Spickzettel: Git Worktrees – Mehrere Branches gleichzeitig nutzen

1. Was sind Git Worktrees?

- Erlauben das parallele Arbeiten an mehreren Branches in separaten Verzeichnissen.
- Nützlich, um mehrere Versionen eines Repositories gleichzeitig zu verwalten.
- Vermeidet das ständige Wechseln (`git checkout`) zwischen Branches.

2. Worktree erstellen

```
git worktree add ../mein-branch feature-branch
```

- Erstellt ein neues Verzeichnis `../mein-branch` mit `feature-branch`.
- Falls der Branch nicht existiert, wird er automatisch erstellt.

3. Bestehende Worktrees anzeigen

```
git worktree list
```

- Zeigt alle aktiven Worktrees und ihre Pfade an.

4. Worktree entfernen

```
git worktree remove ../mein-branch
```

- Entfernt den Worktree (funktioniert nur, wenn keine Änderungen mehr offen sind).
- Falls der Worktree sich nicht entfernen lässt, mit `--force`:

```
git worktree remove --force ../mein-branch
```

5. Worktree für eine ältere Version erstellen

Falls du eine frühere Version des Projekts in einem separaten Ordner brauchst:

```
git worktree add ../old-version <commit-hash>
```

- Erstellt einen Worktree mit dem Stand eines bestimmten Commits.

6. Worktrees für Release- oder Hotfix-Branches

Falls an einem stabilen Release-Branch gearbeitet wird:

```
git worktree add ../release-v1.2 release-branch
```

- Erlaubt das parallele Arbeiten an Fixes, während `main` weiterentwickelt wird.

Best Practices

- Worktrees nutzen, um nicht ständig `git checkout` zwischen Branches zu wechseln.
- Sinnvolle Ordnernamen wählen (`worktree-feature`, `worktree-hotfix`).
- Nicht zu viele Worktrees anlegen, da jeder ein komplettes Repository enthält.
- Regelmäßig ungenutzte Worktrees mit `git worktree remove` aufräumen.