

Лабораторная работа №2

Операционные системы

Дмитрий Юрьевич Дымченко

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Ответы на контрольные вопросы	12
5	Выводы	15

Список иллюстраций

3.1	Установка git и gh	7
3.2	Базовая настройка git	7
3.3	Генерация ssh-ключей	8
3.4	Генерация ssh-ключей	8
3.5	Генерация gpg-ключа	9
3.6	Привязка gpg-ключа	10
3.7	Настройка коммитов	10
3.8	Авторизация	10
3.9	Создание шаблона	11
3.10	Клонирование репозитория	11
3.11	Настройка каталога курса	11
3.12	Настройка каталога курса	11

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 Выполнение лабораторной работы

Установка git и gh командами `dnf install git` и `dnf install gh` (рис. [3.1]).

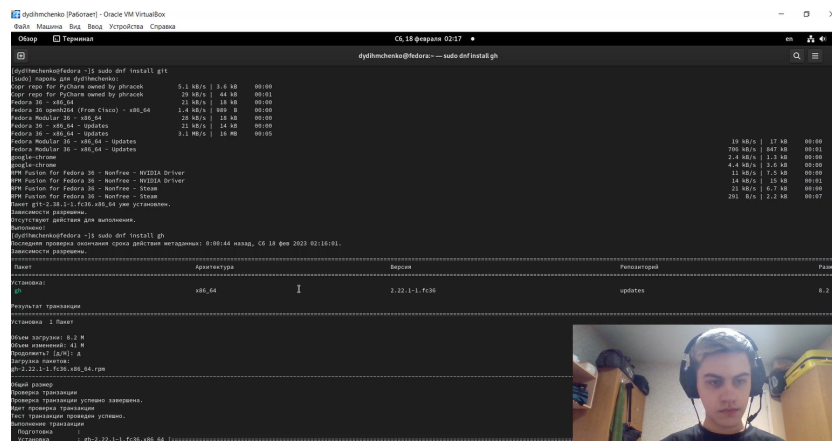


Рис. 3.1: Установка git и gh

Произведем базовую настройку git. (рис. [3.2])

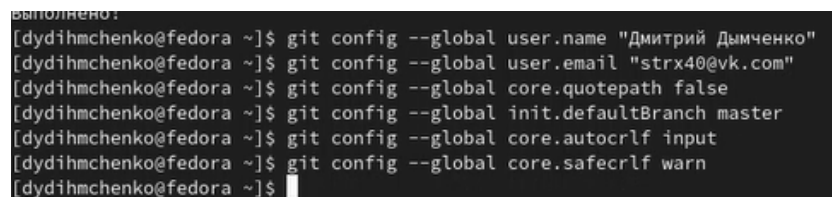


Рис. 3.2: Базовая настройка git

Генерируем ssh-ключи командами `ssh-keygen -t rsa -b 4096` и `ssh-keygen -t ed25519` (рис. [3.3]), (рис. [3.4])

```
[dydihmchenko@fedora ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/dydihmchenko/.ssh/id_rsa):
/home/dydihmchenko/.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/dydihmchenko/.ssh/id_rsa
Your public key has been saved in /home/dydihmchenko/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:020yfr0ozLDo9tU3XNP0zQZDbrE0qhLJKbSADZnGWE dydihmchenko@fedora
The key's randomart image is:
+---[RSA 4096]-----+
|.o E+      . |
|..=      + . ..|
| . o . o = ..|
| o o o o + o .o|
| . o S + = .+=|
| . . o * ..* |
| ... o .o. |
| O+ .+ . .|
| oo.*+... |
+----[SHA256]-----+
[dydihmchenko@fedora ~]$
```

Рис. 3.3: Генерация ssh-ключей

```
[dydihmchenko@fedora ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/dydihmchenko/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/dydihmchenko/.ssh/id_ed25519
Your public key has been saved in /home/dydihmchenko/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:GvcL0BdwyH60LEWiLykW7EsINxRvLVgvxWePdQMZR/s dydihmchenko@fedora
The key's randomart image is:
---[ED25519 256]---+
|o.....oo==o |
|. = +..=++o. |
|. + B ++ *....|
|o = + *+ o. . |
|. = .S+ . E |
|o o *.o. |
|. . . . |
|. . . . |
|. . . . |
+----[SHA256]-----+
[dydihmchenko@fedora ~]$
```

Рис. 3.4: Генерация ssh-ключей

Далее генерируем gpg-ключ командой `gpg --full-generate-key` (рис. [3.5]).


```
[dydihmchenko@fedora ~]$ gpg --full-generate-key
gpg (GnuPG) 2.3.7; 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/dydihmchenko/.gnupg'
gpg: создан щит с ключами '/home/dydihmchenko/.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
  <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: Дмитрий Дымченко
Адрес электронной почты: strx40@vk.com
Примечание:
Используется таблица символов 'utf-8'.
Вы выбрали следующий идентификатор пользователя:
  "Дмитрий Дымченко <strx40@vk.com>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? 0
Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? █
```

Рис. 3.5: Генерация gpg-ключа

Привяжем gpg-ключ к учетной записи Github (рис. [3.6]).

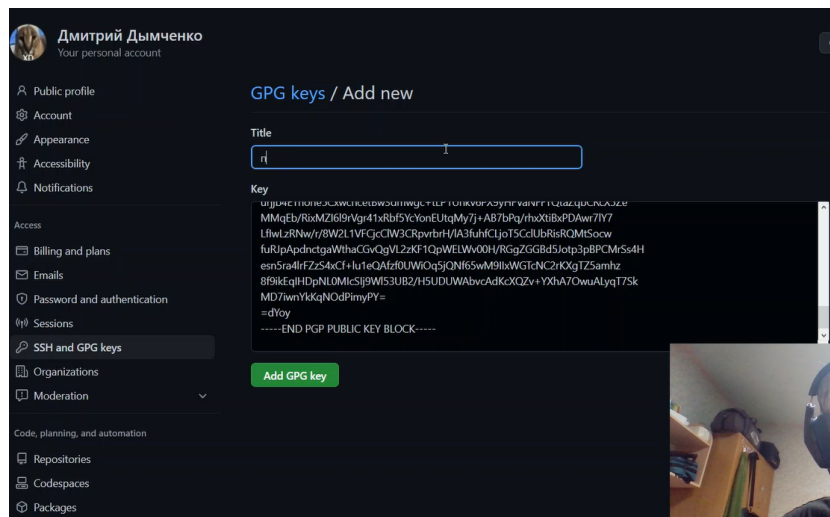


Рис. 3.6: Привязка gpg-ключа

Настраиваем автоматические подписи коммитов git (рис. [3.7]).

```
[dydihmchenko@fedora ~]$ git config --global user.signingkey EA41387AB4B917FB
[dydihmchenko@fedora ~]$ git config --global commit.gpgsign true
[dydihmchenko@fedora ~]$ git config --global gpg.program $(which gpg2)
[dydihmchenko@fedora ~]$
```

Рис. 3.7: Настройка коммитов

Авторизуемся в Github через консоль (рис. [3.8]).

```
[dydihmchenko@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? Paste an authentication token
Tip: you can generate a Personal Access Token here https://github.com/settings/tokens
The minimum required scopes are 'repo', 'read:org', 'workflow'.
? Paste your authentication token: *****
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as strx40
[dydihmchenko@fedora ~]$
```

Рис. 3.8: Авторизация

Создаем шаблон рабочего пространства (рис. [3.9]).

```

dydihmchenko@fedora ~$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
dydihmchenko@fedora ~$ cd ~/work/study/2022-2023/"Операционные системы"
dydihmchenko@fedora Операционные системы$ gh repo create study_2022-2023_os-intro --template=yamadharma/course-directory-student-template --public
Created repository strx40/study_2022-2023_os-intro on GitHub
dydihmchenko@fedora Операционные системы$

```

Рис. 3.9: Создание шаблона

Клонируем репозиторий (рис. [3.10]).

```

dydihmchenko@fedora Операционные системы$ git clone --recursive git@github.com:strx40/study_2022-2023_os-intro.git os-intro
Клонирование в «os-intro»...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 27 (delta 1), reused 11 (delta 0), pack-reused 0
Получение объектов: 100% (27/27), 16.97 Киб | 5.66 МБ/с, готово.
Определение изменений: 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»
Клонирование в «/home/dydihmchenko/work/study/2022-2023/Операционные системы/os-intro/template/presentation»...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 82 (delta 28), reused 77 (delta 23), pack-reused 0
Получение объектов: 100% (82/82), 92.90 Киб | 391.00 Киб/с, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в «/home/dydihmchenko/work/study/2022-2023/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 101, done.
remote: Counting objects: 100% (101/101), done.
remote: Compressing objects: 100% (76/76), done.
remote: Total 101 (delta 40), reused 88 (delta 27), pack-reused 0
Получение объектов: 100% (101/101), 327.25 Киб | 242.80 Киб/с, готово.
Определение изменений: 100% (40/40), готово.
Submodule path 'template/presentation': checked out 'b1bc3800ee1f5809264cb75d316174540b753e'
Submodule path 'template/report': checked out '1d1b1dcac9c267a83917b82e3ae11a33b1e3b2'
dydihmchenko@fedora Операционные системы$

```

Рис. 3.10: Клонирование репозитория

Проводим настройку каталога курса (рис. [3.11]), (рис. [3.12]).

```

dydihmchenko@fedora Операционные системы$ cd ~/work/study/2022-2023/"Операционные системы"/os-intro
dydihmchenko@fedora os-intro$ rm package.json
dydihmchenko@fedora os-intro$ echo os-intro > COURSE
dydihmchenko@fedora os-intro$ make
dydihmchenko@fedora os-intro$ git add .
dydihmchenko@fedora os-intro$ git commit -am 'feat(main): make course structure'

```

Рис. 3.11: Настройка каталога курса

```

create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_fignos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_secnos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/core.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/main.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/pandocatattributes.py
create mode 100644 project-personal/stage6/report/report.md
dydihmchenko@fedora os-intro$ git push
Перечисление объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 343.08 Киб | 736.00 Киб/с, готово.
Всего 38 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:strx40/study_2022-2023_os-intro.git
91c7f7e..fbce87f master -> master
dydihmchenko@fedora os-intro$

```

Рис. 3.12: Настройка каталога курса

4 Ответы на контрольные вопросы

- 1) Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Это программное обеспечение для облегчения работы с изменяющейся информацией. VCS позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.
- 2) Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Хранилище (repository), или репозиторий, — место хранения всех версий и служебной информации. Commit («(трудовой) вклад», не переводится) — синоним версии; процесс создания новой версии. История — место, где сохраняются все коммиты, по которым можно посмотреть данные о коммитах. Рабочая копия — текущее состояние файлов проекта, основанное на версии, загруженной из хранилища.
- 3) Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные VCS: одно основное хранилище всего проекта и каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно. Децентрализованные VCS: у каждого пользователя свой вариант (возможно не один) репозитория.
- 4) Опишите действия с VCS при единоличной работе с хранилищем.
- 5) Опишите порядок работы с общим хранилищем VCS.

- 6) Каковы основные задачи, решаемые инструментальным средством git? Git — это система управления версиями. У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.
- 7) Назовите и дайте краткую характеристику командам git. git –version (Проверка версии Git) git init (Инициализировать ваш текущий рабочий каталог как Git-репозиторий) git clone <https://www.github.com/username/repo-name> (Скопировать существующий удаленный Git-репозиторий) git remote (Просмотреть список текущих удалённых репозиториях Git) git remote -v (Для более подробного вывода) git add my_script.py (Можете указать в команде конкретный файл). git add . (Позволяет охватить все файлы в текущем каталоге, включая файлы, чье имя начинается с точки) git commit -am “Commit message” (Вы можете сжать все индексированные файлы и отправить коммит). git branch (Просмотреть список текущих веток можно с помощью команды branch) git –help (Чтобы узнать больше обо всех доступных параметрах и командах) git push origin master (Передать локальные коммиты в ветку удаленного репозитория).
- 8) Приведите примеры использования при работе с локальным и удалённым репозиториями.
- 9) Что такое и зачем могут быть нужны ветки (branches)? Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций: чтобы не повредить основному проекту, создается новая ветка специально для экспериментов.
- 10) Как и зачем можно игнорировать некоторые файлы при commit? Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые

машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты.

5 Выводы

В ходе выполнения лабораторной работы я познакомился с принципами работы системы контроля версий, а также освоил некоторые умения работы с git.