Качаем <https://git-scm.com/downloads>

Устанавливаем

Запускаем Git Bash

Меняем язык на русский. Это где?

Увеличиваем шрифт в окне до 12-го.

Увеличиваем окно. 48 на 160

Обязательная настройка после установки

Имя и ящик

git config --global user.name "Valeri Klimov"

почта

git config --global user.email "[va-klimov@mail.ru](mailto:va-klimov@mail.ru)"

или

git config --global user.email "klimov1853@gmail.com"

Показывать кириллицу вместо цифр в названии файлов

git config --global core.quotepath false

Отключение преобразование концов строк linux → windows

git config --global core.autocrlf false

Редактор VSCode

git config --global core.editor "code --wait"

Редактор для Notepad++ для commit.

-multiInst запустить ещё один экземпляр Notepad++

-notabbar отключить вкладку интерфейс

-nosession без загрузки предыдущей сессии

-noPlugin без загрузки плагинов

git config --global core.editor "'c:\Program Files (x86)\Notepad++\notepad++.exe' -multiInst -notabbar -nosession -noPlugin"

Где запускать

Нужно запускать там где есть репозиторий. Для этого копируем ярлык в рабочую папку. Удаляем у ярлыка рабочий каталог и ключ --cd-to-home.

Создание рабочей папки

Создаем папку в которой будет вся информация о проекте, включая документацию. Чтобы была вся история.

По умолчанию находимся в папке c:\Users\\_аккаунт\_. Сделаем, чтобы **Git Bash** сразу по умолчанию попадал в эту папку. Для этого нужно добавить в файле .bash\_profile строку перехода в нужный каталог:

# Слэш прмой!

cd [d:/my/Desktop/ALFA/waveapps](file:///d:\my\Desktop\ALFA\waveapps)

# generated by Git for Windows

test -f ~/.profile && . ~/.profile

test -f ~/.bashrc && . ~/.bashrc

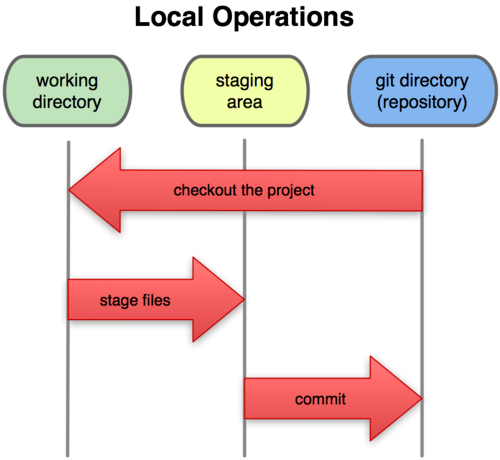
Создание локального репозитория

Создаём папку с программой. Создаем в корне папки базу данных GIT:

git init

Принцип работы

Помещаем файлы в рабочий каталог. После изменения проверяем хэш файла. Сравниваем его с хешем из БД. Если в Бд Другой хэш или его там нет, то делаем сжатую копию изменённых файлов в транзитной области. По необходимости сгружаем все измененные файлы в БД. Фиксируем новый состояние БД (узел или версия)и пишем к нему комментарий. При необходимости этот узел и восстанавливаем.



В самом простом случае это две команды:

git add .

git commit

Все что есть в каталоге подготовить к загрузке в БД. Загрузить в БД.

Удаленное хранилище

Делаем

- На gitlab.com. Логонимся SanMaster/ zaq123edc. Добавляем новый репозиторий по названию папки.

- В GIT Bash перейти в рабочую папку, иначе выдаст ошибку:

cd c:/cloud\_mail/\_КОДИНГ/phenix

- Установить переменную origin адрес удаленного репозитория.

git remote add origin <https://gitlab.com/SanMaster/phenix.git>

Отправить в хранилище.

-u

--set-upstream

For every branch that is up to date or successfully pushed, add upstream (tracking) reference, used by argument-less git-pull[1] and other commands. For more information, see branch.<name>.merge in git-config[1].

git push origin

или

git push <https://gitlab.com/SanMaster/phenix.git>

Ещё точнее

git push --set-upstream https://gitlab.com/SanMaster/call.git master

Забрать из хранилища

git pull origin

### Где хранить

Если один разработчик, то в любом облаке. Нет необходимости куда-то скидывать репозиторий. Минус в том, что можно случайно удалить папку вместе с репозиторием.

Если несколько разработчиков, то хранить на удаленном сервере.

### Основная концепция GIT

GIT репозиторий это база данных мгновенных снимков файлов в каталоге.

Любое изменение в файле изменяет его хеш. Это является признаком изменений в файле.

Две основные операции: считаем хеш файлов и потом у кого он изменился по сравнению с последним снимком — кладем в репозиторий.

Репозиторий находится в той же папке в скрытой папке .git.

Изменяем → Хешируем → Грузим в базу.

### Что такое удаленное хранилище

*Что такое удаленное хранилище?*

Это копия папки .git на удаленном компьютере.

Можно ли под репозиторий использовать обычный сервер?

Можно. Git умеет работать с четырьмя сетевыми протоколами для передачи данных: локальный, Secure Shell (SSH), Git и HTTP.

**Локальный** — можно делать копию на любом компьютере в локальной сети Windows или Linux. Основной недостаток — недоступна работа через интернет.

$ git clone /opt/git/project.git

SSH — это протокол есть на любом сервере. Недостаток — нет анонимности. Все пользователи должны иметь доступ к серверу.

$ git clone ssh:[//user@server](mailto://user@server):project.git

Git. Недостатки. Нужно ставить службу git. Очень сложный в настройке протокол.

HTTP/S. Недостаток повышенная нагрузка на сеть.

$ git clone <http://example.com/gitproject.git>

Каждый снимок имеет дату и связан с другими снимками связями, больше похожими не на дерево а на железнодорожный

Все зависит от потребностей. Если нет необходимости работать на удаленном хранилище через WEB интерфейс, то под репозиторий можно использовать абсолютно любое надежное удаленное хранилище. Фактически на удаленном сервера хранится папка с GIT базой данных — локальная копия такой же базы на локальнм компе. В частности, комагда **clone** делает копию удаленной базы на локальном компе.

Вся дальнейшая работа связана с отсылкой и приемом из базы снимков

Сколько места на сайте под репозитории?

Если репозиторий уже имеется

Адрес репозитория с сайта - https://bitbucket.org/SanMaster/order

$ git remote add bitbucket https://SanMaster@bitbucket.org/SanMaster/order

$ git remote -v

bitbucket https://SanMaster@bitbucket.org/SanMaster/order (fetch)

bitbucket https://SanMaster@bitbucket.org/SanMaster/order (push)

Git Gui + Git Bash

Будем использовать Git Bash и Git Gui. Нужно взять всё лучшее от каждой программы.

Добавляем пользователя и мыло: Valery Klimov / va-klimov@mail.ru. Utf-8 по умолчанию.

Git Gui: Edit -> Options – справа глобальные настройки. Выставил по умолчанию UTF-8.

Изменить редактор для коммитов на Notepad++.

Git Bash:

git config --global core.editor "'c:\Program Files (x86)\Notepad++\notepad++.exe' -multiInst -notabbar -nosession -noPlugin"

Новая жизнь репозитория

Чтобы начать с начала (история точно не нужна) нужно сначала склонировать репозиторий в папку. Потом перейти на ветку с которой необходимо создать новый репозиторий. Удалить папку .git. Сделать git init. Затем делаем стартовый коммит. Создаем ветку develop.

Создаем репо в облаке. Обозначаем адрес удаленного сервера

Переходим туда и начинаем разработку.

Ветвление

Ветвление больше напоминает не дерево, ветки железнодорожных путей.

Согласно рабочей схемы основная ветка не master, а develop.

В ветке develop ведется основная работа. Он неё ветвятся feature branches – ветки с осмысленными названиями в которых разрабатываются дополнительные функции программы а потом сливаются с основной веткой develop. Количество этих веток может быть много.

Из develop создается одна ветка release, которая фактически является веткой устранения грубых ошибок. В процессе устранения release регулярно сливается с веткой develop. После устранения всех найденных ошибок release сливается с master и develop. В master создается релиз Х.0. Устранение ошибок в рабочих версиях осуществляется в ветке hotfix, с последующим слиянием с master и develop.

Создаем ветку develop. Физически создается копия ссылки на последний коммит. Теперь ссылка master и develop указывают на один и тот же коммит. А указатель HEAD на master.

git branch develop

Переходим на ветку develop. Физически указатель HEAD стал указывать на develop. HEAD это указатель на текущий указатель.

git checkout develop

Теперь указатель master будет указывать на последнюю рабочую версию. И сдвинется на следующий коммит только после слияния с ним какой либо ветки.

Переходим на ветку develop. Теперь ссылка HEAD. Будет указывать на ссылку inifile. Теперь при каждом новом коммите будет перемещаться два указателя: inifile и HEAD, а develop останется указывать на коммит, где произошло ветвление.

Слияние веток

Для слияние веток переходим на ветку, в которую сливаем. Это develop.

git checkout master

Далее сливаем в неё ветку develop

git merge develop

По волшебной схеме создаем ветку develop.

git branch develop

Переходим на ветку develop.

Что у нас с репозиторием? В каком он состоянии?

git push bitbucket master

спрашивает имя и пароль

сбоит. Клонируем взад.

git clone https://SanMaster@bitbucket.org/SanMaster/order d:\order

Удобно запускать Git Bash из под Git Gui. Открывается сразу в нужном репозитории.

Visualize All Branch History – визуализация дерева.

1. Связать локальные файлы с репозиторием.

FAQ

Свой сервер GIT

Настройка git

Удаленное хранилище

Клонирование

Тяни толкай на удаленный репозиторий ветки

Ветвление

Восстановление и откат

Начинаем

Создаем пустой репо на своем сервере.

Переходим в новый каталог. Создаем пустую базу репозитория.

git init

Отключение

git config --global core.autocrlf false

Бродим по истории

Сначала нужно найти дорогу.

git log # в прошлое

git log --all # вся истоия

Теперь можно идти. Достаточно нескольких первых символов хэш кода.

git checkout 734713b # Перенос указателя HEAD

Нужно вернуться назад. Указатель develop или master указывают на последнний коммит:

git checkout develop

Правим баги

(progit ст.53-54)

Делаем ответвления для исправления бага. Пусть это будет вылет по ошибке 53. Создаем ветку и уходим на неё:

git branch error53

git checkout error53

Или можно так - сразу создаём и уходим на нашу ветку.

git checkout -b 'error53'

Вдруг появилась ещё одна ошибка. Комитим текущую работу. Переходим на master. Делаем ещё одну ветку. Начинаем работать.

git checkout master

git branch fatal

git checkout fatal

Чтобы продолжить работать с первой ошибкой, комитим и уходим на error53.

git checkout error53

Если ошибка 53 исправлена, то снова коммитим, уходим на master и сливаем ветку error53 с веткой master.

git checkout master

git merge error53

Тоже делаем с веткой fatal после исправления.

Восстановление и откат

Отмена изменений файла

git checkout -- <файл> # откатить все изменения

Откат к последненму коммиту

git reset --hard # восстановить последний коммит

Индексирование файлов

Перед создание общего снимка проектной папки необходимо создать снимки каждого файла в папке. Эта процедура называется индексированием. Индексирование можно делать не файлу, всем файлам в папке всей папке проекта и т.д.:

git add <файл>

git add <каталог>

git add .

Проблемы индексации

Не всё так просто. Файлы могут быть не только изменены, но и удалены, перемещены, переименованы, добавлены и т. д. Git строго смотрит за этим и просит подтвердить всё эти изменения соответствующими командами. Состояние можно посмотреть командой:

git status

Все команды подтверждения отображаются в листинге команды. Ниже список некоторых команд:

git rm <файл> # отменить слежку за удаленным файлом

git rm -r <каталог> # рекурсивно

Фиксация

Фиксация — сохранение образа проектной папки в базе данных git. Все существующие файлы и структура размещения файлов по папкам запоминается в базе данных и этой фиксации присваивается уникальный номер. По этому номеру всегда можно восстановить образ проектной папки из базы данных.

git commit

Должен текстовой редактор в котором записаны все коментарии. Как настроить другой редактор см НАСТРОЙКА git.

git commit -m 'комментарий к образу'

После этой этого команда get status будет сообщать:

On branch master

nothing to commit, working directory clean

Фиксировать нечего. Рабочая папка чистая.

Чтобы отменить последнюю фиксацию, но оставить изменения:

git reset --soft HEAD^

Забыл вставить в коммит

git add <файл>

git commit --amend

Внимание! Откроется редактор. Там будет общий комментарий. Выход сочетание ':q'. В последний коммит добавится файл.

Просмотр коммитов

q (выход из less, когда на экране появляется END)

История в обратном порядке:

git log

В одну строку

git log --pretty=oneline

В формате

git log --pretty=format:"%h - %an, %ar : %s"

git log --pretty=format:"%h %ad | %s%d" --graph --date=short

%h - Хеш коммита

%ad - Дата автора (формат соответствует параметру --date=)

%s Комментарий

Удаление коммитов

Делаем

git rebase -i HEAD~3

чтобы вызвать в редактор три последних коммита. Удаляем ненужный.

Нужно только переместится в нужную ветку.

Ветвление

Уплотнение коммитов

Сброс ветки на удалённый репозиторий

Общая форма. Псевдоним origin – адрес удаленного сайта.

git push origin <ветка>

git push origin master

git push origin develop

Обязательно спросит пароль hy3KU4Hv7kH5. Процесс сброса будет отображен на экране или скажет Already up-to-date (уже было обновлено).

Чтобы запомнить пароль для винды

git config --global credential.helper wincred

Вытягивание ветки

Общая форма. Псевдоним origin – адрес удаленного сайта.

git pull origin <ветка>

git pull origin master

git pull origin develop

Что такое origin, master и т. д. Основные псеводонимы можно посмотреть в .git/config. они автоматически подставляются в командную строку:

[remote "origin"]  
 url = https://klimov1853@code.google.com/p/order-delphi  
 fetch = +refs/heads/\*:refs/remotes/origin/\*

[branch "master"]  
 remote = origin  
 merge = refs/heads/master

[branch "develop"]  
 remote = origin  
 merge = refs/heads/develop

Работа с файлами

Работаем с файлами.

В конце рабочей сессии перед коммитом смотрим что мы наработали.

#--- Git Bash -----------------------------------------------------------------------

git status

#--- Git Bash -----------------------------------------------------------------------

Будет выведен список файлов с пояснениями что необходимо сделать, чтобы с каждым файлом.

Если нужен откат файла, то:

git checkout — <имя файла> #два маленьких тире, а не один большой

=============================

СОЗДАНИЕ И ПОЛУЧЕНИЕ РЕПОЗИТОРИЕВ

=============================

СОЗДАНИЕ

========

Для глобального проекта заводим репозиторий Создание репозитория на https://code.google.com/intl/ru/

Инструкция по созданию там же.

ПРАКТИКА:

---------

Создание репозитория для программы orders.

- Заходим на https://code.google.com/intl/ru/

- ищем на сайте ссылку "Create a new project" и нажимаем

- на новой странице:

Project name - название проекта. это будет названием папки по умолчанию: order-delphi

Project summary - краткое описание проекта: To organize the design process

Description - полное описание проекта: Creating a virtual queue of orders. Accounting for time-layout. Keeping order history.

Version control system - Git

Source code license - GNU GPL 3 (за свободные проекты денег не берут)

Project label(s) - ключевые слова для поиска другими программистами: Delphi MySQL Windows

- нажимаем кнопку создать проект

На странице https://code.google.com/intl/ru/ в Избранных появляется order-delphi

ПОЛУЧЕНИЕ

=========

git clone URL каталог

Как получить с https://code.google.com/hosting/

- В правом углу выпадающий список Избранное все мои проекты.

- Выбираем order-delphi

- Заходим на вкладку Source видим команду для клонирования репозитория git clone git clone

https://klimov1853@code.google.com/p/order-delphi/

Клонируем в каталог "C:\order-delphi". Обратный слешь обязательно в кавычках.

Для URL кавычки тоже не помешают. Если не указать папку, то в текущей папке создастся папка с названием проекта.

git clone "https://klimov1853@code.google.com/p/order-delphi" "C:\order-delphi"

чтоб постоянно не просил пароль, сделай в домашней директории файлик

\_netrc (для винды) или .netrc (для юниксов)

с содержимым:

machine code.google.com login klimov1853@gmail.com password kQ6Ps3aA5cS8

Обратная команда git reset

ИГНОРИРОВАНИЕ ФАЙЛОВ

====================

Файл .gitignore

Список что git игнорирует при индексации.

ФИКСАЦИЯ СЛЕПКА

===============

git commit -m "комментарий"

СОХРАНЕНИЕ В УДАЛЕННЫЙ РЕПОЗИТОРИЙ

==================================

git push origin master

origin - это алиас удаленного репозитория.

master - название ветки

пароль kQ6Ps3aA5cS8

work in a branch RELEASE

work in a branch HOTFIX

work in a branch MASTER

Удаленное хранилище

Если грубо, то удаленное хранилище это копия локального в «другом» месте. Можно копировать туда-сюда весь репозиторий (clone) или кусками (pull и push).

Данные передаются по разным протоколам:

локальный: /opt/git/project.git

ssh: ssh://user@server:project.git или user@server:project.git

git git://example.com/gitproject.git

http(s) http://example.com/gitproject.git

Достоинства и недостатки см. Progit.pdf.

Для протокола git нужно устанавливать сервер git!

Локальное хранилище (на сервере)

1. Выбрать место для хранения.

2. Создать файловую структуру. Корень GIT, каталоги внутри по названию проектов.

3. Создать «голый» репозиторий. В git переходим в любую пустую папку. Запускаем команду:

git --bare init

Появились 4 папки и 3 файла — «голый» репозиторий.

4. Содержимое папки скопировать во все пустые папки проектов. Теперь в каждой папке на сервере у нас куча пустых репозитариев.

Как работать

Команды clone push и pull.

1. Создаем псевдонимы для длинных названий репозиторий:

$ git remote add sserver \\SSERVER\soft\GIT\staff\_html

$ git remote -v

sserver //SSERVER/soft/GIT/staff\_html (fetch)

sserver //SSERVER/soft/GIT/staff\_html (push)

Чтобы сбросить ветку в репо, нужно набрать команду:

$ git push sserver master

Если логин есть на сервере, то просто в репо будет записана ветка master, иначе перед записью попрости пароль.

Удалить ошибочный псевдоним можно командой:

$ git remote rm psevd

Клонирование

git clone "https://klimov1853@code.google.com/p/superstaff-delphi" staff

git clone "https://klimov1853@code.google.com/p/vacant-collector" vacant

Как посмотреть коммиты что в репозитории?

git fetch [имя удал. Сервера]

git fetch origin develop

-Select Components галочку на One the Desktop? чтобы на столе ссылка установилась.

-Use Git Bash only

-Use OpenSSH

-Checkout as-is, commit as-is (чтобы с переводиами строк не мудрила)

На столе появляется ссылка на GIT для работы с командной строкой.

Как получить помощь?

git help <команда>

УСТАНОВКА

скачать и установить

Установка git в Windows

=======================

Качаем на http://git-scm.com

-Ставим в C:\Git (на случай если приспичит patch в системе дописать, чтобы с именеми не трахатся)

-Select Components галочку на One the Desktop? чтобы на столе ссылка установилась.

-Use Git Bash only

-Use OpenSSH

-Checkout as-is, commit as-is (чтобы с переводиами строк не мудрила)

На столе появляется ссылка на GIT для работы с командной строкой.

СКЛЕРОЗ!!!

==========

Настройка git bash

Папка по умолчанию.

При запуске удобно, чтобы git bash запускался по-умолчанию не в домашнем каталоге, а в рабочем. Правой мышкой на git bash выбираем Свойства и меняем Рабочую папку на нужную.

Вставить текст

Вставить в окно из буфера обмена --> Shift-Insert

Скопировать текст

Скопировать из окна в буфер обмена: Иконка → Изменить → Пометить. Выделить нужное и нажать Enter.

Ширина окна

Окно Свойство → Расположение → Ширина

НАСТРОЙКА git

Добавить имя для всех репозиториев

git config --global user.name "Klimov Valeri"

Добавить e-mail для всех репозиториев

git config --global user.email va-klimov@mail.ru

Отключить коррекцию переноса строк

git config --global core.autocrlf false

Убрать контроль прав доступ на файлы

git config core.filemode false

Нужно отключить представление имен файлов в кодировке Unicode в виде кодов символов, то есть в виде «\nnn\nnn…».

git config --global core.quotepath off

Изменить редактор для git

git config --global core.editor "'C:/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin"

Показать используемые настройки

git config --list

Хранилище

Репозиторий, хранилище — место, где хранятся и поддерживаются какие-либо данные. Далее по тексту будем использовать слово хранилище, как рекомендовано для русского языка. В git практикуются локальные и удаленные хранилища. Локальные хранилища в git считаются основными. Удаленные хранилища используются для передачи информации между локальными хранилищами.

Создание Локального хранилища

Git создает базу данных хранилища в корневой папке. База лежит в папке .git. Создание локального хранилища — это создание в проектной папке базы данных git.

Переходим в папку. Создаем пустую БД и разрешаем использование русской кодировки:

git init

git config core.quotepath false

Если есть удаленное хранилище, можно сделать копию в локальном хранилище командой:

git clone "https://klimov1853@code.google.com/p/region-delphi" "c:\sd"

Откуда и куда обязательно в кавычках.

Как работать основное

Вся база находится в папке .git. В удаленное хранилище только делается копия. На случай, если на локальном всё потеряется. Что нужно:

Походить по веткам и посмотреть комментарий к узлам.

Посмотреть файлы и их содержимое.

Загрузить изменения в базу и написать комментарий к узлу.

Ходим по веткам

Загрузит изменения

Делаем копию в удаленный репозиторий

Вводим команду

$ git remote -v

origin https://github.com/HarmonySearch/HiSEO.git (fetch)

origin https://github.com/HarmonySearch/HiSEO.git (push)

Видим, что репозиторий уже привязан к удалённому. Пытаемся записать. Всплывает окно логина и пароля. Как прописать в текущий репозиторий.

git push origin master

Смотрим мануал по генерации ключей

$ ssh-keygen -t rsa -C "klimov1853@gmail.com"

Generating public/private rsa key pair.

Enter file in which to save the key (/c/Users/klimo/.ssh/id\_rsa):

Created directory '/c/Users/klimo/.ssh'.

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /c/Users/klimo/.ssh/id\_rsa

Your public key has been saved in /c/Users/klimo/.ssh/id\_rsa.pub

The key fingerprint is:

SHA256:Y0eo8q/byoapa5vyWPZfyfpVK6Kkqyg4GqyWINv5cKI klimov1853@gmail.com

The key's randomart image is:

+---[RSA 3072]----+

| |

| . |

| . . |

| . . |

| . . S . . |

|+ o o + . . |

|=+\*..oo = o . |

|BX+\*o+.\* o . |

|E=B\*=+O=+ |

+----[SHA256]-----+

Копируем в github как написано в https://hyperhost.ua/info/generatsiya-i-dobavleniya-ssh-klyucha-dlya-podklyucheniya-k-udalennomu-repozitoriyu-github/

Запускаем агент

eval `ssh-agent.exe`

добавляем наш ключ

$ ssh-add id\_rsa

Enter passphrase for id\_rsa:

Identity added: id\_rsa ([klimov1853@gmail.com](mailto:klimov1853@gmail.com))

Меняем протокол:

Перечислите существующие пульты, чтобы получить название пульта, который вы хотите изменить.

$ git remote -v

> origin https://github.com/USERNAME/REPOSITORY.git (fetch)

> origin https://github.com/USERNAME/REPOSITORY.git (push)

Измените URL вашего пульта с HTTPS на SSH с помощью git remote set-urlкоманды.

$ git remote set-url origin git@github.com:HarmonySearch/HiSEO.git

Убедитесь, что удаленный URL-адрес изменился.

$ git remote -v

# Verify new remote URL

> origin git@github.com:USERNAME/REPOSITORY.git (fetch)

> origin git@github.com:USERNAME/REPOSITORY.git (push)

Теперь можно грузить

История

Однострочная

$ git log --pretty=oneline

831cc0d8569b92ef76bee4b3e920940f021bdb57 (HEAD -> master, origin/master) добавил ярлык git в каталог

a5b9795f2ff6e89ac967b80c1de742441484ce77 20 09

f6f8ada3349127401d0da4733a2919a2926f2306 2504

2ccff085a318971b4ee4a293dd247807420e865b 21 04 2020

086319493397253dcab2d27350d3c426614077d0 20-04

86472c0dec7a9a89ec53e6680d6765f9698eff3e 09,04+

98d536fe21f78d106d79b634d87378d767acbe9e 09,04,2020

725e33d23c047d78e1eedce333801d29d6b3e749 31,03

473fdfc66c0b966a4d9de7985b4d653b1cc7daf7 26 02

1c63b7ec07ea6b882caf536c656c25c6af7f20ad 21 02

d133c53b79eab61c14fbb5fa8c5aaba648db0830 19 02

038bcfb8a9fa39fd320a69bb365605bff8beb139 Initial commit

Восстанавливаем точку

$ git checkout a5b9795f2ff6e89ac967b80c1de742441484ce77

Previous HEAD position was f6f8ada 2504

HEAD is now at a5b9795 20 09

Находим в папке нужный файл и смотрим его.

Потом возвращаемся на последнюю точку ветки

git checkout master

Как удалить точку

Фактически удалить невозможно (