**Лабораторная работа №1**

**Выполнил:** Студент гр. 453501 Сацюк Степан вариант 24

**Тема:** Система контроля версий Git.

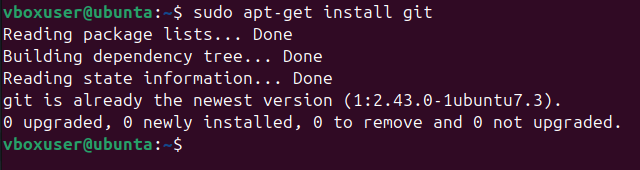
**Цель:** Познакомиться с возможностями и получить практические навыки работы с системой контроля версий Git.

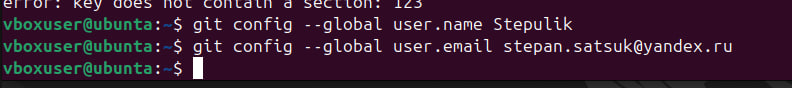
**Выполнение задания:**

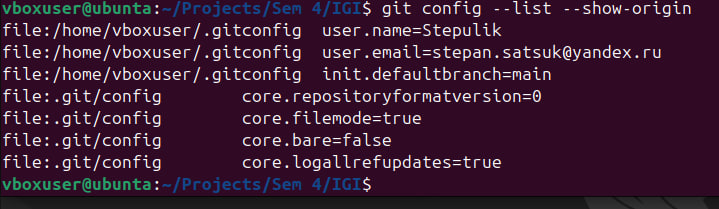
Подготовьте рабочее окружение в соответствии с типом вашей операционной системы(https://smartiqa.ru/courses/git/lesson-1 ):

1. **a.** Установите Git (https://git-scm.com/downloads )

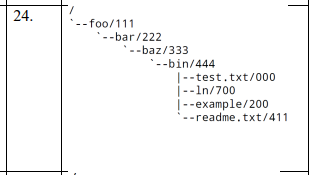
**Установка была выполнена на Ubuntu**



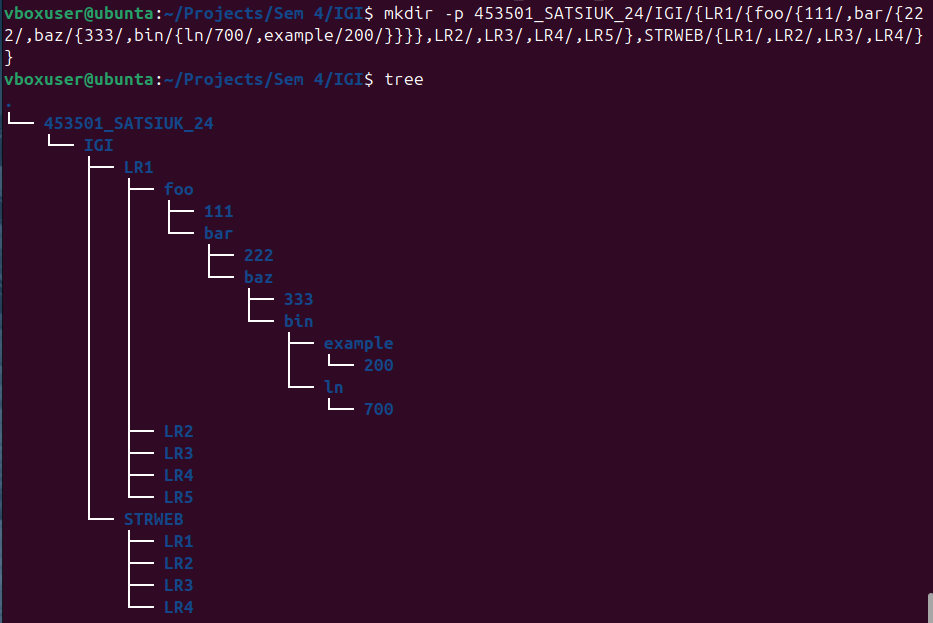
**b.** Выполните базовую настройку

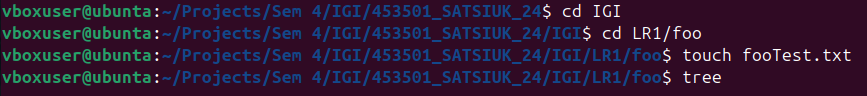


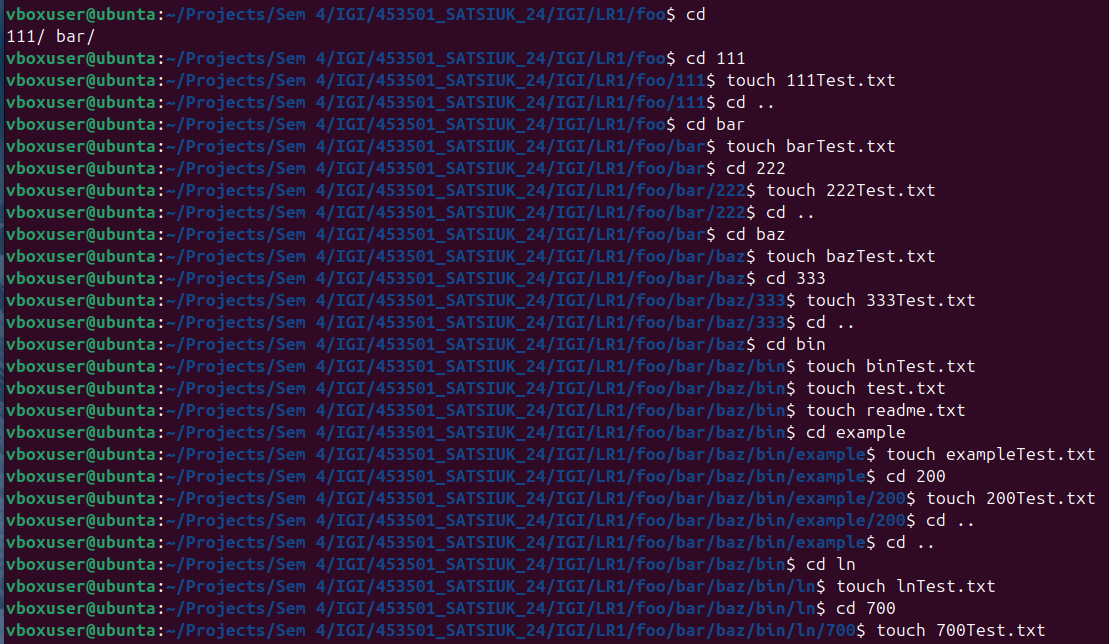
**2)** Создайте структуру папок (использовать командную строку) и поместите в репозиторий Git на своем локальном компьютере:

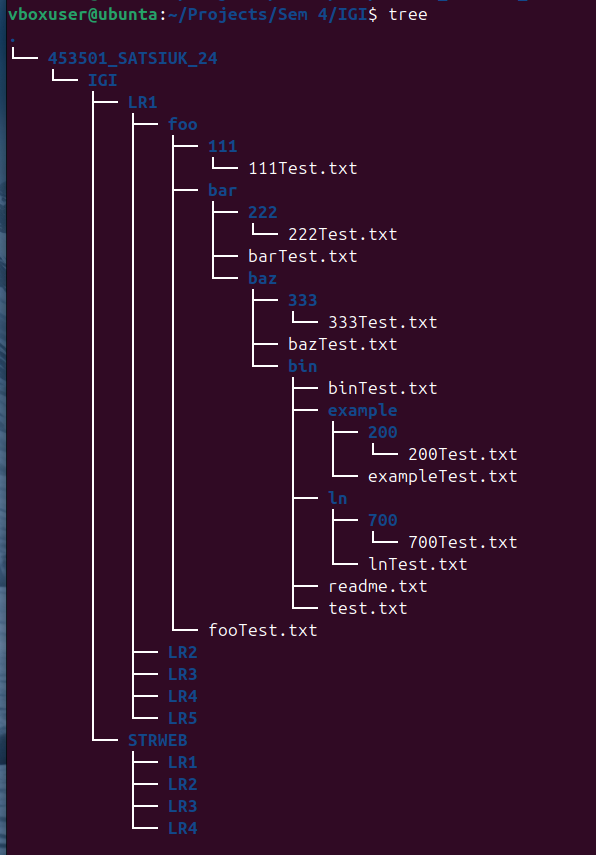


mkdir –p позволяет создавать сразу несколько директорий, это в разы быстрее чем создавать каждую директорию отдельной командой, то что на скрине папки LR2-5 в папке LR1 это я поправил, так же убрал STRWEB из папки IGI

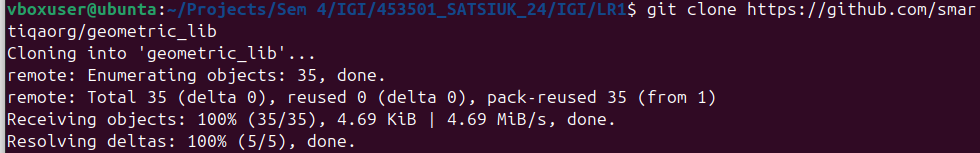








**3)** Создайте репозиторий внутри папки LR1 с помощью клонирования проекта с сайта github.com (<https://github.com/smartiqaorg/geometric_lib>)





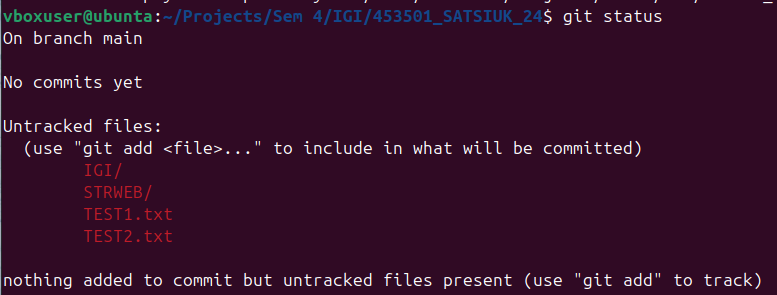
На всякий случай удалю .git папку из клонированного репозитория. Потому что консоль иногда на него ругается, да и в целом она там не сильно нужна.

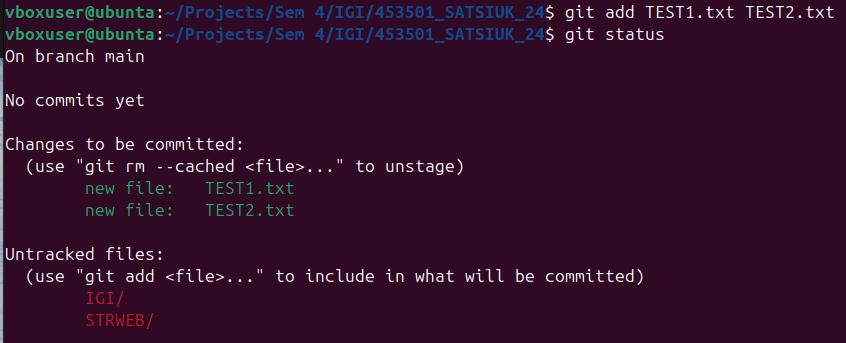


**4)** Выполните следующие действия, каждый раз проверяя состояние файлов в локальном репозитории до и после выполнения операции:

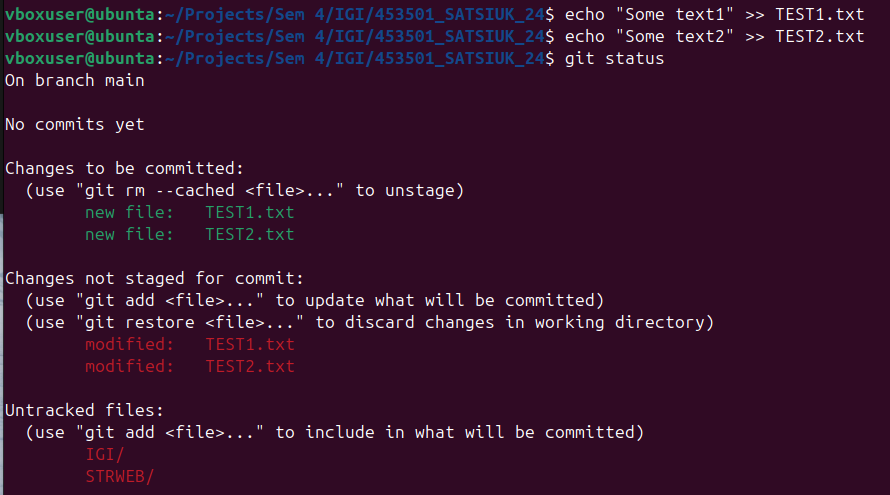
1. сделайте индексацию нескольких файлов в вашем Git-репозитории (git add …),

заранее создал файлы TEST1.txt и TEST2.txt

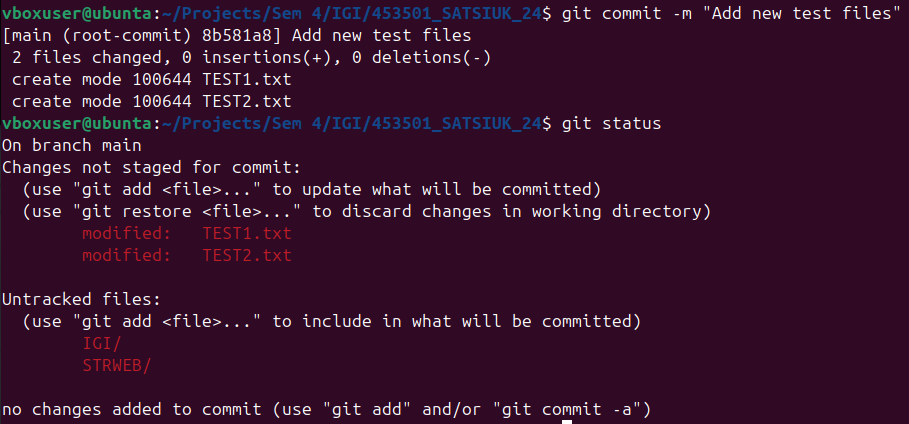




**b.** внесите изменения в файлы,



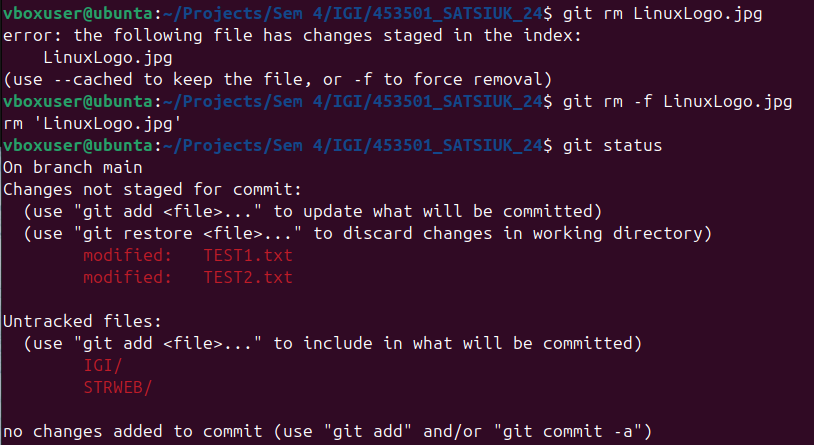
1. выполните коммиты к проиндексированным файлам,



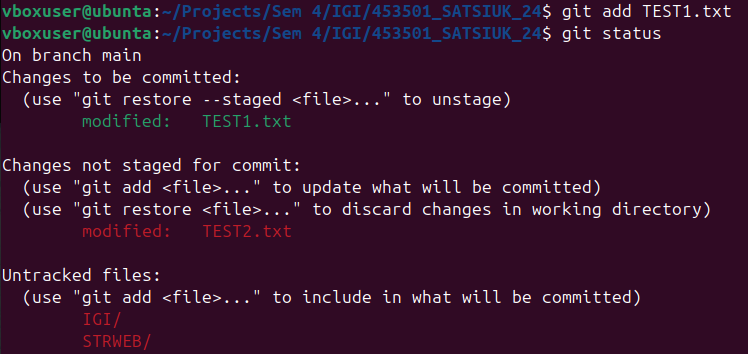
1. добавьте и удалите файлы (один и группу файлов с определенным расширением)

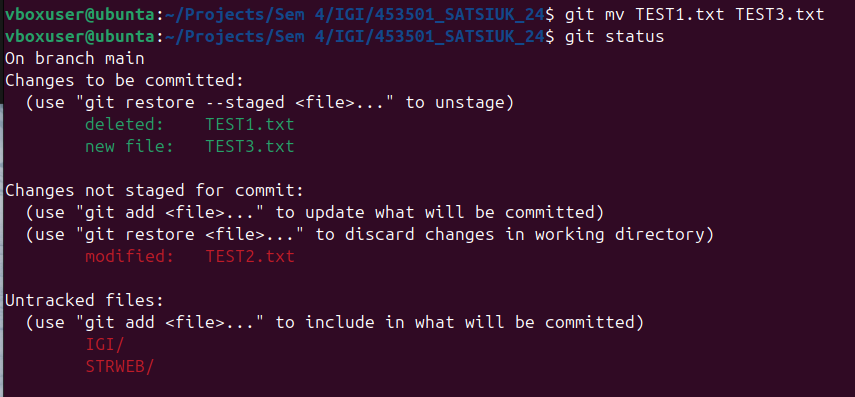
предварительно закинул в проект файл LinuxLogo.jpg



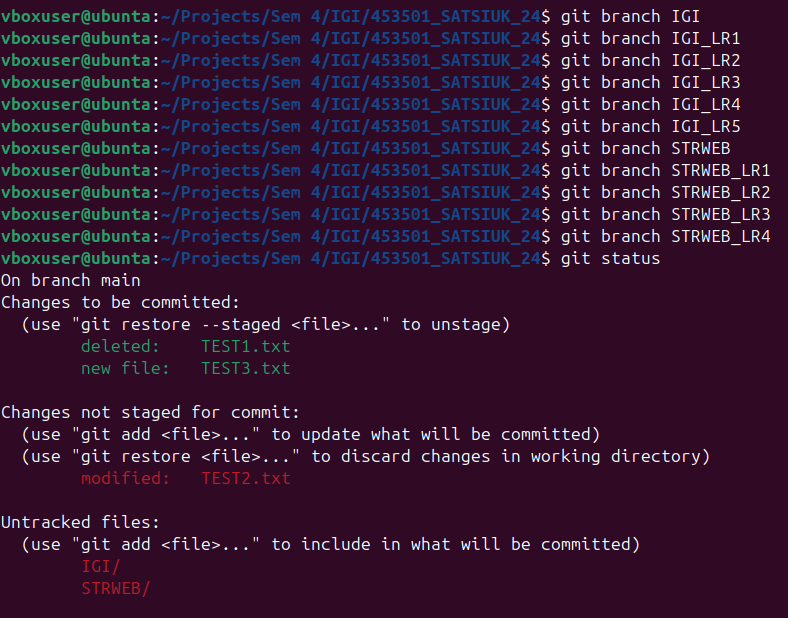
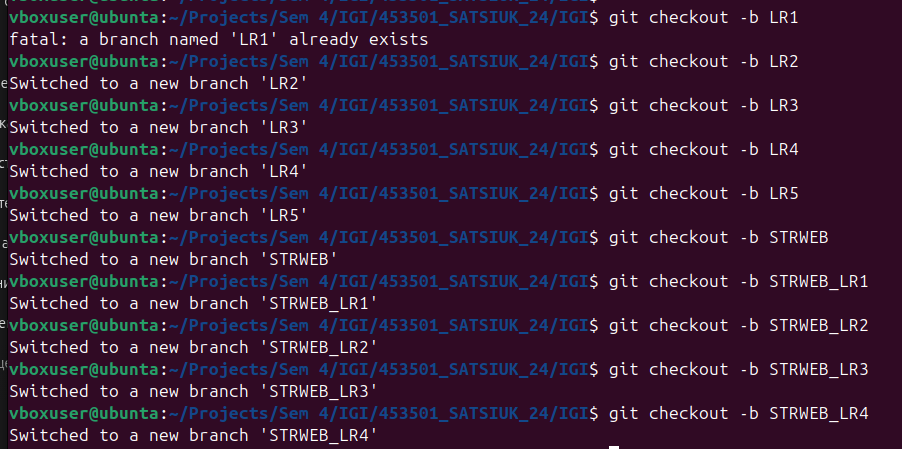


**e.** переименуйте файлы

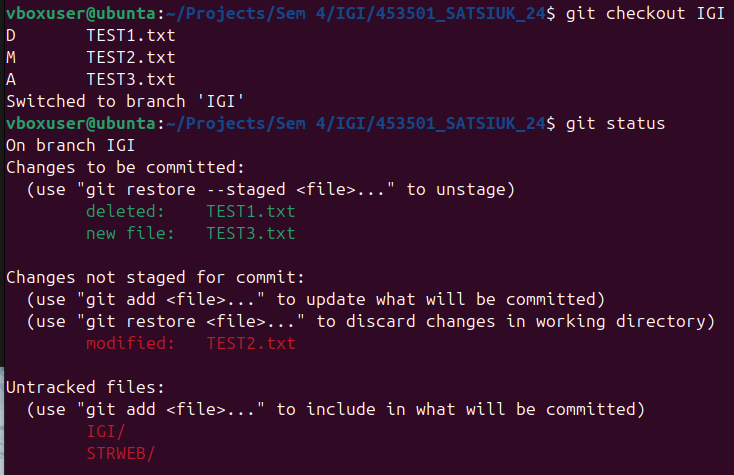


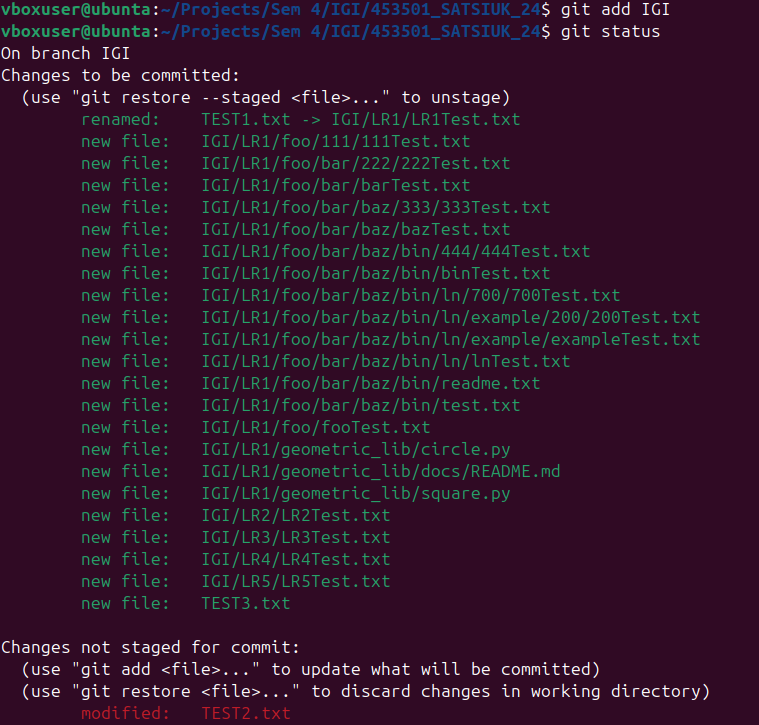


**f.** создайте для IGI и STRWEB, а также для каждой ЛР, вложенной в них, отдельные ветки, поместите файлы на соответствующие ветки

Были созданы ветки для всех папок и для папок LR отдельные

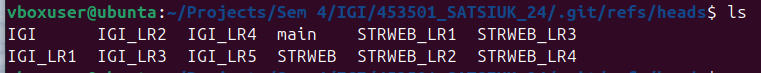
Добавим для примера файлы из папки IGI в ветку IGI, можно использовать как switch так и checkout





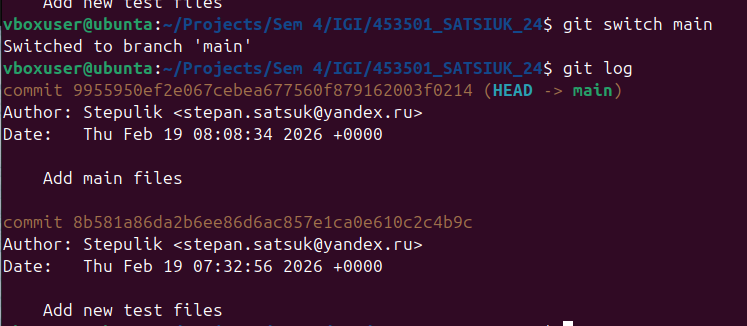


Для остальных веток создаем коммиты по аналогии. При этом предварительно создаем резервную копию всех файлов, так как при коммите папки в ветку она при смене ветки пропадает. В итого получается грамотно разветвленное дерево с ветками, где каждая из них соответствует своим папкам

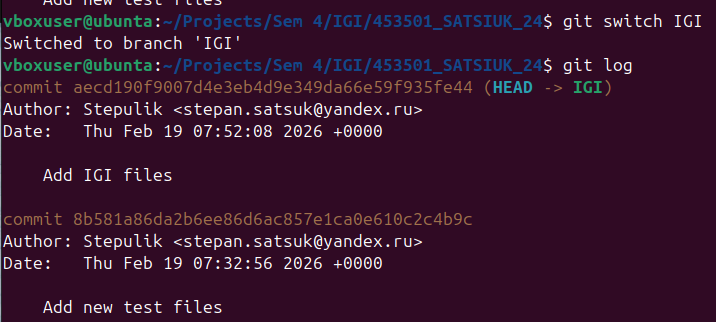


**g.** выведите историю коммитов: полностью и с ограничением вывода – отобрать только последние, до и после указанной даты, по автору файлов, коммитов (флаги n, since, after, until, before, author, committer, grep) https://code.mu/ru/tool/git/basis/limit-output-commit-history-flags/ ,

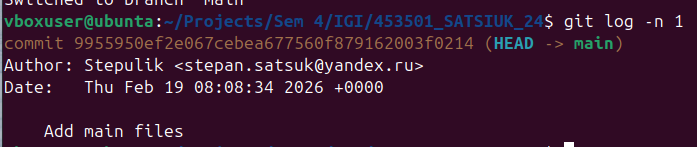
Для веточки main



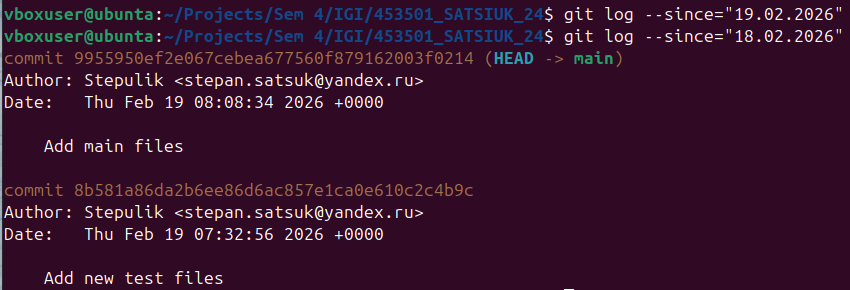
Для веточки IGI (остальные по аналогии)

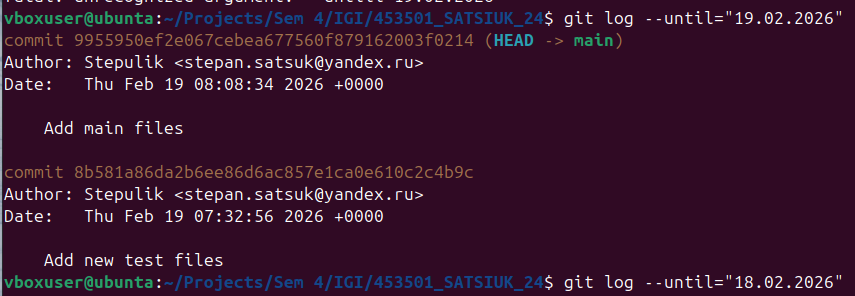


Флаг --n ограничивает показ до k последних логов (-n 1) или k последних коммитов (-n k), где k – число последних коммитов для отображения иформации.

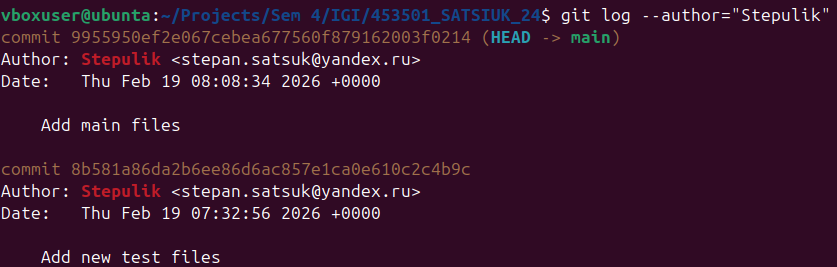


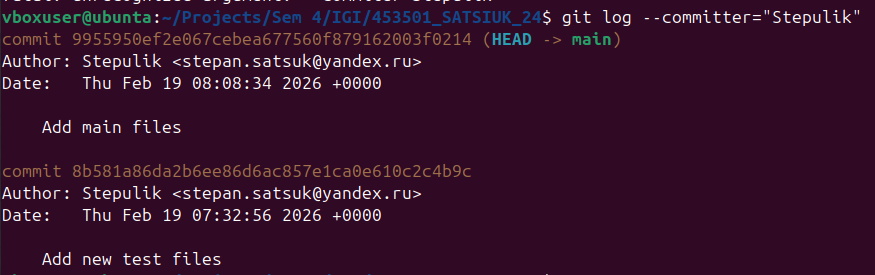
С помощью –since(--after), заметно, что он НЕ ВКЛЮЧАЕТ в саму введенную дату а отсчитывает только после



Флаг --until(--before), он ВКЛЮЧАЕТ саму дату

Флаг --autor ищет коммиты по автору, --commitеer – по тому кто добавил коммит в репозиторий (иногда автор и тот кто залил коммит это разные люди)

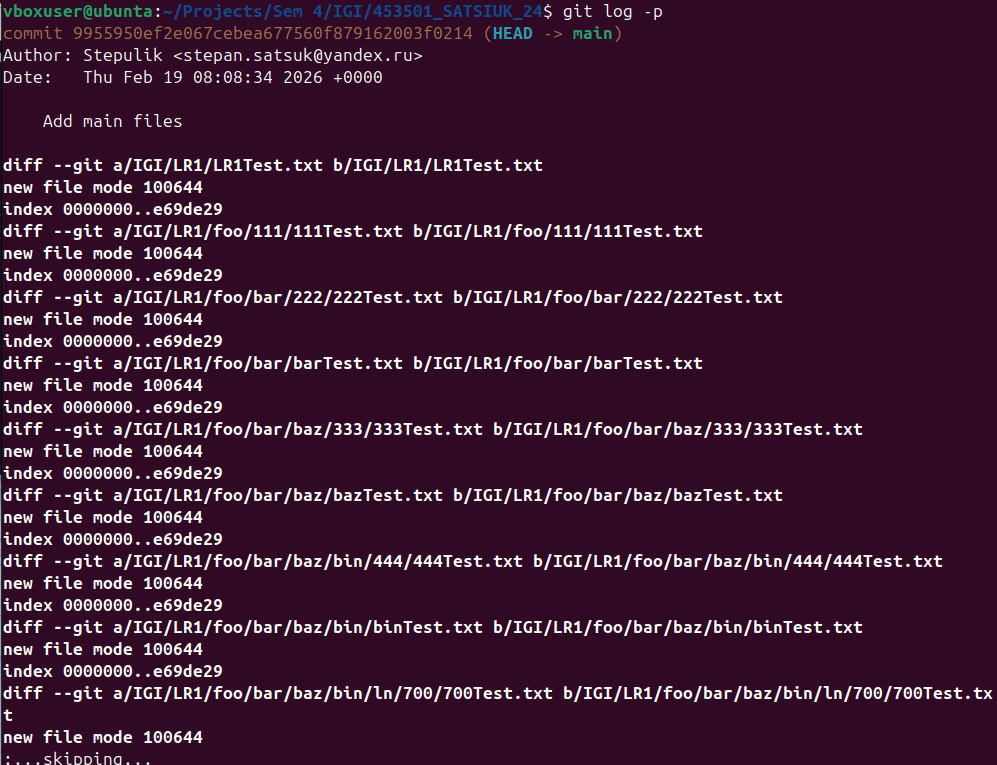




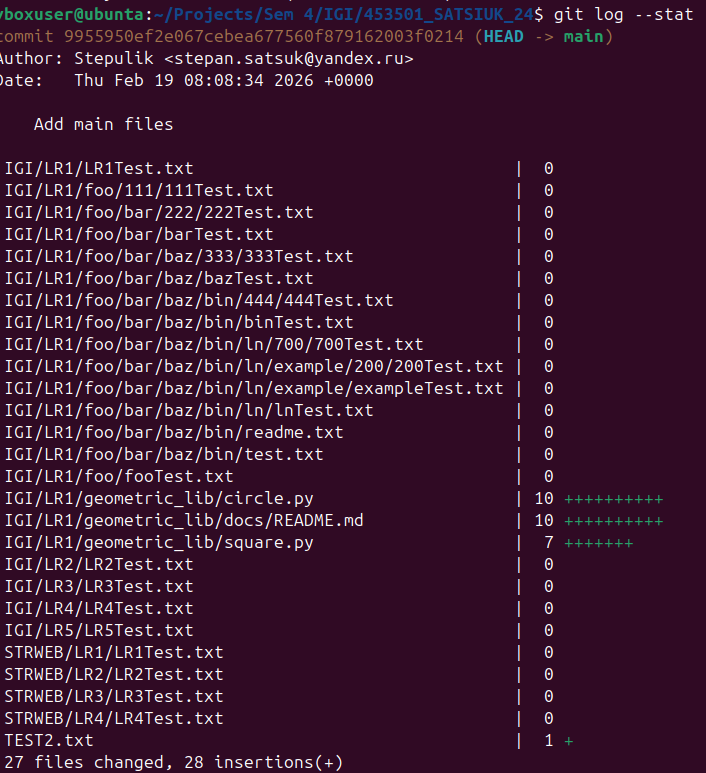
Флаг --grep ищет коммит по его описанию (которое задается при коммите)

**h.** исследуйте флаги для git log: p , stat ,shortstat, name-only, name-status, relative-date, pretty, graph,

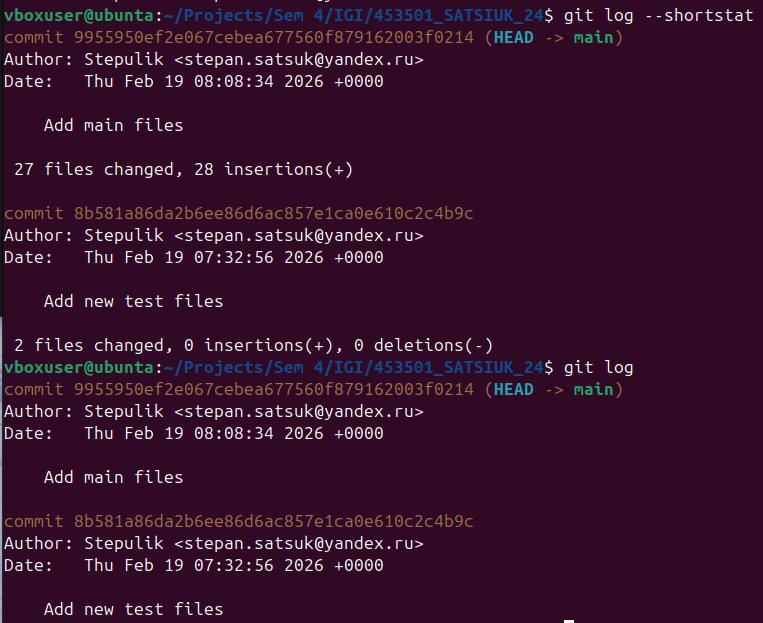
флаг -p показывает что именно изменилось после каждого коммита, можно также ограничить кол-во коммитов до нескольких последних (-p -2) чтобы выйти из окна просмотра надо нажать q



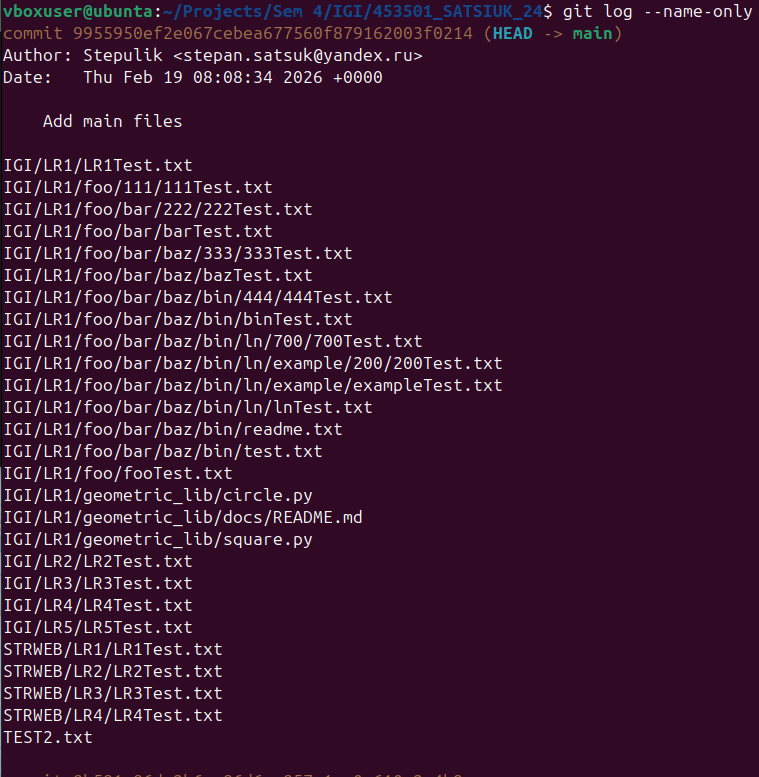
Флаг --stat довольно компактный и удобный, + и – показывают кол-во изменений в строках



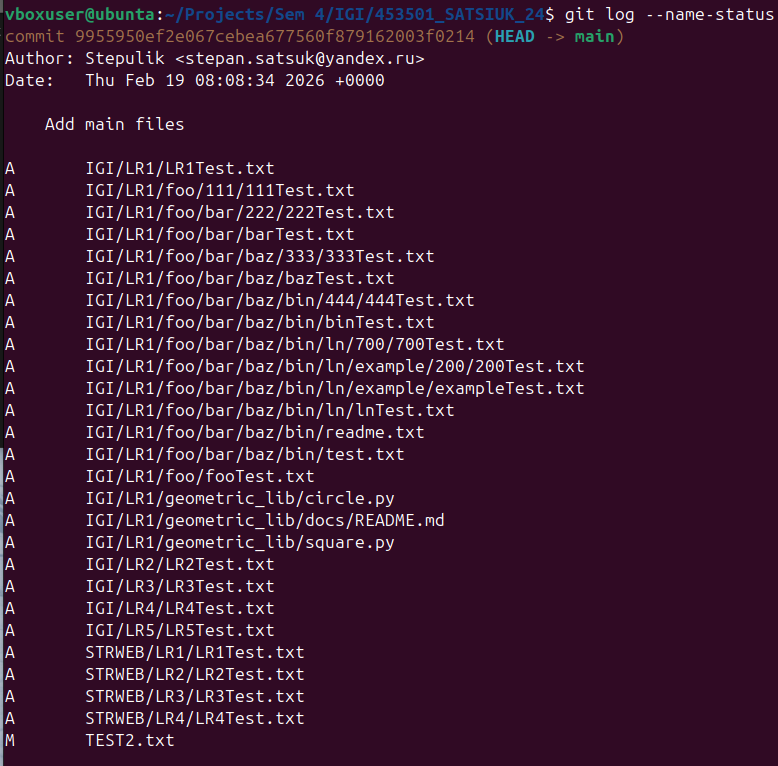
Флаг --shortstat добавляет немного информации о обычному log, по логике отображает статистику без имен файлов в общих чертах



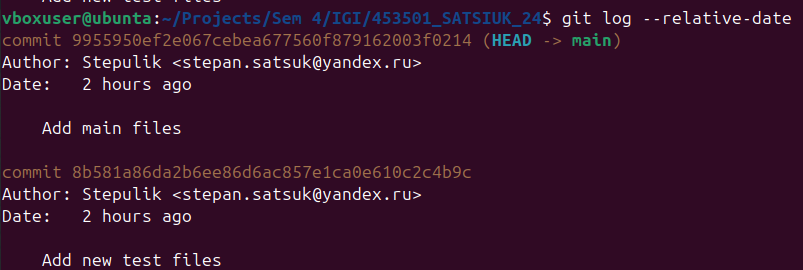
Флаг --name-only показывает только имена измененных файлов без конкретной статистики



Флаг --name-status как и передыдущий, но добавляет пометку что в коммите с файлом случилось (A – add, M – modified, D- deleted)



Флаг --relative-date выводит относительное время коммитов (а не дату)



Флаг --pretty меняет формат отображения для удобства

--pretty=oneline — каждый коммит в одну строку: хеш и заголовок.

--pretty=short — показывает хеш, автор, заголовок.

--pretty=full — добавляет коммиттера.

--pretty=fuller — ещё больше деталей.

--pretty=format:"..." — самый гибкий вариант: вы сами задаёте, что показывать, с помощью заполнителей (placeholders):

%H — полный хеш коммита

%h — сокращённый хеш

%an — имя автора

%ae — email автора

%ad — дата автора (можно менять формат через --date=...)

%ar — относительная дата автора

%cn — имя коммиттера

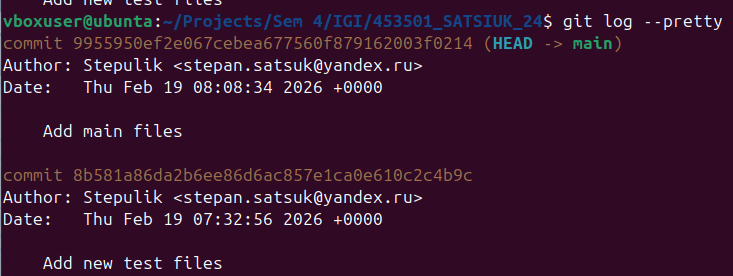
%ce — email коммиттера

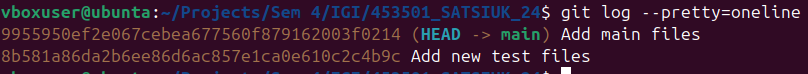
%cd — дата коммиттера

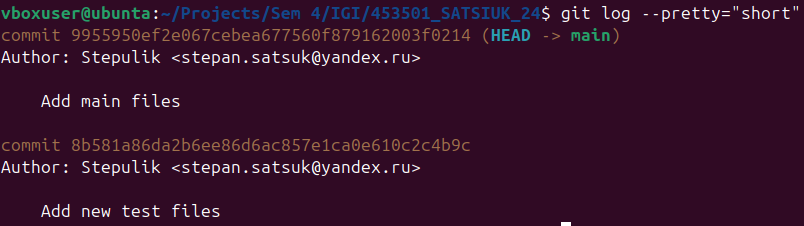
%s — заголовок коммита

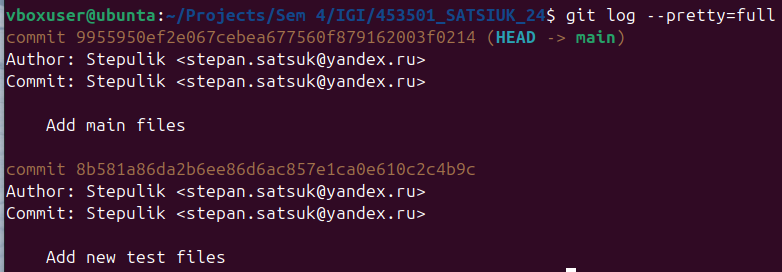
%b — тело сообщения

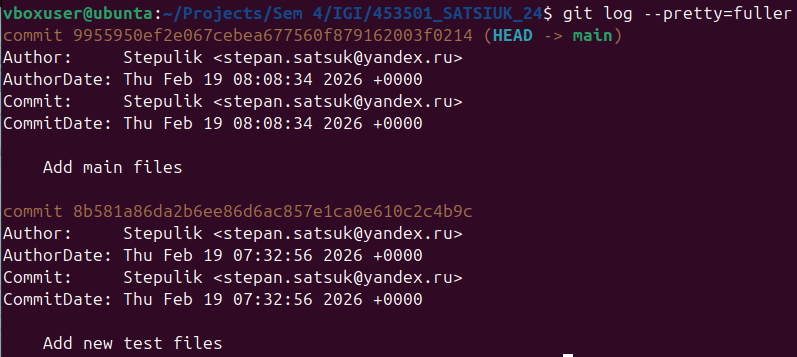
%d — ссылки (ветки, теги), указывающие на коммит

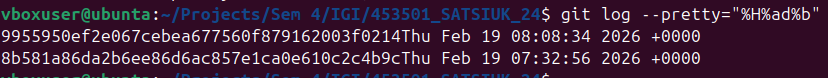




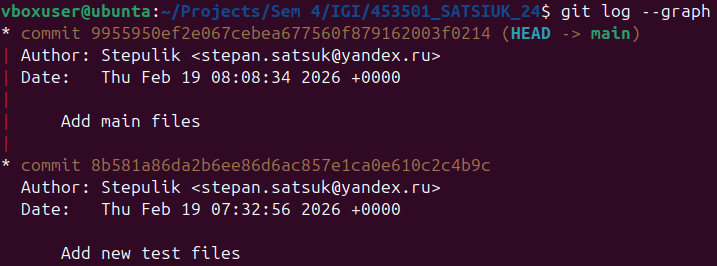


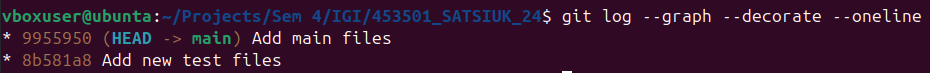




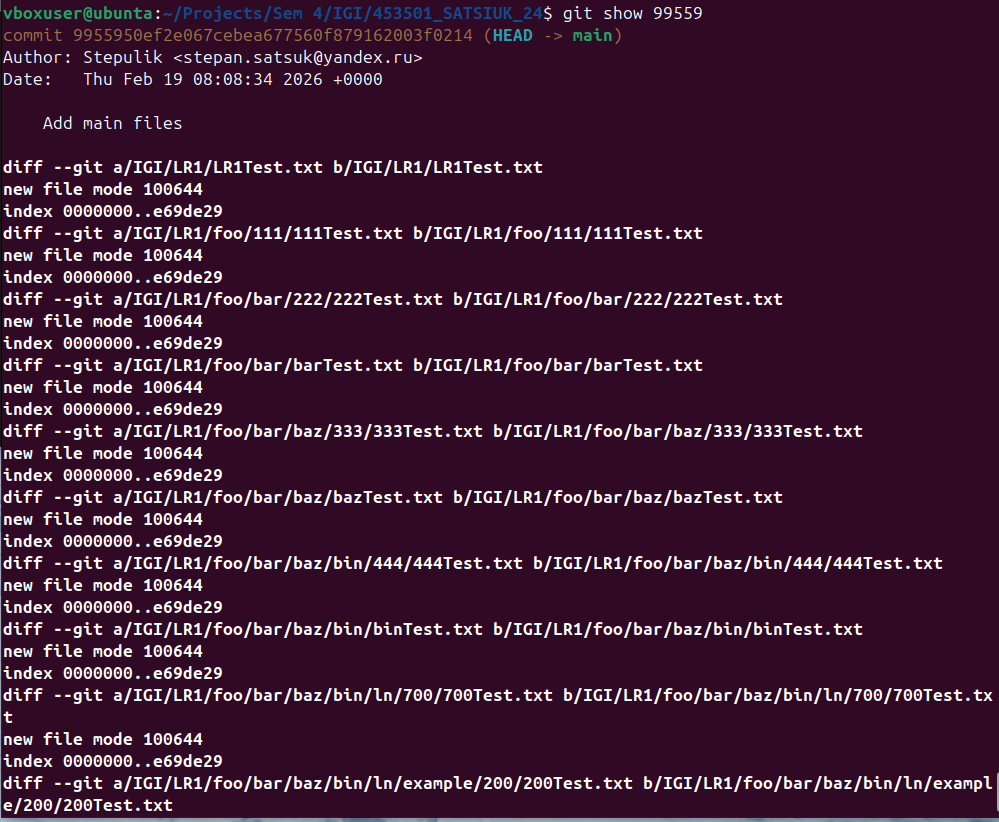


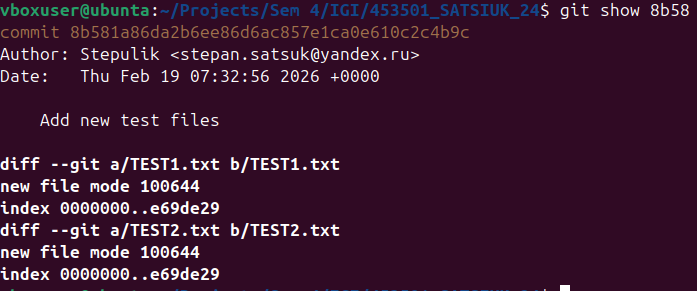
Флаг --graph показывает историю ветвлений и слияний, можно комбинировать с --oneline и –decorate





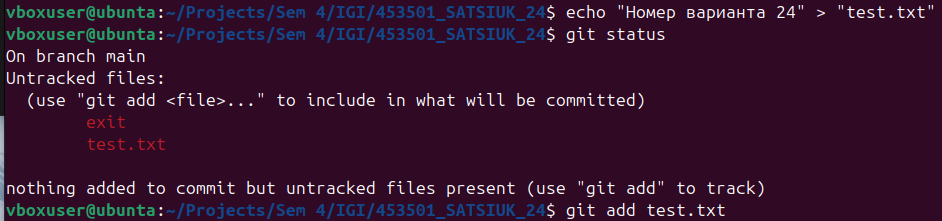
1. найдите разницу в изменениях для каждого коммита

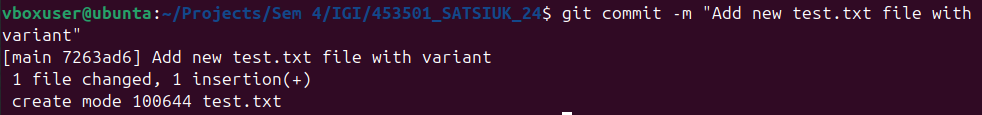


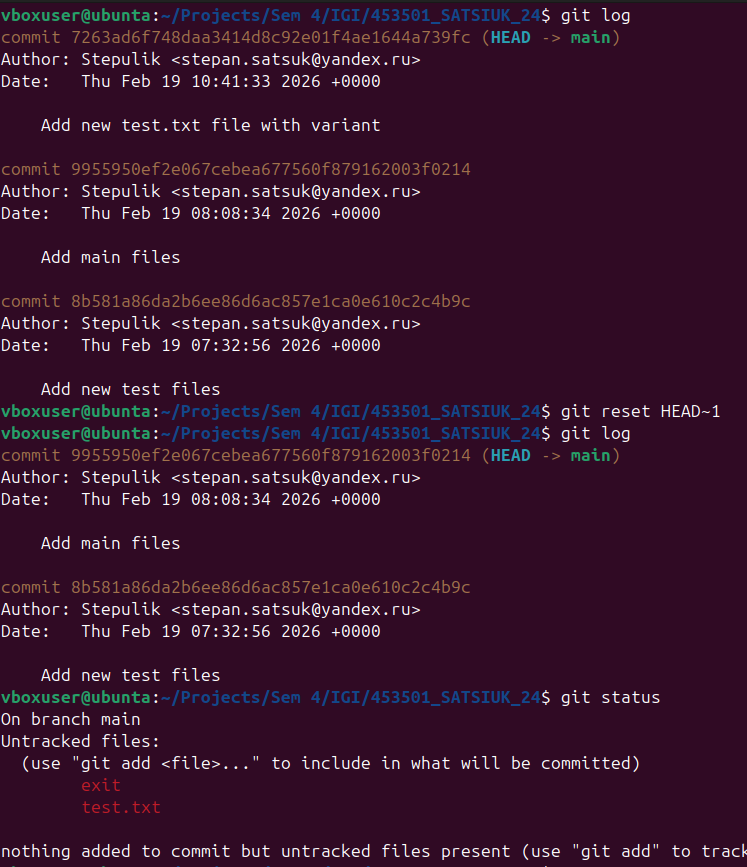


Вот она, использована команда git show (хеш коммита, взятый из git log)

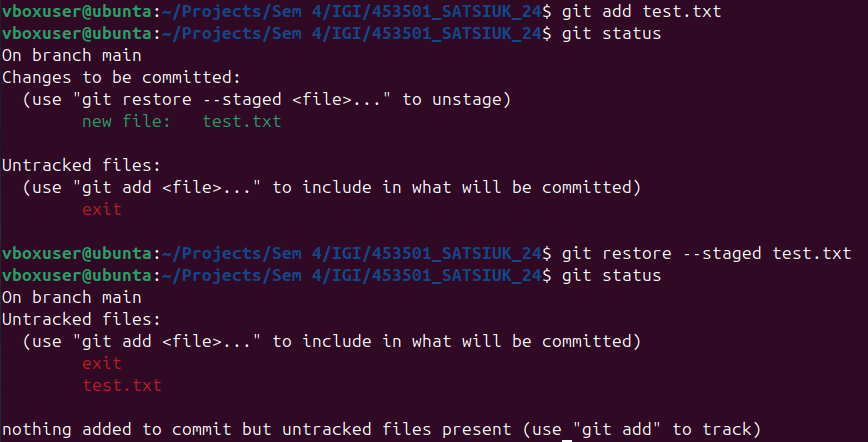
**j.** создайте новый файл test.txt и вставьте в него строчку 'номер варианта'. затем сделайте коммит и отмените его





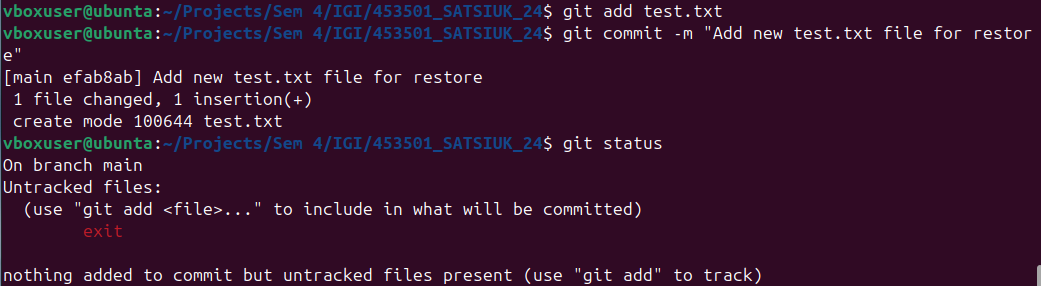
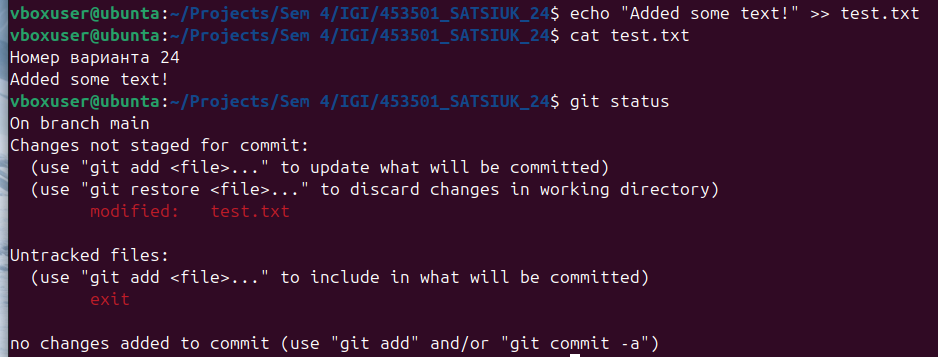


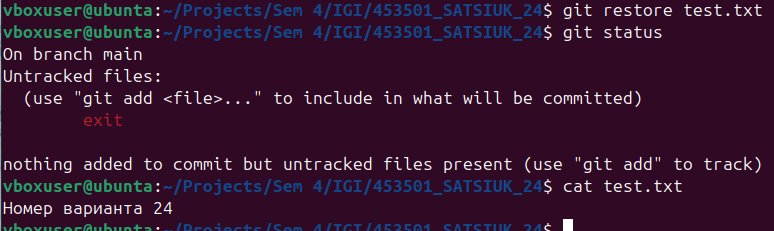
**k**. проиндексируйте файл в вашем Git-репозитории, затем отмените его индексацию



**l.** внесите изменения в файл в вашем Git-репозитории и затем верните его в исходное состояние.

Добавим test.txt в коммит, затем изменим его содержимое для наглядности

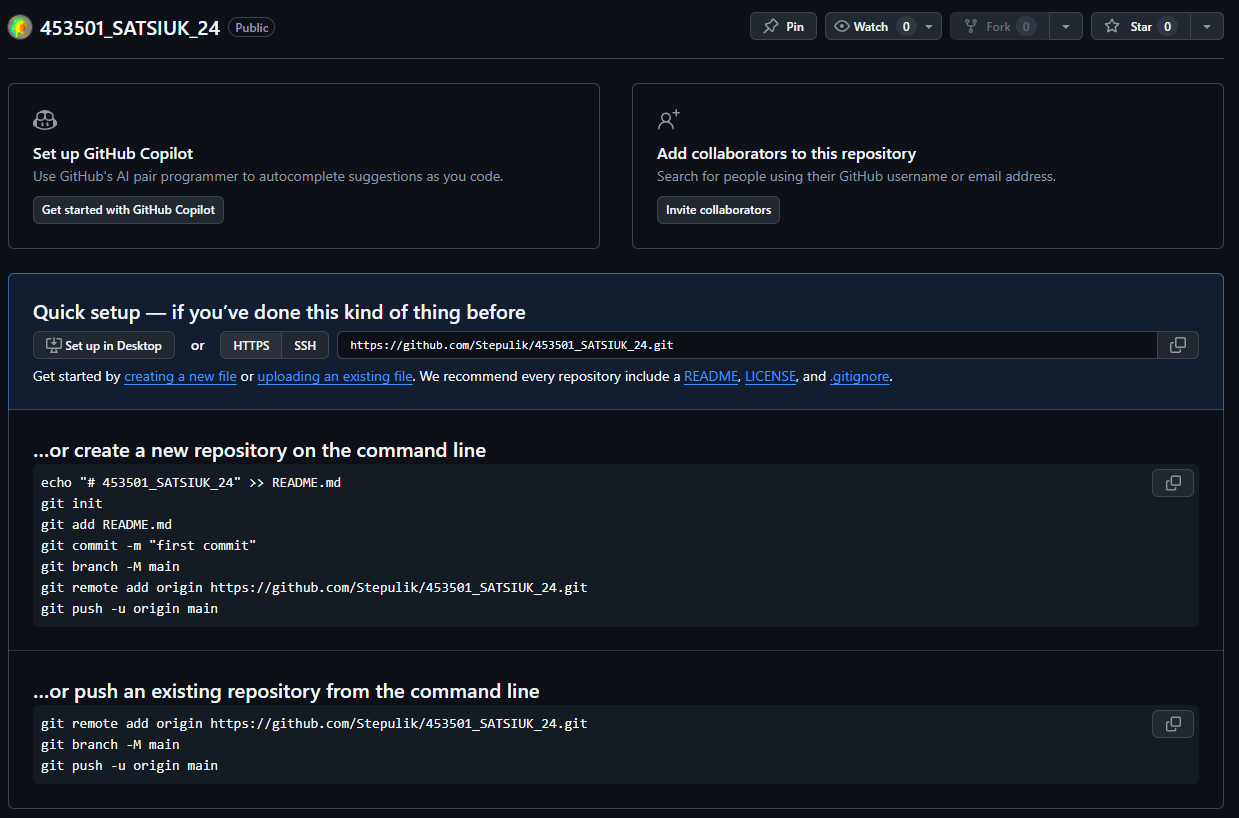


Все вернулось к закоммиченой версии

**5)** Выполните следующие действия с удаленным репозиторием:

**a.** создайте удаленный репозиторий (на github.com)

с помощью git remote add origin … связываем репозиторий с гитхабом



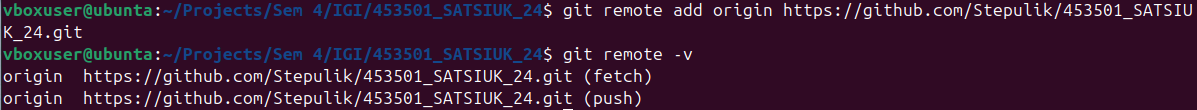
**b.** просмотрите информацию о нем, внесите изменения и опять просмотрите (remote, fetch).

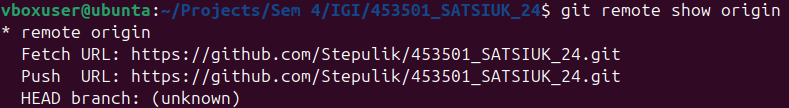
git remote – управление удалёнными репозиториями

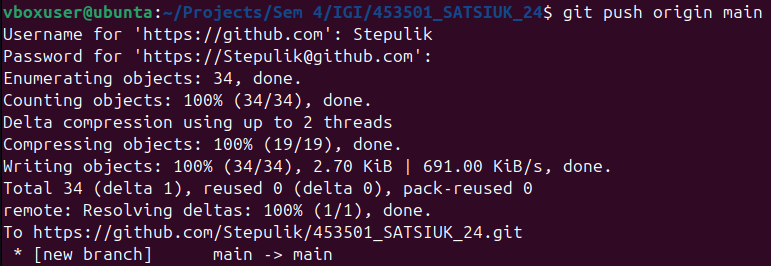
git fetch – получение данных с удалённого репозитория без слияния

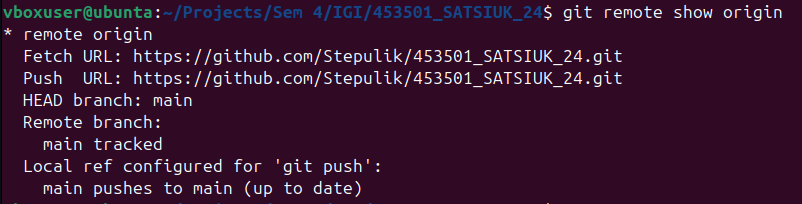
Создал ключик для подключения и использовал в качестве пароля

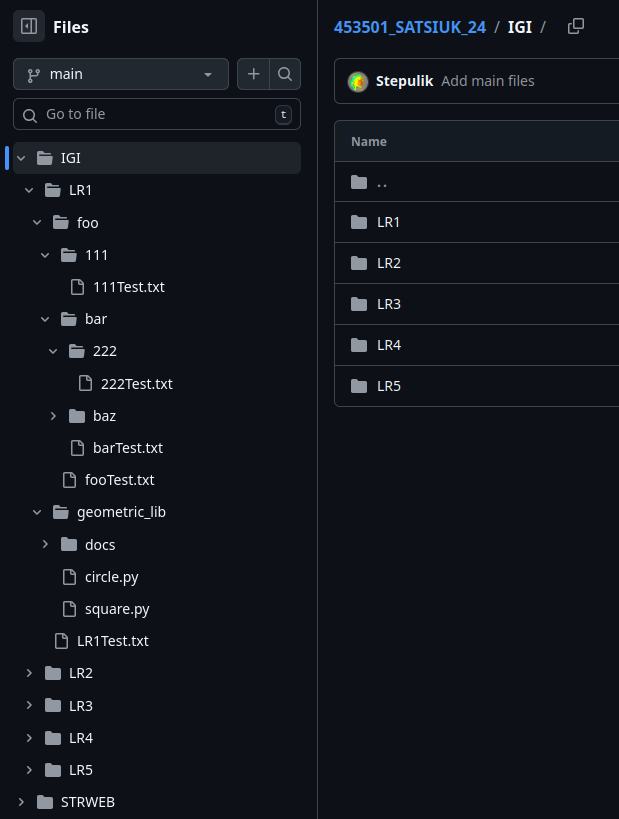








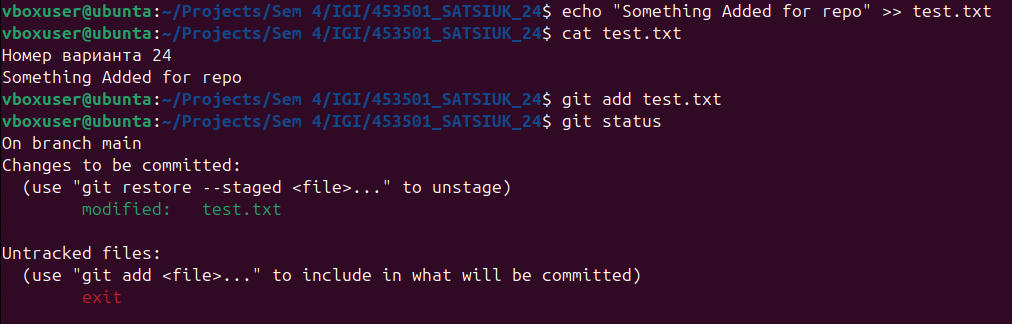


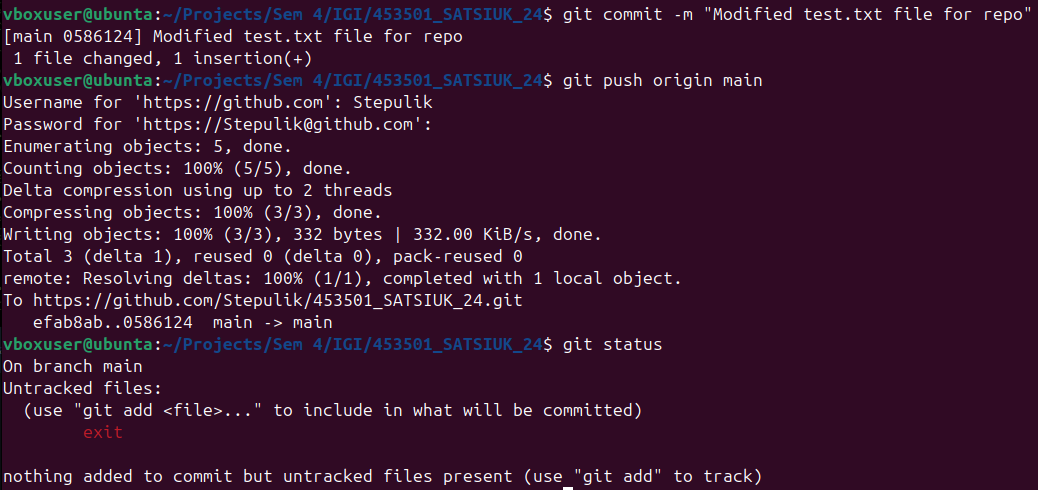


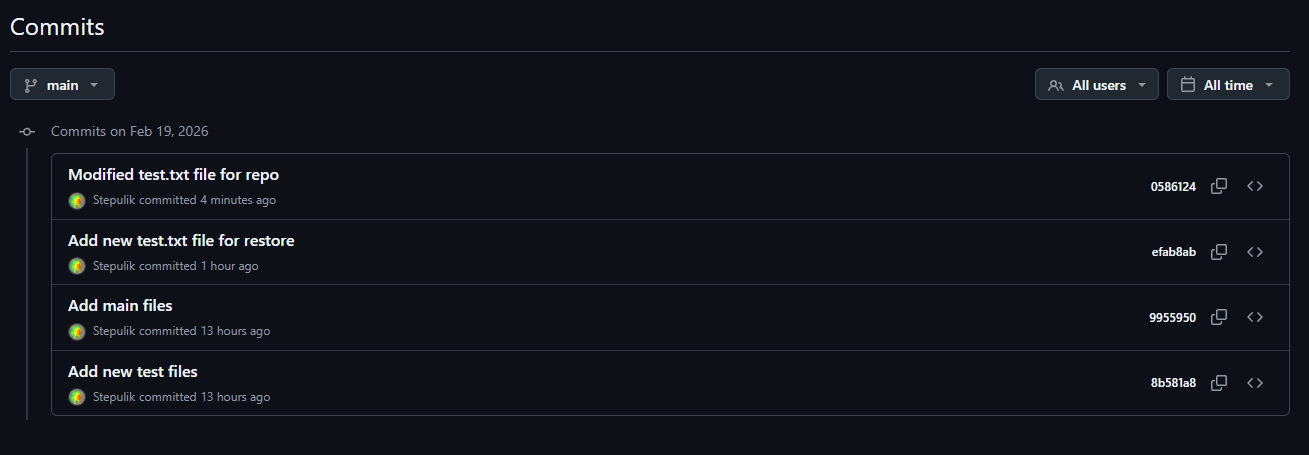
Все получилось

Все успешно восстановлено с помощью команды git restore –source=HEAD –IGI/ где HEAD – ссылка на последний коммит, -- IGI/ - какие файлы загрузить.

1. измените несколько файлов в вашем проекте и перенесите их в удаленный репозиторий





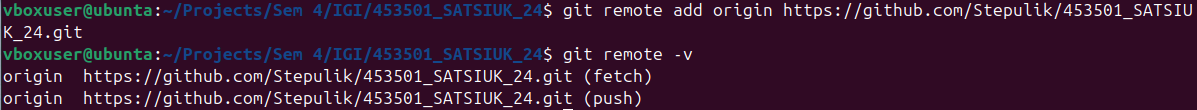




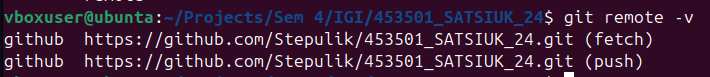
1. переименуйте свой удаленный репозиторий



Было:



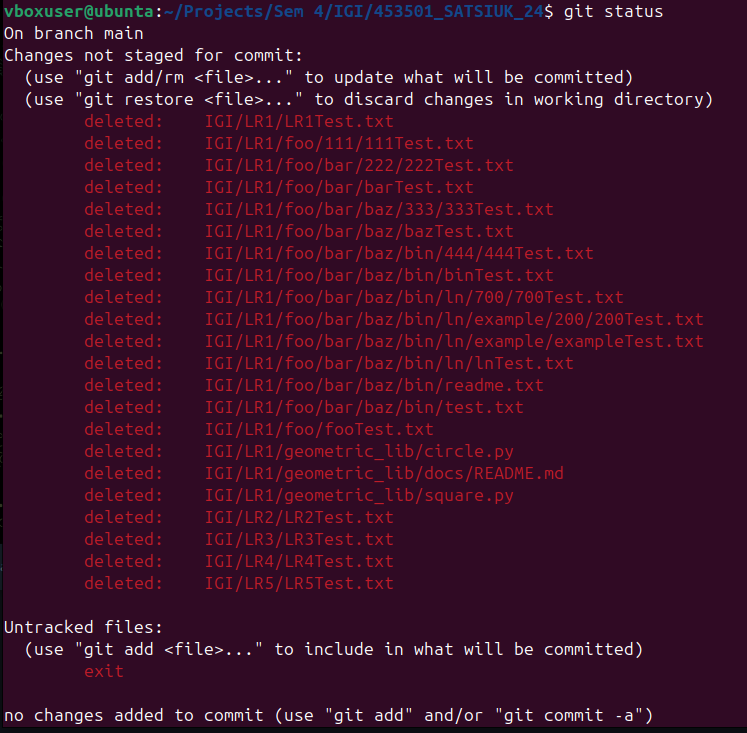
Стало:

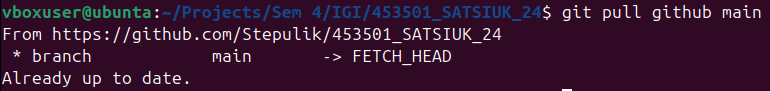


**e.** получите данные из удаленного репозитория

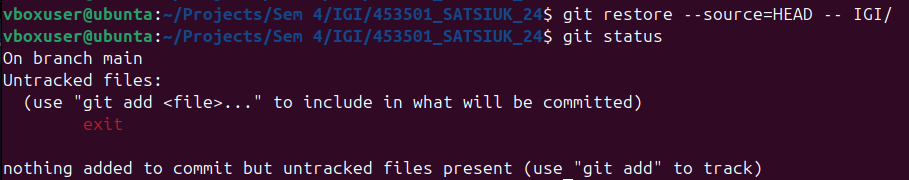
Попробуем удалить папку IGI из локального репозитория и восстановить с удаленного репозитория на гитхабе



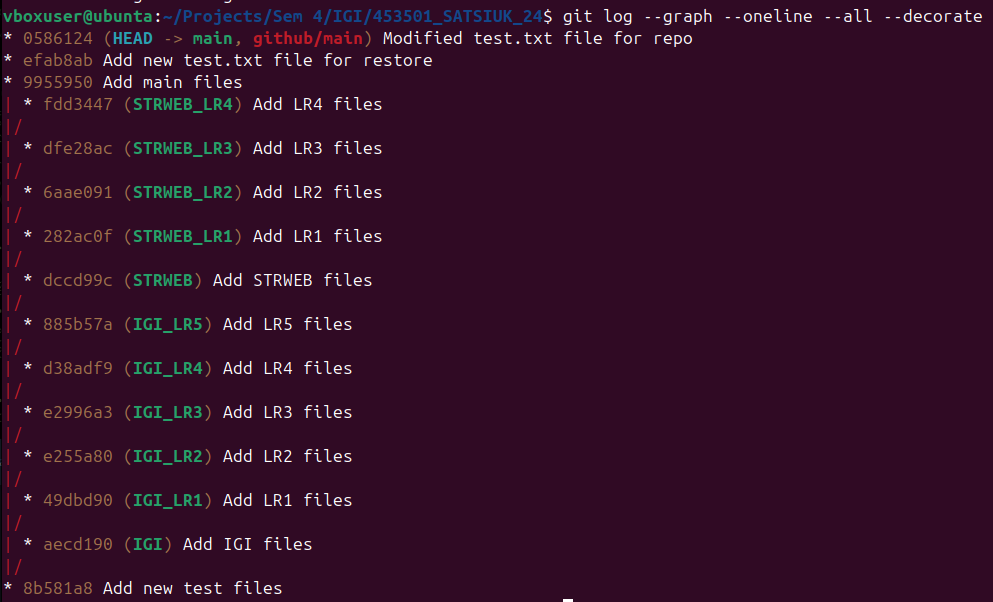




Или так: но вобще git pull = git fetch + git merge



1. просмотреть историю изменений



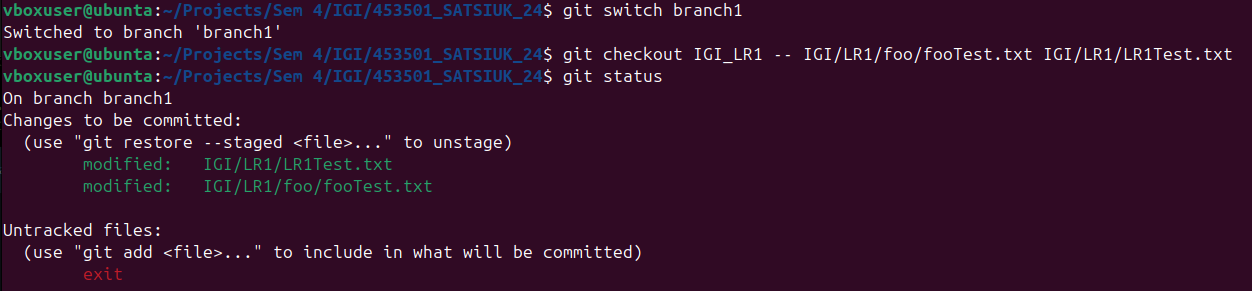
1. создайте дополнительную ветку 1 в удаленном репозитории, перенесите в нее два файла из другой ветки, удалите один из файлов

создаем веточку branch1 на основе main

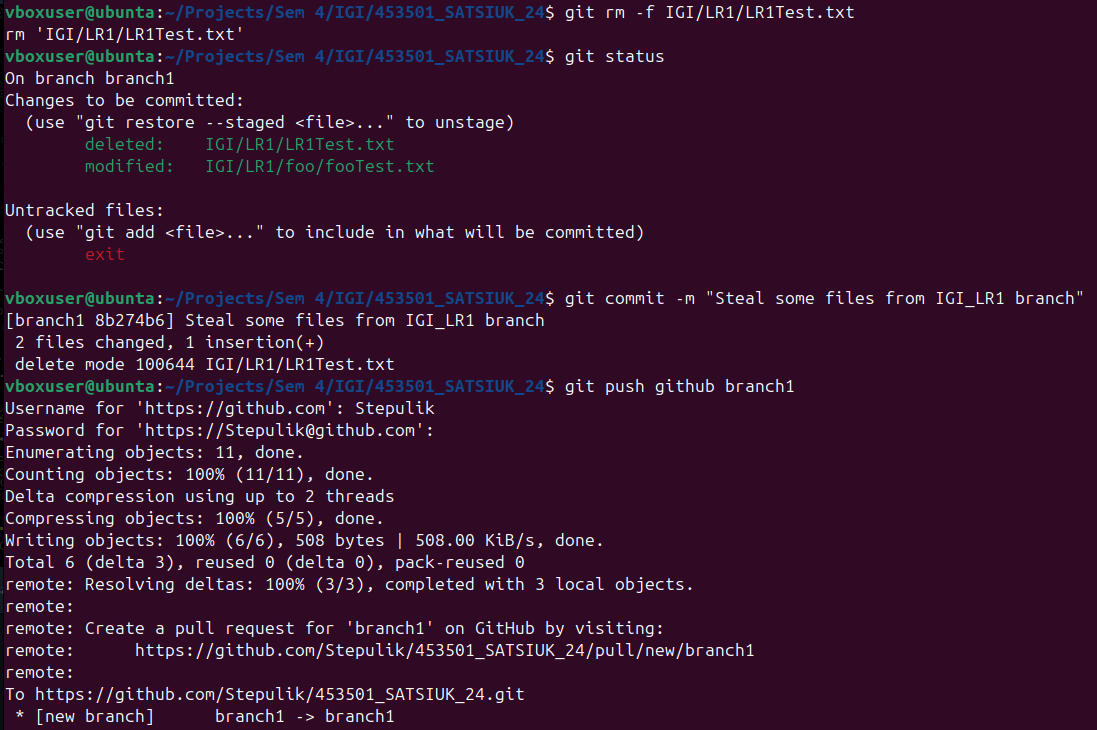


Добавляем файлы из веточки IGI\_LR1, для начала изменим файлы чтобы было что добавлять





Удалим один из файлов, закоммитим и зальем готовую ветку branch1 на github

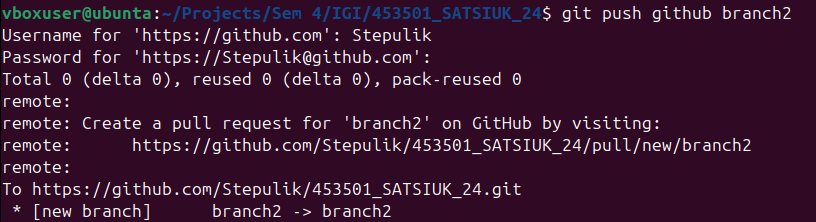


1. создайте дополнительную ветку 2 в удаленном репозитории, перенесите в нее два файла из вашего проекта, слить изменения в основную ветку, удалить дополнительную ветку 2

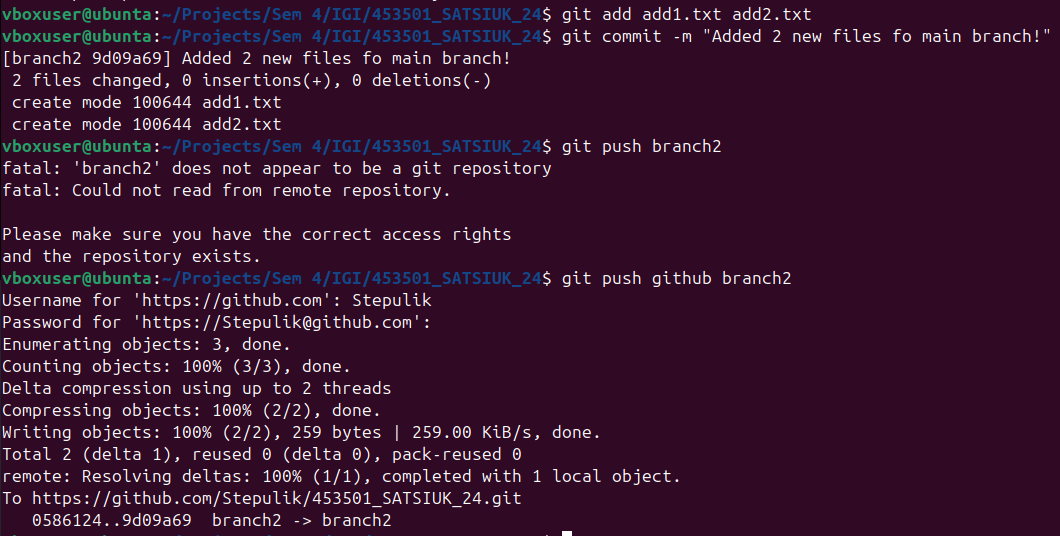




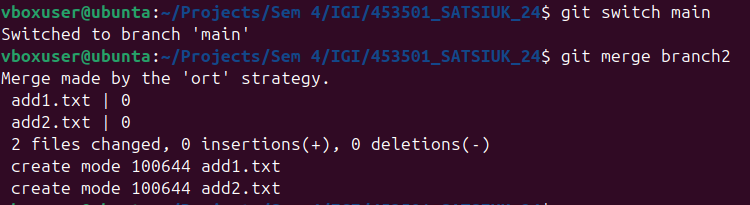
У нас создался репозиторий из еще неизмененной ветки branch2

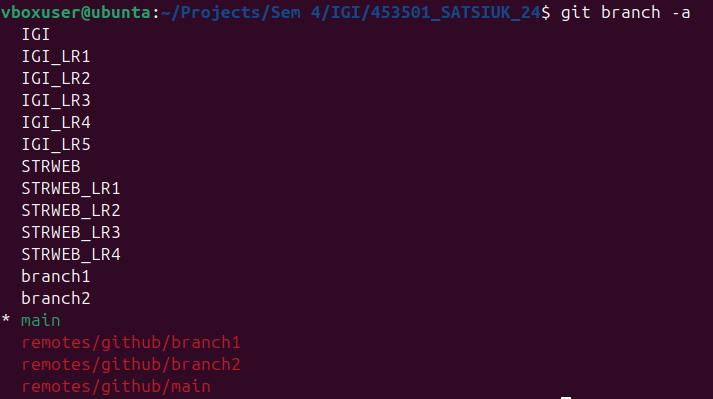


Запушим branch2 с новыми созданными файлами

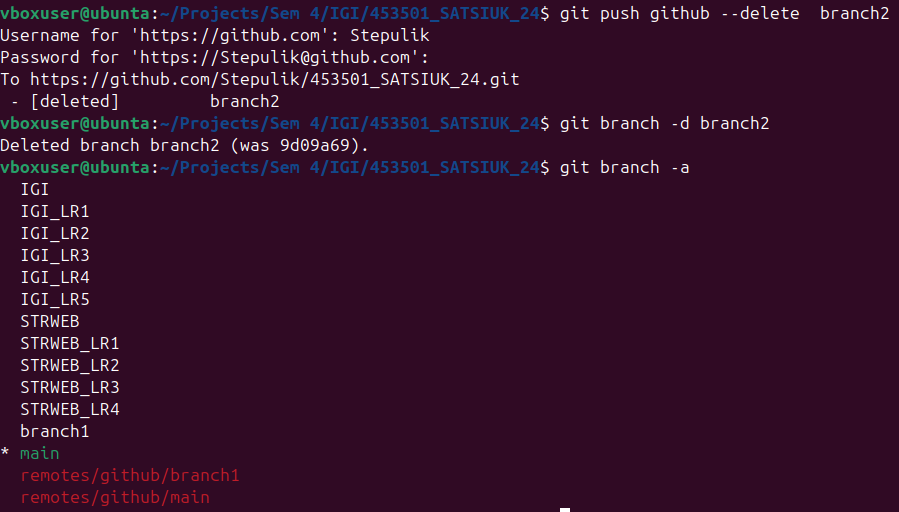


Делаем слияние



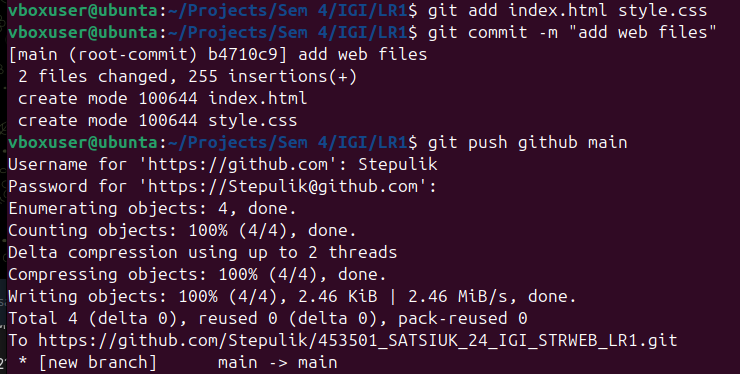


Удаляем ветку



**6)** Сгенерируйте в DeepSeek файл html и файл css для сайта визитки компании или человека с областью занятости на выбор.

Создайте публичный репозиторий на github и загрузите туда оба сгенерированных файла.



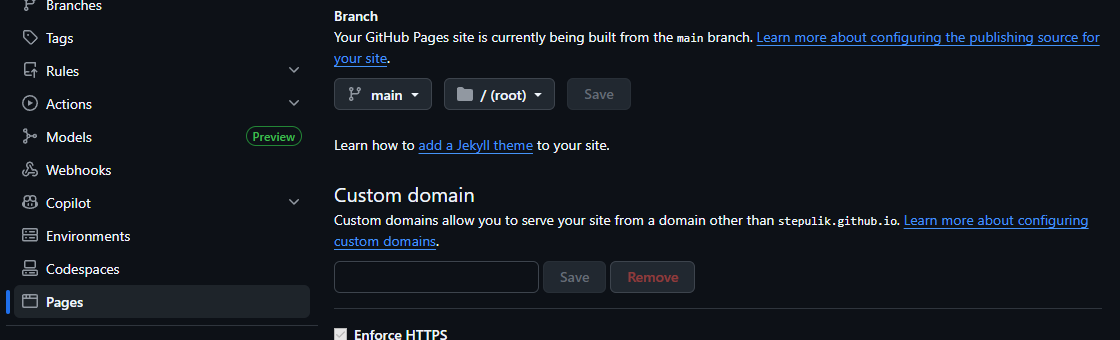


● Разместить сайт в публичном репозитории на github.

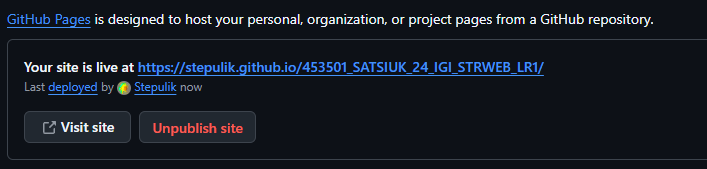
Настроить Settings, чтобы можно было просмотреть сайт в браузере:

перейти на вкладку Pages Default branch/ выбрать в списке main;

нажать Save.



Подождать, Опять кликнуть по Pages и нажать кнопку Visit site вверху вкладки GitHub Pages.



Не забыть дописать в отчет ссылку на сайт для проверки.

**Your site is live at <https://stepulik.github.io/453501_SATSIUK_24_IGI_STRWEB_LR1/>**

Для справки можно посмотреть: Бесплатный хостинг GitHub Pages <https://www.youtube.com/watch?v=S3pkpo5NqJY>

**Индивидуальное задание:**

Структура папок, Примечание:

bar/708 – в папке bar нужно расположить папку 708

test.txt/007 – в файле test.txt нужно поместить текст 007