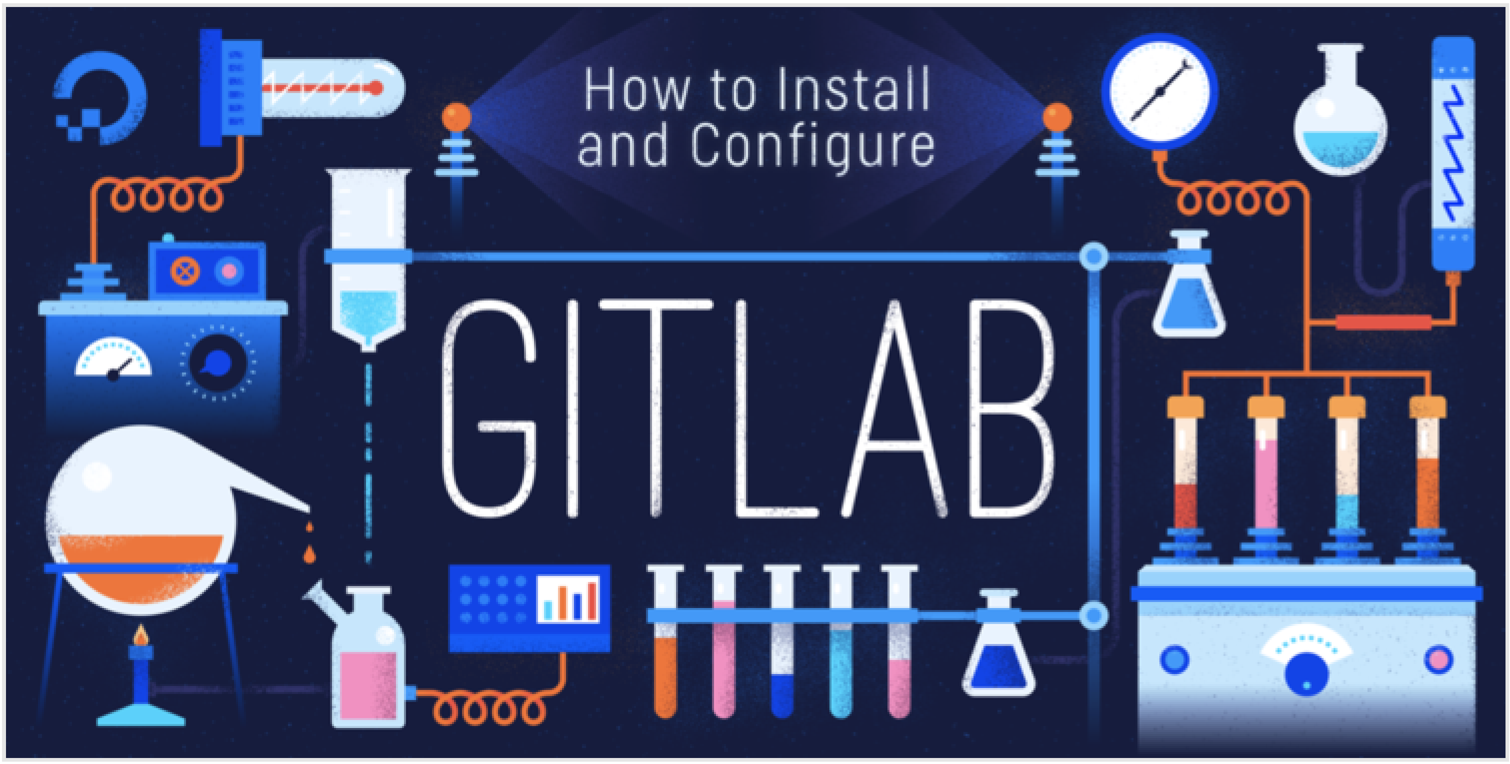
Subscribe Share Contents



How To Install and Configure GitLab on Ubuntu 18.04

Posted July 9, 2018 30.5k APPLICATIONS GIT LET'S ENCRYPT UBUNTU 18.04

8

By: Justin Ellingwood By: Kathleen Juell

Not using **Ubuntu 18.04**? Choose a different version:

Introduction

GitLab CE, or Community Edition, is an open-source application primarily used to host Git repositories, with additional development-related features like issue tracking. It is designed to be hosted using your own infrastructure, and provides flexibility in deploying as an internal repository store for your development team, a public way to interface with users, or a means for contributors to host their own projects.

The GitLab project makes it relatively straightforward to set up a GitLab instance on your own hardware with an easy installation mechanism. In this guide, we will cover how to install and configure GitLab on an Ubuntu 18.04 server.

SCROLL TO TOP

Prerequisites

For this tutorial, you will need:

- An Ubuntu 18.04 server with a non-root sudo user and basic firewall. To set this up, follow our Ubuntu 18.04 initial server setup guide.

The published GitLab hardware requirements recommend using a server with:

- 2 cores
- 8GB of RAM

Although you may be able to get by with substituting some swap space for RAM, it is not recommended. For this guide we will assume that you have the above resources as a minimum.

- A domain name pointed at your server. For more information, see our documentation on how to get started with DNS on DigitalOcean. This tutorial will use the domain name **example.com**.

Step 1 — Installing the Dependencies

Before we can install GitLab itself, it is important to install some of the software that it leverages during installation and on an ongoing basis. Fortunately, all of the required software can be easily installed from Ubuntu's default package repositories.

Since this is our first time using `apt` during this session, we can refresh the local package index and then install the dependencies by typing:

```
$ sudo apt update
$ sudo apt install ca-certificates curl openssh-server postfix
```

You will likely have some of this software installed already. For the `postfix` installation, select **Internet Site** when prompted. On the next screen, enter your server's domain name to configure how the system will send mail.

Step 2 — Installing GitLab

Now that the dependencies are in place, we can install GitLab itself. This is a straightforward process that leverages an installation script to configure your system with the GitLab repositories.

Move into the `/tmp` directory and then download the installation script:

```
$ cd /tmp
$ curl -LO https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/scri
```

SCROLL TO TOP

Feel free to examine the downloaded script to ensure that you are comfortable with the actions it will take. You can also find a hosted version of the script here:

```
$ less /tmp/script.deb.sh
```

Once you are satisfied with the safety of the script, run the installer:

```
$ sudo bash /tmp/script.deb.sh
```

The script will set up your server to use the GitLab maintained repositories. This lets you manage GitLab with the same package management tools you use for your other system packages. Once this is complete, you can install the actual GitLab application with `apt`:

```
$ sudo apt install gitlab-ce
```

This will install the necessary components on your system.

Step 3 — Adjusting the Firewall Rules

Before you configure GitLab, you will need to ensure that your firewall rules are permissive enough to allow web traffic. If you followed the guide linked in the prerequisites, you will have a `ufw` firewall enabled.

View the current status of your active firewall by typing:

```
$ sudo ufw status
```

```
Output
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
```

As you can see, the current rules allow SSH traffic through, but access to other services is restricted. Since GitLab is a web application, we should allow HTTP access. Because we will be taking advantage of GitLab's ability to request and enable a free TLS/SSL certificate from Let's Encrypt, let's also allow HTTPS access.

The protocol to port mapping for HTTP and HTTPS are available in the `/etc/services` file, so we can allow that traffic in by name. If you didn't already have OpenSSH traffic enabled, you should allow that traffic now too:

SCROLL TO TOP

```
$ sudo ufw allow http
$ sudo ufw allow https
$ sudo ufw allow OpenSSH
```

Check the `ufw status` again; you should see access configured to at least these two services:

```
$ sudo ufw status
```

```
Output
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
443/tcp                    ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
80/tcp (v6)                ALLOW       Anywhere (v6)
443/tcp (v6)               ALLOW       Anywhere (v6)
```

The above output indicates that the GitLab web interface will be accessible once we configure the application.

Step 4 — Editing the GitLab Configuration File

Before you can use the application, you need to update the configuration file and run a reconfiguration command. First, open Gitlab's configuration file:

```
$ sudo nano /etc/gitlab/gitlab.rb
```

Near the top is the `external_url` configuration line. Update it to match your domain. Change `http` to `https` so that GitLab will automatically redirect users to the site protected by the Let's Encrypt certificate:

/etc/gitlab/gitlab.rb

```
##! For more details on configuring external_url see:
##! https://docs.gitlab.com/omnibus/settings/configuration.html#configuring-the-external-url-for-gitl
external_url 'https://example.com'
```

Next, look for the `letsencrypt['contact_emails']` setting. This setting defines a list of email addresses that the Let's Encrypt project can use to contact you if there are problems with your domain. It's a good idea to uncomment and fill this out so that you will know of any issues:

SCROLL TO TOP

/etc/gitlab/gitlab.rb

```
letsencrypt['contact_emails'] = ['sammy@example.com']
```

Save and close the file. Run the following command to reconfigure Gitlab:

```
$ sudo gitlab-ctl reconfigure
```

This will initialize GitLab using the information it can find about your server. This is a completely automated process, so you will not have to answer any prompts. The process will also configure a Let's Encrypt certificate for your domain.

Step 5 — Performing Initial Configuration Through the Web Interface

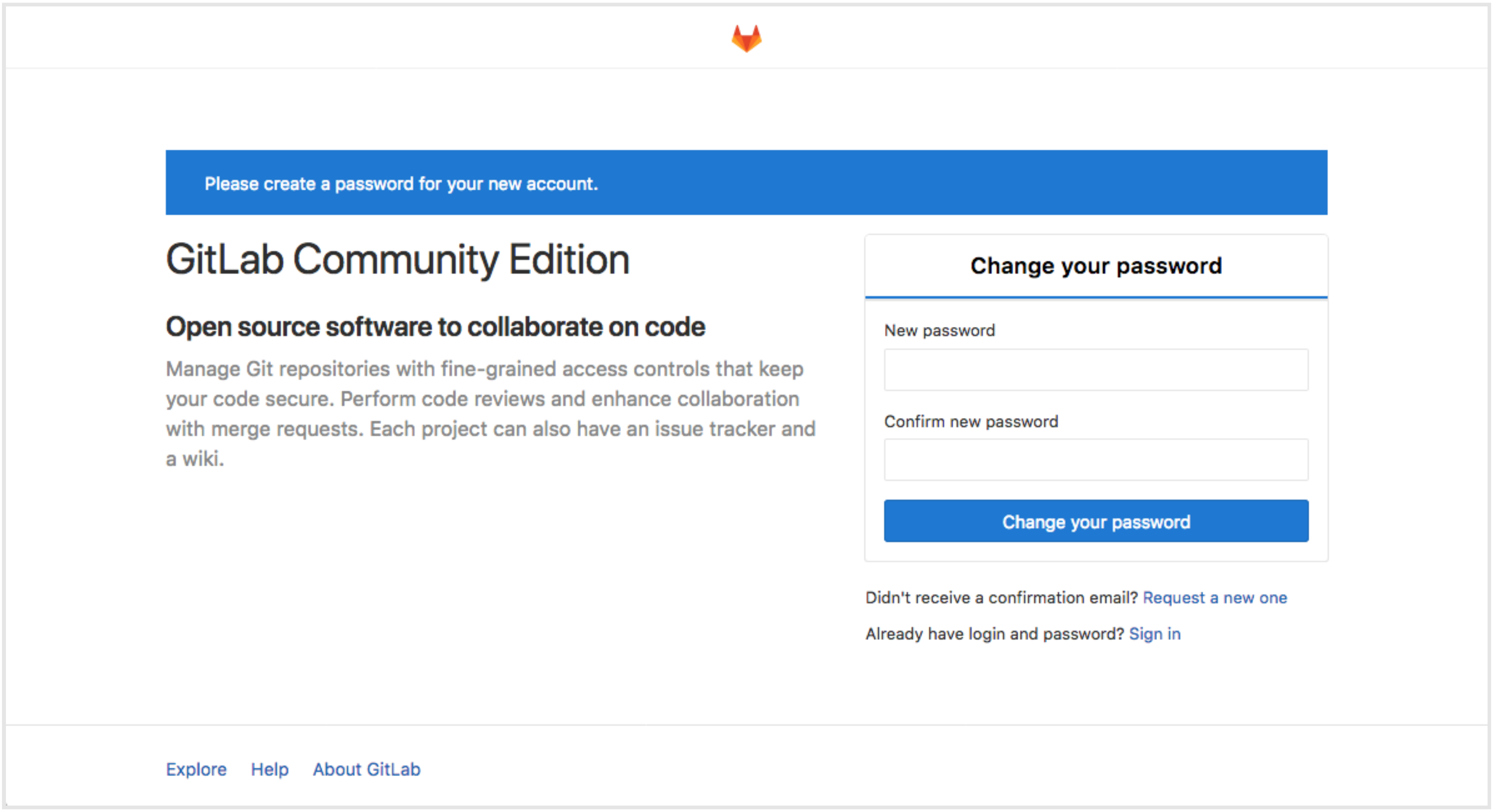
With GitLab running and access permitted, we can perform some initial configuration of the application through the web interface.

Logging In for the First Time

Visit the domain name of your GitLab server in your web browser:

```
https://example.com
```

On your first time visiting, you should see an initial prompt to set a password for the administrative account:



Please create a password for your new account.

GitLab Community Edition

Open source software to collaborate on code

Manage Git repositories with fine-grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

Change your password

New password

Confirm new password

Change your password

Didn't receive a confirmation email? Request a new one

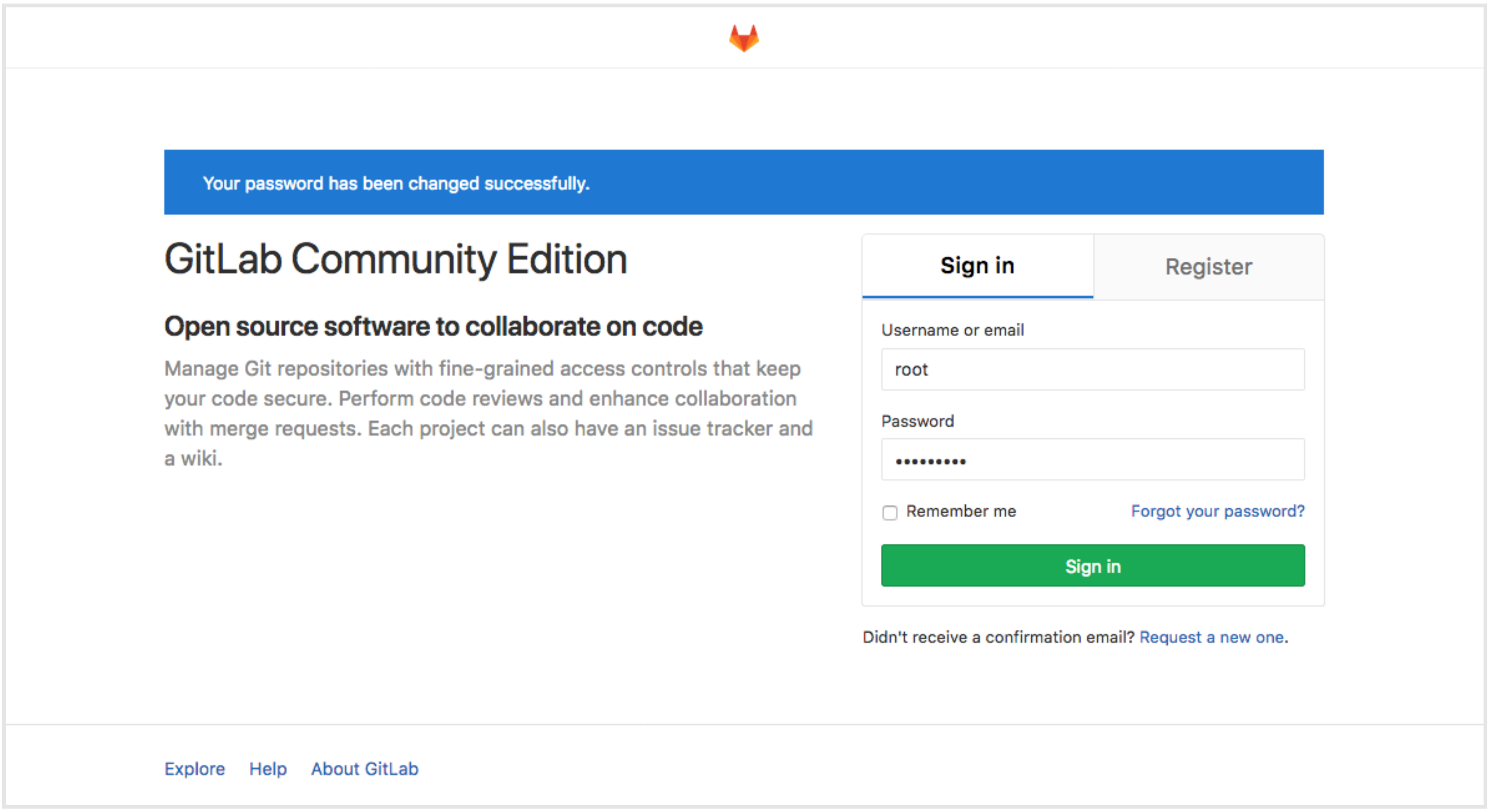
Already have login and password? Sign in

Explore Help About GitLab

SCROLL TO TOP

In the initial password prompt, supply and confirm a secure password for the administrative account. Click on the **Change your password** button when you are finished.

You will be redirected to the conventional GitLab login page:



Your password has been changed successfully.

GitLab Community Edition

Open source software to collaborate on code

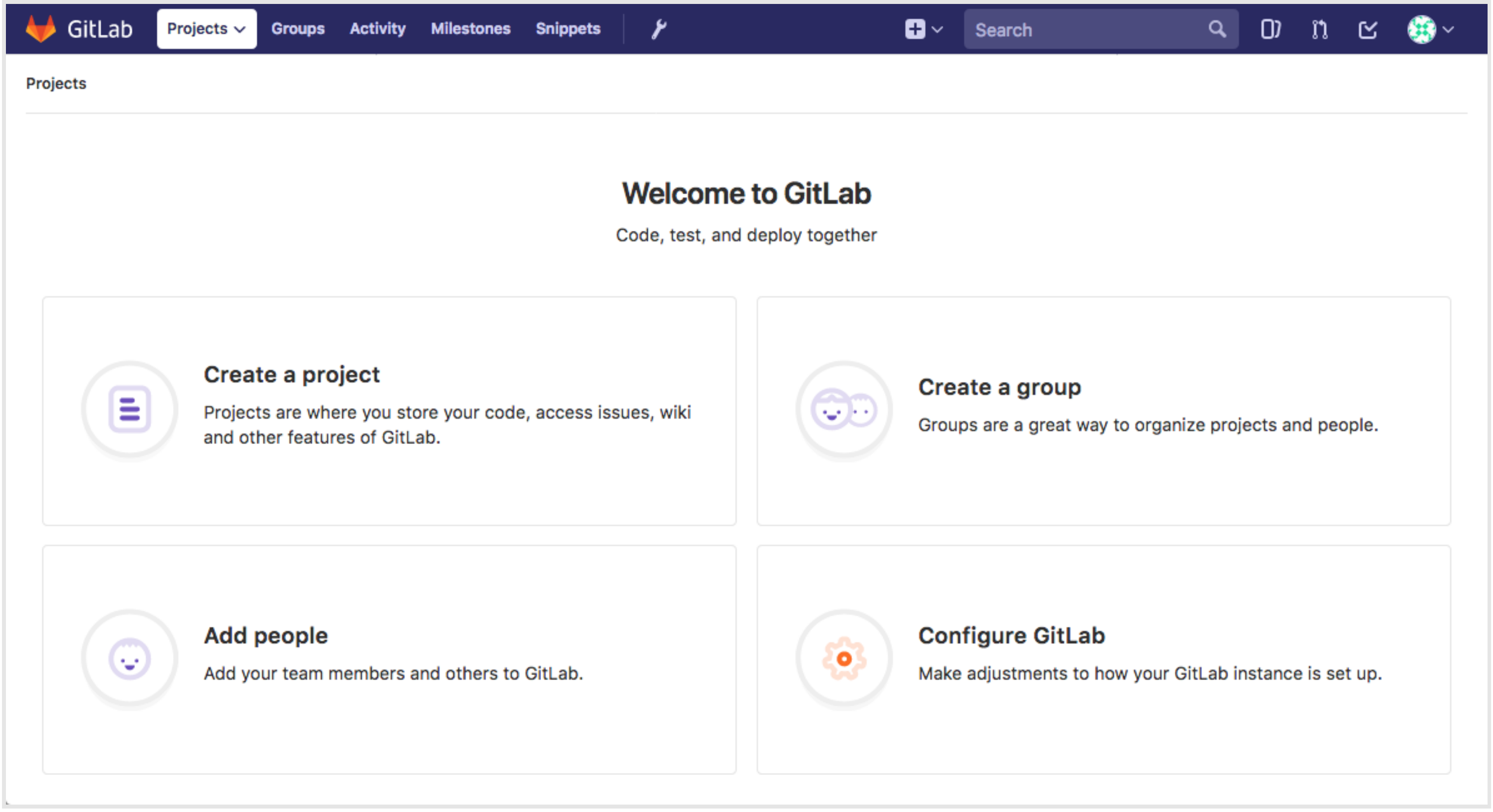
Manage Git repositories with fine-grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

Here, you can log in with the password you just set. The credentials are:

- Username: **root**
- Password: [the password you set]

Enter these values into the fields for existing users and click the **Sign in** button. You will be signed into the application and taken to a landing page that prompts you to begin adding projects:

SCROLL TO TOP



Welcome to GitLab

Code, test, and deploy together

Create a project
Projects are where you store your code, access issues, wiki and other features of GitLab.

Create a group
Groups are a great way to organize projects and people.

Add people
Add your team members and others to GitLab.

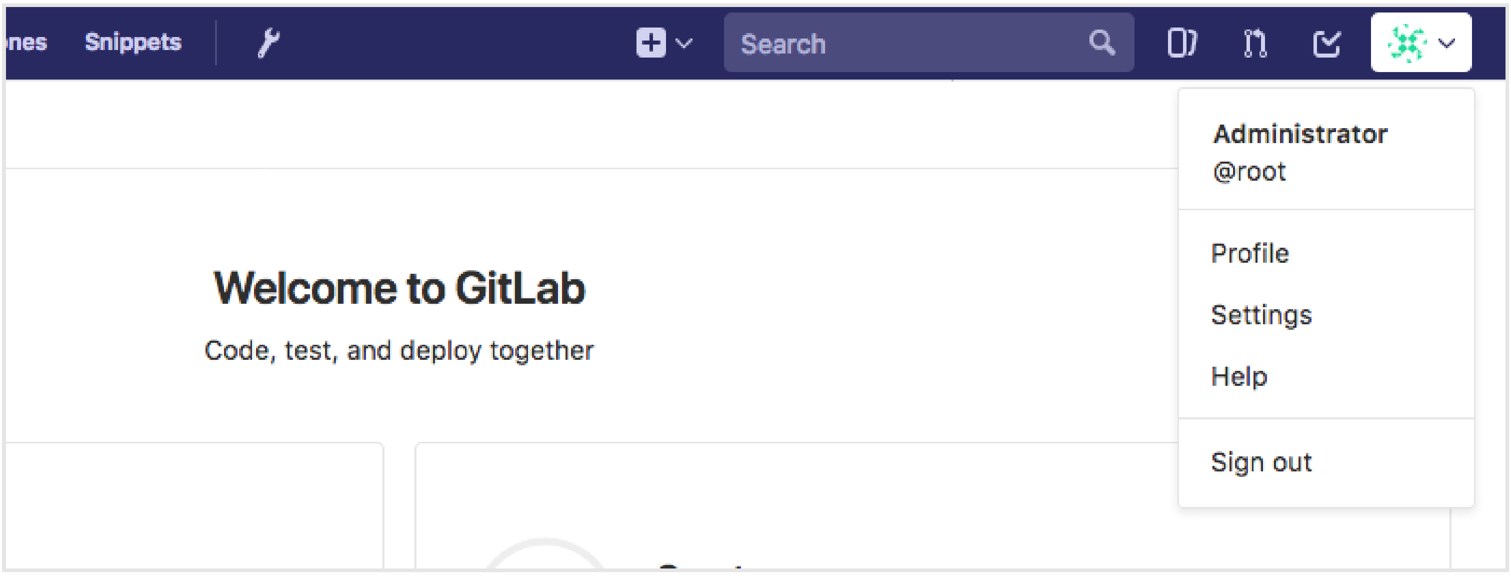
Configure GitLab
Make adjustments to how your GitLab instance is set up.

You can now make some simple changes to get GitLab set up the way you'd like.

Adjusting your Profile Settings

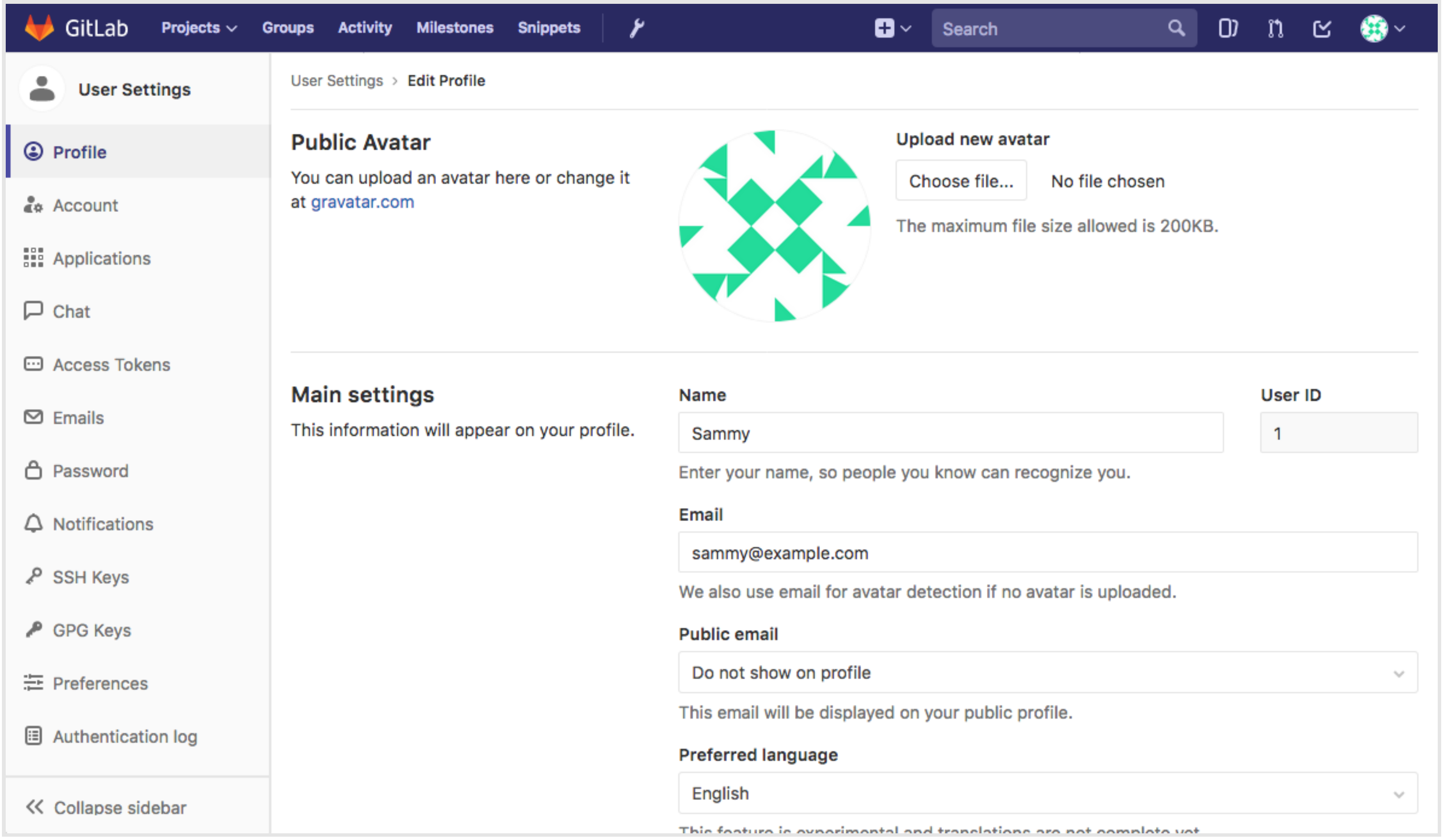
One of the first things you should do after a fresh installation is get your profile into better shape. GitLab selects some reasonable defaults, but these are not usually appropriate once you start using the software.

To make the necessary modifications, click on the user icon in the upper-right hand corner of the interface. In the drop down menu that appears, select **Settings**:



You will be taken to the **Profile** section of your settings:

SCROLL TO TOP



Public Avatar

You can upload an avatar here or change it at gravatar.com

Upload new avatar

No file chosen

The maximum file size allowed is 200KB.

Main settings

This information will appear on your profile.

Name: Sammy

Enter your name, so people you know can recognize you.

User ID: 1

Email: sammy@example.com

We also use email for avatar detection if no avatar is uploaded.

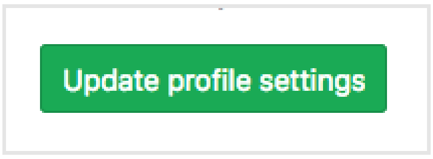
Public email: Do not show on profile

This email will be displayed on your public profile.

Preferred language: English

Adjust the **Name** and **Email** address from "Administrator" and "admin@example.com" to something more accurate. The name you select will be displayed to other users, while the email will be used for default avatar detection, notifications, Git actions through the interface, etc.

Click on the **Update Profile settings** button at the bottom when you are done:

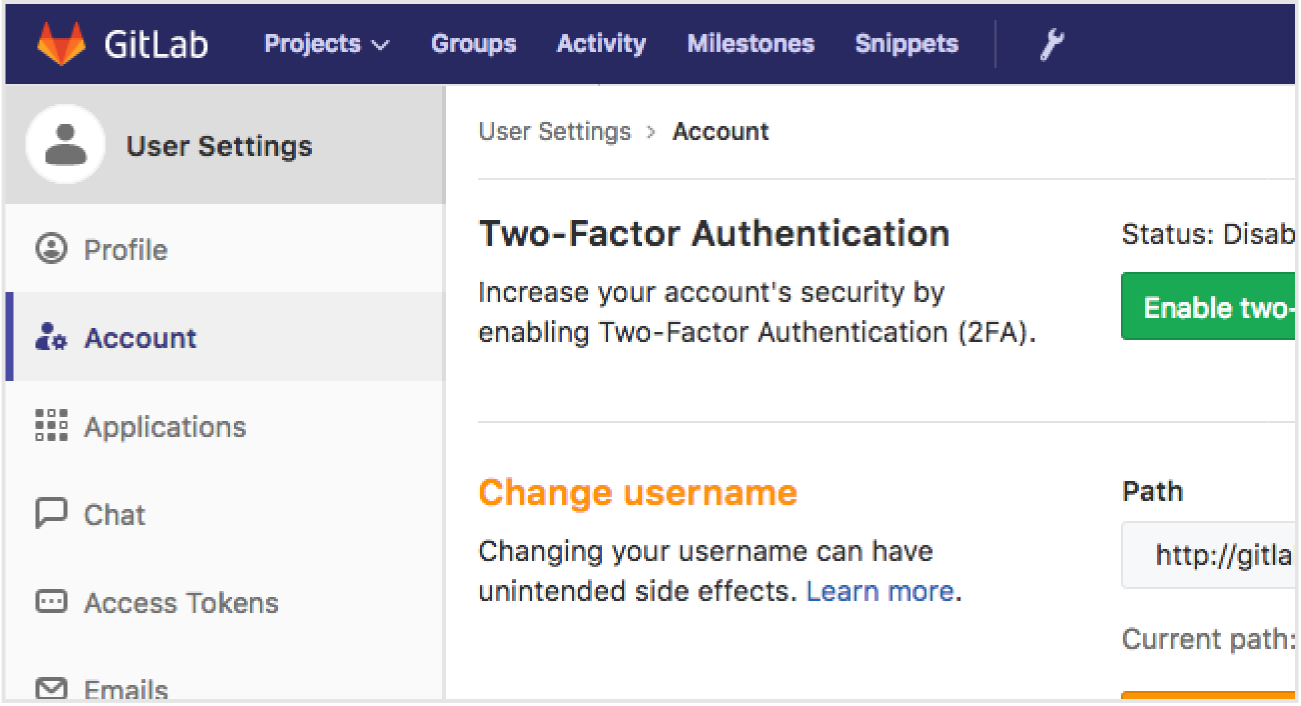


A confirmation email will be sent to the address you provided. Follow the instructions in the email to confirm your account so that you can begin using it with GitLab.

Changing Your Account Name

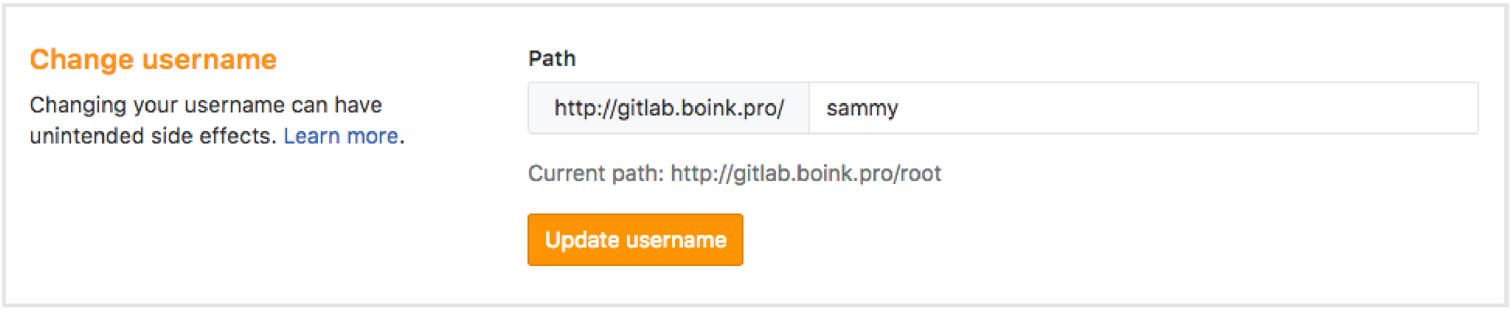
Next, click on the **Account** item in the left-hand menu bar:

SCROLL TO TOP

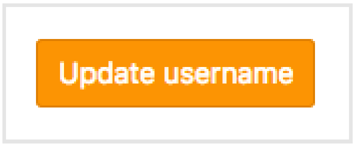


Here, you can find your private API token or configure two-factor authentication. However, the functionality we are interested in at the moment is the **Change username** section.

By default, the first administrative account is given the name **root**. Since this is a known account name, it is more secure to change this to a different name. You will still have administrative privileges; the only thing that will change is the name. Replace **root** with your preferred username:


Change username
Changing your username can have unintended side effects. Learn more.
Path
http://gitlab.boink.pro/
sammy
Current path: http://gitlab.boink.pro/root
Update username

Click on the **Update username** button to make the change:



Next time you log in to the GitLab, remember to use your new username.

Adding an SSH Key to your Account

In most cases, you will want to use SSH keys with Git to interact with your GitLab projects. To do this, you need to add your SSH public key to your GitLab account.

If you already have an SSH key pair created on your **local computer**, you can usually view the public key by typing:

SCROLL TO TOP

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

You should see a large chunk of text, like this:

```
Output
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDMuyMtMl6aWwqBCvQx7YXvZd7bCFVDsyln3yh5/8Pu23LW88VXfJgsBvhZZ9W0r
```

Copy this text and head back to the Profile Settings page in GitLab's web interface.

If, instead, you get a message that looks like this, you do not yet have an SSH key pair configured on your machine:

```
Output
cat: /home/sammy/.ssh/id_rsa.pub: No such file or directory
```

If this is the case, you can create an SSH key pair by typing:

```
$ ssh-keygen
```

Accept the defaults and optionally provide a password to secure the key locally:

```
Output
Generating public/private rsa key pair.
Enter file in which to save the key (/home/sammy/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/sammy/.ssh/id_rsa.
Your public key has been saved in /home/sammy/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:I8v5/M5xOicZRZq/XRcSBNxTQV2BZszjlWaIHi5chc0 sammy@gitlab.docsthat.work
The key's randomart image is:
+---[RSA 2048]----+
|          ..%o==B|
|           *.E =.|
|        . ++= B  |
|         ooo.o . |
|      . S .o  . .|
|     . + .. .   o|
|      +   .o.o ..|
|       o .++o .  |
|        oo=+     |
+----[SHA256]-----+
```

Once you have this, you can display your public key as above by typing:

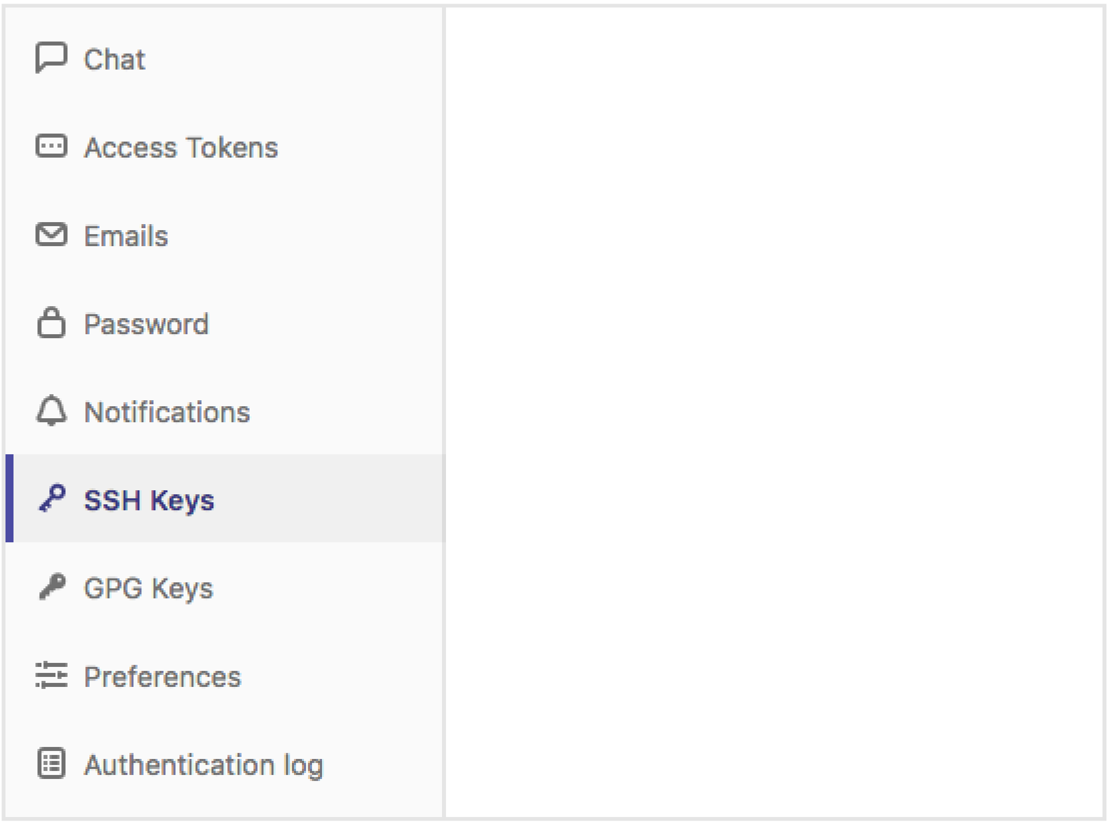
SCROLL TO TOP

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

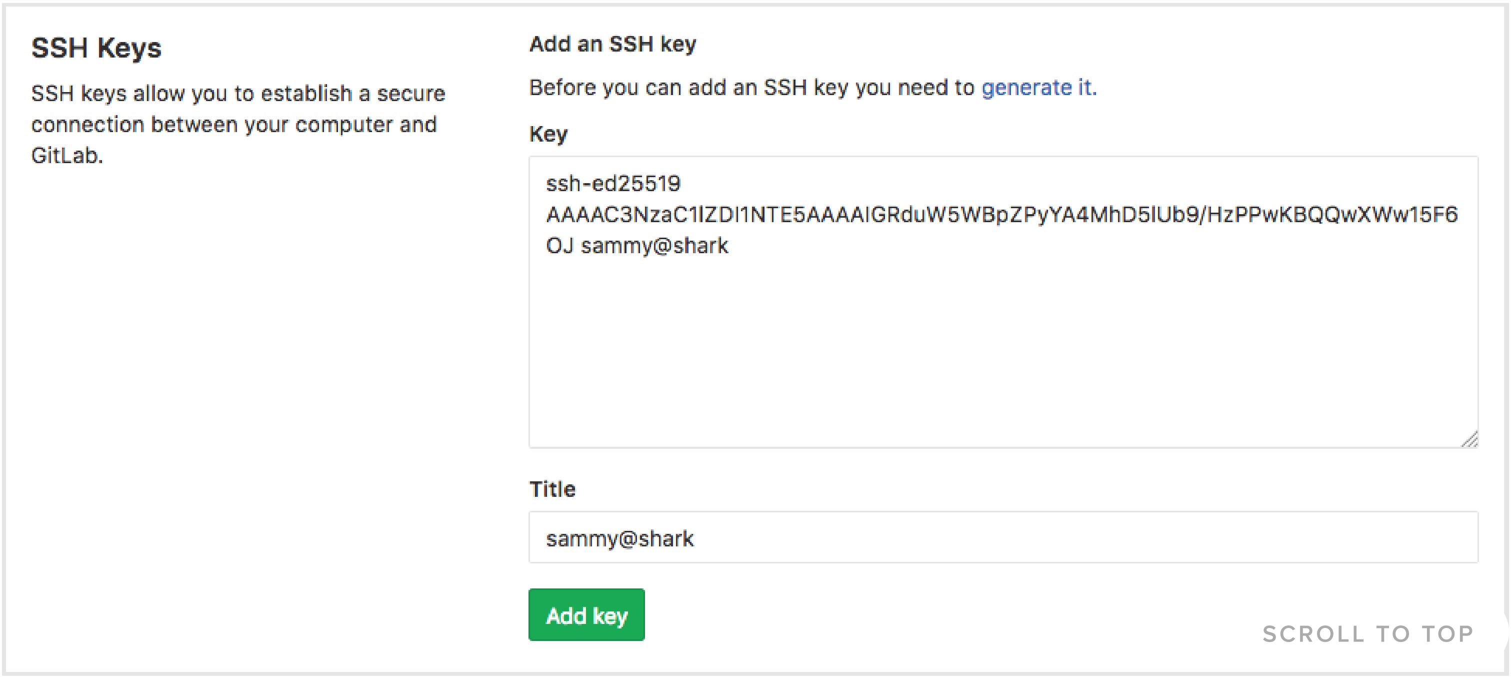
```
Output
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDMuyMtMl6aWwqBCvQx7YXvZd7bCFVDsyln3yh5/8Pu23LW88VXfJgsBvhZZ9W0r
```

Copy the block of text that's displayed and head back to your profile **Settings** in GitLab's web interface.

Click on the **SSH Keys** item in the left-hand menu:



In the provided space paste the public key you copied from your local machine. Give it a descriptive title, and click the **Add key** button:



SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

Add an SSH key

Before you can add an SSH key you need to generate it.

Key

```
ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIGRduW5WBpZPyYA4MhD5lUb9/HzPPwKBQQwXWw15F6
OJ sammy@shark
```

Title

sammy@shark

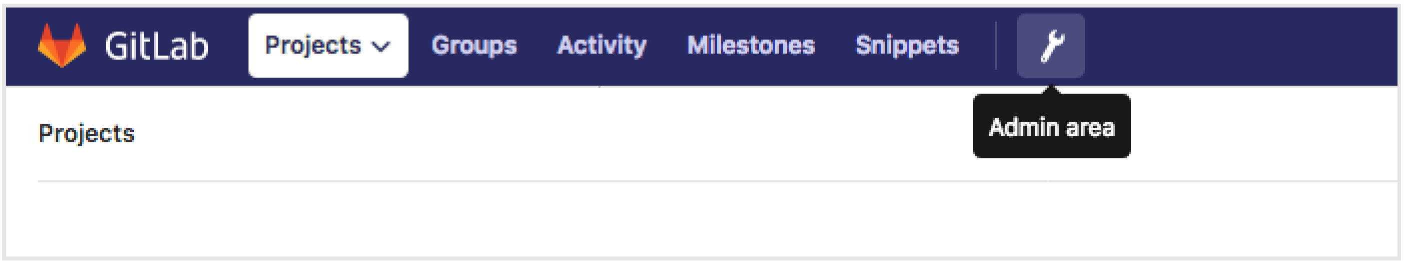
SCROLL TO TOP

You should now be able to manage your GitLab projects and repositories from your local machine without having to provide your GitLab account credentials.

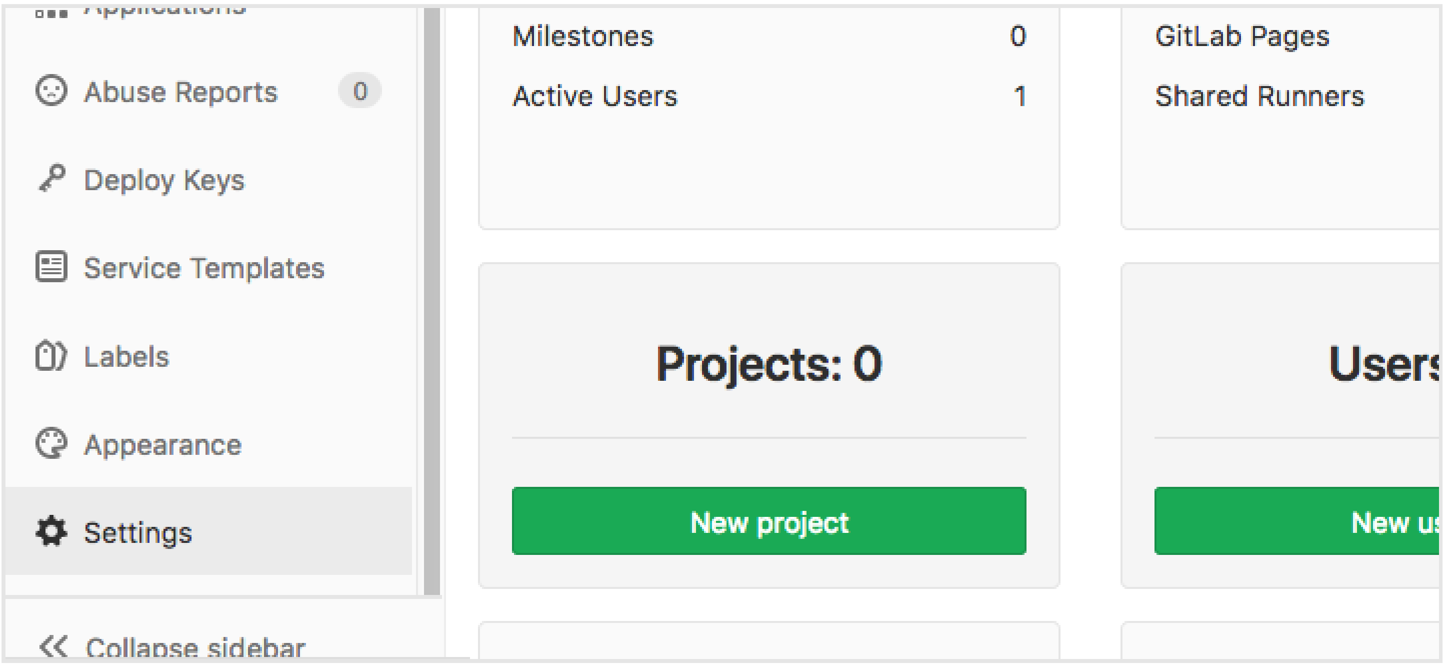
Step 6 — Restricting or Disabling Public Sign-ups (Optional)

You may have noticed that it is possible for anyone to sign up for an account when you visit your GitLab instance's landing page. This may be what you want if you are looking to host public project. However, many times, more restrictive settings are desirable.

To begin, make your way to the administrative area by clicking on the **wrench icon** in the main menu bar at the top of the page:



On the page that follows, you can see an overview of your GitLab instance as a whole. To adjust the settings, click on the **Settings** item at the bottom of the left-hand menu:



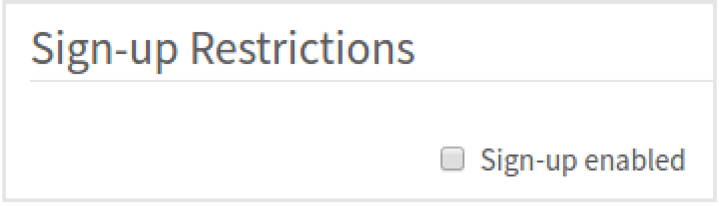
You will be taken to the global settings for your GitLab instance. Here, you can adjust a number of settings that affect whether new users can sign up and their level of access.

Disabling Sign-ups

If you wish to disable sign-ups completely (you can still manually create accounts for new users), scroll down to the **Sign-up Restrictions** section.

Deselect the **Sign-up enabled** check box:

SCROLL TO TOP



Scroll down to the bottom and click on the **Save changes** button:

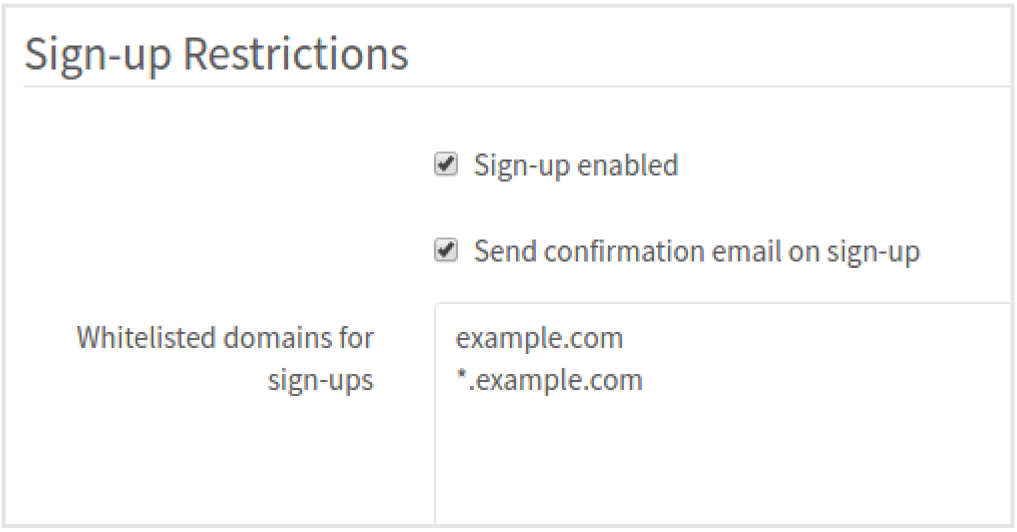
The sign-up section should now be removed from the GitLab landing page.

Restricting Sign-ups By Domain

If you are using GitLab as part of an organization that provides email addresses associated with a domain, you can restrict sign-ups by domain instead of completely disabling them.

In the **Sign-up Restrictions** section, select the **Send confirmation email on sign-up** box, which will allow users to log in only after they've confirmed their email.

Next, add your domain or domains to the **Whitelisted domains for sign-ups** box, one domain per line. You can use the asterisk "*" to specify wildcard domains:



Whitelisted domains for sign-ups

```
example.com
*.example.com
```

Scroll down to the bottom and click on the **Save changes** button:

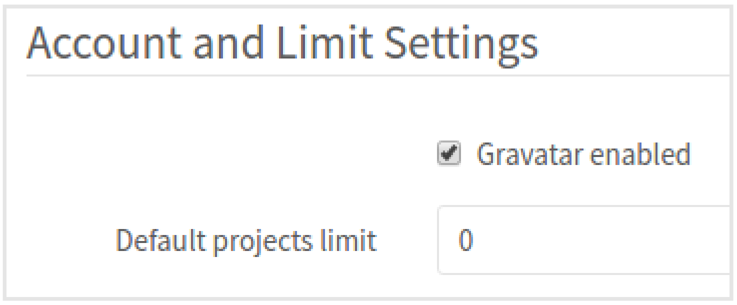
The sign-up section should now be removed from the GitLab landing page.

Restricting Project Creation

By default, new users can create up to 10 projects. If you wish to allow new users from the outside for visibility and participation, but want to restrict their access to creating new projects, you **Account and Limit Settings** section.

SCROLL TO TOP

Inside, you can change the **Default projects limit** to 0 to completely disable new users from creating projects:


Account and Limit Settings

☑ Gravatar enabled

Default projects limit: 0

New users can still be added to projects manually and will have access to internal or public projects created by other users.

Scroll down to the bottom and click on the **Save changes** button:

New users will now be able to create accounts, but unable to create projects.

Renewing Let's Encrypt Certificates

By default, GitLab has a scheduled task set up to renew Let's Encrypt certificates after midnight every fourth day, with the exact minute based on your `external_url`. You can modify these settings in the `/etc/gitlab/gitlab.rb` file. For example, if you wanted to renew every 7th day at 12:30, you could configure this as follows:

/etc/gitlab/gitlab.rb

```
letsencrypt['auto_renew_hour'] = "12"
letsencrypt['auto_renew_minute'] = "30"
letsencrypt['auto_renew_day_of_month'] = "*/7"
```

You can also disable auto-renewal by adding an additional setting to `/etc/gitlab/gitlab.rb`:

/etc/gitlab/gitlab.rb

```
letsencrypt['auto_renew'] = false
```

With auto-renewals in place, you will not need to worry about service interruptions.

Conclusion

You should now have a working GitLab instance hosted on your own server. You can begin to import or create new projects and configure the appropriate level of access for your team. GitLab is regularly adding features and making updates to their platform, so be sure to check out the project's hom SCROLL TO TOP
to-date on any improvements or important notices.

By: Justin Ellingwood By: Kathleen Juell

Upvote (8) Subscribe Share

We just made it easier for you to deploy faster.

TRY FREE

Related Tutorials

How To Sync and Share Your Files with Seafile on Debian 9

How To Install YunoHost on Debian 9

How To Ensure Code Quality with SonarQube on Ubuntu 18.04

How To Use Traefik as a Reverse Proxy for Docker Containers on Debian 9

How To Secure a Containerized Node.js Application with Nginx, Let's Encrypt, and Docker Compose

3 Comments

Leave a comment...

SCROLL TO TOP

Log In to Comment

0 jwdobken *July 25, 2018*

I am following this tutorial but gitlab is not reachable from my browser

---

0 worya *November 1, 2018*

Great!

---

0 nstocking *January 10, 2019*

For those wishing to have gitlab operate on a system with an already-existing SSL key, you should do the following:

1. Don't do any of the let's encrypt stuff in this tutorial. That will try to set up another key and will get in the way.
2. Add the following directives to your /etc/gitlab/gitlab.rb: nginx['ssl*certificate']="$path*tocert" *nginx['ssl*certificate*key']="$path*to*private*key" replacing the variables with the proper paths (keep the quote marks). For example, a let's encrypt key would be installed like nginx['ssl*certificate']="/etc/letsencrypt/live/example.com/fullchain.pem" nginx['ssl*certificate_key']="/etc/letsencrypt/live/privkey.pem"
3. Reconfigure using "gitlab-ctl reconfigure", and test. If it doesn't work, check the gitlab nginx error log with "tail /var/log/gitlab/nginx/error.log". This may be useful if you are running gitlab on a system that already has a key or you don't want to use let's encrypt to get your key.

SCROLL TO TOP