

# **Отчет по лабораторной работе №2**

**Дисциплина: Операционные системы**

**Ситникова Диана Александровна Группа: НПИбд-01-22**

**Москва 2023г**

# Содержание

<b>Цель работы</b>	<b>5</b>
<b>Задание</b>	<b>6</b>
<b>Выполнение лабораторной работы</b>	<b>7</b>
Установка программного обеспечения . . . . .	7
Базовая настройка git . . . . .	8
Создание ключа ssh . . . . .	9
Создание ключа pgp . . . . .	10
Добавление PGP ключа в GitHub . . . . .	12
Настройка автоматических подписей коммитов git . . . . .	13
Настройка gh . . . . .	14
Создание репозитория курса на основе шаблона . . . . .	17
Настройка каталога курса . . . . .	18
<b>Контрольные вопросы</b>	<b>20</b>
<b>Выводы</b>	<b>27</b>

## **Список иллюстраций**

## Список таблиц

# Цель работы

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

# 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 программного обеспечения

Установим git при помощи команды:

`dnf install git`

```
[dasitnikova@dasitnikova ~]$ sudo -i
[sudo] пароль для dasitnikova:
[root@dasitnikova ~]# dnf install git
Fedora 38 - x86_64 - Updates                               65 kB/s | 13 kB    00:00
Fedora 38 - x86_64 - Updates                               1.6 MB/s | 2.5 MB  00:01
Fedora Modular 38 - x86_64 - Updates                       28 kB/s | 18 kB    00:00
Пакет git-2.41.0-1.fc38.x86_64 уже установлен.
Зависимости разрешены.
Отсутствуют действия для выполнения.
Выполнено!
[root@dasitnikova ~]#
```

Установим gh при помощи команды:

`dnf install gh`

```
[root@dasitnikova ~]# dnf install gh
Последняя проверка окончания срока действия метаданных: 0:03:25 назад, Ср 01 ноя 2023 18:47:33.
Зависимости разрешены.
=====
Пакет          Архитектура      Версия           Репозиторий      Размер
=====
Установка:
gh             x86_64           2.36.0-1.fc38   updates          8.9 М
Результат транзакции
=====
Установка 1 Пакет

Объем загрузки: 8.9 М
Объем изменений: 44 М
Продолжить? [д/Н]:
```

```
Установлен:
  gh-2.36.0-1.fc38.x86_64

Выполнено!
[root@dasilnikova ~]#
```

## Базовая настройка git

- Зададим имя и email владельца репозитория при помощи следующих команд:

```
git config --global user.name "Name Surname"
```

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

- Настроим utf-8 в выводе сообщений git командой:

```
git config --global core.quotePath false
```

```
[root@dasilnikova ~]# git config --global user.name "Diana Sitnikova"
[root@dasilnikova ~]# git config --global user.email "sitnickova.diana@gmail.com"
[root@dasilnikova ~]# git config --global core.quotePath false
[root@dasilnikova ~]#
```

- Зададим имя начальной ветки (будем называть её master):

```
git config --global init.defaultBranch master
```

- Параметр autocrlf:

```
git config --global core.autocrlf input
```

- Параметр safecrlf:

```
git config --global core.safecrlf warn
```



```

[root@dositnikova ~]# git config --global init.defaultBranch master
[root@dositnikova ~]# git config --global core.autocrlf input
[root@dositnikova ~]# git config --global core.safecrlf warn
[root@dositnikova ~]# 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 (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:E5o3ZMzaFQlnNMB9uKEnJKJePLFohILkrce25egA root@dositnikova
The key's randomart image is:
+---[RSA 4096]-----+
|oo      =+B0o.      |
|o . o +o==o= .     |
| . o = +X.. +      |
|E . + o0oo .       |
| = + o+ S          |
|o o o . o          |
|. . .              |
|   o               |
|.o                 |
+----[SHA256]-----+
[root@dositnikova ~]# █

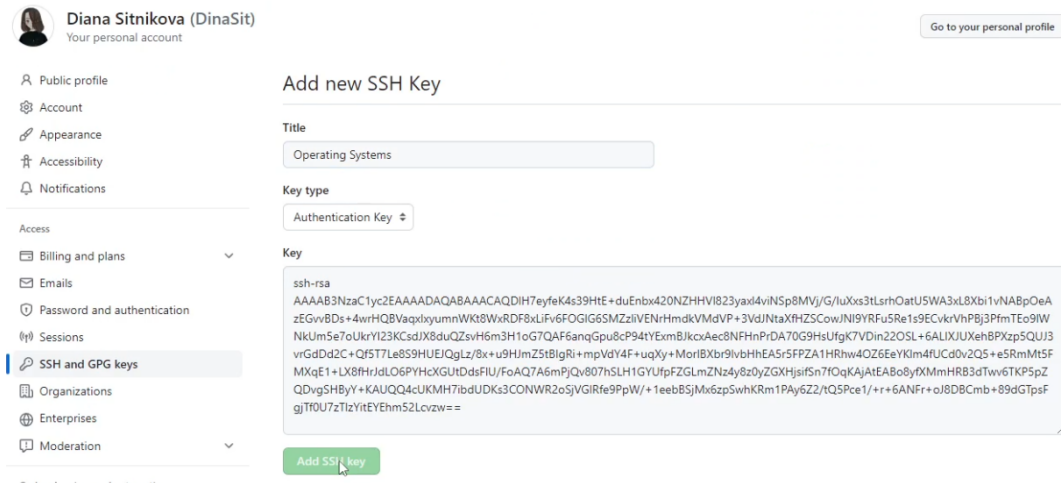
```

## Созданиче ключа ssh

- по алгоритму rsa с ключём размером 4096 бит:  
ssh-keygen -t rsa -b 4096  
(см. “Рис5”)
- для копирования ключа нам понадобится команда:  
cat ~/.ssh/id\_rsa.pub

```
[root@dasitnikova ~]# cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQADlH7eyfeK4s39HtE+duEnbx420NZHHVl823yaxl4viNSp8MVj/G/IuXxs3tL
srhOatU5WA3xL8Xbi1vNABp0eAzEGvvBDs+4wrHQBVaqxIxyumWkT8WxRDF8xLiFv6F0G1G6SMZzliVENrHmdkVMdVP+3VdJN
taXfHZSCowJNl9YRFu5Re1s9ECvkrVhPBj3PfmTEo9lWNkUm5e7oUkrYI23KCsdlX8duQZsvH6m3H1oG7QAF6anqGpu8cP94tY
ExmBJkcxAc8NFHnPrDA70G9HsUfgK7VDin22OSL+6ALIXJUXehBPXzp5QUJ3vrGdDd2C+Qf5T7Le8S9HUEJQgLz/8x+u9HJmZ
5tBIgRi+mpVdY4F+uqXy+MorlBXbr9IvbHhEA5r5FPZA1HRhw4OZ6EeYKlm4fUCd0v2Q5+e5RmMt5FMXqE1+LX8fHrJdL06PYH
cXGUtDdsFlU/FoAQ7A6mPjQv807hSLH1GYUfpFZGLmZNz4y8z0yZGXHjsifSn7f0qKAjAtEABo8yfXMMHRB3dTwv6TKP5pZQDv
gSHByY+KAUQQ4cUKMH7ibUDKs3CONWR2o5jVGIrfe9PpW/+1eeb85jMx6zpSwhKrm1PAy6Z2/rQ5Pce1/+r+6ANFr+oJ8DBCm
b+89dGTpsFgjTf0U7zTlzYitEYEHm52Lcvzw== root@dasitnikova
[root@dasitnikova ~]#
```

- далее добавляем скопированный ключ SSH на GitHub:



## Создание ключа ргр

- Генерируем ключ при помощи команды:  
gpg --full-generate-key
- Из предложенных опций выбираем:
  - тип RSA and RSA;
  - размер 4096;
  - выберите срок действия; значение по умолчанию — 0 (срок действия не истекает никогда).
- GPG запросит личную информацию, которая сохранится в ключе:
  - Имя (не менее 5 символов).

- Адрес электронной почты.
  - \* При вводе email убедитесь, что он соответствует адресу, используемому на GitHub.
- Комментарий. Можно ввести что угодно или нажать клавишу ввода, чтобы оставить это поле пустым.

```
[root@dasilnikova ~]# gpg --full-generate-key
gpg (GnuPG) 2.4.0; Copyright (C) 2021 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.

gpg: создан каталог '/root/.gnupg'
gpg: создан щит с ключами '/root/.gnupg/pubring.kbx'
Выберите тип ключа:
  (1) RSA and RSA
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (9) ECC (sign and encrypt) *default*
 (10) ECC (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? (3072) 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
    I <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: Diana
Адрес электронной почты: sitnickova.diana@gmail.com
Примечание: 
```

```
GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: Diana
Адрес электронной почты: sitnickova.diana@gmail.com
Примечание:
Вы выбрали следующий идентификатор пользователя:
    "Diana <sitnickova.diana@gmail.com>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? O
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: /root/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/root/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/root/.gnupg/openpgp-revocs.d/ABC4309B9AECEDEF4E7EE3A9A014F1F727513168.rev'
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2023-11-01 [SC]
      ABC4309B9AECEDEF4E7EE3A9A014F1F727513168
uid                               Diana <sitnickova.diana@gmail.com>
sub   rsa4096 2023-11-01 [E]

[root@dasitnikova ~]#
```

## Добавление PGP ключа в GitHub

- Выводим список ключей и копируем отпечаток приватного ключа: `gpg --list-secret-keys --keyid-format LONG`
- Отпечаток ключа — это последовательность байтов, используемая для идентификации более длинного, по сравнению с самим отпечатком ключа.
- Формат строки:

sec Алгоритм/Отпечаток ключа Дата\_создания [Флаги] [Годен\_до] ID\_ключа

```
[root@dasitnikova ~]# 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, 0f, 1u
/root/.gnupg/pubring.kbx
-----
sec   rsa4096/A014F1F727513168 2023-11-01 [SC]
      ABC4309B9AECEDEF4E7EE3A9A014F1F727513168
uid                               [ абсолютно ] Diana <sitnickova.diana@gmail.com>
ssb   rsa4096/5866B889423F39CB 2023-11-01 [E]

[root@dasitnikova ~]#
```

- Скопируйте ваш сгенерированный PGP ключ в буфер обмена:

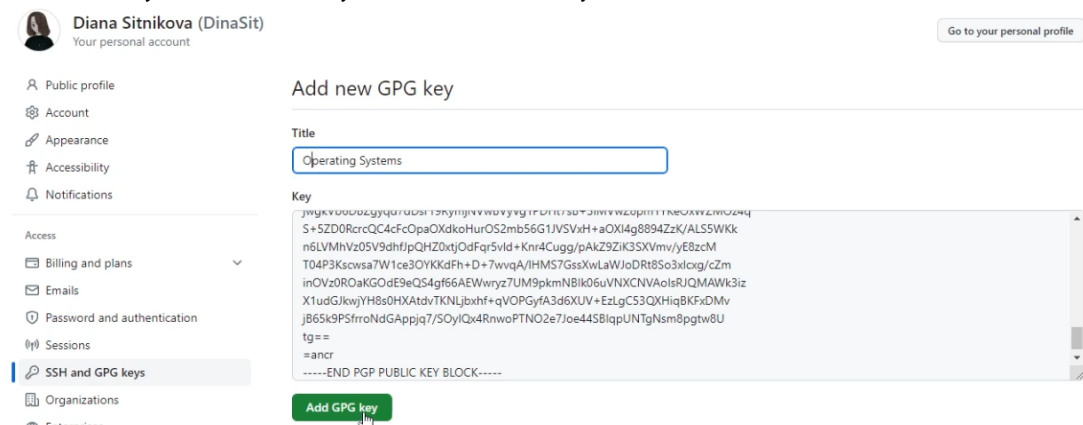
```
gpg --armor --export | xclip -sel clip
```

```
[root@dasilnikova ~]# gpg --armor --export sitnickova.diana@gmail.com
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGVCf2QBEADEE0AW8yytr6OuX1GsRBXglNScWwmmJU/njJN5cBb1fun5EwHc
Nb+IkpyslGW269JzSrfgVE4D7CF/BJQw2LbgoZ/DM7S1fRX7btPZRYcY10QPLk0t
MqIvwINfsGZ1xn4KKZGuPyYnfG1/tdmho2uiv415FrnAMRCuLCJ0hRwBQ7kpZpv
nliXp11F/C5EJ4coDb/mKCKG1bAMl+dn1uydt9dBdBfM4A5J+0bcPjduxXo+Ycge
E4mEUgPDYGKRgfgcXIN11fyTQfKRB0aI6UCBUBPIyy3SkJzK2bWppqR2tCz42z18
U/iZtXBR91X242DPi4Kez7A9tjvnmVTKJq5Lxf+Nk3Rvz0q5IBXs+4cDInzqrq8o
wPEdzPJ7tAWxsfc1THN87d2fwaI2RUd0X6xtwYbi2u8Zng57EJpyK1cyypQHv47V
dA9ygCfhCuLMRm8V0amO6VgFo3dIdQNVrNhstrmRD0MyT4a5YKtdbDGGiyI4hFRY
AJhnegNCkqULAKNureCOik6+jErGLWivBSPJ9wYcrW2vigK+sJQg4w1DHj+fIYFU
wnrFu57SEcRjLjgo+Bqj16FDJt16qmGpccqOk2jEuZiCix7yVlVTzw05vVc0NAhV
XFMFJxZzf1nMw3Q8qneH/Dq3JB/oXpTYHQdMpp3bpC6HAXC5rh7f3XANDQARAQAB
```

Примечание: я использовала в качестве параметра свою почту, указанную при создании ключа PGP, однако, корректнее было бы использовать “Отпечаток\_ключа” как на “Рис10” - A014F1F727513168.

- Перейдите в настройки GitHub (<https://github.com/settings/keys>), нажмите на кнопку New GPG key и вставьте полученный ключ в поле ввода.



## Настройка автоматических подписей коммитов git

- Используя введенный email, укажите Git применять его при подписи КОММИТОВ:

```
git config --global user.signingkey
```

git config --global commit.gpgsign true

git config --global gpg.program S(which gpg2)

```
[root@dasilnikova ~]# git config --global user.signingkey sitnickova.diana@gmail.com
[root@dasilnikova ~]# git config --global commit.gpgsign true
[root@dasilnikova ~]# git config --global gpg.program $(which gpg2)
[root@dasilnikova ~]#
```

## Настройка gh

- Для начала необходимо авторизоваться

gh auth login

- Утилита задаст несколько наводящих вопросов.

```
[root@dasilnikova ~]# gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: D83C-6812
Press Enter to open github.com in your browser...
```

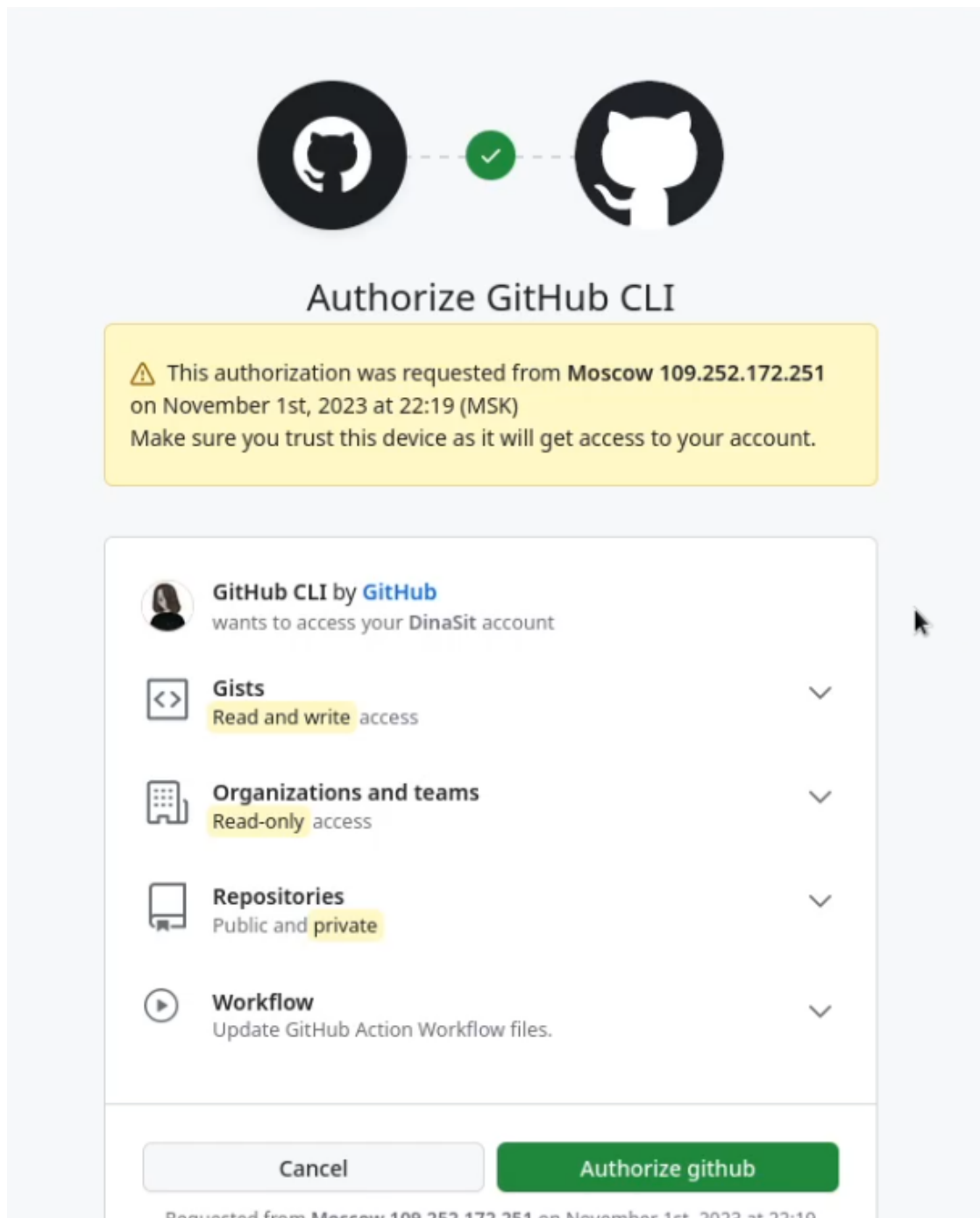
- Авторизоваться можно через браузер.

```

! First copy your one-time code: D83C-6812
Press Enter to open github.com in your browser...
restorecon: SELinux: Could not get canonical path for /root/.mozilla/firefox/* restorecon: No such
file or directory.
Running Firefox as root in a regular user's session is not supported. ($XAUTHORITY is /run/lightd
m/dasitnikova/xauthority which is owned by dasitnikova.)
/usr/bin/xdg-open: строка 881: x-www-browser: команда не найдена
restorecon: SELinux: Could not get canonical path for /root/.mozilla/firefox/* restorecon: No such
file or directory.
Running Firefox as root in a regular user's session is not supported. ($XAUTHORITY is /run/lightd
m/dasitnikova/xauthority which is owned by dasitnikova.)
/usr/bin/xdg-open: строка 881: iceweasel: команда не найдена
/usr/bin/xdg-open: строка 881: seamonkey: команда не найдена
/usr/bin/xdg-open: строка 881: mozilla: команда не найдена
/usr/bin/xdg-open: строка 881: epiphany: команда не найдена
/usr/bin/xdg-open: строка 881: konqueror: команда не найдена
/usr/bin/xdg-open: строка 881: chromium: команда не найдена
/usr/bin/xdg-open: строка 881: chromium-browser: команда не найдена
/usr/bin/xdg-open: строка 881: google-chrome: команда не найдена
/usr/bin/xdg-open: строка 881: www-browser: команда не найдена
/usr/bin/xdg-open: строка 881: links2: команда не найдена
/usr/bin/xdg-open: строка 881: elinks: команда не найдена
/usr/bin/xdg-open: строка 881: links: команда не найдена
/usr/bin/xdg-open: строка 881: lynx: команда не найдена
/usr/bin/xdg-open: строка 881: w3m: команда не найдена
xdg-open: no method available for opening 'https://github.com/login/device'
Failed opening a web browser at https://github.com/login/device
exit status 3
Please try entering the URL in your browser manually

```





- Результат выполнения корректной работы по настройке gh:



```

xdg-open: no method available for opening 'https://github.com/login/device'
! Failed opening a web browser at https://github.com/login/device
exit status 3
Please try entering the URL in your browser manually
✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
! Authentication credentials saved in plain text
✓ Logged in as DinaSit
[root@dasilnikova ~]#

```

## Создание репозитория курса на основе шаблона

- Необходимо создать шаблон рабочего пространства (см. Рабочее пространство для лабораторной работы).
- Например, для 2022–2023 учебного года и предмета «Операционные системы» (код предмета os-intro) создание репозитория примет следующий вид:

```
mkdir -p ~/work/study/2022-2023/"Операционные системы"
```

```
cd ~/work/study/2022-2023/"Операционные системы" gh repo create
study_2022-2023_os-intro --template=yamadharm/course-directory-student-
template --public
```

```
git clone --recursive git@github.com:/study_2022-2023_os-intro.git os-intro
```

```

[root@dasilnikova ~]# mkdir -p ~/work/study/2022-2023/"Операционные системы"
[root@dasilnikova ~]# cd ~/work/study/2022-2023/"Операционные системы"
[root@dasilnikova Операционные системы]# gh repo create study_2022-2023_os-intro --template=yamadh
arma/course-directory-student-template --public
✓ Created repository DinaSit/study_2022-2023_os-intro on GitHub

```

```
[root@dasilnikova Операционные системы]# git clone --recursive git@github.com:DinaSit/study_2022-2023_os-intro.git os-intro
Клонирование в «os-intro»...
The authenticity of host 'github.com (140.82.121.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
remote: Enumerating objects: 28, done.
remote: Counting objects: 100% (28/28), done.
remote: Compressing objects: 100% (27/27), done.
remote: Total 28 (delta 1), reused 18 (delta 0), pack-reused 0
Получение объектов: 100% (28/28), 17.29 КиБ | 4.32 Миб/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/root/work/study/2022-2023/Операционные системы/os-intro/template/presentation»...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 82 (delta 28), reused 77 (delta 23), pack-reused 0
Получение объектов: 100% (82/82), 92.90 КиБ | 1.01 Миб/с, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в «/root/work/study/2022-2023/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 101, done.
remote: Counting objects: 100% (101/101), done.
remote: Compressing objects: 100% (70/70), done.
remote: Total 101 (delta 40), reused 88 (delta 27), pack-reused 0
Получение объектов: 100% (101/101), 327.25 КиБ | 2.11 Миб/с, готово.
Определение изменений: 100% (40/40), готово.
Submodule path 'template/presentation': checked out 'b1be3800ee91f5809264cb755d316174540b753e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a83917b82e3aef11a33b1e3b2'
```

## Настройка каталога курса

- Перейдите в каталог курса:  
`cd ~/work/study/2022-2023/“Операционные системы”/os-intro`
- Удалите лишние файлы:  
`rm package.json`
- Создайте необходимые каталоги:  
`echo os-intro > COURSE`  
`make`

```
[root@dasitnikova Операционные системы]# cd ~/work/study/2022-2023/"Операционные системы"/os-intro
[root@dasitnikova os-intro]# rm package.json
rm: удалить обычный файл 'package.json'? yes
[root@dasitnikova os-intro]# echo os-intro > COURSE
[root@dasitnikova os-intro]# make
[root@dasitnikova os-intro]# ls
CHANGELOG.md  COURSE  LICENSE  prepare  project-personal  README.git-flow.md  template
config        labs    Makefile  presentation  README.en.md      README.md
```

- Отправьте файлы на сервер:

git add .

git commit -am 'feat(main): make course structure'

git push

```
[root@dasitnikova os-intro]# git commit -am "feat(main): make course structure"
error: gpg failed to sign the data:
[GNUPG:] KEY_CONSIDERED ABC4309B9AECEDEF4E7EE3A9A014F1F727513168 2
[GNUPG:] BEGIN_SIGNING H8
[GNUPG:] PINENTRY_LAUNCHED 4041 curses 1.2.1 - xterm-256color :0 - 0/0 0
gpg: подписать не удалось: Неприменимый к данному устройству ioctl
[GNUPG:] FAILURE sign 83918950
gpg: signing failed: Неприменимый к данному устройству ioctl

fatal: сбой записи объекта коммита
[root@dasitnikova os-intro]# ls
CHANGELOG.md  COURSE  LICENSE  prepare  project-personal  README.git-flow.md  template
config        labs    Makefile  presentation  README.en.md      README.md
[root@dasitnikova os-intro]# export GPG_TTY=$(tty)
[root@dasitnikova os-intro]# git commit -am "feat(main): make course structure"
```

Примечание: если при коммите выводит ошибку как на “Рис21”, может помочь команда

export GPG\_TTY=\$(tty)

# Контрольные вопросы

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

Это программное обеспечение для облегчения работы с изменяющейся информацией. VCS позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

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

Хранилище (repository), или репозиторий, — место хранения всех версий и служебной информации.

Commit («[трудовой] вклад», не переводится) — синоним версии; процесс создания новой версии.

История — место, где сохраняются все коммиты, по которым можно посмотреть данные о коммитах.

Рабочая копия — текущее состояние файлов проекта, основанное на версии, загруженной из хранилища.

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

Централизованные VCS: одно основное хранилище всего проекта и каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно. Например: CVS (Concurrent Versions System, Система одновременных версий), Subversion

(SVN).

Децентрализованные VCS: у каждого пользователя свой вариант (возможно не один) репозитория. Например: Git и Mercurial.

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

- Инициализация репозитория: Создайте новый репозиторий VCS для вашего проекта. В большинстве систем управления версиями это делается командой “init” или подобной.
- Добавление файлов: Добавьте все файлы вашего проекта в репозиторий. Это позволяет начать отслеживать изменения в ваших файлах.
- Создание коммитов: После внесения изменений в файлы сделайте коммиты, чтобы сохранить текущее состояние в репозитории. Коммиты могут включать описание ваших изменений.
- Откат к предыдущим версиям: Если вы сделали изменения, которые вы хотите откатить, VCS позволяет вам вернуться к предыдущим версиям файлов или к более старым коммитам.
- Просмотр истории: VCS сохраняет историю всех ваших коммитов. Вы можете просматривать историю, смотреть, какие изменения были внесены в каждом коммите.
- Ветвление и слияние: В случае необходимости, вы можете создавать ветки (branches) для разработки разных функций или экспериментов. Вы также можете сливать ветки, чтобы объединить изменения из разных веток.
- Работа с удаленным репозиторием: Даже при индивидуальной работе с VCS, вы можете создать удаленный репозиторий (например, на платформе GitHub или GitLab) для резервного копирования и совместного доступа к своему коду с разных устройств.
- Резервное копирование: Регулярно отправляйте свои изменения на удаленный репозиторий или делайте резервное копирование локальных репозитория для предотвращения потери данных.

#### **5. Опишите порядок работы с общим хранилищем VCS.**

- Выбор VCS и настройка: В начале проекта выберите VCS, который лучше всего соответствует вашим потребностям. Популярными системами управления версиями являются Git, SVN, Mercurial, и другие. Установите и настройте выбранную систему на вашем компьютере.
- Инициализация репозитория: Создайте новый репозиторий (хранилище) VCS для вашего проекта. Это можно сделать с помощью команды `init` или аналогичной в зависимости от выбранной системы.
- Добавление файлов: Добавьте все файлы вашего проекта в локальный репозиторий. В Git, это делается с помощью команды `git add`, в SVN - команды `svn add`, и так далее.
- Создание коммитов: После добавления файлов, создайте коммиты (snapshots) для сохранения текущего состояния вашего проекта в локальном репозитории. Коммиты должны сопровождаться описанием ваших изменений. В Git, коммиты создаются с помощью команды `git commit`, в SVN - `svn commit`.
- Работа с удаленным репозиторием: Если у вас есть общий проект с другими разработчиками, существует удаленный репозиторий на сервере. Вы можете клонировать (создать локальную копию) этого удаленного репозитория с помощью команды `git clone`, `svn checkout`, и других в зависимости от системы VCS.
- Работа над проектом: Вы и другие разработчики можете работать над проектом, внося изменения в файлы в вашей локальной копии. По мере работы, регулярно создавайте коммиты, чтобы сохранить изменения в вашем локальном репозитории.
- Отправка изменений: Когда вы готовы поделиться своими изменениями с другими участниками проекта, отправьте их на удаленный репозиторий с помощью команды `git push`, `svn commit`, или аналогичных команд в других системах.
- Обновление локальной копии: Для получения изменений, внесенных дру-

гими участниками проекта, используйте команду обновления (git pull, svn update и др.).

- Работа с ветками: При необходимости, создавайте и работайте с ветками (branches) для разработки разных функций. Ветвление и слияние (merge) - важные аспекты работы с VCS.
- Решение конфликтов: Если возникают конфликты при слиянии изменений, решайте их вручную и сохраняйте изменения в конфликтных файлах.
- Отслеживание истории: VCS сохраняет историю всех изменений в проекте. Вы можете просматривать историю, анализировать изменения и возвращаться к предыдущим версиям при необходимости.
- Резервное копирование: Регулярно сохраняйте резервные копии вашего локального репозитория и удаленного репозитория, чтобы избежать потери данных.

## **6. Каковы основные задачи, решаемые инструментальным средством git?**

Git — это система управления версиями. У Git две основных задачи:

- хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки
- обеспечение удобства командной работы над кодом.

## **7. Назовите и дайте краткую характеристику командам git.**

- Создание основного дерева репозитория:  
git init
- Получение обновлений (изменений) текущего дерева из центрального репозитория:  
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 встроенный 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. Приведите примеры использования при работе с локальным и удалённым репозиториями.

Работа с локальным репозиторием:

- `git init`: Инициализация нового локального репозитория.
- `git clone` : Клонирование удаленного репозитория в локальную директорию.
- `git status`: Показ текущего состояния локального репозитория, включая измененные, добавленные и неотслеживаемые файлы.
- `git add` : Добавление файла в индекс, подготовка к коммиту.
- `git commit -m "Сообщение коммита"`: Создание коммита с сообщением, описывающим внесенные изменения.
- `git log`: Просмотр истории коммитов в локальном репозитории.
- `git branch`: Просмотр списка веток в репозитории.
- `git checkout` : Переключение на другую ветку.
- `git merge` : Слияние изменений из указанной ветки в текущую.
- `git reset` : Откат на определенный коммит, отмена изменений.

Работа с удаленным репозиторием:

- `git remote -v`: Просмотр списка удаленных репозиториях, связанных с текущим локальным репозиторием.
- `git pull` : Получение изменений с удаленного репозитория и объединение их с текущей веткой.

- `git push` : Отправка своих локальных изменений на удаленный репозиторий.
- `git fetch` : Получение информации о состоянии удаленного репозитория без слияния изменений.
- `git clone` : Клонирование удаленного репозитория в локальную директорию.

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

Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций: чтобы не повредить основному проекту, создается новая ветка специально для экспериментов.

#### 10. Как и зачем можно игнорировать некоторые файлы при `commit`?

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.

# Выводы

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