O'REILLY®

bhv®

Head First

Git

Лучший способ понять Git изнутри

Раджу Ганди

Прокачай свой мозг!

Оглавление (краткое)

Оглавление (с пояснениями)

Вступление

Заставьте мозг принять Git. Сейчас вы пытаетесь чему-то научиться, в то время как ваш мозг делает вам одолжение, следя за тем, чтобы обучение не закрепилось. Ваш мозг думает: "Лучше оставить место для более важного, например, того, каких диких животных следует избегать и почему кататься голым на сноуборде — плохая идея". Так как же заставить свой мозг уверовать, что ваша жизнь зависит от знания Git?

Знакомство с Git

1

Приступим к делу

Нам нужен контроль версий! Каждый программный проект начинается с идеи, реализованной в исходном коде. Эти файлы — сердце наших приложений, поэтому следует относиться к ним с осторожностью. Мы должны быть уверены, что храним их в безопасности, сохраняем историю изменений и приписываем заслуги (или вину!) законным авторам. Мы также должны обеспечить комфортное сотрудничество между несколькими членами команды.

Вдобавок нам нужно, чтобы все это было в одном инструменте, который всегда под рукой, но не мешает нам и срабатывает только в нужный момент.

Существует ли такой волшебный инструмент? Наверное, вы догадываетесь, каков будет ответ. Конечно, это Git! Разработчики и организации по всему миру любят Git. Так что же делает Git таким популярным?

2 Ветвление кода
Свободный полет вашей мысли

Вы можете ходить и жевать резинку одновременно. Стародавние приверженцы Git скажут вам, откинувшись в шезлонге и потягивая крафтовый зеленый чай, что одним из самых больших преимуществ Git является легкость, с которой вы можете создавать ветки кода. Возможно, вам поручили создать новую опцию интерфейса, и пока вы над ней работаете, ваш менеджер просит вас исправить ошибку в производственной версии. Или, может быть, вы только что закончили вносить изменения в код, но внезапно пришло вдохновение, и вы придумали лучший способ реализации алгоритма. Ветки позволяют вам работать над несколькими, совершенно не связанными частями одной и той же кодовой базы в одно и то же время, независимо друг от друга. Давайте разберемся, как это устроено!

3 Осмотритесь вокруг

Изучим ваш репозиторий Git

Вы готовы копнуть глубже, Шерлок? Продолжая работать в Git, вы будете создавать ветки, делать коммиты и объединять свою работу в ветках интеграции. Каждый коммит — это шаг вперед, а история коммитов показывает, как вы туда попали. Время от времени вы будете оглядываться назад, чтобы увидеть, как вы оказались в текущем положении, или понять, как две ветки отошли друг от друга. Мы начнем эту главу с демонстрации того, как Git помогает визуализировать историю коммитов.

Просмотр истории коммитов важен, но Git также помогает увидеть, как изменился ваш репозиторий. Напомним, что коммиты представляют собой изменения, а ветки отражают последовательность изменений. Как узнать, где именно произошло изменение — между коммитами, между ветвями или даже между вашим рабочим каталогом, индексом и базой данных объектов? Это другая тема данной главы.

Вместе мы проделаем очень интересную детективную работу и разберемся, как устроен Git. Итак, давайте прокачаем навыки расследования!

Отмена действий

4 Исправляем свои ошибки

Мы все делаем ошибки, верно? Люди совершают ошибки с незапамятных времен, и долгое время ошибки обходились нам довольно дорого (в эпоху перфокарт и пишущих машинок нам приходилось все переделывать). Причина очевидна — тогда у нас не было системы контроля версий. Но теперь она есть! Git предоставляет широкие возможности для легкого и безболезненного исправления ошибок. Если вы случайно добавили файл в индекс, допустили опечатку в сообщении коммита или сделали неправильный коммит, Git раскрывает перед вами множество рычагов и кнопок, которыми можно воспользоваться, чтобы никто никогда не узнал о вашем... кхм... промахе.

Прочитав эту главу, вы точно будете знать, что делать — независимо от того какую ошибку вы допустили. Итак, давайте сделаем несколько ошибок и научимся их исправлять.

Командная работа с Git — часть I

5 Удаленная работа

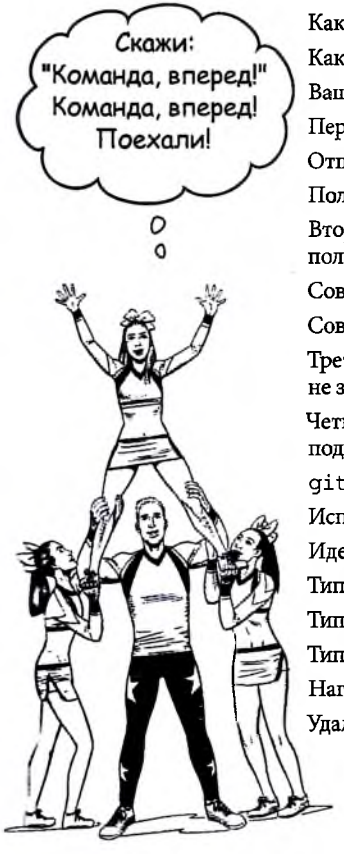
Работа в одиночестве может быстро надоесть. В этой книге вы многое узнали о том, как устроен Git и как работать с его репозиториями. Вместе с вами мы использовали репозитории, которые инициализировали локально с помощью команды `git init`. Мы проделали большую работу — создали ветки, объединили их и использовали утилиты Git, такие как команды `git log` и `git diff`, чтобы отследить развитие репозитория. Но большинство проектов не такие. Мы часто работаем в командах с друзьями или коллегами. Git предлагает очень мощную модель совместной работы, в которой мы все можем делиться своими достижениями, используя общий репозиторий. Все начинается с того, что наш репозиторий становится публичным, что делает историю коммитов проекта "общей". В общедоступном репозитории вы можете делать все, чему научились до сих пор, точно так же, как раньше (за некоторыми исключениями). Вы можете создавать ветки и коммиты и добавлять их в историю коммитов наравне с коллегами или друзьями. Так устроена командная работа в Git. Но прежде чем мы перейдем к командной работе, давайте вместе разберемся, как работают публичные репозитории. Вперед, друзья!

6 Совместная работа с Git — часть II

Команда, вперед!

Готовы собрать команду? Git — фантастический инструмент для совместной работы, и у меня возникла блестящая идея научить вас всему этому — читая эту главу, вы объединитесь с кем-то еще! Вы будете опираться на сведения, полученные в предыдущей главе. Вы знаете, что работа с распределенной системой, такой как Git, включает в себя множество компонентов. Итак, что же Git предлагает нам для упрощения этой задачи и что вам нужно помнить, когда вы начинаете совместную работу с коллегами? Существуют ли какие-то специальные функции, облегчающие совместную работу? Сейчас вы об этом узнаете.

На старт. Внимание. Клон!



7 Поиск в репозитории Git

Git идет искать!

Ваш проект и его история коммитов неизбежно будут расти. Время от времени вам придется искать в ваших файлах определенный фрагмент текста. Возможно, вы захотите узнать, кто изменил файл, когда он был изменен, а также выяснить, какой коммит его изменил. Git охотно поможет вам в этом.

А еще есть ваша история коммитов. Каждый коммит представляет собой изменение. Git позволяет вам искать не только каждый экземпляр фрагмента текста в вашем проекте, но и время его добавления (или удаления). Это облегчает поиск нужных сообщений коммитов. В довершение всего, иногда нужно найти коммит, который привел к ошибке или опечатке. Git предлагает специальную команду, которая позволяет вам быстро сосредоточиться на этом коммите.

Чего же мы ждем? Давайте попробуем найти что-нибудь в репозиториях Git!

8 Эффективная работа с Git #Советы профессионалов

До сих пор в этой книге вы учились использовать Git. Но вы также можете подчинить Git своей воле. Именно поэтому так важно изучить детали его настройки. Вы уже начали настраивать Git в предыдущих главах, а в этой главе мы рассмотрим гораздо больше настроек, облегчающих вашу жизнь. Вы научитесь также определять ярлыки: долой многословные команды Git!

Вы можете значительно упростить взаимодействие с Git. Я покажу, как заставить Git игнорировать определенные типы файлов, чтобы вы случайно не добавили их в коммит. Вы освоите продуманные способы написания сообщений коммитов и узнаете, как я люблю называть свои ветки. И в заключение мы рассмотрим, как графический пользовательский интерфейс Git может сыграть важную роль в вашем рабочем процессе. #поехали #недождусь

Приложение: остатки

Пять важных тем, о которых мы не говорили

Мы рассмотрели много разных тем, и вы почти закончили читать эту книгу. Мне будет вас не хватать, но прежде чем мы расстанемся, было бы неправильно отправить вас в мир без дополнительной подготовки. Git предлагает множество функций, и я не смог бы уместить их все в одной книге. Поэтому я приберег несколько действительно интересных описаний для этого приложения.