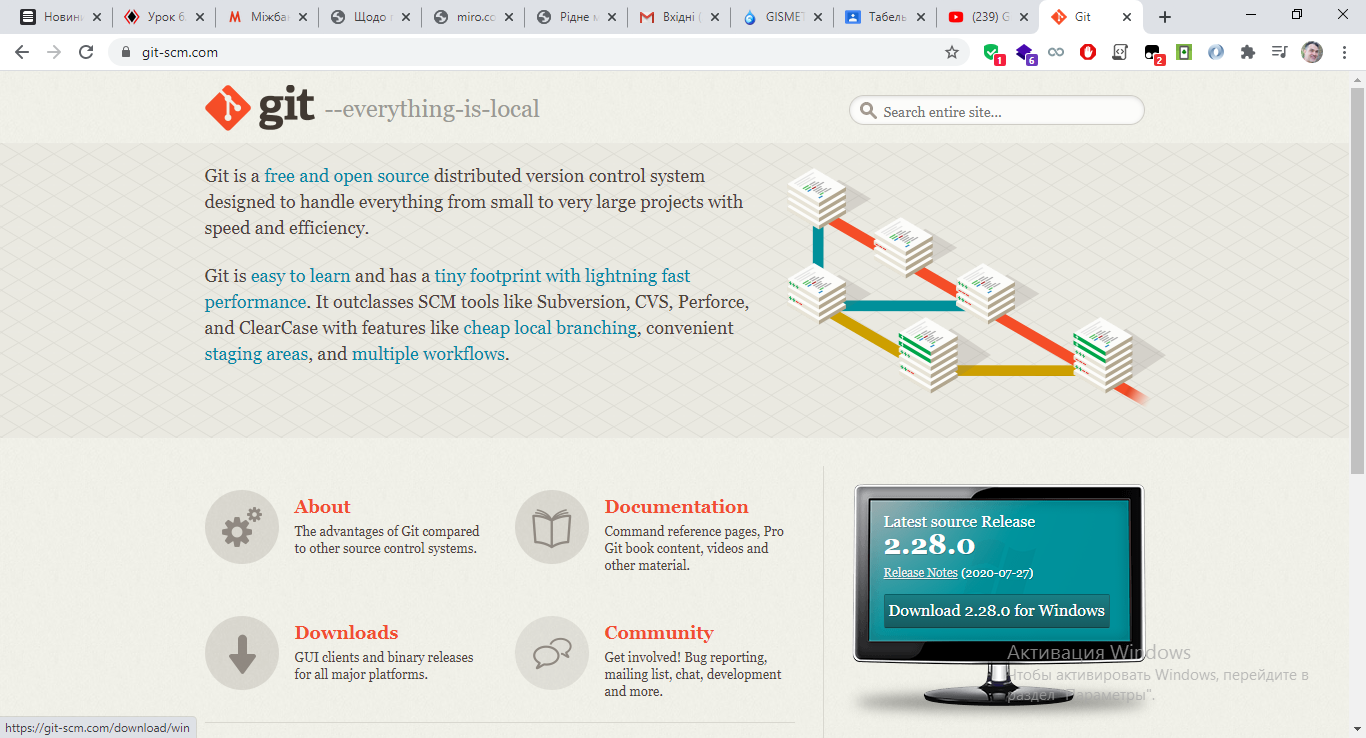
**Git my**

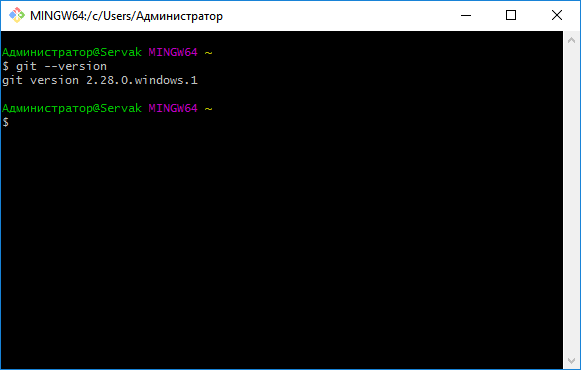
1. **Базова конфігурація.**

Для початку роботи з Git, скачаємо останню версію програми з офіційного сайту <http://git-scm.com/>



та інсталюємо її.

Далі запускаємо GitBash та перевіримо успішність установки



Спершу зробимо глобальну конфігурацію, додавши ім’я користувача (підпис) для внесених змін у віддалений репозиторій (комітів):

$ git config --global user.name "SergiyShchuckiy"

а також електронну пошту

$ git config --global user.email "\*\*\*\*\*\*\*\*\*\*\*@gmail.com"

Далі створюємо папку на диску, де буде зберігатись майбутній проект, наприклад rep1, та ініціалізувати її як Git проект.

Администратор@Servak MINGW64 ~

$ cd D:

Администратор@Servak MINGW64 /d

$ cd Git\_Rep

Администратор@Servak MINGW64 /d/Git\_Rep

$ mkdir rep1

Администратор@Servak MINGW64 /d/Git\_Rep

$ cd rep1

Администратор@Servak MINGW64 /d/Git\_Rep/rep1

$ git init

Initialized empty Git repository in D:/Git\_Rep/rep1/.git/

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

Далі виконаємо локальну конфігурацію. Для цього перейдемо у папку, де буде створено локальний проект, у нашому випадку та ж папка.

$ cd rep1

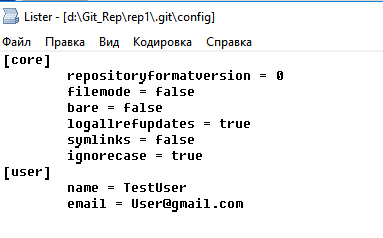
Переглянемо її вміст

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ ls -a

./ ../ .git/

Переглянемо файл config



Локальна конфігурація виконується схоже до глобальної.

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

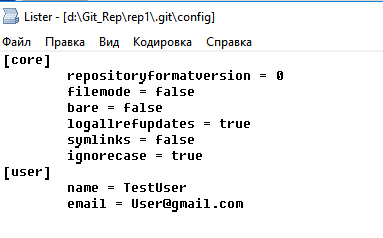
$ git config user.name "TestUser"

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git config user.email "User@gmail.com"

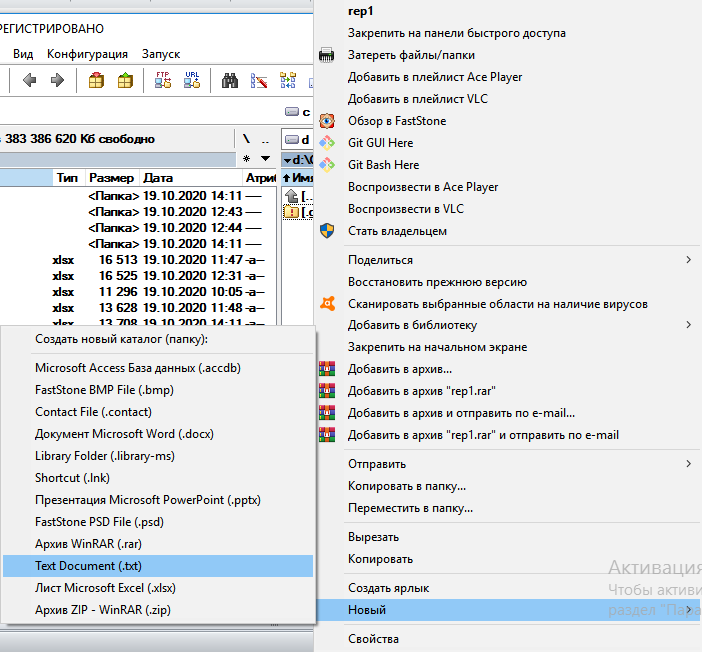
Таким чином поточний проект буде виконаний із локальними налаштуваннями, але усі решта проектів, які будуть створені без задання локальних конфігурацій матимуть глобальні.

Після цього можна переглянути змінений файл config



1. **Перший коміт (git commit).**

Створимо будь-який файл у папці проекту, наприклад, test.txt



І щось там напишемо.

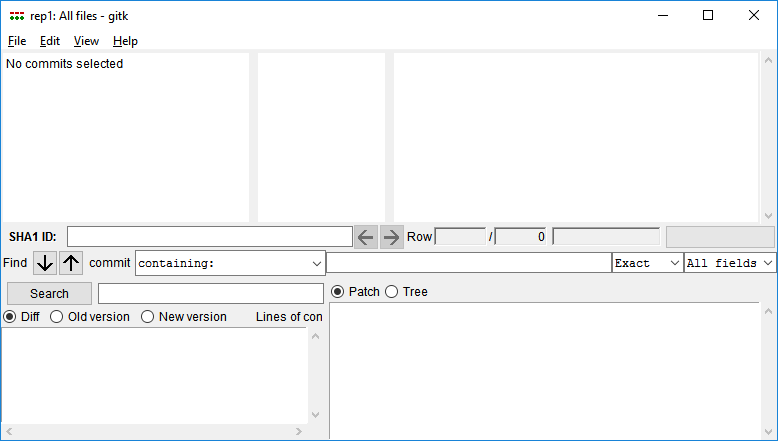
Далі можемо виконувати коміт:

Спочатку виконаємо git add додавши цей файл на коміт кадидат.

$ git add test.txt

Скористаємось ui редактором gitk,щоб бачити усі наші коміти.

$ gitk



Як бачимо комітів поки немає. Створимо його.

$ git commit -m "First commit"

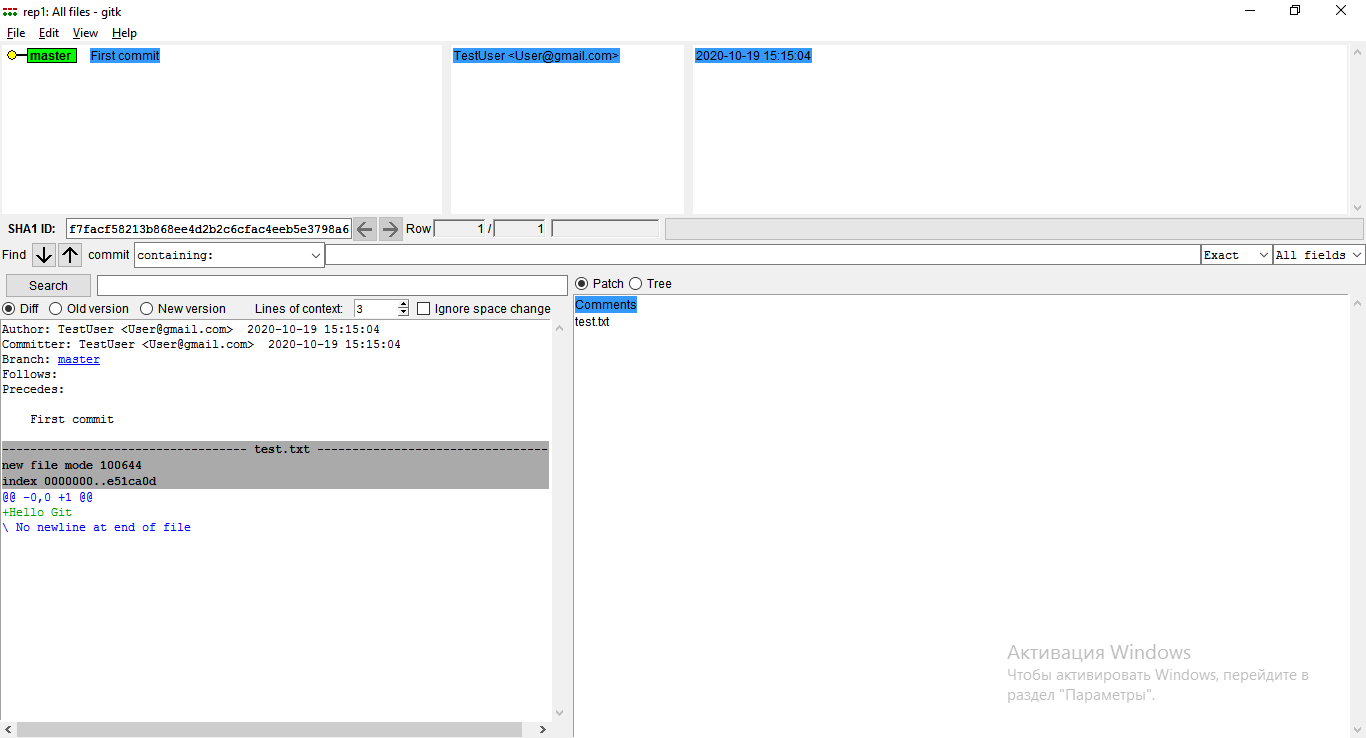
[master (root-commit) f7facf5] First commit

1 file changed, 1 insertion(+)

create mode 100644 test.txt

Прапорець –m додає коментар до каміта.

В gitk побачимо наступні зміни



Внесемо зміни у файл test.txt. та закомітимо його. Але без команди add комміт не виконається, оскільки зміни не були додані у коміт кандидат.

$ git commit -m "second commit"

On branch master

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: test.txt

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

Виправимось

$ git add test.txt

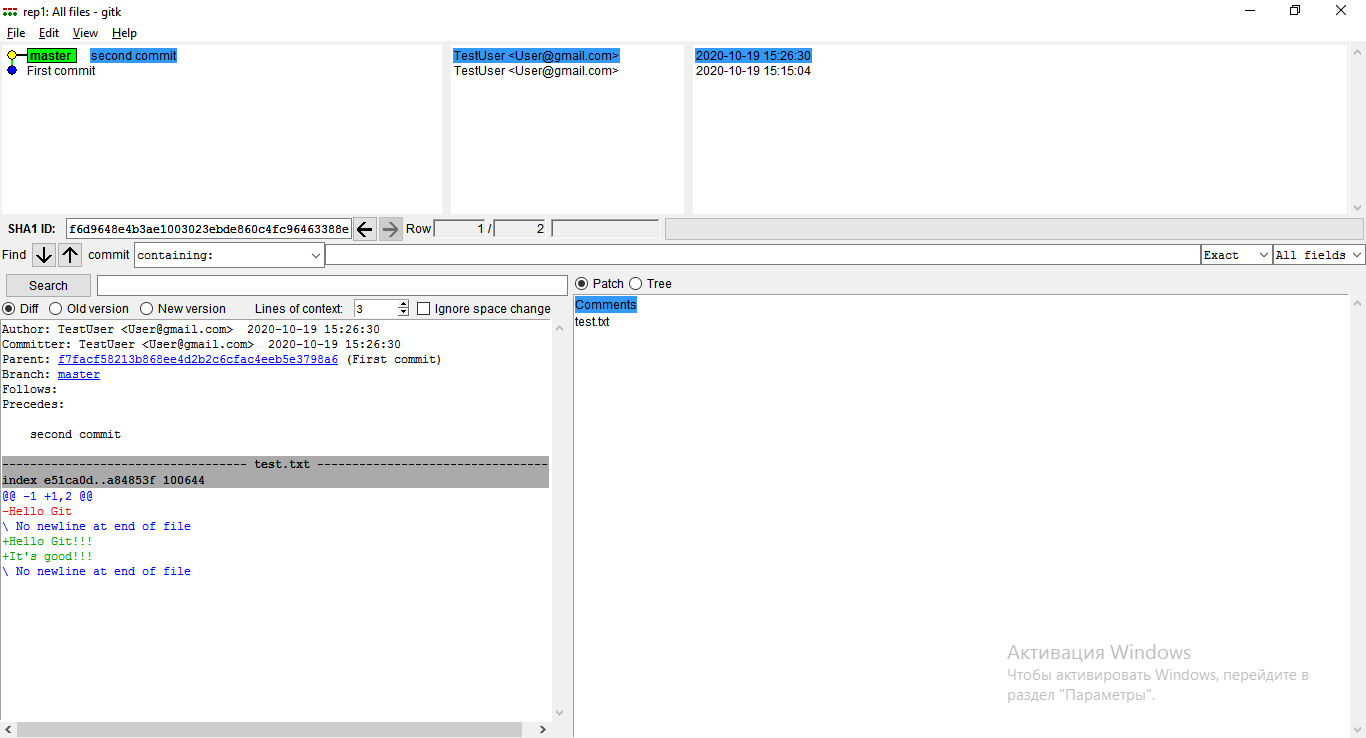
Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git commit -m "second commit"

[master f6d9648] second commit

1 file changed, 2 insertions(+), 1 deletion(-)

В gitk побачимо наступні зміни



1. **Перевірка стану (git status)**

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git status

On branch master

nothing to commit, working tree clean

Як бачимо виводиться поточний стан репозиторія, зокрема виводиться, що ми знаходимося на гілці master, комітів немає і робоче дерево порожнє.

Додамо у тестовий файл новий рядок і виконаємо ще раз git status.

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git status

On branch master

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: test.txt

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

Як бачимо виводиться інформація, що є зміни, і пропонується, що можна зробити (git add (додати зміни) чи git restore (відмінити зміни)).

Додамо зміни у коміт кандидат.

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git add test.txt

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: test.txt

Зміна кольору означає, що змінений файл доданий у коміт кандидат. Закомітимо його.

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git commit -m "Test status"

[master 02545ea] Test status

1 file changed, 2 insertions(+), 1 deletion(-)

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git status

On branch master

nothing to commit, working tree clean

Команда git status досить корисна, щоб переглянути, які файли змінені при коміті, особливо, коли таких змінених файлів багато.

1. **Індексація файлів**

У файлі проекту створимо новий html файл (index.html), та додамо у нього деякі теги.

Далі виконаємо git status.

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git status

On branch master

Untracked files:

(use "git add <file>..." to include in what will be committed)

index.html

nothing added to commit but untracked files present (use "git add" to track)

Як бачимо git повідомляє, що зявився невідстежуваний (неіндексований) файл і git не знає, що з ним робити і рекомендує use "git add" to track

Власне команда git add і додає файл у відстежувані або, іншими словами, додаємо цей файл на індексацію gitу.

Виконаємо це і перевіримо статус.

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git add index.html

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: index.html

Як видно гіт повідомляє, що зміни (доданий файл) були проіндексовані (відстежені), і що гід може слідкувати за його станом. Можна закомітити його.

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git commit -m "New file"

[master 4ee3705] New file

1 file changed, 1 insertion(+)

create mode 100644 index.html

Додамо новий рядок у файл test.txt, а файл index.html перейменуємо на index.txt і перевіримо статус.

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git status

On branch master

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: index.html

modified: test.txt

Untracked files:

(use "git add <file>..." to include in what will be committed)

index.txt

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

Як бачимо гіт перейменування не розуміє, видаляє і створює файл…

Щоб проіндексувати (відстежити, додати їх на коміт кандидат) усі наші зміни можна використати команду add і перелічити усі файли, що були змінені, або ж команду add \*, щоб проіндексувати усі зміни.

Команда ж add . проіндексує змінені файли лише із папки поточного проекту.

Використаємо add \*

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git add \*

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: index.txt

modified: test.txt

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: index.html

Щоб гіт зрозумів перейменування, і про індексував (відстежив) його слід додати окремо:

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git add index.html

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

renamed: index.html -> index.txt

modified: test.txt

Фактично індексація – це додання змінених файлів на відстеження (індексацію) гіту і перевірки, що з ними не так. Після успішної індексації ці файли стають коміт кандидатами, їх далі слід закомітити.

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

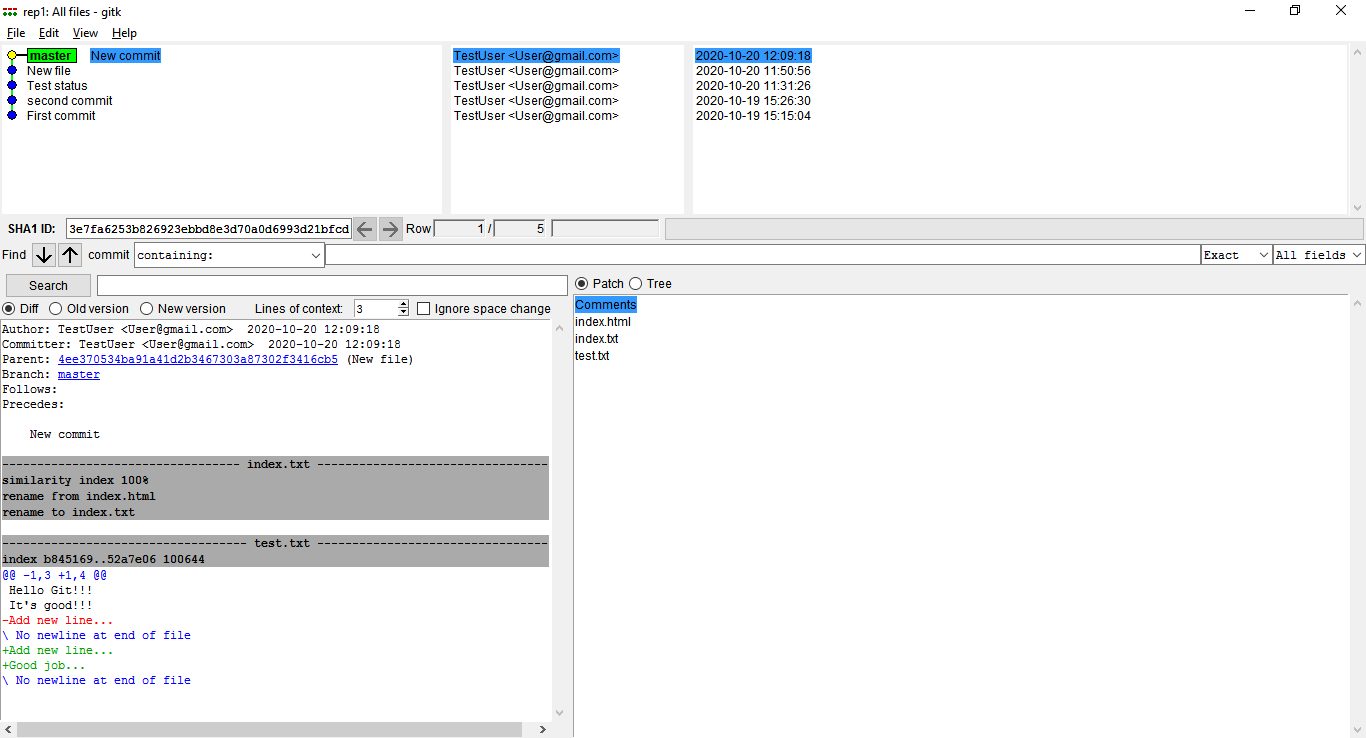
$ git commit -m "New commit"

[master 3e7fa62] New commit

2 files changed, 2 insertions(+), 1 deletion(-)

rename index.html => index.txt (100%)

Тепер gitk покаже усі наші коміти, їх історію…



Як бачимо усі ці зміни індексуються гітом.

Усі ці команди поки працюють із локальним репозиторієм.

1. **Історія комітів (git log).**

Виконаємо команду git log і отримаємо повну історію комітів, що складається з хеш-коду коміта, автора, дати та коментаря до коміта.

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git log

commit 3e7fa6253b826923ebbd8e3d70a0d6993d21bfcd (HEAD -> master)

Author: TestUser <User@gmail.com>

Date: Tue Oct 20 12:09:18 2020 +0300

New commit

commit 4ee370534ba91a41d2b3467303a87302f3416cb5

Author: TestUser <User@gmail.com>

Date: Tue Oct 20 11:50:56 2020 +0300

New file

commit 02545ea92c55725835af0ced04948d82247fbf09

Author: TestUser <User@gmail.com>

Date: Tue Oct 20 11:31:26 2020 +0300

Test status

commit f6d9648e4b3ae1003023ebde860c4fc96463388e

Author: TestUser <User@gmail.com>

Date: Mon Oct 19 15:26:30 2020 +0300

second commit

:...skipping...

commit 3e7fa6253b826923ebbd8e3d70a0d6993d21bfcd (HEAD -> master)

Author: TestUser <User@gmail.com>

Date: Tue Oct 20 12:09:18 2020 +0300

New commit

commit 4ee370534ba91a41d2b3467303a87302f3416cb5

Author: TestUser <User@gmail.com>

Date: Tue Oct 20 11:50:56 2020 +0300

New file

commit 02545ea92c55725835af0ced04948d82247fbf09

Author: TestUser <User@gmail.com>

Date: Tue Oct 20 11:31:26 2020 +0300

Test status

commit f6d9648e4b3ae1003023ebde860c4fc96463388e

Author: TestUser <User@gmail.com>

Date: Mon Oct 19 15:26:30 2020 +0300

second commit

commit f7facf58213b868ee4d2b2c6cfac4eeb5e3798a6

Author: TestUser <User@gmail.com>

Date: Mon Oct 19 15:15:04 2020 +0300

First commit

Зазначимо, що зберігаються саме коміти, а не будь-які зміни…

У команди git log є ряд прапорців. Розглянемо деякі.

$ git log --pretty=oneline

3e7fa6253b826923ebbd8e3d70a0d6993d21bfcd (HEAD -> master) New commit

4ee370534ba91a41d2b3467303a87302f3416cb5 New file

02545ea92c55725835af0ced04948d82247fbf09 Test status

f6d9648e4b3ae1003023ebde860c4fc96463388e second commit

f7facf58213b868ee4d2b2c6cfac4eeb5e3798a6 First commit

Як видно, змінюється форма виведення історії (кожен коміт і інформація про нього виводиться в один рядок)

Можна обмежити кількість рядків, комітів з інформацією…

$ git log --pretty=oneline --max-count=3

3e7fa6253b826923ebbd8e3d70a0d6993d21bfcd (HEAD -> master) New commit

4ee370534ba91a41d2b3467303a87302f3416cb5 New file

02545ea92c55725835af0ced04948d82247fbf09 Test status

Виведеться останні 3 коміта.

$ git log --pretty=oneline виводить обмежену кількість комітів. Якщо їх багато, то щоб вивести усі використовують прапорець --all:

$ git log --pretty=oneline --all

3e7fa6253b826923ebbd8e3d70a0d6993d21bfcd (HEAD -> master) New commit

4ee370534ba91a41d2b3467303a87302f3416cb5 New file

02545ea92c55725835af0ced04948d82247fbf09 Test status

f6d9648e4b3ae1003023ebde860c4fc96463388e second commit

f7facf58213b868ee4d2b2c6cfac4eeb5e3798a6 First commit

Виведення комітів відфільтрованих за автором:

$ git log --pretty=oneline --author="TestUser"

3e7fa6253b826923ebbd8e3d70a0d6993d21bfcd (HEAD -> master) New commit

4ee370534ba91a41d2b3467303a87302f3416cb5 New file

02545ea92c55725835af0ced04948d82247fbf09 Test status

f6d9648e4b3ae1003023ebde860c4fc96463388e second commit

f7facf58213b868ee4d2b2c6cfac4eeb5e3798a6 First commit

Можна задати власний формат виведення:

$ git log --pretty=format:"%h - %s :%ad [%an]"

3e7fa62 - New commit :Tue Oct 20 12:09:18 2020 +0300 [TestUser]

4ee3705 - New file :Tue Oct 20 11:50:56 2020 +0300 [TestUser]

02545ea - Test status :Tue Oct 20 11:31:26 2020 +0300 [TestUser]

f6d9648 - second commit :Mon Oct 19 15:26:30 2020 +0300 [TestUser]

f7facf5 - First commit :Mon Oct 19 15:15:04 2020 +0300 [TestUser]

%h – виводить скорочкеу версію хеш коду комітів;

- текстовий, що буде виводитись (вставляєм розділовий знак)

%s – виводить коментарії комітів

:%ad – виводить дату коли були виконані коміти

[%an] – виводить автора коміта

Змінимо формат виведення дати:

$ git log --pretty=format:"%h - %s :%ad [%an]" --date=short

3e7fa62 - New commit :2020-10-20 [TestUser]

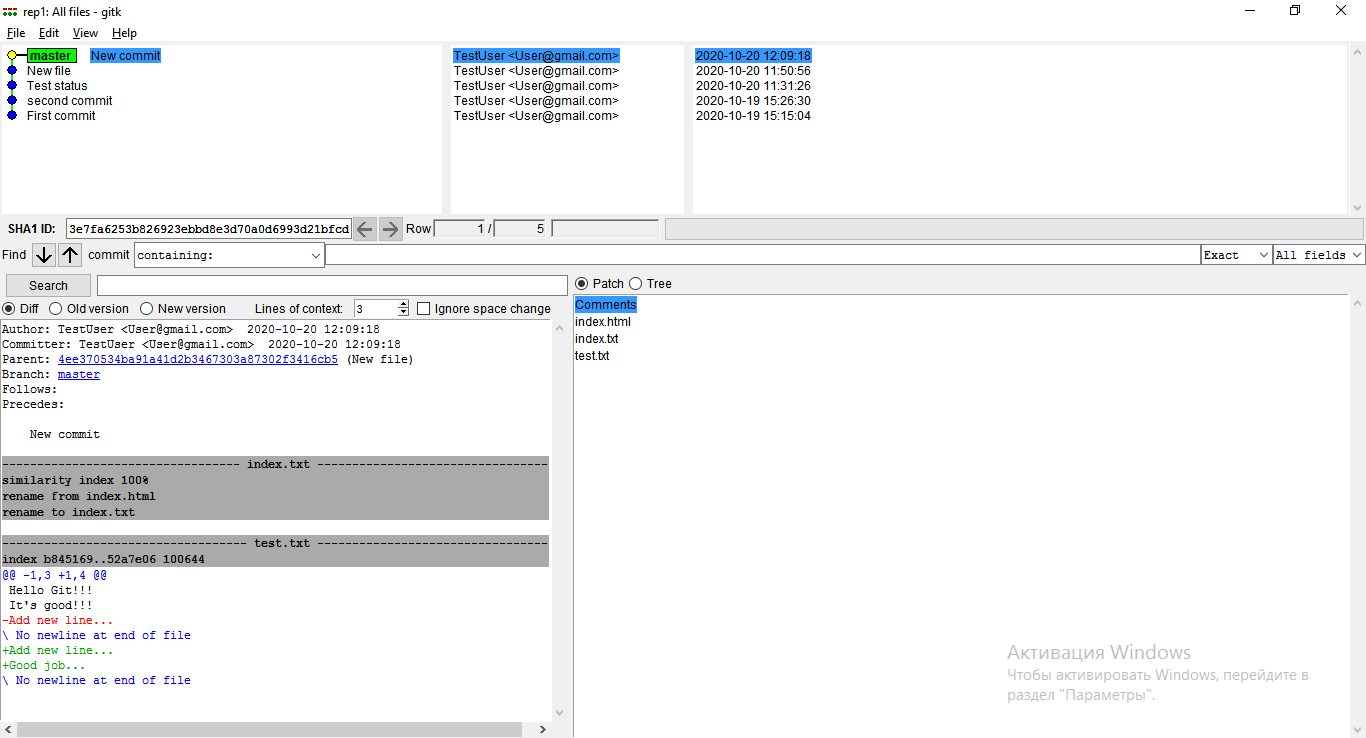
4ee3705 - New file :2020-10-20 [TestUser]

02545ea - Test status :2020-10-20 [TestUser]

f6d9648 - second commit :2020-10-19 [TestUser]

f7facf5 - First commit :2020-10-19 [TestUser]

Звичайно історію комітів можна переглянути, як і раніше, за допомогою утиліти gitk



1. **Git checkout**

Для переходу до попереднього коміту, чи переходу на іншу гілку дерева репозиторія (У нашому проекті поки усі коміти знаходяться на одній гілці. Про гілки пізніше) використовують команду Git checkout.

Припустимо, що ми зробили коміт, але зрозуміли, що він помилковий чи конфліктує з іншими комітами. Тому слід повернутися на попередній коміт. Щоб визначитись, на який коміт слід перейти використаємо вже відому команду git log.

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git log --pretty=format:"%h - %s :%ad [%an]" --date=short

3e7fa62 - New commit :2020-10-20 [TestUser]

4ee3705 - New file :2020-10-20 [TestUser]

02545ea - Test status :2020-10-20 [TestUser]

f6d9648 - second commit :2020-10-19 [TestUser]

f7facf5 - First commit :2020-10-19 [TestUser]

Перейдемо, наприклад, на коміт з коментарем New file.

Для цього використовуємо команду $ git checkout, а далі вказуємо хеш коміта, на який треба перейти. У нашому випадку це – 4ee3705.

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 ((3e7fa62...))

$ git checkout 4ee3705

Previous HEAD position was 3e7fa62 New commit

HEAD is now at 4ee3705 New file

Як бачимо ми переключилися на потрібний коміт.

Перевіримо це ще раз для певності командою ls.

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 ((3e7fa62...))

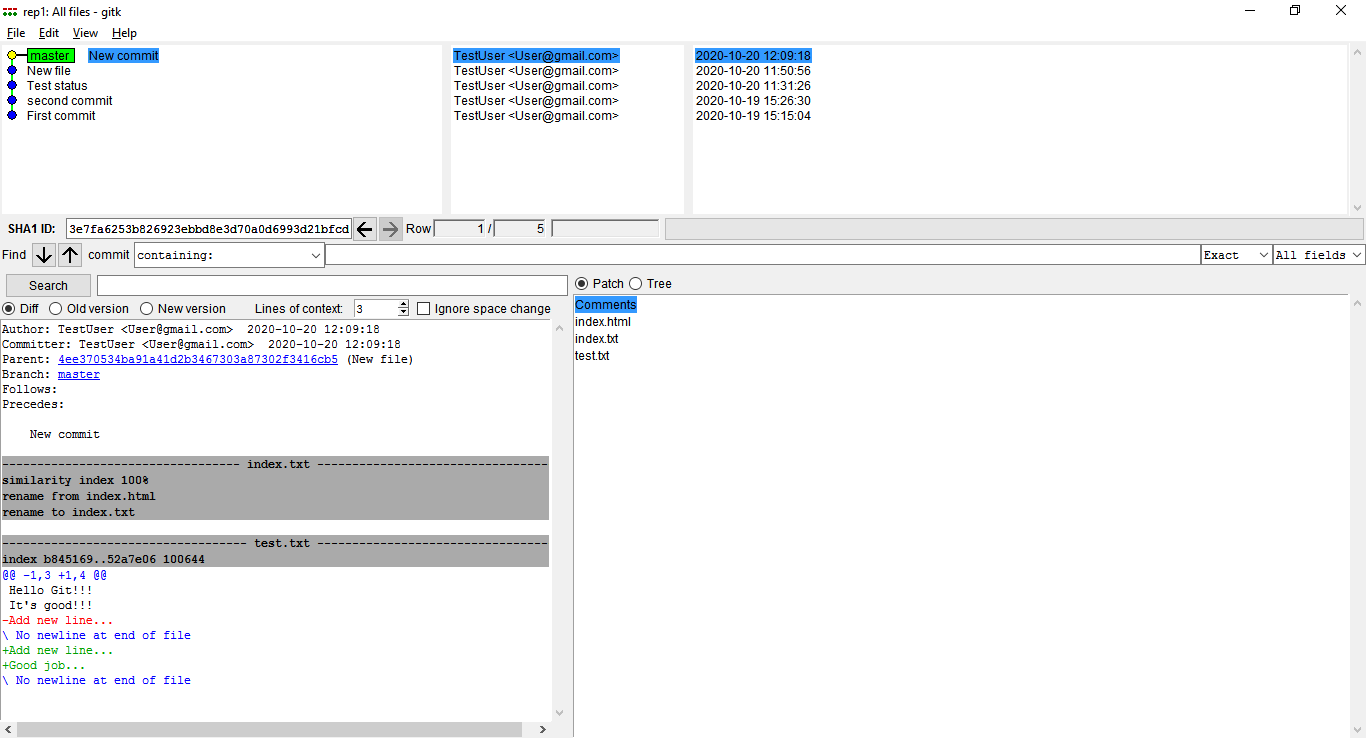
$ ls

index.txt test.txt

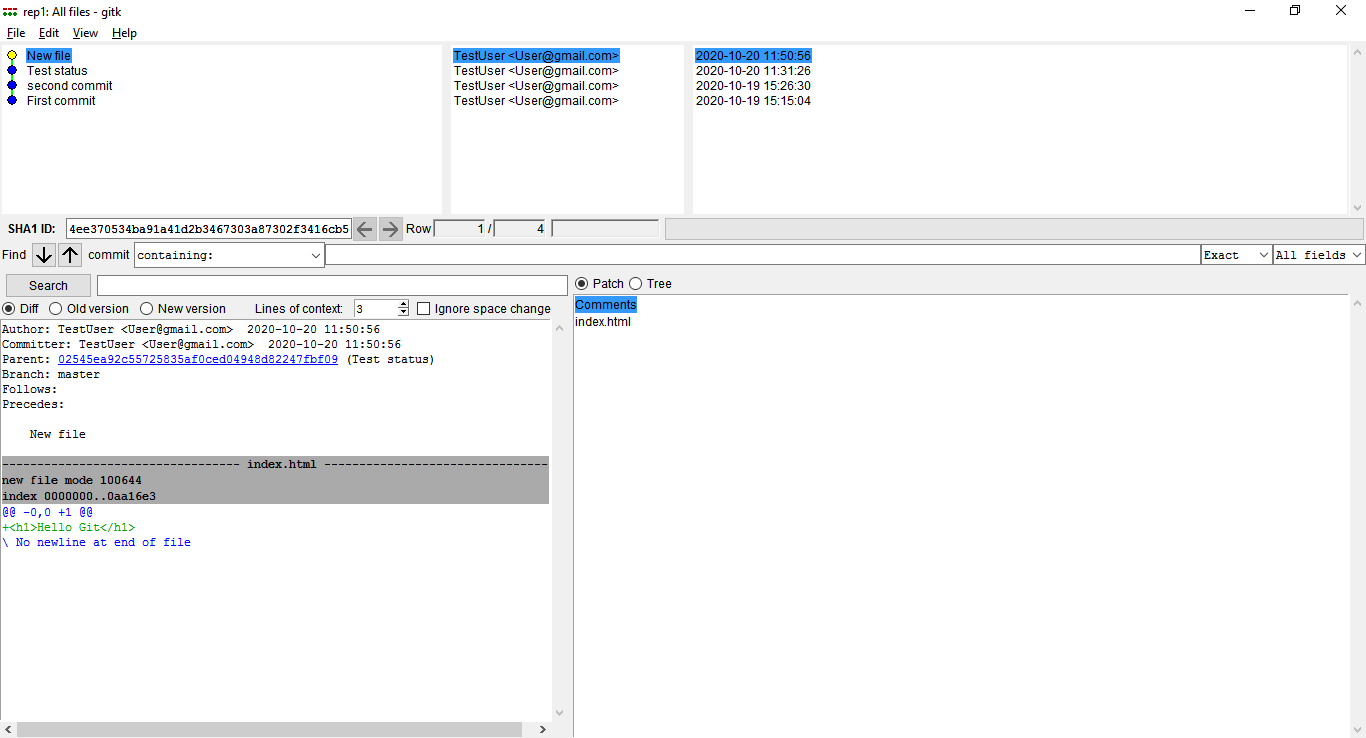
Бачимо, що в цьому коміті були два файли index.txt та test.txt

Переглянемо стан репозиторія у gitk.

Було так:



А стало:



Як видно у цьому коміті у файл index.html ми додали рядок Hello Git.

Давайте знову переглянемо історію комітів

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 ((4ee3705...))

$ git log --pretty=format:"%h - %s :%ad [%an]" --date=short

4ee3705 - New file :2020-10-20 [TestUser]

02545ea - Test status :2020-10-20 [TestUser]

f6d9648 - second commit :2020-10-19 [TestUser]

f7facf5 - First commit :2020-10-19 [TestUser]

І перейдемо на коміт TestStatus

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 ((4ee3705...))

$ git checkout 02545ea

Previous HEAD position was 4ee3705 New file

HEAD is now at 02545ea Test status

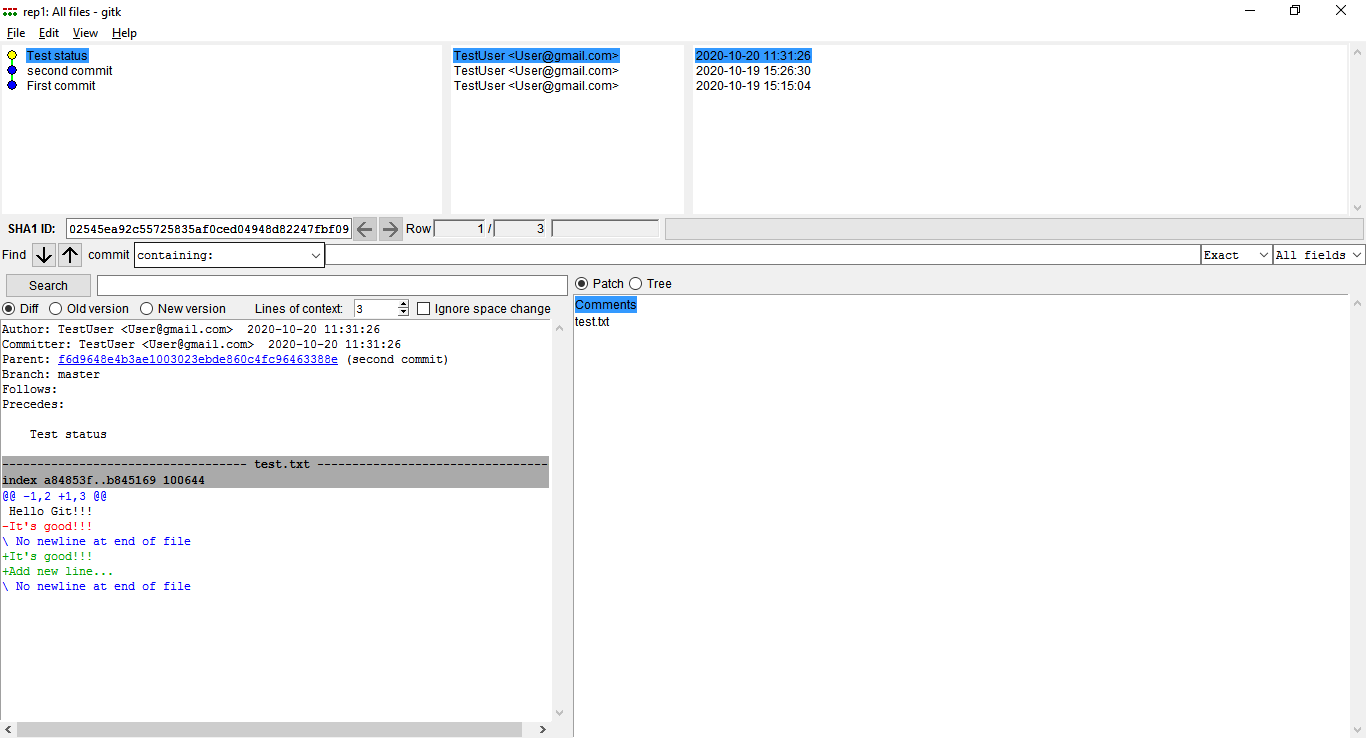
Перевіримо вміст

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 ((02545ea...))

$ ls

test.txt

Як бачимо у репозиторії є лише файл test.txt, оскільки в цьому коміті був створений лише один файл test.txt, а index.html ще не був створений…



За допомогою такого алгоритму здійснюють перехід між комітами, здійснюючи пошук хто й коли, щось «поламав» у програмі…

Якщо потрібно повернутись на останній виконаний коміт, можна просто перейти на гілку master. Оскільки ми поки працюємо з однією гілкою, то ми фактично перейдемо на останній коміт…

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 ((02545ea...))

$ git checkout master

Previous HEAD position was 02545ea Test status

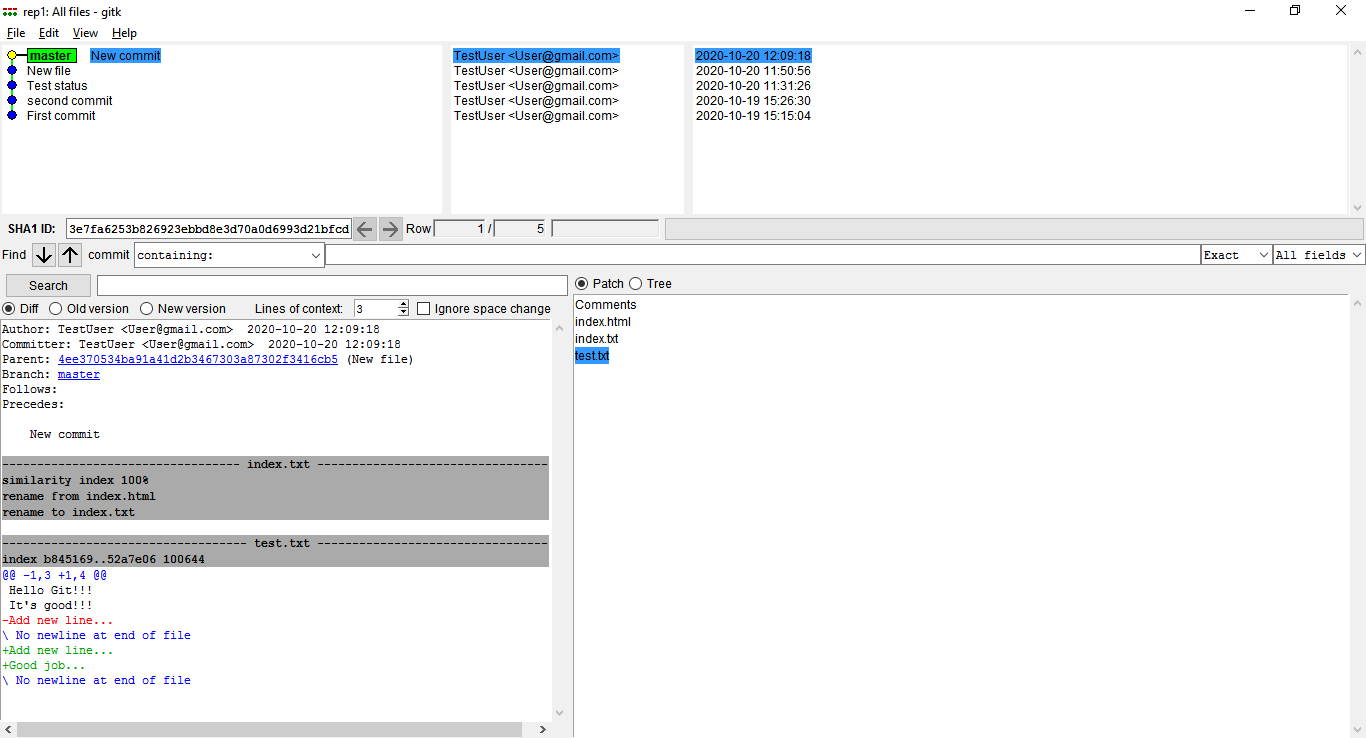
Switched to branch 'master'

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ ls

index.txt test.txt

Бачимо знову є 2 файли. Усі коміти знову в репозиторії.



1. **Відміна індексованих файлів (git reset).**

Git працює зі змінами файлів, або індексаціями файлів. Тому перш ніж закомітитись ми додавали файл на індексацію (add). Інколи, коли ми здійснили ряд змін у файлах, провели індексації, і можна комітитись, виявили, що деякі зміни не потрібно проводити. Тому треба деякі індексації, зміни відмінити, або видалити перед комітом.

Поекспериментуємо.

Переглянемо файли репозитрія.

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ ls

index.txt test.txt

Змінимо файл test.txt і збережемо файл.

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git status

On branch master

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: test.txt

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

Як бачимо файл дійсно змінений і готовий до індексації (додання на коміт кандидат). Додамо.

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git add test.txt

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: test.txt

Як видно файл проіндексовано і гіт розуміє, які у файлі відбулись зміни.

Змінимо інший файл проекту index.txt

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: test.txt

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: index.txt

Як бачимо файл дійсно змінений і готовий до індексації (додання на коміт кандидат). Додамо.

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git add index.txt

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: index.txt

modified: test.txt

Як бачимо оба файли змінені та проіндексовані. Припустимо, що ми зрозуміли, що зроблені зміни у файлі test.txt не потрібні і слід відкотити зміни, виконані у цьому файлі. Для цього існує команда git reset. Ця команда дозволяє повернутись до деякого потрібного стану гілки, коміта чи файлу.

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git reset HEAD test.txt

Unstaged changes after reset:

M test.txt

Таким чином ми повертаємось до останнього закоміченого стану (до останньої версії) файлу test.txt.

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: index.txt

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: test.txt

Видно, що файл test.txt змінений, але не проіндексований. Тобто ми відмінили індексацію.

Щоб повністю видалити непотрібні зміни у файлі test.txt виконаємо git checkout (перейдемо на останній закомічений стан файла test.txt).

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git checkout test.txt

Updated 1 path from the index

Администратор@Servak MINGW64 /d/Git\_Rep/rep1 (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: index.txt

1. Відміна коміта (git revert)

При роботі з гіт, часто необхідно видалити коміт чи серію комітів, що мають критичні, конфліктні зміни проекту, чи просто ці коміти помилкові і їх слід видалити. Подивимося як це зробити.

Спершу зробимо зміни у файлі index.txt, які будуть потенційним невірним комітом.