Git desde la CLI de LINUX

Primeros pasos

1. Comprobar que Git está instalado en nuestro sistema: ***git --help*** / ***git -h*** / ***git --version***
2. Si no estuviera instalado, instalarlo: ***sudo apt install git-all***
3. Realizar búsquedas de ayudas específicas para algunos comandos de Git (ej. *Config* y *add*): ***git config --help*** / ***git add --help***
4. Comprobar la configuración inicial de Git en nuestro sistema: ***git config --list*** Si no hemos utilizado nunca esta herramienta, aparecerá vacía la configuración.
5. Realizar una configuración básica de Git:
    a. Nombre de usuario: ***git config –global user.name "User_Name"***
    b. Correo electrónico: ***git config –global user.email ejemplo@unir.net***

En este punto, en el que hemos realizado una configuración básica mínima para poder utilizar Git, tenemos **dos escenarios posibles**:

1. **Inicializar un repositorio de un proyecto ya existente en nuestra máquina local.**
2. **Clonar un repositorio remoto.**

Inicializar un repositorio local

1. Crear un directorio de trabajo (ej. **Repo_Local**). En este repositorio vacío se recomienda crear un par de archivos o scripts de ejemplo (ej. *script1.sh* y *script2.sh*)
2. Inicializar el repositorio de Git: ***git init***. Utilizando el parámetro *-a* del comando *ls* se podrá observar la creación del subdirectorio *.git*. Este subdirectorio contiene todos los archivos de configuración y datos necesarios para rastrear los cambios en el repositorio. Si alguna vez queremos dejar de utilizar la herramienta Git en este repositorio, todo lo que hay que hacer es eliminarlo.
3. Estado actual de los cambios en el repositorio de Git: ***git status*****.** Este comando permite ver archivos que aún no han sido añadidos al repositorio de Git y archivos que han sido cambiados pero que no han sido añadidos a la preparación para el *commit*.
4. En el directorio, puede que haya archivos en los que no se desea realizar un seguimiento con Git y que, por tanto, tendríamos que intentar "evitar" en cada uno de los *commits*. Para solucionarlo, hay que crear un archivo ***.gitignore*** (se puede crear vacío con el comando *touch*). Este es un archivo de texto muy simple en el que solo hay que indicar qué archivos se desea ignorar:

```
# Ignorar todos los archivos .log
*.log

# Ignorar archivos específicos
Script2.sh
```

5. Volver a utilizar ***git status***. Se podrá observar que los archivos incluidos en *.gitignore* ya no aparecen disponibles para hacer *commit*.
6. Añadir cambios en el árbol de trabajo al área de preparación: ***git add -A*** (para añadir todos los archivos disponibles) / ***git add script.sh*** (para añadir un archivo específico). Estos archivos se están añadiendo temporalmente en el staging area (área de preparación). Esta es una zona intermedia donde se pueden agregar y organizar cambios antes de confirmarlos (*commit*). Es útil para revisar y controlar qué cambios específicos se incluirán en la próxima confirmación, sin afectar aún el historial del repositorio.
7. Volver a utilizar ***git status*** para observar los cambios.
8. Si se deseara eliminar archivos del staging area: ***git reset*** (para todos los archivos) / ***git reset script.sh*** (para un archivo específico).
9. Confirmar los cambios (commit changes): ***git commit -m "Description"***
10. Volver a utilizar ***git status*** para observar los cambios.
11. Mostrar el historial de confirmaciones o commits: ***git log***.

Clonar un repositorio remoto en nuestro repositorio local

1. Crear un directorio de trabajo (ej. **Repo_Remote**).
2. Clonar un repositorio Git en nuestro repositorio local: ***git clone <url.git> Repo_Remote***.
3. Visualizar información sobre el repositorio:
    a. ***git remote -v:*** URLs de los repositorios remotos que están configurados en el repositorio local. Tanto la URL de "fetch" (para obtener datos) como la de "push" (para enviar datos). También aparecerá el nombre del repositorio remoto (ej. origin).
    b. ***git branch -a:*** Este comando lista todas las ramas del repositorio, tanto locales como remotas. Las ramas locales aparecen sin prefijo, mientras que las ramas remotas aparecerán con el prefijo *remotes/.*
4. Realizar cambios en algún documento del repositorio que acabamos de clonar (ej. Usando el comando *nano README.txt*).
5. Observar las diferencias en el repositorio de Git: ***git diff***
6. Utilizar ***git status*** para observar los cambios.
7. Como se ha explicado anteriormente, añadir los cambios al staging area y confirmarlos utilizando ***git add -A*** seguido de ***git commit -m "Description".***
8. Subir los cambios confirmados desde el repositorio local a un repositorio remoto (acción conocida como hacer *push*):
    a. Actualizar la rama local con los últimos cambios del repositorio remoto antes de hacer un push: ***git pull origin main***.
    b. Subir los cambios confirmados desde el repositorio local a la rama principal del repositorio remoto: ***git push origin main***.

Creación de ramas

1. Creación de una rama nueva diferente a la principal: ***git branch "new_branch"***.
2. Comprobar y gestionar las ramas del repositorio: ***git branch***. El asterisco indica la rama en la que se está trabajando.
3. Cambiar de rama de trabajo: ***git checkout new_branch***
4. Volver a realizar cambios en algún documento del repositorio.
5. Como se ha explicado anteriormente, añadir los cambios al staging area y confirmarlos utilizando ***git add -A*** seguido de ***git commit -m "new_branch_changes".***
6. Listar todas las ramas locales y remotas disponibles en el repositorio: ***git branch -a***.
7. De momento, estos cambios no tienen efecto en el repositorio remoto. Para subir esta rama local al repositorio remoto: ***git push -u origin new_branch***.

Fusión de ramas (*merge*)

1. Cambiar la rama de trabajo a la principal: ***git checkout main***
2. De nuevo, hay que actualizar la rama local con los últimos cambios del repositorio remoto antes de hacer cualquier cambio en principal: ***git pull origin main***
3. Fusionar la nueva rama: ***git merge new_branch***.
4. Observar el historial de las ramas fusionadas: ***git branch --merged***.
5. Subir los cambios confirmados desde el repositorio local a la rama principal del repositorio remoto: ***git push origin main.***

Eliminación de ramas

1. Antes de eliminar una rama, volver a comprobar que se haya fusionado previamente: ***git branch --merged*.**
2. Eliminar una rama del repositorio local: ***git branch -d new_branch*.**
3. Listar todas las ramas locales y remotas disponibles en el repositorio, observando que ha sido borrada localmente: ***git branch -a***.
4. Eliminar la rama del repositorio remoto: ***git push origin --delete new_branch***