Manual Git → GitHub

This document is the result of the Junior Enterprise for Science and Technology team's internal training.

Version: v2.0 (2021-05-29)

Installation

1. Install Git.
2. Create a GitHub account.

Git was installed? At the terminal:

```
git --version
```

Configuring your Git environment. At the terminal:

```
# Set your name for version history
git config --global user.name "first_name last_name"
# Set your email for version history
git config --global user.email "valid_email"
# Set automatic staining of command line for Git (optional)
git config --global color.ui auto
# Save credentials (optional)
git config --global credential.helper store
# View your config
git config --list
```

Topics

What is Git?

Git is distributed Version Control Software. 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 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). So, GitHub is a server in the cloud that hosts repositories with git technology.

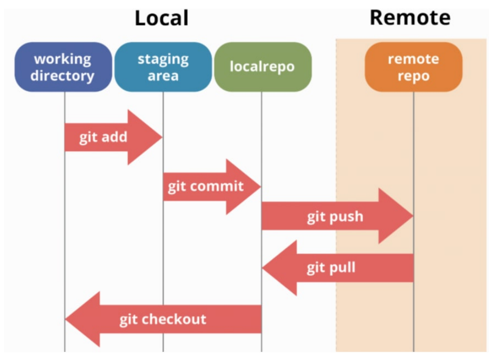

Figure 1: Workflow & Commonly Used Commands [4]

It is considered good practice to divide each feature developed into a branch (Fig. 2).

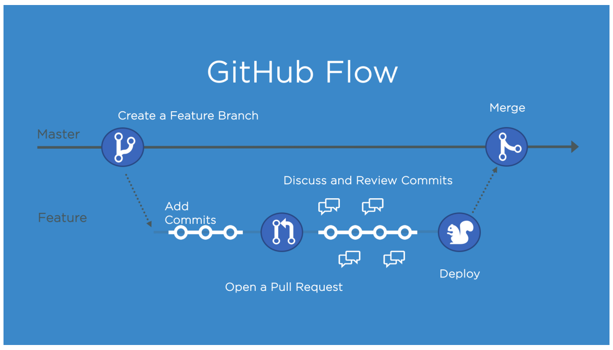

Figure 2: Recommended Project Flow [5]

Basic Git commands

```
# Create a Git repository:
git init
# Clone a remote repository:
git clone
# Add or update changes:
git add file_name
# Commit your added content:
git commit -m "descriptive message"
# Show modified files in working directory:
git status
# Manage your branches:
git branch
# Access existing branches:
git checkout branch_name
# Show history for the current branch:
git log
# Merge branches:
git merge
# Pull from the remote repository:
git pull
# Transmit local branch commits to the remote repository branch:
git push
# Show help for any command:
git command_name -h
```

Tricks to make your life easy

```
# Create branch and select it using:
git checkout -b function_description
# Discard uncommitted changes:
git reset --hard
# Fix a commit message:
git commit --amend -m "new message"
# See difference between branches:
git diff origin/branch1 origin/branch2
# Rename Branch:
git branch -m old_name new_name
# Return to the previous branch:
git checkout -
# Remove file from last commit:
git rm ---cached file_to_remove
git commit ---amend
```

Creating a local repository

At terminal, in the working directory type:

```
# Init local dir as a Git repository:
git init -b main
# Add all files to the local repository:
git add .
# Commit files and prepares them to be pushed:
git commit -m "first commit"

# Go to github.com and create a new empty repository with name_repo.

# Sets the new remote repository:
git remote add origin https://github.com/$user/$name_repo.git
# View the remote repository
git remote -v
# Push local changes to remote repository
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:

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

see gitignore.io and github/gitignore for more details.

Track works in project

```
# Show logs for certain file
git log file_name
# Show logs for the certain author
git log --author=user_name
# Show logs in a graph
git log --graph
# Show logs in just one line
git log --oneline
```

Contribute to an open source project [3]

```
# Fork the repository to your account
git clone https://github.com/your-username/name-of-repo.git
# Enter into folder
cd name-of-repo
# Vet remote repository path
git remote add upstream https://github.com/not-your-username/name-of-
    repo.git
# Verify remote and local branch
git status && git remote -v
# Manually sync up with upstream
git fetch upstream
# Change master/main branch to pointing upstream
git branch --set-upstream-to=upstream/main main
# Note: in old projects it can be master instead of main.

# Verify
git status
# Expected output
$ upstream/main
# Updates the local repository, if necessary
git pull
# Create your feature
git checkout -b my_feature_name
# Add your changes
git add file_name
# Commit your changes
git commit -m "description of fix"
# Push your feature branch to remote repository
git push origin feature-branch
# Delete local branch (optional)
git branch -D my_feature_name
# Delete branch from remote repository (after merge/optional)
git push origin --delete my_feature_name
```

Please direct bug reports and pull requests to the GitHub page. To contact me directly, send an email.

– Tiago

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.