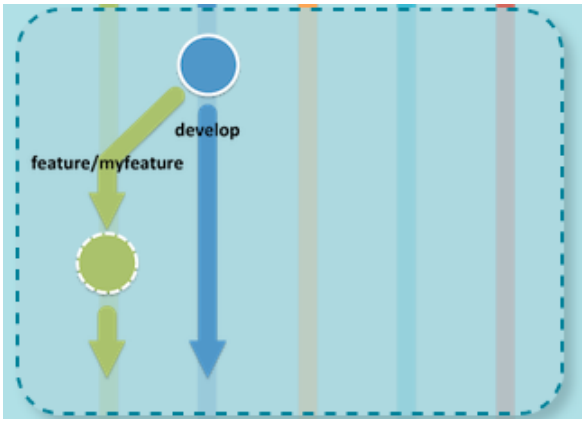
Premessa

Tutti i comandi maven bisogna lanciarli dalla folder del progetto padre `/brewplus`

Creazione di un nuova feature (ex branch svn)

Crea una nuova feature facendo uno snapshot del develop



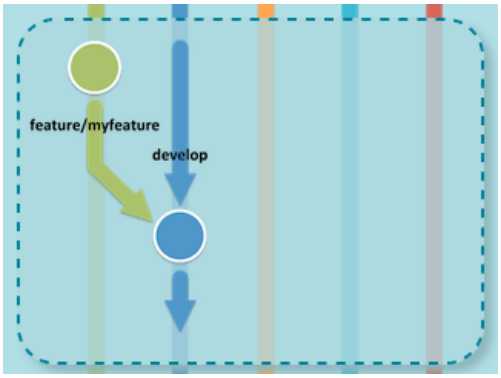
`mvn jgitflow:feature-start`

Questo commando poi ti chiede in modo iterattivo il nome della feature da creare.

`mvn jgitflow:feature-start -DfeatureName=myfeature`

→ Questo comando invece genera automaticamente la feature “feature/myfeature” automaticamente.

Chiusura di un nuova feature col conseguente merge nel develop – NON sarà possibile usarlo se il develop è lockato da pull-request



`mvn jgitflow:feature-finish -DnoFeatureBuild=true`

Questo comando poi ti chiede in modo iterattivo il nome della feature da chiudere.

Viene automaticamente proposto quella attuale dove si è col checkout

`-DnoFeatureBuild=true` è importante senno ti ricompila tutto brewplus e non ha senso perché sicuramente si deve aver già testato la feature

Quando viene chiusa la feature viene eliminata dal repository locale e verrà fatto automaticamente il checkout su develop.

Switch da una branch ad un' altro

Ecco un po' di esempi

```
git checkout feature/myfeature2
```

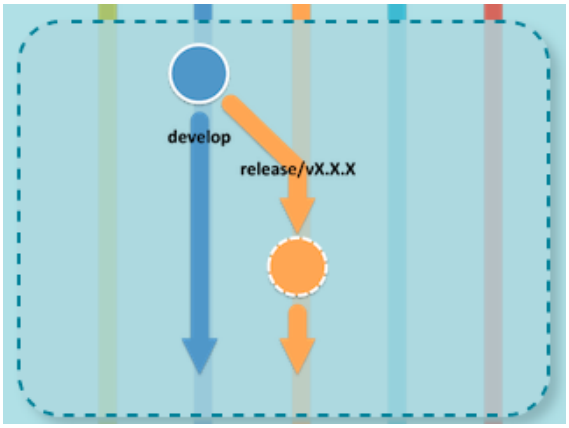
```
git checkout develop
```

```
git checkout master
```

```
git checkout release/2.2.0
```

```
git checkout hotfix/2.1.1
```

Creazione di un branch di release




```
mvn jgitflow:release-start
```

Viene creato un nuovo branch di release prendendo lo stato attuale del branch.

Effettuare quindi questo branch quando si è verso la chiusura del progetto e soprattutto quando si hanno chiuso tutte le feature. (eventualmente se succede che si debbano aggiungere nuove feature… niente paura basta mergiarle direttamente in release oppure si sviluppa direttamente in release)

Qui ti chiede come si chiamerà la release: → mettere quindi ad esempio 2.2.0 **senza RC**

Ti chiederà inoltre quale sarà la versione del pom del develop. Ti proporrà quindi 2.3.0-SNAPSHOT

Creerà quindi la release/2.2.0 e tutti pom verranno marchiati come 2.2.0-SNAPSHOT

Qui quindi si dovrebbe pushare lo stato attuale: il develop (con i nuovi poms) e la nuova release

Puoi farlo da SourceTree oppure da linea comando

```
git push –all
```

➔ Ora dobbiamo mettere tutti i pom della release in RC1

```
mvn versions:set -DnewVersion=2.2.0-RC1
```

se tutto ok

```
mvn versions:commit
```

se invece c'è stato qualche problema

```
mvn versions:rollback
```

Attenzione il commando `mvn versions:commit` non andrà a fare il commit del git ma solo il comando conferma il comando precedente `mvn versions:set`

Qui quindi possiamo commitare in git

```
git commit -a -m "Release Candidate 1"
```

Possiamo fare ulteriori (di solito piccole) modifiche al volo da committare e pushare in release/2.2.0

➔ **Finalmente ora facciamo il depot in RC1 in GitHub in pre-release**

O da SourceTree (flaggando anche la creazione anche in remoto) o da linea comando creare il tag 2.2.0-RC1

```
git tag 2.2.0-RC1
```

```
git push --all
```

```
git push –tags
```

Ora sarebbe bello che partisse un job di Jenkins (quando lo attiveremo) a seguito di una push sul tag. Quando ha finito completamente il job sul tag e viene fuori FINISH SUCCESS allora è possibile andare a rilasciare lo zip su GitHub come pre-release.

Ora ci sono2 possibilità: o va tutto bene e vanno in produzione oppure ci fanno un riciclo e ci mandano in RC2

1. **Riciclo in RC2 (o successive)**

   Bisogna cambiare la versione dei poms

```
mvn versions:set -DnewVersion=2.2.0-RC2
```

```
mvn versions:commit
```

```
git commit -a -m "Release Candidate 2"
```

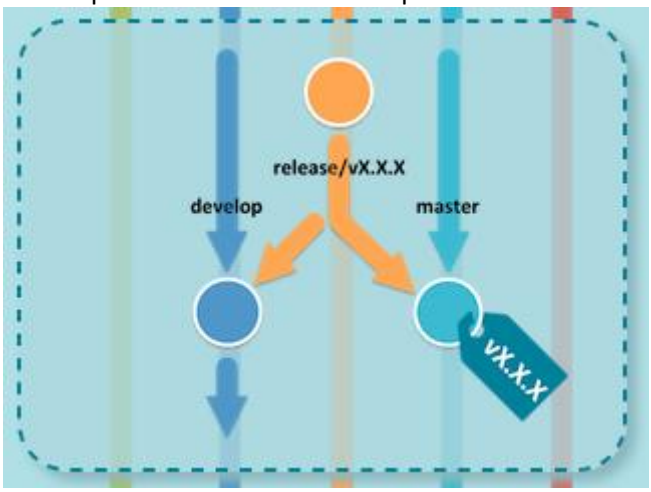
Ora fare le modifiche al codice errato e fare questa volta il tag nuovo 2.2.0-RC2 come spiegato precedentemente

Poi ributtare il nuovo zip su GitHub sempre in pre-release.

2. Chiusura della release perché si va in produzione

E' andato tutto bene si va in produzione

E' quindi ora di chiudere la release e passare il codice in develop e in master



ATTENZIONE

ATTENZIONE

ATTENZIONE

PER CHIUDERE UNA RELEASE BISOGNA PASSARLA IN VERSIONE SNAPSHOT

Se non si mette in SNAPSHOT poi sarà molto difficile mettere a posto… tranquilli … non impossible

`mvn versions:set -DnewVersion=2.2.0-SNAPSHOT`

`mvn versions:commit`

`git commit -a -m "update poms to 2.2.0-SNAPSHOT"`

➔ Pushare tutto

`mvn jgitflow:release-finish -DnoReleaseBuild=true`

➔ PS ricordarsi analogamente lo `-DnoReleaseBuild=true`

Automaticamente verrà creato (localmente), mergiato il codice in develop e in master e il tag 2.2.0 (dal master)

Ora pushare tutto : conviene manualmente con

`git push --all`

`git push –tags`

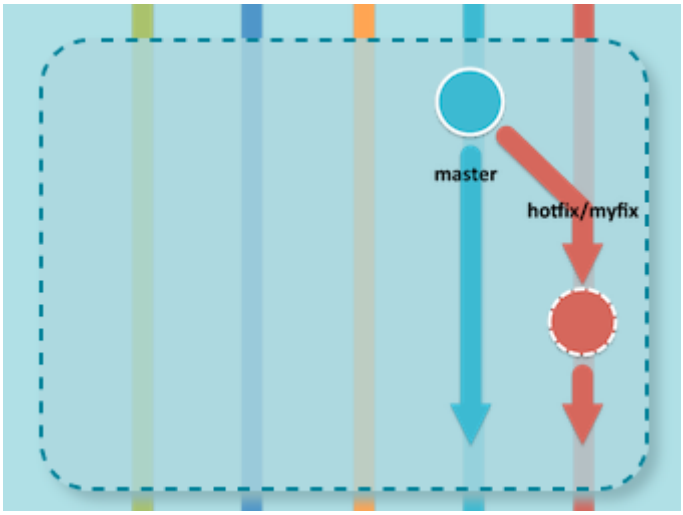
Creazione / chiusura di un branch di hotfix

La hotfix serve quando abbiamo già chiuso delle feature in develop ma si hanno degli errori in produzione da correggere il più velocemente possibile.

Il giro è simile alla release, quindi tutti i flussi di depot, RC1… non li spiego.

```
mvn jgitflow:hotfix-start
```

La hotfix start preleva la situazione dal master



L'unica cosa che tengo a ricordare è questa:

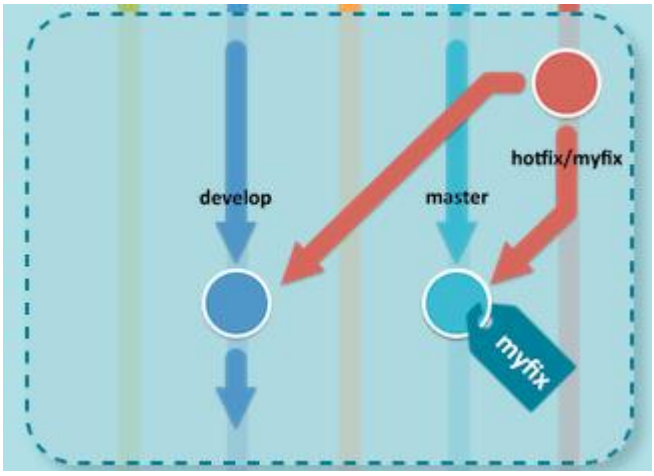
ATTENZIONE

ATTENZIONE

ATTENZIONE

PER CHIUDERE UNA HOTFIX BISOGNA PASSARLA IN VERSIONE SNAPSHOT

```
mvn jgitflow:hotfix-finish -DnoHotfixBuild=true
```



La chiusura di una hotfix effettua quindi un merge in master e in develop.