

Лабораторная работа №2

Операционные системы

Панина Ж. В.

04 марта 2025

Российский университет дружбы народов, Москва, Россия

Информация

- Панина Жанна Валерьевна
- НКАбд-02-24, студ. билет № 1132246710
- студент направления “Компьютерные и информационные науки”
- Российский университет дружбы народов
- 1132246710@pfur.ru
- https://github.com/zvpanina/study_2024-2025_os-intro

Вводная часть

В современном мире разработки программного обеспечения использование систем контроля версий стало неотъемлемой частью работы программистов. Git является одной из самых популярных и мощных распределённых систем контроля версий, обеспечивающей удобную совместную работу, отслеживание изменений в коде и автоматизацию процессов развертывания. Навыки работы с Git, а также использование SSH и GPG-ключей для безопасного взаимодействия с удалёнными репозиториями, являются важными для любого разработчика. Освоение этих инструментов повышает эффективность командной работы, обеспечивает безопасность кода и позволяет интегрироваться в профессиональные рабочие процессы.

Объект исследования:

Система контроля версий Git и средства обеспечения безопасности при работе с репозиториями (SSH и GPG-ключи).

Предмет исследования:

Методы управления версиями кода в Git, настройка и использование SSH и GPG-ключей для аутентификации и подписи коммитов, а также работа с удалёнными репозиториями.

Цель работы - изучить идеологию и применения средств контроля версий, освоить умения по работе с git.

Задачи:

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

Материалы:

- Операционная система (Fedora, установленная в VirtualBox)
- Git
- SSH-ключи для безопасного доступа к репозиторию
- GPG-ключи для подписи коммитов

Методы:

- Установка и настройка Git
- Генерация и настройка SSH-ключей для аутентификации
- Генерация GPG-ключей и настройка подписи коммитов
- Клонирование удалённого репозитория
- Работа с ветками, коммитами и отправка изменений в удалённый репозиторий
- Проверка подписей коммитов и настройка доверенных ключей

Теоретическое введение

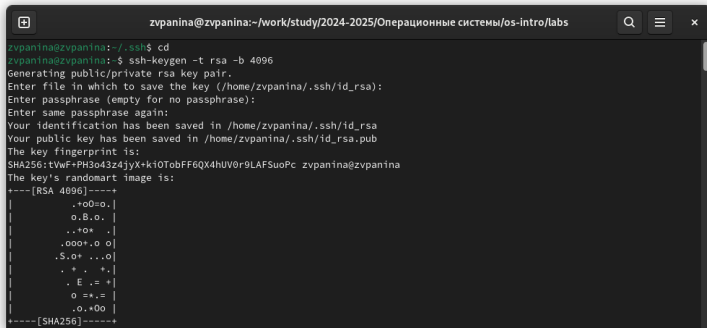
Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля 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.

Выполнение лабораторной работы

Создание ключа SSH

Поскольку я уже была зарегистрирована на GitHub в первом семестре, начинаю работу с создания ключа SSH.

A terminal window with a dark background and light text. The title bar shows the user 'zvpanina' and the directory '~/work/study/2024-2025/Операционные системы/os-intro/labs'. The terminal output shows the execution of 'ssh-keygen -t rsa -b 4096', followed by prompts for a file name, passphrase, and confirmation. It then displays the key fingerprint (SHA256) and a randomart image for the RSA 4096 key.

```
zvpanina@zvpanina:~/work/study/2024-2025/Операционные системы/os-intro/labs
zvpanina@zvpanina:~/.ssh$ cd
zvpanina@zvpanina:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/zvpanina/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/zvpanina/.ssh/id_rsa
Your public key has been saved in /home/zvpanina/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:tVwF+PH3o43z4jyX+ki0TobFF6QX4hUV0r9LAFSuoPc zvpanina@zvpanina
The key's randomart image is:
+---[RSA 4096]---+
|      .oO=o. |
|      o.B.o. |
|      ..+O*  |
|      .ooo+.o |
|      .S.o+ ...o|
|      . + .  +. |
|      . E . = + |
|      o =* =  |
|      .o.*0o  |
+---[SHA256]-----+
```

Рис. 1: Создание ключа SSH

Создание ключа GPG

1. Создаю ключ GPG, указав необходимые параметры.

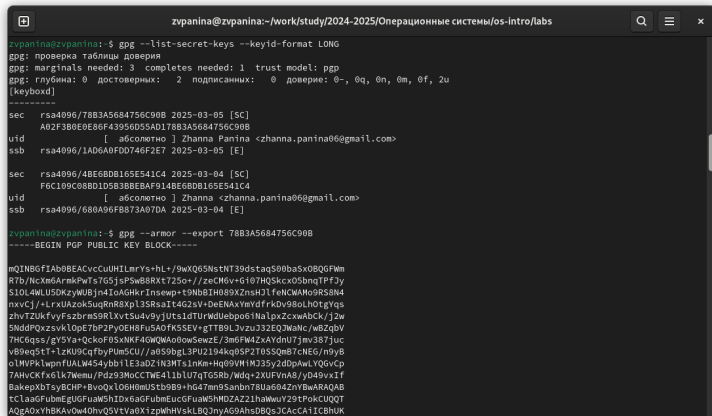
```
zvpanina@zvpanina:~/work/study/2024-2025/Операционные системы/os-intro/labs
zvpanina@zvpanina:~$ gpg --full-generate-key
gpg (GnuPG) 2.4.4; Copyright (C) 2024 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

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: Zhanna Panina
Адрес электронной почты: zhanna.panina06@gmail.com
Примечание:
```

2. Вывожу список ключей и копирую отпечаток приватного ключа. Копирую сгенерированный ключ GPG.



```
zvpanina@zvpanina:~/work/study/2024-2025/Операционные системы/os-intro/labs
zvpanina@zvpanina:~$ 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 rsa4096/78B3A5684756C90B 2025-03-05 [SC]
    A02F3B0E0E86F43956D55AD178B3A5684756C90B
uid          [ а6солютно ] Zhanna Panina <zhanna.panina06@gmail.com>
ssb rsa4096/1AD6A9FDD746F2E7 2025-03-05 [E]

sec rsa4096/4BE6BDB165E541C4 2025-03-04 [SC]
    F6C109C08BD1D583BBEBAF914BE6BDB165E541C4
uid          [ а6солютно ] Zhanna <zhanna.panina06@gmail.com>
ssb rsa4096/680A96F8873A07DA 2025-03-04 [E]

zvpanina@zvpanina:~$ gpg --armor --export 78B3A5684756C90B
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGFIAb0BEACvcCuUHLmrYs+hL+/9wXQ65NstNT39dstaqS00baSx0BQGFwM
R7b/NcXm6ArmKpWtsTG5jsPswB8Rxt725o+//zeCH6v+Gi07HQ5kcx05bnqTPfJy
S10L4WL5DKzyWUBjn4IoAGHkrInsewp+t9Nb8IH089XZnsHJlfeNCWAMo9RS8M4
nxvCj/+LrxUAzok5uqRnR8Xp13SRsaTt4G2sV+DeENaxYmYdfkrDv98oLh0tgYqs
zhvT2UkfvyFs2brms9RLXvtSud4v9yJuts1dTWrdUebpo6iNaIpxZcxwAbCk/j2w
5NddPQxzsvkL0pE7bP2PyOE8Fu5A0fK5SEV+gTTB9LJvzu332EQJwaNc/wBZqbV
7HC6qss/gY9Ya+QckoF05xNKF4GwQMAo0owSewzE/3m6FW4ZxAYdnU7jnv387juc
vB9eq5tT+LzKU9CqfbyPum5CU//a0S9bgL3PU2194k0SP2T0SSQmB7cNEG/n9yB
oLMVPkLwpnFUALW454ybbiLE3aDZiN3MTs1nKm+Hq09VMiM335y2dDpAwLYQGvCp
7AHvCKfx6lk7Wemu/Pdz93M0CCTWtE4L1bLU7qTG5Rb/wdq+2XUFVnA8/yD49vxIf
BakepXbTsybCHP+VvoQxL06H0mJstb9B9+hG47mn9Sanbn78Ua604ZnYBwARAQAB
tClaaGFubmEgUGFuYW5hIDx6aGfuBmEucGFua5hMDAZ21haWwuY292P0kCUQQT
AQGA0xYhBKA0w0hvQ5YtVa0XiZpWHVskLBQJnyAG9AhsDBQsJCAcCAIICBUK
```

Рис. 3: Ключ GPG

3. Перехожу в настройки GitHub, вставляю ключ.

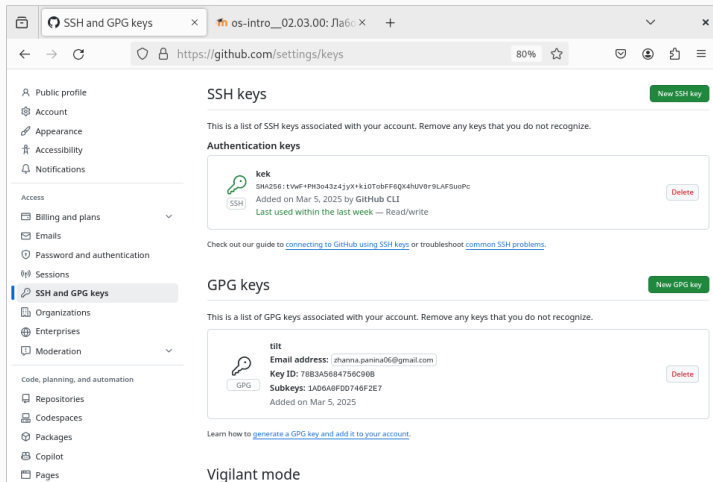
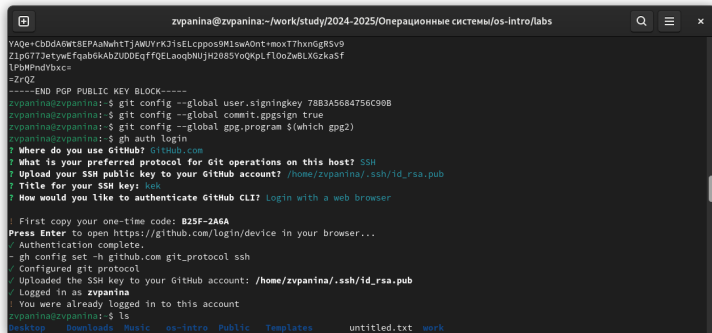


Рис. 4: Ключи в GitHub

Настройка подписей Git

Используя введённый email, указываю Git применять его при подписи коммитов, авторизуюсь.



```
zvpanina@zvpanina:~/work/study/2024-2025/Операционные системы/os-intro/labs
YAQe+CbDDA6Wt8EPAaNwhtTjAWUYrKJisELcpPos9M1swA0nt+moxt7hxnGgRSv9
Z1pG77JetywEfqa6kAbZUDDEqffQELaoqbNUjH2085YoQKpLFl0oZwBLXGzkaSf
1PbMPndYbxc=
=ZrQZ
-----END PGP PUBLIC KEY BLOCK-----
zvpanina@zvpanina:~$ git config --global user.signingkey 78B3A5684756C90B
zvpanina@zvpanina:~$ git config --global commit.gpgsign true
zvpanina@zvpanina:~$ git config --global gpg.program $(which gpg2)
zvpanina@zvpanina:~$ gh auth login
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /home/zvpanina/.ssh/id_rsa.pub
? Title for your SSH key: kek
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: B25F-2A6A
Press Enter to open https://github.com/login/device in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
✓ Uploaded the SSH key to your GitHub account: /home/zvpanina/.ssh/id_rsa.pub
✓ Logged in as zvpanina
! You were already logged in to this account
zvpanina@zvpanina:~$ ls
Desktop  Downloads  Music  os-intro  Public  Templates  untitled.txt  work
```

Рис. 5: Настройка подписей Git

1. Создаю каталог “Операционные системы”, перехожу в него и клонирую репозиторий.

```
root@zvpanina:~/work/study/2024-2025/Операционные системы# gh repo create study_2024-2025_os-intro --te
mplate=yamadharma/course-directory-student-template --public
✓ Created repository zvpanina/study_2024-2025_os-intro on GitHub
https://github.com/zvpanina/study_2024-2025_os-intro
root@zvpanina:~/work/study/2024-2025/Операционные системы# git clone --recursive git@github.com:https:/
/github.com/zvpanina/study_2024-2025_os-intro.git os-intro
Клонирование в «os-intro»...
```

Рис. 6: Клонирование репозитория

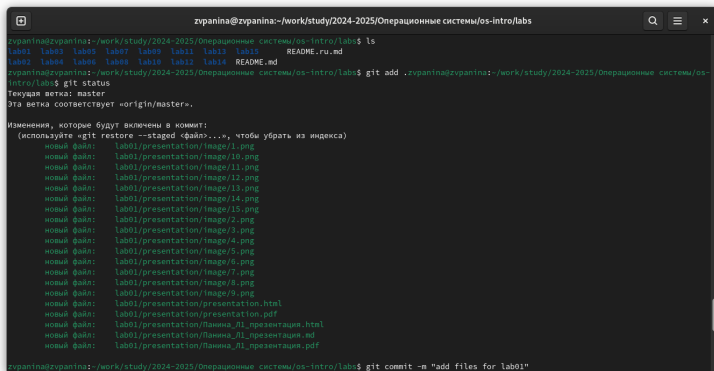
2. Удаляю лишние файлы, создаю необходимые каталоги.

```
zvpanina@zvpanina:~/work/study/2024-2025/Операционные системы$ rm package.json
rm: невозможно удалить 'package.json': Нет такого файла или каталога
zvpanina@zvpanina:~/work/study/2024-2025/Операционные системы/os-intro$ echo os-intro > COURSE
zvpanina@zvpanina:~/work/study/2024-2025/Операционные системы/os-intro$ make
Usage:
  make <target>

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

Рис. 7: Удаление файлов и создание каталогов

3. Отправка файлов на сервер. Команды git add . и git commit -m

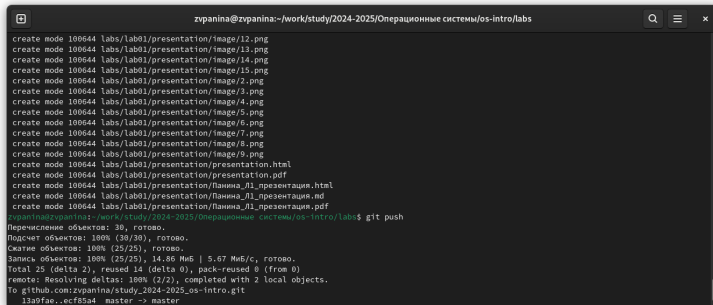


```
zvpalina@zvpalina:~/work/study/2024-2025/Операционные системы/os-intro/labs$ ls
lab01 lab03 lab05 lab07 lab09 lab11 lab13 lab15  README.ru.md
lab02 lab04 lab06 lab08 lab10 lab12 lab14  README.md
zvpalina@zvpalina:~/work/study/2024-2025/Операционные системы/os-intro/labs$ git add .
zvpalina@zvpalina:~/work/study/2024-2025/Операционные системы/os-intro/labs$ git status
Текущая ветка: master
Эта ветка соответствует «origin/master».

Изменения, которые будут включены в коммит:
(используйте «git restore --staged <файл>...», чтобы убрать из индекса)
    новый файл:   lab01/presentation/image/1.png
    новый файл:   lab01/presentation/image/10.png
    новый файл:   lab01/presentation/image/11.png
    новый файл:   lab01/presentation/image/12.png
    новый файл:   lab01/presentation/image/13.png
    новый файл:   lab01/presentation/image/14.png
    новый файл:   lab01/presentation/image/15.png
    новый файл:   lab01/presentation/image/2.png
    новый файл:   lab01/presentation/image/3.png
    новый файл:   lab01/presentation/image/4.png
    новый файл:   lab01/presentation/image/5.png
    новый файл:   lab01/presentation/image/6.png
    новый файл:   lab01/presentation/image/7.png
    новый файл:   lab01/presentation/image/8.png
    новый файл:   lab01/presentation/image/9.png
    новый файл:   lab01/presentation/presentation.html
    новый файл:   lab01/presentation/presentation.pdf
    новый файл:   lab01/presentation/Паннина_И1_презентация.html
    новый файл:   lab01/presentation/Паннина_И1_презентация.md
    новый файл:   lab01/presentation/Паннина_И1_презентация.pdf
zvpalina@zvpalina:~/work/study/2024-2025/Операционные системы/os-intro/labs$ git commit -m "add files for lab01"
```

Рис. 8: Отправка файлов на сервер

4. Команда git push

A terminal window with a dark background. The title bar shows the user 'zvpanina' and the path '~/work/study/2024-2025/Операционные системы/os-intro/labs'. The terminal displays the output of a 'git push' command, listing 25 files being pushed to the remote repository. The files include presentation images (12.png to 15.png, 2.png to 9.png), presentation.html, presentation.pdf, and three Russian presentation files. The output shows progress for object counting, compression, and writing, followed by a summary of the push operation and the remote repository URL.

```
zvpanina@zvpanina:~/work/study/2024-2025/Операционные системы/os-intro/labs$ git push
create mode 100644 labs/lab01/presentation/image/12.png
create mode 100644 labs/lab01/presentation/image/13.png
create mode 100644 labs/lab01/presentation/image/14.png
create mode 100644 labs/lab01/presentation/image/15.png
create mode 100644 labs/lab01/presentation/image/2.png
create mode 100644 labs/lab01/presentation/image/3.png
create mode 100644 labs/lab01/presentation/image/4.png
create mode 100644 labs/lab01/presentation/image/5.png
create mode 100644 labs/lab01/presentation/image/6.png
create mode 100644 labs/lab01/presentation/image/7.png
create mode 100644 labs/lab01/presentation/image/8.png
create mode 100644 labs/lab01/presentation/image/9.png
create mode 100644 labs/lab01/presentation/presentation.html
create mode 100644 labs/lab01/presentation/presentation.pdf
create mode 100644 labs/lab01/presentation/Панина_И1_презентация.html
create mode 100644 labs/lab01/presentation/Панина_И1_презентация.md
create mode 100644 labs/lab01/presentation/Панина_И1_презентация.pdf
zvpanina@zvpanina:~/work/study/2024-2025/Операционные системы/os-intro/labs$ git push
Перечисление объектов: 30, готово.
Подсчет объектов: 100% (30/30), готово.
Сжатие объектов: 100% (25/25), готово.
Запись объектов: 100% (25/25), 14.86 МБ | 5.67 МБ/с, готово.
Total 25 (delta 2), reused 14 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:zvpanina/study_2024-2025_os-intro.git
  13a9fae..ecf85a4  master -> master
```

Рис. 9: Отправка файлов на сервер

Ответы на контрольные вопросы

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

Результаты

Во время выполнения работы я изучила идеологию и применения средств контроля версий, освоила умения по работе с git.