Fondamenti di Git

Corso di Laboratorio di
Gestione progetto e organizzazione di impresa

Prof.ssa De Vincentiis, Anna

Prof. Dei Rossi, Leonardo Essam

Version Control System (VCS) (1)

Un **sistema di versioning** è un software che tiene traccia di tutte le modifiche[1] fatte ai file di un progetto e permette di tornare indietro a versioni precedenti se qualcosa non funziona[2].

Aiuta più persone a lavorare sugli stessi file senza perdere il lavoro degli altri.

Senza versioning il rischio è di sovrascrivere i file o di perdere il lavoro (in tutti i sensi!).

Con il versioning ogni modifica ha: **un autore**, **una data** e **uno storico di modifiche**.

[1] Vita, morte e miracoli
[2] Macchina del tempo

Git (1)

Git è un software per il controllo di versione distribuito utilizzabile da interfaccia a riga di comando (command prompt).

Git[3] nasce per essere uno strumento semplice per facilitare lo sviluppo del kernel Linux. La sua progettazione si ispirò ad analoghi strumenti (al tempo, software proprietari) come BitKeeper e Monotone.

È stato originariamente creato da Linus Torvalds (il padre di Linux, ndr) e pubblicato per la prima volta l'8 aprile 2005.

Git è disponibile per GNU/Linux, macOS, Windows e Solaris.

[3] Nello slang britannico, *Git* significa *idiota*

Git (2)

Git è un sistema distribuito.

Questo significa che ogni sviluppatore ha una **copia completa** del repository (codice + cronologia).

Il vantaggio è che non esiste un *single point-of-failure*!

Esistono altri sistemi VCS che sono invece centralizzati: ovvero che il codice è ospitato su un unico server, ma non li tratteremo in questo corso.

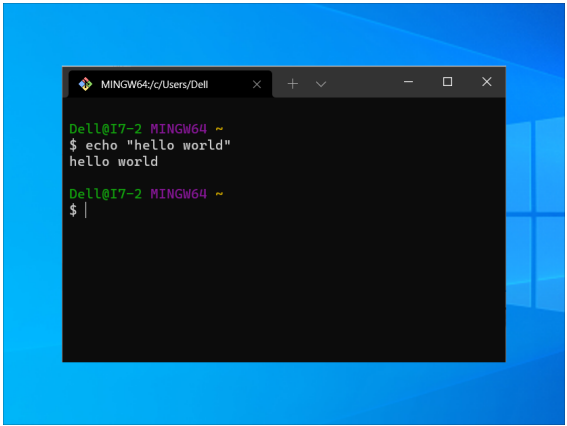
Buonarroti

Primo esercizio con Git

Configurazione iniziale (1)

Per aprire la console di Git, dal menu Start cercare "Git Bash".

Si aprirà una finestra di terminale simile a questa:



Configurazione iniziale (2)

Per personalizzare il nostro profilo locale[4] bisogna utilizzare i seguenti comandi:

```
git config --global user.name "<nome cognome>"
git config --global user.email "<indirizzo e-mail>"
```

Ad esempio:

```
git config --global user.name "Leonardo Essam Dei Rossi"
git config --global user.email "leonardo.deirossi@buonarroti.tn.it"
```

[4] Succesivamente ci collegheremo a GitHub (!)

Creazione del repository (1)

Una volta entrati nella cartella desiderata[5] (vuota o meno che sia), per inizializzare un nuovo *repository* vale il comando:

```
git init
```

Il terminale restituirà in output:

```
leode@ThinkBook-14:~/prova$ git init
Initialized empty Git repository in /home/leode/prova/.git/
```

[5] Ricorda: `cd ...`

Creazione del repository (2)

Per creare, modificare ed eliminare i file ci sono due metodi:

- Dal terminale (Vim, etc.);
- Tramite un IDE (Visual Studio Code, etc.).

Per comodità consiglio di utilizzare Visual Studio Code (link).

Il primo commit (1)

Solitamente, il primo file che viene creato è il file di *read-me*[6].

Da Visual Studio Code create il file `README.md` con il seguente contenuto:

```
1  # Il mio primo progetto
2  ## Ciao, Mondo!
```

L'estensione `.md` indica che il file è un file di tipo Markdown[7] (un particolare modo per formattare i documenti).

[6] Dall'inglese: "leggimi"
[7] Markdown, From Wikipedia, the free encyclopedia (link)

Il primo commit (2)

Per *registrare* il nuovo file all'interno dello storico di Git, bisogna utilizzare il seguente comando:

```
git add README.md
```

Osservazione: in caso si abbiano file multipli e si voglia inserirli tutti contemporaneamente, vale il comando `git add .`

Il primo commit (3)

Una volta selezionato/i il/i file desiderato/i, possiamo registrare nella *tree* del progetto con il comando:

```
git commit -m "<messaggio>"
```

Il terminale restituirà un messagio simile a questo:

```
leode@ThinkBook-14:~/prova$ git add README.md
leode@ThinkBook-14:~/prova$ git commit -m "Aggiunto file: README.md"
[master (root-commit) efeb859] Aggiunto file: README.md
 1 file changed, 3 insertions(+)
 create mode 100644 README.md
```


Buonarroti