

Manual Git → GitHub

This document is the result of the Junior Enterprise for Science and Technology team's internal training workshop held on November 11, 2020.

Version: v1.00 (2020-11-14)

Author: [Tiago Tamagusko](#)

License: MIT © [Tamagusko](#) & [JEST](#).

Installation

1. Install [Git](#).
2. Create a [GitHub](#) account.

Git was installed? At the terminal:

```
1 git --version
```

Configuring your Git environment. At the terminal:

```
1 # Set your name for version history
2 git config --global user.name "first_name last_name"
3 # Set your email for version history
4 git config --global user.email "valid_email"
5 # Set automatic staining of command line for Git (optional)
6 git config --global color.ui auto
7 # Save credentials (optional)
8 git config --global credential.helper store
9 # View your config
10 git config --list
```

Topics

[What is Git?](#)

[Git workflow](#)

[Basic Git commands](#)

[Tricks to make your life easy](#)

[Creating a local repository](#)

[Ignoring files](#)

[Track works in project](#)

[Contribute to an open source project](#)

What is Git?

Git is distributed Version Control Software (VCS). Versioning is a way to keep a project from developing over time, while storing a project history. Each developer working has a local Git repository, which has a copy of the entire repository. This developer can change and test functionality and only make the contribution of the ready/correct part. This is different from the dropbox, where every saved change is synchronized with the main folder. With Git, the chance of you breaking the project substantially decreases.

Git + GitHub workflow

When you work with Git and GitHub, you have a local repository (working folder) and a remote repository (github.com). The advantage is that you can work on your features locally, and when you want, you can send the improvements to the remote repository (git push).

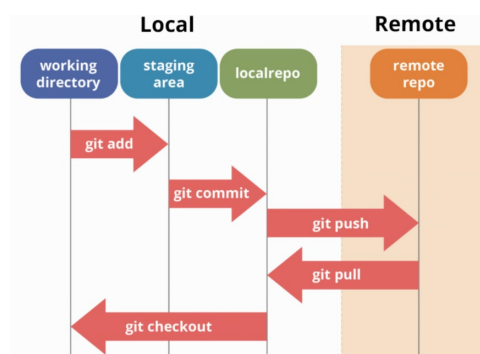


Figure 1: Workflow & Commonly Used Commands [4]

It is considered good practice to divide each feature developed into a branch (Fig. 2), this facilitates future revisions and corrections.

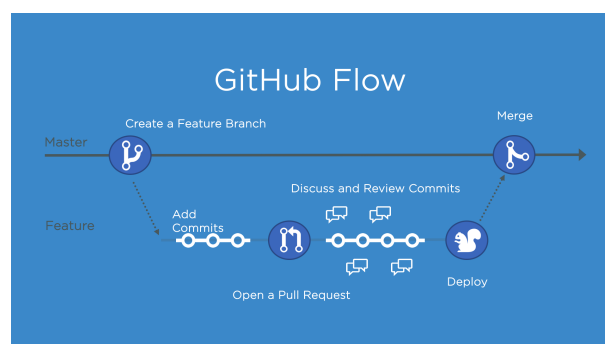


Figure 2: Recommended Project Flow [5]

Basic Git commands

```
1 # Create a Git repository:
2 git init
3 # Clone a remote repository:
4 git clone
5 # Add or update changes:
6 git add file_name
7 # Commit your added content:
8 git commit -m "descriptive message"
9 # Show modified files in working directory:
10 git status
11 # Manage your branches:
12 git branch
13 # Access existing branches:
14 git checkout branch_name
15 # Show history for the current branch:
16 git log
17 # Merge branches:
18 git merge
19 # Pull from the remote repository:
20 git pull
21 # Transmit local branch commits to the remote repository branch:
22 git push
23 # Show help for any command:
24 git command_name -h
```

Tricks to make your life easy

```
1 # Create branch and select it using:
2 git checkout -b function_description
3 # Discard uncommitted changes:
4 git reset --hard
5 # Fix a commit message:
6 git commit --amend -m "new message"
7 # See difference between branches:
8 git diff origin/branch1 origin/branch2
9 # Rename Branch:
10 git branch -m old_name new_name
11 # Return to the previous branch:
12 git checkout -
13 # Remove file from last commit:
14 git rm ---cached file_to_remove
15 git commit ---amend
```

Creating a local repository

At terminal, in the working directory type:

```
1 # Init local dir as a Git repository:
2 git init -b main
3 # Add all files to the local repository:
4 git add .
5 # Commit files and prepares them to be pushed:
6 git commit -m "first commit"
7
8 # Go to github.com and create a new empty repository with name_repo.
9
10 # Sets the new remote repository:
11 git remote add origin https://github.com/$user/$name_repo.git
12 # View the remote repository
13 git remote -v
14 # Push local changes to remote repository
15 git push origin main
```

Ignoring files

Create a file named .gitignore:

linux/mac: touch .gitignore

windows: echo . > .gitignore

At terminal, in the working directory type:

```
1 # ignore file (don't remove it)
2 git rm --cached file_name
```

see gitignore.io and [github/gitignore](https://github.com/gitignore) for more details.

Track works in project

```
1 # Show logs for certain file
2 git log file_name
3 # Show logs for the certain author
4 git log --author=user_name
5 # Show logs in a graph
6 git log --graph
7 # Show logs in just one line
8 git log --oneline
```

Contribute to an open source project [3]

```
1 # Fork the repository to your account
2 git clone https://github.com/your-username/name-of-repo.git
3 # Enter into folder
4 cd name-of-repo
5 # Vet remote repository path
6 git remote add upstream https://github.com/not-your-username/name-of-repo.git
7 # Verify remote and local branch
8 git status && git remote -v
9 # Manually sync up with upstream
10 git fetch upstream
11 # Change master/main branch to pointing upstream
12 git branch --set-upstream-to=upstream/main main
13 # Note: in old projects it can be master instead of main.
14
15 # Verify
16 git status
17 # Expected output
18 $ upstream/main
19 # Updates the local repository, if necessary
20 git pull
21 # Create your feature
22 git checkout -b my_feature_name
23 # Commit your changes
24 $ git commit -m "description of fix"
25 # Push your feature branch to remote repository
26 git push origin feature-branch
27 # Delete local branch (optional)
28 git branch -D my_feature_name
29 # Delete branch from remote repository (after merge/optional)
30 git push origin --delete my_feature_name
```

Found an error or have any suggestions?

Send your suggestions to geral@jest.pt or access [workshop-git](#) and make a PR in the file: MANUAL.md.

References

- [1] [Git Guide](#) - Explanation of almost everything you need on github.
- [2] [Git Cheat Sheet](#) - A quick reference to the Github commands.
- [3] [Open Source Git Workflow](#) - Open Source Git Workflow (an overview).
- [4] [Git workflow](#) - Workflow & Commonly Used Commands.
- [5] [GitHub flow](#) - The GitHub flow.