Prenom : Yanis

Nom : Hammoudi

1- Definition Gitflow :
GitFlow est un modèle de branchement Git alternatif qui requière l'utilisation de branches de fonctionnalités et également de plusieurs branches principales, c'est-à-dire séparer le plus possible le travail pour toucher le moins à la branche MASTER

Pourquoi on l'utilise :
On utilise GitFlow pour plusieurs raisons , simplifie le développement parallèle car il isole le nouveau développement du projet publié
Il est utilisé pour les projets qui ont un cycle de publication planifié et pour la meilleure pratique DevOps de livraison continue

2- Quels sont les avantages du workflow gitflow :
Git Flow possède de nombreux avantages , en effet il permet de travailler plus efficacement , en ''assignant'' des rôles qui sont différents à des branches possédant des noms spécifiques , on peut mettre des bonnes pratiques de développement , permet de faire des correctifs rapidement , posséde des extensions sur les outils git les plus utilisés , il est idéal si on souhaite garder plusieurs versions en production et offre un canal dédié pour les correctifs en production .

3- Quels sont les inconvénients du workflow gitflow ? :
Malgré ses nombreux avantages , Git Flow a également des inconvénients , L'historique de Git devient illisible , la livraison continue et l'intégrations sont considérées comme étant difficile car la séparation de la banche master et la branche develop est considérée comme redondante , GitFlow est déconseillé si on souhaite avoir une seule version en production

4- Définir et donner l'utilité des branches : Feature, Hotfix, Release, Develop, Master :
Feature : Les branches Features sont les branches les plus courantes dans le Workflow de Git flow , quand on travaille sur une nouvelle fonctionnalité , on crée une nouvelle Feature branche depuis la branche develop, puis on fusionne nos modifications dans la branche Develop lorsque la fonctionnalité est terminée et correctement révisée.
Hotfix : la branche Hotfix (branche de correctifs) est utilisée pour apporter rapidement les modifications nécessaires dans la branche principale
Release : La branche Release (branche de publication) est utilisé lorsque l'on souhaite préparer de nouvelles versions de production , ce qu'on fait sur cette branche est souvent les finitions et les correctifs mineurs à la publication de nouveaux codes
Develop : la branche Develop (branche de développement) est une branche que l'on crée au début du projet , elle contient du code de pré production avec des nouvelles fonctionnalités développées qui sont en cours de test
Master : la branche master (la branche principale) est une branche qui contient du code prêt , pouvant être publié. C'est une branche créée au début du projet

5- Donner les commandes git pour créer un tag, sachant que vous êtes sur la branche Develop :
Comme on est dans la branche develop , on part du principe que l'on va créer le tag sur cette branche qui sera une référence sur le dernier commit de cette branche

git tag -a <tag_name> -m "message"
git push --tags

6- Donner les commandes git pour corriger le problème de prod en respectant le workflow git :

On commence par créer la branche du hotfix :

git checkout -b hotfix-x.x.x master

Ensuite,on fait notre correction

git commit -m "Fixed severe production problem"

Enfin on commit dans master et dans develop

git checkout master

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

à la fin on supprime la banche du hotfix

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 checkout Master
git merge Release

8- A quoi sert la commande git stash :

Git stash permet d'enregistrer (stocker) de manière temporaire les modifications apportées au travail , pour pouvoir travailler sur autre chose , uis revenir et les réappliquer plus tard .

Donner la commande qui permet d'avoir un retour arrière de git stash :
Git stash pop