



Шпаргалка с основными командами для Git

Конфигурация

`git config --global user.name "[name]"` — установить имя, которое будет прикрепляться к коммиту.

`git config --global user.email "[email address]"` — установить email, который будет прикрепляться к коммиту.

`git config --global color.ui auto` — включить полезную подсветку командной строки.

`git config --global push.default current` — обновлять удаленную ветку с таким же именем, что и локальная, при пуше изменений (если не указано иного).

`git config --global core.editor [editor]` — установить редактор для редактирования сообщений коммита.

`git config --global diff.tool [tool]` — установить программу для разрешения конфликтов при слиянии.

Создание репозиториев

`git init [project-name]` — создать новый локальный репозиторий с заданным именем.

`git clone [url]` — загрузить проект и его полную историю изменений.

Работа с изменениями

`git status` — полный список изменений файлов, ожидающих коммита.

`git status -s` — краткий вид изменений.

`git diff` — показать изменения в файлах, которые еще не были добавлены в индекс коммита (staged).

`git add [file]` — сделать указанный файл готовым для коммита.

`git add .` — сделать все измененные файлы готовыми для коммита.

`git add '*.txt'` — добавить только файлы, соответствующие указанному выражению.

`git add --patch filename` — позволяет выбрать какие изменения из файла добавятся в коммит.

`git diff --staged` — показать что было добавлено в индекс с помощью `git add`, но еще не было закоммичено.

`git diff HEAD` — показать что изменилось с последнего коммита.

`git diff HEAD^` — показать что изменилось с предпоследнего коммита.

`git diff [branch]` — сравнить текущую ветку с заданной.

`git difftool -d` — то же самое, что и `diff`, но показывает изменения в заданной `difftool`.

`git difftool -d master..` — показать изменения, сделанные в текущей ветке.

`git diff --stat` — показать статистику какие файлы были изменены и как.

`git reset [file]` — убрать файлы из индекса коммита (изменения не теряются).

`git commit` — записать изменения в репозиторий. для написания сообщения откроется назначенный редактор.

`git commit -m "[descriptive message]"` — записать изменения с заданным сообщением.

`git commit --amend` — добавить изменения к последнему коммиту.

Работа с ветками

`git branch` — список всех локальных веток в текущей директории.

`git branch [branch-name]` — создать новую ветку.

`git checkout [branch-name]` — переключиться на указанную ветку и обновить рабочую директорию.

`git checkout -b <name> <remote>/<branch>` — переключиться на удаленную ветку.

`git checkout [filename]` — вернуть файл в первоначальное состояние если он еще не был добавлен в индекс коммита.

`git merge [branch]` — соединить изменения в текущей ветке с изменениями из заданной.

`git merge --no-ff [branch]` — соединить ветки без режима [“fast forwarding”](#).

`git branch -a` — посмотреть полный список локальных и удаленных веток.

`git branch -d [branch]` — удалить заданную ветку.

`git branch -D [branch]` — принудительно удалить заданную ветку, игнорируя ошибки.

`git branch -m <oldname> <newname>` — переименовать ветку.

Работа с файлами

`git rm [file]` — удалить файл из рабочей директории и добавить в индекс информацию об удалении.

`git rm --cached [file]` — удалить файл из репозитория, но сохранить его локально.

`git mv [file-original] [file-renamed]` — изменить имя файла и добавить в индекс коммита.

Отслеживание файлов

`.gitignore` — текстовый файл, в котором задаются правила для исключения файлов из репозитория. Например:

- *.log
- build/
- temp-*

`git ls-files --other --ignored --exclude-standard` — список всех игнорируемых файлов.

Сохранение фрагментов

`git stash` — положить во временное хранилище все отслеживаемые файлы.

`git stash pop` — восстановить последние файлы, положенные во временное хранилище.

`git stash list` — список всех сохраненных изменений во временном хранилище.

`git stash drop` — удалить последние файлы, положенные во временное хранилище.

Просмотр истории

`git log` — список изменения текущей ветки.

`git log --follow [file]` — список изменения текущего файла, включая переименования.

`git log --pretty=format:"%h %s" --graph` — изменение вида отображения истории изменений.

`git log --author='Name' --after={1.week.ago} --pretty=oneline --abbrev-commit` — посмотреть над чем работал заданный пользователь последнюю неделю.

`git log --no-merges master..` — посмотреть историю изменений только для текущей ветки.

`git diff [file-branch]..[second-branch]` — посмотреть различия между двумя заданными ветками.

`git show [commit]` — показать метадату и изменения в заданном коммите.

`git show [branch]:[file]` — посмотреть на файл в другой ветке, не переключаясь на неё.

Отмена коммитов

`git reset` — убрать изменения из индекса коммита, сами изменения останутся.

`git reset [commit/tag]` — отменить все коммиты после указанного коммита, изменения будут сохранены локально.

`git reset --hard [commit]` — принудительно вернуться к указанному коммиту, не сохраняя историю и изменения.

Синхронизация изменений

`git fetch [bookmark]` — загрузить всю историю с заданного удаленного репозитория.

`git merge [bookmark]/[branch]` — слить изменения локальной ветки и заданной удаленной.

`git push` — запустить текущую ветку в удаленную ветку.

`git push [remote] [branch]` — запустить ветку в указанный репозиторий и удаленную ветку.

`git push [bookmark] :[branch]` — в удаленном репозитории удалить заданную ветку.

`git push -u origin master` — если удаленная ветка не установлена как отслеживаемая, то сделать ее такой.

`git pull` — загрузить историю и изменения удаленной ветки и произвести слияние с текущей веткой.

`git pull [remote][branch]` — указать конкретную удаленную ветку для слияния.

`git remote` — посмотреть список доступных удаленных репозиториев.

`git remote -v` — посмотреть детальный список доступных удаленных репозиториев.

`git remote add [remote][url]` — добавить новый удаленный репозиторий.

Полезные ссылки

- 19 советов по повседневной работе с Git. [оригинал](#). [перевод](#).
- Книга [“Pro Git”](#). Издание первое, на русском.
- Книга [“Pro Git”](#). Издание второе, на английском.
- Git Rebase: [руководство по использованию](#).

[Твитнуть](#) [Поделиться](#) [Поделиться](#) 1 [Отправить](#)

2016 [git](#)

[← Первый скрипт на Python](#) [Почему стоит использовать zsh вместо bash →](#)

0 Комментариев

Блог Анатолия Гладкого

Политика конфиденциальности Disqus

Войти

Рекомендовать 5

Твитнуть

Поделиться

Лучшее в начале

Начать обсуждение...

войти с помощью

или через DISQUS

Имя

Прокомментируйте первым.

Подписаться

Добавь Disqus на свой сайтДобавить DisqusДобавить

Do Not Sell My Data