

# Configuring Git

---

Sylvain Bouveret, Grégory Mounié, Matthieu Moy

2021

[first].[last]@imag.fr

[https://git.pages.ensimag.fr/formation-git/slides/  
configuring-git-handout.pdf](https://git.pages.ensimag.fr/formation-git/slides/configuring-git-handout.pdf)



## Goals of the presentation

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

Configuring Git

2 / 12

# Configuration files

---

## 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`

Configuring Git

3 / 12

- 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.

```
# 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
```

- 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

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

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

---

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