


Git

Ingegneria del software

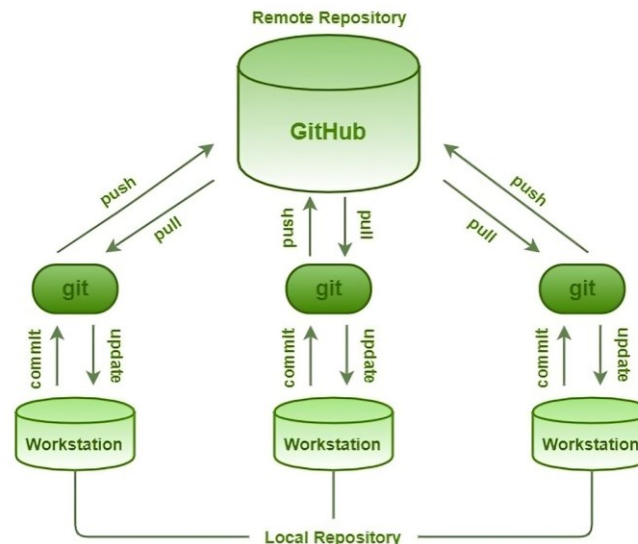
Vincenzo Bonnici
Corso di Laurea in Informatica
Dipartimento di Scienze Matematiche, Fisiche e Informatiche
Università degli Studi di Parma


2025-2026

- Git è **un software di controllo di versione**:
- si tratta di una applicazione che permette di tenere traccia di tutte le modifiche apportate al codice sorgente di un determinato programma (che si intende monitorare durante le varie fasi di sviluppo).
- Git è un programma che permette una **collaborazione decentralizzata**.
- Non è necessario un server centrale e gli sviluppatori possono collaborare tra loro in **maniera parallela**, senza essere per forza connessi in ogni istante.
- Git non è il solo **sistema di controllo di versione** (VCS) esistente, ma di certo è lo standard oggi universalmente riconosciuto in campo informatico.
- Il creatore di Git è nientepopodimeno Linus Torval, il papà di Linux. 
- Torval ha iniziato a sviluppare Git proprio come strumento di controllo per le sue personali attività di development sul kernel del sistema operativo.

Cosa è Git?

- Un **VCS** (version control system) mantiene una base di dati che conserva le modifiche effettuate sui file tracciati
 - **locale**: i file tracciati risiedono sulla propria macchina
 - **centralizzato**: i file tracciati risiedono solamente su un server centrale
 - **distribuito**: la base di dati e i file tracciati risiedono sulle macchine di ciascun utente
- GIT è un VCS decentralizzato, che tiene traccia delle modifiche ai file tramite snapshot
 - la maggior parte delle operazioni sono svolte in locale
 - ogni modifica viene salvata con un controllo di integrità
 - difficile «convincere» GIT a cancellare delle informazioni



- Lo sviluppo del software in Git viene idealizzato con il concetto di rami (**branch**).
- Esiste un ramo principale di partenza, detto **master**.
- Lo sviluppo evolve con la modifica, cancellazione o **inserimento**, di righe a partire da uno **stato iniziale**.
- Ogni registrazione di un **nuovo stato** è detta **commit**.
 - Il commit è un momento importante dello sviluppo in cui si decide di “congelare” lo stato dei lavori, commentando e lasciando traccia del lavoro svolto. 
- Gli sviluppatori possono lavorare contemporaneamente anche su rami paralleli (branch) che partono da un commit del ramo master.
- Successivamente le modifiche di un branch possono essere inglobate nuovamente sul ramo principale tramite una operazione di **merge**.

Come funziona Git?

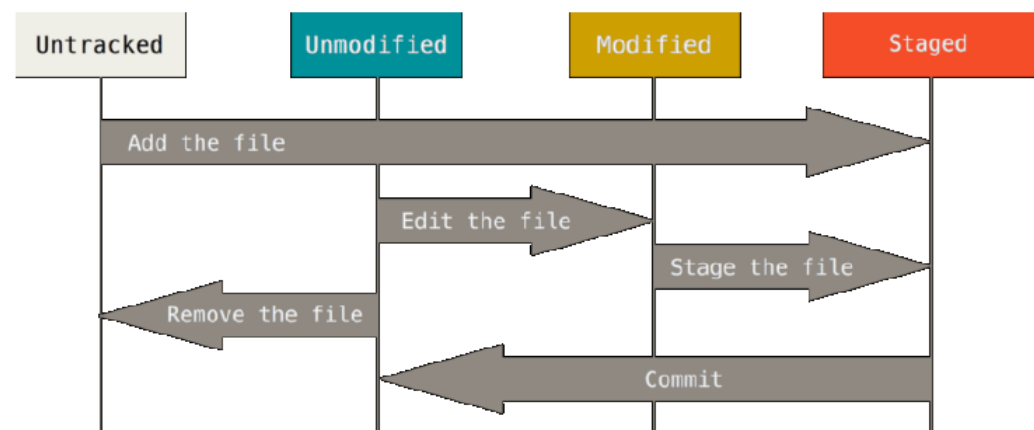
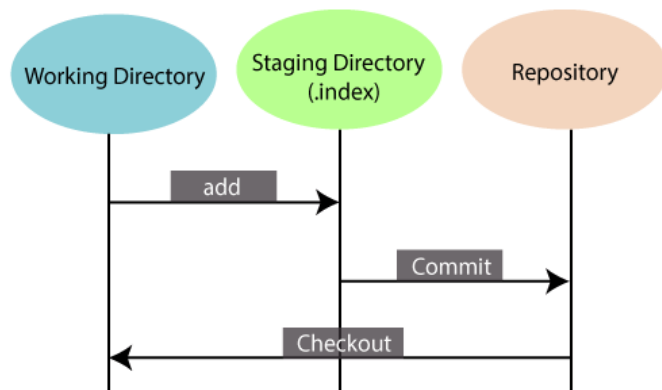
- È Git che si farà carico del compito di individuazione delle differenze tra i vari commit e branch, prima di nuove fasi di commit.
- Evidenzierà all'utente ciò che è variato, tenendo traccia di tutte le modifiche.
- Permetterà inoltre di poter tornare in qualsiasi momento allo stato di una determinata fase di commit.



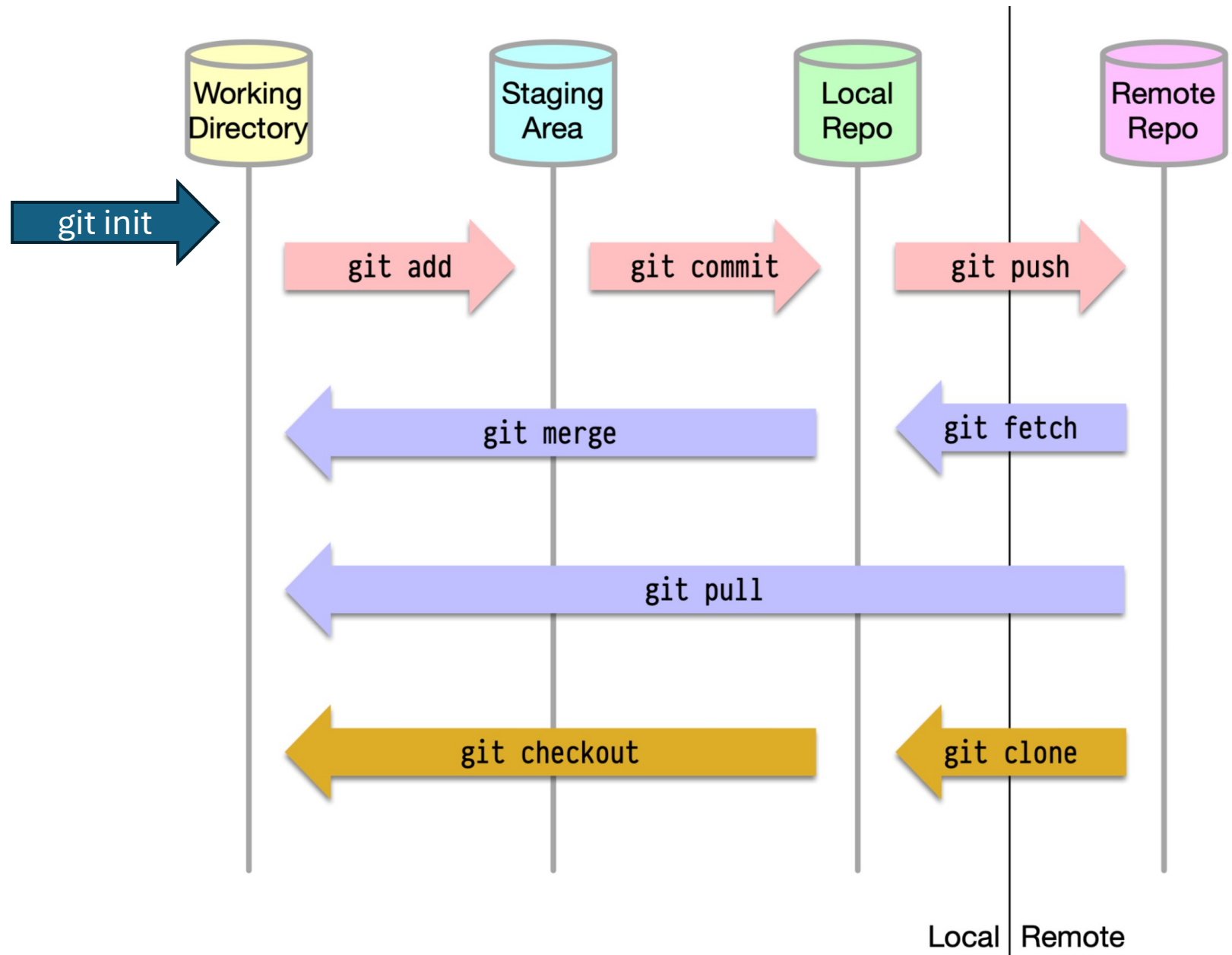
Un flusso di versioning con Git: c'è il ramo principale (master, quello in arancio) e alcune branch di sviluppo parallelo (che in alcuni casi ritornano sul ramo primario). I commit sono rappresentati dai punti.

Lo stato dei file in Git

- Possiamo immaginare Git come un guardiano che “veglia” sui file di una certa cartella-progetto.
- Lo sviluppatore dirà a Git che file guardare e lui sarà lì e osservarne le evoluzioni, i cambiamenti. 👁👁
- I file nella cartella monitorata da Git possono avere vari **stati**:
 - **Non tracciato**, Untracked: il file non è stato preso in considerazione da Git
 - **Modificato**, modified: rispetto l'ultimo salvataggio/commit il file è cambiato
 - **Staged**: il file è marcato per essere salvato sul prossimo commit. Solo i file in **staging area** verranno salvati sul commit!
 - **Committed**: il file, e una sua copia, sono stati congelati in quel preciso istante sul database di Git.

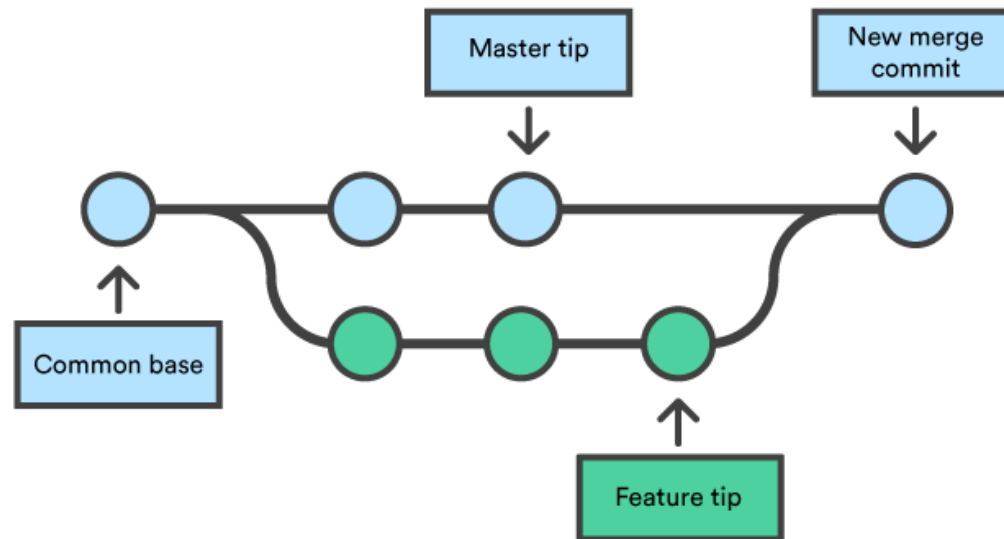


Lo stato dei file in Git



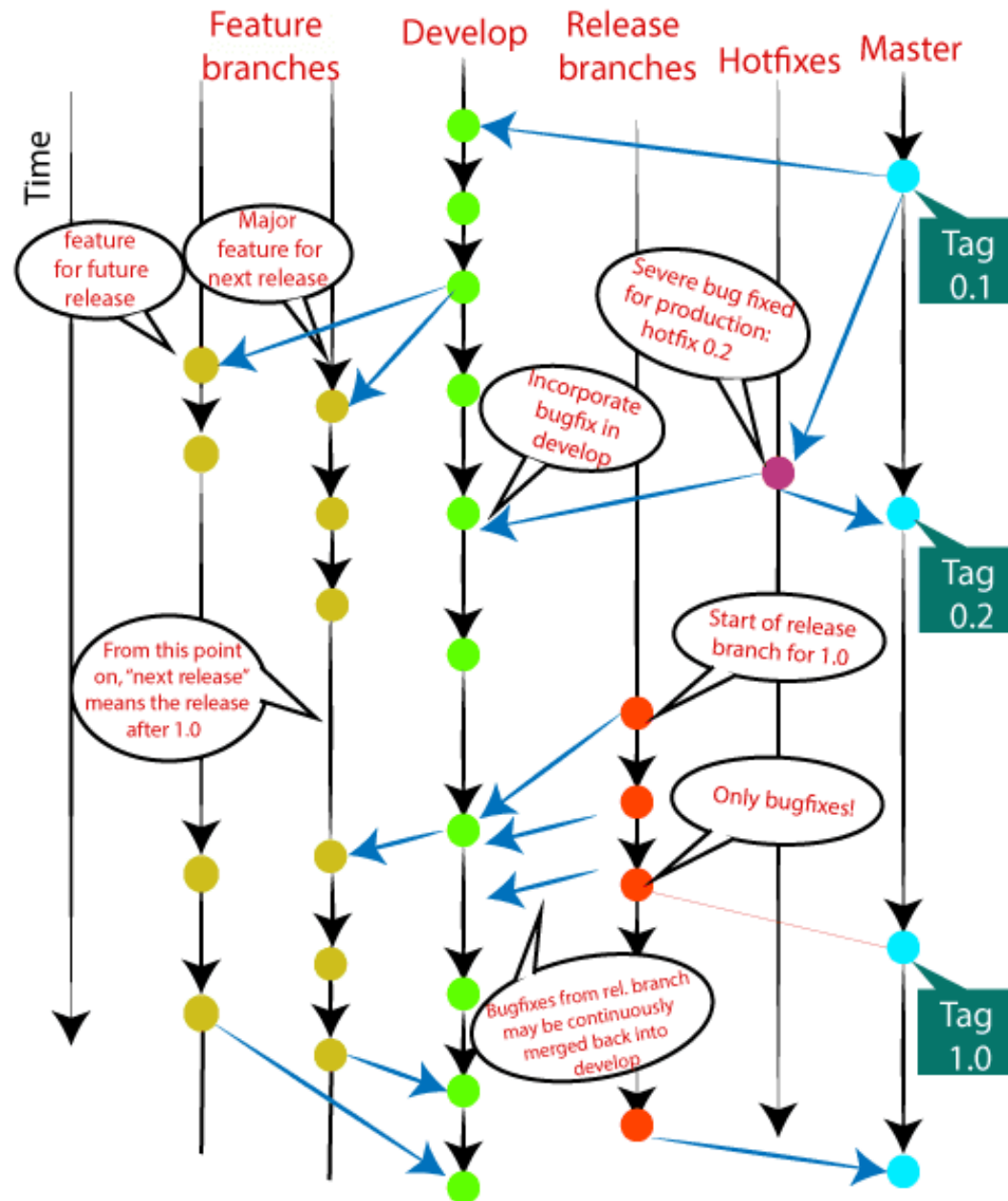
Unire due rami

- Supponiamo che la fantastica idea di implementazione del tuo software sia valida.
- La nuova feature è piaciuta in giro ed è da ritenersi ora matura.
- È quindi giusto che lo sviluppo non sia più considerato solo una fase parallela di testing (il branch idea), ma venga assorbito sul progetto principale, ovvero il ramo master.
- Cosa fare?
 - Quella da effettuare è una operazione di merge, unione di due rami.



- Attenzione, però. In questa fase ci vorrà però un po' più di attenzione da parte tua.
- Quando chiederai a Git di unire due rami il software si comporterà nella maniera più intelligente possibile, ma la scelta di cosa scartare e cosa tenere (nel confronto tra le righe delle due versioni) spetterà a te.
- Per fare merge del branch idea sul branch master dobbiamo in ordine:
 - Spostarci sul ramo finale sul quale finirà il progetto, ovvero master:
 - `git checkout master`
 - Chiedere a Git di effettuare l'unione con il branch idea:
 - `git merge idea`
 - Gestire eventuali conflitti e chiudere il merge con un nuovo commit
- Relativamente il punto 1: teoricamente potremmo voler chiudere il branch master facendolo convogliare su quello idea. È infatti solo regola comune, ma non obbligatoria, che master sia il branch principale su quale sviluppare il progetto operativo nel tempo.
- Relativamente il punto 3: la gestione dei conflitti è l'aspetto per il quale è necessaria la tua attenzione.

Un esempio di branching



- Come già detto `log` è il comando da utilizzare per visualizzare cosa è successo nella storia del nostro (o altrui) sviluppo.
 - `git log --graph --all --oneline`
- È il comando che ti consiglio per visualizzare in modo sintetico ma completo l'evoluzione di tutti i rami del tuo progetto.
- Se vuoi affidarti a un tool grafico:
 - `gitk --all`
 - (ma prima installa `gitk`, su Ubuntu `apt get install gitk`)