

# Git

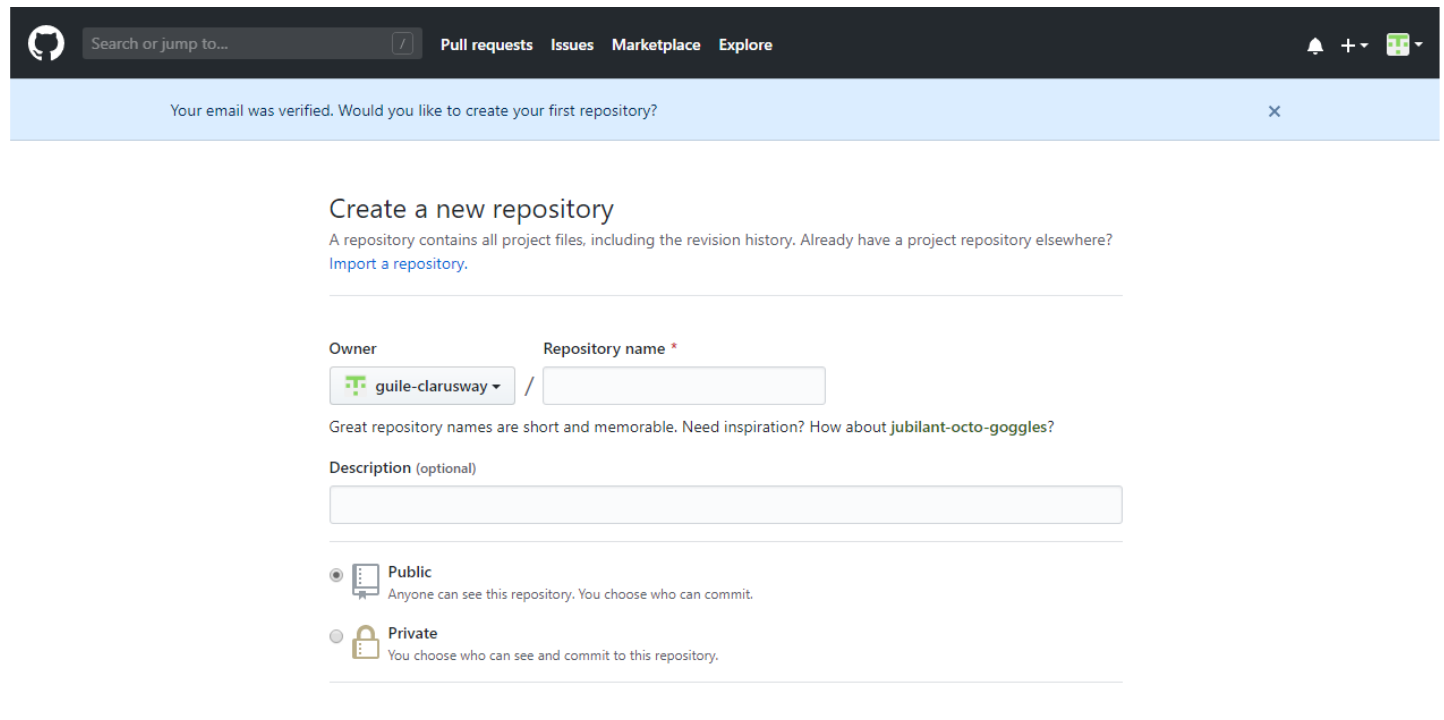
[Dashboard](#) / [Courses](#) / [Miscellaneous](#) / [Git](#) / [GitHub](#) / [GitHub](#)

## GitHub

- ✓ [What is GitHub?](#)
- ✓ [Source Code Hosting Facilities](#)
- ✓ [Creating GitHub Account](#)
- ✓ [Create Your First Repo](#)


## Create Your First Repo

After you get your email address verified, it is time to create your first repository.



**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner:  guile-clarusway /

Repository name \*

Great repository names are short and memorable. Need inspiration? How about [jubilant-octo-goggles?](#)

Description (optional)

☒ **Public**  
Anyone can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.


Type the name of your first repository, "lab1.1" in this case, and click "Create repository".

## Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner

Repository name \*


 guile-clarusway

 / 


lab1.1 

Great repository names are short and memorable. Need inspiration? How about [jubilant-octo-goggles?](#)

Description (optional)

☒  **Public**

Anyone can see this repository. You choose who can commit.

☐  **Private**


You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ **Initialize this repository with a README**






This will let you immediately clone the repository to your computer.

Add .gitignore: **None**

Add a license: **None** 

**Create repository**

Your first repo named lab1.1 is created. Note down your repo URL, since you will need the repo URL when configuring your local repo on your computer.




 Search or jump to...  [Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)   

Learn Git and GitHub without any code!

Using the Hello World guide, you'll start a branch, write comments, and open a pull request.





**Read the guide**

guile-clarusway / lab1.1

 Unwatch 1  Star 0  Fork 0

[Code](#) [Issues 0](#) [Pull requests 0](#) [Actions](#) [Projects 0](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

**Quick setup — if you've done this kind of thing before**

 Set up in Desktop or  HTTPS  SSH  

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

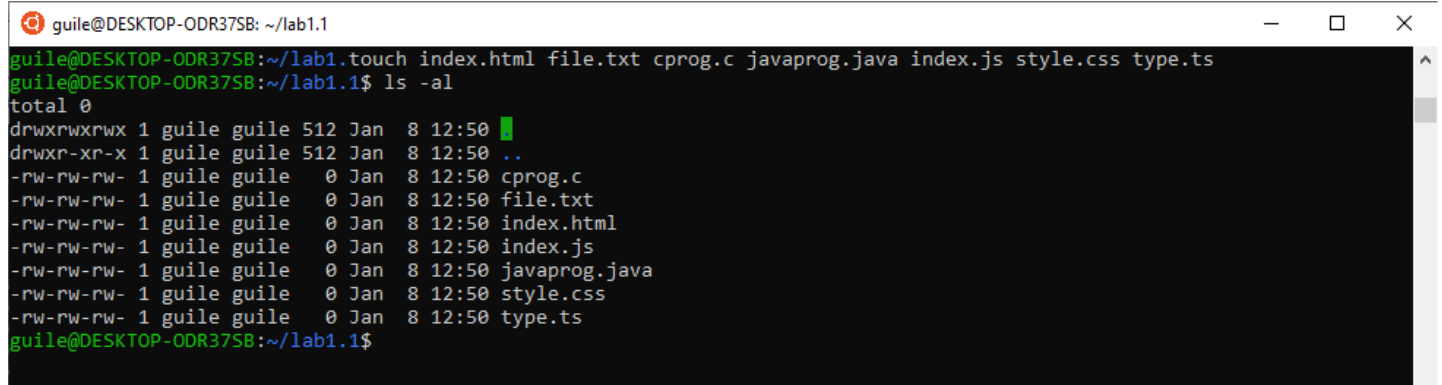
Now open **Git Bash** or a terminal in your computer and let's begin to create our local repo.

First, create some files by entering "touch" command. These files are just empty demo files and we will use them only to interact with GitHub. For the sake of this demo and for a better understanding create these files under the "lab1.1" folder which has the same name with our GitHub repo. This is not mandatory but makes

sense.

```
$ touch index.html file.txt cprog.c javaprog.java index.js style.css type.ts

$ ls -al
```

A terminal window titled 'guile@DESKTOP-ODR37SB: ~/lab1.1' showing the execution of 'touch' and 'ls -al' commands. The 'touch' command creates several files, and 'ls -al' lists them with their permissions, owners, and timestamps.

```
guile@DESKTOP-ODR37SB: ~/lab1.1
guile@DESKTOP-ODR37SB:~/lab1.1$ touch index.html file.txt cprog.c javaprog.java index.js style.css type.ts
guile@DESKTOP-ODR37SB:~/lab1.1$ ls -al
total 0
drwxrwxrwx 1 guile guile 512 Jan  8 12:50 .
drwxr-xr-x 1 guile guile 512 Jan  8 12:50 ..
-rw-rw-rw- 1 guile guile  0 Jan  8 12:50 cprog.c
-rw-rw-rw- 1 guile guile  0 Jan  8 12:50 file.txt
-rw-rw-rw- 1 guile guile  0 Jan  8 12:50 index.html
-rw-rw-rw- 1 guile guile  0 Jan  8 12:50 index.js
-rw-rw-rw- 1 guile guile  0 Jan  8 12:50 javaprog.java
-rw-rw-rw- 1 guile guile  0 Jan  8 12:50 style.css
-rw-rw-rw- 1 guile guile  0 Jan  8 12:50 type.ts
guile@DESKTOP-ODR37SB:~/lab1.1$
```

You then execute "git init" command to create an empty local repo.

```
$ git init
```

A terminal window titled 'guile@DESKTOP-ODR37SB: ~/lab1.1' showing the execution of 'git init' command, which initializes an empty Git repository in the current directory.

```
guile@DESKTOP-ODR37SB: ~/lab1.1
guile@DESKTOP-ODR37SB:~/lab1.1$ git init
Initialized empty Git repository in /home/guile/lab1.1/.git/
guile@DESKTOP-ODR37SB:~/lab1.1$
```

Initialized empty Git repository in "home/guile/lab1.1/.git".

"Git add" to add the files to staging area from the working area.

```
$ git add .
```

### 💡Tip:

- If you have a warning like this, you can ignore it for now.

```
warning: LF will be replaced by CRLF in lemonade.txt.
```

```
The file will have its original line endings in your working directory
```

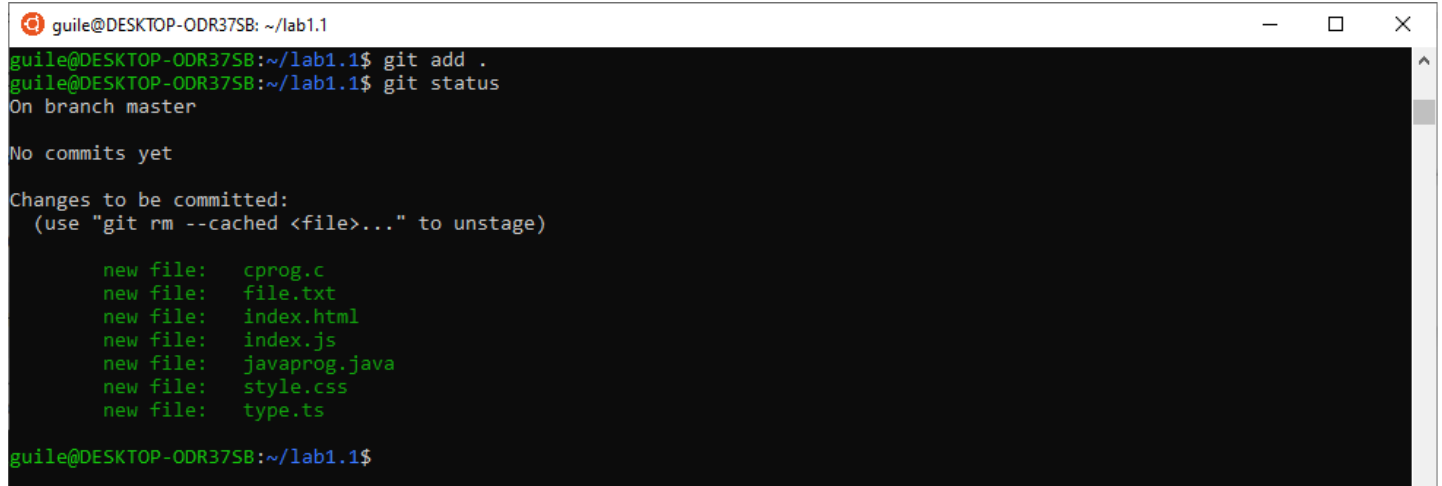
In Unix systems the end of a line is represented with a line feed (LF). In windows a line is represented with a carriage return (CR) and a line feed (LF) thus (CRLF). when you get code from git that was uploaded from a unix system they will only have an LF.

If you are a single developer working on a windows machine, and you don't care that git automatically replaces LF's to CRLF's, you can turn this warning off by typing the following in the git command line

```
git config core.autocrlf true
```

And now we can see the status of our files with the "git status" command.

```
$ git status
```

A terminal window titled 'guile@DESKTOP-ODR37SB: ~/lab1.1' showing the output of 'git add .' and 'git status'. The status shows 'On branch master' and 'No commits yet'. It lists seven new files to be committed: cprog.c, file.txt, index.html, index.js, javaprogram.java, style.css, and type.ts.

```
guile@DESKTOP-ODR37SB:~/lab1.1$ git add .
guile@DESKTOP-ODR37SB:~/lab1.1$ git status
On branch master

No commits yet

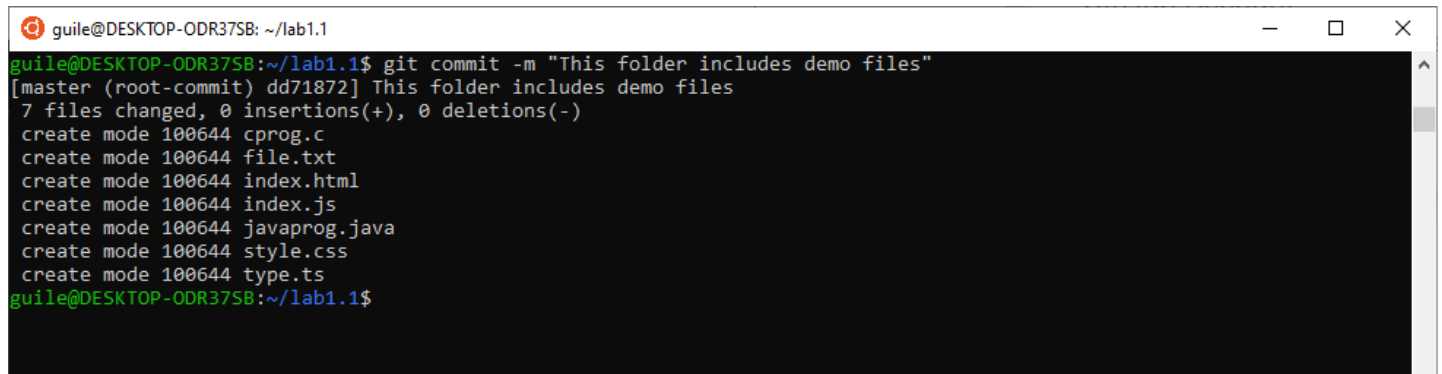
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   cprog.c
        new file:   file.txt
        new file:   index.html
        new file:   index.js
        new file:   javaprogram.java
        new file:   style.css
        new file:   type.ts

guile@DESKTOP-ODR37SB:~/lab1.1$
```

Let's commit our files to local repo with the command "git commit -m".

```
$ git commit -m "This folder includes demo files"
```

A terminal window titled 'guile@DESKTOP-ODR37SB: ~/lab1.1' showing the output of 'git commit -m "This folder includes demo files"'. It shows the commit hash 'dd71872' and lists the files created with their modes.

```
guile@DESKTOP-ODR37SB:~/lab1.1$ git commit -m "This folder includes demo files"
[master (root-commit) dd71872] This folder includes demo files
7 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 cprog.c
create mode 100644 file.txt
create mode 100644 index.html
create mode 100644 index.js
create mode 100644 javaprogram.java
create mode 100644 style.css
create mode 100644 type.ts
guile@DESKTOP-ODR37SB:~/lab1.1$
```

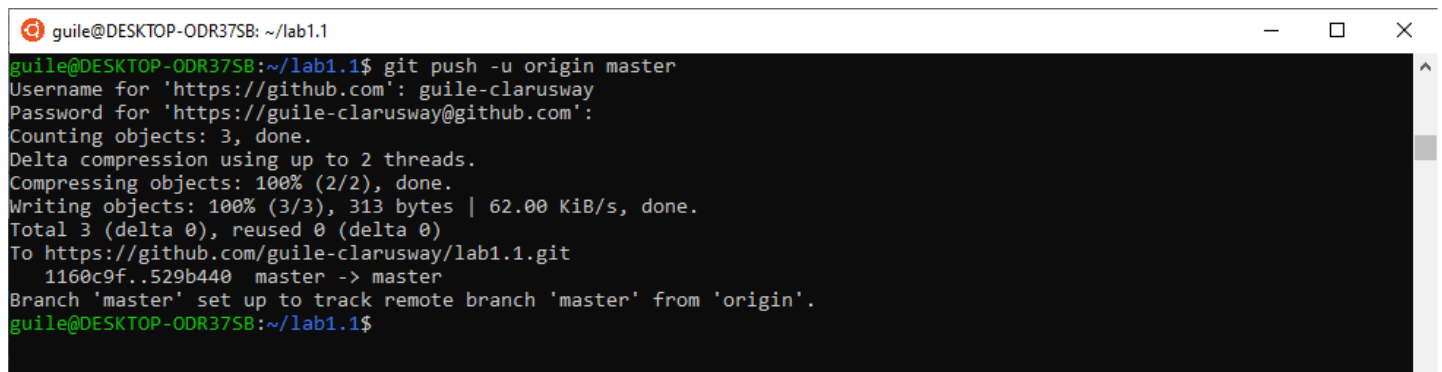
Now it is time to add our file to GitHub with command "git remote add". Use your own repo URL.

```
$ git remote add origin YOUR_REPO_URL
# For example: git remote add origin https://github.com/guile-clarusway/lab1.1.git
```

Local git instance added to remote repository. And then we can go ahead and push our files to remote repository we have just created on GitHub. Enter your GitHub username and password if asked.


```
$ git push -u origin main
```

This is what you may see after the command "git push -u origin main" executed.

A terminal window titled 'guile@DESKTOP-ODR37SB: ~/lab1.1' showing the output of 'git push -u origin master'. It prompts for GitHub username and password, then shows the push progress and the final state of the remote repository.

```
guile@DESKTOP-ODR37SB:~/lab1.1$ git push -u origin master
Username for 'https://github.com': guile-clarusway
Password for 'https://guile-clarusway@github.com':
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 313 bytes | 62.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/guile-clarusway/lab1.1.git
    1160c9f..529b440  master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
guile@DESKTOP-ODR37SB:~/lab1.1$
```

And this is what it looks like in GitHub after "git push command" executed.






Search or jump to...

Pull requests

Issues

Marketplace

Explore



guile-clarusway / lab1.1

Unwatch

1

Star

0

Fork

0

<> Code

Issues 0

Pull requests 0

Actions

Projects 0

Wiki

Security

Insights

Settings

Description

Short description of this repository

Website

Website for this repository (optional)

Save

or Cancel

Manage topics

1 commit

1 branch

0 packages

0 releases

1 contributor

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

guile-clarusway

This folder includes demo files

Latest commit e2227ca 7 hours ago

cprog.c

This folder includes demo files

7 hours ago

file.txt

This folder includes demo files

7 hours ago

index.html

This folder includes demo files

7 hours ago

index.js

This folder includes demo files

7 hours ago

javaprogram.java

This folder includes demo files

7 hours ago

style.css

This folder includes demo files

7 hours ago

type.ts

This folder includes demo files

7 hours ago

Previous

Next

You have completed 75% of the lesson

75%

Jump to...



f

t

i

in

© 2021 Copyright: Clarusway.com