



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  
із дисципліни «**Технології розроблення програмного  
забезпечення**»  
Тема «**Команди Git**»

Виконав:

студент групи

ІА–24Чайка А.П.

Перевірив:

Мягкий М.Ю.

Київ 2024

**Мета:** ознайомитися з основними командами системи контролю версій Git

## Теоретичні відомості

**Git** — це система контролю версій, яка дозволяє розробникам відстежувати зміни в коді, співпрацювати над проектами та зберігати історію змін. Основні функції **Git** включають створення знімків (комітів) поточного стану проекту, можливість роботи з гілками для розвитку різних функціональностей незалежно одна від одної, а також злиття і вирішення конфліктів між різними версіями коду. У цій лабораторній роботі ми розглянемо основні команди **Git**, які використовуються для роботи з локальними репозиторіями, а також способи організації роботи з гілками.

### Основні команди Git

#### **git init**

Створює новий локальний репозиторій у поточній директорії. Ця команда ініціалізує репозиторій і дозволяє Git почати відстеження змін.

- `git init` — створює порожній репозиторій у поточній папці.

#### **git add**

Додає зміни у файлах до індексу (стейджингу) для подальшого коміту.

- `git add <файл>` — додає конкретний файл до індексу.
- `git add .` — додає всі файли з поточної директорії.

#### **git remove**

Видаляє файл з репозиторію та, за необхідності, з робочого каталогу.

- `git rm <файл>` — видаляє файл з індексу та робочого каталогу.
- `git rm --cached <файл>` — видаляє файл з індексу, але залишає його в робочому каталозі.

#### **git commit**

Зберігає знімок стану проекту з файлами, що були додані до індексу.

- `git commit -m "Опис змін"` — створює коміт з описом змін.
- `git commit -a -m "Опис змін"` — додає і фіксує всі змінені файли, оминувши команду `git add`.

## **git status**

Показує інформацію про поточний стан репозиторію: які файли змінені, які готові до коміту, а які потребують додавання до індексу.

- `git status` — перегляд поточного статусу файлів у репозиторії.

## **git log**

Виводить історію комітів, включаючи інформацію про авторів, час та повідомлення комітів.

- `git log` — перегляд історії всіх комітів.
- `git log --oneline` — скорочений перегляд історії комітів.

## **git branch**

Показує існуючі гілки або дозволяє створити нову гілку.

- `git branch` — показує список усіх локальних гілок.
- `git branch <назва-гілки>` — створює нову гілку з поточного стану.

## **git checkout**

Використовується для перемикання між гілками або створення нової гілки і перемикання на неї одночасно.

- `git checkout <назва-гілки>` — перемикається на існуючу гілку.
- `git checkout -b <назва-гілки>` — створює нову гілку і перемикається на неї.

## **git switch**

Новіша команда для перемикання між гілками, яка частково замінює `git checkout`. Команда `git switch` має спрощену синтаксис для роботи з гілками.

- `git switch <назва-гілки>` — перемикається на існуючу гілку.
- `git switch -c <назва-гілки>` — створює нову гілку і перемикається на неї.

## **git merge**

Зливає зміни з однієї гілки в іншу. Під час злиття Git намагається автоматично об'єднати зміни, але може виникнути конфлікт, якщо зміни в одному й тому ж файлі були внесені в обох гілках.

- `git merge <назва-гілки>` — зливає зміни з вказаної гілки в поточну.

### **git rebase**

Інструмент для переписування історії комітів. Використовується для інтеграції змін з однієї гілки в іншу без створення окремого коміту злиття. Це дозволяє зберегти лінійну історію проєкту.

- `git rebase <назва-гілки>` — змінює базу поточної гілки на вказану, інтегруючи зміни.

### **git cherry-pick**

Використовується для вибіркового перенесення окремих комітів з однієї гілки в іншу. Це корисно, коли потрібно інтегрувати певні зміни без злиття всієї гілки.

- `git cherry-pick <хеш-коміту>` — застосовує вказаний коміт до поточної гілки.

### **git reset**

Використовується для скасування комітів або скасування змін в індексі.

- `git reset <файл>` — знімає файл зі стейджингу.
- `git reset --hard <хеш-коміту>` — повертає репозиторій до конкретного коміту, видаляючи всі зміни після нього.

## **Хід роботи**

1. Від основної гілки створити 2 гілки - гілку feature-1 та feature-2

```
C:\Users\kille\git-rep>git init
Initialized empty Git repository in C:/Users/kille/git-rep/.git/

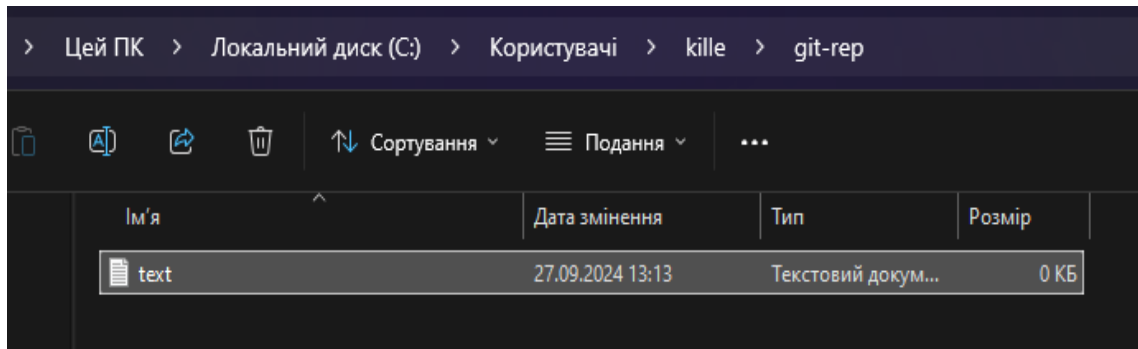
C:\Users\kille\git-rep>git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

C:\Users\kille\git-rep>
```

Ініціалізуємо репозиторій. Як бачимо поки немає відстежуваних файлів



Створюємо файл в нашій директорії

```
C:\Users\kille\git-rep>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  text.txt

nothing added to commit but untracked files present (use "git add" to track)

C:\Users\kille\git-rep>git add .

C:\Users\kille\git-rep>git commit -m "initial commit"
[master (root-commit) eaafa8ba] initial commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 text.txt

C:\Users\kille\git-rep>git status
On branch master
nothing to commit, working tree clean

C:\Users\kille\git-rep>git log --oneline
eaafa8ba (HEAD -> master) initial commit
```

Додаємо цей файл до системи контролю версій та комітимо

```
C:\Users\kille\git-rep>git branch feature-1

C:\Users\kille\git-rep>git status
On branch master
nothing to commit, working tree clean

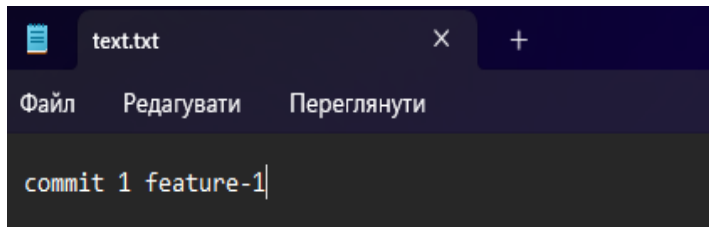
C:\Users\kille\git-rep>git switch -c feature-2
Switched to a new branch 'feature-2'

C:\Users\kille\git-rep>git branch
  feature-1
* feature-2
  master
```

Створюємо гілки feature-1 та feature-2

2. В гілці feature-1 створити 2 коміти, що будуть відноситися тільки до цієї гілки. В гілці feature-2 створити 4 коміти

```
C:\Users\kille\git-rep>git checkout feature-1  
Switched to branch 'feature-1'
```

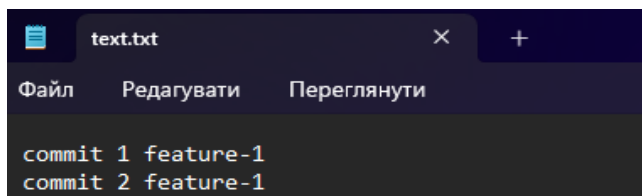


text.txt

Файл Редагувати Переглянути

commit 1 feature-1

```
C:\Users\kille\git-rep>git add .  
  
C:\Users\kille\git-rep>git commit -m "commit 1"  
[feature-1 2efc422] commit 1  
1 file changed, 1 insertion(+)
```



text.txt

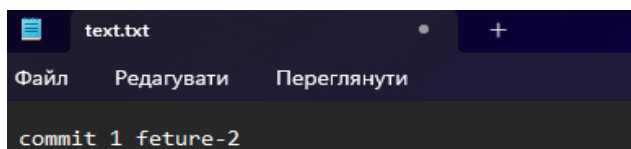
Файл Редагувати Переглянути

commit 1 feature-1  
commit 2 feature-1

```
C:\Users\kille\git-rep>git commit -m "commit 2"  
[feature-1 c437503] commit 2  
1 file changed, 2 insertions(+), 1 deletion(-)
```

Робимо в гілці feature-1 два коміти

```
C:\Users\kille\git-rep>git checkout feature-2  
Switched to branch 'feature-2'
```

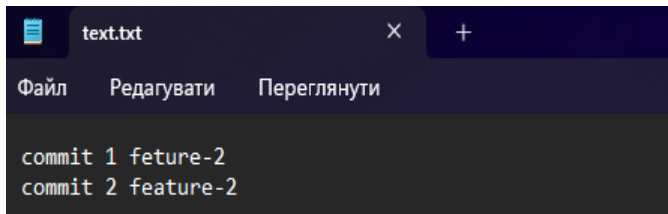


text.txt

Файл Редагувати Переглянути

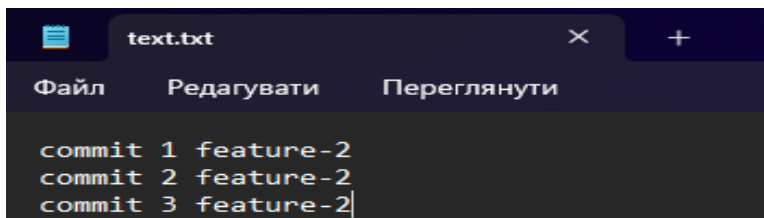
commit 1 feture-2

```
C:\Users\kille\git-rep>git commit -a -m "commit 1"  
[feature-2 13b4022] commit 1  
1 file changed, 1 insertion(+)
```



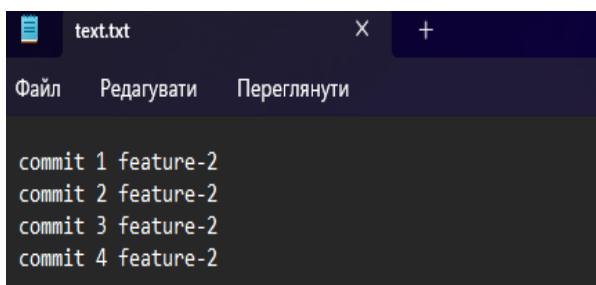
```
commit 1 feture-2
commit 2 feature-2
```

```
C:\Users\kille\git-rep>git commit -a -m "commit 2"
[feature-2 8129912] commit 2
1 file changed, 2 insertions(+), 1 deletion(-)
```



```
commit 1 feature-2
commit 2 feature-2
commit 3 feature-2
```

```
C:\Users\kille\git-rep>git commit -a -m "commit 3"
[feature-2 aa5b7c3] commit 3
1 file changed, 3 insertions(+), 2 deletions(-)
```



```
commit 1 feature-2
commit 2 feature-2
commit 3 feature-2
commit 4 feature-2
```

```
C:\Users\kille\git-rep>git commit -a -m "commit 4"
[feature-2 6f75280] commit 4
1 file changed, 2 insertions(+), 1 deletion(-)

C:\Users\kille\git-rep>git log --oneline
6f75280 (HEAD -> feature-2) commit 4
aa5b7c3 commit 3
8129912 commit 2
13b4022 commit 1
eaafa8ba (master) initial commit

C:\Users\kille\git-rep>
```

Додаємо 4 коміти в гілку feature-2

3. Від гілки feature-1 робимо гілку feature-3 та переносимо в неї третій коміт з гілки feature-2

```
c:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git switch feature-1
Switched to branch 'feature-1'

c:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git checkout -b feature-3
Switched to a new branch 'feature-3'

c:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git cherry-pick 82fc3d4
Auto-merging text.txt
CONFLICT (content): Merge conflict in text.txt
error: could not apply 82fc3d4... Commit 3
hint: After resolving the conflicts, mark them with
hint: "git add/rm <paths>", then run
hint: "git cherry-pick --continue".
hint: You can instead skip this commit with "git cherry-pick --skip".
hint: To abort and get back to the state before "git cherry-pick",
hint: run "git cherry-pick --abort".
```

Вирішуємо конфлікти та продовжуємо перенос коміта

```
commit 1 feature-2
commit 2 feature-2
commit 3 feature-2
```

```
[feature-3 22adad8] commit 3
Date: Fri Sep 27 13:24:48 2024 +0300
1 file changed, 5 insertions(+), 2 deletions(-)

C:\Users\kille\git-rep>git log --oneline
22adad8 (HEAD -> feature-3) commit 3
c437503 (feature-1) commit 2
2efc422 commit 1
eaafa8ba (master) initial commit
```

Коміт перенесено

4. Переносимо всі коміти з гілки feature-3 в основну гілку

```
C:\Users\kille\git-rep>git checkout master
Switched to branch 'master'

C:\Users\kille\git-rep>git merge feature-3
Updating eaafa8ba..22adad8
Fast-forward
 text.txt | 5 +++++
1 file changed, 5 insertions(+)
```

Всі зміни з гілки feature-3 перенесено в master



**Висновок:** в ході виконання даної лабораторної роботи ми познайомились з такою системою контролю версій як Git. Ми вивчили основні команди для роботи з гітом: ініціалізація репозиторію, додавання файлів у систему контролю, створення коміту, перенесення комітів між гілками, злиття гілок та багато інших. Окрім того ми застосували ці команди на практиці, вирішили конфлікти при переносі комітів та засвоїли вивчений матеріал.