UD 03

CONTROL Y GESTIÓN DE VERSIONES

RESUMEN COMANDOS

ENTORNOS DE DESARROLLO

Fecha: 05/03/2023

********* ********* ********* *********

A. Configuración inicial

********* ********* ********* *********

Creación de repositorios

```
git init
```

Opciones obligatorias (nombre y correo)

git config --global user.name "Nombre y apellido"

git config --global user.email CORREO@ELECTRONICO

Almacenamiento de credenciales para no pedir usuario y contraseña de GitHub cada vez que se suban cambios al servidor

Ejecutar sólo uno de los dos comandos siguientes en función del sistema

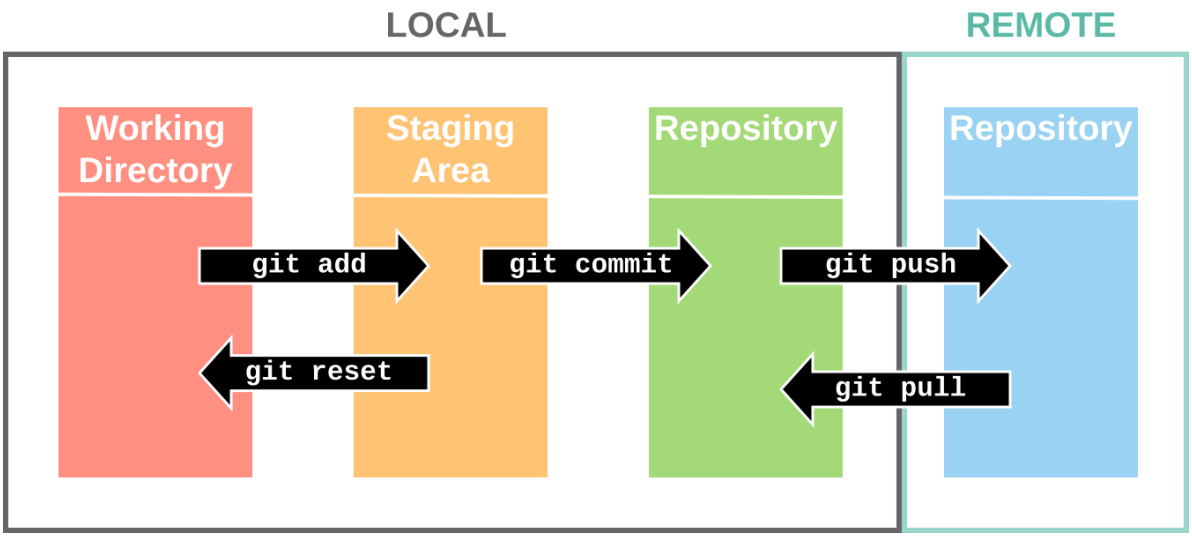
git config --global credential.helper cache # Para Linux

git config --global credential.helper wincred # Para Windows

********* ********* ********* *********

B. Ciclo de vida

********* ********* ********* *********



Añadir archivos al área de preparación (stage)

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

Confirmar cambios (commit)

```
git commit -m "MENSAJE"      # Confirmar modificación con comentario
```

Añadir, eliminar y renombrar remotos

```
git remote add <NOMBRE_REMOTO> <URL_REPOSITORIO>    # Añadir remoto
git remote rm <NOMBRE_REMOTO>                       # Eliminar remoto
```

```
Ejemplo:
            git remote add origin https://github.com/rpalaol/ED_01_REP_INICIAL
```

Enviar cambios al remoto

```
git push [NOMBRE_REMOTO] [NOMBRE_RAMA]

Ejemplo:
            git push origin master
```

Enviar los cambios de una rama al remoto y crear una rama remota asociada

```
git push -u [NOMBRE_REMOTO] [NOMBRE_RAMA]

Ejemplo:
            git push -u origin feature1
```

Traer y fusionar cambios del remoto

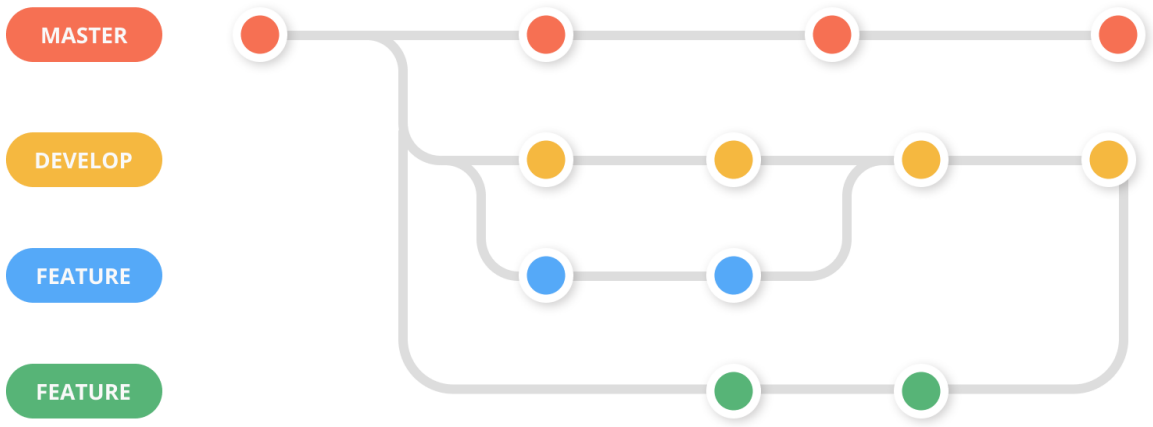
```
git pull [NOMBRE_REMOTO] [NOMBRE_RAMA]
Ejemplo:
            git pull origin master
```

********* ********* ********* *********

C. Ramas

********* ********* ********* *********

- *Un repositorio debe tener una rama como mínimo. El nombre de la rama por defecto es `master`.*
- *Existe un puntero especial llamado HEAD que apunta a la rama en la que estamos en ese momento.*
- *El trabajo con ramas es muy interesante por los siguientes motivos:*
    - *Se pueden hacer pruebas sin modificar el código en producción.*
    - *Se puede separar el trabajo en subproyectos que no afecten a otros.*
    - *Cada miembro del equipo puede trabajar de manera independiente.*
- ***SE CREA UNA RAMA FEATURE PARA CADA CAMBIO EN CÓDIGO y SE FUSIONA PRIMERO CON DEVELOP y LUEGO CON MASTER.***



Crear ramas

```
git branch <nombre_rama>
```

Ver ramas creadas

```
git branch
```

Cambiar de rama

```
git checkout <nombre_rama>
```

Fusionar una rama

Nos posicionamos en la rama sobre la que se va a realizar la fusión y ejecutamos:

```
git merge <nombre_rama_a_fusionar>
```

Eliminar una rama

```
git branch -d <nombre_rama>
```

********* ********* ********* *********

D. Otros

********* ********* ********* *********

Clonar un repositorio

```
git clone <URL_REPOSITORIO>
```

Revisando el estado

```
git status
```

Ignorar archivos

Archivo `.gitignore`

Visualizar cambios

```
git diff
git diff <archivo> #Podemos ver las diferencias entre nuestro fichero y el del último
                   commit.
```

Visualizar cambios de los archivos en el área de preparación

```
git diff --staged
git diff --staged <archivo>  #Podemos ver las diferencias entre nuestro fichero y el del
                             último add.
```

Historial de cambios

```
git log
git log –graph
```

Ver cambios realizados en anteriores commits

```
git show <commit>
```

Quitar archivo del área de preparación

```
git reset HEAD <archivo>
```

Eliminar las modificaciones con respecto al último commit

```
# ¡PELIGRO! Todos los cambios hechos al archivo desde el último commit se eliminarán
git checkout -- <archivo>
```