****

Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

 Кафедра інформаційних систем та технологій

Лабораторна робота №1

**«Системи контролю версій. Git.»**

Виконав

студент групи ІА–22:

Клочков І. Ф.

Київ 2024

**Теоретичні відомості**

**Git** — розподілена система контролю версій. Проект створив Лінус Торвальдс для керування розробкою ядра Linux, а сьогодні підтримується Джуніо Хамано (англ. Junio C. Hamano). Git є однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні та злитті гілок.

**Основні терміни та поняття в Git**

**Репозиторій (repository)** - це основна одиниця зберігання в Git, що є базою даних, де фіксуються всі зміни в проєкті. Репозиторій містить повну історію всіх комітів, включно з повними версіями всіх файлів і директорій проєкту на кожному етапі його розвитку. Це дає змогу будь-якому розробнику в будь-який момент відновити стан будь-якої версії проєкту або відстежити зміни в коді.

**Коміт (commit)** - це, по суті, «знімок» поточного стану проєкту, що зберігається в історії репозиторію. Кожен коміт містить інформацію про всі зміни у файлах порівняно з попереднім комітом, а також метадані, як-от автор, дата і коментар до коміту. Це дає змогу розробникам розуміти, коли і з якої причини було зроблено зміни.

**Гілка (branch)** - це паралельна версія сховища, яку використовують для розробки окремих функцій або експериментів, не заважаючи основній робочій версії проєкту (найчастіше «master» або «main»). Гілки дають змогу розробникам працювати над нововведеннями або виправленнями, а потім зливати свої зміни назад у головну гілку після завершення та перевірки.

**Які бувають стани в GIT і як їх розуміти**

У Git, файли в рамках робочого проєкту можуть перебувати в одному з трьох основних станів: індексований, змінений і зафіксований.

**Індексований (staged):** Коли файли додаються в індекс, Git визначає їх як підготовлені до коміту. Додавання файлів в індекс відбувається за допомогою команди git add. Цей процес дає змогу розробникам вибірково додавати зміни в наступний коміт, що забезпечує гнучкість в управлінні версіями та допомагає уникати непотрібного включення тимчасових або експериментальних змін. Фактично, індекс слугує проміжною зоною, де розробники можуть організувати та супроводжувати свої правки перед тим, як остаточно зафіксувати їх в історії репозиторію.

**Змінений (modified) стан:** Файли в стані «змінені» вказують на те, що їх було відредаговано після останнього коміту, але ще не було додано до індексу. Це первинний стан для будь-яких файлів, які ви змінюєте в робочому каталозі. Git відстежує ці модифікації, які можна переглянути за допомогою команди git status. Перш ніж файли можуть бути зафіксовані, їх необхідно додати до індексу, що дає змогу розробникам ретельно контролювати, які зміни будуть включені до наступного коміту.

**Зафіксований (committed) стан:** Коли файли зафіксовані, вони зберігаються в локальній базі даних Git, що означає, що зміни безпечно зберігаються у вашому репозиторії. Стан «зафіксований» гарантує, що всі зміни, які ви зробили та підготували, тепер будуть частиною історії сховища і можуть бути відновлені в будь-який момент або передані іншим розробникам через віддалене сховище. Фіксація правок відбувається за допомогою команди git commit, яка також вимагає додавання описового повідомлення, що пояснює суть змін для поліпшення спільної роботи та розуміння історії проєкту.

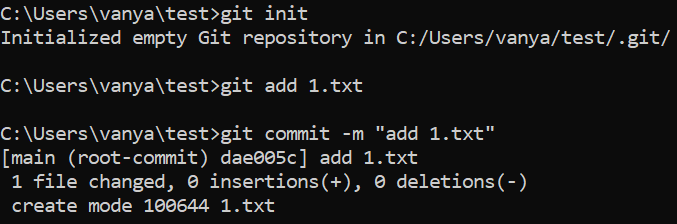
**Основні команди для роботи з Git**

**Ініціалізація та клонування**

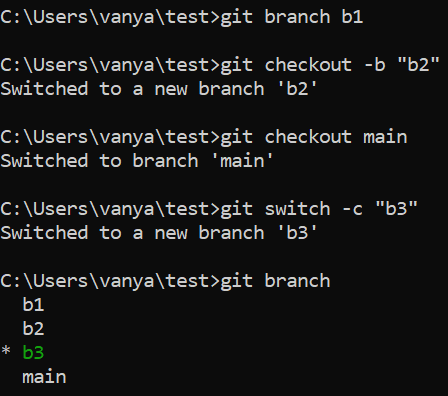
* git init – створити новий порожній локальний репозиторій.
* git clone <url> – клонувати віддалений репозиторій на локальну машину.
* git status – показати статус файлів у робочій директорії.
* git diff – показати відмінності між робочою директорією і індексом.
* git add <file> – додати файл до області індексації.
* git add . – додати всі зміни у поточній директорії до області індексації.
* git commit -m "повідомлення" – створити коміт із повідомленням.
* git log – переглянути історію комітів.
* git reset <file> – скасувати індексацію змін файлу.
* git reset --hard – скинути всі зміни до останнього коміту.
* git branch – показати всі гілки в локальному репозиторії.
* git branch <branch> – створити нову гілку.
* git branch -d <branch> – видалити локальну гілку (якщо вона вже злита).
* git branch -D <branch> – примусово видалити гілку, незалежно від її стану.
* git checkout <branch> – переключитися на іншу гілку.
* git checkout -b <branch > – створити нову гілку і одразу переключитися на неї.
* git switch <branch> – переключитися на іншу гілку.
* git merge <branch> – злити вказану гілку в поточну.
* git rebase <branch> – застосувати зміни з вказаної гілки на поточну.
* git fetch <remote> – завантажити всі зміни з віддаленого репозиторію без злиття.
* git pull <remote> – завантажити зміни з віддаленого репозиторію і автоматично злити їх із локальними.
* git push <remote> <branch> – відправити локальні зміни до віддаленого репозиторію.

**Хід виконання**

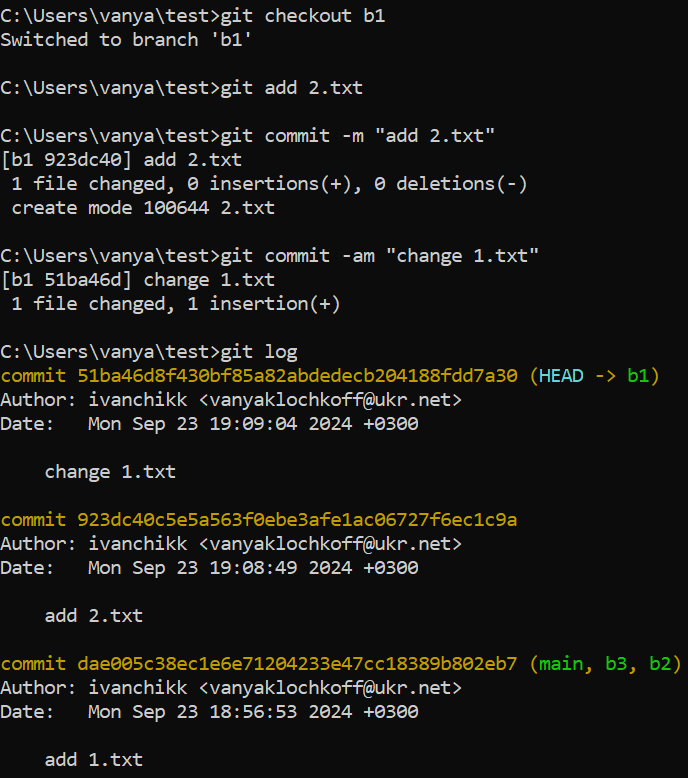
1. Ініціалізуємо новий порожній локальний репозиторій.



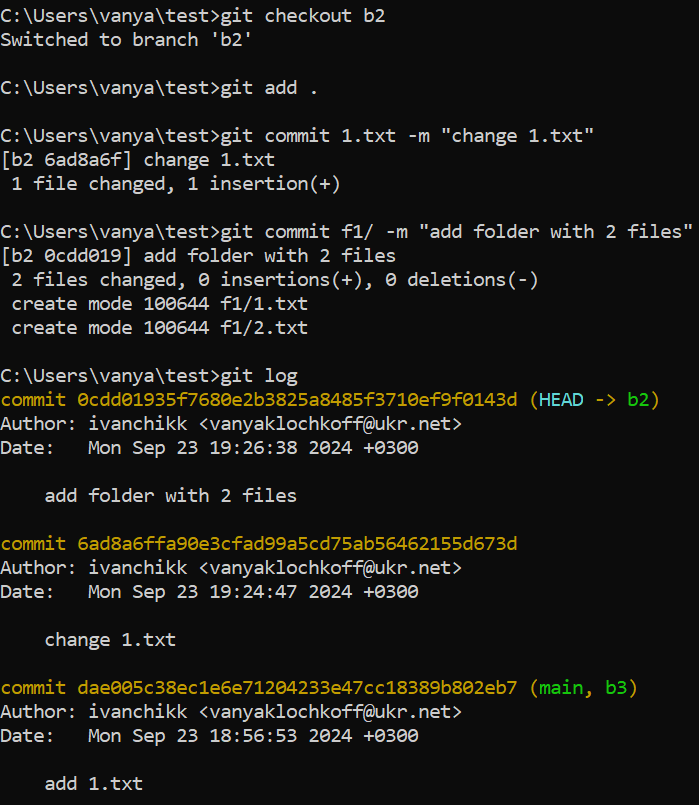
2. Створюємо три гілки різними способами.



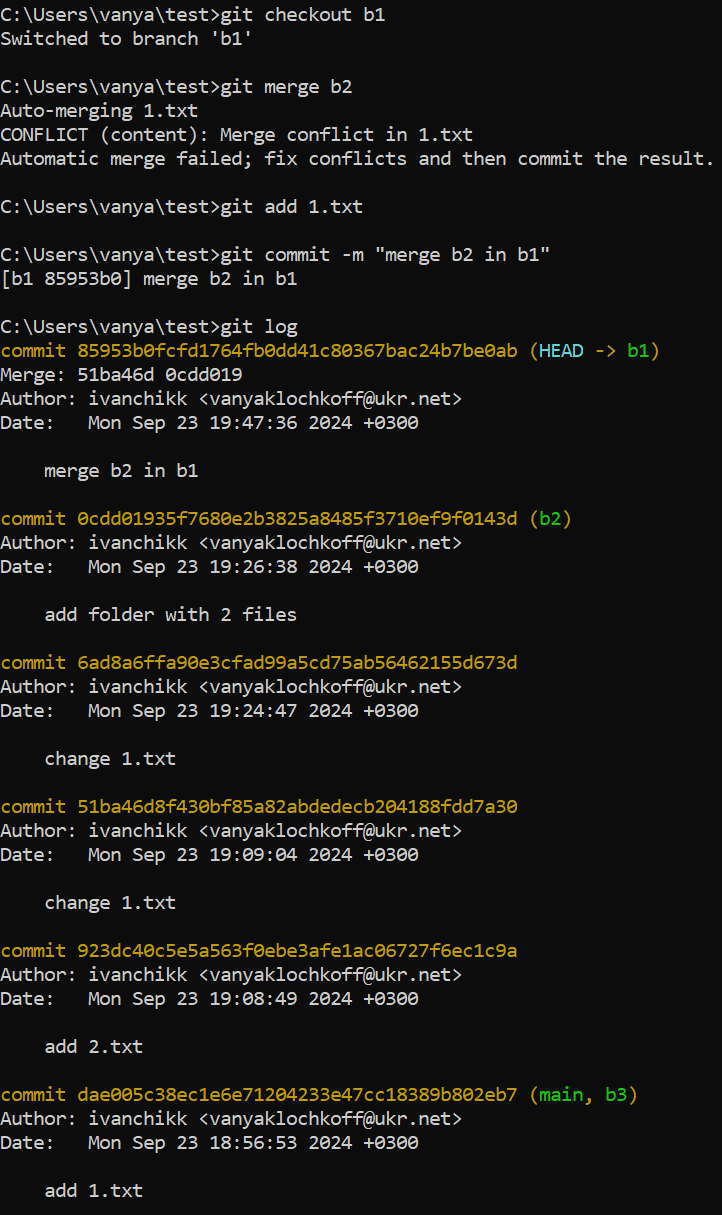
3. Змінюємо один файл та додаємо новий.



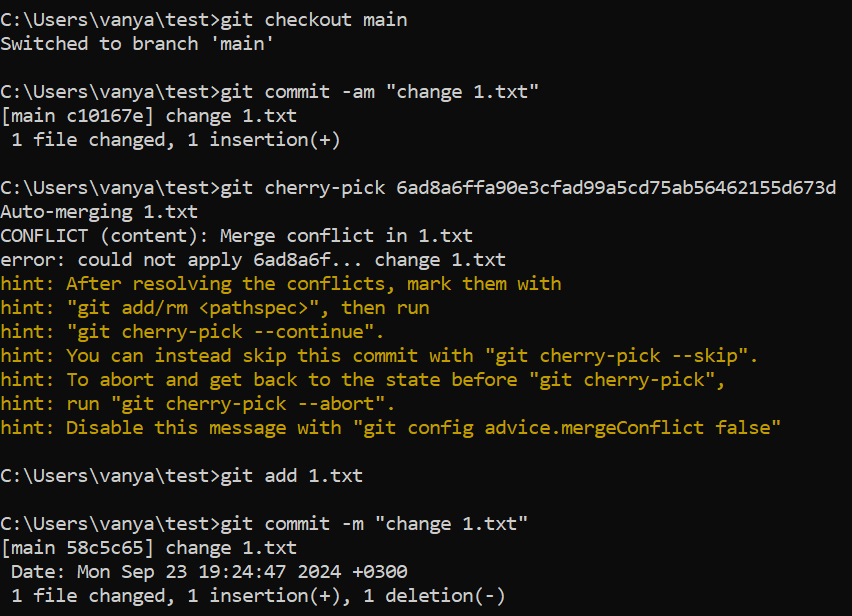
4. Змінюємо файл та додаємо папку з двома файлами двома комітами, трьома командами.



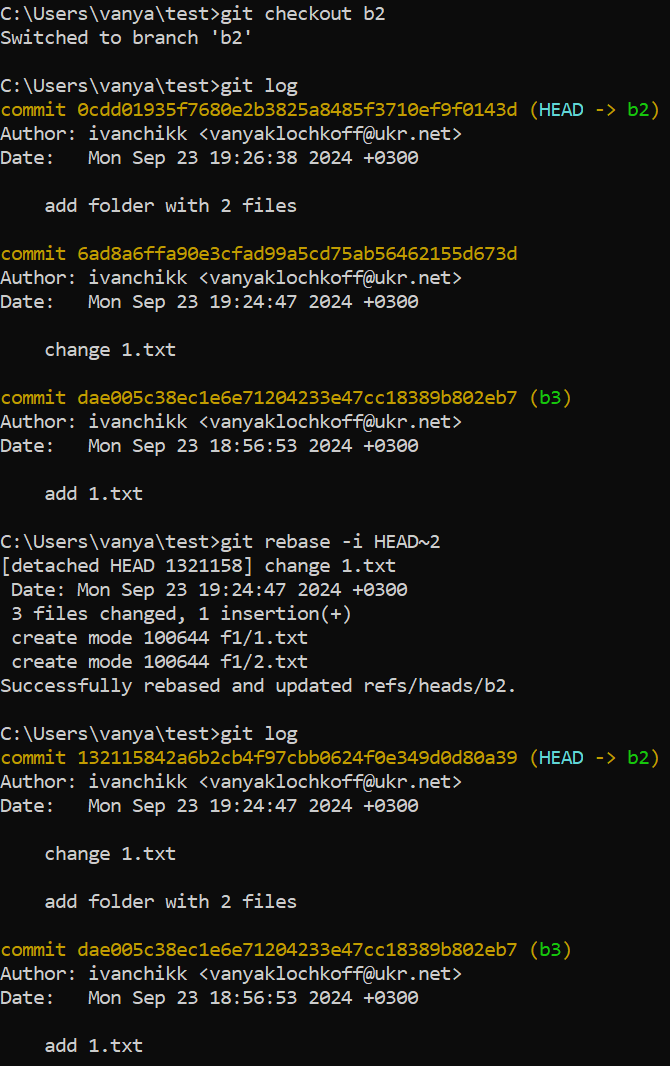
5.Мердж b2 у b1.



6.Чері-пік у main.



7. Робимо сквош за допомогою ребейзу.





**Висновок:** виконавши цю лабораторну роботу, я навчився користуватись git та виконувати базові дії за допомогою нього.