

A1 Git: Repository erzeugen

1. Lassen Sie sich die installierte Version von Git ausgeben.
2. Konfigurieren Sie in Git Ihren Benutzernamen und Ihre E-Mail-Adresse. Überprüfen Sie, ob die Konfiguration richtig ist.
3. Erzeugen Sie auf einem Git-Server Ihrer Wahl, zum Beispiel auf dem GitLab-Server der FH Aachen, ein leeres Remote-Repository.
4. Klonen Sie das neue Repository in ein neues Arbeitsverzeichnis auf Ihrem Rechner.
5. Erstellen Sie in dem neuen Arbeitsverzeichnis eine Hello-World-Anwendung in der Programmiersprache Ihrer Wahl, z.B. Java, C# oder Python (die Sprache sollte allerdings auf Textdateien mit Quellcode basieren).
6. Lassen Sie sich den aktuellen Status ausgeben und übertragen Sie die Änderungen in Ihr lokales Repository.
7. Bringen Sie das Remote-Repository auf den neuesten Stand.
8. Ergänzen Sie in Ihrem Quellcode einige Inline-Kommentare, z.B. Autor und Datum.
9. Übertragen Sie alle Änderungen in das Remote-Repository.
10. Ergänzen Sie eine geeignete .gitignore-Datei für Ihr Projekt, um keine generierten Dateien in die Repositories zu übertragen. Löschen Sie ggf. alle Dateien im lokalen und im Remote-Repositories, die dort nicht hingehören.
11. Benennen Sie mindestens eine Datei lokal um und stellen Sie sicher, dass die Umbenennung auch im Remote-Repository erfolgreich war.
12. Führen Sie eine Änderung am Quellcode durch und machen Sie die Änderung wieder rückgängig, indem Sie den letzten Stand des lokalen Repository wiederherstellen.

Lösung 1

```
^ > ~/tmp > x 128 git version 138866 < 22:07:12 < 1.27
git version 2.42.0
^ > ~/tmp > git config --get user.name 138867 < 22:07:36 < 1.25
Christian Rene Thelen
^ > ~/tmp > git config --get user.email 138868 < 22:07:38 < 1.23
christian.thelen@rwth-aachen.de
^ > ~/tmp > git clone git@git.fh-aachen.de:cortex/swt_h06.git 138869 < 22:07:41 < 1.23
Klone nach 'swt_h06'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Empfange Objekte: 100% (3/3), fertig.
^ > ~/tmp > touch swt_h06/hello_world.py 138870 < 22:07:58 < 1.21
^ > ~/tmp > cd swt_h06 138871 < 22:08:27 < 1.22
drwxr----- - cthelen 19 Nov 22:07 .git/
-rw-r----- 6,2k cthelen 19 Nov 22:07 README.md
-rw-r----- 0 cthelen 19 Nov 22:08 hello_world.py
^ > ~/tm/swt_h06 > git main ?1 > git status 138872 < 22:08:36 < 1.25
Auf Branch main
Ihr Branch ist auf demselben Stand wie 'origin/main'.

Unversionierte Dateien:
(benutzen Sie "git add <Datei>...", um die Änderungen zum Commit vorzumerken)
hello_world.py

nichts zum Commit vorgemerkt, aber es gibt unversionierte Dateien
(benutzen Sie "git add" zum Versionieren)
^ > ~/tm/swt_h06 > git main ?1 > git add hello_world.py 138873 < 22:08:40 < 1.23
^ > ~/tm/swt_h06 > git main +1 > git commit -m 'Diese Hausaufgabe ist langweilig...'
[main 0c71894] Diese Hausaufgabe ist langweilig...
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 hello_world.py
^ > ~/tm/swt_h06 > git main +1 > git push 138875 < 22:10:32 < 1.14
Objekte aufzählen: 4, fertig.
Zähle Objekte: 100% (4/4), fertig.
Delta-Kompression verwendet bis zu 8 Threads.
Komprimiere Objekte: 100% (2/2), fertig.
Schreibe Objekte: 100% (3/3), 323 Bytes | 161.00 KiB/s, fertig.
Gesamt 3 (Delta 0), Wiederverwendet 0 (Delta 0), Pack wiederverwendet 0
To git.fh-aachen.de:cortex/swt_h06.git
e6797bc..0c71894 main -> main
^ > ~/tm/swt_h06 > git main > 138876 < 22:10:35 < 1.17
```

Abbildung 1: Aufgabe 1, Nr. 1-7

```
A > ~/tm/swt_h06 > git P main > ✓ echo '# Autor und Datum' >> hello_world.py 138876 < 22:10:35 < 1.17
A > ~/tm/swt_h06 > git P main !1 > ✓ git add hello_world.py 138877 < 22:13:32 < 1.49
A > ~/tm/swt_h06 > git P main +1 > ✓ git commit -m 'Inline-Kommentare ergänzt'
[main 4e3b461] Inline-Kommentare ergänzt
1 file changed, 1 insertion(+)
A > ~/tm/swt_h06 > git P main +1 > ✓ git push 138878 < 22:14:18 < 1.62
Objekte aufzählen: 5, fertig.
Zähle Objekte: 100% (5/5), fertig.
Delta-Kompression verwendet bis zu 8 Threads.
Komprimiere Objekte: 100% (2/2), fertig.
Schreibe Objekte: 100% (3/3), 333 Bytes | 333.00 KiB/s, fertig.
Gesamt 3 (Delta 0), Wiederverwendet 0 (Delta 0), Pack wiederverwendet 0
To git.fh-aachen.de:cortex/swt_h06.git
0c71894..4e3b461 main -> main
A > ~/tm/swt_h06 > git P main > ✓ wget 'https://gitignore.io/api/python' -O .gitignore
--2023-11-19 22:16:54-- https://gitignore.io/api/python
CA-Zertifikat »/etc/ssl/certs/ca-certificates.crt« wurde geladen
Auflösen des Hostnamens gitignore.io (gitignore.io)... 2a06:98c1:3121::3, 2a06:98c1:3120::3, 188.114.96.3, ...
Verbindungsaufbau zu gitignore.io (gitignore.io)|2a06:98c1:3121::3|:443 ... verbunden.
HTTP-Anforderung gesendet, auf Antwort wird gewartet ... 301 Moved Permanently
Platz: https://www.toptal.com/developers/gitignore/api/python [folgend]
--2023-11-19 22:16:55-- https://www.toptal.com/developers/gitignore/api/python
Auflösen des Hostnamens www.toptal.com (www.toptal.com)... 2606:4700::6812:1cd5, 2606:4700::6812:1dd5, 104.18.28.213
, ...
Verbindungsaufbau zu www.toptal.com (www.toptal.com)|2606:4700::6812:1cd5|:443 ... verbunden.
HTTP-Anforderung gesendet, auf Antwort wird gewartet ... 200 OK
Länge: 3494 (3,4K) [text/plain]
Wird in ».gitignore« gespeichert.

.gitignore          100%[=====] 3,41K --.-KB/s   in 0,001s
2023-11-19 22:16:55 (6,16 MB/s) - ».gitignore« gespeichert [3494/3494]
A > ~/tm/swt_h06 > git P main ?1 > ✓ git add .gitignore 138881 < 22:16:55 < 1.39
A > ~/tm/swt_h06 > git P main +1 > ✓ git mv hello_world.py Hey_World.py 138882 < 22:17:08 < 1.34
A > ~/tm/swt_h06 > git P main +3 > ✓ git status 138883 < 22:17:45 < 1.27
Auf Branch main
Ihr Branch ist auf demselben Stand wie 'origin/main'.

Zum Commit vorgemerkte Änderungen:
(benutzen Sie "git restore --staged <Datei>..." zum Entfernen aus der Staging-Area)
neue Datei:      .gitignore
umbenannt:      hello_world.py -> Hey_World.py
A > ~/tm/swt_h06 > git P main +3 > ✓ git commit -m 'A1 Nr. 8-10' 138884 < 22:17:56 < 1.24
[main 1cef230] A1 Nr. 8-10
2 files changed, 176 insertions(+)
create mode 100644 .gitignore
rename hello_world.py => Hey_World.py (100%)
A > ~/tm/swt_h06 > git P main +1 > ✓ | 138885 < 22:19:07 < 1.14
```

Abbildung 2: Aufgabe 1, Nr. 8-11

```
A > ~/tm/swt_h06 > git push main +1 > ✓ git push 138885 < 22:19:07 < 1.14
Objekte aufzählen: 4, fertig.
Zähle Objekte: 100% (4/4), fertig.
Delta-Kompression verwendet bis zu 8 Threads.
Komprimiere Objekte: 100% (3/3), fertig.
Schreibe Objekte: 100% (3/3), 1.89 KiB | 1.89 MiB/s, fertig.
Gesamt 3 (Delta 0), Wiederverwendet 0 (Delta 0), Pack wiederverwendet 0
To git.fh-aachen.de:cortex/swt_h06.git
4e3b461..1cef230 main -> main
A > ~/tm/swt_h06 > git push main > ✓ echo 'Fehlerhafte Änderung' >> Hey_World.py 138886 < 22:20:04 < 1.12
A > ~/tm/swt_h06 > git push main !1 > ✓ git status 138887 < 22:20:35 < 1.08
Auf Branch main
Ihr Branch ist auf demselben Stand wie 'origin/main'.

Änderungen, die nicht zum Commit vorgemerkt sind:
(benutzen Sie "git add <Datei>...", um die Änderungen zum Commit vorzumerken)
(benutzen Sie "git restore <Datei>...", um die Änderungen im Arbeitsverzeichnis zu verwerfen)
geändert: Hey_World.py

keine Änderungen zum Commit vorgemerkt (benutzen Sie "git add" und/oder "git commit -a")
A > ~/tm/swt_h06 > git push main !1 > ✓ git restore 138888 < 22:20:39 < 1.06
Schwerwiegend: Sie müssen Pfad(e) zur Wiederherstellung angeben.
[1] 333174 exit 128 git restore
A > ~/tm/swt_h06 > git push main !1 > x 128 git restore Hey_World.py 138889 < 22:21:00 < 1.04
A > ~/tm/swt_h06 > git push main > ✓ git status 138890 < 22:21:03 < 1.04
Auf Branch main
Ihr Branch ist auf demselben Stand wie 'origin/main'.

nichts zu committen, Arbeitsverzeichnis unverändert
A > ~/tm/swt_h06 > git push main > ✓ cat Hey_World.py 138891 < 22:21:09 < 1.05
# Autor und Datum
A > ~/tm/swt_h06 > git push main > ✓ | 138892 < 22:21:12 < 1.05
```

Abbildung 3: Aufgabe 1, Nr. 12

A2 Git: Konflikte bereinigen

Nutzen Sie weiterhin Ihr erstelltes Repository aus Aufgabe A1.

1. Für diesen Aufgabenteil müssen zwei entwickelnde Personen an verschiedenen Rechnern mit eigenen Klonen des Repository arbeiten. Alternativ können Sie auch an einem Rechner einen zweiten Klon des Repository in einem anderen Verzeichnis erzeugen und über `git config --local` einen anderen Benutzernamen und eine andere Mailadresse wählen. Die verschiedenen Namen der Entwickler sind für den Commit-Verlauf wichtig.
2. Stellen Sie zunächst sicher, dass beide Repositories auf dem aktuellen Stand sind.
3. Erzeugen Sie nun in einer bestehenden Quelldatei einen Konflikt in einer Codezeile, z.B. indem Sie im ersten Repository „Hello World!“ durch einen Parameter und im zweiten Repository durch einen String in einer anderen Sprache ersetzen.
4. Versuchen Sie nun, beide Änderungen in das Remote-Repository zu übertragen. Bei einem Repository wird es gelingen, beim anderen wird es mit einer Fehlermeldung scheitern.
5. Beheben Sie den Konflikt in dem Repository, bei dem das Übertragen gescheitert ist, indem Sie beide Änderungen geeignet kombinieren.
6. Übertragen Sie dann die neue, konfliktfreie Version zurück ins Remote-Repository.
7. Prüfen Sie den Commit-Verlauf mit `git log --graph`.

Lösung 2

```
A > ~/tm/swt_h06 > git P main +1 > ✓ cd ../swt_h06-b 138905 < 22:30:02 < 1.39
.rw-r----- 6,2k cthelen 19 Nov 22:27 README.md
.rw-r----- 18 cthelen 19 Nov 22:27 Hey_World.py
A > ~/tm/swt_h06-b > git P main > ✓ echo "print(hw)" >> Hey_World.py 138906 < 22:30:08 < 1.37
A > ~/tm/swt_h06-b > git P main !1 > ✓ git add Hey_World.py 138907 < 22:30:44 < 1.38
A > ~/tm/swt_h06-b > git P main +1 > ✓ git commit -m 'Andere Ausgabe hinzugefügt'
[main 77fc3c4] Andere Ausgabe hinzugefügt
1 file changed, 1 insertion(+)
A > ~/tm/swt_h06-b > git P main +1 > ✓ git push 138909 < 22:30:59 < 1.40
Objekte aufzählen: 5, fertig.
Zähle Objekte: 100% (5/5), fertig.
Delta-Kompression verwendet bis zu 8 Threads.
Komprimiere Objekte: 100% (2/2), fertig.
Schreibe Objekte: 100% (3/3), 372 Bytes | 186.00 KiB/s, fertig.
Gesamt 3 (Delta 0), Wiederverwendet 0 (Delta 0), Pack wiederverwendet 0
To git.fh-aachen.de:cortex/swt_h06.git
1cef230..77fc3c4 main -> main
A > ~/tm/swt_h06-b > git P main > ✓ cd ../swt_h06 138910 < 22:31:14 < 1.38
.rw-r----- 6,2k cthelen 19 Nov 22:07 README.md
.rw-r----- 39 cthelen 19 Nov 22:29 Hey_World.py
A > ~/tm/swt_h06 > git P main +1 > ✓ git push 138911 < 22:31:21 < 1.37
To git.fh-aachen.de:cortex/swt_h06.git
! [rejected] main -> main (fetch first)
Fehler: Fehler beim Versenden einiger Referenzen nach 'git.fh-aachen.de:cortex/swt_h06.git'
Hinweis: Aktualisierungen wurden zurückgewiesen, weil das Remote-Repository Commits enthält,
Hinweis: die lokal nicht vorhanden sind. Das wird üblicherweise durch einen "push" von
Hinweis: Commits auf dieselbe Referenz von einem anderen Repository aus verursacht.
Hinweis: Wenn Sie die externen Änderungen integrieren wollen, verwenden Sie 'git pull'
Hinweis: bevor Sie erneut push ausführen.
Hinweis: Siehe auch die Sektion 'Note about fast-forwards' in 'git push --help' für weitere
Hinweis: Informationen.
[1] 337011 exit 1 git push
A > ~/tm/swt_h06 > git P main +1 > ✗ 1 git pull 138912 < 22:31:28 < 1.35
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Entpacke Objekte: 100% (3/3), 352 Bytes | 176.00 KiB/s, fertig.
Von git.fh-aachen.de:cortex/swt_h06
1cef230..77fc3c4 main -> origin/main
automatischer Merge von Hey_World.py
KONFLIKT (Inhalt): Merge-Konflikt in Hey_World.py
Fehler: Konnte 3b73a10... (Ausgabe hinzugefügt) nicht anwenden
Hinweis: Resolve all conflicts manually, mark them as resolved with
Hinweis: "git add/rm <conflicted_files>", then run "git rebase --continue".
Hinweis: You can instead skip this commit: run "git rebase --skip".
Hinweis: To abort and get back to the state before "git rebase", run "git rebase --abort".
Preimage für 'Hey_World.py' aufgezeichnet.
Konnte 3b73a10... (Ausgabe hinzugefügt) nicht anwenden
[1] 337124 exit 1 git pull
A > ~/tm/swt_h06 > git @77fc3c41 rebase-i 1/1 ~1 > ✗ 1 138913 < 22:31:53 < 1.30
```

Abbildung 4: Aufgabe 2, Nr. 1-4


```
^ > ~/tm/swt_h06 > git@77fc3c41 rebase-i 1/1 ~1 > x 1 cat Hey_World.py 138913 < 22:31:53 < 1.30
# Autor und Datum
<<<<<< HEAD
print(hw)
=====
print('Hello World')
>>>>>> 3b73a10 (Ausgabe hinzugefügt)
^ > ~/tm/swt_h06 > git@77fc3c41 rebase-i 1/1 ~1 > ✓ nano Hey_World.py 138914 < 22:33:10 < 1.24
^ > ~/tm/swt_h06 > git@77fc3c41 rebase-i 1/1 ~1 > 16s > ✓ cat Hey_World.py 138915 < 22:33:28 < 1.19
# Autor und Datum
hw = 'Hello World'
print(hw)
^ > ~/tm/swt_h06 > git@77fc3c41 rebase-i 1/1 ~1 > ✓ git add Hey_World.py 138916 < 22:33:32 < 1.17
^ > ~/tm/swt_h06 > git@77fc3c41 rebase-i 1/1 +1 > ✓ git status 138917 < 22:34:13 < 1.33
interaktives Rebase im Gange; auf 77fc3c4
Letzter Befehl erledigt (1 Befehl erledigt):
  pick 3b73a10 Ausgabe hinzugefügt
Keine Befehle verbleibend.
Sie sind gerade beim Rebase von Branch 'main' auf '77fc3c4'.
  (alle Konflikte behoben: führen Sie "git rebase --continue" aus)

Zum Commit vorgemerkte Änderungen:
  (benutzen Sie "git restore --staged <Datei>..." zum Entfernen aus der Staging-Area)
    geändert:    Hey_World.py
^ > ~/tm/swt_h06 > git@77fc3c41 rebase-i 1/1 +1 > ✓ git rebase --continue 138918 < 22:34:40 < 1.35
Konfliktauflösung für 'Hey_World.py' aufgezeichnet.
[losgelöster HEAD 0b063ea] Ausgabe hinzugefügt und Merge Konflikte behoben
  1 file changed, 1 insertion(+)
Erfolgreich Rebase ausgeführt und refs/heads/main aktualisiert.
^ > ~/tm/swt_h06 > git@77fc3c41 rebase-i 1/1 +1 > 16s > ✓ git push 138919 < 22:35:12 < 1.30
Objekte aufzählen: 5, fertig.
Zähle Objekte: 100% (5/5), fertig.
Delta-Kompression verwendet bis zu 8 Threads.
Komprimiere Objekte: 100% (2/2), fertig.
Schreibe Objekte: 100% (3/3), 413 Bytes | 413.00 KiB/s, fertig.
Gesamt 3 (Delta 0), Wiederverwendet 0 (Delta 0), Pack wiederverwendet 0
To git.fh-aachen.de:cortex/swt_h06.git
  77fc3c4..0b063ea main -> main
^ > ~/tm/swt_h06 > git@77fc3c41 rebase-i 1/1 +1 > ✓ | 138920 < 22:35:21 < 1.27
```

Abbildung 5: Aufgabe 2, Nr. 5-6

```
^ > ~/tm/swt_h06 > git main > ✓ git log --graph 138923 < 22:36:29 < 1.17

STDIN

1  * commit 0b063eaeelcfc130a2930c1cf1d0fdb59803e76a (HEAD -> main, origin/main, origin/HEAD)
2  | Author: Christian Rene Thelen <christian.thelen@rwth-aachen.de>
3  | Date: 2023-11-19
4  |
5  | Ausgabe hinzugefügt und Merge Konflikte behoben
6  |
7  * commit 77fc3c4141565b3cb6d9031e85264b7c1a030ca0
8  | Author: Leonie Hellwig <christian.thelen@rwth-aachen.de>
9  | Date: 2023-11-19
10 |
11 | Andere Ausgabe hinzugefügt
12 |
13 * commit 1cef2304aaa861d2db083446cde006f6125b3dcb
14 | Author: Christian Rene Thelen <christian.thelen@rwth-aachen.de>
15 | Date: 2023-11-19
16 |
17 | A1 Nr. 8-10
18 |
19 * commit 4e3b46142ae0404e7de880eab8b3df7d96c11de5
20 | Author: Christian Rene Thelen <christian.thelen@rwth-aachen.de>
21 | Date: 2023-11-19
22 |
23 | Inline-Kommentare ergänzt
24 |
25 * commit 0c71894ea06ed150db2b3201b01a4bb6f75cb858
26 | Author: Christian Rene Thelen <christian.thelen@rwth-aachen.de>
27 | Date: 2023-11-19
28 |
29 | Diese Hausaufgabe ist langweilig...
30 |
31 * commit e6797bcd87509219e1dc03b32cd23d65c72603d0
32 | Author: Christian Rene Thelen <rene.thelen@alumni.fh-aachen.de>
33 | Date: 2023-11-19
34 |
35 | Initial commit

^ > ~/tm/swt_h06 > git main > ✓ | 138924 < 22:36:32 < 1.15
```

Abbildung 6: Aufgabe 2, Nr. 7

A3 Git: Branching

Sie wollen Ihre Hello-World-Anwendung aus Aufgabe A2 um ein neues Feature zur Ausgabe von »Goodbye Moon!« ergänzen. Erzeugen Sie dazu eine neue Klasse in einer neuen Datei. Nutzen Sie den Feature Branch Workflow aus der Vorlesung und dokumentieren Sie zentrale Schritte durch Screenshots (eine Iteration des Workflows genügt). Achten Sie dabei auf eine angemessene Namensgebung, wie sie in der Vorlesung empfohlen wurde.

Lösung 3

```
A > ~/tm/swt_h06 > git P main > ✓ git branch feature-goodbye 138939 < 22:43:08 < 0.79
A > ~/tm/swt_h06 > git P main > ✓ git switch feature-goodbye 138940 < 22:43:14 < 0.78
Zu Branch 'feature-goodbye' gewechselt
A > ~/tm/swt_h06 > git P feature-goodbye > ✓ echo "print('Goodbye Moon!')" > Goodbye.py
A > ~/tm/swt_h06 > git P feature-goodbye ?1 > ✓ python Goodbye.py 138942 < 22:43:25 < 0.76
Goodbye Moon
A > ~/tm/swt_h06 > git P feature-goodbye ?1 > ✓ git add Goodbye.py 138943 < 22:43:29 < 0.76
A > ~/tm/swt_h06 > git P feature-goodbye +1 > ✓ git commit -m 'Goodbye Moon Feature implementiert'
[feature-goodbye 198f2aa] Goodbye Moon Feature implementiert
1 file changed, 1 insertion(+)
create mode 100644 Goodbye.py
A > ~/tm/swt_h06 > git P feature-goodbye > ✓ git switch main 138945 < 22:43:42 < 0.87
Zu Branch 'main' gewechselt
Ihr Branch ist auf demselben Stand wie 'origin/main'.
A > ~/tm/swt_h06 > git P main > ✓ git merge feature-goodbye 138946 < 22:43:50 < 0.84
Aktualisiere 0b063ea..198f2aa
Fast-forward
 Goodbye.py | 1 +
1 file changed, 1 insertion(+)
create mode 100644 Goodbye.py
A > ~/tm/swt_h06 > git P main +1 > ✓ git push 138947 < 22:43:57 < 0.83
Objekte aufzählen: 4, fertig.
Zähle Objekte: 100% (4/4), fertig.
Delta-Kompression verwendet bis zu 8 Threads.
Komprimiere Objekte: 100% (2/2), fertig.
Schreibe Objekte: 100% (3/3), 401 Bytes | 401.00 KiB/s, fertig.
Gesamt 3 (Delta 0), Wiederverwendet 0 (Delta 0), Pack wiederverwendet 0
To git.fh-aachen.de:cortex/swt_h06.git
0b063ea..198f2aa main -> main
A > ~/tm/swt_h06 > git P main > ✓ git branch -d feature-goodbye 138948 < 22:44:03 < 0.81
Branch feature-goodbye entfernt (war 198f2aa).
A > ~/tm/swt_h06 > git P main > ✓ 138949 < 22:44:09 < 0.80
```

Abbildung 7: Aufgabe 3

A4 Git: Änderungen rückgängig machen

Recherchieren Sie, wie Sie die folgenden drei Herausforderungen lösen können und geben Sie eine Lösung an:

- Sie haben bei Ihrem letzten Commit ins lokale Repository leider eine Datei vergessen und Sie wollen den letzten Commit um diese Datei erweitern, ohne dass aber ein ganz neuer Eintrag im Commit-Verlauf entsteht.
- Sie möchten die bereits ins lokale Repository per Commit übertragenen Änderungen wieder zurücknehmen, so dass dieser Commit ganz aus dem Commit-Verlauf gelöscht wird.
- Sie möchten den letzten Commit ins Remote-Repository zurücknehmen, so dass aber die Rücknahme selbst Teil des Commit-Verlaufs wird und für andere Entwickler nachvollziehbar bleibt.

Lösung 4

- a) Für diesen Fall gibt es `git commit --amend`.
- b) Wenn die Commits noch nicht übertragen wurden, kann dazu `git reset ...`
→ `--hard <commit-hash>` verwendet werden. Sollten die Commits bereits übertragen worden sein oder möchte man keinen harten Reset durchführen, so lässt sich mit `git revert <commit-hash>` auch die Änderungen eines Commits rückgängig machen und dies als neuen Commit speichern.
- c) `git revert HEAD~1`