Операторы командной строки

1. команда pwd (от англ. ***p***rint ***w***orking ***d***irectory — «показать рабочую папку»). Она выводит путь к текущей директории.
2. команда — ls (от англ. **l***i***s***t directory contents* — «отобразить содержимое директории»). Показывает содержимое текущей директории (папки). ls -a– выводит расширенный список содержимого (со скрытыми файлами с префиксом «.». ls -lпоказывает содержимое в подробном формате (флаги можно комбинировать, например, ls -al) **.** ls ~ выведет содержимое домашней директории вне зависимости от того, что показывает pwd. А ls .. покажет содержимое родительской директории.
3. команда — cd (от англ. **c***hange* **d***irectory* — «сменить директорию»). Она меняет текущую рабочую директорию на ту, которая указана в качестве параметра: cd имя\_папки. cd ~ переход в домашнюю директорию. cd "Фотографии с дня рождения" – если в названии папки есть пробелы. cd .. **–** возвращает на уровень выше (в род. Директорию).
4. команда touch (англ. «коснуться») с именем файла в качестве параметра: touch %ИМЯ\_ФАЙЛА.ФОРМАТ\_ФАЙЛА% создает файл в текущей директории с данным именем. команда touch ../../file.txt создаст файл file.txt на две папки выше по иерархии.
5. команда — mkdir (от англ. m*a*k*e* dir*ectory* — «создать директорию») создает директорию (папку). mkdir -p позволяет создавать структуру директорий (mkdir -p dir1/dir-inside/dir-deeper-inside).
6. команда cp (от англ. c*o*p*y* — «копировать»). В простом виде cp принимает два параметра: что копируем и куда копируем (cp %имя\_файла% %путь\_куда\_копировать%). можно указать сразу несколько файлов для копирования через пробел.
7. команда mv (от англ. m*o*v*e* — «переместить»). Перемещает указанные файлы и папки в другую директорию. Схема работы совпадает с cp.
8. Команда cat (от англ. c*on***cat***enate and print* — «объединить и распечатать») распечатает то, что содержится в файле. Команда вызывается вместе с именем файла.
9. Чтобы удалить файл, нужно напечатать команду rm (от англ. **r***e***m***ove —* «удалить») и передать ей имя файла. rm -r удаляет папку и все ее содержимое итеративным методом.
10. Удалить папку можно командой rmdir (от англ. **r***e***m***ove* **dir***ectory —* «удалить директорию»). Не забудьте указать имя папки.
11. clip < путь\_и\_название\_файла.формат – копирует содержимое файла в буфер обмена.
12. Команда && позволяет выполнять команды списком, являясь разделителем между ними.

Настройка Git

1. git version – выводит текущую версию гита.
2. git config --list – выводит все глобальные настройки, хранящиеся в файле .gitconfig. Или cat ~/.gitconfig.
3. git config --global user.name “name”; git config --global user.email email. Задает имя и почту текущего пользователя для удобной идентификации внесенных изменений в дерево проекта.

Создание репозитория

1. git init – создание репозитория. Для этого необходимо перейти в нужную директорию и вызвать данную команду. Она создает в текущей директории скрытую папку .git, в которой хранится служебная информация.
2. rm -rf .git – выполняет удаление репозитория («Разгичивает» папку). ключ **-r** (от англ. recursive — «рекурсивно») позволяет удалять папки вместе с их содержимым. ключ **-f** (от англ. force — «заставить») избавит вас от вопросов вроде «Вы точно хотите удалить этот файл? А этот? И этот тоже?».
3. git status – проверить состояние репозитория.
4. git add название\_файла.формат – добавляет в список отслеживаемых указанный файл. Git add --all – добавляет все файлы в папке. Git add . – добавляет текущую папку целиком (“.” – обращение к текущей папке).
5. команда git commit делает коммит. Ключ -m (от англ. **m***essage* — «сообщение») присваивает коммиту сообщение. Сообщение указывается после ключа и должно быть **в кавычках**.
6. git log – выводит историю всех коммитов, начиная с самого свежего. Получить сокращённый лог можно с помощью команды git log –oneline.

Когда компьютеры обмениваются данными в сети, они следуют сетевым протоколам (англ. network protocols) — правилам обмена данными между компьютерами.

Один из наиболее распространённых сетевых протоколов — **SSH** (от англ. Secure Shell Protocol). Он обеспечивает безопасный обмен данными в сети. С помощью этого протокола можно получать данные с удалённого компьютера или отправлять их на него. Трафик шифруется, поэтому протокол безопасен.

SSH использует пару ключей для обеспечения безопасности — публичный и приватный:

* **Приватный ключ** (англ. private key) хранится только на вашем компьютере и не должен передаваться кому-либо ещё. Он используется для расшифровки данных.
* **Публичный ключ** (англ. public key) доступен всем и используется для шифрования данных. Они могут быть расшифрованы парным приватным ключом.

ssh-keygen -t ed25519 -C [fsa1999@yandex.ru](mailto:fsa1999@yandex.ru) – генерация пары ключей.

1. перейдите в каталог локального репозитория и введите команду git remote add (от англ. *remote* — «удалённый» и *add* — «добавить»). Команде необходимо передать два параметра: имя удалённого репозитория и его URL. В качестве имени используйте слово **origin** (это название удаленного репозитория, по которому к нему можно обращаться, вместо **origin** можно выбрать другое имя). А URL нужно скопировать со страницы удалённого репозитория.

git remote add origin git@github.com:%ИМЯ\_АККАУНТА% /%

имя\_репозитория%.git

чтобы переименовать удаленный репозиторий в консоли надо ввести команду:

git remote rename <old> <new>

git remote -v **–** проверка связи репозиториев. Флаг -v — короткая форма флага --verbose (англ. «подробный»).

Каждый коммит сохраняет актуальное состояние файлов. Сами же коммиты хранятся в **ветках** (англ. *branch*).

1. команда git push (от англ. *push* — «толкать») позволяет загрузить содержимое локального репозитория на GitHub.

В первый раз эту команду нужно вызвать с флагом -u и параметрами **origin** (имя удалённого репозитория) и **main** или **master** (название текущей ветки локального репозитория). Флаг -u свяжет локальную ветку с одноимённой удалённой. Подобно связыванию локального и удалённого репозиториев, так же нужно дополнительно связать ветки данной командой и флагом.

git push -u origin main

eval `ssh-agent -s`

ssh-add ~/.ssh/id\_ed25519.pub

Хеш - набор символов всегда одной длины, задаваемой алгоритмом хеширования. Для каждого коммита создается свой уникальный хеш-идентификатор.

Для отображения хеша всех коммитов используется команда git log. Она выводит для каждого коммита его хеш, автора, дату создания, сообщение.  
Для вывода краткой информации о коммитах используется команда git log -oneline, которая выводит сокращенную форму хеша (размер зав. от количества  
коммитов) и сообщение. Чтобы выйти из режима просмотра логов, надо нажать Q.

Файл HEAD содержит ссылку на хеш последнего коммита. Его можно использовать при указании хеша последнего коммита в консольных командах.

У файлов в репозитории могут быть 4 статуса: **untracked, tracked, staged, modified**.

* **untracked** - для всех новых файлов, к которым не была применена команда git add. Они не остлеживаются VSC;
* **tracked** - для всех файлов, к которым была применена команда git add. Они остлеживаются VSC;
* **staged** - для всех файлов, к которым была применена команда git add. Означает, что файл помещен в **staged area**, отслеживается, но еще не сохранен в коммит;
* **modified** - для всех файлов, которые уже отслеживаются VSC, т.е. к которым была применена команда git add и они имели статус **staged** (или единственный статус **tracked**,  
  получаемый после коммита файлов), но которые были изменены.  
  в статусе **staged** могут находиться как новые добавленные к отслеживаемым файлы, так и уже закоммиченные, но модифицированные. отличие будет в том, что для первых префикс в выводе git status  
  будет **new file**, а для старых, но измененных **modified**.

Внесение изменений в коммит

можно внести правки в уже сделанный коммит с помощью опции --amend (от англ. *amend —* «исправить», «дополнить»). --amend рассчитан на работу с последним коммитом (HEAD). Для этого надо добавить нужные файлы в staged (командой add) и выполнить команду

git commit -amend --no-edit. Опция --no-edit сообщает команде **commit**, что сообщение коммита нужно оставить как было.

Изменить сообщение коммита — git commit --amend -m "Новое сообщение"

Откат изменений файлов и коммита

Выполнить unstage изменений — git restore --staged <file>. Данная команда уберет из staged указанный файл. git rm cached

«Откатить» коммит — git reset --hard <commit hash>

«Откатить» изменения, которые не попали ни в **staging**, ни в коммит, — git restore <file>

Просмотр изменений. Игнорирование файлов

команда git diff (от англ. **diff*erence*** — «отличие», «разница») позволяет просмотреть изменения в файле относительно его старой версии. Данная команда работает с **modified**-файлами. Для работы со **staged** -

файлами используется команда git diff –staged.

* Строки **--- a/teremok.txt** и **+++ b/teremok.txt** говорят, что дальше будет выведен результат сравнения файлов a/teremok.txt и b/teremok.txt — исходной и текущей версий.
* Строка **@@ -1,2 +1,2 @@** сообщает, какие строки файла попали в сравнение. Выражение **1,2** (неважно, с плюсом или с минусом) говорит, что были использованы две строки, начиная с первой. Если бы было, например, написано +15,7, это значило бы, что в сравнении участвуют 7 строк, начиная с 15-й.   
  Выражение со знаком минус (-1,2) относится к «оригинальной» версии файла (a/teremok.txt), а со знаком плюс (+1,2) — к «изменённой» (b/teremok.txt).

git diff a9928ab 11bada1 покажет различия коммитов с хешами a9928ab и 11bada1.

Команда echo “text” >> file – добавляет текст в файл.

Команда echo “text” > file – добавляет текст в файл, очистив перед этим файл.