

Лабораторная работа 1

Щетинин Даниил Николаевич

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Контрольные вопросы	13
5	Выводы	17

Список иллюстраций

3.1	Установка git и gh	7
3.2	выполненные команды	8
3.3	Сгенерированный ключ ssh	8
3.4	Сгенерированный ключ pgr	9
3.5	Домашняя страница, созданная учетная запись	9
3.6	добавленный ключ	10
3.7	настройка и авторизация	10
3.8	использованные команды	11
3.9	репозиторий на github	12

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).

```
[dnthetinin@fedora ~]$ sudo dnf install git
[sudo] пароль для dnthetinin:
Последняя проверка окончания срока действия метаданных: 0:08:13 назад, Чт 16 фев 2023 16:12:42.
Пакет git-2.38.1-1.fc36.x86_64 уже установлен.
Зависимости разрешены.
Отсутствуют действия для выполнения.
Выполнено!
[dnthetinin@fedora ~]$ sudo dnf install gh
Последняя проверка окончания срока действия метаданных: 0:08:24 назад, Чт 16 фев 2023 16:12:42.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий  Размер
=====
Установка:
gh          x86_64       2.22.1-1.fc36  updates      8.2 М
=====
Результат транзакции
=====
Установка 1 Пакет

Объем загрузки: 8.2 М
Объем изменений: 41 М
Продолжить? [д/н]: y
Загрузка пакетов:
gh-2.22.1-1.fc36.x86_64.rpm                                1.3 MB/s | 8.2 MB  00:06
=====
Общий размер                                1.2 MB/s | 8.2 MB  00:06
Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции проведен успешно.
Выполнение транзакции
Подготовка      :                               1/1
Установка       : gh-2.22.1-1.fc36.x86_64    1/1
Запуск скрипта  : gh-2.22.1-1.fc36.x86_64    1/1
Проверка        : gh-2.22.1-1.fc36.x86_64    1/1
Установлен:
gh-2.22.1-1.fc36.x86_64
```

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).

```
[dnthetinin@fedora ~]$ git config --global user.email "1132226495@pfur.ru"
[dnthetinin@fedora ~]$ git config --global core.quotepath false
```

Рис. 3.2: выполненные команды

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

```
[dnthetinin@fedora os-intro]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/dnthetinin/.ssh/id_rsa):
/home/dnthetinin/.ssh/id_rsa already exists.
Overwrite (y/n)? n
[dnthetinin@fedora os-intro]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/dnthetinin/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/dnthetinin/.ssh/id_ed25519
Your public key has been saved in /home/dnthetinin/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:bYMM8yGZY+vtN0rjZDRAa8URc6f2WDvILKYha4U1be8 dnthetinin@fedora
The key's randomart image is:
+--[ED25519 256]--+
|      ..=0. .      |
|      o =.o o      |
|      . / . o .      |
|      B @ B = .      |
|      o + S O +      |
|      = B + . .      |
|      o o E          |
|      . * .o         |
|      +o .           |
+----[SHA256]-----+
[dnthetinin@fedora os-intro]$
```

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 gpg

После чего создаем учетную запись на github:
(рис. 3.5).

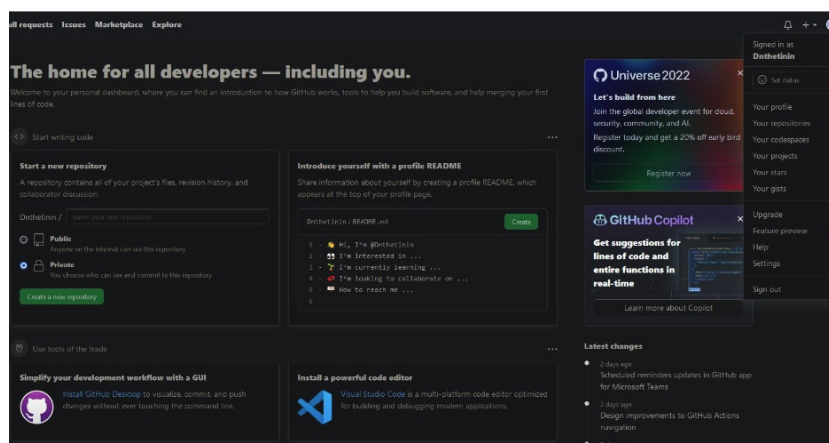


Рис. 3.5: Домашняя страница, созданная учетная запись

Добавим ргр ключ

(рис. 3.6).

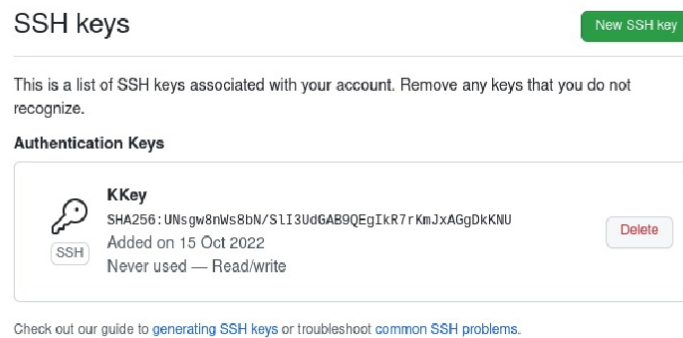


Рис. 3.6: добавленный ключ

Добавив ключ, закончим настройку git, авторизуемся в gh:

(рис. 3.7).

```
To get started with GitHub CLI, please run: gh auth login
Alternatively, populate the GH_TOKEN environment variable with a GitHub API authentication token.
[dnthetin@fedora Операционные системы]$ 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/dnthetin/.ssh/id_rsa.pub
? Title for your SSH key: GitHub CLI
? How would you like to authenticate GitHub CLI? Login with a web browser

First copy your one-time code: 9D31-FBAB
Press Enter to open github.com in your browser...
Authentication complete.
- gh config set -h github.com git_protocol ssh
Configured git protocol
HTTP 422: Validation Failed (https://api.github.com/user/keys)
key is already in use
```

Рис. 3.7: настройка и авторизация

Создадим репозиторий курса и настроим его

(рис. 3.8).

```

[dnthetinin@fedora Операционные системы]$ git clone --recursive git@github.com:dnthetinin/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 11), reused 11 (delta 0), pack-reused 0
Получение объектов: 100% (27/27), 16.93 КиБ | 8.47 МиБ/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/dnthetinin/work/study/2022-2023/Операционные системы/os-intro/template/presentation»...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 82 (delta 28), reused 77 (delta 23), pack-reused 0
Получение объектов: 100% (82/82), 92.99 КиБ | 322.00 КиБ/с, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в «/home/dnthetinin/work/study/2022-2023/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 101, done.
remote: Counting objects: 100% (101/101), done.
remote: Compressing objects: 100% (70/70), done.
remote: Total 101 (delta 40), reused 88 (delta 27), pack-reused 0
Получение объектов: 100% (101/101), 327.25 КиБ | 392.00 КиБ/с, готово.
Определение изменений: 100% (40/40), готово.
Submodule path 'template/presentation': checked out 'b1be3800ee91f5809264cb755d316174540b753e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a83917b82e3aef11a33b1e3b2'
[dnthetinin@fedora Операционные системы]$ cd ~/work/study/2022-2023/"Операционные системы"/os-intro
[dnthetinin@fedora os-intro]$ rm package.json
[dnthetinin@fedora os-intro]$ echo os-intro > COURSE
[dnthetinin@fedora os-intro]$ make
[dnthetinin@fedora os-intro]$ git add .
[dnthetinin@fedora os-intro]$ git commit -am 'feat(main): make course structure'
[master d35a19d] feat(main): make course structure
361 files changed, 100327 insertions(+), 14 deletions(-)

```

Рис. 3.8: использованные команды

(рис. 3.9).

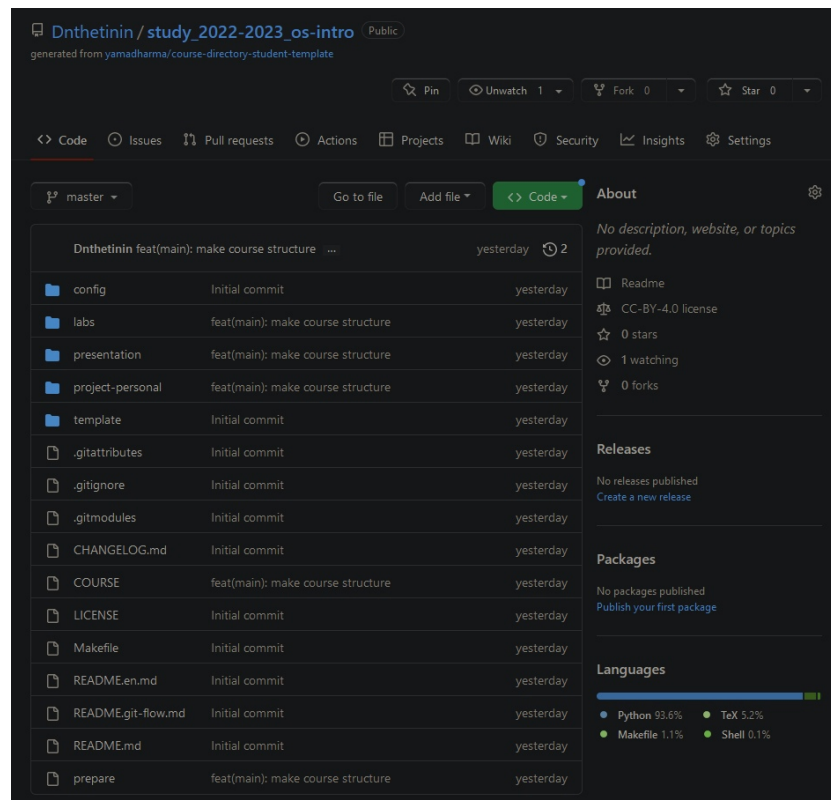


Рис. 3.9: репозиторий на github

4 Контрольные вопросы

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

`git 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 совместную 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