Инструменты контроля версий

Лекция №1, СибГУ, ИСИС, Сафиуллина Н.Ф.

План

1) Системы контроля версий
2) Сервис GitHub
3) Основы работы с Git
4) Что такое Git ветка
5) Командная работа

1 Системы контроля версий

Система контроля версий (далее СКВ) — это программное обеспечение, записывающее изменения в файлах, охваченных системой, в специальный файл или набор специальных файлов, и позволяющее вернуться позже к определённой версии отслеживаемого системой файла.

1.1 Виды систем контроля версий

– локальная,

– централизованная,

– децентрализованная.

1.1.1 Локальные системы контроля версий

Локальные системы контроля версий хранят информацию о всех изменениях, внесенных в файлы в базе данных, которая находится там же локально на компьютере.

В локальных СКВ обычно хранят список изменений, внесенных в файлы.

Достоинства:

– Возможность восстановления данных из определенной версии.

– Высокая скорость выполнения восстановления .

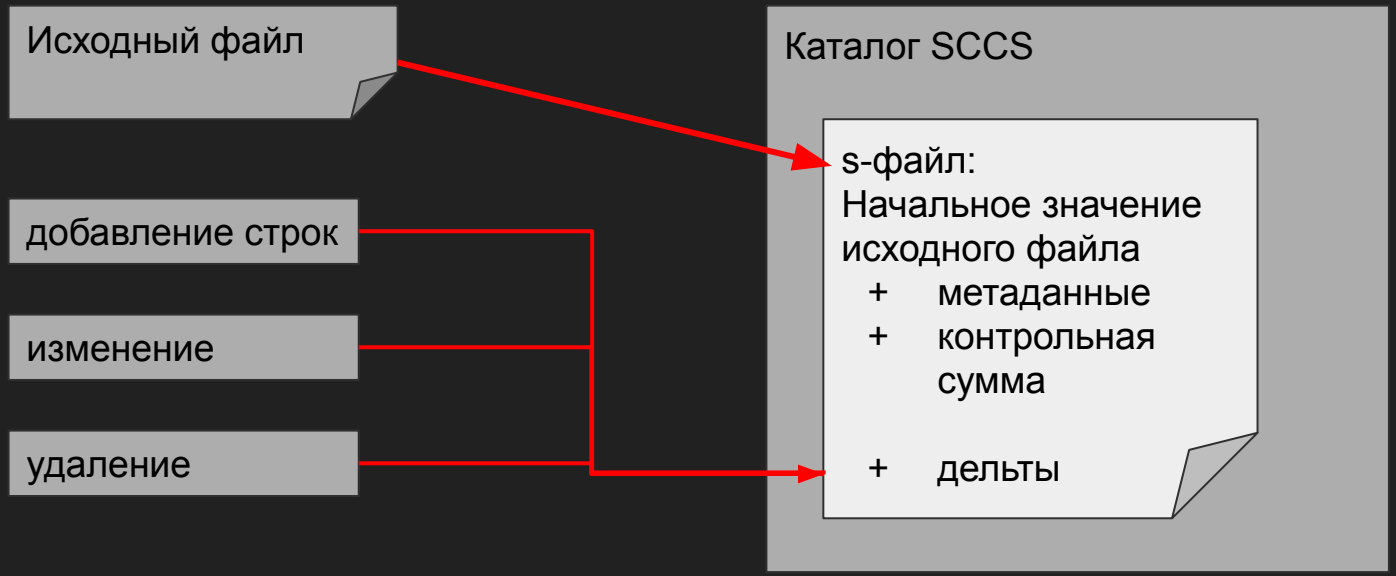
Недостатки:

– Возможность потери данных вследствие возникновения физических поломок оборудования.

– Сложность совместной разработки. Одномоментно изменения в определённый файл могут быть внесены только одним пользователем.

Пример локальной СКВ: SCCS

SCCS *(source code control system)*



1.1.2 Централизованные системы контроля версий

В централизованных системах есть удаленный репозиторий. Проекты, над которыми работают разработчики, импортируются в локальные репозитории.

Преимущество:

– возможность распределенной разработки

– полный контроль администраторов над тем, кто и что может делать.

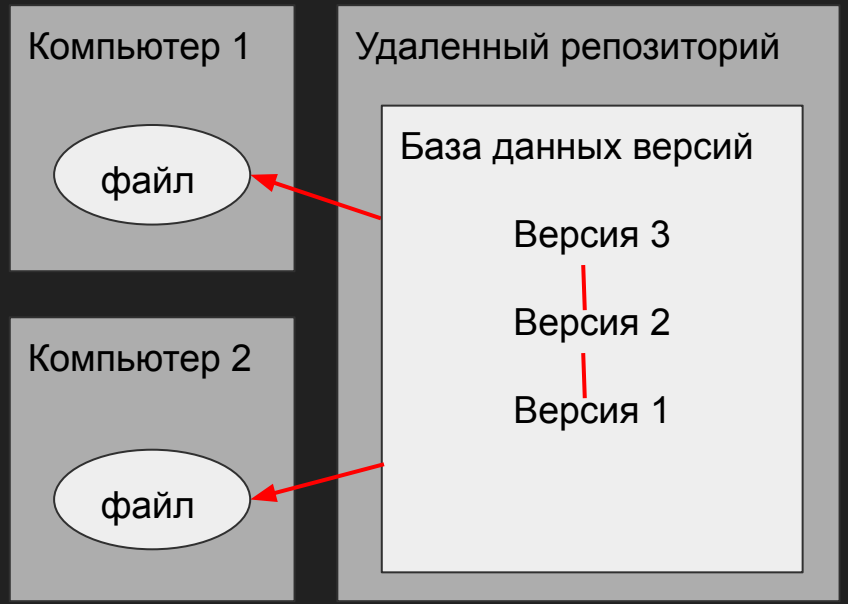
Недостаток:

– единая точка отказа, которой является центральный сервер.

Примеры:

CVS (*Concurrent version system*)

SVN (*Subversion*)



1.1.3 Децентрализованные системы контроля версий

Git (от англ. — *Global Information Tracker*) — это распределённая система управления версиями.

– децентрализованная, центрального репозитория не существует: все копии создаются равными;

– разработчики могут обмениваться изменениями друг с другом перед объединением своих изменений в официальную ветвь;

– разработчики могут вносить свои изменения в локальную копию репозитория и могут работать локально в автономном режиме, пока не будут готовы поделиться своей работой с другими;

– когда файл добавляется для отслеживания в *Git*, он сжимается с помощью алгоритма сжатия *zlib;*

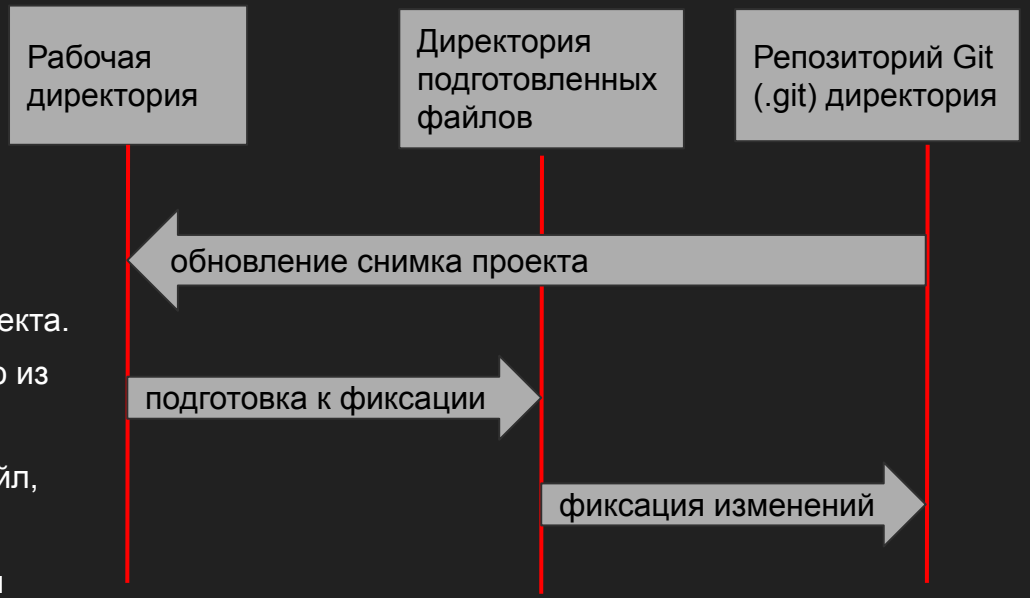
– система делает снимок того, как выглядит каждый файл в данный момент, и сохраняет ссылку на этот снимок каждый раз, когда пользователь делает фиксацию изменений в своём проекте;

– если файл не менялся, то *Git* не запоминает этот файл вновь, а просто сохраняет ссылку на предыдущую версию этого файла, который был уже сохранён.

1.3 Основные состояния Git

Git имеет три основных состояния,
в которых могут находиться файлы репозитория:

1. Измененное.
2. Подготовленное.
3. Зафиксированное.



Рабочая директория — это снимок версии проекта. *Git* распаковывает файлы в рабочую директорию из сжатой базы данных в *Git* директории.

Область подготовленных файлов — это файл, располагающийся в *Git* директории, в котором содержится информация о том, какие изменения попадут в следующий коммит. Эту область еще называют «индекс».

2 Сервис GitHub

GitHub – это одно из облачных решений для развертывания Git репозиториев.

Другие решения: *GitLab*, *BitBucket*, *Microsoft Azur DevOps Repos*, *Microsoft VSTS*, *AWS CodeCommit* и другие.

Ключевые особенности *GitHub*, помимо функций *Git* репозитория:

– Настройка вариантов совместной работы команды с файлами репозитория.

– Публичные и личные репозитории для бесплатного аккаунта.

– Клонирование репозитория целиком с помощью интерфейсов локально установленного Git и сервиса GitHub.

– Создание организации в рамках своего аккаунта, добавление репозиториев и приглашение пользователей для участия. Гибкое назначение им прав на работу с файлами репозитория.

– Функционирование в качестве полноценной системы управления проектами.

– Наличие API, благодаря которому можно интегрировать GitHub в другие приложения, например, в системы управления проектами типа JIRA.

– Подсветка синтаксиса для большинства языков.

– Наличие справочной системы, wiki, для любого репозитория.

– Платные аккаунты для работы с GitHub добавляют массу других весьма полезных функций.

3.3 Основы Git: содержимое директории .git

– В файле config находятся настройки данного репозитория.

– Файл HEAD указывает на текущую ветку.

– В файле index хранится содержимое индекса.

– В директории objects находится, собственно, база данных объектов Git. Если открыть каталог objects, то в нём будут находиться каталоги, имена которых представлены двумя шестнадцатеричными числами, внутри которых будут файлы, имена которых представлены 38 шестнадцатеричными числами. Вместе имя каталога и файл образуют 40-разрядный хэш, взятый от имени файла и его содержимого.

– В директории refs находятся ссылки на объекты коммитов в этой базе (ветки).

– Директория logs хранит логи коммитов.

– В директории info расположен файл с глобальными настройкам игнорирования файлов. Он позволяет исключить из работы файлы, которые вы не хотите помещать в .gitignore. Позднее мы поговорим о назначении файла .gitignore.

– В директории hooks располагаются клиентские и серверные триггеры.

3.4 Команды для работы с репозиторием

Что вы можете делать с файлами:

– Добавлять, изменять и удалять файлы, находящиеся в вашем локальном репозитории.

– Выполнять синхронизацию с репозиторием на удаленном сервере (в нашем случае с репозиторием на GitHub).

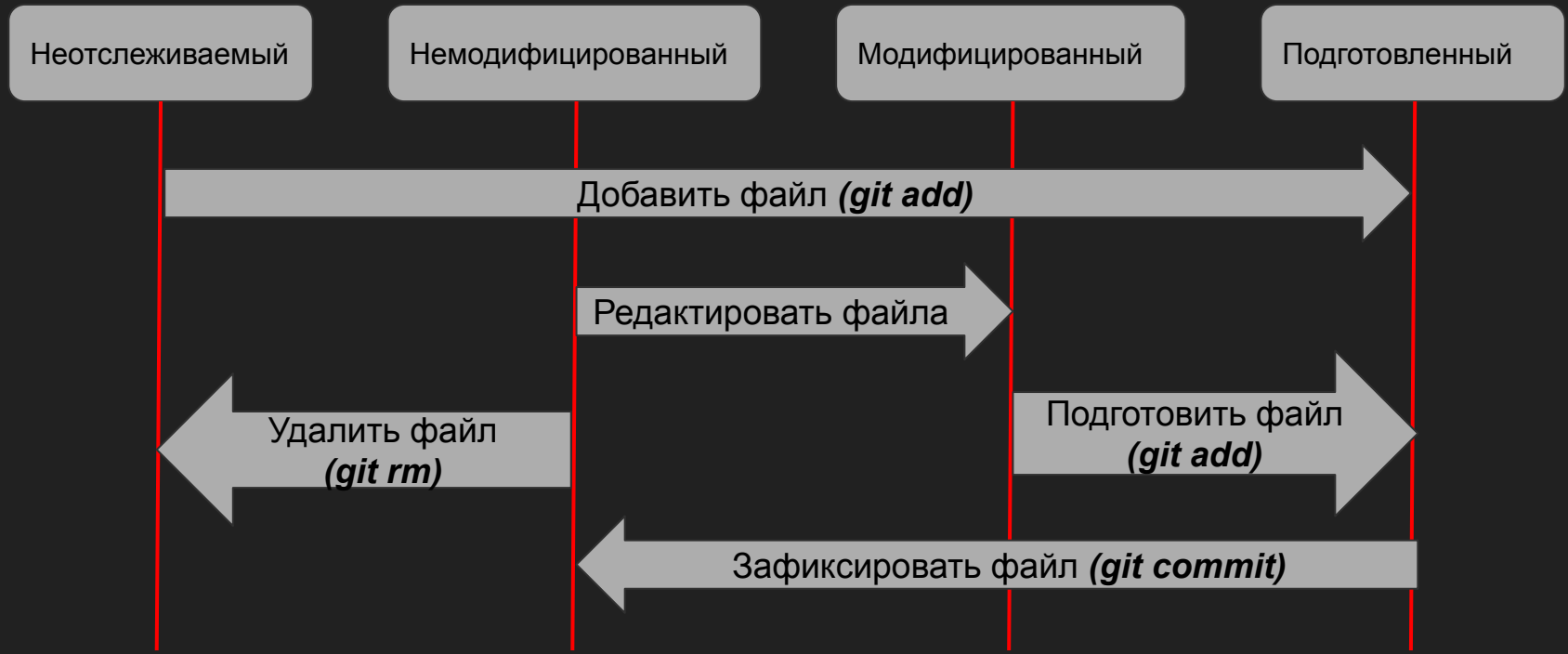
git status — команда получения статуса файлов в репозитории, основная команда для отслеживания изменений файлов, находящихся в репозитории. Команда выводит информацию о тех файлах репозитория, которые пока ещё не находятся под версионным контролем или не отслеживаются, и о тех файлах, которые были изменены, и изменения которых еще не были зафиксированы.

git add — переносит файл в директорию подготовленных файлов. Эта команда принимает как параметр имя файлов, которые нужно перенести в подготовленные для фиксации.

git commit — производит фиксацию изменений.

git push — публикует изменения на сервер.

3.8 Жизненный цикл состояний файлов



4 Что такое ветка в Git

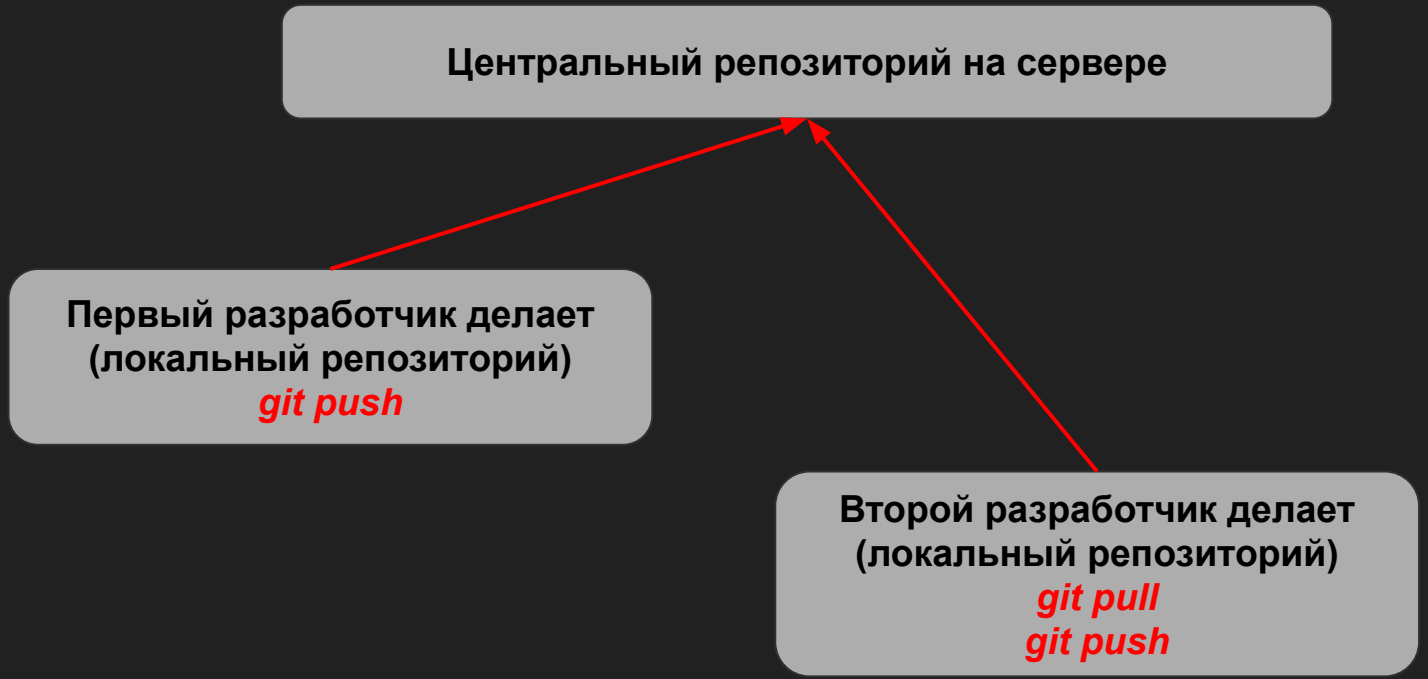
Ветка в Git — это подвижный указатель на один из коммитов. Обычно ветка указывает на последний коммит в цепочке коммитов. Ветка берёт своё начало от какого-то одного коммита.

Ветки нужны для добавления новой функциональности, не изменяя основную.

Слияние веток делается командой: ***git merge***.

Слияние веток это однонаправленный процесс, если мы сливаем ветки, то сливаем одну ветку в другую, а не каждую с каждой.

5.1 Командная работа: централизованная модель



5.2 Командная работа: “менеджер по интеграции”

Шаги процесса по публикации изменений в основной репозиторий следующие:

Разработчики клонируют основной репозиторий, и каждый создаёт локальную и публичную копии.

Разработчики отправляют изменения в свои публичные репозитории.

Разработчики отправляют запрос менеджеру по интеграции, имеющему доступ по записи в основной репозиторий, чтобы тот загрузил их изменения в свой локальный репозиторий.

Менеджер по интеграции загружает изменения, объединяет их с кодом в своём локальном репозитории.

Менеджер по интеграции публикует изменения в основной репозиторий.