

Лабораторная работа №2

Операционные системы

Софич Андрей Геннадьевич

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
3.1	Создание базовой конфигурации для работы с git	6
3.2	Настройка подписи Git	10
3.3	Авторизация на GitHub	11
4	Выводы	16

Список иллюстраций

3.1	Загрузка пакетов	6
3.2	Создание базовой конфигурации	6
3.3	Создание ssh ключа	7
3.4	Создание ssh ключа	7
3.5	Создание ssh ключа	8
3.6	Создание ключа GPG	9
3.7	Создание ключа GPG	9
3.8	Копирование ключа GPG	10
3.9	Добавление GPG	10
3.10	Настройка подписей	10
3.11	Авторизация	11
3.12	Создание директории	12
3.13	Создание репозитория	12
3.14	Клонирование репозитория	12
3.15	Проверка репозитория	12
3.16	Удаление файлов и создание каталогов	12
3.17	Загрузка	13

1 Цель работы

Цель работы- изучение и применения средств контроля версии, освоение умения по работе с git.

2 Задание

1. Создание базовой конфигурации для работы с git.
2. Создание ключа SSH
3. Создание ключа GPG
4. Настройка подписи Git
5. Авторизация на GitHub
6. Создание репозитория
7. Ответы на вопросы

3 Выполнение лабораторной работы

3.1 Создание базовой конфигурации для работы с git

Устанавливаю программное обеспечение git и gh через терминал (рис. 3.1).

```
[sofich132237371@fedora ~]$ sudo dnf -y install git
Последняя проверка окончания срока действия метаданных: 3:26:08 назад, Пн 26 фев 2024 14:55:24.
Пакет git-2.43.2-1.fc39.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
[sofich132237371@fedora ~]$ sudo dnf -y install gh
Последняя проверка окончания срока действия метаданных: 3:26:18 назад, Пн 26 фев 2024 14:55:24.
Пакет gh-2.43.1-1.fc39.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
```

Рис. 3.1: Загрузка пакетов

Задаю имя и email владельца репозитория, настраиваю utf-8 для корректного отображения сообщения git. Задаю имя ветки. Задаю параметры autocrlf и safecrlf (рис. 3.1).

```
[sofich132237371@fedora ~]$ git config --global user.name "AndreySofich"
[sofich132237371@fedora ~]$ git config --global user.email "andrejsofic2@gmail.com"
[sofich132237371@fedora ~]$ git config --global core.quotepath false
[sofich132237371@fedora ~]$ git config --global init.defaultBranch master
[sofich132237371@fedora ~]$ git config --global core.autocrlf input
[sofich132237371@fedora ~]$ git config --global core.safecrlf warn
```

Рис. 3.2: Создание базовой конфигурации

##Создание ключа ssh

Создаю ключ ssh по алгоритму ed25519 (рис. 3.3).

```
[sofich132237371@fedora ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/sofich132237371/.ssh/id_ed25519):
/home/sofich132237371/.ssh/id_ed25519 already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/sofich132237371/.ssh/id_ed25519
Your public key has been saved in /home/sofich132237371/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:FaDRcSU9+BLx0J0CCEDnXvMlyigdM0bxN0Ltkrxgmw8 sofich132237371@fedora
The key's randomart image is:
+--[ED25519 256]--+
|      .+..+..+      |
|    .oBo- +==o      |
|  .+o.o.+o==o      |
|  . . .o==+++..      |
| o . .S+oo ^        |
|  . .E . . + .      |
|    . o o .          |
|      . +           |
|      .              |
+-----[SHA256]-----+
```

Рис. 3.3: Создание ssh ключа

Создаю ключ ssh по алгоритму rsa (рис. 3.4).

```
[sofich132237371@fedora ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/sofich132237371/.ssh/id_rsa):
/home/sofich132237371/.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/sofich132237371/.ssh/id_rsa
Your public key has been saved in /home/sofich132237371/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:lyVtyDd/iQdBNOKjV48EbfZsj1v2Ns1f/sTJ2By3Jpg sofich132237371@fedora
The key's randomart image is:
+---[RSA 4096]-----+
|      o+==          |
|      ..oo+o        |
|    +oo+o..         |
|      .+^o+==       |
|    S.oE +ooo       |
|      .. oo=        |
|      . .^+         |
|      o.o+          |
|      oo+           |
|      .              |
+-----[SHA256]-----+
[sofich132237371@fedora ~]$
```

Рис. 3.4: Создание ssh ключа

Добавляю ключ в репозиторий на GitHub (рис. 3.5).

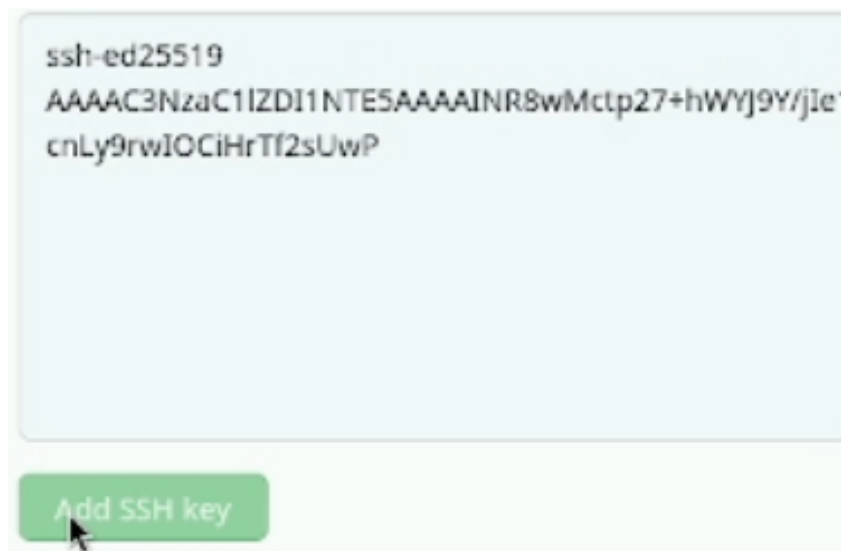


Рис. 3.5: Создание ssh ключа

##Создание ключа GPG

Генерирую ключ GPG, тип ключа RSA, задаю максимальную длину ключа 4096 и ставлю неограниченный срок действия ключа (рис. 3.6).


```
[sofich132237371@fedora ~]$ gpg --full-generate-key
gpg (GnuPG) 2.4.3; Copyright (C) 2023 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.

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.
```

Рис. 3.6: Создание ключа GPG

Вывожу список созданных ключей в терминал, ищу отпечаток ключа и копирую его {#fig:007 width=70%}

```
[sofich132237371@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: 2 подписанных: 0 доверие: 0-, 0q, 0
n, 0m, 0f, 2u
[keyboard]
-----
sec rsa4096/472EF7FFBEC65850 2024-02-26 [SC]
    22F347649CFA900D7C3858BE472EF7FFBEC65850
uid [ абсолютно ] SofichAndrey <132237371@pfur.ru>
ssb rsa4096/18F3E8F707B2F234 2024-02-26 [E]

sec rsa4096/8233EAE8099FD390 2024-02-26 [SC]
    434B70EA548693A82E0ECED1B233EAE8099FD390
uid [ абсолютно ] AndreySofich <andreysofic2@gmail.com>
ssb rsa4096/ABF2AB1823E59ED9 2024-02-26 [E]
```

Рис. 3.7: Создание ключа GPG

Копирую ключ {#fig:008 width=70%}

```
[sofich132237371@fedora ~]$ gpg --armor --export 472EF7FFBEC65858
```

Рис. 3.8: Копирование ключа GPG

Добавляю ключ GPG на GitHub {#fig:009 width=70%}

Рис. 3.9: Добавление GPG

3.2 Настройка подписи Git

Настраиваю автоматические подписи коммитов git: используя введенный ранее email {#fig:010 width=70%}

```
[sofich132237371@fedora ~]$ git config --global user.signingkey 472EF7FFBEC65858
[sofich132237371@fedora ~]$ git config --global commit.gpgsign true
[sofich132237371@fedora ~]$ git config --global gpg.program $(which gpg2)
[sofich132237371@fedora ~]$ gh auth login
```

Рис. 3.10: Настройка подписей

3.3 Авторизация на GitHub

Завершаю авторизацию на сайте через браузер {#fig:011 width=70%}

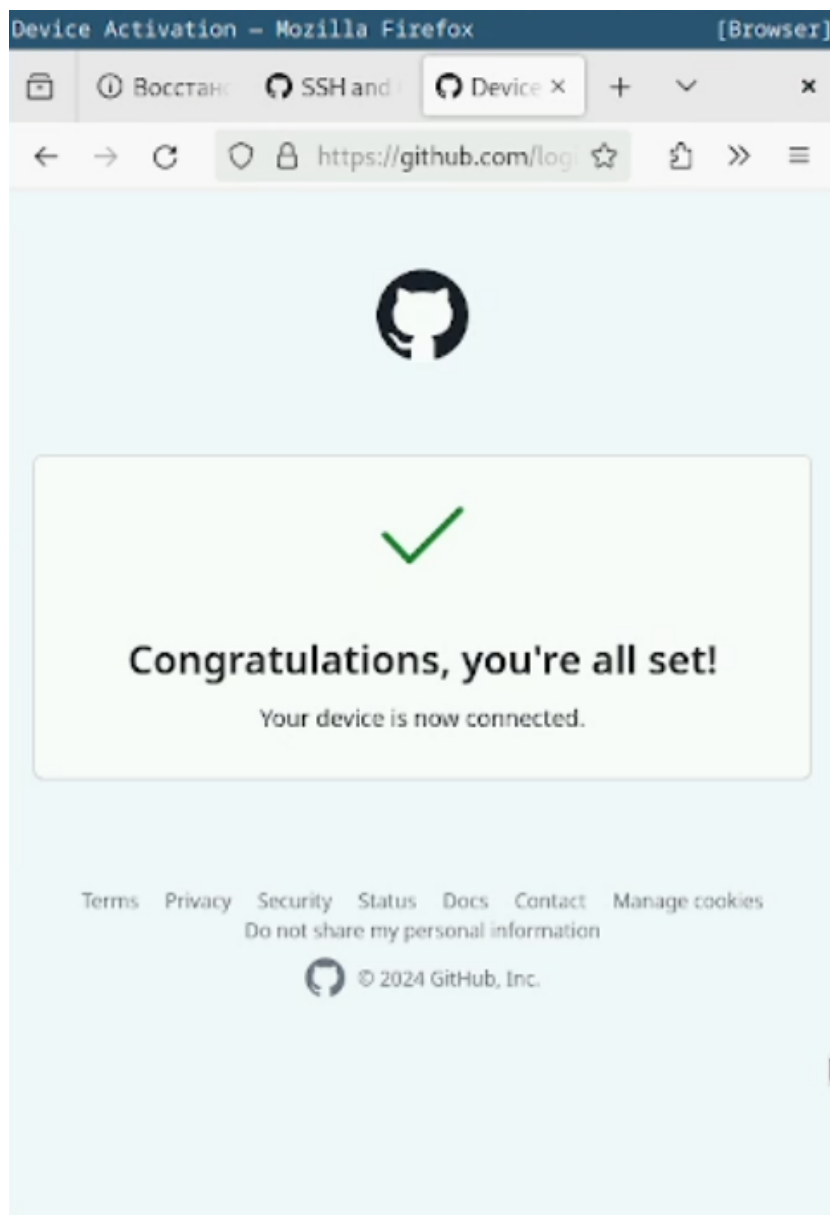


Рис. 3.11: Авторизация

##Создание репозитория

Перехожу в созданную директорию {#fig:012 width=70%}

```
[sofich132237371@fedora ~]$ mkdir -p ~/work/study/2023-2024/'05'
[sofich132237371@fedora ~]$ cd ~/work/study/2023-2024/'05'
```

Рис. 3.12: Создание директории

Создаю репозиторий на основе шаблона {#fig:013 width=70%}

```
[sofich132237371@fedora 05]$ gh repo create study_2023-2024_os-intro --template=yamadharma/course-directory-student-template --p
ublic
Created repository AndreySofich/study_2023-2024_os-intro on GitHub
https://github.com/AndreySofich/study_2023-2024_os-intro
```

Рис. 3.13: Создание репозитория

Клонирую репозиторий в свою директорию {#fig:014 width=70%}

```
[sofich132237371@fedora 05]$ git clone --recursive https://github.com/AndreySofich/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), 10.59 KiB | 1002.00 KiB/c, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по пу
ти «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «templa
te/report»
Клонирование в «/home/sofich132237371/work/study/2023-2024/05/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
Получение объектов: 100% (95/95), 96.99 KiB | 886.80 KiB/c, готово.
Определение изменений: 100% (34/34), готово.
Клонирование в «/home/sofich132237371/work/study/2023-2024/05/os-intro/template/report»...
remote: Enumerating objects: 126, done.
remote: Counting objects: 100% (126/126), done.
remote: Compressing objects: 100% (87/87), done.
remote: Total 126 (delta 52), reused 108 (delta 34), pack-reused 0
Получение объектов: 100% (126/126), 335.80 KiB | 1.48 MiB/c, готово.
Определение изменений: 100% (52/52), готово.
Submodule path 'template/presentation': checked out '40a1761813e197d0e84434ff1c72c6b304f24c'
Submodule path 'template/report': checked out '7c31ab0e50f08c0d2d67cae8a19ef802ced8be'
```

Рис. 3.14: Клонирование репозитория

Перехожу в каталог и проверяю наличие нужных папок {#fig:015 width=70%}

```
[sofich132237371@fedora 05]$ cd os-intro
[sofich132237371@fedora os-intro]$ ls
CHANGELOG.md  config  COURSE  LICENSE  Makefile  package.json  README.en.md  README.git-flow.md  README.md  template
[sofich132237371@fedora os-intro]$
```

Рис. 3.15: Проверка репозитория

Удаляю ненужные файлы и создаю необходимые {#fig:016 width=70%}

```
[sofich132237371@fedora os-intro]$ rm package.json
[sofich132237371@fedora os-intro]$ echo os-intro > COURSE
[sofich132237371@fedora os-intro]$ make
```

Рис. 3.16: Удаление файлов и создание каталогов

Загружаю всё на GitHub {#fig:016 width=70%}

```
[sofich132237371@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), 342.87 Киб | 11.40 Миб/с, готово.
Всего 38 (изменений 4), повторно использовано 1 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To https://github.com/AndreySofich/study_2023-2024_os-intro.git
0777e3a..4701257 master -> master
```

Рис. 3.17: Загрузка

Ответы на 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.

4 Выводы

При выполнении работы я изучил применение средств контроля версий и освоил умение по работе с git