Fachhochschule Nordwestschweiz
Hochschule für Architektur, Bau und Geomatik

Version Control Systems (VCS)
Mit GIT und GITHUB

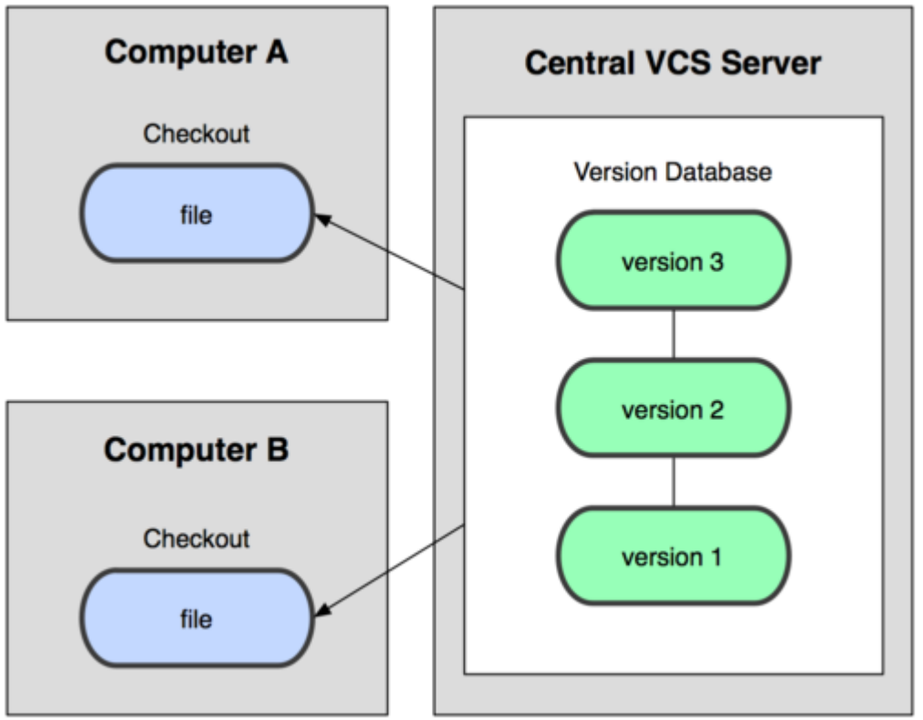
Versionskontrollsytem (Version Control System VCS)

- Ein System zur Verwaltung und Versionierung von Software oder anderer digitaler Information
- Primärzwecke
  - Kollaboration mit digitalen Inhalten
  - Nachvollziehbarkeit von Änderungen und Historie
  - Integrität gewärleisten
  - Widerspruchsfreiheit von konkurrierenden Änderungen an den gleichen Stellen

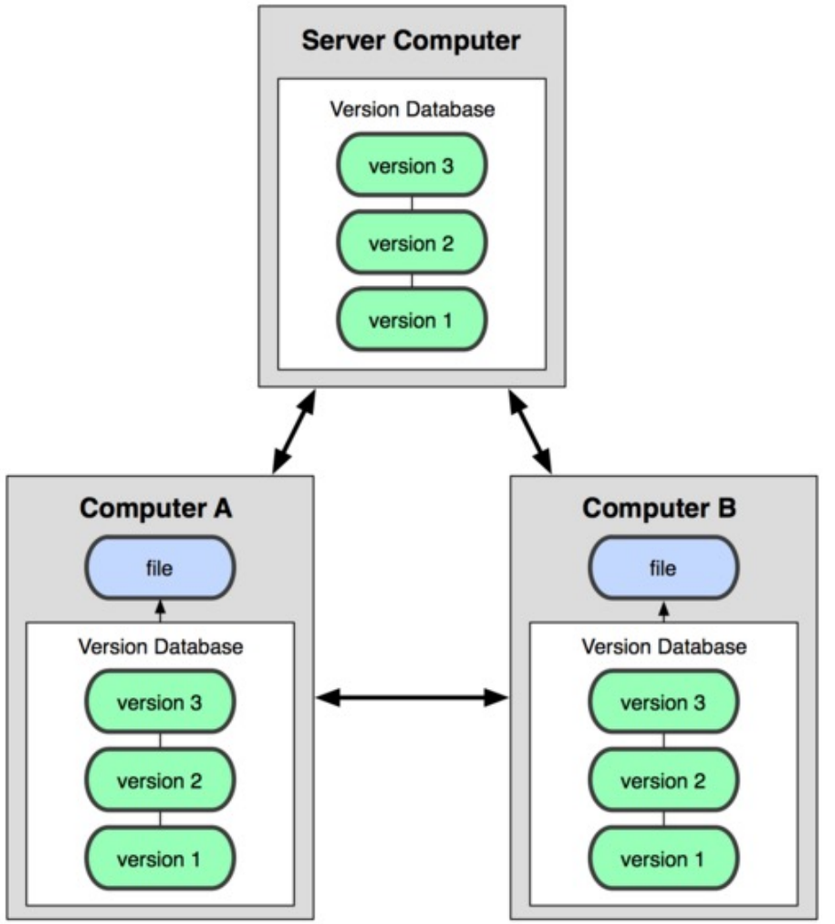
GIT

- Grundidee entwickelt von Linux Torvalds in 2005
- Die Linux Foundation suchte nach einer besseren Lösung zur Versionskontrolle für den Linux Kernel
- Primärziele: Schnelligkeit bzw. Performance und überprüfbare Integrität der verwalteten Daten.
- Besonderheit: Weg von einer zentralistischen Versionsverwaltung zu einer dezentralen Verwaltung von Kopien (Vergleichbar mit dem Prinzip der Blockchain)
- Lizenz: GNU-GPLv2-Lizenz
- Kurz: Vorgenommene Änderungen können jederzeit mit allen anderen Projektteilnehmern ausgetauscht und – sofern relevant – in das Repository aufgenommen werden.

Klassische VCS Systeme vs GIT



Klassisches VCS (SVN)



GIT

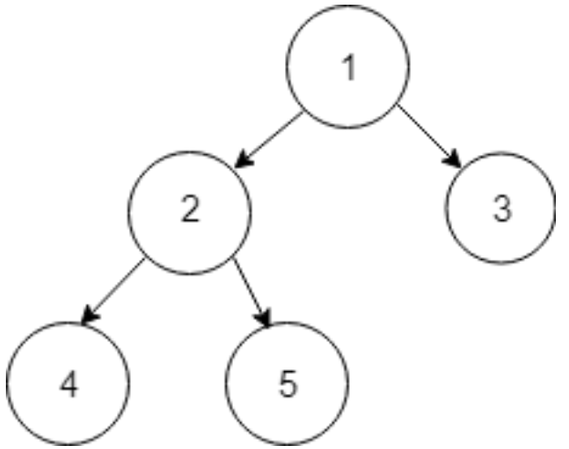
Begriffe: Repository

- Eine Ablage von digitalen Daten
- Die Historie der Änderungen an den digitalen Inhalten wird vollständig gespeichert
- Datenbankähnliches System
- Wird entweder an einem zentralen Ort, in der Cloud oder als sog. “Working Copy” auf Entwicklergeräten gefunden
- Kurzbegriff: Repo

Quelle: http://gitbu.ch/

Begriffe: Working Tree

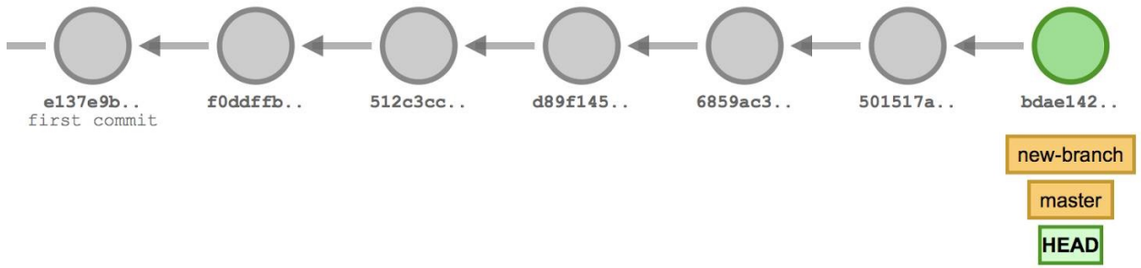
- Das Arbeitsverzeichnis von Git (in anderen Systemen manchmal auch Sandbox oder Checkout genannt).
- Modifikationen am Quellcode warden hier vorgenommen
- Oft auch Working Directory genannt



Quelle: http://gitbu.ch/

Begriffe: Commit

- Veränderungen am Working Tree, also z.B. modifizierte oder neue Dateien, werden im Repository als Commits gespeichert.
- Zustand aller verwalteten Dateien zu einem bestimmten Zeitpunkt.
- Enthält neben den Änderungen auch Metadaten:
  - Timestamp (Datum Uhrzeit)
  - Commit Message (Nachricht mit Informationen zu den Änderungen)
- Bildet immer den Zustand aller versionierten Dateien zum Timestamp “X” ab

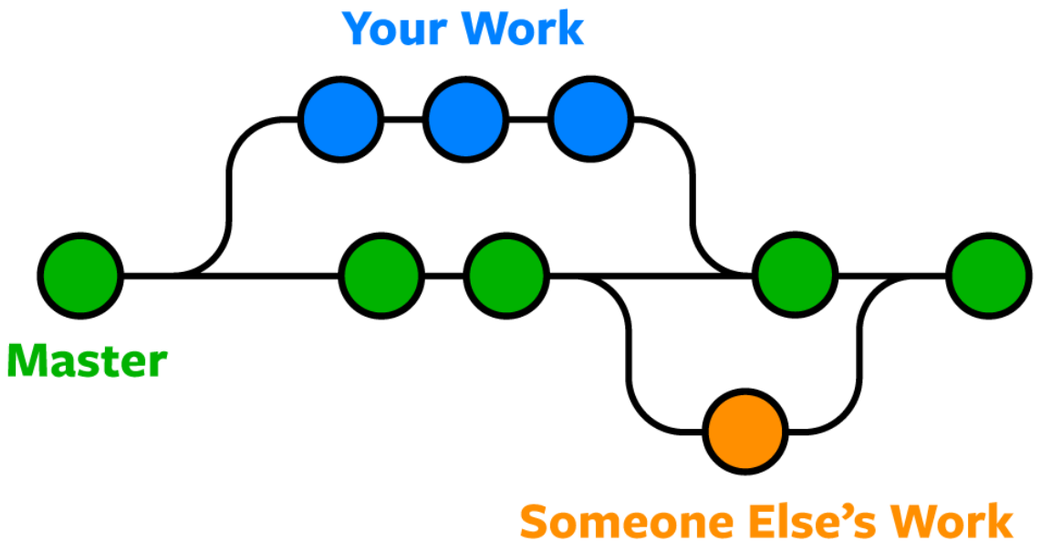


Begriffe: HEAD

- Eine symbolische Referenz auf den neuesten Commit im aktuellen Branch.
- Von dieser Referenz hängt ab, welche Dateien Sie im Working Tree zur Bearbeitung vorfinden.
- Es handelt sich also um den „Kopf“ bzw. die Spitze eines Entwicklungsstrangs

Begriffe: Branch

- Eine Abzweigung in der Entwicklung.
- Dient zur Entwicklung neuer Funktionalität oder sogenannten Features
- Ein neuer Zweig muss später dann mittels ”Merge” wieder in den Hauptzweig integriert werden



Begriffe: Clone

- Wenn man ein Repository klont (clone) dann bekommt man eine lokale Kopie diese Repositories
- Ein Clone enthält die gesamte Änderungshistorie des Repositorie
- Wird ein Git-Repositry aus dem Internet heruntergeladen, so erzeugen Sie automatisch einen Klon (Clone) dieses Repositories.

Quelle: http://gitbu.ch/

Begriffe: SHA-1

- Der Secure Hash Algorithmus
- Erstellt eine eindeutige, 160 Bit lange Prüfsumme (40 hexadezimale Zeichen)
- Kann für beliebige digitale Informationen eingesetzt werden
- In GIT warden alle Commits in Git nach ihrer SHA-1-Summe benannt (Commit-ID), die aus dem Inhalt und den Metadaten des Commits errechnet wird.
- Es ist sozusagen eine inhaltsabhängige Versionsnummer, z.B. f785b8f9ba1a1f5b707a2c83145301c807a7d661.

Quelle: http://gitbu.ch/

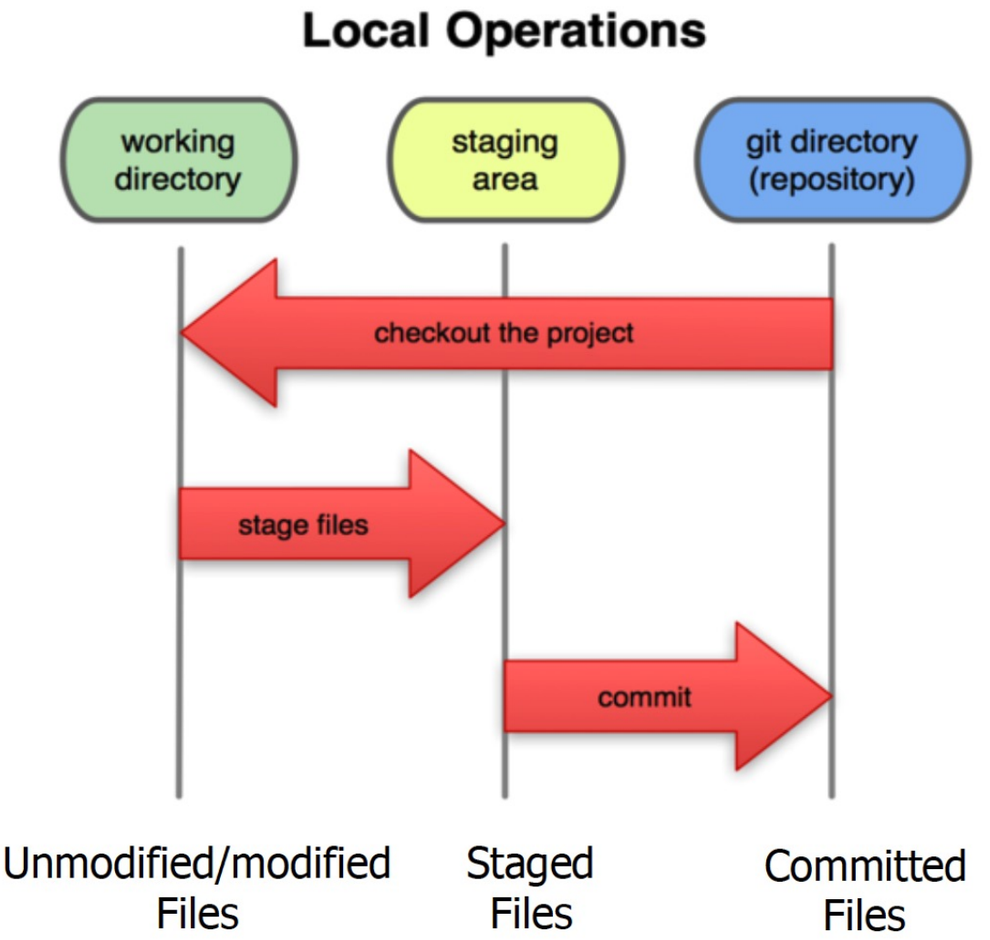
Begriffe: Tag

- Tags sind symbolische Namen für schwer zu merkende SHA-1-Summen.
- Wichtige Commits, wie z.B. Releases, können Sie mit Tags kennzeichnen.
- Ein Tag kann einfach nur ein Bezeichner, wie z.B. v1.6.2, sein, oder zusätzlich Metadaten wie Autor, Beschreibung und GPGSignatur enthalten.

Quelle: http://gitbu.ch/

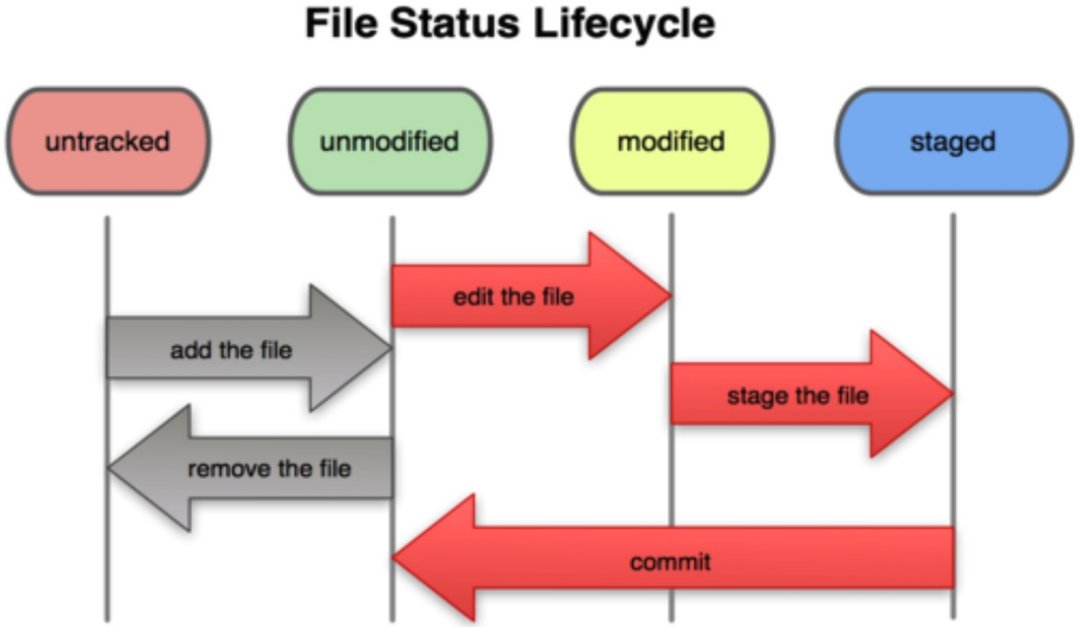
Staging

- Git verwendet ein zweistufiges System um lokale Änderungen nachzuverfolgen
- Staging bedeutet, dass Dateien vorbereitet wurden um in den lokalen Working Tree integriert zu werden. Dabei handelt es sich um Snapshots von Änderungen an mehreren Dateien
- Sobald ein “Commit” ausgelöst wird sind die Änderungen amtlich und sind teil der Historie.
- Was während der Staging Phase alles geändert wurde und beim Commit nicht mehr präsent ist wird verworfen. Nur das Delta zum letzen Commit bleibt bestehen



Staging

- **Modified**: Lokal geänderte Dateien
- **Staged**: Snapshot von gemachten Änderungen (Nur lokal verfügbar)
- **Commit**: Übertragene definitive Änderungen die teil der Historie geworden sind



GIT Installieren

Installation unter Windows

- Git kann für Windows heruntergeladen werden: https://git-scm.com/
- Auch nützlich ist GitHub Desktop Client für git. Dieser kann für Windows und Mac hier heruntergeladen werden: https://desktop.github.com/

Installation unter MacOS X

- Für MacOS gibt man im Terminal git –version ein. Falls es noch nicht installiert ist, erscheint eine Abfrage wie man es installieren will.

Die Installation erfolgt mit allen Standard-Einstellungen.

GIT konfigurieren

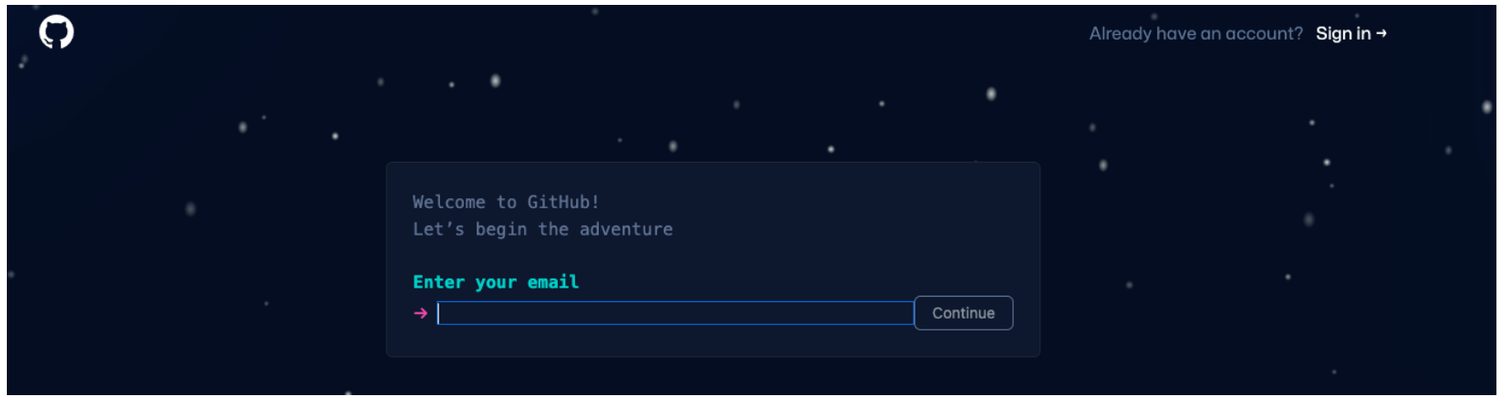
- Nach der Installation müssen wir einmalig eine Konfiguration Durchführen. Dazu öffnen wir die «Git Bash» (oder im Terminal unter MacOS). Es gibt einige Optionen, aber die wichtigsten sind Benutzername und E-Mail.

```
git config --global user.name "Vorname Nachname"
```

```
git config --global user.email name@email.com
```

Github Registrierung

- https://github.com/signup

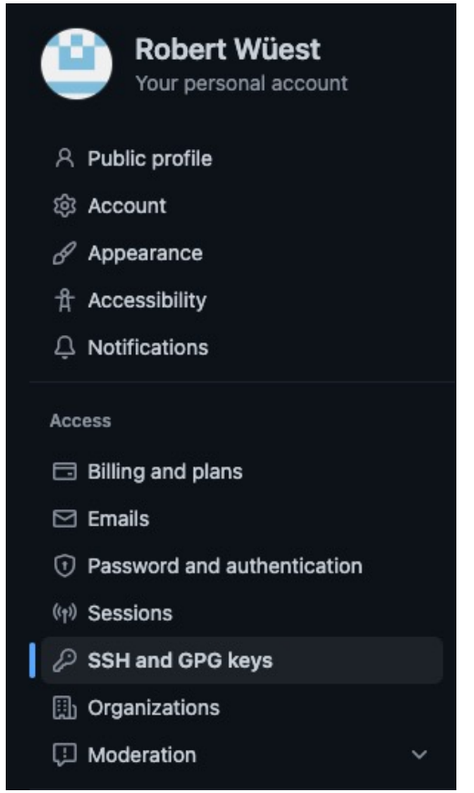


SSH Key erstellen und auf Github einrichten

- In der Git Bash (oder Konsole) mit ssh-keygen einen neuen Schlüssel erzeugen:

```
ssh-keygen
```

- Es wird ein Pfad und Dateiname für den Schlüssel vorgeschlagen, wir belassen das.
- Eine Passphrase wird bei Versionssverwaltung in der Regel nicht verwendet, wir wollen mit dem Schlüssel eine sichere Verbindung um genau das Passwort nicht immer einzuippen müssen.
- Im .ssh Directory befinden sich nun der Private und der öffentliche Schlüssel. Den öffentlichen Schlüssel kopieren wir auf GitHub. Dazu gehen wir auf «Settings» (Klick auf das Profilbild oben rechts). Wir wählen das Menu auf der linken Seite «SSH and GPG keys) und fügen den key mit «New ssh key» hinzu.



Wichtige Befehle

command	description
`git clone` ***`url [dir]`***	copy a Git repository so you can add to it
`git add` ***`file`***	adds file contents to the staging area
`git commit`	records a snapshot of the staging area
`git status`	view the status of your files in the working directory and staging area
`git diff`	shows diff of what is staged and what is modified but unstaged
`git help` ***`[command]`***	get help info about a particular command
`git pull`	fetch from a remote repo and try to merge into the current branch
`git push`	push your new branches and data to a remote repository
others: `init, reset, branch, checkout, merge, log, tag`	

Befehle: git status

Zu den wichtigsten Git-Grundlagen zählt eine gute Organisation des eigenen Arbeitsverzeichnisses. Über dieses schlagen Sie nicht nur eigene Änderungen und Neuerungen an einem Projekt vor, die anschliessend per Commit („Freischaltung“) übernommen werden, sondern beziehen auch Informationen über die Aktivitäten anderer Nutzer. Die Aktualität Ihrer Arbeitskopie können

Sie über folgende Eingabe überprüfen: `git status`

Bei einem gerade neu angelegten Repository bzw. einer absoluten Übereinstimmung von Hauptverzeichnis und Arbeitskopie erhalten Sie in der Regel die Information, dass es keinerlei neue Anpassungen an dem Projekt gab („No commits yet“). Zudem teilt Git mit, dass Sie aktuell keine eigenen Änderungen für den nächsten Commit geteilt haben („nothing to commit“). Um eine eigene, neue Datei zur Versionsverwaltung hinzuzufügen oder eine überarbeitet.

Befehle: git add

Um eine eigene, neue Datei zur Versionsverwaltung hinzuzufügen oder eine überarbeitete Datei für den nächsten Commit anzumelden, wenden Sie den Befehl „git add“ auf diese Datei an (diese muss sich hierfür im Arbeitsverzeichnis befinden).

```
git add Test.txt
```

Befehle: git commit

Sämtliche Änderungen, die Sie (wie im vorherigen Abschnitt beschrieben) für die Versionsverwaltung angemeldet haben, müssen immer per Commit bestätigt werden, damit sie in den HEAD aufgenommen werden. Beim HEAD handelt es sich um eine Art Index, der auf den letzten wirksam gewordenen Commit in Ihrer aktuellen Git-Arbeitsumgebung (auch als „Branch“ bezeichnet) verweist.

Das Kommando für diesen Schritt lautet:

```
git commit -m "Beschreibung der Änderung"
```

Sie erhalten nach der Ausführung von „git commit“ eine zusammenfassende Meldung zu dem Commit: In den voranstehenden eckigen Klammern steht zum einen der Name der Branch (Projektzweig; hier „master“, da unser Arbeitsverzeichnis gleichzeitig auch das Hauptverzeichnis ist), in den die Änderungen übertragen wurden, zum anderen die SHA-1-Prüfsumme des Commits (z.B. „c0fdf55“). Es folgen der frei gewählte Kommentar (hier „Test“) und konkrete Informationen über die vorgenommenen Anpassungen.

Befehle: git branch / git checkout

Einen neuen Branch zu erstellen, ist nicht schwer: Sie benötigen lediglich die Anweisung „git branch“ und hängen dieser die gewünschte Bezeichnung für den Zweig an. Einen Beispiel-Branch mit dem Namen „test_branch“ erstellen Sie beispielsweise auf folgende Weise:

```
git branch test_branch
```

Anschliessend können Sie jederzeit mit der Anweisung „git checkout“ in diesen Branch wechseln:

```
git checkout test_branch
```