TEMAS SELECTOS
DE ESTADISTICA

GitHub Handbook

PAOLO MÉRIGO FREGOSO - 307916
https://github.com/Paolo2113/ClaseEstadistica

INDEX

Table of Content

PAOLO MÉRIGO FREGOSO - 307916
https://github.com/Paolo2113/ClaseEstadistica

INTRODUCTION

What is GitHub?

GitHub is an online software development platform. It's used for storing, tracking, and collaborating on software projects.

Since its founding in 2008, GitHub has acquired millions of users and established itself as a go-to platform for collaborative software projects. This free service comes with several helpful features for sharing code and working with others in real time.

On top of its code-related functions, GitHub encourages users to build a personal profile and brand for themselves. You can visit anyone's profile and see what projects they own and contribute to. This makes GitHub a type of social network for programmers and fosters a collaborative approach to software and website development.

START FROM THE BEGIN

How to Get Started Using GitHub

1. Install GIT.
Install the latest version of Git on your device. You'll need Git installed to work with your GitHub repository. There are various ways to do this, so follow the recommendations on the Git website. The Git software is free.

2. Sign up for GitHub.
After installing Git, go to GitHub and create an account with your email address.

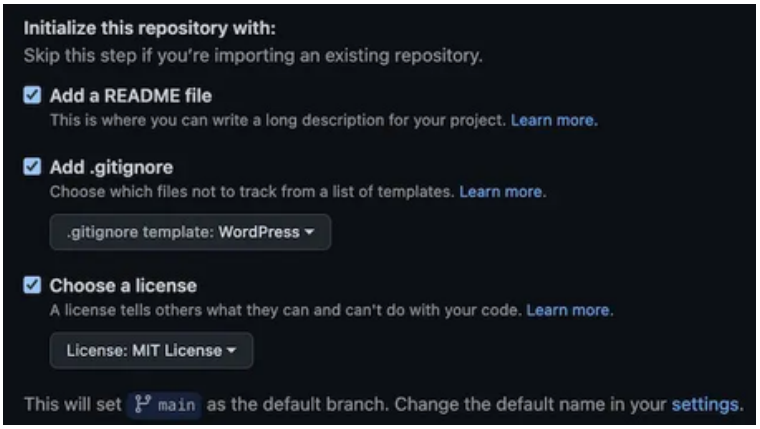
3. Start a repository.
Once your GitHub account is set up, you'll be taken to your dashboard. To start your first repository, click Create repository. This lets you keep all of your code for your new GitHub project in one place.

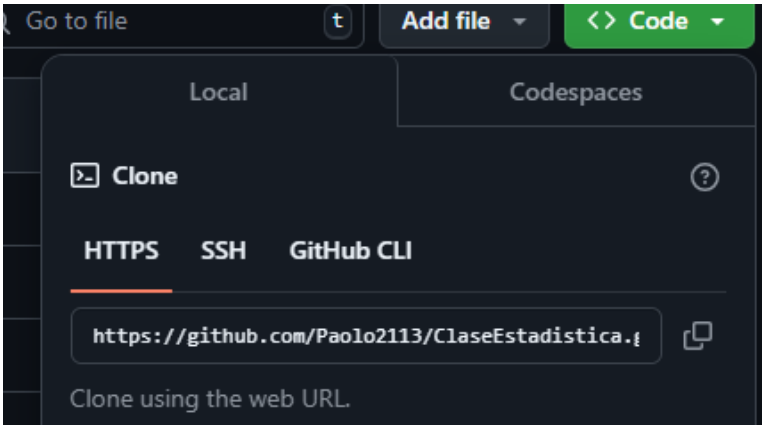
4. Name your project.
On the Create a new repository screen, enter your repository name and an optional description (you can change both later).

5. Add project details.
On the same screen, add a README file (a text file that describes your project, and a best practice in development), a .gitignore (which removes irrelevant files like .DS_Store), and a license for your project.
These details make it easier for collaborators to understand your project and any guidelines you'd like them to follow.

6. Create your repository.
Click Create repository. You'll be taken to your main repository page, which lists your files.

7. Create a local copy of your repository.
You'll now create a local copy of your GitHub repository (or in GitHub terms, "clone" your repository) where you'll edit your files and push your changes. On your main repository page, click the green Code button, then copy the HTTPS URL of your repository.
You already have a copy of your project on your hard drive, so why do you need to clone your repository? This is a best practice because it makes it easy to see when people added or removed files. It also makes it easier to fix merge conflicts.





ALMOST DONE

8. Choose a directory.
Open your terminal and navigate to the directory you want to place your repository copy.

9. Paste your repository URL.
In the terminal, enter git clone. After this paste in the repository URL that you previously copied. Your command should look like this:

```
{% call highlightSyntax('html') %}
git clone https://github.com/your-username/your-repo-name.git
{% endcall %}
```

10. Clone and check your copied repository.
Press Enter to clone the repository. You'll see a new file added to your local filesystem with your repository's name. If you open this file, you'll see it contains the files in your GitHub repository. These are copied versions of your repository's files that you can edit and then push back to your repository.

11. Create a new file in your new repository.
Let's end by creating a new file in your cloned repository, then pushing it to GitHub. In your local clone, create a new text file called hello.txt. In it, paste the text Hello, world! and save the file.

12. Prepare to push your files.
In the terminal, navigate to inside your cloned repository. Then, type git add . and press Enter. This prepares all files in your cloned repository to be pushed.

13. Commit your changes to the changelog.
In the terminal, type git commit -m "added hello.txt" and press Enter. This commits your changes to the changelog. The text in quotes is a comment briefly describing the purpose of the commit.

14. Push your cloned repository files.
Type git push origin main in the terminal and press Enter.

15. Check for your new file.
Back in your GitHub repository, you'll see your new file added. Now you're ready to get to work and begin collaborating on your new project.

THINGS YOU HAVE TO KNOW

Vocabulary

- BRANCH.- A branch represents an independent line of development. Branches serve as an abstraction for the edit/stage/commit process discussed in Git Basics, the first module of this series. You can think of them as a way to request a brand new working directory, staging area, and project history.
- HEAD.- Git's way of referring to the current snapshot. Internally, the git checkout command simply updates the HEAD to point to either the specified branch or commit. When it points to a branch, Git doesn't complain, but when you check out a commit, it switches into a "detached HEAD" state.
- HOOK.- A script that runs automatically every time a particular event occurs in a Git repository. Hooks let you customize Git's internal behavior and trigger customizable actions at key points in the development life cycle.
- MAIN.- The default development branch. Whenever you create a git repository, a branch named "main" is created, and becomes the active branch.
- TAG.- A reference typically used to mark a particular point in the commit chain. In contrast to a head, a tag is not updated by the commit command.
- VERSION CONTROL.- A system that records changes to a file or set of files over time so that you can recall specific versions later.
- WORKING TREE.- The tree of actual checked out files, normally containing the contents of the HEAD commit's tree and any local changes you've made but haven't yet committed.
- PULL REQUEST.- Pull requests are a feature that makes it easier for developers to collaborate using Bitbucket. They provide a user-friendly web interface for discussing proposed changes before integrating them into the official project.
- FORKING.- Instead of using a single server-side repository to act as the "central" codebase, forking gives every developer a server-side repository. This means that each contributor has not one, but two Git repositories: a private local one and a public server-side one.
- GITFLOW WORKFLOW.- The Gitflow Workflow streamlines the release cycle by using isolated branches for feature development, release preparation, and maintenance. Its strict branching model also lends some much needed structure to larger projects.

ESSENTIAL COMMANDS

Commands

A

Initialize a Git Repository

git init

Initializes a new Git repository. This command creates a new Git repository in the current directory.

B

Clone repository

git clon

Clones an existing Git repository. This command creates a copy of an existing repository on your local machine.

C

Add file to Staging

git add “file name”

Adds a file or directory to the staging area. This command prepares the changes for the next commit.

D

Commit with a message

git commit -m “commit message”

This command creates a new commit with the changes you made to your local repository.

E

Pull changes

git pull

Updates the local repository with changes from the remote repository. It pulls the changes from the remote repository and merges them with the local changes.

F

Push chnages

git push

This command pushes the local changes to the remote repository. It updates the remote repository with the changes you made locally.

G

Check status

git status

This command pushes the local changes to the remote repository. It updates the remote repository with the changes you made locally.

I

List all commits

git log

This command shows a list of all commits in the repository. It displays the author, date, and commit message for each commit the repository has.

LETS WORK TOGETHER!

Social Networking

Any GitHub user knows the platform is more than just a place to work on code. All GitHub users have profiles to display their projects, contributions, and activity on the site, and can see anyone's public-facing profile and repositories.

GitHub's social network is critical to its success, as it encourages developers to explore and contribute to open-source projects of all kinds. Previously, aspiring collaborators would have to personally contact project owners asking for permission to contribute.

With GitHub, it's as easy as forking a project and then making a pull request. A project owner can then vet someone's profile and past contributions before accepting their request.

GitHub also serves as a way to showcase projects for employers, acting as a portfolio of sorts. For example, recruiters often use GitHub to scout talent, since prospects' code is available for anyone to review.

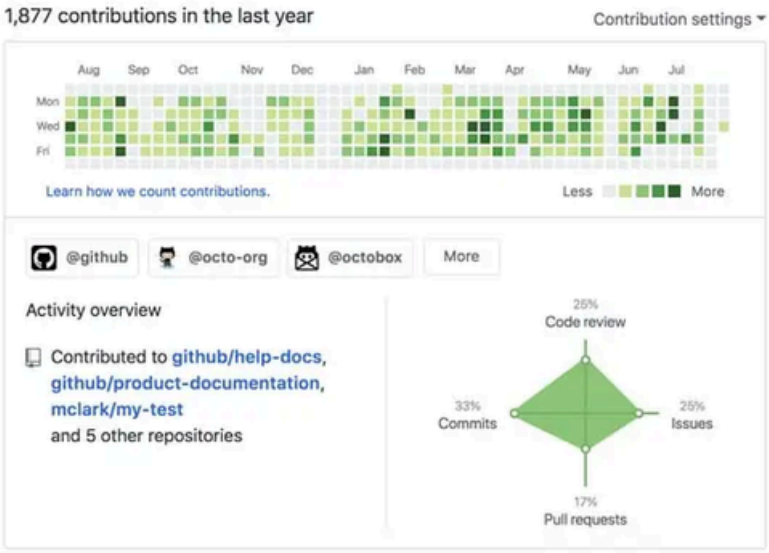
Open-Source Projects

Thanks to the benefits we've learned about, GitHub has fueled a surge of open-source collaboration, leading to the creation of many widely used software technologies. From CSS frameworks to data visualization libraries to a game you might spend too much time playing, a lot of impressive feats wouldn't be around without open GitHub repositories.

Private Repositories

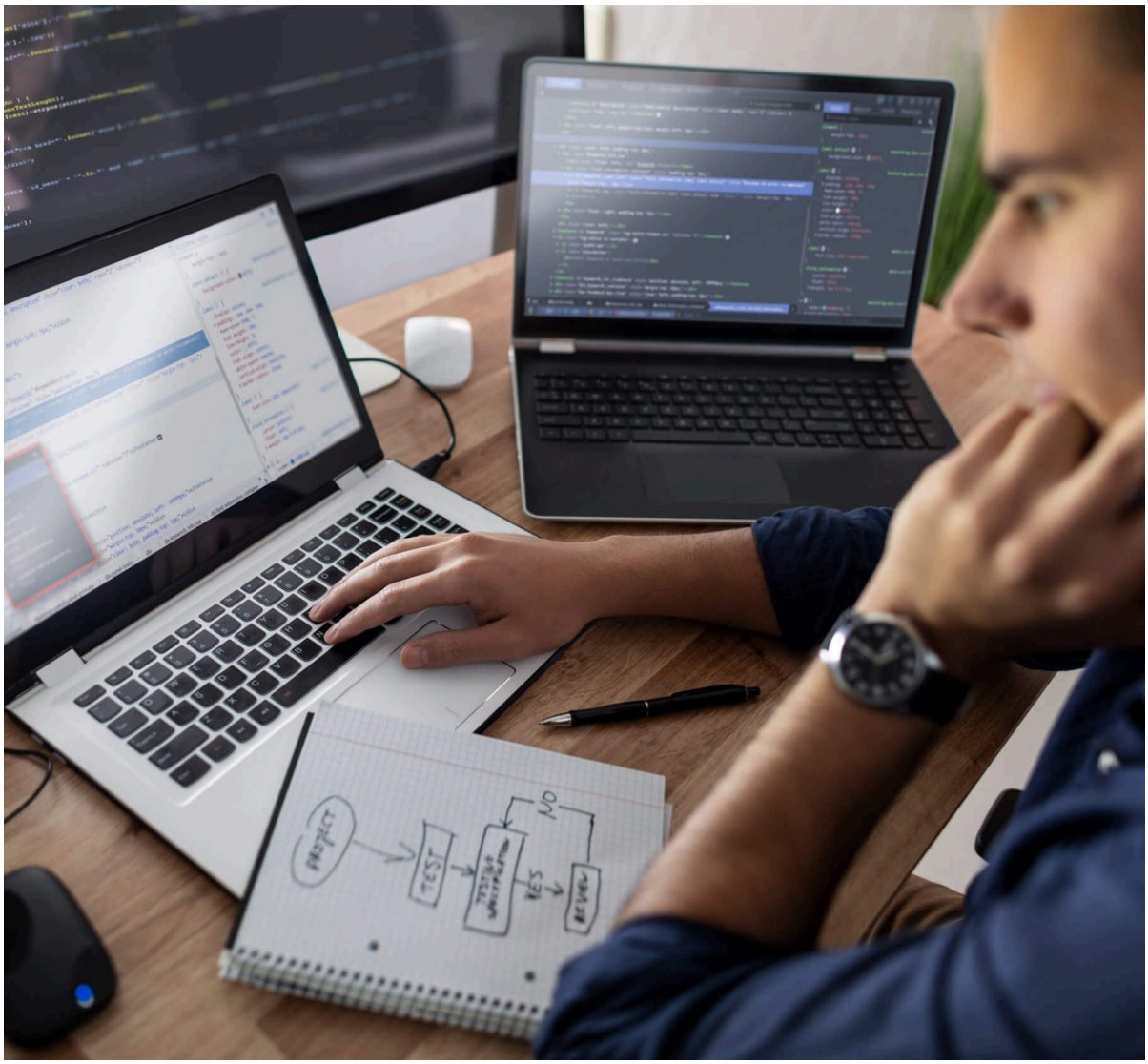
Sure, that's all great, but how does GitHub make money if everything is free and open-source?

The answer is that GitHub provides paid services as well, including private repositories. On a paid plan, teams can collaborate on GitHub while keeping their code behind closed virtual doors. GitHub also offers enterprise solutions that equip organizations with internal collaboration tools.



YOU CAN DO IT!

It's your turn



When we think of technological innovations, it's sometimes easy to attribute everything to one person — think Steve Jobs or Bill Gates. Of course, the reality is that software comes from minds working together. It advances thanks to the millions of professional and amateur developers working together to develop new things in new ways.

That's the beauty of GitHub. It lowers the barrier of entry for collaboration, so anyone can pitch in their ideas toward projects of their choice, or start a project of their own.

It's difficult to imagine the world of open-source software before GitHub. Now, it's ready for anyone to explore, including you.

Thank You