IT Academy

by KIBERNUM

Programa: Desarrollo de Aplicaciones Front-End

-

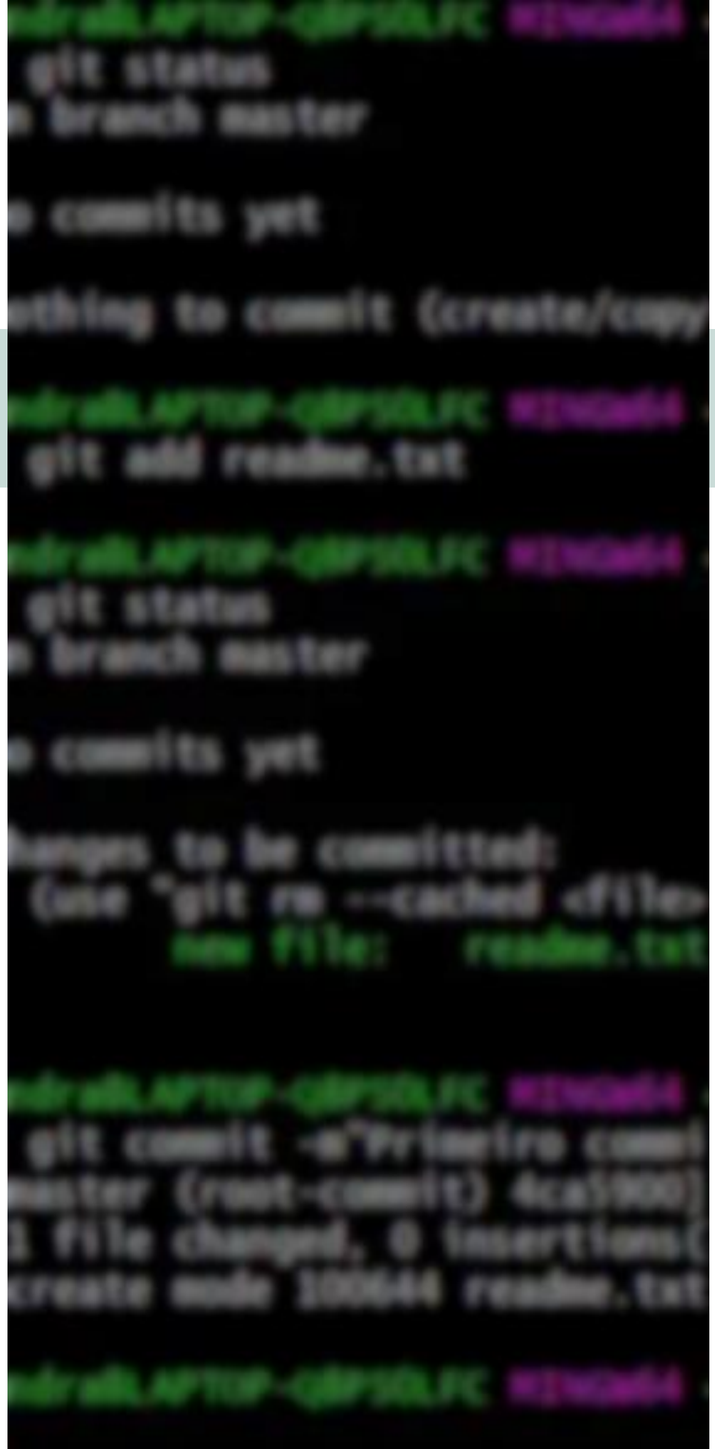
Módulo 2: Fundamentos de Desarrollo Front-End

Aprendizaje esperado

Gestionar el código fuente de un proyecto utilizando GitHub para mantener un repositorio de código remoto seguro y permitir trabajo concurrente.

Stash y Rebase en Git

En el flujo de trabajo con Git, a veces necesitamos gestionar cambios de manera más eficiente. Stash y Rebase son dos herramientas poderosas que nos permiten hacer precisamente eso.

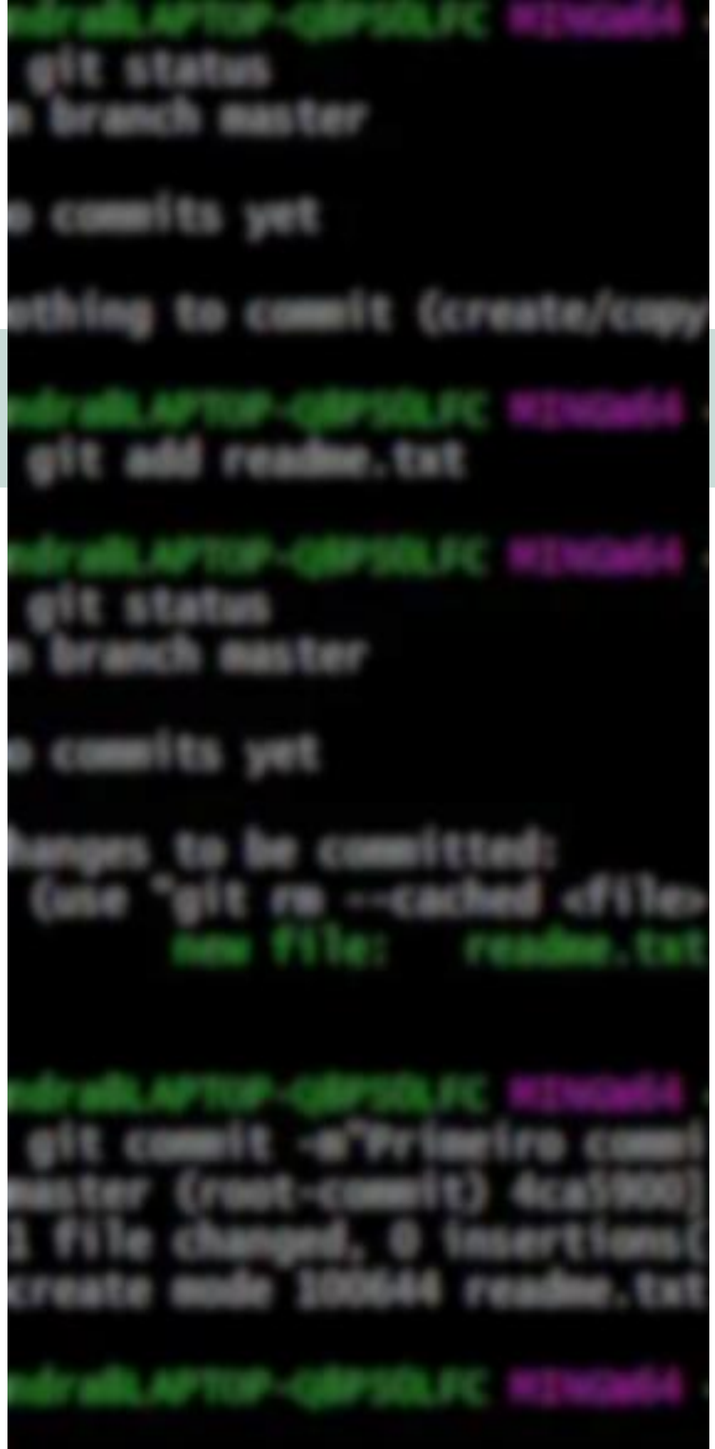


Stash: Guardando Cambios Temporales

- Si necesitas cambiar de contexto sin confirmar los cambios actuales, puedes usar git stash para guardar temporalmente los cambios no confirmados.
- Para guardar cambios en el stash:

```
@DESKTOP-Q7LG5E8 MINGW64 ~/Desktop
$ git stash
```

- Esto guarda los cambios y restaura el área de trabajo a su estado limpio.



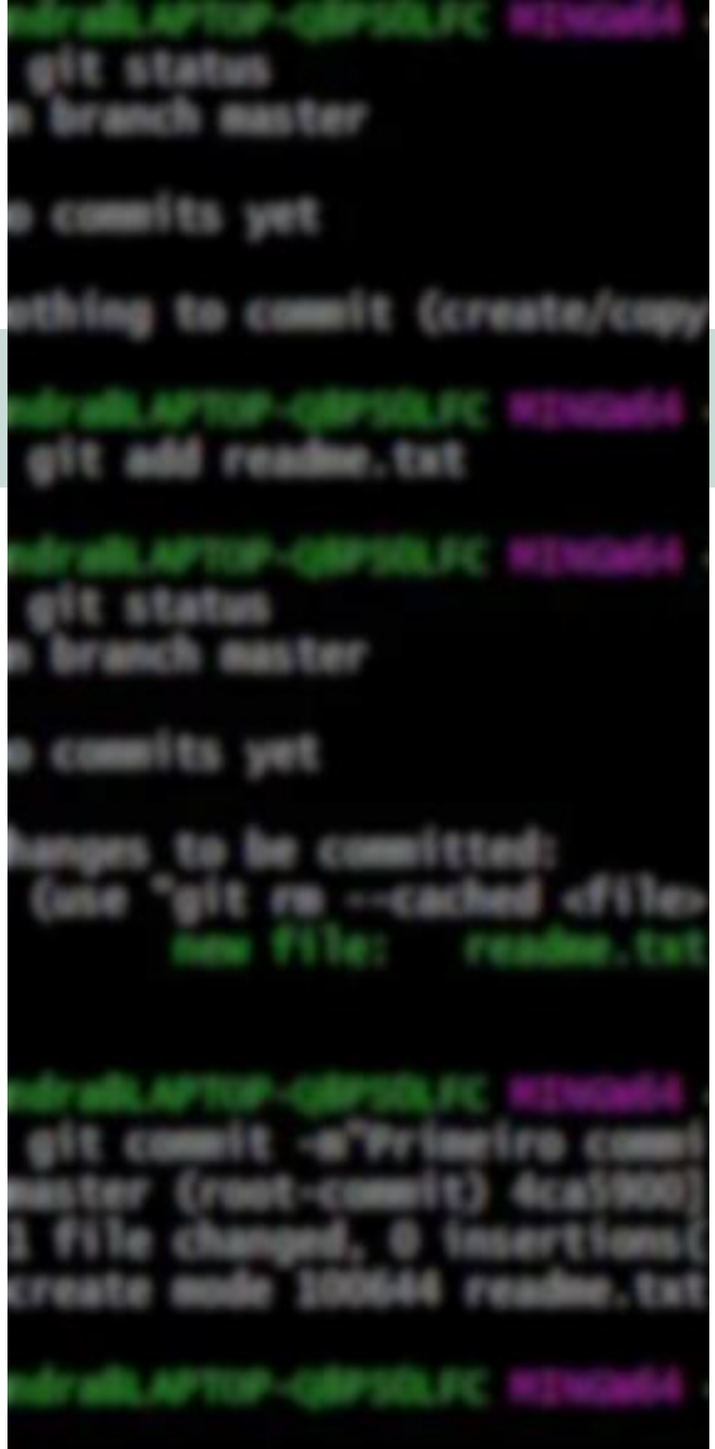
Stash – Opciones Adicionales

- Incluir archivos no rastreados:

```
@DESKTOP-Q7LG5E8 MINGW64 ~/Desktop
$ git stash --include-untracked
```

- Seleccionar partes específicas para guardar en el stash

```
@DESKTOP-Q7LG5E8 MINGW64 ~/Desktop
$ git stash -p
```



Stash – Opciones Adicionales

- Una vez que hayas terminado con la tarea temporal, puedes recuperar los cambios guardados con:

```
@DESKTOP-Q7LG5E8 MINGW64 ~/Desktop
$ git stash apply
```

- Si prefieres eliminar el stash después de aplicarlo:

```
@DESKTOP-Q7LG5E8 MINGW64 ~/Desktop
$ git stash pop
```

Rebase: Reorganizando la Historia del Proyecto

- Rebase permite reorganizar la historia del proyecto aplicando los commits de una rama sobre una nueva base.
- Para realizar un rebase:

```
@DESKTOP-Q7LG5E8 MINGW64 ~/Desktop
$ git rebase nueva_rama
```

Esto genera una historia de commits más lineal, sin los commits de fusión adicionales que se crean con merge.

Resolver Conflicto Durante un Rebase

- Si se encuentran conflictos durante el rebase, Git te pedirá que resuelvas los conflictos manualmente antes de continuar.
- Para continuar el proceso de rebase después de resolver los conflictos:

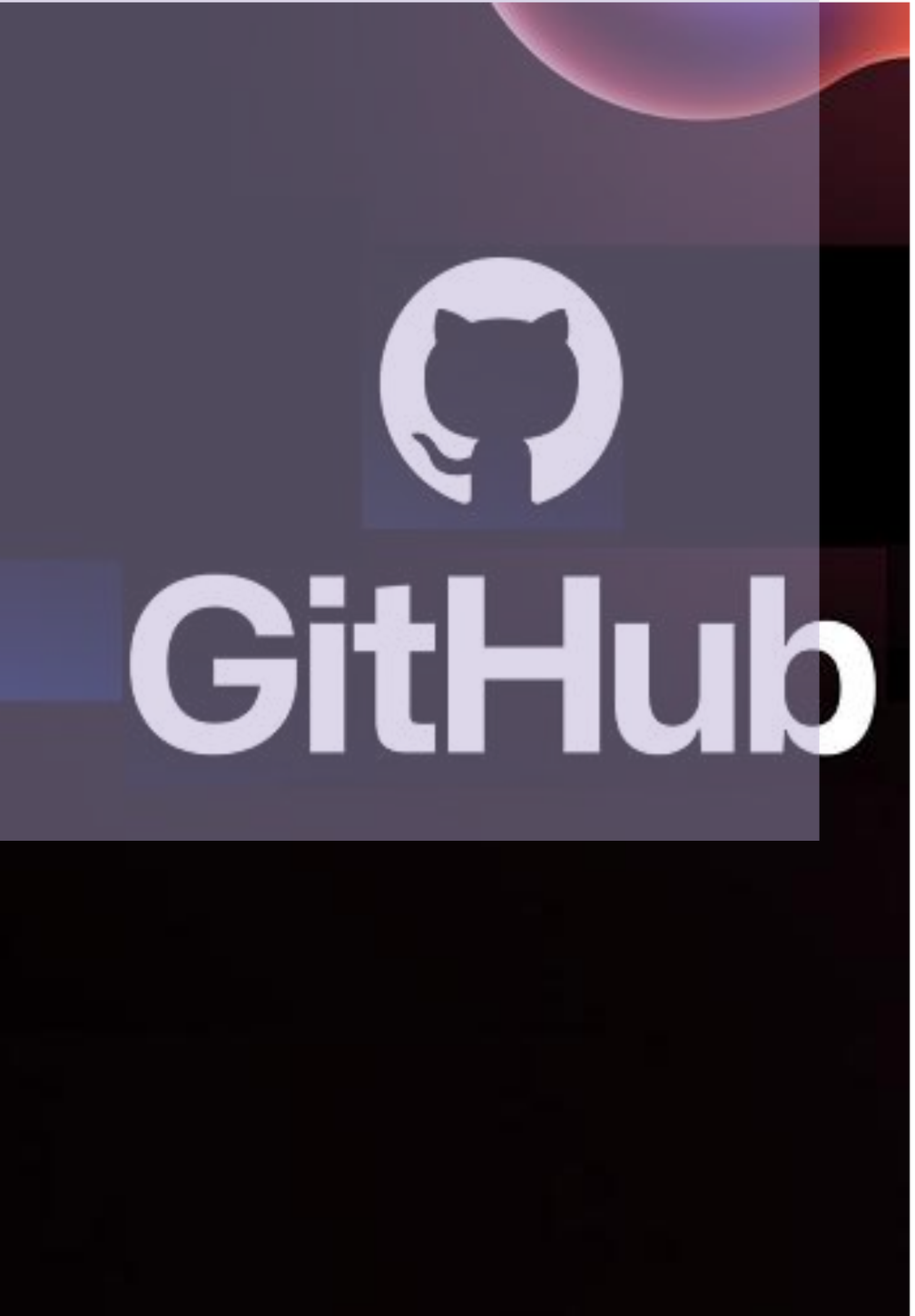
```
    @DESKTOP-Q7LG5E8 MINGW64 ~/Desktop
$ git rebase --continue
```

Abortar un Rebase

🔹 Si decides que no quieres continuar con el rebase, puedes abortar el proceso y restaurar el estado anterior con:

```
    @DESKTOP-Q7LG5E8 MINGW64 ~/Desktop
$ git rebase --abort
```

El uso de rebase debe ser considerado con cuidado, especialmente cuando trabajas en proyectos colaborativos, ya que puede reescribir el historial. No es recomendable hacer rebase sobre ramas que ya hayan sido compartidas con otros desarrolladores, ya que esto puede causar conflictos y confusión en el historial compartido.



GitHub: El centro del desarrollo Web

GitHub es mucho más que una plataforma de código; es un ecosistema completo que empodera a los desarrolladores y facilita la creación de software de alta calidad.

¿Qué es GitHub?

GitHub es una plataforma de alojamiento de código basada en la nube que utiliza Git, un sistema de control de versiones distribuido. Es como una red social para desarrolladores, donde los equipos pueden trabajar en proyectos juntos, rastrear cambios en el código y colaborar en tiempo real.

- **Control de versiones**

Imagínate como un artista que está pintando un cuadro. Con Git, puedes guardar cada cambio que haces en el lienzo (el código) como una nueva versión. Así, puedes volver a versiones anteriores, comparar cambios y ver cómo evolucionó el proyecto.

- **Colaboración en equipo**

Con las ramas (branches), cada miembro del equipo puede trabajar en su propia versión del proyecto sin afectar el código principal. Cuando están listos, pueden enviar sus cambios para que otros los revisen y aprueben.

Cómo GitHub facilita la colaboración

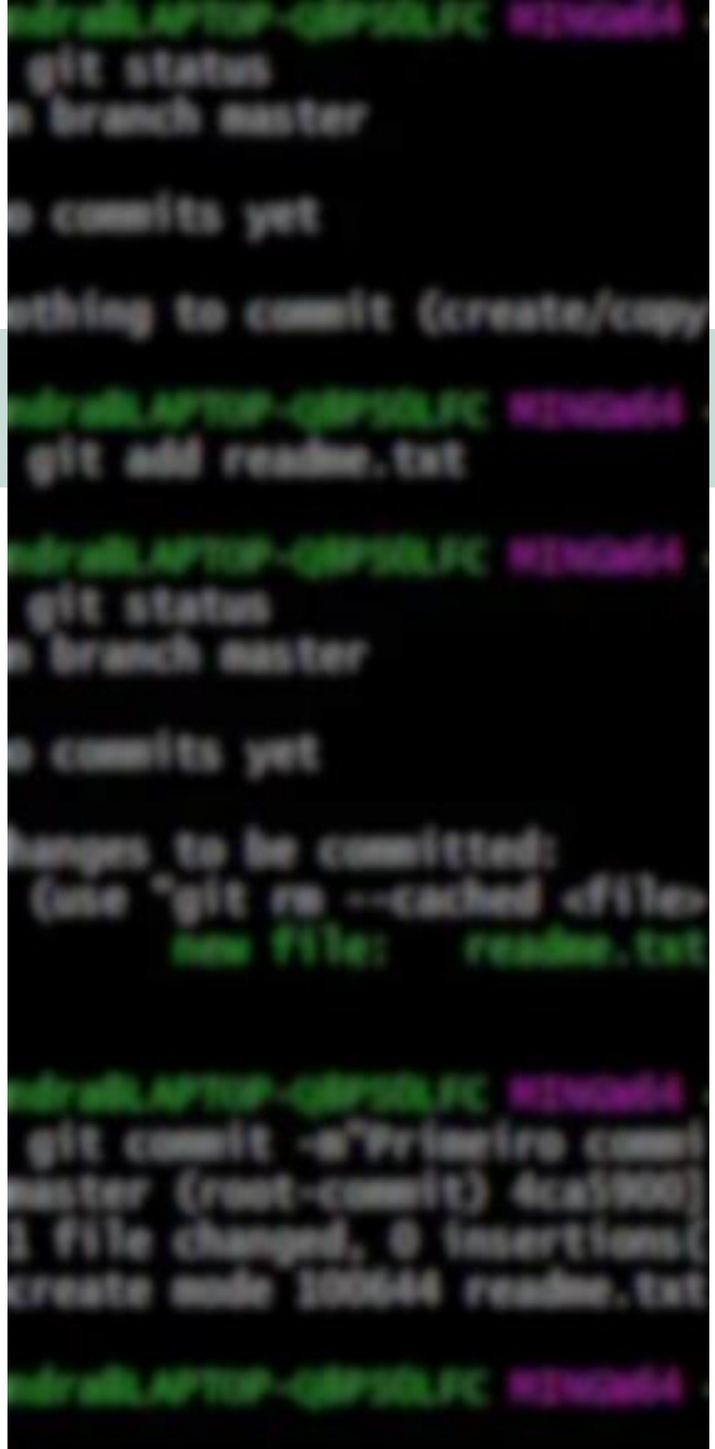
GitHub facilita la colaboración de diferentes maneras. A través de las ramas (branches) y las solicitudes de extracción (Pull requests), se crea un flujo de trabajo ordenado y eficiente.

- **Ramas (Branches)**

Imagina que quieres añadir una nueva funcionalidad a un proyecto. Puedes crear una rama (branch) independiente para trabajar en esa característica, sin afectar el código principal. Esto permite que varios desarrolladores trabajen en diferentes funciones al mismo tiempo.

- **Solicitudes de Extracción (Pull Requests)**

Una vez que una rama está lista, el desarrollador crea una solicitud de extracción (Pull request), que es como un pedido para que los cambios se fusionen con el código principal. Los demás miembros del equipo revisan el código, comentan y aprueban los cambios antes de que se incorporen al proyecto principal.



Documentación y comunicación clara

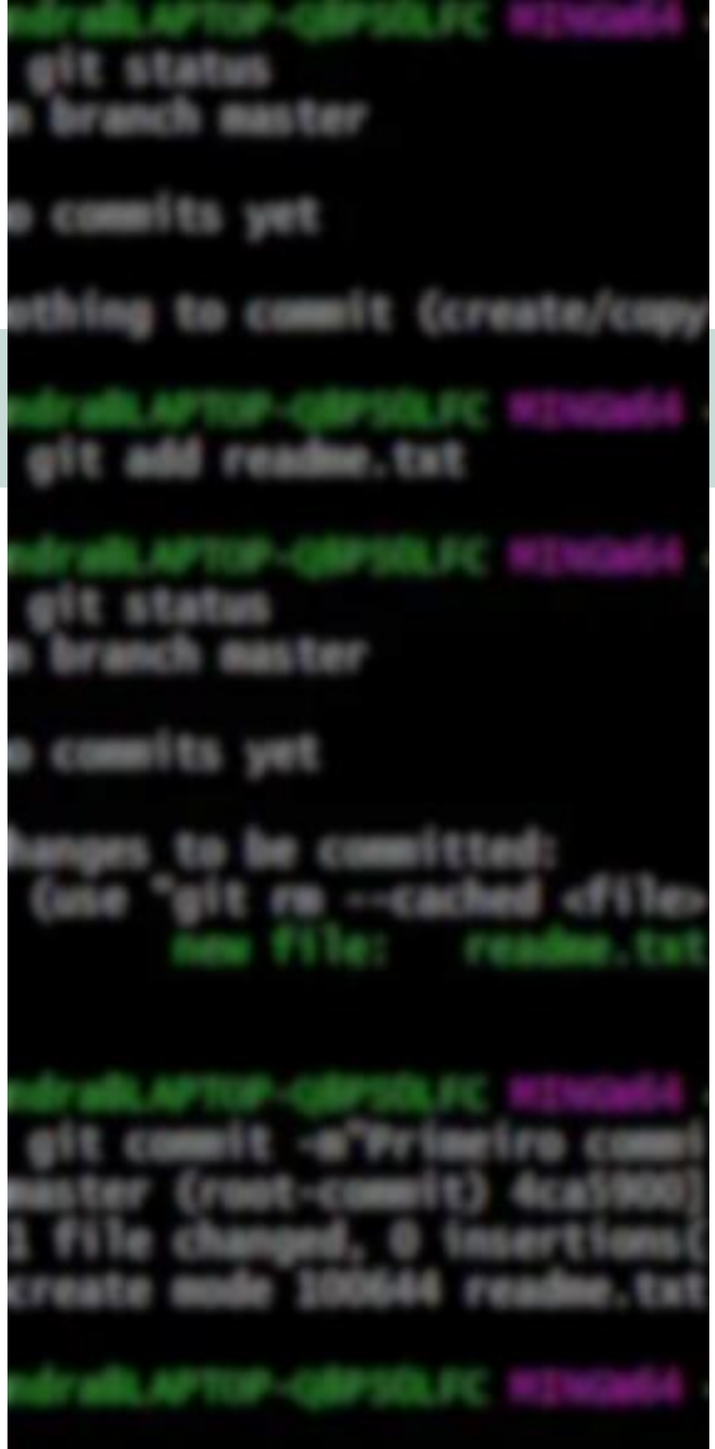
GitHub utiliza Markdown, un lenguaje sencillo para formatear texto, para crear documentación clara y concisa. El archivo README.md es esencial para explicar el proyecto, cómo instalarlo y cómo contribuir.

Markdown

Es un lenguaje fácil de aprender, que te permite añadir encabezados, listas, enlaces y código, todo sin escribir HTML complejo.

Comentarios y Discusiones

GitHub facilita la comunicación entre desarrolladores. Puedes comentar directamente en el código, en las solicitudes de extracción o en los archivos README.md para compartir ideas y resolver problemas.



Repositorios públicos y privados

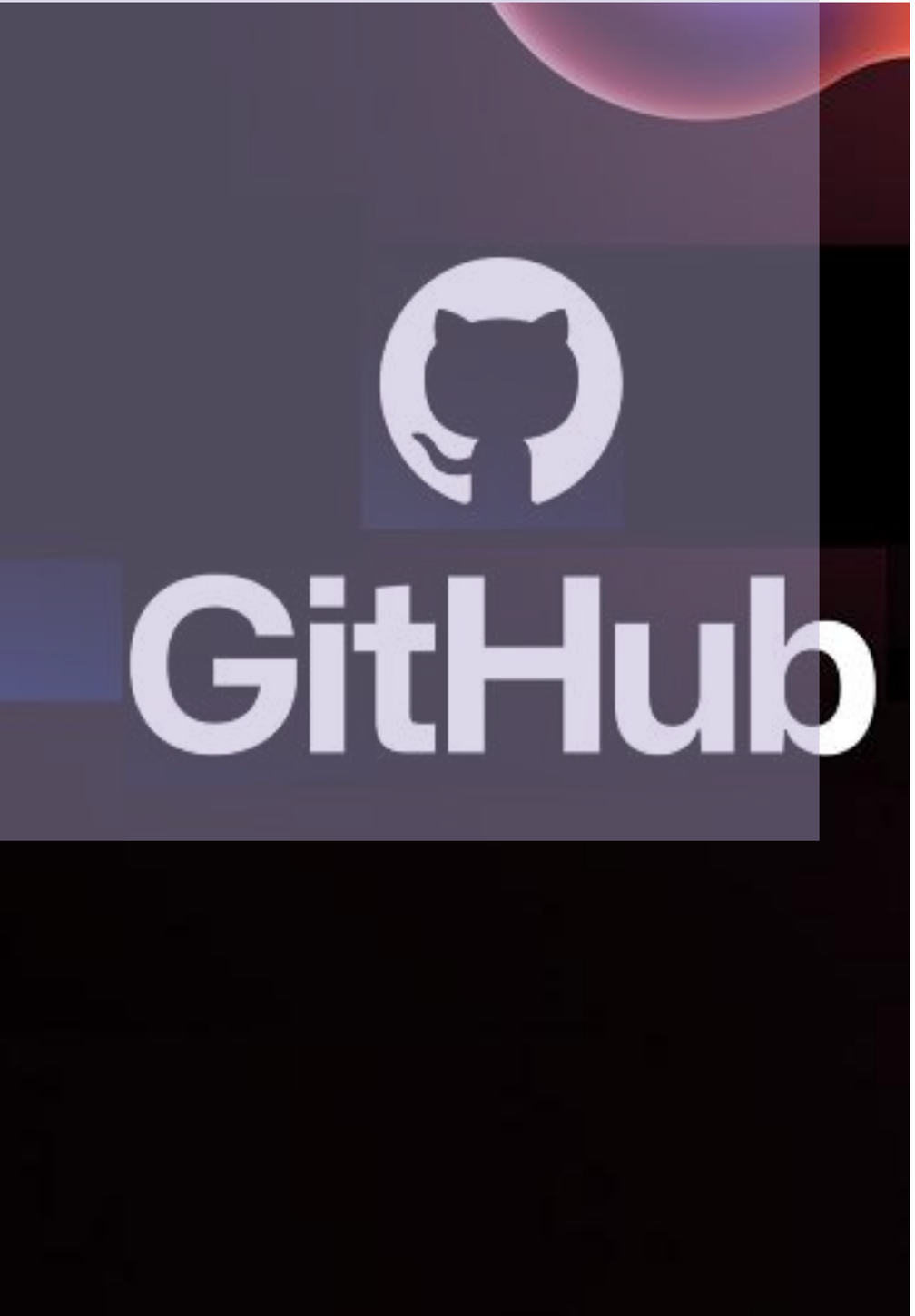
GitHub te permite crear dos tipos de repositorios:

- **Repositorios públicos**

Cualquier persona puede ver y contribuir al código. Ideal para proyectos de código abierto.

- **Repositorios privados**

Solo el equipo o los colaboradores autorizados pueden acceder al código. Perfecto para proyectos comerciales o confidenciales.

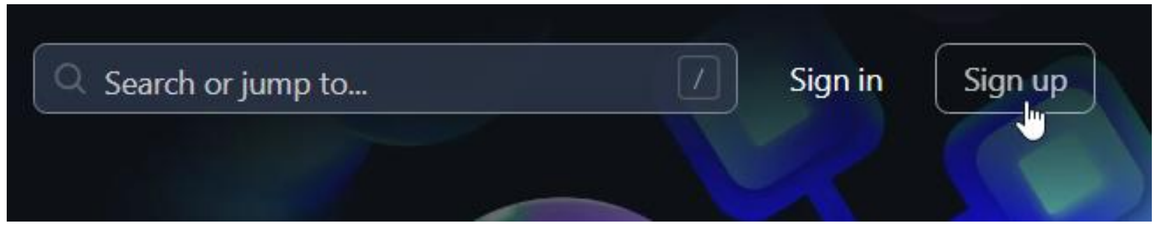


Creando una cuenta GitHub

Ahora aprenderás cómo crear una cuenta en GitHub, la plataforma líder para colaborar en proyectos Web.

Acceder al sitio web de GitHub

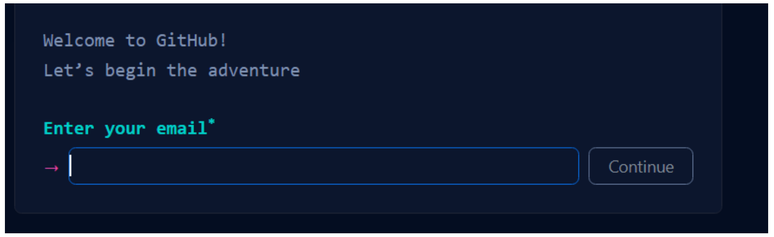
- Abre tu navegador y visita https://github.com.
- Haz clic en el botón Sign up (Registrarse) en la página principal.



Nota: Asegúrate de estar en el sitio oficial para evitar fraudes.

Registrarse en GitHub

- Ingresa tu correo electrónico y haz clic en Continue.



- Ingresa una contraseña segura y haz clic en Continue.

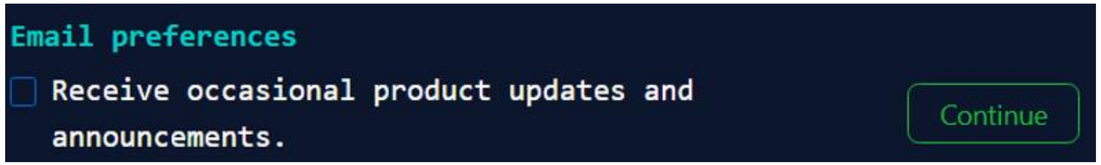
Consejo: Usa una combinación de letras, números y símbolos para mayor seguridad.

Crear Nombre de Usuario

- Elige un nombre de usuario único y haz clic en Continue.

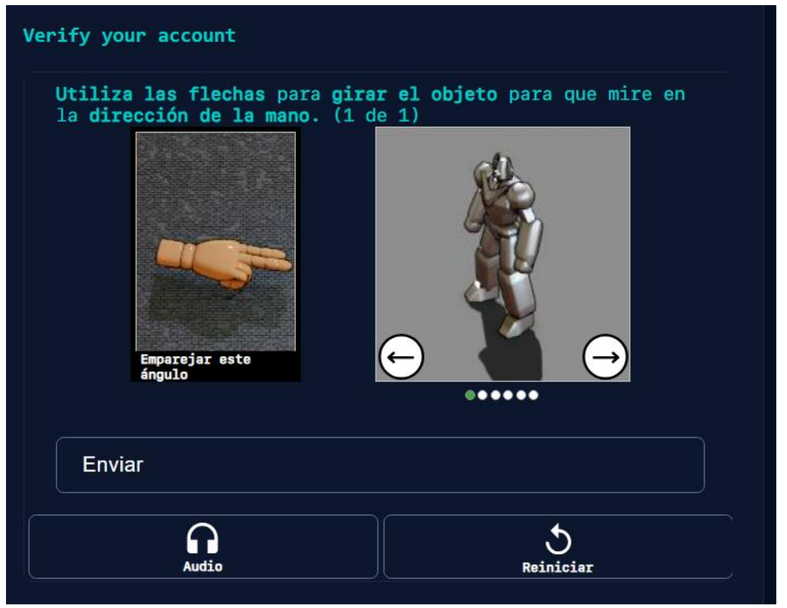
Nota: Este nombre será tu identificación en GitHub, así que elige algo que te represente.

- Selecciona Yes o No para recibir noticias y actualizaciones de GitHub, luego haz clic en Continue.



Verificación de Seguridad

- Completa el captcha para verificar que no eres un robot.



Nota: Este paso es esencial para proteger la plataforma de cuentas falsas.

Últimas Configuraciones

- Selecciona Free para usar la versión gratuita de GitHub.
- Revisa tu bandeja de entrada, abre el correo de GitHub y sigue el enlace para verificar tu cuenta.
- Configura tus preferencias iniciales o salta este paso si lo prefieres.

Consejo: Si no ves el correo, revisa la carpeta de spam.

Notas:

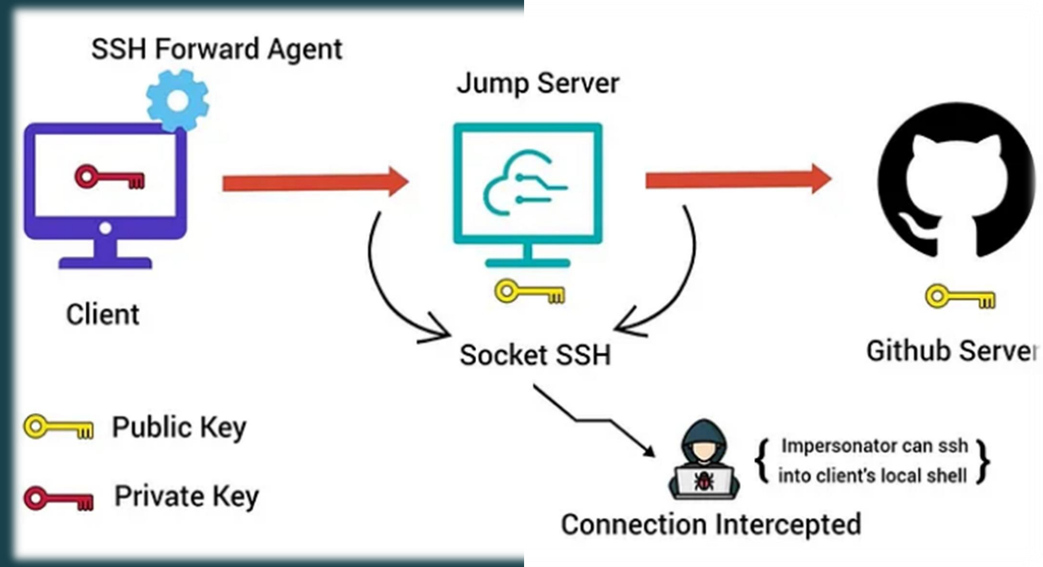
✅ La versión gratuita es suficiente para la mayoría de los usuarios, pero puedes actualizar a una cuenta de pago si necesitas más funcionalidades.

✅ Puedes personalizar tu experiencia más adelante en la configuración de tu perfil.

Configuración de Claves SSH para Autenticación Segura en GitHub

Veremos el paso a paso sobre la configuración de las claves SSH para mejorar la seguridad de tu cuenta de GitHub. Aprenderás cómo generar una clave SSH, agregarla a tu cuenta de GitHub y autenticarte de manera segura utilizando tu nueva clave.

Entendiendo Claves SSH



1

Clave pública

La clave pública es una parte esencial de la autenticación SSH que se comparte con los servidores o servicios a los que deseas acceder. Se almacena en el servidor para verificar tu identidad al conectar con tu clave privada.

2

Clave privada

La clave privada es la parte más importante y debe mantenerse en secreto. Es la clave que utilizas para iniciar sesión en servicios como GitHub y se guarda en tu equipo local.

Generación de una Clave SSH

1

Las claves SSH se generan utilizando herramientas de línea de comandos como ssh-keygen en terminales Unix (Linux o MacOS) o en Git Bash para Windows.

```
    @DESKTOP-Q7LG5E8 MINGW64 ~/Desktop
$ ssh-keygen -t ed25519 -C "tu-email@ejemplo.com"
```

Explicación:

- -t ed25519: Especifica el algoritmo Ed25519 (recomendado).
- -C: Añade una etiqueta (generalmente tu correo).

Elegir Ubicación y Contraseña

2

- Confirma la ubicación donde se guardará la clave. Por defecto, se guardará en el directorio ~/.ssh/ bajo el nombre id_ed25519.

- Ingresa una contraseña para la clave (opcional pero recomendado). Esta contraseña protegerá tu clave privada en caso de que alguien obtenga acceso a tu archivo de clave.

- Las dos claves generadas son: id_ed25519 (clave privada, mantenerla en secreto) y id_ed25519.pub (clave pública, se compartirá con GitHub).

Agregar la Clave Pública a GitHub

1

Ejecuta el comando a continuación para mostrar el contenido de la clave pública SSH almacenada en el archivo id:ed25519

```
@DESKTOP-Q7LG5E8 MINGW64 ~/Desktop
$ cat ~/.ssh/id_ed25519.pub
```

2

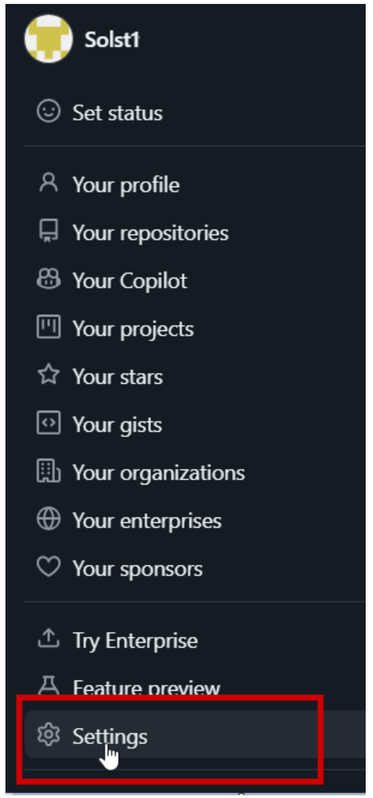
Copia el contenido completo de la clave pública desde tu terminal. La clave pública comienza con ssh-ed25519 y contiene una larga cadena de caracteres.

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIK3C4... your_email@example.com
```

Agregar la Clave Pública a GitHub

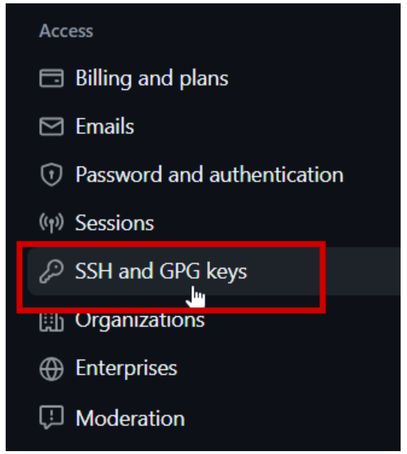
3

Ve a Settings en GitHub.

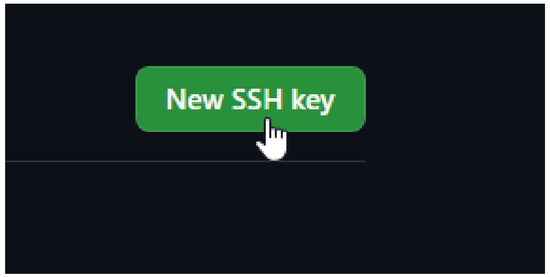


Agregar la Clave Pública a GitHub

4 Selecciona SSH and GPG keys



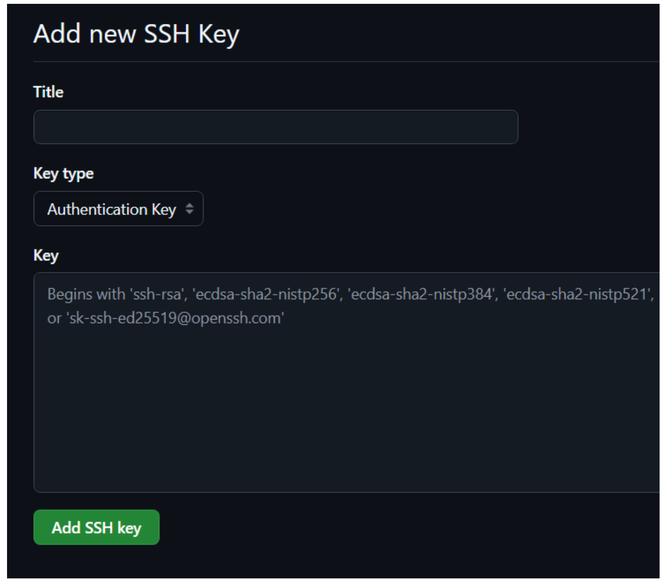
5 Haz clic en New SSH key.

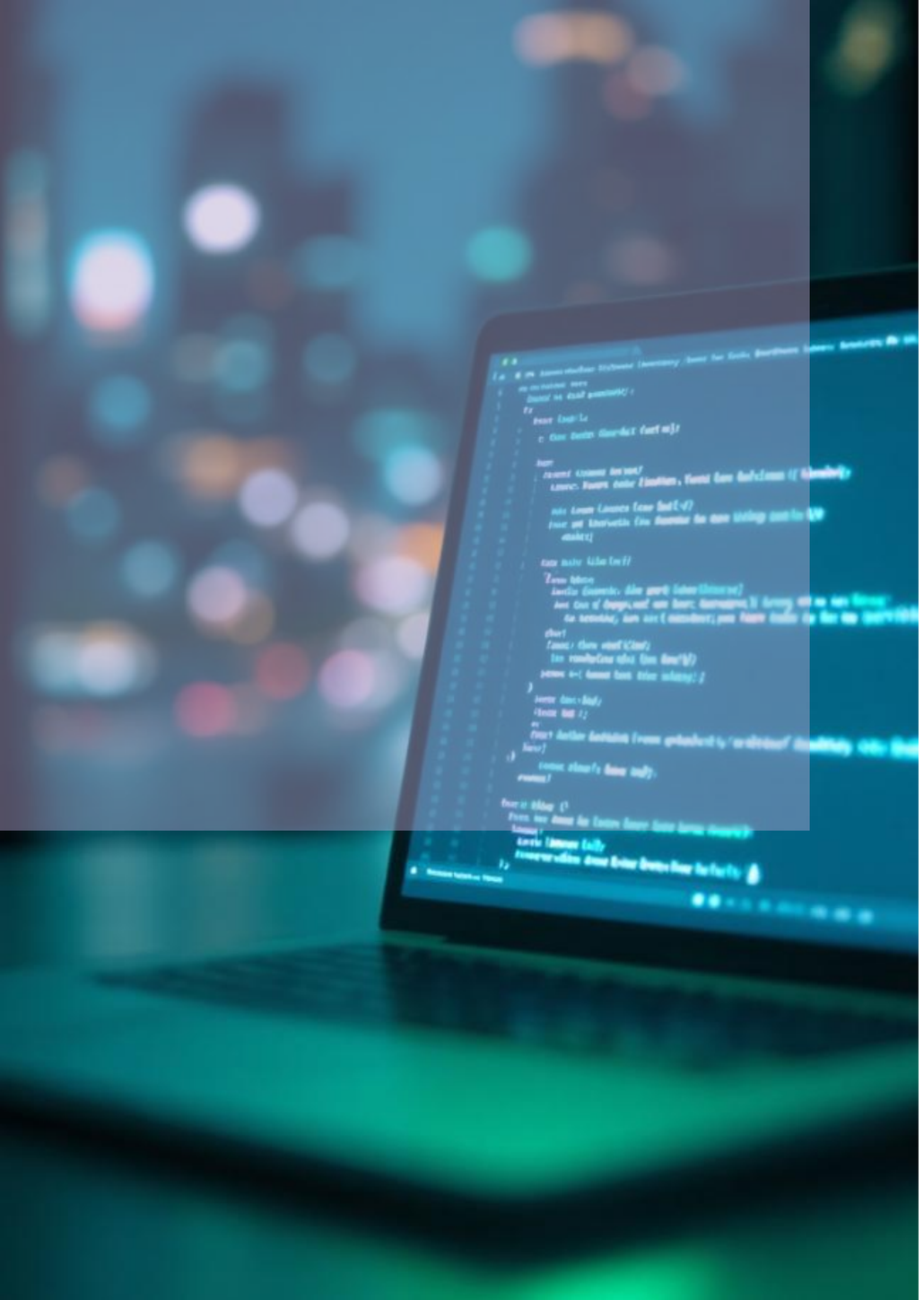


Agregar la Clave Pública a GitHub

6

Pega la clave pública copiada en el campo correspondiente y agrega un título. Puedes usar un nombre descriptivo como "Mi clave SSH personal". Una vez agregada la clave, puedes autenticarte en GitHub usando SSH.





Repositorios Remoto

Los repositorios remotos son esenciales para la colaboración efectiva en proyectos de desarrollo web. Descubriremos cómo funcionan, qué beneficios ofrecen y cuáles son las opciones disponibles.

Sincronización: Manteniendo el Código Actualizado

- **Push**

Envía los cambios del repositorio local al remoto. Es como subir tu trabajo a la nube, compartiendo tus modificaciones con el equipo.

- **Pull**

Descarga las actualizaciones del repositorio remoto al local. Es como actualizar tu trabajo local con los cambios que realizaron otros desarrolladores.

- **Fetch**

Descarga los cambios del repositorio remoto, pero sin aplicarlos automáticamente a tu código local. Te permite revisar los cambios antes de integrarlas en tu trabajo.

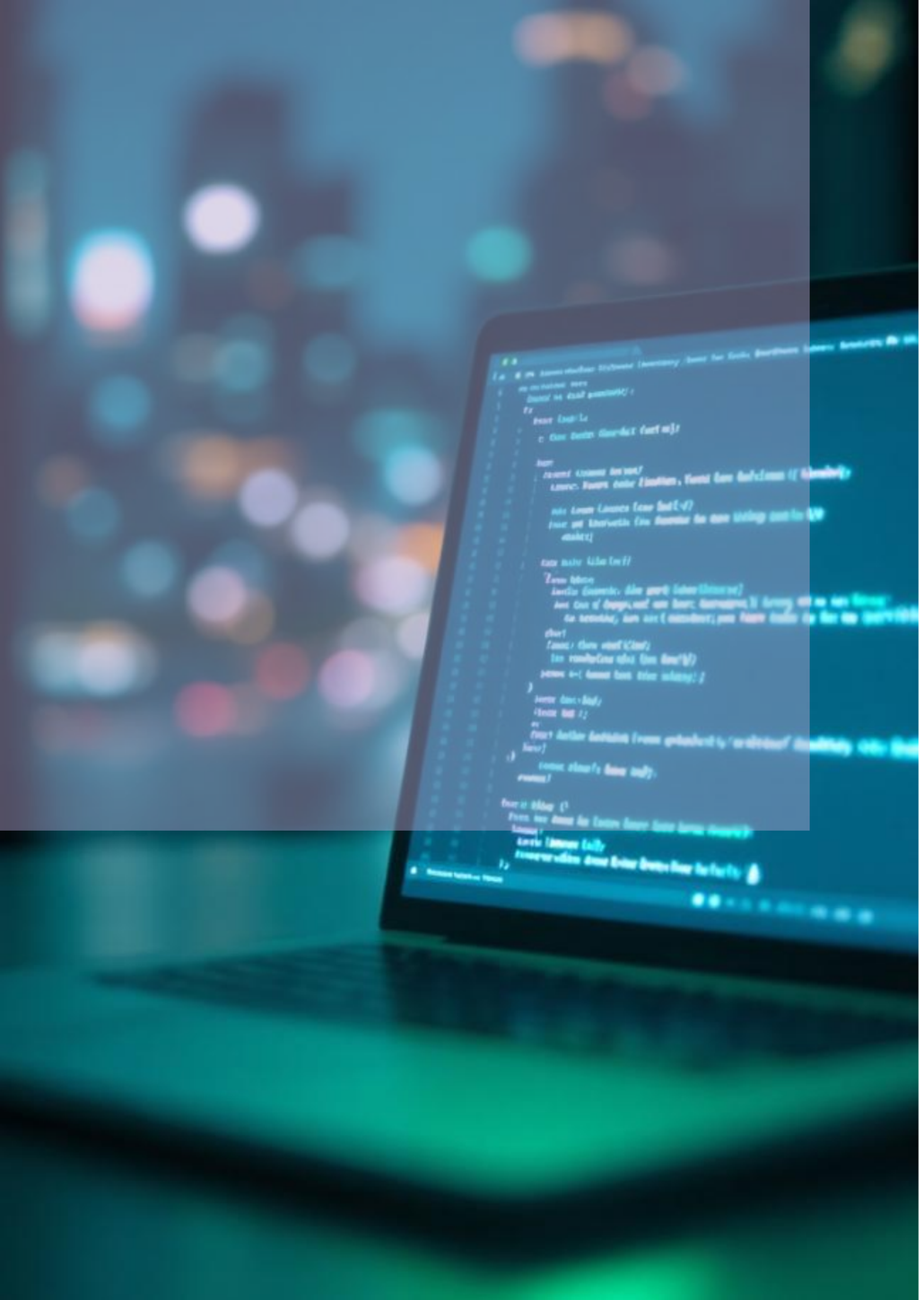
Acciones Clave en un Repositorio Remoto

- **Colaboración en Equipo**

Permite que varios desarrolladores trabajen simultáneamente en un mismo proyecto, cada uno desde su máquina, y sincronizar sus cambios mediante el repositorio remoto.

- **Clonar un Repositorio Remoto**

Crear una copia del repositorio remoto en tu máquina local, para empezar a trabajar con el proyecto.



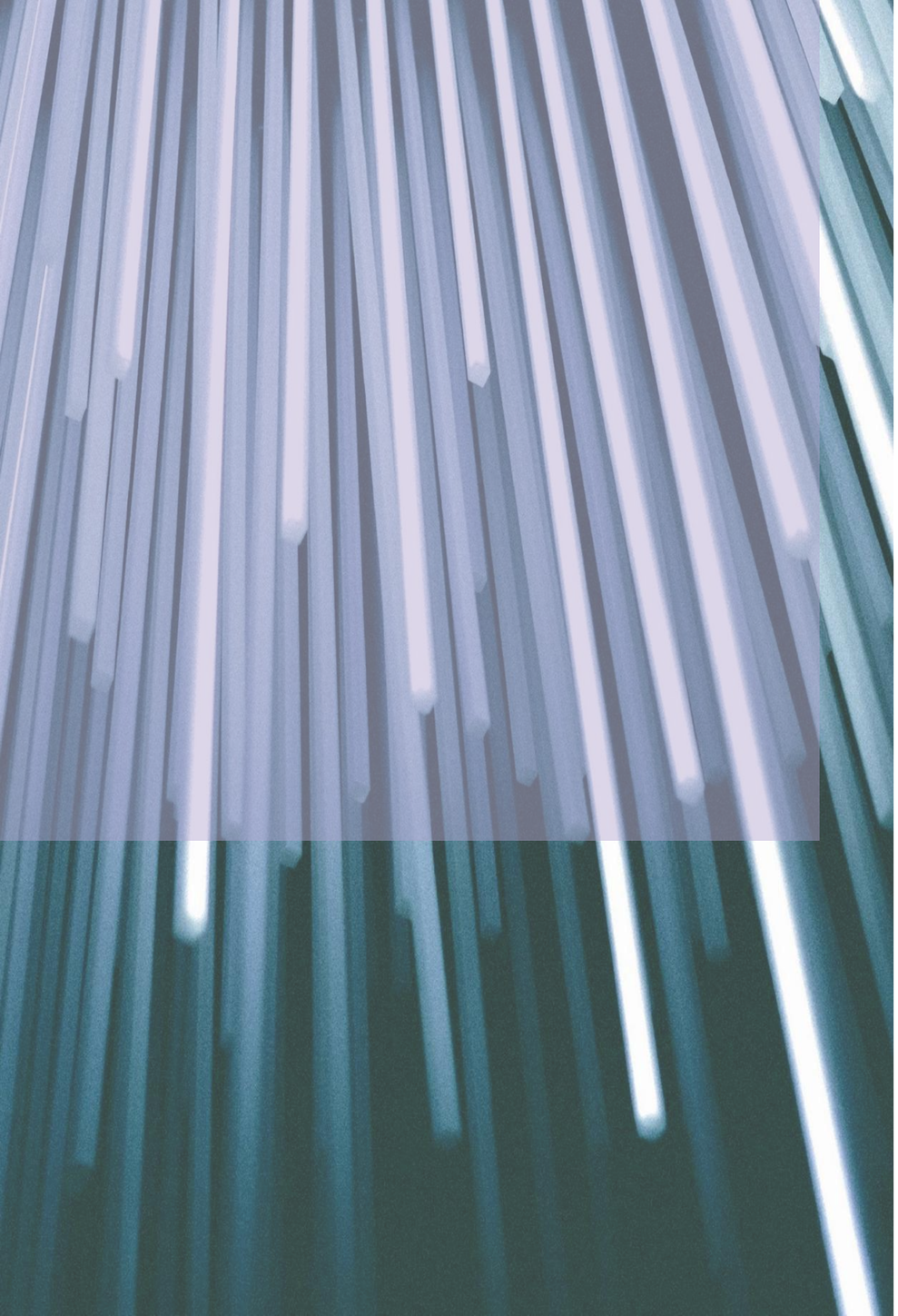
Más Allá de GitHub: Explorando Otras Opciones

GitLab GitLab

Otro servicio basado en Git que proporciona control de versiones y herramientas colaborativas. Ofrecen características adicionales como integración continua y DevOps.

BitBucket Bitbucket

Ofrece control de versiones para Mercurial y Git, y es popular por su integración con otros productos de Atlassian como Jira.



Push y Pull en GitHub: Colaboración Eficaz

Las operaciones push y pull son fundamentales para la gestión de proyectos en GitHub, esenciales para el flujo de trabajo colaborativo en el desarrollo web. Permiten enviar y descargar cambios entre repositorios locales y remotos, facilitando la colaboración y el control de versiones.

Operación Push: Compartiendo tus Cambios

- **¿Qué es Push?**

Permite enviar los cambios realizados en tu repositorio local a un repositorio remoto en GitHub, actualizando la versión del código almacenada en GitHub.

- **Importancia**

Asegura que otros colaboradores puedan acceder a tus cambios. Es crucial verificar que el código esté en un estado estable antes de realizar un push.

- **Comando**

Se utiliza el comando git push, seguido del nombre del repositorio remoto y la rama a actualizar. Por ejemplo: git push origin main.

Ejemplo:

📌 Escenario: Has realizado cambios en index.html y quieres compartirlos con tu equipo.

💻 Comandos a utilizar:

```
    @DESKTOP-Q7LG5E8 MINGW64 ~/Desktop
$ git add index.html
git commit -m "Actualiza el diseño de la página de inicio"
git push origin main
```

Operación Pull: Manteniéndote Actualizado

- **¿Qué es Pull?**

Se utiliza para descargar las actualizaciones del repositorio remoto a tu máquina local, fusionando los cambios realizados por otros colaboradores.

- **Importancia**

Asegura que tu trabajo esté alineado con las últimas modificaciones del proyecto y ayuda a evitar conflictos de fusión.

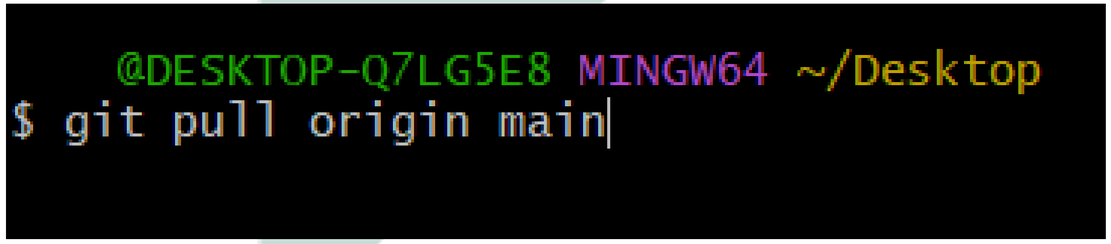
- **Comando**

El comando git pull descarga las actualizaciones de la rama del repositorio remoto y las fusiona con tu versión local. Por ejemplo: git pull origin main.

Ejemplo:

📌 Escenario: Tu compañero de equipo hizo cambios en index.html y realizó un push. Antes de continuar trabajando, necesitas traer esos cambios a tu máquina local.

💻 Comando a utilizar:



Comparativa entre PUSH y PULL

Operación	Función	Propósito
Push	Enviar cambios al repositorio remoto	Hacer que tus cambios estén disponibles para otros colaboradores.
Pull	Obtener y fusionar cambios del repositorio remoto	Asegurar que tu repositorio local esté actualizado con la versión más reciente del código.

Fetch vs. Pull: ¿Cuál Elegir?

- **Fetch**

Descarga las actualizaciones del repositorio remoto a tu repositorio local, pero no modifica tu copia de trabajo actual.

```
@DESKTOP-Q7LG5E8 MINGW64 ~/Desktop
$ git fetch origin
```

- **Pull**

Descarga las actualizaciones del repositorio remoto y las fusiona con tu rama activa en un solo paso.

```
@DESKTOP-Q7LG5E8 MINGW64 ~/Desktop
$ git pull origin main
```

Cuándo Usar Fetch

- **Control Granular**

Si deseas tener un control más granular sobre los cambios.

- **Revisión de Modificaciones**

Si es importante revisar las modificaciones antes de fusionarlas con tu trabajo.

- **Comandos Adicionales**

Después de un fetch, puedes usar git log o git diff para examinar los cambios.

Cuándo Usar Pull

- **Integración Directa**

Cuando estés listo para integrar cambios en tu código de manera directa.

- **Entorno Ágil**

Útil en situaciones donde se trabaja en un entorno ágil.

- **Alineación Constante**

Si deseas mantener tu código alineado con las contribuciones más recientes..

IT Academy

by KIBERNUM