Abdourahamane Balde

Master 1 : Informatique

Outils libres pour le développement logiciel

1- Définir le workflow gitflow, pourquoi on l'utilise ?

Définition : Gitflow est un ensemble de règles simples qui se basent sur le modèle de branchement pour Git et utilise le fonctionnement par branche de Git.

Utilisation : Il s'adapte à la collaboration entre les développeurs selon la dimension de l'équipe de développement.

2- Quels sont les avantages du workflow gitflow ?

- **Travail en parallèle** : on travail sur les branches et pas directement sur le code principal déjà validé
- **Collaboration** : plusieurs développeurs peuvent collaborer facilement sur la même fonctionnalité ou service.
- **Préparation d'une nouvelle version** : Dès que toutes les nouvelles fonctionnalités sont terminées, automatiquement la nouvelle version du produit sera disponible.
- **Supporte la correction d'urgence** : A partir d'une branche déjà utilisé pour une version on peut faire des corrections d'urgence sans risque.

3- Quels sont les inconvénients du workflow gitflow ?

- **Complexité** : Il faut tout d'abord forme les membres de l'équipe comment utiliser correctement le gitflow.
- **Trop des étapes** : Il faut plusieurs étapes supplémentaires pour obtenir un code en production.

- **Difficulté d'un déploiement continu** : Décourager un modèle de déploiement continu

4- Définir et donner l'utilité des branches : Feature, Hotfix, Release, Develop, Master

- **Feature** : Cette branche utilisée pour les nouvelles fonctionnalités à ajouter ou les corrections de bugs non urgents.
- **Hotfix** : Cette branche utilisée pour créer des correctifs d'urgence pendant le processus de développement.
- **Release** : Cette branche est créée lorsqu'il est temps de créer une nouvelle version du produit, cette branche est créée à partir de la branche **Develop**, généralement le code d'une branché release passe à l'étape de test.
- **Develop** : Une fois une version est terminée (**Release**) est mergé dans une branche **Develop** pour continue le cycle de développement
- **Master** : Une fois une version est terminée (**Release**) est mergé dans une branche **Master**, qui suit uniquement le code publié.

5- Donner les commandes git pour créer un tag, sachant que vous êtes sur la branche Develop

git tag -l : pour afficher la liste des versions (exemple release 0.1.0)
git flow release start 0.2.0 : pour suivre la nouvelle release
git flow release finish 0.2.0 : finalizer la nouvelle release

6- Vous êtes sur une branche Feature en train de finaliser un développement, on vous informe qu'il y a un bug de prod à corriger très rapidement. Donner les commandes git pour corriger le problème de prod en respectant le workflow git.

- git flow hotfix start '0.1.0' : Exemple bug d'urgence sur la version 0.1.0
- on fait la correction
- git flow hotfix finish '0.1.0' : Merger la branche dans Master et develop et changer sur master

7- Donner les commandes git à exécuter après la validation de la branche release pour passer en prod

- git merge release-branch
- push origin master
- puis faire déploiement en prod

8- A quoi sert la commande git stash, donner la commande qui permet d'avoir un retour arrière de git stash

- La commande **git stash permet** de cacher (les fichiers/dossiers à ne pas commenter et pusher) et enregistrer l'état actuel du répertoire de travail càd après l'exécution de la commande stash la commande git status affiche que le dépôt du travail est propre (pas de modification) maison peut avoir un retour arrière pour ces modifications.
- Pour faire retour en arrière : git stash pop (en supprimant l'archive de stash) . ou git stash apply (en gardant l'historique de strash)