УДК 004.4
ББК 32.973.2-018.2
П84

Последнюю версию этой книги на английском языке можно скачать с сайта: https://goalkicker.com/GitBook. Пожалуйста, не стесняйтесь поделиться этим PDF-файлом с кем угодно бесплатно.

Книга Git® Notes for Professionals составлена на основе документации Stack Overflow (https://archive.org/details/documentation-dump.7z), содержание которой написано прекрасными людьми из Stack Overflow. ***Текстовые материалы публикуются на условиях Creative Commons BY-SA.*** *В конце книги указаны авторы, внесшие вклад в создание различных глав. Изображения могут являться объектами авторского права соответствующих владельцев, если не указано иное.*

Профессиональная работа с Git. — Москва : Издательство АСТ, 2024. —
П84 160 с. : ил. — (Программирование от экспертов).

ISBN 978-5-17-160274-1.

Каждый раздел данного издания базируется на топовых материалах сайта Stack Overflow, касающихся использования самой современной системы контроля версий Git, и представляет собой описание одной из ее ключевых команд со всеми сопутствующими параметрами. В качестве конкретных примеров рассмотрены практические случаи, с которыми лучшие разработчики программного обеспечения, а «по совместительству» члены сообщества программистов Stack Overflow, сталкивались в процессе длительного использования Git, и именно поэтому за каждым из разделов книги стоит опытный автор, глубоко разбирающийся в анализируемой теме. Многое из представленного здесь ранее не публиковалось в русскоязычных учебниках по Git, например, отдельное внимание уделяется особенностям работы с подмодулями, переносу в Git проектов из других систем контроля версий, а также более широким возможностям некоторых команд.

УДК 004.4
ББК 32.973.2-018.2

ISBN 978-5-17-160274-1

Перевод на русский язык: ООО «Интеджер».
Издание на русском языке: ООО «Издательство АСТ».

Содержание

Глава 1. Начало работы с Git

Дата выпуска версии	
2.13	2017-05-10
2.12	2017-02-24
2.11.1	2017-02-02
2.11	2016-11-29
2.10.2	2016-10-28
2.10	2016-09-02
2.9	2016-06-13
2.8	2016-03-28
2.7	2015-10-04
2.6	2015-09-28
2.5	2015-07-27
2.4	2015-04-30
2.3	2015-02-05
2.2	2014-11-26
2.1	2014-08-16
2.0	2014-05-28
1.9	2014-02-14
1.8.3	2013-05-24
1.8	2012-10-21
1.7.10	2012-04-06
1.7	2010-02-13
1.6.5	2009-10-10
1.6.3	2009-05-07
1.6	2008-08-17
1.5.3	2007-09-02
1.5	2007-02-14
1.4	2006-06-10
1.3	2006-04-18
1.2	2006-02-12
1.1	2006-01-08
1.0	2005-12-21
0.99	2005-07-11

Раздел 1.1. Создание первого репозитория, добавление и фиксация файлов

Для начала убедитесь, что у вас установлен Git. Для этого вызовите командную строку и введите в нее одну из следующих команд.

На всех операционных системах:

```
git --version
```

В UNIX-подобных операционных системах:

```
which git
```

Если после ввода команды вы в ответ ничего не получили или команда не была распознана, возможно, вам придется установить Git на свою операционую систему, загрузив и запустив программу установки. Исключительно понятные и простые инструкции по установке см. на домашней странице Git (https://git-scm.com/book/ru/v2/Введение-Установка-Git).

После установки Git настройте имя пользователя и адрес электронной почты. Сделайте это *перед* выполнением команды `commit` (фиксации).

После установки Git перейдите в каталог, который вы хотите поместить под контроль версий, и создайте пустой Git-репозиторий:

```
git init
```

При этом создается скрытая папка `.git`, содержащая всё необходимое для работы с Git. Далее следует проверить, какие файлы Git добавит в новый репозиторий; этот шаг требует особого внимания:

```
git status
```

Просмотрите полученный список файлов; вы можете указать Git, какие из них следует поместить в систему контроля версий (избегайте добавления файлов с конфиденциальной информацией, например с паролями, или файлов, которые просто загромождают репозиторий):

```
git add <имя файла/директории #1> <имя файла/директории #2> < ... >
```

Если файлы из списка должны быть доступны всем, кто имеет доступ к репозиторию, то можно добавить все, что находится в текущем каталоге и его подкаталогах, при помощи одной команды:

```
git add .
```

Это “поставит” все файлы под версионный контроль, подготовив их к фиксации при первом коммите.

Для файлов, которые никогда не должны попасть под версионный контроль, перед выполнением команды `add` создайте и отредактируйте файл с именем `.gitignore`.

Фиксация всех добавленных файлов вместе с сообщением о фиксации производится командой:

```
git commit -m “Initial commit”
```

При этом создается новый коммит с заданным сообщением. Коммит — это как команда сохранения в любой программе или словно моментальный снимок всего вашего проекта. Теперь вы можете выгрузить его в удаленный репозиторий, а позже вернуться к нему при необходимости.

Если опустить параметр `-m`, то по умолчанию откроется редактор, в котором можно отредактировать и сохранить сообщение о фиксации.

Добавление удаленного репозитория

Удаленные репозитории представляют собой версии проекта, хранимые в Интернете или где-то в сети. Их может быть несколько, и каждый в общем случае доступен вам только для чтения или же для чтения и записи. Вы должны уметь отправлять данные в удаленный репозиторий и извлекать их оттуда каждый раз, когда требуется обменяться результатами работы.

Для добавления нового удаленного репозитория воспользуйтесь командой `git remote add`, которую можно набрать в терминале, в каталоге, в котором хранится ваш репозиторий.

Команда `git remote add` принимает два аргумента:

1. Удаленное имя, например `origin`.
2. Удаленный URL, например `https://<ваш-git-service-address>/user/repo.git`.

```
git remote add origin https://<ваш-git-service-address>/owner/repository.git
```

Примечание: перед добавлением удаленного репозитория необходимо создать требуемый репозиторий в сервисе git. После добавления удаленного репозитория вы сможете выполнять команды push/pull для ваших коммитов.

Раздел 1.2. Клонирование репозитория

Команда `git clone` используется для копирования существующего Git-репозитория с сервера на локальную машину. Например, для клонирования проекта GitHub:

```
cd <путь, где вы хотите, чтобы клон создал каталог>
git clone https://github.com/username/projectname.git
```

Для клонирования проекта BitBucket:

```
cd <путь, где вы хотите, чтобы клон создал каталог>
git clone https://yourusername@bitbucket.org/username/projectname.git
```

В результате на локальной машине создается каталог с именем `projectname`, содержащий все файлы удаленного Git-репозитория. Сюда входят исходные файлы проекта, а также подкаталог `.git`, содержащий всю историю и конфигурацию проекта.

Для указания другого имени каталога, например `MyFolder`:

```
git clone https://github.com/username/projectname.git MyFolder
```

Если необходимо клонировать проект в текущий каталог:

```
git clone https://github.com/username/projectname.git .
```

Примечание:

1. При клонировании в указанный каталог он должен быть пустым или несуществующим.
2. Можно также использовать ssh-версию команды:

```
git clone git@github.com:username/projectname.git
```

Версия `https` и версия **`ssh`** эквивалентны. Однако некоторые хостинги, например GitHub, рекомендуют использовать `https`, а не **`ssh`**.

Раздел 1.3. Совместное использование кода

Для совместного использования кода создается репозиторий на удаленном сервере, куда копируется локальный репозиторий.

Для минимизации использования пространства на удаленном сервере создается “пустой” репозиторий, содержащий только объекты `.git` и не создающий рабочей копии в файловой системе. В качестве бонуса вы устанавливаете этот удаленный сервер как upstream-сервер, чтобы делиться обновлениями с другими программистами.

На удаленном сервере:

```
git init --bare /path/to/repo.git
```

На локальной машине:

```
git remote add origin ssh://username@server:/path/to/repo.git
```

(Обратите внимание, что `ssh:` — это лишь один из возможных способов доступа к удаленному хранилищу). Теперь скопируйте локальное хранилище в удаленное:

```
git push --set-upstream origin master
```

Добавление `--set-upstream` (или `-u`) создает ссылку на upstream (отслеживание), которая используется командами Git без аргументов, например **`git pull`**.

Раздел 1.4. Настройка имени пользователя и электронной почты

Вам необходимо указать, **кем** вы являетесь, ***до*** создания любого коммита. Это позволит коммитам иметь правильное имя автора и e-mail, связанные с ними. Данную информацию Git будет включать в каждую фиксируемую вами версию, и она обязательно включается во все создаваемые вами коммиты (зафиксированные данные).

Это не имеет никакого отношения к аутентификации при отправке в удаленный репозиторий (например, при отправке в удаленный репозиторий с использованием учетной записи GitHub, BitBucket или GitLab).

Чтобы сделать эту идентификацию доступной для *всех* репозиториев, используйте команду **`git config --global`**.

При этом настройка будет сохранена в пользовательском файле `.gitconfig`: например, `$HOME/.gitconfig` или для Windows, `%USERPROFILE%\.gitconfig`.

```
git config --global user.name "Ваше имя"
git config --global user.email mail@example.com
```

Чтобы объявить идентификатор для отдельного репозитория, используйте **`git config`** внутри него. При этом настройка будет храниться внутри отдельного репозитория, в файле `$GIT_DIR/config`. Например `/path/to/your/repo/.git/config`.

```
cd /path/to/my/repo
git config user.name "Ваш логин на работе"
git config user.email mail_at_work@example.com
```

Параметры, хранящиеся в файле конфигурации хранилища, будут иметь приоритет над глобальной конфигурацией при работе с этим хранилищем.

Совет: если у вас есть разные идентификаторы (один для открытого проекта, один на работе, один для частного репозитория, ...), и вы хотите всегда устанавливать правильный идентификатор для каждого репозитория, над которым работаете, необходимо:

- Удалить глобальный идентификатор:

```
git config --global --remove-section user.name
git config --global --remove-section user.email
```

Версия ≥ 2.8

- Чтобы заставить git искать ваш идентификатор только в настройках репозитория, а не в глобальной конфигурации:

```
git config --global user.useConfigOnly true
```

Таким образом, если вы забудете задать `user.name` и `user.email` для данного репозитория и попытаетесь сделать коммит, вы увидите:

```
no name was given and auto-detection is disabled
no email was given and auto-detection is disabled

# имя не указано и автоопределение отключено
# e-mail не указан и автоопределение отключено
```

Раздел 1.5. Настройка удаленного репозитория

В том случае, если вы просто клонировали форк (англ. fork) (например, какой-то проект с открытым исходным кодом на GitHub), у вас может не быть push-доступа к удаленному репозиторию, поэтому вам необходимо иметь как свой форк, так и возможность получить доступ к удаленному репозиторию.

Сначала проверьте имена удаленных репозиториев:

```
$ git remote -v
origin    https://github.com/myusername/repo.git (fetch)
origin    https://github.com/myusername/repo.git (push)
upstream  # здесь эта строка может быть, а может и не быть
```

Если `upstream` уже существует (это верно для *некоторых* версий Git), то вам необходимо задать URL (в настоящее время он пуст):

```
$ git remote set-url upstream https://github.com/projectusername/repo.git
```

Если upstream не существует или если вы также хотите добавить форк друга/коллеги (в настоящее время их не существует):

```
$ git remote add upstream https://github.com/projectusername/repo.git
$ git remote add dave https://github.com/dave/repo.git
```

Раздел 1.6. Получение справочной информации о команде

Для получения более подробной информации о любой команде git, то есть сведений о том, что делает команда, о доступных параметрах и другой документации, используйте параметр --help или команду **help.**

Например, чтобы получить всю доступную информацию о команде **git diff**, используйте:

```
git diff --help
git help diff
```

Аналогично, чтобы получить всю доступную информацию о команде status, используйте:

```
git status --help
git help status
```

Если вам нужна только быстрая справка, показывающая значение наиболее часто используемых флагов командной строки, используйте -h:

```
git checkout -h
```

Раздел 1.7. Настройка SSH для Git

Если вы работаете в среде ОС Windows, откройте Git Bash (https://git-for-windows.github.io/). Если вы работаете в среде Mac или Linux, откройте терминал. Прежде чем сгенерировать SSH-ключ, можно проверить, нет ли у вас существующих SSH-ключей.

Выведите на экран содержимое каталога **~/**.ssh командой :

```
$ ls -al ~/.ssh
# Перечисление всех файлов в каталоге ~/.ssh
```

Проверьте список каталогов на предмет наличия открытого SSH-ключа. По умолчанию имена файлов открытых ключей имеют один из следующих вариантов:

```
id_dsa.pub
id_ecdsa.pub
id_ed25519.pub
id_rsa.pub
```

Если вы нашли в списке существующую пару открытого и закрытого ключей, которую хотели бы использовать в своей учетной записи BitBucket, GitHub (или аналогичной), вы можете скопировать содержимое файла **id_***.pub.

Если не нашли, то можно создать новую пару открытого и закрытого ключей при помощи следующей команды:

```
$ ssh-keygen
```

Нажмите клавишу Enter или Return, чтобы принять расположение по умолчанию. Введите и повторно введите фразу-пароль при появлении запроса или просто нажмите клавишу Enter.

Убедитесь, что ваш SSH-ключ добавлен в ssh-agent. Запустите ssh-агент в фоновом режиме, если он еще не запущен:

```
$ eval "$(ssh-agent -s)"
```

Добавьте свой SSH-ключ в ssh-agent. Обратите внимание, что в команде необходимо заменить `id_rsa` на имя вашего файла, содержащего **закрытый ключ**:

```
$ ssh-add ~/.ssh/id_rsa
```

Если необходимо изменить протокол существующего репозитория с HTTPS на SSH, можно выполнить следующую команду:

```
$ git remote set-url origin ssh://git@bitbucket.server.com:7999/projects/
your_project.git
```

Для клонирования нового репозитория по SSH можно выполнить следующую команду:

```
$ git clone ssh://git@bitbucket.server.com:7999/projects/your_project.git
```

Раздел 1.8. Установка Git

Давайте приступим к использованию Git. Прежде всего необходимо его установить. Это можно сделать несколькими способами, из них два основных — это либо скачать исходный код и самостоятельно его скомпилировать, либо установить из собранного пакета или другого установщика.

Установка на основе исходного кода

Если есть возможность, то лучше установить Git из исходного кода программы, так как в этом случае вы получите самую последнюю версию. Каждая версия Git, как правило, содержит полезные улучшения пользовательского интерфейса, поэтому установка последней версии часто является наилучшим вариантом в том случае, если вы чувствуете себя комфортно при компиляции программ из исходных текстов. Кроме того, многие дистрибутивы Linux содержат очень старые версии программы.

Поэтому, если вы не работаете с актуальным дистрибутивом или не используете бэкпорты, установка на основе компиляции исходного текста программы может оказаться наилучшим вариантом.

Для установки Git необходимо наличие следующих библиотек, от которых зависит Git: `curl, zlib, openssl, expat` и `libiconv`. Например, если вы работаете в системе с `yum` (например Fedora) или `apt-get` (например, в системе на базе Debian), вы можете использовать одну из этих команд для установки всех зависимостей:

```
$ yum install curl-devel expat-devel gettext-devel \
  openssl-devel zlib-devel
$ apt-get install libcurl4-gnutls-dev libexpat1-dev gettext \
  libz-dev libssl-dev
```

После того как установлены все необходимые зависимости, можно приступать к скачиванию исходного текста программы с сайта Git: http://git-scm.com/download, затем скомпилировать на его основе исполняемый файл и установить:

```
$ tar -zxf git-1.7.2.2.tar.gz
$ cd git-1.7.2.2
$ make prefix=/usr/local all
$ sudo make prefix=/usr/local install
```

После того как вы все это проделали, можно получать очередные обновления Git через сам Git:

```
$ git clone git://git.kernel.org/pub/scm/git/git.git
```

Установка в Linux

Если вы хотите установить Git в Linux с помощью установщика бинарных пакетов, в общем случае можно воспользоваться инструментом управления пакетами, входящим в имеющийся у вас дистрибутив. Если вы работаете с Fedora, вы можете воспользоваться утилитой `yum`:

```
$ yum install git
```

Если же вы используете дистрибутив на базе Debian, например Ubuntu, попробуйте воспользоваться утилитой `apt-get`:

```
$ apt-get install git
```

Установка в Mac

Существует три простых способа установки Git на Mac. Самый простой — использовать графический инсталлятор Git, который можно загрузить со страницы SourceForge: http://sourceforge.net/projects/git-osx-installer/.

Другой основной способ — установка Git через MacPorts (http://www.macports.org). Если у вас установлен MacPorts, установите Git с помощью команды:

```
$ sudo port install git +svn +doc +bash_completion +gitweb
```

Не обязательно добавлять все дополнения, но, вероятно, стоит включить `+svn` на случай, если вам когда-нибудь придется использовать Git с репозиториями Subversion (см. главу 8).

Homebrew (http://brew.sh/) является еще одной альтернативой для установки Git. Если у вас установлен Homebrew, установите Git с помощью команды:

```
$ brew install git
```

Установка под Windows

Установка Git под Windows очень проста. Проект msysGit имеет одну из самых простых процедур установки. Достаточно загрузить exe-файл установщика со страницы GitHub и запустить его:

```
http://msysgit.github.io
```

После его установки вы получаете как версию для командной строки (включая SSH-клиент, который пригодится вам в дальнейшем), так и стандартный графический интерфейс.

Примечание по работе с Windows: для работы с Git следует использовать поставляемую оболочку msysGit (стиль Unix), которая позволяет использовать сложные строки команд, приведенные в этой книге. Если по каким-либо причинам необходимо использовать родную оболочку Windows / консоль командной строки, то вместо одинарных кавычек следует использовать двойные (для параметров с пробелами), а также заключать в кавычки параметры, оканчивающиеся знаком (^), если они последние в строке, так как в Windows он является символом продолжения.

Глава 2. Просмотр истории версий

Параметр	Пояснение
`-q, --quiet`	Тихий, подавляет вывод diff
`--source`	Показывает источник фиксации
`--use-mailmap`	Использование файла почтовой карты (изменяет информацию о пользователе для фиксации пользователя)
`--decorate[=...]`	Параметры оформления вывода
`--L <n,m:file>`	Вывести журнал истории версий для определенного диапазона строк в конкретном файле, считая от 1. Начинается со строки n, продолжается до строки m. Также показан diff

`--show-signature`	Отображение подписей подписанных коммитов
`-i, --regexp-ignore-case`	Сопоставлять шаблоны ограничения регулярных выражений без учета регистра букв

Раздел 2.1. “Обычный” журнал истории версий Git

```
git log
```

отобразит все ваши коммиты с указанием автора и хеша. Информация выводится в нескольких строках для каждого коммита. (Если вы хотите отображать одну строку для каждого коммита, следует обратить внимание на параметр `oneline`). Для выхода из журнала нажмите клавишу `q`.

По умолчанию при отсутствии параметров команда `git log` выводит в обратном хронологическом порядке список сохраненных в данный репозиторий версий. То есть первыми отображаются самые свежие коммиты. Как видите, рядом с каждым коммитом указываются его контрольная сумма SHA-1, имя и электронная почта автора, дата создания и сообщение о фиксации (https://git-scm.com/book/en/v2/Git-Basics-Viewing-the-Commit-History).

Пример (из репозитория Free Code Camp (https://github.com/FreeCodeCamp/FreeCodeCamp)):

```
commit 87ef97f59e2a2f4dc425982f76f14a57d0900bcf
Merge: e50ff0d eb8b729
Автор: Brian
Date:    Thu Mar 24 15:52:07 2016 -0700

Merge pull request #7724 from BKinahan/fix/where-art-thou

Fix ‘its’ typo in Where Art Thou description

commit eb8b7298d516ea20a4aadb9797c7b6fd5af27ea5
Author: BKinahan
Date:     Thu Mar 24 21:11:36 2016 +0000

Fix ‘its’ typo in Where Art Thou description

commit e50ff0d249705f41f55cd435f317dcfd02590ee7
Merge: 6b01875 2652d04
Author: Mrugesh Mohapatra
Date:     Thu Mar 24 14:26:04 2016 +0530

Merge pull request #7718 from deathsythe47/fix/unnecessary-comma

Remove unnecessary comma from CONTRIBUTING.md
```

Если вы хотите ограничить команду выводом из журнала последних `n` коммитов, вы можете просто передать соответствующий параметр. Например, если требуется вывести из журнала список, включающий два последних коммита, воспользуйтесь командой:

```
git log -2
```

Раздел 2.2. Более красивое представление журнала

Чтобы увидеть журнал в виде более красивой графоподобной структуры, используйте:

```
git log --decorate --oneline --graph
```

Образец вывода:

```
* e0c1cea (HEAD -> maint, tag: v2.9.3, origin/maint) Git 2.9.3
*    9b601ea Merge branch ‘jk/difftool-in-subdir’ into maint
|\
| *  32b8c58 difftool: use Git::* functions instead of passing around state
```