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»**

Виконала:

студентка групи ІА–24 Мелешко Ю.С.

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 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: 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.

Завдання

Зробити новий репозиторій

```
klubn@DESKTOP-QEG3RPV MINGW64 ~/Documents/trpz
$ git init
Initialized empty Git repository in C:/Users/klubn/Documents/trpz/.git/
```

Додати файл та зробити коміт

```
klubn@DESKTOP-QEG3RPV MINGW64 ~/Documents/trpz (main)
$ git commit -m"Added new file"
[main (root-commit) 0647d15] Added new file
 1 file changed, 1 insertion(+)
 create mode 100644 new.txt
```

Створити нову гілку та перейти на неї

```
$ git checkout -b first
Switched to a new branch 'first'
```

Створити два пустих коміти

```
klubn@DESKTOP-QEG3RPV MINGW64 ~/Documents/trpz (first)
$ git commit --allow-empty -m"Empty commit"
[first 9043cdb] Empty commit

klubn@DESKTOP-QEG3RPV MINGW64 ~/Documents/trpz (first)
$ git commit --allow-empty -m"Second Empty commit"
[first 9eeab0c] Second Empty commit
```

Перейти на гілку main , та зробити squash merge

```
klubn@DESKTOP-QEG3RPV MINGW64 ~/Documents/trpz (first)
$ git checkout main
Switched to branch 'main'

klubn@DESKTOP-QEG3RPV MINGW64 ~/Documents/trpz (main)
$ git merge --squash first
Updating 0647d15..9eeab0c
Fast-forward
Squash commit -- not updating HEAD
```

Додаємо файли first.txt та second.txt двома різними комітами

```
klubn@DESKTOP-QEG3RPV MINGW64 ~/Documents/trpz (main)
$ git add first.txt

klubn@DESKTOP-QEG3RPV MINGW64 ~/Documents/trpz (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   first.txt
```

```
klubn@DESKTOP-QEG3RPV MINGW64 ~/Documents/trpz (main)
$ git commit -m"Added first"
[main 89072be] Added first
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 first.txt
```

```
klubn@DESKTOP-QEG3RPV MINGW64 ~/Documents/trpz (main)
$ git add second.txt

klubn@DESKTOP-QEG3RPV MINGW64 ~/Documents/trpz (main)
$ git commit -m"Added second"
[main 3cd5283] Added second
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 second.txt
```

Перевіряємо історію комітів

```
klubn@DESKTOP-QEG3RPV MINGW64 ~/Documents/trpz (main)
$ git log
commit 3cd5283bdcfd8bdc36d28db33ac6b7491fbb9c7a (HEAD -> main)
Author: Julya <julya.meleshko@gmail.com>
Date:   Sat Nov 2 20:44:54 2024 +0200

    Added second

commit 89072be680c57f0b209c4b3243006d2bbc90de0b
Author: Julya <julya.meleshko@gmail.com>
Date:   Sat Nov 2 20:44:41 2024 +0200

    Added first

commit 0647d158f8b733b16c278fc9bdf0720637493e70
Author: Julya <julya.meleshko@gmail.com>
Date:   Sat Nov 2 20:39:47 2024 +0200

    Added new file
```

Використовуємо git reset

```
klubn@DESKTOP-QEG3RPV MINGW64 ~/Documents/trpz (main)
$ git reset HEAD~1
```

```
klubn@DESKTOP-QEG3RPV MINGW64 ~/Documents/trpz (main)
$ git log
commit 89072be680c57f0b209c4b3243006d2bbc90de0b (HEAD -> main)
Author: Julya <julya.meleshko@gmail.com>
Date:   Sat Nov 2 20:44:41 2024 +0200

    Added first

commit 0647d158f8b733b16c278fc9bdf0720637493e70
Author: Julya <julya.meleshko@gmail.com>
Date:   Sat Nov 2 20:39:47 2024 +0200

    Added new file
```

Робимо інтерактивний rebase

```
klubn@DESKTOP-QEG3RPV MINGW64 ~/Documents/trpz (main)
$ git rebase -i HEAD~1
[detached HEAD 57feb3f] First file
 Date: Sat Nov 2 20:44:41 2024 +0200
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 first.txt
Successfully rebased and updated refs/heads/main.
```

Змінили назву коміту

```
klubn@DESKTOP-QEG3RPV MINGW64 ~/Documents/trpz (main)
$ git log
commit 57feb3f2ec65df4e2c87035b185e6e13ca11e9e8 (HEAD -> main)
Author: Julya <julya.meleshko@gmail.com>
Date:   Sat Nov 2 20:44:41 2024 +0200

    First file

commit 0647d158f8b733b16c278fc9bdf0720637493e70
Author: Julya <julya.meleshko@gmail.com>
Date:   Sat Nov 2 20:39:47 2024 +0200

    Added new file
```

```
klubn@DESKTOP-QEG3RPV MINGW64 ~/Documents/trpz (main)
$ git log --all --oneline --graph
* 57feb3f (HEAD -> main) First file
| * 9eeab0c (first) Second Empty commit
| * 9043cdb Empty commit
|/
* 0647d15 Added new file
```

Висновок: в ході виконання даної лабораторної роботи ми познайомились з такою системою контролю версій як Git , та основні команди для роботи з Git.