# All experience

mrzlanx532

20 марта 2020 г.

## Содержание

# 1 Git commands

## 1.1 Git common's commands

**git help <команда>**
открывает документацию по команде

**git init**
инициализация git-репозитория

## 1.2 Git config

**git config --global core.editor *редактор кода***
установка стандартного редактора кода

**git config --list**
список всех настроек локально (или глобально --global)

**git config --global alias.co checkout**
создание краткого алияса (команда checkout будет выполняться через команду "co")

**git config --global alias.co 'checkout -b'**
создание краткого алияса "co" с параметрами

**git config core.fileMode false**
выключить отслеживания за ПРАВАМИ ФАЙЛА

## 1.3 Git status

**git status**
проверка на существование измененных файлов

**git status -s (--short)**
краткий вывод состояния файлов

## 1.4 Git add

**git add ***
Добавить все файлы в индексацию

**git add <file>**
Добавить <file> в индексацию

## 1.5 Git clone

**git clone https://github.com/libgit2/libgit2.git**
клонирование репозитория

**git clone <url>**
клонировать репозиторий

**git clone -o <локальное_название_источника>**
склонировать репозиторий, и дать источнику имя <локальное_название_источника>

## 1.6 Git diff

**git diff**
посмотреть изменение файлов в консоли до индексации

**git diff --cached**
покажет проиндексированные изменения

**git diff --staged**
что из проиндексированного войдет в следующий коммит, воспользуйтесь командой

## 1.7 Git log

**git log**
показать все последние коммиты

**git log -p**
показать все последние коммиты с изменениями

**git log -p -1**
показать 1 последний коммит с изменениями

**git log --stat**
показать все коммиты с измененными файлами кратко

**git log --pretty=oneline**
показ комитов с описанием

**gitlog --graph**
удобное представление логов

**git log --relative-date**
относительная дата коммита (35 минут назад)

## 1.8 Git branch

**git branch -d <имя_ветки>**
Удалить локальную ветку

**git branch <имя_локальной_ветки>**
создать локальную ветку

**git branch -d <имя_ветки>**
удалить локальную ветку

**git branch -u <удаленная_ветка>**
начать отслеживать удаленную ветку с текущей

**git branch -vv**
показать информацию о ветках и за чем они следят

## 1.9 Git remote

**git remote -v**
посмотреть подключенный удаленный источник

**git remote add <локальное_название_источника> <url>**
добавить новый источник

**git remote rename <старое_имя_уд._реп> <новое_имя_уд._реп>**
переименовать удаленный источник

**git remote rm <имя_уд._реп>**
удалить ссылку на удаленный реп

## 1.10 Git stash

**git stash**
убрать в стэш изменения

**git stash list**
показать список изменений в стэшэ

**git stash pop**
вытащить из стэша (и удалить из стэка стэша) последние изменения

**git stash apply**
вытащить последние изменения (но не удалять из стэка стэша) спрятанные в стэш

**git stash list | pop stash@{0}**
вытащить по порядковому номеру изменения из стэша

**git stash drop stash@{0}**
удалить из стэка стэша изменения у последнего коммита

**git stash --keep-index**
убрать в стэш только непроиндексированные файлы (красные)

**git stash -u**
скрыть файлы в стэш, включая те, которые не находятся в индексации

**git stash show stash@{0}**
показать файлы в последнем стеше

## 1.11 Git merge\mergetool

**git merge <название_ветки>**
слить текущую ветку с <название_ветки>

**git merge --abort**
отменить слияние (например при конфликте)

**git merge --no-commit --squash featureB**
--squash - сливает изменения в один коммит, не являющийся коммитом слияния
--no-commit - отменяет автоматическую запись коммита

**git mergetool**
открыть редактор по умолчанию для разрешения конфликта

**git mergetool --tool=<название_утилиты>**
открыть редактор <редактор> для разрешения конфлита

## 1.12 Git rm\mv

**git rm <файл>**
удаляет файл и вносит в индексацию ‘удаление‘ этого файла

**git rm --cached <файл>**
удаляет из индекса, но физически не удаляет его

**git mv <изначальный_файл> <новое_название_файла>**
переименование с индексацией

## 1.13 Git checkout

**git checkout <имя_ветки>**
переключится на ветку

**git checkout -b <имя_новой_ветки>**
создать ветку и переключится на нее

**git checkout -- <файл>**
Вернуть <файл> к дефолту. (Убрать внесенные изменения)

## 1.14 Git others

**git push origin --delete <имя_ветки>**
Удалить ветку из удаленного репозитория

**git commit -a**
пропуск индексирования и сразу коммит

**git commit --amend**
слияние текущего коммита с предыдущим

**git reset HEAD <имя_файла>**
убрать из индексирования файл

**git fetch <локальное_название_источника>**
скачать без слияния

**git pull <локальное_название_источника>**
скачать со слиянием

**git diff master**
смотрим изменения

**git push origin --delete <имя_ветки>**
удалить ветку с удаленного репозитория

**git rebase <имя_ветки>**
тоже самое что и git merge, но только история коммитов отображается по другому

**git request-pull origin/master myfork**
запрос на включение коммитов в удаленный реп "origin/master"с ветки "myfork"

**git revert -m 1 HEAD**
отменить последний коммит с фиксацией в истории

**git blame -L 12,22 simplegit.rb**
посмотреть кто сделал изменения в файле "simplegit.rb"с 12 строки до 22 строки

## 1.15 Git ignore

Пример файла gitignore

---

# - комментарий
.a - пропускать файлы, заканчивающиеся на ".a"
!lib.a # - но отслеживать файлы "lib.a несмотря на пропуск файлов на ".a"
/TODO - игнорировать только корневой файл "TODO а не файлы вида "subdir/TODO"
build/ - игнорировать все файлы в папке "build/"
doc/*.txt - игнорировать "doc/notes.txt но не "doc/server/arch.txt"

---

https://github.com/github/gitignore - здесь можно посмотреть различные варианты настройки файла .gitignore

# 2 PostgreSQL

Установка клиента и службы:

**sudo apt-get install postgresql postgresql-contrib**

Авторизоваться под пользователем postgres:

**sudo -i -u <postgres>**

Войти в клиент-консоль postgres:

**sudo -i -u <postgres> psql**

Создание пользователя интерактивно

**createuser --interactive**

Создание БД

**createdb <db_name>**

Меняем пароль авторизации в клиент-консоли:

**ALTER USER <mrzlanx532> with PASSWORD 'password';**

Установка Postgres-PDO для PHP7.*

**sudo apt-get install 7.*-pgsql**

**Laravel .env**

*DB_CONNECTION=pgsql*
*DB_HOST=localhost*
*DB_PORT=5432*
*DB_DATABASE=db*
*DB_USERNAME=username*
*DB_PASSWORD=password*

# 3 Regular Expresions (RegExp)

**Паттерн | Пример**

^ = не
duck = "duck"
p|tyre = "p"или "tyre"
(p|t)yre = "pyre"или "tyre"
[dtl]uck = "duck "tuck "luck"

[^t]uck = все буквы вначале, кроме t. "tuck"нельзя, "buck"можно
[a-z] = диапазон от a до z
[A-Z] = диапазон от A до Z
[0-9] = диапазон от 0 до 9
[0-9]+ = бесконечное количество 8784655351518484213548
[0-9]{11} = 11 символов 89645327312
[0-9]{5,7} = 5-7 символов 89645327312
[0-9]{5, } = минимум 5 символов 89645327312

**Мета символы**

\d = [0-9]
\w = [a-zA-Z0-9_]
\s = пробелы, табы, отступы
\t = любой символ табуляции

*Пример*

\d{3}\s\w{5} = 123 hello

**Особые символы**
+ - совпадение встречается один или более раз
? - символ либо есть, либо нет
. - абсолютно любой символ
- совпадение встречается ноль или более раз

*Пример*

hello? = hell, hello

**Начало и конец шаблона**

^ [a-z]{5} = davidgtropgrpk (подойдет только david без продолжения) ^ это начало
[a-z]{5}$ = fefefewwwww (подойдет только wwwww без начала) $ это конец
^[a-z]{5}$ = david (нельзя ни в начале ничего добавить, ни в конце)

# 4 Python

## 4.1 Установка virtualenv и зависимостей Python

1. **py -m pip install virtualenv** - установка **virtualenv** в **pip-packages**
2. **mkdir python_env && cd python_env** - создаем папку под окружения python и переходим в нее
3. **virtualenv <название_проекта>** (e.g. "virtualenv django_proj") - создаем переменную окружения под название "django_proj"
4. **source <название_проекта>/bin/activate** - заходим включаем окружение

5. **pip install <название_модуля>** (e.g. **"pip install Django"**) - устанавливаем модули в окружение [OPTIONAL]
6. **pip freeze --local > req.txt** - сохраняем зависимости локального окружения в файл
7. **deactivate** - выйти из окружения

## 4.2 Деплой проекта питона с зависимостями из файла используя virtualenv

1. **virtualenv -p /path_to_python_exe <название_переменной_окружения>** (e.g **virtualenv -p /usr/bin/python3.8 py3.8-django**) - создаем переменную окружения из зависимостей, используя версию питона
2. **source py3.8-django/bin/activate** - заходим в п.о.
3. **pip install -r req.txt** - ставим зависимости в переменную окружения

## 4.3 Установка PyQt5-designer на Linux

1. sudo apt-get install python3-pyqt5
2. sudo apt-get install python3-pyqt5.qtsql
3. sudo apt-get install qttools5-dev-tools

## 4.4 Запуск PyQt5-designer как приложение

1. Создайте командой файл на рабочем столе **'touch "qt-designer.desktop"**
2. Откройте файл в редакторе и внесите следующее содержимое:

*#!/usr/bin/env xdg-open*
*[Desktop Entry]*
*Name=Qt5 Designer*
*Icon=/usr/share/applications/qt5-designer_.png*
*Exec=/usr/lib/x86_64-linux-gnu/qt5/bin/designer*
*Type=Application*
*Categories=Application*
*Terminal=false*
*StartupNotify=true*
*Actions=NewWindow*
*Name[en_US]=Qt5 Designer*

*[Desktop Action NewWindow]*
*Name=Open a New Window*
*Exec=/usr/lib/x86_64-linux-gnu/qt5/bin/designer*

Если необходима иконка приложения, добавьте созданную иконку в "/usr/share/applications"