

Отчёт по лабораторной работе №2

Управление версиями

Швед Карина Дмитриевна

Содержание

Цель работы	5
Выполнение лабораторной работы	6
Выводы	12
Контрольные вопросы:	13

Список иллюстраций

1	dnf install git	6
2	git config --global user.	7
3	подключение своей учетной записи gitgub	10

Список таблиц

Цель работы

Целью данной работы является изучение идеологии и применения средств контроля версий и освоение умений работать с git.

Выполнение лабораторной работы

Установка git, установка gh

```
[kshved@vbox ~]$ sudo -i
[sudo] пароль для kshved:
[root@vbox ~]# dnf install git
Обновление и загрузка репозитория:
GitHub CLI 100% | 10.5 KiB/s | 56.9
>>> Status code: 404 for https://cli.github.com/packages/rpm/gh-cli.repo/repodata/repomd.xml (IP:
>>> Status code: 404 for https://cli.github.com/packages/rpm/gh-cli.repo/repodata/repomd.xml (IP:
>>> Status code: 404 for https://cli.github.com/packages/rpm/gh-cli.repo/repodata/repomd.xml (IP:
>>> Status code: 404 for https://cli.github.com/packages/rpm/gh-cli.repo/repodata/repomd.xml (IP:
>>> Ошибка Librepo: Cannot download repomd.xml: Cannot download repodata/repomd.xml: All mirrors w
Репозитории загружены.
Пакет Арх. Версия Репозиторий
Установка:
git x86_64 2.48.1-1.fc41 updates
Установка зависимостей:
git-core x86_64 2.48.1-1.fc41 updates
git-core-doc noarch 2.48.1-1.fc41 updates
perl-Error noarch 1:0.17029-16.fc41 fedora
perl-File-Find noarch 1.44-514.fc41 updates
perl-Git noarch 2.48.1-1.fc41 updates
perl-TermReadKey x86_64 2.38-23.fc41 fedora
perl-lib x86_64 0.65-514.fc41 updates
Сводка транзакции:
Установка: 8 пакетов
Общий размер входящих пакетов составляет 8 MiB. Необходимо загрузить 8 MiB.
После этой операции будут использоваться дополнительные 40 MiB (установка 40 MiB, удаление 0 B).
Is this ok [y/N]: y
[1/8] git-0:2.48.1-1.fc41.x86_64 100% | 75.7 KiB/s | 51.7
[2/8] perl-Git-0:2.48.1-1.fc41.noarch 100% | 317.2 KiB/s | 38.4
[3/8] perl-TermReadKey-0:2.38-23.fc41.x86_64 100% | 131.7 KiB/s | 35.6
[4/8] perl-Error-1:0.17029-16.fc41.noarch 100% | 92.9 KiB/s | 40.6
[5/8] perl-File-Find-0:1.44-514.fc41.noarch 100% | 88.5 KiB/s | 25.2
[6/8] perl-lib-0:0.65-514.fc41.x86_64 100% | 125.5 KiB/s | 14.8
[7/8] git-core-0:2.48.1-1.fc41.x86_64 100% | 1.9 MiB/s | 4.7
```

Рис. 1: dnf install git

Далее я задаю имя и email владельца репозитория,utf-8 в выводе сообщений git:

```
[ 9/10] Установка perl-Git-0:2.48.1-1.fc41.noarch      100% | 2.5 MiB/s | 65.0 KiB | 0
[10/10] Установка git-0:2.48.1-1.fc41.x86_64        100% | 54.3 KiB/s | 87.5 KiB | 0
Завершено!
[root@vbox ~]# git config --global user.name "Karina Shved"
[root@vbox ~]# git config --global user.email "karina.shveddd@gmail.com"
[root@vbox ~]#
```

Рис. 2: git config –global user.

Настраиваю верификацию и подписание коммитов git. Создание ssh ключа и добавление его на github

```
[root@vbox ~]# 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:KAxYGE9DWF6MOCZSACm9zhroG4qSTVoqI5ZZs1NpBgM root@vbox
The key's randomart image is:
+---[RSA 4096]-----+
|=%Bo.                |
|EBoo.                |
|B.+                  |
| o.o .               |
|.oo + . S            |
|o.=* .               |
|. @X                 |
| %Bo                 |
|Boo                  |
+----[SHA256]-----+
[root@vbox ~]#
```

SSH keys

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

Authentication keys



Karina

SHA256:Mybvad4NDJtbnklnrZwzabUqT994AaLUX008yEmSV714

Added on Sep 27, 2024

Last used within the last 3 months — Read/write



My Fedora Sway Laptop

SHA256:NIIn75QwKqpfWEehZuu552ub72ox7kC2104PuT8wH+6o

Added on Mar 8, 2025

Never used — Read/write

Check out our guide to [connecting to GitHub using SSH keys](#) or [troubleshoot common SSH problems](#).

Создание ключа gpg Сначала я ввожу `dnf install gnupg`. Эта команда устанавливает GnuPG (GPG) — инструмент для шифрования, подписи данных и управления криптографическими ключами. Затем генерирую ключ

```

[root@vbox ~]# sudo dnf install gnupg
Обновление и загрузка репозитория:
  GitHub CLI                               100% | 10.6 KiB/s | 56.
>>> Status code: 404 for https://cli.github.com/packages/xpm/gh-1.1.repo/repodata/repomd.xml (IP:
>>> Status code: 404 for https://cli.github.com/packages/xpm/gh-cli.repo/repodata/repomd.xml (IP:
>>> Status code: 404 for https://cli.github.com/packages/xpm/gh-cli.repo/repodata/repomd.xml (IP:
>>> Status code: 404 for https://cli.github.com/packages/xpm/gh-cli.repo/repodata/repomd.xml (IP:
>>> Ошибка Librepo: Cannot download repomd.xml: Cannot download repodata/repomd.xml: All mirrors
Репозитории загружены.
Пакет "gnupg2-2.4.5-3.fc41.x86_64" уже установлен.

Нечего делать.
[root@vbox ~]#

```

```

gnupg
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 к...

Ваше полное имя: Karina
Адрес электронной почты: karina.shveddd@gmail.com
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Karina <karina.shveddd@gmail.com>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход?
Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? 0
Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? 0
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 э...
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 э...
grg: создан каталог '/home/kshved/.gnupg/openpgp-revocs.d'
grg: сертификат отзыва записан в '/home/kshved/.gnupg/openpgp-revocs.d'
rev'.
открытый и секретный ключи созданы и подписаны.

pub  rsa4096 2025-03-08 [SC]
     DC6109F2747A9418EAFDDC42C6818638FED6FFDD
uid                               Karina <karina.shveddd@gmail.com>
sub  rsa4096 2025-03-08 [E]

[kshved@vbox ~]$

```

Далее с помощью консоли, собственного токена и API я добавляю grg ключ к себе на гитхаб


```

kshved@vbox ~]$ ^C
kshved@vbox ~]$ git config --global user.signingkey C6818638FED6FFDD
kshved@vbox ~]$ git config --global commit.gpgsign true
kshved@vbox ~]$ git commit -S -m "My signed commit"
fatal: не найден git репозиторий (или один из его каталогов вплоть до точки монтирования /)
Останавливаю поиск на границе файловой системы (так как GIT_DISCOVERY_ACROSS_FILESYSTEM не установлен)
kshved@vbox ~]$ gpg --list-keys --keyid-format LONG
keyboxd]
-----
pub  rsa4096/C6818638FED6FFDD 2025-03-08 [SC]
     DC6109F2747A9418EAFDDC42C6818638FED6FFDD
uid      [ абсолютно ] Karina <karina.shveddd@gmail.com>
sub  rsa4096/24C27FF0B4DEE4F3 2025-03-08 [E]

kshved@vbox ~]$ git config --global user.signingkey
6818638FED6FFDD
kshved@vbox ~]$ git config --global user.signingkey DC6109F2747A9418EAFDDC42C6818638FED6FFDD
kshved@vbox ~]$ git config --global commit.gpgsign true
kshved@vbox ~]$ git config --global gpg.program $(which gpg)
kshved@vbox ~]$ git config --global --list
credential.https://github.com.helper=
credential.https://github.com.helper=!usr/local/bin/gh auth git-credential
credential.https://gist.github.com.helper=
credential.https://gist.github.com.helper=!usr/local/bin/gh auth git-credential
user.name=Karina Shved
user.email=karina.shveddd@gmail.com
user.signingkey=DC6109F2747A9418EAFDDC42C6818638FED6FFDD
commit.gpgsign=true
gpg.program=/usr/bin/gpg

```

Настройка gh и со-

здание рабочего пространства на основе шаблона

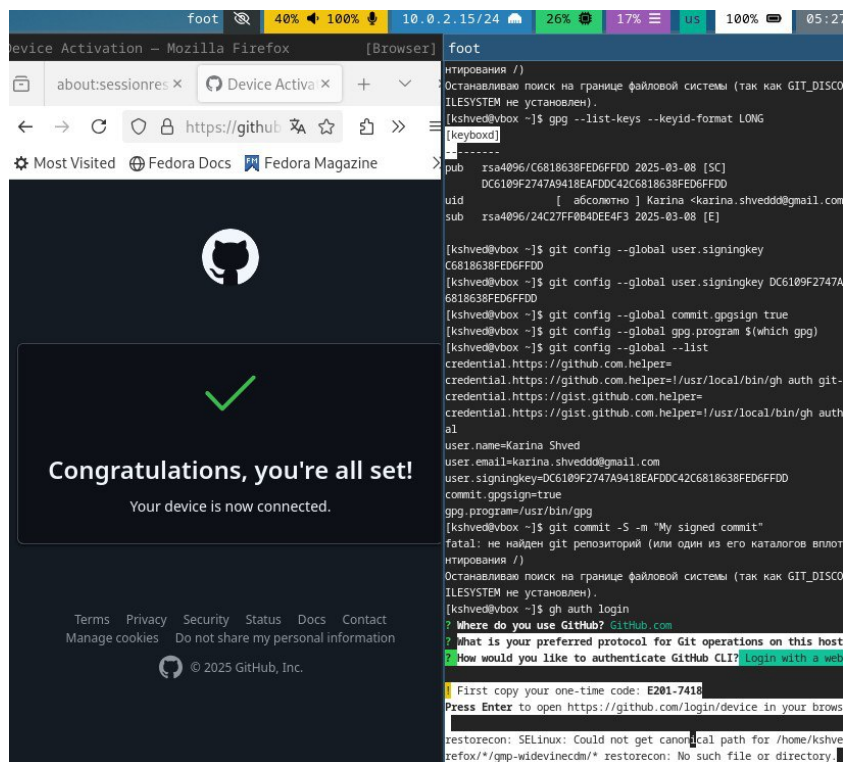


Рис. 3: подключение своей учетной записи gitub

```

[karinashved@vbox Операционные системы]$ git clone --recursive git@github.com:karinashved/os-intro
Клонирование в «os-intro»...
remote: Enumerating objects: 36, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 36 (delta 1), reused 21 (delta 0), pack-reused 0 (from 0)
Получение объектов: 100% (36/36), 19.37 КиБ | 9.69 МБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-pre
истрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laborator
по пути «template/report»
Клонирование в «/home/kshved/work/study/2024-2025/Операционные системы/os-intro»...
remote: Enumerating objects: 111, done.
remote: Counting objects: 100% (111/111), done.
remote: Compressing objects: 100% (77/77), done.
remote: Total 111 (delta 42), reused 100 (delta 31), pack-reused 0 (from 0)
Получение объектов: 100% (111/111), 102.17 КиБ | 649.00 КиБ/с, готово.
Определение изменений: 100% (42/42), готово.
Клонирование в «/home/kshved/work/study/2024-2025/Операционные системы/os-intro»...
remote: Enumerating objects: 142, done.
remote: Counting objects: 100% (142/142), done.
remote: Compressing objects: 100% (97/97), done.
remote: Total 142 (delta 60), reused 121 (delta 39), pack-reused 0 (from 0)
Получение объектов: 100% (142/142), 341.09 КиБ | 1.17 МБ/с, готово.
Определение изменений: 100% (60/60), готово.
Submodule path 'template/presentation': checked out 'c9b2712b4b2d431ad5086c90
Submodule path 'template/report': checked out 'c26e2effe7b3e0495707d82ef561a

```

Создание шаблона рабочего пространства

```

[karinashved@vbox Операционные системы]$ cd ~/work/study/2024-2025/"Операционные системы"/os-intro
[karinashved@vbox os-intro]$ rm package.json
[karinashved@vbox os-intro]$ echo os-intro > COURSE
[karinashved@vbox os-intro]$ make
Usage:
  make <target>

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

[karinashved@vbox os-intro]$ git add .
[karinashved@vbox os-intro]$ git commit -am 'feat(main): make course structure'
[master d80164d] feat(main): make course structure
2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
[karinashved@vbox os-intro]$ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 3 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 952 байта | 952.00 КиБ/с, готово.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:karinashved/study_2024-2025_os-intro.git
   244e136..d80164d  master -> master

```

Настройка каталога курса

Далее отправляю файлы на сервер

Выводы

Мы приобрели практические навыки работы с сервисом github.

Контрольные вопросы:

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

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.

2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.
 - хранилище - пространство на накопителе где расположен репозиторий
 - commit - сохранение состояния хранилища
 - история - список изменений хранилища (коммитов)
 - рабочая копия - локальная копия сетевого репозитория, в которой работает программист. Текущее состояние файлов проекта, основанное на версии, загруженной из хранилища (обычно на последней)
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере. Доступ к нему осуществлялся через специальное клиентское приложение. В качестве примеров таких программных продуктов можно привести CVS, Subversion.

Распределенные системы контроля версий (Distributed Version Control System, DVCS) позволяют хранить репозиторий (его копию) у каждого разработчика, работающего с данной системой. При этом можно выделить центральный репозиторий (условно), в который будут отправляться изменения из локальных и, с ним же эти локальные репозитории будут синхронизироваться. При работе с такой системой, пользователи периодически синхронизируют свои локальные репозитории с центральным и работают непосредственно со своей локальной копией. После внесения достаточного количества изменений в локальную копию они (изменения) отправляются на сервер. При этом сервер, чаще всего, выбирается условно, т.к. в большинстве DVCS нет такого понятия как “выделенный сервер с центральным репозиторием”.

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

Один пользователь работает над проектом и по мере необходимости делает коммиты, сохраняя определенные этапы.

5. Опишите порядок работы с общим хранилищем VCS.

Несколько пользователей работают каждый над своей частью проекта. При этом каждый должен работать в своей ветки. При завершении работы ветка пользователя сливается с основной веткой проекта.

6. Каковы основные задачи, решаемые инструментальным средством git?

- Ведение истории версий проекта: журнал (log), метки (tags), ветвления (branches).
- Работа с изменениями: выявление (diff), слияние (patch, merge).
- Обеспечение совместной работы: получение версии с сервера, загрузка обновлений на сервер.

7. Назовите и дайте краткую характеристику командам git.

- `git config` - установка параметров
- `git status` - полный список изменений файлов, ожидающих коммита
- `git add .` - сделать все измененные файлы готовыми для коммита.
- `git commit -m "[descriptive message]"` - записать изменения с заданным сообщением.
- `git branch` - список всех локальных веток в текущей директории.
- `git checkout [branch-name]` - переключиться на указанную ветку и обновить рабочую директорию.
- `git merge [branch]` — соединить изменения в текущей ветке с изменениями из заданной.
- `git push` - запустить текущую ветку в удаленную ветку.
- `git pull` - загрузить историю и изменения удаленной ветки и произвести слияние с текущей веткой.

8. Приведите примеры использования при работе с локальным и удалённым репозиториями.

- `git remote add [имя] [url]` — добавляет удалённый репозиторий с заданным именем;
- `git remote remove [имя]` — удаляет удалённый репозиторий с заданным именем;
- `git remote rename [старое имя] [новое имя]` — переименовывает удалённый репозиторий;

- `git remote set-url [имя] [url]` — присваивает репозиторию с именем новый адрес;
- `git remote show [имя]` — показывает информацию о репозитории.

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

Ветвление — это возможность работать над разными версиями проекта: вместо одного списка с упорядоченными коммитами история будет расходитьсЯ в определённых точках. Каждая ветвь содержит легковесный указатель HEAD на последний коммит, что позволяет без лишних затрат создать много веток. Ветка по умолчанию называется `master`, но лучше назвать её в соответствии с разрабатываемой в ней функциональностью.

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

- Исключение временных и ненужных файлов (`.log`, `.tmp`).
- Защита конфиденциальных данных (`config.json`, `.env`).
- Исключение средовых файлов (`node_modules/`, `venv/`).
- Уменьшение размера репозитория.

Как:

- Создать `.gitignore`: `touch .gitignore`
- Добавить в него правила, например: `*.log node_modules/ .env`

*Закоммитить `.gitignore`: `git add .gitignore git commit -m "Добавлен .gitignore"`