



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 1
з дисципліни «Технології розроблення програмного
забезпечення»
Тема: «Git»

Виконала

студент групи IA-32

Самойленко С. Д.

Перевірив

Мягкий М.Ю.

Тема: Системи контроля версій. Розподілена система контролю версій «Git».

Мета: Навчитися виконувати основні операції в роботі з децентралізованими системами контролю версій на прикладі роботи з сучасною системою Git.

Теоретичні відомості

Система управління версіями (від англ. Version Control System або Source Control System) – програмне забезпечення яке призначено допомогти команді розробників керувати змінами в вихідному коді під час роботи [1]. Система керування версіями дозволяє додавати зміни в файлах в репозиторій і таким чином після кожної фіксації змін мани нову ревізію файлів. Це дозволяє повернутися до попередніх версій коду для аналізу внесених змін або пошуку, які зміни привели до появи помилки. Таким чином можна знайти хто, коли і які зміни зробив в коді, а також чому ці зміни були зроблені

Гіт є системою розподіленого контролю версій, коли кожен розробник має власний репозиторій, куди він вносить зміни. Далі система гіт синхронізує репозиторії із центральним репозиторієм. Це дозволяє проводити роботу незалежно від центрального репозиторію (на відміну від SVN, коли версіонування передбачало наявність зв'язку з центральним сервером), перекладає складності ведення гілок та склеювання змін більше на плечі системи, ніж розробників та ін. Зміни зберігаються у вигляді наборів змін (changeset), що отримує унікальний ідентифікатор (хеш-сума на основі самих змін).

Основна ідея Git, як і будь-якої іншої розподіленої системи контроля версій – кожен розробник має власний репозиторій, куди складаються зміни (версії) файлів, та синхронізація між розробниками виконується за допомогою синхронізації репозиторіїв. Відповідно, є ряд основних команд для роботи:

1. Клонувати репозиторій (`git clone`) – отримати копію репозиторію на локальну машину для подальшої роботи з ним;
2. Синхронізація репозиторіїв (`git fetch` або `git pull`) – отримання змін із віддаленого (вихідного, центрального, або будь-якого іншого такого ж) репозиторію;
3. Фіксація змін в репозиторій (`git commit`) – фіксація виконаних змін в програмному коді в локальній репозиторій розробника;
4. Синхронізація репозиторіїв (`git push`) – переслати зміни – `push` – передача власних змін до віддаленого репозиторію – Записати зміни – `commit` – створення нової версії;
5. Оновитись до версії – `update` – оновитись до певної версії, що є у репозиторії.
6. Об'єднання гілок (`git merge`) – об'єднання вказаною гілки в поточну (часто ще називається «злиттям»).

Хід роботи

1. Створюємо директорію та переходимо в неї.

```
stanislav@MacBook-Air-Stanislav ~ % mkdir trpz
stanislav@MacBook-Air-Stanislav ~ % cd trpz
stanislav@MacBook-Air-Stanislav trpz %
```

2. Створюємо локальний репозиторій.

```
stanislav@MacBook-Air-Stanislav trpz % git init
Initialized empty Git repository in /Users/stanislav/trpz/.git/
```

3. Робимо перший пустий запис у Git.

```
stanislav@MacBook-Air-Stanislav trpz % git commit --allow-empty -m "First empty
commit"
[main (root-commit) 1c6d39a] First empty commit
```

4. Створюємо 2 гілки різними способами.

```
stanislav@MacBook-Air-Stanislav trpz % git branch b1
stanislav@MacBook-Air-Stanislav trpz % git checkout -b b2
Switched to a new branch 'b2'
```

5. Додаємо два файли у директорію.

```
stanislav@MacBook-Air-Stanislav trpz % echo "text" > f1.txt
stanislav@MacBook-Air-Stanislav trpz % echo "text" > f2.txt
stanislav@MacBook-Air-Stanislav trpz % ls
f1.txt  f2.txt
```

6. Зафіксуємо зміни, додавши файл f1.txt в гілку b1, а f2.txt в гілку b2.

```
stanislav@MacBook-Air-Stanislav trpz % git checkout b1
Switched to branch 'b1'
stanislav@MacBook-Air-Stanislav trpz % git add f1.txt
stanislav@MacBook-Air-Stanislav trpz % git commit -m "Add f1.txt to b1"
[b1 8118a0f] Add f1.txt to b1
 1 file changed, 1 insertion(+)
  create mode 100644 f1.txt
stanislav@MacBook-Air-Stanislav trpz %
```

7. Переглядаємо історію змін у вигляді графа.

```
stanislav@MacBook-Air-Stanislav trpz % git log --oneline --graph --all
* 8118a0f (HEAD -> b1) Add f1.txt to b1
* 1c6d39a (main, b2) First empty commit
```

8. Додаємо файл f2.txt в гілку b1 та фіксуємо зміни.

```
stanislav@MacBook-Air-Stanislav trpz % git checkout b1
Already on 'b1'
stanislav@MacBook-Air-Stanislav trpz % git add f2.txt
stanislav@MacBook-Air-Stanislav trpz % git commit -m "Add f2.txt tot b1"
[b1 e8b1adf] Add f2.txt tot b1
 1 file changed, 1 insertion(+)
  create mode 100644 f2.txt
```

9. Намагаємося об'єднати гілки b1 та b2.

```
stanislav@MacBook-Air-Stanislav trpz % git checkout b1
Already on 'b1'
stanislav@MacBook-Air-Stanislav trpz % git merge b2
Already up to date.
```

Отримуємо конфлікт через те, що один і той самий файл змінений у двох гілках, та вирішуємо його вручну.

10. Продовжуємо злиття гілок b1 та b2.

```
stanislav@MacBook-Air-Stanislav trpz % git add f2.txt
stanislav@MacBook-Air-Stanislav trpz % git commit
On branch b1
nothing to commit, working tree clean
```

11. Переглядаємо історію змін

```
stanislav@MacBook-Air-Stanislav trpz % git log --oneline --graph --all
* e8b1adf (HEAD -> b1) Add f2.txt tot b1
* 8118a0f Add f1.txt to b1
* 1c6d39a (main, b2) First empty commit
```

Висновок: Під час виконання цієї роботи ми навчилися виконувати основні операції в роботі з git, зокрема створювати гілки, виконувати коміти, об'єднувати гілки, а також вирішувати конфлікти.