BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

Programmieren 1
11 – Versionsverwaltung mit Git
- Grundlagen -

Bachelor Medieninformatik
Wintersemester 2015/2016

Dipl.-Inform. Ilse Schmiedecke
schmiedecke@beuth-hochschule.de

... gestern lief es noch!!!

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

- neues Feature eingebaut
- die ganze Nacht am Code entwickelt

- **und plötzlich läuft gar nichts mehr!**

Versionskontrolle 1: Zurück zum lauffähigen Stand

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

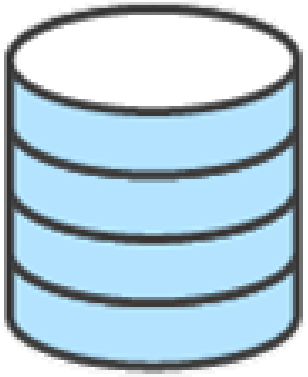
- Zurück zum lauffähigen Stand, aber wie?
  - Dateien wurden verändert
  - manche mehrfach
  - manche gar nicht
  - manche auf dem Laptop, manche auf dem Desktoprechner
  - Unübersichtliche Lage!

- Tricks:
  - Nichts ändern oder löschen, alte Version auskommentieren und ersetzen (interner Backup per Datei)
  - Backup der letzten lauffähigen Version als Kopie (Backup per Version)
  - Hilfsmittel mit Historie, z.B. Dropbox (externer Backup per Datei)
  - Leider auch nicht übersichtlich...

Versionsverwaltung mit einem "Repository"

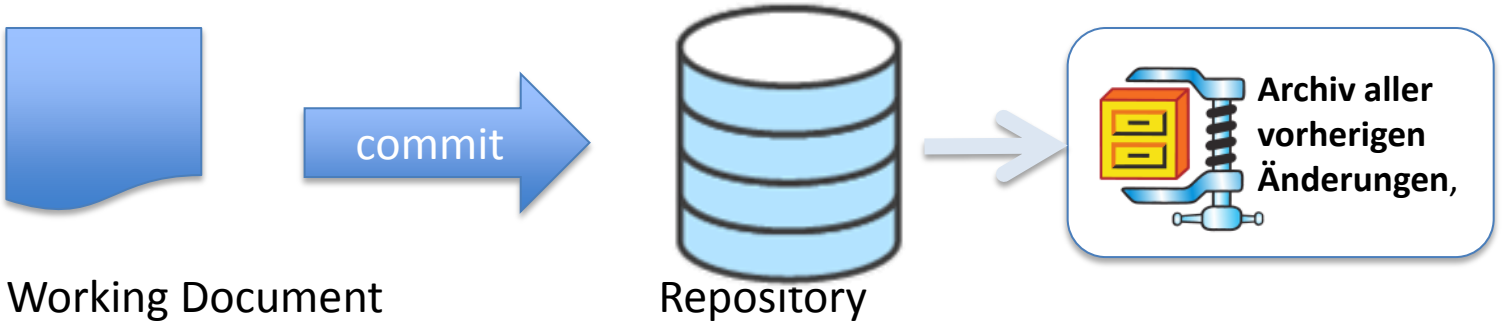
BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

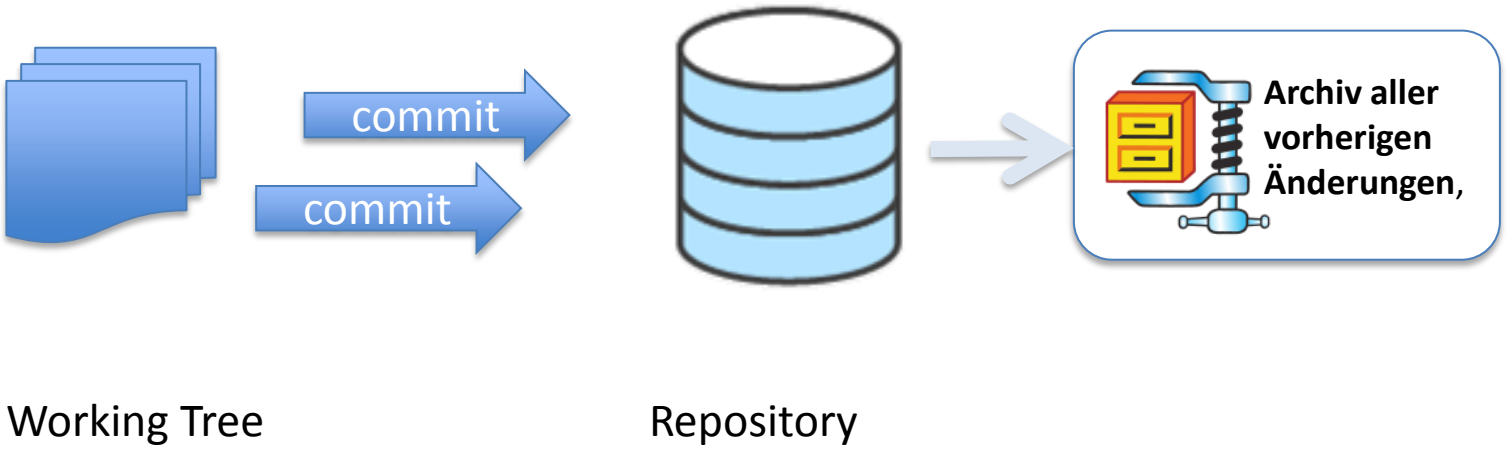
- Die Anwort heißt:
- **Versionsverwaltung**
  - alle Zwischenstände im "Repository" aufbewahren
  - durch geeignete Versionsnummer identifizieren
  - durch ein besonderes Werkzeug verwalten lassen
  - z.B. CVS, Subversion(SVN), Mercurial – und **Git**



- Der neueste Stand ist immer eindeutig erkennbar
- Jeder alte Stand ist wiederherstellbar
- Änderungen können analysiert werden

Strukturiertes Backup im Repository

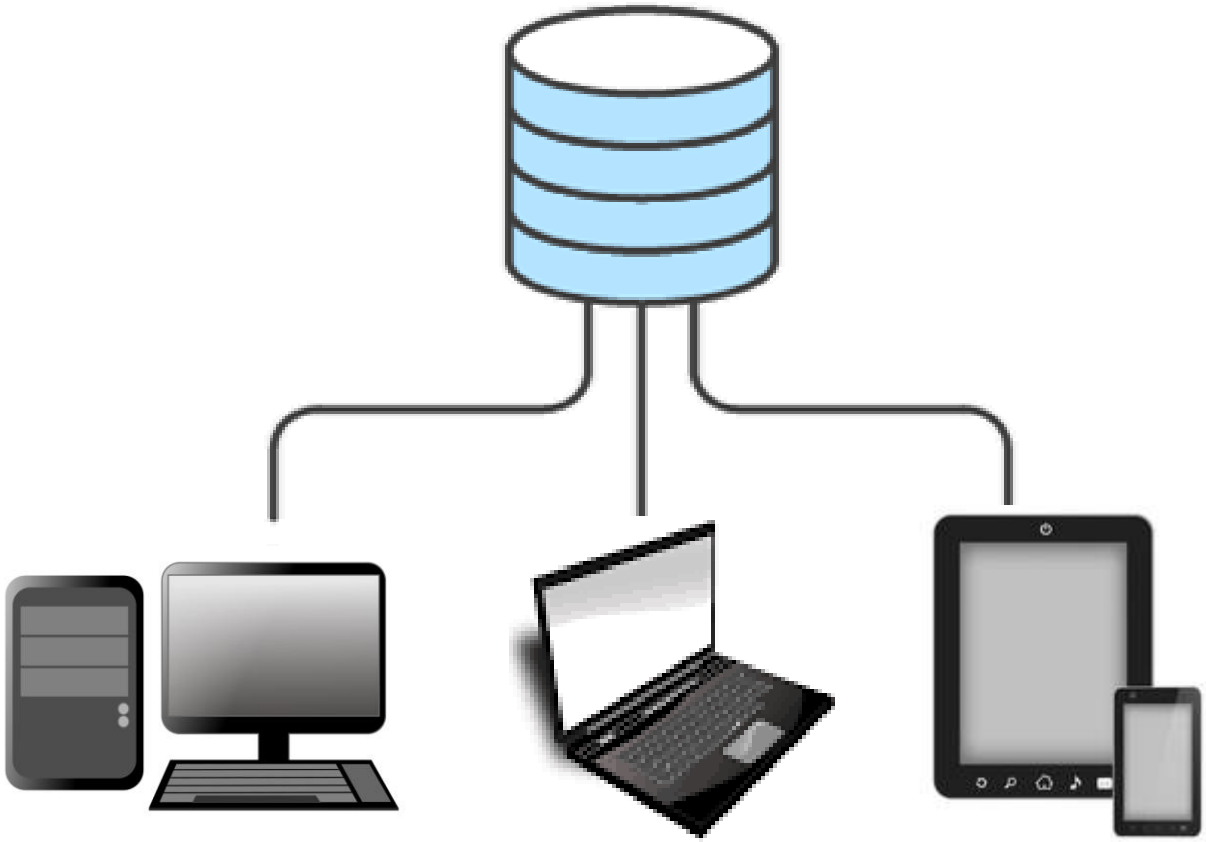




- **Das REPO enthält die gesamte Entwicklungsgeschichte**

Lokales oder Cloud-Repository?

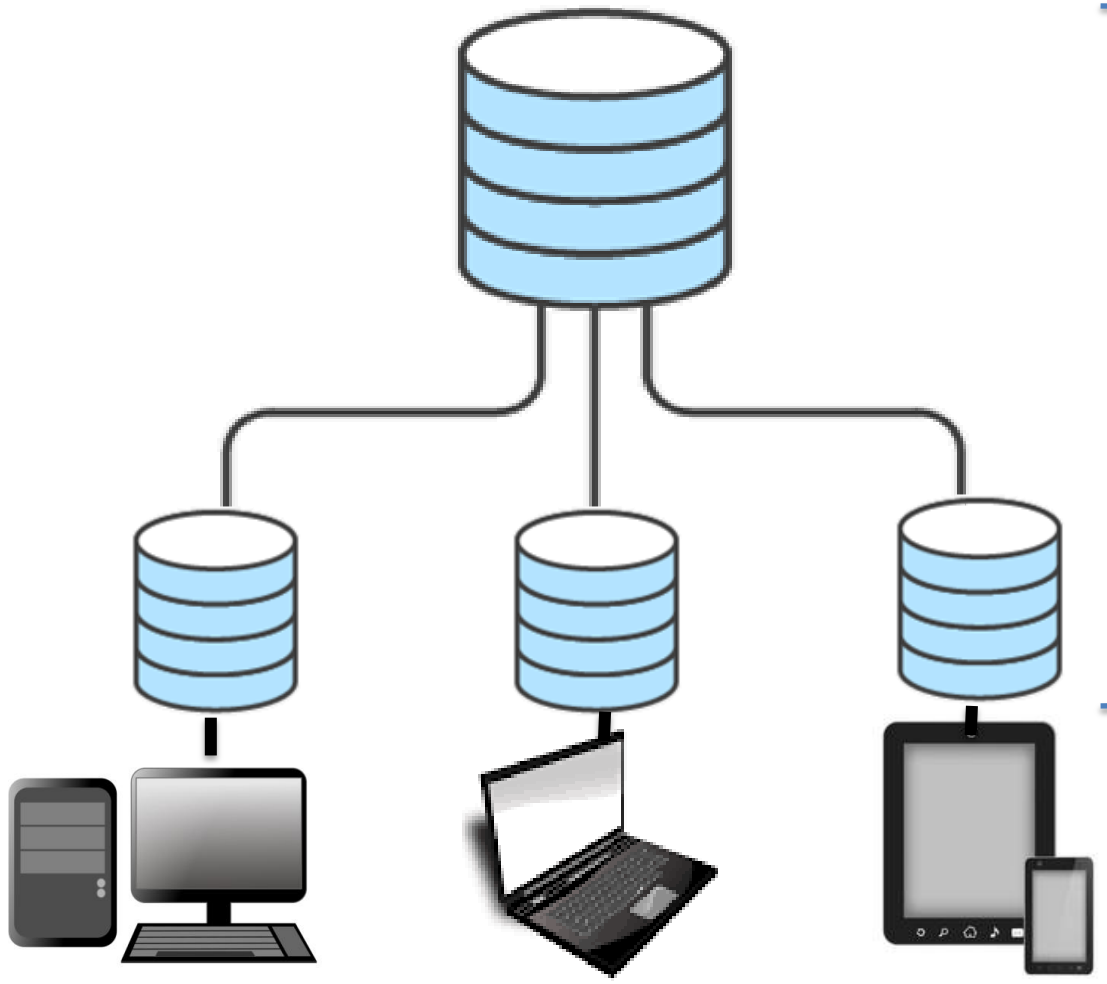
BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences



- Synchronisation ist wichtig → Cloud-Repository

Git: Lokales UND Remote Repository (Cloud-Repository)

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

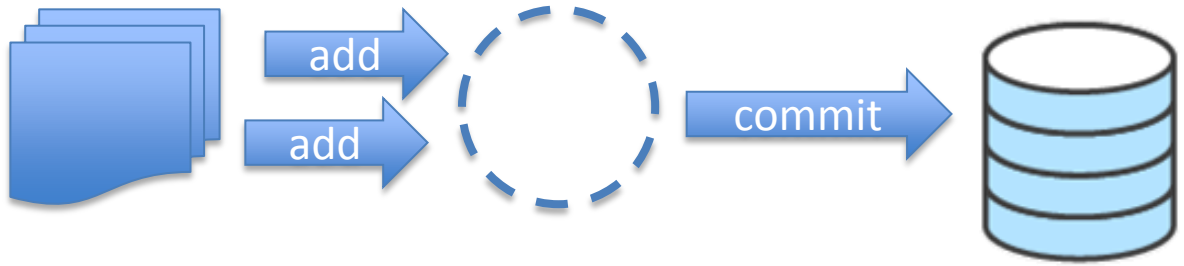


sichere lokale Arbeits-umgebung pro Rechner

sichere synchrone Arbeits-umgebung für alle Rechner

Arbeiten mit dem lokalen Git-Repository

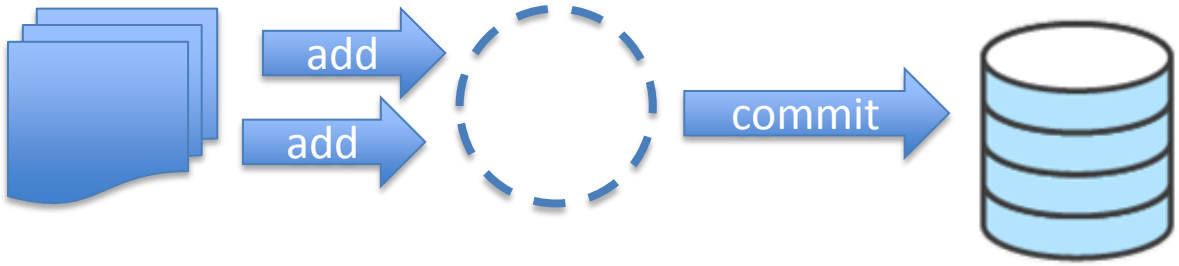
BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences



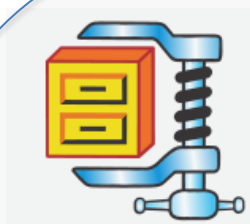
- Commit-Objekt ist immer ein ganzer WorkingTree
- Veränderte Objekte werden zunächst in der Stage zusammengefasst
- Commit kann eine sinnvolle Einheit von mehreren Dateien sein

Strukturiertes Backup im lokalen Repository

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences



Working Tree Stage (local) Repository



Commit-Objekt:
klein, binär, benannt
nicht veränderbar,
mit Kommentar,

Anlegen und Benutzen eines lokalen Repositorys

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

- git installieren (nur "Git for Windows")
- Arbeiten in der Git Bash – Konsole
  - später auch mit grafischen Clients, Sourcetree, TortoiseGit, ...

- Schritte
  - ins Projektverzeichnis wechseln (enthält Working Tree)
  - dort eine leere Datei README und eine Datei **.gitignore** erstellen, die Dateien benennt, die nicht unter die Versionsverwaltung fallen sollen, z.B. *.class

> **git init** dort ein **lokales Repo** erstellen
> **git add .** Dateien im akt. Verzeichnis der Stage (Index) hinzufügen
> **git commit -m "README und .gitignore hizugefügt"** ins Repository einspielen
> **git status** den aktuellen Stand einsehen

Nochmal kurz: Lokales Repo einrichten

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

- In der Git Bash – Kommandozeile:
- Leeres Verzeichnis wählen (oder Verzeichnis, das den Working Tree bereits enthält)
- Git-Repository erzeugen:
  - git init
- Leere Datei README erzeugen
  - touch README
- README ins Repository eintragen:
  - git add README
  - git commit -m "erstes Commit: README-Datei'

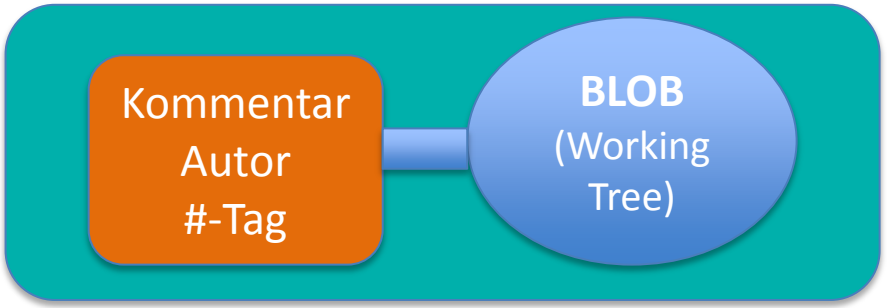
.gitignore unter der Lupe

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

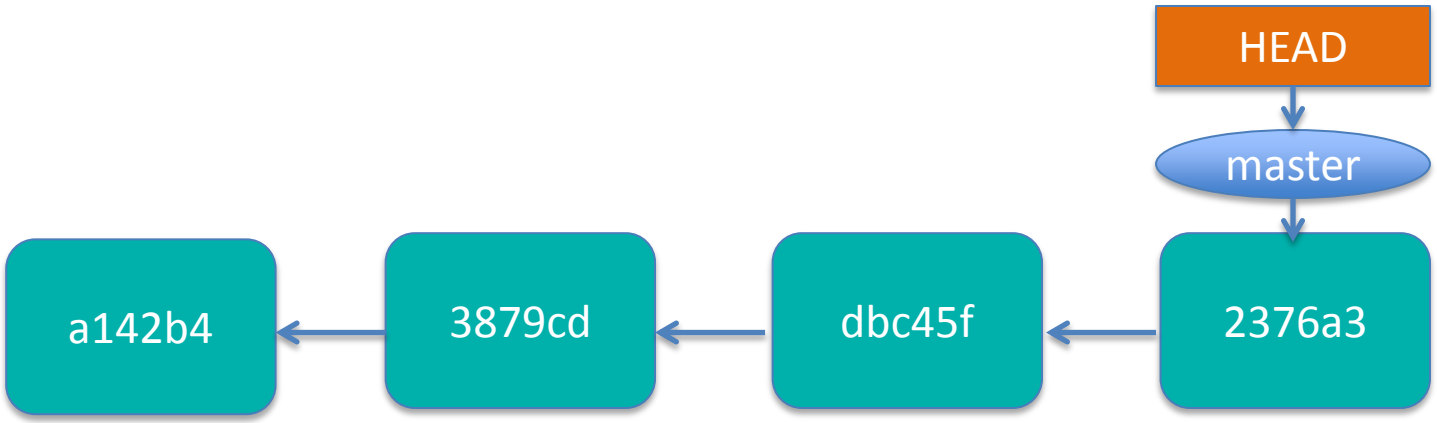
- Nicht alle Dateien im Working Tree müssen ins Repo
  - z.B. .class-Dateien nicht
  - oder das gesamte bin-Verzeichnis

- .gitignore ist eine Datei im Wurzelverzeichnis des WorkingTree
  - enthält Dateien und Muster, die nicht ins Repo sollen
  - z.B. *.class
  - jeweils auf einer neuen Zeile
  - .gitignore committen → beim commit und bei push werden die angegebenen Dateien ignoriert.

Repository-Struktur: Kette von Commit-Objekten



Commit-Objekt:
- klein, binär, unveränderlich
- benannt mit Hash-Tag

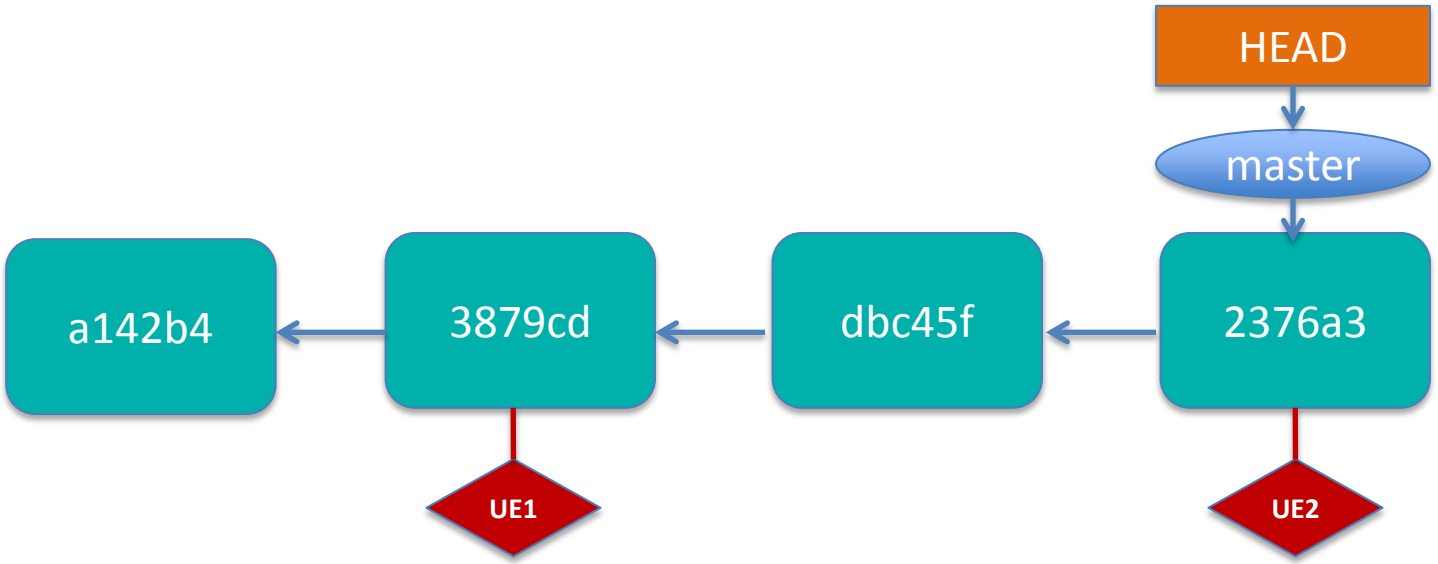


aktuelles Commit-Ziel

Branch "master"

zurück nach 3879cd:
- aktuellen Stand sichern: git **commit**
- alten Stand herstellen: git **checkout 3879cd**

Benannte Versionen: Tags



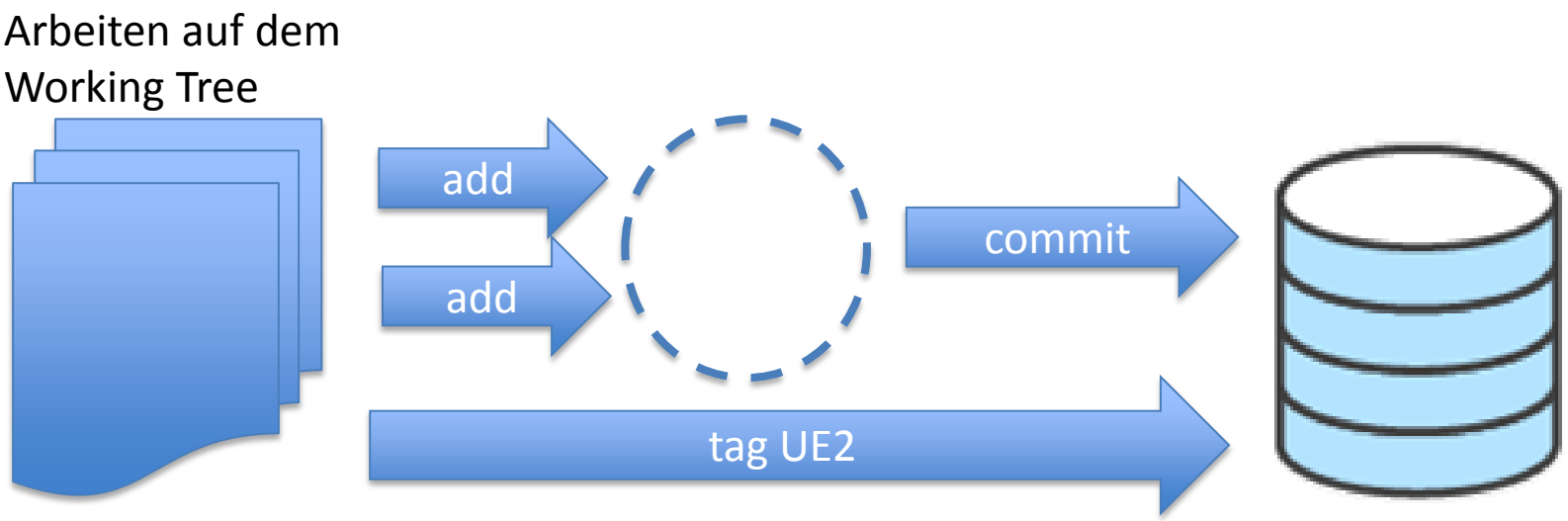
- alte Version herstellen: git **checkout 3879cd**

statt 3879cd lieber ein sinnvoller Name für die Version:

- tag definieren: git **tag UE2 –m 'Abgabe Uebung 2'**
- tag definieren: git **tag UE1 –m 'Abgabe Uebung 1' 3879cd**
- UE1 herstellen: git **checkout UE1**

Normaler Workflow mit lokalem Repository

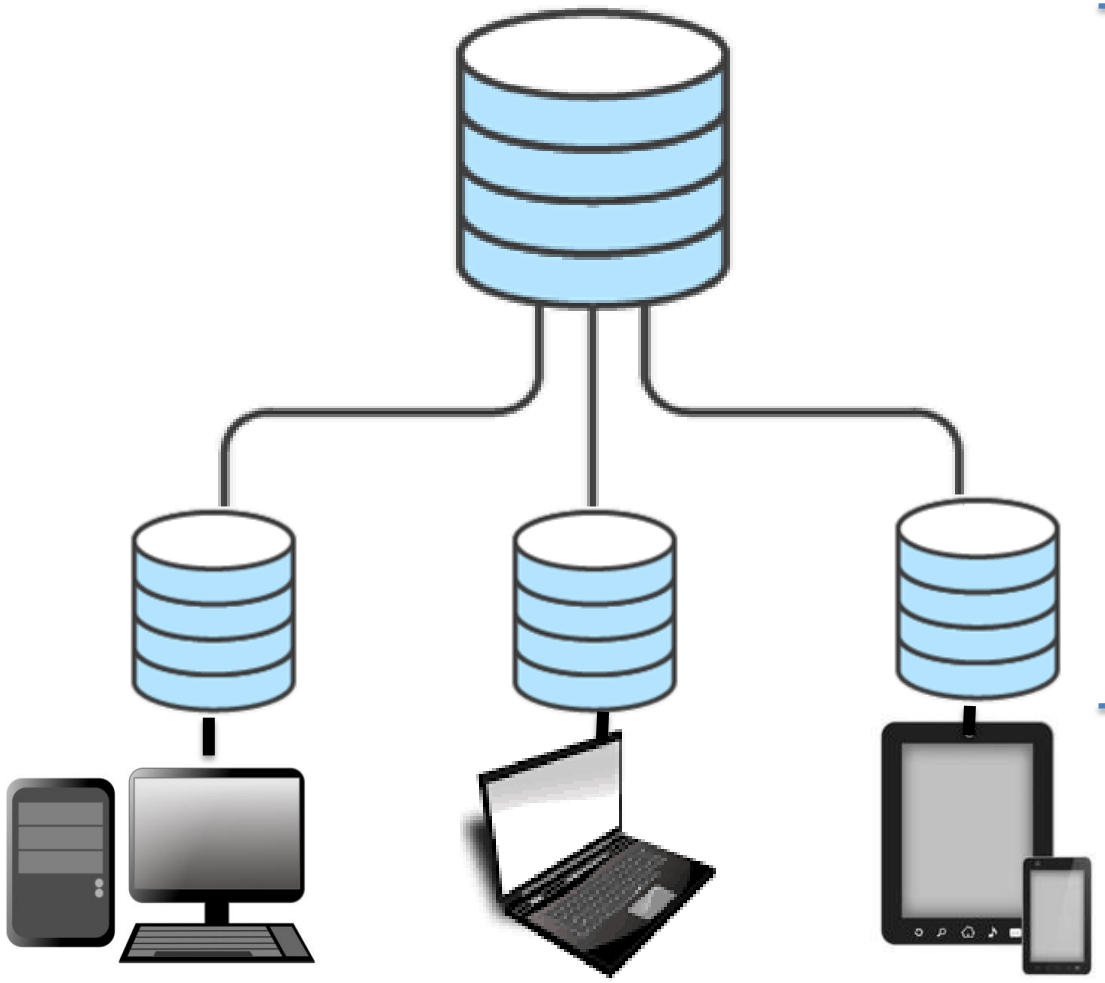
BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences



- Zusammenfassung:
  - Arbeiten wie gewohnt auf dem Working Tree
  - "Staging" von geänderten Dateien mit **add**
    Das ganze aktuelle Verzeichnis: git add .
  - Übertragen ins Repository mit **commit**
  - ggf. Tag setzen mit **tag**
  - Überschreiben des letzten commit mit **commit --amend** (Korrektur)

Git: Remote Repository (Cloud-Repository)

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences



sichere synchrone Arbeits-umgebung für alle Rechner

Remote Repository auf einem Server einrichten

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

- **Account erstellen**
  - z.B. gitlab.beuth-hochschule.de, bitbucket, github
  - Projekt erstellen
    Empfehlung: für Übungen immer private Projekte

- **Protokoll wählen**
  - **http** erfordert bei jeder Übertragung die Eingabe des Account-Passworts
  - **ssh** authentifiziert mit einem Schlüsselpaar
    - Schlüssel muss generiert (ssh-keygen) und eingetragen werden
    - Schlüsselpaar gilt für den Account, nicht nur das Projekt
    - Tipp: ohne Passphrase, damit die Authentifizierung vollautomatisch erfolgen kann.

Local und Remote verbinden

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

- Aus dem ersten lokalen git-Projekt
  - Projektadresse (mit entspr. Protokoll) aus der Serverseite kopieren
  - git remote add origin <projektadresse>
  - git push origin master // Passwortabfrage bei http
  - git clone <projektadresse>

- Weiteres lokales Repository erzeugen
  - Im lokalen Verzeichnis, das das Repository enthalten soll:
  - git clone <projektadresse>
  - git pull // Passwortabfrage bei http

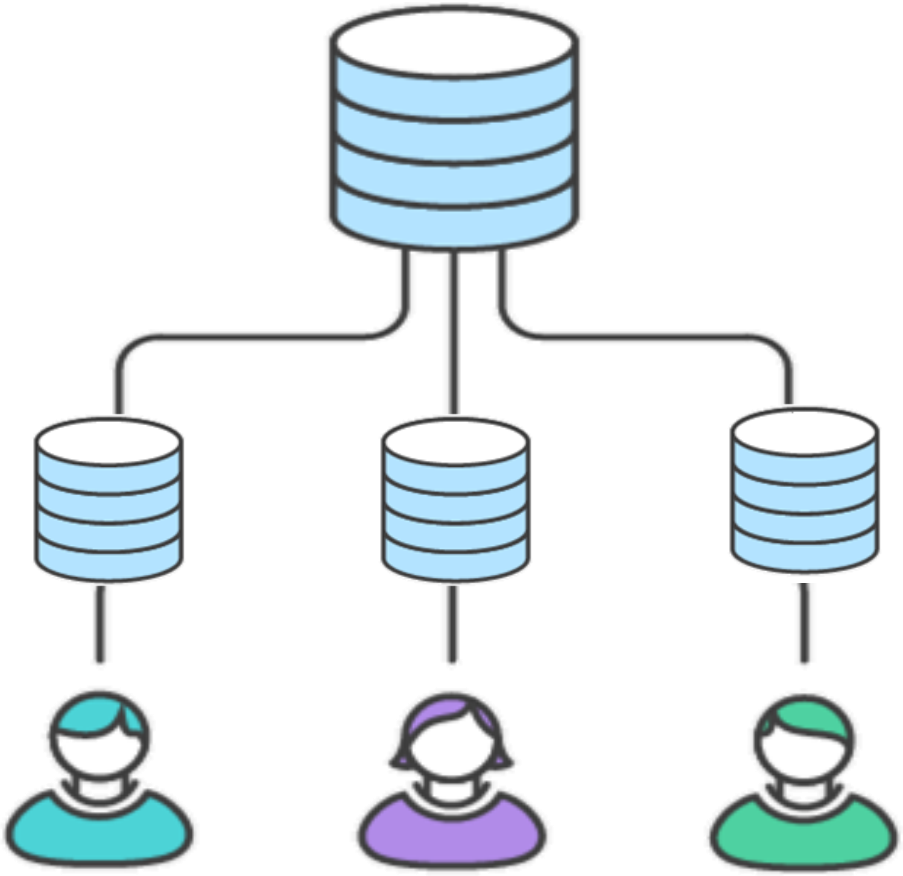
Der normale Workflow

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

- **Arbeitsbeginn auf dem Laptop:**
  - Zu Beginn **git pull**
  - Lokal arbeiten
  - Neue und geänderte Dateien mit **add** "tracken"
  - Jede Arbeitseinheit einspielen: **commit** (mit Kommentar!)
  - Für in sich geschlossene Lösungen **Tags** vergeben
  - Nach jedem neuen Ergebnis/Tag und bei Arbeitsende **push**

- **Arbeitsbeginn auf dem Desktop-Rechner**
  - **genauso!**

Arbeiten im Team



- Das gleiche Bild,
- nur dass die Rechner verschiedenen Leuten gehören...

Kommando zurück

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

- Ein beliebiges Commit ansehen
  - d.h. in den Working Tree kopieren, aber HEAD nicht ändern:
  - **git checkout 4d354** // oder
  - **git checkout myTag**

- Ein beliebiges Commit zurückbauen
  - d.h. die entspr. Änderungen aus der aktuellen Version entfernen
  - es entsteht ein neues(!) rückgebautes Commit
  - **git revert 4d354**

- Das letzte Commit ungeschehen machen (riskant!)
  - d. h. das letzte Commit wird vollständig gelöscht!
  - **git reset**

Erste Hilfe, wenn ...

- ... mein Recher aufgibt:
  - **git clone aus dem Remote Repository**
- ... der git-Server zu teuer wird
  - **git clone aus dem lokalen Repository**
- ... ich von einem fremden Rechner aus arbeiten muss
  - **git clone aus dem Remote Repository**
- *Tipp: In diesen Fällen empfiehlt sich die zwischenzeitliche Umstellung des Protokolls auf HTTP*

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

... die Praxis gibt es in den Tutorien!

Wer einen Laptop besitzt, sollte ihn mitbringen, damit er konfiguriert wird