Git guida (in aggiornamento)

Cos'è git?

Git è un software di controllo versione distribuito utilizzabile da interfaccia a riga di comando, creato da Linus Torvalds nel 2005. Git lavora con i repository. Un repository git ha 4 stati di lavoro. Il primo è la tua directory corrente. Il secondo è l'index che fa da spazio di transito per i files (git add *). Il terzo è l'head che punta all'ultimo commit fatto (git commit -m "messaggio"). E l'ultimo è il repository è online (git push server).

I repository online e locali possono essere divisi in ramificazioni (Branch). I branch (ramificazioni) permettono di creare delle versioni assestanti del codice master. Queste versioni "assestanti" permettono la creazione di features o aggiornamenti in fase alpha che non vanno ad intaccare minimamente il codice del progetto. Finito la scrittura della ramificazione il branch verrà unito con il master

Git permette di gestire i tag. I tag sono le versioni del software in uso. I tag registrano tutti i commit fino al rilascio nel server.

Configurazioni di base di git

Configuriamo il nostro git con le nostre credenziali di GitHub:

```
git config --global user.name 'Tuo Nome GitHub'

git config --global user.email email@github.com
```

Ci sono due modi per instanzire un progetto

1. Inizializziamo un progetto non esistente:

```
git init
```

2. Inizializziamo un progetto esistente su un server git:

git clone serverURL.git Esempio: git clone https://github.com/tesseslol/irixos-websites.git Git clone permette di copiare il .git file del server e anche il repository.

Configurazione del server remoto

Con questo comando visualizziamo la lista di server remoti salvati con relativo url:

```
git remote -v
```

P.S. di solito il server principale si chiama origin

Ora aggiungiamo un server remoto:

```
git remote add identificatoreServerRemoto UrlServerRemoto

Esempio: git remote add origin https://github.com/tesseslol/irixos-websites.git
```

Lavoriamo nel progetto:

Aggiungiamo i file dalla directory del progetto all'index:

```
git add nome_file
```

Si può utilizzare l'asterisco per aggiungere tutti i file. Se si vuole escludere un file dalla selezione totale (con l'asterisco) basta creare un file denominato .gitignore e metterci all'interno i file che non si vogliono aggiungere al INDEX.

Ora aggiungiamo i file dell'index all'head:

```
git commit -m "Messaggio del commit"
```

Per non tracciare il file usiamo l'argomento -a:

```
git commit -a -m "Messaggio del commit"
```

Annullamento dei commit:

```
git commit --amend
```

Cancellare un file da git:

```
git rm nomeFile
```

Il file ritorna allo stato precedente dell'ultimo commit:

```
git checkout -- nomeFile
```

Lavorare con il server remoto

Aggiornare il tuo repository locale alla commit più recente:

```
git pull
```

Se vogliamo fare l'upload dei commit nel progetto usiamo:

```
git push identificatoreServerRemoto nomeBranch
```

```
Esempio: git push origin master
```

Se vogliamo rinominare un file in remoto:

```
git remote rename identificatoreServerRemoto nomeFileVecchio nomeFileNuovo
```

Se vogliamo eliminare un file in remoto:

```
git remote rm nomeFile
```

Stato del progetto

Per vedere le modifiche del progetto digitiamo:

```
git status
```

Per vedere i cambiamenti dei singoli files digitiamo:

```
git diff
```

Vedere tutti i commit:

```
git log
```

Gestire i tag

Per visualizzare tutte le versioni eseguimo il comando:

```
git tag
```

Per visualizzare tutte le versioni con un determinato numero:

```
git tag -l 1*
```

Creazione di un tag:

```
git tag -a versioneSoftware -m "nota sul tag"
```

Esempio: git tag -a 1.2.3rc1 -m "aggiornato la navbar"

Vedere tutte le modifiche di un tag:

```
git show 1.2.3rc1
```

Condividere i tag:

```
git push identificatoreServerRemoto tagDaPubblicare
```

Esempio: git push origin 1.2.3rc1

Condividere tutti i tag:

```
git push identificatoreServerRemoto --tag
```

Esempio: git push origin --tag

Gestire i Branch

Lista dei Rami:

```
git branch
```

Creiamo un branch con:

```
git branch nomeBranch
```

Esempio: git branch feature

Cambia i rami:

```
git checkout nomeBranch
```

Esempio: git checkout feature

Per ritornare al branch originale digitiamo:

git checkout master

Eliminare il ramo:

git branch -d nomeBranch

Esempio: git branch -d feature

Crea il ramo e passa a quel branch:

git checkout -b nomeBranch

Esempio: git checkout -b feature

Per unire il branch al repository originale usiamo (ricordatevi di fare un commit nel branch):

git checkout master

git merge feature

Git Parameters:

*** Inizializza l'area di lavoro ***

clone Clona un repository in una cartella

init Crea un git repository o ne inizializza uno

*** Lavorare nel progetto corrente ***

add Aggiungere i file nel INDEX

mv Muove o rinomina un file, una directory

reset Resetta il corrente HEAD nello stato specificato

rm Rimuove i file dalla directory corrente e nel INDEX

*** Mostra la cronologia e lo stato ***

bisect Use binary search to find the commit that introduced a bug

grep Print lines matching a pattern

log Mostra i commit log

status stato del contenuto di un progetto

show Show various types of objects

*** Grow, mark and tweak your common history ***

branch Visualizza, crea e elimina ramo (branches)

checkout Cambia ramo (branches) o ripristina la strotura dell'area di lavoro

commit Registra le modifiche del repository

diff Confronta i commit (esp: commit e area di lovoro)

merge Unisce una o più cronologie di sviluppo

rebase Reapply commits on top of another base tip

tag Crea, visualizza la lista, elimina o verifica il tag della versione del progetto

*** Collabora ***

fetch Download objects and refs from another repository

pull Fetch from and integrate with another repository or a local branch

push Update remote refs along with associated objects

Citazioni usate: