

Branching and Merging



Branching Basics

```
1 git branch # list local branch
2 git branch -a # list local and remote branch (*master = local & remotes/origine
  /master = remote)
3 git branch mynewbranch # new branch is created
4 git branch -a # new branch listed
5 git checkout mynewbranch # pour changer de branche
6 git branch -a
7 git log --oneline --decorate
8 git checkout master
9 git branch -m mynewbranch newbranch # renommer une branche
10 git branch -a
11 git branch -d newbranch
12 git branch -a
```

Happy Path / Fast Forward Merges

Merge se fait toujours d'une branche sur une autre :

exemple :

La branche courante est master :

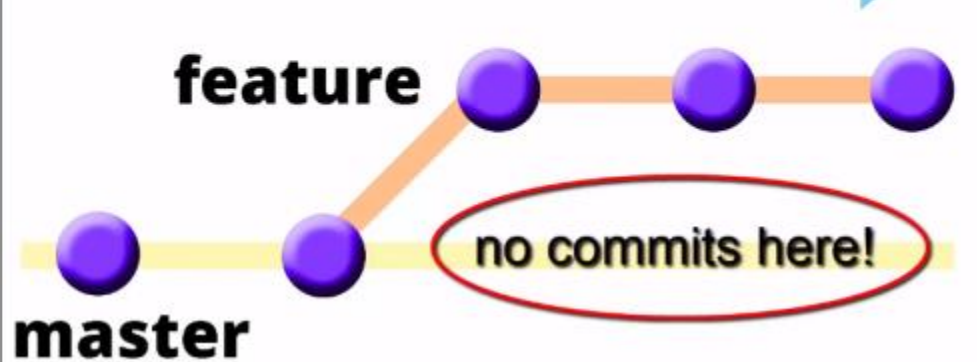
git merge title-change

va provoquer le merge de **title_change** sur **master**.

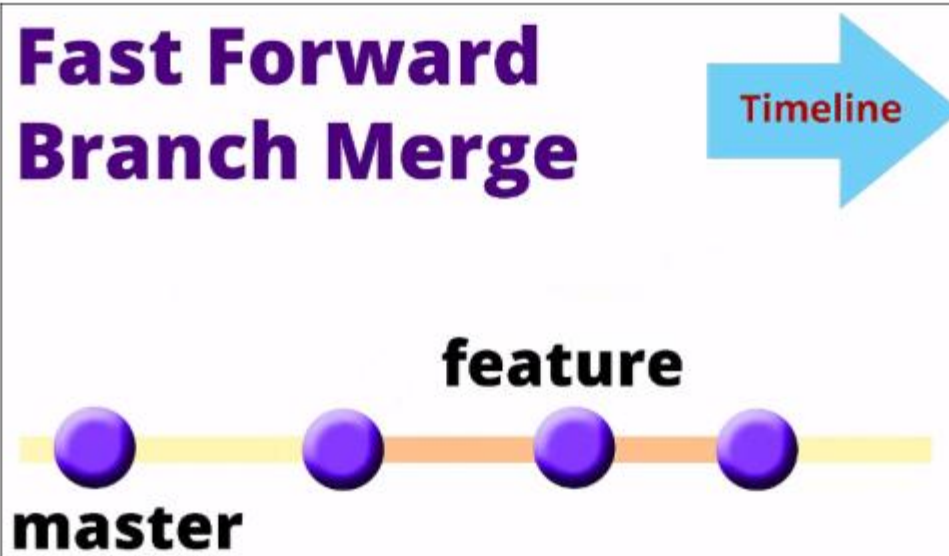
master sera modifiée pour intégrer les changements que title_change apporte.

title_change ne sera pas modifiée et pourra être supprimée après si elle n'est plus utile.

Fast Forward Branch Start



Fast Forward Branch Merge



git hist : correspond à l'alias hist que nous avons créé sur la commande git log dans le chapitre basics

```
1 git checkout -b title-change # crée une branche et fait un checkout sur la branche
2 git status
3 npp simple.html # changer le texte
4 git status
5 git commit -am "Changing title of HTML"
6 git log --oneline
7 git checkout master # On se replace dans master
8 git diff master title-change
9 git difftool master title-change
10 git merge title-change # master intègre les changements de title_change
11 git hist # Correspond à l'alias hist que nous avons créé sur la commande git log
    dans le chapitre basics
12 git branch
13 git branch -d title-change # destruction de de title_change
14 git hist
```

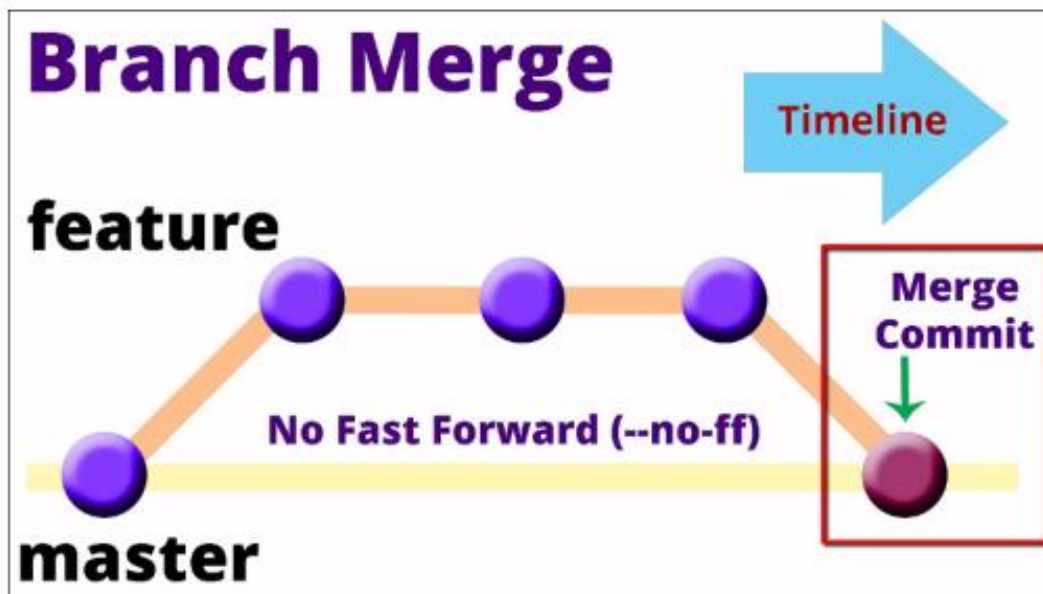
Happy Path / Disable Fast Forward Merges

merge va appliquer tous les commits historiques les uns après les autres jusqu'au dernier.

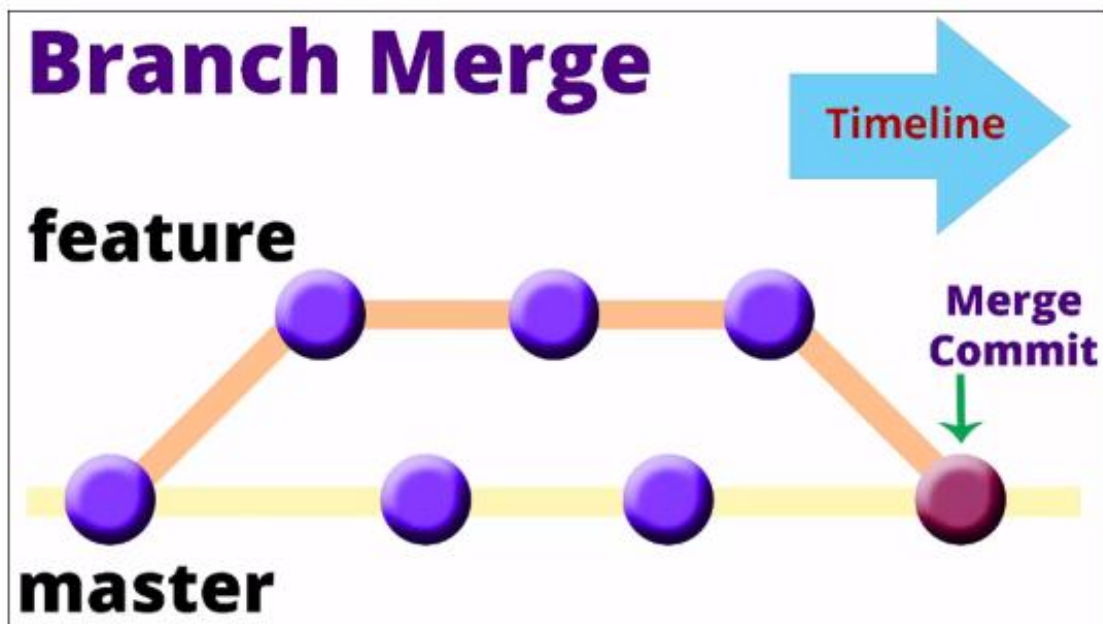
- Si entre les 2 branches , les commits historiques sont cohérents => pas de pb , on applique **Fast Forward** : pas de création de nouveau commit , master reprend le commit de la branche fusionnée.
- Si il y a ambiguïté entre les commits historiques des branches => on applique **No Fast Forward** : Il y a creation d'un nouveau commit, le commit de merge

git hist : correspond à l'alias hist que nous avons créé sur la commande git log dans le chapitre basics

```
1 git checkout -b add-copyright
2 git branch
3 npp simple.html # ajouter du code pour le copyright
4 git status
5 git commit -am "Adding copyright notice"
6 npp README.md # Ajouter du texte pour le copyright
7 git commit -am "Adding copyright notice to readme"
8 git hist
9 git checkout master
10 git merge add-copyright --no-ff
11 git hist
12 git branch -d add-copyright
13 git hist
```



Automatic Merges

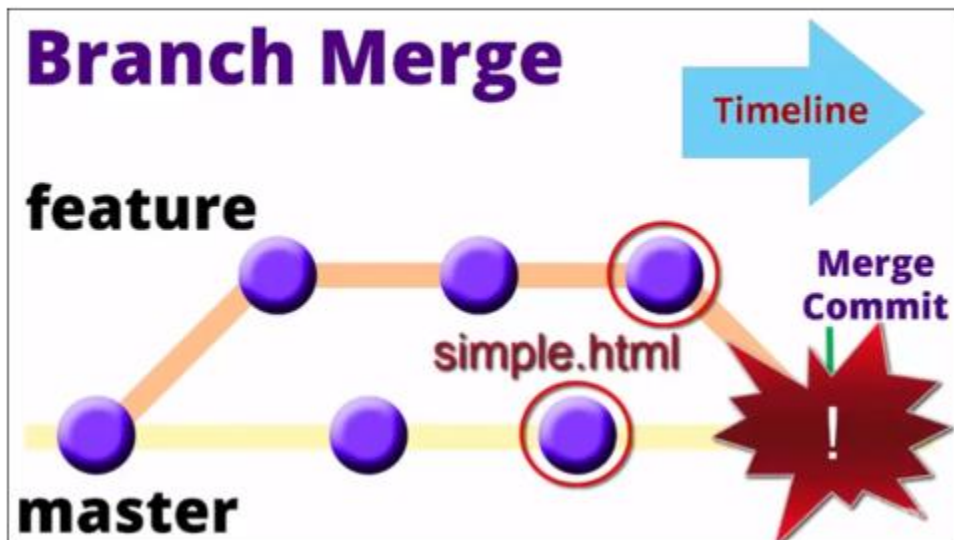


Dans le cas où le master et la branche ont effectué des commits chacun de leur côté, au moment du merge, git va automatiquement créer un commit de merge, sans qu'on ait besoin de préciser le paramètre `no-ff`.

git hist : correspond à l'alias hist que nous avons créé sur la commande git log dans le chapitre basics

```
1 git checkout -b simple-changes
2 git branch
3 npp humans.txt # ajouter du code
4 git commit -am "Changing humans.txt"
5 git checkout master
6 npp index.html # ajouter du code
7 git commit -am "Changing footer on index.html"
8 git hist
9 git merge simple-changes -m "Merging changes from simple-changes branch"
10 git branch
11 git branch -d simple-changes
12 git hist
```

Conflicting Merges and Resolution



Objectif : créer une situation de conflit entre branches et le résoudre au moment du merge.

- Le dernier commit de chaque branche modifie la même ligne dans un fichier

```
1 ***** Etape 1 *****
2 git checkout master
3
4 npp conflict.txt
5 #ajouter les lignes suivantes:
6 ligne1
7 ligne2
8 ligne3
9
10 git add .
11 git commit -m "Creation de confict.txt"
12
13 ***** Etape 2 *****
14 git checkout -b realwork
15 git branch
16
17 npp conflict.txt
18 # Changer la ligne 2 comme suit
19 Je modifie la ligne 2
20
21 git add .
22 git commit -m "Modification de la ligne 2"
23
24
25 ***** Etape 3 *****
26 git checkout master
27
28 npp conflict.txt
29 # Remplacer la ligne 1 :
30 Je mofifie la ligne 1 dans master
31
32 git add .
33 git commit -m "Modification de conflict.txt dans master"
34
35 git hist
36
```

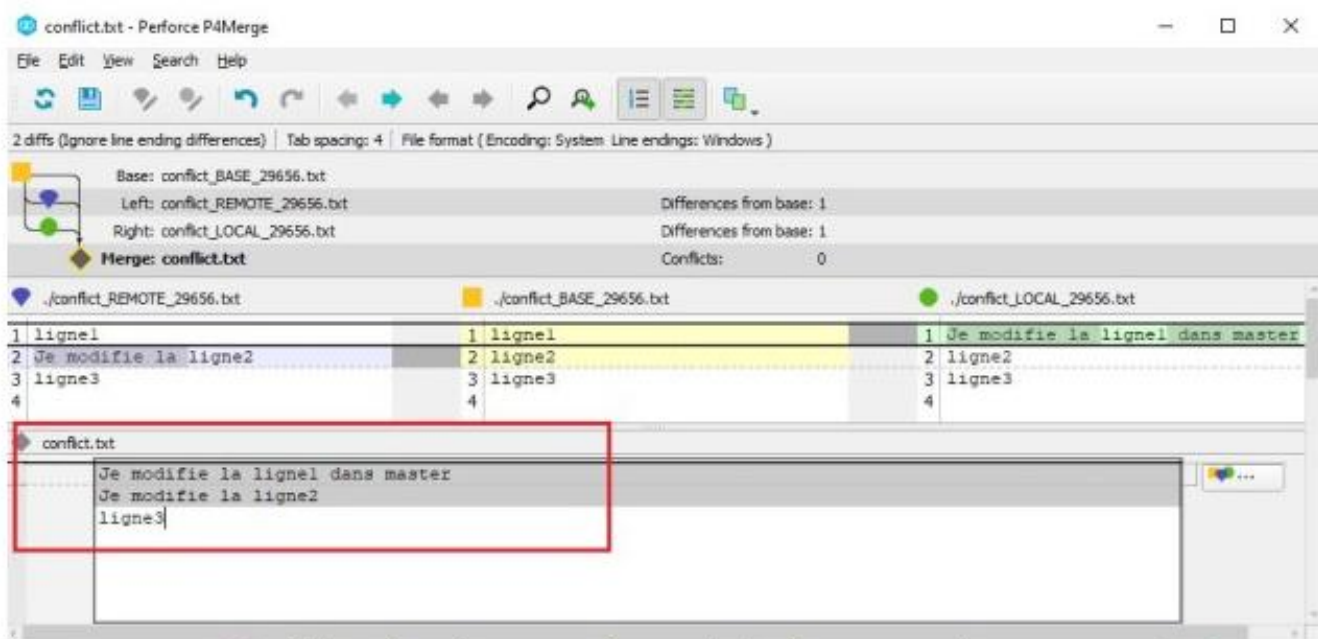


```


37 ***** Etape 4 *****
38 ***** Commencer le merge *****
39 git merge realwork -m "Merging changes from realwork branch"
40
41
42 git diff master realwork # Constater les différences de manière textuelle
43 git difftool master realwork # Constater les différences avec l'outil graphique
44
45 *** Resoudre le conflit ***
46 git mergetool
47 # cf. illustration ci-dessous
48 # Effectuer les modifications dans la partie basse de l'écran et enregistrer.
49 # local représente le contenu du fichier dans la baranche master.
50 # remote représente le contenu du fichier dans la branche realwork
51
52
53 ***** Etape 5 *****
54 ***** Finir le commit *****
55 git status
56 git add.
57 git commit -m "Resolution du conflit merge avec realwork"
58 git status
59
60 git branch
61 git branch -d realwork # Si vous souhaitez supprimer la branche realwork
62 git hist

```

illustration de git mergetool




Complément à effectuer si vous ne souhaitez pas que git suive les fichiers .orig



```
1 npp .gitignore # ajouter *.orig
2 git status
3 git add .gitignore
4 git commit -m "creating ignore file"
```

Cleanup and Push back to GitHub



```
1 git pull origin master
2 git push origin master
3
```