

Отчёт по лабораторной работе №2

Первоначальная настройка git

Чекмарев Александр Дмитриевич | Группа НПИбд-02-23

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Установка программного обеспечения	6
2.2	Базовая настройка git	7
2.3	Создание ключа ssh	8
2.4	Создание ключа pgr	10
2.5	Настройка github	12
2.6	Добавление PGP ключа в GitHub	13
2.7	Настройка автоматических подписей коммитов git	15
2.8	Настройка gh	16
2.9	Шаблон для рабочего пространства	16
3	Контрольные вопросы	21
4	Выводы	24
	Список литературы	25

Список иллюстраций

2.1	Рис 2.1.1: установка git	6
2.2	Рис 2.1.2: установка gh	7
2.3	Рис 2.2.1: запись имени	7
2.4	Рис 2.2.2: запись почты	7
2.5	Рис 2.2.3: настройка utf-8	8
2.6	Рис 2.2.4: задача начальной ветки	8
2.7	Рис 2.2.5: настройка параметра	8
2.8	Рис 2.2.6: настройка параметра	8
2.9	Рис 2.3.1: создание ключа с конкретным размером	9
2.10	Рис 2.3.2: создание ключа ed25519	10
2.11	Рис 2.4.1: процесс перед генерацией ключа	11
2.12	Рис 2.4.2: сгенерированный ключ	12
2.13	Рис 2.5.1: демонстрация профиля на github	13
2.14	Рис 2.6.1: вывод списка ключей	13
2.15	Рис 2.6.2: сгенерированный PGP ключ	14
2.16	Рис 2.6.3: создание нового GPG ключа на гитхабе	14
2.17	Рис 2.6.4: демонстрация завершения работы с GPG ключом	15
2.18	Рис 2.7.1: настройка подписей коммитов git	15
2.19	Рис 2.8.1: демонстрация настройки gh	16
2.20	Рис 2.9.1: создание каталога	16
2.21	Рис 2.9.2: переход в каталог	17
2.22	Рис 2.9.3: создание репозитория	17
2.23	Рис 2.9.4: клонирование репозитория	18
2.24	Рис 2.9.5: переход в другой каталог	19
2.25	Рис 2.9.6: удаление файла .json	19
2.26	Рис 2.9.7: создание каталогов	19
2.27	Рис 2.9.8: сохранение и выбор файлов для отправки на сервер	20
2.28	Рис 2.9.9: отправка файлов на сервер	20

Список таблиц

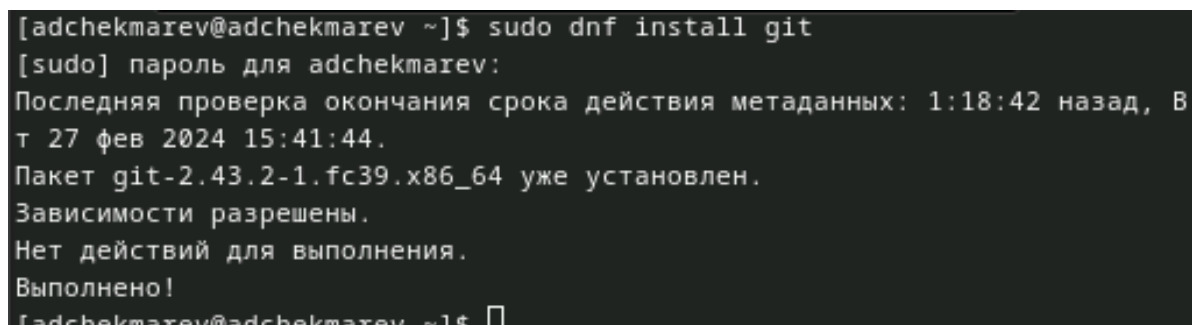
1 Цель работы

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

2 Выполнение лабораторной работы

2.1 Установка программного обеспечения

Установка git • Установим git: `dnf install git`



```
[adchekmarev@adchekmarev ~]$ sudo dnf install git
[sudo] пароль для adchekmarev:
Последняя проверка окончания срока действия метаданных: 1:18:42 назад, В
т 27 фев 2024 15:41:44.
Пакет git-2.43.2-1.fc39.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
[adchekmarev@adchekmarev ~]$
```

Рис. 2.1: Рис 2.1.1: установка git

Установка gh • Установим gh: `dnf install gh`

```
[adchekmarev@adchekmarev ~]$ sudo dnf install gh
Последняя проверка окончания срока действия метаданных: 1:19:19 назад, В
т 27 фев 2024 15:41:44.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий  Размер
=====
Установка:
gh          x86_64       2.43.1-1.fc39 updates      9.1 М
Результат транзакции
=====
Установка  1 Пакет

Объем загрузки: 9.1 М
Объем изменений: 46 М
Продолжить? [д/Н]: y
```

Рис. 2.2: Рис 2.1.2: установка gh

2.2 Базовая настройка git

Зададим имя и email владельца репозитория: *git config --global user.name "Name Surname"*

```
[adchekmarev@adchekmarev ~]$ git config --global user.name "Chekmarev Al
exander"
```

Рис. 2.3: Рис 2.2.1: запись имени

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

```
[adchekmarev@adchekmarev ~]$ git config --global user.email "sasha.cekma
rev4@mail.ru"
```

Рис. 2.4: Рис 2.2.2: запись почты

Настроим utf-8 в выводе сообщений git: *git config --global core.quotePath false*

```
[adchekmarev@adchekmarev ~]$ git config --global core.quotepath false
```

Рис. 2.5: Рис 2.2.3: настройка utf-8

Настроим верификацию и подписание коммитов git (см. Верификация коммитов git с помощью GPG). Зададим имя начальной ветки (будем называть её master): *git config --global init.defaultBranch master*

```
[adchekmarev@adchekmarev ~]$ git config --global init.defaultBranch master
```

Рис. 2.6: Рис 2.2.4: задача начальной ветки

Параметр autocrlf: *git config --global core.autocrlf input*

```
[adchekmarev@adchekmarev ~]$ git config --global core.autocrlf input
```

Рис. 2.7: Рис 2.2.5: настройка параметра

Параметр safecrlf: *git config --global core.safecrlf warn*

```
[adchekmarev@adchekmarev ~]$ git config --global core.safecrlf warn
```

Рис. 2.8: Рис 2.2.6: настройка параметра

2.3 Создание ключа ssh

по алгоритму rsa с ключём размером 4096 бит: *ssh-keygen -t rsa -b 4096*


```

[adchekmarev@adchekmarev ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/adchekmarev/.ssh/id_rsa):
Created directory '/home/adchekmarev/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/adchekmarev/.ssh/id_rsa
Your public key has been saved in /home/adchekmarev/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:qDcSJlaohYj5M6wz6lvvrLilqq3C2N5kq2Z2ZXggqNQ adchekmarev@adchekmarev
The key's randomart image is:
+---[RSA 4096]-----+
|
|oo .
|=+.
|*.E.
|+ B.oo. S
|.o =.o+
|*. o=+o
|+=O+=+ .
|B%B==+
+-----[SHA256]-----+

```

Рис. 2.9: Рис 2.3.1: создание ключа с конкретным размером

по алгоритму ed25519: `ssh-keygen -t ed25519`

```
[adchekmarev@adchekmarev ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/adchekmarev/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/adchekmarev/.ssh/id_ed25519
Your public key has been saved in /home/adchekmarev/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:UZhfGNQrRpJpo0m0iUsoNL4Y/hdSP5nUQyldqk0hwgg adchekmarev@adchekmarev
The key's randomart image is:
+--[ED25519 256]--+
|E   . .  =+0* .   |
|. oo   * . *=o    |
| ++o o  +=. .    |
|++ .o . . . . ++. |
|=o+ . . o+S . .   |
|=o . . . =.       |
|oo   . . .        |
| . . .            |
| . .              |
+-----[SHA256]-----+
```

Рис. 2.10: Рис 2.3.2: создание ключа ed25519

2.4 Создание ключа gpg

- Генерируем ключ: `gpg --full-generate-key`
- Из предложенных опций выбираем:
 - тип RSA and RSA;
 - размер 4096;
 - выберите срок действия; значение по умолчанию – 0 (срок действия не истекает никогда)
- GPG запросит личную информацию, которая сохранится в ключе:
 - Имя (не менее 5 символов).
 - Адрес электронной почты.
 - При вводе email убедитесь, что он соответствует адресу, используемому на GitHub
 - Комментарий. Можно ввести что угодно или нажать клавишу ввода, чтобы оставить это поле пустым.

```

[adchekmarev@adchekmarev ~]$ 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.

gpg: создан каталог '/home/adchekmarev/.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? 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: Chekmarev Alexander
Адрес электронной почты: sasha.cekmarev4@mail.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Chekmarev Alexander <sasha.cekmarev4@mail.ru>"

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

```

Рис. 2.11: Рис 2.4.1: процесс перед генерацией ключа

```

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? o
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
gpg: /home/adchekmarev/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/home/adchekmarev/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/adchekmarev/.gnupg/openpgp-revocs.d/F
94B20E21BBAC097EDAD846A644560047059D595.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2024-02-27 [SC]
      F94B20E21BBAC097EDAD846A644560047059D595
uid           Chekmarev Alexander <sasha.cekmarev4@mail.ru>
sub   rsa4096 2024-02-27 [E]

[adchekmarev@adchekmarev ~]$

```

Рис. 2.12: Рис 2.4.2: сгенерированный ключ

2.5 Настройка github

Чтобы работать с github, нам нужно будет создать аккаунт на сайте <https://github.com/>. У меня аккаунт уже есть с прошлого семестра.



Рис. 2.13: Рис 2.5.1: демонстрация профиля на github

2.6 Добавление PGP ключа в GitHub

- Выводим список ключей и копируем отпечаток приватного ключа: `*gpg --list-secret-keys --keyid-format LONG*`

```
[adchekmarev@adchekmarev ~]$ 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: 1  подписанных: 0  доверие: 0-, 0q, 0n, 0m,
0f, 1u
[keyboard]
-----
sec   rsa4096/644560047059D595 2024-02-27 [SC]
      F94B20E21BBAC097EDAD846A644560047059D595
uid   [ абсолютно ] Chekmarev Alexander <sasha.cekmarev4@mail.ru>
>
ssb   rsa4096/2517F2B5D762FD84 2024-02-27 [E]
```

Рис. 2.14: Рис 2.6.1: вывод списка ключей

- Отпечаток ключа – это последовательность байтов, используемая для идентификации бо
- Формат строки: `sec Алгоритм/Отпечаток_ключа Дата_создания [Флаги] [Годен_до] ID_`

- Скопируем ваш сгенерированный PGP ключ в буфер обмена: `*gpg --armor --export <PGP Fingerprint> | xclip -sel clip*`

```
[adchekmarev@adchekmarev ~]$ gpg --armor --export sasha.cekmarev4@mail.ru | xclip -sel clip
```

Рис. 2.15: Рис 2.6.2: сгенерированный PGP ключ

Перейдем в настройки GitHub (<https://github.com/settings/keys>), нажмем на кнопку New GPG key и вставим полученный ключ в поле ввода.

Add new GPG key

Title

LinuxFedoraSwaySpin39

Key

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGXd7toBEACnIbYTsihplyzI53S5rqMoHVZSI5DOGFC5OZB
guR6N6TMsyjd
polPE2j8QwXGUD40bCmGHimqmsBFw5fXr37IDh8YZZimwgimy
gOOizQ3MumKx5VD
aUGtqC2g/KKBL0xdOnZ315GIoBpBCp5Mx5jgiZPZ5vjzM2/
yProkAXYzHLn87QMN
nGXad2FNHWtKoxueckIPCcyD5Iuqu2ubGcWKtIVEvTy6tZwBOxh
```

Add GPG key

Рис. 2.16: Рис 2.6.3: создание нового GPG ключа на гитхабе

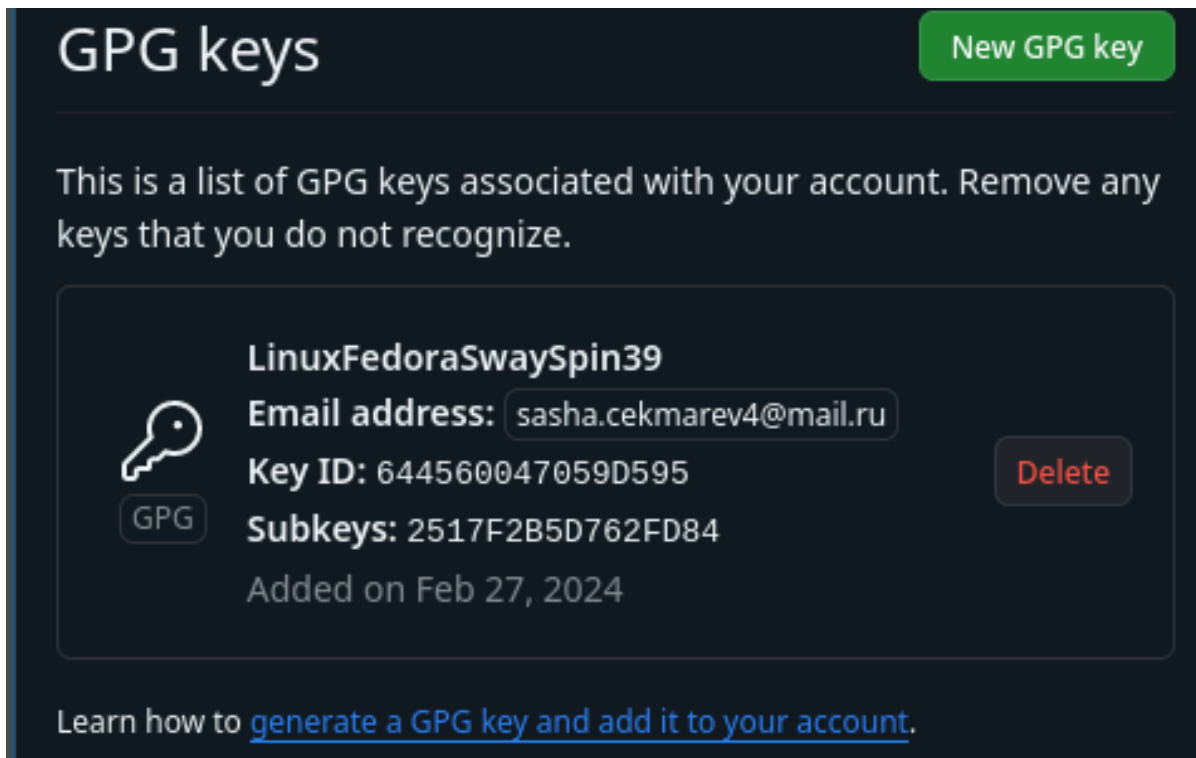


Рис. 2.17: Рис 2.6.4: демонстрация завершения работы с GPG ключом

2.7 Настройка автоматических подписей коммитов git

- Используя введённый email, укажем Git применять его при подписи коммитов:

```
git config --global user.signingkey git config --global commit.gpgsign true git config --global gpg.program $(which gpg2)
```

```
[adchekmarev@adchekmarev ~]$ git config --global user.signingkey sasha.cekmarev4@mail.ru
[adchekmarev@adchekmarev ~]$ git config --global commit.gpgsign true
[adchekmarev@adchekmarev ~]$ git config --global gpg.program $(which gpg2)
[adchekmarev@adchekmarev ~]$
```

Рис. 2.18: Рис 2.7.1: настройка подписей коммитов git

2.8 Настройка gh

Для начала необходимо авторизоваться: `gh auth login` Утилита задаст несколько наводящих вопросов. Авторизоваться можно через браузер.

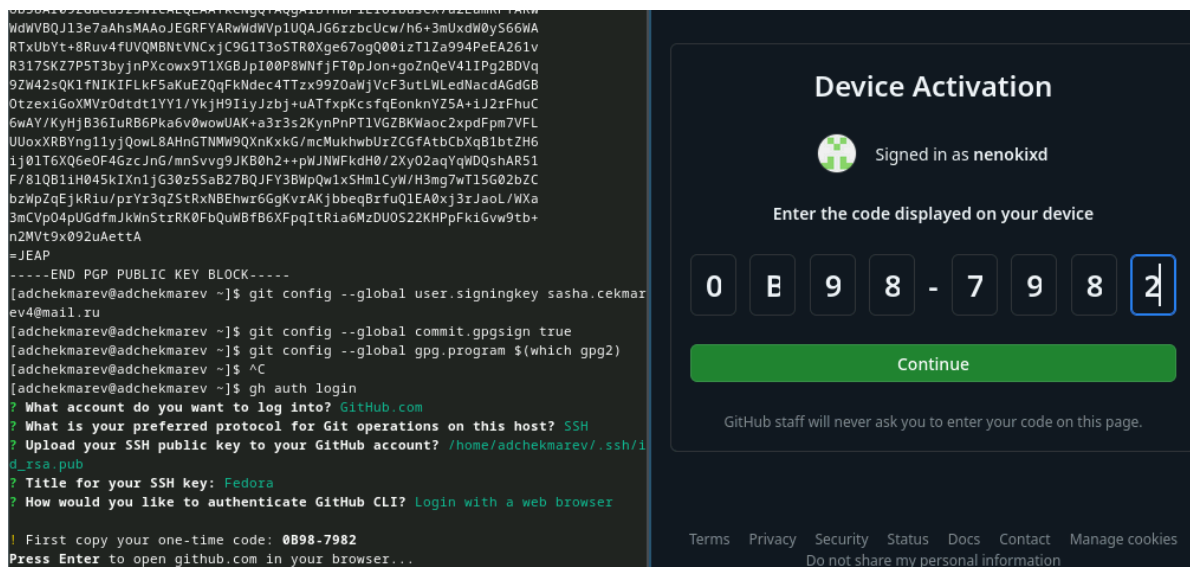


Рис. 2.19: Рис 2.8.1: демонстрация настройки gh

2.9 Шаблон для рабочего пространства

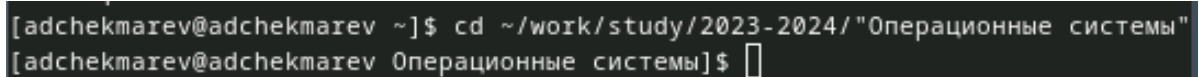
Рабочее пространство для лабораторной работы Репозиторий: <https://github.com/yamadhar/directory-student-template>.

Создание репозитория курса на основе шаблона Необходимо создать шаблон рабочего пространства (см. Рабочее пространство для лабораторной работы). Например, для 2023–2024 учебного года и предмета «Операционные системы» (код предмета `os-intro`) создание репозитория примет следующий вид: `mkdir -p ~/work/study/2023-2024/“Операционные системы”`

```
[adchekmarev@adchekmarev ~]$ mkdir -p ~/work/study/2023-2024/"Операционные системы"
```

Рис. 2.20: Рис 2.9.1: создание каталога

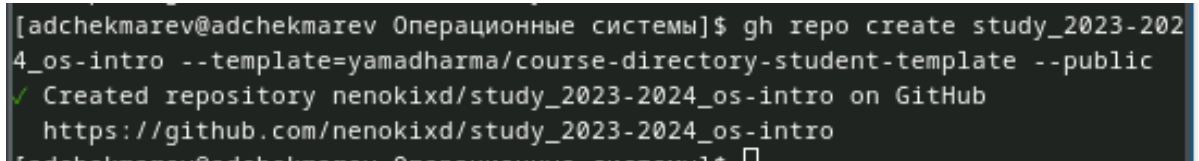
cd ~/work/study/2023-2024/“Операционные системы”



```
[adchekmarev@adchekmarev ~]$ cd ~/work/study/2023-2024/"Операционные системы"
[adchekmarev@adchekmarev Операционные системы]$
```

Рис. 2.21: Рис 2.9.2: переход в каталог

gh repo create study_2023-2024_os-intro --template=yamadharm/course-directory-student-template --public



```
[adchekmarev@adchekmarev Операционные системы]$ gh repo create study_2023-2024_os-intro --template=yamadharm/course-directory-student-template --public
✓ Created repository nenokixd/study_2023-2024_os-intro on GitHub
https://github.com/nenokixd/study_2023-2024_os-intro
```

Рис. 2.22: Рис 2.9.3: создание репозитория

git clone --recursive git@github.com:/study_2023-2024_os-intro.git os-intro

```

[adchekmarev@adchekmarev Операционные системы]$ git clone --recursive git@github.com:nenokixd/study_2023-2024_os-intro.git os-intro
Клонирование в «os-intro»...
The authenticity of host 'github.com (140.82.121.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU
.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
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), 18.59 КиБ | 18.59 МиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-pre
sentation-markdown-template.git) зарегистрирован по пути «template/presentati
on»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laborator
y-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/adchekmarev/work/study/2023-2024/Операционные системы/o
s-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 КиБ | 1.18 МиБ/с, готово.
Определение изменений: 100% (34/34), готово.
Клонирование в «/home/adchekmarev/work/study/2023-2024/Операционные системы/o
s-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 КиБ | 2.47 МиБ/с, готово.
Определение изменений: 100% (52/52), готово.
Submodule path 'template/presentation': checked out '40a1761813e197d00e8443ff
1ca72c60a304f24c'
Submodule path 'template/report': checked out '7c31ab8e5dfa8cdb2d67caeb8a19ef
8028ced88e'

```

Рис. 2.23: Рис 2.9.4: клонирование репозитория

Настройка каталога курса

Перейдите в каталог курса: `cd ~/work/study/2023-2024/“Операционные`

системы"/os-intro

```
[adchekmarev@adchekmarev ~]$ cd ~/work/study/2023-2024/"Операционные системы"/os-intro
[adchekmarev@adchekmarev os-intro]$
```

Рис. 2.24: Рис 2.9.5: переход в другой каталог

Удалите лишние файлы: *rm package.json*

```
[adchekmarev@adchekmarev os-intro]$ rm package.json
```

Рис. 2.25: Рис 2.9.6: удаление файла .json

Создайте необходимые каталоги: *echo os-intro > COURSE* *make*

```
[adchekmarev@adchekmarev os-intro]$ echo os-intro > COURSE
[adchekmarev@adchekmarev os-intro]$ make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare        Generate directories structure
  submodule      Update submules

[adchekmarev@adchekmarev os-intro]$ make prepare
```

Рис. 2.26: Рис 2.9.7: создание каталогов

Отправьте файлы на сервер:

git add . git commit -am 'feat(main): make course structure'

```
[adchekmarev@adchekmarev os-intro]$ git add .
[adchekmarev@adchekmarev os-intro]$ git commit -am 'feat(main): make course structure'
[master fd84143] feat(main): make course structure
361 files changed, 98413 insertions(+), 14 deletions(-)
create mode 100644 labs/README.md
create mode 100644 labs/README.ru.md
create mode 100644 labs/lab01/presentation/Makefile
create mode 100644 labs/lab01/presentation/image/kulyabov.jpg
create mode 100644 labs/lab01/presentation/presentation.md
create mode 100644 labs/lab01/report/Makefile
create mode 100644 labs/lab01/report/bib/cite.bib
create mode 100644 labs/lab01/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab01/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_fignos.py
```

Рис. 2.27: Рис 2.9.8: сохранение и выбор файлов для отправки на сервер

git push

```
[adchekmarev@adchekmarev os-intro]$ git push
Перечисление объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 342.11 КиБ | 2.51 МиБ/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:nenokixd/study_2023-2024_os-intro.git
 fb8ad6f..fd84143 master -> master
```

Рис. 2.28: Рис 2.9.9: отправка файлов на сервер

3 Контрольные вопросы

- 1) Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

Системы контроля версий (VCS) - это инструменты, которые отслеживают изменения в файловой системе с течением времени. Они предназначены для управления изменениями в коде и других файлах проекта, позволяя разработчикам работать над проектом одновременно, откатывать изменения, если что-то идет не так, и отслеживать историю изменений.

- 2) Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

Хранилище (repository): Это место, где хранятся все файлы и история изменений проекта. Commit: Это операция сохранения изменений в репозитории. При коммите фиксируются все изменения, сделанные с момента предыдущего коммита. История (history): Это записи о всех коммитах, сделанных в репозитории. Рабочая копия (working copy): Это копия файлов из репозитория, с которой вы работаете на вашем компьютере.

- 3) Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

Централизованные VCS имеют одно основное хранилище, к которому подключаются все клиенты. Децентрализованные VCS позволяют каждому клиенту иметь собственное полноценное хранилище. Примеры

централизованных VCS: Subversion (SVN). Примеры децентрализованных VCS: Git, Mercurial.

4) Опишите действия с VCS при единоличной работе с хранилищем.

При единоличной работе с хранилищем в VCS вы создаете, изменяете и фиксируете изменения в рабочей копии и затем коммитите их в репозиторий.

5) Опишите порядок работы с общим хранилищем VCS.

Порядок работы с общим хранилищем VCS включает получение последних изменений из репозитория (pull), внесение своих изменений, фиксацию изменений (commit) и отправку их в репозиторий (push).

6) Каковы основные задачи, решаемые инструментальным средством git?

Основные задачи инструмента git:

Отслеживание изменений в файлах.

Управление версиями проекта.

Работа с удаленными репозиториями.

Ветвление и слияние изменений.

Работа с ветками.

7) Назовите и дайте краткую характеристику командам git.

Некоторые команды git:

`git init`: Создает новый репозиторий.

`git add`: Добавляет файлы в индекс для последующего коммита.

`git commit`: Фиксирует изменения в репозитории.

`git push`: Отправляет изменения в удаленный репозиторий.

`git pull`: Получает изменения из удаленного репозитория и объединяет их с текущей веткой.

- 8) Приведите примеры использования при работе с локальным и удалённым репозиториями.

Примеры использования при работе с локальным и удаленным репозиториями:

Локальный: Создание нового репозитория с помощью `git init`.

Удаленный: Клонирование существующего удаленного репозитория с помощью `git clone`.

- 9) Что такое и зачем могут быть нужны ветви (branches)?

Ветви (branches) - это параллельные линии разработки в репозитории, которые позволяют работать над разными фичами или исправлениями, не затрагивая основную ветку. Они могут быть нужны, чтобы изолировать разные функциональные изменения или исправления ошибок.

- 10) Как и зачем можно игнорировать некоторые файлы при commit?

Файлы могут быть проигнорированы при коммите с помощью файла `.gitignore`. В этом файле перечисляются шаблоны файлов или папок, которые не должны быть добавлены в репозиторий.

4 Выводы

Я Изучил идеологию и применение средств контроля версий, а также освоил умения по работе с git.

Список литературы