

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. Много 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

<b>Глава 1. Начало работы с Git</b> .....	9
Раздел 1.1. Создание первого репозитория, добавление и фиксация файлов.....	9
Раздел 1.2. Клонирование репозитория .....	11
Раздел 1.3. Совместное использование кода .....	11
Раздел 1.4. Настройка имени пользователя и электронной почты .....	11
Раздел 1.5. Настройка удаленного репозитория .....	12
Раздел 1.6. Получение справочной информации о команде .....	13
Раздел 1.7. Настройка SSH для Git .....	13
Раздел 1.8. Установка Git .....	14
<b>Глава 2. Просмотр истории версий</b> .....	15
Раздел 2.1. “Обычный” журнал истории версий Git .....	16
Раздел 2.2. Более красивое представление журнала .....	16
Раздел 2.3. Применение в журналах возможности выделения цветом .....	17
Раздел 2.4. Вывод записи журнала в одну строку .....	17
Раздел 2.5. Поиск в журнале .....	18
Раздел 2.6. Список всех материалов, сгруппированных по фамилии автора .....	18
Раздел 2.7. Поиск строки коммита в журнале версий .....	19
Раздел 2.8. Вывести из файла определенный пользователем диапазон строк (поиск в истории строк) .....	19
Раздел 2.9. Параметры, ограничивающие вывод команды git log .....	20
Раздел 2.10. Индексация файлов по частям .....	20
Раздел 2.11. Отображение списка закомиченных файлов .....	21
Раздел 2.12. Показать содержимое одного коммита .....	21
Раздел 2.13. Управление выводом коммитов между двумя ветвями .....	22
Раздел 2.14. Вывод коммитов построчно, с отображением имени автора коммита и времени, прошедшего с момента фиксации .....	22
<b>Глава 3. Работа с удаленными репозиториями</b> .....	22
Раздел 3.1. Удаление ветви из удаленного репозитория .....	22
Раздел 3.2. Изменение URL-адреса удаленного репозитория .....	22
Раздел 3.3. Вывести список существующих удаленных репозитория .....	23
Раздел 3.4. Удаление локальных копий ветвей, удаленных из удаленного репозитория .....	23
Раздел 3.5. Обновление данных локального репозитория в соответствии с данными удаленного репозитория .....	23
Раздел 3.6. Команда ls-remote .....	24
Раздел 3.7. Добавление нового удаленного репозитория .....	24
Раздел 3.8. Настройка Upstream для новой ветки .....	24
Раздел 3.9. Начало работы .....	24
Раздел 3.10. Переименование удаленного репозитория .....	24
Раздел 3.11. Вывести информацию о конкретном удаленном репозитории .....	25
Раздел 3.12. Установка URL для конкретного удаленного репозитория .....	25
Раздел 3.13. Вывод URL для определенного удаленного репозитория .....	25
Раздел 3.14. Заменить URL старого удаленного репозитория на новый .....	26
<b>Глава 4. Индексация изменений</b> .....	26
Раздел 4.1. Индексация всех изменений во всех файлах .....	26
Раздел 4.2. Отменить индексацию изменений в файлах .....	26
Раздел 4.3. Индексация изменений по частям .....	26
Раздел 4.4. Интерактивная индексация .....	27
Раздел 4.5. Вывести на экран проиндексированные изменения .....	28
Раздел 4.6. Проиндексировать отдельный файл .....	28
Раздел 4.7. Индексация удаленных файлов .....	28
<b>Глава 5. Исключение из отслеживания отдельных файлов и папок</b> .....	29
Раздел 5.1. Игнорирование файлов и каталогов с помощью файла .gitignore .....	29
Раздел 5.2. Проверка игнорирования файла .....	31
Раздел 5.3. Исключения в файле .gitignore .....	32
Раздел 5.4. Глобальный файл .gitignore .....	32
Раздел 5.5. Как игнорировать файлы, которые уже были зафиксированы в Git-репозитории .....	33
Раздел 5.6. Игнорирование файлов локально без фиксации правил игнорирования .....	34
Раздел 5.7. Игнорирование последующих изменений файла (без его удаления) .....	34
Раздел 5.8. Игнорирование файла в любом выбранном каталоге .....	34
Раздел 5.9. Наборы готовых шаблонов .gitignore .....	35
Раздел 5.10. Игнорирование файлов во вложенных папках (несколько файлов .gitignore) .....	35
Раздел 5.11. Создание и регистрация пустой папки в Git .....	36
Раздел 5.12. Поиск файлов, игнорируемых при помощи .gitignore .....	36
Раздел 5.13. Игнорирование только части файла [stb] .....	37

Раздел 5.14. Игнорирование изменений в отслеживаемых файлах [stub].....	38
Раздел 5.15. Удаление уже зафиксированных в истории версий файлов, включенных в файл .gitignore .....	38
<b>Глава 6. Утилита сравнения (diff) .....</b>	<b>39</b>
Раздел 6.1. Просмотр разницы между непроиндексированными изменениями в рабочей ветви и внесенными в индекс .....	40
Раздел 6.2. Просмотр разницы между коммитами .....	40
Раздел 6.3. Показать различия между проиндексированными файлами .....	40
Раздел 6.4. Сравнение ветвей .....	40
Раздел 6.5. Отобразить разницу между всеми проиндексированными и непроиндексированными изменениями .....	41
Раздел 6.6. Показать различия для конкретного файла или каталога .....	41
Раздел 6.7. Просмотр в режиме word-diff для длинных строк .....	41
Раздел 6.8. Показать различия между текущей и последней версиями ветви .....	42
Раздел 6.9. Создание дельты (diff), совместимой с патчем .....	42
Раздел 6.10. Отображение разницы между двумя коммитами или двумя ветвями .....	42
Раздел 6.11. Использование параметра meld для просмотра всех модификаций в рабочем каталоге .....	43
Раздел 6.12. Команда Diff для текстовых и бинарных файлов plist в кодировке UTF-16 .....	43
<b>Глава 7. Отмена изменений .....</b>	<b>43</b>
Раздел 7.1. Возврат к предыдущему коммиту .....	43
Раздел 7.2. Отмена изменений .....	44
Раздел 7.3. Использование reflog .....	44
Раздел 7.4. Отмена слияния .....	45
Раздел 7.5. Отмена нескольких выполненных коммитов .....	47
Раздел 7.6. Отмена/переработка серии коммитов .....	47
<b>Глава 8. Слияние .....</b>	<b>48</b>
Раздел 8.1. Автоматическое слияние .....	48
Раздел 8.2. Поиск всех ветвей без уникальных коммитов и объединенных с веткой master .....	48
Раздел 8.3. Остановка слияния .....	49
Раздел 8.4. Слияние с коммитом .....	49
Раздел 8.5. Сохранять информацию только об одной из ветвей сторон, участвующих в слиянии .....	49
Раздел 8.6. Слияние двух ветвей, одна из которых текущая .....	49
<b>Глава 9. Подмодули .....</b>	<b>49</b>
Раздел 9.1. Клонирование Git-репозитория с подмодулями .....	49
Раздел 9.2. Обновление подмодуля .....	50
Раздел 9.3. Добавление подмодуля .....	50
Раздел 9.4. Привязка отслеживания подмодуля к отслеживанию всей ветви .....	50
Раздел 9.5. Перемещение подмодуля .....	51
Раздел 9.6. Удаление подмодуля .....	52
<b>Глава 10. Фиксация изменений .....</b>	<b>52</b>
Раздел 10.1. Индексация и фиксация изменений .....	53
Раздел 10.2. Хорошие сообщения для коммита .....	54
Раздел 10.3. Коммиты с исправлениями .....	55
Раздел 10.4. Отправка коммита без принудительного открытия редактора .....	55
Раздел 10.5. Прямая фиксация изменений .....	56
Раздел 10.6. Выбор конкретных строк, которые должны быть проиндексированы для фиксации .....	56
Раздел 10.7. Создание пустого коммита .....	57
Раздел 10.8. Исполнение обязательств от имени другого лица .....	57
Раздел 10.9. Фиксация подписи GPG .....	57
Раздел 10.10. Фиксация изменений в определенных файлах .....	58
Раздел 10.11. Задать дату для коммита .....	58
Раздел 10.12. Изменение времени коммита .....	58
Раздел 10.13. Изменение автора коммита .....	59
<b>Глава 11. Псевдонимы .....</b>	<b>59</b>
Раздел 11.1. Простые псевдонимы .....	59
Раздел 11.2. Список / поиск существующих псевдонимов .....	59
Раздел 11.3. Расширенные псевдонимы .....	60
Раздел 11.4. Временное игнорирование отслеживаемых файлов .....	60
Раздел 11.5. Показать журнал версий с графом ветвей .....	61
Раздел 11.6. Отобразить файлы, содержащиеся в конфигурации .gitignore .....	61
Раздел 11.7. Обновление кода при сохранении линейной истории .....	62
Раздел 11.8. Отменить индексирование ранее проиндексированных файлов .....	62
<b>Глава 12. Перемещение (команда rebase) .....</b>	<b>62</b>
Раздел 12.1. Перемещение ветвей в локальном репозитории .....	63
Раздел 12.2. Перемещение: “наша” и “их”, локальная ветка и удаленная ветка .....	63
Раздел 12.3. Перемещение в интерактивном режиме .....	65
Раздел 12.4. Откат до начальной фиксации при помощи команды rebase .....	66

Раздел 12.5. Настройка autostash.....	66
Раздел 12.6. Тестирование всех коммитов во время процесса перемещения.....	66
Раздел 12.7. Перемещение перед окончательной подготовкой кода к релизу.....	67
Раздел 12.8. Как прерывать процедуру перемещения в интерактивном режиме .....	69
Раздел 12.9. Настройка команды git-pull для автоматического выполнения rebase вместо команды merge .....	70
Раздел 12.10. Отправка локальной ветки в удаленный репозиторий после процедуры перемещения .....	70
<b>Глава 13. Конфигурация .....</b>	<b>70</b>
Раздел 13.1. Настройка используемого редактора .....	71
Раздел 13.2. Автоматическое исправление опечаток .....	71
Раздел 13.3. Список и редактирование текущей конфигурации.....	71
Раздел 13.4. Имя пользователя и адрес электронной почты .....	72
Раздел 13.5. Множественные имена пользователей и адреса электронной почты.....	72
Раздел 13.6. Несколько конфигураций Git.....	73
Раздел 13.7. Настройка окончания строк.....	73
Раздел 13.8. Конфигурация только для единственной команды.....	74
Раздел 13.9. Настройка прокси-сервера.....	74
<b>Глава 14. Ветвление .....</b>	<b>74</b>
Раздел 14.1. Создание новой ветки и переход на нее.....	75
Раздел 14.2. Вывод ветвей в виде списка .....	76
Раздел 14.3. Удаление ветки из удаленного репозитория.....	76
Раздел 14.4. Быстрый переход к предыдущей ветке.....	77
Раздел 14.5. Переход на новую ветку, которая отслеживает ветку в удаленном репозитории.....	77
Раздел 14.6. Удаление ветки в локальном репозитории.....	77
Раздел 14.7. Создание обособленной ветки (то есть ветки, не имеющей ни одного предшествующего коммита).....	78
Раздел 14.8. Переименование ветки.....	78
Раздел 14.9. Поиск по веткам.....	78
Раздел 14.10. Передача данных из локальной ветки в ветку в удаленном репозитории.....	78
Раздел 14.11. Переместить указатель HEAD текущей ветки на произвольный коммит .....	79
<b>Глава 15. Rev-List.....</b>	<b>79</b>
Раздел 15.1. Список коммитов, содержащихся в master, но не содержащихся в origin/master .....	79
<b>Глава 16. Объединение.....</b>	<b>79</b>
Раздел 16.1. Объединить заданные коммиты без проведения процедуры перемещения.....	79
Раздел 16.2. Объединение коммитов во время слияния.....	80
Раздел 16.3. Объединение коммитов во время выполнения операции перемещения .....	80
Раздел 16.4. Автообъединение и внесение исправлений .....	81
Раздел 16.5. Автообъединение: фиксация кода, который необходимо объединить в общий коммит во время операции перемещения .....	82
<b>Глава 17. Команда cherry-pick .....</b>	<b>83</b>
Раздел 17.1. Копирование коммитов из одной ветки в другую .....	83
Раздел 17.2. Копирование нескольких коммитов из одной ветки в другую .....	84
Раздел 17.3. Проверка необходимости применения команды cherry-pick .....	84
Раздел 17.4. Поиск коммитов, которые еще не были добавлены в удаленный репозиторий .....	84
<b>Глава 18. Восстановление .....</b>	<b>85</b>
Раздел 18.1. Восстановление после сброса .....	85
Раздел 18.2. Восстановление из git-архива.....	85
Раздел 18.3. Восстановление утерянного коммита.....	86
Раздел 18.4. Восстановление удаленного файла после фиксации .....	86
Раздел 18.5. Восстановление предыдущей версии файла .....	86
Раздел 18.6. Восстановление удаленной ветки .....	86
<b>Глава 19. Git Clean .....</b>	<b>87</b>
Раздел 19.1. Интерактивная очистка.....	87
Раздел 19.2. Принудительное удаление неотслеживаемых файлов .....	87
Раздел 19.3. Удаление игнорируемых файлов .....	88
Раздел 19.4. Очистка всех неотслеживаемых каталогов .....	88
<b>Глава 20. Использование файла .gitattributes.....</b>	<b>88</b>
Раздел 20.1. Автоматическая нормализация окончания строк .....	88
Раздел 20.2. Идентификация двоичных файлов.....	88
Раздел 20.3. Готовые шаблоны .gitattribute .....	88
Раздел 20.4. Отключение нормализации окончания строк .....	89
<b>Глава 21. Сопоставление автора и его электронной почты в файле .mailmap .....</b>	<b>89</b>
Раздел 21.1. Объединение различных имен авторов коммитов и разных адресов их электронной почты под псевдонимами для правильного отображения сокращенного журнала.....	89
<b>Глава 22. Анализ моделей организации рабочего процесса .....</b>	<b>89</b>
Раздел 22.1. Централизованный документооборот.....	89
Раздел 22.2. Модель организации рабочего процесса Gitflow.....	90
Раздел 22.3. Модель организации рабочего процесса Feature Branch Workflow.....	92

Раздел 22.4. Поток GitHub.....	92
Раздел 22.5. Модель организации рабочего процесса Forking Workflow.....	93
<b>Глава 23. Извлечение кода из репозитория. Команда pull</b> .....	93
Раздел 23.1. Извлечение изменений в локальный репозиторий.....	94
Раздел 23.2. Обновление с учетом локальных изменений.....	94
Раздел 23.3. Извлечь с перезаписью локальной версии.....	95
Раздел 23.4. Извлечение кода из репозитория на сервере.....	95
Раздел 23.5. Сохранение линейной истории при извлечении.....	95
Раздел 23.6. Pull, “операция запрещена”.....	96
<b>Глава 24. Перехватчики в Git (Hook)</b> .....	96
Раздел 24.1. Перехватчик pre-push.....	96
Раздел 24.2. Проверка сборки Maven (или другой системы сборки) перед коммитом.....	97
Раздел 24.3. Автоматическое перенаправление команд push в другой репозиторий.....	98
Раздел 24.4. Перехватчик commit-msg.....	98
Раздел 24.5. Локальные перехватчики.....	99
Раздел 24.6. Перехватчик post-checkout.....	99
Раздел 24.7. Перехватчик post-commit.....	99
Раздел 24.8. Перехватчик post-receive.....	99
Раздел 24.9. Перехватчик pre-commit.....	99
Раздел 24.10. Перехватчик prepare-commit-msg.....	100
Раздел 24.11. Перехватчик pre-rebase.....	100
Раздел 24.12. Перехватчик pre-receive.....	100
Раздел 24.13. Перехватчик update.....	100
<b>Глава 25. Клонирование репозитория</b> .....	101
Раздел 25.1. Поверхностное клонирование (shallow clone).....	101
Раздел 25.2. Регулярное клонирование.....	101
Раздел 25.3. Клонирование конкретной ветки.....	101
Раздел 25.4. Рекурсивное клонирование.....	102
Раздел 25.5. Клонирование с использованием прокси-сервера.....	102
<b>Глава 26. Прятанье</b> .....	102
Раздел 26.1. Что такое прятанье?.....	103
Раздел 26.2. Создание “тайников”.....	104
Раздел 26.3. Чтение записи из “тайника” с удалением.....	104
Раздел 26.4. Чтение записи из “тайника” без удаления.....	105
Раздел 26.5. Показать “тайник”.....	105
Раздел 26.6. Частичное сохранение в “тайнике”.....	105
Раздел 26.7. Вывести список сохраненных “тайников”.....	105
Раздел 26.8. Перенос незавершенной разработки в другую ветку.....	105
Раздел 26.9. Удаление “тайника”.....	106
Раздел 26.10. Проверить содержимое части “тайника”.....	106
Раздел 26.11. Восстановление ранее внесенных изменений из “тайника”.....	106
Раздел 26.12. Прятанье в интерактивном режиме.....	106
Раздел 26.13. Восстановление удаленного “тайника”.....	107
<b>Глава 27. Поддерживая</b> .....	108
Раздел 27.1. Создание, извлечение и обратная загрузка.....	108
<b>Глава 28. Переименование</b> .....	108
Раздел 28.1. Переименование папок.....	108
Раздел 28.2. Переименование локальной и удаленной веток.....	109
Раздел 28.3. Переименование локальной ветки.....	109
<b>Глава 29. Публикация кода</b> .....	109
Раздел 29.1. Отправка конкретного объекта в удаленную ветку.....	109
Раздел 29.2. Команда push.....	110
Раздел 29.3. Принудительная публикация.....	111
Раздел 29.4. Отправка тегов.....	111
Раздел 29.5. Настройка поведения команды push по умолчанию.....	112
<b>Глава 30. Внутренние компоненты</b> .....	112
Раздел 30.1. Репозиторий.....	112
Раздел 30.2. Объекты.....	112
Раздел 30.3. Ссылка HEAD.....	113
Раздел 30.4. Ссылки.....	113
Раздел 30.5. Коммит.....	113
Раздел 30.6. Объект типа “дерево” (Tree).....	114
Раздел 30.7. Объект Blob.....	114
Раздел 30.8. Создание новых коммитов.....	115
Раздел 30.9. Перемещение HEAD.....	115
Раздел 30.10. Перемещение ссылки HEAD.....	115
Раздел 30.11. Создание новых типов ссылок (Refs).....	115

<b>Глава 31. Плагин git-tfs</b> .....	116
Раздел 31.1. Клонирование в системе git-tfs .....	116
Раздел 31.2. Клонирование в системе git-tfs из “пустого” репозитория Git .....	116
Раздел 31.3. Установка плагина git-tfs через Chocolatey .....	116
Раздел 31.4. Индексация изменений в git-tfs командой Check In .....	116
Раздел 31.5. git-tfs push .....	116
<b>Глава 32. Пустые каталоги в Git</b> .....	117
Раздел 32.1. Git не отслеживает каталоги .....	117
<b>Глава 33. git-svn</b> .....	117
Раздел 33.1. Клонирование SVN-репозитория .....	117
Раздел 33.2. Передача локальных изменений в SVN .....	118
Раздел 33.3. Работа на локальном уровне .....	118
Раздел 33.4. Получение последних изменений из SVN .....	118
Раздел 33.5. Работа с пустыми папками .....	119
<b>Глава 34. Архив</b> .....	119
Раздел 34.1. Создание архива git-репозитория .....	120
Раздел 34.2. Создание архива git-репозитория с названием директории в качестве префикса .....	120
Раздел 34.3. Создание архива git-репозитория на основе определенной ветки, ревизии, тега или каталога .....	121
<b>Глава 35. Переписывание истории с помощью команды filter-branch</b> .....	121
Раздел 35.1. Изменение автора коммитов .....	121
Раздел 35.2. Задать одинаковые имена и для автора содержимого коммита, и для автора, оформившего коммит .....	121
<b>Глава 36. Переход на Git</b> .....	122
Раздел 36.1. SubGit .....	122
Раздел 36.2. Миграция с SVN на Git с помощью утилиты преобразования Atlassian .....	122
Раздел 36.3. Миграция с Mercurial на Git .....	123
Раздел 36.4. Переход с Team Foundation Version Control (TFVC) на Git .....	123
Раздел 36.5. Миграция с SVN на Git с помощью svn2git .....	124
<b>Глава 37. Команда git show</b> .....	125
Раздел 37.1. Обзор .....	125
<b>Глава 38. Разрешение конфликтов слияния</b> .....	125
Раздел 38.1. Разрешение конфликтов вручную .....	125
<b>Глава 39. Команда bundle</b> .....	126
Раздел 39.1. Создание пакета на локальной машине и его использование на другой .....	126
<b>Глава 40. Графическое отображение истории коммитов с помощью команды gitk</b> .....	127
Раздел 40.1. Отображение истории коммитов для одного файла .....	127
Раздел 40.2. Отображение всех коммитов между двумя любыми коммитами .....	127
Раздел 40.3. Отображение коммитов с момента появления тега версии .....	127
<b>Глава 41. Поиск коммита, в котором появился баг</b> .....	127
Раздел 41.1. Бинарный поиск коммита с ошибкой (git bisect) .....	127
Раздел 41.2. Полуавтоматический поиск коммита с ошибкой .....	128
<b>Глава 42. Применение команды blame</b> .....	129
Раздел 42.1. Вывести только определенные строки .....	129
Раздел 42.2. Как узнать, кто изменил файл .....	130
Раздел 42.3. Вывести на экран коммит, в котором осуществлялось последнее редактирование данной строки .....	130
Раздел 42.4. Игнорировать изменения, связанные с изменением пробельных символов .....	130
<b>Глава 43. Работа с версиями проекта в Git</b> .....	130
Раздел 43.1. Указание версии по имени объекта .....	130
Раздел 43.2. Символьные имена ссылок на ветви, метки, ветви в удаленном репозитории .....	130
Раздел 43.3. Версия по умолчанию: HEAD .....	131
Раздел 43.4. Ссылки на журнал ссылок reflog: <refname>@{<n>} .....	131
Раздел 43.5. Ссылки на журнал ссылок reflog: <refname>@{<date>} .....	131
Раздел 43.6. Отслеживаемая / удаленная ветвь: <branchname>@{upstream} .....	132
Раздел 43.7. Цепочка предыдущих версий: <rev>^, <rev>~<n> и т. д. .....	132
Раздел 43.8. Разыменование веток и тегов: <rev>^0, <rev>^<type> .....	133
Раздел 43.9. Самый младший совпадающий коммит: <rev>^</text>, </text> .....	133
<b>Глава 44. Рабочие деревья</b> .....	134
Раздел 44.1. Применение рабочего дерева .....	134
Раздел 44.2. Перемещение рабочего дерева .....	135
<b>Глава 45. Работа с удаленными объектами Git Remote</b> .....	135
Раздел 45.1. Отображение удаленных репозиторияев .....	136
Раздел 45.2. Изменение url удаленного репозитория Git .....	137
Раздел 45.3. Удаление удаленного репозитория .....	137
Раздел 45.4. Добавление удаленного репозитория .....	137
Раздел 45.5. Показать дополнительную информацию об удаленном репозитории .....	137

Раздел 45.6. Переименование удаленного репозитория.....	138
<b>Глава 46. Хранилище крупных файлов Git Large File Storage (LFS) .....</b>	<b>138</b>
Раздел 46.1. Вывести список определенных типов файлов для их внешнего хранения .....	138
Раздел 46.2. Установка конфигурации LFS для всех клонов.....	138
Раздел 46.3. Установка LFS.....	138
<b>Глава 47. Команда git patch .....</b>	<b>139</b>
Раздел 47.1. Создание патча .....	140
Раздел 47.2. Применение патчей .....	140
<b>Глава 48. Статистика в Git.....</b>	<b>141</b>
Раздел 48.1. Вывести строки кода, сгруппированные по разработчикам.....	141
Раздел 48.2. Вывести список всех ветвей с датой их последнего изменения.....	141
Раздел 48.3. Вывести коммиты, сгруппированные по авторам.....	142
Раздел 48.4. Сгруппировать коммиты по дате.....	142
Раздел 48.5. Вывести общее количество коммитов в ветке.....	142
Раздел 48.6. Вывод списка коммитов в удобном для восприятия формате.....	142
Раздел 48.7. Поиск всех локальных git-репозиториях на компьютере.....	142
Раздел 48.8. Вывод общего количества коммитов для каждого автора.....	143
<b>Глава 49. Команда git send-email .....</b>	<b>143</b>
Раздел 49.1. Использование команды git send-email с Gmail.....	143
Раздел 49.2. Формирование.....	143
Раздел 49.3. Отправка патчей по почте.....	144
<b>Глава 50. Графические клиенты Git.....</b>	<b>144</b>
Раздел 50.1. gitk и git-gui.....	144
Раздел 50.2. Рабочий стол GitHub.....	146
Раздел 50.3. Git Kraken.....	146
Раздел 50.4. SourceTree.....	146
Раздел 50.5. Расширения Git.....	146
Раздел 50.6. SmartGit.....	146
<b>Глава 51. Reflog — восстановление коммитов, не отображаемых в журнале Git .....</b>	<b>147</b>
Раздел 51.1. Восстановление после неудачного перемещения.....	147
<b>Глава 52. TortoiseGit .....</b>	<b>147</b>
Раздел 52.1. Объединение коммитов.....	147
Раздел 52.2. Игнорирование отслеживания файла .....	148
Раздел 52.3. Игнорирование файлов и папок.....	149
Раздел 52.4. Ветвление .....	150
<b>Глава 53. Внешнее слияние и инструменты difftools .....</b>	<b>150</b>
Раздел 53.1. Настройка утилиты KDiff3 в качестве инструмента слияния .....	150
Раздел 53.2. Настройка KDiff3 в качестве инструмента для проведения различий (создания дельты diff).....	151
Раздел 53.3. Настройка IDE IntelliJ в качестве инструмента слияния (Windows).....	151
Раздел 53.4. Настройка IDE IntelliJ в качестве инструмента для проведения диффа (Windows).....	151
Раздел 53.5. Настройка Beyond Compare.....	151
<b>Глава 54. Обновление имени объекта в ссылке.....</b>	<b>152</b>
Раздел 54.1. Обновление имени объекта в ссылке.....	152
<b>Глава 55. Как задать имя ветви в командной оболочке Bash под ОС Ubuntu .....</b>	<b>153</b>
Раздел 55.1. Обращение к имени ветки в терминале.....	153
<b>Глава 56. Перехватчики на стороне клиента Git .....</b>	<b>153</b>
Раздел 56.1. Скрипт-перехватчик pre-push.....	153
Раздел 56.2. Установка скрипта-перехватчика.....	154
<b>Глава 57. Команда git rerere.....</b>	<b>155</b>
Раздел 57.1. Включение rerere .....	155
<b>Глава 58. Переименование репозитория.....</b>	<b>155</b>
Раздел 58.1. Изменение локальных настроек.....	155
<b>Глава 59. Метки Git .....</b>	<b>155</b>
Раздел 59.1. Перечисление всех доступных тегов.....	155
Раздел 59.2. Создание и размещение тегов в Git .....	156
<b>Глава 60. Наведение порядка в локальном и удаленном хранилищах .....</b>	<b>156</b>
Раздел 60.1. Удаление локальных ветвей, которые были удалены в удаленном репозитории.....	156
<b>Глава 61. Команда diff-tree .....</b>	<b>157</b>
Раздел 61.1. Просмотр файлов, измененных коммитом.....	157
Раздел 61.2. Использование.....	157
Раздел 61.3. Общие опции diff .....	157
<b>Благодарности.....</b>	<b>158</b>

# 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 операционную 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* репозиториях, 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 репозиториях:

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

<code>--show-signature</code>	Отображение подписей подписанных коммитов
<code>-i, --regexp-ignore-case</code>	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 87ef97f59e2a2f4dc425982f76f14a57d0900bcb
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
```