



# git

---

*"Il controllo di versione è un sistema che registra, nel tempo, i cambiamenti ad un file o ad una serie di file, così da poter richiamare una specifica versione in un secondo momento."*

---

"Un VCS ti permette di ripristinare i file ad una versione precedente, ripristinare l'intero progetto a uno stato precedente, revisionare le modifiche fatte nel tempo, vedere chi ha cambiato qualcosa che può aver causato un problema, chi ha introdotto un problema e quando, e molto altro ancora. Usare un VCS, in generale, significa anche che se fai un pasticcio o perdi qualche file, puoi facilmente recuperare la situazione. E ottieni tutto questo con poca fatica."

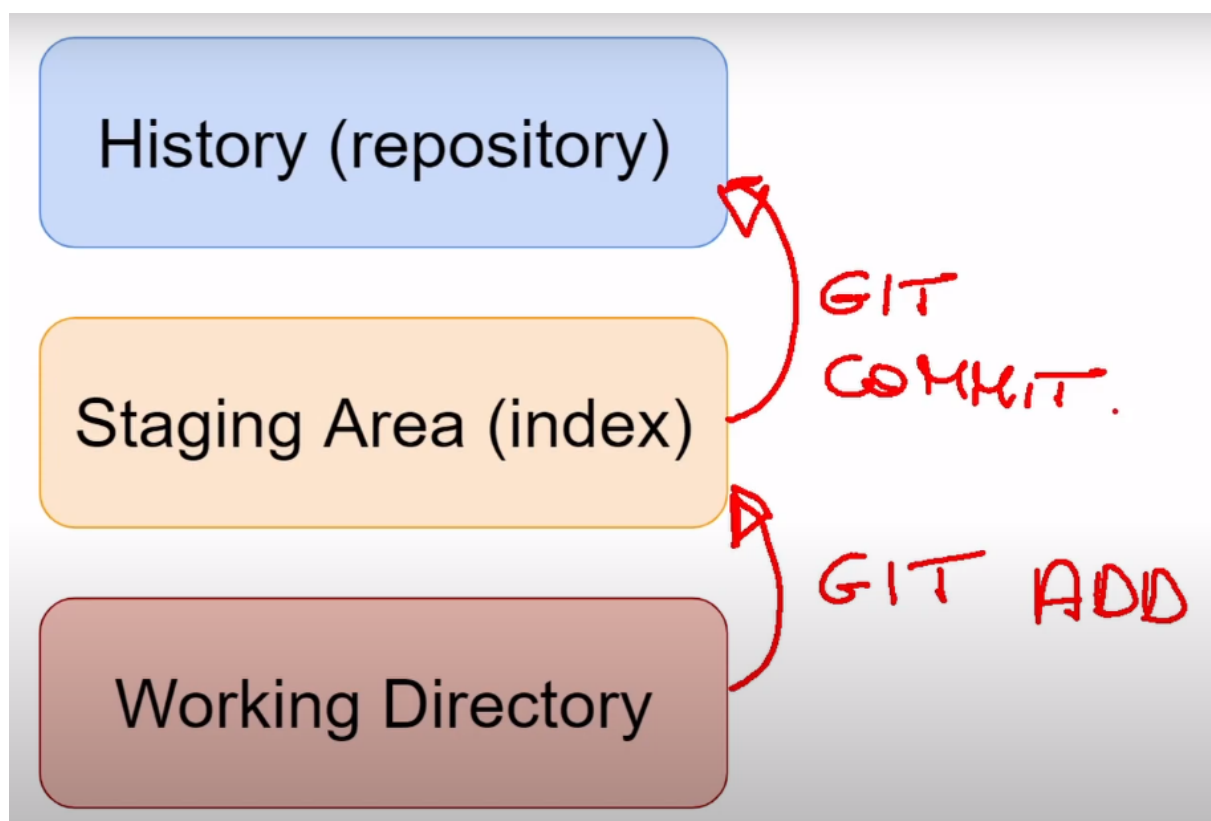
---

### CREAZIONE FILE SU GIT

- creazione nuova cartella → **mkdir nome\_cartella**
- visualizziamo il contenuto compresi file nascosti → **ls -a**
- crea un file → **touch nome\_file.formato**
- crea un file con del testo → **echo [test di prova creazione file] > nome\_file.formato**
- per vedere il contenuto del file → **cat nome\_file.formato**
  
- inizializziamo git → **git init**
- per verificare lo stato dei file del nostro progetto su git → **git status**
- per aggiungerlo alla index/staging di git tutti i file → **git add .**
- per aggiungere uno specifico file alla index/staging → **git add nome\_file.formato**
- per aggiungere una serie di file sulla base del loro formato → **git add \*.txt**

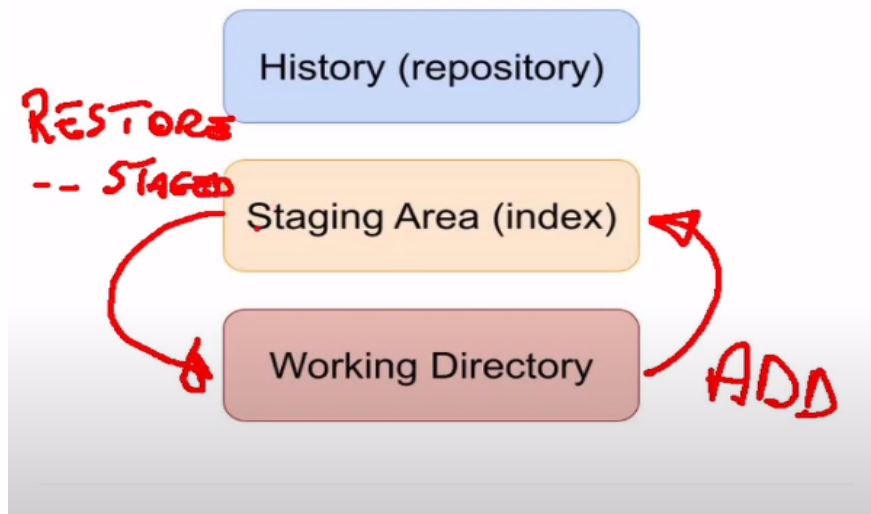
Viene utilizzato per creare un'istantanea delle modifiche effettuate nella cronologia di un progetto Git  
→ **git commit -m "First commit" -m "Descrizione del commit"**

### SCHEMA COMANDI



rimozione file dalla staging area ( index )

→ `git rm --cached nome_file.formato`



ci serve per vedere lo storico dei commit eseguiti → `git log`

```
lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master)
$ git log
commit 6189a33b9145088847364ed58a34bb9dc5db492a (HEAD -> master)
Author: Pio Lav <piolavorgna@gmail.com>
Date: Tue Oct 17 13:48:27 2023 +0200

    Second commit

    Description second commit of test

commit 6812b9dc1287741f76b812200b04724863530c00
Author: Pio Lav <piolavorgna@gmail.com>
Date: Tue Oct 17 13:32:00 2023 +0200

    First commit

    Description first commit

lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master)
$
```

key : sha1 [ 6189a33b9145088847364ed58a34bb9dc5db492a ]

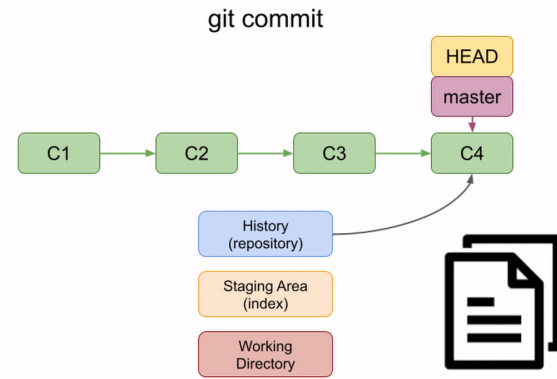
posizione : HEAD → master

per compattare il log e ottenere il titolo della commit → `git log --oneline`

```
lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master)
$ git log --oneline
6189a33 (HEAD -> master) Second commit
6812b9d First commit
```

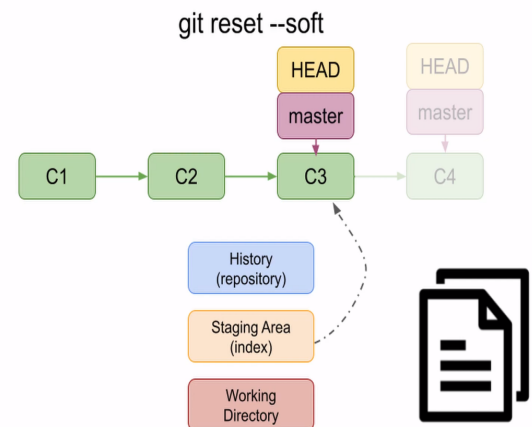
per annullare l'ultima commit in locale ( solo se non si ha fatto il push su un server ) cancella lo storico della commit

- **git commit --amend**  
per uscire → **:wq**



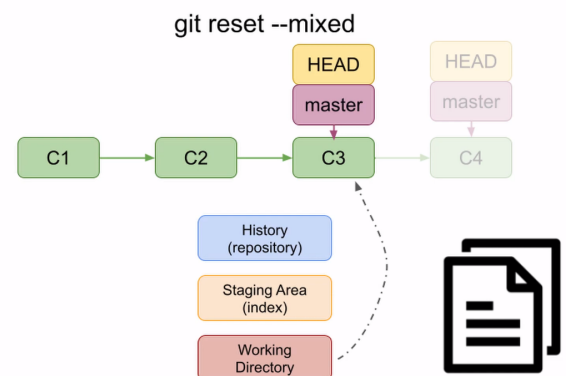
elimina la commit e riporta i file alla staging area. appena prima del comando **[commit]** e subito dopo il comando **[ add ]**

- **git reset [ --soft ] ID\_COMMIT**
- **git reset [ --soft ] HEAD^**(in base a quante ^ si usano si tornerà indietro )
- **git reset [ --soft ] HEAD~2** ( si tornerà indietro di 2 )



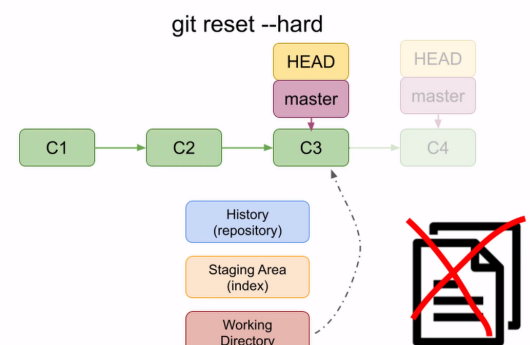
viene presa di default se non si indica un'opzione e rimuove i file dalla staging area

- **git reset [--mixed] ID\_COMMIT**
- **git reset [--mixed] HEAD^**(in base a quante ^ si usano si tornerà indietro )
- **git reset [--mixed] HEAD~2** ( si tornerà indietro di 2 )



va a cancellare tutti i file dell'ultima commit eliminando tutti i file nella working directory

- **git reset [--hard ] ID\_COMMIT**
- **git reset [--hard ] HEAD^**(in base a quante ^ si usano si tornerà indietro)
- **git reset [--hard ] HEAD~2** ( si tornerà indietro di 2 )
- **git revert <commit>**  
Revert è un comando che permette di annullare le modifiche fatte in un determinato commit.



Si crea il file [.gitignore] → touch .gitignore

al suo interno andremo a scrivere i file che non deve considerare:

- nome\_file.formato
- \*.formato\_del\_file

Si può usare il simbolo [ ! ] per aggiungere un'eccezione su determinati file:

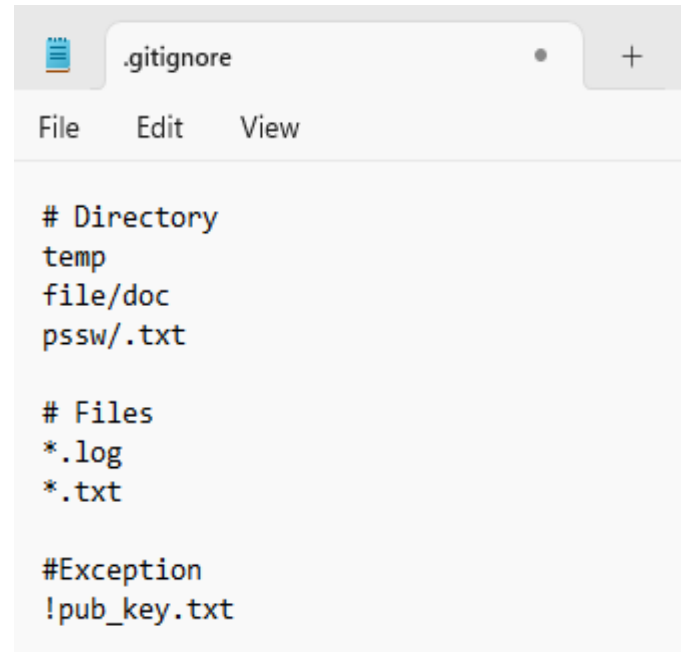
- !nome\_file.formato

per escludere le cartelle/percorsi:

- nome\_cartella
- nome\_cartella/nome\_sotto\_cartella
- nome\_cartella/nome\_sotto\_cartella/\*.formato\_del\_file

per aggiungere commenti:

- # Testo del commento



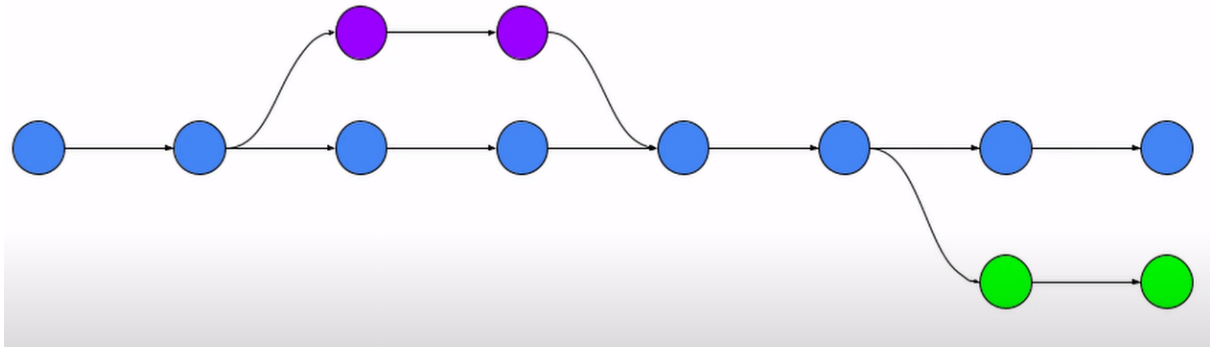
The screenshot shows a code editor window with a single tab titled ".gitignore". The editor has a menu bar with "File", "Edit", and "View". The content of the file is as follows:

```
# Directory
temp
file/doc
pssw/.txt

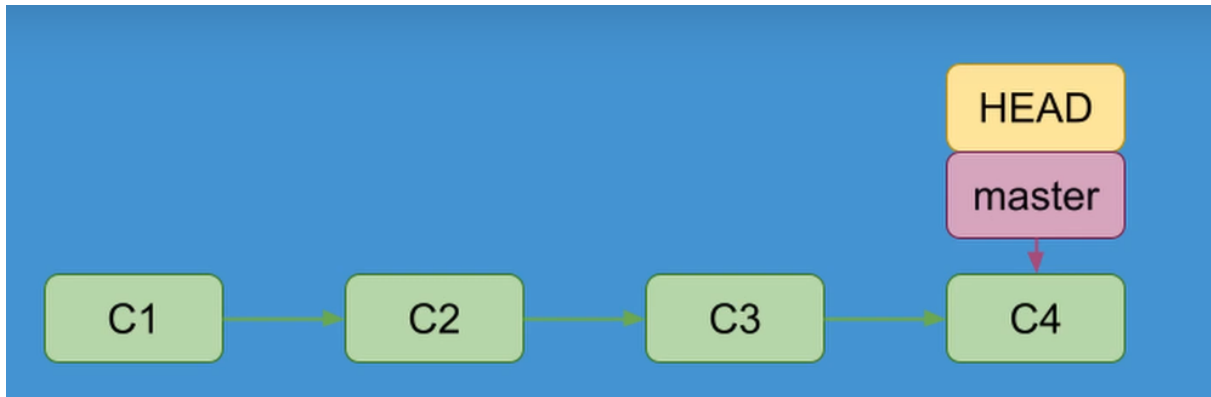
# Files
*.log
*.txt

#Exception
!pub_key.txt
```

## Git branch, checkout, merge



Il **branch** dà la possibilità di lavorare su versione parallele dei nostri file in modo da separare i vari sviluppi

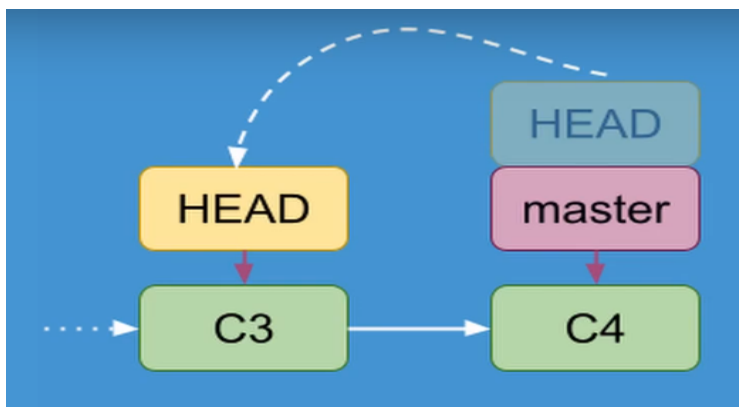


```
lavoro@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master)
$ git log --oneline
1de719f (HEAD -> master) Update gitignore
079bb5c Added Gitingore
6812b9d First commit
```

**Head** punta a **master** e **master** punta al nostro ultimo **commit**

**NB:** master è un nome di convenzione ed è possibile cambiare il suo nome

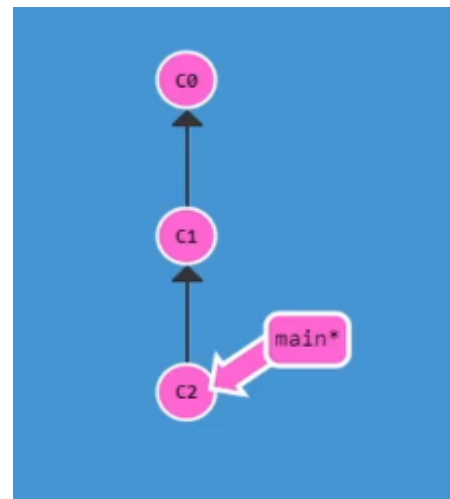
### Esempio di testa distaccata



Per vedere il percorso che stiamo seguendo con le nostre commit si usa il seguente comando:

→ **git log --decorate --oneline --graph**

```
lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master)
$ git log --all --decorate --oneline --graph
* 1de719f (HEAD -> master) Update gitignore
* 079bb5c Added Gitingore
* 6812b9d First commit
```



per evitare di scrivere ogni volta stringhe di codice lunghe si può usare **alias** per salvarla:

→ **alias graph='git log --all --decorate --oneline --graph'**

```
lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master)
$ graph
* 1de719f (HEAD -> master) Update gitignore
* 079bb5c Added Gitingore
* 6812b9d First commit
```

Per creare nuovi branch possiamo usare il seguente comando:

→ **git branch nome\_del\_branch**

Per vedere i branch creati:

→ **git branch**

**NB:** \*(head) indica il branch attivo

```
lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master)
$ git branch fix

lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master)
$ git branch dev

lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master)
$ git branch
dev
fix
* master
```

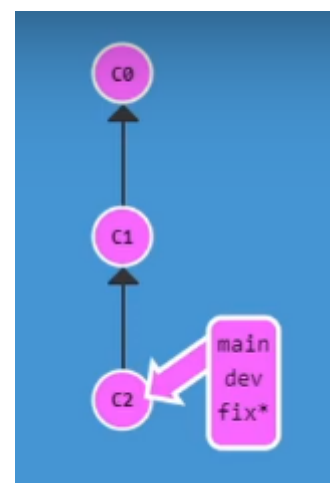
```
lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master)
$ graph
* 1de719f (HEAD -> master, fix, dev) Update gitignore
* 079bb5c Added Gitingore
* 6812b9d First commit
```

per poter spostare il puntatore HEAD:

→ **git checkout fix**

```
lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master)
$ git checkout fix
Switched to branch 'fix'

lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (fix)
$ graph
* 1de719f (HEAD -> fix, master, dev) Update gitignore
* 079bb5c Added Gitingore
* 6812b9d First commit
```



Per usare l'editor integrato con la bash:

→ **vi nome\_file.formato**

→ **i**

→ **esc**

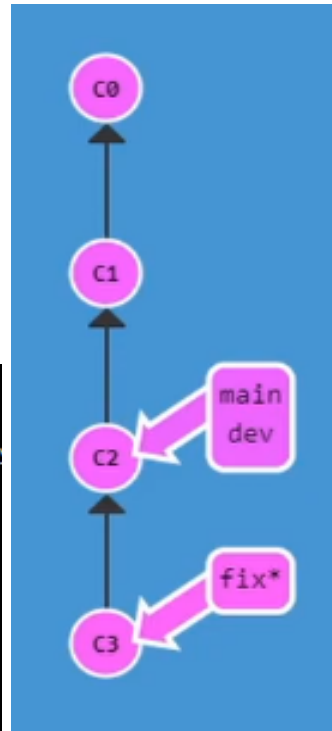
→ **:wq**

Per velocizzare i tempi di 'add' e 'commit':

→ **git commit -am 'Added new line'**

```
lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (fix)
$ git commit -am "Added new line"
warning: in the working copy of 'file01.txt', LF will be replaced by
[fix a7df2e2] Added new line
1 file changed, 1 insertion(+)

lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (fix)
$ graph
* a7df2e2 (HEAD -> fix) Added new line
* 1de719f (master, dev) Update gitignore
* 079bb5c Added Gitingore
* 6812b9d First commit
```

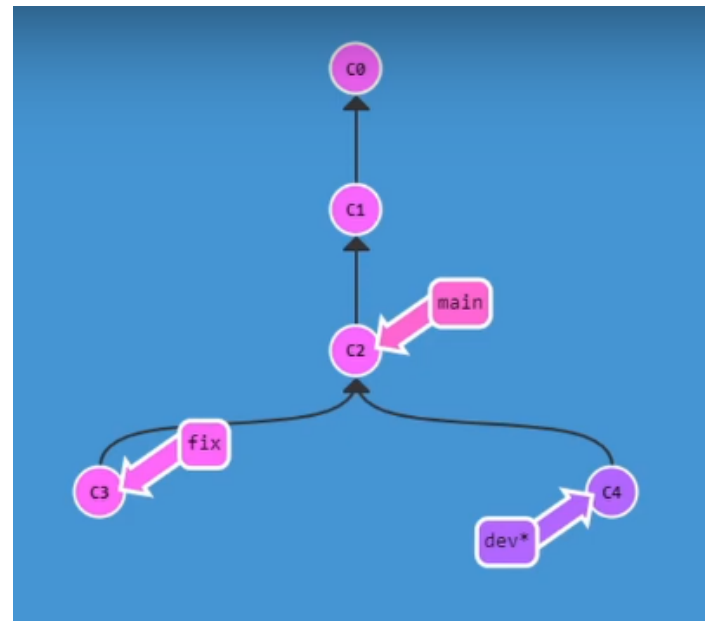


```
MINGW64:/c:/Users/lavor/Desktop/Private/
Primo file di test
Fix line from text vñ

File01.txt [unix] (23:18 17/10/2023)
"file01.txt" [unix] 2L, 41B
```

Spostandosi sul branch **dev** e apportando la stessa modifica si avrà il seguente schema

```
lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (dev)
$ graph
* 00a12bd (HEAD -> dev) Added new line from dev
| * a7df2e2 (fix) Added new line
|/
* 1de719f (master) Update gitignore
* 079bb5c Added Gitingore
* 6812b9d First commit
```





Per unire le modifiche fatte in **DEV** e **FIX** su **MASTER**

→ **git checkout master**

→ **git diff master..fix** [ controlliamo se ci sono differenze (compare) ]

```
lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master)
$ git diff master..fix
diff --git a/file01.txt b/file01.txt
index a21cf83..dfdf94c 100644
--- a/file01.txt
+++ b/file01.txt
@@ -1,2 @@
 Primo file di test
+Fix line from text vi
```

```
lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master)
$ graph
* 00a12bd (dev) Added new line from dev
| * a7df2e2 (fix) Added new line
|/
* 1de719f (HEAD -> master) Update gitignore
* 079bb5c Added Gitingore
* 6812b9d First commit
```

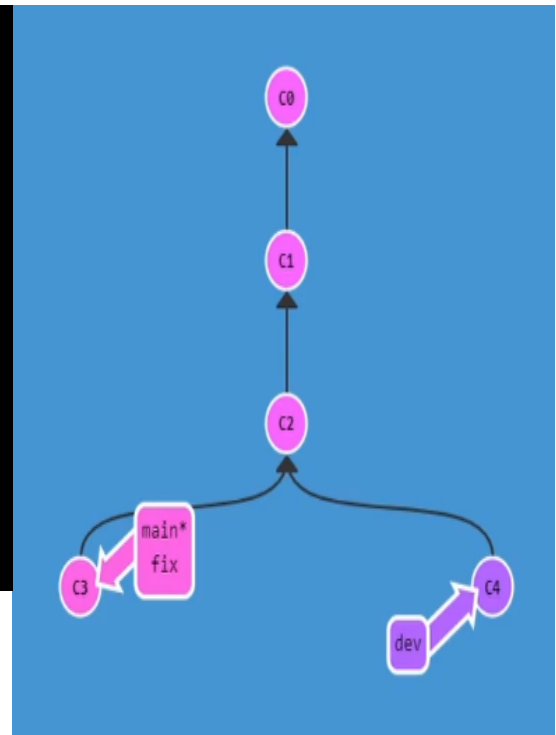
uniamo branch **fix** all'interno di **MASTER** ( merge fast forward )

→ **git merge fix**

```
lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master)
$ git merge fix
Updating 1de719f..a7df2e2
Fast-forward
 file01.txt | 1 +
 1 file changed, 1 insertion(+)

lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master)
$ graph
* 00a12bd (dev) Added new line from dev
| * a7df2e2 (HEAD -> master, fix) Added new line
|/
* 1de719f Update gitignore
* 079bb5c Added Gitingore
* 6812b9d First commit

lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master)
$ cat file01.txt
Primo file di test
Fix line from text vi
```



Per vedere quali branch sono stati uniti si usa il comando:

→ **git branch -merged**

Così sarà possibile eliminare un branch dopo averlo unito

→ **git branch -d fix**

Per cancellare un branch non ancora unito e tutti i file al suo interno:

→ **git branch -D nome\_branch**

Nel caso si cercasse di fare il merge con dev verso master usando il comando merge avremmo il seguente risultato:

```
lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master|MERGING)
$ git merge dev
error: Merging is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.

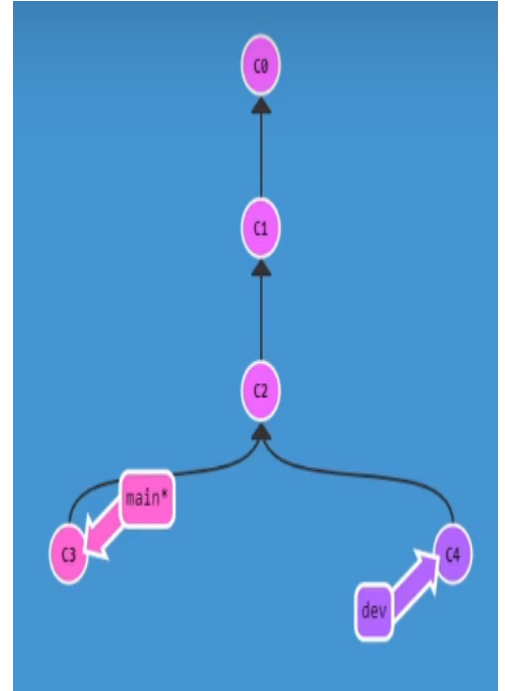
lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master|MERGING)
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   file01.txt

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

lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master|MERGING)
$ git merge --abort

lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master)
$
```



Per risolvere il conflitto ripetiamo il comando merge:

→ **git merge fix**

→ **vi**

[ ===== ] dividono le differenze che ci sono tra HEAD e DEV

una volta modificato il file con i valori che ci interessano

→ **git add .**

git ci conferma che i conflitti sono stati risolti

```
lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master|MERGING)
$ git status
On branch master
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:
  modified:   file01.txt

lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master|MERGING)
$
```

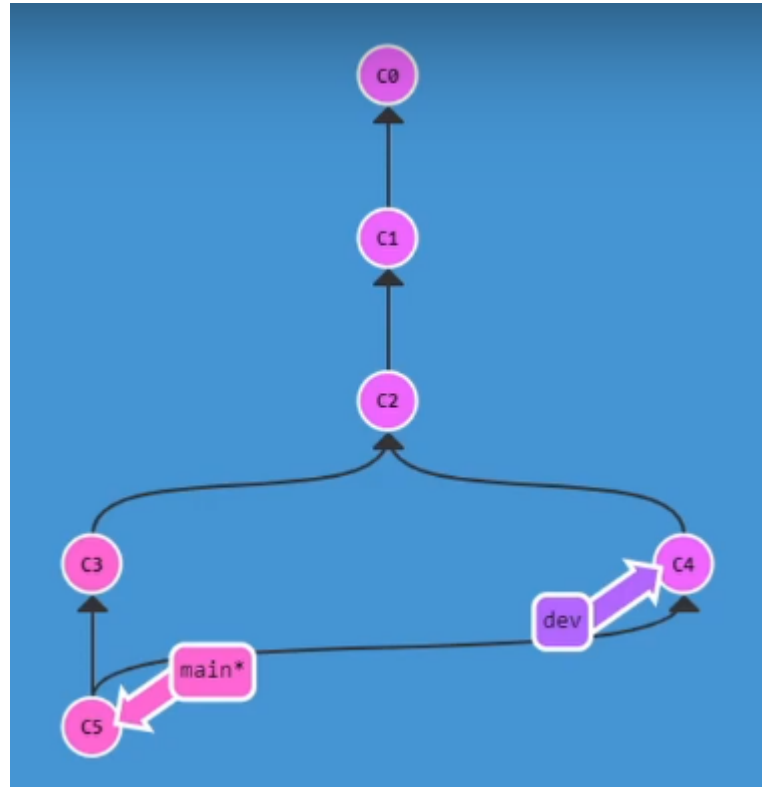
→ **git commit**

```
lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master|MERGING)
$ git status
On branch master
All conflicts fixed but you are still merging.
(use "git commit" to conclude merge)

Changes to be committed:
  modified:   file01.txt

lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master|MERGING)
$ git commit
[master a5292df] Merge branch 'dev'

lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master)
$ graph
* a5292df (HEAD -> master) Merge branch 'dev'
| \
| * 00a12bd (dev) Added new line from dev
| * | a7df2e2 Added new line
| /
| * 1de719f Update gitignore
| * 079bb5c Added Gitingore
| * 6812b9d First commit
```

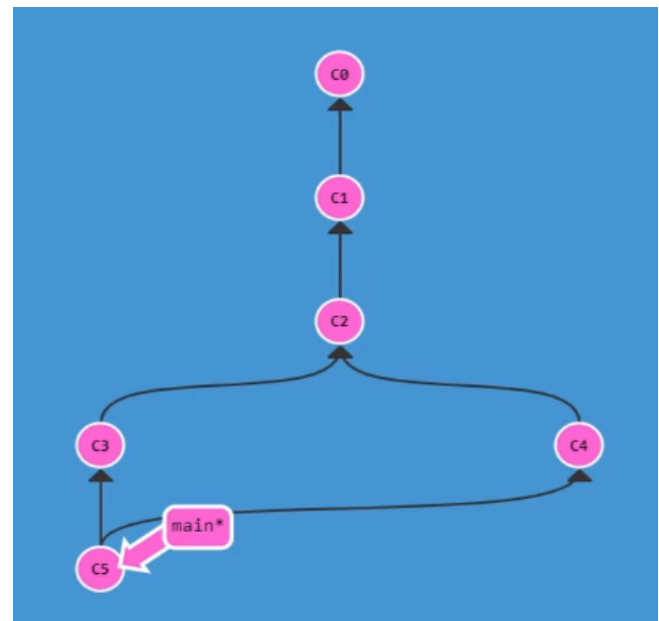


una volta uniti i branch possiamo eliminare dev:

→ **git branch -d dev**

```
lavor@MrCix MINGW64 ~/Desktop/Private/git/random_people/repo (master)
$ graph
* a5292df (HEAD -> master) Merge branch 'dev'
| \
| * 00a12bd Added new line from dev
| * | a7df2e2 Added new line
| /
| * 1de719f Update gitignore
| * 079bb5c Added Gitingore
| * 6812b9d First commit
```

**NB:** per conservare lo storico non si cancellano mai i branch, in caso di errata creazione di un branch è possibile usare i passaggi indicati

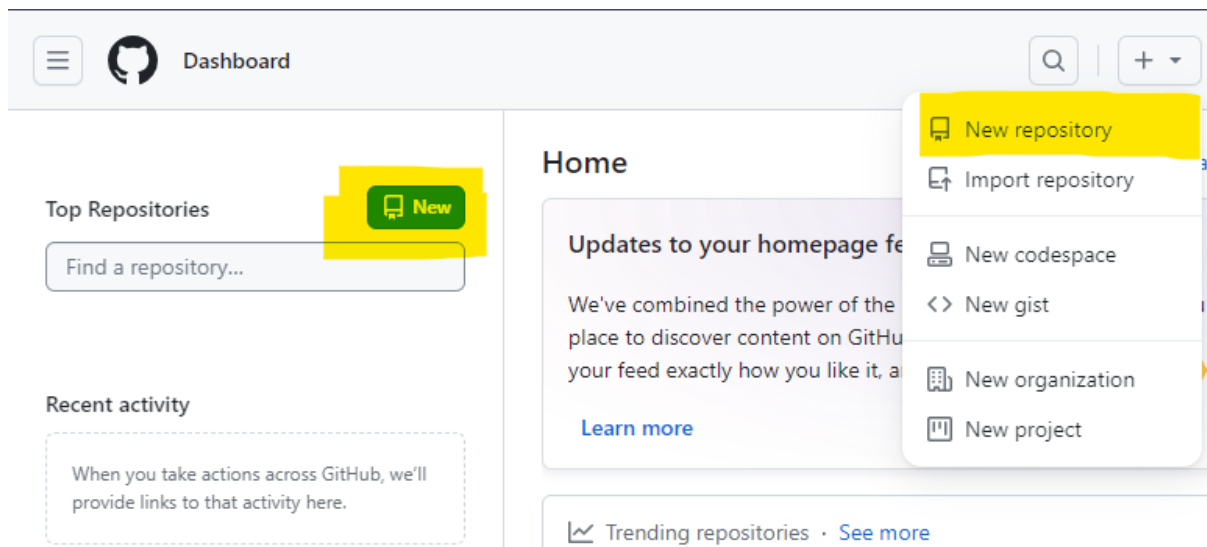


Per poter velocizzare la creazione del branch e spostarsi sul nuovo branch:

→ **git checkout -b nome\_del\_branch**

## CARICAMENTO DELLA REPOSITORY SU GITHUB

Dopo aver effettuato la registrazione/accesso su github si deve prima creare una nuova repository cliccando sul bottone New o dal menu a tendina [ New repository ]:



Una volta cliccato ci troveremo sulla seguente pagina dove andremo ad inserire inizialmente i valori obbligatori:

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

*Required fields are marked with an asterisk (\*).*

Owner \* PioLavorgna /

Great repository names are short and memorable. Need inspiration? How about [ubiquitous-succotash](#) ?

Description (optional)

- ☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.
- ☐ **Private**  
You choose who can see and commit to this repository.

#### Initialize this repository with:

- ☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

#### Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

#### Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

Ci ritroveremo sulla seguente pagina

### Quick setup — if you've done this kind of thing before

Set up in Desktop

 or 

HTTPS

SSH

https://github.com/PioLavorgna/git-training-docs.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

### ...or create a new repository on the command line

```
echo "# git-training-docs" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/PioLavorgna/git-training-docs.git
git push -u origin main
```

### ...or push an existing repository from the command line

```
git remote add origin https://github.com/PioLavorgna/git-training-docs.git
git branch -M main
git push -u origin main
```

### ...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

ProTip! Use the URL for this page when adding GitHub as a remote.

## Info comandi:

→ **git remote add [ origin ] <https://github.com/PioLavorgna/git-training-docs.git>**

ci permette di salvare URL sulla variabile origin così da non doverla riscrivere ogni volta

→ **git branch -M main**

il branch sul quale fare l'upload dei nostri file

→ **git push -u origin main**

ci permette di settare un server remote di default così da usare solo il comando **[ git push ]**

```
lavor@MrCix MINGW64 ~/Desktop/Private/Formazione/corso_git/repo (master)
$ git remote add origin https://github.com/PioLavorgna/git-training-docs.git

lavor@MrCix MINGW64 ~/Desktop/Private/Formazione/corso_git/repo (master)
$ git push -u origin master
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 4 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (18/18), 1.60 KiB | 410.00 KiB/s, done.
Total 18 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/PioLavorgna/git-training-docs.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

**NB:** la prima volta vi chiederà di autenticarvi aprendo una modale con le istruzioni da seguire

Aggiornando la pagina di github si avrà la seguente schermata:

PioLavorgna / git-training-docs

🔍

+

🕒

🔗

📁

🌱

<> Code

🔗 Issues

🔗 Pull requests

🔗 Actions

📁 Projects

📖 Wiki

🛡 Security

📈 Insights

⚙ Settings

git-training-docs

Public

📌 Pin

👁 Unwatch 1

🔗 Fork 0

★ Star 0

🔗 master

Go to file

Add file

<> Code

About

🔗 Branches

🏷 Tags

🌱 PioLavorgna Merge branch 'dev' ...

14 hours ago 6

📄 .gitignore

Update gitignore

20 hours ago

📄 file01.txt

Merge branch 'dev'

14 hours ago

Help people interested in this repository understand your project by adding a README.

Add a README

📄 Activity

★ 0 stars

👁 1 watching

🔗 0 forks

Per controllare la lista dei nomi ( server ) che abbiamo salvato in precedenza:

→ **git remote**

```
lavor@MrCix MINGW64 ~/Desktop/Private/Formazione/corso_git/repo (master)
$ git remote
origin
```

→ **git remote -v**

ci darà lista completa e con l'URL

```
lavor@MrCix MINGW64 ~/Desktop/Private/Formazione/corso_git/repo (master)
$ git remote -v
origin https://github.com/PioLavorgna/git-training-docs.git (fetch)
origin https://github.com/PioLavorgna/git-training-docs.git (push)
```

→ **cat [ percorso del file .config ]**

comando per vedere dove sono salvate le informazioni sulla nostra repository in locale

```
lavor@MrCix MINGW64 ~/Desktop/Private/Formazione/corso_git/repo (master)
$ cat .git/config
[core]
    repositoryformatversion = 0
    filemode = false
    bare = false
    logallrefupdates = true
    symlinks = false
    ignorecase = true
[remote "origin"]
    url = https://github.com/PioLavorgna/git-training-docs.git
    fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
    remote = origin
    merge = refs/heads/master
```

→ **git remote show origin**

ci permetterà di avere maggiori informazioni sulle configurazioni del server

```
lavor@MrCix MINGW64 ~/Desktop/Private/Formazione/corso_git/repo (master)
$ git remote show origin
* remote origin
Fetch URL: https://github.com/PioLavorgna/git-training-docs.git
Push URL: https://github.com/PioLavorgna/git-training-docs.git
HEAD branch: master
Remote branch:
  master tracked
Local branch configured for 'git pull':
  master merges with remote master
Local ref configured for 'git push':
  master pushes to master (up to date)
```

→ **git remote rename [new\_name]**

per modificare il nome del server in locale

→ **git remote remove [name-server]**

per rimuovere un server dalle configurazioni in locale

### → git fetch

il comando che dice al tuo git locale di ottenere i meta-dati più recenti dall'originale (ma non fa nessun trasferimento di file. È più che altro per vedere se ci sono cambiamenti disponibili)

```
lavor@MrCix MINGW64 ~/Desktop/Private/Formazione/corso_git/repo (master)
$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 734 bytes | 48.00 KiB/s, done.
From https://github.com/PioLavorgna/git-training-docs
  2a78cfe..31b1de5  master    -> origin/master

lavor@MrCix MINGW64 ~/Desktop/Private/Formazione/corso_git/repo (master)
$ git status
On branch master
Your branch is behind 'origin/master' by 3 commits, and can be fast-forwarded.
  (use "git pull" to update your local branch)

nothing to commit, working tree clean

lavor@MrCix MINGW64 ~/Desktop/Private/Formazione/corso_git/repo (master)
$ graph
* 31b1de5 (origin/master) Update README.md
* 2a78cfe Update README.md
* cf436e5 Create README.md
* a5292df (HEAD -> master) Merge branch 'dev'
| \
| * 00a12bd Added new line from dev
* | a7df2e2 Added new line
| /
* 1de719f Update gitignore
* 079bb5c Added Gitingore
* 6812b9d First commit

lavor@MrCix MINGW64 ~/Desktop/Private/Formazione/corso_git/repo (master)
$
```

Si può notare che **master** è la “commit” precedente a **origin/master**

Per unire le modifiche si utilizza il comando:

### → git merge origin/master

Per poi verificare che il merge sia andato a buon fine

```
lavor@MrCix MINGW64 ~/Desktop/Private/Formazione/corso_git/repo (master)
$ graph
* 31b1de5 (HEAD -> master, origin/master) Update README.md
* 2a78cfe Update README.md
* cf436e5 Create README.md
* a5292df Merge branch 'dev'
| \
| * 00a12bd Added new line from dev
* | a7df2e2 Added new line
| /
* 1de719f Update gitignore
* 079bb5c Added Gitingore
* 6812b9d First commit
```



→ **git pull**

identico a [ git fetch ] e in più copia i cambiamenti dal repository remoto

```
lavor@MrCix MINGW64 ~/Desktop/Private/Formazione/corso_git/repo (master)
$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 1.32 KiB | 89.00 KiB/s, done.
From https://github.com/PioLavorgna/git-training-docs
  31b1de5..9403902 master    -> origin/master
Updating 31b1de5..9403902
Fast-forward
 LICENSE | 21 ++++++
 1 file changed, 21 insertions(+)
 create mode 100644 LICENSE

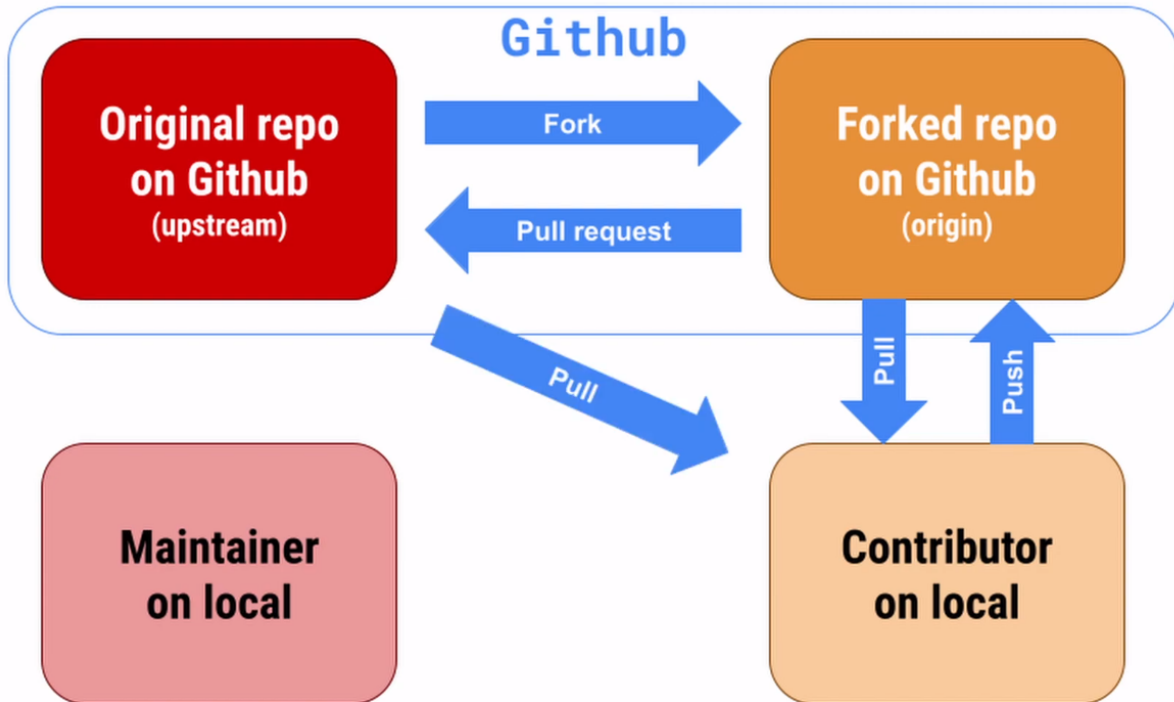
lavor@MrCix MINGW64 ~/Desktop/Private/Formazione/corso_git/repo (master)
$ git graph
* 9403902 (HEAD -> master, origin/master) Create LICENSE
* 31b1de5 Update README.md
* 2a78cfe Update README.md
* cf436e5 Create README.md
* a5292df Merge branch 'dev'
| \
| * 00a12bd Added new line from dev
| * | a7df2e2 Added new line
| /
* 1de719f Update gitignore
* 079bb5c Added Gitingore
* 6812b9d First commit
```

→ **git clone https://github.com/PioLavorgna/git-training-docs.git**

Git ci permette di scaricare un repository esistente da un server. Questo comando crea sul tuo computer una copia completa del progetto, completa di cronologia delle modifiche.

## GITHUB FORK AND PULL REQUEST

Un fork è una copia separata di un repository che puoi gestire e modificare senza influenzare il progetto originale. La clonazione di un repo, d'altra parte, crea semplicemente una copia locale dei file.



Per poter eseguire una sequenza di tipo fork i comandi e le istruzioni sono le seguenti:

- usare un nuovo user diverso dal principale
- **git clone** [ URL\_server\_origin ]
- **git remote add** [ nome\_variabale\_server ( ex. upstream ) ] [ URL\_server\_origin ]
- **git remote -v**
- **git fetch** [ nome\_variabale\_server ]
- **git merge** [ nome\_variabale\_server ]/master
- **git pull** [ nome\_variabale\_server ]/master
- effettuare le modifiche
- **git push**