



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: «Команди Git»

Виконав:
Студент групи ІА-24
Боднар А. Д.

Перевірив:
Мягкий М.Ю.

Київ-2024

Мета: ознайомитися з основними командами системи контролю версій Git

Теоретичні відомості

Git - це система керування версіями, яка допомагає розробникам відслідковувати зміни в коді, спільно працювати над проектами та зберігати історію редагувань. Ключові можливості Git включають створення знімків (комітів) поточного стану проекту, роботу з гілками для розвитку різних функцій окремо, а також об'єднання та вирішення конфліктів між різними версіями коду. У цій лабораторній роботі ми розглянемо основні команди Git для роботи з локальними репозиторіями та методи організації роботи з гілками.

Основні команди

1. **git init**

Ініціалізує новий Git репозиторій у поточній папці.

- **git init <directory>**: Ініціалізує Git репозиторій у вказаній директорії.

2. **git add .**

Додає всі зміни в робочій директорії до індексу (стейджинг).

- **git add <file>**: Додає конкретний файл до індексу.

3. **git rm**

Видаляє файли з робочої директорії та індексу.

- **git rm <file>**: Видаляє файл.

4. **git commit**

Фіксує зміни в репозиторії.

- **git commit -m "<message>"**: Фіксує зміни з повідомленням.
- **git commit <file>**: Фіксує тільки окремий файл

5. git status

Показує статус робочої директорії та індексу.

- **git status -s**: Показує короткий статус (однорядковий формат).
- **git status -b**: Виводить інформацію про поточну гілку.

6. git log

Показує історію комітів.

- **git log --oneline**: Виводить історію комітів у компактному вигляді (хеш і повідомлення в одному рядку).
- **git log --graph --oneline --decorate**: Виводить історію у вигляді графа гілок із комітами.

7. git branch

Управління гілками репозиторію.

- **git branch**: Виводить список гілок.
- **git branch <name>**: Створює нову гілку.
- **git branch -d <name>**: Видаляє локальну гілку.
- **git branch -m <old-name> <new-name>**: Перейменовує гілку.

8. git checkout

Перемикає на іншу гілку або коміт.

- **git checkout <branch>**: Перемикає на зазначену гілку.
- **git checkout -b <branch>**: Створює нову гілку та перемикає на неї.
- **git checkout <commit-hash>**: Перемикає на конкретний коміт

9. git switch

Ця команда використовується для перемикання між гілками, і є новішою альтернативою команді **git checkout** для роботи з гілками.

- **git switch <branch>**: Перемикає на існуючу гілку.
- **git switch -c <new-branch>**: Створює нову гілку та перемикає на неї.

- **git switch --detach <commit-hash>**: Перемикає на конкретний коміт

10. git merge

Зливає зміни з іншої гілки в поточну.

- **git merge <branch>**: Зливає зазначену гілку з поточною.

11. git rebase

Переміщує історію комітів поточної гілки на іншу.

- **git rebase <branch>**: Переміщує поточну гілку поверх іншої гілки.

12. git cherry-pick

Копіює зміни, зроблені в конкретному коміті, в поточну гілку.

- **git cherry-pick <commit-hash>**: Переносить коміт у поточну гілку.

13. git reset

Повертає поточний стан репозиторію до певного коміту.

- **git reset --soft <commit-hash>**: Повертає стан до зазначеного коміта, але зберігає зміни в індексі.
- **git reset --hard <commit-hash>**: Повертає стан до зазначеного коміта та видаляє всі зміни як з індексу, так і з робочої директорії.

14. git rm (remove)

Команда для видалення файлів із робочої директорії та індексу:


- **git rm <file>**: Видаляє файл з робочої директорії та індексу.
- **git rm --cached <file>**: Видаляє файл тільки з індексу (залишає його в робочій директорії).
- **git rm -r <directory>**: Видаляє вказану директорію разом з усіма файлами всередині неї.

Хід роботи

Ініціалізуємо репозиторій

```
C:\Styding\TRPZ\lab1>git init
Initialized empty Git repository in C:/Styding/TRPZ/lab1/.git/
```

Створюємо файл, стейджим його а потім робимо початковий КОМІТ

 main

```
C:\Styding\TRPZ\lab1>git add .

C:\Styding\TRPZ\lab1>git commit -m "start"
[master (root-commit) e75ebf4] start
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 main.txt
```

Створення трьох гілок

```
C:\Styding\TRPZ\lab1>git branch b1
```

```
C:\Styding\TRPZ\lab1>git branch b2
```

```
C:\Styding\TRPZ\lab1>git branch b3
```

Створення ще двох за допомогою двох інших способів

```
C:\Styding\TRPZ\lab1>git switch -c b4
Switched to a new branch 'b4'

C:\Styding\TRPZ\lab1>git switch -
Switched to branch 'master'


C:\Styding\TRPZ\lab1>git checkout -b b5
Switched to a new branch 'b5'
```

Перехід на 1 гілку та створення коміту

```
C:\Styding\TRPZ\lab1>git switch b1
Switched to branch 'b1'

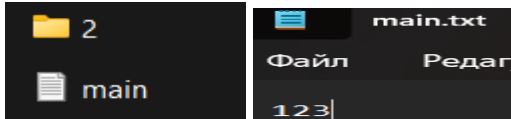
C:\Styding\TRPZ\lab1>git add .

C:\Styding\TRPZ\lab1>git commit -m "c1 on b1"
[b1 8dceb0f] c1 on b1
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt
```

 main

 1

Переходимо на 2 гілку створюємо папку з 2 файлами та редагуємо файл main, після чого за 3 команди їх комітимо за 2 коміти



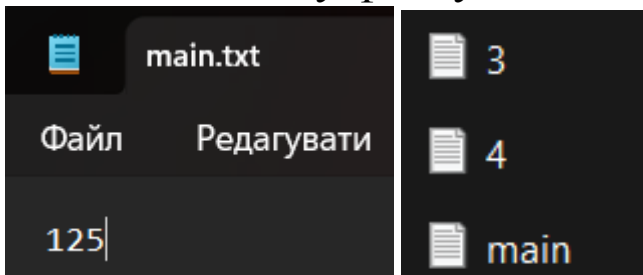
```
C:\Styding\TRPZ\lab1>git switch b2
Switched to branch 'b2'

C:\Styding\TRPZ\lab1>git add .

C:\Styding\TRPZ\lab1>git commit main.txt -m "c1 on b2"
[b2 784319a] c1 on b2
1 file changed, 1 insertion(+)

C:\Styding\TRPZ\lab1>git commit -m "c2 on b2"
[b2 be8b40e] c2 on b2
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 2/21.txt
create mode 100644 2/22.txt
```

Переходимо на 3 гілку, створюємо 2 файли та редагуємо main, потім по одному файлу комітимо, в результаті отримуємо 3 коміти



```
C:\Styding\TRPZ\lab1>git switch b3
Switched to branch 'b3'

C:\Styding\TRPZ\lab1>git add .

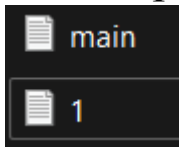
C:\Styding\TRPZ\lab1>git add .

C:\Styding\TRPZ\lab1>git commit 3.txt -m "c1 on b3"
[b3 ad30657] c1 on b3
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 3.txt

C:\Styding\TRPZ\lab1>git commit 4.txt -m "c2 on b3"
[b3 da5c501] c2 on b3
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 4.txt

C:\Styding\TRPZ\lab1>git commit -m "c3 on b3"
[b3 915d79a] c3 on b3
1 file changed, 1 insertion(+)
```

Переходимо на мастер та додаємо один файл

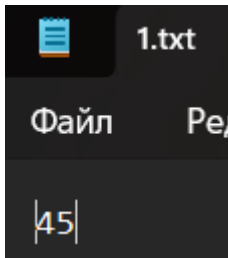


```
C:\Styding\TRPZ\lab1>git switch master
Switched to branch 'master'

C:\Styding\TRPZ\lab1>git add .

C:\Styding\TRPZ\lab1>git commit -m "start2"
[master 450c115] start2
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt
```

Переходимо на 1 гілку, та редагуємо один файл та *неправильно називаємо коміт*



```
C:\Styding\TRPZ\lab1>git switch b1
Switched to branch 'b1'

C:\Styding\TRPZ\lab1>git add .

C:\Styding\TRPZ\lab1>git commit -m "c2 on b2"
[b1 dbc5737] c2 on b2
1 file changed, 1 insertion(+)
```

Пишемо команду log щоб дізнатися хеш другого коміту мастер

```
C:\Styding\TRPZ\lab1>git log --all --oneline
dbc5737 (HEAD -> b1) c2 on b2
450c115 (master) start2
915d79a (b3) c3 on b3
da5c501 c2 on b3
ad30657 c1 on b3
be8b40e (b2) c2 on b2
784319a c1 on b2
8dceb0f c1 on b1
e75ebf4 (b5, b4) start
```

Після чого робимо черрі пік до мастера

```
C:\Styding\TRPZ\lab1>git cherry-pick 450c115
Auto-merging 1.txt
On branch b1
You are currently cherry-picking commit 450c115.
  (all conflicts fixed: run "git cherry-pick --continue")
  (use "git cherry-pick --skip" to skip this patch)
  (use "git cherry-pick --abort" to cancel the cherry-pick operation)

nothing to commit, working tree clean
The previous cherry-pick is now empty, possibly due to conflict resolution.
If you wish to commit it anyway, use:

    git commit --allow-empty

Otherwise, please use 'git cherry-pick --skip'

C:\Styding\TRPZ\lab1>git cherry-pick --continue
On branch b1
You are currently cherry-picking commit 450c115.
  (all conflicts fixed: run "git cherry-pick --continue")
  (use "git cherry-pick --skip" to skip this patch)
  (use "git cherry-pick --abort" to cancel the cherry-pick operation)

nothing to commit, working tree clean
The previous cherry-pick is now empty, possibly due to conflict resolution.
If you wish to commit it anyway, use:

    git commit --allow-empty

Otherwise, please use 'git cherry-pick --skip'

C:\Styding\TRPZ\lab1>git commit --allow-empty
[b1 288d384] start2
Date: Fri Sep 27 11:22:08 2024 +0300
```

Переходимо на 2 гілку та робимо ребейс до мастера

```
C:\Styding\TRPZ\lab1>git switch b2
Switched to branch 'b2'

C:\Styding\TRPZ\lab1>git rebase master
Successfully rebased and updated refs/heads/b2.
```

Переходимо на 3 гілку та робимо мердж до мастера

```
C:\Styding\TRPZ\lab1>git merge master
Merge made by the 'ort' strategy.
 1.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 1.txt
```

В кінці отримуємо таку структуру репозиторія:


```
C:\Styding\TRPZ\lab1>git log --graph --oneline --all --decorate
* 98c5a54 (HEAD -> b3) Merge branch 'master' into b3
|
* 915d79a c3 on b3
* da5c501 c2 on b3
* ad30657 c1 on b3
|
| * 68aa7eb (b2) c2 on b2
| * 21b0eaf c1 on b2
|/
* 450c115 (master) start2
|/
* 288d384 (b1) start2
* dbc5737 c2 on b2
* 8dceb0f c1 on b1
|/
* e75ebf4 (b5, b4) start
```

Висновок: У ході виконання цієї лабораторної роботи я ознайомився з системою керування версіями Git. Я вивчив основні команди для роботи з Git: ініціалізацію репозиторію, додавання файлів до системи, створення комітів, перенесення їх між гілками, злиття гілок та багато іншого. Крім того, я застосував ці команди на практиці та закріпив отримані знання.