Corsi git 2018

Git for teamwork

POLITECNICO OPEN
unix LABS

fcremo <fcremo@poul.org>

Collaborare con git

Esempio ricorrente

Avete visto le basi di git, ora (in vero stile Politecnico) calcolate la massa del sole!

Voi (Bruno) collaborerete con Aldo per realizzare un'app che fornisce informazioni sul sistema solare.

Creare un repo

Aldo ha già creato un repo con l'MVP della vostra app che calcola la massa del sole. Dovrete aggiungere i pianeti.

Per condividerlo con voi crea un progetto (es. su GitLab) e carica il suo codice.

```
git remote add origin \
    ssh://git@gitlab.poul.org:10000/fcremo/esempio-massa-pianeti.git
git push -u origin master
```

`git push` invia i cambiamenti al repo remoto.

Repository remoti

Un remote è un repo git non locale. È identificato da un URL e ha un nome.

Potete manipolare i remote usando i comandi

```
git remote <add|show|remove|rename|...>
```

Due parole su git lato server

Un repo remoto potrebbe consistere solo in una cartella di rete condivisa montata in locale o disponibile via SSH.

Spesso si utilizzano servizi come GitLab o GitHub per gestire repo remoti, comodi perché integrano issue tracking, controllo d'accessi, CI/CD e altro.

Per imparare quello che vi serve per gestire git su un server personale venite ai nostri Corsi Linux Avanzati!

Protocolli

Git opera principalmente su due protocolli: HTTP(S) e SSH.

`Smart HTTP` funziona senza configurare nulla, ma vi verranno richiesti nome utente e password ogni volta.

SSH deve essere configurato, ma è molto più comodo e consigliato.

Configurare SSH

Per configurare ssh dovrete:

- creare una coppia di chiavi pubblica/privata
- associare la chiave pubblica al vostro account sul server
  - la procedura dipende dal software server side
- configurare ssh per autenticarvi con la chiave privata

Alcune guide:

- GitLab
- GitHub
- Atlassian

Clonare un repo

Dovete aggiungere i pianeti mancanti.

Ottenete una copia del repo di Aldo:

```
git clone \
    ssh://git@gitlab.poul.org:10000/fcremo/esempio-massa-pianeti.git
# git clone https://gitlab.poul.org/fcremo/esempio-massa-pianeti.git
```

Branch remoti

Esattamente come il vostro repo locale, i remote hanno dei branch.

Potete mostrarli usando `git branch --all`

```
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
```

Per convenzione `origin` è dato al remote da cui è stato effettuato il clone, e possiede un branch `origin/master`. Non potete committare direttamente su un branch remoto.

Cosa ha fatto Aldo finora

```
commit 27fb93ee06aeeb04ce1d2b83697ae1c08ce4895f
    (HEAD -> master, origin/master, origin/HEAD)
Author: Aldo <aldo@example.com>
Date:   Sat Mar 2 01:00:17 2019 +0100

    Versione iniziale

    Calcola la massa del sole
```

Codice di Aldo

```
def sun_mass():
    return 1.9885e30

print("La massa del sole è {}kg.\n".format(sun_mass()))
```

Aggiungiamo un altro pianeta!

Branching

Per lavorare a una feature, sperimentare o fixare un bug possiamo creare un nuovo branch. In questo modo possiamo lavorare contemporaneamente su diversi cambiamenti e mantenerli isolati.

Aggiungiamo Venere mentre Aldo lavora a Saturno.

```
git branch venere
git checkout venere # equivalente a git checkout -b venere
```

Il nuovo branch

```
git log --all --decorate --oneline --graph # git log a dog
```

```
* 27fb93e (HEAD -> venere, origin/master, origin/HEAD, master) Versi
```

StackOverflow - Pretty branch graphs

Apportiamo le nostre modifiche

```
def venus_mass():
    return 4.867e24

def sun_mass():
    return 1.9885e30

print("La massa del sole è {}kg.\n"
      "La massa di venere è {}kg\n"
      .format(sun_mass(), venus_mass()))
```

E committiamole:

```
git add mass.py
git commit -m "Calcolo massa di Venere"
```

Integrare le nostre modifiche

Il nostro branch `venere` è avanzato rispetto a `master`, il branch principale del progetto. Come possiamo includere le nostre modifiche?

L'unione di un branch dentro un'altro si chiama `merge`.

```
git checkout master
git merge venere
```

```
Updating 27fb93e..1c2c16a
Fast-forward
 mass.py | 7 ++++++-
 1 file changed, 6 insertions(+), 1 deletion(-)
```

Condividere le modifiche

Per caricate sul remote le nostre modifiche usiamo

```
git push -u origin master
```

L'opzione `-u origin` imposta `origin/master` come il remote branch di default per il branch locale `master`.

Non sarà necessario specificarla nuovamente.

```
git log --all --decorate --oneline --graph
```

```
* 1c2c16a (HEAD -> master, origin/master, origin/HEAD, venere) Calco
* 27fb93e Versione iniziale
```

Ora `origin/master` include il nostro commit.

Dopo il merge è possibile rimuovere il feature branch in modo sicuro.

```
git branch -d venere
```

L'opzione `-d` (**minuscola**) non cancella branch che non sono stati mergiati.

Aldo fa delle modifiche analoghe sul suo branch saturno

```
git checkout -b saturno
```

```
def saturn_mass():
    return 5.685e26

def sun_mass():
    return 1.9885e30

print("La massa del sole è {}kg.\n"
      "Quella di Saturno è {}kg"
      .format(sun_mass(), saturn_mass()))
```

```
git add mass.py
git commit -m "Calcolo massa di Saturno"
git checkout master
git merge saturno
```

Ora Aldo prova a caricare le sue modifiche sul server:

```
git push
```

```
To ssh://gitlab.poul.org:10000/fcremo/esempio-massa-pianeti.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'ssh://git@gitlab.poul.org:10000/
hint: Updates were rejected because the remote contains work that yo
hint: not have locally. This is usually caused by another repository
hint: to the same ref. You may want to first integrate the remote ch
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for de
```

Aldo non può caricare le sue modifiche perché non ha integrato le nostre, che quindi andrebbero perdute! Ci viene suggerito di usare `git pull`, comando che:

- scarica le modifiche presenti nel remote (`git fetch`)
- prova ad effettuare il merge automatico

```
git pull
```

```
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From ssh://gitlab.poul.org:10000/fcremo/esempio-massa-pianeti
   27fb93e..433685c  master     -> origin/master
Auto-merging mass.py
CONFLICT (content): Merge conflict in mass.py
Automatic merge failed; fix conflicts and then commit the result.
```

Risolvere il conflitto

Se lo stesso file è stato modificato sia in remoto che in locale il merge potrebbe fallire.

Git ci segnala quali file sono in conflitto e inserisce dei marcatori per evidenziare le righe che non ha potuto processare automaticamente.

Il nostro conflitto

```
<<<<<<< HEAD
def saturn_mass():
    return 5.685e26
=======
def venus_mass():
    return 4.867e24
>>>>>>> 433685c03bc37b0506bb7d491e22c6c882cdd2ea

def sun_mass():
    return 1.9885e30

print("La massa del sole è {}kg.\n"
<<<<<<< HEAD
      "Quella di Saturno è {}kg"
      .format(sun_mass(), saturn_mass()))
=======
```

Risoluzione

In questo caso, manualmente, teniamo entrambe le funzionalità e rimuoviamo i conflict marker.

```
def saturn_mass():
    return 5.685e26

def venus_mass():
    return 4.867e24

def sun_mass():
    return 1.9885e30

print("La massa del sole è {}kg.\n"
      "Quella di Saturno è {}kg\n"
      "La massa di venere è {}kg\n"
      .format(sun_mass(), saturn_mass(), venus_mass()))
```

Per aiutarci possiamo usare anche `git mergetool` con un tool come meld.

Risoluzione (continua)

Per terminare il merge aggiungiamo il file e committiamo:

```
git add mass.py
git commit
```

Ora la history del progetto è questa:

```
*   1f0b218 (HEAD -> master) Merge branch 'master' of ssh://gitlab.p
|\
| * 433685c (origin/master) Calcolo massa di Venere
* | 441dde6 (saturno) Calcolo massa di Saturno
|/
* 27fb93e Versione iniziale
```

E Aldo può pushare i suoi cambiamenti.

Dopo che Aldo ha pushato i cambiamenti, Bruno può pullare e finalmente entrambi saranno alla pari.

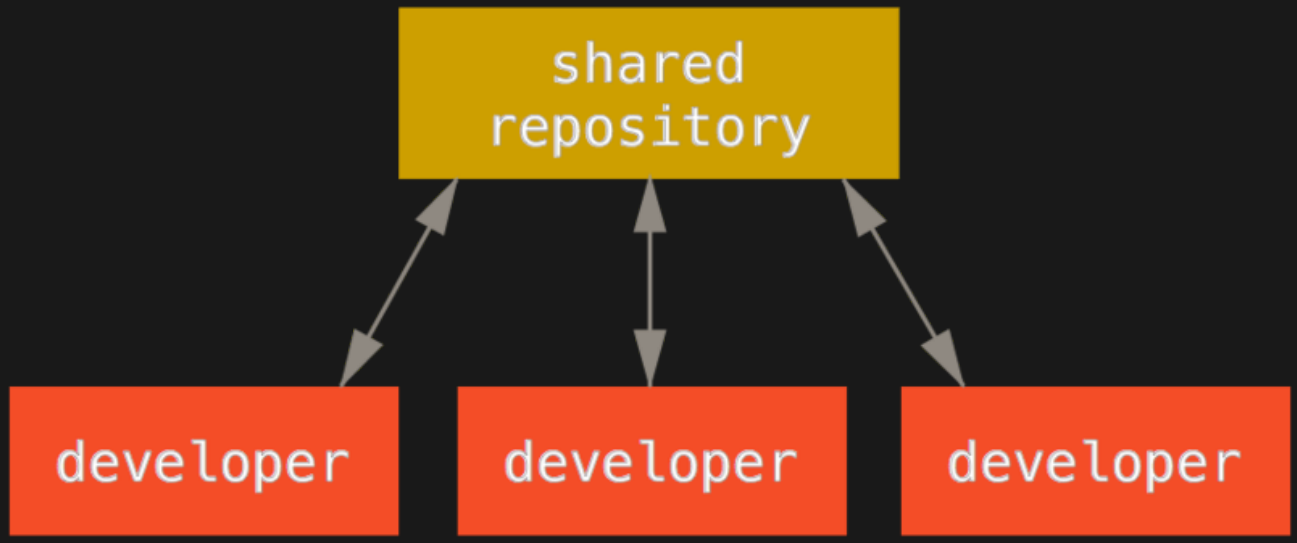
Recap

Abbiamo visto come lavorare contemporaneamente su branch diversi e come riportare le modifiche da un branch all'altro.

Vediamo alcuni modi di organizzare un gruppo che lavora ad un progetto.

Ref. Git Pro book

Centralized workflow

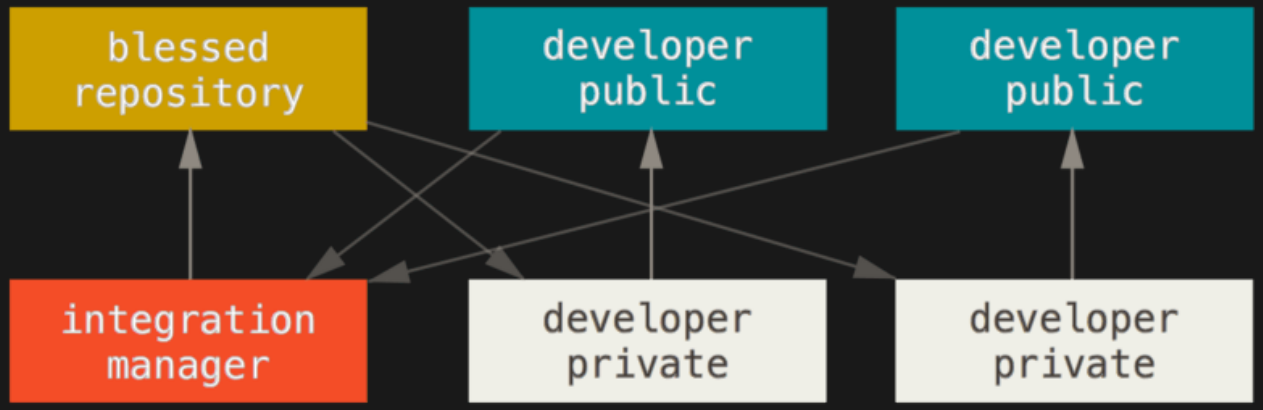


Nell’esempio appena visto tutti pushano su un repo condiviso. Col crescere del numero di componenti del team e della frequenza dei commit cresceranno i conflitti e la disorganizzazione.

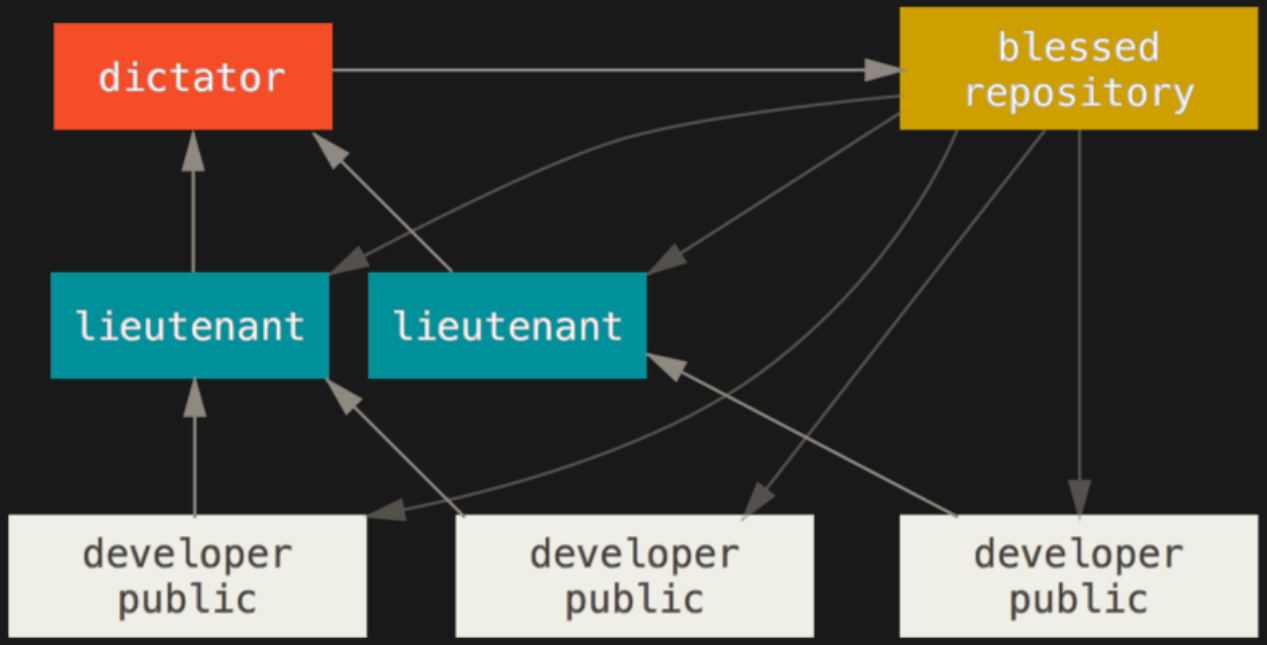
Integration manager workflow

Organizzazione gerarchica adottata da molti progetti

- Un repo ufficiale
  - Spesso master è un branch “stabile” che contiene codice funzionante
- Gli sviluppatori hanno un repo personale
  - I cambiamenti vengono sviluppati su branch personali
- Gli integration manager mergiano branch degli sviluppatori nei branch “ufficiali” su richiesta


blessed repository
developer public
developer public
integration manager
developer private
developer private

Dictator-lieutenants workflow



Variazione del workflow precendente con gerarchie più profonde. Workflow adottato da Linux.

Thank you!

fcremo <fcremo@poul.org>