Лабораторна робота №3

Тема: Робота з командами Git, створення та злиття гілок

Мета: Набути досвід роботи з командами Git, навики роботи по використанню різних версій даних репозиторію, створенню та злиття гілок.

Хід роботи

1) Вивчити теоретичний матеріал до лабораторної роботи.
2) Зайдіть до репозиторію, створеному на попередньому занятті.
3) Створіть дві гілки з назвами version_2 та version_3.
4) Перейдіть по черзі на кожну із гілок.
5) Додайте до обох гілок будь-який файл (не забудьте додати файли, закомітити та залити).
6) Злийте гілку master та version_2.
7) Видаліть гілку version_3.
8) Дайте відповідь на контрольні питання та здайте роботу викладачу.

Контрольні питання:

1. В якому статусі можуть перебувати файли при роботі з Git?
2. Для чого використовується команда add?
3. Для чого використовується команда status?
4. Для чого використовується команда commit?
5. Для чого використовується команда init?
6. Що необхідно писати в коментарях комітів?
7. Для чого використовується команда checkout?
8. Для чого використовується команда reset?
9. Для чого використовується команда log?
10. Для чого використовується команда status?
11. Як відновити кілька файлів з попередньо створеного коміта?
12. Як перейди до попереднього коміта по імені?
13. Як створити нову гілку?
14. Як перейти на нову гілку?
15. Як перейти на основну гілку?
16. Як виконати злиття гілок?

Критерії оцінювання

№ п/п	Види робіт. Критерії оцінювання знань студентів	Максимальна к-сть балів
1	Робота виконана у зазначений термін, у повному обсязі, без помилок і зарахована.	5
2	Робота виконана у зазначений термін, у повному обсязі, зарахована, але є помилки.	4
3	Робота виконана у неповному обсязі, або з порушенням терміну виконання, або при наявності значних помилок.	3
4	Виконання пропущеної роботи або повторне виконання не зарахованої роботи.	2
5	Робота не виконана або не зарахована.	0

Теоретичний матеріал

Гілки. Можливості використання гілок.

Під час розробки нового функціоналу вважається хорошою практикою працювати з копією оригінального проекту, яку називають **гілкою**.

Гілки мають свою власну історію і ізольовані один від одної зміни до тих пір, поки ви не вирішуєте злити зміни разом.

Такої тактики дотримуються з ряду **причин**:

- Уже робоча, стабільна версія коду зберігається.
- Різні нові функції можуть досліджуватися різними програмістами.
- Розробники можуть працювати з власними гілками без ризику, що кодова база зміниться через чужі зміни.
- У випадку сумнівів, різні реалізації однієї і тієї ж ідеї можуть бути розроблені в різних гілках і потім порівнюватися.

4. Робота з гілками.

Якщо ми робимо якісь зміни та фіксуємо їх, наступна фіксація буде зберігати вказівник на попередню.

Гілка в Git – це простий вказівник, що може пересуватись на одну з цих фіксацій. Загальноприйнятим ім’ям першої гілки в Git є master.

Створення нової гілки

Що відбувається, якщо створюємо нову гілку? Це створює новий вказівник, щоб ми могли пересуватися. Припустімо, ви створюємо нову гілку під назвою testing. Ви це робите за допомогою команди **git branch**:

$ **git branch** testing

Це створює новий вказівник на фіксацію, в якій ми зараз знаходимось.

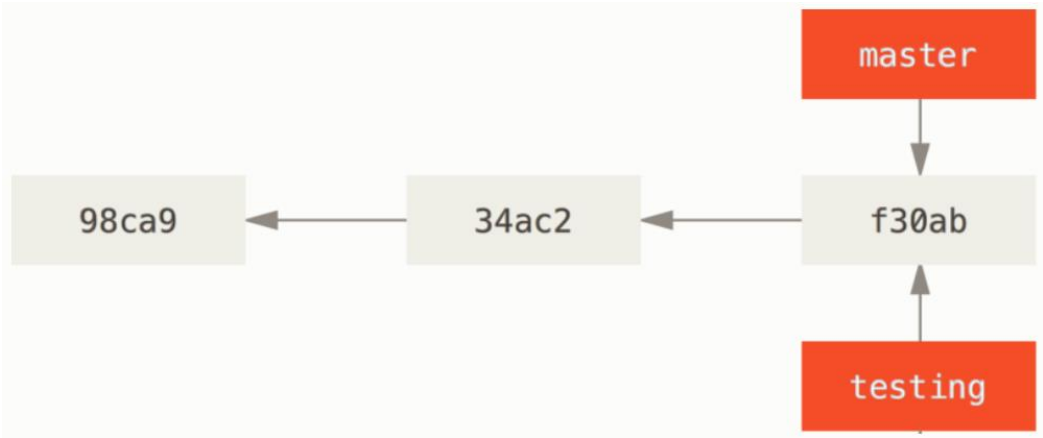

Рисунок - Дві гілки вказують на одну послідовність фіксацій

Звідки Git знає, на якій гілці ми зараз знаходимось? Він зберігає особливий вказівник під назвою HEAD.

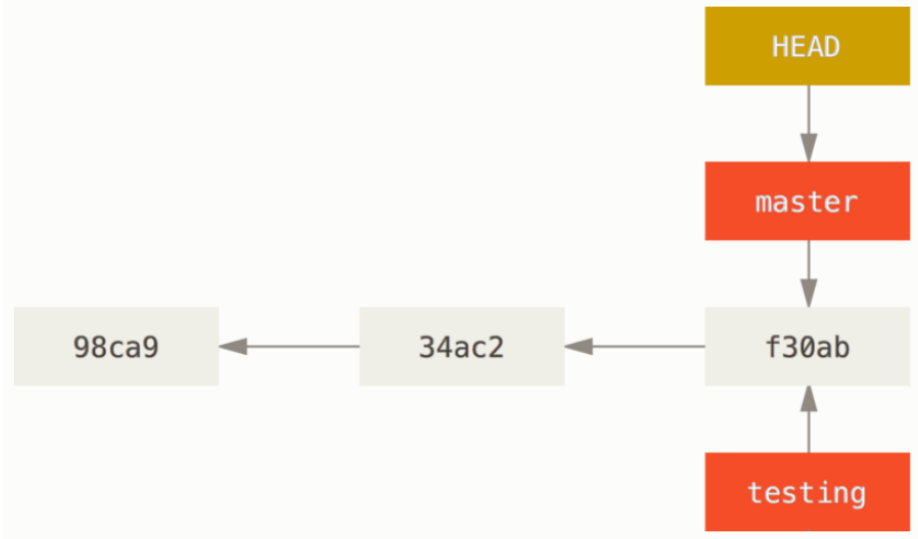

Рисунок - HEAD вказує на гілку

Команда **git branch** лише створює нову гілку – вона не переключає на цю гілку.

Це можна легко побачити за допомогою простої опції **--decorate** команди **git log**, що може показати куди вказують вказівники гілок.

$ **git** log --oneline –decorate

Переключення гілок

Щоб переключитися на існуючу гілку, треба виконати команду git checkout. Переключімося на нову гілку testing:

$ **git checkout** testing

Це пересуває HEAD, щоб він вказував на гілку testing.

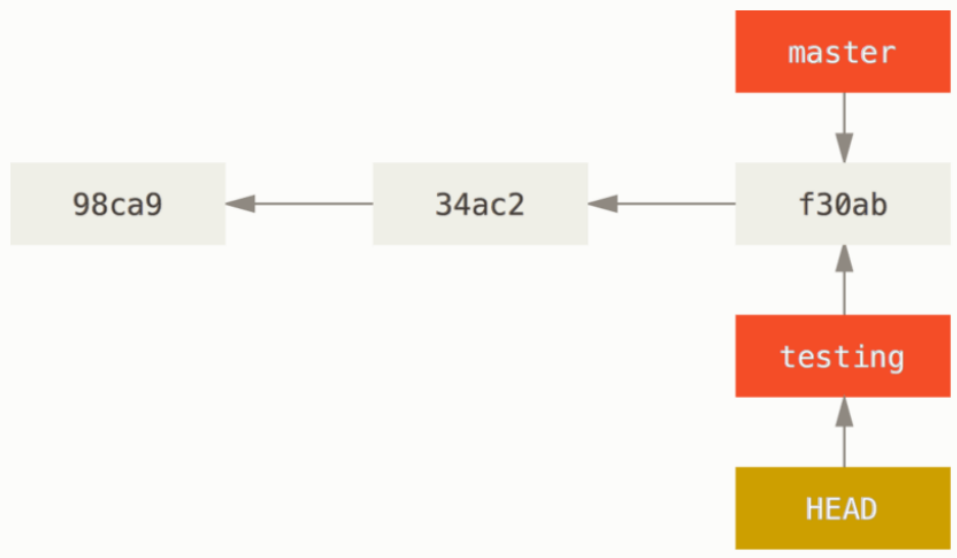

Рисунок - HEAD вказує на поточну гілку

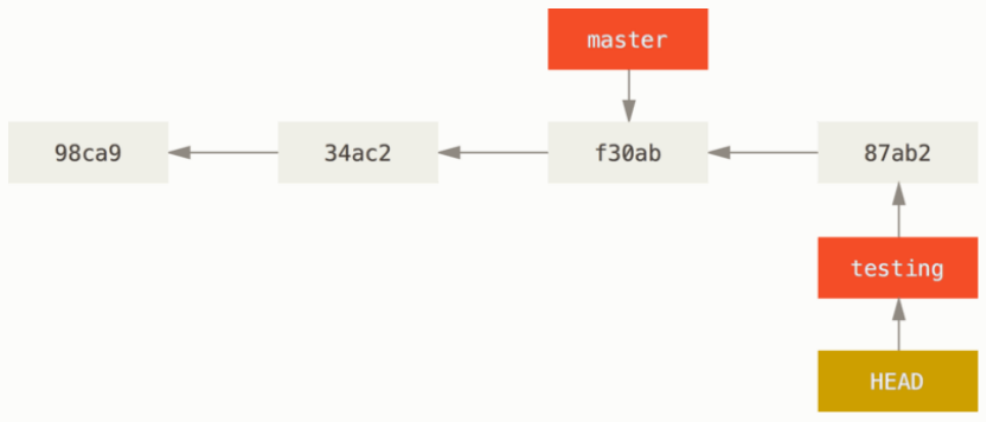

Рисунок - Гілка HEAD пересувається уперед при фіксації

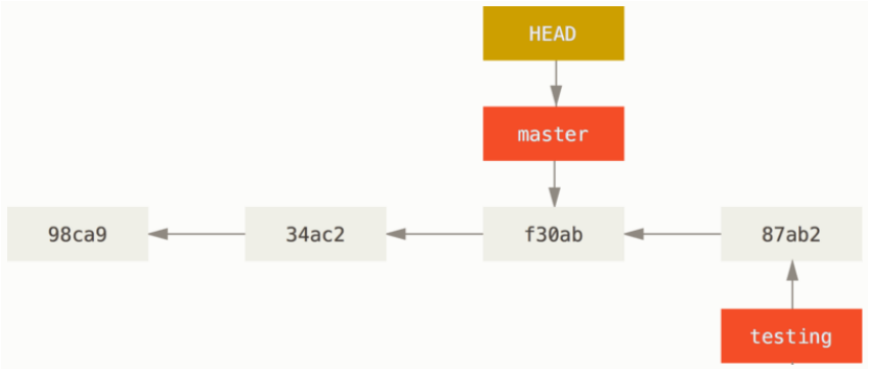

Рисунок - Переключення знову на гілку master

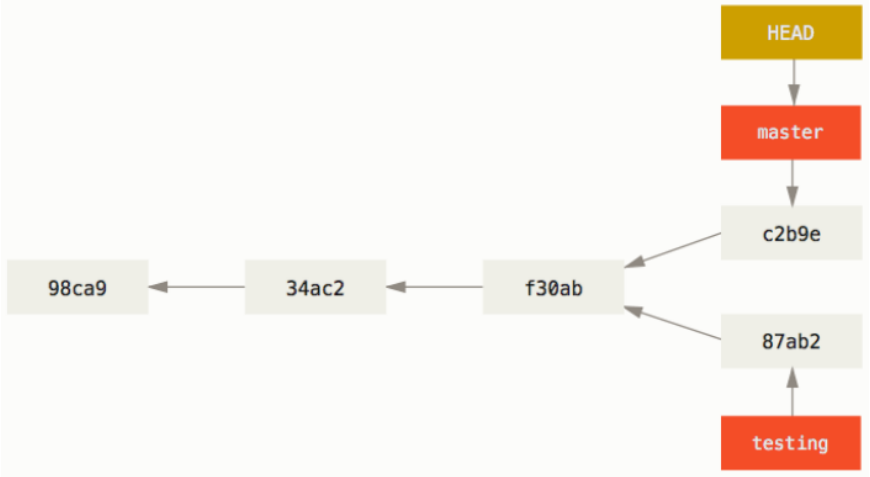

Рисунок - Історія гілки

Якщо ви виконаєте **git log --oneline --decorate --graph --all**, буде надруковано історію фіксацій, показано куди вказують гілки та як розбіглася історія.

Оскільки гілка в Git – це простий файл, що містить 50 символів контрольної суми SHA-1 коміту, на який вказує, гілки дешево створювати та знищувати. Створити гілку так же швидко, як записати 41 байт до файлу (40 символів та символ нового рядка).

Основи зливання

Якщо ми раніше створювали експериментальну гілку, а після закінчення впевнилися в тому, що все працює справно, можемо злити створену гілку із гілкою master. Для цього необхідно переключитись на робочу гілку командою:

$ **git checkout** master

та виконати команду:

$ **git merge** приєднувана_гілка

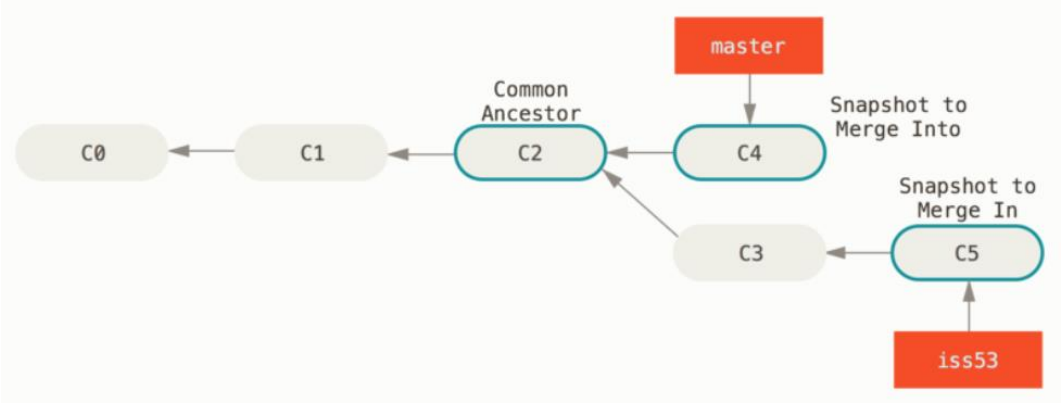

Рисунок - Злиття гілок

Тепер, коли ваші зміни злиті, гілка більше не потрібна, можемо закривати задачу та видаляти гілку:

$ **git branch –d** ім’я_гілки

Команда **git branch** насправді вміє більше ніж просто створювати та знищувати гілки. Запуск її без параметрів показує список усіх гілок:

$ **git branch**

* master

гілка_1

гілка_2

Зверніть увагу на символ * перед **master**: це вказівник на поточно вибрану гілку (тобто ту, на котру вказує HEAD). Це означає, що якщо ми зараз захочемо зробити коміт, master оновиться нашими новими змінами. Щоб побачити ваші останні коміти - запустіть **git branch –v**.

Опції **--merged** та **--no-merged** корисні для фільтрування списку гілок залежно від того чи вони були злиті з поточною гілкою. Для списку гілок, що були злиті з поточною гілкою виконайте **git branch --merged**. Команда **git branch --no-merged** покаже гілки, які ви не зливали з поточною гілкою.

Гілки без * із цього списку можна вже видаляти (за допомогою **git branch -d**).