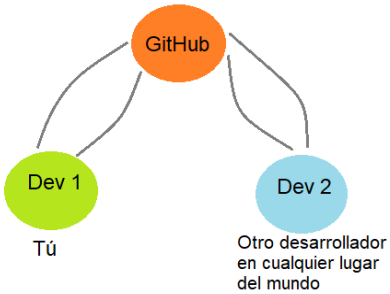
B) Repositorio remoto

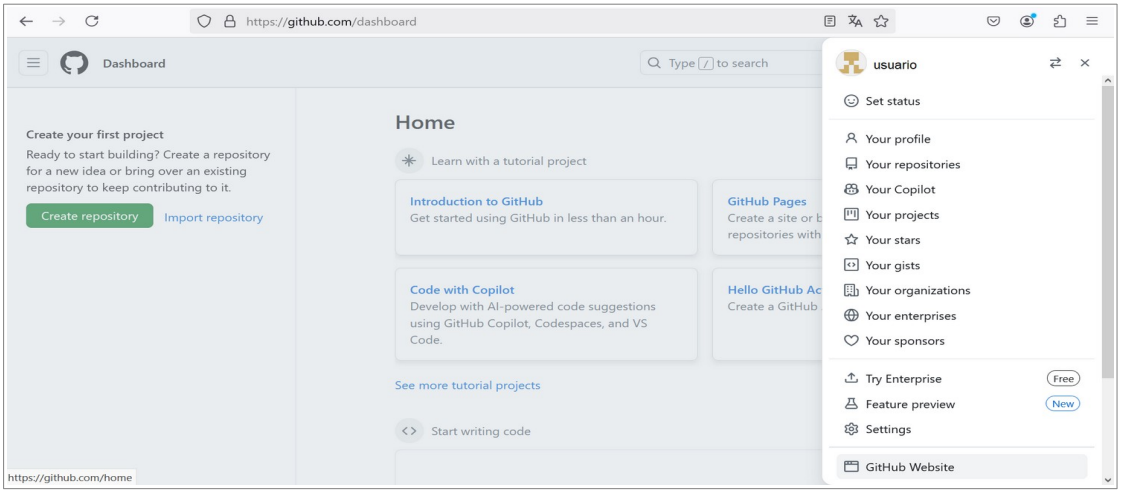
Hasta ahora has usado comandos Git para trabajar con un repositorio local, en tu equipo.

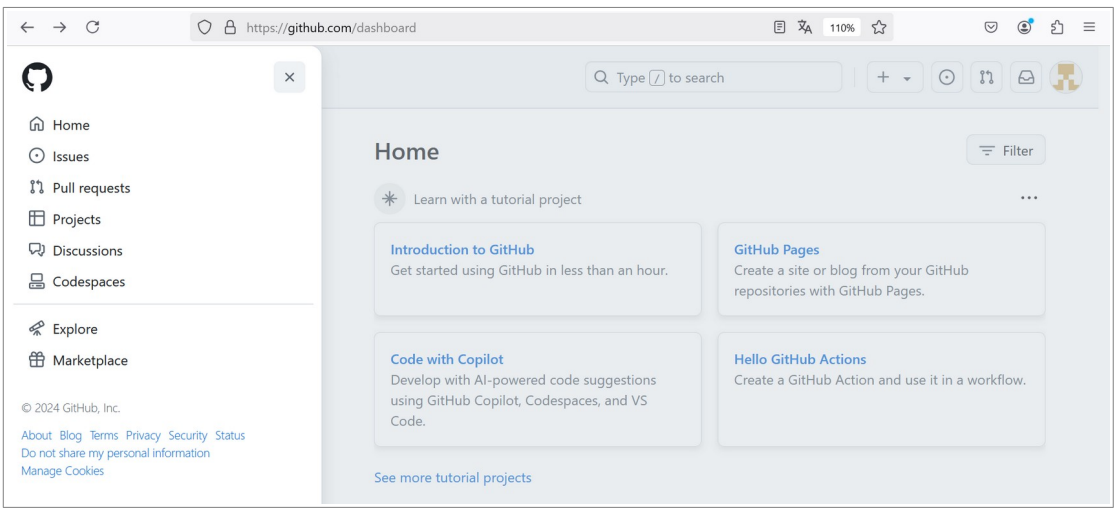
Git es un software de control de versiones para el desarrollo y mantenimiento de aplicaciones software, siendo de gran ayuda cuando las aplicaciones tienen un gran número de archivos de código fuente. Git fue diseñado y creado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de software, su objetivo es llevar un registro de los cambios que se van haciendo en los archivos compartidos en un repositorio de código.

GitHub es una plataforma de desarrollo colaborativo, que permite alojar proyectos en la nube creando repositorios remotos, y que usa el sistema de control de versiones Git. El código de los proyectos alojados en GitHub se almacena generalmente de forma pública, siendo la plataforma más importante de colaboración para proyectos de código abierto.

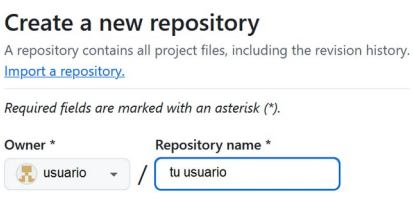


Crea una cuenta en GitHub.





1. Hay dos opciones para crear repositorios en GitHub:

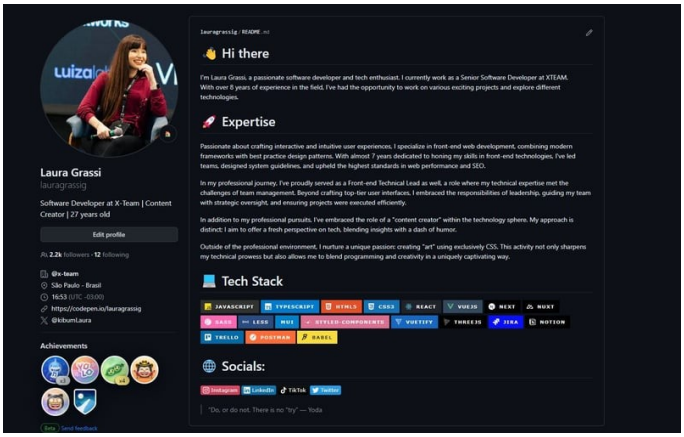


a) Crearlo en local y subirlo a GitHub.

b) Crearlo directamente en GitHub y descargarlo.

Antes de crear el repositorio de trabajo en GitHub, vas a **crear un primer repositorio "especial" en GitHub**, que será "tu página de presentación" en GitHub, ponle el nombre de tu usuario y añade el archivo README, este archivo se usa para describir los proyectos. Licencia no es necesario que añadas, pero es un tema importante si deseas proteger tu proyecto, en GitHub suelen haber muchos proyectos open source. Busca información sobre las características de las licencias que muestra GitHub al crear repositor.io

Este repositorio "especial" que acabas de crear, te permite entrar a tu página web **https://github.com/*tuusuario***, que es tu Overview o página principal de tu GitHub, en la que puedes incluir una descripción general de tu actividad en organizaciones, tus diferentes proyectos y repositorios, equipos, etc. Con lenguaje markdown y html puedes crear una plantilla para tu Overview, no es objetivo de curso pero, si tienes tiempo, puedes investigar y cambiar tu Overview.



Indica la dirección de tu Overview de GitHub: *https://github.com/tuusuario*

2. Seguimos centrados en el trabajo desde terminal, para conectarte a tu GitHub desde tu terminal deberás **autenticarte**, la seguridad en el acceso a tus repositorios es importante. Tienes varios métodos para hacerlo, más info: https://docs.github.com/es/authentication.

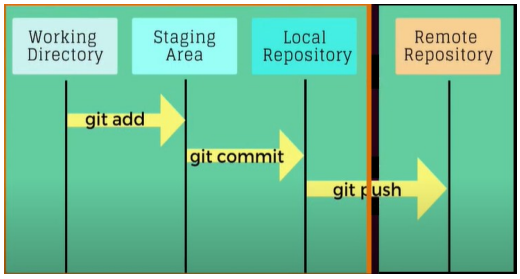
Un método es crear una conexión segura o SSH (Secure Shell), SSH es un protocolo para administración remota que permite a los usuarios controlar y modificar sus servidores remotos a través de Internet mediante un mecanismo de autenticación con clave pública y clave privada, es un protocolo de red destinado principalmente a la conexión con máquinas a las que accedemos por línea de comandos. Sirve para establecer unas credenciales con GitHub, de forma que GitHub, cuando te conectas desde el terminal de tu máquina, sepa que eres un usuario con permisos para trabajar allí.

Otro método de autenticación es crear un "access token" (combinación de números y letras) de GitHub, este token es usado como alternativa para reemplazar la contraseña al autenticarse en GitHub desde la línea de comandos, antes se conectaba con usuario y contraseña, pero GitHub sustituyó el uso de la contraseña por el token porque es más seguro.

Crea un token en GiHub desde Profile/Settings/Developer settings/Personal access tokens, genera uno nuevo token, indica nombre y tiempo de expiración, selecciona solo la función repo, para

control de repositorios privados, copia el token generado y guárdalo en un lugar seguro (en tu carpeta personal) porque desaparecerá de la página.

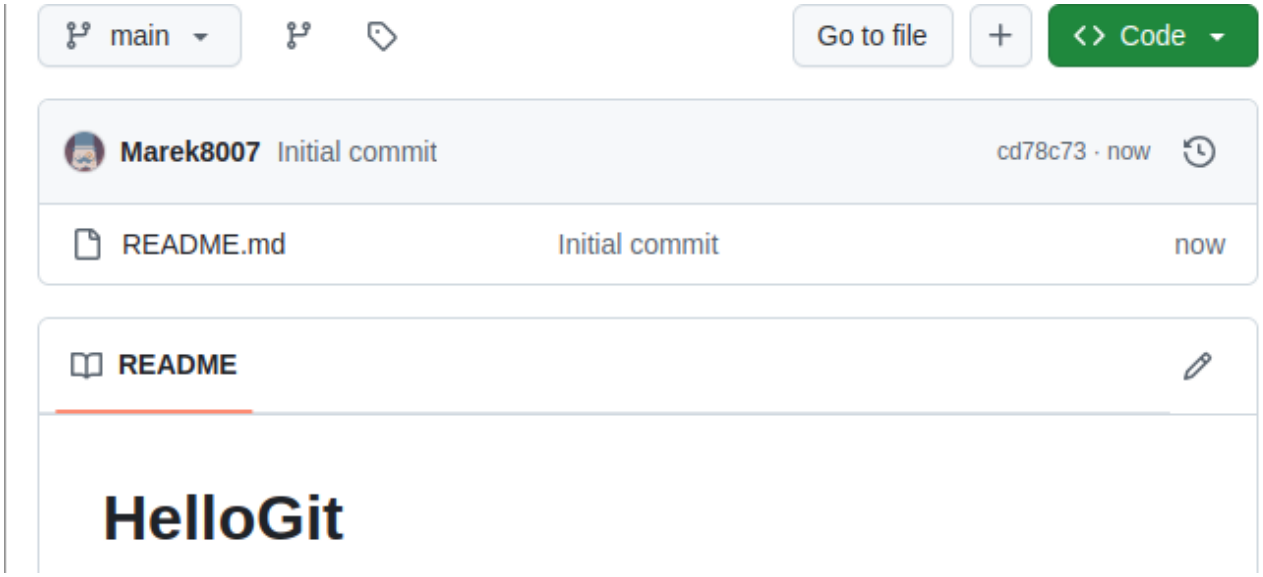
La siguiente imagen es un resumen de lo que has trabajado hasta ahora, la parte izquierda representa el repositorio local en tu equipo, y la parte derecha representa el repositorio remoto alojado en GitHub al que subirás lo que tienes en tu repositorio local, usando el comando **git push**.



Recuerda que **Staging Area** y **Local Repository** son controlados por Git, y **Working Directory** es tu área de trabajo, en la que está el código de tu proyecto.

Lo más habitual es trabajar en el repositorio local de forma sincronizada con el repositorio remoto, compartido con más personas.

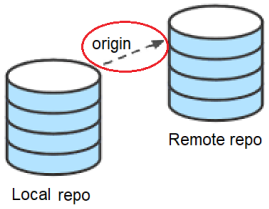
3. **Crea un nuevo repositorio remoto en GitHub** con el nombre **HelloGit,** que lo vas a usar para subir el contenido del proyecto en tu repositorio local en el que has trabajado.



Renombra la rama master de tu repositorio local con el nombre **main**, para que coincida con el nombre de la rama en el repositorio remoto, GitHub crea nuevos proyectos con el nombre de rama main. → git branch -m master main

```
marmiqlis@INF1-03:/media/marmiqlis/Festplatte/repo/HelloGit$ git branch -m master main
```

- Antes de subir algo a GitHub, debes informar a Git sobre la ubicación del repositorio remoto al que quieres subirlo. A esa ubicación se la conoce como "remote" y no es más que una URL apuntando al repositorio remoto, cuando creas un "remote" en tu local, con el comando git remote add, se le pone un nombre al "remote" que, por convención, suele ser el nombre de "**origin**". **Crea en tu repositorio local el enlace a tu repositorio remoto:**

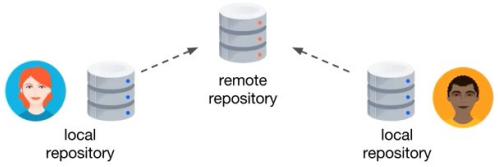


→ git remote add origin https://github.com/tuusuario/HelloGit.git

Si actualizas el repositorio remoto de GitHub, lo podrás ver actualizado con todo lo que tenías en local. Incluye captura con tu terminal y GitHub.

```
/repo/HelloGit$ git remote add origin https://github.com/Marek8007/HelloGit.git
```

- En el caso de que clonaras un repositorio remoto de GitHub a local, el nuevo repositorio local que se crearía ya estaría configurado con su origin, apuntando al repositorio remoto clonado.



4. Crea en tu repositorio remoto de GitHub el fichero **README.md**, con la opción "Add file" con el texto del nombre del proyecto y tu nombre y apellidos, guarda cambios e **imagina que este fichero fue añadido al repo remoto por otro programador**. Tu no lo tienes en tu local.

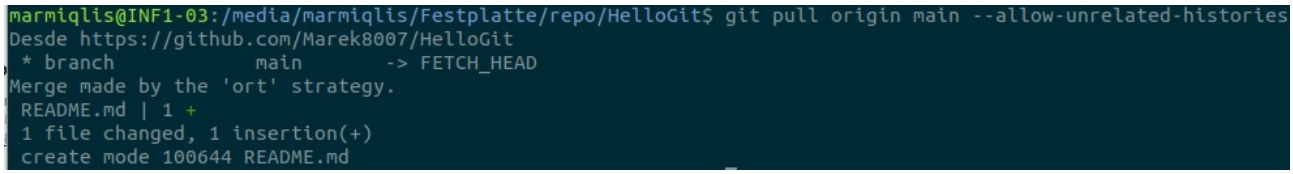
5. Ahora sigue trabajando en tu repo local y modifica el fichero hellogit.py con la linea de código print("Hello GitHub remoto!"), haz un commit con el mensaje "Hello GitHub remoto". Este cambio solo está en tu local, aún no lo has subido a GitHub. Intenta subirlo con el comando **git push** y verás que **Git te avisa de que no puedes subirlo** porque "no estás al día", es decir no estás perfectamente sincronizado, hay arriba algo que tu no tienes (es ese fichero README.md). Incluye captura.

```
marmiqlis@INF1-03:/media/marmiqlis/Festplatte/repo/HelloGit$ git push origin main
Username for 'https://github.com': Marek8007
Password for 'https://Marek8007@github.com':
To https://github.com/Marek8007/HelloGit.git
 ! [rejected]        main -> main (fetch first)
error: falló el empuje de algunas referencias a 'https://github.com/Marek8007/HelloGit.git'
ayuda: Actualizaciones fueron rechazadas porque el remoto contiene trabajo que
ayuda: no existe localmente. Esto es causado usualmente por otro repositorio
ayuda: empujando a la misma ref. Quizás quieras integrar primero los cambios
ayuda: remotos (ej. 'git pull ...') antes de volver a hacer push.
ayuda: Mira 'Notes about fast-forwards' en 'git push --help' para detalles.
```

6. Para solucionar esto usa el comando **git fetch**, que te descarga solo el historial de cambios del remoto (no descarga los cambios realmente), así podrás comprobar con git tree lo que te falta para estar sincronizado, comprueba que hay un fichero README.md en remoto que tu no tienes en local. Incluye captura.

```
marmiqlis@INF1-03:/media/marmiqlis/Festplatte/repo/HelloGit$ git diff 5a15e05 c
78c73
diff --git a/README.md b/README.md
new file mode 100644
index 0000000..8d10833
--- /dev/null
+++ b/README.md
@@ -0,0 +1 @@
+# HelloGit
\ No newline at end of file
```

7. A continuación, vas a sincronizarte con el remoto, es decir vas a descargar historial y cambios usando el comando **git pull** o **git pull origin main.** Comprueba que ya tienes en local el fichero readme.md. Incluye captura.



Nota: el mecanismo básico para hacer un pull, es el merge. En caso de que no esté definido se configura con: git config pull.rebase.false

8. Ahora que estás completamente sincronizado, puedes hacer **git push** para subir lo que habías comiteado en local (el fichero hellogit.py). Comprueba que ha quedado subido el fichero hellogit.py con el cambio en su código. Incluye la captura.