

# An introduction to Github / Gitlab

Welcome to this introduction to Github / Gitlab for people who are learning programming.

This is the second part of the introduction to Git.

If you don't know about Git, you should first go through the first part of this tutorial.

In this second two-hours-module you will learn how to collaborate with other people:

- How to use one of the common platforms for Git for your projects (three of them will be mentioned: Github, Gitlab, Gitea).
- How to contribute to other people projects (or let other people contribute to your projects)

In the same mood as in the first part of this tutorial, we will give a general introduction to the topic but focus on what is most commonly used. This means that we will mainly talk about the two platforms Github and Gitlab and we will use Visual Studio Code to manage the code.

But you're welcome to use any other platform and software you are already comfortable with.

## Table of contents

- [Hosting the source code repositories](#)
- [The hosting platforms](#)
- [Creating an account](#)
  - [Setup your account](#)
  - [Create the ssh keys](#)
- [Creating a project](#)
- [Editing the project online](#)
- [Cloning a project to your computer](#)
- [Pushing an existing project](#)
- [Forking and contributing to a project](#)
- [Syncing a fork](#)
- [Github](#)
- [Git remote add upstream](#)
- [Other services for your project](#)
- [Notes](#)

## Hosting the source code repositories

In the first part of this tutorial, we have learned the basics of Git and it's likely that it has felt a bit strange to do it without Github.

It was probably not what you expected.

Even if most people have an hard time to see a difference between Git and Github:

- you can use Git without a platform like Github
- there are several other platforms for hosting the source code repositories.

But there is a reason why the second part of this workshop is called *An introduction to Github*:

- Hosting the repositories in the web helps for collaboration.
- People might get interested in your project and not only use it but also become contributors.

- Most of the existing platforms do more than host your code and

If your project is open, it's seems obvious why you would want it to be hosted in a public space.

But even for private projects (personal or for a company), it might be useful to have them hosted in a place that can be reached when you're connected to the internet (sharing with friends, clients, employees working from home, freelancers, clients, ...).

## The hosting platforms

At the time of writing, here are the most common hosting platforms:

- [Github](#): for the social network part of it.
- [GitLab](#): the other big player, can be self hosted for free.
- [Bitbucket](#): from the vendors of Jira and Trello.
- [Sourcehut](#): a clean and simple interface; hacker's style and supports Mercurial.
- [Codeberg](#): Free Software style; hosted in the EU.
- [Sourceforge](#): For the nostalgics. With SVN support.

You can of course install a Git platform in your local network:

- [Gitea](#): lightweight and simple to install.
- [Gitlab](#): with all bells and wistles.
- [Gogs](#): lightweight and simple to install.

## Creating an account

Before your can host your projects, you need to create an account:

- For Github, go to <https://github.com> and click on the *Sign up* button in the top right corner.
- For Gitlab, go to <https://gitlab.com> and click on Login, then on the [Register now](#) link towards the end of the page (I'm not sure what the *Free trial* button does...)

and follow the instructions and sign into the platform.

### The email address

As said in the first part of this tutorial, you need to take care about which email address and name you're setting for Git and the related platforms: even if there are way to increase your privacy, you should be comfortable with the idea that they might be publically visible at some time or under certain circumstances.

## Setup your account

- on Gitlab you might need to setup a Group to be able to fork projects (just create a group with your username)... to be checked

## Create the ssh keys

You're used to identify to services by providing a username and a password.

When working with Git platforms it's advised to work with *keys* insted of a password:

- It's tedious to type the password each time you want to push your code (ok, often you can get your system to store and provide the password...)
- Passwords tend to be too short to be secure enough.

Public repositories can be modified by anybody who can figure out your password. On the other side, you (and other people, if your code gets famous) will download the code and run it on their computer without much protection: this makes it interesting to try to inject malicious code into random people's code. This is the main reason why, Git platforms tend to only provide a key based authentication.

- We will be creating and using a *ed25519* key. Or, if your system does not support it, a 2048-bit RSA.

This is often seen as one of the *hardest* steps, when setting up a Git workflow. The major Git platforms have already written good tutorials and we will refer to their instructions for setting up the SSH keys:

- [Generating a new SSH key and adding it to the ssh-agent](#) for Github
- [Use SSH keys to communicate with GitLab](#)

In short:

- We are using the *standard* tool SSH to create the keys.
- The keys will be in the *.ssh* folder in your *home* folder.
- At the end of the process, you have one private key that must be kept as a secret on your computer, and a public key that you can share with other people (like the Git service provider).
- This is called a [asymmetric cryptography](#) (or public key cryptography)

## Creating a project

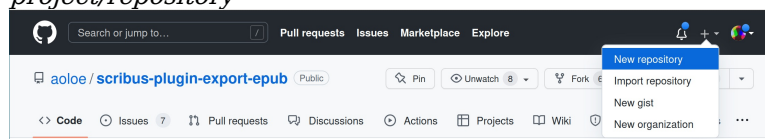
When starting a new project the most simple way is to first create a project in the web platform and then clone it onto your computer.

- When choosing the name (and the slug) you should already use a good name (there are ways to rename the project but they are not completely straightforward)
- Private or public.
- Create a README file

For this tutorial, we will create the new project "Simple translator" with the *slug* `simple-translator`.

## Creating a project on Github

- Click on the **+** button – the left most of the right top toolbar – and pick *New project/repository*




- Fill the repository name, the description. Make it public and do not add the

Create a new repository

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

Owner <sup>\*</sup> Repository name <sup>\*</sup>

 solbe /

Great repository names are short and memorable. Need inspiration? How about [effective-spork?](#)

Description (optional)

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

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

Initialize this repository with:

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

☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: **None**

**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

License: **None**

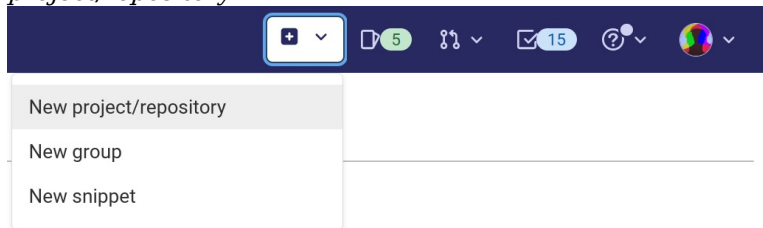
☐ You are creating a public repository in your personal account.

[Create repository](#)

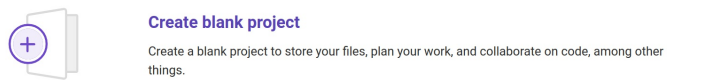
*README.md.*

## Creating a project on Gitlab

- Click on the **+** button – the left most of the right top toolbar – and pick *New project/repository*



- In pick the *Create blank project* option



- Fill the project name, the namespace, and slug. Make it public and uncheck the creation of the *README.md*.

Create blank project

Create a blank project to store your files, plan your work, and collaborate on code, among other things.

New project > Create blank project

**Project name**

**Project URL**

Pick a group or namespace / **Project slug**

Want to organize several dependent projects under the same namespace? [Create a group.](#)

**Project deployment target (optional)**

**Visibility Level** ⓘ

☐ Private  
Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.

☒ **Public**  
The project can be accessed without any authentication.


**Project Configuration**

☐ Initialize repository with a README  
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

☐ Enable Static Application Security Testing (SAST)  
Analyze your source code for known security vulnerabilities. [Learn more.](#)

[Create project](#) [Cancel](#)

## Editing the project online

- Click on the README.md file to see its details.
- Starting editing the file,
  - on Github click on the button with a pencil ;
  - on Gitlab on the *Open in Web IDE* button.

- Commit the file:
  - on Github fill the *Commit changes* section and click on Commit changes.
  - on Gitlab click on the *Create commit...* button. If you want to leave the editor without saving the changes, simply use the back button in your browser or on Github the *Cancel* button.

### The web IDE

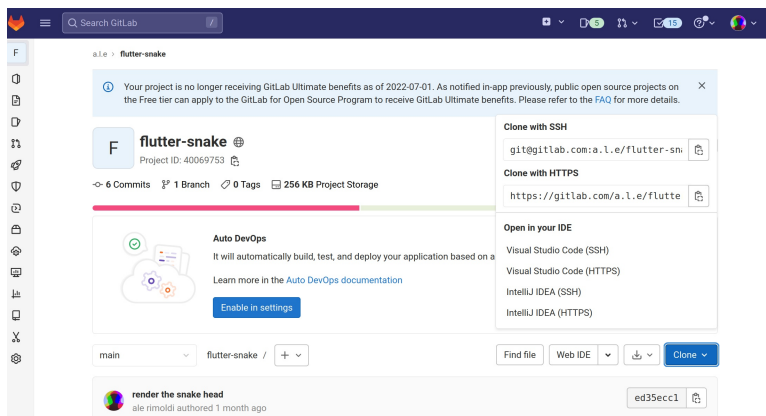
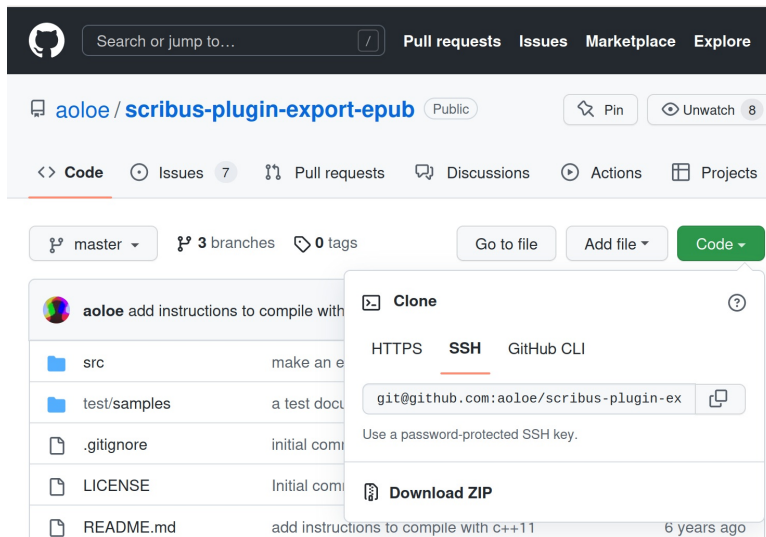
Both Github and Gitlab provide an powerful web editor for managing the files in your repository online. To get into it, you simply need to press the `.` (the dot key) on your keyboard while your at the root of your project.

For leaving the IDE, on Github you can click on the `≡` button in the top left corner and *Go to repository*. On Gitlab you can use the *project* button at the same location.

## Cloning a project to your computer

First, you need the address of the repository you want to clone onto your computer.

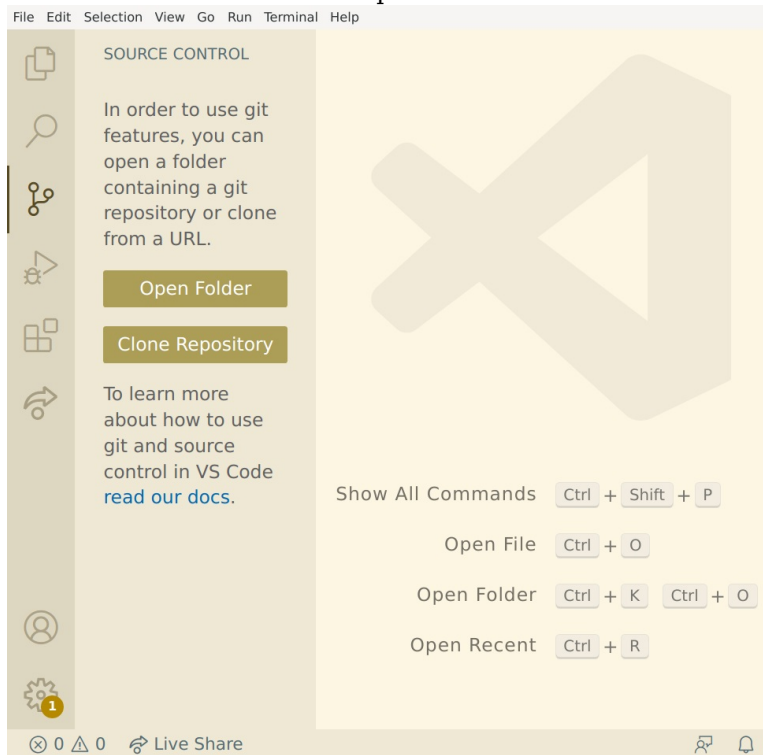
In the browser get to the project page and click on the *Code* (Github) or *Clone* (GitLab) button.



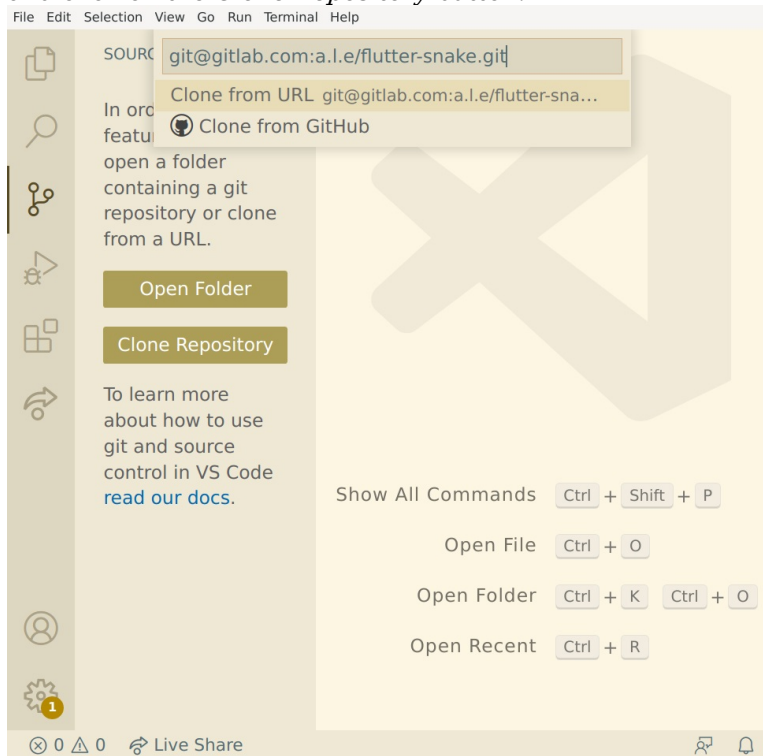
Copy the *SSH* address (`git@gitxxx.com:xxxxx/xxxxx.git`).

In Visual Studio Code we can use the address we have copied :

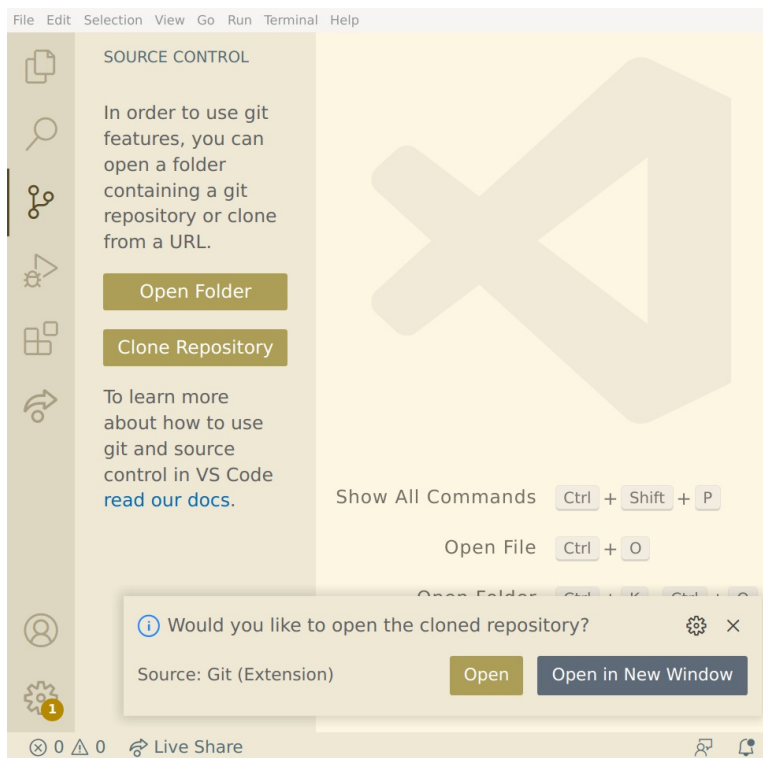
- create a new window
- Go to the *Source Control* left panel



and click on the *Clone Repository* button.



- Select your development folder (~/.Development...) and *Open* the repository in the current window



(and if you're the author of the code you're cloning, you can light hearted tell Visual Studio Code that it should enable all features)

## Pushing an existing project

In real life, most of the time, you will first start writing some code and after some time think that it might be worth creating a Git repository for it.

There an easy way but a bit stupid:

- Create a new project on the web platform as we did above,
- clone it as we did before into a new folder,
- move your existing files into that new folder, and
- add, commit and push the files.

That's easy enough that we won't try it out right now, but focus on the *optimal* way of doing it:

- In the web platform, create a new project without creating a README.md file
- You will get to a page with a few way how you can start adding content to your repository.

Let's publish our *Simple calculator* from the first part of this tutorial:

- On the Github / Gitlab create a new empty project (do not let it crete a README.md file, since we already have one).
- Keep the page with the instructions open.

Push an existing Git repository

```
cd existing_repo
git remote rename origin old-origin
git remote add origin git@gitlab.com:a.l.e/simple-calculator.git
git push -u origin --all
git push -u origin --tags
```

- In Visual Studio Code, open the *Simple calculator* project in a new window.

- Open the *Source Control* section in the left panel.
- Click on the *three dots* menu button and choose *Remote > Add Remote...*
- Paste the url from the Github / Gitlab page (something like `git@gitlab.com:xxx/simple-calculator.gig`) and click on *Add remote from URL*.
- Provide the remote name: *origin*.
- (No, Visual Studio Code does not need to periodically run *fetch*)

You might have noticed that we ignored the big *Publish Branch* button: you would need the Github extension for it and it won't work for Gitlab or any other Git service provider.

## Forking and contributing to a project

- Fork on Github / Gitlab (that's way you might need an account on each platform...)
- Create a new branch (online or in vscode) named after the changes you want to apply
- Edit and commit
- Push
- Got to Github / Gitlab and create the merge request
- Carefully review the changes you have made ()
- Forget about the branch

Try it by taking somebody's else *Simple Calculator* or *Simple translator*, make some changes and issue a pull / merge request.

Notes:

- [Creating a pull request from a fork](#)

## Accepting a pull request

Somebody makes me a pull / merge request please : - )

## Syncing a fork

In Github there is a button for it.

You can then do a *pull* in the *three dots menu* in the *Source Control* panel.

Otherwise:

- Use the *three dots menu* in the *Source Control* panel...
- to add a remote with the HTTPS (you don't have GIT access to that repository!) URL of the forked repository and call it *upstream*.
- In Visual Studio Code terminal:

```
# commit or stash your work
git switch main
git fetch upstream
git merge upstream/main
git switch your-branch
# git switch your-current-branch
```

I will check if there is a more visual way of doing this...



- [Syncing a fork](#) with Github and the command line.

Git remote add upstream

## **Other services for your project**

- Issue tracking
- Documentation (Wiki)
- CI, CD, automation,
- Deliver the releases.
- Websites

## **Notes**

This tutorial is based on:

- <http://opentechschooll.github.io/social-coding/>

-[Working with GitHub in VS Code](#) - [Setting up GitLab Key](#)