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

Технології розроблення програмного забезпечення

Тема роботи: «Опрацювання команд в Git Brash»

Виконав студент групи ІА-23:

Каширов Д. О.

Перевірив:

Мягкий Михайло Юрійович

Київ 2024

Мета роботи:

Робота полягала в тому, щоб навчитися працювати з основними командами Git для ефективного управління версіями проєкту.

Ця робота допомогла набути практичних навичок у використанні Git, що є важливою частиною управління проєктами та спільної розробки коду.

Хід роботи:

1. Створити три нові гілки t1, t2 і t3 від основної гілки.

2. Перейти на гілку t1, створити файл та виконати коміт для додавання файлу.

3. Повторити крок 2 для гілок t2 і t3, створивши у кожній з них новий файл і зробивши коміт для його додавання.

4. Видалити файли в кожній з гілок t1, t2 та t3 окремими комітами.

5. Видалити всі коміти, пов'язані з файлами, повернувшись до попереднього стану гілок.

6. Налаштувати конфігурації Git, додавши ім'я користувача та електронну пошту.

1. **Спочатку перевіремо скільки гілок в мене є на даний момент.**

```
Dima@DESKTOP-5M6HFM1 MINGW64 ~ (main)
$ git branch
* main
```

2. **Створення гілок t1, t2, t3**

Відкриємо термінал у моїй директорії проекту.

Та виконуємо команду для створення трьох нових гілок:

```
Dima@DESKTOP-5M6HFM1 MINGW64 ~ (main)
$ git branch t1

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (main)
$ git branch t2

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (main)
$ git branch t3

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (main)
$ git branch
* main
  t1
  t2
  t3
```

3. **Перехід на кожну гілку та додавання файлів**

Переходимо на першу гілку **t1:**

```
Dima@DESKTOP-5M6HFM1 MINGW64 ~ (main)
$ git checkout t1
A       .gitmodules
A       Amphion
Switched to branch 't1'

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t1)
$
```

4. **Додамо файли до гілки. Додамо ще коментарі до файлів,щоб було зрозуміло. Припустимо, що ми створюємо файл file1.txt:**

```
"Це файл 1" "Додано file1.txt до t1"
```

```
Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t1)
$ echo "Це файл 1" > file1.txt

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t1)
$ git add file1.txt

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t1)
$ git commit -m "Додано file1.txt до t1"
[t1 4787130] Додано file1.txt до t1
 3 files changed, 5 insertions(+), 1 deletion(-)
 create mode 100644 .gitmodules
 create mode 160000 Amphion
```

5. **Повторемо ці кроки для гілок t2 та t3 з новими файлами:**

Гілка t2:

```
Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t1)
$ git checkout t2
warning: unable to rmdir 'Amphion': Directory not empty
Switched to branch 't2'

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t2)
$ echo "Це файл 2" > file2.txt

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t2)
$ git add file2.txt

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t2)
$ git commit -m "Додано file2.txt до t2"
[t2 50ab019] Додано file2.txt до t2
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Гілка t3:

```
Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t2)
$ git checkout t3
Switched to branch 't3'

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t3)
$ echo "Це файл 3" > file3.txt

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t3)
$ git add file3.txt

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t3)
$ git commit -m "Додано file3.txt до t3"
[t3 ae96412] Додано file3.txt до t3
 1 file changed, 1 insertion(+)
 create mode 100644 file3.txt
```

6. Видалення файлів та комітів

Переходимо на кожну гілку та видаляємо файли:

Для t1:

```
Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t3)
$ git checkout t1
Switched to branch 't1'

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t1)
$ rm file1.txt

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t1)
$ git commit -m "Видалено file1.txt"
On branch t1
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    file1.txt
```

Для t2:

```
Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t1)
$ git checkout t2
warning: unable to rmdir 'Amphion': Directory not empty
Switched to branch 't2'

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t2)
$ rm file2.txt

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t2)
$ git commit -m "Видалено file2.txt"
On branch t2
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    file2.txt
```

Для t3:

```
Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t2)
$ git checkout t3
Switched to branch 't3'

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t3)
$ rm file3.txt

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t3)
$ git commit -m "Видалено file3.txt"
On branch t3
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    file3.txt
```

7. **Тепер, щоб видалити всі коміти, пов'язані з файлами, ми можемо використовувати git rebase або git reset. Найпростіший спосіб - це використовувати git reset:**

Для t1:

```
Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t3)
$ git checkout t1
Switched to branch 't1'

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t1)
$ git reset --hard HEAD~1
warning: unable to rmdir 'Amphion': Directory not empty
HEAD is now at 6da9a4f file2.txt
```

Для t2:

```
Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t1)
$ git checkout t2
Switched to branch 't2'

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t2)
$ git reset --hard HEAD~1
HEAD is now at 6da9a4f file2.txt
```

Для t3:

```
Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t2)
$ git checkout t3
Switched to branch 't3'

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t3)
$ git reset --hard HEAD~1
HEAD is now at 6da9a4f file2.txt
```

8. **Налаштування git config**

Тепер налаштуємо мої конфігурації Git. Наприклад я можу налаштувати своє ім’я та електронну пошту:

```
Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t3)
$ git config --global user.name "Dima"

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t3)
$ git config --global user.email "dimakashiroff@gmail.com"

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t3)
$ git config user.name
Dima

Dima@DESKTOP-5M6HFM1 MINGW64 ~ (t3)
$ git config user.email
dimakashiroff@gmail.com
```

Висновок:

У ході роботи було успішно виконано створення та управління гілками в Git. Створено три нові гілки, в кожній з яких додано файл та закомічено зміни. Після цього файли було видалено, а відповідні коміти анульовано для повернення гілок до попереднього стану. Завершальним етапом стало налаштування Git, де було вказано ім’я користувача та електронну пошту для глобальних налаштувань. Проведені дії продемонстрували основні операції з гілками, комітами та конфігурацією Git, що є корисними для ефективної роботи з системою контролю версій.