



C2R1

# Tutorat Git

Version du 30 septembre 2014

# Table des matières

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>Introduction</b>                        | <b>2</b> |
| 1.1      | Histoire . . . . .                         | 2        |
| 1.1.1    | Versionnage de code? . . . . .             | 2        |
| 1.1.2    | Outils de versionnage . . . . .            | 2        |
| 1.2      | Installation du client git . . . . .       | 3        |
| 1.2.1    | Windows . . . . .                          | 3        |
| 1.2.2    | Linux . . . . .                            | 3        |
| 1.2.3    | Mac OS . . . . .                           | 3        |
| 1.3      | Choix du serveur git . . . . .             | 3        |
| <b>2</b> | <b>Manipulation basique de git</b>         | <b>4</b> |
| 2.1      | Début du projet . . . . .                  | 4        |
| 2.1.1    | Créer le dépôt . . . . .                   | 4        |
| 2.1.2    | Récupérer un dépôt existant . . . . .      | 4        |
| 2.2      | Premier commit . . . . .                   | 4        |
| 2.2.1    | pull . . . . .                             | 4        |
| 2.2.2    | readme . . . . .                           | 4        |
| 2.2.3    | add/diff/ignore/commit . . . . .           | 4        |
| 2.2.4    | push . . . . .                             | 6        |
| 2.3      | Historique . . . . .                       | 6        |
| <b>3</b> | <b>Manipulation avancée</b>                | <b>7</b> |
| 3.1      | Méthode de développement . . . . .         | 7        |
| 3.2      | branch . . . . .                           | 7        |
| 3.2.1    | créer sa branche . . . . .                 | 7        |
| 3.2.2    | fusionner . . . . .                        | 7        |
| 3.2.3    | résoudre un conflit . . . . .              | 7        |
| 3.2.4    | savoir qui a fait quoi . . . . .           | 7        |
| 3.3      | Se positionner sur un commit . . . . .     | 7        |
| 3.4      | Contribuer à un autre repository . . . . . | 7        |

# Partie 1

## Introduction

### 1.1 Histoire

#### 1.1.1 Versionnage de code ?

Lorsqu'on travaille à plusieurs sur un projet, il est fréquent de devoir travailler au même moment. C'est ce à quoi servent les gestionnaires de version. Ils permettent à chaque développeur de travailler localement, puis de l'envoyer aux autres développeurs une fois leurs modifications finies.

#### 1.1.2 Outils de versionnage

Il existe de nombreux outils de versionnage de code. Les 3 plus connus sont sans doute Git, Mercurial et SVN.

##### **SVN (subversion)**

Ce fut le plus utilisé pendant longtemps. Développé par la fondation Apache, il s'agit d'une amélioration d'un programme nommé CVS (très peu utilisé aujourd'hui). Le plus gros problème de SVN est qu'il s'agit d'un système centralisé. Un serveur contient donc le code, des clients travaillent dessus. Il n'y a qu'un seul versionning. C'est le gestionnaire de version utilisé notamment par Apache, FreeBSD et sourceforge.

##### **Mercurial**

Contrairement à SVN, il s'agit d'un système décentralisé. Chacun possède son propre *repository* et publie son code sur le *repository* public. Une autre différence avec SVN est qu'il utilise la notion de *changeset*. C'est à dire qu'il préfère garder en mémoire les changements appliqués que les versions des fichiers. Il est utilisé notamment par Mozilla (Firefox, Thunderbird, ...), Facebook, et adblock plus.

##### **Git**

Git possède très peu de différences avec Mercurial, mais l'histoire a fait qu'il c'est plus imposé. C'est en partie grâce à Linus Torvalds qui en a fait la pub et

des projets comme github que nous allons utiliser dans le reste du tutorat. C'est le gestionnaire de version utilisé par l'équipe de développeurs du noyau linux.

Pour plus d'informations sur les différences entre mercurial et Git, vous pouvez vous référer à ces articles : <http://importantshock.wordpress.com/2008/08/07/git-vs-mercurial/>, <http://www.rockstarprogrammer.org/post/2008/apr/06/differences-between-mercurial-and-git/>.

## 1.2 Installation du client git

### 1.2.1 Windows

Lancez le programme que vous pouvez trouver sur cette page : <http://msysgit.github.io> ou <http://www.git-scm.com/>

### 1.2.2 Linux

`apt-get install git` ou `yum install git` selon la distribution

### 1.2.3 Mac OS

`sudo port install git-core +svn +doc +bash_completion +gitweb`

## 1.3 Choix du serveur git

Pour la suite de ce tutorat, il vous faut créer un compte github : <https://github.com/>. Vous pouvez aussi vous intéresser à gitlab : <https://about.gitlab.com/>

## Partie 2

# Manipulation basique de git

## 2.1 Début du projet

### 2.1.1 Créer le dépôt

Pour créer le dépôt, il faut initialiser un dossier. Ceci se fait avec la commande `git init`. Ensuite, il faut créer le dépôt sur le serveur. Sur Github, ceci se fait avec un formulaire, accessible en cliquant sur le bouton *create repository*

### 2.1.2 Récupérer un dépôt existant

## 2.2 Premier commit

### 2.2.1 pull

### 2.2.2 readme

Le README est un fichier de présentation du projet. Généralement, on y décrit son installation, les fonctionnalités à venir, comment contribuer, la licence, ...


### 2.2.3 add/diff/ignore/commit

#### `git diff`

: montre les changements effectués depuis le dernier commit. Voici la forme d'un fichier diff :

```
diff --git a/apps/system/js/sound_manager.js b/apps/system/js/sound_manager.js
index b796d13..3847b93 100644
--- a/apps/system/js/sound_manager.js
+++ b/apps/system/js/sound_manager.js
@@ -686,7 +686,11 @@
    };
 }
```

Owner


 C2R1 ▾


 / 

Repository name

Great repository names are short and memorable. Need inspiration? How about [shiny-octo-avenger](#).

Description (optional)


☒  **Public**  
Anyone can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**  
This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: **None** ▾

 | 

Add a license: **None** ▾ 

Create repository

```
- function setVibrationEnabled(enabled) {  
+ function setVibrationEnabled(enabled) {  
+   //vibrate  
+   if ('vibrate' in navigator && enabled) {  
+     navigator.vibrate([200, 100, 200]);  
+   }  
+   setVibrationEnabledCount++;  
+   SettingsListener.getSettingsLock().set({  
+     'vibration.enabled': enabled
```

On y trouve donc le numéro du commit, les fichiers modifiés, les lignes supprimées ainsi que les lignes ajoutées.

#### 2.2.4 push

### 2.3 Historique

## Partie 3

# Manipulation avancée

### 3.1 Méthode de développement

### 3.2 branch

#### 3.2.1 créer sa branche

#### 3.2.2 fusionner

#### 3.2.3 résoudre un conflit

#### 3.2.4 savoir qui a fait quoi

### 3.3 Se positionner sur un commit

### 3.4 Contribuer à un autre repository

Généralement, il y a deux possibilités : \* Contribuer sans coder, (graphisme, traduction, communication, ouverture de bugs) \* Coder L'ouverture d'issues nécessite de regarder un minimum si l'issue n'a pas déjà été ouverte, de détailler son problème et de bien regarder la version qu'on utilise. Pour programmer, il faut forker le dépôt et le récupérer en local. Puis il suffit de créer sa branche, de réaliser les modifications nécessaires. Quelques fois, il est demandé de créer des tests pour son bout de code. Avant de proposer il faut vérifier son code (norme, lisibilité, ...) Enfin il faut push votre travail sur votre fork et effectuer une pull request. La suite dépend du fonctionnement du projet.