

git cheat sheet

create & clone

создать репозиторий в текущей директории
создать локальную копию удалённого репозитория

```
git init [название проекта]  
git clone /path/to/repository
```

add & remove

добавить указанные файлы в индекс
индексировать все изменения в текущем каталоге

```
git add <filename>  
git add .  
-i - интерактивный режим  
git rm <filename>  
git rm -cached <filename>
```

удалить файл из рабочей директории
удалить файл из-под контроля версий
файл не удаляется с диска

```
git mv <Lastname> <newname>/<directory>  
git clean -fd  
f - force, d - для рекурсивного удаления из подкаталогов  
git status
```

переименовать/переместить файл
удалить неотслеживаемые (untracked) файлы из рабочей директории
посмотреть статус репозитория

commit & synchronize

коммит изменений
add + commit

не работает для добавления в индекс новых неотслеживаемых файлов

```
git commit -m "Commit message"  
git commit -a -m "Commit message"  
git commit <filename> -m "Commit message"  
git commit --amend -m "Commit message"
```

добавить изменения из индекса к последнему коммиту/исправить ошибку в описании
переключиться на указанный коммит

Где я? Git branch

Вернуться обратно: git switch -

git push -u origin master

origin - имя репозитория
master - ветка, которую мы загружаем
-u позволяет в последующем не указывать название ветки, а загружать ту ветку, на которую указывает HEAD
-force перезапишет ветку

git remote add origin <server_link>

git remote

git remote show [удалённый_репозиторий]

git push [удалённый_репозиторий] [ветка]

git pull

git pull --rebase = fetch + rebase

git fetch

добавить удалённый репозиторий
показать доступные удалённые репозитории
показать информацию об удалённом репозитории
 отправить изменения в удалённый репозиторий
загрузить изменения из удалённого репозитория
pull = fetch + merge
загрузить изменения из удалённого репозитория, безопасная синхронизация

branches

создать новую ветку
(новый указатель на текущий коммит)

git branch <branch>

--track - отслеживать ветку

git switch <branch>/git checkout <branch>

git branch

-a список удалённых веток

git checkout -b <branch>

git branch -d <branch>

git push origin <branch>

git rebase master

-i - интерактивный rebase

git branch -f <branch> HEAD~3

git checkout -b <branch> o/main

git branch -u o/main <branch>

создать ветку и переключиться на неё
удалить ветку
загрузить ветку в удаленный репозиторий
перебазировать ветку
изменения выбранной ветки применяются поверх целевой (master)
git копирует набор коммитов и переносит их в другое место
переместить ветку
создать ветку, которая будет следить за удаленной веткой o/main
указать ветке следить за o/main

merge

объединить указанную ветку с основной
отобразить неиндексированные изменения в рабочей директории
отобразить проиндексированные изменения
показать разницу между двумя ветками

`git merge <branch>`
`git diff`
для перемещения вниз нажать *f*, для перемещения наберх *b* или *i*. Для выхода из режима просмотра - *q*
`git diff --staged / git diff --cached`
`git diff <source branch> <target branch>`

tagging

создать новую метку
показать все теги
удалить тег
вывести историю коммитов

`git tag <tag> <commit_id>`
`git tag`
`git tag -d <tag>`
`git log`
--oneline одной строкой
--graph вывести график отображающий структуру ветвления
--all история коммитов по всем веткам
-r показать изменения коммитов
-n2 показать 2 последних коммита
`git log -grep message`
`git log -- <filename>`
`git show <commit_id>`
`git blame <filename>`
`git grep <text>`
-i - искать без учета регистра
`git describe <ref>`
ref - это что-либо, что указывает на конкретный коммит
Выход команды: `<tag>_<nCommits>_g<hash>`, где
tag - это ближайший тег в истории изменений,
nCommits - это на сколько далеко мы от этого тега,
a hash - это хеш коммита, который описывается.

stash

«припрятать» изменения в рабочей директории независимо от того, добавлены они в индекс или нет
восстановить спрятанные изменения работает по принципу стека
список стэшей
удалить хранилище (стэш)

`git stash`
`git stash pop`
`git stash list`
`git stash drop <stash_id>`

restore

вернуть файл в первоначальное состояние
отменить изменения неиндексированных файлов/**восстановить** удаленный файл
отменить индексирование изменений
отменить(откатить) коммит
отменить все коммиты после заданного только для локальных репозиториев!

`git checkout -- <filename>`
`git restore <filename>`

`git restore --staged <filename>`
`git revert 1b2e1d63ff`
`git reset <commit>`
--soft-изменения остаются в индексе и рабочей директории
--mixed-изменения остаются только в рабочей директории
--hard - полное удаление, включая рабочую директорию
^ - перемещение на 1 коммит назад, т.е. найти родителя
^2 - найти второго родителя
main^ - прародитель (родитель родителя) ветки main
HEAD указывает на последний коммит (текущий выбранный)
HEAD~ означает "один коммит от последнего"
HEAD~2 - "два последних коммита"

tips

встроенный в git графический интерфейс выводить в логе коммит на одной строке

добавить алиас

копировать коммиты в HEAD после HEAD справка

`gitk`
`git config format.pretty oneline`
`git log --pretty=format:'%h %ad | %s%d [%an]' --graph --date=short`
`git config --global alias.hist "Log --pretty=format:'%h %ad | %s%d [%an]' --graph --date=short"`
`git cherry-pick <Commit1> <Commit2> <...>`
`git help`