Галуження в git - Віддалені гілки

**Віддалені гілки**

Віддалені посилання — це посилання (вказівники) у ваших віддалених сховищах: гілки, теґи тощо. Для повного списку віддалених посилань виконайте git ls-remote [remote], або git remote show [remote] для детальної інформації про віддалені гілки. Проте, найпоширеніше застосування — це віддалено-відслідковувані гілки.

Віддалено-відслідковувані гілки — це вказівники на стан віддалених гілок. Локально ці вказівники неможливо змінити, але їх змінює Git, коли ви виконуєте мережеві операції, щоб вони точно відповідали стану віддаленого сховища. Вважайте їх закладками, що нагадують вам про стан віддалених репозиторіїв на момент вашого останнього зв’язку з ними.

Віддалені гілки мають такий запис: <віддалене сховище>/<гілка>. Наприклад, якщо ви хочете побачити як виглядала гілка master з віддаленого сховища origin, коли ви востаннє зв’язувалися з ним, перейдіть на гілку origin/master. Припустимо, ви працювали з колегами над одним завданням і вони вже виклали свої зміни. У вас може бути своя локальна гілка iss53, але гілці на сервері відповідатиме віддалена гілка origin/iss53.

Це може трохи спантеличувати, розгляньмо приклад. Скажімо, ви працюєте з Git сервером, що доступний у вашій мережі за адресою git.ourcompany.com. Коли ви склонуєте з нього, команда clone автоматично іменує його origin, стягує всі дані, створює вказівник на те місце, де зараз знаходиться master і локально іменує це посилання origin/master, щоб ви могли з чогось почати працювати.

|  |  |
| --- | --- |
| **Зауваження** | **``origin'' не є особливим**  Так само, як назва гілки master'' не має якогось особливого значення для Git, так само й origin''. Просто master'' дається за-замовчуванням для початкової гілки, коли ви запускаєте git init — ось чому воно так часто зустрічається, а origin'' — це ім’я за-замовчуванням для віддалених посилань команди git clone. Якщо ж ви, натомість, виконаєте git clone -o booyah, то отримаєте booyah/master своєю гілкою за-замовчуванням. |



Рисунок 30. Серверний та локальний репозиторії після клонування

Якщо ви виконали якусь роботу на локальній гілці master, і водночас, хтось виклав зміни на git.ourcompany.com в master, тоді ваші історії прогресують по-різному. Доки ви не синхронізутесь з сервером, вказівник origin/master не буде рухатись.



Рисунок 31. Локальна та віддалена робота розійшлися

Щоб відновити синхронність, виконайте команду git fetch origin. Ця команда шукає який сервер відповідає імені `origin'' (у нашому випадку `git.ourcompany.com), отримує дані, яких ви ще не маєте і оновлює вашу локальну базу даних, переміщаючи вказівник origin/master на нову, більш актуальну, позицію.



Рисунок 32. git fetch оновлює віддалені посилання

Щоб продемонструвати роботу з кількома віддаленими серверами і як виглядають віддалені гілки для віддалених проектів, уявімо, що ви маєте ще один внутрішній Git сервер, котрий використовує лиш одна з ваших спринт команд. Cервер розташований за адресою git.team1.ourcompany.com. Ви можете додати його як нове віддалене посилання до вашого поточного проекту за допомогою команди git remote add, як ми розповідали в [Основи Git](https://git-scm.com/book/uk/v2/ch00/ch02-git-basics-chapter). Дайте йому ім’я teamone, і це буде вашим скороченням для повного URL.



Рисунок 33. Додавання нового віддаленого сервера

Тепер виконайте git fetch teamone щоб витягнути з teamone всі оновлення. Оскільки teamone на даний момент є підмножиною origin, то Git не отримує нових даних і нічого не оновлює, а просто ставить віддалено-відслідковувану гілку teamone/master вказувати на коміт, на котрому зараз знаходиться гілка master для сервера origin.



Рисунок 34. Віддалено-відслідковувана гілка teamone/master

**Надсилання**

Коли ви хочете поділитися гілкою з навколишнім світом, надішліть (push) її на віддалене посилання, до якого маєте право запису. Ваші локальні гілки не синхронізуються з віддаленими автоматично — потрібно явно надсилати гілки, якими хочете поділитися. Таким чином, можете користуватися приватними гілками для роботи, якою не збираєтеся ділитися, а надсилати зміни тільки в тематичні гілки, над якими співпрацюєте.

Щоб поділитися з іншими своєю гілкою з назвою serverfix, надсилайте її так само як це робили із першою гілкою. Виконайте git push <remote> <branch>:

$ git push origin serverfix

Counting objects: 24, done.

Delta compression using up to 8 threads.

Compressing objects: 100% (15/15), done.

Writing objects: 100% (24/24), 1.91 KiB | 0 bytes/s, done.

Total 24 (delta 2), reused 0 (delta 0)

To https://github.com/schacon/simplegit

\* [new branch] serverfix -> serverfix

Це трохи скорочено. Git автоматично замінює назву гілки serverfix з refs/heads/serverfix:refs/heads/serverfix, що означає Візьми мою локальну гілку serverfix та надішли її для оновлення віддаленої гілки serverfix.'' Ми детальніше розглянемо refs/heads/ в [Git зсередини](https://git-scm.com/book/uk/v2/ch00/ch10-git-internals), але, взагалі, можете зараз це пропустити. Також ви можете виконати git push origin serverfix:serverfix, що зробить те саме, тобто скаже Бери мій serverfix та шли на віддалений serverfix.'' Користуйтеся таким підходом щоб надсилати локальні гілки, що називаються інакше ніж віддалені. Якщо ви не хочете мати віддалену назву serverfix, виконайте натомість git push origin serverfix:awesomebranch, щоб надіслати локальну serverfix на віддалену з назвою awesomebranch.

|  |  |
| --- | --- |
| **Зауваження** | **Не вводьте свій пароль щоразу**  Якщо ви користуєтеся HTTPS для надсилання змін, Git сервер запитуватиме ваші ім’я користувача та пароль для аутентифікації. За замовчуванням Git питатиме вас цю інформацію в терміналі для того, щоб перевірити правомірність надсилання змін.  Щоб не вводити це щоразу, налаштуйте `credential cache''. Найпростіше — дозволити тримати цю інформацію в пам’яті протягом кількох хвилин, для чого виконайте `git config --global credential.helper cache.  Більше інформації про способи кешування ідентифікації користувачів у секції [Збереження посвідчення (credential)](https://git-scm.com/book/uk/v2/ch00/_credential_caching). |

Наступного разу, коли співпрацівники будуть отримувати зміни з сервера, — отримають вказівник на поточний стан серверного serverfix у віддалену гілку-посилання origin/serverfix:

$ git fetch origin

remote: Counting objects: 7, done.

remote: Compressing objects: 100% (2/2), done.

remote: Total 3 (delta 0), reused 3 (delta 0)

Unpacking objects: 100% (3/3), done.

From https://github.com/schacon/simplegit

\* [new branch] serverfix -> origin/serverfix

Варто зауважити, що, коли ви отримуєте зміни (за допомогою fetch), — насправді це нові віддалено-відслідковувані гілки, а не автоматичні локальні копії цих гілок, які ви можете редагувати. Іншими словами, в цьому випадку, ви не маєте нової гілки serverfix, а просто вказівку на origin/serverfix, яку не можете змінювати.

Щоб злити ці отримані зміни в вашу поточну робочу гілку виконайте git merge origin/serverfix. Якщо хочете мати свою власну гілку serverfix і працювати над нею, можете створити її, базуючись на віддалено-відслідковуваній гілці:

$ git checkout -b serverfix origin/serverfix

Branch serverfix set up to track remote branch serverfix from origin.

Switched to a new branch 'serverfix'

За допомогою цього отримаєте гілку, де ви можете працювати, та котра відображає стан origin/serverfix.

**Відслідковувані гілки**

Перемикання на локальну гілку з віддалено-відслідковуваної автоматично створює так звану відслідковувану гілку tracking branch'' (а гілка, за якою вона стежить називається upstream branch''). Відслідковувані гілки — це локальні гілки, що мають безпосередній зв’язок з віддаленою гілкою. Якщо ви знаходитеся на відслідковуваній гілці і потягнете зміни, виконуючи git pull, Git відразу знатиме з якого сервера брати та з якої гілки зливати зміни.

Коли ви клонуєте репозиторій, Git автоматично створює гілку master, яка слідкує за origin/master Проте, ви можете налаштувати й інші відслідковувані гілки — такі, що слідкують за іншими віддаленими посиланнями, або за гілкою, відмінною від master. Як у випадку, що ви бачили в прикладі, виконуючи git checkout -b <гілка> <назва віддаленого сховища>/<гілка>. Це досить поширена дія, Git має опцію --track для скороченого запису:

$ git checkout --track origin/serverfix

Branch serverfix set up to track remote branch serverfix from origin.

Switched to a new branch 'serverfix'

Насправді, це настільки поширено, що навіть для цього скорочення є скорочення. Якщо назва гілки, яку ви намагаєтеся отримати (а) не існує і (б) має таку саму назву, як і гілка тільки з одного віддаленого сховища, то Git створить стежачу гілку:

$ git checkout serverfix

Branch serverfix set up to track remote branch serverfix from origin.

Switched to a new branch 'serverfix'

Щоб дати локальній гілці назву, що відрізняється від серверної, виконайте попередню повну команду, вказуючи бажане ім’я гілки:

$ git checkout -b sf origin/serverfix

Branch sf set up to track remote branch serverfix from origin.

Switched to a new branch 'sf'

Тепер локальна гілка sf буде автоматично витягувати зміни з origin/serverfix.

Якщо ж у вас вже є локальна гілка і ви хочете прив’язати її до віддаленої, чи змінити віддалену (upstream) гілку, можете використовувати опції -u чи --set-upstream-to до команди git branch.

$ git branch -u origin/serverfix

Branch serverfix set up to track remote branch serverfix from origin.

|  |  |
| --- | --- |
| **Зауваження** | **Скорочене звертання до upstream**  Коли відслідковувана гілка налаштована, ви можете використовувати скорочений запис @{upstream} чи @{u}. Тобто, при бажанні, знаходячись на master, що слідкує за origin/master, користуйтеся чимось на зразок git merge @{u} замість повного git merge origin/master. |

Опція -vv до git branch дозволяє дізнатися, які у вас налаштовані відслідковувані гілки. Результатом буде список локальних гілок та інформація про них, включаючи, які гілки відслідковуються та деталі про те, чи вони випереджають чи відстають від локальних (чи те й інше).

$ git branch -vv

iss53 7e424c3 [origin/iss53: ahead 2] forgot the brackets

master 1ae2a45 [origin/master] deploying index fix

\* serverfix f8674d9 [teamone/server-fix-good: ahead 3, behind 1] this should do it

testing 5ea463a trying something new

Тут ми бачимо, що гілка iss53 слідкує за origin/iss53 та випереджає її (`ahead'') на два, тобто ми маємо локально два коміти, які ще не надіслані на сервер. Також ми бачимо, що `master слідкує за origin/master та її стан є актуальним. Далі бачимо, що serverfix слідкує за гілкою server-fix-good з сервера teamone та випереджає його на три й відстає на один, тобто існує один коміт на сервері, який ми ще не злили та три коміти локально, які ми ще не надіслали. Насамкінець бачимо, що локальна гілка testing не слідкує за жодною віддаленою.

Варто зауважити, що ці числа відображають стан віддалених гілок на час останнього оновлення з кожного сервера. Сама по собі команда не сягає серверів, вона просто відображає те, що збережено про ці сервери локально. Якщо ж ви хочете отримати найостаннішу інформацію про випередження чи відставання гілок, оновіть спочатку всі віддалені посилання. Можете це зробити ось як:

$ git fetch --all; git branch -vv

**Витягування змін**

Команда git fetch, під час виконання, отримує всі оновлення, яких ви ще не маєте, але, зовсім не змінює вашу робочу директорію. Вона просто отримує дані для того, щоб ви могли самотужки злити зміни. Існує команда git pull, яка, по своїй суті та в більшості випадків, є послідовним виконанням команд git fetch та git merge. Якщо у вас є відслідковувана гілка, як ми показували у попередній секції, — створена та самостійно налаштована, чи як результат clone чи checkout, команда git pull буде звертатися до відслідковуваних сервера та віддаленої гілки, отримувати оновлення і тоді робити спробу зливання.

Переважно простіше користуватися fetch та merge явно, оскільки магічний git pull часом може збивати з пантелику.

**Видаляння віддалених гілок**

Уявіть, що ви закінчили з віддаленою гілкою — тобто всі співпрацівники завершили свій вклад у нову функціональність та гілку було злито з віддаленою master (чи яка там у вас стабільна лінія коду). Для видалення віддаленої гілки користуйтеся опцією --delete до git push. Щоб видалити вашу serverfix виконайте таке:

$ git push origin --delete serverfix

To https://github.com/schacon/simplegit

- [deleted] serverfix

Все, що відбулося, це видалення вказівника на сервері. Git сервер триматиме самі дані ще деякий час, до наступного збирання сміття (garbage collection), тому, якщо щось було видалено випадково, часто це досить легко відновити.