Outils libres pour le développement logiciel

1- Le workflow gitflow est une architecture qui va permettre de séparer le travail notamment de développeur. Gitflow va permettre de mieux organiser le travail, grâce à la création de nombreuses branche qui vont permettre une meilleur structure Le but étant de séparer le plus possible le travail et de toucher le moins possible la branche master .

2- Les avantages sont qu'elle permet de conserver tout les fichiers de chaque version et elle est distribué. Il permet de travailler plus efficacement c'est-à-dire que chaque branche à un rôle spécifique. Tout se passe sur un seul repository donc c'est presque impossible de casser un projet.Le développement est aussi plus claire. On a un gain de temps car moins on évite des conflits et des problèmes dû à son architecture et sa complexité. On a la possibilité de développer chacun de son côté sur des features. C'est un modèle qui est très organisé.

3- Les inconvénient sont qu'il est compliqué avec la création d'un historique qui est très long et complexe.On ne peut pas empêcher un développeur qui à fait une erreur de développer sur la branche master ou develop donc des problèmes de restriction.

4-
Feature: Cette branche permet de commencer le travail pour une ou de nouvelles fonctionnalités . La nouvelle branche à partir de la branche develop et tout les commits seront réalisé dessus, une fois terminé on va fusionner avec la branche develop.

Hotfix : Cette branche va permettre de publier rapidement une correction depuis la branche master, cette correction qui doit être validé. Une fois validé, les modifications vont être fusionnées sur les branches master de develop.Elle va aider à corriger surtout les bugs majeurs directement sur la branche master.

Release: Cette branche va permettre de faire la liaison entre la branche develop et la branche master. Une fois qu'on aura réalisé toutes les fonctionnalités et qu'elles seront dans la branche develop, on va utiliser la release pour mettre à jour la branche Master. Il faut pas oublier que la branche release est directement créée à partir de la branche develop.

Develop: Cette branche va permettre de sauvegarder l'historique du code complet.Cette elle qui va servir de transition c'est-à-dire qu'elle va recevoir tout les modifications. Cette branche est interdit aux développeurs pour travailler car elle doit rester stable.

Master : cette branche est principal, c'est-à-dire qu'elle est la plus importante, elle doit être stable et propre. C'est sur cette branche que les nouvelles versions seront envoyées. Cette branche aussi est interdit aux développeurs pour coder directement. C'est aussi sur cette branche que les versions seront montrées aux clients.

5. Pour créer le tag sachant que je suis sur la branche develop on s'y rend , je vais faire :
- git tag -a <tag_name> -m message
- git push --tags
pour envoyer sur le dépôt distant

6. La branche pour les modification est la branche hotfix donc on doit s'y rendre. Donc je suis sur la branche feature je finis ma fonctionnalité puis je la merge, j'envoie sur la branche de développement et je supprime la branche feature donc :
git merge feature/<nom> --no-ff
git branche -d feature/<nom>

On va créer la branche à partir de dévelop et s'y rendre
- git checkout -b hotfixx/<name> develop
On modifie puis :
- git commit -m "modif prod"
- git checkout master
- git merge hotfix/<name> --no-ff
- git tag -a <name> "messge"
- git merge hotfix/<version>--no-ff
- git branch -d hotfix <name>

7. Les commandes sont :
git checkout master
git merge release/<version> --no-ff

On peut également rajouter un tag :
git tag <version>

Puis on supprime la banche :
git branch -d release/<version>

8. La commande git stash permet de d'enregistrer de manière temporaire tous les fichiers suivi de la version qui a été modifié
La commande qui permets d'avoir un retour en arrière est git stash pop.