

# Trabajo practico Nº2: Git y Github

Alumno: Santiago Pace

## Actividades

**1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada**

**(Desarrollar las respuestas):**

- ¿Qué es GitHub?

- GitHub es una pagina donde podemos compartir nuestros repositorios de forma publica o privada y nos sirve para publicar nuestros trabajos que hayamos realizado para venderse uno mismo a la hora de buscar un empleo.

- ¿Cómo crear un repositorio en GitHub?

- Para crear un repositorio tenes que ir al signo + que esta arriba a la derecha y poner "Nuevo repositorio". Le ponemos nombre y una breve descripción y ponemos "Crear repositorio". Luego ya es ir siguiendo los pasos que te da github que tenes que ir copiando comandos en consola para terminar de crear ese repositorio.

- ¿Cómo crear una rama en Git?

- Para crear una rama en Git hay que ir a la terminal de Visual Studio Code y poner el comando "git branch" + el nombre de la rama.

- ¿Cómo cambiar a una rama en Git?

- Para cambiar una rama en Git tenes que poner el siguiente comando en la terminal: "git checkout" + el nombre donde queres ir.

- ¿Cómo fusionar ramas en Git?

- Para fusionar ramas en Git tenes que poner el siguiente comando en la terminal: "git merge" + el nombre de la rama que queres fusionar. Y es muy importante estar en la rama ORIGINAL para hacer ese comando.

- ¿Cómo crear un commit en Git?

- Para crear un commit en Git tenes que poner el siguiente comando en la terminal: "git commit -m" y luego poner un breve mensaje de lo que estas guardando y entre comillas.

- ¿Cómo enviar un commit a GitHub?

- Para enviar un commit a GitHub se puede usar el comando git push, este comando va a subir los cambios realizados en tu rama local a un repositorio remoto. Por ejemplo: git push origin nombre de la rama.

- ¿Qué es un repositorio remoto?

- Un repositorio remoto es donde se guarda nuestro proyecto y están alojados en un servidor de internet o en cualquier otra red.

- ¿Cómo agregar un repositorio remoto a Git?

- Para agregar un repositorio remoto a Git tienes que poner el siguiente comando en la terminal: "git add .".

- ¿Cómo empujar cambios a un repositorio remoto?

- Para empujar cambios a un repositorio remoto tienes que poner el siguiente comando en la terminal: "git push origin".

- ¿Cómo tirar de cambios de un repositorio remoto?

- Para tirar de cambios de un repositorio remoto tienes que poner el siguiente comando en la terminal: "git pull origin".

- ¿Qué es un fork de repositorio?

- Un FORK en un repositorio es una copia de un repositorio en tu cuenta de GitHub. Te permite modificar el código sin afectar el repositorio original. Es útil para contribuir a proyectos abiertos.

- ¿Cómo crear un fork de un repositorio?

- Para crear una FORK de un repositorio tienes que ir al repositorio que quieres forkar. Luego hacer click en el botón "fork" y va a crear una copia en tu cuenta. Por último puedes clonarlo con el link que aparece.

- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?

- Para enviar una solicitud de extracción a un repositorio debes asegurarte de que tu fork o rama tenga los cambios que quieres hacer. Subes los cambios con el comando "git push", vas a la página de tu fork en GitHub, haces click en el botón "Compare y pull request", haces un título con una descripción y por último tocas el botón "Create pull request".

- ¿Cómo aceptar una solicitud de extracción?

- Para aceptar una solicitud de extracción tienes que ir a la pestaña de Pull Request en GitHub, seleccionar el pull request que deseas aceptar, revisas los cambios y si está todo bien haces click en "Merge pull request". Confirmas la fusión con "Confirm merge".

- ¿Qué es un etiqueta en Git?

- Una etiqueta en Git es un marcador usado para señalar versiones importantes en el historial, como lanzamientos de software. Es útil para identificar versiones estables sin afectar el flujo de trabajo.

- ¿Cómo crear una etiqueta en Git?

- Para crear una etiqueta en Git tienes que poner el siguiente comando en la terminal: `"git tag"`.

- ¿Cómo enviar una etiqueta a GitHub?

- Para enviar una etiqueta a GitHub tienes que poner el siguiente comando en la terminal: `"git push origin --tags"`.

- ¿Qué es un historial de Git?

- Un historial de Git es un registro de todos los cambios realizados en el repositorio. Incluye detalles sobre cada confirmación, como el autor, la fecha, el mensaje de la confirmación, y qué archivos fueron modificados. Este historial permite rastrear la evolución del proyecto y revertir cambios si es necesario.

- ¿Cómo ver el historial de Git?

- Para ver el historial de Git tienes que poner el siguiente comando en la terminal: `"git log"`. O simplemente abrir la terminal en Visual Studio Code.

- ¿Cómo buscar en el historial de Git?

- Para buscar en el historial de Git tienes que poner el siguiente comando en la terminal: `"git log --grep=" + la palabra que quieras buscar`.

- ¿Cómo borrar el historial de Git?

- Para borrar el historial tienes que poner `"cls"` en la terminal y listo. O para borrar completo el historial pones `"git rebase -i --root"`.

- ¿Qué es un repositorio privado en GitHub?

- Un repositorio privado en GitHub es un repositorio cuyo acceso está restringido a un grupo específico de personas. Solo los usuarios con permiso pueden ver, modificar o clonar el contenido de ese repositorio. Esto es útil para proyectos privados o confidenciales, donde no quieres que otros usuarios fuera de tu equipo vean o colaboren en el código.

- ¿Cómo crear un repositorio privado en GitHub?

- Para crear un repositorio privado tienes que hacer lo mismo que la respuesta de la pregunta 2 y poner el candado, o sea ponerlo en privado.

- ¿Cómo invitar a alguien a un repositorio privado en GitHub?

- Para invitar a alguien a un repositorio privado en GitHub tienes que ir a ese repositorio, configuración, gestionar acceso, invitar a un colaborador, ingresar el nombre y agregarlo.

- ¿Qué es un repositorio público en GitHub?

- Un repositorio público en GitHub es un repositorio cuyo contenido está visible para cualquier persona en Internet. Cualquiera puede ver, clonar, bifurcar o contribuir al código si el repositorio lo permite. Los repositorios públicos son ideales para proyectos de código abierto.

- ¿Cómo crear un repositorio público en GitHub?

- Para crear un repositorio publico tenes que hacer lo mismo que la respuesta de la pregunta 2 y poner el candado abierto, o sea publico.

- ¿Cómo compartir un repositorio público en GitHub?

- Para compartir un repositorio publico en GitHub tenes que ir al repositorio que queres compartir, copiar la URL del repositorio, pasársela a la persona que queres que lo vea.

## 2) Realizar la siguiente actividad:

Link al repositorio: [https://github.com/santipace/tp2\\_act2.git](https://github.com/santipace/tp2_act2.git)

### Create a new repository

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



*Required fields are marked with an asterisk (\*).*

Owner *	Repository name *
 santipace ▾	/ tp2_act2
✔ tp2_act2 is available.	

Great repository names are short and memorable. Need inspiration? How about [shiny-tribble](#) ?

Description (optional)

Este es mi repositorio de la actividad 2 |

- ☒  **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.

```
Santi@DESKTOP-1Q0K68D MINGW64 ~/Desktop/repo_tp2_act2
$ git clone https://github.com/santipace/tp2_act2.git
Cloning into 'tp2_act2'...
warning: You appear to have cloned an empty repository.

Santi@DESKTOP-1Q0K68D MINGW64 ~/Desktop/repo_tp2_act2
$ cd tp2_act2

Santi@DESKTOP-1Q0K68D MINGW64 ~/Desktop/repo_tp2_act2/tp2_act2 (main)
$ echo "Este es mi primer archivo en GitHub" > mi-archivo.txt

Santi@DESKTOP-1Q0K68D MINGW64 ~/Desktop/repo_tp2_act2/tp2_act2 (main)
$ git add .
warning: in the working copy of 'mi-archivo.txt', LF will be replaced by CRLF the next time Git touches it

Santi@DESKTOP-1Q0K68D MINGW64 ~/Desktop/repo_tp2_act2/tp2_act2 (main)
$ git commit -m "Agregando mi-archivo.txt"
[main (root-commit) 0342845] Agregando mi-archivo.txt
1 file changed, 1 insertion(+)
create mode 100644 mi-archivo.txt

Santi@DESKTOP-1Q0K68D MINGW64 ~/Desktop/repo_tp2_act2/tp2_act2 (main)
$ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 267 bytes | 267.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/santipace/tp2_act2.git
 * [new branch]      main -> main

Santi@DESKTOP-1Q0K68D MINGW64 ~/Desktop/repo_tp2_act2/tp2_act2 (main)
$ git branch nueva_rama

Santi@DESKTOP-1Q0K68D MINGW64 ~/Desktop/repo_tp2_act2/tp2_act2 (main)
$ git checkout nueva_rama
Switched to branch 'nueva_rama'

Santi@DESKTOP-1Q0K68D MINGW64 ~/Desktop/repo_tp2_act2/tp2_act2 (nueva_rama)
$ git branch
  main
* nueva_rama

Santi@DESKTOP-1Q0K68D MINGW64 ~/Desktop/repo_tp2_act2/tp2_act2 (nueva_rama)
```

```

Santi@DESKTOP-1Q0K68D MINGW64 ~/Desktop/repo_tp2_act2/tp2_act2 (nueva_rama)
$ echo "Este es un archivo en la nueva rama" > archivo-rama.txt

Santi@DESKTOP-1Q0K68D MINGW64 ~/Desktop/repo_tp2_act2/tp2_act2 (nueva_rama)
$ git add .
warning: in the working copy of 'archivo-rama.txt', LF will be replaced by
CRLF the next time Git touches it


Santi@DESKTOP-1Q0K68D MINGW64 ~/Desktop/repo_tp2_act2/tp2_act2 (nueva_rama)
$ git commit -m "Agregando archivo-rama.txt en nueva_rama"
[nueva_rama 57c01bc] Agregando archivo-rama.txt en nueva_rama
1 file changed, 1 insertion(+)
create mode 100644 archivo-rama.txt


Santi@DESKTOP-1Q0K68D MINGW64 ~/Desktop/repo_tp2_act2/tp2_act2 (nueva_rama)
$ git push origin nueva_rama
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 344 bytes | 344.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'nueva_rama' on GitHub by visiting:
remote:   https://github.com/santipace/tp2_act2/pull/new/nueva_rama
remote:
To https://github.com/santipace/tp2_act2.git
 * [new branch]      nueva_rama -> nueva_rama

Santi@DESKTOP-1Q0K68D MINGW64 ~/Desktop/repo_tp2_act2/tp2_act2 (nueva_rama)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Santi@DESKTOP-1Q0K68D MINGW64 ~/Desktop/repo_tp2_act2/tp2_act2 (main)
$ |


```



**tp2\_act2**
Public
Pin
Unwatch
1



**nueva\_rama** had recent pushes 26 seconds ago
Compare & pull request

main
2 Branches
0 Tags

Add file
Code


**santipace**
Agregando mi-archivo.txt
0342845 · 6 minutes ago
1 Commit


mi-archivo.txt
Agregando mi-archivo.txt
6 minutes ago


**README**

### 3) Realizar la siguiente actividad:

Link del repositorio: <https://github.com/santipace/conflict-exercise.git>

## Create a new repository

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

Required fields are marked with an asterisk (\*).

Owner \*

 santipace ▾

/

Repository name \*

conflict-exercise

✔ conflict-exercise is available.

Great repository names are short and memorable. Need inspiration? How about [expert-disco](#) ?

Description (optional)

Este es mi repositorio de la actividad 3



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:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

```
Santi@DESKTOP-1Q0K68D MINGW64 ~
$ git clone https://github.com/santipace/conflict-exercise.git
Cloning into 'conflict-exercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

Santi@DESKTOP-1Q0K68D MINGW64 ~
$ cd conflict-exercise

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (feature-branch)
$ notepad README.md

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (feature-branch)
$ git add README.md

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (feature-branch)
$ notepad README.md

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch 57b8e6e] Added a line in feature-branch
1 file changed, 1 insertion(+)

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (feature-branch)
$ git push origin feature-branch
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 338 bytes | 338.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature-branch' on GitHub by vis
iting:
remote:      https://github.com/santipace/conflict-exercise/pull/ne
w/feature-branch
remote:
To https://github.com/santipace/conflict-exercise.git
 * [new branch]      feature-branch -> feature-branch
```



```

* [new branch]      feature-branch -> feature-branch

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (main)
$ notepad README.md

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (main)
$ git add README.md

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (main)
$ git commit -m "agrego cambios en la main branch"
[main a0c215f] agrego cambios en la main branch
1 file changed, 2 insertions(+)

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (main)
$ git checkout main
Already on 'main'
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (main)
$ notepad README.md

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (main)
$ git add README.md

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (main)
$ git checkout main
Already on 'main'
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (main)
$ git commit -m "cambios main branch"
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (main)
$ git push origin feature-branch
Everything up-to-date

```

```

Everything up-to-date

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (main)
$ git merge feature-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (main|MERGING)
$ notepad README.md

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (main|MERGING)
$ git add README.md

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (main|MERGING)
$ git commit -m "resolvi el merge conflict"
[main 5e97789] resolvi el merge conflict

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (main)
$ git push origin main
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 597 bytes | 597.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local object
.
To https://github.com/santipace/conflict-exercise.git
c4eac2f..5e97789 main -> main

Santi@DESKTOP-1Q0K68D MINGW64 ~/conflict-exercise (main)
$

```

## Cambios en el readme:

