МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ БІЗНЕС-КОЛЕДЖ

***Циклова комісія програмування***

**Робочий план**

з курсу «Тренінг-курс з основ програмування»

Ткаченко Іллі

*ПІБ студента*

студента групи **2П-21**

варіант \_\_\_\_\_

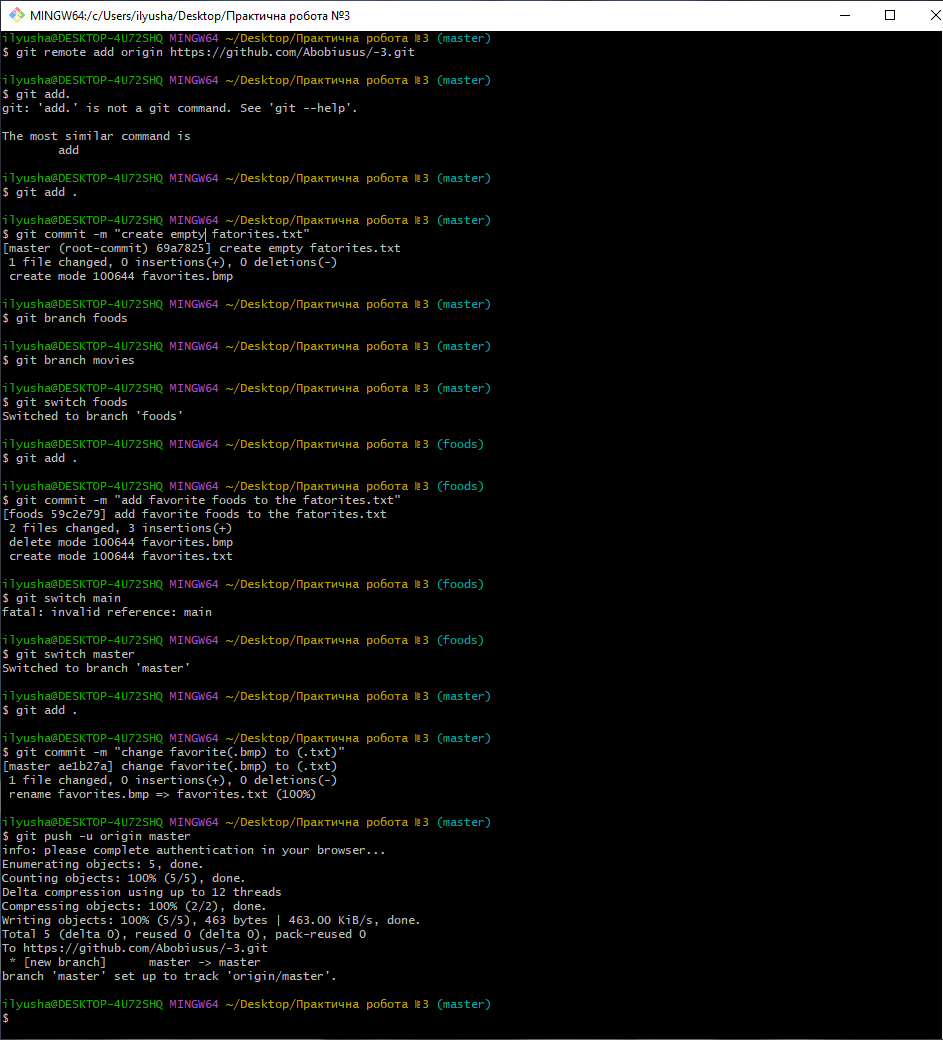
|  |  |  |
| --- | --- | --- |
| **Вид роботи** | **Дата** | **Оцінка** |
| *Пром. контроль* |  |  |
| *Залік* |  |  |

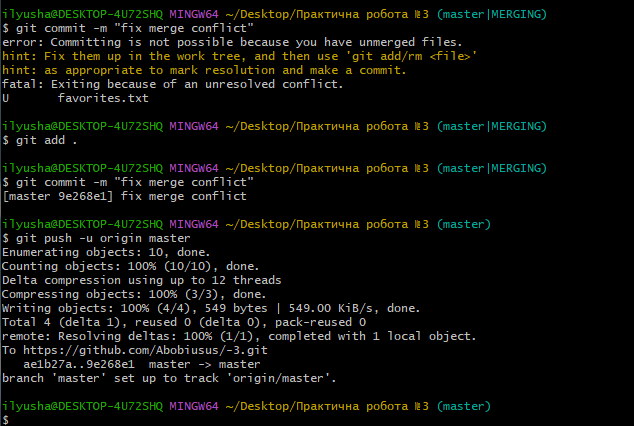
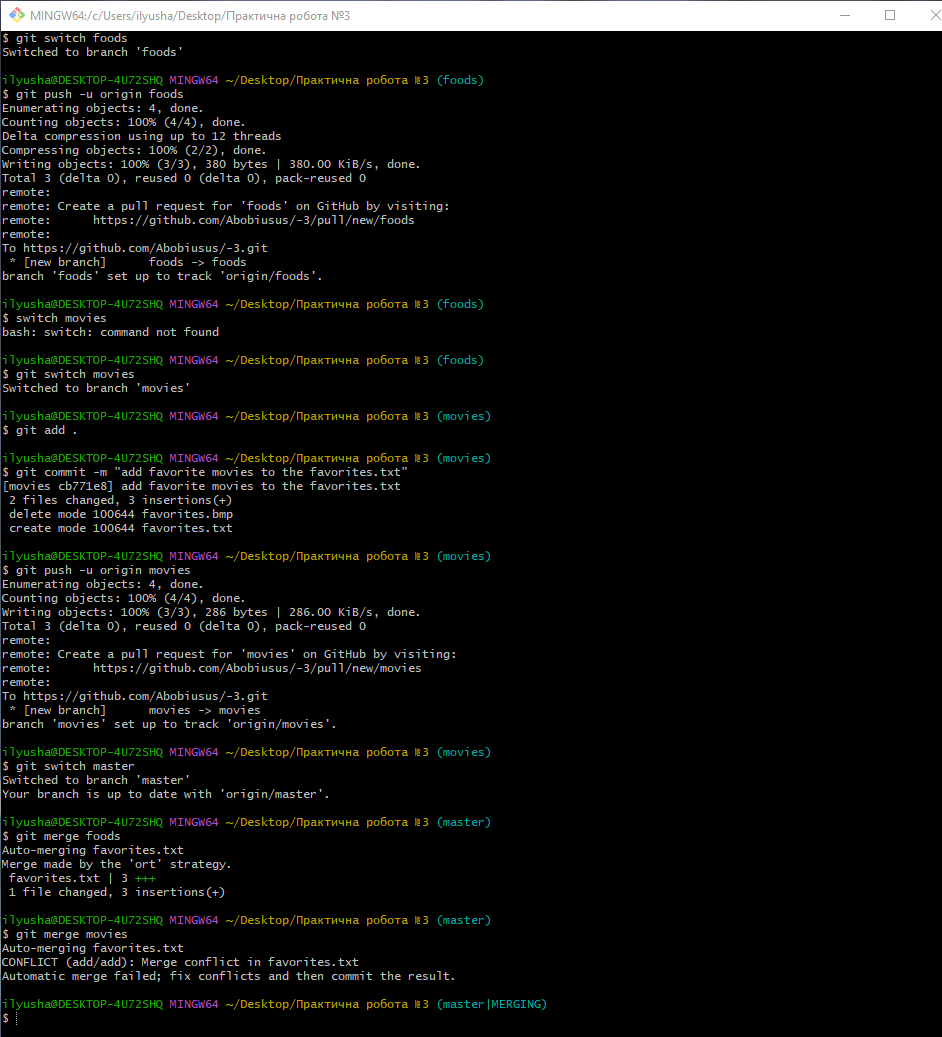
Викладач Марченко С. В.

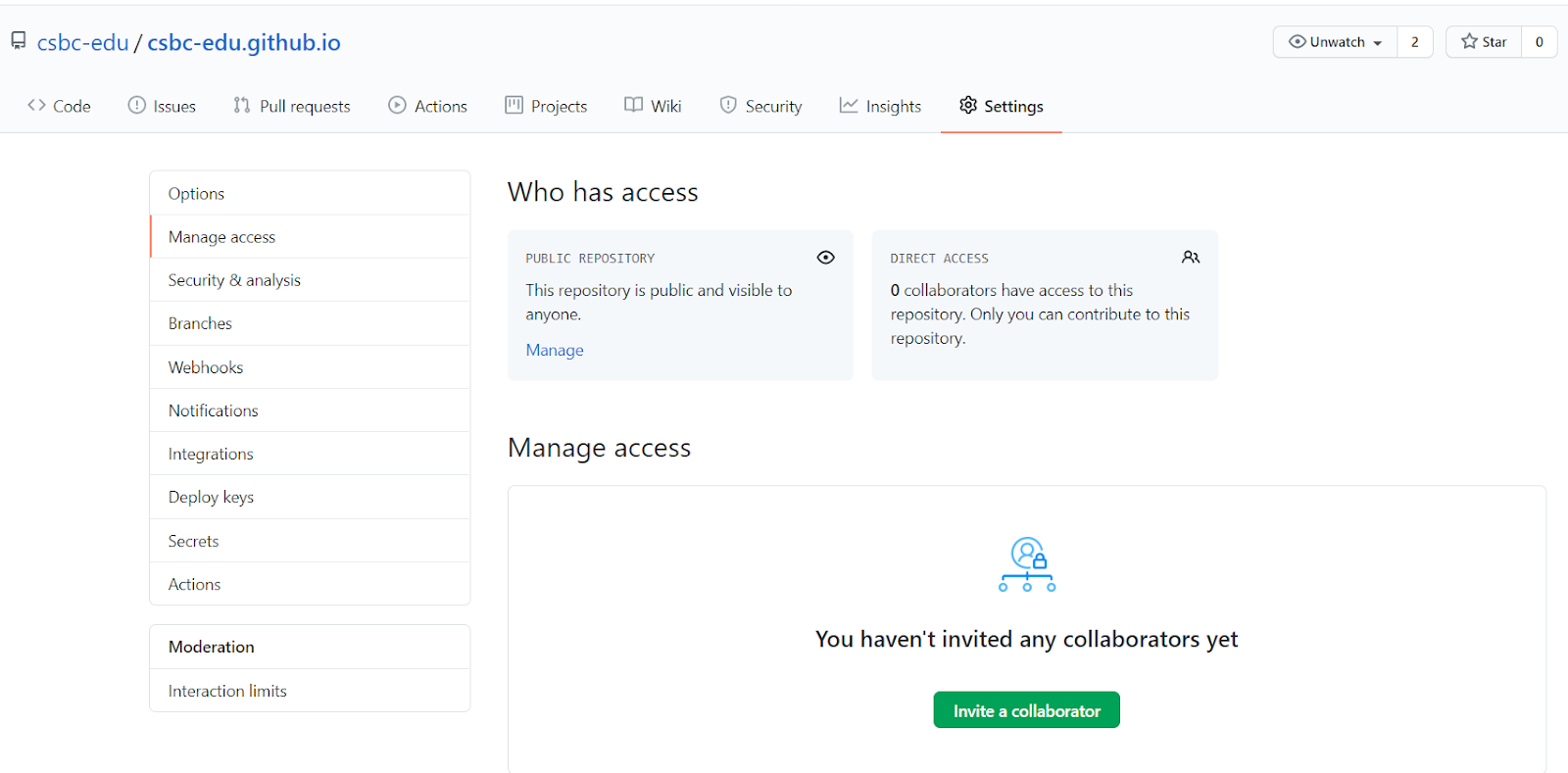
Черкаси-2022

**Практична робота №3**

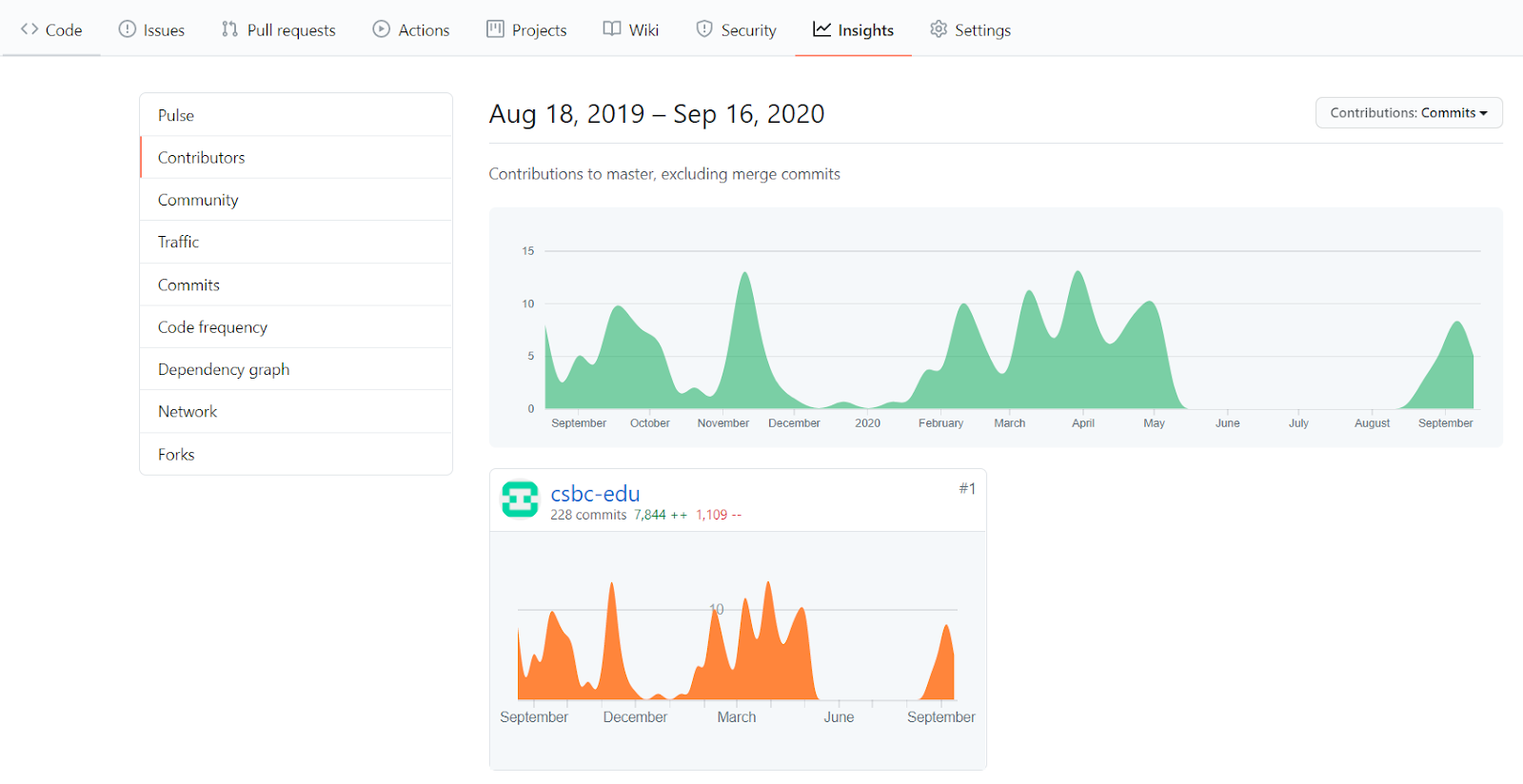
1. *Основи роботи з GitHub. Виконайте в консолі наступні операції:*
2. Створіть новий репозиторій локально на своїй машині.
3. Створіть новий GitHub-репозиторій. Назвати його можете довільним чином.
4. Приєднайте Ваш локальний репозиторій до Github-репозиторію.
   * Необов’язково: перейменуйте типову гілку master на main.
5. Створіть новий файл favorites.txt та залиште його порожнім. Зафіксуйте перший коміт на main-гілці.
6. Запуште Вашу main-гілку на GitHub! Переконайтесь, що отримали порожній файл favorites.txt на GitHub.
7. Далі створіть 2 гілки: foods та movies.
8. Перейдіть на гілку foods. Додайте 3 (або більше) улюблених страви до файлу favorites.txt. Зафіксуйте зміни в гілці foods.
9. Перейдіть на гілку movies та додайте 3 або більше улюблених фільми до файлу favorites.txt file. Зафіксуйте зміни в гілці movies.
10. Запуште гілку foods на GitHub. Переконайтесь, що її стане видно!
11. Запуште гілку movies на GitHub. Переконайтесь, що її стане видно!
12. Виконайте злиття гілки foods з main. Потім злийте гілку movies у main. За потреби вирішіть конфлікт так, щоб улюблені страви та фільми знаходились в одному файлі – favorites.txt.
13. Запуште останні операції в гілку main на Github.



1. 
2. Спробуйте налагодити роботу зі спільним репозиторієм. Об’єднайтесь навколо одного репозиторію командами по 2-3 особи та організуйте доступ команди до нього. На прикладі GitHub-репозиторія розгляньте процес додавання співавторів для управління репозиторієм:



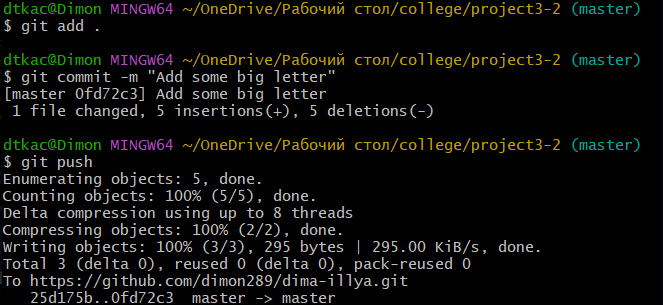
Запросіть команду до спільного внесення змін у наповнення репозиторія. Динаміку внесених змін до проекту можна побачити на вкладці Insights, пункт Collaborators



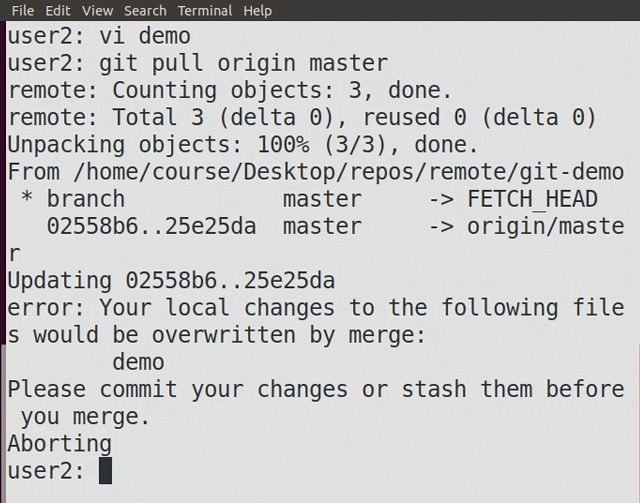
Змоделюйте наступні конфлікти при роботі зі спільним репозиторієм.

***2.1. Конфлікт: Pulling with Untracked Changes***. Відбувається управління одним файлом для двох користувачів.

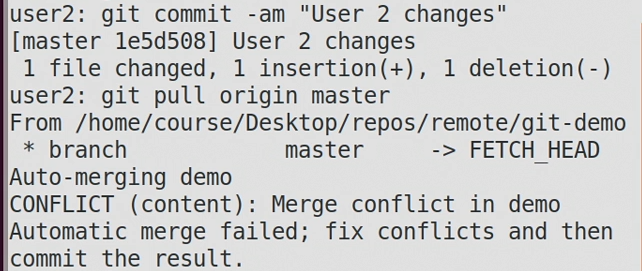
Користувач 1: здійснив зміни в файлі та виконав коміт у вітку master.

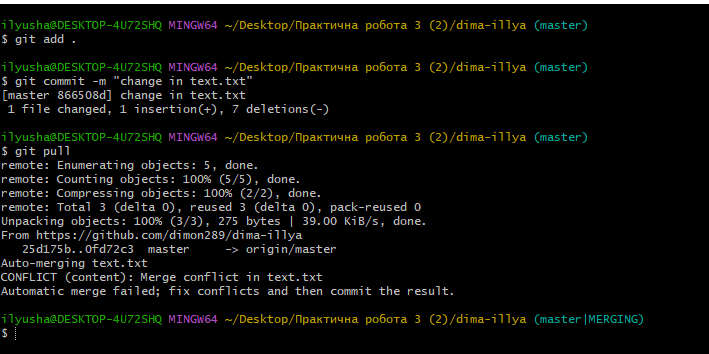


Користувач 2: спочатку вніс та локально зберіг свої зміни у файл (версія без змін користувача 1), а потім вирішив здійснити пулінг з центрального репозиторію.

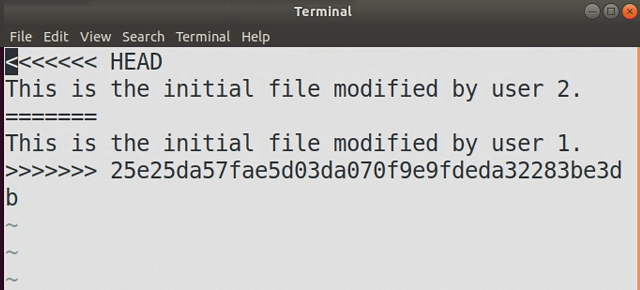


Нові користувачі вважають, що можуть здійснювати пулінг у будь-який момент. Для цього потрібно мати *або* чисту робочу папку (команда git reset --hard), *або* зафіксувати (коміт) свої зміни та виконати пулінг

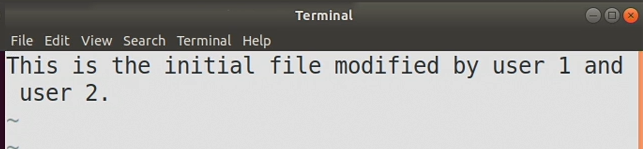


З’явиться конфлікт злиття у результаті проблем додавання локальних змін двома користувачами в один файл. 

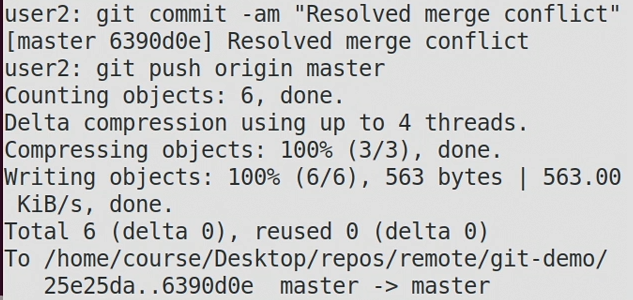
**Вирішення конфлікту**. Відкрити проблемний файл. Git додав метадані з описом конфлікту в цей файл

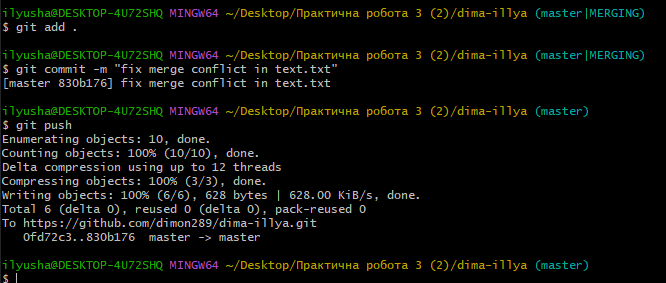


Виправте файл таким чином, щоб урахувати зміни обох користувачів

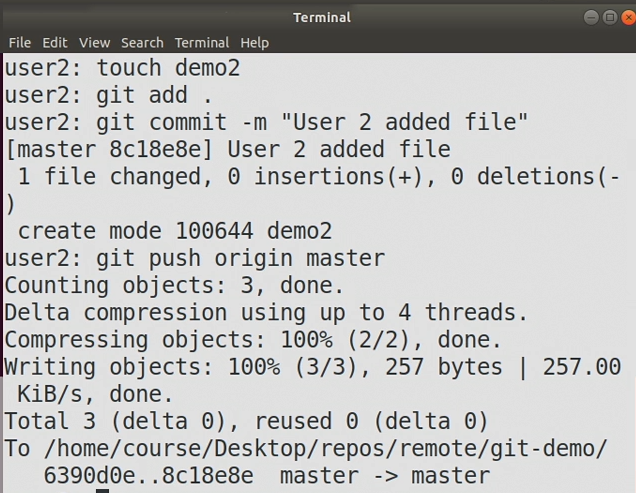


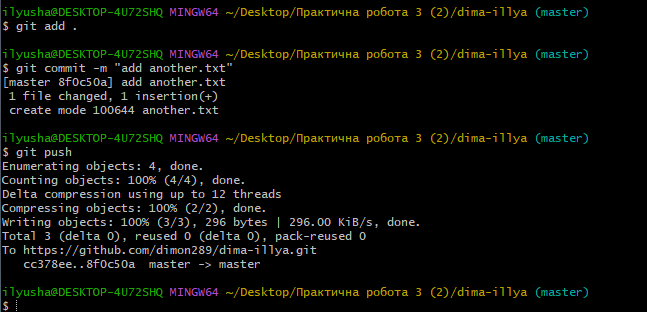
Далі ці зміни фіксуються та пушаться.



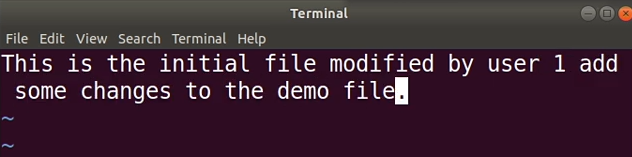


***2.2. Конфлікт: Forced Pushes.*** Користувач 2 створює новий, фіксує та пушить файл у центральний репозиторій:

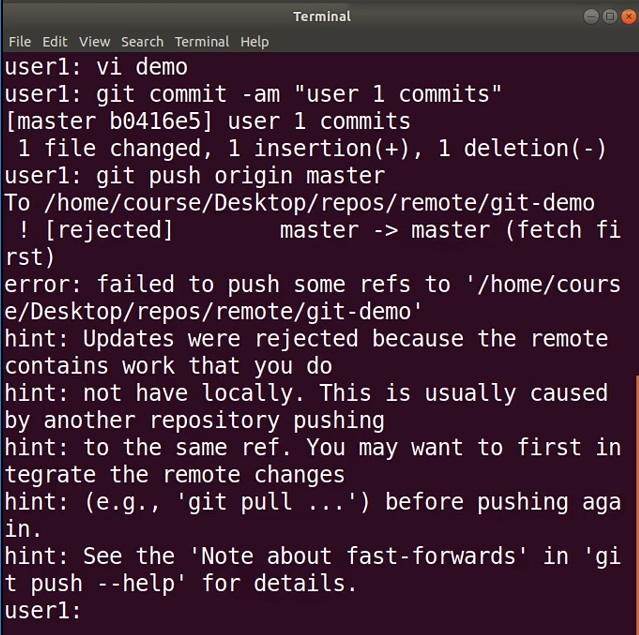




Користувач 1 у цей час бажає внести зміни в інший, раніше створений файл (тут – demo)



та зафіксувати їх у центральному репозиторії. Git попередить, що в цей час зміни були внесені в центральний репозиторій, а в локальній версії Користувача 1 їх немає.



Впертий користувач може захотіти все-одно внести свої зміни:

**git push origin master --force**

Це призведе до втрати змін Користувача 2. *Зробіть скриншот вмісту репозиторія, щоб показати відсутність відповідного файлу*.

