**Теоретичний матеріал до теми «Керування віддаленими репозиторіями. Гілки»**

**Мета вивчення теми:** опанувати основи систем керування версіями на базі системи Git, навчитись керувати віддаленими репозиторіями та створювати гілки.

**План**

1. Клонування існуючого репозиторію.
2. Завантаження репозиторію.
3. Гілки. Можливості використання гілок.
4. Робота з гілками.

**1. Клонування існуючого репозиторію.**

Якщо ви бажаєте отримати копію існуючого Git репозиторію — наприклад, проекту, в якому ви хочете прийняти участь — вам потрібна команда **git clone**.

Так ви отримує повну копію майже всіх даних, що є у сервера. Кожна версія кожного файлу в історії проекту витягується автоматично, коли ви виконуєте **git clone**. Насправді, якщо щось станеться з диском вашого серверу, ви зазвичай можете використати майже будь-який з клонів на будь-якому клієнті щоб повернути сервер до стану на момент клонування.

Щоб клонувати репозиторій треба використати команду **git clone** url\_проекту. Наприклад, якщо ви бажаєте зробити клон бібліотеки Git libgit2, ви можете це зробити так:

$ **git clone** https://github.com/libgit2/libgit2

Це створить директорію під назвою libgit2, проведе ініціалізацію директорії .git, забере всі дані для репозиторія, та приведе директорію до стану останньої версії. Якщо ви зайдете до щойно створеної директорії libgit2, ви побачите, що всі файли проекту на місці, готові для використання.

Якщо ви бажаєте зробити клон репозиторія в директорію з іншою назвою, ви можете передати її як другий параметр команди:

$ **git clone** https://github.com/libgit2/libgit2 mylibgit

Ця команда робить те саме, що й попередня, тільки цільова директорія називається mylibgit.

Git має декілька різних протоколів передачі даних, які ви можете використовувати. Попередній приклад використовує протокол https://, проте ви також можете побачити git:// або user@server:шлях/до/репозиторія.git, що використовує SSH протокол.

**2. Завантаження репозиторію.**

Отримання (fetching) та скачування (pulling) з ваших віддалених сховищ

Щоб отримати дані з ваших віддалених проектів, ви можете виконати:

$ **git fetch** віддалений\_репозиторій

Ця команда заходить на віддалений проект та забирає звідти усі дані, котрих у вас досі нема. Після цього, у вас будуть посилання на всі гілки з того сховища, які ви можете зливати або оглядати в будь-який час.

Якщо ви зробили клон сховища, команда автоматично додає це віддалене сховище під ім’ям “origin”. Команда **git fetch** лише завантажує дані до вашого локального сховища — вона автоматично не зливає їх з вашою роботою, та не змінює вашу поточну роботу. Вам буде потрібно вручну їх злити, коли будете готові.

Якщо ваша поточна гілка налаштована слідкувати за віддаленою гілкою, ви можете виконати команду **git pull**, щоб автоматично отримати зміни та злити віддалену гілку до вашої поточної гілки. Це може бути легшим та зручнішим методом для вас. Виконання **git pull** зазвичай дістає дані з серверу, з якого ви зробили клон, та намагається злити її з кодом, над яким ви зараз працюєте.

**3. 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.

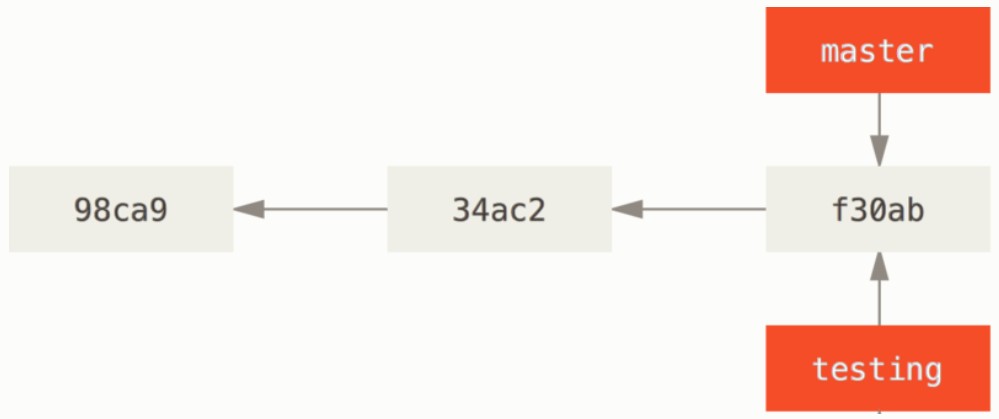
Гілка в Git – це простий вказівник, що може пересуватись на одну з цих фіксацій. Загальноприйнятим ім’ям першої гілки в Git є master.

**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**:

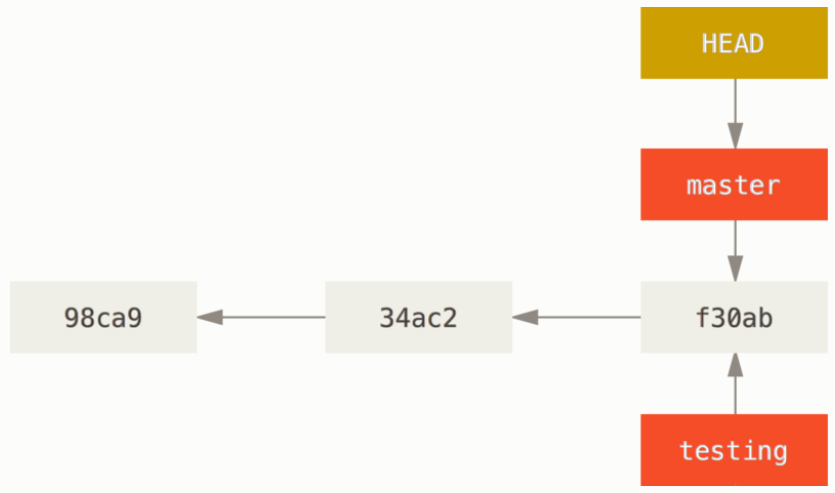
$ **git branch** testing

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



Мал. Дві гілки вказують на одну послідовність фіксацій

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



Мал. 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.

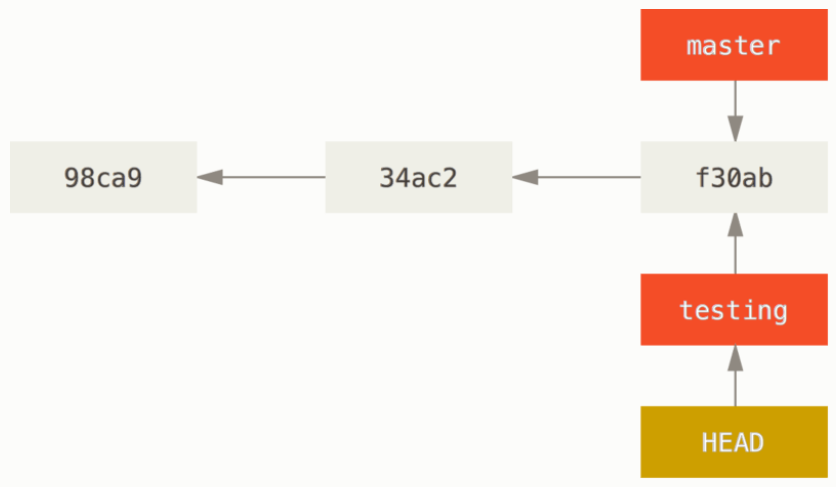
$ **git** log --oneline –decorate

**Переключення гілок**

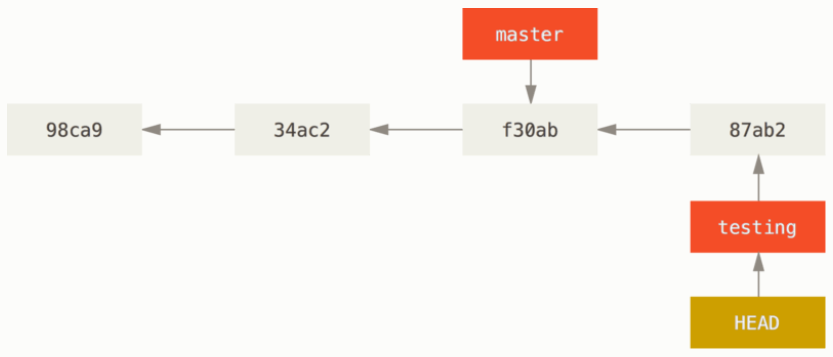
Щоб переключитися на існуючу гілку, треба виконати команду git checkout. Переключімося на нову гілку testing:

$ **git checkout** testing

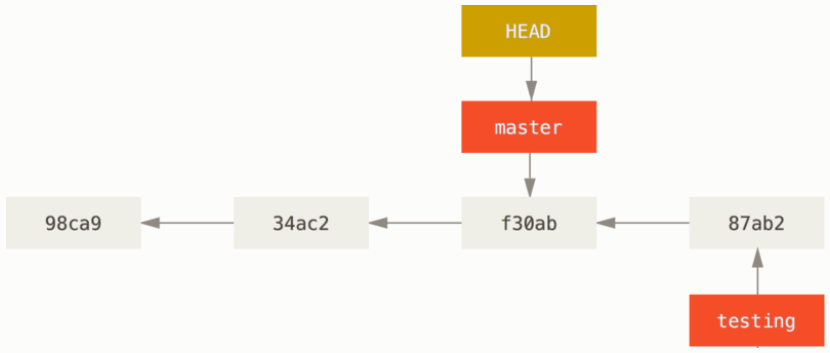
Це пересуває HEAD, щоб він вказував на гілку testing.



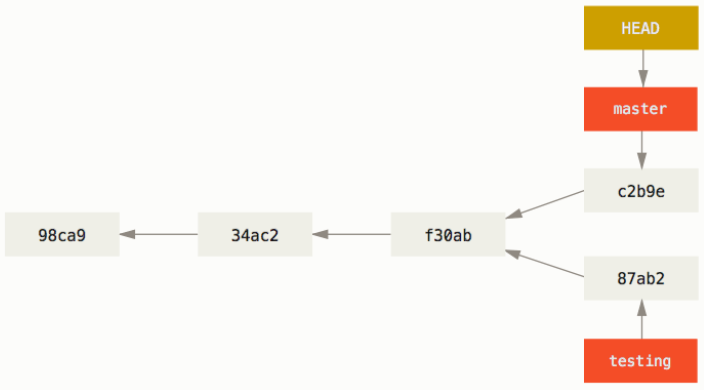
Мал. HEAD вказує на поточну гілку



Мал. Гілка HEAD пересувається уперед при фіксації



Мал. Переключення знову на гілку master



Мал. 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).

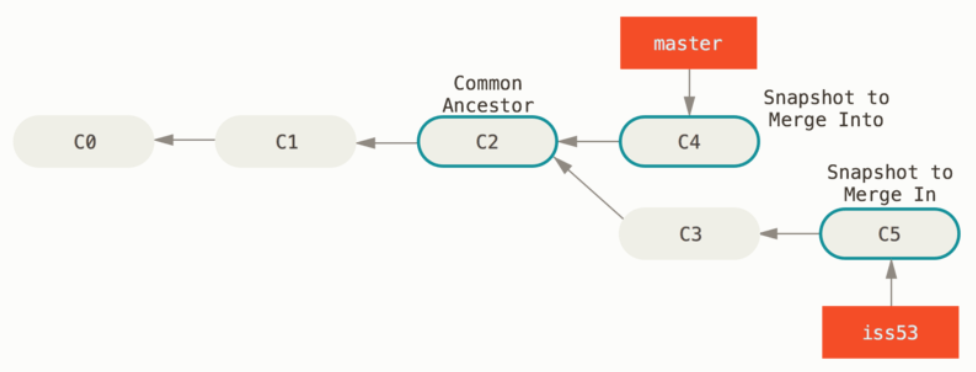
**Основи зливання**

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



Мал. 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**).

**Завдання для самостійної роботи студентів**

Опрацювати відповідний матеріал посібника Pro Git електронного ресурсу: https://git-scm.com/book/ru/v2