Comandi per l’utilizzo di GitLab da riga di comando su S.O. Linux
16 Febbraio 2023

1. Verificare se esiste già una versione di git installata sul proprio S.O.:

```
git --version
```

2. Per aggiornare l’eventuale versione già presente oppure per installare git la prima volta:

```
sudo apt-get update
sudo apt install git
```

3. Impostare nome utente e indirizzo e-mail che si vogliono utilizzare per identificarsi nei progetti:

```
git config --global user.name "NOME_UTENTE"
git config --global user.email INDIRIZZO_E-MAIL
```

4. È consigliato, ma non obbligatorio, impostare l'alias lg per una versione personalizzata del comando log al fine di migliorarne la leggibilità:

```
git config --global alias.lg "log --pretty=format:'%C(yellow)%h
%Cred%d%Creset - %C(cyan)%an %Creset: %s %Cgreen(%cr)' --
decorate --graph --all -abbrev-commit"
```

5. Clonazione di un repository remoto (già esistente sul nostro account GitLab):
    - Creare una directory nella nostra Home da utilizzare per progetti Git, ad esempio GitProject
    - Utilizzando il nostro browser preferito, accedere al nostro repository remoto su GitLab e copiare il link del repository
    - Usando il terminale, all’interno della directory che abbiamo creato, digitare:

```
git clone LINKCOPIATO
```

6. Stato del repository

```
git status
git log
git lg
```

7. Aggiungere un nuovo file nello stato di stage (pronto per essere committato all’interno del repository):

```
git add nomefile
```

oppure

```
git add .
```

oppure, per aggiungere le modifiche di un intero folder

```
git add nomefolder/*
```

Esempio:

Supponiamo di avere già il file pippo.txt e la directory miadir con dentro il file pluto.txt e di volerli aggiungere al nostro repository, digitare:

```
git add pippo.txt      #aggiunge solo il file pippo.txt in stage
git add miadir/*       #aggiunge l'intera dir in stage
```

in alternativa, usare:

```
git add .          #aggiunge tutto ciò che è nuovo in stage
```

verificare che i nuovi elementi siano stati aggiunti nello stato di stage

```
git status
```

8. Per applicare le modifiche fatte, si utilizza il comando commit con le seguenti opzioni:

   -a per committare tutti i files in stato stage

   -m per inserire il messaggio associato al commit

   Esempio di utilizzo:

```
git commit –a –m "ho aggiunto dei nuovi file"
```

Best Practices: https://gitlab.com/itisrivoira/Prova5C.git
- usare l'imperativo
- max 50 chars
- no punti finali
- dichiarare le modifiche fatte ad alto livello
- includere eventuali riferimenti a documentazione (e.g. jira code, tiketing...)

Esempio di utilizzo:

```
git commit –m "[type][branch] What is done "
```

type è il tipo di modifica:
feat: funzionalità nuova
fix: correzione errore
chore: piccola sistemazione
test: copertura del codice
doc: documentazione
Se il branch è main, si esclude dal commit message.

9. Pubblicazione delle nuove modifiche con il repository REMOTO

```
git push
```

10. Sincronizzazione con il server remoto:
    - Il comando pull racchiude in sé il fetch ed il merge
    - Il comando fetch permette di scaricare in locale le modifiche remote, ma NON di applicarle
    - Il comando merge combina le modifiche fatte

    Esempi di utilizzo:

```
git pull
git fetch origin
git merge origin/main
```

11. Gestione dei branch

```
git branch nomeBranch            #Creazione
git branch -d nomeBranch         #Eliminazione
git checkout nomeBranch          #Cambiare il branch corrente

#Effettuare il push su GitLab del nuovo Branch
git push --set-upstream origin nomeBranch

#Riportare le modifiche su main:
git checkout main      #Spostarsi sul branch di destinazione
git merge nomeBranch   #Effettuare il merge

#NOTA:  per fare il merge occorre essere nel branch di
#       destinazione, mentre nel comando bisogna indicare il
#       branch sorgente.
```

Riferimenti online:

- https://git-scm.com/
- https://get-git.readthedocs.io/it/latest/index.html#
- https://rogerdudler.github.io/git-guide/index.it.html