Git для самых маленьких

Предыстория

Однажды Петя захотел написать телеграмм бота

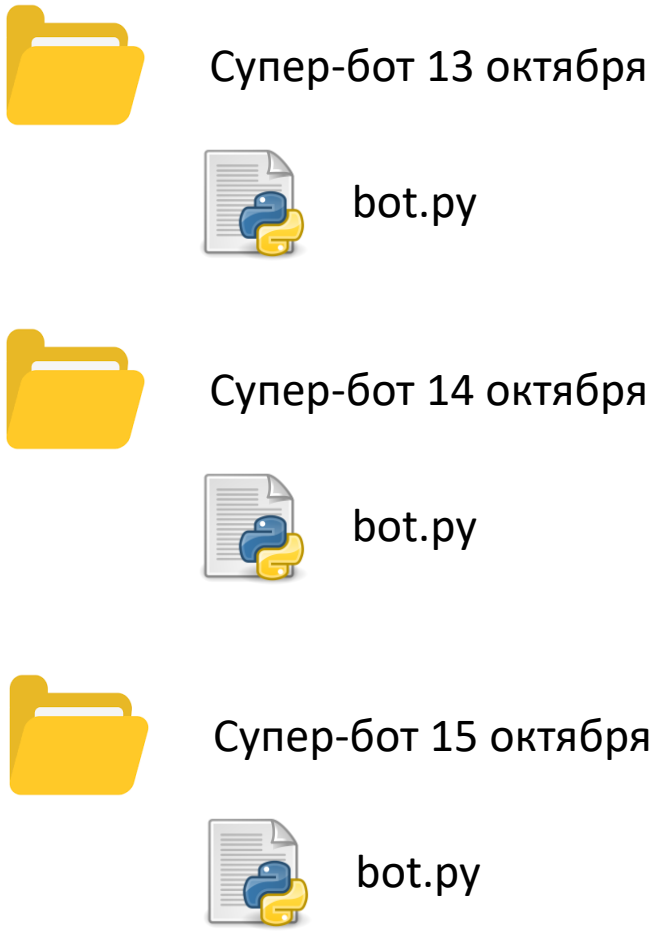
```python
import asyncio
import telegram


async def main():
    bot = telegram.Bot("TOKEN")
    async with bot:
        print(await bot.get_me())


if __name__ == '__main__':
    asyncio.run(main())
```

bot.py

Предыстория

Но он постоянно придумывал что-то новое, менял старое и часто не мог вспомнить, что же он писал раньше…

Тогда он придумал копировать проект каждый день, чтобы можно было подсматривать то, как было раньше

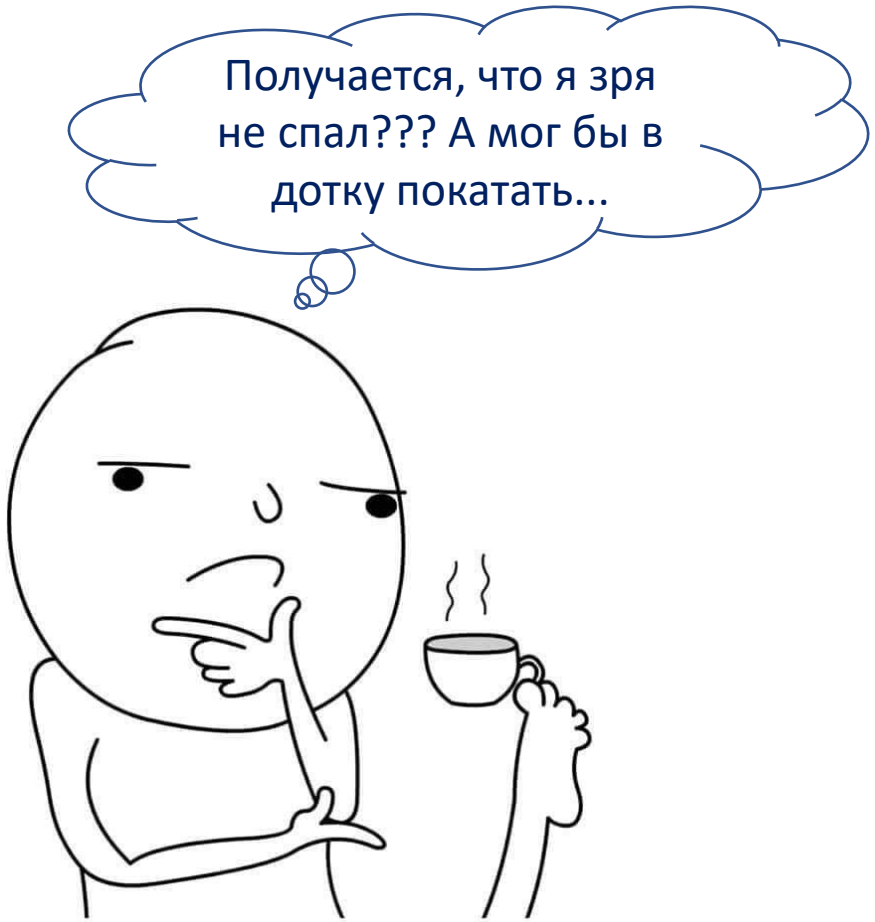


Предыстория

Но он однажды папочек стало так много, что он в них запутался =(

Предыстория

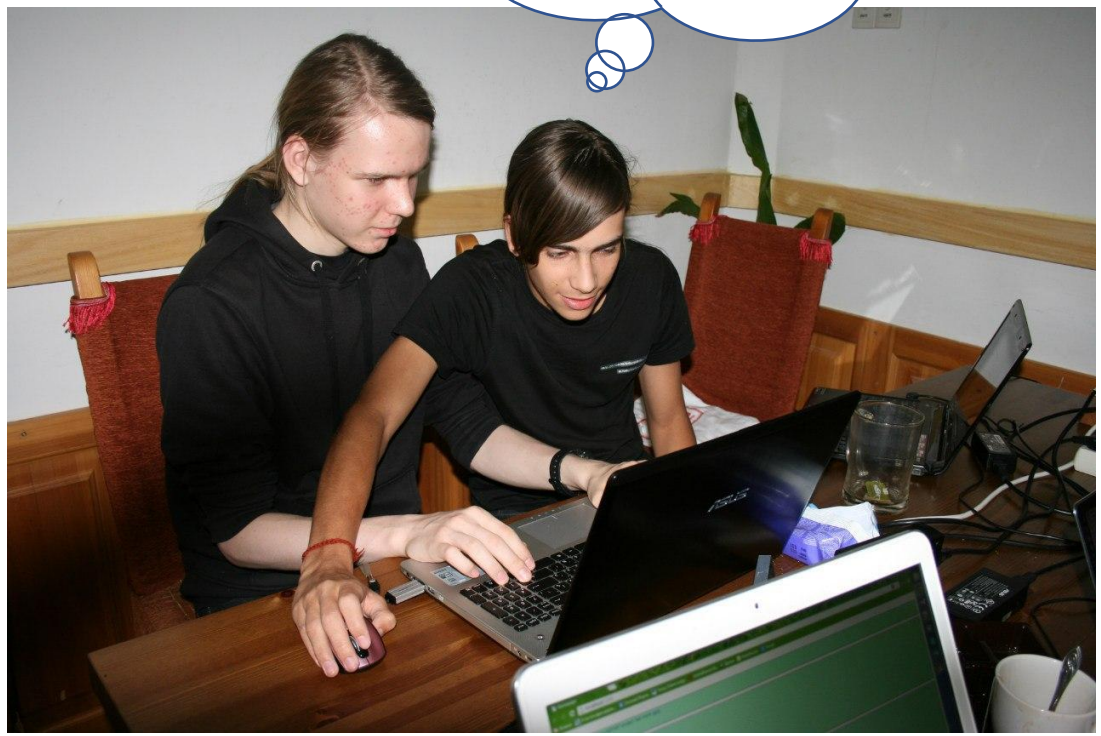
Более того он обнаружил, что последнюю версию бота, которую он делал поздно ночью, он потерял 😔



Предыстория

А еще друг Федя захотел помочь с разработкой бота. Но дома уже есть сестра и кошка.

Места для Феди нет 😣

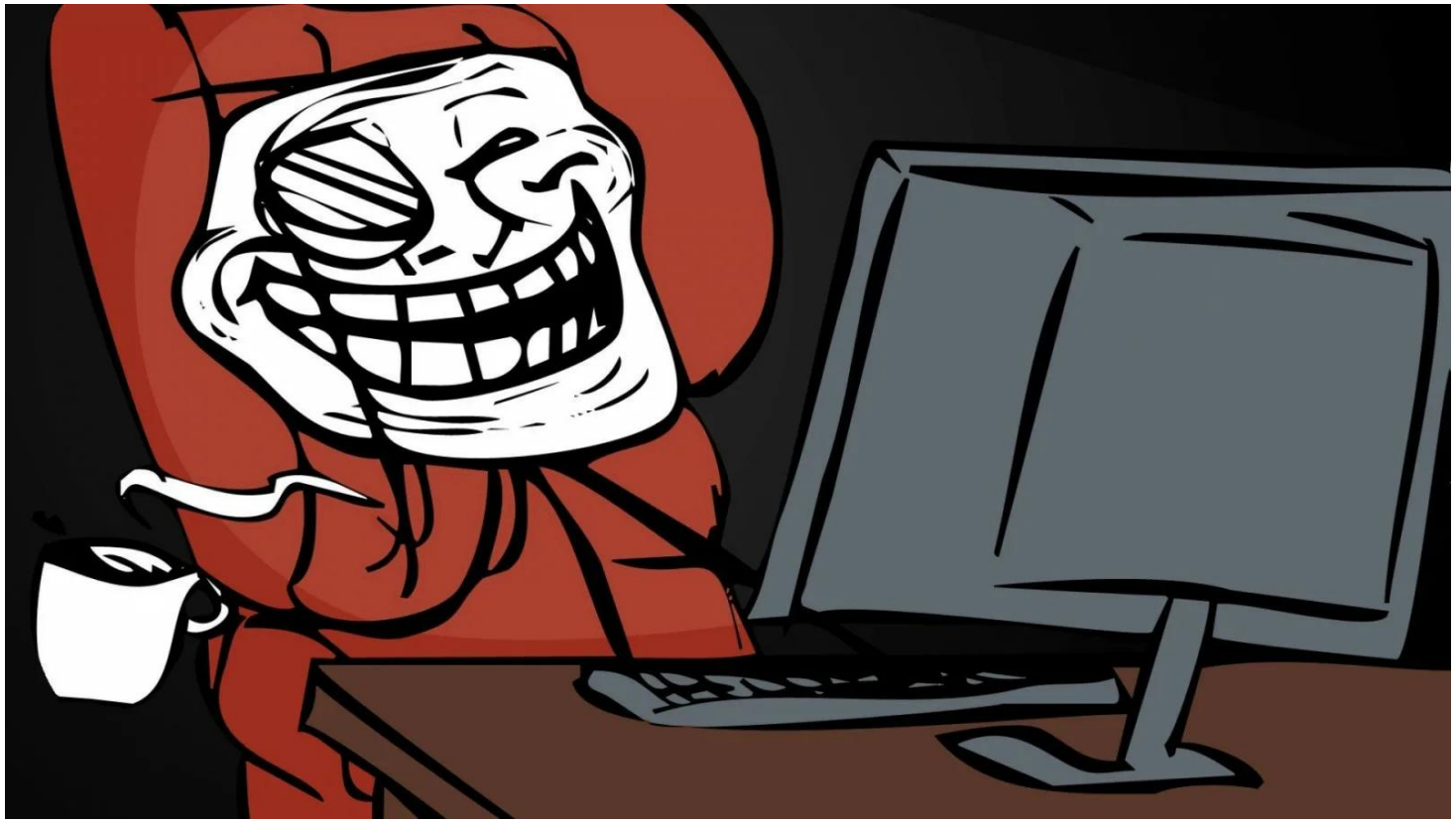


Предыстория

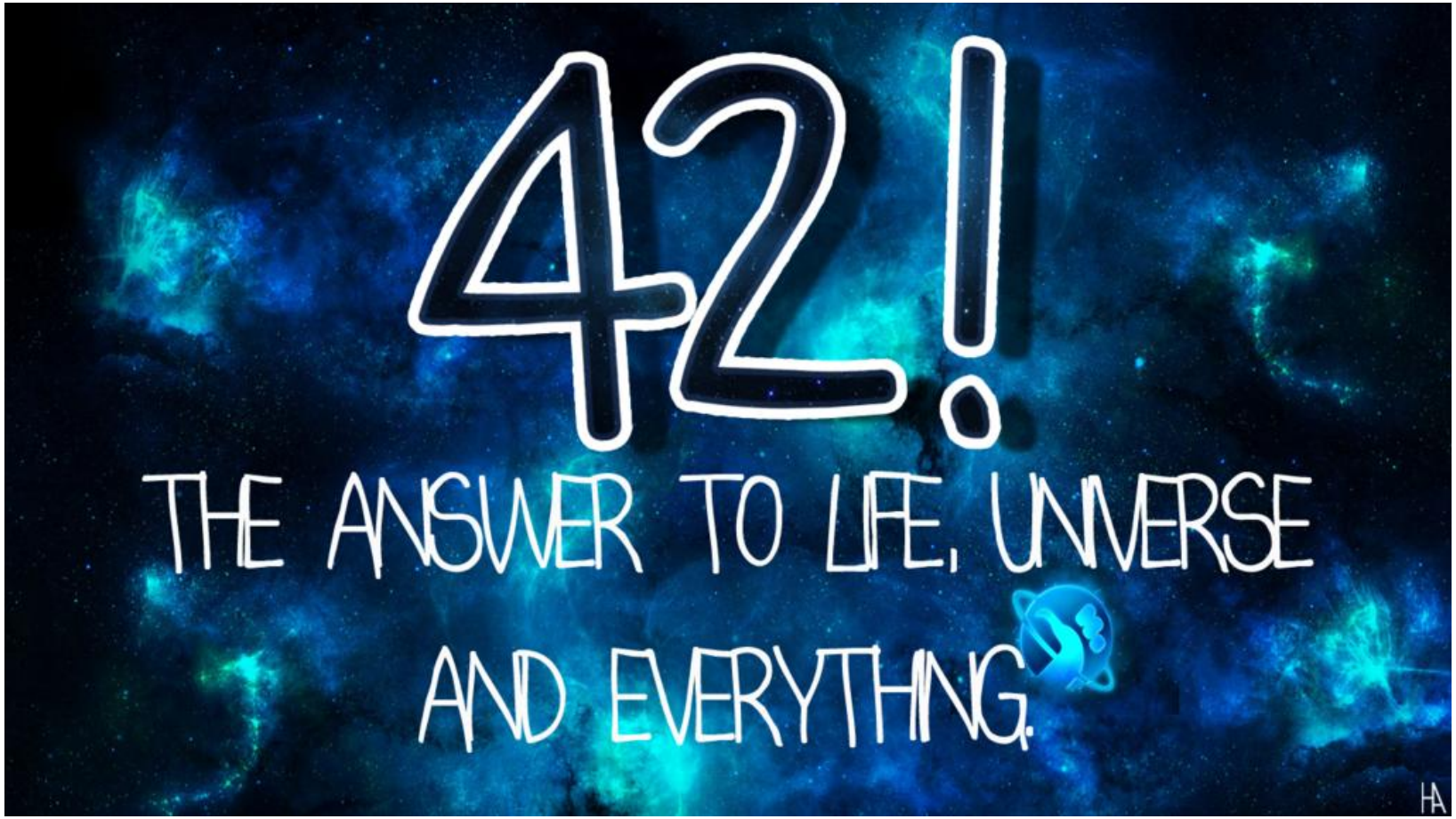
И тогда он пошел серфить в интернет. Там он хотел найти ответы на вопросы "как хранить и не терять код" и "как делиться кодом с товарищем"?

Предыстория

И тогда он нашел ответ на все вопросы...



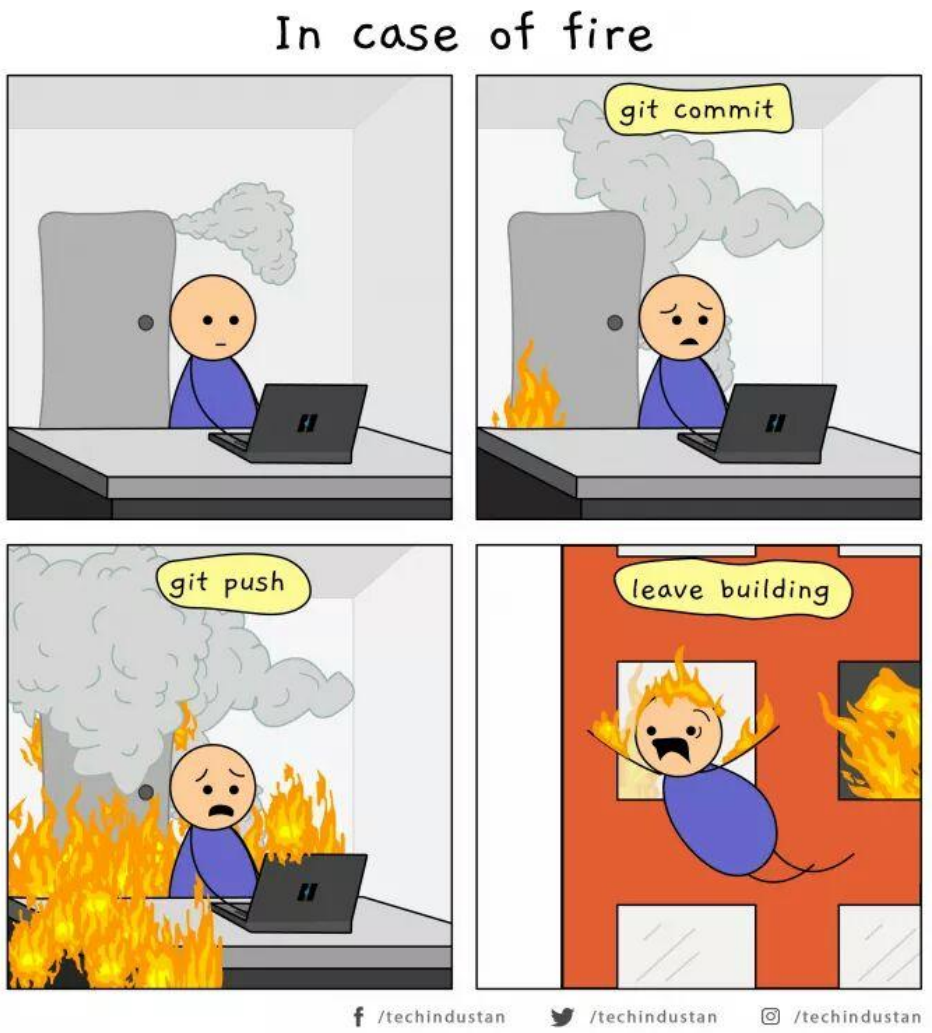
Предыстория



P.S. Вообще, это ответ на главный вопрос жизни, вселенной и всего такого

Предыстория

На самом деле он нашел, что решением его проблем является **«Система контроля версий»**



Контроль версий

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

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

Преимущества SVC

Петя зацепился за следующие возможности системы контроля версий:

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

Управление исходным кодом

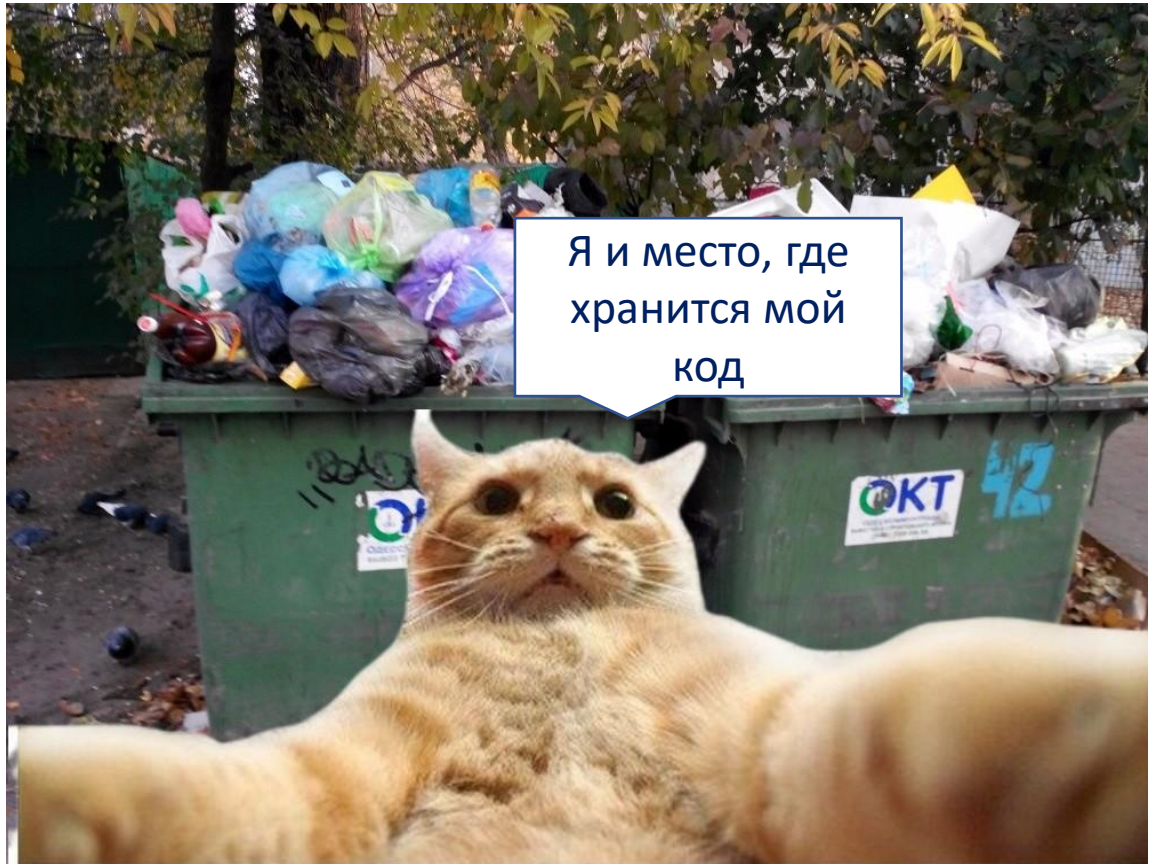
<u>Управление исходным кодом</u> *(англ. source control management, SCM)* используется для отслеживания изменений в *репозитории исходного кода*.

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

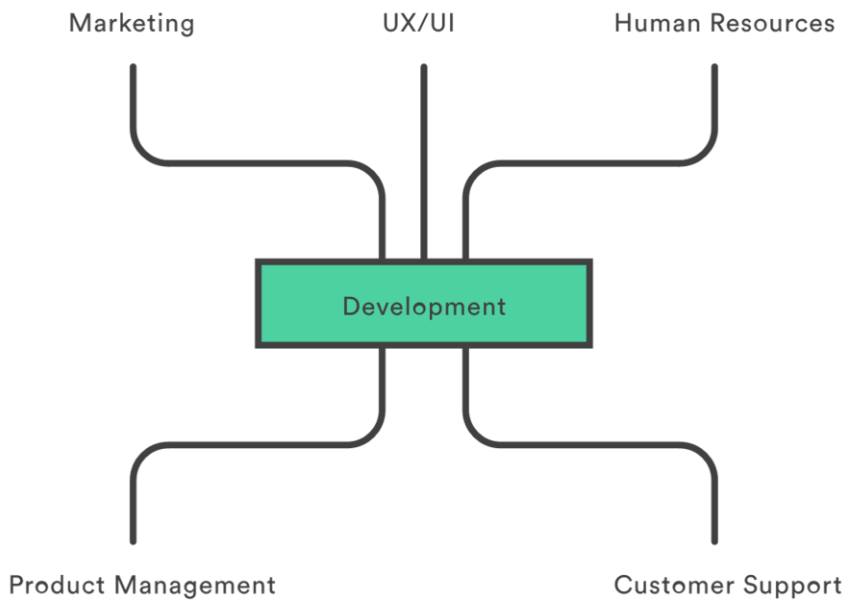
Репозиторий

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 файлы журналов конфигураций и операций, выполняемых над репозиторием, а также сами контролируемые файлы.



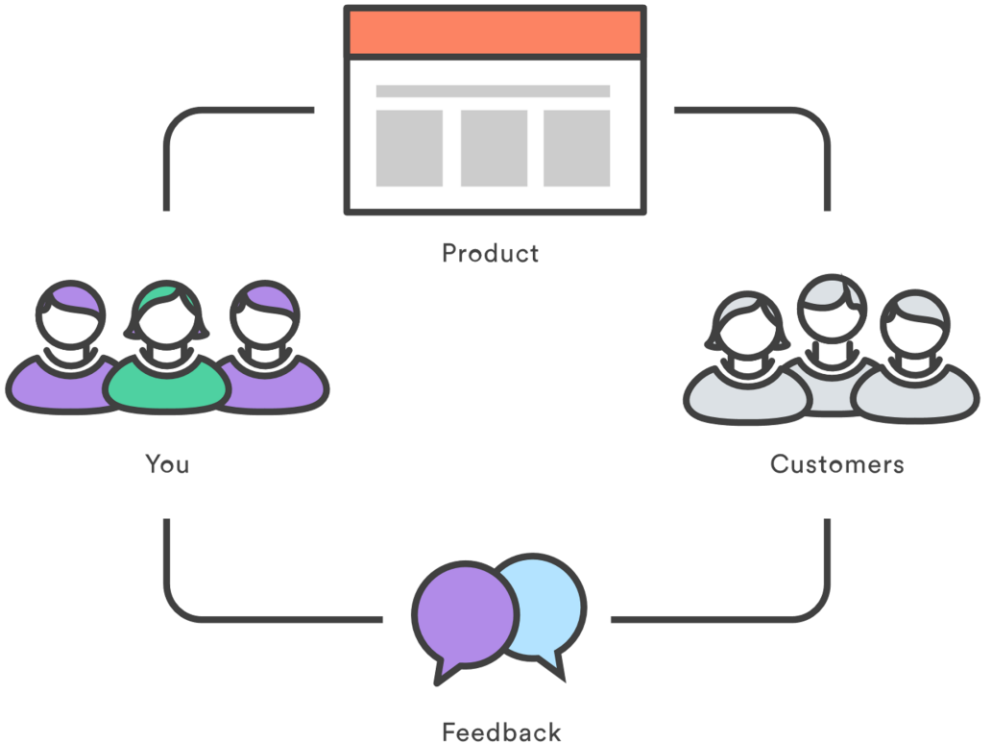
Важность инструментов управления исходным кодом



Если у Пети появится команда, то инструменты управления исходным кодом будут интересны всем участникам команды!

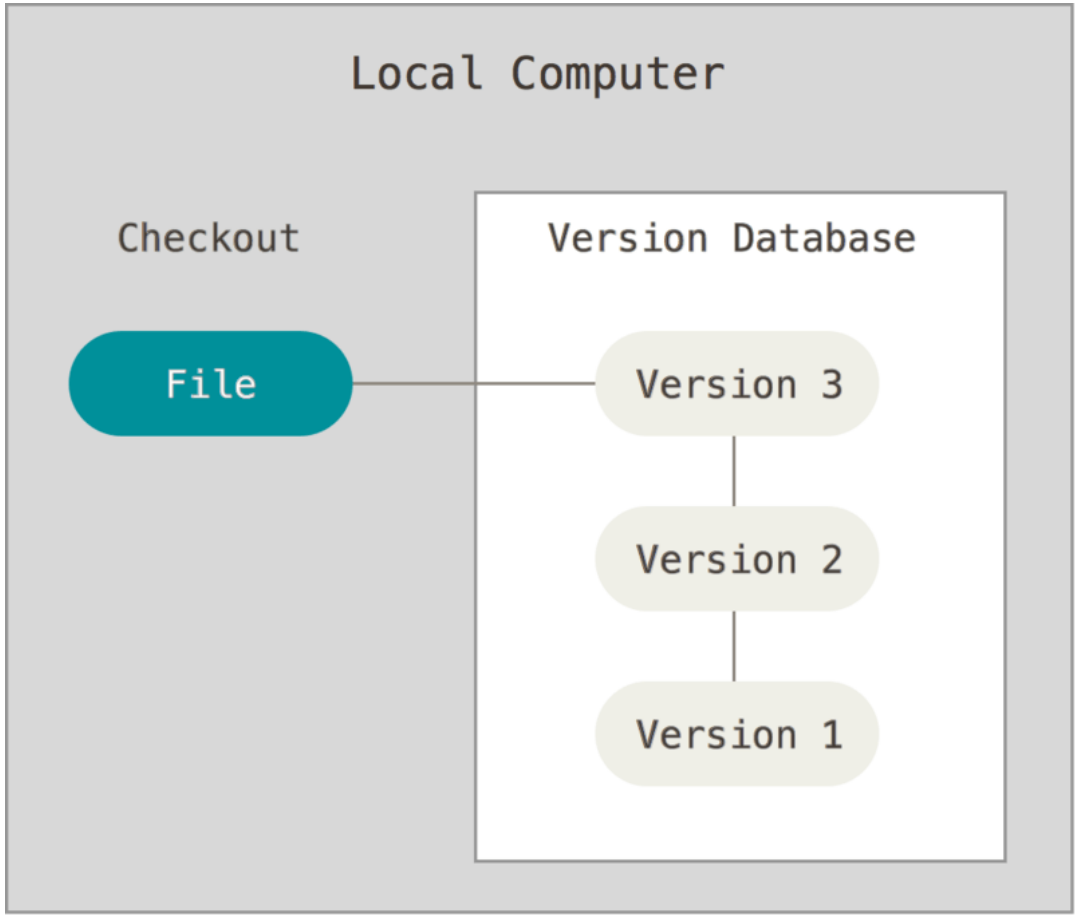
Преимущества управления исходным кодом

- ✓Учет примечаний к выпускной версии проекта.
- ✓Управление исходным сокращает коммуникационные издержки команды и увеличит скорость выпуска.



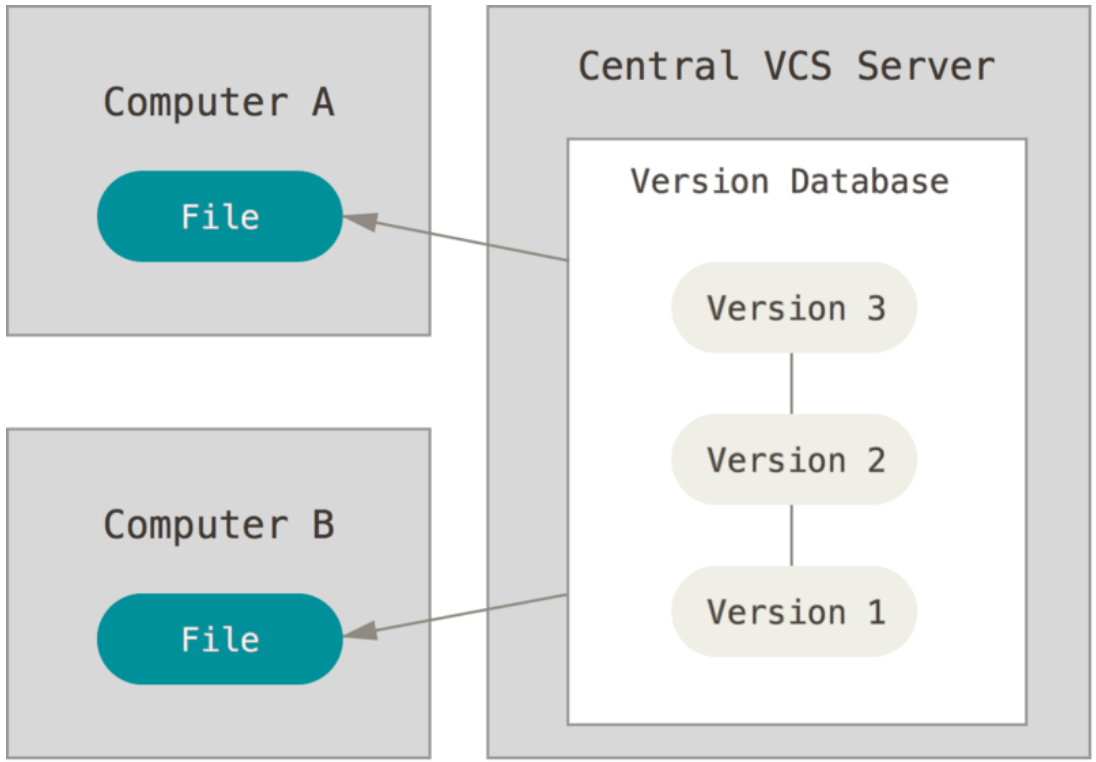
Виды систем контроля версий

Локальные системы контроля версий



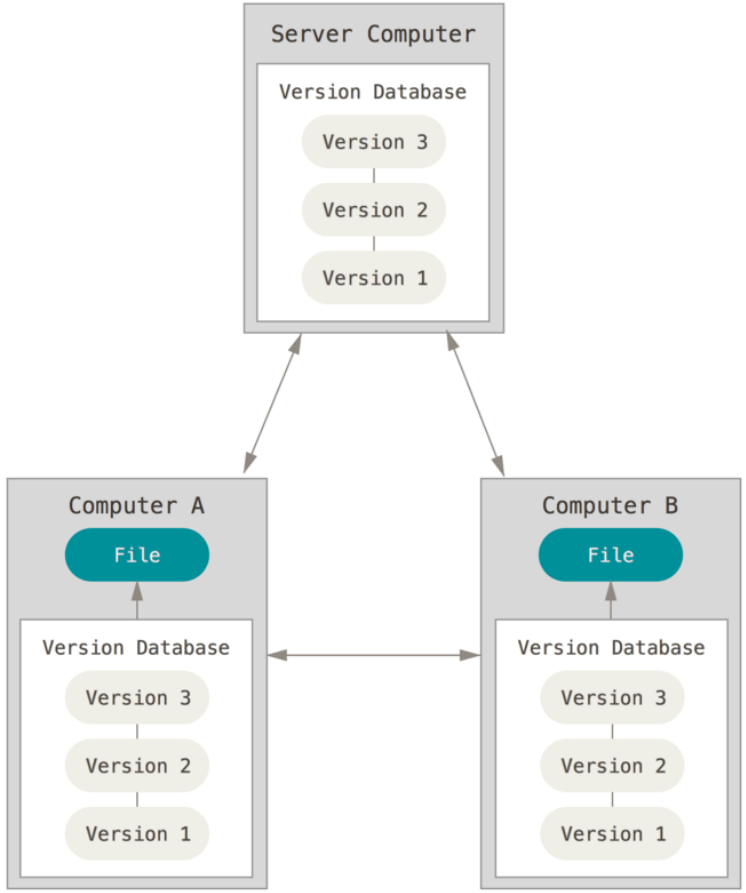
Виды систем контроля версий

Централизованные системы контроля версий



Виды систем контроля версий

Распределённые системы контроля версий



Git

Рассмотрим работу с системой контроля версий на примере самого популярного инструмента **Git**

Git – программка, которая работает в командной строке, но есть множество готовых интерфейсов для удобства)

git

Краткая 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.

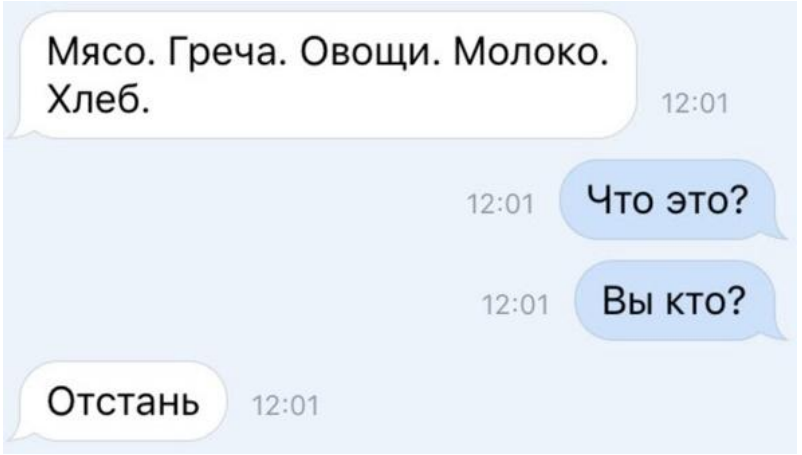
Так началась разработка **Git**...

Это Линус Торвальдс, "известный" фанат Nvidia

Основы работы с git

Рассмотрим основы работы с git на примере файлика со списком покупок. Когда мы добавляем туда новые позиции их надо купить, а когда уже купили, то удаляем из файла имеющиеся позиции.

Все просто, но если захочется узнать, что мы купили неделю назад, мы снова запутаемся. Вот тут то нам и пригодится система контроля версий! Приступим)



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 нудной нам папочке.

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 файлы журналов конфигураций и операций, выполняемых над репозиторием, а также сами контролируемые файлы.

Создадим репозиторий

Создадим новую папочку «Покупки», перейдем в нее с помощью команды

$ cd Покупки

И создадим наш первый репозиторий

$ git init .

Поздравляю! Теперь гит будет следить за всеми файлами в папочке «покупки»

Покупки

Создаем изменения

Создадим новый файлик 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

Добавляем 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 может быть закоммичен (to be committed). Отлично! …А что это значит?

Покупки

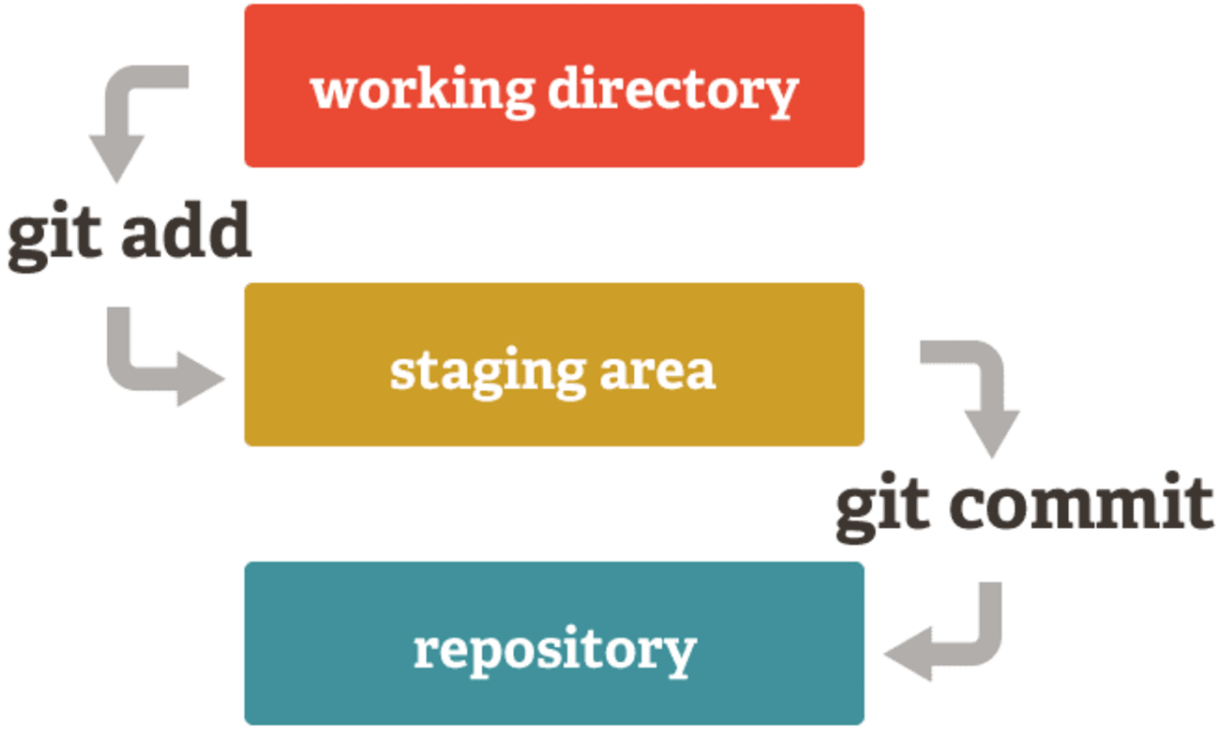
Продукты

+ Молоко
+ Яблоки
+ Пельмени

Состояния гита

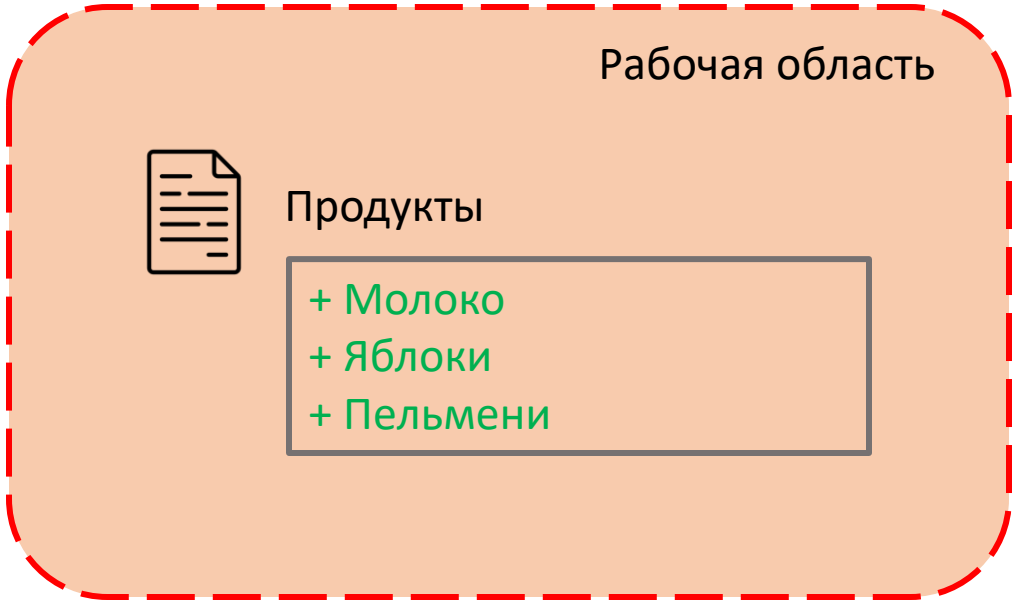
В процессе работы над файлами в репозитории для гита они могут находиться в трех состояниях:

- Рабочая область
- Индекс
- Каталог git



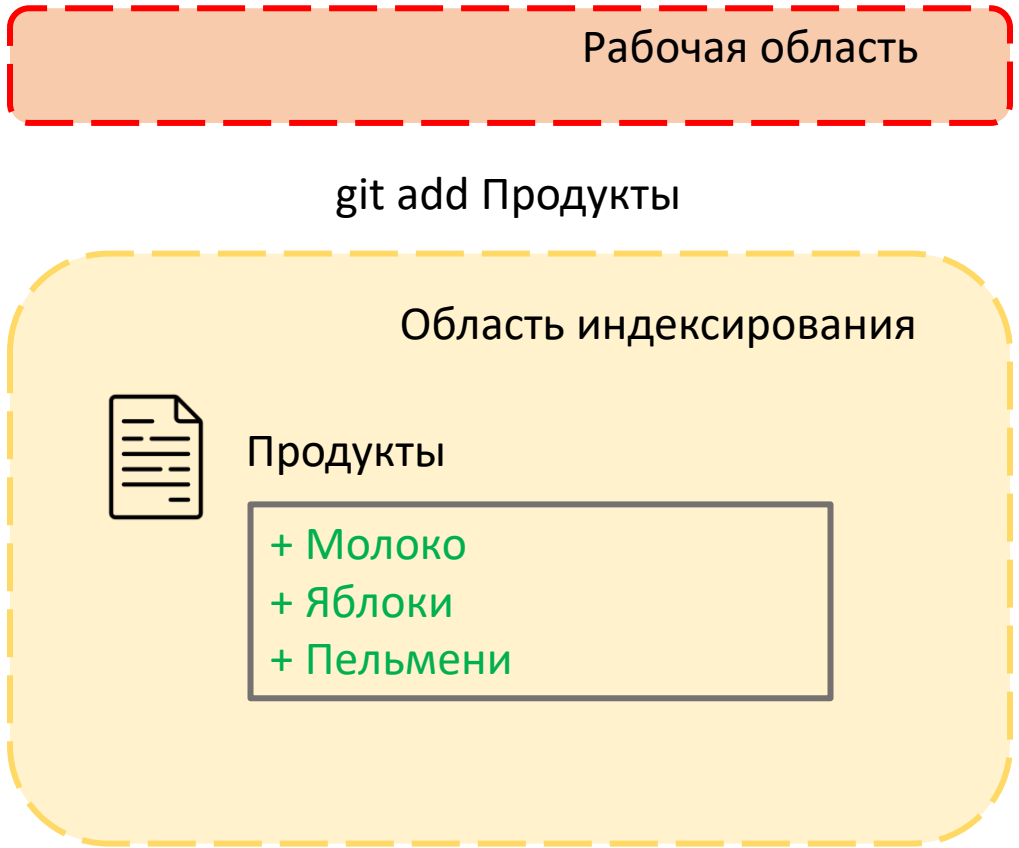
Working directory

Рабочая область, она же рабочая копия, она же working directory – просто пространство в репозитории и является снимком одной версии проекта. Эти файлы извлекаются из сжатой базы данных в каталоге Git и помещаются на диск, для того чтобы их можно было использовать или редактировать.



Область индексирования

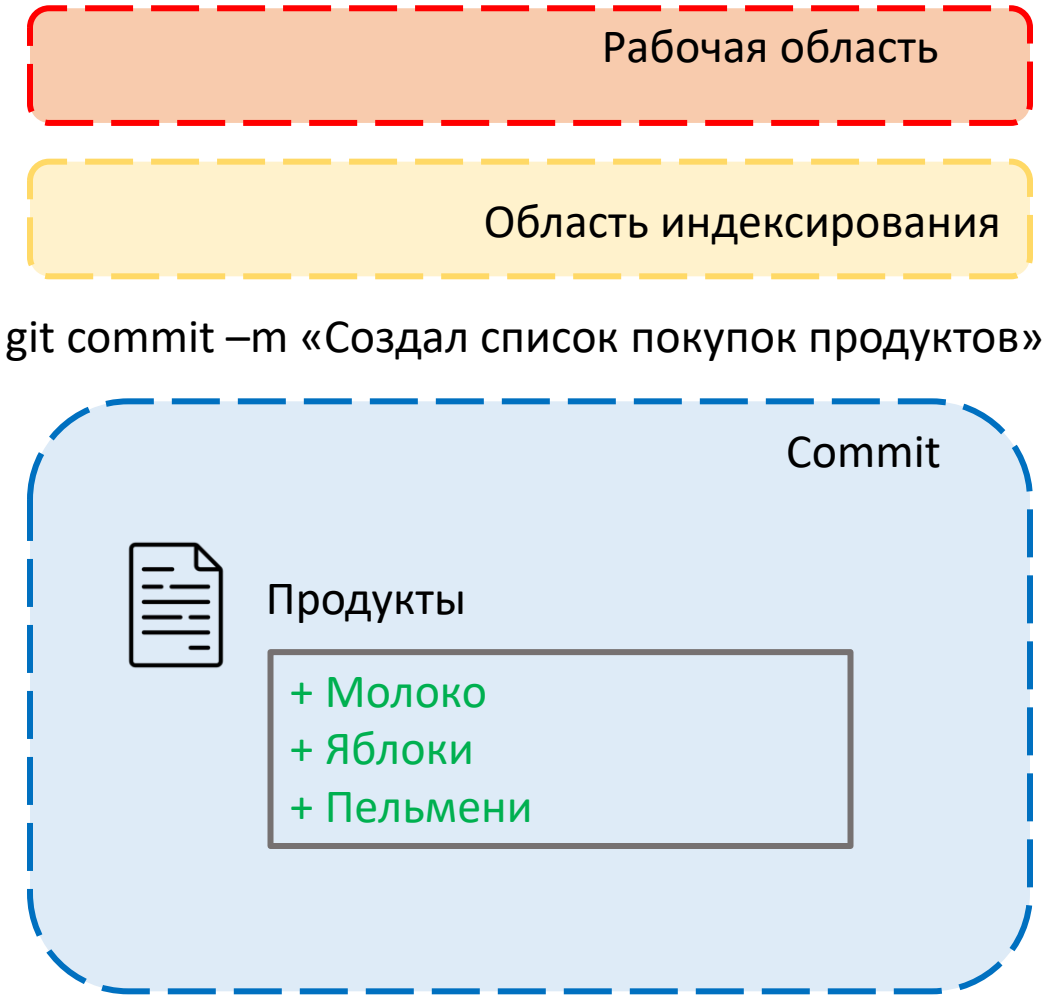
Область индексирования, он же индекс – заготовка для коммита, которую потом можно сохранить в истории. Просто выберете понравившиеся изменения и добавьте их в индекс с помощью команды git add



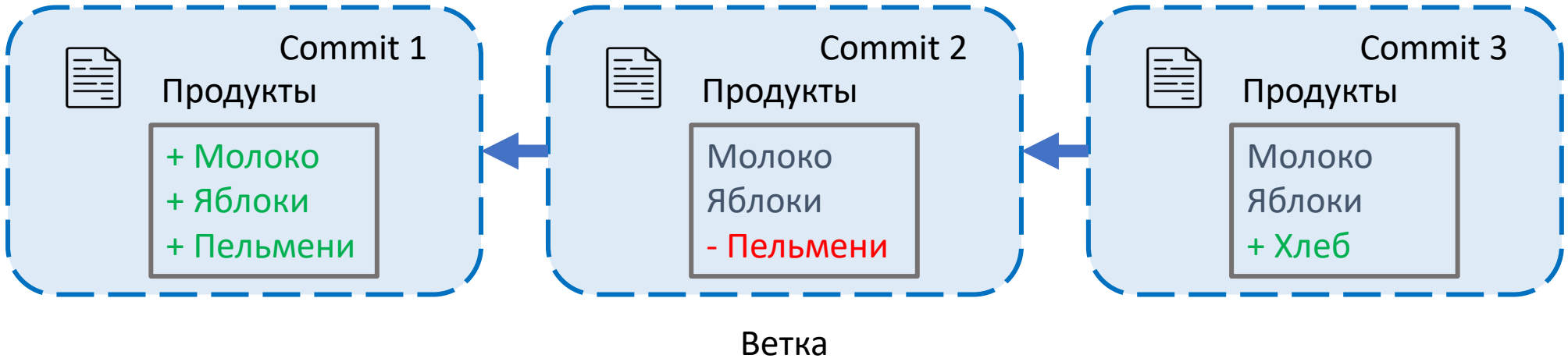
Каталог git

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»



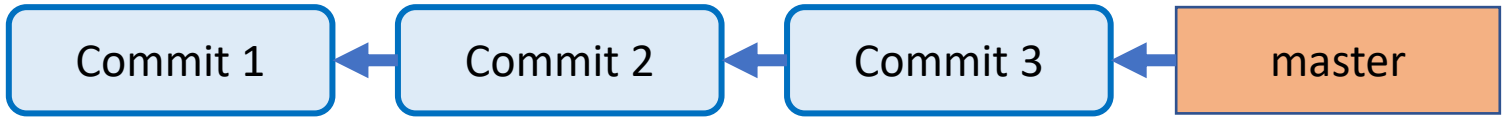
Ветка



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

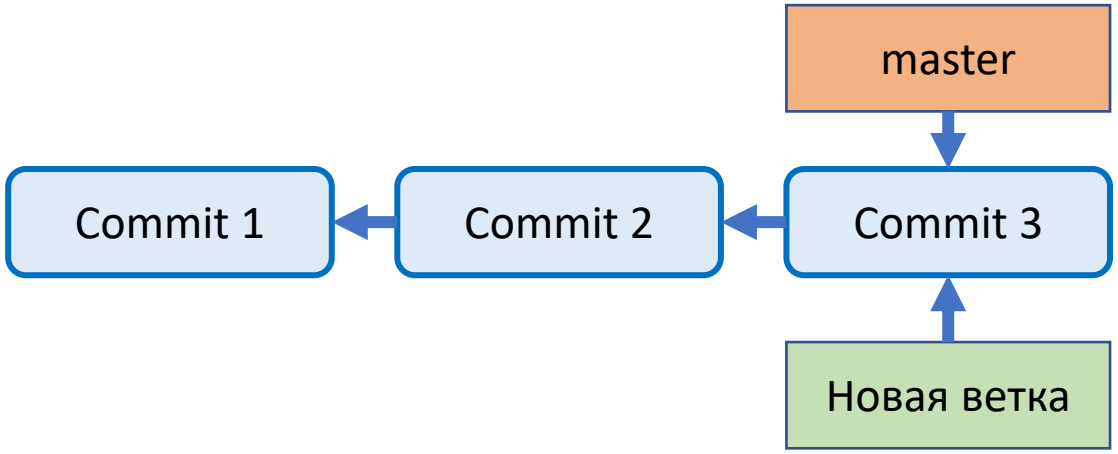
Основная ветка

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**.



Много веток

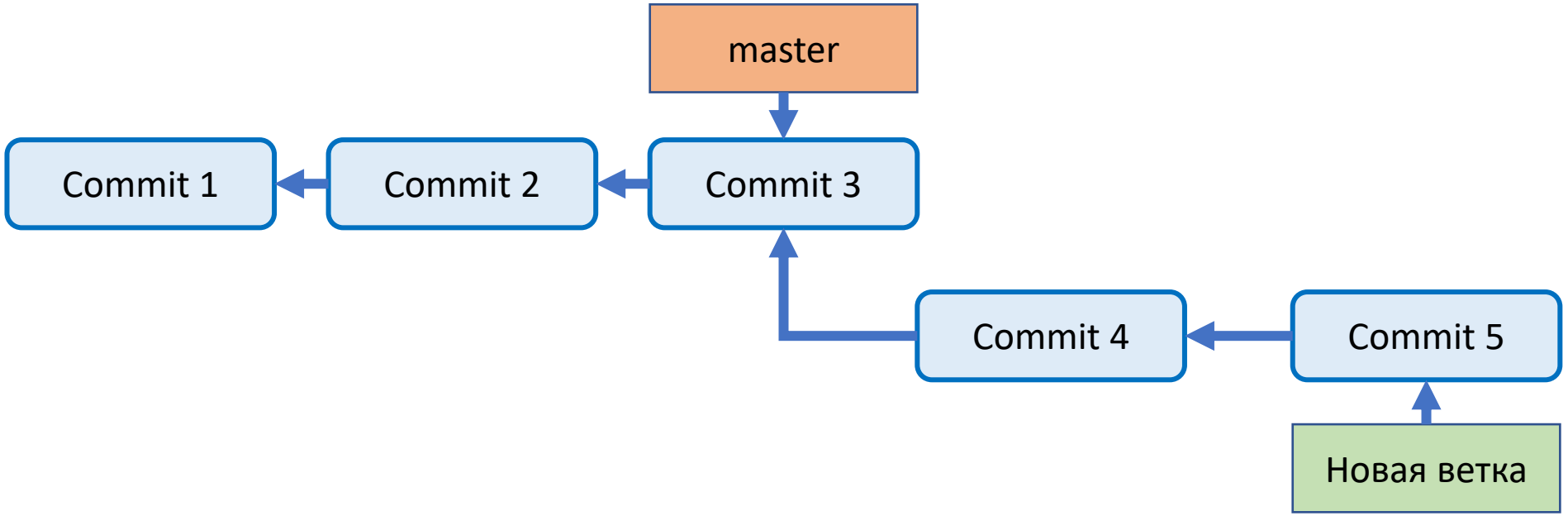
Когда работаешь с репозиторием в одиночку, можно обойтись и одной веткой, но система контроля версий создавалась для совместной организации кода многими людьми, поэтому для того, чтобы не мешать друг другу можно создать отдельные ветки (пути изменений) под различные нужды.



Создание веток

Ветку можно создать с помощью команды **git branch имя_ветки**

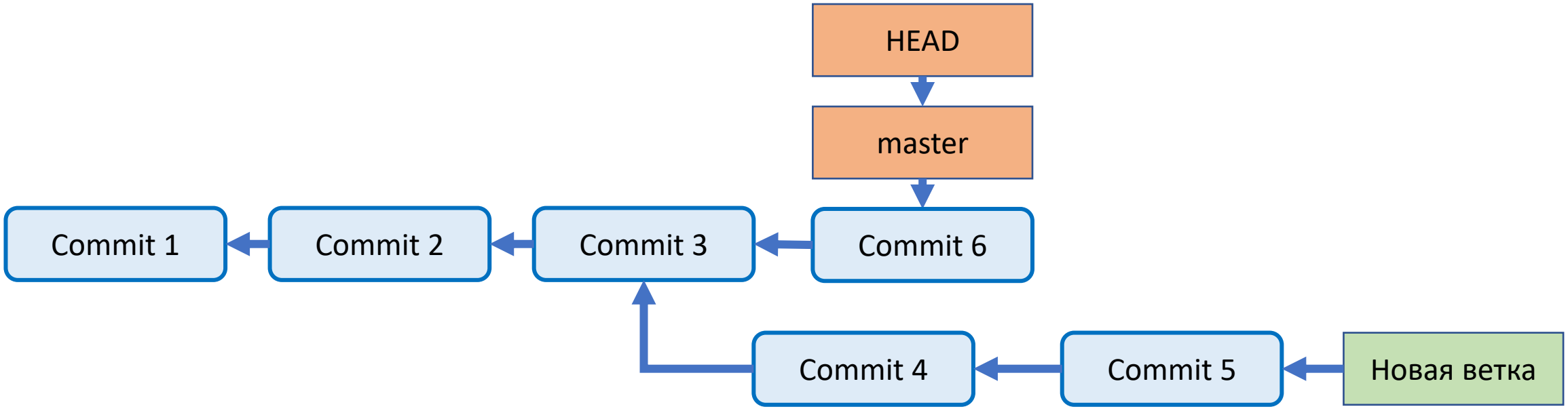
Сама ветка по сути – это указатель на последний коммит, к которому добавится новые правки.



Переключение веток

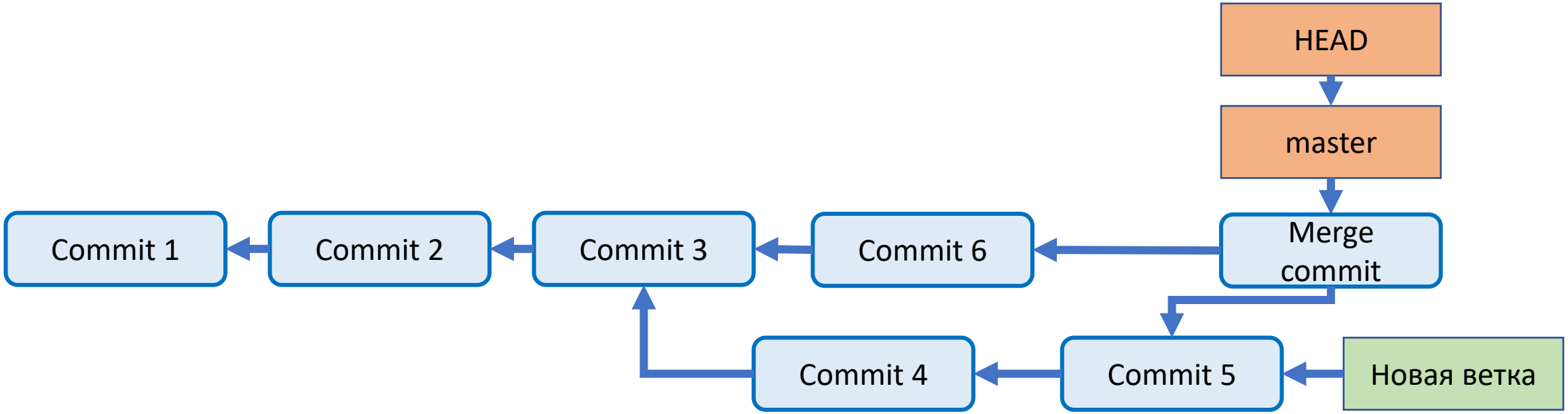
Существует еще один указатель HEAD, который указывает на коммит, чье состояние сейчас и развернуто в рабочей области. Те файлы, которые есть в папочке, образовались с помощью последовательности коммитов, на которую указывает HEAD.

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.



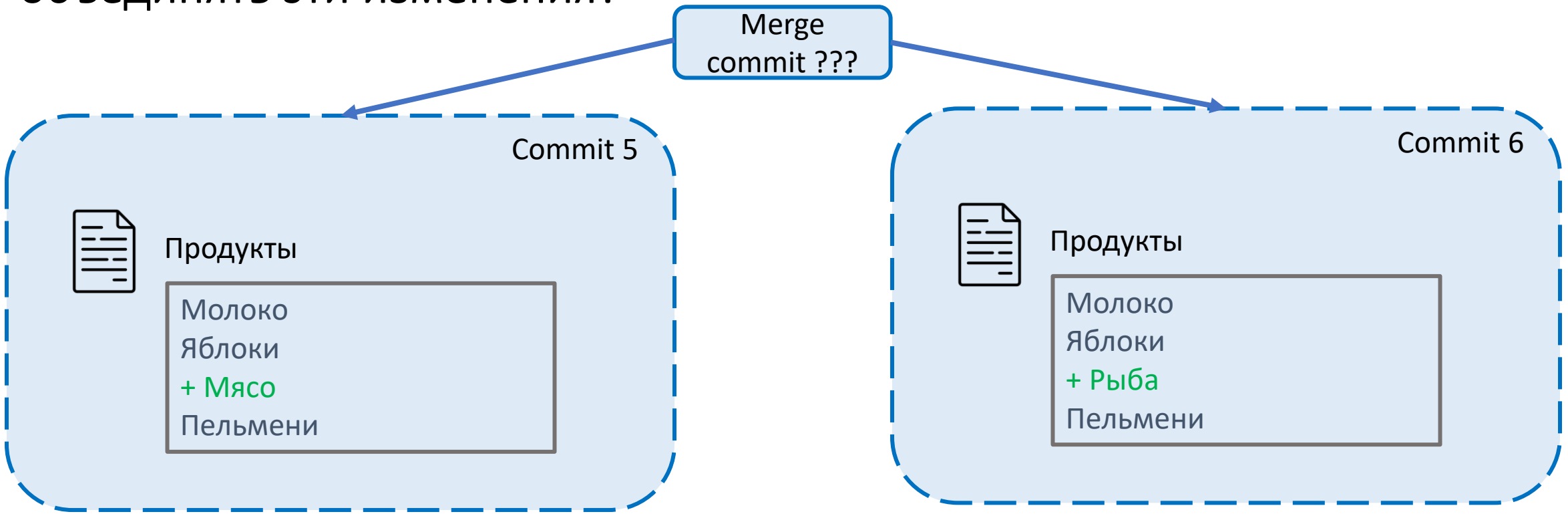
Слияние веток

Когда работа над отдельной задачей завершается, то необходимо внести изменения в основную ветку. Эта процедура называется слияние и выполняется командой **git merge**. В результате появляется новый коммит, который ссылается на два предыдущих коммита одновременно.



Конфликт слияния

В результате слияния изменения всех путей сливаются в один путь. Но что делать, если мы общие файлы в разных ветках, как объединять эти изменения?



Конфликт слияния

Полученная ситуация называется конфликтом и git не может сам решить за пользователя, что ему важнее, поэтому в дело вступает автор, который уже вручную выбирает все нужные изменения.

Удаленный репозиторий

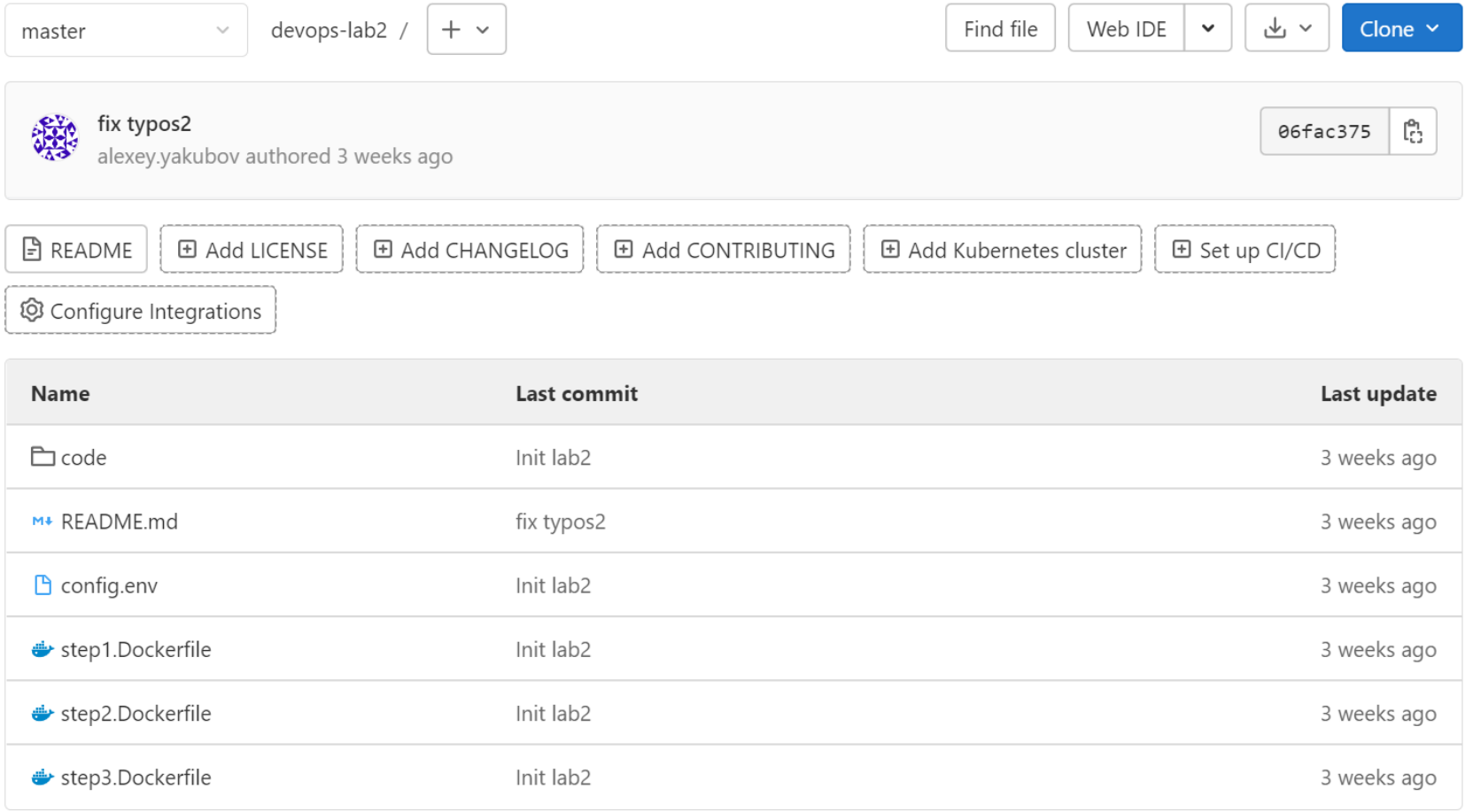
До текущего момента мы работали на локальной машине, в нашей папочке. Но как же делиться кодом с коллегами, как совместно решать множество задач?

Для этого существуют удаленные репозитории кода, которые живут в интернете.

Из самых известных это github, gitlab и bitbucket.

Push изменений

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



Pull изменений

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

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

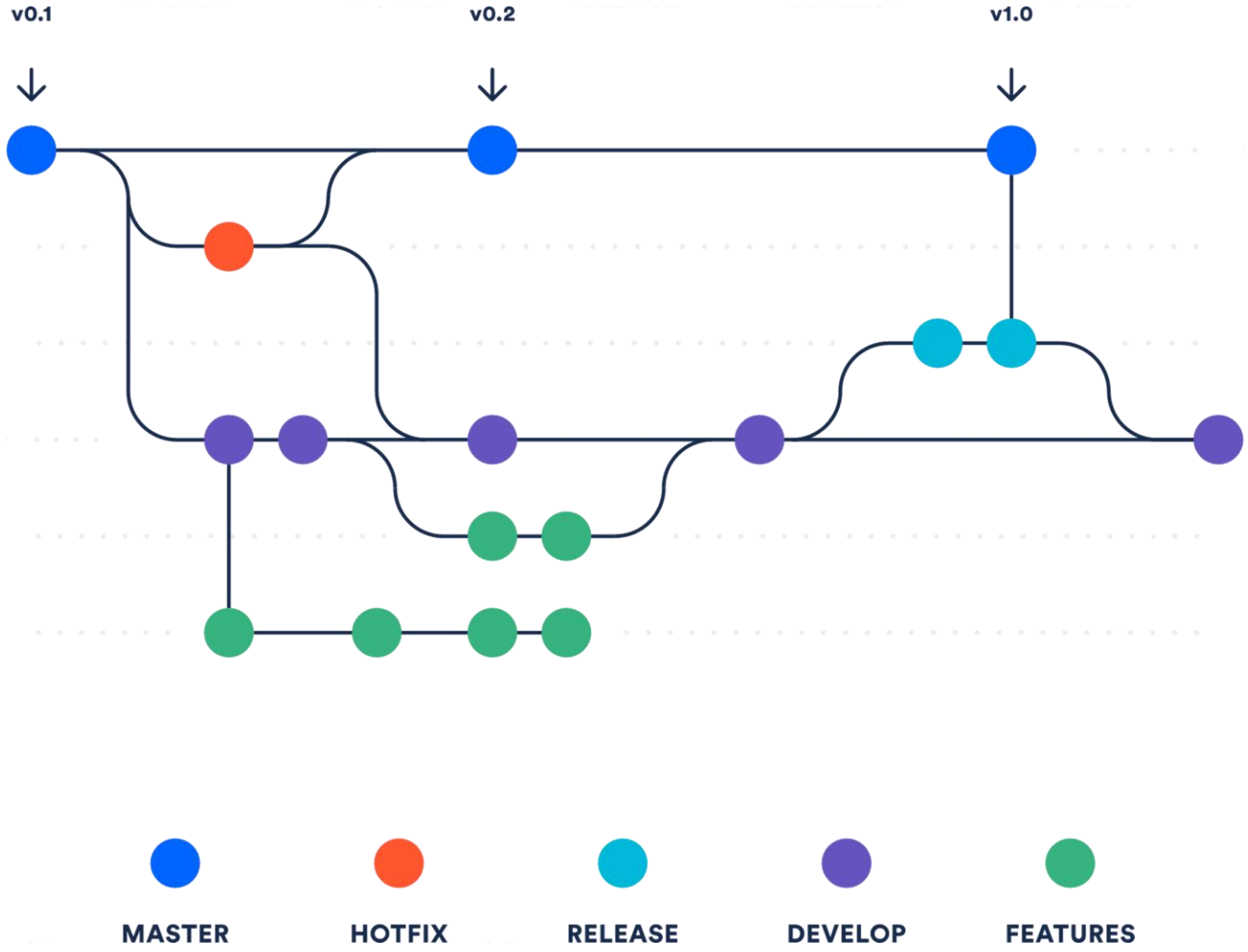
Search (e.g. *.vue) (Ctrl+P)

README.md +2 -1

README.md +2 -1 Viewed

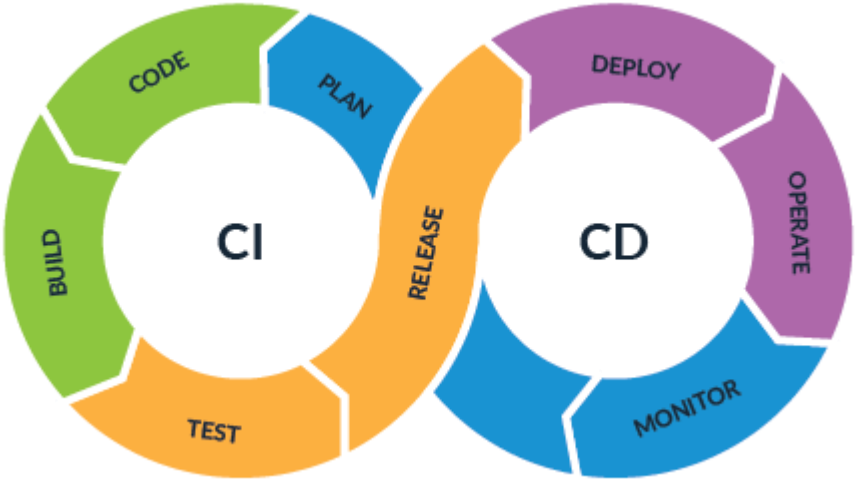
```
@@ -132,4 +132,5 @@ docker run --link mysql -p 8000:8000 --env-file=config.env step3
132   1. Перезапустите сборку собранного образа, оцените время пересборки, объясните причины
133   2. К какому число слоев стремиться в образе, правила оптимизации
134   3. Опишите базовые команды Dockerfile, что они делают, где смотреть документацию?
135 - 4. Что такое контекст сборки, как его оптимизировать?
    \ No newline at end of file
135 + 4. Что такое контекст сборки, как его оптимизировать?
136 + 5. Как работает Merge Request?
```

Иллюстрация контроля версий

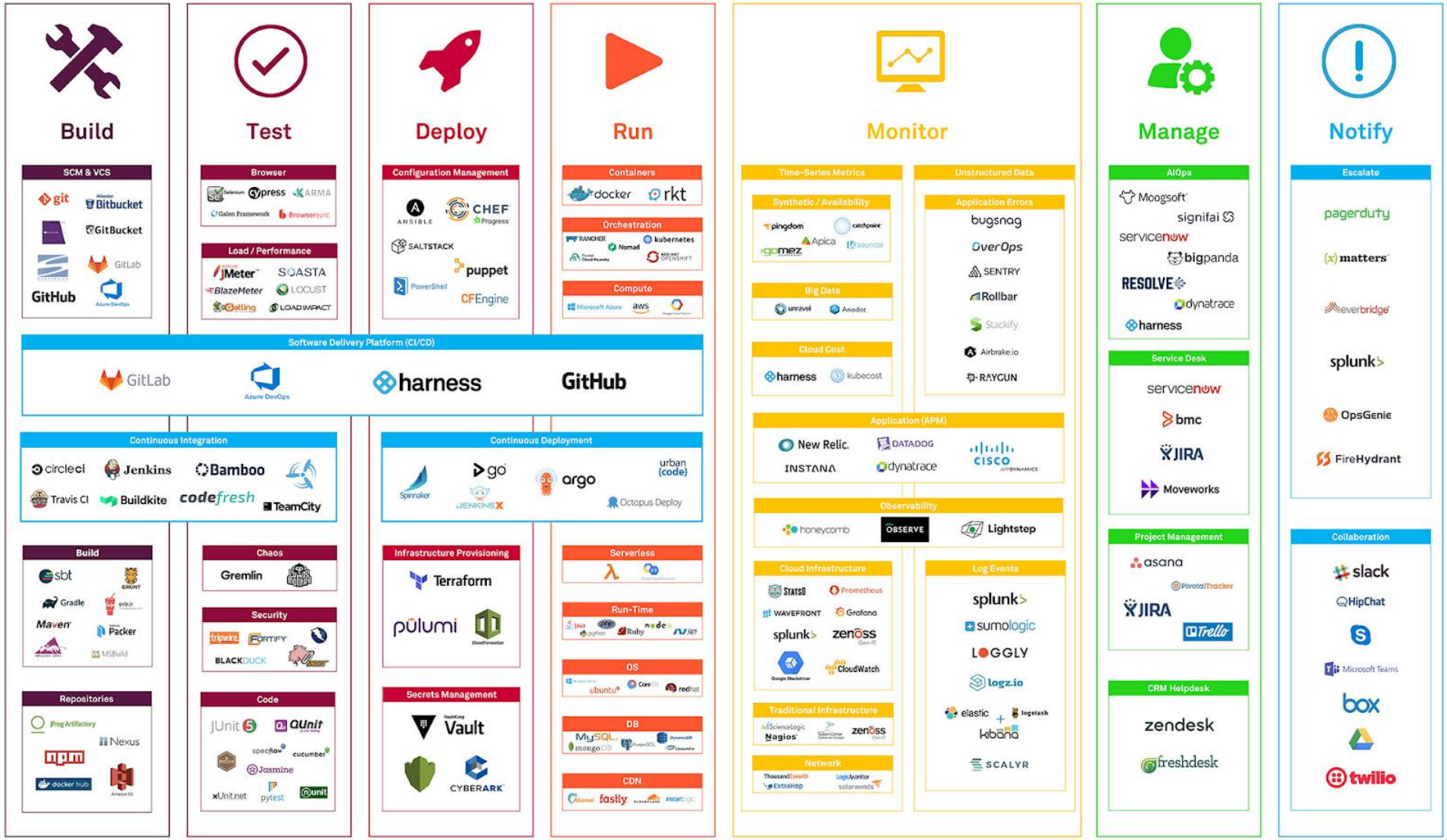


CI/CD

Удаленный репозиторий кода позволяет настривать автоматические действия, которые выполняются при появлении нового кода. Это позволяет автоматически проверять проверять новый код и даже автоматически доставлять новый код пользователям



DevOps Tools Ecosystem 2021



harness

The Platform for Software Delivery

www.harness.io

Полезные ссылки

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