# Git, GitLab und GitHub Übersicht

## Inhaltsverzeichnis


1. Einführung in Git
2. GitLab vs GitHub
3. SSH-Protokoll und Schlüsselerstellung
4. Ein Repository forken
5. Ein Repository klonen
6. Grundlegende Git-Befehle
    - Änderungen hinzufügen und committen
    - Branches erstellen und wechseln


---

### Einführung in Git

**Git** ist ein verteiltes Versionskontrollsystem, das Entwicklern hilft, Änderungen am Quellcode nachzuverfolgen, mit anderen zusammenzuarbeiten und eine Historie aller Änderungen zu speichern. Es ermöglicht Entwicklern, unabhängig zu arbeiten und Änderungen zu integrieren, ohne die Arbeit anderer zu überschreiben.

### GitLab vs GitHub

**GitLab** und **GitHub** sind beide Plattformen zum Hosten von Git-Repositories, weisen jedoch einige Unterschiede auf:

- **GitLab**:
  - Bietet einen kompletten DevOps-Lebenszyklus in einer einzigen Anwendung, einschließlich CI/CD, Projektplanung und Sicherheitstests.
  - Ideal für Unternehmen oder Teams, die integrierte Bereitstellungspipelines und Projektmanagement benötigen.
  - Kann selbst gehostet werden, um Code-Repositories privat zu verwalten.
- **GitHub**:
  - Bekannt für eine starke Community und das Hosten großer Open-Source-Projekte.
  - Bietet umfangreiche Integrationen mit Drittanbieterdiensten und GitHub Actions für CI/CD.
  - Wird hauptsächlich in der Open-Source-Entwicklung genutzt, ist jedoch auch im Unternehmensbereich weit verbreitet.

---

### SSH-Protokoll und Schlüsselerstellung

Um mit Repositories auf GitHub oder GitLab zu interagieren, wird oft SSH (Secure Shell) als sichere Kommunikationsmethode verwendet. SSH-Schlüssel sind notwendig, um den Zugriff zu authentifizieren, ohne jedes Mal ein Passwort eingeben zu müssen.

### Ein SSH-Schlüsselpaar erstellen

1. **Terminal öffnen** (oder Git Bash unter Windows).
2. Führen Sie den folgenden Befehl aus, um einen SSH-Schlüssel zu generieren (verwenden Sie Ihre OTH-E-Mail anstelle von `mm99999@st.oth-regensburg.de`):

   ```
   ssh-keygen -t rsa -b 4096 -C "mm99999@st.oth-regensburg.de"
   ```

3. Drücken Sie `Enter`, um den Schlüssel im Standardverzeichnis zu speichern (normalerweise `~/.ssh/id_rsa`).
4. Sie können ein Passwort festlegen (optional), um die Sicherheit zu erhöhen.

### Ihren SSH-Schlüssel zu GitLab hinzufügen

1. Kopieren Sie den SSH-Schlüssel in die Zwischenablage:

   ```
   cat ~/.ssh/id_rsa.pub
   ```

2. Melden Sie sich bei GitLab an.
3. Gehen Sie zu **Preferences > SSH Keys**.
4. Fügen Sie Ihren SSH-Schlüssel in das Feld **Key** ein und geben Sie ihm einen Namen (zur einfachen Identifizierung).
5. Klicken Sie auf **Add key**, um den Schlüssel zu speichern.

Für GitHub sind die Schritte ähnlich, SSH-Schlüssel werden jedoch unter **Settings > SSH and GPG keys** hinzugefügt.

---

### Ein Repository forken

Forken erstellt eine persönliche Kopie des Repositories eines anderen Benutzers. In GitLab und GitHub ist das Forken nützlich, um Änderungen zu testen, ohne das ursprüngliche Repository zu beeinflussen.

1. Gehen Sie zu dem Repository, das Sie forken möchten.
2. Klicken Sie auf die **Fork**-Option (in GitLab).
3. Nach dem Forken haben Sie eine Kopie des Repositories unter Ihrem eigenen Konto.

### Ein Repository klonen

Nachdem Sie geforkt haben, können Sie das Repository auf Ihren lokalen Rechner klonen, um damit zu arbeiten:

1. Kopieren Sie die SSH-URL des geforkten Repositories.
2. Navigieren Sie in Ihrem Terminal zu dem Verzeichnis, in dem Sie das Repository speichern möchten, und führen Sie dann aus:

   ```
   git clone <ssh-url>
   ```

   Beispiel:

   ```
   git clone git@gitlab.com:your-username/your-forked-repo.git
   ```

---

### Grundlegende Git-Befehle

### Änderungen hinzufügen und committen

1. **Dateien zum Staging-Bereich hinzufügen**: Bevor Sie committen, müssen Sie Änderungen zum Staging-Bereich hinzufügen.

   ```
   git add <file>      # Fügt eine bestimmte Datei hinzu
   git add .           # Fügt alle Änderungen im aktuellen Verzeichnis hinzu
   ```

2. **Änderungen committen**: Sobald Dateien gestaged sind, committen Sie sie mit einer Nachricht.

   ```
   git commit -m "Ihre Commit-Nachricht hier"
   ```

**Branches erstellen und wechseln** Branches werden verwendet, um verschiedene Entwicklungslinien innerhalb eines Projekts zu trennen.

1. **Einen neuen Branch erstellen**:

   ```
   git branch <neuer-branch-name>
   ```

2. **Zum neuen Branch wechseln**:

   ```
   git checkout <neuer-branch-name>
   ```

   Alternativ können Sie einen neuen Branch erstellen und direkt wechseln:

   ```
   git checkout -b <neuer-branch-name>
   ```

### Änderungen pushen

1. **In einen bestimmten Branch pushen**:

   ```
   git push origin <branch-name>
   ```