Introducción

¡Hola! 👋 Te damos la bienvenida a nuestra clase de *“Git & GitHub: Repositorios y colaboración en proyectos*”.

En esta clase aprenderemos de forma amigable y práctica cómo utilizar estas herramientas y cómo pueden facilitar nuestra vida al trabajar con proyectos de programación.

En las clases anteriores vimos como Git es una herramienta fundamental para ordenar las versiones y cambios de tus documentos o proyectos, similar a como Google Drive te permite guardar versiones de tus archivos y compartirlos con otros.

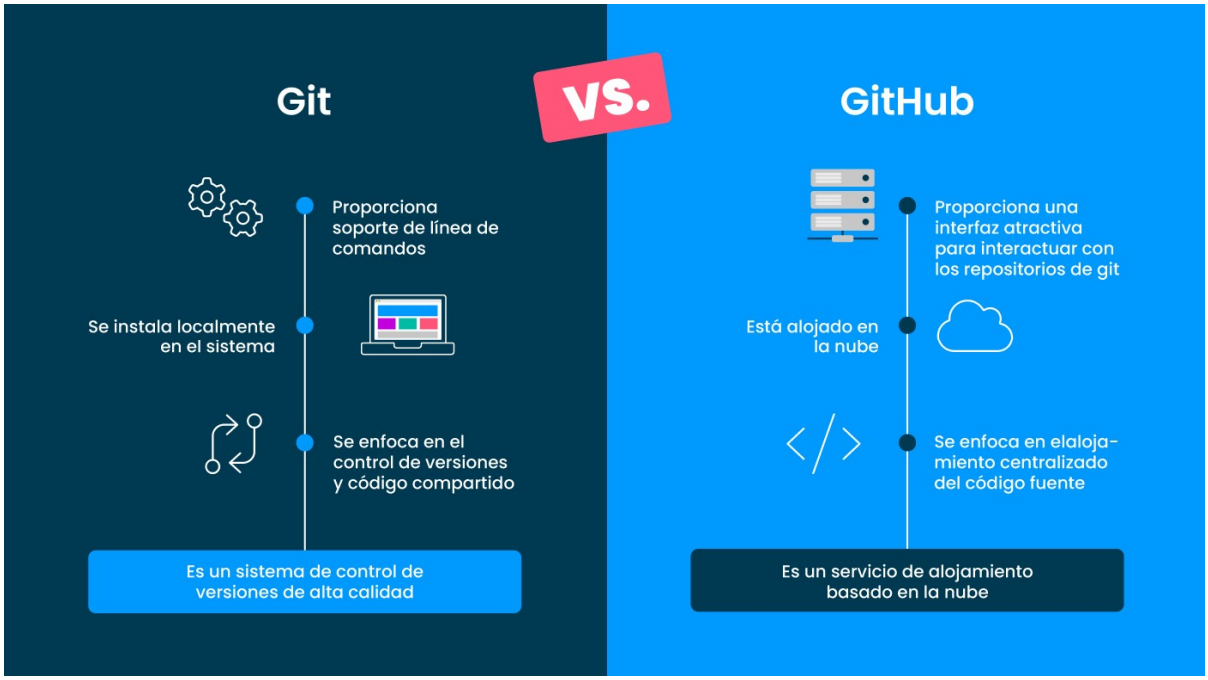
Hoy veremos cómo puedes continuar sacándole provecho y utilizar Git en conjunto con Github.

¡Comencemos! 🚀

GitHub & Git

¿Recuerdas que en el curso de Programación Web desde cero te pedimos que crearas una cuenta en Github para poder compartir tu proyecto? ¡Genial! Ahora vamos a ver cómo la podemos utilizar para alojar nuestros proyectos y aprovechar sus funciones adicionales.

GitHub es un servicio de hosting que nos permite almacenar nuestros proyectos en línea. **Lo interesante es que GitHub utiliza Git para el control de versiones, lo que nos brinda ventajas como la colaboración en tiempo real y la opción de compartir repositorios públicos y privados**.



Hasta ahora, tenemos una versión de nuestro proyecto en nuestra computadora (*repositorio local*). Sin embargo, si queremos colaborar y compartirlo, necesitamos crear una versión en línea (*repositorio remoto*) en GitHub para que otros puedan acceder a él.

En esta clase, exploraremos dos formas de lograrlo:

1. Subir un repositorio local a GitHub:

- Crearemos un nuevo repositorio en GitHub.
- Enlazaremos nuestro repositorio local con el repositorio remoto en GitHub.
- Subiremos los cambios al repositorio remoto en GitHub.

2. Descargar un repositorio remoto de GitHub a nuestra computadora:

- Clonaremos un repositorio remoto en nuestra computadora.
- Realizaremos cambios en los archivos de nuestro repositorio local.
- Subiremos los cambios al repositorio remoto en GitHub.

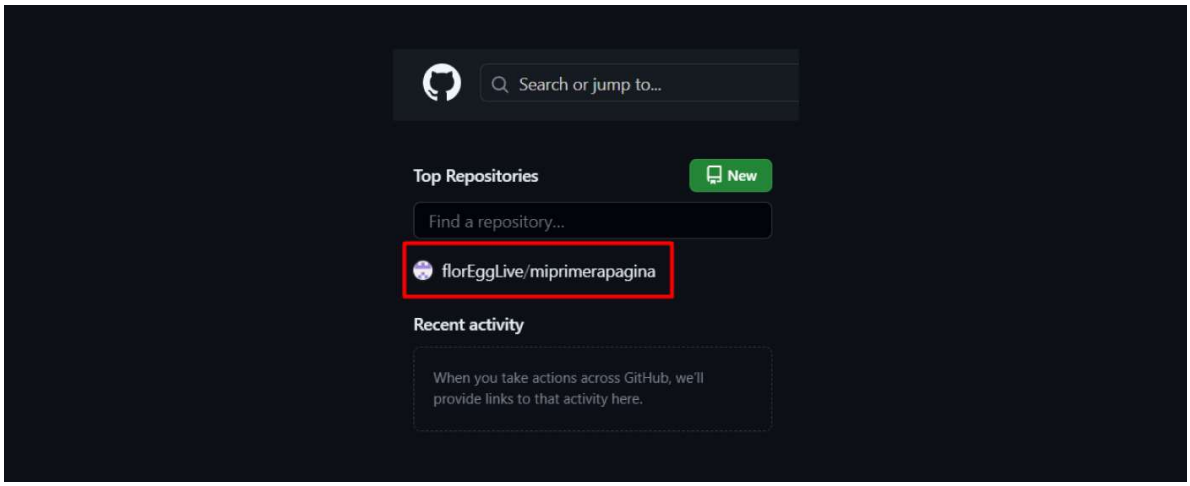
Con estas dos opciones, podrás gestionar tu proyecto tanto localmente como en GitHub, permitiendo la colaboración y la disponibilidad en línea. ¡Asegúrate de aprovechar al máximo esta herramienta para mejorar tus habilidades de desarrollo web!

Subir repositorio local a Github

Durante el curso de *Programación Web desde Cero* habíamos subido el repositorio de nuestra página web con GitHub Desktop, ¿te acuerdas?.

¡Bien! Ten presente que puedes acceder al mismo iniciando sesión con tu usuario y contraseña en la página de ☞github.com.

En la columna lateral izquierda deberías poder ver el link de tu repositorio. Te mostramos un ejemplo:



En el siguiente video, te mostramos cómo crear un nuevo repositorio pero directamente desde la página de Github (es decir, sin usar la herramienta de GitHub Desktop como hicimos anteriormente):

Una vez creado, vamos a tener que **conectar nuestro repositorio local con el repositorio remoto** que acabamos de crear. Para esto, te guiamos cómo hacer para lograrlo en el siguiente video:

¡Perfecto! Hasta acá tenemos tanto lo mismo en nuestro repositorio local como en nuestro repositorio remoto.

Ahora bien, *¿qué sucede si queremos hacer cambios en el sitio y los queremos volver a subir para que el repo remoto quede actualizado?*

¡Pongámoslo en práctica!

1. Realizar cambios en tu página web, como modificar texto o agregar nuevas imágenes.

2. Abrir tu repositorio local en Git Bash.

3. Verificar los cambios realizados utilizando el comando "git status" en la terminal. Esto te mostrará los archivos que han sido modificados.

4. Agregar los cambios al *área de preparación* (“staging area”).

5. Realizar el *commit* de los cambios con una breve descripción de los mismos. Debería aparecer un mensaje confirmando la aplicación de los cambios.

6. Finalmente, subir los cambios al repositorio remoto en Github utilizando el comando: “**git push origin main**”. Esto enviará tus cambios y los actualizará en el repositorio en línea.

¡Fantástico! Ahora tus cambios realizados en el repositorio local se han reflejado en el repositorio remoto en Github. Tu página web estará actualizada y lista para que otros puedan ver los cambios realizados.

Git push

Ahora que hemos creado nuestro repositorio en GitHub, es importante comprender cómo utilizar el comando **git push** para subir nuestros cambios al repositorio remoto.

El comando git push nos permite enviar los cambios realizados en nuestro repositorio local al repositorio remoto en GitHub. Esto es especialmente útil cuando hemos realizado modificaciones en nuestra página web y queremos que esas actualizaciones estén disponibles para que otros las vean.

Hay dos formas de utilizar este comando:

- **git push:** Cuando ejecutamos simplemente *git push*, el comando enviará los cambios locales de la rama actual al repositorio remoto configurado por defecto. Es útil cuando trabajamos en una rama local y queremos enviar los cambios a la rama correspondiente en el repositorio remoto.
- **git push origin “nombre de la rama”**: Si estamos trabajando con varias ramas en nuestro repositorio, podemos especificar el nombre de la rama en lugar de dejarlo en blanco. Por ejemplo, al ejecutar *git push origin feature*, se enviarán los cambios de la rama local *"feature"* al repositorio remoto en la misma rama.

💡 *"origin" es el nombre comúnmente utilizado para referirse al repositorio remoto principal. Sin embargo, es posible tener múltiples repositorios remotos configurados y utilizar diferentes nombres en lugar de "origin".*

Descargar un repositorio remoto de GitHub

A menudo, nos encontramos con repositorios ya creados y subidos a GitHub. En esta situación para poder trabajarlos, **es necesario descargar el repositorio a nuestra computadora, realizar cambios y luego subir esos cambios, teniendo en cuenta las modificaciones que puedan haber realizado otros colaboradores**.

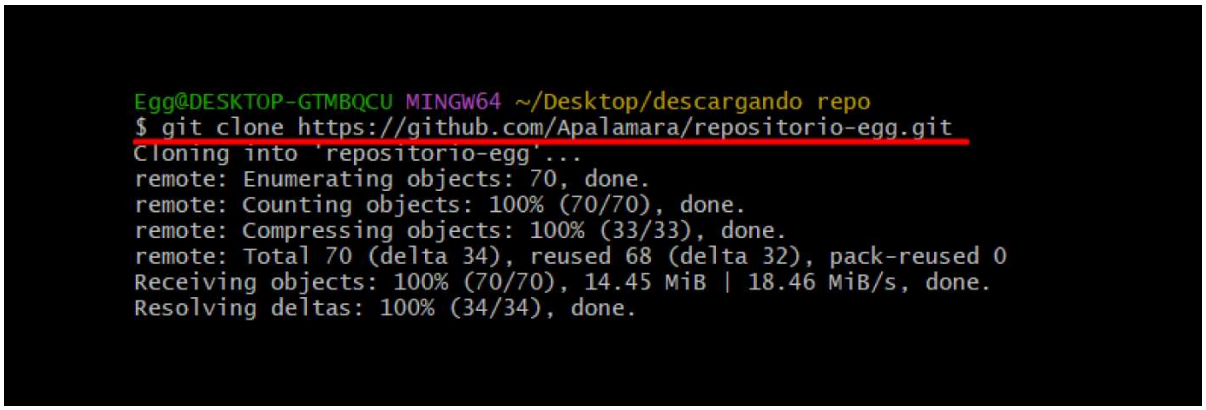
A continuación, crearemos un nuevo repositorio en GitHub, para luego poder trabajar sobre el mismo y colaborar .

Para ello, comenzaremos siguiendo los pasos del siguiente video:

“Git clone”

Como recién vimos, **el comando git clone es el que se utiliza para descargar un repositorio remoto a nuestra computadora**.

Para clonar un repositorio, simplemente necesitamos proporcionar la URL del repositorio remoto, como por ejemplo: *git clone “URL del repositorio”.*



Git creará una carpeta nueva en nuestra computadora con el nombre del repositorio y copiará todos los archivos y la estructura del proyecto dentro de ella incluyendo todo su historial de versiones y ramas.

Clonar un repositorio es útil cuando queremos trabajar en un proyecto existente y colaborar con otros desarrolladores. Una vez que hemos clonado el repositorio, podemos realizar cambios locales, crear nuevas ramas y luego enviar nuestros cambios al repositorio remoto utilizando el comando *git push*, una vez que tengamos los permisos correspondientes (esto lo veremos más tarde).

Ahora ya aprendiste a utilizar el comando *"git clone"* y puedes empezar a trabajar en repositorios existentes.

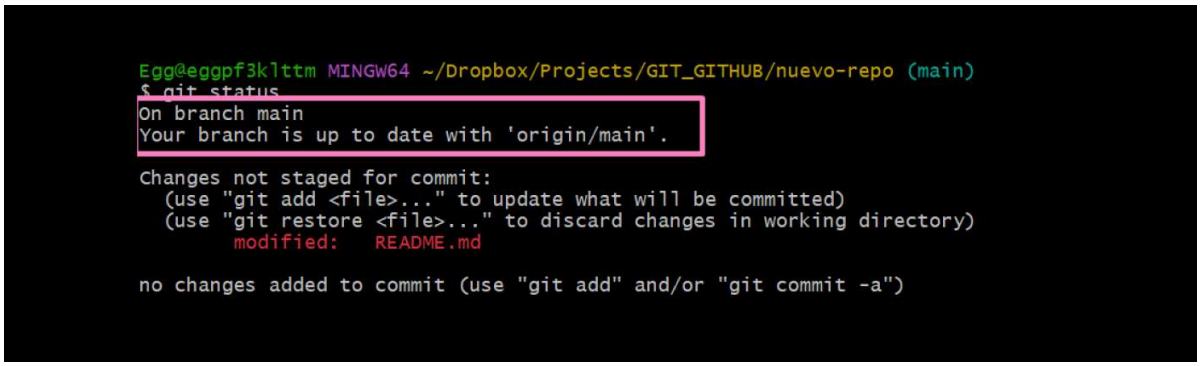
¡Manos a la obra!

1. En Visual Studio Code, abrir la carpeta donde clonamos el repositorio que creamos en el video.

2. Realizar un cambio en el archivo README. Por ejemplo, puedes colocar el texto de: *“Probando un cambio en el archivo README”*.

3. Verificar el estado del archivo con **"git status"**. Deberías ver el mensaje de: “*Your branch is up to date con origin/main”,* lo cual indica que los archivos que tienes en tu computadora están actualizados y coinciden con los archivos en GitHub.

Esto significa que no hay cambios pendientes de subir y estás al día con la última versión del repositorio remoto.

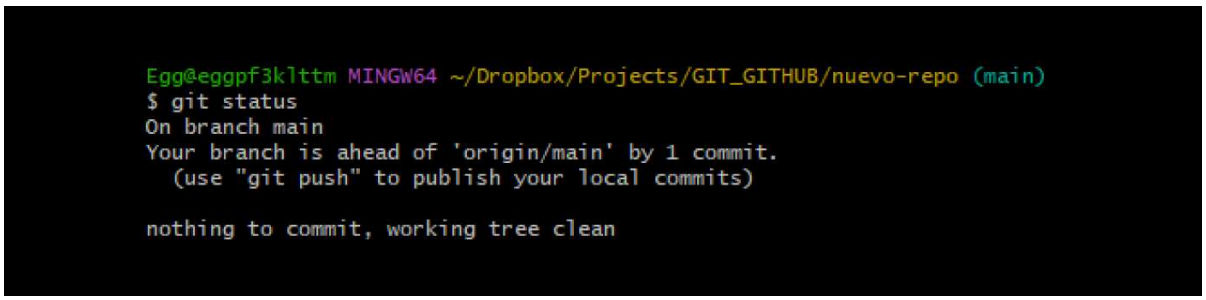


¿Por qué sucede esto si hicimos un cambio? Recuerda que hasta que realicemos un commit, ese cambio solo se encuentra en el *directorio de trabajo* (*"working directory"*). Es necesario confirmar esos cambios mediante el commit para que se reflejen en el repositorio y estén listos para ser subidos.

4. Para ello, agregar el archivo de la carpeta al *área de preparación* ("staging area") con el comando: **"git add ."** y realizar un commit de los cambios utilizando el comando: **git commit -m "Mensaje descriptivo"**.

5. Ahora que hemos realizado cambios en el repositorio local, verificar nuevamente el estado utilizando el comando **"git status"**.

Veremos que nos indica que estamos adelantados por un cambio (*"ahead of (...) by one commit"*) en comparación con el repositorio en GitHub. Esto significa que tenemos un commit adicional en nuestra rama local que aún no se ha reflejado en el repositorio remoto.

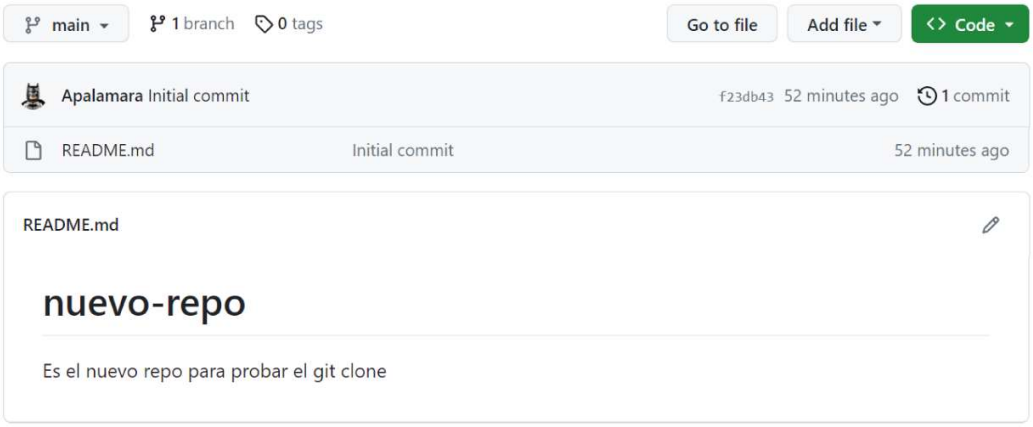


Por consiguiente, es necesario sincronizar esos cambios con el repositorio remoto (GitHub) para que estén alineados.

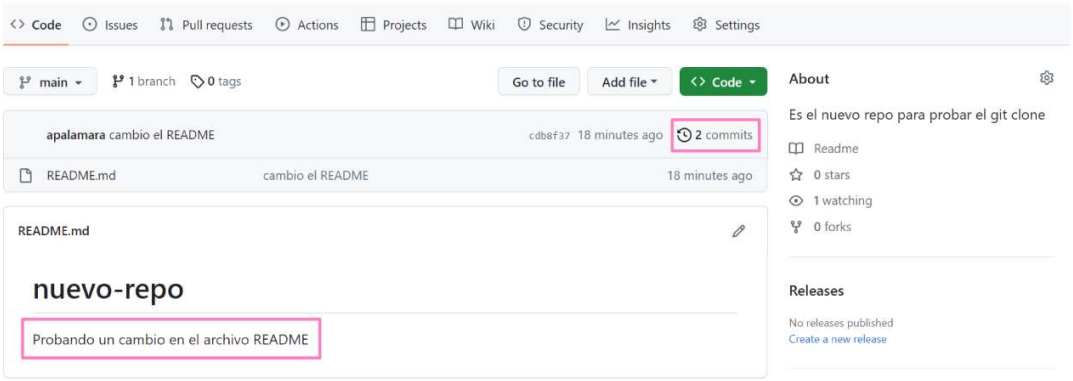
6. Utilizar el comando que aprendimos anteriormente de **"git push"** para publicar los cambios en Github.

¡Listo! 👌 Te mostramos un ejemplo del antes y después que deberías ver reflejado:

Antes



Después



Descargar cambios remotos a repositorio local

Como ya vimos al principio, "*git push"* es el comando que se utiliza para enviar cambios locales a un repositorio remoto.

Pero, *¿qué pasa si hacemos cambios en un repositorio remoto y necesitamos descargarlos en el local?* (es decir, hacer el camino inverso).

¡Veamos! Sigue estos pasos:

1. Realizar un cambio en el archivo *README d*e tu repositorio remoto en GitHub. Puedes agregar una nueva línea de texto o modificar el contenido existente. Asegúrate de guardar los cambios.

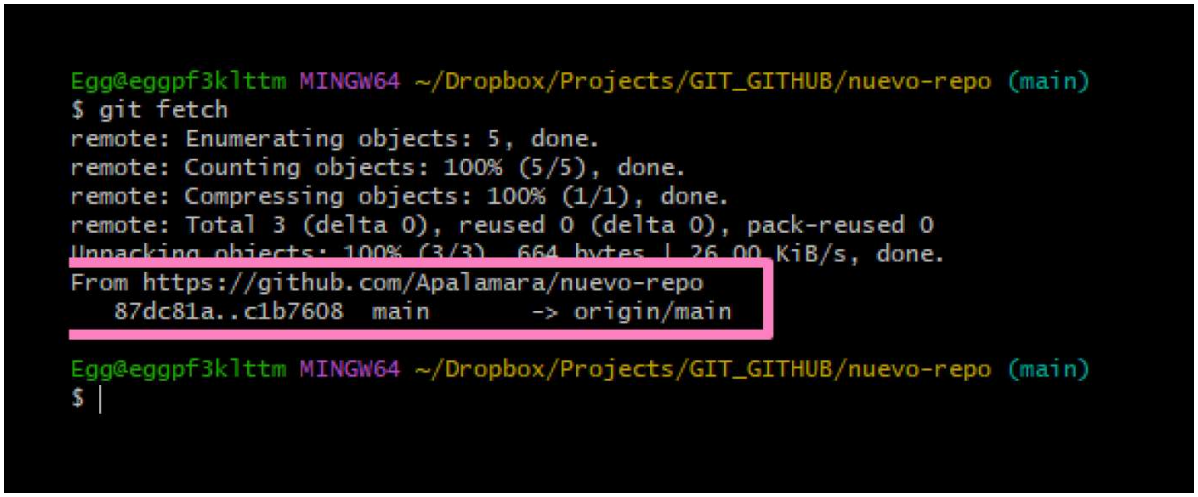
2. Abrir la terminal de Git Bash y navegar hasta el directorio de tu repositorio local utilizando el comando: **"cd ruta-del-repositorio"** (es decir, la carpeta donde clonaste anteriormente el repositorio). Por ejemplo, si tu repositorio está ubicado en la carpeta "Documents/GitRepo", tendrías que ejecutar el comando *"cd Documents/GitRepo"*.

3. Verificar el estado actual de tu repositorio local utilizando el comando **"git status"**. Asegúrate de que no tengas cambios pendientes por confirmar antes de descargar los cambios remotos.

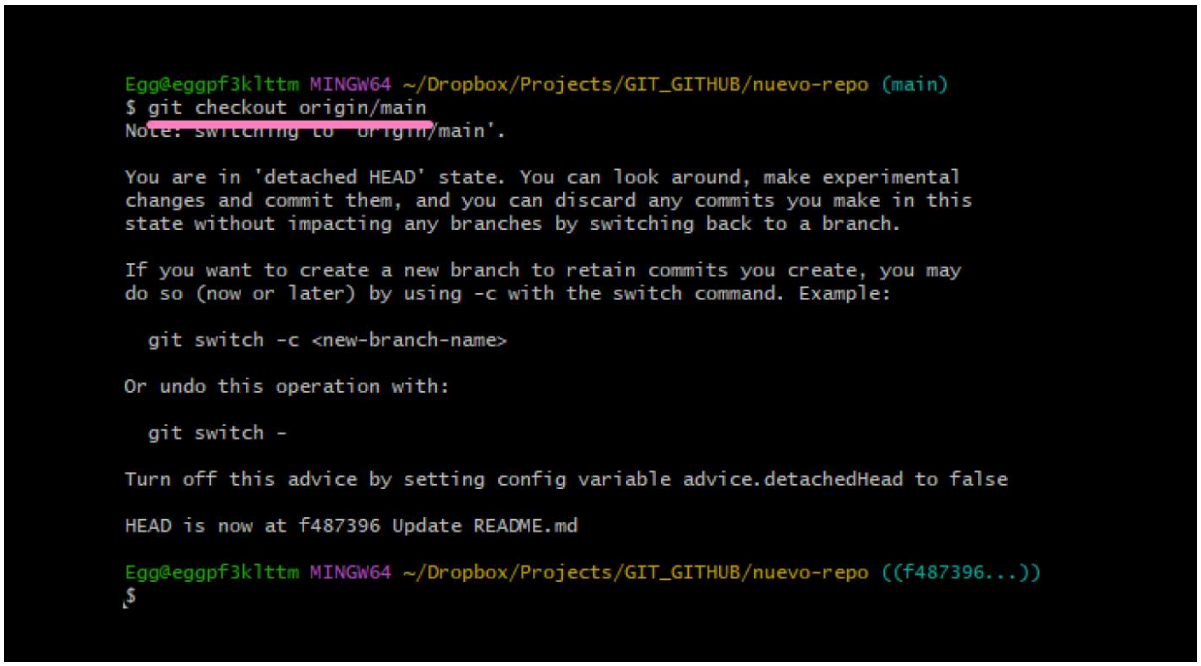
4. Ejecutar el comando **"git fetch"**. Esto buscará y descargará los cambios remotos, pero *no los aplicará automáticamente a tu repositorio local*.

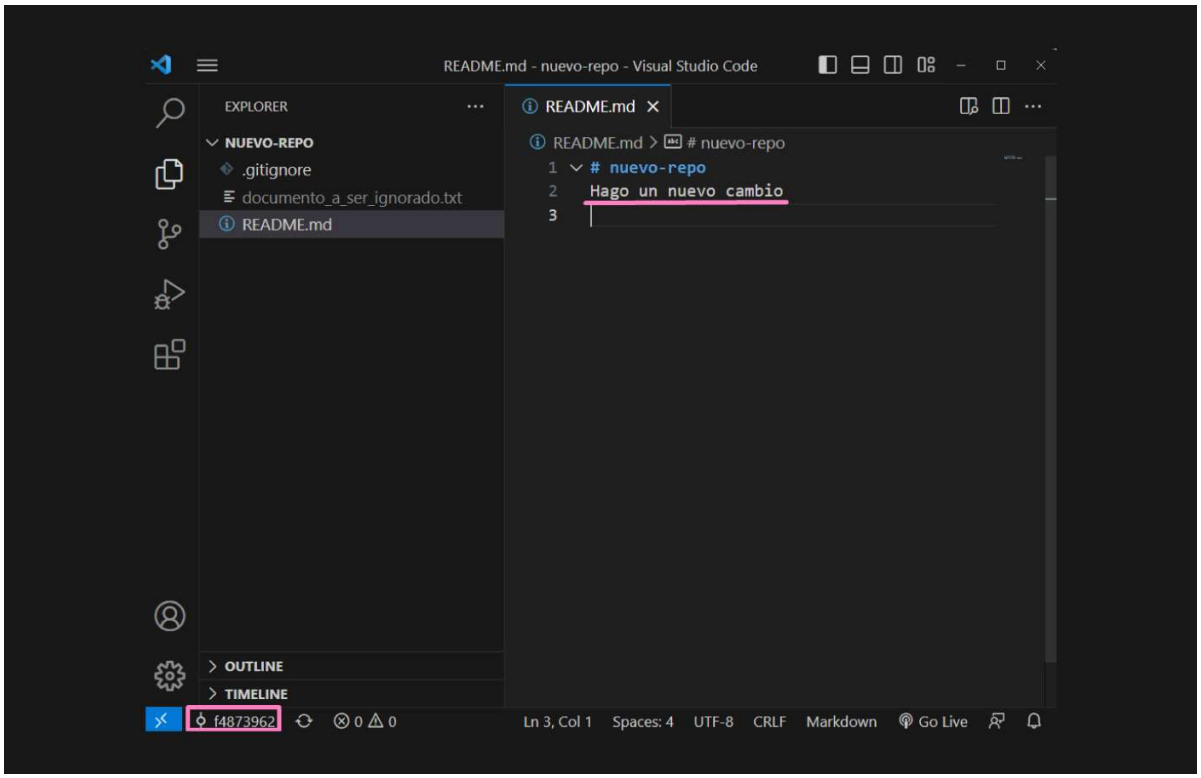
En otra palabras, verás los cambios que se realizaron en *origin/main* que aún no están en *main*:

- **origin/main** → Rama principal en el repositorio remoto de GitHub.
- **main** → Rama principal en el repositorio local.

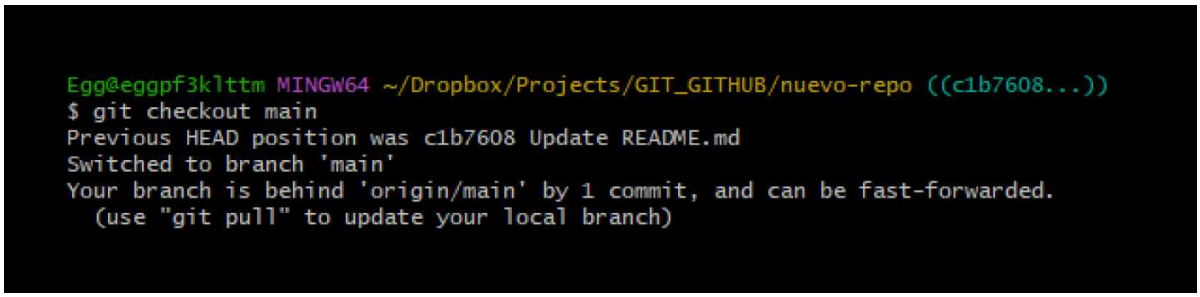


5. Verificar el cambio utilizando el comando de: **“git checkout origin/main”**. Este comando mostrará los cambios que están en *origin/main* en GitHub, sin aún aplicarlos (es decir, sólo los muestra).





6. Luego, volver a la rama local con el comando: **"git checkout main"** (o también podrías utilizar: *"git switch main*").



Ahora deberías ver un mensaje que indica que estás "*adelantado por X commits"* en relación con el repositorio remoto. Esto significa que los cambios remotos han sido descargados correctamente.

7. Ahora, para fusionar los cambios remotos en tu rama local, ejecutar el comando: **“git pull origin/main”**. Este comando combina los cambios remotos en tu rama actual.

Luego, Git intentará combinar automáticamente los cambios remotos con tu rama local.

💡*Si hay conflictos de fusión, Git te mostrará los archivos afectados y te pedirá que resuelvas los conflictos manualmente. Puedes utilizar una herramienta de resolución de conflictos o editar los archivos manualmente para solucionar los conflictos.*

8. Una vez que los cambios remotos se hayan fusionado exitosamente en tu rama local, puedes verificar nuevamente el estado con **“git status”**. Ahora deberías tener los cambios remotos reflejados en tu repositorio local.

¡Excelente! Ahora has descargado con éxito los cambios realizados en el repositorio remoto hacia tu repositorio local. Tu repositorio local estará actualizado con las últimas modificaciones y listo para seguir trabajando.

Recuerda que es importante sincronizar regularmente tu repositorio local con el remoto para mantener un flujo de trabajo colaborativo eficiente.

Git fetch” & “Git pull

Como vimos en la actividad,el comando **git fetch se utiliza para verificar los cambios realizados en el repositorio remoto sin fusionarlos automáticamente con el repositorio local**. Es decir, te permite conocer si el repositorio remoto ha tenido algún cambio, pero no aplica esos cambios directamente (no realiza el “merge” de los mismos).

Por otro lado, el comando **git pull es utilizado para descargar el contenido de un repositorio remoto e inmediatamente actualizar un repositorio local para que ambos tengan la misma información**. En otras palabras, realiza tanto el *git fetch* como el *git merge* en una sola operación. Descarga los cambios remotos y los fusiona automáticamente con la rama local.

Desafío

Antes de avanzar con el desafío, repasemos todo lo visto hasta el momento. Aprendimos:

- A pasar de un repositorio local a uno remoto en GitHub.
- A utilizar el comando **"git push**" para llevar los cambios hechos en el repositorio local al repositorio remoto en GitHub.
- A pasar de un repositorio remoto de GitHub a uno local utilizando el comando de **"git clone"**.
- A utilizar los comandos **"git fetch"** y **"git pull"** para ver y llevar los cambios hechos en el repositorio remoto de GitHub al repositorio local.

Ahora llegó el momento de aplicarlos pero de manera colaborativa. El desafío de hoy consiste en hacer cambios en el repositorio de un/a compañer@ de nuestra mesa de trabajo, y permitir que ese compañero también realice cambios en nuestro repositorio.

¡Manos a la obra!

1. Descargar el repositorio de tu compañer@ en tu ordenador.

2. Realizar al menos un cambio en ese mismo repositorio. Por ejemplo: puedes agregar una nueva línea de texto, modificar el contenido existente, etc.

3. Subir los cambios al repositorio remoto de tu compañer@ en GitHub.

❗Es importante tener en cuenta que para llevar a cabo este paso, GitHub solicitará permisos adicionales. Asegúrate de haber obtenido los permisos necesarios antes de proceder. Es decir, al momento de descargar el repositorio, no se requieren permisos; sin embargo, para realizar modificaciones en el repositorio, es necesario contar con dichos permisos.

En el siguiente video te mostramos cómo dar u obtener estos permisos:

4. En este punto, ya deberías tener aplicados en tu repositorio remoto de GitHub los cambios realizados por tu compañer@. Ahora, llevar esos cambios a tu repositorio local para mantenerlo actualizado.

¡Desafío terminado! 🤖

Como resultado, deberíamos ver aplicados:

- Los cambios realizados en el repositorio remoto de nuestro compañer@.
- Los cambios que nuestro compañer@ hizo tanto en nuestro repositorio remoto como en el local.

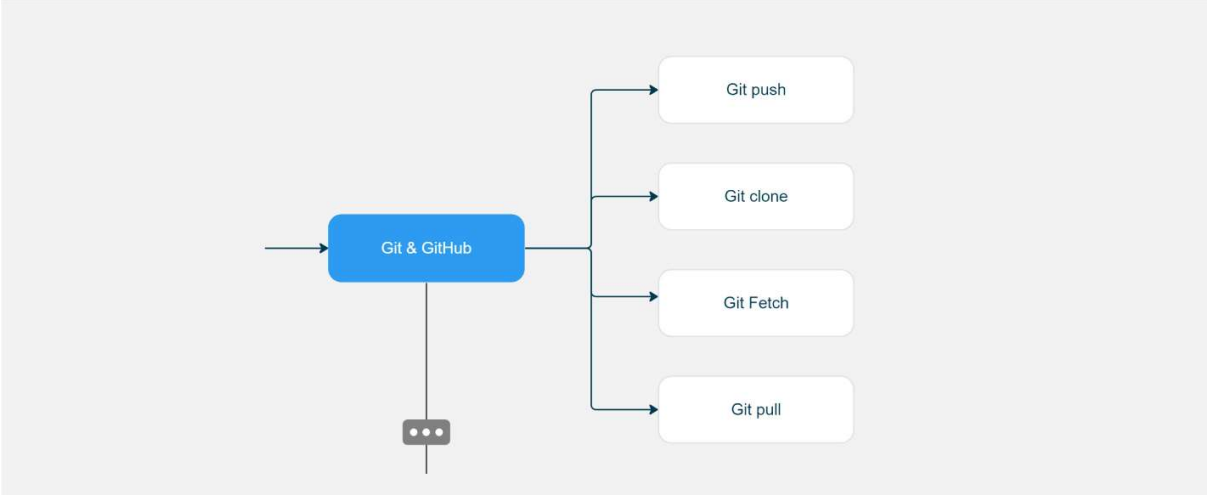
Si te quedaron dudas sobre el desafío anterior podés ver los siguientes videos que te van a ayudar a esclarecer los pasos

Mapa de conceptos vistos

¡Has completado la clase del día hoy! 🎉

Hoy aprendiste sobre la integración entre Git y GitHub, cómo subir un repositorio local a GitHub utilizando el comando **"git push"** y cómo descargar un repositorio remoto de GitHub a tu computadora utilizando el comando **"git clone"**.

También vimos cómo descargar cambios remotos a tu repositorio local utilizando los comandos **"git fetch"** y **"git pull"**.



Además, exploramos la importancia de la colaboración y te presentamos un desafío para practicar tus habilidades trabajando en equipo.

¡Felicitaciones por todo lo que has hecho hasta ahora y sigue adelante en tu viaje de desarrollo web!

Hasta la próxima 👋