

IronWorks

Utility per la Costruzione di
Software Robusto



Swear on Code

Guida Git

Versione	1.0.0
Redattori	Mirko Gibin
Verificatori	Anna Poletti
Responsabili	Francesco Sacchetto
Uso	Interno
Distribuzione	Gruppo Swear on Code Prof. Tullio Vardanega Prof. Riccardo Cardin

Descrizione

Guida dei comandi principali di Git Swear on Code.

Registro delle modifiche

Descrizione	Autori	Ruolo	Data	Versione
Approvazione	Francesco Sacchetto	Responsabile	2018/04/07	1.0.0
Verifica	Anna Poletti	Verificatore	2018/03/07	0.1.0
Aggiunta sezione "Branching"	Mirko Gibin	Amministratore	2018/03/06	0.0.3
Stesura del documento	Mirko Gibin	Amministratore	2018/03/05	0.0.2
Creazione del documento	Mirko Gibin	Amministratore	2018/03/05	0.0.1



Indice

1	Introduzione	1
1.1	Scopo del documento	1
1.2	Ambiguità	1
1.3	Riferimenti	1
1.3.1	Normativi	1
1.3.2	Informativi	1
2	Istruzioni	2
2.1	Inizializzazione	2
2.2	Modifiche ai <i>file_G</i>	2
2.3	Branching	3
2.4	Sostituire i cambiamenti locali	4
2.5	Lista comandi	4

1 Introduzione

1.1 Scopo del documento

Lo strumento di versionamento utilizzato dal gruppo è *Git_G*, che viene usato assieme a *GitHub_G*. Questo documento si pone l'obiettivo di raccogliere i comandi da usare e delle regole generali da seguire per permettere un uso uniforme dello strumento, secondo quanto stabilito nel documento *Norme di Progetto v1.0.0*.

1.2 Ambiguità

Al fine di dipanare qualsiasi dubbio o ambiguità relativa al linguaggio impiegato nel documento viene fornito il *Glossario v1.0.0*, documento contenente la definizione di tutti i termini scritti in corsivo e marcati con una 'G' pedice.

1.3 Riferimenti

1.3.1 Normativi

- *Norme di Progetto v1.0.0*.

1.3.2 Informativi

- *Glossario v1.0.0*;
- *Git Book*:
<https://git-scm.com/book/it/v1>

2 Istruzioni

2.1 Inizializzazione

Per poter lavorare sul *repository_G* Iron Works presente su *GitHub_G* è necessario effettuarne una copia in locale.

A questo scopo bisogna installare preventivamente sul proprio dispositivo lo strumento di versionamento *Git_G*, seguendo le istruzioni relative al proprio sistema operativo presenti nel *Git_G* Book.

Successivamente:

- aprire il terminale (Linux) o la *Git Bash_G* (Windows);
- inserire il proprio username con `git config --global user.name username-Personale`;
- inserire la propria email con `git config --global user.email email-Personale`;
- spostarsi con il comando `cd percorso/cartella` nella cartella in cui si desidera posizionare il *repository_G*;
- digitare `git clone indirizzo-url-repository`.

In questo modo il *repository_G* viene copiato in locale e, dopo essersi spostati all'interno della nuova cartella creata (da terminale `cd cartella-creata`), sarà possibile, tramite appositi comandi, scaricare e caricare direttamente gli aggiornamenti.

2.2 Modifiche ai *file_G*

Quando si effettua una modifica ad un *file_G* è necessario validare l'operazione tramite un apposito messaggio, il commit. Per fare ciò da terminale digitare:

- `git add .` se si vogliono aggiungere al tracking di git tutti i *file_G*;
- `git add percorso/file` se si vuole aggiungere solo uno specifico *file_G*.

e a seguire

```
git commit -m "commento alla modifica".
```

Se si aggiungono tutti i *file_G*, basta un commit solo. Se si aggiunge un *file_G* per volta va inserito un commit dopo ogni add.

Successivamente tramite

```
git push
```

si inviano le modifiche al *repository_G* remoto.

Git_G impedisce di eseguire l'operazione di push se il proprio *repository_G* locale non è aggiornato all'ultimo commit del *repository_G* remoto, e questa operazione è possibile tramite il comando

```
git pull
```

che permette di scaricare l'ultima versione di ogni *file_G*.

Git_G impedisce di eseguire l'operazione di pull se prima non si aggiungono al tracking tutti i *file_G* e non si effettuano tutti i commit necessari.

2.3 Branching

L'operazione di branching consiste nel creare un nuovo ramo del *repository_G* dove poter sviluppare funzionalità in modo isolato, senza modificare il *branch_G* principale, chiamato master. Una volta terminato il lavoro si può incorporare il nuovo *branch_G* nel master.

Tramite il comando

```
git branch
```

vengono elencati i *branch_G* esistenti e viene evidenziato quello in cui ci si trova al momento. Tramite il comando

```
git checkout -b nome-branch
```

si crea un nuovo *branch_G* locale a partire da quello in cui ci si trova. Le modifiche effettuate in questo nuovo *branch_G* non avranno side-effect sul *branch_G* di partenza.

Il comando

```
git push origin nome-branch
```

renderà il *branch_G* disponibile nel *repository_G* remoto.

Per unire le modifiche apportate al *branch_G* "A" nel *branch_G* "B" è necessario posizionarsi sul *branch_G* "B" con

```
git checkout B
```

ed eseguire

```
git merge A
```

permettendo così di aggiornare il *branch_G* B solo una volta che si è valutata la correttezza delle modifiche effettuate in A.

Tramite

```
git diff branch-sorgente branch-target
```

è possibile visualizzare l'anteprima delle modifiche prima di effettuare il merge.

Per cancellare un *branch_G* locale è sufficiente eseguire

```
git branch -d nome-branch
```

oppure

```
git branch -d -r nome-branch-remoto
```

se è un *branch_G* remoto. In questo modo verrà cancellato solamente il riferimento locale al *branch_G* remoto.

2.4 Sostituire i cambiamenti locali

Tramite

```
git checkout -- nome-file
```

è possibile sovrascrivere il *file_G* locale indicato con la versione più aggiornata presente nel *repository_G* remoto, annullando così eventuali modifiche.

Per recuperare l'ultima versione dal *server_G* dell'intero progetto usare il comando

```
git fetch origin git reset -hard origin/master
```

che sovrascriverà tutte le modifiche locali effettuate all'intero progetto.

2.5 Lista comandi

- `git clone indirizzo-url-repository`: permette di clonare il *repository_G* remoto, contenuto nell'indirizzo URL indicato, nella cartella in cui viene digitato il comando;
- `git init`: permette di inizializzare un *repository_G* locale vuoto nella cartella in cui viene digitato il comando;
- `git config --global user.name username`: permette di impostare il proprio nome *utente_G*, a cui verrà associata ogni operazione effettuata;
- `git config --global user.email email`: permette di inserire la propria email;
- `git pull`: aggiorna il *repository_G* locale con le ultime modifiche effettuate e caricate online dagli altri collaboratori;
- `git add percorso/file`: permette di aggiungere al tracking di *Git_G* nuovi *file_G*, o *file_G* modificati, che dovranno aggiornare il *repository_G*; per aggiungere tutti i *file_G* digitare `--all`, oppure `."` (un punto), al posto di *percorso/file*;
- `git commit -m "messaggio del commit"`: una volta aggiungi i *file_G* nuovi o modificati al tracking, è necessario commentare con un messaggio chiaro le operazioni che sono state effettuate;
- `git status`: visualizza le modifiche avvenute dall'ultimo pull e i *file_G* che devono essere aggiunti al tracking o per i quali manca ancora un commit;
- `git push`: una volta effettuati i commit, questo comando permette di aggiornare il *repository_G* online con le proprie modifiche;

- `git checkout nome-del-branch`: si passa dal *branch_G* in cui ci si trova al *branch_G* indicato;
- `git checkout -- percorso/file`: permette di annullare le modifiche apportate ad un *file_G* locale;
- `git checkout -b nome-del-branch`: si crea il nuovo *branch_G* locale con il nome indicato e si passa a lavorare su questo;
- `git checkout master`: si passa al *branch_G* principale;
- `git merge nome-branch`: permette di unire al *branch_G* in cui mi trovo le modifiche effettuate nel *branch_G* indicato;
- `git branch`: vengono elencati i *branch_G* disponibili, e viene indicato quello in cui ci si trova al momento;
- `git branch -d nome-branch-locale`: permette di eliminare il *branch_G* locale indicato;
- `git branch -d -r nome-branch-remoto`: permette di eliminare il riferimento locale al *branch_G* remoto;
- `git fetch`: vengono raccolti i commit non presenti nel *repository_G* locale e salvati, tuttavia non avviene il merge automatico;
- `git reset --hard X`: scarta tutte le modifiche effettuate nel *repository_G* locale e ripristina i *file_G* alla versione del commit X.