

**Formazione  
specialistica  
per sviluppatore  
front-end**





# Hello!

## Alessandro Pasqualini

Full-stack developer, appassionato di  
programmazione e sviluppo software in generale, amo  
le serie tv e i gatti

**1**

# **Versionamento del software: GIT**



# Cos'è il versionamento

I sistemi di controllo di versione sono strumenti software che permettono ad un team di gestire modifiche al codice durante un periodo di tempo (tipicamente la vita dell'intero progetto).

Questi sistemi tengono traccia di ogni modifica apportata al codice in uno speciale database.

Se viene commesso un errore, i developer possono "riportare indietro l'orologio" e comparare vecchie versioni del codice senza causare ulteriori problemi.



# Versionamento **NON** è backup

L'idea fondamentale del controllo di versione è mantenere diverse revisioni della stessa unità di informazione (es. codice).

L'idea di backup è mantenere al sicuro l'ultima versione dell'informazione (es. codice). Le vecchie versioni possono essere sovrascritte o eliminate.



# Perchè usare un VCS?

VCS significa Version Control System.

Quando più programmatori lavorano sullo stesso codice (codebase) succede spesso di eseguire modifiche nello stesso file (magari sulla stessa linea di codice).

Ad esempio lo "sviluppatore 1" che lavora sulla "feature 1" potrebbe apportare alcune modifiche e scoprire in seguito che lo "sviluppatore 2" che lavora sulla "feature 2" ha eseguito modifiche contrastanti alle sue.



# Perchè usare un VCS?

Prima della nascita dei VCS era un incubo!

Il "programmatore 1" eseguiva le modifiche al codice per "feature 1" e doveva passare il codice modificato al "programmatore 2" in modo che potesse sviluppare la sua "feature 2" sul codice del "programmatore 1".

Se si dimenticava di avvisare il "programmatore 2" delle sue modifiche quest'ultimo poteva sovrascrivere, senza saperlo, il lavoro del "programmatore 1".



# Perchè usare un VCS?

Se due programmatore lavoravano senza saperlo sullo stesso file successivamente dovevano fare il "merge" delle modifiche manualmente (sempre se si ricordavano di avvisarsi a vicenda).

Era difficilissimo quindi lavorare in team ad uno stesso progetto!

Da questi problemi (ed altri) nasce il concetto di versionamento del codice. Si capisce quindi che la funzione "backup" è solo un effetto collaterale (benevolo) dell'utilizzo di questi software.



# Perchè usare un VCS?

Un VCS permette di salvaguardare le modifiche al codice in caso di conflitto (stesso file e stessa riga modificata da più persone).

Permette anche di:

- Agevolare il lavoro in team
- Conservare più versioni dello stesso codice
- Semplificare il "merge" del lavoro fatto da programmatore diversi
- "Aprire le porte" allo sviluppo moderno del software (modern software development)



# Git

Git è il più usato software di versionamento del codice, creato nel 2005 da Linus Torvalds (il creatore di Linux).

Git è usato da alcuni dei più famosi progetti al mondo: il kernel Linux né è un esempio.

Git è open source ed è liberamente (e gratuitamente) utilizzabile.



“

*I'm an egotistical bastard, and I name all my projects after myself. First Linux, now git*

- Linus Torvalds



The name "git" was given by Linus Torvalds when he wrote the very first version. He described the tool as "the stupid content tracker" and the name as (depending on your mood):

- random three-letter combination that is pronounceable, and not actually used by any common UNIX command. The fact that it is a mispronunciation of "get" may or may not be relevant.
- stupid. contemptible and despicable. simple. Take your pick from the dictionary of slang.
- "global information tracker": you're in a good mood, and it actually works for you. Angels sing, and a light suddenly fills the room.
- "goddamn idiotic truckload of sh\*t": when it breaks



# Github

Github è un servizio di hosting per progetti che usano Git per il versionamento del codice.

E' usato principalmente per contenere il codice di progetti open source (anche se è possibile creareni di provati).

E' gratuito e mette a disposizione di ogni progetto circa 1GB di spazio. Naturalmente dobbiamo versionare solo codice (e qualche asset) e quindi 1GB è abbastanza per praticamente tutti i progetti.



The screenshot shows the GitHub homepage with a dark background featuring a faint circuit board pattern. On the left, the GitHub logo and navigation links like 'Why GitHub?', 'Enterprise', 'Explore', 'Marketplace', and 'Pricing' are visible. On the right, there's a search bar, a 'Sign in' button, and a 'Sign up' button. A large white call-to-action box is centered on the page. It contains fields for 'Username' (with the value 'alessandro.pasqualini.1105@gmail.com'), 'Email' (an empty field), and 'Password' (a field with several dots). Below the password field is a note: 'Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.' A green 'Sign up for GitHub' button is at the bottom of the box. At the very bottom of the page, small text states: 'By clicking "Sign up for GitHub", you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account related emails.'

**Built for developers**

GitHub is a development platform inspired by the way you work. From **open source** to **business**, you can host and review code, manage projects, and build software alongside 40 million developers.

Search GitHub

Sign in

Sign up

Username

alessandro.pasqualini.1105@gmail.com

Email

Password

.....

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more](#).

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account related emails.



# Bitbucket

Bitbucket è un altro hosting di repository e funziona in maniera identica a Github.

Principalmente è preferito dalle aziende perché originariamente permetteva di creare progetti privati gratuitamente, al contrario di Github dove erano a pagamento.

Da quando Microsoft ha comprato Github ha reso gratuiti i progetti privati anche su Github.

[Why Bitbucket](#) ▾[Product Guide](#) ▾[Self-Hosted](#)[Pricing](#)[Log in](#)[Get started](#)

12 days of CI/CD celebrates all things CI/CD with new feature announcements, use cases, competitions, and more. [Learn more](#)

[X](#)

## Built for professional teams

Bitbucket is more than just Git code management. Bitbucket gives teams one place to plan projects, collaborate on code, test, and deploy.

[Get started for free](#)

Or host it yourself with [Bitbucket Data Center](#) →

The screenshot shows a pull request titled "bugfix/123-bug remove extra padding". The status bar indicates "bugfix/123-bug" is merging into "master" and the PR is "OPEN". The message says: "I made a few changes to tidy up the code. Please let me know if all looks good!" Three reviewers are listed with their profile icons. A code diff for "js / core.js" shows a single line of code: "12 - // Takes an input date string and related date format". There are "Approve" and "Merge" buttons at the top right of the PR card.



# Repository

Il concetto di repository è centrale e fondamentale quando si parla di VCS.

Un repository è simile ad una cartella del pc dove sono contenuti tutti i file del progetto e tutte le loro revisioni.

Una modifica ad un file in terminologia VCS è chiamata revisione.



Screenshot of the GitHub repository `octocat / Hello-World`. The repository has 1.7k stars, 214 pull requests, and 1.4k forks.

Key statistics:

- 3 commits
- 3 branches
- 0 packages
- 0 releases
- 2 contributors

Latest commit: `7fd1a60` on 7 Mar 2012 by `octocat`. Merge pull request #6 from Spaceghost/patch-1.

README file content:

```
Hello World!
```

Footer:

© 2020 GitHub, Inc. Terms Privacy Security Status Help

Contact GitHub Pricing API Training Blog About



# Repository

Il concetto di repository è centrale e fondamentale quando si parla di VCS.

Un repository è simile ad una cartella del pc dove sono contenuti tutti i file del progetto e tutte le loro revisioni.

Una modifica ad un file in terminologia VCS è chiamata revisione.

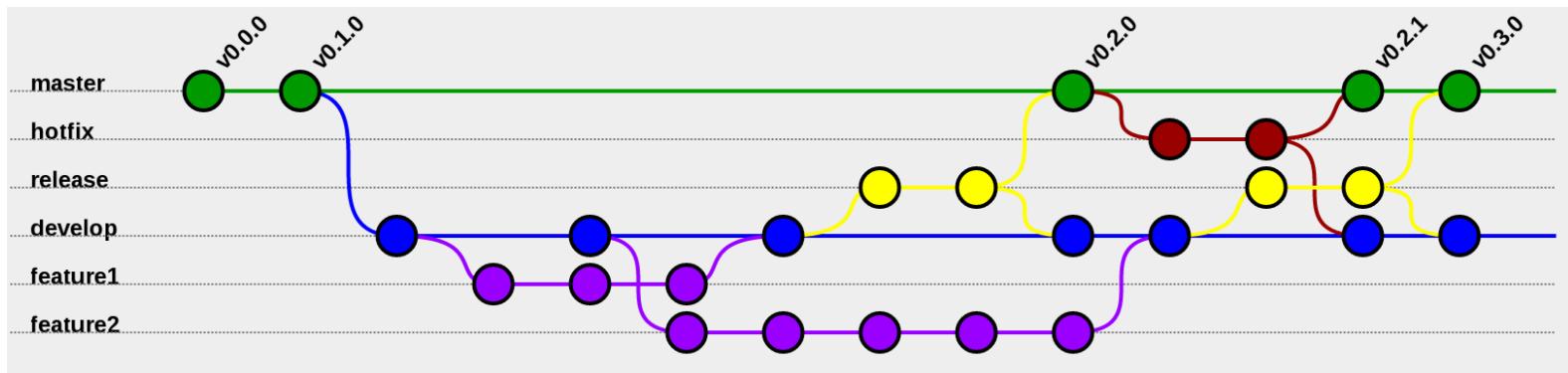


# Commit

Quando si apportano delle modifiche a dei file del progetto è necessario salvarle e darci una descrizione.

Nella terminologia Git, salvare le modifiche equivale a create un **commit** (dall'inglese commettere).

Quindi un commit è un insieme di modifiche apportate ad uno o più file, insieme ad un messaggio testuale che descrive le modifiche fatte. (Il messaggio è obbligatorio).



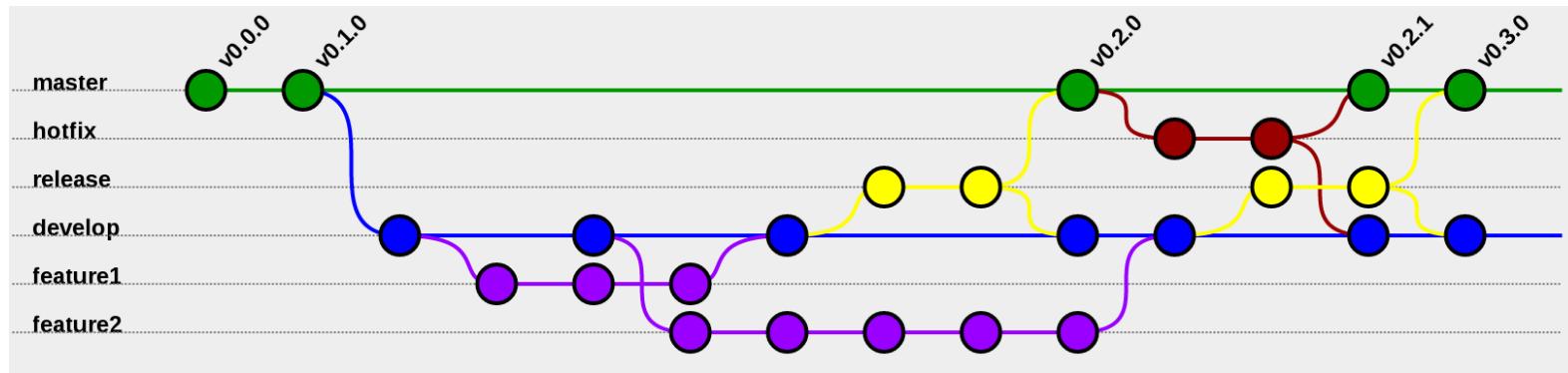


# Branch

Dall'inglese significa ramificazione.

Durante lo sviluppo è possibile che sia necessario intraprendere diverse strade di sviluppo: pensate allo sviluppo di funzionalità diverse.

In Git un branch è una ramificazione dello sviluppo, una strada alternativa di sviluppo.





# Branch

Nei repository esiste sempre un branch "speciale" chiamato master che rappresenta la linea di sviluppo principale.

Tutte i branch (ramificazioni del codice) possono successivamente essere inclusi nel branch master (quello principale), abbandonate e quindi eliminate dal repository, etc.

Nei casi più semplici si usa solamente il branch master.



# "Clonare" un repository

Quando si lavora con un repository è necessario "scaricarene" una copia in locale in modo da poterci lavorare.

Fare un "clone" di un repository equivale a scaricare una copia locale dello stesso, mantenendo un collegamento alla "sorgente" dalla quale è stato scaricato.

Il comando da eseguire è:

```
git clone url_del_repository
```



# Es: "Clonare" un repository

Cloniamo il nostro primo repository:

```
git clone https://github.com/howtosrc/hello-world.git
```

Git creerà una cartella chiamata con lo stesso nome del repository in cui scaricherà tutto il suo contenuto (e tutte le revisioni dei file).



# ES: "Clonare" un repository

```
[MacBook-Pro-di-Alessandro-2:repo alessandro$ git clone https://github.com/howtos]
rc/hello-world.git
Cloning into 'hello-world'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
MacBook-Pro-di-Alessandro-2:repo alessandro$ █
```



hello-world



.git



README.md



# Aggiungere un file al repository

Per aggiungere un file al repository:

```
git add nome_del_file
```

Il file è semplicemente stato aggiunto, ma non è ancora parte di un commit, quindi le sue modifiche non sono tracciate.

Aggiungere un file significa dire a Git che quel file verrà inserito all'interno di un commit a breve.



# Es: Aggiungere un file al repository

Creiamo il file vuoto file1.txt e successivamente lo aggiungiamo con:

```
git add file1.txt
```

Il file è semplicemente stato aggiunto, ma non è ancora parte di un commit, quindi le sue modifiche non sono tracciate.

Aggiungere un file significa dire a Git che quel file verrà inserito all'interno di un commit a breve.



# Vedere lo stato del repository

Per vedere lo stato del repository e quindi conoscere quali file sono stati modificati, aggiunti, cancellati, etc, ma non committati:

```
git status
```

Git ci mostra lo stato corrente e quindi solo quello che è stato modificato dal commit precedente (le modifiche precedenti).



# Es: Vedere lo stato del repository

```
[MacBook-Pro-di-Alessandro-2:hello-world alessandro$ git status  
On branch master  
Your branch is up to date with 'origin/master'.  
  
Changes to be committed:  
(use "git reset HEAD <file>..." to unstage)  
  
      new file:   file1.txt
```

```
MacBook-Pro-di-Alessandro-2:hello-world alessandro$ █
```



.git



README.md



file1.txt



# Es: Creare il nostro primo commit

Creiamo un commit che contiene il nostro nuovo file file1.txt.

```
git commit -m "Aggiunto il file file1.txt"
```

Il messaggio deve essere una descrizione, seppur concisa, delle modifiche contenute nel commit (le modifiche fatte ai file che inseriamo nel commit).

Sono PROIBITI messaggi inutili o "tanto perché è obbligatorio il messaggio"



# Es: Creare il nostro primo commit

```
git commit -m "Aggiunto il file file1.txt"
```

Il messaggio deve essere una descrizione, seppur concisa, delle modifiche contenute nel commit (le modifiche fatte ai file che inseriamo nel commit).

Sono PROIBITI messaggi inutili o "tanto perché è obbligatorio il messaggio"



# Es: Creare il nostro primo commit

```
[MacBook-Pro-di-Alessandro-2:hello-world alessandro$ git commit -m "Aggiunto il file file1.txt"
[master 66d8654] Aggiunto il file file1.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 file1.txt
MacBook-Pro-di-Alessandro-2:hello-world alessandro$ ]
```





# Caricare il commit su Github

Una volta che abbiamo creato il nostro commit dobbiamo caricarlo online, ad esempio su Github in modo che le altre persone possano scaricarlo.

```
git push
```

Se non abbiamo commit da caricare Git ci dice che non ha apportato modifiche al repository online.



# Es: Caricare il commit su Github

```
[MacBook-Pro-di-Alessandro-2:hello-world alessandro$ git push
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 301 bytes | 301.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/howtosrc/hello-world.git
  61e7ca8..66d8654  master -> master
MacBook-Pro-di-Alessandro-2:hello-world alessandro$ ]
```





# Es: Github prima del push

howtosrc / hello-world

Unwatch ▾ 2 Star 0 Fork 0

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security Insights Settings

No description, website, or topics provided. Edit

Manage topics

1 commit 1 branch 0 packages 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download ▾

alessandro1105 Initial commit Latest commit 61e7ca8 20 minutes ago

README.md Initial commit 20 minutes ago

README.md

hello-world



# Es: Github dopo il push

howtosrc / hello-world

Unwatch ▾ 2 Star 0 Fork 0

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security Insights Settings

No description, website, or topics provided. Edit

Manage topics

2 commits 1 branch 0 packages 0 releases 1 contributor

Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾

alessandro1105	Aggiunto il file file1.txt	Latest commit 66d8654 5 minutes ago
README.md	Initial commit	22 minutes ago
file1.txt	Aggiunto il file file1.txt	5 minutes ago

README.md

hello-world



# Git step by step

Primo passo obbligatorio è clonare il repository, altrimenti non abbiamo il codice su cui lavorare!

Successivamente questi step si ripetono a ciclo:

1. Modificare, creare, eliminare file (quindi fare il lavoro sul codice)
2. Aggiungere a Git i file modificati da includere nel commit
3. Creare il commit insieme al messaggio di descrizione
4. Fare il push del commit
5. Tornare al punto (1) e ricominciare il ciclo



# Thanks!

## Any questions?

Per qualsiasi domanda contattatemi:  
[alessandro.pasqualini.1105@gmail.com](mailto:alessandro.pasqualini.1105@gmail.com)