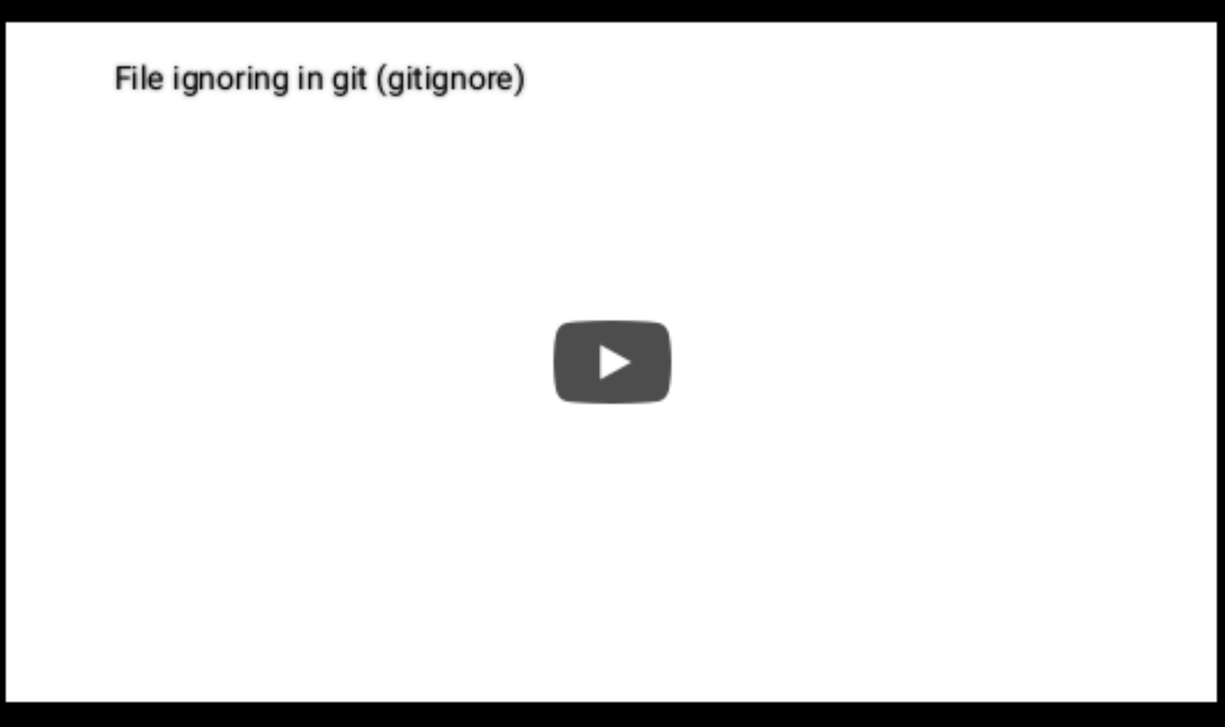
Игнорирование файлов в Git



Видео может быть заблокировано из-за расширений браузера. В статье вы найдете решение этой проблемы.

Я не знаю ни одного проекта, в рабочей директории которого не появлялось бы таких файлов, которые не нужно игнорировать. Зачастую, у вас имеется группа файлов, которые вы не только не хотите автоматически добавлять в репозиторий, но и видеть в списках неотслеживаемых (а чем взрослее проект, тем больше таких файлов может быть). К таким файлам обычно относятся автоматически генерируемые файлы (различные логи, результаты сборки программ и т.п.).

Как я говорил в вводной лекции, в таком случае, вы можете создать файл .gitignore с перечислением шаблонов соответствующих таким файлам. Хорошая практика заключается в настройке файла .gitignore до того, как начать серьёзно работать, это защитит вас от случайного добавления в репозиторий файлов, которых вы там видеть не хотите.

Вот пример файла .gitignore для типового Rails приложения (пример взят с github и является стандартом де-факто для большинства репозиториев с проектами на Ruby on Rails):

```
$ cat .gitignore

*.rbc
capybara-*.html
.rspec
/log
/tmp
/db/*.sqlite3
/public/system
/coverage/
/spec/tmp
**.orig
rerun.txt
pickle-email-*.html

# TODO Comment out these rules if you are OK with secrets being uploaded to
the repo
config/initializers/secret_token.rb
config/secrets.yml

## Environment normalisation:
/.bundle
/vendor/bundle

# these should all be checked in to normalise the environment:
# Gemfile.lock, .ruby-version, .ruby-gemset

# unless supporting rvm < 1.11.0 or doing something fancy, ignore this:
.rvmrc
```

Давайте повнимательнее рассмотрим этот файл: В первой строке написано, что необходимо игнорировать все файлы, которые заканчиваются на .rbc, в четвертой, что необходимо игнорировать директорию log, в шестой - все файлы конфигураций баз данных sqlite3. Сразу видно, что для большинства правил используются некоторые шаблоны. Но можно указывать и полный путь до конкретного файла, как сделано в 15 и 16 строках. Git всегда мягко применяет эти правила. Если вы указали, что необходимо игнорировать просто строку test - то он будет игнорировать и директории (вместе со вложенными файлами) и файлы, которые называются test, вне зависимости, где они располагаются. Если вы хотите ограничиться только корневым уровнем в репозитории - необходимо явно это указать поставив "/" перед шаблоном.

К шаблонам в файле .gitignore применяются следующие правила:

- Пустые строки, а также строки, начинающиеся с #, игнорируются.
- Можно использовать стандартные glob шаблоны.
- Можно заканчивать шаблон символом слэша (/) для указания каталога.
- Можно инвертировать шаблон, использовав восклицательный знак (!) в качестве первого символа.

Glob-шаблоны представляют собой упрощённые регулярные выражения используемые командными интерпретаторами. Символ * соответствует 0 или более символам; последовательность [abc] — любому символу из указанных в скобках (в данном примере a, b или c); знак вопроса (?) соответствует одному символу; [0-9] соответствует любому символу из интервала (в данном случае от 0 до 9). Вот ещё один пример файла .gitignore:

```
# комментарий — эта строка игнорируется
# не обрабатывать файлы, имя которых заканчивается на .a
*.a
# НО отслеживать файл lib.a, несмотря на то, что мы игнорируем все .a файлы с
помощью предыдущего правила
!lib.a
# игнорировать только файл TODO находящийся в корневом каталоге, не относится
к файлам вида subdir/TODO
/TODO
# игнорировать все файлы в каталоге build/
build/
# игнорировать doc/notes.txt, но не doc/server/arch.txt
doc/*.txt
# игнорировать все .txt файлы в каталоге doc/
doc/**/*.txt
```

Шаблон **/ доступен в Git, начиная с версии 1.8.2.

Временно игнорировать изменения в файле можно командой:

```
git update-index --assume-unchanged <file>
```

Отключается командой:

```
git update-index --no-assume-unchanged <file>
```

Если файл попал в индекс и нужно убрать его из индекса:

```
git rm --cached path/to/file
```

Однако, стоит быть осторожнее с командой git rm.

Однако, стоит обратить внимание на то, что файл .gitignore не решение от всех проблем. То есть он проблему то решает, но не стоит его всегда использовать. Иногда встречаю в файле .gitignore то, чего там быть никак не должно. Давайте представим такую ситуацию: Я и вы работаем в одной команде над одним проектом. Я в своей работе использую vim, а вы, например, IDE от JetBrains. У меня временные файлы создаются в специальном каталоге в профиле пользователя, тем самым в директории проекта я никак не создаю временных файлов при редактировании проекта. У вас же в проекте создается директория .idea, в которой лежат конфиги вашей IDE для этого проекта. Это часть вашего рабочего окружения и она никаким боком не относится к проекту и репозиторию. По идее, вы можете добавить строчку /.idea в файл .gitignore в проекте, однако, если над проектом работает несколько человек и каждый из них добавит конфиги своего окружения в .gitignore, то он превратится в нечитаемую помойку.

Что делать и как быть? Есть несколько разных способов игнорирования файлов в git.

1. Исключения для проекта

Этот как раз тот самый .gitignore, в корне репозитория. В него стоит помещать в основном только то, что имеет непосредсвенное оношение к проекту и его архитектуре. Например, у всех участников проекта есть директория с локами, или у всех в одном и том же месте создаются временные файлы. Если эти правила имеют отношение ко всем участникам проекта - то стоит правила игнорирования разместить в .gitignore, который будет распространяться вместе с репозиторием.

1. Исключение для компьютера

Когда у вас несколько проектов и везде создается что-либо, что вы не хотите коммитить(например, *.swp файлы Vim) используйте ~/.gitconfig. Вышеприведённый пример папки .idea, которая создается для каждого проекта как раз подходит сюда. Создайте файл .gitexcludes и выполните:

```
git config --global core.excludesfile ~/.gitexcludes
```

или вручную добавьте в ~/.gitconfig:

```
[core]
  excludesfile = ~/.gitexcludes
```

1. Исключение для репозитория

Иногда возможны случаи, когда у вас есть файлы, которые специфичны для данного проекта и для вашего рабочего окружения (например, логи third-party утилиты, которую вы любите использовать и используете только в этом проекте). К проекту отнести шаблон игнорирования - не верно. Вы не используете ее во всех проектах и значит игнорировать глобально на всем компьютере - тоже не стоит. В данном случае используйте .git/info/exclude. Этот файл не коммитится и остается только в локальном репозитории.

Сразу после сохранения ~/.gitconfig вы не должны видеть указанные файлы/папки в списке Untracked files.

Я люблю Git, но и он порой непоследователен в мелочах. Обращаю ваше внимание на то, что в первом случае мы редактируем файл.git/info/exclude (без s на конце), а во втором используем опцию excludeSfile (с s в середине). Не потеряйте время из-за возможной опечатки.

Работа с Git в большинстве случаев означает работу в команде, поэтому не усложняйте жизнь тем, кто будет работать с вами деталями вашего рабочего окружения. Хороших коммитов! :)