Introduzione a Git e GitHub

Progetto ICT Forma
Istituto Statale di Istruzione Superiore "Gobetti - Volta"

Ciao!

Massimiliano Atzori

- lavoro in Develer
- programmo dal 1999
- mi occupo di progetti software e sviluppo Agile

Come contattarmi

- social **@amaxis** (Twitter)
- email **massimiliano.atzori@gmail.com**

Cosa faremo

1. Installiamo Git e vediamo com'è organizzato GitHub
2. Breve introduzione a Git
3. Primi passi con Git: clone, add, remove, commit
4. Portiamo le modifiche in remoto
5. Conflitti e merge
6. Riepilogo e consigli utili

Photo by Brendan Steeves on Unsplash

Cosa portare a casa

1. Capire che i programmatori non lavorano **mai** da soli

2. Scoprire che esistono Git e GitHub

3. Imparare i comandi base più semplici

4. Iniziare a studiare i concetti più avanzati

5. Collaborare ad un progetto open source

1. Installare Git e account su GitHub

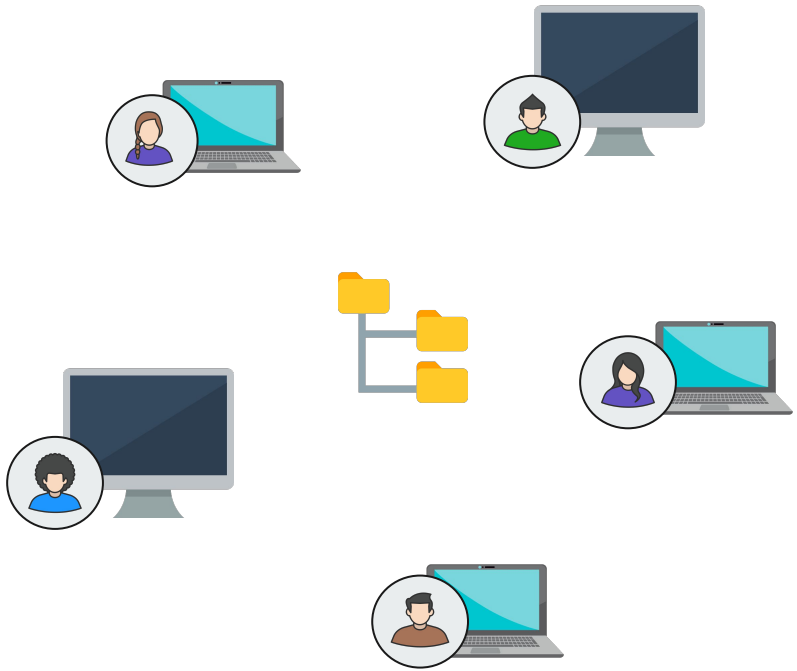
Cos'è Git?

È un **software** di c**ontrollo di versione** (VCS)

Serve a:

- Memorizzare il codice sorgente dei programmi
- Riportare i file ad uno stato precedente
- Riportare un progetto ad uno stato precedente
- Confrontare modifiche fatte nel tempo
- Capire chi ha modificato cosa

In poche parole, è un insieme di strumenti per evitare di pestarsi i piedi (il più possibile)



Linus Torvalds, Aprile 2005 (4 giorni!)

Junio Hamano, Luglio 2005 - oggi

Prende il nome da una canzone dei Beatles (I'm so tired)

Linus Torvalds

Junio Hamano

Installare Git sulla propria macchina

Installare Git

Windows	https://git-scm.com/download/win	
MacOS	https://git-scm.com/download/mac	
Linux	`sudo apt-get install git`	Debian
	`sudo yum install git`	Fedora

GitHub

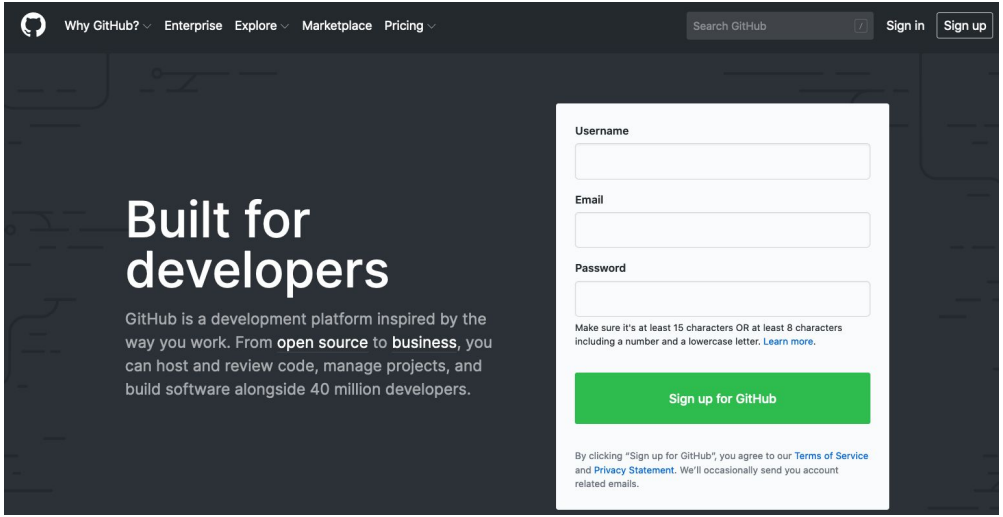
Permette di creare repository Git online

GitHub nasce nel 2008

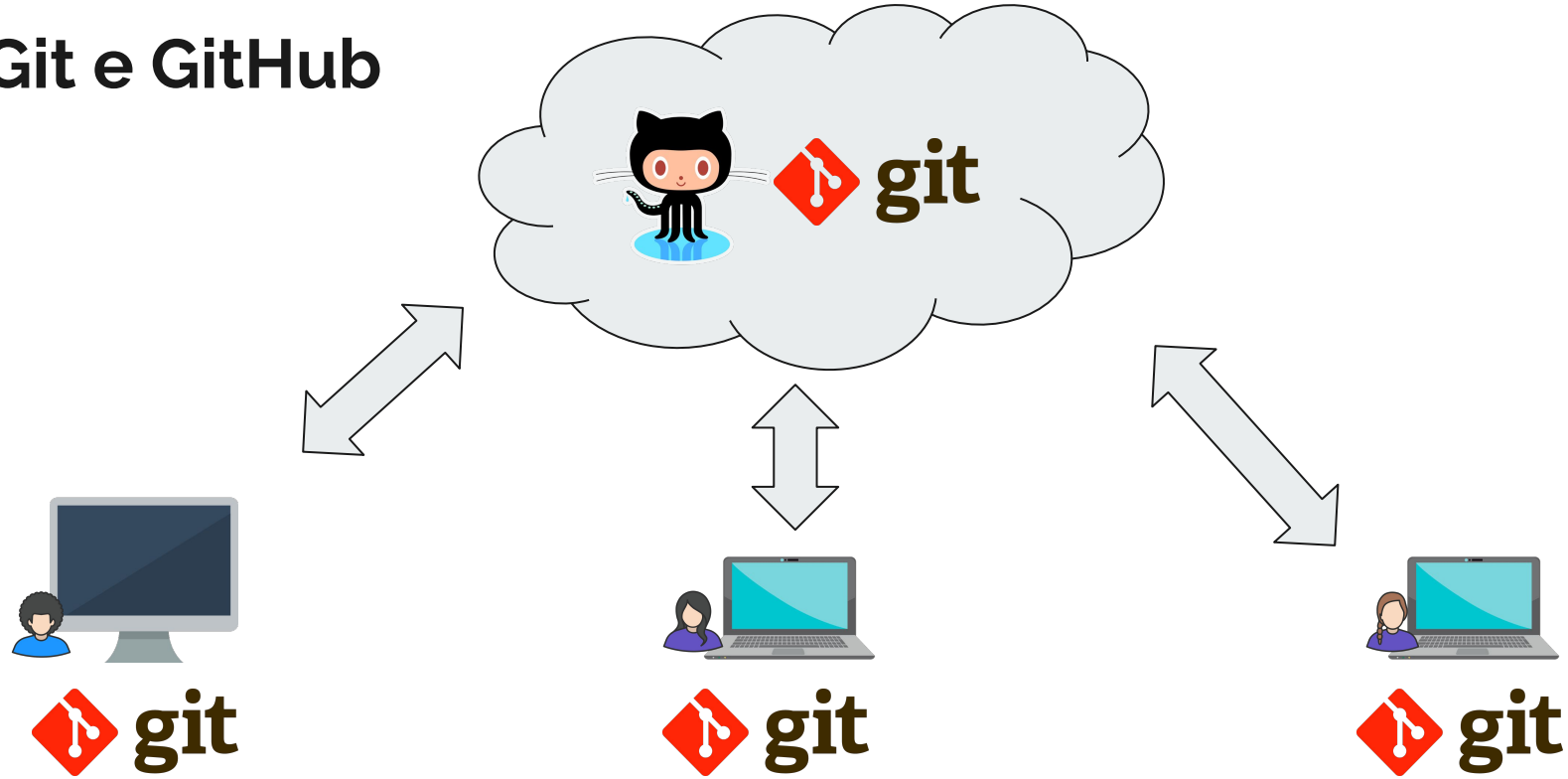
Viene acquisito da Microsoft nel 2018

Gratuito per progetti Open Source

https://github.com/



Git e GitHub



Come si usano Git e GiHub

1. Scriviamo del software con quello che ci pare (Eclipse, VSCode, Sublime Text, il blocco note e simili)

2. Quando siamo arrivati ad un punto “stabile” o significativo, chiediamo a Git di r**egistrare le modifiche**

3. Git registra le modifiche e le rende **permanenti**

4. Quando siamo pronti, chiediamo a Git di inviare le modifiche permanenti su GitHub

5. Altre persone prenderanno le nostra modifiche da Github e le useranno

2. Breve introduzione a Git

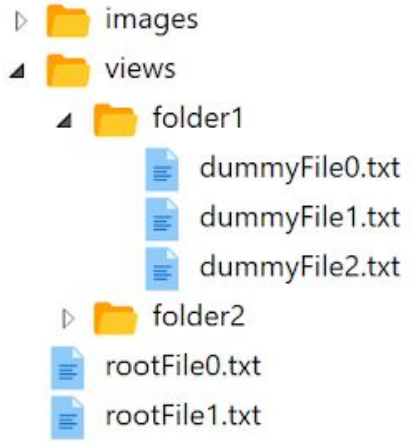
Repository

Un repository è un contenitore di file e cartelle appartenenti ad un sotware

Es. creiamo un progetto con Eclipse: tutto il codice che scriviamo lo possiamo mettere in un repository

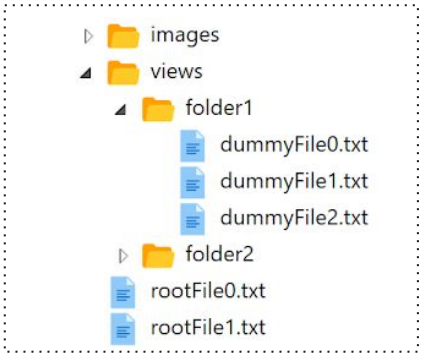
Un software può stare in più repository

git-gobetti-volta



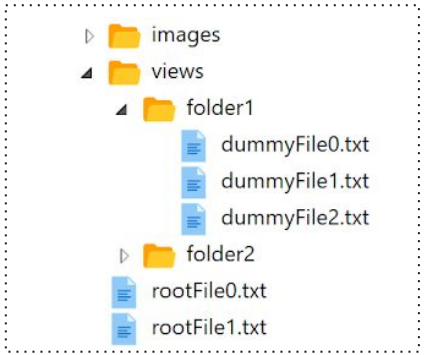
Repository

File system



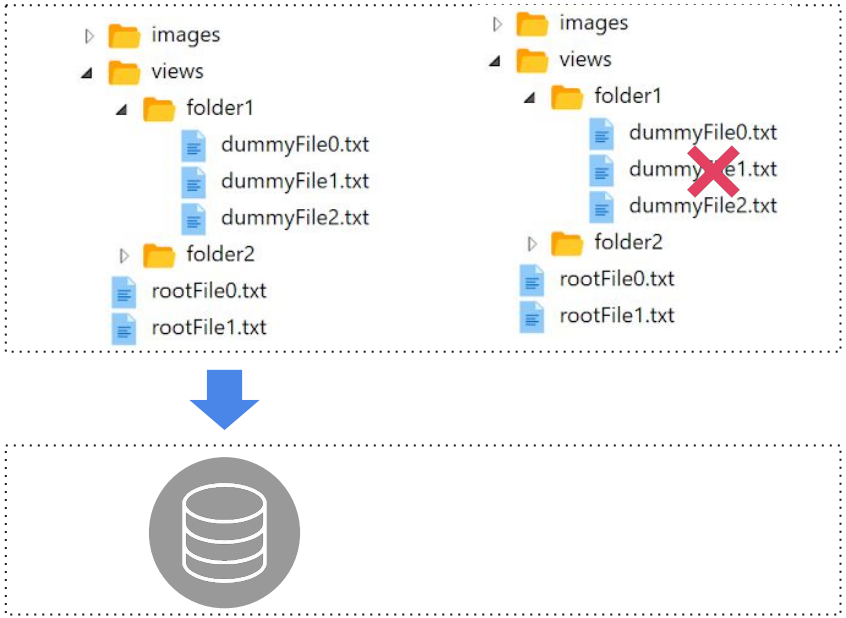
Repository

File system



Repository

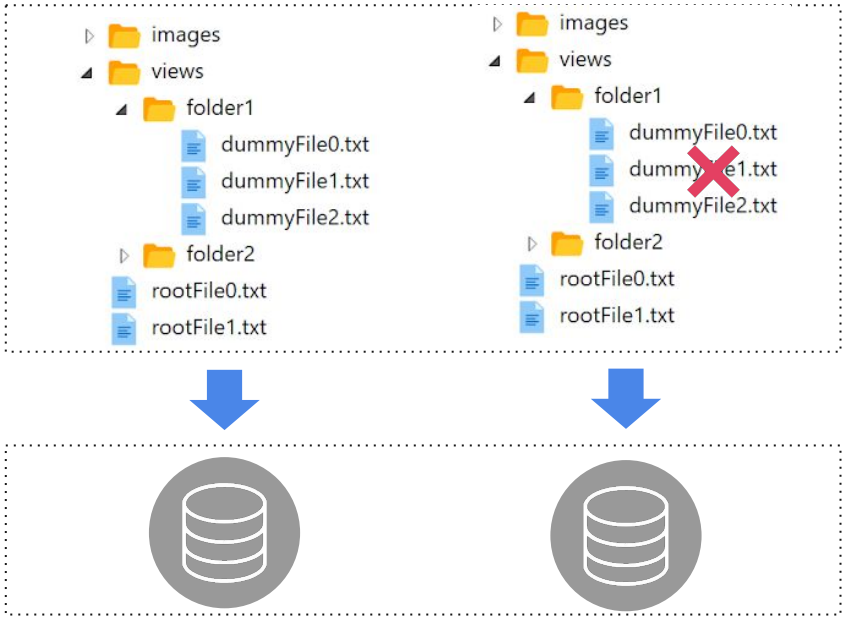
File system



git

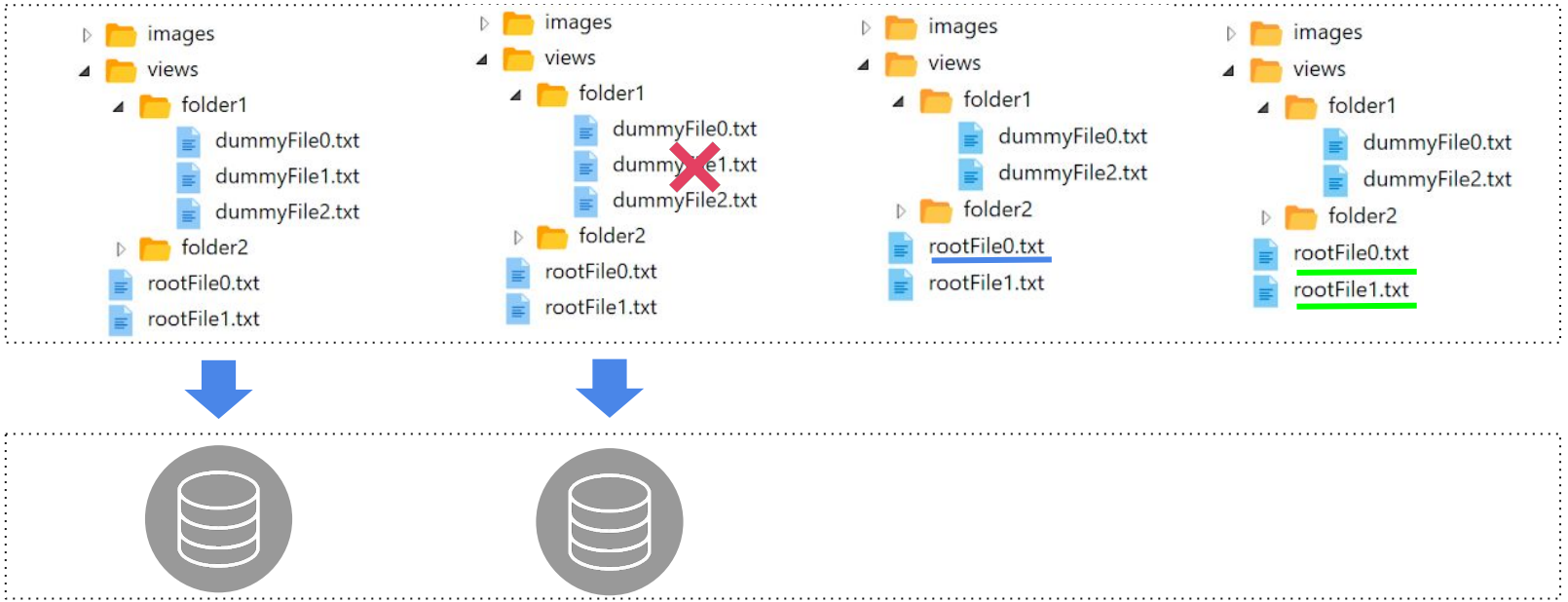
Repository

File system



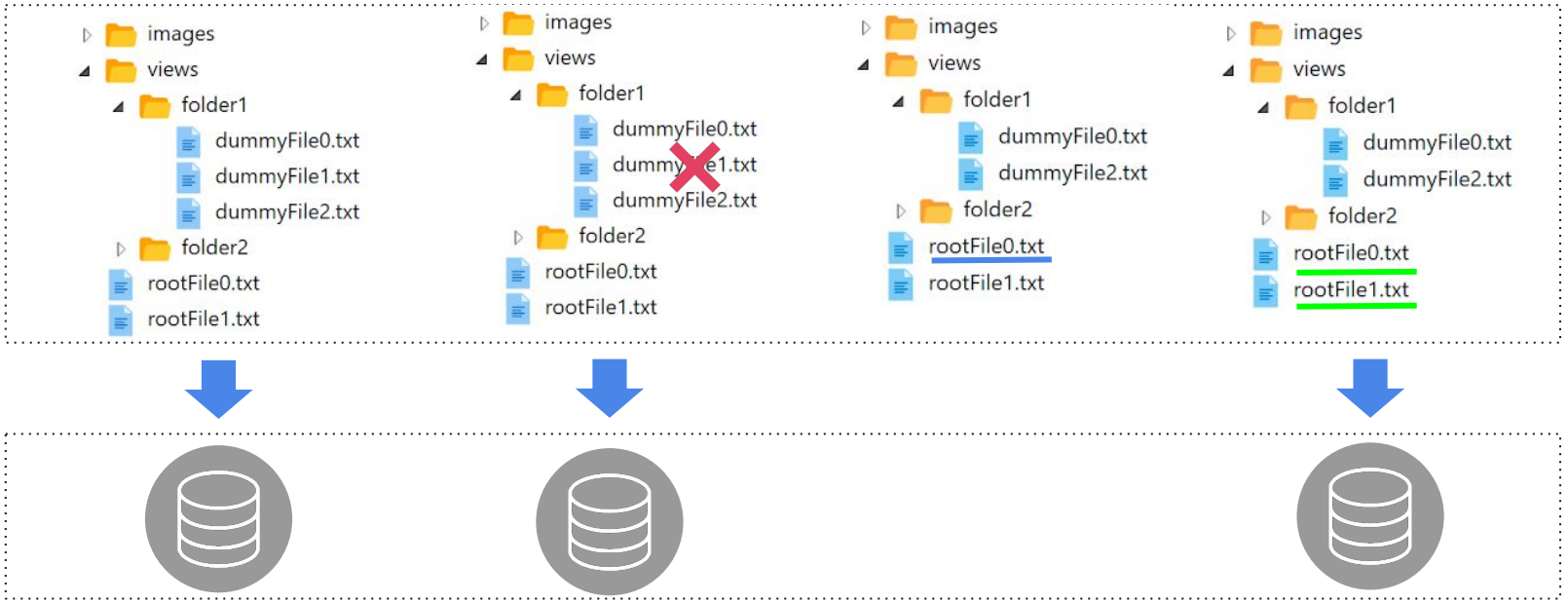
Repository

File system



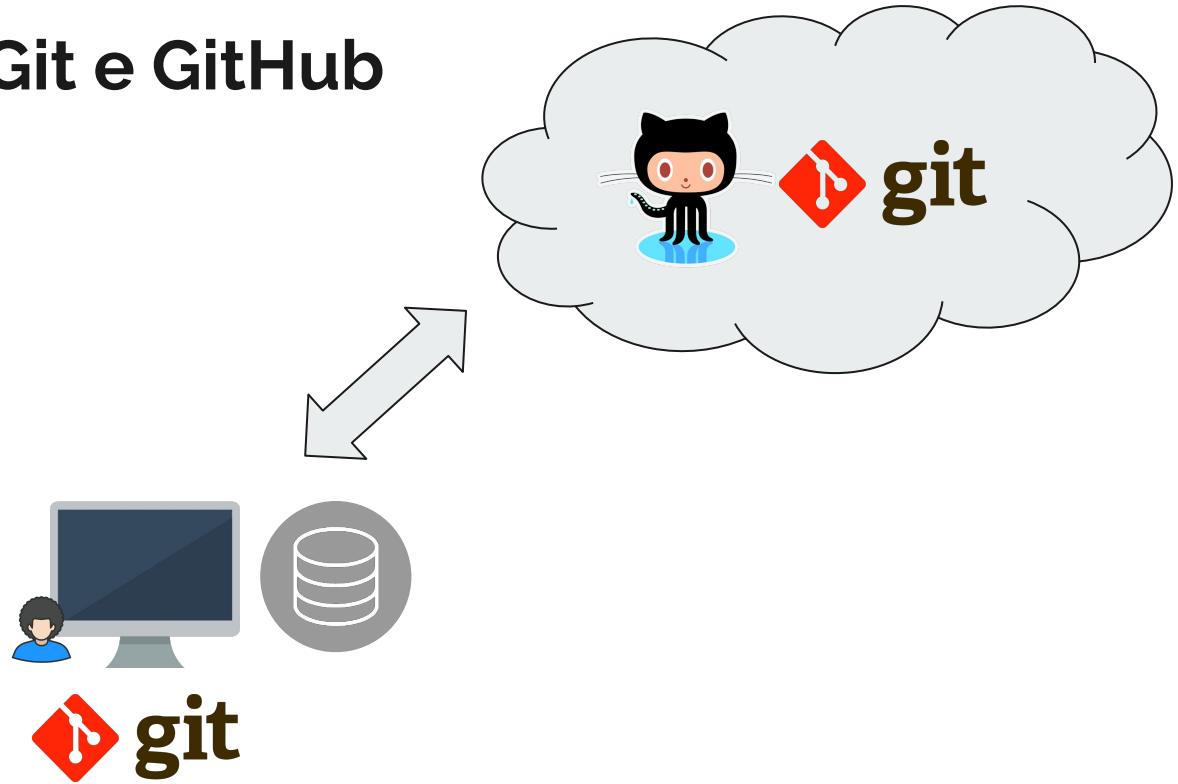
Repository

File system

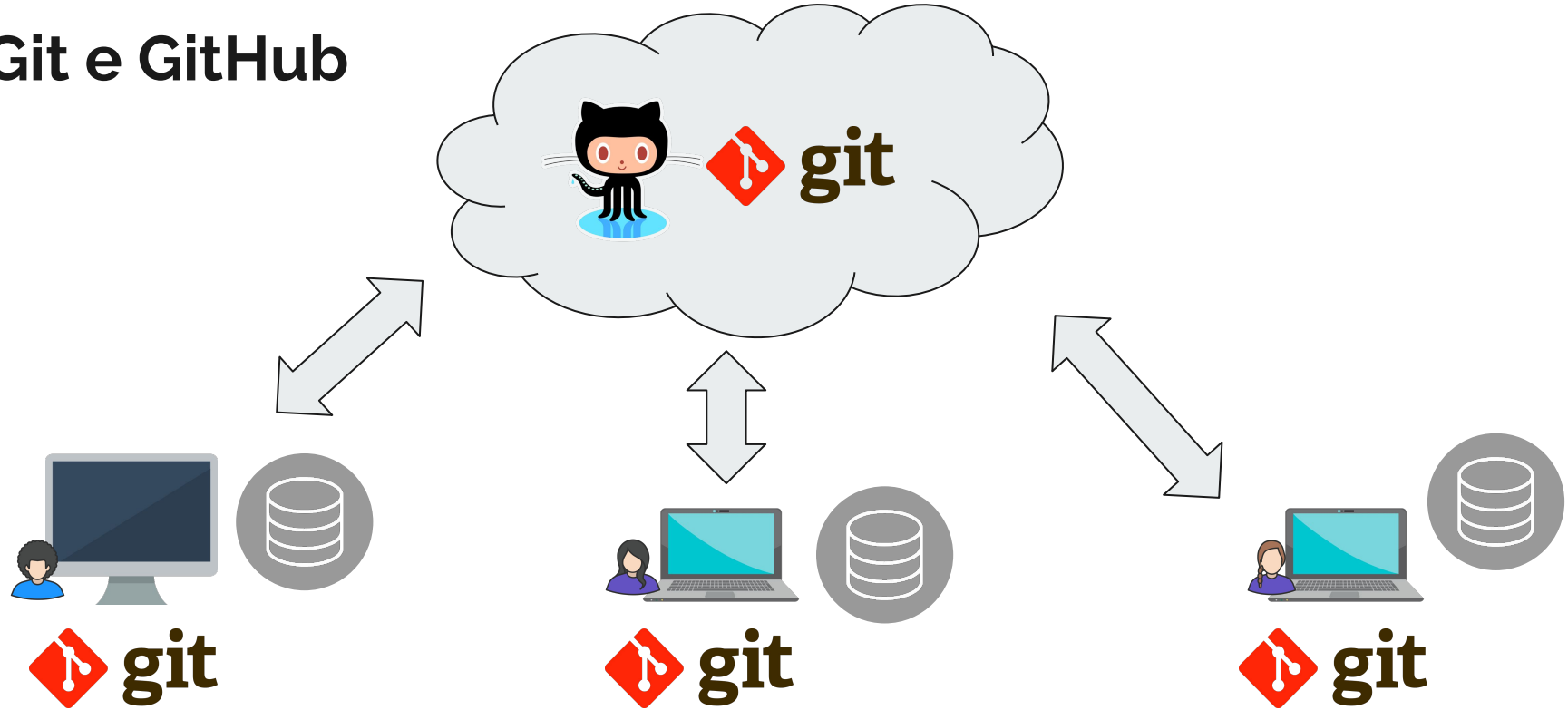


git

Git e GitHub



Git e GitHub



Primi passi con Git: clone, add, remove, commit

Primi passi con Git

Si può “parlare” con Git in **tre modi**:

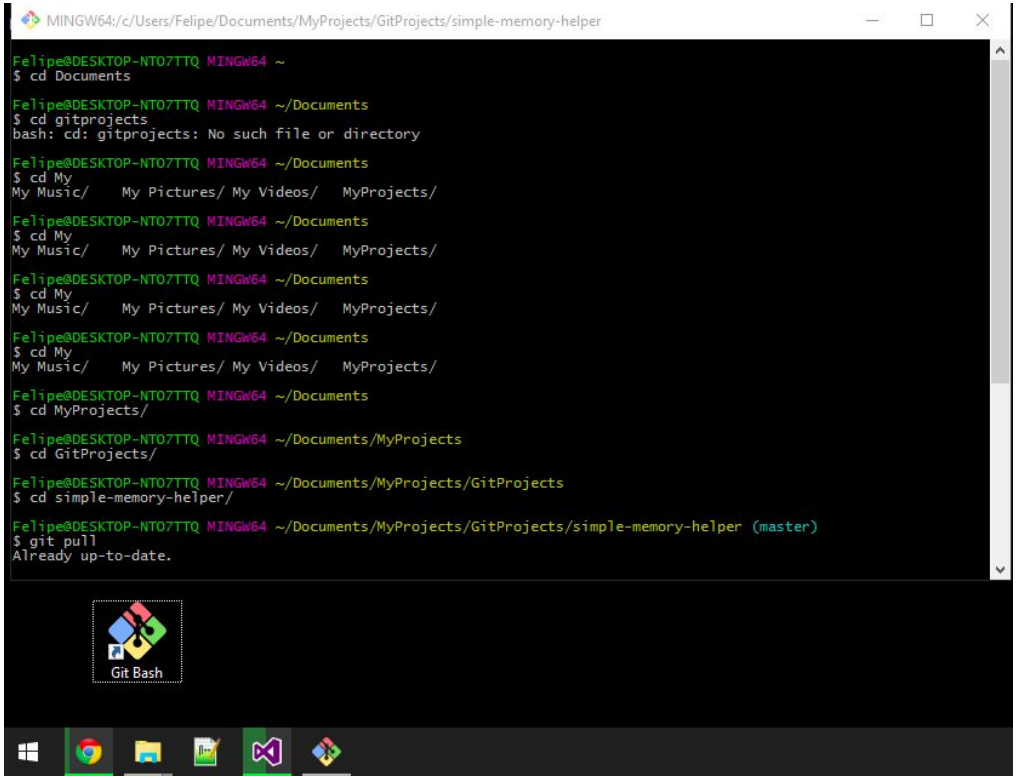
- da linea di comando
- tramite programmi con interfaccia grafica (TortoiseGit, SourceTree, **GitHub desktop** e simili)
- direttamente dall’IDE (Eclipse, VSCode, SublimeText, QtCreator e simili)

Lo useremo solo da linea di comando

- Terminale (Linux, MacOS)
- Git Bash (Windows)

Windows: Git Bash

Git Bash



Terminale (o linea di comando)

La linea di comando è molto utile per eseguire programmi e script. Alcuni comandi utili

`pwd`	Stampa la directory corrente
`ls -la`	Stampa il contenuto della directory corrente
`cd <directory>`	Cambia la directory corrente
`clear`	Cancella l'output del video

Non vi precipitate a scrivere comandi! Li vediamo prima insieme

Proviamo!

Git e terminale

Per dare comandi a Git

```
git <comando> <opzioni>
```

Si possono dare tanti comandi a Git: per esempio

- **Operazioni sul repository**: aggiungi e togli file dal repository, registra le modifiche fatte ai file,
- **Operazioni sullo stato**: dimmi qual è la situazione
- **Operazioni sulle registrazioni**: chi ha fatto una registrazione

Primi comandi

Verificare la presenza di Git

```
git --version
```

Configurare nome e email

```
git config --global user.name “Massimiliano Atzori”
git config --global user.email massimiliano.atzori@gmail.com
```

Output con colori

```
git config --global color.ui.auto
```

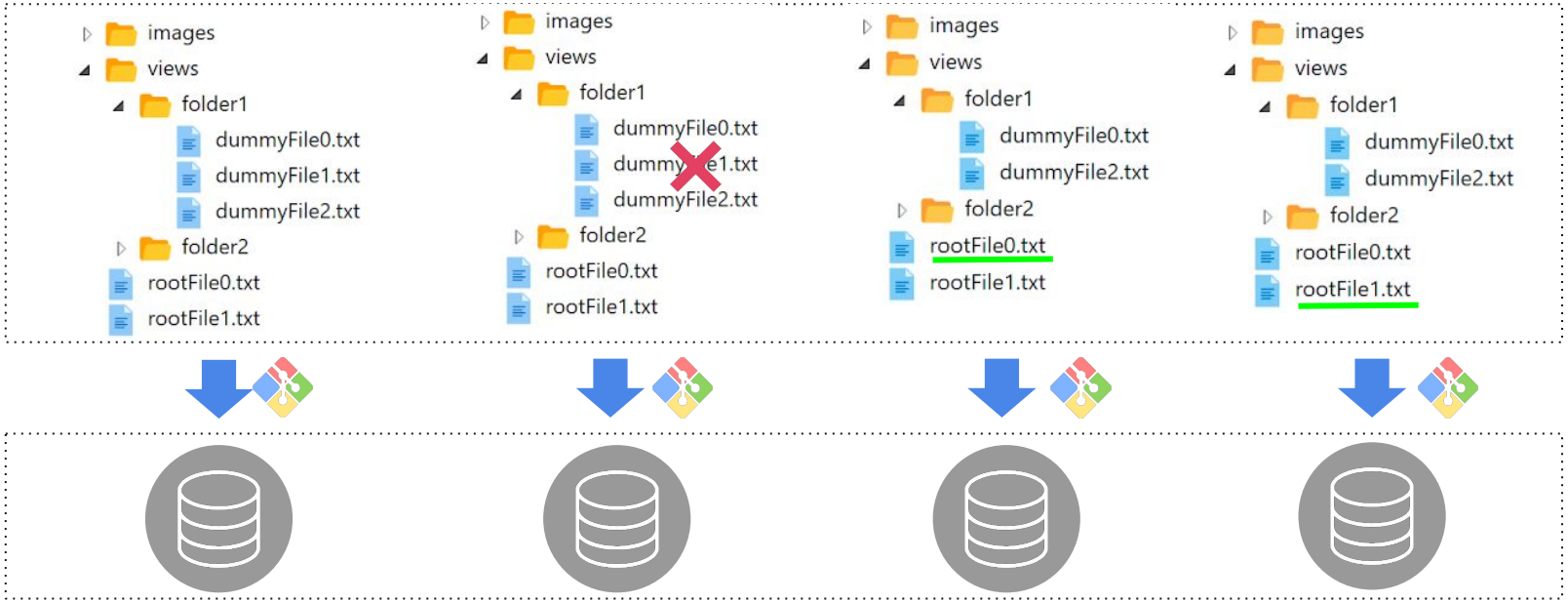
Verificare la configurazione

```
git config --list
```

Proviamo!

Terminale e repository

File system



git

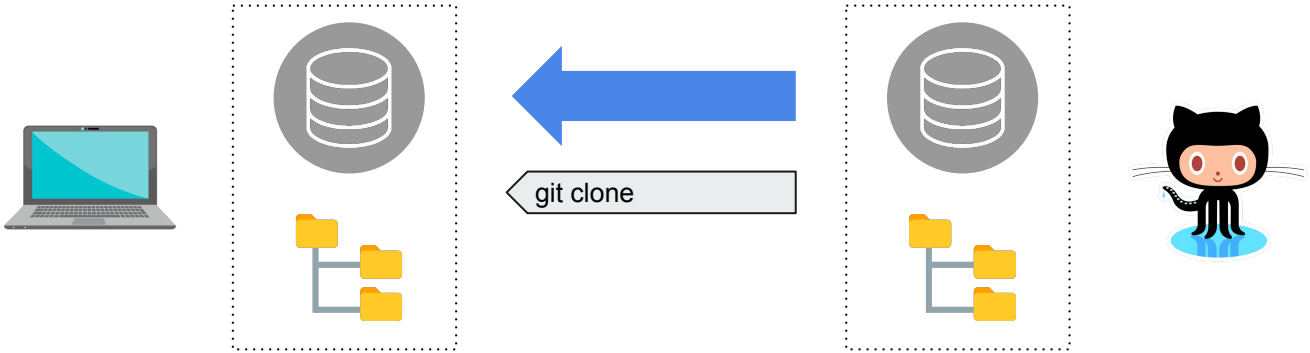
Clonare un repository da GitHub

```
cd ~
```

Comando per il terminale

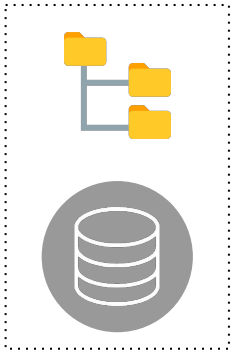
```
git clone https://github.com/amaxis/git-gobetti-volta.git
```

Comando per Git



Cos'è successo?

Con il comando clone, abbiamo scaricato da GitHub due oggetti:



L'ultima versione dei file e directory del progetto

Il repository completo, dove ci sono le informazioni su tutte le modifiche registrate

Da qui in poi, possiamo lavorare normalmente con i file e poi "comunicare" a Git quando è il momento di registrare delle modifiche in modo permanente

Le aree di lavoro di Git

Workspace

Repository

Remote repository

Le aree di lavoro di Git

Workspace

Index (staging area)

Local repository

Remote repository

Le aree di lavoro di Git

Workspace

- Sono i file su cui lavoriamo

Index (staging area)

- Modifiche **pronte** per essere registrate

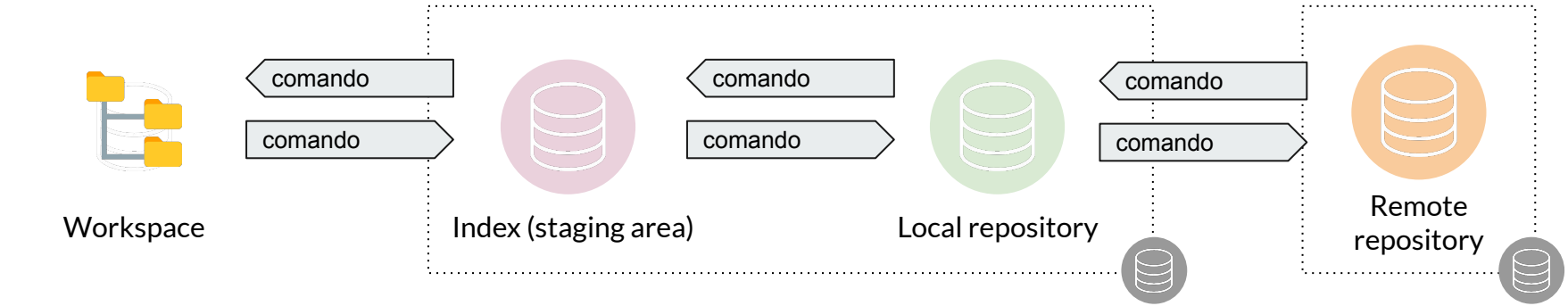
Local repository

- Modifiche **registrate**

Remote repository

- Modifiche **condivise**

Le aree di lavoro di Git



- Sono i file su cui lavoriamo
- Modifiche **pronte** per essere registrate
- Modifiche **registrate**
- Modifiche **condivise**

Controllare lo stato del nostro repo

```
git status
```

Cosa ci dice

- se abbiamo fatto modifiche ai file rispetto al contenuto del local repository
- in quali stage si trovano i file che abbiamo modificato
- se ci sono conflitti da risolvere
- se ci sono altre situazioni particolari

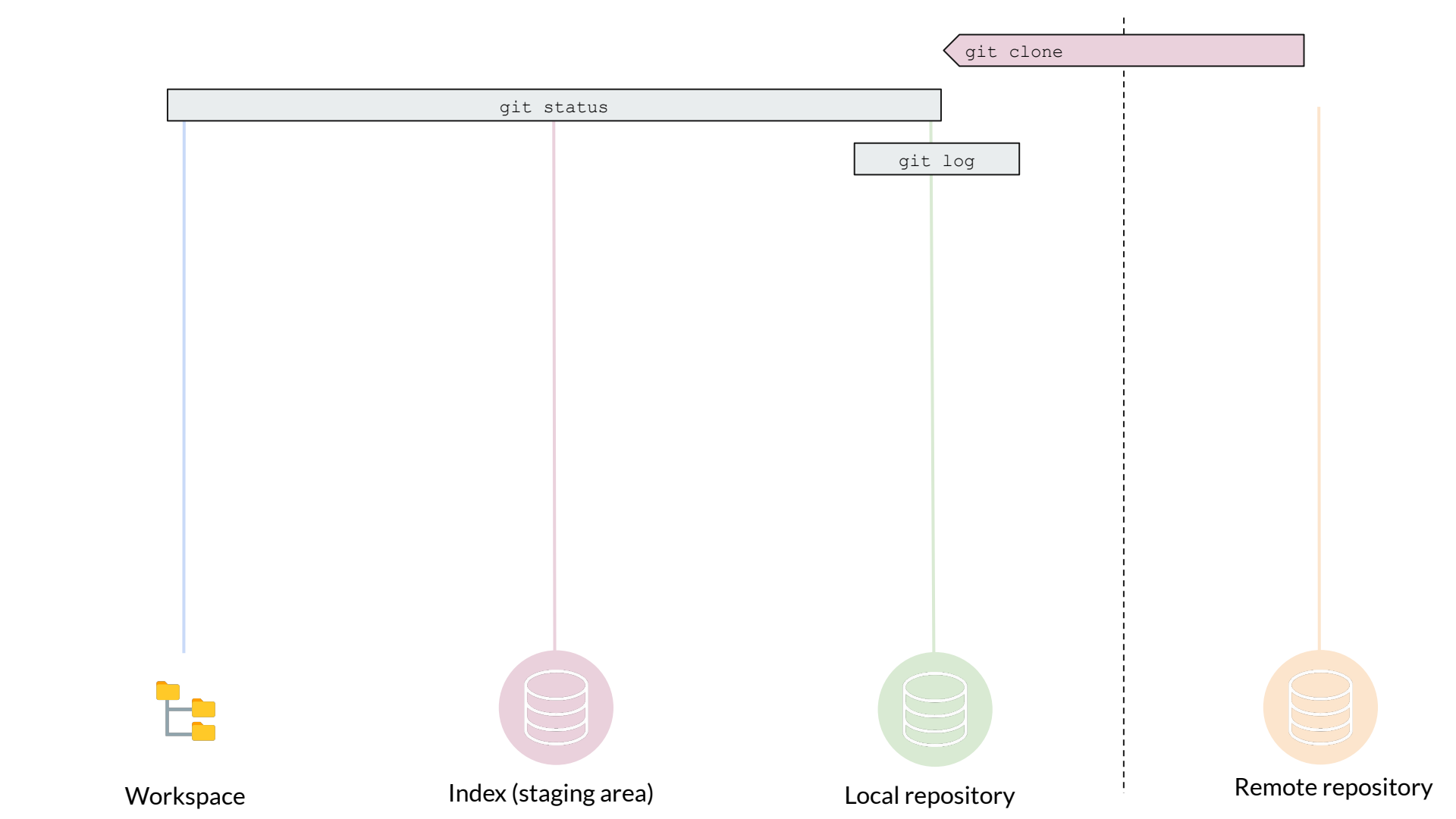
Controllare lo stato del nostro repo

```
git log
```

Cosa ci dice

- ci fa vedere la lista delle modifiche permanenti fatte al repository locale
- due opzioni utili
  - -n: mi fa vedere solo le ultime n modiche
  - --oneline: mi fa vedere una lista più compatta

```
git log -n <numero di linee> --oneline
```


git clone
git status
git log
Workspace
Index (staging area)
Local repository
Remote repository

Proviamo!

Modificare i file e portarli in remoto

Le aree di lavoro di Git

Workspace

- Sono i file su cui lavoriamo

Index (staging area)

- Modifiche **pronte** per essere registrate

Local repository

- Modifiche **registrate**

Remote repository

- Modifiche **condivise**

Aggiungiamo un file nel repository Git locale

Apriamo un file nella directory `git-gobetti-volta/java/CalcoloArea.java`

Modifichiamolo cambiando il testo dell'input dato all'utente

Salviamo le modifiche

Vediamo lo status

```
git status
```

Chiediamo a Git di aggiungerlo

```
git add java/CalcoloArea.java
```

Vediamo lo status

```
git status
```

Proviamo a togliere il file dallo staging

Chiediamo a Git di non considerare più il file

```
git rm java/CalcoloArea.java
```

Vediamo lo status

```
git status
```

Aggiungiamolo di nuovo

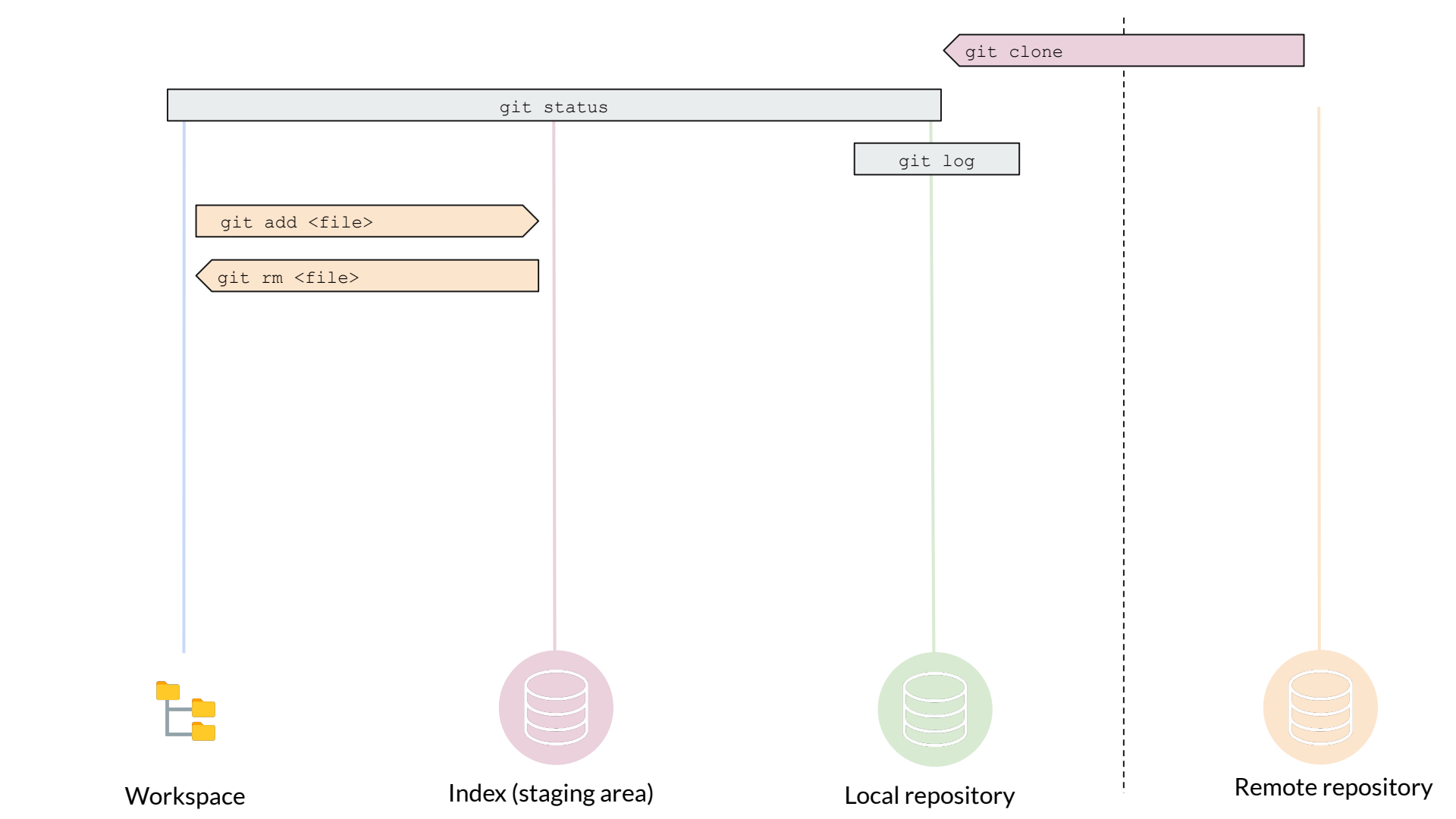
```
git add java/CalcoloArea.java
```

Vediamo lo status

```
git status
```

Vediamo la situazione del repository locale

```
git log --oneline
```


git clone
git status
git log
git add <file>
git rm <file>
Workspace
Index (staging area)
Local repository
Remote repository

Aggiorniamo il repository locale

Vediamo lo status

```
git status
```

Aggiorniamo il repository

```
git commit -m “Cambio testo input per utente”
```

Vediamo lo status

```
git status
```

Vediamo la situazione del repository

```
git log --oneline
```

Cos’è successo?

1. Abbiamo modificato un file

2. Lo abbiamo aggiunto nella staging area con il comando add

3. Abbiamo creato un commit con il comando commit, aggiornando il repository locale

I commit

I commit ci raccontano la storia delle modifiche al nostro codice

Vanno sempre avanti, non si cancellano mai: sono **snapshot permanenti**

Il messaggio di commit è obbligatorio, deve descrivere cos'è successo nelle modifiche, e si scrive al presente in 1 persona

Tutte le modifiche sono ancora in locale sulla nostra macchina, sono nel Local Repository

Portare le modifiche in remoto

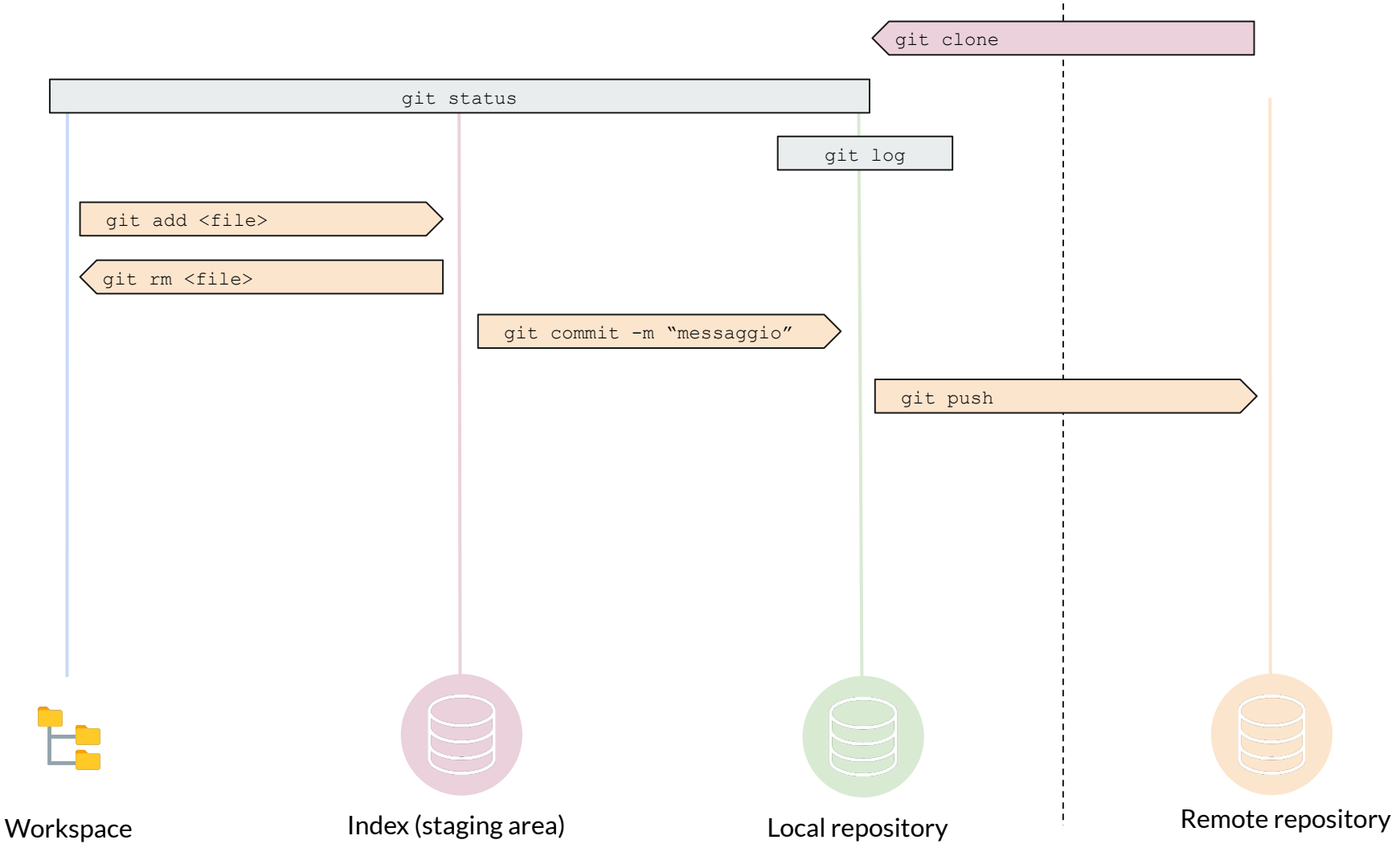
Per portare le modifiche sul repository remoto

```
git push
```

Porta i commit aggiunti nel local repository sul remote repository

Attenzione

- per l'operazione di push, ci vogliono le autorizzazioni sul repository remoto
- portare i commit sul remote può causare **conflitti**


git clone
git status
git log
git add <file>
git rm <file>
git commit -m “messaggio”
git push
Workspace
Index (staging area)
Local repository
Remote repository

Un primo semplice flusso di lavoro con Git

Clono un repository (o lo creo in locale)	`git clone / git init`
Faccio delle modifiche: aggiungo, modifico, elimino dei file	`git add / git rm`
Tengo d'occhio lo stato dei file	`git status / git log`
Quando arrivo ad un punto importante, registro le modifiche nel repository locale	`git commit`
Quando sono pronto a condividere le modifiche, le porto sul repository remoto	`git push`

Altri comandi utili

Annulla le modifiche registrate nell'Index

```
git reset
```

Riporta i file dal Local Repository al workspace

```
git reset --hard
```

Differenza

- la prima annulla le modifiche fatte nell'index, perciò **il contenuto dei file rimane invariato**
- la seconda riporta i file dal local repository a workspace, perciò **il contenuto dei file viene sovrascritto**

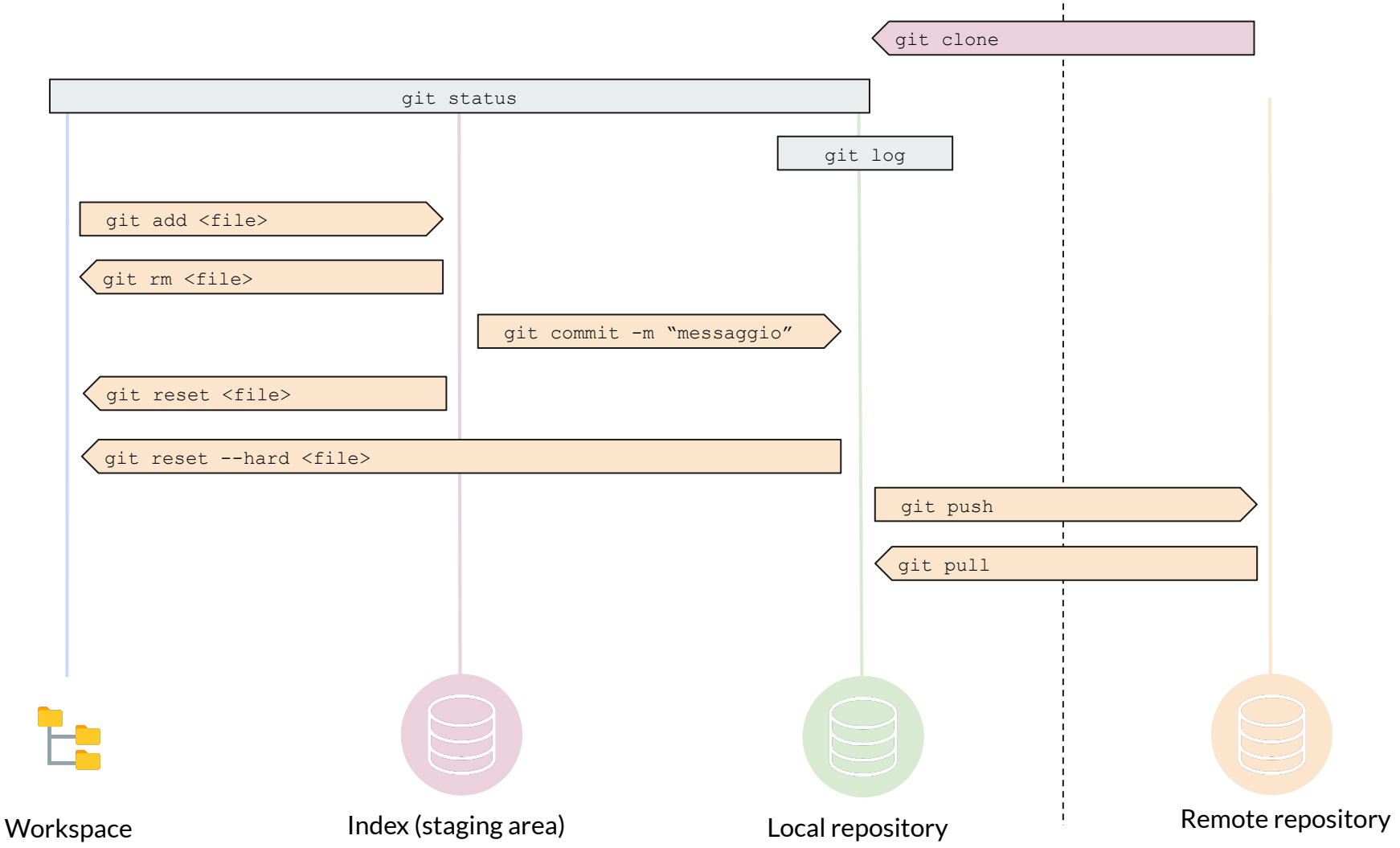
Altri comandi utili

Prende i nuovi commit dal repository remoto

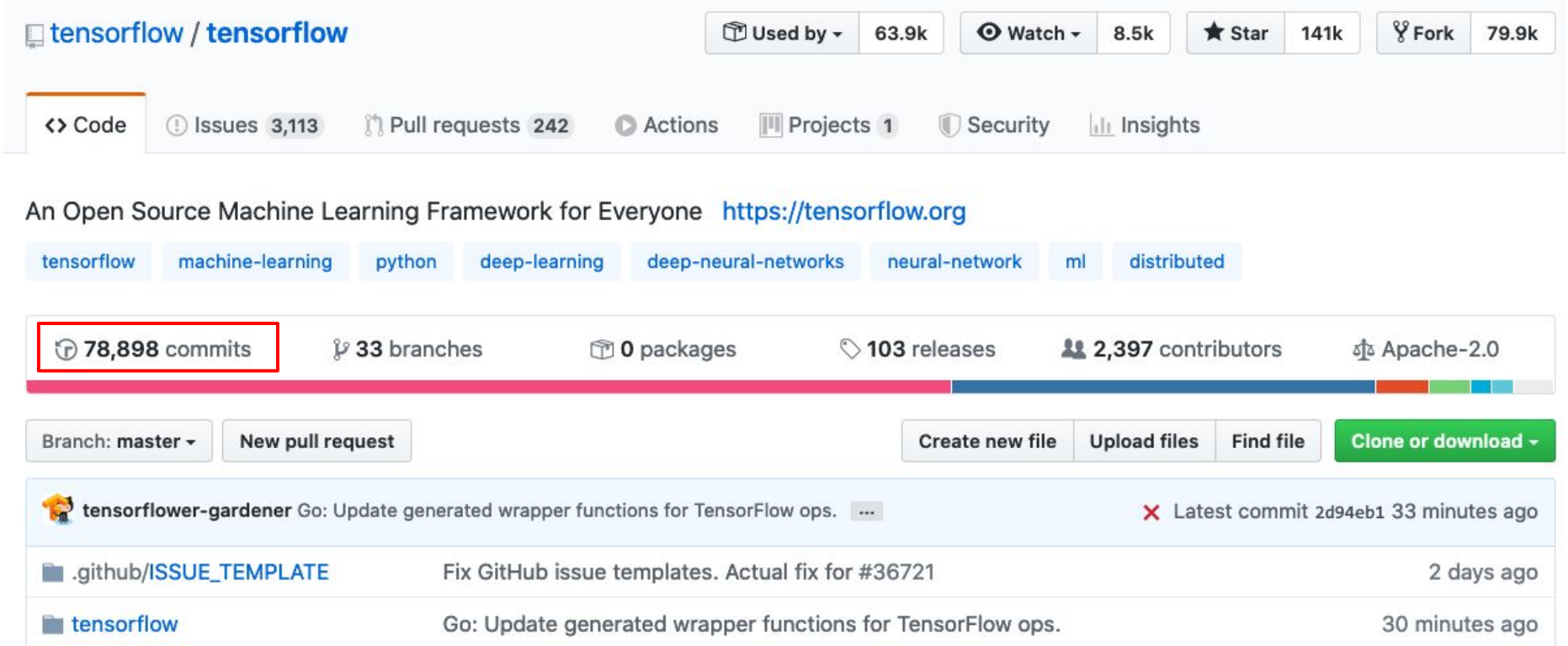
```
git pull
```

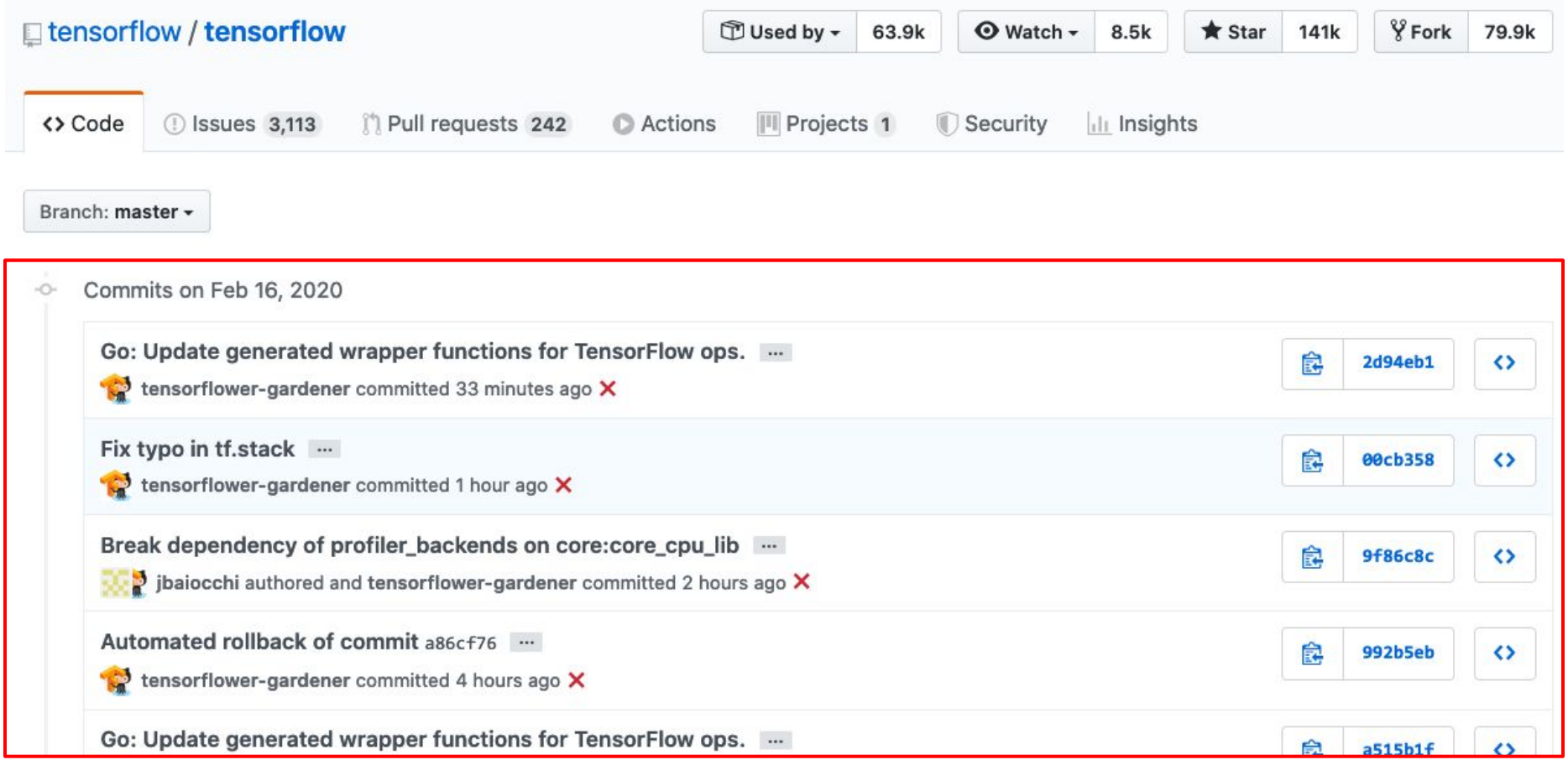
Da notare

- il comando `git clone` fa una copia esatta del repository e lo collega a quello remoto
- il comando git pull si esegue su un repository già collegato ad uno remoto e copia solo i commit che non si hanno già

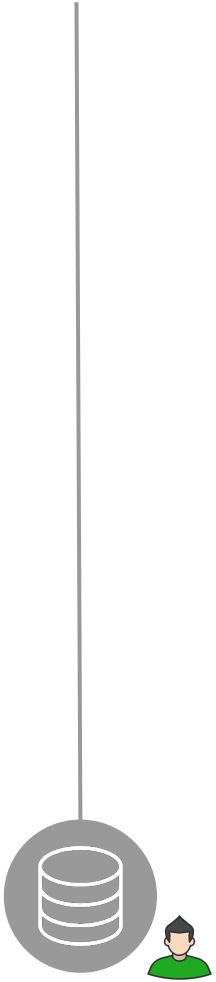

git clone
git status
git log
git add <file>
git rm <file>
git commit -m "messaggio"
git reset <file>
git reset --hard <file>
git push
git pull
Workspace
Index (staging area)
Local repository
Remote repository

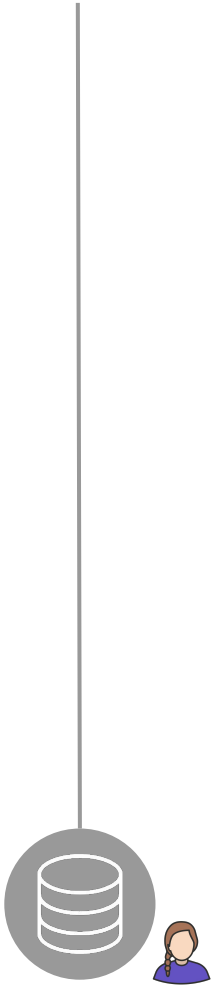
Repository e commit: un esempio


tensorflow / tensorflow
Used by 63.9k
Watch 8.5k
Star 141k
Fork 79.9k
Code
Issues 3,113
Pull requests 242
Actions
Projects 1
Security
Insights
An Open Source Machine Learning Framework for Everyone https://tensorflow.org
tensorflow
machine-learning
python
deep-learning
deep-neural-networks
neural-network
ml
distributed
78,898 commits
33 branches
0 packages
103 releases
2,397 contributors
Apache-2.0
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
tensorflower-gardener Go: Update generated wrapper functions for TensorFlow ops.
Latest commit 2d94eb1 33 minutes ago
.github/ISSUE_TEMPLATE
Fix GitHub issue templates. Actual fix for #36721
2 days ago
tensorflow
Go: Update generated wrapper functions for TensorFlow ops.
30 minutes ago

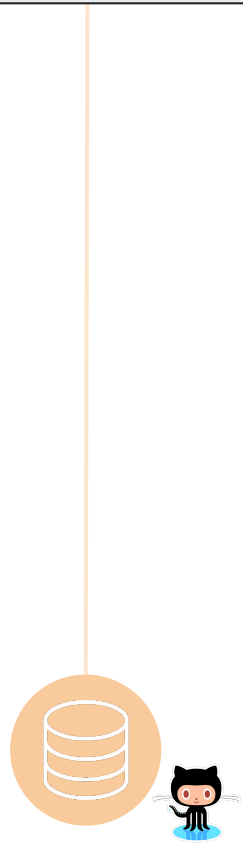

tensorflow / tensorflow
Used by 63.9k
Watch 8.5k
Star 141k
Fork 79.9k
Code
Issues 3,113
Pull requests 242
Actions
Projects 1
Security
Insights
Branch: master
Commits on Feb 16, 2020
Go: Update generated wrapper functions for TensorFlow ops.
tensorflower-gardener committed 33 minutes ago
2d94eb1
Fix typo in tf.stack
tensorflower-gardener committed 1 hour ago
00cb358
Break dependency of profiler_backends on core:core_cpu_lib
jbaiocchi authored and tensorflower-gardener committed 2 hours ago
9f86c8c
Automated rollback of commit a86cf76
tensorflower-gardener committed 4 hours ago
992b5eb
Go: Update generated wrapper functions for TensorFlow ops.
a515b1f

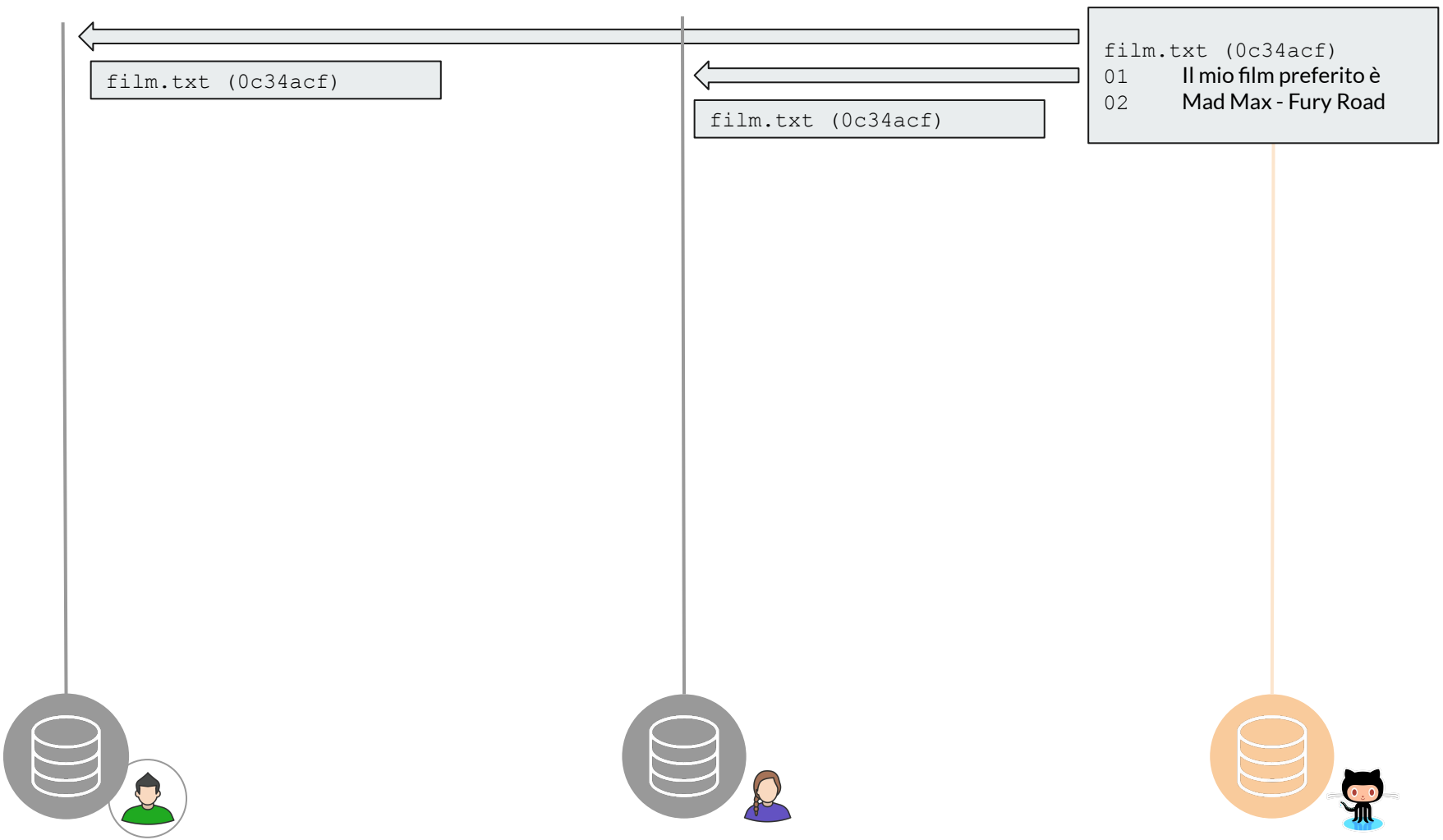
Conflitti e merge

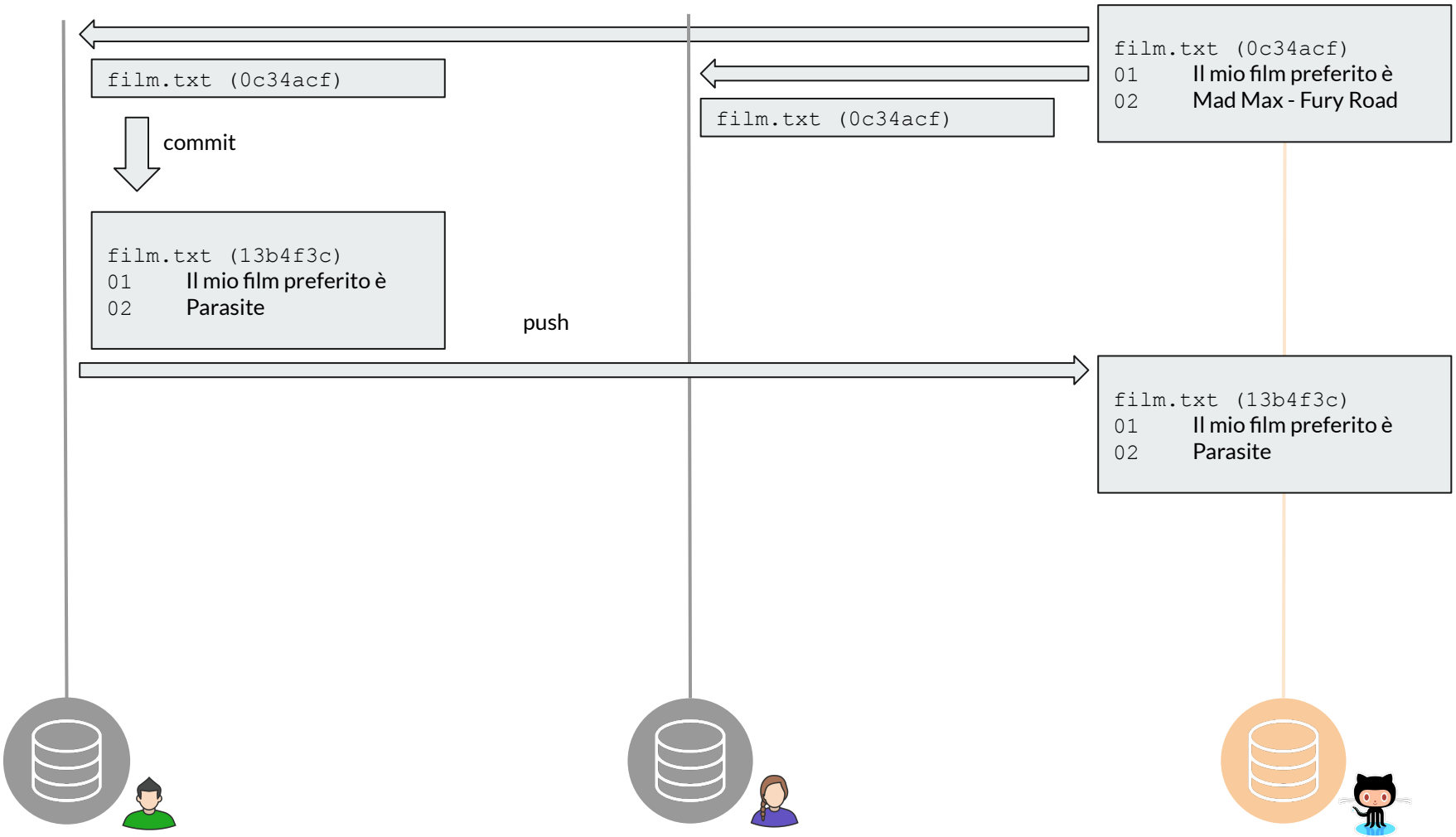


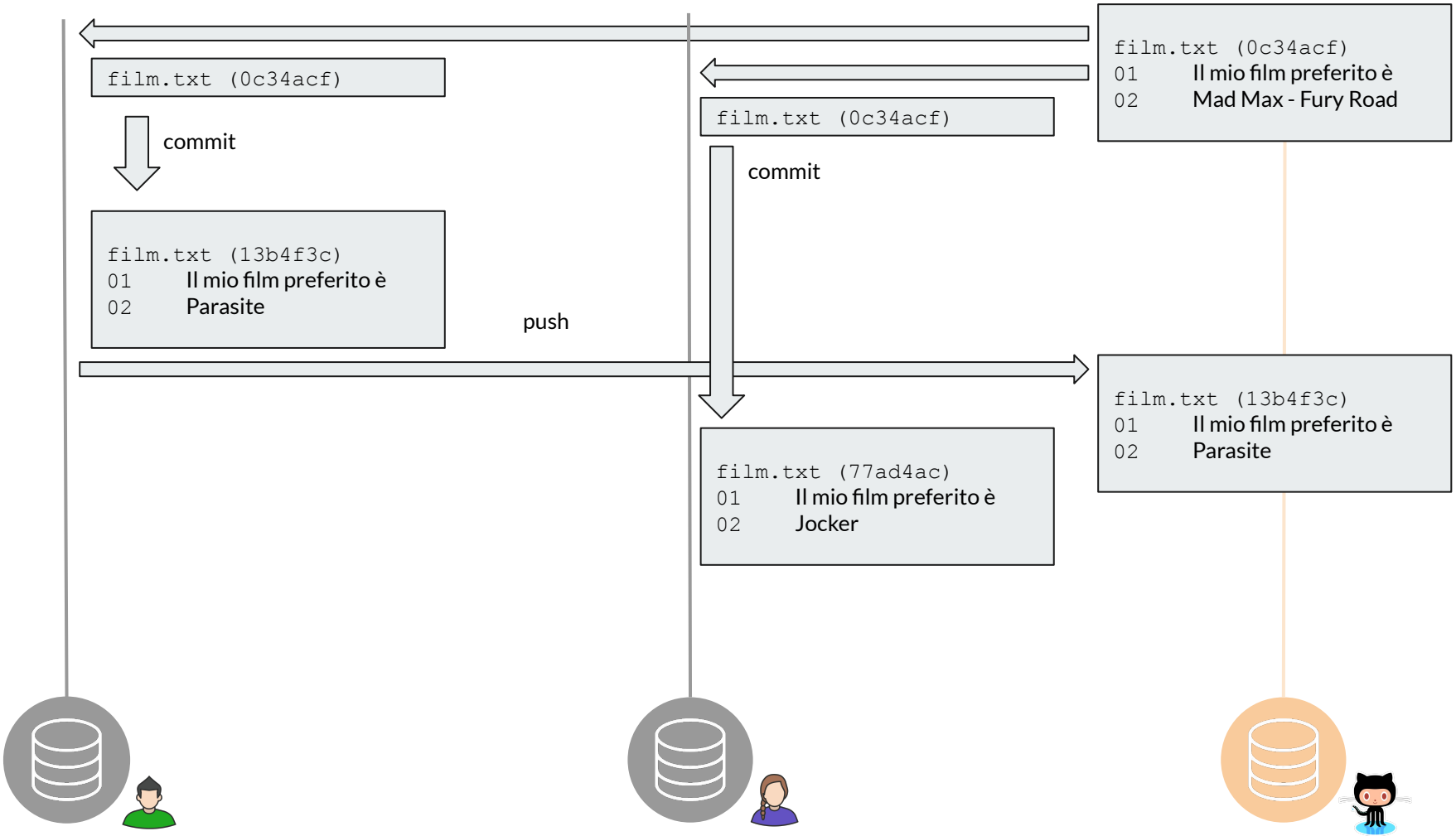


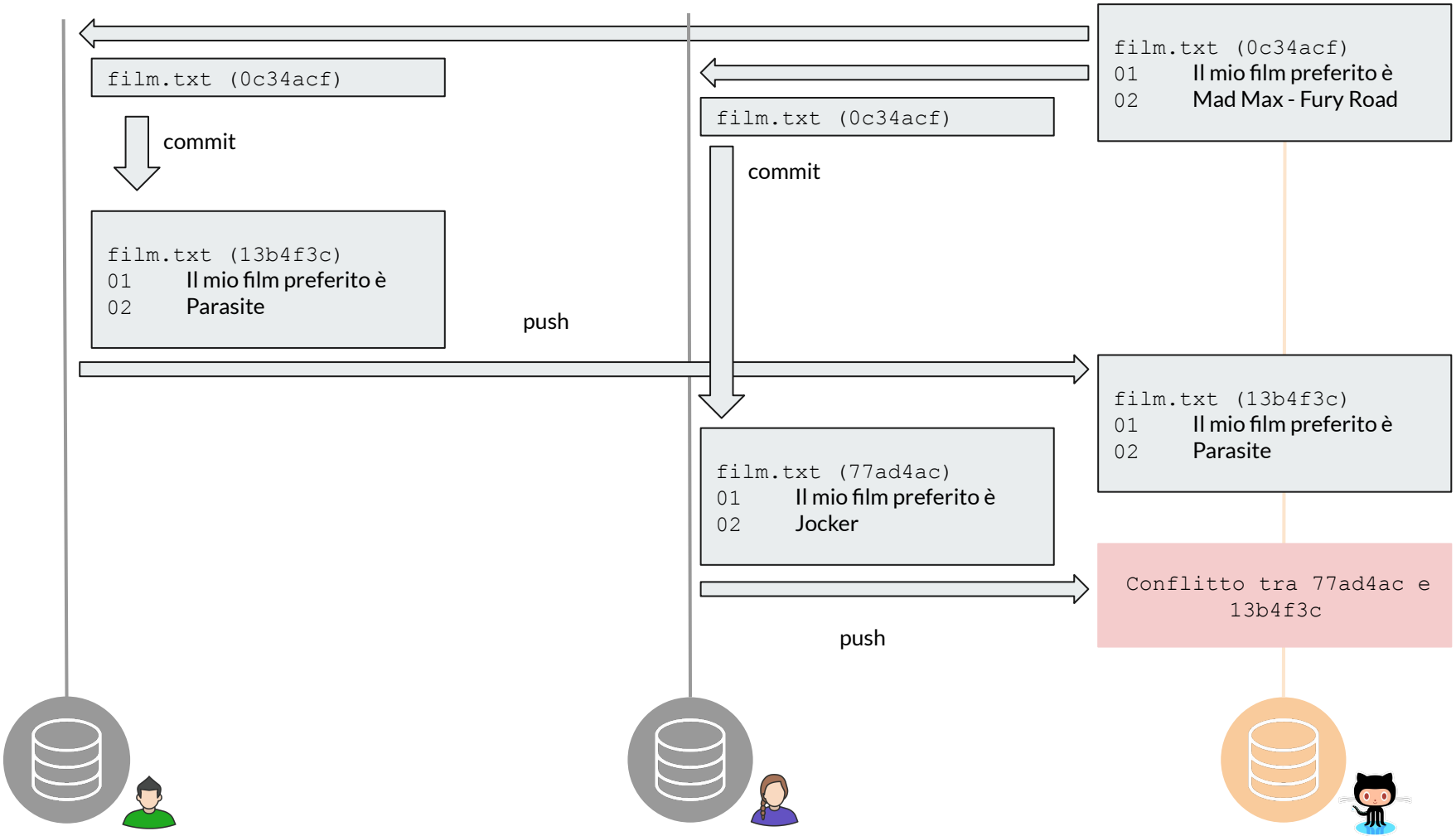
```
film.txt (0c34acf)
01      Il mio film preferito è
02      Mad Max - Fury Road
```




film.txt (0c34acf)
film.txt (0c34acf)
film.txt (0c34acf)
01 Il mio film preferito è
02 Mad Max - Fury Road


film.txt (0c34acf)
commit
film.txt (13b4f3c)
01 Il mio film preferito è
02 Parasite
push
film.txt (0c34acf)
film.txt (0c34acf)
01 Il mio film preferito è
02 Mad Max - Fury Road
film.txt (13b4f3c)
01 Il mio film preferito è
02 Parasite


film.txt (0c34acf)
commit
film.txt (13b4f3c)
01 Il mio film preferito è
02 Parasite
push
film.txt (0c34acf)
film.txt (0c34acf)
01 Il mio film preferito è
02 Mad Max - Fury Road
commit
film.txt (77ad4ac)
01 Il mio film preferito è
02 Jocker
film.txt (13b4f3c)
01 Il mio film preferito è
02 Parasite


film.txt (0c34acf)
commit
film.txt (13b4f3c)
01 Il mio film preferito è
02 Parasite
push
film.txt (0c34acf)
film.txt (0c34acf)
01 Il mio film preferito è
02 Mad Max - Fury Road
commit
film.txt (77ad4ac)
01 Il mio film preferito è
02 Jocker
push
film.txt (13b4f3c)
01 Il mio film preferito è
02 Parasite
Conflitto tra 77ad4ac e 13b4f3c

Proviamo!

Cos’è successo?

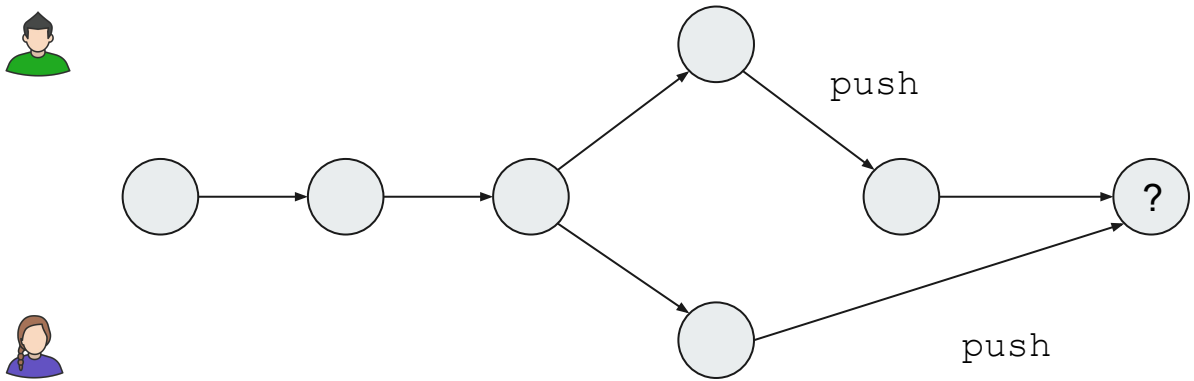
Git ha identificato un conflitto: ci sono due modifiche, fatte a partire dallo stesso commit, che sono in contraddizione

Non tutte le modifiche sono classificate automaticamente come conflitti: quelle che non di pestano i piedi, vanno bene (per esempio, aggiunte)

Ogni volta che si fa un’operazione di push o pull, **si possono verificare dei conflitti**

Cos’è successo?

Albero dei commit (snapshot)



Cosa fare in caso di conflitto

Quando c'è un conflitto, Git scrive nei file quali parti sono in contraddizione. Per risolvere il conflitto, basta scrivere qual è la versione corretta

```
film.txt (77ad4ac)
01 Il mio film preferito è
02 <<<<<<< HEAD
03 Parasite
04 =======
05 Jocker
06 >>>>>>> 77ad4ac
```

Cosa fare in caso di conflitto

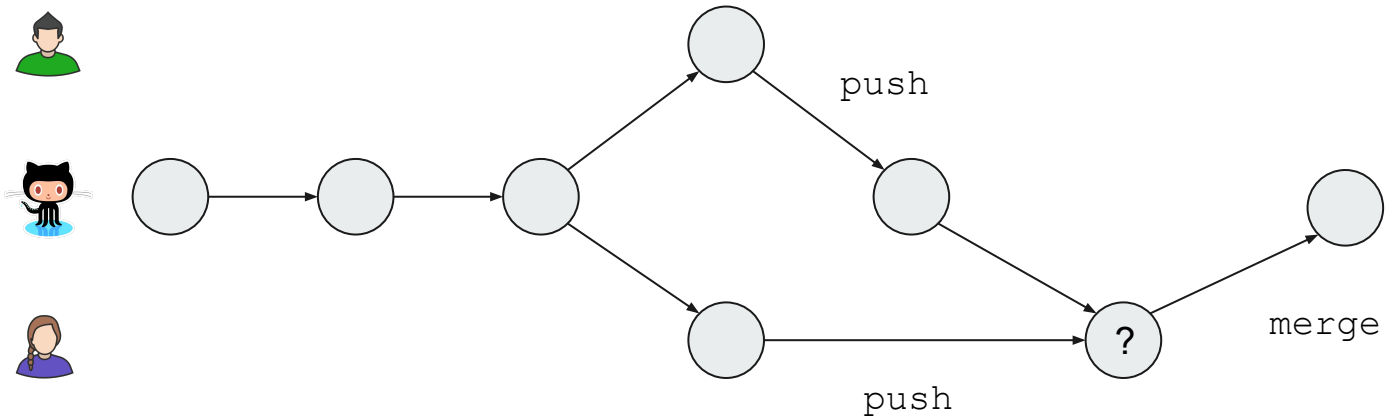
Quando c'è un conflitto, Git scrive nei file quali parti sono in contraddizione. Per risolvere il conflitto, basta scrivere qual è la versione corretta

```
film.txt (77ad4ac)
01 Il mio film preferito è
02 <<<<<<< HEAD
03 Parasite
04 =======
05 Jocker
06 >>>>>>> 77ad4ac
```

```
film.txt (77ad4ac)
01 Il mio film preferito è
02 Jocker
```

Cos'è successo?

Albero dei commit (snapshot)



Esempio

1. Modifico un file	`README.md`
2. Lo aggiungo in staging	`git add README.md`
3. Lo aggiungo al repository (faccio un commit)	`git commit -m “Modifica nome istituto”`
4. Provo a fare un push delle modifiche	`git push`
5. Conflitto: apro il file e faccio le modifiche	`README.md`
6. Rifaccio un commit	`git commit -m “Modifica nome istituto”`
7. È andata! 🎉	

Proviamo!

Riepilogo

1. Installiamo Git e creiamo un account su GitHub
2. Breve introduzione a Git
3. Primi passi con Git: clone, add, remove, commit
4. Portare le modifiche in remoto
5. Conflitti e merge
6. Riepilogo e consigli utili

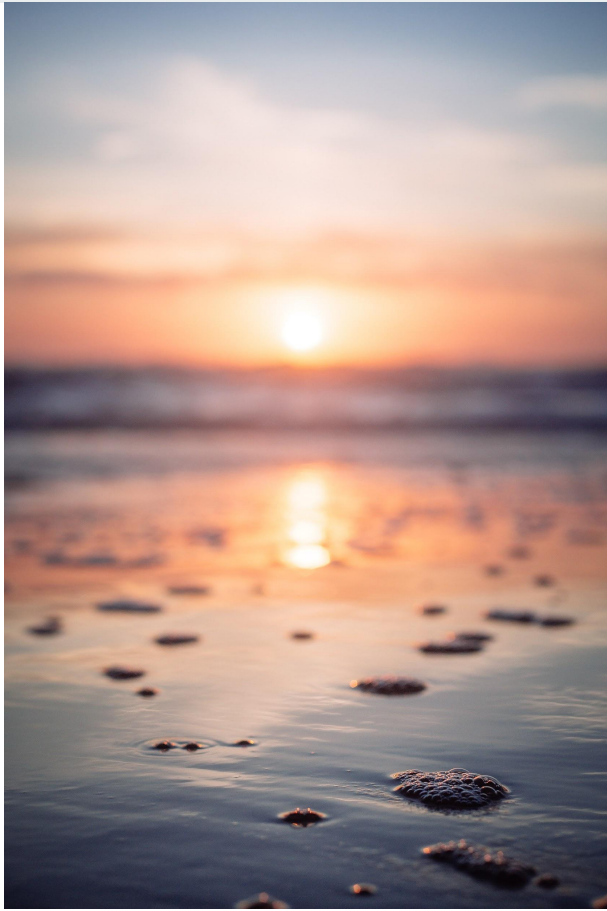


Photo by Cole Keister on Unsplash

Cosa manca

Branch, tag, remotes

Merge, rebase

Git tools (comandi avanzati)

Git workflow

Photo by Joshua Earle on Unsplash

Link e risorse

Libro	https://git-scm.com/book/en/v2
Slides e video	https://channel9.msdn.com/Series/Introduzione-a-Git
	https://www.slideshare.net/valix85/introduzione-a-git-ita-2017
	Google :)
Questa presentazione	https://github.com/amaxis/git-gobetti-volta
	`git clone https://github.com/amaxis/git-gobetti-volta`

Grazie

Massimiliano Atzori

massimiliano.atzori@gmail.com

@amaxis