Simone Di Meglio - Full Stack Web Dev

GitHub for breakfast

Capitolo 0

Introduzione

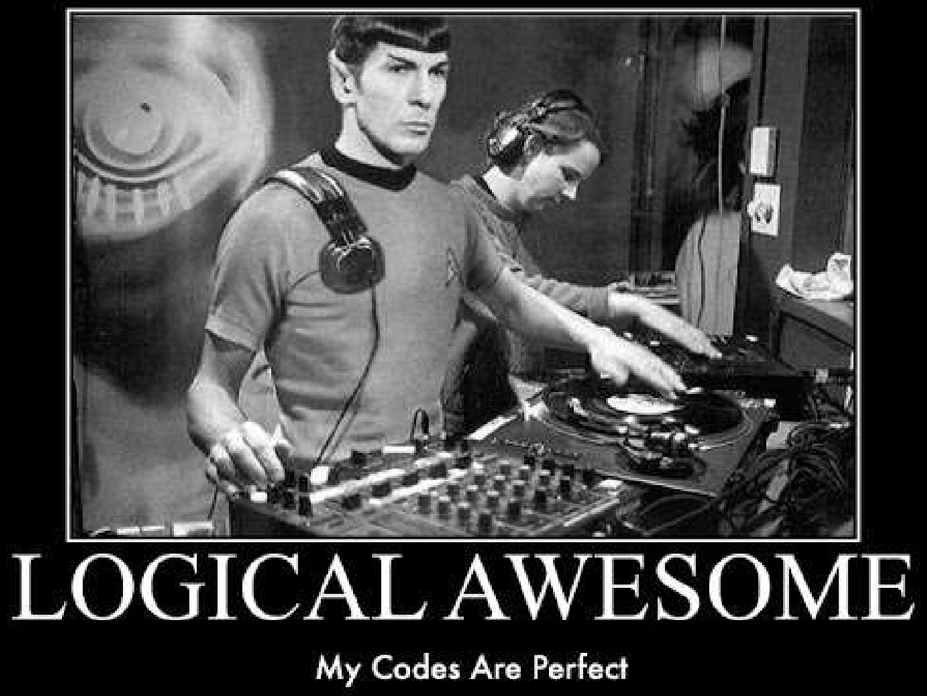
Introduzione

GitHub è una piattaforma di hosting di repository Git, fondata nel 2008.

La storia di GitHub è un racconto affascinante di innovazione, collaborazione e impatto nel mondo dello sviluppo software.

Tutto ebbe inizio nel 2005, quando Tom Preston-Werner, Chris Wanstrath e PJ Hyett crearono una piccola startup chiamata **Logical Awesome LLC.**

All'epoca, lavoravano su vari progetti di software e si resero conto dell'importanza di un sistema di controllo delle versioni per gestire in modo efficace il codice sorgente dei loro progetti.



Iniziarono a lavorare su un'idea che avrebbe rivoluzionato il modo in cui gli sviluppatori collaboravano e condividevano il loro lavoro.

Nel 2007, lanciarono la prima versione di GitHub, una piattaforma di hosting di repository Git che semplificava notevolmente la gestione del codice sorgente e la collaborazione tra gli sviluppatori.

GitHub ha rapidamente guadagnato popolarità tra gli sviluppatori grazie alla sua interfaccia intuitiva, alle potenti funzionalità di collaborazione e alla capacità di ospitare facilmente progetti open source.

La piattaforma ha contribuito a cambiare il modo in cui gli sviluppatori lavorano insieme, incoraggiando la trasparenza, la condivisione del sapere e la creazione di comunità di sviluppatori in tutto il mondo.

Nel corso degli anni, GitHub ha continuato a crescere e a evolversi, introducendo nuove funzionalità e strumenti per migliorare l'esperienza degli sviluppatori.

Nel 2018, GitHub è stata acquisita da Microsoft per 7,5 miliardi di dollari, confermando il suo status come una delle piattaforme più importanti nel mondo dello sviluppo software.

Oggi, GitHub conta **milioni di utenti in tutto il mondo**, da sviluppatori indipendenti a grandi aziende, e ospita una vasta gamma di progetti open source in tutti i settori.

La piattaforma continua a essere al centro dell'innovazione nel mondo dello sviluppo software, fornendo agli sviluppatori gli strumenti di cui hanno bisogno per creare, condividere e collaborare sui loro progetti.

Tenetevi pronti, cinture allacciate.

Let’s go.

Capitolo 1

Lista dei Comandi

Iniziamo:

git init: Inizia un nuovo repository Git nella directory corrente.

git add <file>: Aggiunge un file alla "staging area", ossia alla lista dei file pronti per il commit.

git commit –m "messaggio": Registra le modifiche ai file nella "staging area" e le memorizza nel repository con un messaggio descrittivo.

git status: Mostra lo stato attuale del repository, inclusi i file modificati, i file nella "staging area" e altro ancora.

git push: Invia i commit locali al repository remoto su GitHub.

git pull: Scarica le modifiche dal repository remoto e le integra nel repository locale.

git clone <URL>: Crea una copia locale di un repository remoto su GitHub.

git branch: Mostra l'elenco dei rami nel repository locale.

git branch <nome_del_ramo>: Crea un nuovo ramo nel repository locale.

git checkout <nome_del_ramo>: Passa a un altro ramo nel repository locale.

git merge <nome_del_ramo>: Unisce le modifiche da un ramo all'altro. Solitamente si utilizza per unire il lavoro di un ramo di sviluppo al ramo principale (come "master" o "main").

Capitolo 2

I grandi classici

Tipo “Il Gladiatore”, o quasi..

Questi sono da sapere, non si scappa.

COMMIT, PULL & PUSH!

Il comando **commit** viene utilizzato per registrare le modifiche fatte ai file nel repository locale. Quando esegui un commit, devi fornire un messaggio che descriva le modifiche apportate. Per fare un commit, segui questi passaggi.

Aggiungi i file che desideri includere nel commit alla "staging area" utilizzando il comando:

```
git add <file>
```

Esegui il commit utilizzando il comando

```
git commit -m "messaggio di commit"
```

Sostituisci "messaggio di commit" con una breve descrizione delle modifiche apportate.

Il comando **push** viene utilizzato per inviare i commit locali al repository remoto su GitHub (o altro host Git). Questo è particolarmente importante se vuoi condividere il tuo lavoro con altri membri del team o semplicemente mantenere una copia di backup del tuo lavoro online. Assicurati di essere sul ramo da cui desideri fare il push (ad esempio, main o master) utilizzando il comando

```
git checkout <nome_ramo>
```

Esegui il push utilizzando il comando:

```
git push origin <nome_ramo>
```

Ovviamente, sostituisci <nome_ramo> con il nome del ramo su cui stai lavorando.

Il comando **pull** viene utilizzato per scaricare le modifiche dal repository remoto e integrarle nel repository locale. Questo è utile quando vuoi aggiornare il tuo repository locale con le modifiche fatte da altri collaboratori. Come sempre, se necessario, prima esegui un bel:

```
git checkout <nome_ramo>
```

Esegui il pull utilizzando il comando

```
git pull origin <nome_ramo>
```

Anche in questo caso, sostituisci <nome_ramo> con ciò che ti serve.

Capitolo 3

Creare un branch

Con calma..

Per creare un nuovo branch e portare con te tutte le modifiche attuali presenti nel ramo principale (solitamente chiamato "main" o "master"), segui questi passaggi:

Assicurati di essere sul ramo principale eseguendo:

```
git checkout main
```

IMPORTANTE:

Aggiorna il ramo principale scaricando eventuali modifiche dal repository remoto:

```
git pull origin main
```

Ora sei sul ramo principale e hai tutte le modifiche più recenti.

A questo punto, puoi creare un nuovo ramo con il comando:

```
git checkout -b nuovo_ramo
```

Questo comando crea un nuovo ramo chiamato "nuovo_ramo"
e ti sposta su di esso.

Ora sei sul nuovo ramo e puoi iniziare a lavorare sulle **tue** modifiche.

Tutte le modifiche fatte su questo ramo saranno separate dal ramo principale
finché non le unirai.

Ricorda che se vuoi condividere il tuo nuovo ramo con gli altri dev che partecipano al progetto, dovrai eseguire:

```
git push origin nuovo_ramo
```

Questo comando invierà il tuo nuovo ramo al repository remoto su GitHub. Una volta che hai finito di lavorare sul tuo nuovo ramo e sei pronto per integrare le modifiche nel ramo principale, puoi fare un "**merge**" ma assicurati di avere una visione **chiara** di come le modifiche influiranno sul progetto prima di eseguire questa operazione.

Capitolo 4

Merge: how to

Un passo per volta.

Vediamo passo per passo, non abbiate paura.

Invio del nuovo ramo al repository remoto su GitHub
(NB: Questo passaggio l'hai visto come ultima slide del capitolo precedente!)

Dopo aver lavorato sul tuo nuovo ramo localmente e aver completato le modifiche che desideri, è ora il momento di caricare questo ramo sul repository remoto su GitHub in modo che altri possano vederlo e collaborare se necessario.

```
git push origin nuovo_ramo
```

Questo comando invierà il tuo nuovo ramo chiamato "nuovo_ramo" al repository remoto su GitHub (sostituisci "nuovo_ramo" con il nome del tuo ramo effettivo).

Merge delle modifiche nel ramo principale:

Una volta che hai completato tutte le modifiche sul tuo nuovo ramo e sei sicuro di voler integrare queste modifiche nel ramo principale (come "main" o "master"), è necessario fare un "merge".

Assicurati di essere sul ramo principale utilizzando il comando:

```
git checkout main
```

Poi, fai il "merge" del tuo nuovo ramo nel ramo principale con il comando:

```
git merge nuovo_ramo
```

Questo comando combinerà tutte le modifiche fatte nel tuo nuovo ramo con il ramo principale.

È importante assicurarsi di comprendere appieno le modifiche che hai apportato nel tuo nuovo ramo prima di unire i tuoi cambiamenti al ramo principale.

In caso di dubbi, controlla le modifiche apportate e assicurati che siano corrette prima di eseguire il merge.

Una volta che il merge è completato con successo, le modifiche sul tuo nuovo ramo saranno ora integrate nel ramo principale.

Fine