Git installieren

GitHub bietet Desktop-Clients an, die eine grafische Benutzeroberfläche für die häufigsten Aktionen auf Repositories beinhalten, sowie eine automatisch aktualisierte Kommandozeilen-Version von Git für erweiterte Szenarien.

GitHub für Windows

windows.github.com

GitHub für Mac

mac.github.com

Git-Distributionen für Linux- und POSIX-Systeme sind auf der offiziellen Git SCM-Webseite verfügbar.

Git für alle Plattformen

git-scm.com

Werkzeuge konfigurieren

Konfigurieren von Benutzerinformationen für alle lokalen Repositories

```
$ git config --global user.name "[name]"
```

Setzt den Benutzernamen, den du an deine Commit-Transaktionen hängen willst

```
$ git config --global user.email "[email address]"
```

Setzt die Emailadresse, die du an deine Commit-Transaktionen hängen willst

Repositories anlegen

Ein neues Repository anlegen, oder eines von einer bestehenden URL herunterladen

```
$ git init [project-name]
```

Legt ein neues lokales Repository mit dem angegebenen Namen an

```
$ git clone [url]
```

Klont ein Projekt und lädt seine gesamte Versionshistorie herunter

Änderungen vornehmen

Änderungen überprüfen und eine Commit-Transaktion anfertigen

```
$ git status
```

Listet alle zum Commit bereiten neuen oder geänderten Dateien auf

```
$ git diff
```

Zeigt noch nicht indizierte Dateiänderungen an

```
$ git add [file]
```

Indiziert den derzeitigen Stand der Datei für die Versionierung

```
$ git diff --staged
```

Zeigt die Unterschiede zwischen dem Index (“staging area”) und der aktuellen Dateiversion

```
$ git reset [file]
```

Nimmt die Datei vom Index, erhält jedoch ihren Inhalt

```
$ git commit -m"[descriptive message]"
```

Nimmt alle derzeit indizierten Dateien permanent in die Versionshistorie auf

Änderungen gruppieren

Benennen von Commit-Serien und Zusammenfassen erledigter Tasks

```
$ git branch
```

Listet alle lokalen Branches im aktuellen Repository auf

```
$ git branch [branch-name]
```

Erzeugt einen neuen Branch

```
$ git switch -c [branch-name]
```

Wechselt auf den angegebenen Branch und aktualisiert das Arbeitsverzeichnis

```
$ git merge [branch-name]
```

Fasst die Historie des angegeben Branches mit der des aktuell ausgecheckten Branches zusammen

```
$ git branch -d [branch-name]
```

Löscht den angegebenen Branch

Dateinamen refaktorisieren

Verschieben und Löschen versionierter Dateien

```
$ git rm [file]
```

Löscht die Datei aus dem Arbeitsverzeichnis und wird im Index zur Löschung markiert, dadurch wird die Datei beim nächsten commit aus der Versionskontrolle gelöscht

```
$ git rm --cached [file]
```

Entfernt die Datei aus der Versionskontrolle, behält sie jedoch lokal

```
$ git mv [file-original] [file-renamed]
```

Ändert den Namen der Datei und bereitet diese für den Commit vor

Tracking unterdrücken

Temporäre Dateien und Pfade ausschließen

```
*.log
build/
temp-*
```

Eine Textdatei namens `.gitignore` verhindert das versehentliche Committen von Dateien und Pfaden mit den spezifizierten Patterns

```
$ git ls-files --others --ignored --exclude-standard
```

Listet alle innerhalb dieses Projekts ignorierten Dateien auf

Fragmente speichern

Aufschieben und Wiederherstellen unvollständiger Änderungen

```
$ git stash
```

Speichert temporär alle getrackten Dateien mit Änderungen

```
$ git stash pop
```

Stellt die zuletzt zwischengespeicherten Dateien wieder her

```
$ git stash list
```

Listet alle zwischengespeicherten Änderungen auf

```
$ git stash drop
```

Verwirft die zuletzt zwischengespeicherten Änderungen

Historie überprüfen

Durchsuchen und Inspizieren der Entwicklung von Projektdateien

```
$ git log
```

Listet die Versionshistorie für den aktuellen Branch auf

```
$ git log --follow [file]
```

Listet die Versionshistorie für die aktuelle Datei auf, inklusive Umbenennungen

```
$ git diff [first-branch]...[second-branch]
```

Zeigt die inhaltlichen Unterschiede zwischen zwei Branches

```
$ git show [commit]
```

Gibt die Änderungen an Inhalt und Metadaten durch den angegebenen Commit aus

Commits wiederholen

Fehler beseitigen und die Historie bereinigen

```
$ git reset [commit]
```

Macht alle Commits nach `[commit]` rückgängig, erhält die Änderungen aber lokal

```
$ git reset --hard [commit]
```

Verwirft die Historie und Änderungen seit dem angegebenen Commit

Änderungen synchronisieren

Registrieren eines externen Repositories (URL) und Tauschen der Repository-Historie

```
$ git fetch [remote]
```

Lädt die gesamte Historie eines externen Repositories herunter

```
$ git merge [remote]/[branch]
```

Integriert den externen Branch in den aktuell lokal ausgecheckten Branch

```
$ git push [remote] [branch]
```

Pusht alle Commits auf dem lokalen Branch zu GitHub

```
$ git pull
```

Pullt die Historie vom externen Repository und integriert die Änderungen

---