Лабораторная работа №2

Отчёт

Сергеев Д. О.

08 марта 2025

Российский университет дружбы народов, Москва, Россия

Информация

Докладчик

- Сергеев Даниил Олегович
- Студент
- Направление: Прикладная информатика
- Российский университет дружбы народов
- 1132246837@pfur.ru

Цель работы

Цель работы

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

Задание

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.

Выполнение лабораторной работы

Установка необходимого ПО (git, gh)

Установим git и gh.

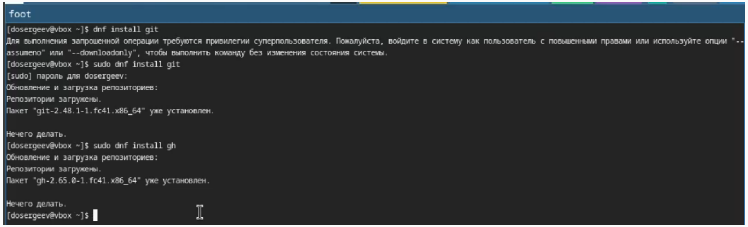

Рис. 1: Процесс установки git и gh.

Необходимые пакеты уже установлены, поэтому продолжим.

Базовая настройка git

Зададим имя и почту владельца репозитория, настроим utf-8 в выводе сообщений git, зададим имя начальной ветки master, параметр autocrlf и safecrlf.

```
[dosergeev@vbox ~]$ git config --global user.name "Daniel Sergeev"
[dosergeev@vbox ~]$ git config --global user.email "den.sergeev17@yandex.ru"
[dosergeev@vbox ~]$ git config --global core.quotepath false
[dosergeev@vbox ~]$ git config --global init.defaultBranch master
[dosergeev@vbox ~]$ git config --global core.autocrlf input
[dosergeev@vbox ~]$ git config --global core.safecrlf warn
[dosergeev@vbox ~]$
```

Рис. 2: Настройка параметров git.

Создание ключа ssh

Создадим ключ ssh по алгоритму ed25519.

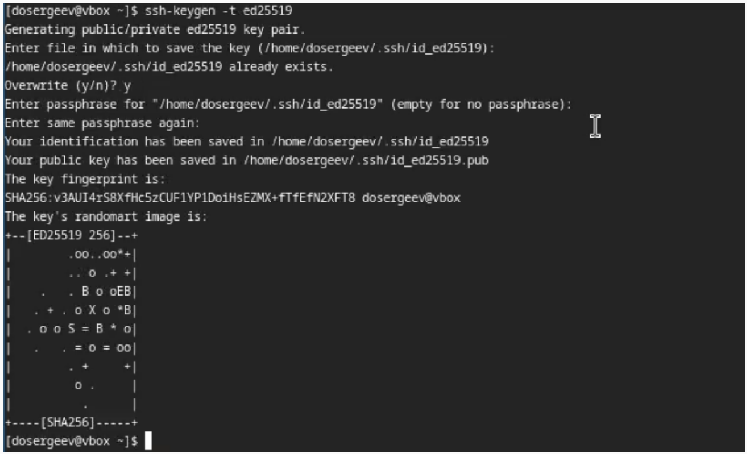

Рис. 3: Генерация ключа ssh.

Создание ключа ssh

Выведем ключ, скопируем и вставим в гит.

Рис. 4: Копирование ключа ssh.

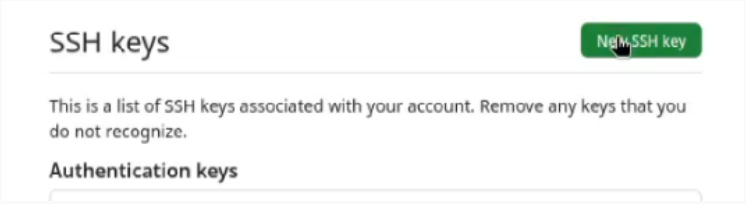

Рис. 5: Добавялем ключ ssh в гите.

Создание ключа ssh

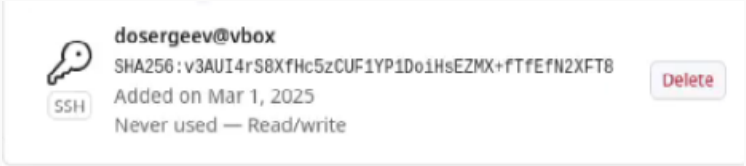

Рис. 6: Вид ключа ssh в гите.

Создание ключа gpg

сгенерируем ключ с помощью команды gpg –full-generate-key. В предложенных вариантах выберем: - тип RSA and RSA. - размер 4096. - срок действия по умолчанию. - имя. - адрес электронной почты. - без комментария.

Рис. 7: Генерация ключа gpg.

Выведем список ключей.

```
[dosergeev@vbox ~]$ gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: глубина: 0  достоверных:   2  подписанных:   0  доверие: 0-, 0q, 0n, 0m, 0f, 2u
[keyboxd]
---------
sec   rsa4096/05C705DF9D20ECD1 2025-03-01 [SC]
      32678AE5CFD81A6FDE0D7AA905C705DF9D20ECD1
uid                 [  абсолютно ] Daniel <den.sergeev17@yandex.ru>
ssb   rsa4096/F0C676A33E6CBC16 2025-03-01 [E]

sec   rsa4096/D26A655795F05626 2025-03-01 [SC]
      89D5B4E2E3851038DF030508D26A655795F05626
uid                 [  абсолютно ] Daniel <den.sergeev17@yandex.ru>
ssb   rsa4096/8A934455760D3D71 2025-03-01 [E]

[dosergeev@vbox ~]$
```

Рис. 8: Список ключей gpg.

Скопируем сгенерированный ключ в буфер обмена с помощью его отпечатка

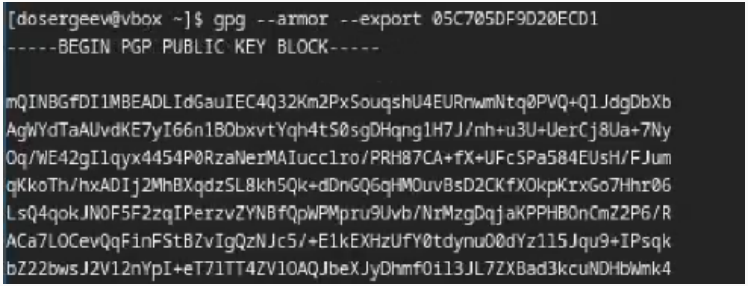

Рис. 9: Копирование ключа gpg.

Добавление ключа gpg в GitHub

Добавим ключ gpg в Github.

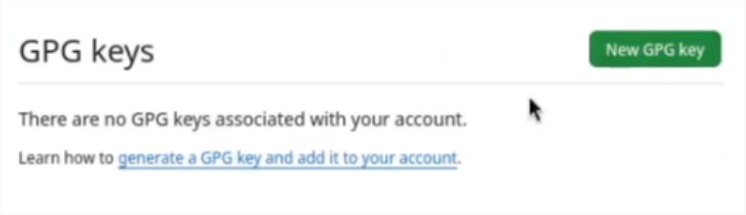

Рис. 10: Добавляем ключ gpg в гите.

Добавление ключа gpg в GitHub

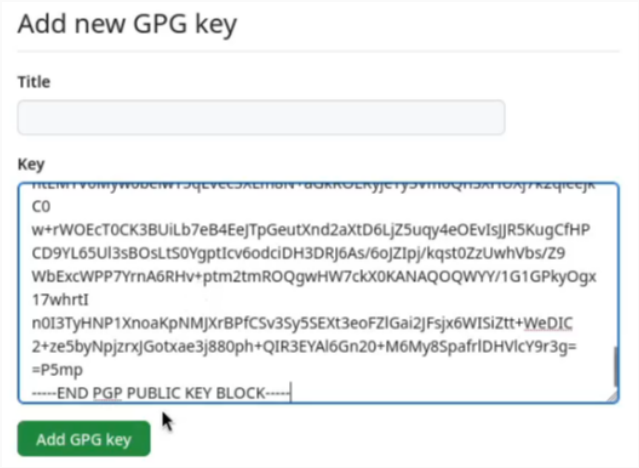

Рис. 11: Вводим в окно для ключа gpg.

Добавление ключа gpg в GitHub

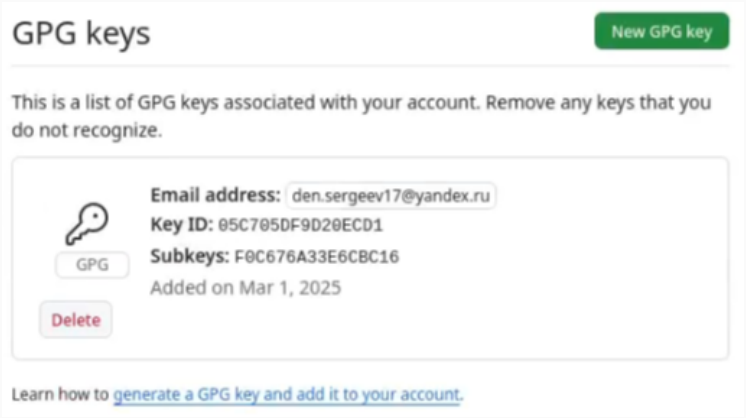

Рис. 12: Вид ключа gpg в гите.

Настройка автоматических подписей коммитов git

Указываем git-у применять адрес почты при подписи коммитов.

```
[dosergeev@vbox ~]$ git config --global user.signingkey 05C705DF9D20EC
D1
[dosergeev@vbox ~]$ git config --global commit.gpgsign true
[dosergeev@vbox ~]$ git config --global gpg.program $(which gpg2)
[dosergeev@vbox ~]$
```

Рис. 13: Настройка подписей коммитов git.

Авторизуемся через браузер.

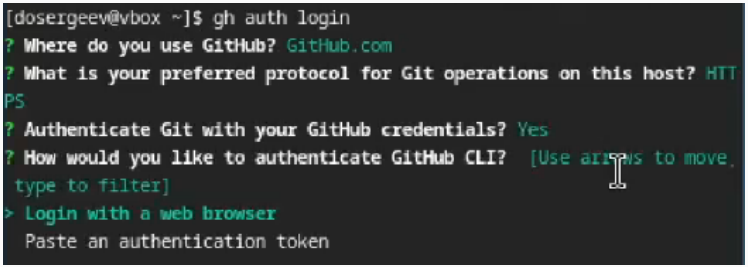

Рис. 14: Выбираем вход через браузер.

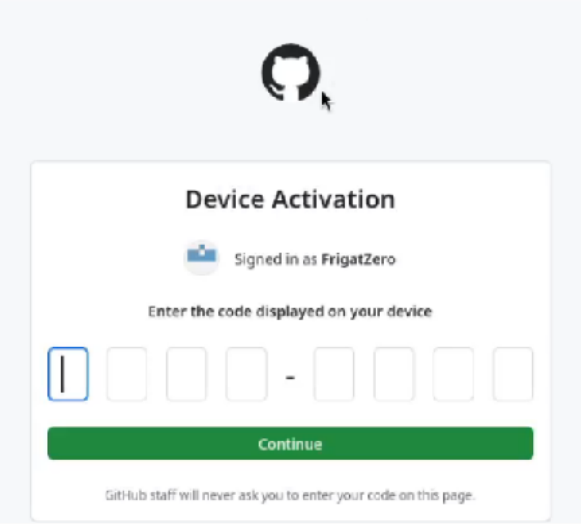

Рис. 15: Вводим ключ.

Создание репозитория курса на GitHub

Создадим репозиторий гит и каталог курса.

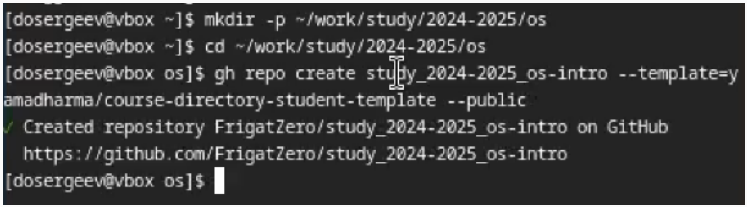

Рис. 16: Создание репозитория курса.

Создание репозитория курса на GitHub

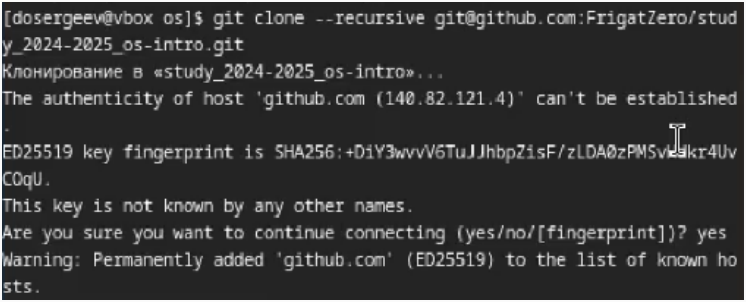

Рис. 17: Клонирование удалённого репозитория в локальный.

Перейдем в каталог курса и удалим лишние файлы. Создадим необходимые каталоги с помощью make prepare.

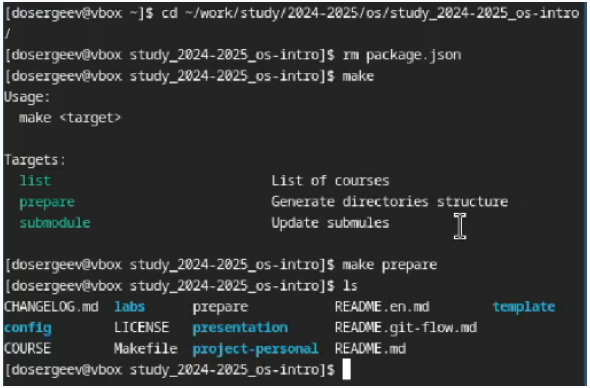

Рис. 18: Удаление лишних файлов и создание необходимых каталогов make.

Отправим файлы на сервер.

```
[dosergeev@vbox study_2024-2025_os-intro]$ git add .
[dosergeev@vbox study_2024-2025_os-intro]$ git commit -am 'feat(main): ma
ke course stucture'
```

Рис. 19: Сохраняем изменения.

```
Перечисление объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 342.32 КиБ | 1.47 МиБ/с, готово.
Total 38 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:FrigatZero/study_2024-2025_os-intro.git
   b52ba75..1642e1f  master -> master
[dosergeev@vbox study_2024-2025_os-intro]$
```

Рис. 20: Отправляем файлы на сервер.

Ответы на контрольные вопросы

Ответы на контрольные вопросы

1.Системы контроля версий - это инструменты, позволяющие организовать работу над проектом разработки, выкладывать его обновления и контролировать релизы и изменения кода. Они предназначаются для отслеживания изменений, защиты исходного кода от удаления и изменения, возможности отката изменений и командной работы для 10 и более человек.

2.

- Хранилище - репозиторий в котором храняться файлы проекта в различных версиях.
- Commit - комментарий внесённых изменений в репозитории.
- История - история изменений файлов проекта.
- Рабочая копия - копия, созданная из определенной версии репозитория, которую модифицирует разработчик.

3.Централизованные системы контроля версий имеют единый сервер под хранение проекта, для изменения которых необходимо скачать необходимые файлы, изменить и вернуть обратно на сервер. Пример центроализованной VCS: Subversion. Децентрализованные системы полностью копируют удалённый репозиторий в локальный. При этом внесенные изменения отправляются на сервер в качестве новой версии. Пример: git.

4.Создается репозиторий для работы с проектом, при необходимости файлы обновляются локально и отправляются на сервер в качестве новой версии.

5.Репозиторий копируется локально. После внесения изменений файлы загружаются на сервер в качестве отдельной версии. После этого измененная ветка может быть объединена с текущей и отправлена в релиз.

Ответы на контрольные вопросы

6.Хранение файлов проекта, отслеживание версий, защита от изменений, работа в команде.

7.

- git clone – клонирование проекта с сервера в указанный локальный репозиторий.
- git add – добавляет все изменённые или созданные файлы или каталоги.
- git commit – сохраняет изменения репозитория с комментарием.
- git push – загружает добавленные изменения на сервер
- git pull – получить последние изменения с сервера
- git rm – удаляет файл из индекса репозитория
- git status – просмотривает список измененных файлов в текущем репозитории

8.С локальным: git add . , git commit -am commit, git push - добавление всех измененнных файлов в текущем каталоге, их сохранение с комментарием commit, git pull - получаем последние изменения с удалённого репозитория. С удалённым: git push - отправляем сохраненные изменения на сервер.

Ответы на контрольные вопросы

9.Ветки - это различные версии исходного репозитория, являющиеся копиями с внесенными изменениями. Нужны для параллельной работы над проектом. Могут быть объединенны для внесения изменений. Их можно проигнорировать, добавив имя в .gitignore.

Ответы на контрольные вопросы

10.Мы можем игнорировать некоторые файлы при коммитах, когда не хотим добавлять их в удаленный репозиторий для чистоты или упрощения.

Вывод

Вывод

В результате выполнения лабораторной работы я изучил как применять средства контроля версий для работы с удалённым репозиторием и освоил умения по работе с git.