Git - справка

Составлено по материалам курса Ильи Кантора от JavaScript.ru:
https://www.youtube.com/watch?v=W4hoc24K93E

Содержание

1 Введение

1.1 Настройки

Сохранить локальные настройки:

```
$git config user.name <user_name>
```

```
   $git config user.email <user_email>
```

Варианты настроек:

```
   $git config --system ...    # Системные: /etc/git/config
   $git config --global ...    # Пользователя:  /.gitconfig
   $git config --local ...    # Проекта (default): <project>/.git/config
```

```
$git config --list    # - все конфигурационные параметры
$git config --unset user.name    # - сбросить параметр
$git config --remove-section user    # - удалить секцию
$git config --global core.editor ...    # редактор для commit
```

Алиасы, пример:

```
   $git config --global alias.c 'config --global'
   $git c --list
```

Добавить в список игнорируемых файлов:

```
   $git config --global core.excludesFile '~/.gitignore'
```

Сохранять учетные данные при регистрации на удаленных сервисах:

```
   $git config --global credential.helper store
```

1.2 Простые команды

```
$git init    # - инициализация каталога проекта
$git status    # - текущий статус
$git add <file>    # - добавить файл в индекс
```

Сделать commit:

```
   $git commit    # - запустится редактор для ввода комментария
   $git commit -m <comment>
$git add -- chmod=+x <file>    # - сделать файл исполняемым
```

Специальные файлы:

```
   .gitignore    - содержит список игнорируемых файлов
   .gitkeep    - в пустых каталогах, тогда они добавляются в индекс
```

```
$git reset HEAD <file>    # - убрать изменения в файле
$git add -f <file>    # - добавить файл из игнорируемого каталога
```

```
   $git add -p <file>    # - с вопросом по каждому изменению
```

Добавление всех изменений сразу в commit:

```
$git commit -all
$git commit -am <comment>
$git commit -m <comment> file1 file2 ..."# - несколько файлов
```

Удалить файл:

```
$git rm <file>    # - удаление из каталога и индекса
$git rm -r <file>    # - удаление из каталога
$git rm --cached <file>    # - удаление из индекса
$git rm -f <file>    # - удаление с несохраненными изменениями
```

Переименовать файл:

```
$git mv <old_name> <new_name>
```

2 Удаление изменений

```
$git checkout -f    # - удалить все изменения
$git checkout -f file    # - удалить изменения в файле
$git reset -hard    # - сброс изменений
$git clean -dxf    # - очистка от изменений
```

-d - директории

-x - файлы которые не отслеживаются

3 Ветки

Работа с ветками:

```
$git branch -v    # - список веток с краткой информацией
$git branch <name>    # - создать новую ветку
$git checkout <name>    # - переключиться на ветку
$git checkout -b <name>    # = create + checkout
```

Изменения в ветках:

```
$git stash    # - убрать все изменения и запомнить
$git stash pop    # - вернуть запомненные изменения
$git stash -m <name>    # - задать имя для сохраняемых изменений
```

Сделать новую ветку указывающую на commit:

```
$git branch <name> <commit>
```
Передвинуть ветку на commit (или другую ветку):
```
$git branch -f <name> <commit|branch>
$git checkout -B <name> <commit>    # = git branch -f + git checkout
```
Передвинуться на commit:
```
$git checkout <commit>
```

Скопировать commit на текущую ветку:
```
$git cherry-pick <commit>
```

Восстановить версии файлов из commit:
```
$git checkout <commit> <filenames>
```
Сбросить индекс для файла:
```
$git reset <file>
```
Вернуть файл из repository:
```
$git checkout HEAD <file>
```
Вернуть файл из индекса:
```
$git checkout <file>
```

Просмотр истории:
```
$git log --oneline
$git log <branch> --oneline
$git show <commit>
$git show HEAD~   # - родитель commit
   HEAD~~~ = HEAD~3
   HEAD = @
```

Показать содержимое файла из commit:
```
$git show <commit>:<file>
```
Найти commit по строке:
```
$git show <commit>:/<string>
```

Выполнить merge master и branch, используется Fast-Forward:
```
$git merge <branch>
```

Продвинуть ветку:
```
$git branch -f <branch> <commit>    # - на commit
$git branch -f <branch> ORIG_HEAD    # - на предыдущий HEAD
```

Удаление ветки:
```
$git branch -d <branch>    # -только если объединение с текущей
```

```
$git branch -D <branch>     # - если в ней есть отличные commit
```

Восстановить удаленную ветку:

```
$git branch <branch> <commit>
```

4 Reference logs

```
$cat .git/logs/HEAD
$git reflog
$git reflog <branch>
$git reflog    алиас для    $git log --oneline -g
$git reflog --date=iso
```

5 Reset

Сброс (по умолчанию mix):

```
$git reset --hard <commit>     # - Жесткий сброс
$git reset --mix <commit>     # - Смешанный сброс
$git reset --soft <commit>     # - Мягкий сброс
   @     - последний commit
   @~    - предыдущий commit
```

Отличия видов reset (из `$git reset --help`):

working	index	HEAD	target		working	index	HEAD
A	B	C	D	--soft	A	B	D
				--mixed	A	D	D
				--hard	D	D	D

```
$git reflog master
```

Взять комментарий из другого commit:

```
$git commit -c <commit>     # - откроется редактор
$git commit -C <commit>     # - не будет открываться редактор
    --reset-author      - сбросить автора (по умолчанию из commit)
```

```
$git commit --amend     # - исправить в последнем commit
    --no-edit    # - не открывать редактор
```

```
--reset-author
```

Очистка индекса от всех изменений:
```
$git reset HEAD
```
Удалить изменения по файлу в индексе
```
$git reset <file>
```
Поместить файл из commit в индекс (в директории не меняется):
```
$git reset <commit> <file>
```

6 Просмотр изменений

```
$git diff <commit1> <commit2>
$git diff <commit1>..<commit2>
```

Что изменилось в branch2 с момента отделение от ветки branch1:
```
$git diff <branch1>...<branch2>
```
Изменения в директории с момента последнего commit:
```
$git diff HEAD
```
`$git diff` - Изменения по сравнению с индексом
`$git diff --cached` - Изменения индекса с последним commit

При commit в редакторе показывать изменения:
```
$git commit -v
```
Включить автоматическое отображение изменений при commit:
```
$git config --global commit.verbose true
```

`$git diff <file>` - Изменения в файле
Только имена изменившихся файлов:
```
$git diff --name-only <commit1> <commit2>
```
Сравнить разные файлы из разных commit:
```
$git diff <commit1>:<file1> <commit2>:<file2>
```
Сравнить 2 произвольных файла без привязки к git:
```
$git diff --no-index <file1> <file2>
```

7 Просмотр истории

```
$git log    # - Вывод истории commit достижимых из HEAD
$git log --name-only    # - Выводить имена файлов
$git log -n <count>    # - Вывести count последних commit
$git log --pretty=<value>
   medium (default)
   oneline
   format:'%h %cd | %s%d [%an]'  - id, date | comment ссылки автор
      %C(<colorname)    - задать цвет

   --abbrev-commit    # - обрезать id commit
```

Пример вывода лога в определенном формате:

```
   $git log --reverse --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset
%s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit
```

Можно задать в конфигурации:

```
   $git config format.pretty '%Cred%h%Creset -%C(yellow)%d%Creset
%s %Cgreen(%cr) %C(bold blue)<%an>%Creset'
```

```
$git log --oneline = $git log --pretty=oneline --abbrev-commit
   --decorate-short
   --date=short    # - только даты
   --date=format:'%F %R'    # - берет флаги из strftime

$git log <branch>    # - log по commit достижимым из ветки
   --graph    # - в виде дерева
   --all    # - из всех commit

$git log gui
$git log feature ^master    # - log по feature кроме master
$git log master..feature
   --boundary    # - включая пограничный commit

$git log ..feature - симметрическая разница

$git log <file> - история по файлу
   -p - показывать различия
   --follow - если был переименован, то и по старым названиям
```

История между commit по списку файлов:

```
$git log <commit1>..<commit2> <file1> <file2> ...
```

Поиск по строке по commit достижимым из HEAD:

```
$git log --grep <string>
$git log --grep <str1> --grep <str2> # - str1 или str2 в сообщении
      --all-match     - по всем совпадение
      -p     - Perl type совместимые регулярные выражения
      -F     - отключить регулярные выражения
      -i     - регистро-независимый поиск
```

Поиск по содержимому:

```
$git log -G <string>
```

Все изменения со строки i1 по i2 в <file>:

```
$git log -L i1,i2:<file>
```

Регулярные выражения для начала и конца фрагмента:

```
$git log -L r1,r2:<file> - r1,r2
```

Регулярное выражение <func> для имени функции:

```
$git log :<func>:<file>
```

```
$git log:
   --author
   --commiter
   --before 'before 3 months'
   --after

$git blame <file>    # - кто менял строки файла
```

8 Слияние

```
$git merge <branch>    # - объединить ветки
```

При конфликте взять версию файла:

```
$git checkout --ours <file>    # - из текущей ветки
$git checkout --theirs <file>    # - из объединяемой ветки
```

```
$git checkout --merge
```

Вернуть в состоянии до merge:

```
$git reset --hard
```
Оставить которые хорошо объединились, проблемные вернуть в начальное состояние:
```
$git reset --merge
```

```
$git checkout --conflict-diff3 --merge <file>
```

После окончания разрешения конфликтов создает окончательный commit слияния:
```
$git merge --continue
```
id сливаемого commit в .git/MERGE_HEAD

Отмена слияния:
```
$git reset --hard @~
```

Слияние без автоматического commit:
```
$git merge <branch> --no-commit
```

Слияние без Fast Forward (перемотки), создается новый commit:
```
$git merge --no-ff --no-edit <commit>
```

Слияние без сохранения истории сливаемого commit:
```
$git merge --squash <commit>
```

9 Копирование commit

`$git cherry-pick <commit>` `#` - скопировать на текущую ветку
`-x` - добавить строку с информацией откуда скопирован commit

Скопировать все commit из <commit2> в <commit1> с момента разделения:
```
$git cherry-pick <commit1>..<commit2>
```

При возникновении конфликта:
```
$git cherry-pick --abort    # - отмена
$git cherry-pick --continue    # - продолжить
$git cherry-pick --quit    # - оставить только успешно скопированные
```

Скопировать без создания commit

```
$git cherry-pick --no-commit <commit>
```

10 Перенос веток

Чтобы текущая ветка начиналась с последнего commit в <commit>:

```
$git rebase <commit>
```

Полностью отказаться, если были неразрешенные конфликты:

```
$git rebase --abort
```

Остаться в текущем незавершенном состоянии:

```
$git rebase --quit
```

Пропустить все изменения из текущего проблемного commit и продолжить дальше:

```
$git rebase --skip
```

Продолжить после разрешение конфликта:

```
$git rebase --continue
```

Узнать id ветки до предыдущего состояния:

```
$cat .git/ORIG_HEAD
```

Просмотр изменений по ветке:

```
$git reflog <branch> -1
```

Просмотр информации по последнему commit:

```
$git show --quiet <branch>@{1}
```

Вернуться на состояние последнего commit:

```
$git reset --hard <commit>@{1}
```

```
$git rebase <commit1> <commit2> =
      $git checkout <commit2> + $git rebase <commit1>
```

Для каждого переносимого commit запустить команду (обычно тест), при ошибке произойдет откат изменения:

```
$git rebase -x <command> <branch>
```

После исправления ошибок:

```
$git add ...
$git commit --amend --no-edit
```

Перенос всех commit в ветку branch из текущей ветки начиная с <commit>:

```
$git rebase --onto <branch> <commit>
```

Скопировать 2 последних commit:
```
$git cherry-pick master 2..master
```
В master откатиться на commit назад:
```
$git branch -f master master 2
```

Если делали в ветке commit слияния, скопировать commit кроме commit слияния:
```
$git rebase --rebase-merges <commit>
```

Интерактивное преобразование commit начиная с <commit>:
```
$git rebase -i <commit>
```
Просмотр списка commit для преобразования:
```
$git rebase --edit-todo
```

`$git rebase --continue`
`$git rebase -i @ 3` - 3 commit назад
`$git rebase --no-ff` - (no Fast Forward) - копировать все commit, а не только с первого измененного

Внести исправления в commit:
`$git commit -a --fixup=<commit>` - можно @ - предыдущий
`$git rebase -i --autosquash` - сольет fixup по commit

11 Отмена commit

`$git revert <commit> #` - создает commit отменяющий изменения в <commit>
`$git revert <commit> -m 1 #` - отмена commit слияния

12 Примеры

12.1 Разбить существующий commit на несколько

Выполняем интерактивный rebase:
`$git rebase -i @~<n>` - ставим e у нужного commit
После этого отменяем регистрацию изменений всех файлов этого commit:
```
$git reset HEAD~
```
Для каждого нового commit:
```
$git add <file>
```

```
$git commit -m '...'
$git rebase --continue
```