Git - Mode avancé

Alexandre GIRARD - Conseiller en Nouvelles Technologies - alexandre.girard@maif.fr

Git - Mode avancé

Les conflits

git stash

git rebase

Alexandre GIRARD - Conseiller en Nouvelles Technologies - alexandre.girard@maif.fr

Les conflits

Les conflits surviennent lorsqu'un fichier est modifié en même temps dans deux branches différentes.

Pour résoudre un conflit, il faut éditer le fichier en question pour choisir la version à conserver. Il existe plusieurs outils pour résoudre les conflits. Par exemple lors d'une pull request sur GitHub, il est possible de résoudre les conflits directement dans l'interface web.

Pour les cas plus complexes il est préférable de corriger les conflits en local, par exemple avec Visual Studio Code.

Les conflits - Simulation d'un conflit

Créer un conflit en modifiant le même fichier dans deux branches différentes.

```
git checkout -b feature-1
echo "feature-1" > file.txt
git add file.txt
git commit -m "Add feature-1"
git push --set-upstream origin feature-1
```

Les conflits - Simulation d'un conflit (suite)

```
git checkout -b feature-2
echo "feature-2" > file.txt
git add file.txt
git commit -m "Add feature-2"
git push --set-upstream origin feature-2
```

Les conflits - Simulation d'un conflit (suite)

```
git checkout feature-1
echo "feature-1" >> file.txt
git commit -am "Update feature-1"
git push
```

Les conflits - Simulation d'un conflit (suite)

Tenter de merger la branche `feature-2` dans la branche `feature-1` via une pull request sur GitHub.
Un conflit survient, github propose de résoudre le conflit directement dans l'interface web.

Il s'agit d'un conflit simple, il est possible de le résoudre directement dans l'interface web.

Les conflits - Résolution d'un conflit - Interface web

```
<<<<<<< feature-2
feature-2
=======
feature-1
feature-1
>>>>>>> feature-1
```

Les lignes entre `<<<<<<<` et `=======` correspondent à la branche `feature-2`, les lignes entre `=======` et `>>>>>>>` correspondent à la branche `feature-1`. Il faut choisir la version à conserver.

Ne corriger pas le conflit pour le moment. Nous allons voir comment résoudre le conflit en local. Github propose les commandes à exécuter pour résoudre le conflit en local.

Les conflits - Résolution d'un conflit - depuis Visual Studio Code

Récupérer la dernire version de la branche `feature-1` et `feature-2` en local et tenter de merger la branche `feature-2` dans la branche `feature-1`. Un conflit survient.

```
git pull origin feature-1
git checkout feature-2
git merge feature-1
```

Les conflits - Résolution d'un conflit - depuis Visual Studio Code

Visual Studio Code propose de résoudre le conflit directement dans l'interface. Pour chaque ligne en conflit, il est possible de choisir la version à conserver (celle de la branche `feature-1` ou `feature-2` ou une combinaison des deux).

Pour des modifications plus complexes, il est possible de modifier le fichier directement. Une fois les conflits résolus, il faut ajouter les fichiers modifiés et commiter les changements.

Pour des modifications complexes le fait de résoudre en local est préférable et permet de tester les modifications avant de les pousser (lancer les tests unitaires par exemple).

git stash

`git stash` permet de sauvegarder les modifications en cours pour les appliquer plus tard. Cela permet de changer de branche sans perdre les modifications en cours (travail non terminé) ou de sauvegarder les modifications pour les appliquer plus tard sans pour autant les commiter.

```
git stash
```

Pour appliquer les modifications sauvegardées:

```
git stash pop
```

git stash - Visual Studio Code

Il est possible de sauvegarder les modifications en cours directement depuis Visual Studio Code et de les visualiser, pour ensuite les appliquer, tout en les gardant en mémoire (il est possible de les appliquer plusieurs fois) ou de les supprimer.

git rebase

`git rebase` permet de réécrire l'historique des commits. Cela permet de fusionner plusieurs commits en un seul, de changer l'ordre des commits, de supprimer des commits, de modifier des commits...

Il est possible de faire un rebase interactif pour modifier l'historique des commits.

git rebase - Création d'un jeu de commits

Créer une branche `feature-3` et ajouter des commits. Certains que vous souhaitez fusionner en un seul commit, d'autres que vous souhaitez supprimer puis enfin un commit dont vous souhaitez modifier le message.

Puis tester le rebase interactif pour modifier l'historique des commits.