
Git

sotto il cofano

Fondamentale per il versionamento del codice



git

Salvatore La Spata (salvatore.la.spata@gmail.com)



Agenda

1. Basi solide
2. Il modello di storage di GIT
 - a. Come ragiona GIT
 - b. Cosa c'è dietro GIT
 - c. Cosa avviene dietro le quinte di una COMMIT

Cos'è **GIT**



Definizione

Git è un sistema di controllo versione utilizzato per gestire le modifiche apportate ai file in un progetto software.

Con Git, è possibile **tenere traccia delle modifiche** apportate ai file nel tempo, in modo che si possa facilmente tornare indietro a una versione precedente di un file se necessario.

Inoltre, Git consente a più persone di **lavorare sullo stesso progetto contemporaneamente**, facilitando la collaborazione su progetti di grandi dimensioni

Caratteristiche

- **Vedere tutte le modifiche apportate al progetto**, quando sono state apportate e chi le ha fatte.
- Ogni modifica può essere accompagnata da un **messaggio che ne spiega il motivo**.
- **Recuperare le versioni precedenti** dell'intero progetto o di singoli file.
- **Creare rami (branch)**, dove è possibile apportare modifiche in via sperimentale. Questa funzione consente di **lavorare contemporaneamente su diverse serie di modifiche** (ad esempio, funzionalità o correzioni di bug), eventualmente da parte di persone diverse, senza influenzare il ramo principale. In seguito, è possibile unire le modifiche che si desidera mantenere al ramo principale.
- **Assegnare un tag** a una versione, ad esempio per contrassegnare una nuova release.

Attori di GIT



HEAD



LOCAL



REMOTE/ORIGIN

Componenti di GIT

BRANCH



TAG





Come si installa

Per installare Git, è necessario seguire i passaggi specifici per il proprio sistema operativo. Di seguito sono riportati i passaggi generali per l'installazione su alcuni sistemi operativi comuni:

Windows

<https://git-scm.com/download/win>

MacOS

Git viene già installato di default su macOS, quindi non è necessario fare nulla. Se non avete Git installato,

Linux

Sulla maggior parte delle distribuzioni Linux, è possibile installare Git utilizzando il gestore di pacchetti del sistema.

Una volta installato Git, potete verificare che l'installazione sia andata a buon fine digitando `git --version` nel terminale.

Comandi principali

<code>git init</code>	Inizializzare un progetto git (crea la cartella .git nascosta)
<code>git clone</code>	scarica una copia di un repository Git in locale.
<code>git add</code>	aggiunge uno o più file al prossimo commit.
<code>git commit</code>	crea una nuova commit (snapshot dei file modificati)
<code>git push</code>	invia le commit locali al repository remoto.
<code>git pull</code>	scarica le commit più recenti dal repository remoto e le integra con la propria copia locale del repository.

Altri comandi

<code>git checkout</code>	Cambiare branch
<code>git status</code>	Stato attuale del repository
<code>git merge</code>	Unire due o più storie di sviluppo
<code>git rebase</code>	Riscrive la storia di un branch
<code>git stash</code>	Salva le modifiche che non sono state ancora committate in un'area temporanea in modo da poter passare a un altro branch.
<code>git cherry-pick</code>	Applica le modifiche di un commit specifico a un branch diverso da quello in cui sono state originariamente fatte.



.gitignore

Il file **.gitignore** è un file speciale utilizzato in un repository Git per specificare i **file e le cartelle** che devono essere **ignorati dal controllo delle versioni**.

Quando si esegue un'operazione come il commit delle modifiche in Git, il contenuto del file `.gitignore` viene preso in considerazione per determinare **quali file devono essere esclusi**.

Ma...

GIT e GITHUB sono la stessa cosa?!

La differenza è un pò come il porno e pornhub 😎. **GITHUB** è un hub che permette di centralizzare i sorgenti software che utilizzano GIT come sistema di versionamento del codice.

#GIT #GITHUB



Github - Pull Request

Le pull request consentono di comunicare ad altri le modifiche apportate a un ramo di un repository su GitHub. Possono essere soggette ad approvazione.

The screenshot shows the GitHub interface for a repository. The top navigation bar includes links for Code, Issues, Pull requests (which is the active tab), Actions, Projects, Security, Insights, and Settings. A search bar contains the query "is:pr is:open". Below the search bar are buttons for Labels (9) and Milestones (0), and a green "New pull request" button. The main content area displays a summary: "0 Open" and "2 Closed". It also includes filters for Author, Label, Projects, Milestones, Reviews, Assignee, and Sort. A message at the bottom states, "There aren't any open pull requests." followed by a tip: "You could search [all of GitHub](#) or try an [advanced search](#)".

Github - Issue

Utilizzate GitHub Issues per tenere traccia di idee, feedback, compiti o bug per il lavoro su GitHub.

The screenshot shows the GitHub Issues interface. At the top, there's a navigation bar with tabs for Code, Issues (which is selected and highlighted in orange), Pull requests, Actions, Projects, Security, Insights, and Settings. Below the navigation is a search bar with the query "is:issue is:open". There are also buttons for Labels (9) and Milestones (0), and a prominent green "New issue" button. Underneath the search bar are filters for Open (0) and Closed (0) issues, along with dropdown menus for Author, Label, Projects, Milestones, Assignee, and Sort. The main content area is a large white space with a small circular icon containing a dot. The text "There aren't any open issues." is centered in this space. Below this, a note says "You could search [all of GitHub](#) or try an [advanced search](#)". At the bottom, there's a "ProTip!" note: "Adding [no:label](#) will show everything without a label."

Github - Actions

Automatizzate, personalizzate ed eseguite workflow di sviluppo software direttamente nel vostro repository con GitHub Actions.

```
name: learn-github-actions
run-name: ${{ github.actor }} is learning GitHub Actions
on: [push]
jobs:
  check-bats-version:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v3
        with:
          node-version: '14'
      - run: npm install -g bats
      - run: bats -v

```

.github/workflows/lean.yml

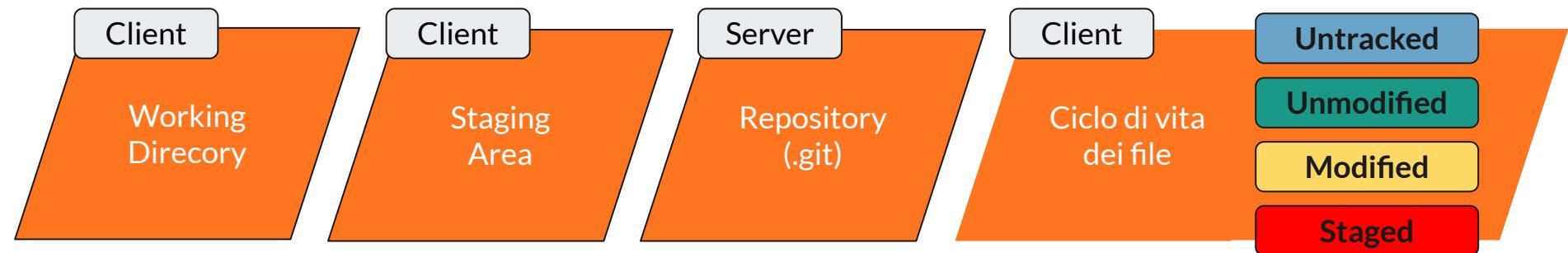
Iniziamo...
i primi passi con **GIT**

La nostra prima commit

```
> git init  
> git add <nome_file>  
> git commit -m "<messaggio_di_commit>"
```

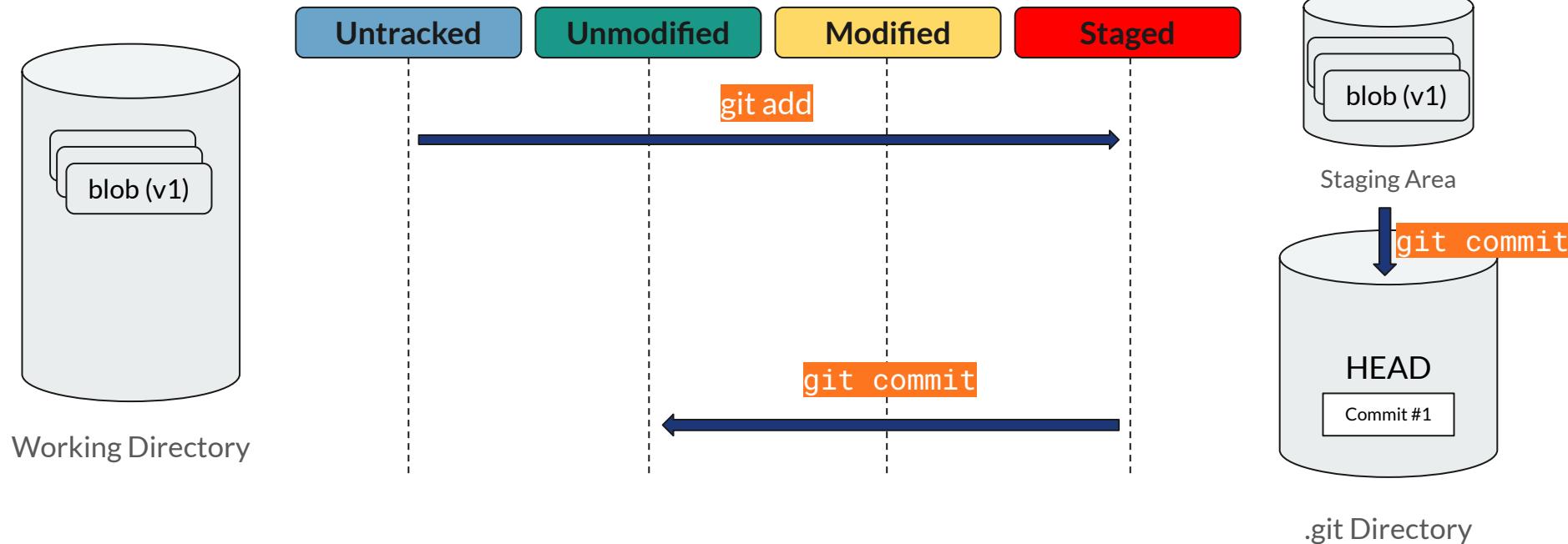
La nostra prima commit

Workflow

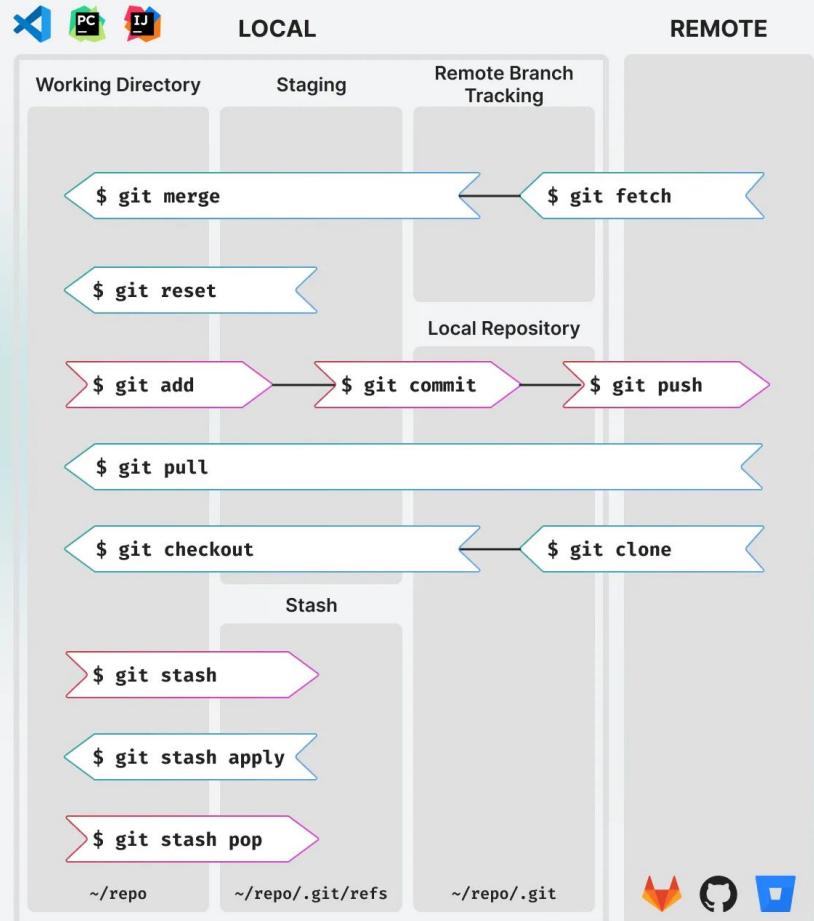


La nostra prima commit

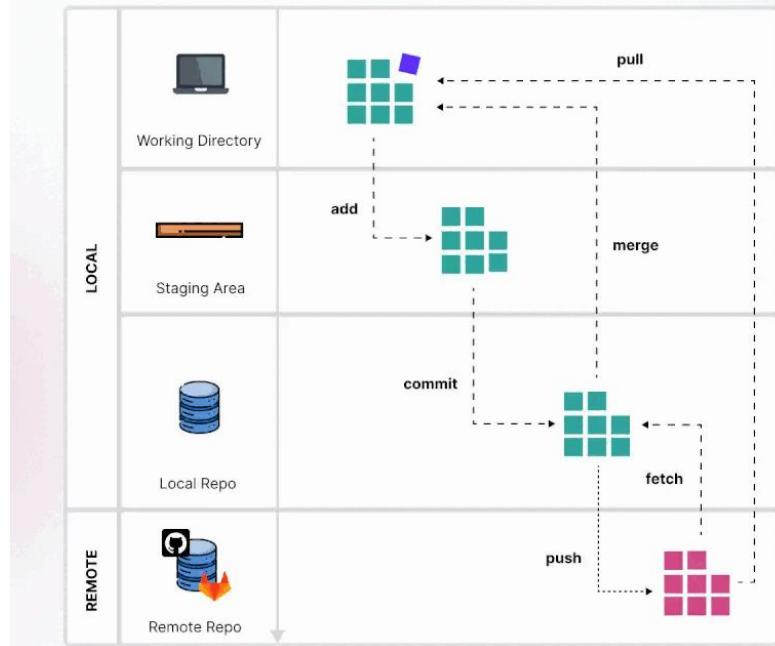
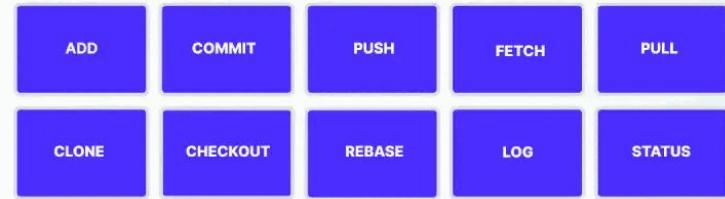
Diagram



Workflow



Workflow



Il modello di storage



Rappresenta un file versionato (compresso con zlib) nel nostro progetto

obj:blob | 2c6a
annotazione



Un albero che contiene gli oggetti blob e altri alberi. Una sorta di filesystem.

obj:tree | 8e2a
blob 2c6a filename



E' un riferimento all'albero, i relativi file, l'autore della modifica, la data e l'annotazione.

obj:commit | 1a15
tree 8e2a
author timestamp
committer timestamp
annotazione



Puntatore ad una commit o ad un altro puntatore (commit/branch/tag)

ref | HEAD
refs/heads/main

Integrità

L'integrità viene garantita grazie agli **identificativi (SHA-1)** legati al **contenuto**
Qualsiasi cosa in GIT è controllata tramite checksum

obj :blob	2c6a
obj :tree	8e2a
obj :commit	1a15

SHA-1
2c6a03aa17ddb9053f769147dff01ccb40a81ccd
(30 caratteri esadecimali)

Best Practices...
Like GIT flow









Best Practices

Mantenere le commit piccole e focalizzate

Utilizzare messaggi di commit chiari e concisi

Diramazione frequente dei branch per isolare le modifiche

L'uso delle Pull Request per le Code Reviews

Lavorare su progetti con i membri del team

Possibilità di lavorare su progetti open-source

tips & tricks



Visual Studio Code Extensions:

- Git Graph A small icon showing a simplified Git commit graph with three nodes connected by lines.
- Git Lens A circular icon featuring a stylized blue and purple Git commit graph.

Live Demo

