

Práctico 2: Git y GitHub

Objetivo:

El estudiante desarrollará competencias para trabajar con Git y GitHub, aplicando conceptos fundamentales de control de versiones, colaboración en proyectos y resolución de conflictos, en un entorno simulado y guiado.

Resultados de aprendizaje:

1. Comprender los conceptos básicos de Git y GitHub: Identificar y explicar los principales términos y procesos asociados con Git y GitHub, como repositorios, ramas, commits, forks, etiquetas y repositorios remotos.
2. Manejar comandos esenciales de Git: Ejecutar comandos básicos para crear, modificar, fusionar y gestionar ramas, commits y repositorios, tanto en local como en remoto.
3. Aplicar técnicas de colaboración en GitHub: Configurar y utilizar repositorios remotos, realizar forks, y gestionar pull requests para facilitar el trabajo colaborativo.
4. Resolver conflictos en un entorno de control de versiones: Identificar, analizar y solucionar conflictos de merge generados en un flujo de trabajo con múltiples ramas.

Actividades

- 1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada
Desarrollar las respuestas

- ¿Qué es GitHub?

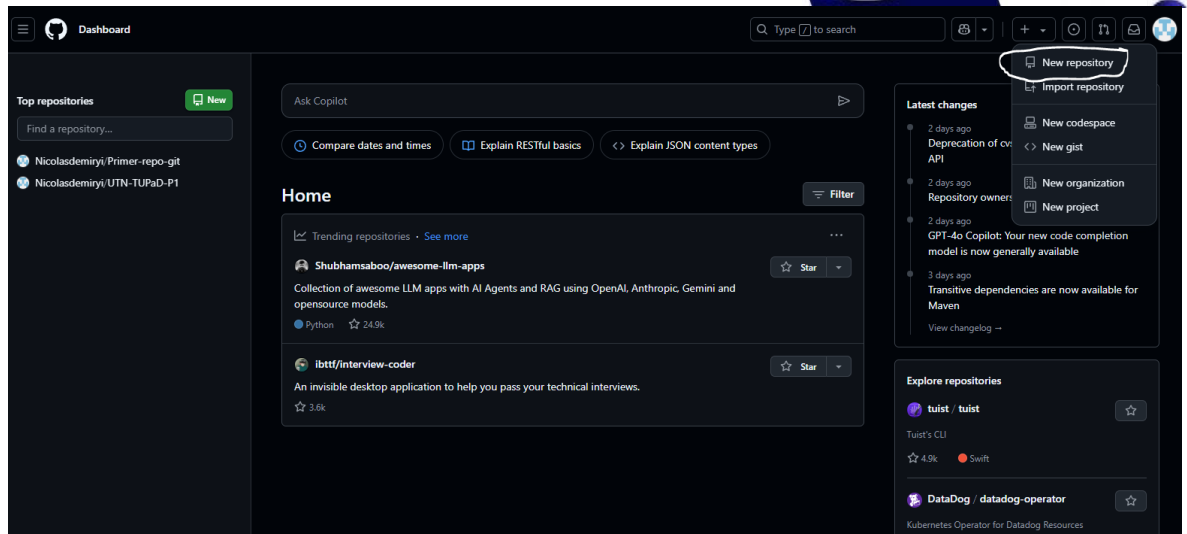
GitHub es una plataforma o comunidad donde se pueden compartir repositorios de forma pública o privada. Esta plataforma basada en la nube permite almacenar, compartir y trabajar con otros usuarios para escribir código.

Almacenar código en un repositorio en GitHub permite:

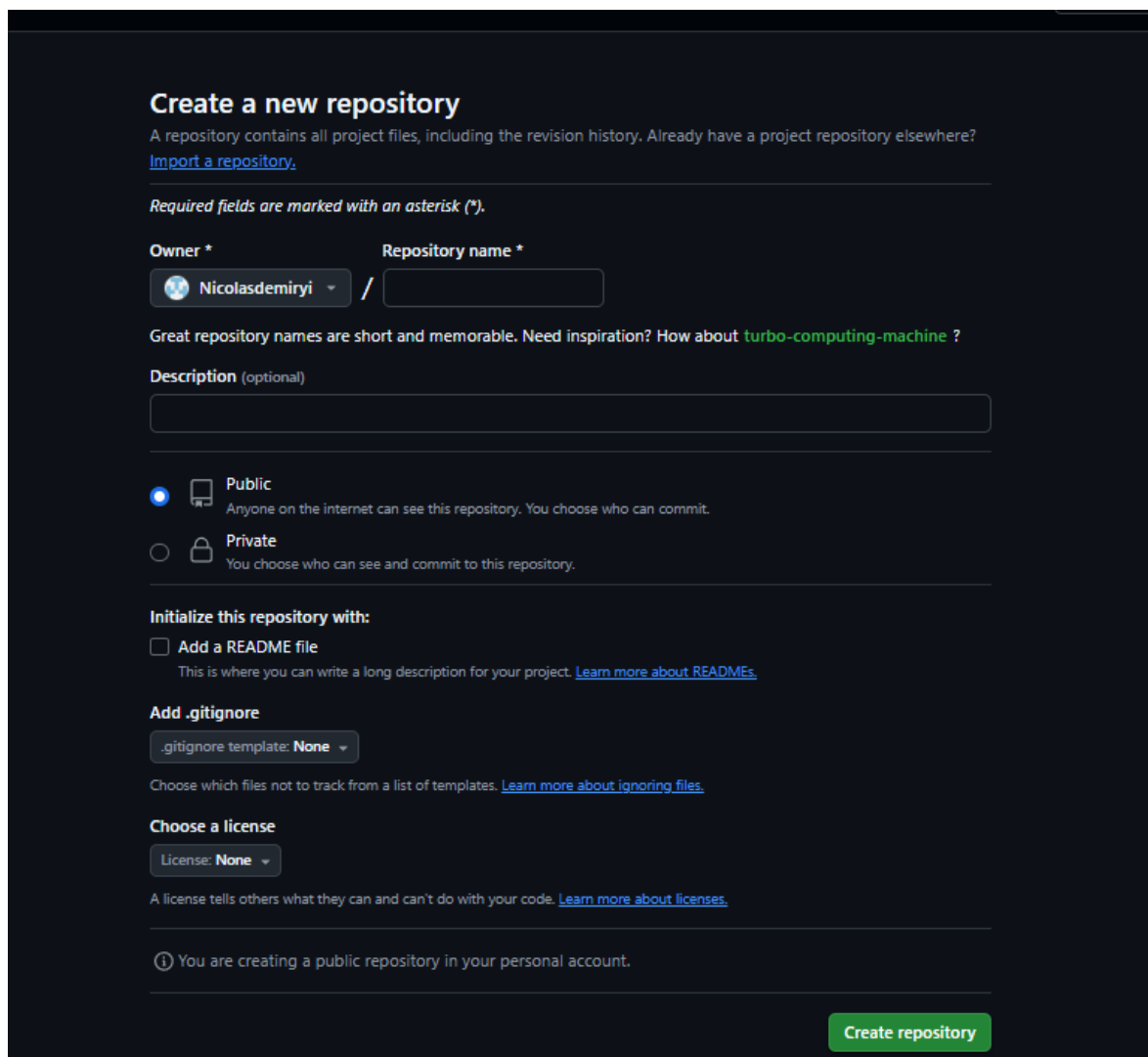
- + Presentar o compartir trabajo.
- + Seguir y administrar cambios en el código y realicen sugerencias para mejorarlo.
- + Colaborar en proyectos compartidos, sin preocuparse de que los cambios afecten al trabajo entre los colaboradores antes de que esté listo para integrarlos.

- ¿Cómo crear un repositorio en GitHub?

Para crear un repositorio en GitHub primero se debe crear una cuenta en GitHub. Una vez dentro de la cuenta seleccionamos la opción redondeada con blanco que dice “New repository” en la imagen debajo:



Luego completamos la información correspondiente en la imagen debajo:

A screenshot of the 'Create a new repository' form on GitHub. The form includes fields for 'Owner' (set to 'Nicolasdemiryi'), 'Repository name', and 'Description' (optional). It also has radio buttons for 'Public' (selected) and 'Private'. Below these are sections for 'Initialize this repository with:' (including 'Add a README file'), 'Add .gitignore' (with a dropdown set to 'None'), and 'Choose a license' (with a dropdown set to 'None'). A green 'Create repository' button is at the bottom right. A note at the bottom states: 'You are creating a public repository in your personal account.'

Ponemos el nombre que se prefiera para el repositorio, añadimos una breve descripción,

elegimos si va a ser público o privado y luego damos click en la opción “Create repository”.

- ¿Cómo crear una rama en Git?

Para crear una rama en Git se debe ejecutar en la consola de Git el comando “git branch NOMBRE-NUEVA-RAMA”

- ¿Cómo cambiar a una rama en Git?

Para cambiar a una rama en Git se utiliza el comando “git checkout NOMBRE-NUEVA-RAMA”.

- ¿Cómo fusionar ramas en Git?

Para fusionar ramas en Git utilizamos el comando “git merge” seguido del nombre de la rama que queremos fusionar

- ¿Cómo crear un commit en Git?

Para crear un commit en Git se utiliza el comando “git commit -m” luego de -m se escribe el mensaje que se desea dejar sobre las modificaciones o avances hechos

- ¿Cómo enviar un commit a GitHub?

Con el comando “git push” insertamos las confirmaciones realizadas en la rama local en un repositorio remoto.

- ¿Qué es un repositorio remoto?

Un repositorio remoto es el lugar donde se guarda o almacena el código. Dicho código se guardará en GitHub y se podrá acceder a través de un enlace URL.

- ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto nuevo, se usa el comando “git remote add” en la terminal, dentro del directorio donde está almacenado el repositorio.

- ¿Cómo empujar cambios a un repositorio remoto?

Para “empujar” o subir confirmaciones de cambios a un repositorio remoto se utiliza el comando “git push”.

- ¿Cómo tirar de cambios de un repositorio remoto?

Para “tirar” cambios de un repositorio remoto a un repositorio local utilizamos el comando “git pull”

- ¿Qué es un fork de repositorio?

Un fork o “bifurcación” es un nuevo repositorio que comparte código y configuración de visibilidad con el repositorio original. Los forks se utilizan para iterar ideas o cambios antes de proponerlos en el repositorio original.

- ¿Cómo crear un fork de un repositorio?

Para crear un fork de un repositorio se deben seguir los siguientes pasos

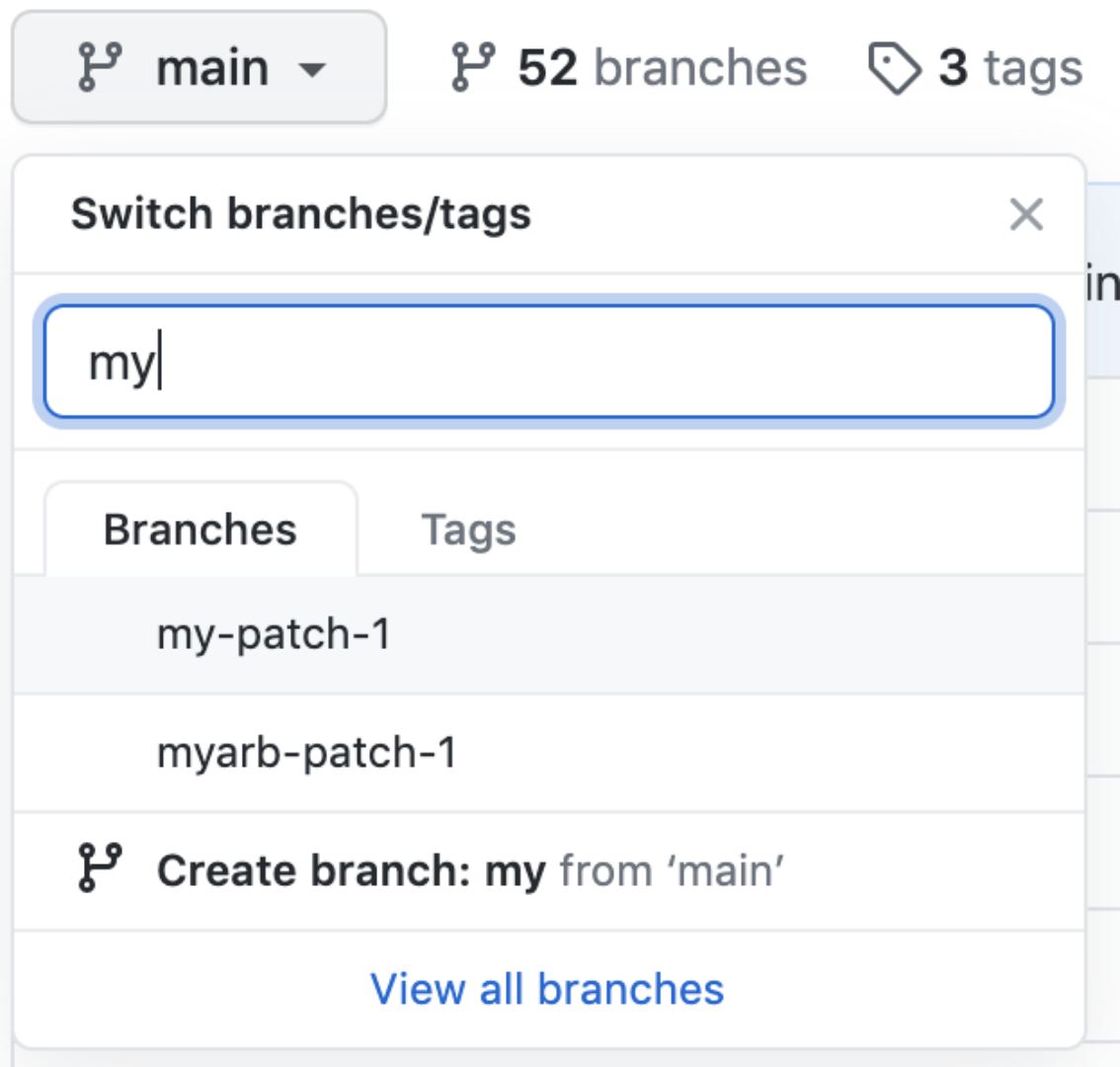
- 1- Ir a la página que se quiere clonar
- 2- Pulsar el botón Fork
- 3- Si es necesario, cambiar el nombre o agregar una descripción
- 4- Pulsar “Create fork”

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

Para enviar una solicitud de extracción (pull request) a un repositorio, en primer lugar, los cambios realizados se proponen en una rama, lo cual garantiza que la rama predeterminada contenga únicamente trabajo finalizado y aprobado.

Los pasos son los siguientes:

- 1- En GitHub, se navega hasta la página principal del repositorio.
- 2- En el menú “Branch” (Rama), se elige la rama que contiene las confirmaciones hechas:



- 3- Encima de la lista de archivos, en el banner amarillo, hacemos clic en “Compare & pull request” para crear una solicitud de incorporación de cambios para la rama asociada.

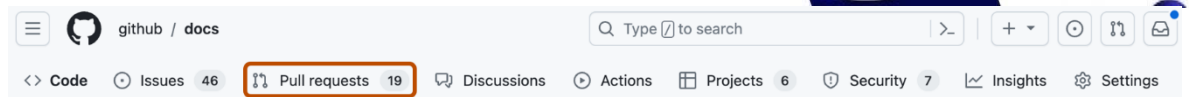


- 4- Usando el menú desplegable de la rama base para seleccionar la rama en la que quiera combinar los cambios, y después, se usa el menú desplegable de la rama de comparación para elegir la rama de tema en la que ha realizado los cambios
- 5- Escribir título y una descripción para la solicitud de extracción

- ¿Cómo aceptar una solicitud de extracción?

Para aceptar una solicitud de extracción (pull request) se siguen los siguientes pasos:

- 1- En el dentro del repositorio, se hace clic en “Pull requests”



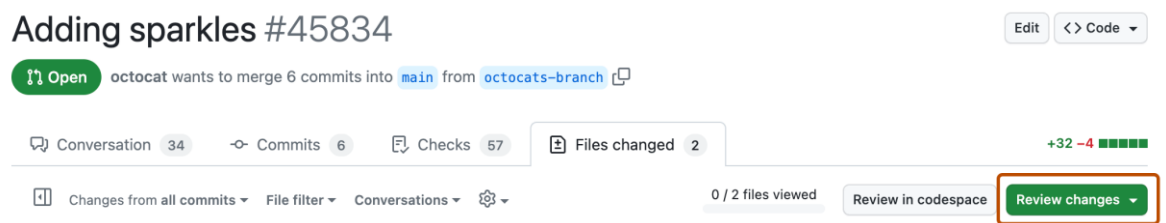
2- En la lista de solicitudes de incorporación de cambios, se hace clic en la que se quiera revisar.

3- En la solicitud de incorporación de cambios, se hace clic en “Files changed”



4- Revisar los cambios en la solicitud de cambios y, opcionalmente, se comentan líneas o archivos específicos.

5- Encima del código cambiado, se hace clic en “Review changes”



6- Luego escribimos algún comentario que resuma la retroalimentación de cambios propuestos.

7- Seleccionamos la opción “Aprobar” para aprobar la combinación de cambios propuestos en la solicitud de incorporación de cambios

8- Enviamos revisión

- ¿Qué es un etiqueta en Git?

Una etiqueta en git (git tag) es una marca que se aplica a una confirmación para identificarla. Se utilizan para marcar puntos importantes en el historial de un repositorio.

- ¿Cómo crear una etiqueta en Git?

Para crear una etiqueta en Git utilizamos el comando “git tag”

- ¿Cómo enviar una etiqueta a GitHub?

Para enviar una etiqueta a GitHub utilizamos el comando “git push” seguido del nombre de la etiqueta.

- ¿Qué es un historial de Git?

Es un historial de cambios realizados en un repositorio, almacenado como un gráfico de instantáneas. Cada instantánea se llama confirmación y contiene un puntero a una o varias confirmaciones anteriores.

- ¿Cómo ver el historial de Git?

Para ver el historial de Git se utiliza el comando “git log”.

- ¿Cómo buscar en el historial de Git?

Para buscar en el historial de Git usamos el comando “git log -L” para ver historial de una función o línea de código.

- ¿Cómo borrar el historial de Git?

Para borrar el historial de Git utilizamos el comando “git ribase – i”

- ¿Qué es un repositorio privado en GitHub?

Es un espacio donde se puede guardar código, archivos y revisiones de forma segura y solo visible para los usuarios autorizados.

- ¿Cómo crear un repositorio privado en GitHub?

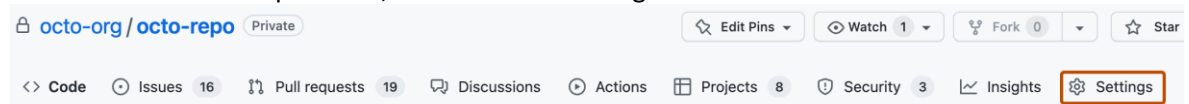
Seguimos los mismos pasos que para crear un repositorio público pero seleccionamos la opción para que sea privado.

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

Se deben seguir los siguientes pasos:

1- Solicitar nombre de usuario a la persona que se está invitando como colaboradora.

2- En el nombre del repositorio, hacer clic en “Settings”



3- En la sección “Acceso” de la barra lateral hacer clic en colaboradores

4- Hacer clic en Agregar personas

- ¿Qué es un repositorio público en GitHub?

Es un repositorio donde se puede almacenar, compartir y trabajar en código de forma abierta.

- ¿Cómo crear un repositorio público en GitHub?

De la forma que fue explicada anteriormente eligiendo la opción de “Público”

- ¿Cómo compartir un repositorio público en GitHub?

Enviando el URL

2) Realizar la siguiente actividad:

- Crear un repositorio.
 - Dale un nombre al repositorio.
 - Elije el repositorio sea público.
 - Inicializa el repositorio con un archivo.
- Agregando un Archivo
 - Crea un archivo simple, por ejemplo, "mi-archivo.txt".
 - Realiza los comandos `git add .` y `git commit -m "Agregando mi-archivo.txt"` en la línea de comandos.
 - Sube los cambios al repositorio en GitHub con `git push origin main` (o el nombre de la rama correspondiente).

- Creando Branchs
 - Crear una Branch
 - Realizar cambios o agregar un archivo
 - Subir la Branch

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, por ejemplo, conflict-exercise.
- Opcionalmente, añade una descripción.
- Marca la opción "Initialize this repository with a README".
- Haz clic en "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:

```
cd conflict-exercise
```

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"
```

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"
```

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.

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"
```

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
```

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.