Hands on tutorial: GIT

Marco Flores-Coronado

November 22, 2024

Outline

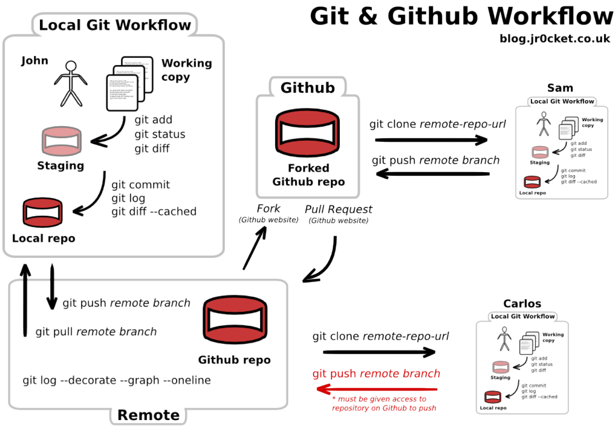
1. What in the world is Git?
2. git, github, or gitlab?
3. git –version control– outline
4. Creating a ssh key
5. Git clone
6. Git branch
7. Git checkput -b
8. Task
9. Git status, Git add
10. Git commit
11. Git push
12. Pull request (merge)
13. If everything breaks?

What in the world is Git?

git, github, or gitlab?

git –version control– outline

image source



git add
git status
git diff

git commit
git log
git diff --cached

git clone *remote-repo-url*

git push *remote branch*

git push *remote branch*

git pull *remote branch*

git log --decorate --graph --oneline

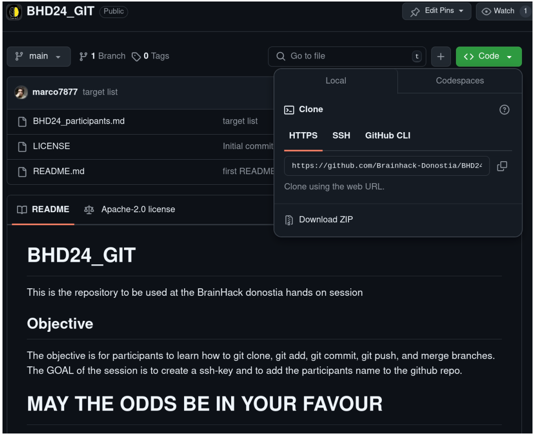
git clone *remote-repo-url*

git push *remote branch*

** must be given access to repository on Github to push*

Creating a ssh key

Cloning a repo



https://github.com/Brainhack-Donostia/BHD24

Clone using the web URL.

marco7877	target list
BHD24_participants.md	target list
LICENSE	Initial commit
README.md	first README

BHD24_GIT

This is the repository to be used at the BrainHack donostia hands on session

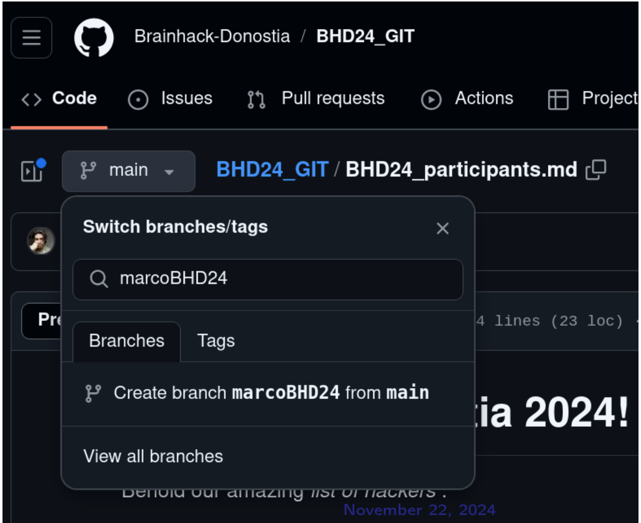
Objective

The objective is for participants to learn how to git clone, git add, git commit, git push, and merge branches. The GOAL of the session is to create a ssh-key and to add the participants name to the github repo.

MAY THE ODDS BE IN YOUR FAVOUR

```
git clone https://github.com/Brainhack−Donostia/BHD24_GIT.
```

Creating a branch



Brainhack-Donostia / BHD24_GIT

BHD24_GIT / BHD24_participants.md

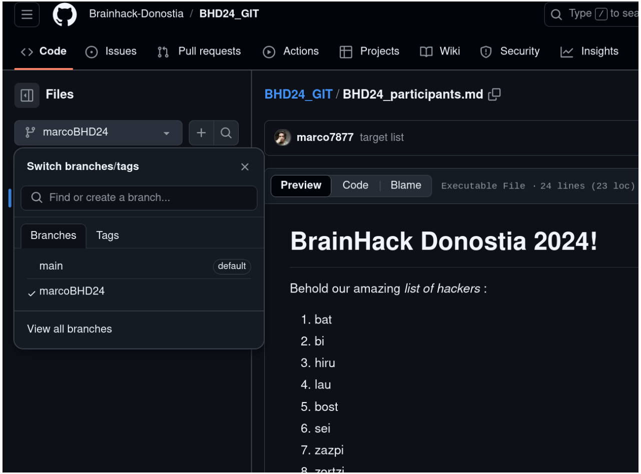
Switch branches/tags

marcoBHD24

Create branch **marcoBHD24** from **main**

View all branches

Creating a branch



BHD24_GIT / BHD24_participants.md

marco7877 target list

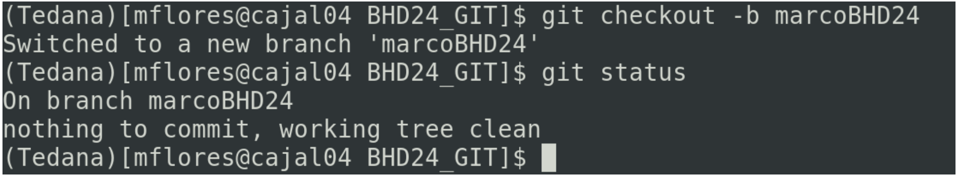
Executable File · 24 lines (23 loc)

BrainHack Donostia 2024!

Behold our amazing *list of hackers* :

1. bat
2. bi
3. hiru
4. lau
5. bost
6. sei
7. zazpi
8. zortzi

Creating a branch *terminal*



```
(Tedana)[mflores@cajal04 BHD24_GIT]$ git checkout -b marcoBHD24
Switched to a new branch 'marcoBHD24'
(Tedana)[mflores@cajal04 BHD24_GIT]$ git status
On branch marcoBHD24
nothing to commit, working tree clean
(Tedana)[mflores@cajal04 BHD24_GIT]$ █
```

```
git checkout -b marcoBHD24
```

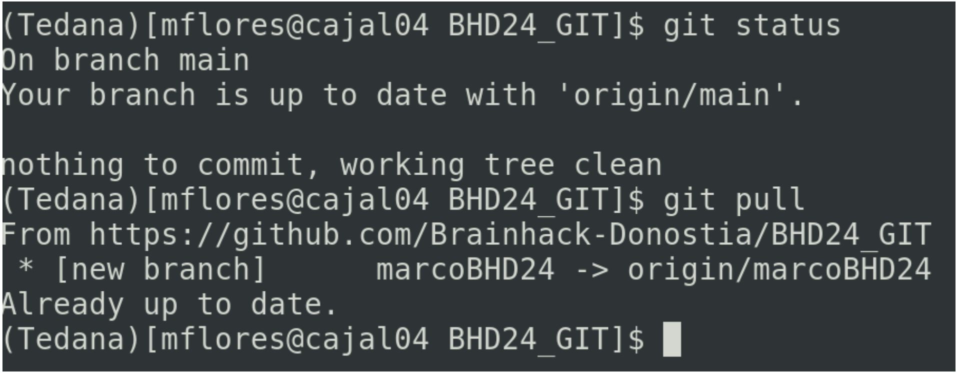
Edit BHD_participants

Change your assigned number written in euskera to your name

Remember to be do so over your newly created branch

Always *git pull* before editing

Git status, Git add

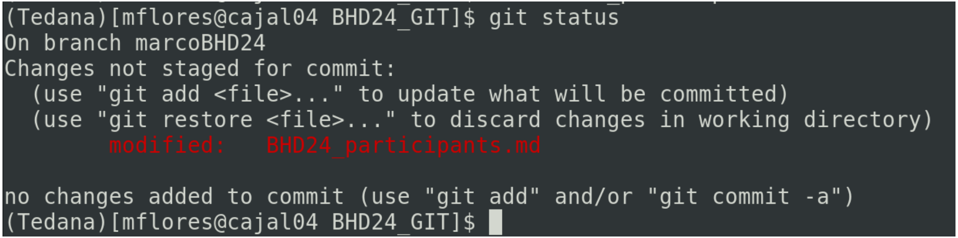


```
(Tedana)[mflores@cajal04 BHD24_GIT]$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
(Tedana)[mflores@cajal04 BHD24_GIT]$ git pull
From https://github.com/Brainhack-Donostia/BHD24_GIT
 * [new branch]      marcoBHD24 -> origin/marcoBHD24
Already up to date.
(Tedana)[mflores@cajal04 BHD24_GIT]$
```

```
git pull
git status
git add <file>
```

Git status, Git add

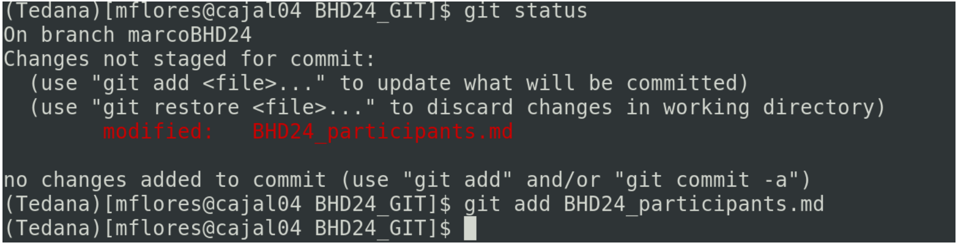


```
(Tedana)[mflores@cajal04 BHD24_GIT]$ git status
On branch marcoBHD24
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   BHD24_participants.md

no changes added to commit (use "git add" and/or "git commit -a")
(Tedana)[mflores@cajal04 BHD24_GIT]$ 
```

```
git pull
git status
git add <file>
```

Git status, Git add

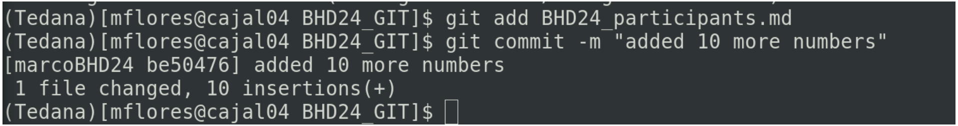


```
(Tedana)[mflores@cajal04 BHD24_GIT]$ git status
On branch marcoBHD24
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   BHD24_participants.md

no changes added to commit (use "git add" and/or "git commit -a")
(Tedana)[mflores@cajal04 BHD24_GIT]$ git add BHD24_participants.md
(Tedana)[mflores@cajal04 BHD24_GIT]$ 
```

```
git pull
git status
git add <file>
```

Git commit



```
(Tedana)[mflores@cajal04 BHD24_GIT]$ git add BHD24_participants.md
(Tedana)[mflores@cajal04 BHD24_GIT]$ git commit -m "added 10 more numbers"
[marcoBHD24 be50476] added 10 more numbers
 1 file changed, 10 insertions(+)
(Tedana)[mflores@cajal04 BHD24_GIT]$ 
```

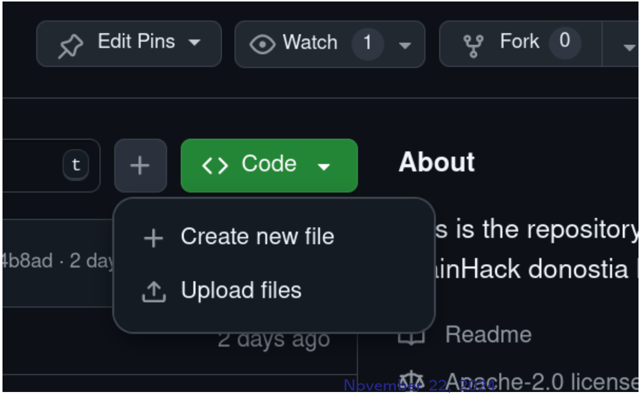
```
git commit —m "your selfcontained message"
```

Git commit & add GUI

You can do it from a browser or the app

Pros: GUI, clicking

Cons: Slow, depend on user self track of changes

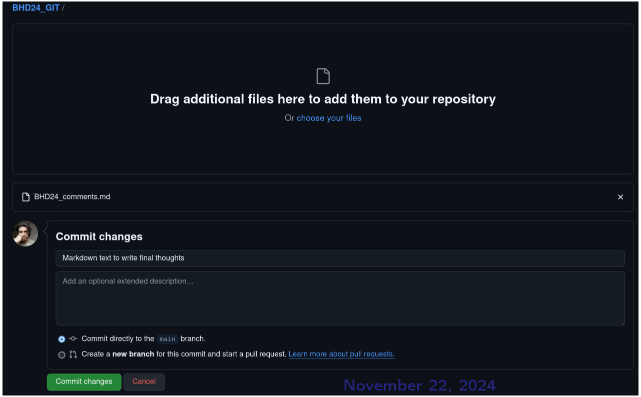


Git commit & add GUI

You can do it from a browser or the app

Pros: GUI, clicking

Cons: Slow, depend on user self track of changes



BHD24_GIT /

Drag additional files here to add them to your repository

Or choose your files

BHD24_comments.md ×

Commit changes

Markdown text to write final thoughts

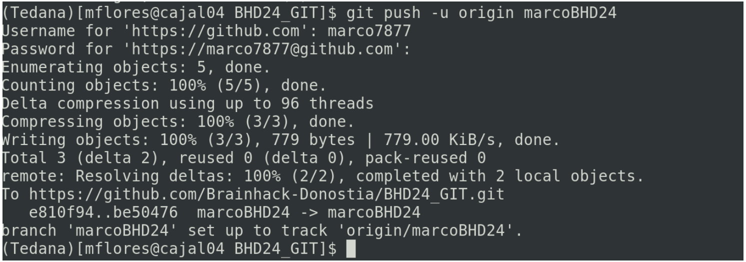
Add an optional extended description…

● Commit directly to the `main` branch.

○ Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

Git push *terminal*

Only needed if you work from terminal!

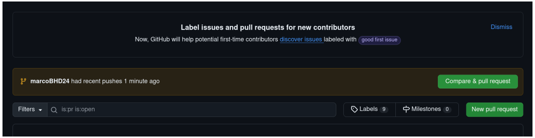


```
(Tedana)[mflores@cajal04 BHD24_GIT]$ git push -u origin marcoBHD24
Username for 'https://github.com': marco7877
Password for 'https://marco7877@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 96 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 779 bytes | 779.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Brainhack-Donostia/BHD24_GIT.git
   e810f94..be50476  marcoBHD24 -> marcoBHD24
branch 'marcoBHD24' set up to track 'origin/marcoBHD24'.
(Tedana)[mflores@cajal04 BHD24_GIT]$ 
```

```
git push —u origin <branch | main>
```

Pull request

**Merges tracks from one branch (i.e., develop) to other (i.e., main)*



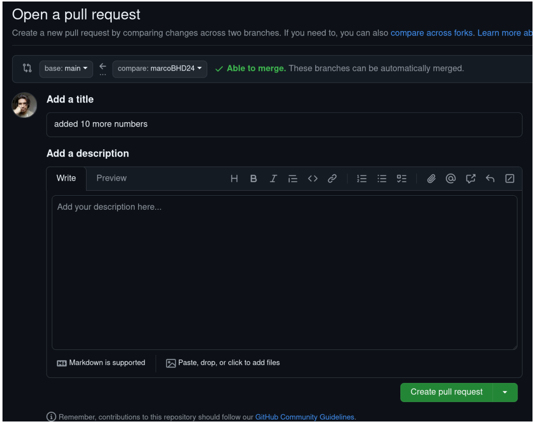
Label issues and pull requests for new contributors

Now, GitHub will help potential first-time contributors discover issues labeled with good first issue

marcoBHD24 had recent pushes 1 minute ago

Pull request

**Merges tracks from one branch (i.e., develop) to other (i.e., main)*



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks. Learn more ab

base: main ← compare: marcoBHD24 ✓ **Able to merge.** These branches can be automatically merged.

Add a title

added 10 more numbers

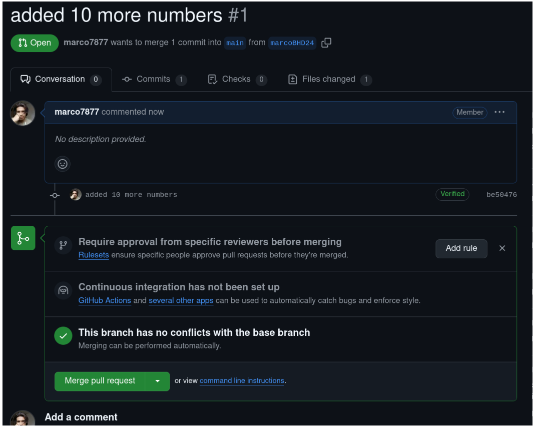
Add a description

Add your description here...

Remember, contributions to this repository should follow our GitHub Community Guidelines.

Pull request

**Merges tracks from one branch (i.e., develop) to other (i.e., main)*



added 10 more numbers #1

Open marco7877 wants to merge 1 commit into main from marcoBHD24

marco7877 commented now

No description provided.

added 10 more numbers

be50476

Require approval from specific reviewers before merging

Rulesets ensure specific people approve pull requests before they're merged.

Continuous integration has not been set up

GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch

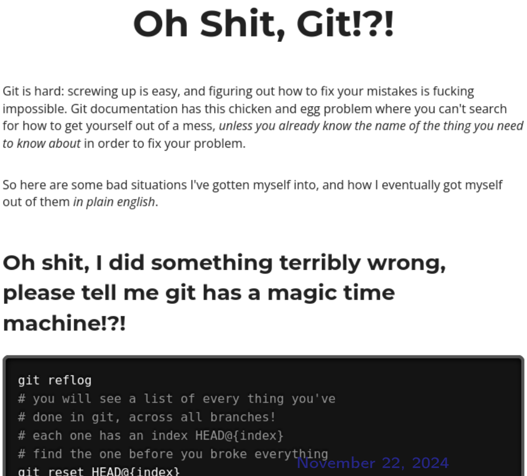
Merging can be performed automatically.

or view command line instructions.

Add a comment

Oh shit, Git

OhShitGit.com



Oh Shit, Git!?!

Git is hard: screwing up is easy, and figuring out how to fix your mistakes is fucking impossible. Git documentation has this chicken and egg problem where you can't search for how to get yourself out of a mess, *unless you already know the name of the thing you need to know about* in order to fix your problem.

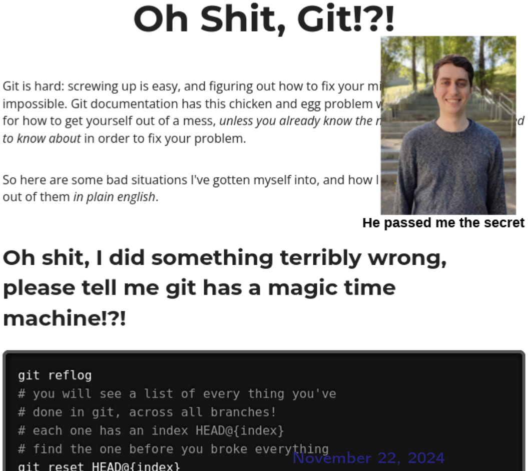
So here are some bad situations I've gotten myself into, and how I eventually got myself out of them *in plain english*.

Oh shit, I did something terribly wrong, please tell me git has a magic time machine!?!

```
git reflog
# you will see a list of every thing you've
# done in git, across all branches!
# each one has an index HEAD@{index}
# find the one before you broke everything
git reset HEAD@{index}
```

Oh shit, Git

OhShitGit.com



Oh Shit, Git!?!

Git is hard: screwing up is easy, and figuring out how to fix your mi
impossible. Git documentation has this chicken and egg problem w
for how to get yourself out of a mess, *unless you already know the n* *d*
to know about in order to fix your problem.

So here are some bad situations I've gotten myself into, and how I
out of them *in plain english*.

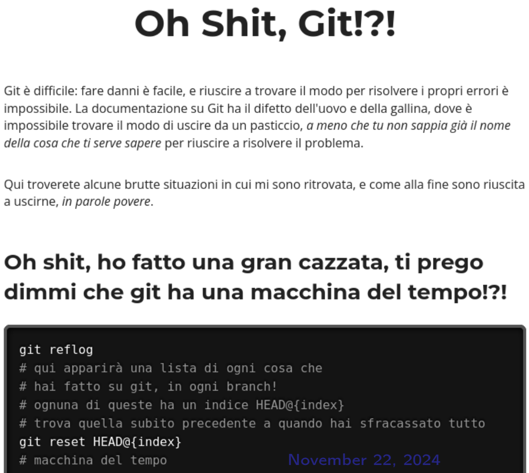
He passed me the secret

Oh shit, I did something terribly wrong, please tell me git has a magic time machine!?!

```
git reflog
# you will see a list of every thing you've
# done in git, across all branches!
# each one has an index HEAD@{index}
# find the one before you broke everything
git reset HEAD@{index}
```

Oh shit, Git

OhShitGit.com



Oh Shit, Git!?!

Git è difficile: fare danni è facile, e riuscire a trovare il modo per risolvere i propri errori è impossibile. La documentazione su Git ha il difetto dell'uovo e della gallina, dove è impossibile trovare il modo di uscire da un pasticcio, *a meno che tu non sappia già il nome della cosa che ti serve sapere* per riuscire a risolvere il problema.

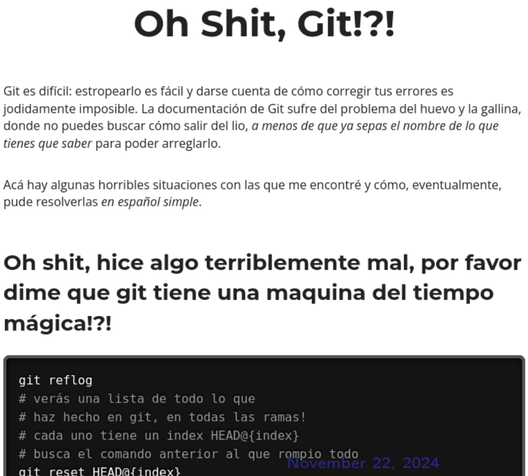
Qui troverete alcune brutte situazioni in cui mi sono ritrovata, e come alla fine sono riuscita a uscirne, *in parole povere*.

Oh shit, ho fatto una gran cazzata, ti prego dimmi che git ha una macchina del tempo!?!

```
git reflog
# qui apparirà una lista di ogni cosa che
# hai fatto su git, in ogni branch!
# ognuna di queste ha un indice HEAD@{index}
# trova quella subito precedente a quando hai sfracassato tutto
git reset HEAD@{index}
# macchina del tempo
```

Oh shit, Git

OhShitGit.com



Oh Shit, Git!?!

Git es difícil: estropearlo es fácil y darse cuenta de cómo corregir tus errores es jodidamente imposible. La documentación de Git sufre del problema del huevo y la gallina, donde no puedes buscar cómo salir del lio, *a menos de que ya sepas el nombre de lo que tienes que saber* para poder arreglarlo.

Acá hay algunas horribles situaciones con las que me encontré y cómo, eventualmente, pude resolverlas *en español simple*.

Oh shit, hice algo terriblemente mal, por favor dime que git tiene una maquina del tiempo mágica!?!

```
git reflog
# verás una lista de todo lo que
# haz hecho en git, en todas las ramas!
# cada uno tiene un index HEAD@{index}
# busca el comando anterior al que rompio todo
git reset HEAD@{index}
```

Thanks