MANUALE GIT

Clonare una repository già esistente

- Mettersi nella cartella dove si vuole clonare la repo
o Cd “path cartella” à vado nel path indicato
o Git clone “indirizzo della repo remota”

Creare una repository in locale e metterla poi online

- Cd “path cartella”
- Git init —> dentro la cartella dove voglio creare la repo inizializzo git
- Git – version verifico la versione installata di git
- Mkdir
- Git add nome.estensione
- Git commit -m “messaggio del commit”
- Git branch —> comando per vedere i branch che ci sono in locale
- Git branch -r —> comando per vedere i branch che ci sono in remoto, non è detto che
coincidano. Può succedere che il branch locale non si veda in remoto, se ho creato la repo a
Partire dal locale. O viceversa se sto clonando una repo può darsi che io non veda il locale tutti i
branch che ho in remoto.

o Aggiugnere come gestire questa cosa

- Git branch -a —> visualizzo i branch in locale
- Git log —> cronologia dei commit che sono stati fatti in locale
- Git status
- Dir —> vedo cosa c’è dentro il path dove sono
- Git checkout “nome branch dove voglio andare”

- Git Push origin “branch dove effettuo il push”
- Git fetch --all —> questo comando lo uso quando per esempio ho clonato la repo e in locale
non vedo tutti i branch presenti in remoto, per allineare il tutto devo fare questo.
- git pull —> sinconizza in locale tutte le modifiche che possono essere avvenute in remoto, fare sempre
quando si apre la repo
- git checkout -b “nuovo branch” —> va a creare un nuovo branch e mi sposta su quello

Percorso da fare quando apro la repo:
- cd percoso/repo
- git status —> vedo lo stato attuale
- git pull —> sincronizzo con il remoto

Quando faccio una **modifica il locale e la voglio caricare in remoto** devo fare il push, ma prima devo stagiare tutti i cambiamenti altrimenti non funziona .
Procedimento:
- Aprire il terminale
- Andare nel path della repo
- Git status —> verifico se ci sono stati dei cambiamenti, se ci sono file non trackati
- Git add filemodificato.estensione —> stage dei cambiamenti che sono stati fatti
- Git commit -m “messaggio”
- Git push origini branchsucuisono
N.B. prima di fare un push devo avere fatto un pull, cioè la cartella locale deve essere sincronizzata a quella in remoto, ma vedi sotto—>

Quando NON fare subito git pull —>Se hai **modifiche locali non salvate** (committate). In questo caso, meglio fare prima:

- git add .
- git commit -m "Salvo le mie modifiche locali"
- git pull

Fare un **git pull** prima di un **git push** è una **buona pratica** proprio per evitare problemi di **branch divergenti**.
branch divergenti COSA FARE:
- Git pull origin main - - rebase —> questa operazione può andare a buon fine oppure viene chiesto di risolvere i conflitti manualmente. Se va a buon fine significa che Hai allineato il tuo branch locale con il branch remoto, mantenendo le tue modifiche sopra le modifiche remote.
- git push origin main —> si può fare il push
A questo non avrebbe senso fare un pull.

Cosa fare quando ho dei cambiamenti che non Voglio pushare ma voglio cambiare branch:
In questi casi bisogna fare lo stash che sonstalziamente mantiene in maniera tempora i commit che non si vogliono caricare in locale.
Ho lavorato su un file ma ora voglio cambiare branch senza perdere le modifiche:
- git stash —> qui può verificarsi che i file non sono tracciati (per esempio non ho fatto l'add). Per includere nello stash i file non tracciati devo fare: git stash -u

- git checkout branch “dove voglio andare”
—— faccio quello che devo fare
- Git checkout main —> torno al branch dove avevo stagiato i cambiamenti
- git stash pop
- git commit -m “messaggio”
- git log —> verifico il commit
- git push origin main

Git stash ha senso se non fai il commit, altrimenti non ha senso.

Se il file contiene degli spazi devo usare il backslash —>git add SENSOR\ PCB\ .rtf —> meglio usare le virgolette “ciao Lucia.txt”. Se è una cartella non devi indicare l’estensione.

Come fare il merge di due branche partendo dalla creazione di uno:
- git pull
- git checkout main
- git checkout -b “nuovo branch”
— — lavoro sul nuovo branch facendo quello che voglio
- git checkout main —> torno sul branch main
- Git pull origin main —> mi assicuro che sia aggiornato
- git merge “branch che voglio unire al main”

A questo punto il branch main contiene tutto quello che c’era dentro il branchprova quindi avrebbe senso eliminarlo per tenere le repo più ordinata:
- git checkout main —> prima di eliminare un branch mi

devo assicurare di essere su un altro branch
- git branch -d branchprova —> -d elimina il branch solo se le modifiche sono già state mergiate, -D posso forzarlo, meglio non usarlo. Cosi lo elimino solo in locale
- git push origin - - delete branchprova —> eliminate il branch in remoto
- git fetch -p —> pulisci i branch locali
Perchè sparisca anche dalla visualizzazione in locale devo eliminarlo in remoto.

Git init

- rm -rf ~/.git

Se la repo l'ho creata in locale non devo aggiungere il readme quando la creo anche in remoto.

git remote add origin https://github.com/tuo-username/tuo-repo.git
git remote -v