

Лабораторная работа 2

по архитектуре компьютеров

Екатерины Алексеевны Козловой

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Выводы	19
	Список литературы	20

Список иллюстраций

4.1	Установка git и gh	14
4.2	Базовая настройка git	15
4.3	Настройка github	15
4.4	Добавление pgr и ssh ключей в github	16
4.5	Настройка автоматических подписей коммитов git	17
4.6	Настройка каталога курса	17
4.7	Настройка каталога курса	18
4.8	Настройка каталога курса	18

Список таблиц

1 Цель работы

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

2 Задание

1. Установка git
2. Установка gh
3. Базовая настройка git
4. Создание ключей ssh
5. Создание ключа pgp
6. Настройка github
7. Добавление pgp и ssh ключей в github
8. Настройка автоматических подписей коммитов git
9. Настройка gh
10. Создание репозитория курса на основе шаблона
11. Настройка каталога курса

3 Теоретическое введение

Системы контроля версий. Общие понятия

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.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными

участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.

Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

4 Выполнение лабораторной работы

1. Установка git

У меня уже был установлен git с прошлого года, делала это я с помощью утилиты `dnf install git`.

(рис. 4.1)

2. Установка gh

Устанавливаю gh также с помощью утилиты `dnf install gh`.

(рис. 4.1)

3. Базовая настройка git

Задаю имя и почту владельца, настраиваю utf-8 в выводе сообщений, задаю имя начальной ветки, параметр `autocrlf` и параметр `safecrlf`.

(рис. 4.2)

4. Создание ключей ssh

Создаю ключи ssh по алгоритму `rsa` с ключём размером 4096 бит и по алгоритму `ed25519` через утилиты.

5. Создание ключа pgp

Генерирую ключ командой `gpg --full-generate-key` и выбираю нужные опции, заполняю личную информацию.

6. Настройка github

Создаю учётную запись (она у меня была с прошлого года) на своё имя и почту, заполняю всю необходимую информацию.

(рис. 4.3)

7. Добавление pgr и ssh ключей в github

Копирую утилитой xclip ключи, которые сгенерировались и находятся в файлах, а дальше в настройках github добавляю их в разделе ssh и gpg ключи.

(рис. 4.4)

8. Настройка автоматических подписей коммитов git

Указываю git применять мою почту для подписи коммитов через команды:
git config –global user.signingkey git config –global commit.gpgsign true git config –global gpg.program \$(which gpg2)

(рис. 4.5)

9. Настройка gh

Авторизовываюсь и отвечаю на наводящие вопросы в терминале.

(рис. 4.5)

10. Сознание репозитория курса на основе шаблона

Создаю репозиторий курса с взятой template основного, потом утилитой mkdir создаю репозиторий здесь и переношу содержимое себе.

11. Настройка каталога курса

Дальше я перехожу в созданный каталог, удаляю лишние файлы и создаю необходимые каталоги. Отправляю файлы на сервер.

(рис. 4.6)

(рис. 4.7)

(рис. 4.8)

12. Ответы на контрольные вопросы

- 1) Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Система контроля версий — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями 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. Системы контроля версий (Version Control System, VCS) применяются для:

- Хранение полной истории изменений
- причин всех производимых изменений
- Откат изменений, если что-то пошло не так
- Поиск причины и ответственного за появления ошибок в программе
- Совместная работа группы над одним проектом
- Возможность изменять код, не мешая работе других пользователей

- 2) Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Репозиторий - хранилище версий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией. Commit — отслеживание изменений, сохраняет разницу в изменениях. Рабочая копия - копия проекта, связанная с репозиторием (текущее состояние файлов проекта, основанное на версии из хранилища (обычно на последней)) История хранит все изменения в проекте и позволяет при необходимости обратиться к нужным данным.

- 3) Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные VCS (Subversion; CVS; TFS; VAULT; AccuRev):
- Одно основное хранилище всего проекта
 - Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно

Децентрализованные VCS (Git; Mercurial; Bazaar): • У каждого пользователя свой вариант (возможно не один) репозитория • Присутствует возможность добавлять и забирать изменения из любого репозитория

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

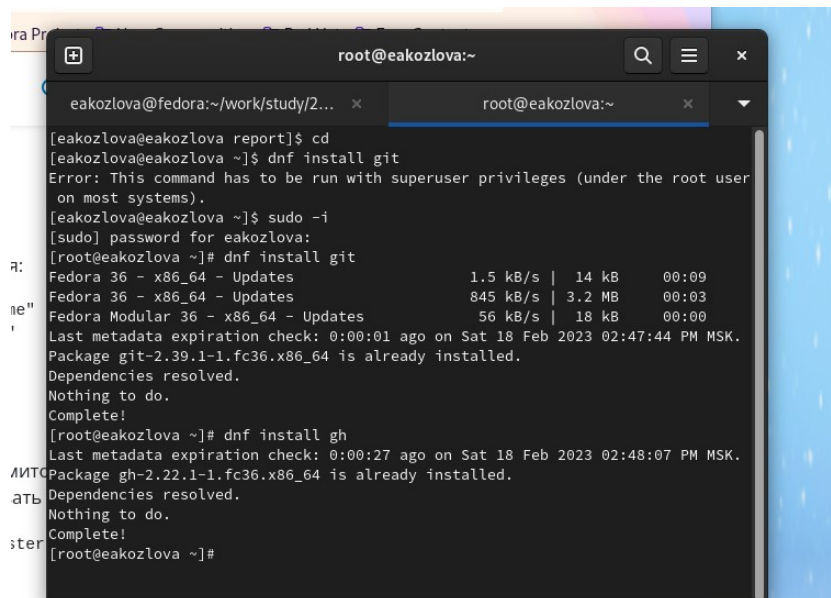
- 4) Опишите действия с VCS при единоличной работе с хранилищем. Сначала создаем и подключаем удаленный репозиторий. Затем по мере изменения проекта отправлять эти изменения на сервер.
- 5) Опишите порядок работы с общим хранилищем VCS. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент.
- 6) Каковы основные задачи, решаемые инструментальным средством git? 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.

Наиболее часто используемые команды git: • 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) Приведите примеры использования при работе с локальным и удалённым репозиториями. `git push --all` (`push origin master/любой branch`)
- 9) Что такое и зачем могут быть нужны ветви (branches)? Ветвление («ветка», `branch`) — один из параллельных участков истории в одном хранилище, исходящих из одной версии (точки ветвления). • Обычно есть главная ветка (`master`), или ствол (`trunk`). • Между ветками, то есть их концами, возможно слияние. Используются для разработки новых функций.
- 10) Как и зачем можно игнорировать некоторые файлы при `commit`? Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные

файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов.



```
root@eakozlova:~  
[eakozlova@eakozlova report]$ cd  
[eakozlova@eakozlova ~]$ dnf install git  
Error: This command has to be run with superuser privileges (under the root user on most systems).  
[eakozlova@eakozlova ~]$ sudo -i  
[sudo] password for eakozlova:  
[root@eakozlova ~]# dnf install git  
Fedora 36 - x86_64 - Updates          1.5 kB/s | 14 kB      00:09  
Fedora 36 - x86_64 - Updates          845 kB/s | 3.2 MB     00:03  
Fedora Modular 36 - x86_64 - Updates  56 kB/s | 18 kB      00:00  
Last metadata expiration check: 0:00:01 ago on Sat 18 Feb 2023 02:47:44 PM MSK.  
Package git-2.39.1-1.fc36.x86_64 is already installed.  
Dependencies resolved.  
Nothing to do.  
Complete!  
[root@eakozlova ~]# dnf install gh  
Last metadata expiration check: 0:00:27 ago on Sat 18 Feb 2023 02:48:07 PM MSK.  
Package gh-2.22.1-1.fc36.x86_64 is already installed.  
Dependencies resolved.  
Nothing to do.  
Complete!  
[root@eakozlova ~]#
```

Рис. 4.1: Установка git и gh

```
root@eakozlova:~  
[eakozlova@eakozlova ~]$ sudo -i  
[sudo] password for eakozlova:  
[root@eakozlova ~]# dnf install git  
Fedora 36 - x86_64 - Updates          1.5 kB/s | 14 kB    00:00  
Fedora 36 - x86_64 - Updates          845 kB/s | 3.2 MB   00:03  
Fedora Modular 36 - x86_64 - Updates  56 kB/s | 18 kB    00:00  
Last metadata expiration check: 0:00:01 ago on Sat 18 Feb 2023 02:47:44 PM MSK.  
Package git-2.39.1-1.fc36.x86_64 is already installed.  
Dependencies resolved.  
Nothing to do.  
Complete!  
[root@eakozlova ~]# dnf install gh  
Last metadata expiration check: 0:00:27 ago on Sat 18 Feb 2023 02:48:07 PM MSK.  
Package gh-2.22.1-1.fc36.x86_64 is already installed.  
Dependencies resolved.  
Nothing to do.  
Complete!  
[root@eakozlova ~]# git config --global user.name "Ekaterina Kozlova"  
[root@eakozlova ~]# git config --global user.email "miserabletime11@yandex.ru"  
[root@eakozlova ~]# git config --global core.quotepath false  
[root@eakozlova ~]# git config --global init.defaultBranch master  
[root@eakozlova ~]# git config --global core.autocrlf input  
[root@eakozlova ~]# git config --global core.safecrlf warn  
[root@eakozlova ~]#
```

Рис. 4.2: Базовая настройка git

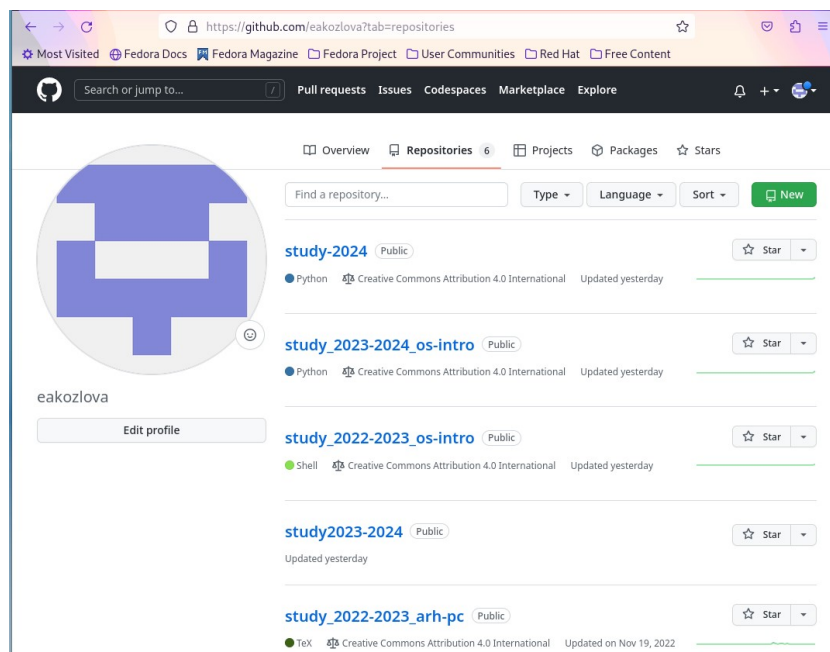




Рис. 4.3: Настройка github

SSH keys

[New SSH key](#)

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

Authentication Keys


 SSH	Ekaterina SHA256: 1S5D0ukjeNf+LbK9NLLFhDbRnC15Q7sXZV04aBhYlgM Added on Feb 16, 2023 Last used within the last week — Read/write	Delete
 SSH	ed25519 SHA256: eR38y8jPI8+NqMXU50LUTIzEUyvpX6fJ/NjDHNCNnYU Added on Feb 16, 2023 Never used — Read/write	Delete

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

GPG keys

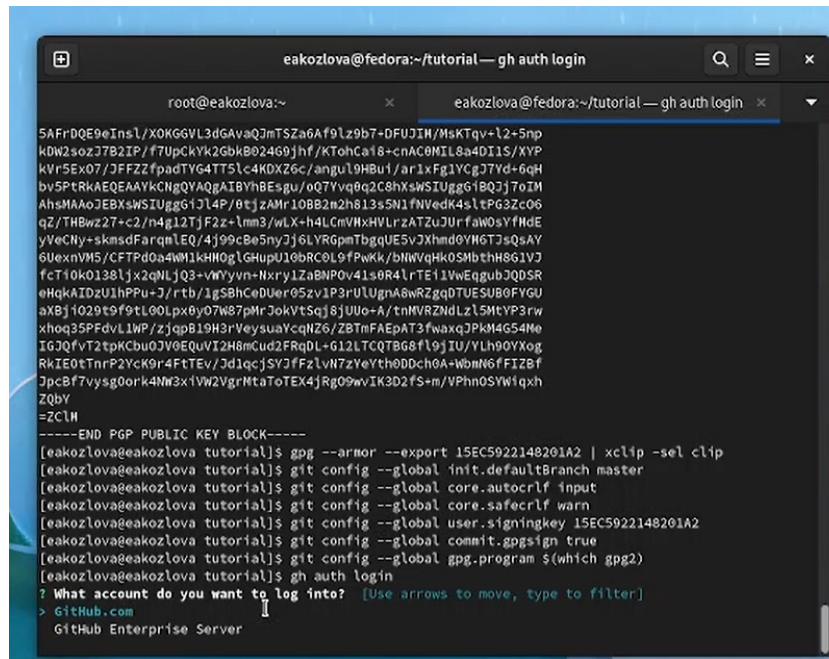
[New GPG key](#)

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

 GPG	Email address: miserabletime11@yandex.ru Key ID: 15EC5922148201A2 Subkeys: 200285AD4C343712 Added on Feb 16, 2023	Delete
--	---	------------------------

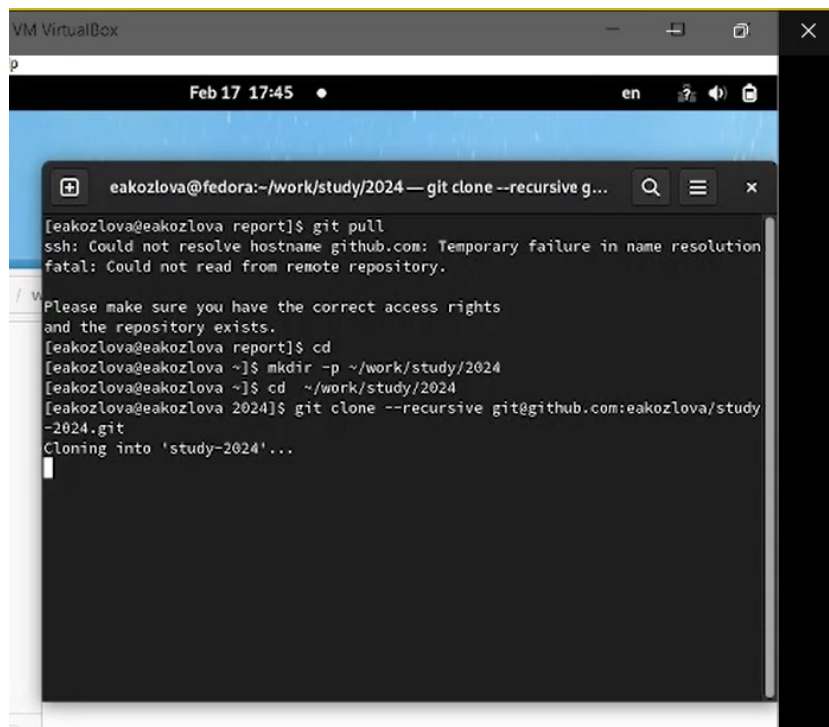
Learn how to [generate a GPG key](#) and [add it to your account](#).

Рис. 4.4: Добавление pgr и ssh ключей в github



```
SAFrDQE9eInsl/XOKGVL3dGavaQJnTSZa6A9Lz9b7-DFUJH/MsKTqv+l2-5np
kDW2soz37B2IP/f7UpckYk2GbkB02469jh/f/KTohCa18-cnAC0MI18a4DI1S/XYP
kVr5Ex07/JFFZzfpadTYG4TT5Lc4KDXZ6c/angul9HBui/ar1xfg1YcgJ7Yd-6qH
bV5PTrKAQEAAVKNgQYAQgAIBYhBE5gu/o07Yvq8e2C8hXsWSIUggG1BQJ7J7oIM
AhsMAAo7EBXsWSIUggG1JL4P/0tjzAMr10B82m2h813s5N1fNVedK4slEPG3Zc06
qZ/THBwz27+c2/n4g12TjF2z+lmm3/wLX-h4LCmVHxHVLrZATZuJurfaw0sYfHdE
yVeCNy-skmsdFarqmlEQ/4j99cBe5ny7j6LYRGpmTbgqUESVJXhmd0Yh6TJsQsAY
6UexnVM5/CFTPd0a4W1kHMOgLGHupU10bRC0L9fPwKk/bNwVqHk0SmbthH8G1VJ
fct10k0138ljx2qNLjQ3-vMyvvn-Nxry1ZaBNPov41s0R4lrTE11VwEgubJQDSR
eHqkAIDzUihPPu-7/rbt/1gSBhCeDUer05zv1P3rULUgnA6wRZgqDTUESUB0FYGU
aXbj1029t9f9tL00Lpx0y07w87pMrJokVtSqj8jUuo-A/tmMVRZNdLz15MtYP3rw
xhoq35PFdvL1WP/zjqpB19H3rVeysuaYcqNZ6/ZBTmFAEpAT3fwaxqJPkM4G54Me
IGJQfvT2tpKCbU0Jv0EQvI2H8mCud2FrqDL+G12LTCQTB68fl9jIU/VLh90YXog
RkIE0etTnrP2YcK9r4FtEv/Jd1ecjSY3fFzLvn7zYeYh0DDch0A-WbmN6fFIZBf
JpcBf7vysg0ork4Nm3x1Vw2VgrHtaToEX4jRg09wvIK3D2fS+m/VPhn0SYWiqxh
ZQby
-----END PGP PUBLIC KEY BLOCK-----
[eakozlova@eakozlova tutorial]$ gpg --armor --export 15EC5922148201A2 | xclip -sel clip
[eakozlova@eakozlova tutorial]$ git config --global init.defaultBranch master
[eakozlova@eakozlova tutorial]$ git config --global core.autocrlf input
[eakozlova@eakozlova tutorial]$ git config --global core.safecrlf warn
[eakozlova@eakozlova tutorial]$ git config --global user.signingkey 15EC5922148201A2
[eakozlova@eakozlova tutorial]$ git config --global commit.gpgsign true
[eakozlova@eakozlova tutorial]$ git config --global gpg.program $(which gpg2)
[eakozlova@eakozlova tutorial]$ gh auth login
? What account do you want to log into? [Use arrows to move, type to filter]
> GitHub.com
  GitHub Enterprise Server
```

Рис. 4.5: Настройка автоматических подписей коммитов git



```
Feb 17 17:45 • en
[eakozlova@eakozlova report]$ git pull
ssh: Could not resolve hostname github.com: Temporary failure in name resolution
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
[eakozlova@eakozlova report]$ cd
[eakozlova@eakozlova ~]$ mkdir -p ~/work/study/2024
[eakozlova@eakozlova ~]$ cd ~/work/study/2024
[eakozlova@eakozlova 2024]$ git clone --recursive git@github.com:eakozlova/study
-2024.git
Cloning into 'study-2024'...
```

Рис. 4.6: Настройка каталога курса

```
eakozlova@fedora:~/work/study/2024/study-2024
Receiving objects: 100% (82/82), 92.90 KiB | 201.00 KiB/s, done.
Resolving deltas: 100% (28/28), done.
Cloning into '/home/eakozlova/work/study/2024/study-2024/template/report'...
remote: Enumerating objects: 101, done.
remote: Counting objects: 100% (101/101), done.
remote: Compressing objects: 100% (70/70), done.
remote: Total 101 (delta 40), reused 88 (delta 27), pack-reused 0
Receiving objects: 100% (101/101), 327.25 KiB | 594.00 KiB/s, done.
Resolving deltas: 100% (40/40), done.
Submodule path 'template/presentation': checked out 'b1be380ee91f5809264cb755d3
16174540b753e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a83917b82e3aef11a3
3b1e3b2'
[eakozlova@eakozlova 2024]$ rm package.json
rm: cannot remove 'package.json': No such file or directory
[eakozlova@eakozlova 2024]$ mc

[eakozlova@eakozlova 2024]$ cd ~/work/study/2024/study-2024
[eakozlova@eakozlova study-2024]$ rm package.json
[eakozlova@eakozlova study-2024]$ echo os-intro > COURSE
[eakozlova@eakozlova study-2024]$ make
[eakozlova@eakozlova study-2024]$ git add .
[eakozlova@eakozlova study-2024]$ git commit -am 'feat(main): make course struct
ure'
```

Рис. 4.7: Настройка каталога курса

```
VirtualBox
Feb 17 17:51 • en
eakozlova@fedora:~/work/study/2024/study-2024
py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_secnos.
py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_tableno
s.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/_i
nit__.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/cor
e.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/mai
n.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/pan
docattributes.py
create mode 100644 project-personal/stage6/report/report.md
[eakozlova@eakozlova study-2024]$ git push
Enumerating objects: 40, done.
Counting objects: 100% (40/40), done.
Compressing objects: 100% (30/30), done.
Writing objects: 100% (38/38), 343.05 KiB | 2.35 MiB/s, done.
Total 38 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:eakozlova/study-2024.git
c49ef14..8b673a4 master -> master
[eakozlova@eakozlova study-2024]$
```

Рис. 4.8: Настройка каталога курса

5 Выводы

Я изучила идеологию и применение средств контроля версий, а также приобрела навыки по работе с git.

Список литературы