

**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»

Дисциплина «Основы системного программного обеспечения»

**Отчет по лабораторной работе №1**

Выполнил:

студент гр. АДБ-21-08 \_\_\_\_\_\_\_\_\_\_\_\_ Герасимов Д.С.

(дата) (подпись)

Принял:

к.т.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_ Ковалев И.А.

(дата) (подпись)

**Москва 2024**

**Цель работы:** получение навыком работы с системой контроля версий на примере GitHub.

Создадим в папке со своей фамилией текстовый файл test.

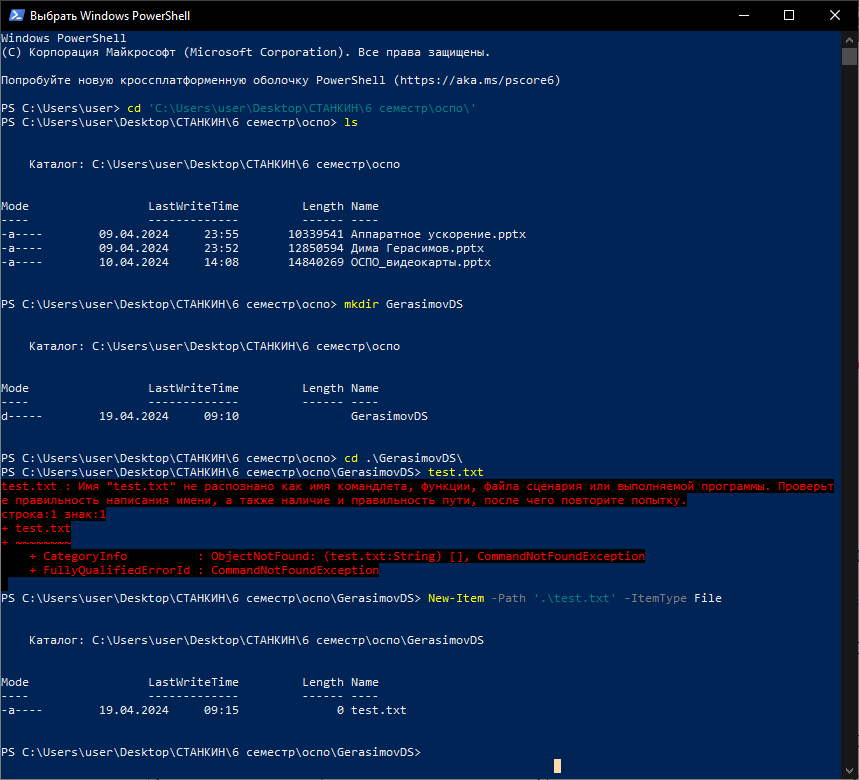


Рис. 1. Создание файла в PowerShell.

С помощью команды cd перейдем в нужный каталог, затем проинициализируем его как git репозиторий. После этого проверим состояние рабочего каталога и увидим, что есть новый файл, но он не отслеживается.

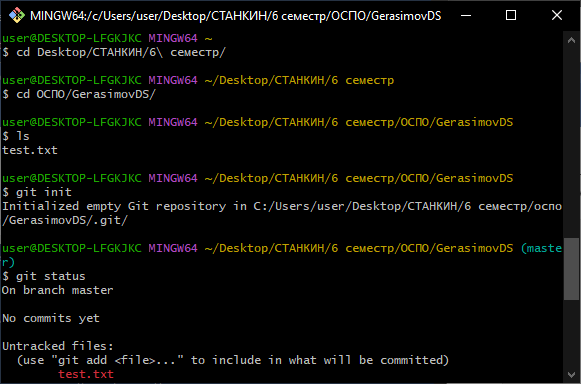


Рис. 2. Работа в консоли Git Bash.

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.

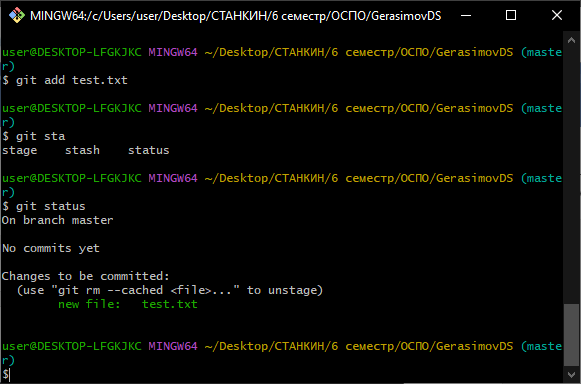


Рис. 3. Добавление в область подготовленных файлов.

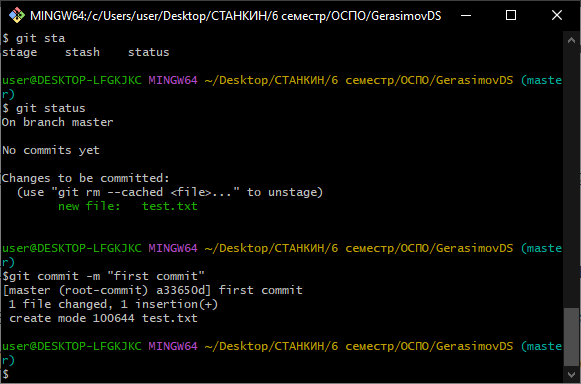


Рис. 4. Коммит файла.

Свяжем наш локальный репозиторий с репозиторием на GitHub.

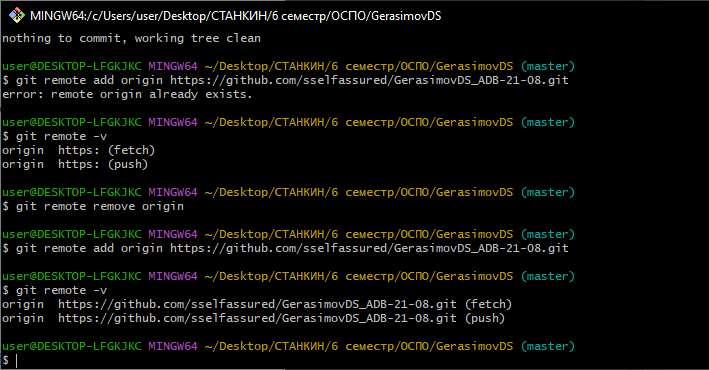


Рис. 5. Связывание локального и удалённого репозитория.

Вначале надо получить все данные с нашего репозитория при помощи команды pull, а потом уже их заливать на сервер нашими локальными данными.

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)

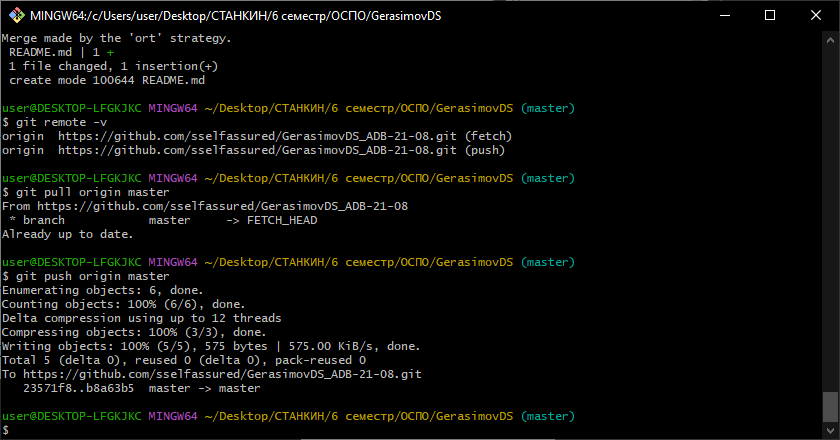


Рис. 6. Запрос изменений с сервера и отправка локального коммита на сервер.

Посмотрим наши изменения с использованием команды git log.

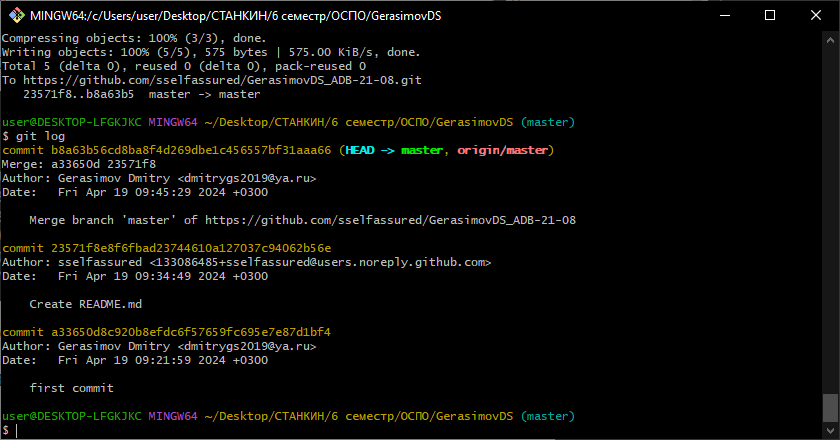


Рис. 7. Лог действий с репозиторием.

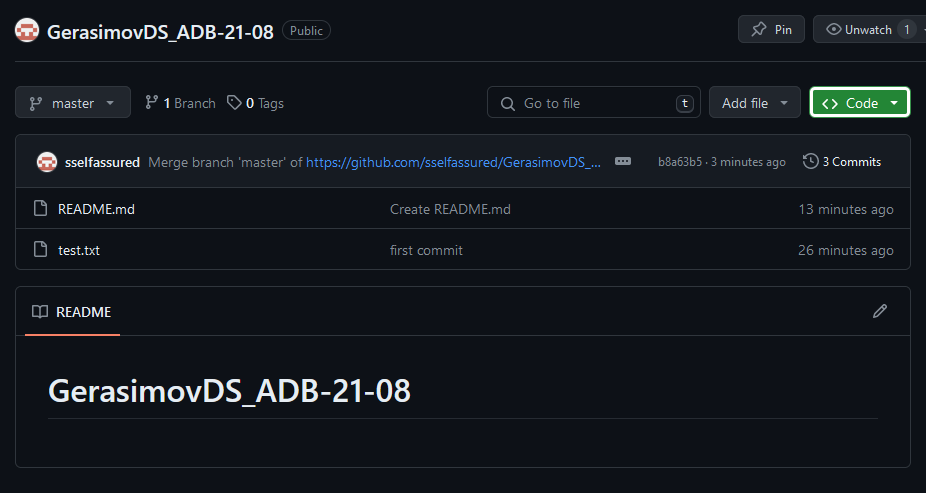


Рис. 8. Изменения на github.

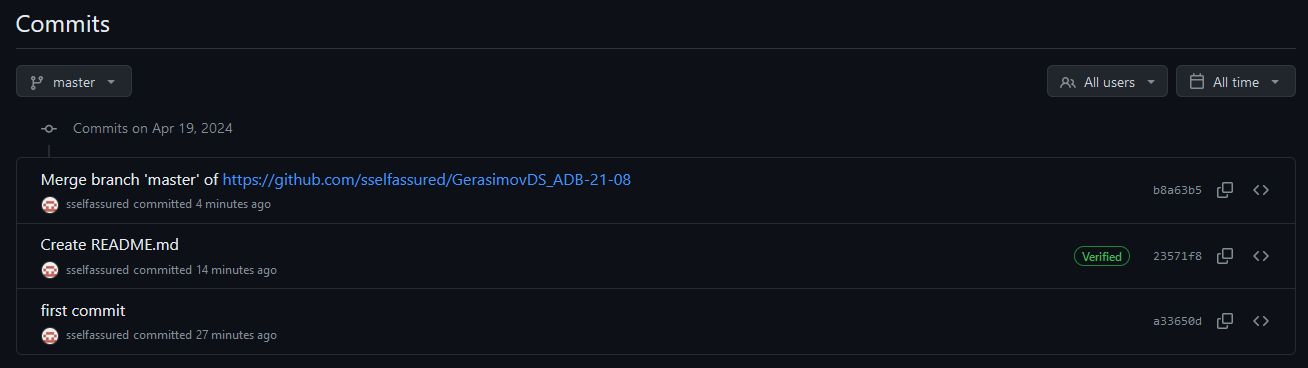


Рис. 9. История изменений на github.

Обычно в проекте создается минимум 2 ветки (branch) – master и ветка для разработки. Ветка master создается сам, 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

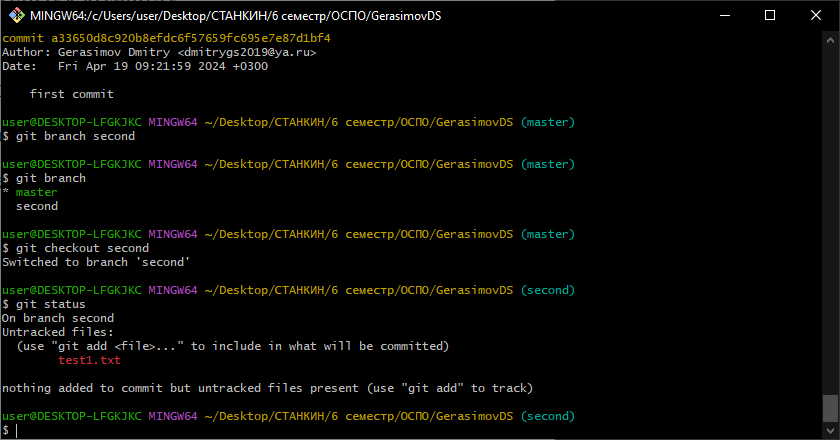


Рис. 10. Создание новой ветки и переключение на нее.

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

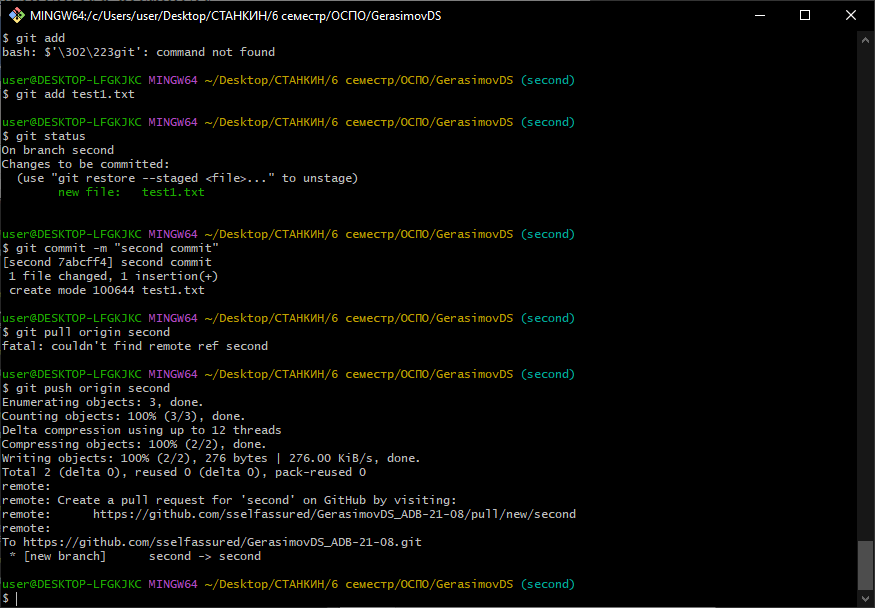


Рис. 11. Коммит нового файла в другую ветку.

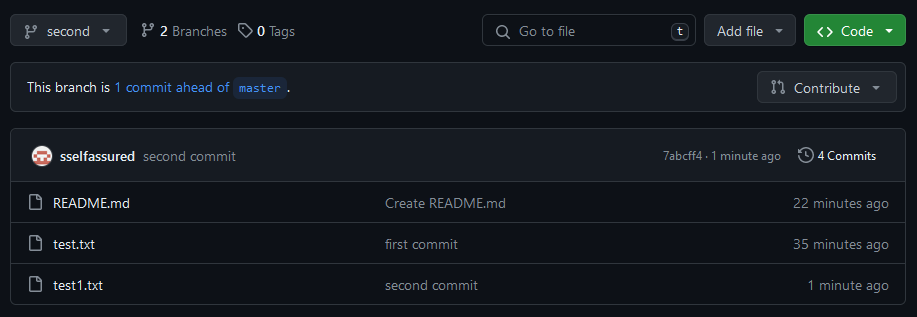


Рис. 12. Присутствие нового файла в ветке second.

Если через проводник windows в папку с проектом, то файла test1.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

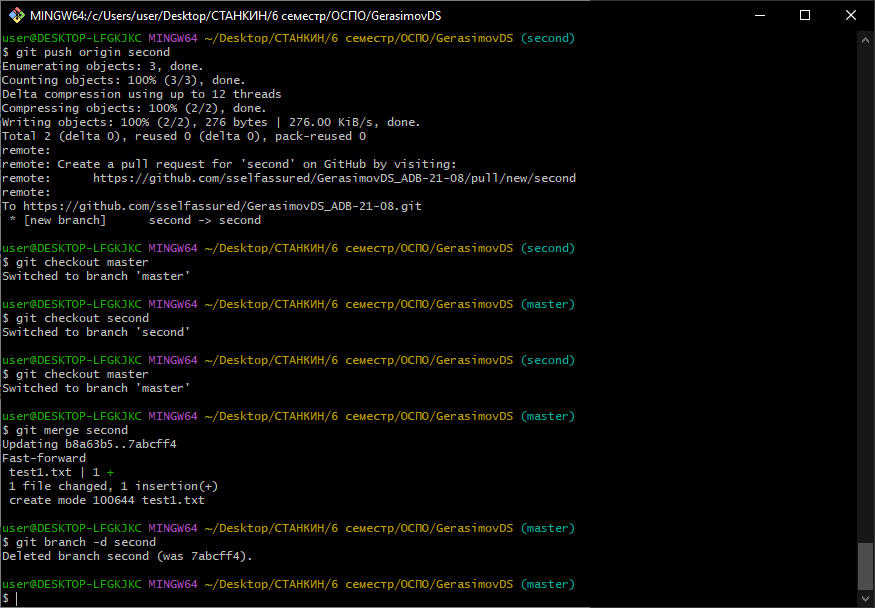


Рис. 13. Объединение веток и удаление ветки second.

Предположим, что в двух ветках могут быть одинаковые файлы и над ними работают разные разработчики. Создадим ветку с названием newdev и переключимся на нее. Добавим в файл с именем отчество. После коммита вернемся в master и объединим ветки. Далее переключимся между этими ветками, внеся изменения в файл и попытаемся объединить ветки.

Наберем команду, для просмотра изменений - git diff

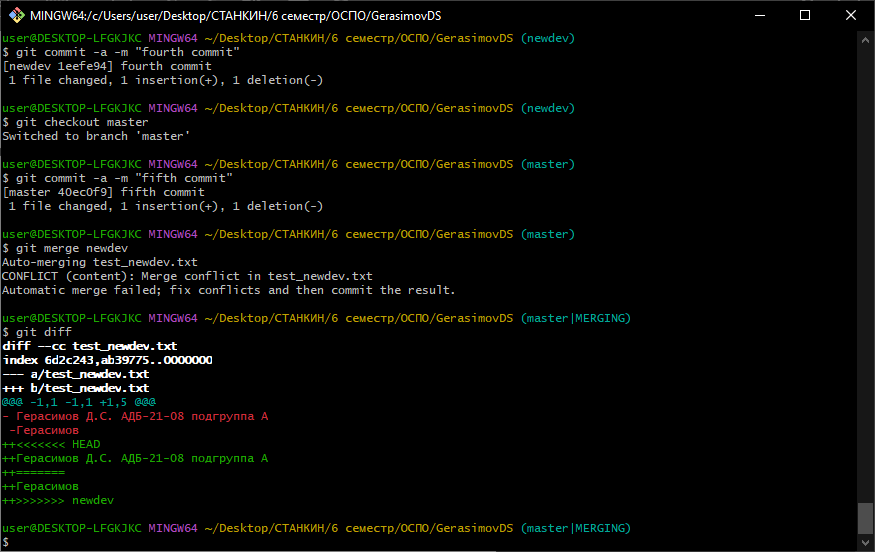


Рис. 14. Неудачное слияние веток.

Выбираем нужную строку, удаляем разделители, сохраняем файл, фиксируем изменения, коммитим, отправляем на сервер и удаляем ветку newdev.

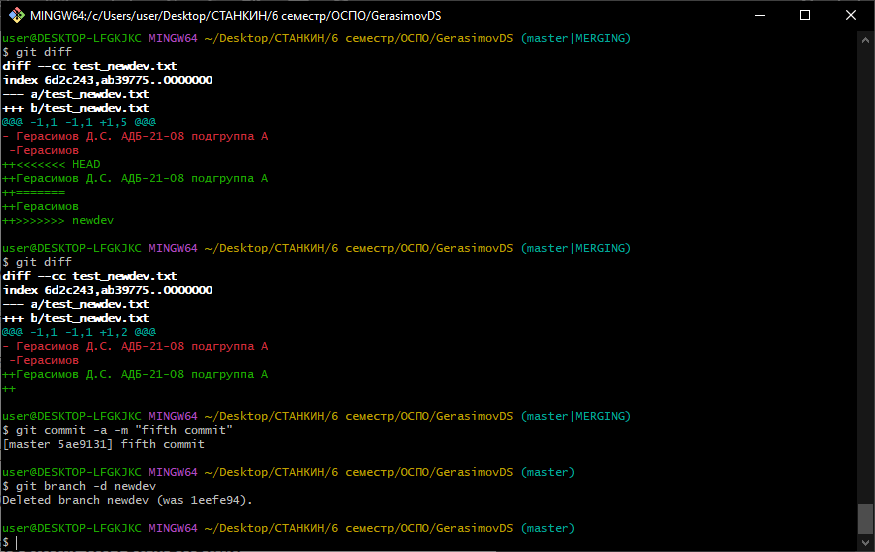


Рис. 15. Разрешение конфликта и удаление ветки newdev.

Если посмотреть на github, то наши ветки там остались. Удалим их из github:

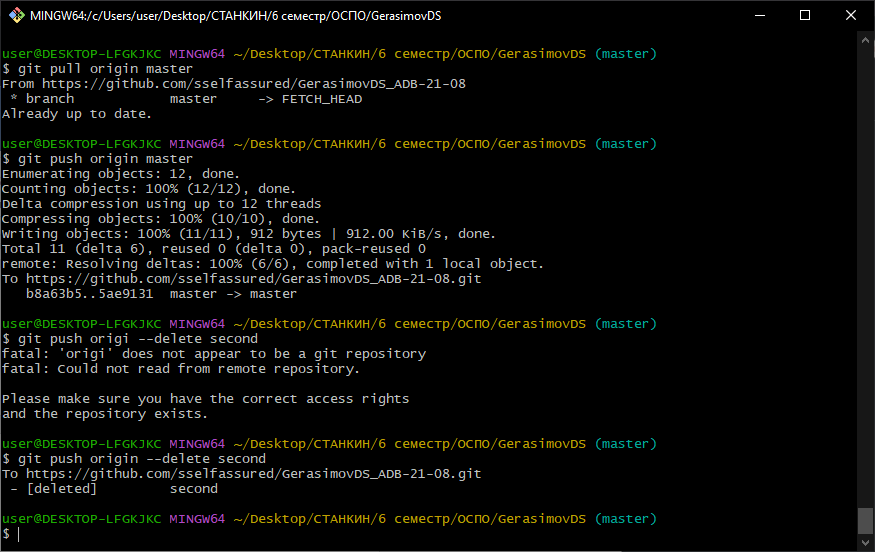


Рис. 16. Удаление веток из github.

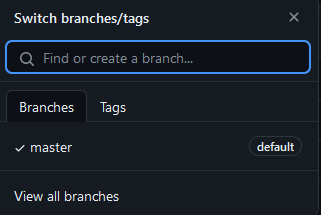


Рис. 17. Результат удаления, осталась лишь ветка master.

Гит позволяет вернуть выбранный файл к состоянию на момент определенного коммита. Это делается уже знакомой нам командой checkout, которую мы ранее использовали для переключения между ветками. Но она также может быть использована для переключения между коммитами.

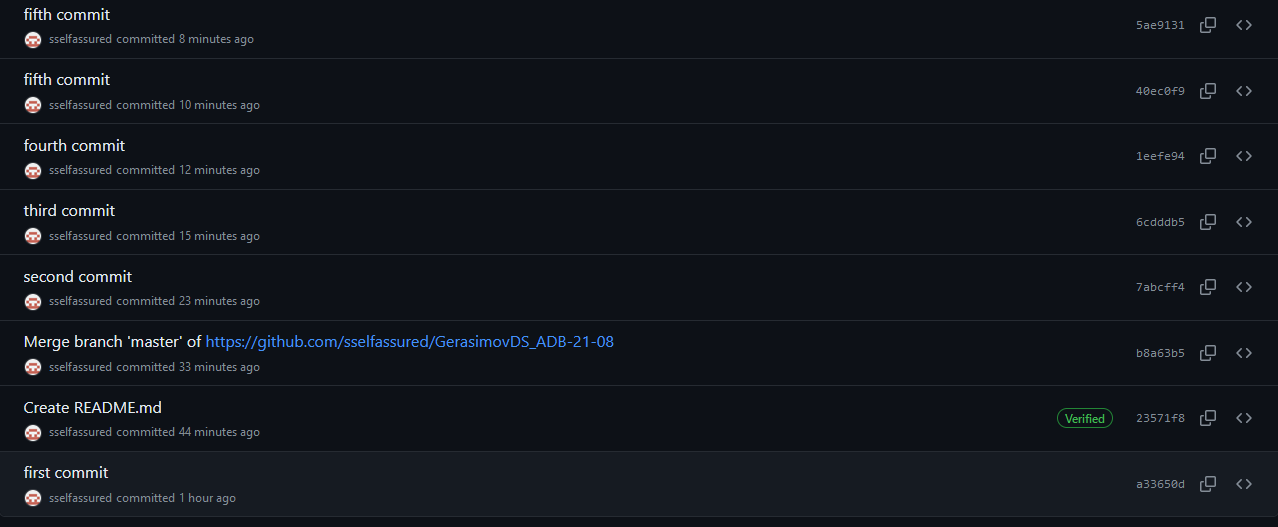


Рис. 18. История коммитов.

Откатимся к second commit.

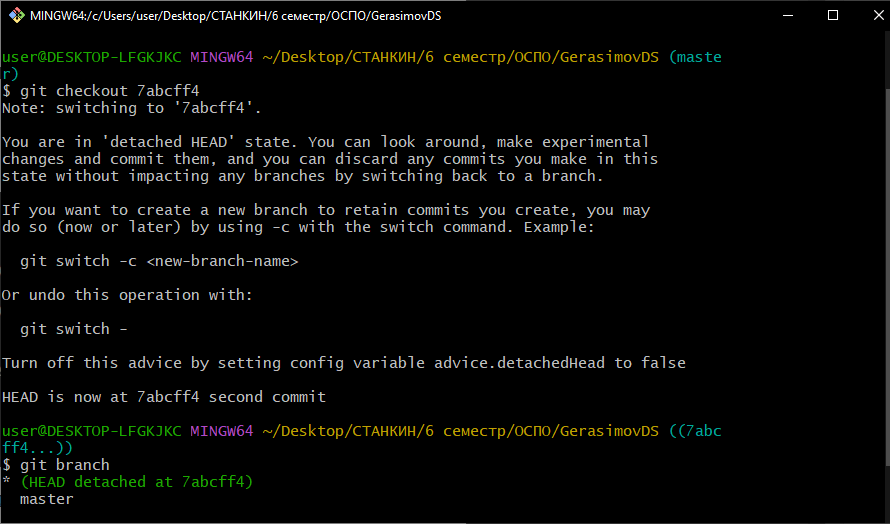


Рис. 19. Создание псевдо-ветки, начинающейся на этом коммите.

Возможна ситуация, что вы закрепили файлы, но еще не коммитили, хотите убрать файлы из области закрепления, вызовите команду:

git reset HEAD

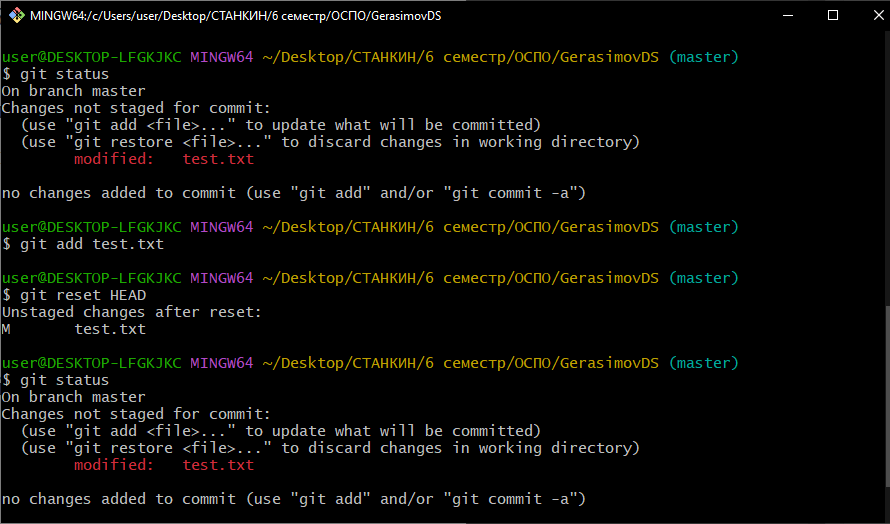


Рис. 20. Удаление файла из области закрепления.

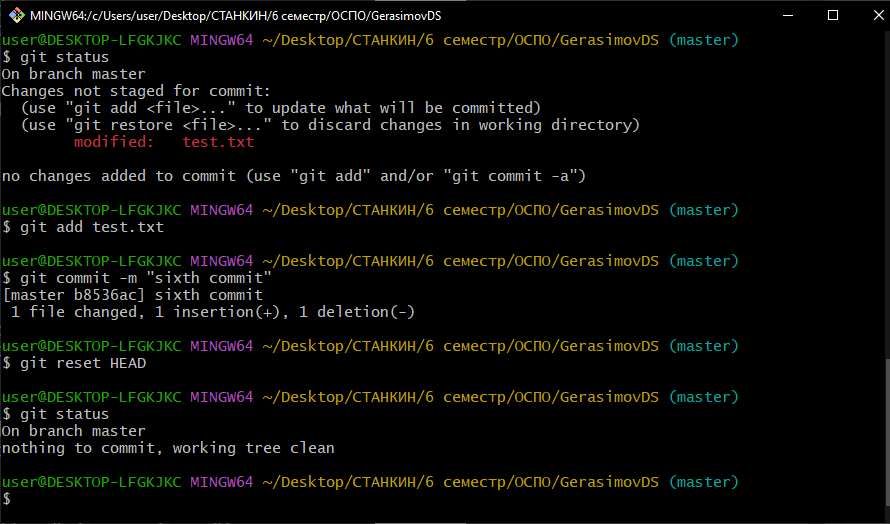


Рис. 21. Последующий коммит файла.

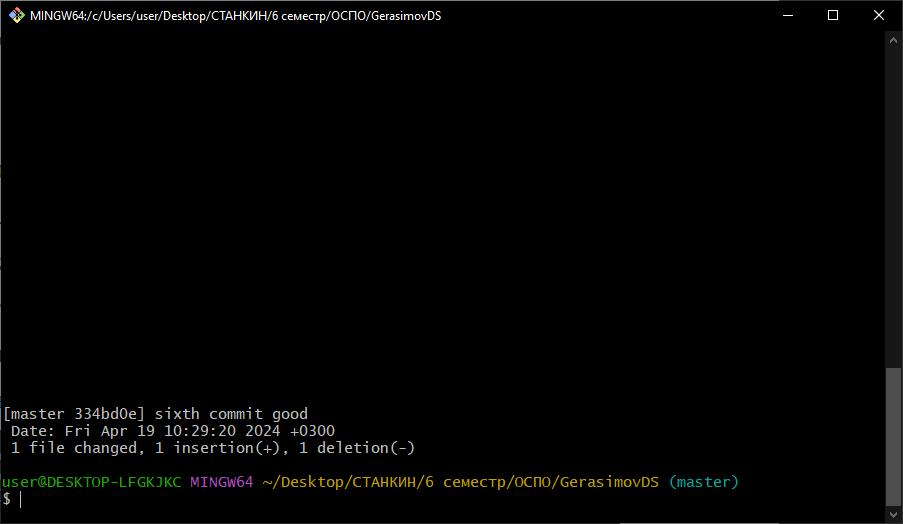


Рис. 22. Изменение комментария к коммиту.

В большинстве проектов есть файлы или целые директории, в которые мы не хотим (и, скорее всего, не захотим) коммитить. Мы можем удостовериться, что они случайно не попадут в git add -A при помощи файла «.gitignore»

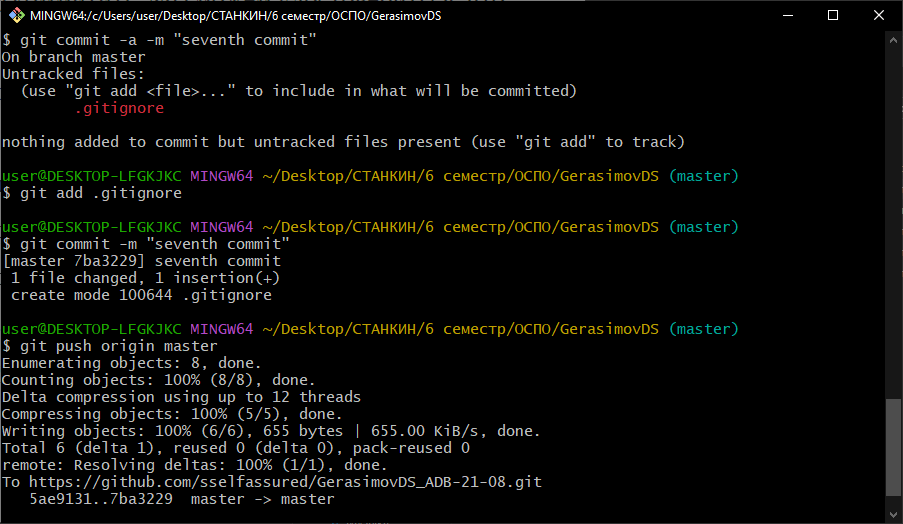


Рис. 23. Отправка на сервер файла .gitignore.

Клонируем репозиторий напарника на локальный компьютер, предварительно создав папку, в которую будете клонировать и в консоли перейдя в нее

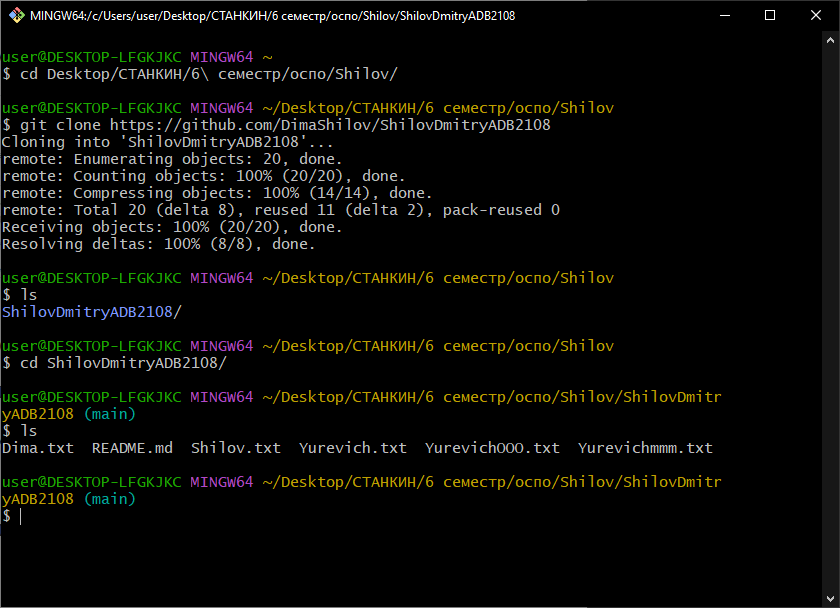


Рис. 24. Клонирование репозитория напарника.

Поработаем над одним файлом. Добавим новый файл, отправим его на сервер, чтобы, когда все разработчики сделают pull себе он появился локально у них. Сделаем push нескольких новых файлов.

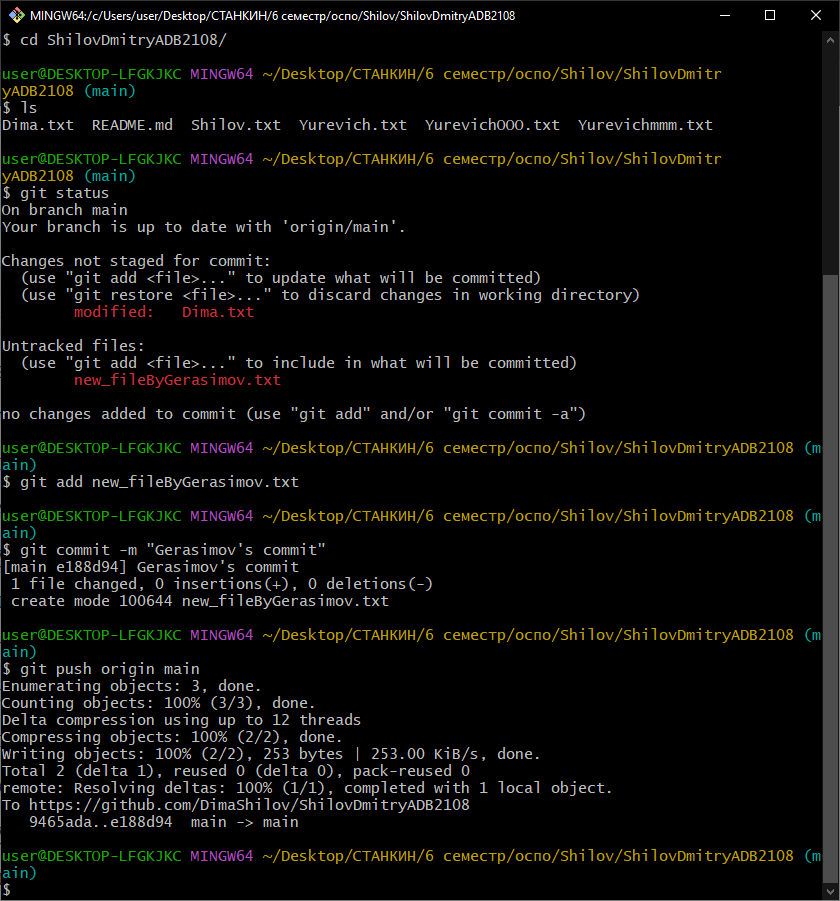


Рис. 25. Отправка на сервер нового файла.

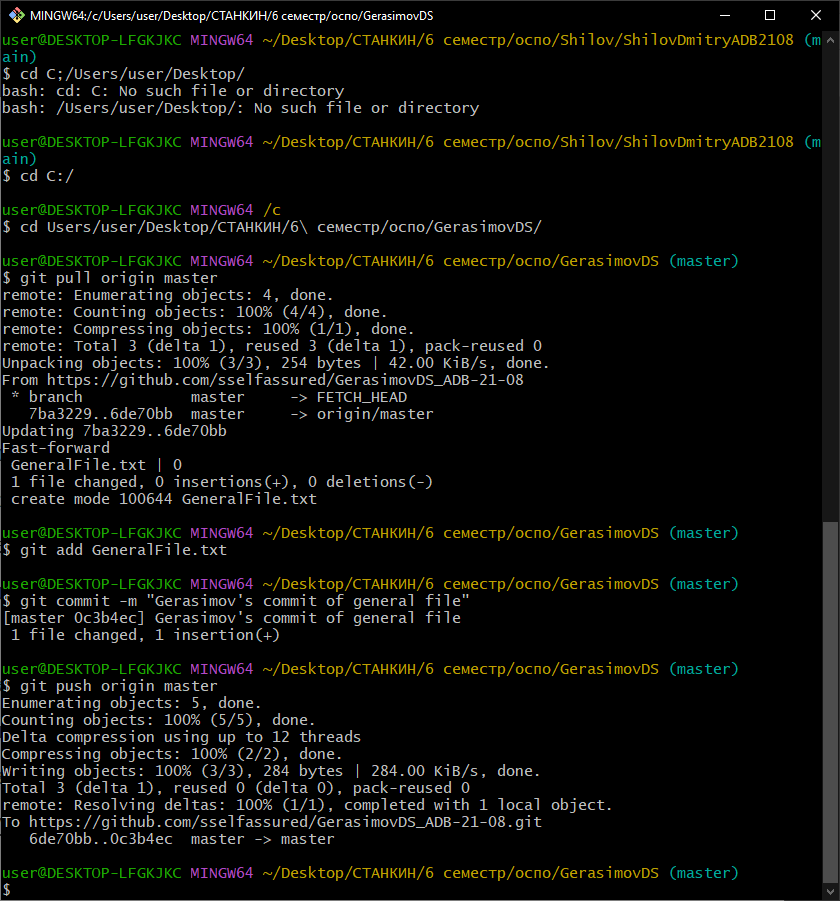


Рис. 26. Отправка на сервер отредактированного файла.

**Вывод**

В ходе лабораторной работы были получены навыки по работе с системой контроля версий Git и синхронизацией локального и удаленного репозитория.