Centro Integrado de Formación Profesional

AVILÉS

Principado de Asturias

UNIDAD 1:
ARQUITECTURAS Y HERRAMIENTAS DE PROGRAMACIÓN
ACTIVIDAD 2

DESARROLLO WEB EN ENTORNO CLIENTE

DESARROLLO DE APLICACIONES WEB

Enunciado de la actividad

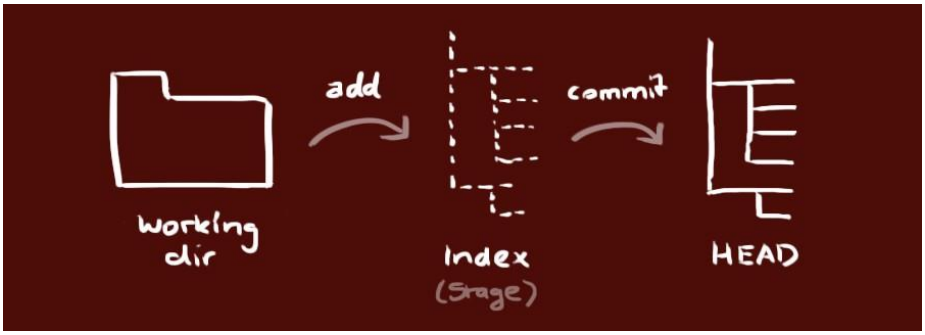
1. GIT

Git es un sistema de control de versiones distribuidas de código abierto y gratuito diseñado para manejar desde proyectos pequeños a muy grandes, con velocidad y eficiencia. Suele utilizarse para realizar un control de versiones entre un repositorio local y uno remoto. Se van haciendo cambios en el repositorio local y en cualquier momento se pueden enviar al remoto para que los demás participantes en el proyecto puedan disponer de los cambios realizados. Por tanto, puede verse que hay (o puede haber) dos zonas:

- La local, la cual siempre debe existir ya que es donde reside el proyecto en sí mismo.
- La remota, cuya existencia no es obligatoria, por ejemplo, en caso de un proyecto personal.

Flujo de trabajo normal

La zona de trabajo de un usuario está compuesta por tres "árboles" administrados por git. El primero es el **Directorio de trabajo,** el cual contiene los archivos con todos los cambios que se van haciendo. El segundo es el **Index** (área de preparación o stage area), el cual actúa como una zona intermedia. Por último´, queda el HEAD o repositorio local que apunta al último commit (o transacción) realizado.



1. Flujo Git

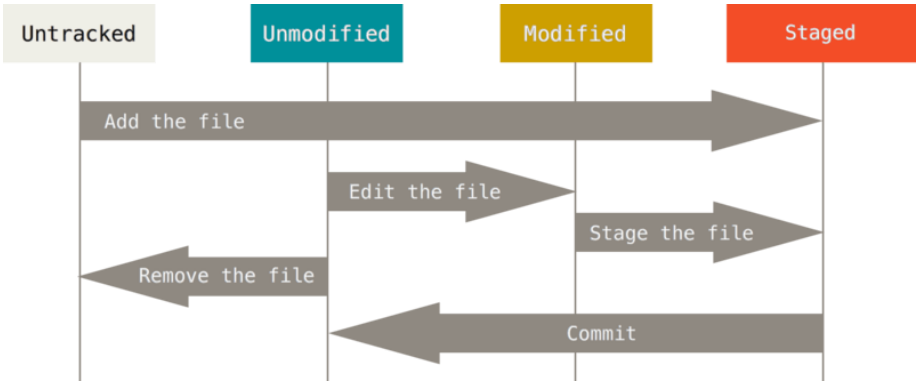
Cuando se realizan cambios en un archivo local, llega un momento en que el usuario decide que quiere guardar una copia de este con el fin de poder volver a la misma en cualquier momento. Para ello debe pasarlo a la zona stage (ensayo).

En ese momento, el usuario tiene una copia del archivo. Si decide que esa copia que ha pasado a staged ya es candidata para subirla al repositorio remoto, se debe mover a la zona Head.

En resumen:

- Working Directory. Mantiene los archivos en su estado actual.
- Index. Mantiene los archivos en un estado intermedio.
- Head. Mantiene los archivos candidatos a subir al repositorio remoto.

Si una versión concreta de un archivo está en el directorio de HEAD, se considera confirmada (committed). Si ha sufrido cambios desde que se obtuvo del repositorio, pero ha sido añadida al área de preparación, está preparada (staged). Y si ha sufrido cambios desde que se obtuvo del repositorio, pero no se ha preparado, está modificada (modified). Si un archivo está en el directorio de trabajo del usuario, pero no está gestionado por git, está en estado untracked.



2. Ciclo de vida Git

Zona Remota

La parte remota sólo está compuesta por el repositorio remoto y refleja el estado actual del proyecto. Es preciso tener en cuenta que en él pueden participar muchos desarrolladores.

Cuando se decide que los cambios hechos en local ya pueden pertenecer al repositorio remoto, se deben subir.

Configuración

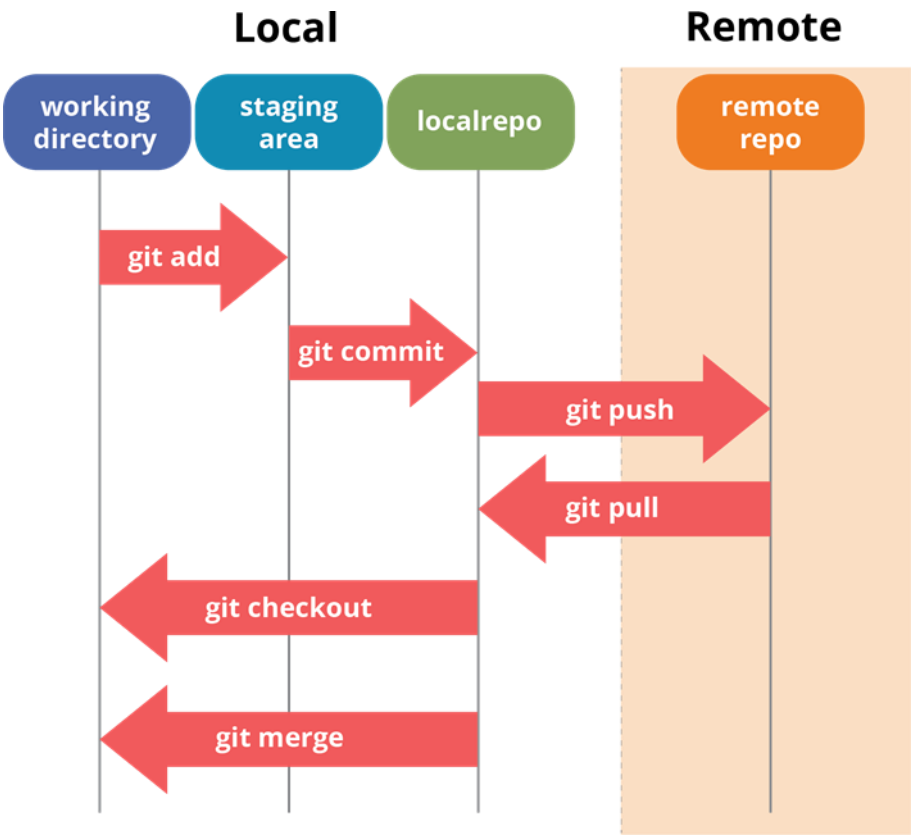
Lo primero que se debe hacer al instalar Git es establecer el nombre de usuario y la dirección de correo electrónico. Esto es importante porque los "commits" de Git usan esta información, y es introducida de manera inmutable en ellos cuando se produce su envío.

Antes de lanzar los comandos de configuración, crea una carpeta donde alojar los ficheros y posiciónate en ella con el comando cd (recuerda que el formato de las rutas de ficheros y directorios en Git Bash es el de Linux). A continuación, lanza los comandos de configuración:

```
git config --global user.name "Nombre Apellido"
git config --global user.email "micorreo@educastur.es"
```

Para ver la configuración basta con hacer:

```
git config --list
```



3. Git en local y remoto

Ayuda en Git

Si se necesita ayuda usando Git, existen tres formas de ver la página del manual (manpage) para cualquier comando de Git:

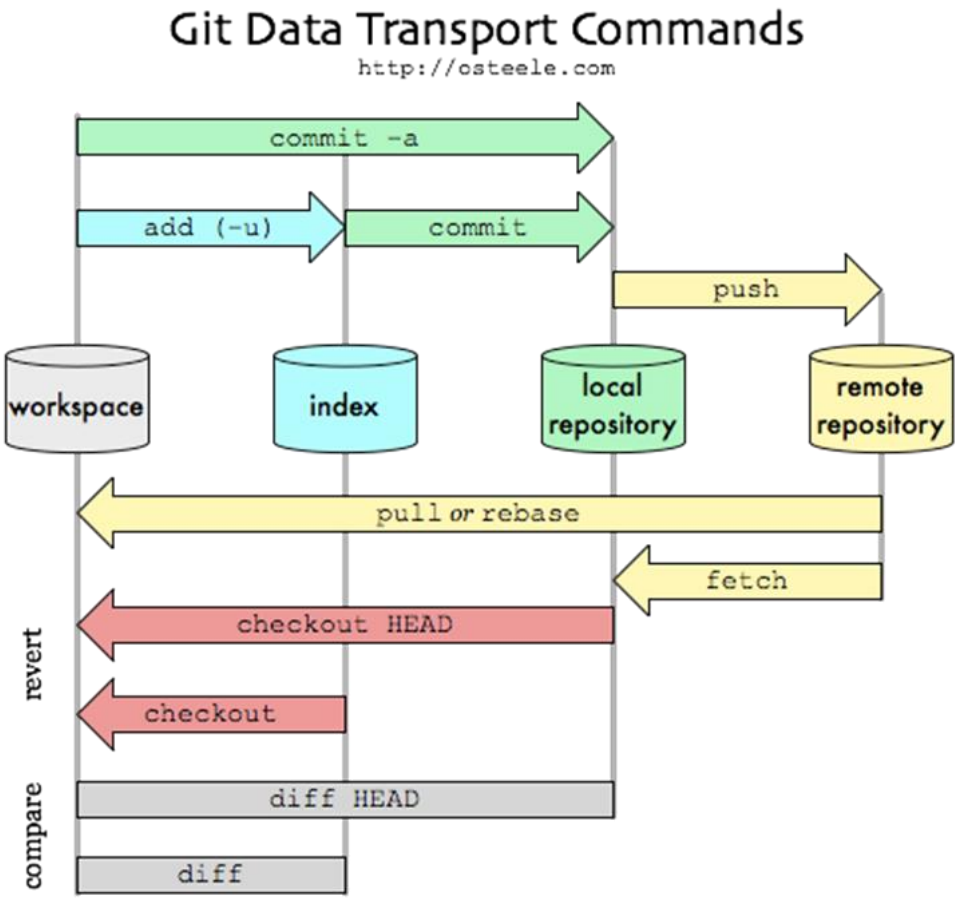
```
git help <verb>
git <verb> --help
man git <verb>;
```

Por ejemplo, se puede la página del manual para el comando config ejecutando

```
git help config
```

Comandos básicos

El siguiente diagrama muestra el flujo de trabajo junto con los comandos git



4. Comandos básicos de transporte Git

GitHub

Para poder trabajar correctamente con git, es necesario tener un repositorio donde alojar los proyectos. Para ello se va a usar GitHub que permite tener repositorios gratuitos.

Se van a realizar las siguientes operaciones relacionadas con GitHub:

1. En caso de tener una cuenta, autenticarse en ella. Si no, registrar una nueva. Si se prefiere tener un entorno separado para las prácticas de este módulo, registrarse con otra cuenta distinta de la habitual.
2. Una vez registrado, hacer clic en el botón "Read the guide" el cual sirve de guía para crear un nuevo repositorio en GitHub. Crear un nuevo repositorio con el nombre **hola-mundo**. En la descripción, indicar lo siguiente: "Proyecto Hola Mundo". El repositorio puede ser público o privado según se desee. Hacer que incluya un fichero README-md.
3. Una vez creado el repositorio, clonarlo asegurando que se está en el directorio creado anteriormente. Si el repositorio es privado, volverá a pedir autenticación.

   La orden es:

   ```
   git clone [url]
   ```

   Donde [url] es la url del repositorio a clonar. Para ver la url, hacer clic en **Code / Clone.**

4. Con el fin de hacer unas primeras pruebas, se va a crear un archivo de tipo "hola mundo" en Python dentro de un directorio de proyecto.

   Dentro del directorio recién clonado, lanzar la siguiente orden:

   ```
   echo "print('Hola desde DWEC (DAW)')" > holamundo.py
   ```

   Para comprobar el estado del repositorio local: git status

   ```
   git status
   ```

5. Se añade el archivo al área de ensayo (staged area):

   ```
   git add holamundo.py
   ```

   Se pueden usar wildchars (https://en.wikipedia.org/wiki/Wildcard_character). El archivo ahora estará en el área de ensayo, preparado para formar parte del historial del proyecto.

6. Se registran cambios en el historial:

   ```
   git commit -m "Primera versión de holamundo"
   ```

7. Se suben los cambios al repositorio:

   ```
   git push origin main
   ```

   En algunos casos, por ejemplo, cuando se utiliza git en máquinas Linux, el sistema pide usuario y contraseña de GitHub, pero al autenticarse, lanza un error ya que desde mediados del 2021 no se permite el acceso con contraseña sino mediante un token personal. Para crear dicho token basta con seguir esta URL propia de GitHub:

   https://docs.github.com/es/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token

Bitbucket

En este apartado, se probará a realizar operaciones con BitBucket (https://bitbucket.org/), otro alojamiento de proyectos Git. Para ello, también es preciso crear una cuenta.

Se van a realizar las siguientes operaciones relacionadas con BitBucket:

1. Crear un repositorio **vacío** llamado **holabb-mundo**. Dejar todas las demás opciones por defecto (privado, sin crear README, etc.) **excepto la de incluir .gitignore, que debe estar a NO**. Una vez creado, proporciona la URL del repositorio que se utilizará en el punto 5
2. Crear un directorio **con el mismo nombre** en el home del usuario y posicionarse en él. Lanzar la siguiente orden:

   ```
   git init
   ```

3. Ahora se añaden todos los archivos. **Utilizar el mismo script holamundo.py** anterior. Ejecutar la siguiente orden:

   ```
   git add *
   ```

4. Hacer el primer commit.

   ```
   git commit -m "Primer commit en Bitbucket"
   ```

5. Se añade la URL del repositorio remoto:

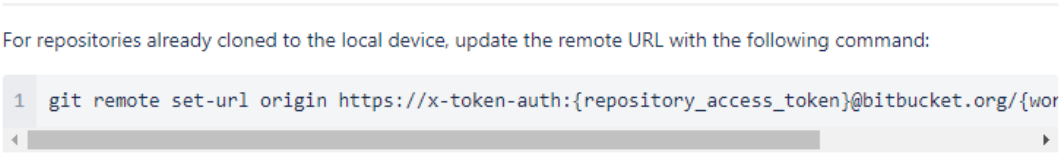
   ```
   git remote add origin URL-del-repositorio-remoto
   ```

6. Subir los cambios al repositorio remoto

```
git push origin main
```

Al igual que en Github, si se usa la consola de Linux, no sirve la contraseña de la cuenta y hay que obtener una. En el propio mensaje se muestra la URL para conseguirla, aunque a continuación se detalla el proceso:

- Primeramente, es preciso crear un token de acceso en el repositorio. Para ello hay que acceder a **Repository Settings / Access tokens / Create Repository Access Token**
- Se le proporciona un nombre identificativo y se marcan todos los permisos para repositorios, pipelines, etc.
- Una vez creado, el sistema muestra el token completo **el cual hay que conservar porque no se va a poder consultar más.** Aunque muestra varias opciones partiendo del clonado del repositorio, dado que en este caso se inicializa y se suben los cambios, se realizará de otra forma. Lo que sí es interesante es copiar la última parte, la referida a la configuración del correo del usuario y lanzarla en el repositorio local.
- En este enlace se indica cómo incluir el token en el repositorio local. En el apartado **Include the Repository Access Token in the URL**, dentro del apartado que aparece en la imagen, se indica cómo debe modificarse la URL del repositorio. Básicamente, consiste en añadir **x-token-auth:tokencompleto@...** a la URL original.



Una vez modificado el repositorio, al hacer el **git push origin main** se suben los cambios de forma automática sin solicitar ninguna credencial.