

Отчет по лабораторной работе №2

Первоначальная настройка git

Симонова Полина Игоревна

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
3.1	Установка программного обеспечения	6
3.2	Базовая настройка git	6
3.3	Создание ключа SSH	7
3.4	Создание ключа GPG	8
3.5	Регистрация на Github	10
3.6	Добавление ключа GPG в Github	10
3.7	Настроить подписи Git	12
3.8	Настройка gh	12
3.9	Создание репозитория курса на основе шаблона	13
4	Выводы	15
5	Ответы на контрольные вопросы.	16
6	Список литературы	19

Список иллюстраций

3.1	Установка git и gh	6
3.2	Задаю имя и email владельца репозитория	6
3.3	Настройка utf-8 в выводе сообщений git	7
3.4	Задаю имя начальной ветки	7
3.5	Задаю параметры autocrlf и safecrlf	7
3.6	Генерация ssh ключа по алгоритму rsa	7
3.7	Генерация ssh ключа по алгоритму ed25519	8
3.8	Генерация ключа	9
3.9	Защита ключа GPG	9
3.10	Аккаунт на Github	10
3.11	Вывод списка ключей	10
3.12	Копирование ключа в буфер обмена	11
3.13	Настройки GitHub	11
3.14	Добавление нового PGP ключа	11
3.15	Добавленный ключ GPG	11
3.16	Настройка подписей Git	12
3.17	Авторизация в gh	12
3.18	Завершение авторизации через браузер	12
3.19	Завершение авторизации	13
3.20	Создание репозитория	13
3.21	Перемещение между директориями	13
3.22	Удаление файлов и создание каталогов	14
3.23	Отправка файлов на сервер	14
3.24	Отправка файлов на сервер	14

1 Цель работы

Цель данной лабораторной работы – изучение идеологии и применения средств контроля версий, освоение умения по работе с git.

2 Задание

1. Создать базовую конфигурацию для работы с git
2. Создать ключ SSH
3. Создать ключ GPG
4. Настроить подписи Git
5. Зарегистрироваться на GitHub
6. Создать локальный каталог для выполнения заданий по предмету.

3 Выполнение лабораторной работы

3.1 Установка программного обеспечения

Устанавливаю необходимое программное обеспечение git и gh через терминал с помощью команд: `dnf install git` и `dnf install gh` (рис. fig. 3.1).

```
pisimonova@vbox:~$ sudo dnf -y install git
[sudo] пароль для pisimonova:
Последняя проверка окончания срока действия метаданных: 2:19:45 назад, Сб 01 мар 2025 16:35:02.
Пакет git-2.44.0-1.fc40.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
pisimonova@vbox:~$ sudo dnf -y install gh
Последняя проверка окончания срока действия метаданных: 2:20:08 назад, Сб 01 мар 2025 16:35:02.
Пакет gh-2.65.0-1.fc40.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
```

Рис. 3.1: Установка git и gh

3.2 Базовая настройка git

Задаю в качестве имени и email владельца репозитория свои имя, фамилию и электронную почту (рис. fig. 3.2).

```
pisimonova@vbox:~$ git config --global user.name "Polina Simonova"
pisimonova@vbox:~$
pisimonova@vbox:~$ git config --global user.email "burntorunn@gmail.com"
```

Рис. 3.2: Задаю имя и email владельца репозитория

Настраиваю utf-8 в выводе сообщений git для их корректного отображения (рис. fig. 3.3).

```
pisimonova@vbox:~$ git config --global core.quotepath false
```

Рис. 3.3: Настройка utf-8 в выводе сообщений git

Начальной ветке задаю имя master (рис. fig. 3.4).

```
pisimonova@vbox:~$ git config --global init.defaultBranch master
```

Рис. 3.4: Задаю имя начальной ветки

Задаю параметры autocrlf и safecrlf для корректного отображения конца строки (рис. fig. 3.5).

```
pisimonova@vbox:~$ git config --global core.autocrlf input
pisimonova@vbox:~$ git config --global core.safecrlf warn
```

Рис. 3.5: Задаю параметры autocrlf и safecrlf

3.3 Создание ключа SSH

Создаю ключ ssh размером 4096 бит по алгоритму rsa (рис. fig. 3.6).

```
pisimonova@vbox:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/pisimonova/.ssh/id_rsa):
Created directory '/home/pisimonova/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pisimonova/.ssh/id_rsa
Your public key has been saved in /home/pisimonova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:B2p834DMWCdrBIAUN0aJgTDXhDlA2noRDyLXIh001Ps pisimonova@vbox
The key's randomart image is:
+---[RSA 4096]-----+
| %B&@o.. |
| +%B* = . |
| o ==.. = . |
| . .o. B * |
| . . . = S o |
| . .Eo o o |
| . . . . |
| . . . . |
+-----[SHA256]-----+
```

Рис. 3.6: Генерация ssh ключа по алгоритму rsa

Создаю ключ ssh по алгоритму ed25519 (рис. fig. 3.7).

```

pisimonova@vbox:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/pisimonova/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pisimonova/.ssh/id_ed25519
Your public key has been saved in /home/pisimonova/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:V7Jc0ZVRdIO+Z749L0DSFoPkBwtb47u2swG7BTi90Zg pisimonova@vbox
The key's randomart image is:
+--[ED25519 256]--+
|      ..=...=0|
|      *+.oo..o|
|      . =.=o  |
|      o ..B+.  |
|      o S =+   |
|      + B ... o|
|      E = = .+ |
|      =.o .o.  |
|      . oo  o* |
+-----[SHA256]-----+

```

Рис. 3.7: Генерация ssh ключа по алгоритму ed25519

3.4 Создание ключа GPG

Генерирую ключ GPG, затем выбираю тип ключа RSA and RSA, задаю максимальную длину ключа: 4096, оставляю неограниченный срок действия ключа. Далее отвечаю на вопросы программы о личной информации (рис. fig. 3.8).


```

pisimonova@vbox:~$ 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.

gpg: создан каталог '/home/pisimonova/.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: Polina Simonova
Адрес электронной почты: burntorunn@gmail.com

```

Рис. 3.8: Генерация ключа

Ввожу фразу-пароль для защиты нового ключа (рис. fig. 3.9).

Рис. 3.9: Защита ключа GPG

3.5 Регистрация на Github

У меня уже был создан аккаунт на Github, соответственно, основные данные аккаунта я так же заполняла и проводила его настройку, поэтому просто вхожу в свой аккаунт (рис. fig. 3.10).

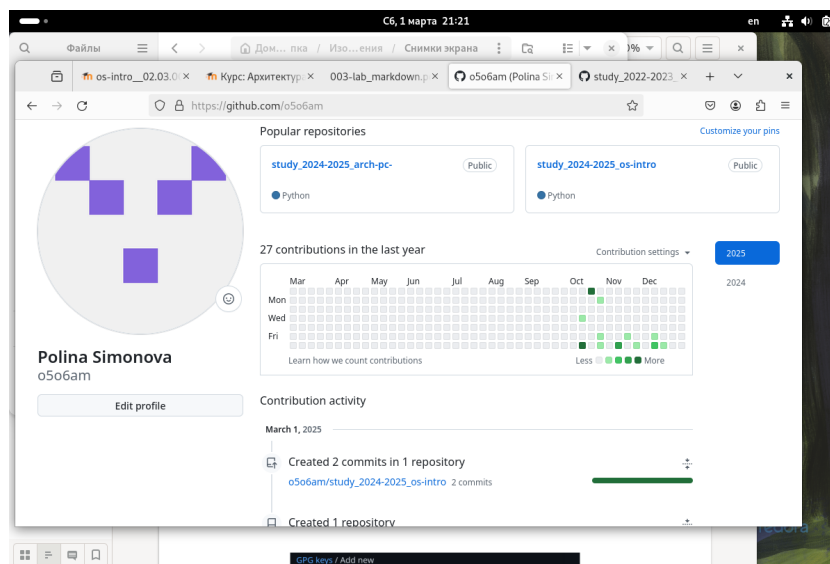


Рис. 3.10: Аккаунт на Github

3.6 Добавление ключа GPG в Github

Вывожу список созданных ключей в терминал, ищу в результате запроса отпечаток ключа (последовательность байтов для идентификации более длинного, по сравнению с самим отпечатком, ключа), он стоит после знака слеша, копирую его в буфер обмена (рис. fig. 3.11).

```
pisimonova@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: 1 подписанных: 0 доверие: 0-, 0q, 0n, 0m, 0f, 1u
[keyboard]
-----
sec   rsa4096/CEEE991BC504487D 2025-03-01 [SC]
      127FA2DB5EA4C15B32A15946CEEE991BC504487D
uid           [ абсолютно ] Polina Simonova <burntorunn@gmail.com>
ssb   rsa4096/828A2822AF101303 2025-03-01 [E]
```

Рис. 3.11: Вывод списка ключей

Ввожу в терминале команду, с помощью которой копирую сам ключ GPG в буфер обмена, за это отвечает утилита xclip (рис. fig. 3.12).

```
pisimonova@vbox: ~$ gpg --armor --export CEE991BC504487D | xclip -sel clip
```

Рис. 3.12: Копирование ключа в буфер обмена

Открываю настройки GitHub, ищу среди них добавление GPG ключа (рис. fig. 3.13).



Рис. 3.13: Настройки GitHub

Нажимаю на “New GPG key” и вставляю в поле ключ из буфера обмена (рис. fig. 3.14).



Рис. 3.14: Добавление нового PGP ключа

Я добавила ключ GPG на GitHub (рис. fig. 3.15).

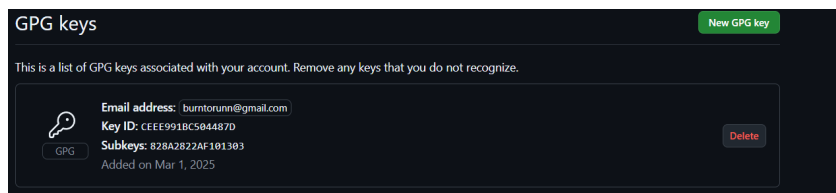


Рис. 3.15: Добавленный ключ GPG

3.7 Настроить подписи Git

Настраиваю автоматические подписи коммитов git: используя введенный ранее email, указываю git использовать его при создании подписей коммитов (рис. fig. 3.16).

```
pisimonova@vbox:~$ git config --global user.signingkey CEEE991BC504487D
pisimonova@vbox:~$ git config --global commit.gpgsign true
pisimonova@vbox:~$ git config --global gpg.program $(which gpg2)
```

Рис. 3.16: Настройка подписей Git

3.8 Настройка gh

Начинаю авторизацию в gh, отвечаю на наводящие вопросы от утилиты, в конце выбираю авторизоваться через браузер (рис. fig. 3.17).

```
pisimonova@vbox:~$ gh auth login
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? Skip
? How would you like to authenticate GitHub CLI? Login with a web browser
```

Рис. 3.17: Авторизация в gh

Завершаю авторизацию на сайте (рис. fig. 3.18).

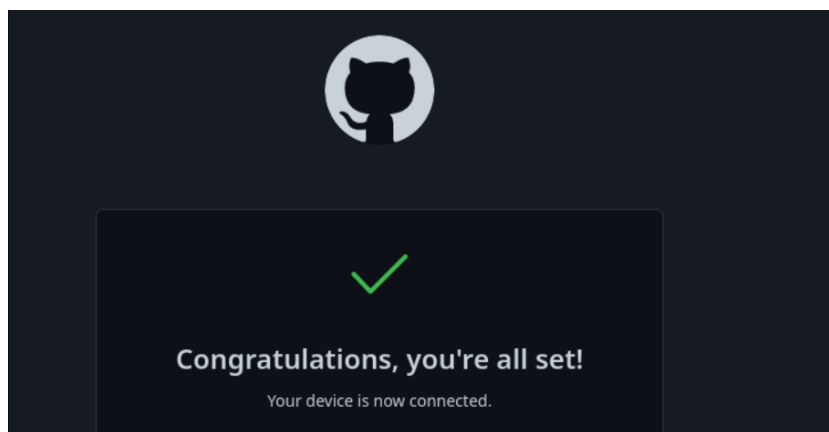


Рис. 3.18: Завершение авторизации через браузер

Видю сообщение о завершении авторизации под моим именем на гитхабе (рис. fig. 3.19).

```
✓ Authentication complete.  
- gh config set -h github.com git_protocol ssh  
✓ Configured git protocol  
✓ Logged in as o5o6am
```

Рис. 3.19: Завершение авторизации

3.9 Создание репозитория курса на основе шаблона

Сначала создаю директорию с помощью утилиты `mkdir` и флага `-p`, который позволяет установить каталоги на всем указанном пути. После этого с помощью утилиты `cd` перехожу в только что созданную директорию “Операционные системы”. Далее в терминале ввожу команду `gh repo create study_2022-2023_os-intro --template yamadharm/course-directory-student-trmplate --public`, чтобы создать репозиторий на основе шаблона репозитория. После этого клонирую репозиторий к себе в директорию, я указываю ссылку `ssh`. (рис. fig. 3.20).

```
pisimonova@vbox: $ mkdir -p ~/work/study/2024-2025/"Операционные системы"  
pisimonova@vbox: $ cd ~/work/study/2024-2025/"Операционные системы"  
pisimonova@vbox: ~/work/study/2024-2025/Операционные системы$ gh repo create study_2024-2025_os-intro --template=yamadharm/course-directory-student-template --public  
✓ Created repository o5o6am/study_2024-2025_os-intro on GitHub  
https://github.com/o5o6am/study_2024-2025_os-intro  
pisimonova@vbox: ~/work/study/2024-2025/Операционные системы$ git clone --recursive git@github.com:o5o6am/study_2024-2025_os-intro.git  
Клонирование в «study_2024-2025_os-intro»...
```

Рис. 3.20: Создание репозитория

Перехожу в каталог курса с помощью утилиты `cd`, проверяю содержание каталога с помощью утилиты `ls` (рис. fig. 3.21).

```
pisimonova@vbox: ~/work/study/2024-2025/Операционные системы$ cd os-intro
```

Рис. 3.21: Перемещение между директориями

Удаляю лишние файлы с помощью утилиты `rm`, далее создаю необходимые каталоги используя `makefile` (рис. fig. 3.22).

```

pislmonova@vbox:~/work/study/2024-2025/Операционные системы/os-intro$ rm package.json
pislmonova@vbox:~/work/study/2024-2025/Операционные системы/os-intro$ echo os-intro > COURSE
pislmonova@vbox:~/work/study/2024-2025/Операционные системы/os-intro$ make

```

Рис. 3.22: Удаление файлов и создание каталогов

Добавляю все новые файлы для отправки на сервер (сохраняю добавленные изменения) с помощью команды `git add` и комментирую их с помощью `git commit` (рис. fig. 3.23).

```

pislmonova@vbox:~/work/study/2024-2025/Операционные системы/os-intro$ git add .
pislmonova@vbox:~/work/study/2024-2025/Операционные системы/os-intro$ git commit -am 'feat(main): make course structure'
[master 1b8c141] feat(main): make course structure
405 files changed, 98413 insertions(+), 14 deletions(-)
create mode 100644 labs/README.md

```

Рис. 3.23: Отправка файлов на сервер

Отправляю файлы на сервер с помощью `git push`

```

pislmonova@vbox:~/work/study/2024-2025/Операционные системы/os-intro$ git push
Перечисление объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 342.31 КиБ | 9.01 МБ/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 https://github.com/o5o6am/study_2024-2025_os-intro.git
d9d7330..1b8c141 master -> master
pislmonova@vbox:~/work/study/2024-2025/Операционные системы/os-intro$ s

```

Рис. 3.24: Отправка файлов на сервер

4 Выводы

При выполнении данной лабораторной работы я изучила идеологию и применение средств контроля версий, освоила умение по работе с git.

5 Ответы на контрольные вопросы.

1. Системы контроля версий (VCS) - программное обеспечение для облегчения работы с изменяющейся информацией. Они позволяют хранить несколько версий изменяющейся информации, одного и того же документа, может предоставить доступ к более ранним версиям документа. Используется для работы нескольких человек над проектом, позволяет посмотреть, кто и когда внес какое-либо изменение и т. д. VCS применяются для: Хранения полной истории изменений, сохранения причин всех изменений, поиска причин изменений и совершивших изменение, совместной работы над проектами.
2. Хранилище – репозиторий, хранилище версий, в нем хранятся все документы, включая историю их изменения и прочей служебной информацией. commit – отслеживание изменений, сохраняет разницу в изменениях. История – хранит все изменения в проекте и позволяет при необходимости вернуться/обратиться к нужным данным. Рабочая копия – копия проекта, основанная на версии из хранилища, чаще всего последней версии.
3. Централизованные VCS (например: CVS, TFS, AccuRev) – одно основное хранилище всего проекта. Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет, затем добавляет изменения обратно в хранилище. Децентрализованные VCS (например: Git, Bazaar) – у каждого пользователя свой вариант репозитория (возможно несколько вариантов), есть возможность добавлять и забирать изменения из любого

репозитория. В отличие от классических, в распределенных (децентрализованных) системах контроля версий центральный репозиторий не является обязательным.

4. Сначала создается и подключается удаленный репозиторий, затем по мере изменения проекта эти изменения отправляются на сервер.
5. Участник проекта перед началом работы получает нужную ему версию проекта в хранилище, с помощью определенных команд, после внесения изменений пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются. К ним можно вернуться в любой момент.
6. Хранение информации о всех изменениях в вашем коде, обеспечение удобства командной работы над кодом.
7. 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. `git push -all` отправляем из локального репозитория все сохраненные изменения в центральный репозиторий, предварительно создав локальный репозиторий и сделав предварительную конфигурацию.
9. Ветвление - один из параллельных участков в одном хранилище, исходящих из одной версии, обычно есть главная ветка. Между ветками, т. е. их концами возможно их слияние. Используются для разработки новых функций.
10. Во время работы над проектом могут создаваться файлы, которые не следуют добавлять в репозиторий. Например, временные файлы. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл `.gitignore` с помощью сервисов.

6 Список литературы

1. Лабораторная работа № 2 [Электронный ресурс] URL: <https://esystem.rudn.ru/mod/page/view>