
GitHub
Handbook
By Dariana Reynoso

CONTENT

INTRODUCTION TO GIT HUB

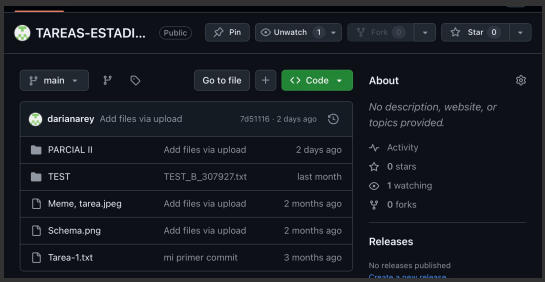
WHAT IS GITHUB?

A web-based plataform for version control and collaboration, primarily used by software developers to manage code. It uses GIT, a distributed version control system to track canges in proyects, allowing multiple developers to work on the same proyect simultaneously without conflicts.

KEY FEATURES

Version Control: Tracks and manages changes to code across versions.

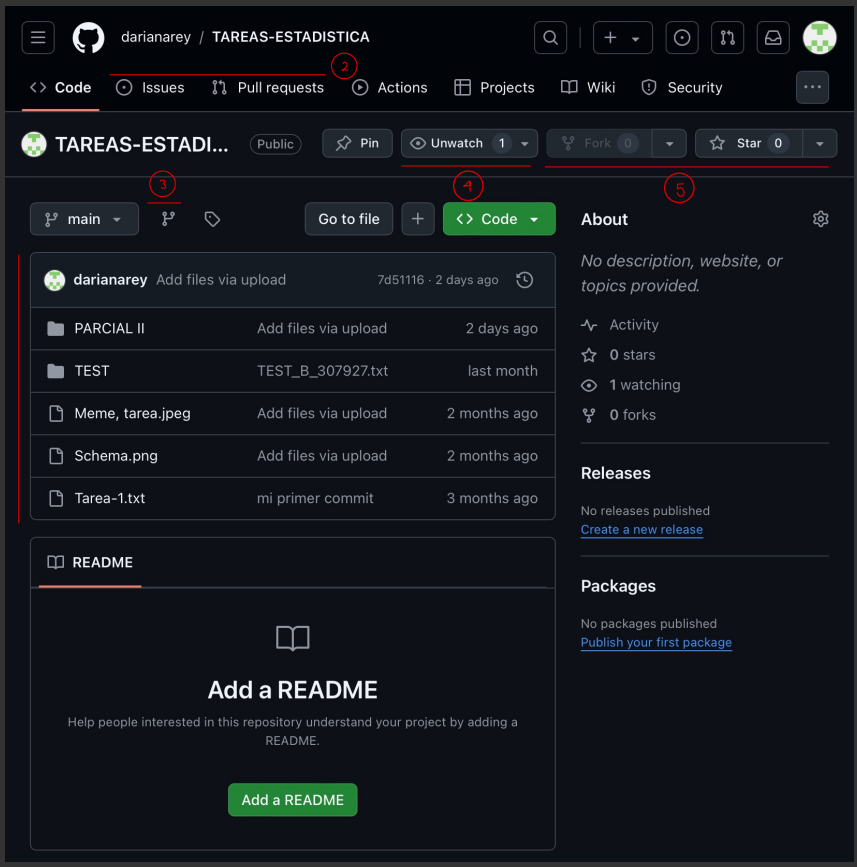
1. Repositories: Storage spaces for proyects where code and related files are stored



2. Colaboration Tools:
   - → Pull requests: Proposed code changes that others can review, discuss and approve.
   - → Issues: Used to track bugs, tasks or enhancements.
3. Branching and Merging: Enables developers to work on different features or bug fixes simultaneously and then merge them back into the main codebase
4. Hosting: Stores code securely and makes it accessible from anywhere.
5. Comunity Features:
   - → Stars: Users can "star" projects to show appreciation or bookmark them.
   - → Forks: Users can create their own copy of a repository to modify or experiment.

Integration with tools: Works seamlessly with tools like CI/CD pipelines proyect management tools and cloud plataforms.

ILUSTRATION



WHO USES GITHUB?

Developers: To collaborate on software proyects.
Organizations: For managing team proyects.
Open source contributors: To share and contribute to public proyects
Learners: To access tutorials, guides and open-source proyects.

Git has become a hub for millions of developers worldwide, offering both free and paid plans for private and public repositories.

INSTALLATION AND SIGN UP

INSTALL GITHUB FOR WINDOWS

→ Go to the weboside https://git-scm.com/ and downland the installer for windows.

→ Run the downloaded .exe file and follow the setup instructions.

→ During the installation, make sure to select the following options: "Git from the command line" to use Git from the terminal

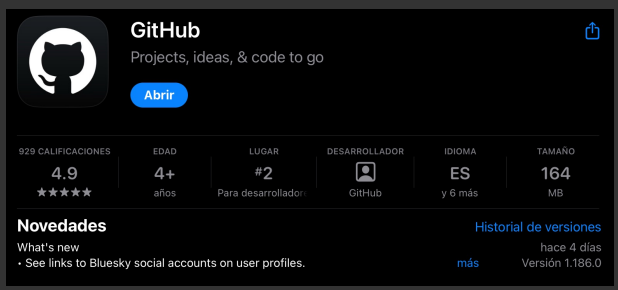
→ To verify that Git has been installed correctly, open the terminal (cmd or Powershell) and run **git --version.**

OPTIONAL: Install GitHub Desktop

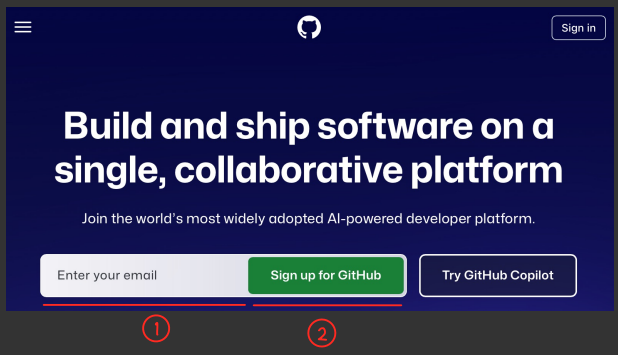
→ Download it from desktop.github.com

→ Follow the instructions to install and sync with your GitHub account.

GITHUB APP AVALIABLE FOR ANDROID AND iOS



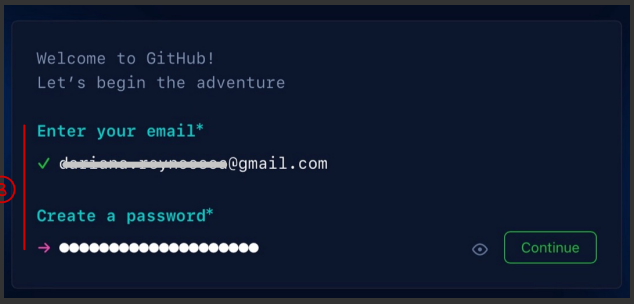
CREATE A GITHUB ACCOUNT



→ Go to github.com

① Enter your email

② Click on Sign up



③ Fill the necessary information
→ Follow the instructions to verify your account and set up your profile

BASIC CONCEPTS

REPOSITORY

Container that stores all the files and folders of a proyecr. along with its history of changes and versions. It can be hosted locally on your computer or remotely on platforms like GitHub.

TYPES

→ Local repository : Stored on your computer.
→ Remote repository : Stored on a server.

CREATE A LOCAL REPOSITORY IN GITHUB

→ Sign in github.com
→ Click "New repository"
→ Enter a name
→ Choose Public or Private.
→ Click "Create repository"
→ Link local Repository to Remote

(Go to cmd)
git init # Initialize Git (if not already)
git remote add origin https://github.com/your_username/your_repository.git
git branch -M main # Set main branch (optional)
git add . && git commit -m "Initial commit" && git push -u origin main
Clone an Existing Repository (if needed):
git clone https://github.com/username/repository-name.git

COMMITS

→ A commit is a "snapshot" of the changes in your proyect. Each commit saves the current state of the files in the repository along with a descriptive message.

→ Commites allow you to undo changes and review the proyect's history.

MAKE A COMMIT

→ Add files to the staging area: `git add file.txt`

→ Or to add all modified files: `git add`

→ Create a commit: `git commit -m "Clear description of the change".`

→ View the commit history: `git long`

→ Use `git log --online` for a simplified history.

BRANCHES

A branch is an independent line of development within a repository. It allows you to work on new features of fixes without affecting the main branch.

→ Branches are particularly useful for teams, as they enable parallel work.

MAIN BRANCHES

→ Main (master): The primary, stable branch.

→ Feature/xxx: Branches used for developing new features.

→ Hotfix/xxx: Branches used for urgent bug fixes.

CREATE AND SWITCH BRANCHES

→ Create a new branch: `git branch branch-name`

→ Swith to another branch: `git checkout branch-name`

→ Or staring from Git 2.23 use: `git swith branch-name`

MERGING BRANCHES

Once you finished working on a branch, you can merge it into the main branch to integrate the changes:

→ Git checkout main

→ Git merge branch-name
This merges the changes from branch-name into main.

MERGE

Process of combining changes from one branch into another. Typically, this involves integrating a feature branch or changes from a fork into the main branch or into another working branch.

WHEN DO YOU MERGE?

→ After completing work on a feature or bug fix.
→ When you need to sync changes from one branch (e.g. main) into another branch.
→ When you accept a Pull Request.

TYPES OF MERGE

→ Fast-Foward Merge: When the branch you're merging from has no divergent changes.
→ Three-way Merge: Two branches that have diverged.
→ Squash Merge: Combines all commits from a branch into a single commit before merging into the target branch.
→ Rebase Merge: Reapplies the branch commits on top of the target branch for a cleaner history.

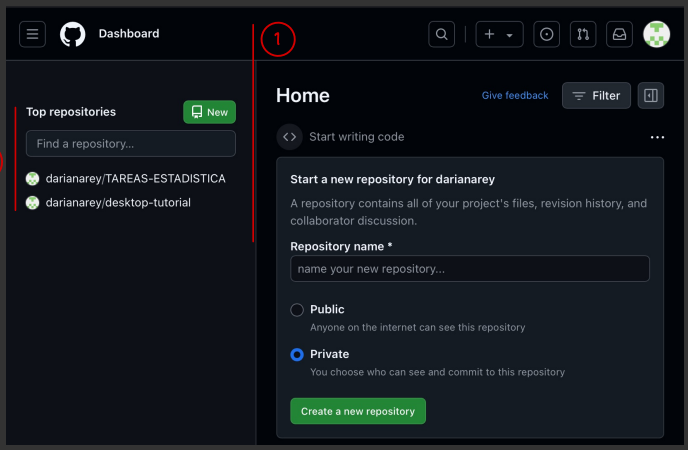
REMOTE

Reference to a repository hosted on a server, typically on a platform like GitHub, Gitlab or Bitbucket. It serves as the connection poin between your local Git repository and the repository stored on the server, enabling collaboration and synchronization between multiple developers.

KEY FEATURES OF A REMOTE

→ Connection to a Repository: Allows your local project to communicate with the version hosted online.
→ Pull changes: Download updates made by others to the remote repository.
→ Push Changes: Upload your local commits to the remote repository.

GITHUB INTERFACE OVERVIEW



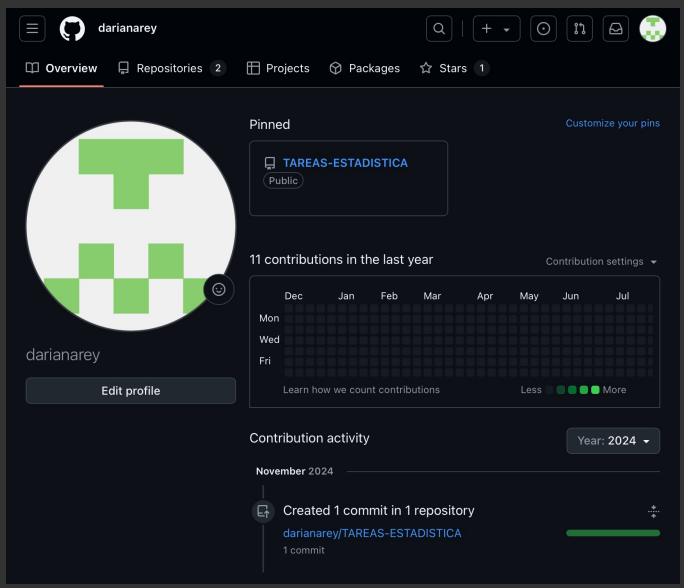
① DASHBOARD

The main hub of your GitHub account. It shows recent activity, repositories and organizations.

② REPOSITORIES

Every repository has:

- → Code Tab: View and manage files
- → Issues Tab: Track tasks, enhancements and bugs.
- → Pull requests Tab: Collaborate on proposed changes.
- → Actions Tab: Automate workflows using GitHub Actions.



③ PROFILE PAGE

Displays your bio, contributions and repositories. Customize your README.md to make your profile unique.

ESSENTIAL GIT COMMANDS

Basic commands every Git user should know to work efficiently with local and remote repositories.

INITIALIZE A REPOSITORY

Command: `git init`

This command creates a new Git repository in the current directory. If it doesn't exist a .git subfolder will be created to store all Git information about the proyect.

CLONE A REPOSITORY

Command: `git clone <URL>`

Use this command to copy a remote repository to your local machine.
github.com/user/repo.git

ADD CHANGES TO THE STAGING AREA

Command: `git add`

Moves modified files to the staging area, indicating to Git that they should be included in the next commit. Add a specific file: `git add file.txt`

MAKE A COMMIT

Command: `git commit -m "message"`

Creates a new commit with the changes in the staging area and assigns a descriptive message to it. Usage:

```
git commit -m "Added a new file".
```

Create a commite while automatically including all modified files (without using git add):

```
git commit -am "commit message".
```

CHECK THE REPOSITORY STATUS

Command: `git status`

Show the current status of the repository, including modified files, files in the staging area and untracked files.
Expected Output: You will see information about files that haven't been staged.

VIEW COMMIT HISTORY

Command: git long

Displays the commit history, including the commit ID (hash), author, date and message.
Useful option for a simplefied history: git log --online
To view the history of a specific file: git log file.txt

CREATE AND SWITCH BRANCHES

Comands: git branch

→ Create a new branch: git branch *branch name*
→ Switch to an existing branch: git checkout *branch name*

MERGE BRANCHES

Command: git merge *branch-name*

Use this commana to combine changes from one branch into another.
Usage: git checkout main, git merge feature.

PUSH CHANGES TO A REMOTE REPOSITORY

Command: git push
Usage: git push origin main
Push a specific branch: git push origin *branch-name*

UPDATE YOUR LOCAL REPOSITORY WITH REMOTE CHANGES

Command: git pull

Downloads and merges changes from the remote repository into your current branch.
Usage: git pull origin main.

DELETE FILES FROM A REMOTE REPOSITORY

Delete the file locally

git rm <file-name>

To delete files recursively in a directory: git rm -r <directory>

Commit the changes: git commit -m "Deleting <file name> from repository".

Push the changes to the remote repository: git push.

To stop tracking a file without deleting it locally:

git rm --cached <file name>

git commit -m "stopped tracking <file-name>"

git push

WORKING ON GITHUB

We can use GitHub to manage our projects, colaborate with other developers and make the most of the platform's features.

→ Link your local repository to a new repository on GitHub.

If you already have a local repository and want to connect it to GitHub

→ git remote add origin

https://github.com/your_username/repository-name.git

Verify that the link is correct: git remote

COLLABORATE ON A PROJECT

GitHub makes collaboration easy by allowing multiple people to work on the same proyect without interfering with each other's work.

COLLABORATION TOOL: FORKS

Fork: A copy of a repository created in your account, allowing you to make changes without affecting the original project.

TO FORK A REPOSITORY

→ Go to the repository you want to copy and click on "Fork"

→ Now you'll have a copy in your GitHub profile.

CLONING A FORK AND WORKING ON IT

→ Clone your fork to your local

git clone github.com/your_username/my-fork.git

PULL REQUEST

Request for the maintainers of the original project to review and, if appropriate, accept changes.

HOW TO CREATE A PULL REQUEST

→ Make the changes in a new branch.

→ Push your changes to your repository.

git push origin feature / new - functionality

→ Go to your repository on GitHub and click "Compare & pull request".

→ Add a little and a detailed description, then click "Create pull request".

Reviewing and Merging Pull Requests

→ Once you create a PR, other collaborators can review it and leave comments.

→ The repository owner can approve the PR and merge it using the "Merge pull request" button.

HOW TO REVIEW MY CODES

→ Access the pull request and use the "Files changed" tab to view the changes made.

→ Leave comments on specific lines of code if you identify issues or have suggestions.

→ Aprove the changes or request modifications before merging the PR.

TOOLS TO IMPROVE CODE REVIEW

→ GitHub Actions: Set up automated workflows to run tests and analyze code before a pull request is reviewed.

→ Check and CI/CD: Configure integrations to automatically validate that new code meets project standards.

COMMON ISSUES AND TROUBLESHOOTING

AUTHENTICATION ISSUES

Problem: Permission Denied (PublicKey)

Occurs when SSH Keys are not configured or incorrect.

Solution:

1. Generate an SSH Key (If not already done).
   ssh-keygen -t rsa -b 4096 -C "your.email@example.com"
2. Add the SSH Key to your GitHub account. Copy the key:
   cat ~/.ssh/id_rsa.pub
   Go to GitHub → Settings → SSH and GPG Keys → New SSH Key → Paste your Key.
3. Test the connection:
   ssh -T git@github.com

MERGE CONFICTS:

Problem: Conflicts During a Merge.

Occurs when changes in two branches overlap.

Solution:

1. Identify conflicts in the affected files (marked with <<<<<<<, =======, >>>>>>>).
2. Resolve conflicts by editing the file to keep the correct changes.
3. Stage and commit the resolved file:
   git add <file>
   git commit -m "Resolve merge conflict in <file>"

DETACHED HEAD

Problem: Working in Detached HEAD State.

Occurs when you check out a specific commit instead of a branch.

Solution:

1. Switch to a branch:
   git checkout main
2. If you've made changes and want to keep them:
   git checkout -b new-branch

OUTDATED LOCAL BRANCH.

Problem: Can't push due to Out-of-Date branch.

Occurs when your branch is behind the remote branch.

Solution:

1. Pull the latest changes:
   git pull origin <branch>
2. Resolve any conflicts if necessary.
3. Push your changes again:
   git push origin <branch>

INCORRECT COMMITS

Problem: Committed Changes to the wrong branch.

Solution:

1. Create a new branch to move the commits:
   git branch new-branch
   git reset --hard HEAD~n # Replace `n` with the number of commits to undo
2. Switch to the correct branch and cherry-pick the commits.
   git checkout correct-branch
   git cherry-pick <commit-hash>

LARGE FILES UPLOAD

Problem: "File exceeds GitHUB's size limit of 100MB"

GitHUb imposes a file size limit of 100MB for repositories.

Solution:

1. Use Git Large File Storage (LFS):
   git lfs track "*.filetype" # Replace filetype with your file's extension
   git add .gitattributes
   git add <large-file>
   git commit -m "Track large file with Git LFS"
   git push origin <branch>
2. Alternatively, host large files elsewhere (e.g, cloud storage).

MISSING FILES AFTER CLONING

Problem: Files are missing after cloning.

This occurs if some files are ignored by .gitignore or intentionally removed.

Solution:

Check the .gitignore file for patterns that might exclude files. If files are meant to be included, remove the pattern from .gitignore and add the files again:

```
git add <file>
git commit -m "Add missing file"
git push origin <branch>
```

PULL REQUEST ISSUES

Problem: Unable to merge a pull request

Occurs if there are unresolved conflicts or failed checks.

Solutions:

1. Pull the latest changes from the base branch.

```
git pull origin main
git merge main
```

2. Resolve any conflicts and push updates to the PR branch.

ACCIDENTAL FILE DELETION

Problem: Deleted a file by mistake.

Solution: Recover the file using Git

1. Identify the commit where the file existed.

```
git log -- <file>
```

2. Restore the file

```
git checkout <commit-hash> -- <file>
```

ERROR: "UPDATES WERE REJECTED BECAUSE THE TIP OF YOUR CURRENT BRANCH IS BEHIND".

Occurs when your local branch is out of sync with the remote branch.

Solution:

1. Pull the latest changes
   git pull origin <branch>
2. Push your changes again:
   git push origin <branch>

STALE FORK

Problem: your fork is outdated.

Occurs when the original repository (upstream) has new changes.

Solution:

1. Add the origin repository as a remote.
   git remote add upstream https://github.com/original-owner/repository.git
2. Fetch the latest changes:
   git fetch upstream
3. Merge or rebase them into your fork's branch.
   git checkout main
   git merge upstream/main

BEST PRACTICES FOR GITHUB

COMMIT OFTEN
Make small, atomic commits with clear messages

USE BRANCHES
Keep the main branch clean by working in feature branches.

PULL REGULARLY
Sync your branch with the main branch to prevent conflicts.

WRITE DESCRIPTIVE PULL REQUESTS
Include context and screenshots if applicable.

REVIEW CODE THOROUGHLY
Maintain code quality through peer reviews.

ADDITIONAL RESOURCES

- Official GitHub Documentation: docs.github.com
- Pro Git Book: git-scm.com/book/en/v2
- GitHub Learning Lab: lab.github.com
- Oh My Git! Game: ohmygit.org