

Schnelleinstieg in CVS

[Heiko Beiküfner](#)

Inhaltsverzeichnis

- [Einrichten des Repository](#)
- [cvs checkout](#)
- [cvs commit](#)
- [Hinzufügen von Dateien](#)
- [Entfernen von Dateien](#)
- [Entfernte Datei zurückholen](#)
- [Datei umbenennen](#)
- [Unterschiede zwischen Versionen](#)
- [Branches](#)
- [Die History](#)
- [Exportieren](#)
- [update](#)
- [Ignorieren von Dateien](#)
- [Anhang](#)
 - [a. Kommandoübersicht](#)
 - [b. Anmerkungen](#)
 - [c. Software](#)
 - [d. Links zum Thema CVS](#)

Dieser Schnelleinstieg bietet eine kurze Referenz der wichtigsten und nützlichsten CVS-Befehle, kann aber keine ausführliche Auseinandersetzung mit den theoretischen Hintergründen der einzelnen Aktionen ersetzen. Hierzu bietet sich der Artikel [&content/server/cvs.html">CVS for Dummies](#) an.

Einrichten des Repository

Am Beispiel des CVS-Modules zur Verwaltung dieser Anleitung wird die Einrichtung des CVS-Repository vorgestellt. Eingaben sind mit '>' gekennzeichnet. Unter dem Begriff "Repository" stellt man sich im Zusammenhang mit CVS **das** zentrale Verzeichnis vor, unter dem alle Dateien und Unterverzeichnisse zusammengefaßt werden, die nachher von CVS verwaltet werden sollen. "Version" meint in diesem Kontext die Version eines Verzeichnisses, die ein Tag <Versionsname> trägt. Ein Repository kann als root angelegt werden, mit:

```
> mkdir -p /var/log/CVS/CVSR00T
> mkdir -p /var/log/CVS/Doku_CVS
```

In /var/log/CVS/ speichert CVS seine Daten. CVS nennt dies sein Repository. In /var/log/CVS/Doku_CVS speichert CVS die Daten zum "Projekt" Doku_CVS. CVS nennt dies ein Modul.

```
> groupadd cvs
> useradd -g users -G cvs -m doc_cvs
> cd /var/log/CVS
```

```
> chown doc_cvs.cvs Doku_CVS
```

Nach `groupadd cvs` und `useradd` sind die Schreibrechte auf das Modul festgelegt. Das Modul `Doku_CVS` und alle Rechte daran gehören dem neuangelegten Benutzer `doc_cvs`.

```
> cd
> mkdir Doku_CVS
> cd Doku_CVS
```

Alle dazugehörigen Dateien in dieses Verzeichnis kopieren oder darin erzeugen.

```
> cvs import -m "initial Import" Doku_CVS Original alpha
```

Beim Import erwartet CVS diese Syntax:

```
cvs import -m <Kommentar> <Modul-Name> <Vendorname> <Revision-tag>
```

Vendorname und Revision-Tag müssen angegeben werden, der Inhalt ist an sich beliebig. CVS nimmt nun die Daten in seine Verwaltung auf.

Wichtig!!! Nie die Dateien im Repository direkt ändern !!!

Wichtig!!! Verzeichnisse sind später nicht zu entfernen !!!

Damit CVS eine History der Änderungen anlegt, muß

```
> cvs -d /var/CVS init
```

aufgerufen werden. `cvs init` ist "vorsichtig". D.h. es löscht nie Dateien im Repository, könnte also gefahrlos auch später aufgerufen werden.

```
> cvs tag -F Original Doku_CVS
```

Die Daten im jetzigen Zustand bekommen den Tag "Original".

```
> cd ..
> rm -r Doku_CVS
```

Verzeichnis löschen, um CVS-Mechanismus zu prüfen.(d.h. lokal löschen und nicht im Repository, beim checkout kann man die Daten wieder auslesen.)

cvs checkout

```
> cvs checkout -r Original Doku_CVS
```

CVS erzeugt im aktuellen Arbeitsverzeichnis das Unterverzeichnis `Doku_CVS` neu und legt dort Daten ab. Überprüfung darauf, ob alles da ist:

```
> cvs checkout -P Doku_CVS
```

holt die aktuelle Version, jedoch ohne leere Verzeichnisse.

```
> cd Doku_CVS
> vi cvsDoc ... <esc> :wq!
```

Änderungen vornehmen und speichern.

Am Rande bemerkt:

Hat man die Variable \$CVSROOT nicht gesetzt oder will sie nicht ändern, weil sie auf ein weiteres, benutztes Repository verweist, kann mit

```
> cvs -d 10.0.0.216:/var/CVS checkout alice
```

der Pfad nach -d übergeben werden. Die Pfadangabe muß absolut sein.

cvs commit

```
> cvs commit Doku_CVS
```

Nun werden die geänderten Daten ins Repository übernommen.

```
> cvs tag -F alpha1 Doku_CVS
```

versieht die aktuelle Version mit einem neuen Versions-Tag.

Hinzufügen von Dateien

Hinzufügen einer Datei: (in diesem Fall der Datei Kommandos)

```
> cd Doku_CVS
> vi Kommandos ... <esc> :wq!
```

```
> cvs add Kommandos
```

Datei wird fürs Hinzufügen NUR vorgemerkt!

```
> cvs commit
```

Nun ist die Datei Kommandos ins Repository aufgenommen.

```
> cd ..
> cvs tag -F "foo_exists" Doku_CVS
```

Der neue Versions-Tag ist gesetzt. Verzeichnisse werden analog hinzugefügt. Jedoch geschieht dies nicht rekursiv! Bitte [Anmerkung Verzeichnisse](#) beachten!

Entfernen von Dateien

Im CVS-Kontext bedeutet das Entfernen einer Datei, daß die alte Version im Repository erhalten bleibt, aber beim checkout nicht mehr ausgeliefert wird. Siehe auch Anmerkung über [Verzeichnisse!](#)

```
> cd Doku_CVS
> rm foo
```

entfernt die Datei foo. Mit

```
> cvs remove foo
```

werden die letzten beiden Schritte zusammengefasst.

Nach dem "Einchecken" der Änderungen mit:

```
> cvs commit foo
```

Die Datei wird nun nicht mehr bei `cvs checkout` ausgegeben.

```
> cd ..  
> cvs tag -F "foo_deleted" Doku_CVS
```

Der aktuelle Versions-Tag ist gesetzt.

Entfernte Datei zurückholen

Um die Datei `foo` zu erhalten:

```
> cvs checkout -r foo_exists Doku_CVS
```

Dies hat aber einen "sticky" Tag zur Folge. "sticky" meint in diesem Fall, daß die durch `-r <Version>` mitgegebene Version bis auf weiteres diejenige bleibt, auf die sich die folgenden Updates beziehen.

```
> cvs update -A
```

holt die neueste Version und entfernt dabei "sticky" Tags

```
> cvs update -p -r foo_exists Doku_CVS
```

holt die ältere Version, in der die Datei noch existiert, gibt diese aber einfach in die Konsole aus. U.U. mit

```
> cvs update -p -r foo_exists Doku_CVS >Old_Doku_CVS
```

in eine Datei holen.

Datei umbenennen

Um eine Datei umzubenennen, genügt ein einfaches `mv`-Kommando:

```
> mv foo bar # in bar umbenennen (lokal)  
> cvs remove foo # foo entfernen  
> cvs add bar # lege unter CVS die neue Datei bar an  
> cvs commit -m "Renamed foo to bar" foo bar # Übernehmen der Änderung
```

Dabei ändert sich die Versionsnummer normalerweise auf 1.1

Angleichen von Versionsnummern:

```
cvs commit -r 3.0
```

Setzt die Versionsnummer aller Dateien im Modul auf 3.0. Achtung: Die neue Versionsnummer muß höher sein als alle momentan verwendeten.

Unterschiede zwischen Versionen

Der `diff`-Befehl liefert die Unterschiede zwischen einzelnen Versionen:

```
cv s diff
```

Um direkt die Unterschiede zwischen zwei Versionsnummern anzeigen zu lassen, bekommt `diff` die Option `-r` für Revision mit auf den Weg.

```
> cvs diff -r 3.0 -r 3.12 Doku_CVS
```

Unterschiede zwischen Versionen vom 11.09.2000 und vom 19.09.2000:

```
> cvs diff -D 09/11/2000 -D 09/19/2000 Doku_CVS
```

(Datum im Format mm/dd/yyyy)

Alle Unterschiede seit dem letzten checkout anzeigen:

```
> cvs diff -u Doku_CVS
```

Führt man vor jedem commit

```
> cvs diff -u Doku_CVS >>Doku_CVS.changelog
```

aus, erhält man mit der Zeit eine `changelog` Datei ohne diese selbst zu bearbeiten.

Branches

Erzeugen eines Branches (Ast), der unabhängig vom Haupt-(Datei)-Baum bearbeitbar ist. Branches können eigenständig bleiben oder später zum Haupt-Baum hinzugefügt werden ("merge"):

```
> cvs tag -b test_branch
```

Erzeugt einen Branch-Tag im Repository, ausgehend von der Version im Arbeitsverzeichnis. Man arbeitet also nicht automatisch im neuen Branch!!!

VARIANTE:

```
> cvs rtag -b -r foo_exists
```

Erzeugt einen Branch-Tag im Repository, unabhängig von der Version im Arbeitsverzeichnis, basierend auf der nach `-r` angegebenen Version.

Um den neuen Branch zu bearbeiten:

```
>cvs checkout -r test_branch Doku_CVS
```

Achtung: Die Dateien landen im normalen Arbeitsverzeichnis (Doku_CVS)

VARIANTE:

```
> cvs checkout -d ./test_branch -r test_branch Doku_CVS
```

Erzeugt ein Verzeichnis `test_branch` und legt die Dateien in diesem ab. Dadurch kann man mehrere Branches parallel bearbeiten.

Um die Änderungen in das Repository zurückzuspielen ohne die Daten im Haupt-Tree zu verändern, muß die `commit`-Strategie geändert werden:

```
> cvs commit -r test_branch Doku_CVS
```

Die History

Auslesen der History:

```
> cvs history -e           # alles ausgeben
> cvs history -m <Doku_CVS> # gibt nur Daten von diesem Modul aus
> cvs history -c           # Info über "committed files"
> cvs history -o           # Info über "checked out files"
```

Exportieren

Um nur die "Nutzdaten" vom 19.09.2000 ohne die Verwaltungsdateien von CVS zu erhalten (in das lokale Verzeichnis ExportDoku):

```
> cvs export -d ExportDoku -D 09/19/2000 Doku_CVS
```

update

Um die eigene Arbeitsversion einer Datei mit den Änderungen anderer Bearbeiter zu synchronisieren, führt man `cvs update` aus.

```
> cvs update Doku_CVS
```

Ignorieren von Dateien

Befinden sich bei einem CVS Update Dateien im lokalen Arbeitsverzeichnis, die nicht von CVS verwaltet werden, ergibt sich eine Meldung wie:

```
? Doku_CVS//Comment
```

Um solche Meldung zu verhindern, kann man sich in seinem Homeverzeichnis die Datei `.cvsignore` anlegen. Pro Zeile ein Dateiname, * als Joker.

cvs ignoriert von selbst bereits alle Dateien mit:

```
.bak      .BAK      .o        .obj      CVS       RCS       core
```

Anhang

a. Kommandoübersicht

CVS Kommandos mit Syntax

```
cvs import [-m <Kommentar>] <Modulname> <Vendor-Tag> <Revision-Tag>
cvs checkout [-P] [-r <tag>] <Modulname>
cvs commit [-r <tag>] [<datei>] [<Modulname>]
cvs update [-p] [-r] [-A] [<tag>] [<Modulname>]
cvs tag [-b] [-F] <tag> <Modulname>
cvs rtag [-b] -r <tag> <Modulname>
```

```
cvs add <datei>
cvs remove [-f] [-l] [-R] <datei>
cvs -d <CVSROOT-path> init
cvs history [-m <Modulname>] [-e] [-o]
cvs export -d <Verzeichnisname> [-D <Datum>] [-r <tag>]<Modulname> cvs update
```

b. Anmerkungen

Verzeichnisse

Verzeichnisse, die beim Import mit angelegt oder auch später hinzugefügt wurden, lassen sich nicht einfach wie Dateien entfernen. Um sie "loszuwerden", hilft Folgendes:

```
> cvs remove -fR <Verz-Name>      # entfernt rekursiv!
> cvs commit                      # dem Repository mitteilen
> rm -r <lokales cvs-Verz>        # lokales Verzeichnis löschen
> cvs checkout -P <modulname>     # ohne leere Verzeichnisse holen
```

`cvs checkout -P <Modulname>` kann man sich also ruhig angewöhnen.

Im Repository bleibt das Verzeichnis erhalten, um frühere Versionen, die dieses Verzeichnis enthielten, weiter komplett vorhalten zu können.

IM NOTFALL: Als root in `/var/CVS/<modul>` das Verzeichnis löschen. CVS kommt damit - soweit bisher getestet - klar. Diese Methode passt allerdings kaum zu der Idee die hinter CVS steht. Auf diese Weise wird eines der hervorstechendsten Merkmale von CVS, nämlich die Rekonstruierbarkeit aller Versionen, torpediert.

c. Software

CVS

Das CVS-Paket selbst. Auf SuSE 7.0 enthalten, Serie d1 (development)

tkcvs

Recht brauchbares tk-CVS-Interface, das die meisten CVS Befehle unterstützt..

<http://www.twobarleycorns.net/tkcvs.html>

pharmacy

Gnome-CVS-Frontend. Auf SuSE 7.0 enthalten, Serie gnm (Gnome)

<http://pharmacy.sourceforge.net/index.html>

wincvs

Win98/NT-CVS-Frontend

<http://www.wincvs.org/>

d. Links zum Thema CVS

Die Website des CVS-Projekts:

<http://www.cvshome.org>

Sammlung von CVS-Tools:


<http://www.loria.fr/cgi-bin/molli/wilma.cgi/rel>

Ein CVS-Tutorial für Entwickler:

<http://cvsbook.red-bean.com/cvsbook.html>

CVS und Websites:

<http://www.durak.org/cvswebsites/howto-cvs/>

Heiko Beiküfner arbeitet als Systemadministrator bei der SuSE Linux GmbH. 

Linux auf dem Server 16.01.2001