

Лабораторная работа № 2

GitHub

Жукова София Викторовна

Содержание

Цель работы	5
Задание	6
Выполнение лабораторной работы	7
Выводы	20

Список иллюстраций

1	Установим git	7
2	Установим gh	7
3	Задаем имя и email владельца репозитория	7
4	Настраиваем utf-8 в выводе сообщений git	8
5	Задаем имя начальной ветки	8
6	Параметр autocrlf	8
7	Параметр safecrlf	8
8	Создадим ключи ssh	9
9	Генерируем ключ	9
10	Генерируем ключ	10
11	добавляем ключ	10
12	Копируем	11
13	New GPG key	11
14	вставим	12
15	укажем Git	13
16	авторизуемся	13
17	авторизуемся	13
18	Создадим шаблон	14
19	Перейдем	14
20	Удалим	15
21	Создадим	15
22	Отправим	15

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

Установка git (рис. [-@fig:001]).

```
[root@svzhukova ~]# dnf install git
Fedora 40 - aarch64 - Updates           i 31 kB/s | 16 kB    00:00
Последняя проверка окончания срока действия метаданных: 0:00:01 назад, Пт 28 фев 20
25 22:34:04.
Пакет git-2.48.1-1.fc40.aarch64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
[root@svzhukova ~]# dnf install g
```

Рис. 1: Установим git

Установка gh (рис. [-@fig:002]).

```
[root@svzhukova ~]# dnf install gh
Последняя проверка окончания срока действия метаданных: 0:00:23 назад, Сб 01 ма
25 11:21:40.
Пакет gh-2.65.0-1.fc40.aarch64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
```

Рис. 2: Установим gh

Базовая настройка git

Зададим имя и email владельца репозитория: (рис. [-@fig:003]).

```
[root@svzhukova ~]# git config --global user.name "Sofia Zhukova"
[root@svzhukova ~]# git config --global user.email "1032240966@pfur.ru"
```

Рис. 3: Задаем имя и email владельца репозитория

Настроим utf-8 в выводе сообщений git: (рис. [-@fig:004]).

```
[root@svzhukova ~]# git config --global core.quotepath false
```

Рис. 4: Настраиваем utf-8 в выводе сообщений git

Зададим имя начальной ветки (будем называть её master): (рис. [-@fig:005]).

```
[root@svzhukova ~]# git config --global init.defaultBranch master
```

Рис. 5: Задаем имя начальной ветки

Параметр autocrlf: (рис. [-@fig:006]).

```
[root@svzhukova ~]# git config --global core.autocrlf input~
```

Рис. 6: Параметр autocrlf

Параметр safecrlf: (рис. [-@fig:007]).

```
[root@svzhukova ~]# git config --global core.autocrlf input
```

```
[root@svzhukova ~]# git config --global core.safecrlf warn
```

Рис. 7: Параметр safecrlf

Создайте ключи ssh

по алгоритму rsa с ключём размером 4096 бит:

(рис. [-@fig:008]).

```
[root@svzhukova ~]# 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:M0VKsCM7HyjUx6lMmMH29pwsfXeTpBfJc6vj2Yz7p4 root@svzhukova
The key's randomart image is:
+---[RSA 4096]---+
| .. . . .
| o= . + o
| .+.+ * . .
| . oo* . . o .
| ..*=..S o * ..
| ..o*..o. * o .
| . . . . o o |
| *o.|
| *E=.|
+---[SHA256]---+
```

Рис. 8: Создадим ключи ssh

Создадим ключи pgp

Генерируем ключ (рис. [-@fig:009]).

```
[root@svzhukova ~]# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519): /home/svzhukova/.ssh
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/svzhukova/.ssh
Your public key has been saved in /home/svzhukova/.ssh.pub
The key fingerprint is:
SHA256:YPhn1ySeXfkXyxUWpUYPACWB1QteqMhR0FhwlZCBhY root@svzhukova
The key's randomart image is:
+--[ED25519 256]--+
| E**=o. ==o.o++|
| ...++ . + oo ..=|
| . . .+oo oo. =o|
| oo...+.. o...+
| . S o . oo|
| + . . |
| . |
| . |
+---[SHA256]---+
[root@svzhukova ~]#
```

Рис. 9: Генерируем ключ

Из предложенных опций выбираем: тип RSA and RSA; размер 4096; выберем срок действия; значение по умолчанию — 0 (срок действия не истекает никогда). GPG запросит личную информацию, которая сохранится в ключе: Комментарий.

Нажму клавишу ввода, чтобы оставить это поле пустым.

(рис. [-@fig:010]).

```
[root@svzhukova ~]# gpg --full-generate-key
gpg (GnuPG) 2.4.4; Copyright (C) 2024 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

Рис. 10: Генерируем ключ

Добавление PGP ключа в GitHub

Выводим список ключей и копируем отпечаток приватного ключа:(рис. [-@fig:011]).

```
gpg (GnuPG) 2.4.4; Copyright (C) 2024 g10 Code GmbH
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 '/home/svzhukova/.gnupg'
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
  <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: Sofia Zhukova
Адрес электронной почты: 1032240966@pfur.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Sofia Zhukova <1032240966@pfur.ru>"
```

Рис. 11: добавляем ключ

Скопируйте ваш сгенерированный PGP ключ в буфер обмена:(рис. [-@fig:012]).

```
[svzhukova@svzhukova ~]$ 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,
f, 1u
[keyboxd]
-----
sec  rsa4096/E88BB427F63B43DF 2025-03-01 [SC]
      C6F547C41BC9F9842C814A92E88BB427F63B43DF
uid  [ абсолютно ] Sofia Zhukova <1032240966@pfur.ru>
ssb  rsa4096/7E2B982747D85DD9 2025-03-01 [E]
```

Рис. 12: Копируем

Перейдем в настройки GitHub (<https://github.com/settings/keys>), нажмем на кнопку New GPG key и вставим полученный ключ в поле ввода.(рис. [-@fig:013]).

```
[svzhukova@svzhukova ~]$ gpg --armor --export 1032240966@pfur.ru xclip -sel clip
```

Рис. 13: New GPG key

(рис. [-@fig:014]).

-----BEGIN PGP PUBLIC KEY BLOCK-----

MQ==
=moGM
-----END PGP PUBLIC KEY BLOCK-----

Рис. 14: вставим

Настройка автоматических подписей коммитов git

Используя введённый email, укажем Git применять его при подписи коммитов: (рис. [-@fig:015]).

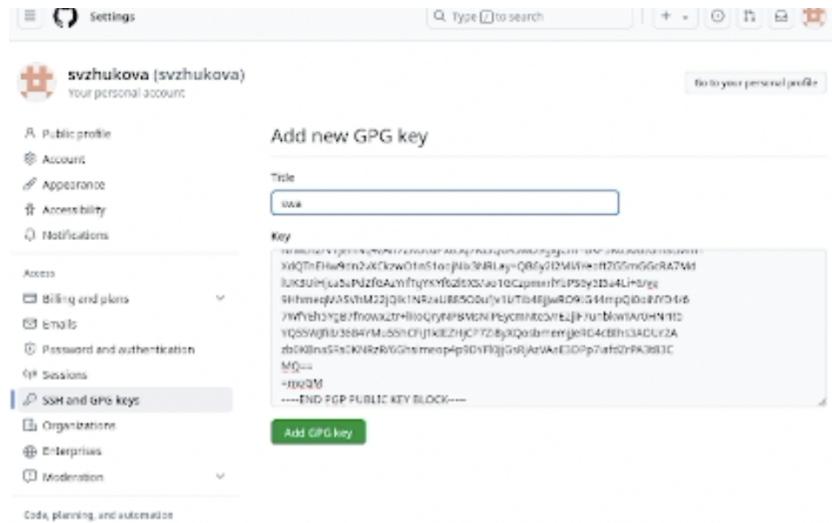


Рис. 15: укажем Git

Настройка gh

Для начала необходимо авторизоваться (рис. [-@fig:016]).

```
-----END PGP PUBLIC KEY BLOCK-----
[svzhukova@svzhukova ~]$ ^C
[svzhukova@svzhukova ~]$ git config --global user.signingkey 1032240966@pfur.ru
[svzhukova@svzhukova ~]$ git config --global commit.gpgsign true
[svzhukova@svzhukova ~]$ git config --global gpg.program $(which gpg2)
```

Рис. 16: авторизуемся

(рис. [-@fig:017]).

```
[svzhukova@svzhukova ~]$ gh auth login
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Generate a new SSH key to add to your GitHub account? No
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 4387-42E7
Press Enter to open https://github.com/login/device in your browser...
[]
```

Рис. 17: авторизуемся

Создание репозитория курса на основе шаблона

Создадим шаблон рабочего пространства (рис. [-@fig:018]).

```
[svzhukova@svzhukova ~]$ mkdir -p ~/work/study/2024-2025/"Операционные системы"
[svzhukova@svzhukova ~]$ cd ~/work/study/2024-2025/"Операционные системы"
[svzhukova@svzhukova Операционные системы]$ gh repo create study_2024-2025_os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository svzhukova/study_2024-2025_os-intro on GitHub
  https://github.com/svzhukova/study_2024-2025_os-intro
[svzhukova@svzhukova Операционные системы]$ git clone --recursive git@github.com:svzhukova/study_2024-2025_os-intro.git os-intro
Клонирование в «os-intro»...
hostkeys_find_by_key_hostfile: hostkeys_foreach failed for /home/svzhukova/.ssh/known_hosts: Not a directory
The authenticity of host 'github.com (140.82.121.4)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbZisF/zLDA0zPMSvHdkr4UvC0qU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Рис. 18: Создадим шаблон

Настройка каталога курса

Перейдем в каталог курса:(рис. [-@fig:019]).

```
[svzhukova@svzhukova ~]$ cd ~/work/study/2024-2025/"Операционные системы"/os-intro
```

Рис. 19: Перейдем

Удалим лишние файлы: (рис. [-@fig:020]).

```
[svzhukova@svzhukova ~]$ git clone --recursive git@github.com:svzhukova/study_2024  
-2025_os-intro.git os-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 | 215.00 КиБ/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 «/home/svzhukova/os-intro/template/presentation»...  
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 | 1015.00 КиБ/с, готово.  
Определение изменений: 100% (42/42), готово.  
Клонирование в «/home/svzhukova/os-intro/template/report»...  
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 | 1.62 МиБ/с, готово.  
Определение изменений: 100% (60/60), готово.  
Submodule path 'template/presentation': checked out 'c9b2712b4b2d431ad5086c9c72a02  
bd2fc1d4a6'  
Submodule path 'template/report': checked out 'c26e22effe7b3e0495707d82ef561ab185f  
5c748'  
[svzhukova@svzhukova ~]$
```

Рис. 20: Удалим

Создадим необходимые каталоги: (рис. [-@fig:021]).

```
[svzhukova@svzhukova os-intro]$ rm package.json
```

Рис. 21: Создадим

Отправим файлы на сервер: (рис. [-@fig:022]).

```
[svzhukova@svzhukova os-intro]$ echo os-intro > COURSE  
[svzhukova@svzhukova os-intro]$ make
```

Рис. 22: Отправим

Контрольные вопросы

- 1) Что такое системы контроля версий (VCS) и для решения каких задач они предназначают

Системы контроля версий (Version Control Systems, VCS) — это программные инструменты, которые помогают разработчикам отслеживать изменения в исходном коде и управлять различными версиями файлов. Они предназначены для решения следующих задач: Отслеживание изменений, Совместная работа, Управление версиями.

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

Хранилище: Это место, где сохраняются все версии проекта, включая историю изменений. Хранилище может быть локальным (на компьютере) или удаленным (на сервере). Commit: Это действие, при котором изменения в рабочей копии фиксируются в хранилище. Каждый commit содержит уникальный идентификатор (обычно хеш), описание изменений и может ссылаться на предыдущую версию. История: Это последовательность всех commit-ов, записанных в хранилище, с возможностью просмотра изменений, дат, авторов и описаний. Рабочая копия: Это локальное копирование проекта на компьютере разработчика, где он вносит изменения.

3) Что представляют собой и чем отличаются централизованные и децентрализованные VCS?

Централизованные VCS (CVCS): В этой модели имеется центральный сервер, содержащий все версии проекта. Разработчики работают с копией, загружая или обновляя её из центрального хранилища. Примеры CVCS: Subversion (SVN), CVS.

Децентрализованные VCS (DVCS): Каждое рабочее пространство разработчика является полным репозиторием, что позволяет работать локально и синхронизировать изменения с другими разработчиками, не зависимо от централизованного сервера. Примеры DVCS: Git, Mercurial.

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

При работе исключительно с хранилищем, разработчик может:

- Клонировать репозиторий для создания локальной рабочей копии.

- Вносить изменения в файлы.
- Использовать команды `add` для добавления изменений в индекс.
- Выполнять `commit` для записи изменений в локальное хранилище.
- Использовать `checkout` для переключения между версиями или ветками.
- Выполнять `push` для отправки изменений в удаленное хранилище, если оно используется.

5) Опишите порядок работы с общим хранилищем VCS.

1. Клонирование репозитория
 2. Внесение изменений в рабочей копии.
 3. Добавление (`add`) изменений в индекс.
 4. Фиксация изменений (`commit`) с описанием внесенных изменений.
 5. Получение последних изменений из общего хранилища (`pull`).
 6. Решение конфликтов, если они имеются.
 7. Отправка изменений на общий сервер (`push`).
- 6) Каковы основные задачи, решаемые инструментальным средством `git`?

Отслеживание изменений кода: Git регистрирует изменения, что позволяет всегда иметь доступ к предыдущим версиям. Совместная работа: Git позволяет нескольким разработчикам эффективно работать над одним проектом. Управление ветками: Легкое создание и использование веток для разработки новых функций или исправления ошибок. Слияние изменений: Хранит историю изменений и помогает объединять разные версии кода. 7) Назовите и дайте краткую характеристику командам `git`.

- `git init`: Инициализация нового репозитория.
 - `git clone`: Клонирование удаленного репозитория на локальную машину.
 - `git add`: Добавление изменений в индекс для последующего коммита.
 - `git commit`: Фиксация изменений в истории.
 - `git status`: Проверка статуса рабочей копии.
 - `git push`: Отправка локальных коммитов в удаленное хранилище.
 - `git pull`: Получение изменений из удаленного репозитория.
 - `git branch`: Управление ветками (создание, удаление, показ).
 - `git merge`: Объединение изменений из одной ветки в другую.
- 8) Приведите примеры использования при работе с локальным и удалённым репозиториями.

Локальный репозиторий: `git init` для создания нового репозитория; `git add .` и `git commit -m "Initial commit"` для фиксации изменений. Удаленный репозиторий: `git clone <url>` для клонирования репозитория; после изменения кода использовать `git add`, `git commit` и `git push` для загрузки изменений на сервер.

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

Ветви в Git позволяют создавать независимые линии разработки внутри одного репозитория. Исходный код может развиваться в разных направлениях (например, для новых функций или исправлений), без влияния на основной код. Ветвление позволяет: Упрощать параллельную разработку и слияние изменений после завершения работы над функцией, легко тестировать новые функции. Упрощать

10) Как и зачем можно игнорировать некоторые файлы при commit?

Иногда нужно исключить определённые файлы из коммитов, например, временные файлы, конфигурационные файлы среды или файлы сборки. Для этого создаётся файл `.gitignore`, в который добавляются паттерны файлов и директорий, которые Git будет игнорировать. Это позволяет избежать случайного добавления нежелательных файлов в систему контроля версий, сохраняя чистоту репозитория.

Выводы

Мы изучили идеологию и применение средств контроля версий и освоили умения по работе с git.