git status посмотреть есть ли изменённые файлы и подготовленные к коммиту

git add <file(s)> например git add . Точка означает добавить все файлы в индекс

git commit –m <message> - сделать коммит

git commit -a –m <message> - Добавить файлы в индекс и закомитить

git log - смотреть все комиты

git log - p -<n> - Смотреть n-ое число последних коммитов

git reflog -n 10 или git reflog (выход q) - показывает все команды

git log --graph --topo-order --decorate --oneline –all - смотреть коммиты немного с графикой

git show - смотреть только последний коммит

git config --global –-list - смотреть настройки пользователя

git config --global user.name "Ваше И.Ф.О" - сменить отображаемой имя

git config --global user.email “email” - сменить email

После установки гита появится Git Bash (можно найти в поиске) Все команды нужно писать в нём.

В git status (В статусе)

Changes to be committed: - Добавлены файлы но не закомичены

Changes not staged for commit: - Файлы изменены но не добавлены

Untracked files: - Новые файлы не добывлены и не отслеживается системой

git add –p - (-p) означает что нужно добавить не весь файл а часть из него

.gitignore - если создать в проекте файл с этим названием, то изменения вообще по нему не отслежив-ся. И все те файлы или папки, названия которых будут там прописаны тоже не отсеж-ся.

git rm --cached <file name> - удаляем этот файл из отслеживания

git rm <file name> - реально удалится файл из директорий.(осторожно!!!)

git mv <old name> <new name> - переименовать файл

git diff - смотреть изменения в версиях комита

пример просмотра: git diff HEAD~1 смотреть изменения в последнем комите

git diff HEAD~2 смотреть изменения в предпоследнем комите и т.д

git diff <file name> - показывает какие были сделаны изменения в файле, но не добавлены в гид и не закомичены.

git diff –-staged <file name> - посмотреть, что вы проиндексировали и что войдёт в следующий коммит (--staged и --cached синонимы)

git diff --name-only <номер хеша> - смотреть список файлов которые были изменены в этом комите.

git checkout -- <имя файла> - отменить изменения которые были сделаны в файле но не добавлены в гит

git commit --amend - добавить изменения в существующий последний комит

git revert <номер хеша> - отменить сделанный комит по хешу и переходит на предыдущий комит (изменения удаляются) Если есть и после него комиты , то они не удаляются

git revert –m 1 <номер хеша на котором merge> - Отмена ручного merge. Нужно перейти на ветку в которую был merge.

git reset --hard <номер хеша> - Удаляет от выбранного по хешу комита все вышестоящие комиты со всеми изменениями (лучше не делать, можно потерять все изменения после него) Пример: набираем git reflog смотрим до какого HEAD хотим откатиться (как пример HEAD@{84}) и пишем git reset –-hard HEAD@{84} и всё что выше него удаляется. Если вы откатываете коммит, в котором добавили файл, то после отката этот файл будет удалён из репозитория.Здесь можно использовать флаг --soft, который означает, что после отката коммита файл останется как не зафиксированные изменения.

git reset --hard - если писать без номера хеша, то будет удалены все изменениями которые небыли закомичены до последнего сделанного комита.

git reset --soft <номер хеша> - будут удалены все комиты сделанные после этого комита а те изменения которые былы сделаны в этих удалённых комитах снова помечаются как изменённые но не удаляются. (Изменения не теряются)

git reset --mixed HEAD <имя файла> - команда выкидывает уже добавленный в индекс файл. Он перекидывается в Changes not staged for commit. (данные не удаляются)

Можно не писать --mixed Например можно написать git reset <номер хеша> (Это то же самое)

git reset <номер хеша> Отличие от git revert в том что при git revert удаляется данный комит, а при git reset удаляется всё что было после после данного комита.

git init - команда которая создаёт репозиторий в папке

git clone <URL> - склонировать к себе удалённый репозиторий по url

git push - команда которая отправляет комиты сделаные локально на удалённый репозиторий (отправляются только комиты)

git push -u origin <имя репозитория> - так можно отправить новый созданный репозиторий в локальном компе на удалённый репозиторий в которм его ещё не существует. Или оправить все изменения в этот репозиторий если он уже есть.

git pull - команда которая позволяет получить из удалённого репозитория изменения которые там были.

git pull –-rebase - Использовать эту команду если кто то перебазировал ветку на сервере, чтобы чисто слить в свою ветку. Либо сделать это вручную с помощью git fetch, а затем git rebase <имя ветки>

git checkout –b <имя> - команда которая создаёт ветку в гите (главная ветка это master)

git checkout без параметров - просто переключает на другую ветку

например: git checkout master

git remote - посмотреть список всех удалённых серверов

git remote –v - посмотреть адреса всех удалённых серверов

git remote show - для получения удалённых веток и дополнительной информации

git remote add <имя удалённого сервера> <ссылка на репозитори> - Для того, чтобы добавить удалённый репозиторий и присвоить ему имя (shortname)

пример: git remote add shortname https://github.com/paulboone/ticgit

git branch –m <имя старой ветки><имя новой ветки> - команда переименовать ветку (переименует в локальном репозитории, на удалённом будет по старому)

git branch –l - показать все ветки которые есть в проекте

git branch –v - Чтобы посмотреть последний коммит на каждой из веток

git branch –vv - Если вы хотите посмотреть как у вас настроены ветки слежения

Это выведет список локальных веток и дополнительную информацию о том, какая из веток отслеживается, отстаёт, опережает или всё сразу относительно отслеживаемой.

git fetch --all; git branch –vv - Если вы хотите иметь актуальную информацию с серверов

git branch –-merged - Чтобы посмотреть те ветки, которые вы уже слили с текущей

git branch –-no-merged - Чтобы посмотреть те ветки, которые вы ещё не слили с текущей

git branch –d <имя ветки> - удалить ветку (в локальном репозитории, на удалённом она будет)

git push --delete origin <имя ветки> - удалить ветку на удалённом репозитории

git diff master..dev - сравнить две ветки (какие изменения)

Чтобы сравнить ветку на локальном репозитории с удалённым репозиторием, нужно сначала вытянуть эту ветку из удалённого репозитория (не скачать) с помощью команды: git fetch а потом уже сравнивать командой:

git diff master..origin/master (сравниваю ветку master)

git merge <имя ветки которую будем сливать к себе> - слияние веток

Можно так же делать слияние в IntelajIdea VCS-> Git-> Merge Changes -> там выбираем с чем сливаться и кто первый а кто второй.

git merge --abort - команда которая отменяет последний merge, но только тот который произошёл автоматически. (Ручной не отменяет)

git rebase master - это то куда будем вливать. Перед этим нужно переключится на верку откуда будем вливать.

git rebase --onto master server client - В этой команде говорится: “Переключись на ветку client, найди изменения относительно ветки server и примени их для ветки master”.

git rebase master server - Эта команда добавит изменения ветки server в ветку master (без предварительного переключения на неё)

git blame <имя файла> - можно посмотреть кем и когда были внесены изменения и что именно и в каких комитах. Это же можно посмотреть и через IntelajIdea -> правой кнопкой на имени файла -> Git -> Annotate

git stash - Ты делал изменения в коде, и вдруг нужно заняться чем-нибудь другим и не добавлять и не комитеть это пока.(Сохраняется на компе)

git stash list - Показать все сохранённые stash-и

git stash pop - Вставить последний stash

git gc - позволяет привести дапозиторий в порядок, уменьшить его размер и т.д