

# Práctico 2: Git y GitHub

## 1) ¿Qué es GitHub?

GitHub es una plataforma de alojamiento de código basada en la nube que utiliza el sistema de control de versiones Git. Permite a los desarrolladores colaborar en proyectos, gestionar repositorios, revisar código, realizar pull requests y más. GitHub ofrece tanto repositorios públicos (gratis) como privados (de pago en planes gratuitos limitados).

## 2) ¿Cómo crear un repositorio en GitHub?

1. Inicia sesión en GitHub.
2. Haz clic en "New" (o "+" > "New repository").
3. Ingresa un nombre para el repositorio.
4. Elige si será público o privado.
5. Opcional: Añade un README.md, .gitignore o licencia.
6. Haz clic en "Create repository".

## 3) ¿Cómo crear una rama en Git?

`git branch nombre-de-la-rama` # Crea la rama

## 4) ¿Cómo cambiar a una rama en Git?

`git checkout nombre-de-la-rama` # Cambia a la rama

## 5) ¿Cómo fusionar ramas en Git?

`git merge nombre-de-la-rama`

## 6) ¿Cómo crear un commit en Git?

`git commit -m "Mensaje descriptivo del cambio"`

## 7) ¿Cómo enviar un commit a GitHub?

`git push origin nombre-de-la-rama`

## 8) ¿Qué es un repositorio remoto?

Es una copia del repositorio alojada en un servidor (como GitHub, GitLab o Bitbucket). Permite sincronizar cambios entre varios colaboradores.

## 9) ¿Cómo agregar un repositorio remoto a Git?

`git remote add origin https://github.com/usuario/repositorio.git`

**10) ¿Cómo empujar cambios a un repositorio remoto?**

git push origin nombre-de-la-rama

**11) ¿Cómo tirar de cambios de un repositorio remoto?**

git pull origin nombre-de-la-rama

**12) ¿Qué es un fork de repositorio?**

Un fork es una copia personal de un repositorio de otro usuario en tu cuenta de GitHub, permitiéndote modificarlo sin afectar el original.

**13) ¿Cómo crear un fork de un repositorio?**

Ve al repositorio en GitHub.

Haz clic en "Fork" (esquina superior derecha).

Elige tu cuenta como destino.

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

Haz cambios en tu fork.

Ve a la pestaña "Pull requests" en GitHub.

Haz clic en "New pull request".

Compara tu rama con la original y envía el PR.

**15) ¿Cómo aceptar una solicitud de extracción?**

Ve al PR en GitHub.

Revisa los cambios.

Haz clic en "Merge pull request".

**16) ¿Qué es un etiqueta en Git?**

Es una marca en un commit específico (ej. versiones como v1.0.0).

**17) ¿Cómo crear una etiqueta en Git?**

git tag v1.0.0 # Etiqueta ligera

git tag -a v1.0.0 -m "Versión 1.0.0" # Etiqueta anotada

**18) ¿Cómo enviar una etiqueta a GitHub?**

git push origin v1.0.0 # Envía una etiqueta

git push origin --tags # Envía todas las etiquetas

**19) ¿Qué es un historial de Git?**

Es un registro de todos los commits realizados en un repositorio, mostrando autores, fechas y cambios.

**20) ¿Cómo ver el historial de Git?**

git log # Muestra commits

git log --oneline # Versión resumida

git log --graph # Con gráfico de ramas

**21) ¿Cómo buscar en el historial de Git?**

git log --grep "palabra clave" # Busca en mensajes

git show commit-hash # Muestra cambios de un commit

**22) ¿Cómo borrar el historial de Git?**

git checkout --orphan nueva-rama

git add -A

git commit -m "Nuevo inicio"

```
git branch -D main # Borra la rama main
git branch -m main # Renombra la rama actual
git push -f origin main # Fuerza el push (cuidado)
```

### **23) ¿Qué es un repositorio privado en GitHub?**

Es un repositorio solo accesible para usuarios con permisos. Requiere suscripción en planes gratuitos (ahora GitHub permite repos privados gratis con limitaciones).

### **24) ¿Cómo crear un repositorio privado en GitHub?**

Al crear un repo, selecciona "Private" en lugar de "Public".

### **25) ¿Cómo invitar a alguien a un repositorio privado en GitHub?**

Ve a "Settings" > "Collaborators".

Ingresa el nombre de usuario o email.

Haz clic en "Add collaborator".

### **26) ¿Qué es un repositorio público en GitHub?**

Es visible para todos, pero solo los colaboradores pueden editarlo (a menos que sea un fork).

### **27) ¿Cómo crear un repositorio público en GitHub?**

Al crear el repo, selecciona "Public".

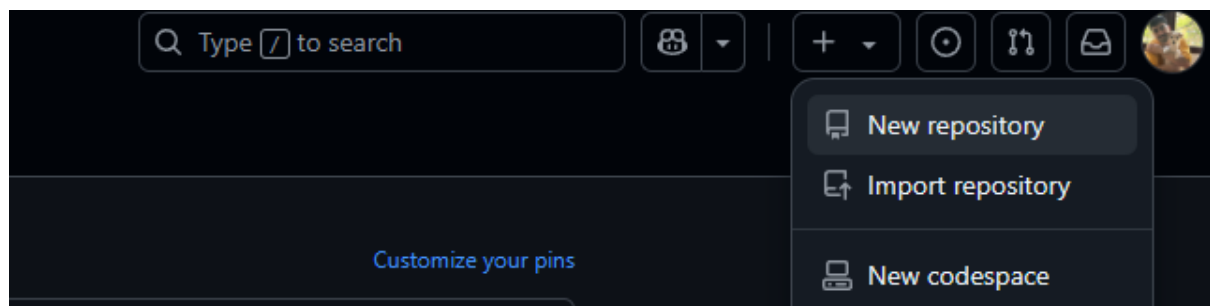
Comparte el enlace: <https://github.com/usuario/repo>.

Usa la opción "Share" en GitHub.

## **28. ¿Cómo compartir un repositorio público en GitHub?**

### **2) Realizar la siguiente actividad:**

Crear un repositorio.



Dale un nombre al repositorio.

Elige el repositorio sea público.


Inicializa el repositorio con un archivo

## 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 \*

 Gastuzar

Repository name \*

TP.UTN

✔ TP.UTN is available.

Great repository names are short and memorable. Need inspiration? How about **bookish-fiesta** ?

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:

☒ 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.](#)

Choose a license

License: None

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

This will set  main as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

Create repository

### Agregando un Archivo

Crea un archivo simple, por ejemplo, "mi-archivo.txt".

Abrir Git Bash Here en la carpeta donde esta el archivo creado

Realiza los comandos git init, git add . y git commit -m "Agregando mi-archivo.txt" en la línea de comandos.

```

gasto@Gastuzar MINGW64 ~/OneDrive/Desktop/Git TP (main)
$ git init
Initialized empty Git repository in C:/Users/gasto/OneDrive/Desktop/Git TP/.git/

gasto@Gastuzar MINGW64 ~/OneDrive/Desktop/Git TP (master)
$ git add .

gasto@Gastuzar MINGW64 ~/OneDrive/Desktop/Git TP (master)
$ git commit -m "Agregando mi-archivo.txt"
[master (root-commit) ebc0eae] Agregando mi-archivo.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 mi-archivo.txt.txt

```

Al realizarlo en remoto debes conectarte con el repositorio con git remote add origin (URL).

Sube los cambios al repositorio en GitHub con git push origin master(o el nombre de la rama correspondiente)

```

gasto@Gastuzar MINGW64 ~/OneDrive/Desktop/Git TP (master)
$ git remote add origin https://github.com/Gastuzar/TP.UTN.git

gasto@Gastuzar MINGW64 ~/OneDrive/Desktop/Git TP (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 231 bytes | 231.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/Gastuzar/TP.UTN/pull/new/master
remote:
To https://github.com/Gastuzar/TP.UTN.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

```

Al trabajar sobre la rama master creamos una rama nueva.

Ya que en el repositorio remoto la rama principal es main

The screenshot shows the GitHub interface for a repository named 'TP.UTN' owned by 'Gastuzar'. At the top, a notification bar indicates 'master had recent pushes 2 minutes ago' with a 'Compare & pull request' button. Below this, the repository's main branch is set to 'main', with 2 branches and 0 tags. A search bar and buttons for 'Add file' and 'Code' are visible. The commit history shows an 'Initial commit' by 'Gastuzar' 17 minutes ago, with commit hash 'b13bc55'. Below the commit list, the 'README' file is displayed, containing the text 'TP.UTN'.

### 3) Realizar la siguiente actividad:

#### Paso 1: Crear un repositorio en GitHub

- Ve a GitHub e inicia sesión en tu cuenta.
- Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.
- Asigna un nombre al repositorio.
- Opcionalmente, añade una descripción.
- Marca la opción "Initialize this repository with a README".
- Haz clic en "Create repository".

**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 \***

Gastuzar / TP.UTN

✓ TP.UTN is available.

Great repository names are short and memorable. Need inspiration? How about **bookish-fiesta** ?

**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:**

☒ **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.](#)

**Choose a license**

License: **None**

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

This will set **main** as the default branch. Change the default name in your [settings](#).

You are creating a public repository in your personal account.

**Create repository**

#### Paso 2: Clonar el repositorio a tu máquina local

- Copia la URL del repositorio (usualmente algo como `https://github.com/tuusuario/conflict-exercise.git`).

- Abre la terminal o línea de comandos en tu máquina.
- Clona el repositorio usando el comando: `git clone https://github.com/tuusuario/conflict-exercise.git`
- Entra en el directorio del repositorio

The screenshot shows a VS Code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a project named 'TP.UTN' with files like 'README.md', '.gitignore', and 'mi-archivo.txt.txt'. The terminal window shows the output of the command `git clone https://github.com/Gastuzar/TP.UTN.git`, which successfully clones the repository into the 'TP.UTN' directory.

```

PS C:\Users\gasto\OneDrive\Desktop\Git TP> git clone https://github.com/Gastuzar/TP.UTN.git
Cloning into 'TP.UTN'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (6/6), done.
PS C:\Users\gasto\OneDrive\Desktop\Git TP>

```

### Paso 3: Crear una nueva rama y editar un archivo

- Crea una nueva rama llamada feature-branch: `git checkout -b feature-branch`
- Abre el archivo README.md en un editor de texto y añade una línea nueva, por ejemplo:  
Este es un cambio en la feature branch.
- Guarda los cambios y haz un commit: `git add README.md git commit -m "Added a line in feature-branch"`

The screenshot shows a terminal window with the following commands and output:

```

PS C:\Users\gasto\OneDrive\Desktop\Git TP> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\gasto\OneDrive\Desktop\Git TP> git commit -m "Added a line in feature-branch"
On branch feature-branch
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    TP.UTN/

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\gasto\OneDrive\Desktop\Git TP>

```

### Paso 4: Volver a la rama principal y editar el mismo archivo

- Cambia de vuelta a la rama principal (main): `git checkout main`
- Edita el archivo README.md de nuevo, añadiendo una línea diferente:  
Este es un cambio en la main branch.

- Guarda los cambios y haz un commit: `git add README.md` `git commit -m "Added a line in main branch"`

The screenshot shows a VS Code editor with a file named `README.md` open. The file content is as follows:

```

1 # TP.UTN
2 Este es un cambio en la feature branch.
3 Este es un cambio en la rama principal
4

```

Below the editor, the **TERMINAL** tab is active, displaying the output of the following commands:

```

PS C:\Users\gasto\OneDrive\Desktop\Git TP> git add README.md
PS C:\Users\gasto\OneDrive\Desktop\Git TP> git commit -m "Added a line in main branch"
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        TP.UTN/

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\gasto\OneDrive\Desktop\Git TP>

```

#### Paso 5: Hacer un merge y generar un conflicto

- Intenta hacer un merge de la feature-branch en la rama main: `git merge feature-branch`
- Se generará un conflicto porque ambos cambios afectan la misma línea del archivo `README.md`.



```
PS C:\Users\gasto\OneDrive\Desktop\Git TP> git merge feature-branch
>>
fatal: refusing to merge unrelated histories
```

#### Paso 6: Resolver el conflicto

- Abre el archivo README.md en tu editor de texto. Verás algo similar a esto:

<<<<<<< HEAD

Este es un cambio en la main branch. =====

Este es un cambio en la feature branch. >>>>>>> feature-branch

- Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.
- Edita el archivo para resolver el conflicto y guarda los cambios (Se debe borrar lo marcado en verde en el archivo donde estes solucionando el conflicto. Y se debe borrar la parte del texto que no se quiera dejar).
- Añade el archivo resuelto y completa el merge: `git add README.md git commit -m "Resolved merge conflict"`

```
1 # TP.UTN
2 Este es un cambio en la rama main
```

PROBLEMAS    SALIDA    CONSOLA DE DEPURACIÓN    **TERMINAL**    PUERTOS    COMENTARIOS

```
-u, --[no-]update      update tracked files
--[no-]renormalize     renormalize EOL of tracked files (implies -u)
-N, --[no-]intent-to-add
                        record only the fact that the path will be added later
-A, --[no-]all          add changes from all tracked and untracked files
--[no-]ignore-removal  ignore paths removed in the working tree (same as --no-all)
--[no-]refresh         don't add, only refresh the index
--[no-]ignore-errors   just skip files which cannot be added because of errors
--[no-]ignore-missing  check if - even missing - files are ignored in dry run
--[no-]sparse          allow updating entries outside of the sparse-checkout cone
--[no-]chmod (+|-)x    override the executable bit of the listed files
--[no-]pathspec-from-file <file>
                        read pathspec from file
--[no-]pathspec-file-nul
                        with --pathspec-from-file, pathspec elements are separated with NUL
```

```
PS C:\Users\gasto\OneDrive\Desktop\Git TP> git add README.md
PS C:\Users\gasto\OneDrive\Desktop\Git TP> git commit -m "Resolved merge conflict"
>>
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        TP.UTN/

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\gasto\OneDrive\Desktop\Git TP>
```

Paso 7: Subir los cambios a GitHub

- Sube los cambios de la rama main al repositorio remoto en GitHub: `git push origin main`
- También sube la feature-branch si deseas: `git push origin feature-branch`

```
PS C:\Users\gasto\OneDrive\Desktop\Git TP> git push origin main
>>
Everything up-to-date
PS C:\Users\gasto\OneDrive\Desktop\Git TP> git push origin feature-branch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:   https://github.com/Gastuzar/TP.UTN/pull/new/feature-branch
remote:
To https://github.com/Gastuzar/TP.UTN.git
 * [new branch]      feature-branch -> feature-branch
PS C:\Users\gasto\OneDrive\Desktop\Git TP> 
```

#### Paso 8: Verificar en GitHub

- Ve a tu repositorio en GitHub y revisa el archivo README.md para confirmar que los cambios se han subido correctamente.
- Puedes revisar el historial de commits para ver el conflicto y su resolución.

