TP : Collaboration Git à Trois

Travail Collaboratif avec Gestion de Conflits

Présentation du TP

Coordonner vos contributions, gérer les branches et surtout à résoudre les conflits qui peuvent survenir lors du développement en équipe.

Objectifs pédagogiques

À l'issue de ce TP, vous serez capable de :

- Collaborer efficacement sur un projet Git à plusieurs
- Gérer les branches et les Pull Requests en équipe
- Identifier et résoudre des conflits Git
- Communiquer et coordonner vos modifications avec vos coéquipiers
- Comprendre l'importance de la synchronisation dans le travail collaboratif

Organisation de l'équipe

Désignez les rôles suivants dans votre équipe :

- **Développeur A (Chef de projet) :** Crée le dépôt et gère les Pull Requests
- **Développeur B (Développeur 1) :** Travaillera sur le HTML et la page d'accueil
- **Développeur C (Développeur 2) :** Travaillera sur le CSS et la page contact

Phase 1 : Configuration Initiale (Tous ensemble)

Tous les Développeurs : A, B et C

Configurer le projet. Suivre les instructions du Développeur A.

Étape 1.1 : Développeur A - Création du projet

1. **Créer le dépôt sur GitHub** :
   - Connectez-vous à GitHub
   - Cliquez sur « New » pour créer un nouveau dépôt
   - Nommez-le : projet-equipe-X (remplacez X par votre numéro d'équipe)
   - Cochez « Add a README file »
   - Cliquez sur « Create repository »

2. **Inviter les collaborateurs** :
   - Settings → Collaborators → Add people
   - Ajoutez les noms d'utilisateur GitHub de B et C
   - Envoyez les deux invitations

3. **Communiquer l'URL du dépôt** : Partagez l'URL du dépôt avec B et C (ex: https://github.com/VotreNom/projet-equipe-X)

Étape 1.2 : Développeurs B et C - Accepter et cloner

4. **Accepter l'invitation** : Vérifiez votre email et acceptez l'invitation à collaborer sur le projet
5. **Cloner le projet** : Sur votre machine, ouvrez un terminal et exécutez :

```
git clone [URL-du-depot]
cd projet-equipe-X
```

Étape 1.3 : Tous ensemble - Premier commit collaboratif

Objectif : Chacun modifie le README pour se présenter, créant ainsi votre premier conflit contrôlé.

6. **Développeur A** : Modifiez le README.md en ajoutant :

```
# Projet Équipe X
## Membres de l'équipe
- [Votre nom] - Chef de projet
```

Puis commitez et poussez :

```
git add README.md
git commit -m "Ajout du chef de projet"
git push
```

7. **Développeurs B et C** :
   - **ATTENDEZ** que le développeur A ait terminé
   - Récupérez les modifications :

```
git pull
```

   - Ajoutez chacun votre nom à la liste :

```
- [Nom de B] - Développeur Frontend
- [Nom de C] - Développeur CSS
```

   - Commitez et poussez (chacun votre tour) :

```
git add README.md
git commit -m "Ajout de [votre nom]"
git push
```

Phase 2 : Développement du Mini-Projet

Vous allez maintenant créer un petit site web composé de trois pages : une page d'accueil (index.html), une feuille de style (style.css) et une page de contact. Chaque membre de l'équipe travaillera sur une partie spécifique.

Répartition des tâches

Voici comment vous allez vous répartir le travail :

Développeur A	Développeur B	Développeur C
Structure HTML de base	**Page d'accueil**	**Page de contact**
Header et navigation	Contenu principal	Formulaire
Fichier : `index.html`	**Fichier :** `index.html`	**Fichier :** `contact.html`
Tous : `style.css` (chacun stylise sa partie)		

Étape 2.1 : Développeur A - Structure de base

1. **Créer une branche** :

```
git checkout -b structure-html
```

2. **Créer index.html** avec la structure de base :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<meta charset="UTF-8">
<title>Notre Projet</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<header>
<h1>Bienvenue sur notre site</h1>
<nav>
<a href="index.html">Accueil</a>
<a href="contact.html">Contact</a>
```

```
</nav>
</header>
<main>
<!-- Contenu à ajouter par développeur B -->
</main>
</body>
</html>
```

3. **Créer style.css** vide (pour l'instant)
4. **Commit et push** :

```
git add .
git commit -m "Ajout structure HTML de base"
git push -u origin structure-html
```

5. **Créer une Pull Request** sur GitHub et la fusionner dans main
6. **Informer B et C** que la structure de base est prête

Étape 2.2 : Développeur B - Page d'accueil

7. **Récupérer les modifications** :

```
git checkout main
git pull
```

8. **Créer votre branche** :

```
git checkout -b page-accueil
```

9. **Modifier index.html** : Ajoutez du contenu dans la balise <main> :

```
<main>
<h2>À propos de nous</h2>
<p>Nous sommes une équipe de 3 développeurs passionnés.</p>
<h3>Nos services</h3>
<ul>
<li>Développement Web</li>
<li>Design</li>
<li>Consulting</li>
</ul>
</main>
```

10. **Commit et push** :

```
git add index.html
git commit -m "Ajout contenu page accueil"
```

```
git push -u origin page-accueil
```

11. **Créer une Pull Request** et attendre la review de A

Étape 2.3 : Développeur C - Page de contact

12. **Récupérer les modifications** :

```
git checkout main
git pull
```

13. **Créer votre branche** :

```
git checkout -b page-contact
```

14. **Créer contact.html** :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<meta charset="UTF-8">
<title>Contact</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<header>
<h1>Contactez-nous</h1>
<nav>
<a href="index.html">Accueil</a>
<a href="contact.html">Contact</a>
</nav>
</header>
<main>
<form>
<label>Nom : <input type="text" name="nom"></label>
<label>Email : <input type="email" name="email"></label>
<button type="submit">Envoyer</button>
</form>
</main>
</body>
</html>
```

15. **Commit et push** :

git add contact.html

git commit -m "Ajout page de contact"

git push -u origin page-contact

16. **Créer une Pull Request** et attendre la review de A

Étape 2.4 : Développeur A - Review et fusion

17. Examinez les Pull Requests de B et C
18. Laissez des commentaires constructifs
19. Fusionnez les deux PR dans main

Phase 3 : Gestion de Conflits (Expérience pratique)

Dans cette phase, vous allez volontairement créer des conflits en modifiant le même fichier CSS simultanément. Cela vous permettra d'apprendre à résoudre les conflits, une situation très courante dans le développement collaboratif.

Étape 3.1 : Création volontaire de conflits

Consigne : Tous les trois, vous allez modifier le fichier style.css EN MÊME TEMPS pour créer un conflit.

Tous ensemble (A, B et C)

1. **Synchronisez-vous** :

git checkout main

git pull

2. **Chacun crée sa branche** :
   - **A :** git checkout -b style-header
   - **B :** git checkout -b style-main
   - **C :** git checkout -b style-colors

3. **Chacun modifie style.css** (SANS communiquer sur ce que vous faites) :

Développeur A ajoute :

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

}

```
header {
background-color: #333;
color: white;
padding: 20px;
}
```

Développeur B ajoute :

```
body {
font-family: 'Helvetica', sans-serif;
line-height: 1.6;
}
main {
padding: 40px;
max-width: 1200px;
margin: 0 auto;
}
```

Développeur C ajoute :

```
body {
background-color: #f0f0f0;
color: #333;
}
h1, h2 {
color: #2E5090;
}
```

4. **Chacun commit et push** (dans cet ordre : A, puis B, puis C) :

```
git add style.css
git commit -m "Stylisation [votre partie]"
git push -u origin [votre-branche]
```

5. **Créez chacun une Pull Request** vers main

Étape 3.2 : Résolution des conflits

Que va-t-il se passer ? La première PR pourra être fusionnée sans problème. Mais les suivantes vont créer des conflits car vous avez tous modifié le même fichier !

Développeur A - Fusionner la première PR

6. Fusionnez votre propre PR (celle de la branche style-header)
7. Cette fusion va réussir sans conflit

Développeur B - Résoudre le conflit

Maintenant, essayez de fusionner votre PR. GitHub va détecter un conflit !

8. **En local, mettez à jour main** :

git checkout main

git pull

9. **Revenez sur votre branche** :

git checkout style-main

10. **Fusionnez main dans votre branche** :

git merge main

→ **CONFLIT !** Git va vous indiquer que style.css est en conflit.

11. **Ouvrez style.css** : Vous verrez des marqueurs de conflit comme ceci :

<<<<<<< HEAD

Votre version

=======

Version de main

>>>>>>> main

12. **Résolvez le conflit** : Combinez intelligemment les deux versions en gardant le meilleur des deux. Supprimez les marqueurs <<<<<<<, ======= et >>>>>>>

13. **Finalisez la résolution** :

git add style.css

git commit -m "Résolution conflit style.css"

git push

14. **Fusionnez votre PR** : Maintenant que le conflit est résolu, vous pouvez fusionner votre PR sur GitHub.

Développeur C - Résoudre le deuxième conflit

Répétez exactement les mêmes étapes que le développeur B pour résoudre votre conflit. Cette fois, vous devrez combiner vos modifications avec celles de A ET de B !

Étape 3.3 : Débriefing en équipe

Prenez 10 minutes pour discuter ensemble :

- Qu'avez-vous ressenti lors de la résolution du conflit ?
- Qu'auriez-vous pu faire différemment pour éviter les conflits ?
- Comment mieux communiquer dans le futur ?
- Quelles bonnes pratiques allez-vous mettre en place ?

Bonnes Pratiques Collaboratives

Retenez ces principes essentiels pour le travail en équipe :

- **Communiquez constamment** : Informez votre équipe de ce sur quoi vous travaillez
- **Synchronisez régulièrement** : Faites git pull souvent pour éviter les gros conflits
- **Branches courtes** : Ne gardez pas une branche ouverte trop longtemps
- **Commits atomiques** : Un commit = une modification logique
- **Messages clairs** : Vos coéquipiers doivent comprendre vos commits
- **Revues constructives** : Commentez les PR avec bienveillance et précision
- **Ne pas avoir peur des conflits** : Ils font partie du processus et vous apprennent à collaborer

Félicitations pour ce travail d'équipe !