Отчет по лабораторной работе №2

Операционные системы

Глущенко Евгений Игоревич

Содержание

# 1 Цель работы

Цель данной работы - изучить идеологию и применение средств контроля версий, освоить умения по работе с git.

# 2 Задание

Создать базовую конфигурацию для работы с git.  
Создать ключ SSH.  
Создать ключ PGP.  
Настроить подписи git.  
Зарегистрироваться на Github.  
Создать локальный каталог для выполнения заданий по предмету.

# 3 Выполнение лабораторной работы

##Установка программного обеспечения

Устанавливаю необходимое обеспеченье через терминал (рис fig. 1)

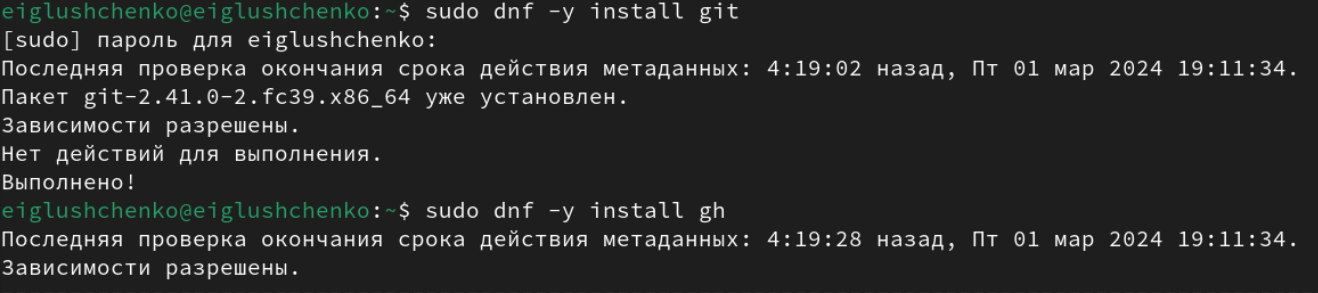


Рис. 1: Установка git и gh

##Базовая настройка git

Записываю свои данные в строках (почта и имя) (рис fig. 1)

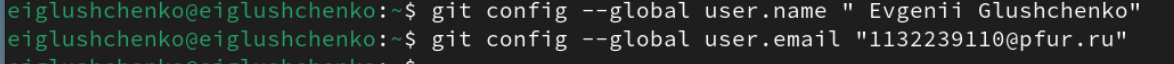


Рис. 2: Задаю email и имя

Настраиваю utf-8 в выводе сообщений (рис fig. 3)

Настраиваю utf-8

Рис. 3: Настраиваю utf-8

Задаю имя master начальной ветке (рис fig. 4)

Задаю имя master

Рис. 4: Задаю имя master

Задаю параметры для коректного отображения (рис fig. 5)

Задаю autoclrf и safeclrf

Рис. 5: Задаю autoclrf и safeclrf

##Создайте ключи ssh

Создаю ssh ключ размером 4096 бит (рис fig. 6)

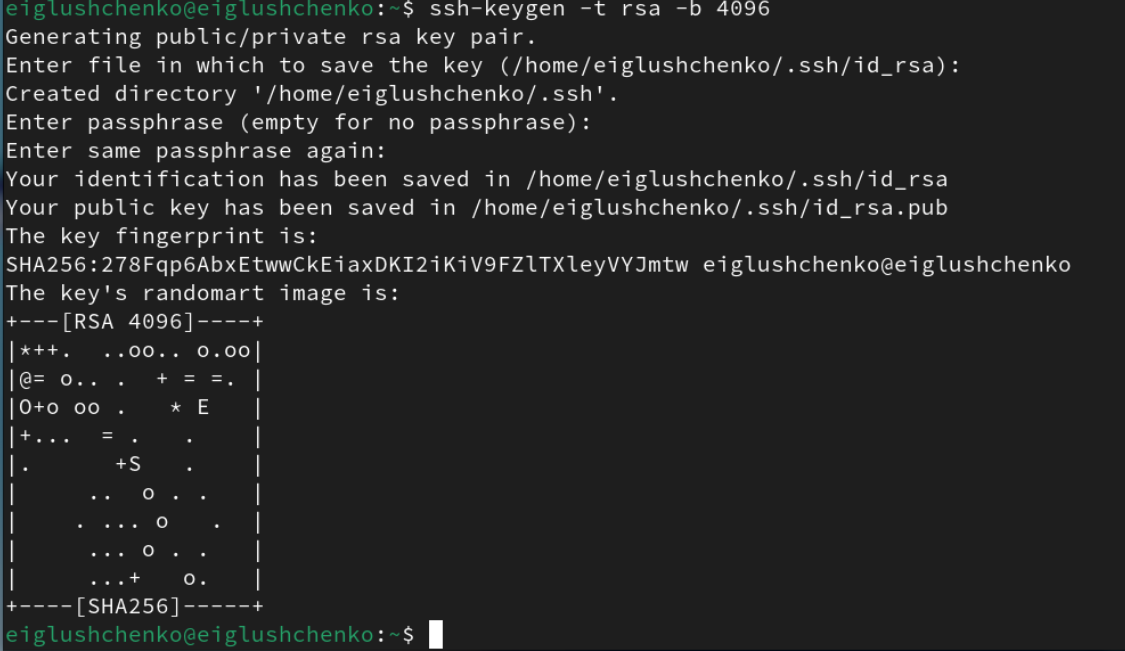


Рис. 6: Создания SSH ключа

Создаю ключ по алгоритму ed25519 (рис fig. 7)

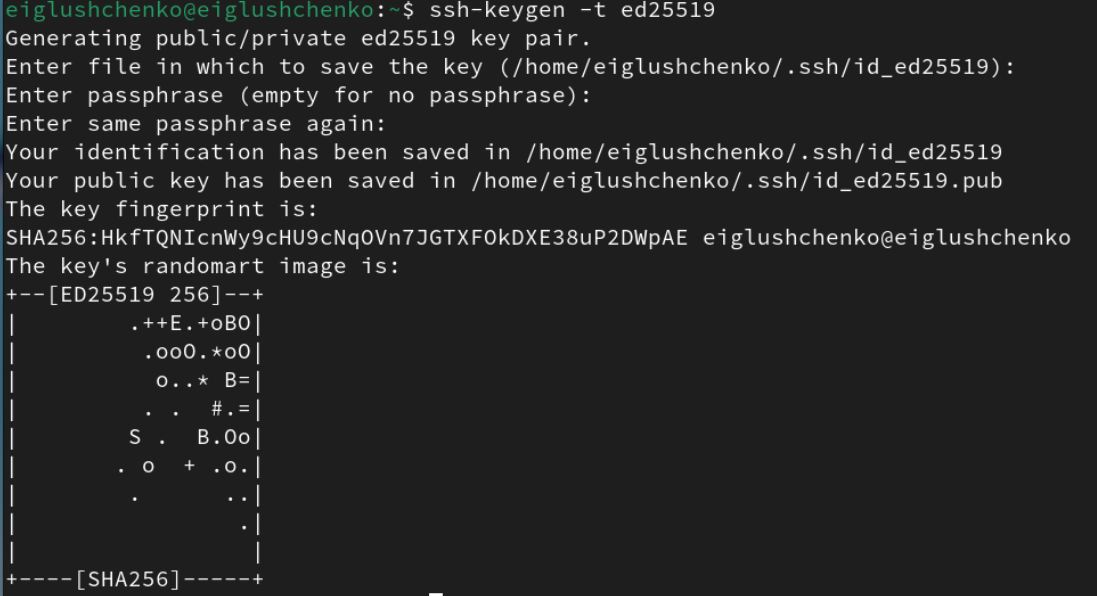


Рис. 7: Создание ed25519

##Создайте ключи pgp

Генерирую ключ GPG, затем выбираю тип ключа RSA and RSA, задаю максим мальную длину ключа: 4096, оставляю неограниченный срок действия ключа. Далее отвечаю на вопросы программы о личной информации (рис fig. 8)

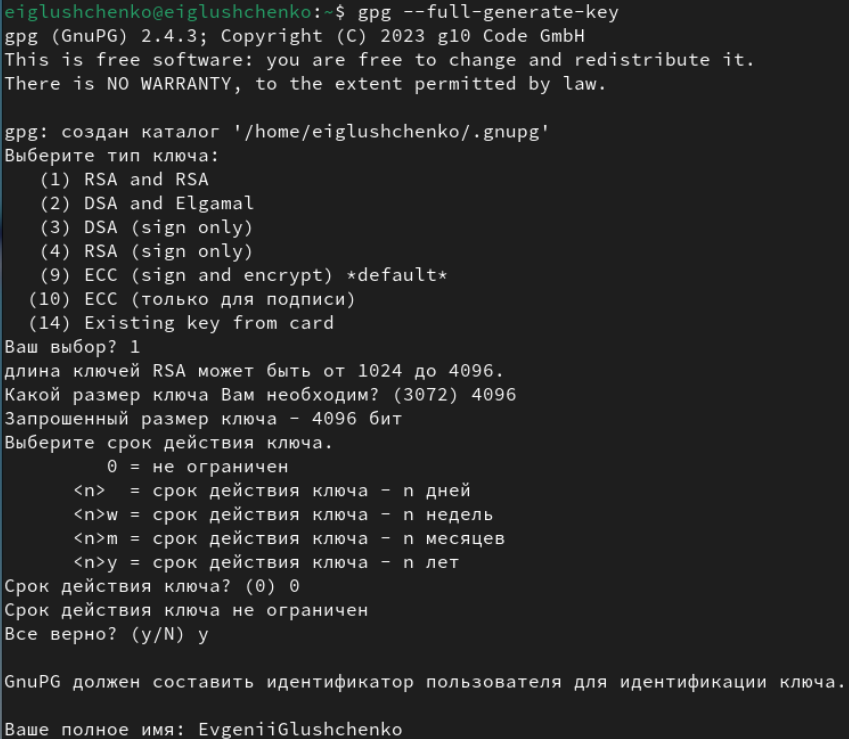


Рис. 8: Генирация ключа

Создаю защитную фразу (рис fig. 9)

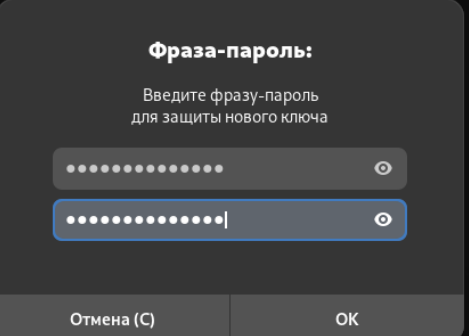


Рис. 9: Пароль для ключа

##Настройка github

У меня имеется профиль в гитхабе, работаю на нем (рис fig. **¿fig:010?**)

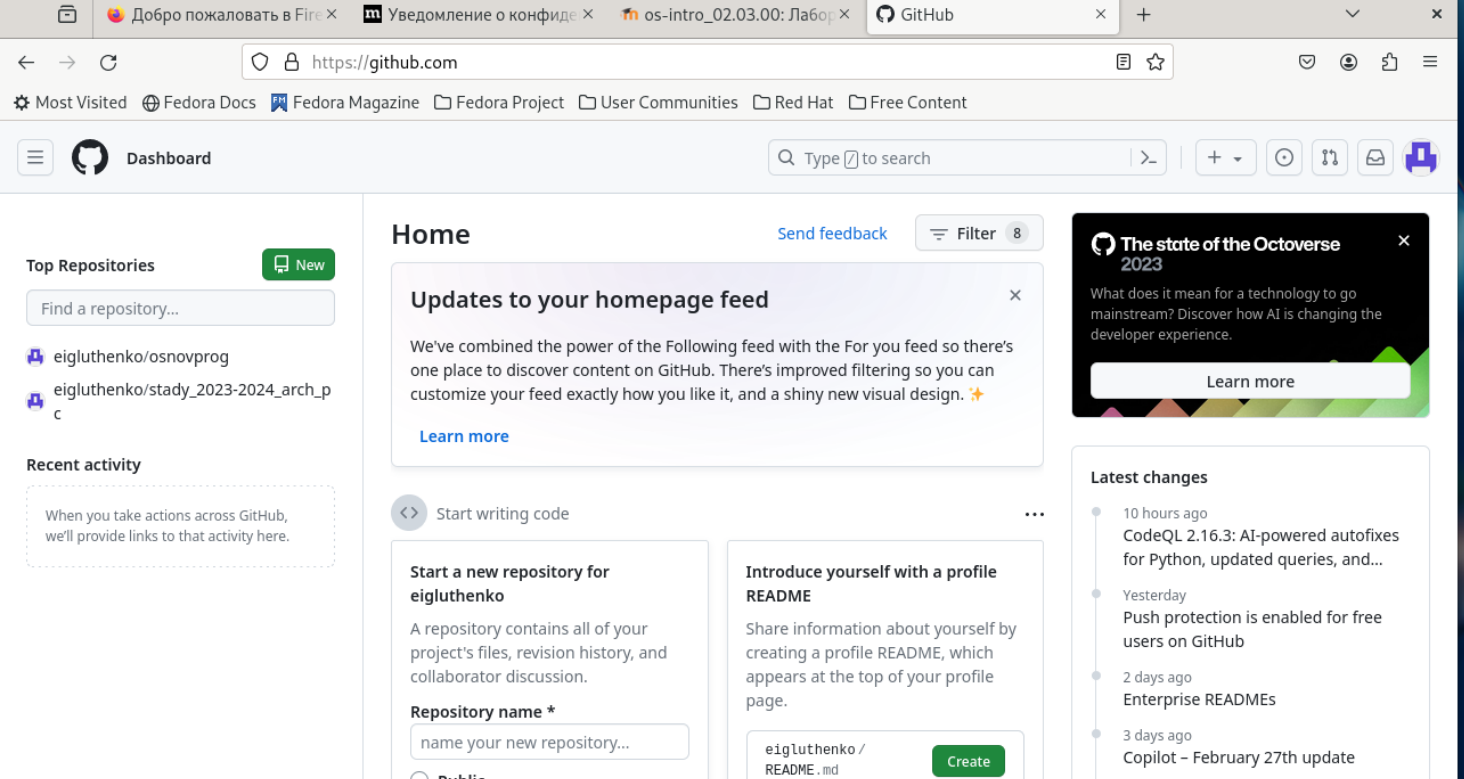


Рис. 10: Профиль в github

##Добавление PGP ключа в GitHub

Вывожу список созданных ключей в терминал, ищу в результате запроса отпечаток ключа он стоит после знака слеша, копирую его в буфер обмена (рис fig. 11)

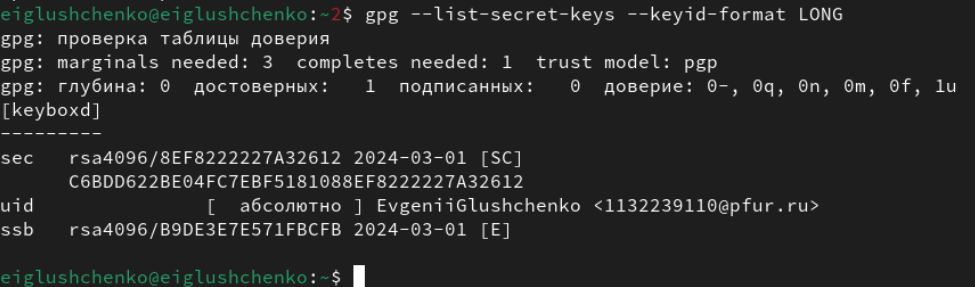


Рис. 11: ВЫвод списка ключей

Копирую ключ с помощью xclip (рис fig. 12)

Копирую ключ

Рис. 12: Копирую ключ

Вставляю ключ в GitHub (рис fig. 13)

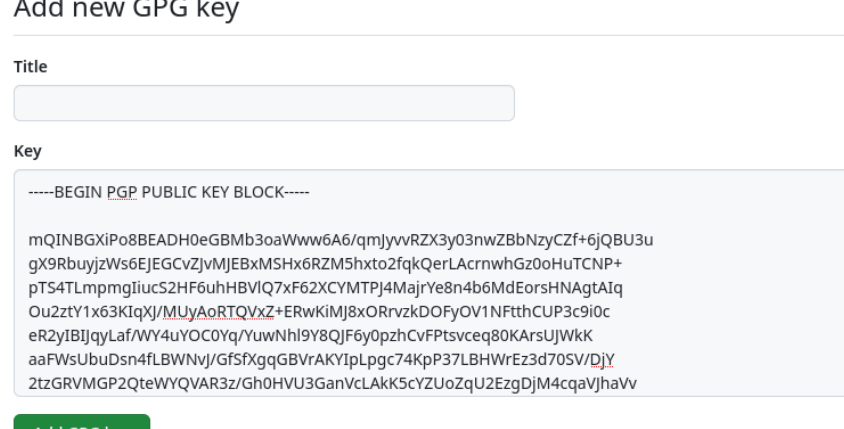


Рис. 13: Ключ

Ключ добавлен (рис fig. 14)

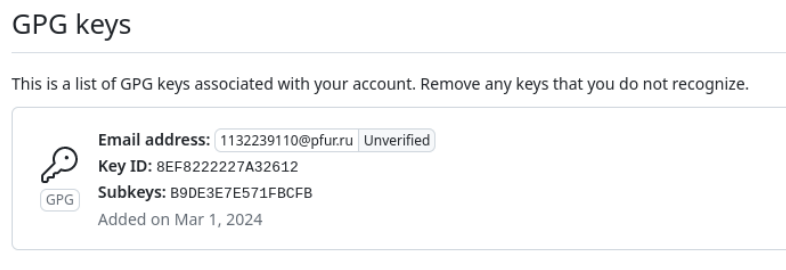


Рис. 14: Проверка ключа

##Настройка автоматических подписей коммитов git

Настраиваю автоматические подписи коммитов git: используя введенный ранее email, указываю git использовать его при создании подписей коммитов (рис fig. 15)

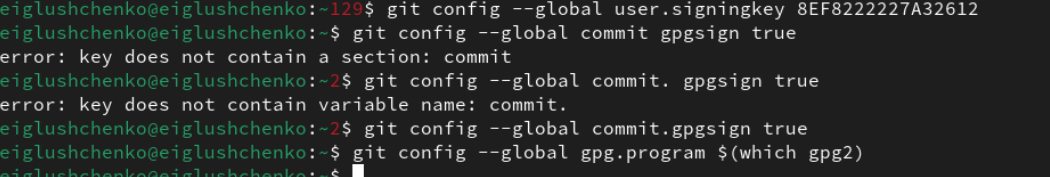


Рис. 15: Подписи GIT

##Настройка gh

Настраиваю gh, отвечаю на вопросы, авторизуюсь через браузер (рис fig. 16)

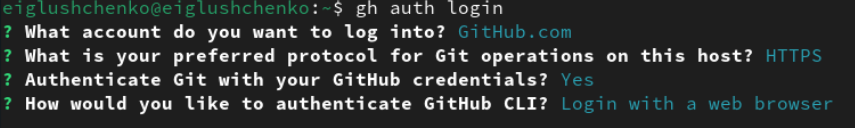


Рис. 16: Настройка gh

Прохожу авторизацию (рис fig. 17)

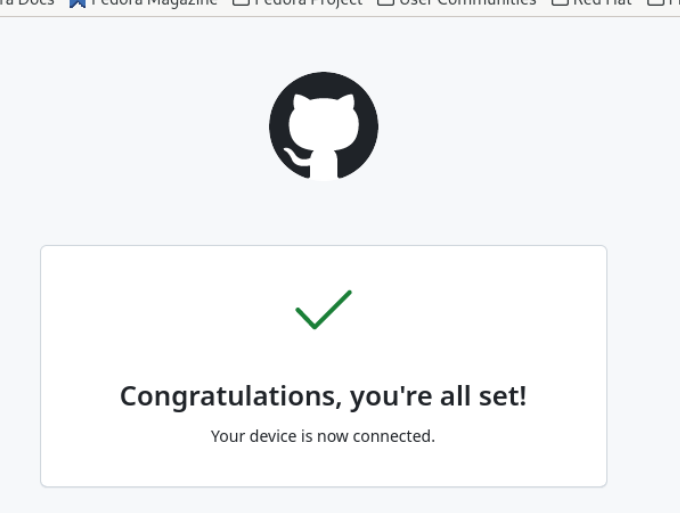


Рис. 17: Завершение авторизации в браузере

Завершаю авторизацию в терминале (рис fig. 18)

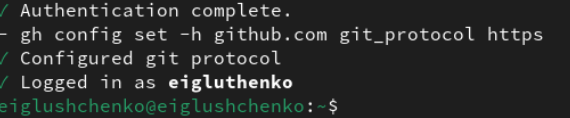


Рис. 18: Завершение авторизации

##Шаблон для рабочего пространства

Создаю рабочую папку, перехоху в нее, и создаю репозиторий на основе шаблона. (рис fig. 19)

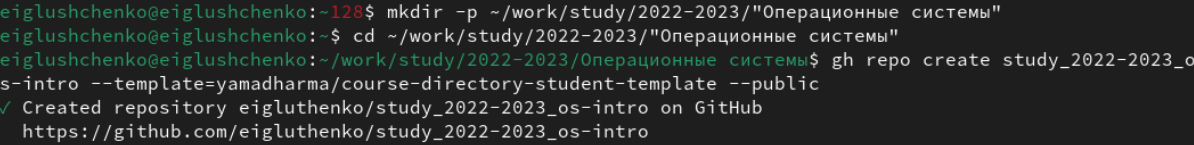


Рис. 19: Создание рабочего пространства

Клонирую репозиторий к себе в директорию (рис fig. 20)

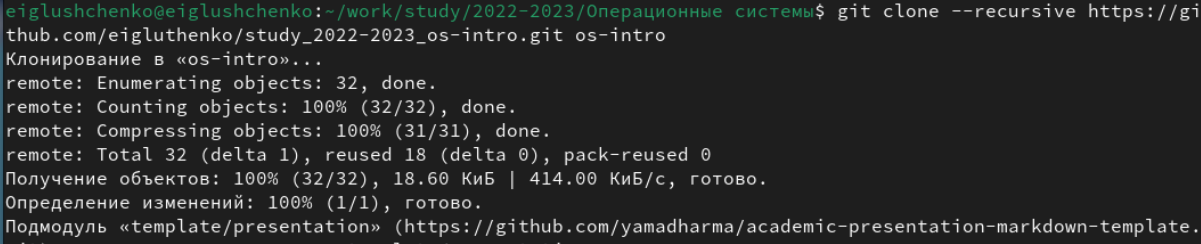


Рис. 20: Создание репозитория

Перехожу в католог курса и проверяю его содержание. (рис fig. 21)

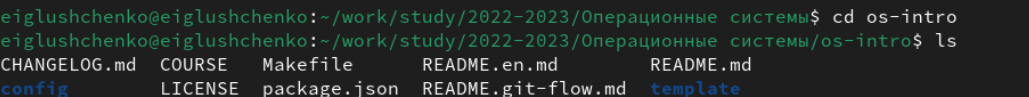


Рис. 21: Перемещение между дерикториями

Удаляю лишние файлы (рис fig. 22)

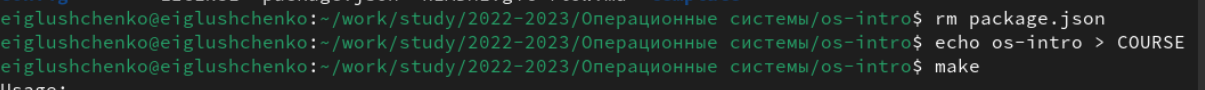


Рис. 22: Удаление

Отправка файлов на github (рис fig. 23)

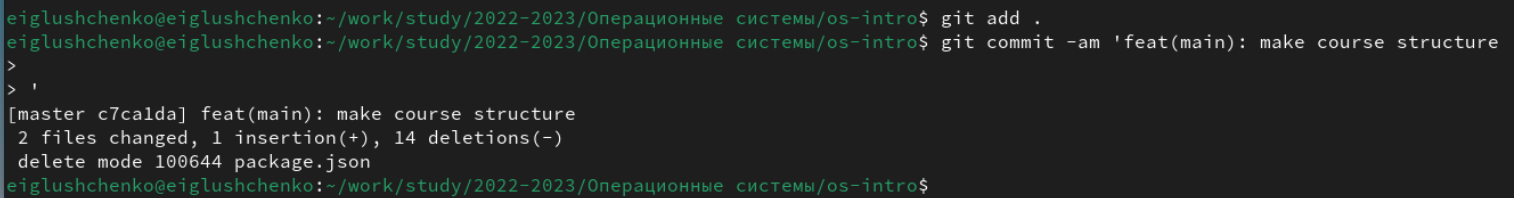


Рис. 23: Отправка файлов

# 4 Выводы

При выполнении данной работы я изучил основы работы с github.

# 5 Ответы на контрольные вопросы

1. Системы контроля версий (VCS) - программное обеспечение для облегчения работы с изменяющейся информацией. Они позволяют хранить несколько версий изменяющейся информации, одного и того же документа, может предоставить доступ к более ранним версиям документа. Используется для работы нескольких человек над проектом, позволяет посмотреть, кто и когда внес какое-либо изменение и т. д. VCS ррименяются для: Хранения понлой истории изменений, сохранения причин всех изменений, поиска причин изменений и совершивших изменение, совместной работы над проектами.
2. Хранилище – репозиторий, хранилище версий, в нем хранятся все доку менты, включая историю их изменения и прочей служебной информацией. commit – отслеживание изменений, сохраняет разницу в изменениях. Ис тория – хранит все изменения в проекте и позволяет при необходимости вернуться/обратиться к нужным данным. Рабочая копия – копия проекта, основанная на версии из хранилища, чаще всего последней версии.
3. Централизованные VCS (например: CVS, TFS, AccuRev) – одно основное хра нилище всего проекта. Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет, затем добавляет изменения обратно в хранилище. Децентрализованные VCS (например: Git, Bazaar) – у каждого пользователя свой вариант репозитория (возможно несколько вариантов), есть возможность добавлять и забирать изменения из любого 18 репозитория. В отличие от классических, в распределенных (децентралио ванных) системах контроля версий центральный репозиторий не является обязательным.
4. Сначала создается и подключается удаленный репозиторий, затем по мере изменения проекта эти изменения отправляются на сервер.
5. Участник проекта перед началом работы получает нужную ему версию проекта в хранилище, с помощью определенных команд, после внесения изменений пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются. К ним можно вернуться в любой момент.
6. Хранение информации о всех изменениях в вашем коде, обеспечение удоб ства командной работы над кодом.
7. Создание основного дерева репозитория: git init Получение обновлений (изменений) текущего дерева из центрального репози тория: git pull Отправка всех произведённых изменений локального дерева в центральный репозиторий: git push Просмотр списка изменённых файлов в текущей директории: git status Просмотр текущих изменений: git diff Сохранение текущих изменений: добавить все изменённые и/или созданные файлы и/или каталоги: git add . добавить конкретные изменённые и/или созданные файлы и/или каталоги: git add имена\_файлов удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): git rm имена\_файлов Сохранение добавленных изменений: сохранить все добавленные изменения и все изменённые файлы: git commit -am ‘Описание коммита’ сохранить добавленные изменения с внесением комментария через встроен ный редактор: git commit создание новой ветки, базирующейся на текущей: git checkout -b имя\_ветки переключение на некоторую ветку: git checkout имя\_ветки (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой) отправка изменений конкретной ветки в центральный репозиторий: git push origin имя\_ветки слияние ветки с текущим деревом: git merge –no-ff имя\_ветки Удаление ветки: удаление локальной уже слитой с основным деревом ветки: git branch -d имя\_ветки принудительное удаление локальной ветки: git branch -D имя\_ветки удаление ветки с центрального репозитория: git push origin :имя\_ветки
8. git push -all отправляем из локального репозитория все сохраненные из менения в центральный репозиторий, предварительно создав локальный репозиторий и сделав предварительную конфигурацию.
9. Ветвление - один из параллельных участков в одном хранилище, исходящих из одной версии, обычно есть главная ветка.Между ветками,т. е. их концами возможно их слияние. Используются для разработки новых функций.
10. Во время работы над проектом могут создаваться файлы, которые не следу ют добавлять в репозиторий. Например, временные файлы. Можно пропи сать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов.

# Список литературы

https://esystem.rudn.ru/mod/page/view.php?id=1098790