Системы контроля версий

Это что за покемон?

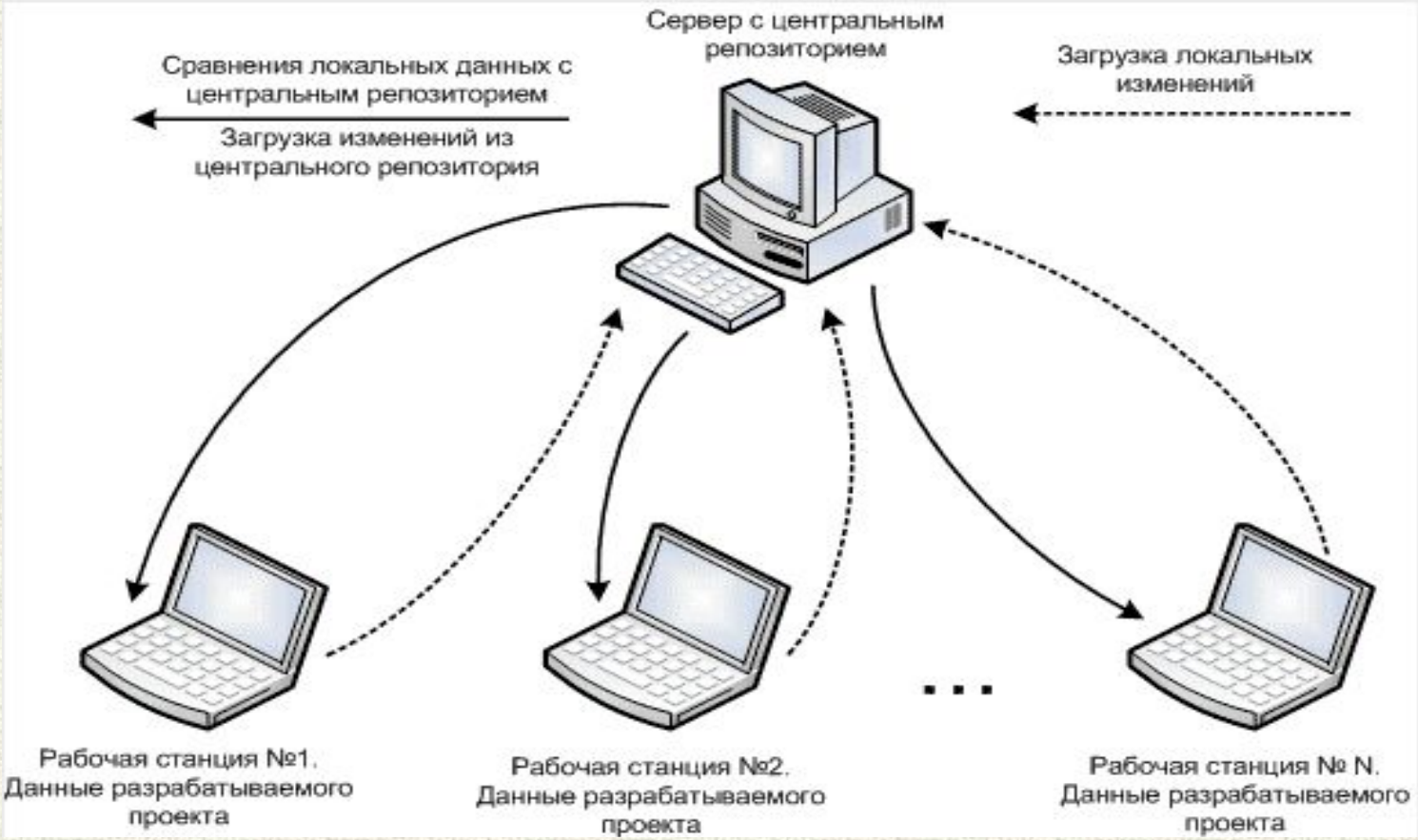
Система контроля версий – программное обеспечение, регистрирующая изменения в одном или нескольких файлах, и позволяющая вернуться к определенным версиям этих файлов.

Типы СКВ

1. Локальные
   - Папка с копиями файлов
2. Централизованные
   - CVS;
   - Subversion;
   - Perforce.
3. Распределённые
   - Git
   - Mercurial
   - Bazaar
   - Darcs

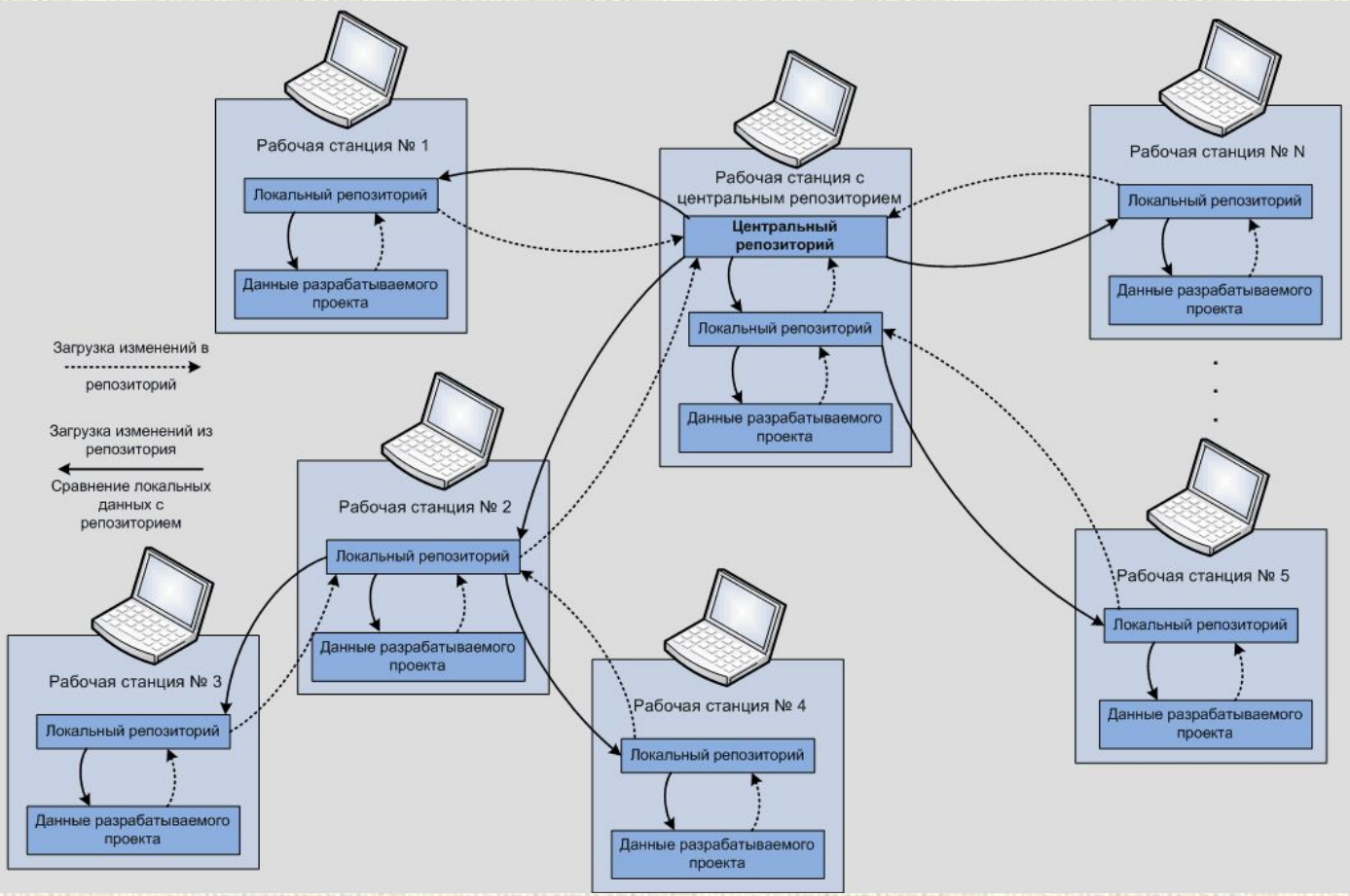
ЦСКВ

ЦСКВ – все файлы под версионным контролем хранятся на сервере. А клиенты получают копии файлов с сервера.



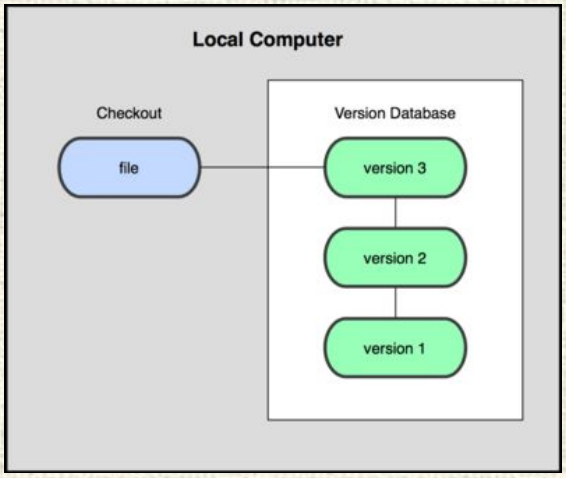
РСКВ

РСКВ – рабочие копии проектов хранятся на компьютерах разработчиков. При этом есть центральный узел, с которым синхронизируются локальные версии проектов.

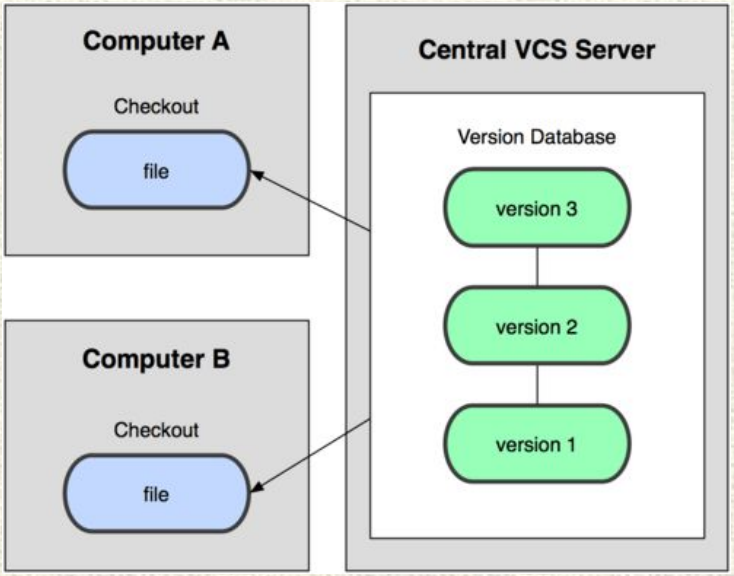


Типы СКВ

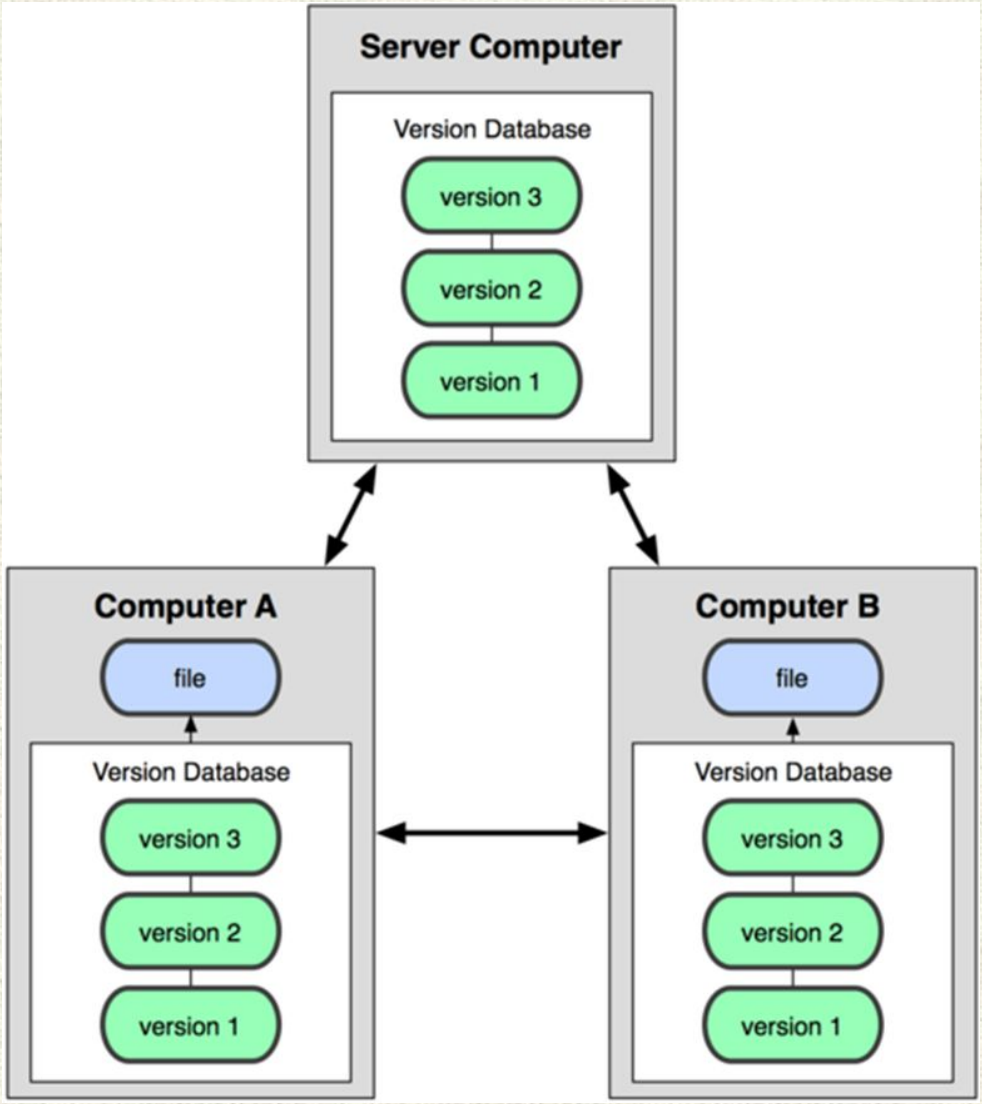
Локальная



Централизованная



Распределенная



А пользоваться-то чем?

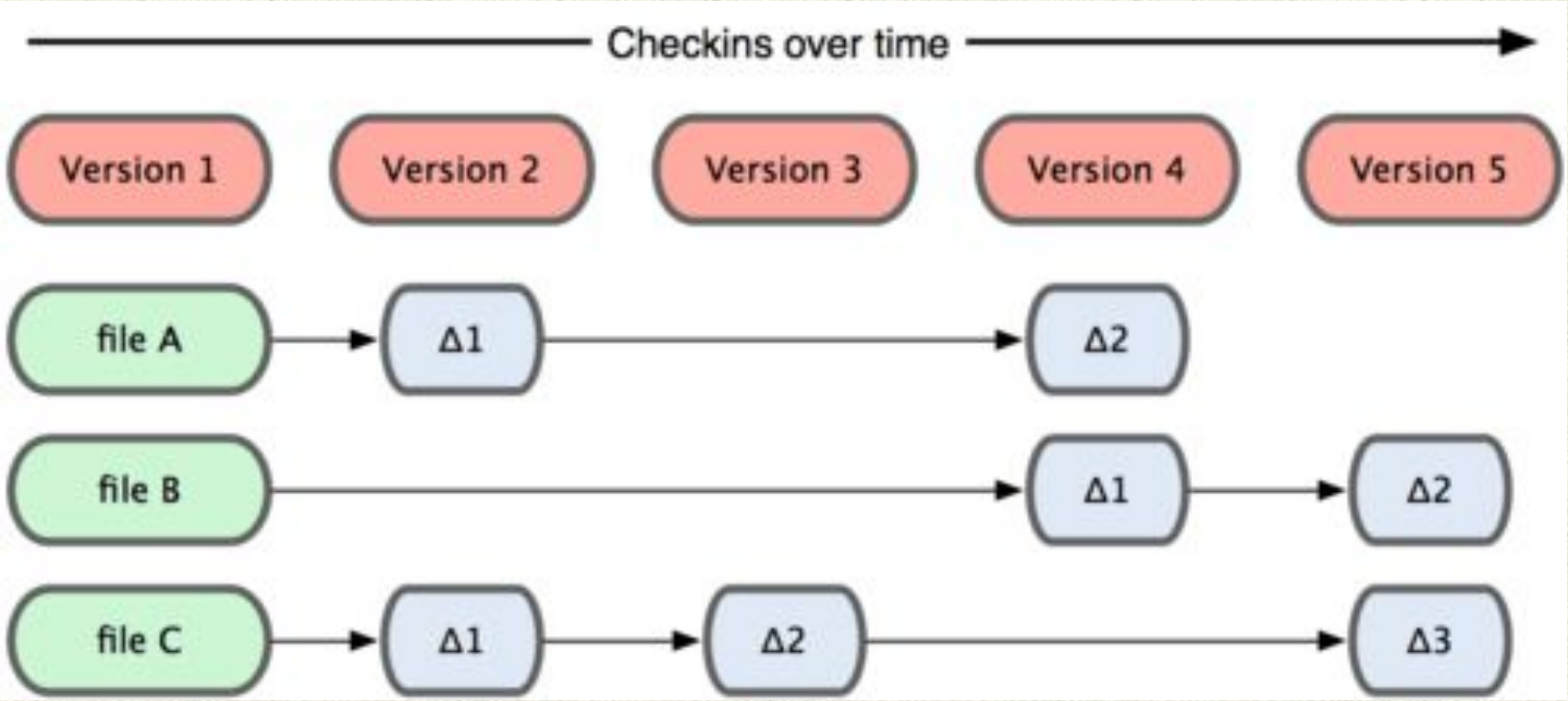
Git — распределённая система контроля версий, которая даёт возможность разработчикам отслеживать изменения в файлах и работать совместно с другими разработчиками.

Скажите, дети, как его зовут?

Джунио Хамано

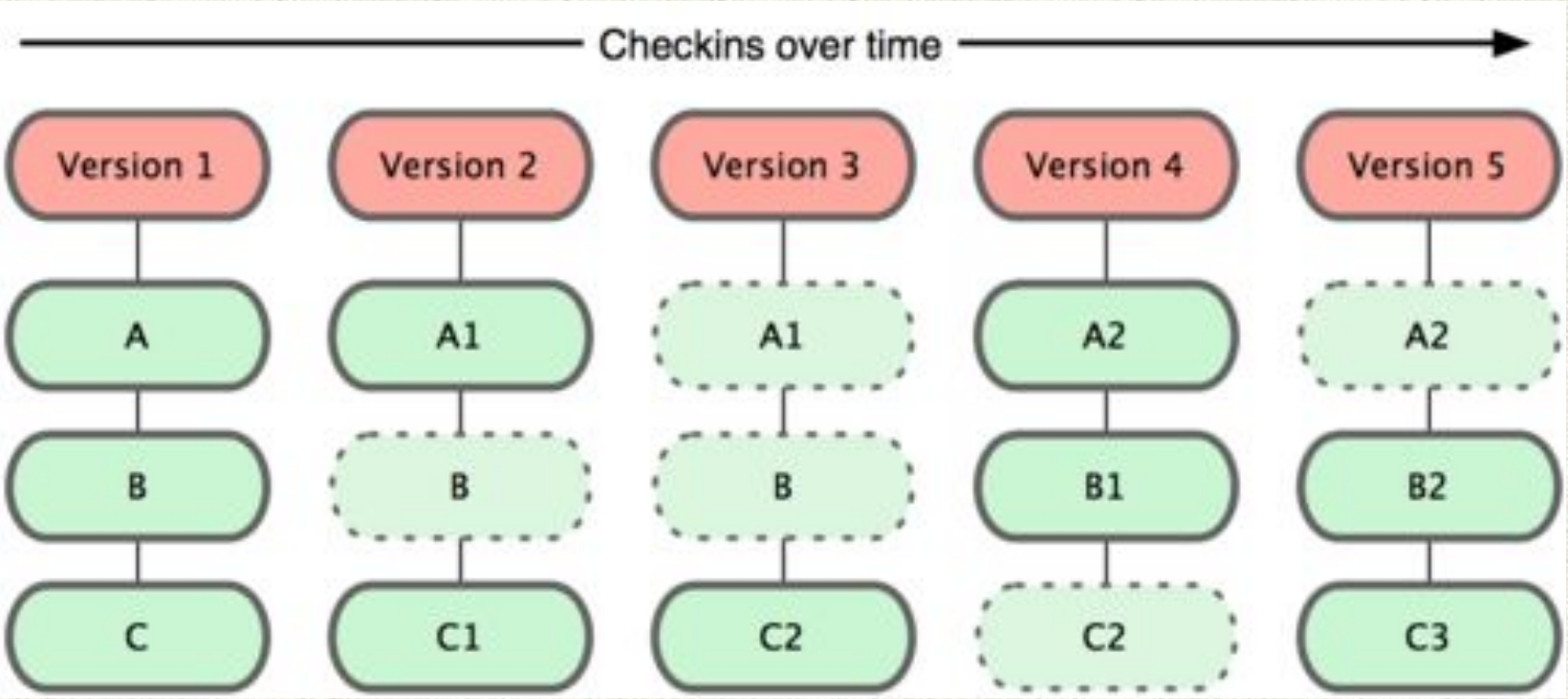
Как работает?

Другие СКВ: хранят информацию как список изменений (патчей) для файлов. Они относятся к хранимым данным, как к набору файлов и изменений, сделанных для каждого из этих файлов во времени.



Как работает?

Git считает хранимые данные набором слепков небольшой файловой системы. Каждый раз, когда вы фиксируете текущую версию проекта, Git сохраняет слепок того, как выглядят все файлы проекта на текущий момент. Ради эффективности, если файл не менялся, Git не сохраняет файл снова, а делает ссылку на ранее сохранённый файл.



Особенности работы

- Большинство операций – локальные
- Большое внимание целостности данных (контрольные суммы)
- Чаще всего данные только добавляются.

Состояние файлов в GIT

- Изменённое (файл претерпел изменения, но эти изменения ещё не были зафиксированы)
- Подготовленное (файл изменён и отмечен для включения в следующую версию – «слепок» всего проекта, к которому можно вернуться при необходимости)
- Зафиксированное (файл сохранён в локальной базе)

Основные термины

- Репозиторий - хранилище версий файлов.
- Коммит (Commit) — фиксация изменений или запись изменений в репозиторий (некая точка фиксации, пункт). Коммит происходит на локальной машине.
- Рабочая копия - текущее состояние файла.
- Ветка (Branch) — это параллельная версия репозитория.
- Мастер (Master) — главная или основная ветка репозитория. Когда вы делаете свой первый коммит, у вас создается ветка мастер.
- Пуш (Push) — отправка всех неотправленных коммитов на удалённый сервер репозитория.
- Пул (Pull) — получение последних изменений с удалённого сервера репозитория.

Схема работы

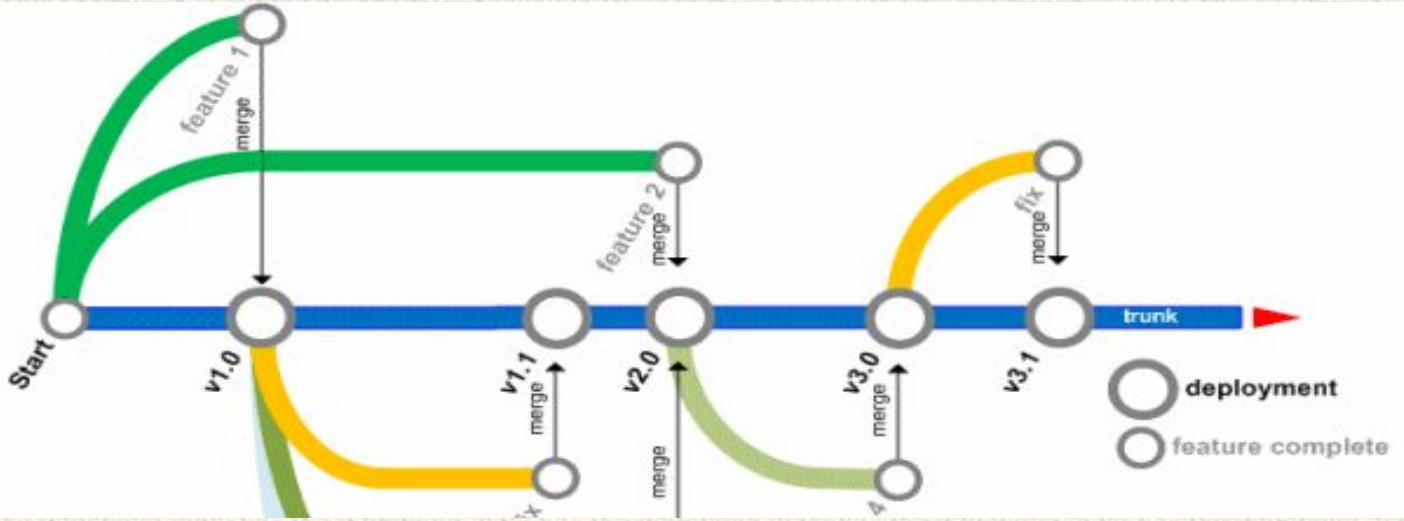
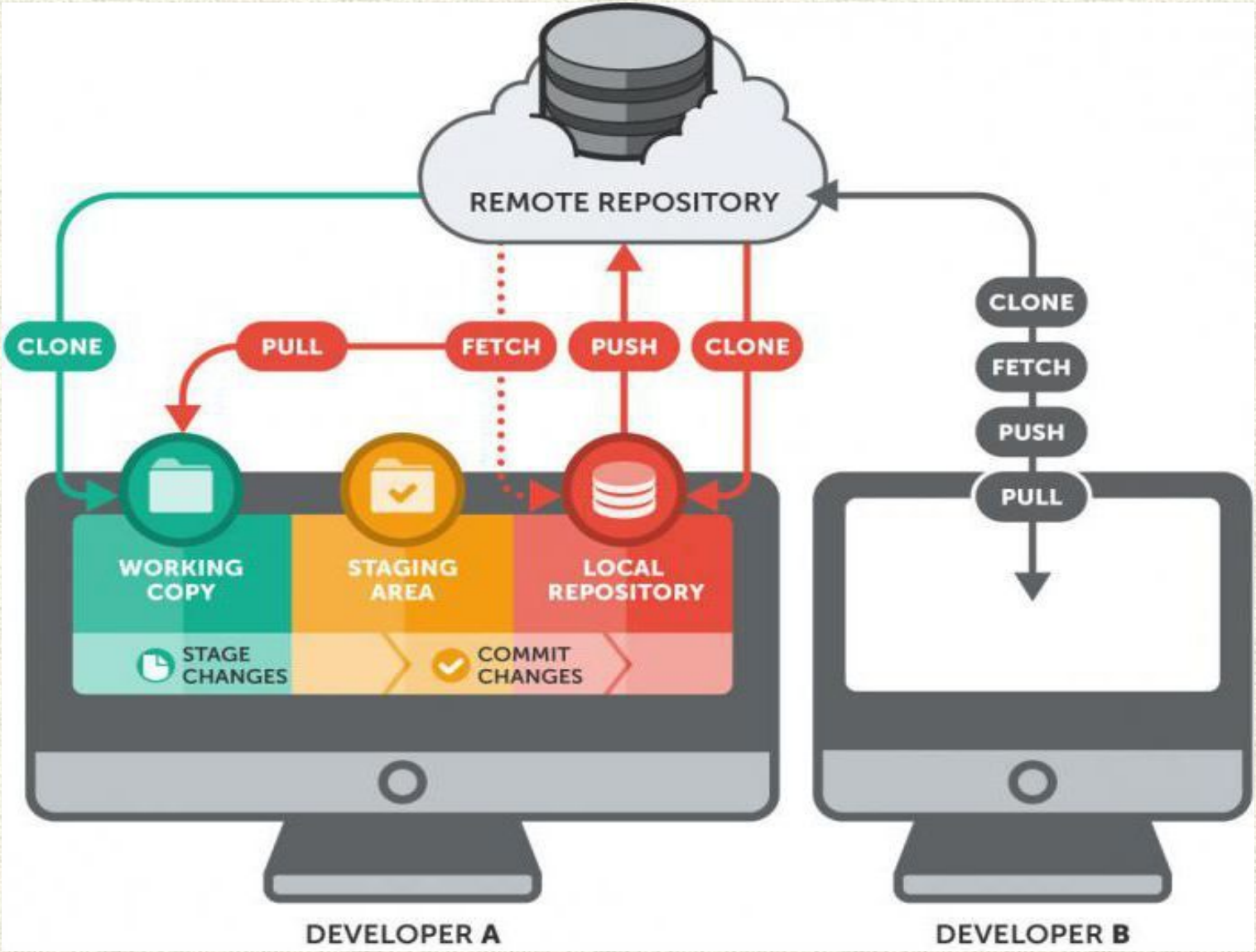


Схема работы



А откуда там сервер?

GitHub — сервис онлайн-хостинга репозиториев. Это просто веб-портал, который предлагает хостинг для исходного кода на базе системы контроля версий GIT. GitHub это не GIT!!

Алгоритм работы с гитом

Стандартный рабочий процесс с использованием Git'a выглядит примерно так:

1. Вы вносите изменения в файлы в своём рабочем каталоге.
2. Подготавливаете файлы, добавляя их слепки в область подготовленных файлов.
3. Делаете коммит, который берёт подготовленные файлы из индекса и помещает их в каталог Git'a на постоянное хранение.
4. * Устанавливаете связь с глобальным репозиторием.
5. * Отправляете данные в облачное хранилище.

И ещё термины

- Локальный репозиторий — репозиторий, расположенный на локальном компьютере разработчика в каталоге. Именно в нём происходит разработка и фиксация изменений, которые отправляются на удалённый репозиторий.
- Удалённый репозиторий — репозиторий, находящийся на удалённом сервере. Это общий репозиторий, в который приходят все изменения и из которого забираются все обновления.
- Клонирование (Clone) — скачивание репозитория с удалённого сервера на локальный компьютер в определённый каталог для дальнейшей работы с этим каталогом как с репозиторием.
- Мёрдж (Merge) — слияние изменений из какой-либо ветки репозитория с любой веткой этого же репозитория. Чаще всего слияние изменений из ветки репозитория с основной веткой репозитория.

И ещё термины

- Форк (Fork) — копия репозитория. Его также можно рассматривать как внешнюю ветку для текущего репозитория. Копия вашего открытого репозитория на Гитхабе может быть сделана любым пользователем, после чего он может прислать изменения в ваш репозиторий через пулреквест.
- Пулреквест (Pull Request) — запрос на слияние форка репозитория с основным репозиторием. Пулреквест может быть принят или отклонён вами, как владельцем репозитория.
- Обновиться из апстрима — обновить свою локальную версию форка до последней версии основного репозитория, от которого сделан форк.
- Обновиться из ориджина — обновить свою локальную версию репозитория до последней удалённой версии этого репозитория.

Команды Git

Предподготовка.

- Создаете папку проекта
- Заходите в нее через консоль

Работа с гитом выполняется преимущественно через консоль.

1) git init - создать репозиторий в директории
2) git config [параметр] [значение] – настройка файла конфигурации

git config --global user.name “User Name”

git config --global user.email “usermail@example.com”

Ключ --global говорит, что эти настройки - для всех репозиториев, где настройки не указаны явно.

Для локальных изменений нужно через консоль зайти в нужную папку и написать тоже самое, но без –global:

git config user.name “User Name”

git config user.email “usermail@example.com”

3) git config --list - просмотреть файл конфигурации

Команды Git

Предподготовка.

- Внесите изменения в файлы проекта.

4) git status - посмотреть статусы файлов
5) git add [имя_файла] - добавить к коммиту конкретный файл
6) git add . - добавить все файлы к коммиту
7) **git commit -m “commit message” - создание коммита**
8) git log - просмотреть историю коммитов
9) git diff - просмотреть изменения в файлах
10) git diff [имя_файла] - просмотреть изменения в конкретном файле
11) git rm [имя_файла] - убрать файл из списка фиксируемых в коммите

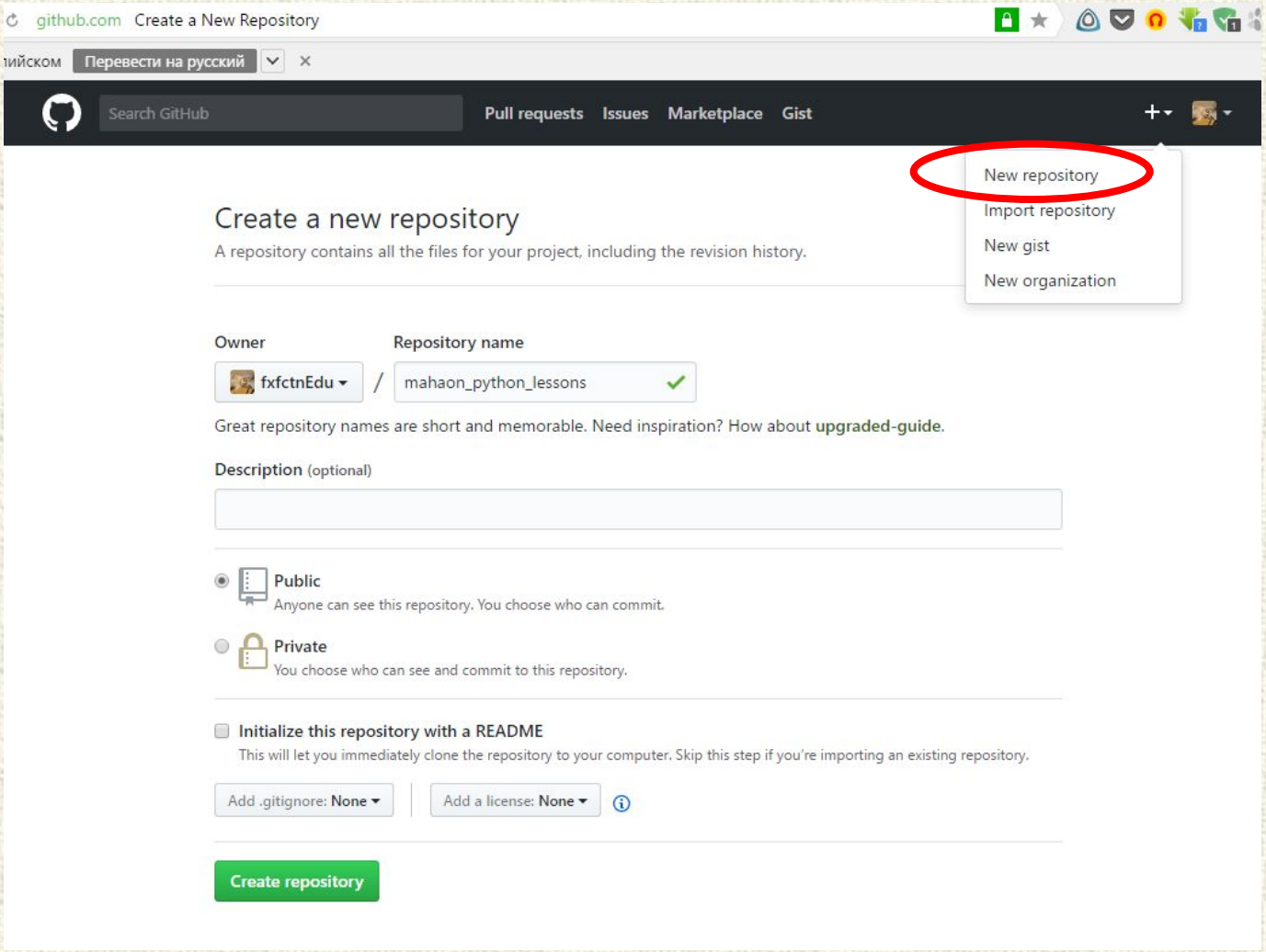
Правила хорошего коммита

- Сообщения должны фиксировать по возможности атомарные изменения.
- Коммиты лучше писать на английском языке.
- Коммиты правильно писать с Заглавной буквы, используя повелительное наклонение (если пишите по-английски), В КОНЦЕ ЗАГОЛОВКА ТОЧКУ НЕ СТАВЬТЕ.
- Длину предложения коммита желательно ограничивать 72 символами, а заголовок 50-ю.
- В сообщении пишите что и почему сделано, а не как
- Отделяйте тело коммита от заголовка пустой строкой:

Домашнее чтение: «Как писать сообщения коммитов?»

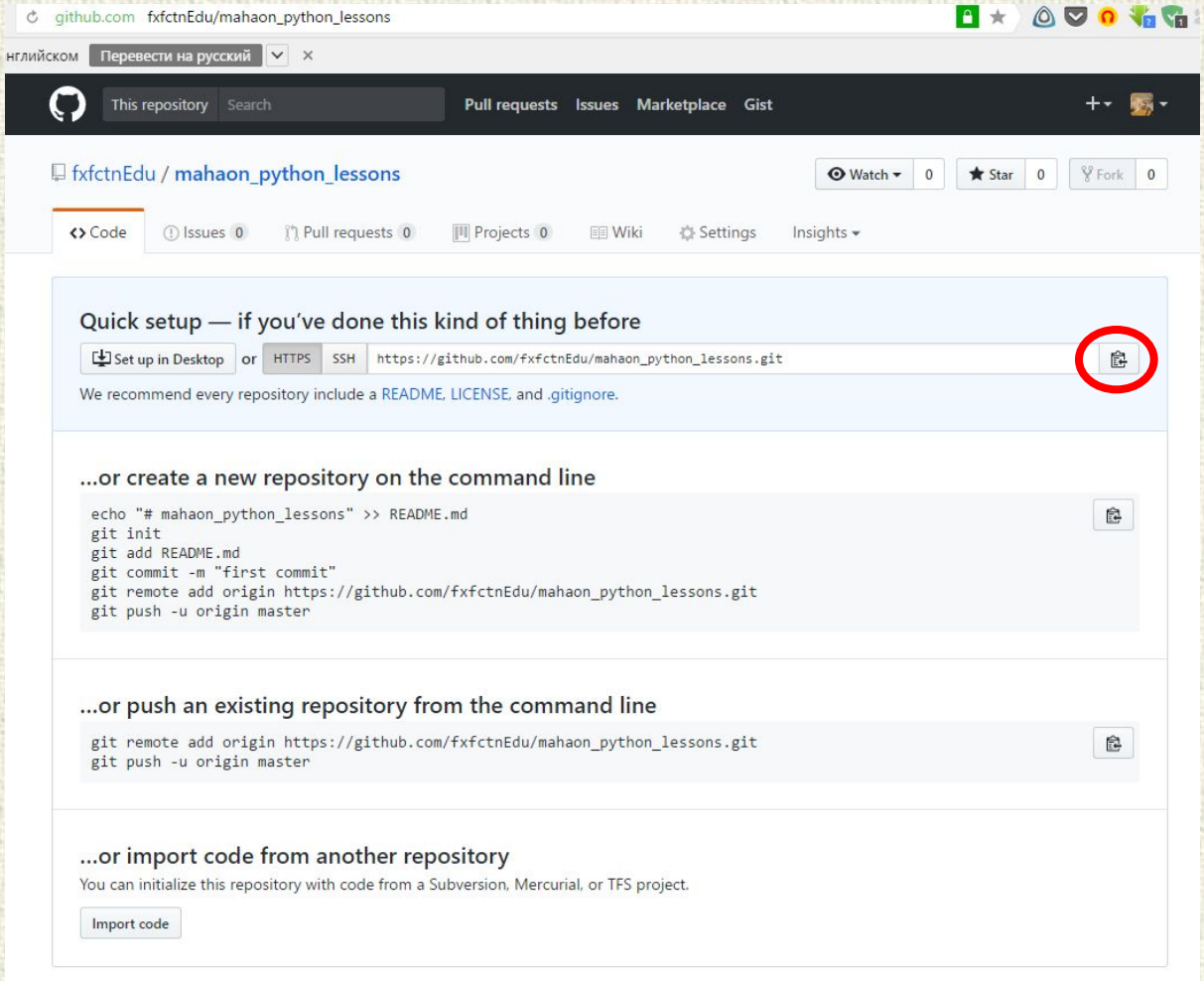
Удаленный репозиторий

Зарегистрируйтесь на Github. После успешной регистрации можно создать удаленный репозиторий. Для этого справа вверху щелкните на + и выберете в выпадающем меню New repository.



Удаленный репозиторий

В поле Repository name вы вводите имя репозитория. Можно добавить Readme.md файл, он содержит описание проекта, оформленное специальной разметкой, которую воспринимает GitHub и отображает на странице репозитория.



Команды Git

Предподготовка.

- Скопируйте ссылку на репозиторий с Github.

12) git remote add [alias] [link] - добавить ссылку на удаленный репозиторий
13) git remote -v - просмотреть список всех удаленных репозиториев

origin - это алиас (имя) вашего удаленного репозитория в настройках локального. Можно указать любое, например:
git remote add github https://github.com/[ваш_ник]/[имя _репозитория].git

```
G:\repository>git remote add origin https://github.com/ad-frost/firstproject.git


G:\repository>git remote -v
origin  https://github.com/ad-frost/firstproject.git (fetch)
origin  https://github.com/ad-frost/firstproject.git (push)

G:\repository>git status
On branch master

No commits yet
```

Команды Git

14) git push [alias] [branch] - отправить изменения на удаленный репозиторий
15) git pull [alias] [branch] - взять изменения из удаленного репозитория
16) git clone [link] - склонировать удаленный репозиторий

Чтобы продублировать ваши локальные коммиты в удаленный репозиторий, примените команду **git push origin master**.
origin - это имя удаленного репозитория.
master - это имя ветки, коммиты из которой мы хотим отправить.

Для клонирования репозитория в папке с проектом можно не создавать локальный репозиторий. Достаточно просто зайти через консоль в папку с проектом и клонировать репозиторий.

Что почитать?

- https://git-scm.com/book/ru/v2
- https://www.hostinger.ru/rukovodstva/osnovi-git-chto-takoe-git#gref
- https://docs.google.com/document/d/1e5ubOuWmr9ekgE-nQnCebgMVW2oh8pMtQdnJjd491Bo
- https://habr.com/post/174467/
- https://guides.github.com/activities/hello-world/