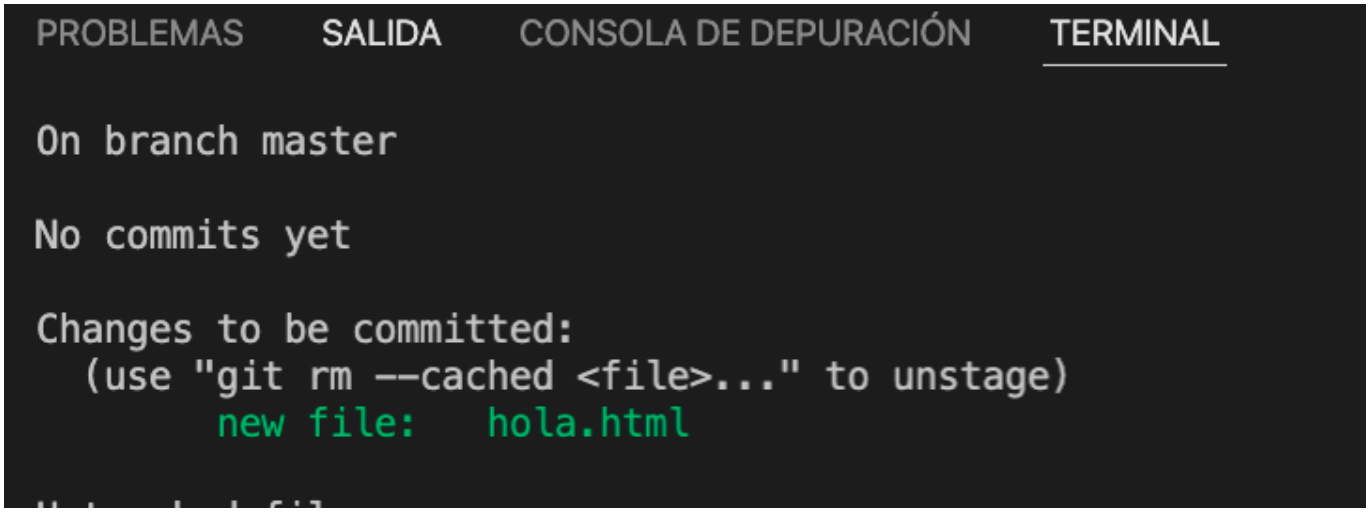
Tabla de Contenidos

- Git commit -a -m “mensaje”
- Git Commit y buenas prácticas (Premium)
- Git Commit y Mensajes
- Git Amend y corrección de commits.
- Otros artículos relacionados

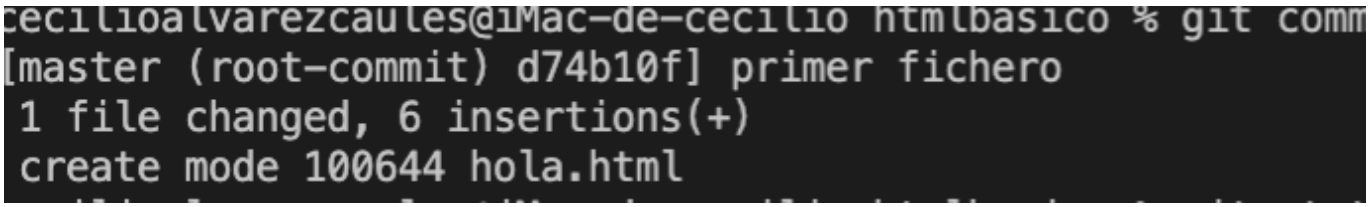
Git Commit es la operación más básica que podemos hacer con Git a la hora de guardar nuestros cambios en el repositorio. Ahora bien se trata de una operación que a veces cuesta entender como funciona realmente y para que sirven los diferentes pasos. que tenemos que realizar . Vamos a explicarlo . Para ello partiremos de un fichero html de hola.html

```
<html>
    <body>hola</body>
</html>
```

Nosotros lo primero que querremos hacer es añadirlo al staged Area a nivel de Git . Este area se encarga de almacenar ficheros en una situación temporal que van a ser salvados a futuro en el repositorio . Para ello usaremos el comando git add hola.html.Una vez que hemos realizado la operación de add es momento de usar git status para que nos muestre los ficheros que están disponibles en el staged area para realizar un commit.



En este caso únicamente tenemos el fichero de hola.html . Es momento de realizar una operación de commit y añadir el fichero al repositorio . Recordemos que al añadir el fichero tendremos que adjuntar un mensaje. Por lo tanto escribimos git commit hola.html -m “primer fichero”

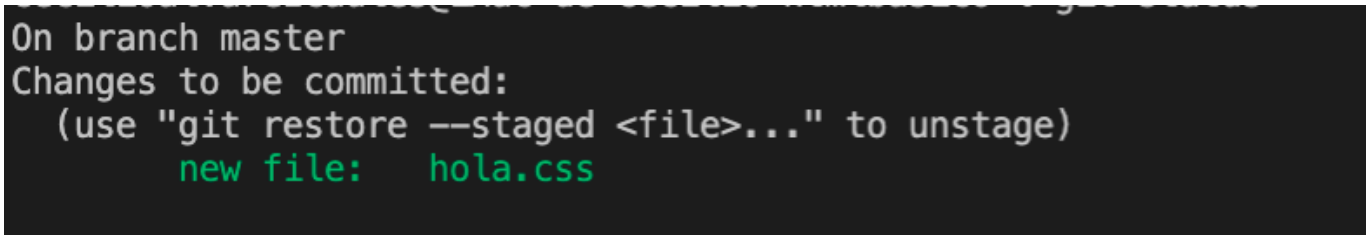


Vamos a añadir un fichero de css y hacer que el fichero html se apoye en la hoja de estilo para mostrar de otro color el texto.

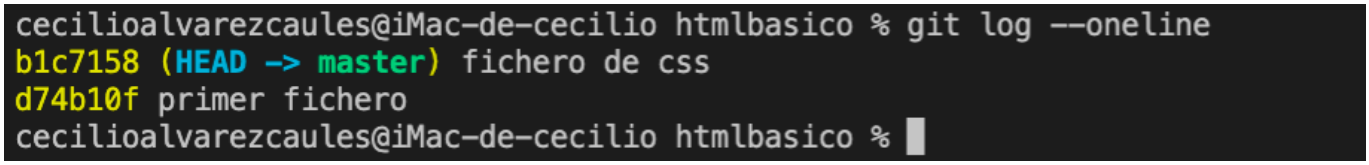
```
body {

    color:blue;
}
```

Volvemos a realizar la operación de git add hola.css y nos añadirá al staged area un nuevo fichero con lo cual ya podremos ver el contenido en el staged area y realizar una operación

de commit.



Escribimos git commit -m “fichero de css” y automaticamente tendremos realizados dos commits en nuestro repositorio que podemos mostrar con git log –oneline.



Git commit -a -m “mensaje”

Acabamos de realizar dos commits y tenemos el repositorio inicializado. Mucha gente considera que esto es demasiado repetitivo y prefiere usar un comando compacto que es git commit -a -m “mensaje”. De esta forma compacta las operaciones sobre Git . Esta operación se puede realizar con cualquier fichero que ya tengamos bajo control del repositorio y deseemos realizar cambios sobre él. Por ejemplo decido que el fichero de hola va a tener el texto de adios.

```
<html>
    <head>
        <link rel="stylesheet" href="hola.css">
    </head>
    <body>adios</body>
</html>
```

Para realizar un commit automático lo que debemos hacer es escribir git commit -a m

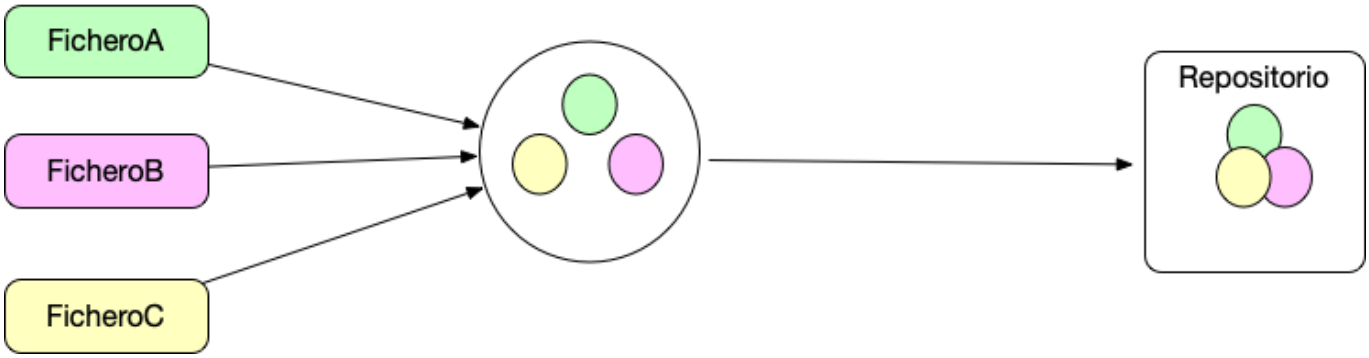
“modificaciones de adios” y eso hará que automaticamente los cambios sean salvados en el repositorio de golpe.

```
cecilioalvarezcaules@iMac-de-cecilio htmlbasico % git log --oneline
bcf8e25 (HEAD -> master) modificaciones adios
a6c4437 fichero3
0936620 segundo fichero
b1c7158 fichero de css
d74b10f primer fichero
cecilioalvarezcaules@iMac-de-cecilio htmlbasico %
```

Git Commit y buenas prácticas (Premium)

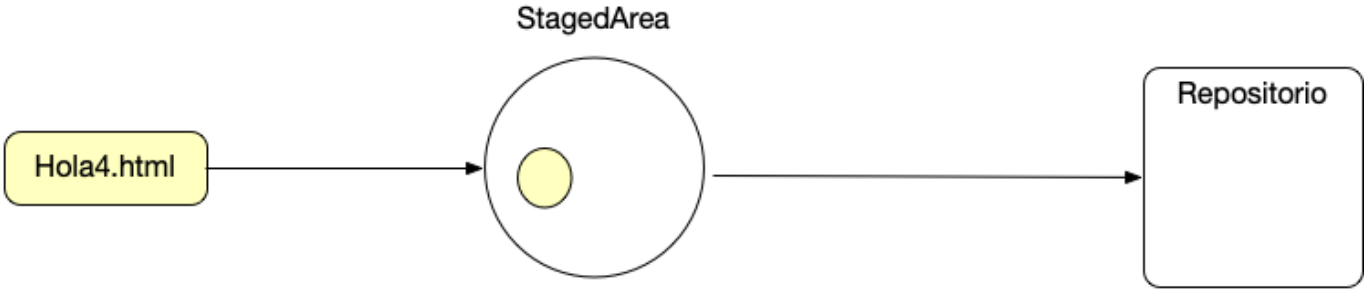
[ihc-hide-content ihc_mb_type=”show” ihc_mb_who=”4″ ihc_mb_template=”1″]

Muchas veces me encuentro que los desarrolladores abusan de este concepto . No es ni una buena ni una mala práctica pero si que es importante entender mejor cómo funcionan las cosas. Cuando nosotros trabajamos con el stagedarea muchas personas tienen dudas sobre para que sirve este área de trabajo. La mayoría de la gente piensa que se trata únicamente de un área donde se almacenan temporalmente los ficheros antes de realizar una operación de commit.

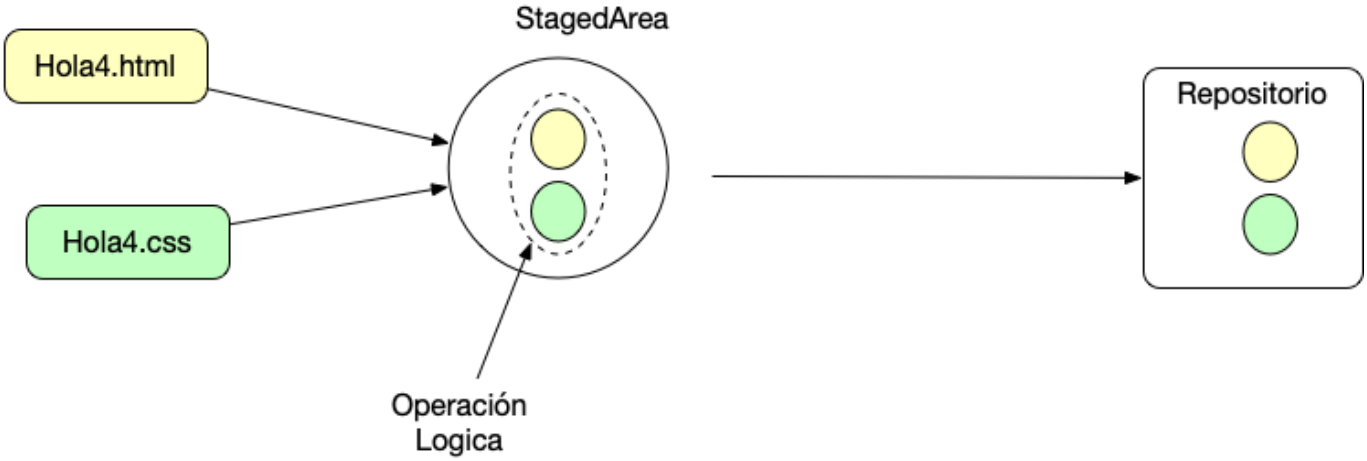


Pero la funcionalidad del Staged Area va más allá de esa situación ya que se encarga de agrupar operaciones lógicas de tal forma que cuando realicemos el commit estas tengan sentido y no simplemente realizamos commits a lo loco. Por ejemplo imaginemos que

añadimos al staged Area el fichero hola4.html y realizamos su commit con un mensaje , es razonable. Pero probablemente este fichero irá ligado también a una 004.css que es la que se encarga de añadir estilos. Por lo tanto una opción de trabajo razonable es crear el fichero hola4.html y realizar una operación de add e introducirlo en el stagedArea.



Realizada esta operación el siguiente paso es crear la css que va a estar ligada al documento e introducirla en el StagedArea:



Por último podemos realizar la operación de commit que incluya estos ficheros . De tal forma que por un lado reducimos el número de commits ya que no cada pequeño cambio que realicemos es un commit directo sino que intentamos que tenga sentido a nivel de funcionalidad almacenando varios ficheros en el stagedArea y luego realizando un commit de ellos en su conjunto.

Git Commit y Mensajes

Mi experiencia es que cuando nosotros vamos avanzando en el repositorio y los commits se acumulan nos encontraremos con que es muy difícil entender que es lo que hicimos hace 1 mes . Es importante que los mensajes de commit sean concisos y claros , hay gente incluso que le encaja añadir los ficheros que han sido modificados aunque git soporte comandos adicionales que permitan verlos.

- añadir tarea4 y su hola de estilo
- añadir tarea4 con los ficheros hola4.html y hola4.css

Git Amend y corrección de commits.

Hay situaciones en las que nosotros queremos modificar el mensaje del último commit o añadir ficheros al último commit antes de avanzar ya que hemos cometido un error y se nos han olvidado algunas cosas. Una solución rápida es usar –amend el el commit esto actualizará el mensaje.

git commit –amend -m “mensaje de modificacion”

[/ihc-hide-content]

Otros artículos relacionados

- Git Branch Java y Eclipse
- ¿Qué es el Versionado Semántico?
- Eclipse Git , Repositorios locales y remotos
- Curso introducción GIT (Gratuito)