Gitlab – Terraform CI/CD

What is Gitlab?

Gitlab is a service that provides remote access to Git repositories. In addition to hosting your code, the services provide additional features designed to help manage the software development lifecycle.

These additional features include managing the sharing of code between different people, bug tracking, wiki space and other tools for 'social coding'.

Git

Git - It is a source code versioning system that lets you locally track changes and push or pull changes from remote resources.

Github v/s Gitlab

GitHub is a publicly available, free service which requires all code (unless you have a paid account) be made open. Anyone can see code you push to GitHub and offer suggestions for improvement. GitHub currently hosts the source code for tens of thousands of open source projects.

GitLab is a github like service that organizations can use to provide internal management of git repositories. It is a self hosted Git-repository management system that keeps the user code private and can easily deploy the changes of the code.

Gitlab License model

GitLab is built on an open core model. That means there are two versions of GitLab: Community Edition and Enterprise Edition.

GitLab Community Edition is open source, with an MIT Expat license.

GitLab Enterprise Edition is built on top of Community Edition: it uses the same core, but adds additional features and functionality on top of that. This is under a proprietary license.

For both versions: All javascript code in GitLab is open source. All javascript code written by GitLab is under the same MIT license.

Install Gitlab-CE

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.

Prerequisites

For this tutorial, you will need:

The published GitLab hardware requirements recommend using a server with:

- **2 cores**
- **8 GB 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.**

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.

```
# sudo apt update
```

```
# sudo apt install ca-certificates curl openssh-server postfix
```

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/script.deb.sh
```

Continue ...

Feel free to examine the downloaded script to ensure that you are comfortable with the actions it will take.

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

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

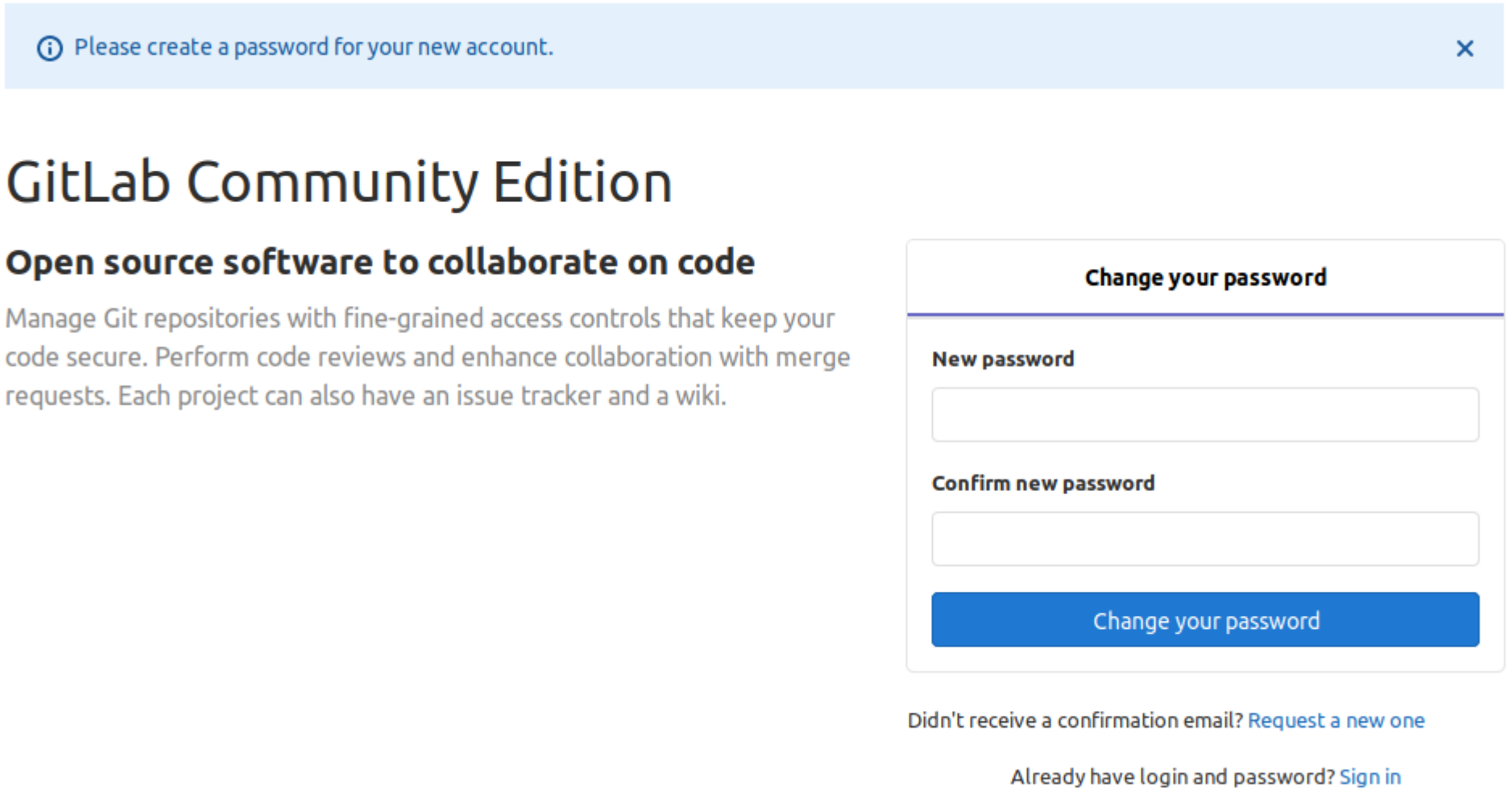
```
# sudo bash /tmp/script.deb.sh
```

The script will set up your server to use the GitLab maintained repositories. 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.

Setting Password



Creating Project

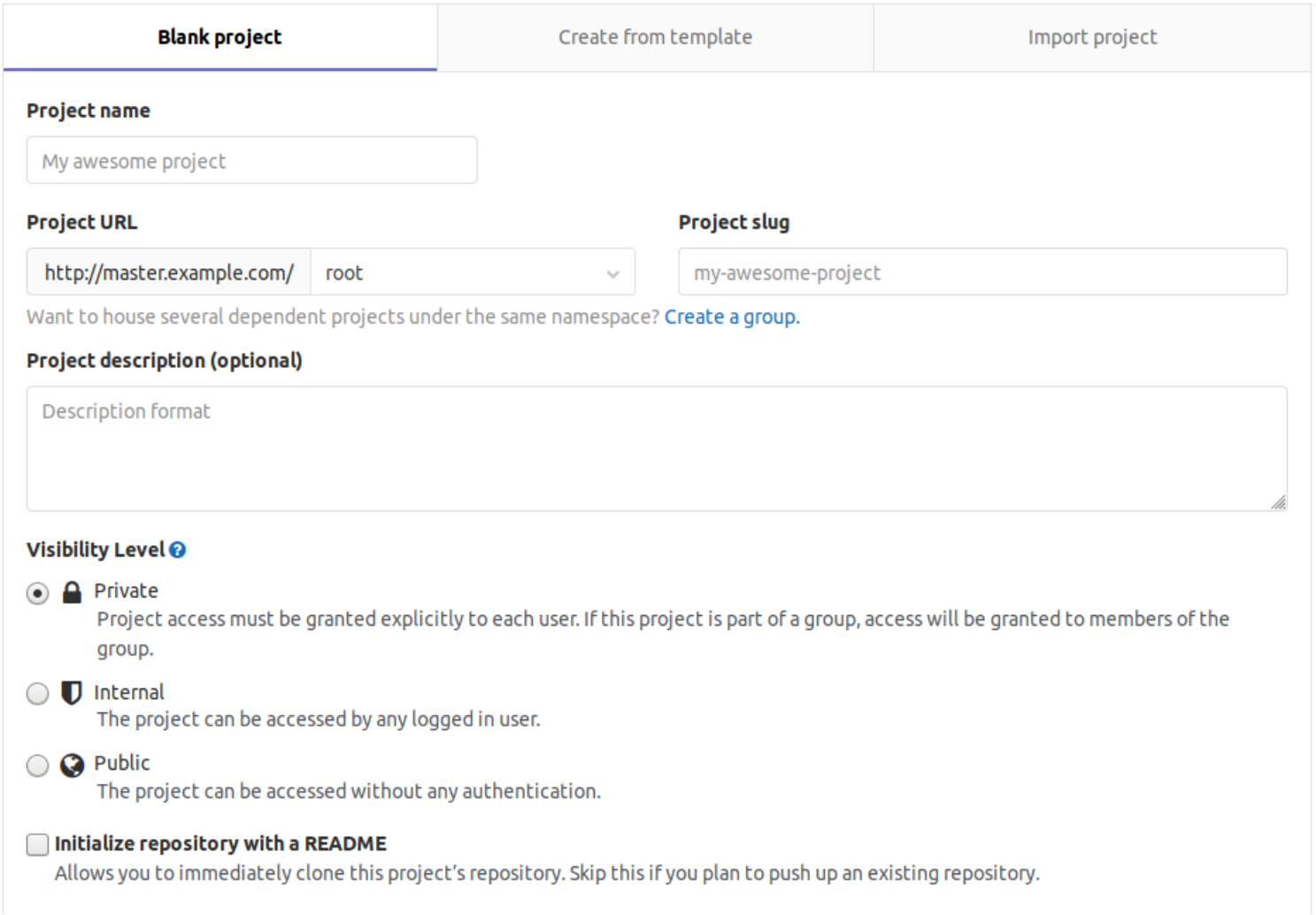
New project

A project is where you house your files (repository), plan your work (issues), and publish your documentation (wiki), among other things.

All features are enabled for blank projects, from templates, or when importing, but you can disable them afterward in the project settings.

Information about additional Pages templates and how to install them can be found in our Pages getting started guide.

Tip: You can also create a project from the command line. Show command



Accessing It

```
 /tmp$ --> git clone http://master.example.com/root/project1
Cloning into 'project1'...
Username for 'http://master.example.com': root
Password for 'http://root@master.example.com':
warning: redirecting to http://master.example.com/root/project1.git/
warning: You appear to have cloned an empty repository.
 /tmp$ --> cd project1/
 /tmp/project1$ -->
 /tmp/project1$ --> echo "hi there file1" > file1
 /tmp/project1$ --> git add .
 /tmp/project1$ --> git commit -m "c1"
[master (root-commit) 4533521] c1
 1 file changed, 1 insertion(+)
 create mode 100644 file1
 /tmp/project1$ --> git push origin master
Username for 'http://master.example.com': root
Password for 'http://root@master.example.com':
warning: redirecting to http://master.example.com/root/project1.git/
Counting objects: 3, done.
Writing objects: 100% (3/3), 209 bytes | 209.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To http://master.example.com/root/project1
 * [new branch]      master -> master
 /tmp/project1$ --> 
```

CI and CD

Continuous Integration is the practice of merging all the code that is being produced by developers. The merging usually takes place several times a day in a shared repository.

Continuous Delivery adds that the software can be released to production at any time, often by automatically pushing changes to a staging system.

Continuous Deployment goes further and pushes changes to production automatically.

GitLab CI

GitLab CI (Continuous Integration) service is a part of GitLab which manages the project and user interface and allows unit tests on every commit and indicates with warning message when there is an unsuccessful of build.

Getting Started

GitLab CI/CD is configured by a file called .gitlab-ci.yml placed at the repository's root. This file creates a pipeline, which runs for changes to the code in the repository.

Pipelines consist of one or more stages that run in order and can each contain one or more jobs that run in parallel. These jobs (or scripts) get executed by the GitLab Runner agent.

GitLab Runner

GitLab Runner is the open source project that is used to run your jobs and send the results back to GitLab. It is used in conjunction with GitLab CI/CD, the open-source continuous integration service included with GitLab that coordinates the jobs.

GitLab Runner is written in Go and can be run as a single binary, no language specific requirements are needed.

It is designed to run on the GNU/Linux, macOS, and Windows operating systems. Other operating systems will probably work as long as you can compile a Go binary on them.

Installing GitLab runner

Simply download one of the binaries for your system:

 # sudo curl -L --output /usr/local/bin/gitlab-runner https://gitlab-runner-downloads.s3.amazonaws.com/latest/binaries/gitlab-runner-linux-amd64

 # sudo chmod +x /usr/local/bin/gitlab-runner

 # sudo useradd --comment 'GitLab Runner' --create-home gitlab-runner --shell /bin/bash

 # sudo gitlab-runner install --user=gitlab-runner --working-directory=/home/gitlab-runner

 # sudo gitlab-runner start

Register GitLab Runner

 # sudo gitlab-runner register

 # Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com)

https://gitlab.com

 # Please enter the gitlab-ci token for this runner

 # Please enter the gitlab-ci description for this runner

[hostname] my-runner

 # Please enter the executor: