

Les fondamentaux de Git

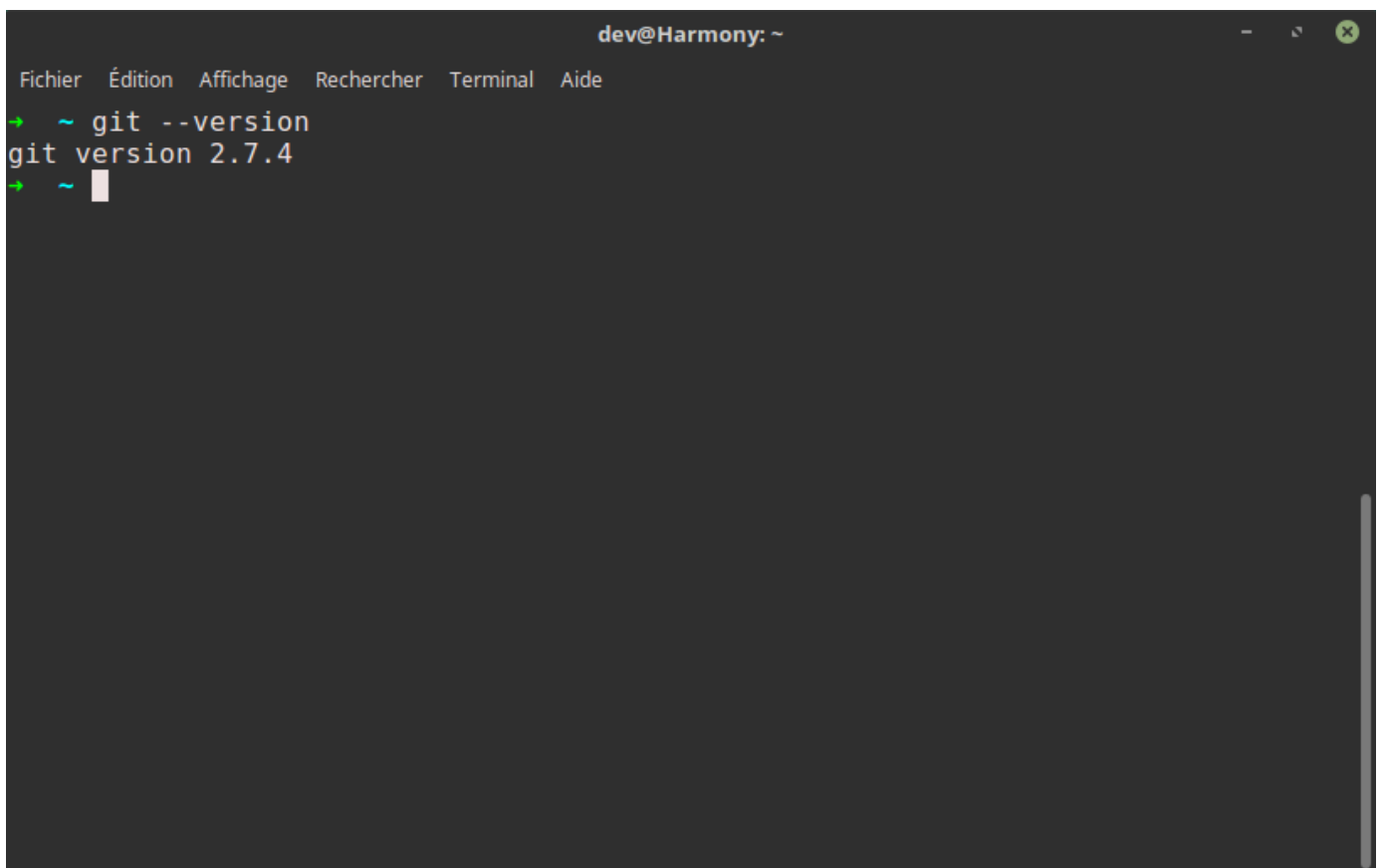
Installer Git

Sur toutes les distributions GNU/Linux basées sur Debian (comme Ubuntu), lancer la commande suivante dans un terminal:

```
sudo apt install git
```

Afin de vérifier que Git est bien installé, lancer la commande suivante:

```
git --version
```

A screenshot of a terminal window titled 'dev@Harmony: ~'. The window has a menu bar with 'Fichier', 'Édition', 'Affichage', 'Rechercher', 'Terminal', and 'Aide'. The terminal shows a prompt '~' followed by the command 'git --version'. The output is 'git version 2.7.4'. The prompt '~' is followed by a cursor. The terminal has a dark background and a light-colored text.

L'affichage de la console dépend beaucoup du système d'exploitation et de la configuration utilisateur. J'utilise zsh, le package oh-my-zsh et un plugin spécifique pour Git. Vous trouverez énormément d'informations sur Google pour customiser votre terminal.

Pour les personnes sur un autre type de distribution Linux, sur Mac ou Windows, se référer à la documentation officielle: <https://git-scm.com/book/fr/v1/D%C3%A9marrage-rapide-Installation-de-Git>.

Configurer Git

Git possède plusieurs niveaux de paramétrage :

- Système (Tous les utilisateurs du système)
- Global (L'utilisateur)
- Projet
-

La configuration de Git se fait soit via le terminal, soit en editant directement des fichiers textes. Git a besoin de 2 paramètres pour fonctionner:

- un nom d'utilisateur
- un email

Ces 2 données seront utilisées et servent à identifier chaque commit d'un historique de projet.

Pour les renseigner, exécuter les 2 commandes suivantes:

```
git config --global user.name "Mickaël Andrieu"
git config --global user.email mickael.andrieu@solvolabs.com
```

Evidemment, remplacer les informations d'exemple par vos propres informations.

Il est possible d'afficher la config complète que Git utilise en lançant la commande:

```
git config --list
```

En savoir plus

La documentation de toutes les options de configuration sont disponibles dans la documentation officielle: <https://git-scm.com/book/fr/v1/D%C3%A9marrage-rapide-Param%C3%A9trage-%C3%A0-la-premi%C3%A8re-utilisation-de-Git>.

Premières commandes Git

Maintenant que Git est installé, voici la liste des commandes qui vont être utiles dans ce TP:

- **init** : Cette commande initialise le répertoire où elle est exécutée comme un projet Git. Concrètement, cela va créer un répertoire “.git” contenant tout le nécessaire au bon fonctionnement de Git pour suivre toutes les modifications à l’intérieur de ce répertoire;
- **add** : Ajoute des fichiers à l’index de Git (stage ou cache sont des synonymes). À partir du moment où un fichier est ajouté une fois dans l’index, il sera “tracké” par Git;
- **commit** : Enregistre l’index vers un nouveau commit avec un message décrivant les modifications effectuées (log) obligatoire;
- **status** : Affiche les fichiers ayant des différences entre index et HEAD, ceux ayant des différences entre le répertoire de travail et l’index ainsi que les fichiers non “trackés”;
- **branch** : Lister, créer ou supprimer une branche;
- **checkout** : Permet de se déplacer d’une branche à l’autre (et bien d’autres choses);

Expérimenter quelques commandes dans le terminal

- Ouvrir le terminal, qui démarre sur le dossier utilisateur (~ sur Ubuntu par exemple).
- Créer un dossier *git-basics* et se déplacer dedans:

```
mkdir git-basics && cd git-basics
```

- Initialiser le dossier *git-basics* en tant que dépôt git:

```
git init
```

*Les utilisateurs d’Ubuntu et de Mac OS X pourront vérifier que le dossier **.git** a bien été créé en exécutant la command **ls -la** .*

- Créer ensuite un nouveau fichier "README.md" dans ce dossier qui contiendra le texte suivant: “# Hello world from Git!”.

Dans le monde Open Source et sur GitHub, ajouter un fichier README est considéré comme une bonne pratique pour documenter un projet.

- A nouveau dans le terminal et dans le dossier **git-basics** , exécuter la commande **git status** .

```
dev@Harmony: ~/git-basics
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
Sur la branche master
Validation initiale
Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)

    README.md

aucune modification ajoutée à la validation mais des fichiers non suivis sont pr
ésents (utilisez "git add" pour les suivre)
→ git-basics git:(master) x █
```

Comme on peut le voir, Git a détecté un nouveau fichier qu'il ne "suit" pas pour l'instant. En effet, le fichier "README.md" ne fait pas encore partie de l'historique.

- Exécuter la commande `git add README.md` puis ré-exécuter la commande `git status`.
- Que s'est-il passé?
- Exécuter maintenant la commande `git commit -m "Ajout du fichier README.md"`:

```
dev@Harmony: ~/git-basics
Fichier  Édition  Affichage  Rechercher  Terminal  Aide

Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)

  README.md

aucune modification ajoutée à la validation mais des fichiers non suivis sont pr
ésents (utilisez "git add" pour les suivre)
→ git-basics git:(master) ✕ git add README.md
→ git-basics git:(master) ✕ git status
Sur la branche master

Validation initiale

Modifications qui seront validées :
  (utilisez "git rm --cached <fichier>..." pour désindexer)

  nouveau fichier : README.md

→ git-basics git:(master) ✕ git commit -m "Ajout du fichier README.md"
[master (commit racine) 22680a9] Ajout du fichier README.md
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
→ git-basics git:(master) █
```

Félicitations! Ceci est le premier commit créé à l'aide du terminal. Pour supprimer le fichier de l'historique git, exécuter la commande `git rm --cached README.md` : le fichier sera toujours présent dans le dossier mais plus dans l'historique de git.

Comprendre les branches Git

En développement logiciel, la bonne pratique consiste à créer une nouvelle branche Git pour chaque nouvelle fonctionnalité à ajouter un logiciel.

Mais qu'est-ce qu'une branche Git? C'est une "version" du projet, une dérivation par rapport au projet d'origine qui est identifié par la branche *master*.

Pour connaître la branche du projet actuellement active, on utilise la commande `git branch` et celle précédée par une astérisque est la branche active.

- Exécuter la commande `git branch`.
- Créer une nouvelle branche "ajout-texte":

```
git branch ajout-texte
git checkout ajout-texte
git branch
```

Il existe un raccourci pour créer une branche git et se déplacer directement dessus:

```
git checkout -b ajout-texte
```

- Effectuer une modification sur le fichier "README.md", ajouter le fichier et le commiter.
- Se déplacer sur la branche *master*: `git checkout master` et ouvrir le fichier "README.md".
- Que s'est-il passé?

Ignorer des fichiers / dossiers

Dans tout projet logiciel, certains fichiers ne devraient pas être versionnés (identifiés et conservés par Git). Par exemple, un projet hébergé sur GitHub ne doit pas contenir les mots de passe de base de données d'un site en production. De nombreux fichiers générés par les IDE (Environnement de développement comme {Web,PHP}Storm) ne devraient pas être commités puisqu'ils sont spécifiques à chaque utilisateur du projet.

D'un autre côté, le développeur a parfois besoin de commiter un dossier vide. Par exemple, un dossier *uploads* qui contient les images de profil des utilisateurs du site. Mais Git ignore les dossier vides.

- Créer un fichier "A_IGNORER.txt" avec le contenu "ESGI".
- Exécuter la commande `git status`.
- Créer le fichier ".gitignore" (le "." en préfixe est indispensable!) avec le contenu "A_IGNORER.txt".
- Exécuter la commande `git status`.
- Créer le dossier "ESGI".
- Exécuter la commande `git status` : le dossier n'apparaît pas dans la liste des "fichiers" non suivis.
- Créer un fichier vide ".gitkeep" (le "." en préfixe est important!) vide.
- Exécuter la commande `git status`.
- Exécuter la commande `git add ESGI/`
- Que s'est-il passé?

L'ajout d'un fichier vide permet de révéler le dossier à Git. Le nom n'est pas obligatoire mais l'utilisation d'un fichier vide nommé ".gitkeep" est une convention reconnue par les utilisateurs de Git.

Pour en savoir plus sur Git ignore, parcourir la documentation officielle: <https://git-scm.com/docs/gitignore>

A lire impérativement et à expérimenter

- Parcourir la présentation sur Git: <http://rogerdudler.github.io/git-guide/index.fr.html>.
- Récupérer le Cheat sheet “Git & Github”: <https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf>, également disponible sur [Myges.fr](http://myges.fr).