Отчёт по лабораторной работе №2

Управление версиями

Швед Карина Дмитриевна

Содержание

Список иллюстраций

Список таблиц

Цель работы

Целью данной работы является изучение идеологии и применения средств контроля версий и освоение умений работать с git.

Выполнение лабораторной работы

Установка git, установка gh

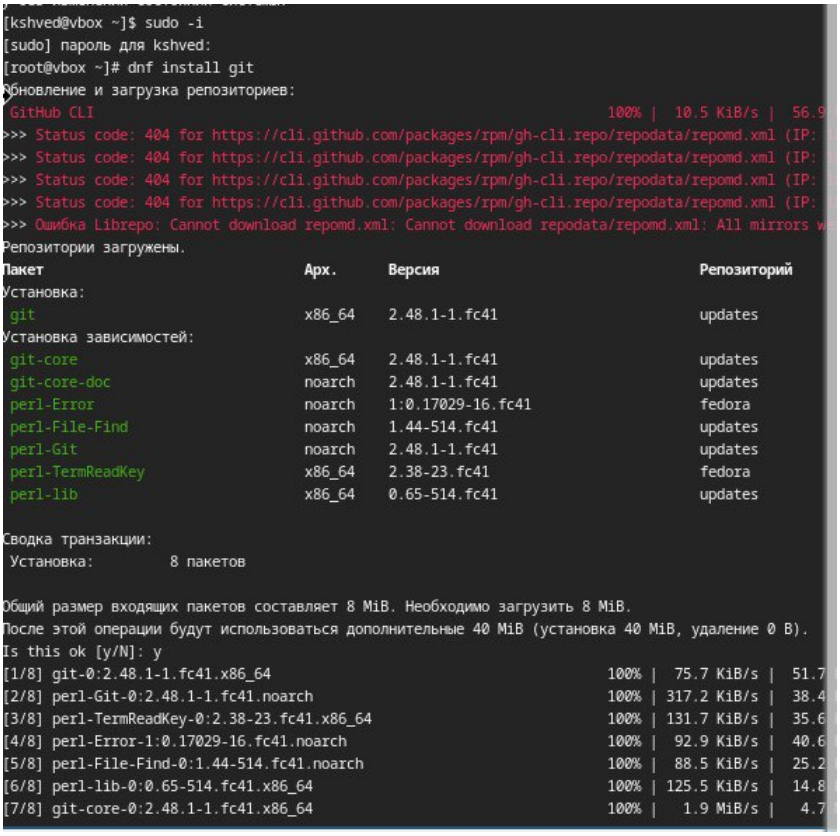


Рис. 1: dnf install git

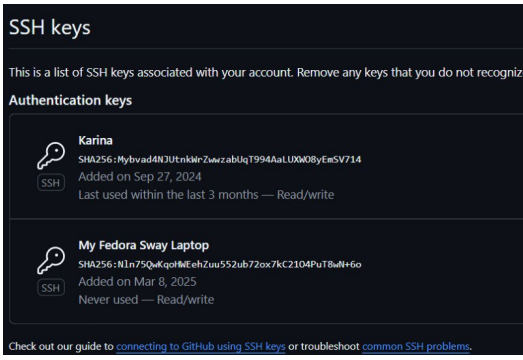
Далее я задаю имя и email владельца репозитория,utf-8 в выводе сообщений git:

```
[ 9/10] Установка perl-Git-0:2.48.1-1.fc41.noarch              100% |   2.5 MiB/s |  65.0 KiB |  0
[10/10] Установка git-0:2.48.1-1.fc41.x86_64                   100% |  54.3 KiB/s |  87.5 KiB |  0
Завершено!
[root@vbox ~]# git config --global user.name "Karina Shved"
[root@vbox ~]# git config --global user.email "karina.shveddd@gmail.com"
[root@vbox ~]#
```

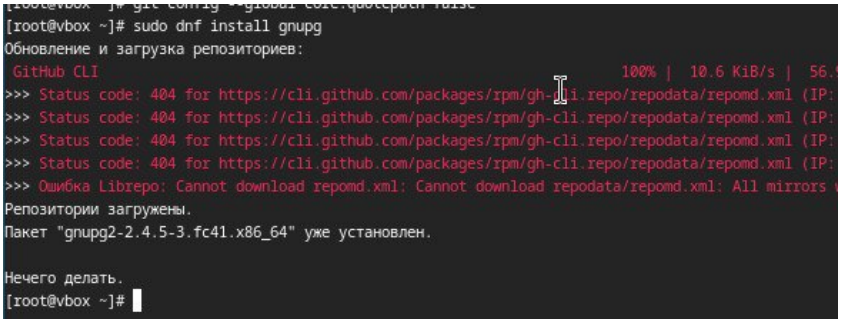
Рис. 2: git config –global user.

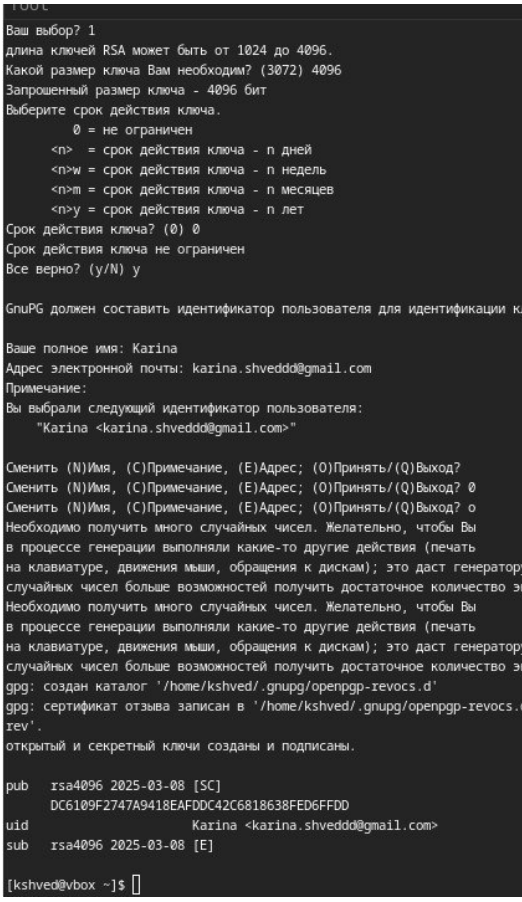
Настраиваю верификацию и подписание коммитов git. Создание ssh ключа и добавление его на github

```
[root@vbox ~]# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase for "/root/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:KAxYGE9DWF6MOCZSACm9zhroG4qSTVoqI5ZZs1NpBgM root@vbox
The key's randomart image is:
+---[RSA 4096]----+
|=%Bo.            |
|EBoo.            |
|B.+.             |
| o.o   .         |
|.oo + . S        |
|o.=* .           |
|.@X              |
|%Bo              |
|Boo              |
+----[SHA256]-----+
[root@vbox ~]#
```

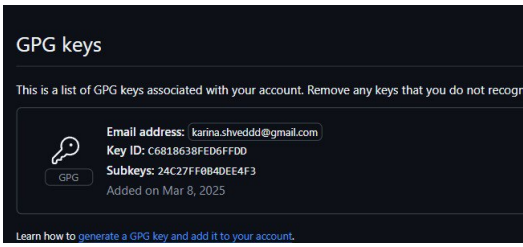


Создание ключа pgp Сначала я ввожу dnf insstall gnupg. Эта команда устанавливает GnuPG (GPG) — инструмент для шифрования, подписи данных и управления криптографическими ключами.Затем генерирую ключ





Далее с помощью консоли, собственного токена и API я добавляю gpg ключ к себе на гитхаб



Настройка автоматических подписей коммитов

```
kshved@vbox ~]$ ^C
kshved@vbox ~]$ git config --global user.signingkey C6818638FED6FFDD
kshved@vbox ~]$ git config --global commit.gpgsign true
kshved@vbox ~]$ git commit -S -m "My signed commit"
atal: не найден git репозиторий (или один из его каталогов вплоть до точки монтирования /)
станавливаю поиск на границе файловой системы (так как GIT_DISCOVERY_ACROSS_FILESYSTEM не установле
kshved@vbox ~]$ gpg --list-keys --keyid-format LONG
keyboxd]
--------
ub   rsa4096/C6818638FED6FFDD 2025-03-08 [SC]
     DC6109F2747A9418EAFDDC42C6818638FED6FFDD
id              [  абсолютно ] Karina <karina.shveddd@gmail.com>
ub   rsa4096/24C27FF0B4DEE4F3 2025-03-08 [E]

kshved@vbox ~]$ git config --global user.signingkey
6818638FED6FFDD
kshved@vbox ~]$ git config --global user.signingkey DC6109F2747A9418EAFDDC42C6818638FED6FFDD
kshved@vbox ~]$ git config --global commit.gpgsign true
kshved@vbox ~]$ git config --global gpg.program $(which gpg)
kshved@vbox ~]$ git config --global --list
redential.https://github.com.helper=
redential.https://github.com.helper=!/usr/local/bin/gh auth git-credential
redential.https://gist.github.com.helper=
redential.https://gist.github.com.helper=!/usr/local/bin/gh auth git-credential
ser.name=Karina Shved
ser.email=karina.shveddd@gmail.com
ser.signingkey=DC6109F2747A9418EAFDDC42C6818638FED6FFDD
ommit.gpgsign=true
pg.program=/usr/bin/gpg
```

Настройка gh и создание рабочего пространства на основе шаблона

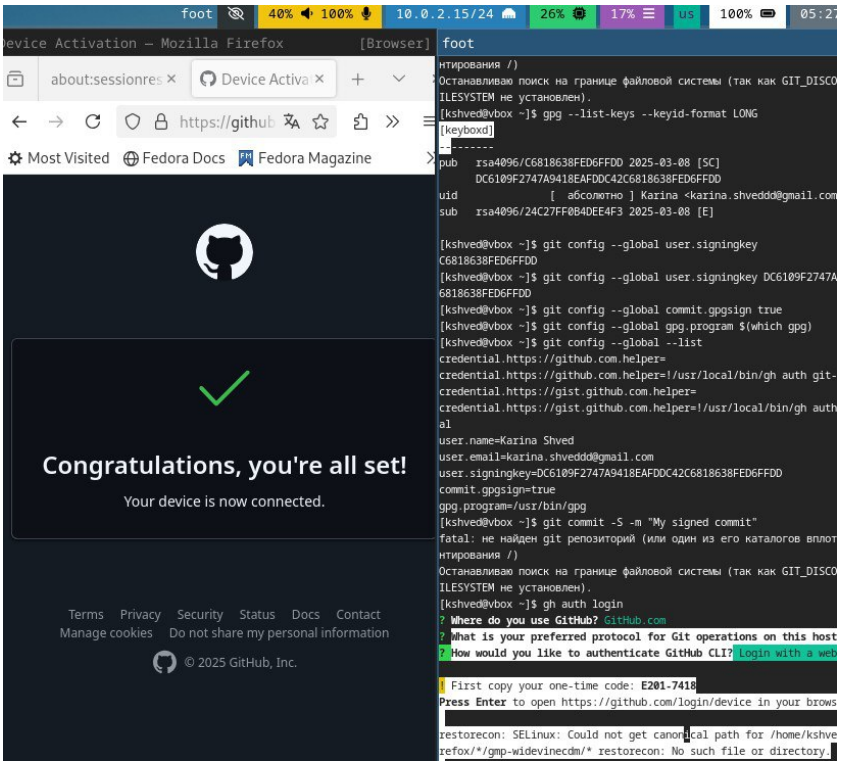


Рис. 3: подключение своей учетной записи gitgub

```
Hi karinashved! You've successfully authenticated, but GitHub does not provid
[kshved@vbox Операционные системы]$ git clone --recursive git@github.com:kari
s-intro
Клонирование в «os-intro»...
remote: Enumerating objects: 36, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 36 (delta 1), reused 21 (delta 0), pack-reused 0 (from 0)
Получение объектов: 100% (36/36), 19.37 КиБ | 9.69 МиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-pre
истрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laborator
 по пути «template/report»
Клонирование в «/home/kshved/work/study/2024-2025/Операционные системы/os-int
remote: Enumerating objects: 111, done.
remote: Counting objects: 100% (111/111), done.
remote: Compressing objects: 100% (77/77), done.
remote: Total 111 (delta 42), reused 100 (delta 31), pack-reused 0 (from 0)
Получение объектов: 100% (111/111), 102.17 КиБ | 649.00 КиБ/с, готово.
Определение изменений: 100% (42/42), готово.
Клонирование в «/home/kshved/work/study/2024-2025/Операционные системы/os-int
remote: Enumerating objects: 142, done.
remote: Counting objects: 100% (142/142), done.
remote: Compressing objects: 100% (97/97), done.
remote: Total 142 (delta 60), reused 121 (delta 39), pack-reused 0 (from 0)
Получение объектов: 100% (142/142), 341.09 КиБ | 1.17 МиБ/с, готово.
Определение изменений: 100% (60/60), готово.
Submodule path 'template/presentation': checked out 'c9b2712b4b2d431ad5086c9c
Submodule path 'template/report': checked out 'c26e22effe7b3e0495707d82ef561a
```

Создание шаблона рабочего пространства

```
талога
[kshved@vbox Операционные системы]$ cd ~/work/study/2024-2025/"Операционные системы"/os-intro
[kshved@vbox os-intro]$ rm package.json
[kshved@vbox os-intro]$ echo os-intro > COURSE
[kshved@vbox os-intro]$ make
Usage:
 make <target>

Targets:
 list                        List of courses
 prepare                     Generate directories structure
 submodule                   Update submules

[kshved@vbox os-intro]$ git add .
[kshved@vbox os-intro]$ git commit -am 'feat(main): make course structure'
[master d80164d] feat(main): make course structure
 2 files changed, 1 insertion(+), 14 deletions(-)
 delete mode 100644 package.json
[kshved@vbox os-intro]$ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 3 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 952 байта | 952.00 КиБ/с, готово.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:karinashved/study_2024-2025_os-intro.git
   244e136..d80164d  master -> master
```

Настройка каталога курса

Далее отправляю файлы на сервер

Выводы

Мы приобрели практические навыки работы с сервисом github.

Контрольные вопросы:

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

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

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

- хранилище - пространство на накопителе где расположен репозиторий
- commit - сохранение состояния хранилища
- история - список изменений хранилища (коммитов)
- рабочая копия - локальная копия сетевого репозитория, в которой работает программист. Текущее состояние файлов проекта, основанное на версии, загруженной из хранилища (обычно на последней)

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

Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере. Доступ к нему осуществлялся через специальное клиентское приложение. В качестве примеров таких программных продуктов можно привести CVS, Subversion.

Распределенные системы контроля версий (Distributed Version Control System, DVCS) позволяют хранить репозиторий (его копию) у каждого разработчика, работающего с данной системой. При этом можно выделить центральный репозиторий (условно), в который будут отправляться изменения из локальных и, с ним же эти локальные репозитории будут синхронизироваться. При работе с такой системой, пользователи периодически синхронизируют свои локальные репозитории с центральным и работают непосредственно со своей локальной копией. После внесения достаточного количества изменений в локальную копию они (изменения) отправляются на сервер. При этом сервер, чаще всего, выбирается условно, т.к. в большинстве DVCS нет такого понятия как “выделенный сервер с центральным репозиторием”.

4. Опишите действия с VCS при единоличной работе с хранилищем.

Один пользователь работает над проектом и по мере необходимости делает коммиты, сохраняя определенные этапы.

5. Опишите порядок работы с общим хранилищем VCS.

Несколько пользователей работают каждый над своей частью проекта. При этом каждый должен работать в своей ветки. При завершении работы ветка пользователя сливается с основной веткой проекта.

6. Каковы основные задачи, решаемые инструментальным средством git?

- Ведение истории версий проекта: журнал (log), метки (tags), ветвления (branches).
- Работа с изменениями: выявление (diff), слияние (patch, merge).
- Обеспечение совместной работы: получение версии с сервера, загрузка обновлений на сервер.

7. Назовите и дайте краткую характеристику командам git.

- git config - установка параметров
- git status - полный список изменений файлов, ожидающих коммита
- git add . - сделать все измененные файлы готовыми для коммита.
- git commit -m “[descriptive message]” - записать изменения с заданным сообщением.
- git branch - список всех локальных веток в текущей директории.
- git checkout [branch-name] - переключиться на указанную ветку и обновить рабочую директорию.
- git merge [branch] — соединить изменения в текущей ветке с изменениями из заданной.
- git push - запушить текущую ветку в удаленную ветку.
- git pull - загрузить историю и изменения удаленной ветки и произвести слияние с текущей веткой.

8. Приведите примеры использования при работе с локальным и удалённым репозиториями.

- git remote add [имя] [url] — добавляет удалённый репозиторий с заданным именем;
- git remote remove [имя] — удаляет удалённый репозиторий с заданным именем;
- git remote rename [старое имя] [новое имя] — переименовывает удалённый репозиторий;

- git remote set-url [имя] [url] — присваивает репозиторию с именем новый адрес;
- git remote show [имя] — показывает информацию о репозитории.

9. Что такое и зачем могут быть нужны ветви (branches)?

Ветвление — это возможность работать над разными версиями проекта: вместо одного списка с упорядоченными коммитами история будет расходиться в определённых точках. Каждая ветвь содержит легковесный указатель HEAD на последний коммит, что позволяет без лишних затрат создать много веток. Ветка по умолчанию называется master, но лучше назвать её в соответствии с разрабатываемой в ней функциональностью.

10. Как и зачем можно игнорировать некоторые файлы при commit? Зачем:

- Исключение временных и ненужных файлов (*.log,* .tmp).
- Защита конфиденциальных данных (config.json, .env).
- Исключение средовых файлов (node_modules/, venv/).
- Уменьшение размера репозитория.

Как:

- Создать .gitignore: touch .gitignore
- Добавить в него правила, например: *.log node_modules/ .env

*Закоммитить .gitignore: git add .gitignore git commit -m “Добавлен .gitignore”