

Introducción a git y Github

Laboratorio de Datos, IC - FCEN - UBA - 1er. Cuatrimestre 2025

Trabajando en grupo...

- tp1.py

Trabajando en grupo...

- `tp1.py`
- `tp1_final.py`

Trabajando en grupo...

- `tp1.py`
- `tp1_final.py`
- `tp1_final_modificadonaza.py`

Trabajando en grupo...

- `tp1.py`
- `tp1_final.py`
- `tp1_final_modificadonaza.py`
- `tp1_final_modificadonazafinal.py`

Trabajando en grupo...

- `tp1.py`
- `tp1_final.py`
- `tp1_final_modificadonaza.py`
- `tp1_final_modificadonazafinal.py`
- `tp1_final_modificadonazafinal (copia).py`

Trabajando en grupo...

- `tp1.py`
- `tp1_final.py`
- `tp1_final_modificadonaza.py`
- `tp1_final_modificadonazafinal.py`
- `tp1_final_modificadonazafinal (copia).py`
- `tp1_final_modificadonazafinal (copia) entrega.py`

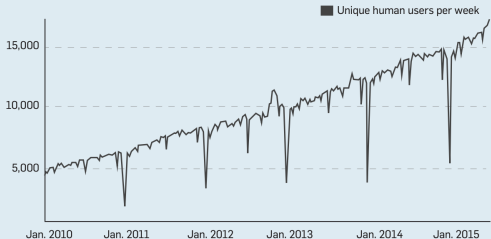
Trabajando en grupo...

- `tp1.py`
- `tp1_final.py`
- `tp1_final_modificadonaza.py`
- `tp1_final_modificadonazafinal.py`
- `tp1_final_modificadonazafinal (copia).py`
- `tp1_final_modificadonazafinal (copia) entrega.py`



...¿y en una empresa?

Figure 2. Human committers per week.



Estadísticas del repositorio de Google, enero 2015.

- Archivos Totales: 1.000 millones
- Archivos Fuente: 9 millones
- Líneas de código fuente: 2.000 millones
- Historial: 35 millones de *commits*
- Tamaño en disco: 86TB
- Commits por día laboral: 40.000

¿Qué es git?

Es un **Sistema de Control de Versiones** (CVS)

- Creado en 2005 por la comunidad que desarrollaba el kernel de Linux
- Muy eficiente en grandes proyectos y tiene un sistema de ramificación (branching)



git

Para descargar **git**: <https://git-scm.com/downloads>

Libro oficial en español: <https://git-scm.com/book/es/v2>

Vamos a crear una carpeta que se llame [COLOR] - [FRUTA] - [DIA CUMPLEAÑOS]. Yo elegí *naranja-naranja-11*.

Abrimos la consola (Ctrl+Alt+T en Ubuntu) y ejecutamos:

```
mkdir naranja-naranja-11
```

Y entramos a esa carpeta con:

```
cd naranja-naranja-11
```

Actividad 1

Configurando git

Lo primero que vamos a hacer es inicializar el **repositorio**:

```
git init
```

Actividad 1

Configurando git

Lo primero que vamos a hacer es inicializar el **repositorio**:

```
git init
```

A continuación, configuramos nuestro nombre y mail
(preferentemente el que usamos para GitHub):

```
git config --local user.name naza
```

```
git config --local user.email naza@hotmail.com
```

Actividad 1

Configurando git

Lo primero que vamos a hacer es inicializar el **repositorio**:

```
git init
```

A continuación, configuramos nuestro nombre y mail
(preferentemente el que usamos para GitHub):

```
git config --local user.name naza
```

```
git config --local user.email naza@hotmail.com
```

Obs: el `--local` es importante cuando trabajen en computadoras públicas (por ejemplo, en los labos). Si trabajan en su propia computadora, pueden usar `--global` en vez de `--local` para omitir la configuración cada vez que inician un repositorio.

Actividad 1

Creando un archivo

Vamos a crear `archivo.txt` en `act1`. Ejecutemos el siguiente comando:

```
gedit archivo.txt
```


Actividad 1

Creando un archivo

Vamos a crear `archivo.txt` en `act1`. Ejecutemos el siguiente comando:

```
gedit archivo.txt
```

En la ventana que se abrió, escribamos:

Mi color favorito es: [COLOR]

Actividad 1

Creando un archivo

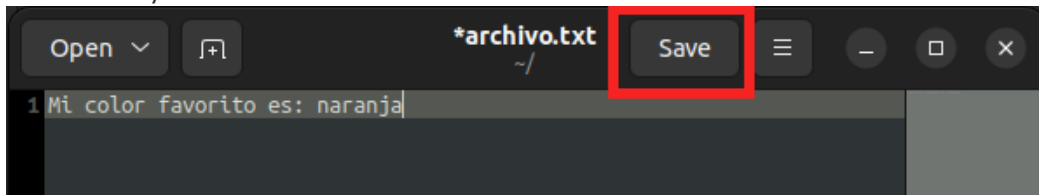
Vamos a crear `archivo.txt` en `act1`. Ejecutemos el siguiente comando:

```
gedit archivo.txt
```

En la ventana que se abrió, escribamos:

Mi color favorito es: [COLOR]

Guardamos y cerramos la ventana del editor de texto.



Actividad 1

git status

Ejecutemos:

```
git status
```

Actividad 1

git status

Ejecutemos:

```
git status
```

Obtenemos el siguiente mensaje:

```
On branch master
```

```
No commits yet
```

```
Untracked files:
```

```
(use "git add <file>..." to include in what will be committed)
```

```
archivo.txt
```

```
nothing added to commit but untracked files present (use "git add" to track)
```

Actividad 1

¿Qué es un commit?

Cada vez que hacemos un commit, estamos registrando en Git los cambios en los archivos que forman parte de ese commit. Es como sacarles una foto. El commit también tiene información importante:

- Un mensaje (generalmente indica qué se agregó/modificó/corrigió, etc.)
- Identificación (mediante un hash)
- Autor/a, fecha y hora

Actividad 1

git add

Agreguemos archivo.txt al commit:

```
git add archivo.txt
```

Actividad 1

git add

Agreguemos archivo.txt al commit:

```
git add archivo.txt
```

Volvemos a ejecutar:

```
git status
```

Actividad 1

git add

Agreguemos archivo.txt al commit:

```
git add archivo.txt
```

Volvemos a ejecutar:

```
git status
```

```
On branch master
```

```
No commits yet
```

```
Changes to be committed:
```

```
(use "git rm --cached <file>..." to unstage)
```

```
new file: archivo.txt
```


Actividad 1

git commit

Vamos a realizar un *commit*, para registrar la creación de `archivo.txt`:

```
git commit -m "Agrega color favorito"
```

Todos los commit siempre van acompañados de un mensaje. El argumento `-m` nos permite escribir el mensaje directamente en la consola (siempre entre comillas).

```
[master (root-commit) 61645a0] Agrega color favorito
1 file changed, 1 insertion(+)
create mode 100644 archivo.txt
```

Actividad 1

Si ahora corremos:

```
git status
```

Tenemos que:

```
On branch master  
nothing to commit, working tree clean
```

Actividad 1

Modifiquemos nuestro archivo

Abrimos de nuevo `archivo.txt`:

```
gedit archivo.txt
```

Y agreguemos en una nueva línea nuestra fruta favorita:

Mi fruta favorita es: [FRUTA]

Guardamos y cerramos el editor de texto.

Actividad 1

Commit de modificaciones

Como siempre que querramos saber dónde estamos parados, ejecutamos:

Actividad 1

Commit de modificaciones

Como siempre que querramos saber dónde estamos parados, ejecutamos:

```
git status
```

Que nos muestra:

Actividad 1

Commit de modificaciones

Como siempre que querramos saber dónde estamos parados, ejecutamos:

```
git status
```

Que nos muestra:

```
On branch master
```

```
Changes not staged for commit:
```

```
  (use "git add <file>..." to update what will be committed)
```

```
  (use "git restore <file>..." to discard changes in working directory)
```

```
    modified: archivo.txt
```

```
no changes added to commit (use "git add" and/or "git commit -a")
```

Actividad 1

Commit de modificaciones

Como `archivo.txt` está en rojo, todavía no está agregado al commit, así que lo agregamos con:

Actividad 1

Commit de modificaciones

Como `archivo.txt` está en **rojo**, todavía no está agregado al commit, así que lo agregamos con:

```
git add archivo.txt
```


Actividad 1

Commit de modificaciones

Como `archivo.txt` está en **rojo**, todavía no está agregado al commit, así que lo agregamos con:

```
git add archivo.txt
```

Y chequeamos que esté agregado al commit:

Actividad 1

Commit de modificaciones

Como `archivo.txt` está en **rojo**, todavía no está agregado al commit, así que lo agregamos con:

```
git add archivo.txt
```

Y chequeamos que esté agregado al commit:

```
git status
```

```
On branch master
```

```
Changes to be committed:
```

```
  (use "git restore --staged <file>..." to unstage)
```

```
    modified:   archivo.txt
```

Actividad 1

Finalmente, *commiteamos*:

Actividad 1

Finalmente, *commiteamos*:

```
git commit -m "Agrega fruta favorita"
```

```
[master aef49de] Agrega fruta favorita  
1 file changed, 1 insertion(+)
```

Actividad 1

Resumen

Configurando el repo (una sola vez)

git init	inicializar el repo
git config	configurar nombre de autor/a y mail (usar --local en los labos!)

Agregando y quitando archivos

git add <i>nombre_archivo</i>	añadir la creación o modificación del archivo al commit
git commit -m "mensaje"	comitear, registrar los cambios en git
git status	mostrar el estado del commit actual

Recordar el orden: modifiko/agrego archivo → lo añado al commit → comiteo

Actividad 1

Ejercicio:

Agregar una línea a `archivo.txt`:

Mi banda favorita es: [NOMBRE DE LA BANDA]

y registrar el cambio en Git.

El mensaje del commit debe ser: Agrega banda favorita

Actividad 1

git log

Con `git log` podemos ver el historial de commits:

```
git log --oneline
```

Actividad 1

git log

Con `git log` podemos ver el historial de commits:

```
git log --oneline
```

```
435fa82 (HEAD -> master) Agrega banda favorita  
2015257 Agrega fruta favorita  
58a8d12 Agrega color favorito
```


Actividad 1

Volviendo al pasado

Si queremos volver a un commit anterior, usamos `git reset --hard` con el hash del commit al que queremos volver. Volvamos al segundo commit:

```
git reset --hard 2015257
```

Actividad 1

Volviendo al pasado

Si queremos volver a un commit anterior, usamos `git reset --hard` con el hash del commit al que queremos volver. Volvamos al segundo commit:

```
git reset --hard 2015257
```

¿Cuál es el contenido de `archivo.txt`?

Actividad 1

Volviendo al pasado

Si queremos volver a un commit anterior, usamos `git reset --hard` con el hash del commit al que queremos volver. Volvamos al segundo commit:

```
git reset --hard 2015257
```

¿Cuál es el contenido de `archivo.txt`?

Observación: usar `git reset --hard` es conveniente cuando deshacemos commits locales, o sea, **antes de hacer push**, para evitar problemas con nuestros colaboradores. En otras ocasiones, se utiliza `git revert`. [\[Más info\]](#)



2015257 Agrega fruta favorita

58a8d12 Agrega color favorito

Github

Repositorios remotos

Para colaborar en proyectos con otras personas con Git, utilizamos **repositorios remotos**. Son versiones de nuestros proyectos que están **hospedadas en internet**.

Repositorios remotos

Para colaborar en proyectos con otras personas con Git, utilizamos **repositorios remotos**. Son versiones de nuestros proyectos que están **hospedadas en internet**.



Usaremos GitHub. Es el más popular, pero existen alternativas como [Bitbucket](#) y [Gitlab](#).

GitHub - Personal Access Token (PAT)

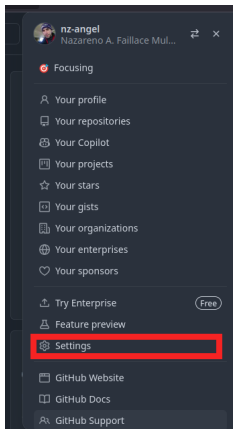
Los «Tokens de Acceso Personal» (PAT) son una alternativa al uso de contraseñas para la autenticación en GitHub cuando se usa la API o la línea de comandos. El PAT está diseñado para acceder a los recursos de GitHub en tu nombre.

GitHub - Personal Access Token (PAT)

1. Nos logueamos en [GitHub](#).

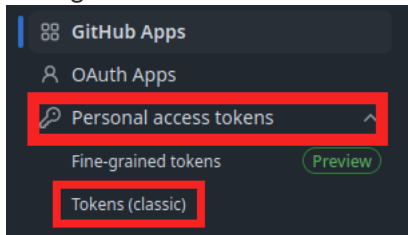
GitHub - Personal Access Token (PAT)

1. Nos logueamos en **GitHub**.
2. Hacemos click en nuestro *avatar* (arriba a la derecha) y elegimos **Settings** del menú desplegable.



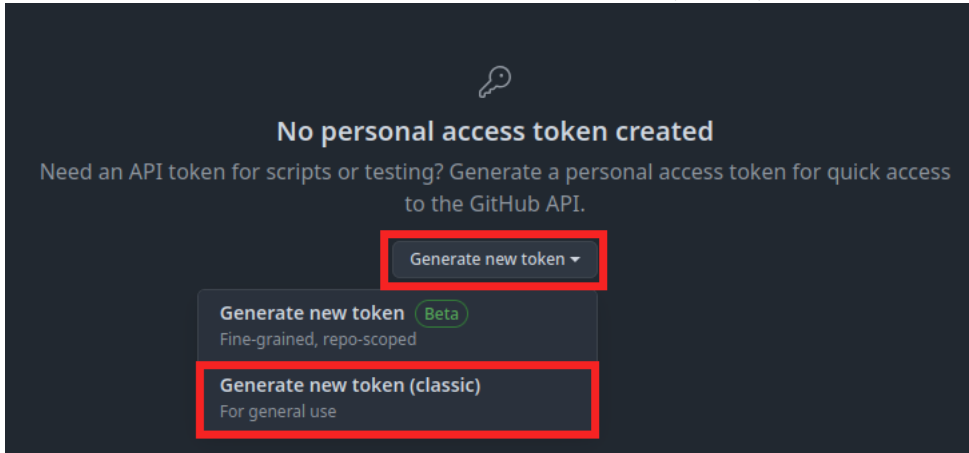
GitHub - Personal Access Token (PAT)

3. Elegimos **Developer settings** (último ítem del menú de la izquierda)
4. En el menú de la izquierda elegimos **Personal Access Tokens** → **Tokens (classic)**



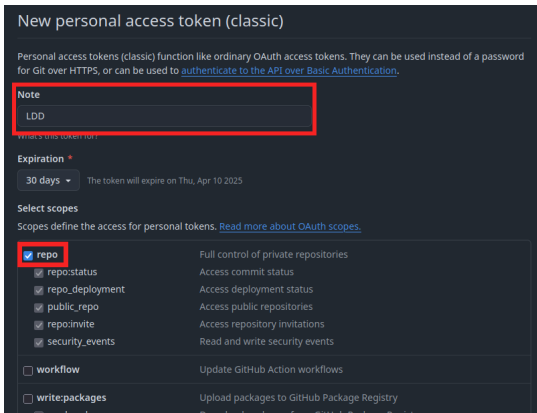
GitHub - Personal Access Token (PAT)

5. Le damos a **Generate new token** → **Generate new token (classic)**



GitHub - Personal Access Token (PAT)

6. Escribimos una descripción (puede ser cualquier cosa) y marcamos la casilla de **repo**. Después, **Generate token**.



New personal access token (classic)

Personal access tokens (classic) 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

LDD

Expiration *

30 days ▾ The token will expire on Thu, Apr 10 2025

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"/> repo:status	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

GitHub - Personal Access Token (PAT)

7. **Copiar** el token (es la única vez que GitHub se los muestra).

Si tienen un gestor de contraseñas pueden guardarlo ahí. Una opción mucho menos segura pero práctica es guardarlo en el campus.

Alternativa a PAT: Conectarse mediante SSH (para compus personales)

Actividad 2 - GitHub

Comenzamos con la Actividad 2.

Vamos a aprender cómo crear un repo en GitHub y como *pushear* nuestro repo local.

Actividad 2 - GitHub

1. Habiéndonos logueado, vamos a **GitHub**.

Actividad 2 - GitHub

1. Habiéndonos logueado, vamos a **GitHub**.
2. Hacemos click en **New** (botón verde arriba a la izquierda)

Actividad 2 - GitHub

1. Habiéndonos logueado, vamos a **GitHub**.
2. Hacemos click en **New** (botón verde arriba a la izquierda)
3. Configuramos el repo:
 - El nombre debe ser exactamente igual al de la carpeta donde veníamos trabajando.
 - Agregamos una descripción breve como, por ejemplo, *Actividad de LDD*.
 - Lo dejamos como público.
 - Destildamos *Add a README file* .
 - No usaremos un .gitignore

Actividad 2 - GitHub

1. Habiéndonos logueado, vamos a **GitHub**.
2. Hacemos click en **New** (botón verde arriba a la izquierda)
3. Configuramos el repo:
 - El nombre debe ser exactamente igual al de la carpeta donde veníamos trabajando.
 - Agregamos una descripción breve como, por ejemplo, *Actividad de LDD*.
 - Lo dejamos como público.
 - Destildamos *Add a README file* .
 - No usaremos un .gitignore
4. Le damos click a **Create Repository**.

Actividad 2 - GitHub

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

HTTPS

SSH

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



...or push an existing repository from the command line

```
git remote add origin https://github.com/nz-angel/naranja-naranja.git
git branch -M main
git push -u origin main
```



Actividad 2 - GitHub

Copiamos esas líneas y las ejecutamos en la consola.

Nos va a pedir:

- Username for 'https://github.com': ponemos nuestro nombre de usuario de GitHub
- Password for 'https://{usuario}@github.com': pegamos nuestro PAT (después de copiarlo, vamos a la consola y click derecho → pegar)

Actividad 2 - GitHub

Debería imprimir en consola:

```
Enumerating objects: 6, done.  
Counting objects: 100% (6/6), done.  
Delta compression using up to 4 threads  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (6/6), 502 bytes | 502.00 KiB/s, done.  
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0  
To https://github.com/nz-angel/naranja-naranja.git  
* [new branch]      main -> main  
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

Actividad 2 - GitHub

Debería imprimir en consola:

```
Enumerating objects: 6, done.  
Counting objects: 100% (6/6), done.  
Delta compression using up to 4 threads  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (6/6), 502 bytes | 502.00 KiB/s, done.  
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0  
To https://github.com/nz-angel/naranja-naranja.git  
* [new branch]      main -> main  
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

Volvemos a la página de nuestro repo en Github y actualizamos (F5).

Actividad 2 - Github

¿Qué hace cada línea?

```
git remote add origin https://github.com/{usuario}/{nombre de repo}.git
```

Vincula nuestra carpeta con nuestro repo en GitHub.

```
git branch -M main
```

Cambia el nombre de la rama principal a main.

```
git push -u origin main
```

Envía (*pushea*) nuestros commits al repo de GitHub.

Resumen:

Después de modificar/agregar archivos, añado los cambios al commit:

```
git add nombre_archivo
```

Cuando quiera *guardar localmente* los cambios en Git, comiteo:

```
git commit -m "mensaje de commit"
```

Cuando quiera enviar los commits a mi repo de GitHub, pusheo:

```
git push origin nombre_de_branch
```

Cuando quiero actualizar mi repo local con los cambios del repo de GitHub:

```
git pull --all
```



2015257 Agrega fruta favorita
58a8d12 Agrega color favorito





2015257 Agrega fruta favorita

58a8d12 Agrega color favorito

PUSH →



2015257 Agrega fruta favorita
58a8d12 Agrega color favorito

PUSH →



2015257 Agrega fruta favorita
58a8d12 Agrega color favorito



2015257 Agrega fruta favorita
58a8d12 Agrega color favorito



2015257 Agrega fruta favorita
58a8d12 Agrega color favorito
4e570a3 Agrega banda favorita



2015257 Agrega fruta favorita
58a8d12 Agrega color favorito



2015257 Agrega fruta favorita
58a8d12 Agrega color favorito
4e570a3 Agrega banda favorita
671a9cb Modifica fruta favorita



2015257 Agrega fruta favorita
58a8d12 Agrega color favorito



2015257 Agrega fruta favorita
58a8d12 Agrega color favorito
4e570a3 Agrega banda favorita
671a9cb Modifica fruta favorita
4f5c911 Agrega nuevo archivo



2015257 Agrega fruta favorita
58a8d12 Agrega color favorito



2015257 Agrega fruta favorita
58a8d12 Agrega color favorito
4e570a3 Agrega banda favorita
671a9cb Modifica fruta favorita
4f5c911 Agrega nuevo archivo

PULL ←



2015257 Agrega fruta favorita
58a8d12 Agrega color favorito
4e570a3 Agrega banda favorita
671a9cb Modifica fruta favorita
4f5c911 Agrega nuevo archivo

PULL ←



2015257 Agrega fruta favorita
58a8d12 Agrega color favorito
4e570a3 Agrega banda favorita
671a9cb Modifica fruta favorita
4f5c911 Agrega nuevo archivo

Actividad 2 - Github

Ejercicio:

Agregar una línea a `archivo.txt`:

Mi comida favorita es: [COMIDA]

Subir este cambio a GitHub. Para pushear, el nombre de la rama principal es `main`

GitHub

Lo que vimos hasta ahora alcanza para trabajar **individualmente** con GitHub. Esto nos puede servir, por ejemplo, para usarlo como almacenamiento en la nube y/o si trabajamos en el mismo proyecto en distintas computadoras.

En general es bueno hacer un pull al comenzar la jornada laboral.

Trabajando en equipo

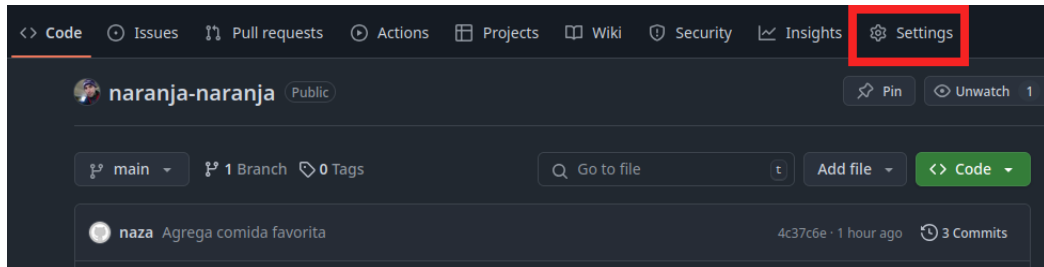
Actividad 3

Vamos a ver como intervenir el trabajo de los demás. Nos ponemos en equipos de dos.

Actividad 3

Vamos a ver como intervenir el trabajo de los demás. Nos ponemos en equipos de dos.

Primero, vamos a registrar al compa como colaborador/a en nuestro repo. Vamos a **Settings**



1

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Codespaces

Pages

Security

Code security

Deploy keys

Who has access

Public repository

This repository is public and visible to anyone

Manage

PUBLIC REPOSITORY

This repository is public and visible to anyone.

[Manage](#)

DIRECT ACCESS

0 collaborators have access to this repository. Only you can contribute to this repository.

Manage access

2

You haven't invited any collaborators yet

Add people

Buscamos el nombre de nuestro compa y le damos a **Add**. Nuestro compa tiene que aceptar la invitación, que le llega al mail. Una vez haga esto, tendrá permiso para subir sus cambios a nuestro repo.

Buscamos el nombre de nuestro compa y le damos a **Add**. Nuestro compa tiene que aceptar la invitación, que le llega al mail. Una vez haga esto, tendrá permiso para subir sus cambios a nuestro repo.

Lo que sigue es **clonar** el repo del compa: lo queremos copiar de GitHub a nuestra compu.

Actividad 3

git clone

En el navegador vamos al repo de Github de nuestro compa:

https://github.com/usuario_compa/nombre_repo

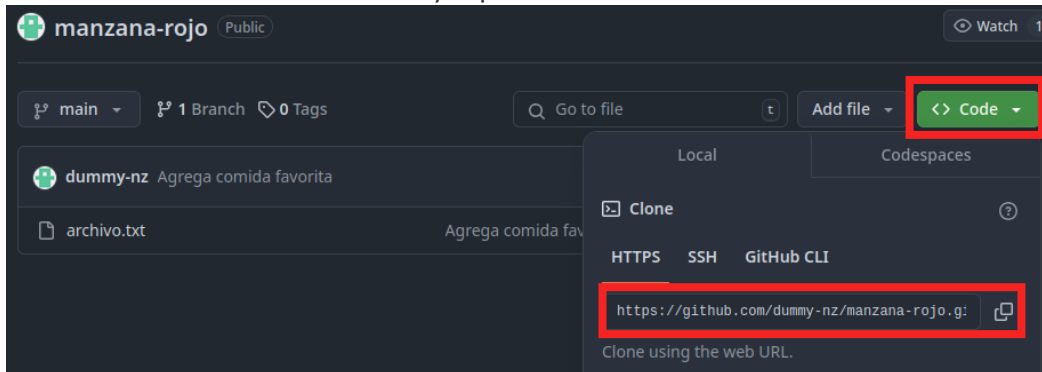
Actividad 3

git clone

En el navegador vamos al repo de Github de nuestro compa:

`https://github.com/usuario_compa/nombre_repo`

Hacemos click en el botoncito **Code** y copiamos la dirección



Abrimos una nueva terminal (**Ctrl + Alt + T**) y ejecutamos `git clone` seguido de la dirección que copiamos recién:

```
git clone https://github.com/dummy-nz/manzana-rojo.git
```

Abrimos una nueva terminal (**Ctrl + Alt + T**) y ejecutamos `git clone` seguido de la dirección que copiamos recién:

```
git clone https://github.com/dummy-nz/manzana-rojo.git
```

Y a continuación, entramos a la carpeta que acaba de ser creada, que tiene el nombre del repo del compa:

```
cd manzana-rojo
```

Configuramos nuestro nombre y mail (preferentemente el que usamos para GitHub):

```
git config --local user.name naza
```

```
git config --local user.email naza@hotmail.com
```

Configuramos nuestro nombre y mail (preferentemente el que usamos para GitHub):

```
git config --local user.name naza  
git config --local user.email naza@hotmail.com
```

Abrimos el archivo de nuestro compa:

```
gedit archivo.txt
```

Configuramos nuestro nombre y mail (preferentemente el que usamos para GitHub):

```
git config --local user.name naza  
git config --local user.email naza@hotmail.com
```

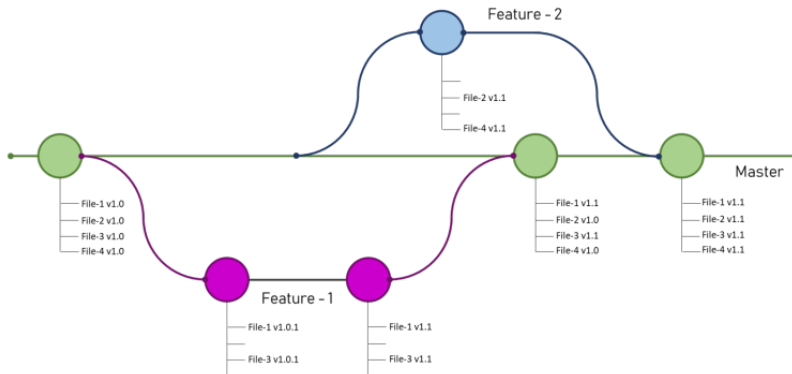
Abrimos el archivo de nuestro compa:

```
gedit archivo.txt
```

Podemos corroborar que tiene la info sobre su color, fruta y comida favorita. **Cerramos el archivo sin modificar nada.**

Actividad 3

Branch



Fuente de la imagen: <https://digitalvarys.com/git-branch-and-its-operations/>

Actividad 3

Para crear una rama que se llame `mi_modificacion` y movernos a ella:

```
git checkout -b mi_modificacion
```

Actividad 3

Para crear una rama que se llame `mi_modificacion` y movernos a ella:

```
git checkout -b mi_modificacion
```

Para ver las ramas actuales:

```
git branch
```

Actividad 3

Para crear una rama que se llame `mi_modificacion` y movernos a ella:

```
git checkout -b mi_modificacion
```

Para ver las ramas actuales:

```
git branch
```

Para movernos a `otra_rama`:

```
git checkout otra_rama
```

Chequeamos que estemos en la rama correcta (mi_modificacion). Ejecutando git branch deberíamos tener el siguiente output:

```
main  
*mi_modificacion
```

Chequeamos que estemos en la rama correcta (mi_modificacion). Ejecutando git branch deberíamos tener el siguiente output:

```
main  
*mi_modificacion
```

Y ahora modifiquemos archivo.txt. Agreguemos una línea, por ejemplo:

Ah, mirá vos!

Agregamos el cambio al commit y comitteamos con un mensaje.

Volvamos a la rama main (`git checkout main`) y abramos `archivo.txt`. ¿Qué sucede?

Volvamos a la rama `main` (`git checkout main`) y abramos `archivo.txt`. ¿Qué sucede?

Volvamos a la rama `mi_modificacion`. Esta rama y sus cambios en `archivo.txt`, ¿ya están subidos a GitHub?

Volvamos a la rama `main` (`git checkout main`) y abramos `archivo.txt`. ¿Qué sucede?

Volvamos a la rama `mi_modificacion`. Esta rama y sus cambios en `archivo.txt`, ¿ya están subidos a GitHub? **No, ¡nos falta pushear!**

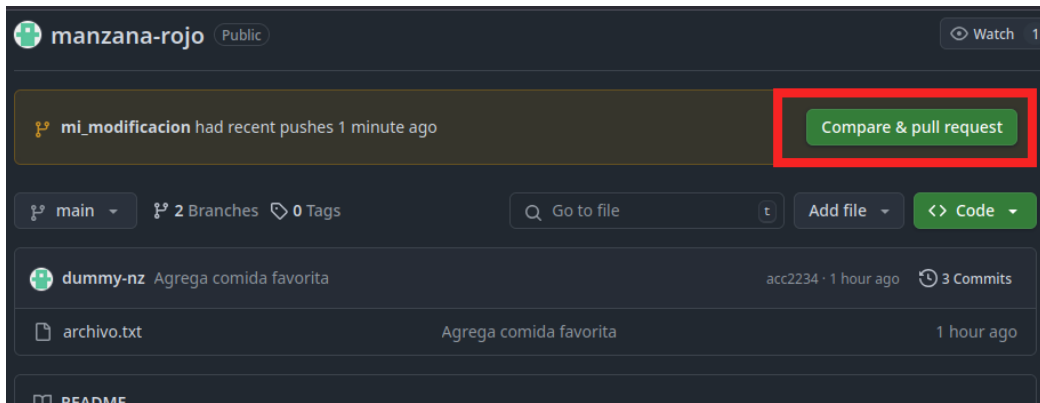
Volvamos a la rama main (`git checkout main`) y abramos `archivo.txt`. ¿Qué sucede?

Volvamos a la rama `mi_modificacion`. Esta rama y sus cambios en `archivo.txt`, ¿ya están subidos a GitHub? **No, ¡nos falta push!**

Para push **esta rama**, hacemos:

```
git push origin mi_modificacion
```

Si ahora vamos al repo de GitHub de nuestro compa, vamos a ver que el número de *Branches* aumentó y apareció un botón **Compare & pull request**:



Generalmente creamos un Pull Request cuando estamos listos para unir (**merge**) nuestra rama a main .

Generalmente creamos un Pull Request cuando estamos listos para unir (**merge**) nuestra rama a `main`. Igual podemos crear el Pull Request y seguir pusheando commits a nuestra rama hasta que el Pull Request sea *mergeado*.

Generalmente creamos un Pull Request cuando estamos listos para unir (**merge**) nuestra rama a `main`. Igual podemos crear el Pull Request y seguir pusheando commits a nuestra rama hasta que el Pull Request sea *mergeado*.

Podemos designar a nuestro compa como reviewer del Pull Request, para que acepte el cambio si está de acuerdo.

base: main ← compare: mi_modificacion ✓ Able to merge. These branches can be automatically merged.



Add a title

Título del Pull Request

Agrega mi comentario

Add a description

Write Preview

H B I

Agrega mi opinión sobre lo que estaba escrito

Descripción general

Markdown is supported Paste, drop, or click to add files

Create pull request

Reviewers

Suggestions

dummy-nz

Request

Agregar compa
No one—[assign yourself](#)
como reviewer

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Use [Closing keywords](#) in the description to automatically close issues

Helpful resources

[GitHub Community Guidelines](#)

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

Crear Pull Request

1 commit 1 file changed 1 contributor

Commits on Mar 11, 2025

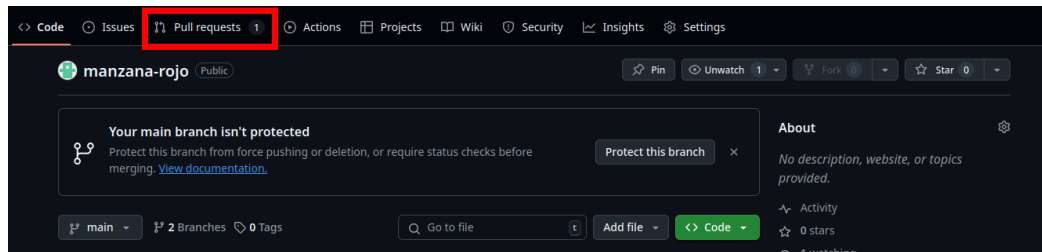
Agrega mi comentario

nz-angel committed 32 minutes ago

8c4f53b

Para revisar un Pull Request

Si queremos revisar el Pull Request de nuestro compa a nuestro repo, desde nuestro repo vamos a Pull Request



Elegimos el Pull Request de la lista, y le damos a **Add you review**.

The screenshot displays a GitHub Pull Request interface. At the top, a dark blue banner contains the text "nz-angel requested your review on this pull request." To the right of this banner is a green button labeled "Add your review", which is highlighted with a red rectangular border. Below the banner, the title "Agrega mi comentario #1" is visible, followed by "Edit" and "<> Code" buttons. A green "Open" button with a fork icon is next to the text "nz-angel wants to merge 1 commit into main from mi_modificacion". Below this, a navigation bar shows "Conversation 0", "Commits 1", "Checks 0", and "Files changed 1". A status bar on the right indicates "+1 -0" with a progress indicator. At the bottom, a comment by "nz-angel" is shown, stating "commented 13 minutes ago". To the right of the comment are buttons for "Collaborator" and "Reviewers".

Ahora podemos ver los cambios en todos los archivos con respecto a main, comentar, aceptar o rechazar el PR.

The screenshot displays a GitHub Pull Request (PR) interface. The main area shows a file diff for 'archivo.txt' with changes relative to the 'main' branch. The diff shows four lines of code, with the fourth line being a new addition: '+ Ah, mirá vos!'. A red box labeled '2' highlights the diff area.

Overlaid on the right is a 'Finish your review' modal. The modal has a 'Write' tab selected, showing a text input field with the text 'Aceptado!'. A red box labeled '3' highlights the text input field. Below the input field are three radio button options: 'Comment', 'Approve', and 'Request changes'. A red box labeled '4' highlights the 'Submit review' button at the bottom right of the modal. On the right side of the modal, there is a 'Review changes' button, also highlighted with a red box labeled '1'.

The background interface includes a header with the title 'Agrega mi comentario #1', a status bar showing 'nz-angel wants to merge 1 commit into main from mi_modificaci...', and a sidebar with navigation links for 'Conversation', 'Commits', 'Checks', and 'Files'.

Si aprobamos los cambios, podemos clicar en **Merge pull request** → **Commit merge**.
Ahora, los cambios de mi compa fueron aplicados a la rama main.

Si aprobamos los cambios, podemos clicar en **Merge pull request** → **Commit merge**.
Ahora, los cambios de mi compa fueron aplicados a la rama `main`.

Si quiero actualizar mi repo local para que refleje los cambios que hizo mi compa en mi repo, hago:

```
git pull
```

en la carpeta de mi repo.

Resumiendo:

Una vez que soy colaborador de un proyecto, puedo crear mi rama con:

```
git checkout -b mi_rama
```

Resumiendo:

Una vez que soy colaborador de un proyecto, puedo crear mi rama con:

```
git checkout -b mi_rama
```

En mi rama, efectúo las modificaciones que crea necesarias. Voy usando `git add` y `git commit` para registrarlas en mi repo local. Cuando quiero subir los cambios a mi rama en el repo de GitHub:

```
git push origin mi_rama
```

Resumiendo:

Una vez que soy colaborador de un proyecto, puedo crear mi rama con:

```
git checkout -b mi_rama
```

En mi rama, efectúo las modificaciones que crea necesarias. Voy usando `git add` y `git commit` para registrarlas en mi repo local. Cuando quiero subir los cambios a mi rama en el repo de GitHub:

```
git push origin mi_rama
```

Cuando estoy listo para unir mi rama a `main`, creo un Pull Request a través de GitHub y, opcionalmente, pongo a mi compañero/a como reviewer.