

# **Отчёт по лабораторной работе №2**

**Дисциплина: Операционные системы**

Батова Ирина Сергеевна, НММбд-01-22

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	15
4	Контрольные вопросы	16

# Список иллюстраций

2.1	Установка git . . . . .	6
2.2	Установка gh . . . . .	6
2.3	Базовая настройка git . . . . .	6
2.4	Генерация ssh ключа . . . . .	7
2.5	Создание ssh ключа на github . . . . .	7
2.6	Копирование ssh ключа . . . . .	8
2.7	Добавление ssh ключа на github . . . . .	8
2.8	Ключ создан . . . . .	8
2.9	Выбор типа gpg ключа . . . . .	9
2.10	Выбор размера и срока действия gpg ключа . . . . .	9
2.11	Ввод имени и почты . . . . .	9
2.12	Задание пароля . . . . .	10
2.13	gpg ключ сгенерирован . . . . .	10
2.14	Вывод списка ключей . . . . .	11
2.15	Копирование gpg ключа в буфер обмена . . . . .	11
2.16	Добавление pgr ключа на github . . . . .	11
2.17	Настройка автоматических подписей коммитов . . . . .	12
2.18	Настройка gh . . . . .	12
2.19	Завершение настройки gh . . . . .	12
2.20	Создание каталогов курса . . . . .	13
2.21	Создание репозитория . . . . .	13
2.22	Клонирование репозитория . . . . .	13
2.23	Переход в каталог курса . . . . .	13
2.24	Удаление лишних файлов и создание каталогов . . . . .	13
2.25	Отправление файлов на сервер 1 . . . . .	14
2.26	Ввод фразы-пароля . . . . .	14
2.27	Отправление файлов на сервер 1 . . . . .	14

## Список таблиц

# 1 Цель работы

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

## 2 Выполнение лабораторной работы

Первым делом устанавливаем git (рис. 2.1).

```
[isbatova@fedora ~]$ dnf install git
```

Рис. 2.1: Установка git

Далее устанавливаем gh (рис. 2.2).

```
[isbatova@fedora ~]$ dnf install gh
```

Рис. 2.2: Установка gh

После этого приступаем к базовой настройке git (рис. 2.3). Для этого задаем имя и email владельца репозитория (строки 1 и 2), далее настраиваем utf-8 в выводе сообщений git (строка 3), а также задаем имя начальной ветки (строка 4), параметр autorclf (строка 5) и параметр saferclf (строка 6).

```
[isbatova@fedora ~]$ git config --global user.name "<Irina Batova>"
[isbatova@fedora ~]$ git config --global user.email "<1132226490@pfur.ru>"
[isbatova@fedora ~]$ git config --global core.quotepath false
[isbatova@fedora ~]$ git config --global init.defaultBranch master
[isbatova@fedora ~]$ git config --global core.autocrlf input
[isbatova@fedora ~]$ git config --global core.saferclf warn
```

Рис. 2.3: Базовая настройка git

После этого необходимо создать ssh ключ. Для этого используется команда “ssh-keygen -C” (рис. 2.4).

```
[isbatova@fedora ~]$ ssh-keygen -C "Ирина Батова <1132226490@pfur.ru>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/isbatova/.ssh/id_rsa):
Created directory '/home/isbatova/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/isbatova/.ssh/id_rsa
Your public key has been saved in /home/isbatova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:uWw0/BQ1zIni9AUZGfLCYluF9GAfBKeJhxfkNZ9F2I8 Ирина Батова <1132226490@pfur.ru>
The key's randomart image is:
+---[RSA 3072]-----+
|      o@@B  +o      |
|      o=B#  =.o.    |
|      .-=B.X o o    |
|      o.=+.. E .    |
|      . S.          |
+---+-----+

```

Рис. 2.4: Генерация ssh ключа

После этого нам необходимо загрузить этот ключ. Для этого заходим на сайт [github.com](https://github.com), переходим в Setting – SSH and GPG keys – New SSH key (рис. 2.5).

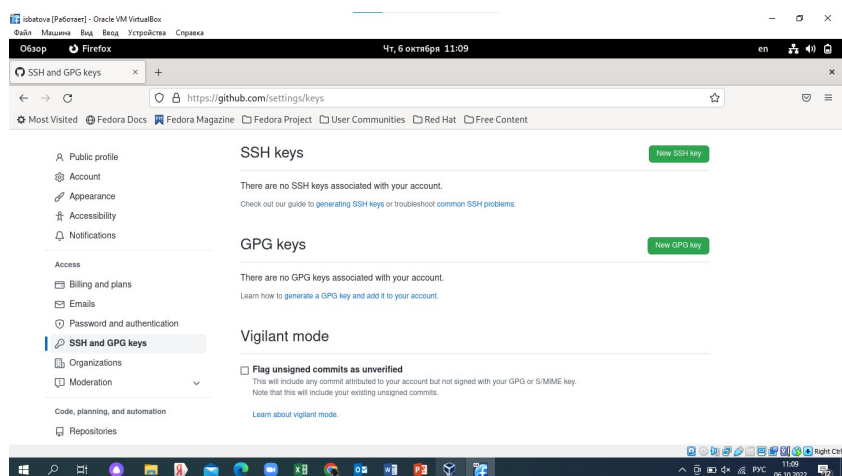


Рис. 2.5: Создание ssh ключа на github

Далее возвращаемся в терминал и вводим команду “`cat ~/.ssh/id_rsa.pub | xclip -sel clip`”, чтобы скопировать ключ. В процессе соглашаемся на установление пакета “xclip” (рис. 2.6).

```
[isbatova@fedora ~]$ cat ~/.ssh/id_rsa.pub | xclip -sel clip
bash: xclip: команда не найдена...
Установить пакет «xclip», предоставляющий команду «xclip»? [N/y] y
```

Рис. 2.6: Копирование ssh ключа

После этого возвращаемся на github.com, вводим название ключа “Laptop\_home” и в поле “Key” вставляем ключ (рис. 2.7). Ключ создан (рис. 2.8).

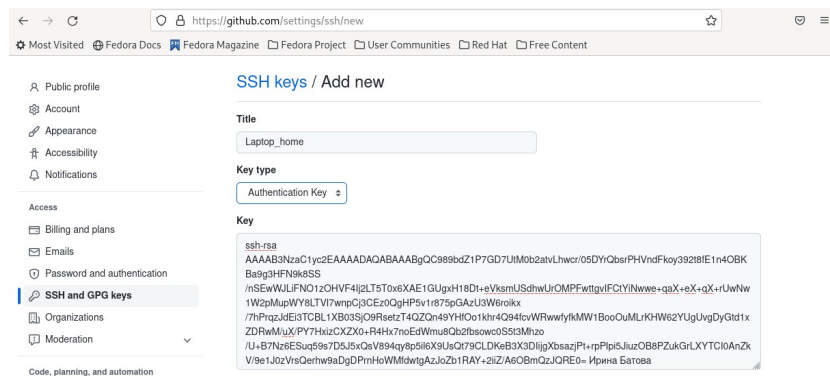


Рис. 2.7: Добавление ssh ключа на github

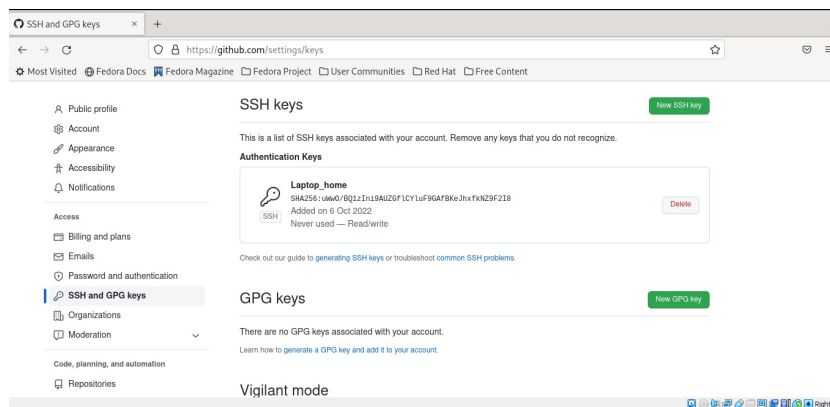


Рис. 2.8: Ключ создан

После создания ssh ключа нам также нужно создать gpg ключ. Для этого заходим в терминал и вводим команду ‘gpg –full-generate-key’. Нам будут предлагаться разные опции. Тип ключа выбираем ‘RSA and RSA’ (рис. 2.9).



```
[isbatova@fedora ~]$ gpg --full-generate-key
gpg (GnuPG) 2.3.7; 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.

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
```

Рис. 2.9: Выбор типа gpg ключа

Следующим шагом выбираем размер (4096 бит) и срок действия ключа - неограниченный (рис. 2.10).

```
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
```

Рис. 2.10: Выбор размера и срока действия gpg ключа

Далее вводим имя и email (рис. 2.11). Примечание оставляем пустым.

```
Ваше полное имя: Irina
Адрес электронной почты: 1132226490@pfur.ru
Примечание:
```

Рис. 2.11: Ввод имени и почты

Далее задаем фразу-пароль (рис. 2.12). Ключ сгенерирован (рис. 2.13).

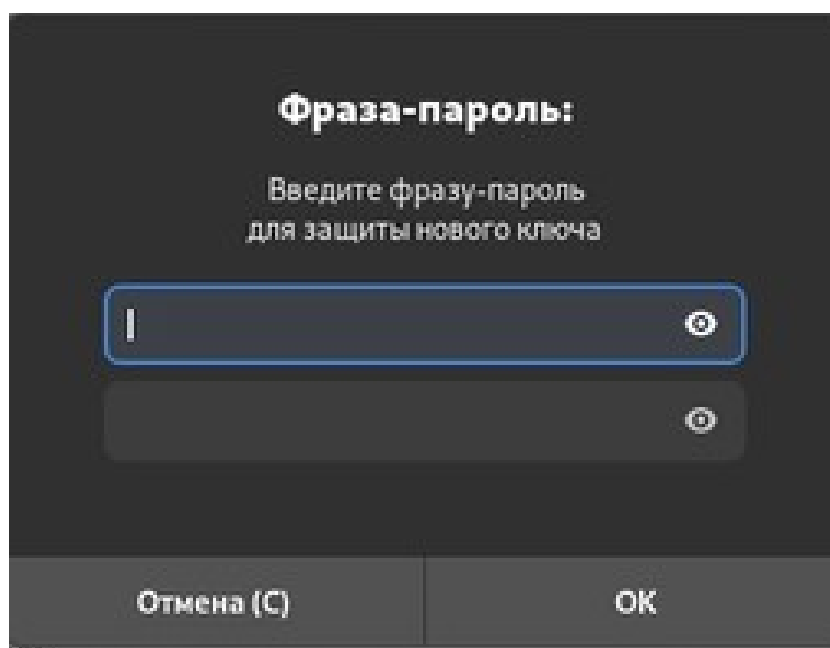


Рис. 2.12: Задание пароля

```
gpg: сертификат отзыва записан в '/home/isbatova/.gnupg/openpgp-revocs.d/223B697  
CE4EC6421DE3901EAA824FAE2D14D8630.rev'.  
открытый и секретный ключи созданы и подписаны.  
  
pub  rsa4096 2023-02-14 [SC]  
    223B697CE4EC6421DE3901EAA824FAE2D14D8630  
uid                               Irina <1132226490@pfur.ru>  
sub  rsa4096 2023-02-14 [E]
```

Рис. 2.13: gpg ключ сгенерирован

После этого выводим список ключей командой 'gpg -list-secret-keys -keyid-format LONG' (рис. 2.14) и находим отпечаток нужного нам ключа.

```
[isbatova@fedora ~]$ gpg --list-secret-keys --keyid-format LONG
/home/isbatova/.gnupg/pubring.kbx
-----
sec   rsa4096/37C3386CE078AC01 2023-02-14 [SC]
      7CC060E219FD4FCB953CF5FC37C3386CE078AC01
uid   [ абсолютно ] Irina <1112226490@pfur.ru>
ssb   rsa4096/0356DE4BF8D171EE 2023-02-14 [E]

sec   rsa4096/A824FAE2D14DB630 2023-02-14 [SC]
      223B697CE4EC6421DE3901EAA824FAE2D14DB630
uid   [ абсолютно ] Irina <1132226490@pfur.ru>
ssb   rsa4096/18BE42F03FE630AD 2023-02-14 [E]

[isbatova@fedora ~]$
```

Рис. 2.14: Вывод списка ключей

С помощью отпечатка и команды 'gpg --armor --export PGP Fingerprint | xclip -sel clip' копируем ключ в буфер обмена (рис. 2.15).

```
[isbatova@fedora ~]$ gpg --armor --export 37C3386CE078AC01 | xclip -sel clip
[isbatova@fedora ~]$
```

Рис. 2.15: Копирование gpg ключа в буфер обмена

После этого заходим на github и загружаем на ключ туда (рис. 2.16).

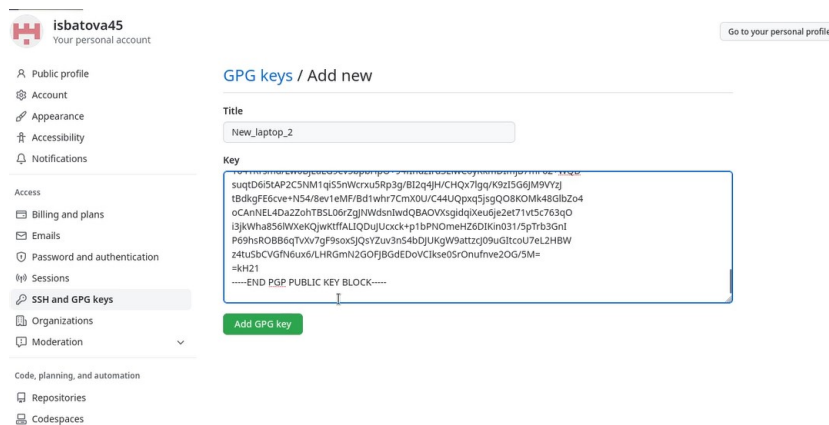


Рис. 2.16: Добавление pgr ключа на github

Следующим нашим шагом будет настройка автоматических подписей коммитов git (рис. 2.17).

```
[isbatova@fedora ~]$ git config --global user.signingkey 37C3386CE078AC
[isbatova@fedora ~]$ git config --global commit.gpgsign true
[isbatova@fedora ~]$ git config --global gpg.program $(which gpg2)
```

Рис. 2.17: Настройка автоматических подписей коммитов

Далее настраиваем gh. Для этого вводим команду 'gh auth login' и отвечаем на наводящие вопросы (рис. 2.18).

```
[isbatova@fedora ~]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? SSH
? Upload your SSH public key to your GitHub account? /home/isbatova/.ssh/id_rsa.
pub
? Title for your SSH key: New_laptop_2
? How would you like to authenticate GitHub CLI? [Use arrows to move, type to f
filter]
> Login with a web browser
  Paste an authentication token
```

Рис. 2.18: Настройка gh

Позже нас перекидывает в браузер, где мы вводим код и успешно заканчиваем настройку gh (рис. 2.19).

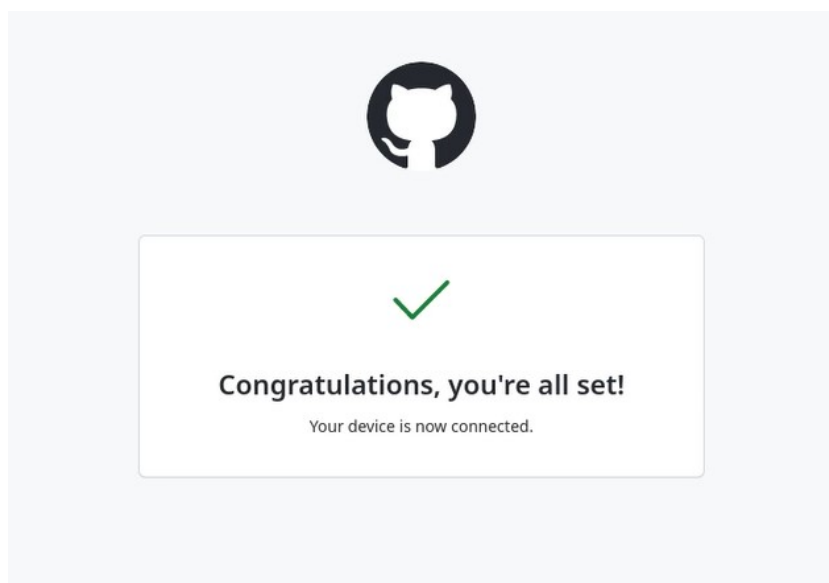


Рис. 2.19: Завершение настройки gh

Далее нам необходимо создать репозиторий курса и настроить каталог курса.

Для начала с помощью команды 'mkdir -p' создаем необходимые каталоги и с помощью команды 'cd' переходим в него (рис. 2.20).

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

Рис. 2.20: Создание каталогов курса

Далее создаем репозиторий на github (рис. 2.21) и клонируем его (рис. 2.22).

```
[isbatova@fedora Операционные системы]$ gh repo create study_2022-2023_os-intro --template=yamadharma/course-directory-student-template --public
Created repository isbatova45/study_2022-2023_os-intro on GitHub
```

Рис. 2.21: Создание репозитория

```
[isbatova@fedora Операционные системы]$ git clone --recursive git@github.com:isbatova45/study_2022-2023_os-intro.git os-intro
Клонирование в os-intro...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 27 (delta 10), reused 11 (delta 0), pack-reused 0
Получение объектов: 100% (27/27), 16.93 Кб | 184.08 Кб/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/isbatova/work/study/2022-2023/Операционные системы/os-intro/template/presentation...
```

Рис. 2.22: Клонирование репозитория

Продолжаем настройку каталога курса. Переходим в каталог курса (рис. 2.23).

```
[isbatova@fedora Операционные системы]$ cd ~/work/study/2022-2023/"Операционные системы"/os-intro
```

Рис. 2.23: Переход в каталог курса

Далее удаляем лишние файлы и создаем необходимые каталоги (рис. 2.24).

```
[isbatova@fedora os-intro]$ rm package.json
[isbatova@fedora os-intro]$ echo os-intro > COURSE
[isbatova@fedora os-intro]$ make
```

Рис. 2.24: Удаление лишних файлов и создание каталогов

Последним шагом отправляем файлы на сервер (рис. 2.25, рис. 2.27). Для отправки вводим фразу-пароль (рис. 2.26).

```
[isbatova@fedora os-intro]$ git add .
[isbatova@fedora os-intro]$ git commit -am 'feat(main): make course structure'
```

Рис. 2.25: Отправление файлов на сервер 1

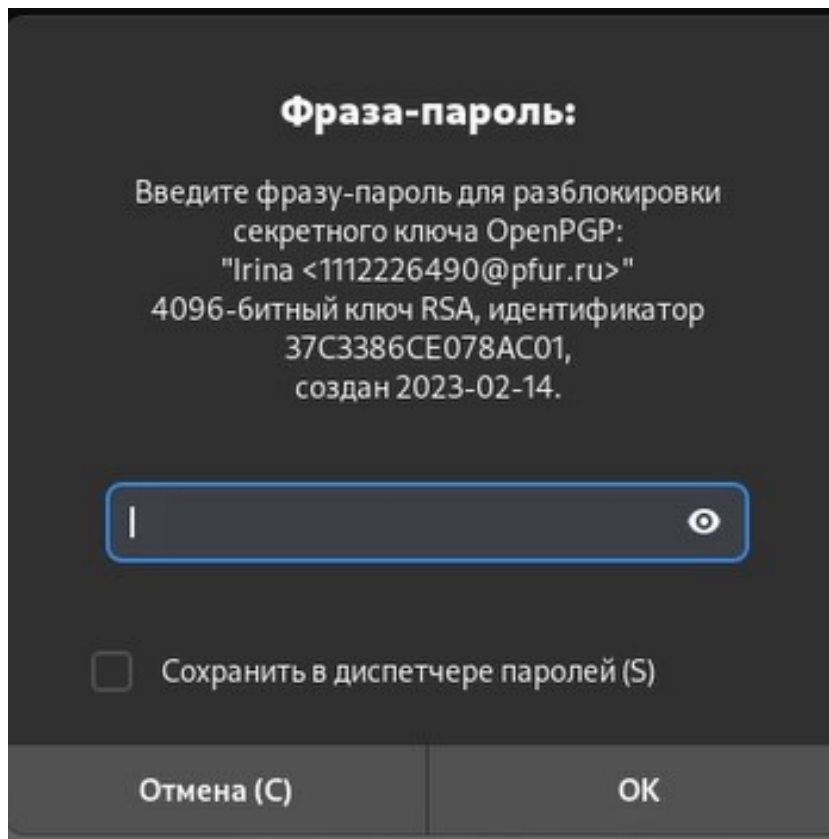


Рис. 2.26: Ввод фразы-пароля

```
[isbatova@fedora os-intro]$ git push
Перечисление объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 343.04 Киб | 2.79 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 github.com:isbatova45/study_2022-2023_os-intro.git
  767bb97...fbccde8 master -> master
```

Рис. 2.27: Отправление файлов на сервер 1

## **3 Выводы**

В данной лабораторной работе мной были изучены идеологии и применение средств контроля версий, а также освоены умения по работе с git.

## 4 Контрольные вопросы

1. Система контроля версий (VCS) - это система, регистрирующая изменения в файлах. VCS может хранить несколько версий одного документа, позволяет возвращаться к более ранним версиям, показывает, кто и какие конкретно изменения внес в документ. Система контроля версий обычно применяется в случае, если над одним проектом работает группа людей.
2. При выполнении участником проекта своей части работы он получает нужную ему версию файлов из хранилища, а затем сохраняет новую версию файлов в хранилище. То есть централизованное хранилище сохраняет все файлы - и до правки участником, и после. Следующий пользователь перед работой начнет работу с новой, измененной версией файла. Команда `commit` как раз осуществляет сохранение изменений (но при этом на сервер они уходят с помощью другой команды). История проекта - это история изменения файлов, то есть кто изменил, какие файлы, какие изменения были внесены. Рабочую версию участник проекта всегда извлекает перед началом работы - то есть, рабочей версией называется та, в которой сохранены все необходимые для работы конкретного пользователя изменения.
3. Централизованные VCS созданы для бэкапирования, отслеживания и синхронизации файлов. Все изменения происходят через центральный сервер. А в децентрализованных VCS у каждого пользователя есть свой полноценный репозиторий и нет жестко заданной структуры репозитория с центральным сервером. Децентрализованные VCS были созданы для обмена



изменениями, например, Git. Пример централизованной VCS - Subversion (SVN).

4. При единоличной работе с хранилищем для начала создаем локальный репозиторий и делаем предварительную конфигурацию: задаем имя и email владельца репозитория, настраиваем utf-8 в выводе сообщений, задаем имя начальной ветки, параметр autorclf и параметр saferclf. Далее при внесении изменений в файлы необходимо вводить последовательность команд 'git add .', 'git commit -am', 'git push' для сохранения изменений в репозитории.
5. При работе с общим хранилищем VCS нам необходим ssh ключ. Для этого используется команда "ssh-keygen -C", после этого копируем ключ и вставляем в соответствующее окно в веб-браузере
6. У git можно выделить следующие основные задачи:
  - хранение всей информации о любых изменениях в файлах
  - фиксирование и совмещение изменений
  - сохранение истории и возможность вернуться к ранней версии файла
7. Основные команды git:
  - git init - создание основного дерева репозитория
  - git pull - получение обновлений (изменений) текущего дерева из центрального репозитория
  - git push - отправка всех произведённых изменений локального дерева в центральный репозиторий
  - git status - просмотр списка изменённых файлов в текущей директории
  - git diff - просмотр текущих изменений
  - git add . - добавление всех изменённых и/или созданных файлов и/или каталогов

- `git add имена_файлов` - добавление конкретных изменённых и/или созданных файлов и/или каталогов
- `git rm имена_файлов` - удаление файла и/или каталога из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории)
- `git commit -am 'Описание коммита'` - сохранение всех добавленных изменений и всех изменённых файлов
- `git commit` - сохранение добавленных изменений с внесением комментария через встроенный редактор
- `git checkout -b имя_ветки` - создание новой ветки, базирующейся на текущей
- `git checkout имя_ветки` - переключение на некоторую ветку
- `git push origin имя_ветки` - отправка изменений конкретной ветки в центральный репозиторий
- `git merge --no-ff имя_ветки` - слияние ветки с текущим деревом:
- `git branch -d имя_ветки` - удаление локальной уже слитой с основным деревом ветки
- `git branch -D имя_ветки` - принудительное удаление локальной ветки
- `git push origin :имя_ветки` - удаление ветки с центрального репозитория

8. Например, добавление файла 'text.txt'

- `git add text.txt`
- `git commit -am 'add file text.txt'`
- `git push`

9. Ветка в git - это указатель на один из коммитов. Ветка берет свое начало от какого-то из коммитов. Ветки существуют для того, чтобы пользователи

могли работать над проектом, не мешая друг другу. Обычно при работе участник проекта заводит новую ветку от последнего рабочего коммита базовой ветки, а после решения задачи объединяет созданную ветку с базовой.

10. При работе над проектом могут создаваться файлы, которые не нуждаются в выгрузке в репозиторий. Игнорируемые файлы отслеживаются в специальном файле `.gitignore` - для игнорирования каких-то файлов достаточно их здесь указать.