

# **Лабораторная работа №2**

**Отчет**

Устинова Виктория Вадимовна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>4</b>	<b>Выводы</b>	<b>15</b>
<b>5</b>	<b>Ответы на контрольные вопросы</b>	<b>16</b>

# Список иллюстраций

3.1	Процесс установки . . . . .	7
3.2	задаем имя и настраиваем utl-8 . . . . .	8
3.3	задаем имя ветке . . . . .	8
3.4	пишем параметры . . . . .	8
3.5	вводим команду . . . . .	8
3.6	Вводим команду . . . . .	9
3.7	Генерируем ключ командой и выбираем опции . . . . .	10
3.8	Копируем отпечаток это почта . . . . .	11
3.9	Копируем ключ и устанавливаем команду xclip . . . . .	11
3.10	Вставляем ключ в гитхаб . . . . .	12
3.11	Настраиваем подписи . . . . .	12
3.12	Авторизовываемся, чтобы настроить gh . . . . .	12
3.13	Получилось успешно авторизоваться . . . . .	13
3.14	Создаем шаблон используя mkdir и переходим в него cd . . . . .	13
3.15	Вводим команды из туиса . . . . .	13
3.16	Вводим команды . . . . .	13
3.17	Получилось успешно отправить . . . . .	14

## **Список таблиц**

# 1 Цель работы

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

## 2 Задание

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.

### 3 Выполнение лабораторной работы

Установка программного обеспечения git и gh(рис. 3.1).

```
[vvustinova@vvustinova ~]$ sudo dnf install git
[sudo] пароль для vvustinova:
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет "git-2.48.1-1.fc41.x86_64" уже установлен.

Нечего делать.
[vvustinova@vvustinova ~]$ sudo dnf install gh
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет Арх. Версия Репозиторий Размер
Установка:
gh x86_64 2.65.0-1.fc41 updates 42.6 MiB

Сводка транзакции:
Установка: 1 пакета

Общий размер входящих пакетов составляет 10 MiB. Необходимо загрузить 10 MiB.
После этой операции будут использоваться дополнительные 43 MiB (установка 43 MiB, удаление 0 B).
Is this ok [y/N]: y
[1/1] gh-0:2.65.0-1.fc41.x86_64 100% | 14.2 MiB/s | 10.3 MiB | 00m01s
-----
[1/1] Total 100% | 9.2 MiB/s | 10.3 MiB | 00m01s
Выполнение транзакции
[1/3] Проверить файлы пакетов 100% | 8.0 B/s | 1.0 B | 00m00s
[2/3] Подготовить транзакцию 100% | 1.0 B/s | 1.0 B | 00m01s
[3/3] Установка gh-0:2.65.0-1.fc41 100% | 2.4 MiB/s | 42.7 MiB | 00m18s
Завершено!
[vvustinova@vvustinova ~]$
```

Рис. 3.1: Процесс установки

Зададим имя и email владельца репозитория и настроим utf-8 в выводе сообщений git(рис. 3.2).

```
[vvustinova@vvustinova ~]$ git config --global user.name "Viktoria Ustinova"
[vvustinova@vvustinova ~]$ git config --global core.quotepath false
[vvustinova@vvustinova ~]$
```

Рис. 3.2: задаем имя и настраиваем utl-8

Зададим имя начальной ветки (будем называть её master)(рис. 3.3).

```
[vvustinova@vvustinova ~]$ git config --global core.quotepath false
[vvustinova@vvustinova ~]$ git config --global init.defaultBranch master
[vvustinova@vvustinova ~]$
```

Рис. 3.3: задаем имя ветке

Параметр autocrlf и параметр safecrlf(рис. 3.4).

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

Рис. 3.4: пишем параметры

Создаем ключ ssh по алгоритму rsa с ключём размером 4096 бит(рис. 3.5).

```
[vvustinova@vvustinova ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/vvustinova/.ssh/id_rsa):
Created directory '/home/vvustinova/.ssh'.
Enter passphrase for "/home/vvustinova/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/vvustinova/.ssh/id_rsa
Your public key has been saved in /home/vvustinova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:8LrVD4zFTJESzK311kAzGUN0bu9/C1JdGj90R6qbCtQ vvustinova@vvustinova
The key's randomart image is:
+---[RSA 4096]-----+
|  o.. .0+ |
|  o.o. +o+ |
|  ooo. +o .|
| .+ o.....o |
|  S E.o +. |
|  * o * +. |
| . B * = . |
|  - + + = ..|
|  o . .+ .o |
+-----[SHA256]-----+
```

Рис. 3.5: вводим команду



Создаем ключ ssh по алгоритму ed25519(рис. 3.6).

```
[vvustinova@vvustinova ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/vvustinova/.ssh/id_ed25519):
Enter passphrase for "/home/vvustinova/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/vvustinova/.ssh/id_ed25519
Your public key has been saved in /home/vvustinova/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:tycD4wY0WbhcMuAKUyU2w4QGMeT86KgtY+cJw3md5lg vvustinova@vvustinova
The key's randomart image is:
+--[ED25519 256]--+
|*=@o.          |
|o^,*o.         |
|+...+.o        |
|.oo. B         |
|.. - . S .     |
|o...o.o + .    |
|+^..o +. o + . |
|.E* - . +      |
|. + .          |
+----[SHA256]-----+
[vvustinova@vvustinova ~]$
```

Рис. 3.6: Вводим команду

Генерируем ключ pgpr(рис. 3.7).

```

[vvustinova@vvustinova ~]$ gpg --full-generate-key
gpg (GnuPG) 2.4.5; 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.

gpg: создан каталог '/home/vvustinova/.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: Viktoria Ustinova
Адрес электронной почты: vktstnv65@bk.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Viktoria Ustinova <vktstnv65@bk.ru>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? o
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать

```

Рис. 3.7: Генерируем ключ командой и выбираем опции

Выводим список ключей и копируем отпечаток приватного ключа(рис. 3.8).

```

vvustinova@vvustinova ~]$ gpg --list-secret-keys --keyid-format LONG
pg: проверка таблицы доверия
pg: marginals needed: 3 completes needed: 1 trust model: pgp
pg: глубина: 0 достоверных: 1 подписанных: 0 доверие: 0-, 0q, 0n, 0m, 0f, 1u
[keybox]
-----
ec rsa4096/EC6398422D8278D3 2025-03-01 [5C]
4655F759A39AD9A19E9900A9EC6398422D8278D3
id [ абсолютно ] Viktor Ustinova <vktrstnv65@bk.ru>
sb rsa4096/1FAB78A3FACC039E 2025-03-01 [E]
vvustinova@vvustinova ~]$

```

Рис. 3.8: Копируем отпечаток это почта

Скопируйте ваш сгенерированный PGP ключ в буфер обмена(рис. 3.9).

```

[vvustinova@vvustinova ~]$ gpg --armor --export vktrstnv65@bk.ru | xclip -sel clip
bash: xclip: команда не найдена
gpg: (stdout): write error: Обрыв канала
gpg: filter_flush failed on close: Обрыв канала
[vvustinova@vvustinova ~]$ sudo dnf install xclip
[sudo] пароль для vvustinova:
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет Арх. Версия Репозиторий Размер
Установка:
xclip x86_64 0.13-22.git11cbe61.fc41 fedora 62.4 KiB

Сводка транзакции:
Установка: 1 пакета

Общий размер входящих пакетов составляет 37 KiB. Необходимо загрузить 37 KiB.
После этой операции будут использоваться дополнительные 62 KiB (установка 62 KiB, удаление 0 B).
Is this ok [y/N]: y
[1/1] xclip-0:0.13-22.git11cbe61.fc41.x86_64 100% | 301.9 KiB/s | 36.5 KiB | 00m00s
-----
[1/1] Total 100% | 17.5 KiB/s | 36.5 KiB | 00m02s
Выполнение транзакции
[1/3] Проверить файлы пак 100% | 37.0 B/s | 1.0 B | 00m00s
[2/3] Подготовить транзак 100% | 2.0 B/s | 1.0 B | 00m00s
[3/3] Установка xclip-0:0.13-22.g1 100% | 24.2 KiB/s | 64.3 KiB | 00m03s
Завершено!
[vvustinova@vvustinova ~]$ gpg --armor --export vktrstnv65@bk.ru | xclip -sel clip
[vvustinova@vvustinova ~]$

```

Рис. 3.9: Копируем ключ и устанавливаем команду xclip

Переходим в гитхаб и находим gpg ключ(рис. 3.10).

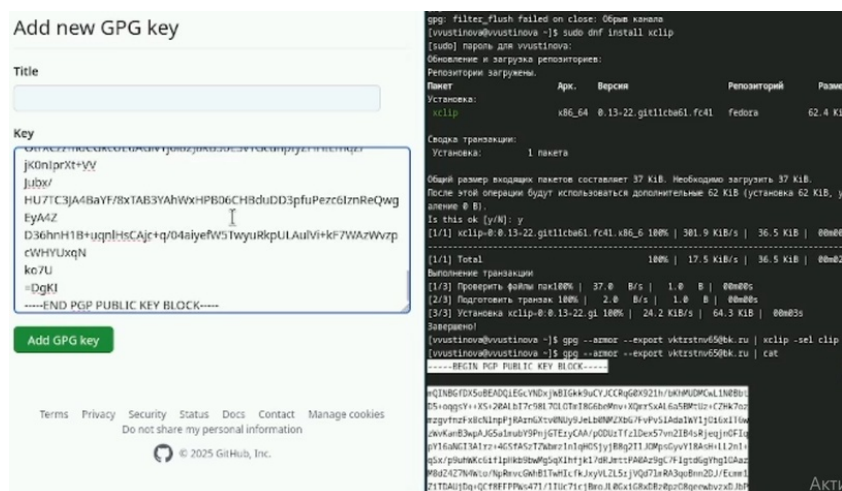


Рис. 3.10: Вставляем ключ в гитхаб

Используя введённый email, укажите Git применять его при подписи коммитов(рис. 3.11).

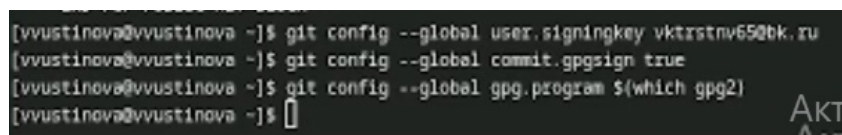


Рис. 3.11: Настраиваем подписи

Для начала необходимо авторизоваться(рис. 3.12).

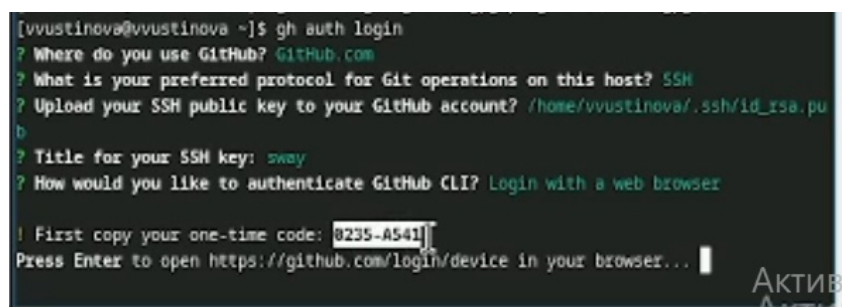


Рис. 3.12: Авторизовываемся, чтобы настроить gh

Копируем код,выделенный на предыдущем фото и вставляем в окно(рис. 3.13).

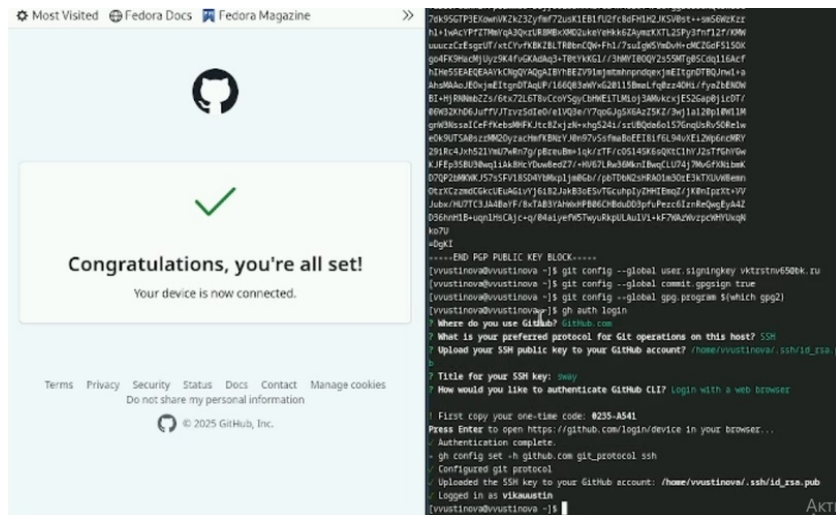


Рис. 3.13: Получилось успешно авторизоваться

Необходимо создать шаблон рабочего пространства(рис. 3.14).

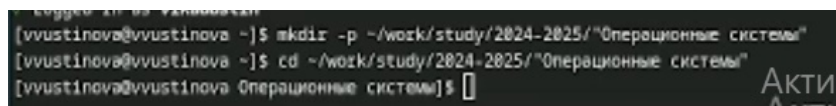


Рис. 3.14: Создаем шаблон используя mkdir и переходим в него cd

Создаем репозиторий и клонируем все на гитхаб(рис. 3.15).

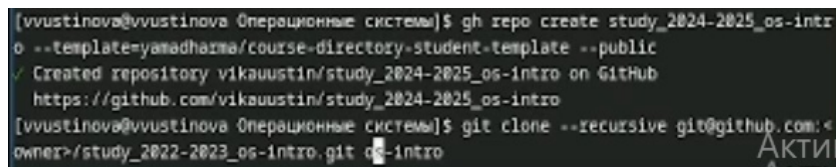


Рис. 3.15: Вводим команды из туиса

Переходим в каталог курса и создаем необходимые каталоги(рис. 3.16).

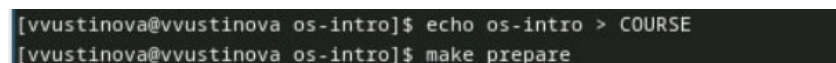


Рис. 3.16: Вводим команды

Отправляем все файлы на гитхаб(рис. 3.17).

```

[vvustinova@vvustinova os-intro]$ git add .
[vvustinova@vvustinova os-intro]$ git commit -am 'feat(main): make course structure'
Текущая ветка: master
Ваша ветка опережает «origin/master» на 2 коммита.
(используйте «git push», чтобы опубликовать ваши локальные коммиты)

ничего коммитить, нет изменений в рабочем каталоге
[vvustinova@vvustinova os-intro]$ git push origin master
The authenticity of host 'github.com (140.82.121.4)' can't
be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zL
DA0zPMSvHdkr4UvCOqU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Перечисление объектов: 42, готово.
Подсчет объектов: 100% (42/42), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (32/32), готово.
Запись объектов: 100% (40/40), 343.20 Киб | 3.36 Миб/с, готово.
Total 40 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (5/5), completed with 1 local object.
To github.com:vikauustin/study_2024-2025_os-intro.git
dfe381e..82b7917 master -> master
[vvustinova@vvustinova os-intro]$

```

Рис. 3.17: Получилось успешно отправить

## 4 Выводы

У нас получилось изучить идеологию и применение средств контроля версий и освоить умения по работе с git.



## 5 Ответы на контрольные вопросы

1. Система контроля версий (Version Control System, VCS) – это инструмент, который отслеживает изменения в наборе файлов с течением времени. Представьте себе возможность вернуться к любой предыдущей версии документа, программы или веб-сайта, посмотреть, кто, когда и почему внес изменения.
2. VCS позволяет нескольким разработчикам работать над одним проектом одновременно. Если что-то пошло не так, VCS позволяет быстро вернуться к более ранней версии файла или всего проекта. VCS позволяет создавать отдельные ветки разработки, чтобы экспериментировать с новыми функциями или исправлять ошибки, не затрагивая основную линию разработки. Хранилище (Repository): Это центральное место, где хранится вся история проекта. Commit: Это фиксация набора изменений в хранилище. История (History): Это последовательность всех commit-ов, выполненных в проекте, упорядоченных во времени. История позволяет видеть эволюцию проекта и отслеживать все изменения. Рабочая копия (Working Copy): Это локальная копия файлов проекта на компьютере разработчика.
3. Централизованные VCS: В централизованных VCS есть одно центральное хранилище, где хранится вся история проекта. Разработчики получают рабочую копию из этого хранилища, вносят изменения и отправляют их обратно в центральное хранилище. Примеры: Subversion (SVN), CVS. Децентрализованные VCS: В децентрализованных VCS каждый разработчик имеет полную копию хранилища со всей историей проекта. Разработчики могут



работать независимо друг от друга и обмениваться изменениями напрямую, минуя центральный сервер. Примеры: Git, Mercurial.

4.
  1. Инициализация хранилища: Создается новое хранилище для проекта (например, с помощью `git init`).
  2. Добавление файлов: Файлы проекта добавляются в область отслеживания VCS.
  3. Фиксация изменений (`commit`): Внесенные изменения фиксируются в хранилище с комментарием, описывающим изменения.
  4. Повторение шагов 3 и 4: Процесс внесения изменений и их фиксации повторяется по мере необходимости.
  5. Просмотр истории: Просматривается история изменений, чтобы понять, что и когда было изменено.
  6. Возврат к предыдущим версиям: При необходимости выполняется возврат к одной из предыдущих версий файла или проекта.
5.
  1. Клонирование хранилища: Разработчик клонирует удаленное хранилище на свой компьютер, создавая локальную рабочую копию.
  2. Создание ветки (`branch`): (Опционально) Создается отдельная ветка для внесения изменений, чтобы не затрагивать основную линию разработки.
  3. Внесение изменений: Разработчик вносит изменения в локальную рабочую копию.
  4. Фиксация изменений (`commit`): Изменения фиксируются в локальном хранилище.
  5. Отправка изменений (`push`): Локальные изменения отправляются в удаленное хранилище (обычно в свою ветку).
  6. Создание запроса на слияние (`pull request`): Разработчик создает запрос на слияние своих изменений из ветки в основную ветку.
  7. Обзор изменений (`code review`): Другие разработчики просматривают изменения и оставляют комментарии.
  8. Слияние изменений (`merge`): После одобрения изменений они сливаются в основную ветку.
  9. Разрешение конфликтов: Если при слиянии возникают конфликты (например, два разработчика изменили один и тот же участок кода), они разрешаются вручную.
  10. Синхронизация (`pull`): Разработчик получает последние изменения из удаленного хранилища в свою локаль-

ную рабочую копию.

6. Основные задачи, решаемые Git: • Контроль версий: Git отслеживает изменения в файлах и позволяет возвращаться к предыдущим версиям. • Совместная работа. • Ветвление и слияние: Git позволяет создавать отдельные ветки разработки для экспериментов и исправления ошибок, а затем сливать эти ветки обратно в основную линию разработки. • Распределенная разработка: Git - это децентрализованная VCS, поэтому каждый разработчик имеет полную копию хранилища.
7. Основные команды Git:  
git init: Инициализирует новый Git-репозиторий.  
git clone : Клонировать существующий Git-репозиторий с удаленного сервера.  
git add : Добавляет файлы в область подготовки (staging area).  
git commit -m "Сообщение": Фиксирует изменения из области подготовки в локальном репозитории.  
git status: Показывает текущий статус файлов в репозитории.  
git branch: Управляет ветками (создание, удаление, переключение).  
git push : Отправляет локальные коммиты в удаленный репозиторий.  
git pull : Получает изменения из удаленного репозитория и сливает их в текущую ветку.  
git revert: Создает новый коммит, отменяющий изменения из предыдущего коммита.  
git stash: Временно сохраняет изменения в рабочей директории, чтобы переключиться на другую ветку.
8. Локальный: mkdir my\_project cd my\_project git init Создаем файл hello.txt  
echo "Hello, world!" > hello.txt Добавляем файл в область подготовки git add hello.txt  
Фиксируем изменения git commit -m "Initial commit: Added hello.txt"  
Удаленный: git remote add origin Отправляем локальную ветку master в удаленную ветку master  
git push origin master Клонировать репозиторий с GitHub  
git clone
9. Ветвь – это указатель на определенный commit в истории проекта. Ветви позволяют работать над разными функциями или исправлениями ошибок параллельно, не затрагивая основную линию разработки. Зачем нужны ветви: 1.Изоляция изменений. 2.Параллельная разработка. каждая в своей

ветке. 3.Эксперименты. 4.Подготовка к релизу.

10. Зачем это нужно
- 1.Очистка репозитория: Предотвращает засорение репозитория ненужными файлами.
- 2.Безопасность: Предотвращает попадание конфиденциальных данных (например, паролей) в репозиторий.
- 3.Производительность: Уменьшает размер репозитория и ускоряет операции Git.