Git Cheat Sheet für Einsteiger

1. Repository einrichten

- **Neues Repo initialisieren**:

```
git init             # Erstellt ein neues Git-Repository im Verzeichnis
```

- **Existierendes Repo klonen**:

```
git clone <url>      # Klont ein Remote-Repository ins aktuelle Verzeichnis
```

---

2. Konfiguration

- **Benutzername & Email**:

```
git config --global user.name "Dein Name"
git config --global user.email "email@example.com"
```

- **Status prüfen**:

```
git config --list    # Zeigt alle Git-Konfigurationen an
```

---

3. Änderungen verfolgen

- **Dateien zum Index hinzufügen**:

```
git add <datei>      # Fügt eine Datei zur Staging-Area hinzu
git add .            # Alle geänderten und neuen Dateien hinzufügen
```

- **Änderungen committen**:

```
git commit -m "Aussagekräftige Nachricht"  # Speichert die gestagten Änderungen
```

- **Kurze Commit-Übersicht**:

```
git status           # Zeigt den aktuellen Status von Arbeitsverzeichnis und Staging-Area
git diff             # Zeigt nicht gestagte Änderungen
git diff --staged    # Zeigt Unterschiede der gestagten Dateien
```

---

4. Branches & Merging

- **Branch anlegen & wechseln**:

```
git branch <name>    # Neuen Branch erstellen
git checkout <name>  # Zu einem Branch wechseln
```

  Oder in einem Schritt:

```
git checkout -b <name>  # Erstellt und wechselt
```

- **Branches auflisten**:

```
git branch           # Lokale Branches
git branch -r        # Remote-Branches
```

- **Branch mergen**:

```
git checkout main
git merge <feature-branch>  # Fügt Änderungen aus <feature-branch> in main ein
```

---

5. Remote-Repositories

- **Remote hinzufügen**:

```
git remote add origin <url>  # Fügt ein Remote namens 'origin'
```

- **Änderungen pushen**:

```
git push -u origin main       # Erstmaliges Pushen und Upstream setzen
git push                      # Weitere Pushes
```

- **Neues ziehen / Pull**:

```
git pull origin main          # Holt und merged Änderungen aus Remote
```

---

6. Stashing & Cleaning

- **Änderungen zwischenspeichern**:

```
git stash save "Beschreibung"  # Speichert uncommitted changes
git stash list                  # Zeigt alle Stashes
git stash apply                 # Holt letzten Stash zurück
git stash pop                   # Holt und löscht letzten Stash
```

- **Untracked Files entfernen**:

```
git clean -f     # Entfernt alle unversionierten Dateien
git clean -fd    # Entfernt auch unversionierte Verzeichnisse
```

---

7. Tags & Releases

- **Tag erstellen**:

```
git tag -a v1.0 -m "Version 1.0"  # Annotiertes Tag
git tag v1.0                        # Leichtgewichtiges Tag
```

- **Tags pushen**:

```
git push origin v1.0    # Einzelnes Tag
git push --tags         # Alle Tags
```

---

8. Historie & Logs

- **Commit-Historie anzeigen**:

```
git log                  # Detaillierte Historie
git log --oneline --graph --all  # Kompakte Darstellung
```

- **Amend & Reset**:

```
git commit --amend  # Letzten Commit bearbeiten (Message/Änderungen)
git reset --soft HEAD~1  # Entfernt letzten Commit, behält Änderungen gestaged
git reset --hard HEAD~1  # Entfernt letzten Commit und Änderungen
```

---

9. .gitignore

- **Datei erstellen**:

```
# Ignoriere Log-Dateien
*.log

# Ignoriere virtuelle Umgebungen
venv/
```

- **Änderungen übernehmen**:

```
git rm -r --cached .  # Entfernt Dateien aus Tracking, behält sie lokal
git add .
git commit -m ".gitignore aktualisiert"
```

---

10. Nützliche Kurzbefehle

Shortcut	Bedeutung
`git checkout -`	Zurück zum vorherigen Branch
`git reset HEAD <file>`	Unstage Datei
`git show <commit>`	Zeigt Details zu einem Commit
`git remote -v`	Zeigt konfigurierte Remotes
`git reflog`	Zeigt alle Aktionen (inkl. verlorener Commits)

Dieses Cheat Sheet bietet einen kompakten Einstieg – für detaillierte Informationen besuche die offizielle Git-Dokumentation.