Gestion des versions – Versioning

Git - *Fonctionnalités avancées*

Git Pull, c'est quoi ?

Git pull

- Git Pull permet de télécharger le contenu d'un dépôt distant pour mettre à jour son dépôt local.
- Il permet de récupérer le dernier commit effectué
- C'est le moyen le plus efficace de s'assurer de travailler sur la version actuelle.
- Après avoir effectué les modifications sur votre dépôt local, téléchargez les modifications et les nouveautés dans le dépôt distant à l'aide de la commande git Push.
- Travailler avec les commandes Git Pull et Push est important, notamment lorsque plusieurs utilisateurs travaillent en même temps sur un projet et que chacun doit avoir la dernière version à chaque instant.
- On peut considérer la commande Git Pull comme une sorte de mise à jour.
- Cette commande est composée d'un **fetch** et d'un **merge**.

```
git pull [url_depot_distant]
```

Git Pull, c'est quoi ?

```
git pull [url_depot_distant]
```

Dans le cas où la commande git pull ne permet de mettre à jour le dépôt local, on doit récupérer le dernier commit et puis lancer le git pull :

```
git reset --hard HEAD~2

git pull [url_depot_distant]
```

Les branches

Définition et intérêts des branches

- Créer une **branche** signifie diverger de la ligne principale de développement et continuer à travailler sans se préoccuper de cette ligne principale.
- La branche par défaut dans Git quand vous créez un dépôt s'appelle main (***master)*** et elle pointe vers le dernier des **commits** réalisés.
- Créer une branche, c'est en quelque sorte comme créer une "copie" de votre projet pour développer et tester de nouvelles fonctionnalités sans impacter le projet de base.
- Une branche, dans Git, est simplement un pointeur vers un commit (une branche n'est qu'un simple fichier contenant les 40 caractères de l'empreinte SHA-1 du commit sur lequel elle pointe).

Pourquoi des branches ?

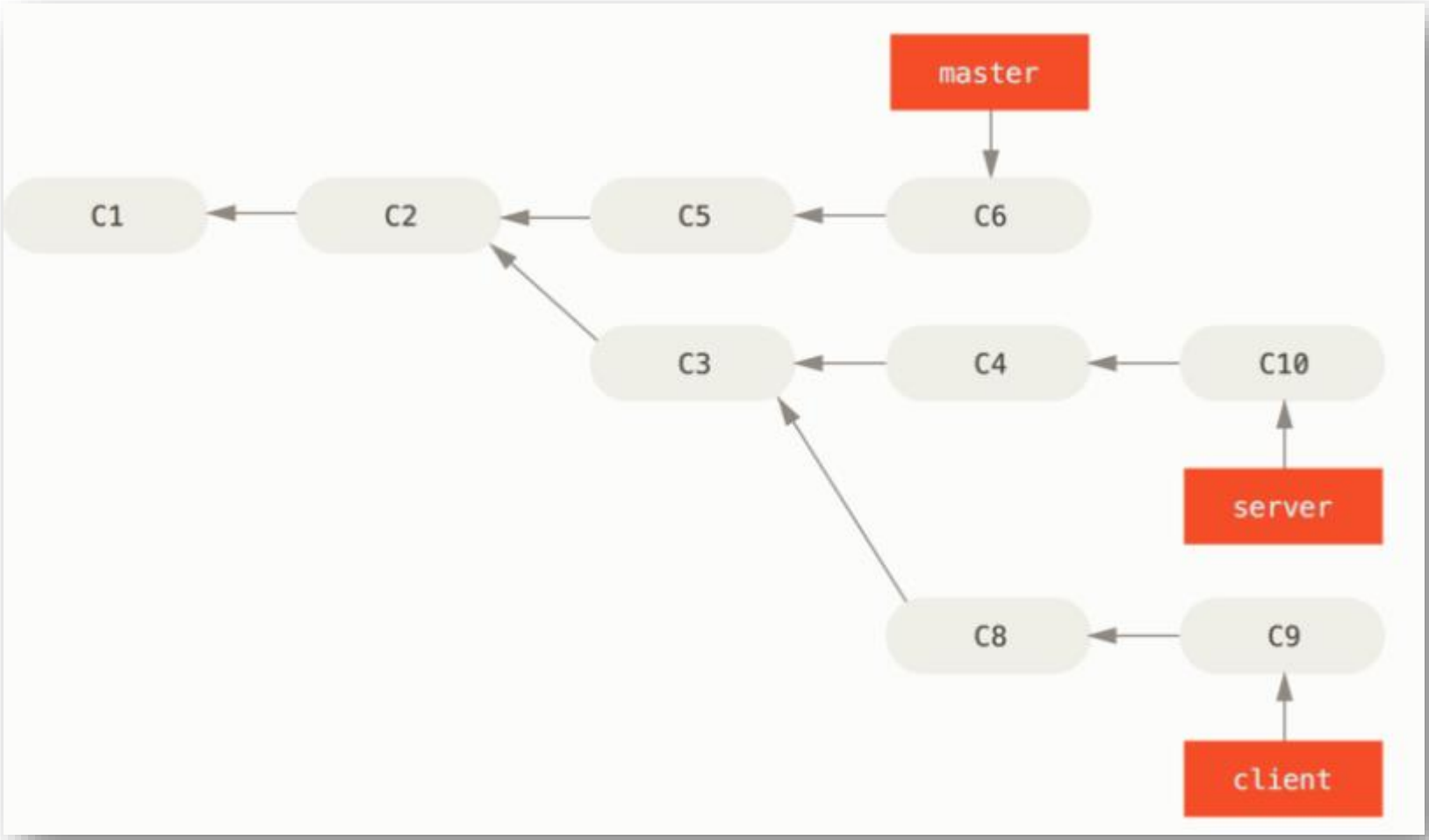
Pouvoir se lancer dans des évolutions ambitieuses en ayant toujours la capacité de revenir à une version stable que l'on peut continuer à maintenir indépendamment.

Pouvoir tester différentes implémentations d'une même fonctionnalité de manière indépendante.

Structure des branches

- Exemple présentant la cohabitation de 2 branches (server, client) en plus de la branche principale (main).

- Plusieurs commits ont été réalisés à partir des différentes branches.



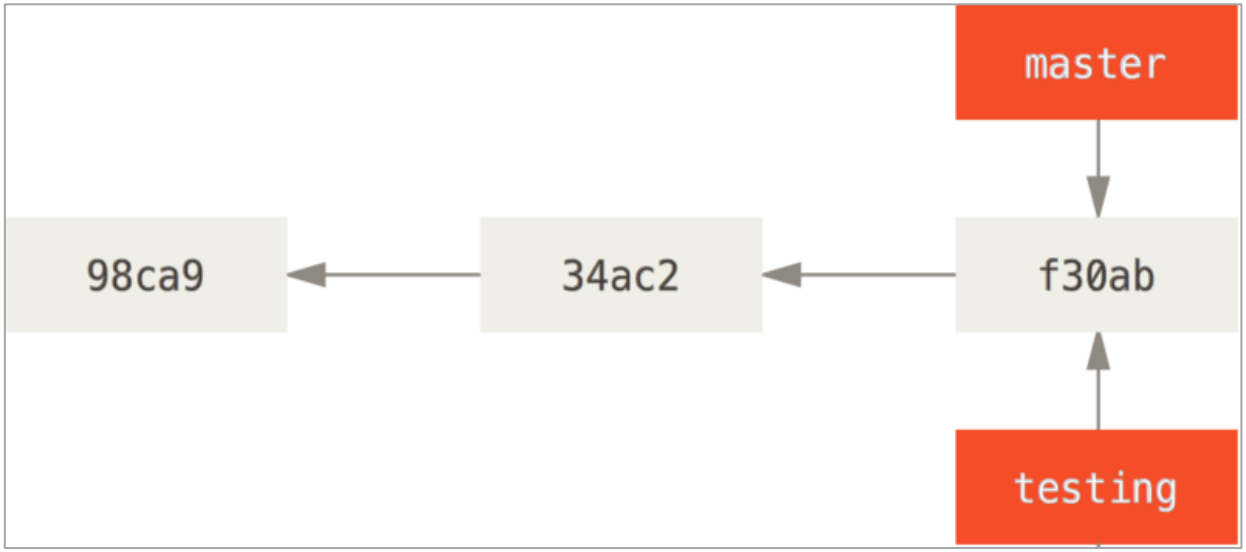
Manipulation des branches

Pour créer une nouvelle branche, on utilise la commande :

```
git branch nom_branche

git checkout -b nom_branche  //Crée une branche et l'active
```

```
git branch testing
```



Pour savoir sur quelle branche vous vous trouvez, git utilise le pointeur HEAD.

Le basculement (switching) d'une branche vers l'autre passe par la commande :

```
git checkout nom_branche
```

Manipulation des branches

On peut vérifier cela facilement grâce à la commande git log qui vous montre vers quoi les branches pointent.

Il s'agit de l'option --decorate.

```
git log --oneline --decorate
```


```
git status
```

Vous donne la branche courante

```
>>
aaa7a7e (HEAD -> develop, origin/main, origin/develop, main) branche développement 2
a73dd25 Nouvelle branche développement
f6610e9 Modification fichiers POO
9f56e49 Ajout dossier POO
71308a7 Ajout Partiel
a332280 Modif Tableau2
4e5db54 Modif tableau1
b866ab9 Ajout creationSpec
0c74f06 Ajout des fchiers
2fcd9c9 First Commit
```

Fusion des branches

- La fusion est l'opération la plus courante sur les branches est LA FUSION.
- Quand la commande git merge se lance, Git détermine systématiquement l'algorithme de merge et va merger la branche de fonctionnalité dans la branche principale.
- Lorsque une fusion est lancée dans Git, les différentes étapes de modification de votre code sont rassemblées dans une seule chronologie séquentielle.
- Pour faire une fusion, il y a différentes étapes à suivre pour que la fusion soit faite sans bug :

Confirmer la branche cible

- Lancer la commande **git status** pour s'assurer que le HEAD pointe bien vers la bonne branche.
- Vous pouvez aussi vous rendre directement dans la branche cible grâce à la commande git checkout.

Fusion des branches

Mettre à jour les branches

- On doit mettre à jour la branche cible afin d'avoir les derniers changements distants. Pour faire cela, il faut exécuter la commande **git fetch**.

- Et une fois cette étape faite, il faut s'assurer que la branche principale soit à jour grâce à la commande **git pull**.

Fusionner

- Fusionner après avoir fini avec les étapes de préparation à la fusion évoquée ci-dessus. C'est là qu'il faut lancer la commande **git merge**.

```
# Démarrer une nouvelle fonctionnalité
git checkout -b nouvelle-fonctionnalité main
# Éditer quelques fichiers
git add
git commit -m "Démarrer une fonctionnalité"
# Éditer quelques fichiers
git add
git commit -m "Finir une fonctionnalité"
# Fusionner dans la branche new-feature
git checkout main
git merge nouvelle-fonctionnalité
git branch -d nouvelle-fonctionnalité
```

Commandes de manipulation des branches

Pour supprimer une branche, on utilise la commande :

```
git branch -d nom_branche
```

Fusionner une branche dans la branche courante

```
git merge nom_branche
```

Pousser des modification vers une branche

```
git push –u origin nom_branche
```

Afficher les différences entre deux branches

```
git diff nom_branche1...nom_branche2
```

Lister les branches distantes

```
git branch -r
```