Introduction to GIT and the PARI repository

Bill Allombert

January 23, 2012

Introduction to GIT (`http://git-scm.com/`).

Conventions

1. Please enter lines prefixed by `>` in a terminal.
2. You can get the list of command to type in plain text for easy copy-pasting at `http://pari.math.u-bordeaux.fr/~bill/git/tutorial_en.txt`.

1 Access to the PARI repository

The following commands allow to access the PARI repository: See http://pari.math.u-bordeaux.fr/git.html for general information. Since the initial import is slow, move to the next session while waiting.

```
> git clone http://pari.math.u-bordeaux.fr/git/pari.git
```

2 Basic use

In this section, we use git to manage a simple file on a single computer. For example, you can manage a LaTeX paper this way.

2.1 Configuration

Set your identity as follow:

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

2.2 Initialization

We start a new project:

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

2.3 Basic commands

- `git help` Show internal help.
- `git commit -a` Commit all changes.
- `git diff` Show uncommited changes.
- `git show` Show most recent commit.
- `git log` Show history.
- `git add` Add a new file.

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

2.4 Fixing mistake

Git allow you to fix mistakes as long as you did not publish your changes.

```
> echo '\end{docment}' >> project.tex
> git commit -a
> git show
> perl -pi -e s/docment/document/ project.tex
> git diff
> git commit -a --amend
> git show
```

2.5 Branches and tags

The command git clone should have completed by now.

- `git branch` List local branches.
- `git branch -a` List all branches.
- `git checkout` Move to a branch.
- `git checkout -b` Create a new branch.

```
> cd ../pari
> git branch
> git branch -a
> git tag -l
> head CHANGES
> git log
```

```
> git checkout origin/pari-2.5
> head CHANGES
> git log
> git checkout -b local-pareval origin/bill-pareval
> git log
> git branch
> git checkout master
```

3 PARI development

3.1 Compiling PARI

```
> ls
> ./Configure --prefix=amd64
> make -j4 bench
> make gp
> make install
```

3.2 Making a change

```
> git checkout -b local-symbolic_help
> cd src/functions/symbolic_operators
```

Now pick a file randomly from this list: `and concat eq ge gt le lt mm ne neg not pl pp shiftl shiftr` and edit it to improve the short help in the line "Help:"

```
> cd ../../..
> make gp
> git commit -a
```

Check that the help works.

```
> git show
> git format-patch master
```

You should get a file `0001-xxxx.patch`. Please send it to `Bill.Allombert@math.u-bordeaux.fr`. Thanks for your contribution to PARI/GP!

3.3 Updating your repository

- `git fetch` Download changes from remote repositories.
- `git rebase` Update a branch with changes from another.

```
> git fetch
> git checkout master
> git rebase origin/master
   You can also update your local branches:
> git checkout local-symbolic_help
> git rebase origin/master
```

4 Handling conflicts

```
> cd ../project
> git checkout -b end-theorem
> echo '\end{theorem}' >> project.tex
> git commit -a
> git checkout master
> echo 'Let K be a field' >> project.tex
> git commit -a
> git checkout end-theorem
> git rebase master
> git diff
> perl -pi -e 's/^[<=>].*//' project.tex
> git diff
> git add project.tex
> git rebase --continue
```

5 Advanced use

5.1 Concepts

- A commit is a state of the repository with all its history. It has a hexadecimal number.
- A ref is a symbolic name for a commit. It can be a branch, a tag, etc.
- A remote is a symbolic name for another (possibly remote) repository. When cloning a repository, a remote `origin` pointing to the remote repository is added.

5.2 Advanced commands

- `git remote` Allow to manipulate remotes.
- `git cherry-pick` Allow to apply a single commit from a branch.
- `git reset` Reset repository state and history.
- `git revert` Create a commit canceling the effect of a previous commit.
- `git stash` Save current changes in a stack to be pop'ed later.
- `git reflog` Give the list of recent checkouts.
- `git remote` Allow to manipulate access to remote repository.
- `git bisect` Allow to find by binary search the change that introduced a bug.