Git Flow (parte 1)

Corso di Laboratorio di
Gestione progetto e organizzazione d’impresa

Prof. Dei Rossi, Leonardo Essam

Che cos'è Git Flow?

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.

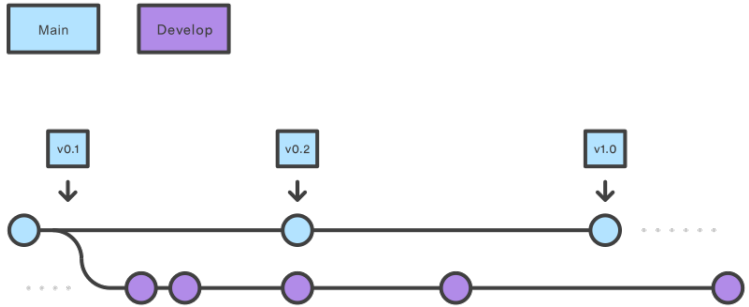
Con questo modello, gli sviluppatori creano un *feature branch* e ritardano l'unione al ramo principale fino al completamento della feature.

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

Come funziona

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

Il ramo principale memorizza la cronologia ufficiale delle release, mentre il ramo di sviluppo funge da ramo di integrazione per le funzionalità.



Branch "main" e "develop"

Il primo passo è integrare il ramo principale (`main`) con un ramo di sviluppo (`development`):

```
git checkout main
git branch development
git checkout development
```

Il branch "feature" (1)

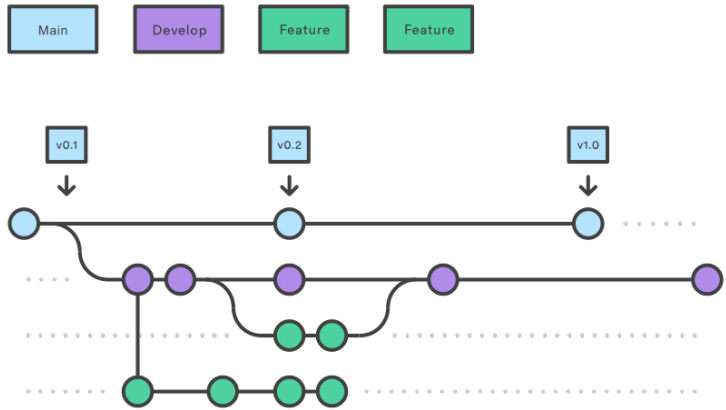
Ogni nuova funzionalità viene quindi sviluppata in un proprio branch, di cui può essere poi fatto il merge per applicare le modifiche.

Tuttavia, invece di diramarsi da `main`, i branch delle funzionalità utilizzano `development` come branch padre.

Quando una funzionalità è completa, viene eseguito quindi un merge con `development`.

I branch `feature` non dovrebbero mai interagire direttamente con main!

Il branch "feature" (2)



Il branch "feature" (3)

Per creare un nuovo branch `feature` vale:

```
git checkout development

git branch feature/<nome>
git checkout feature/<nome>
```

Osserviamo che `feature` è il prefisso, segue quindi poi il nome del branch!

Il branch "feature" (3)

Una volta concluso lo sviluppo di una feature:

```
git checkout development
git merge feature/<nome>
```

Il branch "release" (1)

Una volta che il ramo `development` ha ricevuto un numero sufficiente di feature per il nuovo rilascio (o si avvicina a una data di rilascio prefissata), si crea un nuovo branch di release che origina da development.

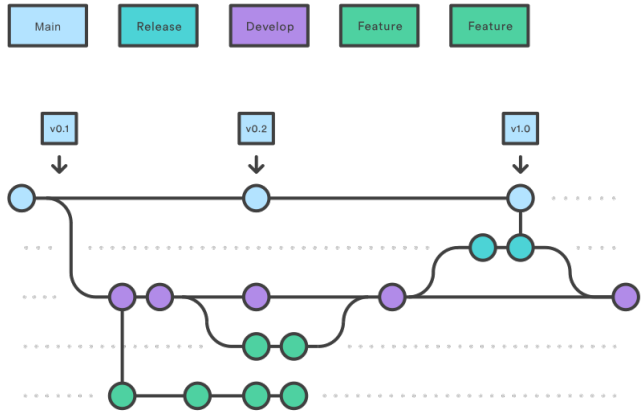
```
git checkout development

git branch release/<versione>
git checkout release/<versione>
```

Il branch "release" (2)

Una volta conclusi gli ultimi ritocchi sul branch di release, si può procedere al merge del ramo con il `main`:

```
git checkout main
git merge release/<versione>
```

Il branch "release" (3)



Il branch "hotfix" (1)

A volte succede che, tra una modifica e l'altra, si verifichi qualche disastro nel codice.

In questo caso, vengono in aiuto i branch `hotfix/<nome>` che nascono e si riuniscono in questo caso direttamente dal `main`.

Il branch "hotfix" (2)

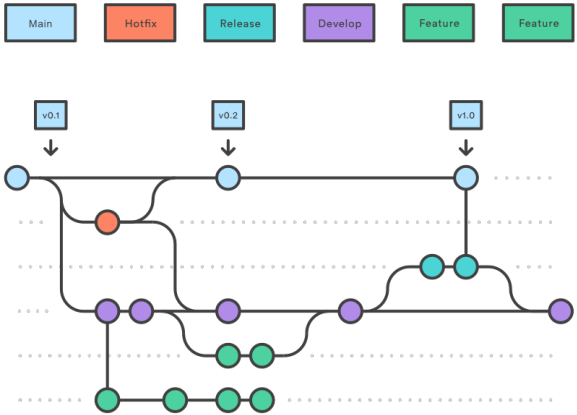
Per creare un nuovo branch di hotfix:

```
git checkout main

git branch hotfix/<nome>
git checkout hotfix/<nome>
```

E per applicare la patch:

```
git checkout main
git merge hotfix/<nome>
```

Il branch "hotfix" (3)



Buonarroti