

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО МГТУ «СТАНКИН»)**

Институт цифровых интеллектуальных систем

Кафедра компьютерных систем управления

Образовательная программа 15.03.04

«Автоматизация технологических процессов и производств»

Лабораторная №1 Работа с githab

Дисциплина «Основы системного программного обеспечения»

Выполнил:

студент гр. АДБ-21-07 \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ Елсуков М.В.

Принял:

Преподаватель           Ковалев И.А.

**Москва 2024г.**

Цель работы: Работа с системами контроля

Основные задачи работы:

1. Создать учетную запись на github.com

2. Создать локальный репозиторий

3. Зафиксировать изменения в области заготовленных файлов

4. Переслать локальный коммит на сервер

5. Сделать слияние веток

6. Просмотреть изменения и разрешение конфликтов

7. Удалить ветку на сервере и вернуть к предыдущему состоянию

8. Исправить коммит.

Ход работы:

1) 1. Создание учетной записи на github.com

Мы зашли на github.com и зарегистрировались на нем ,подтвердили регистрацию на своей почте.

Перед тем, как создавать репозиторий мы перешли в Settings и изменили название главной ветки с «main» на «master»

Далее создали новый репозиторий с названием Elsukov\_lab1 на github.com

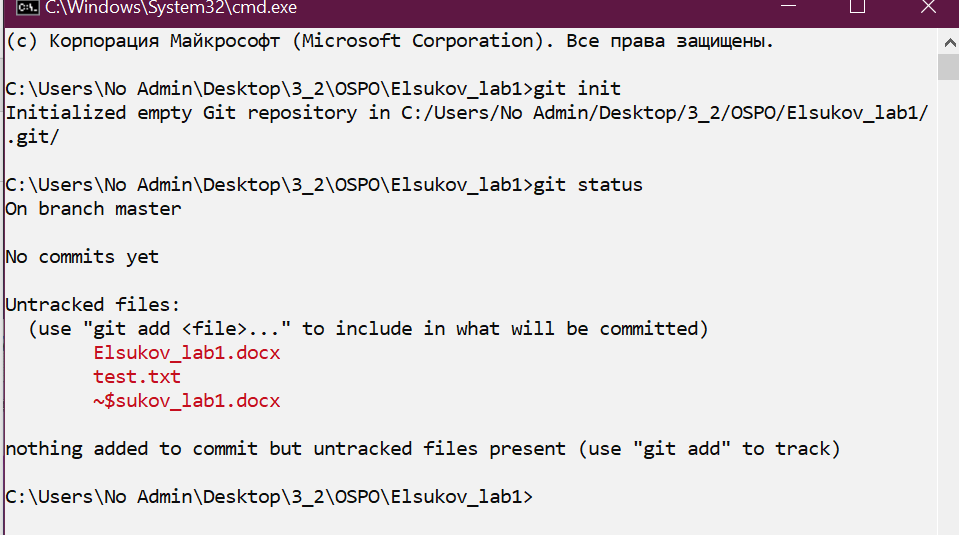
Создадим теперь локальную папку, которая будет связана с нашим облачным git хранилищем. В общем случаем облачный git может представлять также локальный сервер, объединенный в единую информационную сеть со всеми ПК.

Создадим на диске папку со своей фамилие и в ней файлик test.txt с фамилией.

Перейдите в созданную вами папку в командной строке используя команду cd.

Теперь необходимо проинициализировать эту папку как git репозиторий командой git init. Командная строка вернет вам сообщение, что проинициализирована пустая git директория.

Введем команду git status и отобразиться сообщение, что есть новый файл, но он не отслеживается. Это значит, что файл новый и система еще не знает, нужно ли следить за изменениями в файле или его можно просто игнорировать. Для того, чтобы начать отслеживать новый файл, нужно его специальным образом объявить.

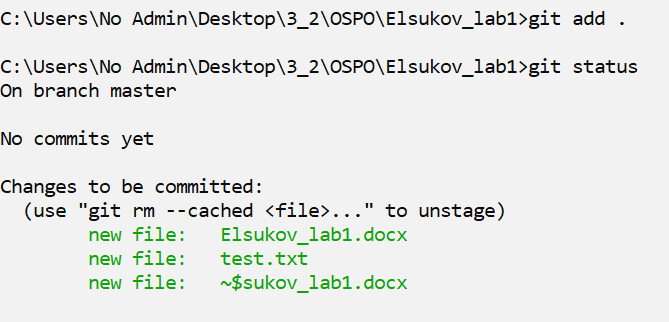


В git есть концепция области подготовленных файлов. Можно представить ее как холст, на который наносят изменения, которые нужны в коммите. Сперва он пустой, но затем мы добавляем на него файлы (или части файлов, или даже одиночные строчки) командой add и, наконец, коммитим все нужное в репозиторий (создаем слепок нужного нам состояния) командой commit.

Фиксация изменений в области заготовленных файлов

Если хотим добавить только один файл, можно написать:git add test.txt Если измененных файлов много, то можно сделать так: git add.

Выполнив нужную команду снова посмотрим статус репозитория:



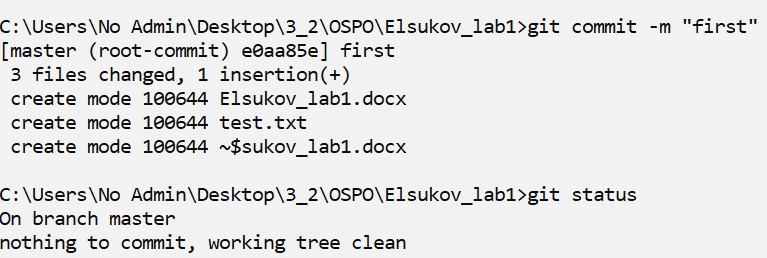
Файл готов к коммиту. Сообщение о состоянии также говорит нам о том, какие изменения относительно файла были проведены в области подготовки — в

данном случае это новый файл, но файлы могут быть модифицированы или удалены.

Теперь закомитим наши файлы. Коммит представляет собой состояние репозитория в определенный момент времени как некий опечаток во времени. Коммит отмечается hash суммой и к которому мы можем в любой момент времени вернуться.

Чтобы зафиксировать изменения, нам нужно хотя бы одно изменение в области подготовки (мы как раз создали новый файл): git commit –m “first”

Команда создаст новый коммит со всеми изменениями из области подготовки. Считается хорошей практикой делать коммиты часто и всегда писать содержательные комментарии. Вызовете git status и посмотрите, что поменялось.



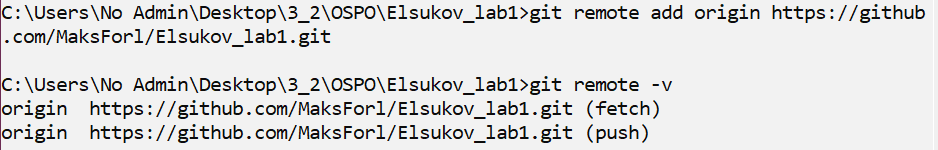
Отправка коминат на сервер

Перед тем как перейти к более продвинутым вещам, рассмотрим, как связать наш локальный репозиторий с облачным репозиторием, который мы создали на github.com

Чтобы связать наш локальный репозиторий с репозиторием на GitHub, выполним следующую команду в терминале.

git remote add origin <https://github.com/MaksForl/Elsukov_lab1.git> Проект может иметь несколько удаленных репозиториев одновременно. Чтобы их различать, мы дадим им разные имена. Обычно главный репозиторий называется origin.

Можно вызвать команду, для просмотра, к какому проекту мы подключены git remote -v



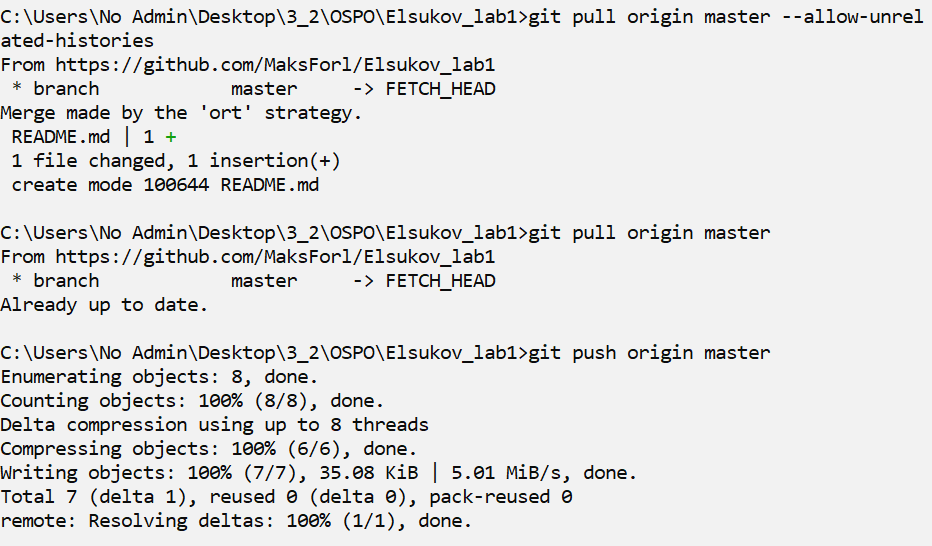
Запросим изменения с сервера

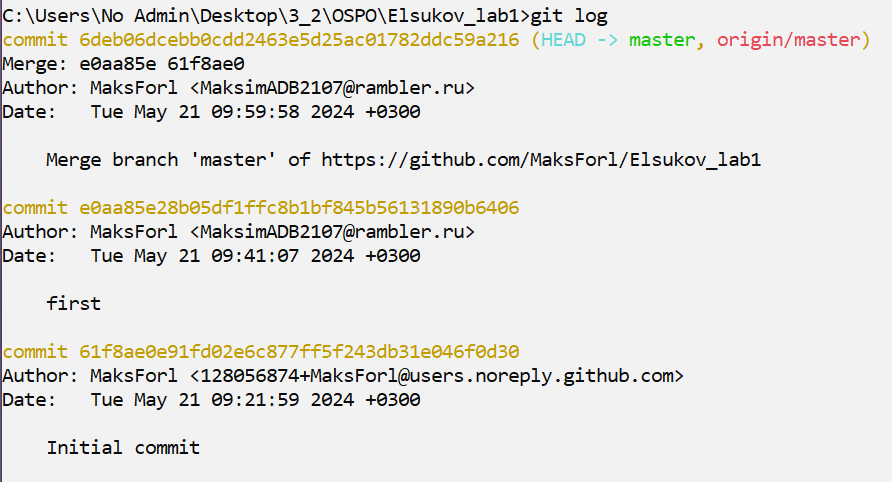
В нашем случае у нас есть файл и на сервере и локальном репозитории, поэтому нам вначале надо получить все данные с нашего репозитория, а потом уже их заливать на сервер нашими локальными данными.

git pull origin master (*--*allow-unrelated-histories)

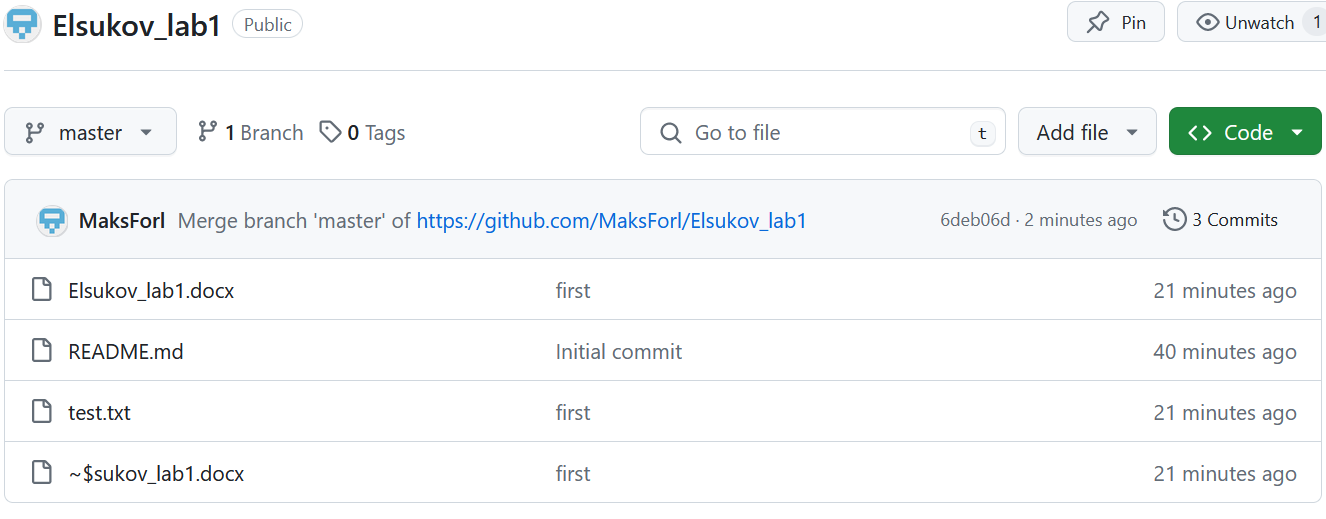
Теперь отравим коммит на сервер, команда, предназначенная для этого — push. Она принимает два параметра: имя удаленного репозитория (мы назвали наш origin) и ветку, в которую необходимо внести изменения (master — это ветка по умолчанию для всех репозиториев).

git push origin master



Посмотрим наши изменения с использованием команды 

Посмотрим изменения на github.com



Создание новой ветки

Во время разработки новой функциональности считается хорошей практикой работать с копией оригинального проекта, которую называют веткой. Ветви имеют свою собственную историю и изолированные друг от друга изменения до тех пор, пока вы не решаете слить изменения вместе. Это происходит по набору причин:

• Уже рабочая, стабильная версия кода сохраняется.• Различные новые функции могут разрабатываться параллельно разными программистами.• Разработчики могут работать с собственными ветками без риска, что кодовая база поменяется из-за чужих изменений.• В случае сомнений, различные реализации одной и той же идеи могут быть разработаны в разных ветках и затем сравниваться.

Обычно в проекте создается минимум 2 ветки (branch) – master и ветка для разработки. Ветка master создается сама, а дополнительные ветки создаются по желанию пользователя (локально), заливаются они в удаленный репозиторий уже с учетом доступных прав.

Создадим новую ветку second:

git branch second

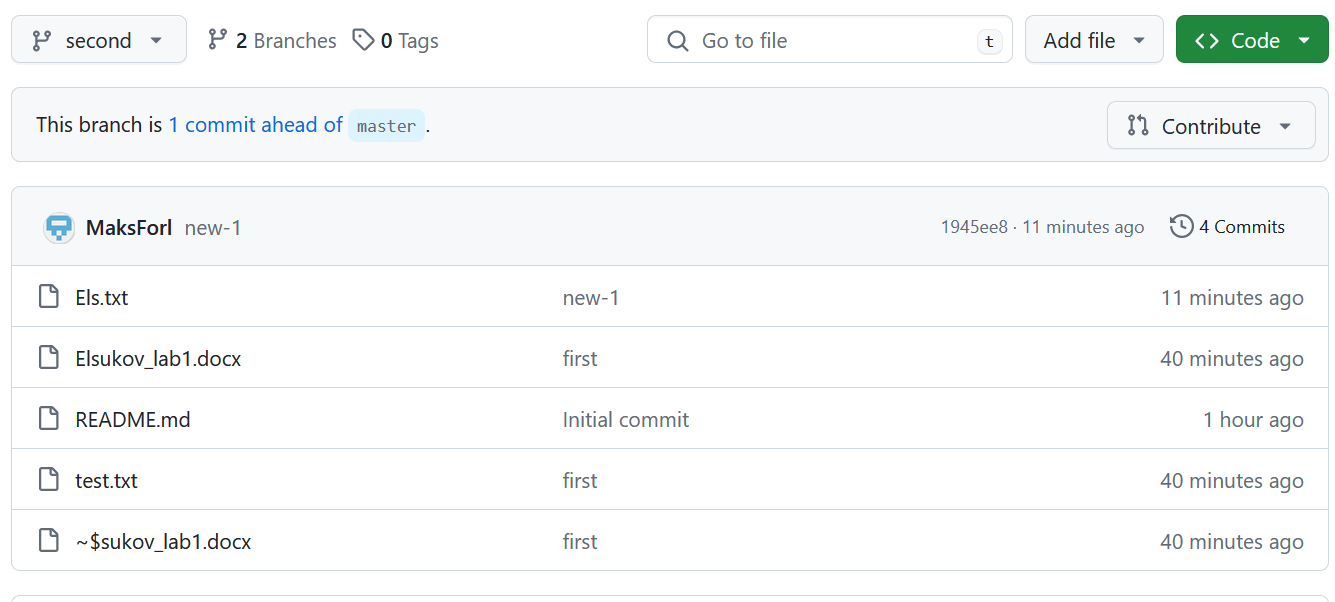
Можно посмотреть в какой ветки находимся, используя команду

git branch

Переключимся на другую ветку, используя команду

git checkout second

Создадим новый файл в нашем локальном репозитории и напишем в нем свою фамилию, добавим в область подготовленных файлов и отправим на сервер.



Слияние веток

Переключитесь снова на ветку master. Если через проводник windows в папку с проектом, то файла test\_new.txt не будет виден, потому что мы переключились обратно на ветку master, в которой такого файла не существует. Чтобы он появился, нужно воспользоваться merge для объединения веток (применения изменений из ветки second к основной версии проекта).

git merge second

Если все прошло без ошибок, можно удалить ветку

git branch -d second

Давайте немного усложним это задание, предположив, что в двух ветках могут быть одинаковые файлы и над ними работают разные разработчики

Создайте ветку с названием newdev. Переключитесь на нее

Добавьте в файл с именем отчество. Зафиксируем изменения. Закомитьте изменения

Попробуйте переключиться обратно на ветку master и посмотреть файл: в нем только имя, отчество в другой ветке

Объединим ветки git merge newdev

git merge newdev

Откройте файл снова, там есть и имя, и отчество. Переключитесь на ветку newdev и удалите в отчестве несколько букв Зафиксируйте, закоммите

Переключимся на ветку master, добавьте к отчеству несколько букв Зафиксируйте, закоммите

Объединим ветки git merge newdev

Теперь ничего не получится, т.к. есть изменения в обеих ветках