# TP initiation à GIT

Bill Allombert

11 septembre 2015

**Initiation au système de version distribué GIT (`http://git-scm.com/`).**

## 1 Conventions

1. Les lignes préfixées par `>` désignent des commandes à entrer dans un shell. La liste des commandes est disponible à l'URL `http://pari.math.u-bordeaux.fr/~bill/git/tutorial.txt`.
2. Les commandes `echo` et `perl` simulent la modification de fichiers par l'utilisateur.

## 2 Concepts GIT

1. Les objets GIT (noms en hexadécimal)
   - Les blob : contenu des fichiers
   - Les tree : arborescence de repertoire (liste de blob)
   - Les commit : historique (un tree + un message + commit parent)
2. Les références (refs) : noms symboliques pour des commit. Plusieurs type :
   - Dynamique : HEAD, HEAD 1
   - Statique : les tags
   - Ajout : les branches changent par ajout de commit
   - Flottante : les branches distantes
3. Les "remote" : URL vers un dépot distant. Permet de gérer les transferts entre dépots.

## 3 Commandes de base

Dans cette section, nous allons utiliser git pour la gestion d'une présentation TeX.

### 3.1 Configuration

Configurez votre identité ainsi : (Important : l'indentité permet d'identifier l'auter d'un commit)

```
> git config --global user.name "François Viète"
> git config --global user.email Francois.Viete@math.u-bordeaux1.fr
```

La commande "git config" permet de configurer git.

### 3.2 Initialisation

Initialisation d'un nouveau projet GIT.

```
> mkdir projet
> cd projet
> git init
> ls -la
> echo '\documentclass{article}' > projet.tex
> git add projet.tex
> git commit
```

### 3.3 Commandes de bases

```
> echo '\begin{document}' >> projet.tex
> git diff
> git show
> git commit
> git help commit
> git add projet.tex
> git diff
> git diff --cached
> git commit
> echo 'my project' >> projet.tex
> git commit projet.tex
> echo '\end{document}' >> projet.tex
> git commit -a
> git log
> git show HEAD~1
> git show HEAD~2
> git log -p HEAD~2..HEAD
```

### 3.4 Clonage d'un dépot existant

```
> cd ..
> git clone ssh://acces.math.u-bordeaux.fr/tmp/bill/projet2
```

### 3.5 Création de branches

`git init` crée une branche de départ appelé 'master'

```
> git branch -a
> git checkout -b slides-montpellier
> echo '\usetheme{Montpellier}' >> projet.tex
> git diff
> git diff master
> git diff HEAD master
> git commit -a
> git diff
> git diff master
> git diff HEAD master
```

```
> git checkout master
> echo 'all:' > Makefile
> git add Makefile
> git commit -a
```

### 3.6 Ajout d'un dépot distant

La commande `git remote` permet de manipuler les remote.

```
> ssh acces.math.u-bordeaux.fr git init --bare projet.git
> git remote add origin ssh://acces.math.u-bordeaux.fr/~/projet.git
> git push origin HEAD
> echo '\begin{theorem}' >> projet.tex
> git commit -a
```

### 3.7 Accès à un dépot distant

```
> cd ..
> git clone ssh://acces.math.u-bordeaux.fr/~/projet.git projet3
> cd projet3
> git branch -a
> git diff origin/master
> git fetch
> git diff origin/master
> git merge origin/master
> printf '\tlatex projet.tex' >> Makefile
> git commit Makefile
> git push origin master:fix-makefile
> cd ../projet
> git branch -a
> git fetch
> git branch -a
> git rebase origin/fix-makefile
> git log
> git push origin :fix-makefile
> cd ../projet2
> git fetch
> git diff origin/master
```

### 3.8 Gestion des conflits

```
> git checkout -b end-theorem
> echo '\end{theorem}' >> projet.tex
> git commit -a
> cd ../projet
> echo 'Soit K un corps' >> projet.tex
> git commit -a
> git push origin HEAD
> cd ../projet2
> git fetch
> git rebase origin/master
```

```
> git diff
> perl -pi -e 's/^[<=>].*//' projet.tex
> git diff
> git add projet.tex
> git rebase --continue
> git push origin HEAD
> git push origin :end-theorem
> git push origin HEAD
> cd ../projet
> echo 'Soit p un nombre premier' >> projet.tex
> git commit -a
> git fetch
> git merge origin/end-theorem
> git diff
> perl -pi -e 's/^[<=>].*//' projet.tex
> git diff
> git commit -a
```

### 3.9 Accès à l'historique

```
> git log
> git show montpellier
> git archive montpellier > montpellier.tar
> tar tf montpellier.tar  | less
> git checkout montpellier
> git branch
> git checkout montpellier -b new-theme
> git branch
```

## 4 Commandes avancées

- `git cherry-pick` Permet de faire un `git merge` sur un seul commit.
- `git reset` Permet de supprimer des commit.
- `git revert` Crée un commit annulant l'éffet d'un commit précédent.
- `git stash` Sauve les modifications dans une pile pour etre récupérées plus tard.
- `git reflog` Donne la liste des commit récémment accédés.
- `git remote` Permet de configurer les dépots distants.
- `git bisect` Permet de faire une recherche de bug par bisection.