

git seminar

Introduction to the version control system

Preface

- Robbe De Greef
- robbe@infogroep.be
- discord.infogroep.be
- Slides will be available on seminars.infogroep.be
- Questions? Just ask

What is git

- Distributed version control system
- Created by Linus Torvalds, for Linux
- Keeps track of your project
 - History of changes
 - Data integrity
- Makes collaboration easy

Linux was 'just' a reimplementation of Unix. Git proved I could be more than a one-hit wonder.

Concept of branches

git works on the concept of branches

- There is always at least one branch
 - main / master (master is not often used anymore)
- You can make changes on a different branch
- Merge branches
- Try out different things on different branches

Installation

Intermezzo: GitHub

- Most used git sharing platform
- Social media for developers
- Find repos, work together, brag about yourself
- others:
 - gitlab
 - gitea
 - ...

The screenshot shows the GitHub profile of Robbe De Greef. The profile includes a circular profile picture, the name "Robbe De Greef", and a bio: "Mainly a low level developer but I'm pretty fond of programming in general. Google SWE & STEP intern 2021 & 2022". It also shows 20 followers and 11 following, location in Brussels, Belgium, and contact information. The "Pinned" section features repositories: yanix (Public), vubhub (Public), ASWJ (Public), SafeCC (Public), CppPathTracer (Public), and AsmRayTracer (Public). The "492 contributions in the last year" section displays a heatmap for the year 2024. The "Activity overview" section shows a bar chart with 94% Commits, 5% Pull requests, and 1% Issues. The "Contribution activity" section shows the month of October 2024.

Profile: Robbe De Greef (RobbeDGreef)
Mainly a low level developer but I'm pretty fond of programming in general. Google SWE & STEP intern 2021 & 2022

Followers: 20 followers · 11 following
Location: Brussels, Belgium
Email: robbedegreef@gmail.com
In: in/robbe-de-greef

Achievements: 4 icons (100%, 2x, etc.)

Highlights: 1 (PRO)

Organizations: 2 icons

Pinned Repositories:

- yanix** (Public): Yanix is a UNIX-like kernel / operating system build completely from scratch. 83 stars, 5 forks.
- vubhub** (Public): The unofficial VUB app - VubHub. 3 stars, 1 fork.
- ASWJ** (Public): A Slicer Writing Journey. G-code.
- SafeCC** (Public): C Compiler that strives to be as (memory) safe as possible. 8 stars, 1 fork.
- CppPathTracer** (Public): Simple C++ path tracer. 1 star, 1 fork.
- AsmRayTracer** (Public): VUB second bachelor assembly programming project. 1 star.

492 contributions in the last year (Contribution settings)

Activity overview: Contributed to Awper-Dev/maply, soft-compilers/2023-24-project..., RobbeDGreef/CppPathTracer and 9 other repositories.

Contribution activity: 94% Commits, 5% Pull requests, 1% Issues.

Contribution activity: October 2024

Installing git

Useful link: <https://github.com/git-guides/install-git>

Linux

- ``sudo apt update && sudo apt install git``
- ``sudo pacman -Syu git``

Windows

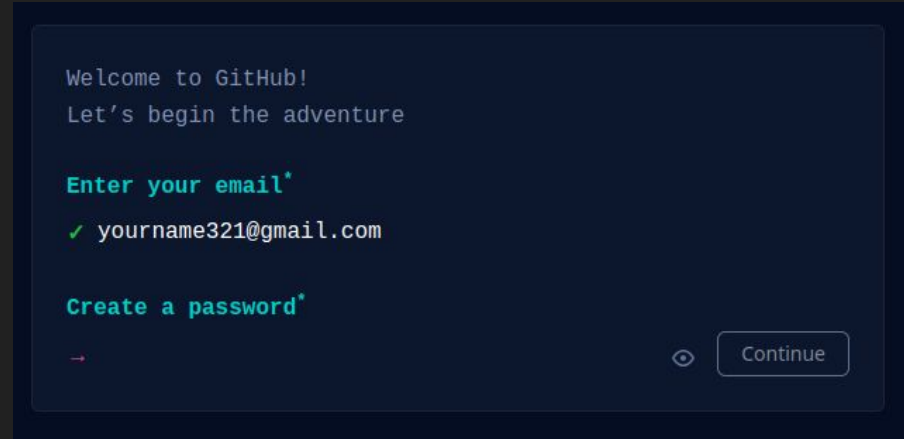
- <https://gitforwindows.org/> (actually git might already be installed these days)

MacOS

- <https://git-scm.com/downloads/mac> (probably already installed)

Creating github account

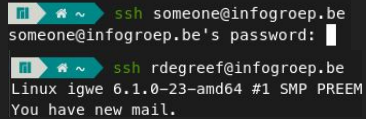
- <https://github.com/>
- Go to sign up
- Do your thing



Intermezzo: SSH keys

Authentication with SSH (Secure SHell) server

- Username/password
- SSH keys
 - “dude trust me, its me”



```
ssh someone@infogroep.be
someone@infogroep.be's password: 
ssh rdegreef@infogroep.be
Linux igwe 6.1.0-23-amd64 #1 SMP PREEM
You have new mail.
```

The image shows a terminal window with two SSH sessions. The first session shows a user logging in with a password. The second session shows a user logging in with a key, displaying system information and a mail notification.

They come in pairs

- Public key - you give this to the server
- Private key - **you keep this to yourself**

Intermezzo: SSH keys

Useful link:

<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

Generating your own SSH key

- ``ssh-keygen -t ed25519 -C "your_email@example.com"``
- ``eval "$(ssh-agent -s)"``
- ``ssh-add ~/.ssh/id_ed25519``

Setting your username / email

- ``git config --global user.name "Your username"``
- ``git config --global user.email "Your email"``

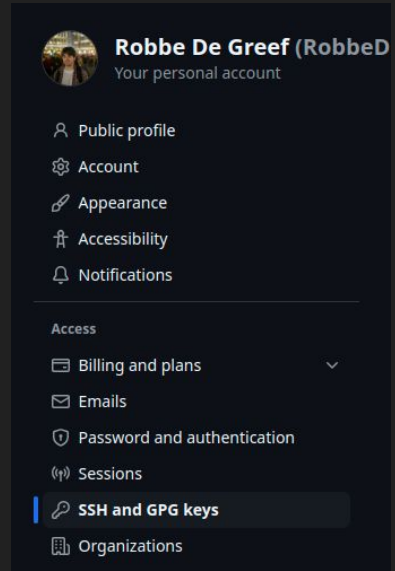
tip:

You can override these per repository. This can be useful if you want to work under a different name / email. Such as for school.
Just run these commands **without** the `--global` flag

Adding SSH key to GitHub

GitHub only allows communication via SSH key

- Copy output of public key
 - ``cat ~/.ssh/id_e25519.pub``
- Go to Settings > SSH and GPG keys > New SSH key
- Add contents of public key here
- Give it a name and save

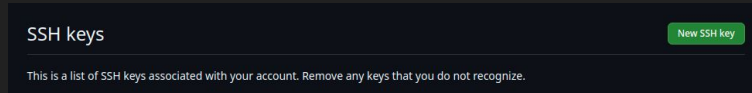


Robbe De Greef (RobbeD)
Your personal account

- Public profile
- Account
- Appearance
- Accessibility
- Notifications

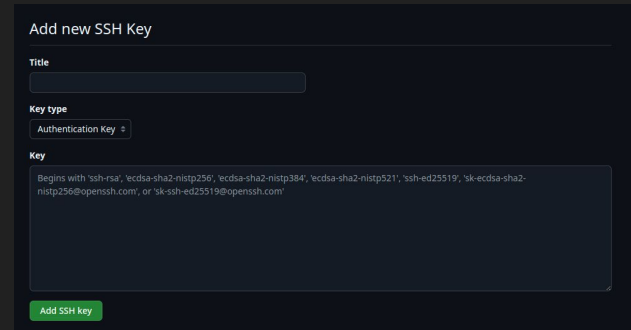
Access

- Billing and plans
- Emails
- Password and authentication
- Sessions
- SSH and GPG keys**
- Organizations



SSH keys New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.



Add new SSH Key

Title

Key type

Authentication Key

Key

Begin with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

Add SSH key

Using git

Creating a new repo

Creating a new repo is very easy, GitHub tells you how to do it

- First create folder for your project
 - `mkdir your-repo`
 - `cd your-repo`
- Then setup git
 - `git init` - initializes git in the folder
 - `git branch -M main` - create main branch and switch to it
 - `git remote add origin <your repo>` - add upstream remote
- You are done

Dashboard

RobbeDGreef

Top repositories New

Find a repository...

Create a new repository

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

Required fields are marked with an asterisk ().*

Repository template
No template

Start your repository with a template repository's contents.

Owner * Repository name *
RobbeDGreef /

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

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

Add .gitignore
.gitignore template: None
Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license
License: None
A license tells others what they can and can't do with your code. [Learn more about licenses](#).

You are creating a public repository in your personal account.

Create repository

...or create a new repository on the command line

```
echo "# git-seminar" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:RobbeDGreef/git-seminar.git
git push -u origin main
```

git workflow

Files go through multiple stages in a typical git workflow

- **untracked** - files are not tracked by the git system
- **changed** - files are tracked by the git system and have been changed since the last commit
- **staged** - files are “ready to be committed”
- **Committed** - files have been committed in git, they are up to date with gits database

First commit

Once git is initialized, we can add files to the system

- Create file README.md
 - ``vim README.md``
 - Add some information about your project
- Add file to **staged** files
 - ``git add README.md``
- **Commit** all staged files
 - ``git commit -m "First commit"``
- **Push** all commits to the remote git server
 - ``git push -u origin main``

tip:

You can just write ``git commit`` to get a editor in which you can write the commit message

Syncing with the remote

Pushing code to a remote

- `git push -u origin main` -
 - Push code and set the upstream to the origin/main branch`
- `git push`
 - Once the upstream has been set, this is a shorthand that pushes the code into the remote`
- `git push --force`
 - [DANGEROUS] pushes code and overwrites the remote branch with your local code`

Syncing with the remote

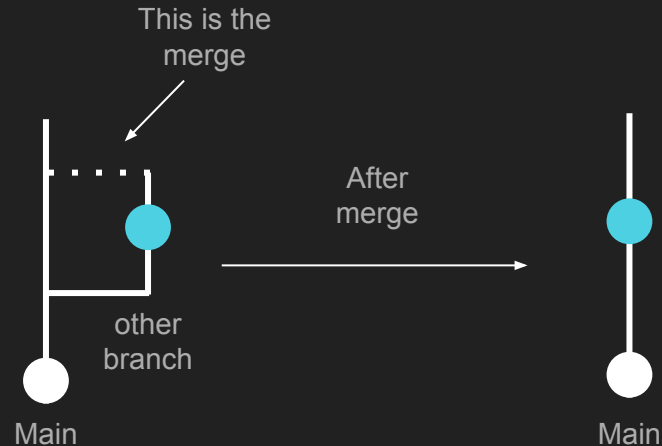
Pulling code from a remote

- `git pull``
 - Fetches changes from remote and applies them
- `git fetch``
 - Fetches changes from remote but does not apply them
- `git reset --hard origin/<branch>``
 - [DANGEROUS] remove all your local changes and reset your local code with the remote branch

Merging code

Merging code from one branch into the other

- `git merge <other-branch>`
 - Merge code from the other-branch into the current branch you are in



Rebasing onto branch

Rebase one branch on top of another branch

- `git rebase <other-branch>`
 - Rebase your current branch on top of the other branch



Working with branches

- `git checkout <branch>`
 - Set the given branch as the current branch you are working on
- `git checkout -b <branch>`
 - Create a new branch called <branch> and switch to it
- `git branch`
 - List all existing branches
- `git branch -d <branch>`
 - Delete an existing branch (locally)
- `git branch -m <new-name>`
 - Rename branch locally

.gitignore

File that specifies which files git has to ignore

- You can use wildcards `*`
 - To ignore all files of a certain type. e.g. `*.pyc` ignores all Python bytecode files
- You can list files you want to keep for yourself

```
# dependencies
/node_modules
/.pnp
.pnp.js
.yarn/install-state.gz

# testing
/coverage

# next.js
/.next/
/out/

# production
/build
```

Contributing to an existing repo

Clone an existing repository

- `git clone <repo>`

We are going to work on

- <https://github.com/RobbeDGreef/git-seminar>

So please clone this repo

tip:

Sometimes for large project, git submodules are used. These are references to other git repos inside the project. You often need to clone them too to build the project. This can be done with `git clone --recurse-submodules`

Contributing to an existing repo - forking

The typical workflow for contributing goes as follows:

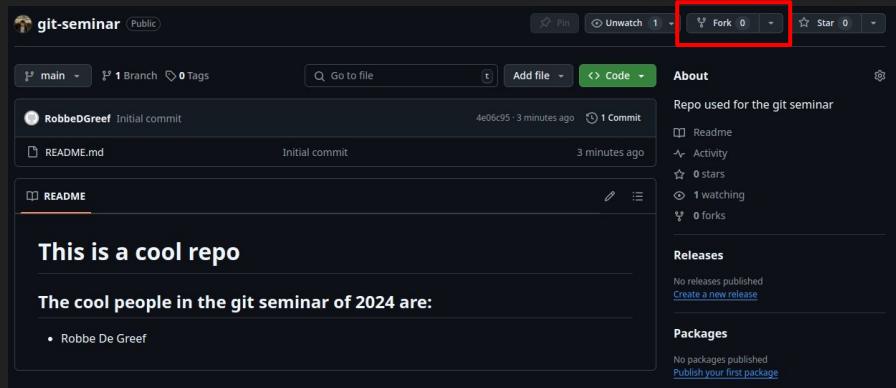
- You **fork** the repo
- You make your own changes
- You create a **pull request**
- That **pull request** is then **merged** or **rejected**

tip:

Sometimes a pull request is also called a merge request

Fork the repo

- Press the fork button
- Click create fork



Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Required fields are marked with an asterisk ().*

Owner * **Repository name ***

Choose an owner / git-seminar

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

Repo used for the git seminar

Copy the main branch only
Contribute back to RobbeDGreef/git-seminar by adding your own branch. [Learn more.](#)

Create fork