

Отчёта по лабораторной работе №2

Операционные системы

Сахно Алёна Юрьевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	10
	Список литературы	16

Список иллюстраций

4.1	Установка git & gh	10
4.2	Установка git & gh	10
4.3	Бфзовая настройка git	11
4.4	Создание ключа ssh	11
4.5	Создание ключа ssh & pgr	12
4.6	Добавления pgr в GitHub	13
4.7	Настройка автоматических подписей коммитетов git и вывод ключа SSH	13
4.8	GitHub ключи	13
4.9	Настрока gh	14
4.10	Настройка для рабочего прстранства	14
4.11	Настройка каталога курса	15

Список таблиц

1 Цель работы

Изучить идеологию и применение средств контроля версий.

Освоить умения по работе с `git`.

2 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.

3 Теоретическое введение

Примеры использования git

Система контроля версий Git представляет собой набор программ командной строки. Доступ к репозиториям осуществляется по протоколу ssh. Благодаря тому, что Git является распределённой системой контроля версий, резервную копию репозитория можно хранить на любом компьютере.

Стандартные процедуры работы при наличии центрального репозитория

Работа пользователя со своей веткой начинается с проверки и получения изменений из центрального репозитория.

```
git checkout master
git pull
git checkout -b имя_ветки
```

Работа с локальным репозиторием

Создадим локальный репозиторий.

Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория.

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

Работа с сервером репозиториев

Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать ключи.

```
ssh-keygen -C "Имя Фамилия <work@mail>"
```

Ключи сохраняться в каталоге ~/.ssh/.

Базовая настройка git

Первичная настройка параметров git

Зададим имя и email владельца репозитория:

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

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

Настроим utf-8 в выводе сообщений git:

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

Настройте верификацию и подписание коммитов git.

Зададим имя начальной ветки (будем называть её master):

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

Учёт переносов строк

В разных операционных системах приняты разные символы для перевода строк:

Windows: \r\n (CR и LF);

Unix: \n (LF);

Mac: \r (CR).

Верификация коммитов с помощью PGP

Как настроить PGP-подпись коммитов с помощью gpg.

Общая информация

Коммиты имеют следующие свойства:

- author (автор) – контрибьютор, выполнивший работу (указывается для справки);
- committer (коммитер) – пользователь, который закоммитил изменения.

Эти свойства можно переопределить при совершении коммита.

Авторство коммита можно подделать.

В git есть функция подписи коммитов.

Для подписывания коммитов используется технология PGP (см. Работа с PGP).

Подпись коммита позволяет удостовериться в том, кто является коммитером. Авторство не

Проверка коммитов в Git

GitHub и GitLab будут показывать значок Verified рядом с вашими новыми коммитами.

Режим бдительности (vigilant mode)

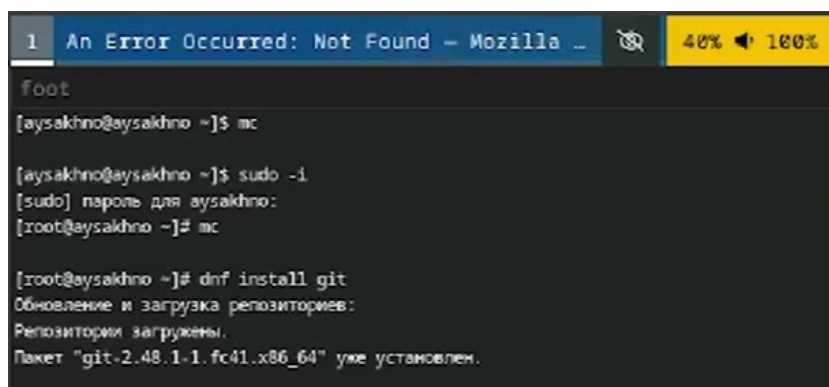
На GitHub есть настройка vigilant mode.

Все неподписанные коммиты будут явно помечены как Unverified.

Включается это в настройках в разделе SSH and GPG keys. Установите метку на Flag unsig

4 Выполнение лабораторной работы

Для начала переходим в супер пользователь и устанавливаем git, а также производим установку gh (рис.1 4.1).



```
1 An Error Occurred: Not Found - Mozilla ... 40% 100%
foot
[aysakhno@aysakhno ~]$ mc

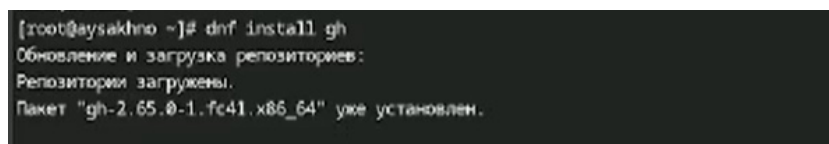
[aysakhno@aysakhno ~]$ sudo -i
[sudo] пароль для aysakhno:
[root@aysakhno ~]$ mc

[root@aysakhno ~]$ dnf install git
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет "git-2.48.1-1.fc41.x86_64" уже установлен.

[root@aysakhno ~]$ dnf install gh
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет "gh-2.65.0-1.fc41.x86_64" уже установлен.
```

Рис. 4.1: Установка git & gh

(рис.2 4.2).



```
[root@aysakhno ~]$ dnf install gh
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет "gh-2.65.0-1.fc41.x86_64" уже установлен.
```

Рис. 4.2: Установка git & gh

После чего переходим к разделу Базовая настройка git 1. задаем наше имя и email репозитория 2. Настроим utf.8, задаем имя начальной ветки (рис.3 4.3).

```
[root@aysakhno ~]# git config --global user.name "Asakhno"
[root@aysakhno ~]# git config --global user.email "saxno.06@icloud.com"
[root@aysakhno ~]# git config --global core.quotepath false
[root@aysakhno ~]# git config --global init.defaultBranch master
[root@aysakhno ~]# git config --global core.safecrlf warn
```

Рис. 4.3: Бфзовая настройка git

Создаем ключи ssh По алгоритму rsa с ключем размером 4096 и по алгоритму ed25519

(рис.4 4.4).

```
[root@aysakhno ~]# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): /root/.ssh/id_rsa
Enter passphrase for "/root/.ssh/id_rsa" (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:MerTep8sobnmCXbBslfa1EkakFlN468VoD3zjYxGow root@aysakhno
The key's randomart image is:
+---[RSA 4096]-----+
| .oo=0 |
| .+... |
| .+ = E o |
| . + + .. o |
| + + 0. S |
| + = . = . |
| .+o*oo.. |
| .oo+=+o. |
| .oo=0 0+ |
+---[SHA256]-----+
[root@aysakhno ~]# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519): /root/.ssh/id_ed25519
Enter passphrase for "/root/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519
Your public key has been saved in /root/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:PeuwwcPJInBFfBLvXJHGyWPS4h8a1/hMS/b4PVG7wI root@aysakhno
The key's randomart image is:
```

Рис. 4.4: Создание ключа ssh

И переходим к созданию ключа pgr генерируя ключ, из предложенных опций выбираем : 1. тип RSA and RSA 2. размер 4096 3. срок действия 0 4. GPG запросит личную информацию , которая сохранится в ключе

(рис.5 4.5).

```
foot
The key fingerprint is:
SHA256:PeVwWcPJ7hBF7BLvXJHGyWPS4h2a1/iMS/b4PV67wI root@aysakhno
The key's randomart image is:
+--[ED25519 256]--+
|      o+... . + . |
|      ooo.= . X |
|      o .o= o 0 B |
|      . . o . B 0= |
|      S = E =.= |
|      = B o o |
|      . 0 . . . |
|      + o . . |
|      o |
+-----[SHA256]-----+
[root@aysakhno ~]# gpg --full-generate-key
gpg (GnuPG) 2.4.5; 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.

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
```

Рис. 4.5: Создание ключа ssh & gpg

Затем переходим к добавления PGP ключа в GitHub. Выполняя команды выводим список ключей, после чего на экран выйдет сам ключ, мы должны его скопировать и перенести в ГитХабе. SSH добавляется в GitHub таким же образом как и pgr. Настройка автоматических подписей коммитов git. Используя введенную почту, указываем git применять его при подписи коммитов (рис.6 4.6).

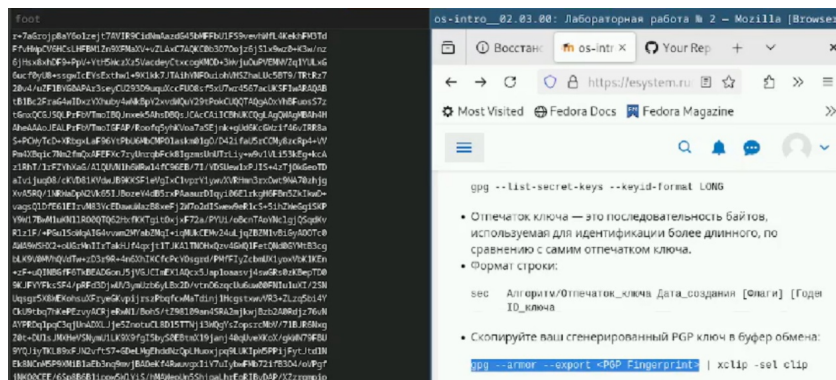


Рис. 4.6: Добавления ргр в GitHub

(рис.7 4.7).

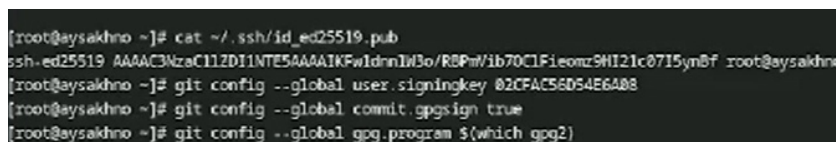


Рис. 4.7: Настройка автоматических подписей коммитов git и вывод ключа SSH

В GitHub отображения созданных ключей

(рис.8 4.7).

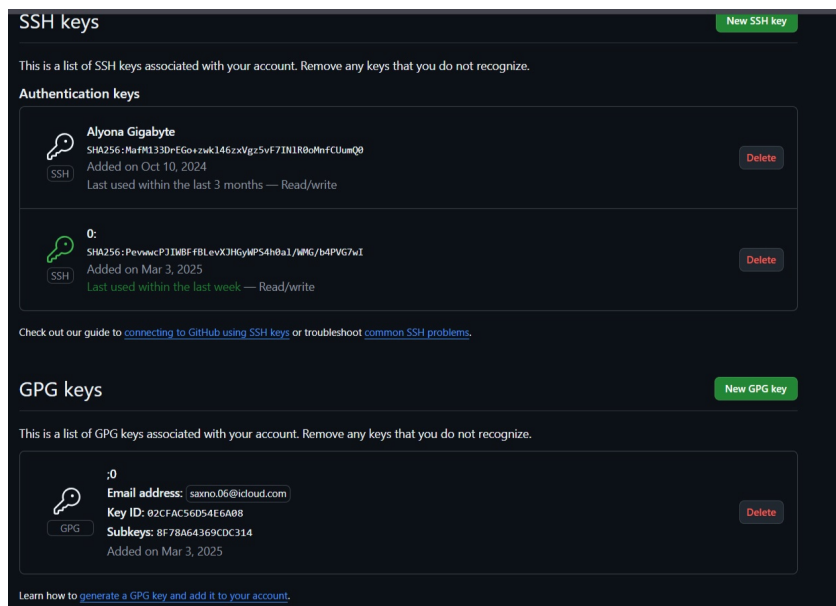


Рис. 4.8: GitHub ключи

Настройка gh Для начало необходимо авторизоваться и утилита задаст несколько наводящих вопросов
(рис.9 4.8).

```
[root@aysakhno ~]# gh auth login
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? HTTPS
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 2307.688F
Press Enter to open https://github.com/login/device in your browser...
Authorization required, but no authorization protocol specified

Error: cannot open display: :0
✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
! Authentication credentials saved in plain text
✓ Logged in as Asakhno
! You were already logged in to this account
```

Рис. 4.9: Настройка gh

Настройка для рабочего пространства Это создание репозитория курса на основе шаблона
(рис.10 4.9).

```
[root@aysakhno ~]# git clone --recursive git@github.com:Asakhno/study_2024-25_os-intro.git os-intro
Cloning into 'os-intro'...
The authenticity of host 'github.com (140.82.121.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhpZisF/zLDA8zPM5vHdKz4UvC0qJ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? [fingerprint]
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
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)
Receiving objects: 100% (36/36), 19.37 KiB | 1.76 MiB/s, готово.
Resolving deltas: 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»
Создание в «/root/work/study/2024-2025/Операционные системы/os-intro/template/presentation»
```

Рис. 4.10: Настройка для рабочего пространства

Настройка каталога курса Перед тем в каталог курса , удалим лишние файлы , создаем необходимые каталоги и отправляем файлы на север
(рис.11 4.11).

```

[root@aysakhno os-intro]# rm package.json
[root@aysakhno os-intro]# echo os-intro > COURSE
[root@aysakhno os-intro]# make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare       Generate directories structure
  prepare-test  Generate test structure

```

Рис. 4.11: Настройка каталога курса

(рис.12 ??).

```

[root@aysakhno os-intro]# git add .
[root@aysakhno os-intro]# git commit -am 'feat(main): make course structure'
[master aeb652e] feat(main): make course structure
2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
[root@aysakhno os-intro]# git push

```

Выводы

Я изучила идеологию и применение средств контроля версий и освоила умения по работе с git.

Список литературы

::: {#refs} <https://esystem.rudn.ru/mod/page/view.php?id=1224371> :::