Spickzettel: Umgang mit großen Dateien & Binaries in Git & GitHub

1. Problem mit großen Dateien in Git

- Git speichert jede Version einer Datei → große Dateien blähen das Repository auf.
- Standard-Git ist für **Textdateien optimiert**, nicht für große Binärdateien.
- GitHub hat **Limit von 100 MB pro Datei** (Soft-Limit) und **2 GB Repository-Limit**.

2. `.gitignore` für große oder unnötige Dateien

```
echo "*.log" >> .gitignore
```

- Vor dem ersten Commit große oder temporäre Dateien ignorieren.
- Beispiel `.gitignore` für große Dateien:

```
*.iso
*.zip
*.tar.gz
*.mp4
/node_modules/
```

3. Git LFS (Large File Storage) nutzen

- Speichert große Dateien separat und ersetzt sie in Git durch **Referenzen**.

Installation & Einrichtung von Git LFS

```
git lfs install
```

- **Dateitypen für LFS registrieren:**

```
git lfs track "*.psd"
git lfs track "*.mp4"
```

- `.gitattributes` wird automatisch aktualisiert.

Nutzung von Git LFS

```
git add .gitattributes  # LFS-Dateitypen zur Versionskontrolle hinzufügen
git add <große_datei>
git commit -m "Große Datei mit LFS getrackt"
git push origin main
```

- Beim Klonen oder Pullen:

```
git lfs pull
```

4. Alternative: GitHub Releases für große Dateien

- Statt große Dateien direkt ins Repository zu committen, über **GitHub Releases** verwalten.
- Dateien als **Release-Assets** hochladen.
- Automatisierte Builds und Artefakte dort bereitstellen.

5. Repository aufräumen, falls große Dateien bereits committed wurden

Falls eine große Datei versehentlich im Git-Verlauf ist:

```
git filter-branch --tree-filter 'rm -f <große_datei>' HEAD
```

- **Achtung:** Erzeugt neue Commit-Hashes, erfordert ein **erzwungenes Pushen** (`git push --force`).
- Alternativ **BFG Repo Cleaner** verwenden:

```
bfg --delete-files <große_datei>
```

Best Practices

- **Immer `.gitignore` nutzen**, um unnötige Dateien gar nicht erst zu committen.
- **Git LFS verwenden**, wenn große Dateien versioniert werden müssen.
- **GitHub Releases nutzen**, um Binärdateien zu verwalten.
- **Repository regelmäßig prüfen**, um Speicherplatzprobleme zu vermeiden (`git count-objects -vH`).