Documentation Git, GitHub, GitLab et autres outils basés sur Git

Retour à toute les documentations

Règles

- "`Saisie utilisateur`"
- '`Elément cliquable/sélectionnable`'
- `Nom de fichier, dossier ou autre`
- <Élément à remplacer>

> lien, raccourci clavier et phrase de demande de saisie

```
commande, extrait code et extrait de fichier
```

Table des matières

Installation

Installation de Git sur Ubuntu Desktop et Debian

- Installer le paquet `git` depuis le dépot `apt` :

```
sudo apt install git
```

- Pour vérifier l'installation ouvrez un terminal et lancer la commande :

```
git --version
```

  - Résultat attendu :

```
git version 2.34.1
```

Installation de Git sur Windows 10 et 11

- Télécharger le fichier d'installation sur le site officiel :

> https://git-scm.com/download/win

- Ouvrir le fichier d'installation
- Suivre les instructions d'installation
- Séléctionner l'option '`Add a Git Bash Profile to Windows Terminal`'
- Ne pas séléctionner l'option '`Use Git from Git Bash only`' pour pouvoir utiliser git depuis le terminal windows
- Changer l'éditeur de texte par défaut si vous en avez envie
- Laisser toutes les autres options par défaut
- Cliquer sur '`Install`'
- Pour vérifier l'installation ouvrez un terminal et lancer la commande :

```
git --version
```

  - Résultat attendu :

```
git version 2.41.0.windows.1
```

Configuration

Configuration pour Linux et Windows

- Si vous avez besoins d'aide pour la configuration de git, lancer la commande :

```
git help config
```

- Configurer l'adresse mail :

```
git config --global user.email <adresse mail>
```

- Configurer l'adresse mail :

```
git config --global user.name <votre prenom>
```

Liaison avec Github sur Linux et Windows

- Générer une clé SSH :

```
ssh-keygen -t rsa -b 4096 -C "votre adresse mail"
```

- laisser vide les trois champs suivant :

> Enter file in which to save the key (/home/$USER/.ssh/id_rsa):

> Enter passphrase (empty for no passphrase):

> Enter same passphrase again:

- Ajouter la clé public (présente dans le fichier `~/.ssh/id_rsa.pub`) à Github dans '`Settings`' > '`SSH and GPG keys`' > '`New SSH key`'

Utilisation

Utilisation de plusieurs comptes Github sur le même ordinateur - Ubuntu

- Source

> https://gist.github.com/bonnopc/c78920431284ce3fc2a5270016205116

La documentation ci-dessous utilise deux comptes Github, avec deux clés SSH différentes, mais vous pouvez l'adapter pour autant de comptes que vous voulez.

Configuration pour Git

- Placez-vous dans le dossier de votre projet

```
cd /chemin/vers/votre/projet
```

- Modifier votre adresse mail pour un projet spécifique

```
git config user.email "Votre adresse mail"
```

- Modifier votre nom d'utilisateur pour un projet spécifique

```
git config user.name "Votre nom"
```

Configuration pour SSH et Github

- Générer une nouvelle clé SSH comme indiqué dans la section Liaison avec Github sur Linux et Windows
- Effacer les clés mises en cache précédentes

```
ssh-add -D
```

  - Si vous obtenez l'erreur ci-dessous :

```
Error connecting to agent: Connection refused
```

  - Activer l'agent SSH

```
eval "$(ssh-agent)"
```

  - Réessayer d'effacer les clés mises en cache

```
ssh-add -D
```

- Ajouter vos clés SSH à l'agent SSH

```
ssh-add ~/.ssh/id_rsa
ssh-add ~/.ssh/id_rsa_2
...
```

- Si vous le souhaitez, vous pouvez vérifier les clés ajoutées à l'agent SSH avec la commande

```
ssh-add -l
```

- Si vous n'en avez pas déjà un, créer un fichier de configuration SSH

```
touch ~/.ssh/config
```

- Ouvrir le fichier de configuration

```
open ~/.ssh/config
```

- Modifier le fichier de configuration pour ajouter les configurations suivantes

```
# Compte github 1
Host <NOM>.github.com
HostName github.com
IdentitiesOnly yes
IdentityFile <~/.ssh/id_rsa_1>

# Compte github 2
Host <NOM>.github.com
HostName github.com
IdentitiesOnly yes
IdentityFile <~/.ssh/id_rsa_rsa_2>
```

  - Remplacer `<NOM>` par le nom de votre compte Github
  - Remplacer `~/.ssh/id_rsa_1` et `~/.ssh/id_rsa_2` par le chemin complet de vos clés SSH

- Cloner un dépôt Github

```
git clone git@<NOM>.github.com:exemple/exemple-repo.git
```

  - Remplacer `<NOM>` par le nom du compte Github que vous voulez utiliser

- Modifier un dépôt Github existant

    - Aller dans le dossier du dépôt
    - Modifier le fichier `.git/config`

```
open .git/config
```

    - Modifier l'URL du dépôt

```
[remote "origin"]
  url = git@<NOM>.github.com:exemple/exemple-repo.git
```

- Vérifier que tout fonctionne correctement en poussant un changement

```
git add .
git commit -m "Message de commit"
git push
```

Remplacer la branch master par une autre branch

- Source

> https://stackoverflow.com/questions/2862590/how-to-replace-master-branch-in-git-entirely-from-another-branch

- Vérifier que la branch master est bien à jour

```
git checkout master
git pull
```

- Passer sur la branch que vous voulez mettre à la place de la branch master que nous appelons `<new_branch>` et vérifier qu'elle est bien à jour

```
git checkout <new_branch>
git pull
```

- Fusionner la branch `<new_branch>` avec la branch master en écrasant la branch master

```
git merge -s ours master
```

- Passer sur la branch master

```
git checkout master
```

- Fusionner la branch `<new_branch>` avec la branch master

```
git merge <new_branch>
```

- Mettre à jour la branch master sur Github

```
git push origin master
```

Mise en attente de changements

- Source

> https://git-scm.com/book/fr/v2/Utilitaires-Git-Remisage-et-nettoyage

- Ajouter les changements à la pile des modifications non finies

```
git stash
```

- Faire des changements que vous voulez
  - Vous pouvez faire autant de changements que vous voulez
  - Vous pouvez changer de branch
  - Vous pouvez faire des commits
  - Vous pouvez faire des pulls
  - Vous pouvez faire des pushs
  - etc...
- Récupérer la liste des changements mis en attente

```
git stash list
```

- Merge les changements mis en attente

```
git stash apply
```

- Supprimer les changements mis en attente

```
```

```
git stash clear
```

Retourner à un commit précédent

Attention, cette commande est dangereuse, elle va supprimer tous l'historique des commits après le commit que vous avez choisi.

- Revenir à un commit précédent

```
git reset --hard <commit>
```

  - Remplacer `<commit>` par le SHA complet du commit auquel vous voulez revenir
    - Vous pouvez trouver le SHA complet d'un commit avec la commande `git log`
      - Voici à quoi ressemble un SHA : `6ae4b917362ce6ac196f651e83b8afb14452fae0`
  - Si vous voulez supprimer un commit fait par erreur, vous devez prendre le SHA du commit précédent l'erreur
  - Si vous voulez supprimer le dernier commit, vous pouvez utiliser la commande suivante :

```
git reset --hard HEAD~1
```

    - Vous pouvez remplacer 1 par le nombre de commit que vous voulez supprimer

- Faire un push pour mettre à jour le dépôt

```
git push --force
```

Tirer (Pull) la nouvelle version d'un repertoire forked

- Source

> https://stackoverflow.com/questions/3903817/pull-new-updates-from-original-github-repository-into-forked-github-repository

- Créer une nouvelle télécommande (Remote) nommer `upstream` qui pointe vers l'url du dépos original

```
git remote add upstream <url/to/depos>
```

- Comparer les modifications ?

```
git fetch upstream
```

- Récupérer les modification en local

```
git merge upstream/<orginal-branch> <local-branch>
```

  - `<orginal-branch>` : Branch du dépos original à récupérer (souvent `master`)
  - `<local-branch>` : Branch local sur laquelle vous voulez récupérer les modifications (souvent identique à `<orginal-branch>`)

- Pousser (Push) les modifications vers votre dépos forked

```
git push
```

Création d'un workflow Github pour Laravel

- Source

> https://docs.github.com/fr/enterprise-cloud@latest/actions/using-workflows/creating-starter-workflows-for-your-organization

Erreur

Erreur d'authentification HTTPS

- Ne pas passer par HTTPS, il faut utiliser le ssh pour communiquer avec Github. Si votre clé ssh est bien configurer comme indiquer au dessus vous n'aurez normalement pas de problème.
- Sinon vous pouvez aussi mêtre un token à la place du mot de passe mais il faudra le remettre à chaque action que vous effectuer sur Github
  - Générer un token d'authentification sur Github dans '`Settings`' > '`Developer settings`' > '`Personal access tokens`' > '`Fine-grained personal access tokens`' > '`Generate new token`'
  - Sélectionner quel repo vous voulez accéder avec votre token, dans mon cas j'ai sélectionné '`All repositories`'
  - Donner les autororisations que vous voulez, dans mon cas j'ai sélectionné la plus haute autorisation de chaque élément de la catégorie `Repository permissions` et j'ai rien toucher à l'autre catégorie.
  - Cliquer sur '`Generate token`'

Erreur le nom gitbug.com ne peut pas être résolu

- Si vous obtenez l'erreur ci-dessous :

```
ssh: Could not resolve hostname github.com: Temporary failure in name
resolution
fatal: Impossible de lire le dépôt distant.
```

- Vérifier que vous avez bien une connexion internet
- Si votre connexion internet fonctionne, vérifier que vous pouvez accéder à Github

```
ping github.com
```

- Si vous obtenez l'erreur ci-dessous :

```
ping: github.com: Temporary failure in name resolution
```

- Ça veut dire que vous avez un problème de DNS, ouvrez le fichier `/etc/resolv.conf`

```
sudo nano /etc/resolv.conf
```

- Ajouter donc les lignes suivantes dans le fichier `/etc/resolv.conf`

```
...
nameserver 8.8.8.8
nameserver 8.8.4.4
```

- Sauvegarder le fichier
- Relancer la commande ping pour vérifier que vous avez bien accès à Github

```
ping github.com
```

Licence

Retour à toute les documentations