

Comandos / conceptos útiles para el trabajo en equipo usando git

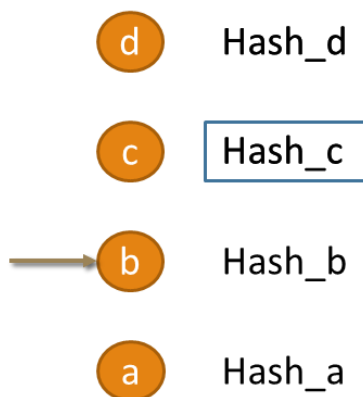
Revert / Reset

Ambos comandos sirven para regresar al estado del repositorio a un commit anterior, sin embargo, lo hacen de maneras diferentes.

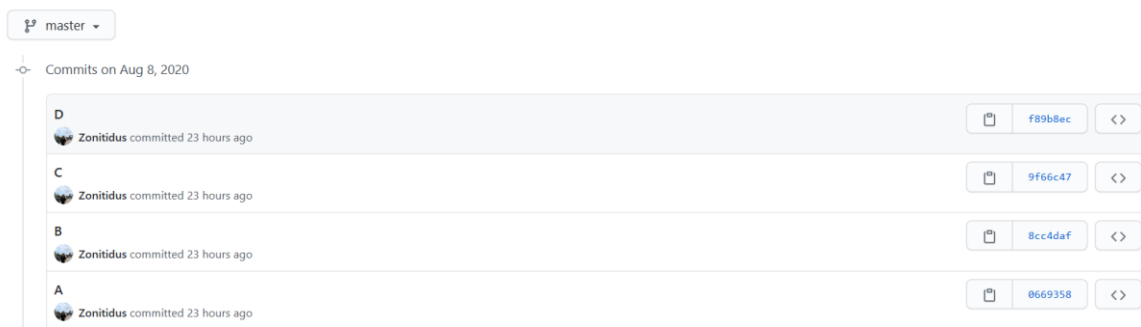
Revert

Revierte los cambios hechos en un commit específico. Cuando se utiliza, git regresa el estado del repositorio al commit anterior sobre el cuál se especificó.

Por ejemplo, si tuviésemos un repositorio con 4 commits y quisiéramos regresar al commit B, usaríamos el identificador del commit C.



En Git/GitHub se vería así:

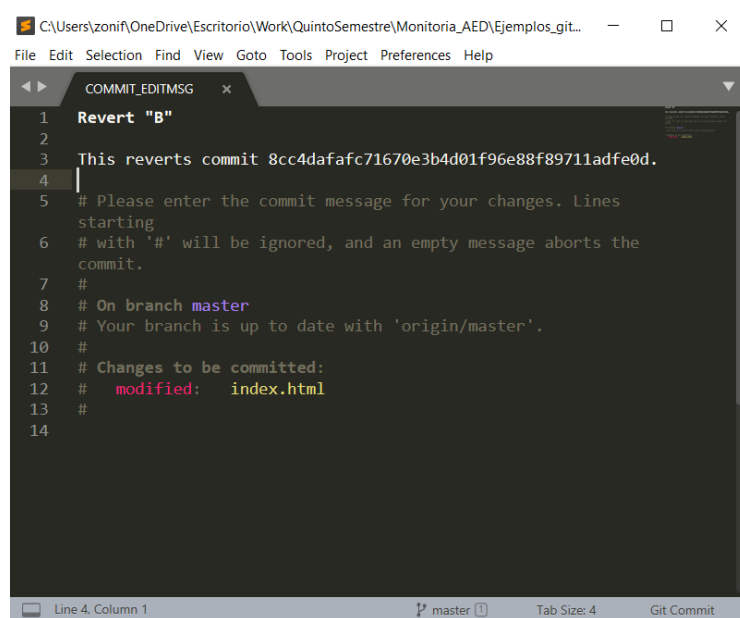


```
MINGW64:/c:/Users/zonif/OneDrive/Escritorio/Work/QuintoSemestre/Monitori...
zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori...
a_AED/Ejemplos_git/Revert-Reset (master)
$ git log --oneline
f89b8ec (HEAD -> master, origin/master) D
9f66c47 C
8cc4daf B
0669358 A

zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori...
a_AED/Ejemplos_git/Revert-Reset (master)
$ git revert 9f66c473de9174be62295427e8db46669574d932 <-- Hash del commit C
```

Al hacer esto, crearse o no un conflicto:

Cuando no hay conflicto



```
C:\Users\zonif\OneDrive\Escritorio\Work\QuintoSemestre\Monitoria_AED\Ejemplos_git...
File Edit Selection Find View Goto Tools Project Preferences Help
COMMIT_EDITMSG
1 Revert "B"
2
3 This reverts commit 8cc4dafafc71670e3b4d01f96e88f89711adfe0d.
4
5 # Please enter the commit message for your changes. Lines
6 # with '#' will be ignored, and an empty message aborts the
7 # commit.
8 #
9 # On branch master
10 # Your branch is up to date with 'origin/master'.
11 #
12 # Changes to be committed:
13 #   modified:   index.html
14
```

Se abrirá un archivo en el editor de texto con el cual tengan configurado git que básicamente se encargará de hacer el commit con el revert. Pueden editar el mensaje, la descripción del commit y los archivos modificados.

(En este caso, el editor de texto es sublime pero el que git usa por defecto es Vim, el cual se ejecuta en la misma consola)

Una vez configuramos el commit, debemos cerrar el editor para continuar y ejecutamos el comando git push para subirlo a GitHub



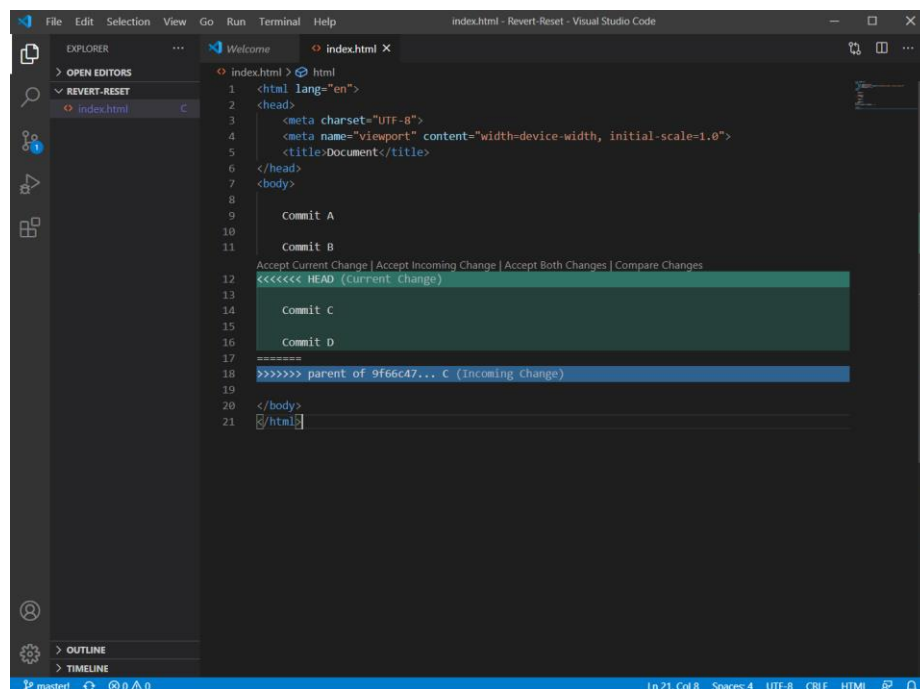
Cuando hay conflicto

Cuando ocurra este caso, saldrá algo similar e esto en la consola y tendremos que resolverlos.

```
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
error: could not revert 9f66c47... C
hint: after resolving the conflicts, mark the corrected paths
hint: with 'git add <paths>' or 'git rm <paths>'
hint: and commit the result with 'git commit'

zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori
a_AED/Ejemplos_git/Revert-Reset (master|REVERTING)
$ |
```

Para hacerlo, debemos ir al archivo y modificarlo según las marcas que deja git en el código



En el caso de VS Code, este marca los cambios y nos permite aceptar el cambio entrante con un solo botón. Aceptamos el cambio entrante y guardamos el archivo.

Una vez hecho esto, solo resta agregar lo cambios, hacer el commit y subirlo a git.

```
MINGW64:/c:/Users/zonif/OneDrive/Escritorio/Work/QuintoSemestre/Monitori...
zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori...
a_AED/Ejemplos_git/Revert-Reset (master)
$ git revert 9f66c473de9174be62295427e8db46669574d932
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
error: could not revert 9f66c47... C
hint: after resolving the conflicts, mark the corrected paths
hint: with 'git add <paths>' or 'git rm <paths>'
hint: and commit the result with 'git commit'

zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori...
a_AED/Ejemplos_git/Revert-Reset (master|REVERTING)
$ git add .

zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori...
a_AED/Ejemplos_git/Revert-Reset (master|REVERTING)
$ git commit -m "Revert del commit C"
[master 80d7c4a] Revert del commit C
1 file changed, 3 insertions(+)

zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori...
a_AED/Ejemplos_git/Revert-Reset (master)
$ git push
```

master

Commits on Aug 9, 2020

Revert del commit C

Zonitidus committed 2 minutes ago

80d7c4a

Commits on Aug 8, 2020

D

Zonitidus committed 23 hours ago

f89b8ec

C

Zonitidus committed 23 hours ago

9f66c47

B

Zonitidus committed 23 hours ago

8cc4daf

A

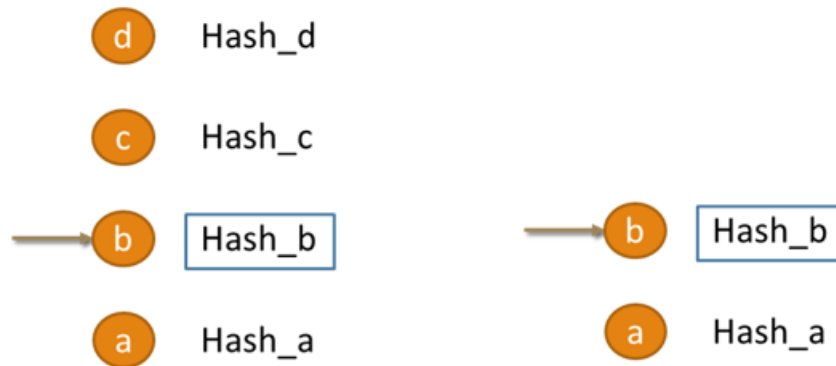
Zonitidus committed 23 hours ago

0669358

Reset

Este comando tiene muchas variaciones (`--soft` | `--mixed` | `--hard` | `--merge` | `--keep`); sin embargo, para el objetivo de esta guía, solo explicaremos cómo utilizar el `git reset --hard`

Este comando se encarga tanto de reiniciar el index como el directorio de trabajo y hace que todos los cambios hechos luego del commit seleccionado sean descartados.



Una forma de regresar al commit B usando reset sería la siguiente:

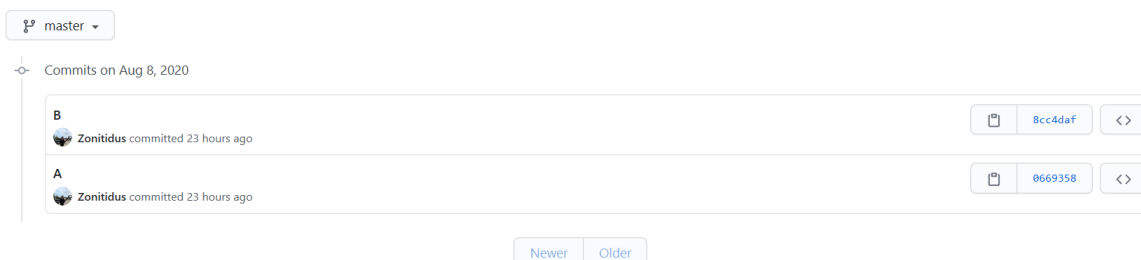
```
MINGW64:/c:/Users/zonif/OneDrive/Escritorio/Work/QuintoSemestre/Monitori...
zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori...
a_AED/Ejemplos_git/Revert-Reset (master)
$ git reset --hard 8cc4daf7c71670e3b4d01f96e88f89711adfe0d < -- Hash del commit B
HEAD is now at 8cc4daf B

zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori...
a_AED/Ejemplos_git/Revert-Reset (master)
$ git push -f
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Zonitidus/Rervert-Reset
+ f89b8ec...8cc4daf master -> master (forced update)

zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori...
a_AED/Ejemplos_git/Revert-Reset (master)
$ |
```

Usamos el `git push -f` para forzar el push. De otro modo, es probable que tengamos que resolver conflictos.

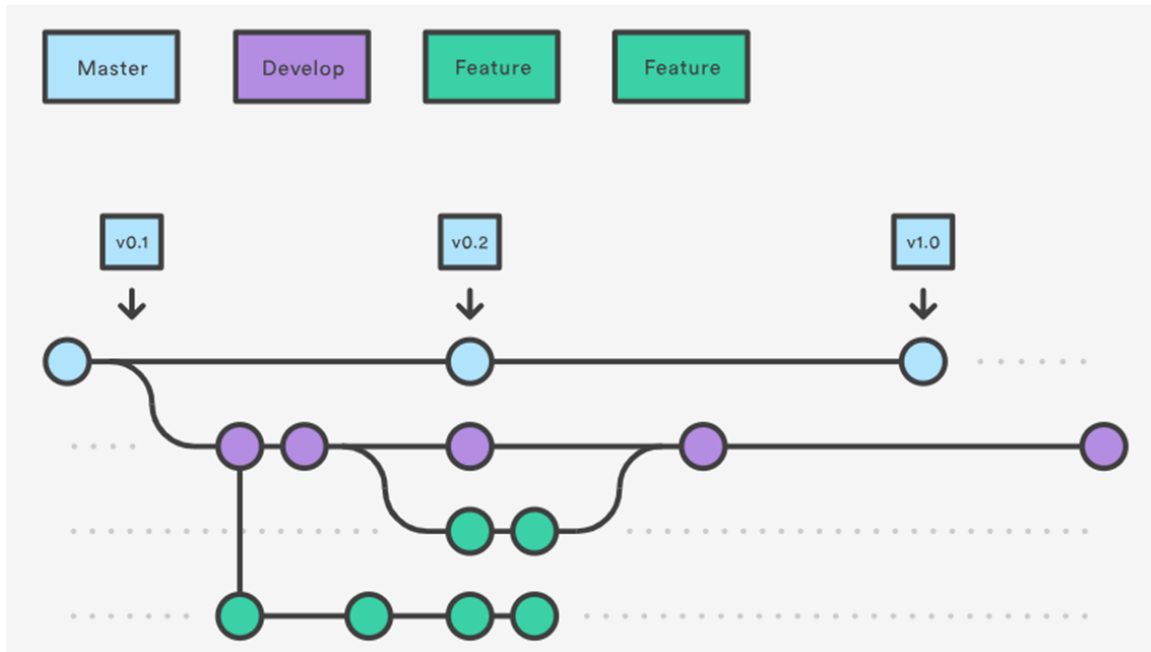
Resultado en Github



git revert	git reset
Es necesario hacer un commit	No es necesario según la variación que se escoja. En el caso de git reset --hard, no lo es.
No elimina el registro de cambios.	Elimina todos los cambios después del commit seleccionado.
Propenso a conflictos.	No suele tener conflictos. (Debido a la forma en que se implementó en la guía)

Gitflow Workflow

En general, el Gitflow Workflow es un modelo de ramificación para git que ayuda a que nuestro trabajo colaborativo sea más organizado y sencillo.



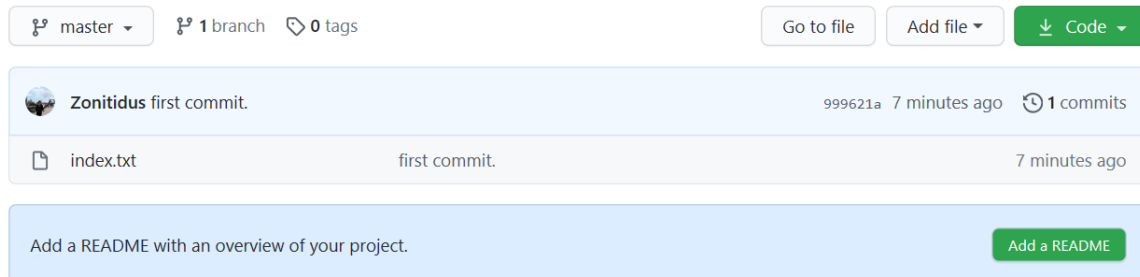
En su forma más sencilla, este modelo propone el uso de tres tipos de ramas:

- **Master:** Rama en la que se suben las versiones funcionales del proyecto. Si bien en la guía únicamente estábamos haciendo cambios en la rama master, esto no es lo ideal.
- **Feature:** Se desarrollan nuevas características. Cada colaborador debería tener su propia rama según las características en las que estén trabajando.
- **Develop:** Es aquí donde convergen todas las ramas de desarrollo antes de llevarlas a la rama master.

Cada desarrollador debería tener una rama propia; sin embargo, debe haber un único encargado en el proyecto de unir las ramas en develop.

Configuración de un proyecto usando el gitflow workflow

Partamos de un repositorio con un único archivo. Crearemos una rama develop y dos ramas de desarrollo: feat/login y feat/path_finding_algorithms.



Para crear una nueva rama usamos el comando `git checkout -b [Nombre]` y para subirla a Github usamos `git push [nombre del remoto] [nombre de la rama]`.

Es importante que la primera rama que creemos sea la rama develop, ya que de esta saldrán todas las ramas de desarrollo.

```
MINGW64:/c:/Users/zonif/OneDrive/Escritorio/Work/QuintoSemestre/Monitori...
zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori...
a_AED/Ejemplos_git/giflow-workflow (master)
$ git checkout -b develop
Switched to a new branch 'develop'

zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori...
a_AED/Ejemplos_git/giflow-workflow (develop)
$ git push origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/Zonitidus/giflow-workflow/pull/new/develop
remote:
To https://github.com/Zonitidus/giflow-workflow
 * [new branch]      develop -> develop

zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori...
a_AED/Ejemplos_git/giflow-workflow (develop)
```

Creamos la primera rama de desarrollo y la subimos a github.

```
zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori...
a_AED/Ejemplos_git/giflow-workflow (develop)
$ git checkout -b feat/Login
Switched to a new branch 'feat/Login'

zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori...
a_AED/Ejemplos_git/giflow-workflow (feat/Login)
$
```



```

zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori
a_AED/Ejemplos_git/giflow-workflow (feat/Login)
$ git push origin feat/Login
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feat/Login' on GitHub by visiting:
remote:   https://github.com/Zonitidus/giflow-workflow/pull/new/feat/Login
remote:
To https://github.com/Zonitidus/giflow-workflow
 * [new branch]      feat/Login -> feat/Login

zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori
a_AED/Ejemplos_git/giflow-workflow (feat/Login)
$ |

```

En este punto ya estamos listos para trabajar y editar nuestro código en la rama `feat/Login`. Para subir cambios a esta rama usaremos los comandos de siempre: `git add`, `git commit` y `git push`. Cuando ejecutemos el `git push`, debemos especificar el remoto y la rama; para este caso sería `git push origin feat/Login`.

Suponiendo que hayamos hecho algunos cambios en la rama y queramos dar por finalizado el desarrollo de esta funcionalidad, procederemos a llevar nuestro trabajo a la rama `develop`.

Para ello, primero debemos llevar la rama del repositorio remoto a nuestro repositorio local. Esto lo hacemos mediante el comando `checkout`. Una vez tenemos ambas ramas, nos situamos en `develop` y ejecutamos el comando `git merge [rama]`.

```

MINGW64:/c:/Users/zonif/OneDrive/Escritorio/Work/QuintoSemestre/Monitori...
zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori
a_AED/Ejemplos_git/giflow-workflow (develop)
$ git checkout feat/Login
Switched to a new branch 'feat/Login'
Branch 'feat/Login' set up to track remote branch 'feat/Login' from 'origin'.

zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori
a_AED/Ejemplos_git/giflow-workflow (feat/Login)
$ git checkout develop
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.

zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori
a_AED/Ejemplos_git/giflow-workflow (develop)
$ git merge feat/Login
Updating 999621a..3b19624
Fast-forward
 index.txt | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)

zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori
a_AED/Ejemplos_git/giflow-workflow (develop)
$ |

```

Y eso sería todo para esta funcionalidad. (Recuerden eliminar la rama con el comando `git Branch -d [rama]`)

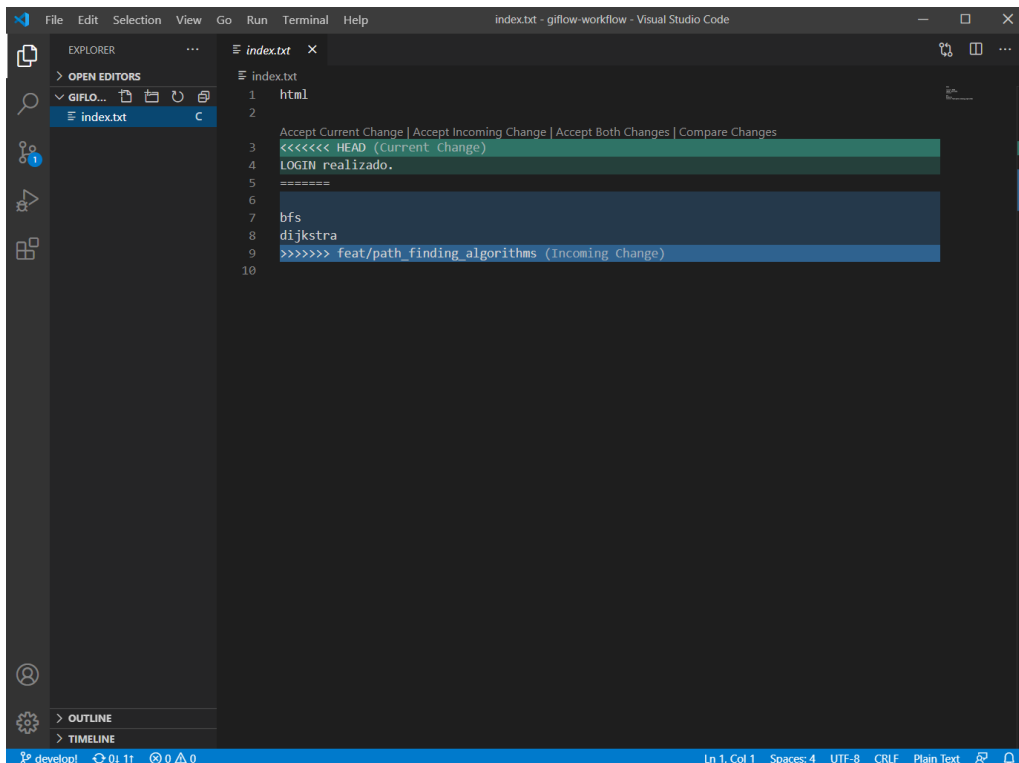
El primer merge a develop fue Fast-Forward, lo que indica que no hubo ningún problema. Sin embargo, esto solo sucede la primera vez, los demás merge están sujetos a conflictos que debemos solucionar.

Esto sucede cuando vamos a subir los cambios de la rama `feat/path_finding_algorithms` a develop:

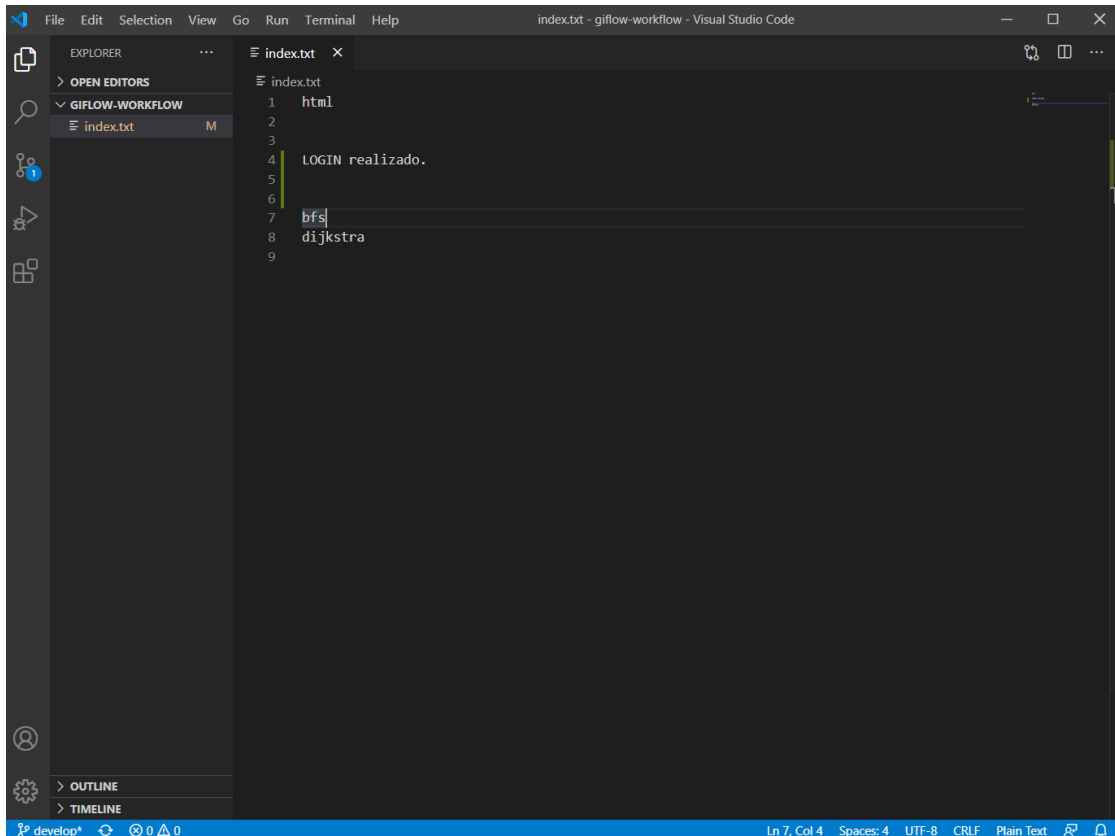
```
zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori
a_AED/Ejemplos_git/giflow-workflow (develop)
$ git merge feat/path_finding_algorithms
Auto-merging index.txt
CONFLICT (content): Merge conflict in index.txt
Automatic merge failed; fix conflicts and then commit the result.

zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori
a_AED/Ejemplos_git/giflow-workflow (develop|MERGING)
$ |
```

Para continuar debemos solucionar el conflicto de la misma manera en que se hizo con el comando `revert`, modificando el código.



Se acepta el cambio entrante y el actual (importante que sean ambos)



```
File Edit Selection View Go Run Terminal Help
index.txt - giflow-workflow - Visual Studio Code

EXPLORER
> OPEN EDITORS
  GIFLOW-WORKFLOW
    index.txt M

index.txt
1  html
2
3
4  LOGIN realizado.
5
6
7  bfs
8  dijkstra
9
```

Por último, registramos el commit y lo subimos al repositorio en Github.

```
zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori
a_AED/Ejemplos_git/giflow-workflow (develop)
$ git merge feat/path_finding_algorithms
Auto-merging index.txt
CONFLICT (content): Merge conflict in index.txt
Automatic merge failed; fix conflicts and then commit the result.

zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori
a_AED/Ejemplos_git/giflow-workflow (develop|MERGING)
$ git add .

zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori
a_AED/Ejemplos_git/giflow-workflow (develop|MERGING)
$ git commit -m "Merged path_finding_algorithms"
[develop 8571014] Merged path_finding_algorithms

zonif@DESKTOP-M89GCV5 MINGW64 ~/OneDrive/Escritorio/Work/QuintoSemestre/Monitori
a_AED/Ejemplos_git/giflow-workflow (develop)
$ git push origin develop
```

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Overview Yours Active Stale All branches Search branches...

Default branch

master Updated 2 hours ago by Zonitidus Default Change default branch

Your branches

develop Updated 30 minutes ago by Zonitidus	0 3	New pull request	
feat/path_finding_algorithms Updated 43 minutes ago by Zonitidus	0 1	New pull request	
feat/Login Updated 1 hour ago by Zonitidus	0 1	New pull request	

Active branches

develop Updated 30 minutes ago by Zonitidus	0 3	New pull request	
feat/path_finding_algorithms Updated 43 minutes ago by Zonitidus	0 1	New pull request	
feat/Login Updated 1 hour ago by Zonitidus	0 1	New pull request	

Quise ser lo más conciso posible, pero si en algún punto no quedó muy claro el procedimiento o tienen alguna duda al respecto, pueden escribirla por el canal de discord y la contestaré en un espacio que tenga libre o durante la clase de laboratorio.

Espero le den buen uso a esta información. Éxitos :)