

Отчёт по лабораторной работе №2

Дисциплина: Опереционные системы

Долгаев Евгений НММбд-01-24

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Ответы на контрольные вопросы	11
5	Выводы	17
	Список литературы	18

Список таблиц

Список иллюстраций

3.1	Установка git и gh	7
3.2	Настройка git	7
3.3	Создание ключей	8
3.4	Загруженный ключ	8
3.5	Загруженный ключ	9
3.6	Настройка автоматических подписей	9
3.7	Создание каталога	9
3.8	Клонирование репозитория	10
3.9	Загрузка файлов на github	10

1 Цель работы

Изучить идеологию и применение средств контроля версий. Освоить умения по работе с git.

2 Задание

1) Выполнение лабораторной работы

1. Создать базовую конфигурацию для работы с git.
2. Создать ключ SSH.
3. Создать ключ PGP.
4. Настроить подписи git.
5. Создать локальный каталог для выполнения заданий по предмету.

2) Ответы на контрольные вопросы

3 Выполнение лабораторной работы

Для начала установим git и gh (рис. 3.1).

```
root@esdolgaev:~# dnf install git
Обновление и загрузка репозитория:
Fedora 41 - x86_64 - Updates
Fedora 41 - x86_64 - Updates
Репозитории загружены.
Пакет "git-2.48.1-1.fc41.x86_64" уже установлен.

Нечего делать.
root@esdolgaev:~# dnf install gh
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет                                Арх.
Установка:                           gh                                x86_64

Сводка транзакции:
Установка: 1 пакета

Общий размер входящих пакетов составит 10 MiB. Необходимо загрузить 10 MiB.
После этой операции будут использоваться дополнительные 43 MiB (установка 43 MiB, удаление 0 B).
Is this ok [y/N]: y
[1/1] gh-0:2.65.0-1.fc41.x86_64
-----
[1/1] Total
Выполнение транзакции
[1/3] Проверить файлы пакета
[2/3] Подготовить транзакцию
[3/3] Установка gh-0:2.65.0-1.fc41.x86_64
Завершено!
```

Рис. 3.1: Установка git и gh

Проведём первоначальную настройку git (рис. 3.2).

```
root@esdolgaev:~# git config --global user.name "Dolgaev Evgeniy"
-bash: git: команда не найдена
root@esdolgaev:~# git config --global user.name "Dolgaev Evgeniy"
root@esdolgaev:~# git config --global user.email "edolgaev@gmail.com"
root@esdolgaev:~# git config --global init.defaultBranch master
root@esdolgaev:~# git config --global core.autocrlf input
root@esdolgaev:~# git config --global core.safecrlf warn
```

Рис. 3.2: Настройка git

Создадим SSH и GPG ключи (рис. 3.3).

```
root@esdolgaev:~# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): key1
Enter passphrase for "key1" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in key1
Your public key has been saved in key1.pub
The key fingerprint is:
SHA256:mWDDvWUTcRX4U+x5wAaQzjGwekQq9N04p7BvU9BY/jI root@esdolgaev
The key's randomart image is:
+---[RSA 4096]-----+
|      . oo+*+=      |
|      o . *X  *      |
|      * B@. + + o    |
|      . =+B* o o .    |
|      . Soo. . .    |
|      o =E .         |
|      . o. o         |
|      .o            |
|      ...           |
+---[SHA256]-----+
root@esdolgaev:~# 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.

gpg: создан каталог '/root/.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? 
```

Рис. 3.3: Создание ключей

Загрузим ключи на github (рис. 3.4, 3.5).

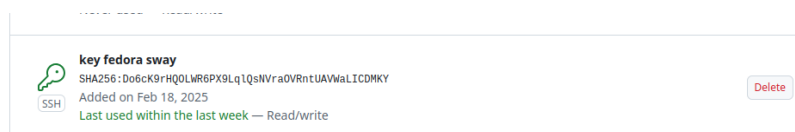


Рис. 3.4: Загруженный ключ

PGP keys

[New GPG key](#)

This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.



key1
Email address: edolgaev@gmail.com
Key ID: 190C17135627BA9E
Subkeys: 8C5F93E4E007085C
Added on Feb 18, 2025

Delete

Рис. 3.5: Загруженный ключ

Настроим автоматических подписей коммитов git (рис. 3.6).

```
root@esdolgaev:~# git config --global user.signingkey <PGP Fingerprint>
-bash: синтаксическая ошибка рядом с неожиданным маркером «newline»
root@esdolgaev:~# git config --global user.signingkey 7BD63F84B36919F04C73A847190C17135627BA9E
root@esdolgaev:~# git config --global commit.gpgsign true
root@esdolgaev:~# git config --global gpg.program $(which gpg2)
root@esdolgaev:~# 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: CFF5-7C2E
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 ssh
✓ Configured git protocol
! Authentication credentials saved in plain text
✓ Logged in as eugerne
```

Рис. 3.6: Настройка автоматических подписей

Создадим рабочий каталог и скопируем в него наш созданный репозиторий (рис. 3.7, 3.8).

```
root@esdolgaev:~# mkdir -p ~/work/study/2024-2025/"Операционные системы"
root@esdolgaev:~# cd ~/work/study/2022-2023/"Операционные системы"
-bash: cd: /root/work/study/2022-2023/Операционные системы: нет такого файла или каталога
root@esdolgaev:~# cd ~/work/study/2024-2025/"Операционные системы"
root@esdolgaev:~# cd ~/work/study/2024-2025/Операционные системы# gh repo create study_2022-2023-os-intro --template=yamadharma/course-directory-student-template --public
Created repository eugerne/study_2022-2023-os-intro on GitHub
https://github.com/eugerne/study_2022-2023-os-intro
```

Рис. 3.7: Создание каталога

```

root@esdolgayev:~/work/study/2024-2025/Операционные системы# git clone --recursive git@github.com:eugene/study_2024-2025_ash-pc.git os-intro
Клонирование в os-intro...
remote: Enumerating objects: 368, done.
remote: Counting objects: 100% (83/83), done.
remote: Compressing objects: 100% (78/78), done.
remote: Total 368 (delta 7), reused 68 (delta 5), pack-reused 285 (from 2)
Получение объектов: 100% (368/368), 117.23 MdB | 6.71 MdB/c, готово.
Определение изменений: 100% (69/69), готово.
Подсказка: «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подсказка: «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/root/work/study/2024-2025/Операционные системы/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 KdB | 491.00 KdB/c, готово.
Определение изменений: 100% (42/42), готово.
Клонирование в «/root/work/study/2024-2025/Операционные системы/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), 341.89 KdB | 779.00 KdB/c, готово.
Определение изменений: 100% (68/68), готово.
Submodule path 'template/presentation': checked out 'c9b2712b4b2d431ad5866c9c72a82d2fca0d4b6'
Submodule path 'template/report': checked out 'c26e22effe7b3e4957d2e7561ab185f5c748'
root@esdolgayev:~/work/study/2024-2025/Операционные системы#

```

Рис. 3.8: Клонирование репозитория

Загрузим файлы на github (рис. 3.9).

```

root@esdolgayev:~/work/study/2024-2025/Операционные системы/os-intro# git add .
root@esdolgayev:~/work/study/2024-2025/Операционные системы/os-intro# git commit -am 'feat(main): make course structure'
[master c4ab858] feat(main): make course structure
1 file changed, 1 insertion(+), 1 deletion(-)
root@esdolgayev:~/work/study/2024-2025/Операционные системы/os-intro# git push
Перечисление объектов: 5, готово.
Пакет объектов: 100% (5/5), готово.
При сканировании используется до 2 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 945.66 KdB | 945.00 KdB/c, готово.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:eugene/study_2024-2025_ash-pc.git
   b488891..c4ab858  master -> master
root@esdolgayev:~/work/study/2024-2025/Операционные системы/os-intro#

```

Рис. 3.9: Загрузка файлов на github

4 Ответы на контрольные вопросы

- 1) Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется.
- 2) Хранилище - это единый репозиторий для хранения файлов. С помощью `commit` можно сохранить в репозитории добавленные изменения. Все изменения сохраняются в истории изменений. Рабочей копией называется файл, находящийся в хранилище.
- 3) В централизованной системе все пользователи подключены к центральному владельцу сети или «серверу». Центральный владелец хранит данные, к которым могут получить доступ другие пользователи, а также информацию о пользователях. У децентрализованных систем нет единого центрального владельца. Вместо этого они используют нескольких центральных владельцев, каждый из которых обычно хранит копию ресурсов, к которым пользователи могут получить доступ. Централизованные: Subversion, децентрализованные: Git
- 4) Действия с VCS 1.Стандартные процедуры работы при наличии центрального репозитория

- a) Работа пользователя со своей веткой начинается с проверки и получения изменений из центрального репозитория
- b) Затем можно вносить изменения в локальном дереве и/или ветке.

2. Работа с локальным репозиторием

- a) Создание локального репозитория
- b) Первичная конфигурация репозитория

3. Работа с сервером репозитория

- a) Для идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый)
- b) Далее ключ нужно связать с репозиторием
- c) Теперь на локальном компьютере можно выполнять стандартные процедуры для работы с git при наличии центрального репозитория

5) Стандартные процедуры работы при наличии центрального репозитория

1. Работа пользователя со своей веткой начинается с проверки и получения изменений из центрального репозитория (при этом в локальное дерево до начала этой процедуры не должно было вноситься изменений):

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

2. Затем можно вносить изменения в локальном дереве и/или ветке.

3. После завершения внесения какого-то изменения в файлы и/или каталоги проекта необходимо:

```
git status
```

4. При необходимости удаляем лишние файлы, которые не хотим отправлять в центральный репозиторий:

5. Затем полезно просмотреть текст изменений на предмет соответствия правилам ведения репозитория:

```
git diff
```

6. Если какие-либо файлы не должны попасть в коммит, то помечаем только те файлы, изме

```
git add ...
```

```
git rm ...
```

7. Если нужно сохранить все изменения в текущем каталоге, то используем:

```
git add .
```

8. Затем сохраняем изменения, поясняя, что было сделано:

```
git commit -am "Some commit message"
```

9. Отправляем изменения в центральный репозиторий:

```
git push
```

6) Основные задачи, решаемые инструментальным средством git

1. Как не потерять файлы с исходным кодом?
2. Как защититься от случайных исправлений и удалений?
3. Как отменить изменения, если они оказались некорректными?
4. Как одновременно поддерживать рабочую версию и разработку новой?

7) Основные команды git

1. Создание основного дерева репозитория:

```
git init
```

2. Получение обновлений (изменений) текущего дерева из центрального репозитория:

```
git pull
```

3. Отправка всех произведённых изменений локального дерева в центральный репозиторий:

```
git push
```

4. Просмотр списка изменённых файлов в текущей директории:

```
git status
```

5. Просмотр текущих изменений:

```
git diff
```

6. Сохранение текущих изменений:

a) добавить все изменённые и/или созданные файлы и/или каталоги:

```
git add .
```

b) добавить конкретные изменённые и/или созданные файлы и/или каталоги:

```
git add имена_файлов
```

c) удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в директории):

```
git rm имена_файлов
```

7. Сохранение добавленных изменений:

a) сохранить все добавленные изменения и все изменённые файлы:

```
git commit -am 'Описание коммита'
```

b) сохранить добавленные изменения с внесением комментария через встроенный редактор:

```
git commit
```

c) создание новой ветки, базирующейся на текущей:

```
git checkout -b имя_ветки
```

d) переключение на некоторую ветку:

`git checkout имя_ветки`

- е) отправка изменений конкретной ветки в центральный репозиторий:

`git push origin имя_ветки`

- ф) слияние ветки с текущим деревом:

`git merge --no-ff имя_ветки`

8. Удаление ветки:

- а) удаление локальной уже слитой с основным деревом ветки:

`git branch -d имя_ветки`

- б) принудительное удаление локальной ветки:

`git branch -D имя_ветки`

- с) удаление ветки с центрального репозитория:

`git push origin :имя_ветки`

- 8) Команда `status` для просмотра изменений в рабочем каталоге, сделанных с момента последней ревизии:

`git status`

Команда, которая связывается с указанным удалённым проектом и забирает все те данные проекта, которых у вас ещё нет.

`git fetch []`

- 9) Команда `git branch` — главный инструмент для работы с ветвлением. С ее помощью можно добавлять новые ветки, перечислять и переименовывать существующие и удалять их.

- 10) Если какие-либо файлы не должны попасть в коммит, то помечаем только те файлы, изменения которых нужно сохранить. Для этого используем команды добавления и/или удаления с нужными опциями:

```
git add ...
```

```
git rm ...
```


5 Выводы

В ходе выполнения лабораторной работы я освоил навыки по работе с git.

Список литературы