UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

GIT HANDBOOK

by Mariana Paula Velasco Ortiz
ID: 260735

November 2024

TABLE OF CONTENT

*This handbook is not an official version by any of the platforms mentioned.

INTRODUCTION

What is Git?

Is a command-line tool used for track changes in your files and manage versions of your project. It runs on your local computer and doesn´t require an internet connection.
Git is a distributed version control system that enables GitHub activities. Then **GitHub** is a platform that leverages Git to enable online collaboration and repository hosting.
You can use Git without GitHub, but you can't use GitHub effectively without Git.

Why should students use it?

- Simplifies teamwork for group projects (like final assignments or team tasks).
- Maintains a history of changes, making it easier to identify and fix mistakes.
- Essential skill for the professional world.

*

*This handbook is not an official version by any of the platforms mentioned.

INSTALLATION

*

VISION

- On Windows: Use Git for Windows.
- On macOS/Linux: Often pre-installed; if not, install it via the terminal (cmd or powershell):

```
sudo apt-get install git # for Linux
```

MISSION

- On Windows:
https://windows.github.com
- On macOS:
h▯ps://mac.github.com
- For any platforma:
h▯p://git-scm.com

*This handbook is not an official version by any of the platforms mentioned.

DEFINITIONS

*

REPOSITORY

A storage location where all the files of a project, along with their entire history of changes, are tracked and managed.

LOCAL REPOSITORY

The repository on your computer.

REMOTE REPOSITORY

A repository hosted on a server (like GitHub, GitLab, or Bitbucket).

COMMIT

A record of changes made to a project's files. It acts as a "checkpoint" that captures the current state of the repository at a specific point in time. Commits allow you to save your work, document what was modified, and revisit those changes later if needed.

WORKING TREE

Three basic areas:

WORKING DIRECTORY

The files in your project folder.

DEFINITIONS

*

STAGING AREA

Files prepared for the next commit.

REPOSITORY

Where commits are stored.

Example: You change your project code and save it in Git as a commit, so later, you can return to that point if something goes wrong.

BRANCH

A separate line of development within a repository. It allows you to create an independent version of your project where you can experiment, add features, or fix issues without affecting the main codebase. Also, multiple people can work on different branches simultaneously and later combine their work. Each branch keeps a unique record of changes specific to its purpose.

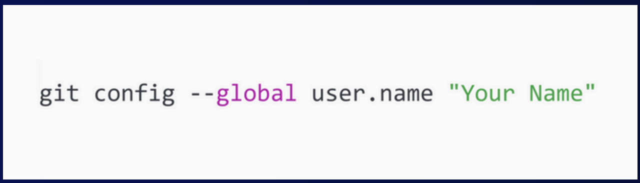
STASH

Temporarily saves changes without commiting them.

BASIC CONFIGURATION

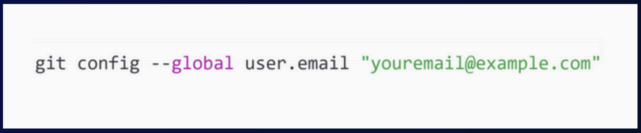
*

Configure your user´s information for all local repositories.



```
git config --global user.name "Your Name"
```

Set up the name you want to be connected to your local repositories.



```
git config --global user.email "youremail@example.com"
```

Set up the email you want to be connected to your commits.

*This handbook is not an official version by any of the platforms mentioned.

CREATING *REPOSITORIES

```
git init [project-name]
```

Create a new local repository with an especific name.

```
git clone [url]
```

Clone an existing repository. Also, download a project and all its versions

*This handbook is not an official version by any of the platforms mentioned.

REMOTE REPOSITORIES

*

Connecting to a Remote Server (like GitHub):

Add a remote:

```
git remote add origin <repo-URL>
```

Push changes:

```
git push origin main
```

Pull changes:

```
git pull origin main
```

Tip for Students: If you're working in a team, communicate about who is pushing or pulling changes to avoid conflicts.

*This handbook is not an official version by any of the platforms mentioned.

MAKE CHANGES

```
git status
```

List all the new or modificated files that need confirmations for adding.

```
git diff
```

Show the files´ differences that haven´t been sent to the staging area.

```
git add [file]
```

Add a file to the staging area.

*This handbook is not an official version by any of the platforms mentioned.

MAKE CHANGES

```
git diff --staged
```

Show the file´s differences between the one on the stage area and the final version.

```
git reset [file]
```

Move off the file of the stage area, but the keeps the content.

```
git commit -m "[descriptive message]"
```

Save a commit.

*This handbook is not an official version by any of the platforms mentioned.

WORKING WITH BRANCHES

```
git branch
```

List all branches in the current repository.

```
git branch [branch-name]
```

Create a new branch.

```
git checkout [branch-name]
```

Switch to the specified branch and update the working directory.

WORKING WITH BRANCHES

```
git merge [branch]
```

Merge the history of the specified branch into the current branch.

```
git branch -d [branch-name]
```

Delete the specified branch.

OTHER COMMANDS

```
git rm [file]
```

Delete the file from the working directory and stage the deleted file.

```
git rm --cached [file]
```

Remove the file from version control, but keep the file locally.

```
git mv [file-original] [file-renamed]
```

Rename the file and stage it for commit.

OTHER COMMANDS

```
git ls-files --other --ignored --exclude-standard
```

List all ignored files in this project.

```
git stash
```

Temporarily store all modified tracked files.

```
git stash pop
```

Restore the most recently saved files.

*This handbook is not an official version by any of the platforms mentioned.

OTHER COMMANDS

```
git stash list
```

List all stashed changes.

```
git stash drop
```

Remove the most recent stash.

```
git log
```

List the version history for the current branch.

```
git log --follow [file]
```

List the version history for the file, including rename changes.

OTHER COMMANDS

```
git diff [first-branch]...[second-branch]
```

Show the content differences between two branches.

```
git reset [commit]
```

Undo all commits after [commit], preserving changes locally.

```
git reset --hard [commit]
```

Discard all history and revert to the specified commit.

TIPS

- Write clear messages: A good commit message describes what you changed and why.
- Make small, frequent commits: Save changes regularly to make it easier to track issues.
- Collaboration: Assign tasks and use branches to avoid conflicts.

Final Tip: Practice! You can start by creating repositories for your assignments or personal projects. Over time, using Git will feel as natural as saving a file on your computer.

www.reallygreatsite.com +123-456-7890

THANK YOU

FOR YOUR ATTENTION

Get more information about using GitHub and Git. Send an email to the Training Team or visit our website to learn more.

training@github.com

training.github.com