Spickzettel: Wichtige & hilfreiche Git-Kommandos

1. Grundlegende Git-Befehle

```
git init               # Neues Repository erstellen
git status            # Status der Dateien anzeigen
git add <datei>       # Datei für Commit vormerken
git commit -m "msg"   # Commit mit Nachricht erstellen
git log               # Commit-Historie anzeigen
```

2. Arbeiten mit Branches

```
git branch               # Lokale Branches anzeigen
git checkout -b <name>   # Neuen Branch erstellen und wechseln
git merge <branch>       # Branch in aktuellen Branch mergen
git branch -d <name>     # Lokalen Branch löschen
```

3. Remote-Repositories

```
git clone <url>              # Repository klonen
git remote -v                # Verbundene Remotes anzeigen
git push origin <branch>     # Änderungen pushen
git pull origin <branch>     # Änderungen vom Remote holen
git fetch origin             # Remote-Änderungen abrufen, ohne zu mergen
```

4. Änderungen rückgängig machen

```
git checkout -- <datei>      # Datei zurücksetzen
git reset HEAD <datei>       # Datei aus Staging entfernen
git reset --soft HEAD~1      # Letzten Commit zurücknehmen (Änderungen
bleiben)
git reset --hard HEAD~1      # Letzten Commit verwerfen (Änderungen gehen
verloren!)
```

5. Historie und Unterschiede analysieren

```
git log --oneline            # Kompakte Commit-Historie
git diff                     # Unterschiede anzeigen
git blame <datei>            # Zeigt, wer welche Zeile geändert hat
```

6. Arbeiten mit Tags

```
git tag v1.0                 # Versionstag erstellen
git tag                      # Alle Tags anzeigen
git push origin v1.0         # Tag pushen
git tag -d v1.0              # Tag lokal löschen
git push origin --delete v1.0  # Tag remote löschen
```

7. Nützliche Git-Aliase setzen

```
git config --global alias.st status   # `git st` für `git status`
git config --global alias.co checkout # `git co` für `git checkout`
git config --global alias.br branch   # `git br` für `git branch`
```

8. Troubleshooting & Wiederherstellung

```
git reflog                    # Zeigt alle Aktionen (auch gelöschte
Branches)
git fsck                      # Überprüfung der Git-Datenbank
git cherry-pick <commit>      # Einzelnen Commit aus anderem Branch
übernehmen
git stash                     # Änderungen zwischenspeichern
git stash pop                 # Änderungen aus Stash wiederherstellen
```

Best Practices

- Häufig `git status` nutzen, um den aktuellen Stand zu prüfen.
- Vor `git reset --hard` immer prüfen, ob Daten verloren gehen könnten.
- Aliase für häufig genutzte Befehle setzen, um schneller zu arbeiten.
- `git stash` verwenden, um Änderungen temporär zu sichern.
- `git reflog` nutzen, um gelöschte Branches oder Commits wiederzufinden.