## Источники

<https://git-scm.com/book/ru/v2>

<https://htmlacademy.ru/blog/187-git-console>

[правка коммитов](https://www.calculate-linux.org/main/ru/recommit)

[Перенос SVN-репозитория в git](http://leonid.shevtsov.me/22-07-2009/perenos-svn-repozitariya-v-git/)

[Учебник-введение в git](http://freesource.info/wiki/RuslanHihin/GitTutorial1?v=3pr)

[Руководство пользователя GIT](http://freesource.info/wiki/RuslanHihin/GitUserManual?v=3fj)

[20 повседневных команд git](http://freesource.info/wiki/RuslanHihin/20povsedevnyxkomandgit)

[Введение в структуру хранилища git](http://blog.tarantsov.com/2008/11/essential-git.html)

[Практическое введение в git](http://admdev.blogspot.com/2009/02/git.html)

[Работа с git для начинающих](http://blog.nsws.ru/rabota-s-git-dlya-nachinayushhix.html)

[Переходим с SVN на Git](http://git.or.cz/course/svn.html)

[ХХ полезных советов для пользователей Git среднего уровня. Часть 1](http://habrahabr.ru/blogs/Git/75728/)

[ХХ полезных советов для пользователей Git среднего уровня. Часть 2](http://habrahabr.ru/blogs/Git/76084/)

[Внешние зависимости в гите: submodule или subtree?](http://habrahabr.ru/blogs/Git/75964/)

[Командная работа в Git](http://habrahabr.ru/blogs/Git/75990/)

## init

Создать репозитарий в локальной директории с нуля

git init

запускается с папки проекта

после создания выдаст

Initialized empty Git repository in E:/OpenServer/domains/test-git/.git/

test-git – рабочая папка

## сlone - клонирование проекта на комп с репозитария

git clone <https://vladKramatorsk@bitbucket.org/vladKramatorsk/flat.git>

будет выполнено клонирование в папку flat

если папки нет, она будет создана

[git@bitbucket.org:vladKramatorsk/flat.git](mailto:git@bitbucket.org:vladKramatorsk/flat.git) - эту ссылку выдает сервер хранения

### Клонирование в произвольную папку

git clone git://github.com/schacon/grit.git mygrit

будет выполнено клонирование в mygrit

### Клонировать в текущую папку

Надо в конце поставить точку

git clone git://github.com/schacon/grit.git .

## remote - настроить удаленный сервер

### git Получить список удаленных серверов

git remote

### Получить список удаленных серверов с урлами

git remote –v

origin git://github.com/schacon/ticgit.git (fetch)

origin git://github.com/schacon/ticgit.git (push)

### [Инспекция удалённого репозитория](https://git-scm.com/book/ru/v1/Основы-Git-Работа-с-удалёнными-репозиториями" \l "Инспекция-удалённого-репозитория)

git remote show [удал. сервер]

### Привязка к новому

git remote add [сокращение] [url]

git remote add origin <https://vladKramatorsk@bitbucket.org/vladKramatorsk/testgit.git>

origin – название удаленного сервера

git commit -m "first commit"

git remote add origin https://github.com/VladSholeninov/test-git.git

git push -u origin master

привязать хранилище на GitHub

git remote add origin https://github.com/VladSholeninov/test-git.git

привязать хранилище на bitbucket

git remote add origin <https://vladKramatorsk@bitbucket.org/vladKramatorsk/testgit.git>

git push -u origin master

### [Переименование](https://git-scm.com/book/ru/v1/Основы-Git-Работа-с-удалёнными-репозиториями" \l "Удаление-и-переименование-удалённых-репозитор%D0%25B) удалённых репозиториев

git remote rename pb paul

pb – старое название

paul – новое название

### Удалить привязку

git remote rm paul

paul – имя репозитария в гите

## Зарегистрировать пользователя в git

git config --global user.name "Your Name"

git config --global user.name "VladSholeninov"

git config --global user.email "your\_email@whatever.com"

## status - получить статус изменения

git status

### Статус в сокращенном виде

git status –s

git status –short

## add - добавить файл в список отслеживаемых

git add README

README – файл, который надо добавить для остлеживание

### Добавить все новые файлы, которые надо отслеживать

git add .

точка в конце означает взять все файлы

### Добавить пустую папку

Чтобы сохранить пустую директорию, создайте в ней пустой файл .gitkeep и добавьте его в git.

## Rm – удалить из отслеживаемых

Удаляет физически с диска

Можно удалять несколько файлов

**git rm FILE1 FILE2**

### Удалить по маске

**git rm Documentation/\\*.txt**

Удалить из отслеживаемых, не физически. Т.е. если надо удалить и не комитить этот файл

**git rm --cached \_ide\_helper.php**

## diff - отличия между деревьями проекта, коммитами

**git diff**

Показать изменения, не внесенные в индекс

### Показать отличия между ветвями

git diff branch1 branch2

## commit

### Закомититься

git commit

автоматически добавить файлы “-a”

git commit –a –m ‘Текст изменения’

текст обязательно иначе вываливает внутренний редактор

#### ****git commit --amend****

Добавить изменения к последнему комиту

### Внешний редактор

git config --global core.editor notepad

git config --global core.editor "'D:\program\Notepad++\notepad++.exe' -multiInst -notabbar -nosession -noPlugin"

### Изменить комментарий к комиту

#### Изменить последний комментарий в ветке

git commit --amend

Появится редактор в котором можно вносить изменения

#### Изменить комментарий к произвольному комиту

Шаги

1. git rebase --interactive c4d25d6^

c4d25d6 – имя коммита который надо изменить

2. Меняем слово pick на слово reword,

reword c4d25d6 order status

Сохраняем и после сохранения появится окно в котором меняем комментарий

### Слияние комитов

#### 1. Узнать сколько комитов было относительно ветки

Детальный список

git cherry -v dev

dev – ветка относительно которой сравниванием

+ cf08390415beeb03160e6b36b1f760755a4aea50 BT-1404: refactored Danial Notification

+ bea9086a64b6f052b578a119e205dd32cace2131 BT-1404: refactored Goal Add New Measure

+ de8e029769c27880d45fcb33cff40050186741b4 BT-1404: refactored setup Assessment add new measure

количество комитов относительно заданной

git cherry -v dev | wc -l

## push - отправить на сервер хранения изменения

git push [удал. сервер] [ветка]

git push

по умолчанию берет origin и активную ветку

### Отправить на определенный сервер и определенную ветку

git push origin master

Более предпочтительный вариант так как определенно указывает место хранения

git push -u origin master

origin – название хранилища, указанного в remote

master – название ветки, которую отправляем на сервер

### Отправить из одной ветки в другую ветку сервера

Отправить изменения из ветки master в ветку experimental удаленного репозитория:

git push ssh://yourserver.com/~you/proj.git master:experimental

### Отправить на сервер метки

git push origin master —tags

**Залить локальную ветку на сервер, чтобы локальная была предпочтительна**

git push --force origin

## stash – сохранить рабочее состояние в стек гита

кладёт все не закоммиченные изменения в специальный стек Git.

git stash

### stash list – просмотреть состояние стека

git stash list

### apply

git stash apply

применить последнюю сохраненную запись из стека

Если надо применить определенную, то надо указать ее имя

git stash apply stash@{2}

### git stash save name

Сохранить stash с заданым именем

### drop

Удаляет stash по имени

git stash drop stash@{0}

### pop

Забрать изменения с автоматическим удалением из стека

git stash pop

равносильно выполнению двум командам

git stash apply + git stash drop

применить все изменения назад.

## branch

создать отдельную ветку с тем что находится в заначке

git stash branch testchanges

testchanges – имя ветки

### branch - ветвление

По умолчанию создается ветка master

### Создать ветку

git branch testing

testing – имя новой ветки

### Просмотреть список ветвей

#### Существующих

**git branch**

#### всех даже удаленных

**git branch -a**

*ddd-currency*

*ddd-district*

*\* master*

*remotes/origin/ddd-city*

*remotes/origin/ddd-currency*

*remotes/origin/ddd-region*

*remotes/origin/ddd-roles*

*remotes/origin/master*

### Просмотреть текущую ветку

git checkout

### Смена ветки

git checkout develop

develop – название ветки

### Смена ветки с созданием новой

git checkout -b develop

-b это сокращение branch

develop – название ветки

должно выдать

Switched to a new branch 'develop'

-b говорит, что надо сперва создать ветку а потом checkout переключиться на нее

Если ветка уже существует то опцию –b писать не надо

Важно! Изменение состояния папок и файлов происходит только после того как закомитилась ветка.

### Переименовать ветку

Переименовать текущую ветку в новую

git branch -m new-name-branch

переименовать произвольную существующую ветку в новую

git branch -m current\_name new\_name

### Показать ветки содержащие информацию

--contains

Показать ветки содержащие версии

git branch --contains v1.2

### Удалить ветку

#### Удалить на локальном

git branch -d iss53

в нижнем регистре –d имеет меньший приоритет чем –D

т.е. есть две команды с –d и –D

#### Удалить на удаленном сервере

git push origin :experimental

experimental – имя ветки которую надо удалить

## checkout

### Восстановление файла

git checkout name\_file – восстанавливает с репозитария определенный файл. При этом необходимо находится в той папке, где был удален файл

например

git checkout RolesModel.php

### Восстановление содержимое папки

Если еще не были закомичены изменения то

git checkout -- path/to/folder

folder –папка содержимое которое надо восстановить

Если были закомичены изменения надо сперва откатится

git reset -- path/to/folder

git checkout -- path/to/folder

### Игнорировать файл

**Вариант 1**

git checkout -- composer.lock

composer.lock – название файла который надо игнорировать

если файла не существует то произойдет восстановление файла

Если продолжает выдавать то вариант 2

Вариант 2

Записать в файл

.git/info/exclude

Пример

backend/services/LogClass.php

## merge - слияние веток

1. надо переключиться на ветку, с которой будет слияние

2. надо произвести слияние с указанием ветки, которую будем сливать

$ git checkout master

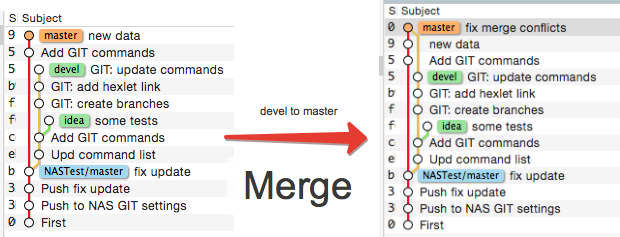
$ git merge hotfix

master – ветка, с которой сливаем и она остается

hotfix – ветка которую сливаем.

*Пояснения*

git **merge** devel - все коммиты ветки devel крепятся в конец, образуется пересечение (devel остается отдельной веткой, к которой можно вернуться)



#### ****--no-ff****

Добавление параметра --no-ff позволяет оставить вид древовидной структуры

**git merge --no-ff some-branch-name**

#### --no-merged

бранчи, изменения в которых еще не перенесены в текущий бранч

**git branch --no-merged**

#### ***-- squash***

Перед слиянием объединяет все комиты ветки в один. Описание возмет с последнего комита

**git branch --squash**

#### -m

Добавить комментарий к мержу

**git branch -m Коммент**

## fetch - забрать изменения с репозитария

### git fetch

git fetch получает изменения с сервера и сохраняет их в каталог refs/remotes/. Это никак не влияет на локальные ветки и текущие изменения. А git merge уже вливает все эти изменения в локальную копию.

#### Забрать из именованного репозитария

**git fetch origin**

origin – именованный репозитарий

#### .gitkeep

Размещение этого пустого файла позволяет закомитить пустую папку без ее содержимого.

(на самом деле можно и *любое другое имя файла*, просто это уже устоялось и другие программисты поймут, что мы не случайно забыли данный файл в директории, а именно постарались сохранить её для будущих поколений)

## pull - Забрать из именованного репозитрия

git pull — это по сути команда git fetch, после которой сразу же следует git merge.

**git pull origin master**

origin master – имя репозитария

origin – название сервера откуда скачиваем

master – название ветки

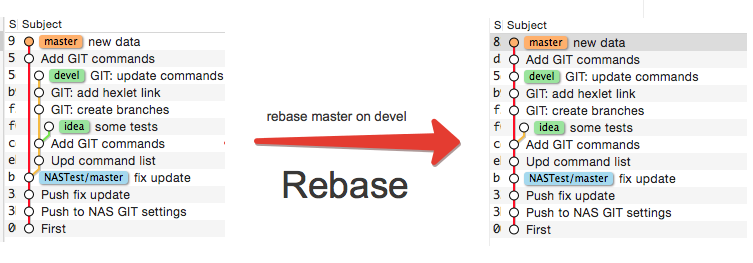
#### ****git pull --rebase****

Git сначала заберёт изменения из удалённой копии ветки, а потом накатит ваши незапушенные коммиты после самого последнего коммита удалённой ветки, так, как будто бы всё так и было. Таким образом необходимость мёржа отпадает и мы избежим уродливого коммита о мёрже в истории изменения.

**git pull --rebase origin master`**, аналог 3х команд (`git checkout master; git pull origin master; git checkout mybrach; git rebase master`)

git **rebase** devel - "сшивает" коммиты по дате их создания.

Важно! Отдельной ветки с которой работали больше нет.

Что выбрать?

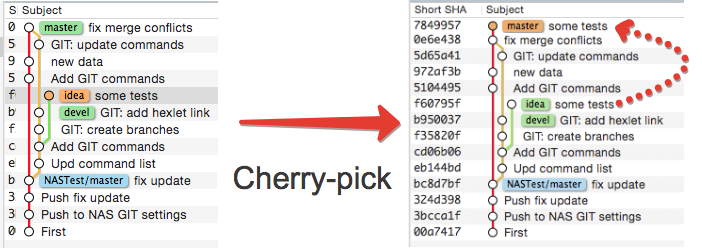
* Если вы сомневаетесь, то используйте обычное слияние.
* Выбор между merge и rebase обусловлен тем, какой вы хотите видеть историю коммитов: линейной или ветвящейся.

Учитывайте следующие факторы:

1. Если ветка, в которую вы хотите внести изменения доступна для других разработчиков (например, в open source проекте), не используйте rebase. Эта команда удаляет ветку целиком и приводит к рассинхронизации копий репозиториев.
2. Представляет ли исходная ветка ценность? Некоторые команды работают по принципу «одна функция – одна ветка», при этом ветка идентифицирует последовательность коммитов. В модели «один разработчик – одна ветка» в этом нет особой необходимости, так как автор коммита известен.
3. Не захотите ли вы вдруг отменить слияние? Возврат rebase значительно затруднен по сравнению с обычным слиянием, а иногда даже невозможен.

## **сherry-pick - забрать коммиты из ветки**

git **chery-pick** idea



## reset - сделать откат

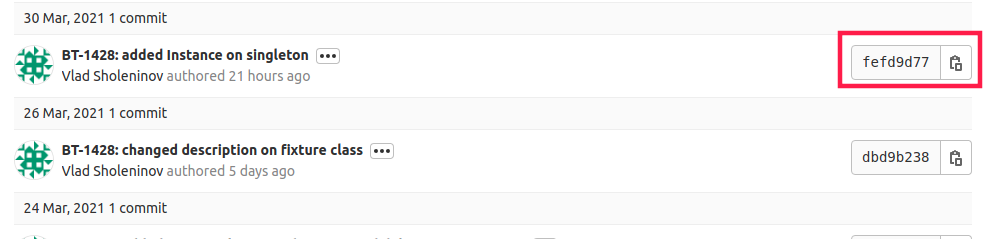
### -- hard

**git reset --hard {{some-commit-hash}}** — вернуться на определённый коммит в истории. Все изменения, сделанные после этого коммита пропадут.

Если я хочу отменить все внесённые изменения и начать работу с чистого листа, я использую команду **git reset --hard HEAD** (самый частый случай).

- git reset --hard fefd9d77e6c8891402848a2a8d78f26422cb32f6

Номер можно взять с гитлаба



### Без параметра

**git reset {{some-commit-hash}}** — вернуться на определённый коммит в истории. Все изменения, сделанные после этого коммита, получат состояние «Not staged for commit». Чтобы вернуть их обратно, нужно использовать команды **git add** и **git commit**.

### --soft

**git reset --soft {{some-commit-hash}}** — вернуться на определённый коммит в истории. Все изменения, сделанные после этого коммита, получат состояние «Staged for commit». Чтобы вернуть их обратно, нужно использовать команду **git commit**.

Сперва надо выбрать ветку, потом

git reset --hard HEAD~2

в примере на 2 откат

### filename

Удаляет файл из индекса

Команда git reset <path> противоположна git add <path>

## revert - мягкая отмена коммитов

git revert -n

### Отменить коммит, помеченный тегом

git revert config-modify-tag

### Отменить коммит по хешу

a42a3b96e8

git revert cgsjd2h

## show - показать информацию по комиту

#### git show

Просмотреть информацию по последнему комиту

#### git show a23232

показать информацию по определенному комиту

git show a23232

a23232 – начало хеша комита, количество символов может быть любым

## log - логирование

### Просмотреть весь лог

git log

Если лог длинный, то нажать “q” для выхода

**--author="Alex Kras"** — выводит коммиты, сделанные конкретным человеком

**--name-only** — выводит только названия изменённых файлов

**--oneline** — выводит сокращённые данные коммита (в виде одной строки)

**--graph** — выводит дерево зависимостей для всех коммитов

**--reverse** — выводит коммиты в обратном хронологическом порядке (сначала старые)

**--after** — выводит коммиты, сделанные после определённой даты

**--before** — выводит коммиты, сделанные до определённой даты

**--stat** - Статистика изменения файлов, вроде числа измененных файлов, внесенных в них  
строк

**--summary** - информацию по созданиям, переименованиям и правам доступа файлов

### Отправить лог в файл

git log > log-git.txt

### Построить граф логов

–graph

git log --oneline –graph

### Просмотреть логи по автору

--author

git log --author="vladSholeninov"

### Просмотреть историю изменений определенного файла

git log README

### - - pretty

Вывести на каждый из коммитов по строчке, состоящей из хэша (здесь — уникального идентификатора каждого коммита, подробней — дальше):

git log --pretty=oneline

Лаконичная информация о коммитах, приводятся только автор и комментарий:

git log --pretty=short

Более полная информация о коммитах, с именем автора, комментарием, датой создания и внесения коммита:

git log --pretty=full/fuller

В принципе, формат вывода можно определить самостоятельно:

git log --pretty=format:'FORMAT'

### лог изменения файла

git log templates/header.tmpl

## config - настройки Git

### Просмотреть все настройки

git config –-list

глобальные

git config --global --list

### Установить имя

git config --global user.name "Vladislav Sholeninov"

VladSholeninov – имя

### Установить email

git config --global user.email [sholeninov@mail.ru](mailto:sholeninov@mail.ru)

[sholeninov@mail.ru](mailto:sholeninov@mail.ru) – емайл

### Просмотреть значение определенной переменной

git config user.name

user.name – имя переменной

### Установить настройки конфига git

#### Установить/сбросить замену переноса строчек

git config core.autocrlf true

### Посмотреть версию git

git –version

git version 1.9.4.msysgit.2

### Игнорировать изменения permition для файлов

git config --global core.fileMode false

## tag - метки

### Просмотреть список

git tag

### Добавить метку

git tag -a v1.4 -m 'my version 1.4'

-a – означает что аннотированную

v1.4 – любой текст который означает версию

-m означает что будет комментарий к версии

### Отправить метки на сервер

Надо добавить опцию -–tags

git push origin –tags

## alias - Алиасы

git config --global alias.l "log --oneline --graph"

l – имя алиаса

log --oneline –graph что должен выполнять новый алиас

к алиасам Git можно так же добавлять дополнительные параметры (например, **git l --author="Alex"**).

## grep – поиск слова в проекте

### Поиск слова tst в проекте

git grep tst

### Подсчитать число упоминаний

git grep -с tst

### Поиск в версии проекта по tag

git grep tst v1

Команда позволяет использовать логическое И и ИЛИ.

Найти строки, где упоминаются и первое слово, и второе:

git grep -e 'first' --and -e 'another'

Найти строки, где встречается хотя бы одно из слов:

git grep --all-match -e 'first' -e 'second'

## Относительная адресация

обратиться прямо к предку последнего коммита ветки master

git diff master^

Если после «птички» поставить цифру, то можно адресоваться по нескольким предкам коммитов слияния

.gitignore

#комментарий к файлу .gitignore

#игнорируем сам .gitignore

.gitignore

#все html-файлы...

\*.html

#...кроме определенного

!special.html

#не нужны объектники и архивы

\*.[ao]

# Модули гит

Если папка с отдельным гит репозитарием (модулем) лежит в вложенной папке, например, /nova то можно установить все сразу указав что есть второй вложенный репозитарий

1. создать в корне файл .gitmodules

вписать в нем

[submodule "nova"]

path = nova

url = <https://gitlab.quartsoft.com/nekrasov/krampost-nova.git>

2. добавить опцию --recurse-submodules при слонировании базового репозитария

git clone --recurse-submodules https://gitlab.quartsoft.com/php5/krampost.git

# SSh

### Сгенерировать ключ

ssh-keygen -t rsa -b 4096 -C [sholeninov@mail.ru](mailto:sholeninov@mail.ru)

## Часто используемые команды

### Исправление конфликтов

Если возникла ошибка, и ее исправили надо

git add .

git commit –m “fixed error”

git merge --no-ff master