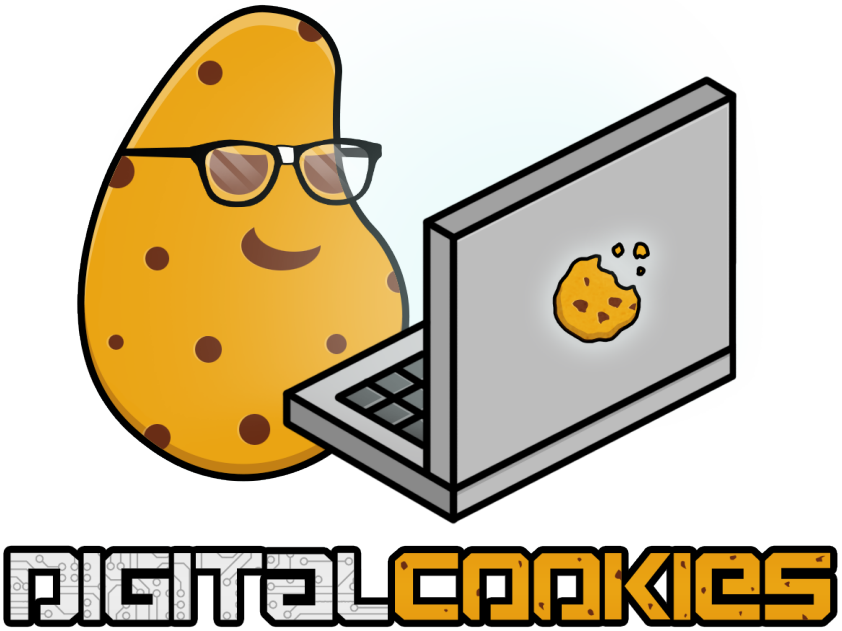

Guida ai comandi Git

Gruppo DigitalCookies — ***Progetto SWEDesigner***

digitalcookies.group@gmail.com

Informazioni sul documento

Versione	1.0.0
Redazione	Christian Cabrera
Verifica	Saverio Follador
Approvazione	Davide Albertini
Uso	Interno
Distribuzione	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
	Gruppo DigitalCookies

Descrizione

Questo documento contiene elenco e relativa spiegazione dei comandi che sarà necessario conoscere per l'utilizzo di Git in modo conforme.

Registro delle modifiche

Versione	Data	Collaboratori	Ruolo	Descrizione
1.0.0	23-03-2017	Davide Albertini	Responsabile	Approvazione
0.1.0	22-03-2017	Saverio Follador	Verificatore	Verifica del documento
0.0.1	22-02-2017	Christian Cabrera	Amministratore	Stesura documento

Indice

1 Introduzione

1.1 Scopo del documento

Questo documento ha lo scopo di indicare e spiegare quali sono i comandi di *Git*$_G$ a cui il documento *Norme di Progetto v1.0.0* fa riferimento. Il presente documento non è fra quelli obbligatori e la sua destinazione è solamente interna. Esso è volto a formare in maniera uniforme i componenti del *team di sviluppo*$_G$ circa lo strumento di supporto alla versione adottato. La struttura del documento è organizzata in un elenco di comandi.

1.2 Ambiguità

Al fine di evitare ogni ambiguità relativa al linguaggio impiegato nei documenti viene fornito il *Glossario v1.0.0*, contenente la definizione dei termini in corsivo marcati con una G pedice.

1.3 Riferimenti

1.3.1 Normativi

- **Norme di Progetto**: *Norme di Progetto v1.0.0*.

1.3.2 Informativi

- **Documentazione Git** : `https://git-scm.com/docs`.

2 Elenco comandi

I comandi sono da utilizzare in una *shell*$_G$. In *Windows*$_G$ si può emulare la shell di accesso a Git installando Git per Windows ed eseguendo *Git Bash*$_G$. Per indicare percorsi sarà necessario usare il simbolo `/`. Si devono sempre utilizzare percorsi relativi alla directory corrente. Di seguito si fornisce l'elenco dei comandi con la relativa spiegazione:

- `git init`: utilizzato per creare un nuovo *repository*$_G$ git vuoto nella cartella in cui viene dato il comando;
- `git config --global user.name username`: utilizzato per settare il nome che Git assocerà come autore di ogni commit;
- `git config --global user.email email`: utilizzato per settare l'email che Git assocerà come autore di ogni commit;
- `git clone indirizzo`: utilizzato per clonare, nella directory corrente, il repository corrispondente all'indirizzo fornito;
- `git add percorso/file`: utilizzato per aggiungere al tracking di Git un file nuovo o modificato da integrare al repository. Per aggiungere tutti i file contenuti in locale bisogna eseguire il comando aggiungendo `--all`;
- `git status`: utilizzato per visualizzare i file creati o modificati dall'ultimo `git pull` eseguito. Segnala anche i file che non sono ancora stati aggiunti al tracking di Git;
- `git diff percorso/file`: utilizzato per visualizzare in maniera specifica le modifiche locali del file. Togliendo `percorso/file` verranno visualizzate tutte le modifiche locali;
- `git pull`: utilizzato per aggiornare tutti i file locali all'ultima versione disponibile. Eventuali nuovi file verranno anch'essi scaricati;
- `git commit -m "motivo del commit"`: utilizzato per confermare le modifiche apportate al file locale. Il file deve essere stato aggiunto al tracking di Git con il comando `git add`;
- `git push`: esegue il *merge*$_G$ fra il repository locale e quello remoto. Invia i file confermati precedentemente tramite il comando `git commit`;
- `git log`: utilizzato per visualizzare la storia dei commit del repository;
- `git checkout`: utilizzato per saltare da un *branch*$_G$ ad un altro. Utilizzando il comando `git checkout <nomeDelBranch>` passa al ramo <nomeDelBranch>;
- `git checkout percorso/file`: utilizzato riportare il file alla versione dell'ultimo commit, scartando le modifiche successive. Utilizzando `.` al posto di `percorso/file` si annulleranno tutte le modifiche locali;
- `git show`: utilizzato per visualizzare diversi tipi di oggetti;

- `git help`: utilizzato per visualizzare il manuale;
- `git checkout -b nomeDelBranch`: utilizzato per creare un nuovo branch denominato <nomeDelBranch>. I successivi comandi (status,commit,ect) vengono riferiti a questo branch;
- `git checkout master`: utilizzato per tornare al ramo principale, chiamato $master_G$;
- `git merge`: utilizzato per fondere un ramo secondario con il principale. Occorre portarsi su master ed eseguire *git merge nomedelBranch*;
- `git branch`: utilizzato per visualizzare tutti i rami locali;
- `git branch nuovo_ramo`: utilizzato per creare un nuovo ramo;
- `git branch -d`: utilizzato per eliminare un ramo dal nostro repository. Tramite i comandi `git branch -d <nome_ramo_locale>` e `git branch -d -r <nome_ramo_remoto>` è possibile eliminare rami sia locali che remoti. Eliminando un ramo remoto, però, viene eliminato solo il riferimento ad esse nel repository locale e NON il repository remoto. Questo significa chce al prossimo pull e/o $fetch_G$ tali rami verrano ricreati;
- `git fetch`: utilizzato per recuperare dati da un repository remoto appena aggiunto o per recuperarne gli aggiornamenti senza unirlo ad un ramo locale.