Comment créer une nouvelle branche dans GIT

Git est un système de contrôle de version distribué utilisé pour suivre les modifications du code source pendant le développement de logiciels. Il prend en charge la collaboration, permettant à plusieurs développeurs de travailler simultanément sur différentes parties d'un projet. Git est connu pour sa rapidité, son intégrité des données et sa prise en charge des flux de travail non linéaires.

Le branchement dans Git est une fonctionnalité puissante qui permet à plusieurs développeurs de travailler sur différentes parties d'un projet. simultanément sans interférer les uns avec les autres.

Dans cet article, je vais vous montrer comment créer une nouvelle branche dans le système de contrôle de version Git, ainsi que des exemples et Réponses aux questions fréquemment posées.

Créer une nouvelle branche dans GIT

Le processus de création d'une nouvelle branche GIT se fait en 3 étapes. Les étapes sont les suivantes :

Vérification de votre succursale actuelle

Avant de créer une nouvelle branche, il est important de savoir sur quelle branche vous vous trouvez actuellement. Utilisez la commande suivante :

```
git status
```

Cette commande affichera votre branche actuelle et toutes les modifications non validées.

Création de la nouvelle branche

Pour créer une nouvelle branche et y basculer, utilisez la nouvelle `git paiement` commande avec le `-b` option, suivi du nom du branche :

```
git checkout -b [branch-name]
```

Remplacez [branch-name] par le nom de la branche souhaitée.

Alternativement, vous pouvez créer une branche sans y accéder en utilisant :

```
git branch [branch-name]
```

Pousser la nouvelle branche vers un référentiel distant

Après avoir créé une nouvelle branche localement, vous pouvez la transférer vers le référentiel distant en utilisant :

```
git push -u origin [branch-name]
```

Cette commande établit une connexion de suivi entre votre branche locale et la branche distante.

Exemples

Création d'une branche de fonctionnalités

```
git checkout -b feature/login-system
```

Cela crée et bascule vers une branche nommée « feature/login-system ».

Création d'une branche de correctif

```
git checkout -b hotfix/critical-bug
```

Cette commande est utilisée lorsque vous devez corriger rapidement un bug critique.

Extraire une branche distante existante

Tout d'abord, répertoriez toutes les succursales, y compris celles distantes :

```
git branch -a
```

Ensuite, extrayez la branche distante :

```
git checkout -b [branch-name] origin/[branch-name]
```

Questions fréquemment posées

Comment renommer une succursale ?

Pour renommer une branche, utilisez :

```
git branch -m [old-name] [new-name]
```

Si vous souhaitez renommer la branche actuelle, vous pouvez utiliser cette commande :

```
git branch -m [new-name]
```

Comment puis-je supprimer une branche ?

Pour supprimer une branche locale, utilisez :

```
git branch -d [branch-name]
```

Pour forcer la suppression d'une branche (à utiliser avec précaution) :

```
git branch -D [branch-name]
```

Pour supprimer une branche distante :

```
git push origin --delete [branch-name]
```

Comment fusionner les modifications d'une branche à une autre ?

Tout d'abord, passez à la branche dans laquelle vous souhaitez fusionner :

```
git checkout [target-branch]
```

Fusionnez ensuite l'autre branche :

```
git merge [source-branch]
```

Quelle est la différence entre git branch et git checkout -b ?

La commande git branch [nom de la branche] checkout crée une nouvelle branche mais ne vous y fait pas basculer tandis que la commande crée une git -b [nom de la branche] nouvelle branche et vous y fait également basculer immédiatement.

Comment puis-je voir toutes les branches de mon référentiel ?

Pour répertorier toutes les succursales locales, utilisez :

```
git branch
```

Pour voir les branches locales et distantes, utilisez :

```
git branch -a
```

Conclusion

La création et la gestion de branches dans Git permettent aux équipes de travailler sur différentes fonctionnalités, correctifs ou expériences en parallèle sans perturber la base de code principale. Comprendre ces concepts est crucial pour une collaboration efficace et efficiente