

MAURO CASADEI

GIT E IL VERSIONING

Introduzione a Git

IL SISTEMA DI VERSIONING DISTRIBUITO

Cos'è Git?

Git è un sistema di controllo di versione distribuito utilizzato per gestire il codice in progetti software.

✓ A cosa serve?

📁 Tiene traccia delle modifiche ai file nel tempo.

👥 Permette a più sviluppatori di collaborare sullo stesso progetto.

⬅️ Consente di tornare a versioni precedenti del codice.

🔍 Come funziona?

Repository locale: Ogni sviluppatore ha una copia completa del progetto.

Commit: Salva una versione specifica del codice.

Branch: Permette di lavorare su funzionalità diverse in parallelo.

Merge: Unisce i cambiamenti di più sviluppatori.

🌐 Perché è utile?

✓ Migliora la collaborazione

✓ Previene la perdita di codice

✓ Rende più semplice risolvere errori e bug

Perché usare Git?

- Tracciamento delle modifiche
- Lavoro in team
- Backup sicuro
- Sperimentazione senza rischi

Installazione di Git

Scarica Git da git-scm.com e segui le istruzioni per l'installazione.

Configurazione iniziale

```
git config --global user.name "Il Tuo Nome"
```

```
git config --global user.email "tuo@email.com"
```

Le istruzioni principali

Istruzioni di base

Comando	Descrizione
<code>git init</code>	Inizializza un nuovo repository Git locale.
<code>git clone <url></code>	Clona un repository remoto sul tuo computer.
<code>git status</code>	Mostra lo stato dei file (modificati, aggiunti o in staging).
<code>git add <file></code>	Aggiunge un file all'area di staging.
<code>git commit -m "messaggio"</code>	Salva una versione del codice con un messaggio descrittivo.

Le istruzioni principali

Gestione dei branch

Comando	Descrizione
<code>git branch <nome-branch></code>	Crea un nuovo branch.
<code>git checkout <branch></code>	Passa a un altro branch.
<code>git merge <branch></code>	Unisce un branch con quello corrente.
<code>git branch -d <branch></code>	Elimina un branch.

Gestione dei branch



Gestione dei branch

Comando	Descrizione
<code>git branch <nome-branch></code>	Crea un nuovo branch.
<code>git checkout <branch></code>	Passa a un altro branch.
<code>git merge <branch></code>	Unisce un branch con quello corrente.
<code>git branch -d <branch></code>	Elimina un branch.

Gestione del repository remoto



Gestione del repository remoto

Comando	Descrizione
<code>git remote add origin <url></code>	Collega il repository locale a quello remoto.
<code>git push origin <branch></code>	Invia i cambiamenti al repository remoto.
<code>git pull origin <branch></code>	Scarica e unisce le modifiche dal remoto.
<code>git fetch</code>	Recupera i dati dal remoto senza unirli.

Gestione della cronologia

Gestione della cronologia

Comando	Descrizione
<code>git log</code>	Mostra la cronologia dei commit.
<code>git reset --soft <commit></code>	Torna a un commit precedente mantenendo le modifiche nello staging.
<code>git reset --mixed <commit></code>	Torna a un commit precedente mantenendo le modifiche nei file.
<code>git revert <commit></code>	Crea un nuovo commit che annulla le modifiche di un commit precedente.

Creare un repository Git

git init -> Inizializza un repository Git in una cartella

```
● PS C:\Corsi\git\repository-test> git init  
Initialized empty Git repository in C:/Corsi/git/repository-test/.git/
```

git clone URL -> Clona un repository esistente

Aggiungere file al repository

`git add nomefile` -> Aggiunge un file al repository

`git commit -m "Messaggio"` -> Conferma le modifiche

Stato del repository

git status -> Mostra lo stato attuale dei file

```
● PS C:\Corsi\git\repository-test> git status
On branch dev2
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:   prodotti.html

no changes added to commit (use "git add" and/or "git commit -a")
○ PS C:\Corsi\git\repository-test> █
```

Storico delle modifiche

git log -> Mostra la cronologia dei commit

```
PS C:\Corsi\ITS Turismo Marche\2024-2026 FullStack Senigallia\database-er> git log
commit b7ffd3234265417b1c6b34115aa77a2e1f1f8719 (HEAD -> main)
Author: soluzione-software <info@soluzionesoftware.com>
Date:   Mon Mar 3 13:06:50 2025 +0100

    test

commit f82adaffac05b48eeb38b9ddcd2a4664ac0d8168 (origin/main)
Author: soluzione-software <info@soluzionesoftware.com>
Date:   Mon Mar 3 12:50:00 2025 +0100

    DISPENSA AGGIORNATA

commit 58936b41cad76d320b22ffa8a263851b7644ef83
Author: soluzione-software <info@soluzionesoftware.com>
Date:   Fri Feb 28 16:27:57 2025 +0100
```

Aggiungere le modifiche

In Git, il comando `git add .` (o `git add *`) serve per aggiungere le modifiche all'area di staging, ovvero un'area temporanea dove Git tiene traccia dei file che vuoi includere nel prossimo commit.

```
git add .
```


Creare e gestire branch

git branch nomebranch -> Crea un nuovo branch

Oppure git checkout -b nomebranch

git checkout nomebranch -> Cambia branch

```
git checkout -b dev2
```

```
git branch dev  
git checkout dev
```

Mostrare i branch

Git branch (lista branch)

Git branch -r (lista branch remoti)

```
PS C:\Corsi\git\repository-test> git branch
dev
* dev2
master
```

Unire branch

git merge nomebranch -> Unisce le modifiche di un branch in un altro

```
● PS C:\Corsi\git\repository-test> git merge dev  
Updating 5fdcd06..3681da5
```

Eliminare un branch

Per eliminare un branch locale:

```
git branch -d nome-del-branch
```

Se il branch **non è stato unito** e vuoi forzare l'eliminazione:

```
git branch -D nome-del-branch
```

```
● PS C:\Corsi\git\repository-test> git branch -d dev2  
Deleted branch dev2 (was f464da7).
```

Git reset

git reset serve per tornare indietro nel tempo e annullare modifiche in Git.

Puoi usarlo per rimuovere commit, modifiche o file dall'area di staging, senza perdere definitivamente i tuoi file.

git reset --soft HEAD~1 _ torna indietro di 1 commit

git reset --soft 8df4b26 torna a quel commit

 Quando si usa `git reset`?

Situazione	Quale usare
Ho committato troppo presto e voglio aggiungere altro	<code>--soft</code>
Ho committato per errore e voglio correggere il codice	<code>--mixed</code>
Voglio cancellare tutto e ricominciare da zero	<code>--hard</code>

--soft e --mixed

`git reset --soft`

Mantiene tutto nell'index (staging area).

Non tocca il working directory (i file modificati rimangono intatti).

Sposta semplicemente il puntatore di HEAD al commit specificato.

Le modifiche risultano ancora "staged", pronte per essere committate di nuovo.

✅ Uso tipico: Quando vuoi cambiare l'ultimo commit senza perdere le modifiche già aggiunte all'index.

`git reset --mixed` (di default)

Resetta l'index (staging area), ma mantiene intatti i file nel working directory.

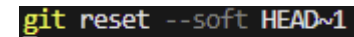
Le modifiche vengono "un-staged" (rimosse dall'index), ma i file modificati rimangono nel progetto.

✅ Uso tipico: Quando vuoi annullare l'ultimo commit e fare delle modifiche ai file prima di aggiungerli di nuovo.

Eliminare una commit locale

Eliminare l'ultima commit (locale, non pushata)

```
git reset --soft HEAD~1
```



```
git reset --soft HEAD~1
```

eliminare completamente anche le modifiche dai file:

```
git reset --hard HEAD~1
```

Se vuoi rimuovere una commit nel mezzo della cronologia:

```
git rebase -i HEAD~N
```

Se hai già pushato e vuoi eliminarla dal repository remoto (**può causare problemi agli altri sviluppatori**):

```
git reset --hard HEAD~1
```

```
git push origin --force
```

Ignorare file

Usa un file .gitignore per specificare i file da ignorare

Rimuovere file dal repository

git rm nomefile -> Rimuove un file tracciato

```
● PS C:\Corsi\git\repository-test> git restore --staged about.html  
● PS C:\Corsi\git\repository-test> git restore about.html  
○ PS C:\Corsi\git\repository-test> █
```

Per recuperare un file ese (cancellare.txt)

git restore --staged cancellare.txt

Git restore cancellare.txt

Creare repository remoto su github + git remote add

```
echo "# repository-test" >> README.md
```

```
git init
```

```
git add README.md
```

```
git commit -m "first commit"
```

```
git branch -M main
```

```
git remote add origin https://github.com/docentemaurocasadei/repository-test.git
```

```
git push -u origin main
```

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

```
git remote add origin https://github.com/docentemaurocasadei/repository-test.git
```

```
git branch -M main
```

```
git push -u origin main
```

Git remote add

```
PS C:\Corsi\git\repository-test> git remote add origin https://github.com/docentema
urocasadei/repository-test.git
>>
PS C:\Corsi\git\repository-test> git branch -M main
>>
PS C:\Corsi\git\repository-test> git push -u origin main
>>
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 12 threads
Compressing objects: 100% (17/17), done.
```

Condividere modifiche con remoto

git push origin branch -> Carica modifiche su un repository remoto

```
git push -u origin main
```

Ottenere modifiche dal remoto

git pull origin branch -> Scarica e applica le modifiche dal remoto

```
● PS C:\Corsi\git\repository-test> git pull origin main
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 972 bytes | 121.00 KiB/s, done.
From https://github.com/docentemaurocasadei/repository-test
 * branch                main                -> FETCH_HEAD
    bbbf6e7..a7cd2fd      main                -> origin/main
Updating bbbf6e7..a7cd2fd
Fast-forward
```