

Лабораторная работа 2

Первоначальна настройка git

Иванова Ангелина Олеговна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Установка программного обеспечения	7
3.2	Базовая настройка git	8
3.3	Создание ключей	12
3.4	Настройка автоматических подписей коммитов gi	13
3.5	Настройка gh	14
3.6	Создание репозитория курса на основе шаблона	14
3.7	Настройка каталога курса	15
4	Контрольные вопросы	17
5	Выводы	22
6	Список литературы	23

Список иллюстраций

3.1	Установка git	7
3.2	Установка gh	7
3.3	Установка имени и почты	8
3.4	Настройка utf-8	8
3.5	Генерация ключа	9
3.6	Вывод списка ключей	10
3.7	Вывод списка ключей	10
3.8	Ввод ключа	11
3.9	Готовый ключ	11
3.10	Задание названия ветки и параметры autocrlf и safecrlf	12
3.11	Создание по алгоритму rsa с ключём размером 4096 бит	12
3.12	Создание по алгоритму ed25519	13
3.13	Создание по алгоритму ed25519	13
3.14	Авторизация	14
3.15	Создание рабочего пространства	15
3.16	Создание рабочего пространства	15
3.17	Создание рабочего пространства	16

Список таблиц

1 Цель работы

Изучить идеологию и применение средств контроля версий, а также освоить базовые умения по работе с git.

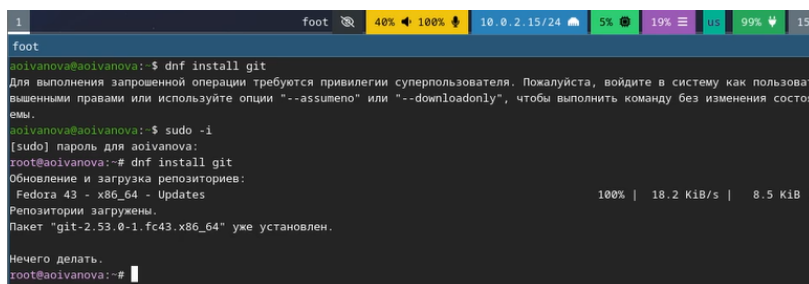
2 Задание

- 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

3.1 Установка программного обеспечения

Установили git (рис. 3.1).

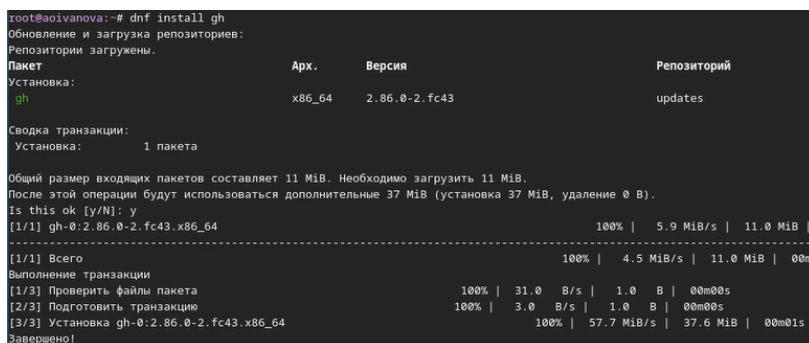


```
1 foot 40% 100% 10.0.2.15/24 5% 19% 0% 99% 15:
foot
aoivanova@aoivanova:~$ dnf install git
Для выполнения запрошенной операции требуются привилегии суперпользователя. Пожалуйста, войдите в систему как пользователь с повышенными правами или используйте опции "--assume-no" или "--downloadonly", чтобы выполнить команду без изменения состояния.
aoivanova@aoivanova:~$ sudo -i
[sudo] пароль для aoivanova:
root@aoivanova:~# dnf install git
Обновление и загрузка репозитория:
Fedora 43 - x86_64 - Updates 100% | 18.2 KiB/s | 8.5 KiB |
Репозитории загружены.
Пакет "git-2.53.0-1.fc43.x86_64" уже установлен.

Нечего делать.
root@aoivanova:~#
```

Рисунок 3.1: Установка git

Установили gh (рис. 3.2).



```
root@aoivanova:~# dnf install gh
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет
Установка:
  gh
  Арх.      Версия      Репозиторий
  x86_64    2.86.0-2.fc43    updates
Сводка транзакции:
  Установка:      1 пакета
Общий размер входящих пакетов составляет 11 MiB. Необходимо загрузить 11 MiB.
После этой операции будут использоваться дополнительные 37 MiB (установка 37 MiB, удаление 0 B).
Is this ok [y/N]: y
[1/1] gh-0.2.86.0-2.fc43.x86_64 100% | 5.9 MiB/s | 11.0 MiB |
-----
[1/1] Всего 100% | 4.5 MiB/s | 11.0 MiB | 00m0s
Выполнение транзакции
[1/3] Проверить файлы пакета 100% | 31.0 B/s | 1.0 B | 00m00s
[2/3] Подготовить транзакцию 100% | 3.0 B/s | 1.0 B | 00m00s
[3/3] Установка gh-0.2.86.0-2.fc43.x86_64 100% | 57.7 MiB/s | 37.6 MiB | 00m01s
Завершено!
```

Рисунок 3.2: Установка gh

3.2 Базовая настройка git

Задали имя и email владельца репозитория (рис. 3.3).

```
root@aoivanova:~# git config --global user.name "Ivanova Angelina"
root@aoivanova:~# git config --global user.email "1032252598@rudn.ru"
```

Рисунок 3.3: Установка имени и почты

Настроили utf-8 в выводе сообщений git (рис. 3.4).

```
root@aoivanova:~# git config --global core.quotepath false
```

Рисунок 3.4: Настройка utf-8

Настроили верификацию и подписание коммитов git. Для этого сгенерировали ключ(рис. 3.5).


```

root@aoivanova:~# gpg --full-generate-key
gpg (GnuPG) 2.4.9; Copyright (C) 2025 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 '/root/.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: Angelina
Адрес электронной почты: 1032252598@rudn.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Angelina <1032252598@rudn.ru>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? o

```

Рисунок 3.5: Генерация ключа

Из предложенных опций выбрали:

- тип RSA and RSA;
- размер 4096;
- срок действия не истекает никогда;
- Имя (Angelina);
- Адрес электронной почты (1032252598@rudn.ru).

Вывели список ключей и скопировали отпечаток приватного ключа(в моем случае BASE624972FBE7B)(рис. 3.6).

```

root@aoivanova:~# 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
[keyboard]
-----
sec   rsa4096/8ACE6249F72FBE7B 2026-02-22 [SC]
      89E69332A1EB56EC85CCB234BACE6249F72FBE7B
uid       [ абсолютно ] Angelina <1032252598@pfur.ru>
ssb   rsa4096/E9DAFA5C81F1D485 2026-02-22 [E]

```

Рисунок 3.6: Вывод списка ключей

Вывели ключ в формате ASCII по его отпечатку и скопировали его(рис. 3.7).

```

root@aoivanova:~# gpg --armor --export BACE6249F72FBE7B
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGma+sYBEADNW2bS78LRfxt+LzaudNqc4XtdOi404F+zcAviJCJnkFUPEvG
g
IRiKY1IuhXEznTIg/fWM3j8GRrPgGsC65LY6ViirmpBkU5Wreo0puiTJKCw3iFT
+
cF1B5B5pNo9TApXKULbfEoHfueqbwQ+K56A25c+0Gr1X/6ZmSMjK71kXZsRdZuK
m
AGlRrC0HkQXvDEy5b2hyLxgIz4Q+wHILv9nRIrtQzDQ/E7o8SXpleaX0FDbFYM9
X
rfzdU5ImUIFpodi9Ltqjuc4zWh2YhvXurV/k6QMf0s/r8z1K2YcvSVZMXt3xxip
Q
6dHiTeMwhvbY0G5T4Te+0mzcrhjpvG7H+iaZWyoFPadps41/MFAXVvWdVebLyDF
z
N7Dk5Wfu8F1hGWORvr7IDwiCzaUaa0xKzDIKL/TZ15za/+pWftvECBL5apz9WfB
k
0EL26p0LT01IhPGFwiUxno4EKeoYaNc9LP9/mBiky8I07mDYXmFcPTIhVrb1NTA
P
dgLI+DQG+1+K0BBQuI9fp6GQhDEYEMFDGKH7F1DveuapL4y6qX4xW1KbdHCBkaX
J
N1hRHYLRI159MwAjbvIUd6kgxp4uqcrgwFRH00xaKze7JauU4fsZDrJu8kUlw2Hp
1
7+PKCqo+3c0HpIx6InotbkY+He+cmXdv19o+RN5r9B2K6ZcbAsYz18ks0wARAQA
B
tB1BbmdlBGlUYSA8MTAzMjI1MjU50EBwZnVyLnJ1PokCUQQTaQoA0xYhBInmkzK

```

Рисунок 3.7: Вывод списка ключей

Переикb в настройки GitHub (<https://github.com/settings/keys>), нажfkб на кнопку New GPG key и вставили полученный ключ в поле ввода, после чего получили готовый загруженный ключ(рис. 3.8),(рис. 3.9).

Add new GPG key

Title

Key

```
|-----BEGIN PGP PUBLIC KEY BLOCK-----  
  
mQINBGma+sYBEADNW2bS78LRfxt+LzaudNqc4XtdOi4O4F+zcAvijCJ  
nkFUEvGg  
IRiKY1IuhXEznTIg/  
fWM3j8GRrPgGsC65LY6ViirmpBkU5Wreo0puiTJKCw3iFT+  
cFIB5B5pNo9TApXKULbfEoHfueqbwQ+K56A25c+0Gr1X/6ZmSMjk7lk  
XZsRdZuKm
```


Add GPG key

Рисунок 3.8: Ввод ключа

GPG keys

New GPG key

This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.


GPG

fedora OS
Email address: 1032252598@pfur.ru Unverified
Key ID: BACE6249F72FBE7B
Subkeys: E9DAFA5CB1F1D485
Added on Feb 22, 2026
[Delete](#)

Рисунок 3.9: Готовый ключ

Возвращаемся к базовой настройке git. Задали имя начальной ветки (будем

называть её master) и параметры autocrlf и safecrlf(рис. 3.10).

```
root@aoivanova:~# git config --global init.defaultBranch master
root@aoivanova:~# git config --global core.autocrlf input
root@aoivanova:~# git config --global core.safecrlf warn
```

Рисунок 3.10: Задание названия ветки и параметры autocrlf и safecrlf

3.3 Создание ключей

По алгоритму rsa с ключём размером 4096 бит создали ключ SSH (рис. 3.11).

```
root@aoivanova:~# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase for "/root/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:dMlofolLnCb7lNUwA7hpp002yzB0dGVbHCgFcT0gWr4 root@aoivano
va
The key's randomart image is:
+---[RSA 4096]-----+
|      .=*Boo.      |
|      o.o*.*.      |
|      .o+++.O      |
|      .+B=.+ *      |
|      ..=o+S +.      |
|      o +* =        |
|      +Eo+          |
|      oo            |
|      .              |
+----[SHA256]-----+
```

Рисунок 3.11: Создание по алгоритму rsa с ключём размером 4096 бит

По алгоритму ed25519 создали ключ SSH (рис. 3.12).

3.5 Настройка gh

Для начала авторизовались с помощью команды `gh auth login` (рис. 3.14).

```
root@aioivanova:~# 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? /root/.ssh/id_rsa.pub
? Title for your SSH key: fedora
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 0672-C757
Press Enter to open https://github.com/login/device in your browser...
Authorization required, but no authorization protocol specified

Error: cannot open display: :0

✓ Authentication complete.
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
! Authentication credentials saved in plain text
✓ Uploaded the SSH key to your GitHub account: /root/.ssh/id_rsa.pub
✓ Logged in as IvanovaAngelina
```

Рисунок 3.14: Авторизация

3.6 Создание репозитория курса на основе шаблона

Создали по шаблону рабочее пространство. На GitHub я уже создала необходимый репозиторий, поэтому необходимо только клонировать его на свою виртуальную машину я (рис. 3.15).

```

root@aoivanova:~# mkdir -p ~/work/study/2022-2023/"Операционные
системы"
root@aoivanova:~# cd ~/work/study/2022-2023/"Операционные систе
мы"
root@aoivanova:~/work/study/2022-2023/Операционные системы# git
clone --recursive git@github.com:IvanovaAngelina/study_2025-20
26_os-intro.git os-intro
Клонирование в «os-intro»...
The authenticity of host 'github.com (140.82.121.4)' can't be e
stablished.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zP
MSvHdkr4UvCOqU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerpri
nt])?

```

Рисунок 3.15: Создание рабочего пространства

3.7 Настройка каталога курса

Перешли в каталог курса, удалили лишние файлы и создали необходимые каталоги (рис. 3.16).

```

root@aoivanova:~/work/study/2022-2023/Операционные системы# cd
~/work/study/2022-2023/"Операционные системы"/os-intro
root@aoivanova:~/work/study/2022-2023/Операционные системы/os-i
ntro# rm package.json
rm: удалить обычный файл 'package.json'?
root@aoivanova:~/work/study/2022-2023/Операционные системы/os-i
ntro# echo os-intro > COURSE
root@aoivanova:~/work/study/2022-2023/Операционные системы/os-i
ntro# make
Usage:
  make <target>

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

```

Рисунок 3.16: Создание рабочего пространства

Отправили файлы на сервер (рис. 3.17).

```
root@aoivanova:~/work/study/2022-2023/Операционные системы/os-i
ntro# git add .
root@aoivanova:~/work/study/2022-2023/Операционные системы/os-i
ntro# git commit -am 'feat(main): make course structure'
[master ff2e9be] feat(main): make course structure
 1 file changed, 1 insertion(+), 1 deletion(-)
root@aoivanova:~/work/study/2022-2023/Операционные системы/os-i
ntro# git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 949 bytes | 949.00 KiB/s, готово.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local ob
ject.
To github.com:IvanovaAngelina/study_2025-2026_os-intro.git
 0046f90..ff2e9be master -> master
root@aoivanova:~/work/study/2022-2023/Операционные системы/os-i
ntro#
```

Рисунок 3.17: Создание рабочего пространства

4 Контрольные вопросы

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

Система контроля версий (Version Control System, VCS) представляет собой программное обеспечение, которое позволяет отслеживать изменения в документах, при необходимости производить их откат, определять, кто и когда внес исправления и т.п.

2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

хранилище - это репозиторий, в котором хранятся все файлы и документы, включая историю изменений. commit - отслеживание и сохранение изменений. история - сохраняет в себе изменения проекта на всех этапах. рабочая копия - копия проекта, связанная с репозиторием.

3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

Централизованные VCS: В централизованных VCS весь код и его история хранятся в одном центральном репозитории. Разработчики работают с копией основного репозитория на своих локальных машинах, откуда отправляют изменения в центральное хранилище.

Примеры: - Subversion (SVN): Один из популярных централизованных VCS. Разработчики могут коммитить изменения в центральный репозиторий и обновлять свои локальные копии. - Perforce: Еще один пример системы контроля версий с централизованным подходом, который широко применяется в больших коммерческих проектах.

Децентрализованные VCS: Децентрализованные VCS позволяют каждому участнику проекта иметь полноценную копию всего репозитория. Это означает, что разработчики имеют доступ ко всей истории проекта локально и могут работать независимо от подключения к сети.

Примеры: - Git: Самая популярная децентрализованная система контроля версий. Разработчики могут коммитить, откатывать изменения и создавать ветки без необходимости доступа к центральному серверу. - Mercurial: Еще один пример децентрализованной VCS, обеспечивающий высокую скорость работы и гибкость в управлении кодом.

4. Опишите действия с VCS при единоличной работе с хранилищем.

При единоличной работе с хранилищем (репозиторием) в системе контроля версий (VCS), разработчик ведет работу над кодом самостоятельно без коллективного взаимодействия. В такой ситуации основной упор делается на сохранение версий кода и отслеживание изменений для личного удобства и безопасности. Вот основные действия, которые могут выполняться при единоличной работе с хранилищем:

- Инициализация репозитория
- Клонирование репозитория
- Коммит изменений
- Просмотр истории изменений
- Создание веток
- Обновление репозитория

- Удаление, перемещение файлов
- Откат изменений
- Игнорирование файлов
- Резервное копировани

5. Опишите порядок работы с общим хранилищем VCS.

Работа с общим хранилищем (репозиторием) в системе контроля версий (VCS) включает в себя совместную работу нескольких разработчиков над одним проектом. Вот порядок действий при работе с общим хранилищем VCS:

- Создание или клонирование репозитория
- Получение последних изменений
- Внесение изменений
- Коммит
- Отправка изменений (push)
- Работа с веткам и т.д.

6. Каковы основные задачи, решаемые инструментальным средством git?

Основные задачи, решаемые инструментальным средством Git, включают в себя управление версиями кода, обеспечение совместной работы над проектами, отслеживание изменений, создание и объединение ветвей разработки, а также возможность отката к предыдущим версиям кода. Git также предоставляет возможность создания резервных копий (backup) и управление изменениями в коде, что делает его ключевым инструментом для разработчиков программного обеспечения.

7. Назовите и дайте краткую характеристику командам 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 .` - добавить все изменённые и/или созданные файлы и/или каталоги
- `git add имена_файлов` - добавить конкретные изменённые и/или созданные файлы и/или каталоги
- `git rm имена_файлов` - удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории)
- `git commit -am „Описание коммита“` - сохранить все добавленные изменения и все изменённые файлы
- `git commit` - сохранить добавленные изменения 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. Приведите примеры использования при работе с локальным и удалённым репозиториями.

`Git pull`: Команда `git pull` используется для извлечения и загрузки содержимого из удаленного репозитория и немедленного обновления локального репозитория

этим содержимым. Слияние удаленных вышестоящих изменений в локальный репозиторий — это обычное дело в процессе совместной работы на основе Git.

9. Что такое и зачем могут быть нужны ветви (branches)?

Ветви (branches) в контексте систем контроля версий, таких как Git, представляют собой параллельные линии разработки, которые позволяют команде разработчиков работать над отдельными фрагментами кода независимо друг от друга. Ветви могут быть полезны для разработки новых функций, исправления ошибок, экспериментов с кодом и поддержания стабильной основной версии программного обеспечения. Создание и использование ветвей помогает упростить процесс разработки, избегая конфликтов и обеспечивая возможность параллельной работы над различными задачами.

10. Как и зачем можно игнорировать некоторые файлы при commit?

Игнорируемый файл — файл, явным образом помеченный для Git как файл, который необходимо игнорировать.

Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты. Вот некоторые распространенные примеры таких файлов (например: /bin, .lock, .tmp, /packages)

Игнорируемые файлы отслеживаются в специальном файле .gitignore, который регистрируется в корневом каталоге репозитория. В Git нет специальной команды для указания игнорируемых файлов: вместо этого необходимо вручную отредактировать файл .gitignore, чтобы указать в нем новые файлы, которые должны быть проигнорированы. Файлы .gitignore содержат шаблоны, которые сопоставляются с именами файлов в репозитории для определения необходимости игнорировать эти файлы.

5 Выводы

В ходе выполнения лабораторной работы мы изучили идеологию и применение средств контроля версий, а также освоили умения по работе с git.

6 Список литературы

1. Лабораторная работа №2 [Электронный ресурс] URL: <https://esystem.rudn.ru/mod/page/view.php>
2. GitHub [Электронный ресурс] URL: <https://github.com/>