GIT

Instalación

https://git-scm.com/download

Configuración

```
# Opciones obligatorias (nombre y correo)
git config --global user.name "Nombre y apellido"
git config --global user.email CORREO@ELECTRONICO
```

```
# Editor de preferencia: elegir solo una opción

# Editor de preferencia. Como primer ejemplo se incluye Notepad++ en Windows
git config --global core.editor "'C:/Program Files/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin"

# Editor de preferencia. Como segundo ejemplo se incluye Visual Studio Code
#  Referencia:  https://stackoverflow.com/questions/30024353/how-to-use-visual-studio-code-as-default-editor-for-git
git config --global core.editor "code --wait"
```

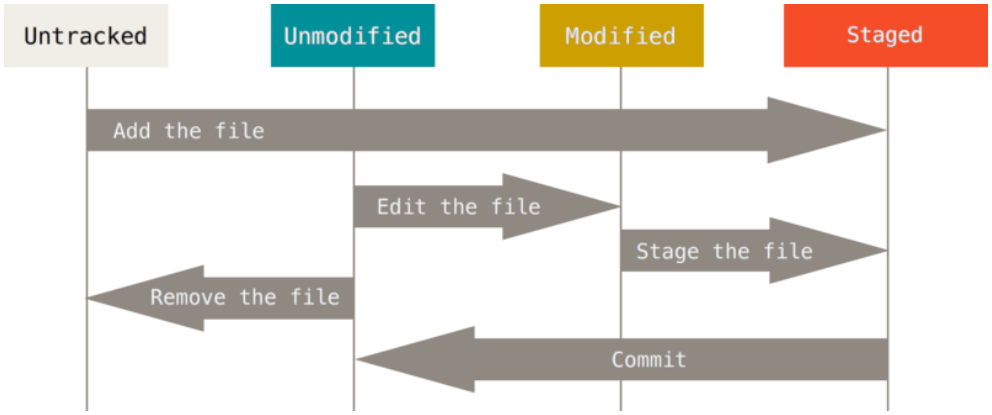
Si no se indica un editor de preferencia git utilizará el editor vim cuando tenga que solicitar la intervención del usuario (al hacer un merge, o si el usuario ejecuta git commit sin indicar el mensaje). Este editor es complicado de utilizar para alguien no iniciado, por lo que es muy recomendable cambiar el editor por defecto.

Creación de repositorios

Para crear un repositorio hay que situarse en la carpeta deseada y ejecutar:

```
git init
```

Ciclo de vida



Revisando el estado

```
git status
```

Esquema de colores:

- **Rojo** - Identifica los archivos **modificados o nuevos**. Si se crean archivos dentro de carpetas nuevas, git status solo mostrará el nombre de la carpeta, no su contenido. Si se desea ver el contenido de las carpetas nuevas se deberá ejecutar git status -u.
- **Verde** - Identifica los archivos en el **área de preparación**.

Visualizar cambios

```
git diff
git diff <archivo_o_ruta>
```

Este es uno de los comandos más utilizados en git. Nos permite ver los cambios en los archivos del repositorio o en una ruta específica.

Añadir archivos al área de preparación (stage)

```
git add <archivo> # Añadir archivos individuales
git add .  # Añadir todos los archivos nuevos o modificados
```

El **área de preparación** contiene los **cambios que se añadirán a la nueva versión** cuando ejecutemos un commit. Es posible la siguiente situación:

- Modificar un fichero (aparecerá en color rojo al hacer un git status)
- Añadir el fichero al área de preparación mediante git add FICHERO
- El fichero aparecerá en color **verde** al hacer un git status
- Volver a modificar el fichero
- El fichero aparecerá **dos veces** al hacer un git status:
    - En color **verde**, indicando que se ha añadido el **primer cambio** al área de preparación
    - En color **rojo**, indicando que hay un **segundo cambio** posterior que **no se ha incluido** en el área de preparación
- Si se ejecuta un git commit en este momento **solamente se incorporará el primer cambio** al repositorio como nueva versión. El segundo cambio seguirá existiendo (el archivo no habrá cambiado), pero no estará guardado en el commit
- Si se desea agregar el segundo cambio se deberá ejecutar nuevamente git add para añadirlo al área de preparación

Visualizar cambios de los archivos en el área de preparación

```
git diff --staged
git diff --staged <archivo>
```

Este comando muestra los cambios que se han agregado al área de preparación (diferencia entre la última versión guardada en el repositorio y el área de preparación).

Confirmar cambios (commit)

```
git commit -m "MENSAJE"
```

Un commit equivale a una nueva **versión** en el repositorio. Cada commit tiene un **identificador único**, denominado hash. Los commits están relacionados entre sí mediante una **red de tipo grafo**.

Ignorar archivos

- Archivo .gitignore
- Las rutas y nombres de archivo que aparezcan en el fichero .gitignore serán ignoradas por git **siempre que no hayan sido añadidas previamente al área de preparación o al repositorio**. Por ejemplo, si añadimos un archivo al área de preparación mediante git add y a continuación lo añadimos al fichero .gitignore, git lo seguirá manteniendo en el área de preparación, por lo que será incluido en el repositorio si ejecutamos un git commit.
- De igual manera, si previamente hemos guardado un archivo en el repositorio mediante git commit y a continuación lo incluimos en el fichero .gitignore, git no lo borrará: será necesario borrarlo del sistema de ficheros (a través de la consola o el navegador de archivos) y añadir los cambios (git add y git commit) para que se borre del repositorio. Una vez borrado, si lo volvemos a crear veremos que git sí que lo ignora si está incluido en el fichero .gitignore.

Historial de cambios

- git log
- git log --graph

Este comando muestra el histórico de los commits del repositorio. Se puede navegar en el listado mediante los cursores y la barra espaciadora. Para salir hay que pulsar la tecla q.

Ver cambios realizados en anteriores commits

```
git show <commit>
```

Este comando nos permite mostrar los cambios que se introdujeron en un determinado commit. En primer lugar se puede ejecutar git log para buscar el hash del commit que nos interese y a continuación ejecutar git show indicando después el hash del commit correspondiente.

Los hash de los commits tienen 40 caracteres, pero no es necesario copiarlos enteros: basta con indicar entre los 8 y 10 primeros caracteres para identificar un commit correctamente.

Quitar archivo del área de preparación

```
git reset <archivo>
```

En ocasiones nos encontramos con que hemos añadido cambios al área de preparación que no queremos incorporar al commit. Para ello podemos utilizar este comando, que elimina los cambios del fichero correspondiente del área de preparación. **Los cambios no se pierden** en ningún caso.

Eliminar las modificaciones con respecto al último commit
¡PELIGRO! Todos los cambios que se hayan hecho al archivo desde el último commit se eliminarán

```
git checkout -- <archivo>
```

Este comando es peligroso, ya que **elimina todos los cambios del archivo** que no hayan sido guardados en el repositorio. Es decir, si el archivo tiene cambios y está en color **rojo**, se perderán dichos cambios. Este comando puede ser útil para dejar un archivo tal como estaba en la última versión guardada del repositorio.

Etiquetado

```
git tag NOMBRE_TAG
```

Este comando crea un tag en el commit en que nos encontremos en este momento. Un tag es un **alias** que se utiliza para **hacer referencia a un commit** sin necesidad de saber su hash. Normalmente se utiliza para **indicar números o nombres de versiones** asociadas a un determinado commit. De esta manera podemos **identificar una versión de una manera más amable**.

El nombre de los tag se puede utilizar con los comandos de git: por ejemplo, git show.

Tareas:

1. Instala Git en tu sistema operativo. Adjunta una captura de pantalla en la que aparezca el resultado de la ejecución del comando git --version.
2. Realiza la **configuración de Git** según lo indicado en el tema (nombre, correo electrónico y editor de preferencia). Adjunta una captura de pantalla con el resultado de la ejecución de los comandos de configuración.
3. Crea una carpeta denominada DAW1. Realiza las siguientes acciones en ella:
    1. Crea un repositorio Git.
    2. Crea un fichero denominado libros.txt. Añade tres títulos de libros cada uno en una línea distinta.
    3. Haz un primer *commit*.
    4. Añade dos libros al archivo libros.txt.
    5. Haz un segundo *commit*.
    6. Crea un fichero denominado peliculas.txt. Añade tres títulos de películas a dicho archivo.
    7. Haz una captura de pantalla del comando git status.
    8. Crea un fichero denominado comidas.txt. Añade tres nombres de comidas a dicho archivo.
    9. Haz un tercer *commit* que incluya los archivos peliculas.txt y comidas.txt.
    10. Elimina el archivo comidas.txt desde el navegador de archivos.
    11. Añade dos películas más al archivo peliculas.txt.
    12. Haz una captura de pantalla que muestre los cambios en el directorio de trabajo.
    13. Añade los cambios al área de preparación.
    14. Haz una captura de pantalla del comando git status. Debe indicar que se ha borrado el archivo comidas.txt y que se ha modificado el archivo peliculas.txt.
    15. Haz un cuarto *commit*.
    16. Crea un archivo denominado datos.bak. Añade tres títulos de libros a dicho archivo. **¡IMPORTANTE! No añadas el archivo al área de preparación ni hagas ningún commit.**
    17. Crea una subcarpeta denominada output. Crea un archivo denominado salida.txt en su interior. Escribe tu nombre y apellidos en dicho archivo. **¡IMPORTANTE! No añadas los archivos al área de preparación ni hagas ningún commit.**

18. Haz una captura de pantalla del comando git status. Deben aparecer los archivos datos.bak y output/salida.txt como archivos nuevos (color rojo).
19. Crea un archivo .gitignore para que los ficheros con extensión .bak y el contenido de la carpeta output/ no se incluyan en el repositorio.
20. Haz una nueva captura de pantalla del comando git status. Ahora no deben aparecer los archivos datos.bak y output/salida.txt como archivos nuevos, sino que en su lugar debe aparecer únicamente el archivo .gitignore.
21. Haz un último *commit* para incluir el archivo .gitignore en el repositorio.
22. Haz una captura de pantalla que muestre el histórico de cambios del repositorio.