Помощь:

$git help команда или

$git команда --help или

$man git-команда

Создание репозитория:

Через gitbash заходим в папку с файлами

git init

Задаем глобальные настройки:

git config --global user.name "имя пользователя"

git config --global user.email почтовый адрес

или локальные:

git config --local user.name "имя пользователя"

git config --local user.email почтовый адрес

Ставим нормальный редактор:

git config --global core.editor "'C:/Program Files (x86)/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin" - для глобальных настроек и

git config --local core.editor "'C:/Program Files (x86)/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin" - для отдельного репозитория

Смотрим наши настройки:

git config --list или

git config ключ

Сам файл настроек в папке пользователя с расширением .gitconfig там все прописано

Все настройки репозитория можно глянуть в папке гита

Делаем файл исключений .gitignore:

Создаем текстовый файл с именем .gitignore при сохранении указываем тип файла Все файлы, пишем в нем:

примеры на https://github.com/github/gitignore

Чтобы посмотреть игнорируемые файлы пишем:

git status --untracked-files=all

Добавим файлы в индекс:

git add . добавится вся папка

git add имя файла добавится конкретный файл

git add "*.txt" добавятся файлы с заданным расширением

Удалим файлы из индекса:

git rm --cached имя файла

Делаем коммит:

git commit -m имя файла"комментарии"

это закоммитили конкретный файл

git commit -a -m"комментарии"

это закоммитили все добавленные в индекс файлы

git commit -a -m'комментарии'

комментировать можно и в одиночных кавычках

git commit -a

коммитит сразу все отслеживаемые файлы позволяя обойтись без git add

Если же набрать просто

git commit

то нас выкинет в редактор гита vim, и чтобы выйти из него надо:

нажать Escape, набрать :wq или :x(сохранить и выйти)

Просмотр состояния:

git status или пока не проиндексировал

git diff или git diff --cached или git diff --staged

Откат изменений:

git checkout -- имя файла

История коммитов :

git log

если вылезет журнал, выйти из него можно нажав q

git log -p -количество нужных коммитов

git log имя файла коммиты этого файла

git log --stat сокращенная статистика для каждого коммита

gitk - просмотр в графическом клиенте более подробно

Операции отмены:

git commit --amend переделать коммит

git reset HEAD файл отменить индексирование файла

git checkout --файл отбросить сделанные изменения

Чтобы удалить файл из GIT:

git rm файл или

git rm директория/*расширение

удаляет файлы в директория с расширение

git rm *символы

удаляет файлы , чьи имена заканчиваются на символы

git rm -f файл

принудительное удаление, если файл изменен и уже проиндексирован

git rm --cached имя файла

удаляет файлы из индекса

Удаленные репозитории.

git remote просмотр доступных удаленных репозиториев

git remote -v просмотреть адреса для чтения и записи, привязанные к репозиторию

git remote add имя адрес добавить удаленный репозиторий и присвоить ему имя. Теперь вместо указания полного пути можно использовать имя

origin - это имя по умолчанию, которое GIT присваивает серверу, с которого вы склонировали

git fetch имя извлечь всю информацию, которая есть в репозитории имя. Ветка master имя теперь доступна локально как имя/master

Клонирование репозитория:

git clone адрес или

git clone адрес нужная папка

По умолчанию git clone автоматически настраивает вашу локальную ветку master на отслеживание удаленной ветки master на сервере, с которого вы клонировали

git pool автоматически извлекает и затем сливает данные из удаленной ветки, с которой вы изначально склонировали в вашу текущую ветку, над которой вы в данный момент работаете

git push удаленный_сервер ветка отправить данные на удаленный сервер

пример $ git push origin master

Создание веток:

git branch название ветки

git checkout -b название ветки

-b - создать новую ветку и перейти в нее

Посмотреть какие ветки есть:

git branch

git branch -v посмотреть последние коммиты в ветках

Перейти на нужную ветку:

git checkout имя ветки

Слияние веток:

Сначала укажем утилиту для конфликта мержа:

```
git config --local merge.tool kdiff3 или
git config --global merge.tool kdiff3
```

Мержим:

```
git merge название ветки
```

При наличии конфликта набираем:

```
git mergetool
```