Università degli studi di Perugia
Dipartimento di Ingegneria
Corso di Laurea Triennale in Ingegneria Informatica ed Elettronica

IL SISTEMA DI CONTROLLO DI VERSIONE DISTRIBUITO GIT

Candidato:
Domenico Pepino

Relatore:
Luca Grilli

Questa tesi si è proposta di valutare:

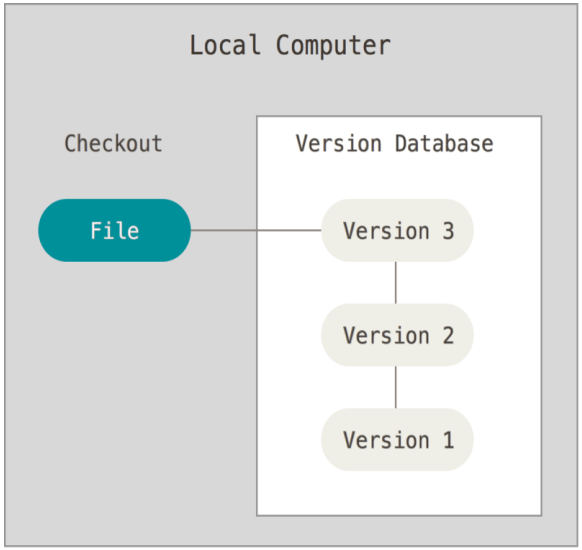
-
-

A tale scopo:

-
-

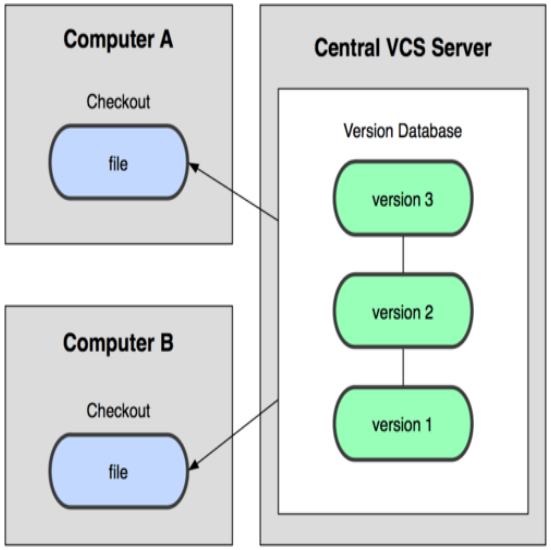
Sistemi di Controllo di Versione

Locale



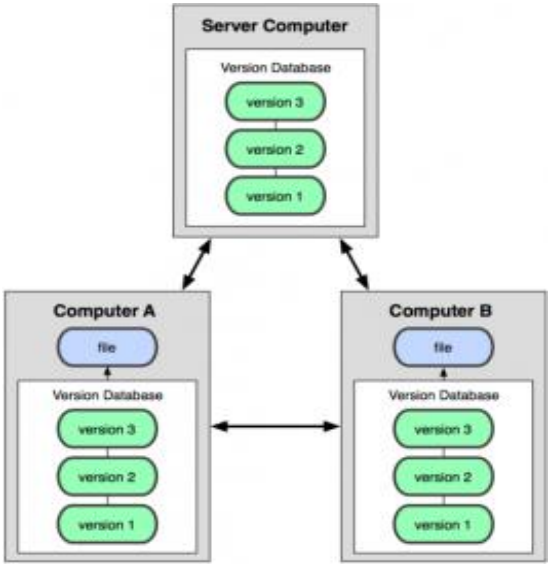
- Database semplice
- Informazioni Locali

Centralizzato



- Permette la collaborazione tra sviluppatori
- Informazioni sul server
- Miglior protocolli: CVS, SUBVERSION

Distribuito



- I client controllano una copia completa della repository
- Evita la problematica della rottura del server

Il Sistema di Controllo di Versione Distribuito GIT

Nasce nel 2005, dopo la revoca dell'uso gratuito di BitKeeper, con i seguenti obiettivi prestazionali:

- Velocità
- Design Semplice
- Ottimo supporto allo sviluppo non-lineare
- Gestione del controllo di versione totalmente distribuito
- Capacitò di gestire grandi progetti

È possibile installarlo su qualsiasi sistema operativo, tramite pannello di controllo o interfaccia grafica, ad esempio per i sistemi linux:

```
$ yum install curl-devel expat-deval gettext-devel \ openssl-devel zlib-devel
```

Gestione del Controllo

Ogni volta che si opera una modifica in un file o un salvataggio dello stato, Git fa un'immagine di tutti i file modificati in quel momento, creando una serie di istantanee.

Controlla tramite checksum ogni operazione.

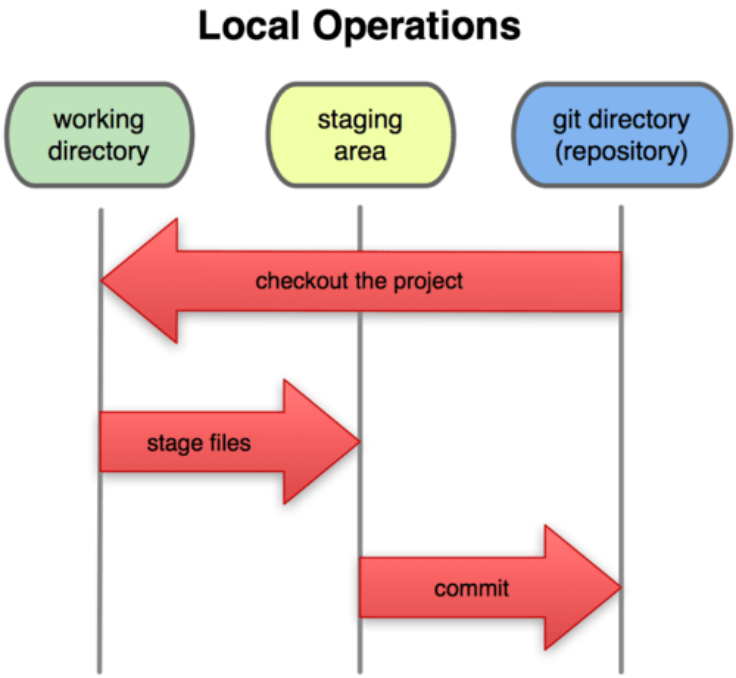
Utilizza il checksum denominato SHA-1.

SHA-1 è una funzione di hash crittografica che accetta un input e produce un valore di hash a 20 byte, denominato come digest del messaggio.

Divisione per stati dei file e del piano di lavoro

All’interno di Git i file possono entrare in tre stati
- Committed
- Modified
- In Staged

Questi tre strati dividono il piano di lavoro in:
- Directory di Git
- Working directory
- Area di stage



Comandi per la creazione e gestione di un progetto

- git clone, per importare una repository esistente
- git init, per creare una repository
- git add, per poter aggiungere un file in area di stage o nella directory di git
- git status, per conoscere lo stato dei file (tracciato o non tracciato e unmodified o modified)
- git diff, per conoscere le modifche di un file presente in due aree diverse del progetto
- git commit, per ufficializzare un file nel progetto
- git checkout -- , per eliminare tutte le modifiche
- git rm, per rimuovere un file.
- git mv nomefile nomenuovofile, per rinominare un file (Git non traccia questa operazione)
- git log, per controllare la cronologia delle commit

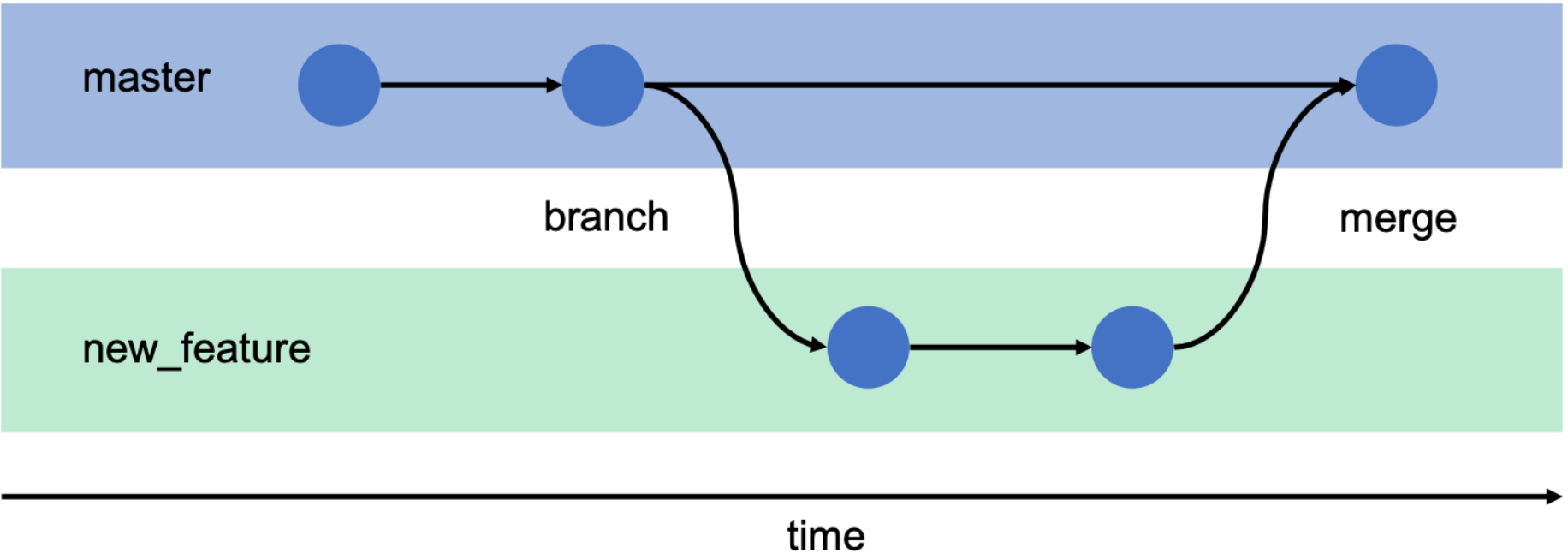
Etichette

Permettono di contrassegnare i file e mantenere una migliore gestione della cronologia.
Esistono due tipi di etichette:

- Semplici, semplice riferimento ad un commit specifico
- Annotate, oggetti complessi , ne viene calcolato il checksum, possono essere firmate con GPG

Comandi utili:
- git tag, per creare una etichetta (-a per creare le etichette Annotate)
- git show per ottenere tutte le informazioni aggiuntive (nome, email e data del creatore)
- git tag, per creare un elenco di tutte le etichette create

Branching



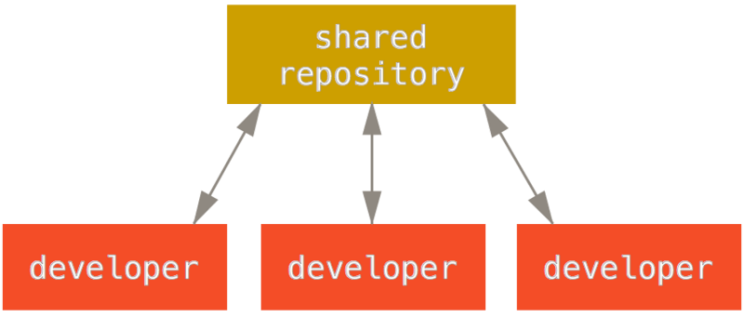
Branching : Comandi

- git branch, permette la creazione dei rami o di elencare tutti i rami creati
- git checkout, seleziona il ramo indicato
- git merge, per unire un ramo al progetto originale
- git branch – merged, per visualizzare tutti i rami uniti
- git branch –no-merged, per visualizzare solo i rami non uniti
- git rebase, unisce un ramo al progetto originale

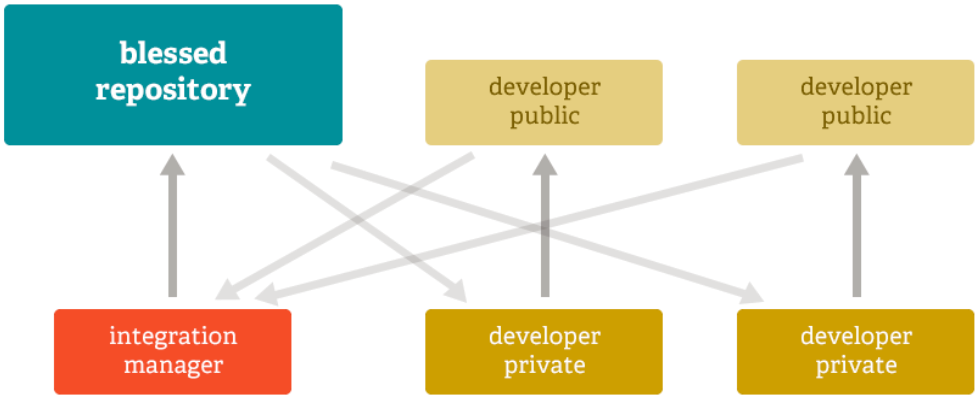
Differenza tra rebasing e merging

Effettuando l'operazione di rebasing, il ramo viene aggiunto alla fine del progetto originale.
Non ho sovrapposizione di rami
Cronologia più comprensibile

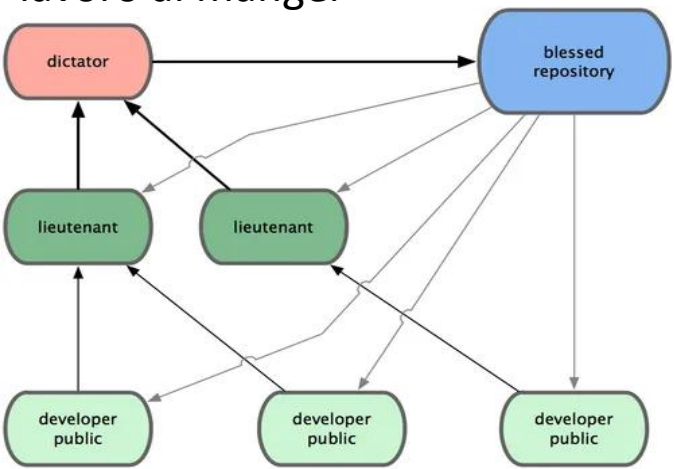
Workflows



Workflows centralizzato
• Utilizzato per progetti piccoli con pochi sviluppatori
• Tutti gli sviluppatori posso accedere e modificare il progetto originale



Workflows con un manager
• Il manager è l'unico che può effettuare modifiche sul progetto universare
• gli sviluppatori inviano il proprio lavoro al manger



Workflows con un dittatore e tenenti:
• Utilizzata su grandi progetti
• I tenenti sono responsabili solo di alcuni sottoteam
• il dittatore è l'unico che può effettuare modifiche sul progetto originale

Protocolli per la condivisione di un progetto

Un aspetto importante per la condivisione di un progetto è il tipo di protocollo scelto per comunicare con la repository remota.

In Git esistono quattro principale protocolli:

- Locale
- Secure Shell
- Git
- HTPP

GitHub

- Offre la possibilità di utilizzare sia Git che Subversioni
- Consente di creare in modo illimitato le repository pubbliche, pagamento mensile per le private
- Permette di creare team che funzionano come account normali
- Permette di configurare velocemente la propria repository con i protocolli di HTTPS e SHH
- Permettte l'accesso in solo lettura tramite il link di condivisione

Servizi di hosting alternativi a GitHub

	GitHub	Bitbucket	SourceForge	GitLab
Costo	Gratuito	Gratuito	Gratuito	Gratuito
Repository private	A pagamento	Illimitate e gratuite	Illimitate e gratuite	Illimitate e gratuite
Repository pubbliche	Illimitate e gratuite	Illimitate e gratuite	Illimitate e gratuite	Illimitate e gratutite
Limite di archiviazione	1 GB per repository	2 GB per repository	Nessuno	Nessuno
Utenti	Illimitati	5 per gli account gratuiti	Non permette la collaborazione	Illimitati
Tipi di sistemi di controllo di versione	Git SVN	Git, HG	Git , SVN, HG	Nessuno
Wiki	Disponibile	Disponibile	Disponibile	Disponibile
Revisione del codice	Disponibile	Disponibile	Disponibile	Disponibile

Conclusioni