https://github.com/sajibcse68/MyDailyLearn/blob/git/git/git-cheatsheet.md#configure-users-name-and-email

PythonProjectDeveloper/rsschool-2019Q1-cv

GIT BUSH консоль

**mkdir** hello — создать папку hello

**mkdir -p** **{**app1,app2**}** — Создать сразу несколько папок

**mkdir -p** app**/{**css,js**}** — Создать сразу несколько вложенных папок

**cd** hello — перейти в папку hello

**cd ../**cloned\_hello — перейти в клонированную папку hello

**cd !$** — перейти в только что созданную папку

**touch** hello.html — создать файл hello.html

**touch app/{**css/main.css,js/main.js,index.html**}** — Создать сразу несколько файлов, никаких лишних пробелов быть не должно

**echo** " https://bol2n.github.io/rsschool-cv/cv " **>>** README.md — добавить текст в файл hello.html

**cat** hello.html — выводит текст содержимого

**mv** hello.html lib — переместить файл hello.html в новую директорию lib

**mv** app1**/\*.\*** app2 — Переместить все файлы из папки app1 в папку app2

**rm** test — удалить пустую папку test

**rm -r** test — удалить папку test с файлами внутри неё

**clear** — очистить консоль

**pwd** — показать текущий каталог

ls - Показать файлы в данной папке, кроме скрытых

ls -f — Показать файлы в данной папке, включая и скрытые

ls .git/refs —

ls .git/refs/heads —

ls .git/refs/tags —

cat .git/refs/tags/v1

cd c:/ — Перейти в конкретный каталог

cd - — Вернуться назад

cd .. — Выйти на 1 уровень вверх

cd ../.. — Выйти на 2 уровня вверх

Alias

**git config --global alias.**co checkout

**git config --global alias.**ci commit

**git config --global alias.**st status

**git config --global alias.**br branch

**git config --global alias.**pu push origin

**$ git log --pretty=format:"%h %ad | %s%d [%an]" --graph --date=short**

\* fa3c141 2011-03-09 | Added HTML header (HEAD, master)

**--pretty="..."** — определяет формат вывода.

%h — укороченный хэш коммита

%d — дополнения коммита («головы» веток или теги)

%ad — дата коммита

%s — комментарий

%an — имя автора

--graph — отображает дерево коммитов в виде ASCII-графика

--date=short — сохраняет формат даты коротким и симпатичным

git config --global --unset alias.st //удалить alias

git config --get-regexp alias // просмотр alias

git config --global alias.st status (или 'status') //вносит свой псевдоним

git help //показывает основные команды

git show //показывает последние изменения

**HEAD** - то место, где находится репозитория текущая

**master** — имя ветки по умолчанию

**git config --global alias.**hist"*log --pretty=format:'%h %ad | %s%d [%an]' --graph --date=short*"

***делаем alias*** *hist* ***построчно с дополнительной инфой***

**git hist** master **–all**

***показывает все теги в логе, даже удаленные***

**git hist --max-count=1**

***показывает последний коммит в репозиторий***

**git log --oneline**

***показывает коммиты в одну строчку***

**git cat-file –p**aa7e590

***вывод дерева, каталогов, файлов в*** *git* ***по хэшу SHA1***

**git tag -d** v1

***удаляет тег*** *v1* ***и коммиты, на которые он ссылался, сборщиком мусора***

**git tag**

***показывает какие теги доступны***

**git tag** v1

***назовем текущую версию страницы v1***

**git checkout** v1**^**

***переключиться на предыдущую версию, используя знак ^, обозначающий родитель v1***

**git init**

***если создали новый реп, которого нет на гитхабе, то нужно инициализировать***

**git mv** hello.html lib

***перемещая файлы с помощью*** *git****, мы информируем*** *git* ***что файл*** *hello.html* ***был удален, а***

***файл*** *lib/hello.html* ***был создан, при этом оба эти факта сразу же проиндексированы и готовы к коммиту.***

**git log**

***показывает все коммиты***

**git add** hello.html

***индексация изменений для последующего коммита***

*Изменения файла hello.html были проиндексированы. Это означает, что git теперь знает об изменении, но изменение пока не перманентно (читай, навсегда) записано в репозиторий. Следующий коммит будет включать в себя проиндексированные изменения. Если вы решили, что не хотите коммитить изменения, команда состояния напомнит вам о том, что с помощью команды git reset можно снять индексацию этих изменений.*

**git add . / git add --all**

***индексация всех изменений для последующего коммита***

**git add** lib/hello.html

**git rm** hello.html

***проиндексировать файл в новой директории*** *lib* ***и удалить из старой***

**git commit -m "**Changes for hello.html**"**

***коммитим изменения в репозиторий***

**git commit**

***если пропустить метку -m из командной строки,*** *git* ***перенесет вас в редактор***

**git checkout** aa7e590

***можно вернуться в состояние commit-а с хашем*** *aa7e590****, также и обратно, проверить через*** *git branch*

**git checkout** master

***переход к последней версии в ветке*** *master****, отменям изменения в рабочем каталоге***

**git checkout** hello.html

***переход в версию файла*** *hello.html* ***в репозитории***

**git reset HEAD** hello.html

***отмена проиндексированных изменений (перед коммитом)***

*Команда reset сбрасывает буферную зону к HEAD. Это очищает буферную зону от изменений, которые мы только что проиндексировали. Команда reset (по умолчанию) не изменяет рабочий каталог. Поэтому рабочий каталог все еще содержит внесенные изменения.*

**git reset --hard** v1 (ссылка на коммит, т.е. хэш, ветка или имя тега)

***переписывает текущую ветку, чтобы она указывала на нужный коммит***

*Параметр --hard указывает, что рабочий каталог должен быть обновлен в соответствии с новым head ветки.*

**git revert HEAD / git revert HEAD --no-edit**

***отмена коммита путем создания нового коммита, отменяющего нежелательные изменения.***

*Чтобы отменить коммит, нам необходимо сделать коммит, который удаляет изменения, сохраненные нежелательным коммитом, с открытием редактора или без него. Однако, и оригинальный и «отмененный» коммиты видны в истории ветки.*

**git commit --amend**

***позволяет изменить предыдущий коммит через редактор (после индексации)***

**git commit --amend –m** **"**Changes for hello.html**"**

***позволяет изменить предыдущий коммит, добавив новое сообщение в коммит (после индексации)***

**git commit --amend --no-edit**

***позволяет изменить предыдущий коммит (после индексации)***

**git checkout -b** style

***создание и переход на новую ветку style***

*Является шорткатом для git branch style за которым идет git checkout style.*

**git branch**

***выводит только список локальных веток по умолчанию***

**git branch –a**

***увидеть все ветки***

**git branch -d** style

***удаляем ветку*** *style* ***локально***

**git checkout** style

**git merge** master

***переходим к ветке*** *style* ***и сливаем*** *master* ***со*** *style*

*Забираем изменения из master и переносим в свою текущую ветку* **style***.*

**git checkout** style

**git rebase** master

***обновить*** *master* ***в новой ветке***

*Конечный результат перебазирования очень похож на результат слияния. Ветка style в настоящее время содержит все свои изменения, а также все изменения ветки master. Однако, дерево коммитов значительно отличается. Дерево коммитов ветки style было переписано таким образом, что ветка master является частью истории коммитов. Это делает цепь коммитов линейной и гораздо более читабельной. Не используйте перебазирование, если ветка является публичной и расшаренной. Переписывание общих веток будет мешать работе других членов команды. И когда важна точная история коммитов ветки (так как команда rebase переписывает историю коммитов. Чтобы ветки style и master стали идентичны, теперь давайте сольем изменения style в ветку master.*

**git clone** hello cloned\_hello

***клонировать репозиторий*** *hello*

*В рабочем каталоге теперь должно быть два репозитория: оригинальный репозиторий «hello» и клонированный репозиторий «cloned\_hello»*

**git clone** git@github.com:bol2n/rsschool-2019Q1-cv.git

***скопировать репозиторий из GitHub к себе на компьютер***

**git remote**

***узнать об удаленных репозиториях***

**git remote show** origin

***получить подробную информацию об удаленном репозитории***

**git fetch**

**git merge** origin**/**master

***извлечь локально удаленный репозиторий и слить извлеченные изменения в локальную ветку*** *master*

*Команда «git fetch» будет извлекать новые коммиты из удаленного репозитория, но не будет сливать их с вашими наработками в локальных ветках. Команда git pull эквивалентна комбинации git fetch и git merge.*

**git pull**

***положили изменения к себе в рабочий каталог (полностью забрали)***

**git branch --track** styleorigin**/** style

**git branch -a**

**git hist --max-count=2**

***добавить локальную ветку*** *style****, которая отслеживает удаленную ветку.***

**git push** origin master

***заливаем на сервак локальный репозиторий (после индексации)***

**git push** origin **--delete** style

***удаляем ветку на удаленном репозитории***

git reflog //показывает все изменения

git rebase //применить изменения у себя локально

git stash //??????? отменить изменения до коммита

git revert 70e12d880c4f84b768eea753cbc2d69afbd6cb8c //отменить commit с хашем 70e12d880c4f84b768eea753cbc2d69afbd6cb8c

git reset HEAD~2 //отменить изменения и вернуться на 2 commit-а назад

git reset --soft 45745cc //вернуться на состояние коммита 45745cc, не меняя историю коммитов и оставив все изменения

git reset --hard 45745cc //вернуться на состояние коммита 45745cc и изменить ветку

git cherry-pick ca49081 //применить commit с хашем ca49081 (из ветки master) к нашей ветке

git push --force //

rm -rf ~/.ssh /удаляет ключи из git bash

git checkout HEAD имя /Discards changes in the working directory.

git reset HEAD имя /Unstages file changes in the staging area.

git global --config --list /показывает аккаунт

git /показывает все команды

git rebase master /обновить мастер в новой ветке

git push /после merge, отправить изменения на удаленный сервер

git push --force /после rebase, изменение текущего состояния

git push -u origin

git remote add origin https://github.com/bol2n/имя репозитория.git

git push -u origin master пушим на сервак

git checkout staging /

git cherry-pick <coomit\_hash> /забрать изменения с другой ветки на мастер

git push origin master заливаем на сервак

echo "# rsschool-2019Q1-html-css" >> README.md

git init

git add README.md

git commit -m "first commit"

git remote add origin git@github.com:bol2n/test.git

git push -u origin master

git config --global core.editor \"'C:\Program Files\Notepad++\notepad++.exe' -multiInst -notabbar -nosession -noPlugin"

Игорь, вы смотрите старую инструкцию в интернете. Сейчас гитхаб значительно упростил создание gh-pages. Всё, что требуется для их создания - в любом репозитории создать ветку с именем gh-pages и поместить туда файлы. Если на верхнем уровне находится файл с расширением .md или .html страница станет доступна по адресу https://user.github.io/repository/ в котором user - ваш ник на гитхабе, repository - имя репозитория