

FACUNDO UFERER

GIT



PARA PRINCIPIANTES

Contenidos

¿Qué es GIT?	3
¿Para qué se usa GIT?	3
Los Repositorios	4
Repositorio local	4
Repositorio remoto	4
Las partes de GIT	5
Los tres componentes principales:	5
La arquitectura de GIT	5
Work directory	5
INDEX	6
HEAD	6
Remote Repo	7
Instalar GIT en nuestro sistema	7
Windows	7
Descargue e instale Git	7
Linux	7
¿Cómo usar GIT?	8
Configurar Git	8
Crear un repositorio	8
Actualizar el repositorio	9
git add	9
git commit	9
git push	9
git fetch	10
git pull	10
Ramas	10
git branch	11
git checkout	11
git merge	11
git clone	11
git status	13
Los elementos que muestra el comando "git status":	13
Branch	13
Working Tree	13

Staging Area	13
Untracked Files	14
Cómo usar el comando "git status"	14
GitHub	15
Login con Token	15
Generar un token de acceso personal en GitHub.	15
Configurar Git en tu ordenador para utilizar el token.	15

GIT

¿Qué es GIT?

GIT es un sistema de control de versiones distribuido utilizado para rastrear los cambios en el código fuente y coordinar el trabajo en equipo en proyectos de software. Fue creado por Linus Torvalds en 2005 y se ha convertido en uno de los sistemas de control de versiones más populares del mundo.

GIT permite a los desarrolladores trabajar en el mismo código fuente simultáneamente, manteniendo un historial completo de todos los cambios realizados en el código a lo largo del tiempo. Cada vez que un desarrollador hace cambios en el código, estos cambios se registran y se pueden guardar en una "rama" separada, lo que permite a otros desarrolladores trabajar en diferentes partes del proyecto sin interferir entre sí.

Además, GIT ofrece una serie de herramientas útiles para trabajar con proyectos de software, como la capacidad de revertir cambios, fusionar diferentes ramas de código y colaborar con otros desarrolladores de manera eficiente. Esto hace que GIT sea una herramienta esencial para el desarrollo de software en equipo.

¿Para qué se usa GIT?

GIT se utiliza para administrar el control de versiones de proyectos de software. Los desarrolladores lo usan para mantener un registro completo de todos los cambios que se hacen en el código fuente de un proyecto y para colaborar con otros desarrolladores en el mismo proyecto. Algunas de las tareas que se pueden realizar con GIT son:

- **Control de versiones:** GIT permite a los desarrolladores mantener un historial completo de todos los cambios realizados en el código fuente, lo que facilita la colaboración y la gestión de diferentes versiones del mismo proyecto.
- **Colaboración en equipo:** GIT permite a los desarrolladores trabajar en diferentes partes del mismo proyecto al mismo tiempo, lo que ayuda a mejorar la eficiencia del equipo de desarrollo.

- **Gestión de ramas:** Los desarrolladores pueden crear diferentes ramas de un proyecto en GIT para trabajar en diferentes características del mismo proyecto sin interferir con el trabajo de otros desarrolladores.
- **Control de calidad:** GIT permite a los desarrolladores realizar pruebas y experimentos en diferentes ramas de un proyecto antes de incorporar los cambios en la rama principal.
- **Despliegue de software:** Los desarrolladores pueden utilizar GIT para gestionar el despliegue de software en diferentes entornos, como producción, pruebas y desarrollo.

Los Repositorios

En GIT, un repositorio es una base de datos que almacena todos los archivos y carpetas de un proyecto de software, junto con el historial completo de cambios realizados en el código fuente a lo largo del tiempo. Los repositorios se pueden dividir en dos tipos principales: repositorios locales y repositorios remotos.

Repositorio local

Un repositorio local es una copia completa del repositorio en la computadora local de un desarrollador. Los desarrolladores pueden clonar un repositorio remoto para crear una copia local en su propia computadora utilizando el comando "git clone". Una vez que se ha clonado el repositorio, los desarrolladores pueden trabajar en el código fuente localmente, realizar cambios y registrarlos en el repositorio local utilizando el comando "git commit". Los cambios realizados en el repositorio local solo están disponibles para ese desarrollador y no se pueden compartir con otros desarrolladores hasta que se suban al repositorio remoto.

Repositorio remoto

Un repositorio remoto es una copia completa del repositorio que se encuentra en un servidor remoto en línea, como GitHub, GitLab o Bitbucket. Los repositorios remotos permiten a los desarrolladores compartir el código fuente y colaborar en un proyecto de software en tiempo real. Los desarrolladores pueden subir sus cambios locales al repositorio remoto utilizando el comando "git push". Esto sincronizará los cambios realizados en el repositorio local con el repositorio remoto y permitirá a otros desarrolladores acceder a los cambios.

Las partes de GIT

Los tres componentes principales:

- **El repositorio de GIT:** es donde se almacenan los archivos de código fuente y su historial de cambios. Este repositorio es una base de datos que contiene todos los cambios realizados en los archivos desde que se inició el repositorio.
- **La línea de comandos de GIT:** es la interfaz de usuario principal para interactuar con GIT. Se utiliza para realizar acciones como agregar archivos al repositorio, hacer confirmaciones, crear ramas, fusionar cambios y otras tareas.
- **El cliente de GIT:** es el software que se utiliza para interactuar con un servidor remoto de GIT. Es utilizado para subir y descargar cambios del repositorio remoto. El cliente de GIT también proporciona herramientas para gestionar el control de acceso y la seguridad de los repositorios remotos.

Estos tres componentes trabajan juntos para proporcionar una herramienta completa de control de versiones y gestión de código fuente.

La arquitectura de GIT



Work directory

El "Work directory" (también conocido como "directorio de trabajo" o "directorio de trabajo actual") es la copia local de los archivos del proyecto que está trabajando un desarrollador. En otras palabras, es la carpeta en su computadora que contiene los archivos y directorios del proyecto.

El "Work directory" es la ubicación en la que los desarrolladores realizan cambios y modificaciones en los archivos del proyecto. Cada vez que se realiza una modificación en un archivo, el estado del archivo cambia de "no modificado" a "modificado" y se refleja en el directorio de trabajo.

Una vez que se han realizado los cambios necesarios, el desarrollador utiliza el comando "git add" para agregar los archivos modificados al "Index" o "staging area" de GIT. Luego, cuando el desarrollador está satisfecho con los cambios y está listo para confirmarlos en el historial del repositorio de GIT, utiliza el comando "git commit" para crear una nueva confirmación.

En resumen, el "Work directory" es el lugar donde los desarrolladores realizan cambios y trabajan en los archivos del proyecto antes de confirmarlos en el historial del repositorio de GIT. Es la copia local del proyecto y el lugar donde se realizan la mayoría de las operaciones de edición de archivos y desarrollo de código.

INDEX

El "INDEX" (también conocido como "staging area" o "área de preparación") en GIT es un área intermedia entre el directorio de trabajo y el repositorio de GIT. Es el lugar donde se preparan los archivos antes de ser confirmados mediante un "commit" en el repositorio.

En otras palabras, cuando se realiza un "git add" en un archivo, este archivo se agrega al "INDEX" o "staging area". Allí, se pueden realizar cambios adicionales en los archivos agregados, como agregar más cambios o eliminar cambios no deseados, antes de confirmarlos con un "git commit". Esto permite seleccionar cuidadosamente los cambios que se desean confirmar y, por lo tanto, evitar la confirmación de cambios no deseados.

El "INDEX" o "staging area" es una herramienta muy útil para trabajar en equipo, ya que permite a los desarrolladores revisar los cambios y resolver posibles conflictos antes de confirmarlos en el repositorio compartido. También facilita la tarea de mantener un historial de versiones ordenado y coherente, ya que permite controlar con precisión qué cambios se confirman en cada commit.

HEAD

En GIT, el término "HEAD" se refiere al puntero que indica la última confirmación en una rama determinada del repositorio. En otras palabras, "HEAD" es el estado actual del repositorio en el que se encuentra el usuario.

Cuando se realiza una confirmación en GIT, el puntero "HEAD" se mueve hacia adelante, apuntando a la confirmación más reciente. Esto significa que "HEAD" siempre apunta a la confirmación más reciente de la rama actual en la que se está trabajando.

El "HEAD" puede moverse hacia atrás y hacia adelante en el historial de confirmaciones del repositorio mediante el uso de los comandos "git checkout" o "git reset". Por ejemplo, "git checkout" se utiliza para cambiar a una rama diferente y, por lo tanto, mover el puntero "HEAD" a la última confirmación de esa rama.

Además, el "HEAD" también puede utilizarse como referencia para comparar el estado actual del repositorio con otro punto en el historial de confirmaciones. Por ejemplo, se puede utilizar el comando "git diff HEAD~1" para mostrar las diferencias entre el estado actual del repositorio y el anterior al último commit.

Remote Repo

Es un repositorio GIT que se encuentra en un servidor remoto y que se utiliza para permitir la colaboración y el intercambio de código entre desarrolladores y equipos en diferentes lugares geográficos. Es una copia completa del historial de cambios del proyecto y se puede actualizar y descargar utilizando los comandos "git push" y "git pull".

Instalar GIT en nuestro sistema

Windows

Para instalar GitHub en una computadora con Windows, siga los siguientes pasos:

Descargue e instale Git

Lo primero que debe hacer es descargar e instalar Git en su computadora. Puede descargar Git desde la página oficial de Git en <https://git-scm.com/download/win>.

Linux

Git suele venir por defecto en los repositorios oficiales de la mayoría de distribuciones, por lo que podrás instalarlo directamente utilizando el gestor de paquetes de tu distribución.

Si trabajas con Debian, Ubuntu, o cualquier otra distribución derivado, puedes instalarlo con el siguiente comando:

```
sudo apt install git
```

¿Cómo usar GIT?

Una vez que tenemos GIT instalado en nuestro sistema operativo podremos realizar las siguientes operaciones.

Configurar Git

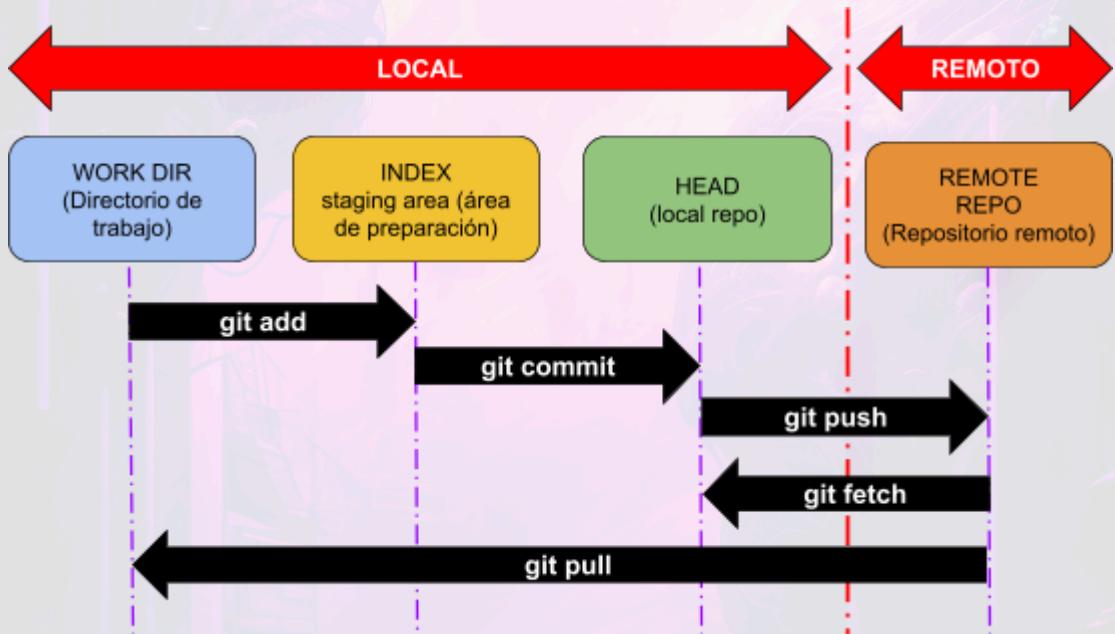
Una vez que haya instalado Git, deberá configurarlo para que funcione correctamente en su computadora.

Abra la consola de **Git Bash** en su computadora y ejecute los siguientes comandos para configurar su nombre y dirección de correo electrónico:

```
git config --global user.name "Su nombre de usuario"  
git config --global user.email "su_correo@ejemplo.com"
```

Crear un repositorio

Una vez que hayas instalado GIT, puedes crear un nuevo repositorio. Para hacer esto, navega hasta la carpeta donde quieras crear el repositorio y utiliza el comando "**git init**" en la línea de comandos. Esto creará un nuevo repositorio vacío en la carpeta.



Actualizar el repositorio

git add

Ya creado el repositorio, puedes agregar archivos al mismo. Utiliza el comando "**git add**". Por ejemplo, si quieres agregar un archivo llamado "archivo.txt", utiliza el comando "**git add archivo.txt**".

También podemos agregar todos los cambios con "**git add .**" colocando un punto al final del add

git commit

Una vez que hayas agregado los archivos al repositorio, debes hacer un commit de los cambios. Un commit es una confirmación de los cambios que has realizado en el repositorio. Utiliza el comando "**git commit**" para hacer un commit de los cambios. Por ejemplo, si quieres hacer un commit con el mensaje "Agregando archivo.txt al repositorio", utiliza el comando "**git commit -m 'Agregando archivo.txt al repositorio'**".

git push

Si estás trabajando en un proyecto de software colaborativo, debes subir tus cambios a un repositorio remoto para que otros desarrolladores puedan acceder a ellos. Utiliza el comando "**git push**" para subir tus cambios al repositorio remoto. Por ejemplo, si quieras subir tus cambios al repositorio remoto llamado "origin", utiliza el comando "**git push origin**".

git fetch

Descarga los cambios del repositorio remoto al repositorio local, actualizando las referencias remotas en el repositorio local, pero sin fusionar esos cambios con la rama actual. Es decir, git fetch descarga los cambios del repositorio remoto y los almacena en un nuevo "apuntador" llamado `FETCH_HEAD`, que puede ser inspeccionado o fusionado con la rama actual utilizando otros comandos como `git merge` o `git rebase`.

Es importante destacar que git fetch no modifica los archivos locales del repositorio de trabajo, sino que simplemente actualiza las referencias remotas en el repositorio local, por lo que es útil para mantener actualizado el repositorio local y prevenir conflictos con otras personas que estén trabajando en el mismo proyecto.

git pull

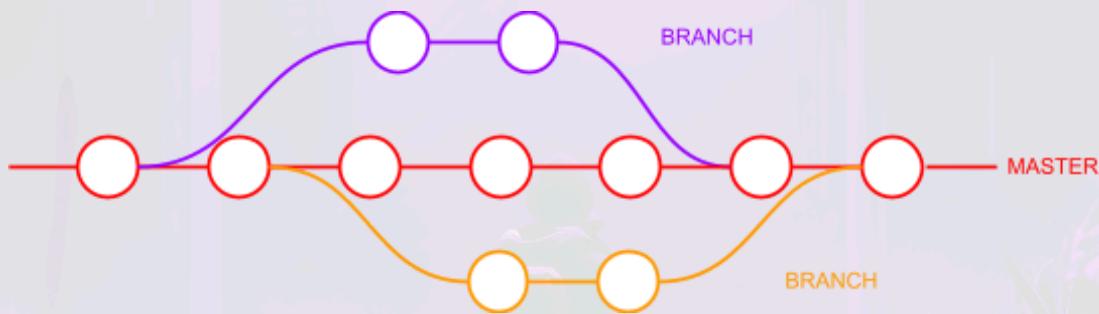
Si otros desarrolladores han subido cambios al repositorio remoto, debes descargarlos a tu repositorio local para tener la última versión del código. Utiliza el comando "**git pull**" para descargar los cambios. Por ejemplo, si quieras descargar los cambios del repositorio remoto llamado "origin", utiliza el comando "`git pull origin`".

Ramas

En Git, una rama (branch en inglés) es una línea independiente de desarrollo que permite trabajar en diferentes características o funcionalidades del proyecto sin afectar la rama principal (también conocida como rama master). Es decir, una rama es una copia del repositorio que se puede modificar sin afectar la rama principal.

Cuando se crea un nuevo repositorio en Git, automáticamente se crea una rama principal llamada "master". Cada vez que se crea una nueva rama, se está creando una copia independiente de la rama principal en la que se pueden hacer cambios y experimentar sin afectar la rama principal o cualquier otra rama. Las ramas son útiles para colaborar en equipo, ya que permiten trabajar en diferentes partes del proyecto de forma aislada y sin interferir con el trabajo de otros miembros del equipo.

Una vez que se está trabajando en una rama, se pueden realizar cambios, agregar archivos, confirmar los cambios con git commit, etc. Es importante destacar que los cambios realizados en una rama no afectan la rama principal ni otras ramas hasta que no se fusionan (merge) los cambios en la rama principal.



git branch

Si quieras trabajar en una nueva característica del proyecto sin afectar el código base, puedes crear una rama. Utiliza el comando "**git branch**" para crear una rama. Por ejemplo, si quieres crear una rama llamada "nueva-caracteristica", utiliza el comando "**git branch nueva-caracteristica**".

git checkout

Ahora que has creado una rama, puedes cambiar a la misma utilizando el comando "**git checkout**". Por ejemplo, si quieres cambiar a la rama "nueva-caracteristica", utiliza el comando "**git checkout nueva-caracteristica**".

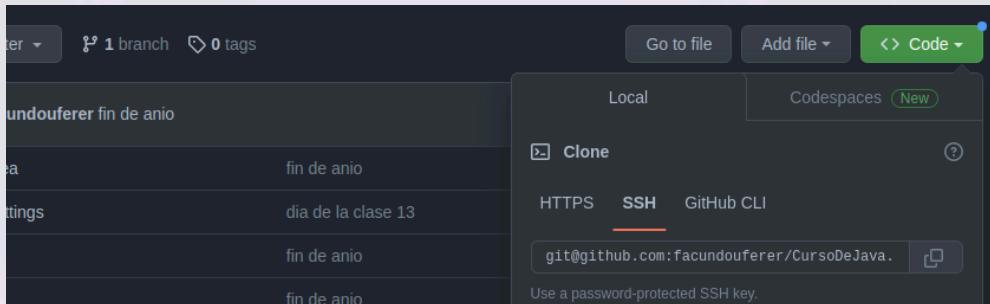
git merge

Una vez que hayas terminado de trabajar en la nueva característica, puedes fusionar la rama en la rama principal utilizando el comando "**git merge**". Por ejemplo, si quieras fusionar la rama "nueva-caracteristica" en la rama principal, utiliza el comando "**git merge nueva-caracteristica**".

git clone

En GitHub significa crear una copia local del repositorio en tu computadora, lo que te permite trabajar con el código y contribuir al proyecto. Clonar un repositorio es fácil y se puede hacer siguiendo estos pasos:

1. En GitHub, navega hasta el repositorio que deseas clonar. Por ejemplo, puedes buscar un repositorio utilizando la barra de búsqueda o navegar por la página de perfil del propietario.
2. Haz clic en el botón "Code" que se encuentra en la parte superior derecha del repositorio. Esto abrirá un menú desplegable con varias opciones para clonar el repositorio.
3. Copia la URL del repositorio. Puedes hacer clic en el botón "Copy" junto a la URL para copiarla al portapapeles.



4. Abre una terminal en tu computadora y navega hasta la carpeta donde deseas clonar el repositorio.
5. Ejecuta el comando "git clone" seguido de la URL del repositorio. Por ejemplo, si la URL del repositorio es "<https://github.com/usuario/repositorio.git>", ejecuta el siguiente comando:

```
git clone https://github.com/usuario/repositorio.git
```

6. Git descargará una copia completa del repositorio en la carpeta seleccionada. Una vez que la descarga esté completa, puedes trabajar con el código localmente y hacer cambios en el repositorio.

Es importante tener en cuenta que, si el repositorio es privado, es posible que debas autenticarte en GitHub antes de poder clonar el repositorio. Puedes hacerlo utilizando el protocolo HTTPS con tus credenciales de GitHub o utilizando el protocolo SSH y configurando una clave SSH en tu cuenta de GitHub.

git status

El comando "**git status**" es una herramienta muy útil en Git que permite conocer el estado actual del repositorio y de los archivos que se han agregado o modificado en el directorio de trabajo.

Cuando ejecutas el comando "git status", Git muestra información sobre los archivos que han sido modificados, añadidos o eliminados, así como los archivos que aún no han sido rastreados o que no han sido confirmados (staged) para la siguiente confirmación (commit).

Los elementos que muestra el comando "git status":

Branch

El comando "git status" muestra en qué rama se encuentra actualmente el repositorio y si hay cambios pendientes de fusionar con la rama actual.

```
On branch main
Your branch is up to date with 'origin/main'.
```

Working Tree

El comando "git status" muestra información sobre los archivos que se han modificado o eliminado en el directorio de trabajo.

```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git checkout -- <file>..." to discard changes in working
     directory)
```

```
modified: index.html  
modified: script.js
```

Staging Area

El comando "git status" muestra información sobre los archivos que han sido añadidos o confirmados (staged) para la siguiente confirmación (commit).

```
Changes to be committed:  
(use "git restore --staged <file>..." to unstage)  
  
modified: index.html
```

Untracked Files

El comando "git status" muestra información sobre los archivos que aún no han sido rastreados por Git.

```
Untracked files:  
(use "git add <file>..." to include in what will be committed)  
  
newfile.txt
```

Cómo usar el comando "git status"

Verificar el estado actual del repositorio:

```
$ git status
```

Verificar el estado actual del repositorio en detalle:

```
$ git status -s
```

Verificar el estado actual del repositorio y mostrar información adicional:

```
$ git status -v
```

Verificar el estado actual del repositorio y mostrar información detallada sobre los archivos no rastreados:

```
$ git status --untracked-files=all
```

GitHub

Existen varias alternativas de plataformas para utilizar Git además de GitHub y cada una de ellas tiene sus propias características y ventajas, pero para no hacerla tan larga, en este apunte nos centraremos en GitHub, una plataforma en línea que ofrece alojamiento de repositorios de código fuente y herramientas de colaboración para desarrolladores de software. Fue fundada en 2008 y desde entonces se ha convertido en una de las plataformas más populares para alojar y compartir proyectos de código abierto y privados.

GitHub ofrece una amplia gama de herramientas y servicios para apoyar el ciclo de vida del desarrollo de software, incluyendo la revisión de código, la gestión de problemas y errores, la integración continua y la implementación continua.

Además de ser utilizado por desarrolladores de software, GitHub también es utilizado por estudiantes, investigadores, empresas y organizaciones para alojar proyectos y colaborar en iniciativas de código abierto.

Login con Token

Para autenticarte en GitHub en tu ordenador utilizando un token, puedes seguir los siguientes pasos:

Generar un token de acceso personal en GitHub.

- 1) Inicia sesión en tu cuenta de GitHub.
- 2) Haz clic en tu foto de perfil en la esquina superior derecha y selecciona "Settings".
- 3) En la barra lateral izquierda, selecciona "Developer settings".
- 4) Selecciona "Personal access tokens".

- 5) Haz clic en "Generate new token".
- 6) Proporciona una descripción para el token y selecciona los permisos necesarios para las operaciones que deseas realizar con Git en GitHub.
- 7) Haz clic en "Generate token" y copia el token generado en el portapapeles.

Configurar Git en tu ordenador para utilizar el token.

- 1) Abre la línea de comandos en tu ordenador y utiliza el siguiente comando para establecer el nombre de usuario que se utilizará en Git:

```
git config --global --add user.name nombre_de_usuario
```

- 2) Utiliza el siguiente comando para establecer la dirección de correo electrónico que se utilizará en Git:

```
git config --global --add user.email correo
```

- 3) Utiliza el siguiente comando para establecer el token generado en github:

```
git config --global --add user.password token
```

Una vez que hayas seguido estos pasos, deberías poder autenticarte en GitHub en tu ordenador utilizando el token de acceso personal.