**Ветка - это просто указание на вершинный коммит и запись о предыдущих коммитах, образовавших ее текущее состояние.**

**Команды:**

git init - создать пустой репозиторий git

git add . - добавить все изменения в индекс. Можно вместо . указать [название файла]. Чтобы вернуть файл из индекса обратно в рабочую директорию используется команда git reset HEAD [название файла] или git rm --cashed [название файла] или git checkout [название файла]

git commit -m ‘текст коммита’ - сделать коммит в репозиторий. Можно добавить префикс -a после git commit для пропуска этапа git add . Изменить коммит можно командой git commit --amend. Другой путь для отмены коммита: чтобы отменить коммит, то нужно ввести git reset --hard [название коммита или @~ для последнего коммита] (переносит ветку на указанный коммит, удаляются все изменения из рабочей директории и из индекса) или git reset --soft [название коммита или @~ для последнего коммита] (переносит ветку на указанный коммит, но изменения остаются в рабочей директории и индексе), а потом снова git commit -m ‘текст коммита’. Если изменения уже у команды, то нужно использовать git revert @, чтобы создать коммит противоположный текущему коммиту.

git push - позволяет передать коммиты из локального репозитория в удаленный

git status - показывает текущий статус файлов

git show - покажет информацию о последнем коммите. Можно после команды указать конкретный [коммит], чтобы получить информацию о нем. Можно после команды указать HEAD~1 - посмотреть предпоследний коммит или HEAD~2 - посмотреть предпредпоследний коммит, а если после этого еще указать [:название файла], то можно посмотреть и содержимое файла

git rm [файл]- удаление файла из рабочей директории и из индекса. Можно удалить целую директорию, если поставить после команды -r [название директории]. Но лучше удалить файл или директорию вручную, потом сделать git add ., потом git commit -m ‘текст коммита’)

git branch - посмотреть список веток. Можно добавить префикс -v, чтобы посмотреть на вершинные коммиты этих веток

git branch [название ветки] или git checkout -b [название ветки] - это создание новой ветки, во втором случае происходит автоматический переход на созданную ветку. Можно использовать git branch [название ветки] [название коммита], чтобы создать новую ветку с вершиной в указанном коммите

git branch -f [название ветки] [название коммита] - возвращает или перемещает вперед выбранную ветку до состояния выбранного коммита (перед выполнением команды нужно уйти с перемещаемой ветки). Идентичная команда: git checkout -B [название

ветки] [название коммита], т.е. просто перемещаем существующую ветку в нужный коммит.

git branch -d [название ветки] - удаление ветки. Можно использовать префикс -D для удаления ветки, даже если коммиты в ней не смержены.

git checkout [название ветки] - переключает на указанную ветку. Можно добавить префикс -f после git checkout, чтобы переключиться на другую ветку даже при незакоммиченных изменениях, но если такие изменения нужно сохранить, то лучше сначала git stash, потом git checkout [название ветки], потом git checkout [изначальная ветка], потом git stash pop

git checkout -f - если надо удалить незакоммиченные изменения в текущей ветке и в ней же остаться. Можно после команды указать [название файла], чтобы удалились изменения только из этого файла. Если после этой команды нужна более глобальная чистка: удаление всех неотслеживаемых файлов, в том числе из gitignore, то можно использовать git clean -dxf

git checkout [название коммита] [название файла] - восстанавливает конкретный файл в состояние, которое было при указанном коммите. Чтобы отменить такое восстановление, то надо ввести git checkout HEAD [название файла] - вернет в рабочую директорию такой файл, который был последний раз закоммичен в репозиторий (можно указать без HEAD, если файл надо восстановить не из репозитория, а из индекса)

git checkout [название коммита] - переключение на конкретный коммит, так лучше не делать, поскольку дальнейшее создание коммитов отсюда будут без ветки, т.е. в подвешенном состоянии, и их придется вручную перетаскивать в созданную ветку с помощью git cherry-pick [название коммита] или создавать под такие коммиты новую ветку - по факту этой командой можно просто посмотреть каким был проект на момент выбранного коммита, но желательно не создавать новые коммиты, не переключившись обратно в нужную ветку с помощью git checkout [название ветки]

git log - информация о сделанных коммитах. Можно использовать префикс --oneline, чтобы сделать вывод коммитов более компактным. Можно использовать префикс -p, чтобы подробнее узнать про изменения каждого коммита. Можно использовать git log [ветка 1]..[ветка 2], чтобы посмотреть коммиты в ветке 1 с момента ответвления от ветки 2.

git reflog - посмотреть историю перемещения HEAD

git merge [название ветки] - вливает в текущую ветку указанную ветку. Чтобы отменить слияние - нужно из текущей ветки ввести команду git reset --hard @~, но если изменения уже у команды, то нужно ввести git revert [коммит слияния] -m 1

git cherry-pick [название коммита] - копирует дельту указанного коммита в текущую ветку

git diff [коммит/ветка 1] [коммит/ветка 2] - сравнение двух коммитов или веток. Можно использовать git diff HEAD, чтобы сравнить состояние рабочей директории и состояние репозитория. Можно использовать git diff, чтобы сравнить состояние рабочей директории и состояние индекса. Можно использовать git diff -cashed, чтобы сравнить состояние индекса и состояние репозитория.

git rebase master - переносит начало ответвления текущей ветки на последний коммит в мастере (т.е. текущая ветка будет основана на актуальном состоянии мастера). Использовать git rebase --abort для отмены, если в процессе возникли конфликты. Использовать git rebase --continue, если конфликт разрешен

git pull - используется для извлечения и загрузки содержимого из удаленного репозитория и немедленного обновления локального репозитория этим содержимым