Содержание

# **лаборатория 2**

**Кусоро Майова Джеймс**

**НК-Абд-01-24** **1032244581**

**Теоретические сведения** **Системы контроля версий. Общие понятия Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта.** При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

**Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом**

**Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.**

**В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательныm.**

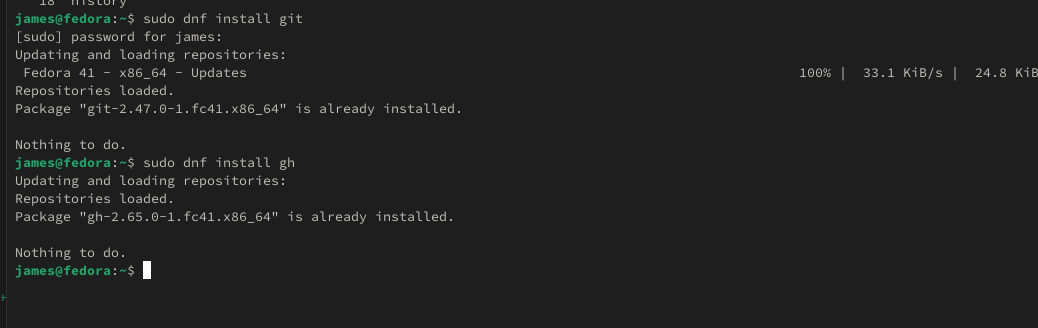
**Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.**

## цель работы

**Изучить идеологию и применение средств контроля версий.** **Освоить умения по работе с git.**

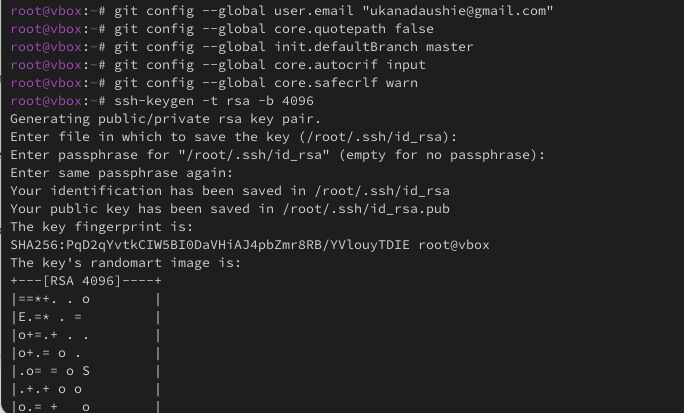
### **Описание задачи**

1.*во-первых, я установил git и gh, используя команды dnf install git и dnf install gh*



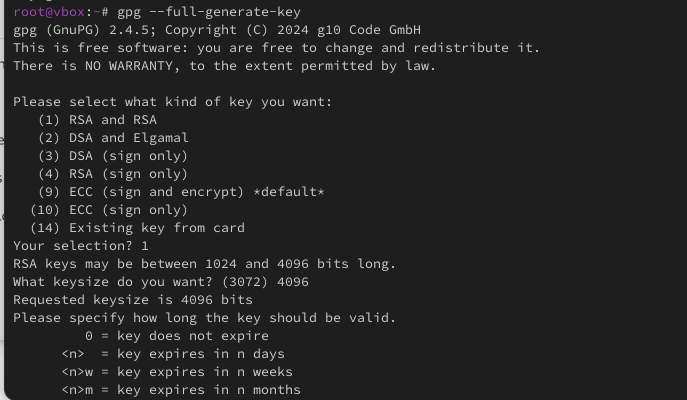
во-первых

2.*Я ввел свое имя и адрес электронной почты. Затем я настроил utf-8 в выходных данных git-сообщений. Я настроил проверку и подписание git-коммита.Я также создал ssh-ключ*



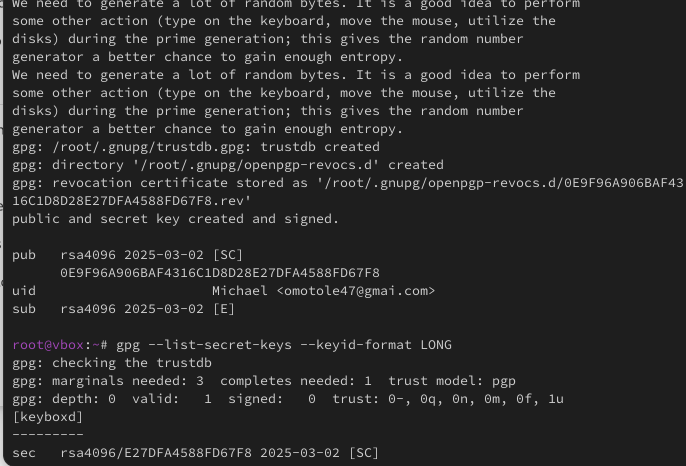
Screenshot 2

3.*Я сгенерировал ключ pgp, используя команду gpg –full-generate-key.*

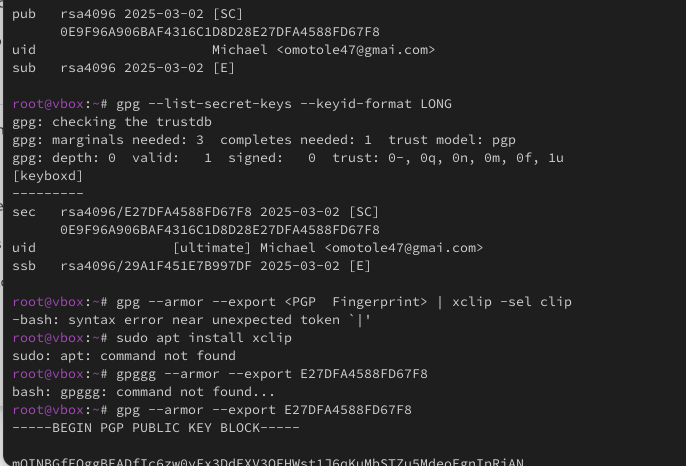


Screenshot 3

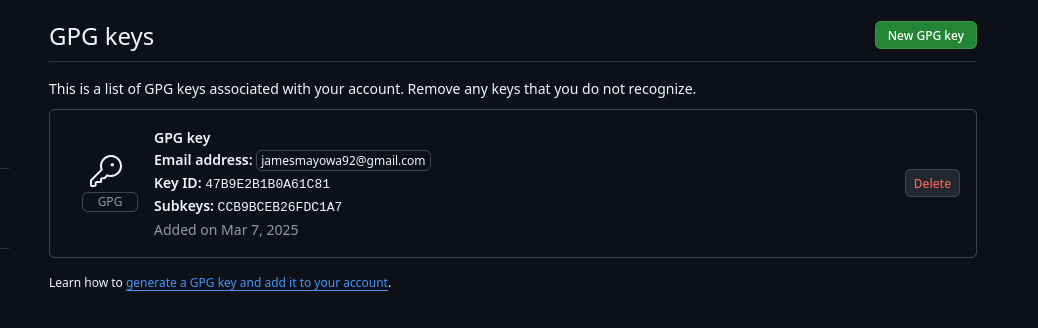
4.*Я добавил ключ pgp на github*



SCreenshot 4

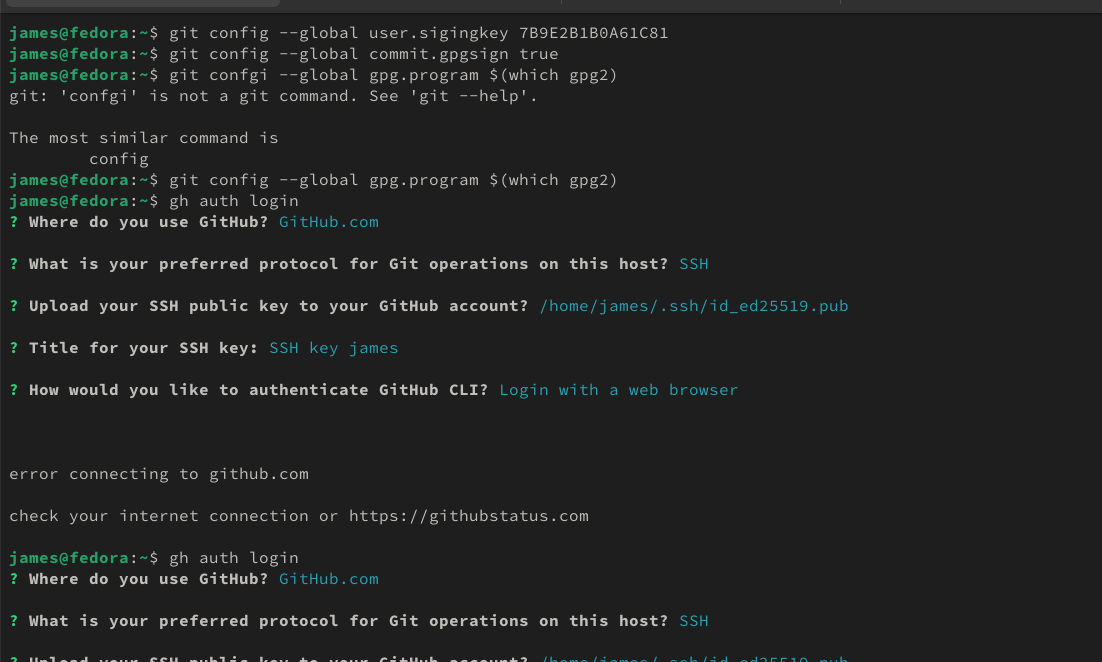


Screenshot 5



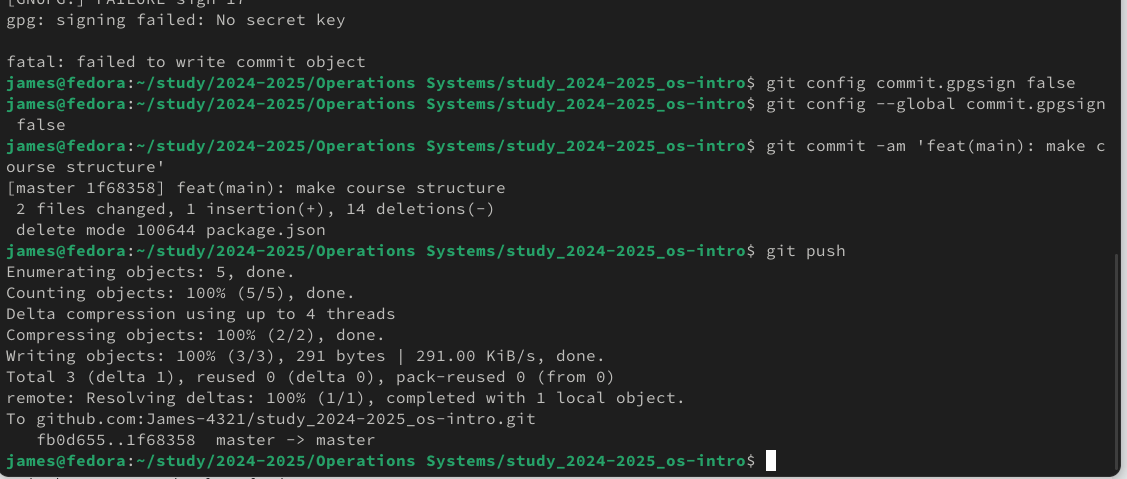
SCreenshot 6

5.*Настройка подписей автоматической фиксации git* git config –global user.signingkey git config –global commit.gpgsign true git config –global gpg.program $(which gpg2)



Screenshot 7

6.*Настройка каталога курсов*



Screenshot 8

**Выход:**В этой лабораторной работе мы проверили, как использовать git. Создаем репозитории и клонируем их. Как создать ssh- и pgn-ключи. А также как извлекать и отправлять в github

#### Ответы на контрольные вопросы

1.Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? *Системы контроля версий (VCS) отслеживают изменения в файлах, позволяя нескольким людям сотрудничать, вести историю и возвращаться к предыдущим версиям. Они решают такие проблемы, как случайная потеря данных, конфликты кода и отслеживание изменений.*

2.Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

*Хранилище: Центральное место, где хранятся все версии файлов.* *Фиксация: моментальный снимок изменений, сохраненных в репозитории.* *История: Запись коммитов, показывающая прошлые изменения.* *Рабочая копия: локальная версия файлов, изменяемых пользователем.*

3.Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

*Централизованная система контроля версий (CVCS): Все изменения хранятся в едином центральном репозитории. Пример: Subversion (SVN), Perforce.* *Децентрализованная система контроля версий (DVCS): У каждого пользователя есть полная копия репозитория, что позволяет работать в автономном режиме. Пример: Git, Mercurial.*

4.Действия VCS при работе с хранилищем в одиночку

*Инициализируйте репозиторий.* *Вносите изменения и фиксируйте их.* *Просмотрите историю и при необходимости вернитесь к ней.*

5.Работа с VCS Shared Storage

*Clone/pull the latest version.* *Make changes and commit them locally.* *Push changes to the shared repository.* *Resolve conflicts if necessary.*

6.Основные задачи, решаемые Git

*Отслеживание изменений* *Совместная работа и слияние* *Ветвление для параллельной разработки* *Откат версии*

7.Команды Git и их роли

*Developers: Write and commit code.* *Maintainers: Review and merge changes.* *Release Managers: Manage versions and releases.*

8.Использование локальных и удаленных репозиториев - Локальный репозиторий:

git init   
git add .   
git commit -m "Initial commit"

* Удаленный репозиторий:

git remote add origin <repo\_url>   
git push origin main   
git pull origin main

9.Ветви и их назначение: Ветки позволяют вести параллельную разработку, не затрагивая основную кодовую базу.

git branch feature-branch   
git checkout feature-branch

10.Игнорирование файлов в Git

node\_modules/   
\*.log

This prevents unnecessary files from being committed.