



Основы работы с репозиториями кода

CI/CD

Git

как хранить и не терять код программ?
как делиться кодом с товарищем?

Есть множество способов хранения исходников программ:

- создать для них папку в разделе «Мои документы»,
- закидывать их в облако (Яндекс.Диск) и подписывать версии,
- загружать в «Избранное» в Telegram или «ВКонтакте»,
- можно записывать список изменений текстом в приватном Telegram-канале,
- можно деплоить проект с помощью простого скачивания и распаковки ZIP-архива с файлами вашей программы,
- можно сообщать о багах в вашем любимом софте сообществу анонимов в публице в VK...

как хранить и не терять код программ?
как делиться кодом с товарищем?

Контроль версий (англ. Version control), также известный как контроль исходного кода (англ. Source control) - это практика отслеживания и управления изменениями в программном коде.

Системы контроля версий (англ. version control systems) - это программные инструменты, которые помогают группам разработчиков управлять изменениями исходного кода с учетом хронологии.

Главные возможности системы контроля версий:

- Полная история изменений каждого файла за длительный период
- Возможность отслеживать каждое изменение, внесенное в программное обеспечение

Репозиторий

Репозиторий — место, где хранятся и поддерживаются какие-либо данные. Чаще всего данные в репозитории хранятся в виде файлов, доступных для дальнейшего распространения по сети.

Также под репозиторием понимается каталог файловой системы, в котором могут находиться:

- сами контролируемые файлы
- журналы конфигураций сборки проекта
- журнал операций, выполняемых над репозиторием

Репозиторий

Управление исходным кодом (source control management, SCM) используется для отслеживания изменений в репозитории исходного кода.

SCM отслеживает текущую историю изменений в базе кода и помогает разрешать конфликты при объединении обновлений от нескольких участников.

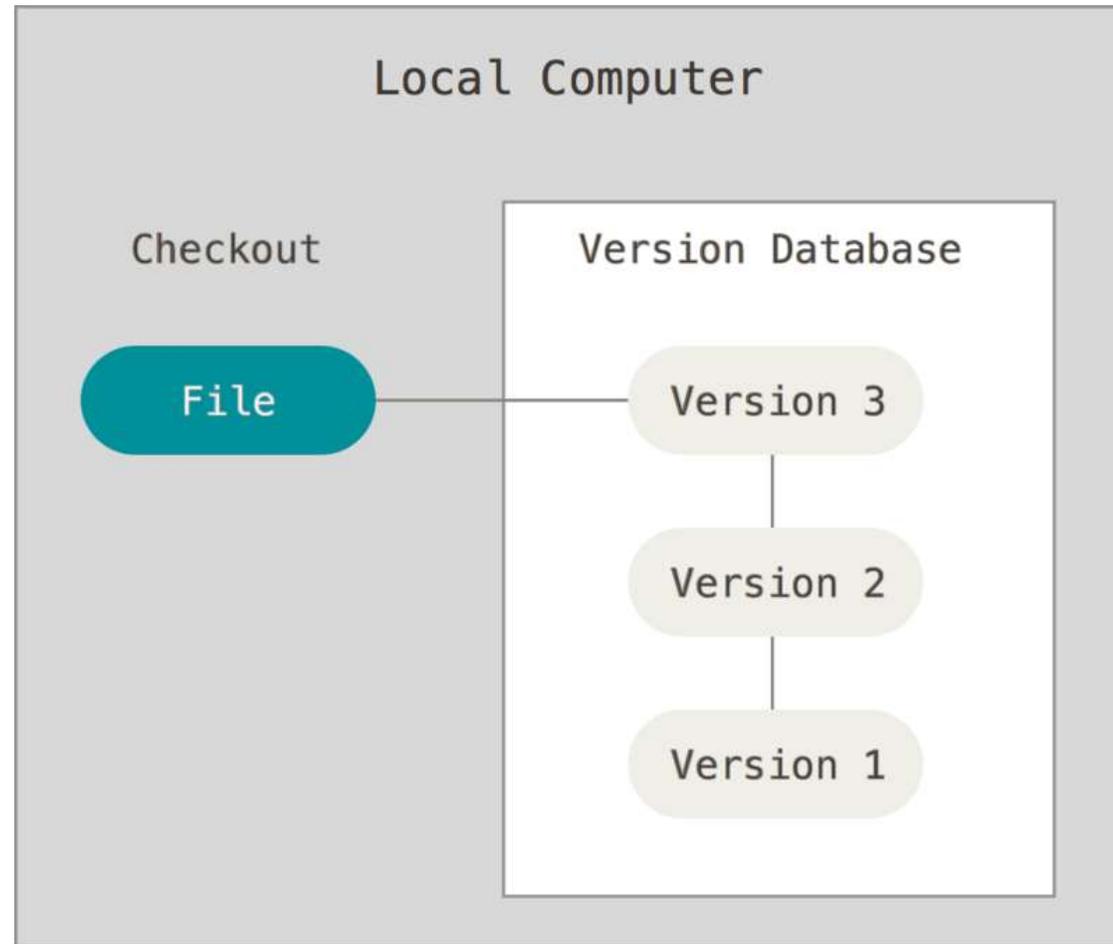
Виды систем контроля версий

Локальные системы контроля версий

Одной из наиболее известных VCS такого типа является [rcs](#) (Revision Control System).

Утилита основана на работе с наборами патчей между парами версий (патч — файл, описывающий различие между файлами), которые хранятся в специальном формате на диске.

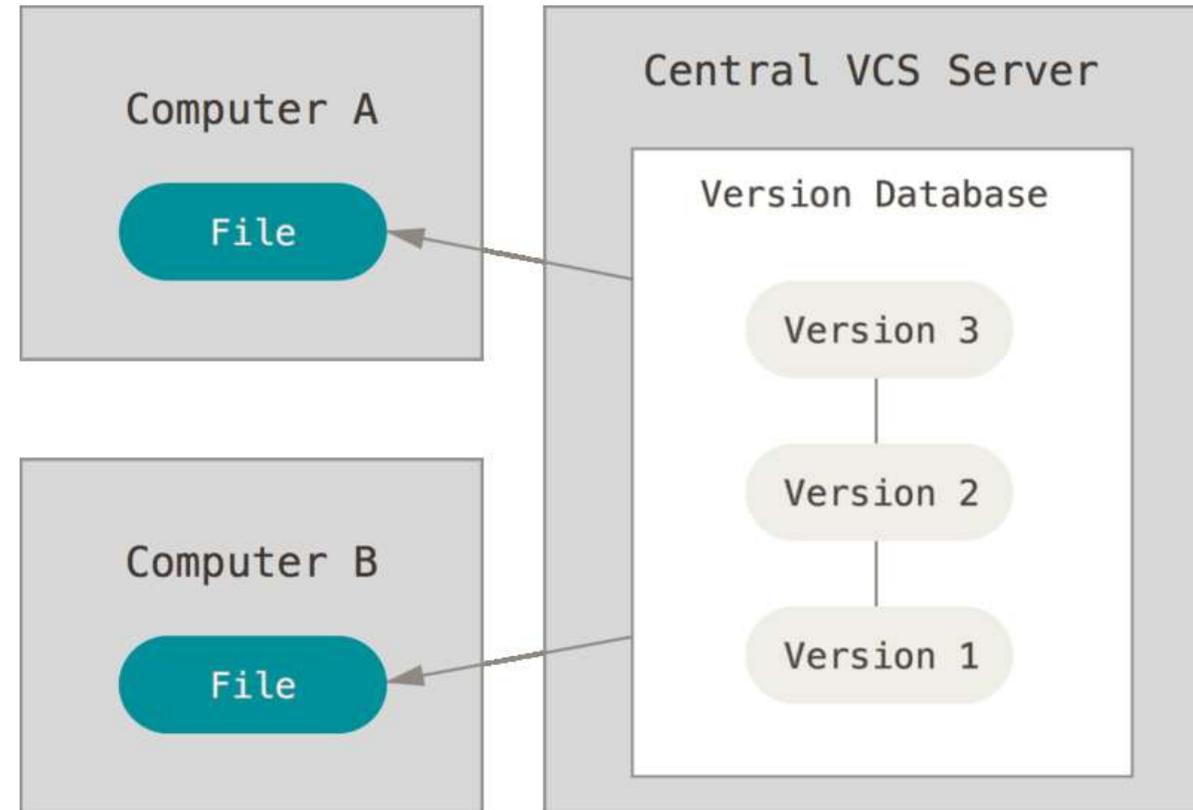
Она позволяет пересоздать любой файл на любой момент времени, последовательно накладывая патчи.



Виды систем контроля версий

Централизованные системы контроля версий

В таких системах, например [Apache Subversion \(SVN\)](#), есть центральный сервер, на котором хранятся все файлы под версионным контролем, и ряд клиентов, которые получают копии файлов из него.



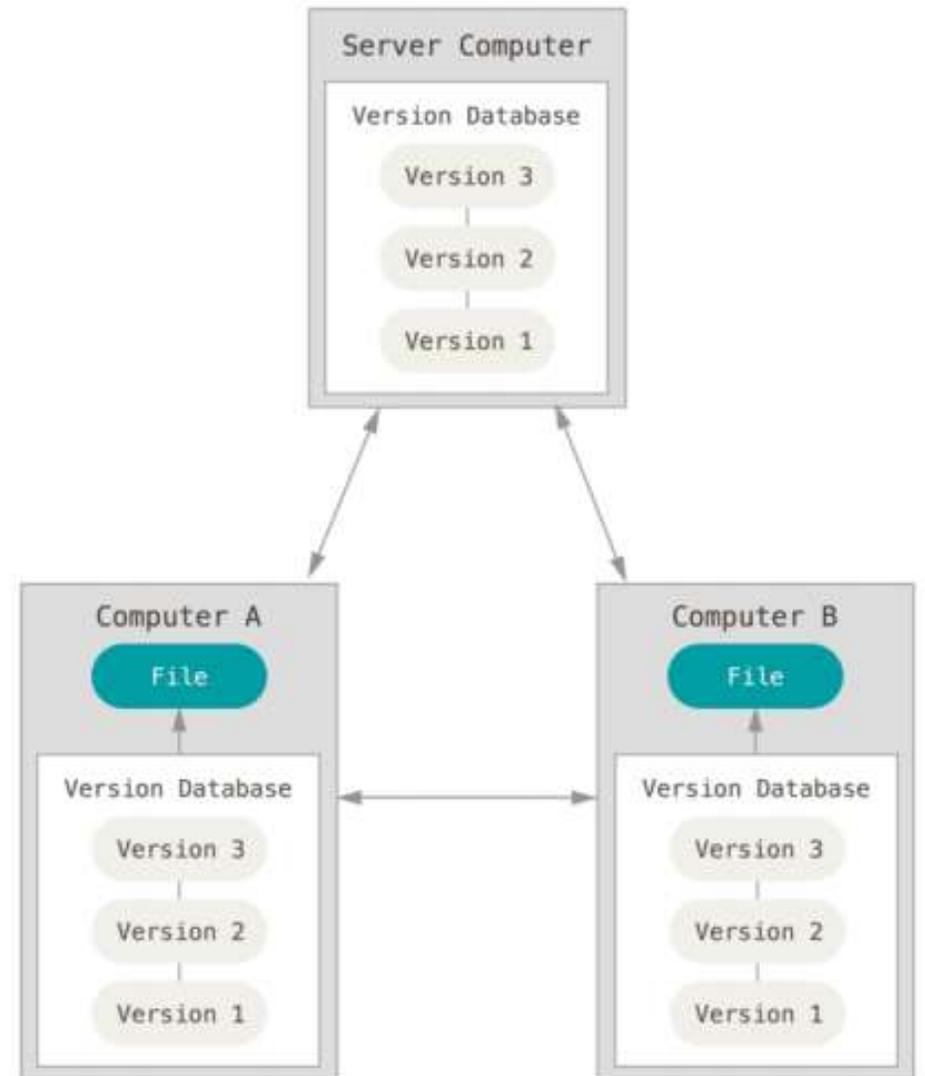
Виды систем контроля версий

Распределённые системы

контроля версий

В таких системах как [Git](#), [Mercurial](#) клиенты не просто выгружают последние версии файлов, а полностью копируют репозиторий.

При этом можно выделить центральный репозиторий, в который будут отправляться изменения из локальных и с ним же эти локальные репозитории будут синхронизироваться.



История Git

В 2002 году проект ядра Linux начал использовать проприетарную децентрализованную VCS BitKeeper.

В 2005 году отношения между сообществом разработчиков ядра Linux и коммерческой компанией, которая разрабатывала BitKeeper, прекратились, и бесплатное использование утилиты стало невозможным.

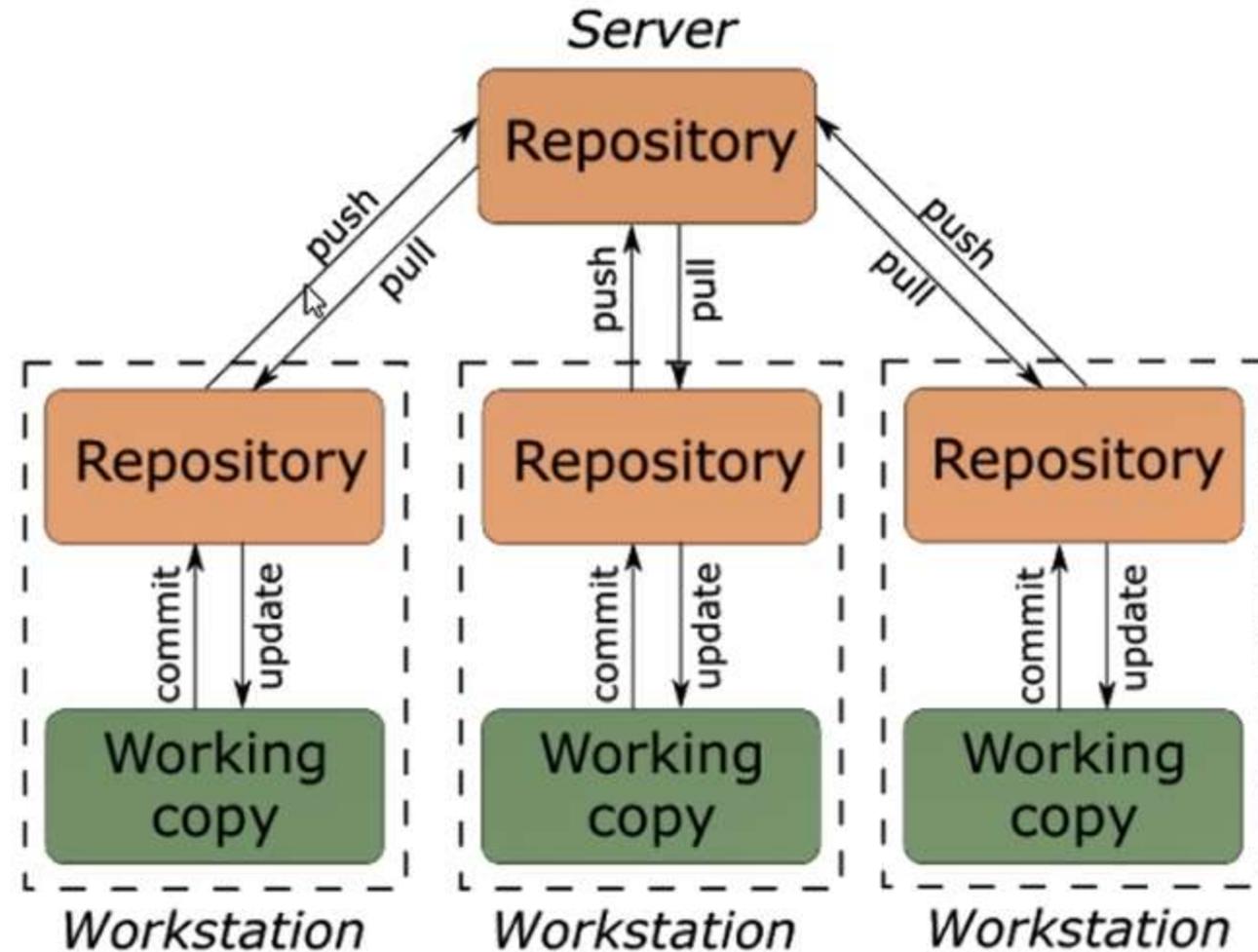
Так началась разработка Git...



Это Линус Торвальдс -инициатор Linux и фанат Nvidia

Система контроля версий Git

- Скорость
- Понятная архитектура
- Хорошая поддержка нелинейной разработки (тысячи параллельных веток)
- Полная децентрализация
- Возможность эффективного управления большими проектами (ядро Linux):
 - скорость работы и разумное использование дискового пространства



Система контроля версий **Git**

(опрос 2018)

All Respondents

Git
87.2%

Subversion
16.1%

Team Foundation Version Control
10.9%

Zip file back-ups
7.9%

Copying and pasting files to network shares
7.9%

I don't use version control
4.8%

Mercurial
3.6%

74,298 responses; select all that apply

Git

Основное отличие Git от любой другой системы контроля версий — это **подход к работе со своими данными**.

Концептуально, большинство других систем хранят информацию в виде **списка изменений в файлах**.

Эти системы (CVS, Subversion, Perforce и т.д.) представляют хранимую информацию в виде набора файлов и изменений, сделанных в каждом файле по времени (это называют контролем версий, основанным на различиях).

Git

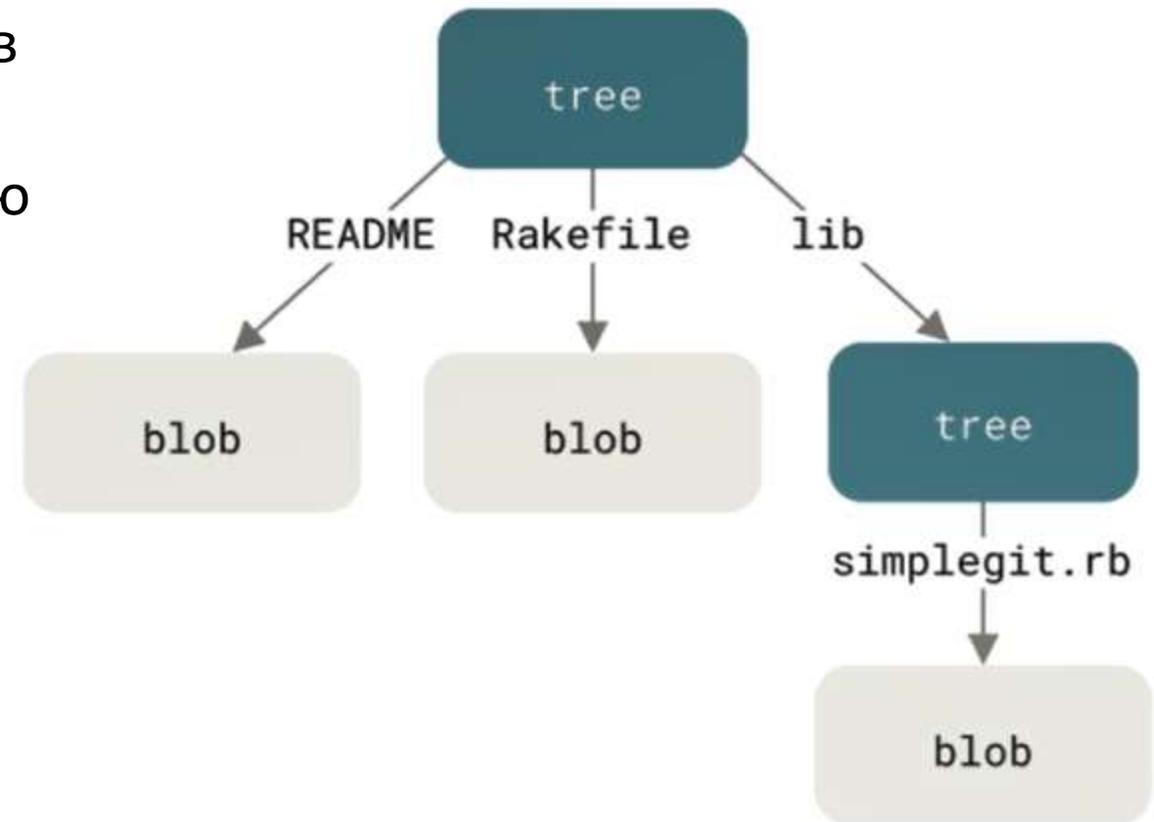
Основное отличие Git от любой другой системы контроля версий — это **подход к работе со своими данными**.

Подход Git к хранению данных больше похож на **набор снимков** мини-файловой системы.

Каждый раз, когда вы делаете коммит, т.е. сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок.

Принцип работы Git

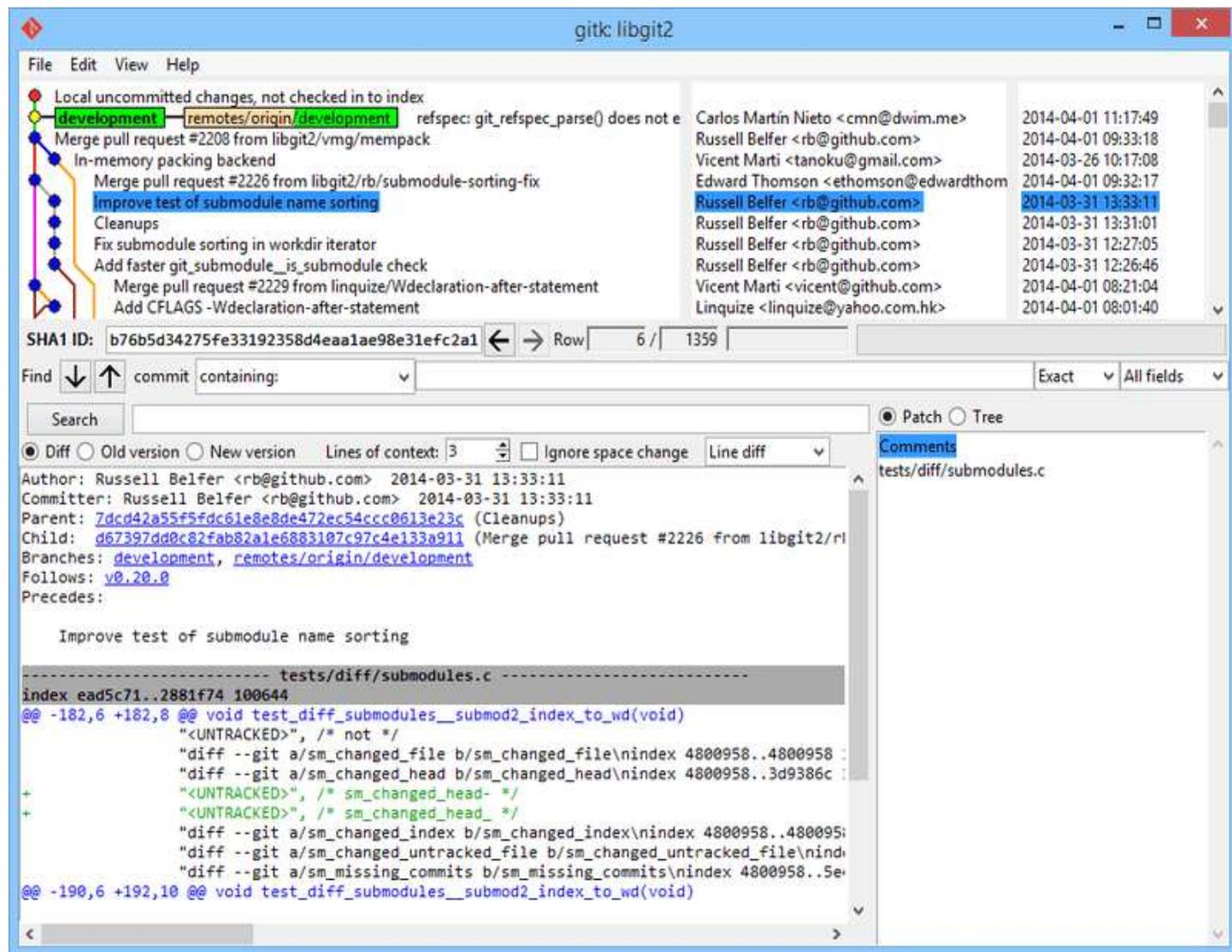
Когда файл добавляется для отслеживания в Git, он сжимается с помощью алгоритма сжатия zlib. Результат хэшируется с помощью хэш-функции SHA-1. Этот уникальный хэш соответствует содержимому в этом файле. Git хранит его в базе объектов, которая находится в скрытой папке `.git/objects`. Имя файла это сгенерированный хэш, а файл содержит сжатый контент. Такие файлы называются блобами и создаются каждый раз при добавлении в репозиторий нового файла (изменённой версии существующего файла).



Git

Git – программа имеет интерфейс командной строки (CLI), но к ней есть несколько готовых интерфейсов GUI для удобства.

Графический интерфейс инструмента gitk



Репозиторий

Чтобы git начал следить за изменениями, надо сказать ему за чем именно наблюдать (не будет же он за всеми файлами следить!)

Для этого необходимо создать репозиторий в нужной нам папке. Например, сделаем новую папку «Покупки»

\$ mkdir Покупки

перейдем в нее с помощью команды

\$ cd Покупки

создадим наш первый репозиторий

\$ git init

Теперь git сможет следить за всеми файлами в папке «Покупки»



Покупки

Создаем изменения

Создадим новый файл с покупками «Продукты» и запишем первые заказы.



Покупки



Продукты

Молоко
Яблоки
Пельмени

Отслеживаем изменения

Файл создали, а дальше
что? Как git его видит?

Чтобы это узнать, есть команда

\$ git status

которая нам скажет, что файл
«Продукты» не отслеживается,
давайте это исправим!



Покупки



Продукты

Молоко
Яблоки
Пельмени

Отслеживаем изменения

Чтобы **git** начал следить за нашим файлом надо явно добавить его с помощью команды

```
$ git add Продукты
```

Теперь команда

```
$ git status
```

покажет нам, что файл
можно закоммитить
(to be committed)

Commit - совершить ... что это значит?



Покупки



Продукты

```
+ Молоко  
+ Яблоки  
+ Пельмени
```

Репозиторий

предполагается, что вы будете хранить в нём файлы с исходным кодом и какие-нибудь дополнительные материалы - необходимую для GUI или вёрстки графику (картинки, иконки, звуки).

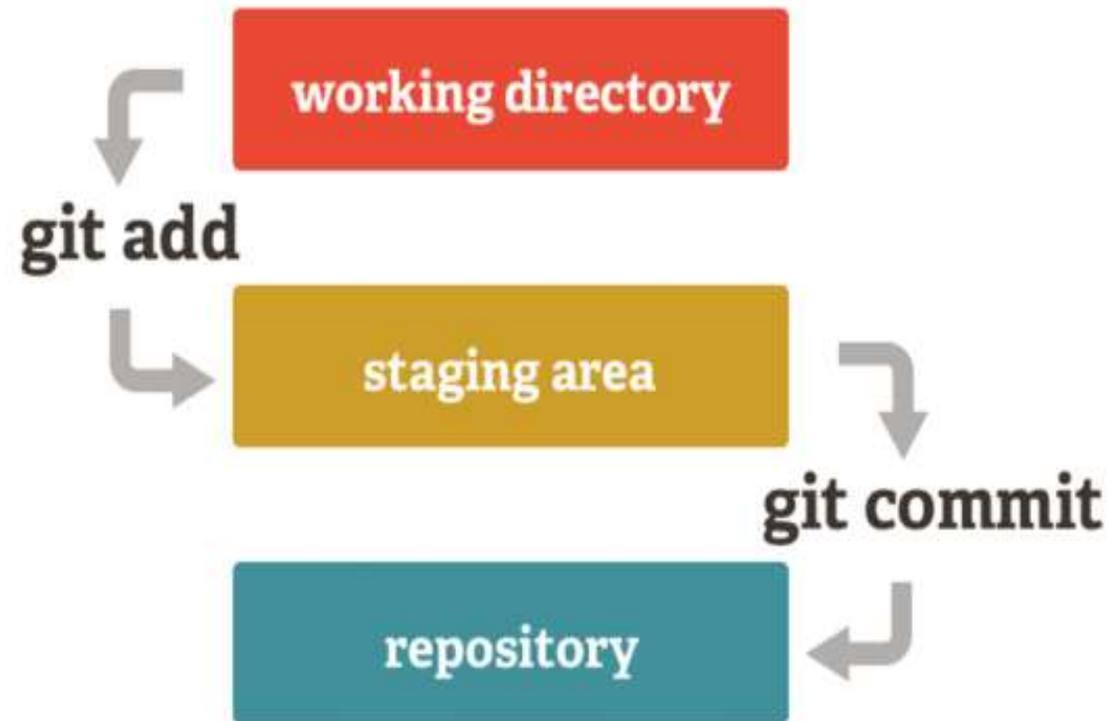
Репозитории могут быть публичными и приватными, в них можно создавать другие папки и отслеживать изменения версий.

Управлять своими репозиториями можно прямо через интерфейс сайта GitHub, командную строку, десктопное приложение GitHub или различные средства разработки (IDE).

Состояния в git

есть три основных состояния, в которых могут находиться ваши файлы:

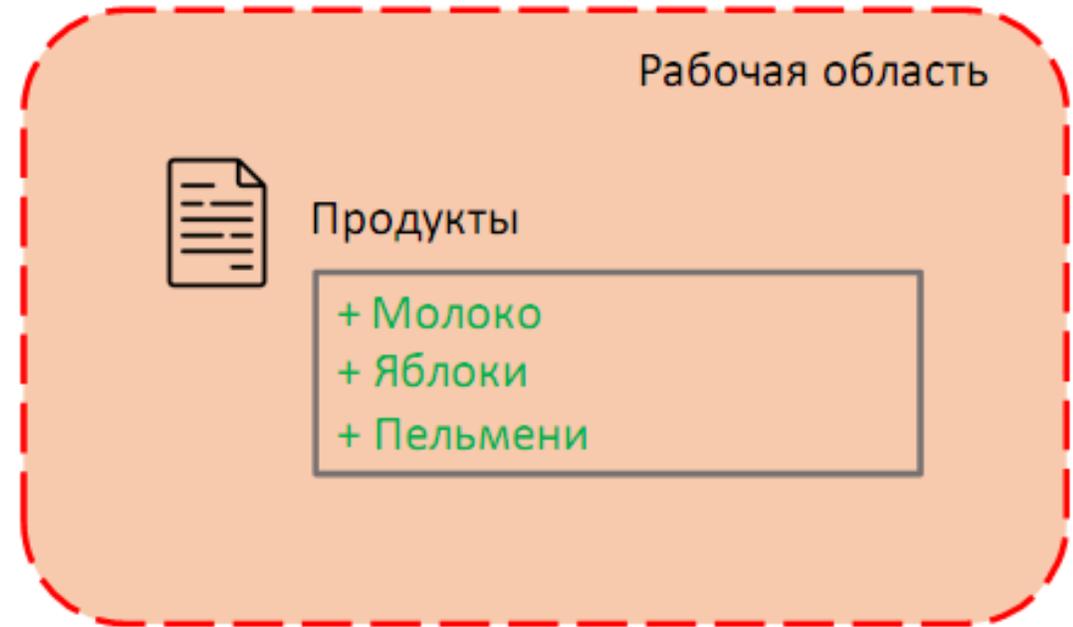
- к **изменённым** (modified) относятся файлы, которые поменялись, но ещё не были зафиксированы.
- **индексированный** (staged) — это изменённый файл в его текущей версии, отмеченный для включения в следующий коммит.
- **зафиксированный** (committed) значит, что файл уже сохранён в вашей локальной базе.



Рабочая область

или рабочая копия, или `working directory` – пространство в репозитории, является снимком одной версии проекта.

Файлы извлекаются из сжатой базы данных в каталоге `Git` и помещаются на диск, для того чтобы их можно было использовать или редактировать.



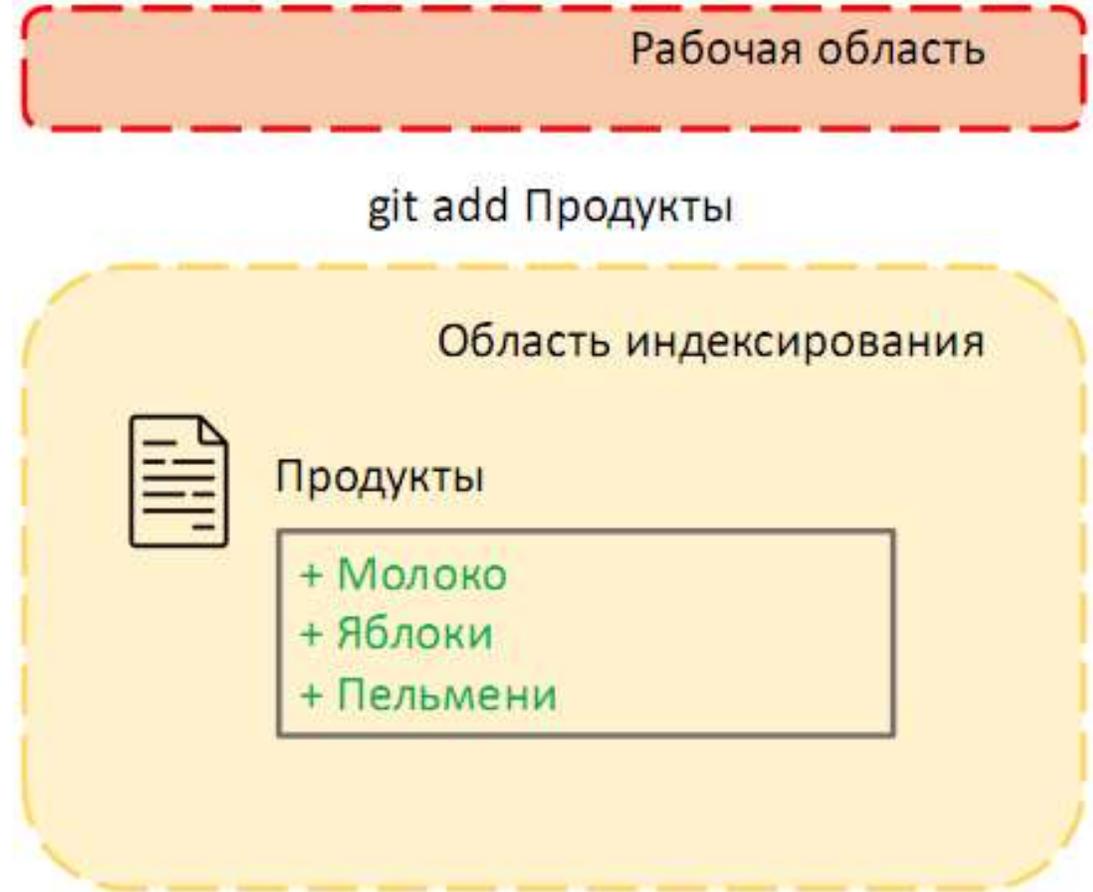
Область индексирования

Область индексирования, или индекс – заготовка для коммита, которую потом можно сохранить в истории.

Просто выберите понравившиеся изменения и добавьте их в индекс

с помощью команды

\$ **git add**



Каталог git

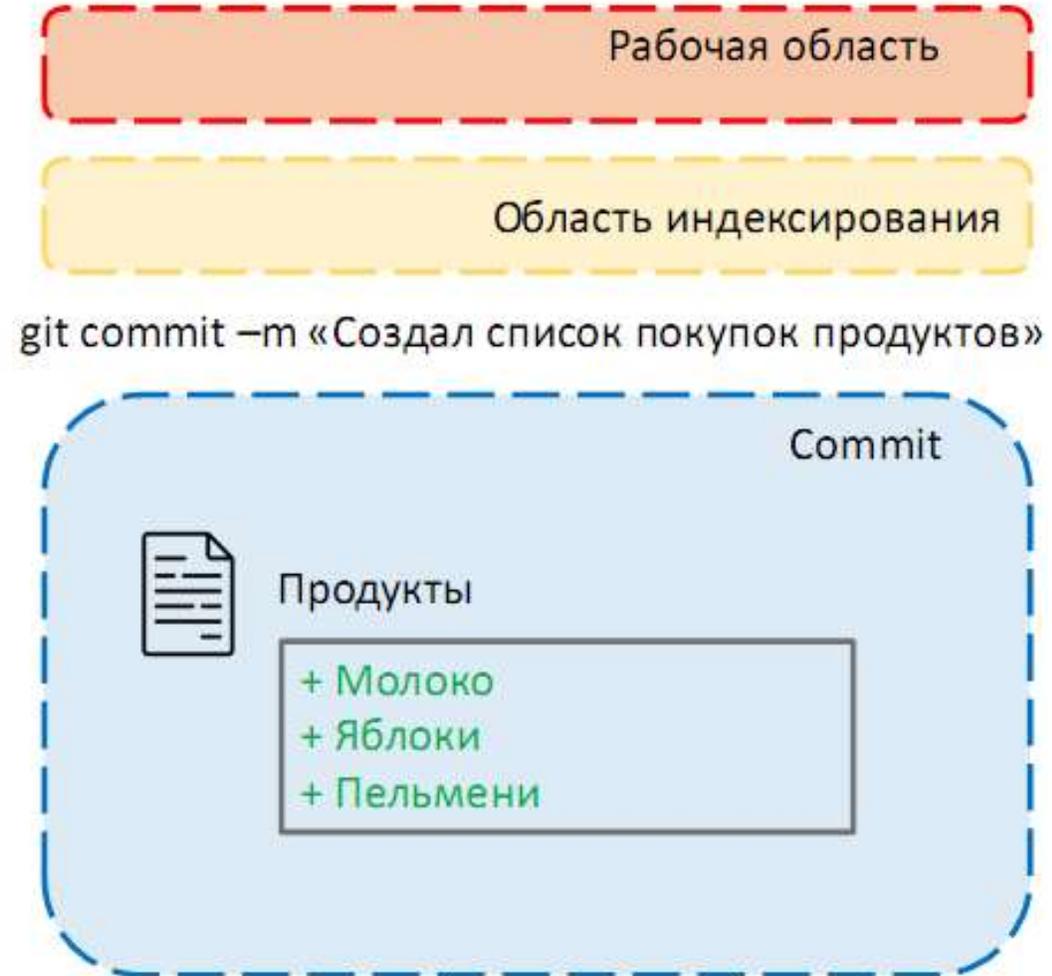
Каталог git – это цепочка сохраненных изменений (коммитов) в репозитории, а сам коммит – это и есть сохраненное состояние репозитория в какой-то момент времени.

Чтобы из индекса сделать новый коммит достаточно сделать команду

\$ git commit

И не забудьте указывать, что вы сделали в этом коммите с помощью опции –m

\$ git commit –m «Создал список покупок продуктов»

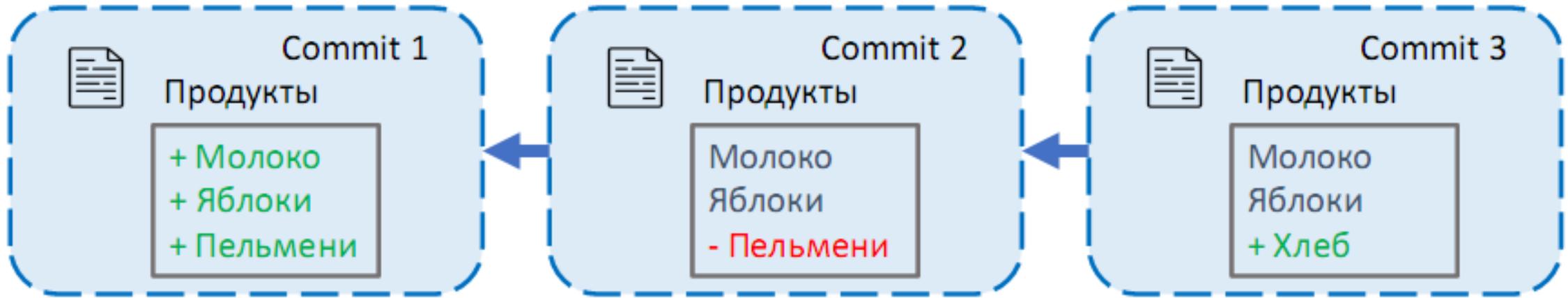


Каталог git

Базовый подход в работе с Git выглядит так:

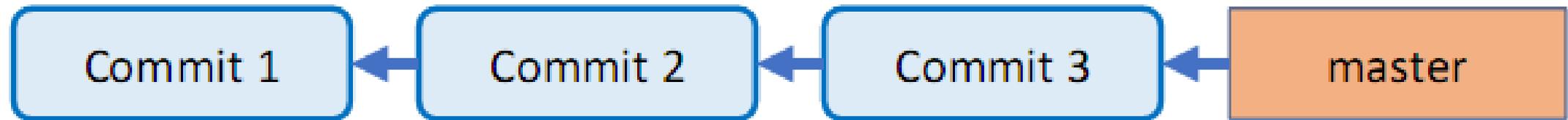
- Изменяете файлы вашей *рабочей копии*.
- Выборочно добавляете в индекс только те изменения, которые должны попасть в следующий коммит, добавляя тем самым снимки только этих изменений в *индекс*.
- Когда вы делаете коммит, используются файлы из собранного индекса, и этот снимок сохраняется в ваш *каталог Git*.

Ветка (branch)



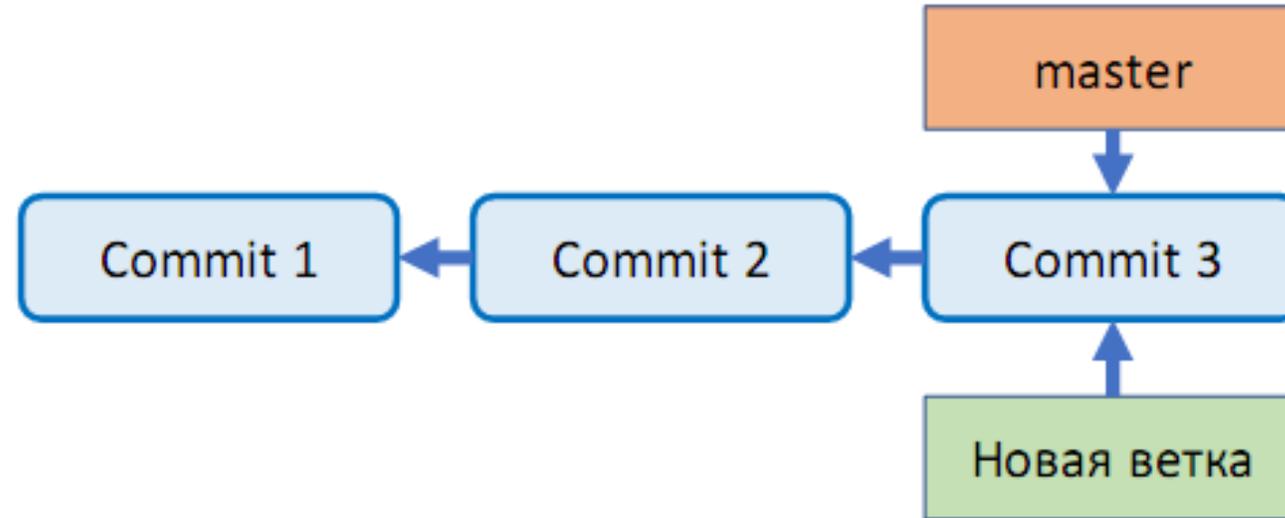
Коммиты неразрывно связаны друг с другом, последовательность коммитов называется веткой. Ветка нужна для того, чтобы понять, какие изменения нужны, чтобы получить текущую версию файла.

Основная ветка (master)



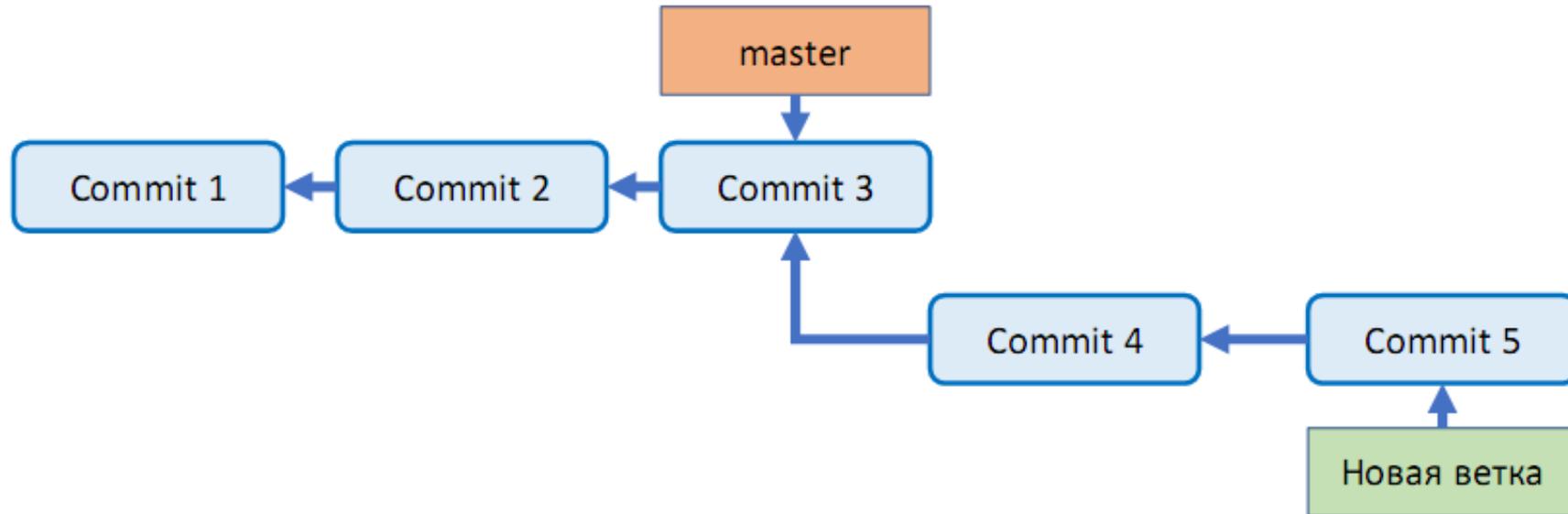
Когда вы создаете новый репозиторий, то вместе с ним создается и основная ветка, в которую и будут добавляться все новые коммиты. Обычно она называется **master** или **main**.

МНОЖЕСТВО ВЕТОК



Когда работаешь с репозиторием в одиночку, можно обойтись и одной веткой, но система контроля версий создавалась для совместной разработки кода многими людьми, поэтому для того, чтобы не мешать друг другу можно создать отдельные ветки (пути изменений) под различные нужды.

МНОЖЕСТВО ВЕТОК

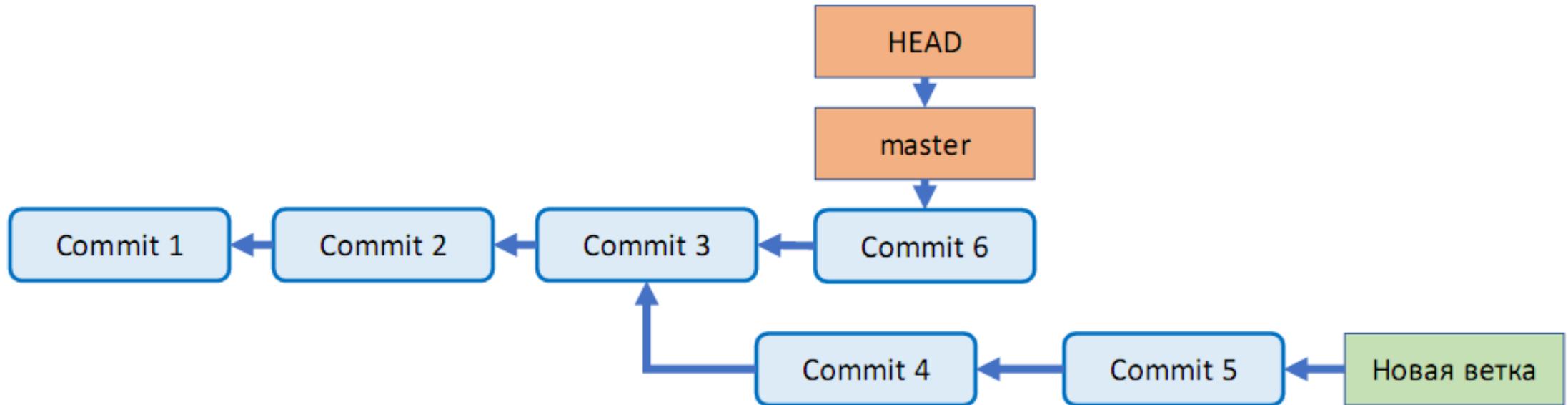


Ветку можно создать с помощью команды

`$ git branch новая_ветка`

Сама ветка по сути – это указатель на последний коммит, к которому добавятся новые правки.

Переключение веток



Существует указатель HEAD, который указывает на коммит, чье состояние сейчас развернуто в рабочей области. Те файлы, которые есть в папчке, образовались с помощью последовательности коммитов, на которую указывает HEAD.

Если переключить указатель HEAD с помощью команды `git checkout` на master или «Новая ветка», то мы получим разные цепочки изменений, а значит и разные состояния рабочих областей.

Облачный репозиторий

До текущего момента мы работали на локальной машине, в нашей папке.

Но как же делиться кодом с коллегами, как совместно решать множество задач?

Для этого существуют удаленные репозитории кода, которые-hostятся в интернете.

Из самых известных это GitHub, GitLab, GitFlic



Облачный репозиторий

Git — это программа, которую нужно установить и подключить к проекту для управления системой контроля версий.

GitHub — это сайт-хранилище для историй версий проектов: вы подключаете Git, регистрируетесь на GitHub, создаёте онлайн-репозиторий и переносите файлы с Git на GitHub.

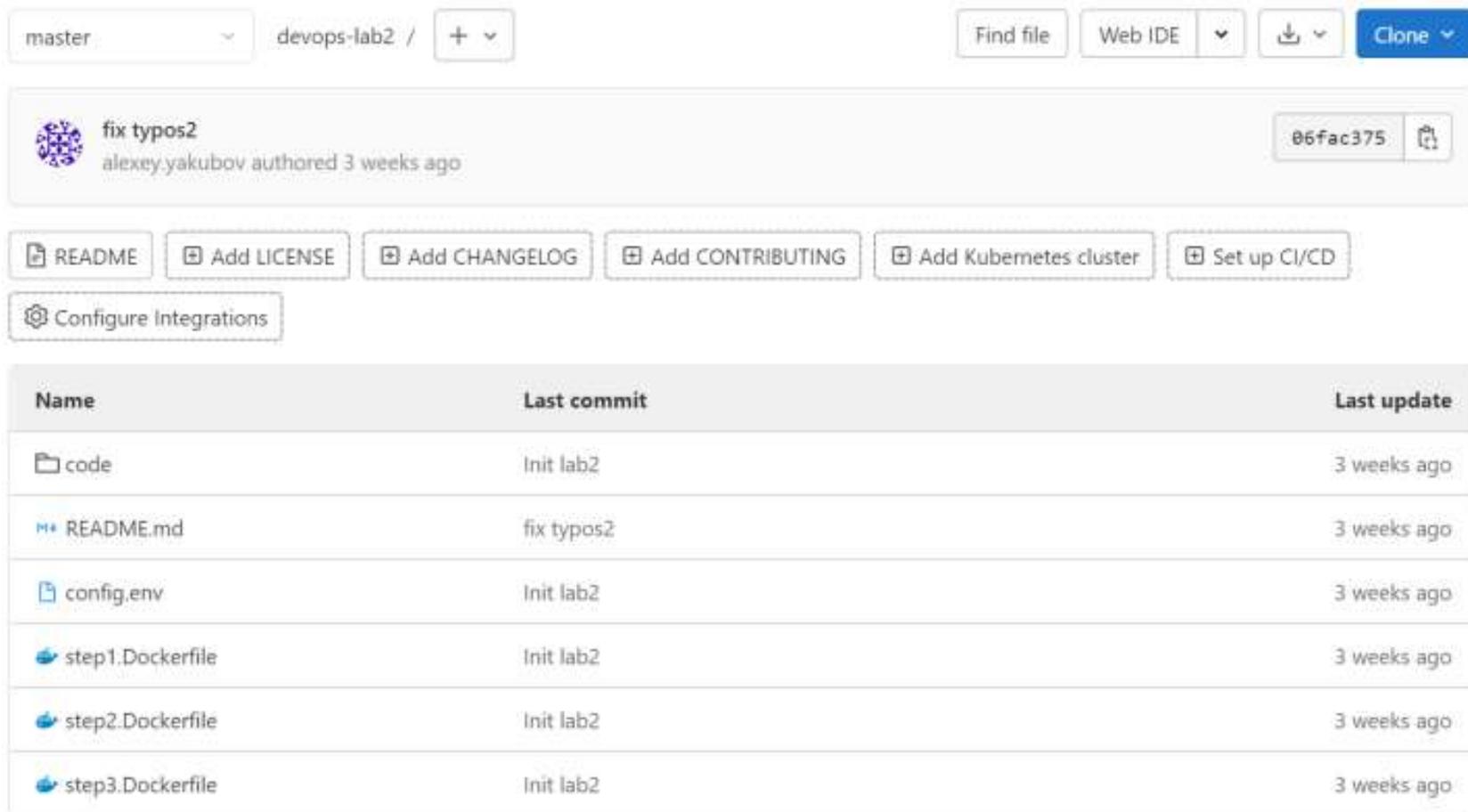
Git — это самая популярная система контроля версий, а GitHub — онлайн-хранилище кода.

Git и GitHub настроены на взаимодействие и поэтому часто используются как единый механизм работы с проектом.

Общий репозиторий

Локальный репозиторий можно загрузить в удаленный, чтобы хранить код в облаке и работать совместно с коллегами.

\$ git push



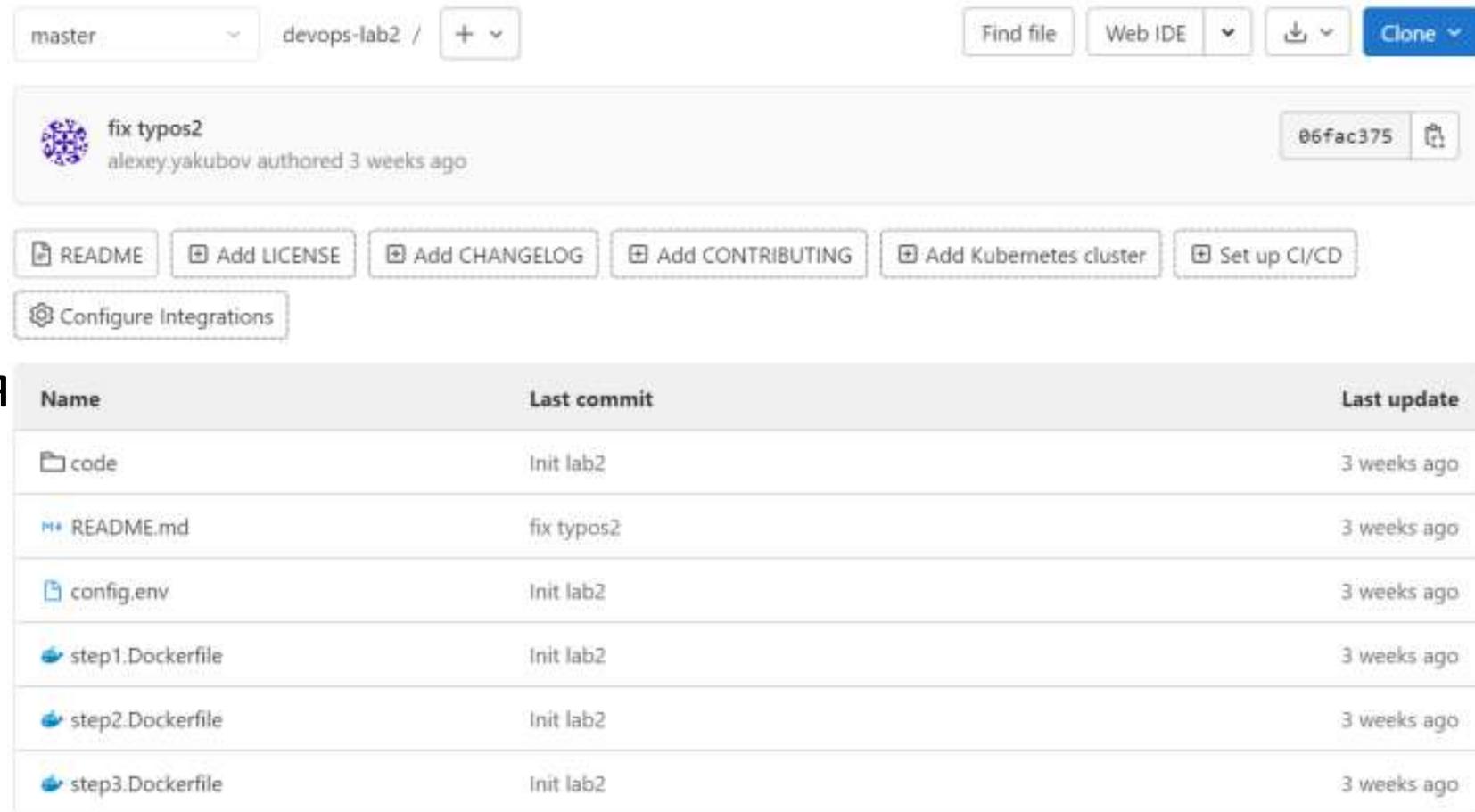
The screenshot shows a GitHub repository interface for 'devops-lab2'. At the top, there are navigation elements: a dropdown menu set to 'master', the repository name 'devops-lab2 /', a plus sign, and buttons for 'Find file', 'Web IDE', a download icon, and 'Clone'. Below this is a commit summary for 'fix typos2' by 'alexey.yakubov' from 3 weeks ago, with a commit hash '06fac375'. A row of utility buttons includes 'README', 'Add LICENSE', 'Add CHANGELOG', 'Add CONTRIBUTING', 'Add Kubernetes cluster', and 'Set up CI/CD'. A 'Configure Integrations' button is also present. The main content is a table with the following data:

Name	Last commit	Last update
code	Init lab2	3 weeks ago
README.md	fix typos2	3 weeks ago
config.env	Init lab2	3 weeks ago
step1.Dockerfile	Init lab2	3 weeks ago
step2.Dockerfile	Init lab2	3 weeks ago
step3.Dockerfile	Init lab2	3 weeks ago

Общий репозиторий

Разумеется из удаленного репозитория можно скачать все изменения, операция называется pull и выполняется с помощью команды

\$ git pull



The screenshot shows a GitHub repository page for 'devops-lab2' on the 'master' branch. The repository name is 'devops-lab2 / +'. The current commit is 'fix typos2' by 'alexey.yakubov', authored 3 weeks ago, with the commit hash '06fac375'. Below the commit information, there are several buttons for adding files and configurations: 'README', 'Add LICENSE', 'Add CHANGELOG', 'Add CONTRIBUTING', 'Add Kubernetes cluster', and 'Set up CI/CD'. There is also a 'Configure Integrations' button. The commit history table below shows the following files and their last commit details:

Name	Last commit	Last update
code	Init lab2	3 weeks ago
README.md	fix typos2	3 weeks ago
config.env	Init lab2	3 weeks ago
step1.Dockerfile	Init lab2	3 weeks ago
step2.Dockerfile	Init lab2	3 weeks ago
step3.Dockerfile	Init lab2	3 weeks ago

Общий репозиторий

Скопировать репозиторий для внесения изменений в копию можно двумя основными способами:

- клонировать (clone) — то есть просто скопировать на локальный компьютер или сервер;
- форкнуть (от английского fork — развилка) — сделать отдельную копию репозитория (обычно чужого) для продолжения разработки «по другому пути развилки».

Общий репозиторий

Если вы форкнули чужой проект, чтобы предложить автору конкретные улучшения, нужно по готовности «запулить» их в исходный репозиторий, то есть сделать запрос на изменения

pull request

Команды для GitHub CLI начинаются с сокращения `gh` —

gh repo clone.

Ведение проекта

Основное описание вашего проекта задаётся в файле `Readme.md`, который можно создать сразу в репозитории или после.

Расширение ***md*** — сокращение от названия популярного языка упрощённой (проще чем `html`) разметки текста — Markdown.

Содержимое файла `Readme` отображается на главной странице репозитория и отвечает на вопрос, что это за проект, чем он может быть полезен другим разработчикам и как им пользоваться.

Ведение проекта

Чтобы оформить Readme стильно, можно почитать руководство по markdown-разметке.

В оформлении *md* можно использовать — заголовки разных уровней, выделение жирным/курсивом, изображения, эмодзи, ссылки, диаграммы, графики и так далее. Файл Readme может быть довольно длинным, но всё же для оформления большой документации GitHub рекомендует создать «Вики» (wiki)

Ведение проекта

На GitHub можно заhostить сайт с помощью функции GitHub Pages.

Это просто:

- Зайдите в настройки репозитория.
- В блоке Code and automation выберите Pages.
- Выберите источник (Deploy from a branch, затем нужную ветку).
- Кликните на Save. Обновите страницу, и вверху страницы появится ссылка на ваш новый сайт.

Ведение проекта

Сервисом Git пользуются все: это один из важных общих навыков вне зависимости от выбранного вами языка программирования и направления разработки.

И, как уроки ОБЖ, тот же ***git clone*** когда-нибудь вас спасёт.

Поэтому важно начинать пользоваться Git как можно раньше — хотя бы даже для бэкапов учебного кода, и уже скоро это станет полезной привычкой.

Merge request

Update README.md

Edit Code 

 Open Yakubov Alexey requested to merge `new-branch` into `master` just now

Overview 0 Commits 1 Pipelines 0 **Changes 1**

 Compare `master` and latest version

 1 file +2 -1 

Search (e.g. *.vue) (Ctrl+P)

M+ README.md

+2 -1 

▼ **README.md** 

+2 -1 Viewed 

↑ @@ -132,4 +132,5 @@ docker run --link
mysql -p 8000:8000 --env-file=config.env
step3

↑ @@ -132,4 +132,5 @@ docker run --link
mysql -p 8000:8000 --env-file=config.env
step3

132 1. Перезапустите сборку собранного
образа, оцените время пересборки,
объясните причины
133 2. К какому число слоев стремиться в
образе, правила оптимизации
134 3. Опишите базовые команды Dockerfile,
что они делают, где смотреть
документацию?

132 1. Перезапустите сборку собранного
образа, оцените время пересборки,
объясните причины
133 2. К какому число слоев стремиться в
образе, правила оптимизации
134 3. Опишите базовые команды Dockerfile,
что они делают, где смотреть
документацию?

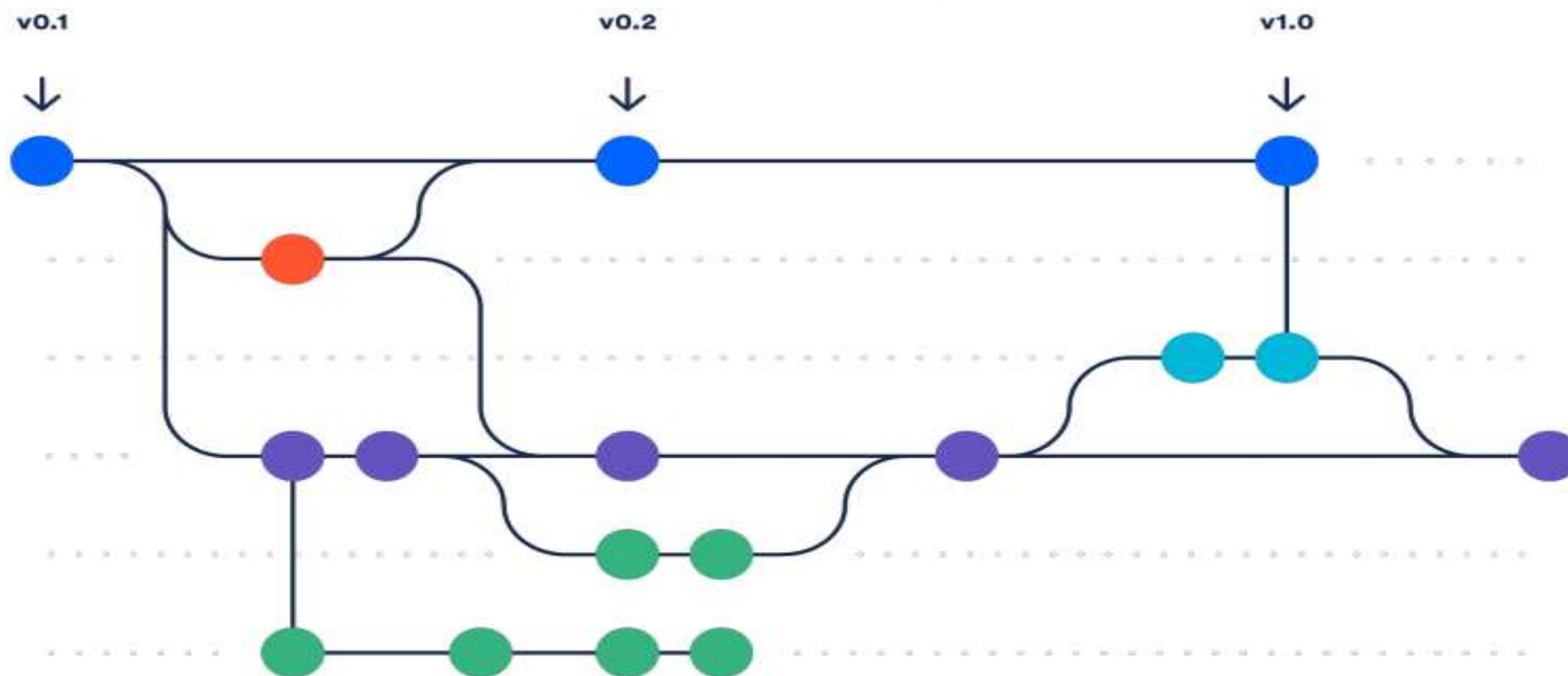
135 - 4. Что такое контекст сборки, как его
оптимизировать?

135 + 4. Что такое контекст сборки, как его
оптимизировать?

\ No newline at end of file

136 + 5. Как работает Merge Request?

Иллюстрация контроля версий




MASTER


HOTFIX


RELEASE

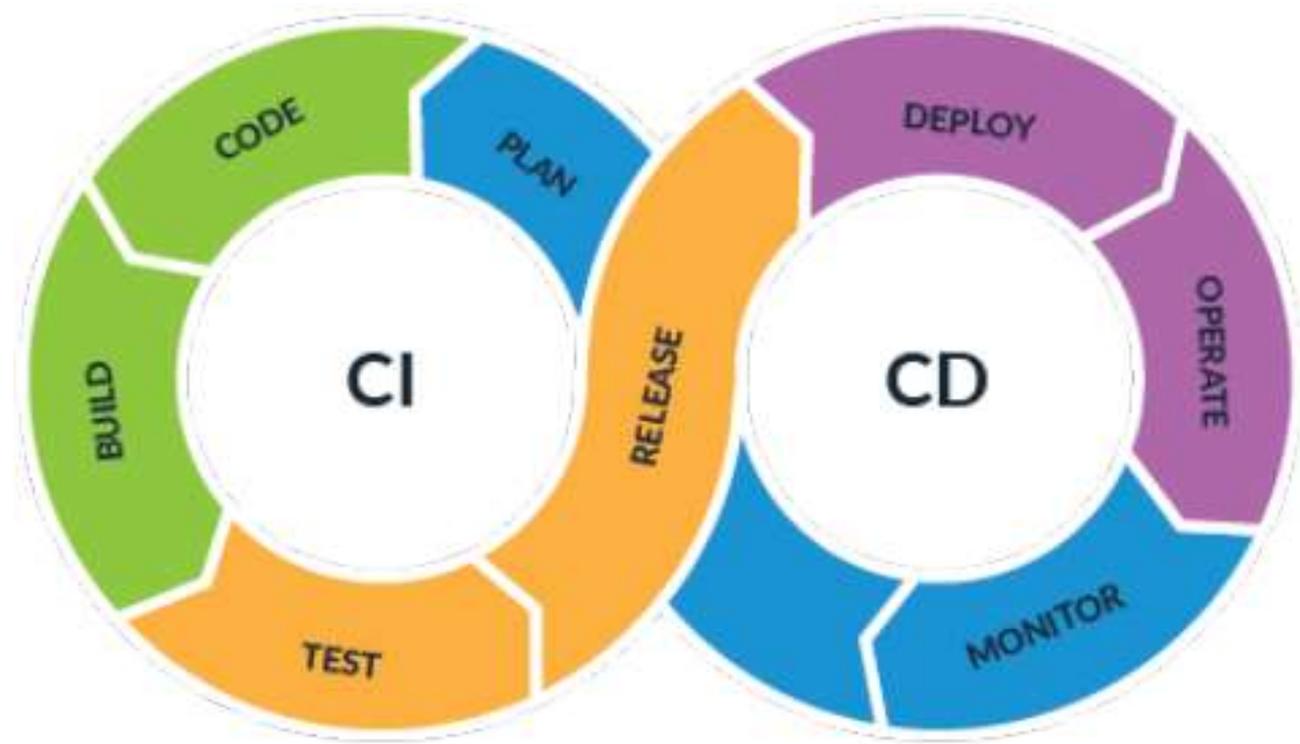

DEVELOP


FEATURES

CI/CD

Удаленный репозиторий кода позволяет настраивать автоматические действия, которые выполняются при появлении нового кода.

Это позволяет автоматически проверять новый код и даже автоматически доставлять новый код пользователям



SaaS



Self-hosted

Бесплатный

Подходит для open source проектов и частных репозиториях с командой до 5 человек.

0 руб/мес
за пользователя

- Базовый функционал Git
- Git LFS
- API/Webhooks
- Code Owners
- Merge request approvals

Бесплатный

Подходит для работы на собственном сервере с неограниченным количеством пользователей.

0 руб/мес
за пользователя

- Базовый функционал Git
- Git LFS
- API/Webhooks
- CI/CD
- Wiki



Search or jump to...

[Pull requests](#) [Issues](#) [Codespaces](#) [Marketplace](#) [Explore](#)



[DrKLO / Telegram](#) Public

[Watch](#) 1.2k [Fork](#) 7.1k [Star](#) 21k

[Code](#) [Pull requests](#) 227 [Actions](#) [Projects](#) [Security](#) [Insights](#)

master [2 branches](#) [88 tags](#)

[Go to file](#) [Add file](#) [Code](#)

About

Telegram for Android source

[telegram](#)

[Readme](#)

[GPL-2.0 license](#)

[21k stars](#)

[1.2k watching](#)

[7.1k forks](#)

Releases

[Update to 9.1.0 \(2885\)](#) [Latest](#)

6 days ago

[+ 75 releases](#)

Packages

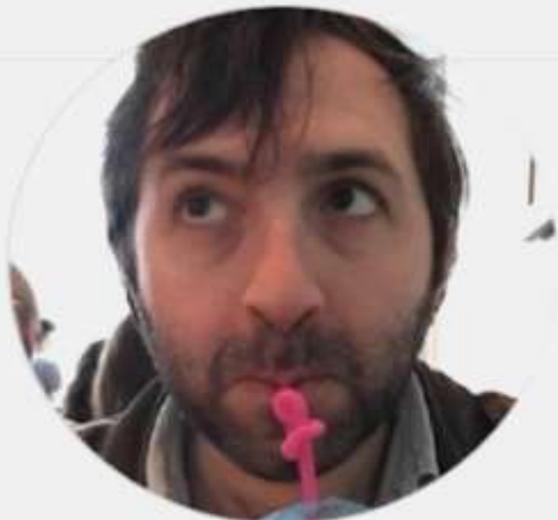
No packages published

xaxtix update to 9.1.2

23118a4 5 days ago

[462 commits](#)

TMessagesProj	update to 9.1.2	5 days ago
TMessagesProj_App	update to 9.0.2	2 months ago
TMessagesProj_AppHockeyApp	update to 9.0.2	2 months ago
TMessagesProj_AppHuawei	update to 9.0.0	2 months ago
Tools	Some new features and bug fixes	9 years ago
gradle/wrapper	Update to 8.0.0 (2406)	15 months ago
.gitignore	Update to 5.13.0 (1818)	3 years ago
Dockerfile	update to 8.9.0	3 months ago
LICENSE	first commit	9 years ago
README.md	Update README.md	3 years ago
apkdiff.py	Fix apkdiff not checking entire files	3 years ago
build.gradle	update to 8.9.0	3 months ago



Stephen Celis

stephencelis

Follow

Working on @pointfreeco:

<https://www.pointfree.co/>

3.1k followers · 36 following

@pointfreeco

Brooklyn

me@stephencelis.com

<http://stephencelis.com>

Highlights



Pinned

[pointfreeco/swift-composable-architecture](#) Public

A library for building applications in a consistent and understandable way, with composition, testing, and ergonomics in mind.

Swift 7.4k 778

[pointfreeco/swift-snapshot-testing](#) Public

Delightful Swift snapshot testing.

Swift 2.9k 390

[kickstarter/ios-oss](#) Public

Kickstarter for iOS. Bring new ideas to life, anywhere.

Swift 8k 1.2k

[pointfreeco/pointfreeco](#) Public

The source for www.pointfree.co, a video series on functional programming and the Swift programming language.

Swift 945 82

[pointfreeco/swift-overture](#) Public

A library for function composition.

Swift 1.1k 55

[SQLite.swift](#) Public

A type-safe, Swift-language layer over SQLite3.

Swift 8.6k 1.4k

1,725 contributions in the last year



Learn how we count contributions

Less ■ ■ ■ ■ More

intros22

Обзор проекта

Информация

Активность

Релизы

Репозиторий

Обсуждения 0

Запросы на слияние 0

Операции

Пакеты и реестры

Аналитика

Участники

Настройки

Михаил Черненкоий > intros22



intros22

ID проекта: 6367



В избранные 0

Fork 0

83 Коммита 3 Ветки 0 Тегов 256 КБ Файлов 256 КБ в Хранилище

master

intros22 / +

История

Найти файл

Web IDE



Клонировать



Update forma101.md

Михаил Черненкоий создал 1 day ago

95c0d0c7



README

Добавить LICENSE

Добавить CHANGELOG

Добавить CONTRIBUTING

Настройка CI/CD

Наименование	Последний коммит	Последнее обновление
README.md	Update README.md	1 month ago
forma101.md	Update forma101.md	1 day ago
start.html	Обновить start.html	1 month ago
statediag.md	Update statediag.md	4 days ago

README.md

Пример документа Markdown with Mermaid

Полезные ссылки

<https://git-scm.com/book/ru/v2>

[https://ru.hexlet.io/courses/intro to git](https://ru.hexlet.io/courses/intro%20to%20git)

<https://smartiqa.ru/courses/git>

Бесплатный курс

Введение в Git

48898 студентов [4358 сообщений](#)

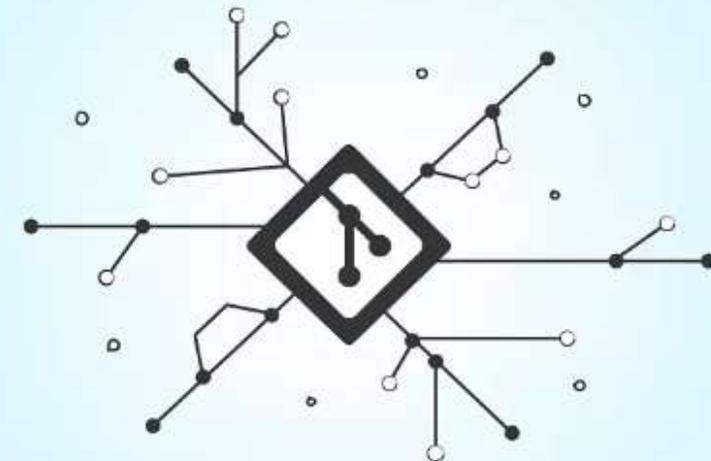
🔗 Последнее обновление: 03 марта 2023

Включено в курс

- ✓ 15 уроков (видео и/или текст)
- ✓ 67 проверочных тестов
- ✓ Дополнительные материалы
- ✓ 18 упражнений в тренажере
- ✓ Самостоятельная работа
- ✓ Помощь в «[Обсуждениях](#)»

Чему вы научитесь

- Вести разработку в соответствии с современными инженерными практиками
- Эффективно управлять исходным кодом, добавлять в общее хранилище, анализировать историю и изменять ее
- Работать с GitHub и контрибьютить в открытые проекты



Бесплатный курс

Зарегистрироваться

- 📺 Тренажер с практикой
- ✅ Бессрочный доступ к теории ⓘ
- 👤 Асинхронный формат обучения ⓘ