1. **git init** - инициализировать репозиторий
2. **git status** - проверить статус изменений
3. **git add .** - добавить изменения
4. **git status** - проверить статус изменений
5. **git commit -m "add index und style"** - создать коммит с сообщением "добавлен индекс и стиль"
6. **git fetch** - это команда, которая связывается с удаленным репозиторием и извлекает все данные о нем, которых у вас нет, но которые есть на удаленном репозитории. После выполнения этой команды у вас появятся все изменения, которые были в удаленном репозитории, но которых у вас не было на локальной машине. Однако эта команда не изменяет вашу рабочую директорию и текущую ветку.
7. **git remote add origin git@github.com:VersinaV/Git-Lesson03.git** - добавить удаленный репозиторий с именем "origin" и URL-адресом "[git@github.com](mailto:git@github.com):VersinaV/Git-Lesson03.git"
8. **git remote -v** - это команда, которая показывает URL-адреса для удаленных репозиториев, связанных с вашим локальным репозиторием. Она выводит список удаленных репозиториев, указывая их имена (обычно "origin") и соответствующие URL-адреса для извлечения (fetch) и отправки (push) данных.
9. **git push -u origin master** - отправляет ваши локальные изменения из текущей ветки (обычно master) в удаленный репозиторий с именем origin. Опция -u или --set-upstream устанавливает соответствие между вашей локальной веткой и удаленной веткой master на origin. Это обычно делается при первой отправке ветки, чтобы в будущем вы могли просто использовать git push без указания имени ветки и удаленного репозитория.

Измененные шаги:

1. **git add .** - добавить изменения
2. **git commit -m "add change css and html"** - создать коммит с сообщением "добавлены изменения в CSS и HTML"
3. **git status** - проверить статус изменений
4. **git push -u origin master** - отправить изменения на удаленный репозиторий.

## Ссылки

* [Markdown](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet)
* [Ветвление](https://git-scm.com/book/ru/v2/%D0%92%D0%B5%D1%82%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5-%D0%B2-Git-%D0%9E-%D0%B2%D0%B5%D1%82%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B8-%D0%B2-%D0%B4%D0%B2%D1%83%D1%85-%D1%81%D0%BB%D0%BE%D0%B2%D0%B0%D1%85)
* [Ассиметричное шифрование](https://encyclopedia.kaspersky.ru/glossary/asymmetric-encryption/)
* [Перенос ключей между компьютерами](https://btip.ru/perenos-klyuchej-ssh-s-odnogo-kompyutera-na-drugoj/)

## Этапы работы

1. **Подготовительный этап**
   * Установка GIT на компьютер
   * Настройка Git на компьютере
   * Наличие аккаунта на GitHub
   * Выгрузить публичный ключ на GitHub
   * Просмотр публичного ключа
   * Генерация пары ключей (ВЫПОЛНИТЬ ТОЛЬКО В НАЧАЛЕ КАРЬЕРЫ)
   * Скопировать публичный ключ на GitHub
   * Проверить, что ключ выгружен на GitHub

## Проверка текущей версии Git

bash

Copy code

git -v git --version

## GIT (2005)

**Что это?**

* Система контроля версий
* Хранилище
* Машина времени
* Умная система управления кодом проекта
* Содержит все файлы проекта
* Своебразная БД, т.е. хранит в структурированном виде
* Можно восстановить предыдущие версии файлов
* Есть обмен версиями между разработчиками
* Проверка целостности - надежная система

## Git: эко-система

1. **Git** - прикладная программа
2. **GitHub** - интернет-площадка
3. **GitHub Pages** - бесплатный хостинг (**html/css/js**)

### Основные способы авторизации в Интернет

1. По паролю
2. По публичному ключу

## Виды программных интерфейсов

1. **GUI** (Graphical User Interface) - графический интерфейс пользователя
2. **CLI** (Command Line Interface) - интерфейс командной строки

## Необходимые знания для работы в команде (Git/GitHub)

1. Освоить базовый порядок работы с Git/GitHub
2. Освоить механизм ветвления
3. Освоить модели совместной разработки

## Базовые понятия

* **проект** - каталог с файлами исходного кода (кодовая база)
* **репозиторий** - хранилище истории разработки проекта
* **коммит (как объект)** - точка сохранения проекта (снимок снимать)

## Базовый (основной) порядок работы с Git/GitHub

1. **Подключения Git к проекту / инициализация нового репозитория**
   * **git init**
   * Создается "скрытое хранилище" - каталог **.git/**
   * Выполнять только для новых проектов
2. **Сохранение**
   * Индексация файлов (добавить в очередь на сохранение)
   * **git add .**
   * Фиксация (сохранение как таковое)
   * **git commit -m 'описание изменений'**
3. **Создание нового репо на GitHub**
4. **Привязка репо LOCAL ↔ REMOTE**
   * **git remote add origin скопированная\_ссылка**
5. **Выгрузка ветки на GitHub**
   * **git push -u origin название\_ветки** (**main**, **master** и т.д.)
   * **git push -u origin main** (пример)
   * **git push** (если ветку уже выгружал)

## Стандартная последовательность команд

bash

Copy code

git add . git commit -m 'update' git push

## Просмотр текущей привязки

bash

Copy code

git remote -v

## Файл README.md

* Использует формат **Markdown**
* Описание репо на **GitHub**
* Помещается в корень проекта, как правило

**Пример**

markdown

Copy code

## Test Project

## Ветвление в Git

* Ветка - еще одна версия проекта (изолированный поток разработки)

### Стратегии ветвления в Git

1. **Git Flow**
   * **main/master/stable** - long-term
   * **develop/current** - long-term
   * **login/bugfix1/payments** - short-term
2. **GitHub Flow**
   * **main/master/stable** - long-term
   * **login/bugfix1/payments** - short-term

## Работа с ветками