SSH-Keys für GitHub – Sichere Authentifizierung einrichten □□

Wenn du regelmäßig mit GitHub arbeitest, wirst du schnell merken, dass die Eingabe von Benutzername und Passwort bei jedem `push` oder `pull` lästig wird. Außerdem hat GitHub die Passwort-Authentifizierung über HTTPS für Git-Operationen inzwischen eingeschränkt. Die elegante und sichere Lösung heißt **SSH-Key-Authentifizierung**. In diesem Kapitel erfährst du, was SSH-Keys sind, warum sie für GitHub so wichtig sind und wie du sie Schritt für Schritt einrichtest.

Was sind SSH-Keys überhaupt?

SSH steht für **Secure Shell** und ist ein Protokoll für verschlüsselte Verbindungen zwischen Computern. Ein SSH-Key ist ein **kryptografisches Schlüsselpaar**, das aus zwei Teilen besteht:

- **Privater Schlüssel** (*Private Key*): Dieser bleibt **ausschließlich auf deinem Computer** und wird niemals weitergegeben. Er ist wie der Schlüssel zu deiner Haustür – nur du besitzt ihn.
- **Öffentlicher Schlüssel** (*Public Key*): Diesen gibst du an Dienste wie GitHub weiter. Er ist wie ein spezielles Schloss, das nur mit deinem privaten Schlüssel geöffnet werden kann.

Das Geniale an diesem System: Wenn du dich mit GitHub verbindest, beweist dein Computer durch den privaten Schlüssel, dass er zu dem öffentlichen Schlüssel gehört, der bei GitHub hinterlegt ist – **ohne dass ein Passwort übertragen wird**. Selbst wenn jemand die Verbindung abhören würde, könnte er sich nicht als du ausgeben.

```
flowchart LR
    A["□□Privater Schluessel\nBleibt auf deinem PC"] --- B["□□
Kryptografisches\nSchluesselpaar"]
    B --- C["□□Oeffentlicher Schluessel\nWird bei GitHub hinterlegt"]

    D["□□Dein Computer"] -->|Authentifizierung\nohne Passwort| E["☁□ GitHub Server"]
```

Warum brauche ich SSH-Keys für GitHub?

Es gibt mehrere gute Gründe, warum du SSH-Keys einrichten solltest:

1. **Sicherheit**
   SSH-Keys sind deutlich sicherer als Passwörter. Ein typischer SSH-Key hat eine Länge von 256 bis 4096 Bit – das ist praktisch unknackbar. Selbst wenn jemand deinen Netzwerkverkehr mitliest, kann er deinen privaten Schlüssel nicht rekonstruieren.
2. **Komfort im Alltag**
   Nach der einmaligen Einrichtung musst du bei `git push` oder `git pull` **keine Zugangsdaten mehr eingeben**. Der SSH-Agent auf deinem Computer übernimmt die Authentifizierung automatisch im Hintergrund.
3. **GitHub hat HTTPS-Passwörter eingeschränkt**
   Seit August 2021 akzeptiert GitHub keine einfachen Passwörter mehr für Git-Operationen über HTTPS. Stattdessen müsstest du einen *Personal Access Token* verwenden – der ist allerdings lang, unpraktisch und muss regelmäßig erneuert werden. SSH-Keys sind die elegantere Alternative.
4. **Mehrere Geräte, eine Identität**
   Du kannst auf jedem deiner Computer einen eigenen SSH-Key erstellen und bei GitHub hinterlegen. So behältst du die Kontrolle darüber, welche Geräte Zugriff auf deine Repositories haben, und kannst einzelne Keys jederzeit widerrufen.

SSH-Key erstellen – Schritt für Schritt

Die Einrichtung unterscheidet sich je nach Betriebssystem nur minimal. Ich zeige dir den Prozess für **Windows**, **macOS** und **Linux** – die Befehle sind weitgehend identisch.

Voraussetzungen prüfen

Bevor du loslegst, stelle sicher, dass du ein Terminal öffnen kannst:

- **Windows**: Nutze **Git Bash** (wird mit Git installiert), **PowerShell** oder das **Windows Terminal**.
- **macOS**: Öffne die **Terminal**-App (findest du unter Programme → Dienstprogramme).
- **Linux**: Öffne dein bevorzugtes Terminal.

Schritt 1: Prüfen, ob bereits ein SSH-Key existiert

Es ist möglich, dass du schon einen SSH-Key hast, ohne es zu wissen. Prüfe das mit folgendem Befehl:

```
ls -la ~/.ssh
```

Wenn du Dateien wie `id_rsa`, `id_ed25519` oder ähnliche siehst (jeweils mit und ohne `.pub`-Endung), hast du bereits ein Schlüsselpaar. Du kannst dieses verwenden oder ein neues erstellen.

> **Tipp:** Wenn der Ordner `~/.ssh` nicht existiert, ist das kein Problem – er wird beim Erstellen eines Keys automatisch angelegt.

Schritt 2: Neuen SSH-Key generieren

GitHub empfiehlt den modernen **Ed25519-Algorithmus**, der sicherer und schneller ist als das ältere RSA. Führe folgenden Befehl aus und ersetze die E-Mail-Adresse durch **deine GitHub-E-Mail**:

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

Was passiert jetzt?

1. **Speicherort wählen**
   Du wirst gefragt, wo der Key gespeichert werden soll:
   ```
   Enter file in which to save the key (/home/deinname/.ssh/id_ed25519):
   ```
   Drücke einfach **Enter**, um den Standardpfad zu akzeptieren. Das ist in den allermeisten Fällen die richtige Wahl.
2. **Passphrase festlegen**
   Anschließend wirst du nach einer **Passphrase** gefragt:

```
Enter passphrase (empty for no passphrase):
```

Hier hast du zwei Optionen:

- **Mit Passphrase** *(empfohlen)*: Gib ein sicheres Passwort ein. Falls jemand Zugriff auf deinen Computer bekommt, ist der Key trotzdem geschützt.
- **Ohne Passphrase**: Drücke einfach Enter. Der Key funktioniert dann ohne zusätzliche Eingabe, ist aber weniger sicher.

> **Empfehlung:** Nutze eine Passphrase! Mit dem SSH-Agent (siehe weiter unten) musst du sie trotzdem nur einmal pro Sitzung eingeben.

3. **Bestätigung**
   Nach erfolgreicher Erstellung siehst du eine Ausgabe wie diese:

```
Your identification has been saved in /home/deinname/.ssh/id_ed25519
Your public key has been saved in /home/deinname/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:aBcDeFgHiJkLmNoPqRsTuVwXyZ1234567890 deine-email@beispiel.de
```

Du hast jetzt zwei neue Dateien:

- `~/.ssh/id_ed25519` – dein **privater Schlüssel** (niemals teilen!)
- `~/.ssh/id_ed25519.pub` – dein **öffentlicher Schlüssel** (kommt zu GitHub)

Schritt 3: SSH-Agent starten und Key hinzufügen

Der **SSH-Agent** ist ein Hintergrundprogramm, das deine SSH-Keys verwaltet und bei Bedarf automatisch verwendet. So musst du die Passphrase nicht bei jeder Git-Operation eingeben.

Auf macOS und Linux:

```
eval "$(ssh-agent -s)"
```

Du siehst eine Ausgabe wie `Agent pid 12345` – der Agent läuft jetzt.

Auf Windows (Git Bash):

```
eval "$(ssh-agent -s)"
```

Der Befehl ist identisch. In PowerShell kann es etwas anders sein – Git Bash ist hier die einfachere Wahl.

Key zum Agent hinzufügen:

```
ssh-add ~/.ssh/id_ed25519
```

Falls du eine Passphrase gesetzt hast, wirst du jetzt danach gefragt. Nach der Eingabe ist der Key geladen und du musst die Passphrase erst wieder eingeben, wenn du deinen Computer neu startest oder dich ab- und anmeldest.

Öffentlichen Schlüssel bei GitHub hinterlegen

Jetzt kommt der entscheidende Schritt: Du musst GitHub deinen öffentlichen Schlüssel mitteilen, damit GitHub weiß, dass Verbindungen mit deinem privaten Schlüssel vertrauenswürdig sind.

Schritt 1: Öffentlichen Schlüssel kopieren

Zeige den Inhalt deines öffentlichen Schlüssels an:

```
cat ~/.ssh/id_ed25519.pub
```

Du siehst eine lange Zeile, die ungefähr so aussieht:

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIG... (viele Zeichen) ...deine-email@beispiel.de
```

Kopiere diese gesamte Zeile in die Zwischenablage – inklusive `ssh-ed25519` am Anfang und der E-Mail am Ende.

> □□ **Tipp für macOS:** Du kannst den Key direkt in die Zwischenablage kopieren mit:
>
> ```
> pbcopy < ~/.ssh/id_ed25519.pub
> ```
>
> **Auf Windows (Git Bash):**

```
clip < ~/.ssh/id_ed25519.pub
```

Schritt 2: Key in GitHub einfügen

1. Öffne github.com und melde dich an.
2. Klicke auf dein **Profilbild** oben rechts und wähle **Settings**.
3. In der linken Seitenleiste findest du den Bereich **Access** – klicke dort auf **SSH and GPG keys**.
4. Klicke auf den grünen Button **New SSH key**.
5. Fülle die Felder aus:
   - **Title**: Gib einen beschreibenden Namen ein, z. B. „MacBook Pro Arbeit" oder „Windows Desktop". So weißt du später, welcher Key zu welchem Gerät gehört.
   - **Key type**: Lass die Auswahl auf **Authentication Key** (Standard).
   - **Key**: Füge hier den kopierten öffentlichen Schlüssel ein.
6. Klicke auf **Add SSH key**.
7. GitHub fragt dich möglicherweise nach deinem Passwort zur Bestätigung – gib es ein.

Dein SSH-Key ist jetzt bei GitHub registriert! □□

Verbindung testen

Bevor du loslegst, solltest du prüfen, ob alles funktioniert:

```
ssh -T git@github.com
```

Beim **ersten Mal** wirst du gefragt, ob du dem Server vertraust:

```
The authenticity of host 'github.com (IP-Adresse)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Gib **yes** ein und drücke Enter. Wenn alles klappt, siehst du:

```
Hi dein-username! You've successfully authenticated, but GitHub does not provide shell access.
```

Diese Meldung bedeutet: **Es funktioniert!** Du bist authentifiziert, aber GitHub erlaubt keinen direkten Shell-Zugriff – das ist normal und beabsichtigt.

Repositories mit SSH verwenden

Jetzt, wo SSH eingerichtet ist, musst du darauf achten, die **SSH-URL** statt der HTTPS-URL zu verwenden, wenn du Repositories klonst oder verbindest.

Beim Klonen eines Repositories

Wenn du auf GitHub ein Repository öffnest und auf den grünen **Code**-Button klickst, siehst du mehrere Optionen. Wähle **SSH** und kopiere die URL, die so aussieht:

```
git@github.com:dein-username/dein-repository.git
```

Dann klonst du mit:

```
git clone git@github.com:dein-username/dein-repository.git
```

Ein bestehendes Repository auf SSH umstellen

Falls du ein Repository bereits mit HTTPS geklont hast, kannst du die Remote-URL ändern:

```
git remote set-url origin git@github.com:dein-username/dein-repository.git
```

Prüfe die Änderung mit:

```
git remote -v
```

Du solltest jetzt SSH-URLs sehen:

```
origin  git@github.com:dein-username/dein-repository.git (fetch)
origin  git@github.com:dein-username/dein-repository.git (push)
```

SSH in PhpStorm konfigurieren

PhpStorm verwendet normalerweise automatisch die SSH-Konfiguration deines Systems. Trotzdem lohnt es sich, die Einstellungen zu prüfen:

1. Öffne **File → Settings** (auf macOS: **PhpStorm → Settings**).
2. Navigiere zu **Version Control → Git**.
3. Stelle sicher, dass **SSH executable** auf **Native** oder **Built-in** steht. Die Option **Native** nutzt die SSH-Installation deines Systems inklusive des SSH-Agents – das ist meist die beste Wahl.
4. Unter **Version Control → GitHub** kannst du deinen GitHub-Account hinzufügen. Wähle hier **Log In via GitHub** oder, falls du Token bevorzugst, die entsprechende Option. Für Git-Operationen über SSH ist dieser Schritt optional, aber für einige PhpStorm-Features wie das Erstellen von Pull Requests nützlich.

Wenn du jetzt in PhpStorm ein Repository mit SSH-URL klonst oder pushst, sollte alles reibungslos funktionieren – ohne Passworteingabe.

Häufige Probleme und Lösungen

„Permission denied (publickey)"

Diese Fehlermeldung bedeutet, dass GitHub deinen SSH-Key nicht erkennt. Mögliche Ursachen:

- **Key nicht zum SSH-Agent hinzugefügt**: Führe `ssh-add ~/.ssh/id_ed25519` erneut aus.
- **Falscher Key bei GitHub hinterlegt**: Überprüfe auf GitHub unter *Settings → SSH and GPG keys*, ob der richtige öffentliche Schlüssel eingetragen ist.
- **Falscher Dateiname**: Wenn du den Key unter einem anderen Namen gespeichert hast, musst du diesen explizit angeben.

SSH-Agent startet nicht automatisch

Auf Windows kann es sein, dass der SSH-Agent-Dienst nicht läuft. Öffne PowerShell als Administrator und führe aus:

```
Get-Service ssh-agent | Set-Service -StartupType Automatic
Start-Service ssh-agent
```

Auf macOS kannst du den Key dauerhaft im Agent speichern, indem du diese Zeilen zu `~/.ssh/config` hinzufügst (erstelle die Datei, falls sie nicht existiert):

```
Host github.com
  AddKeysToAgent yes
  UseKeychain yes
  IdentityFile ~/.ssh/id_ed25519
```

Mehrere GitHub-Accounts

Falls du mehrere GitHub-Accounts hast (z. B. privat und beruflich), kannst du für jeden einen eigenen Key erstellen und in `~/.ssh/config` konfigurieren:

```
# Privater Account
Host github.com-privat
  HostName github.com
  User git
  IdentityFile ~/.ssh/id_ed25519_privat

# Beruflicher Account
Host github.com-arbeit
  HostName github.com
  User git
  IdentityFile ~/.ssh/id_ed25519_arbeit
```

Beim Klonen verwendest du dann den jeweiligen Host:

```
git clone git@github.com-privat:username/repo.git
git clone git@github.com-arbeit:firmenname/repo.git
```

Zusammenfassung

SSH-Keys sind die **sichere und komfortable Methode**, um dich bei GitHub zu authentifizieren. Nach der einmaligen Einrichtung sparst du dir die ständige Eingabe von Zugangsdaten und profitierst von einer deutlich höheren Sicherheit. Hier nochmal die wichtigsten Schritte im Überblick:

```
flowchart TD
    A["1. SSH-Key generieren\nssh-keygen -t ed25519 -C ..."] --> B["2. SSH-Agent starten\nund
Key hinzufuegen"]
```

```
    B --> C["3. Oeffentlichen Key kopieren\ncat ~/.ssh/id_ed25519.pub"]
    C --> D["4. Key bei GitHub einfuegen\nSettings - SSH Keys"]
    D --> E["5. Verbindung testen\nssh -T git@github.com"]
    E --> F["6. SSH-URLs verwenden\ngit@github.com:user/repo.git"]
```

Mit dieser Einrichtung bist du bestens vorbereitet, um deine Repositories sicher und effizient mit GitHub zu synchronisieren – sei es über die Kommandozeile oder direkt aus PhpStorm heraus. □□

Revision #1
Created 2026-06-17 02:03:36 UTC by art10m
Updated 2026-06-17 02:05:53 UTC by art10m