Примеры:

* копии файлов
* Microsoft Word
* Adobe Photoshop
* Wiki

История:

SCCS (1972)

RCS (1982)

SCCS и RCS отслеживают изменения только в одном файле

CVS (1986-1990) (параллельная работа с несколькими файлами + удаленная работа)

SVN (2000) (история директории)

BitKeeper SCM (2002-2005 год использовались в Linux) - первый распределенный VCS

Распределенные системы контроля версий:

* каждый пользователь имеет свои репозиторий
* изменения отслеживаются как наборы патчей (наборы изменений)

# Настройка Git

System:

/etc/gitconfig

Program Files\Git\etc\gitconfig

git config --system

User:

~/.gitconfig

$HOME\.gitconfig

git config --global

Project

project/.git/config

git config

## Config Samples

git config --global user.name “name”

git config --global user.email “email”

git config --list

git config --global core.editor “nano”

git config --global color.ui true

## Help

git help

# Начинаем

создание репозитория

git init

файл с конфигурацией

.git/config

добавление изменений

git add .

коммит

git commit -m “Initial commit”

best practices for commit massage:

* небольшие коммиты
* длина < 72 символов
* писать то, что делает этот коммит

Лог коммитов

git log

Лимитированное количество комитов

git log -n 1

Временной период( начиная с / до этой даты)

git log --since=2012-06-15

git log --until=2012-06-15

Автор

git log --author=”marat”

Использование grep

git log --grep=”Init”

# Концепция и архитектура

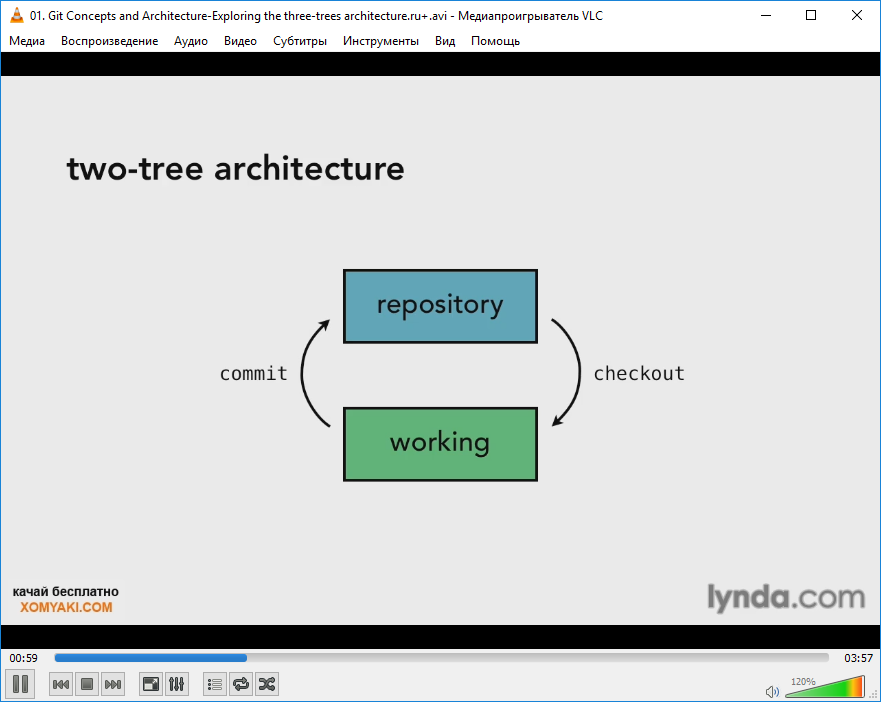
*Обычно*

Архитектура двух деревьев:

репо

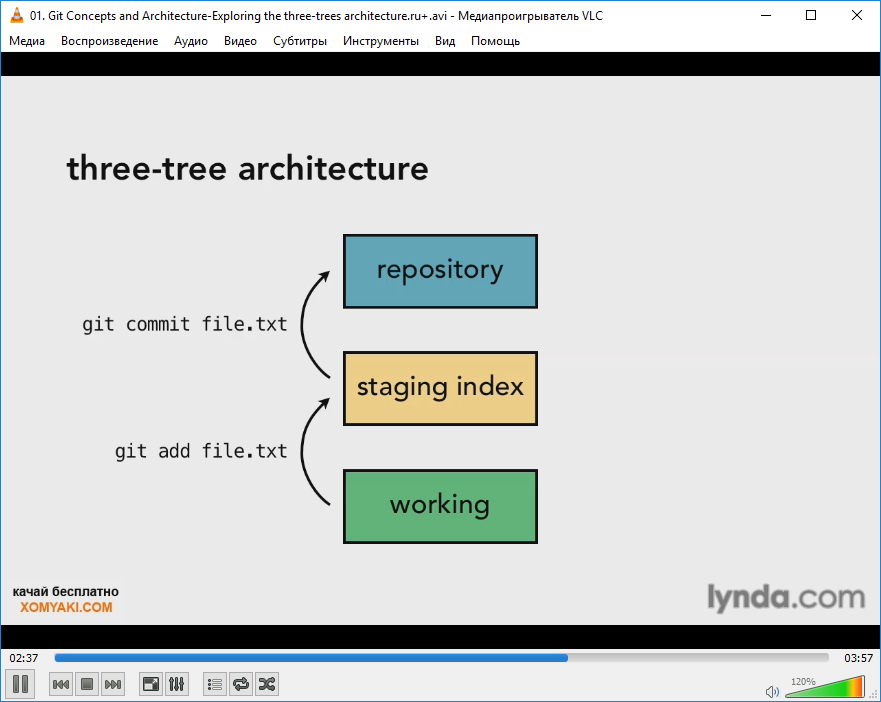
|

рабочая копия



*ГИТ*

В гите: архитектура трех деревьев



Git workflow

Гит не работает с файлами, он работает с наборами

изменения

git add

git commit

и т.д.

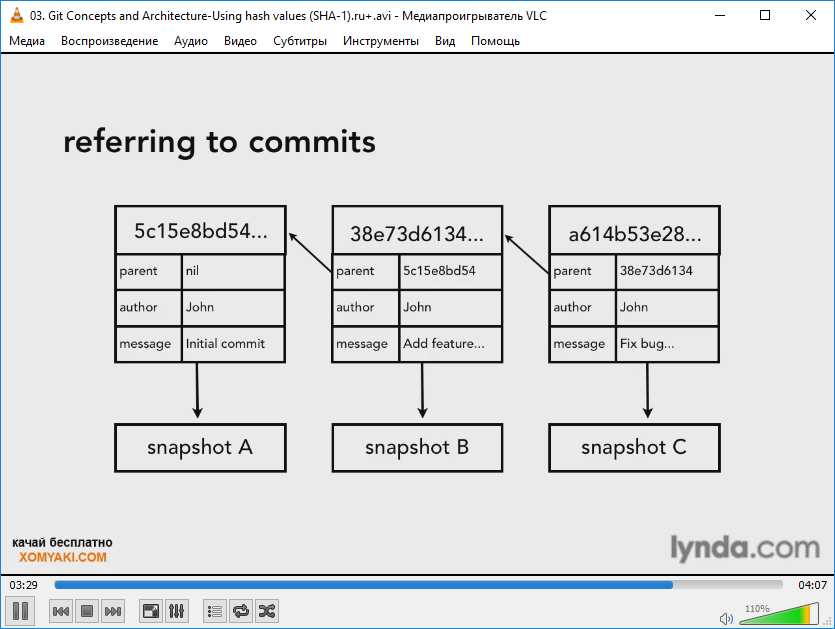
Маркеры (айдишники) коммитов

Алгоритм контрольной суммы генерирует число - это число используется в гит

Нельзя поменять данные в коммите без изменения этого числа

Используется хеш алгоритм SHA-1

Хеш - это шестнадцатиричная строка 40 символов



HEAD указатель

HEAD - ссылочная переменная (указатель) указывает на верхушку текущего бранча в нашем репо.

Последнее состояние нашего репо.

HEAD перемещается на верхушку того бранча, где был сделан checkout.

.git/HEAD - указывает на ветку

refs/heads/master - конкретный коммит

# Внесение изменений в файлы

Различия между рабочей директорией, буфером и репо

git status

Просмотр различий между рабочей копией и репозиторием

git diff

Просмотр различий между буфером и репозиторием

git diff --staged

*Удаление файлов (можно руками, можно из гит)*

git rm file\_to\_delete1.txt

Перемещение и переименование (можно руками или из гит)

git mv old\_name new\_name

# Отмена изменений в git

## Отмена изменений в рабочей директории

checkout - берез из репозитория

Восстановление файла из репозитория (если мы что-то сломали в файле и хотим это вернуть)

git checkout index.html

-- говорит оставаться в текущем бранче и взять из него файл index.html

git checkout -- index.html

## Отмена изменений в буфере

Вывод изменения из буфера (вытащить файл). Из буфера в рабочую директорию.

git reset HEAD resources.html

## Отмена изменений в репозитории (отмена коммитов)

Внесение изменений в последний коммит. Можно редактировать только последний коммит, тот на который указывает HEAD

git commit --amend -m “Hello”

Работа со старыми коммитами. checkout из предыдущей ревизии вставляет изменения в буфер.

git checkout 3434239ab99d8e -- resoures.html

git diff --staged

различия между буфером и рабочей директорией

git diff

различия между буфером и репозиторием

git diff --staged

## Отмена изменения коммита

git revert 3434239ab99d8e

## Перемещение HEAD

git reset --soft 97808у435а

Перемещает HEAD, но не трогает буфер или рабочую директорию

git reset --midex (default)

Перемещает HEAD и изменяет буфер, но не трогает рабочую директорию

git reset --hard

Перемещает HEAD и изменяет буфер и рабочую директорию

## Удаление неотслеживаемых файлов

Говорит, что сделает

git clean -n

Удаляет

git clean -f

# Игнорирование файлов

Создать:

.gitignore

# - комментарий

# Перемещение по дереву коммитов

tree-ish - по простому - это ссылка на коммит

full SHA-1 hash

short SHA-1 hash

HEAD pointer

branch reference, tag reference

Родительский коммит:

HEAD^

acd3422^

master^

HEAD~1

HEAD~

Дальний предок:

HEAD^^

HEAD~2

список дерева объектов

git ls-tree <tree-ish>

список файлов и папок которые видны для данной точки

git ls-tree HEAD

git ls-tree master

Просмотр директории

git ls-tree master^ foldername

Способ вывода

git log --oneline

git log --oneline -3

git log --since=”data”

git log --name=”Marat”

git log --grep=”temp”

git log 2341ac9..38aeff93 --oneline

Посмотреть разницу между файлами (patch)

git log -p

Статистика изменений

git log --stat --summary

Граф

git log --graph

git log --oneline --graph --all --decorate

Подробно о коммите

git show 93810fe

## Сравнение коммитов

различие между workdir и 3832ef0

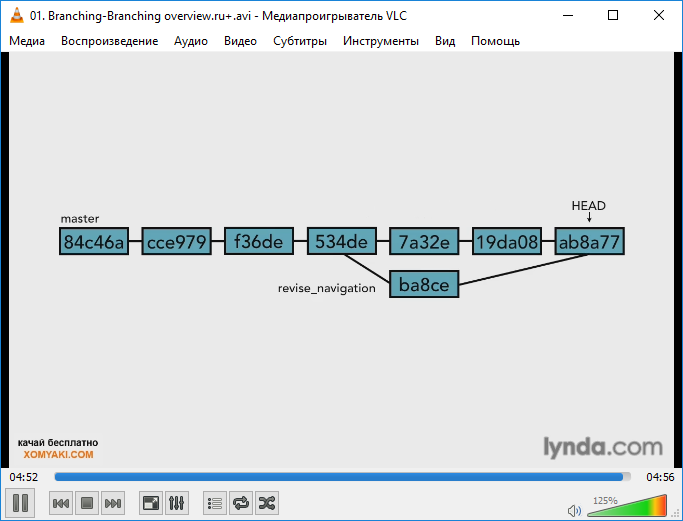
git diff 3832ef0

git diff 3832ef0 tours.html

git diff 3832ef0..39e10fd

git diff 3832ef0..39e10fd tours.html

# Бранчинг



список бранчей на локальном компьютере

git branch

cat .git/HEAD

список бранчей

ls -la .git/refs/heads

create branch

git branch new\_feature

переключение между бранчами

git checkout new\_feature

создание и переключение на новый бранч

git checkout -b name\_branck

просмотр разной информации

git log --graph --oneline --decorate --all

Рабочая директория должна быть чистой при переключении

Сравнение бранчей

git diff master..new\_feature

git diff --color-words master..new\_feature

Произошло ли полное слияние бранчей

git branch --merged

Переименование бранчей

git branch --move new\_feature seo\_title

Удаление бранчей

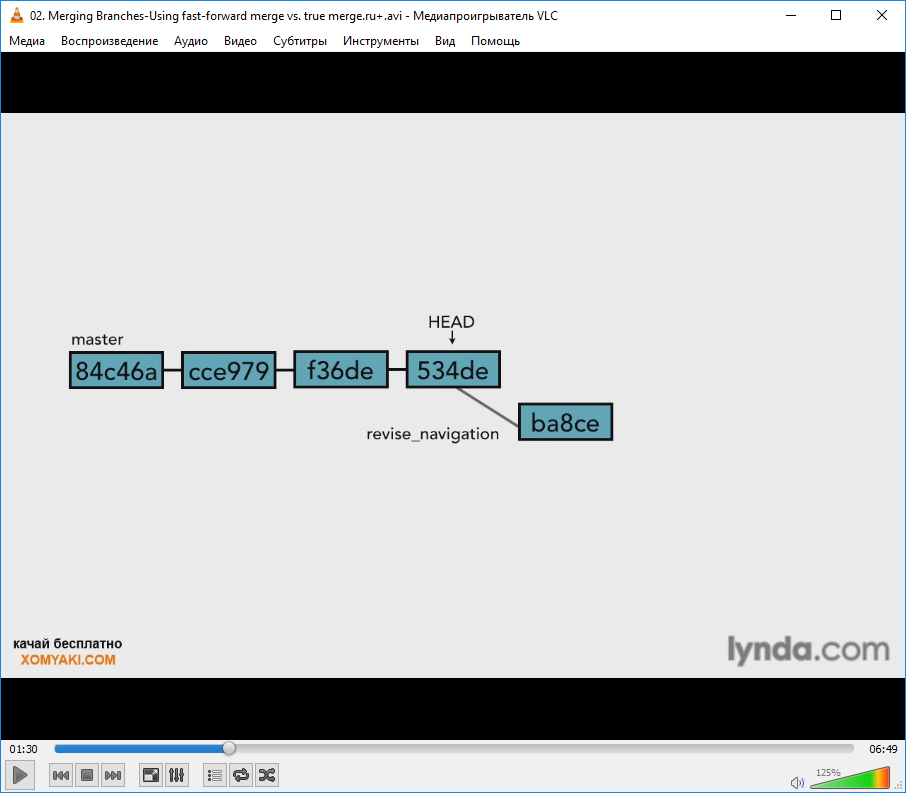
git branch -d branch\_to\_delete

# Слияние бранчей

git checkout master

git merge seo\_title

fast-forward



слияние с комитом (не ff)

git merge --no-ff branch\_name

Но в остальных случаях будет сложное слияние

Конфликты слияния

Когда есть два изменения в одной строке

