**Git-GitHub**

Репозитории - это хранилища кода

Системы управления репозиториями позволяют работать над проектом в команде.

Так-же гит дает доступ к управлению версий вашего проекта пошагово, из чего и образуется ветка.

**Системы контроля версий — VCS (version control systems)**

**Репозиторий** — это общее название для хранилища исходного кода. Часто говорят об **удалённом репозитории** (копия кода на каком-то сервере) и **локальном репозитории** (копия на компьютере разработчика).

Код в репозитории хранится в виде истории **коммитов** — зафиксированных состояний исходного кода в определённый момент времени. Получить любое состояние можно практически мгновенно, не применяя последовательно все изменения.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Крч сам git это походу консольное дополнение а git hub это прога с gui для удобства

В Git имеется следующий набор команд:

* Git config
* Git init
* Git add
* Git diff
* Git commit
* Git reset
* Git status
* Git merge
* Git push
* Git pull

Команда **Git push** используется для отправки содержимого локального репозитория в удаленный репозиторий. После того, как локальный репозиторий был изменен, должна выполняться отправка, чтобы поделиться изменениями с удаленными членами команды. Отправка — это **способ передачи коммитов из локального репозитория в удаленный репозиторий**.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Если вы пользуетесь Visual Studio Code, то можно установить расширение для работы с Git и пользоваться консолью в VS Code.

Расширение называется start git bash

Качнул и то и то

git - показывает все команды

git -–version показывает версию гита

git help config – открывает справку в браузере

git init – создает новый репозиторий проекта

git clone <repository\_url> - клонировать удаленный репозиторий на свой пк

git branch – создать ветку проекта

git checkout <name> - переключиться на ветку

git add <имя\_файла> перед внесением изменений необходимо их обработать такой командой

или обработать все изменения вот такой командой git add .

Используйте следующие команды, чтобы задать свое имя пользователя и адрес электронной почты, которые будут связаны с вашими коммитами:

git config --global user.name "Your Name"

git config --global user.email "youremail@example.com"

Чтобы сохранить ваши изменения в репозитории Git, выполните следующие действия:

* Внесите изменения в свои файлы.
* Внесите изменения с помощью git add.
* Зафиксируйте изменения с помощью git commit -m "Сообщение о фиксации".

Git обеспечивает отличную поддержку совместной работы над проектами. Чтобы эффективно сотрудничать с другими, учитывайте следующее:

* Нажатие и извлечение: используйте git push для отправки ваших локальных коммитов в удаленный репозиторий и используйте git pull для извлечения и объединения изменений из удаленного репозитория в вашу локальную копию.
* Работа с ветвями: создавайте отдельные ветви для разных функций или исправлений ошибок. Используйте git merge для объединения ветвей, когда будете готовы интегрировать изменения.

Вот несколько основных команд Git, которые вы часто будете использовать:

* git init: инициализирует новый репозиторий Git.
* git clone <repository\_url>: клонирует удаленный репозиторий на ваш локальный компьютер.
* git add <имя\_файла>: создает определенный файл для следующего коммита.
* git commit -m "Сообщение о фиксации": создает коммит с поэтапными изменениями.

Начнем, создадим проект, перейдем в его папку, потом инициализируем проект. **git init**

Создастся в этой репозитории скрытая папка.

**Git status** выведет инфу о том где мы что мы и как мы

Untracked files:

(use "git add <file>..." to include in what will be committed)

Test.py

В статусе отобразился антрекэд который говорит, что есть наш файл и гит его не воспринимает чтобы с ним работать нужно прописать **git add name.py** потом опять пишем статус, чтобы позырить че поменялось.

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: Test.py

файл зафиксирован.

**git rm --cached <file>** - а с помощью этой команды можно опять не отслеживать его.

Далее в нашей репозитории давай создадим еще один файл, тоже пайтоновский.

Далее опять статус смотрим

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: Test.py

Untracked files:

(use "git add <file>..." to include in what will be committed)

Myfile\_git.py

Ну и логично второй не добавлен.

А чтобы добавить все файлы сразу можно прописать **git add .**

Далее погнали изменим один из наших файлов и посмотрим что в статусе.

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: Test.py

гит вывел что произошли изменения в файле и нужно его опять добавить add-ом.

Следующий этап это комиты.

Прописываем **git commit -m “name”**

В name указываем названия нашего коммита так же в кавычках.

У нас вылетела ошибка гит запросил мою почту и имя

Это две вот такие команды:

**git config --global user.email "elik@example.com"**

**git config --global user.name "elik"**

после их ввода комит заработает и вот что он вывел

[master (root-commit) 0cf6af8] One commit

2 files changed, 8 insertions(+)

create mode 100644 Myfile\_git.py

create mode 100644 Test.py

0cf6af8 – это ъэш коммита по которому к нему можно вернуться.

Далее важная тема если есть файлы которые мы хотим чтобы не обрабатывались гитом

Для этого мы должны создать файл с именем **.gitignore**

Внутри этого файла мы перечисляем список файлов и директорий которые хотим чтобы гит не обрабатывал.

Как то так name.py

Теперь в статусе этот файл не отображаеться а вот .gitignor отображаеться и его придеться фиксировать.

Создадим директорию и в ней к примеру новый файл txt допустим, назовем log

И чтобы ее тоже гит не хавал прописываем в гит игнор **/logs**

Теперь создадим комит новый так как мы добавили новый файл.

Теперь поговорим про ветки. Мы сейчас в основной ветке master.

Команда **git branch**

Выведет в какой мы сидим ветке.

А давай теперь попробуем создать новую ветку.

Делается так **git branch name**

После опять пропишем git branch и увидем следущее

New\_Vetka

\* master

Под звездой та ветка где мы.

Ветку можно удалять **git branch -D name**

Далее перключимся на новую ветку командой **git checkout name**

Так же можно не париться и создать новую ветку притом сразу перейти на нее командой **git checkout -b name**

Теперь в новой ветке давай че нидь сделаем, зафиксируем изменения и закомитим.

Условно мы в новой ветке создали новый файл.

А в старой его быть не должно там типо работа над другой херней.

И если мы перейдем на новую ветку, то оо о ооооооо этого файла там просто не будет.

Теперь надо как-то совместить ветки чтобы файл редми появился в нашей ветки мастера.

Находясь в мастер ветке мы прописываем

**git merge name\_vetka**

Updating a773ff7..6560651

Fast-forward

read\_me.txt | 7 +++++++

1 file changed, 7 insertions(+)

create mode 100644 read\_me.txt

Вот че мы получили на выходе.

И вот мы видим этот файл.

Теперь в принципе мы можем удалить ту не нужную ветку.

ВОТ ВПРНЦП И ВСЕ ОСНОВЫ ГИТА.

Сам гит локальный, а теперь давайте поработаем с сервером, с глобальным, будем использовать сайт git.hub

Крч регаемся на гит хабе.

После регистрации в самом гит хабе жмем на плюсик и добавляем новую репозиторию.

После создания репозитория, махаем обратно в вску.

**git config --global user.email "elik@example.com"**

**git config --global user.name "elik"**

и вот эти две команды выше сейчас пипец нужны чтобы связаться с нашим сервером на гит хабе.

Далее надо ввывести команду с гит хаба в гит, вот эту

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

И после ввода этой команды нам нужно залить то что у нас есть в гите на гит хаб командой **git push -u origin master(тут имя нашей ветки)**

И после обновления данной страницы мы увидим наш репозиторию с гита.

И у нас есть список из коммитов в гите.

Теперь локально еще раз че нидь в скрипте изменим.

Зафиксируем, закомитим, и зальем на гит хаб

**Git push**

А если наоборот на серваке произошли изменения а локально нет, то прописываем **git pull** и забираем все изменения с ги хаба в гит.