Spickzettel: Git Cherry-Picking & Patching

1. Was ist Cherry-Picking?

- Cherry-Picking ermöglicht das gezielte Übernehmen einzelner Commits in einen anderen Branch.
- Ideal für das schnelle Einbringen von Bugfixes oder spezifischen Änderungen.

Commit in aktuellen Branch übernehmen

```
git cherry-pick <commit-hash>
```

Mehrere Commits übernehmen

```
git cherry-pick <commit1> <commit2>
```

Konflikte während des Cherry-Pick lösen

1. Git meldet einen Konflikt → Datei manuell bereinigen
2. Änderungen speichern und hinzufügen:

```
git add <datei>
```

3. Cherry-Picking fortsetzen:

```
git cherry-pick --continue
```

Cherry-Picking abbrechen

Falls das Cherry-Picking fehlschlägt oder abgebrochen werden soll:

```
git cherry-pick --abort
```

---

2. Was sind Patches in Git?

- Patches sind Dateien mit Änderungen, die auf ein anderes Repository oder einen anderen Branch angewendet werden können.
- Ideal für das Weitergeben von Änderungen ohne direkten Push.

Patch-Datei erstellen

```
git format-patch -1 <commit-hash>
```

- Erstellt eine `.patch`-Datei mit den Änderungen des angegebenen Commits.
- `-1` bedeutet: Nur den letzten Commit als Patch erstellen.

Patch anwenden

```
git apply <datei.patch>
```

Patch mit Commit übernehmen

```
git am <datei.patch>
```

- Wendet den Patch an und erstellt direkt einen Commit.

Patch auf spezifischen Branch anwenden

```
git checkout feature-branch
git apply <datei.patch>
```

Best Practices

- Cherry-Picking nur gezielt nutzen, um Verwirrung in der Historie zu vermeiden.
- Bei regelmäßigen Übernahmen besser mit `git rebase` arbeiten.
- Vor Anwendung eines Patches mit `git apply --check <datei.patch>` prüfen, ob der Patch sauber übernommen werden kann.
- Konflikte nach einem Cherry-Pick direkt bereinigen und nicht unaufgelöst pushen.