GIT

- Remove unwanted files from a git repo AFTER adding a .gitignore.
- origin
- Umzug eines Repositories
- Umzug eines Repositories

Remove unwanted files from a git repo AFTER adding a .gitignore.

```
# See the unwanted files:
git ls-files -ci --exclude-standard

# Remove the unwanted files:
git ls-files -ci --exclude-standard -z | xargs -0 git rm --cached

# Commit changes
git commit -am "Removed unwanted files marked in .gitignore"

# Push
git push origin master # or whatever branch you're on
```

source

origin

Wofür steht eigentlich "origin"?

Wenn du ein Repository mit `git clone <URL>` von GitHub, GitLab oder Bitbucket auf deinen Rechner holst, passiert im Hintergrund automatisch Folgendes:

1. Git lädt den Code herunter.
2. Git merkt sich die URL, von der der Code kam.
3. Weil sich niemand lange URLs wie `https://github.com/user/projekt-name.git` merken oder sie ständig eintippen will, gibt Git dieser URL den Standard-Spitznamen **`origin`** (engl. für *Ursprung* oder *Herkunft*).

1. Beim Anlegen eines Remotes (`git remote add...`)

Wenn du ein brandneues Projekt lokal auf deinem Rechner startest (per `git init`) und es danach auf GitHub hochladen willst, existiert diese automatische Verknüpfung noch nicht. Du musst sie selbst anlegen:

```
git remote add origin https://github.com/deinname/dein-projekt.git
```

Dieser Befehl sagt Git: *„Hey, merk dir bitte diese lange GitHub-URL unter dem kurzen Namen `origin`." (Hinweis: Du könntest das Ding hier auch `banane` oder `server` nennen. Aber `origin` hat sich als weltweiter Standard etabliert, so wie `main` oder `master` für den Haupt-Branch).*

2. Beim Pushen eines neuen Branches (`git push -u origin...`)

Wenn du einen neuen lokalen Branch namens `feature-xyz` erstellt hast und ihn das erste Mal hochladen willst, sagst du:

```
git push -u origin feature-xyz
```

Das bedeutet übersetzt: *„Git, pushe meinen lokalen Branch `feature-xyz` zu dem Server, den ich vorhin **`origin`** genannt habe, und richte dort einen gleichnamigen Branch ein."* (Das `-u` sorgt nur dafür, dass sich Git diese Verbindung für die Zukunft merkt, sodass danach ein einfaches `git push` reicht).

Umzug eines Repositories

von GitHub zu Forgejo via Migration

Der Umzug eines Repositories von GitHub zu Forgejo lässt sich schnell durchführen, da Forgejo eine eingebaute **Migration-Funktion** bietet. Dadurch werden nicht nur der Code und die Commits, sondern optional auch Issues, Pull Requests, Releases und Wikis automatisch übernommen.

Hier sind die Schritte für die Migration:

1.GitHub Personal Access Token erstellen:
Erforderlich für private Repositories oder Meta-Daten wie PRs und Issues.

1. Gehe auf GitHub zu **Settings** -> **Developer Settings** -> **Personal Access Tokens** -> **Tokens (classic)**.
2. Erstelle einen neuen Token (*Generate new token*).
3. Gib ihm mindestens die Berechtigung `repo` (für Zugriff auf Repositories) und erstelle den Token. Kopiere ihn direkt ab.

2.Migration in Forgejo starten:

1. Melde dich bei deiner Forgejo-Instanz an.
2. Klicke oben rechts auf das **+**-Symbol und wähle **Neue Migration** (oder *New Migration*).
3. Wähle **GitHub** als Quell-Plattform aus.

3.Zugangsdaten und Optionen konfigurieren:

1. Trage dein GitHub Personal Access Token im Feld **Access Token** ein.
2. Gib die **Clone URL** des GitHub-Repositories an (z. B. `[https://github.com/user/repo.git](https://github.com/user/repo.git)`).
3. Wähle den neuen Repository-Namen und den Eigentümer (User oder Organisation) in Forgejo aus.
4. Setze Haken bei den Elementen, die du migrieren möchtest:
   - **Code & Commits**
   - **Issues & Issue-Kommentare**
   - **Pull Requests**
   - **Releases & Attachments**
   - **Wiki**
5. Klicke auf **Migration starten**.

4.Lokales Git Remote aktualisieren: Lokal auf deinem Computer ausführen.

Wenn du das Repository bereits lokal gekontrollierst oder daran arbeitest, passe die Remote-URL an deine Forgejo-Instanz an:

Bash

```
# Aktuelle URL prüfen
git remote -v

# Remote auf Forgejo umstellen
git remote set-url origin https://deine-forgejo-instanz.de/user/repo.git

# Änderungen zur Bestätigung pushen/fetchen
git fetch origin
```

Was du beachten solltest

- **GitHub Actions vs. Forgejo Actions:** CI/CD-Pipelines (`.github/workflows/`) werden als Dateien mitkopiert, funktionieren auf Forgejo aber nur, wenn dort **Forgejo Actions** (basierend auf *act_runner*) aktiviert ist. Eventuell musst du Syntax-Details anpassen.
- **Webhooks & Integrations:** Externe Anbindungen (Discord-Bots, Deployment-Hooks etc.) werden nicht automatisch übertragen und müssen in Forgejo unter *Repository-Einstellungen -> Webhooks* neu angelegt werden.

Umzug eines Repositories

via git push --mirror

Der manuelle Umzug per `--mirror` ist ideal, wenn du ein reines Git-Repository inklusive aller **Branches, Tags und der kompletten Commit-Historie** übertragen möchtest – ganz ohne GitHub-Tokens oder Metadaten wie PRs und Issues.

Hier ist der genaue Ablauf über das Terminal:

1.Leeres Repository in Forgejo erstellen: Auf der Ziel-Plattform.

1. Melde dich bei deiner Forgejo-Instanz an.
2. Klicke auf das **+**-Symbol oben rechts und wähle **Neues Repository**.
3. Gib den Repository-Namen ein.
4. **Wichtig:** Initialisiere das Repository **nicht** (keine README, keine `.gitignore` und keine Lizenz anlegen lassen). Es muss vollkommen leer sein.
5. Kopiere die Git-URL des neuen Repositories (z. B. `[https://forgejo.dein-server.de/user/repo.git](https://forgejo.dein-server.de/user/repo.git)` oder per SSH `git@forgejo.dein-server.de:user/repo.git`).

2.Bare-Clone des GitHub-Repositories erstellen: Lokal auf deinem Computer.

Erstelle einen temporären "Bare-Clone" des GitHub-Repositories. Dieser enthält nur die Git-Datenbank ohne Arbeitsverzeichnis:

Bash

```
git clone --bare https://github.com/nutzername/repository.git mirror-temp.git
cd mirror-temp.git
```

3.Repository zu Forgejo spiegeln:

Pushe den gesamten Stand (alle Referenzen, Branches und Tags) per Mirror-Option zu Forgejo:

Bash

```
git push --mirror https://forgejo.dein-server.de/user/repo.git
```

(Wenn du SSH nutzt, verwende die SSH-URL: `git push --mirror git@forgejo.dein-server.de:user/repo.git`)

4.Aufräumen und Remote-URL aktualisieren: Saubermachen und bestehende Arbeitskopie anpassen.

1. Lösche den temporären Bare-Ordner wieder:

Bash

```
cd ..
rm -rf mirror-temp.git
```

2. Wenn du bereits einen normalen Arbeitsordner des Repositories auf deinem Rechner hast, wechsle hinein und passe die Remote-URL an:

Bash

```
cd dein-projekt-ordner
git remote set-url origin https://forgejo.dein-server.de/user/repo.git
git fetch origin
```

Was macht `--mirror` anders als ein normaler Push?

- **`git push --all`** überträgt nur alle lokalen Branches.
- **`git push --mirror`** spiegelt den *exakten* Zustand aller Referenzen: Lokale Branches, Remote-Tracking-Branches, Tags sowie Notizen (`refs/notes/`). Es überschreibt das Ziel-Repository komplett, damit es exakt dem Quell-Repository entspricht.