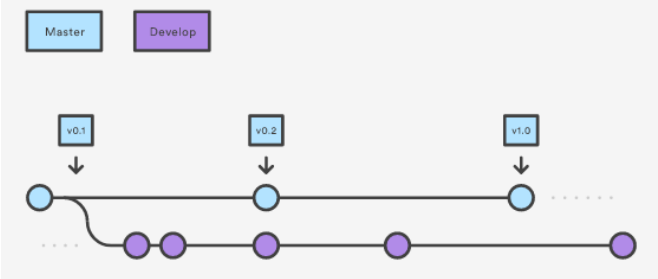
GIT FLOW

(https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow)

Creazione master e develop



Per creare un git flow senza git-flow extension

Si Lancia il git clone

Poi per creare il master:

git branch develop

git push -u origin develop

Oppure con le estensioni git-flow:

git flow init

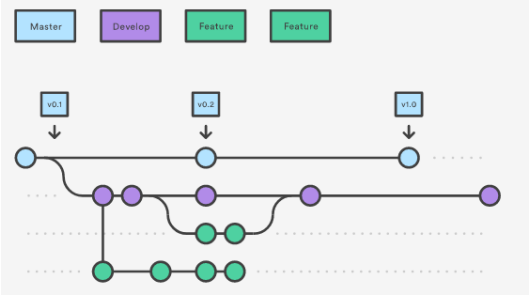
```
$ git flow init
Initialized empty Git repository in ~/project/.git/
No branches exist yet. Base branches must be created n
Branch name for production releases: [master]
Branch name for "next release" development: [develop]

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

$ git branch
* develop
  master
```

il branch develop conterrà la storia del progetto mentre il master solo una versione ridotta

Il branch feature.



Viene creato ogni qual volta si sviluppa una nuova caratteristica, generalmente rimane in locale ma può essere pushato nella repository centrale per la collaborazione con altri sviluppatori, **aprire e chiudere molti feature branches, idealmente uno per features**.

I rami delle features costituiscono a tutti gli effetti con il ramo di sviluppo il flusso di lavoro principale e **vengono creati sull'ultimo commit del branch develop**.

Il ramo parente del ramo feature è develop e mai master!.

Per creare un ramo feature:

git checkout develop

git checkout -b feature_branch

ultilizzando le estensioni git-flow:

git flow feature start feature_branch

Per chiudere un ramo feature:

git checkout develop

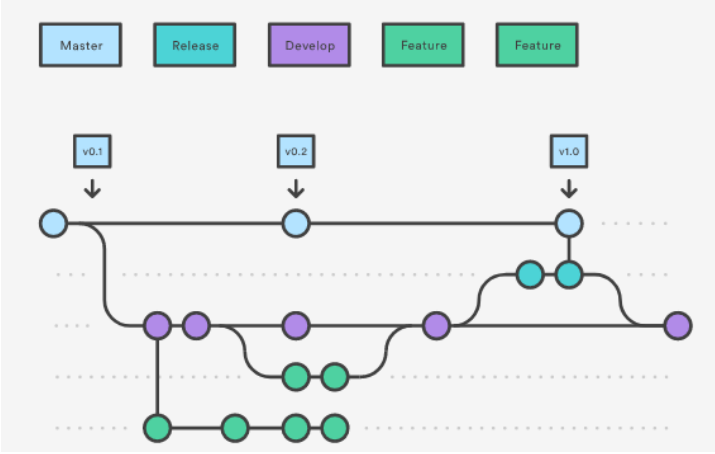
git merge feature_branch

ultilizzando le estensioni git-flow:

git flow feature finish feature_branch

(NB fino a quando non si fa finish feature la feature esiste ancora, si può saltare da un branch ad un altro , mettere in area di staging e committare, quando si eseguirà finish verrà margiato su dev e verrà cancellata la feature

Il Branch release



Quando si sono accumulate un numero abbastanza cospiquo di features è ora di creare un branch release dal branch develop. **Nessuna nuova features verrà aggiunta a questo punto ma solo fix dei bug**. Quando la **release è pronta per andare online viene mergiata nel branch master e taggata col numero di versione**.

Creare un banch release permette di rilasciare una release mentre gli altri vanno avanti con lo sviluppo.

L'uso di un ramo dedicato per preparare la release consente a un team di perfezionare la release corrente correggendo i bachi mentre un altro team continua a lavorare sulle funzionalità per la versione successiva.

Crea un branch release senza le estensioni git-flow

git checkout develop

git checkout -b release/0.1.0

git tag -a v1.0.2 -m "Questa è la versione 1.0.2"

con le estensioni git-flow:

$ git flow release start 0.1.0

Switched to a new branch 'release/0.1.0'

Una volta che la release è pronta va mergiata nel master e nel develop, e quindi la release corrente va cancellata.

Per mergiare nel master la release senza le estensioni git-flow:

git checkout master

git merge release/0.1.0

git checkout develop

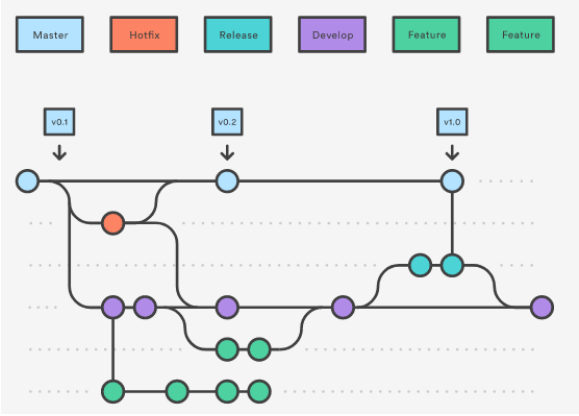
git merge release/0.1.0

delete del branch release…

con le estensioni git-flow:

git flow release finish '0.1.0'

Hot fix branches



Se vi è qualche baco in produzione possiamo fissarlo con l'hot fix, una volta finita la modifica occorre mergiare sul master e sul develop la modifica (o sulla release se è in corso una release). Il master dovrebbe essere taggato con un update version number.

Come creare un branch hotfix senza estensioni git-flow:

git checkout master

git checkout -b hotfix_branch

Con estensioni git-flow

$ git flow hotfix start hotfix_branch

Per chiudere un hotfix:

git checkout master

git merge hotfix_branch

git checkout develop

git merge hotfix_branch

git branch -D hotfix_branch

con estensioni di git-flow:

$ git flow hotfix finish hotfix_branch

IN SINTESI COSE IMPORTANTI DA CAPIRE:

FEATURE

feature: quando si apre e poi si chiude una feature viene in automatico **mergiata sul develop**, quando una **feature viene chiusa viene automaticamente cancellata**, le feature si aprono solo sulla repository locale

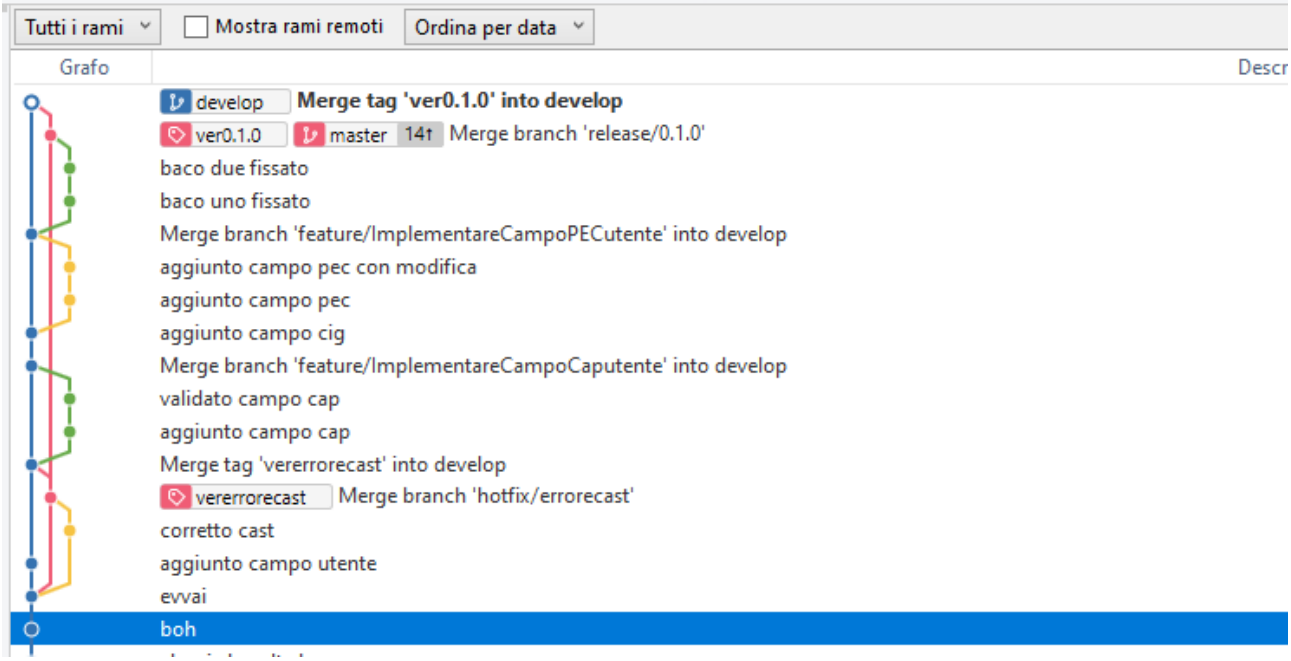
HOTFIX:

quando si apre un **hotfix viene aperto sull'ultimo commit del master**, quando si **finisce un hotfix il contenuto viene mergiato automaticamente sul master e develop** ed il branch hotfix cancellato

RELEASE

La release viene creata con il tag di versione e possiamo correggere solo dei bachi con dei commit e quando siamo pronti a rilasciare in produzione **facciamo un finish della release che provvede ad aggiornare il master ed il develop branch**.

Descrizione di una storia di aggiornamenti su treesource sperimentati da me:



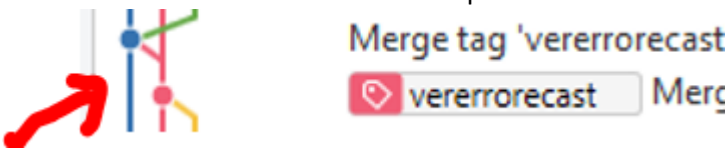
Dev=blu

Master=rosso

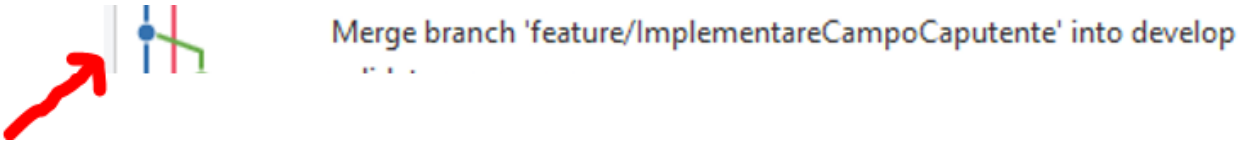
Feature ed hotfix =ora gialli ora verdi

Durante la linea dello sviluppo abbiamo

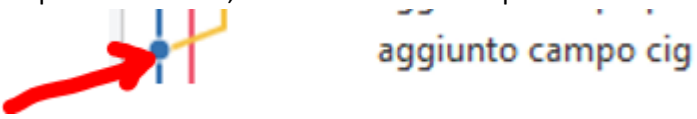
- Aggiunto il campo utente su dev
- Poi ci siamo accorti di un baco in produzione ed abbiamo aperto un hotfix sul master, abbiamo committato lhotfix con il commento “corretto cast” quindi abbiamo finalizzato l’hotfix (durante la fase di finalizzazione ci viene chiesto un tag e noi abbiamo inserito come nome tag vererrorcast e poi viene chiesto un messaggio per la finalizzazione, confermiamo il messaggio di default proposto “merge branch “hotfix/errorecast”. Come possiamo vedere la chiusura dell’hot fix mergia le modifiche sia sul master che sul develop branch:



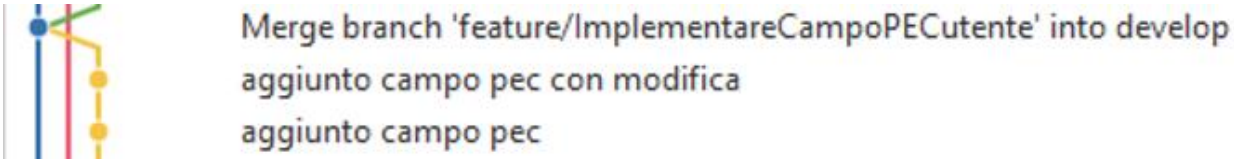
- Poi abbiamo aperto la feature implementarecampoutente, su questa feature abbiamo eseguito un paio di commit, ed infine abbiamo finalizzato la feature, è possibile notare che la finalizzazione interessa solamente il merge su develop branch, infatte una feature ha impatto solo su develop branche e non sul master!:



- Poi abbiamo aggiunto una nuova feature che è aggiunto campo cig, poiché in questa feature abbiamo committato una sola volta su tree source non viene creato un percorso colorato ma solo un puntino azzurro, come se fosse un semplice commit direttamente su develop:



- Poi abbiamo aperto un’altra feature implementacampopecUtente con due commit e l’abbiamo finalizzata



- Infine abbiamo deciso di rilasciare in produzione la release 0.1.0 quindi abbiamo creato un branch release, durante il processo di rilascio abbiamo corretto un paio di bachi e quindi abbiamo finalizzato la release (durante la finalizzazione ci è stato chiesto di inserire un nome per il tag, di inserire un commento per la finalizzazione, come possiamo vedere la release ha aggiornato il maser

ed il branch develop: