

# **Лабораторная работа №1**

**Научное программирование**

Леонтьева Ксения Андреевна | НПИМд-02-23

# Содержание

1	Цель работы	4
2	Теоретическое введение	5
3	Выполнение лабораторной работы	6
4	Контрольные вопросы	16
5	Вывод	21
	Список литературы	22

## Список иллюстраций

3.1	Учетная запись на GitHub . . . . .	6
3.2	Установка git-flow . . . . .	7
3.3	Базовая настройка git и создание ключа ssh по алгоритму rsa . . .	8
3.4	Создание ключа ssh по алгоритму ed25519 . . . . .	8
3.5	Генерация ключа pgr . . . . .	9
3.6	Генерация ключа pgr . . . . .	9
3.7	Вывод списка ключей . . . . .	10
3.8	Копирование сгенерированного ключа PGP . . . . .	10
3.9	Добавление gpg-ключа на GitHub . . . . .	11
3.10	Настройка автоматических подписей коммитов git . . . . .	11
3.11	Установка gh . . . . .	12
3.12	Установка gh . . . . .	13
3.13	Настройка gh и авторизация . . . . .	13
3.14	Создание репозитория курса . . . . .	14
3.15	Генерация ssh-ключа . . . . .	14
3.16	Добавление ssh-ключа на GitHub и завершение создания репозитория курса . . . . .	15
3.17	Настройка каталога курса . . . . .	15
3.18	Настройка каталога курса . . . . .	15

# 1 Цель работы

Изучить идеологию и применение средств контроля версий, а также освоить умения по работе с git.

## 2 Теоретическое введение

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

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.

Более подробно см. в [1]

### 3 Выполнение лабораторной работы

Для начала была создана учетная запись и заполнены основные данные на сайте <https://github.com> (рис. fig. 3.1).

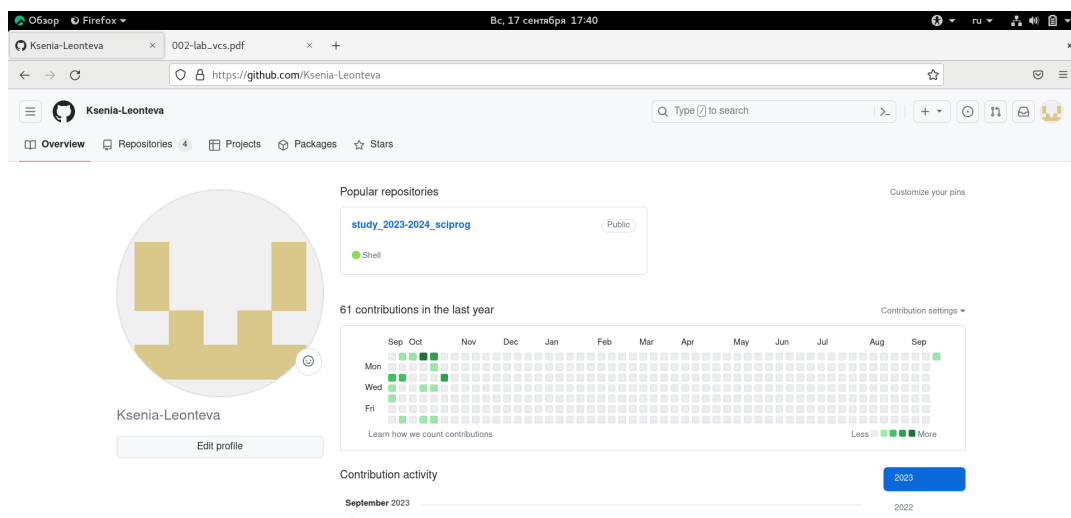


Рис. 3.1: Учетная запись на GitHub

Вручную установили программное обеспечение git-flow (рис. fig. 3.2).

```
[kaleontjeval@kaleontjeva raw.github.com]$ cd /tmp
[kaleontjeval@kaleontjeva tmp]$ wget --no-check-certificate -q https://raw.github.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh
[kaleontjeval@kaleontjeva tmp]$ chmod +x gitflow-installer.sh
[kaleontjeval@kaleontjeva tmp]$ sudo ./gitflow-installer.sh install stable
## git-flow no-make installer ##
Installing git-flow to /usr/local/bin
Cloning repo from GitHub to gitflow
Клонирование в «gitflow»...
remote: Enumerating objects: 4270, done.
remote: Total 4270 (delta 0), reused 0 (delta 0), pack-reused 4270
Получение объектов: 100% (4270/4270), 1.74 Мис | 675.00 Кис/с, готово.
Определение изменений: 100% (2533/2533), готово.
Уже обновлено.
Ветка «master» отслеживает внешнюю ветку «master» из «origin».
Переключено на новую ветку «master»
install: создание каталога '/usr/local/share/doc'
install: создание каталога '/usr/local/share/doc/gitflow'
install: создание каталога '/usr/local/share/doc/gitflow/hooks'
'gitflow/git-flow' -> '/usr/local/bin/git-flow'
'gitflow/git-flow-init' -> '/usr/local/bin/git-flow-init'
'gitflow/git-flow-feature' -> '/usr/local/bin/git-flow-feature'
'gitflow/git-flow-bugfix' -> '/usr/local/bin/git-flow-bugfix'
'gitflow/git-flow-hotfix' -> '/usr/local/bin/git-flow-hotfix'
'gitflow/git-flow-release' -> '/usr/local/bin/git-flow-release'
'gitflow/git-flow-support' -> '/usr/local/bin/git-flow-support'
'gitflow/git-flow-version' -> '/usr/local/bin/git-flow-version'
'gitflow/gitflow-common' -> '/usr/local/bin/gitflow-common'
'gitflow/gitflow-shFlags' -> '/usr/local/bin/gitflow-shFlags'
'gitflow/git-flow-config' -> '/usr/local/bin/git-flow-config'
'gitflow/hooks/filter-flow-hotfix-finish-tag-message' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-hotfix-finish-tag-message'
'gitflow/hooks/filter-flow-hotfix-start-version' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-hotfix-start-version'
'gitflow/hooks/filter-flow-release-branch-tag-message' -> '/usr/local/share/doc/gitflow/hooks/filter-flow-release-branch-tag-message'
```

Рис. 3.2: Установка git-flow

Осуществили базовую настройку git (рис. fig. 3.3), а именно:

- задали имя и email владельца репозитория,
- настроили utf-8 в выводе сообщений git,
- настроили верификацию и подписание коммитов git,
- задали имя начальной ветки (master),
- параметр autocrlf,
- параметр safecrlf.

Также создали ключи ssh (рис. fig. 3.3 и рис. fig. 3.4):

- по алгоритму rsa с ключом размером 4096 бит,
- по алгоритму ed25519.

```

[kaleontjeval@kaleontjeva ~]$ git config --global user.name "Kseniia Leonteva"
[kaleontjeval@kaleontjeva ~]$ git config --global user.email "ksuleo23@gmail.com"
[kaleontjeval@kaleontjeva ~]$ git config --global core.quotepath false
[kaleontjeval@kaleontjeva ~]$ git config --global init.defaultBranch master
[kaleontjeval@kaleontjeva ~]$ git config --global core.autocrlf input
[kaleontjeval@kaleontjeva ~]$ git config --global core.safecrlf warn
[kaleontjeval@kaleontjeva ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kaleontjeval/.ssh/id_rsa): /home/kaleontjeval/.ssh/id_rsa
/home/kaleontjeval/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kaleontjeval/.ssh/id_rsa.
Your public key has been saved in /home/kaleontjeval/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:YiGPwd7T7zdgbrIl7LpYh3Bnvt0XcegjjWs9Jyc6JQ kaleontjeval@kaleontjeva
The key's randomart image is:
+---[RSA 4096]-----+
|
| ..
| . =0. . . o
| o +. +oS = . o
| o o++ % * .
| ..oE.X .
| oo+0o+ . .
| . o**0..
+---[SHA256]-----+

```

Рис. 3.3: Базовая настройка git и создание ключа ssh по алгоритму rsa

```

[kaleontjeval@kaleontjeva ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/kaleontjeval/.ssh/id_ed25519): /home/kaleontjeval/.ssh/id_ed25519
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kaleontjeval/.ssh/id_ed25519.
Your public key has been saved in /home/kaleontjeval/.ssh/id_ed25519.pub.
The key fingerprint is:
SHA256:mkuuLS6ZHSCL0I+TyfvLUQGSydKBCorYcA2yBN8aI kaleontjeval@kaleontjeva
The key's randomart image is:
+--[ED25519 256]--+
|*o=o + .
|=XE.o =
|0+=. . o
|=*o . o
|* . o .S
|.o.o .o.
|.+++ +o
| =..+.
| o+++
+---[SHA256]-----+
[kaleontjeval@kaleontjeva ~]$

```

Рис. 3.4: Создание ключа ssh по алгоритму ed25519

Сгенерировали ключ pgr и выбрали из предложенных опций необходимые (рис. fig. 3.5 и рис. fig. 3.6).



```
Обзор Терминал Вc, 17 сентября 15:08
kaleontjeva1@kaleontjeva:~
Файл Правка Вид Поиск Терминал Справка
[kaleontjeva1@kaleontjeva ~]$ gpg --full-generate-key
gpg (GnuPG) 2.2.20; Copyright (C) 2020 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Выберите тип ключа:
  (1) RSA и RSA (по умолчанию)
  (2) DSA и Elgamal
  (3) DSA (только для подписи)
  (4) RSA (только для подписи)
  (14) Existing key from card
Ваш выбор? 1
длина ключей RSA может быть от 1024 до 4096.
Какой размер ключа Вам необходим? (2048) 4096
Запрошенный размер ключа - 4096 бит
Выберите срок действия ключа.
  0 = не ограничен
  <n> = срок действия ключа - n дней
  <n>w = срок действия ключа - n недель
  <n>m = срок действия ключа - n месяцев
  <n>y = срок действия ключа - n лет
Срок действия ключа? (0) 0
Срок действия ключа не ограничен
Все верно? (y/N) y

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: Kseniia Leonteva
Адрес электронной почты: ksuleo23@gmail.com
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Kseniia Leonteva <ksuleo23@gmail.com>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? 0
Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? 0
```

Рис. 3.5: Генерация ключа gpg

```
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
gpg: /home/kaleontjeva1/.gnupg/trustdb.gpg: создана таблица доверия
gpg: ключ 5CA14D668A9372F5 помечен как абсолютно доверенный
gpg: создан каталог '/home/kaleontjeva1/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/kaleontjeva1/.gnupg/openpgp-revocs.d/08799DC530BA2F505BB51F6E5CA14D668A9372F5.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2023-09-17 [SC]
      08799DC530BA2F505BB51F6E5CA14D668A9372F5
uid    Kseniia Leonteva <ksuleo23@gmail.com>
sub    rsa4096 2023-09-17 [E]

[kaleontjeva1@kaleontjeva ~]$
```

Рис. 3.6: Генерация ключа gpg

Выводим список ключей и копируем отпечаток приватного ключа (рис. fig. 3.7).

```
[kaleontjeval@kaleontjeva ~]$ 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: 1 подписанных: 0 доверие: 0-, 0q, 0n, 0m, 0f, 1u
/home/kaleontjeval/.gnupg/pubring.kbx
-----
sec   rsa4096/5CA14D668A9372F5 2023-09-17 [SC]
      08799DC530BA2F505BB51F6E5CA14D668A9372F5
uid           [ абсолютно ] Ksenia Leonteva <ksuleo23@gmail.com>
ssb   rsa4096/377758E703B8BD1C 2023-09-17 [E]
```

Рис. 3.7: Вывод списка ключей

Копируем сгенерированный PGP ключ в буфер обмена (рис. fig. 3.8).

```
[kaleontjeval@kaleontjeva ~]$ gpg --output public.pgp --armor --export ksuleo23@gmail.com
[kaleontjeval@kaleontjeva ~]$ cat prv.key
cat: prv.key: Нет такого файла или каталога
[kaleontjeval@kaleontjeva ~]$ cat public.pgp
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGUG6/QBEACr+Omdim5+kaifwiIT0mZ6Vac1z1qphE+tjCzmhHUXJjIxxIpI
RjGZwV3xp3pVxEXmFXcQUUnQhARgif5TBMobJ82jMSe0wNbg6uu2NxjN2PntJfyp9
yLg9CPGsGEVMO/kQ1K//aTQtSS3ScpfwZQ3ym8zPZHDJoUerj90GQjh+N7RHv5uG
VPjxcNUFbrwFkKl5J1CaqYSQxdvhX5NDN/OnNpCPHzrxhip+u7JwmkMJ7x5gBYnR
3YZs502S6eg0sP7p75KPu4M5pcjDpQRHuc7u6MF2DPpfidTjz1bKrUJaPde+Jdnk
Ya00S7pZbkm0qxWaBIkF68G7dxw5HcqRjLqz1I/PTMKRE1Ao2LCFI8i+f6DtqQxR
Iu0jJPhDyMZISM7k06J5Y3FL5m4c+UUnk2Wxc5x7KftnC28kvw0v50WhwMBoHwWj
mCieQlxLy2mpGBY+raKvYuk00fatKj96/QkwLKHp0UAbgZGJyHzPz847aTJeUE6J
EURBcqF0f9a2myQT40acZINfz5aG2lgU589PFth8jynp37pp4WNJuyTvzmd4X6ay
qiMl3URcBJoGa1Skuek49poLXR99M6xXbp3eJeEw6d0YnivE27S1bu63epM71Qtd
cBs+5Cogr3flvQ8jzfdJrJEG+3SF1UIg0HYFbd0L4bw560laUB/ugbkpiwARAQAB
tCVLc2VuaWlhIEIxb250ZXZhIDxrc3VsZW8yM0BnbWVpbC5jb20+iQJ0BBMBCAA4
FiEECHmdxTC6L1BbtR9uXKFNZoqTcvUFAMUG6/QCGwMFCwkIBWIGFQoJCAAsCBBYC
AwECHgECF4AACgkQXKFNZoqTcvXwMw//c/G1JFZ519vgIKoYyUNRd8eMN1ZH6xG4
oCZK33m7sQIdJCJB8P4ZvzrKPL3R/KI2RmMsjwPhrMmaeJaJqpc09/hEa8it+6G
gkdqItHfahNtGjtCFyXxvmrXK8hh2fothUkfVUKSF/XQoh0cxBqVpZyoqUcMlFG/
Wc29Jgia3Dhg2ipQoAuxPAepG9GKJUxV8IS1B49W6/uSLBVS+2wWVmzxbooITNRD
Qo9ccT3thWt+TBkMQNi8GkEeUdta0tWTdgcTStmqLs/lcB0LxRfdFbuaXi0Z0xw
wsjIRL8Adq+nEC2stv4X+k7ACr9xM+52ZznK0stsKW40BUwg55lnE1QV2a3pRHNZ
C55I0qDJ0ladZfZBlIZ1xX/+3I8vC8as06GvX92LDh0LUAxhm6tjUgnJXmwGebQY
CKXB5jaLn8ijmpKt3LDFx0TB1Nrc0KSCCrcSS0AYR/hx/F0bc7ykjxgRU3NxyR+e
TwkwhC0pRhj9qG5GBIbxW20+iu0Lgy1jJ7/RZMnTLJPiBqk2kqyYtGv9ns4+b53o
ppg80X7GvR9Sfkkqz/QwwjglPhBzGj7pMtJg/zUBtP7XopgzGLGBjmNRFa6LUtN4
U/Jmi9KDFLDa9dh0GXJxhBzQ01hbSySN2SfzzbB8ZAWo9EGSZAI2u4MTkQ0PIbhb
```

Рис. 3.8: Копирование сгенерированного ключа PGP

Затем переходим в настройки GitHub, в раздел keys и, нажав на кнопку New GPG key, вставляем полученный ключ в поле ввода (рис. fig. 3.9).

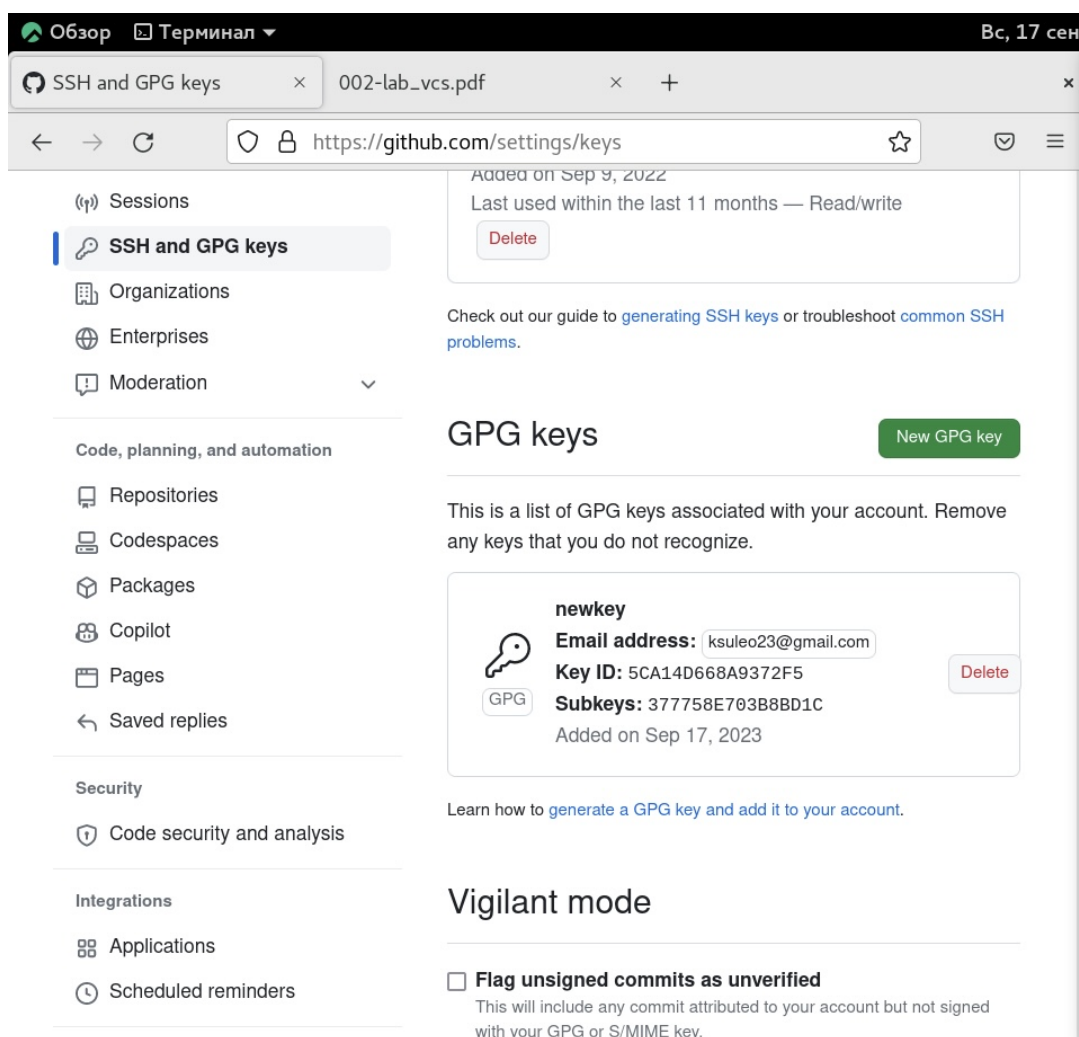


Рис. 3.9: Добавление gpg-ключа на GitHub

Настраиваем автоматические подписи коммитов git. Используя введенный email, указываем Git поменять его при подписи коммитов (рис. fig. 3.10).

```
[kaleontjeval@kaleontjeva ~]$ git config --global user.signingkey ksuleo23@gmail.com
[kaleontjeval@kaleontjeva ~]$ git config --global commit.gpgsign true
[kaleontjeval@kaleontjeva ~]$ git config --global gpg.program $(which gpg2)
[kaleontjeval@kaleontjeva ~]$
```

Рис. 3.10: Настройка автоматических подписей коммитов git

Устанавливаем и настраиваем gh. Отвечаем на наводящие вопросы утилиты и авторизуемся через браузер (рис. fig. 3.11 - рис. fig. 3.13).

```

Обзор Терминал
Бс, 17 сентября 16:08
kaleontjeva1@kaleontjeva:~
Файл Правка Вид Поиск Терминал Справка
[kaleontjeva1@kaleontjeva ~]$ type -p yum-config-manager >/dev/null || sudo yum install yum-utils
Последняя проверка окончания срока действия метаданных: 0:12:29 назад, Бс 17 сен 2023 15:53:25.
Зависимости разрешены.
=====
Пакет                Архитектура  Версия                Репозиторий  Размер
=====
Установка:
yum-utils             noarch       4.0.21-19.el8_8      baseos       74 k
Обновление:
dnf-plugins-core      noarch       4.0.21-19.el8_8      baseos       74 k
python3-dnf-plugins-core noarch       4.0.21-19.el8_8      baseos      260 k
=====
Результат транзакции
=====
Установка   1 Пакет
Обновление  2 Пакета

Объем загрузки: 408 k
Продолжить? [д/Н]: l
Продолжить? [д/Н]: д
Загрузка пакетов:
(1/3): dnf-plugins-core-4.0.21-19.el8_8.noarch.rpm 199 kB/s | 74 kB 00:00
(2/3): yum-utils-4.0.21-19.el8_8.noarch.rpm       196 kB/s | 74 kB 00:00
(3/3): python3-dnf-plugins-core-4.0.21-19.el8_8.noarc 516 kB/s | 260 kB 00:00
-----
Общий размер                332 kB/s | 408 kB 00:01
Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции проведен успешно
Выполнение транзакции
Подготовка      : 1/1
Обновление      : python3-dnf-plugins-core-4.0.21-19.el8_8.noarch 1/5
Обновление      : dnf-plugins-core-4.0.21-19.el8_8.noarch 2/5
Установка       : yum-utils-4.0.21-19.el8_8.noarch 3/5

```

Рис. 3.11: Установка gh

```
Обзор Терминал Вc, 17 сентября 16:08
kaleontjeva1@kaleontjeva:~
Файл Правка Вид Поиск Терминал Справка
Обновлен:
dnf-plugins-core-4.0.21-19.el8_8.noarch
python3-dnf-plugins-core-4.0.21-19.el8_8.noarch
Установлен:
yum-utils-4.0.21-19.el8_8.noarch
Выполнено!
[kaleontjeva1@kaleontjeva ~]$ sudo yum-config-manager --add-repo https://cli.github.com/packages/rpm/gh-cli.repo
Добавление репозитория из: https://cli.github.com/packages/rpm/gh-cli.repo
[kaleontjeva1@kaleontjeva ~]$ sudo yum install gh
packages for the GitHub CLI 2.1 kB/s | 2.6 kB 00:01
Последняя проверка окончания срока действия метаданных: 0:00:01 назад, Вc 17 сен 2023 16:06:49.
Зависимости разрешены.
=====
Пакет Архитектура Версия Репозиторий Размер
=====
Установка:
gh x86_64 2.34.0-1 gh-cli 11 M
Результат транзакции
=====
Установка 1 Пакет
Объем загрузки: 11 M
Объем изменений: 41 M
Продолжить? [д/н]: д
Загрузка пакетов:
gh_2.34.0_linux_amd64.rpm 1.5 MB/s | 11 MB 00:07
-----
Общий размер 1.5 MB/s | 11 MB 00:07
packages for the GitHub CLI 4.9 kB/s | 3.1 kB 00:00
Импорт GPG-ключа 0x75716059:
Идентификатор пользователя: "GitHub CLI <opensource+cli@github.com>"
Отпечаток: 2C61 0620 1985 B60E 6C7A C873 23F3 D4EA 7571 6059
Источник: https://keyserver.ubuntu.com/pks/lookup?op=get&search=0x23F3D4EA75716059
=====
```

Рис. 3.12: Установка gh

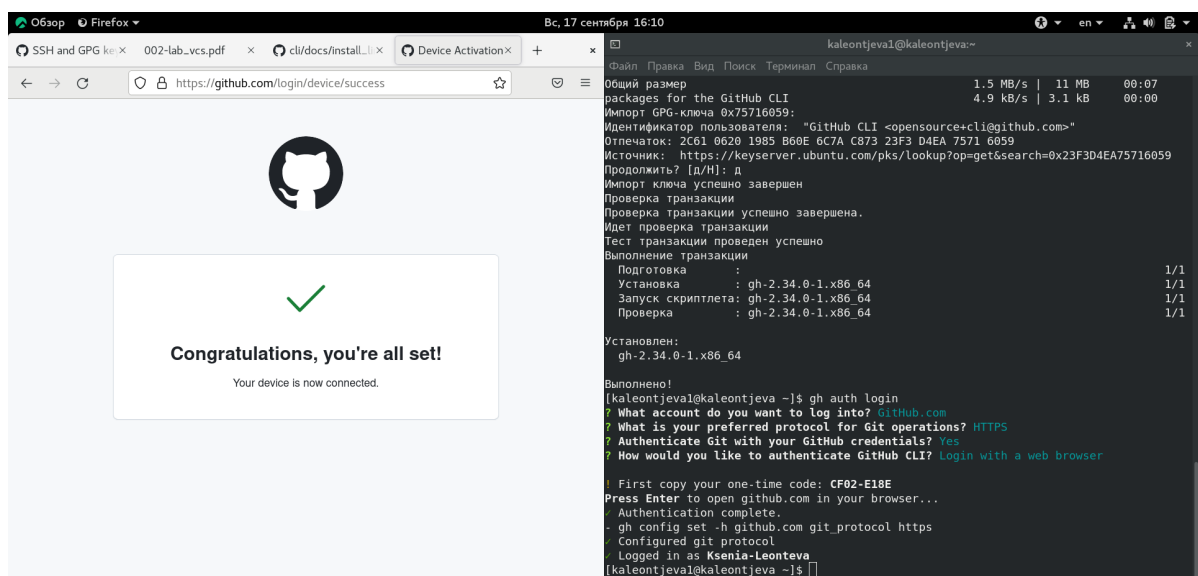


Рис. 3.13: Настройка gh и авторизация

Создаем репозиторий курса на основе шаблона (рис. fig. 3.14), параллельно сгенерировав и добавив на GitHub еще один ssh-ключ (рис. fig. 3.15 и рис.



fig. 3.16).

```
[kaleontjeval@kaleontjeva ~]$ cd ~/work/study
[kaleontjeval@kaleontjeva study]$ mkdir -p 2023-2024/"Научное программирование"
[kaleontjeval@kaleontjeva study]$ cd 2023-2024/"Научное программирование"
[kaleontjeval@kaleontjeva Научное программирование]$ gh repo create study_2023-2024_sc
iprog-intro--template=yamadharma/course-directory-student-template --public
HTTP 404: Not Found (https://api.github.com/users/study_2023-2024_sciprog-intro--templ
ate=yamadharma)
[kaleontjeval@kaleontjeva Научное программирование]$ gh repo create study_2023-2024_sc
iprog--template=yamadharma/course-directory-student-template --public
HTTP 404: Not Found (https://api.github.com/users/study_2023-2024_sciprog--template=ya
madharma)
[kaleontjeval@kaleontjeva Научное программирование]$ gh repo create study_2023-2024_sc
iprog --template=yamadharma/course-directory-student-template --public
✓ Created repository Ksenia-Leonteva/study_2023-2024_sciprog on GitHub
```

Рис. 3.14: Создание репозитория курса

```
[kaleontjeval@kaleontjeva Научное программирование]$ ssh-keygen -C "Kseniia Leonteva <
ksuleo23@gmail.com>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kaleontjeval/.ssh/id_rsa):
/home/kaleontjeval/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kaleontjeval/.ssh/id_rsa.
Your public key has been saved in /home/kaleontjeval/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:T9SJ94A/aWhP7ca80Vbfm1GGt9Kzk2c7GhuRUKV+sLE Kseniia Leonteva <ksuleo23@gmail.co
m>
The key's randomart image is:
+---[RSA 3072]-----+
|           +o  |
|          + .o= |
|         + =E.. |
|        . +.=... |
|       S +.*O+ + |
|      + +.=oo+ |
|     . .+=*+ |
|    .B*X |
|   ++B* |
+---[SHA256]-----+
[kaleontjeval@kaleontjeva Научное программирование]$ cat ~/.ssh/id_rsa.pub | xclip -se
l clip
bash: xclip: команда не найдена...
Установить пакет «xclip», предоставляющий команду «xclip»? [N/y] y
```

Рис. 3.15: Генерация ssh-ключа

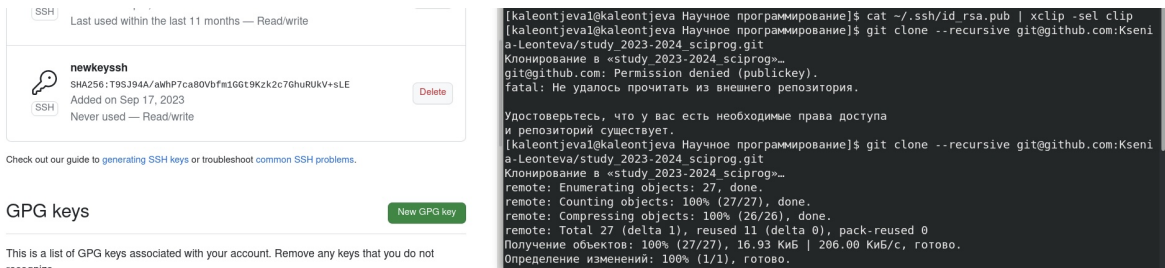


Рис. 3.16: Добавление ssh-ключа на GitHub и завершение создания репозитория курса

Настраиваем каталог курса (рис. fig. 3.17 и рис. fig. 3.18):

- переходим в каталог курса,
- удаляем лишние файлы,
- создаем необходимые каталоги,
- отправляем файлы на сервер.

```
[kaleontjeval@kaleontjeva Научное программирование]$ cd sciprog
[kaleontjeval@kaleontjeva sciprog]$ rm package.json
[kaleontjeval@kaleontjeva sciprog]$ make COURSE=sciprog
[kaleontjeval@kaleontjeva sciprog]$ git add .
[kaleontjeval@kaleontjeva sciprog]$ git commit -am 'feat(main): make course structure'
[master 39f1839] feat(main): make course structure
314 files changed, 86648 insertions(+), 14 deletions(-)
create mode 100644 labs/README.md
create mode 100644 labs/README.ru.md
create mode 100644 labs/lab1/presentation/Makefile
create mode 100644 labs/lab1/presentation/image/kulyabov.jpg
create mode 100644 labs/lab1/presentation/presentation.md
create mode 100644 labs/lab1/report/Makefile
create mode 100644 labs/lab1/report/bib/cite.bib
create mode 100644 labs/lab1/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab1/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
```

Рис. 3.17: Настройка каталога курса

```
[kaleontjeval@kaleontjeva sciprog]$ git push
Перечисление объектов: 41, готово.
Подсчет объектов: 100% (41/41), готово.
Сжатие объектов: 100% (31/31), готово.
Запись объектов: 100% (40/40), 343.21 КиБ | 2.84 МиБ/с, готово.
Всего 40 (изменений 5), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (5/5), completed with 1 local object.
To github.com:Ksenia-Leonteva/study_2023-2024_sciprog.git
aaadffb..39f1839 master -> master
[kaleontjeval@kaleontjeva sciprog]$
```

Рис. 3.18: Настройка каталога курса

## 4 Контрольные вопросы

1. Система контроля версий (Version Control System, VCS) — это программное обеспечение, которое помогает отслеживать изменения в файловой системе и эффективно управлять версиями файлов и кода в проекте. Она позволяет разработчикам работать над проектами совместно, отслеживать, комментировать и объединять свои изменения.
2. Хранилище — место, куда помещается документ после внесения в него нужных правок. Оно является местом хранения служебной информации и всех версий документов. Commit — это пакет изменений, хранящий информацию с добавленными, отредактированными или удаленными файлами (в Git — это команда для записи индексированных изменений в репозиторий Git). История (в Git) — точный реестр всех коммитов, содержащих произведенные с файлами изменения. Рабочая копия — текущее состояние файлов проекта, основанное на версии, загруженной из хранилища.
3. Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере. Доступ к нему осуществляется через специальное клиентское приложение. В качестве примеров таких программных продуктов можно привести CVS, Subversion.

Децентрализованные (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”. Две наиболее известные DVCS – это Git, Bazaar, Mercurial.

В отличие от централизованных, в распределенных системах контроля версий центральный репозиторий не является обязательным.

4. Создание локального репозитория. Делается предварительная конфигурация:

```
git config --global user.name "Имя Фамилия"
```

```
git config --global user.email "work@mail"
```

Настраивается utf-8 в выводе сообщений git:

```
git config --global quotepath false
```

Для инициализации локального репозитория, расположенного, например, в каталоге ~/tutorial, необходимо ввести в командной строке:

```
cd
```

```
mkdir tutorial
```

```
cd tutorial
```

```
git init
```

5. Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый):

```
ssh-keygen -C "Имя Фамилия work@mail"
```

Ключи сохраняются в каталоге ~/.ssh/. Скопировав из локальной консоли ключ в буфер обмена:

```
cat ~/.ssh/id_rsa.pub | xclip -sel clip
```

вставляем ключ в появившееся на сайте поле.

6. Основные задачи, решаемые инструментальным средством git: обеспечение удобной командной работы над проектом и хранение информации о всех изменениях в проекте.

7. Наиболее часто используемые команды git:

- git init – создание основного дерева репозитория,
- git pull – получение обновлений (изменений) текущего дерева из центрального репозитория,
- git push – отправка всех произведённых изменений локального дерева в центральный репозиторий,
- git status – просмотр списка изменённых файлов в текущей директории,
- git diff – просмотр текущих изменения,
- git add . – добавить все изменённые и/или созданные файлы и/или каталоги,
- git add имена\_файлов – добавить конкретные изменённые и/или созданные файлы и/или каталоги,
- git rm имена\_файлов – удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории),
- git commit -am 'Описание коммита' – сохранить все добавленные изменения и все изменённые файлы,
- git commit – сохранить добавленные изменения с внесением комментария через встроенный редактор,
- git checkout -b имя\_ветки – создание новой ветки, базирующейся на текущей,

- `git checkout имя_ветки` – переключение на некоторую ветку,
- `git push origin имя_ветки` – отправка изменений конкретной ветки в центральный репозиторий,
- `git merge --no-ff имя_ветки` – слияние ветки с текущим деревом,
- `git branch -d имя_ветки` – удаление локальной уже слитой с основным деревом ветки,
- `git branch -D имя_ветки` – принудительное удаление локальной ветки,
- `git push origin :имя_ветки` – удаление ветки с центрального репозитория.

8. Создание тестового файла `hello.txt` и добавление его в локальный репозиторий:

```
echo 'hello world' > hello.txt
git add hello.txt
git commit -am 'Новый файл'
```

9. Ветки нужны для того, чтобы при работе над проектом программисты работали независимо друг от друга. В Git ветки — это элемент повседневного процесса разработки. По сути ветки в Git представляют собой указатель на снимок изменений. Если нужно добавить новую возможность или исправить ошибку (незначительную или серьезную), создается новая ветка, в которой будут размещаться эти изменения.
10. Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты. Игнорируемые файлы отслеживаются в специальном файле `.gitignore`, который регистрируется в корневом каталоге репозитория. В Git нет специальной команды для указания игнорируемых файлов: вместо этого необходимо

вручную отредактировать файл `.gitignore`, чтобы указать в нем новые файлы, которые должны быть проигнорированы. Файлы `.gitignore` содержат шаблоны, которые сопоставляются с именами файлов в репозитории для определения необходимости игнорировать эти файлы.

## 5 Вывод

В ходе выполнения данной лабораторной работы я изучила идеологию и применение средств контроля версий, а также освоила умения по работе с git.

## Список литературы

1. Системы контроля версий [Электронный ресурс]. 2023. URL: <https://apptractor.ru/info/articles/chto-takoe-sistema-kontrolya-versiy.html>.