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
«Системи контролю версій. Git.»

Виконав
студент групи ІА–22:
Клочков І. Ф.

Київ 2024

Теоретичні відомості

Git — розподілена система контролю версій. Проект створив Лінус Торвальдс для керування розробкою ядра Linux, а сьогодні підтримується Джуніо Хамано (англ. Junio C. Hamano). Git є однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні та злитті гілок.

Основні терміни та поняття в Git

Репозиторій (repository) - це основна одиниця зберігання в Git, що є базою даних, де фіксуються всі зміни в проєкті. Репозиторій містить повну історію всіх комітів, включно з повними версіями всіх файлів і директорій проєкту на кожному етапі його розвитку. Це дає змогу будь-якому розробнику в будь-який момент відновити стан будь-якої версії проєкту або відстежити зміни в коді.

Коміт (commit) - це, по суті, «знімок» поточного стану проєкту, що зберігається в історії репозиторію. Кожен коміт містить інформацію про всі зміни у файлах порівняно з попереднім комітом, а також метадані, як-от автор, дата і коментар до коміту. Це дає змогу розробникам розуміти, коли і з якої причини було зроблено зміни.

Гілка (branch) - це паралельна версія сховища, яку використовують для розробки окремих функцій або експериментів, не заважаючи основній робочій версії проєкту (найчастіше «master» або «main»). Гілки дають змогу розробникам працювати над нововведеннями або виправленнями, а потім зливати свої зміни назад у головну гілку після завершення та перевірки.

Які бувають стани в GIT і як їх розуміти

У Git, файли в рамках робочого проєкту можуть перебувати в одному з трьох основних станів: індексований, змінений і зафіксований.

Індексований (staged): Коли файли додаються в індекс, Git визначає їх як підготовлені до коміту. Додавання файлів в індекс відбувається за допомогою команди git add. Цей процес дає змогу розробникам вибірково додавати зміни в наступний коміт, що забезпечує гнучкість в управлінні версіями та допомагає уникати непотрібного включення тимчасових або експериментальних змін. Фактично, індекс слугує проміжною зоною, де розробники можуть організувати

та супроводжувати свої правки перед тим, як остаточно зафіксувати їх в історії репозиторію.

Змінений (modified) стан: Файли в стані «змінені» вказують на те, що їх було відредаговано після останнього коміту, але ще не було додано до індексу. Це первинний стан для будь-яких файлів, які ви змінюєте в робочому каталозі. Git відстежує ці модифікації, які можна переглянути за допомогою команди git status. Перш ніж файли можуть бути зафіксовані, їх необхідно додати до індексу, що дає змогу розробникам ретельно контролювати, які зміни будуть включені до наступного коміту.

Зафіксований (committed) стан: Коли файли зафіксовані, вони зберігаються в локальній базі даних Git, що означає, що зміни безпечно зберігаються у вашому репозиторії. Стан «зафіксований» гарантує, що всі зміни, які ви зробили та підготували, тепер будуть частиною історії сховища і можуть бути відновлені в будь-який момент або передані іншим розробникам через віддалене сховище. Фіксація правок відбувається за допомогою команди git commit, яка також вимагає додавання описового повідомлення, що пояснює суть змін для поліпшення спільної роботи та розуміння історії проєкту.

Основні команди для роботи з Git

Ініціалізація та клонування

- git init – створити новий порожній локальний репозиторій.
- git clone <url> – клонувати віддалений репозиторій на локальну машину.
- git status – показати статус файлів у робочій директорії.
- git diff – показати відмінності між робочою директорією і індексом.
- git add <file> – додати файл до області індексації.
- git add . – додати всі зміни у поточній директорії до області індексації.
- git commit -m "повідомлення" – створити коміт із повідомленням.
- git log – переглянути історію комітів.
- git reset <file> – скасувати індексацію змін файлу.
- git reset --hard – скинути всі зміни до останнього коміту.
- git branch – показати всі гілки в локальному репозиторії.
- git branch <branch> – створити нову гілку.

- git branch -d <branch> – видалити локальну гілку (якщо вона вже злита).
- git branch -D <branch> – примусово видалити гілку, незалежно від її стану.
- git checkout <branch> – переключитися на іншу гілку.
- git checkout -b <branch > – створити нову гілку і одразу переключитися на неї.
- git switch <branch> – переключитися на іншу гілку.
- git merge <branch> – злити вказану гілку в поточну.
- git rebase <branch> – застосувати зміни з вказаної гілки на поточну.
- git fetch <remote> – завантажити всі зміни з віддаленого репозиторію без злиття.
- git pull <remote> – завантажити зміни з віддаленого репозиторію і автоматично злити їх із локальними.
- git push <remote> <branch> – відправити локальні зміни до віддаленого репозиторію.

Хід виконання

1. Ініціалізуємо новий порожній локальний репозиторій.

```
C:\Users\vanya\test>git init
Initialized empty Git repository in C:/Users/vanya/test/.git/

C:\Users\vanya\test>git add 1.txt

C:\Users\vanya\test>git commit -m "add 1.txt"
[main (root-commit) dae005c] add 1.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 1.txt
```

2. Створюємо три гілки різними способами.

```
C:\Users\vanya\test>git branch b1

C:\Users\vanya\test>git checkout -b "b2"
Switched to a new branch 'b2'

C:\Users\vanya\test>git checkout main
Switched to branch 'main'

C:\Users\vanya\test>git switch -c "b3"
Switched to a new branch 'b3'

C:\Users\vanya\test>git branch
  b1
  b2
* b3
  main
```

3. Змінюємо один файл та додаємо новий.

```
C:\Users\vanya\test>git checkout b1
Switched to branch 'b1'

C:\Users\vanya\test>git add 2.txt

C:\Users\vanya\test>git commit -m "add 2.txt"
[b1 923dc40] add 2.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 2.txt

C:\Users\vanya\test>git commit -am "change 1.txt"
[b1 51ba46d] change 1.txt
 1 file changed, 1 insertion(+)

C:\Users\vanya\test>git log
commit 51ba46d8f430bf85a82abdedecb204188fdd7a30 (HEAD -> b1)
Author: ivanchikk <vanyaklochkoff@ukr.net>
Date:   Mon Sep 23 19:09:04 2024 +0300

    change 1.txt

commit 923dc40c5e5a563f0ebe3afe1ac06727f6ec1c9a
Author: ivanchikk <vanyaklochkoff@ukr.net>
Date:   Mon Sep 23 19:08:49 2024 +0300

    add 2.txt

commit dae005c38ec1e6e71204233e47cc18389b802eb7 (main, b3, b2)
Author: ivanchikk <vanyaklochkoff@ukr.net>
Date:   Mon Sep 23 18:56:53 2024 +0300

    add 1.txt
```

4. Змінюємо файл та додаємо папку з двома файлами двома комітами, трьома командами.

```
C:\Users\vanya\test>git checkout b2
Switched to branch 'b2'

C:\Users\vanya\test>git add .

C:\Users\vanya\test>git commit 1.txt -m "change 1.txt"
[b2 6ad8a6f] change 1.txt
 1 file changed, 1 insertion(+)

C:\Users\vanya\test>git commit f1/ -m "add folder with 2 files"
[b2 0cdd019] add folder with 2 files
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 f1/1.txt
 create mode 100644 f1/2.txt

C:\Users\vanya\test>git log
commit 0cdd01935f7680e2b3825a8485f3710ef9f0143d (HEAD -> b2)
Author: ivanchikk <vanyaklochkoff@ukr.net>
Date:   Mon Sep 23 19:26:38 2024 +0300

    add folder with 2 files

commit 6ad8a6ffa90e3cfad99a5cd75ab56462155d673d
Author: ivanchikk <vanyaklochkoff@ukr.net>
Date:   Mon Sep 23 19:24:47 2024 +0300

    change 1.txt

commit dae005c38ec1e6e71204233e47cc18389b802eb7 (main, b3)
Author: ivanchikk <vanyaklochkoff@ukr.net>
Date:   Mon Sep 23 18:56:53 2024 +0300

    add 1.txt
```

5.Мердж b2 у b1.

```
C:\Users\vanya\test>git checkout b1
Switched to branch 'b1'

C:\Users\vanya\test>git merge b2
Auto-merging 1.txt
CONFLICT (content): Merge conflict in 1.txt
Automatic merge failed; fix conflicts and then commit the result.

C:\Users\vanya\test>git add 1.txt

C:\Users\vanya\test>git commit -m "merge b2 in b1"
[b1 85953b0] merge b2 in b1

C:\Users\vanya\test>git log
commit 85953b0fcfd1764fb0dd41c80367bac24b7be0ab (HEAD -> b1)
Merge: 51ba46d 0cdd019
Author: ivanchikk <vanyaklochkoff@ukr.net>
Date:   Mon Sep 23 19:47:36 2024 +0300

    merge b2 in b1

commit 0cdd01935f7680e2b3825a8485f3710ef9f0143d (b2)
Author: ivanchikk <vanyaklochkoff@ukr.net>
Date:   Mon Sep 23 19:26:38 2024 +0300

    add folder with 2 files

commit 6ad8a6ffa90e3cfad99a5cd75ab56462155d673d
Author: ivanchikk <vanyaklochkoff@ukr.net>
Date:   Mon Sep 23 19:24:47 2024 +0300

    change 1.txt

commit 51ba46d8f430bf85a82abdedecb204188fdd7a30
Author: ivanchikk <vanyaklochkoff@ukr.net>
Date:   Mon Sep 23 19:09:04 2024 +0300

    change 1.txt

commit 923dc40c5e5a563f0ebe3afe1ac06727f6ec1c9a
Author: ivanchikk <vanyaklochkoff@ukr.net>
Date:   Mon Sep 23 19:08:49 2024 +0300

    add 2.txt

commit dae005c38ec1e6e71204233e47cc18389b802eb7 (main, b3)
Author: ivanchikk <vanyaklochkoff@ukr.net>
Date:   Mon Sep 23 18:56:53 2024 +0300

    add 1.txt
```

6.Чері-пік у main.

```
C:\Users\vanya\test>git checkout main
Switched to branch 'main'

C:\Users\vanya\test>git commit -am "change 1.txt"
[main c10167e] change 1.txt
 1 file changed, 1 insertion(+)

C:\Users\vanya\test>git cherry-pick 6ad8a6ffa90e3cfad99a5cd75ab56462155d673d
Auto-merging 1.txt
CONFLICT (content): Merge conflict in 1.txt
error: could not apply 6ad8a6f... change 1.txt
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".
hint: Disable this message with "git config advice.mergeConflict false"

C:\Users\vanya\test>git add 1.txt

C:\Users\vanya\test>git commit -m "change 1.txt"
[main 58c5c65] change 1.txt
 Date: Mon Sep 23 19:24:47 2024 +0300
 1 file changed, 1 insertion(+), 1 deletion(-)
```

7. Робимо сквош за допомогою ребейзу.

```
C:\Users\vanya\test>git checkout b2
Switched to branch 'b2'

C:\Users\vanya\test>git log
commit 0cdd01935f7680e2b3825a8485f3710ef9f0143d (HEAD -> b2)
Author: ivanchikk <vanyaklochkoff@ukr.net>
Date:   Mon Sep 23 19:26:38 2024 +0300

    add folder with 2 files

commit 6ad8a6ffa90e3cfad99a5cd75ab56462155d673d
Author: ivanchikk <vanyaklochkoff@ukr.net>
Date:   Mon Sep 23 19:24:47 2024 +0300

    change 1.txt

commit dae005c38ec1e6e71204233e47cc18389b802eb7 (b3)
Author: ivanchikk <vanyaklochkoff@ukr.net>
Date:   Mon Sep 23 18:56:53 2024 +0300

    add 1.txt

C:\Users\vanya\test>git rebase -i HEAD~2
[detached HEAD 1321158] change 1.txt
 Date: Mon Sep 23 19:24:47 2024 +0300
 3 files changed, 1 insertion(+)
 create mode 100644 f1/1.txt
 create mode 100644 f1/2.txt
Successfully rebased and updated refs/heads/b2.

C:\Users\vanya\test>git log
commit 132115842a6b2cb4f97cbb0624f0e349d0d80a39 (HEAD -> b2)
Author: ivanchikk <vanyaklochkoff@ukr.net>
Date:   Mon Sep 23 19:24:47 2024 +0300

    change 1.txt

    add folder with 2 files

commit dae005c38ec1e6e71204233e47cc18389b802eb7 (b3)
Author: ivanchikk <vanyaklochkoff@ukr.net>
Date:   Mon Sep 23 18:56:53 2024 +0300

    add 1.txt
```

```
pick 6ad8a6f change 1.txt
squash 0cdd019 add folder with 2 files
```

Висновок: виконавши цю лабораторну роботу, я навчився користуватись git та виконувати базові дії за допомогою нього.