

Отчёт по лабораторной работе №2

Операционные системы

Калашникова Ольга Сергеевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Установка программного обеспечения	7
3.2	Базовая настройка git	7
3.3	Создайте ключи ssh	8
3.4	Создайте ключи pgr	9
3.5	Добавление PGP ключа в GitHub	11
3.6	Настройка gh	13
3.7	Создание репозитория курса на основе шаблона	14
3.8	Настройка каталога курса	16
4	Контрольные вопросы	20
5	Выводы	23
	Список литературы	24

Список иллюстраций

3.1	Установка git и gh	7
3.2	Задаём имя и почту	8
3.3	utf-8	8
3.4	Настройка верификации	8
3.5	Создание ключей ssh	9
3.6	Генерация ключа pgr	10
3.7	Генерация ключа pgr 2	11
3.8	Список ключей	12
3.9	Копирование ключа	12
3.10	Готовый ключ	13
3.11	Подпись коммитов	13
3.12	Авторизация	14
3.13	Создание папки	14
3.14	Создание репозитория	14
3.15	Клонирование	15
3.16	Проверка	15
3.17	Удаление файла	16
3.18	Проверка	17
3.19	Создание	17
3.20	Проверка	18
3.21	make	18
3.22	git add . , git commit	19
3.23	git push	19

Список таблиц

1 Цель работы

Целью данной работы является изучение идеологии и применение средств контроля версий, а так же освоение умения по работе с git.

2 Задание

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

3 Выполнение лабораторной работы

3.1 Установка программного обеспечения

Установим git при помощи `dnf install git`, а так же установим gh при помощи `dnf install gh` (в моём случае программное обеспечение уже установлено, так как вышло не с первого раза) (рис. 3.1).

```
root
[oskalashnikova@oskalashnikova ~]$ sudo -i
[sudo] пароль для oskalashnikova:
[root@oskalashnikova ~]# dnf install git
Fedora 39 - x86_64 - Updates          20 kB/s | 20 kB      00:01
Fedora 39 - x86_64 - Updates          2.1 MB/s | 3.6 MB     00:01
Последняя проверка окончания срока действия метаданных: 0:00:09 назад, Ч
т 29 фев 2024 07:11:36.
Пакет git-2.43.2-1.fc39.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
[root@oskalashnikova ~]# dnf install gh
Последняя проверка окончания срока действия метаданных: 0:00:32 назад, Ч
т 29 фев 2024 07:11:36.
Пакет gh-2.43.1-1.fc39.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
```

Рис. 3.1: Установка git и gh

3.2 Базовая настройка git

Зададим имя владельца репозитория при помощи `git config --global user.name "lacrimell"` и email при помощи `git config --global user.email "lacrimell@yandex.by"` (рис. 3.2).

```
[root@oskalashnikova ~]# git config --global user.name "lacrimell"
[root@oskalashnikova ~]# git config --global user.email "lacrimell@yandex.by"
[root@oskalashnikova ~]#
```

Рис. 3.2: Задаём имя и почту

Настроим utf-8 в выводе сообщений git при помощи `git config --global core.quotePath false` (рис. 3.3).

```
[root@oskalashnikova ~]# git config --global core.quotePath false
[root@oskalashnikova ~]#
```

Рис. 3.3: utf-8

Зададим имя начальной ветки (будем называть её master) при помощи `git config --global init.defaultBranch master`, далее параметр `autocrlf` с помощью `git config --global core.autocrlf input` и параметр `safecrlf` при помощи `git config --global core.safecrlf warn` (рис. 3.4).

```
[root@oskalashnikova ~]# git config --global init.defaultBranch master
[root@oskalashnikova ~]# git config --global core.autocrlf input
[root@oskalashnikova ~]# git config --global core.safecrlf warn
[root@oskalashnikova ~]#
```

Рис. 3.4: Настройка верификации

3.3 Создайте ключи ssh

Go алгоритму rsa с ключём размером 4096 бит создаём ключ ssh при помощи `ssh-keygen -t rsa -b 4096`, а по алгоритму ed25519 с `ssh-keygen -t ed25519` (рис. 3.5).


```

[root@oskalashnikova ~]# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (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:A0qcztsgy/EVhExHr5jRMtdup4fY12L3aav5AZDchPc root@oskalashnikova
The key's randomart image is:
+---[RSA 4096]-----+
|  o.o+  ..          |
|  .o= +.+          |
|  B = *....         |
|  + O = . E         |
|  o B o S o         |
|  . = + = o         |
|  o o o * = .         |
|  . = o.o           |
|  o+=.              |
+---[SHA256]-----+
[root@oskalashnikova ~]# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
Enter passphrase (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:pPu8jqLb5paQVkwWnnf5t1WpnDjES/X3oJiFwUuxKMIQ root@oskalashnikova
The key's randomart image is:
+--[ED25519 256]--+
|   +o ...          |
|   E ++ o..         |
|   ..o*.o .         |
|   . +.+.. . +      |
|   o ..S.o + = o    |
|   +   ... + B .    |
|   . . . . B =      |
|   . = + o B        |
|   o*o...=.         |
+---[SHA256]-----+
[root@oskalashnikova ~]#

```

Рис. 3.5: Создание ключей ssh

3.4 Создайте ключи ргр

Генерируем ключ при помощи `gpg --full-generate-key` и выбираем тип RSA and RSA, размер 4096, срок действия не истекает никогда (рис. 3.6).

```

[root@oskalashnikova ~]# 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.

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: lacrimell
Адрес электронной почты: lacrimell@yandex.by
Примечание: 123cake!
Вы выбрали следующий идентификатор пользователя:
  "lacrimell (123cake!) <lacrimell@yandex.by>"

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

```

Рис. 3.6: Генерация ключа gpg

Вводим имя, адрес почты используемый на GitHub. Я так же ввела комментарий, чтобы не забыть пароль (рис. 3.7).

```

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: lacrimell
Адрес электронной почты: lacrimell@yandex.by
Примечание: 123cake!
Вы выбрали следующий идентификатор пользователя:
    "lacrimell (123cake!) <lacrimell@yandex.by>"

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: сертификат отзыва записан в '/root/.gnupg/openpgp-revocs.d/46AD87DC
366EA410C2A48EFBC9BF719AA2931FD4.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2024-02-29 [SC]
       46AD87DC366EA410C2A48EFBC9BF719AA2931FD4
uid           lacrimell (123cake!) <lacrimell@yandex.by>
sub   rsa4096 2024-02-29 [E]

[root@oskalashnikova ~]#

```

Рис. 3.7: Генерация ключа pgp 2

3.5 Добавление PGP ключа в GitHub

У меня уже была создана учетная запись и заполнены основные данные на <https://github.com>, так что я перешла сразу к выводу список ключей при помощи `gpg --list-secret-keys --keyid-format LONG` (рис. 3.8).

```
[root@oskalashnikova ~]# 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: 2 подписанных: 0 доверие: 0-, 0q, 0n
, 0m, 0f, 2u
[keyboxd]
-----
sec   rsa4096/C9BF719AA2931FD4 2024-02-29 [SC]
      46AD87DC366EA410C2A48EFBC9BF719AA2931FD4
uid           [ абсолютно ] lacrimell (123cake!) <lacrimell@yandex.
by>
ssb   rsa4096/A04E62633CAA5EA 2024-02-29 [E]

sec   rsa4096/E92D0B168D9F3929 2024-02-27 [SC]
      E021F68B5D6169D503FC39F8E92D0B168D9F3929
uid           [ абсолютно ] lacrimell (cake) <lacrimell@yandex.by>
ssb   rsa4096/F05BC18BC87FD728 2024-02-27 [E]
```

Рис. 3.8: Список ключей

Далее мы должны скопировать сгенерированный PGP ключ в буфер обмена при помощи `gpg --armor --export | xclip -sel clip`, но данная команда у меня не сработала и я копировала вручную (рис. 3.9).

```
[root@oskalashnikova ~]# gpg --armor --export 46AD87DC366EA410C2A48EFBC9
BF719AA2931FD4
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGXgB68BEAC/Psa0y9yotE8Hh/QgJaxnZJ9FRf0rEeaSo+srIMAgwB6ZbYNK
oUw4A4QG8s8CXTMbUsE38XoE07xnQzkwsyQwC2NTDvZ+ETQ0EOy/K++8Fb00bNie7
7hALpjwWkJeh+/tovj5lZWY2cxe9T07j+r5WNb8tlujjkhdYZvegycXagRdf0rrR
xDHYZ0fEmVXMYBYCdQ/9+HxtVqG5pnk2UbZcasqD3XrV7iqtVziwsESdrln/6yMk
pnE5uXKhbYkMPbDrhK4wcgeKm2GaelT/7AAhCb6S6PrPhAHgv1HDp+e7eiwNjkuD
l++rVHZG0oFmbKXYk6gXi/ZhRjxi3UPEWGckj6b0vLZcgyrr0xFvpWobHF69B69K
8fwyxfuwZa1IGiT2dN9nU1+Dv9SLDo3UnQfsE2YiT59fwuelcvh+vRwW/BYsyNUj
c7AHT8B3cbHVra1MvaB0U0R1chuygpE077n8i97NCmdF8fWHkWI XIUkt1utpNqth
```


Рис. 3.9: Копирование ключа

Переходим в настройки GitHub (<https://github.com/settings/keys>), нажимаем на кнопку New GPG key и вставляем полученный ключ в поле ввода (рис. 3.10).

GPG keys

[New GPG key](#)

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



fedora sway
Email address: lacrimell@yandex.by
Key ID: C9BF719AA2931FD4
Subkeys: A04E62633CAA5EA
Added on Feb 29, 2024

Delete

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

Рис. 3.10: Готовый ключ

##Настройка автоматических подписей коммитов git

Используя введённый email, указываем Git применять его при подписи коммитов с помощью `git config --global user.signingkey`, `git config --global commit.gpgsign true`, `git config --global gpg.program $(which gpg2)` (рис. 3.11).

```
[root@oskalashnikova ~]# git config --global user.signingkey C9BF719AA2931FD4
[root@oskalashnikova ~]# git config --global commit.gpgsign true
[root@oskalashnikova ~]# git config --global gpg.program $(which gpg2)
[root@oskalashnikova ~]#
```

Рис. 3.11: Подпись коммитов

3.6 Настройка gh

Авторизируемся при помощи `gh auth login`. Утилита задаёт несколько наводящих вопросов, после авторизируемся через браузер (рис. 3.12).

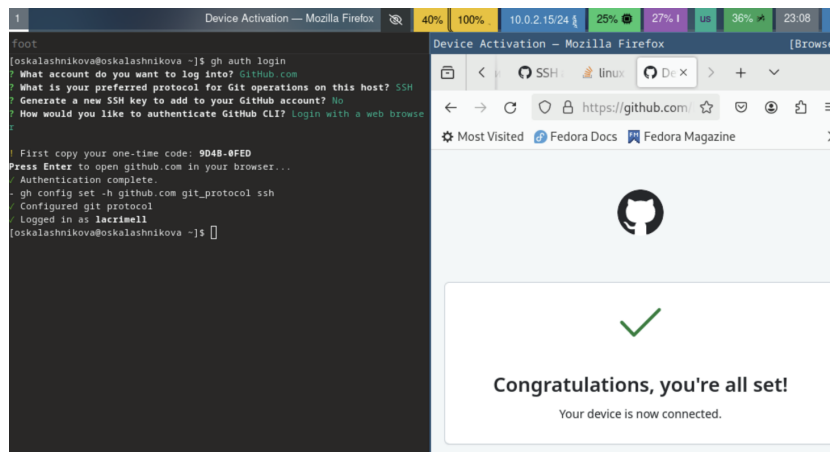


Рис. 3.12: Авторизация

3.7 Создание репозитория курса на основе шаблона

Создаём папку при помощи `mkdir -p ~/work/study/2023-2024/“Операционные системы”` и переходим в неё `cd ~/work/study/2023-2024/“Операционные системы”` (рис. 3.13).

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

Рис. 3.13: Создание папки

Создаём репозиторий `gh repo create study_2023-2024_os-intro --template=yamadharma/course-directory-student-template --public` (рис. 3.14).

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

Рис. 3.14: Создание репозитория

Клонируем его на виртуальную машину `git clone --recursive git@github.com:oskalashnikova/study_2023-2024_os-intro.git os-intro` (рис. 3.15).

```
[oskalashnikova@oskalashnikova Операционные системы]$ git clone --recursive git@github.com:lacrimell/study_2023-2024_os-intro.git
Клонирование в «study_2023-2024_os-intro»...
The authenticity of host 'github.com (140.82.121.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4U
```

Рис. 3.15: Клонирование

Проверяем с помощью `tree` (рис. 3.16).

Левая панель		Файл	Команда	
<-- ...dy_2023-2024_os-intro			-. [^]>	
.и	Имя	Размер	Время	правки
/..		-ВВЕРХ-	мар	1 08:20
/.git		152	мар	1 08:20
/config		24	мар	1 08:20
/template		36	мар	1 08:20
.gitat~butes		1765	мар	1 08:20
.gitignore		4637	мар	1 08:20
.gitmodules		278	мар	1 08:20
CHANGELOG.md		4507	мар	1 08:20
COURSE		0	мар	1 08:20
LICENSE		18657	мар	1 08:20
Makefile		980	мар	1 08:20
README.en.md		152	мар	1 08:20
README~ow.md		5653	мар	1 08:20
README.md		4304	мар	1 08:20
package.json		401	мар	1 08:20

Рис. 3.16: Проверка

3.8 Настройка каталога курса

Переходим в каталог курса при помощи `cd ~/work/study/2022-2023/“Операционные системы”/os-intro` и удаляем лишние файлы с помощью `rm package.json` (рис. 3.17).

```
foot
[oskalashnikova@oskalashnikova ~]$ cd ~/work/study/2023-2024/"Операционн
[oskalashnikova@oskalashnikova study_2023-2024_os-intro]$ rm package.jso
n
[oskalashnikova@oskalashnikova study_2023-2024_os-intro]$
```

Рис. 3.17: Удаление файла

Проверяем, удалился ли файл (рис. 3.18).

foot

Левая панель		Файл	Команда	
< ..dy_2023-2024_os-intro			-.[^]>	
.и	Имя	Размер	Время правки	
/..		-ВВЕРХ-	map	1 08:20
/.git		152	map	1 08:20
/config		24	map	1 08:20
/template		36	map	1 08:20
.gitat~butes		1765	map	1 08:20
.gitignore		4637	map	1 08:20
.gitmodules		278	map	1 08:20
CHANGELOG.md		4507	map	1 08:20
COURSE		0	map	1 08:20
LICENSE		18657	map	1 08:20
Makefile		980	map	1 08:20
README.en.md		152	map	1 08:20
README~ow.md		5653	map	1 08:20
README.md		4304	map	1 08:20

Рис. 3.18: Проверка

Создаём необходимые каталоги при помощи `echo os-intro > COURSE` (рис. 3.19).

```
[oskalashnikova@oskalashnikova study_2023-2024_os-intro]$ echo os-intro
> COURSE
[oskalashnikova@oskalashnikova study_2023-2024_os-intro]$
```

Рис. 3.19: Создание

Проверяем созданся ли (рис. 3.20).

.gitignore	1037	map	1 08:20
.gitmodules	278	map	1 08:20
CHANGELOG.md	4507	map	1 08:20
COURSE	9	map	1 08:33
LICENSE	18657	map	1 08:20
Makefile	980	map	1 08:20

Рис. 3.20: Проверка

Используем make (рис. 3.21).

```
[oskalashnikova@oskalashnikova study_2023-2024_os-intro]$ make
Usage:
  make <target>

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

[oskalashnikova@oskalashnikova study_2023-2024_os-intro]$ make list
  net-admin      Администрирование локальных сетей
  net-os-admin    Администрирование сетевых подсистем
  arch-pc         Архитектура ЭВМ
  sciprogram-intro Введение в научное программирование
  infosec         Информационная безопасность
  computer-practice Компьютерный практикум по статистическому анализу данных
  mathsec         Математические основы защиты информации и информационной безопасности
  mathmod         Математическое моделирование
  simulation-networks Моделирование сетей передачи данных
  sciprogram      Научное программирование
  os-intro        Операционные системы

[oskalashnikova@oskalashnikova study_2023-2024_os-intro]$ make prepare
[oskalashnikova@oskalashnikova study_2023-2024_os-intro]$ make submodule
git submodule update --init --recursive
git submodule foreach 'git fetch origin; git checkout $(git rev-parse --abbrev-ref HEAD); git reset --hard origin/$(git rev-parse --abbrev-ref HEAD); git submodule update --recursive; git clean -dfx'
Entering 'template/presentation'
Указатель HEAD сейчас на коммите 40a1761 Merge branch 'release/1.0.3'
Entering 'template/report'
Указатель HEAD сейчас на коммите 7c31ab8 Merge branch 'release/1.0.4'
[oskalashnikova@oskalashnikova study_2023-2024_os-intro]$
```

Рис. 3.21: make

Отправляем файлы на сервер (рис. 3.22), (рис. 3.23).

```
[oskalashnikova@oskalashnikova study_2023-2024_os-intro]$ git add .
[oskalashnikova@oskalashnikova study_2023-2024_os-intro]$ git commit -am '
feat(main): make course structure'
Текущая ветка: master
Ваша ветка опережает «origin/master» на 1 коммит.
(используйте «git push», чтобы опубликовать ваши локальные коммиты)
```

Рис. 3.22: git add . , git commit

```
[oskalashnikova@oskalashnikova study_2023-2024_os-intro]$ git push
Username for 'https://github.com': lacrimell
Password for 'https://lacrimell@github.com':
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), 341.45 КиБ | 10.67 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 https://github.com/lacrimell/study_2023-2024_os-intro.git
  4fc3003..9459418 master -> master
[oskalashnikova@oskalashnikova study_2023-2024_os-intro]$
```

Рис. 3.23: git push

4 Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Системы контроля версий (VCS) позволяют хранить проекты, отслеживать изменения, управлять релизами и контролировать версии проектов. Они могут использоваться для индивидуальной и командной работы, обеспечивая контроль над изменениями и возможность отката к предыдущим версиям.
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.
 - Хранилище: это репозиторий, где хранятся все файлы и версии проекта.
 - Commit: это фиксация изменений в репозитории.
 - История: представляет собой запись изменений файлов проекта.
 - Рабочая копия: это копия репозитория, с которой работает разработчик.
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные VCS имеют один центральный репозиторий, в то время как децентрализованные VCS позволяют использовать несколько репозиторий и объединять изменения между ними. Примеры: CVS (централизованная) и Git (децентрализованная).

4. Опишите действия с VCS при единоличной работе с хранилищем. Создается репозиторий, в котором разрабатывается проект. После внесения изменений файлы добавляются в репозиторий.
5. Опишите порядок работы с общим хранилищем VCS. Разработчик клонирует репозиторий, вносит изменения, фиксирует их, и выгружает на сервер. Разработчики могут объединять свои версии в общем репозитории.
6. Каковы основные задачи, решаемые инструментальным средством git? Git предназначен для хранения файлов проекта, контроля версий, обеспечения командной работы и отслеживания изменений.
7. Назовите и дайте краткую характеристику командам git.
 - git clone: клонирует проект с сервера на компьютер.
 - git add: добавляет файлы для выгрузки на сервер.
 - git commit: фиксирует изменения репозитория.
 - git push: выгружает изменения на сервер.
 - git pull: получает изменения с сервера.
 - git rm: удаляет файл.
 - git status: отображает статус репозитория.
8. Приведите примеры использования при работе с локальным и удалённым репозиториями.
 - Локальный: git commit -am "added files" — создает коммит.
 - Удаленный: git push — отправляет данные на удаленный сервер.
9. Что такое и зачем могут быть нужны ветви (branches)? Как и зачем можно игнорировать некоторые файлы при commit?

Ветви (branches) представляют собой независимые линии разработки, позволяющие параллельно вносить изменения в проект, которые могут быть объединены позднее.

10. Как и зачем можно игнорировать некоторые файлы при commit? Игнорировать файлы можно, добавив их в .gitignore. Это полезно, когда не нужно добавлять определенные файлы в репозиторий, например, файлы виртуального окружения (например, venv).

5 Выводы

Я изучила идеологию и применение средств контроля версий, а так же освоила умения по работе с git

Список литературы

<https://git-scm.com/book/ru/v2/Основы-Git-Работа-с-удалёнными-репозиториями>

<https://devpractice.ru/git-for-beginners-part-1-what-is-vcs/>

<https://blog.skillfactory.ru/glossary/git/>

туис