NGUYEN Tuan Hung – 21006891
Outils libres pour le développement logiciel

1. Définir le workflow gitflow, pourquoi on l'utilise
   - Le workflow gitflow est une extension de git qui gère les dépôts plus faciles et plus efficaces en se basant sur le modèle de branchement de Vincent Driessen. Le workflow gitflow définit une structure de branchement stricte autour de la version d'un projet. Cela fournit un autre cadre puissant pour la gestion de grands projets.
   - Dans le cas où de nombreux développeurs sont participés dans le développement d'un projet depuis longtemps, mais si les règles de fonctionnement ne sont pas convenues et décidées, il est normal d'avoir un conflit ou une erreur lors de la fusion. Par conséquent, pour minimiser les erreurs lors de l'utilisation de Git, il est nécessaire d'utiliser le workflow gitflow.

2. Quels sont les avantages du workflow gitflow
   - Cela n'affecte pas la progression du développement
   - Les branches sont utilisées de manière relativement méthodique
   - Permet de s'assurer que la version en ligne est stable
   - Diminuer les erreurs lors de la fusion

3. Quels sont les inconvénients du workflow gitflow
   - Le maintien de Master et Develop en même temps n'est généralement pas nécessaire, car dans de nombreux cas le contenu de Master et Develop est similaire.
   - Surtout si vous voulez restaurer le commit de quelqu'un, vous trouverez cela un peu difficile à faire.
   - Dans le processus de travail, vous déplacez entre les branches de travail, parfois vous ne déplacez pas, faites-la commit à la mauvaise branche, vous devez faire des rolback.

4. Définir et donner l'utilité des branches : Feature, Hotfix, Release, Develop, Master
   - Master : La branche Master est disponible dans git, qui contiennent le code d'initialisation de l'application et les versions qui sont prêtes pour publier aux utilisateurs.
   - Hotfix : Basé sur la branche Master, il permet de corriger rapidement des bugs ou de réaliser des configurations spéciales uniquement disponibles en production.
   - Release : Avant de publier un logiciel, une équipe de développement doit être créée pour effectuer les tests finaux avant de diffuser le produit afin que les utilisateurs puissent l'utiliser.
   - Develop : Initiée à partir de la branche master pour sauvegarder tout l'historique des modifications du code source. La branche Develop est la fusion de code de toutes les branches feature.
   - Feature : Basé sur la branche Develop. Chaque fois que nous développons une nouvelle fonctionnalité, nous devons créer une branche Feature pour écrire le code source de chaque fonctionnalité.

5. Donner les commandes git pour créer un tag, sachant que vous êtes sur la branche Develop
   $ git checkout Master
   $ git tag -a v1.0.0 -m "Releasing version v1.0.0"

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 checkout Master
   $ git checkout -b Hotfix Master
   //corriger le bug
   $ git add .
   $ git commit -m "Bug resolu"
   $ git checkout Master
   $ git merge --no-ff Hotfix
   $ git tag -a v1.0.1 -m "corriger bug"
   $ git checkout Develop
   $ git merge --no-ff Hotfix
   $ git branch -d Hotfix

7. Donner les commandes git à exécuter après la validation de la branche release pour passer en prod
   $ git checkout Master
   $ git merge --no-ff Release
   $ git tag -a v2.0.0 -m " Releasing version v2.0.0"
   $ git checkout Develop
   $ git merge --no-ff Release

8. A quoi sert la commande git stash, donner la commande qui permet d'avoir un retour arrière de git stash
   - git stash est utilisé lorsque vous voulez sauvegarder des modifications non validées, souvent très utile lorsque vous voulez passer à une autre branche mais que vous êtes en train de travailler sur la branche actuelle.
   - $ git stash apply