**Git –** представитель системы контроля версий кода.

**Репозиторий** (локальный и удаленный) – место хранения файлов или версий кода.

**Master branch** – ствол кода (готовая к развертыванию и запуску в рабочей среде)

**Branch** – ветка кода. Создается пользователем при работе над задачей.

**Commit** – момент внесения изменений (CRUD) в ветку кода. Commit 1, 2, 3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Инициализация репозитория**

**git init** – установи текущую папку в качестве локального репозитория.

**Синхронизация удалённого с локальным репозиториев**

**git remote add origin адрес репозитория** — привяжи Локальный репозиторий к Удалённому

**git remote -v** (*verbose*, «подробный») — проверь, что репозитории действительно связались

**git clone адрес репозитория** – клонировать удаленный репозиторий

**git fetch** – скачать обновления из удаленного репозитория но не сливать их с локальным

**git merge имя ветки которую нужно влить** – слить (мёрджить) указанный бранч с текущем бранчем

**git pull** – (fetch + merge) загрузка файлов из удаленного репозитория на локальный

**Перемещение по репозиторию**

**pwd –** проверка нахождения в репозитории

**cd папка или путь к файлу –** переход в папку или к файлу

**Создание, редактирование, удаление файлов**

**Touch имя файла и расширение –** создание файла в текущем репозитории

**Echo ‘текст’ >> название файла** – добавление строчки в файл. Если только > то замена последней строчки

**Cat имя файла** – просмотр содержимого файла

**git rm** **имя файла** – удалить файл из коммита

**chmod +x имя файла** - эта команда сделает файл исполняемым

**./имя файла** – исполнить скрипт

**Подготовка файла к коммиту**

**git add** имя файла (от англ. *add*, «добавить») — подготовь файл к коммиту

**git add --all** (от англ. *add*, «добавить» + *all*, «всё») — подготовь к коммиту сразу все файлы, в которых были изменения, и все новые файлы

**git add .** — подготовь к коммиту текущую папку и все файлы в ней

**Создание и публикация коммита**

**git commit -m** "Комментарий к коммиту." (от англ. *commit,* «совершать», фиксировать» + ***m****essage,* «сообщение») — сделай коммит и оставь комментарий, чтобы было проще понять, какие изменения сделаны. После git add

**git push** (от англ. *push*, «толкать») — добавь изменения в удалённый репозиторий. После git add и commit -m

**git push -u origin main** (от англ. *push*, «толкать») — в первый раз загрузи все коммиты из локального репозитория в удалённый с названием origin

💡 Ваша ветка может называться master, а не main. Подправьте команду, если это необходимо

**Просмотр информации о коммитах**

**git log** (от англ. *log*, «журнал [записей]») — выведи подробную историю коммитов;

**git log --oneline** (от англ. *log*, «журнал [записей]» + *oneline*, «одной строкой») — покажи краткую информацию о коммитах: сокращённый хеш и сообщение

**Просмотр состояния файлов**

**git status** (от англ. *status*, «статус», «состояние») — покажи текущее состояние репозитория

**git branch** - позволяет получить список существующих веток проекта

**git branch имя ветки** –создать новую ветку

**git checkout имя ветки** - переключиться на другую ветку

**git checkout -b имя ветки** – создать ветку и сразу перейти в нее

**Добавление изменений в последний коммит**

**git commit --amend --no-edit имя файла** (от англ. *amend*, «исправить») — добавь изменения к последнему коммиту (HEAD) и оставь сообщение прежним

**git commit --amend -m "Новое сообщение"** — измени сообщение к последнему коммиту на Новое сообщение

💡 Выйти из редактора Vim: нажать Esc, ввести :qa!, нажать Enter

**«Откат» файлов и коммитов**

**git restore --staged** имя файла — переведи файл из состояния staged обратно в untracked или modified. т.е. убрать файл из коммита (после git add)

**git restore –staged.** – убрать все файлы из коммита

**git restore имя файла** — верни файл к последней версии, которая была сохранена через git commit или git add

**git reset --hard** номер хэша который станет HEAD — удали все незакоммиченные изменения из staging и «рабочей зоны» вплоть до указанного коммита

**git branch -d имя ветки -** удалить ветку донора после слияния (merge)

**Просмотр изменений**

**git diff** (от англ. ***diff****erence*, «отличие», «разница») — покажи изменения в «рабочей зоне», то есть в modified-файлах

**git diff хэш 1 хэш 2** — выведи разницу между двумя коммитами

**git diff --staged** — покажи изменения, которые добавлены в staged-файлах