
git
Semifir
Adrien Vossough
https://www.semifir.com/

Branches et tags

Les Tags

Un tag

- C'est un alias (un nom) défini par un développeur
- Il pointe vers un commit.
- Il facilite l'identification d'un commit
- Il nomme un moment précis l’état du dépôt.
- Il permet de marquer des numéros de version sur des commits.

Le numéro de version du type: x.y.z

- Le x va marquer une version majeure pour des modifications importantes ou encore induit une incompatibilité avec une version précédente.
- Le y va marquer une version mineure qui ajoute des fonctionnalités tout en conservant une compatibilité avec l’ancien système. (Attention aux dépréciations de fonctionnalités)
- Le z va marquer un patch pour corrections de bugs sans ajout de fonctionnalités.

Les Tags

Deux types de tags

- Les tags légers stockent le nom du tag sur un commit donné.
- Les tags annotés stockent davantage d'informations en conservant également la date de création et le créateur du tag.

créer un tag léger en se positionnant sur le commit
git tag nom_du_tag **# exemple :** **git tag v1.22.91**

Les Tags

créer un tag léger en spécifiant le hash du le commit

```
git tag v1.22.91 be33a4
```

Création d'un tag **annoté**

```
git tag -a nom_du_tag

# exemple
git tag -a v1.9

# avec message
git tag -a v1.9 - "message du tag"
```

Les Tags

Lister les tags

```
git tag --list
# ou
git tag -l

# nombre de ligne du tag à afficher
git tag -l -n3

# détail d'un tag
git show v1.8.5
```

Les Tags

Envoie des tags sur le dépôt distant

```
git push origin –tags

# envoie des tags un à un

git push origin nom_du_tag
```

Suppression d'un tag

```
git tag -d nom_du_tag

# pour l'indiquer dans le dépôt

git push origin :refs/tags/nom_du_tag
```

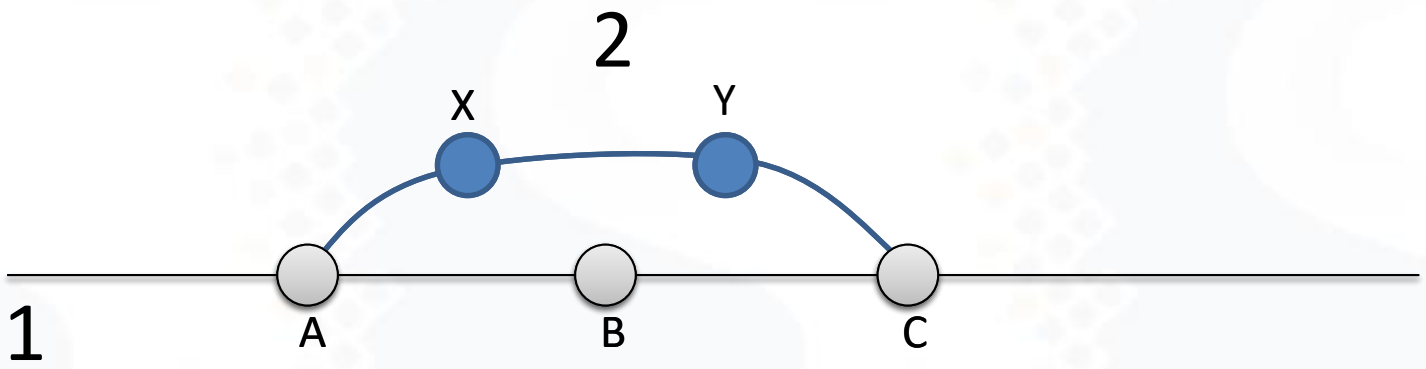
Les branches

Une branche

- correspond à une version parallèle de celle en cours de développement.
- peut servir à développer de nouvelles fonctionnalités
- Peut servir à corriger des bugs
- Permet de segmenter différentes versions en cours de développement.

Par défaut, sur un dépôt Git, une branche master est créée. C'est sur cette branche qu'il faut effectuer toutes les manipulations.
Les branches sont des alias dont la référence est dynamique contrairement aux tags qui sont statics.

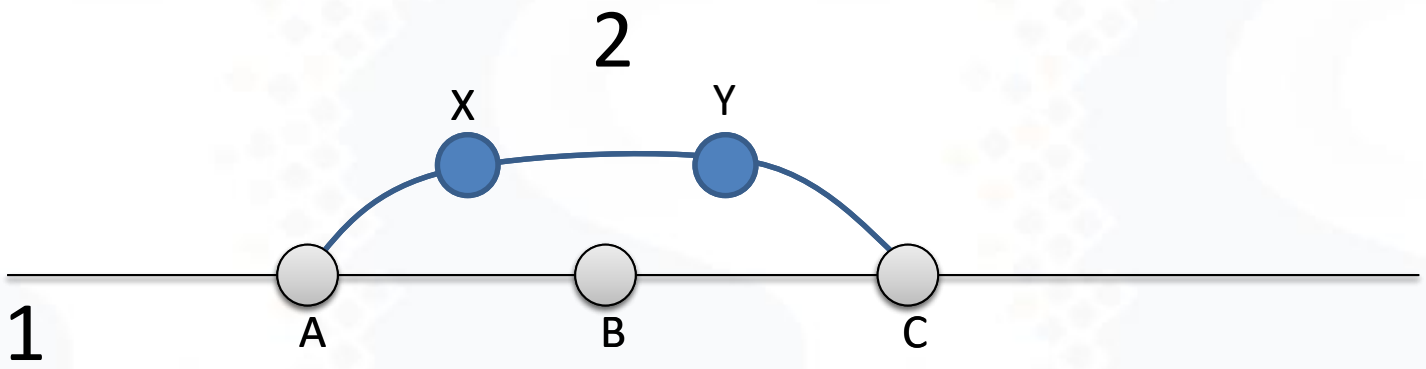
Les branches



Le développeur a créé une branche, 2 à partir du commit A pour ajouter une nouvelle fonctionnalité.

Le deuxième commit X correspond au premier commit après la création d'une branche, le commit Y a suivi. Le développeur fusionne sur la branche principal une fois qu'il a finit ses modifications

Les branches



Une fois les modifications intégrées, le développeur va commiter pour enregistrer le résultat de la fusion. À ce moment-là, les 1 et 2 pointent vers le même commit "C"

Les branches

Liste des branches existantes

```
git branch
* Master

liste des branches avec le hash et le message du dernier commit
 git branch -v
```

Création d'une branche. Vérifier d'abored de quelle branche elle va dériver !

```
git branch nom_branche
```

Se placer sur une branche

```
git checkout nom_branche
```

Les branches

Avant de changer de branche, il faut s'assurer que la branche actuelle n'a pas de modification à "commiter"

Créer une branche et se placer dessus

```
git checkout -b connexion_sociale
```

4. Fusionner deux branches

Les fusions, également appelées merges, correspondent à l'opération de récupération des différences d'une branche vers une autre. Par exemple, lorsqu'un développeur utilise git pull pour récupérer les commits de la branche suivie à distance, après avoir téléchargé les commits sur le serveur distant, Git effectue une fusion entre la branche courante et la branche suivie à distance. Une fusion revient à intégrer les modifications d'une branche dans une autre.

Par exemple, si un développeur a utilisé une branche ajout_tarif pour ajouter la gestion des tarifs, cela signifie que s'il souhaite fusionner la branche master avec la branche ajout_tarif qui hérite de master, il doit tout d'abord se poser la question suivante : dans quelle branche dois-je intégrer les modifications de l'autre branche ?

Dans le cas actuel, le développeur souhaite ajouter dans la branche master les modifications apportées par la branche ajout_tarif. Pour cela, il va tout d'abord se positionner sur la branche master puis va utiliser la commande git merge pour fusionner les modifications de la branche ajout_tarif dans la branche master :

```
git checkout master
git merge ajout_tarif
```

Historique

Exemple de conflit

La fusion n'est pas toujours évidente à appréhender, c'est la raison pour laquelle il vaut mieux l'aborder au travers d'un cas pratique simple.

Le dépôt résultant de cet exemple se trouve dans les fichiers en téléchargement : 06/merge_test.

Pour cela, il faut créer un nouveau dépôt :

git init merge_test
Il faut ensuite créer le fichier index.html avec le contenu suivant :

```
<html>
<head><title>Mon serpent préféré</title></head>
<body>
   Le python est un serpent bien et majestueux.
</body>
</html>
```

6. Rebaser une branche dans une autre

Il existe un autre moyen d'intégrer les modifications d'une branche dans une autre branche : git rebase. C'est une fonctionnalité fondamentalement différente du merge. Un merge a pour but d'intégrer plusieurs commits cohérents entre eux dans un commit unique représentant tous les commits mergés. Un rebase correspond à une réécriture d'historique, par exemple pour intégrer les commits d'une branche dans une autre. Au lieu d'intégrer toutes les modifications d'une branche dans un unique commit, il est possible de modifier la base d'une branche en incluant les commits d'une autre branche.

Pour expliquer de façon plus pragmatique la différence entre un merge et un rebase, il convient d'analyser les résultats des deux fonctionnalités. Le schéma ci-dessous présente la situation initiale qui permettra d'effectuer la comparaison :

images/06E07.png

Le schéma qui se trouve précédemment dans ce chapitre et qui explique le principe du merge présente l'exemple du merge de la branche nommée branche A dans la branche master. Dans cet exemple, le commit C5 implémente les modifications de branche A (représentée par les commits C2 et C4) dans la branche master.