

INFORMATICA

Intelligenza Artificiale & Data Analytics

Precorsi a.a. 2023/24

Docente: Gloria Pietropolli

5. VERSION CONTROL SYSTEMS

Richiami di markdown e la necessità del
versioning

RECAP: MARKDOWN

Markdown è introduce una formattazione basilare all'interno dei file di testo

- Possiamo rendere il testo grassetto, italico e sottolineato
- Possiamo indicare titoli e sottotitoli
- Possiamo indicare formule matematiche

Funzione

Una funzione è una relazione tra due insiemi (<https://it.wikipedia.org/wiki/Insieme>), chiamati dominio e codominio della funzione, che associa a ogni elemento del dominio uno e un solo elemento del codominio.

Sia X il dominio e Y il codominio, la relazione è indicata così: $f: X \rightarrow Y$.

Funzione

Una funzione è una relazione tra due insiemi, chiamati dominio e codominio della funzione, che associa a ogni elemento del dominio uno e un solo elemento del codominio.

Sia X il dominio e Y il codominio, la relazione è indicata così: $f: X \rightarrow Y$.

MARKDOWN PER UN SITO

Markdown viene usato per creare **siti internet**

Siamo un gruppo di studenti che si uniscono a formare un'associazione, di cui dobbiamo gestire un sito

Dobbiamo effettuare una manutenzione costante del sito. Es:

- Aggiungere nuovi eventi
- Aggiornare l'organigramma del direttivo
- Aggiungere nuove pagine per offrire aiuto alle nuove matricole ...

IL PERCHÉ DEL VERSIONING

Roberta viene incaricata di creare una nuova pagina per l'*helpdesk*

Prepara una prima versione e la passa tramite chiavetta USB ad Andrea, il presidente, per la revisione finale

Tuttavia Andrea non è soddisfatto e inizia ad operare modifiche alla versione di Roberta, salvando il file e sovrascrivendolo sulla chiavetta

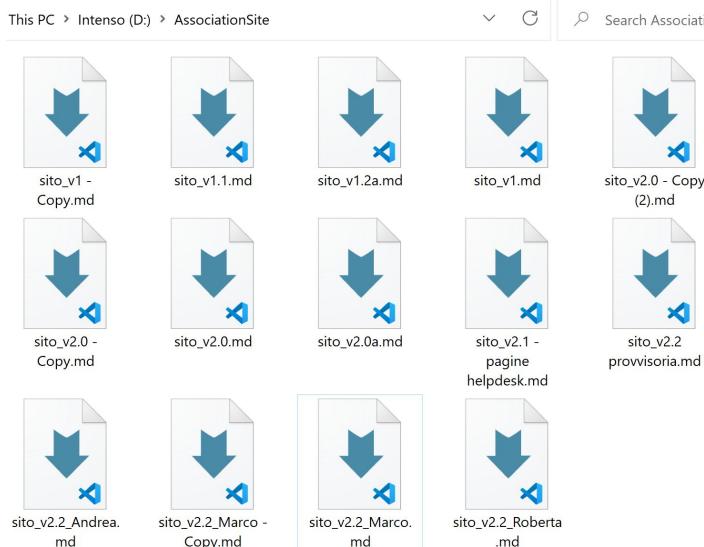
Si accorge però che preferiva la versione precedente, che però non è più disponibile

Il file precedente è perso e non c'è modo di recuperarlo

COME TENERE TRACCIA DELLE VERSIONI DI UN FILE

Avete in mente un modo per poter tenere traccia delle varie versioni di un file?

Tenere varie copie di un file, una per ogni versione



COME TENERE TRACCIA DELLE VERSIONI DI UN FILE - THE RIGHT WAY

La maniera corretta di fare versioning è quella di affidarsi ad uno strumento che tenga traccia della storia di un insieme di file e cartelle

Un tale strumento viene chiamato SISTEMA DI CONTROLLO VERSIONE o, in inglese, VERSION CONTROL SYSTEM (VCS)

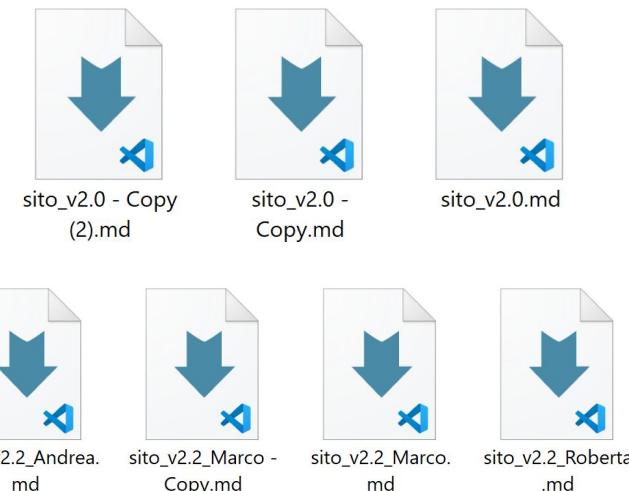
CHE COS'È UN VCS

Un VCS è uno strumento che permette di...

Mantenere uno storico di un insieme di file e cartelle, che chiameremo **repository** o **repo**, tramite opportuni *checkpoint* nel tempo

Tenere più versioni *alternative* degli stessi file

Collaborare con più persone facilitando l'unificazione dei diversi lavori



5. VERSION CONTROL SYSTEMS

Git

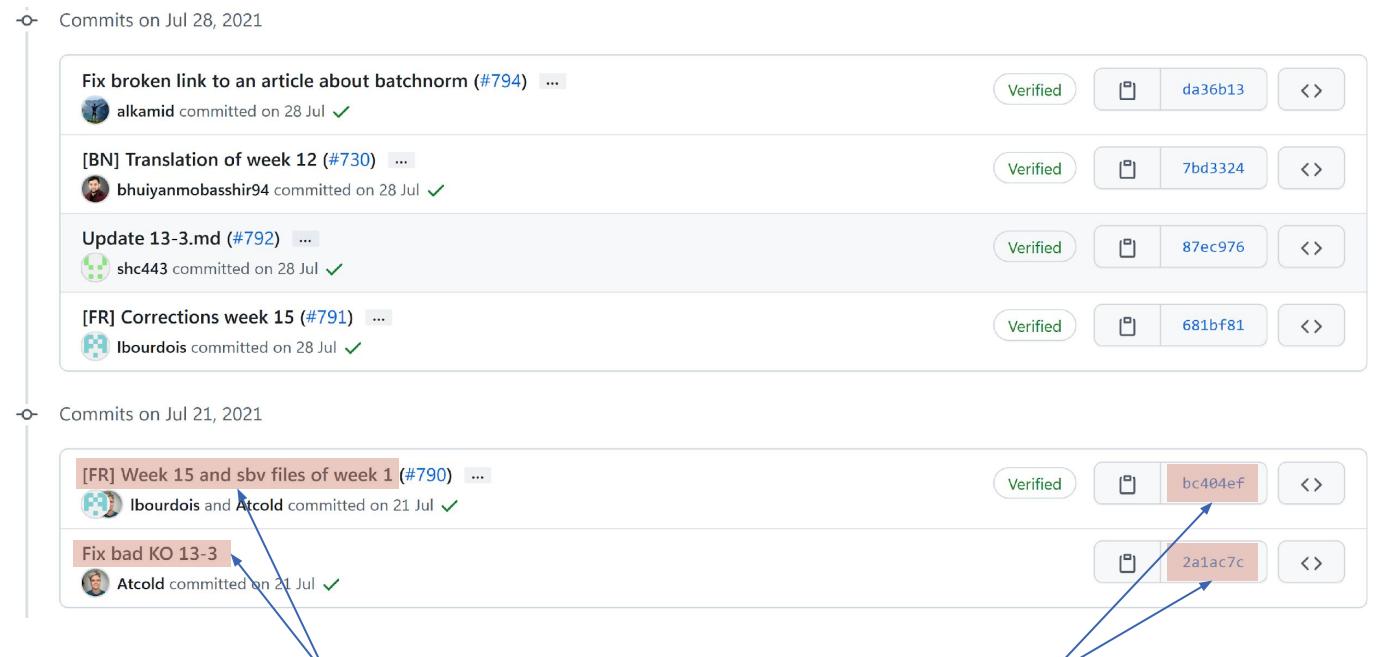


- ❖ VCS ampiamente utilizzato nel mondo dello sviluppo software.
- ❖ consente di tenere traccia delle modifiche apportate al codice sorgente in modo efficace e collaborativo.
- ❖ strumento fondamentale per:
 - gestire progetti software
 - monitorare le modifiche
 - facilitare la collaborazione tra sviluppatori.

GESTIONE DELLA STORIA DELLA REPO

Git si occupa di gestire la storia della repo tramite dei **checkpoint**, che in gergo Git si chiamano **commit**

Ogni commit viene creato dall'utente che sta effettuando modifiche sulla repo



Ogni commit è caratterizzato da un messaggio descrivente le modifiche apportate...

...ed è definito univocamente da un codice numerico esadecimale

RECUPERO DI UNA VERSIONE PRECEDENTE

Git è in grado di ricostruire la situazione della repo per ogni commit

Nell'esempio di Andrea e Roberta, se l'associazione avesse tenuto traccia dei file del sito tramite Git, Andrea sarebbe riuscito a recuperare la versione di Roberta semplicemente andando a ripescare l'ultimo commit fatto da Roberta.

PRACTICUM – GIT CON VISUAL STUDIO CODE



Andremo a visualizzare l'utilizzo di Git tramite il text editor Visual Studio Code, che presenta una GUI per utilizzare Git senza linea di comando.

Più tardi andremo a vedere come possiamo utilizzare anche Git in CLI.

Andremo a gestire con Git una repo con un insieme di documenti markdown per un sito di matematica.

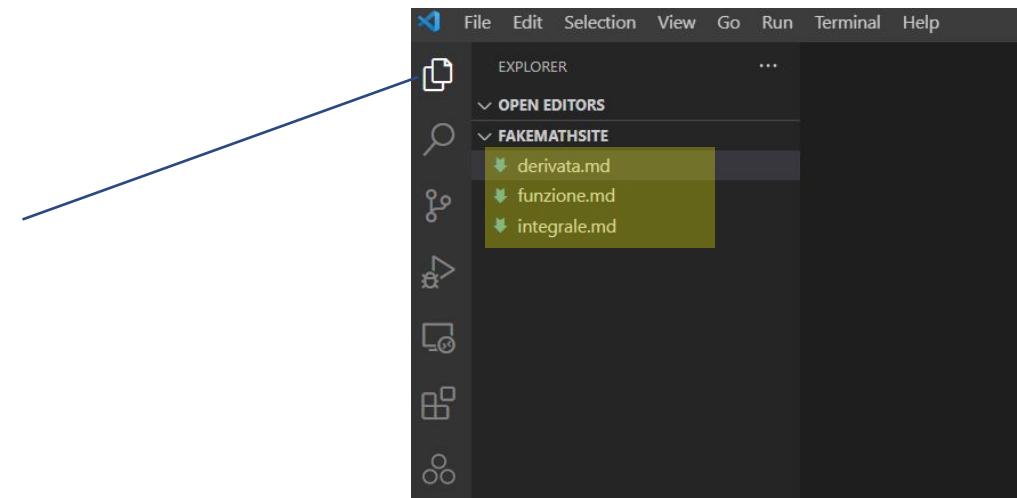
Potete scaricare l'archivio da questo link

<https://1drv.ms/u/s!AtnkZEKwlUIUhuMTz-kXgjJSpsfJtA?e=Jnsdde>

APERTURA CARTELLA CON VS CODE

1. aprire VSCode
2. selezionare *Open Folder*
3. selezionare la cartella scaricata (e decompressa)

Vediamo che i file della cartella sono tutti rappresentati nella barra di navigazione «Explorer» situata a sinistra

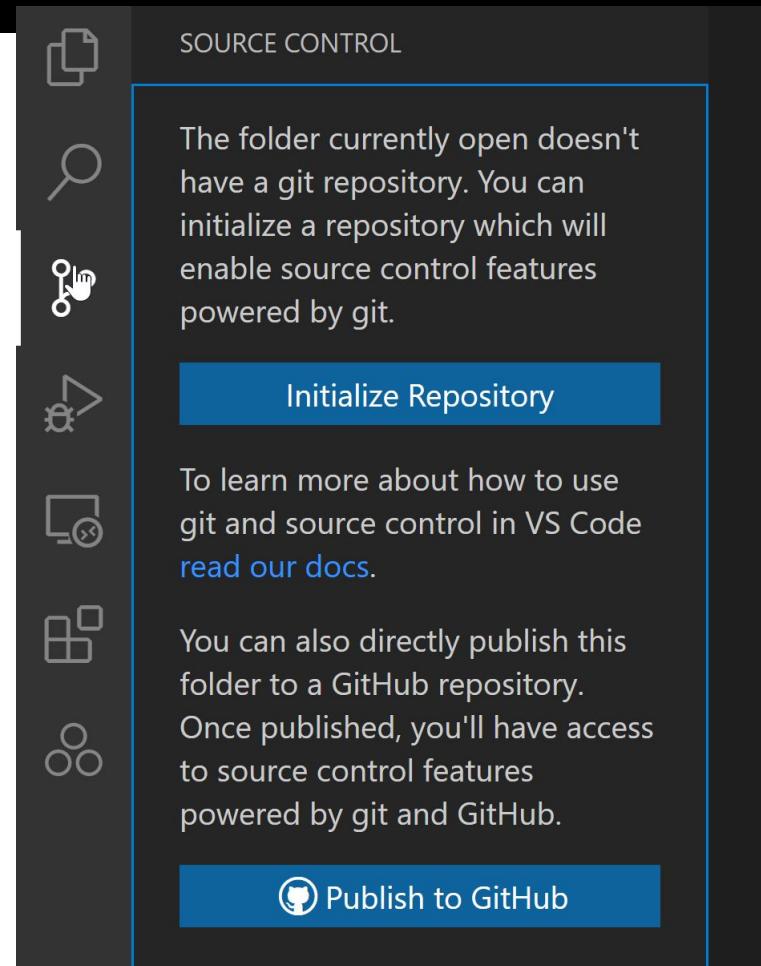


INIZIALIZZAZIONE DI UNA REPO GIT

inizializziamo una repo git per questa cartella

1. apriamo *Source Control*
2. selezioniamo *Initialize Repository*

Nota: se Git non è installato, non riusciremo ad accedere a questo menu



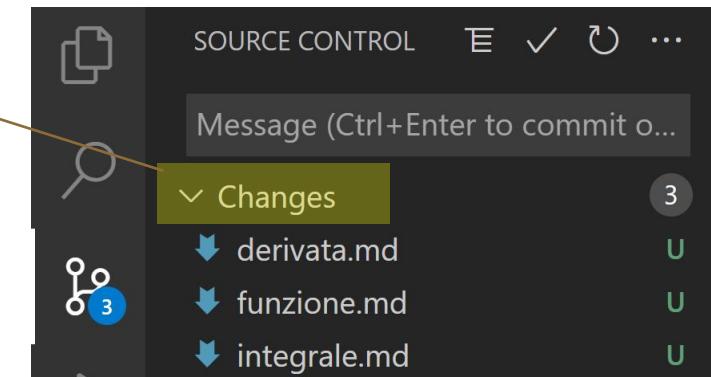
MENU SOURCE CONTROL

il menu cambierà aspetto - simile all'Explorer

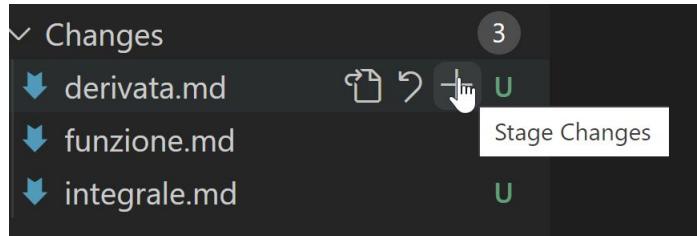
Notiamo un menu «Changes», contenente i 3 file della repo

Domanda: perché questi file sono presentati come «cambiamenti»?

Perché Git ha iniziato a tracciare tutti i file nella repo. Siccome la repo è nuova, tutti i file sono dei «cambiamenti» in quanto la repo era vuota.



AGGIUNTA DI FILE ALLA REPO (I)



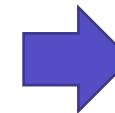
Passando col mouse sopra ai file nel menu, noteremo alcune opzioni, tra cui uno «stage changes» identificato da un'icona con un «+»

Domanda: perché VSCode ci chiede se vogliamo aggiungere questi cambiamenti?

Perché la repo in realtà è scollegata dalla configurazione attuale della cartella nel filesystem.

Infatti, dobbiamo aggiungere «a mano» i file modificati all'interno della repo premendo il bottone con il «+»

Potremmo persino fare un *commit* includendo solo una parte dei file modificati, e tenere gli altri per un *commit* successivo.



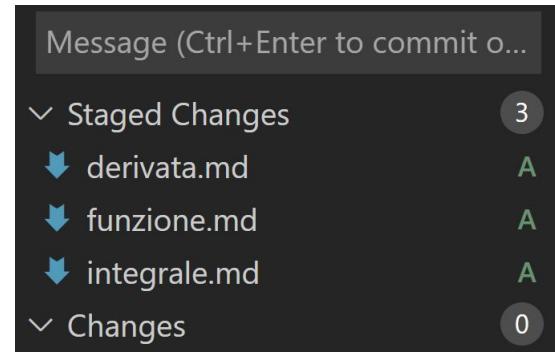
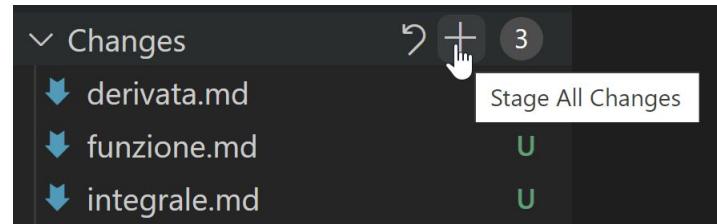
Prima di fare un *commit*, dobbiamo aggiungere (*add*) alla repo i file che ci interessa *committare*

AGGIUNTA DI FILE ALLA REPO (II)

Aggiungiamo tutti i file alla repo
→ Possiamo farlo in una volta premendo «+» accanto a changes

Il menù è cambiato:

- Non ci sono più **changes**
- I file aggiunti sono sotto **staged changes** - sono pronti per essere **committati**

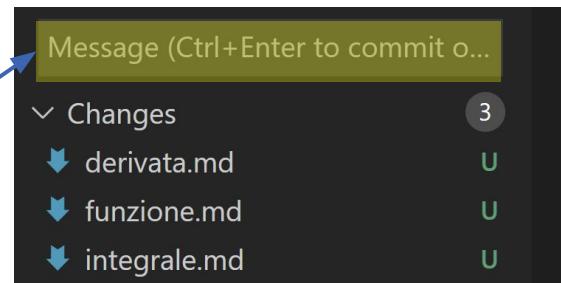


ESECUZIONE DEL COMMIT

Reminder:



Il codice alfanumerico viene aggiunto automaticamente da Git, il messaggio dobbiamo aggiungerlo noi



Scriviamo «primo commit» come messaggio; digitiamo Alt+Enter per confermare il commit



Congratulazioni, abbiamo creato il nostro primo commit!

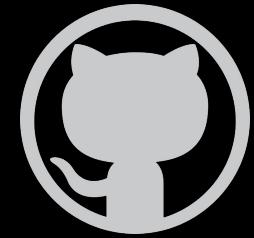
LAVORO COLLABORATIVO – REMOTE

Abbiamo visto solo come creare un commit sul nostro computer personale.

Ma vogliamo vedere anche i lavori fatti da Roberta e da Andrea, senza scambiarci chiavette o mail.

Git supporta anche i remote, ovvero degli spazi cloud che possono parlare con Git dove è possibile caricare una repo affinché più persone possano lavorarci contemporaneamente

I PRINCIPALI SERVIZI REMOTE



Due piattaforme molto famose per l'*hosting* di repo Git sono GitHub e GitLab.

L'idea di base riguardo al loro funzionamento è la seguente:

- 1 • Creo una repo GitHub/GitLab remota vuota
- 2 • Sincronizzo la repo *remota* con la mia repo *locale*
- 3 • Dopo uno o più commit, carico (*push*) la mia repo locale nello spazio remoto
- 4 • Andrea vuole fare delle modifiche: scarica (*pull*) la repo remota aggiornando la situazione della sua repo locale
- 5 • Andrea fa i suoi commit e fa il *push* della sua repo locale aggiornata
- 6 • Scarico la versione aggiornata (*pull*) della repo di Andrea sul mio computer, sovrascrivendo la mia versione locale non più aggiornata

CREARE LA REPO SU GITHUB.COM



INTERFACCIA CREAZIONE REPO

Descrizione della repo

Pubblico: visibile da tutti (anche su Google!)
Privato: visibile solo da collaboratori (ma nr. collaboratori limitato con account gratuito)

Funzionalità aggiuntive: vedrete nei corsi futuri a cosa servono e come utilizzarle.
Per adesso lasciamo tutto vuoto

Create a new repository

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

Repository template

Start your repository with a template repository's contents.

No template ▾

Owner *

 marcozullich ▾

Repository name *

FakeMathSite 

Great repository names are short and memorable. Need inspiration? How about [animated-computing-machine](#)?

Description (optional)

progetto fittizio per precorsi informatica DMG, a.a. 2021/'22

 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:

Skip this step if you're importing an existing repository.

Add a README file

This is where you can write a long description for your project. [Learn more](#).

Add .gitignore

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

Choose a license

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

[Create repository](#)

Nome identificativo
della repo (senza spazi).
Non deve essere per
forza uguale al nome
della cartella sul nostro

QUICK SETUP

The screenshot shows a GitHub repository named "narcozullich / FakeMathSite" (Private). The "Code" tab is selected. A blue banner at the top right reads "Ora dobbiamo collegare il nuovo remote alla nostra repo locale". Below the banner, the GitHub interface shows the "Quick setup" section. It includes options for "Set up in Desktop" or "HTTPS" (which is highlighted in yellow), and the URL "https://github.com/marcozullich/FakeMathSite.git". A green box highlights the URL. A blue callout box on the left says "Selezioniamo HTTPS". Another blue callout box on the right says "Copiamo questo indirizzo (è l'indirizzo della nostra repo online)". Below the quick setup section, there's a command-line guide with a large red X drawn over it, and a note "...or create a new repository on the command line".

Ora dobbiamo collegare il nuovo remote alla nostra repo locale

narcozullich / FakeMathSite Private

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Selezioniamo HTTPS

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

Set up in Desktop or **HTTPS** SSH https://github.com/marcozullich/FakeMathSite.git

Copiamo questo indirizzo (è l'indirizzo della nostra repo online)

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

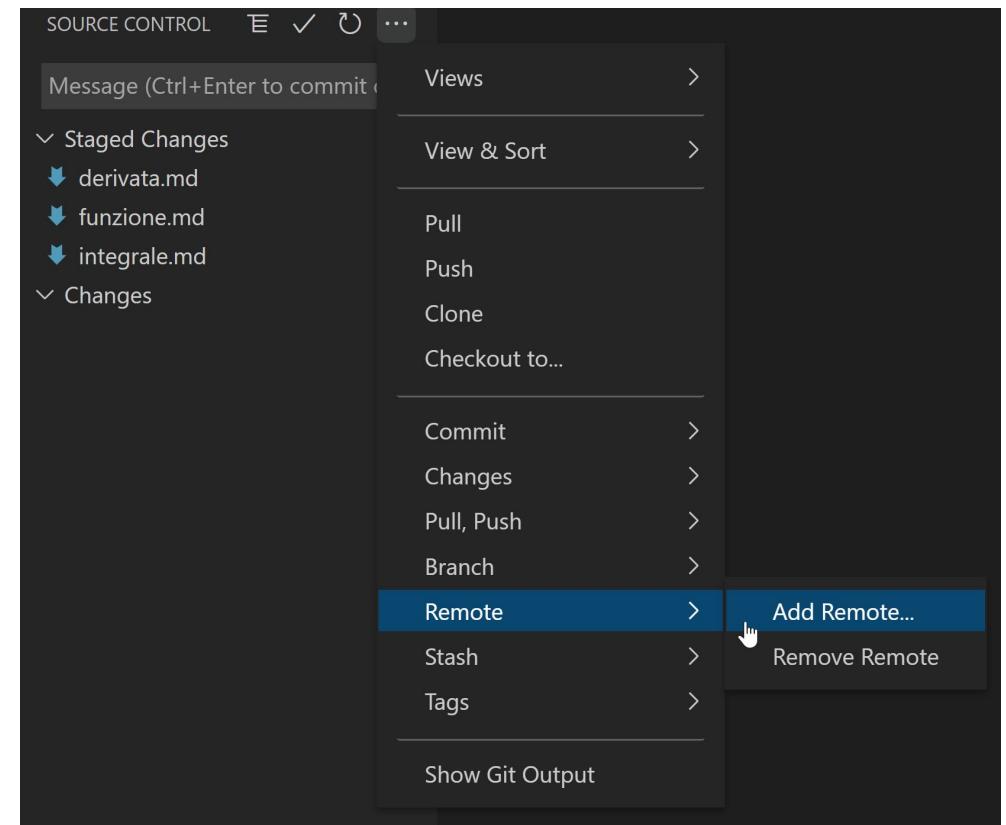
```
echo "# FakeMathSite" > README.md  
git init  
git add README.md  
git commit -m "first commit"  
git branch -M main  
git remote add origin https://github.com/marcozullich/FakeMathSite.git  
git push -u origin main
```

Ignoriamo tutto il resto

COLLEGAMENTO CON LA REPO LOCALE (I)

Torniamo su VSCode

Tramite il menu «...» accanto a «Source Control», selezioniamo «Remote» e «Add Remote...»



COLLEGAMENTO CON LA REPO LOCALE (II)

Incolliamo l'URL che abbiamo prima copiato

`https://github.com/marcozullich/FakeMathSite.git`

Add remote from URL `https://github.com/marcozullich/FakeMathSite.git`

 Add remote from GitHub

e scegliamo un nome da dare al remote. Normalmente si usa il nome «origin» per il primo remote

origin

Please provide a remote name (Press 'Enter' to confirm or 'Escape' to cancel)

COLLEGAMENTO CON LA REPO LOCALE (III)

- VSCode richiederà l'accesso alla vostra repo remota
- Effettuiamo il login
- Autorizziamo Git Credential Manager
- Ora possiamo effettuare *push* e *pull* automaticamente



Authorize Visual Studio Code to access GitHub

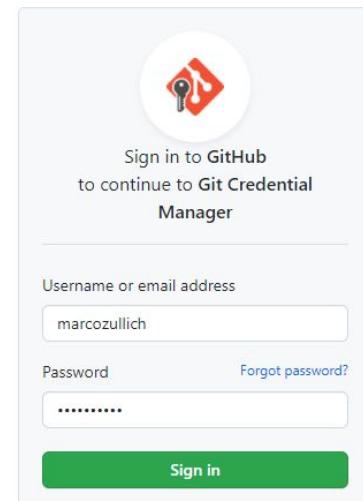
If you initiated this authorization from Visual Studio Code, click 'Continue' to authorize access to GitHub

[Continue](#)

[Do not authorize](#)



Authorize Git Credential Manager



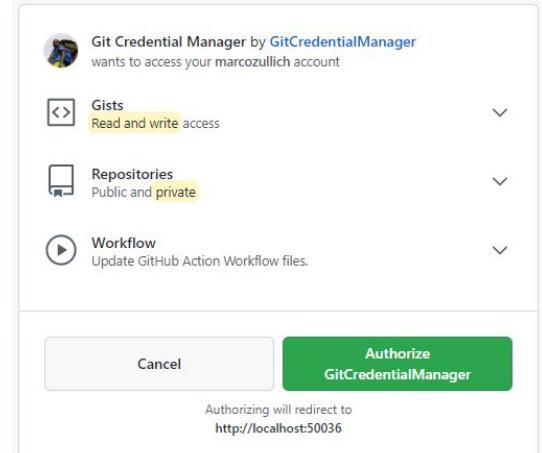
Sign in to GitHub to continue to Git Credential Manager

Username or email address: marcozullich

Password: [Forgot password?](#)

.....

[Sign in](#)



Git Credential Manager by [GitCredentialManager](#) wants to access your marcozullich account

Gists: Read and write access

Repositories: Public and private

Workflow: Update GitHub Action Workflow files.

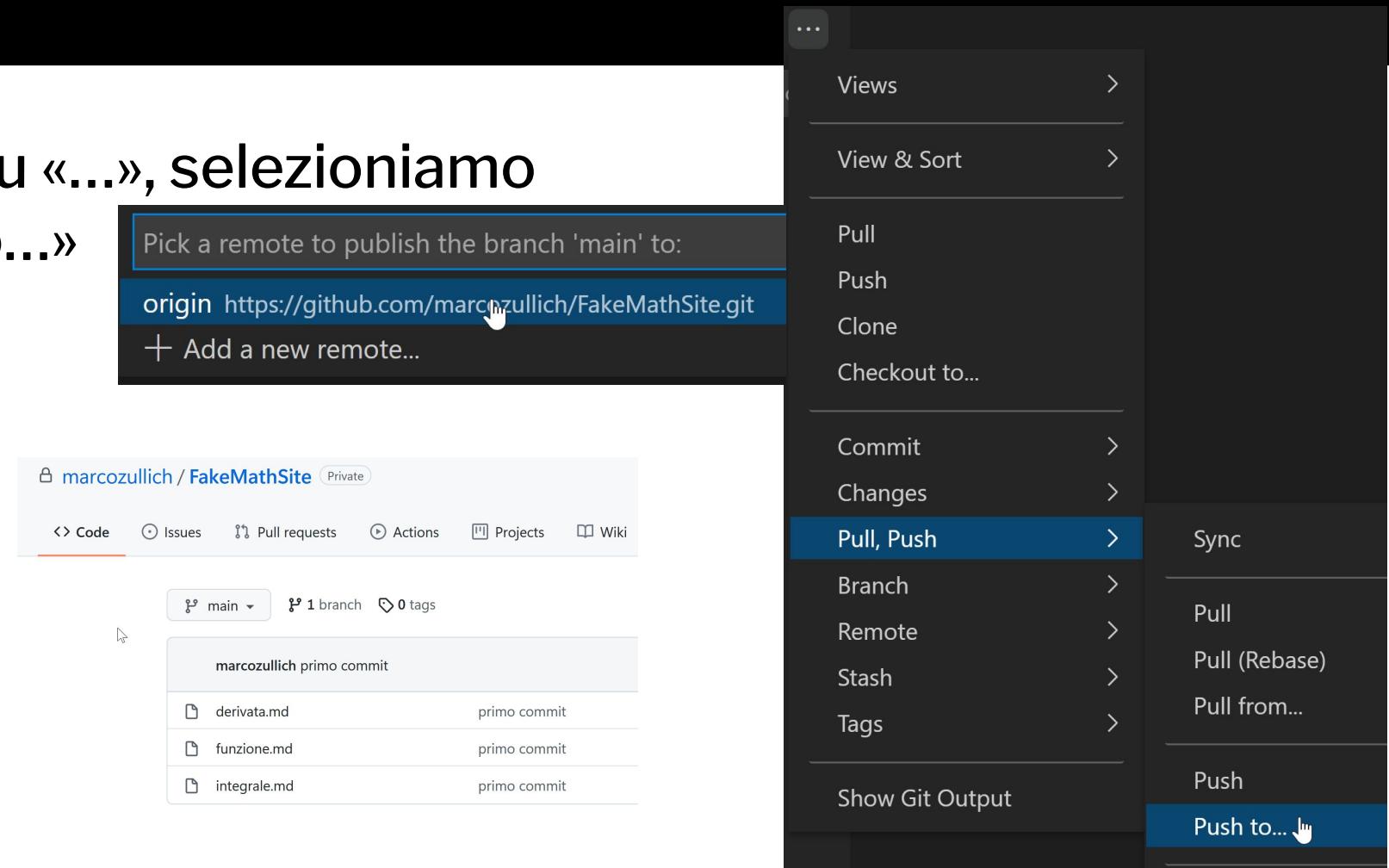
[Cancel](#) [Authorize GitCredentialManager](#)

Authorizing will redirect to <http://localhost:50036>

IL NOSTRO PRIMO PUSH...

Sempre dal menu «...», selezioniamo
«Push», «Push to...»
...e «origin»

Ricarichiamo ora la pagina
di GitHub della repo, e
noteremo qualcosa di
diverso...



The image shows a GitHub repository page for 'marcozullich / FakeMathSite'. The repository has 1 branch and 0 tags. The first commit is by 'marcozullich' and is labeled 'primo commit'. Below the commit, there are three files: 'derivata.md', 'funzione.md', and 'integrale.md', all with the same 'primo commit' message.

To the right of the repository page is a screenshot of a Git context menu. The menu items include: Views, View & Sort, Pull, Push, Clone, Checkout to..., Commit, Changes, Pull, Push, Branch, Remote, Stash, Tags, Show Git Output, and Push to... (which is highlighted in blue).

PER CHI HA MANDATO IL NOME DELL'ACCOUNT GITHUB ENTRO LA DEADLINE...

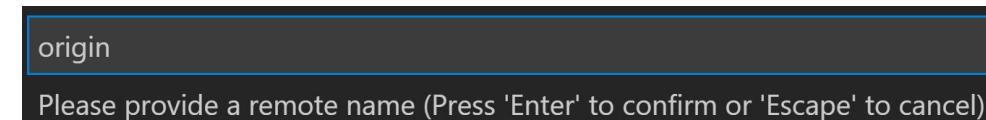
Siete stati aggiunti come collaboratori del mio progetto, che ho già caricato su GitHub prima della lezione.

Il link alla mia repo GitHub è il seguente:

<https://github.com/marcozullich/FakeMathSite>

Aggiungete il seguente URL come remote verso origin

<https://github.com/marcozullich/FakeMathSite.git>



CLONE ()



Possiamo anche decidere di ottenere una **copia locale** di una repo a cui ho accesso.

Questa azione è detta **clone**.

Chi mi ha mandato il nome dell'account GitHub può procedere a clonare la seguente repo

<https://github.com/marcozullich/FakeMathSite.git>

CLONE (II)

Per clonare da VSCode:

- Apro una nuova finestra di Code
- Digito F1
- Nel menu che si apre, digito gitclone



In collo qui l'URL <https://github.com/marcozullich/FakeMathSiteCollab.git>

Dopodiché, devo scegliere con il File Explorer dove clonare la repo.

Completato ciò, posso provvedere ad aprire la cartella con VSCode.

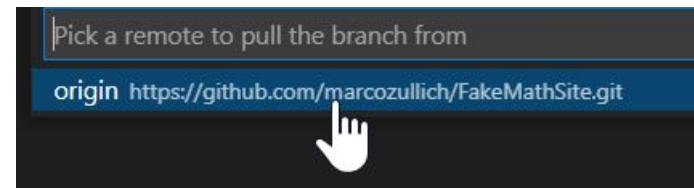
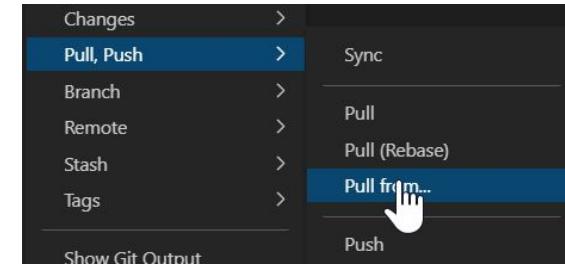
LASCIATEMI FARE ALCUNE
MODIFICHE...

IL NOSTRO PRIMO PULL

marcozullich aggiunta fz suriettiva e iniettiva	
derivata.md	primo commit
funzione.md	aggiunta fz suriettiva e iniettiva
integrale.md	primo commit

Per integrare questa aggiunta al vostro repo locale, potete direttamente fare un pull da VSCode

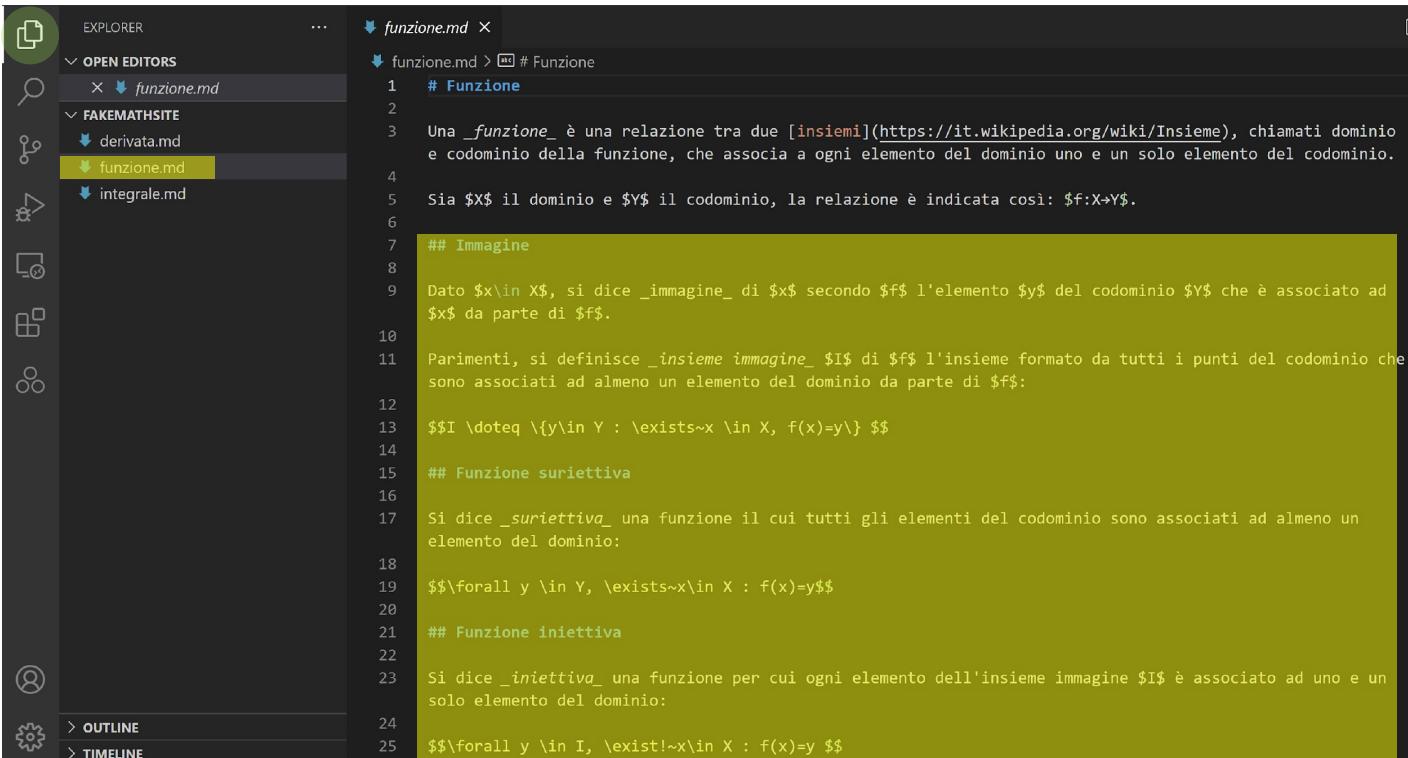
Se date un'occhiata alla mia repo, noterete che c'è un ulteriore commit rispetto al «primo commit»



VEDIAMO I RISULTATI

Se torniamo sul menu
Explorer...
E apriamo il file funzione.md

Noteremo un file più
abbondante di prima



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar lists files in the 'FAKEMATHSITE' folder: 'derivata.md', 'funzione.md' (which is currently selected and highlighted in yellow), and 'integrale.md'. On the right, the main editor pane displays the content of the 'funzione.md' file:

```
funzione.md x
funzione.md > # Funzione
1 # Funzione
2
3 Una _funzione_ è una relazione tra due [insiemi](https://it.wikipedia.org/wiki/Insieme), chiamati dominio e codominio della funzione, che associa a ogni elemento del dominio uno e un solo elemento del codominio.
4
5 Sia $X$ il dominio e $Y$ il codominio, la relazione è indicata così: $f:X\rightarrow Y$.
6
7 ## Immagine
8
9 Dato $x\in X$, si dice _immagine_ di $x$ secondo $f$ l'elemento $y$ del codominio $Y$ che è associato ad $x$ da parte di $f$.
10
11 Parimenti, si definisce _insieme immagine_ $I$ di $f$ l'insieme formato da tutti i punti del codominio che sono associati ad almeno un elemento del dominio da parte di $f$:
12
13 \$I \doteq \{y\in Y : \exists x\in X, f(x)=y\} \$ 
14
15 ## Funzione suriettiva
16
17 Si dice _suriettiva_ una funzione il cui tutti gli elementi del codominio sono associati ad almeno un elemento del dominio:
18
19 \$\forall y \in Y, \exists x\in X : f(x)=y\$
20
21 ## Funzione iniettiva
22
23 Si dice _iniettiva_ una funzione per cui ogni elemento dell'insieme immagine $I$ è associato ad uno e un solo elemento del dominio:
24
25 \$\forall y \in I, \exists! x\in X : f(x)=y \$
```

INTEGRARE MODIFICHE LOCALI

Ammettiamo ora di andare avanti con la modifica dei file, ad esempio aggiungendo un paragrafo a derivata.md

Salviamo il file.

```
# Derivata

La *derivata* di una funzione rappresenta il _**tasso di cambiamento**_ di questa funzione rispetto ad una variabile.
Per tasso di cambiamento si intende la misura di quanto il valore della funzione cambi al variare di questa variabile.

#### Definizione formale

La derivata di una funzione reale  $f: \mathbb{R} \rightarrow \mathbb{R}$  in un punto  $x_0$  si definisce come


$$f' (x_0) = \lim_{h \rightarrow 0} \frac{f(x_0 + h) - f(x_0)}{h}$$

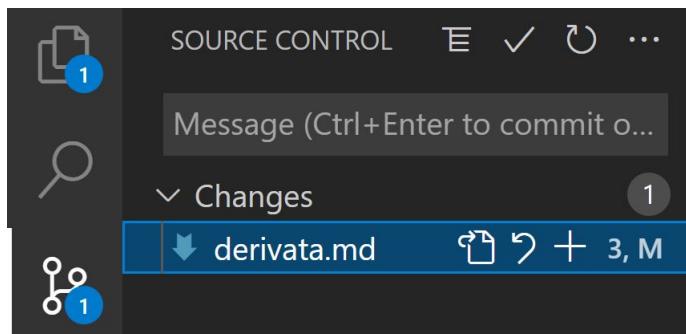

## Interpretazione geometrica

Nel caso della funzione reale  $f$ , possiamo interpretare la derivata  $f'(x_0)$  come la pendenza della retta tangente ad  $f$  nel punto  $x_0$ .
```

MENU SOURCE CONTROL

Diamo un'occhiata alla barra laterale: vediamo che sul menu «Source Control» è presente 1 notifica

Spostandoci all'interno di esso, vediamo di cosa si tratta...



Vediamo che la modifica operata a derivata.md è stata intercettata da Git, che ora ci propone di aggiungere il file alla repo (bottone «+»)

VISUALIZZAZIONE MODIFICHE

Cliccando sul file (NB: NON sul «+») possiamo andare a vedere quali sono le modifiche apportate al file

Vengono proposte le due versioni del file affiancate:
Precedente | Nuova

Possiamo immediatamente notare come le aggiunte sono state automaticamente evidenziate a destra

E la fila di «+» ci indica che quelle sono effettivamente aggiunte rispetto alla versione precedente

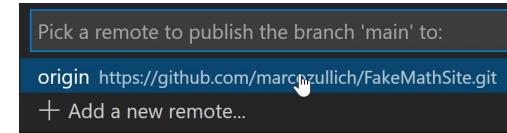
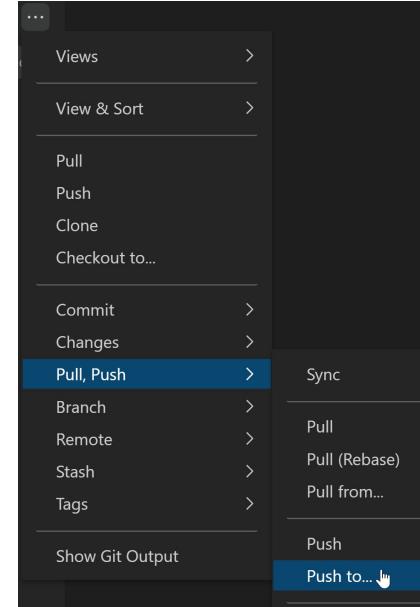
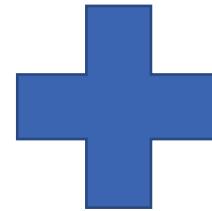
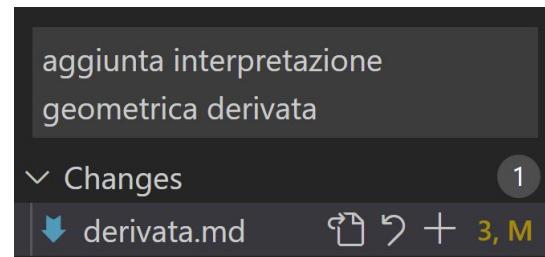
```
derivata.md 3, M • derivata.md (Working Tree) 3, M •
Message (Ctrl+Enter to commit or ...)
Changes 1
derivata.md 3, M • derivata.md (Working Tree) 3, M •
1 La *derivata* di una funzione rappresenta il
  **tasso di cambiamento** di questa funzione
  rispetto ad una variabile.
2 Per tasso di cambiamento si intende la misura di
  quanto il valore della funzione cambi al variare di
  questa variabile.
3
4 #### Definizione formale
5
6 La derivata di una funzione reale $f:\mathbb{R} \rightarrow \mathbb{R}$ in un punto $x_0$ si
  definisce come
7
8 $f'(x_0) = \lim_{h \rightarrow 0} \frac{f(x_0 + h) - f(x_0)}{h}$
9
10 $f'(x_0) = \lim_{h \rightarrow 0} \frac{f(x_0 + h) - f(x_0)}{h}$
11+
12+ ## Interpretazione geometrica
13+
14+ Nel caso della funzione reale $f$, possiamo
  + interpretare la derivata $f'(x_0)$ come la
  + pendenza della retta tangente ad $f$ nel punto
  + $x_0$.
15+
```

COMMIT & PUSH



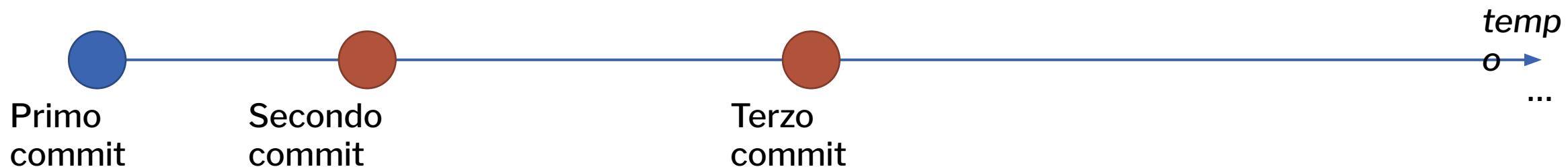
Chi mi ha mandato il suo nome utente di GitHub entro la deadline non deve svolgere questa parte!

Possiamo quindi andare a fare un nuovo commit e push delle nuove modifiche



VISUALIZZAZIONE DELLA STORIA DELLA REPO

Per le funzionalità che abbiamo visto ora di Git, possiamo visualizzare lo sviluppo di una repo come dei punti su una semiretta ordinata nel tempo



REPO LOCALI NON COINCIDENTI

Ogni repo che utilizza lo stesso remote avrà caricata una storia dei commit al suo interno

Ma non è detto che la storia coincida del tutto a quella del remote!

Esempio

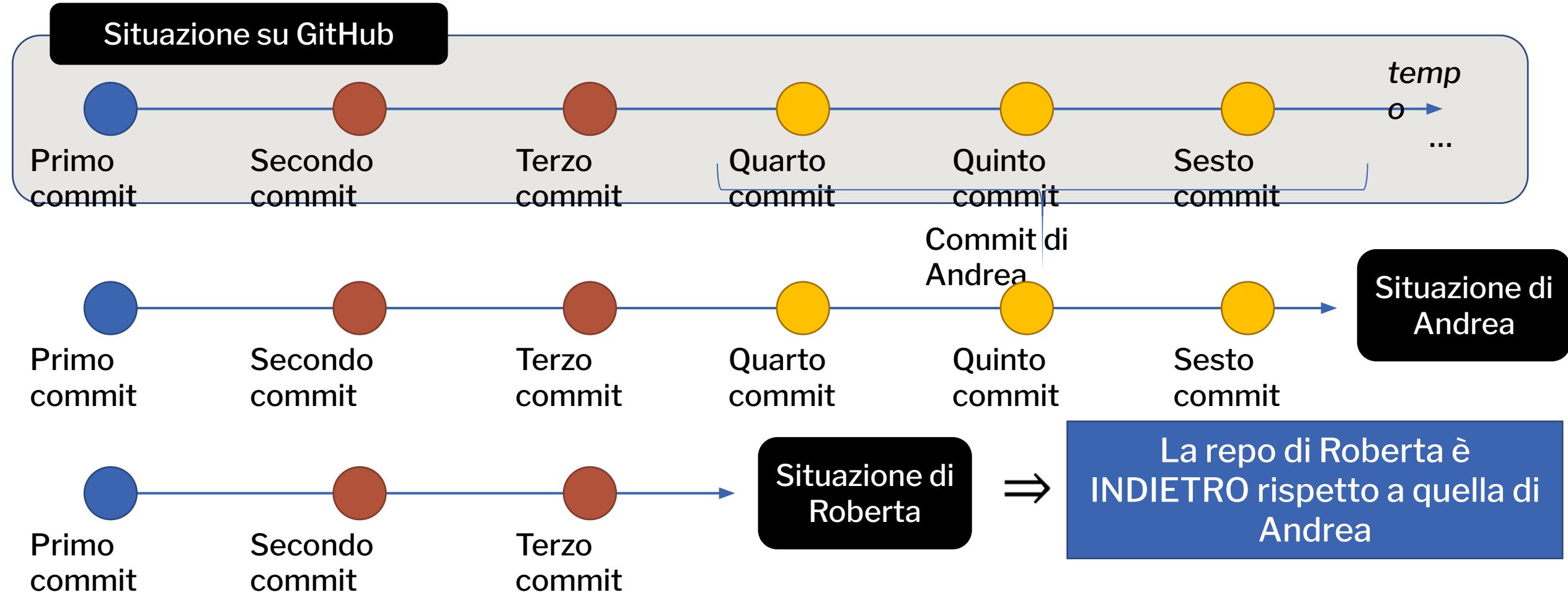
Andrea e Roberta stanno lavorando sulla stessa repo del sito dell'associazione

Roberta va in ferie per 2 settimane

Nel frattempo, Andrea decide invece di lavorare e *pusha* 3 commit sul remote

Domanda: come saranno la repo di Roberta e la repo di Andrea al rientro di Roberta dalle ferie?

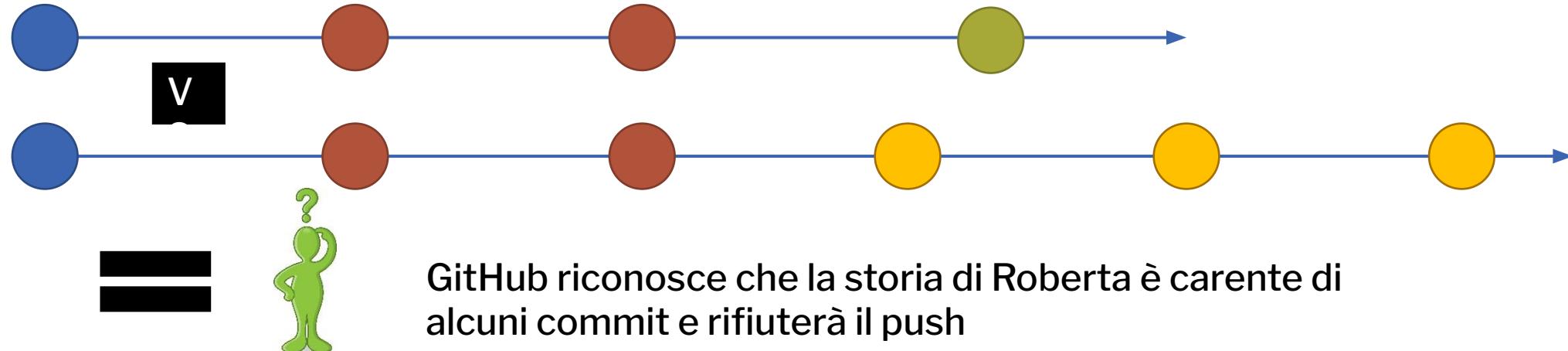
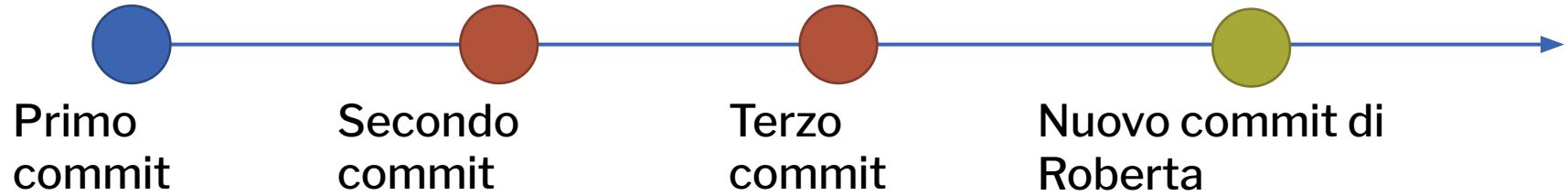
VISUALIZZAZIONE DELLE REPO DI ANDREA E ROBERTA



PROBLEMA DEL LAVORO COLLABORATIVO

Roberta si mette al lavoro e decide di fare un nuovo commit. La sua situazione è ora questa

Roberta decide di pushare il suo lavoro. Cosa pensate succederà?



CHE COSA AVREBBE DOVUTO FARE ROBERTA?

Roberta, al suo rientro dalle ferie, avrebbe dovuto innanzitutto fare un pull per mettere la sua repo in pari con quella di GitHub (e di Andrea)

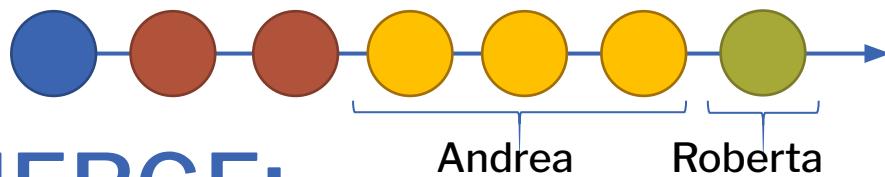
Prima di mettersi al lavoro, è buona norma fare un pull per evitare queste situazioni

MERGE E MERGE CONFLICT

E ora?

Roberta dovrà fare un pull per integrare le modifiche di Andrea
Tuttavia, noi non sappiamo quali file siano stati modificati da Andrea
Potrebbe essere che le due situazioni non siano compatibili
Es. Andrea e Roberta hanno entrambi modificato lo stesso paragrafo di un file

Se le modifiche sono compatibili



MERGE:

Git integra i commit di Andrea e vi pospone il commit di Roberta

Se le modifiche sono incompatibili

Git non sa come integrare il commit di Roberta con quelli di Andrea

Si verifica un cosiddetto **merge conflict**

I merge conflict vanno risolti a mano da Roberta indicando quali parti della versione di Andrea e di quali della versione di Roberta vanno mantenute

5. VERSION CONTROL SYSTEMS

Git da linea di comando

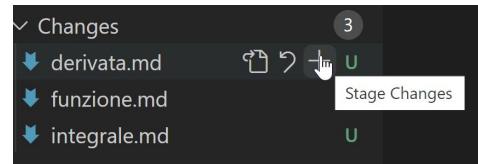
GIT DA LINEA DI COMANDO

VSCode offre una versione user-friendly di Git con una GUI
Tuttavia, è anche possibile utilizzare Git da shell testuale
Vedremo rapidamente come le varie fasi di gestione della repo
possono essere usate da CLI
Innanzitutto, per lavorare correttamente con Git da CLI dobbiamo
impostare la working directory alla base della cartella Git
Posso creare la repo localmente tramite il comando git init

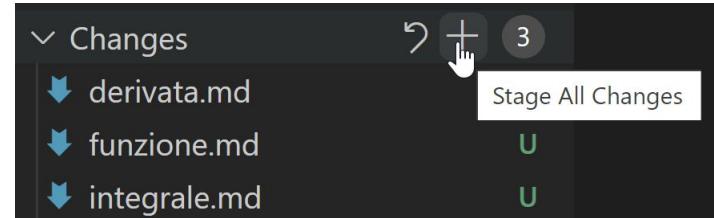
ADD

git add è il comando per aggiungere file alla repo

Possiamo specificare singoli file, es. git add file1 file2 ...



Se vogliamo aggiungere tutti i file modificati in blocco, possiamo specificare git add –A



STATUS

Ricordiamo che su VSCode vedevamo rapidamente quali file sono stati modificati e quali modifiche siano state aggiunte alla repo pronte per il commit.

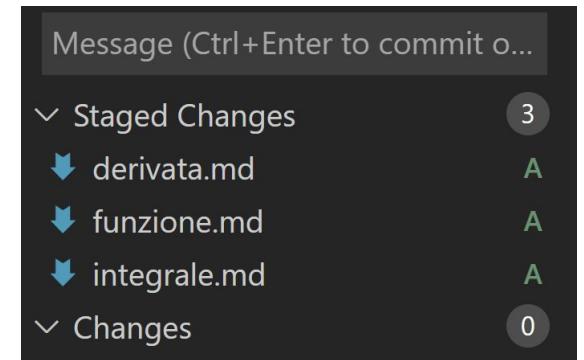
Lo stesso lo possiamo fare con git status

Nessuna modifica pronta per il commit (non abbiamo ancora fatto git add)

```
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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:   derivata.md

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



È presente una modifica che non è stata ancora aggiunta alla repo

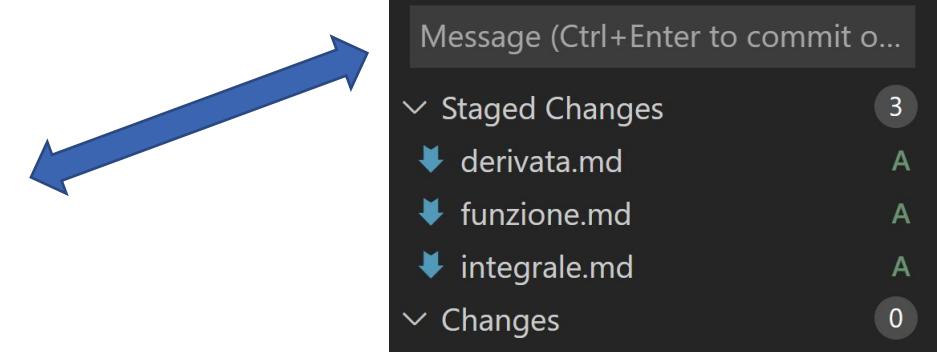
AGGIUNTA DI UN REMOTE

Il remote può essere aggiunto con il seguente comando

```
git remote add <NomeRemote> <URLremote>
```

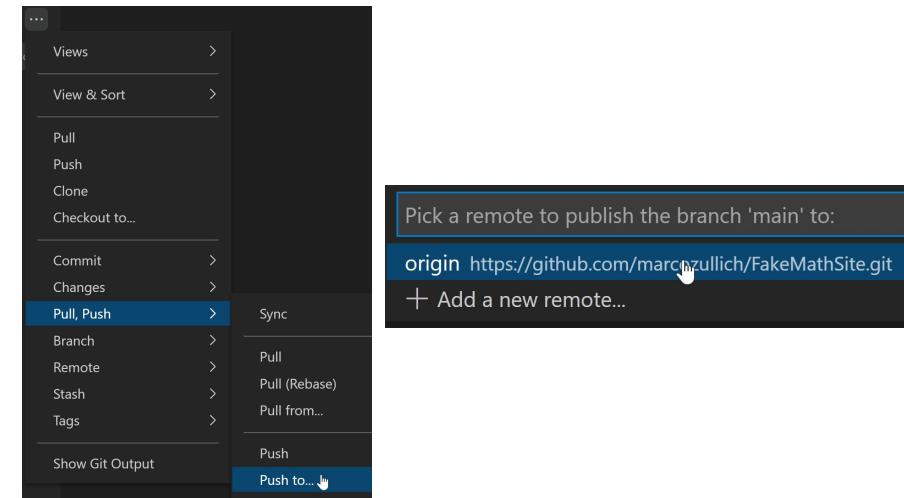
COMMIT, PUSH E PULL

Possiamo fare il commit digitando
`git commit -m "messaggio commit"`



Per il push invece indichiamo
`git push <NomeRemote>` (es. origin)

Analogamente, per il pull
`git pull <NomeRemote>` (es. origin)



LOG

git log è una funzionalità aggiuntiva rispetto a VSCode

Ci permette di vedere la storia dei vari commit (comprensivi del loro codice)

```
mzullich@CYB-TS-100 MINGW64 /d/FakeMathSite (main)
$ git log
commit 3bf902cf78d5d7372f595477823723e423634eef (HEAD -> main, origin/main)
Author: marcozullich <zuzzu90@gmail.com>
Date:   Wed Sep 8 16:42:58 2021 +0200

    aggiunta fz suriettiva e iniettiva

commit fed63f7ebbdca6f852a1205e1f67d7282edd8c8b
Author: marcozullich <zuzzu90@gmail.com>
Date:   Wed Sep 8 16:23:50 2021 +0200

    primo commit
(base)
```

UTILIZZO DEL TERMINALE IN VSCODE

VSCode ci dà anche la possibilità di usufruire di un terminale integrato



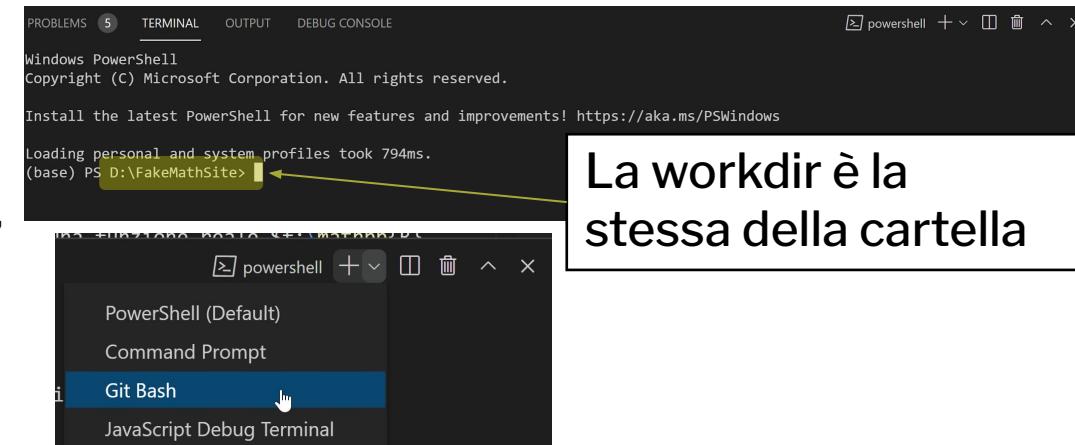
Selezioniamo «Terminal» → «New Terminal» dalla barra menu in alto

Di default su Win si apre una Powershell di Windows

(su Linux si apre una normale shell bash)

Possiamo anche selezionare altri terminali, come, ad esempio Git Bash (su Win) tramite il bottone

Possiamo utilizzare questo terminale per tutte le funzionalità viste finora (Git compreso)



RIASSUMENDO

`git init`

Inizializza repo Git
locale

`git add <files>`

Aggiunge file alla
repo

`git commit -m <messaggio>`

Crea un checkpoint con un dato
messaggio

`git remote add <nome> <URL>`

Aggiunge un remote (*cloud*) alla
repo

`git pull <nomeRemote>`

Scarica i commit in eccedenza dal remote indicato e li aggiunge alla
repo locale

`git push <nomeRemote>`

Carica i commit in eccedenza dalla repo locale al remote
indicato

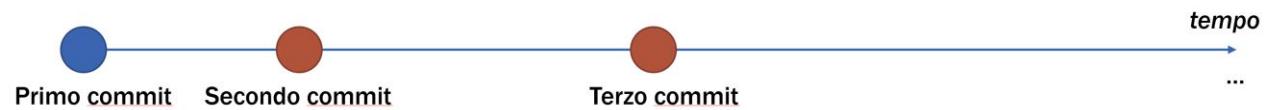
Comand
i locali

Comand
i remoti

BRANCH (I)

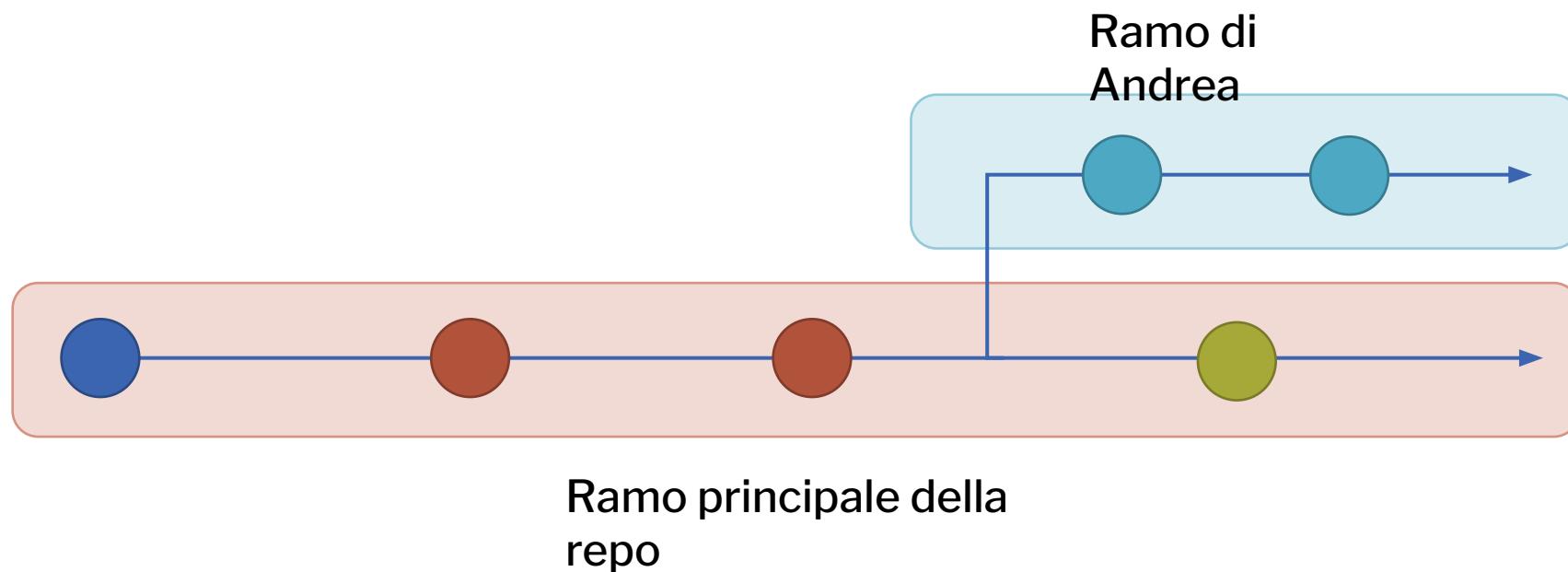
Recap:

Per le funzionalità che abbiamo visto ora di Git, possiamo visualizzare lo sviluppo di una repo come dei punti su una semiretta ordinata nel tempo



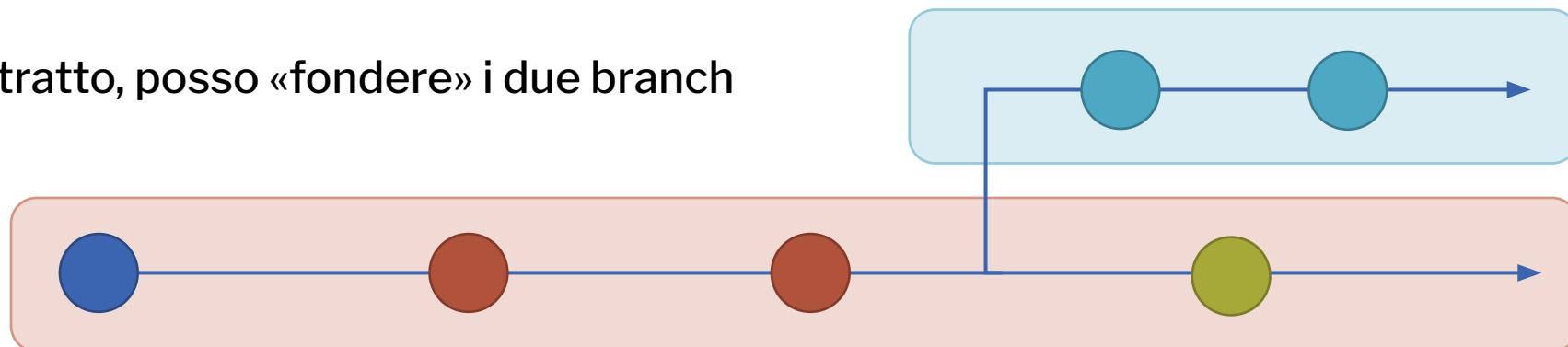
Git ammette anche la ramificazione di una storia

BRANCH (II)

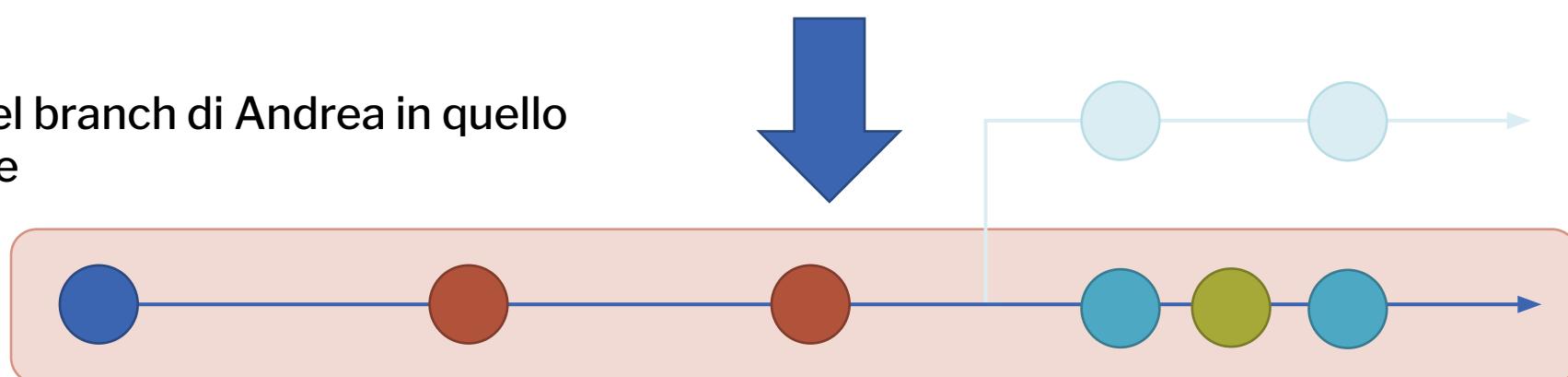


MERGE DEI BRANCH

Ad un certo tratto, posso «fondere» i due branch assieme

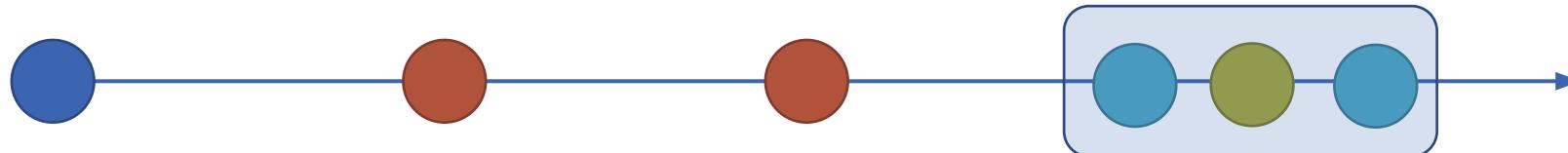


Merge del branch di Andrea in quello principale

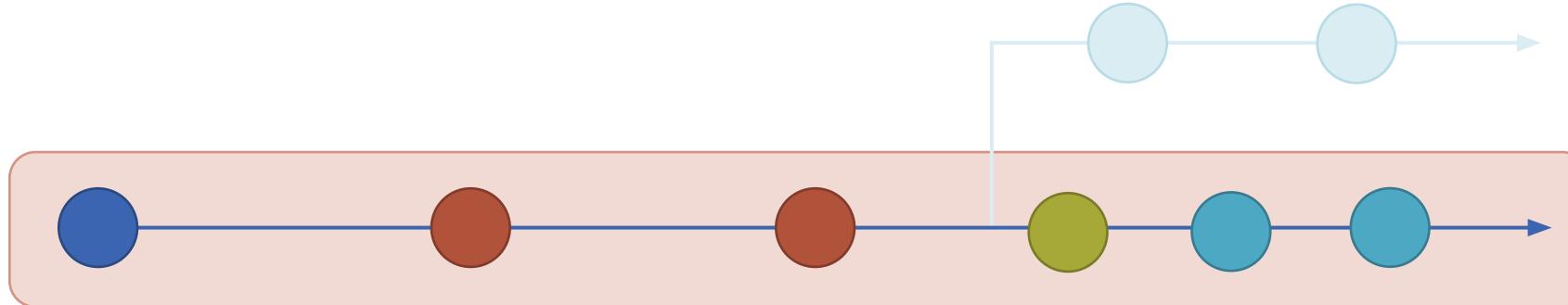


Il branch di Andrea continua comunque ad esistere

REBASE



Vediamo che la storia è un po'
mescolata



Il rebase, invece, aggiunge i commit del branch
di Andrea in testa al branch principale

RIASSUNTO

<https://girliemac.com/blog/2017/12/26/git-purr/>

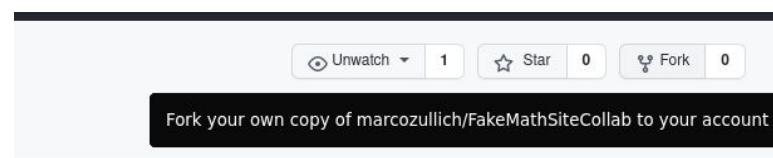
5. VERSION CONTROL SYSTEMS

Azioni repo da remoto

FORK



GitHub ci dà inoltre la possibilità di fare un fork della repository

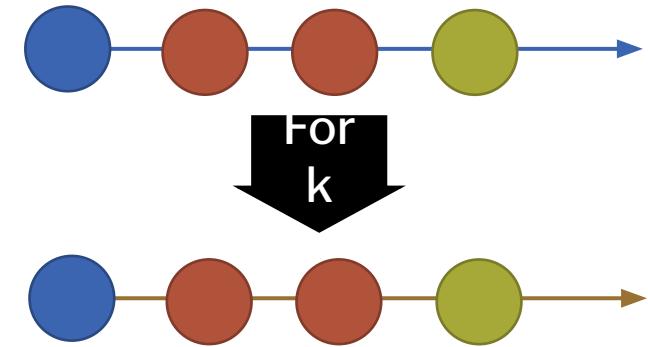


Un fork è una copia remota e personale della repo

Possiamo usarla per scaricarci una repo pubblica di un altro programmatore

E modificarla secondo le nostre necessità

github.com/TizioCaio/cool_program.git



github.com/marcozullich/cool_program.git

STORIE IN COMUNE

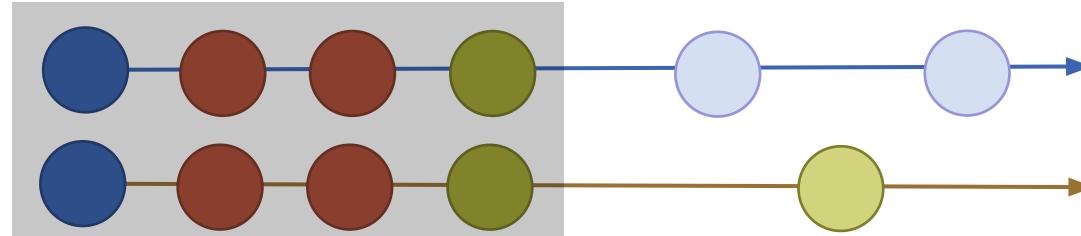
Un fork può essere visto come un branch della repo principale

Solo che la repo viene "incollata" in un'altra repository

E al suo interno vengono copiate anche tutti i branch già esistenti nella repo principale

Git è in grado di riconoscere che una repo è un fork di un'altra in quanto hanno una storia in comune

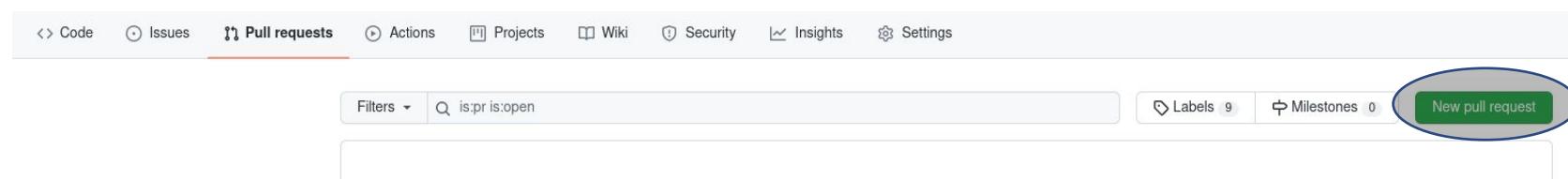
github.com/TizioCaio/cool_program.git



github.com/marcozullich/cool_program.git

PULL REQUEST (I)

Posso fare il merge di due repo di cui una è fork di un'altra tramite un'azione detta Pull Request



Tramite la pull request, io richiedo al/ai proprietario/i di una repo di integrare le modifiche da me effettuate nella sua repo principale

Praticamente, funziona come un merge, ma la cosa viene fatta tutta a livello remoto (risoluzione conflitti, commit...)

PULL REQUEST (II)

La Pull Request è in generale uno strumento per effettuare un merge di due branch in maniera remota

 NB: questo significa che le modifiche apportate dal merge non venir integrate a livello locale con un *pull*

Il branch può provenire sia da un fork della repo che dalla stessa repo



Pull request per merge di due branch (main e ramo) della stessa repo



Pull request per merge di due branch (09_part2 e master) di due repo differenti, una il fork di un'altra

LA PROSSIMA VOLTA

Inizieremo ad introdurre concetti basilari di programmazione:

Vedremo uno schema di computer semplificato per eseguire programmini molto semplici

Inizieremo a scrivere alcuni programmi basilari in pseudocodice

Vedremo un'implementazione reale del computer semplificato: il TC²

Impareremo a *compilare* e ad eseguire un programma in pseudocodice nel linguaggio del TC²