**Отчет о лабораторной работе №3**

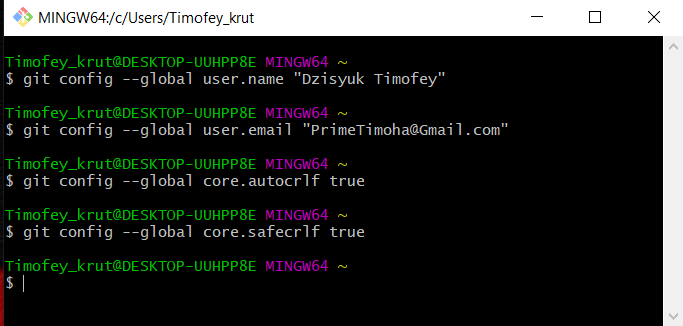
**Отчет выполнил Дзисюк Тимофей Степанович**

**группа БПИ21-01**

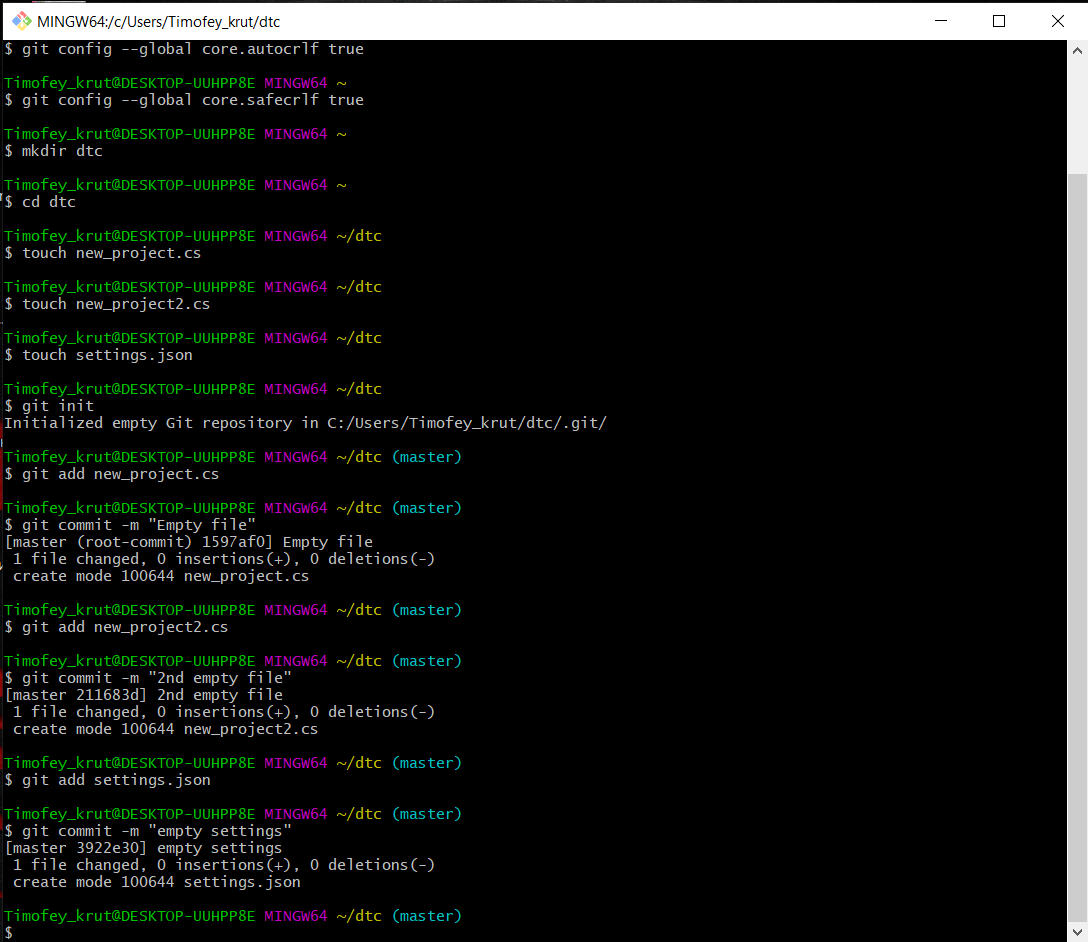
**Ход работы**

**6 Вариант**: Использовать в работе 3 разных файла, и количество версий каждого файла не должно быть меньше 3.

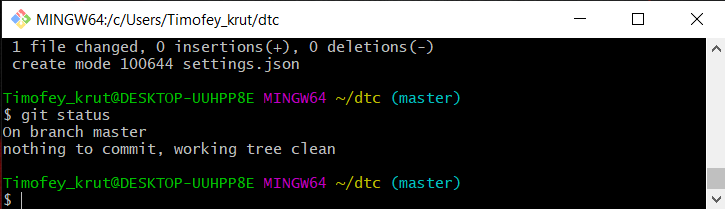
1. Подготовка к работе с Git. Первоначальная настройка.



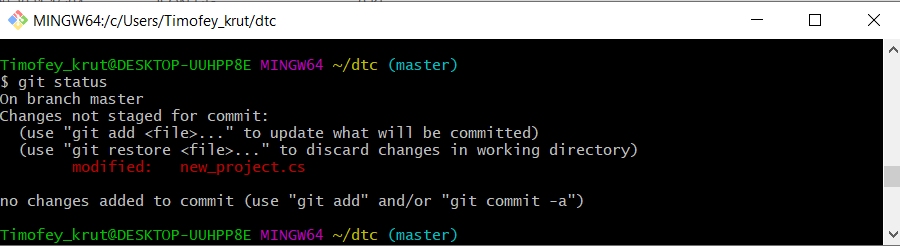
2. Создание каталога, 3 файлов, репозитория.



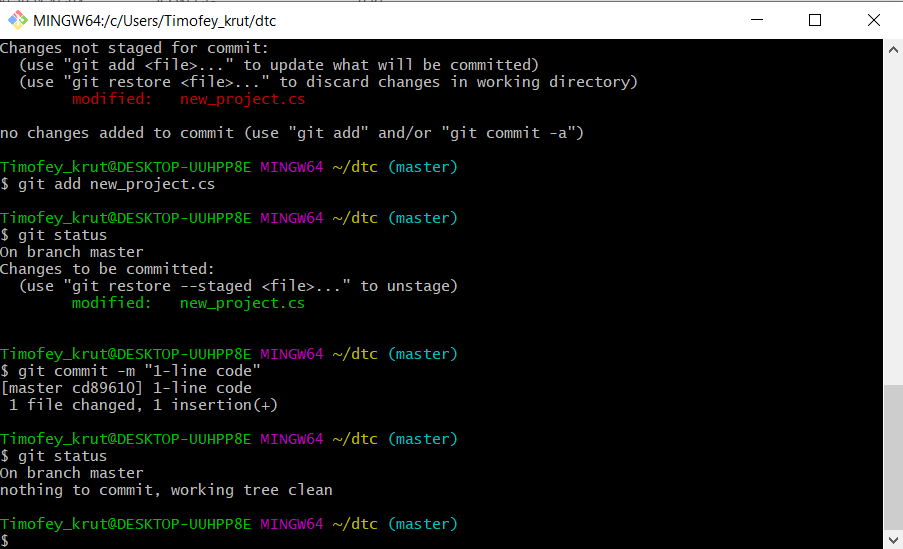
3. Проверка состояния репозитория.



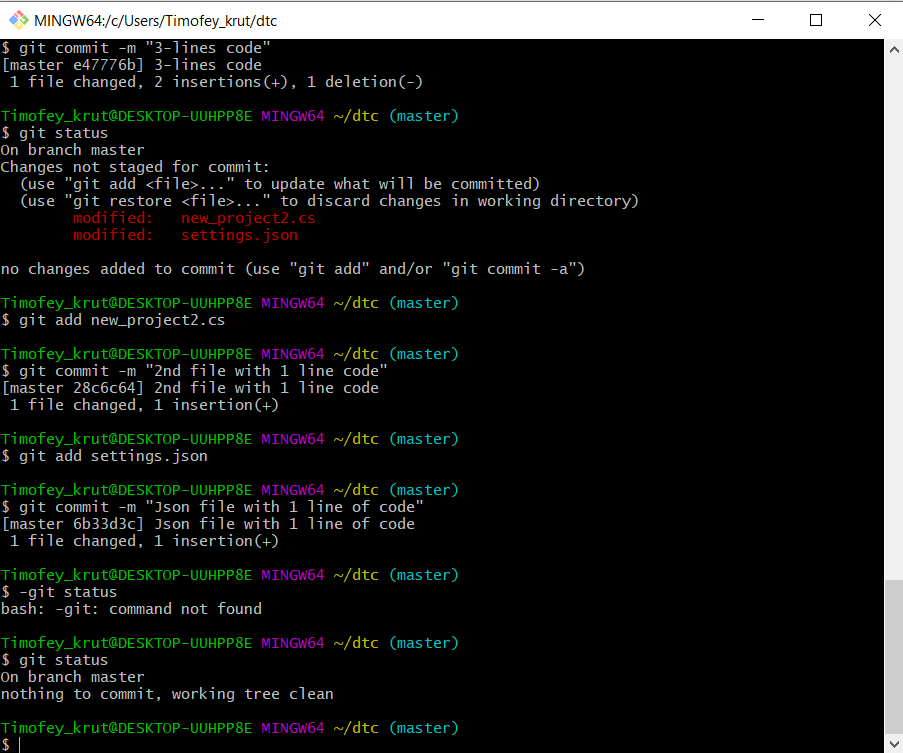
4. Изменение файла, новая проверка состояния репозитория.



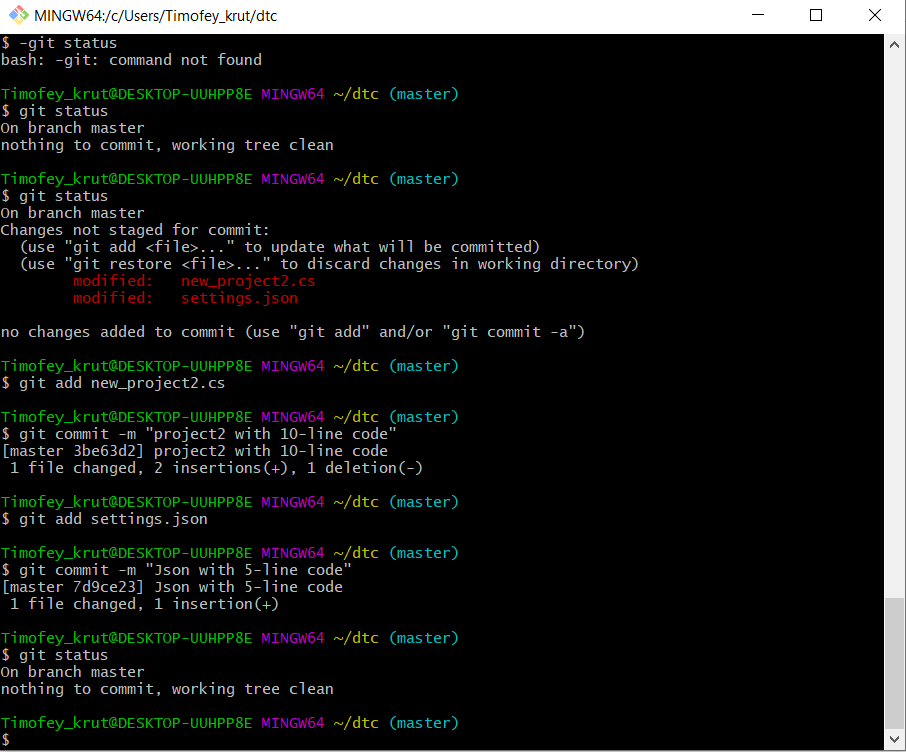
5. Индексирование изменений и добавление коммита.



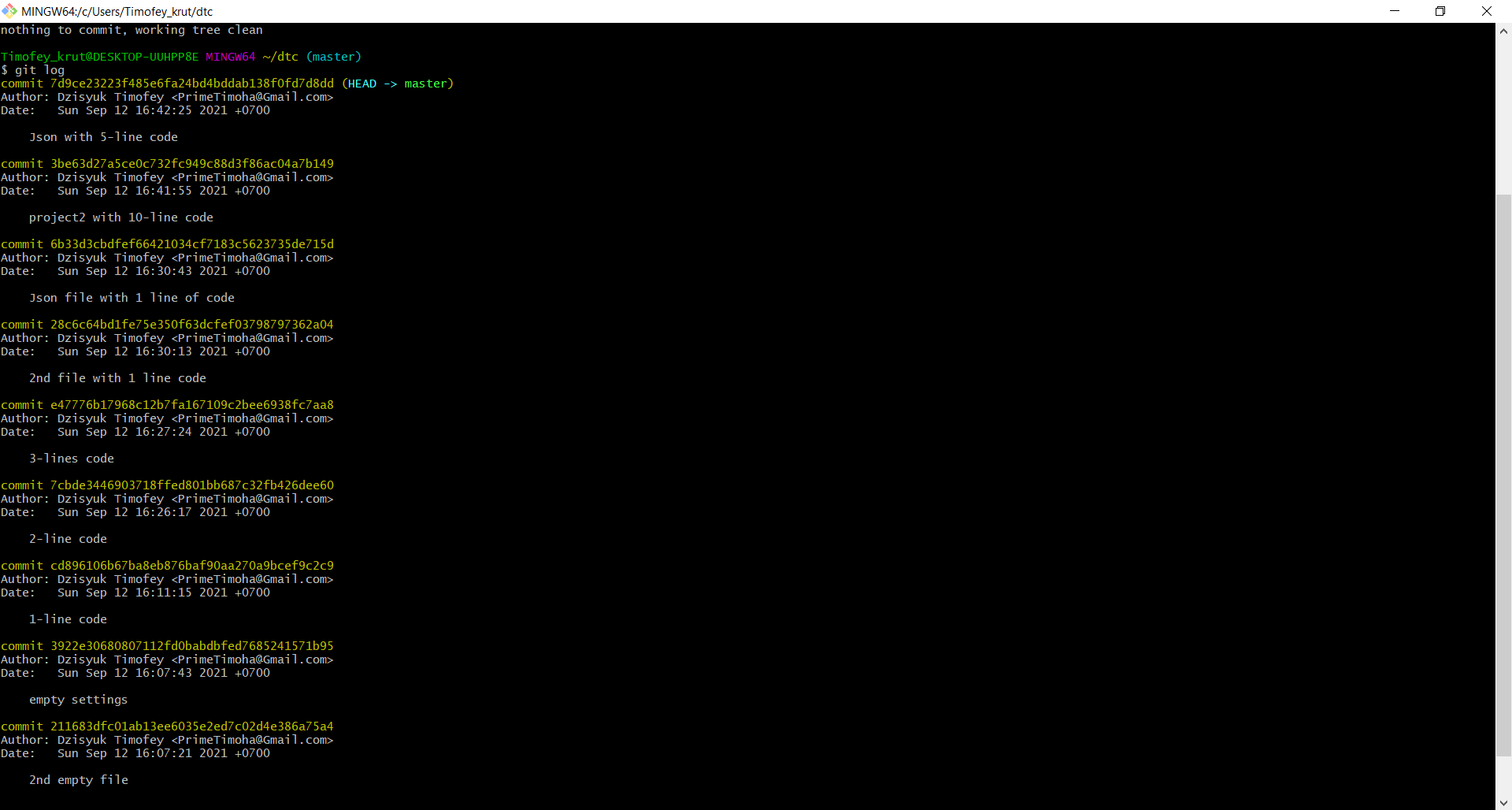
6. Индексирование новых изменений и добавление коммитов.



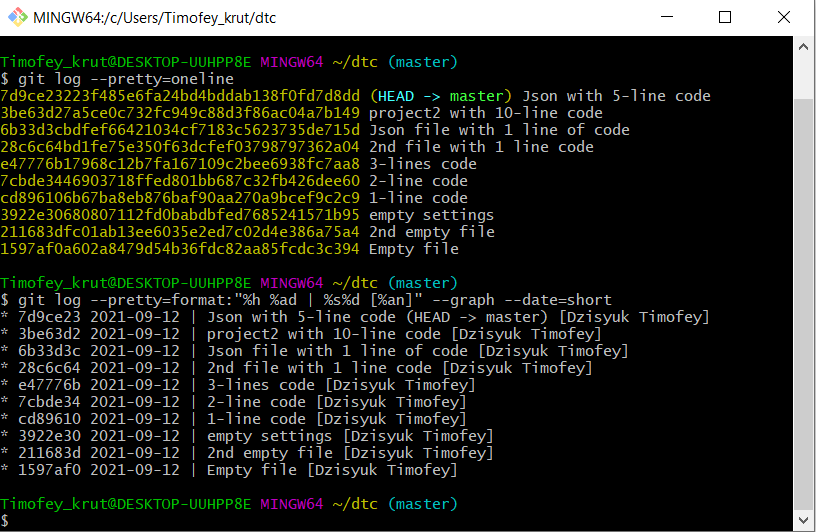
Индексирование новых изменений и добавление коммитов у двух оставшихся файлов.



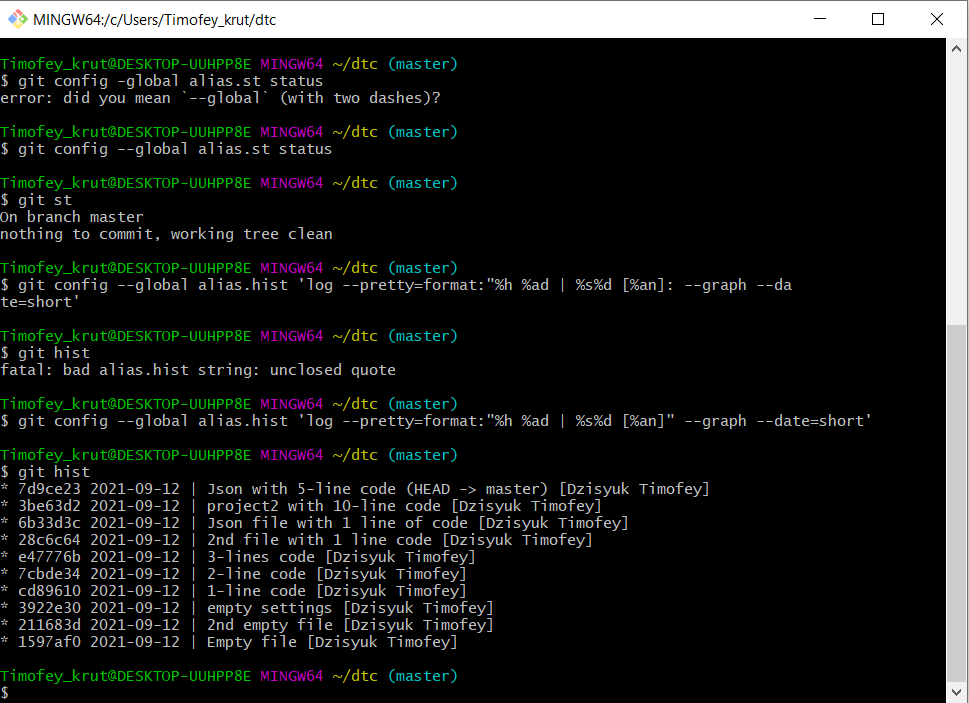
7. Просмотр истории проекта с помощью команды: git log



8. История в формате одной строки: git log --pretty=oneline  
Форматированная история проекта.

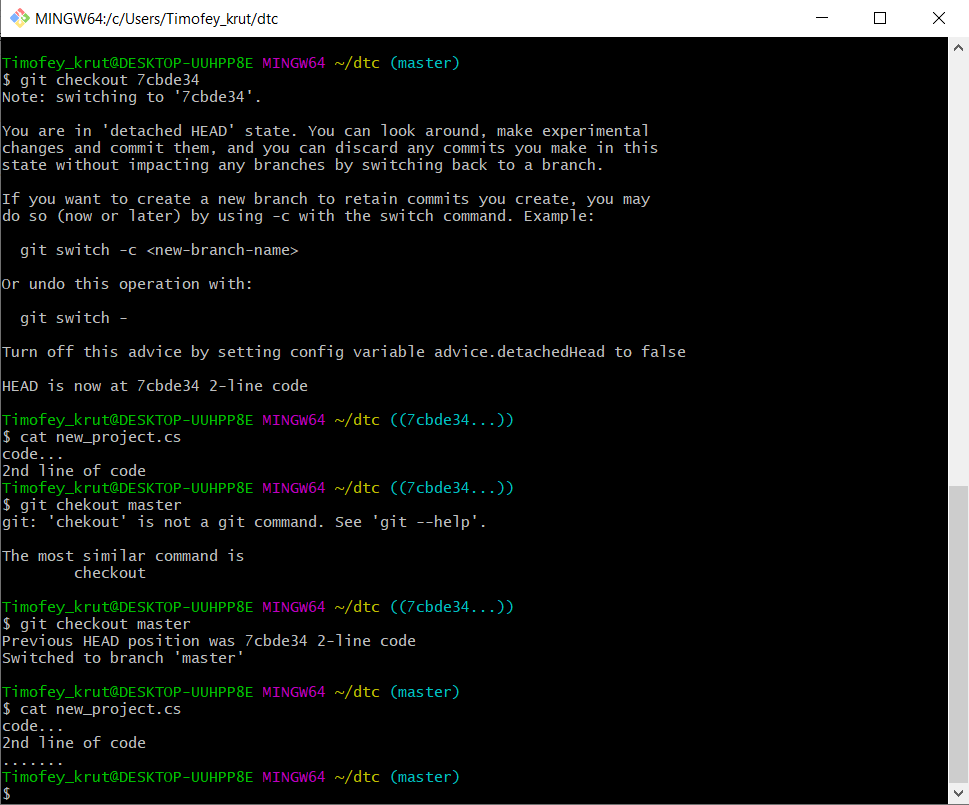


9. Использование алиасов.

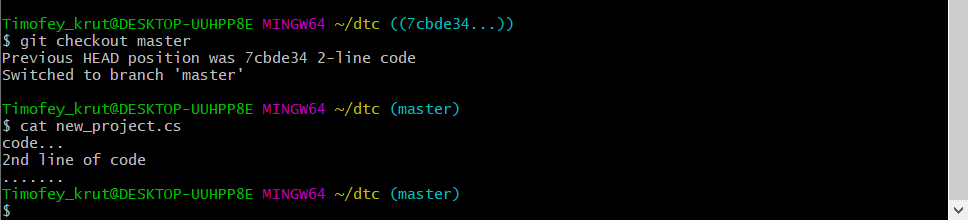


10. Возврат к прошлым версиям. Есть 2 варианта:

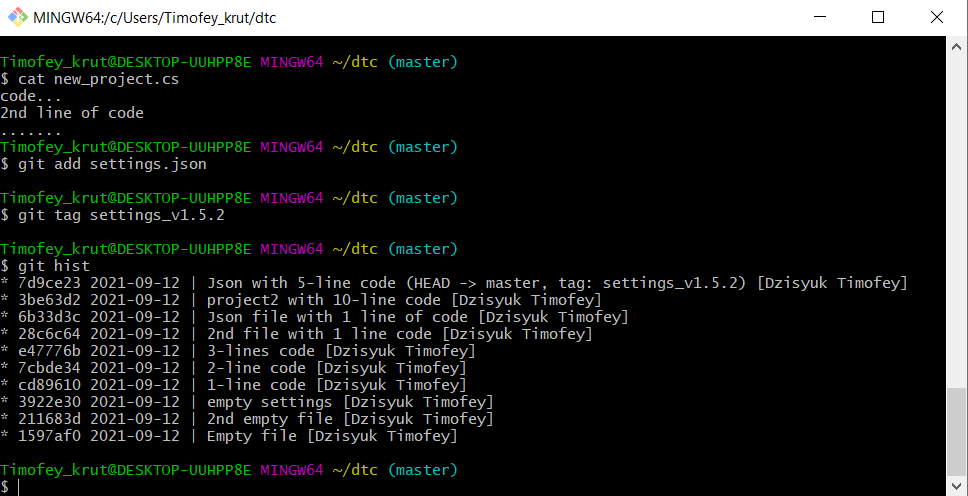
1-ый: вернуться к версии по ее индексу: git checkout <...>



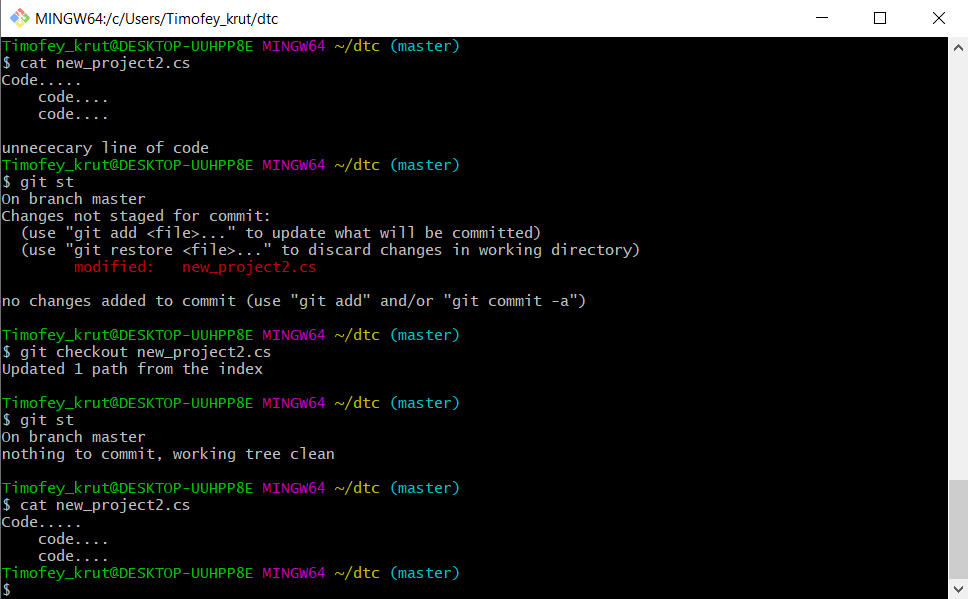
2-ой: Вернуться к последней версии: git checkout master



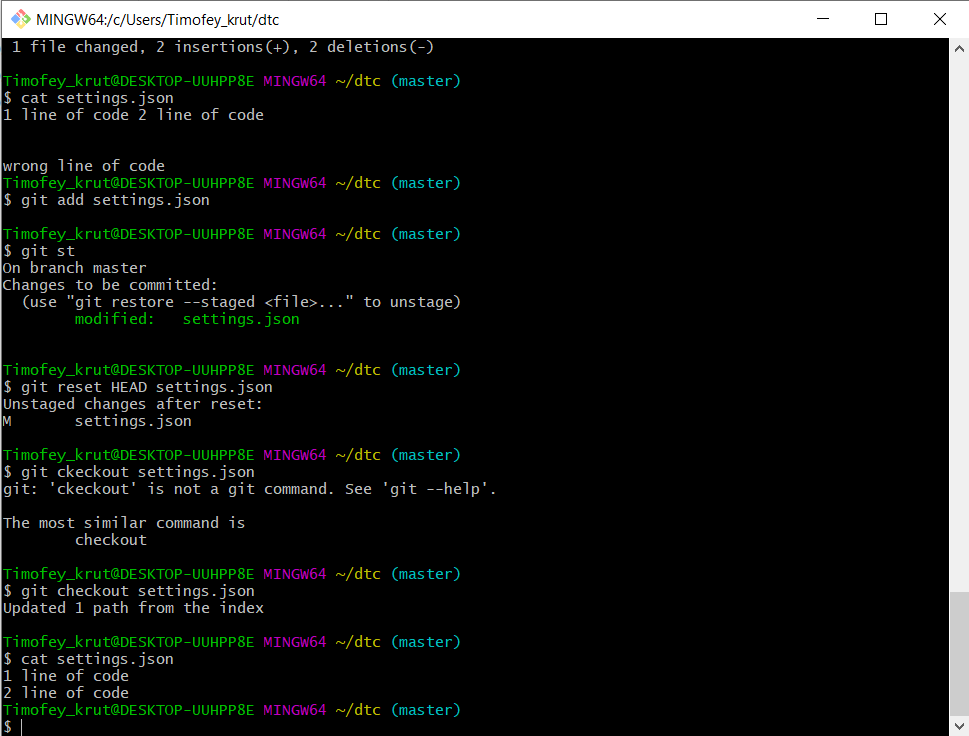
11. Тегирование версий с помощью команды: git tag <some\_tag>



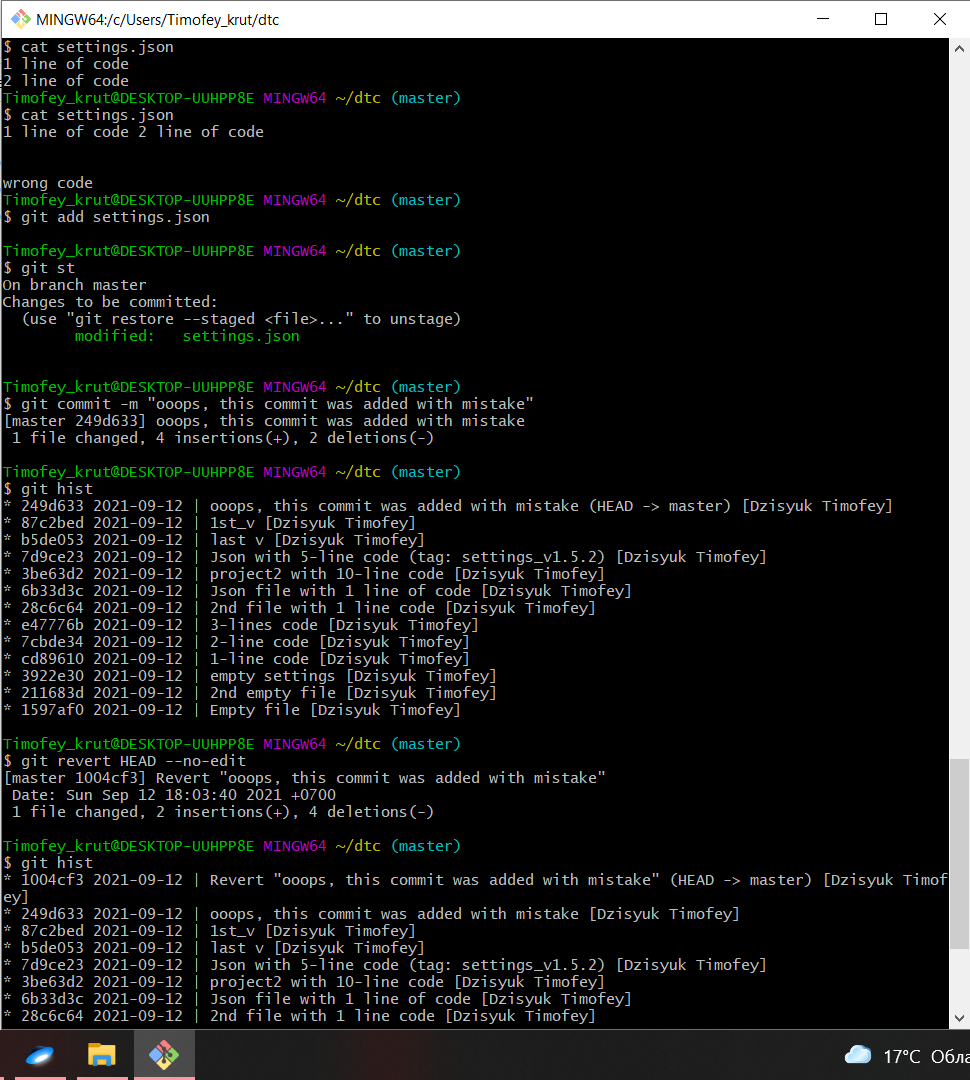
12. Отмена локальных изменений и возврат на последнюю индексированную версию при помощи команды: git checkout <file's name>

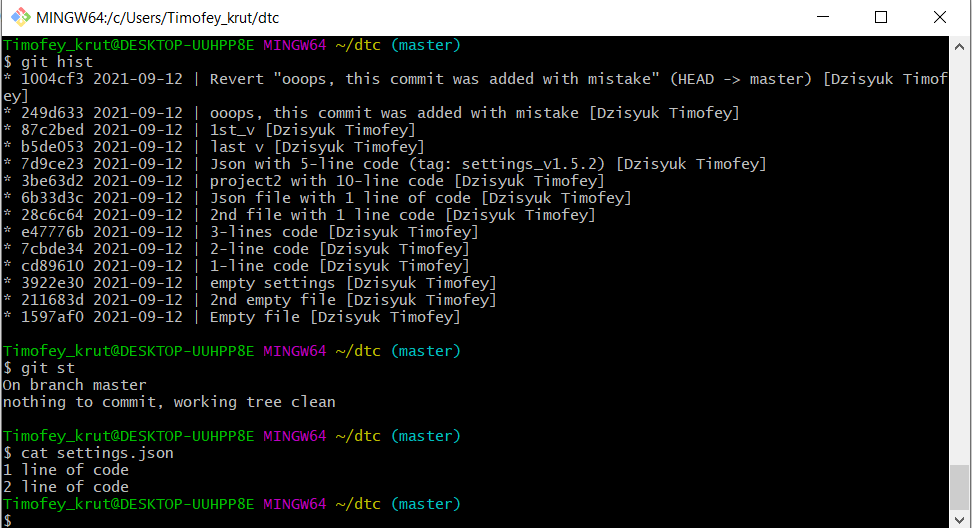


13. Отмена изменений для проиндексированного файла. Для этого нужно использовать команду: git reset HEAD <file's name>

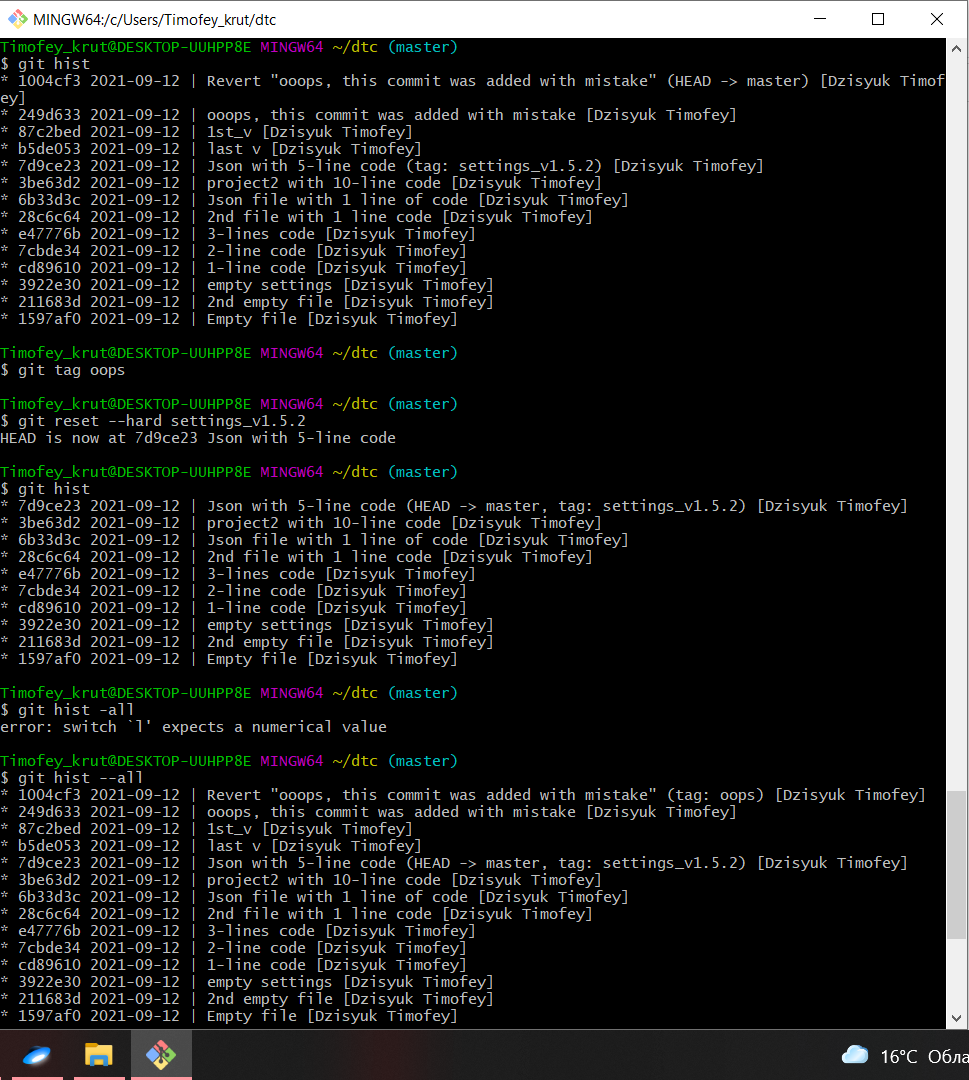


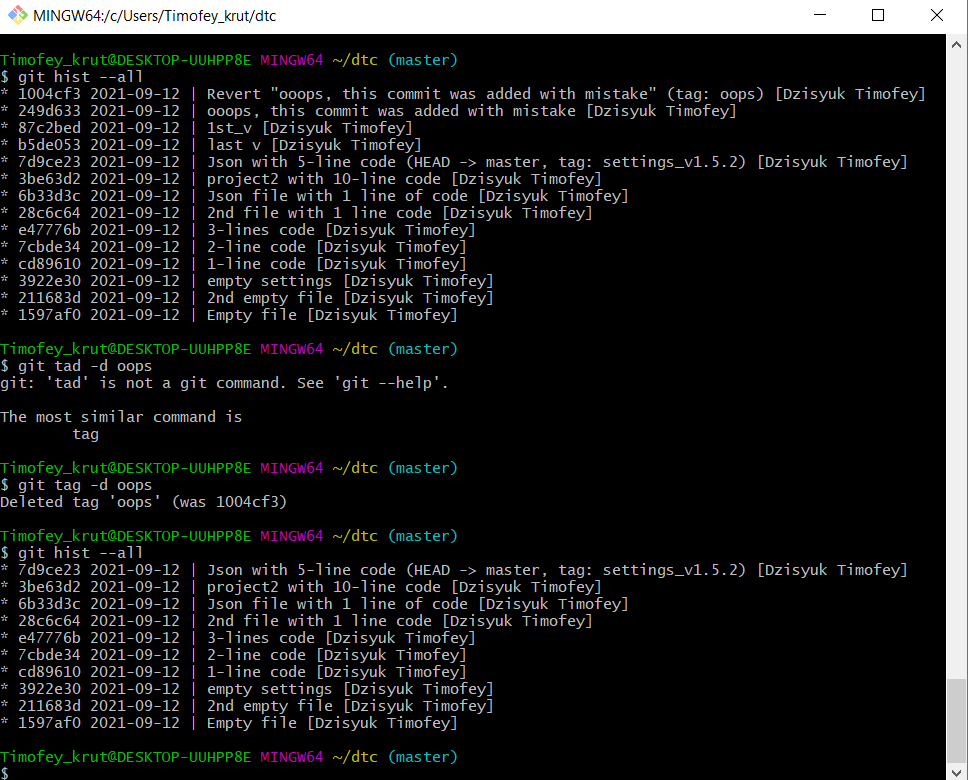
14. Отмена изменений для проиндексированного файла, с добавленным коммитом при помощи команды: git revert HEAD



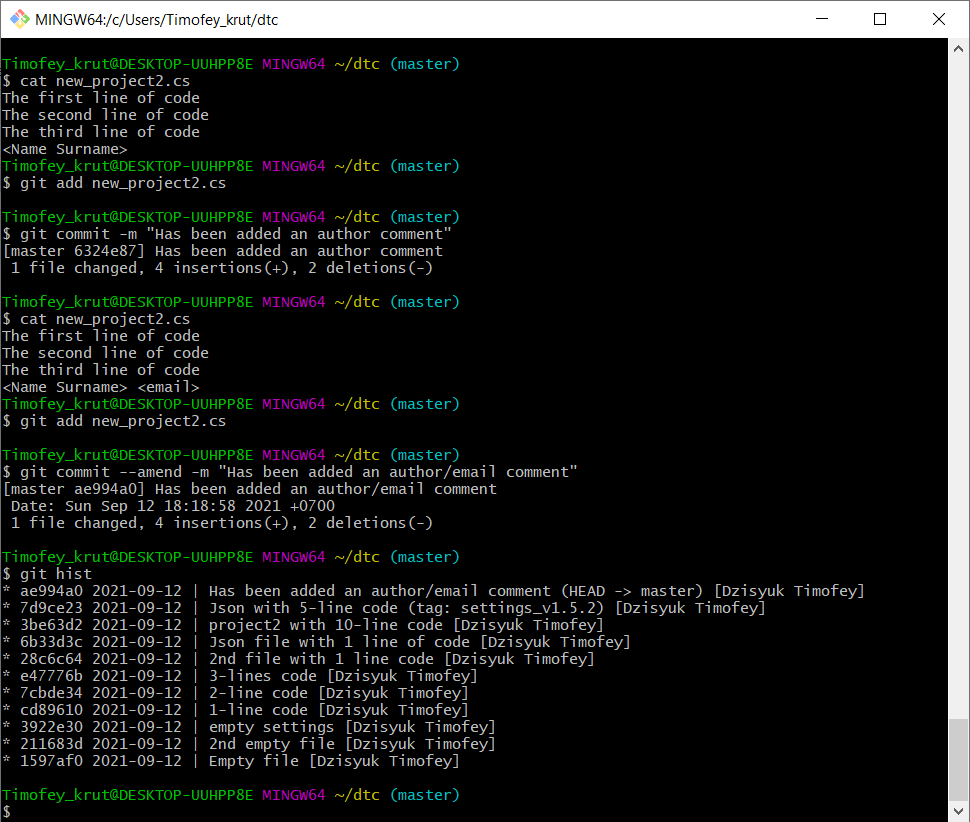


15. Удаление файла с коммитом из истории при помощи команды: git tag -d <tag>

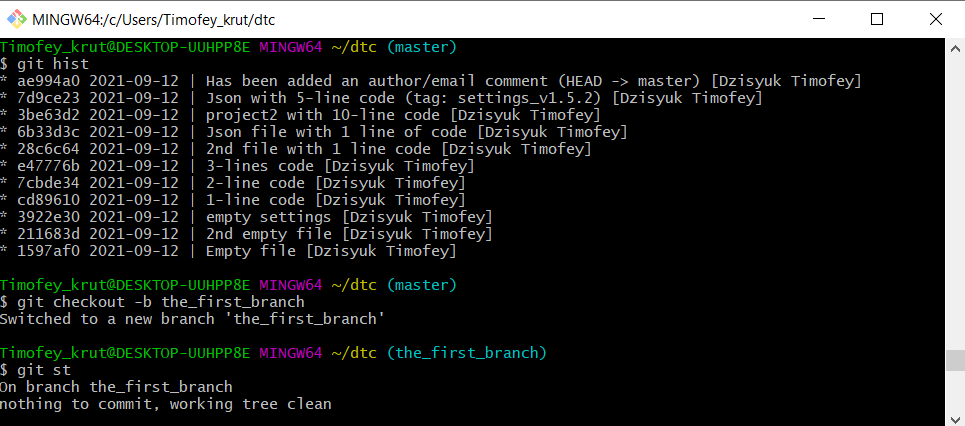


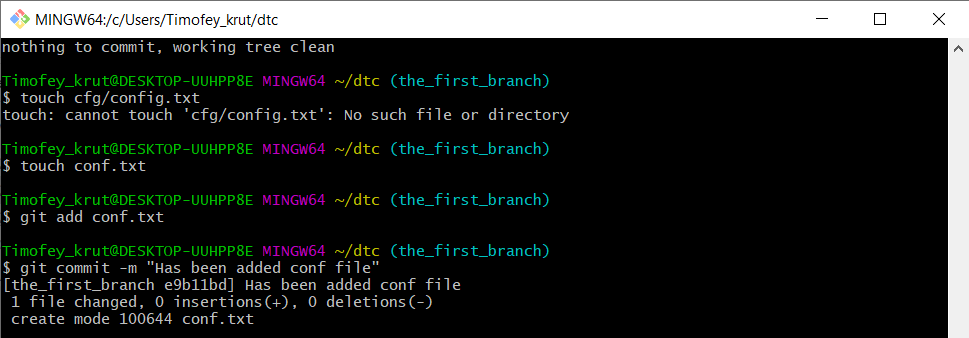


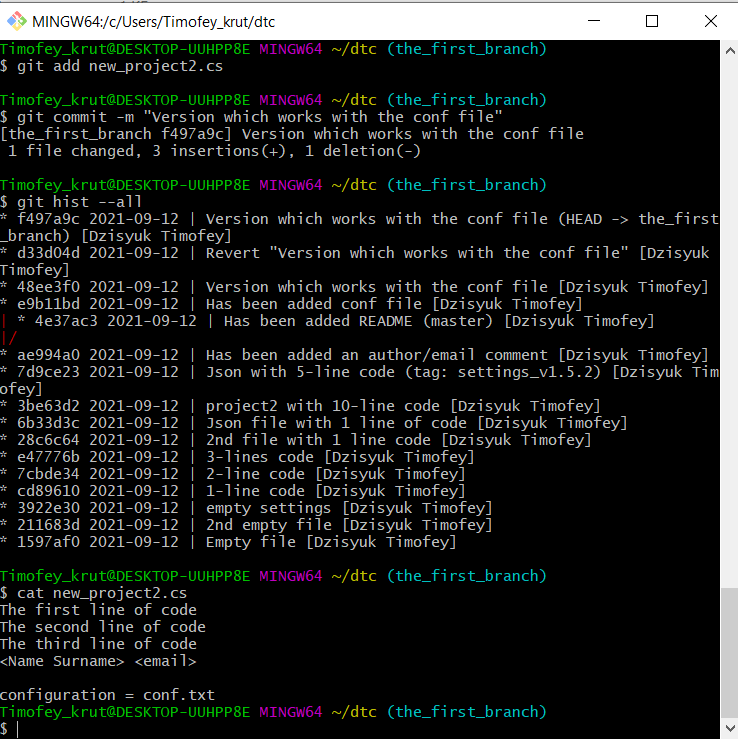
16. Исправление коммита.

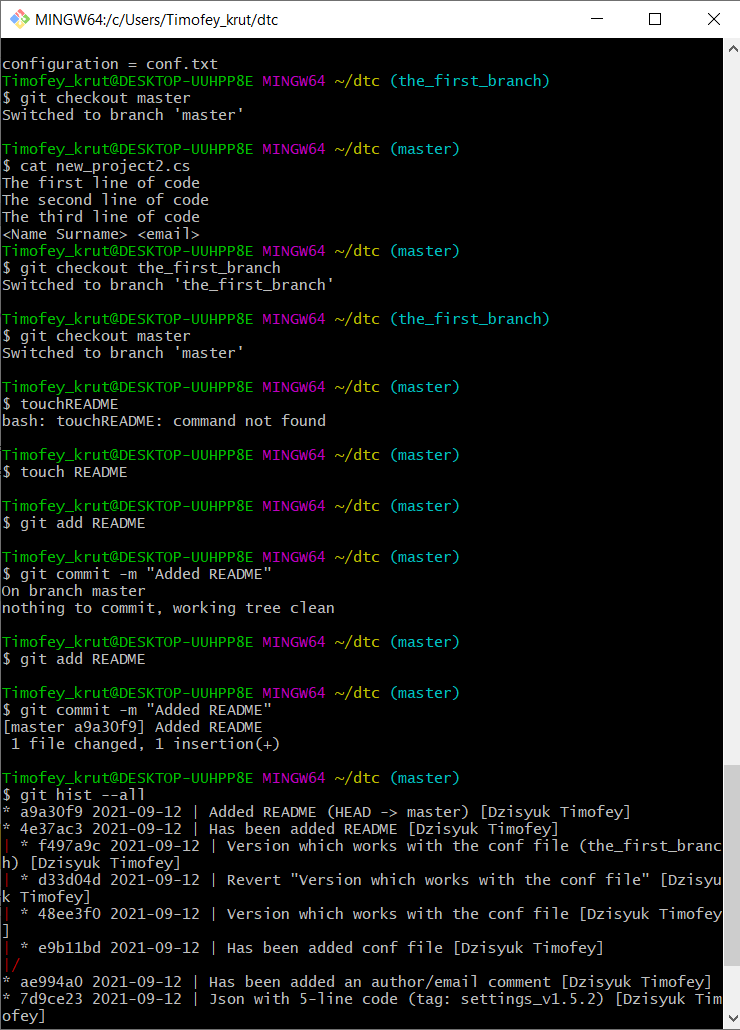


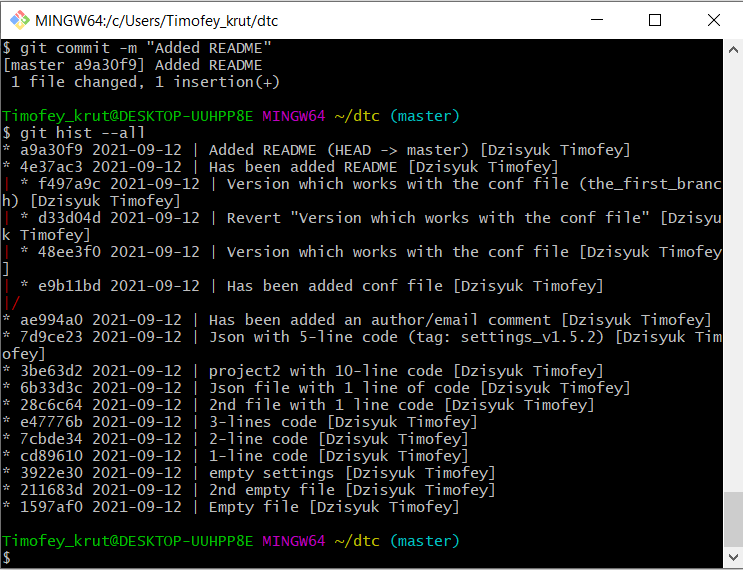
17. Создание ветки. Работа с ветками.



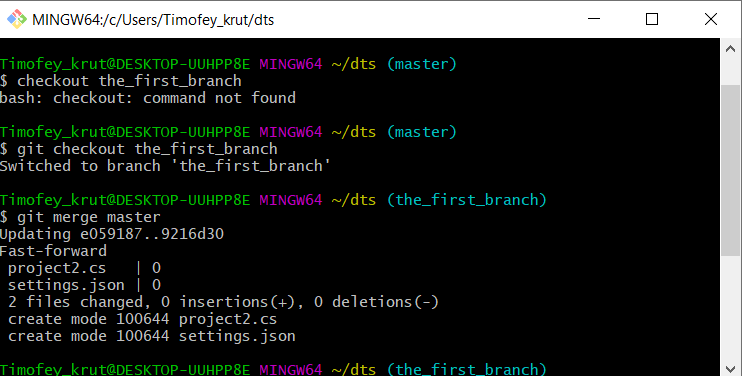


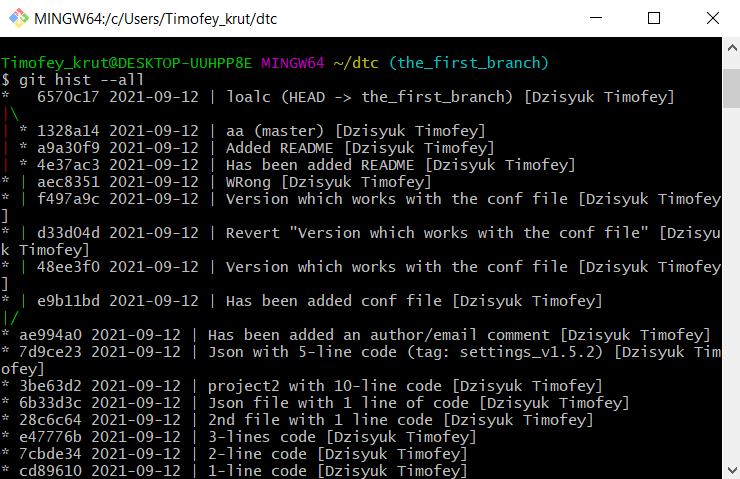


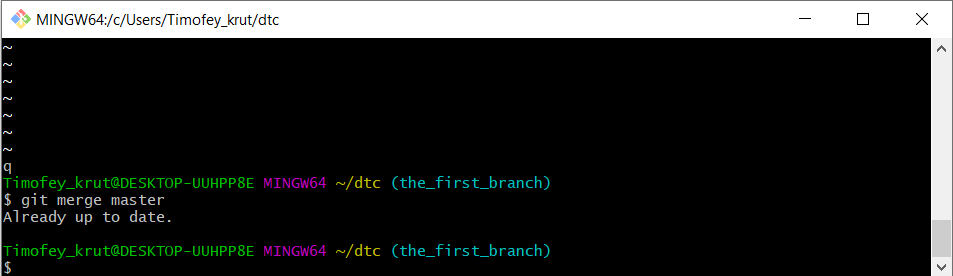




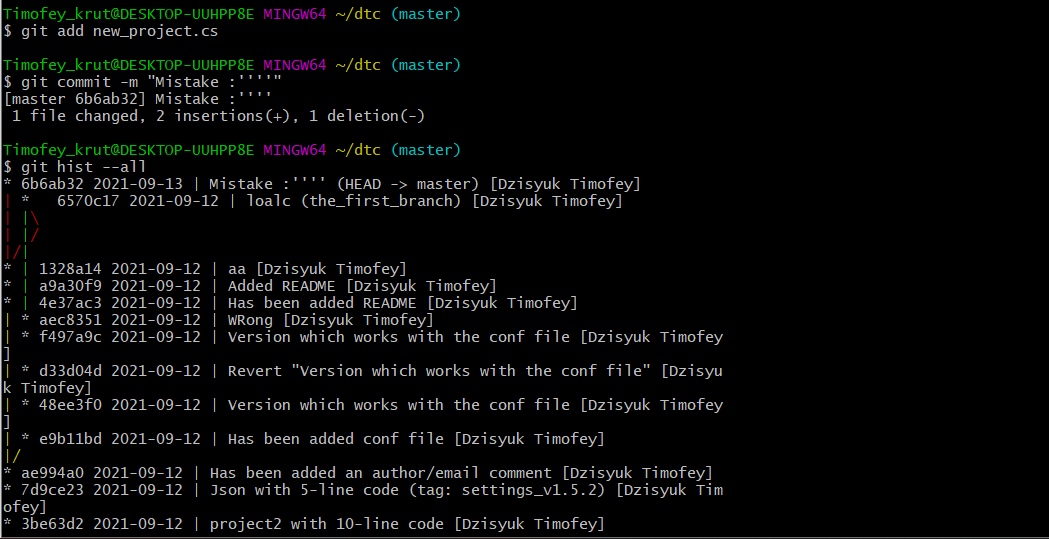
18. Слияние веток.



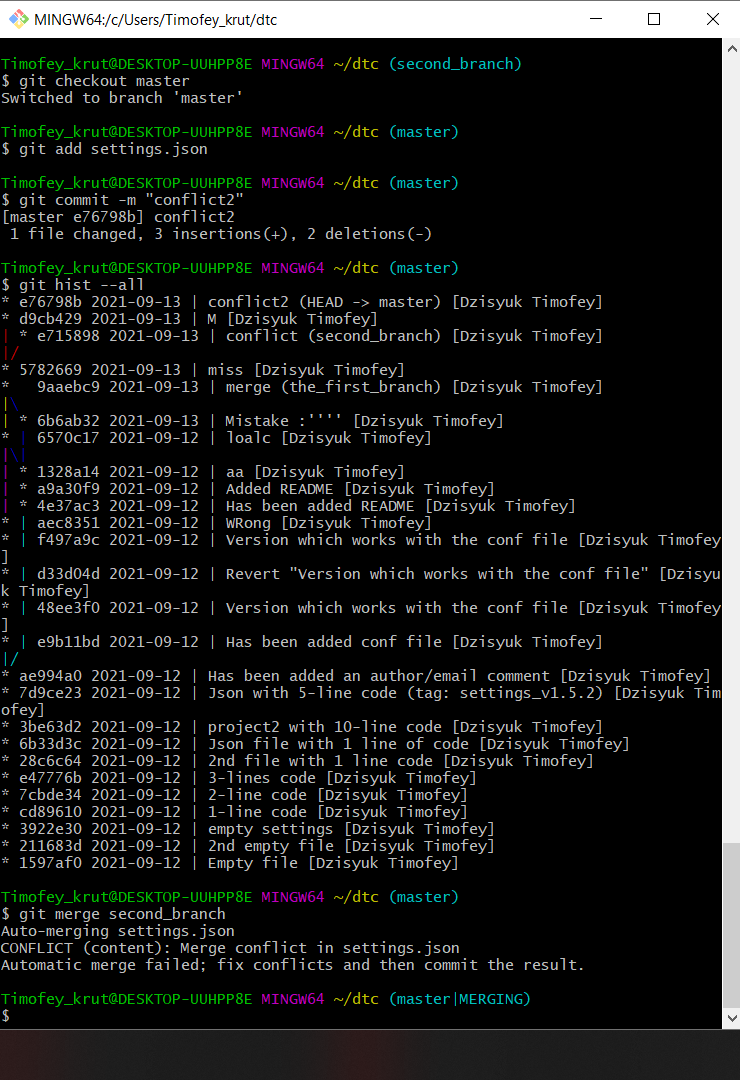


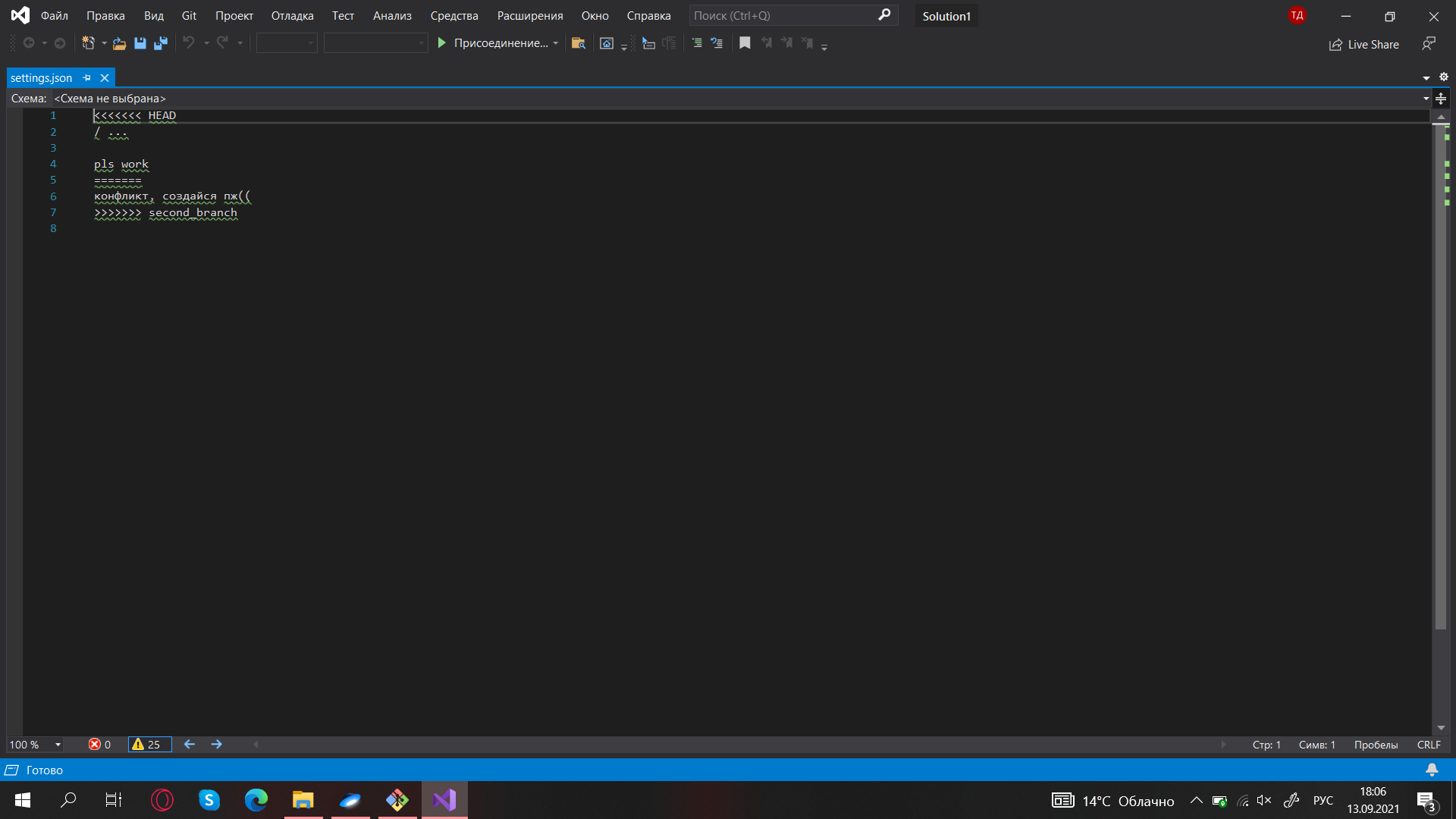


19. Конфликты.

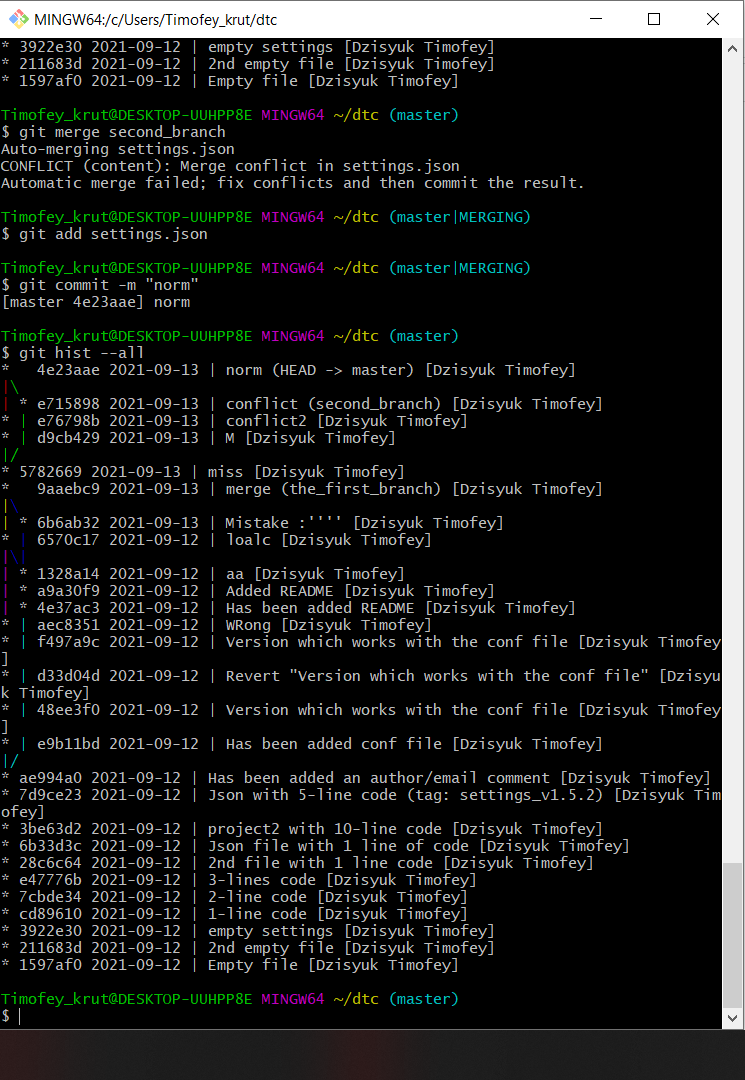
Первый пример конфликта.

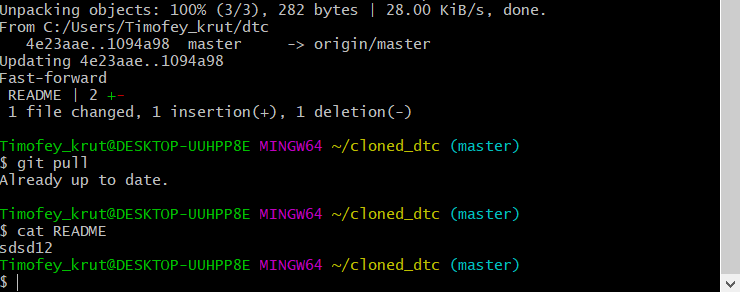
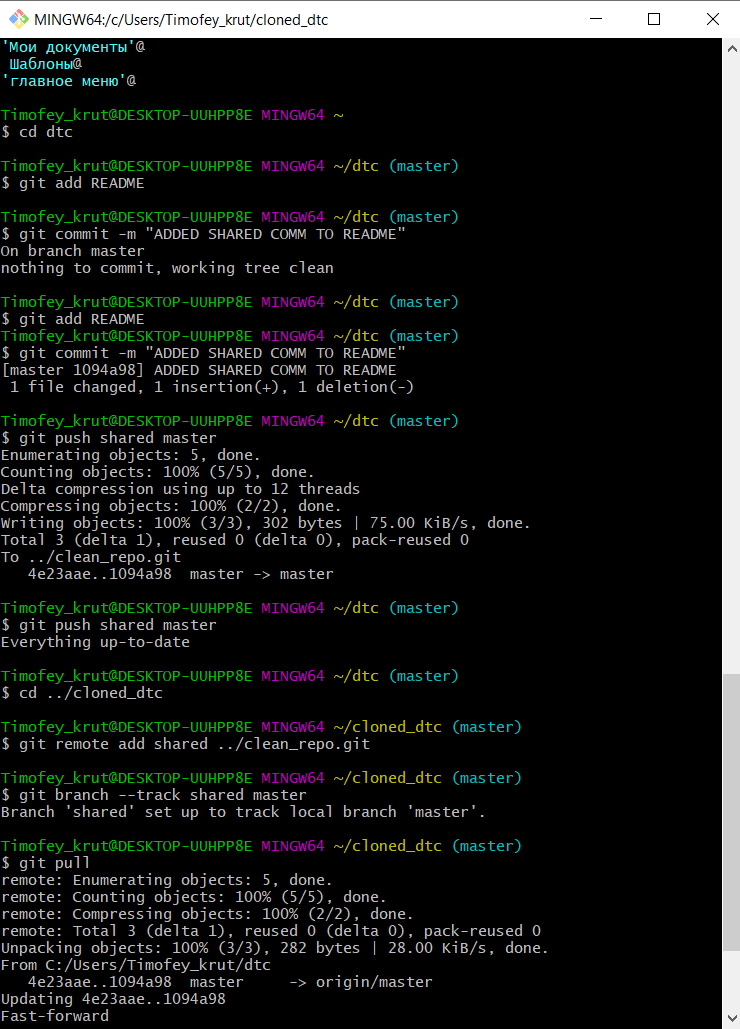
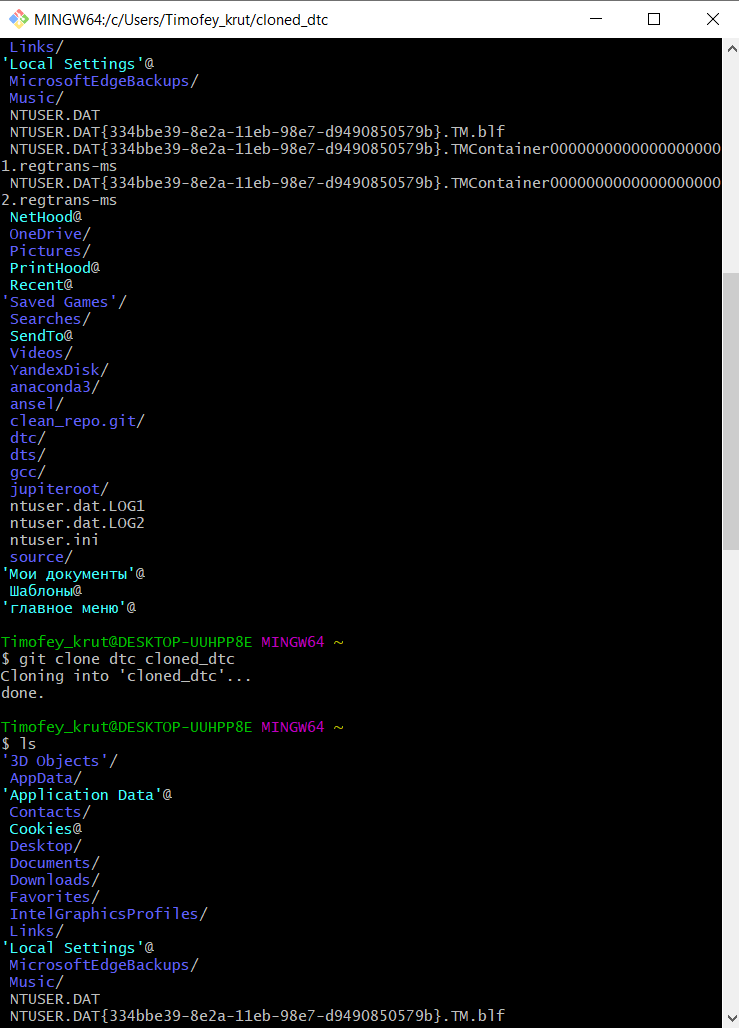
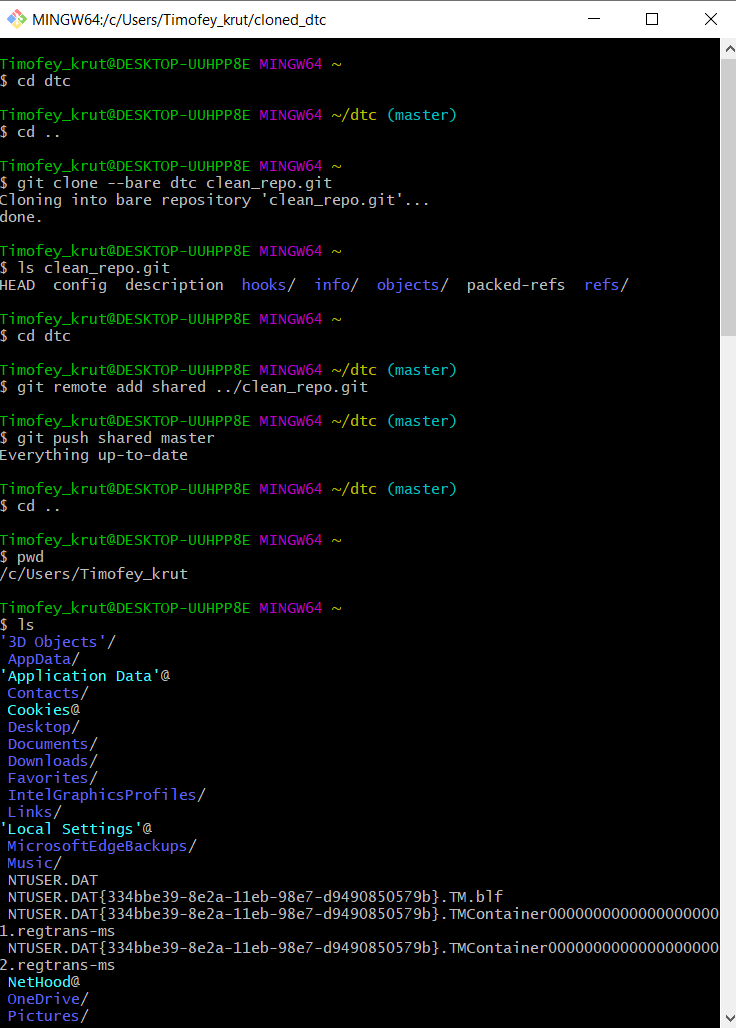
Второй пример конфликта.





20. Решение конфликта.



21. Создание чистого репозитория и работа с ним.

**Вывод**: Проделав лабораторную работу номер 3, я познакомился с ПО под названием Git. Я научился создавать репозитории для работы с файлами и их версиями до/после изменений. Git - очень удобное и незаменимое ПО, так как очень сильно облегчает работу с проектами, позволяя делать откаты до разных версий файла.

**Контрольные вопросы**

1. Git - ПО, система контроля версий. Эта система позволяет следить за изменениями файлов, хранит разные версии этих файлов, а так же позволяет вернуться к любой сохраненной версии файла.

2. Система контроля версий позволяет отслеживать изменения файлов, показывая кто и когда изменял файл, позволяет хранить разные версии одного файла, а так же обращаться к ним в любое время.

3. Коммит - команда с помощью которой файлу дается название и он помещается в репозиторий.

4. Чтобы посмотреть историю коммитов, достаточно воспользоваться командой:

*git log*

5. Для создания пустого Git репозитория, необходимо ввести команду : *git init* в рабочей области.

6. Git состоит из двух разделов. первый - репозиторий, место где хранятся все за сохраненные операцией commit файлы. И второй - рабочая область (working copy), набор файлов с которым мы работаем в данный момент.

7. Для создания новой ветки необходимо ввести команду:

*git checkout -b <branch\_name>*

для соединения ветвей необходима команда:

*git merge <branch\_name>*

8. Staging area - промежуточный этап между working space и репозиторием. Позволяет хранить и изменять файлы, и при этом непосредственно отправлять их в репозиторий в разных коммитах.

9. Для сравнения необходимо использовать команду:

*git diff*