

About

Documentation

- Reference

Book

Videos

External Links

Downloads

Community

This book is available in [English](#).

Full translation available in

- български език,

Εσπαñοl,

Français,

日本語,

한국어,

Nederlands,

Русский,

Українська

简体中文,

Partial translations available in

- Čeština,

Indonesian,

Polski,

Српски,

Tagalog,

繁體中文,

Translations started for

- Беларуская,

Deutsch,

فارسی

Ελληνικά,

Italiano,

Македонски,

Bahasa Melayu,

Polski,

Portugués (Brasil),

Türkçe,

Ўзбекча.

The source of this book is [hosted on GitHub](#).
Patches, suggestions and comments are welcome.

Chapters ▾1st Edition

.5 Основы Git - Работа с удалёнными репозиториями

Работа с удалёнными репозиториями

Чтобы иметь возможность совместной работы над каким-либо Git-проектом, необходимо знать, как управлять удалёнными репозиториями. Удалённые репозитории — это модификации проекта, которые хранятся в интернете или ещё где-то в сети. Их может быть несколько, каждый из которых, как правило, доступен для вас либо только на чтение, либо на чтение и запись. Совместная работа включает в себя управление удалёнными репозиториями и помещение (push) и получение (pull) данных в и из них тогда, когда нужно обменяться результатами работы. Управление удалёнными репозиториями включает умение добавлять удалённые репозитории, удалять те из них, которые больше не действуют, умение управлять различными удалёнными ветками и определять их как отслеживаемые (tracked) или нет и прочее. Данный раздел охватывает все перечисленные навыки по управлению удалёнными репозиториями.

Отображение удалённых репозиториев

Чтобы посмотреть, какие удалённые серверы у вас уже настроены, следует выполнить команду `git remote`. Она перечисляет список имён-сокращений для всех уже указанных удалённых дескрипторов. Если вы клонировали ваш репозиторий, у вас должен отобразиться, по крайней мере, origin — это имя по умолчанию, которое Git присваивает серверу, с которого вы клонировали:

```
$ git clone git://github.com/schacon/ticgit.git
Initialized empty Git repository in /private/tmp/ticgit/.git/
remote: Counting objects: 595, done.
remote: Compressing objects: 100% (269/269), done.
remote: Total 595 (delta 255), reused 589 (delta 253)
Receiving objects: 100% (595/595), 73.31 KiB | 1 KiB/s, done.
Resolving deltas: 100% (255/255), done.
$ cd ticgit
$ git remote
origin
```

Чтобы посмотреть, какому URL соответствует сокращённое имя в Git, можно указать команде опцию `-v`:

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

Если у вас больше одного удалённого репозитория, команда покажет их все. Например, мой репозиторий Grit выглядит следующим образом.

```
$ cd grit
$ git remote -v
bakdoor git://github.com/bakdoor/grit.git
cho45 git://github.com/cho45/grit.git
defunkt git://github.com/defunkt/grit.git
koke git://github.com/koke/grit.git
origin git@github.com:mojombo/grit.git
```

Это означает, что мы легко можем получить изменения от любого из этих пользователей. Но, заметьте, что origin — это единственный удалённый сервер прописанный как SSH-ссылка, поэтому он единственный, в который я могу помещать свои изменения (это будет рассмотрено в главе 4).

Добавление удалённых репозиториев

В предыдущих разделах мы упомянули и немного продемонстрировали добавление удалённых репозиториев, сейчас мы рассмотрим это более детально. Чтобы добавить новый удалённый Git-репозиторий под именем-сокращением, к которому будет проще обращаться, выполните `git remote add [сокращение] [url]`:

```
$ git remote
origin
$ git remote add pb git://github.com/paulboone/ticgit.git
$ git remote -v
origin git://github.com/schacon/ticgit.git
pb git://github.com/paulboone/ticgit.git
```

Теперь вы можете использовать в командной строке имя pb вместо полного URL. Например, если вы хотите извлечь (fetch) всю информацию, которая есть в репозитории Павла, но нет в вашем, вы можете выполнить `git fetch pb`:

```
$ git fetch pb
remote: Counting objects: 58, done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 44 (delta 24), reused 1 (delta 0)
Unpacking objects: 100% (44/44), done.
From git://github.com/paulboone/ticgit
 * [new branch] master -> pb/master
 * [new branch] ticgit -> pb/ticgit
```

Ветка master Павла теперь доступна локально как `pb/master`. Вы можете слить (merge) её в одну из своих веток или перейти на эту ветку, если хотите её проверить.

Fetch и Pull

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

```
$ git fetch [имя удал. сервера]
```

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. Теперь эти ветки в любой момент могут быть просмотрены или слиты. (В главе 3 мы перейдём к более детальному рассмотрению, что такое ветки и как их использовать.)

Когда вы клонируете репозиторий, команда clone автоматически добавляет этот удалённый репозиторий под именем origin. Таким образом, `git fetch origin` извлекает все наработки, отправленные (push) на этот сервер после того, как вы клонировали его (или получили изменения с помощью fetch). Важно отметить, что команда `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 у вас есть ветка, настроенная на отслеживание удалённой ветки (для дополнительной информации смотри следующий раздел и главу 3), то вы можете использовать команду `git pull`. Она автоматически извлекает и затем сливает данные из удалённой ветки в вашу текущую ветку. 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 (подразумевается, что на удалённом сервере есть ветка master). 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.

Push

Когда вы хотите поделиться своими наработками, вам необходимо отправить (push) их в главный репозиторий. Команда для этого действия проста: `git push [удал. сервер] [ветка]`. Чтобы отправить вашу ветку master на сервер `origin` (повторимся, что клонирование, как правило, настраивает оба этих имени автоматически), вы можете выполнить следующую команду для отправки наработок на сервер:

```
$ 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 затем команду push выполняете вы, то ваш push точно будет отклонён. Вам придётся сначала вытянуть (pull) их изменения и объединить с вашими. Только после этого вам будет позволено выполнить push. Смотри главу 3 для более подробного описания, как отправлять (push) данные на удалённый сервер.

Инспекция удалённого репозитория

Если хотите получить побольше информации об одном из удалённых репозиториев, вы можете использовать команду `git remote show [удал. сервер]`. Если вы выполните эту команду с некоторым именем, например, `origin`, вы получите что-то подобное:

```
$ git remote show origin
* remote origin
  URL: git://github.com/schacon/ticgit.git
  Remote branch merged with 'git pull' while on branch master
  master
  Tracked remote branches
  master
  ticgit
```

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 наверняка вы встретились 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: git@github.com:defunkt/github.git
  Remote branch merged with 'git pull' while on branch issues
  issues
  Remote branch merged with 'git pull' while on branch master
  master
  New remote branches (next fetch will store in remotes/origin)
  caching
  Stale tracking branches (use 'git remote prune')
  libwalker
  walker2
  Tracked remote branches
  acl
  apiv2
  dashboard2
  issues
  master
  postgres
  Local branch pushed with 'git push'
  master:master
```

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`.

Удаление и переименование удалённых репозиториев

Для переименования ссылок в новых версиях Git'a можно выолнить `git remote rename`, это изменит сокращённое имя, используемое для удалённого репозитория. Например, если вы хотите переименовать `pb` в `paul`, вы можете сделать это следующим образом:

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

Стоит упомянуть, что это также меняет для вас имена удалённых веток. То, к чему вы обращались как `pb/master`, стало `paul/master`.

Если по какой-то причине вы хотите удалить ссылку (вы сменили сервер или больше не используете определённое зеркало, или, возможно, контрибьютор перестал быть активным), вы можете использовать `git remote rm`:

```
$ git remote rm paul
$ git remote
origin
```

[prev](#) | [next](#)