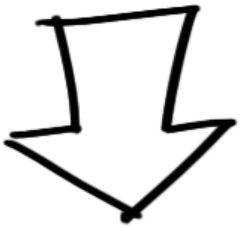
git - petit guide

juste un petit guide pour bien démarrer avec git. no deep shit ;)

par Roger Dudler (translation by KokaKiwi)

Remerciements à @tfnico, @fhd, Namics

this guide in english, deutsch, español, italiano, nederlands, português, русский, türkçe, မြန်မာ, 日本語, 中文, 한국어



installation

Télécharger git pour Mac OSX

Télécharger git pour Windows

Télécharger git pour Linux

créer un nouveau dépôt

créez un nouveau dossier, ouvrez le et exécutez la commande

```
git init
```

pour créer un nouveau dépôt.

cloner un dépôt

créez une copie de votre dépôt local en exécutant la commande

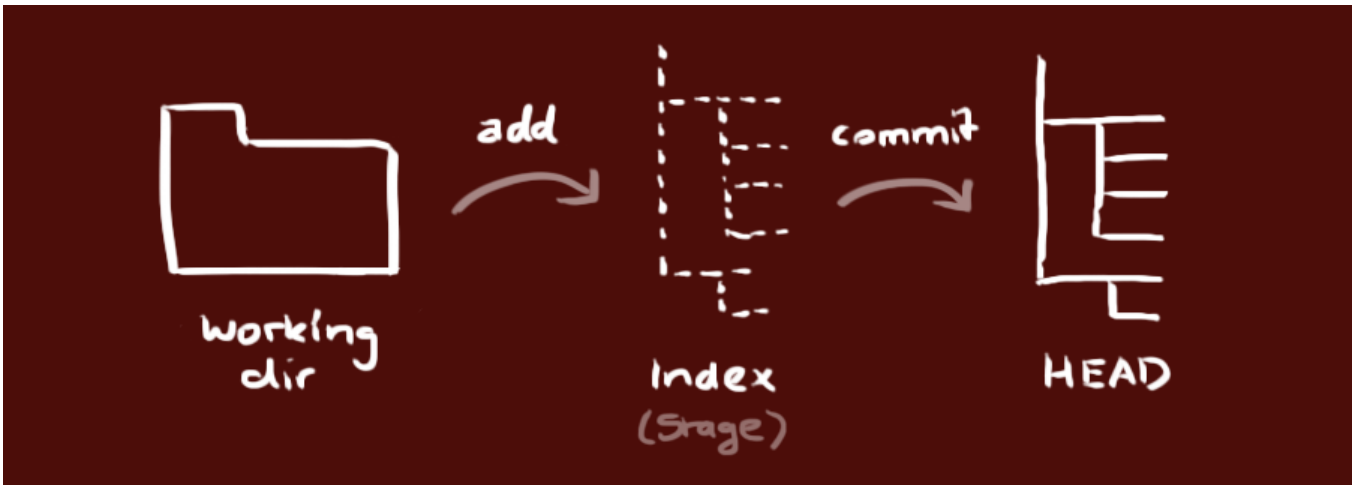
```
git clone /path/to/repository
```

si vous utilisez un serveur distant, cette commande sera

```
git clone username@host:/path/to/repository
```

arbres

votre dépôt local est composé de trois "arbres" gérés par git. le premier est votre `espace de travail` qui contient réellement vos fichiers. le second est un `Index` qui joue un rôle d'espace de transit pour vos fichiers et enfin `HEAD` qui pointe vers la dernière validation que vous ayez fait.



ajouter & valider

Vous pouvez proposer un changement (l'ajouter à l'**Index**) en exécutant les commandes

```
git add <filename>
git add *
```

C'est la première étape dans un workflow git basique. Pour valider ces changements, utilisez

```
git commit -m "Message de validation"
```

Le fichier est donc ajouté au **HEAD**, mais pas encore dans votre dépôt distant.

envoyer des changements

Vos changements sont maintenant dans le **HEAD** de la copie de votre dépôt local. Pour les envoyer à votre dépôt distant, exécutez la commande

```
git push origin master
```

Remplacez *master* par la branche dans laquelle vous souhaitez envoyer vos changements.

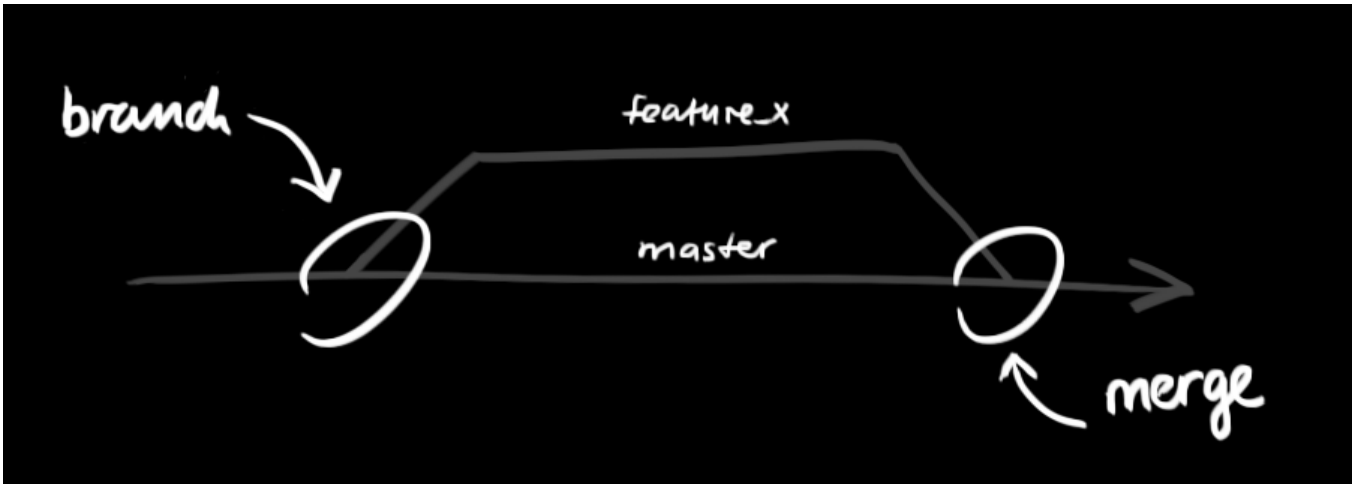
Si vous n'avez pas cloné votre dépôt existant et voulez le connecter à votre dépôt sur un serveur distant, vous devez l'ajouter avec

```
git remote add origin <server>
```

Maintenant, vous pouvez envoyer vos changements vers le serveur distant sélectionné

branches

Les branches sont utilisées pour développer des fonctionnalités isolées des autres. La branche *master* est la branche par défaut quand vous créez un dépôt. Utilisez les autres branches pour le développement et fusionnez ensuite à la branche principale quand vous avez fini.



créer une nouvelle branche nommée "feature_x" et passer dessus pour l'utiliser

```
git checkout -b feature_x
```

retourner sur la branche principale

```
git checkout master
```

et supprimer la branche

```
git branch -d feature_x
```

une branche n'est *pas disponible pour les autres* tant que vous ne l'aurez pas envoyée vers votre dépôt distant

```
git push origin <branch>
```

mettre à jour & fusionner

pour mettre à jour votre dépôt local vers les dernières validations, exécutez la commande

```
git pull
```

dans votre espace de travail pour *récupérer* et *fusionner* les changements distants.

pour fusionner une autre branche avec la branche active (par exemple master), utilisez

```
git merge <branch>
```

dans les deux cas, git tente d'auto-fusionner les changements.

Malheureusement, ça n'est pas toujours possible et résulte par des *conflits*.

Vous devez alors régler ces *conflits* manuellement en éditant les fichiers indiqués par git. Après l'avoir fait, vous devez les marquer comme fusionnés avec

```
git add <filename>
```

après avoir fusionné les changements, vous pouvez en avoir un aperçu en utilisant

```
git diff <source_branch> <target_branch>
```

tags

il est recommandé de créer des tags pour les releases de programmes. c'est un concept connu, qui existe aussi dans SVN. Vous pouvez créer un tag nommé *1.0.0* en exécutant la commande

```
git tag 1.0.0 1b2e1d63ff
```

le *1b2e1d63ff* désigne les 10 premiers caractères de l'identifiant du changement que vous voulez référencer avec ce tag. Vous pouvez obtenir cet identifiant avec

```
git log
```

vous pouvez utiliser moins de caractères de cet identifiant, il doit juste rester unique.

remplacer les changements locaux

Dans le cas où vous auriez fait quelque chose de travers (ce qui bien entendu

n'arrive jamais ;) vous pouvez annuler les changements locaux en utilisant cette commande

```
git checkout -- <filename>
```

cela remplacera les changements dans votre arbre de travail avec le dernier contenu du HEAD. Les changements déjà ajoutés à l'index, aussi bien les nouveaux fichiers, seront gardés.

Si à la place vous voulez supprimer tous les changements et validations locaux, récupérez le dernier historique depuis le serveur et pointez la branche principale locale dessus comme ceci

```
git fetch origin
git reset --hard origin/master
```

conseils utiles

Interface git incluse

```
gitk
```

utiliser des couleurs dans la sortie de git

```
git config color.ui true
```

afficher le journal sur une seule ligne pour chaque validation

```
git config format.pretty oneline
```

utiliser l'ajout interactif

```
git add -i
```

liens et ressources

clients graphiques

GitX (L) (OSX, open source)
Tower (OSX)
Source Tree (OSX, free)
GitHub for Mac (OSX, free)
GitBox (OSX)
Git Extensions (WIN, open source)

guides

Git Community Book
Pro Git
Think like a git
GitHub Help
A Visual Git Guide

commentaires

15 Comments git - the simple guide Login

Sort by Newest Share Favorite

Join the discussion…

lolman • 2 months ago
lol
• Reply • Share ›

Ozgeek • 3 months ago
Merci !
• Reply • Share ›

Jean • 5 months ago
Très agréable à lire et utiliser. Du doodling ?
Juste pour le message 'git commit', il n'y a pas d'espace entre le -m et le double quote.

Jean.
• Reply • Share ›

ndna ➚ Jean • 3 months ago
Oui, il y en a un. Toutes les lettres derrière '-' sont utilisées pour spécifier le comportement du programme. Ce qui suit "-m" est en rapport avec l'argument 'm', c'est à dire le message de commit.
On est supposés utiliser des espaces ou des tabulations pour séparer les différents arguments d'une ligne de commande.
• Reply • Share ›

elmahdi • 6 months ago
merci
• Reply • Share ›

Renrhaf • 6 months ago
merci !
• Reply • Share ›

Chriscrat • 6 months ago
Simple, synthétique, clair, ce qu'il faut pour pouvoir bosser avec notre amis Git. Bravo
• Reply • Share ›

mick • 8 months ago
bravo, beau site et super boulot
• Reply • Share ›

Spywen • 9 months ago
cool ! Merci
• Reply • Share ›

Arfaoui • a year ago
TOP !! un jeu d'enfant :D !!!! Merci
• Reply • Share ›

mik3fly.4ster5ik • a year ago
Parfait. Juste parfait.

• Reply • Share ›

NewRuby • a year ago

Extra, franchement, c'est simple...

1 • Reply • Share ›

Agro • 2 years ago

Simple, clair et concis. Parfait.

2 • Reply • Share ›

SGH • 2 years ago

Très utile, juste ce qu'il faut pour GIt et pas besoin de lire un bouquin encore Merci

3 • Reply • Share ›

ibasaw • 2 years ago

merci :) clair et simple

• Reply • Share ›