Конспект по Git / **"Что? ~~Где? Когда?"~~ Зачем? Как работает?"**

1. **Что такое система контроля версий и зачем она нужна?**

Система контроля версий (СКВ) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов.

1. **Как происходит инициализация git?**

С помощью команды $ git init в необходимой папке, которая становится репозиторием

1. **Какие изменения происходят при этом в файловой системе?**

Создается скрытая папка «.git», в которой находятся все файлы Git’a

1. **Из каких логических частей состоит git?**
2. Рабочий каталог
3. Область подготовленных файлов
4. Каталог Git’a

**Работа с коммитами.**

команда Git для записи индексированных изменений в репозиторий. Коммит будет включать текущие состояния индексированных файлов плюс последние сохраненные состояния неиндексированных (но отслеживаемых) файлов. Обратите внимание: коммит включает в себя не изменения (дельты, патчи) относительно предыдущего коммита, а "снимок" (англ. shapshot) текущего состояния рабочей области.

**команда git add .** (нужна для индексирования файлов перед коммитом)

Команда git add принимает параметром путь к файлу или каталогу. Если это каталог, команда рекурсивно добавляет (индексирует) все файлы в данном каталоге.

**Ветвление.**

Новая ветка: $ git branch <имя\_ветки>

Склонировать ветку: git clone --branch=branch-name http://whatever.git

Удалить ветку локально: $ git branch -d <имя\_ветки>

Удалить ветку в репозитории: $ git push origin --delete <имя\_ветки>

Переименовать ветку: git branch -m <oldname> <newname>

Переместить ветку: git branch -f <имя ветки> <куда переместить, можно указать хэш коммита или относительную ссылку, например, относительная ссылка HEAD~3 обозначает перемещение на три родителя назад от HEAD >

**Что такое режим Detaching HEAD?**

HEAD - это символическое имя текущего выбранного коммита — это, по сути, тот коммит, над которым мы в данный момент работаем. HEAD всегда указывает на последний коммит из локального дерева.

Обычно HEAD указывает на имя ветки.

Переходим к сути вопроса:

Отделение (detaching) HEAD означает лишь присвоение его не ветке, а конкретному коммиту с помощью того же checkout

**Слияние веток. Rebase и Merge.**

+3. В чем особенности работы rebase?

При ребейзе Git по сути копирует набор коммитов и переносит их в другое место. То есть можно взять все изменения, которые попали в коммиты на одной из веток, и повторить их на другой ветке.

находится общий предок для двух веток (на которой вы находитесь сейчас и на которую вы выполняете перемещение); для каждого из коммитов в текущей ветке берётся его дельта и сохраняется во временный файл; текущая ветка устанавливается на тот же коммит, что и ветка, на которую выполняется перемещение; и, наконец, одно за другим применяются все изменения.

Преимущество rebase в том, что c его помощью можно делать чистые и красивые линейные последовательности коммитов. История коммитов будет чище.

+4. Как работает интерактивный rebase?

вызывается диалог параметром '-i' или '--interactive' в команде 'git rebase'. Включается интерактивный режим выполнения операции ребазирования. Открывается редактор по умолчанию. Редактор даст вам одну линию на каждый коммит следующего формата:

(action) (partial-sha) (short commit message)

Далее можно изменить действие (по умолчанию 'pick'("выбрать")) на 'edit'("редактировать") или 'squash'("сдавить") или просто удалить строку. Вы можете также переупорядочить коммиты просто передвигая линии так как вам этого хочется. Затем, когда вы выйдете из редактора, git попытается уложить коммиты тем образом каким вы это определили и выполнить действия определенные вами.

+5. Каковы особенности merge?

Слияния в Git создают особый вид коммита, который имеет сразу двух родителей. Коммит с двумя родителями обычно означает, что мы хотим объединить изменения из одного коммита с другим коммитом и всеми их родительскими коммитами.

**Разрешение конфликтов**. Add . commit / rebase –continue/--skip

**Прятанье в git.**

Прятанье поглощает грязное состояние рабочего каталога, то есть изменённые отслеживаемые файлы и изменения в индексе, и сохраняет их в стек незавершённых изменений, которые вы потом в любое время можете снова применить.

Чтобы посмотреть, что у вас есть припрятанного, используйте git stash list:

Вы можете снова применить ту, которую только что спрятали, с помощью команды, показанной в справке в выводе первоначальной команды stash: git stash apply. Если вы хотите применить одну из старых заначек, можете сделать это, указав её имя так: git stash apply stash@{2}. Если не указывать ничего, Git будет подразумевать, что вы хотите применить последнюю спрятанную работу.

**Изменение истории: cherry-pick**

Команда git cherry-pick используется для того чтобы взять изменения, внесённые каким-либо коммитом, и попытаться применить их заново в виде нового коммита наверху текущей ветки. Это может оказаться полезным чтобы забрать парочку коммитов из другой ветки без полного слияния с той веткой.

Проще говоря, это очень простой и прямолинейный способ сказать, что ты хочешь копировать несколько коммитов на место, где сейчас находишься (HEAD).

**Не перемещайте коммиты, которые вы уже отправили в публичный репозиторий.**

Перемещение работает следующим образом: находится общий предок для двух веток (на которой вы находитесь сейчас и на которую вы выполняете перемещение); для каждого из коммитов в текущей ветке берётся его дельта и сохраняется во временный файл; текущая ветка устанавливается на тот же коммит, что и ветка, на которую выполняется перемещение; и, наконец, одно за другим применяются все изменения. Рисунок 3-29 иллюстрирует этот процесс.



вы можете вызвать интерактивный диалог передав параметр '-i' или '--interactive' в команду 'git rebase'. Это вызовет интерактивный режим выполнения операции ребазирования

Как только вы выполнили команду 'rebase -i', тут же откроется ваш редактор по умолчанию

это даст вам одну линию на каждый коммит следующего формата:

(action) (partial-sha) (short commit message)

Теперь, вы можете изменить действие (по умолчанию 'pick'-выбрать) на 'edit'-редактировать или 'squash'- сдавить или просто оставить это как 'pick'. Вы можете также переупорядочить коммиты просто передвигая линии так как вам этого хочется. Затем, когда вы выйдете из редактора, git попытается уложить коммиты тем образом каким вы это определили и выполнить действия определенные вами.

Если определен 'pick', то он просто попробует применить патч и сохранить коммит с тем же сообщением-описанием как и было до этого.

Если определено 'squash', то он будет комбинировать это коммит с предыдущим чтобы создать новый коммит. Вы опять попадете в редактор чтобы объединить оба сообщение-описания обоих коммитов сложив их вместе.

Если определен 'edit', то процесс будет идти также, но приостановится перед тем как двигаться к следующему вас выбросит в коммандую строку и вы сможете изменить коммит, или изменить содержимое коммита некоторым образом.

Если вы хотите разделить коммит, например, вы определите 'edit' для этого коммита:

И затем когда вы попадете в командную строку, вы вернетесь к этому коммиту и создадите два (или более) новых.

Последняя вещь которую интерактивное выполнение ребазирования может делать для вас - это сбрасывать коммиты. Если вместо того чтобы выбрать 'pick', 'squash' или 'edit' для линии коммита, вы просто удалите линию, то это удалит коммит из истории.

**Изменение сообщений нескольких коммитов**

В Git отсутствуют инструменты для изменения истории, но вы можете использовать команду rebase, чтобы перебазировать группу коммитов туда же на HEAD, где они были изначально, вместо перемещения их в другое место. С помощью интерактивного режима команды rebase, вы можете останавливаться после каждого нужного вам коммита и изменять сообщения, добавлять файлы или делать что-то другое, что вам нужно. Вы можете запустить rebase в интерактивном режиме, добавив опцию -i к git rebase. Вы должны указать, какие коммиты вы хотите изменить, передав команде коммит, на который нужно выполнить перебазирование.

Например, если вы хотите изменить сообщения последних трех коммитов, или сообщение какого-то одного коммита этой группы, то передайте как аргумент команде git rebase -i родителя последнего коммита, который вы хотите изменить – HEAD~2^ или HEAD~3. Может быть, проще будет запомнить ~3, так как вы хотите изменить последние три коммита; но не забывайте, что вы, в действительности, указываете четвертый коммит с конца – родителя последнего коммита, который вы хотите изменить:

$ git rebase -i HEAD~3

Напомним, что это команда перебазирования – каждый коммит, входящий в диапазон HEAD~3..HEAD, будет изменен вне зависимости от того, изменили вы сообщение или нет. Не включайте в такой диапазон коммит, который уже был отправлен на центральный сервер: сделав это, вы можете запутать других разработчиков, предоставив вторую версию одних и тех же изменений.

Выполнение этой команды отобразит в вашем текстовом редакторе список коммитов.

Команда rebase в интерактивном режиме предоставит вам скрипт, который она будет выполнять. Она начнет с коммита, который вы указали в командной строке (HEAD~3) и повторит изменения, внесенные каждой из коммитов, сверху вниз. Наверху отображается самый старый коммит, а не самый новый, потому что он будет повторен первым.

Вам необходимо изменить скрипт так, чтобы он остановился на коммите, который вы хотите изменить. Для этого измените слово ‘pick’ на слово ‘edit’ напротив каждого из коммитов, после которых скрипт должен остановиться. Например, для изменения сообщения только третьего коммита, измените файл следующим образом:

Когда вы сохраните сообщение и выйдете из редактора, Git переместит вас к самому раннему коммиту из списка и вернет вас в командную строку со следующим сообщением:

$ git rebase -i HEAD~3

Stopped at f7f3f6d... changed my name a bit

You can amend the commit now, with

git commit --amend

Once you’re satisfied with your changes, run

git rebase --continue

Эти инструкции говорят вам в точности то, что нужно сделать. Введите

$ git commit --amend

Измените сообщение коммита и выйдите из редактора. Затем выполните

$ git rebase --continue

Эта команда автоматически применит два оставшиеся коммита и завершится. Если вы измените ‘pick’ на ‘edit’ в других строках, то можете повторить эти шаги для соответствующих коммитов. Каждый раз Git будет останавливаться, позволяя вам исправить коммит, и продолжит, когда вы закончите.

**Объединение коммитов**

С помощью интерактивного режима команды rebase также можно объединить несколько коммитов в один.

Если вместо “pick” или “edit” вы укажете “squash”, Git применит изменения из текущего и предыдущего коммитов и предложит вам объединить их сообщения.

Когда вы сохраните скрипт и выйдете из редактора, Git применит изменения всех этих коммитов и затем вернет вас обратно в редактор, чтобы вы могли объединить сообщения коммитов:

После сохранения сообщения, вы получите один коммит, содержащий изменения всех трех коммитов, существовавших ранее.

**Разбиение коммита**

Разбиение коммита отменяет его и позволяет затем по частям индексировать и фиксировать изменения, создавая таким образом столько коммитов, сколько вам нужно. Например, предположим, что вы хотите разбить средний коммит на три. Вы можете добиться этого, изменив в скрипте rebase -iинструкцию для разбиваемой коммита на “edit”:

Затем, когда скрипт вернет вас в командную строку, вам нужно будет отменить индексацию изменений этого коммита, и создать несколько коммитов на основе этих изменений. Когда вы сохраните скрипт и выйдете из редактора, Git переместится на родителя первого коммита в вашем списке, применит первый коммит (f7f3f6d), применит второй (310154e), и вернет вас в консоль. Здесь вы можете отменить коммит с помощью команды git reset HEAD^, которая, фактически, отменит этот коммит и удалит из индекса измененные файлы. Теперь вы можете добавлять в индекс и фиксировать файлы, пока не создадите требуемые коммиты, а после этого выполнить команду git rebase --continue:

**Удаление коммитов**

Вы также можете использовать интерактивное перебазирование для переупорядочивания или полного удаления коммитов. Если вы хотите удалить коммит и изменить порядок, в котором были внесены другие коммиты, то вы можете изменить порядок следования в скрипте и удалив ненужные коммиты.

Когда вы сохраните скрипт и выйдете из редактора, Git переместит вашу ветку на родителя этих коммитов, применит первый коммит, затем еще один и еще, и после этого остановится, не переходя к удаленным коммитам.