Git Flow (parte 2)

Corso di Laboratorio di
Gestione progetto e organizzazione d’impresa

Prof. Dei Rossi, Leonardo Essam

Che cos'è Git Flow? (recap)

Gitflow è un modello di branching che prevede l'uso di *feature branch* e molteplici branch primari.

È stato pubblicato per la prima volta e reso popolare da Vincent Driessen nel 2010. Rispetto allo sviluppo basato su trunk[1], Gitflow presenta numerosi branch di lunga durata e commit più grandi.

Con questo modello, gli sviluppatori creano un *feature branch* e ritardano l'unione al ramo principale fino al completamento della feature.

[1] Ovvero uno sviluppo che prevede l'integrazione frequente delle modifiche nel ramo principale del codice

Come funziona (recap)

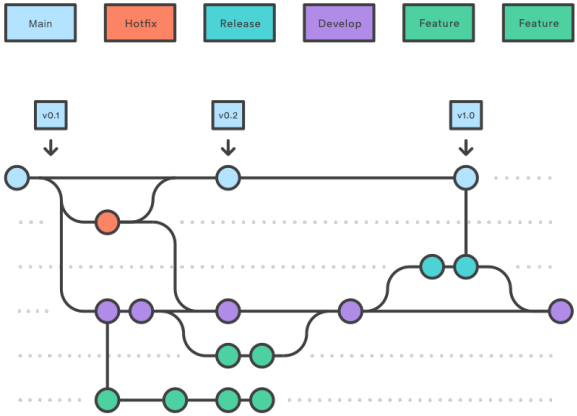
Invece di un singolo ramo principale, Git FLow utilizza diversi rami per registrare la cronologia delle modifiche del progetto.

Gitflow può essere utilizzato per progetti con un ciclo di rilascio pianificato e per le best practice *DevOps* di *continuous delivery*.

Questo flusso di lavoro non aggiunge nuovi concetti o comandi oltre a quanto richiesto dal Feature Branch Workflow, ma piuttosto assegna ruoli molto specifici ai diversi branch e definisce come e quando devono interagire.

Oltre ai feature branch, utilizza branch individuali per la preparazione, la manutenzione e la registrazione dei rilasci.

Schema riassuntivo (recap)



L’estensione di Git Flow (1)

Git contiene al suo interno un’estensione (quindi un sottoinsieme di comandi dedicato) per implementare in modo automatico le regole di Git Flow.

Questo permette di creare un nuovo repository Git e gli opportuni branch di `develop`, `feature/` e `release/` con i giusti nomi/parenti/ecc.

Il comando per inizializzare una nuova repository con Git Flow è:

```
git flow init
```

Attenzione: questo comando va usato in sostituzione di `git init` e non assieme!

L’estensione di Git Flow (2)

```
$ git flow init

Initialized empty Git repository in ~/project/.git/
No branches exist yet. Base branches must be created now.
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
```

L’estensione di Git Flow (3)

I suggerimenti tra parentesi quadre (es. `[feature/]`) rappresentano i valori predefiniti, possono essere cambiati alla bisogna ma solitamente non è necessario. Basta premere INVIO senza inserire nulla e verrà usato il valore predefinito.

Il `Version tag prefix` è un identificativo che viene messo prima del numero di versione vero e proprio, ad esempio:

```
Version tag prefix? [] ver

...

git flow release finish '0.1.0'
Version: 'ver0.1.0'
```

Il branch di feature (1)

Senza le estensioni git-flow:

```
git checkout develop

git branch feature_branch
git checkout feature_branch
```

Quando si utilizza l'estensione git-flow:

```
git flow feature start feature_branch
```

Il branch di feature (2)

Una volta completato il lavoro di sviluppo sulla funzionalità, il passaggio successivo consiste nell'unire feature_branch in develop.

Senza le estensioni git-flow:

```
git checkout develop
git merge feature_branch
```

Quando si utilizza l'estensione git-flow:

```
git flow feature finish feature_branch
```

Il branch di release (1)

Creare rami di rilascio è un'altra operazione di branching semplice. Come i rami di funzionalità, i rami di rilascio si basano sul ramo di sviluppo. Un nuovo ramo di rilascio può essere creato utilizzando i seguenti metodi.

Senza le estensioni git-flow:

```
git checkout develop

git branch release/0.1.0
git checkout release/0.1.0
```

Quando si utilizza l'estensione git-flow:

```
$ git flow release start 0.1.0
Switched to a new branch 'release/0.1.0'
```

Il branch di release (2)

Per completare un ramo di rilascio, utilizzare i seguenti metodi:

```
git checkout main
git merge release/0.1.0
```

Quando si utilizza l'estensione git-flow:

```
git flow release finish '0.1.0'
```

Il branch di hotfix (1)

Avere una linea di sviluppo dedicata alla correzione dei bug consente al team di affrontare i problemi senza interrompere il resto del flusso di lavoro o attendere il ciclo di rilascio successivo.

È possibile pensare ai branch di manutenzione come branch di rilascio ad hoc che interagiscono direttamente con il ramo principale.

Il branch di hotfix (2)

Un branch di hotfix può essere creato utilizzando i seguenti metodi:

Senza le estensioni git-flow:

```
git checkout develop

git branch bugfix_branch
git checkout bugfix_branch
```

Quando si utilizza l'estensione git-flow:

```
git flow hotfix start bugfix_branch
```

Il branch di hotfix (3)

Similmente al completamento di un ramo di rilascio, un ramo di hotfix viene unito sia al ramo principale che a quello di sviluppo.

Senza le estensioni git-flow:

```
git checkout main
git merge bugfix_branch

git checkout develop
git merge bugfix_branch
```

Quando si utilizza l'estensione git-flow:

```
git flow hotfix finish bugfix_branch
```

Buonarroti

Cosa notiamo?

Un doppio merge

Notiamo che viene effettuato un dobbio merge di `bugfix_branch`: sia un `main` che in `develop`.

Ma perché?

Questo perché `main` ha il solo scopo di tenere traccia dei rilasci definitivi dell'applicazione, mentre `develop` tiene traccia dello sviluppo del progetto.

Segue che è necessario che anche `develop` sia aggiornato con l'hotfix appena creato!

Buonarroti