

Отчёт по лабораторной работе №2

Дисциплина: Операционные системы

Бызова Мария Олеговна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Установка программного обеспечения.	7
3.2	Базовая настройка git.	8
3.3	Создание ключа ssh.	8
3.4	Создание ключа PGP.	10
3.5	Настройка github.	10
3.6	Добавление ключа PGP в Github	11
3.7	Настройка автоматических подписей коммитов git.	12
3.8	Настройка gh.	12
3.9	Создание репозитория курса на основе шаблона.	14
3.10	Настройка каталога курса.	15
4	Выводы	17
5	Ответы на контрольные вопросы.	18
	Список литературы	21

Список иллюстраций

3.1	Установка git	7
3.2	Установка gh	7
3.3	Задаю имя и email владельца репозитория	8
3.4	Настройка utf-8 в выводе сообщений git	8
3.5	Задаю имя начальной ветки	8
3.6	Задаю параметр autocrlf	8
3.7	Задаю параметр safecrlf	8
3.8	Генерация ключа ssh по алгоритму rsa	9
3.9	Генерация ключа ssh по алгоритму ed25519	9
3.10	Генерация ключа	10
3.11	Аккаунт Github	11
3.12	Вывод списка ключей	11
3.13	Копирование ключа в буфер обмена	11
3.14	Добавление нового ключа	12
3.15	Настройка подписей git	12
3.16	Авторизация	13
3.17	Завершение авторизации	13
3.18	Результат авторизации	13
3.19	Создание и перемещение между директориями	14
3.20	Создание репозитория на основе шаблона репозитория	14
3.21	Клонирование репозитория	15
3.22	Перемещение между директориями	15
3.23	Удаление лишних файлов	16
3.24	Создание необходимых каталогов	16
3.25	Отправка файлов на сервер	16

Список таблиц

1 Цель работы

Целью данной лабораторной работы является изучение идеологии и применение средств контроля версий, 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.

Устанавливаю необходимое программное обеспечение: устанавливаю git (рис. 3.1).

```
mobihzova@mobihzova ~]$ sudo dnf install git
[sudo] пароль для mobihzova:
Fedora 39 - x86_64 - Updates      8.2 kB/s | 18 kB      00:02
Fedora 39 - x86_64 83% [=====] 844 kB/s | 3.0 MB      00:00 ETA
```

Рис. 3.1: Установка git

Устанавливаю gh (рис. 3.2).

```
[mobihzova@mobihzova ~]$ dnf install gh
Ошибка: Эту команду нужно запускать с привилегиями суперпользователя (на
большинстве систем - под именем пользователя root).
[mobihzova@mobihzova ~]$ sudo dnf install gh
Последняя проверка окончания срока действия метаданных: 0:00:46 назад, С
р 21 фев 2024 15:34:35.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий  Размер
=====
Установка:
gh         x86_64       2.43.1-1.fc39 updates      9.1 M
=====
Результат транзакции
=====
Установка 1 Пакет

Объем загрузки: 9.1 M
Объем изменений: 46 M
Продолжить? [д/н]:
```

Рис. 3.2: Установка gh

3.2 Базовая настройка git.

Задаю имя и email владельца репозитория (рис. 3.3).

```
[mobihzova@mobihzova ~]$ git config --global user.name "mobyzova"
[mobihzova@mobihzova ~]$ git config --global user.email "1132236129@pfur.ru"
[mobihzova@mobihzova ~]$
```

Рис. 3.3: Задаю имя и email владельца репозитория

Настраиваю utf-8 в выводе сообщений git (рис. 3.4).

```
ru"
[mobihzova@mobihzova ~]$ git config --global core.quotePath false
[mobihzova@mobihzova ~]$
```

Рис. 3.4: Настройка utf-8 в выводе сообщений git

Задаю имя начальной ветки (будем называть её master) (рис. 3.5).

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

Рис. 3.5: Задаю имя начальной ветки

Задаю параметр autocrlf (рис. 3.6).

```
[mobihzova@mobihzova ~]$ git config --global init.defaultBranch master
[mobihzova@mobihzova ~]$ git config --global core.autocrlf input
[mobihzova@mobihzova ~]$
```

Рис. 3.6: Задаю параметр autocrlf

Задаю параметр safecrlf (рис. 3.7).

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

Рис. 3.7: Задаю параметр safecrlf

3.3 Создание ключа ssh.

Создаю ключ ssh по алгоритму rsa с ключём размером 4096 бит (рис. 3.8).


```

[mobihzova@mobihzova ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/mobihzova/.ssh/id_rsa):
Created directory '/home/mobihzova/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/mobihzova/.ssh/id_rsa
Your public key has been saved in /home/mobihzova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:f12uNvUioTZTEBTu23H8DG5UsPbfGuVtkqX1GjEmCE0 mobihzova@mobihzova
The key's randomart image is:
+---[RSA 4096]---+
|      .E.  .  |
|      +.   o  |
|      . o.  o  |
|      o.. o o  |
|      S o.o 0 *|
|      . oo0 /*|
|      ooo.X.%|
|      =...+0.|
|      . o o+o |
+---[SHA256]-----+
[mobihzova@mobihzova ~]$

```

Рис. 3.8: Генерация ключа ssh по алгоритму rsa

Создаю ключ ssh по алгоритму ed25519 (рис. 3.9).

```

[mobihzova@mobihzova ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/mobihzova/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/mobihzova/.ssh/id_ed25519
Your public key has been saved in /home/mobihzova/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:CfwII4Yhf5AjkJ7Q3GENq0jwnE0B+yHW09XMLA1rpus mobihzova@mobihzova
The key's randomart image is:
+--[ED25519 256]--+
| |=.+++      |
| *+Bo.o.     |
| |=+*o+ @    |
| *=o+ 0 X .   |
| +* o* o S    |
| . *+         |
|  .o          |
|  .           |
|  .E          |
+--[SHA256]-----+
[mobihzova@mobihzova ~]$

```

Рис. 3.9: Генерация ключа ssh по алгоритму ed25519

3.4 Создание ключа PGP.

Генерирую ключ PGP, затем выбираю тип ключа RSA and RSA, задаю максимальную длину ключа: 4096, оставляю неограниченный срок действия ключа. Далее отвечаю на вопросы программы о личной информации (рис. 3.10).

```
[mobihzova@mobihzova ~]$ 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/mobihzova/.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: MariaVuzova
Адрес электронной почты: 1132236129@pfur.ru
```

Рис. 3.10: Генерация ключа

3.5 Настройка github.

У меня уже был создан аккаунт на Github, соответственно, основные данные аккаунта я так же заполняла и проводила его настройку, поэтому просто вхожу в свой аккаунт (рис. 3.11).

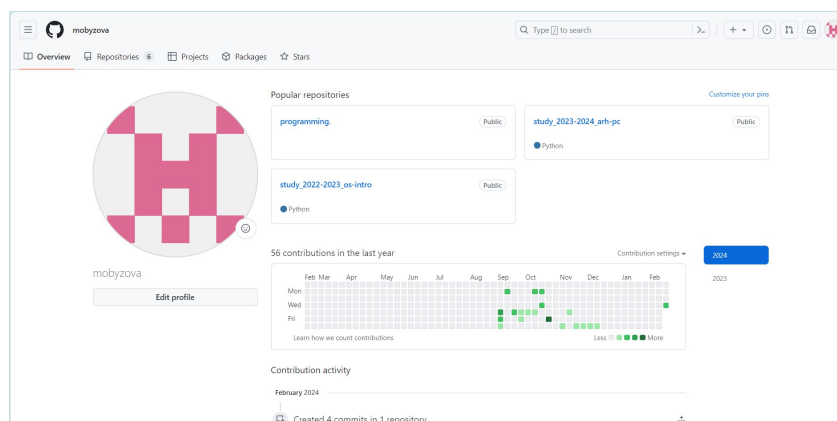


Рис. 3.11: Аккаунт Github

3.6 Добавление ключа PGP в Github

Вывожу список ключей и копирую отпечаток приватного ключ (рис. 3.12).

```
[mobihzova@mobihzova ~]$ 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   rsa4096/94499505E2FAC668 2024-02-21 [SC]
      5C340D3AC5C7AA332BDFE0B94499505E2FAC668
uid           [ абсолютно ] MariaByzova <1132236129@pfur.ru>
ssb   rsa4096/87CE5F52F96B347D 2024-02-21 [E]

[mobihzova@mobihzova ~]$
```

Рис. 3.12: Вывод списка ключей

Скопирую мой сгенерированный PGP ключ в буфер обмена (рис. 3.13).

```
Выполнено!
[mobihzova@mobihzova ~]$ gpg --armor --export 94499505E2FAC668 | xclip -
sel clip
[mobihzova@mobihzova ~]$
```

Рис. 3.13: Копирование ключа в буфер обмена

Перейду в настройки GitHub, нажму на кнопку New GPG key и вставлю полученный ключ в поле ввода (рис. 3.14).

Add new GPG key

Title

Key

```

KnaXjWjEh1Z305Gj0p0c1bZnVQn1v1b0p3WBZ0y00XKf0XW1W0G0Dg
KdUh
IPXQ0IuIYM8xgneF3B/I33rFxNf/
ZSZ5ZD+Ly1pBwzIbFoYdwwElko8xHxxIrew+
Nbij4djychwCf5ct9MwQE+Ipps+cbZ3pWTVYqB81nHkQiveDsZyW8ak
C4ulzlulH
esKsLGEsRZPRzIfc0KQqjCLmuZC+HUj4VM3vb/
rkjc6H4zUr19wucPuqqPppX8=
=XxiY
-----END PGP PUBLIC KEY BLOCK-----

```

Add GPG key

Рис. 3.14: Добавление нового ключа

3.7 Настройка автоматических подписей коммитов git.

Используя введённый email, укажу Git применять его при подписи коммитов (рис. 3.15).

```

[mobihzova@mobihzova ~]$ git config --global user.signingkey 94499505E2F
AC668
[mobihzova@mobihzova ~]$ git config --global commit.gpgsign true
[mobihzova@mobihzova ~]$ git config --global gpg.program $(which gpg2)
[mobihzova@mobihzova ~]$

```

Рис. 3.15: Настройка подписей git

3.8 Настройка gh.

Для начала авторизируюсь: отвечаю на наводящие вопросы от утилиты, в конце выбираю авторизоваться через браузер (рис. 3.16).

```
[mobihzova@mobihzova ~]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 44D7-87A2
Press Enter to open github.com in your browser...
```

Рис. 3.16: Авторизация

Завершаю авторизацию на сайте (рис. 3.17).

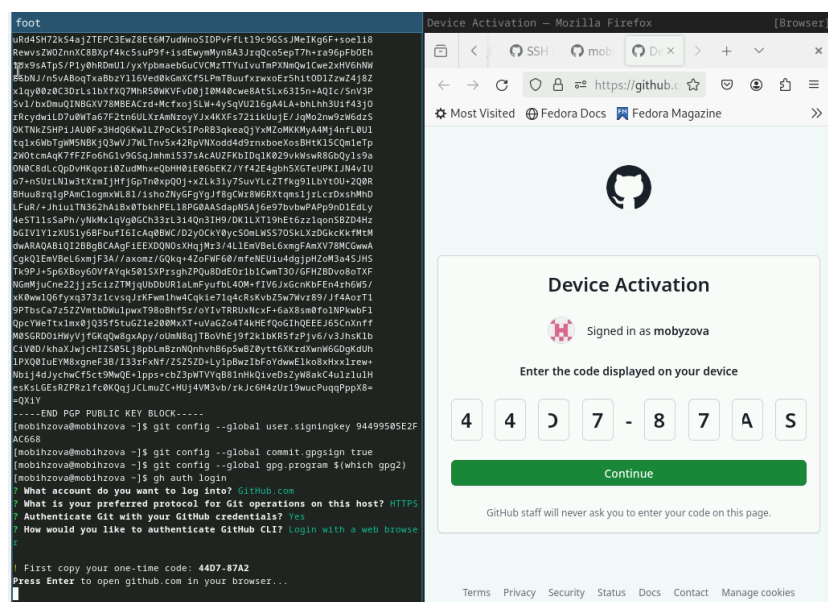


Рис. 3.17: Завершение авторизации

Авторизация прошла успешно (рис. 3.18).

```
Press Enter to open github.com in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
! Authentication credentials saved in plain text
✓ Logged in as mobyzova
[mobihzova@mobihzova ~]$
```

Рис. 3.18: Результат авторизации

3.9 Создание репозитория курса на основе шаблона.

Сначала создаю директорию с помощью утилиты `mkdir` и флага `-p`, который позволяет установить каталоги на всем указанном пути. После этого с помощью утилиты `cd` перехожу в только что созданную директорию (рис. 3.19).

```
[mobihzova@mobihzova ~]$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
[mobihzova@mobihzova ~]$ cd ~/work/study/2022-2023/"Операционные системы"
```

Рис. 3.19: Создание и перемещение между директориями

Далее создаю репозиторий на основе шаблона репозитория (рис. 3.20).

```
[mobihzova@mobihzova Операционные системы]$ gh repo create study_2022-2023_os-intro --template=yamadharm/course-directory-student-template --public
✓ Created repository mobyzova/study_2022-2023_os-intro on GitHub
https://github.com/mobyzova/study_2022-2023_os-intro
[mobihzova@mobihzova Операционные системы]$
```

Рис. 3.20: Создание репозитория на основе шаблона репозитория

Клонирую репозиторий (рис. 3.21).

```
[mobihzova@mobihzova Операционные системы]$ gh repo create study_2022-2023_os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository mobyzova/study_2022-2023_os-intro on GitHub
https://github.com/mobyzova/study_2022-2023_os-intro
[mobihzova@mobihzova Операционные системы]$ git clone --recursive https://github.com/mobyzova/study_2022-2023_os-intro
Клонирование в «study_2022-2023_os-intro»...
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 | 6.20 МиБ/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/mobihzova/work/study/2022-2023/Операционные системы/study_2022-2023_os-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 | 973.00 КиБ/с, готово.
Определение изменений: 100% (34/34), готово.
Клонирование в «/home/mobihzova/work/study/2022-2023/Операционные системы/study_2022-2023_os-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 | 1.64 МиБ/с, готово.
Определение изменений: 100% (52/52), готово.
Submodule path 'template/presentation': checked out '40a1761813e197d00e8443ff1ca72c60a304f24c'
Submodule path 'template/report': checked out '7c31ab8e5dfa8cdb2d67caeb8a19ef8028ced88e'
[mobihzova@mobihzova Операционные системы]$
```

Рис. 3.21: Клонирование репозитория

3.10 Настройка каталога курса.

Перехожу в каталог курса (рис. 3.22).

```
[mobihzova@mobihzova ~]$ cd /home/mobihzova/work/study/2022-2023/"Операционные системы"/study_2022-2023_os-intro/
[mobihzova@mobihzova study_2022-2023_os-intro]$
```

Рис. 3.22: Перемещение между директориями

Удаляю лишние файлы (рис. 3.23).

```
[mobihzova@mobihzova study_2022-2023_os-intro]$ rm package.json
[mobihzova@mobihzova study_2022-2023_os-intro]$
```

Рис. 3.23: Удаление лишних файлов

Создаю необходимые каталоги (рис. 3.24).

```
[mobihzova@mobihzova study_2022-2023_os-intro]$ echo os-intro > COURSE
[mobihzova@mobihzova study_2022-2023_os-intro]$ make
Usage:
  make <target>

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

[mobihzova@mobihzova study_2022-2023_os-intro]$
```

Рис. 3.24: Создание необходимых каталогов

Отправляю файлы на сервер (рис. 3.25).

```
[mobihzova@mobihzova study_2022-2023_os-intro]$ git add .
[mobihzova@mobihzova study_2022-2023_os-intro]$ git commit -am 'feat(main): make course structure'
error: gpg failed to sign the data:
[GNUPG:] KEY_CONSIDERED 5C340D3AC5C7AA332BDFFE0B94499505E2FAC668 2
[GNUPG:] BEGIN_SIGNING H8
[GNUPG:] PINENTRY_LAUNCHED 2196 gnome3 1.2.1 - foot :0 - 1000/1000 0
gpg: подписать не удалось: Операция отменена
[GNUPG:] FAILURE sign 83886179
gpg: signing failed: Операция отменена

fatal: сбой записи объекта коммита
[mobihzova@mobihzova study_2022-2023_os-intro]$ git add .
[mobihzova@mobihzova study_2022-2023_os-intro]$ git commit -am 'feat(main): make course structure'
[master 4dbe485] feat(main): make course structure
 2 files changed, 1 insertion(+), 14 deletions(-)
 delete mode 100644 package.json
[mobihzova@mobihzova study_2022-2023_os-intro]$ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 949 байтов | 949.00 КиБ/с, готово.
Всего 3 (изменений 1), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/mobyzova/study_2022-2023_os-intro
 12a30fb..4dbe485 master -> master
[mobihzova@mobihzova study_2022-2023_os-intro]$
```

Рис. 3.25: Отправка файлов на сервер

4 Выводы

В ходе выполнения данной лабораторной работы я изучила идеологию и применение средств контроля версий, освоила умения по работе с git.

5 Ответы на контрольные вопросы.

1. Системы контроля версий (VCS) - программное обеспечение для облегчения работы с изменяющейся информацией. Они позволяют хранить несколько версий изменяющейся информации, одного и того же документа, может предоставить доступ к более ранним версиям документа. Используется для работы нескольких человек над проектом, позволяет посмотреть, кто и когда внес какое-либо изменение и т. д. VCS применяются для: Хранения полной истории изменений, сохранения причин всех изменений, поиска причин изменений и совершивших изменение, совместной работы над проектами.
2. Хранилище – репозиторий, хранилище версий, в нем хранятся все документы, включая историю их изменения и прочей служебной информацией. commit – отслеживание изменений, сохраняет разницу в изменениях. История – хранит все изменения в проекте и позволяет при необходимости вернуться/обратиться к нужным данным. Рабочая копия – копия проекта, основанная на версии из хранилища, чаще всего последней версии.
3. Централизованные VCS (например: CVS, TFS, AccuRev) – одно основное хранилище всего проекта. Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет, затем добавляет изменения обратно в хранилище. Децентрализованные VCS (например: Git, Bazaar) – у каждого пользователя свой вариант репозитория (возможно несколько вариантов), есть возможность добавлять забирать

изменения из любого репозитория. В отличие от классических, в распределенных (децентрализованных) системах контроля версий центральный репозиторий не является обязательным.

4. Сначала создается и подключается удаленный репозиторий, затем по мере изменения проекта эти изменения отправляются на сервер.
5. Участник проекта перед началом работы получает нужную ему версию проекта в хранилище, с помощью определенных команд, после внесения изменений пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются. К ним можно вернуться в любой момент.
6. Хранение информации о всех изменениях в вашем коде, обеспечение удобства командной работы над кодом.
7. 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. `git push -all` отправляем из локального репозитория все сохраненные изменения в центральный репозиторий, предварительно создав локальный репозиторий и сделав предварительную конфигурацию.
9. Ветвление - один из параллельных участков в одном хранилище, исходящих из одной версии, обычно есть главная ветка. Между ветками, т. е. их концами возможно их слияние. Используются для разработки новых функций.
10. Во время работы над проектом могут создаваться файлы, которые не следуют добавлять в репозиторий. Например, временные файлы. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл `.gitignore` с помощью сервисов.

Список литературы

1. Лабораторная работа №2 [Электронный ресурс] URL: <https://esystem.rudn.ru/mod/page/view.php?id=11111>