Inhalt

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 5 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "ucsencs.def" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

Kapitel 1
Versionskontrolle

»Zurück in die Zukunft«

Versionskontrollsysteme oder auch Versionsverwaltungssysteme, im Englischen *Version Control Systems* (VCS), bieten die Möglichkeit, »in die Vergangenheit zu reisen« und alte Stände von Dateien wiederherzustellen sowie verschiedene Versionen von Dateien miteinander zu vergleichen. Versionskontrollsysteme werden überwiegend in der Softwareentwicklung zur Verwaltung von Quelltexten eingesetzt, leisten aber auch in der Systemadministration zur Verwaltung von Konfigurationsdateien gute Dienste.

In diesem Kapitel werden Versionskontrollsysteme nur als Mittel zur Verwaltung von Konfigurationsdateien behandelt. Insbesondere wird davon ausgegangen, dass nur wenige Personen an den Dateien arbeiten. Das ganze Aufgabenfeld rund um Konflikte beim Zusammenführen (*Mergen*) von unterschiedlichen Versionsständen wird dabei bewusst ausgespart.

Die Systeme erfüllen dabei folgende Vorgaben:

- **Archivierung**
  von alten Ständen, zum Nachvollziehen, wie sich eine Konfiguration entwickelt hat, und um alte Stände bei Bedarf wiederherstellen zu können
- **Protokollierung**
  Es kann jederzeit geprüft werden, wer zu welcher Zeit was verändert hat.
- **Wiederverwendbarkeit**
  Auch wenn eine alte Konfiguration auf einem neuen System eventuell nicht mehr das tut, was sie tun soll, so kann sie dennoch als Vorgabe für Konfigurationen auf Altsystemen dienen. Das ist ein Spezialfall der Archivierung.

Um diese Vorgaben zu erfüllen, haben sich in den letzten Jahren verschiedene Sichtweisen gebildet, und innerhalb der verschiedenen Sichtweisen existieren unterschiedliche Implementationen der Sichtweisen.

Im Großen und Ganzen lassen sich drei verschiedene Philosophien – *lokal*, *zentral* und *dezentral* – unterscheiden, die in Abschnitt 1.1, »Philosophien«, näher beschrieben werden.

Es gibt eine ganze Reihe an Versionskontrollsystemen, die als Open-Source-Software veröffentlicht wurden, und es existieren ebenfalls einige Vertreter aus dem Lager der Closed-Sour-

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 7 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

ce-Software. In Tabelle 1.1 finden Sie einige Beispiele für die verschiedenen Versionskontrollsysteme, die der Wikipedia[1] entnommen wurden.

	Open-Source-Systeme	Proprietäre Systeme
Zentrale Systeme	SCCS RCS CVS Subversion (SVN)	Alienbrain Perforce Team Foundation Server Visual SourceSafe ClearCase IBM Ration Synergy PTC Integrity SAP Design in Time Repository (DTR) versiondog Sourcegear Vault
Verteilte Systeme	Bazaar Bitkeeper Darcs Fossil Git GNU arch Mercurial Monotone	Rational Team Concert

Tabelle 1.1 Beispiele für Versionskontrollsysteme aus der Wikipedia

1.1 Philosophien

Es gibt viele verschiedene Ansätze für die Versionskontrolle, aber im Großen und Ganzen lassen sich drei verschiedene Philosophien unterscheiden, die wir im Folgenden betrachten.

1.1.1 Lokal

Alle Versionen von Dateien werden lokal gespeichert und nicht an einen Server übertragen. Als einfachstes Beispiel mag Listing 1.1 dienen:

```
$ cp config config.alt
$ edit config
```

Listing 1.1 Einfachste Art der lokalen Versionskontrolle

1 *https://de.wikipedia.org/wiki/Versionsverwaltung*

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 8 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

Die originale Datei *config* wird damit in *config.alt* kopiert, und die ursprüngliche Datei wird verändert. Nun existieren zwei Versionen der Datei.

Es gibt Systeme und sogar Dateisysteme, die ein solches Vorgehen unterstützen und die Dateien rotierend benennen. Aus *config* wird *config.1*, bei der nächsten Änderung wird aus *config.1* die Datei *config.2*, und aus *config* wird erneut *config.1*, so wie es Listing 1.2 zeigt:

```
$ cp config.1 config.2
$ cp config config.1
$ edit config
```

Listing 1.2 Lokale Versionskontrolle mit verschiedenen Versionen

Das lässt sich beliebig weiterdenken oder durch einen Schwellenwert – beispielsweise zwei Versionen wie in Listing 1.3 – abbrechen:

```
$ rm config.2
$ cp config.1 config.2
$ cp config config.1
$ edit config
```

Listing 1.3 Lokale Versionskontrolle, rotierend

Wenn die *config* ein weiteres Mal verändert wird, dann wird die älteste Version gelöscht. Allgemein üblich ist es, statt der Nummern Daten bzw. Zeitstempel einzusetzen. Damit ist auf einen Blick ersichtlich, bis zu welchem Zeitpunkt eine Konfigurationsdatei gültig war, wie Listing 1.4 zeigt:

```
$ cp config config.20180830
$ edit config
```

Listing 1.4 Lokale Versionskontrolle mit Zeitstempel

1.1.2 Zentral

Einen Schritt weiter gehen zentrale Versionskontrollsysteme. Dort wird mit den Begriffen *Arbeitskopie* und *Repository* unterschieden, in welchem Teil der Arbeitskette man sich befindet. Dateien werden in der Arbeitskopie verändert und nach Abschluss der Arbeiten – zusammen mit einer Nachricht – an eine zentrale Stelle, einen zentralen Server, geschickt. Der Server, der das Repository verwaltet, besitzt alle Versionsstände der verwalteten Dateien. Lokal in der Arbeitskopie liegt immer nur eine einzige Version der Datei vor. Will man sich Unterschiede zwischen verschiedenen Versionen anzeigen lassen, so wird für jede Version der Server beauftragt, die angeforderte Version zu schicken.

Der große Vorteil eines solchen Verfahrens besteht darin, dass lokal eine sehr aufgeräumte Arbeitskopie vorliegt, ohne dass man Dateien in vielen verschiedenen Versionen vorlie-

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 9 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

gen hat, und dass zentral auf dem Server ein Backup aller alten (und auch der aktuellen) Konfigurationen vorliegt. In einer Pseudosprache sieht der Arbeitsablauf so wie in Listing 1.5 beschrieben aus:

```
$ vcs get datei
** Version 42 von datei ausgecheckt **
$ edit datei
$ vcs send datei "Parameter x auf y gesetzt"
** Version 43 von datei eingecheckt, Meldung "Parameter x auf y gesetzt" **
```

Listing 1.5 Arbeitsablauf bei einem zentralen Versionskontrollsystem

Vor der Bearbeitung wird die aktuelle Version einer Datei geholt. Diese wird bearbeitet, und die bearbeitete Datei wird wieder zurück an das zentrale System gesendet. Die Versionsnummern werden von der zentralen Stelle verwaltet.

1.1.3 Dezentral

Einen der größten Nachteile von zentralen Versionskontrollsystemen, nämlich die Tatsache, dass der zentrale Server immer verfügbar sein muss, beheben dezentrale Versionskontrollsysteme. Sie gelten auch als die zeitgemäße Variante der Versionskontrolle.

Jedes ausgecheckte Repository enthält die komplette Historie aller jemals durchgeführten Änderungen in komprimierter Form. Aus diesem Grund spricht man bei der Kopie auch von *Klonen*. Unterschiede zwischen Versionen oder zwischen der aktuellen Arbeitskopie und historischen Ständen lassen sich anzeigen, ohne dass eine Verbindung zu einem zentralen Server nötig wäre.

Dieser Vorteil wird allerdings durch einen zusätzlichen Schritt erkauft. Änderungen, die bei zentralen Systemen an einen Server übertragen werden, müssen erst lokal festgeschrieben werden, bevor der aktuelle Stand an den Server übertragen wird (siehe Listing 1.6):

```
$ vcs fetch
** Synchronisierung von Aenderungen **
$ edit datei
$ vcs send datei "Parameter x auf y gesetzt"
** datei wurde lokal festgeschrieben **
$ vcs transfer
** Alle lokalen Aenderungen an Server uebertragen **
```

Listing 1.6 Arbeitsablauf bei einem dezentralen Versionskontrollsystem

Zunächst beginnt in Listing 1.6 der Arbeitsablauf mit einer Synchronisation des eigenen Repositorys mit dem entfernten. Das ist nicht zwingend nötig, da alle Änderungen auch lokal vorhanden sind, aber es erleichtert das spätere Zusammenführen von Klonen erheblich.

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 10 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

Anschließend wird die Datei bearbeitet und lokal festgeschrieben (wobei die Änderung mit einer Meldung versehen wird). In einem letzten Schritt werden alle Änderungen an einen anderen Server oder an ein anderes Repository übertragen.

1.2 Versionskontrollsysteme

Versionskontrollsysteme gibt es wie Sand am Meer. Die englischsprachige Wikipedia-Seite *https://en.wikipedia.org/wiki/Comparison_of_version_control_software* listet aktuell bereits 35 verschiedene Systeme und zeigt ihre verschiedenen Stärken und Schwächen auf. In diesem Abschnitt werden nur die fünf Systeme behandelt, die im Open-Source-Umfeld am häufigsten anzutreffen sind.

Von den zentralen Systemen stellen wir *CVS* und *Apache Subversion* vor, wobei die Nutzung von CVS sehr stark zurückgegangen ist. CVS wird auch nicht mehr aktiv weiterentwickelt. Bei den dezentralen Systemen finden sich *GNU Bazaar*, *Mercurial* und *Git*, wobei Git die weiteste Verbreitung findet, gefolgt von Mercurial und zum Schluss GNU Bazaar. Anfang 2021 sah die Verteilung auf *openhub.net* (siehe *https://www.openhub.net/repositories/compare*, einer Seite, die Auswertungen vielfältiger Open-Source-Projekte durchführt) so aus, wie Tabelle 1.2 zeigt.

Versionskontrollsystem	Herbst 2018	Anfang 2021
Git	60 %	72 %
Apache Subversion	33 %	23 %
CVS	2 %	1 %
Mercurial	1 %	1 %
GNU Bazaar	1 %	0 %

Tabelle 1.2 Verbreitung von Versionskontrollsystemen

Die Daten aus Tabelle 1.2 sind nicht repräsentativ, da es sich um selbst gemeldete und öffentliche Repositorys handelt, aber sie geben einen guten Anhaltspunkt über die Verbreitung.

1.2.1 CVS

CVS (*Concurrent Versioning System*, siehe *https://savannah.nongnu.org/projects/cvs*) begann 1989 als Weiterentwicklung von RCS, dem *Revision Control System*. CVS benötigt die Umgebungsvariable `CVSROOT`, um auf Repositorys zugreifen zu können. Ein neues Repository erzeugt man mit `cvs init` (siehe Listing 1.7):

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 11 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

```
$ export CVSROOT=/var/tmp/cvsrepository
$ cvs init
```

Listing 1.7 Initialisierung eines Repositorys mit CVS

Danach kann man beliebige Verzeichnisse in dieses Repository importieren, beispielsweise */etc*. In Listing 1.8 wird die Benutzung des `import`-Befehls gezeigt:

```
$ cd /etc
$ cvs import -m "Initialer Import" etc Dirk start
N etc/netconfig
N etc/securetty
[…]
N etc/insserv.conf.d/rpcbind

No conflicts created by this import
```

Listing 1.8 Import eines Verzeichnisses bei CVS

Verallgemeinert folgt der `import`-Befehl folgendem Muster:
`cvs import -m Log-Nachricht Modul Hersteller-Information Release-Information`

Mit den Angaben aus Listing 1.8 legen wir fest, dass das Modul *etc* heißen soll, die Herstellerinformation ist *Dirk*, und die Release-Information lautet *start*.

Der Aufbau von `CVSROOT` ist abhängig von der Art und Weise, wie man das Repository auschecken möchte: `CVSROOT=<Methode:><Name:><Verzeichnis>`.

Dabei können für *Methode* verschiedene Werte eingetragen werden. Das Feld muss leer sein, wenn ein lokal erreichbares Verzeichnis das Ziel ist. Alternativ kann es den Wert `:ext` für den Zugriff über Remote Shell oder SSH enthalten, oder es nimmt den Wert `:pserver` für den Zugriff über ein CVS-eigenes Protokoll an. *Name* setzt sich aus dem User- und dem Rechnernamen zusammen. `username@hostname` besagt, dass der CVS-Server auf dem Server *hostname* läuft und sich *username* mit diesem verbinden will. Wenn man lokal arbeitet, kann der *hostname* weggelassen werden. Wenn der aktuelle User sich verbinden soll, dann kann man den *username* ebenfalls weglassen. Zum Schluss gibt *Verzeichnis* das Hauptverzeichnis an, in dem die Repositorys verwaltet werden. Das gerade angelegte Modul wird jetzt im Homeverzeichnis des angemeldeten Benutzers wieder ausgecheckt, wie in Listing 1.9 beschrieben:

```
$ export CVSROOT=/var/tmp/cvsrepository
$ cd $HOME
$ cvs checkout etc
cvs checkout: Updating etc/python2.7
cvs checkout: Updating etc/rc0.d
cvs checkout: Updating etc/rc1.d
[…]
```


Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 12 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

```
U etc/w3m/mailcap
U etc/xml/catalog
U etc/xml/xml-core.xml
```

Listing 1.9 Checkout eines Moduls mit CVS

Mit den Dateien im ausgecheckten Modulverzeichnis kann jetzt gearbeitet werden (siehe Listing 1.10):

```
$ cd $HOME/etc
$ vim hosts
$ cvs diff hosts
Index: hosts
===================================================================
RCS file: /var/tmp/cvsrepository/etc/hosts,v
retrieving revision 1.1.1.1
diff -r1.1.1.1 hosts
1c1
< 127.0.0.1     localhost
---
> 127.0.0.1     localhost meincomputer

$ cvs commit -m "meincomputer zu hosts hinzugefuegt"
cvs commit: Examining .
[…]
cvs commit: Examining xml
/var/tmp/cvsrepository/etc/hosts,v  <--  hosts
new revision: 1.2; previous revision: 1.1
```

Listing 1.10 Arbeit mit dem ausgecheckten Verzeichnis bei CVS

In Listing 1.10 sehen Sie die einzelnen Schritte. Dem Wechsel in das Arbeitsverzeichnis folgt die Bearbeitung der Datei *hosts*. Der `cvs diff` zeigt die Unterschiede zwischen der lokalen Datei und dem Repository, schließlich wird mit `cvs commit` die Änderung an das Repository übertragen. Wie Sie feststellen können, sucht CVS im gesamten Verzeichnis nach Änderungen und würde diese alle auf einmal übertragen. Listing 1.11 zeigt alle Änderungen, die an der Datei *hosts* vorgenommen wurden:

```
$ cvs log hosts
RCS file: /var/tmp/cvsrepository/etc/hosts,v
Working file: hosts
head: 1.2
branch:
locks: strict
access list:
```


Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 13 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

```
symbolic names:
         start: 1.1.1.1
         Dirk: 1.1.1
keyword substitution: kv
total revisions: 3;     selected revisions: 3
description:
----------------------------
revision 1.2
date: 2014-07-27 19:06:22 +0200;  author: dirk;  state: Exp;  lines: +1 -1; \
      commitid: 10053D5318E0BCD08F4;
meincomputer zu hosts hinzugefuegt
----------------------------
revision 1.1
date: 2014-07-27 18:30:32 +0200;  author: root;  state: Exp;  \
      commitid: 10053D529280B56DA03;
branches:  1.1.1;
Initial revision
----------------------------
revision 1.1.1.1
date: 2014-07-27 18:30:32 +0200;  author: root;  state: Exp;  lines: +0 -0; \
      commitid: 10053D529280B56DA03;
Initialer Import
=============================================================================
```

Listing 1.11 Log-Einträge anschauen mit CVS

Detailliertere Informationen finden sich im frei zugänglichen Buch »Open Source Development with CVS, 3rd Edition« von Karl Fogel und Moshe Bar unter *http://cvsbook.red-bean.com*.

1.2.2 Apache Subversion

Die Entwicklung von SVN, *Apache Subversion* (*https://subversion.apache.org*), begann im Jahre 2000 bei CollabNet, und im Februar 2004 erschien die erste stabile Version. Apache Subversion ist damit wesentlich jünger als CVS.

```
$ svnadmin create /var/tmp/svnrepository
$ cd $HOME
$ svn checkout file:///var/tmp/svnrepository
Checked out revision 0.
```

Listing 1.12 Ein Repository mit Apache Subversion erzeugen

Listing 1.12 zeigt, dass Apache Subversion es uns Benutzern etwas leichter macht: Mit `svnadmin create` wird ein Repository angelegt, und mit `svn checkout` wird es ausgecheckt.

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 14 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

Wir bearbeiten in Listing 1.13 die Datei *liesmich* und schauen mit `svn status` nach, ob es im aktuellen Verzeichnis Änderungen gibt, die sich noch nicht im Repository befinden. Diese Änderungen fügen wir mittels `svn add` der Versionskontrolle hinzu und übertragen sie per `svn commit` an das Repository (siehe Listing 1.13):

```
$ svn status
?       liesmich
$ svn add liesmich
A         liesmich
$ svn commit -m "Lies das" liesmich
Adding         liesmich
Transmitting file data .
Committed revision 1.
```

Listing 1.13 Mit Apache Subversion Dateien unter Versionskontrolle stellen

In Listing 1.14 sehen Sie, wie man mit Apache-Subversion-Befehlen herausfinden kann, was sich an einer Datei verändert hat:

```
$ svn diff liesmich
Index: liesmich
===================================================================
--- liesmich    (revision 1)
+++ liesmich    (working copy)
@@ -1 +1,3 @@
 Lies das!
+
+Mehr habe ich nicht zu sagen.

$ svn commit -m "mehr nicht"
Sending        liesmich
Transmitting file data .
Committed revision 2.
```

Listing 1.14 Ändern einer Datei mit Apache Subversion

Die Datei *liesmich* wird in Listing 1.14 weiterbearbeitet, und mit `svn diff` finden wir den Unterschied zur Datei im Repository heraus. Diese Änderungen werden wie gewohnt an das Repository übertragen.

Mit dem `svn log`-Kommando können wir uns in Listing 1.15 alle Änderungen der Datei anzeigen lassen:

```
$ svn log liesmich
------------------------------------------------------------------------
r2 | root | 2014-07-27 19:21:14 +0200 (Sun, 27 Jul 2014) | 1 line
```


Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 15 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

```
mehr nicht
------------------------------------------------------------------------
r1 | root | 2014-07-27 19:20:30 +0200 (Sun, 27 Jul 2014) | 1 line

Lies das
------------------------------------------------------------------------
```

Listing 1.15 Anzeigen aller Änderungen bei Apache Subversion

Weitere Informationen können Sie im deutschsprachigen und frei verfügbaren Buch »Versions-Kontrolle mit Subversion« von Ben Collins-Sussman, Brian W. Fitzpatrick und C. Michael Pilato unter *http://svnbook.red-bean.com* nachlesen.

1.2.3 GNU Bazaar

GNU Bazaar (kurz *BZR*, siehe *https://bazaar.canonical.com*) ist im März 2005 als Nachfolger des Systems *baz* entstanden und wurde maßgeblich vom britischen Unternehmen Canonical Ltd. – der Firma hinter Ubuntu Linux – unterstützt. Nachdem Canonical Anfang 2012 fast alle Entwickler abgezogen hat, ist die Entwicklung nahezu eingeschlafen.

Alle drei vorgestellten dezentralen Versionskontrollsysteme haben gemeinsam, dass sie anmahnen, die Userinformationen anzugeben, wenn man in ein Repository schreibt, und so beginnt die Konfiguration mit der einmaligen Angabe der Userinformationen:

```
$ bzr whoami "Dirk Deimeke <dirk@deimeke.net>"
$ cat ~/.bazaar/bazaar.conf
[DEFAULT]
email = Dirk Deimeke <dirk@deimeke.net>
```

Listing 1.16 Eingeben der Userinformationen bei GNU Bazaar

GNU Bazaar erlaubt das Anlegen eines Repositorys mit verschiedenen Teilbäumen. Im Beispiel aus Listing 1.17 wird das *bzr*-Repository angelegt und darin der Zweig *trunk* erstellt, der für die weiteren Schritte benutzt wird:

```
$ bzr init-repository /var/tmp/bzrrepository
Shared repository with trees (format: 2a)
Location:
  shared repository: /var/tmp/bzrrepository
$ bzr init /var/tmp/bzrrepository/trunk
Created a repository tree (format: 2a)
Using shared repository: /var/tmp/bzrrepository/
$ bzr branch /var/tmp/bzrrepository/trunk
Branched 0 revisions.
```

Listing 1.17 Erzeugen eines Repositorys mit GNU Bazaar

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 16 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

In Listing 1.18 wird eine *liesmich*-Datei angelegt. Das `status`-Kommando bezeichnet diese Datei als nicht unter Versionskontrolle stehend. Mit `add` wird sie als zugehörig markiert und mit `commit` lokal übertragen. Der `push` überträgt schließlich alle lokalen Änderungen an das »entfernte« Repository (siehe Listing 1.18):

```
$ vim liesmich
$ bzr status
unknown:
  liesmich
$ bzr add liesmich
adding liesmich
$ bzr commit -m "Lies das" liesmich
Committing to: /home/dirk/trunk/
added liesmich
Committed revision 1.
$ bzr push :parent
All changes applied successfully.
Pushed up to revision 1.
```

Listing 1.18 Mit GNU Bazaar Dateien unter Versionskontrolle stellen

In Listing 1.19 werden die Befehlsausgaben gezeigt, die erwartet werden können, wenn sich Dateien ändern. Neu ist hier der `diff`-Befehl, der die Unterschiede zwischen der lokalen Datei und der aktuellen Datei im Repository aufzeigt.

```
$ bzr status
modified:
  liesmich
$ bzr diff liesmich
=== modified file 'liesmich'
--- liesmich    2014-07-30 17:05:39 +0000
+++ liesmich    2014-07-30 17:06:05 +0000
@@ -1,1 +1,3 @@
 Lies das!
+
+Mehr habe ich nicht zu sagen.
$ bzr commit -m "mehr nicht" liesmich
Committing to: /home/dirk/trunk/
modified liesmich
Committed revision 2.
$ bzr push :parent
All changes applied successfully.
Pushed up to revision 2.
```

Listing 1.19 Ändern einer Datei mit GNU Bazaar

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 17 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

Zum guten Schluss zeigt der log-Befehl alle Änderungen einer Datei (siehe Listing 1.20):

```
$ bzr log liesmich
------------------------------------------------------------
revno: 2
committer: Dirk Deimeke <dirk@deimeke.net>
branch nick: trunk
timestamp: Wed 2014-07-30 19:06:31 +0200
message:
  mehr nicht
------------------------------------------------------------
revno: 1
committer: Dirk Deimeke <dirk@deimeke.net>
branch nick: trunk
timestamp: Wed 2014-07-30 19:05:39 +0200
message:
  Lies das
```

Listing 1.20 Anzeige aller Änderungen bei GNU Bazaar

Versionskontrollsysteme, GNU Bazaar ist da keine Ausnahme, gehören mit zu den am besten dokumentierten Open-Source-Produkten. Daher ist der Userguide von GNU Bazaar eine echte Empfehlung: *http://doc.bazaar.canonical.com/bzr.dev/en/user-guide/index.html*

1.2.4 Mercurial

Mercurial (kurz *HG*, siehe *www.mercurial-scm.org*) startete ebenfalls im Jahr 2005 und wurde initiiert, um einen Nachfolger des Systems *BitKeeper* zu schaffen, mit dem damals der Linux-Kernel entwickelt wurde und das durch eine Lizenzänderung nicht mehr frei verwendbar war. Heute ist Mercurial in der Community rund um die Programmiersprache *Python* weit verbreitet.

Die Befehle bei Mercurial sehen nicht sehr unterschiedlich aus im Vergleich zu dem, was Sie bereits bei GNU Bazaar kennengelernt haben. Ein großer Unterschied ist, dass Mercurial weniger gesprächig ist und dass es deutlich weniger Ausgaben liefert. Mercurial nutzt das Kommando hg. Das leitet sich vom Kürzel für Quecksilber im Periodensystem der Elemente her: *Mercurial* ist das englische Wort für *Quecksilber* bzw. *quecksilberhaltig*.

Anders als bei GNU Bazaar und Git müssen bei Mercurial die Userinformationen in eine Datei geschrieben werden. Es gibt kein Kommando dafür. Wie das genau aussieht, sehen Sie in Listing 1.21:

```
$ vim .hgrc
$ cat ~/.hgrc
```


Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 18 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

```
[ui]
username = Dirk Deimeke <dirk@deimeke.net>
```

Listing 1.21 Eingeben der Userinformationen bei Mercurial

Das Erstellen eines Repositorys geht direkt, wie Listing 1.22 zeigt:

```
$ hg init /var/tmp/hgrepository
$ hg clone /var/tmp/hgrepository
destination directory: hgrepository
updating to branch default
0 files updated, 0 files merged, 0 files removed, 0 files unresolved
```

Listing 1.22 Erzeugen eines Repositorys bei Mercurial

Auch die Befehle, um eine Datei unter Versionskontrolle zu stellen, stimmen mit denen von GNU Bazaar überein (siehe Listing 1.23):

```
$ hg status
? liesmich
$ hg add liesmich
$ hg commit -m "Lies das" liesmich
$ hg push
pushing to /var/tmp/hgrepository
searching for changes
adding changesets
adding manifests
adding file changes
added 1 changesets with 1 changes to 1 files
```

Listing 1.23 Dateien unter Versionskontrolle stellen mit Mercurial

Auch bei geänderten Dateien finden sich in Listing 1.24 keine Überraschungen; einzig die Ausgaben sind andere:

```
$ vim liesmich
$ hg status
M liesmich
$ hg diff liesmich
diff -r f34871bf678a liesmich
--- a/liesmich  Wed Jul 30 19:11:07 2014 +0200
+++ b/liesmich  Wed Jul 30 19:13:12 2014 +0200
@@ -1,1 +1,3 @@
 Lies das!
+
+Mehr habe ich nicht zu sagen.
$ hg commit -m "mehr nicht" liesmich
```


Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 19 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

```
$ hg push
pushing to /var/tmp/hgrepository
searching for changes
adding changesets
adding manifests
adding file changes
added 1 changesets with 1 changes to 1 files
```

Listing 1.24 Ändern einer Datei bei Mercurial

Auch bei der Ausgabe des `log`-Kommandos zeigt Mercurial leicht unterschiedliche Informationen (siehe Listing 1.25):

```
$ hg log
changeset:   1:b8dad3b41ff8
tag:         tip
user:        Dirk Deimeke <dirk@deimeke.net>
date:        Wed Jul 30 19:13:25 2014 +0200
summary:     mehr nicht

changeset:   0:f34871bf678a
user:        Dirk Deimeke <dirk@deimeke.net>
date:        Wed Jul 30 19:11:07 2014 +0200
summary:     Lies das
```

Listing 1.25 Anzeige aller Änderungen bei Mercurial

Mercurial hat ebenfalls einen sehr guten Userguide, den Sie unter der folgenden URL finden können: *https://www.mercurial-scm.org/guide*

1.2.5 Git

Git (*https://git-scm.com*) entstand ebenfalls aufgrund der Lizenzänderung von BitKeeper und wurde von Linus Torvalds ins Leben gerufen. Git ist derzeit das populärste der dezentralen (verteilten) Versionskontrollsysteme.

Die folgenden Listings zeigen, dass Git als »Plappermaul« gelten kann: Keines der vorgestellten Versionskontrollsysteme hat so ausführliche Ausgaben.

In Git lässt sich mit einem Kommando der eigene User einstellen (siehe Listing 1.26):

```
$ git config --global user.name "Dirk Deimeke"
$ git config --global user.email "dirk@deimeke.net"
$ cat ~/.gitconfig
[user]
```


Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 20 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

```
        name = Dirk Deimeke
        email = dirk@deimeke.net
```

Listing 1.26 Eingeben der Userinformationen bei Git

Auch in Git wird mit `init` ein neues Repository erstellt. Der Zusatz `--bare` sorgt in Listing 1.27 dafür, dass ein Repository ohne Arbeitskopie erstellt wird:

```
$ git init --bare /var/tmp/gitrepository
Initialized empty Git repository in /var/tmp/gitrepository/
$ git clone /var/tmp/gitrepository
Cloning into 'gitrepository'...
warning: You appear to have cloned an empty repository.
done.
```

Listing 1.27 Erzeugen eines Repositorys mit Git

Git zeigt keine Besonderheiten, was das Hinzufügen von Dateien in Listing 1.28 angeht:

```
$ cd gitrepository
$ vim liesmich
$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       liesmich
nothing added to commit but untracked files present (use "git add" to track)

$ git add liesmich
$ git commit -m "Lies das" liesmich
[master (root-commit) 8497685] Lies das
 1 file changed, 1 insertion(+)
 create mode 100644 liesmich
$ git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 218 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
To /var/tmp/gitrepository
 * [new branch]      master -> master
```

Listing 1.28 Dateien unter Versionskontrolle stellen mit Git

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 21 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

Der diff-Befehl wurde auch bereits vorgestellt. Seine Ausgaben unter Git sind nahezu identisch mit denen bei GNU Bazaar und Mercurial (siehe Listing 1.29):

```
$ git status
[…]
#       modified:   liesmich
#
no changes added to commit (use "git add" and/or "git commit -a")
$ git diff liesmich
diff --git a/liesmich b/liesmich
index 0abf191..21bb927 100644
--- a/liesmich
+++ b/liesmich
@@ -1 +1,3 @@
 Lies das!
+
+Mehr habe ich nicht zu sagen.
$ git commit -m "mehr nicht" liesmich
[master f325a3a] mehr nicht
 1 file changed, 2 insertions(+)
$ git push origin master
Counting objects: 5, done.
Writing objects: 100% (3/3), 275 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
To /var/tmp/gitrepository
   8497685..f325a3a  master -> master
```

Listing 1.29 Ändern einer Datei bei Git

In Listing 1.30 zeigt sich Git sehr aufgeräumt bei der Ausgabe der Log-Informationen:

```
$ git log liesmich
commit f325a3abc2d2902e8d172140005127a4f703a2f1
Author: Dirk Deimeke <dirk@deimeke.net>
Date:   Wed Jul 30 19:18:38 2014 +0200

    mehr nicht
commit 8497685b84e0852c504578d9cf95b19b5b9fc6b9
Author: Dirk Deimeke <dirk@deimeke.net>
Date:   Wed Jul 30 19:17:40 2014 +0200

    Lies das
```

Listing 1.30 Anzeige aller Änderungen bei Git

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 22 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

Auch für Git gibt es weiterführende Lektüre. Hier empfehlen sich das freie Pro-Git-Buch unter *https://git-scm.com/book* (englisch) bzw. das freie Git-Buch unter *https://gitbu.ch/* (deutsch).

1.3 Kommandos

In Tabelle 1.3 werden die Kommandos der einzelnen Versionskontrollsysteme einander gegenübergestellt. In den Spalten werden die Kommandos der Systeme benutzt: So steht `cvs` für Concurrent Versioning System, `svn` für Apache Subversion, `bzr` für GNU Bazaar, `hg` für Mercurial und `git` für Git.

	cvs	**svn**	**bzr**	**hg**	**git**
Paketname der Installation	cvs	subversion	bzr	mercurial	install git
Repository erstellen	init	svnadmin create	init init-repository	init	init init bare
Repository klonen	–	svnadmin hotcopy	branch	clone	clone
Arbeitskopie auschecken	checkout	checkout	checkout	clone	checkout
Hinzufügen	add	add	add	add	add
Löschen	rm	rm	rm	rm	rm
Verschieben (umbenennen)	–	mv	mv	mv	mv
Änderungen festschreiben	commit	commit	commit	commit	commit
Entfernte Änderungen übertragen	–	–	pull	pull	fetch
Lokale Änderungen übertragen	–	–	push	push	push
Arbeitskopie synchronisieren	update	update	update	pull -u	pull

Tabelle 1.3 Kommandos bei unterschiedlichen Versionskontrollsystemen

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 23 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

1.4 Serverdienste

Für das am weitesten verbreitete Versionskontrollsystem Git werden wir im Folgenden zwei Varianten betrachten, die es ermöglichen, einen eigenen Git-Server zu betreiben.

1.4.1 Git-Server mit Gitolite

Gitolite (*https://gitolite.com/*) bietet eine einfache Möglichkeit, eigene Git-Repositorys zu hosten. Das Einzige, was dazu benötigt wird, ist ein Server, auf dem SSH läuft, und das sollten alle Server bieten.

Gitolite installieren

Zu Beginn legen wir einen Systemuser und eine Systemgruppe namens »git« an. Die Befehle in Listing 1.31 dienen dazu als Referenz:

```
$ groupadd --system git
$ useradd --comment "Gitolite Service" --home-dir /srv/git --gid git --create-home \
   --system git
```

Listing 1.31 User und Gruppe anlegen

Sollte die Software `git` bis jetzt noch nicht installiert sein, dann ist jetzt ein guter Zeitpunkt, das nachzuholen. Ein Server mit Gitolite wird selbst auch mittels Git verwaltet. Dazu wird der SSH-Public-Key eines Users benötigt, der später die Repositorys administrieren soll. In Listing 1.32 beispielsweise liegt der öffentliche SSH-Schlüssel des Benutzers »dirk« auf der gleichen Maschine, das muss aber nicht zwangsweise so sein.

```
$ cp /home/dirk/.ssh/id_rsa.pub /srv/git/dirk.pub
$ chown git:git /srv/git/dirk.pub
```

Listing 1.32 Öffentlichen Teil des Admin-Keys kopieren

Die Installation führen wir mit dem User »git« aus. Dazu klonen wir zuerst das Repository von GitHub (siehe Listing 1.33):

```
$ git clone https://github.com/sitaramc/gitolite.git
Cloning into 'gitolite'...
[…]
Checking connectivity... done.
```

Listing 1.33 Klonen des Repositorys

Nach dem Klonen besteht die Installation aus den drei in Listing 1.34 beschriebenen Schritten, die als User »git« ausgeführt werden müssen. Anschließend ist die Installation für den User funktionsfähig, dessen Key im vorherigen Schritt (siehe Listing 1.32) kopiert wurde.

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 24 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

```
$ mkdir $HOME/bin
$ gitolite/install -to $HOME/bin
$ $HOME/bin/gitolite setup -pk dirk.pub
Initialized empty Git repository in /srv/git/repositories/gitolite-admin.git/
Initialized empty Git repository in /srv/git/repositories/testing.git/
WARNING: /srv/git/.ssh missing; creating a new one
    (this is normal on a brand new install)
WARNING: /srv/git/.ssh/authorized_keys missing; creating a new one
    (this is normal on a brand new install)
```

Listing 1.34 Installation von Gitolite

Damit ist serverseitig alles getan. Die weiteren Schritte kann und muss der angelegte Admin-User per SSH und geklontem Admin-Repository durchführen.

Gitolite konfigurieren

Im Weiteren zeigen wir Ihnen, wie sich Gitolite verwalten lässt. Wir verwenden dazu einen Client, der sich auf der gleichen Maschine wie die Serverinstallation befindet. Wenn Sie das in Ihrer Umgebung ausführen wollen, verwenden Sie bitte statt *localhost* die Adresse oder den Namen Ihres Servers.

Der erste Schritt zur Verwaltung des eigenen Servers ist das Klonen des Admin-Repositorys (siehe Listing 1.35). Im Verzeichnis finden sich – bis auf die Git-Verwaltungsinformationen – nur zwei Dateien: der öffentliche Teil des Administratorschlüssels (hier im Beispiel *dirk.pub*) und eine Konfigurationsdatei mit dem Namen *gitolite.conf*.

```
$ git clone ssh://git@localhost/gitolite-admin.git
Cloning into 'gitolite-admin'...
remote: Counting objects: 6, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (6/6), done.

$ cd gitolite-admin
$ ls *
conf:
gitolite.conf

keydir:
dirk.pub

$ cat conf/gitolite.conf
repo gitolite-admin
    RW+     =   dirk
```


Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 25 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

```
repo testing
    RW+     =   @all
```

Listing 1.35 Klonen des Admin-Repositorys

Wie Sie leicht feststellen können, sind in der Konfigurationsdatei zwei Repositorys eingestellt: zum einen das, in dem wir uns gerade befinden, und zum anderen ein Repository namens *testing*. Weiterhin sehen Sie, dass der User `dirk` Lese-, Schreib- und Sonderrechte (»fast-forward«, »rewind« und Löschrechte auf Branches und Tags) hat und niemand anderes sonst. Die Gruppe `all` hat die gleichen Rechte auf das Repository `testing`.

Ein User in Gitolite entspricht einem öffentlichen Schlüssel, der so, wie in Listing 1.36 gezeigt, dem Repository hinzugefügt wird. Wichtig ist, dass die Namen der öffentlichen Schlüssel immer auf *.pub* enden.

```
$ cp /tmp/paul.pub keydir
$ git add keydir/paul.pub
$ git commit -m "User Paul added"
[master a741b6b] User Paul added
 1 file changed, 1 insertion(+)
 create mode 100644 keydir/paul.pub
$ git push
Counting objects: 6, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 676 bytes | 0 bytes/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To ssh://git@localhost/gitolite-admin.git
   5e8f3ee..a741b6b  master -> master
```

Listing 1.36 Hinzufügen eines neuen Users

Das Anlegen einer Gruppe erfolgt ähnlich einfach. Gruppen beginnen bei Gitolite mit einem »@«-Zeichen, in Listing 1.37 legen wir die Gruppe `adminbuch` mit den Mitgliedern `dirk` und `paul` an und zusätzlich je eine Gruppe, die nur einen User enthält.

```
$ head -3 conf/gitolite.conf
@dirk      = dirk
@paul      = paul
@adminbuch = paul @dirk
```

Listing 1.37 Anlegen einer Gruppe

Wie Sie sehen, können Gruppen sowohl andere Gruppen wie auch »Personen« enthalten. Mitglieder werden durch Leerzeichen getrennt. Wir empfehlen Ihnen, aus Gründen der Übersichtlichkeit möglichst immer Gruppen einzusetzen. Nach einem `git commit` und `git push`

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 26 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

ist diese neue Konfiguration aktiv. Ein neues Repository können Sie nach dem Muster der bestehenden Repositorys anlegen. Listing 1.38 zeigt Ihnen, wie das geht:

```
$ tail -2 conf/gitolite.conf
repo adminbuch
    RW+     =   @adminbuch
$ git commit -am "New repository"
[master dc58e17] New repository
 1 file changed, 6 insertions(+), 3 deletions(-)
$ git push
Counting objects: 7, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 406 bytes | 0 bytes/s, done.
Total 4 (delta 0), reused 0 (delta 0)
remote: Initialized empty Git repository in /srv/git/repositories/adminbuch.git/
To ssh://git@localhost/gitolite-admin.git
   f04e394..dc58e17  master -> master
```

Listing 1.38 Hinzufügen eines neuen Repositorys

Sobald Sie die veränderte Datei zum Server übertragen haben, gibt es ein neues Repository namens `adminbuch`, für das die User `dirk` und `paul` als Mitglieder der Gruppe `adminbuch` volle Rechte haben.

Sollten Sie dem User `paul` erlauben wollen, lesend auf das Admin-Repository zuzugreifen, fügen Sie analog zu Listing 1.39 eine Zeile für die Leseberechtigungen ein:

```
$ grep -A 2 gitolite-admin conf/gitolite.conf
repo gitolite-admin
    RW+     =   dirk
    R       =   paul
```

Listing 1.39 Vergabe von Lese- und Schreibberechtigungen

Selbstverständlich können Sie anstelle der User auch die entsprechenden Gruppen verwenden. Gitolite bietet Ihnen noch viele weitere Möglichkeiten, Repositorys granular zu konfigurieren und beispielsweise serverseitige Hooks einzusetzen (und vieles andere mehr). Das zu beschreiben, würde aber den Rahmen dieses Kapitels bei Weitem sprengen, bitte konsultieren Sie dazu die Homepage des Projekts: *https://gitolite.com*

1.4.2 Git-Server mit Gitea

Gitea (*https://gitea.io/*) bietet eine einfache Möglichkeit, eigene Git-Repositorys zu hosten. Als Besonderheit bietet Gitea eine Weboberfläche, die der von GitHub (*https://github.com/*)

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 27 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

nachempfunden ist. Repositorys können nicht nur via SSH, sondern auch via HTTP oder HTTPS benutzt werden.

[+] Die Quelltexte des vor Ihnen liegenden Buches werden ebenfalls auf einem Server mittels Gitea verwaltet.

Gitea installieren

Analog zu unserem Vorgehen bei Gitolite legen wir zu Beginn einen Systemuser und eine Systemgruppe namens »gitea« an:

```
$ groupadd --system gitea
$ useradd --comment "Gitea Service" --home-dir /srv/gitea --gid gitea --create-home \
   --system gitea
```

Listing 1.40 User und Gruppe anlegen

[+] Bitte beachten Sie, dass Sie den User natürlich beliebig benennen können. Sollten Sie Gitea nicht verwenden, spricht gar nichts dagegen, den User »git« zu nennen.

Selbstverständlich benötigen Sie für einen Server, der Git-Repositorys verwaltet, auch die Software `git`. Bitte installieren Sie Git mit den Mitteln Ihres Betriebssystems. Zur Installation von Gitea laden Sie sich die entsprechende Binärdatei von der Download-Seite des Projekts *https://dl.gitea.io/gitea/* herunter. In Listing 1.41 finden Sie die Befehle, die Sie benötigen, um die zum Druckzeitpunkt aktuelle Version von Gitea auf Ihrem Server zu installieren:

```
$ sudo -u gitea -i
$ mkdir gitea
$ cd gitea
$ curl -LO https://dl.gitea.io/gitea/1.13.1/gitea-1.13.1-linux-amd64
[…]
$ chmod 750 gitea-1.13.1-linux-amd64
$ ln -s gitea-1.13.1-linux-amd64 gitea
```

Listing 1.41 Installation von Gitea

Der letzte Befehl, das Setzen des Links, wäre nicht unbedingt nötig, erleichert aber bei künftigen Versionen die Aktualisierung. Starten Sie Gitea einmalig, um fehlende Dateien und Verzeichnisse zu erstellen:

```
$ ./gitea web
2018/09/22 13:11:09 [W] Custom config '/srv/gitea/gitea/custom/conf/app.ini' \
                        not found, ignore this if you're running first time
2018/09/22 13:11:09 [T] AppPath: /srv/gitea/gitea/gitea
2018/09/22 13:11:09 [T] AppWorkPath: /srv/gitea/gitea
2018/09/22 13:11:09 [T] Custom path: /srv/gitea/gitea/custom
2018/09/22 13:11:09 [T] Log path: /srv/gitea/gitea/log
```


Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 28 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

```
2018/09/22 13:11:09 [I] Log Mode: Console(Info)
2018/09/22 13:11:09 [I] XORM Log Mode: Console(Info)
2018/09/22 13:11:09 [I] Cache Service Enabled
2018/09/22 13:11:09 [I] Session Service Enabled
2018/09/22 13:11:09 [I] SQLite3 Supported
2018/09/22 13:11:09 [I] Run Mode: Development
2018/09/22 13:11:09 [I] Gitea v1.5.1 built with: bindata, sqlite
2018/09/22 13:11:09 [I] Listen: http://0.0.0.0:3000
2018/09/22 13:11:09 Serving [::]:3000 with pid 17937
```

Listing 1.42 Erster Start von Gitea

Rufen Sie nun die URL *http://localhost:3000/install* einmal auf, um die Grundkonfiguration zu erstellen. Als Datenbank für Tests können Sie SQLite verwenden. Falls Sie ein größeres Setup erstellen wollen, empfehlen wir eine Verbindung zu einem »echten Datenbanksystem«. Vergessen Sie dabei nicht, einen Admin-Usernamen und ein Passwort zu vergeben.

Beenden können Sie den Server mit der Tastenkombination Strg + C.

Gitea konfigurieren

Ein rudimentäres Beispiel für eine *systemd-service*-Datei finden Sie in Listing 1.43. Erstellen Sie eine Datei */etc/systemd/system/gitea.service* mit dem folgenden Inhalt:

```
[Unit]
Description=Gitea (Git with a cup of tea)
After=syslog.target
After=network.target

[Service]
Type=simple
User=gitea
Group=gitea
WorkingDirectory=/srv/gitea/gitea
ExecStart=/srv/gitea/gitea/gitea web
Restart=always
Environment=USER=gitea HOME=/srv/gitea

[Install]
WantedBy=multi-user.target
```

Listing 1.43 »systemd-service«-Datei für Gitea

Um diesen Dienst zu aktivieren, sollten Sie einmal als root den Befehl `systemd daemon-reload` ausführen. Anschließend können Sie mit `systemctl start gitea` den Dienst starten.

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 29 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle/versionskontrolle" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

Weitere Konfigurationsmöglichkeiten mit Bezug zu *systemd* sind auf der offiziellen Webseite unter *https://docs.gitea.io/en-us/linux-service/* beschrieben.

Da der Zugriff auf die Gitea-Instanz unverschlüsselt via HTTP auf Port 3000 erfolgt, wäre es sinnvoll, einen Webserver oder Squid als Reverse-Proxy vor die Gitea-Instanz zu stellen. Der Webserver würde in diesem Fall die eingehenden Anfragen auf Port 443 (HTTPS) an Gitea weiterleiten. Das können Sie dadurch erreichen, dass Sie beispielsweise die folgenden Zeilen aus Listing 1.44 der Konfigurationsdatei Ihres *VirtualHosts* hinzufügen:

```
ProxyPreserveHost On
ProxyPass / http://localhost:3000/
ProxyPassReverse / http://localhost:3000/
```

Listing 1.44 Reverse-Proxy mit Apache

Passen Sie in der Datei */srv/gitea/gitea/custom/conf/app.ini* die Einträge im Abschnitt `[server]` entsprechend an. Im Beispiel aus Listing 1.45 nennen wir den VirtualHost *gitea.example.com*:

```
[server]
SSH_DOMAIN       = gitea.example.com
HTTP_PORT        = 3000
ROOT_URL         = https://gitea.example.com/
DISABLE_SSH      = false
SSH_PORT         = 22
LFS_START_SERVER = false
OFFLINE_MODE     = false
```

Listing 1.45 Gitea-Server-Konfiguration

Alle Möglichkeiten, die Gitea Ihnen bietet, können wir leider in diesem Kapitel nicht beschreiben. Daher möchten wir Ihnen die sehr umfangreiche Dokumentation unter *https://docs.gitea.io/en-us/* ans Herz legen.

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 30 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle.ind" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x

Index

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:02 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 31 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "versionskontrolle.ind" , Aktueller Job: versionskontrolle, Aktuelles Encoding: utf8x