

Git et Gitlab au quotidien

Commandes Principales et Workflows

Généralités - Concepts - Premières Commandes

Benoît Bayol

June 9, 2015

Outline

Licence

Gestion de Version

Git

Concepts

Premières Commandes

Topic

Licence

Gestion de Version

Git

Concepts

Premières Commandes

- ▶ Les graphiques présents dans les slides sont extraits du livre "Pro Git 2" qui est aussi présent dans l'archive.
- ▶ La licence de ce document est :
<https://creativecommons.org/licenses/by-nc-sa/3.0/>

Topic

Licence

Gestion de Version

Git

Concepts

Premières Commandes

Principes

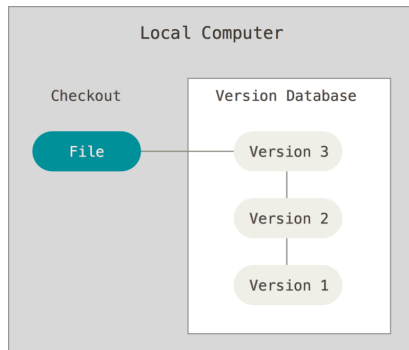
- ▶ enregistre l'évolution d'un fichier ou d'un ensemble de fichiers afin de :
 - ▶ suivre l'évolution
 - ▶ rappeler une version antérieure
 - ▶ diffuser les changements auprès des autres développeurs

Historique

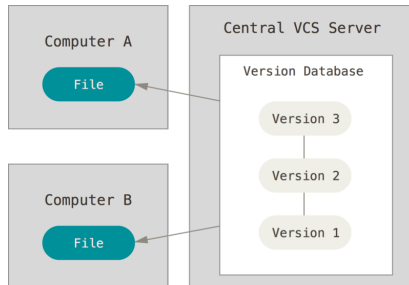
Generation	Commentaire	Exemples
0eme	À l'Arrache	<code>cp -r data/ data.old/</code>
1ere	Modèle Local	RCS (1982) - SCSS - ...
2eme	Modèle Centralisé	CVS (1990) - SVN (2000) - ...
3eme	Modèle Distribué	GIT (2005) - Mercurial - ...

source : http://ericsink.com/vcbe/html/history_of_version_control.html

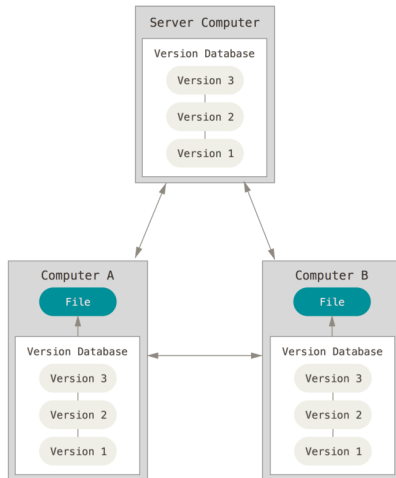
Modèle Local



Modèle Centralisé



Modèle Distribué



Démonstration du modèle décentralisé

```
1  cd /tmp
2  mkdir code
3  cd code
4  git init
5  echo "Hello World !" > README.txt
6  git add README.txt
7  git commit -m "first commit"
8  cd ~
9  git clone /tmp/code/
10 cd code
11 pwd
```

Topic

Licence

Gestion de Version

Git

Concepts

Premières Commandes

Introduction

- ▶ git est un logiciel :
 - ▶ développé depuis 2005

Période	Fait
fin-2005	git 1.0
2008	création github
2010	git flow (nvie)
2011	création gitlab
2012	déploiement gitlab @digiplante
début-2014	git 2.0
mi-2014	déploiement gitlab.ecp.fr
présent	git 2.4.1

- ▶ qui est découpé en plusieurs sous-programme (git-add, git-commit, ...)
- ▶ qui se configure
- ▶ qui établit un certain nombre de concepts afin de manipuler plus facilement l'ensemble
 - ▶ HEAD
 - ▶ index
 - ▶ push/pull
 - ▶ ...
- ▶ qui fonctionne presque entièrement de manière locale (à l'exception de clone, fetch, push, pull notamment)

- ▶ Toutes les commandes ont un man très bien fait.
- ▶ Il y a en plus des man relatifs à l'utilisation :
 - ▶ `man gittutorial`
 - ▶ `man gittutorial2`
 - ▶ `man giteveryday`
 - ▶ `man gitworkflows`
 - ▶ `man gitglossary`
- ▶ disponible sur le net
- ▶ grande communauté
 - ▶ en règle générale on trouve toujours une réponse sur [stackoverflow](#) par rapport à son utilisation

Topic

Licence

Gestion de Version

Git

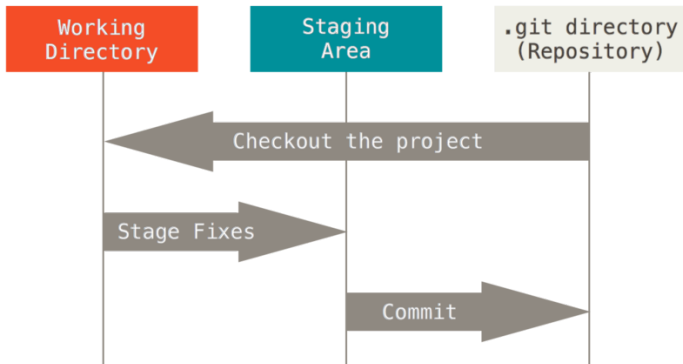
Concepts

Premières Commandes

- ▶ La notion de contenu est importante car si dans les anciens gestionnaires de version il y a une identification entre contenu et fichier, ce n'est plus le cas avec git. Ce dernier permet de descendre au niveau de la ligne de texte et faire des mix entre des lignes de textes présentes dans des fichiers différents.

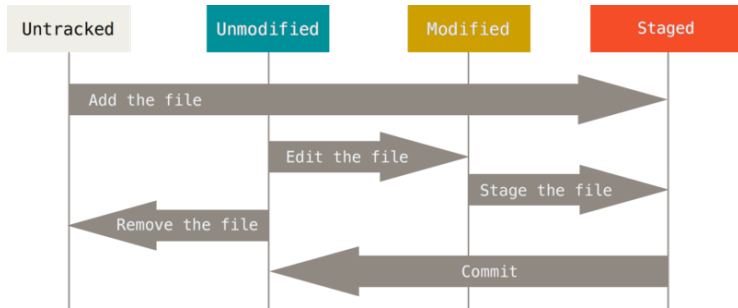
Lieux

- ▶ le dossier `.git` (`.git repository`)
- ▶ le dossier de travail (`working tree`)
- ▶ l'index (`index - staging area`)



- ▶ valide (stocké dans la base locale)
- ▶ modifié (pas encore valide)
- ▶ indexé (en attente dans l'index pour le prochain commit)

Exemple de cycle classique



- ▶ master le nom de la branche créée par défaut son nom n'a aucune importance et peut être changée dès le début
- ▶ branche un pointeur vers un commit
- ▶ commit un sha1 qui permet d'accéder à des données que git sait extraire dans le répertoire de travail
- ▶ sha1 un identifiant unique au sein d'un projet
- ▶ HEAD un pointeur sur la branche en cours

Topic

Licence

Gestion de Version

Git

Concepts

Premières Commandes

- ▶ La configuration peut être :
 - ▶ sur le système tout entier
 - ▶ pour l'utilisateur en cours
 - ▶ pour un projet en particulier
- ▶ Chaque niveau surcharge le précédent
- ▶ Elle concerne :
 - ▶ l'identité
 - ▶ l'éditeur par défaut
 - ▶ les adresses des remotes
 - ▶ les espaces blancs
 - ▶ ...
- ▶ `man git-config`

- ▶ initialise le dépôt git dans le dossier en cours si aucun nom sinon crée un dossier
- ▶ on aura donc :
 - ▶ un dossier .git
 - ▶ un répertoire de travail (working directory)
 - ▶ un moyen de passer de l'un à l'autre grâce à la manipulation de la zone d'indexation visible avec git-status
- ▶ options intéressantes
 - ▶ `-bare` (on décrira cette option au moment des pull/push)

- ▶ il faut bien garder à l'esprit que git versionne des instantanés du contenu et non pas des fichiers
- ▶ add rajoute du contenu dans le prochain commit
- ▶ options intéressantes
 - ▶ -u mettre dans l'index les fichiers déjà validé une fois (mieux que git add .)
 - ▶ -patch (-p) pour commiter de super commits

- ▶ options intéressantes
 - ▶ -m <message> permet de passer le message de commit dans la ligne de commande au lieu d'ouvrir un éditeur
 - ▶ -a permet de mettre dans l'index les fichiers déjà validé une fois automatiquement s'utilise bien avec -am <message>
 - ▶ -dry-run

- ▶ git-log
- ▶ options intéressantes
 - ▶ `-graph`
 - ▶ `-decorate`
 - ▶ `-since/after`
 - ▶ `-extended-regexp` `-regexp-ignore-case` `-grep <pattern>` (en plus court `-E -i -grep`)
 - ▶ `-author=<email>`
 - ▶ `-patch`
 - ▶ `-stat`

Marqueurs

- ▶ `sha1^` veut dire un commit avant sha1
- ▶ `sha1^^` veut dire 2 commits avant sha1
- ▶ `sha1~N` veut dire N commits avant sha1

double dot



- A..B signifie "atteignable par B mais pas par A"

```
1 git log master..experiment
2 git log experiment --not master
3 D
4 C
```

```
1 git log experiment..master
2 git log experiment --not master
3 F
4 E
```

triple dot

- ... signifie "atteignable par A et B mais pas les deux"

```
1 git log master...experiment
2 F
3 E
4 D
5 C
```

- ▶ diff montre les changements entre différents commits
- ▶ par défaut, diff fais la différence entre le working directory et l'index cela vous permet de voir ce que vous pouvez ajouter pour le prochain commit

```
1 git diff
```

- ▶ si on veut voir la différence entre le HEAD et le commit que l'on va faire il faut ajouter l'option `--cached` (ou `--staged`)

```
1 git diff --staged
```

- ▶ options intéressantes
 - ▶ `-b`, `--ignore-space-change`
 - ▶ `-w`, `--ignore-all-space`
 - ▶ `--ignore-blank-lines`
 - ▶ `stat`

- ▶ status permet d'avoir un aperçu des différents états en cours et de comprendre ce que l'on a dans l'index
- ▶ options intéressantes
 - ▶ uno

.gitignore

- ▶ .gitignore est un fichier spécial dans lequel vous pouvez mettre des patterns qui seront exclus du versionnement
- ▶ peut être surchargé par `–ignored` sur `git status`