**git --version** - показывает версию GIT установленную на локальном компьютере.

**git --help** - справка по GIT

1. **git init** - команда создаёт в текущем каталоге новый подкаталог с именем .git, содержащий все необходимые файлы репозитория — структуру Git репозитория

(вводится в терминале в созданной для проекта папке).

1. **git status -** выводит список проиндексированных и неотслеживаемых файлов, а также файлов, удаленных из индекса Git.
2. **git add -** команда git add добавляет (индексирует) изменение из рабочего каталога в раздел проиндексированных файлов. Она сообщает Git, что вы хотите включить изменения в конкретном файле в следующий коммит. Однако на самом деле команда git add не оказывает существенного влияния на репозиторий: изменения регистрируются в нем только после выполнения команды git commit.

**git add name file –** добавляет на отслеживание конкретный файл.

**git add . -** добавляет все файлы.

1. **git commit** - делает для проекта снимок текущего состояния изменений, добавленных в раздел проиндексированных файлов. Коммит проиндексированного состояния кода. Эта команда откроет текстовый редактор с предложением ввести комментарий к коммиту. После ввода комментария сохраните файл и закройте текстовый редактор, чтобы выполнить коммит.

**git commit –a -** выполнение коммита состояния со всеми изменениями в рабочем каталоге. Эта команда включает только изменения отслеживаемых файлов (тех, которые были в какой-то момент добавлены в историю с помощью команды git add).

**git commit -m "commit message"** - быстрая команда, которая создает коммит с указанным комментарием. По умолчанию команда git commit открывает локально настроенный текстовый редактор с предложением ввести комментарий к коммиту. При передаче параметра -m текстовый редактор не открывается, а используется подставленный комментарий.

**git commit -am "commit message"** - команда для опытных пользователей с параметрами -a и -m. Эта комбинация параметров создает коммит всех проиндексированных изменений и добавляет к коммиту подставленный комментарий.

**git commit –amend - э**тот параметр добавляет команде commit новый уровень функциональности. При передаче этого параметра будет изменен последний коммит. Вместо создания нового коммита проиндексированные изменения будут добавлены в предыдущий коммит. Кроме того, эта команда откроет настроенный текстовый редактор системы с предложением изменить указанный ранее комментарий к коммиту.

1. **git branch** - позволяет **создавать, просматривать, переименовывать и удалять ветки**. Она не дает возможности переключаться между ветками или выполнять слияние разветвленной истории. Именно поэтому команда git branch тесно связана с командами git checkout и git merge.

**git branch** - отображение списка веток в репозитории. Это синоним команды git branch --list.

**git branch <branch> -** cоздание новой ветки с именем ＜ветка＞. Эта команда ***не*** выполняет переключение на эту новую ветку.

**git branch -d <branch> -** удаление указанной ветки. Это «безопасная» операция, поскольку Git не позволит удалить ветку, если в ней есть неслитые изменения.

**git branch -D <branch>** - принудительное удаление указанной ветки, даже если в ней есть неслитые изменения. Эта команда используется, если вы хотите навсегда удалить все коммиты, связанные с определенным направлением разработки.

**git branch -m <branch> -** изменение имени текущей ветки на ＜ветка＞.

**git branch –a -** вывод списка всех удаленных веток.

1. **git checkout ＜branch＞**  - позволяет перемещаться между ветками, созданными командой git branch. При переключении ветки происходит обновление файлов в рабочем каталоге в соответствии с версией, хранящейся в этой ветке, а Git начинает записывать все новые коммиты в этой ветке. Рассматривайте эту команду как способ выбрать направление своей разработки.

**git checkout -b ＜new-branch＞ -** создается ветка <новая-ветка> и сразу же выполняется переключение на нее. По умолчанию команда git checkout -b создает ветку новая-ветка **от текущего указателя HEAD**.

**git checkout -b ＜new-branch＞ ＜existing-branch＞-** в приведенном выше примере передается ＜существующая-ветка>, поэтому новая-ветка будет создана от ветки существующая-ветка, а **не от текущего указателя HEAD**.