Basic usage of Git and GitHub

Hubert Rybka

October 22, 2025

Contents

1 Introduction

This tutorial introduces some basic concepts and the setup process of **Git** and **GitHub**, tools widely used for version control and collaboration in software and research projects. It is published as a learning resource for students attending the 2025/2026 Basics of Machine Learning (Podstawy Uczenia Maszynowego) course at Jagiellonian University, Faculty of Chemistry.

2 What Are Git and GitHub?

- **Git** is a *version control system* that records changes to files over time, allowing you to revert to earlier versions or explore the history of your project.
- **GitHub** is an online platform that hosts Git repositories and allows people to collaborate on them.

Think of Git as a local notebook that tracks every change you make and GitHub as an online library where you can store and share those notebooks.

3 Installing Git

3.1 Windows

If you already have **Miniconda** installed, you can install Git directly through the Anaconda environment without downloading a separate installer.

1. Open the **Anaconda Prompt** from your Start Menu.
2. Type the following command and press `Enter`:

```
conda install git
```

3. When prompted, type `y` (for "yes") and press `Enter` to confirm the installation.

3.2 Linux (Ubuntu/Debian)

Open a terminal and type:

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

Installing Git is not necessary on Linux machines in our computer lab, as the Faculty administrators took care of it already!

4 Configuring Git

Once Git is installed, you need to set up your user information so that Git can record who made each change.

Open a terminal (or Anaconda Prompt on Windows) and type:

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

You can check your configuration with:

```
git config --list
```

5 Creating a GitHub Account and Repositories

1. Go to `https://github.com` and create an account.
2. After logging in you may click the `+` icon in the top-right corner and choose `New repository` to create an empty repository which will store some project of yours.
3. **During our classes we will not need to work with a completely empty code repository**. What you need to do instead is create your own fork of the course repository `https://github.com/hubertrybka/pum-25` and clone it. This fork is your own personal copy of the main course repository (containing all the exercises), but unlike the course repository, you can commit any changes you like to your own fork. Those changes will be, for example, your solutions to the exercises.

6 Cloning an Existing Repository

A *repository* (or *repo*) is a project folder that Git tracks. It may contain files and directories; usually those are text files in which we store our code. The main goal of Git is keeping track of changes we make in those files, so that we can identify who and when was responsible for certain changes in our project files, and even use it as a "time machine" if we accidentally break something important in the code and want to go back to the last working version.

If you want to copy someone else's project to your computer, use:

```
git clone https://github.com/username/robocop.git
```

This creates a new folder with all the project's files and history. Do not forget to change the working directory in your console to the newly-cloned repo.

```
cd robocop
```

In our case, we used `git clone` command to download our own forks of the original course repository:
`https://github.com/hubertrybka/pum-25`.

7 Creating a Local Repository

This section of the tutorial will not be useful during the classes as we are not creating any local repositories from scratch, but it will help you when creating your first code repository for some project of yours.

1. Create a new folder for your project and navigate to it:

```
mkdir my_project
cd my_project
```

2. Initialize Git in this folder:

```
git init
```

   This creates a hidden folder called `.git`, where Git stores all version history. From now on, you can use git to track changes in this directory, according to the workflow explained in the following paragraph.

8 Basic Git Workflow

Here are the most common steps in a Git workflow:

1. **Check status:**

```
git status
```

   Shows which files are new, modified, or ready to be saved.

2. **Add files:**

```
git add filename
```

   or to add all files:

```
git add .
```

3. **Commit changes:**

```
git commit -m "Write a short message describing the
   change"
```

   The message is very important, especially when working on a project in collaboration with other people - it should be brief and describe exactly what changes this commit introduces.

4. **View history:**

```
git log
```

5. **Push changes to GitHub:**

   If you are working on a repository cloned from GitHub, you can ignore the first step. If you started from an empty local repository instead, you first need to provide the address of the remote repo to which the changes will be pushed.

```
git remote add origin https://github.com/username/
   robocop.git
```

```
git push -u origin master
```

   In order to perform this step successfully, you need to authorize your access to GitHub first, as explained in the following paragraph.

9 Authorizing GitHub Access

Before Git can communicate with GitHub (for example, to *push* - upload or *pull* - download changes from the remote repo), you need to authorize your local machine to access your GitHub account. There are two main ways to do this:

1. Using the **GitHub CLI tool** (`gh auth login`)
2. Using a **Personal Access Token (PAT)** for manual setup

9.1 Option 1: Using the GitHub CLI

The `gh` command is part of the official GitHub Command-Line Interface (CLI), which simplifies authentication.

1. Install the GitHub CLI:

```
conda install gh --channel conda-forge
```

   This should already be installed in our conda environment

2. Once installed, log in to GitHub by typing:

```
gh auth login
```

3. Follow the prompts:
   - Choose `GitHub.com` as the host.
   - Choose the protocol `HTTPS`.
   - When asked how to authenticate, select `Login with a web browser`.
   - The command will open a web browser window for you to log into GitHub and authorize access.
4. When finished, verify the login with:

```
gh auth status
```

If successful, it should display your GitHub username and confirm that the authentication is active.

Advantages: This method is simple, secure, and recommended for beginners. Automatically store your credentials securely.

9.2 Option 2: Using a Personal Access Token (PAT)

If you prefer not to install the GitHub CLI, you can manually generate a Personal Access Token and use it as a password when pushing code.

1. In your browser, go to: `https://github.com/settings/tokens`
2. Click on `Generate new token` (or `Fine-grained tokens` for newer accounts).
3. Give the token a descriptive name, such as `Laptop-Git`.
4. Set an expiration date (for example, 90 days).
5. Under permissions, enable at least:
    - `repo` (full control of private repositories)
6. Click `Generate token`, then copy it. (You will not be able to see it again!)

Important: Keep your token private — treat it like a password.

9.3 Using the Token in Git

The next time you push to GitHub, Git will ask for your username and password:

```
git push -u origin master
```

Enter:

- Your GitHub username
- The token (instead of your password)

To avoid entering the token every time, you can tell Git to store your credentials securely:

```
git config --global credential.helper store
```

Then, after the first successful login, your token will be saved locally.

10 Summary of Common Commands

Command	Description
`git init`	Create a new Git repository
`git status`	Show the current state of the repo
`git add file`	Stage a file for commit
`git commit -m "msg"`	Save staged changes with a message
`git log`	Show commit history
`git remote add origin URL`	Link local repo to GitHub
`git push`	Upload changes to GitHub
`git pull`	Download updates from GitHub
`git clone URL`	Copy an existing repository

11 Further Resources

- Official Git documentation: `https://git-scm.com/doc`
- GitHub Guides: `https://guides.github.com/`
- Interactive tutorial: `https://learngitbranching.js.org/`

12 Exercise: your first GitHub repo

12.1 Goal

Create a simple project, initialize it as a Git repository, connect it to a GitHub remote repository, and push your first commit.

12.2 Prerequisites

- A GitHub account.
- Git installed (or installed via Anaconda/Miniconda).
- If on Windows, open **Anaconda Prompt** (or Miniconda Prompt). On macOS/Linux, open your terminal.
- You should be authenticated with GitHub (e.g., via `gh auth login` or by using a Personal Access Token). See the tutorial section on authentication if needed.

12.3 Steps

1. **Create a new repository on GitHub (remote).**
   - In your web browser, go to `https://github.com` and sign in.
   - Click the `+` button (top right) and choose `New repository`.
   - Click `Create repository`. Copy the repository URL, e.g. `https://github.com/yourusername/exercise.git`.
2. **On your computer: create a project folder and initialize Git.**

```
mkdir exercise   # mkdir creates a new folder
cd exercise      # cd navigated to that folder
git init         # initializes git
```

3. **Add a simple text file with any contents you like to the exercise folder and check the status.**

```
git status
```

Expected output snippet from `git status`:

```
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be
     committed)
    new-file.txt
```

4. **Stage and commit the file.**

```
git add new-file.txt
git commit -m "Add new-file.txt"
```

5. **Set the main branch name (recommended).** Some Git installations still default to `master`. GitHub uses `main` by default, so set your local branch:

```
git branch -M main
```

6. **Link the local repository to the GitHub remote.**

```
git remote add origin https://github.com/yourusername/
   exercise.git
git remote -v
```

Expected output from `git remote -v`:

```
origin  https://github.com/yourusername/exercise-demo.
   git (fetch)
origin  https://github.com/yourusername/exercise-demo.
   git (push)
```

7. **Push your commit to GitHub.**

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

Notes:

- If asked for credentials: enter your GitHub username and then your password. If your account requires a Personal Access Token (PAT), paste the token instead of a password.
- If you used `gh auth login` earlier, the push should not ask for your password.

8. **Verify on GitHub.**

- Open the repository page (`https://github.com/yourusername/exercise.git`) in your browser.
- You should see `new-file.txt` and your commit message listed.