

TortoiseGit

TortoiseGit è un client GUI free per la gestione delle versioni sul sistema Git; in particolare, gestisce i file nel corso del tempo che sono memorizzati in un *repository* locale. Ciò consente di recuperare versioni precedenti dei file ed esaminare la cronologia di come e quando i dati sono cambiati.

Git è un *sistema di gestione di versione open source* progettato per gestire dai piccoli ai grandi progetti con velocità ed efficienza.

Tutti i comandi Git sono disponibili dal menu contestuale di Explorer e TortoiseGit viene gestito attraverso un proprio sottomenu.

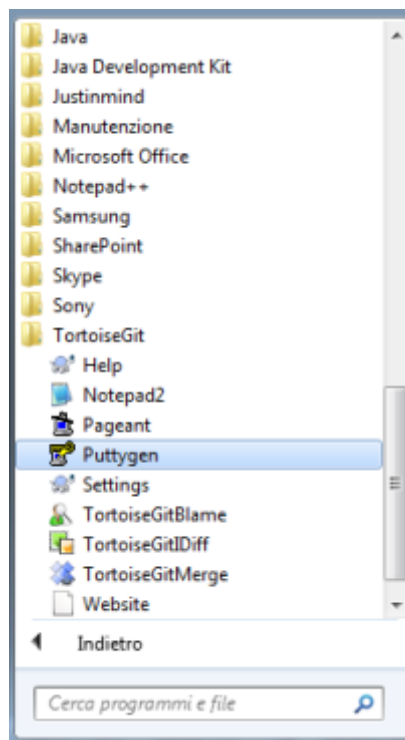
Git fornisce a ogni sviluppatore una copia locale di tutta la history dello sviluppo e le modifiche vengono copiate da un repository ad un altro. Questi cambiamenti sono importati come ulteriori rami di sviluppo (development branches), e possono essere fuse nello stesso modo come un development branch locale. I repositories sono facilmente accessibili tramite il protocollo Git (contenuto in ssh per l'autenticazione e la sicurezza) o semplicemente utilizzando http.

Installazione TortoiseGit e utilizzo con GitHub

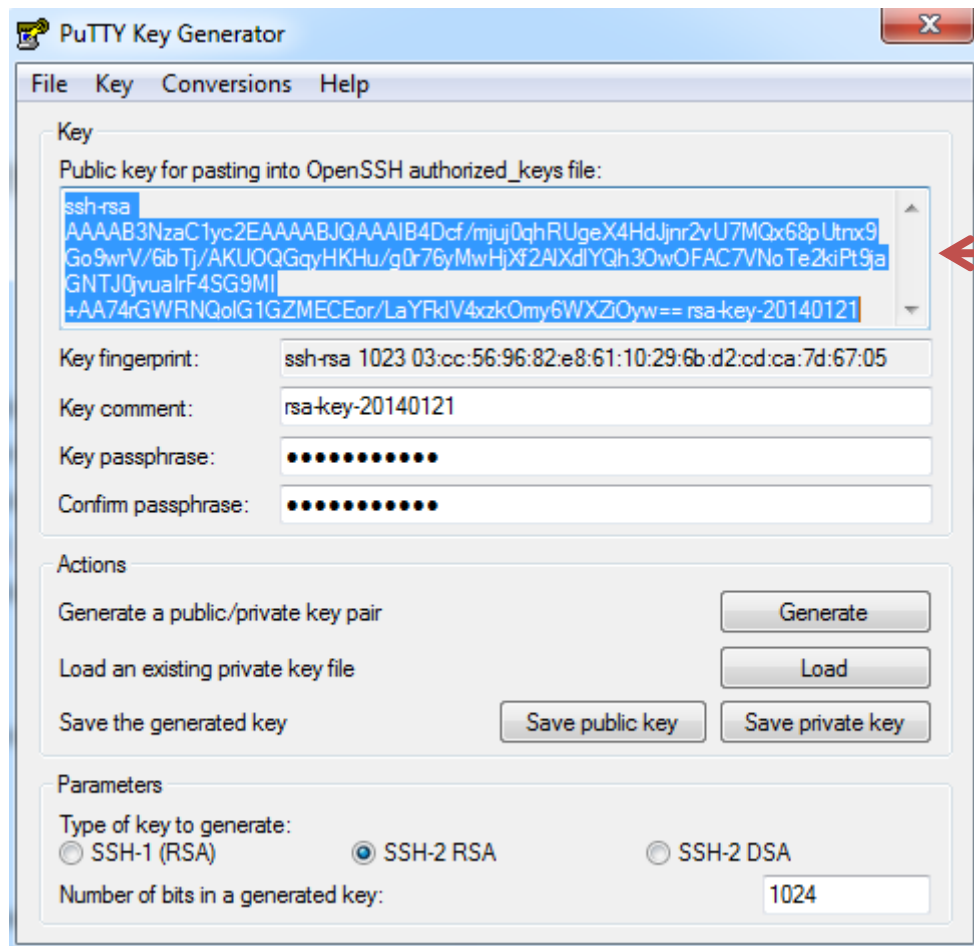
E' necessario scaricare il file di installazione al seguente link:

<https://code.google.com/p/tortoisegit/wiki/Download>. E' possibile scaricare anche il language package in Italiano.

Prima di iniziare a configurare un progetto su GitHub, è necessario creare delle key con "Puttygen"



- selezionare “Generate”
- salvare la private key e public key nella cartella .ssh presente al percorso C:\Users\nomexxx (nel caso in cui la cartella non è presente, crearla da linea di comando con il comando mkdir .ssh)
- selezionare e copiare l’ssh key



- andare su <https://github.com/> e selezionare account setting



- nel menù di sinistra selezionare “SSH Keys”
- aggiungere la key inserendo un titolo e l’ssh key

Add an SSH Key

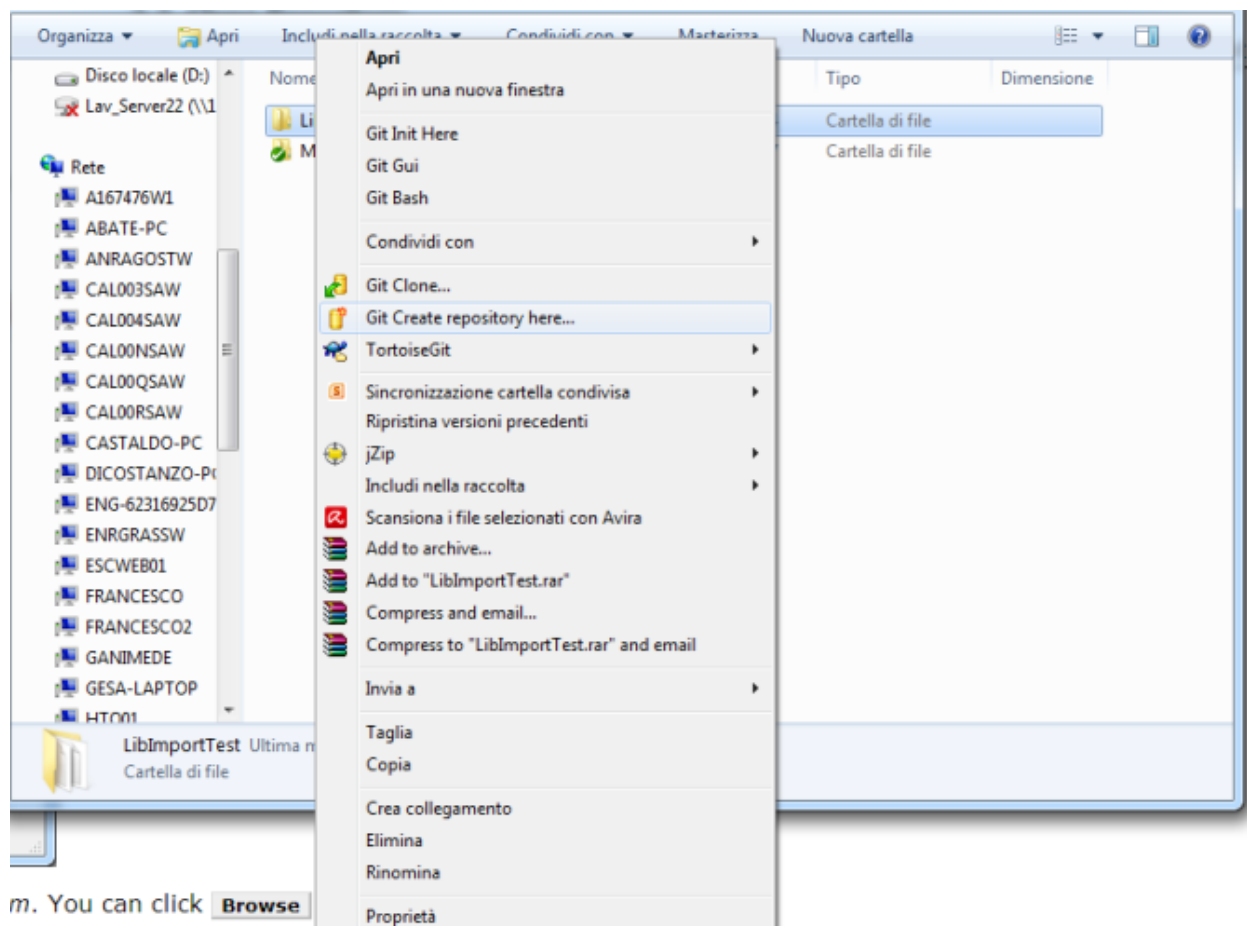
Title

Key

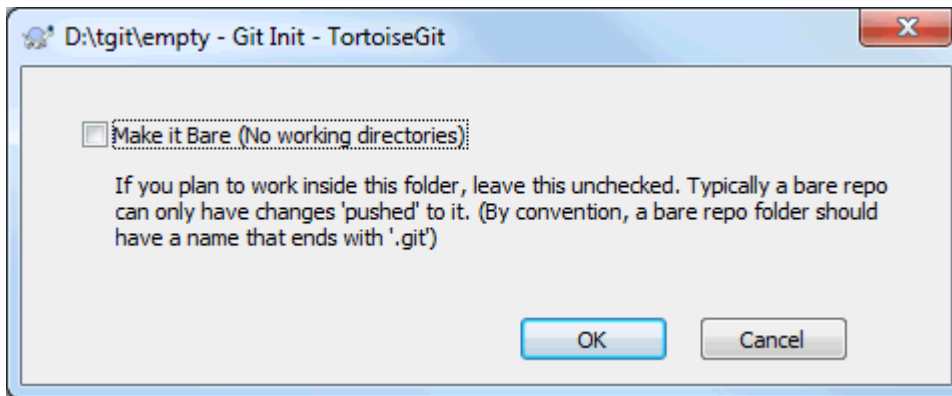
```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAIB4Dcf/mjuj0qhRUgeX4HdJnr2vU7MQx68pUtnx9Go9wrV/6ibTj/AKUOQGqyHKH
u/g0r76yMwHjXf2AIXdlYQh3OwOFAC7VNoTe2kiPt9jaGNTJ0jvualrF4SG9MI+AA74rGWRNQoIG1GZMECEor/LaYFkl
V4xzkOmy6WXZiOvw== rsa-key-20140121
```

Add key

Adesso è necessario creare un progetto attraverso Eclipse o Android Studio. Per caricare un repository su GitHub è necessario cliccare con il destro sul package del progetto e selezionare “Git Create repository here”.



Uscirà una schermata come la seguente, in cui è necessario cliccare su Ok.



In questo modo si è creato un repository vuoto.

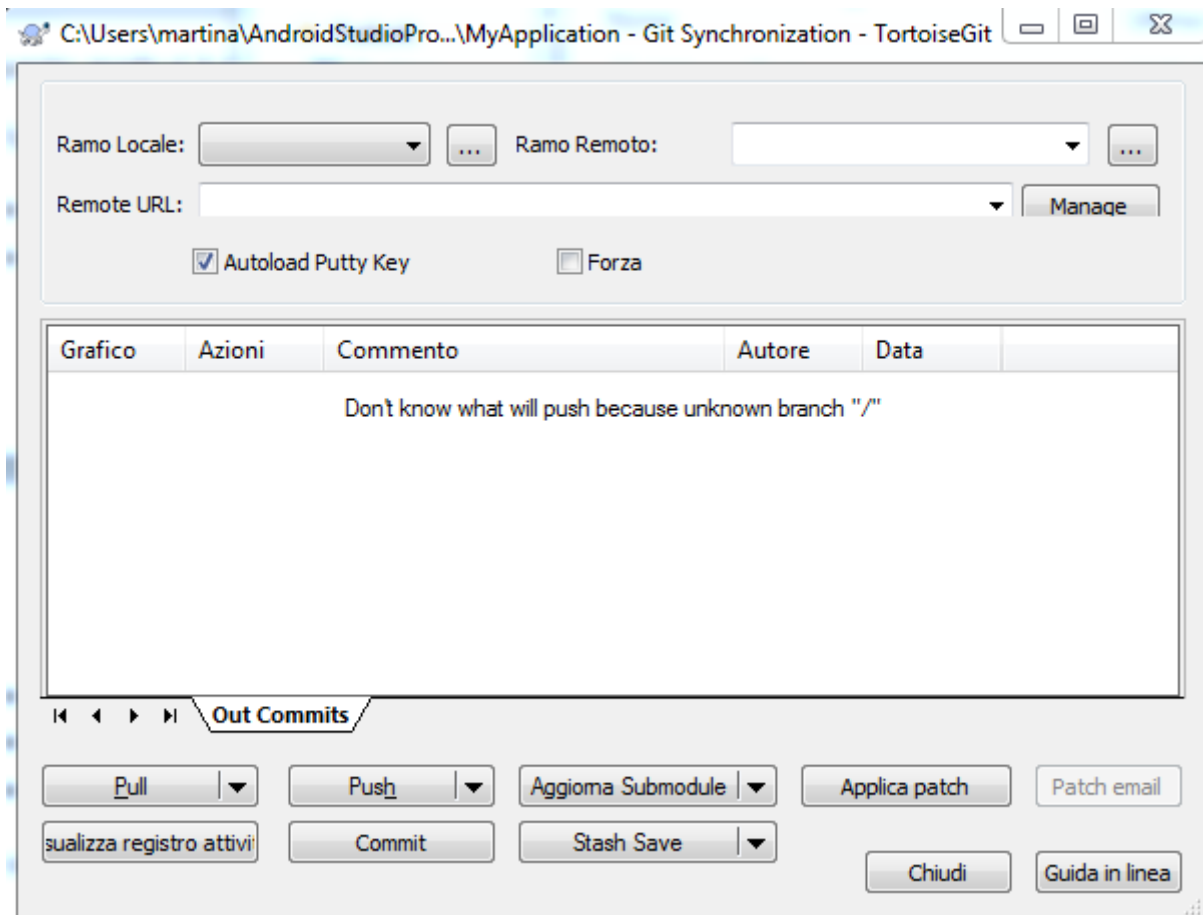
Per fare il commit sarà necessario:

- cliccare di nuovo con il tasto destro sul package del progetto
- selezionare "Git Commit ->"master"..."
- selezionare i file relative alla sezione "Modifiche effettuate"
- inserire un messaggio di commit e premere Ok
- Premere Close

Per sincronizzare su git sarà necessario:

- cliccare di nuovo con il tasto destro sul package del progetto
- selezionare "Git Sync.."

si aprirà una nuova schermata in cui sarà necessario inserire l'URL del repository su GitHub.



Per fare ciò, sarà necessario creare un nuovo repository su GitHub, copiare l'SSH

Quick setup — if you've done this kind of thing before

Set up in Desktop

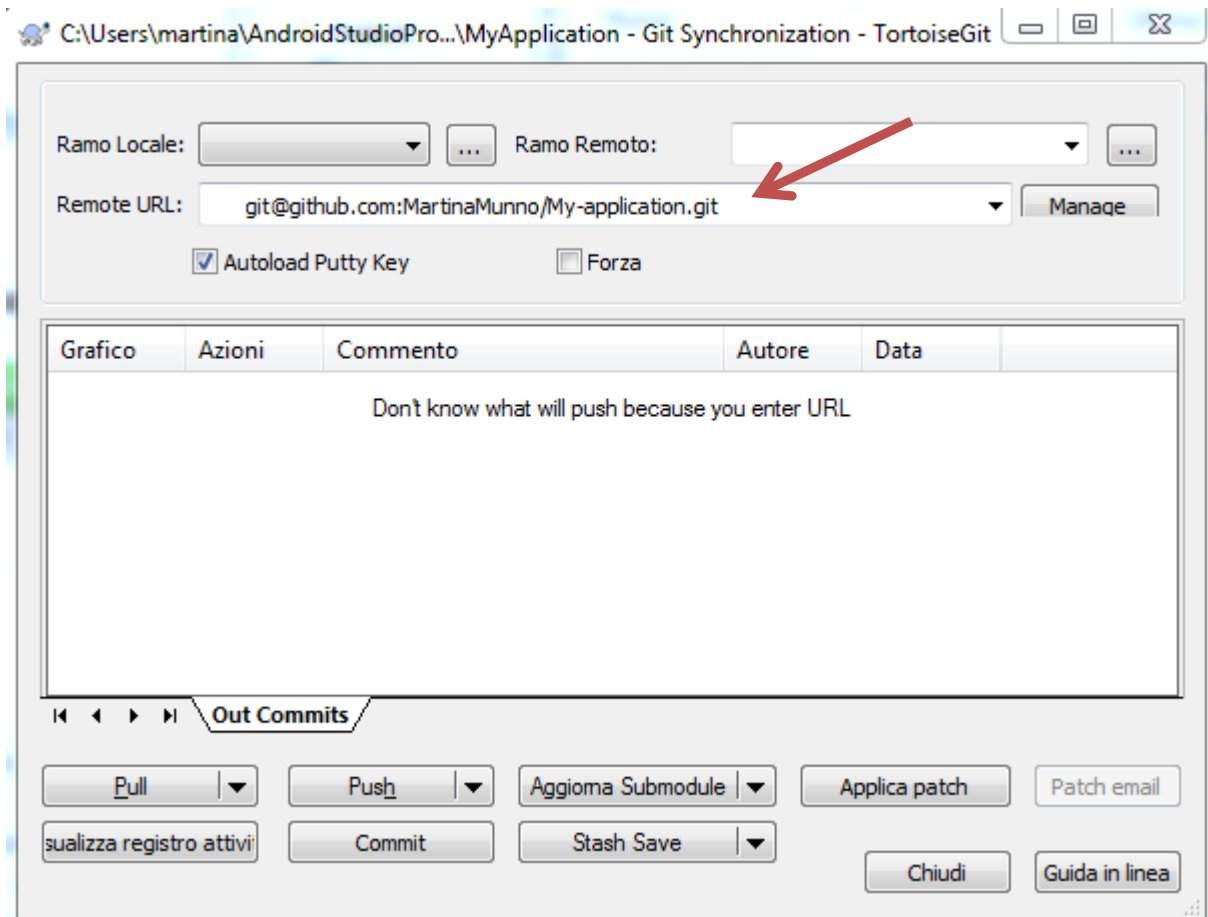
 or

HTTP

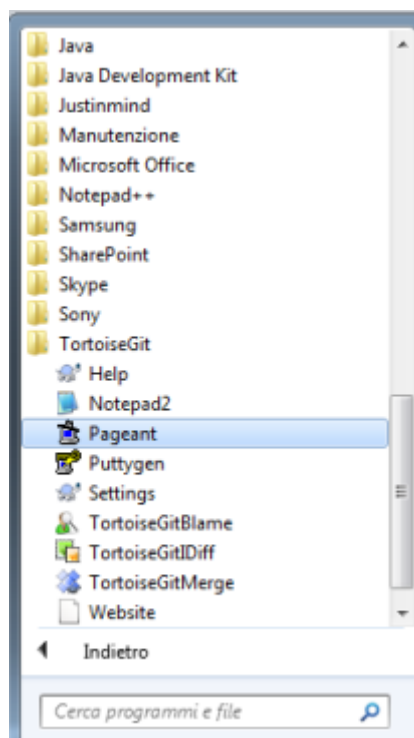
SSH

git@github.com:MartinaMunno/My-application.git

e inserirlo in Remote URL

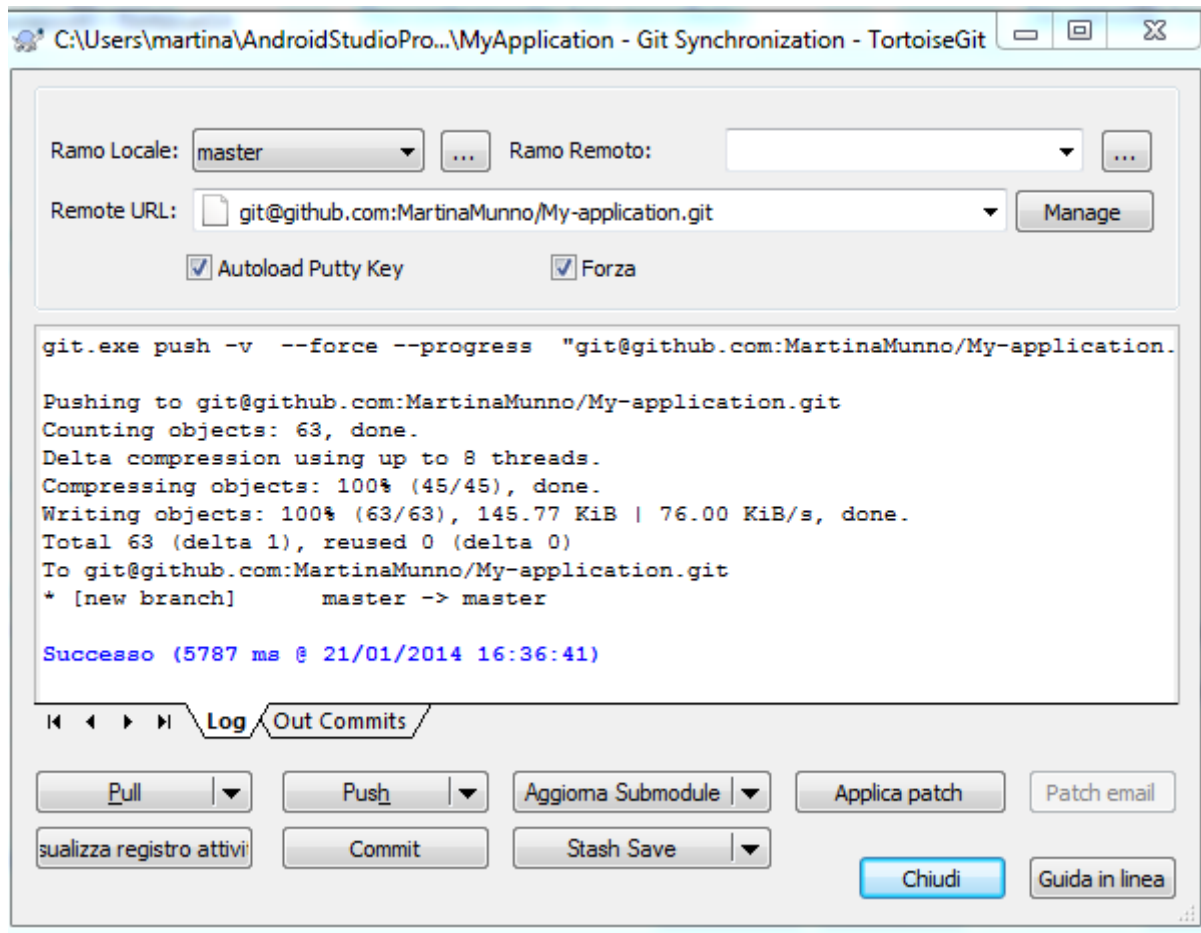


Aprire Pageant dalla directory di TortoiseGit




- aggiungere la private key creata precedentemente e premere Close
- tornare alla schermata precedente e premere push


(BUG: inizialmente non trova “master” per il ramo locale. Bisogna cliccare su “...” e aggiungerlo manualmente)





Ora il progetto è stato sincronizzato su GitHub:

Prova — Edit

 1 commit

 1 branch

 0 releases

 1 contributor

 branch: **master** **My-application** / 

prova

 **MartinaMunno** authored 11 minutes ago latest commit 68bd25e244 

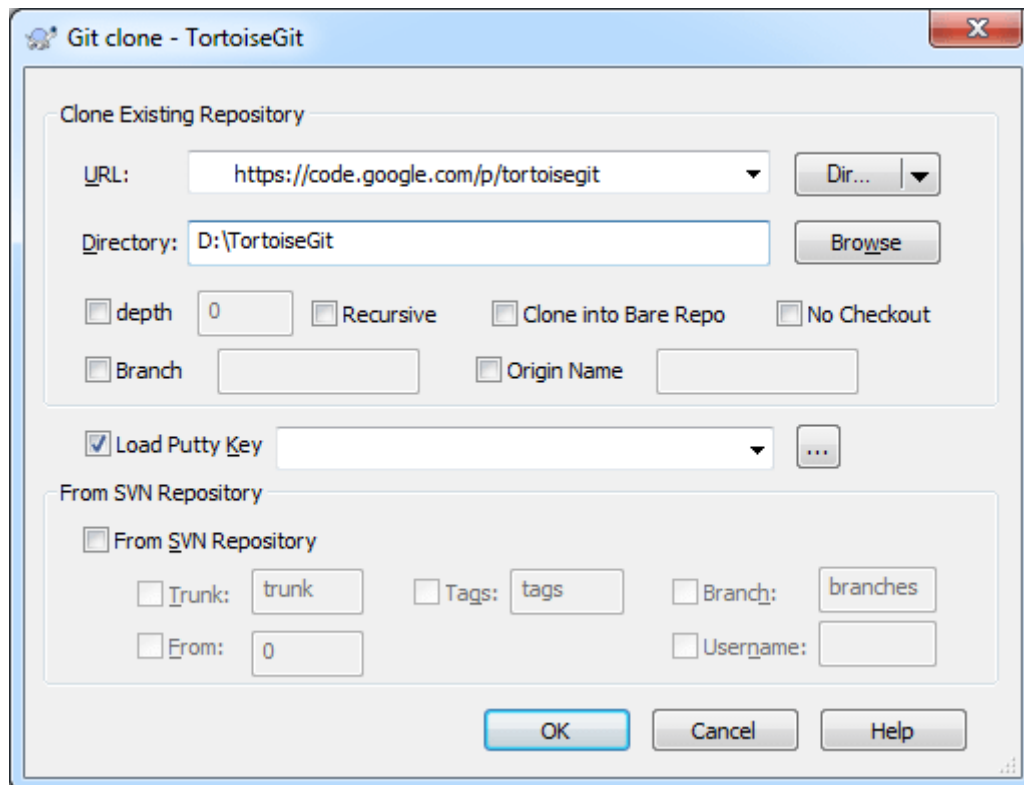
 .idea	prova	11 minutes ago
 app	prova	11 minutes ago
 gradle	prova	11 minutes ago
 .gitignore	prova	11 minutes ago
 MyApplication.iml	prova	11 minutes ago
 build.gradle	prova	11 minutes ago
 gradle.properties	prova	11 minutes ago
 gradlew	prova	11 minutes ago
 gradlew.bat	prova	11 minutes ago
 settings.gradle	prova	11 minutes ago

Ogni volta che si modificano dei file, è necessario effettuare il commit e la sincronizzazione su git.

Per configurare l'autenticazione con un repository su GitHub, è utile la seguente guida online:

<https://www.youtube.com/watch?v=fNPLuJTTto0>

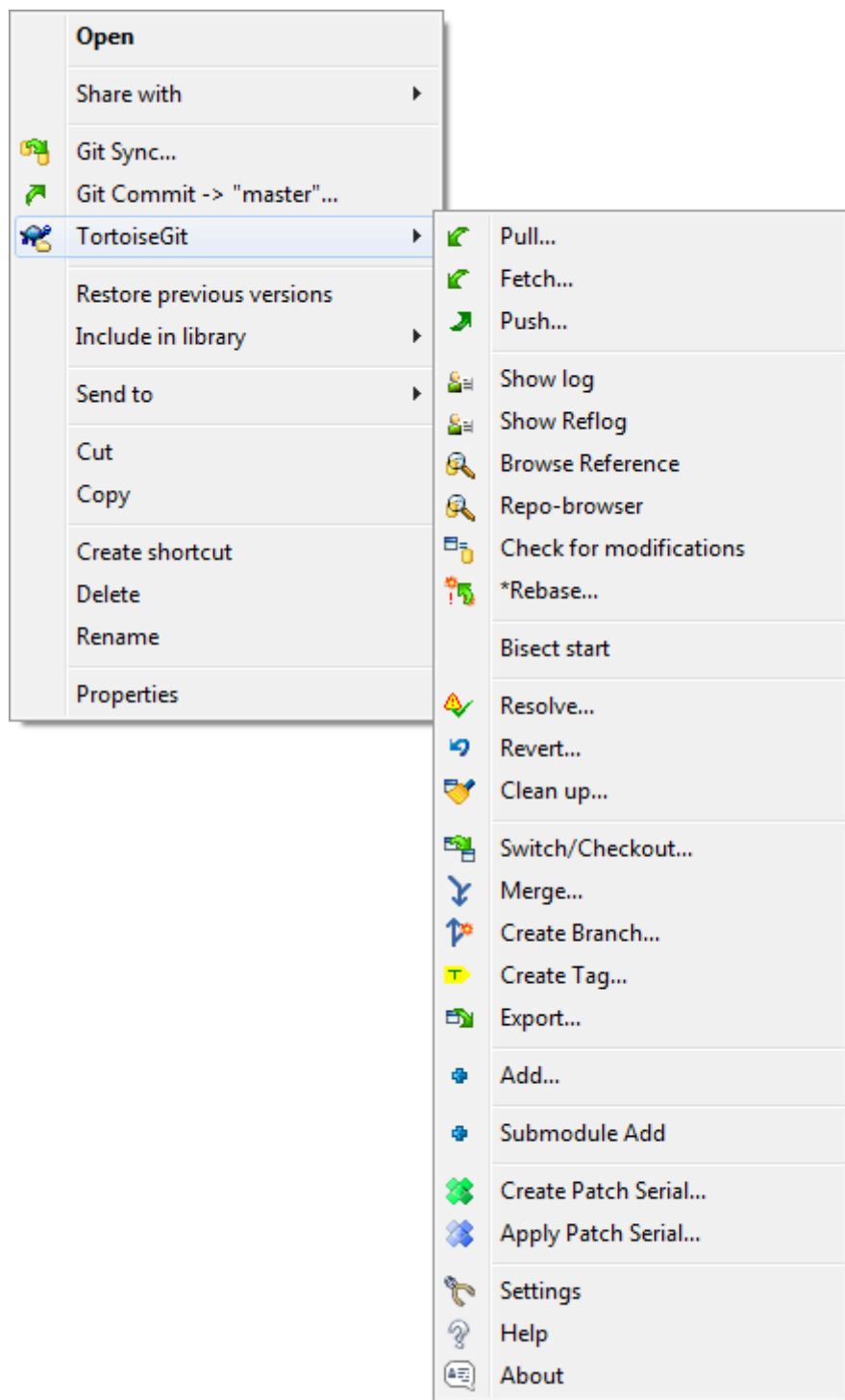
E' possibile clonare con repository su Git. Bisogna selezionare "Git Clone.." e si aprirà una finestra di dialogo come la seguente immagine.



Url: Inserire l'indirizzo URL del repository da cui si deve clonare. È possibile fare clic **su Sfoggia** per navigare fino alla directory.

Directory: Inserisci la directory locale. È possibile fare clic **su Sfoggia** per navigare fino alla directory.

La seguente immagine presenta i comandi che è possibile effettuare utilizzando TortoiseGit:



- Switch/Checkout: Per ottenere uno special version working tree è necessario fare un *checkout* da un repository locale. Se si seleziona Force sovrascriverà il working tree sulla versione del repository. Se si seleziona Tag o Commit , si preferisce creare un nuovo ramo, altrimenti si lavorerà al "no branch".
- Check for modifications: è utile prima del commit, per verificare quali file sono stati modificati localmente.

- Fetch: si può fare in qualsiasi momento e serve ad aggiornare la copia locale in un branch remoto, senza effettuare il merge. Questo permette di rivedere le modifiche prima di integrarle nella copia del progetto.
- Pull: unisce le modifiche del branch remoto e branch locale facendo il merge.
- Push: sovrascrive le modifiche dal repository locale a quello remoto.
- Merge: permette di fare il merge tra due rami di sviluppo.