

Git per piccoli team e progetti

Una doverosa premessa



Non ho alba di quello di cui sto parlando...

Davvero!



*Git permette di tenere traccia
dei **cambiamenti** nel tuo
codice in modo distribuito
con supporto multi utente,
workflow non lineari e
senza accesso alla rete.*

Perché è essenziale?

I problemi capitano,
ma con **git** si risolvono ;)



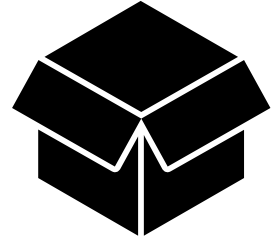
**SHIT HAPPENS
GIT OVER IT**

3-2-1 Backup Rule



- Almeno **3 copie indipendenti** del tuo codice
- Salva le copie su **2 differenti tipi di media**
- Tieni almeno **1 copia** del backup offsite

3-2-1 Backup Rule

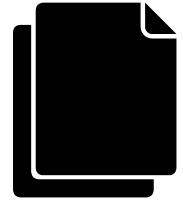


- **3 copie:** git locale + 2 git remoti
- **2 media:** hard disk e cloud
- **1 offsite:** BitBucket/GitHub + Amazon CodeCommit



Concetti Base

- **Repository:** archivio con i cambiamenti (locale e remoto)
- **Branch:** una serie di cambiamenti raggruppati per «*nome*»
- **Staging:** area temporanea dove ospitare le modifiche
- **Commit:** salvataggio dei cambiamenti in un branch
- **.gitignore:** file per ignorare contenuti (*create-gitignore.bat*)



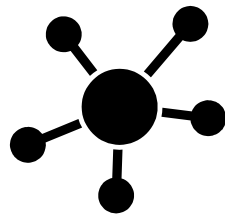
Struttura Cartella

- /
- /.gitignore
- /data (*ignored*)
- *File e cartelle con i sorgenti*



Utilizzo Base (Locale)

- Un solo branch (master)
- `git add .`
- `git tag -a 1.0.1 -m "Nuova feature"`
- `git commit -m "Nuova feature"`
- `git checkout 1.0.1`



Utilizzo Base (Remoto)

- *Tutto come in locale*
- `git push origin master`
- `git pull origin master`

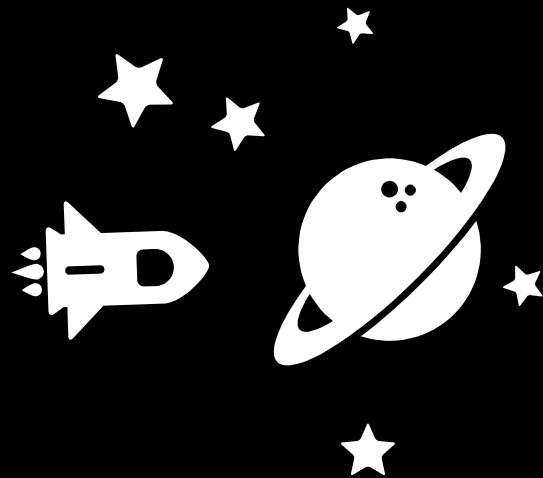


Comandi Essenziali

- `git status` Visualizza i file diversi dal commit corrente
- `git log` Visualizza l'elenco dei commit
- `git checkout` Rende attivo un branch o un commit
- `git add .` Effettua lo staging di tutti i file modificati
- `git commit -m "Hello world!"` Esegue il commit dei file
- `git stash / git stash drop` Mette da parte e scarta

Utilizzo in Team

La storia si complica...

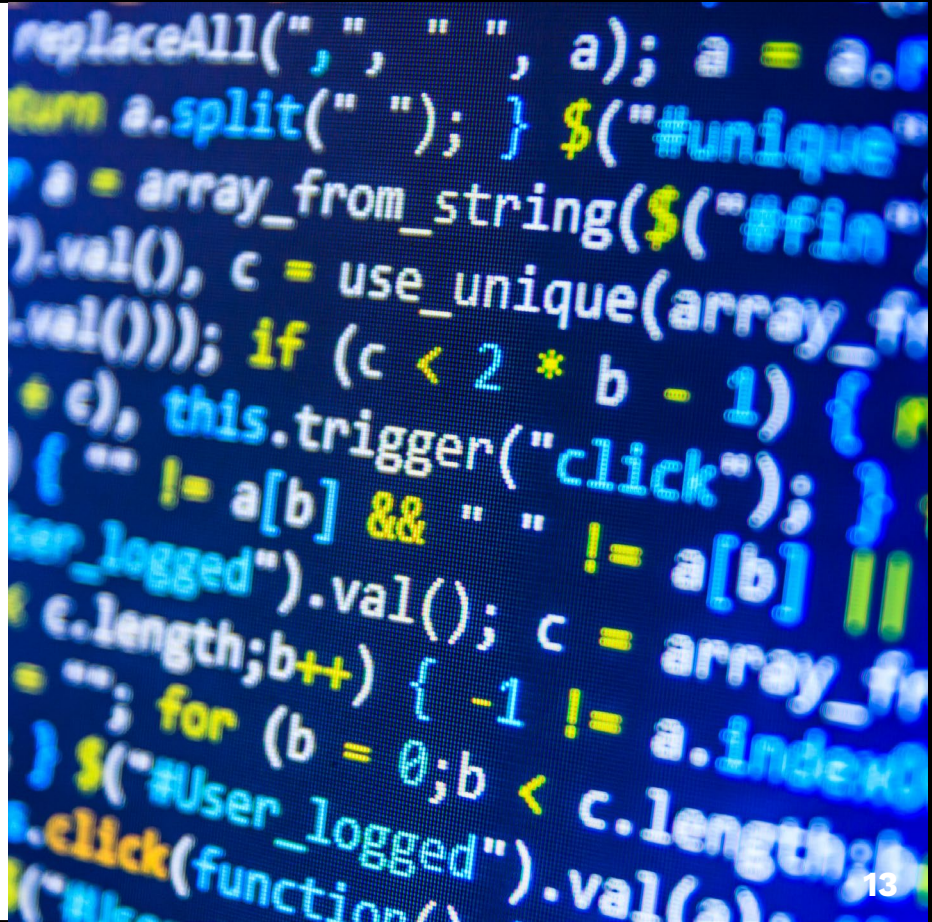


Git!

Potente ma complesso

Saper usare bene git richiede tempo (mille comandi, opzioni e workflow).

Ma c'è una scorciatoia...



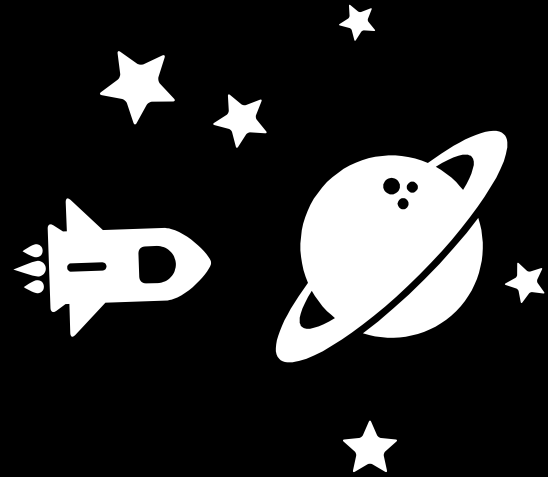


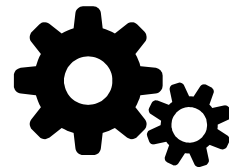
Concetti Chiave

- Due branch long-term: **master** e *develop*
- Lo sviluppo si fa in *develop*
- Le release ufficiali in **master** (con tag di versione)
- Branch «*volatili*» (*develop*) per nuove funzioni
- Branch «*volatili*» (*master*) per bugfix in produzione

Ogni Commit

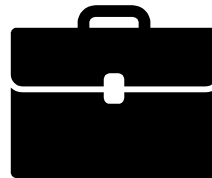
Sul branch **master** equivale a una release di produzione con relativa descrizione e numero di versione





Workflow Sviluppo

- Sviluppare sul branch *develop* locale
- Eseguire il fetch delle modifiche altrui da *origin/develop*
- Creare un branch «*volatile*» con tali modifiche
- Effettuare il merge con il branch *develop* locale
- Fare il push del *develop* locale su *origin/develop* remoto

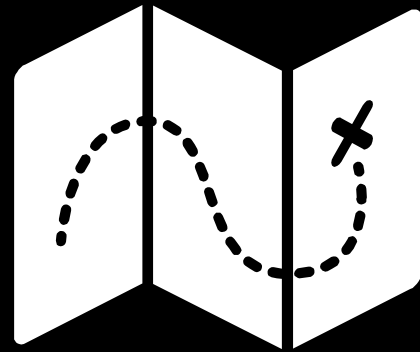


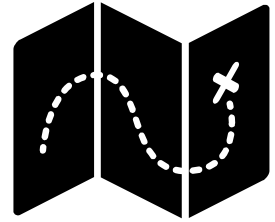
Workflow Release

- Eseguire il fetch delle modifiche altrui da **origin/master**
- Creare un branch «*volatile*» con le modifiche
- Effettuare il merge con il branch **master** locale
- Compiere il merge dal *develop* locale al **master** locale
- Fare il push del **master** locale su **origin/master** remoto

La Scorciatoia

Come aggirare la **complessità** di git
senza perdere in potenza e flessibilità





Creare e Usare Script Batch

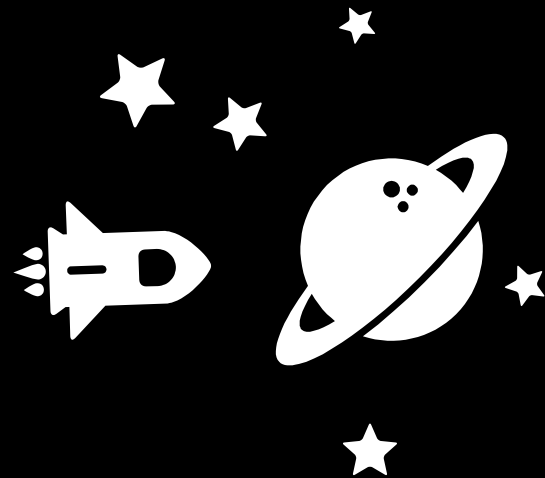
Ridurre la complessità automatizzando i principali task



*Don't reinvent the
wheel, just realign it.*

Anthony J. D'Angelo

Script Batch



Se si seguono i due workflow
la maggior parte delle interazioni
con **git** puo' essere gestita con script!

Script Batch: Setup Ambiente



setup-repository.bat

Setup del repository locale con i branch master e develop e un'origine remota. Tutti i file della cartella corrente vengono aggiunti al repository.

clone-repository.bat

Clone di un repository remoto con i branch master e develop e un'origine remota. La cartella corrente deve essere vuota per ricevere il repository.

Script Batch: Workflow Sviluppo



create-develop-feature-branch.bat

Crea un feature branch a partire dal branch develop. Utile per bugfix e sviluppo nuove feature.

commit-to-current-branch.bat

Aggiunge tutti i file ed effettua il commit al branch locale. Preferibilmente da evitare nel branch **master**.

Script Batch: Workflow Sviluppo



fetch-develop.bat

Esegue il fetch di *origin/develop*. Se viene passato il nome di un branch (es. *working-develop*) come parametro, le modifiche di terzi verranno salvate in questo branch per revisione e merge.

merge-to-develop.bat

Effettua il merge di un feature branch sul branch *develop*. Se viene passato il parametro «*delete*», il feature branch verrà cancellato dopo il merge.

Script Batch: Workflow Sviluppo



delete-feature-branch.bat

Elimina un feature branch.
Include un filtro per evitare di
cancellare per errore i branch
master e *develop*.

push-develop.bat

Effettua il push remoto
dell'ultimo commit eseguito sul
branch *develop*.

Script Batch: Workflow Release



create-master-bugfix-branch.bat

Crea un bugfix branch a partire dal branch **master**. Utile per risolvere bug in produzione.

fetch-master.bat

Esegue il fetch di **origin/master**. Se viene passato il nome di un branch (es. *working-master*) come parametro, le modifiche di terzi verranno salvate in questo branch per revisione e merge.

Script Batch: Workflow Release



merge-to-master.bat

Effettua il merge di un bugfix branch sul branch **master**. Se viene passato il parametro «*delete*», il bugfix branch verrà cancellato dopo il merge.

build-master-release.bat

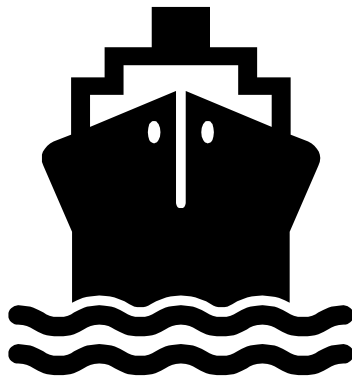
Crea un file di versione, aggiunge tutti i file ed effettua il commit a **master** con un tag di versione. Infine esegue il push remoto verso **origin/master**.

Script Batch: Workflow Release



merge-master-to-develop.bat

Effettua il merge di **master** sul branch *develop* per allineare lo sviluppo con l'ultima release. Va eseguito subito dopo la creazione di una release da parte di tutti gli sviluppatori in modo da evitare la divergenza dei sorgenti.

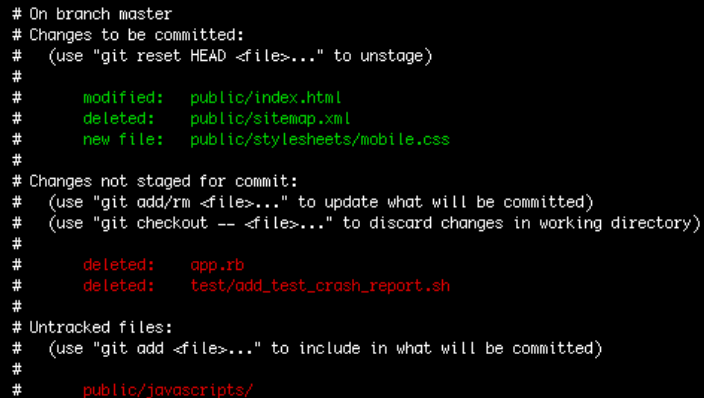


github.com/filippotoso/git-for-team

Un repository con tutti gli script di cui abbiamo parlato ;)

Sembra difficile...

...ma, una volta capito il
meccanismo, è davvero
semplice =)



```
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   public/index.html
#       deleted:    public/sitemap.xml
#       new file:   public/stylesheets/mobile.css
#
# Changes not staged for commit:
#   (use "git add/rm <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       deleted:    app.rb
#       deleted:    test/add_test_crash_report.sh
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       public/javascripts/
```



bit.ly/git-per-team

Il miglior tutorial testuale sull'utilizzo di git

Grazie!



Domande?

Potete trovarmi su Skype ;)

- @FilippoToso
- www.filippotoso.com