**GIT (l’essenziale da sapere)**

**Configurazione del file config di git con le nostre credenziali:**

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

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

**Creazione repositories**

Per creare una repository occorre:

se vuoi clonare una repository remota da git

git clone

oppure se vuoi agganciare una repository locale ad una repository su git:

git init

git add .

git commit -m “primo commit”

git remote add origin https://github.com/lucaSchiavon/t\_for\_git

l’indirizzo remote della repository di git passato a git remote add può essere:

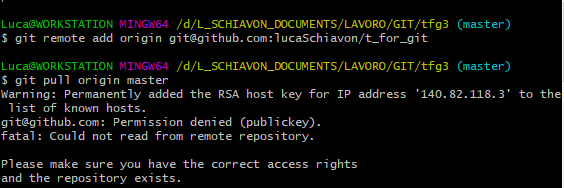
Https: https://github.com/lucaSchiavon/t\_for\_git.git

Oppure usare protocollo ssh.

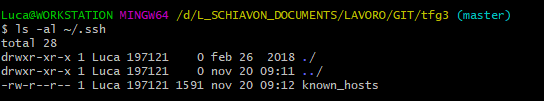
[git@github.com:lucaSchiavon/t\_for\_git](mailto:git@github.com:lucaSchiavon/t_for_git)

Uso chiavi nel secondo caso occorre usare criptografia assimmetrica ossia generare una chiave privata e pubblica sulla macchina locale e configurare git con la chiave pubblica.

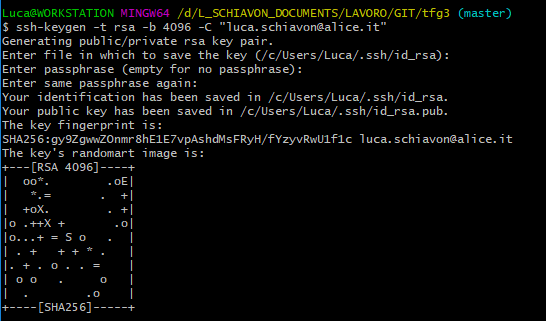
Infatti se si cerca di agganciare una directory remota di git da locale via ssh si ottiene il seguente errore:



Quindi occorre innanzitutto controllare di non avere già chiavi generate sulla macchina ispezionando la directory ~/.ssh.:



Devi cercare una coppia di chiavi dal nome simile a qualcosa e qualcosa.pub, dove quel qualcosa in genere è id\_dsa o id\_rsa. Il file .pub è la tua chiave pubblica e l'altro file è la chiave privata. Se non hai questi file (o non hai una directory .ssh), puoi crearle avviando un programma chiamato ssh-keygen.

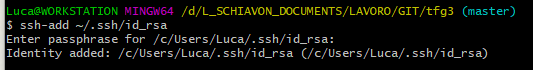


Qui si può specificare in che file .pub salvare la chiave e la passphrase che si può lasciare vuota se non vuoi usare la password quando usi la chiave.

Quindi verificare che l’agente ssh stia girando



Aggiungo la chiave privata all’agent:

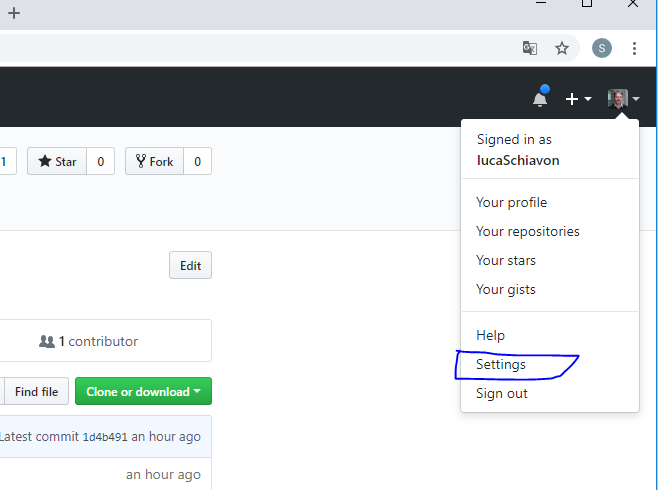


Copio la chiave pubblica negli appunti

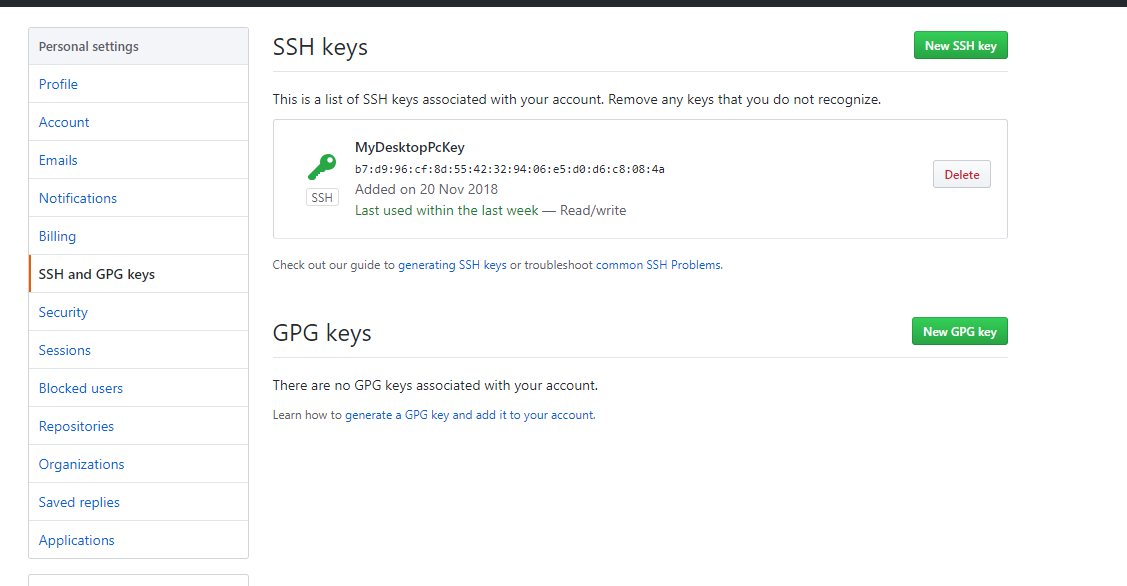




Vado su github setting:



Ed incollo la chiave pubblica:



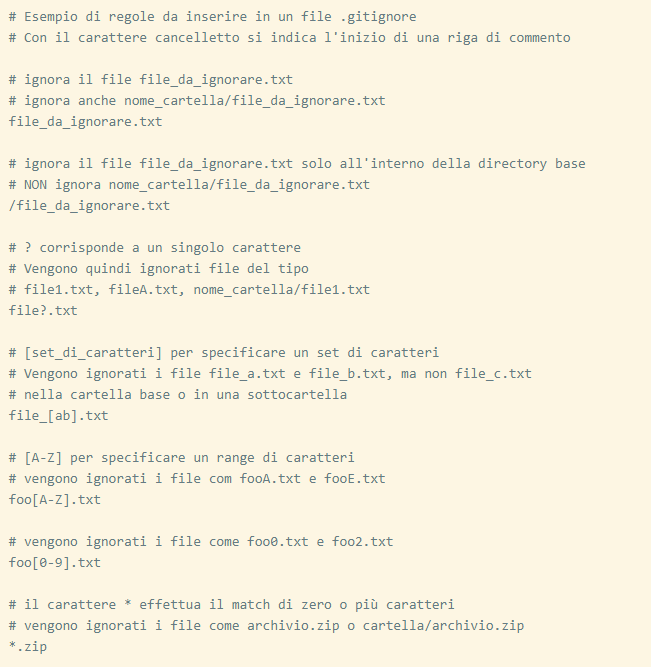
GIT Ignore

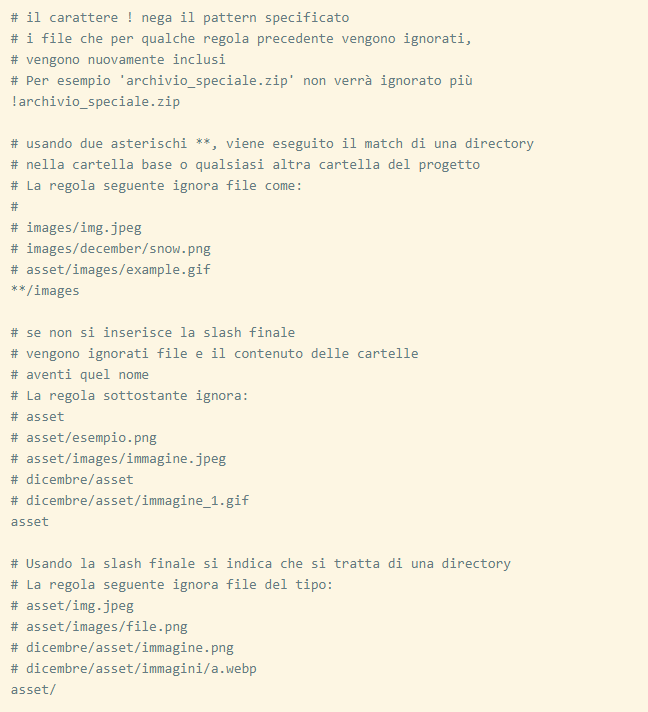
**Git** si accorge delle modifiche apportate e ci segnalava quali file potevano essere inseriti nella Staging Area e quindi all'interno del prossimo commit. Ci sono però alcune situazioni in cui è necessario **ignorare dei file e fare in modo che Git non tenga traccia di eventuali modifiche**. Alcuni possibili casi sono:

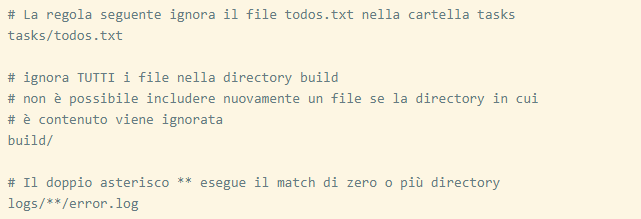
* cartelle create da un package manager come NPM o bower
* directory contenenti i file da caricare su un server per un'applicazione web
* File di supporto generati dal sistema operativo o da altri strumenti
* file eseguibili generati a partire dal codice sorgente
* archivi compressi come file.gz o.zip

Per selezionare dei file da ignorare in un determinato progetto, possiamo creare un file **.gitignore**nella cartella base. (Sarebbe possibile inserire più file.gitignore in cartelle diverse ma solitamente si usa un file solo).

Sintassi da usare in gitignore:







Rimuovere da gitignore dopo avere già committato i file

Se al momento del clone o dell’inizializzazione di una nuova repository non sono già presenti le istruzioni di esclusione in gitignore e si fa una modifica o si aggiunge un file/cartella questa risorsa, se successivamente viene esclusa nel file git non viene effettivamente esclusa dalla cache, per farlo bisogna lanciare un rm –cached.

Rimuove fisicamente I file in public con estensione txt

Git rm public/\\*.txt

Oppure per rimuovere fisicamente la cartella Public ed i suoi contenuti

Git rm -r Public

Se si vuole rimuovere solo dall’area di staging e mantenere in locale Public usare –cached:

Git rm -r –cached Public

By default git won't sync / add empty folders. If you need an

empty folder place an empty .**gitkeep** file in it

Commit

Se si aggiunge un file per committare occorre:

aggiungere il file all’area di stage:

git add

committare il file

git commit -m “blabla”

se il file è solo stato modificato si puo bypassare l’area di stage aggiornando direttamente al repository:

git commit -a -m “bypassing staging”

Branch

Lista rami:

git branch

creazione branch

git branch nomeBranch

mi sposto su un branch

git checkout nomeBranch

per unire due branch:

git checkout master

git merge feature

Comandi vari

Git reset –hard HEAD rollback all’ultimo commit

git revert 85431554a2c932310a6eed0f1ef6b0bfdad70693 torna ad un commit specifico

git revert HEAD~1 torna all’ultimo commit

git checkout 85431554a2c932310a6eed0f1ef6b0bfdad70693 path/to/file.html

git checkout HEAD~1 path/to/file.html

Git log (lista dei commit)

Git stash

Git rebase

Git fetch

Pull request

Una pull request serve per collaborare allo sviluppo , con una pull request puoi chiedere ad altri di revisionare i cambiamenti che hai fatto:

Ambienti visuali – tool:

consigliato source tree come ambiente visuale e cmder al posto di gitbash

Comandi principali git:

https://gist.github.com/tesseslol/da62aabec74c4fed889ea39c95efc6cc