Лекция 3

Системы контроля версий. Git.

Система управления версиями (от англ. *Version Control System, VCS* или *Revision Control System*) — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

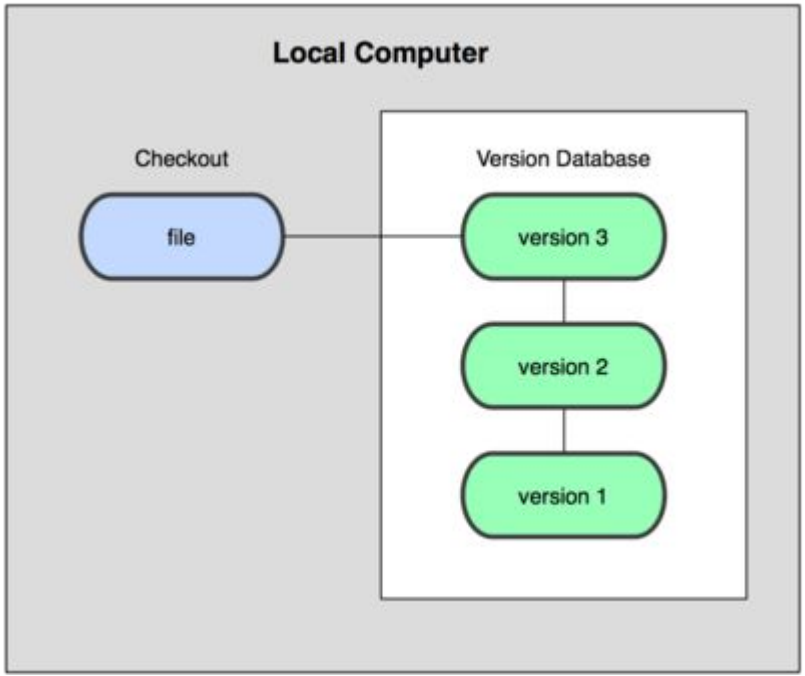
Все системы контроля версий можно разделить на 3 основных типа:

- локальные
- централизованные
- распределенные.

Локальные СКВ.

Локальные СКВ - СКВ, которые работают в пределах одного ПК, фиксация изменений производится путем сохранения изменений нужных файлов в простой базе данных.

Недостатком такой СКВ является то, что работает она локально в пределах одного ПК.

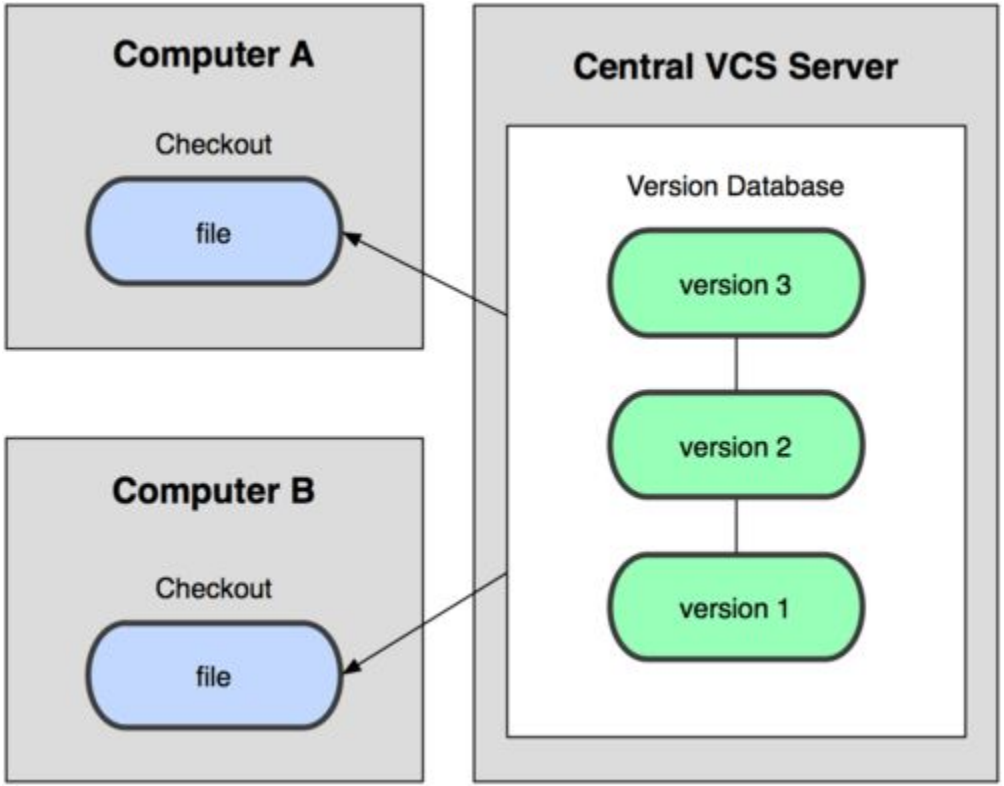


Централизованные СКВ.

Проблема совместного доступа к ресурсам подтолкнула разработчиков на создание централизованных СКВ. Такие системы имеют центральный сервер, где хранятся файлы под версионным контролем, и ряд клиентов на ПК пользователей, которые получают копии.

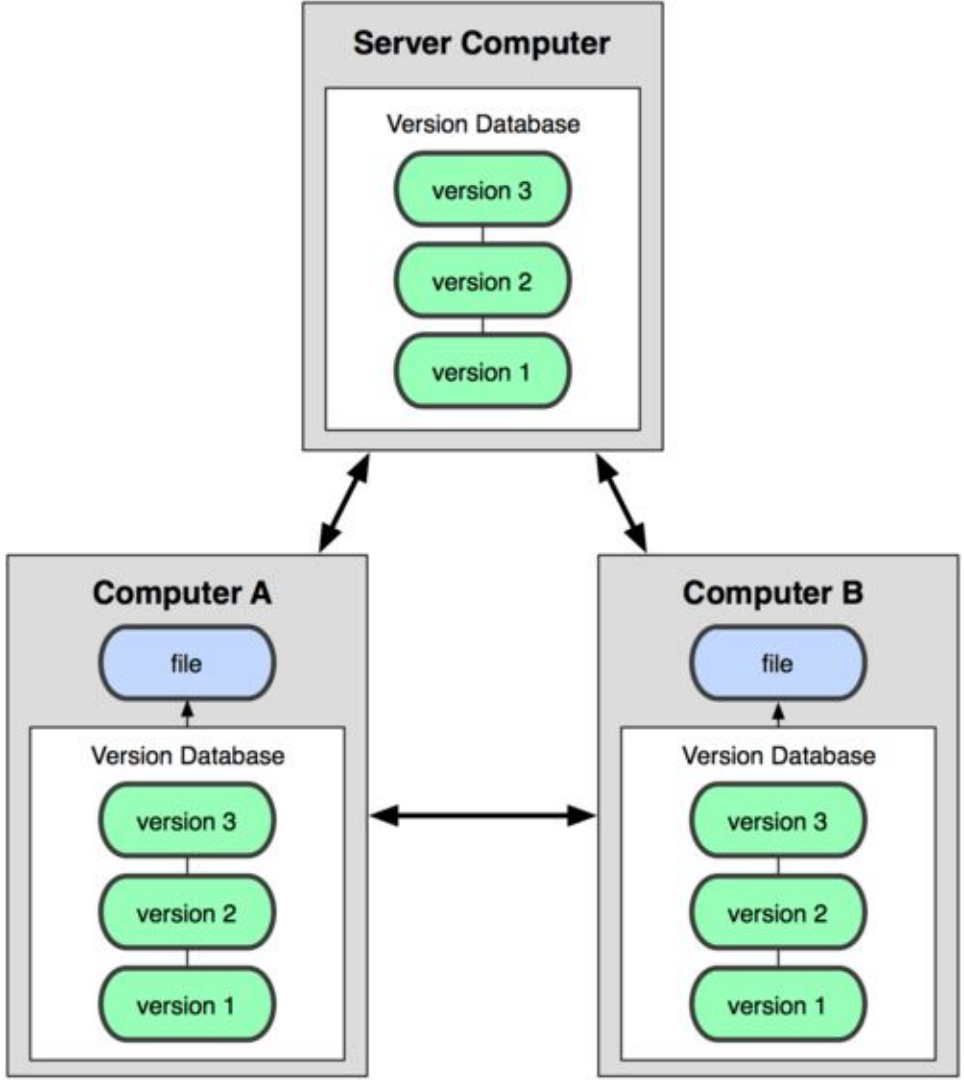
Данный подход имеет ряд преимуществ над локальными СКВ, такие как возможность совместной работы с файлами, организация доступа к файлам а так же относительная легкость администрирования, по сравнению с локальными СКВ.

Тем не менее они имеют и ряд недостатков - если теряется связь с сервером - фиксация изменений а так же обновление становятся невозможными, более того, при физическом повреждении носителей теряется вся информация, за исключением нескольких рабочих копий, которые находились на локальных машинах.


Computer A
Checkout
file
Computer B
Checkout
file
Central VCS Server
Version Database
version 3
version 2
version 1

Распределенные СКВ.

Следующей вехой в развитии СКВ стали распределенные СКВ. Основным преимуществом таких систем является то, что пользователь при получении данных с СКВ получает не конкретную версию, а полную копию репозитория со всей историей изменений, поэтому в случае технической неисправности центрального хранилища вся база данных может быть восстановлена с любой пользовательской машины .


Server Computer
Version Database
version 3
version 2
version 1
Computer A
file
Version Database
version 3
version 2
version 1
Computer B
file
Version Database
version 3
version 2
version 1

Что такое Git.

Git - популярная распределенная система контроля версий, разработанная Линусом Торвальдсом для управления разработкой ядра Linux. Программа является свободной и выпущена под лицензией GNU GPL версии 2.

Git спроэктирован как набор программ, разработанных с учетом их использования в скриптах, поддерживает быстрое разделение и слияние версий, включает инструменты для визуализации и навигации по нелинейной истории разработки.

Для общения с удаленным сервером используются 3 протокола:

- git:// - открытый протокол, не имеющий аутентификации, тредует запещенного демона на серверной стороне
- ssh:// - использует аутентификацию посредством пар ключей, требует создания аккаунтов со стороны сервера
- http(s):// - исподьзует в своем механизме утилиту curl, которая для Windows поставляется вместе с git, а так же его возможности http-аутентификации, как и поддержку SSL и сертификатов.

Начало работы с Git.

Для того, чтобы начать использовать Git необходимо произвести его установку и начальную настройку.

Для операционных систем Windows Git можно скачать перейдя по ссылке https://git-for-windows.github.io/, после чего произвести установку скачанного пакета. В результате будут установлены ядро Git, а так же ряд компонентов, необходимых для работы с ним.

Первоначальная настройка.

Первоначальная настройка подразумевает задание набора параметров, необходимых для работы Git и идентификации/аутентификации пользователя, совершающего действия.

Для указания настроек Git использует утилиту **git config**.

Так для идентификации пользователя нам необходимо указать имя пользователя и E-mail, что можно сделать выполнив в консоли git-клиента следующие строки:

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

и

git config --global user.email youremail@some.domain

Так же можно проверить текущие настройки, вызвав утилиту git config и указав ключ, значение которого необходимо узнать. Например, если нам необходимо узнать имя текущего пользователя, то необходимо выполнить следующую команду

git config user.name

результатом выполнения которой будет вывод имени пользователя, которое было введено при первичной конфигурации системы.

Настройка локального репозитория и работа с ним.

Для того, чтобы создать локальный репозиторий необходимо перейти в необходимую директорию и выполнить команду **git init**. Результатом выполнения данной команды будет создание репозитория, готового для работы с Git и имеющего минимально необходимые настройки.

Подключение к удаленному репозиторию.

После создания локального репозитория его необходимо подключить к удаленному, для этого нужно выполнить в консоли команду

git remote add [name] [url]

где name - имя репозитория (опционально выбранное), url - путь к репозиторию.

Клонирование удаленного репозитория.

Для клонирования уже существующего репозитория необходимо выполнить команду

git clone [protocol://path.git] [folder name]

где protocol - протокол, по которому производится действие, path - путь к репозиторию, folder name - имя директории, которая будет создана при клонировании репозитория (данный параметр не является обязательным).

Основные операции с репозиторием.

Основные операции, проводимые над репозиторием это добавление изменений, их фиксация локально, фиксация на удаленном сервере, получение редакции с удаленного сервера.

Если говорить кратко, то большинство операций git производятся локально, т.к. мы имеем полную копию репозитория на своем ПК. Исключение составляют лишь те операции, которые предполагают работу с удаленным репозиторием (получение редакции, фиксация изменений на удаленном сервере).

Для работы с репозиторием необходимо находиться в его корне, там где лежат все настройки репозитория, а так же хранится вся информация о нем.

Добавление изменений в репозиторий.

Для добавления модифицированных данных используется команда **git add,** синтаксис которой выглядит следующим образом:

git add [path]

где path - путь к файлам/файлу или же папкам, изменения которых необходимо зафиксировать.

Выполнив данную команду утилита произведет добавление информации об измененных файлах, которая пойдет в следующую фиксацию изменений. Данная операция является локальной и не затрагивает удаленный репозиторий.

Фиксация изменений.

Фиксация изменений (commit) так же является локальной и затрагивает исключительно локальный репозиторий. Фиксация может быть произведена только после добавления текущих изменений репозитория.

Типичный синтаксис команды следующий:

git commit -m “commit message”

где аргумент -m указывает на то, что для фиксации изменений необходимо добавить описание изменений либо другую текстовую информацию о фиксации, commit message - собственно само описание либо другая информация.

Фиксация на удаленном сервере.

Для фиксации всех изменений на удаленном сервере используется команда push, синтаксис которой следующий:

git push [repo_name] [branch]

где repo_name - имя удаленного репозитория, на котором фиксируются наши изменения, branch - ветвь, для которой вносятся текущие изменения. Данные параметры являются опциональными, если их не указать, то будут использованы данные по умолчанию - репозиторий по умолчанию а так же текущая ветвь.

Получение редакции.

Для получения редакции с удаленного репозитория используют команды fetch или pull. Синтаксис данных команд следующий:

git fetch [repo_name]

git pull [repo_name]

где repo_name - имя (псевдоним) удаленного репозитория.

Однако данные команды имеют существенные отличия - fetch получает все данные с удаленого репозитория, которых нет в вашем, но не производит их слияния (автоматической модификации), позволяя выполнить данные действия пользователю. pull же наоборот, получая данные с удаленного репозитория пытается произвести автоматическое слияние всех полученных изменений с теми наработками, которые имеются локально.