Spickzettel: Merges, Merge-Probleme & Konfliktlösung

1. Merge-Strategien in Git

- **Fast-Forward Merge:**
  - Einfacher Merge ohne zusätzlichen Commit, wenn keine neuen Commits auf `main` existieren.
- `git merge --ff feature-branch`
- **Merge mit Commit:**
  - Erstellt einen expliziten Merge-Commit, um die Historie nachvollziehbar zu halten.
- `git merge --no-ff feature-branch`
- **Squash Merge:**
  - Vereinigt alle Commits eines Branches zu einem einzigen Commit.
- `git merge --squash feature-branch`
- **Rebase statt Merge:**
  - Setzt die Commits des Feature-Branches neu auf `main` auf (keine Merge-Commits, lineare Historie).
- `git rebase main`

---

2. Typische Merge-Probleme

1. Merge-Konflikte

- Entstehen, wenn zwei Commits dieselbe Stelle in einer Datei geändert haben.
- Git zeigt Konflikte in der betroffenen Datei so an:

```
<<<<<<< HEAD
Änderung aus main
=======
Änderung aus feature-branch
>>>>>>> feature-branch
```

2. Merge-Konflikt lösen

1. Datei öffnen und Konfliktstellen manuell bearbeiten.
2. Bereinigte Datei als gelöst markieren:

```
git add <datei>
```

3. Commit erstellen, um den Konflikt abzuschließen:

```
git commit -m "Merge-Konflikt in <datei> gelöst"
```

3. Merge-Konflikt abbrechen

Falls der Merge fehlschlägt oder neu gestartet werden soll:

```
git merge --abort
```

4. Konflikte während Rebase lösen

- Falls Konflikte bei einem `rebase` auftreten:

```
git rebase --continue
```

- Falls das Rebase zu komplex wird:

```
git rebase --abort
```

Best Practices für Merges

- **Regelmäßig `git pull` ausführen**, um Konflikte frühzeitig zu vermeiden.
- **Feature-Branches klein halten**, damit weniger Konflikte entstehen.
- **Merge-Konflikte frühzeitig erkennen und direkt lösen.**
- **Merges niemals ungetestet pushen!** Erst lokal überprüfen und dann ins zentrale Repository hochladen.