Introduzione a Git, Parte 2 - Quali sono le funzioni principali di Git

Nella prima puntata di “Introduzione a Git” abbiamo visto cos’è un sistema di controllo versione e perché Git è tra i sistemi più usati per la gestione di progetti a distanza e non.

Nella guida abbiamo spiegato come installare e impostare Git, quindi ora dovreste averne una versione funzionante e configurata con i vostri dati.

Nell’articolo di oggi, invece, impareremo a utilizzare i comandi base per operare in tutta tranquillità.

Inizializzare o clonare una repository

Una repository è la location dove sono conservati tutti i dati relativi a un progetto; nel nostro caso può essere riferita alla cartella contente tutto il codice a cui stiamo lavorando.

Per tracciare un progetto con Git per la prima volta è possibile sia inizializzare la repository che clonarla.

Se decidiamo di inizializzare la repository, tracceremo il progetto partendo da zero. Se invece decidiamo di clonarla, scaricheremo un progetto già tracciato che si trova online.

1) Inizializzare una repository

Per inizializzare la repository dovete:

- aprire il terminale (prompt dei comandi)

- posizionarvi nella root del vostro progetto

- digitare “git init”

Verrà creata una cartella nascosta chiamata .git che conterrà tutti i file necessari per il tracciamento della vostra repository. Ricordate però che, a questo punto, il vostro progetto non è ancora tracciato.

Clonare una repository

Per avere una copia di una repository Git già esistente e reperibile online, basta aprire il terminale,

entrare nella cartella dove vogliamo posizionare il codice e digitare:

ovviamente, sostituiremo URL con l'indirizzo del repository.

Ad esempio, mettiamo caso che dovete utilizzare bootstrap nel vostro progetto:

Normalmente andreste su http://getbootstrap.com/ per scaricare il file zip e inserireste i file presenti nell'archivio all'interno del vostro progetto.

Con Git invece basterà eseguire sul terminale il comando:

Questo comando, nell'ordine:

- crea una cartella "bootstrap"

- inizializza una cartella .git dentro di essa

- scarica tutti i dati per questo repository

- imposta la copia di lavoro dell'ultima versione

In questo modo il framework sarà presente nel vostro progetto e potrete aggiornarlo in qualsiasi momento (dopo vedremo come).

Controllare lo status del repository

Abbiamo creato la cartella "mioprogetto" e abbiamo inizializzato Git, ora possiamo cominciare a lavorarci e a inviare le modifiche al database di Git ogni volta che il progetto raggiunge uno stato che vogliamo registrare.

Ogni file presente nella cartella di lavoro può avere due stati: tracciato o non tracciato. I file tracciati possono essere **non modificati**, **modificati** o **staged**.

I file non tracciati sono tutti gli altri.

Lo strumento principale per determinare quali file sono in un certo stato è il comando *git status*. Lanciando questo comando direttamente dopo aver fatto una clonazione accadrà questo:

Questo significa che la cartella su cui stiamo lavorando è pulita (clean), non c'è traccia di file

modificati. Git inoltre non individua altri file non tracciati, altrimenti li elencherebbe.

Infine, il comando ci dice in quale ramo del progetto ci troviamo (branch) ma di questo parleremo più tardi.

Ora provate ad aggiungere un file “myfile.html” all’interno della cartella “mioprogetto” ed eseguite di nuovo Git status. Vedremo il file non tracciato come segue.

Come potete vedere, il nuovo file è nell’elenco degli “Untracked files”; ciò significa che non è tracciato nel database. Git inizierà ad includerlo nel db solo quando gli daremo il comando atto a farlo.

Tracciare un file: git add

Per iniziare a tracciare un nuovo file, usiamo il comando:

Per tracciare il file da noi aggiunto poco fa useremo quindi:

e lanciando nuovamente il nostro git status noteremo che Git ci risponde nel seguente modo:

Il file risulta tracciato e in stato di stage. Ora possiamo eseguire il nostro primo commit ovvero possiamo memorizzare lo stato attuale in cui si trova il nostro progetto. Ricordate che qualsiasi file che non si trovi in stato di stage non andrà nel commit, rimarrà solamente nel disco rigido e non verrà registrato nel database.

Registrare lo stato del progetto: git commit

Per poter ripristinare in qualsiasi momento una qualsiasi fase del progetto a cui stiamo lavorando è necessario registrare man mano lo stato in cui si trova il progetto. Il modo più semplice per farlo è attraverso il comando:

Possiamo inoltre aggiungere un parametro che ci permette di inserire una descrizione del cambiamento che stiamo registrando. Per farlo basta specificare l’opzione -m seguita dalla descrizione tra virgolette.

Ora abbiamo creato il nostro primo commit!

Possiamo vedere che il commit ha riportato alcune informazioni sull'operazione. Indica infatti:

- a quale ramo abbiamo affidato il commit (in questo caso, master)
- quale checksum (versione) ha il commit
- quanti file sono stati modificati
- le statistiche sulle linee aggiunte e rimosse nel commit

Se volessimo vedere i vari stati del progetto che sono stati registrati ci basterà usare il comando *git log* che restituirà un output con gli ultimi commit fatti nel repository. Ecco un esempio:

Queste stringhe indicano il codice del commit, il nome dell'autore, la sua mail, la data di scrittura e la descrizione inserita dall'autore del commit.

Rimuovere la tracciatura di un file

Abbiamo visto come si può tracciare un file con il comando *git add* ma se volessimo rimuovere la tracciatura, cosa dobbiamo fare? Semplice, inserire il comando:

che permette di rimuovere un file da quelli tracciati.

Ignorare i file

Spesso, si ha una classe di file che si vogliono nascondere a Git. In questi casi, potete creare un file chiamato .gitignore con una lista di file o pattern che non devono essere mostrati a Git. Questo è un file d'esempio:

La prima linea dice di ignorare tutti i file con estensione .log, la seconda linea di ignorare tutti i files chiamati .DS_Store e infine l'ultima riga dice di ignorare tutti i files presenti all'interno della cartella /app/cache/.

Conclusioni

In questa seconda guida abbiamo imparato a utilizzare i comandi di base di Git, abbiamo visto come tracciare i file e come registrare uno stato del repository nello storico di Git.

Nella prossima guida capiremo come usare Git come strumento di collaborazione online: come lavorare su più rami, come pubblicare il nostro codice, come scaricare gli aggiornamenti e così via.

Infine nell’ultima puntata della guida vedremo un esempio pratico di come fare per pubblicare il codice su Github.

INTRODUZIONE A GIT: INDICE LEZIONI

- Cos’è e come installarlo
- Quali sono le funzioni principali di Git
- Usare Git come strumento di collaborazione online