1 - Exercice Git

Objectif :

- Appliquer les commandes de base de Git pour gérer les modifications apportées à un projet.

Instructions : Veuillez executer les actions git demandées (elle doivent être exécutées dans l'ordre chronologique).

Pour chaque action demandée, veuillez mentionner dans ce fichier la ou les commandes effectuées, sous cette forme:

```
# exemple - ma ou mes commandes ici ...
```

> Uniquement en ligne de commande avec git Bash !

---

1. Créez un nouveau répertoire nommé *exercice_git_base* et initialisez un dépôt Git.
2. Créez un fichier `README.md` et ajoutez-y le contenu *# Mon exercice git 1* dans la branch `main`.
3. Vérifiez l'état du dépôt pour voir les fichiers non suivis.
4. Ajoutez le fichier `README.md` à l'index / au Stage.
5. Faites un commit avec un message approprié.
6. Ajouter votre repo local à un repo distant privé GitHub
7. Synchroniser votre repo distant avec le repo local
8. Créez une nouvelle branche locale appelée `develop`.
9. Changez de branche pour `develop`.

- Quel commande pouvez-vous utiliser pour faire les actions 6 et 7 en use seule commande?

10. Modifiez le fichier `README.md`, ajoutez-y le contenu supplémentaire *## Description de l'exercice*.
11. Vérifiez ce qui a été modifier dans le fichier
12. Ajoutez et commitez les modifications.
13. Poussez les modifications vers le dépôt distant.
14. Créez une Pull Request pour fusionner `develop` dans `main` sur GitHub. Comment avez-vous procédé?
15. Fusionnez la Pull Request après révision. Comment avez-vous procédé?
16. Revenez à la branche `main` et récupérer les modifications.

17. Créez le tag `v1.0` avec comme commentaire `Version 1.0` pour marquer une version stable.
18. Poussez les tags vers le dépôt distant.

> Verifier sur GitHub si vous avez bien le tag `v1.0`.

19. Affichez l'historique des commits.
20. Affichez les différences entre les deux derniers commits.
21. Annulez le dernier commit (sans supprimer les modifications).
22. Supprimez la branche `develop` après fusion.
23. Configurez un alias Git **st** pour simplifier une commande fréquente * git status*.
24. Assurez-vous d'être dans la branche `main`, puis ajoutez-y le fichier `data.txt` avec comme contenu: `Hello World!`.

> Ne pas ajouter cette modification à la **staging area**!

25. Mettez temporairement de côté `stash` pour sauvegarder temporairement les modifications non commitées.
26. Ajouter une branche `feature/new-data` avec `git checkout` et vérifier l'état de votre repo.

> Que constatez-vous ?

26. Listez les sauvegarde `stash` effectué au point 25.
27. Restaurez les modifications mises de côté (avec le `stash` au point 25) dans la branche `feature/new-data`.
28. Vérifiez le statut du repo.