

# GIT

## Controllo di versione

Marco Alberti



**Dipartimento  
di Matematica  
e Informatica**



**Università  
degli Studi  
di Ferrara**

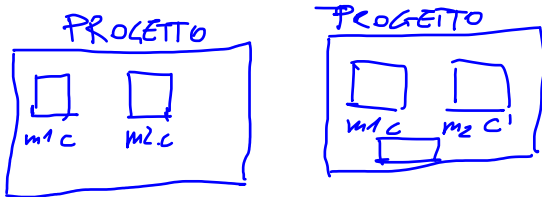
Programmazione e Laboratorio, A.A. 2021-2022

Ultima modifica: 3 ottobre 2021

Attenzione! Questo materiale didattico è per uso personale dello studente ed è coperto da copyright.  
Ne sono vietati la riproduzione e il riutilizzo anche parziale, ai sensi e per gli effetti della legge sul diritto d'autore.

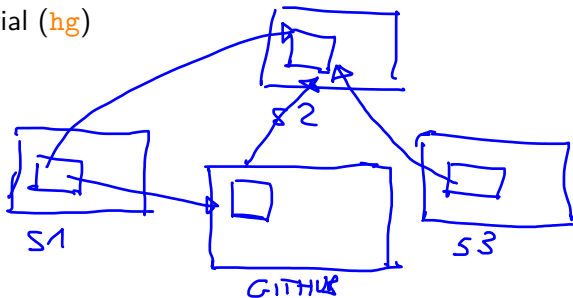
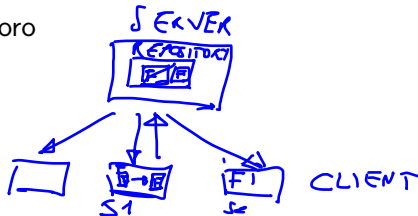
# Controllo di versione

- La cartella contenente i file sorgenti di un progetto viene inevitabilmente modificata nel tempo (aggiunta di funzionalità, correzione di errori)
- I sistemi di controllo di versione permettono di archiviare e, all'occorrenza, recuperare tutte le versioni significative della cartella in un **repository**, permettendo
  - Confronto con versioni precedenti
  - Ripristino di una versione precedente (*rollback*)



# Sistemi di controllo di versione

- Locali: repository sul computer di lavoro
  - RCS
- Centralizzati: repository su server
  - CVS
  - Subversion (**svn**)
- Distribuiti: repository su più computer (alcuni possono fungere da server)
  - Git
  - Mercurial (**hg**)



- Autore: Linus Torvalds (lo stesso di Linux)
- Attualmente è il sistema di controllo di versione distribuito dominante
- Disponibile per
  - Windows: <https://git-scm.com/download/win>
  - Linux: ad esempio in Debian e Ubuntu `sudo apt-get install git`
  - MacOSX: <https://git-scm.com/download/mac>
- Cheat sheet:  
<https://education.github.com/git-cheat-sheet-education.pdf>
- Tutorial: <https://learngitbranching.js.org/>

# Utilizzo di Git da linea di comando

Sintassi comandi: `git __comando__`

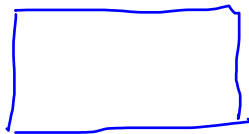
Ad esempio il comando `add esercizio.c` va scritto come `git add esercizio.c`.

`git init`

crea un nuovo repository nella cartella corrente

`git status`

eseguito in un repository Git, mostra informazioni sul suo stato (ad esempio quali file aggiunti al repository sono stati modificati).



## Repository remoti

Oltre ai repository (**locali**) sul computer di lavoro, Git permette di gestire repository (**remoti**) su altri computer e di sincronizzarli con quelli locali.

Servizi di repository remoti (previa registrazione)

- BitBucket: <https://bitbucket.org/>
- • GitHub (il più usato per progetti open source): <https://github.com/>
- GitLab: <https://gitlab.com/>

## Accesso a repository remoti (soggetto a permessi)



```
git clone __url__
```

crea in locale un clone del repository identificato da `__url__`.

### Esempio

```
git clone https://github.com/lbrmrc/Programmazione2021
```

crea in locale una copia del repository degli esempi di questo corso.

```
git pull
```

(eseguito in una cartella Git) aggiorna il repository all'ultima versione del repository remoto.


### Esercizio

Creare una nuova cartella. Dentro essa, clonare il repository degli esempi di codice del corso.

Quando il repository remoto viene aggiornato, si può aggiornare la propria copia all'ultima versione con il comando `git pull`.

- 1 Collegarsi al sito <https://github.com/>.
- 2 Creare un proprio utente (link "Sign Up", usando il proprio indirizzo email [@edu.unife.it](mailto:@edu.unife.it)). Collegarsi con l'utente appena creato.
- 3 Creare un nuovo repository **privato**, cioè accessibile solo al proprietario (quelli **pubblici** sono visibili a tutti), di nome **Programmazione** configurato così:


Owner \*      Repository name \*


 lbmrnc / Programmazione ✓

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

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:**  
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.](#)

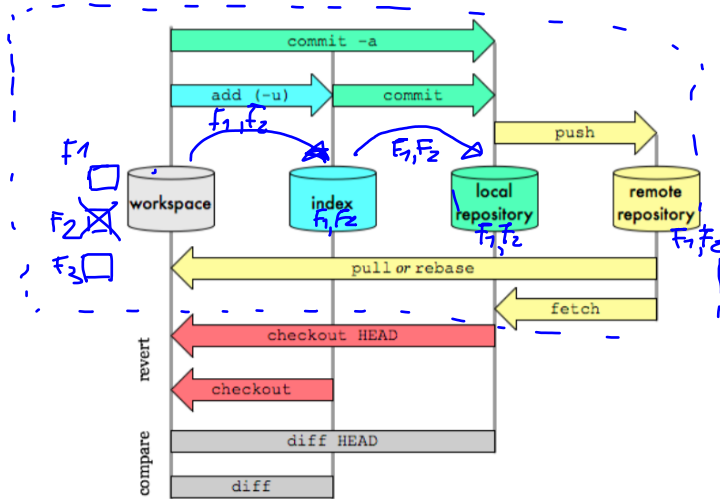
☐ **Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

This will set  **main** as the default branch. Change the default name in your [settings](#).

[Create repository](#)



# Principali comandi Git



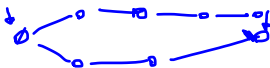
(credit: <http://krishnaiitd.github.io/gitcommands/>)



Recentemente, GitHub ha disabilitato l'accesso HTTPS tramite nome utente e password, richiedendo l'uso (più granulare) dei Personal Access Token, una sorta di password ad hoc.

- ① Nella propria pagina utente GitHub, selezionare Settings → Developer settings → Personal access tokens
- ② Attivare la generazione di un nuovo token
- ③ Selezionare
  - una nota che lo renda riconoscibile (ad esempio "Esercizi programmazione")
  - una scadenza adatta all'uso (ad esempio fino al 28 febbraio 2022)
  - lo scope "repo" visto che lo si userà per gestire i repository
- ④ Cliccare su "Generate new token"
- ⑤ Copiare e incollare il token in un posto sicuro (un file di testo nella macchina virtuale?)

Ora si può usare il PAT al posto della password quando richiesto dalla linea di comando. Il PAT si può revocare dalla pagina utente. Si possono creare nuovi PAT se necessario.



L'indirizzo Git del repository remoto, che si usa nei comandi sul repository, si trova solitamente nella pagina web del repository. Ce ne sono di due tipi: HTTPS e SSH (quest'ultimo è preferibile ma richiede un po' di configurazione).

- 1 Clonare il repository **Programmazione** creato nell'esercizio alla slide 8, usando il link HTTPS, in una nuova cartella (non dentro un altro repository)
- 2 Spostarsi nella cartella del repository clonato
- 3 Impostare il proprio nome e il proprio indirizzo email per il repository con i comandi

- `git config user.name "__nome e cognome__"`
- `git config user.email "__indirizzo email__"`

PINCO PALLINO

pincopallino@edu.unife.it

~~master~~

- 4 Selezionare il ramo principale (v. slide 17) con il comando `git checkout main`.
- 5 Nella cartella del repository clonato, salvare un file di nome **dati.txt** contenente solo la parola "Primo".

## Esercizio II

- 6 Aggiungere il file appena creato all'indice con il comando `git add dati.txt`.
- 7 Eseguire un commit dell'indice con il comando `git commit -m "Prima versione"`.
- 8 Modificare il file sostituendo la parola "Primo" con la parola "Secondo" e salvarlo.
- 9 Eseguire un commit di tutte le modifiche con il comando `git commit -a -m "Seconda versione"`.

# Storia delle versioni del repository

## `git log`

mostra l'elenco di tutti i commit fatti nel repository. Ogni commit è identificato da una sorta di lungo numero esadecimale

## `git checkout __id-commit__`

dove `__id-commit__` è la parte iniziale (almeno 6 caratteri) dell'identificatore di un commit, riporta la cartella di lavoro allo stato corrispondente a quel commit.

## `git checkout master`

esegue il checkout del commit più recente del ramo principale (detto `master`, o `main`)

- 1 Visualizzare l'elenco dei commit del repository con il comando `git log`. Dovrebbe assomigliare a questo:

```
commit ddafa576ca9b663ef9350b67249d7ce5da3cd8a6
Author: Marco Alberti <malbertife@gmail.com>
Date:   Sun Oct 6 11:47:20 2019 +0200
```

Seconda versione

```
commit de3e6cd1c247cac8298f2570e8e82d5b704cfe47
Author: Marco Alberti <malbertife@gmail.com>
Date:   Sun Oct 6 11:46:55 2019 +0200
```

Prima versione

```
commit ce58986b01b7348f047099a79145daf61e3af8fe
Author: Marco Alberti <malbertife@gmail.com>
Date:   Sun Oct 6 09:31:28 2019 +0000
```

Initial commit

- ② Eseguire il checkout del primo commit con il comando `git checkout __ID__` (dato il log precedente, `__ID__` sarebbe `de3e6c`).
- ③ Visualizzare il contenuto del file `dati.txt` (ad esempio con il comando `cat dati.txt`).
- ④ Riportare il repository all'ultimo commit con il comando `git checkout master`.
- ⑤ Visualizzare il contenuto del file `dati.txt`.

# Modifiche locali e repository remoto

Le modifiche locali (anche se **committate**) a un repository ottenuto per clonazione da un repository remoto non si riflettono automaticamente su quello remoto.

## Esercizio

Verificare che il repository remoto è invariato.

## **git push**

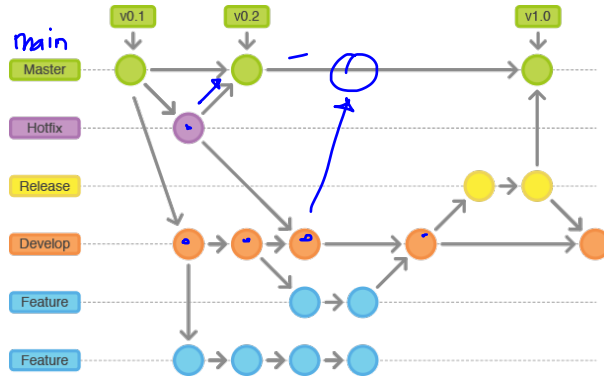
carica sul repository remoto gli ultimi commit del repository locale.

## Esercizio

- 1 Nella cartella locale del repository, dare il comando **git push** per inviare al repository remoto i commit locali
- 2 Verificare che il repository remoto è stato aggiornato con tutte le modifiche:
  - i file sono all'ultima versione
  - nel link "commits" sono elencati tutti i commit, che si possono esaminare



# Branching



Principali comandi: **branch**, **checkout**, **merge**.

## Collegamento fra repository remoti e locali

Un repository locale, non nato per clonazione di uno remoto ma creato con il comando `git init`, deve essere collegato a quest'ultimo perché sia possibile lo scambio di commit.

```
git remote add __nome__ __indirizzo__
```

collega un repository remoto di indirizzo `__indirizzo__` al repository locale, con nome `__nome__` (tipicamente `origin`).

# Integrazione con Visual Studio Code

- VS Code, come tutti gli editor avanzati, permette di gestire i repository Git e mette a disposizione i principali comandi
- Aprendo una cartella (File -> Apri Cartella o CTRL-K CTRL-O) dentro un repository Git diventano disponibili i comandi Git per il repository (icona Git o CTRL+SHIFT+G)
- Dettagli: <https://code.visualstudio.com/Docs/editor/versioncontrol>
- Estensione consigliata per uso avanzato: Git Lens

## Utilizzo consigliato di Git per questo corso

- Tenere una copia del repository degli esempi (slide 7), usando `git pull` di tanto in tanto per aggiornarla.
- Salvare i propri esercizi nel repository creato alla slide 8 e sincronizzarla dopo ogni sessione di lavoro con il proprio account GitHub (slide 11) in modo da averne una copia sicura e facile da consultare.
- (facoltativo) Dare accesso in lettura e scrittura al docente (utente `lbrmrc`) è il modo più semplice per richiedere commenti e correzioni sui propri esercizi.
- Chi usa i PC del laboratorio (e quindi non usa sempre lo stesso) può clonare il repository a ogni sessione, oppure usare una chiavetta USB.