

Windows Anleitungen

Dieser Bereich enthält Anleitungen für Windows 10 & 11.

- [Git Anleitungen](#)
 - [GitHub Commit-Signierung per SSH](#)

Git Anleitungen

GitHub Commit-Signierung per SSH

Disclaimer

Diese Anleitung zeigt dir, wie du Git-Commits **mit einem SSH-Schlüssel signieren** kannst. Das sorgt dafür, dass Plattformen wie GitHub deine Commits als „**Verified**“ anzeigen, sofern dein öffentlicher Schlüssel dort hinterlegt ist.

Die Einrichtung erfolgt **zunächst lokal** für ein einzelnes Repository – so kannst du dich langsam mit signierten Commits vertraut machen. Eine **globale Konfiguration** ist optional am Ende beschrieben.

Voraussetzungen

- [Git](#) ist installiert (`git --version`)
- Du verfügst über ein [GitHub-Konto](#)
- SSH ist auf deinem System verfügbar (`ssh -V`)

Schritt 1: SSH-Schlüssel erzeugen

Wenn du bereits einen SSH-Schlüssel unter `~/.ssh/id_ed25519` verwendest, kannst du diesen Schritt überspringen.

1. Öffne ein Terminal (am besten cmd.exe)
2. Führe folgenden Befehl aus:

```
ssh-keygen -t ed25519 -C "deine@email.de"
```

3. Drücke Enter, um den Standardspeicherort zu wählen: `~/.ssh/id_ed25519`
4. Optional: Vergib eine Passphrase für zusätzlichen Schutz

Schritt 2: SSH-Schlüssel auf GitHub hochladen

1. Gehe zu <https://github.com/settings/ssh>
2. Klicke auf "**New SSH key**"
3. Vergib einen sinnvollen **Titel** (z. B. „*GitHub_Sign_Key*“)
4. Ändere den **Key type** auf "**Signing Key**"
5. Gib deinen öffentlichen Schlüssel im Terminal aus:

```
type %USERPROFILE%\ssh\id_ed25519.pub
```

5. Kopiere den gesamten Inhalt (beginnt mit `ssh-ed25519`) und füge ihn bei GitHub ein.
6. Klicke auf "**Add SSH key**"

Schritt 3: SSH-Signierung für *ein* Repository aktivieren

Wechsle in dein Repository-Verzeichnis und führe folgende Befehle aus:

```
git config commit.gpgsign true
git config gpg.format ssh
git config user.signingkey ~/.ssh/id_ed25519.pub
```

Das aktiviert die Signaturpflicht für Commits **nur in diesem Repository**.

Schritt 4: Signatur Testen

1. Erstelle eine Testdatei:

```
echo "Signatur-Test" > test.txt
```

2. Commit erstellen:

```
git add .
git commit -m "Signierter Commit per SSH"
```

3. Push auf GitHub:

```
git push origin main
```

Rufe danach den Commit auf GitHub auf – er sollte mit einem „**Verified**“-**Badge** markiert sein.

Fehlerbehebung

| Problem | Mögliche Ursache |
|---------------------------|---|
| Commit ist nicht signiert | <code>commit.gpgsign</code> ist nicht aktiviert oder falscher Schlüsselpfad |
| „Unverified“ auf GitHub | Öffentlicher Schlüssel fehlt in GitHub oder ist nicht korrekt hinterlegt |
| Fehlermeldung beim Commit | SSH-Agent läuft nicht oder Pfad stimmt nicht |

Hilfreiche Prüf-Befehle:

```
git config --get commit.gpgsign
git config --get gpg.format
git config --get user.signingkey
```

Optional: SSH-Signatur global aktivieren

Wenn du deine Commits dauerhaft signieren möchtest, kannst du die Einstellungen global aktivieren:

```
git config --global commit.gpgsign true
git config --global gpg.format ssh
git config --global user.signingkey ~/.ssh/id_ed25519.pub
```

Das wirkt sich auf **alle** zukünftigen Repositories aus.

Abschließender Hinweis

Wenn ein Projekt Branch-Regeln wie „*Require signed commits*“ verwendet, **musst** du signieren, um Commits pushen zu können. Diese Anleitung hilft dir, diese Voraussetzung sicher zu erfüllen.

⚠ Falls du die Commit-Signierung konfiguriert hast und noch unsichere Commits offen hast, musst du diese zurücksetzen oder nachsignieren. Andernfalls wird der nächste Push auf deinen Branch blockiert, sofern dieser „*Require signed commits*“ aktiviert hat.