TP 2 : Git

Partie 0 : Prise en main

Étape 1 : Configuration de Git

1. Installer Git :
    - Téléchargez et installez Git depuis https://git-scm.com/.
    - Vérifiez l'installation :
2. Configurer votre identité Git : Définissez votre nom et votre e-mail (ces informations seront visibles dans les commits) :
3. Afficher la configuration : Vérifiez que la configuration a été appliquée correctement :

Étape 2 : Création d'un dépôt local et initialisation de Git

1. Créez un dossier pour le projet :
2. Initialisez un dépôt Git local :
3. Créez un fichier README.md avec le contenu suivant : Ceci est mon premier dépôt Git !
4. Ajoutez et committez ce fichier :

Étape 3 : Connexion à un dépôt GitHub

1. Créez un dépôt distant sur GitHub, nommé MonProjet.
2. Connectez votre dépôt local à GitHub :
3. Poussez les modifications locales vers GitHub :
4. Vérifiez que le fichier README.md est visible sur GitHub.

Partie 1 : Création du projet et configuration du dépôt

1. L'un de vous (Développeur A) doit créer un dépôt distant sur GitHub nommé `SiteWebCollaboratif`. Le lien du dépôt sera partagé avec votre binôme (Développeur B) afin qu'il puisse cloner le projet.
2. Développeur B clone le dépôt distant avec la commande pour récupérer le projet sur son ordinateur.
3. Développeur A crée un fichier « index.html » avec le contenu suivant :

```
<!DOCTYPE html>
<html>
<head>
<title>Mon Site Collaboratif</title>
<link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
<h1>Bienvenue sur notre site collaboratif</h1>
</body>
</html>
```

4. Développeur B crée un fichier `styles.css` avec ce contenu :

```
body { font-family: Arial, sans-serif;
background-color: #f4f4f4; color: #333; }
```

5. Développeur A ajoute et commit le fichier HTML, tandis que Développeur B fait de même avec le fichier CSS. Ensuite, chacun pousse ses modifications sur le dépôt distant.

Création et manipulation des branches pour les fonctionnalités

Vous allez maintenant diviser le travail en deux fonctionnalités : une fonctionnalité de menu et une fonctionnalité de pied de page.

1. Développeur A crée une branche « feature-menu » pour ajouter un menu de navigation au site.
2. Développeur B crée une branche « feature-footer » pour ajouter un pied de page.
3. Développeur A : Ajoute un menu dans le fichier « index.html ».

```
<nav>
 <ul>
  <li><a href="#">Accueil</a></li>
  <li><a href="#">Contact</a></li>
 </ul>
</nav>
```

- Il ajoute et commite les changements dans la branche « feature-menu ». Ensuite, il pousse ses modifications sur le dépôt distant.

4. Développeur B : Ajoute un pied de page dans le fichier « index.html ». Ensuite, il ajoute et commite ses modifications dans la branche « feature-footer » puis pousse ces modifications sur le dépôt distant.

```
<footer>
 <p>© 2024 Mon Site Collaboratif</p>
</footer>
```

Collaboration, Conflits et Rebase

Après avoir tous deux travaillé sur la même section du fichier « index.html », vous devez maintenant intégrer vos modifications tout en évitant de polluer l'historique avec de multiples commit de merge.

1. Merge avec conflits :

- Développeur A décide de fusionner la branche « feature-menu » dans « main ». Il bascule sur « main », fusionne et pousse les modifications.
- Développeur B tente ensuite de fusionner la branche « feature-footer » dans « main », mais un conflit apparaît. Le pied de page et le menu se trouvent dans des sections proches du fichier « index.html ».

2. Résolution des conflits : Utiliser Git pour résoudre le conflit manuellement dans le fichier « index.html », puis valider la fusion. Les deux développeurs doivent s'assurer que les deux sections, menu et pied de page, sont correctement intégrées.
3. Pour garder un historique propre, Développeur B peut utiliser « git rebase » au lieu de « git merge » pour intégrer les modifications de la branche « main » dans « feature-footer » avant de fusionner.

Utilisation de Git Stash pour suspension de travail

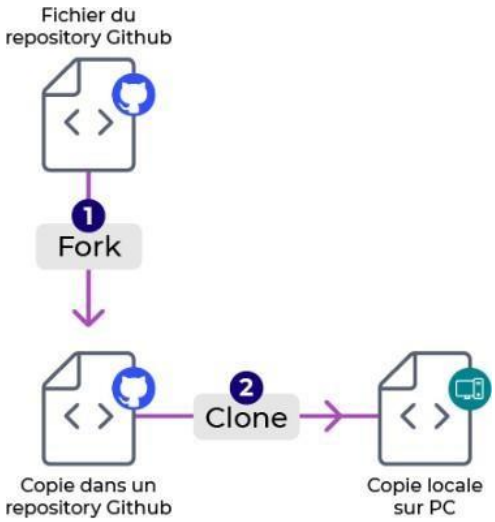
Développeur A travaille sur une nouvelle fonctionnalité mais reçoit une demande urgente pour résoudre un bug de design. Il doit mettre de côté son travail temporairement.

Utiliser « git stash » pour sauvegarder les modifications en cours : Développeur A stashe son travail sur la branche « feature-menu » pour basculer rapidement sur une autre branche de correction. Puis, il applique le stash après avoir résolu le bug pour récupérer ses modifications.

Partie 2 : Contribuer à un projet open source

Le projet Open Transport est disponible sur GitHub à cette adresse : https://github.com/OpenClassrooms-Student-Center/7688581-Expert-Git-GitHub.

1. Allez sur GitHub, Forkez puis clonez en local le projet Open Transport. Forker permet de faire une copie du repository distant de manière à travailler sans impacter l'original. Le fork est complètement isolé du projet original. Pour envoyer une modification auprojet originel, vous devez soumettre un pull request.



Pour permettre à tous de contribuer, Open Transport utilise la fonctionnalité GitHub **Issues**. Les issues permettent d'identifier une tâche à réaliser sur le projet. Elles peuvent représenter de nouvelles fonctionnalités, des corrections, des problèmes à résoudre, etc.

2. Parcourez la liste des issues du projet Open Transport. Commencez par traiter la première issue, qui correspond à une erreur de texte dans le fichier README.md.

- Modifiez en local le fichier README.md.
- Faites un git add, un git commit (avec le numéro de l'issue dans le message) et un git push pour envoyer la modification sur votre repository distant (le fork).
- Créez une pull request via l'interface graphique de GitHub.

Le travail collaboratif permet à plusieurs développeurs de manipuler les fichiers du projet. Mais que se passe-t-il si 2 développeurs modifient la même ligne d'un fichier ? Comment Git choisit-il la modification à retenir ?

La réponse est simple : Git ne choisit pas. Et c'est justement sa limite ! il va vous demander de faire ce choix vous-même. Comment ? En créant un conflit!

Pour simuler un conflit, suivez les étapes suivantes :

- git pull ;

- modifier en local le fichier README.md, la ligne 20 par exemple ;
- git add README.md ;
- git commit -m “ Modification d’une ligne" ;
- modifier via l’interface web de GitHub le fichier README.md (Modifier la même ligne avec untexte différent de la modification locale) ;
- git push ;
- git pull.

Un **conflit** apparaît !

En effet, la ligne 20 du même fichier README.md a été modifiée par un de vos collaborateurs. Git ne sait plus quoi faire car il ne peut pas deviner quelle est la bonne ligne entre les 2. Doit-il conserver la version de votre collaborateur ou la vôtre ?

- Pour comprendre le conflit obtenu, utilisez la commande **git status** et analysez le résultat affiché. git status est une commande précieuse car elle nous informe de l’état de notre repository Git local.
- Résoudre un conflit signifie prendre une décision. Dans notre cas, quelle version de la ligne 20 dois- je conserver ? La version qui vient du dépôt distant ? Celle qui a été écrite en local ? Ou bien une version hybride entre les 2 ?

En résumé, nous avons 3 possibilités :

- Conserver la version locale.
- Conserver la version distante.
- Modifier le fichier en conflit et en faire une 3e version qui sera ensuite conservée.

3. Ouvrez le fichier README.md, et constatez le changement du contenu (Git a modifié le fichier pour y écrire le détail du conflit.).

4. Résolvez ensuite le conflit **avec chacune des possibilités** citées précédemment.

   - Pour conserver la version distante, utilisez la commande : **git checkout --theirs README.md**.
   - Pour conserver la version locale, utilisez la commande : **git checkout --ours README.md**.
   - Pour corriger le conflit manuellement, ouvrez le fichier et modifier son contenu. Cette modification implique d’enlever les lignes ajoutées automatiquement par Git, et de conserver le texte de votre choix.

Conservez la version locale puis pousser ces modifications dans le dépôt distant.

Une fois le conflit résolu, recréez le conflit (Refaites les étapes de 1 à 7 en modifiant une autre ligne ou un autre fichier), puis résolvez le avec une autre méthode.