

# 01 - Ejercicio de Git - Forty - local y remoto

## 01 - Ejercicio de Git - Forty - local y remoto

Trabajo en local

Trabajo en remoto

## Trabajo en local



Este trabajo se ha realizado en Windows y en Mac. Para evitar confusiones se ha renombrado el proyecto a `forty2` en Mac para que se vieran todos los flujos de trabajo en local en mi dispositivo .



Antes de nada, abrimos la terminal y nos colocamos en la carpeta donde queremos inicializar el repositorio o si estamos en Windows directamente desde la carpeta damos al boton derecho y escogemos la terminal bash.

1. Inicializar un nuevo repositorio Git en una carpeta llamada `"forty"` y agrega los archivos proporcionados en el aula virtual.

```
$ git init
```

Aqui se crea una carpeta `.git` que es lo que nos indica que es un repositorio

Como no quiero añadir todos los archivos porque algunos los vamos a ignorar, los voy metiendo uno a uno.

Listamos para ver que archivos contiene la carpeta:

```
$ ls
```

Vamos agregando los archivos que no vamos a ignorar:

```
$ git add assets
```

Comprobamos que estan en el staging: (nos dice los que estan trackeados y los que no):

```
$ git status
```

Hacemos commit de los que hemos agregado:

```
$ git commit -m "Añadidos los archivos iniciales"
```

2. Renombra la rama master a `main`.

Este paso no se hace porque en la configuración de Git ya lo hemos hecho.

3. Haz que los ficheros `README.txt`, `LICENSE.txt` y `passwords.txt` sean ignorados por el control de versiones.

Creamos el `.gitignore`

```
$ touch .gitignore
```

Después abrimos el archivo generado e incluimos el nombre de los archivos que tenemos que ignorar. (editamos el archivo directamente). Aquí ya dejamos escrito el nombre del fichero `passwords.txt` para que lo ignore cuando lo creamos.

Después debemos añadir este archivo al repositorio .git

```
$ git add .gitignore
```

Se hace un commit de este añadido.

```
$ git commit -m "Añadido .gitignore para ignorar los archivos sensibles"
```

4. Crea el archivo `password.txt`. Comprueba que el control de versiones lo ignora.

Para introducir los ficheros que queremos ignorar los escribimos en el archivo de `.gitignore`. si necesitamos crear alguno como en esta caso el `password.txt`

lo hacemos con el comando: (Se puede usar este o nano)

```
$ notepad password.txt
```

Escribimos en él lo que queramos y lo guardamos.

Para saber que archivos están ignorados lo haremos con una de estas dos formas:

```
$ git ls-files --others -i --exclude-standard
```

```
$ git status
```

5. Crea una rama llamada "feature-content" . Muevete a esa rama. Cambia, en la línea 3477, el font-size por 1.5em en el archivo main.css . Ver los logs de la forma más gráfica posible.

```
$ git branch feature-content
```

Comprobar que ramas tenemos:

```
$ git branch
```

Para cambiarnos de rama:

```
$ git checkout feature-content
```

Hacemos los cambios en el fichero que nos indican y guardamos.

Para ver los logs de la manera más gráfica:

```
$ git log --graph --oneline --all --decorate
```

```
fatal: an unrecognized argument: --oneline
2dawv02@A306PC02 MINGW64 ~/Documents/forty (feature-content)
$ git log --graph --oneline --all --decorate
* 8ae084c (HEAD -> feature-content, main) añadidos archivos al .gitignore
* 404440d Añadido .gitignore inicial para ignorar archivos
* f13bbf6 añadidos los ficheros del proyecto forty
2dawv02@A306PC02 MINGW64 ~/Documents/forty (feature-content)
$
```

Tenemos que añadir los cambios que hemos hecho.

```
$ git add assets/css/main.css
```

```
$ git commit -m "Añadidos cambios en linea 3477, font-size: 1.5em"
```

```
2dawv02@A306PC02 MINGW64 ~/Documents/forty (feature-content)
$ git status
On branch feature-content
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   assets/css/main.css

no changes added to commit (use "git add" and/or "git commit -a")

2dawv02@A306PC02 MINGW64 ~/Documents/forty (feature-content)
$ git add assets/css/main.css
warning: in the working copy of 'assets/css/main.css', LF will be replaced by CRLF the next time Git touches it

2dawv02@A306PC02 MINGW64 ~/Documents/forty (feature-content)
$ git status
On branch feature-content
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   assets/css/main.css

2dawv02@A306PC02 MINGW64 ~/Documents/forty (feature-content)
$ git commit -m "Cambios en la linea 3477 de main.css"
[feature-content 8fa666e] Cambios en la linea 3477 de main.css
 1 file changed, 1 insertion(+), 1 deletion(-)

2dawv02@A306PC02 MINGW64 ~/Documents/forty (feature-content)
$ git status
On branch feature-content
nothing to commit, working tree clean
```

Verificamos si hay cambios pendientes:

```
$ git status
```

6. Elimina el archivo "passwords.txt" en la carpeta forty . Verifica el estado del repositorio. ¿Hay cambios pendientes?

Voy a la carpeta de forty directamente y borro el archivo. Ahora compruebo que ha pasado con el comando:

```
$ git status
```

7. Crea un nuevo archivo llamado "about.html" , partiendo del archivo generic.html y agrégalo al repositorio, haz un nuevo commit.

Podemos crearlo directamente en la carpeta o desde comandos:

Copiamos el estilo y contenido de genreric.html en about.html

```
$ cp generic.html about.html
```

Hacemos un cambio en el título y lo renombramos con `About` . Esto lo hacemos desde nano

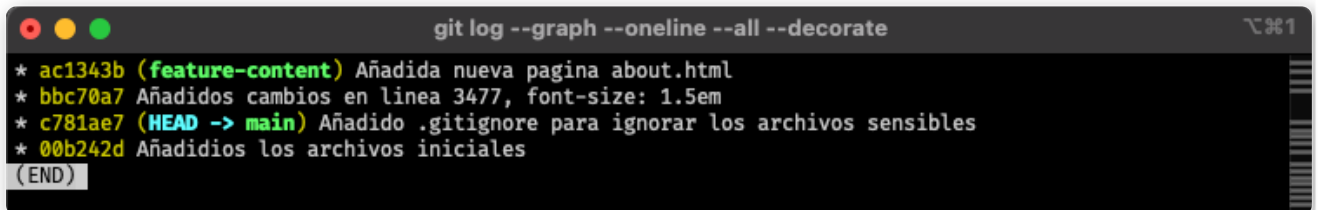
```
$ nano about.html
```

Agregamos este nuevo archivo al repositorio:

```
$ git add about.html  
$ git commit -m "Añadida nueva pagina about.html"
```

8. Cambia a la rama `main` . Examina los logs del repositorio de forma gráfica.

```
$ git checkout main  
$ git log --graph --oneline --all --decorate
```



```
git log --graph --oneline --all --decorate
* ac1343b (feature-content) Añadida nueva pagina about.html
* bbc70a7 Añadidos cambios en linea 3477, font-size: 1.5em
* c781ae7 (HEAD -> main) Añadido .gitignore para ignorar los archivos sensibles
* 00b242d Añadidos los archivos iniciales
(END)
```

9. Modifica algo en el archivo `generic.html` , comprueba que hay cambios, y realiza otro commit . Examina los logs del repositorio de forma gráfica.

Abrimos nano para realizar cambios en `generic.html` :

```
$ nano generic.html
```

Añadimos los cambios y realizamos commit al repositorio:

```
$ git add generic.html  
$ git commit -m "Añadidos cambios en generic.html"  
$ git log --graph --oneline --all --decorate
```

```
git log --graph --oneline --all --decorate
* 2850041 (HEAD -> main) Añadidos cambios en generic.html
| * ac1343b (feature-content) Añadida nueva pagina about.html
| * bbc70a7 Añadidos cambios en línea 3477, font-size: 1.5em
|/
* c781ae7 Añadido .gitignore para ignorar los archivos sensibles
* 00b242d Añadidos los archivos iniciales
(END)
```

10. Modifica algo en el fichero `elements.html` . Confirma los cambios, pero no hagas commit.

Abrimos nano para realizar cambios en `elements.html` :

```
$ nano elements.html
```

Añadimos los cambios pero no los commiteamos:

```
$ git add elements.html
```

Comprobamos que los cambios estan pendientes de confirmación:

```
$ git status
```

```
user: cris forty2 [?] main][+]
09:43:07 > git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   elements.html

user: cris forty2 [?] main][+]
09:43:10 >
```

11. Mira las diferencias de `elements.html` . Los cambios no nos gustan, deshaz los cambios de `elements.html` . Comprueba que no hay cambios pendientes.

Para deshacer los cambios en staging primero debemos deshacer el "git add"

```
$ git restore --staged elements.html
```

Después deshacer la modificacion que hemos hecho del archivo:

```
$ git restore elements.html
```

Por último comprobamos que todo está correcto:

```
$ git status
```



La forma anterior es la mas moderna pero también hay otra que siempre se ha hecho que es:

```
$ git reset HARD elements.html  
$ git checkout --elements.html
```

```
user: cris forty2 [?] main][+]  
09:43:10 > git restore --staged elements.html  
user: cris forty2 [?] main][!]  
09:57:10 > git status  
On branch main  
Changes not staged for commit:  
  (use "git add <file>..." to update what will be committed)  
  (use "git restore <file>..." to discard changes in working directory)  
        modified:   elements.html  
  
no changes added to commit (use "git add" and/or "git commit -a")  
user: cris forty2 [?] main][!]  
09:57:14 > git restore elements.html  
user: cris forty2 [?] main]  
09:57:37 > git status  
On branch main  
nothing to commit, working tree clean
```

## 12. Muestra las diferencias entre dos ramas.

```
$ git diff main feature-content
```

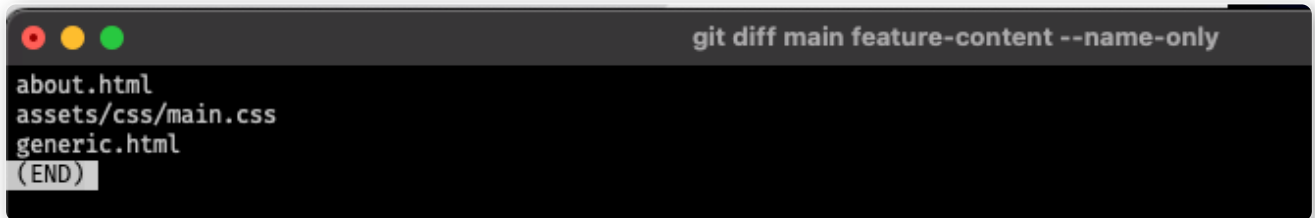
Muestra los cambios que estan en feature-content que no estan en main. En nuestro caso los cambios qu se hicieron en el archivo `about.html`, en `main.css` y en `generic.html`

Para que no se vea tanta información de los archivos cambiados se puede cambiar

el comando para que solo te muestre el nombre los archivos cambiados:  
(Cualquiera de las dos nos da una informacion más resumida)

```
$ git diff main feature-content --name-only
```

```
$ git diff main feature-content --stat
```

A terminal window with a dark background and light text. The title bar at the top right reads 'git diff main feature-content --name-only'. The terminal output lists three files: 'about.html', 'assets/css/main.css', and 'generic.html'. Below these, it shows '(END)' on a new line. The window has standard macOS window control buttons (red, yellow, green) in the top left corner.

```
git diff main feature-content --name-only
about.html
assets/css/main.css
generic.html
(END)
```

13. Fusiona la rama "feature-content" con la rama principal (main). Muestra los logs del repositorio de una forma gráfica y completa.

Primero nos aseguramos de estar en la rama main:

```
$ git checkout main
```

Fusionamos la rama feature-content:

```
$ git merge feature-content
```

Mostrar los logs completos:

```
$ git log --all --decorate --graph
```



```
commit eef94cefccfe78beaa591d6462c5f471ffe28847 (HEAD -> main)
```

```
Date: Fri Oct 17 09:21:41 2025 +0200
```

```
Añadida nueva pagina about.html
```

```
* commit bbc70a7f150e7ce44c2f8429269598a762cb611c
```

```
Author: CrisMuniz <cristinamuniz85@gmail.com>
```

```
Date: Fri Oct 17 09:03:49 2025 +0200
```

```
Añadidos cambios en linea 3477, font-size: 1.5em
```

```
* commit 285004119c2e3f0a1ec56439dc59ea89574b7a1e
```

```
Author: CrisMuniz <cristinamuniz85@gmail.com>
```

```
Date: Fri Oct 17 09:35:54 2025 +0200
```

```
Añadidos cambios en generic.html
```

```
* commit c781ae73d605b5f7a26cf1999044d3656271cd86
```

```
Author: CrisMuniz <cristinamuniz85@gmail.com>
```

```
Date: Fri Oct 17 08:46:05 2025 +0200
```

```
Añadido .gitignore para ignorar los archivos sensibles
```

```
* commit 00b242d50da048959ba40b253fbe0b610b3fa768
```

```
Author: CrisMuniz <cristinamuniz85@gmail.com>
```

```
Date: Fri Oct 17 08:36:02 2025 +0200
```

```
Añadidos los archivos iniciales
```

```
* commit ac1343b5dd4ab7dad4ae5e6be0755cfdac847c4 (feature-content)
```

```
Author: CrisMuniz <cristinamuniz85@gmail.com>
```

```
Date: Fri Oct 17 09:21:41 2025 +0200
```

```
Añadida nueva pagina about.html
```

```
* commit bbc70a7f150e7ce44c2f8429269598a762cb611c
```

```
Author: CrisMuniz <cristinamuniz85@gmail.com>
```

```
Date: Fri Oct 17 09:03:49 2025 +0200
```

```
Añadidos cambios en linea 3477, font-size: 1.5em
```

```
* commit 285004119c2e3f0a1ec56439dc59ea89574b7a1e
```

```
Author: CrisMuniz <cristinamuniz85@gmail.com>
```

```
Date: Fri Oct 17 09:35:54 2025 +0200
```

```
Añadidos cambios en generic.html
```

```
* commit c781ae73d605b5f7a26cf1999044d3656271cd86
```

```
Author: CrisMuniz <cristinamuniz85@gmail.com>
```

```
Date: Fri Oct 17 08:46:05 2025 +0200
```

```
Añadido .gitignore para ignorar los archivos sensibles
```

```
* commit 00b242d50da048959ba40b253fbe0b610b3fa768
```

```
Author: CrisMuniz <cristinamuniz85@gmail.com>
```

```
Date: Fri Oct 17 08:36:02 2025 +0200
```

```
Añadidos los archivos iniciales
```



Cuando hacemos el merge en Mac nos sale una pantalla de vim para que pongamos un mensaje al merge que queremos realizar. Por lo general ya te pone un mensaje predeterminado.

Para aceptar ese mensaje tenemos que realizar unos pasos y así poder salir de la pantalla:

- Presionamos ESC
- Escribimos `:wq`
- Presionamos Enter

Este merge no da errores porque se han hecho cambios diferentes en cada rama pero no coinciden que se hallan hecho cambios en ambas ramas en el mismo archivo en el mismo sitio.

14. Crea una nueva rama llamada `"hotfix"` y en ella, corrige un error crítico en el archivo `"index.html"`. (Por ejemplo, añade el enlace a la nueva página `about.html`)

Creemos la rama y nos cambiamos a ella:

```
$ git branch hotfix
$ git checkout hotfix
```

Entramos con nano en el archivo y hacemos el cambio que nos indican

```
$ nano index.html
```

Añadimos los cambios y hacemos el commit

```
$ git add index.html
$ git commit -m "Añadido enlace a about.html en index.html"
$ git status
```

15. Fusiona la rama `"hotfix"` con la rama principal y verifica el historial de commits de forma que se vean todas las ramas gráficamente. ¿Borrarías la rama `hotfix`? ¿En qué caso? ¿Cómo?

Nos cambiamos a la rama main:

```
$ git checkout main
```

Fusionamos las ramas:

```
$ git merge hotfix
```

Mostramos el historial de los logs:

```
$ git log --all --decorate --oneline --graph
```

```
* 6d83217 (HEAD -> main, hotfix) Añadido enlace a about.html en index.html
* eef94ce erge branch 'feature-content' this merge is necessary,
|\
| * ac1343b (feature-content) Añadida nueva pagina about.html
| * bbc70a7 Añadidos cambios en linea 3477, font-size: 1.5em
| | 2850041 Añadidos cambios en generic.html
|/
* c781ae7 Añadido .gitignore para ignorar los archivos sensibles
* 00b242d Añadidos los archivos iniciales
(END)
```

Borraría la rama porque ya se fusionó y no es necesaria ya que era una corrección rápida de un error crítico. Esto evita que halla errores por ver muchas ramas de este tipo, una vez solucionado el problema ya no hacen falta.

```
$ git branch -d hotfix
```

Podemos verificar que se ha borrado con:

```
$ git branch
```

16. Muestra el historial de cambios limitado a los últimos 3 commits.

```
$ git log -3 --oneline
```

```
6d83217 (HEAD -> main) Añadido enlace a about.html en index.html
eef94ce erge branch 'feature-content' this merge is necessary,
2850041 Añadidos cambios en generic.html
(END)
```

17. Etiqueta el commit actual como "v1.0" y muestra las etiquetas existentes.

```
$ git tag -a v1.0 -m "Versión 1.0 del proyecto Forty"
```

O de forma mas simple:

```
$ git tag v1.0
```

Para visualizar que se ha realizado la etiqueta con éxito lo comprobamos con uno de estos dos comandos:

```
$ git tag //Nos da la versión (v1.0)
$ git show v1.0 // Nos muestra mas detalles incluido el comentario
```

## Trabajo en remoto



Antes de nada es necesario crear un repositorio en GitHub llamado "forty" y no debemos crear el README.md en el remoto. Una vez creado esto se nos dará una URL con la dirección de nuestro nuevo repositorio.

### 1. Sube al remoto los ficheros de tu repositorio local

Una vez creado nuestro repositorio llamando `forty`:

- Conectamos nuestro local con el remoto desde nuestra consola

```
$ git remote add origin
https://github.com/CrisMunizMarin/Forty.git
```

- Subimos los archivos al repositorio

```
$ git push -u origin main
```

- Subimos la tag que habíamos creado

```
$ git push origin --tags
```

2. En local, crea una rama 'feature-head'. Cambia el título en la sección del head de index.html, borra los comentarios del head, o previos, también. Confirma y sube los cambios al remoto.

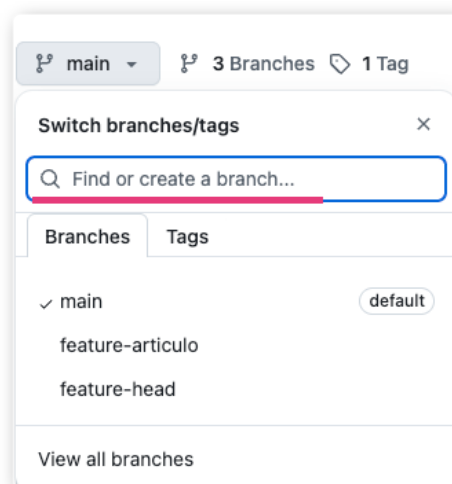
```
$ git checkout -b feature-head
```

- Ahora realizamos los cambios en el index.html
- Añadimos los cambios y hacemos commit y también subimos la nueva rama al remoto

```
$ git add index.html  
$ git commit -m "Cambios en el index.html: Borrado de Title  
y comentarios previos al HEAD"  
$ git push -u origin feature-head
```

3. En remoto, crea una rama 'feature-articulo'. Duplica la página generic, nómbrala como articulo.html, y añade como contenido un artículo sobre Git. Confirma los cambios y realiza un commit. Muestra los commits del repositorio tal como se ven en GitHub.

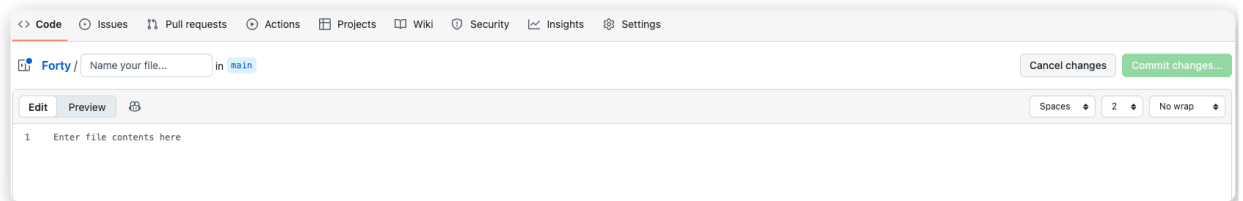
- Nos vamos a GitHub y creamos una nueva rama



- Creamos articulo.html:

Vamos al archivo genérico.html, hacemos click en RAW y copiamos todo el contenido. Vamos a la vista principal y añadimos un fichero y lo llamamos articulo.html y pegamos lo que hemos copiado.

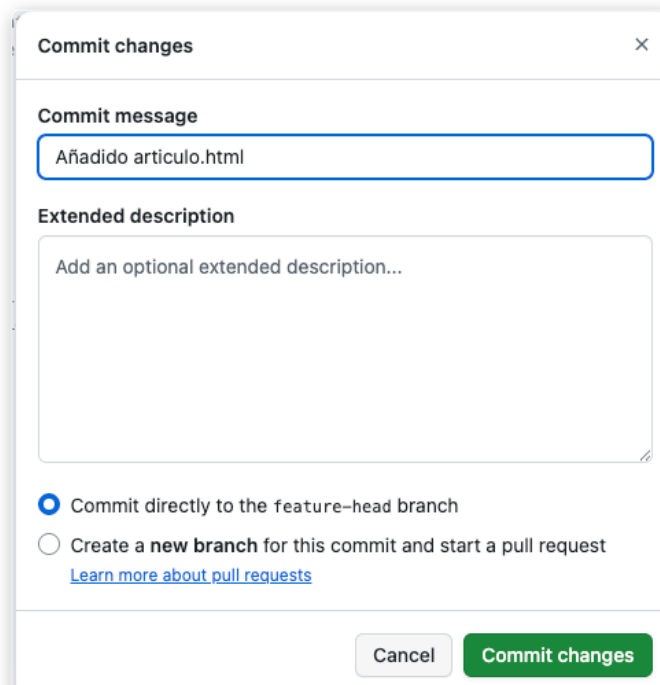
Add file > Create new file < introducimos lo que queremos y lo nombramos



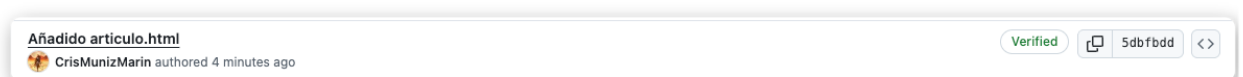
- Hacemos el cambio que nos piden



- Confirmamos los cambios desde el propio GitHub



- Commits



4. En el repositorio local examina los cambios. Actualiza el repositorio con el remoto. Fusiona en 'main' las dos ramas 'feature'. Crea la etiqueta 'v2.0'. Muestra los logs,

commits, etiquetas y ramas actuales, en local y en remoto

- volvemos a main

```
$ git checkout main
```

- Nos traemos los cambios del remoto al local

```
$ git pull
```

- fusionamos las ramas a main

```
$ git merge feature-head  
$ git merge feature-articulo
```

- Creamos etiquetas v2.0

```
$ git tag -a v2.0 -m "Version 2.0 con nuevas  
funcionalidades de head y articulo"
```

- Mostramos los logs en local y remoto

```
$ git log --all --decorate --oneline --graph
```

```
* 5dbfbdd (origin/feature-head) Añadido articulo.html  
* 928dd2e (HEAD -> main, tag: v2.0, feature-head) Cambios en el index.html: Borrado de Title y comentarios previos al HEAD  
* 6d83217 (tag: v1.0, origin/main, origin/feature-articulo, feature-articulo) Añadido enlace a about.html en index.html  
* eef94ce erge branch 'feature-content' this merge is necessary,  
/ \  
* ac1343b (feature-content) Añadida nueva pagina about.html  
* bbc70a7 Añadidos cambios en linea 3477, font-size: 1.5em  
* | 2850041 Añadidos cambios en generic.html  
/ \  
* c781ae7 Añadido .gitignore para ignorar los archivos sensibles  
* 00b242d Añadidos los archivos iniciales  
END)
```

```
$ git tag
```

```
v1.0  
v2.0  
END)
```

```
$ git branch -a
```

```
feature-articulo  
feature-content  
feature-head  
* main  
  remotes/origin/feature-articulo  
  remotes/origin/feature-head  
  remotes/origin/main  
(END)
```

5. En tu copia local, crea una rama nueva. En la rama nueva, cambia los enlaces de la página index.html para que apunten correctamente a la nueva página cambios.

- Crear rama correccion-enlaces

```
$ git checkout -b correccion-enlaces
```

- Hacemos cambios en index.html (En mi caso he sustituido el enlace que apuntaba a generic.html por articulo.html)

```
<!-- Menu -->  
<nav id="menu">  
  <ul class="links">  
    <li><a href="index.html">Home</a></li>  
    <li><a href="landing.html">Landing</a></li>  
    <li><a href="articulo.html">Articulo</a></li>  
    <li><a href="elements.html">Elements</a></li>  
    <li><a href="about.html">About</a></li>  
  </ul>  
  <ul class="actions stacked">  
    <li><a href="#" class="button primary fit">Get Started</a></li>  
    <li><a href="#" class="button fit">Log In</a></li>  
  </ul>  
</nav>
```

- Añadimos los cambios y realizamos el commit

```
$ git add index.html
```

```
$ git commit -m "Sustitución de enlace generic.html por  
articulo.html"
```

6. Muestra los logs de forma que se vean las ramas en tu copia local.

- Mostramos los logs en local



```
$ git log --all --decorate --oneline --graph
```

```
* 4853a28 (HEAD -> correccion-enlaces) Sustitución de enlace generic.html por articulo.html
| * 5dbfbdd (origin/feature-head) Añadido articulo.html
|/
* 928dd2e (tag: v2.0, main, feature-head) Cambios en el index.html: Borrado de Title y comentarios previos al HEAD
* 6d83217 (tag: v1.0, origin/main, origin/feature-articulo, feature-articulo) Añadido enlace a about.html en index.html
|
| * eef94ce erge branch 'feature-content' this merge is necessary,
|/
| * ac1343b (feature-content) Añadida nueva pagina about.html
| * bbc70a7 Añadidos cambios en línea 3477, font-size: 1.5em
| * 2850041 Añadidos cambios en generic.html
|/
* c781ee7 Añadido .gitignore para ignorar los archivos sensibles
* 00b242d Añadidos los archivos iniciales
(END)
```

- Volvemos a la rama main

```
$ git checkout main
```

Nos avisa que tenemos que subir los cambios que hemos realizado con la rama corrección-enlaces

7. Te gusta el resultado de los cambios. Incorpora los cambios de la rama nueva a la principal.

- Fusionamos las ramas

```
$ git merge correccion-enlaces
```

```
user: cris forty2 [main] took 10s
10:14:24 > git merge correccion-enlaces
Updating 928dd2e..4853a28
Fast-forward
 index.html | 5 +++--
 1 file changed, 3 insertions(+), 2 deletions(-)
user: cris forty2 [main]:1
```

8. Sube los cambios al remoto borrando la rama nueva, si es necesario. Comprueba primero con un comando en local, las ramas que hay en el repositorio remoto.

- Subimos los cambios a la rama main del remoto

```
$ git push
```

- Comprobamos las ramas que tenemos en remoto para verificar que se han subido correctamente

```
$ git branch -r
```

```
origin/feature-articulo
origin/feature-head
origin/main
(END)
```

Aquí como se puede ver he cometido un error ya que no había subido la rama al remoto con `git push -u origin corrección-enlaces` por lo que no aparece en remoto pero sí en local, así que solo borraré la rama en local y podré continuar con el ejercicio.

```
correccion-enlaces
feature-articulo
feature-content
feature-head
* main
(END)
```

```
$ git branch -d correccion-enlaces
```

```
feature-articulo
feature-content
feature-head
* main
(END)
```

```
user: cris forty2 [?]main]took 1m10s
🕒10:44:38 > git branch -d correccion-enlaces
Deleted branch correccion-enlaces (was 4853a28).

user: cris forty2 [?]main]
🕒10:44:58 > git branch
```

9. Muestra en local los cambios en el archivo `index.html` entre la versión actual y la anterior.

```
$ git diff HEAD~1 HEAD index.html
```

```

diff --git a/index.html b/index.html
index 02d8d99..e880499 100644
--- a/index.html
+++ b/index.html
@@ -26,9 +26,10 @@
                                     <ul class="links">
                                         <li><a href="index.html">Home</a></li>
                                         <li><a href="landing.html">Landing</a></li>
                                         <li><a href="generic.html">Generic</a></li>
                                         <li><a href="articulo.html">Articulo</a></li>
                                         <li><a href="elements.html">Elements</a></li>
                                         <li><a href="about.html">About</a></li>
                                         </ul>
                                     <ul class="actions stacked">
                                         <li><a href="#" class="button primary fit">Get Started</a></li>
                                     </ul>
                                     (END)

```

- En el repositorio en GitHub, navega hasta el archivo index.html y selecciona la opción "History".

History for [Forty](#) / [index.html](#) on [main](#)

[All users](#) [All time](#)

Commits on Oct 24, 2025

Sustitución de enlace generic.html por articulo.html

 CrisMuniz committed 50 minutes ago

4853a28 [Copy](#) [Diff](#) [View](#)

Commits on Oct 23, 2025

Cambios en el index.html: Borrado de Title y comentarios previos al HEAD

 CrisMuniz committed yesterday

928dd2e [Copy](#) [Diff](#) [View](#)

Commits on Oct 17, 2025

Añadido enlace a about.html en index.html

 CrisMuniz committed last week

6d83217 [Copy](#) [Diff](#) [View](#)

Añadidos los archivos iniciales

 CrisMuniz committed last week

00b242d [Copy](#) [Diff](#) [View](#)

End of commit history for this file