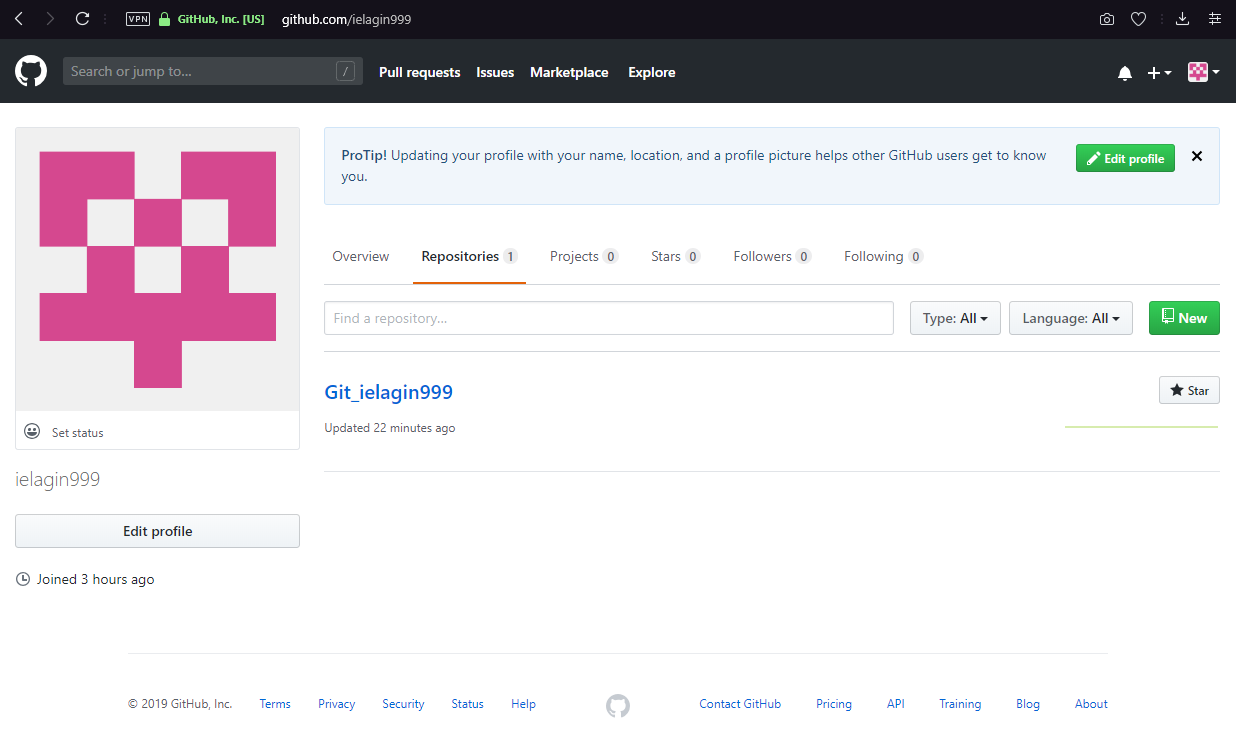
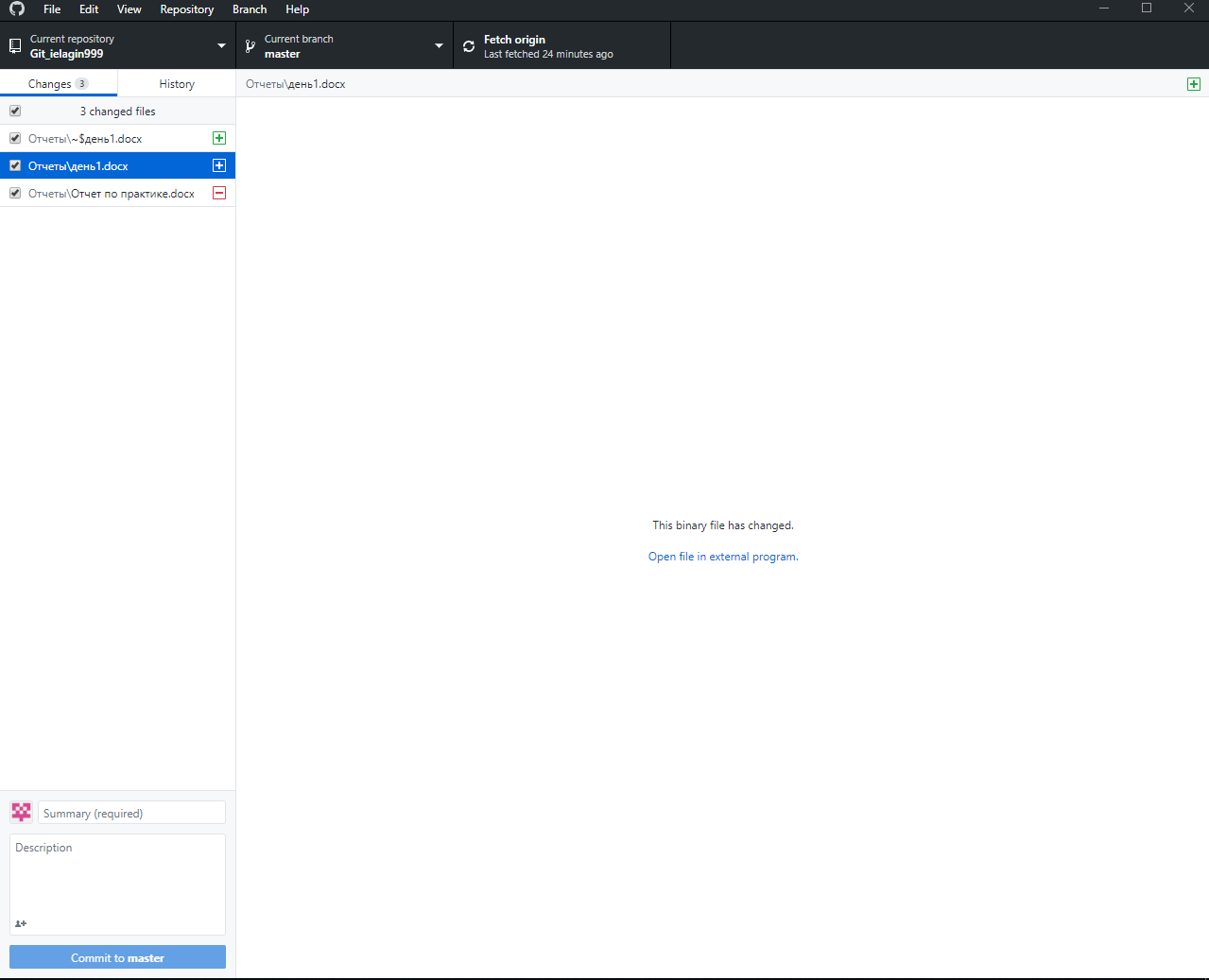
Регистрация на Github.com:



Установка клиента(в качестве клиента я выбрал GitHub Desktop):



**Что такое Git и зачем его использовать?**

Git — это система управления версиями. У Git две основных задачи: первая —  хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.

Репозиторий Git — это место, где хранится ваш код и вся информация о его изменениях. Репозитории могут находиться у вас на компьютере, на компьютерах ваших коллег и на удалённом сервере.

## КАК ПОЛЬЗОВАТЬСЯ GIT?

Кроме установки, вам нужно указать правильный адрес электронной почты и имя пользователя для доступа к серверу Git, например, на GitHub.

Обычно, структура проекта в Git будет зависеть от масштаба и сложности вашей программы. Но для начала мы будем использовать проект, состоящий только из одной ветви. Каждый проект содержит одну ветку по умолчанию, она называется master.

### СОЗДАНИЕ ПРОЕКТА

Когда настройка git завершена перейдем к вашему проекту. В самом начале вам достаточно создать папку для файлов проекта. Если вы собираетесь работать над несколькими проектами, создайте папку git в вашем домашнем каталоге, а уже туда поместите папки ваших проектов:

 mkdir -p ~/git/testing ; cd ~/git/testing

Эта команда создаст нужную структуру папок и переводит текущий каталог в только что созданный. Теперь создадим первый файл нашего проекта:

 touch file

Проект готов, но система контроля версий git еще не знает об этом.

### НАСТРОЙКА ПРОЕКТА В GIT

Перед тем как git начнет отслеживать изменения, нужно подготовить все необходимые конфигурационные файлы. Сначала инициализируем пустой репозиторий в нашей папке:

 git init

После того как репозиторий будет создан, вам нужно добавить свои файлы в него. Каждый файл нужно добавлять отдельно или сказать утилите, что необходимо добавить все файлы явно. Пока вы не добавите файл сам он не будет отслеживаться. Новые файлы в будущем тоже нужно добавлять, они не добавляются автоматически.  Сначала добавим текущую папку:

 git add .

Если все прошло хорошо, то команда ничего не выведет.

### ФИКСАЦИЯ ИЗМЕНЕНИЙ

Изменения тоже автоматически не отслеживаются. Фиксация изменений выполняется с помощью команды commit. Вам нужно указать что было изменено с помощью небольшого комментария, буквально в несколько предложений. Хорошая практика выполнять фиксацию перед каждым серьезным изменением.

Таким образом, вы будете хранить все версии проекта, от самой первой и до текущей, а также сможете знать что, когда и где было изменено. Чтобы создать свой первый коммит выполните:

 git commit -m "Initial Commit" -a

Команде необходимо передать два параметра, первый - это -m, ваш комментарий, второй -a, означает, что нужно применить действие ко всем измененным файлам. Для первого раза используется этот параметр, но обычно вам нужно указать измененные файлы или каталоги. Например, можно делать так:

 git commit -m "Changed file" file

### ОТПРАВКА ИЗМЕНЕНИЙ

До этого момента мы делали все в локальном репозитории. Вы можете использовать git локально, если нужен только контроль версий, но иногда нужно обменяться информацией с другими разработчиками и отправить данные в удаленный репозиторий.

Сначала нужно добавить удаленный репозиторий с помощью команды remote. Для этого нужно передать ей URL:

 git remote add origin https://github.com/Seriyyy95/testing.git

Затем можно посмотреть список удаленных репозиториев:

 git remote -v

Вы можете использовать не только github сервера, но и любые другие. Теперь для отправки ваших изменений используйте такую команду:

 git push origin master

Команда push указывает, что нужно отправить данные в удаленный репозиторий, origin - наш настроенный репозиторий, а master - ветвь.

### УПРАВЛЕНИЕ ВЕТВЯМИ

Для простых проектов достаточно одной ветви. Но если проект большой и он имеет несколько версий, в том числе тестовую, то может понадобиться создать для каждой из них отдельную ветвь. Сначала смотрим доступные ветви:

git branch -a

Опция -a указывает что нужно вывести все ветви, даже не синхронизированные. Звездочка указывает на активную ветвь. Теперь создадим ветвь для разработки с помощью команды checkout:

 git checkout -b develop

Переключаться между ветвями можно тоже с помощью той же команды:

 git checkout master  
$ git checkout develop

Теперь создадим еще один файл:

 touch develop

И добавим его в нашу новую ветвь develop:

 git add develop

Сделаем коммит для внесенных изменений:

 git commit -m "develop file" develop

Дальше проверим существует ли этот файл в основной ветке master или только в дополнительной. Смотрим текущую ветку:

 git branch  
$ ls

Затем переключаемся на ветку master и снова смотрим:

 git checkout master  
$ git branch  
$ ls

Здесь файла нет, так и должно быть. В git есть такая полезная вещь, как слияние. С помощью нее вы можете объединить две ветви. Например, переместить код из рабочей ветки в стабильную. Для этого достаточно выполнить команду merge:

 git merge develop --no-ff

Перед тем как будет выполнено слияние вам нужно ввести комментарий, зачем это нужно. Затем если вы еще раз выполните ls, то увидите, что здесь уже есть нужный файл. Наши примеры git подошли к концу.