Configuring Git

Matthieu Moy

Matthieu.Moy@imag.fr

http://www-verimag.imag.fr/~moy/cours/formation-git/configuring-git-slides.pdf

2015

Configuration of Git

- What you already did:
  - Introduce yourself (`user.name` = ... , `user.email` = ...)
  - Tell Git about your favorite editor (`core.editor`)
  - Tell Git to ignore some files (`.gitignore`)
- What we're about to do:
  - Learn where the config files are
  - Learn how to read the docs

Outline

Git Configuration: Which Files

- 3 places
  - System-wide: `/etc/gitconfig`
  - User-wide (“global”): `~/.gitconfig` or `~/.config/git/config`
  - Per-repository: `$project/.git/config`
- Precedence: per-repo overrides user-wide overrides system-wide.
- Not versionned by default, not propagated by `git clone`

Git Configuration: Syntax

- Simple syntax, key/value:

```
[section1]
 # This is a comment
        key1 = value1 # comment as well
        key2 = value2
[section2 "subsection"]
        key3 = value3
```

- Semantics:
  - “`section1.key1` takes value `value1`”
  - “`section1.key2` takes value `value2`”
  - “`section2.subsection.key3` takes value `value3`”
- “section” and “key” are case-insensitive.

Querying/Modifying Config Files

```
# Modify per-repo .git/config:
$ git config user.name 'Matthieu Moy'
$ cat .git/config
...
[user]
        name = Matthieu Moy
# Modify user-wide ~/.gitconfig:
$ git config --global user.name 'Matthieu Moy'
# Get the value of a variable:
$ git config user.name
Matthieu Moy
```

Some Useful Config Variables

- User-wide:

  user.name, user.email Who you are (used in `git commit`)

  core.editor Text editor to use for `commit`, `rebase -i`, ...

- Per-repo:

  remote.origin.url Where to fetch/push

Aliases

```
# Definition
$ cat .git/config
...
[alias]
        lg = log --graph --oneline

# Use
$ git lg
*   a5da80c Merge branch 'master' into HEAD
|\
| * 048e8c1 bar
* | 5034527 boz
|/
* 1e0e4a5 foo
```

Documentation about Configuration

- `man git-config` : documents all configuration variables (> 350)
- Example:

```
user.email
    Your email address to be recorded in any
    newly created commits. Can be overridden
    by the GIT_AUTHOR_EMAIL, GIT_COMMITTER_EMAIL,
    and EMAIL environment variables.
    See git-commit-tree(1).
```

Outline

Ignore Files: Why?

- Git needs to know which files to track (`git add`, `git rm`)
- You don't want to forget a `git add`
- ⇒ `git status` shows `Untracked files` as a reminder. Two options:
  - `git add` them
  - ask Git to ignore: add a rule to `.gitignore`
- Only impacts `git status` and `git add`.

Ignore Files: How?

- `.gitignore` file contain one rule per line:

```
# This is a comment

# Ignore all files ending with ~:
*~
# Ignore all files named 'core':
core
# Ignore file named foo.pdf in this directory:
/foo.pdf
```

Ignore Files: Where?

- User-wide: `~/.config/git/ignore`:
  - Example: your editor's file like `*~` or `.*.swp`
  - Don't disturb your co-workers with your personal preferences
  - Set once and for all
- Per-repo, not versionned: `.git/info/exclude`
  - Not very useful ;-)
- Tracked within the project (`git add` it): `.gitignore` in any directory, applies to this directory and subdirectories.
  - Generated files (especially binary)
  - Example: `*.o` and `*.so` for a C project
  - Share with people working on the same project

About Generated Files

- Versionning (`git add`-ing) generated files is bad

About Generated Files

- Versionning (`git add`-ing) generated files is bad
- Versionning generated binary files is very bad

About Generated Files

- Versionning (`git add`-ing) generated files is bad
- Versionning generated binary files is very bad
- Why?
  - breaks `make` (timestamp = `git checkout` time)
  - breaks merge
  - eats disk space (inefficient delta-compression)