
git
Semifir
Adrien Vossough
https://www.semifir.com/

Avant-propos

Présentation

Un système de gestion de versions
ou VCS pour Version Control System
ou SCM pour Source Content Management
ou système de versionning :

- Est un système qui enregistre toutes les modifications apportées à une liste de fichiers.
- Il permet de suivre précisément l'évolution du contenu des fichiers.

Exemple de VCS :

- les Wiki (comme Wikipédia) pour les modifications des articles
- Stack Overflow pour les versions des questions et des réponses.

Présentation

Intérêt :

- Permet d'avoir un historique des modifications d'un projet et de revenir en arrière
- Garde les auteurs de chaque modification
- Permet le partage de code
- Facilite le travail en équipe
- Facilite la gestion des conflits lors de modifications sur un même fichier
- Facilite la documentation par un système de message pour chaque sauvegarde

Présentation

Git est décentralisé, et autorise un mode déconnecté :

Les sauvegardes d'un projet peut donc se faire localement puis peut être partagé sur un serveur

Les bases

Configuration obligatoire :

```
git config --global user.name "Prenom Nom"
```

```
git config --global user.email email@domaine.extension
```

Permet de configurer l'utilisateur

Les bases

Pour créer un **dépôt (lieu de sauvegarde)** locale, il faut se placer dans le répertoire contenant le projet.
Le répertoire doit contenir au moins un fichier.

git init

Permet d'initialiser un dépôt

Le fait d'initialiser un projet créé un répertoire .git caché.
Ce répertoire contient toutes les données du dépôt.
Supprimer ce répertoire efface l'historique Git, il faut donc refaire "git init"

Contenu du répertoire .git :

Il est rarement utile de modifier le contenu de ce répertoire

HEAD : référence du commit (sauvegarde) sur lequel on travail

branches : contient les branches du projet.
Les branches sont des versions qui divergent du développement principal et sont particulièrement utiles pour développer de nouvelles fonctionnalités sans créer de conflit avec la version stable du projet.

config : contient la configuration et les alias propres au dépôt.

description : ce fichier sera affiché lors de l'utilisation de l'interface web GitWeb.

hooks : contient les hooks , des mécanismes de contrôle utilisés par Git.

Contenu du répertoire .git :

info : contient un fichier exclude utilisé pour ignorer des fichiers.

objects : contient toutes les informations concernant les fichiers, dossiers, commits, ou tout autre objet que Git traite dans un projet

refs : comprend les références qui pointent vers des commits.
Ces références sont soit des branches, soit des tags

Le fichier README :

Permet d'expliquer, de présenter le projet. Il se place à la racine du projet.

Généralement :

- Une partie expliquant très rapidement le but du projet.
- Pour une bibliothèque, l'utilisation du projet dans un exemple de 5 à 10 lignes de code.
- Liens vers des ressources tel qu'une documentation plus complète, le site officiel, …
- Les informations pour les futurs contributeurs du projet.
- La liste des auteurs et contributeurs au projet, parfois dans le fichier AUTHORS à la racine du projet.
- La ou les licences, parfois dans le fichier LICENSE à la racine du projet.
- Moyen de contacter les auteurs.

Markdown est un langage de présentation :

Il a une structure XML comme pour le HTML et a pour but d'être lisible directement en mode texte, ses extensions sont : md, .markdown, .mdown , .mkdn , .mkd , .mdwn , .mdtxt , .mdtext , .text , .Rmd

La syntaxe officielle : http://daringfireball.net/projects/markdown/syntax

Exemple de fichier Markdown :

 # Titre principal

Titre secondaire

+ liste à puce
+ liste à puce

Markdown : Syntaxe

Le dièse # indique un titre ou un sous-titre.

Titre équivalent à h1 en HTML :
Titre de premier niveau

Sous-titre équivalent à h2 en HTML:
Titre de deuxième niveau

Listes non ordonnées équivalent de la balise ul en HTML
il faut utiliser un des caractères suivants au début de chaque ligne formant la liste : + * -

+ 64 unités différentes !
+ La meilleure IA jamais vue !
+ Un jeu compatible avec toutes les plateformes !

c. Listes ordonnées

Pour établir des listes ordonnées (équivalent de la balise ol en HTML), il faut préfixer chaque ligne de

Markdown : Syntaxe

Le dièse # indique un titre ou un sous-titre.

Titre équivalent à h1 en HTML :
Titre de premier niveau

Sous-titre équivalent à h2 en HTML:
Titre de deuxième niveau

Listes non ordonnées équivalent de la balise ul en HTML
il faut utiliser un des caractères suivants au début de ligne : + * -

+ premier élément
+ second élément
+ troisième élément

Markdown : Syntaxe

Pour des listes ordonnées équivalent de la balise ol en HTML

1. premier élément
2. second élément
3. troisième élément

Mettre en gras: exemple d'un **texte en gras**.

Mettre en italique : exemple d'un *texte en italique*.

Ajouter une ligne de séparation horizontale équivalente à <hr /> en HTML, il faut utiliser l'un des trois caractères suivants _ - * ainsi :
Avant

Après

Markdown : Syntaxe

Pour insérer du code il faut l'entourer de trois backtick : ```.
Il est également possible pour certains convertisseurs Markdown de spécifier le langage.

```html
<button>Bonjour</button>
```

Pour insérer une ligne de code dans une phrase
Pour déclarer une variable en java, **`int a = 45;`**

Pour insérer du code dans une liste, il faut l'indenter de huit espaces ou de deux tabulations.

Markdown : Syntaxe

Exemple de tableau :

```
|Prénom|Nom|
|---|----------|
|Adrien|Vossough|
|Moi|Lui|
```

Pour mettre des liens, la syntaxe est : [Texte du lien](URL du lien)
[Lien de Google](http://google.fr)

Pour insérer des notes de bas de page (ne marche pas partout) :
Projet sous licence MIT**[^1]**
[^1]: La licence MIT ou licence X11 est une licence de logiciel utilisée …