étudiant: **Ilyes ZEROUAL.**
Master 1 Big Data.

Evaluation 2: Outils libres pour le développement logiciel

1. Le workflow Gitflow est un modèle de branchement git développé par Vincent Driessen. En termes de programmation orientée objet, git-flow est une classe héritière de la classe git. Il est basé sur le fait qu'il y ait plusieures branches principales(main, develop) et secondaires(feature, hotfix, release).
   gitflow nous permet de mieux gérer le versionnement du projet et ne pas encombrer la branche main, corriger rapidement et d'une façon efficace les bugs, développée indépendamment les fonctionnalités majeures…
2. les avantages du Gitflow:
   - Il facilite le développement en parallèle.
   - Il permet une meilleure collaboration entre les développeurs.
   - Il facilite la correction des bugs majeurs, en permettant leur correction dans une branche séparée lors d'un sprint.
3. les inconvénients du Gitflow:
   - modèle un peu compliqué.
   - Il rend plus difficile la mise en production continuée.
   - Il rend difficile la lecture de l'historique du projet.
4. définition et utilité des branches:
   - Master: l'une des deux branches principales du Gitflow. Elle sert à archiver les versions officiels des projets (versions mises en production).
   - develop: l'autre branche principale du Gitflow. Elle sert comme une branche d'intégration des fonctionnalités pour la prochaine version qui sera mise en production. Quand toutes les fonctionnalités de la nouvelle version sont prêtes, on fait un merge avec Master.
   - feature: elle sort du branche develop et merger avec develop a la fin. Elle sert a développé de nouvelles fonctionnalités pour la version suivante ou une version future.
   - release: sort du branche develop. mergée avec develop et master à la fin. Elle sert à préparer une mise en production d'une version, corriger les bugs mineurs et préparer les métadonnées.
   - hotfix: sort du branche master. mergée avec develop et master à la fin. Tout comme la branche release, elle sert à préparer une mise en production d'une version. Cependant, elle est créée dans la nécessité de corriger un bug majeur qui se trouve dans la version actuelle.

5. git checkout -b release develop \\on cree une branche release
   git commit -a -m "version number to 2.0" \\on exécute des commites après avoir fait les derniers changements dans la branche release
   git checkout master \\on change vers la branche master

git merge --no-ff release \\on merge la branche master avec la branche release
git tag -a 2.0 \\on crée un tag

6. git checkout master feature \\on switch vers la branche master
   git checkout -b hotfix master \\on crée et switch vers une branche hotfix
   \\on fait les corrections nécessaires
   git commit -m "problèmes corrigés" \\on enregistre les changements
   git checkout master \\on retourne vers la branche master
   git merge --no-ff hotfix \\on merge la branche master avec hotfix
   git tag -a 2.0.1 \\on cree une nouveau tag
   git checkout develop \\on switch vers la branche develop
   git merge --no-ff hotfix \\on merge avec hotfix
   git branch -d hotfix \\on supprime la branche hotfix

7. git checkout master \\on change vers la branche master
   git merge --no-ff release \\on merge la branche master avec la branche release
   git tag -a 2.1 \\on crée un tag
   git checkout develop \\ on retourne vers la branche develop
   git merge --no-ff release \\on fait un merge avec la branche release
   git branch -d release \\on supprime la branche release

8. La commande git stash sert à enregistrer les fichiers du projets temporairement en local pour protéger les changements faits avant un git pull qui peut overwriter ces changements.
   la commande qui permet d’avoir un retour arrière est: git reset.