Spickzettel: GitHub Repositories – Funktionen & Nutzung

1. Was ist ein GitHub Repository?

- Speicherort für Code, Dokumentation & Projektdateien.
- Unterstützt **Branches, Tags, Pull Requests & Issues**.
- Kann **öffentlich oder privat** sein.
- Synchronisiert mit lokalem Git-Repository.

2. Repository erstellen

1. Auf GitHub einloggen → **New Repository** klicken.
2. **Name, Beschreibung & Sichtbarkeit** festlegen.
3. Optionale Initialisierung mit **README, .gitignore, LICENSE**.

Alternativ per Terminal erstellen:

```
git init --bare my-repo.git
```

3. Repository klonen

```
git clone https://github.com/user/repo.git
```

- Erstellt eine lokale Kopie des Remote-Repos.
- Mit `--depth=1` nur die letzte Historie klonen (schneller).

4. Repository-URL anzeigen & ändern

```
git remote -v              # Zeigt die aktuelle Remote-URL
git remote set-url origin <neue-url>  # Ändert die Remote-URL
```

5. Änderungen synchronisieren

```
git pull origin main       # Änderungen vom Remote abrufen
git push origin main       # Änderungen hochladen
```

6. Branches im Remote verwalten

```
git branch -r              # Zeigt alle Remote-Branches
git push origin <branch>   # Branch zum Remote hochladen
git push --delete origin <branch>  # Remote-Branch löschen
```

7. Repository verwalten

- **Zugriffsrechte:** Teams & Kollaborateure verwalten
- **GitHub Actions:** CI/CD-Pipelines für Automatisierung

- **Code Scanning & Dependabot:** Sicherheitsprüfungen
- **Wikis & Projects:** Dokumentation & Aufgabenverwaltung

8. Repository löschen (Vorsicht!)

- **Einstellungen → Danger Zone → Delete Repository**
- Lokal entfernen:

```
rm -rf .git  # ACHTUNG: Entfernt alle Versionsdaten lokal!
```

Best Practices

- **README & LICENSE hinzufügen**, um das Projekt verständlich zu machen.
- **.gitignore nutzen**, um unerwünschte Dateien auszuschließen.
- **Regelmäßig `git pull` nutzen**, um aktuelle Änderungen zu erhalten.
- **GitHub Issues & Projects verwenden**, um den Workflow zu organisieren.