

Configuring Git

Sylvain Bouveret, Grégory Mounié, Matthieu Moy
2017

[first].[last]@imag.fr

<http://recherche.noiraudes.net/resources/git/configuring-git-slides.pdf>



- Explain how to configure git
- Explain where and how configuration files are stored
- Explain how to tell git to ignore files

Configuration 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`

- 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.

```
1 # Modify per-repo .git/config:
2 $ git config user.name 'Matthieu Moy'
3 $ cat .git/config
4 ...
5 [user]
6     name = Matthieu Moy
7 # Modify user-wide ~/.gitconfig:
8 $ git config --global user.name 'Matthieu Moy'
9 # Get the value of a variable:
10 $ git config user.name
11 Matthieu Moy
```

- 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

```
1 # Definition
2 $ cat .git/config
3 ...
4 [alias]
5     lg = log --graph --oneline
6
7 # Use
8 $ git lg
9 *   a5da80c Merge branch 'master' into HEAD
10  |\
11  | * 048e8c1 bar
12  * | 5034527 boz
13  |/\
14  * 1e0e4a5 foo
```


- `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).
```

(Git)Ignore files

- Git needs to know which files to track (`git add`, `git rm`)
- You don't want to forget a `git add`
- \Rightarrow `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`.

- `.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 in any auto directory:
```

```
auto/
```

```
# Ignore html file in subdir of any Doc directory:
```

```
Doc/**/*.html
```

- 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

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

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

- 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)