GRUPO 5

USO DE HERRAMIENTAS DE CONTROL DE VERSIONES

Martina Vásconez, Sthefano Ulloa, Gabriela Coloma

Control de versiones

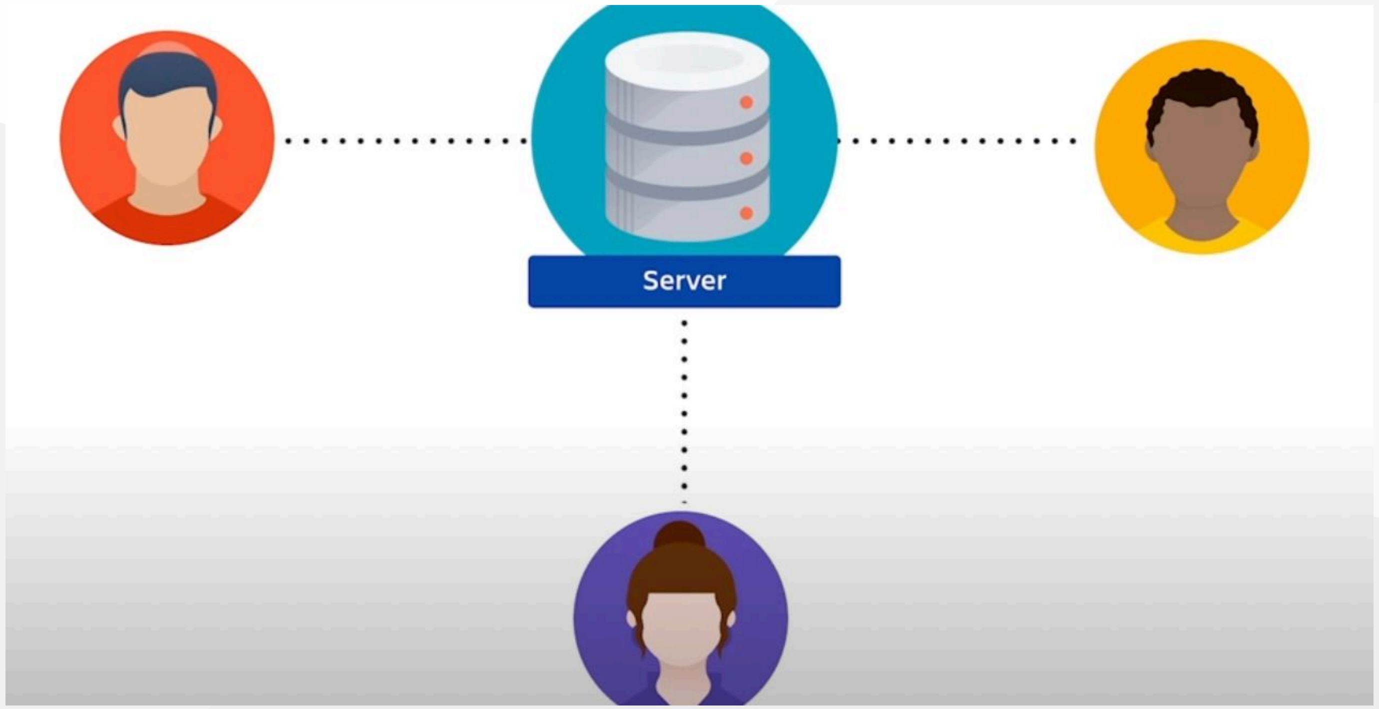
¿Qué es?

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

Categorías principales

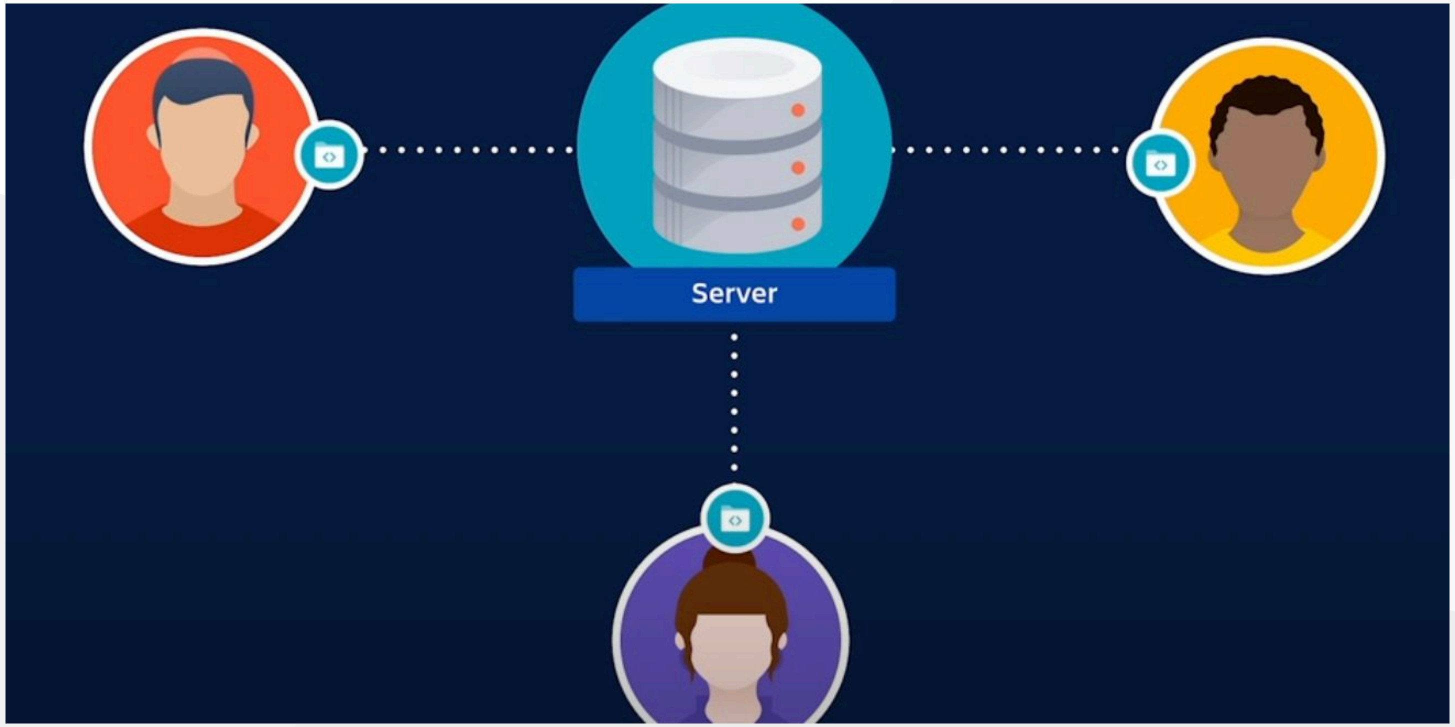
Centralizado

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 que 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

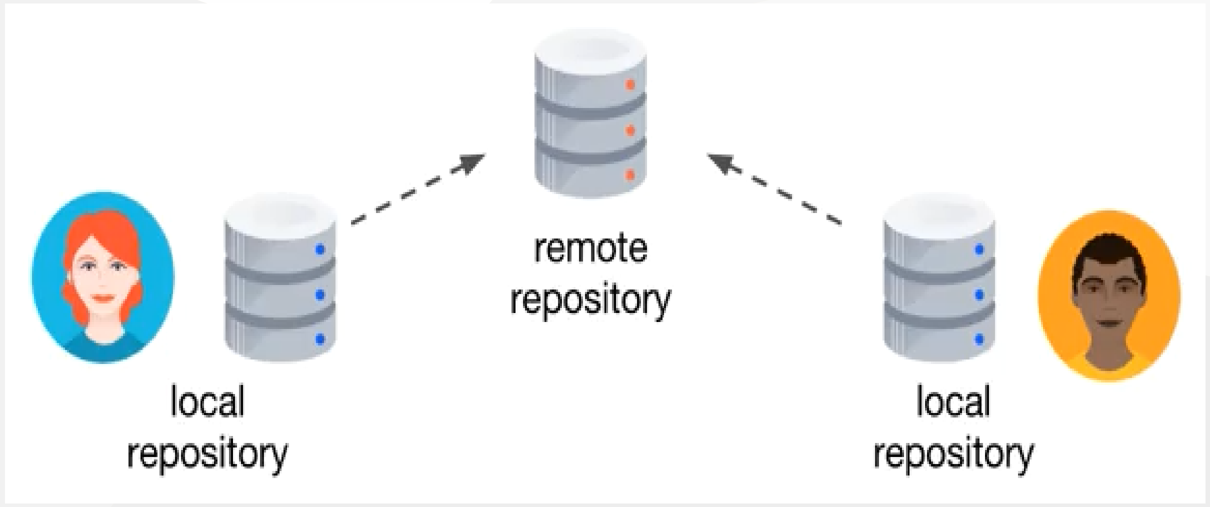
Permite detectar que ha sido cambiado de una versión a otra, línea por línea.

Experimentar

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

Repository

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.



Using a existing repository

If the repository already exists, we need to clone it to our local machine:

```
git clone <repository_URL>
```

Create a new repository

Let's start by creating a new repository and adding some files to it.

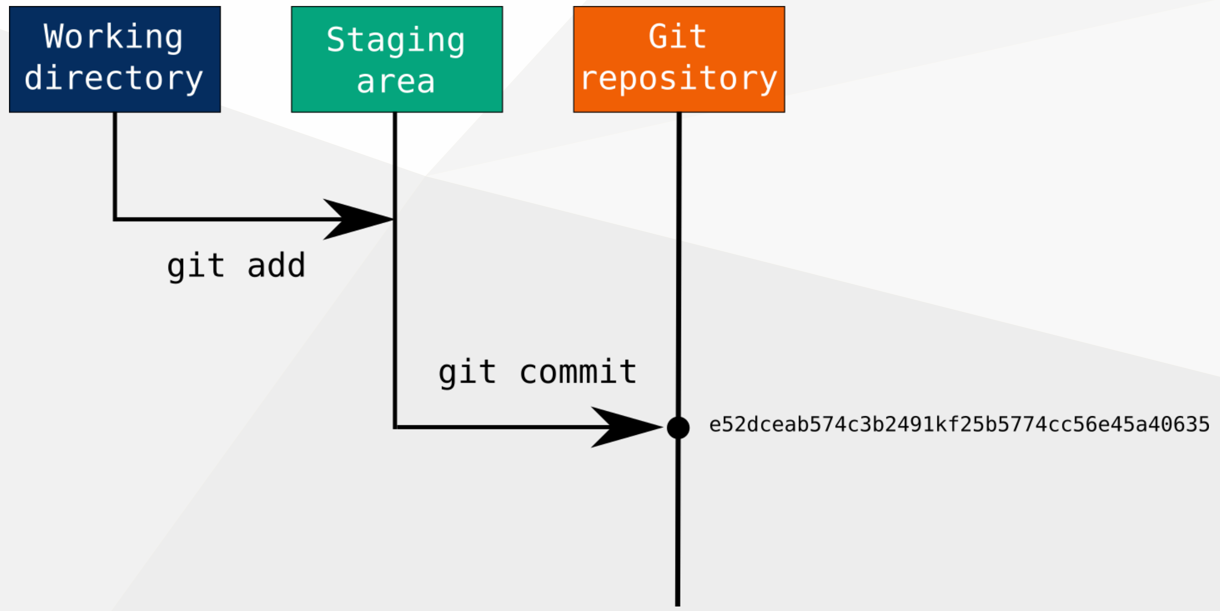
```
mkdir my_project
cd my_project
git init
```

Now, let's create a new text file and add some content to it.

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

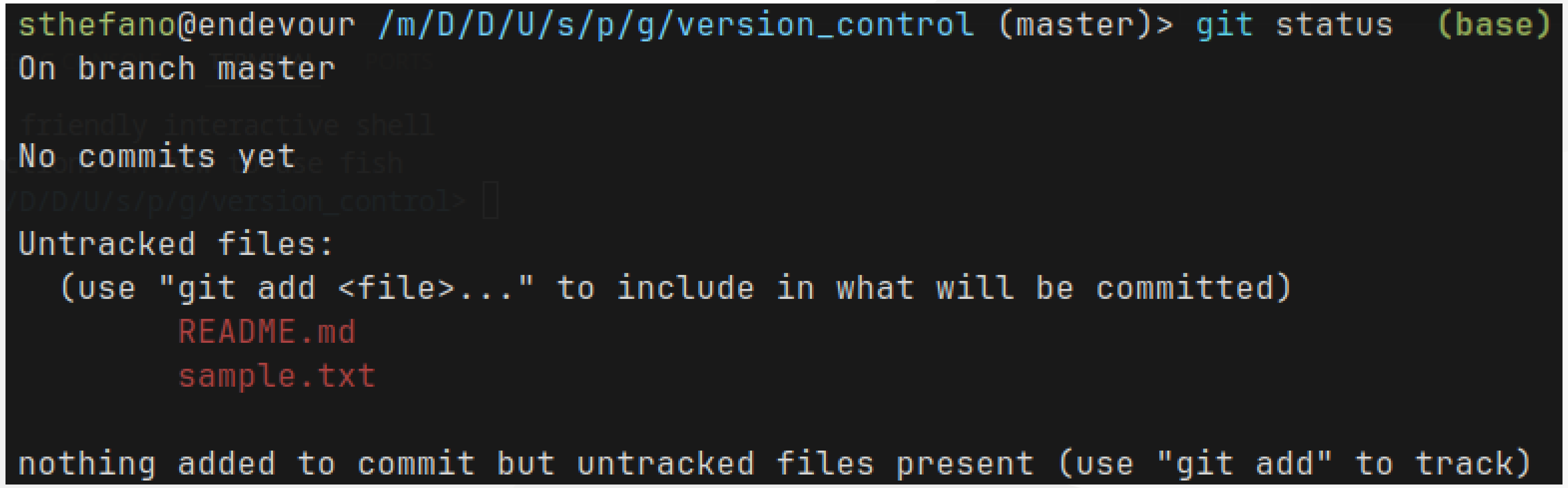
Making Changes

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.



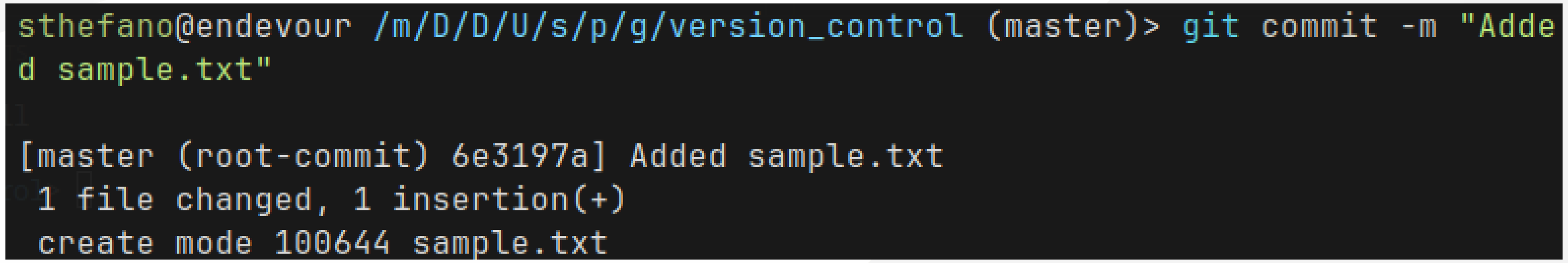
Next, let's track the changes we made and commit them to the repository.

```
git status
```



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

No commits yet

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 "Adde
d sample.txt"

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

Now, let's modify the content of `sample.txt`. Note the flag `-am`, where `a` adds the tracked files for the commit.

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

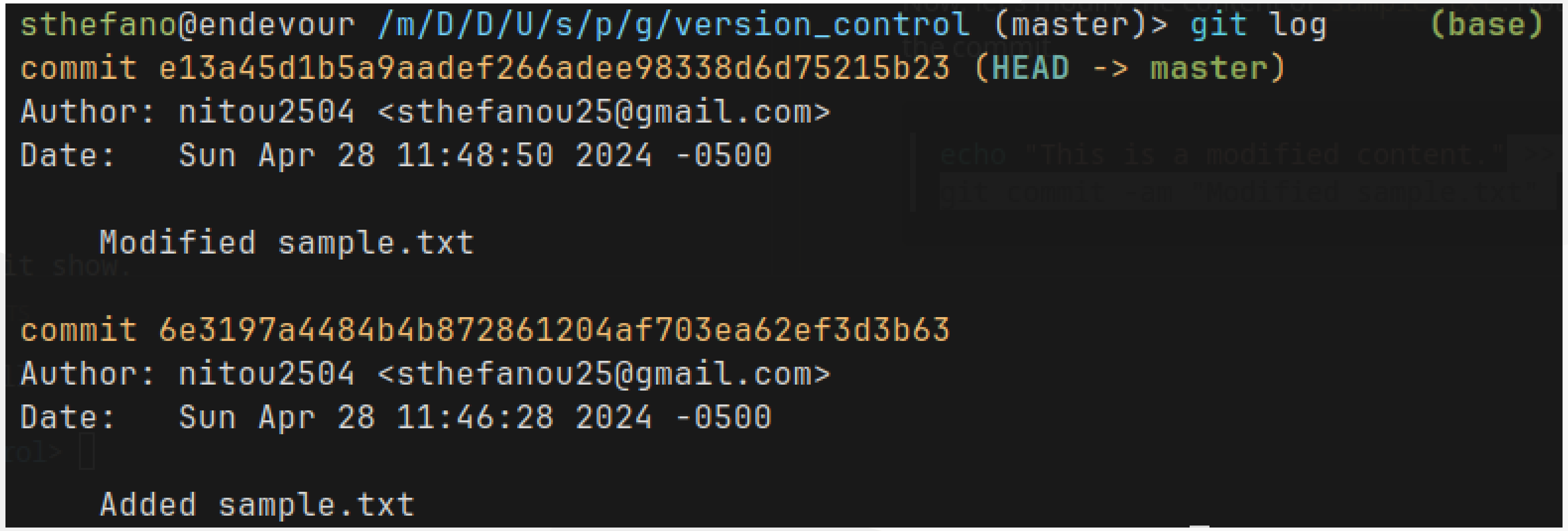
Viewing History

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.

```
>_
commit 2d358ee491.....
commit c2995fda.....
commit 258a5387124a....
```

We can view the commit history to see our changes.

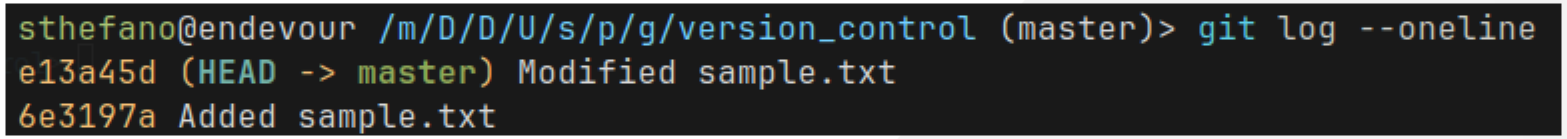
```
git log
```



```
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git log     (base)
commit e13a45d1b5a9aadef266adee98338d6d75215b23 (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
```

For a concise view:

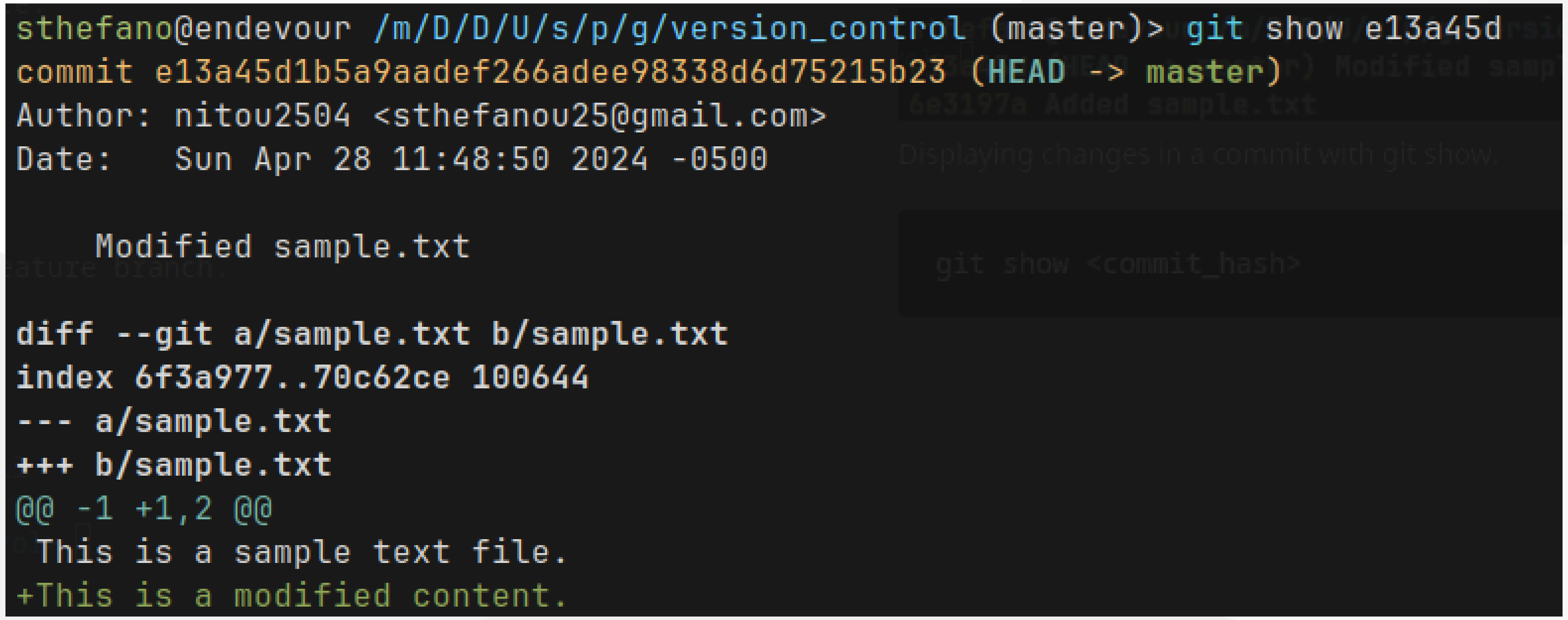
```
git log --oneline
```



```
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
```

Displaying changes in a commit with git show.

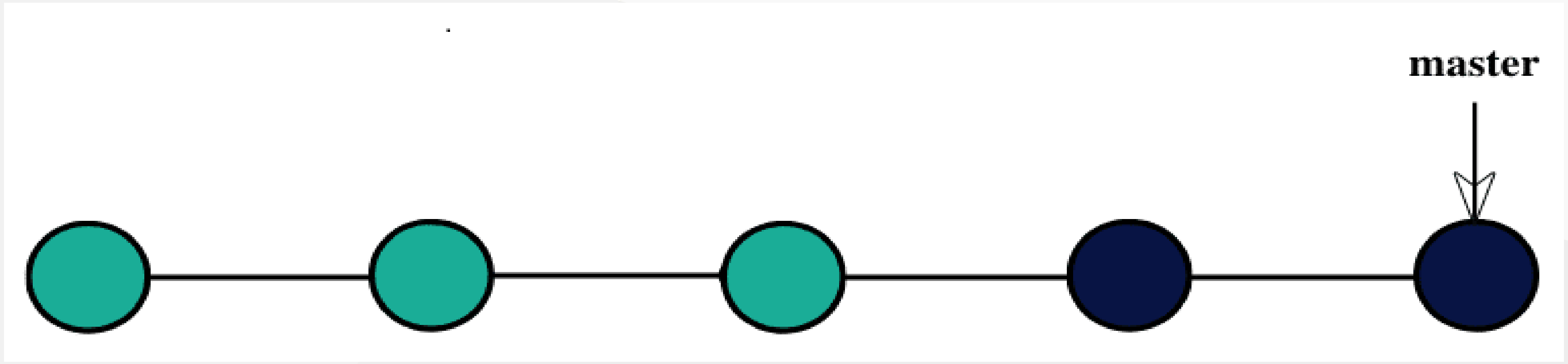
```
git show <commit_hash>
```



```
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git show e13a45d
commit e13a45d1b5a9aadef266adee98338d6d75215b23 (HEAD -> master)
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 +1,2 @@
 This is a sample text file.
+This is a modified content.
```

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.

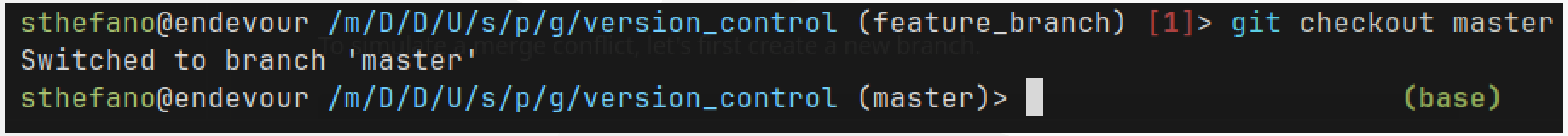


Now, let's make some changes in the feature branch.

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

Switch back to the main branch.

```
git checkout master
```

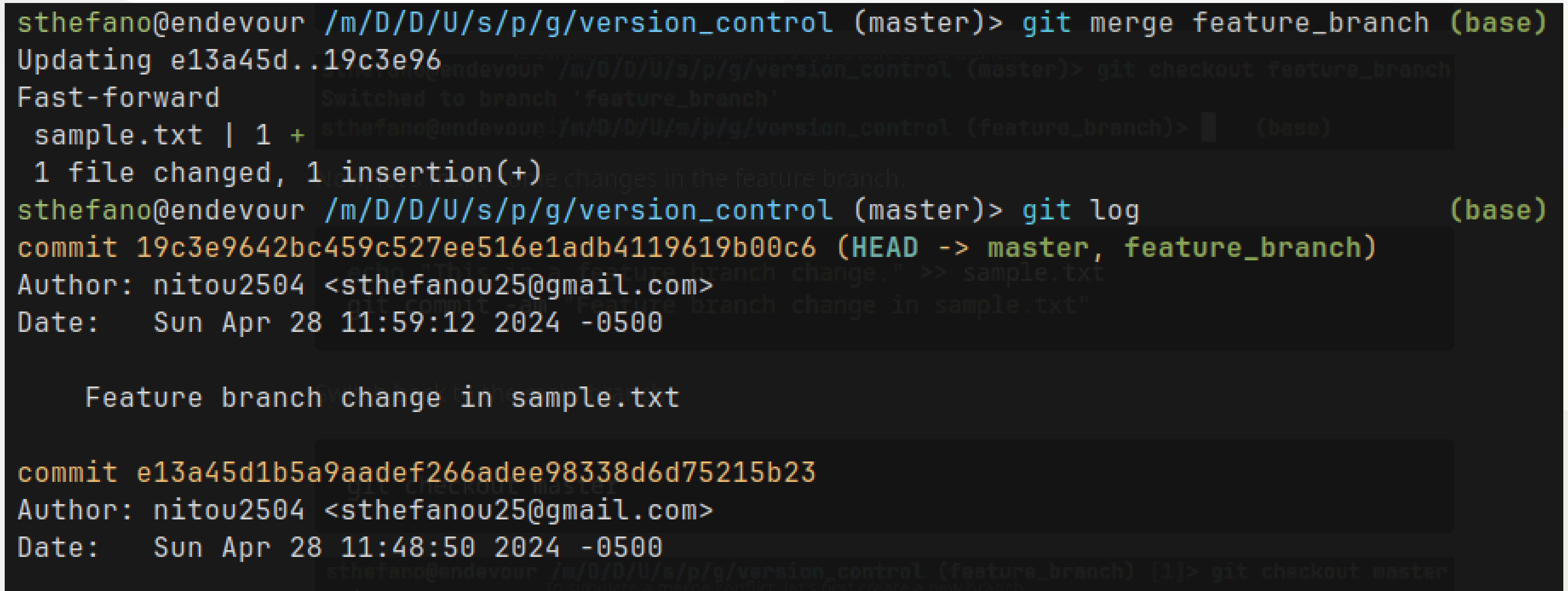


```
sthefano@endevour /m/D/D/U/s/p/g/version_control (feature_branch) [1]> git checkout master
Switched to branch 'master'
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)>                         (base)
```

Merge the changes from the feature branch into the main branch.

```
git merge feature_branch
```

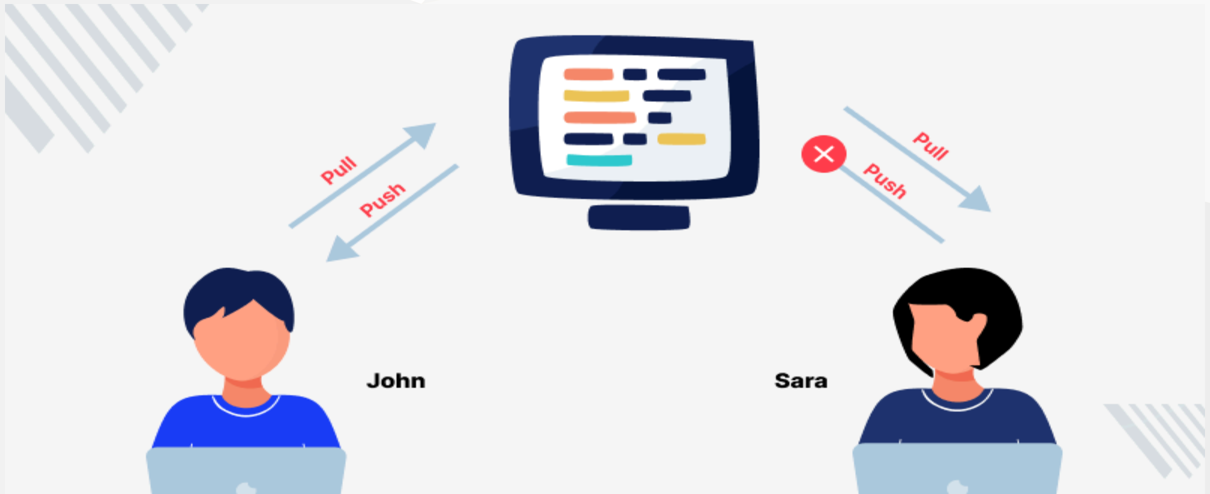
Note that the commit from the `feature_branch` is now in the `master` branch:



```
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 (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 e13a45d1b5a9aadef266adee98338d6d75215b23
Author: nitou2504 <sthefanou25@gmail.com>
Date:   Sun Apr 28 11:48:50 2024 -0500
```

Merge Conflict

Un merge conflict 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.

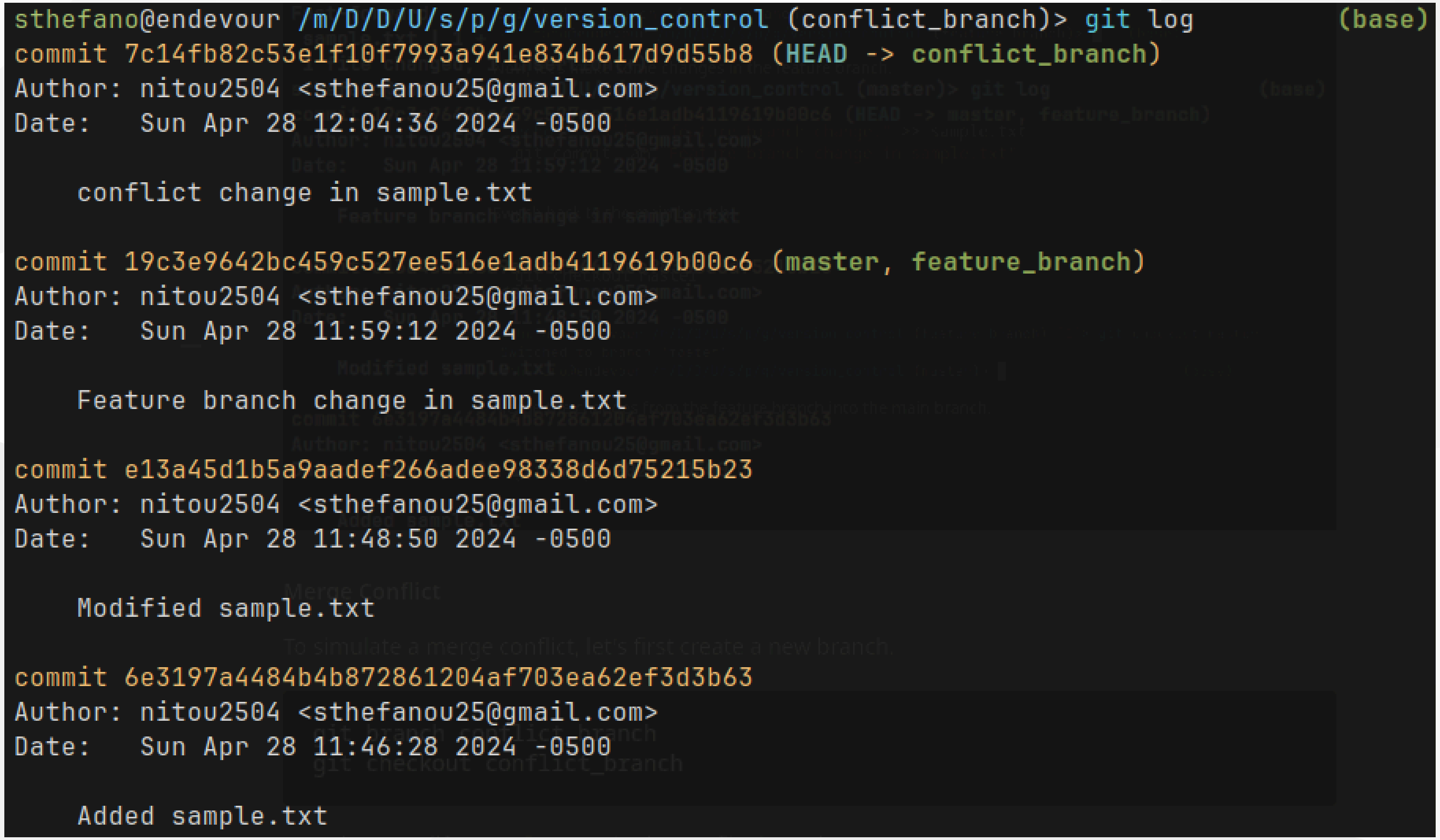


To simulate a merge conflict, let's first create a new branch.

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

Now, let's modify `sample.txt` in the conflict branch.

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

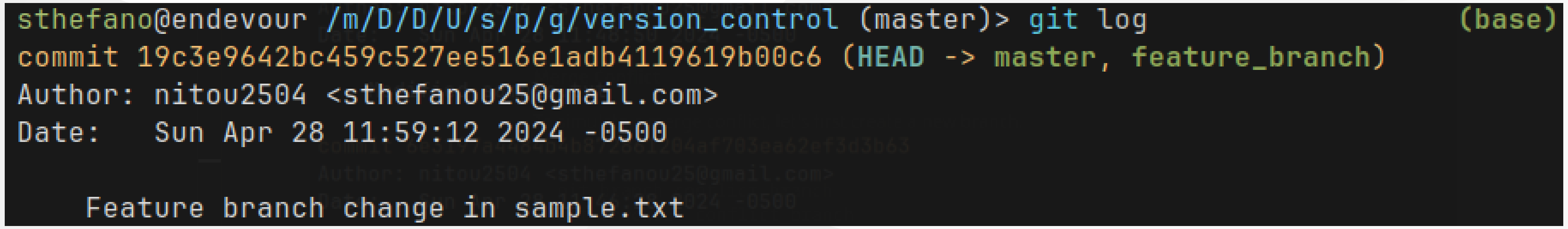


```
sthefano@endevour /m/D/D/U/s/p/g/version_control (conflict_branch)> git log          (base)
commit 7c14fb82c53e1f10f7993a941e834b617d9d55b8 (HEAD -> conflict_branch)
Author: nitou2504 <sthefanou25@gmail.com>
Date:   Sun Apr 28 12:04:36 2024 -0500

    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 e13a45d1b5a9aadef266adee98338d6d75215b23
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
```

Switch back to the main branch.

```
git checkout master
```

Note that the last change in the `master` was from the previous branch merge:



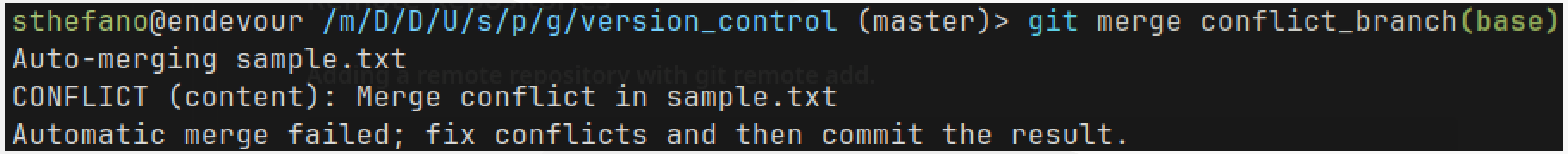
```
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git log                   (base)
commit 19c3e9642bc459c527ee516e1adb4119619b00c6 (HEAD -> master, feature_branch)
Author: nitou2504 <sthefanou25@gmail.com>
Date:   Sun Apr 28 11:59:12 2024 -0500

    Feature branch change in sample.txt
```

Now, let's modify the same line in `sample.txt` in the main branch.

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

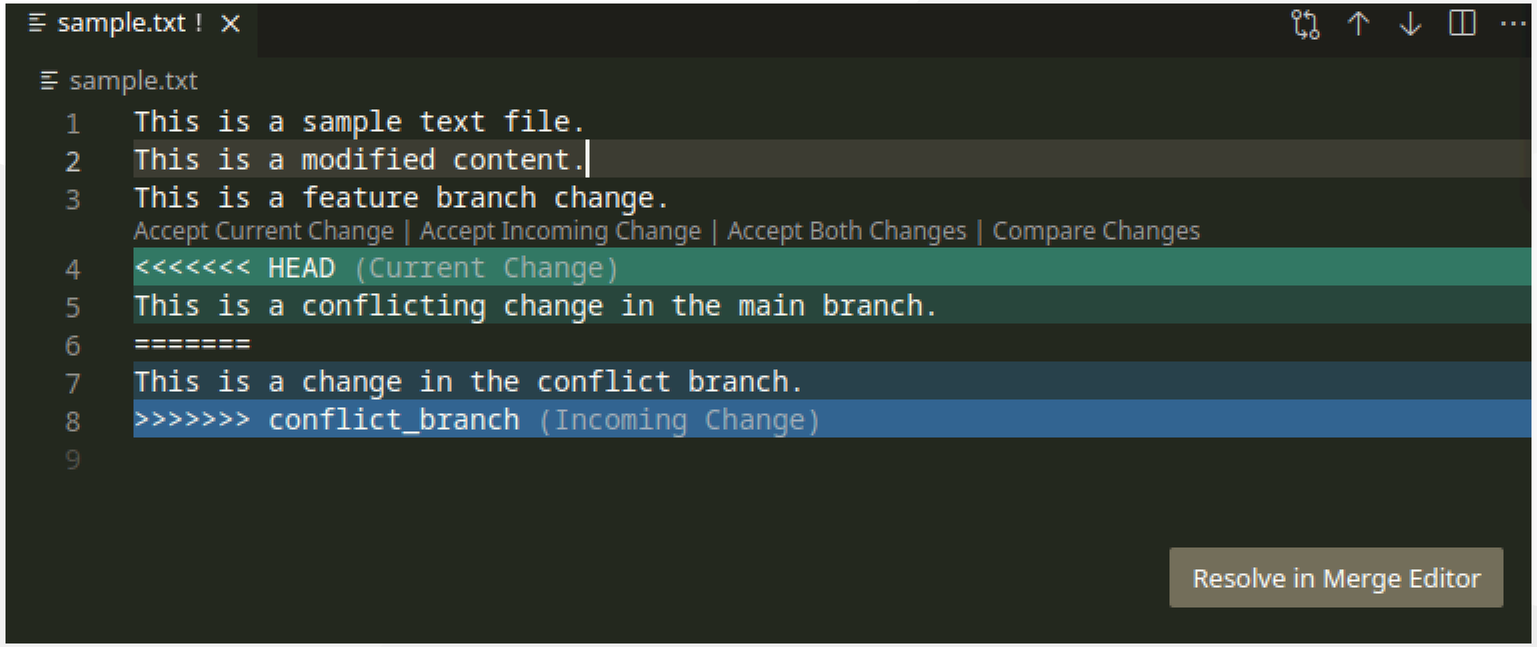
Now, try to merge the conflict branch into the main branch.

```
git merge conflict_branch
```



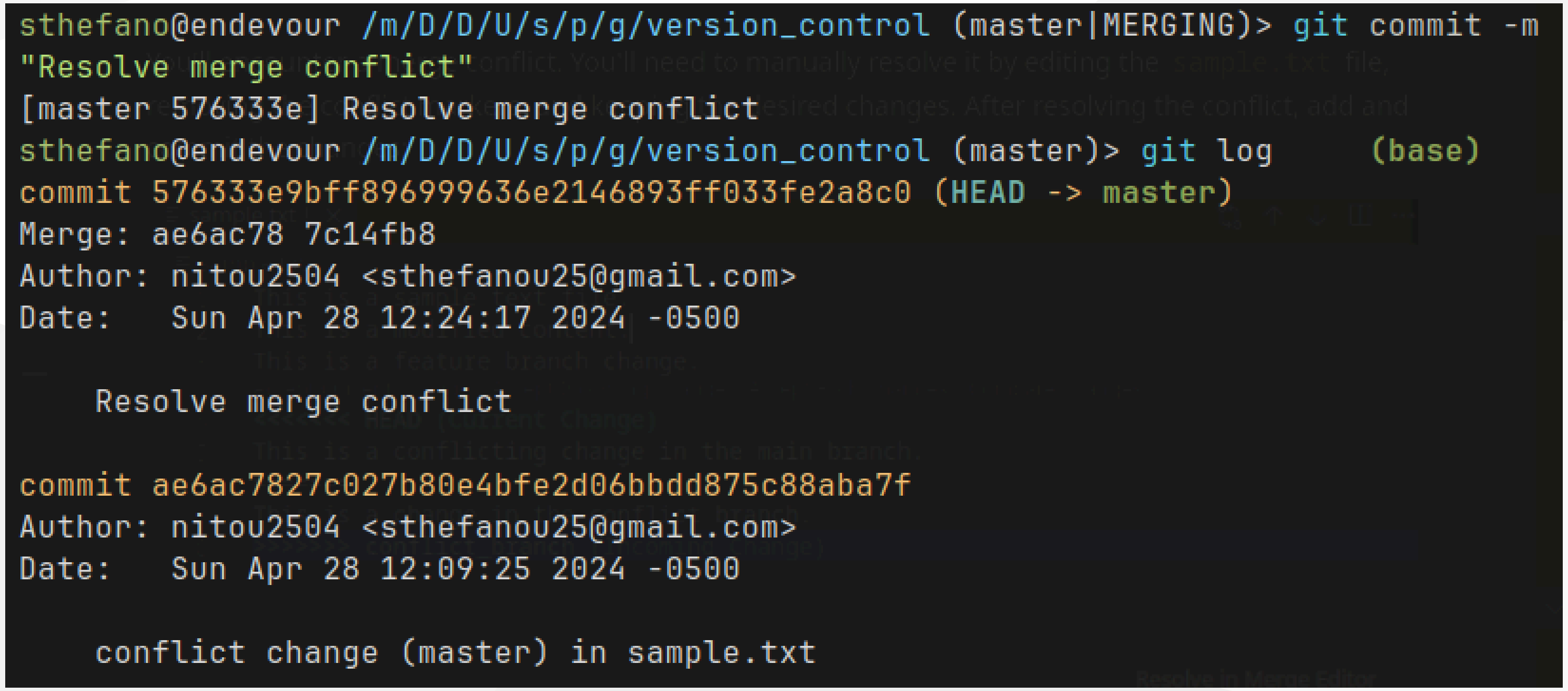
```
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git merge conflict_branch(base)
Auto-merging sample.txt
CONFLICT (content): Merge conflict in sample.txt
Automatic merge failed; fix conflicts and then commit the result.
```

You'll encounter a merge conflict. You'll need to manually resolve it by editing the `sample.txt` file, removing the conflict markers, and keeping the desired changes. After resolving the conflict, add and commit the changes:


sample.txt !
sample.txt
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
Resolve in Merge Editor

```
This is a sample text file.
This is a modified content.
This is a feature branch change.
<<<<<<< HEAD (Current Change)
This is a conflicting change in the main branch.
=======
This is a change in the conflict branch.
>>>>>>> conflict_branch (Incoming Change)
```

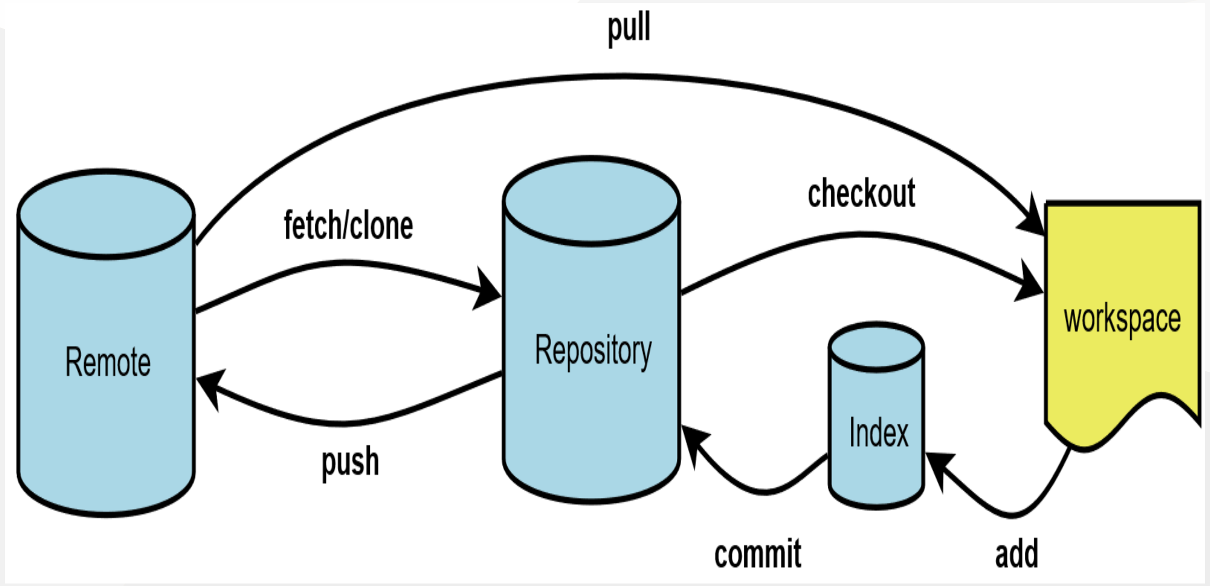
```
git add sample.txt
git commit -m "Resolve merge conflict"
```



```
sthefano@endevour /m/D/D/U/s/p/g/version_control (master|MERGING)> git commit -m
"Resolve merge conflict"
[master 576333e] Resolve merge conflict
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git log      (base)
commit 576333e9bff896999636e2146893ff033fe2a8c0 (HEAD -> master)
Merge: ae6ac78 7c14fb8
Author: nitou2504 <sthefanou25@gmail.com>
Date:   Sun Apr 28 12:24:17 2024 -0500

    Resolve merge conflict

commit ae6ac7827c027b80e4bfe2d06bbdd875c88aba7f
Author: nitou2504 <sthefanou25@gmail.com>
Date:   Sun Apr 28 12:09:25 2024 -0500

    conflict change (master) in sample.txt
```

Remote Repositories

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.

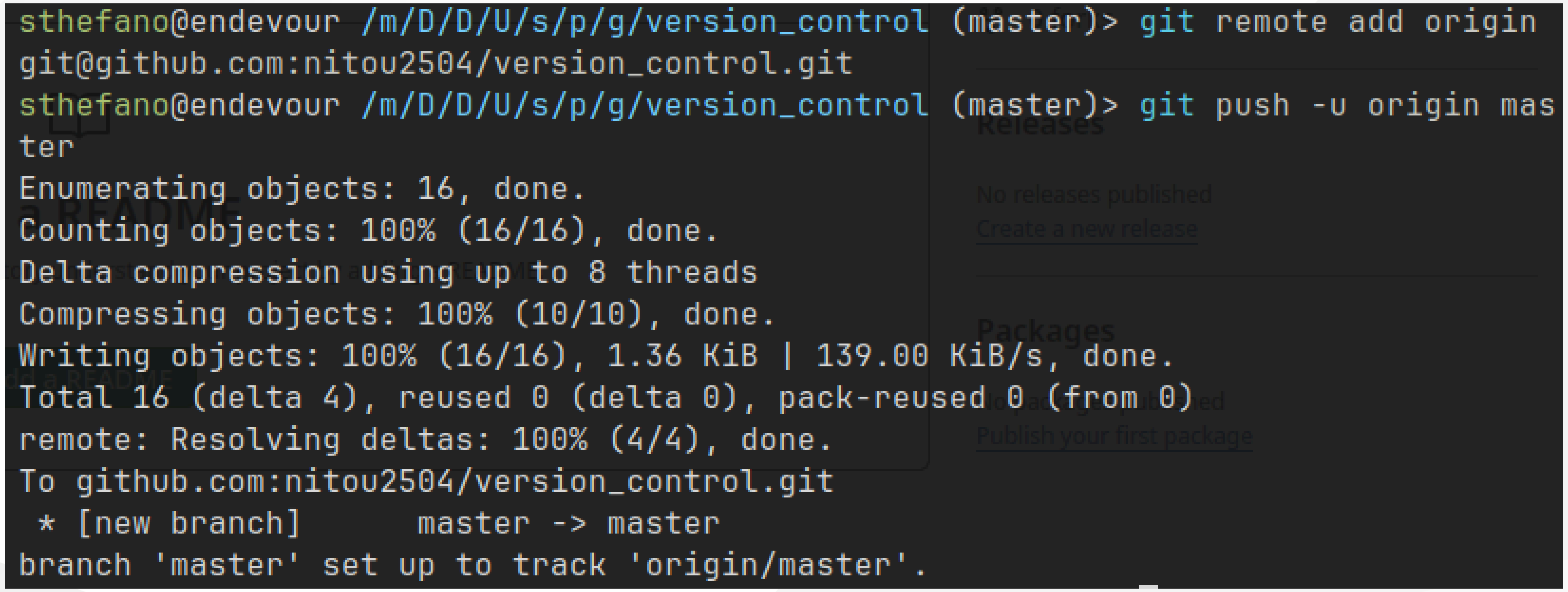


Adding a remote repository with git remote add.

```
git remote add origin <repository_URL>
```

If you already have changes in your local git and want to push them to a new empty remote repo:

```
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'.
```

Pushing changes to a remote repository with git push.

```
git push origin <branch_name>
```

Pulling changes from a remote repository with git pull.

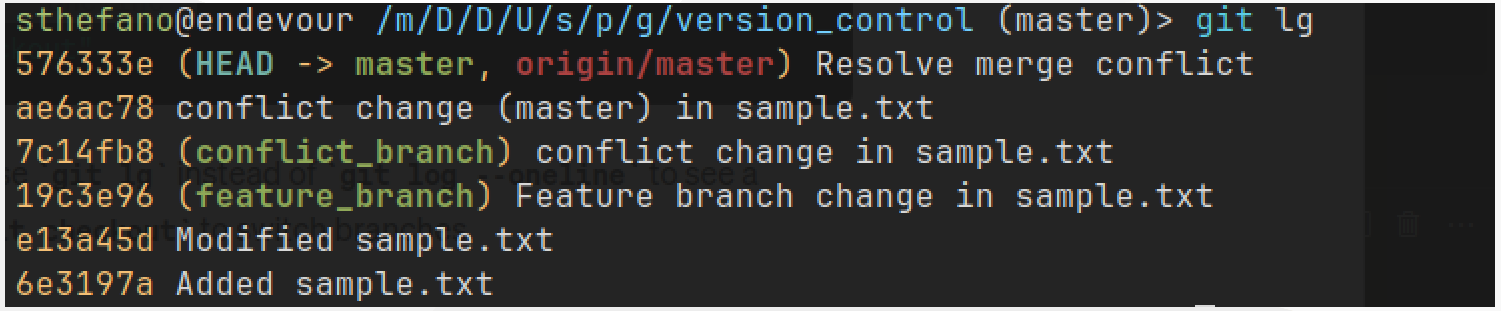
```
git pull origin <branch_name>
```

Useful Tips and Tricks

Aliases for common commands.

You can set up aliases in your `~/.gitconfig` file:

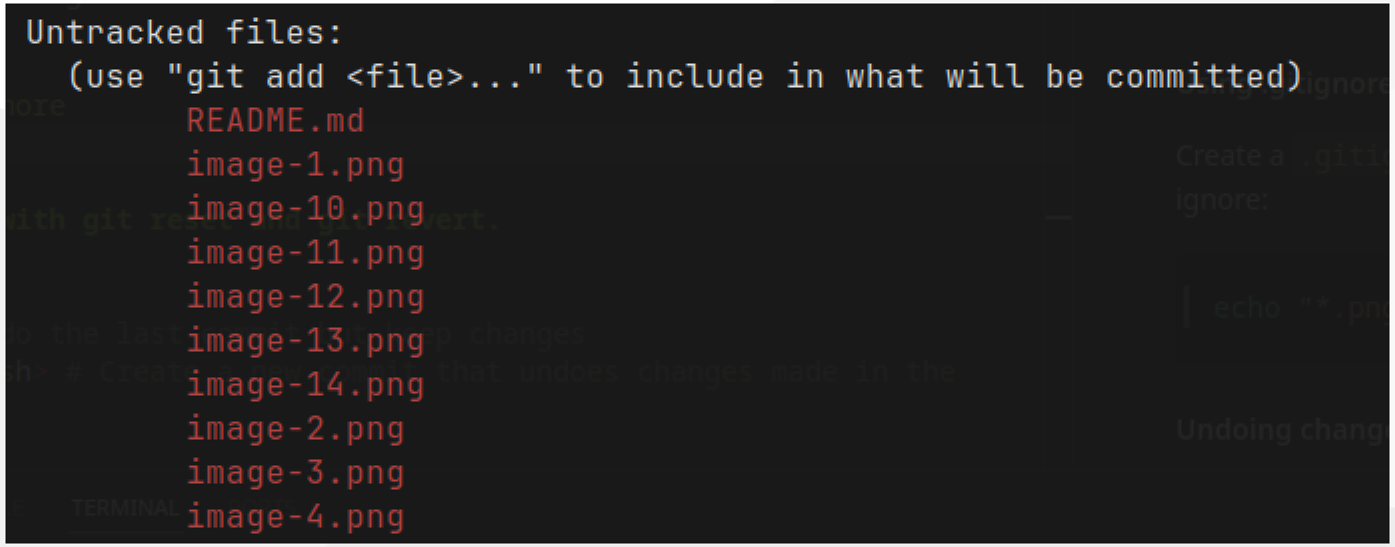
```
git config --global alias.lg "log --oneline"
```



```
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
```

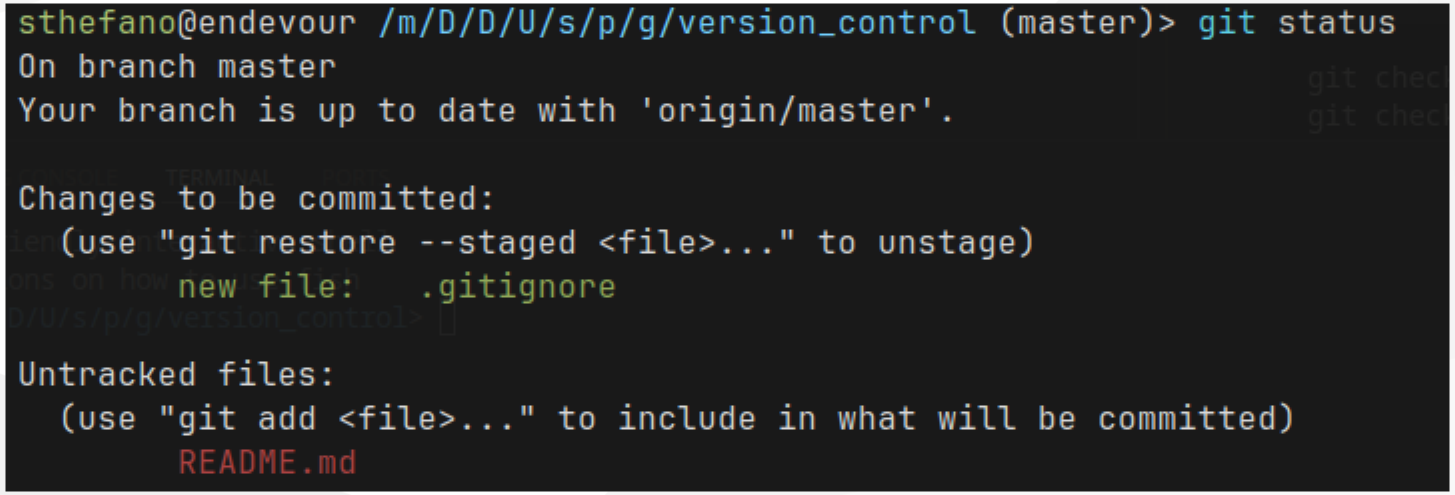
Using .gitignore to ignore files.

Create a `.gitignore` file in your repository's root directory and list the files or patterns you want to ignore. This is commonly done for binaries such as images, since they will appear on status messages:



```
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
        image-3.png
        image-4.png
```

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

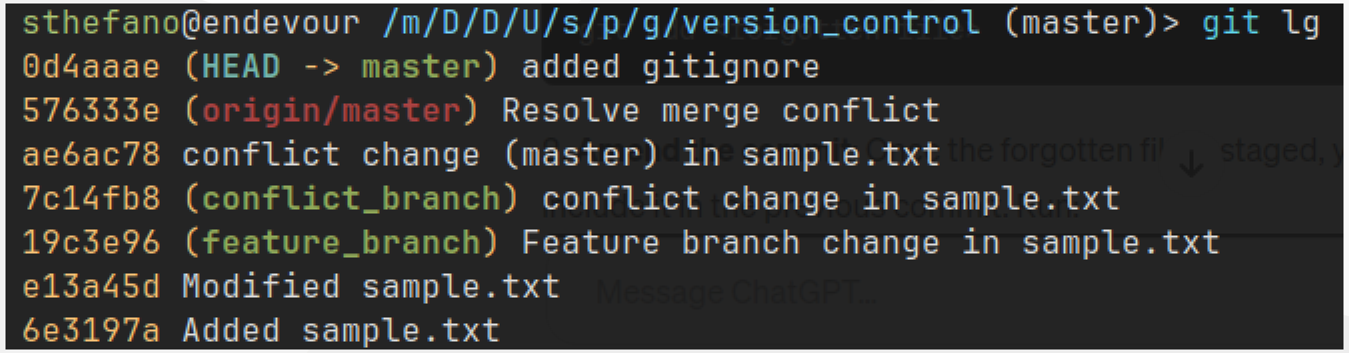


```
sthefano@endevour /m/D/D/U/s/p/g/version_control (master)> git status
On branch master
Your branch is up to date with 'origin/master'.

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
```

Fix last commit with `amend`

`--amend` is a useful flag to use when we make a commit but we forgot to add files or changes (or fix typos in the commit message). For example, we forgot to add the README.md in our last commit:

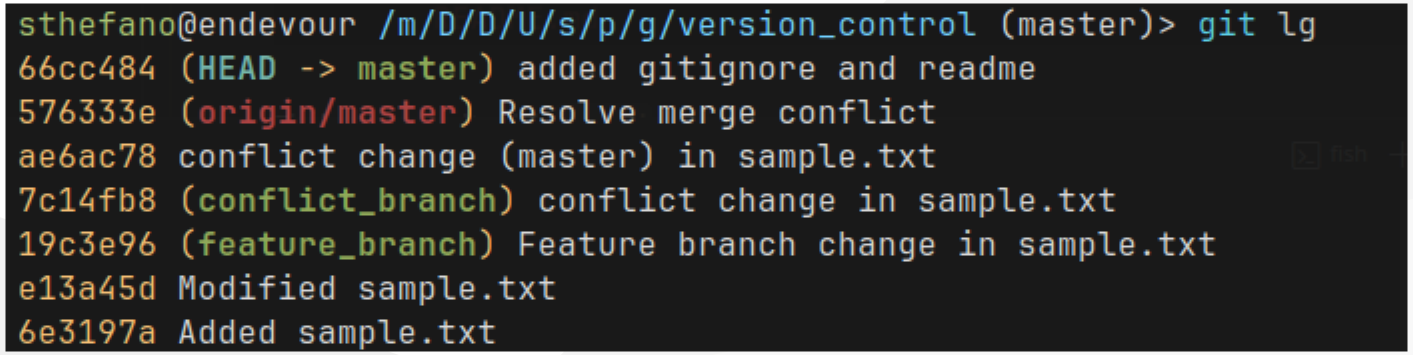
```
git commit -am "added gitignore"
```



```
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
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
```

We need to stage the forgotten file, and ammend the 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
```

Thus, our last commit was fixed.

Jumping between old and current versions with checkout `<commit>` and checkout `<HEAD>`.

```
git checkout <commit_hash> # To switch to an old commit
git checkout <branch_name> # To switch back to the current branch
```