

GIT Básico

¿Qué es GIT?

[GIT](#) es un programa de control de versiones.

Configuración básica:

- `--global` se usa para configurar en todos los proyectos el mismo usuario.

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

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

```
git config --global core.editor "editor_de_texto_prefer."
```

Ver archivo configuración global:

- El archivo de global puede ser modificado.

```
git config --global -e
```

Al descargar archivos desde windows y al subirlos hay que cambiar unas variables que son CR y LF. Para no tener que hacer esto de forma manual hay que configurarlo.

- También es recomendable activar esto en MAC y Linux.

```
git config --global core.autocrlf true (o input en Linux)
```

Comandos Básicos GIT:

Iniciar un proyecto de GIT:

- Esto convertirá el directorio en un proyecto de git.

```
git init
```

Mostrar el estado del repositorio:

- Esto nos ayuda a ver qué archivos se pueden commitear o añadir para commitear.

```
git status
```

Seleccionar archivo:

- Esto hace que GIT pueda seguir las modificaciones del archivo. En esta etapa no se están guardando los archivos solo las modificaciones en ellos.

```
git add Nombre_archivo (o . para añadir todos los archivos)
```

Asegurar archivos etapa (staged):

- Esto asegura el archivo de esa forma ya no se puede modificar esa versión ni hacer un roll back.

```
git commit -m "accion_que_estamos_haciendo"
```

Eliminar archivos:

- Esto solo lo lleva a la etapa de selección después de eso debemos asegurar el archivo con un commit.

```
git rm Nombre_archivo
```

Echar atras cambio etapa de selección (staged):

```
git restore —staged Nombre_archivo
```

Descartar cambios:

```
git restore Nombre_del_archivo
```

Renombrar archivo:

```
git mv Nombre_archivo Nuvo_nombre_archivo
```

Ver todos los cambios que se van a realizar de forma más profunda:

```
git diff
```

Ver todos los commits:

- Nos permite ver todos los commit que se han realizado, con el nombre de quien y el email. También se puede acortar con solo la información del commit.

```
git log (o git log --oneline)
```

Branch:

Verificar en qué branch (ramas):

```
git branch
```

Crear una branch (ramas):

```
git checkout -b Nombre_de_la_rama
```

Cambiar entre branch (ramas):

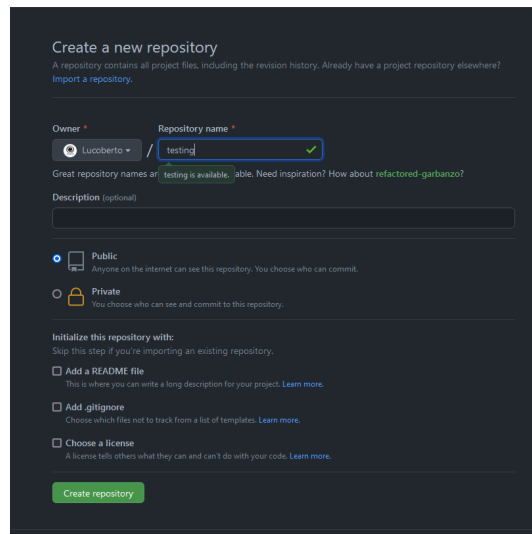
```
git checkout Nombre_de_la_rama
```

Traer modificaciones de otras ramas:

```
git merge Nombre_de_la_rama_de_la_que_quieres_el_cambio
```

Repositorios remotos (GitHub):

Crear un repo en GitHub:



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 ^{*}

Lucoberto / testing ✓

Great repository names are [testing is available](#), [able](#), [Need inspiration?](#) How about [refactored-garbanzo](#)?

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:

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

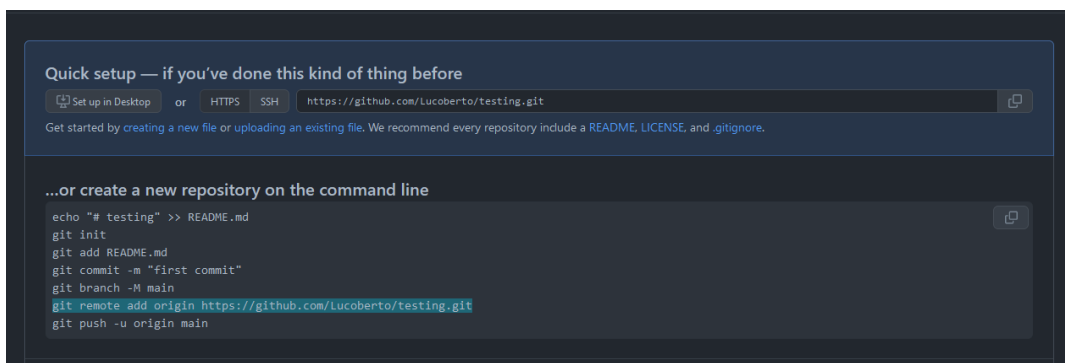
☐ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

☐ Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

Copiar el siguiente comando en la terminal para conectar con el repo de GitHub:



Quick setup — if you've done this kind of thing before

[Set up in Desktop](#) or [HTTPS](#) [SSH](#) `https://github.com/Lucoberto/testing.git`

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# testing" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Lucoberto/testing.git
git push -u origin main
```

git remote add origin url_repo

Comando de repositorios:

Crear una rama en GitHub:

- Al hacer este paso te pedirá un token de tu cuenta de GitHub.
- Para pushear una rama tienes que estar en ella.

```
git push -u origin nombre_de_la_rama
```

Enviar todos los cambios al repo de GitHub:

- Solo se enviarán los archivos de la rama en la que estés ubicado en ese momento.

```
git push
```

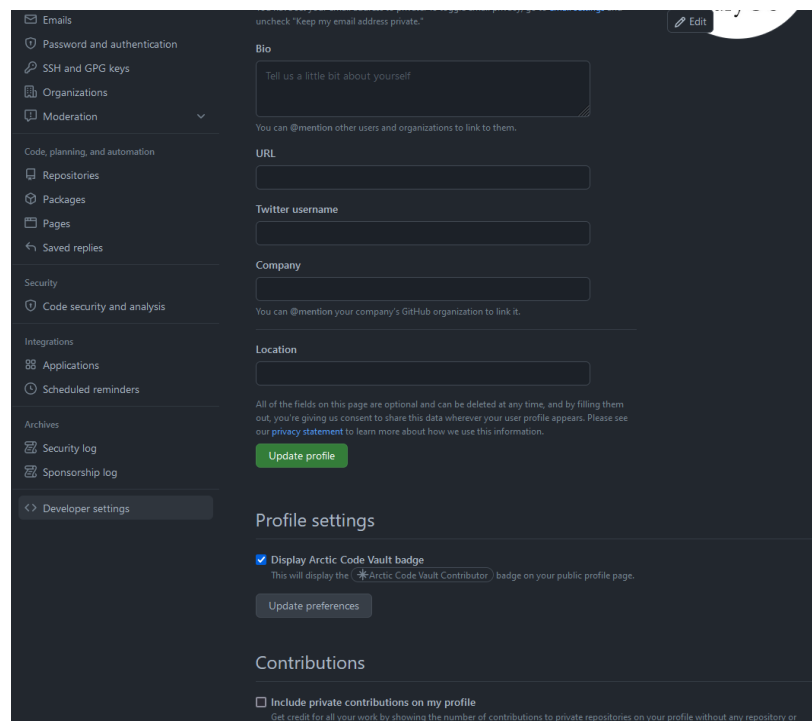
Extras:

Archivos de configuración:

Crear un archivo “.gitingnore” para poder ignorar archivos a la hora de comitar, de esta forma podremos evitar un fallo y publicar contraseñas de bases de datos o cualquier otro archivo de configuración local. Para configurar este archivo solo tienes que escribir dentro de el nombre de los archivos que no quieres comitear. (también se pueden ignorar rutas)

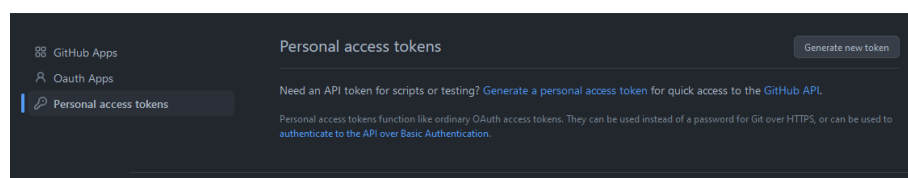
Crear un token GitHub:

Ir a developer settings en settings de tu perfil:



The screenshot shows the GitHub 'Developer settings' page. On the left is a sidebar with navigation links: Emails, Password and authentication, SSH and GPG keys, Organizations, Moderation, Code, planning, and automation (Repositories, Packages, Pages, Saved replies), Security (Code security and analysis), Integrations (Applications, Scheduled reminders), and Archives (Security log, Sponsorship log). The main content area is titled 'Developer settings' and includes sections for 'Profile settings' (with a checkbox for 'Display Arctic Code Vault badge') and 'Contributions' (with a checkbox for 'Include private contributions on my profile').

Ir a personal access token y click en generate new access token:



The screenshot shows the GitHub 'Personal access tokens' page. The left sidebar has links for 'GitHub Apps', 'OAuth Apps', and 'Personal access tokens'. The main content area is titled 'Personal access tokens' and includes a 'Generate new token' button. Below the button, there is text explaining that personal access tokens function like ordinary OAuth access tokens and can be used instead of a password for Git over HTTPS or to authenticate to the API over Basic Authentication.

Creas un token con el nombre que quieras la caducidad de unos 20 días y seleccionas donde pone repo:

GitHub Apps

OAuth Apps

Personal access tokens

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

My-git-token

What's this token for?

Expiration

30 days The token will expire on Mon, Mar 21 2022

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repostatus	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo_invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Update user public keys