

Installation et configuration

Romain THERRAT

POCKOST

14 décembre 2020

Sommaire

- 1 CLI et GUI
- 2 Installation
- 3 Configuration
- 4 Labs

- Créé à l'origine pour Linux
- Donc : CLI <3
- Commande : git
- CLI vs GUI

CLI Command Line Interpreter

GUI Graphical User Interface

Nous présenterons les exemples en CLI mais des alternatives graphiques existent

- La CLI couvre 100% des possibilités
- Les interfaces graphiques peuvent :
 - Masquer des fonctionnalités
 - Complexifier la gestion (afficher trop de possibilités)
- Nous présenterons tous de même quelques solutions graphiques :).

Utilisation de la CLI

Les commandes git sont toujours composées d'une commande et peuvent avoir un ensemble d'arguments

```
romain@laptop:~$ git command arg1 arg2 ...
```

Par exemple si vous souhaitez créer un dépôt vide dans un dossier

```
romain@laptop:~$ git init ./myproject  
Initialized empty Git repository in  
/home/romain/myproject/.git/
```

Sommaire

- 1 CLI et GUI
- 2 Installation**
- 3 Configuration
- 4 Labs

Installation sous Linux

- Sous Linux installation simple

```
Debian apt-get install git  
RH,CentOS yum install git  
Gentoo emerge install git  
...
```

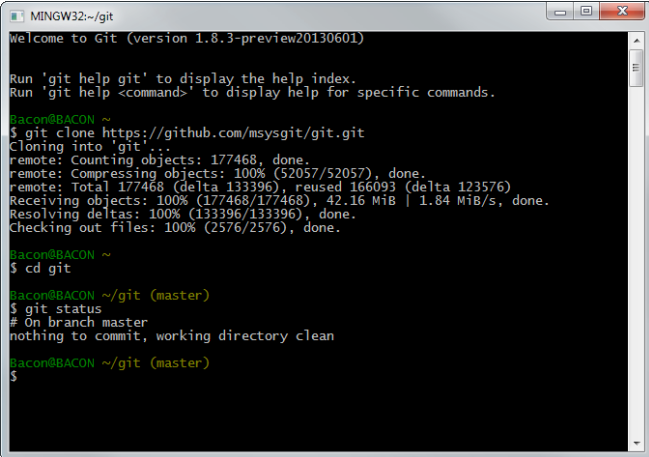
- La commande git disponible dans le shell

```
romain@laptop:~$ git --version  
git version 2.20.1
```

- Sous Windows plusieurs solutions
 - msysgit Obsolète
 - git for windows
 - Intégration à des IDEs
 - TortoiseGIT

- Git For Windows
 - Basé sur le code source officiel de git
 - Installation simple
 - <https://github.com/git-for-windows/git/releases/latest>
 - Version 32 ou 64bit
- Contient aussi
 - git Bash : Émulation de bash pour Windows
 - git GUI : Deux interfaces graphiques pour la visualisation et l'exécution d'actions
 - Intégration dans le menu contextuel (clique droit)

Git For Windows



```
MINGW32:~/git
welcome to Git (version 1.8.3-preview20130601)

Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

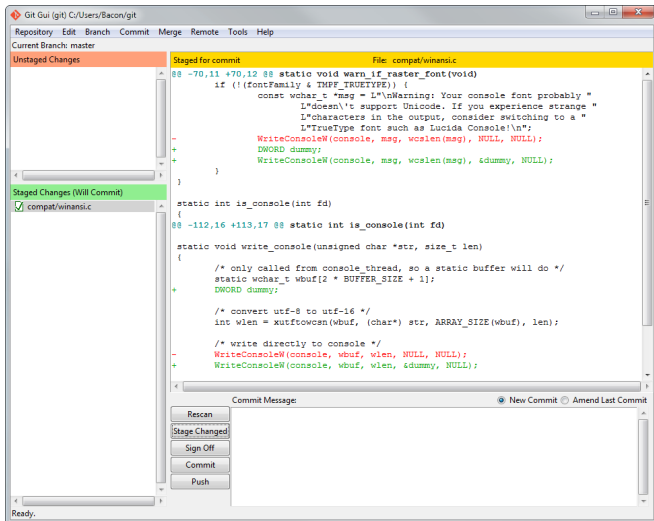
Bacon@BACON ~
$ git clone https://github.com/msysgit/git.git
Cloning into 'git'...
remote: Counting objects: 177468, done.
remote: Compressing objects: 100% (52057/52057), done.
remote: Total 177468 (delta 133396), reused 166093 (delta 123576)
Receiving objects: 100% (177468/177468), 42.16 MiB | 1.84 MiB/s, done.
Resolving deltas: 100% (133396/133396), done.
Checking out files: 100% (2576/2576), done.

Bacon@BACON ~
$ cd git

Bacon@BACON ~/git (master)
$ git status
# On branch master
nothing to commit, working directory clean

Bacon@BACON ~/git (master)
$
```

Git For Windows



- Pour SVN on avait TortoiseSVN
- Équivalent
- Peut être utile si on sait ce qu'on fait
- Intégration au menu contextuel

Sommaire

- 1 CLI et GUI
- 2 Installation
- 3 Configuration**
- 4 Labs

Nous avons trois types de configuration

- Pour tout le monde (`/etc/gitconfig`)
- Par utilisateur (`/home/user/.gitconfig`)
- Par dépôt (`.git/config`)

Configuration

- Toutes les configurations sont réalisées via la commande `git config`
- Arguments
 - `-system` Configuration pour tous les utilisateurs
 - `-global` Configuration pour tous les projets de l'utilisateur courant
 - `rien` Configuration pour le projet dans lequel nous sommes actuellement
- Exemples

```
$ git config --global user.name "Romain THERRAT"  
$ git config --global user.email "romain@pockost.com"
```

Voici quelques exemples de configuration

`user.name` Nom de l'utilisateur (Affiché dans l'historique)

`user.email` Email de l'utilisateur (Affiché dans l'historique)

`commit.template` Template de base pour les messages d'historique

`core.excludesfile` Liste de fichier à ne jamais versionner (MacOS ?)

`core.autocrlf` Convertir les CRLF en LF

L'ensemble des options peuvent être visualisées via `man git-config`

- Une exclusion par projet
- Versionné avec celui-ci
- Le fichier `.gitignore`
- Une ligne par exclusion (avec ou sans *)
- À la base du projet ou par dossier
- Liste de fichier `gitignore` par type de projet
<https://github.com/github/gitignore>

En plus des configurations standards il est possible d'appliquer des scripts custom : Les hooks (crochets en français ...)

- Côté client
- Côté serveur
- dossier `.git/hooks`
 - Des exemples : `type.sample`
 - Doivent être exécutables
- Exemples
 - Valider la forme d'un commentaire de commit
 - Empêcher certaines actions
 - Valider le contenu des fichiers (plus de TODO, ...)

Sommaire

- 1 CLI et GUI
- 2 Installation
- 3 Configuration
- 4 Labs**

Nous allons procéder à notre premier Labs !

Dans celui-ci nous allons bien évidemment commencer par procéder à l'installation de git sur votre poste. Nous configurerons ensuite notre compte utilisateur et réaliserons quelques commandes git de base.

- Si vous êtes sous Windows installer Git for Windows (<https://github.com/git-for-windows/git/releases/latest>)
- Si vous êtes sous Linux installer git via votre gestionnaire de paquet.

Une fois l'installation réussis, valider le bon fonctionnement de celle-ci via la commande `git --version`. Cette commande est à exécuter dans le shell et doit vous retourner la version de git installée.

Nous allons maintenant configurer votre compte utilisateur. Il conviendra donc d'utiliser la commande `git config`. Nous souhaitons modifier :

- Le nom de l'utilisateur
- Son adresse mail

De la même manière nous allons réaliser des configurations spéciales pour un dépôt.

- Initialiser un projet vide via la commande `git init <path/to/project>`
- Déplacer vous dans ce dossier
- Exécuter la commande `git config -l` pour afficher la configuration actuelle
- Modifier la valeur de `core.autocrlf` à `true` pour votre projet
- Constater le changement via la commande évoquée précédemment

Pour finir nous allons créer un fichier `.gitignore` dans notre projet. Pour constater son fonctionnement nous utiliserons la commande `git status` (ou un affichage graphique). Cette commande sera décrite dans le prochain chapitre.

- Créer un fichier `identifiants.txt` à la racine du projet
- Afficher le résultat de la commande `git status`. Noter l'état du fichier `identifiants.txt`
- Ajouter un fichier `.gitignore` à la racine de votre projet
- Dans ce fichier ajouter la ligne `identifiants.txt`
- Afficher le résultat de la commande `git status`. Constater que le fichier `identifiants.txt` n'apparaît plus
- Rendez-vous sur github.com/github/gitignore et consulter des exemples de fichiers `.gitignore`