KI Werkzeug Cloude

GitHub mit Claude verbinden und Dateien vom V-Server oder Handy (Termux) hochladen

07.11.2026 · @Sebastian Ebner · Quelle: KI Werkzeug Cloude

> Alle Namen sind Platzhalter: `BENUTZER` (GitHub-Benutzername), `REPONAME` (Name des privaten Repos), `GHT` (Variable für den Token). Durch die eigenen ersetzen. Den Token niemals in einen Chat kopieren.

---

1. Überblick

Große Dateien wie Server-Logs lassen sich nicht direkt in den Chat laden, und Links zu Google Drive oder Dropbox kann Claude nicht abrufen. GitHub dagegen erreicht Claude direkt. Der Weg führt deshalb über ein **privates GitHub-Repo**.

Schritt	Wo	Was passiert
1	GitHub (Browser)	Privates Repo anlegen
2	Claude-App	GitHub mit Claude verbinden, nur dieses Repo freigeben
3	GitHub (Browser)	Token mit Schreibrecht nur für dieses Repo erstellen
4	V-Server oder Handy (Termux)	Repo klonen, Dateien hineinlegen, `git push`
5	Claude-Chat	„Aus GitHub abholen“: Claude lädt das Repo und wertet es aus
6	GitHub, V-Server, Termux	Repo, Token und lokale Kopien löschen

Claude braucht den Token **nicht**. Der Token ist nur für den V-Server bzw. Termux, Claude liest über die verbundene GitHub-App.

---

2. GitHub vorbereiten

2.1 Privates Repo anlegen

Schritt	Aktion
1	Auf github.com anmelden, oben rechts **+** → **New repository**
2	Repository name: `REPONAME`
3	**Private** wählen (wichtig, sonst sind die Dateien öffentlich)
4	Haken bei **Add a README file** setzen, damit das Repo nicht leer ist

Schritt	Aktion
5	**Create repository**

2.2 GitHub mit Claude verbinden

Schritt	Aktion
1	In der Claude-App: Einstellungen → **Konnektoren** (Connectors) → **GitHub** → Verbinden
2	Bei GitHub anmelden und die Claude-App installieren
3	Bei Repository access **Only select repositories** wählen und nur `REPONAME` auswählen
4	Speichern. Danach in einer neuen Chat-Nachricht prüfen: Claude kann das Repo sehen

Findet sich GitHub nicht in der Liste der Konnektoren, die App neu starten oder im Browser auf claude.ai unter Einstellungen nachsehen.

2.3 Token für den V-Server erstellen (fine-grained)

Schritt	Aktion
1	Profilbild oben rechts → **Settings**
2	Ganz unten links **Developer settings** → **Personal access tokens** → **Fine-grained tokens**
3	**Generate new token**, Name z. B. `REPONAME`
4	**Expiration**: kurz halten, z. B. 7 Tage
5	**Repository access**: **Only select repositories** → `REPONAME`
6	**Permissions** → **Repository permissions** → **Contents**: **Read and write** (Metadata: Read-only wird automatisch gesetzt)
7	**Generate token** und den Token sofort kopieren, er wird nur einmal angezeigt

Der Token beginnt mit `github_pat_`. Er ersetzt bei git das Passwort. Mit dem normalen GitHub-Passwort funktioniert `git push` nicht.

---

3. Auf dem V-Server hochladen

3.1 Einmalig: git installieren

```
apt install -y git
```

3.2 Token sicher eingeben und prüfen

```
read -rs GHT
echo ${#GHT}
curl -s -o /dev/null -w "%{http_code}\n" -H "Authorization: Bearer $GHT" https://api.github.com/repos/BENUTZER/REPONAME
```

- `read -rs GHT`: Token einfügen und Enter drücken. Er wird nicht angezeigt und nicht im Verlauf gespeichert.
- `echo ${#GHT}`: zeigt nur die Länge (bei fine-grained Tokens um 93 Zeichen), nicht den Token.
- `curl …`: **200** = Token passt, **401** = Token falsch oder abgelaufen, **404** = Token hat keinen Zugriff auf das Repo.

3.3 Repo klonen

```
cd /root
git clone https://BENUTZER:$GHT@github.com/BENUTZER/REPONAME.git
cd REPONAME
git remote set-url origin https://github.com/BENUTZER/REPONAME.git
git config user.name "BENUTZER"
git config user.email "BENUTZER@users.noreply.github.com"
```

- Zeile 2 lädt das Repo nach `/root/REPONAME`.
- Zeile 4 entfernt den Token wieder aus der gespeicherten Adresse in `.git/config`.
- Zeilen 5 und 6 setzen den Namen für die Commits, die noreply-Adresse verrät keine echte E-Mail.

3.4 Dateien in den Repo-Ordner legen

Beispiel: Logs packen, direkt in den Repo-Ordner.

```
cd /root/REPONAME
last -F > anmeldungen.txt
tar czf logs-$(hostname)-$(date +%F).tar.gz /var/log/auth.log* /var/log/syslog* /var/log/
kern.log* /var/log/dpkg.log* /var/log/apt/history.log* anmeldungen.txt
ls -lh
```

Dateien vom Handy kommen per SFTP (z. B. Cx Datei Explorer) in denselben Ordner `/root/REPONAME`.

Größengrenze: GitHub nimmt einzelne Dateien bis **100 MB** an (ab 50 MB mit Warnung). Größere Dateien vorher teilen:

```
split -b 90M grosse-datei.tar.gz grosse-datei.tar.gz.teil-
rm grosse-datei.tar.gz
```

Zusammensetzen später mit `cat grosse-datei.tar.gz.teil-* > grosse-datei.tar.gz`. Das Repo als Ganzes sollte unter etwa 1 bis 5 GB bleiben.

3.5 Hochladen (push)

```
cd /root/REPONAME
git add .
git commit -m "Logs vom $(date +%d.%m.%Y)"
git push https://BENUTZER:$GHT@github.com/BENUTZER/REPONAME.git HEAD
unset GHT
```

- `git add .`: alle neuen Dateien vormerken.

- `git commit`: Stand mit einer Beschreibung festhalten.
- `git push … HEAD`: auf GitHub hochladen, der Token wird nur für diesen einen Befehl verwendet.
- `unset GHT`: Token aus der Sitzung löschen.

Danach auf github.com im Repo prüfen, ob die Dateien da sind.

---

4. Alternative: Vom Android-Handy mit Termux hochladen

Termux ist ein Linux-Terminal für Android. Damit funktioniert alles wie auf dem V-Server, nur direkt auf dem Handy. So lassen sich Dateien vom Handy (Speicherlisten, Screenshots, Log-Exporte) ohne Umweg über SFTP und Server zu GitHub hochladen.

4.1 Einmalig einrichten

```
pkg update && pkg upgrade -y
pkg install -y git openssh p7zip
termux-setup-storage
```

- Zeile 2 installiert git, ssh/scp und 7-Zip zum Packen und Teilen.
- Zeile 3 fragt nach der Speicherberechtigung. Danach gibt es `~/storage/shared` (interner Speicher) und `~/storage/downloads` (Download-Ordner).
- Termux aus **F-Droid** oder von **GitHub** installieren. Die Version im Play Store ist veraltet, dort funktioniert `pkg` oft nicht.

4.2 Token eingeben und Repo klonen

```
read -rs GHT
echo ${#GHT}
cd ~
git clone https://BENUTZER:$GHT@github.com/BENUTZER/REPONAME.git
cd REPONAME
git remote set-url origin https://github.com/BENUTZER/REPONAME.git
git config user.name "BENUTZER"
git config user.email "BENUTZER@users.noreply.github.com"
```

Befehl	Was er macht
`read -rs GHT`	Token unsichtbar einfügen, wird nicht im Verlauf gespeichert
`git clone …`	lädt das **bestehende** Repo als Ordner `~/REPONAME` herunter (das Repo muss vorher auf github.com angelegt sein)
`git remote set-url origin …`	entfernt den Token aus der gespeicherten Adresse in `.git/config`
`git config user.name / user.email`	Verfasser für die Commits, ohne diese Angaben bricht `git commit` ab

Wichtig: Das Repo gehört in den Termux-Ordner `~`, **nicht** in den Handyspeicher `~/storage/shared`. Dort kann git keine Dateirechte setzen und bricht ab. Jeder Befehl muss in

einer Zeile stehen, ein Zeilenumbruch auf dem Handy teilt ihn sonst.

Den Klon braucht es nur einmal. Für spätere Uploads genügt `cd ~/REPONAME`.

4.3 Dateien und Ordner ins Repo kopieren

```
cp ~/storage/downloads/DATEI.txt ~/REPONAME/
cp -r ~/storage/downloads/ORDNER ~/REPONAME/
cp -r ~/storage/downloads/ORDNER/. ~/REPONAME/
```

- Zeile 1: eine einzelne Datei.
- Zeile 2: einen Ordner als Unterordner (`-r` = mit allem Inhalt).
- Zeile 3: nur den **Inhalt** eines Ordners, ohne den Ordner selbst (`/.` am Ende).

Dateien vom V-Server lassen sich über das Handy mitnehmen: `scp root@SERVER-IP:/root/DATEI.tar.gz ~/REPONAME/`

Vor dem Hochladen prüfen, was drin ist, und Schlüsseldateien ausschließen:

```
cd ~/REPONAME
ls -laR | head -50
find . -path ./.git -prune -o \( -name '*.rbk' -o -name '*.conf' -o -name '*.key' \) -pri
nt
```

Die zweite Zeile muss leer bleiben. Im Download-Ordner liegen oft RethinkDNS-Sicherungen und WireGuard-Konfigurationen, deshalb gezielt kopieren statt den ganzen Ordner.

4.4 Große Ordner und Dateien in Teile packen (wie rar)

Mit 7-Zip, Teile zu je 90 MB:

```
cd ~/storage/downloads
7z a -v90m archiv.7z ORDNER
```

Teil	Bedeutung
`cd ~/storage/downloads`	in den Download-Ordner des Handys wechseln, dort entsteht das Archiv
`7z`	Packprogramm 7-Zip
`a`	„add“, Archiv anlegen
`-v90m`	„volume“, Teile zu 90 MB (`-v500k`, `-v1g` ebenso möglich)
`archiv.7z`	Name des Archivs, frei wählbar
`ORDNER`	was gepackt wird, relativ zum aktuellen Ordner, mehrere Angaben möglich

Ergebnis: `archiv.7z.001`, `archiv.7z.002`, … Der Originalordner bleibt unverändert.

Zweck	Befehl
Mit Passwort, auch Dateinamen verschlüsselt	`7z a -v90m -p -mhe=on archiv.7z ORDNER`

Zweck	Befehl
Stärker packen (gut bei Logs)	`7z a -v90m -mx=9 archiv.7z ORDNER`
Teile prüfen	`7z t archiv.7z.001` („Everything is Ok“)
Entpacken	`7z x archiv.7z.001` (alle Teile im selben Ordner)

Ohne Zusatzprogramm, mit tar und split:

```
tar czf - ORDNER | split -b 90M - archiv.tar.gz.teil-
cat archiv.tar.gz.teil-* | tar xzf -
```

Zeile 1 packt und teilt in einem Schritt, Zeile 2 setzt zusammen und entpackt.

Ohne Teilen geht es nur, wenn jede Datei unter 100 MB bleibt. Für größere Einzeldateien gibt es **Git LFS** (`pkg install git-lfs`, `git lfs install`, `git lfs track "*.zip"`, bis 2 GB pro Datei). LFS hat ein begrenztes Gratis-Kontingent, und ob Claude LFS-Dateien abholen kann, ist nicht getestet. Teilen ist der sichere Weg.

4.5 Hochladen

Die Befehle **einzeln** eingeben:

```
cd ~/REPONAME
git add .
git commit -m "Dateien vom $(date +%d.%m.%Y)"
git push https://BENUTZER:$GHT@github.com/BENUTZER/REPONAME.git HEAD
unset GHT
```

- Nach `git commit` muss eine Zeile wie „3 files changed“ erscheinen. „nothing to commit“ heißt: nichts Neues im Ordner.
- Nach `git push` muss etwas wie „main -> main“ erscheinen. „Everything up-to-date“ heißt: Es gab keinen neuen Commit.
- Verkettet mit `&&` bricht die Kette nach „nothing to commit“ ab, und der push läuft nicht.
- Bei großen Uploads Termux im Vordergrund lassen oder in der Termux-Benachrichtigung **Acquire wakelock** antippen, damit Android nicht abbricht.
- Läuft Termux in RethinkDNS über einen WireGuard-Tunnel, geht der Upload über diesen Tunnel, das funktioniert normal.

---

5. In Claude abholen

Im Chat genügt eine Nachricht wie: **„Hole die Dateien aus GitHub REPONAME und erstelle einen Bericht.“**

Claude lädt das Repo in seine Arbeitsumgebung, entpackt die Archive (auch geteilte und mehrteilige) und wertet sie aus. Der Token wird dafür nicht gebraucht und gehört nicht in den Chat.

---

6. Danach aufräumen

Was	Wie
Repo löschen	Im Repo **Settings** → **General** → ganz unten **Danger Zone** → **Delete this repository**
Token löschen	Settings → Developer settings → Fine-grained tokens → Token → **Delete**
Kopie auf dem V-Server	`rm -rf /root/REPONAME`
Kopie in Termux	`rm -rf ~/REPONAME` und `unset GHT`
Zugriff der Claude-App (optional)	Settings → Applications → Installed GitHub Apps → Claude → Configure → Repo entfernen oder **Uninstall**
Kopien bei Claude	Im Chat: „Lösche alle Kopien“

Wichtig: Eine Datei nachträglich aus dem Repo zu löschen reicht nicht, sie bleibt in der git-Historie. Bei versehentlich hochgeladenen geheimen Dateien das **ganze Repo löschen**.

Niemals hochladen: private Schlüssel und Dateien, die sie enthalten, z. B. RethinkDNS-Sicherungen (`.rbk`), WireGuard-Konfigurationen (`.conf` mit `PrivateKey`), `server.key` , das EOTK-Verzeichnis `secrets.d` mit den Onion-Schlüsseln, Passwortdateien. Ist es doch passiert: Repo löschen und die betroffenen Schlüssel neu erzeugen.

7. Fehler und Lösungen

Meldung	Ursache	Lösung
`Password authentication is not supported` oder `Authentication failed`	Passwort statt Token verwendet	Token mit `read -rs GHT` eingeben und `$GHT` in der Adresse verwenden
`Repository not found`	Platzhalter `REPONAME` wörtlich übernommen oder Token ohne Zugriff auf das Repo	Echten Namen einsetzen, beim Token „Only select repositories“ prüfen
`fatal: not a git repository`	Befehl nicht im Repo-Ordner ausgeführt	`cd /root/REPONAME` , in Termux `cd ~/REPONAME`
`exceeds GitHub's file size limit of 100.00 MB`	Datei zu groß	Mit `split -b 90M` teilen, mit `git reset HEAD~1` den Commit zurücknehmen, dann neu committen
`curl` zeigt 401	Token falsch kopiert oder abgelaufen	Neuen Token erstellen
Claude findet das Repo nicht	GitHub nicht verbunden oder Repo bei der Claude-App nicht freigegeben	Schritt 2.2 wiederholen, Repo unter „Only select repositories“ hinzufügen

Meldung	Ursache	Lösung
`pkg` : Fehler beim Aktualisieren, Spiegel nicht erreichbar	Termux aus dem Play Store (veraltet)	Termux aus F-Droid oder GitHub neu installieren
`~/storage/downloads: No such file or directory`	Speicherzugriff nicht eingerichtet	`termux-setup-storage` ausführen und Berechtigung erlauben
`unable to set permissions` oder `dubious ownership`	Repo im Handyspeicher (`~/storage/…`) angelegt	Repo in `~` klonen, Dateien nur hineinkopieren
`usage: git remote set-url …` und danach `No such file or directory`	Befehl durch Zeilenumbruch in zwei Zeilen geteilt	Befehl in einer Zeile eingeben
`cp: -r not specified; omitting directory`	Ordner ohne `-r` kopiert	`cp -r` verwenden
Doppelter Ordner `REPONAME/REPONAME`	Ordner mit gleichem Namen als Ganzes kopiert	Mit `/.` am Ende nur den Inhalt kopieren
`nothing to commit, working tree clean`	Keine neuen Dateien im Repo-Ordner	Erst kopieren, mit `git status` prüfen
`Everything up-to-date`	push ohne vorherigen `git add` und `git commit`	`git add .` , `git commit -m "…"` , dann push
`Your branch is ahead of 'origin/main' by 1 commit` , obwohl hochgeladen	Push über die volle Adresse aktualisiert die Anzeige nicht	Harmlos, auf github.com prüfen, ob die Dateien da sind
push bricht mitten im Upload ab	Android beendet Termux im Hintergrund	„Acquire wakelock“ antippen, Termux im Vordergrund lassen
Nur eine kleine Textdatei statt der echten Datei auf GitHub	Datei über Git LFS hochgeladen	Lieber mit 7z oder split teilen

Quellen

1. GitHub Docs, Neues Repository erstellen: https://docs.github.com/en/repositories/creating-and-managing-repositories/creating-a-new-repository
2. GitHub Docs, Personal Access Tokens (fine-grained) verwalten: https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens
3. GitHub Docs, Große Dateien auf GitHub (Grenzen 50 MB / 100 MB): https://docs.github.com/en/repositories/working-with-files/managing-large-files/about-large-files-on-github
4. GitHub Docs, Repository löschen: https://docs.github.com/en/repositories/creating-and-managing-repositories/deleting-a-repository
5. GitHub Docs, Sensible Daten aus einem Repository entfernen: https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/removing-sensitive-data-from-a-repository
6. Git Dokumentation (clone, add, commit, push, remote): https://git-scm.com/docs
7. Termux, App und Installation (F-Droid / GitHub): https://github.com/termux/termux-app
8. Termux Wiki, Speicherzugriff mit termux-setup-storage: https://wiki.termux.com/wiki/Termux-setup-storage
9. Termux Wiki, Paketverwaltung mit pkg: https://wiki.termux.com/wiki/Package_Management
10. 7-Zip, Programm und Befehlszeile (a, x, t, -v, -p, -mhe, -mx): https://www.7-zip.org/
11. GNU coreutils, split: https://www.gnu.org/software/coreutils/manual/html_node/split-invocation.html
12. Git LFS (Large File Storage): https://git-lfs.com/
13. GitHub Docs, Git LFS Speicher und Grenzen: https://docs.github.com/en/repositories/working-with-files/managing-large-files/about-git-large-file-storage
14. GitHub Docs, Installierte GitHub-Apps verwalten: https://docs.github.com/en/apps/using-github-apps/reviewing-and-modifying-installed-github-apps
15. Claude Hilfe-Center (Konnektoren, GitHub-Integration): https://support.claude.com
16. Eigene Einrichtung und Tests von Sebastian Ebner (Token-Erstellung, Upload der Server-Logs vom V-Server und der Speicherlisten mit Termux, Abholung im Chat, aufgetretene Fehlermeldungen)