**Система контроля версий (СКВ)** — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов.

## Локальные системы контроля версий представляют из себя простейшую базу данных, которая хранит записи обо всех изменениях в файлах. Одним из примеров таких систем является система контроля версий RCS, которая была разработана в 1985 году (последний патч был написан в 2015 году) и хранит изменений в файлах (патчи), осуществляя контроль версий. Набор этих изменений позволяет восстановить любое состояние файла. RCS поставляется с Linux'ом. Локальная система контроля версий хорошо решает поставленную перед ней задачу, однако ее проблемой является основное свойство — локальность. Она совершенно не преднезначена для коллективного использования.

**Централизованная система контроля версий** предназначена для решения основной проблемы локальной системы контроля версий.  
 Для организации такой системы контроля версий используется единственный сервер, который содержит все версии файлов. Клиенты, обращаясь к этому серверу, получают из этого централизованного хранилища. Применение централизованных систем контроля версий на протяжении многих лет являлась стандартом. К ним относятся CVS, Subversion, Perforce.  
 Такими системами легко управлять из-за наличия единственного сервера. Но при этом наличие централизованного сервера приводит к возникновению единой точки отказа в виде этого самого сервера. В случае отключения этого сервера разработчики не смогут выкачивать файлы. Самым худшим сценарием является физическое уничтожение сервера (или вылет жесткого диска), он приводит к потерю кодовой базы.  
 Несмотря на то, что мода на SVN прошла, иногда наблюдается обратный ход — переход от Git'а к SVN'у. Дело в том, что SVN позволяет осуществлять селективный чекаут, который подразумевает выкачку лишь некоторых файлов с сервера. Такой подход приобретает популярность при использовании монорепозиториях, о которых можно будет поговорить позже.

## Распределенная система контроля версий

Для устранения единой точки отказа используются распределенные системы контроля версий. Они подразумевают, что клиент выкачает себе весь репозиторий целиком заместо выкачки конкретных интересующих клиента файлов. Если умрет любая копия репозитория, то это не приведет к потере кодовой базы, поскольку она может быть восстановлена с компьютера любого разработчика. Каждая копия является полным бэкапом данных.  
 Все копии являются равноправным и могут синхронизироваться между собой. Подобный подход очень напоминает (да и является) репликацией вида master-master.  
 К данному виду систем контроля версий относятся Mercurial, Bazaar, Darcs и Git.

**Git** относится к третьему поколению систем контроля версий (СКВ). Они называются распределенными СКВ, поскольку имеют хранилище данных (репозиторий) не только на сервере, но и локально на тех машинах, которыми пользуются разработчики.

От других СКВ **Git отличается** особым подходом к обработке информации: он не записывает отдельно внесенные правки, а делает подробный снимок проекта в момент сохранения, то есть фиксирует состояние каждого файла, и создает ссылку на эту версию.

Команды:

**Команда git init** создает новый репозиторий Git. С ее помощью можно преобразовать существующий проект без управления версиями в репозиторий Git или инициализировать новый пустой репозиторий. Большинство остальных команд Git невозможно использовать без инициализации репозитория, поэтому данная команда обычно выполняется первой в рамках нового проекта.

**Команда git add** — это первая команда в цепочке операций, предписывающей Git «сохранить» снимок текущего состояния проекта в истории коммитов. Когда git add используется как отдельная команда, она переносит ожидающие изменения из рабочего каталога в раздел проиндексированных файлов. Команда git status проверяет текущее состояние репозитория; с ее помощью можно убедиться, что команда git add добавила нужные изменения.

**Команда git commit** – это команда Git для записи индексированных изменений в репозиторий. Используйте эту метку для обозначения всех вопросов, связанных с созданием, редактированием и внутренней структурой коммитов в Git.

Команда git commit - это команда для записи индексированных изменений в репозиторий Git. Прежде чем создавать очередной коммит, необходимо проиндексировать файлы в рабочей области с помощью команды git-add.

**Команда git status** отображает состояние рабочего каталога и раздела проиндексированных файлов. С ее помощью можно проверить индексацию изменений и увидеть файлы, которые не отслеживаются Git. Информация об истории коммитов проекта не отображается при выводе данных о состоянии. Для этого используется команда git log. Связанные команды git. git tag Теги — это ссылки, указывающие на определенные точки в истории Git.

**Команда git reset** позволяет буквально вернуться в прошлое - Вы сообщаете к какой версии вы хотите вернуться, и она восстанавливает именно это состояние — отменяя все изменения, которые были осуществлены после этой временной точки. Нужно указать ей коммит, к которому вы хотите вернуться:

$ git reset --hard 2be18d9.

Чтобы восстановить просто самый свежий коммит:

$ git reset --hard HEAD.

**Git log** — разнообразная информация о коммитах в целом, по отдельным файлам и различной глубины погружения в историю.

Иногда требуется получить информацию об истории коммитов, коммитах, изменивших отдельный файл; коммитах за определенный отрезок времени и так далее. Для этих целей используется команда git log.

**Команда git branch** – это команда Git для управления ветками. Используйте эту метку для обозначения всех вопросов, связанных с ветками, их созданием, структурой, управлением и удалением. Общие сведения. git branch – это команда для управления ветками в репозитории Git. Ветка в Git'е — это просто «скользящий» указатель на один из коммитов. Когда вы создаёте новые коммиты, указатель ветки автоматически сдвигается вперёд, к вновь созданному коммиту.