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: «Системи контролю версій. Git»

Виконала:

Студентка групи ІА-24

Орловська А.В.

Перевірив:

Мягкий М.Ю.

Київ-2024

Мета: У межах даної лабораторної роботи передбачено ознайомитися з основними командами системи контролю версій Git.

Теоретичні відомості

Git: Система Управління Версіями

Git — це розподілена система контролю версій, розроблена для відстеження змін у файлах і координування роботи над проектами, особливо програмним забезпеченням. Вона дозволяє кільком розробникам одночасно працювати над одним проектом, зберігаючи при цьому всю історію змін. Нижче наведено декілька основних команд Git:

---

1. git init

Опис: Ініціалізує новий репозиторій Git у поточній директорії.
Деталі:

- Створює підкаталог `.git`, який міститиме всю метадані та історію версій.
- Використовується для початку нового проекту, щоб Git міг відстежувати зміни.

---

2. git clone <url>

Опис: Клонує існуючий репозиторій з віддаленого сервера на ваш комп'ютер.
Деталі:

- Дозволяє працювати з проектами, які вже існують.

---

3. git add

Опис: Додає зміни з файлів до індексу (staging area).
Деталі:

- Команда `git add <file>` додає вказаний файл.
- `git add .` додає всі зміни у всіх файлах в поточному каталозі.
- Важливо використовувати цю команду перед комітом, щоб відзначити, які зміни потрібно зафіксувати.

---

4. git commit

Опис: Зберігає зміни, які були додані до індексу, з описом.
Деталі:

- `git commit -m "<message>"` зберігає зміни з коротким описом.
- Для створення пустого коміту можна використовувати `git commit --allow-empty -m "Empty commit"`.

5. git status

Опис: Показує статус файлів у репозиторії.
Деталі:

- Відображає, які файли були змінені, які готові до коміту, а які не відстежуються.
- Допомагає відстежувати стан репозиторію перед комітом.

6. git log

Опис: Виводить історію комітів в репозиторії.
Деталі:

- Показує список комітів з інформацією, такою як автор, дата та повідомлення.
- Можна використовувати `git log --oneline` для компактного виводу.

7. git branch

Опис: Показує список гілок у репозиторії.
Деталі:

- Команда `git branch <branch-name>` створює нову гілку.
- Використовуйте `git branch -d <branch-name>` для видалення локальної гілки.

8. git checkout

Опис: Перемикається на зазначену гілку або відновлює файли до певного стану.
Деталі:

- `git checkout <branch>` — перехід на вказану гілку.
- `git checkout <file>` — відновлення файлу з останнього коміту.

9. git switch

Опис: Альтернативна команда для переключення між гілками.
Деталі:

- `git switch <branch>` — зручний спосіб перемикатися на іншу гілку без зміни файлів.

10. git merge

Опис: Об'єднує зміни з однієї гілки в іншу.
Деталі:

- Зазвичай використовується для інтеграції змін з гілки `feature` у основну гілку `main`.
- Під час об'єднання можуть виникати конфлікти, які потрібно буде вирішити.

---

11. git rebase

Опис: Переміщує або об'єднує зміни з однієї гілки на іншу.
Деталі:

- Використовується для "переписування" історії, щоб зробити її більш лінійною.
- Зручно використовувати, коли потрібно інтегрувати зміни з основної гілки в вашу поточну.

---

12. git cherry-pick

Опис: Додає конкретний коміт з однієї гілки до поточної гілки.
Деталі:

- Корисно, коли ви хочете перенести певні зміни без об'єднання всіх змін.
- Використовуйте `git cherry-pick <commit-hash>` для вибору коміту.

---

13. git reset

Опис: Скасовує зміни, повертаючи файли до певного стану.
Деталі:

- `git reset <commit>` повертає гілку до вказаного коміту.
- `git reset --hard` видаляє всі зміни в робочому каталозі без можливості відновлення.

---

14. git remove

Опис: Видаляє файли з робочого каталогу та з індексу.
Деталі:

- `git rm <file>` видаляє вказаний файл з репозиторію.
- Після виконання цієї команди потрібно зробити коміт, щоб зафіксувати зміни.

---

15. git fetch

Опис: Завантажує зміни з віддаленого репозиторію без їх інтеграції у вашу локальну гілку.
Деталі:

- `git fetch <remote>` завантажує всі нові коміти з віддаленого репозиторію, але не об'єднує їх з вашою поточною гілкою.
- Використовується для перевірки, що нового з'явилося в віддаленому репозиторії перед тим, як об'єднати зміни з локальною копією.
- Дозволяє переглядати зміни в інших гілках, не змінюючи свою робочу копію, зокрема, за допомогою `git log <remote>/<branch>` для перегляду історії комітів віддаленої гілки.

Хід роботи

1. Ініціалізація порожнього Git-репозиторію та створення порожнього коміту:

```
Microsoft Windows [Version 10.0.22631.4317]
(c) Корпорація Майкрософт. Усі права захищені.

C:\Users\Anna>D:

D:\>cd Навчання 3 курс

D:\Навчання 3 курс>mkdir трпз

D:\Навчання 3 курс>cd трпз

D:\Навчання 3 курс\трпз>git init
Initialized empty Git repository in D:/Навчання 3 курс/трпз/.git/

D:\Навчання 3 курс\трпз>git commit --allow-empty -m "Initial empty commit"
[master (root-commit) e99b085] Initial empty commit
```

2. Створення трьох гілок від master (різними способами):

```
D:\Навчання 3 курс\трпз>git branch branch1

D:\Навчання 3 курс\трпз>git checkout -b branch2
Switched to a new branch 'branch2'

D:\Навчання 3 курс\трпз>git checkout master
Switched to branch 'master'

D:\Навчання 3 курс\трпз>git switch -c branch3
Switched to a new branch 'branch3'

D:\Навчання 3 курс\трпз>git branch
  branch1
  branch2
* branch3
  master
```

3. Створення різної кількості комітів у кожній гілці:

```
D:\Навчання 3 курс\трпз>git checkout branch1
Switched to branch 'branch1'

D:\Навчання 3 курс\трпз>echo "Commit in branch1" > file1.txt

D:\Навчання 3 курс\трпз>git add file1.txt

D:\Навчання 3 курс\трпз>git commit -m "Commit 1 in branch1"
[branch1 d9849a4] Commit 1 in branch1
 1 file changed, 1 insertion(+)
 create mode 100644 file1.txt

D:\Навчання 3 курс\трпз>git checkout branch2
Switched to branch 'branch2'

D:\Навчання 3 курс\трпз>echo "First commit in branch2" > file2.txt

D:\Навчання 3 курс\трпз>git add file2.txt

D:\Навчання 3 курс\трпз>git commit -m "Commit 1 in branch2"
[branch2 f8404a4] Commit 1 in branch2
 1 file changed, 1 insertion(+)
 create mode 100644 file2.txt

D:\Навчання 3 курс\трпз>
D:\Навчання 3 курс\трпз>echo "Second commit in branch2" >> file2.txt

D:\Навчання 3 курс\трпз>git add file2.txt

D:\Навчання 3 курс\трпз>git commit -m "Commit 2 in branch2"
[branch2 27e279d] Commit 2 in branch2
 1 file changed, 1 insertion(+)

D:\Навчання 3 курс\трпз>git checkout branch3
Switched to branch 'branch3'

D:\Навчання 3 курс\трпз>echo "First commit in branch3" > file3.txt

D:\Навчання 3 курс\трпз>git add file3.txt
```

```
D:\Навчання 3 курс\трпз>git commit -m "Commit 1 in branch3"
[branch3 48122d8] Commit 1 in branch3
 1 file changed, 1 insertion(+)
 create mode 100644 file3.txt

D:\Навчання 3 курс\трпз>
D:\Навчання 3 курс\трпз>echo "Second commit in branch3" >> file3.txt

D:\Навчання 3 курс\трпз>git add file3.txt

D:\Навчання 3 курс\трпз>git commit -m "Commit 2 in branch3"
[branch3 b4074d9] Commit 2 in branch3
 1 file changed, 1 insertion(+)

D:\Навчання 3 курс\трпз>
D:\Навчання 3 курс\трпз>echo "Third commit in branch3" >> file3.txt

D:\Навчання 3 курс\трпз>git add file3.txt

D:\Навчання 3 курс\трпз>git commit -m "Commit 3 in branch3"
[branch3 fc84ca7] Commit 3 in branch3
 1 file changed, 1 insertion(+)

D:\Навчання 3 курс\трпз>git log --oneline --all
fc84ca7 (HEAD -> branch3) Commit 3 in branch3
b4074d9 Commit 2 in branch3
48122d8 Commit 1 in branch3
27e279d (branch2) Commit 2 in branch2
f8404a4 Commit 1 in branch2
d9849a4 (branch1) Commit 1 in branch1
e99b085 (master) Initial empty commit
```

4. Створення коміту "FILE" у гілці master:

```
D:\Навчання 3 курс\трпз>git checkout master
Already on 'master'

D:\Навчання 3 курс\трпз>git commit --amend -m "Commit FILE in master"
[master 8e5ff16] Commit FILE in master
 Date: Wed Oct 16 22:22:02 2024 +0300
 1 file changed, 1 insertion(+)
 create mode 100644 masterfile.txt

D:\Навчання 3 курс\трпз>git log --oneline
8e5ff16 (HEAD -> master) Commit FILE in master
e99b085 Initial empty commit
```

5. Операція cherry-pick: перенесення коміту з гілки master в гілку branch1:

```
D:\Навчання 3 курс\трпз>git checkout branch1
Switched to branch 'branch1'

D:\Навчання 3 курс\трпз>git cherry-pick master
[branch1 685d860] Commit FILE in master
 Date: Wed Oct 16 22:22:02 2024 +0300
 1 file changed, 1 insertion(+)
 create mode 100644 masterfile.txt
```

6. Операція merge: злиття змін з гілки master в гілку branch2:

```
D:\Навчання 3 курс\трпз>git checkout branch2
Switched to branch 'branch2'

D:\Навчання 3 курс\трпз>git merge master
Merge made by the 'ort' strategy.
 masterfile.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 masterfile.txt
```

7. Операція rebase: базування гілки branch3 на master:

```
D:\Навчання 3 курс\трпз>git checkout branch3
Switched to branch 'branch3'

D:\Навчання 3 курс\трпз>git rebase master
Successfully rebased and updated refs/heads/branch3.
```

Висновок: У ході виконання даної лабораторної роботи було детально ознайомлено з основними командами системи контролю версій Git. Виконані операції включали ініціалізацію репозиторію, створення гілок трьома різними способами, додавання комітів та застосування операцій `cherry-pick`, `merge` та `rebase`. Команди, такі як `git init`, `git clone`, `git add`, `git commit`, та інші, дозволяють ініціалізувати репозиторії, відстежувати зміни та синхронізувати роботу з віддаленими репозиторіями. Використання Git спрощує процес командної роботи, забезпечуючи надійне збереження та відстеження змін у проекті, а також полегшує роботу над великими проектами, дозволяючи ефективно керувати версіями коду.