein bereits bestehender SSH Public Key auf das Serversystem kopiert, der entsprechende Benutzer angelegt und *gitolite* mit dem kopierten Key eingerichtet.

```
# Desktop: fbaumann@fbaumann-desktop
$ cp /home/fbaumann/.ssh/id_rsa.pub root@vcs.tmt.de:/tmp/fbaumann.pub

# Server: root@vcs.tmt.de
$ adduser --home /srv/git/ git
$ su git
$ gl-setup /tmp/fbaumann.pub
```

Somit war der Key fbaumann.pub berechtigt das gitolite Administrationsrepository zu klonen und Änderungen zu übernehmen. Standardmäßig heißt dieses Repository *gitolite-admin*. Dass die Konfiguration von *gitolite* selbst ebenfalls in einem Git Repository erfolgt hat mehrere Vorteile. Mehrere Mitarbeiter können so ohne direkten SSH Zugang parallel daran arbeiten und außerdem verfügt es über alle Eigenschaften einer Versionsverwaltung wie unter anderem Wiederherstellbarkeit und einer History.

Wie bei jedem git Repository werden Änderungen daran lokal getätigt. Ein Klon von gitolite-admin wird also auf dem Arbeitsplatzrechner benötigt.

```
$ cd /home/fbaumann
$ git clone git@vcs.tmt.de:gitolite-admin gitolite-admin
```

Die Verbindung zum VCS-Server wird aufgebaut und mit Hilfe des SSH-Schlüsselpaares sichergestellt, dass der Benutzer berechtigt ist das Repository zu klonen. In dem neu entstandenen Ordner befinden sich folgende Dateien und Unterordner:

```
$ tree gitolite-admin/
gitolite-admin/
|-- conf
|   `-- gitolite.conf
`-- keydir
    `-- fbaumann.pub
```

Im Ordner *keydir* werden alle öffentlichen Schlüssel aufbewahrt welche Zugang auf das VCS erhalten sollen. Vorerst befindet sich dort nur der Schlüssel mit dem gitolite initalisiert worden ist, nämlich *fbaumann.pub*.

Das Verzeichnis *conf* enthält nur die generelle Konfigurationsdatei *gitolite.conf*. Diese folgt einer speziellen Syntax in welcher Repositories und deren zugehörige Rechte hinterlegt werden können. Im Filesystem befinden sich diese im Home-Verzeichnis des Benutzers *git* im Ordner *repositories*.

```
$ cat conf/gitolite.conf

repo    gitolite-admin
        RW+     =   fbaumann
```

Der Eintrag definiert also die Zugriffsrechte für */srv/git/repositories/gitolite-admin* mit Lese- und Schreibrecht für den Benutzer *fbaumann*.

Um die Konfiguration möglichst einfach und dynamisch zu halten, empfahl sich die Einführung von Benutzergruppen. Die nötigen Public Keys meiner Kollegen sind bereits an einer zentralen Stelle gesammelt und waren somit einfach in *keydir* kopiert.

```
$ tree keydir/
keydir
|-- fbaumann.pub
|-- mitarbeiter1.pub
|-- mitarbeiter2.pub
|-- mitarbeiter3.pub
|-- mitarbeiter4.pub
|-- techleiter1.pub
`-- webleiter1.pub
```

Die nun dem System bekannten Mitarbeiter konnten demnach in der Konfigurationsdatei verwendet und Gruppen zugeordnet werden.

```
$ cat conf/gitolite.conf

# usergroups
@admins   =   fbaumann techleiter1
@tech     =   fbaumann techleiter1 mitarbeiter1 mitarbeiter2
```

```
@web      =   webleiter1 mitarbeiter3 mitarbeiter4

repo    gitolite-admin
        RW+     =   @admins
```

Alle Mitglieder der Gruppe *admins* wären jetzt berechtigt Änderungen an dem Verwaltungsrepository *gitolite-admin* vorzunehmen. Da aber neben Softwareentwicklung durch Mitarbeiter auch Konfigurationsverzeichnisse der Server vorgehalten werden sollen, ist es nötig jedem Server seinen eigenen Public Key zu generieren. Sie stellen sich gegenüber dem Authentifizierungssystem als separater Benutzer dar. Die Serververwaltung bekommt aus Gründen der Übersichlichkeit ein eigenes Konfigurationsfile.

```
$ cat conf/hosts.conf

# /etc repository server1
repo    hosts/server1.tmt.de
        RW+     =   server1.tmt.de @tech

# /etc repository server2
repo    hosts/server2.tmt.de
RW+     =   server2.tmt.de @tech

[...]
```

Aus Sicherheitsgründen werden die öffentlichen Schlüssel der Server nicht in einer Gruppe zusammengefasst. Sollte eine dieser Maschinen durch Angriffe fremder Kontrolle unterworfen werden, wäre es so ausgeschlossen Zugriff auf andere Konfigurationsarchive zu erlangen.

Nach diesem Schema konnte die Vervollständigung der gitolite Konfiguration nun zu Ende geführt werden. Die vollständige Konfiguration befindet sich im Anhang Seite 13 (gitolite Konfiguration).

6.4 Inbetriebnahme des Servers

Anschließend wurde das fertig konfigurierte Versionskontrollsystem in eines der Rechenzentren transportiert und entsprechend in Betrieb genommen. Nach Überprüfung der Konnektivität und der Sicherheitseinstellungen konnte der Client bezogene Teil des Projekts vollzogen werden.

6.5 Erstellung des Client Skripts

Um die tatsächliche Benutzung des VCS auf den Servern zu ermöglichen, stehen bestimmte Anforderungen aus. Diese müssen auf jedem Server bestehen, der unter

Versionskontrolle steht.

- git und etckeeper müssen installiert und konfiguriert sein
- SSH Schlüsselpaar muss generiert und im VCS hinterlegt sein

Da diese nur begrenzt fehlerfrei und zeitsparend per Hand ausgeführt werden konnten, fiel die Entscheidung diese Vorraussetzungen mit Hilfe eines Shell Skriptes zu implementieren. Das Skript hatte nicht nur die Aufgabe der Installation und Einrichtung, sondern musste auch in der Lage sein systemspezifische Einstellungen (wie z.B. den Hostnamen) in die Software *etckeeper* zu integrieren.

Außerdem ist die Sicherheit und Stabilität solcher Skripte sehr wichtig, da durch falsch interpretierte Umgebungen und Tatsachen schnell Schaden entstehen kann. Hierzu wurde bei Unklarheiten auch auf Benutzereingaben zurückgegriffen um die Korrektheit zu verifizieren.

Generell sieht das Skript folgenden Ablauf vor:

- Auslesen und Sammeln von Informationen (installierte Software? Schlüsselpaar vorhanden?)
- Verifikation der gesammelten Informationen durch den Benutzer
- Ausführen den Einrichtung und Konfiguration des Servers

Nach mehrfachen Tests in virtuellen Maschinen unter Rücksichtnahme diverser Eventualitäten konnte das Skript schliesslich fertiggestellt werden. Es befindet sich im Anhang Seite 14 (Host Skript).

6.6 Server in das Versionskontrollsystem hinzufügen

Die Einrichtung eines Servers gestaltete sich nach Erstellung des Skriptes sehr einfach. Im Grunde musste dieses nur von einer Quelle bezogen und ausgeführt werden.

```
# Beispiel server1.tmt.de
$ wget -O /tmp/config-vcs-host.bash http://vcs.tmt.de/config-vcs-host.bash
$ chmod +x /tmp/config-vcs-host.bash
$ ./tmp/config-vcs-host.bash

host configuration
=======================================
hostname: server1.tmt.de
gitserver: vcs.tmt.de
pubkey for auth: /root/.ssh/id_rsa.pub
git-core install: no
etckeeper install: yes
```

```
copy pubkey: yes
key destination: root@vcs.tmt.de:/tmp/server1.tmt.de.pub
=======================================
configure host? (y/N) y
configuring...
[...]
```

So ist es auch später Kollegen möglich auf einfache Weise ein neu aufgesetztes System der Versionskontrolle hinzuzufügen. Unter Zuhilfenahme der Software ClusterSSH, durch welche sich viele Server zeitgleich steuern lassen, verlief die Installation durch das Shell-Skript fehlerfrei und zeitsparend. Darauf folgte die serverseitige Anpassung von gitolite, um die neuen Archive zugänglich zu machen.

Schließlich befanden sich gemäß den Projektanforderungen alle Systemkonfigurationsverzeichnisse (*/etc*) aller Server unter Versionskontrolle. Der Arbeitsfluss innerhalb der Serverpflege konnte nun folgendermaßen aussehen.

```
# Kunden Mailserver Einstellungen anpassen
$ vi /etc/postfix/main.cf
[...]
$ etckeeper commit "Aenderungen an Mail Daemon vorgenommen -- fbaumann"

# FTP Server des TMT Hosting Paketes
$ useradd --home /home/ftp/ftpbenutzer3 ftpbenutzer3
$ vi /etc/passwd
$ vi /etc/vsftpd.conf
$ etckeeper commit "FTP-User ftpbenutzer3 konfiguriert -- mitarbeiter2"
$ git push origin master

# Zurücksetzen des Webservers wegen Fehlkonfiguration des Apache2
$ git log
[...]
$ git reset --hard 20ee2c2273c2b56e245c7c3e1cd2f86773797b3c
```

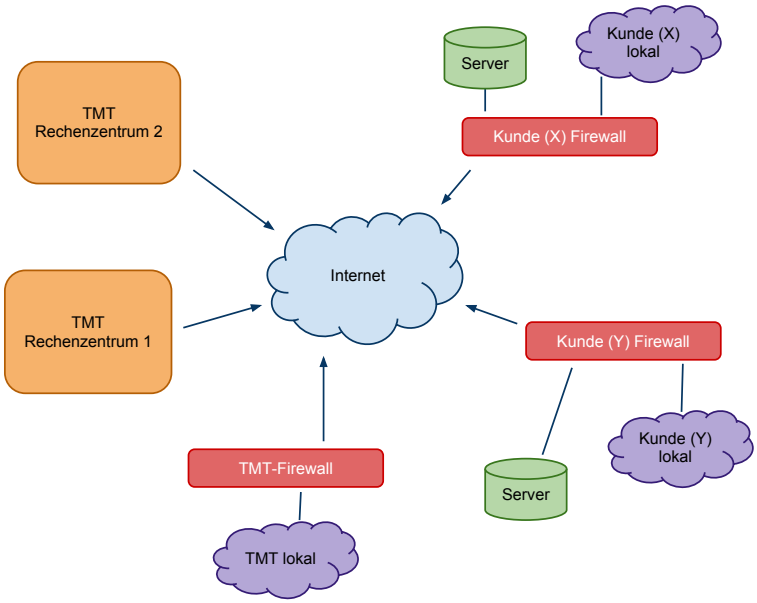
Außerdem ermöglicht der Einsatz von *etckeeper* die Automatisierung von Routinen. So wird beispielsweise nach jeder Aktion des Paketmanagers *aptitude* automatisch ein *Commit* erstellt welcher die Details der (De-)Installation und deren Änderungen kurz beschreibt ohne Benutzereingaben zu benötigen.

7 Schlussbetrachtung

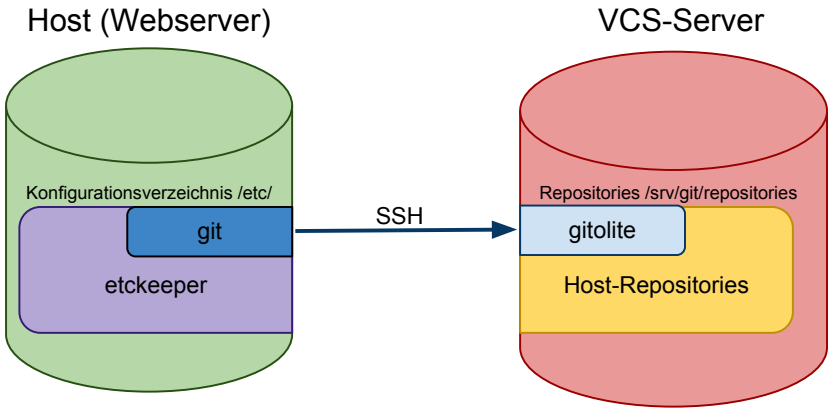
Das von mir im Rahmen meiner betrieblichen Ausbildung bei der TMT GmbH & Co.KG durchgeführte Projekt konnte erfolgreich abgeschlossen werden. Das VCS wurde nach anschließender Schulung der Mitarbeiter der Abteilungen „IT- und Netzwerk-Sicherheit“ und „Webdevelopment und Design“ fest in den Arbeitsablauf integriert und stellt seither einen großen Nutzen im Bereich der professionellen Systembetreuung dar. Insgesamt wurde der Zeitrahmen eingehalten.

8 Anhang

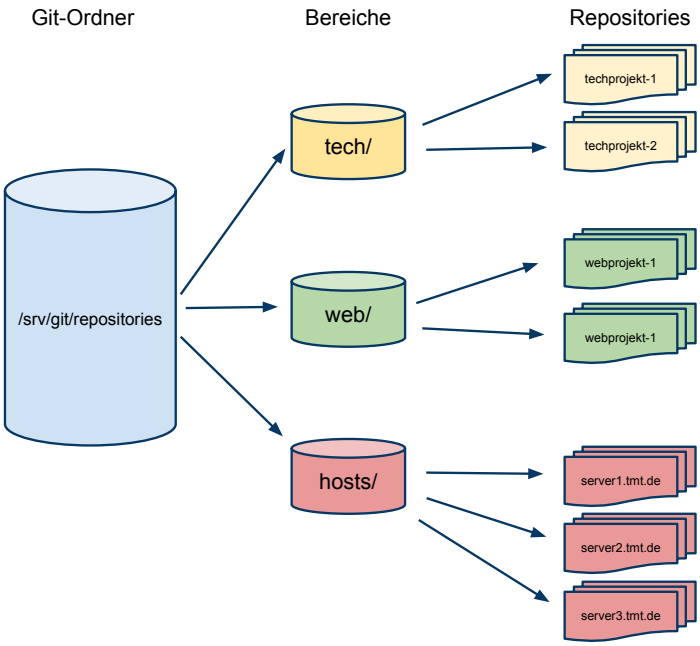
8.1 TMT Netzwerk



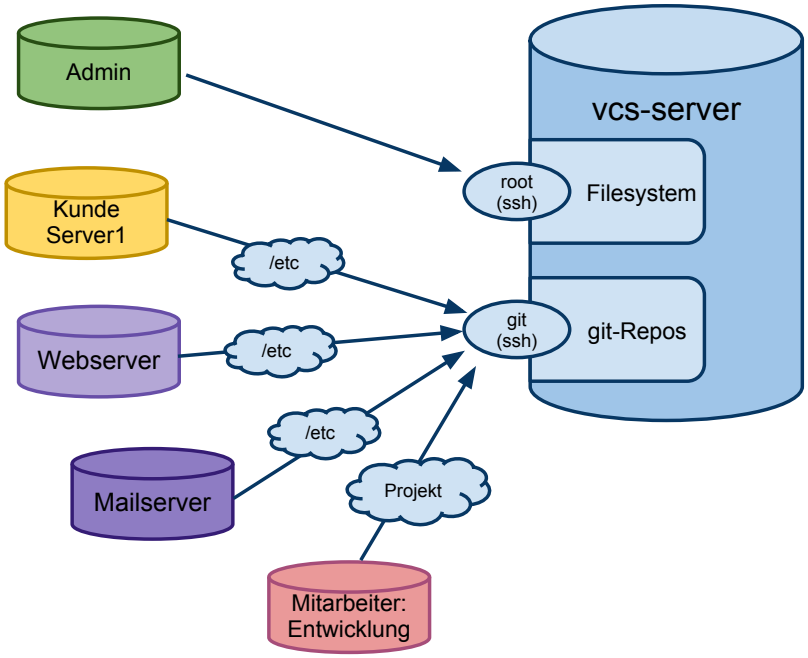
8.2 VCS Software



8.3 VCS Filesystem



8.4 VCS Topologie



8.5 git Konfigurationsdatei

```
[color]
    status = auto
    branch = auto
    diff = auto

[alias]
    ci = commit -a -m
    co = checkout
    br = branch
    st = status -a
    praise = blame
    ll = log --pretty=oneline --abbrev-commit --max-count=15
    cdiff = diff --cached

[color "diff"]
    meta = blue bold
    frag = magenta bold
    old = red bold
    new = green bold

[color "branch"]
    current = yellow reverse
    local = yellow bold
    remote = green bold
    plain = red bold

[color "status"]
    added = yellow
    changed = green bold
    untracked = blue bold
```

8.6 gitolite Konfiguration

8.6.1 gitolite.conf

```
# usergroups
@admins   =   fbaumann techleiter1
@tech     =   fbaumann techleiter1 mitarbeiter1 mitarbeiter2
@web      =   webleiter1 mitarbeiter3 mitarbeiter4

# admin repository
repo    gitolite-admin
        RW+     =   @admins

# tech repos
repo  tech/techprojekt-1
```

```
RW+ =  @admins @tech

repo  tech/techprojekt-2
RW+ =  @admins @tech
R =  @web

# web repos
repo  web/webprojekt-1
RW+ =  @admins @web

repo  web/webprojekt-2
RW+ =  @admins @web
R =  @tech

# hosts
include "hosts.conf"
```

8.6.2 hosts.conf

```
# server1
repo    hosts/server1.tmt.de
        RW+     =   server1.tmt.de

# server2
repo    hosts/server2.tmt.de
        RW+     =   server2.tmt.de

# server3
repo    hosts/server3.tmt.de
RW+     =   server3.tmt.de
```

8.7 Host Skript

```
#!/bin/bash
# config-vcs-host.bash: configures host for etckeeper, git
#  version control and specific git-remote. use with care.
#  serverside config is just one simple line in gitolite.
# Copyright: (C) 2011 Florian Baumann <florian.baumann@tmt.de>
# License: GPL-3 <http://www.gnu.org/licenses/gpl-3.0.txt>

### configuration ##############################################################
gitserver=vcs.tmt.de
gitserverkeydir=/tmp
localpubkey=/root/.ssh/id_rsa.pub
host=$(hostname --fqdn)
copykey=yes

### security ###################################################################
```

```
# root ?
if [ "$UID" -ne 0 ]; then echo "need to be root" ; exit 1; fi
# git installed ?
if [ $(dpkg -l | grep -c '^ii  git-core\W') -lt 1 ]; then
    installgit=yes
else
    installgit=no
fi
# etckeeper installed ?
if [ $(dpkg -l | grep -c '^ii  etckeeper\W') -lt 1 ]; then
    installetckeeper=yes
else
    installetckeeper=no
fi
# already configured for gitserver?
if [ -e /etc/.git ]; then
    echo "etckeeper seems to be already configured";
    exit 1
fi
#fqdn hostname
if [[ ! "$host" =~ .*\..*\..*  ]]; then
    echo "hostname doesn't look like FQDN"
    echo "please edit /etc/hosts"
    exit 1
fi

# ctrl+c exit
trap interrupt INT
interrupt() {
        echo -e "\ninterrupted"; exit 1
}


### print ##############################################################################
echo -e "\nhost configuration"
echo "========================================="
echo "hostname: $host"
echo "gitserver: $gitserver"
echo "pubkey for auth: $localpubkey"
echo "git-core install: ${installgit:-no}"
echo "etckeeper install: ${installetckeeper:-no}"
if [ ${copykey:-no} = yes ]; then
    echo "copy pubkey: ${copykey:-no}"
fi
echo "key destination: root@${gitserver}:/tmp/$host.pub"
echo "========================================="
echo -n "configure host? (y/N) ";  read ready
if [ ${ready:-n} = "y" ] || [ ${ready:-n} = "Y" ]; then
    echo -e "\nconfiguring..."
    echo "========================================="
else
    echo "CANCELD. maybe next time." ; exit 1
```

```
fi

### execute ##############################################################################
# install
if [ "${installgit:-no}" = "yes" ]; then
echo "installing git-core..."
    aptitude install git-core &> /dev/null
fi
if [ "${installetckeeper:-no}" = "yes" ]; then
    echo "installing etckeeper..."
aptitude install etckeeper &> /dev/null
fi

# configure /etc
cd /etc/
echo "switched to $PWD"
sed -i -e 's/^VCS="bzr"/VCS="git"/' /etc/etckeeper/etckeeper.conf
echo "etckeeper initialization..."
etckeeper init &> /dev/null
echo "committing on $host..."
/usr/bin/git commit -a -m "Inital commit on $host by config-host.bash" &> /dev/null
echo "configuring git-server $gitserver..."
/usr/bin/git remote add origin git@${gitserver}:hosts/${host}
echo -e "use the following line to push /etc \ngit push origin master"

# ssh pubkey required
echo -e "\ncopying pubkey \n========================================="
if [ "$copykey" = "yes" ] ; then
    if [ -r $localpubkey ] ; then
        echo "copying pubkey to $gitserver"
        scp $localpubkey root@${gitserver}:/tmp/$host.pub
    else
echo "no pubkey available. create? (y/N) ";  read create
if [ ${create:-n} = "y" ] || [ ${create:-n} = "Y" ]; then
    echo -e "creating ssh keypair..."
ssh-keygen -t rsa
fi
        echo "copying pubkey to $gitserver"
        scp $localpubkey root@${gitserver}:/tmp/$host.pub
echo "returned: $? public key copied"
    fi
fi
echo "host $host successfully configured!"
```

8.8 Software

Software	Lizenz	Quelle
aptitude	GPL	http://wiki.debian.org/Aptitude
bash	GPL	http://tiswww.case.edu/php/chet/bash/bashtop.html
git-core	GPL	http://git-scm.com
gitolite	GPL	https://github.com/sitaramc/gitolite
ssh	BSD	http://openssh.org/
LaTeX	LPPL	http://www.latex-project.org/

8.9 Quellen

https://github.com/sitaramc/gitolite/blob/pu/doc/2-admin.mkd
https://github.com/sitaramc/gitolite/blob/pu/doc/1-INSTALL.mkd
http://progit.org/book/
http://www.ostc.de/howtos/ssh-HOWTO.txt
http://tools.ietf.org/rfc/rfc4252.txt
http://git.kitenet.net/?p=etckeeper.git;a=blob;f=README

8.10 Glossar

Abkürzung	Beschreibung
BASH	Bourne-again Shell
CSSH	Cluster Secure Shell
SCM	Source Control Manager
SSH	Secure Shell
VCS	Version Control System
GPL	General Public License
BSD	Berkeley Software Distribution
LPPL	LaTeX Project Public License
HTTP	Hypertext Transfer Protocol
HE	Höheneinheit
SAS	Serial Attached SCSI
SCSI	Small Computer System Interface
RAID	Redundant Array of Independent Disks

8.11 Selbstständigkeitserklärung

Hiermit versichere ich, dass ich diese Projektdokumentation selbstständig und unter ausschließlicher Nutzung der angegebenen Quellen und Hilfsmittel erstellt habe. Sollten Teilarbeiten, welche zur Projektrealisierung notwendig waren, durch weitere Personen durchgeführt worden sein, so ist dies ausdrücklich gekennzeichnet.

Desweiteren wurden aus Sicherheitsgründen alle auf IPs oder DNS-Namen verweisende Inhalte entsprechend abgeändert.

Diese Dokumentation wurde mit Hilfe des Textsatzprogramms LaTeX erstellt.