Usage SwarmLab gitea!

Πίνακας περιεχομένων

This HowTo teaches you how to Use SwarmLab gitea.

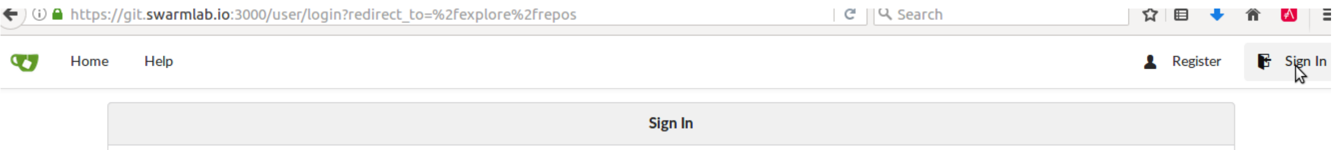
Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

1. create a new repository

Open Swarmlab Gitea

Use any web browser on your computer to join

1.1. Sign In



https://git.swarmlab.io:3000/user/login?redirect_to=%2fexplore%2frepos
Search
Home
Help
Register
Sign In
Sign In

Figure 1. Click on "Sign-in"

Proxy Error

```
Proxy Error

The proxy server received an invalid response from an upstream server.
The proxy server could not handle the request GET /user/login.

Reason: Error reading from remote server

Apache/2.4.25 (Debian) Server at git.swarmlab.io Port 3000
```

Reload Page!!!

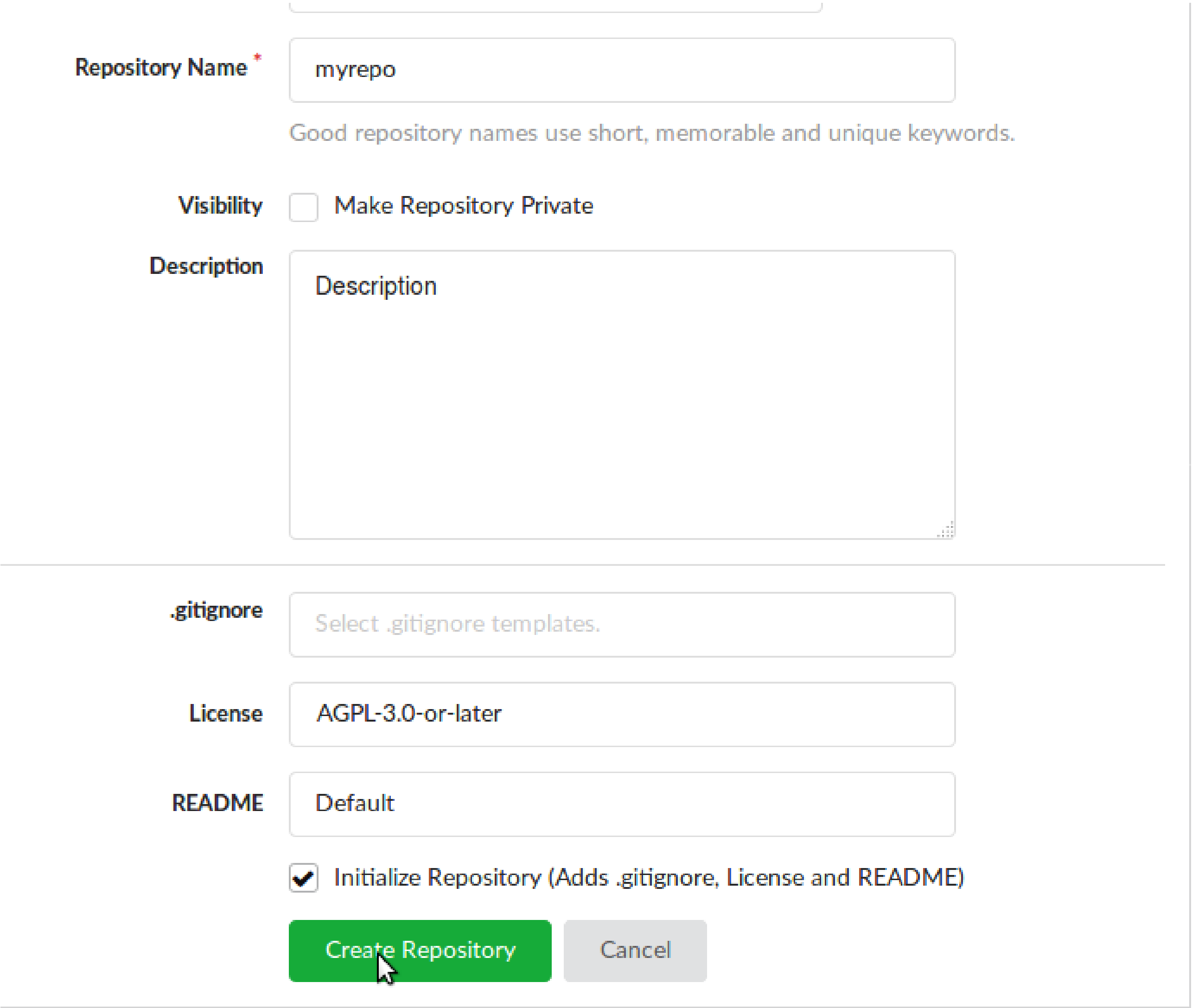
1.2. New Repository

1.2.1. step 1

[SwarmLab New Repository] | *./repo-1.png*

Figure 2. New Repository

1.2.2. step 2



Repository Name * myrepo

Good repository names use short, memorable and unique keywords.

Visibility □ Make Repository Private

Description: Description

.gitignore: Select .gitignore templates.

License: AGPL-3.0-or-later

README: Default

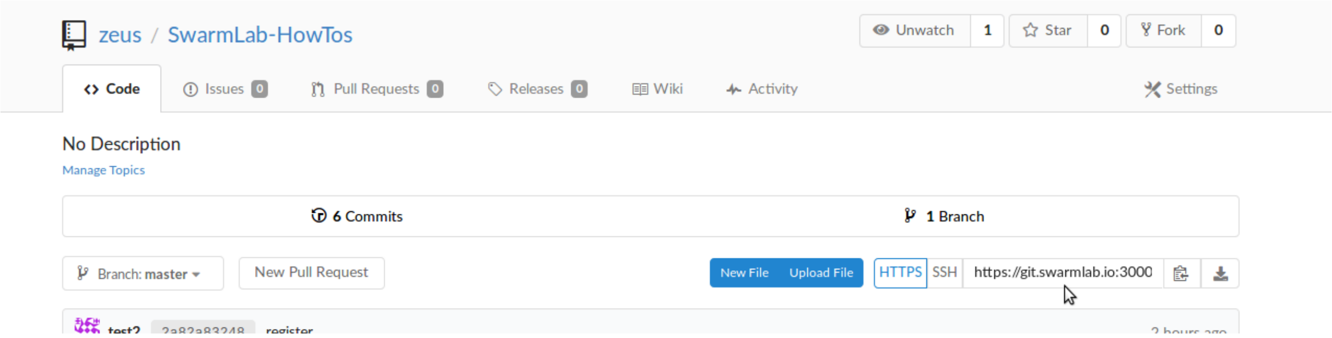
☑ Initialize Repository (Adds .gitignore, License and README)

Figure 3. Create New Repository

2. clone repository

2.1. git clone

2.1.1. copy url



zeus / SwarmLab-HowTos

No Description

Manage Topics

6 Commits 1 Branch

Branch: master HTTPS SSH https://git.swarmlab.io:3000

Figure 4. clone Repositor

2.1.2. paste url

- On your computer!

git clone

```
git clone paste-url-here
```

Install Git

```
sudo apt update
sudo apt install git
```

git error

The requested URL returned error: 502

Try again!!!

3. workflow

3.1. add & commit

You can propose changes (add it to the Index) using

git add

```
git add <filename>
git add *
```

This is the first step in the basic git workflow. To actually commit these changes use

git status

git status

git commit

```
git commit -a -m "Commit message"
```

Now the file is committed to the HEAD, but not in your remote repository yet.

3.2. pushing changes

Your changes are now in the HEAD of your local working copy.

To send those changes to your remote repository, execute *git push*

```
git push origin master
```

Change master to whatever branch you want to push your changes to.

3.3. update

to update your local repository to the newest commit, execute

git pull

```
git pull origin
```

in your working directory to fetch and merge remote changes.

3.4. log

in its simplest form, you can study repository history using..

git log

```
git log
```

You can add a lot of parameters to make the log look like what you want. To see only the commits of a certain author:

git log

```
git log --author=bob
```

To see a very compressed log where each commit is one line:

git log

```
git log --pretty=oneline
```

Or maybe you want to see an ASCII art tree of all the branches, decorated with the names of tags and branches:

git log

```
git log --graph --oneline --decorate --all
```

See only which files have changed:

git log

```
git log --name-status
```

These are just a few of the possible parameters you can use. For more, see git log --help

4. links & resources

Git Community Book

A Visual Git Reference