

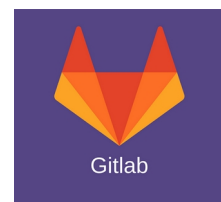


## Gitlab-CE-Server unter Linux installieren und betreiben

### Vorwort zur Anleitung:

Diese Anleitung hat einen hohen Schwierigkeitsgrad. Installation von Linux und Kenntnisse in der Befehlszeile sind vorausgesetzt.

### Was ist ein Gitlab-CE-Server?



Ein **GitLab CE Server** ist ein Server, auf dem die **Community Edition** (CE) der GitLab-Software installiert und betrieben wird. GitLab ist eine DevOps-Plattform, die Entwicklern hilft, Code zu verwalten und zu entwickeln.

#### GitLab CE (Community Edition)

- **Open Source:** Die Community Edition ist die kostenlose, Open-Source-Version von GitLab. Sie bietet Funktionen für die Versionskontrolle und DevOps-Prozesse..
- **Funktionsumfang:** GitLab CE umfasst Funktionen wie:
  - Git-Repository-Management
  - Code-Reviews
  - CI/CD-Pipelines (mit begrenztem Funktionsumfang im Vergleich zur kostenpflichtigen Version)
  - Issue-Tracking
  - Merge-Requests
  - Wiki und Snippets

#### Einsatzmöglichkeiten eines GitLab CE Servers

- **Entwicklerteams** können ihre Projekte und Code-Repositorys auf diesem Server verwalten.
- **CI/CD-Prozesse** für automatisierte Tests und Deployments werden auf dem GitLab CE Server eingerichtet.
- **Private Git-Instanzen:** Unternehmen oder Einzelpersonen, die GitLab für ihre eigene Infrastruktur verwenden möchten, ohne eine kommerzielle Lizenz (Enterprise Edition) zu benötigen, können GitLab CE auf einem eigenen Server betreiben.

Zusammengefasst ist ein GitLab CE Server einfach eine selbstgehostete Installation von GitLab in der Community Edition. Er kann auf einer Vielzahl von Betriebssystemen installiert werden und ermöglicht es Teams, GitLab mit den grundlegenden Funktionen zu nutzen.

## Vorbereitungen / Anforderungen:

|   |  |
|---|--|
| <b>Anforderungen</b> <ul style="list-style-type: none"><li>• <b>Schwierigkeitsgrad-Installation:</b> Hoch</li><li>• <b>Schwierigkeitsgrad-Anwendung:</b> Hoch</li><li>• <b>Erforderliche Kenntnisse:</b> Linux-Befehlszeile</li></ul> <b>Systemvoraussetzungen:</b> <ul style="list-style-type: none"><li>• <b>Basis:</b> Debian</li><li>• <b>Betriebssystem:</b> Ubuntu Server 22.04 (Headless)</li><li>• <b>Festplattenspeicher:</b> Mindestens 17 GB</li><li>• <b>CPU:</b> Mindestens 2 CPU</li><li>• <b>RAM:</b> Mindestens 5 GB</li><li>• <b>Internetverbindung:</b> Verfügbar</li></ul> | Die Systemanforderungen wurden in einer virtualisierten Umgebung getestet.<br><br><b>Tipp:</b><br><br>Wenn du mit Virtualisierung, z.B. ProxMox, vertraut bist, kannst du deine eigenen Server effizient und stromsparend betreiben!<br><br><b>Getestet auf:</b><br><br>Ubuntu-Server 22.04<br>Ubuntu-Server 24.10 |
|---|--|

|   |   |
|---|---|
| <b>Installationsablauf:</b> <ol style="list-style-type: none"><li>1. Durchführung von System-Upgrade</li><li>2. Statische IP-Adresse festlegen und anwenden</li><li>3. (Optional) Hostnamen festlegen</li><li>4. Installation und Einstellungen</li><li>5. Systembedienung</li><li>6. Resümee</li><li>7. GitLab-User anlegen</li><li>8. GitLab-Projekte anlegen</li><li>9. GitLab-Projekt klonen (Push&amp;Pull)</li><li>10. Resümee</li><li>11. Schlussbemerkung</li></ol> | Optionale Schritte können weggelassen werden, da sie lediglich unterstützende Funktionen bieten, wie zum Beispiel die Zusammenfassung von Servern in einem Rechenzentrum.<br><br>Für den Betrieb in einem Rechenzentrum sollten die optionalen Abläufe jedoch aus Gründen des beruflichen Stolzes in Betracht gezogen werden. |
|---|---|

## 1. Durchführung von System-Upgrade

Vor jeder Installation ist ein System-Update erforderlich:

|  |  |
|--|--|
| <code>\$&gt; ssh DEINUSER@192.168.1.X</code> | Stelle eine SSH-Verbindung zum Server her, um Remote-Operationen durchzuführen.  |
| <code>\$&gt; sudo apt update</code>          | Aktualisiere die Paketquellen, um sicherzustellen, dass du die neuesten Versionen der Pakete erhältst  |
| <code>\$&gt; sudo apt upgrade -y</code>      | Starte das System-Upgrade und verwende dabei die Option <code>--yes-to-all</code> , um alle Bestätigungsabfragen automatisch zu beantworten. |
| <code>\$&gt; sudo apt autoclean</code>       | Entferne Pakete, die nicht mehr benötigt werden, um Speicherplatz zu sparen und das System zu optimieren.                                    |
| <code>\$&gt; sudo apt autoremove</code>      | Bereinige das System von überflüssigen Abhängigkeiten, die nach Paket-Deinstallationen übrig geblieben sind.                                 |

## 2. Statische IP-Adresse festlegen und anwenden

Ein Gitlab-Server benötigt zwingend eine statische IP-Adresse, weil Client-System ausfallen, wenn sich diese Adresse ändert.

Es besteht die Möglichkeit, sich selbst vom Server auszuschließen, wenn die IP-Adresse nicht korrekt konfiguriert ist.

|  |   |
|--|---|
| <code>\$&gt; ip addr</code>  | Verwende den Befehl <i>ip addr</i> , um alle Netzwerk-Adapter und deren Namen anzuzeigen.   |
| <code>\$&gt; sudo su</code>  | Stelle sicher, dass du über Root-Rechte verfügst, um Änderungen vorzunehmen.  |
| <code>\$&gt; cd /etc/netplan</code>  | Gehe in das Verzeichnis, in dem sich die Netzwerk-Konfigurationsdateien befinden.   |
| <code>\$&gt; ls</code>   | Zeige mit dem Befehl <i>ls</i> alle relevanten YAML-Dateien im Verzeichnis an.  |
| <code>\$&gt; for i in \$( ls ); do mv \$i \$i.bak; done</code>   | Sichere alle bestehenden Konfigurationsdateien, bevor du Änderungen vornimmst.  |
| <code>\$&gt; touch /etc/netplan/01_static_ip.yaml</code>   | Erstelle eine neue Konfigurationsdatei  |
| <code>\$&gt; nano /etc/netplan/01_static_ip.yaml</code>  | Öffne die Datei mit einem Texteditor  |
| <pre># Beispiel-Konfiguration von Linux-Schweiz.ch network:   version: 2   renderer: networkd   ethernets:     ens18: #Edit this line according to your network            interface name.       dhcp4: no       addresses:         - 192.168.1.150/24       gateway4: 192.168.1.1       nameservers:         addresses:           - 8.8.8.8           - 8.8.4.4</pre> | <p>(Copy-Paste)</p> <p>Ersetze <b>ens18</b> durch den Namen deines Netzwerk-Adapters (siehe Ausgabe von <i>ip addr</i>).</p> <p>Ändere die <b>IP-Adresse</b> und die <b>Netzmaske</b> nach Bedarf.</p> <p>Passen die <b>DNS-Adressen</b> an, falls erforderlich.</p> <p>Überprüfe und passe ggf. die <b>Routen (Gateway)</b> an.</p> <p><b>Speichern und Verlassen</b></p> <ul style="list-style-type: none"><li>• Speichern der Datei: Ctrl + O</li><li>• Editor verlassen: Ctrl + X</li></ul> |
| <code>\$&gt; netplan generate &amp;&amp; netplan apply</code>  | Wende die neuen Netzwerkeinstellungen an  |
| --- Verbindungsunterbruch - IP wird neu gesetzt  |   |



Der Server wechselt nun zu einer neuen IP-Adresse, was dazu führt, dass deine aktuelle SSH-Sitzung unterbrochen wird.

### 3. (Optional) Hostnamen festlegen

Durch das Ändern des Hostnamens der Server-Station vergibst du einen eindeutigen Namen für das System.

In unserem Fall könnte der Hostname beispielsweise „S5-Gitlab-Server“ lauten.

Die Hosts-Datei wird verwendet, um auf Server-Ebene IP-Adressen Namen zuzuordnen, die intern genutzt werden. Wenn du den Hostnamen änderst, muss auch der entsprechende Eintrag in der Hosts-Datei aktualisiert werden. Bei produktiven Systemen sollte diese Kleinigkeit unbedingt beachtet werden, da sie ein Zeichen von Berufsethos und Professionalität ist. Für Tests und Experimente ist dies jedoch nicht zwingend erforderlich.

Der neue Hostname muss zwingen der gleiche sein wie auch im Hosts-Files.

|   |   |
|---|---|
| <code>\$&gt; ssh DEINUSER@192.168.1.X</code>  | Stelle eine SSH-Verbindung zum Server über die neue IP-Adresse her.   |
| <code>\$&gt; sudo hostnamectl set-hostname <b>DEINHOSTNAME</b></code>   | Setze einen neuen Hostnamen.  |
| <code>\$&gt; sudo nano /etc/hosts</code>  | Bearbeite die Datei /etc/hosts oder die entsprechende Datei, in der der Hostname definiert ist.   |
| <code>127.0.0.1 localhost</code><br><code>127.0.1.1 <b>DEINHOSTNAME</b></code><br><br><code># The following lines are desirable for IPv6 capable hosts</code><br><code>::1 ip6-localhost ip6-loopback</code><br><code>fe00::0 ip6-localnet</code><br><code>ff00::0 ip6-mcastprefix</code><br><code>ff02::1 ip6-allnodes</code><br><code>ff02::2 ip6-allrouters</code> | Ändere ausschließlich den markierten Bereich, der den Hostnamen enthält. Achte darauf, dass du keine anderen Einträge unbeabsichtigt veränderst.<br><br><b>Speichern und Verlassen</b> <ul style="list-style-type: none"><li>• Speichern der Datei: Ctrl + O</li><li>• Editor verlassen: Ctrl + X</li></ul> |
| <code>\$&gt; sudo reboot</code>   | Starte den Server neu, damit die Änderungen wirksam werden.   |

### 4. Installation und Einstellungen

GitLab-CE ist keine Anwendung aus dem Repository von Ubuntu-Linux. Falls die Installation fehlschlägt muss die Installations-URL angepasst werden. Universe und Multiverse-Quellen müssen im Sources-File aktiviert sein. Mehr unter:

<https://packages.gitlab.com/gitlab/gitlab-ce/install>

|  |   |
|--|---|
| <code>\$&gt; ssh DEINUSER@192.168.1.X</code>   | Stelle eine SSH-Verbindung zum Server über die neue IP-Adresse her. |
| <code>\$&gt; sudo apt install curl -y</code><br><code>\$&gt; sudo apt install openssh-server -y</code><br><code>\$&gt; sudo apt install ca-certificates -y</code><br><code>\$&gt; sudo apt install tzdata -y</code><br><code>\$&gt; sudo apt install perl -y</code><br><code>\$&gt; sudo apt install postfix -y</code> | Unterstützende Pakete installieren                                  |
| <code>\$&gt; cd</code><br><code>\$&gt; mkdir downloads &amp;&amp; cd downloads</code>  | Optional - wechselt in ein anderes Verzeichnis für den Download     |
| <code>\$&gt; curl -O</code><br><code><a href="https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.deb.sh">https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.deb.sh</a></code>  | Installationshilfe für den GitLab-Server herunterladen              |

|                                |                                 |
|--------------------------------|---------------------------------|
| \$> chmod +x script.deb.sh     | Bash-Script ausführbar machen   |
| \$> sudo bash script.deb.sh    | Bash-Script ausführen           |
| \$> sudo apt install gitlab-ce | Installieren des Gitlab-Servers |

Die GitLab-Installation ist eine umfangreiche Software mit großem Dateivolumen, die eher mit einer eigenständigen Betriebssystem vergleichbar ist. Es wird in dieser Anleitung nur das Nötigste zur Lauffähigkeit erläutert.

|  |  |
|--|--|
| \$> sudo nano /etc/gitlab/gitlab.rb  | Einstellungen bearbeiten, externe URL ändern – verwende deine IP-Adresse |
| Erstellen einer lokalen Domain oder einer Weltweiten Domain auf deine IP-Adresse.<br>In der Konfiguration kann auch mit deiner IP verwendet werden, jedoch muss sie zwingend im Format <a href="http://DEINE-IP">http://DEINE-IP</a> sein. |  |
| ...<br>external_url 'http://192.168.1.151'<br>...  |  |
| \$> sudo gitlab-ctl reconfigure  | Neue Einstellung laden   |
| \$> sudo cat /etc/gitlab/initial_root_password   | Initial-Passwort anzeigen  |
| \$> sudo gitlab-rake "gitlab:password:reset[root]"   | Setzen eines neuen Admin-Passworts                                       |
| Das Weblogin wird den Benutzernamen root und dein gewähltes Passwort verlangen.<br>Wähle ein starkes Passwort  |  |
| \$> sudo reboot  | Server-Neustart (Optional)   |

Die Basis-Installation ist nun installiert und kann über das Weblogin, welches deine URL oder IP-Adresse sein wird, erreicht werden. Eine Menge an Einstellungen werden dich im Web-GUI erwarten.

## Hinweise zur Installation

GitLab ist eine leistungsintensive Anwendung, die sich von klassischen Linux-Anwendungen unterscheidet. Viele gängige Verwaltungsbefehle werden hier durch ein eigenes Programm, **gitlab-ctl**, gesteuert.

Daher sollte man vorsichtig sein und nicht willkürlich am Datenstamm herumbasteln. Die Daten und Projekte, die in diesem System gespeichert sind, können nicht einfach kopiert und wiederhergestellt werden. Aus Gründen der Einfachheit empfiehlt es sich daher, immer den gesamten Server zu sichern.

Erfahrungen zeigen, dass es sinnvoller ist, den gesamten Server neu zu starten, anstatt nur die Anwendung lokal zu stoppen und neu zu starten.

## 5. Systembedienung

|  |   |
|--|---|
| <pre>\$&gt; sudo gitlab-ctl restart</pre>                            | Startet Gitlab-CE, um die Zeit-Synchronisation zu aktivieren.   |
| <pre>\$&gt; sudo gitlab-ctl stop</pre>                               | Stoppt Gitlab-CE, wenn du ihn vorübergehend deaktivieren möchtest.  |
| <pre>\$&gt; sudo gitlab-ctl status</pre>                             | Zeigt den Status aller GitLab-Dienste   |
| <pre>\$&gt; sudo nano /etc/gitlab/gitlab.rb</pre>                    | Bearbeitet die Gitlab-Konfigurationsdatei.  |
| <pre>\$&gt; gitlab-ctl --help</pre>                                  | Zeigt die Vielzahl an Steuerungsoptionen  |
| <pre>\$&gt; sudo gitlab-ctl upgrade</pre>                            | Führt ein Update aus  |
| <pre>\$&gt; sudo apt install nmap -y<br/>\$&gt; nmap localhost</pre> | Listet alle offenen Ports auf, um sicherzustellen, dass der GitLab-Dienst ordnungsgemäß kommunizieren kann. |

## 6. Resümee

Der Gitlab-Server ist nicht pflegeleicht und kein Leichtgewicht. Je nach Nutzungszweck ist nicht nur das Installations-Wissen oder Wartungs-Wissen nötig. Im Betrieb sind die Anwender-Kenntnisse und Features höher gewichtet.


## 7. GitLab-User anlegen

Lege auf dem Web-Gui einen neuen Gitlab-User an, der sich vom Root-Account unterscheidet.

Für das erste Login ist der Account «root» zu verwenden.

Anmelden · GitLab

Nicht sicher 192.168.1.111/users/sign\_in



GitLab Community Edition

Benutzername oder primäre E-Mail-Adresse

root

Passwort

.....

[Hast du dein Passwort vergessen?](#)

☐ Login merken

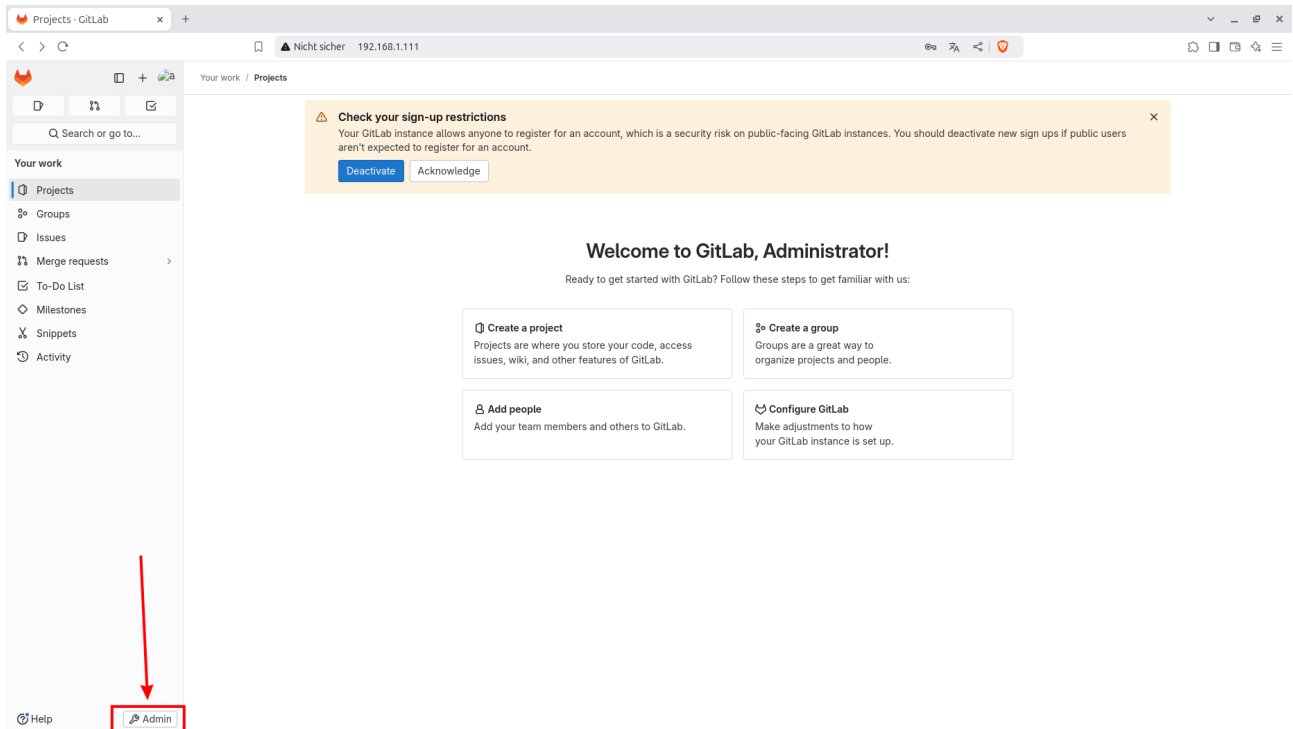
Anmelden

[Noch keine Konto? Jetzt registrieren](#)

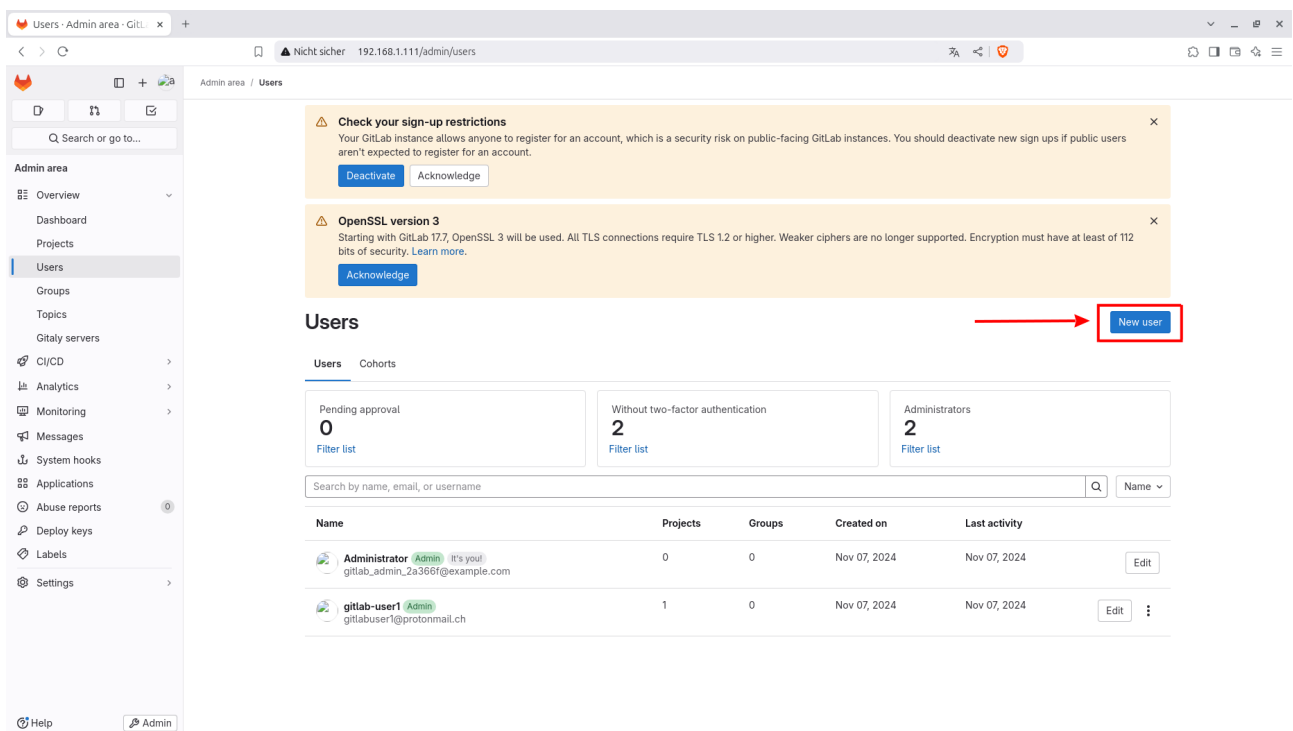
[Erkunden](#) [Hilfe](#) [Über GitLab](#) [Community-Forum](#)

Deutsch

Wechsle auf die Admin-Seite wo du auf eine Nutzer-Übersicht gelangst.



Erstelle einen neuen GitLab-Benutzer:



Wähle passende Credententials und siehe dir die möglichen Einstellungen genau an:

The screenshot shows the 'New User' form in the GitLab Admin area. The form is divided into three main sections: Account, Password, and Access. The 'Account' section has input fields for Name, Username, and Email. The 'Password' section has a button to 'Reset link' and a note that the user will be forced to set a password on first sign in. The 'Access' section has a 'Projects limit' input field set to 100000, and two checkboxes: 'Can create top level group' (checked) and 'Private profile' (unchecked). Below these are 'Access level' options: 'Regular' (selected) and 'Administrator'. At the bottom are 'Create user' and 'Cancel' buttons.

**New user**

**Account**

Name

Username

Email

**Password**

Reset link will be generated and sent to the user. User will be forced to set the password on first sign in.

**Access**

Projects limit

100000

☒ Can create top level group

☐ Private profile

**Access level**

☒ Regular

Regular users have access to their groups and projects.

☐ Administrator

The user has unlimited access to all groups, projects, users, and features.

Create user Cancel

In unserem Fall existiert noch kein Mail-Server, richte für den Benutzer X also ein Passwort ein:

(Wenn ein Mail-Server definiert ist, erhält die Person ein Email mit einem Link. Das Passwort wird dann von der betreffenden Person selber festgelegt.)

The screenshot shows the user profile page for 'gitlabuser2' in the GitLab Admin area. A notification at the top says 'User was successfully created.' The page has tabs for Account, Groups and projects, SSH keys, Identities, and Impersonation Tokens. The 'Account' tab is selected. The profile section shows the user's name, username, email (gitlabuser2@protonmail.ch, Verified), and member since date (Nov 7, 2024 6:36pm). The 'Edit' button is highlighted with a red arrow. The 'Account' section shows details like Name, Username, Email, Email confirmation code, Locked account email verification code, ID, Namespace ID, and Two-factor Authentication (Disabled).

User was successfully created.

**gitlabuser2**

Impersonate New identity Edit

Account Groups and projects SSH keys Identities Impersonation Tokens

gitlabuser2

Profile

Member since Nov 7, 2024 6:36pm

**Account**

Name: gitlabuser2

Username: gitlabuser2

Email: gitlabuser2@protonmail.ch Verified

Email confirmation code last sent at: never

Locked account email verification code last sent at: never

ID: 3

Namespace ID: 4

Two-factor Authentication: Disabled



Auch hier - Sieh dir die möglichen Einstellungen genau an!

The screenshot shows the GitLab Admin interface for editing the user 'gitlabuser2'. The left sidebar contains the 'Admin area' menu with options like Overview, Dashboard, Projects, Users, Groups, Topics, GitLab servers, CI/CD, Analytics, Monitoring, Messages, System hooks, Applications, Abuse reports, Deploy keys, Labels, and Settings. The main content area is titled 'Edit user: gitlabuser2' and contains three sections: 'Account', 'Password', and 'Access'. The 'Account' section has fields for Name (gitlabuser2), Username (gitlabuser2), and Email (gitlabuser2@protonmail.ch). The 'Password' section, highlighted with a red box, has fields for Password and Password confirmation. The 'Access' section has a 'Projects limit' field (100000), checkboxes for 'Can create top level group' (checked) and 'Private profile' (unchecked), and an 'Access level' dropdown set to 'Regular'. At the bottom of the 'Access' section are 'Save changes' and 'Cancel' buttons.

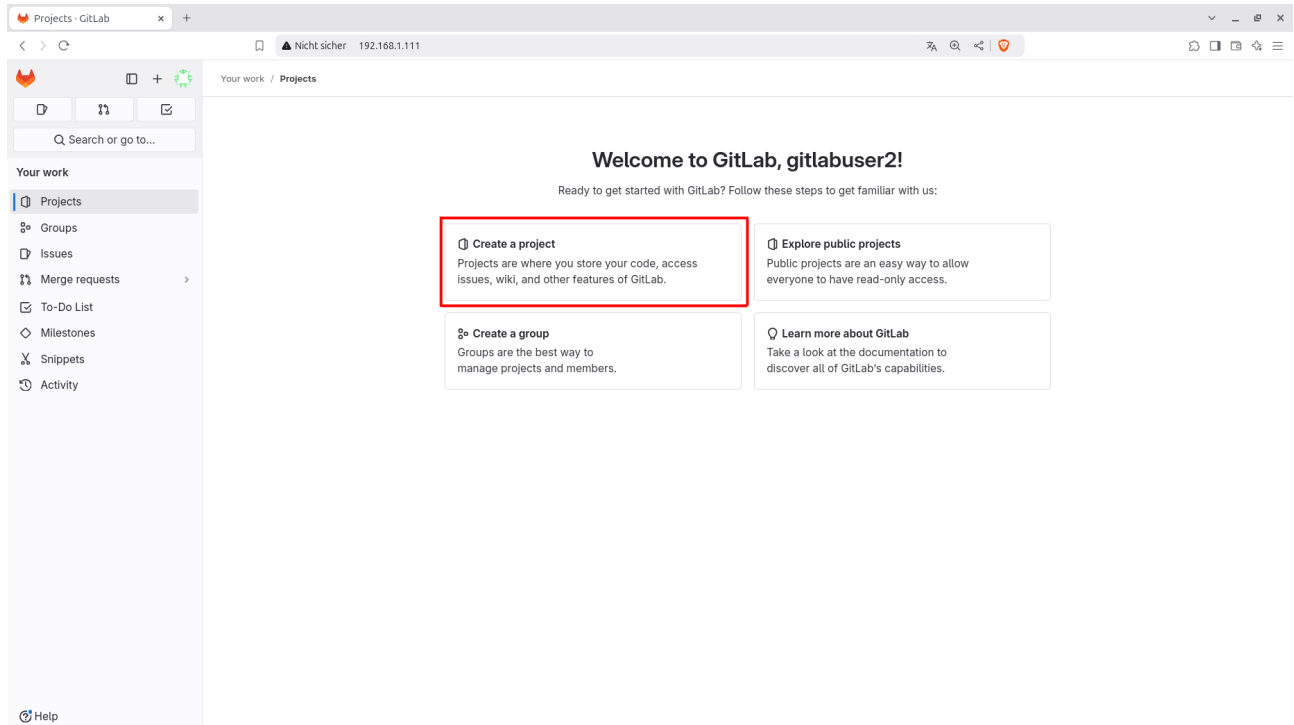
Logge dich mit dem neuen Benutzer ein:

The screenshot shows the GitLab login page. At the top is the GitLab logo and the text 'GitLab Community Edition'. Below this is a form with the following fields: 'Benutzername oder primäre E-Mail-Adresse' (Username or primary email address) with the value 'gitlabuser2', and 'Passwort' (Password) with masked characters. There is a link 'Hast du dein Passwort vergessen?' (Forgot your password?). Below the password field is a checkbox for 'Login merken' (Remember login). At the bottom of the form is an 'Anmelden' (Login) button. Below the button is a link 'Noch keine Konto? Jetzt registrieren' (No account yet? Register now). At the very bottom of the page is a footer with links 'Erkunden' (Explore), 'Hilfe' (Help), 'Über GitLab' (About GitLab), 'Community-Forum' (Community Forum), and a language selector 'Deutsch'.

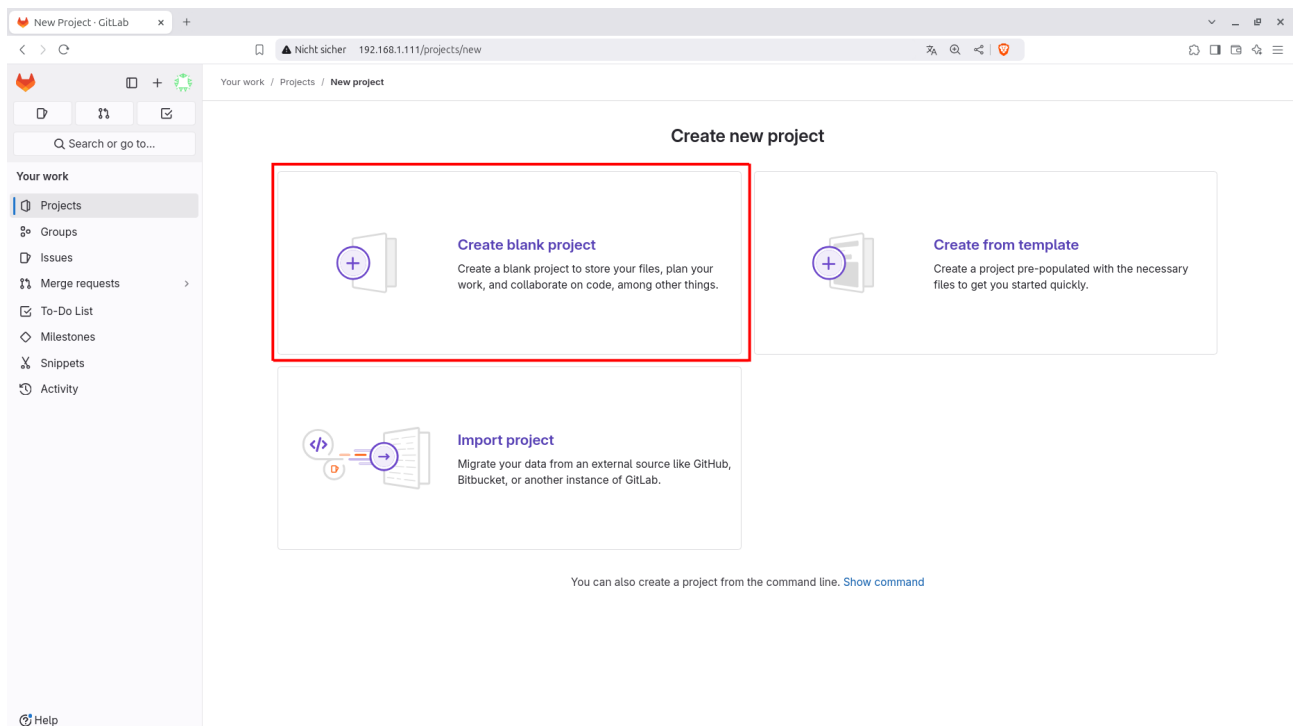
## 8. GitLab-Projekte anlegen

Lege unter dem neuen User ein Projekt an.

Hinweis: ein Projekt kann auch über die Kommando-Zeile angelegt werden, jedoch ist es für Einsteiger einfacher dies als erstes über das Web-GUI zu erledigen:



Immer schön easy bleiben... und ein leeres Projekt erstellen – man kann die Funktionalitäten damit unverfälscht testen und erkunden:



Erstelle ein Projekt nach deinem Geschmack! Sieh dir die möglichen Einstellungen genau an. Wir kratzen nur an der Oberfläche und haben den Server für professionellen Bedarf noch nicht komplett eingerichtet. Wichtig: Vermeide Tippfehler und wähle die Projekt-Namen klug!

The screenshot shows the 'Create blank project' page in GitLab. The left sidebar contains navigation links: 'Your work' (Projects, Groups, Issues, Merge requests, To-Do List, Milestones, Snippets, Activity) and 'Help'. The main content area is titled 'Create blank project' and includes the following fields and options:

- Project name:** A text input field containing 'My awesome project'. Below it, a note states: 'Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.'
- Project URL:** A text input field containing 'http://192.168.1.111/gitlabuser2/'.
- Project slug:** A text input field containing 'my-awesome-project'.
- Visibility Level:** Three radio button options: 'Private' (selected), 'Internal', and 'Public'. Descriptions for each are provided.
- Project Configuration:** Two checkboxes: 'Initialize repository with a README' (checked) and 'Enable Static Application Security Testing (SAST)' (unchecked).

At the bottom, there are 'Create project' and 'Cancel' buttons.

Dein Test-Projekt wurde erfolgreich erstellt. Hier findest du die Projekt-Übersicht:

The screenshot shows the project overview for 'meinlinux-test' in GitLab. The left sidebar contains navigation links: 'Project' (meinlinux-test, Pinned, Issues, Merge requests), 'Manage' (Plan, Code, Build, Secure, Deploy, Operate, Monitor, Analyze, Settings), and 'Help'. The main content area displays the project details:

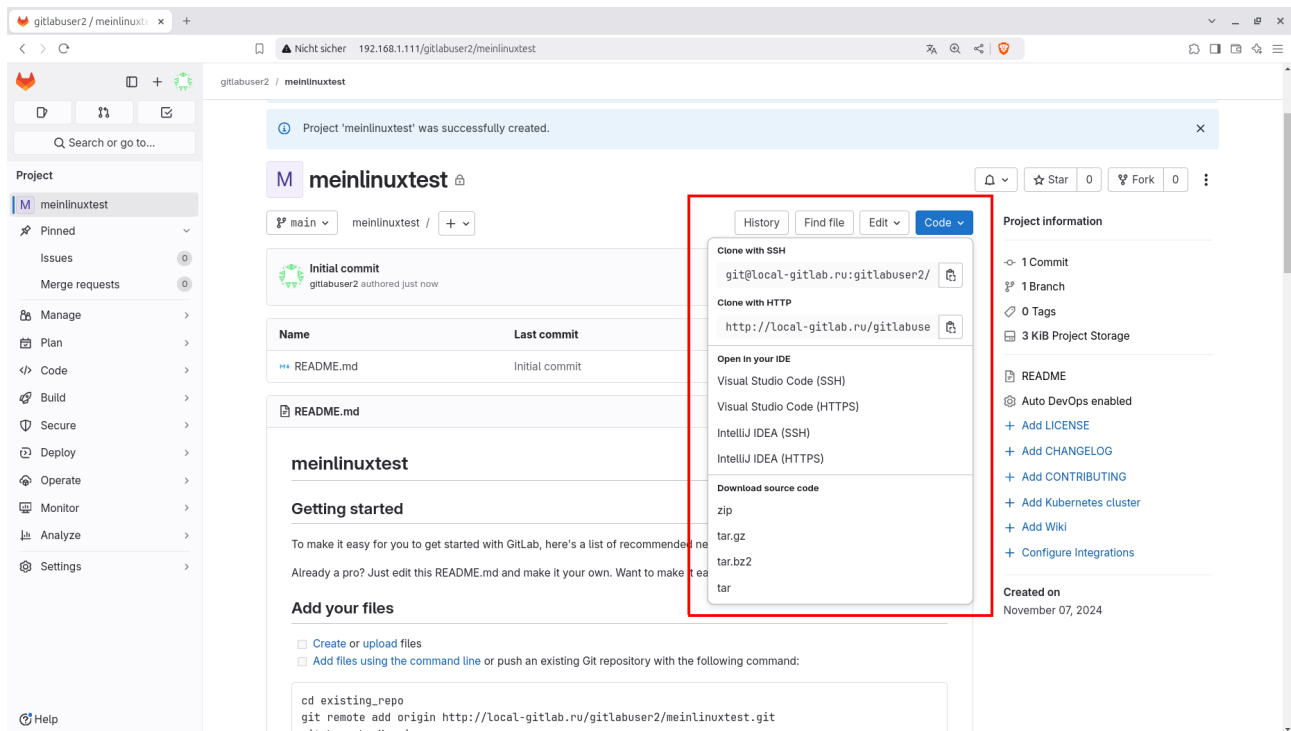
- Project name:** 'meinlinux-test'.
- Initial commit:** 'gitlabuser2 authored just now' with commit hash '5ae8ebb6'.
- Project information:** A table showing the project's history and status.
- Project information sidebar:** A list of project details and links.

| Name      | Last commit    | Last update |
|-----------|----------------|-------------|
| README.md | Initial commit | just now    |

**Project information sidebar:**

- 1 Commit
- 1 Branch
- 0 Tags
- 3 KIB Project Storage
- README
- Auto DevOps enabled
- + Add LICENSE
- + Add CHANGELOG
- + Add CONTRIBUTING
- + Add Kubernetes cluster
- + Add Wiki
- + Configure Integrations

Für weiter Schritte ist es hilfreich unter «Code» die http-Sektion des Projektes zu kopieren:



## 9. GitLab-Projekt klonen (Push&Pull)

|   |   |
|---|---|
| \$> ssh CLIENT1@192.168.1.X   | Stelle eine SSH-Verbindung zum Client über eine IP-Adresse her. |
| \$> sudo apt install git  | Installiere Git auf deinem Client                               |
| \$> mkdir projekte  | Erstelle einen Ordner für deine Projekte                        |
| \$> cd projekte   | Wechsle in das neue Verzeichnis                                 |
| \$> git clone http://DeineIP/User/PROJEKT.git   | Klone dein Git-Repository in das neue Verzeichnis               |
| \$> touch neuedatei1  | Erstelle eine neue Datei  |
| Dein Projekt-Repository ist nun vom Server-Stand abweichend. Dein lokales Projekt hat jetzt einen aktuelleren Status als der GitLab-Server. |   |
| \$> git status  | Zeige den aktuellen Status an                                   |
| \$> git add .   | Füge alle neuen Dateien der Änderung zu                         |
| \$> git commit -m «neue-Datei»  | Kommentiere deine Änderung                                      |
| \$> git status  | Zeige den aktuellen Status an                                   |
| \$> git push  | Uploade «pushe» dein Änderung                                   |
| \$> git status  | Zeige den aktuellen Status an                                   |

GitLab-Projekt von einem anderen eingerichteten Client auf den neuesten Stand bringen  
(einfaches Pull-Beispiel wenn mehrere Benutzer das gleiche Projekt betreuen)

|                             |   |
|-----------------------------|---|
| \$> ssh CLIENT1@192.168.1.Y | Stelle eine SSH-Verbindung zum Client über eine IP-Adresse her. |
|-----------------------------|---|

|                 |  |
|-----------------|--|
| \$> cd projekte | Wechsle in das Projekt-Verzeichnis         |
| \$> git pull    | Downloade «pull» dein Änderung vom Server. |

## 10. Resümee

Wer zum ersten Mal mit GitLab in der Kommandozeile arbeitet, muss einiges an Geduld und Durchhaltevermögen mitbringen – vor allem, wenn man sich das Thema alleine erarbeiten muss. Fehlerbehandlung und unterschiedliche Betriebsmodi wurden in diesem Zusammenhang noch gar nicht angesprochen. Zwar gibt es viele Anleitungen im Internet, doch die Herangehensweisen variieren oft stark. Dieses Thema könnte problemlos eine eigene Anleitung füllen. Mein Tipp: Wenn du dich mit GitLab vertraut machst, geh es spielerisch an – etwa mit einem Testprojekt. Experimentiere, mache Fehler und lerne daraus.

## 11. Schlussbemerkung

Die GitLab-Technologie wurde hier anhand eines einfachen Beispiels vorgestellt. Mit GitLab können Projekte versioniert und geteilt werden – in diesem Fall anschaulich über die Kommandozeile. Viele grafische Entwicklungsumgebungen nutzen ebenfalls GitLab. Dadurch können mehrere Entwickler gleichzeitig an einem Projekt arbeiten und den Versionsstand aktuell halten.

Allerdings erfordert dies eine koordinierte Zusammenarbeit. Mit dieser Technologie hat GitLab nicht nur einen neuen Ansatz in der Softwareentwicklung etabliert, sondern auch eine neue Ideologie des Entwicklungsprozesses geprägt. Die Entwicklung kann nun nicht nur dezentral organisiert werden, sondern es ist auch möglich, in alle Richtungen zu arbeiten.

Entwickler können unabhängig an Projekten arbeiten oder diese mit anderen teilen – unabhängig davon, ob nur ein einzelner Entwickler oder ein ganzes Team beteiligt ist oder ob einfach nur Entwicklungen ausgetauscht werden.

Hier wurde lediglich eines der einfachsten Beispiele sowie die Installation demonstriert. Dabei wurden Fehlerbehandlung und Versionskonflikte noch nicht berücksichtigt.