Лабораторная работа 1

Щетинин Даниил Николаевич

Содержание

Список иллюстраций

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

Установим git и gh, посредством введения команд

```
dnf install git
dnf install gh
```

(рис. 3.1).

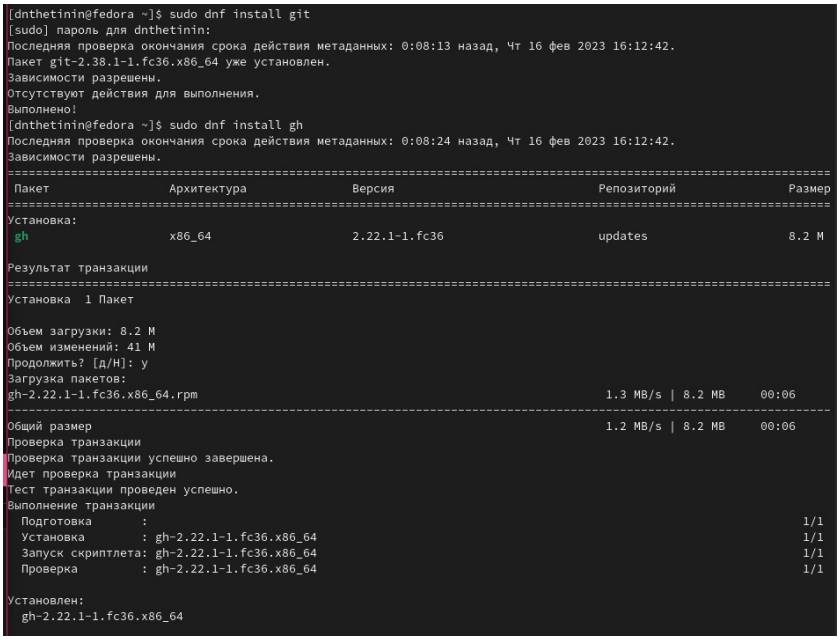

Рис. 3.1: Установка git и gh

Проведём базовую настройку git: Зададим имя и email владельца репозитория, Настроим utf-8 в выводе сообщений git и прочее

(рис. 3.2).

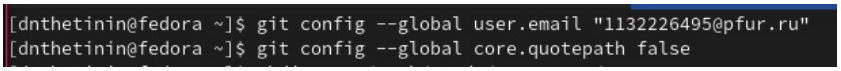

Рис. 3.2: выполненные команды

Перейдём к генерации ключей, начнем с ssh (уже сгенерирован) и pgp (рис. 3.3).

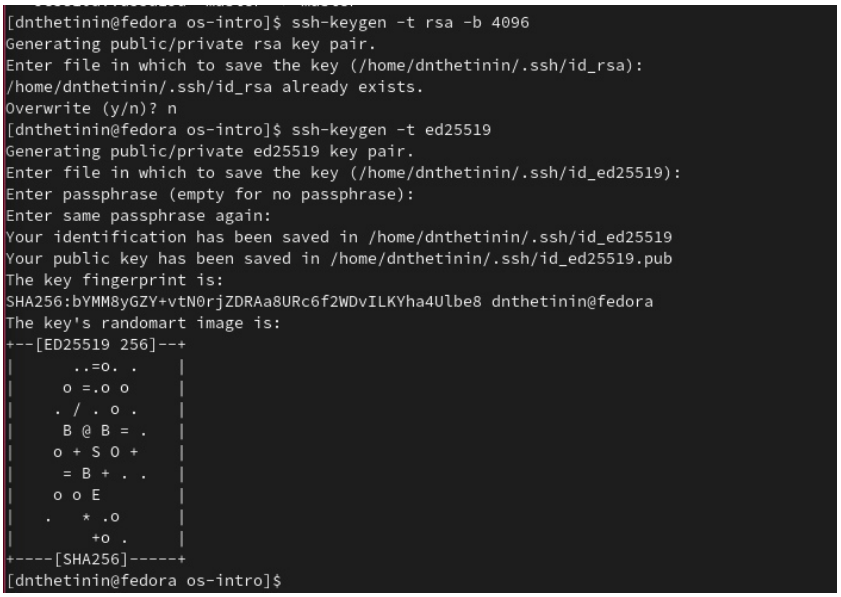

Рис. 3.3: Сгенерированный ключ ssh

(рис. 3.4).

```
[dnthetinin@fedora os-intro]$ gpg --list-secret-keys --keyid-format LONG
gpg: /home/dnthetinin/.gnupg/trustdb.gpg: создана таблица доверия
[dnthetinin@fedora os-intro]$ 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
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)
Срок действия ключа не ограничен
Все верно? (y/N) y

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: Даниил
Адрес электронной почты: 1132226495@pfur.ru
Примечание:
Используется таблица символов 'utf-8'.
Вы выбрали следующий идентификатор пользователя:
    "Даниил <1132226495@pfur.ru>"
```

Рис. 3.4: Сгенерированный ключ pgp

После чего создаем учетную запись на github:

(рис. 3.5).

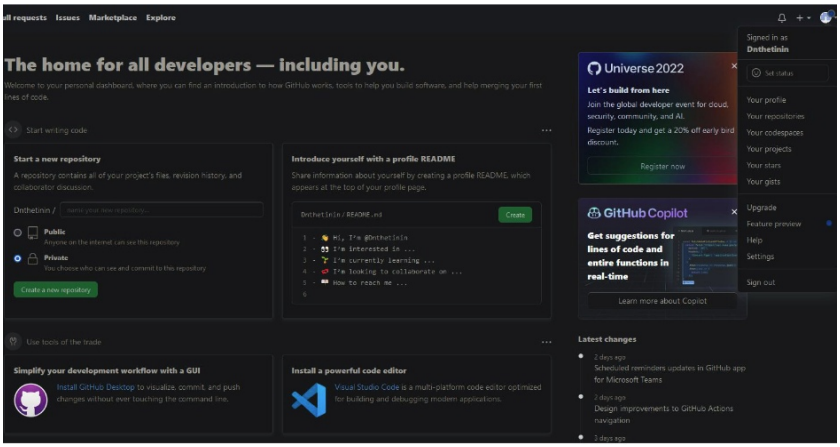

Рис. 3.5: Домашняя страница, созданная учетная запись

Добавим pgp ключ
(рис. 3.6).

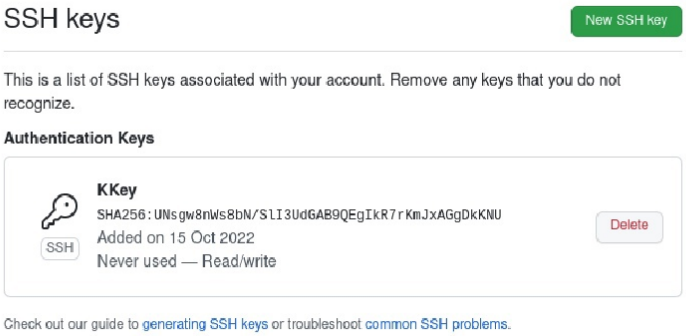

Рис. 3.6: добавленный ключ

Добавив ключ, закончим настройку git, авторизуемся в gh:
(рис. 3.7).

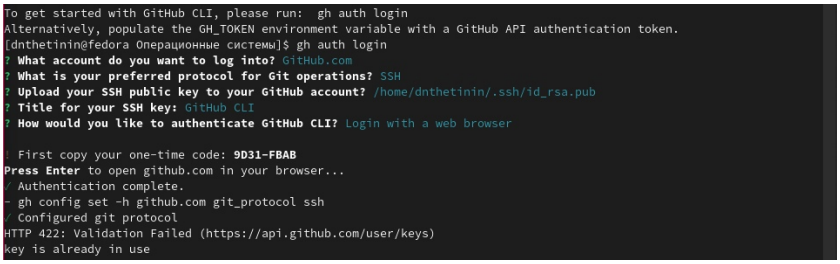

Рис. 3.7: настройка и авторизация

Создадим репозиторий курса и настроем его
(рис. 3.8).

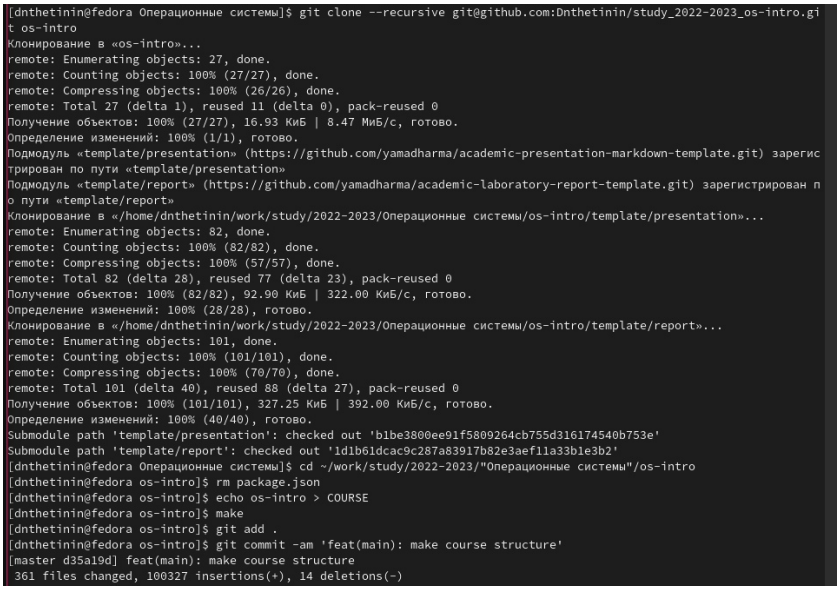

Рис. 3.8: использованные команды

(рис. 3.9).

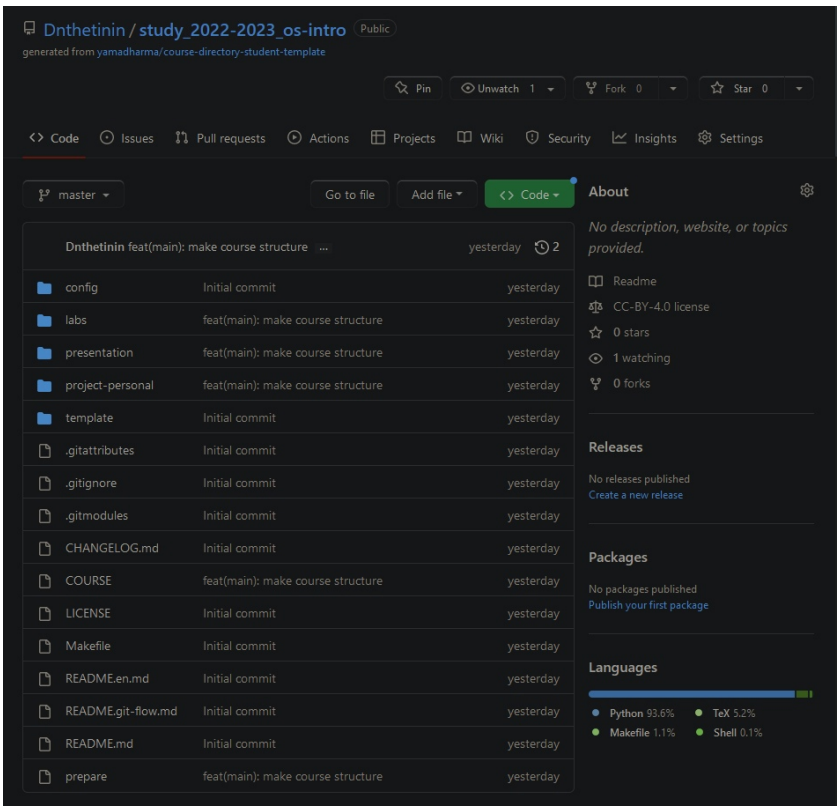

Рис. 3.9: репозиторий на github

4 Контрольные вопросы

Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Контроль версий, также известный как управление исходным кодом, — это практика отслеживания изменений программного кода и управления ими. Системы контроля версий — это программные инструменты, помогающие командам разработчиков управлять изменениями в исходном коде с течением времени. **Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.** Репозиторий - хранилище версий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией. Рабочая копия - копия проекта, связанная с репозиторием commit - сохранение изменений в репозитории

```
**Что представляют собой и чем отличаются централизованные и децентрализованные V
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. Децентрализованные системы контроля версий —
СКВ, которые позволяют клиенту полностью хранить у себя копию
репозитория проекта. Примеры: Git, Mercurial, Bazaar или, например, Darcs.

**Опишите действия с VCS при единоличной работе с хранилищем.**
```

Хранилище является разновидностью файл-сервера, однако не совсем обычного. •Хранилище запоминает каждое внесенное изменение: -любое изменение любого файла, -изменения в самом дереве каталогов, такие как добавление, удаление и реорганизация файлов и каталогов. •При чтении данных из хранилища клиент обычно видит только последнюю версию дерева файлов. •Клиент также имеет возможность просмотреть предыдущие состояния файловой системы. •Вопросы типа «Что содержал этот каталог в прошлую среду?», «Кто был последним, изменявшим этот файл, и какие вносились изменения?»

Опишите порядок работы с общим хранилищем VCS. Пользователь получает нужную версию файлов. После того, как он внес необходимые изменения, пользователь размещает новую версию в хранилище.

Каковы основные задачи, решаемые инструментальным средством git? Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом. Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю

изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

Назовите и дайте краткую характеристику командам git.
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 ветк
it checkout имя_ветки

Приведите примеры использования при работе с локальным и удалённым репозиториями Отправка всех произведённых изменений локального дерева в центральный репозиторий: git push

Что такое и зачем могут быть нужны ветви (branches)?
Ветви нужны для того, чтобы программисты могли вести с
овместную работу над проектом и не мешать друг другу при
этом. При создании проекта, Git создает базовую ветку.
Она называется master веткой.
Как и зачем можно игнорировать некоторые файлы при commit?
Чтобы игнорировать файл, для которого ранее был сделан
коммит, необходимо удалить этот файл из репозитория, а
затем добавить для него правило в .gitignore .

5 Выводы

Я успешно применил средства контроля версий.

и освоил умения по работе с git, настроил учетную запись на гитхаб и создал локальный репозиторий