

# Отчёт по Лабораторной работе №2

Операционные системы

Петрова Алевтина Александровна

## Содержание

### 1 Цель работы

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

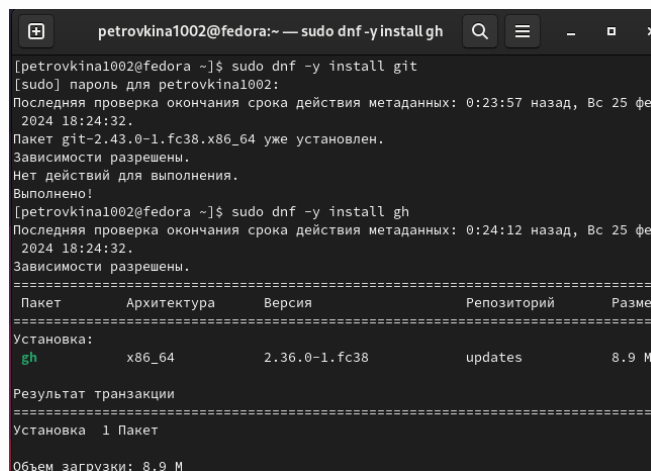
### 2 Задание

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 выполнений заданий по предмету

# 3 Выполнение лабораторной работы

## 3.1 Установка программного обеспечения

Устанавливаю необходимое программное обеспечение git и gh через терминал с помощью команд: `dnf install git` и `dnf install gh` (рис 1.)

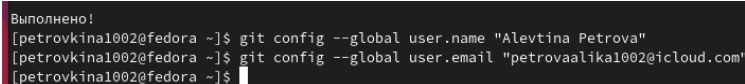


```
[petrovkina1002@fedora ~]$ sudo dnf -y install git
[sudo] пароль для petrovkina1002:
Последняя проверка окончания срока действия метаданных: 0:23:57 назад, Вс 25 фев 2024 18:24:32.
Пакет git-2.43.0-1.fc38.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
[petrovkina1002@fedora ~]$ sudo dnf -y install gh
Последняя проверка окончания срока действия метаданных: 0:24:12 назад, Вс 25 фев 2024 18:24:32.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий  Размер
=====
Установка:
gh          x86_64       2.36.0-1.fc38 updates      8.9 М
=====
Результат транзакции
=====
Установка 1 Пакет
=====
Объем загрузки: 8.9 М
```

Рис. 1 Установка git и gh

## 3.2 Базовая настройка git

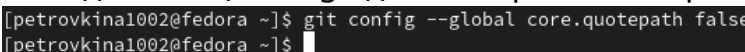
Задаю в качестве имени и email владельца репозитория свои имя, фамилию и электронную почту (рис.2)



```
Выполнено!
[petrovkina1002@fedora ~]$ git config --global user.name "Alevtina Petrova"
[petrovkina1002@fedora ~]$ git config --global user.email "petrovaalika1002@icloud.com"
[petrovkina1002@fedora ~]$
```

Настраиваю utf-8 в

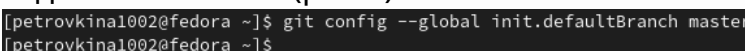
выводе сообщений git для их верного отображения (рис.3)



```
[petrovkina1002@fedora ~]$ git config --global core.quotepath false
[petrovkina1002@fedora ~]$
```

Начальной ветке

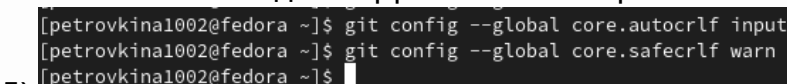
задаю имя master (рис.4)



```
[petrovkina1002@fedora ~]$ git config --global init.defaultBranch master
[petrovkina1002@fedora ~]$
```

Задаю параметры

autocrlf и safecrlf для корректного отображения в конце строки (рис.



```
[petrovkina1002@fedora ~]$ git config --global core.autocrlf input
[petrovkina1002@fedora ~]$ git config --global core.safecrlf warn
[petrovkina1002@fedora ~]$
```

5)

## Создание SSH

ключа Создаю ключ SSH размером 4096 бит по алгоритму rsa (рис.6)

```
[petrovkina1002@fedora ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/petrovkina1002/.ssh/id_rsa):
Created directory '/home/petrovkina1002/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/petrovkina1002/.ssh/id_rsa
Your public key has been saved in /home/petrovkina1002/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:lnIzFpPtNkHF1BUcyQONEM2J0KAYyCv8wXm2SoFyDw petrovkina1002@fedora
The key's randomart image is:
+---[RSA 4096]-----+
| . . . .+o*0oB+=|
| .+ o ...*o..=* |
|* . . . B... . |
|oE. + o+ |
| o.o S. . |
| o . |
| + * |
| . . o |
| . |
+-----[SHA256]-----+
[petrovkina1002@fedora ~]$
```

Создаю ключ SSH

по алгоритму ed25519 (рис.7)

```
[petrovkina1002@fedora ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/petrovkina1002/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/petrovkina1002/.ssh/id_ed25519
Your public key has been saved in /home/petrovkina1002/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:8haH3jHRH4zPvQm9WomLvDnNbWtyWJE+T/An6SmJxSg petrovkina1002@fedora
The key's randomart image is:
+--[ED25519 256]--+
|
|      o
|      o o .
|      o + ++
|      . S .o=.o=
|      +E*. oo*=+
|      *..=.*.*.
|      . oooB+B .
|      =oo*..
+-----[SHA256]-----+
[petrovkina1002@fedora ~]$
```

## Создание ключа

GPG

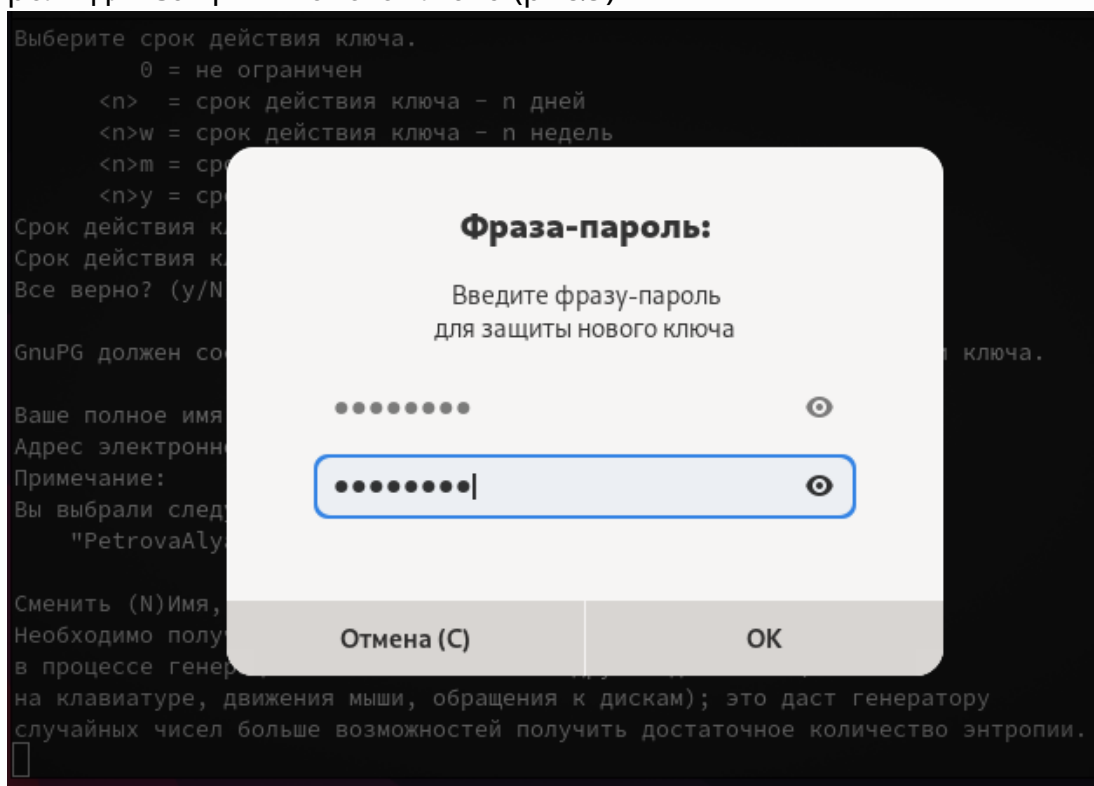
Генерирую ключ GPG, затем выбираю тип ключа RSA and RSA, затем задаю максимальную длину ключа: 4096, оставляю неограниченный срок действия ключа. Далее отвечаю на вопросы программы о личной информации (рис.8)

```
[petrovkina1002@fedora ~]$ gpg --full-generate-key
gpg (GnuPG) 2.4.0; 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/petrovkina1002/.gnupg'
gpg: создан щит с ключами '/home/petrovkina1002/.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
```

Ввожу фразу-па-

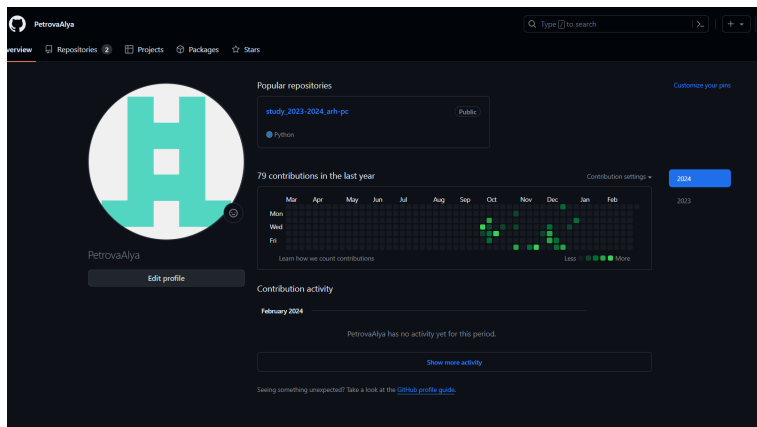
роль для защиты нового ключа (рис.9)



{#fig:009 width=70%

### 3.3 Регистрация на GitHub

У меня уже был создан аккаунт в GitHub, соответственно, основные данные аккаунта я так же заполняла и проводила его настройку, поэтому просто вхожу в свой аккаунт (рис.10)



## Добавление

ключа GPG на GitHub Вывожу список созданных ключей в терминал, ищу в результате запроса отпечаток ключа, копирую его в буфер обмена (рис.11)

```
[petrovkina1002@fedora ~]$ 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
/home/petrovkina1002/.gnupg/pubring.kbx
-----
sec   rsa4096/38783C7D6CFB8C5B 2024-02-25 [SC]
      4418C4286FA11B1624B2DF4A38783C7D6CFB8C5B
uid           [ абсолютно ] PetrovaAlya <petrovaalika1002@icloud.com>
ssb   rsa4096/7FADDC2B4F632DD5 2024-02-25 [E]

[petrovkina1002@fedora ~]$
```

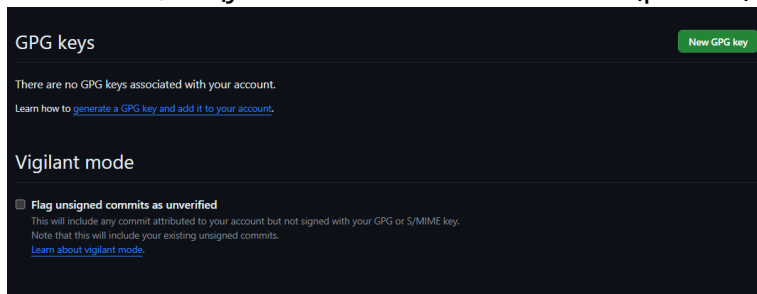
Ввожу в терминал

команду, с помощью которой копирую сам ключ GPG в буфер обмена, за это отвечает команда xclip

```
[petrovkina1002@fedora ~]$ gpg --armor --export 38783C7D6CFB8C5B | xclip -sel clip
```

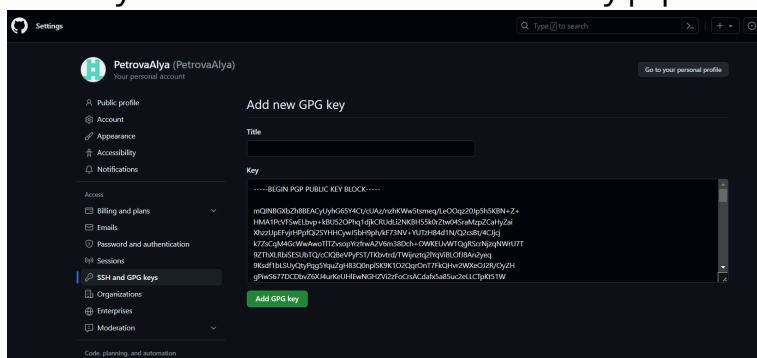
Открываю настрой-

ки GitHub, ищу обновление GPG ключа (рис.13)



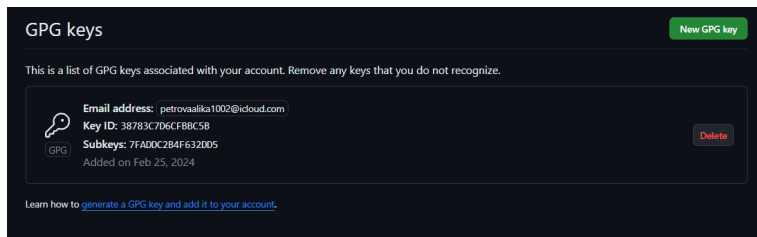
Нажимаю на “New

GPG key” и вставляю в поле ключ из буфера обмена (рис.14)



Так, я добавила

ключ GPG в GitHub(рис.15)



## Настроить подписи Git Настраиваю автоматические подписи коммитов git: используя введенный ранее email, указываю git использовать его при создании подписей коммитов (рис.16)

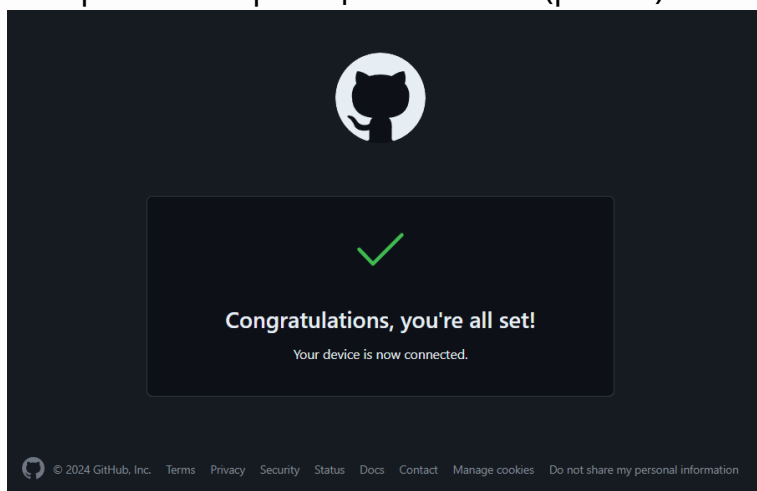
```
[petrovkina1002@fedora ~]$ gpg --armor --export 38783C7D6CFB8C5B | xclip -sel clip
[petrovkina1002@fedora ~]$ git config --global user.signingkey 38783C7D6CFB8C5B
[petrovkina1002@fedora ~]$ git config --global commit.gpgsign true
[petrovkina1002@fedora ~]$ git config --global gpg.program $(which gpg2)
[petrovkina1002@fedora ~]$
```

## 3.4 Настройка gh

Начинаю авторизацию в gh, отвечаю на наводящие вопросы от утилиты, в конце выбираю авторизоваться через браузер (рис.17).

```
[petrovkina1002@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? Login with a web browser
! First copy your one-time code: 1CAB-A614
Press Enter to open github.com in your browser...
```

Завершаю авторизацию на сайте (рис.18).



Вижу сообщение о завершении авторизации под именем PetrovaAlya (рис.19).

```
✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as PetrovaAlya
[petrovkina1002@fedora ~]$
```

## 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 “Операционные системы”. Далее в терминале ввожу команду `gh repo create study_2023-2024_os-intro -template yamadharm/course-directory-student-trmplate -public`, чтобы создать репозиторий на основе шаблона репозитория. После этого клонирую репозиторий к себе в директорию, я указываю ссылку с протоколом `https`, а не `ssh`, потому что при авторизации в `gh` выбрала протокол `https` (рис.20).

```
[petrovkina1002@fedora Операционные системы]$ git clone --recursive git@github.com:<PetrovaAlya>/st
24_os-intro.git os-intro
bash: PetrovaAlya: Нет такого файла или каталога
[petrovkina1002@fedora Операционные системы]$ git clone --recursive https://github.com/PetrovaAlya/
2024_os-intro.git os-intro
Клонирование в «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), 18.69 Киб | 1.86 МБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharm/academic-presentation-markdown-tem
зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharm/academic-laboratory-report-template.git)
рован по пути «template/report»
```

Рис. 20 Создание репозитория

Перехожу в каталог курса с помощью утилиты `cd`, проверяю содержание каталога с помощью утилиты `ls` (рис.21).

```
[petrovkina1002@fedora Операционные системы]$ cd os-intro
[petrovkina1002@fedora os-intro]$ ls
CHANGELOG.md  COURSE  Makefile  README.en.md  README.md
config        LICENSE  package.json  README.git-flow.md  template
[petrovkina1002@fedora os-intro]$
```

Рис.21 Перемещение между директориями

Удаляю лишние файлы с помощью утилиты `rm`, далее создаю необходимые каталоги используя `makefile` (рис.22).

```
config        LICENSE  package.json  README.git-flow.md  template
[petrovkina1002@fedora os-intro]$ rm package.json
[petrovkina1002@fedora os-intro]$ echo os-intro > COURSE
[petrovkina1002@fedora os-intro]$ make
```

Рис.22 Удаление файлов и создание каталогов

Добавляю все новые файлы для отправки на сервер с помощью команды `git add` и комментирую с помощью `git commit` (рис.23)

```
[petrovkina1002@fedora os-intro]$ git add .
[petrovkina1002@fedora os-intro]$ git commit -am 'feat(main): make course structure'
[main 0a2a178] feat(main): make course structure
 2 files changed, 1 insertion(+), 14 deletions(-)
 delete mode 100644 package.json
[petrovkina1002@fedora os-intro]$
```

Отправляю файлы

на сервер с помощью `git push` (рис.24)

```
[petrovkina1002@fedora os-intro]$ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 955 байтов | 86.00 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/PetrovaAlya/study_2023-2024_os-intro.git
 e59ba78..0a2a178 master -> master
[petrovkina1002@fedora os-intro]$
```

## 4 Выводы

При помощи данной лабораторной работы я изучила идеологию и применение средств контроля версий, освоила умение 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 и забирать изменения из любого репозитория. В отличие от классических, в распределённых (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`

1. `git push -all` отправляем из локального репозитория все сохранённые изменения в центральный репозиторий, предварительно создав локальный репозиторий и сделав предварительную конфигурацию.

2. 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.
3. 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 <https://esystem.rudn.ru/mod/page/view.php?id=1098790>