Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційні систем та технологій

# Лабораторна робота №1

із дисципліни «*Технології розроблення програмного забезпечення*»

**Тема: «**Системи контролю версій. Git**»**

Виконав:
Студент групи ІА-24
Котлярчук М. С.

Перевірив:
Мягкий М. Ю.

Київ-2024

**Мета:** навчитися базових команд системи контролю версій Git.

## Теоретичні відомості

**Git** — це система контролю версій, яка дозволяє розробникам відстежувати зміни в коді, співпрацювати над проєктами та зберігати історію змін. Основні функції Git включають створення знімків (комітів) поточного стану проєкту, можливість роботи з гілками для розвитку різних функціональностей незалежно одна від одної, а також злиття і вирішення конфліктів між різними версіями коду. Ось опис основних команд Git:

### Основні команди Git:

**git init**
Створює новий локальний репозиторій у поточній директорії.

- 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 проєкту 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.

- git status — перегляд поточного статусу файлів у репозиторії.

**git log**
Виводить історію комітів.

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

**git branch**
Показує існуючі гілки або дозволяє створити нову.

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

**git checkout**
Перемикається між гілками або створює нову гілку і перемикається на неї.

- 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.

- 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.

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

**git rebase**
Використовується для переписування історії комітів і інтеграції змін без створення окремого коміту злиття.

- git rebase <назва-гілки> — змінює базу поточної гілки на вказану.

**git cherry-pick**
Переносить окремі коміти з однієї гілки в іншу.

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

**git reset**
Скасовує коміти або зміни 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 3 гілки трьома різними способами

```
C:\Users\User\Desktop\TRPZ>git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in C:/Users/User/Desktop/TRPZ/.git/

C:\Users\User\Desktop\TRPZ>touch text.txt

C:\Users\User\Desktop\TRPZ>git add .

C:\Users\User\Desktop\TRPZ>git commit -m "Init"
[master (root-commit) 41d2c69] Init
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 text.txt
```

Ініціалізуємо репозиторій, тут не міститься відстежуваних файлів, тому створюємо його та додаємо до стейджу, після чого комітимо.

```
C:\Users\User\Desktop\TRPZ>git branch br1

C:\Users\User\Desktop\TRPZ>git checkout -b br2
Switched to a new branch 'br2'

C:\Users\User\Desktop\TRPZ>git switch master
Switched to branch 'master'

C:\Users\User\Desktop\TRPZ>git switch -c br3
Switched to a new branch 'br3'

C:\Users\User\Desktop\TRPZ>git branch
  br1
  br2
* br3
  master
```

Створюємо гілки br1, br2 та br3. Виконуємо команду git branch для перевірки, що гілки були створені.

2. Додати до першої гілки 1 коміт, для другої 2 коміти та для третьої 3 коміти

```
C:\Users\User\Desktop\TRPZ>git commit -am "com1 br3"
[br3 2c78786] com1 br3
 1 file changed, 1 insertion(+)

C:\Users\User\Desktop\TRPZ>git commit -am "com2 br3"
[br3 7eecbcc] com2 br3
 1 file changed, 2 insertions(+), 1 deletion(-)

C:\Users\User\Desktop\TRPZ>git commit -am "com3 br3"
[br3 378d7ef] com3 br3
 1 file changed, 2 insertions(+), 1 deletion(-)

C:\Users\User\Desktop\TRPZ>git switch br2
Switched to branch 'br2'

C:\Users\User\Desktop\TRPZ>git commit -am "com1 br2"
[br2 47c11af] com1 br2
 1 file changed, 1 insertion(+)

C:\Users\User\Desktop\TRPZ>git commit -am "com2 br2"
[br2 29c0f3d] com2 br2
 1 file changed, 2 insertions(+), 1 deletion(-)

C:\Users\User\Desktop\TRPZ>git switch br1
Switched to branch 'br1'

C:\Users\User\Desktop\TRPZ>git commit -am "com1 br1"
[br1 061aa4e] com1 br1
 1 file changed, 1 insertion(+)
```

Переглянемо результат:

```
C:\Users\User\Desktop\TRPZ>git log --all --oneline
061aa4e (HEAD -> br1) com1 br1
29c0f3d (br2) com2 br2
47c11af com1 br2
378d7ef (br3) com3 br3
7eecbcc com2 br3
2c78786 com1 br3
41d2c69 (master) Init
```

3. Об’єднати 3 коміти гілки br3 в один

```
C:\Users\User\Desktop\TRPZ>git rebase -i HEAD~3
[detached HEAD 64881e9] com1 br3
 Date: Fri Sep 27 11:40:07 2024 +0300
 1 file changed, 3 insertions(+)
Successfully rebased and updated refs/heads/br3.

C:\Users\User\Desktop\TRPZ>git log
commit 64881e90cfe9d2c76c81c657d0ceb57b7505c84a (HEAD -> br3)
Author: tmips <kotlyarchuk.maxim@gmail.com>
Date:   Fri Sep 27 11:40:07 2024 +0300

    com1 br3

    com2 br3

    com3 br3

commit 41d2c694f3a62152e51b79b11614f9fdd2c5d200 (master)
Author: tmips <kotlyarchuk.maxim@gmail.com>
Date:   Fri Sep 27 11:35:06 2024 +0300

    Init
```

У текстовому редакторі ми отримали три коміти зі словом pick перед ними. Для другого та третього комітів ми написали команду squash, для того щоб об’єднати їх з першим. Виконали git log, де побачили що все правильно.

4. Створити конфлікти за допомогою команд rebase і merge, вирішити їх

```
C:\Users\User\Desktop\TRPZ>git switch master
Switched to branch 'master'

C:\Users\User\Desktop\TRPZ>git merge br1
Auto-merging text.txt
CONFLICT (content): Merge conflict in text.txt
Automatic merge failed; fix conflicts and then commit the result.

C:\Users\User\Desktop\TRPZ>git add .

C:\Users\User\Desktop\TRPZ>git merge --continue
[master 7390f39] Merge branch 'br1' with master
```

Ми додали коміт на гілці master та виконали команду merge, але через те що зміни в одному й тому ж файлі на двох гілках суперечать одна одній виник конфлікт. Ми відредагували файл та продовжили виконання команди.

```
C:\Users\User\Desktop\TRPZ>git switch br2
Switched to branch 'br2'

C:\Users\User\Desktop\TRPZ>git rebase master
warning: skipped previously applied commit 47c11af
hint: use --reapply-cherry-picks to include skipped commits
hint: Disable this message with "git config advice.skippedCherryPicks false"
Auto-merging text.txt
CONFLICT (content): Merge conflict in text.txt
error: could not apply 29c0f3d... com2 br2
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --abort".
Could not apply 29c0f3d... com2 br2
```

```
C:\Users\User\Desktop\TRPZ>git add .

C:\Users\User\Desktop\TRPZ>git rebase --continue
[detached HEAD bc9b278] br2 rebase master
 1 file changed, 3 insertions(+)
Successfully rebased and updated refs/heads/br2.
```

Потім виконували команду rebase, вирішили конфлікт та закомітили зміни.

5. Переглянули історію комітів у вигляді графу, щоб краще було видно зв’язки між комітами та гілками

```
C:\Users\User\Desktop\TRPZ>git log --all --oneline --graph
* bc9b278 (HEAD -> br2) br2 rebase master
*   7390f39 (master) Merge branch 'br1' with master
|\
| * 061aa4e (br1) com1 br1
* | 05a5229 com 1 master
|/
| * 64881e9 (br3) com1 br3
|/
* 41d2c69 Init
```

**Висновки:** під час виконання даної лабораторної роботи я навчився працювати з основними командами Git. Продемонстрував створення гілок, додавання комітів, об’єднання комітів, створення та вирішення конфліктів за використання команд merge та rebase.