



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–24
Яблонський Д.Б.

Перевірів:
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 проекту.

- `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 стеїджингу.
- `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

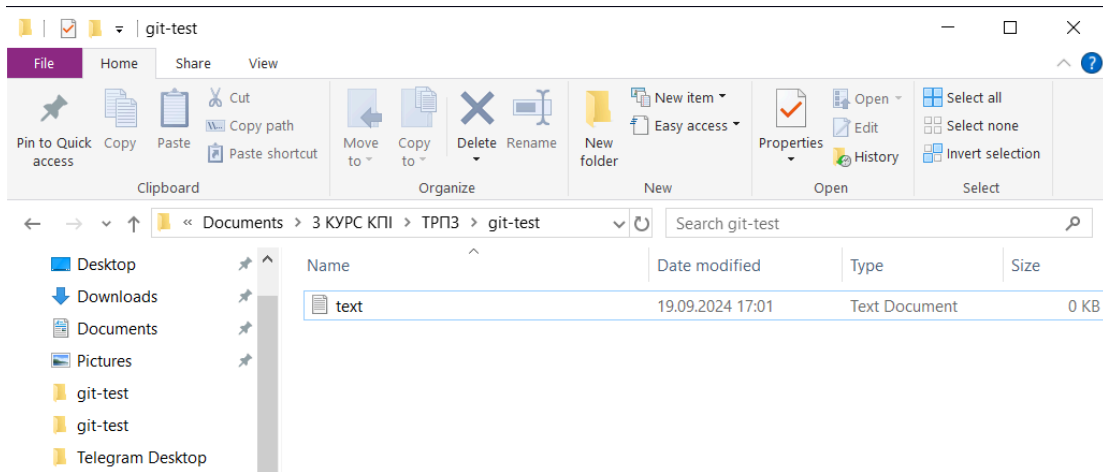
```
c:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git init
Initialized empty Git repository in C:/Users/yablonya/Documents/3 КУРС КПІ/ТРПЗ/git-test/.git/

c:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git status
On branch master

No commits yet

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

Ініціалізуємо репозиторій. Як бачимо поки немає відстежуваних файлів



Створюємо файл в нашій директорії

```
c:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>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\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git add .

c:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git commit -m "initial commit"
[master (root-commit) c844b34] initial commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 text.txt

c:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git status
On branch master
nothing to commit, working tree clean

c:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git log --oneline
c844b34 (HEAD -> master) initial commit
```

Додаємо цей файл до системи контролю версій та комітимо

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

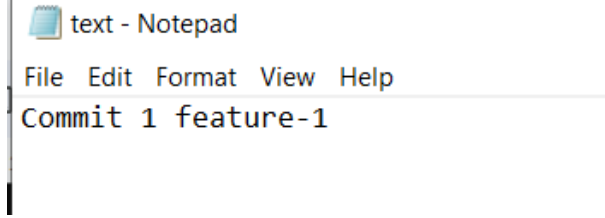
c:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git switch -c feature-2
Switched to a new branch 'feature-2'

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



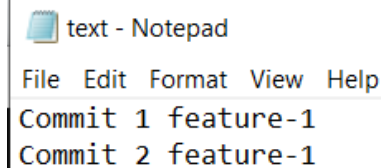
text - Notepad

File Edit Format View Help

Commit 1 feature-1

```
c:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git add text.txt

c:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git commit -m "Commit 1"
[feature-1 85aaff0] Commit 1
1 file changed, 1 insertion(+)
```



text - Notepad

File Edit Format View Help

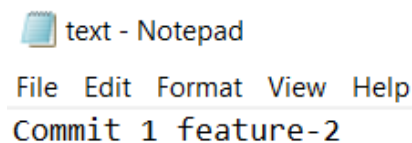
Commit 1 feature-1
Commit 2 feature-1

```
c:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git commit -a -m "Commit 2"
[feature-1 a92f7a9] Commit 2
1 file changed, 2 insertions(+), 1 deletion(-)

c:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git status
On branch feature-1
nothing to commit, working tree clean
```

Робимо в гілці feature-1 два коміти

```
c:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git switch feature-2
Switched to branch 'feature-2'
```



text - Notepad

File Edit Format View Help

Commit 1 feature-2

```
c:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git commit -a -m "Commit 1"
[feature-2 0ea23de] Commit 1
1 file changed, 1 insertion(+)
```

```
text - Notepad
File Edit Format View Help
Commit 1 feature-2
Commit 2 feature-2
```

```
c:\Users\yablonya\Documents\3 КУРС КПИ\ТРПЗ\git-test>git commit -a -m "Commit 2"
[feature-2 3aaf332] Commit 2
1 file changed, 2 insertions(+), 1 deletion(-)
```

```
*text - Notepad
File Edit Format View Help
Commit 1 feature-2
Commit 2 feature-2
Commit 3 feature-2
```

```
c:\Users\yablonya\Documents\3 КУРС КПИ\ТРПЗ\git-test>git commit -a -m "Commit 3"
[feature-2 82fc3d4] Commit 3
1 file changed, 2 insertions(+), 1 deletion(-)
```

```
text - Notepad
File Edit Format View Help
Commit 1 feature-2
Commit 2 feature-2
Commit 3 feature-2
Commit 4 feature-2
```

```
c:\Users\yablonya\Documents\3 КУРС КПИ\ТРПЗ\git-test>git commit -a -m "Commit 4"
[feature-2 205f5a2] Commit 4
1 file changed, 2 insertions(+), 1 deletion(-)
```

```
c:\Users\yablonya\Documents\3 КУРС КПИ\ТРПЗ\git-test>git log --oneline
205f5a2 (HEAD -> feature-2) Commit 4
82fc3d4 Commit 3
3aaf332 Commit 2
0ea23de Commit 1
c844b34 (master) initial commit
```

Додаємо 4 коміти в гілку feature-2

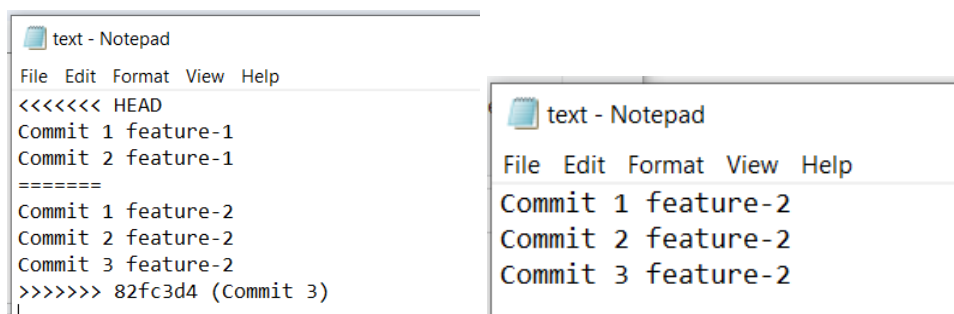
3. Від гілки feature-1 робимо гілку feature-3 та переносимо в неї третій коміт з гілки feature-2

```
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



```
C:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git add .

C:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git cherry-pick --continue
[feature-3 431a34c] Commit 3
Date: Thu Sep 19 17:19:31 2024 +0300
1 file changed, 3 insertions(+), 2 deletions(-)

C:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git log --oneline
431a34c (HEAD -> feature-3) Commit 3
a92f7a9 (feature-1) Commit 2
85aaff0 Commit 1
c844b34 (master) initial commit
```

Коміт перенесено

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

```
C:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git checkout master
Switched to branch 'master'

C:\Users\yablonya\Documents\3 КУРС КПІ\ТРПЗ\git-test>git merge feature-3
Updating c844b34..431a34c
Fast-forward
 text.txt | 3 +++
1 file changed, 3 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.