?????? ? ????????? ?????????????

Для того, чтобы внести вклад в какой-либо Git-проект, вам необходимо уметь работать с удалёнными репозиториями. Удалённые репозитории представляют собой версии вашего проекта, сохранённые в интернете или ещё где-то в сети. У вас может быть несколько удалённых репозиториев, каждый из которых может быть доступен для чтения или для чтения-записи. Взаимодействие с другими пользователями предполагает управление удалёнными репозиториями, а также отправку и получение данных из них. Управление репозиториями включает в себя как умение добавлять новые, так и умение удалять устаревшие репозитории, а также умение управлять различными удалёнными ветками, объявлять их отслеживаемыми или нет и так далее. В данном разделе мы рассмотрим некоторые из этих навыков.

> Удалённый репозиторий может находиться на вашем локальном компьютере. Вполне возможно, что удалённый репозиторий будет находиться на том же компьютере, на котором работаете вы. Слово «удалённый» не означает, что репозиторий обязательно должен быть где-то в сети или Интернет, а значит только — где-то ещё. Работа с таким удалённым репозиторием подразумевает выполнение стандартных операций отправки и получения, как и с любым другим удалённым репозиторием.

???????? ????????? ????????????

Для того, чтобы просмотреть список настроенных удалённых репозиториев, вы можете запустить команду `git remote`. Она выведет названия доступных удалённых репозиториев. Если вы клонировали репозиторий, то увидите как минимум `origin` — имя по умолчанию, которое Git даёт серверу, с которого производилось клонирование:

```
$ git clone https://github.com/schacon/ticgit
Cloning into 'ticgit'...
remote: Reusing existing pack: 1857, done.
remote: Total 1857 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (1857/1857), 374.35 KiB | 268.00 KiB/s, done.
Resolving deltas: 100% (772/772), done.
Checking connectivity... done.
$ cd ticgit
$ git remote
```

```
origin
```

Вы можете также указать ключ `-v`, чтобы просмотреть адреса для чтения и записи, привязанные к репозиторию:

```
$ git remote -v
origin	https://github.com/schacon/ticgit (fetch)
origin	https://github.com/schacon/ticgit (push)
```

Если у вас больше одного удалённого репозитория, команда выведет их все. Например, для репозитория с несколькими настроенными удалёнными репозиториями в случае совместной работы нескольких пользователей, вывод команды может выглядеть примерно так:

```
$ cd grit
$ git remote -v
bakkdoor  https://github.com/bakkdoor/grit (fetch)
bakkdoor  https://github.com/bakkdoor/grit (push)
cho45     https://github.com/cho45/grit (fetch)
cho45     https://github.com/cho45/grit (push)
defunkt   https://github.com/defunkt/grit (fetch)
defunkt   https://github.com/defunkt/grit (push)
koke      git://github.com/koke/grit.git (fetch)
koke      git://github.com/koke/grit.git (push)
origin    git@github.com:mojombo/grit.git (fetch)
origin    git@github.com:mojombo/grit.git (push)
```

Это означает, что мы можем легко получить изменения от любого из этих пользователей. Возможно, что некоторые из репозиториев доступны для записи и в них можно отправлять свои изменения, хотя вывод команды не даёт никакой информации о правах доступа.

Обратите внимание на разнообразие протоколов, используемых при указании адреса удалённого репозитория; подробнее мы рассмотрим протоколы в разделе Установка Git на сервер главы 4.

?????????? ????????? ????????????

В предыдущих разделах мы уже упоминали и приводили примеры добавления удалённых репозиториев, сейчас рассмотрим эту операцию подробнее. Для того, чтобы добавить удалённый репозиторий и присвоить ему имя (shortname), просто выполните команду `git remote add <shortname> <url>`:

```
$ git remote
origin
$ git remote add pb https://github.com/paulboone/ticgit
$ git remote -v
origin https://github.com/schacon/ticgit (fetch)
origin https://github.com/schacon/ticgit (push)
pb https://github.com/paulboone/ticgit (fetch)
pb https://github.com/paulboone/ticgit (push)
```

Теперь вместо указания полного пути вы можете использовать `pb`. Например, если вы хотите получить изменения, которые есть у Пола, но нету у вас, вы можете выполнить команду `git fetch pb`:

```
$ git fetch pb
remote: Counting objects: 43, done.
remote: Compressing objects: 100% (36/36), done.
remote: Total 43 (delta 10), reused 31 (delta 5)
Unpacking objects: 100% (43/43), done.
From https://github.com/paulboone/ticgit
 * [new branch]      master     -> pb/master
 * [new branch]      ticgit     -> pb/ticgit
```

Ветка `master` из репозитория Пола сейчас доступна вам под именем `pb/master`. Вы можете слить её с одной из ваших веток или переключить на неё локальную ветку, чтобы просмотреть содержимое ветки Пола. Более подробно работа с ветками рассмотрена в главе Ветвление в Git.

???????? ? ????????? ???????????? ????????????

После добавления репозитория устанавливается два способа взаимодействия с ним - получение изменений и их отправка. Для отправки изменений в несколько репозиториев, необходимо выполнить команду.

```
$ git remote set-url --add --push origin <origin_repository_url>
$ git remote set-url --add --push origin <additional_repository_url>
```

где `origin` - имя репозитория, в который необходимо добавить возможность отправки изменений, `origin_repository_url` - исходный репозиторий, `additional_repository_url` - дополнительный репозиторий.

????????? ????????? ?? ??????????? ??????????? — Fetch ? Pull

Как вы только что узнали, для получения данных из удалённых проектов, следует выполнить:

```
$ git fetch [remote-name]
```

Данная команда связывается с указанным удалённым проектом и забирает все те данные проекта, которых у вас ещё нет. После того как вы выполнили команду, у вас должны появиться ссылки на все ветки из этого удалённого проекта, которые вы можете просмотреть или слить в любой момент.

Когда вы клонируете репозиторий, команда `clone` автоматически добавляет этот удалённый репозиторий под именем «origin». Таким образом, `git fetch origin` извлекает все наработки, отправленные на этот сервер после того, как вы его клонировали (или получили изменения с помощью fetch). Важно отметить, что команда `git fetch` забирает данные в ваш локальный репозиторий, но не сливает их с какими-либо вашими наработками и не модифицирует то, над чем вы работаете в данный момент. Вам необходимо вручную слить эти данные с вашими, когда вы будете готовы.

Если ветка настроена на отслеживание удалённой ветки (см. следующий раздел и главу Ветвление в Git чтобы получить больше информации), то вы можете использовать команду `git pull` чтобы автоматически получить изменения из удалённой ветки и слить их со своей текущей. Этот способ может для вас оказаться более простым или более удобным. К тому же, по умолчанию команда `git clone` автоматически настраивает вашу локальную ветку `master` на отслеживание удалённой ветки `master` на сервере, с которого вы клонировали репозиторий. Название веток может быть другим и зависит от ветки по умолчанию на сервере. Выполнение `git pull`, как правило, извлекает (fetch) данные с сервера, с которого вы изначально клонировали, и автоматически пытается слить (merge) их с кодом, над которым вы в данный момент работаете.

> Начиная с версии 2.27, команда `git pull` выдаёт предупреждение, если настройка `pull.rebase` не установлена. Git будет выводить это предупреждение каждый раз пока настройка не будет установлена. Если хотите использовать поведение Git по умолчанию (простое смещение вперёд если возможно — иначе создание коммита слияния): `git config --global pull.rebase "false"`. Если хотите использовать перебазирование при получении изменений: `git config --global pull.rebase "true"`.

???????? ????????? ? ????????? ??????????? (Push)

Когда вы хотите поделиться своими наработками, вам необходимо отправить их в удалённый репозиторий. Команда для этого действия простая: `git push <remote-name> <branch-name>`. Чтобы отправить вашу ветку `master` на сервер `origin` (повторимся, что клонирование обычно настраивает оба этих имени автоматически), вы можете выполнить следующую команду для отправки ваших коммитов:

```
$ git push origin master
```

Эта команда срабатывает только в случае, если вы клонировали с сервера, на котором у вас есть права на запись, и если никто другой с тех пор не выполнял команду `push`. Если вы и кто-то ещё одновременно клонируете, затем он выполняет команду `push`, а после него выполнить команду `push` попытаетесь вы, то ваш `push` точно будет отклонён. Вам придётся сначала получить изменения и объединить их с вашими и только после этого вам будет позволено выполнить `push`. Обратитесь к главе [Ветвление в Git]() для более подробного описания, как отправлять изменения на удалённый сервер.

???????? ??????????? ???????????

Если хотите получить побольше информации об одном из удалённых репозиториев, вы можете использовать команду `git remote show <remote>`. Выполнив эту команду с некоторым именем, например, `origin`, вы получите следующий результат:

```
$ git remote show origin
* remote origin
  Fetch URL: https://github.com/schacon/ticgit
  Push  URL: https://github.com/schacon/ticgit
  HEAD branch: master
  Remote branches:
    master                               tracked
    dev-branch                           tracked
  Local branch configured for 'git pull':
    master merges with remote master
  Local ref configured for 'git push':
    master pushes to master (up to date)
```

Она выдаёт URL удалённого репозитория, а также информацию об отслеживаемых ветках. Эта команда любезно сообщает вам, что если вы, находясь на ветке `master`, выполните `git pull`, ветка `master` с удалённого сервера будет автоматически влита в вашу сразу после

получения всех необходимых данных. Она также выдаёт список всех полученных ею ссылок.

Это был пример для простой ситуации и вы наверняка встречались с чем-то подобным. Однако, если вы используете Git более интенсивно, вы можете увидеть гораздо большее количество информации от `git remote show`:

```
$ git remote show origin
* remote origin
  URL: https://github.com/my-org/complex-project
  Fetch URL: https://github.com/my-org/complex-project
  Push  URL: https://github.com/my-org/complex-project
  HEAD branch: master
  Remote branches:
    master                           tracked
    dev-branch                       tracked
    markdown-strip                   tracked
    issue-43                         new (next fetch will store in remotes/origin)
    issue-45                         new (next fetch will store in remotes/origin)
    refs/remotes/origin/issue-11     stale (use 'git remote prune' to remove)
  Local branches configured for 'git pull':
    dev-branch merges with remote dev-branch
    master     merges with remote master
  Local refs configured for 'git push':
    dev-branch                     pushes to dev-branch                     (up to date)
    markdown-strip                 pushes to markdown-strip                 (up to date)
    master                         pushes to master                         (up to date)
```

Данная команда показывает какая именно локальная ветка будет отправлена на удалённый сервер по умолчанию при выполнении `git push`. Она также показывает, каких веток с удалённого сервера у вас ещё нет, какие ветки всё ещё есть у вас, но уже удалены на сервере, и для нескольких веток показано, какие удалённые ветки будут в них влиты при выполнении `git pull`.

???????? ? ??????????????? ????????? ??????????

Для переименования удалённого репозитория можно выполнить `git remote rename`. Например, если вы хотите переименовать `pb` в `paul`, вы можете это сделать при помощи `git remote rename`:

```
$ git remote rename pb paul
$ git remote
origin
paul
```

Стоит упомянуть, что это также изменит имена удалённых веток в вашем репозитории. То, к чему вы обращались как `pb/master`, теперь стало `paul/master`.

Если по какой-то причине вы хотите удалить удалённый репозиторий — вы сменили сервер или больше не используете определённое зеркало, или кто-то перестал вносить изменения — вы можете использовать `git remote rm`:

```
$ git remote remove paul
$ git remote
origin
```

При удалении ссылки на удалённый репозиторий все отслеживаемые ветки и настройки, связанные с этим репозиторием, так же будут удалены.

Revision #10
Created 2024-05-02 07:09:29 UTC by Антон Сергеевич Абраменко
Updated 2024-10-25 06:03:13 UTC by Антон Сергеевич Абраменко