**Контроль версий**, также известный как управление исходным кодом, — это практика отслеживания изменений программного кода и управления ими.

**Система контроля версий** — это программный инструменты, который позволяет управлять изменениями в исходном коде и других файлов проекта. Она позволяет отслеживать изменения, вносить их, восстанавливать предыдущие версии файлов и совместно работать над проектом с другими разработчиками.

Системы контроля не ограничиваются программным кодом. Они могут использоваться для управления версиями любых файлов: документации, графических фалов, настроек проекта.

Наиболее популярные типы VCS:

**Централизованные системы контроля версий (Centralized Version Control Systems, CVCS)** - такие как Subversion (SVN). В CVCS все файлы и история изменений хранятся на центральном сервере, а разработчики получают последние версии файлов из этого сервера и отправляют изменения на сервер. Это позволяет иметь единую историю изменений и контроль над доступом к файлам.

**Минусы:**

* Если сервер выйдет из строя на какое-то время, контроль версий будет недоступен. А если повредится жесткий диск, то вы потеряете всю историю проекта, не считая единичных снимков репозитория, которые сохранили другие разработчики.

**Распределенные системы контроля версий (Distributed Version Control Systems, DVCS) -** такие как Git и Mercurial. В DVCS каждый разработчик получает полную копию репозитория с историей изменений на свой локальный компьютер. Это позволяет им работать независимо друг от друга, без необходимости постоянного подключения к центральному серверу. Изменения могут быть коммитированы и синхронизированы с другими разработчиками при необходимости.

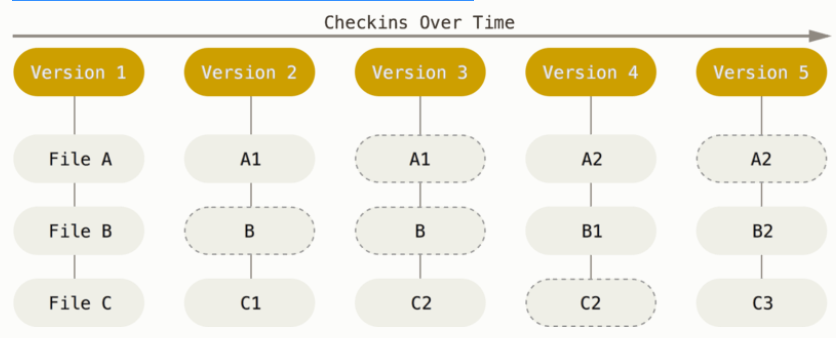
Если сервер, через который разработчики обменивались данными, умрет любой клиентский репозиторий может быть скопирован на другой сервер для продолжения работы. Каждая копия репозитория является полным бэкапом всех данных.

Программное обеспечение контроля версий отслеживает все вносимые в код изменения в специальной базе данных. При обнаружении ошибки разработчики могут вернуться назад и выполнить сравнение с более ранними версиями кода для исправления ошибок.

**GIT**

Большинство систем контроля версий хранят информацию в виде списка изменений в файлах. Эти системы (CVS, Subversion, Perforce, Bazaar и т. д.) представляют хранимую информацию в виде набора файлов и изменений, сделанных в каждом файле, по времени.

В Git, каждый раз, когда вы делаете коммит, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок. Если файлы не были изменены, то создается просто ссылка на предыдущую версию идентичного файла, который уже сохранен. Git представляет свои данные как, скажем, **поток снимков**.



Почти все операции в гит выполняются локально, если только вам не нужно получать какие-то данные с сервера. Если вам необходимо посмотреть изменения, сделанные между текущей версией файла и версией, созданной месяц назад, Git может найти файл месячной давности и локально вычислить изменения.

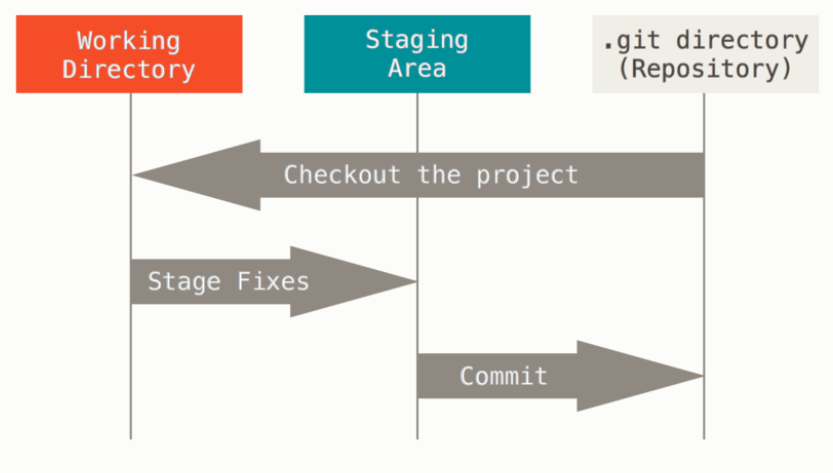
В Git для всего вычисляется хеш-сумма, и только потом происходит сохранение. В дальнейшем обращение к сохранённым объектам происходит по этой хеш-сумме. Это значит, что невозможно изменить содержимое файла или каталога так, чтобы Git не узнал об этом.

Файлы в GIT имеют 3 основных **состояния:**

**изменён (modified)** – файлы, которые поменялись, но еще не были закомичены.

**индексирован (staged) –** измененный файл, отмеченный для включения в следующий коммит.

**зафиксирован (committed):** файл сохранен в локальной базе.



Есть 3 основные секции:

**рабочая копия (working tree)** – снимок одной версии проекта. Эти файлы извлекаются из сжатой бд в каталоге Git и помещаются на диск, для того чтобы их можно было использовать или редактировать.

**Область индексирования (staging area) –** файл, обычно находящийся в каталоге Git. В нем содержится информация о том, что попадет в следующий коммит. Её техническое название на языке Git — «индекс».

**каталог Git (Git directory) -**это то место, где Git хранит метаданные и базу объектов вашего проекта. Это самая важная часть Git и это та часть, которая копируется при **клонировании** репозитория с другого компьютера.

**Базовый подход в работе с Git выглядит так:**

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

Если определённая версия файла есть в каталоге Git, эта версия считается **зафиксированной** (**committed**). Если файл был изменён и добавлен в индекс, значит, он **индексирован** (**staged**). И если файл был изменён с момента последнего распаковывания из репозитория, но не был добавлен в индекс, он считается **изменённым** (**modified**).

В состав Git входит утилита **git config**, которая позволяет просматривать и настраивать параметры, контролирующие все аспекты работы Git, а также его внешний вид. Эти параметры могут быть сохранены в трёх местах:

1. **SYSTEM.** Файл **[path]/etc/gitconfig** содержит значения, общие для всех пользователей системы и для всех их репозиториев. Если при запуске git config указать параметр **--system**, то параметры будут читаться и сохраняться именно в этот файл. Является системным файлом.
2. **GLOBAL.** хранит настройки конкретного пользователя. ~/.config/git/config, C:\Users\$USER.
3. **LOCAL.** Файл config  в каталоге Git (т. е. .git/config) репозитория, который вы используете в данный момент, хранит настройки конкретного репозитория.

Настройки на каждом следующем уровне перекрывают настройки предыдущих уровней. Чтобы посмотреть все установленные настройки и узнать, где именно они заданы, используйте команду:

$ git config --list --show-origin

Чтобы посмотреть конкретную настройку:

$ git config user.name

**Основные настройки:**

* Имя пользователя $ git config --global user.name "John Doe"
* Email $ git config --global user.email johndoe@example.com
* Текстовый редактор, чтобы набирать сообщение git. $ git config --global core.editor emacs
* Имя ветки по умолчанию (изначально **master**) $ git config --global init.defaultBranch main

**Чтобы получить git-репозиторий можно:**

**Клонировать** существующий Git репозиторий.

**Создать** Git репозиторий в существующем каталоге. Выполняется это командой

$ git init

Эта команда создаёт в текущем каталоге новый подкаталог с именем .git, содержащий все необходимые файлы репозитория — структуру Git репозитория.

**git clone <url> [catalogname] –** получить Git репозиторий с сервера в указанный каталог. Cкачивает все данные для этого репозитория и извлекает рабочую копию последней версии. В качестве транспортных протоколов можно использовать http или SSH.

**git add –** добавить файл(файлы в индекс).

**git commit** – зафиксировать изменения в каталоге Git.

https://git-scm.com/book/ru/v2/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%8B-Git-%D0%97%D0%B0%D0%BF%D0%B8%D1%81%D1%8C-%D0%B8%D0%B7%D0%BC%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D0%B9-%D0%B2-%D1%80%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D0%B9