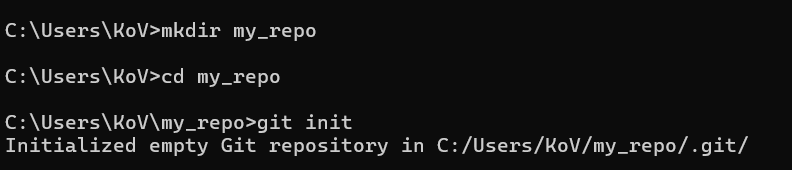
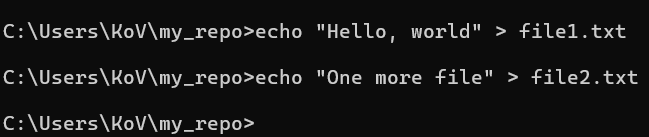
Задание

**Цель работы:** получить навыки по работе с командной строкой и git’ом.

# Часть 1. Основные команды Git

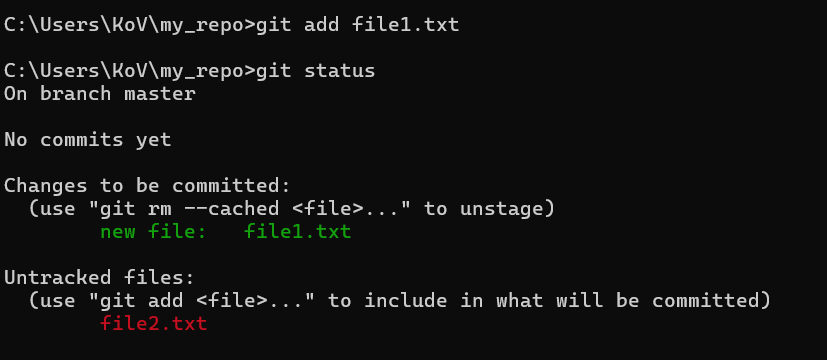
1. Установите и настройте клиент git на своей рабочей станции.
2. Создайте локальный репозиторий и добавьте в него несколько файлов. 



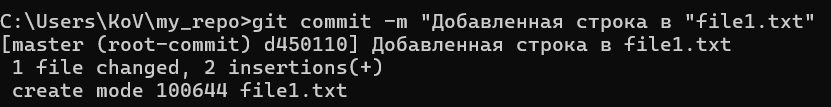
1. Внесите изменения в один из файлов.



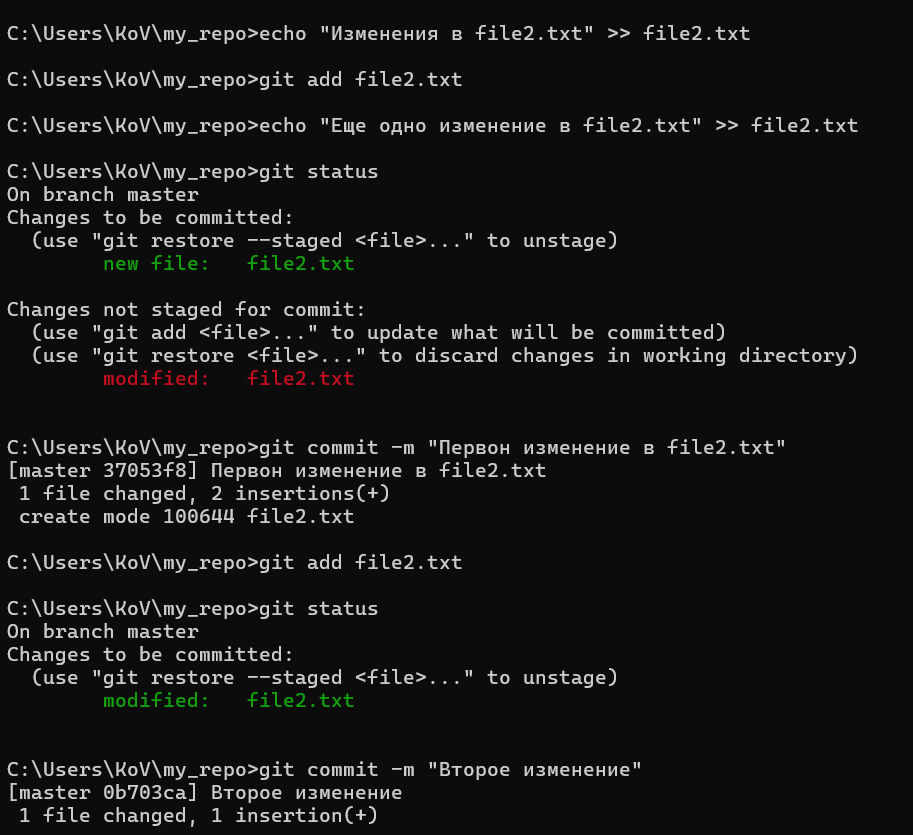
1. Проиндексируйте изменения и проверьте состояние.



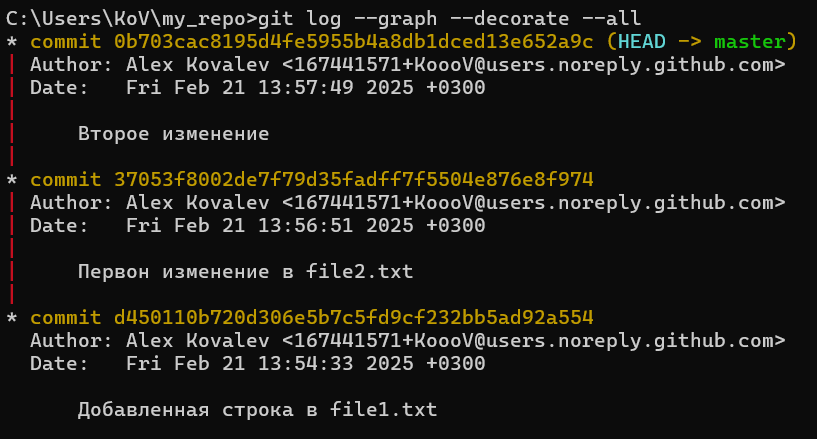
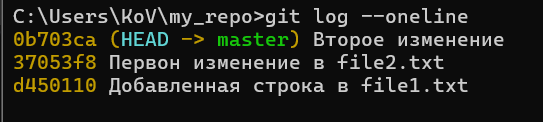
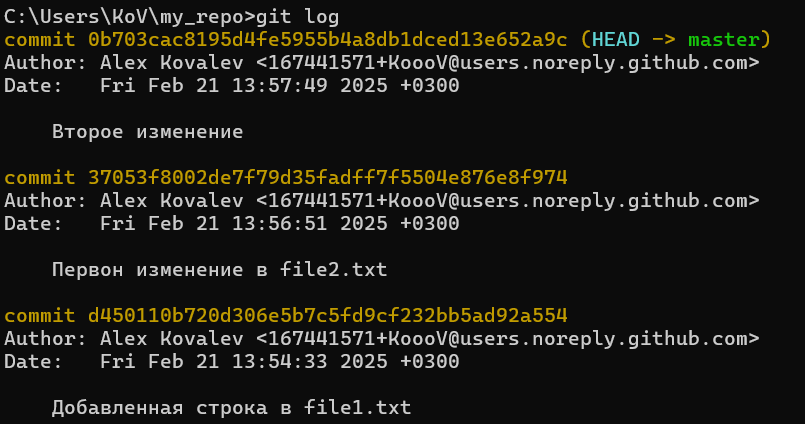
1. Сделайте коммит того, что было проиндексировано в репозиторий. Добавьте к коммиту комментарий.



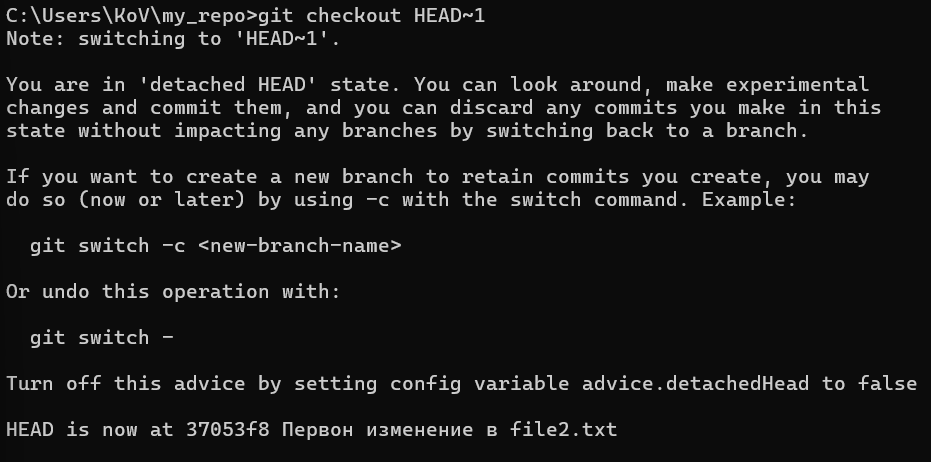
1. Измените еще один файл. Добавьте это изменение в индекс git. Измените файл еще раз. Проверьте состояние и произведите коммит проиндексированного изменения. Теперь добавьте второе изменение в индекс, а затем проверьте состояние с помощью команды git status. Сделайте коммит второго изменения.



1. Просмотрите историю коммитов с помощью команды git log. Ознакомьтесь с параметрами команды и используйте некоторые из них для различного формата отображения истории коммитов.



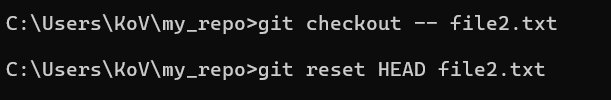
1. Верните рабочий каталог к одному из предыдущих состояний.



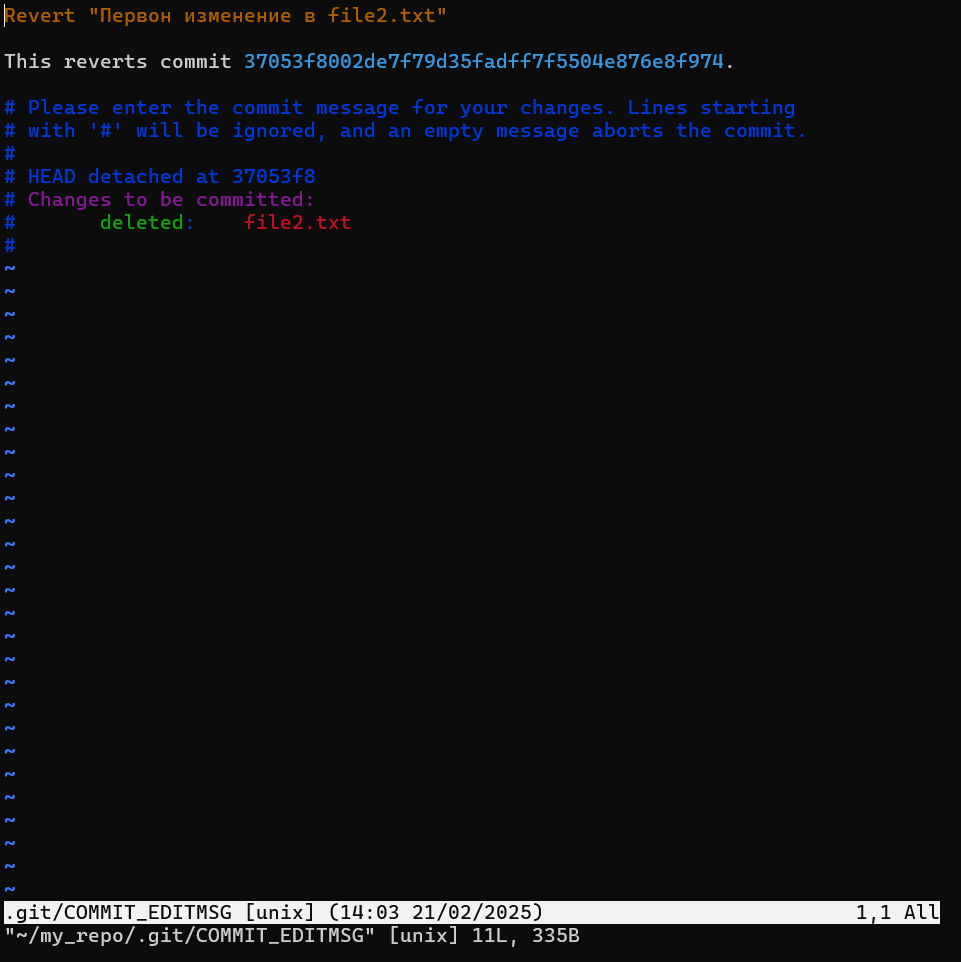
1. Изучите, как создавать теги для коммитов для использования в будущем.



1. Отмените некоторые изменения в рабочем каталоге (до и после индексирования).



1. Отмените один из коммитов в локальном репозитории.

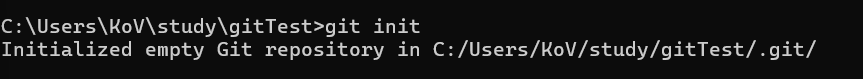


# Часть 2. Системы управления репозиториями

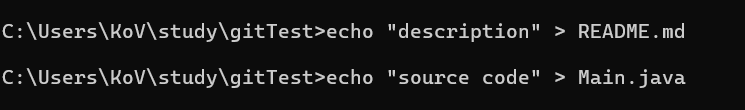
1. Создайте аккаунт на GitHub (у кого нет),

https://github.com/KoooV

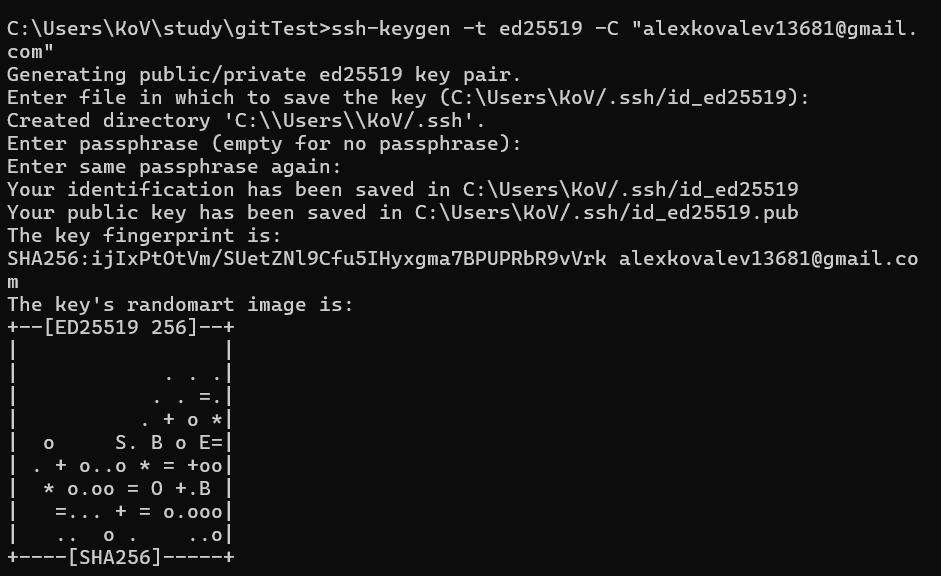
1. Создайте репозиторий на GitHub и на локальной машине, согласно выбранной теме проекта,



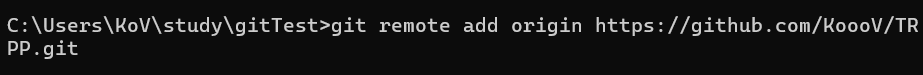
1. Создайте несколько файлов на локальной машине при помощи консоли,

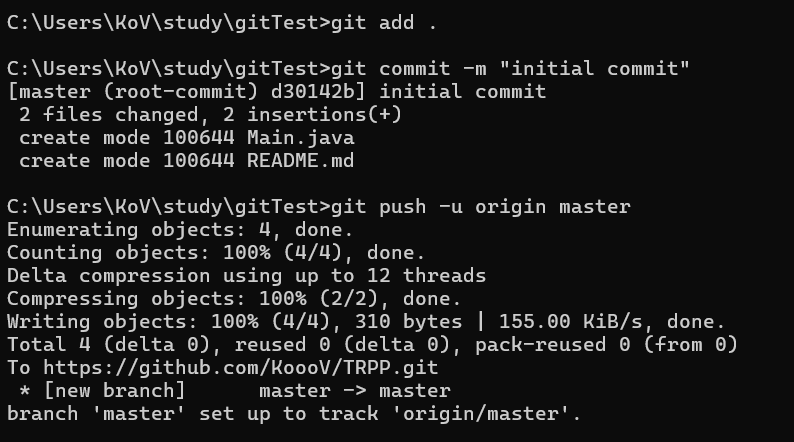


1. Создайте SSH-ключ для авторизации,



1. Свяжите репозиторий локальной машины с репозиторием на GitHub при помощи консоли,





1. Создайте новую ветку в репозитории с помощью команды, произведите в ней какие-нибудь изменения, а после слейте с веткой master,



1. Выполните цепочку действий в репозитории, согласно вариантам.

Обоснование: V=Ост((N-1)/M)+1

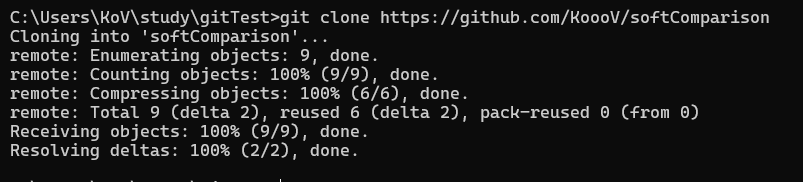
N = 11(номер в списке), M = 11(количество тем)

V = Ост((11-1)/11)+1 = 10 + 1 = 11

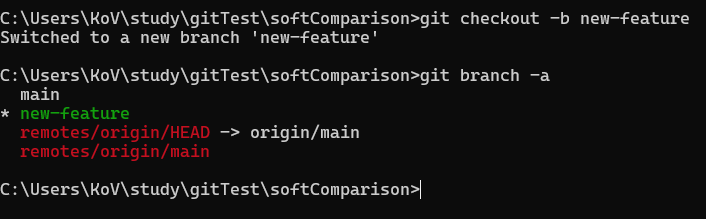
Ост(10/11) = 10

Вариант 11.

1. Клонируйте непустой удаленный репозиторий на локальную машину

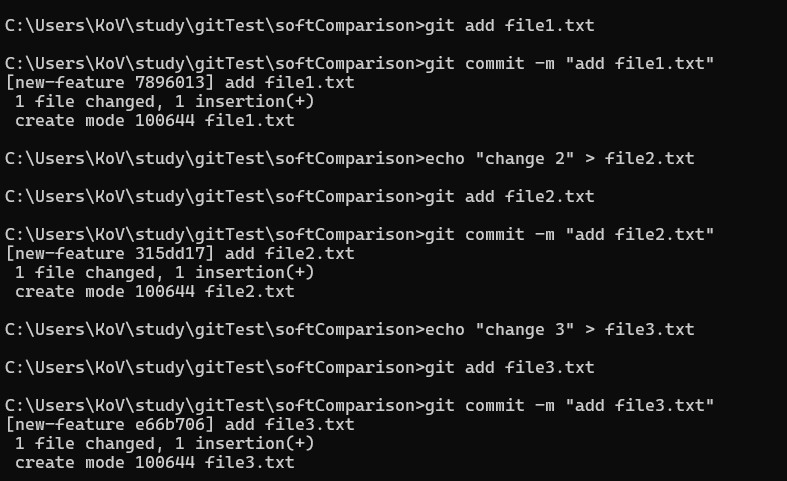


1. Создайте новую ветку и выведите список всех веток

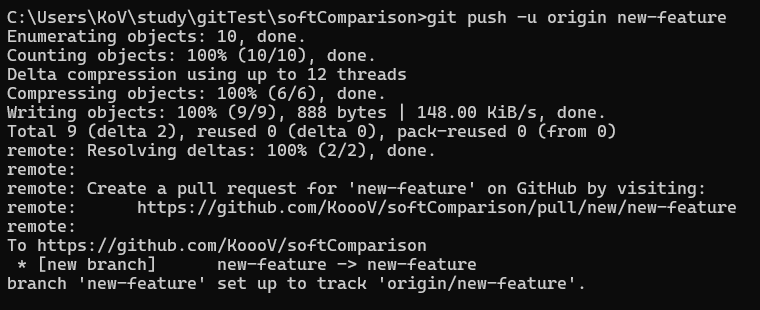


1. Произведите 3 коммита в новой ветке в разные файлы

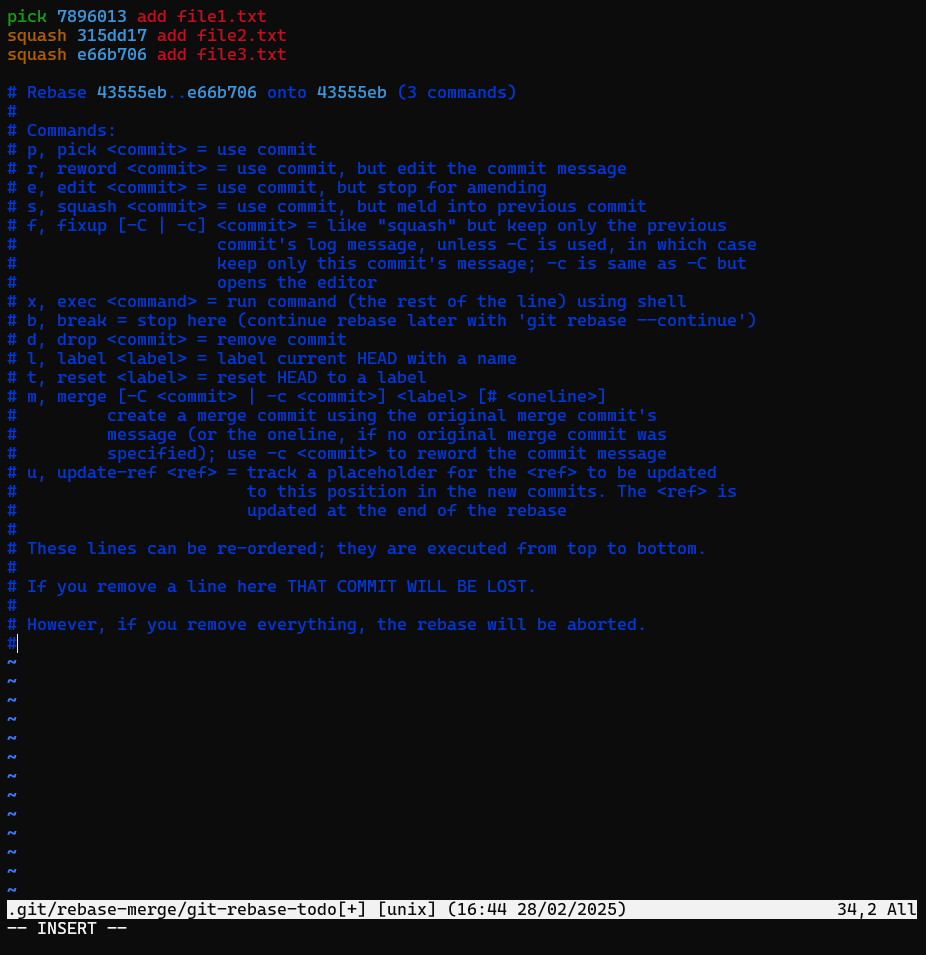




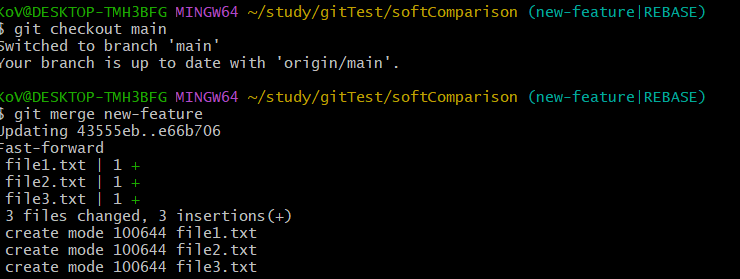
1. Выгрузите изменения в удаленный репозиторий



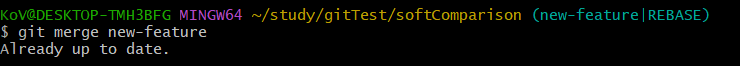
1. Добейтесь того, чтобы эти три изменения были представлены одним коммитом



1. Выведите в консоли различия между веткой master и новой веткой



1. Слейте новую ветку с master при помощи merge



# Часть 3. Работа с ветвлением и оформление кода

1. Сделайте форк репозитория в соответствии с вашим вариантом

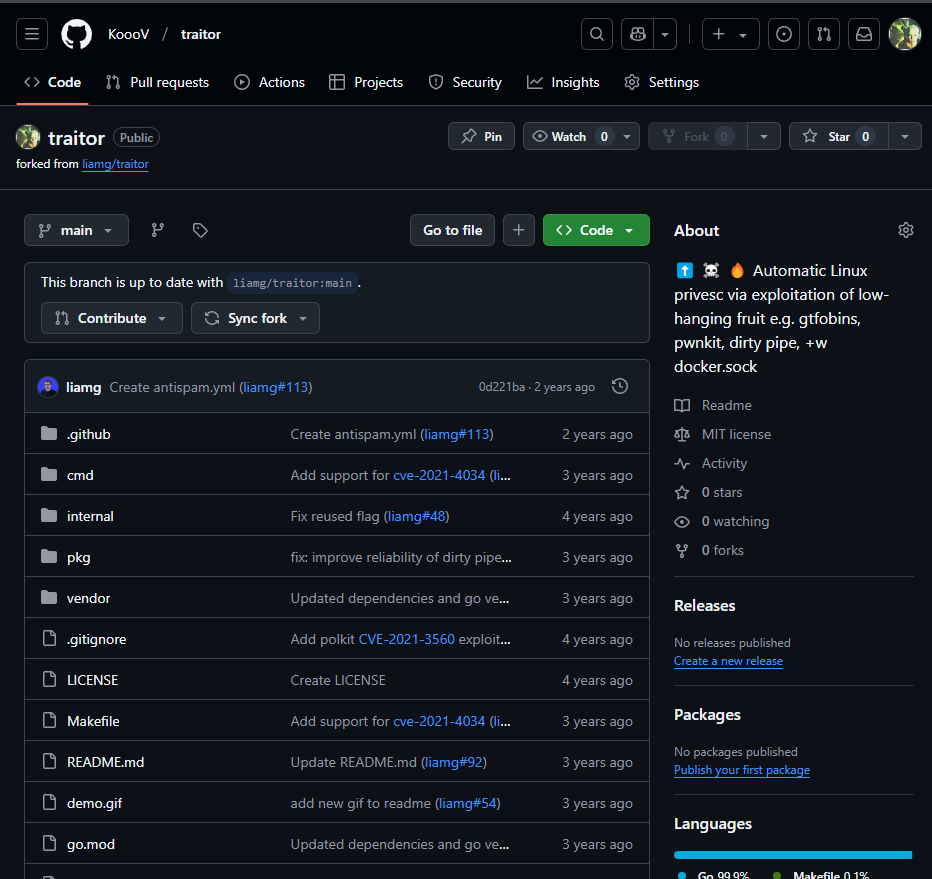
V=Ост((N-1)/M)+1

N = 11(номер в списке), M = 11(количество тем)

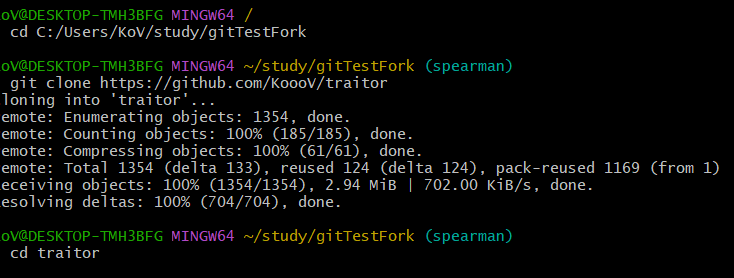
V = Ост((11-1)/10)+1 = 0 + 1 = 1 => Вариант 1

11 -1 = 10

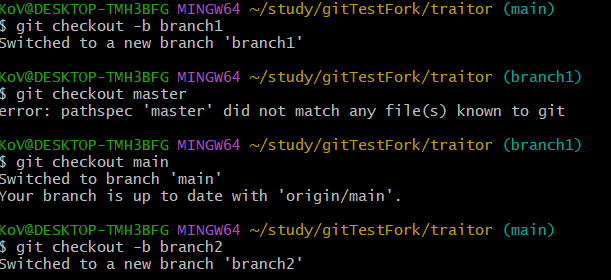
10 /10 = 1 остаток от деления = 0



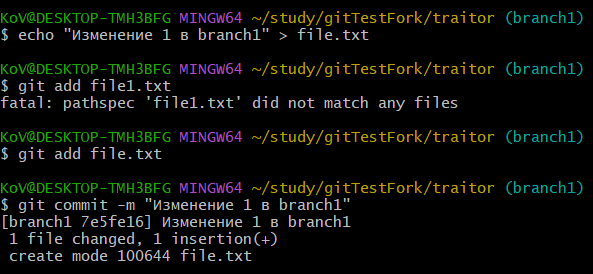
1. Склонируйте его на локальную машину

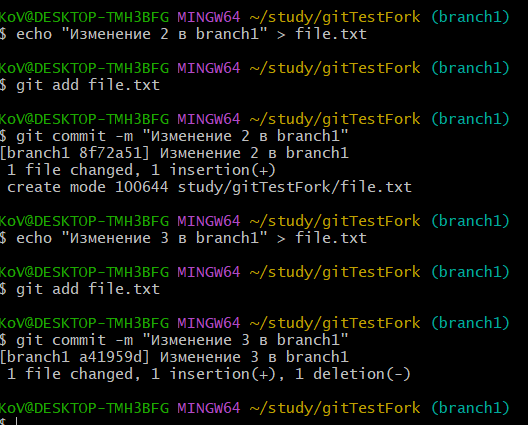


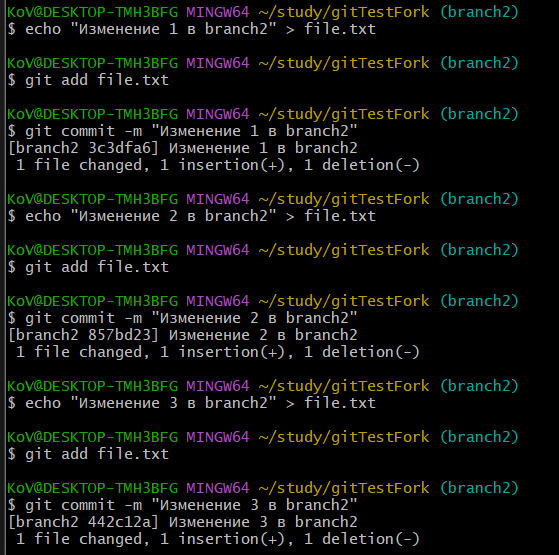
1. Создайте две ветки branch1 и branch2 от последнего коммита в master'е



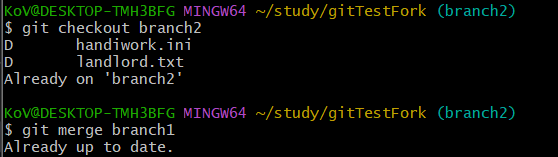
1. Проведите по 3 коммита в каждую из веток, которые меняют один и тот же кусочек файла



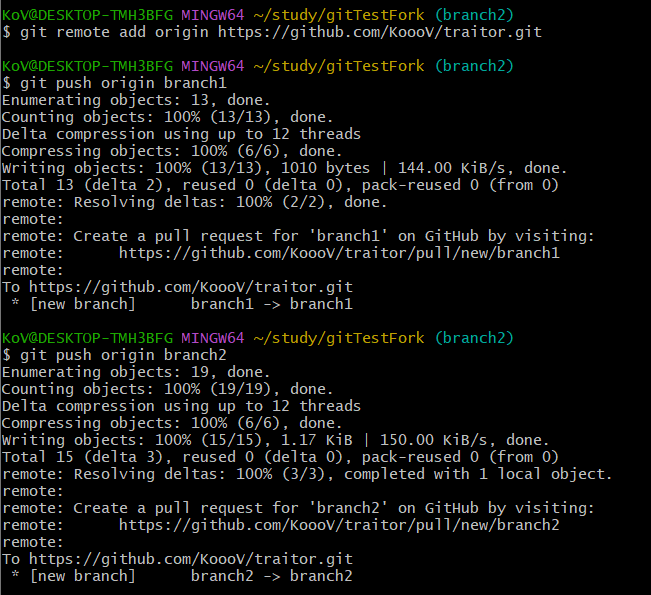




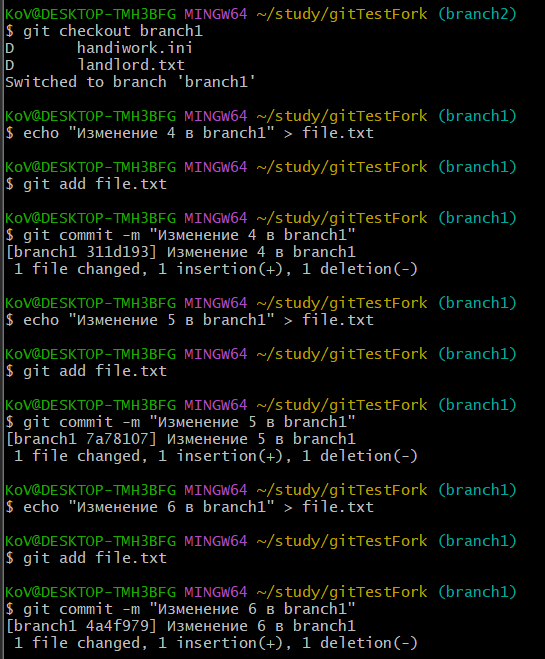
1. Выполните слияние ветки branch1 в ветку branch2, разрешив конфликты при этом



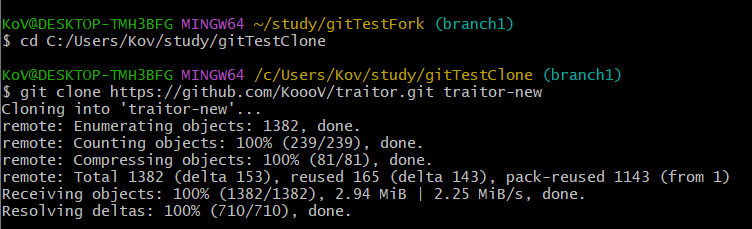
1. Выгрузите все изменения во всех ветках в удаленный репозиторий



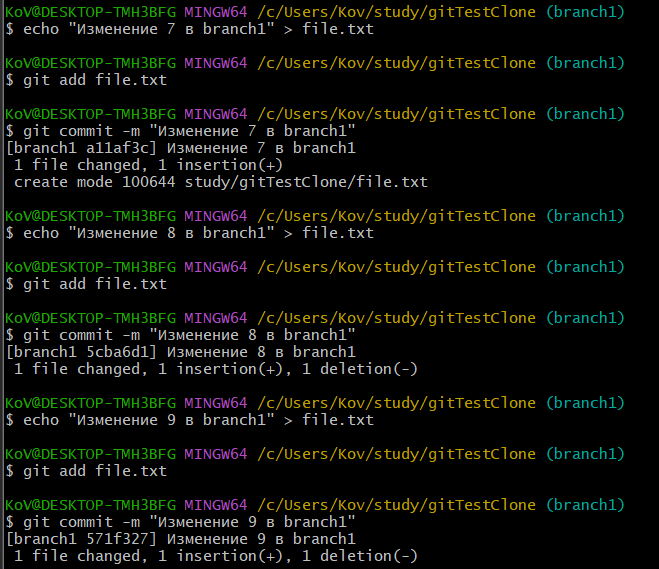
1. Проведите еще 3 коммита в ветку branch1



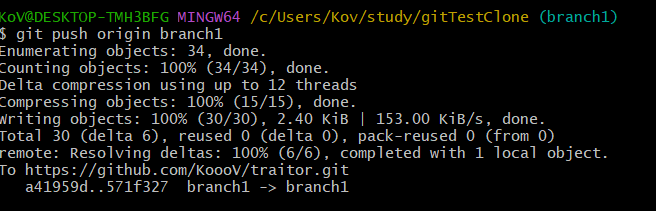
1. Склонируйте репозиторий еще раз в другую директорию



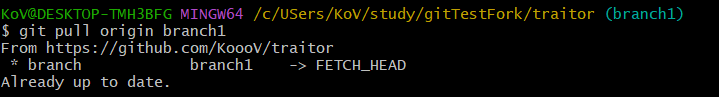
1. В новом клоне репозитории сделайте 3 коммита в ветку branch1

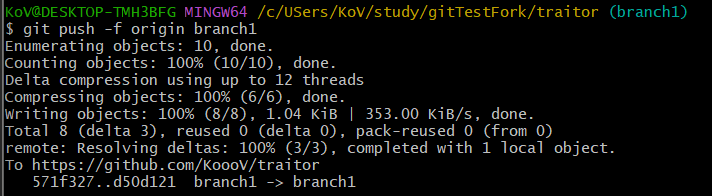


1. Выгрузите все изменения из нового репозитория в удаленный репозиторий

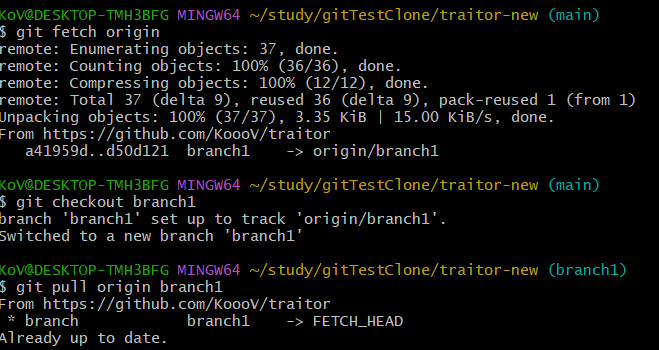


1. Вернитесь в старый клон с репозиторием, выгрузите изменения с опцией –force





1. Получите все изменения в новом репозитории



|  |  |
| --- | --- |
| № варианта | Репозиторий |
| 1 | https://github.com/liamg/traitor |
| 2 | https://github.com/google/model\_search |
| 3 | https://github.com/gto76/python-cheatsheet |
| 4 | https://github.com/ripienaar/free-for-dev |
| 5 | https://github.com/pystardust/ytfzf |
| 6 | https://github.com/airbnb/javascript |
| 7 | https://github.com/kettanaito/naming-cheatsheet |
| 8 | https://github.com/Sairyss/domain-driven-hexagon |
| 9 | https://github.com/react-native-camera/react-native-camera |
| 10 | https://github.com/spring-projects/spring-petclinic |

**Контрольные вопросы**

1. Какие существуют типы систем контроля версий? Приведите примеры к каждому типу.

Локальные (RCS, SCCS) — хранят изменения локально.

Централизованные (CVCS: SVN, CVS) — единый сервер-репозиторий.

Распределенные (DVCS: Git, Mercurial) — полные копии репозитория у каждого пользователя.

2. К какому типу систем контроля версий относится Git?

Git — распределенная система контроля версий (DVCS).

3. Что такое репозиторий Git?

Это хранилище данных проекта, включающее историю изменений, ветки, теги и метаданные. Располагается в папке .git.

4. Что такое коммит?

Снимок изменений в репозитории на определенный момент времени. Имеет уникальный хеш (например, a1b2c3d).

5. Что такое ветка в репозитории Git?

Указатель на определенный коммит. Позволяет изолировать разработку новой функциональности.

6. Что такое тег в репозитории Git?

Метка, закрепляющая состояние репозитория (например, версия релиза: ). Отмечает значимые коммиты.

7. Что такое слияние двух веток?

Объединение изменений из двух веток в одну. Например, слияние в.

8. Что такое конфликт в Git? Как его решить и почему они бывают?

Возникает, когда изменения в разных ветках противоречат друг другу.  
Решение:

Отредактировать конфликтующие участки в файлах.

Добавить исправленные файлы (git add).

Завершить слияние (git commit).  
Причина: Git не может автоматически определить, какие изменения оставить.

9. Как отменить слияние веток, если произошел конфликт?

git merge --abort

10. Для чего нужен .gitignore?

Файл, указывающий Git, какие файлы/папки игнорировать (например, временные файлы, бинарники).

11. Что делает команда git status?

Показывает состояние рабочей директории и индекса: измененные, добавленные, неотслеживаемые файлы.

12. Что делает команда git add?

Добавляет изменения из рабочей директории в индекс (staging area) для последующего коммита.

13. Что делает команда git log?

Отображает историю коммитов текущей ветки. Параметры:

--oneline — краткий вывод.

--graph — визуализация веток.

14. Что делает команда git diff?

Показывает различия:

Между рабочим каталогом и индексом: git diff.

Между индексом и последним коммитом: git diff --staged.

15. Что делает команда git show?

git show a1b2c3d

16. Что делает команда git stash?

git stash pop # Восстановить изменения

17. Как добавить новую директорию в Git?

Git не отслеживает пустые папки. Чтобы добавить директорию, создайте в ней файл (например, .gitkeep).

18. Что сделает команда "git branch" без какого-либо параметра?

Выводит список локальных веток. Текущая ветка помечена звездочкой (\*).

19. Чем отличаются команды "git push" и "git pull"?

git push — отправляет локальные изменения в удаленный репозиторий.

git pull — забирает изменения из удаленного репозитория и объединяет с локальным.

20. Что означает статус файла untracked в выводе команды git status?

Файл существует в рабочей директории, но не добавлен в Git (не отслеживается).

21. Что означает статус файла new в выводе команды git status?

Файл добавлен в индекс (git add), но еще не закоммичен.

22. Что означает статус файла modified в выводе команды git status?

Файл изменен после добавления в индекс, но изменения не добавлены в новый коммит.

23. Чем отличается master и origin/master?

master — локальная ветка.

origin/master — удаленная ветка на сервере (origin — имя удаленного репозитория).

24. Как узнать, кто автор строчки в файле, используя систему Git?

git blame file.txt -L 5,10 # Покажет автора строк 5-10

25. Как отменить действие команды "git add" на файл?

git reset file.txt

26. Как привести измененный файл в начальное состояние (до изменения)?

git restore file.txt

27. Как сделать ветку с названием my\_branch?

git branch my\_branch

28. Как сделать коммит для ветки my\_branch?

git checkout my\_branch

git add .

git commit -m "Message"

29. Как удалить локальную ветку my\_branch?

git checkout my\_branch

git add .

git commit -m "Message"

30. Как исправить ошибку "fatal: The current branch

my\_branch has no upstream branch", возникающую при вводе git push?

git push --set-upstream origin my\_branch

31. Как удалить все untracked файлы?

git clean -fd # -f: force, -d: директории

32. Что такое GitHub?

Платформа для хостинга Git-репозиториев с инструментами для совместной работы (Issues, Pull Requests).

33. Что такое форк репозитория?

Копия чужого репозитория на вашем аккаунте GitHub. Позволяет вносить изменения без прямого доступа к исходному репозиторию.

34. Что такое pull request?

Запрос на слияние ваших изменений из форка или ветки в основной репозиторий. Позволяет обсудить код перед мержем.