Verteilte Systeme
Versionskontrolle mit Mercurial

Probleme der Gruppenarbeit

Mehrere Personen arbeiten an einer Aufgabe

Was passiert, wenn eine Datei von mehreren Leuten verändert wurde?

Welche ist die aktuelle Version einer Datei?

Wie tauschen die Gruppenmitglieder Änderungen an den Dateien aus?

Von wem stammt welche Änderung?

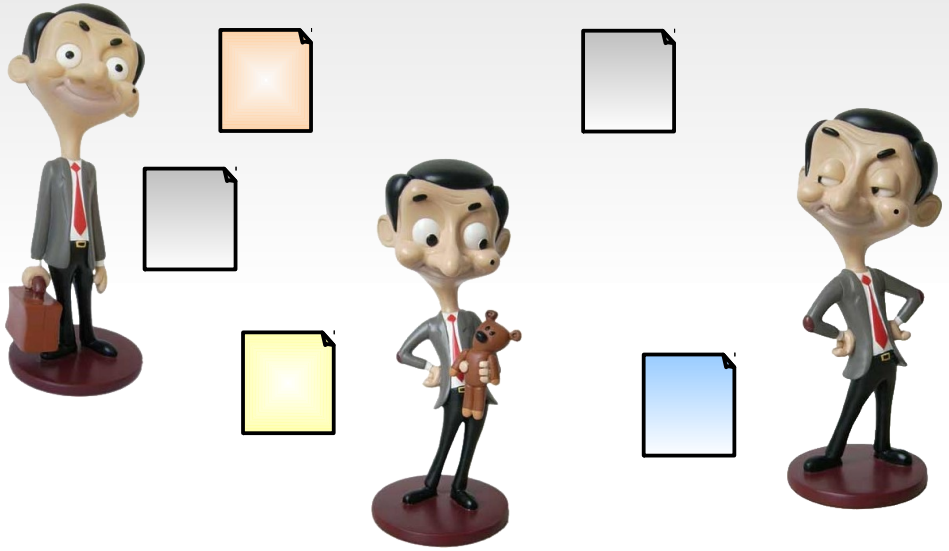
Austausch per E-Mail

1. Ein Mitglied der Gruppe fängt an, eine Aufgabe zu bearbeiten.

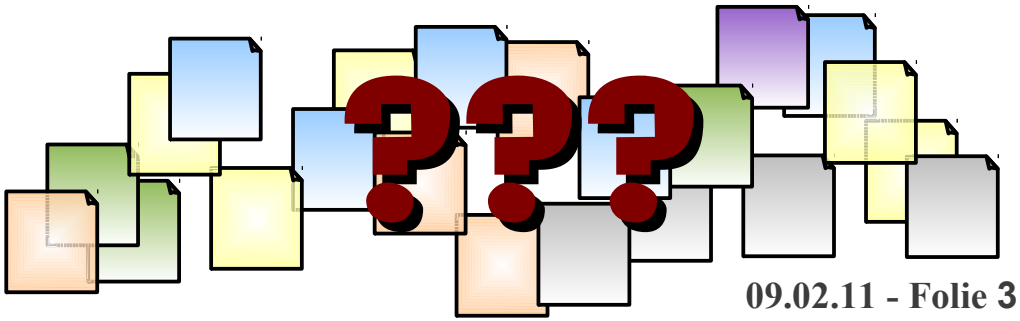
```
public class RemoteServer {
  private String host;
  private int port;
  private Socket socket;

  public RemoteServer
  (String host, int port) {
    this.host = host
    this.port = port;
  }
}
```

2. Die anderen Teilnehmer erhalten eine E-Mail mit der bisherigen Lösung.

3. Jeder löst seinen Teil der Aufgabe. Dabei wird für jede Änderung an einer Datei eine E-Mail an die gesamte Gruppe geschickt.



4. Am Ende weiß keiner mehr so recht, wer was gemacht hat und ob alle Teilnehmer mit denselben Dateien arbeiten ...

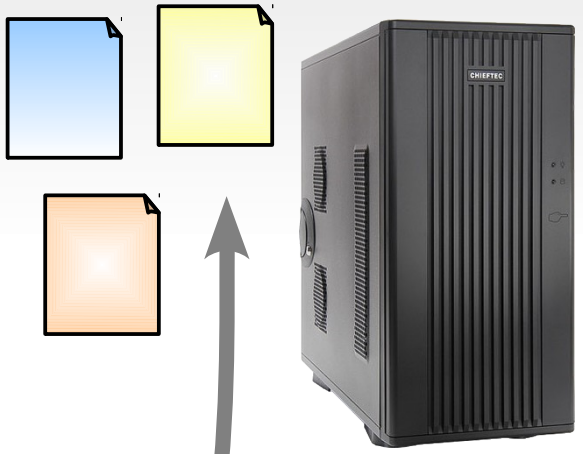


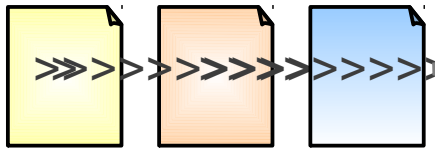
Zentraler Dateiserver

1. Ein Mitglied der Gruppe fängt an, eine Aufgabe zu bearbeiten

2. Hochladen der Dateien auf den zentralen Server

```
public class RemoteServer {
  private String host;
  private int port;
  private Socket socket;

  public RemoteServer
  (String host, int port) {
    this.host = host
    this.port = port;
  }

  public void connect() {
    this.socket = new socket(
      host, port);
  }
}
```

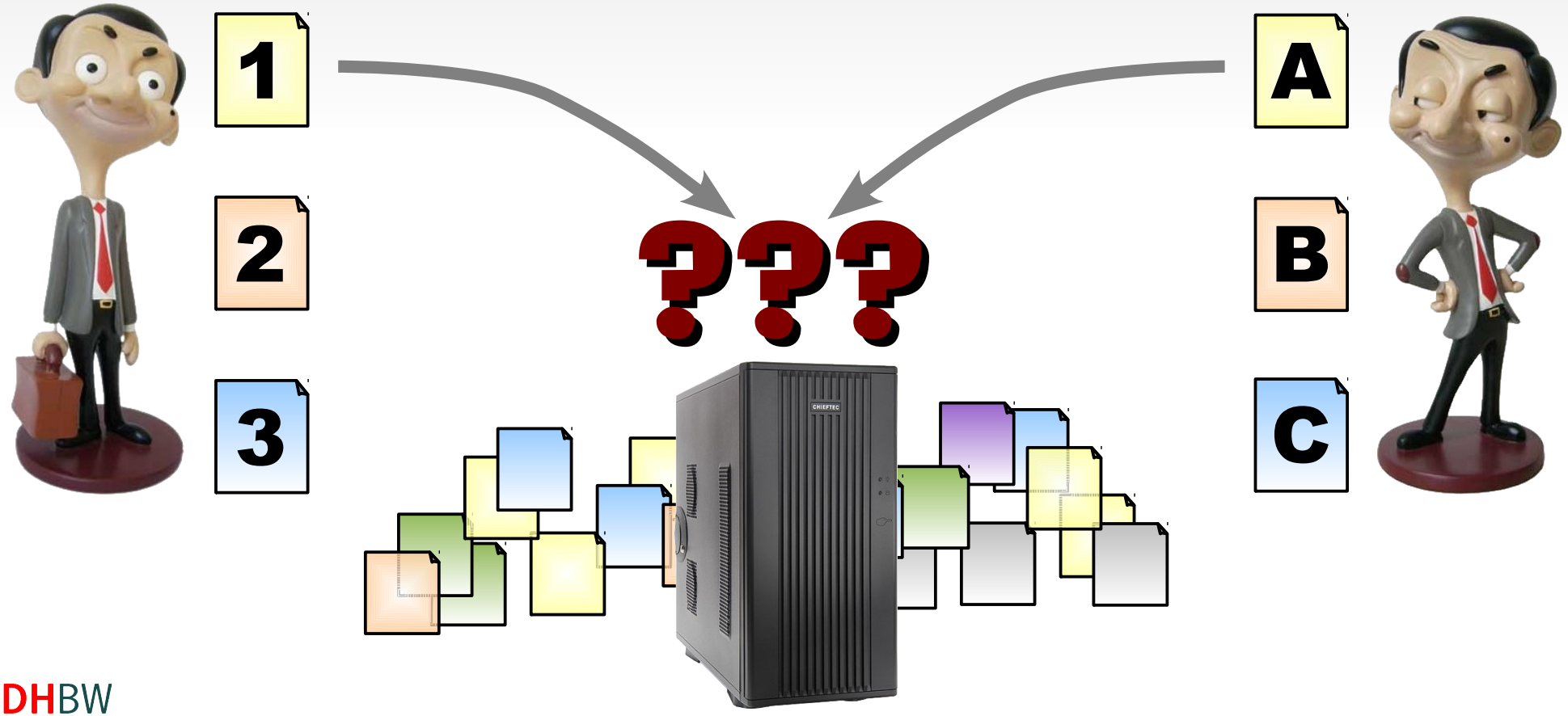




Zentraler Dateiserver

3. Die anderen Teilnehmer laden die Dateien runter und lösen je ihren Teil der Aufgabe.

4. Jeder will seine Lösung auf den Server hochladen. Doch wessen Version ist die richtige? Was ist mit den Änderungen der anderen Benutzer?



Lösung: Versionskontrolle

Zentrale Versionskontrollsysteme (CVS, Subversion):

- Alle Projektdateien liegen auf einem zentralen Server
- Änderungen an den Dateien werden an den Server übertragen
- Der Server verwaltet eine automatische Änderungshistorie
- Regelmäßiger Abgleich der lokalen Dateien mit dem Server

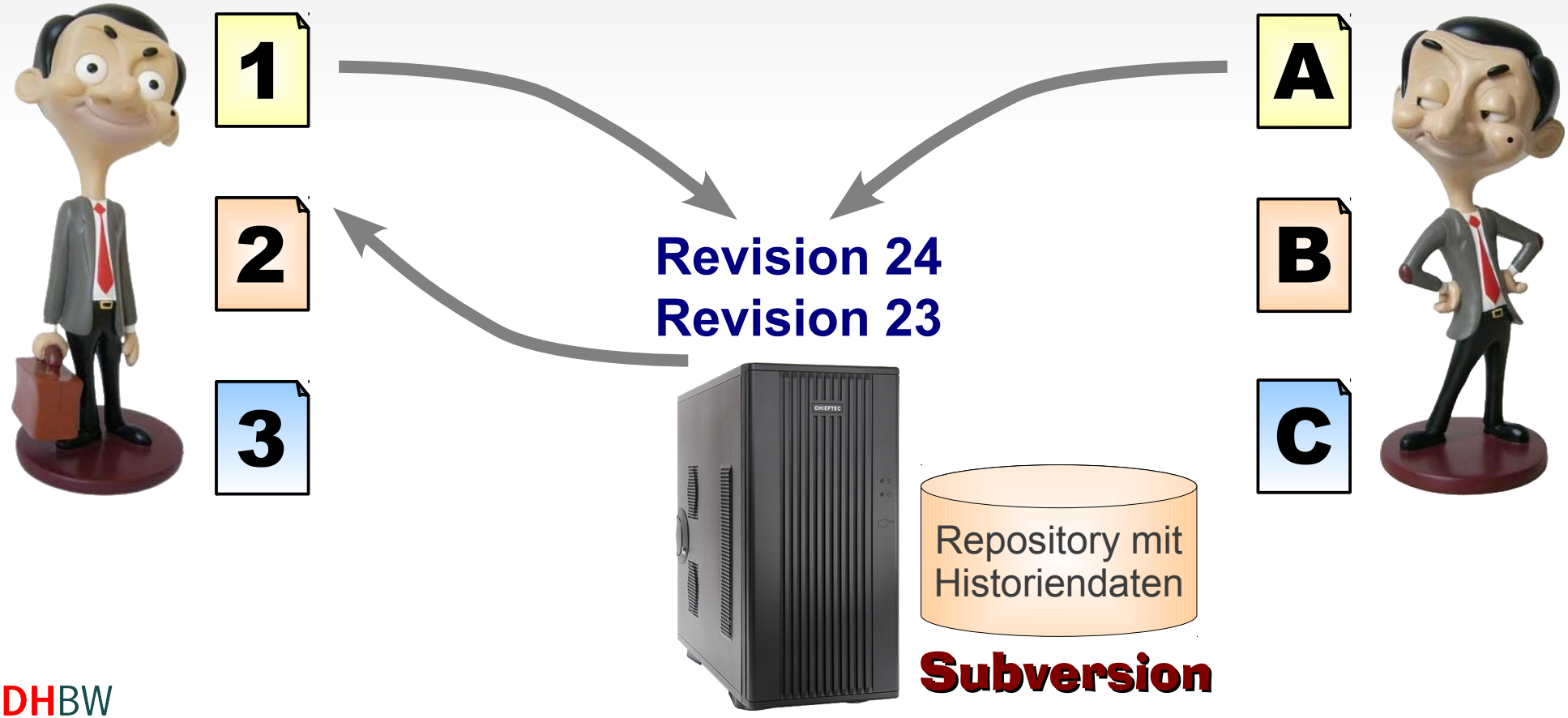
Verteilte Versionskontrollsysteme (Bitkeeper, Git, Mercurial):

- Jeder Entwickler besitzt ein eigenes, lokales Repository
- Versionskontrolle daher auch ohne Internetzugang möglich
- Übertragung des Repositories an beliebige Server möglich
- Abgleich des Repositories ebenso mit beliebigen Servern möglich

Zentrale Versionskontrolle

1. Jeder Benutzer überträgt nur seine letzten Änderungen an den Server.

2. Der Server verwaltet eine lückenlose Änderungsaufzeichnung. Jede Änderung der anderen Benutzer kann automatisch vom Server abgerufen werden.

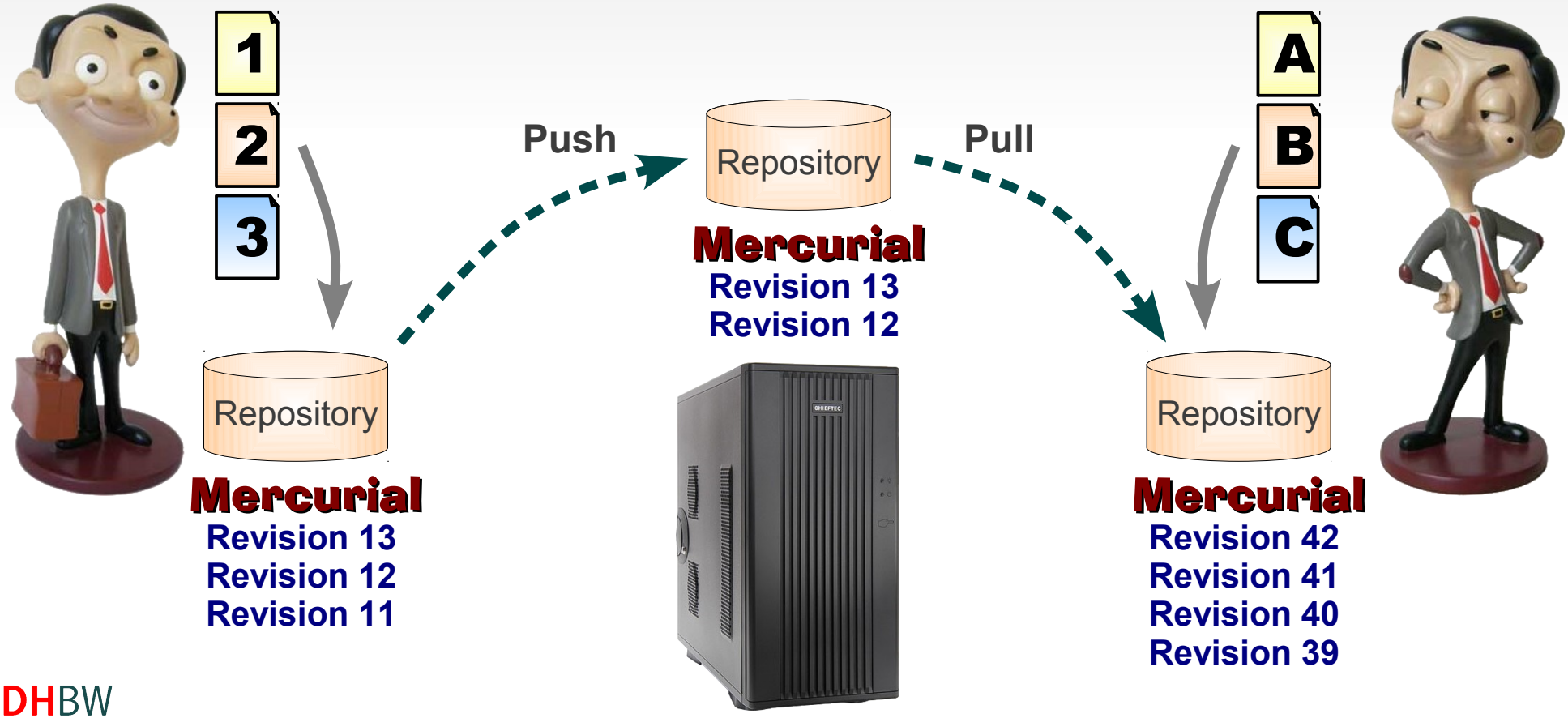


Verteilte Versionskontrolle

1. Jeder Entwickler besitzt sein eigenes, lokales Repository mit Historienfunktion. Zum Arbeiten ist kein Server notwendig.

2. Das eigene Repository kann auf beliebige Server hochgeladen werden (Push).

3. Das lokale Repository kann mit beliebigen Servern abgeglichen werden (Pull).



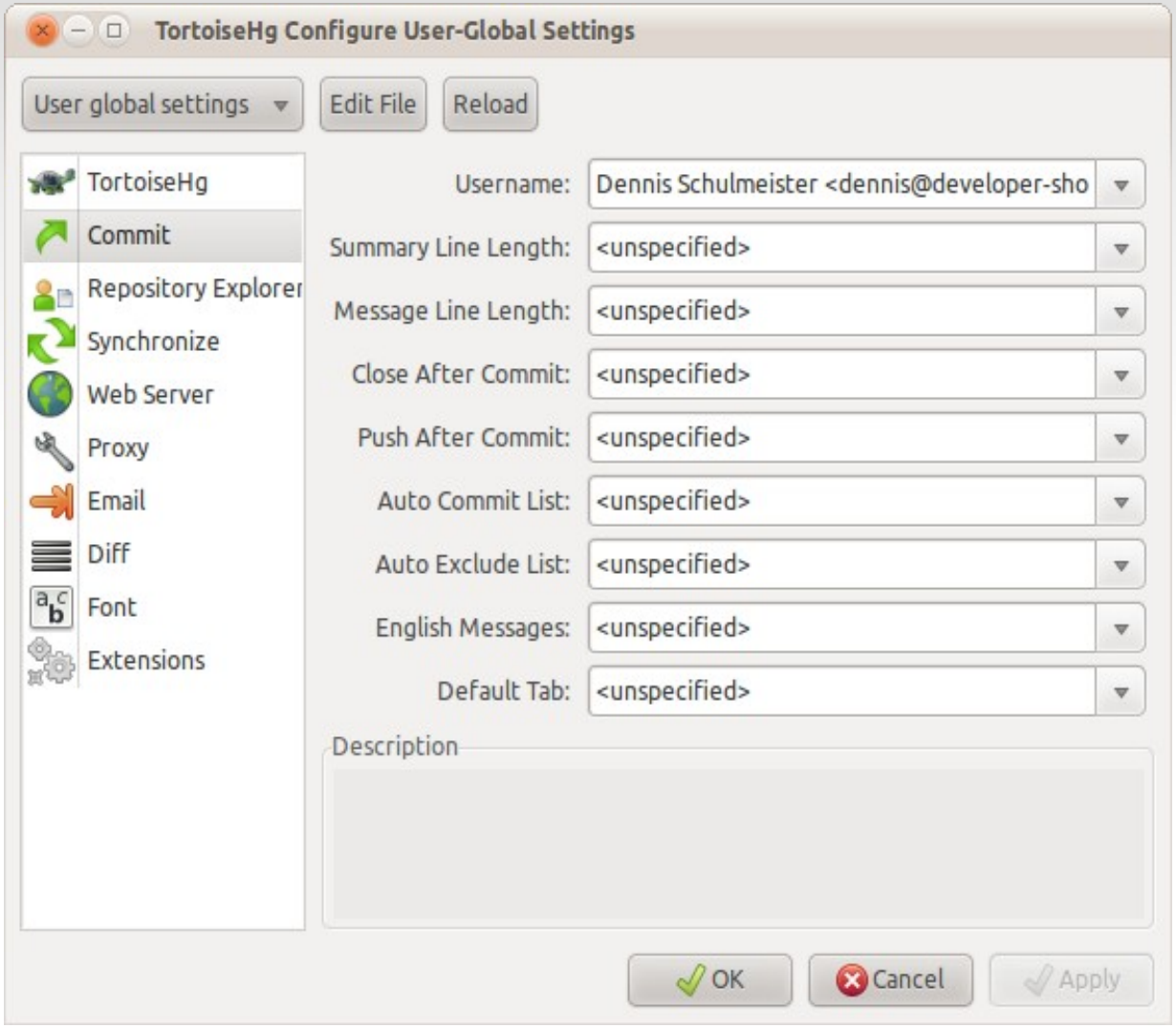
Mercurial Grundfunktionen

Benutzername festlegen:

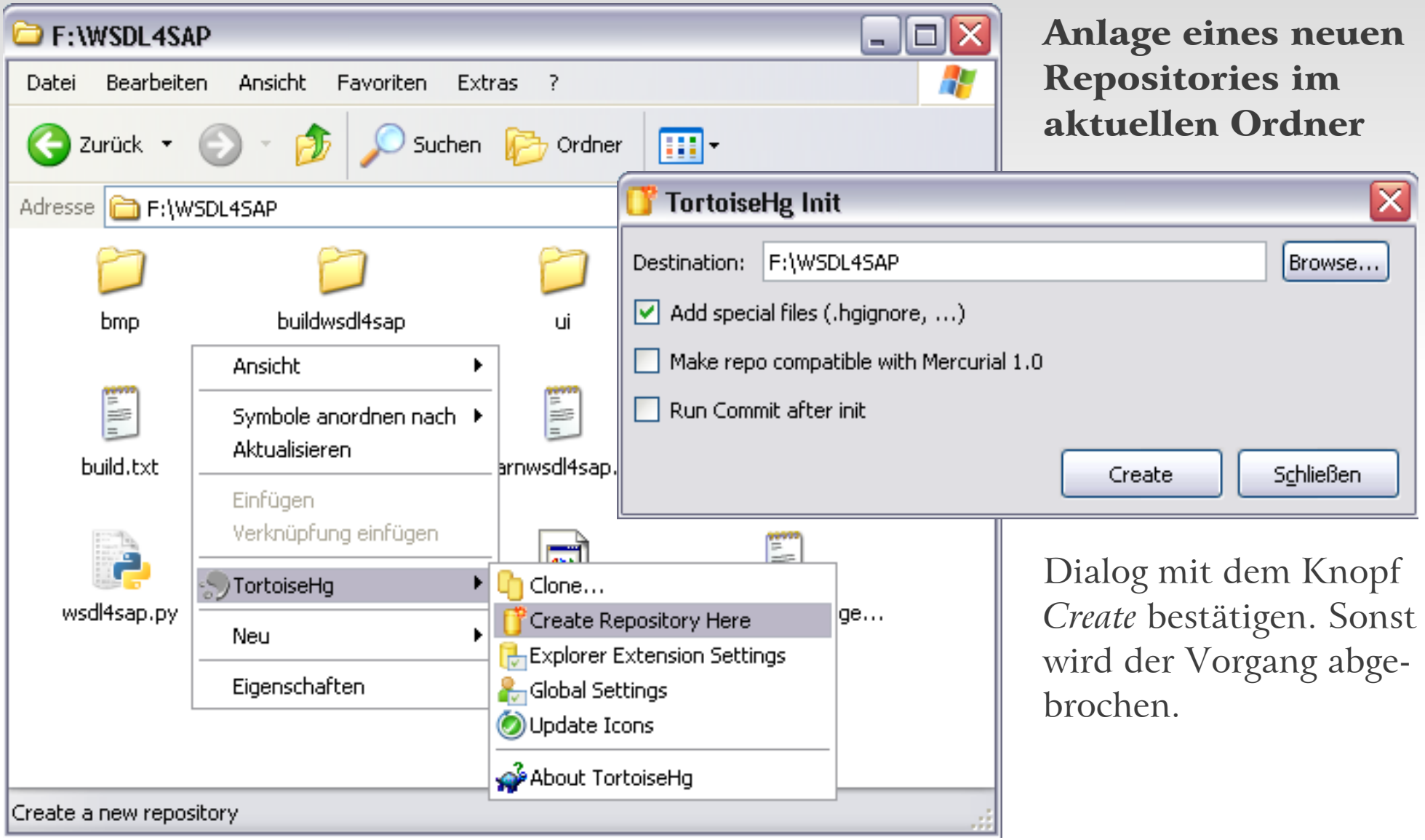
» Rechtsklick in einem beliebigen Ordner
» *TortoiseHG → Global Settings*
» Linkerhand *Commit* auswählen
» Feld *Username* ausfüllen

Übliche Konvention:

`Vorname Nachname <E-Mail>`



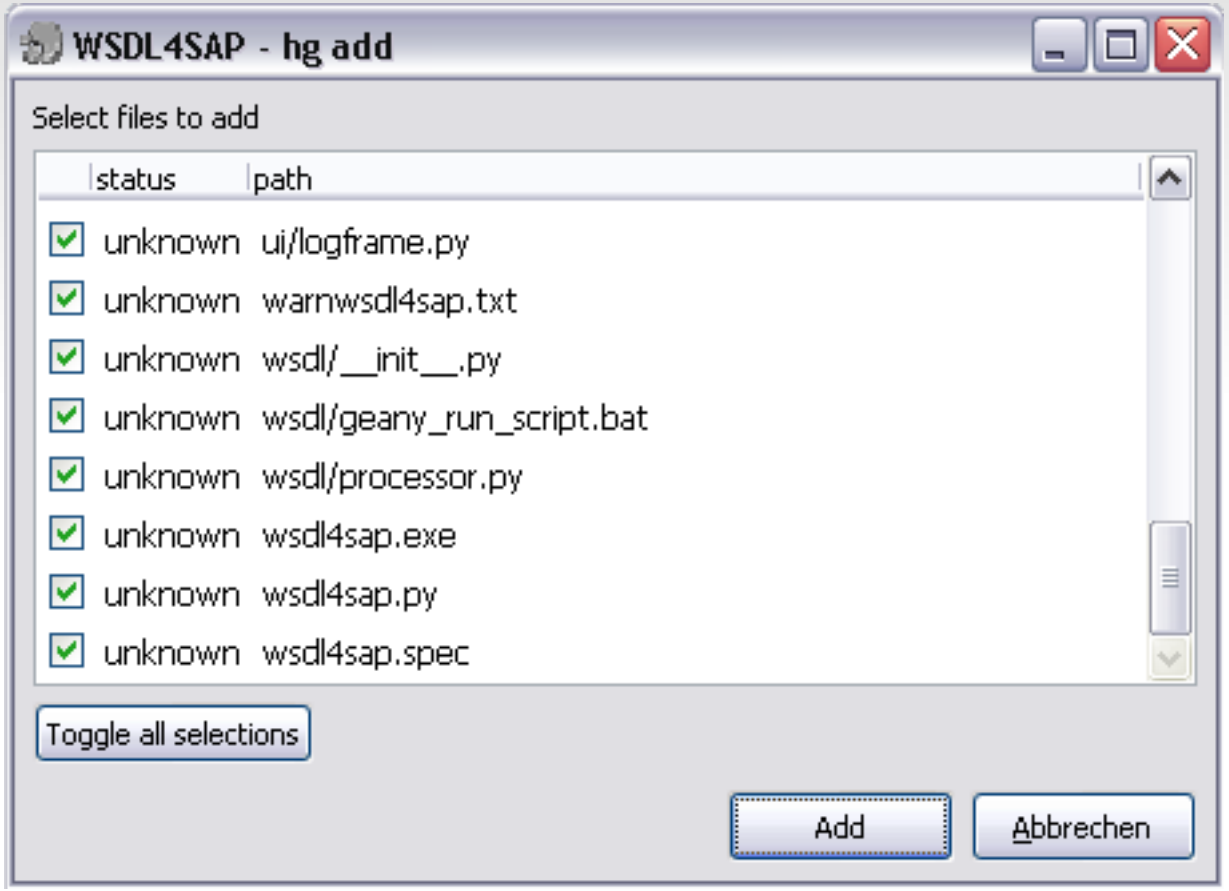
Mercurial Grundfunktionen



Anlage eines neuen Repositories im aktuellen Ordner

Dialog mit dem Knopf *Create* bestätigen. Sonst wird der Vorgang abgebrochen.

Mercurial Grundfunktionen



Neue Dateien hinzufügen:

» Rechtsklick auf eine Datei
» *TortoiseHG* → *Add Files*
» Fenster mit *Add* bestätigen
» *Commit* ausführen

Dieser Vorgang muss immer wiederholt werden, wenn dem Repository neue Dateien hinzugefügt werden sollen. Neue Dateien werden nicht automatisch unter Versionskontrolle gestellt!

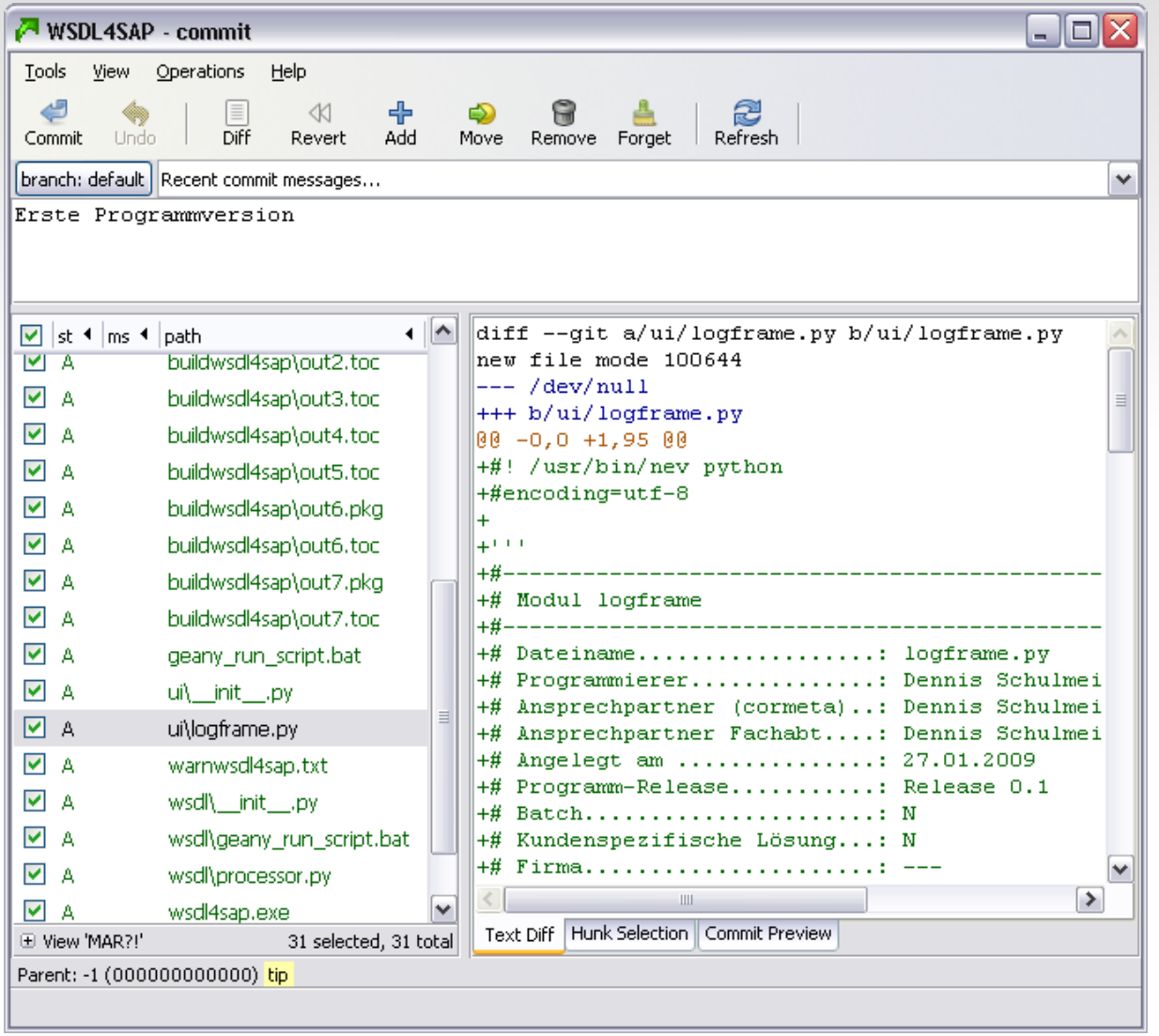
Die Dateien werden erst beim nächsten *Commit* dem Repository hinzugefügt.

Mercurial Grundfunktionen

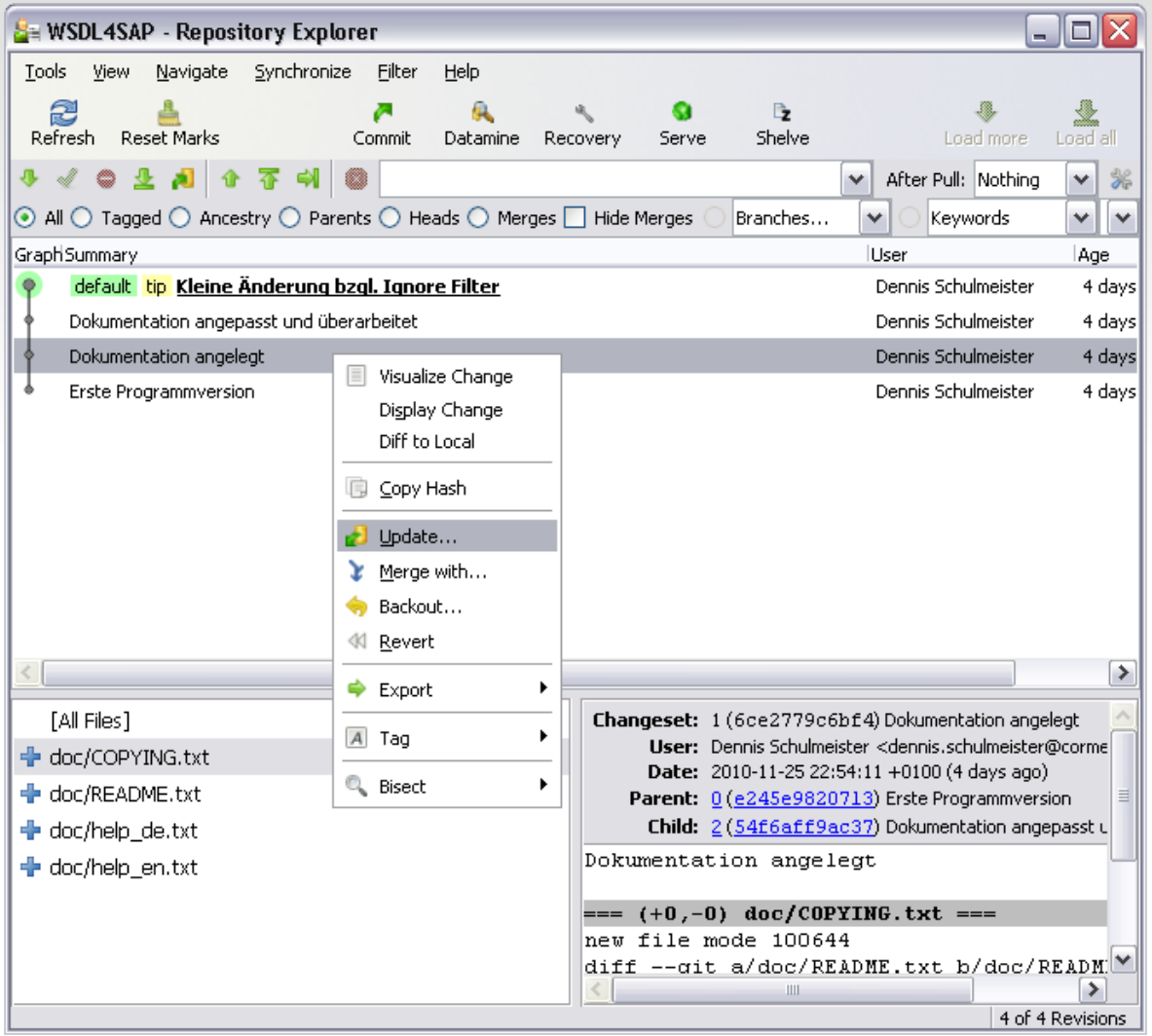
Änderungen fortschreiben (Commit genannt):

» Rechtsklick im Ordner
» Menüeintrag *Commit* auswählen
» Angezeigte Änderungen prüfen und mit *Commit* bestätigen
» Fenster schließen

Dieser Vorgang führt zu einem neuen Eintrag in der lokalen Änderungshistorie. Darum muss nach jeder größeren Änderung ein *Commit* ausgeführt werden, um den Zwischenstand der Arbeit zu sichern.



Mercurial Grundfunktionen



Änderungshistorie anzeigen:

» Rechtsklick im Ordner
» *Hg Repository Explorer* auswählen

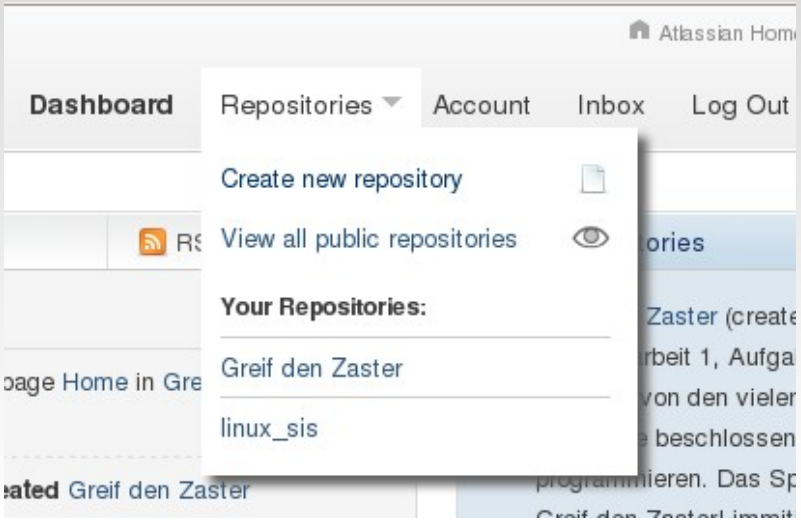
Es erscheint eine graphische Liste, die alle vorgenommenen Änderungen anzeigt. Jeder Eintrag entspricht einem *Commit*. Der Eintrag selbst wird auch *Change Set* genannt.

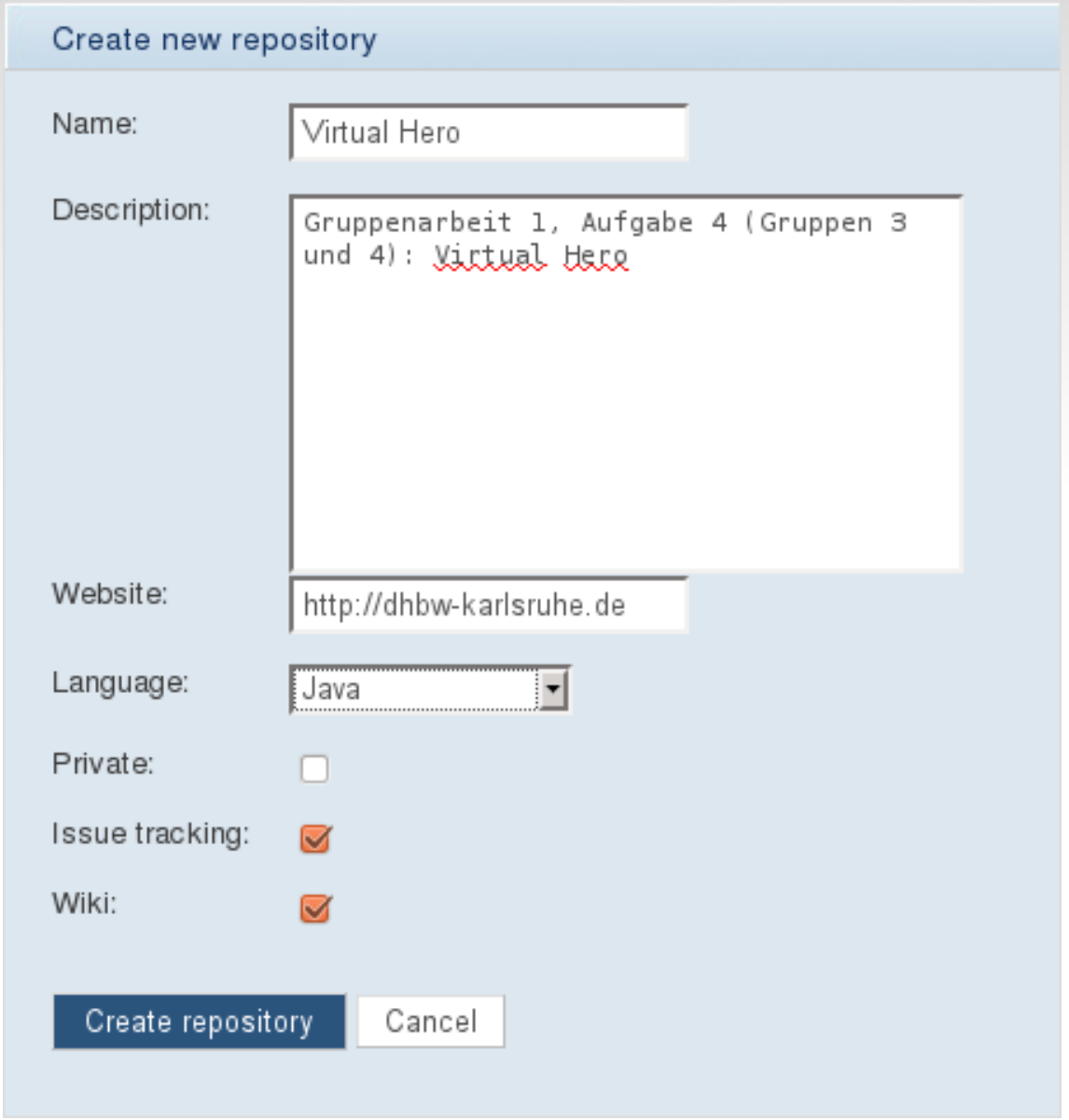
Wechsel der aktiven Version (Update genannt):

» Rechtsklick auf einem Change Set
» Menüeintrag *Update* auswählen
» Sicherheitsabfrage bestätigen

Führt dazu, dass die bearbeiteten Dateien des Arbeitsverzeichnisses angepasst werden.

Nutzung von Bitbucket





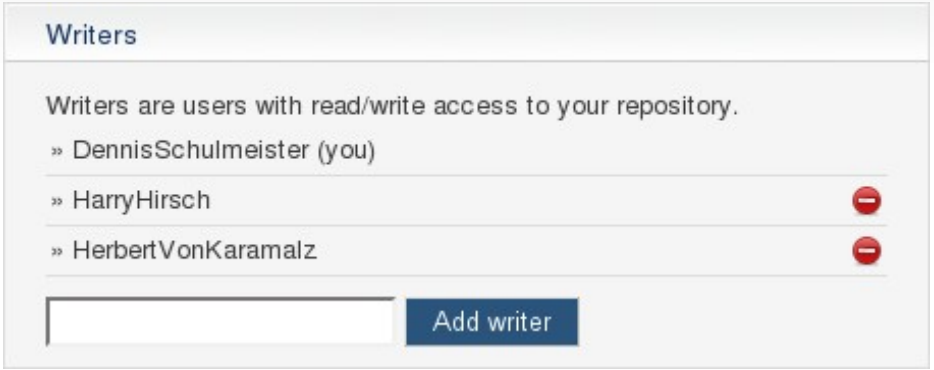
Neues Repository anlegen:

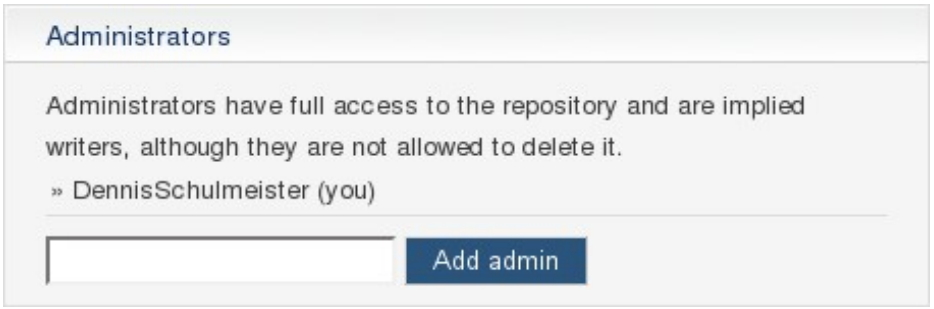
» Menü *Repositories* öffnen
» *Create new repository* auswählen
» Name und Beschreibung pflegen und Formular abschicken

Nutzung von Bitbucket

Schreibrechte für andere Benutzer:

» Neu angelegtes Repository anzeigen
» Tabreiter *Admin* öffne
» Bitbucket-Benutzernamen der anderen Benutzer bei *Writers* oder *Administrators* hinzufügen





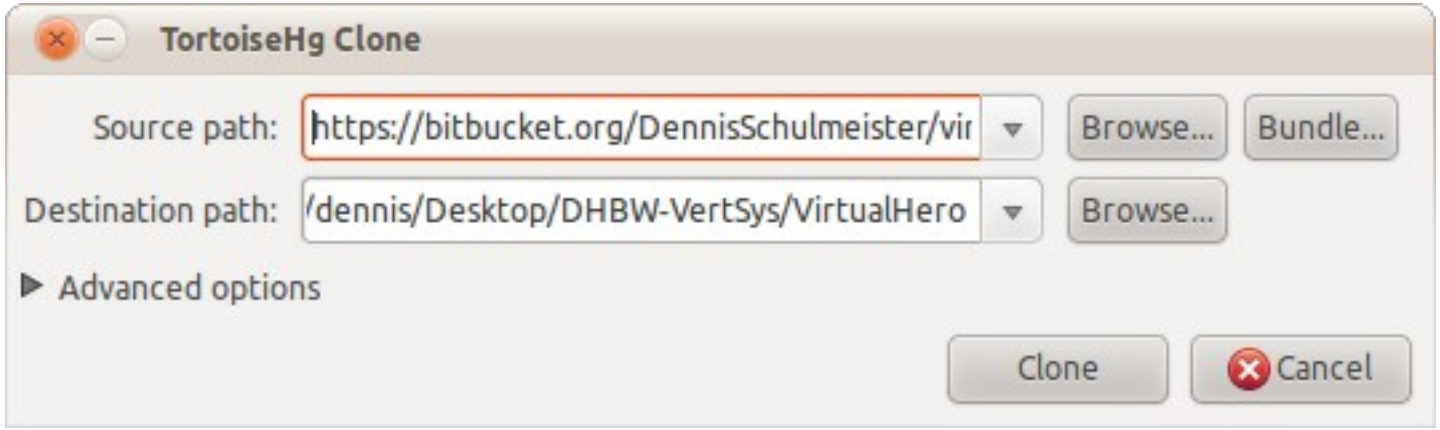
Auf öffentliche Repositories hat automatisch jeder Lesezugriff. *Writers* können die Dateien des Repositories ändern, *Administrators* auch das Repository selbst.

Nutzung von Bitbucket

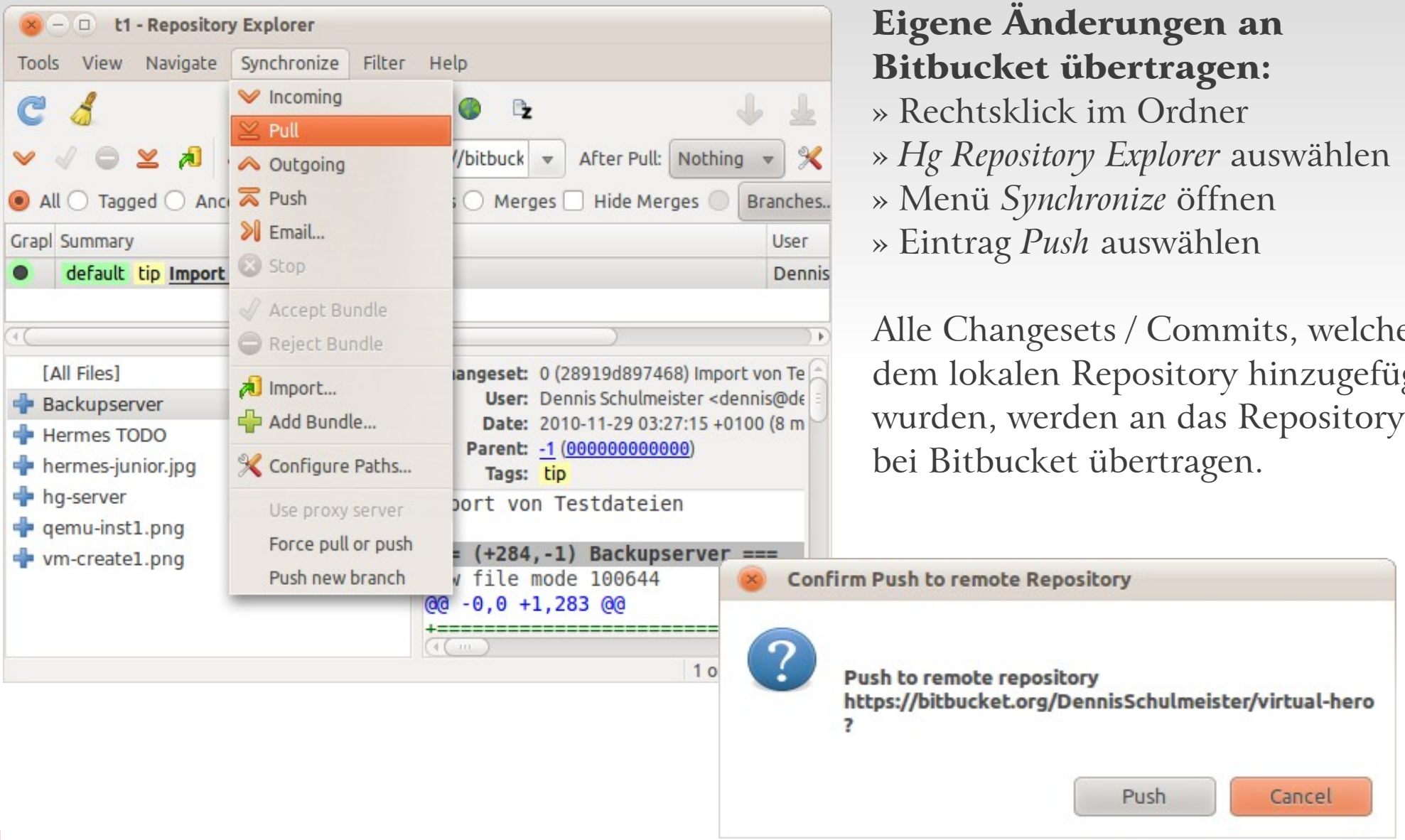
Repository auf den eigenen Rechner übertragen

Damit alle Mitglieder einer Gruppe mit dem Repository arbeiten können, müssen sie es auf ihre Computer kopieren (klonen). Alle Änderungen und *Commits* werden dann an der lokalen Kopie des Repositories vorgenommen. Die eigenen Änderungen können aber zu jeder Zeit an Bitbucket zurückgesendet werden.

» Neuen Ordner anlegen und öffnen
» Kontextmenü öffnen (Rechtsklick)
» Eintrag *TortoiseHG* → *Clone* auswählen
» Webadresse des Repositories in das Feld *Source path* eintragen
» Vorgang mit *Clone* bestätigen



Nutzung von Bitbucket



Eigene Änderungen an Bitbucket übertragen:

» Rechtsklick im Ordner
» *Hg Repository Explorer* auswählen
» Menü *Synchronize* öffnen
» Eintrag *Push* auswählen

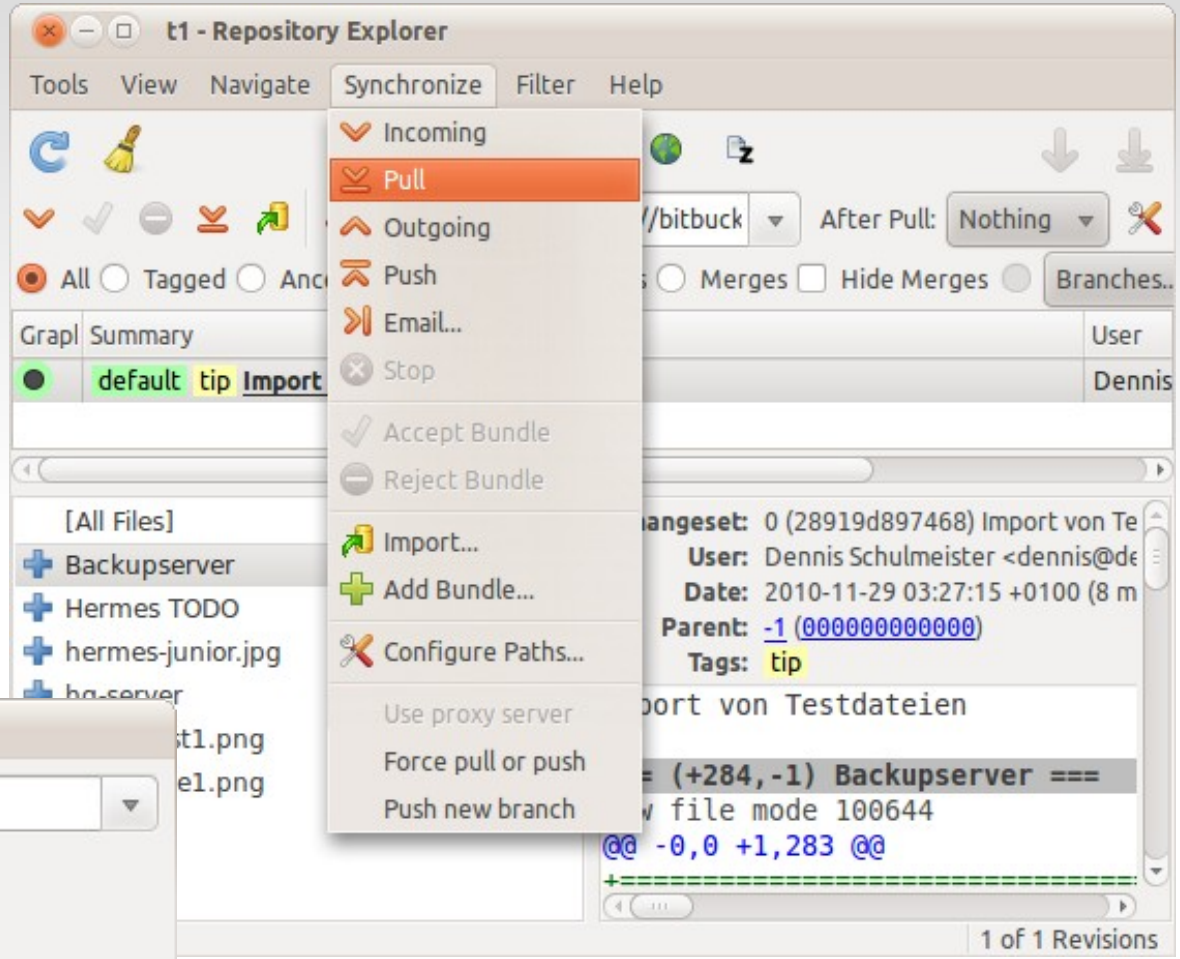
Alle Changesets / Commits, welche dem lokalen Repository hinzugefügt wurden, werden an das Repository bei Bitbucket übertragen.

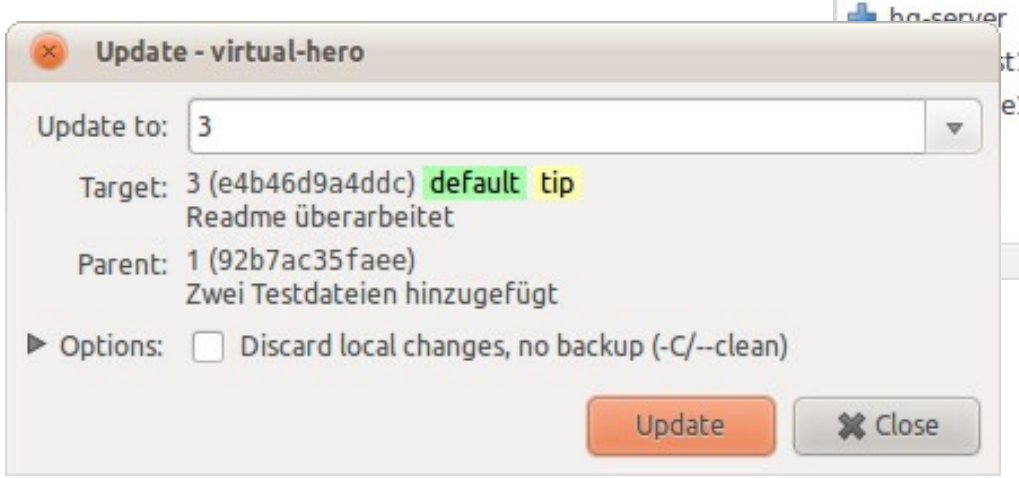
Nutzung von Bitbucket

Abgleich mit Bitbucket:

» Rechtsklick im Ordner
» *Hg Repository Explorer* auswählen
» Menü *Synchronize* öffnen
» Eintrag *Pull* auswählen

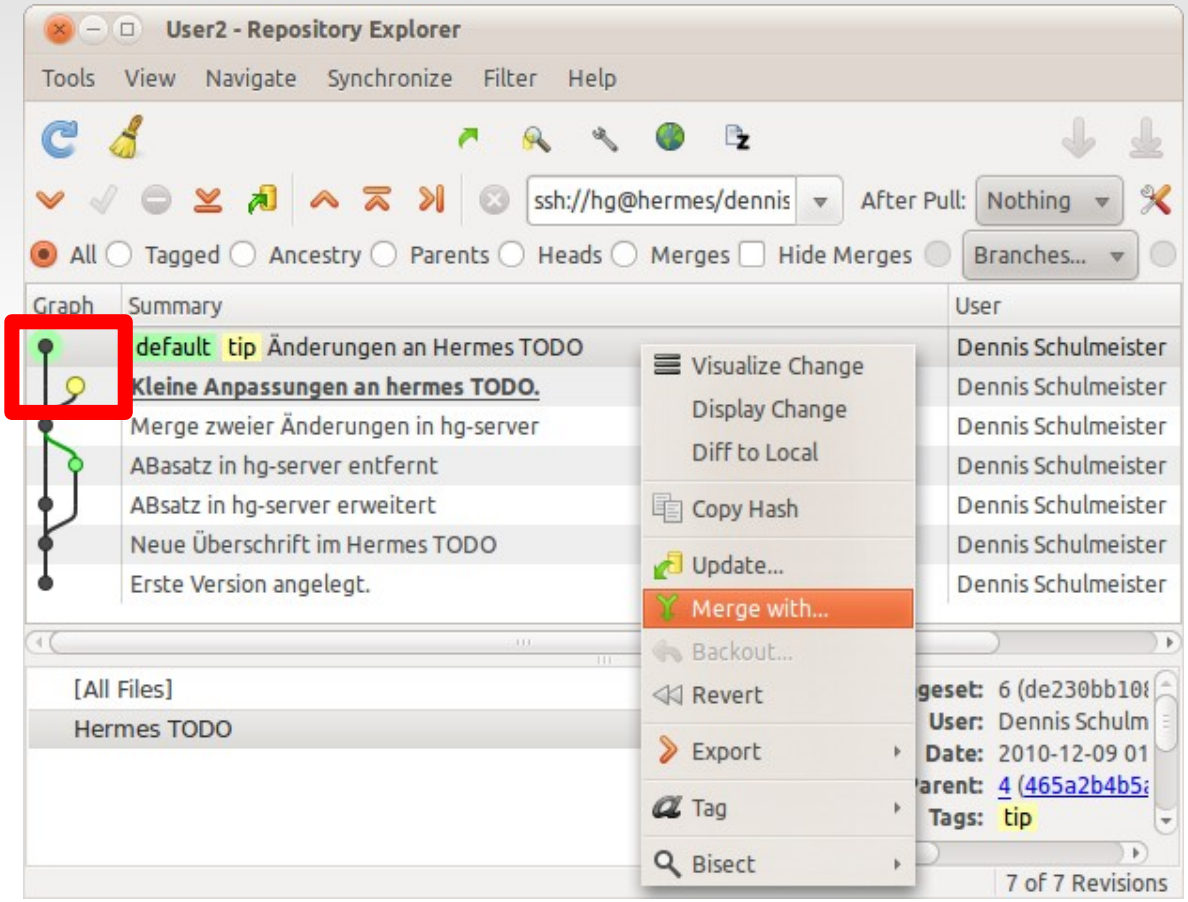
Es werden alle Changesets von Bitbucket abgerufen, die Dateien des Arbeitsverzeichnisses ändern sich aber nicht. Sie werden durch ein anschließendes *Update* aktualisiert.





Konflikte zwischen Ihrem Repository und der bei Bitbucket abgelegten Version müssen durch ein *Merge* aufgelöst werden, bevor Sie mit dem Repository arbeiten können.

Konflikte auflösen



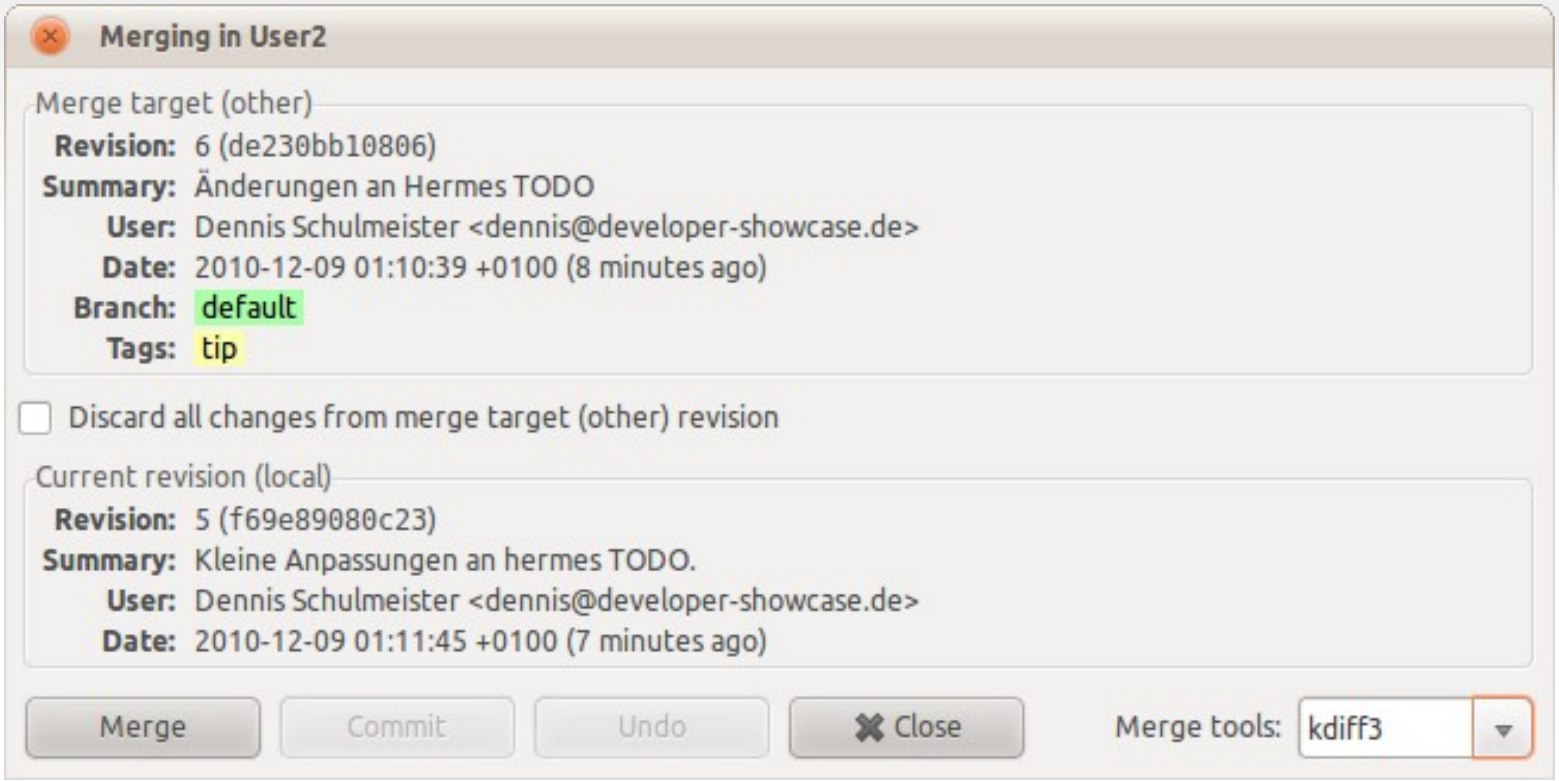
Konflikte behandeln:

Ein Konflikt entsteht immer dann, wenn zwei Personen dieselbe Steller einer Datei unabhängig voneinander verändern. Der Konflikt wird im Repository Browser durch die zwei offenen Enden (*Heads* genannt) im Versionsgraphen dargestellt.

» Kontextmenü der neusten Version
» Eintrag *Merge with...* auswählen
» Es erscheint ein neues Fenster
» Darin auf *Merge* drücken
» Konflikt lösen
» Änderung per *Commit* bestätigen

Konflikte auflösen

Das Merge-Fenster zeigt, welche Revisionen miteinander verschmolzen werden sollen. Durch einen Klick auf *Merge* wird der Vorgang angestoßen. Es öffnet sich ein externes Programm, in welchem der Konflikt behoben werden muss.

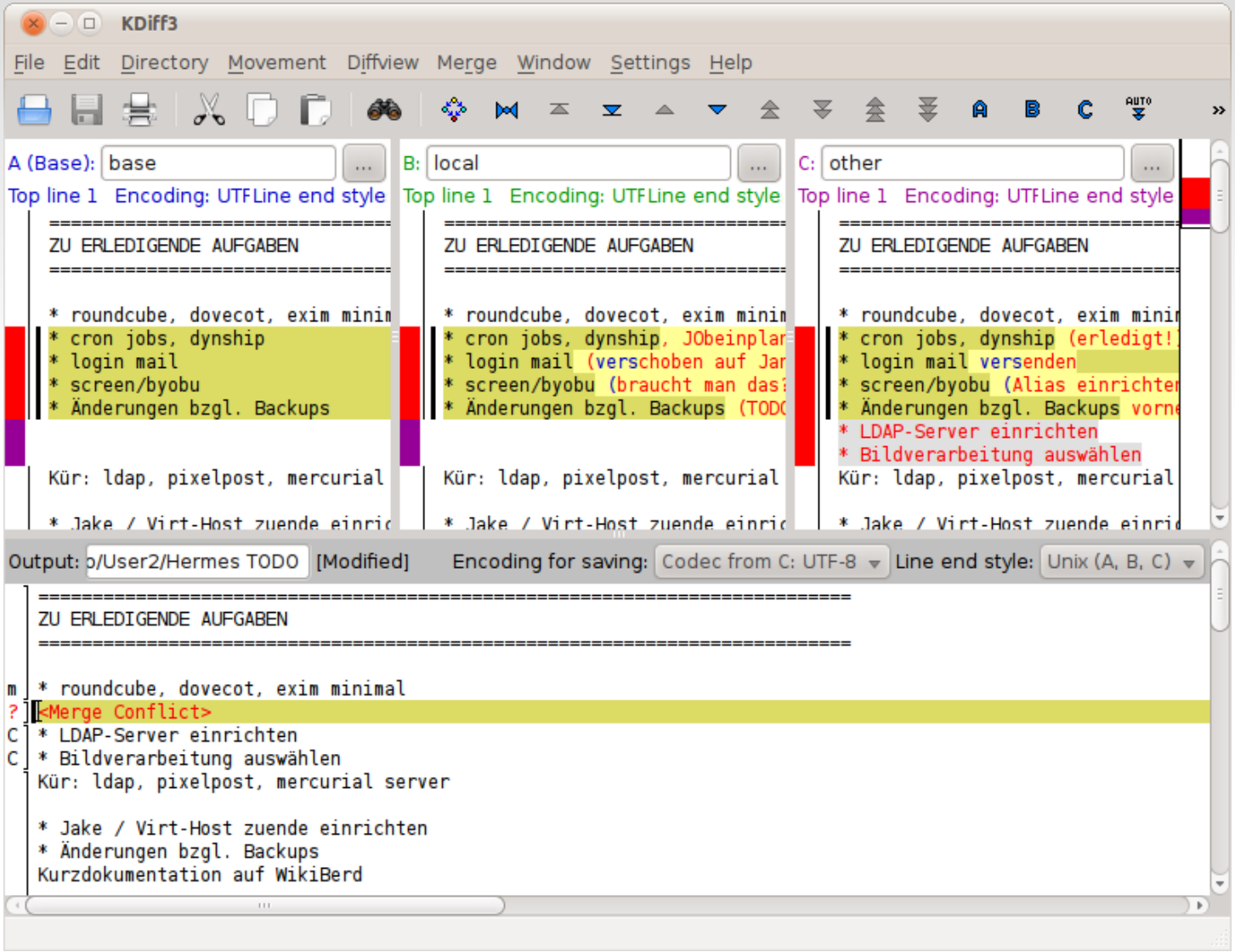


Konflikte auflösen

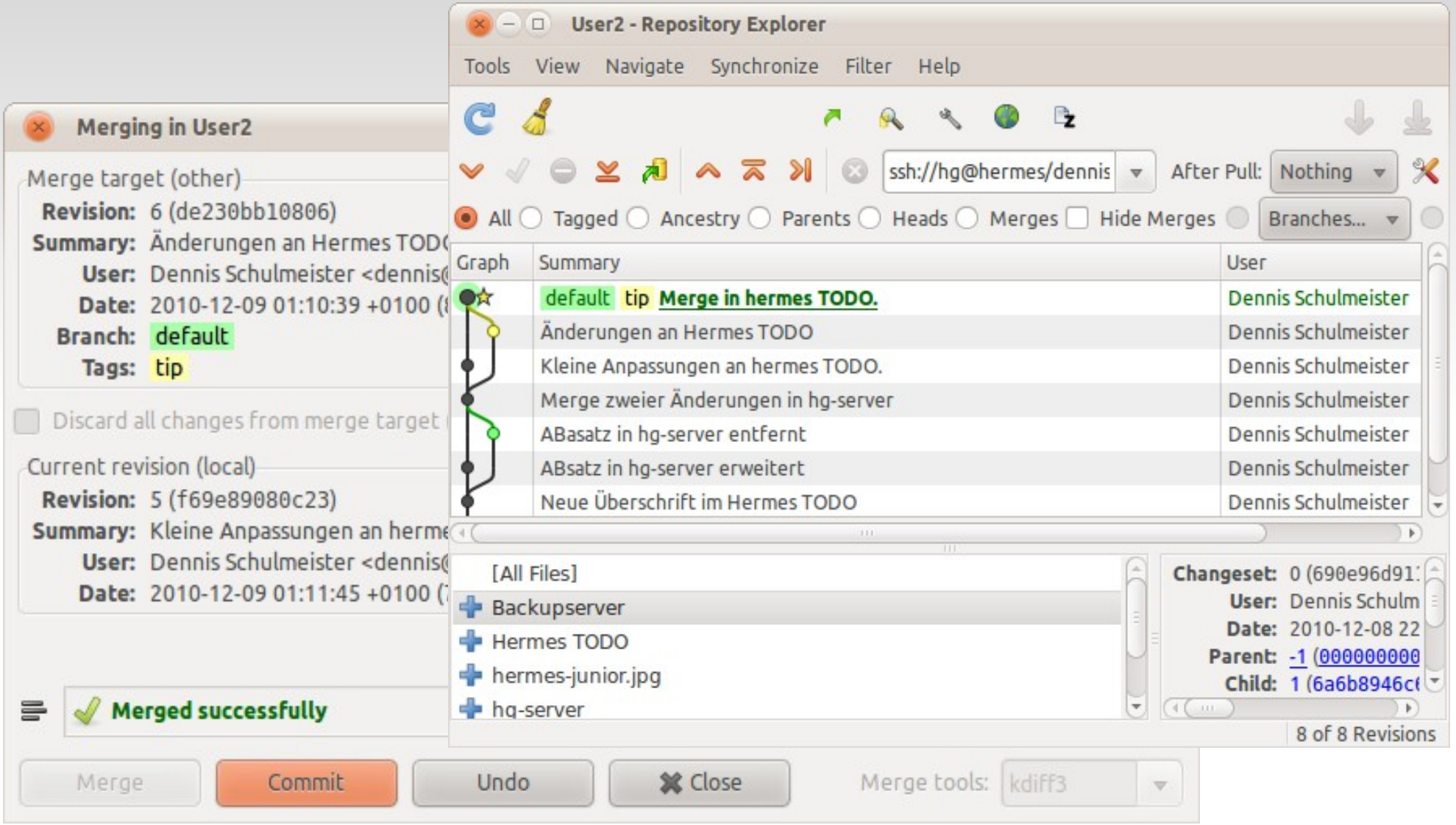
Die meisten Programme (hier KDiff3) führen einen *Three Way-Merge* durch. Das heißt, folgende Informationen werden angezeigt:

» Ursprüngliche Datei
» Eigene Änderungen
» Fremde Änderungen

Im unteren Bereich von KDiff3 müssen Sie die Datei bearbeiten. Dann die Datei speichern und Programm schließen.



Konflikte auflösen



Zum Schluss wird die Änderung durch ein *Commit* ins Repository übernommen.

Empfohlenes Vorgehen

1) Abgleich mit Bitbucket

- Neuste Version von Bitbucket herunterladen (Pull)
- Falls erforderlich Merge und Commit ausführen

2) Änderungen vornehmen

- Dateien bearbeiten
- Regelmäßiges Commit ausführen

3) Änderungen an Bitbucket senden

- Neuste Version von Bitbucket herunterladen (Pull)
- Falls erforderlich Merge und Commit ausführen
- Änderungen übertragen (Push)