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: «Використання основних команд Git»

Виконав:

студент групи ІА-23
Пожар Д. Ю.

Перевірив:

Мягкий М.Ю.

Київ 2024

Мета: У межах даної лабораторної роботи передбачено освоїти використання локальної системи контролю версій Git.

1. Що таке система контролю версій (СКВ)

Система контролю версій (СКВ) — це програмне забезпечення, яке дозволяє відстежувати зміни у файлах та координувати роботу над проєктом між кількома розробниками. Основне завдання СКВ — зберігати історію змін, підтримувати різні версії файлів і забезпечувати можливість повернутися до попередніх версій, якщо це необхідно.

2. Відмінності між розподіленою та централізованою СКВ

Централізована СКВ підходить для невеликих команд з простим робочим процесом, де є необхідність контролювати доступ до змін через єдиний сервер. Розподілена СКВ забезпечує більшу гнучкість, надійність та незалежність, особливо для великих проєктів з багатьма учасниками, де важлива автономна робота та можливість працювати офлайн.

3. Різниця між stage та commit в Git

Stage (індекс): це підготовка змін для коміту. Файли додаються в індекс за допомогою команди git add, але ще не збережені в історії.

Commit: це фактичне збереження змін в історію репозиторію. Після виконання git commit зміни, що були додані в індекс, фіксуються у вигляді нового коміту з коментарем.

4. Як створити гілку в Git

Для створення гілок є наступні команди **git branch <name>** — створює нову гілку, не перемикаючи на неї; **git checkout -b <name>**

— створює нову гілку і перемикає на неї (старіший спосіб); **git switch -c <name>** — створює нову гілку і перемикає на неї (сучасний спосіб).

5. Як створити або скопіювати репозиторій Git з віддаленого серверу

git clone [url]

6. Що таке конфлікт злиття, як створити конфлікт, як вирішити конфлікт?

Конфлікт злиття виникає, коли Git не може автоматично об'єднати зміни з двох різних гілок, оскільки ці зміни конфліктують. Найчастіше це трапляється, коли два або більше розробників редагують один і той самий рядок в одному файлі, або коли один видаляє файл, який інший модифікував.

Git сигналізує про конфлікт, коли під час виконання команди git merge або git rebase не може об'єднати зміни без ручного втручання.

Вирішити конфлікт: Вручну видалити конфліктні позначки (<<<<<<<, =======, >>>>>>>) і обрати, які зміни залишити. Можна залишити одне із двох, об'єднати обидва варіанти або внести власні корективи. **Зберегти файл і додати його до індексу:** Після вирішення конфлікту потрібно зберегти файл і додайте його в індекс

Завершити злиття: Завершити процесс –continue

7. **Ситуації використання команд: merge, rebase, cherry-pick** git merge: використовується для об'єднання двох гілок із збереженням повної історії. git rebase: використовується для чистої, лінійної

історії шляхом переписування комітів поверх іншої гілки. git cherry-pick: використовується для вибіркового копіювання окремих комітів з однієї гілки в іншу.

8. **Як переглянути історію змін Git репозиторію в консолі?** git log
9. **Як створити гілку в Git не використовуючи команду git branch** **git checkout -b <name>** — створює нову гілку і перемикає на неї (старіший спосіб); **git switch -c <name>** — створює нову гілку і перемикає на неї (сучасний спосіб).
10. **Як підготувати всі зміни в поточній папці до коміту?**

Переглянути статус, занести в індекс всі файли які повинні бути в комміті.

11. **Як підготувати всі зміни в дочірній папці до коміту?**

Перевірити статус репозиторію (git status).

Додати зміни в дочірній папці (git add my-subfolder/).

12. **Як переглянути перелік наявних гілок в репозиторії?**

Використати git branch для перегляду локальних гілок.

Використати git branch -a для перегляду всіх гілок (локальних та віддалених).
Використати git branch -r для перегляду тільки віддалених гілок.

13. **Як видалити гілку?**

Для локальної гілки використовуйте git branch -d <назва_гілки> для видалення з перевіркою злиття або git branch -D <назва_гілки> для примусового видалення.

Для віддаленої гілки використовуйте git push <remote> --delete <назва_гілки>.

Хід роботи

Крок 1: Для початку роботи потрібно проініціалізувати новий репозиторій Git.

```
David@Dcomp MINGW64 /d/My/David/KPI/lab1
$ git init
Initialized empty Git repository in D:/My/David/KPI/lab1/.git/
```

Крок 2: Створити перший пустий комміт

```
David@Dcomp MINGW64 /d/My/David/KPI/lab1 (main)
$ git commit -m "first commit" --allow-empty
[main (root-commit) b844c36] first commit
```

Крок 3: Створити три гілки трьома різними способами

```
David@Dcomp MINGW64 /d/My/David/KPI/lab1 (main)
$ git branch br1

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (main)
$ git checkout -b br2
Switched to a new branch 'br2'

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br2)
$ git switch main
Switched to branch 'main'

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (main)
$ git switch -c br3
Switched to a new branch 'br3'

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br3)
$ git switch main
Switched to branch 'main'

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (main)
$ git branch
  br1
  br2
  br3
* main
```

Крок 4: Створи у кожній гілці файл main.txt з різним текстом

```
David@Dcomp MINGW64 /d/My/David/KPI/lab1 (main)
$ echo "main" >> main.txt

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (main)
$ git add .
warning: in the working copy of 'main.txt', LF will be replaced by CRLF the next
 time Git touches it

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (main)
$ git commit -m "main commit"
[main e9d2589] main commit
 1 file changed, 1 insertion(+)
 create mode 100644 main.txt

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (main)
$ git checkout br1
Switched to branch 'br1'

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br1)
$ echo "br1" >> main.txt

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br1)
$ git add .
warning: in the working copy of 'main.txt', LF will be replaced by CRLF the next
 time Git touches it

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br1)
$ git commit -m "br1 to main"
[br1 a056b93] br1 to main
 1 file changed, 1 insertion(+)
 create mode 100644 main.txt

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br1)
$ git switch br2
Switched to branch 'br2'

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br2)
$ echo "br2" >> main.txt

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br2)
$ git add .
warning: in the working copy of 'main.txt', LF will be replaced by CRLF the next
 time Git touches it

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br2)
$ git commit -m "br2 to main"
[br2 87ca660] br2 to main
 1 file changed, 1 insertion(+)
 create mode 100644 main.txt

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br2)
$ git checkout br3
Switched to branch 'br3'

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br3)
$ echo "br3" >> main.txt
```

```
David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br3)
$ git add .
warning: in the working copy of 'main.txt', LF will be replaced by CRLF the next
 time Git touches it

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br3)
$ git commit -m "br3 to main"
[br3 d772b0a] br3 to main
 1 file changed, 1 insertion(+)
 create mode 100644 main.txt
```

Крок 5: Оновити гілки 1,2,3 до версії main за допомогою 3-х різних команд(git merge, git rebase, git cherry-pick). При вирішенні конфліктів зберігався повний текст з обох гілок.

```
David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br1)
$ git merge main
Auto-merging main.txt
CONFLICT (add/add): Merge conflict in main.txt
Automatic merge failed; fix conflicts and then commit the result.

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br1|MERGING)
$

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br1|MERGING)
$ git add .
```

```
David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br1|MERGING)
$ git commit
[br1 5059f42] Merge branch 'main' into br1

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br1)
$ git switch br2
error: Your local changes to the following files would be overwritten by checkou
t:
        main.txt
Please commit your changes or stash them before you switch branches.
Aborting

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br1)
$ git status
On branch br1
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   main.txt

no changes added to commit (use "git add" and/or "git commit -a")

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br1)
$ git add .

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br1)
$ git commit
Aborting commit due to empty commit message.

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br1)
$ git commit -m "br1 to main commit"
[br1 1050fbb] br1 to main commit
 1 file changed, 1 insertion(+), 1 deletion(-)

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br1)
$ git switch br2
Switched to branch 'br2'

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br2)
$ git rebase main
Auto-merging main.txt
CONFLICT (add/add): Merge conflict in main.txt
error: could not apply 87ca660... br2 to main
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 87ca660... br2 to main

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br2|REBASE 1/1)
$ git add .

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br2|REBASE 1/1)
$ git rebase --continue
[detached HEAD cfe8358] br2 to main
 1 file changed, 1 insertion(+), 1 deletion(-)
Successfully rebased and updated refs/heads/br2.
```

```
David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br2)
$ git switch br3
Switched to branch 'br3'

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br3)
$ git switch main
Switched to branch 'main'

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (main)
$ git log
commit e9d2589a3fd0d406c93b5cdbb160c485821d030a (HEAD -> main)
Author: david.pozhar@gmail.com <david.pozhar@gmail.com>
Date:   Thu Dec 12 12:06:02 2024 +0100

    main commit

commit b844c36ae0420d80fd21fceab528858d3bb9d97e
Author: david.pozhar@gmail.com <david.pozhar@gmail.com>
Date:   Thu Dec 12 12:02:27 2024 +0100

    first commit

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (main)
$ git switch br3
Switched to branch 'br3'

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br3)
$ git cherry-pick e9d2589a3fd0d406c93b5cdbb160c485821d030a
Auto-merging main.txt
CONFLICT (add/add): Merge conflict in main.txt
error: could not apply e9d2589... main commit
hint: After resolving the conflicts, mark them with
hint: "git add/rm <pathspec>", 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".

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br3|CHERRY-PICKING)
$ git add .

David@Dcomp MINGW64 /d/My/David/KPI/lab1 (br3|CHERRY-PICKING)
$ git cherry-pick --continue
[br3 b0bec39] main commit
 Date: Thu Dec 12 12:06:02 2024 +0100
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Був отриманий результат:

Висновок

У ході виконання лабораторної роботи ми ознайомилися з основними командами Git для створення і роботи з гілками (git branch, git checkout, git switch) та операціями інтеграції змін (merge, rebase, cherry-pick).