Verteilte Versionskontrolle mit Git

Generelles

Versionskontrollsysteme (VCS):

- Protokollieren Änderungen von Dateien über die Zeit hinweg
- Bieten die Möglichkeit auf ältere Versionen zurückzugreifen
- Helfen beim Nachvollziehen welche Änderung wann & durch welche Person vorgenommen wurde (Kann bei Fehlersuche helfen)

Lokale Versionskontrollsysteme

- Speichert den Änderungsverlauf in eine separate lokale Datei
- Probleme:
    - Umständlich für Zusammenarbeiten
    - Kann nicht zur Verwaltung ganzer Projekte genutzt werden
    - Ohne Backup besteht ein „Single Point of Failure“

Zentralisierte Versionskontrollesysteme (CVCS)

- Zentraler Server wird benötigt
    - Verwaltet alle versionierten Dateien
    - Viele Clients können die Dateien, zum Bearbeiten, vom Server *abholen* (Checkout)
    - Nach Bearbeitung werden die Änderungen wieder beim Server *abgeliefert* (Checkin)
    - Leicht zu administrieren
- Probleme
    - Wenn Server nicht erreichbar ist, kann niemand mit anderen arbeiten oder neue Versionen speichern
    - „Single Point of Failure“

Verteilte Versionskontrollsysteme (DVCS)

- Jeder erhält eine vollständige Kopie des Repositories (Redundanz)
- Jedes Repository kann mit jedem anderem synchronisiert werden
- Dateien können von mehreren Leuten gleichzeitig bearbeitet werden
- Arbeiten auch ohne Verbindung zum Server möglich
- Höhere Geschwindigkeit, da alle Dateien lokal vorhanden sind

Was ist Git?

- Verteiltes Versionskontrollsystem (Open Source (github.com/git/git))
- Wurde von Linus Torvalds Aus Notwendigkeit, zur Entwicklung des Linux-Kernels initiiert
- Ziele:
    - Hohe Geschwindigkeit → Alle Dateien lokal vorhanden
    - Einfaches Design
    - Vollständig verteilt
    - Fähigkeit große Projekte zu verwalten (beispielsweise den Linux-Kernel)
    - Sicherheit gegen Verfälschung (Checksummen: SHA-1) https://shattered.io/

Repository erstellen

- `git init [project-name]`
  - Initiiert ein lokales Repository mit gegebenem Namen
- `git clone [url]`
  - Lädt ein Repository von der gegebenen URL herunter

Änderungen vornehmen

- `git status`
  - Listet alle neuen oder bearbeitete Dateien auf
- `git diff`
  - Zeigt Änderungen der nicht-staged Dateien
- `git add *`
  - Stages alle geänderten Dateien
- `git reset [file]`
  - Setzt staged Dateien auf nicht-staged, aber behält deren Änderungen im Arbeitsverzeichnis bei
- `git commit -m "[commit message]"`
  - Alle staged Dateien werden mit der angegebenen *commit message* commited
- `git push`
  - Lädt lokale Änderungen in das ursprüngliche Repository
- `git pull`
  - Lädt die alle neuen Änderungen aus dem ursprünglichen Repository

Branching

- `git branch`
  - Listet alle lokalen Branches des aktuellen Repositorys auf
- `git branch [branch-name]`
  - Erstellt neuen Branch mit *branch-name* als Name
- `git checkout [branch-name]`
  - Wechselt zu dem angegebenen Branch
- `git merge [branch-name]`
  - Kombiniert den aktuellen Branch mit dem angegebenen
- `git branch -d [branch-name]`
  - Löscht den angegebenen Branch

Commits zurücksetzen

- `git reset [commit]`
  - Macht alle Commits nach dem gegebenen Commit rückgängig

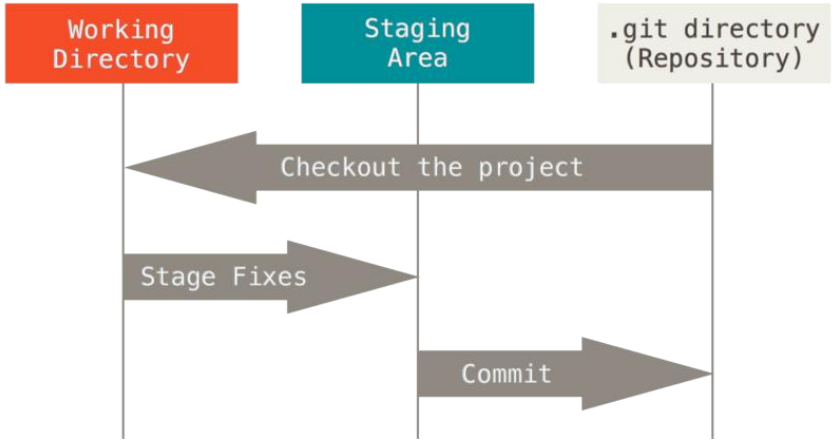

Abbildung 1 Die drei Zustände
(https://git-scm.com/book/en/v2/images/areas.png)

VCS / Kriterium	VCS-Typen	OS	Server nötig	Kolla-boration möglich	Open-Source	Check-summe
RCS	Lokales VCS	?	✘	✘	✔	✘
Subversion	CVCS	Win/Mac/ Linux/...	✔	✔	✔	✔
Perforce	CVCS	Win/Mac/ Linux/...	?	✔	✘	✔ (MD5)
Git	**DVCS**	**Win/Mac/ Linux**	✘	✔	✔	✔ **(SHA-1)**
Bazaar	DVCS	Win/Mac/ Linux/...	✘	✔	✔	✔

www.git-scm.com

GitHub-Repository