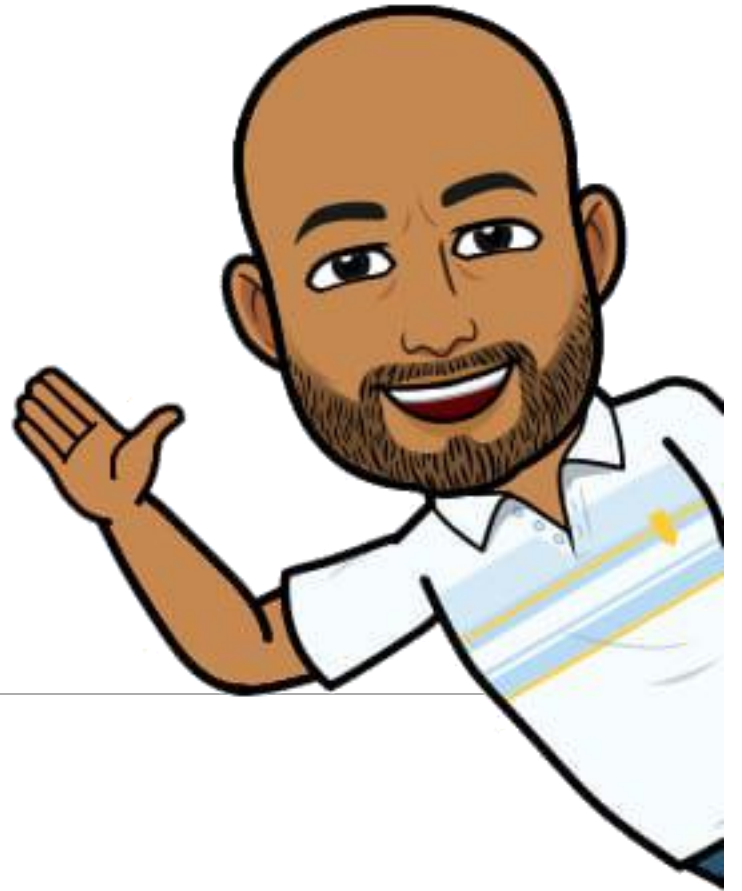




git



PRIMEROS PASOS EN GIT



CControl de
versiones ultimo



Cntrl revision2



Cntrl revision3



Control bueno



Control bueno
corregido



Control de
versiones 1



Control de
versiones 1.2



Control de
versiones Final



Control de
versiones Final1



Control de
versiones Final2



Control de
versiones Revisado



CONTROOOOL



Versiones listo



Versiones listo (2)



Configuración Básica

Quienes somos para git?

```
git config --global user.name "usuario"
```

```
git config --global user.email correo
```

Diferencia de Colores

```
git config --global color.ui true
```

Listar Configuraciones

```
git config --list
```

Configuración Básica

Archivos o Carpetas con nombres largos

`git config --system core.longpaths true`

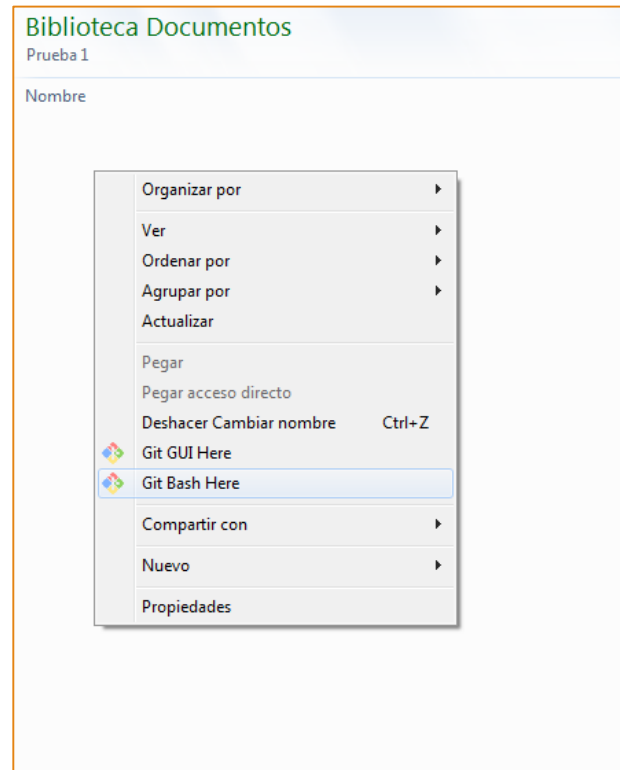
Finales de Linea o retornos de Carro (Windows)

`git config core.autocrlf true`

Inicio de Repositorio

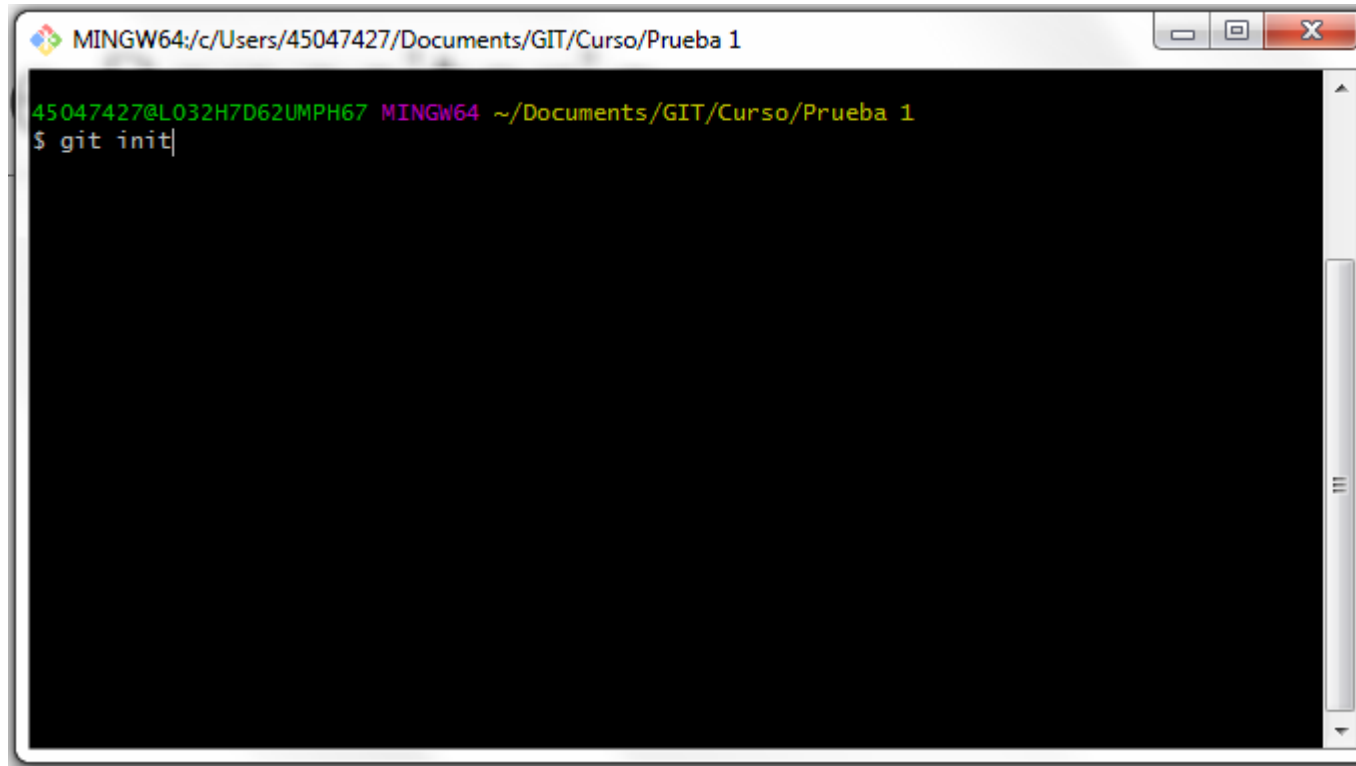
Dirigirte a la carpeta de tu Proyecto

1. Clic derecho del mouse
2. Git Bash Here



Inicio de Repositorio

1. git init



A screenshot of a Windows terminal window titled "MINGW64:/c/Users/45047427/Documents/GIT/Curso/Prueba 1". The terminal shows the command prompt "45047427@L032H7D62UMPH67 MINGW64 ~/Documents/GIT/Curso/Prueba 1" followed by the command "\$ git init". The terminal is otherwise empty, indicating the repository has been initialized successfully.

Si esta vacío el repositorio te mostrara una advertencia de inicializado pero se encuentra vacío

Estados de GIT

1. Modificas los archivos en tu directorio de trabajo.
2. Preparas los archivos, añadiéndolos a tu área de preparación.
3. Confirmas los cambios, lo que toma los archivos tal y como están en el área de preparación, y almacena “Fotos” de manera permanente en tu directorio de Git.



Seguimiento de GIT

git add

Añade contenido al Área de Preparación

git status

Muestra los diferentes estados de los archivos en tu directorio de trabajo y área de preparación

git diff

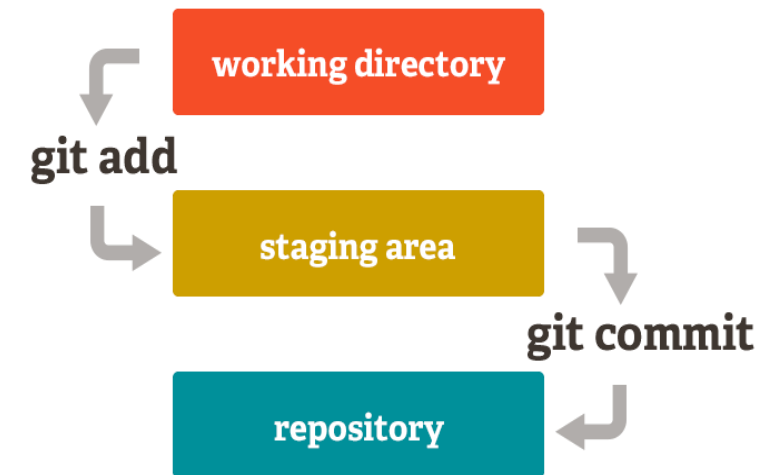
Muestra las diferencias entre Archivos

git commit

Toma todos los contenidos de los archivos que se encuentran en el área de preparación y toma un “foto” de tu repositorio

git log

Muestra el historia de “Fotos”

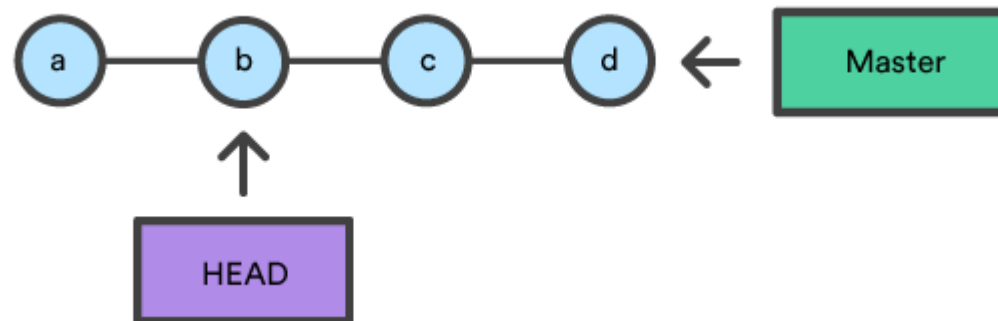
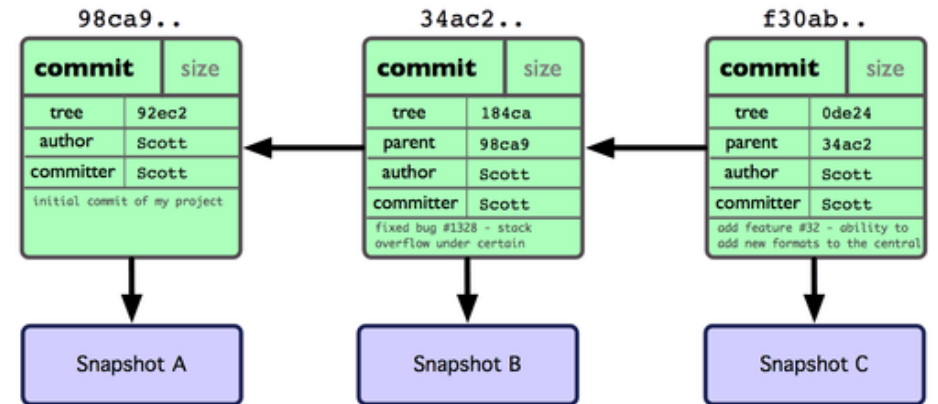


Commit

Cada commit en git es una “Foto” que representa el estado del proyecto al momento de la instrucción.

En algunos momentos es necesario posicionarnos en un commit específico, para ello usaremos el comando:

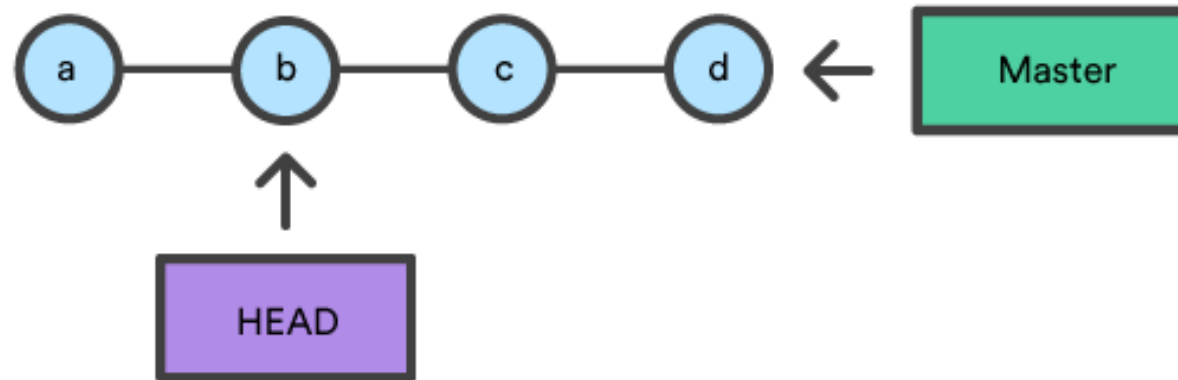
- `git checkout codigoSHA`



Posicionarnos entre commit

En algunos momentos es necesario posicionarnos en un commit específico, para ello usaremos el comando:

- `git checkout codigoSHA`



Reset

A veces esa “Foto” no es deseada por lo que necesitaríamos eliminarla

Todos estos eliminan los commits posteriores al reset

git reset --soft

conserva los cambios en el stage area

conserva los cambios que tengas en tus archivos (working directory)

git reset --mixed

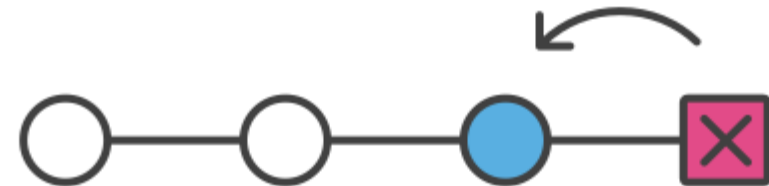
Deshace los cambios en el stage area

conserva los cambios que tengas en tus archivos (working directory)

git reset --hard

Deshace los cambios en el stage area

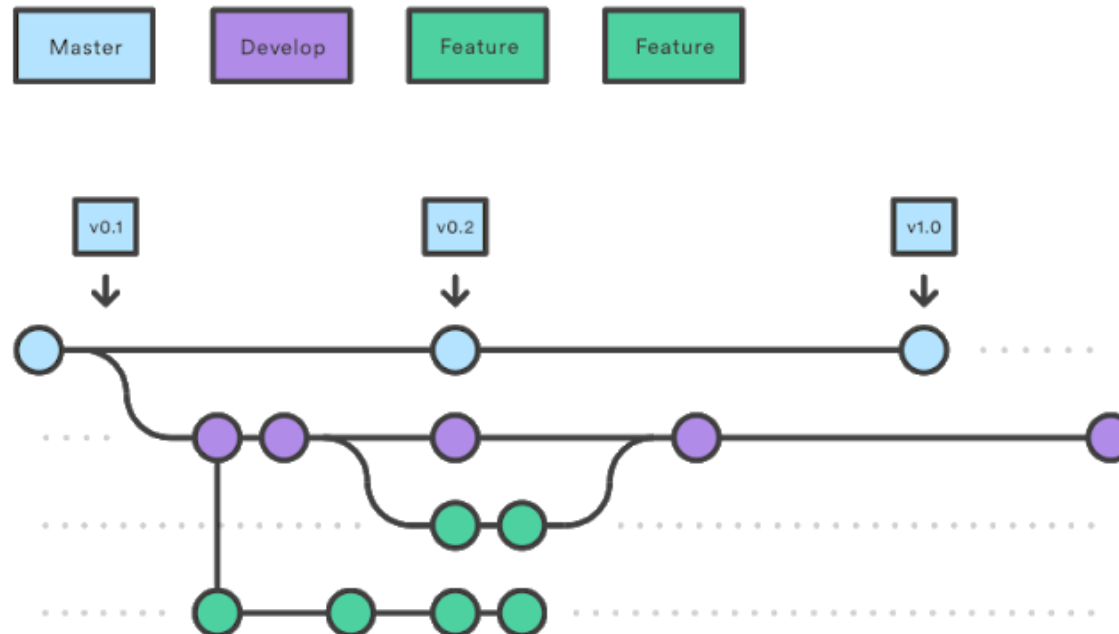
Deshace los cambios que tengas en tus archivos (working directory)



Ramas en GIT

Todos los repositorios de **git**, tienen la rama **master**

Cuando queremos hacer una modificación de un archivo en un repositorio que está en producción o bien en correcto funcionamiento, crearemos una rama alternativa con la que podremos trabajar y así no alterar el funcionamiento del repositorio **master**.



Ramas en GIT

git branch

Lista las ramas que hay en nuestro proyecto

git branch nombreRama

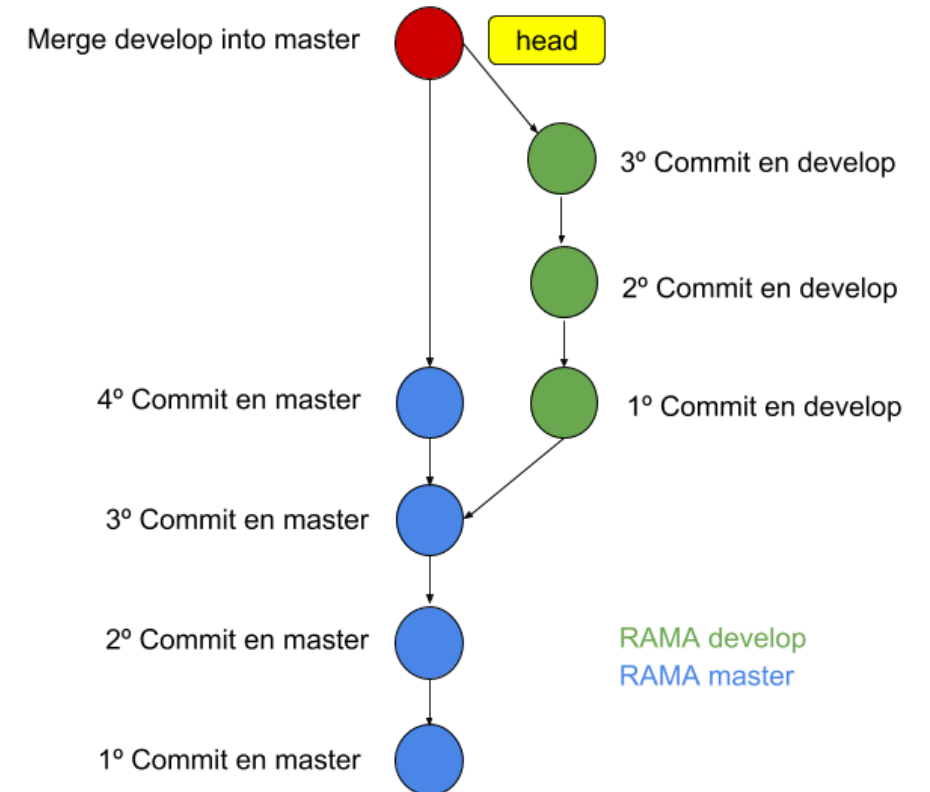
Crea una rama nueva a partir del commit posicionado

git checkout nombreRama

Nos mueve a la rama

git branch -D nombreRama

Elimina la rama



Merge

git merge nombreRama

Incorpora los cambios realizados en la rama posicionada.

Si no hay ningún conflicto ya habríamos terminado. En caso de que hubiera conflictos pueden darse tres situaciones.

La primera, es que Git haga un merge fast-forward, esto quiere decir que ambas ramas estaban alineadas y no se necesita hacer nada más.

La segunda, es que Git haga un auto merge. Esto es que ha necesitado hacer algunos cambios y es necesario hacer un commit adicional, con los cambios realizados. Hacemos el commit y habríamos terminado.

Y la tercera, es que Git no pueda resolver automáticamente los conflictos y nos pida que lo hagamos nosotros manualmente. Para ello, debemos abrir los archivos que Git nos indique y debemos buscar las zonas de código que nos haya anotado con HEAD y caracteres del tipo >>>>>>> y =====.

