

Лабораторная работа № 2

Операционные системы

Четвергова Мария Викторовна

Содержание

1	Цель работы	5
2	Задание	6
2.1	Базовая настройка Git	7
2.2	Создание ключа SSH	7
2.3	Создание ключа PGP	8
2.4	Настройка GitHub	9
2.5	Добавление PGP ключа в GitHub	10
2.6	Настройка автоматических подписей коммитов	12
2.7	Настройка gh	12
2.8	Шаблон для рабочего пространства	13
2.9	Ответы на контрольные вопросы	15
3	Выводы	18
	Список литературы	19

Список иллюстраций

2.1	Установка программного обеспечения	6
2.2	Базовая настройка git	7
2.3	Создание ключа SSH	8
2.4	Создание PGP ключа	9
2.5	Аккаунт на сайте GitHub	10
2.6	Вывод и копирование ключа в буфер обмена	11
2.7	Добавление ключа в GitHub	12
2.8	Настройка автоматических подписей коммитов	12
2.9	Настройка gh	13
2.10	Настройка gh	13
2.11	Создание шаблона рабочего пространства. Репозиторий на основе шаблона	14
2.12	Настройка gh	14

Список таблиц

1 Цель работы

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

2 Задание

В ходе выполнения лабораторной работы №2 необходимо выполнить следующие задания: - создать базовую конфигурацию для работы с git - создать ключ SSH - Создать ключ PGP - Настроить подписи git - Зарегистрироваться на GitHub - Создать локальный каталог для выполнения заданий по предмету

[1] # Выполнение лабораторной работы ## Установка программного обеспечения

1. Установка Git Установим git с помощью команды, вбиваемой в терминал:

git install git

2. Установка gh Установим gh с помощью вбиваемых в терминал команды:

git install gh

```
foot
[mvchetzergova@mvchetzergova ~]$ dnf install git
Ошибка: Эту команду нужно запускать с привилегиями суперпользователя (на большинстве систем - под именем пользователя root).
[mvchetzergova@mvchetzergova ~]$ sudo dnf install git
[sudo] пароль для mvchetzergova:
Попробуйте ещё раз.
[sudo] пароль для mvchetzergova:
Попробуйте ещё раз.
[sudo] пароль для mvchetzergova:
Fedora 39 - x86_64 - Updates
Fedora 39 - x86_64 - Updates
Последняя проверка окончания срока действия метаданных: 0:00:14 назад, Вс 25 фев 2024 14:13:22.
Пакет git-2.43.2-1.fc39.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
[mvchetzergova@mvchetzergova ~]$ dnf install gh
Ошибка: Эту команду нужно запускать с привилегиями суперпользователя (на большинстве систем - под именем пользователя root).
[mvchetzergova@mvchetzergova ~]$ sudo dnf install gh
Последняя проверка окончания срока действия метаданных: 0:00:42 назад, Вс 25 фев 2024 14:13:22.
Пакет gh-2.43.1-1.fc39.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
[mvchetzergova@mvchetzergova ~]$
```

Рис. 2.1: Установка программного обеспечения

2.1 Базовая настройка Git

Проведём базовую настройку git: для этого необходимо задать имя владельца, емейл владельца, а также настроить utf-8 в выводе сообщений git. При работе с этими задачами воспользуемся командами типа *git config --global ...*

git config --global user.name "Maria02-23 git config --global user.email"my_email.com

Необходимо и настроить верификацию и подписание коммитов git, а также задать имя начальной ветки(master). Настроим параметры autocrlf и safecrlf. осуществить эти действия возможно с помощью команд командной строки:

```
[nvchetvergova@nvchetvergova ~]$ git config --global user.name "Maria02-23"
[nvchetvergova@nvchetvergova ~]$ git config --global user.email '1132232886@pfur.ru'
[nvchetvergova@nvchetvergova ~]$ git config --global core.quotepath false
[nvchetvergova@nvchetvergova ~]$ git config --global init.defaultBranch master
[nvchetvergova@nvchetvergova ~]$ git config --global core.autocrlf input
[nvchetvergova@nvchetvergova ~]$ git config --global core.safecrlf warn
[nvchetvergova@nvchetvergova ~]$
```

Рис. 2.2: Базовая настройка git

2.2 Создание ключа SSH

В инструкции к лабораторной работе указаны два способа создания SSH. Воспользуемся обоими.

1. по алгоритму rsa с ключём размером 4096 бит:

ssh-keygen -t rsa -b 4096

2. По алгоритму ed25519:

ssh-keygen -t ed25519

```
[mvchetvergova@mvchetvergova ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/mvchetvergova/.ssh/id_rsa):
Created directory '/home/mvchetvergova/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/mvchetvergova/.ssh/id_rsa
Your public key has been saved in /home/mvchetvergova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:f+k2K273DYZjb7zVBapnZqMykDPRcBFcuy0ag1Hc/8 mvchetvergova@mvchetvergova
The key's randomart image is:
+----[RSA 4096]-----+
|
| ..oo
| .o.+
| ..++
| o . .o + o
| . o 5* + = o
| . . . +B =o .
| o o o . .o .
| + .E=+.+
| . .o+.+=o.
+----[SHA256]-----+
[mvchetvergova@mvchetvergova ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/mvchetvergova/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/mvchetvergova/.ssh/id_ed25519
Your public key has been saved in /home/mvchetvergova/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:YQ9P5p1I0Rh4GahoyG35FyeFX0HGx4zhMYH6TBUFFg0 mvchetvergova@mvchetvergova
The key's randomart image is:
+--[ED25519 256]--+
|
| . o+=E/+
| . .B*0+o
| . . + =.o.o+
| o = =.o
| + +S+
| = . o
| . o
+----[SHA256]-----+
[mvchetvergova@mvchetvergova ~]$
```

Рис. 2.3: Создание ключа SSH

2.3 Создание ключа PGP

1. Генерируем ключ с помощью команды в терминале:

gpg --full-generate-key

2. На мониторе появится несколько вариантов ответов. Для того, чтобы успешно создать ключ, необходимо выбрать следующие варианты:

- тип RSA
- размер 4096
- срок действия - бессрочный (0) (далее gpg запрашивает личную информацию: сперва вводим своё полное имя, а затем - емейл, привязанный к

аккаунту на GitHub. В конце можно ввести комментарий, но это уже неважно
- можно вводить что угодно)

После заполнения информации на экране всплывает окошко с требованием заполнить пароль-фразу. Пропускаем этот этап, выбрав “защита не нужна”

```
[michetvergova@michetvergova ~]$ 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?
Выберите эллиптическую кривую:
(1) Curve 25519 *default*
(4) NIST P-384
(6) Brainpool P-256
Ваш выбор? 1
Выберите срок действия ключа.
0 = не ограничен
<n> = срок действия ключа - n дней
<nw> = срок действия ключа - n недель
<nm> = срок действия ключа - n месяцев
<ny> = 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: Maria Chetvergova
Адрес электронной почты: 1132232886@pfur.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
"Maria Chetvergova <1132232886@pfur.ru>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (Q)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 отныне записан в '/home/michetvergova/.gnupg/openpgp-revocs.d/8244E9F602DC42969440D5D19A3203543C2AE026.rev'.
открытый и секретный ключи созданы и подписаны.

pub ed25519 2024-02-25 [SC]
      8244E9F602DC42969440D5D19A3203543C2AE026
uid          Maria Chetvergova <1132232886@pfur.ru>
sub cv25519 2024-02-25 [E]
```

Рис. 2.4: Создание PGP ключа

2.4 Настройка GitHub

В инструкции к лабораторной работе сказано, что необходимо создать учётную запись на сайте GitHub и заполнить основные данные. В первом семестре мы уже имели дело с системой git, поэтому аккаунт на GitHub у меня уже есть:

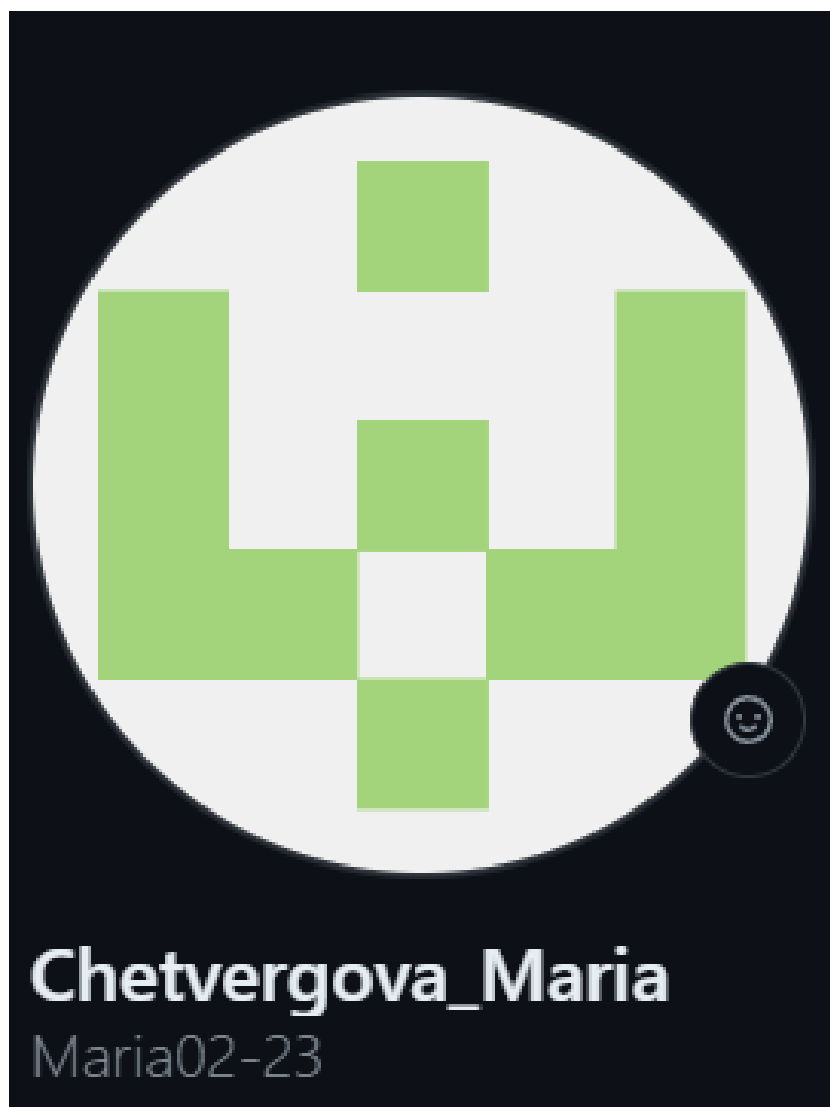


Рис. 2.5: Аккаунт на сайте GitHub

2.5 Добавление PGP ключа в GitHub

Для добавления PGP ключа в GitHub необходимо выполнить следующие действия:

- выведем список ключей и скопируем отпечаток приватного ключа с помощью команды в терминале:

```
gpg --list-secret-keys --keyid-format LONG
```

- Далее нужно скопировать сгенерированный PGP ключ в буфер обмена командой

`gpg --armor --export | xclip -sel clip`

```

root
[mvchetvergova@mvchetvergova ~]$ 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, 0f, 2u
[keybox]
-----
sec rsa4096/131C7AE61C138EEF 2024-02-23 [SC]
    C2A5BED8FD1108C41200024131C7AE61C138EEF
uid [ абсолотно ] Maria Chetvergova <1132232886@pfur.ru>
ssb rsa4096/C5430215466875F3 2024-02-23 [E]

sec ed25519/9A3203543C2AE026 2024-02-25 [SC]
    8244E9F682DC429694A0D5D19A3203543C2AE026
uid [ абсолотно ] Maria Chetvergova <1132232886@pfur.ru>
ssb cv25519/F2EB0D5503246EB8 2024-02-25 [E]

[mvchetvergova@mvchetvergova ~]$ gpg --armor --export 1132232886@pfur.ru | xclip -sel clip
bash: xclip: команда не найдена
gpg: [stdout]: write error: Обрыв канала
gpg: filter_flush failed on close: Обрыв канала
[mvchetvergova@mvchetvergova ~]$ sudo dnf install xclip
[sudo] пароль для mvchetvergova:
Последняя проверка окончания срока действия метаданных: 0:20:36 назад, Вс 25 фев 2024 14:13:22.
Зависимости разрешены.
=====
Пакет                Архитектура          Версия
-----
Установка:
xclip                x86_64                0.13-20.fc39
Результат транзакции
=====
Установка 1 Пакет
Объем загрузки: 37 к
Объем изменений: 62 к
Продолжить? [a/N]: y
Загрузка пакетов:
xclip-0.13-20.fc39.x86_64.rpm
-----
Общий размер
Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции проведен успешно.
Выполнение транзакции
Подготовка:
  Установка      : xclip-0.13-20.fc39.x86_64
  Запуск скрипта : xclip-0.13-20.fc39.x86_64
  Проверка       : xclip-0.13-20.fc39.x86_64
Установлено:
  xclip-0.13-20.fc39.x86_64
Выполнено!
  
```

Рис. 2.6: Вывод и копирование ключа в буфер обмена

В случае, если предыдущая команда не сработала, можно вывести содержимое ключа командой

`gpg --armor --export | cat`

и скопировать вручную :-)

- переходим в настройки GitHub и вставляем ключ в нужное поле

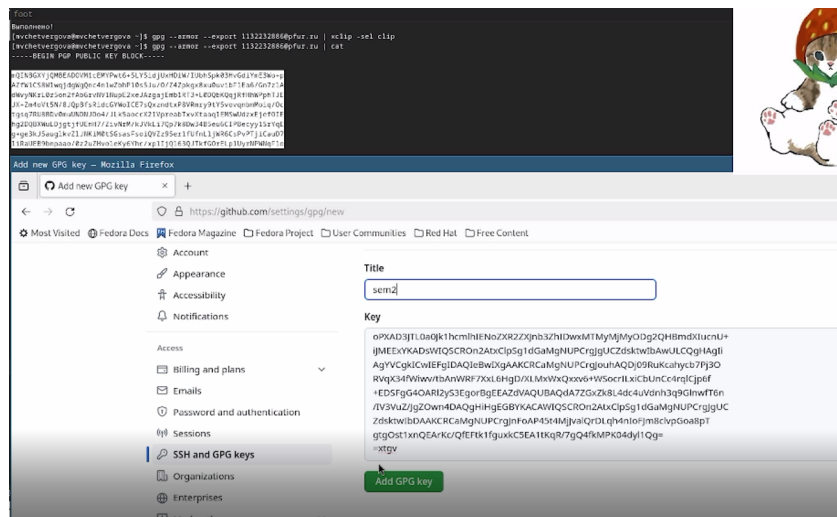


Рис. 2.7: Добавление ключа в GitHub

2.6 Настройка автоматических подписей коммитов

Используя введённый емейл, укажем Git применять его при подписи коммитов. Это можно сделать при помощи следующих команд:

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

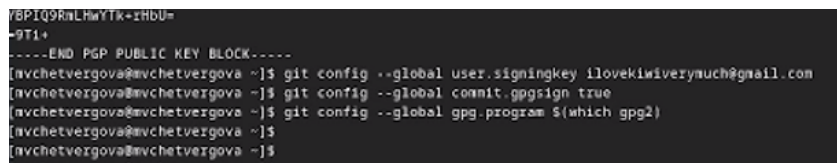


Рис. 2.8: Настройка автоматических подписей коммитов

2.7 Настройка gh

- Для начала необходимо авторизоваться, ответив на несколько вопросов после вбивания этой программы:

```
gh auth login
```

Авторизация происходит через браузер. Важно выбрать SSH в одном из вопросов :-)

```
foot
[mvchetvergova@mvchetvergova ~]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /home/mvchetvergova/.ssh/id_rsa.pub
? Title for your SSH key: mvchetvergova
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 7SA7-6F18
Press Enter to open github.com in your browser...
█
```

Рис. 2.9: Настройка gh

- Утилита задаст несколько наводящих вопросов, после чего можно авторизоваться через браузер

```
mvchetvergova@mvchetvergova ~]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /home/mvchetvergova/.ssh/id_rsa.pub
? Title for your SSH key: mvchetvergova
? How would you like to authenticate GitHub CLI? Login with a web browser

First copy your one-time code: 7SA7-6F18
Press Enter to open github.com 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/mvchetvergova/.ssh/id_rsa.pub
Logged in as Maria02-23
You were already logged in to this account
mvchetvergova@mvchetvergova ~]$
mvchetvergova@mvchetvergova ~$ █
```

Рис. 2.10: Настройка gh

2.8 Шаблон для рабочего пространства

1. Создание репозитория курса на основе шаблона Создадим шаблон рабочего пространства для 2023-2024 годов обучения. Для этого введём в терминал следующие команды:

```

mvchetvergova@mvchetvergova ~]$
mvchetvergova@mvchetvergova ~]$ mkdir -p ~/work/study/2023-2024/'Операционные системы'
mvchetvergova@mvchetvergova ~]$ mkdir cd ~/work/study/2023-2024/'Операционные системы'
mkdir: невозможно создать каталог '/home/mvchetvergova/work/study/2023-2024/Операционные системы': Файл существует
mvchetvergova@mvchetvergova ~]$ cd ~/work/study/2023-2024/'Операционные системы'
mvchetvergova@mvchetvergova Операционные системы]$ gh repo create study_2023-2024_os-intro --template=yamadharma/c
GraphQL: Could not clone: Name already exists on this account (cloneTemplateRepository)
mvchetvergova@mvchetvergova Операционные системы]$ git clone --recursive git@github.com:Maria02-23/study_2023-2024-
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), 18.60 KiB | 3.72 MiB/c, готово.
пределение изменений: 100% (1/1), готово.
подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован
подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован
клонирование в «~/home/mvchetvergova/work/study/2023-2024/Операционные системы/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 KiB | 800.00 KiB/c, готово.
пределение изменений: 100% (34/34), готово.
клонирование в «~/home/mvchetvergova/work/study/2023-2024/Операционные системы/os-intro/template/report»...
[[Dremote: 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 KiB | 1.63 MiB/c, готово.
пределение изменений: 100% (52/52), готово.
submodule path 'template/presentation': checked out '48a1761813e197d00e8443ff1ca72c60a304f24c'
[[D[[DSubmodule path 'template/report': checked out '7c31ab8e5dffa8c0b2d67caeb8a19ef8028ced88e'
[[D[mvchetvergova@mvchetvergova Операционные системы]$ █

```

Рис. 2.11: Создание шаблона рабочего пространства. Репозиторий на основе шаблона

2. Настройка каталога курса

Перейдём в каталог курса и удалим лишние файлы с помощью команды *rm package.json*. Затем создадим необходимые файлы, которые помогут с работой. В конце необходимо отправить файлы на сервер

```

[[D[mvchetvergova@mvchetvergova Операционные системы]$ cd
mvchetvergova@mvchetvergova ~]$ cd ~/work/study/2023-2024/'Операционные системы'/os-intro
mvchetvergova@mvchetvergova os-intro]$ rm package.json
mvchetvergova@mvchetvergova os-intro]$ echo os-intro > COURSE
mvchetvergova@mvchetvergova os-intro]$ make
Usage:
  make <target>

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

mvchetvergova@mvchetvergova os-intro]$ git add .
mvchetvergova@mvchetvergova os-intro]$ git commit -am 'feat(main): make course structure'
[master b3fe632] feat(main): make course structure
2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
mvchetvergova@mvchetvergova os-intro]$ git push
Перечисление объектов: 5, готово.
Подсчет изменений: 100% (5/5), готово.
При сканировании используется до 2 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 958 байтов | 958.00 KiB/c, 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 github.com:Maria02-23/study_2023-2024_os-intro.git
771cc4c..b3fe632 master -> master
mvchetvergova@mvchetvergova os-intro)$ █

```

Рис. 2.12: Настройка gh

2.9 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 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.

3 Выводы

в ходе выполнения лабораторной работы мы изучили идеологию и применение средств контроля версий и освоили умения по работе с системой git.

Список литературы

1. U. L.E. studing. 2018.