Méthodologie de travail pour les informaticiens

Utiliser des branches pour gérer son code

Bachelor Cyber 1

Marie Puren - marie.puren@epita.fr

Pourquoi utiliser des branches ?

L'utilisation des branches dans Git et GitLab est une **bonne pratique essentielle** dans le développement informatique, notamment lorsqu'on travaille en équipe.

Isolement des modifications

- Les branches permettent de travailler isolément sur des fonctionnalités spécifiques ou des corrections de bugs sans affecter la branche principale (`main` ou `master`).
- Cela permet d'éviter d'introduire des bugs dans la version stable du projet.

Collaboration efficace

- Les branches facilitent la collaboration entre plusieurs développeurs. Chaque développeur peut travailler sur une branche séparée et, lorsqu'il est prêt, intégrer ses modifications dans la branche principale.
- Cela réduit le risque de conflits et améliore l'efficacité des équipes.

Types de branches

- **Branches de fonctionnalité** : utilisées pour développer de nouvelles fonctionnalités.
- **Branches de correctif** : créées pour corriger des bugs spécifiques.
- **Branches de release** : utilisées pour préparer des versions spécifiques du logiciel.

Commandes clés

Créer une nouvelle branche

```
git branch <nom-de-la-branche>
```

Basculer sur une autre branche

```
git checkout <nom-de-la-branche>
```

Ou bien :

```
git switch <nom-de-la-branche>
```

Commandes clés (suite)

Fusionner une branche dans une autre

```
git merge <nom-de-la-branche>
```

Résoudre un conflit

Lorsque des conflits apparaissent pendant une fusion, Git vous demande de les résoudre manuellement.

Étape 1 : Créer une nouvelle branche

```
git branch <nom-de-la-branche>
```

- Une nouvelle branche est créée pour développer une nouvelle fonctionnalité ou corriger un bug.

Étape 2 : Basculer sur une branche

```
git checkout <nom-de-la-branche>
```

ou

```
git switch <nom-de-la-branche>
```

Étape 3 : Fusionner la branche

```
git merge <nom-de-la-branche>
```

- La branche est fusionnée dans la branche principale (`main` ou `master`).
- Résolution des conflits si nécessaire.

Étape 4 : Résoudre les conflits

Lorsqu'un conflit survient, Git vous indique les fichiers concernés. Vous devrez éditer ces fichiers pour choisir les modifications à garder.

Quand pousser une branche sur un dépôt distant ?

- **Après avoir créé une nouvelle branche localement :** Lorsque vous créez une nouvelle branche et commencez à travailler dessus localement, il est recommandé de la pousser sur le dépôt distant pour que les autres membres de l'équipe puissent y accéder.
- **Avant de faire une Merge Request :** Lorsque votre travail sur la branche est terminé et que vous souhaitez faire une Merge Request, vous devez d'abord pousser la branche sur le dépôt distant.
- **Pour partager le travail en cours :** Même si la fonctionnalité n'est

Commande pour pousser une branche

Pour pousser une branche sur un dépôt distant, utilisez la commande suivante :

```
git push origin <nom-de-la-branche>
```

Merge Requests (MR)

La **Merge Request** (MR) est une étape clé dans les bonnes pratiques de développement collaboratif.

Pourquoi faire des Merge Requests ?

1. Revue de code

- La Merge Request permet à d'autres développeurs (ou à vos enseignants) de **relire** et **vérifier** votre code.
- Cela permet de détecter d'éventuelles erreurs, d'améliorer la qualité du code, et de garantir qu'il respecte les conventions du projet.
- C'est une excellente opportunité d'**apprentissage** grâce aux retours des autres.

Pourquoi faire des Merge Requests ? (suite)

2. Collaborer efficacement

- Une Merge Request facilite la **communication entre les membres de l’équipe**.
- Vous pouvez expliquer vos modifications, ce qui aide les autres à comprendre sans lire tout le code.
- Cela aide à éviter que plusieurs personnes modifient le même fichier en même temps.

Quand faire une Merge Request ?

- **Après avoir terminé une fonctionnalité** : Une Merge Request doit être créée une fois qu'une fonctionnalité ou un correctif est complet.
- **Avant de fusionner une branche dans `main` ou `master`** : Cela garantit que le code est relu avant d'être intégré dans la version stable du projet.
- **Lorsque vous avez besoin de feedback** : Même si la fonctionnalité n'est pas complètement terminée, une Merge Request peut être utilisée pour obtenir des commentaires sur le code.

Étape 1 : Créer une Merge Request

Dans GitLab, allez dans votre projet, sélectionnez votre branche, et cliquez sur **New Merge Request**.

Étape 2 : Décrire les modifications

- Ajoutez une description claire de ce que vous avez modifié.
- Assurez-vous que les autres membres de l’équipe puissent facilement comprendre les changements.

Étape 3 : Révision et commentaires

- Les autres membres de l’équipe peuvent laisser des commentaires et des suggestions.
- Cela permet d'améliorer le code avant qu'il ne soit fusionné dans la branche principale.

Étape 4 : Fusionner la Merge Request

- Une fois approuvée, la Merge Request peut être fusionnée dans la branche principale (`main` ou `master`).
- Cela intègre les modifications de manière sécurisée.