МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт Информационных технологий (ИТ)

Кафедра Математического обеспечения и стандартизации информационных технологий (МОСИТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 1
по дисциплине
«Технология разработки программных приложений»
Тема: «Система контроля версий»

Выполнил студент группы ИКБО-20-21 Хитров Н.С.

Принял к.т.н. доцент Петренко А.А.

Практическая работа выполнена «4» марта 2023г. *(подпись студента)*

«Зачтено» «__»_____2023г. *(подпись руководителя)*

Москва 2023

ОГЛАВЛЕНИЕ

Часть 1. Основные команды git

1. Установите и настройте клиент git на своей рабочей станции

На рисунке 1 показан процесс настройки конфигурации git.

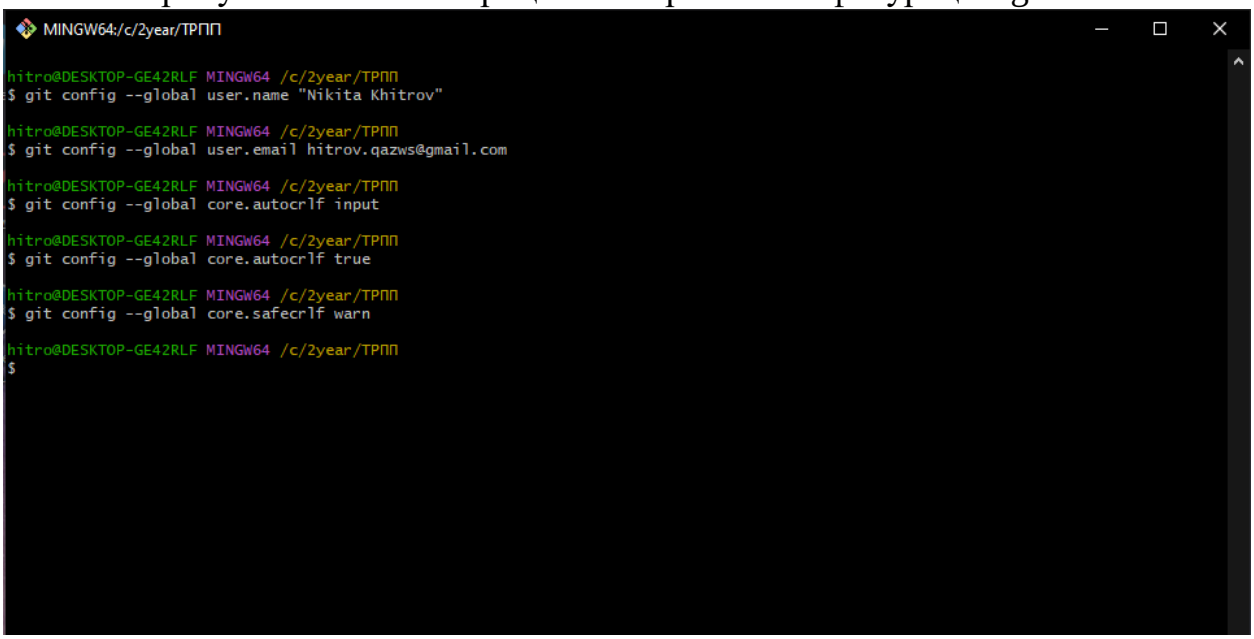


Рисунок 1 – Настройка конфигурации git

2. Создайте локальный репозиторий и добавьте в него несколько файлов.

На рисунке 2 показан процесс инициализации пустого репозитория и добавления файлов в папку проекта.

Рисунок 2 – Инициализация репозитория git

3. Внесите изменения в один из файлов.

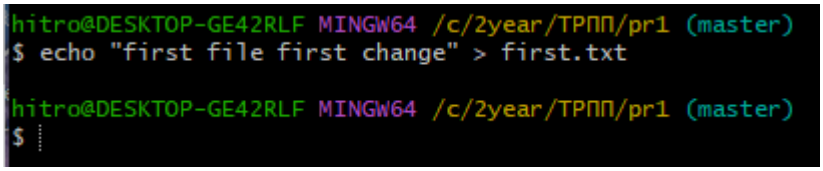


Рисунок 3 – Внесение изменений в файл

4. Проиндексируйте изменения и проверьте состояние.

Рисунок 4 – Индексация изменений

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

На рисунке 5 показан процесс изменения файла index.html, индексация изменений и создание коммита.

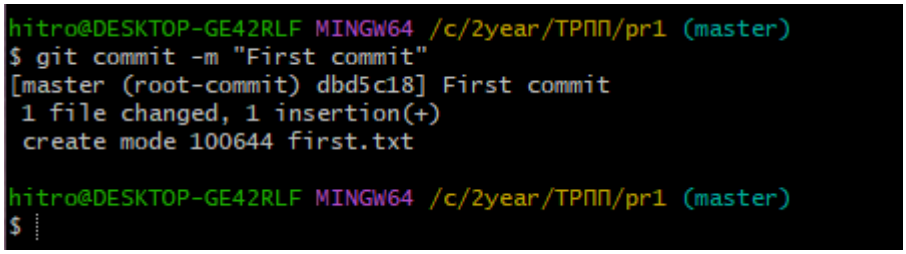


Рисунок 5 – изменение файла и создание коммита

6. 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.

На рисунке 6 показан процесс создания коммитов с разными изменениями одного файла.

```
hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/pr1 (master)
$ echo "second file first change" > second.txt

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/pr1 (master)
$ git add second.txt
warning: in the working copy of 'second.txt', LF will be replaced by CRLF the next time Git touches it

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/pr1 (master)
$ echo "second file second change" > second.txt

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/pr1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   second.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   second.txt


hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/pr1 (master)
$ git commit -m "Second commit"
[master 15f5e7e] Second commit
 1 file changed, 1 insertion(+)
 create mode 100644 second.txt

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/pr1 (master)
$ git add second.txt
warning: in the working copy of 'second.txt', LF will be replaced by CRLF the next time Git touches it

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/pr1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   second.txt


hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/pr1 (master)
$ git commit -m "Third commit second.txt_v2"
[master 08f0617] Third commit second.txt_v2
 1 file changed, 1 insertion(+), 1 deletion(-)

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/pr1 (master)
$
```

Рисунок 6 – Изменение другого файла, индесация, повторное изменение и тд.

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

На рисунке 7 показана пользовательская конфигурация команд.

config

```
[core]
    repositoryformatversion = 0
    filemode = false
    bare = false
    logallrefupdates = true
    symlinks = false
    ignorecase = true

[alias]
    co = checkout
    cm = commit
    st = status
    br = branch
    hist = log --graph --date=short --pretty=format:\"%h %ad | %s%d [%an]\"
    type = cat-file -t
    dump = cat-file -p
```

Рисунок 7 – пользовательская конфигурация команд

На рисунке 8 показана пользовательская конфигурация команд.

```
$ git hist
* 08f0617 2023-02-16 | Third commit second.txt_v2 (HEAD -> master) [Nikita Khitrov]
* 15f5e7e 2023-02-16 | Second commit [Nikita Khitrov]
* dbd5c18 2023-02-16 | First commit [Nikita Khitrov]

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/pr1 (master)
$
```

Рисунок 8 – История коммитов

1. **Верните рабочий каталог к одному из предыдущих состояний.**

На рисунке 9 показан процесс возврата репозитория к одному из предыдущих состояний.

```
hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/pr1 (master)
$ git co 15f5e7e
Note: switching to '15f5e7e'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 15f5e7e Second commit

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/pr1 ((15f5e7e...))
$
```

Рисунок 9 – Возврат репозитория к предыдущему состоянию

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

На рисунке 10 показан процесс создания тегов для коммита.

```
hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/pr1 ((v1))
$ git co dbd5c18
Previous HEAD position was 15f5e7e Second commit
HEAD is now at dbd5c18 First commit

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/pr1 ((dbd5c18...))
$ git tag v1_1

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/pr1 ((v1_1))
$ git co 15f5e7e
Previous HEAD position was dbd5c18 First commit
HEAD is now at 15f5e7e Second commit

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/pr1 ((v1))
$ git tag v1_2

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/pr1 ((v1))
$ git co 08f0617
Previous HEAD position was 15f5e7e Second commit
HEAD is now at 08f0617 Third commit second.txt_v2

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/pr1 ((08f0617...))
$ git tag v2_1

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/pr1 ((v2_1))
$
```

Рисунок 10 – Создание тегов для коммита

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

На рисунке 11 показан процесс отмены изменений до индексации.

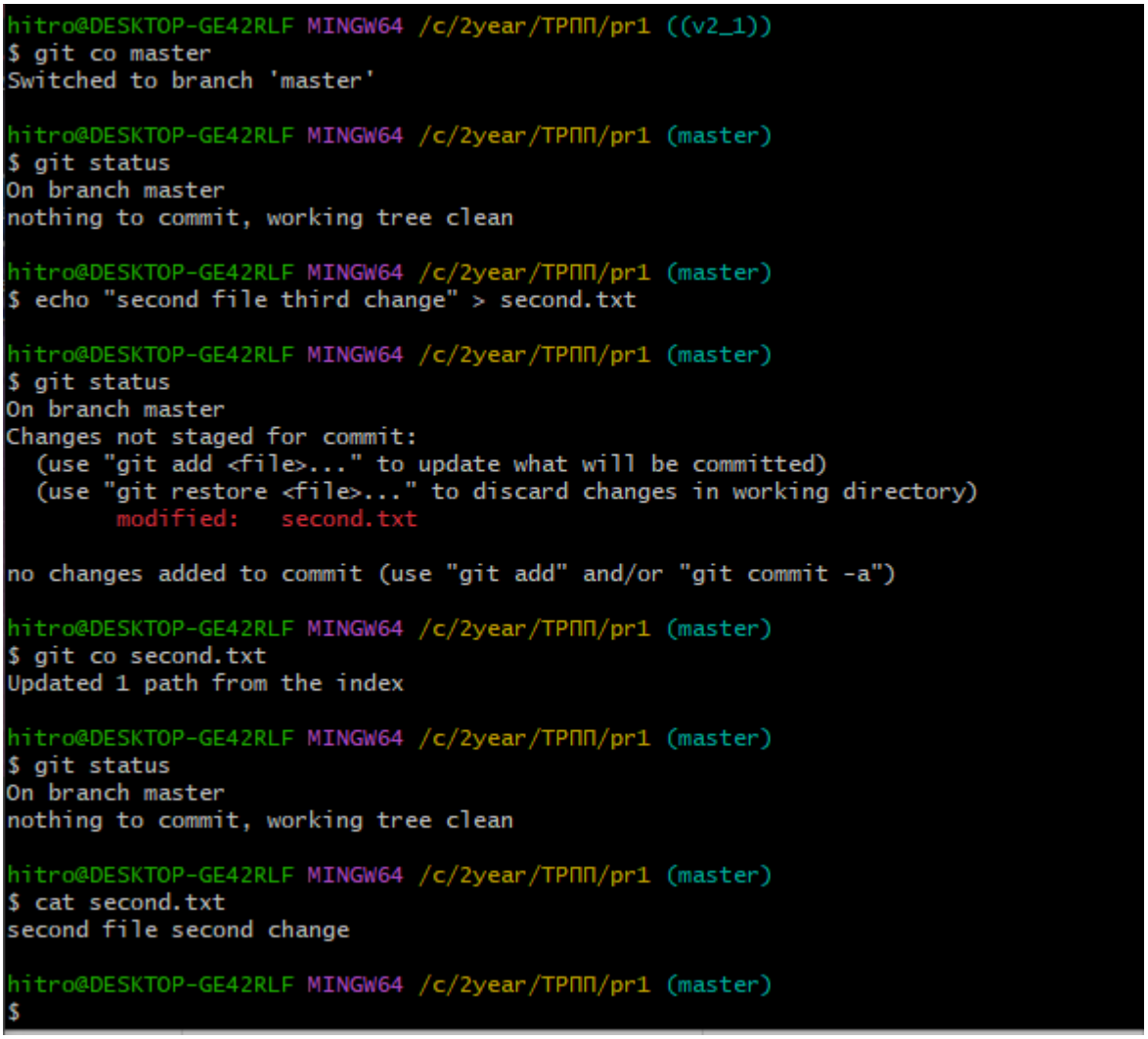


Рисунок 11 – Отмена изменений до индексации

На рисунке 12 показан процесс отмены изменений после индексации.

Рисунок 12 – Отмена проиндексированных изменений

4. Отмените один из коммитов в локальном репозитории.

На рисунке 14 показан процесс создания нежелаемого коммита.

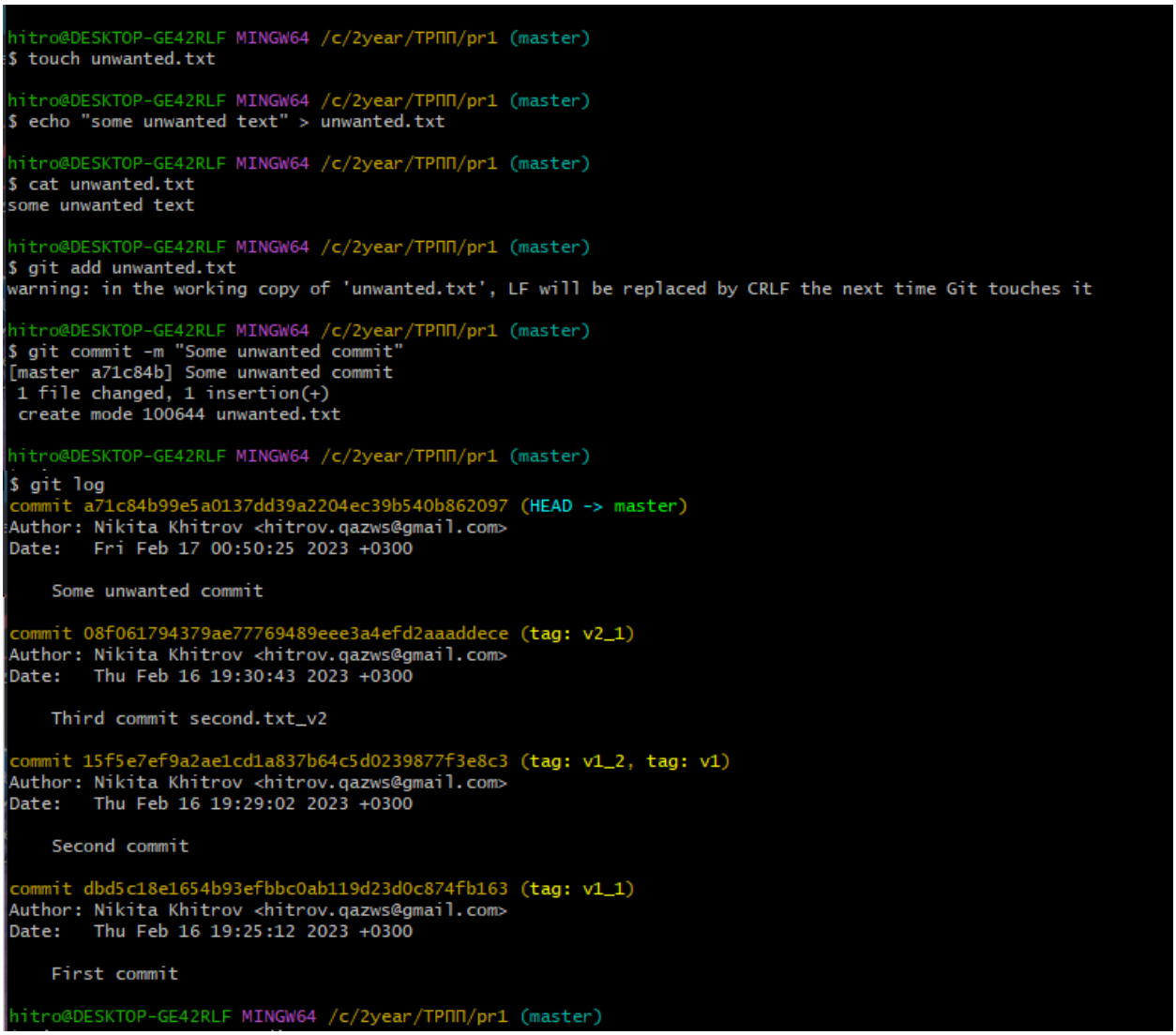


Рисунок 14 – создание нежелаемого коммита

На рисунке 15 показан процесс удаления последнего коммита.

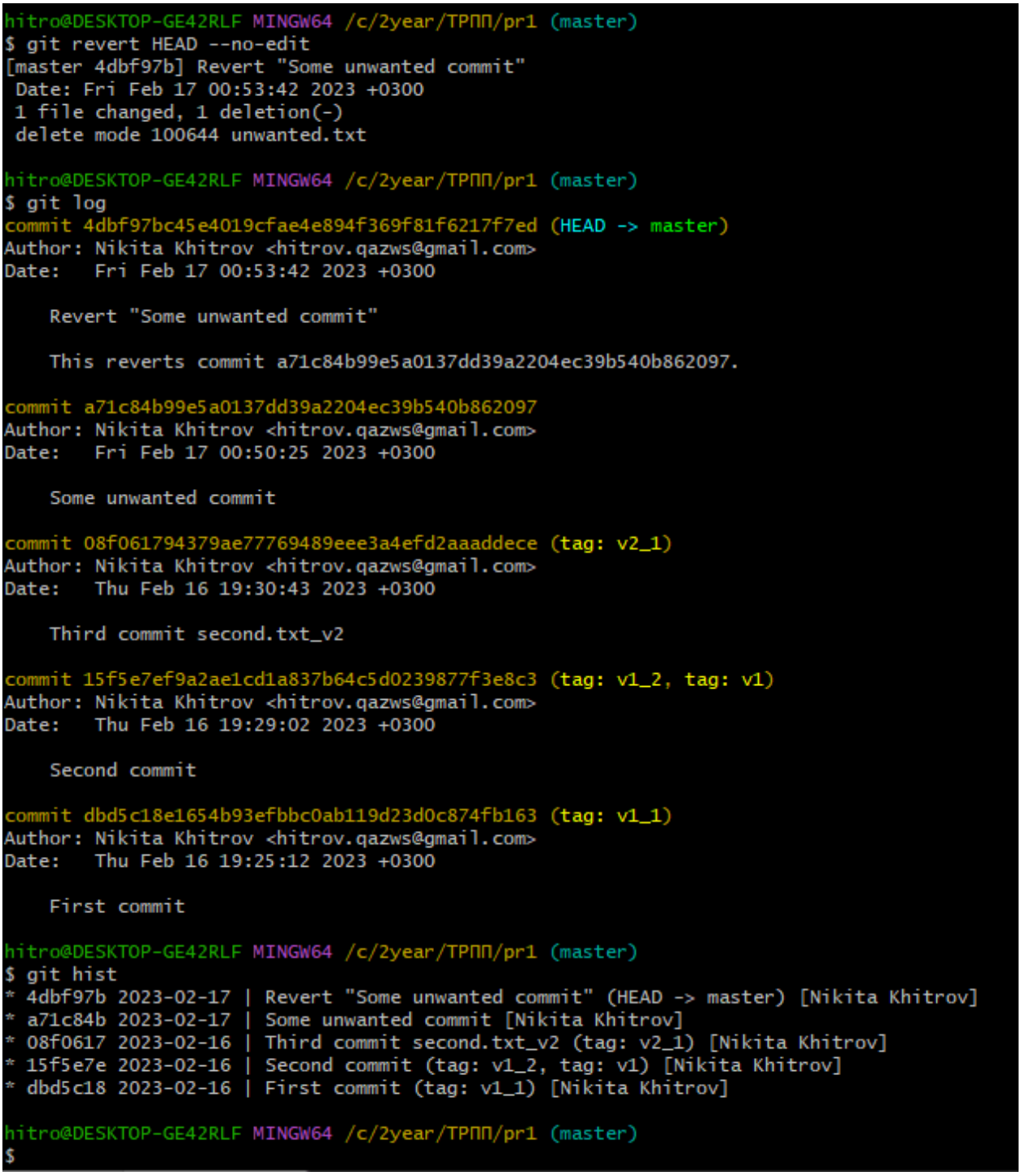


Рисунок 15 – Удаление последнего коммита

Часть 2. Системы управления репозиториями

Вариант 3

1. Создайте аккаунт на GitHub

На рисунке 16 показан созданный аккаунт на GitHub.

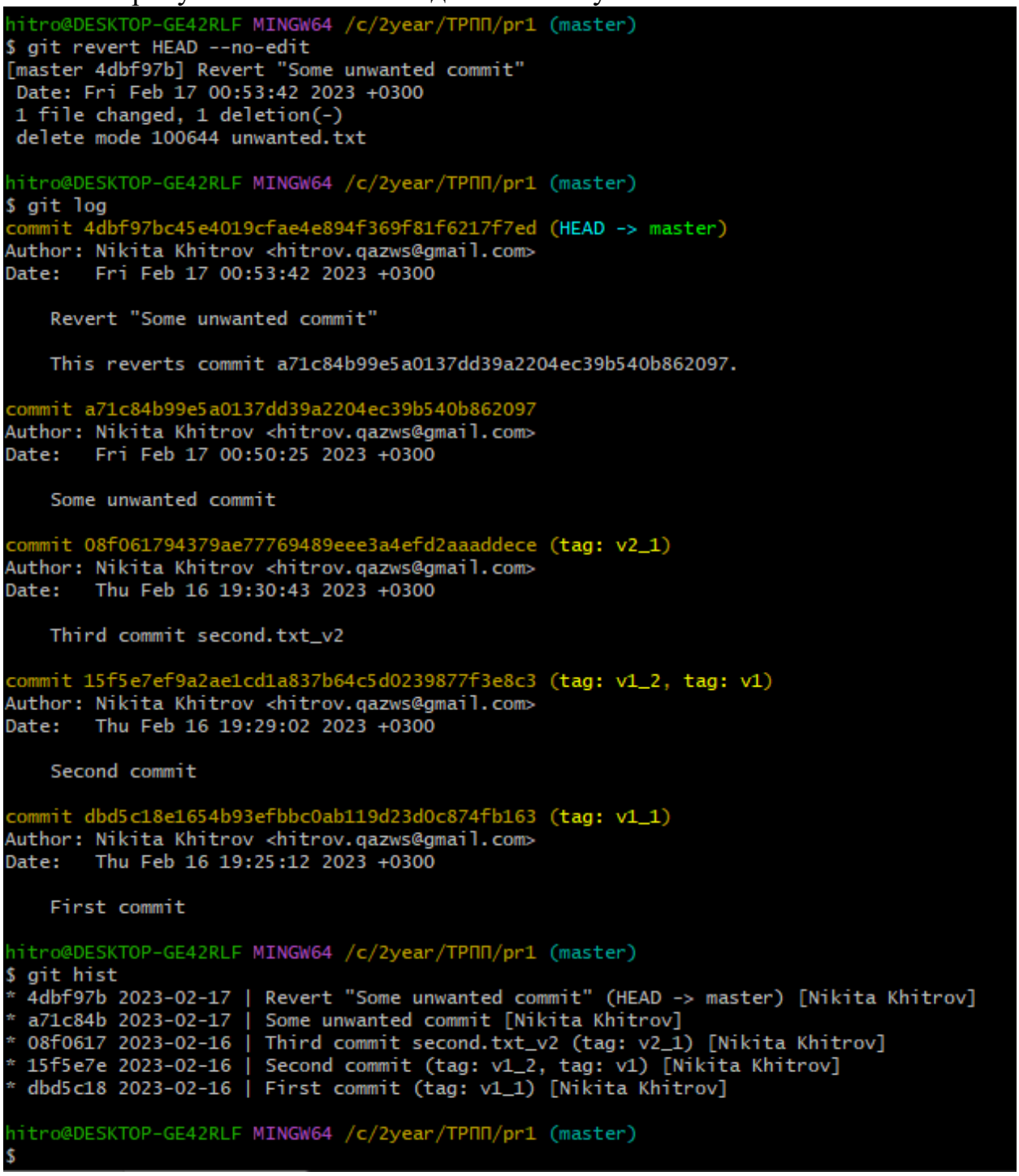


Рисунок 16 – Аккаунт GitHub

2. Создайте репозиторий согласно варианту

На рисунке 17 показан процесс создания удалённого репозитория.

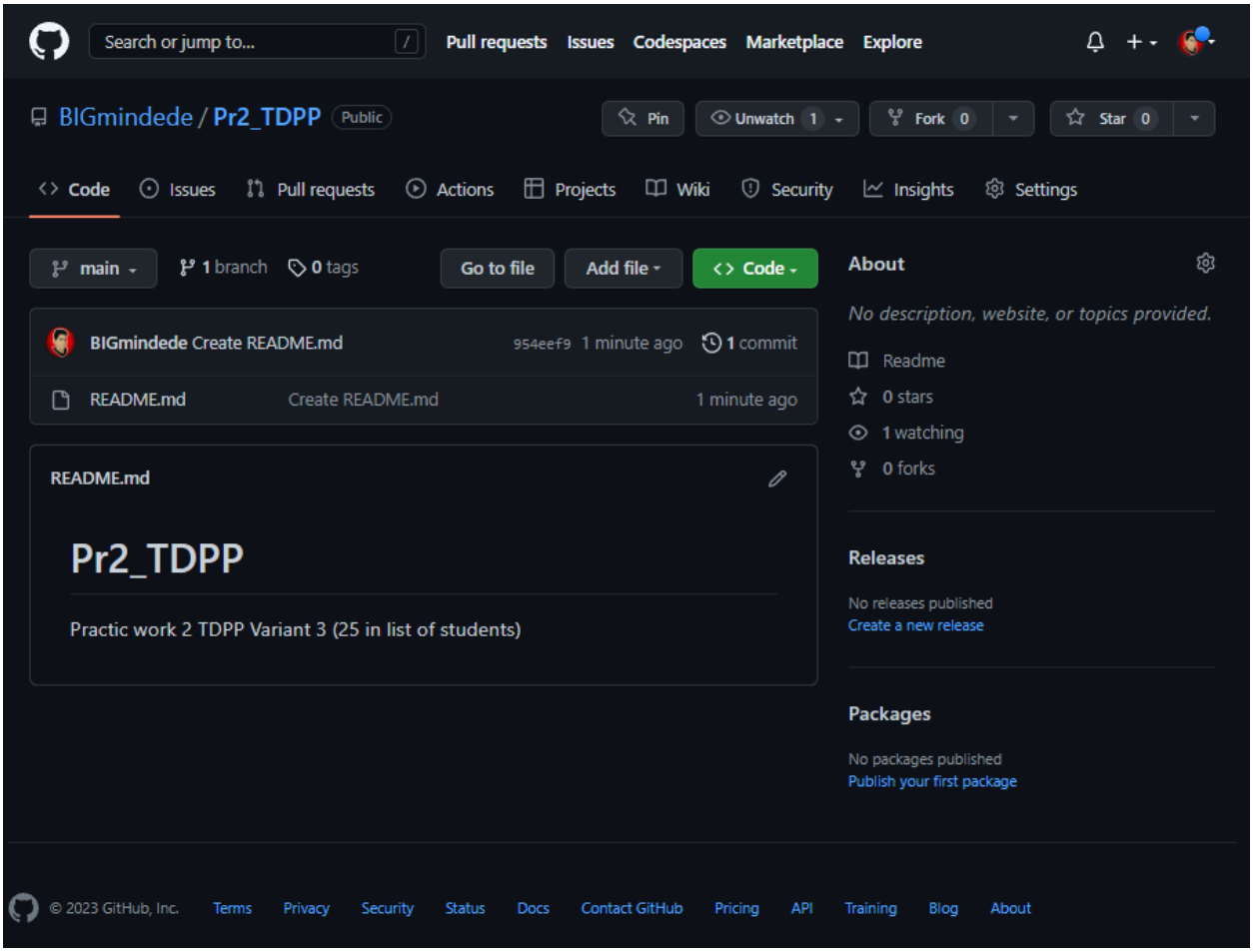


Рисунок 17 – Создание удалённого репозитория

3. Создайте новый локальный репозиторий с несколькими файлами на рабочей станции и загрузите его содержимое на GitHub

На рисунке 17 показан процесс загрузки существующего репозитория на GitHub.

```
hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2
$ git remote add origin https://github.com/BIGmindede/Pr2_TDPP
fatal: not a git repository (or any of the parent directories): .git

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2
$ git init
Initialized empty Git repository in C:/2year/ТРПП/Pr1/Pr1_2/.git/

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (master)
$ git touch file_1.txt
git: 'touch' is not a git command. See 'git --help'.

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (master)
$ touch file_1.txt

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (master)
$ touch file_2.txt

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (master)
$ git add file_1.txt

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (master)
$ git add file_2.txt

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (master)
$ git commit -m "Added empty .txt files #1 and #2"
[master (root-commit) 956c106] Added empty .txt files #1 and #2
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file_1.txt
 create mode 100644 file_2.txt

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (master)
$ git remote add origin https://github.com/BIGmindede/Pr2_TDPP

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (master)
$ git push -u origin main
error: src refspec main does not match any
error: failed to push some refs to 'https://github.com/BIGmindede/Pr2_TDPP'

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 244 bytes | 244.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/BIGmindede/Pr2_TDPP/pull/new/master
remote:
To https://github.com/BIGmindede/Pr2_TDPP
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (master)
$
```

Рисунок 17 – Загрузка репозитория на GitHub

4. Чтобы избежать ввода логина и пароля, создайте SSH-ключ для авторизации

На рисунке 18 показан процесс генерации SSH ключа.

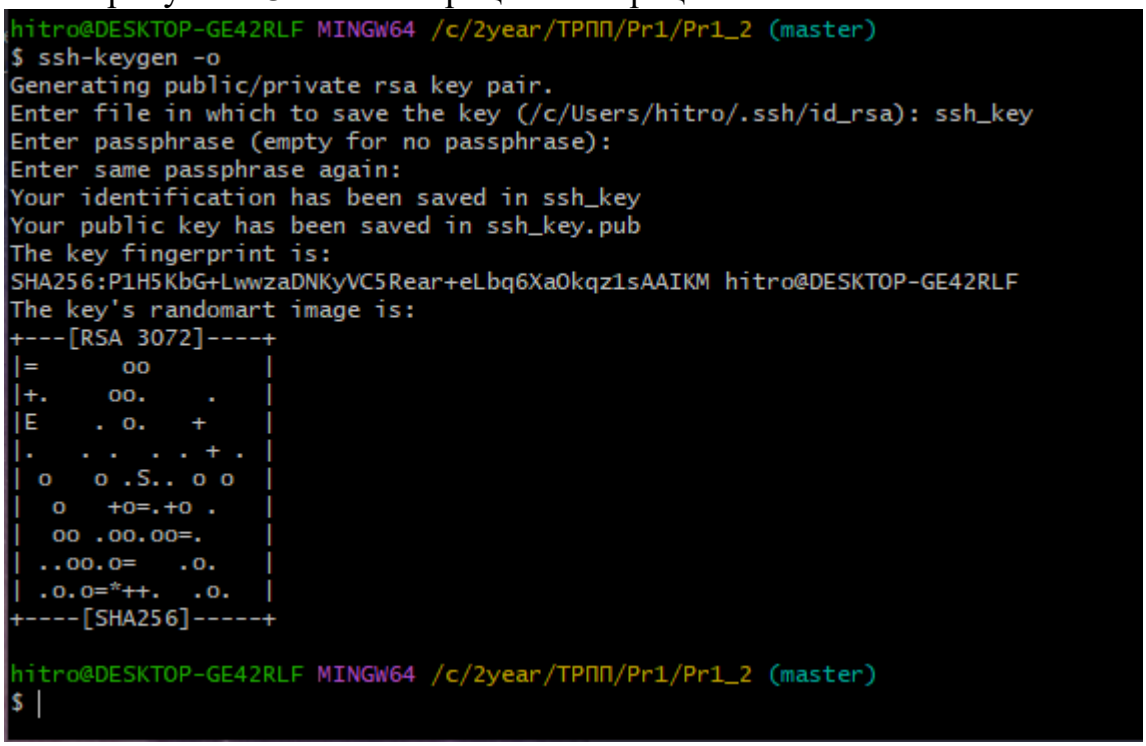


Рисунок 18 – Генерация SSH ключа

5. Создайте в репозитории новую ветку. Произведите в ней несколько изменений и слейте с веткой master

На рисунках 19-20 показан процесс создания новой ветки и слияние её с веткой master.

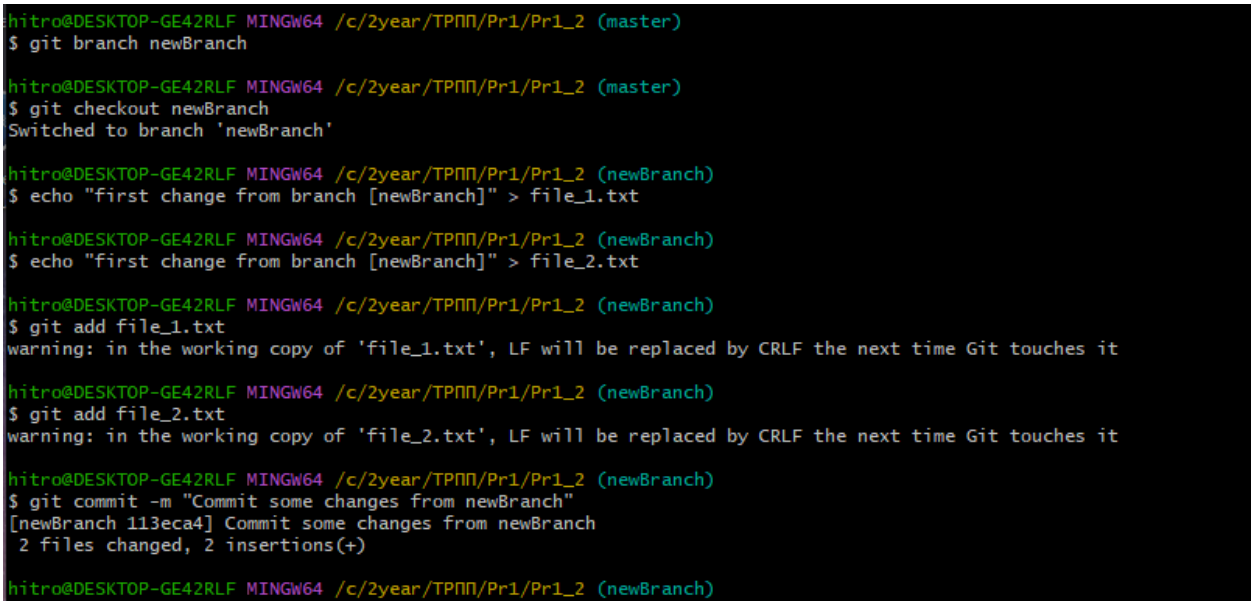


Рисунок 19 – Создание новой ветки и первый коммит в нее

Рисунок 19 – Слияние новой ветки с веткой master

6. Клонируйте непустой удаленный репозиторий на локальную машину

На рисунке 21 показан процесс клонирования удалённого репозитория.

```
hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (master)
$ git clone https://github.com/BIGmindede/Pr2_TDPP
Cloning into 'Pr2_TDPP'...
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 15 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (15/15), done.

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (master)
$
```

Рисунок 21 – Клонирование репозитория

7. Создайте тег указывающий на последний коммит в ветке

На рисунке 22 показан процесс создания тега, указывающего на последний коммит в ветке.

```
$ git log
commit 113eca40c6fe1ecec9fe0b4546c3d62ab2fa26f9 (HEAD -> master, newBranch)
Author: Nikita Khitrov <hitrov.qazws@gmail.com>
Date:   Fri Feb 17 23:47:33 2023 +0300

    Commit some changes from newBranch

commit 956c106124c5129c1bf13fe6745c426992b933ad (origin/master)
Author: Nikita Khitrov <hitrov.qazws@gmail.com>
Date:   Fri Feb 17 23:28:58 2023 +0300

    Added empty .txt files #1 and #2

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (master)
$ git tag v1

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (master)
$ git log
commit 113eca40c6fe1ecec9fe0b4546c3d62ab2fa26f9 (HEAD -> master, tag: v1, newBranch)
Author: Nikita Khitrov <hitrov.qazws@gmail.com>
Date:   Fri Feb 17 23:47:33 2023 +0300

    Commit some changes from newBranch

commit 956c106124c5129c1bf13fe6745c426992b933ad (origin/master)
Author: Nikita Khitrov <hitrov.qazws@gmail.com>
Date:   Fri Feb 17 23:28:58 2023 +0300

    Added empty .txt files #1 and #2

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (master)
$
```

Рисунок 22 – Создание тега, указывающего на последний коммит в ветке

8. Создайте новую ветку и выведите список всех веток

На рисунке 23 показан процесс создания новой ветки и вывода списка всех веток.

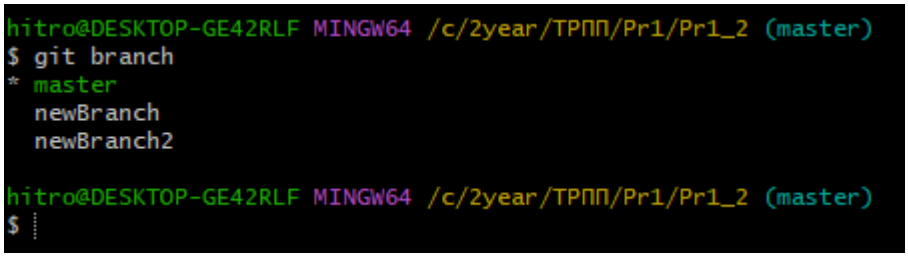


Рисунок 23 – Создание новой ветки и вывод списка всех веток

9. Произведите 3 коммита в новой ветке

На рисунках 24-26 показан процесс произведения трех коммитов в новой ветке.

```
hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (master)
$ git checkout newBranch2
Switched to branch 'newBranch2'

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (newBranch2)
$ touch file_3.txt

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (newBranch2)
$ echo "some code" > file_3.txt
```

```
$ git add file_3.txt
warning: in the working copy of 'file_3.txt', LF will be replaced by CRLF the next time Git touches it

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (newBranch2)
$ git commit -m "Added .txt file #3 with some code"
[newBranch2 a7e494c] Added .txt file #3 with some code
 1 file changed, 1 insertion(+)
 create mode 100644 file_3.txt

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (newBranch2)
$ echo "second change in code of first file" > file_1.txt
```

```
hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (newBranch2)
$ git add file_1.txt
warning: in the working copy of 'file_1.txt', LF will be replaced by CRLF the next time Git touches it

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (newBranch2)
$ git commit -m "Added chenges in .txt file #1"
[newBranch2 8c25608] Added chenges in .txt file #1
 1 file changed, 1 insertion(+), 1 deletion(-)

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (newBranch2)
$ echo "Second change in code to second file" > file_2.txt

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (newBranch2)
$ git add file_2.txt
warning: in the working copy of 'file_2.txt', LF will be replaced by CRLF the next time Git touches it

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (newBranch2)
$ git commit -m "Added changes in .txt file #2"
[newBranch2 a03ebf7] Added changes in .txt file #2
 1 file changed, 1 insertion(+), 1 deletion(-)

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_2 (newBranch2)
$
```

Рисунки 24-26 – Произведение трех коммитов в новой ветке

10. Выгрузите все изменения в удаленный репозиторий

На рисунке 27 показана выгрузка изменений в удаленный репозиторий.

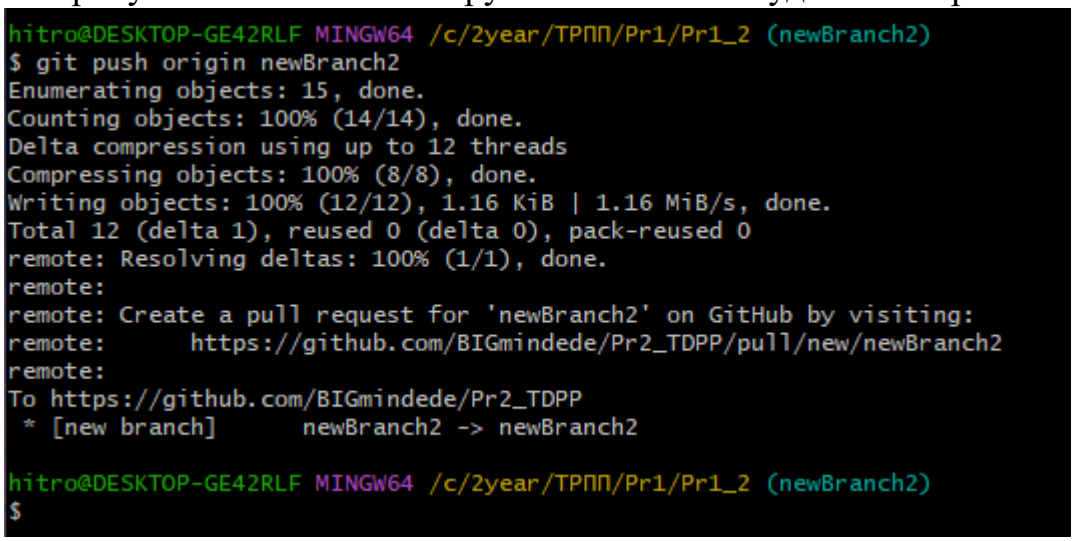


Рисунок 27 – Выгрузка изменений в удаленный репозиторий

11. Откатите ветку к созданному тегу (в том числе в удаленном репозитории)

На рисунке 28 показан процесс отката ветки к созданному тегу, втом числе в удаленном репозитории.

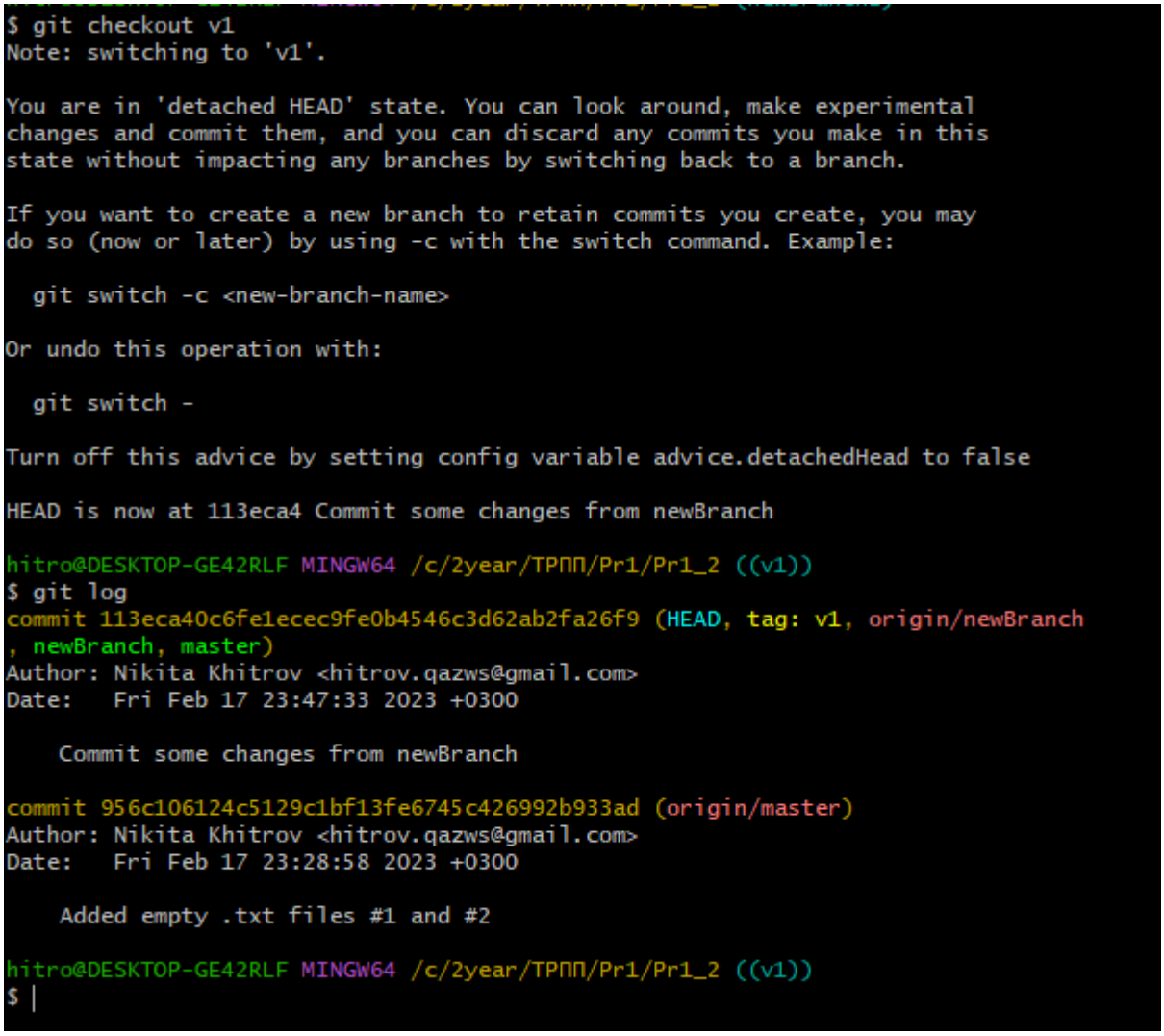


Рисунок 26 – откат ветки к созданному тэгу

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

На рисунке 29 показаны различия между веткой newBranch и master.

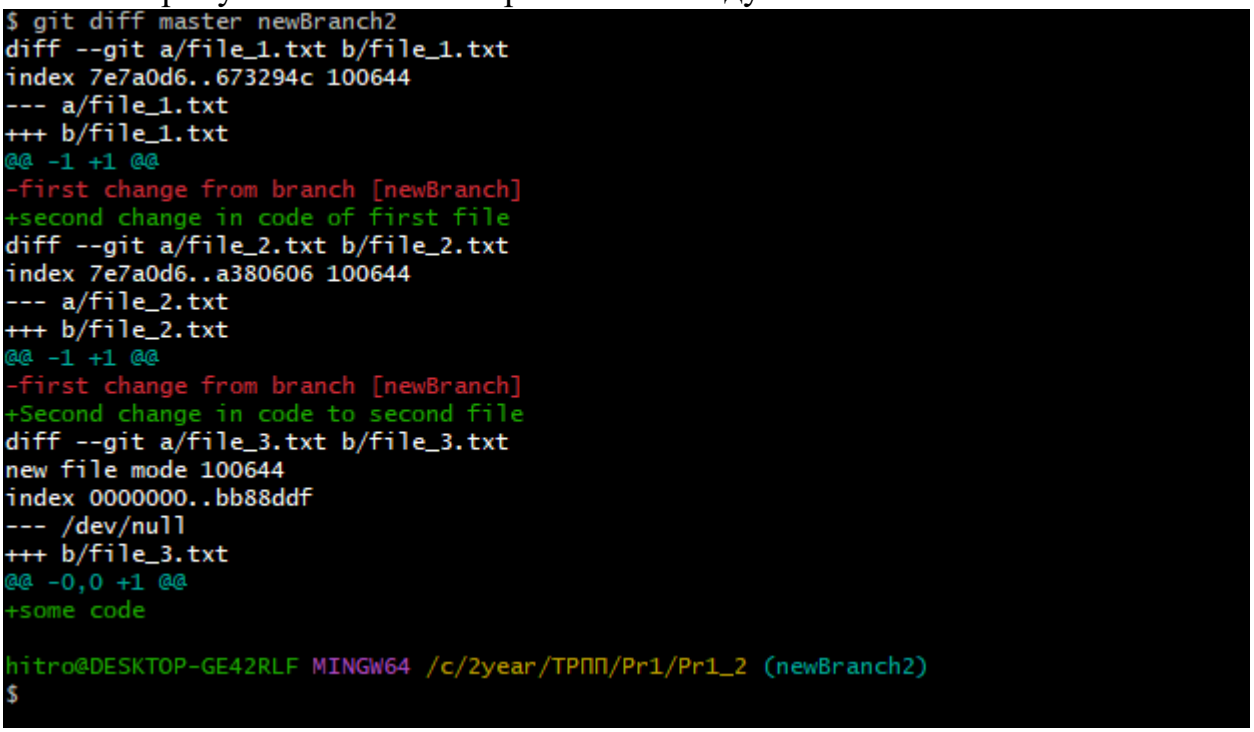


Рисунок 29 – Сравнение веток branch2 и master

Часть 3. Работа с ветвлением и оформление кода

Вариант 5

1. Сделайте форк репозитория в соответствии с Вашим вариантом

На рисунке 30 показан процесс создания форка репозитория.

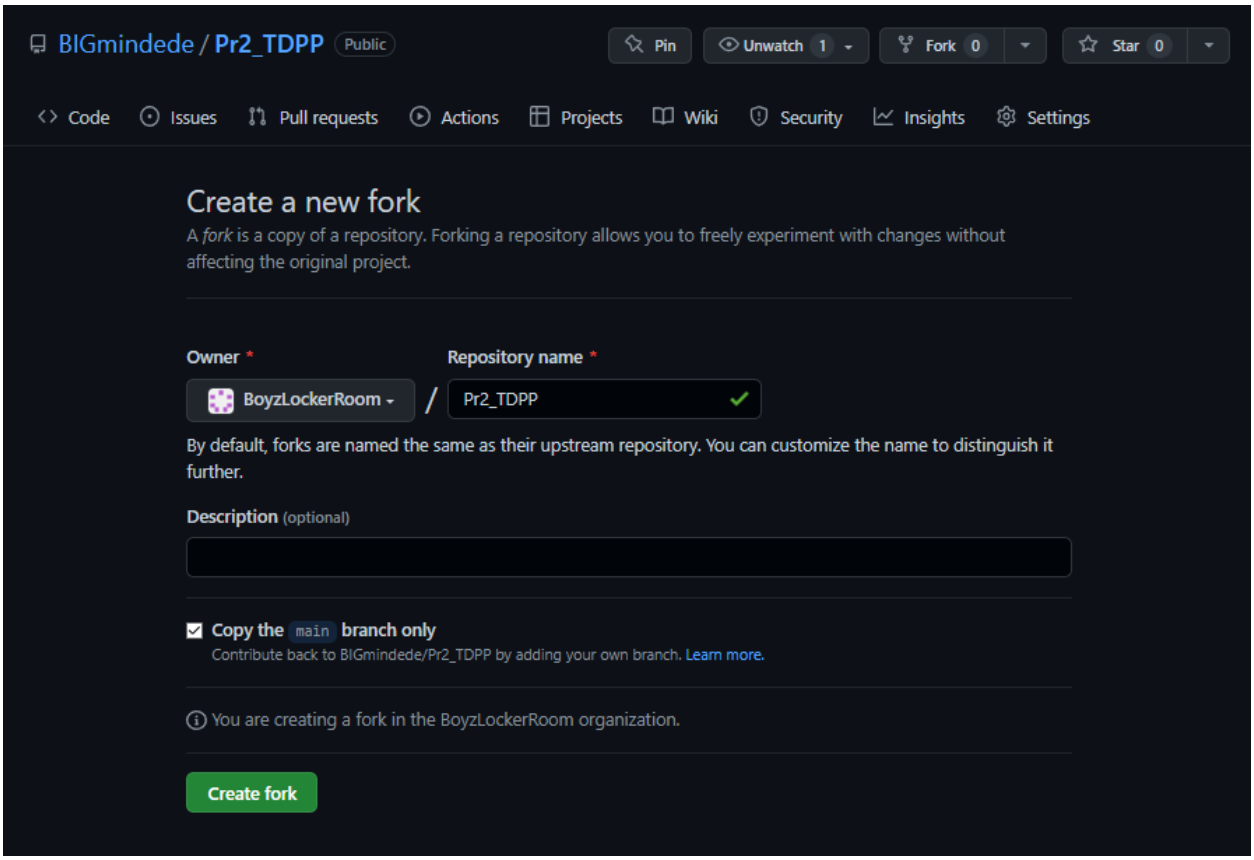


Рисунок 30 – Форк репозитория

2. Склонируйте его (форк репозитория) на локальную машину.

На рисунке 31 показан процесс клонирования репозитория на локальную машину.

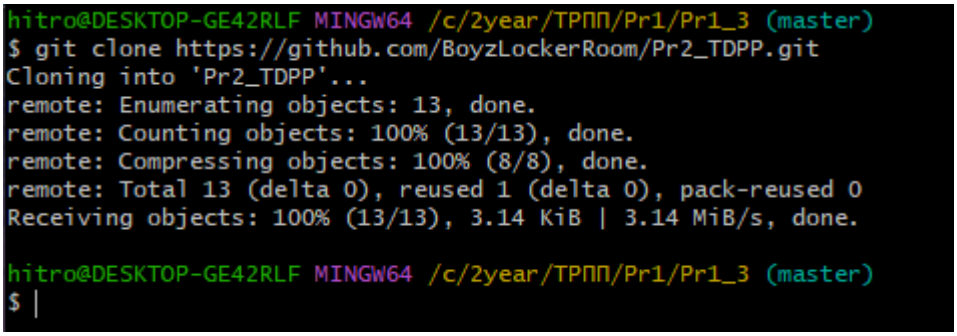


Рисунок 31 – Клонирование репозитория на локальную машину

3. Создайте две ветки branch1 и branch2 от последнего коммита в master'e

На рисунке 32 показан процесс создания двух веток branch1 и branch2 от последнего коммита в master.

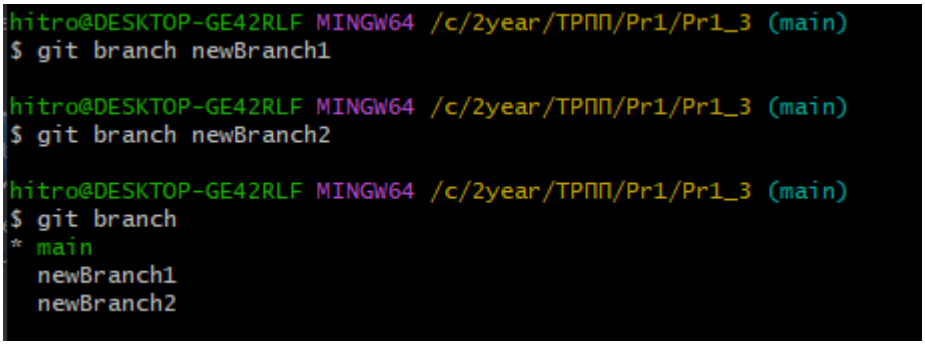
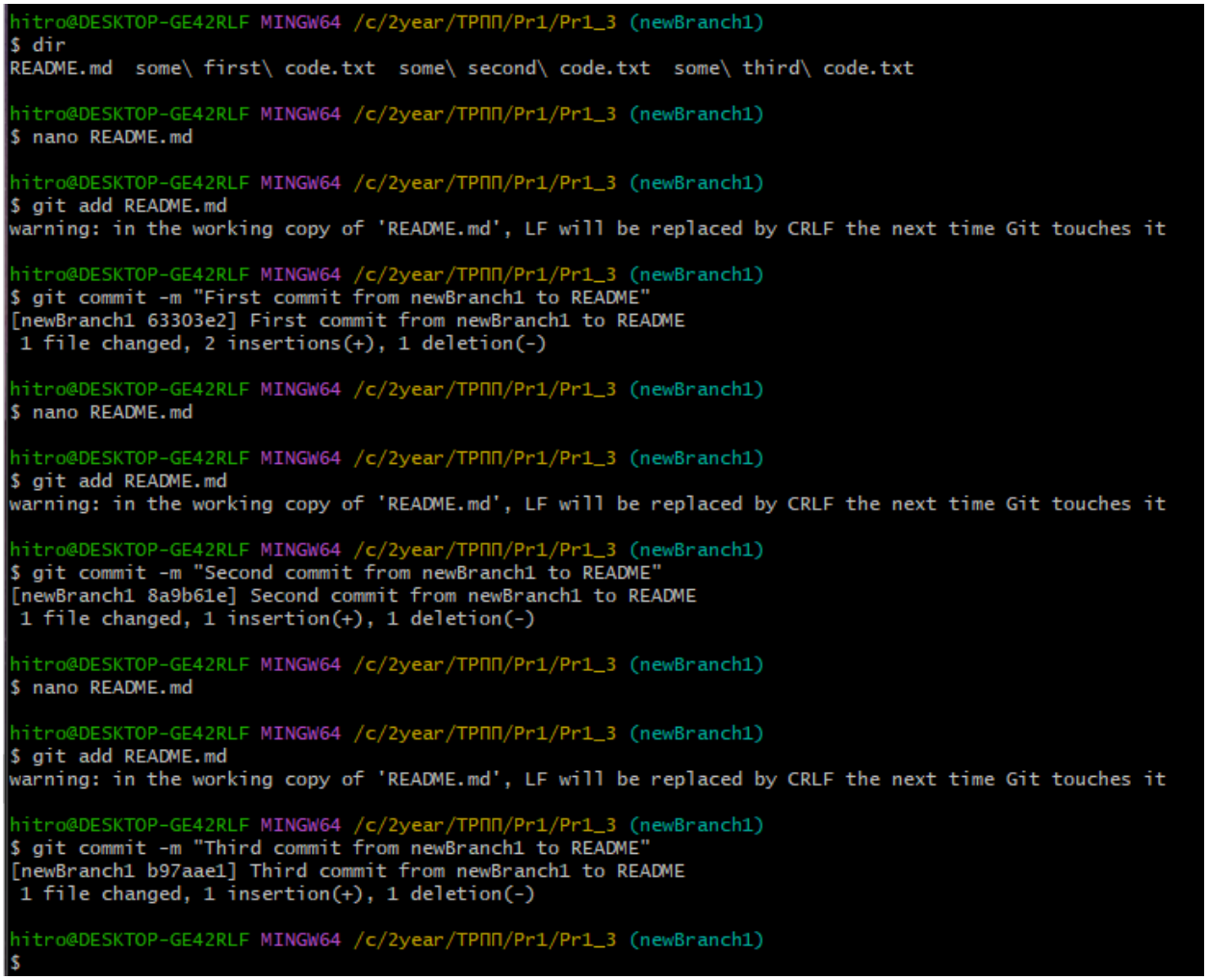


Рисунок 32 – Создание двух веток от последнего коммита

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

На рисунках 33 - 34 показаны 3 коммита на каждой из веток.



```
hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_3 (newBranch1)
$ git checkout newBranch2
Switched to branch 'newBranch2'
D       somde third code.txt

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_3 (newBranch2)
$ nano README.txt

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_3 (newBranch2)
$ nano README.md

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_3 (newBranch2)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_3 (newBranch2)
$ git commit -m "First commit from newBranch2 to README"
[newBranch2 40ed373] First commit from newBranch2 to README
 1 file changed, 2 insertions(+), 1 deletion(-)

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_3 (newBranch2)
$ nano README.md

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_3 (newBranch2)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_3 (newBranch2)
$ git commit -m "Second commit from newBranch2 to README"
[newBranch2 e807aa5] Second commit from newBranch2 to README
 1 file changed, 1 insertion(+), 1 deletion(-)

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_3 (newBranch2)
$ nano README.md

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_3 (newBranch2)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_3 (newBranch2)
$ git commit -m "Third commit from newBranch2 to README"
[newBranch2 24d9def] Third commit from newBranch2 to README
 1 file changed, 1 insertion(+), 1 deletion(-)

hitro@DESKTOP-GE42RLF MINGW64 /c/2year/ТРПП/Pr1/Pr1_3 (newBranch2)
$
```

Рисунки 33-34 – Создание 3-ёх коммитов на ветках

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

На рисунке 35 показан процесс слияния ветки branch и branch2 и разрешения при этом конфликтов.

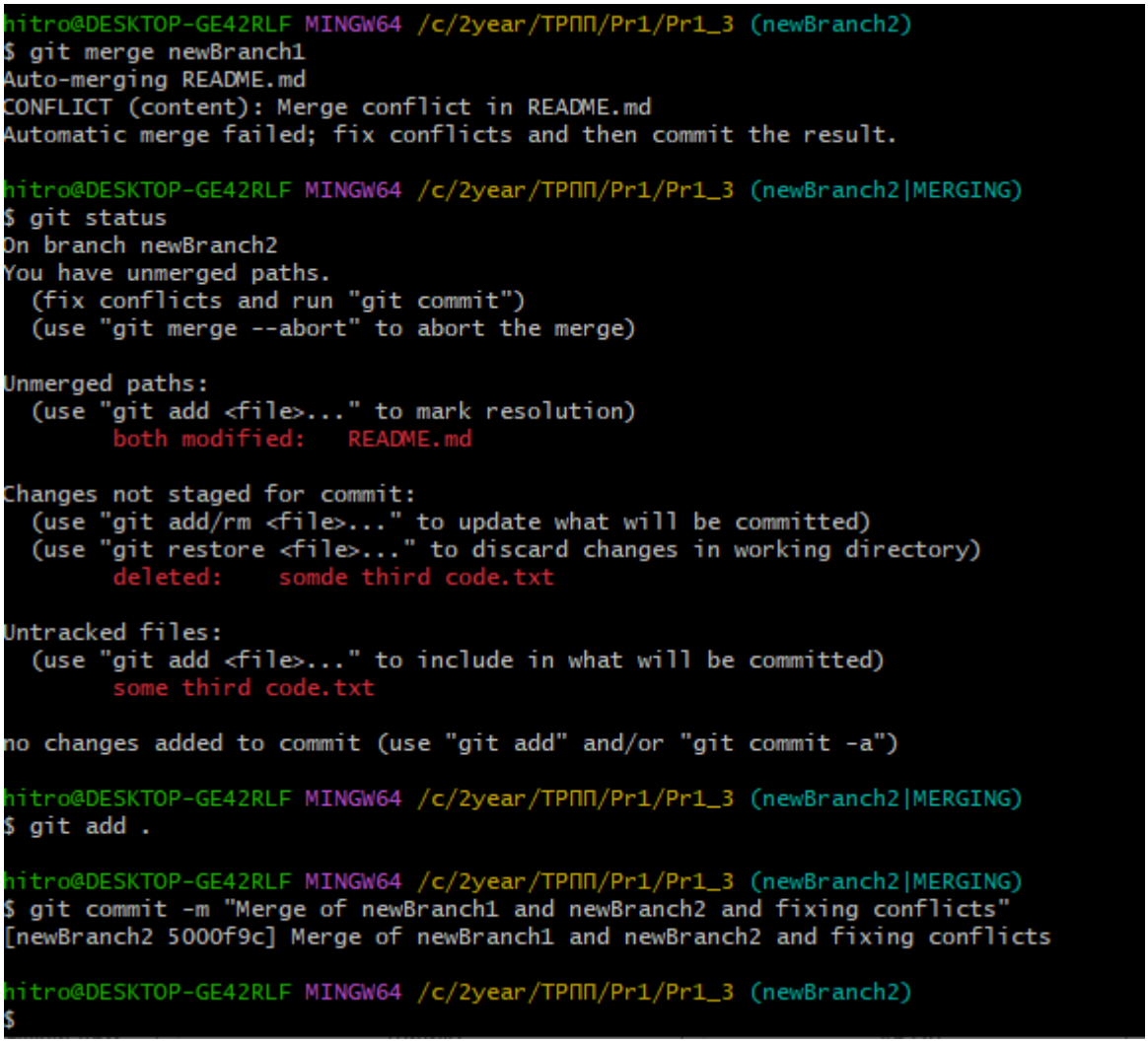


Рисунок 35 – Слияние двух веток

6. Выгрузите все изменения во всех ветках в удаленный репозиторий.

На рисунке 36 показан процесс выгрузки всех изменений в удалённый репозиторий.

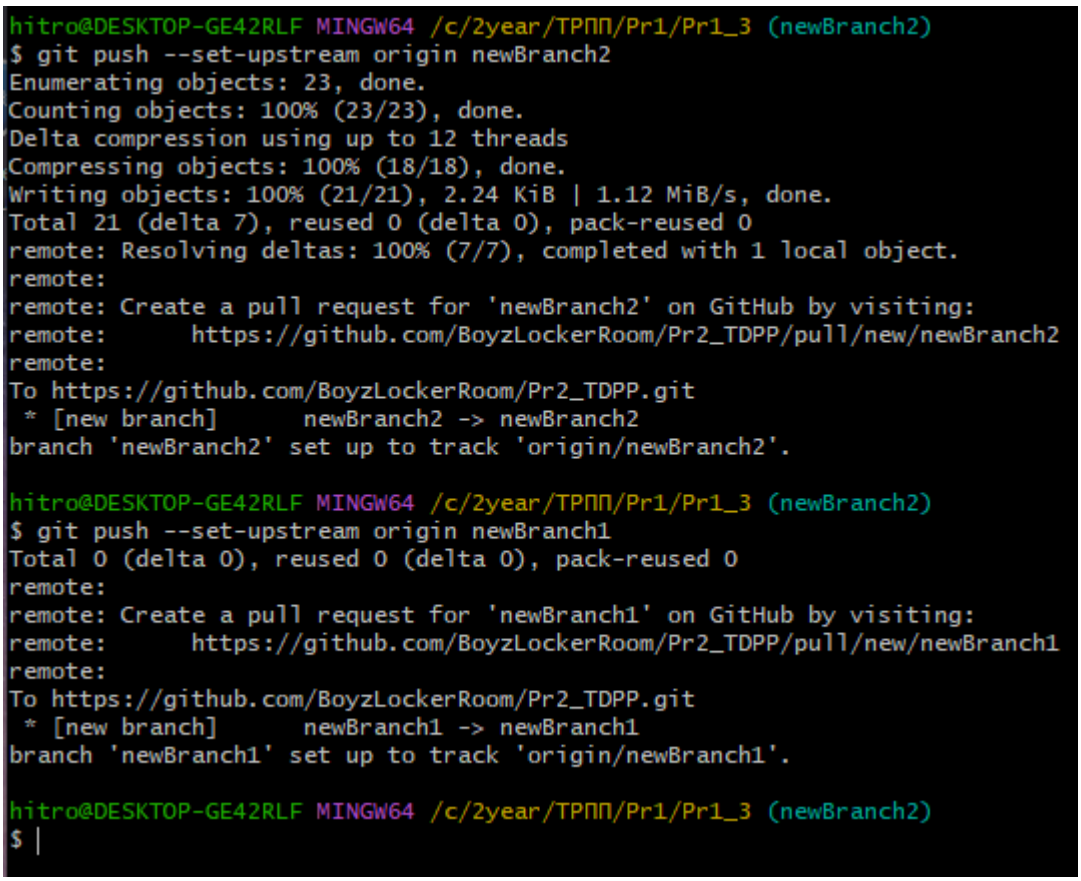


Рисунок 36 – Выгрузка изменений в удалённый репозиторий

7. Проведите еще 3 коммита в ветку branch1

На рисунке 37 показан процесс создания 3-ёх коммитов на ветке branch1.

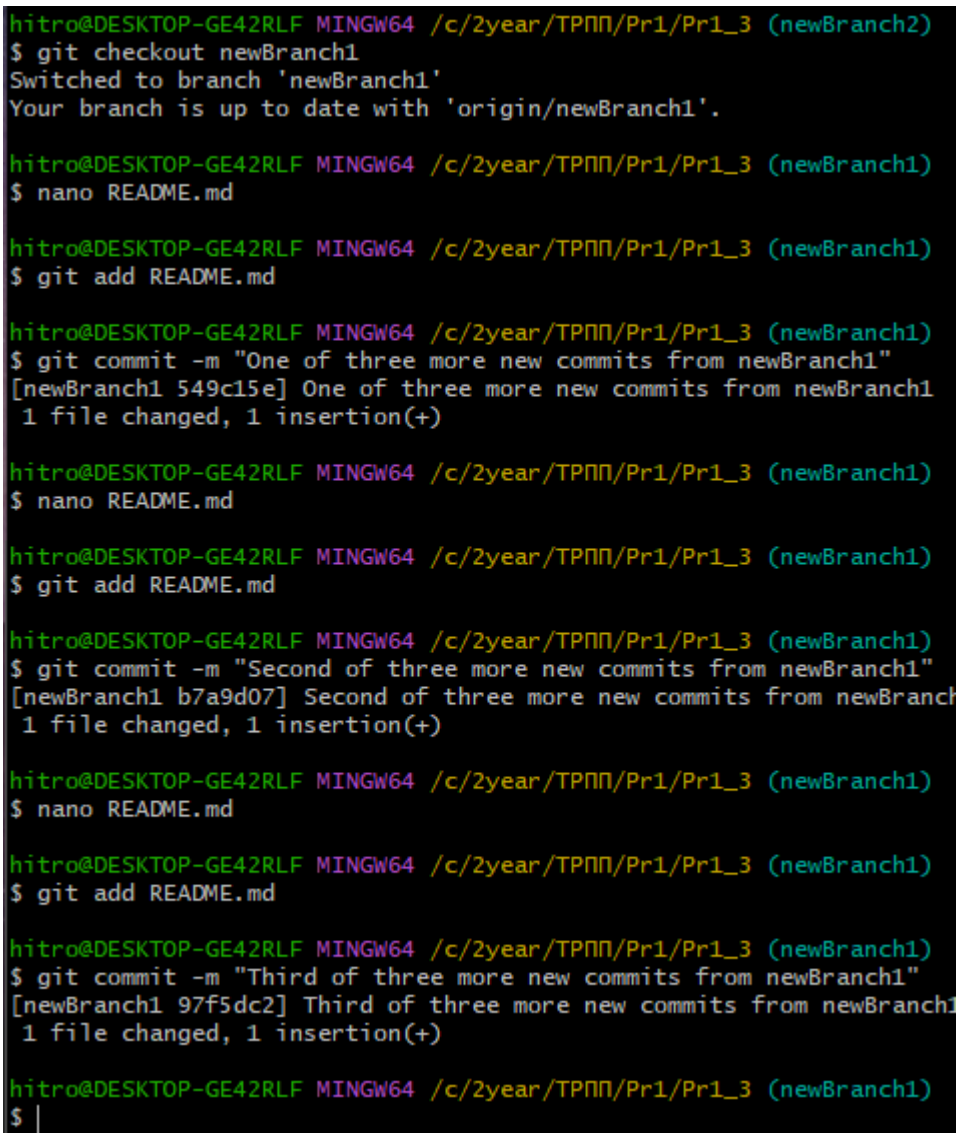


Рисунок 37 – 3 коммита на ветке branch1

8. Склонируйте репозиторий еще раз в другую директорию

На рисунке 38 показан процесс клонирования репозитория в другую директорию.

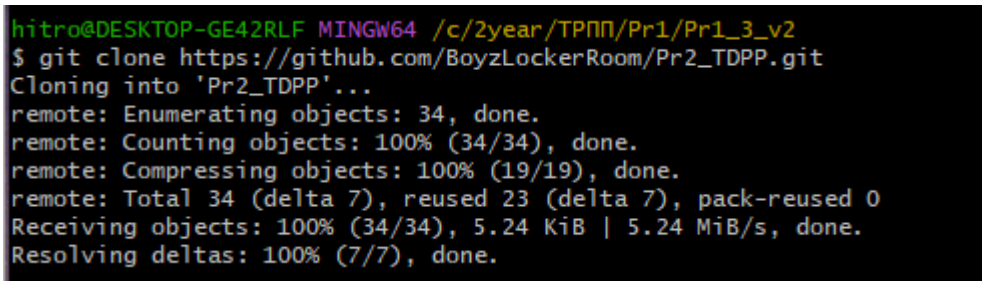


Рисунок 38 – Клонирование репозитория

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

На рисунке 39 показан процесс создания 3-ёх коммитов на ветке branch1 в новом репозитории.

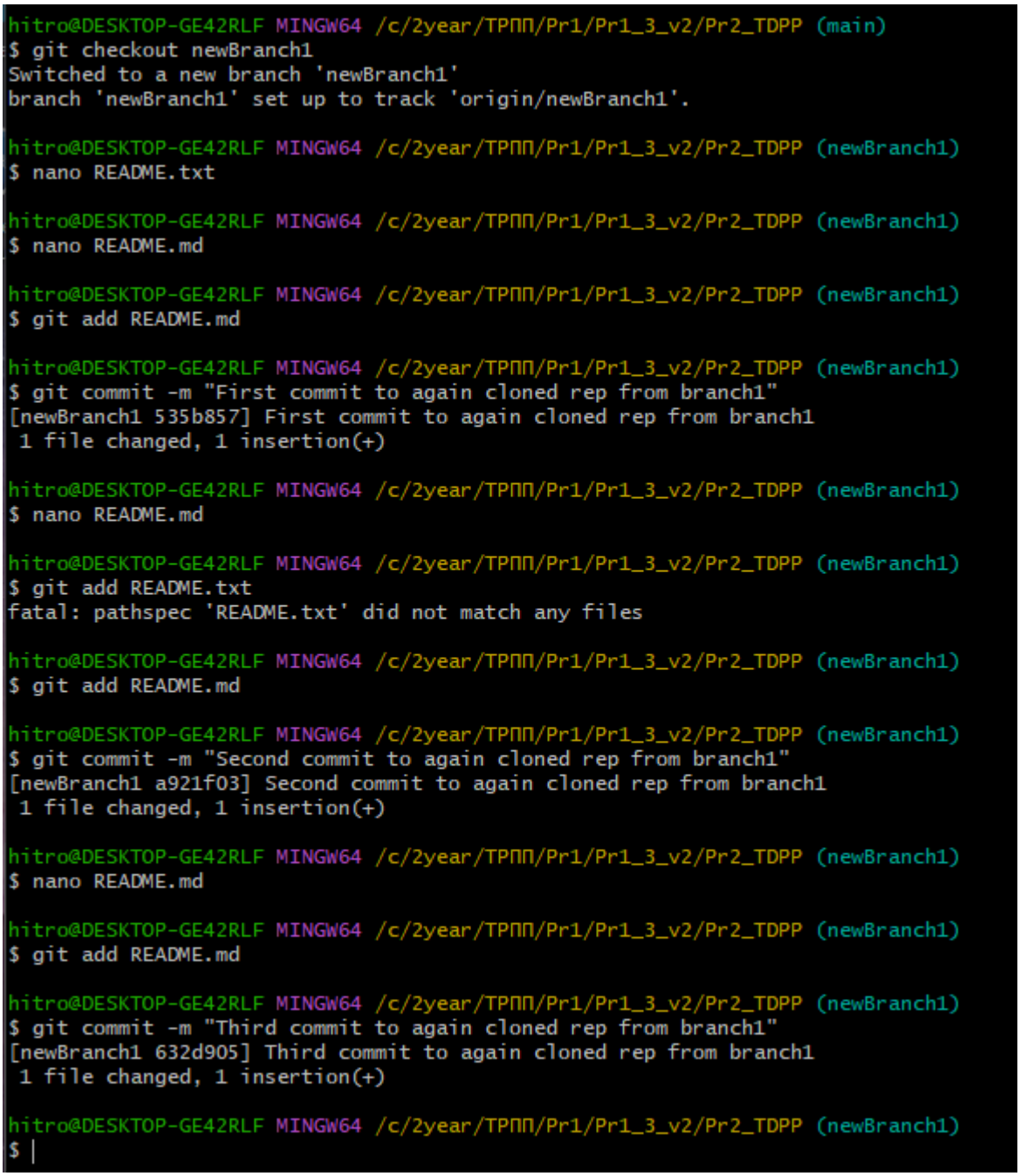


Рисунок 39 – Создание коммитов в новом репозитории

10. Выгрузите все изменения из нового репозитория в удаленный репозиторий

На рисунке 40 показан процесс выгрузки изменений в удалённые репозиторий.

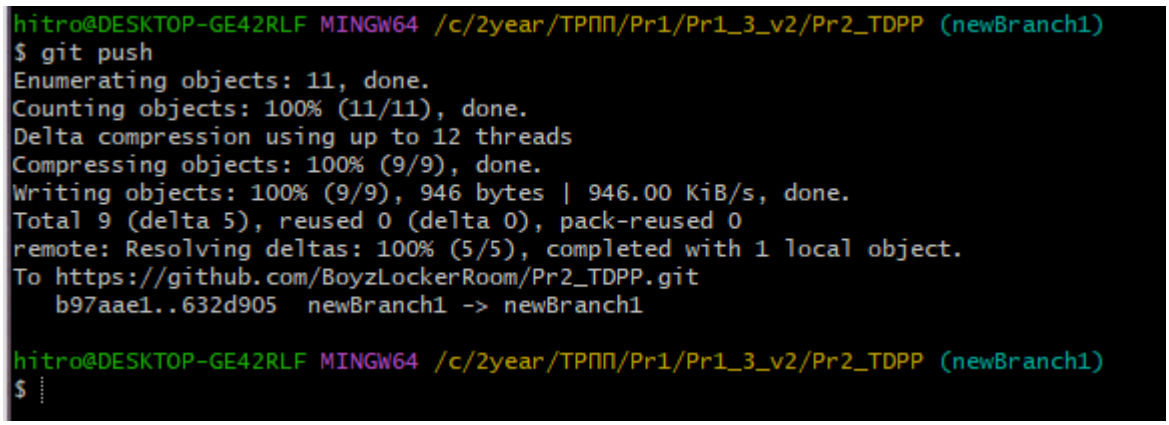


Рисунок 40 – Выгрузка изменений в удалённый репозиторий

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

На рисунке 41 показан процесс выгрузки изменений с тегом –force.

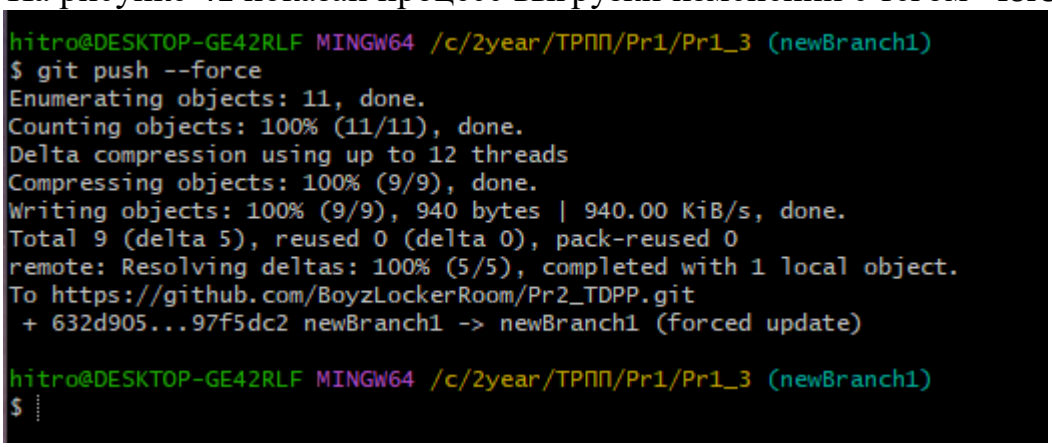


Рисунок 41 – Выгрузка в удалённый репозиторий с тегом –force

12. Получите все изменения в новом репозитории

На рисунке 42 показан процесс получения изменений из удалённого репозитория.

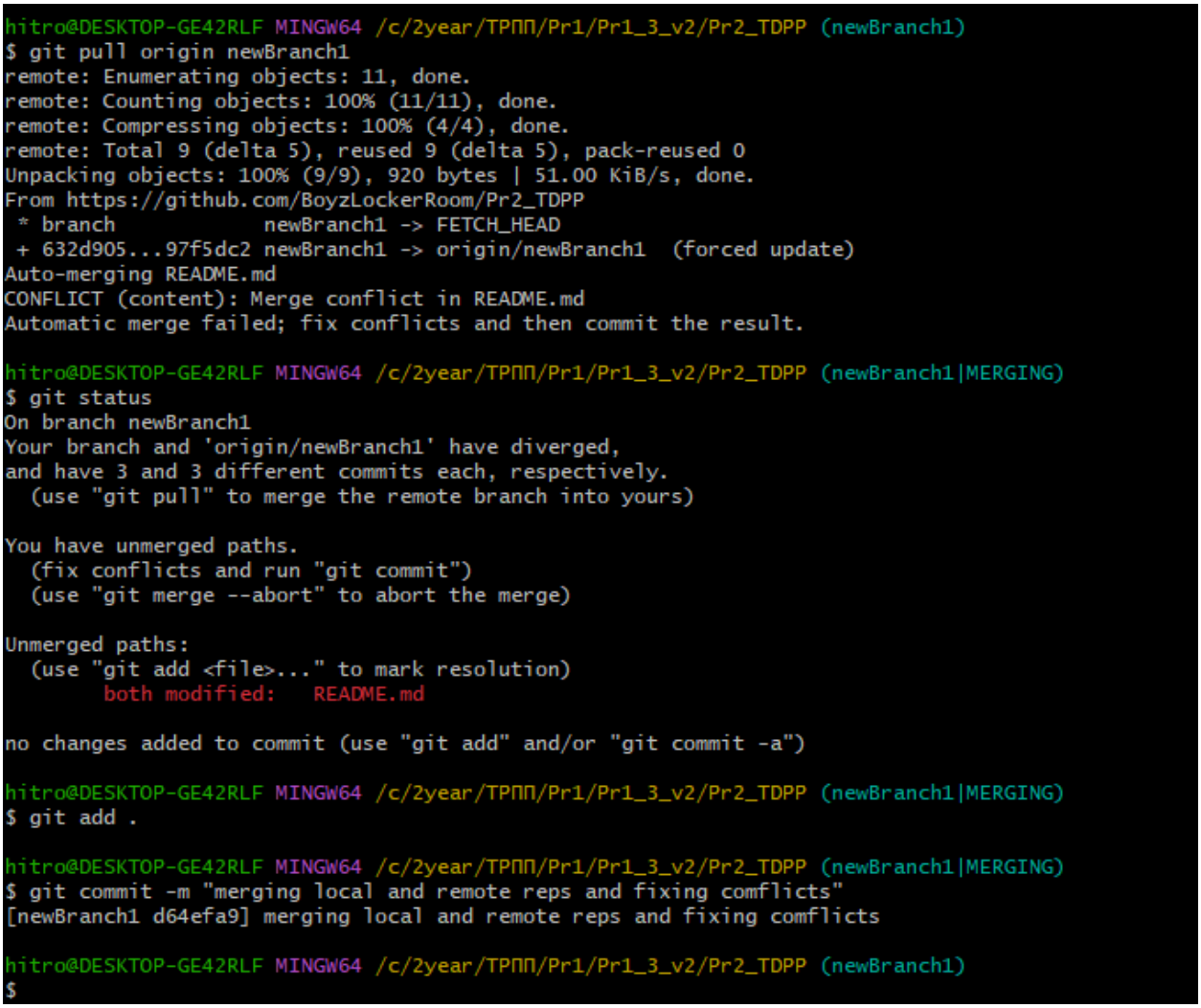


Рисунок 42 – Получение изменений из удалённого репозитория

Ответы на вопросы

2. К какому типу систем контроля версий относится Git?

Git относится к распределённой системе контроле версий.

3. Что такое репозиторий Git?

Репозиторий – место, где система контроля версий хранит свои метаданные и базу данных объектов проекта.

Локальный репозиторий — репозиторий, расположенный на локальном компьютере разработчика в каталоге. Именно в нём происходит разработка и фиксация изменений, которые отправляются на удалённый репозиторий.

Удалённый репозиторий — репозиторий, находящийся на удалённом сервере. Это общий репозиторий, в который приходят все изменения и из которого забираются все обновления.

4. Что такое коммит?

Коммиты – основные конструктивные элементы временной шкалы Git. Их можно рассматривать как снимки состояния или контрольные точки на временной шкале проекта Git. Коммиты создаются с помощью команды git commit, которая делает снимок состояния проекта на текущий момент времени.

5. Что такое ветка в репозитории Git?

Ветка (Branch) — это параллельная версия репозитория. Она включена в этот репозиторий, но не влияет на главную версию, тем самым позволяя свободно работать в параллельной. Когда вы внесли нужные изменения, то вы можете объединить их с главной версией.

11. Что делает команда git status?

Команда git status отображает состояние рабочего каталога и раздела проиндексированных файлов. С её помощью можно проверить индексацию изменений и увидеть файлы, которые не отслеживаются Git.

12. Что делает команда git add?

Команда git add добавляет изменение из рабочего каталога в раздел проиндексированных файлов. Она сообщает Git, что вы хотите включить изменения в конкретном файле в следующий коммит.

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

Команда git branch без параметров выведет список всех веток локального репозитория. Также она выделит зелёным цветом ветку, на которой находится указатель HEAD.

27. Как сделать ветку с названием my_branch?

Для создания ветки с именем my_branch необходимо написать команду git branch my_branch в консоль, либо можно написать команду git checkout -b my_branch, которая создаст новую ветку и переключит указатель HEAD на неё.

Выводы

В ходе данной работы был изучен функционал и возможности системы контроля версий git.