MESSAOUI Lydia

1) Le workflow Gitflow est une stratégie qui permet l'utilisation efficace de git, c'est un ensemble d'extensions git permettant de mettre en place un cadre de travail pour les developpeurs de faire des opérations de changement de haut niveau, et permettant de gerer le versionnage du projet.
En l'adaptant, on permet le fonctionnement par branches de Git dont les deux branches principales sont "master" et "develop", les developpeurs sont interdits d'ecrire dessus. Trois autres branches sont didiées à travailler dessus qui sont "feature", "release" et "hotfix".

On l'utilise car Git est généralement pas utilisé d'une manière efficace et devient contre-productif, ce qui se traduit par des conflits "commit/merge" et des differents conflits qui peuvent se produire lors de la création de nouvelles fonctionnalités dans le projet ou lors de la correction des bugs, etc

2) Quels sont les avantages du workflow gitflow
- Il permet d'adopter de bonnes pratiques de developpement.
- Il permet d'eviter la desorganisation du système lors de la correction.
- Il permet une correction plus rapide.
- Il permet la mise en place de logicielles permettans l'automatisation des developpements.

3) Quels sont les inconvénients du workflow gitflow
- Compliqué à l'utilisation, on doit former les utilisateurs avant qu'ils puissent l'utiliser.
- On doit effectuer de longs processus pour qu'on puisse faire marcher le code (on doit merger dans "develop" ensuite créer une branche release depuis laquelle on va deployer, etc).
- Pas de "rebasing".
- On ne peut pas travailler avec quand on a plusieurs repositories.

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

Feature c'est la creation d'une branche a partir de la branche develop, on travaille dessus et on commit et on fusionne avec la branche develop à la fin.
Elle est utile lorsqu'on developpe une nouvelle fonctionnalité.

Hotfix elle permet une publication rapide d'une correction depuis la branche master, les modifications apportées apres la correction sont ensuite fusionnées sur les branches "develop" et "master".

Release Elle crée une liaison avec la branche develop et la branche, generalement cette branche contient des commits pour preparer une nouvelle release qui peut etre par exemple le changement du numéro de version dans le fichier readme.

Develop c'est une branche principale sur laquelle travailleront les developpeurs.

Master c'est une branche principale reliée à la mise en production du projet.

5) On se derige vers la branche "develop", et on crée le tag avec la commande:
$ git tag -a <tag_name> -m "message"
$ git push –tags

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.

- On crée la branche "hotfix" avec la commande:
$ git checkout -b hotfix-x.x.x master

- On fait notre correction.
- On commit les changements dans "master" et "develop"
$ git commit -m "message indiquant la corretion du probleme"

$ git merge --no-ff hotfix-x.x.x git tag -a x.x.x -am "message"
$ git checkout develop git merge --no-ff hotfix-x.x.x
-On supprime la banche du hotfix avec:
$ git branch -d hotfix-x.x.x

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

$ git merge release prod

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

git stash permet de stocker (temporairement) l'etat du répertoire de travail au moment actuel et l'index, le temps qu'on effectue d'autres taches dans un répertoire de travail propre. Donc, elle enregistre les modifications locales et rétablit le répertoire de travail quand on revenient pour réeappliquer les taches.
Pour le retour en arriere on fait
$ git stash pop
ou
$ git stash apply –index si on souhaite garder l'état des fichiers.