

GRUPO 5

USO DE HERRAMIENTAS DE CONTROL DE VERSIONES

Martina Vásconez, Sthefano Ulloa, Gabriela Coloma

Control de versiones

¿Qué es?

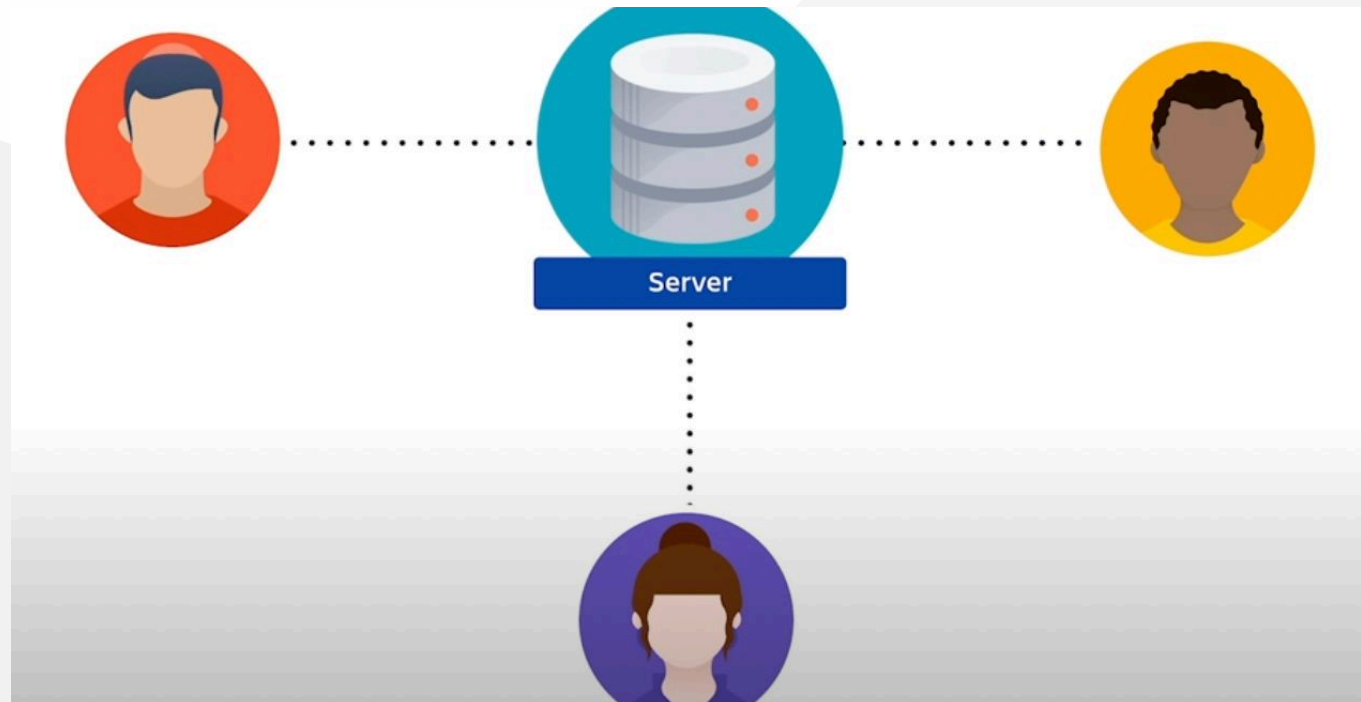
Es una herramienta en el desarrollo del software que te permite registrar cambios en tu archivo a lo largo del tiempo



Categorías principales

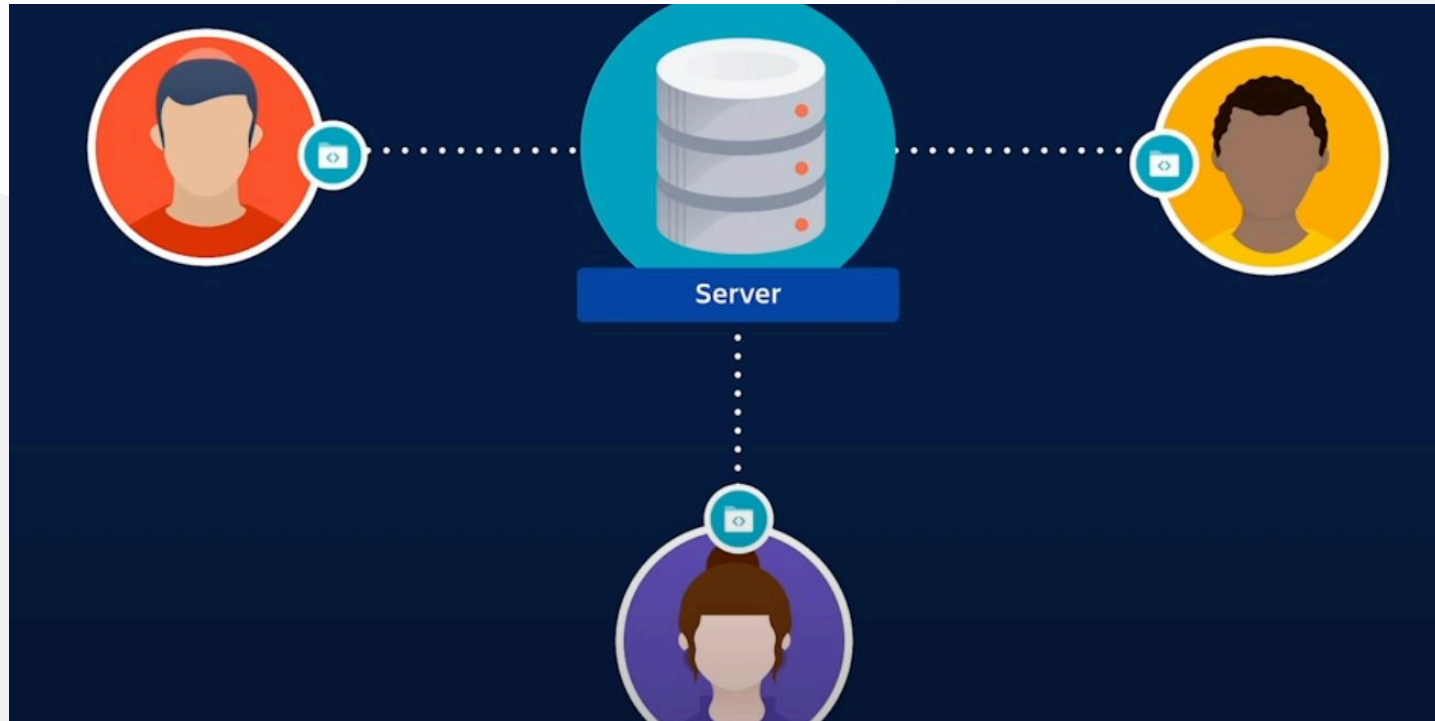
Centralizado

Existe un único repositorio central que almacena toda el proyecto



Distribuido

Cada usuario tiene su propio repositorio local en donde pueden trabajar de forma independiente



Dos etapas

Working

El directorio de trabajo es donde realizas todas las modificaciones a tus archivos

Staging

El área de preparación, también conocida como "índice", es un área intermedia donde se registran los cambios antes de confirmarlos realmente en la historia del proyecto

Funciones

Seguimiento de cambios

Listado en el que se guarda qué usuario ha modificado en el documento

Historial de versiones

Almacena todas las versiones del documento y puedes regresar a cualquiera

Notificación de cambios

Avisa a los usuarios sobre cualquier modificación que se haya hecho.

Comparación de documentos

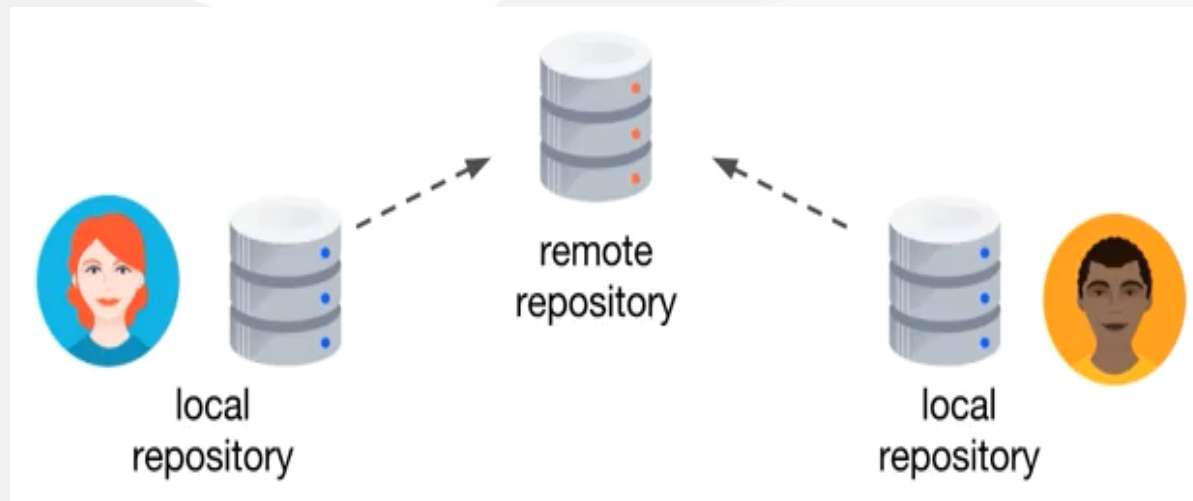
Permite detectar qué ha sido cambiado de una versión a otra, línea por línea.

Experimental

Pueden sacar una sección del documento para modificar algo y probar si sirve. Lo puedes guardar o simplemente dejar la versión anterior.

Repositorio

Un repositorio en Git es un lugar donde se almacenan todos los archivos y carpetas de un proyecto, junto con el historial completo de cambios realizados en esos archivos a lo largo del tiempo. Es como una base de datos que registra todas las modificaciones realizadas en el código fuente de un proyecto.



Utilizando un repositorio existente

Si el repositorio ya existe, necesitamos clonarlo en nuestra máquina local:

```
git clone <URL_del_repositorio>
```

(Creamos un repositorio en github con esta guía)



Crear un nuevo repositorio

Comencemos creando un nuevo repositorio y agregando algunos archivos a él.

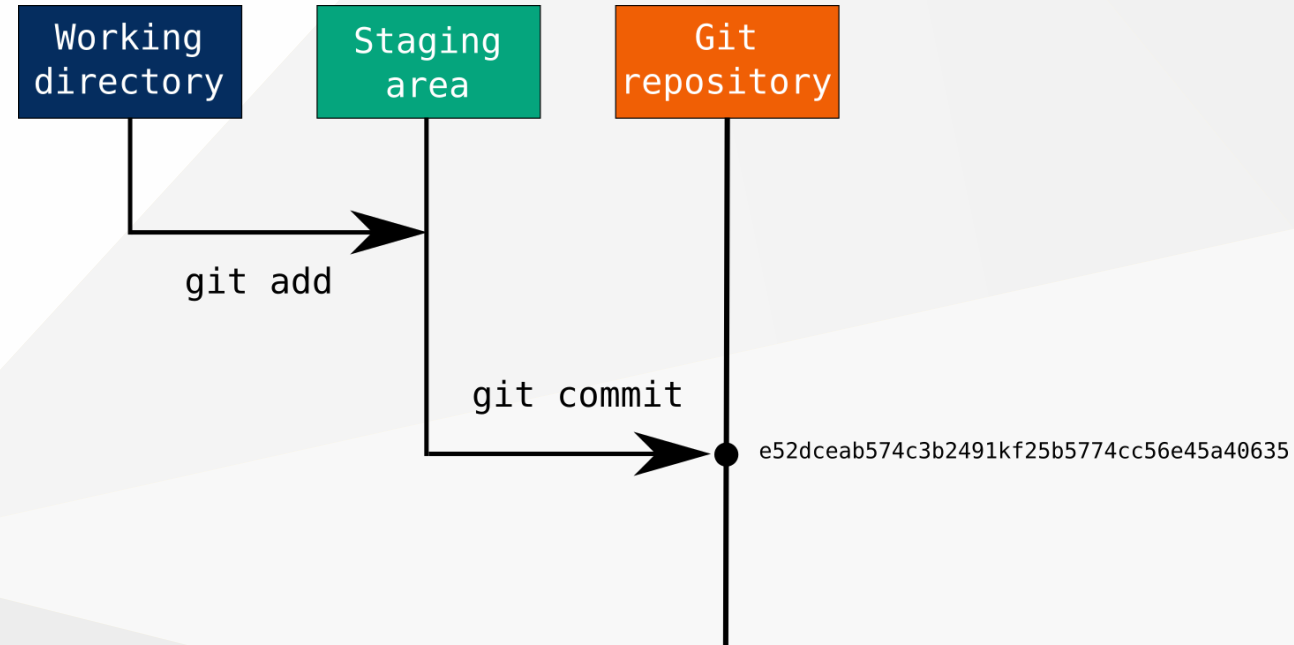
```
mkdir my_project  
cd my_project  
git init
```

Ahora, creemos un nuevo archivo de texto y agreguemos algo de contenido.

```
echo "This is a sample text file." > sample.txt
```

Realizando Cambios

Al realizar cambios en los archivos de un repositorio, es importante seguir ciertos pasos para registrar y confirmar esos cambios de manera adecuada. Estos pasos aseguran que los cambios se registren correctamente en el historial del repositorio.



A continuación, hagamos un seguimiento de los cambios que hicimos y los confirmemos en el repositorio.

```
git status
```

```
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git status (base)
On branch master
friendly interactive shell
No commits yet
D/D/U/s/p/g/version_control> 
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    README.md
    sample.txt

nothing added to commit but untracked files present (use "git add" to track)
```

```
git add sample.txt  
git commit -m "Added sample.txt"
```

```
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git commit -m "Added sample.txt"
```

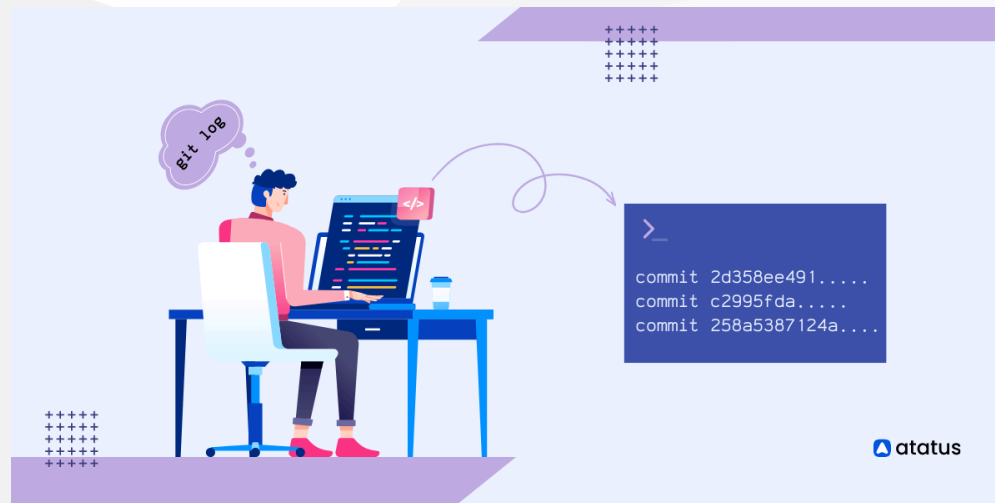
```
[master (root-commit) 6e3197a] Added sample.txt  
1 file changed, 1 insertion(+)  
create mode 100644 sample.txt
```

Ahora, modifiquemos el contenido de `sample.txt`. Observa la bandera `-am`, donde `a` agrega los archivos trackeados para el commit.

```
echo "This is a modified content." >> sample.txt  
git commit -am "Modified sample.txt"
```

Visualización del Historial

El historial de un repositorio muestra todos los commits realizados, junto con información sobre quién hizo cada cambio, qué se cambió y cuándo se realizó. Esto permite a los desarrolladores revisar el progreso del proyecto a lo largo del tiempo y entender cómo ha evolucionado el código fuente.



Podemos ver el historial de commits para ver nuestros cambios.

```
git log
```

```
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git log (base)
commit e13a45d1b5a9aadeef266adee98338d6d75215b23 (HEAD -> master)
Author: nitou2504 <sthefanou25@gmail.com>
Date: Sun Apr 28 11:48:50 2024 -0500

    Modified sample.txt

commit 6e3197a4484b4b872861204af703ea62ef3d3b63
Author: nitou2504 <sthefanou25@gmail.com>
Date: Sun Apr 28 11:46:28 2024 -0500

    Added sample.txt
```

Para una vista concisa:

```
git log --oneline
```

```
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git log --oneline
e13a45d (HEAD -> master) Modified sample.txt
6e3197a Added sample.txt
```


Mostrando cambios en un commit con git show.

```
git show <commit_hash>
```

```
sthefano@endevour /m/D/D/U/s/p/g/version_control(master)> git show e13a45d
commit e13a45d1b5a9aadeef266adee98338d6d75215b23 (HEAD -> master) Modified sample.txt
Author: nitou2504 <sthefanou25@gmail.com>
Date: Sun Apr 28 11:48:50 2024 -0500
```

```
    Modified sample.txt

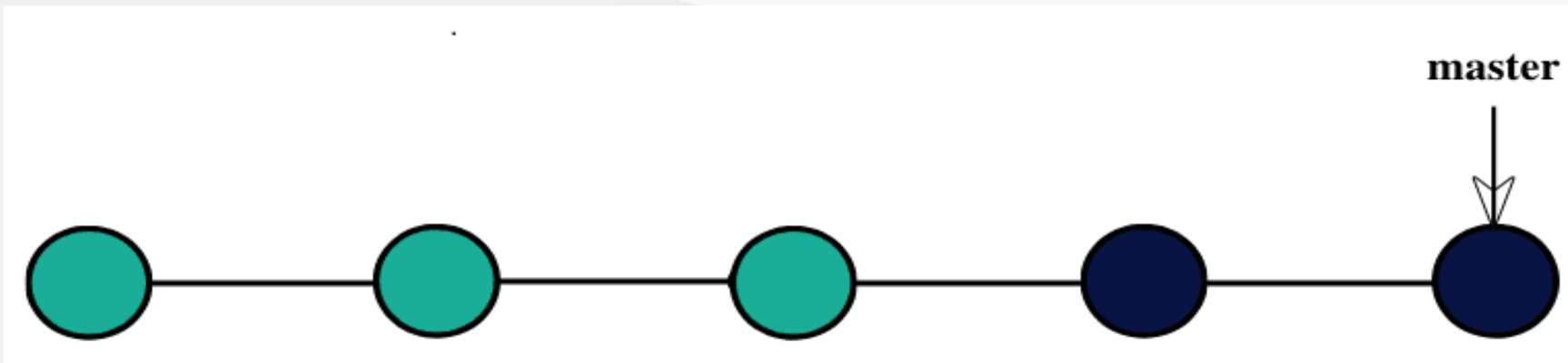
diff --git a/sample.txt b/sample.txt
index 6f3a977..70c62ce 100644
--- a/sample.txt
+++ b/sample.txt
@@ -1,2 @@
-This is a sample text file.
+This is a modified content.
```

Displaying changes in a commit with git show.

```
git show <commit_hash>
```

Branching

Las ramas en Git permiten a los desarrolladores trabajar en nuevas características o experimentos sin afectar el código principal del proyecto. Cada rama representa una línea independiente de desarrollo y puede contener diferentes conjuntos de cambios. Las ramas se utilizan para organizar el trabajo y facilitar la colaboración entre varios miembros del equipo.



Creemos una nueva rama para una característica nueva.

```
git branch feature_branch  
git checkout feature_branch
```

```
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git checkout feature_branch  
Switched to branch 'feature_branch'  
sthefano@endevour /m/D/D/U/s/p/g/version_control (feature_branch)> █ (base)
```

Ahora, hagamos algunos cambios en la rama de la característica.

```
echo "This is a feature branch change." >> sample.txt  
git commit -am "Feature branch change in sample.txt"
```

Cambiamos de nuevo a la rama principal.

```
git checkout master
```

```
sthefano@endevour /m/D/D/U/s/p/g/version_control (feature_branch) [1]> git checkout master  
Switched to branch 'master'  
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> (base)
```

Fusionemos los cambios de la rama de la característica en la rama principal.

```
git merge feature_branch
```

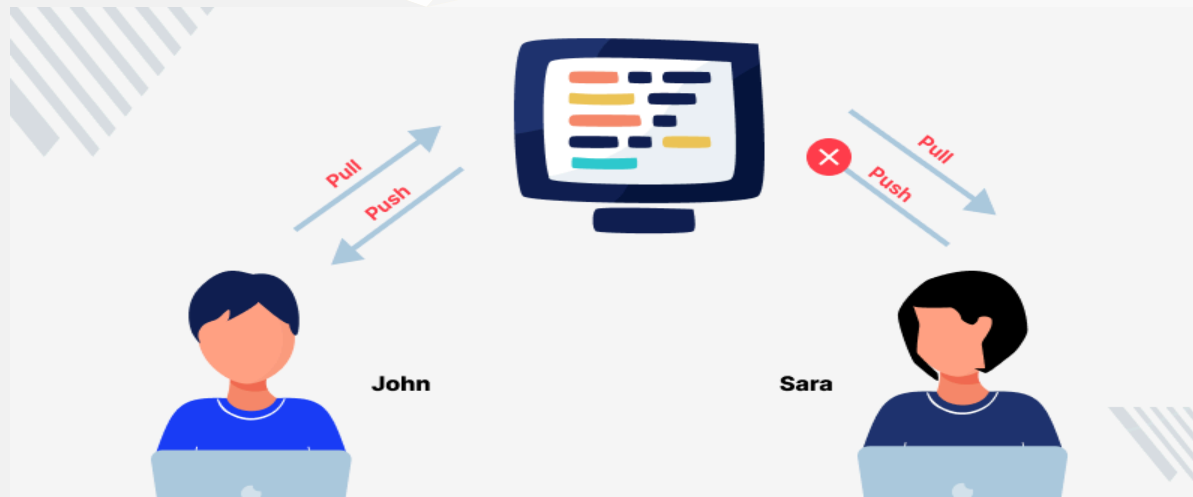
```
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git merge feature_branch (base)
Updating e13a45d..19c3e96
Fast-forward
 sample.txt | 1 +
 1 file changed, 1 insertion(+)
sthefano@endevour /m/D/D/U/s/p/g/version_control (feature_branch)> (base)
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git log (base)
commit 19c3e9642bc459c527ee516e1adb4119619b00c6 (HEAD -> master, feature_branch)
Author: nitou2504 <sthefanou25@gmail.com>
Date: Sun Apr 28 11:59:12 2024 -0500

    Feature branch change in sample.txt

commit e13a45d1b5a9aadeef266adee98338d6d75215b23
Author: nitou2504 <sthefanou25@gmail.com>
Date: Sun Apr 28 11:48:50 2024 -0500
```

Merge conflict

Un conflicto de fusión ocurre cuando Git no puede combinar automáticamente los cambios de dos ramas debido a modificaciones conflictivas en el mismo archivo o línea de código. Es necesario resolver el conflicto manualmente editando el archivo afectado y eligiendo qué cambios mantener. Una vez resuelto el conflicto, se pueden agregar los cambios y confirmar la fusión.



Para simular un conflicto de fusión, primero creemos una nueva rama.

```
git branch conflict_branch  
git checkout conflict_branch
```

Ahora, modifiquemos `sample.txt` en la rama de conflicto.

```
echo "This is a change in the conflict branch." >> sample.txt  
git commit -am "conflict change in sample.txt"
```

```
sthefano@endevour /m/D/D/U/s/p/g/version_control (conflict_branch)> git log (base)
commit 7c14fb82c53e1f10f7993a941e834b617d9d55b8 (HEAD -> conflict_branch)
Author: nitou2504 <sthefanou25@gmail.com>
Date: Sun Apr 28 12:04:36 2024 -0500
    conflict change in sample.txt

commit 19c3e9642bc459c527ee516e1adb4119619b00c6 (master, feature_branch)
Author: nitou2504 <sthefanou25@gmail.com>
Date: Sun Apr 28 11:59:12 2024 -0500
    Feature branch change in sample.txt

commit e13a45d1b5a9aadeef266adee98338d6d75215b23
Author: nitou2504 <sthefanou25@gmail.com>
Date: Sun Apr 28 11:48:50 2024 -0500
    Feature branch change in sample.txt

commit 6e3197a4484b4b872861204af703ea62ef3d3b63
Author: nitou2504 <sthefanou25@gmail.com>
Date: Sun Apr 28 11:46:28 2024 -0500
    Added sample.txt
```


Cambiamos de nuevo a la rama principal.

```
git checkout master
```

Observa que el último cambio en `master` fue de la fusión de la rama anterior:

```
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git log (base)
commit 19c3e9642bc459c527ee516e1adb4119619b00c6 (HEAD -> master, feature_branch)
Author: nitou2504 <sthefanou25@gmail.com>
Date: Sun Apr 28 11:59:12 2024 -0500
    Feature branch change in sample.txt
```

Ahora

, modifiquemos la misma línea en `sample.txt` en la rama principal.

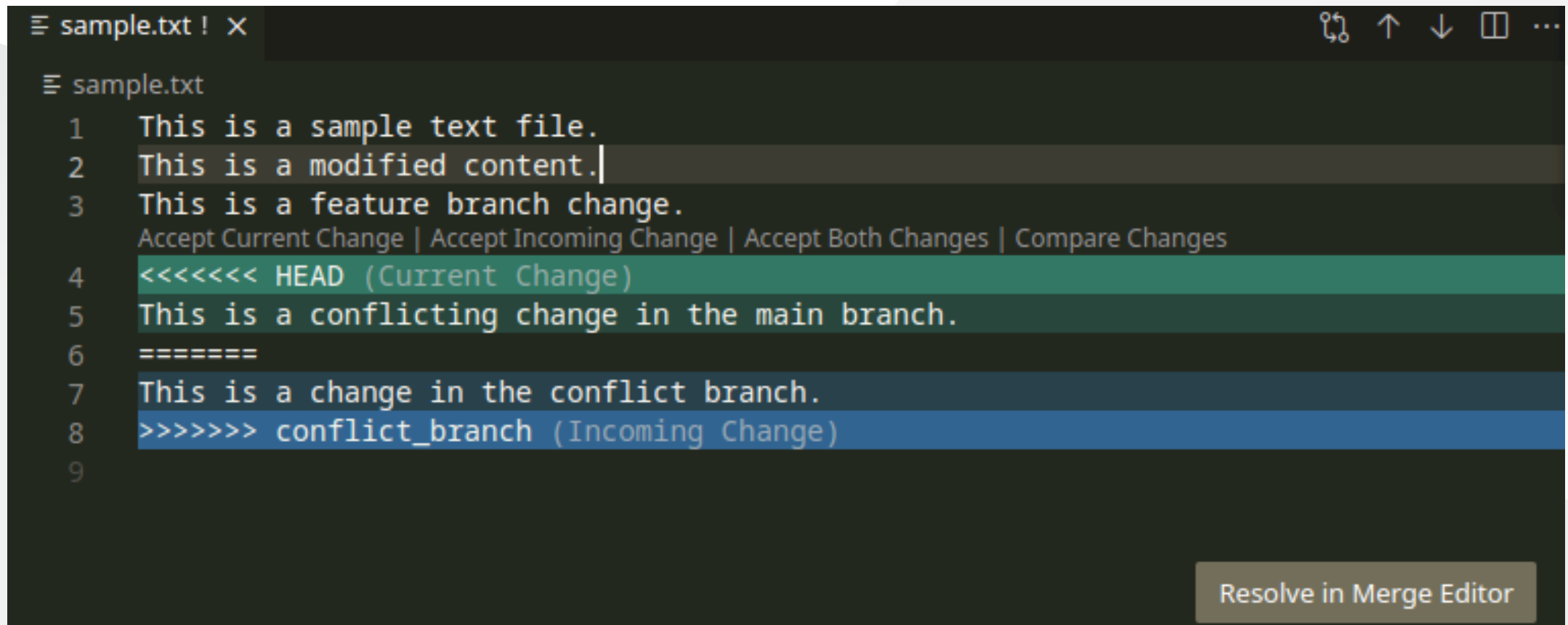
```
echo "This is a conflicting change in the main branch." >> sample.txt  
git commit -am "conflict change (master) in sample.txt"
```

Ahora, intenta fusionar la rama de conflicto en la rama principal.

```
git merge conflict_branch
```

```
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git merge conflict_branch(base)  
Auto-merging sample.txt  
CONFLICT (content): Merge conflict in sample.txt  
Automatic merge failed; fix conflicts and then commit the result.
```

Encontrarás un conflicto de fusión. Deberás resolverlo manualmente editando el archivo `muestra.txt`, eliminando los marcadores de conflicto y conservando los cambios deseados. Después de resolver el conflicto, agrega y confirma los cambios:



```
sample.txt ! X
sample.txt
1 This is a sample text file.
2 This is a modified content.|
3 This is a feature branch change.
  Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
4 <<<<<< HEAD (Current Change)
5 This is a conflicting change in the main branch.
6 =====
7 This is a change in the conflict branch.
8 >>>>>> conflict_branch (Incoming Change)
9

Resolve in Merge Editor
```

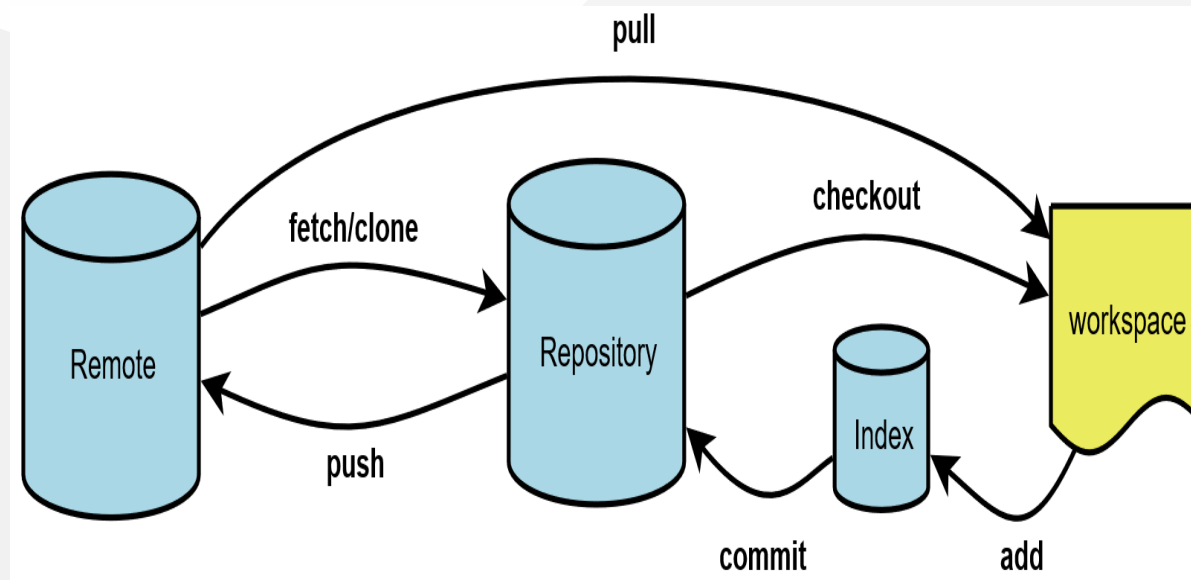
```
git add sample.txt
git commit -m "Resolve merge conflict"
```

```
sthefano@endevour /m/D/D/U/s/p/g/version_control (master|MERGING)> git commit -m
"Resolve merge conflict"
[master 576333e] Resolve merge conflict
Merge: ae6ac78 7c14fb8
Author: nitou2504 <sthefanou25@gmail.com>
Date: Sun Apr 28 12:24:17 2024 -0500
    This is a feature branch change
    Resolve merge conflict
    This is a conflicting change in the main branch
commit ae6ac7827c027b80e4bfe2d06bbdd875c88aba7f
Author: nitou2504 <sthefanou25@gmail.com>
Date: Sun Apr 28 12:09:25 2024 -0500
    conflict change (master) in sample.txt
```

Resolved in Merge Editor

Repositorios Remotos

Los repositorios remotos en Git son versiones de un proyecto alojadas en servidores en línea, como GitHub o GitLab. Estos repositorios permiten a los desarrolladores colaborar en un proyecto compartiendo sus cambios de forma remota.



Agregar un repositorio remoto con git remote add.

```
git remote add origin <URL_del_repositorio>
```

Si ya tienes cambios en tu git local y quieres enviarlos a un nuevo repositorio remoto vacío:

```
git push -u origin master
```

```
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git remote add origin
git@github.com:nitou2504/version_control.git
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git push -u origin mas
ter
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 8 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (16/16), 1.36 KiB | 139.00 KiB/s, done.
Total 16 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), done.
To github.com:nitou2504/version_control.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

Releases

No releases published
[Create a new release](#)

Packages

[Publish your first package](#)

Enviar cambios a un repositorio remoto con git push.

```
git push origin <nombre_rama>
```

Traer cambios de un repositorio remoto con git pull.

```
git pull origin <nombre_rama>
```

Consejos y Trucos Útiles

Alias para comandos comunes.

Puedes configurar alias en tu archivo `~/.gitconfig`:

```
git config --global alias.lg "log --oneline"
```

```
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git lg
576333e (HEAD -> master, origin/master) Resolve merge conflict
ae6ac78 conflict change (master) in sample.txt
7c14fb8 (conflict_branch) conflict change in sample.txt
19c3e96 (feature_branch) Feature branch change in sample.txt
e13a45d Modified sample.txt
6e3197a Added sample.txt
```


Usar .gitignore para ignorar archivos.

Crea un archivo `.gitignore` en el directorio raíz de tu repositorio y enumera los archivos o patrones que deseas ignorar. Esto se hace comúnmente para binarios como imágenes, ya que aparecerán en los mensajes de estado:

```
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  README.md
  image-1.png
  image-10.png
  image-11.png
  image-12.png
  image-13.png
  image-14.png
  image-2.png
```

```
echo "*.png" > .gitignore  
git add .gitignore
```

```
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git status
```

On branch master

Your branch is up to date with 'origin/master'.

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: .gitignore

Untracked files:

(use "git add <file>..." to include in what will be committed)

README.md

Arreglar el último commit con `amend`

`--amend` es una bandera útil para usar cuando hacemos un commit pero olvidamos agregar archivos o cambios (o corregir errores tipográficos en el mensaje de commit). Por ejemplo, olvidamos agregar el README.md en nuestro último commit:

```
git commit -am "added gitignore"
```

```
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git lg
0d4aaae (HEAD -> master) added gitignore
576333e (origin/master) Resolve merge conflict
ae6ac78 conflict change (master) in sample.txt the forgotten file staged, y
7c14fb8 (conflict_branch) conflict change in sample.txt
19c3e96 (feature_branch) Feature branch change in sample.txt
e13a45d Modified sample.txt Message ChitGPT..
6e3197a Added sample.txt
```

Necesitamos preparar el archivo olvidado (agregarlo al staging) y enmendar el commit:

```
git add README.md  
git commit --amend -m "added gitignore and readme"
```

```
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git lg  
66cc484 (HEAD -> master) added gitignore and readme  
576333e (origin/master) Resolve merge conflict  
ae6ac78 conflict change (master) in sample.txt  
7c14fb8 (conflict_branch) conflict change in sample.txt  
19c3e96 (feature_branch) Feature branch change in sample.txt  
e13a45d Modified sample.txt  
6e3197a Added sample.txt
```

Así, nuestro último commit fue corregido.

Saltar entre versiones antiguas y actuales con checkout

`<hash_del_commit>` y checkout `<nombre_rama>`.

```
git checkout <hash_del_commit> # Para cambiar a un commit antiguo  
git checkout <nombre_rama> # Para cambiar de nuevo a la rama actual
```

Recursos Útiles

Guía corta de git



Cheatsheet de git (de Github)

