

# Comandos Básicos de Git

Kenneth Rodriguez

# ¿Qué Es?

Git es un sistema de control de versión (VCS en inglés), que se utiliza para gestionar los cambios de un archivo o proyecto a través del tiempo.

Git permite guardar los cambios de un proyecto, y “moverse en el tiempo” a los distintos cambios realizados. Esto permite realizar regresar a versiones anteriores del proyecto y realizar cambios, sin perder las modificaciones más recientes.



# Configuración

Antes de utilizar git propiamente, es muy útil configurar algunas propiedades para agilizar el uso de esta herramienta.

El comando **git config** servirá para esto, ya que permite modificar la configuración del software.

Si utilizamos **git config**, podremos configurar las características de git en el repositorio actual, mientras que **git config -global** servirá para configurar las características de todos los repositorios en un futuro.

Lo común es configurar el nombre y usuario, así que utilizaremos

- **Git config -global user.name “Nombre Apellido”**
- **Git config -global user.email “correo@email.com”**


# Iniciar Repositorio

- Utilizando la terminal, nos movemos al directorio donde se encuentra nuestro proyecto.
- El comando **git init** iniciará un repositorio local de git en el directorio actual.
  - Sin embargo, la rama por defecto se llama “master” y últimamente se está abandonando el nombre por otros como “main”. Podemos cambiar el nombre default con **git config --global init.defaultBranch “main”**.
- El comando **git status** nos dará información del repositorio actual, como el nombre de la rama, archivos nuevos, modificados y eliminados, etc.



# Preparando archivos

Durante esta etapa, seleccionaremos los cambios que queremos incluir en nuestro repositorio.

- El comando **git add [archivos]** se utiliza para añadir los archivos nuevos o modificados que se incluirán. Podemos incluir múltiples archivos simultáneamente.
    - La opción **git add .** incluye todos los archivos en el directorio actual.
    - Podemos utilizar el carácter **\*** para reemplazar texto. Por ejemplo, **git add \*.java** incluye todos los archivos con terminación **.java**
  - El comando **git rm** se usa para eliminar archivos o deshacer cambios
  - El comando **git restore --staged** se usa para quitar archivos del área de preparación, sin deshacer sus cambios.
- 

# Guardar Cambios

- Una vez tenemos preparado los cambios que queremos guardar, utilizamos el comando **git commit** para confirmar los cambios.
- Cada *commit* es como un momento o fotografía del proyecto que se guarda. Utilizamos los commits para observar los cambios del proyecto a través del tiempo.
- La opción **git commit** por sí sola abrirá un editor de texto (como Nano, VIM, etc.) para escribir un mensaje sobre el commit.
  - Para escribir un mensaje desde la consola, se utiliza el comando **git commit -m "Mensaje"** para guardar los cambios y escribir el mensaje de manera simultanea.


# Gestión de Ramas

- Si un commit es un momento en el tiempo de nuestro proyecto, podemos ver las ramas o *branches* como líneas del tiempo alternas.
- Las ramas permiten hacer cambios a un mismo archivo de forma paralela, sin que estos interactuen.
- El comando **git branch** se utiliza para listar las ramas disponibles en el repositorio.
- Si utilizamos **git branch nueva-rama** se creará una nueva rama a partir del commit en que nos encontramos, de nombre “nueva-rama”.
- Para cambiar entre ramas se utiliza el comando **git switch rama-destino**, que cambiará la rama actual a la rama-destino.




# Gestión de Ramas (cont.)


- Al cambiar de ramas es posible perder los cambios realizados si no hemos hecho un commit. Para esto se recomienda usar el comando **git stash save** para guardar los cambios en un buffer.
- Para observar los cambios guardados disponibles se usa el comando **git stash list**, que listará los distintos cambios.
- Cuando queramos recuperar los archivos del buffer se utiliza el comando **git stash pop NOMBRE** para sacar los cambios del buffer a la rama actual. El nombre de los cambios de puede obtener a través de **git stash list**.







# Clonar, Subir y Bajar Cambios

- Uno de los usos de git es descargar en nuestro equipo un repositorio completo. Esto se hace mediante el comando **git clone URL**, donde URL es la dirección del repositorio remoto.
  - Por otro lado, si nuestro repositorio local aún no cuenta con un repositorio remoto, podemos crear uno utilizando servicios como GitHub o GitLab.
    - Una vez hemos creado el repositorio remoto para nuestro proyecto, podemos utilizar el comando **git remote add origin URL** para agregar la ubicación a nuestro proyecto. Este comando solo conecta al repositorio local con el remoto, pero no envía ningún archivo.
  - Para subir una rama nueva y guardar los cambios en el repositorio remoto se utiliza el comando **git push -set-upstream origin rama**
    - Usos posteriores solo requieren del comando **git push**
- 
- 
- 



# Clonar, Subir y Bajar Cambios (cont.)

- Para obtener información sobre los cambios entre el repositorio local se utilizan dos comandos: **git fetch** y **git pull**.
    - **Git fetch** se utiliza para obtener los cambios en las ramas remotas de tu repositorio, sin afectar tus cambios actuales.
    - **Git pull** se utiliza para combinar los cambios de las ramas remotas en el repositorio actual, potencialmente afectando los cambios realizados.
  - Finalmente, el comando **git merge otra-rama** se utiliza para combinar los cambios de otra rama a la rama actual. Estas ramas deben tener historias relacionadas (es decir, tener un punto común en su historia).
    - Si ambas ramas modificaron el mismo archivo con cambios distintos, se debe arreglar cualquier conflicto entre las ramas.
    - El comando **git pull** básicamente hace un **fetch** de las ramas remotas y después hace un **merge** sobre la rama actual.
- 
- 
- 