

Отчет по лабораторной работе №2

Операционные системы

Самарханова Полина Тимуровна

Содержание

1 Цель работы

Целью данной работы является изучение идеологии и применения средств контроля версий, а также освоение умений по работе с git.

2 Задания

1. Создание базовой конфигурации для работы с git
2. Создание SSH ключа
3. Создание GPG ключа
4. Настройка подписей git
5. Регистрация на Github
6. Создание локального каталога для выполнения заданий по предмету

3 Выполнение лабораторной работы

3.1 Установка ПО

Устанавливаю необходимое ПО git и gh через терминал с помощью команд `dnf install git`, `dnf install gh`

```
[spolina@fedora ~]$ sudo dnf -y install git
Fedora 38 - x86_64 - Updates          15 kB/s | 20 kB      00:01
Fedora 38 - x86_64 - Updates          517 kB/s | 3.7 MB    00:07
Fedora Modular 38 - x86_64 - Updates  54 kB/s | 20 kB     00:00
Пакет git-2.41.0-1.fc38.x86_64 уже установлен.
Зависимости разрешены.
Отсутствуют действия для выполнения.
Выполнено!
[spolina@fedora ~]$ sudo dnf -y install gh
Последняя проверка окончания срока действия метаданных: 0:00:29 назад, Чт 29 фев 2024 08:39:39.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий  Размер
=====
Установка:
gh          x86_64       2.36.0-1.fc38  updates      8.9 М
=====
Результат транзакции:
```

Рис. 3.1 Установка git и gh

3.2 Базовая настройка git

Задаю в качестве имени и почты владельца свои имя с фамилией и почту соответственно

```
[spolina@fedora ~]$ git config --global user.name "Polina Samarkhanova"
[spolina@fedora ~]$ git config --global user.email "spolina211105@icloud.com"
[spolina@fedora ~]$
```

Рис. 3.2 Задаю имя и email владельца репозитория

Настраиваю utf-8 в выводе сообщений git для их корректного отображения

```
[spolina@fedora ~]$ git config --global core.quotePath false
[spolina@fedora ~]$
```

Рис. 3.3 Настройка utf-8 в выводе сообщений git

Начальной ветке задаю имя master

```
[spolina@fedora ~]$ git config --global init.defaultBranch mas
[spolina@fedora ~]$
```

Рис. 3.4 Задаю имя начальной ветки

Задаю параметры `autocrlf` и `safecrlf` для корректного отображения конца строки

```
[spolina@fedora ~]$ git config --global core.autocrlf input
[spolina@fedora ~]$ git config --global core.safecrlf warn
[spolina@fedora ~]$
```

Рис. 3.5 Задаю параметры autocrlf и safecrlf

3.3 Создание ключа SSH

Создаю ключ SSH размером 4096 бит по алгоритму rsa

```
[spolina@fedora ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/spolina/.ssh/id_rsa):
/home/spolina/.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/spolina/.ssh/id_rsa
Your public key has been saved in /home/spolina/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:sFi5rEbXizgPWLCKSG6QU0f/hgPIyj0BgE/nn/XruTI spolina@fedora
The key's randomart image is:
+---[RSA 4096]-----+
|o . . |
|o..o. . |
|,+.o. + |
|.oo+. =. |
|o,+. =.*oS. |
|Bo,+. =o+ .. |
|=.o* + + . |
|. . .+ oE .. |
| . . ++. |
+---[SHA256]-----+
[spolina@fedora ~]$
```

Рис. 3.6 Генерация SSH ключа по алгоритму rsa

Создаю ключ SSH по алгоритму ed25519

```
[spolina@fedora ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/spolina/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/spolina/.ssh/id_ed25519
Your public key has been saved in /home/spolina/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:00KHA04kzmxFCjm43TSd05J52hwfZYSn2FHdxWDMah8 spolina@fedora
The key's randomart image is:
+---[ED25519 256]---+
| . . . +=+o+o|
| = + . = oo..+ o|
|. 0 = 0 +o.+ . |
| + 0 = 0.oo.o E |
|. . = S .. . . |
| . o . . |
|. o |
|. . |
+---[SHA256]-----+
[spolina@fedora ~]$
```

Рис. 3.7 Генерация SSH ключа по алгоритму ed25519

3.4 Создание ключа GPG

Генерирую ключ GPG, затем выбираю тип ключа RSA и RSA, длина ключа 4096 бит, срок действия - неограниченный. Затем отвечаю на запрашиваемую личную информацию

```
[spolina@fedora ~]$ 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 '/home/spolina/.gnupg'
gpg: создан щит с ключами '/home/spolina/.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
Срок действия ключа - 0 дней
```

Рис. 3.8 Генерация GPG ключа

3.5 Регистрация на GitHub

Аккаунт у меня уже был, поэтому просто вхожу в него

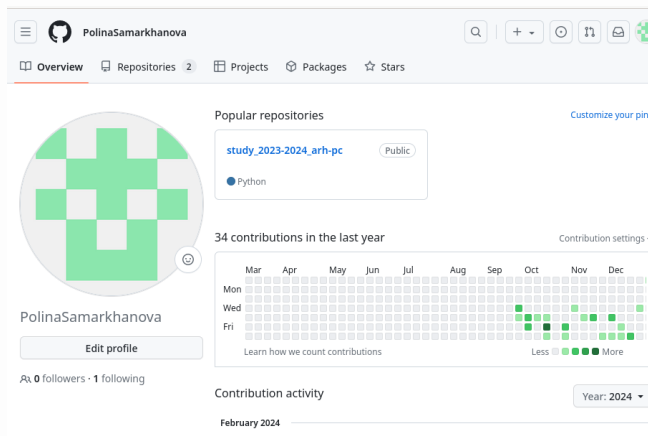


Рис. 3.9 Аккаунт на GitHub

3.6 Добавление ключа GPG в Github

Выводим список ключей и копируем в буфер обмена отпечаток приватного ключа (последовательность байтов, используемая для идентификации более длинного, по сравнению с самим отпечатком ключа), который находится после знака слеша

```
[spolina@fedora ~]$ 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/spolina/.gnupg/pubring.kbx
-----BEGIN PGP PRIVATE KEY BLOCK-----
sec  rsa4096/B5E85C70E953B3AA 2024-02-29 [SC]
      1B0EA244315FDF440CE71A7DB5E85C70E953B3AA
uid      [ абсолютно ] PolinaSamarkhanova <spolina211105@icloud.com>
ssb  rsa4096/522E7AA1D9DB29A2 2024-02-29 [E]
[spolina@fedora ~]$
```

Рис. 3.10 Вывод списка ключей

Ввожу в терминале команду `gpg --armor --export <КЛЮЧ> | xclip -sel clip` С ее помощью копирую сам ключ в буфер обмена

```
[spolina@fedora ~]$ gpg --armor --export B5E85C70E953B3AA | xclip -sel clip
```

Рис. 3.11 Копирование ключа в буфер обмена

Далее открываю настройки Github и ищу раздел GPG ключей

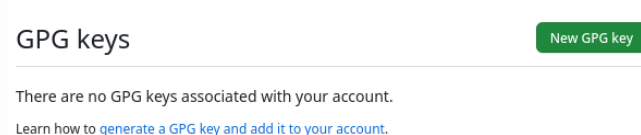


Рис. 3.12 Настройка Github

Нажимаю кнопку New GPG кеуи вставляю в нужное поле скопированный ключ

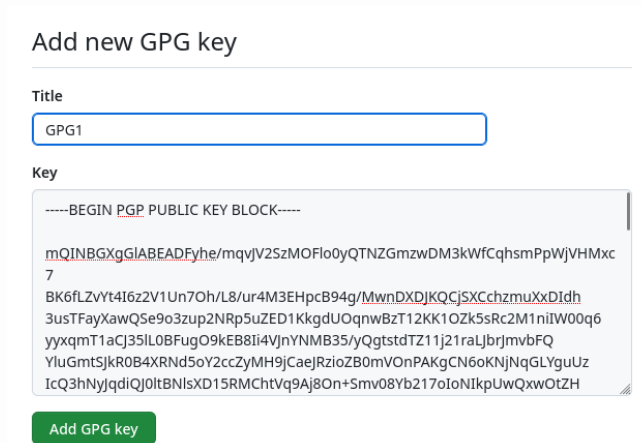


Рис. 3.13 Добавление ключа

Теперь можно увидеть добавленный на Github ключ GPG

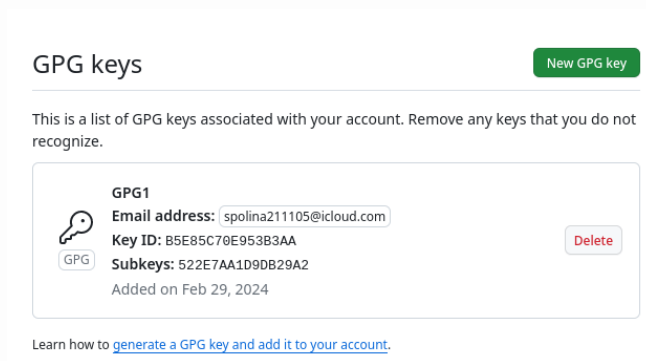


Рис. 3.14 Добавленный ключ GPG

Далее настраиваю автоматические подписи коммитов Git

```
[spolina@fedora ~]$ git config --global user.signingkey B5E85C70E953B3AA
[spolina@fedora ~]$ git config --global commit.gpgsign true
[spolina@fedora ~]$ git config --global gpg.program $(which gpg2)
[spolina@fedora ~]$
```

Рис. 3.15 Настройка подписей Git

3.7 Настройка gh

Начинаю с авторизации в gh: нужно было ответить на вопросы утилиты и в конце выбрать авторизацию через браузер.

```
[spolina@fedora ~]$ 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: EA6F-8381
Press Enter to open github.com in your browser...
```

Рис. 3.16 Авторизация в gh

Скопировав 8-значный код и вставив его в открывшемся окне браузера, завершаю авторизацию

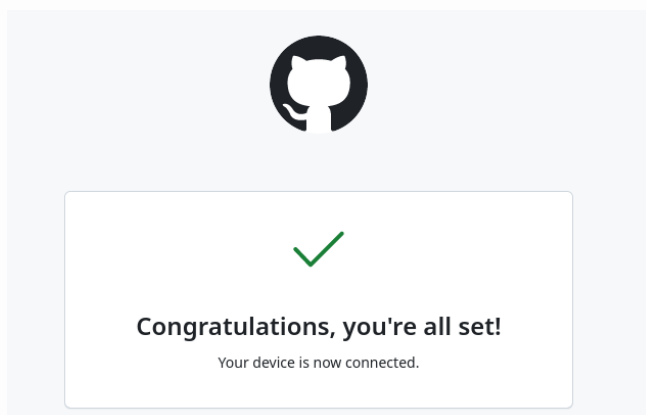


Рис. 3.17 Завершение авторизации через браузер

Теперь можно увидеть завершение авторизации и в терминале

```
✓ Authentication complete.  
- gh config set -h github.com git_protocol https  
✓ Configured git protocol  
✓ Logged in as PolinaSamarkhanova  
[spolina@fedora ~]$
```

Рис. 3.18 Завершение авторизации

3.8 Сознание репозитория курса на основе шаблона

Сначала я создала новую папку с помощью утилиты mkdir: `mkdir -p ~/work/study/2023-2024/“Операционные системы”` Далее я перешла в саму папку Операционные системы через утилиту cd Затем я в терминале ввела команду: `gh repo create study_2023-2024_os-intro --template=yamadharm/course-directory-student-template --public` Она нужна для того, чтобы создать репозиторий на основе указанного шаблона после этого я клонировала репозиторий к себе в папку, но указала ссылку с протоколом https, а не ssh

```
[spolina@fedora Операционные системы]$ git clone --recursive https://github.com/  
PolinaSamarkhanova/study_2023-2024_os-intro.git os-intro  
Клонирование в «os-intro»...  
remote: Enumerating objects: 32, done.  
remote: Counting objects: 100% (32/32), done.  
remote: Compressing objects: 100% (31/31), done.  
remote: Total 32 (delta 1), reused 18 (delta 0), pack-reused 0  
Получение объектов: 100% (32/32), 18.60 КиБ | 307.00 КиБ/с, готово.  
Определение изменений: 100% (1/1), готово.  
Подмодуль «template/presentation» (https://github.com/yamadharm/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»  
Подмодуль «template/report» (https://github.com/yamadharm/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»  
Клонирование в «/home/spolina/work/study/2023-2024/Операционные системы/os-intro/template/presentation»...  
remote: Enumerating objects: 95, done.  
remote: Counting objects: 100% (95/95), done.  
remote: Compressing objects: 100% (67/67), done.  
remote: Total 95 (delta 34), reused 87 (delta 26), pack-reused 0
```

Рис. 3.19 Создание репозитория

Перехожу в каталог курса с помощью утилиты cd, затем с помощью ls проверяю содержимое каталога

```
[spolina@fedora Операционные системы]$ cd os-intro  
[spolina@fedora os-intro]$ ls  
CHANGELOG.md  COURSE  Makefile  README.en.md  README.md  
config        LICENSE  package.json  README.git-flow.md  template  
[spolina@fedora os-intro]$
```

Рис. 3.20 Перемещение в нужную директорию

Удаляю лишние файлы с помощью команды rm, затем создаю необходимые каталоги, используя makefile, а именно: `make list make prepare make submodule`

```
[spolina@fedora os-intro]$ rm package.json
[spolina@fedora os-intro]$ echo os-intro > COURSE
[spolina@fedora os-intro]$ make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare       Generate directories structure
  submodule     Update submules
[spolina@fedora os-intro]$
```

Рис. 3.21 Удаление файлов и создание каталогов

Добавляю все новые файлы для отправки на сервер с помощью git add и комментирую при помощи git commit

```
[spolina@fedora os-intro]$ git add .
[spolina@fedora os-intro]$ git commit -am 'feat(master): make course structure'
[master fe8194f] feat(master): make course structure
359 files changed, 98412 insertions(+)
create mode 100644 labs/README.md
create mode 100644 labs/README.ru.md
create mode 100644 labs/lab01/presentation/Makefile
create mode 100644 labs/lab01/presentation/image/kulyabov.jpg
create mode 100644 labs/lab01/presentation/presentation.md
create mode 100644 labs/lab01/report/Makefile
```

Рис. 3.22 Отправка файлов на сервер

Отправляю файлы на сервер с помощью git push

```
[spolina@fedora os-intro]$ git push
Перечисление объектов: 39, готово.
Подсчет объектов: 100% (39/39), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 342.09 КиБ | 2.30 МБ/с, готово.
Всего 38 (изменений 4), повторно использовано 1 (изменений 0), повторно использо
вано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To https://github.com/PolinaSamarkhanova/study_2023-2024_os-intro.git
 eaca73d..fe8194f master -> master
[spolina@fedora os-intro]$
```

Рис. 3.23 Отправка файлов на сервер

4 Выводы

При выполнении данной лабораторной работы я идеологию и применение средств контроля версий, а также освоила навыки 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`

1. `git push -all` отправляем из локального репозитория все сохраненные изменения в центральный репозиторий, предварительно создав локальный репозиторий и сделав предварительную конфигурацию.
2. 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.
3. 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.

Список литературы

1. Лабораторная работа №2 (электронный ресурс) URL:<https://esystem.rudn.ru/mod/page/view.php?id=1098790>