



Instalación:

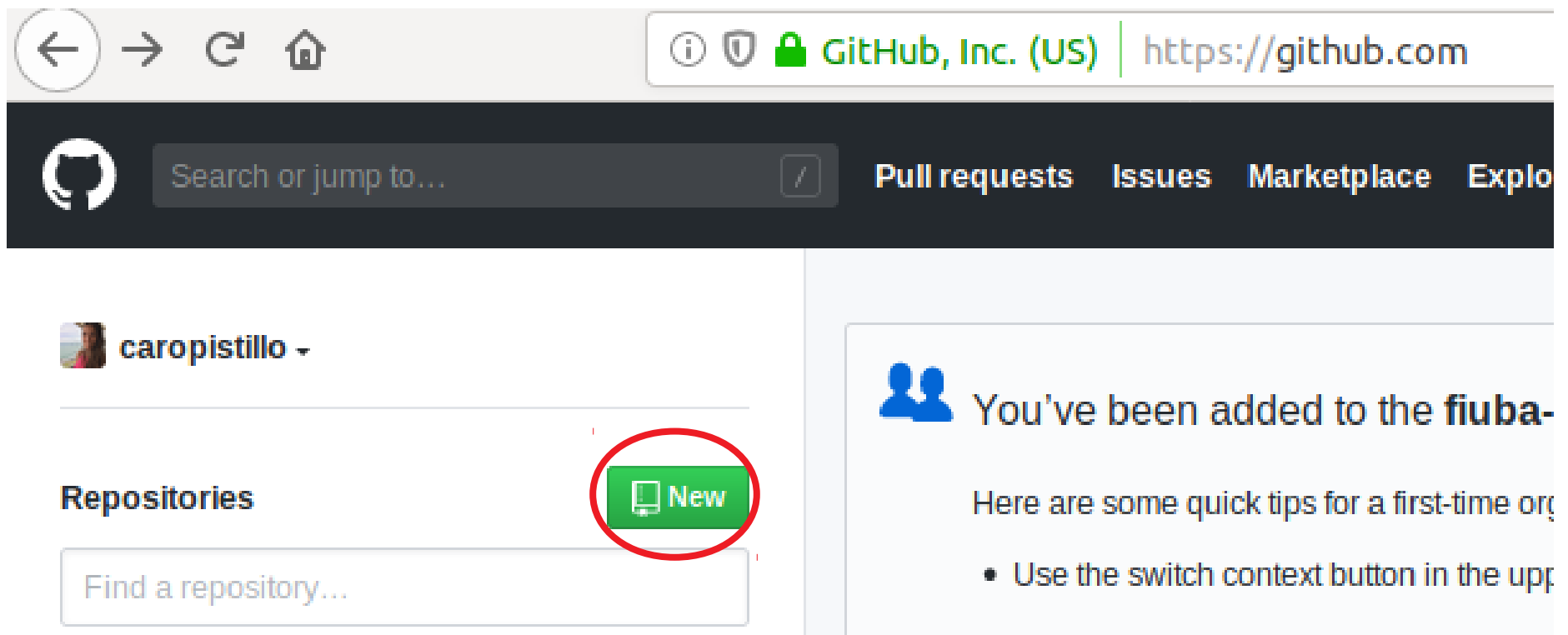
```
sudo apt-get install git-all
```

Configuración local:

```
git config --global user.name "Nombre"
```

```
git config --global user.email "email"
```

Crear un nuevo repositorio en github.com




Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)



Owner

Repository name *

 **caropistillo** ▾ / ✓

Great repository names are short and memorable. Need inspiration? How about **congenial-goggles**?

Description (optional)

- ☐  **Public**
Anyone can see this repository. You choose who can commit.
- ☒  **Private**
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

- ☒ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer.

Add .gitignore: **None** ▾

Add a license: **None** ▾



Create repository

Dentro de la carpeta donde sea desea descargar el repositorio:

```
git clone url
```

```
git remote add remoteName url.git
```

En el ejemplo:

```
git clone https://github.com/caropistillo/repoPrueba  
git remote add origin https://github.com/caropistillo/  
repoPrueba.git
```

Al ejecutar:

`git remote -v`

Debe aparecer:

origin <https://github.com/username/repoName.git> (fetch)

origin <https://github.com/username/repoName.git> (push)

Redireccionarse a la carpeta del repositorio:

`cd repoPrueba`

Inicializar git en esa dirección:

`git init`

Descargar los archivos base del campus y copiarlos a la carpeta del repositorio

Una vez hecho esto, ejecutar dentro de esa carpeta:

`git add -all`

Para agregar los archivos al repositorio

Commits

Una vez realizados los cambios (agregado los archivos nuevos o modificados):

`git commit -m "Agrego archivos base"`

Push

Una vez hecho el commit:

`git push`

El proyecto puede abrirse directamente con CodeBlocks y al hacer cambios, solo debe ejecutarse nuevamente:

`git add -all`

parados sobre la carpeta del repositorio correspondiente. Luego commitear y pushear.

Trabajar colaborativamente

Para invitar a un usuario al repositorio:

The screenshot shows the GitHub repository settings page for 'caropistillo / repoPrueba'. The repository is marked as 'Private'. The top navigation bar includes links for 'Code', 'Issues' (0), 'Pull requests' (0), 'Projects' (0), 'Security', 'Insights', and 'Settings' (which is highlighted). On the right, there are buttons for 'Unwatch' (1), 'Star' (0), and 'Fork' (0). The left sidebar contains a list of settings: 'Options', 'Collaborators' (highlighted), 'Branches', 'Webhooks', 'Notifications', 'Integrations & services', and 'Deploy keys'. The main content area is titled 'Collaborators' and includes a sub-header 'Push access to the repository'. It shows a list of collaborators with one entry: a user icon, the text 'Awaiting cpistillo's response', and buttons for 'Copy invite link' and 'Cancel invite'. Below this is a search section titled 'Search by username, full name or email address' with a note: 'You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.' There is a search input field and an 'Add collaborator' button. At the bottom, it shows '1 of 3 collaborators' with a green progress bar.

caropistillo / repoPrueba Private

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Security Insights Settings

Options

Collaborators

Branches

Webhooks

Notifications

Integrations & services

Deploy keys

Collaborators Push access to the repository

Awaiting cpistillo's response

Copy invite link Cancel invite

Search by username, full name or email address

You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.

Add collaborator

1 of 3 collaborators

Si otro usuario modifica algo y no actualizamos nuestra base, al intentar pushear saldrá el mensaje:

```
To https://github.com/caropistillo/repoPrueba ! [rejected]
master -> master (fetch first)
error: fallo el push de algunas referencias a '
https://github.com/caropistillo/repoPrueba
```

Para evitar esto, se hace un **git pull** antes de pushear, y preferentemente antes de ponerse a cambiar algo

Si no hay conflictos (se modificaron distintas porciones de código), no habrá problemas y podremos pushear directamente.

Si se modificaron mismas porciones de código al realizar **git pull** saldrá un error del estilo:

Desde <https://github.com/caropistillo/repoPrueba>

3449d7b..48937b1 master -> origin/master

Auto-fusionando main.cpp

CONFLICTO (contenido): Conflicto de fusión en main.cpp

Fusión automática falló; arregle los conflictos y luego realice un commit con el resultado

Al hacer **git status** puede verse que:

En la rama master

Tu rama y 'origin/master' han divergido,

y tienen 1 y 1 commits diferentes cada una respectivamente.

(usa "git pull" para fusionar la rama remota en la tuya)

Tienes rutas no fusionadas.

(arregla los conflictos y corre "git commit"

(usa "git merge --abort" para abortar la fusion)

Rutas no fusionadas:

(usa "git add <archivo>..." para marcar una resolución)

ambos modificados: main.cpp

sin cambios agregados al commit (usa "git add" y/o "git commit -a")

¿Cómo resolver conflictos?

- A mano
- Con un software que marca los conflictos y da a elegir entre opciones.

Ejemplo: Gitkraken

```
File Edit View Help
repoPrueba x +
repository: git/ > repoPrueba > master
submodule
branch
Undo Redo Pull Push Branch Stash Pop Boards
main.cpp (1 conflict)
[Open in external merge tool] [Save] [X]

[A] Commit 7ffc16 on master
10 int main(int argc, char** argv) {
11
12     juego = new Juego();
13     juego->iniciar("Nannnobottttt", 100, 100, 0);
14     juego->correr();
15
16     juego->limpiar();
17
18     delete juego;

[B] Commit 48937b on master
10 int main(int argc, char** argv) {
11
12     juego = new Juego();
13     juego->iniciar("Nanobot", 100, 100, 0);
14     juego->correr();
15
16     juego->limpiar();
17
18     delete juego;

Output
conflict 1 of 1
10 int main(int argc, char** argv) {
11
12     juego = new Juego();
13     juego->iniciar("Nanobot de Prueba", 100, 100, 0);
14     juego->correr();
15
16     juego->limpiar();
17
18     delete juego;
```

Branches:

Branch: Rama para aislar el trabajo de desarrollo sin afectar otras ramas en el repositorio.

`git branch nuevaBranch`

Crea una nueva branch con el nombre “nuevaBranch” desde el commit anterior.

`git commit -m "Nueva branch"`
`git push`

Commit-push sobre la nueva branch:

`git commit -m "add message"`

`git push --set-upstream origin nuevaBranch`

Este tipo de push solo es necesario para el primer push de la nueva branch

*Si se hace un **git status** se devuelve:*

En la rama nuevaBranch

Tu rama está actualizada con 'origin/nuevaBranch'.

Si se desea volver a master (u otra rama):

checkout master

git status

En la rama master

Tu rama está actualizada con 'origin/master'.

Una vez en master (o determinada rama), si queremos combinarla con la nueva rama que se baso en algún commit anterior de la rama sobre la que se esta parado, se usa **merge**:

git merge nuevaBranch

git push

