## git config credential.helper "" => supprimer le token dans git

## Accès aux référentiels

Lorsque vous accédez à un dépôt et que vous utilisez l'authentification HTTPS, assurez-vous que vous avez l'accès/l'autorisation de vous connecter, puis utilisez simplement l'adresse HTTPS du dépôt Git lui-même.

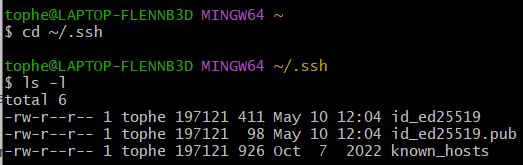
Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

## Accès ssh

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement





Ouvrer le fichier \*.pub, et copier la clé dans le presse papier.

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquement

Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

git init : initialisez votre répertoire courant en local.

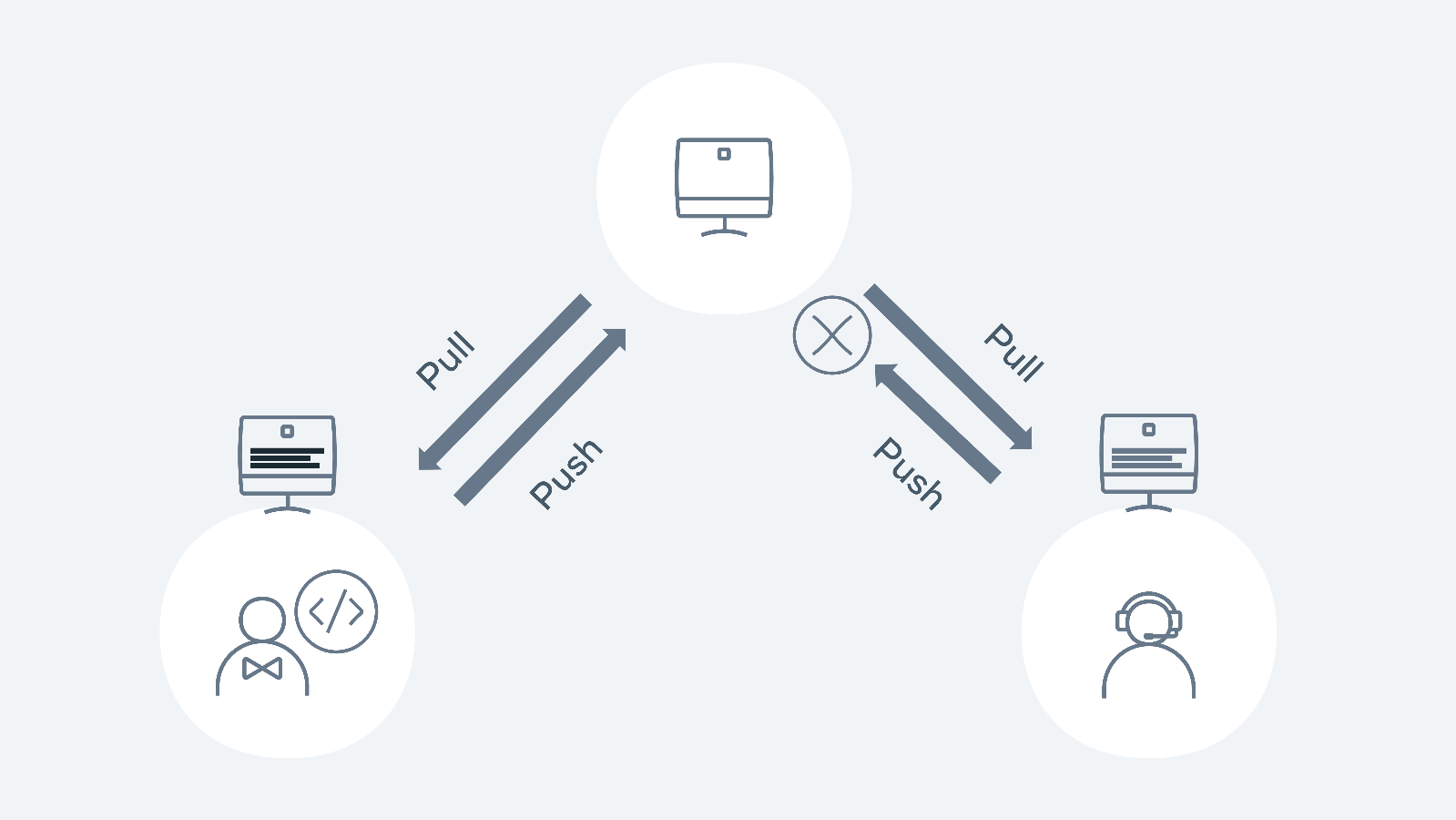
## Résolution des conflits

Les conflits se produisent normalement lorsque vous essayez de fusionner une branche qui peut avoir des modifications concurrentes. Git essaiera normalement de fusionner automatiquement (auto-merge), mais dans le cas d'un conflit, il aura besoin d'une confirmation. Les modifications concurrentes doivent être résolues par l'utilisateur final. Ce processus s'appelle la fusion ou le rebasage.

Le développeur doit examiner les modifications sur le serveur et dans son environnement local et valider celles qui doivent être résolues.

Un exemple de conflit de fusion est celui de deux développeurs qui travaillent sur des branches dépendantes l'une de l'autre. Les deux développeurs travaillent sur le même fichier appelé Feature.js. Chacune de leurs tâches consiste à ajouter une nouvelle fonctionnalité à une méthode existante. Le développeur 1 a une branche appelée feature1, et le développeur 2 a une branche appelée feature2.

Le développeur 1 envoie le code avec les modifications dans le référentiel distant. Le développeur 2 apporte ses modifications.

Demonstration of the push pull method

Voyons comment cela se passe dans Git. Les développeurs 1 et 2 consultent le dépôt principal le lundi matin. Ils ont tous deux la même copie. Les deux développeurs consultent une nouvelle branche - fonctionnalité 1 et 2.

git pull

git checkout -b feature1

git pull

git checkout -b feature2

Le développeur 1 apporte ses modifications à un fichier appelé Feature.js et les soumet à l'approbation du dépôt via une PR (pull request)

git add Feature.js

git commit -m 'chore: added feature 1!!'

git pull origin main

git push -u origin feature1

Le PR est revu et ensuite fusionné dans la branche principale. Pendant ce temps, le développeur 2 commence à coder sur sa fonctionnalité. Là encore, il suit le même processus que le développeur 1 :

git add Feature.js

git commit -m 'chore: added feature 2!!!'

git pull origin main

From github.com:demo/demo-repo

\* branch main -> FETCH\_HEAD

9012934..d3b3cc0 main -> origin/main

Auto-merging Feature.js

CONFLICT (content): Merge conflict in Feature.js

Automatic merge failed; fix conflicts and then commit the result.

Git nous indique qu'un conflit de fusion s'est produit et qu'il doit être corrigé avant d'être poussé vers le repo distant. L'exécution de git status nous donnera également le même niveau de détail :

git status

On branch feature2

You have unmerged paths.

(fix conflicts and run "git commit")

(use "git merge --abort" to abort the merge)

Unmerged paths:

(use "git add <file>..." to mark resolution)

both modified: Feature.js

no changes added to commit (use "git add" and/or "git commit -a")

Pour fusionner, le développeur 2 doit voir et comparer les modifications du développeur 1. C'est une bonne pratique de voir d'abord quelle branche est à l'origine du problème de fusion. Le développeur 2 exécute la commande suivante :

git log --merge

commit 79bca730b68e5045b38b96bec35ad374f44fe4e3 (HEAD -> feature2)

Author: Developer 2

<developer2@demo.com>

Date: Sat Jan 29 16:55:40 2022 +0000

chore: add feature 2

commit 678b0648107b7c53e90682f2eb8103c59f3cb0c0

Author: Developer 1

<developer1@demo.com>

Date: Sat Jan 29 16:53:40 2022 +0000

chore: add feature 1

Nous pouvons voir dans le code ci-dessus que les modifications conflictuelles de l'équipe se sont produites dans les branches des fonctionnalités 1 et 2. Le développeur 2 veut maintenant voir la modification qui est à l'origine du conflit.

git diff

diff --cc Feature.js

index 1b1136f,c3be92f..0000000

--- a/Feature.js

+++ b/Feature.js

@@@ -1,4 -1,4 +1,8 @@@

let add = (a, b) => {

++<<<<<<< HEAD

+ if(a + b > 10) { return 'way too much'}

++=======

+ if(a + b > 10){ return 'too much' }

++>>>>>>> d3b3cc0d9b6b084eef3e0afe111adf9fe612898e

return a + b;

}

La seule différence réside dans la formulation de l'instruction de retour. Le développeur 1 a ajouté "trop", mais le développeur 2 a ajouté "beaucoup trop". Tout le reste est identique en termes de fusion et il s'agit d'une correction assez facile. Git affiche des flèches <<< >>> pour indiquer les modifications. Le développeur 2 supprime les marqueurs afin que le code soit prêt à être soumis :

let add = (a, b) => {

if(a + b > 10) { return 'way too much'}

return a + b;

}

git add Feature.js

git commit -m 'fix merge conflicts'

git push -u origin feature2

Le développeur 2 a maintenant corrigé un conflit de fusion et peut créer son PR pour que le code soit fusionné dans la ligne principale.