git

Git è un sistema di gestione delle revisioni creato da Linus Torvalds per ovviare alle carenze degli RCV (Revision Control System) liberi allora esistenti (CVS, SVN, Monotone)

Indice

Caratteristiche

Distribuito

Git fa parte della moderna famiglia dei sistemi di controllo delle revisioni distribuiti, ovvero non prevede un server centralizzato al quale gli sviluppatori devono avere accesso, ma fornisce un repository completo ad ogni sviluppatore, in modo che possa effettuare localmente qualunque operazione, offrendo poi strumenti appositi per la raccolta dei contributi e la gestione unitaria del progetto.

Sviluppo non lineare

Git favorisce un modello di sviluppo non lineare nel quale le modifiche non sono in un semplice ordine cronologico, ma si suddividono in più rami intrecciati tra di loro, in modo da favorire la collaborazione tra più persone e la libera sperimentazione. Per farlo permette la creazione economica di branch e fornisce degli ottimi strumenti per la loro gestione con merge efficaci e cherry picking.

Veloce

Git è stato sviluppato con l'obiettivo esplicito di raggiungere buone velocità di funzionamento.

Scalabile

Git è abbastanza semplice da essere adatto già per progetti di dimensioni minime, ma è stato creato per essere adatto allo sviluppo del kernel Linux e non ha quindi problemi ad adattarsi a complessità e dimensioni di tutto riguardo.

Tradizione Unix

L'interfaccia di git segue la tradizione unix dell'uso di numerosi piccoli programmi, ciascuno dedicato ad un solo compito.

Vantaggi

Rispetto a CVS/SVN

Un DVCS come git permette di lavorare in modo molto meno rigido rispetto ad un VCS centralizzato: in particolare rende semplici e naturali operazioni che con un VCS centralizzato sarebbero problematici.

- È molto facile e veloce usare branch, anche solo locali, per sperimentare nuove capacità del programma, senza bisogno di lasciare traccia dei fallimenti sul repository più pubblico del progetto.
- È possibile lavorare anche in assenza di connessione ad internet, dato che si ha una copia completa del repository locale.
- Clonare un repository è estremamente facile, ad esempio per effettuarne un backup o per salvare un progetto abbandonato dall'autore originale.

Rispetto ad altri DVCS

Le principali alternative a git sono gli altri DCVS oggi esistenti; vale la pena di segnalarne alcuni in particolare degni di nota.

Monotone

Monotone è un DCVS preesistente a git che ne implementava gia' buona parte delle funzionalità: non è stato scelto per lo sviluppo del kernel a causa del suo più grosso difetto, l'estrema lentezza. È ora in un periodo di calo , soppiantato dai DCVS piu' recenti.

Bitkeeper

Bitkeeper è il DCVS usato per un breve periodo per lo sviluppo del kernel; il suo principale difetto è il non essere software libero. Molte delle sue caratteristiche sono state implementate in git.

Mercurial e Bazaar

Le caratteristiche fondamentali di questi DCVS sono fondamentalmente comparabili a quelle di git, quello che fa la diferenza sono la comodità d'uso, anche rispetto alle proprie abitudini, e la diffusione nei vari progetti ai quali si può essere interessati a partecipare. In particolare bazaar e' usato da launchpad, e quindi da molti progetti correlati ad ubuntu.

Adozione

Git sta riscuotendo un discreto successo tra gli sviluppatori di software libero; oltre ad essere usato per il suo stesso sviluppo ed ovviamente per il kernel Linux, e' stato adottato da numerosi progetti, tra cui alcuni degni di nota per dimensioni o importanza.

- Perl
- Android
- WINE
- Fedora
- X.org
- GNOME
- OpenEmbedded

Funzionamento elementare

Struttura

Essendo git distribuito, ogni copia di un progetto è un repository completo, nel quale sono salvate tutte le informazioni relative ai contenuti e alla loro storia. In questo repository sono salvate delle immagini dello stato del progetto in vari momenti della sua storia, chiamati commit e correlati tra di loro da rapporti di parentela non lineare, ma organizzata su linee di sviluppo, dette branch, che possono procedere parallelamente, unirsi o divergere.

Ogni commit è univocamente identificato in ogni repository che lo contenga da un hash di 40 cifre esadecimali, calcolato sui contenuti del commit stesso. I vari branch, dotati di un nome, sono rappresentati ciascuno da un'head, che punta all'ultimo commit di quel branch. Un ulteriore riferimento a particolari commit può essere dato dai tag, usati per riferirsi ad una versione specifica del progetto.

La maggior parte dei repository contengono poi una working directory, una copia della versione corrente dei file del progetto; inoltre git mantiene un index, dove sono salvate tutte le modifiche che andranno a comporre il prossimo commit.

Più repository contenenti lo stesso progetto sono generalmente in una relazione paritaria per la quale ciascuno può scambiare commit, branch e tag con gli altri, tramite operazioni di push e pull. In pratica, si usa in realtà una struttura parzialmente gerarchica nella quale ogni sviluppatore ha uno o più repository per uso personale dal quale invia periodicamente (push) le modifiche ad un repository "bare", ovvero privo di working copy" ed accessibile in lettura a tutti gli altri sviluppatori ed eventualmente al pubblico, che possono scaricare (pull) tali modifiche nei loro repository personali. Alcuni sviluppatori centrali avranno poi il ruolo di integratori e si occuperanno di scaricare le modifiche dai repository pubblici dei vari sviluppatori e pubblicarle in un repository considerato autorevole, dal quale generalmente verranno prese le release.

Git permette comunque, pur senza incoraggiarla, una gestione più tradizionale nella quale gli sviluppatori hanno accesso diretto al repository centrale, al quale possono inviare i commit direttamente dai loro repository privati di uso personale.

Primi comandi

Configurazione iniziale

Per dichiarare la propria identità a git, perché possa attribuire correttamente i successivi commit:

```
git config --global user.name "Nome Cognome"
git config --global user.email "nome.cognome@example.org"
```

Creazione del repository

Per creare la copia locale di un repository pubblicato online o disponibile sul disco locale, rispettivamente:

```
git clone git://git.kernel.org/pub/scm/git/git.git
git clone /home/$COLLEGA/$REPOSITORY ~/repo/$REPOSITORY
```

Per creare invece un nuovo repository nella directory `$MY_REPOSITORY`:

```
cd $MY_REPOSITORY
git init
```

Branch

Per visualizzare i branch attualmente presenti sul repository, rispettivamente locali, remoti o tutti:

```
git branch
git branch -r
git branch -a
```

Per passare ad un branch già esistente:

```
git checkout $BRANCH
```

Per creare un branch e cominciare a lavorarci:

```
git checkout -b $BRANCH
```

Modifiche dei contenuti e commit

Dopo aver modificato alcuni file si può aggiungere il loro stato corrente all'index con il comando:

```
git add $FILES
```

contrariamente ad altri VCS, questo non fa sì che ulteriori modifiche a quel file vengano automaticamente inserite nei commit futuri.

I contenuti dell'index possono poi essere trasformati in un commit:

```
git commit
```

Link utili

Git - Fast Version Control System Il sito principale di Git.

Git User's Manual Manuale completo di Git.

Why Git is better than X Una descrizione del perché Git è meglio di altri VCS.

Mercurial DCVS
Bazaar DCVS

Questo documento

Authors: Elena “of Valhalla” Grandi <valhalla@lifolab.org>

Version: 2009-02-16