Лабораторная работа №2.

Первоначальная настройка git.

Кучеренко София

Содержание

1 Цель работы **5**

2 Задание **6**

3 Выполнение лабораторной работы **7**

4 Контрольные вопросы **11**

5 Выводы **14**

Список иллюстраций

3.1 Имя и email пользователя 7
3.2 Настройка utf-8 7
3.3 Создание ключа 8
3.4 Список ключей 8
3.5 Название рисунка 9
3.6 Название рисунка 9
3.7 Создание рабочего пространства 10

Список таблиц

1 Цель работы

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

2 Задание

1. Создать базовую конфигурацию для работы с git.
2. Создать ключ SSH.
3. Создать ключ PGP.
4. Настроить подписи git.
5. Зарегистрироваться на Github.
6. Создать локальный каталог для выполнения заданий по предмету.

3 Выполнение лабораторной работы

Установим git и зададим имя и email владельца репозитория:

```
smkucherenko@dk8n53 ~/work/study/2022-2023/Операционные системы/os-intro $ git c
onfig --global user.name "София Кучеренко"
smkucherenko@dk8n53 ~/work/study/2022-2023/Операционные системы/os-intro $ git c
onfig --global user.email "kucherenko.sophie@gmail.com"
```

Рис. 3.1: Имя и email пользователя

Настроим utf-8 в выводе сообщений git:

```
smkucherenko@dk8n53 ~/work/study/2022-2023/Операционные системы/os-intro $ git c
onfig --global core.quotepath false
```

Рис. 3.2: Настройка utf-8

Настройте верификацию и подписание коммитов git:

Генерируем ключ и из предложенных опций выбираем: тип RSA and RSA; размер 4096; срок действия 0 (не истекает никогда), указываем имя и email.

```
smkucherenko@dk8n53 ~/work/study/2022-2023/Операционные системы/os-intro $ gpg --full-generate-key
gpg (GnuPG) 2.2.40; Copyright (C) 2022 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.

Выберите тип ключа:
   (1) RSA и RSA (по умолчанию)
   (2) DSA и Elgamal
   (3) DSA (только для подписи)
   (4) RSA (только для подписи)
  (14) Имеющийся на карте ключ
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
```

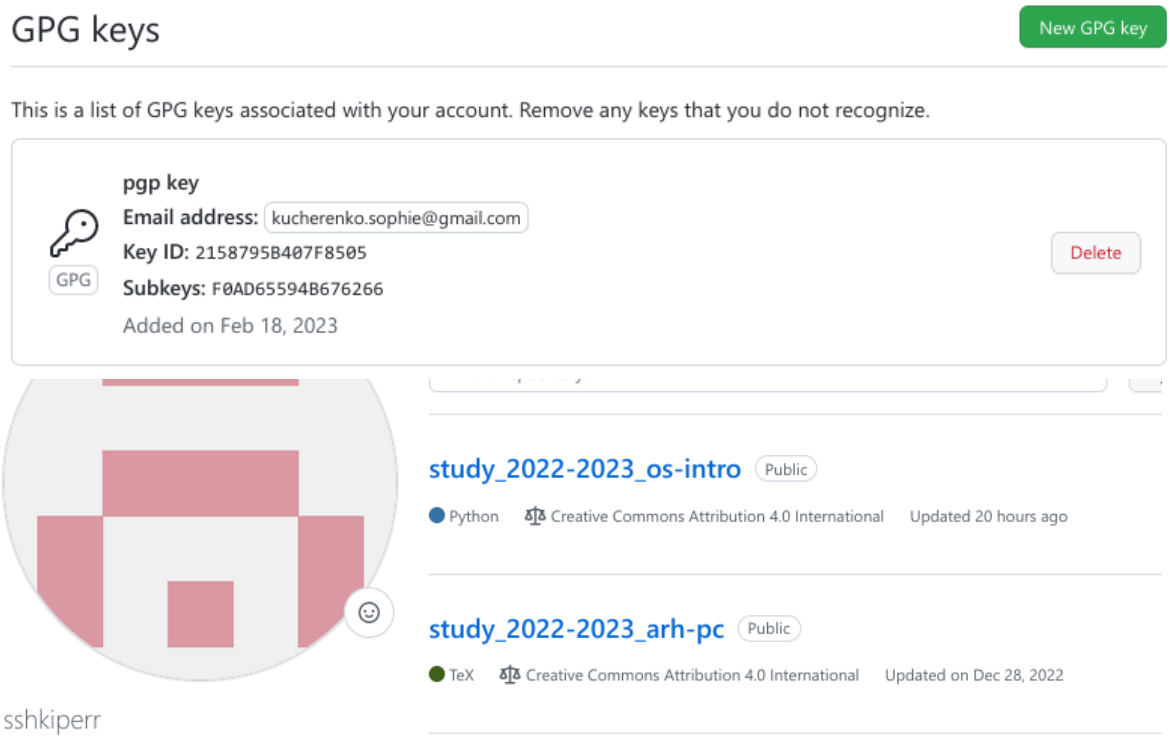
Рис. 3.3: Создание ключа

Выводим список ключей и копируем отпечаток приватного ключа:

```
smkucherenko@dk8n53 ~/work/study/2022-2023/Операционные системы/os-intro $ 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
/afs/.dk.sci.pfu.edu.ru/home/s/m/smkucherenko/.gnupg/pubring.kbx
------------------------------------------------------------------
sec   rsa4096/2158795B407F8505 2023-02-18 [SC]
      A96A303581D0EFC27B36C62D2158795B407F8505
uid                 [  абсолютно ] София <kucherenko.sophie@gmail.com>
ssb   rsa4096/F0AD65594B676266 2023-02-18 [E]
```

Рис. 3.4: Список ключей

Копируем ключ и добавляем его в настройках профиля на GitHub, предварительно авторизоваашись:



Используя введёный email, укажем Git применять его при подписи коммитов:

```
smkucherenko@dk8n53 ~/work/study/2022-2023/Операционные системы/os-intro $ git config --global
 user.signingkey kucherenko.sophie@gmail.com
smkucherenko@dk8n53 ~/work/study/2022-2023/Операционные системы/os-intro $ git config --global
 commit.gpgsign true
smkucherenko@dk8n53 ~/work/study/2022-2023/Операционные системы/os-intro $ git config --global
 gpg.program $(which gpg2)
```

Рис. 3.5: Название рисунка

Зададим имя начальной ветки и укажем некоторые параметры:

```
smkucherenko@dk8n53 ~/work/study/2022-2023/Операционные системы/os-intro $ git config --global
 init.defaultBranch master
smkucherenko@dk8n53 ~/work/study/2022-2023/Операционные системы/os-intro $ git config --global
 core.autocrlf input
smkucherenko@dk8n53 ~/work/study/2022-2023/Операционные системы/os-intro $ git config --global
 core.safecrlf warn
```

Рис. 3.6: Название рисунка

Также перед началом выполнения рабораторных работ я создала рабочее пространство, выполнив ряд действий:

```
mkdir -p ~/work/study/2022-2023/"Операционные системы"
cd ~/work/study/2022-2023/"Операционные системы"
gh repo create study_2022-2023_os-intro --template=yamadharma/course-directory-student-template --public
git clone --recursive git@github.com:<owner>/study_2022-2023_os-intro.git os-intro
```

Настройка каталога курса

- Перейдите в каталог курса:

```
cd ~/work/study/2022-2023/"Операционные системы"/os-intro
```

- Удалите лишние файлы:

```
rm package.json
```

- Создайте необходимые каталоги:

```
echo os-intro > COURSE
make
```

- Отправьте файлы на сервер:

```
git add .
git commit -am 'feat(main): make course structure'
git push
```

Рис. 3.7: Создание рабочего пространства

4 Контрольные вопросы

1. 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. В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в *хранилище*. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

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. Изменив содержимое рабочей копии, разработчик фиксирует (*commit*) сделанные изменения в репозитории. Как правило, фиксация сопровождается небольшим текстовым комментарием.

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.

Для работы с содержимым репозитория каждый разработчик имеет собственную *рабочую копию*. Рабочая копия- «снимок» содержимого репозитория и некоторая служебная информация.

3. В *централизованной* системе все пользователи подключены к центральному владельцу сети или «серверу». Центральный владелец хранит данные, к которым могут получить доступ другие пользователи, а также информацию о пользователях. Эта информация о пользователе может включать профили пользователей, пользовательский контент и многое другое. Централизованная система проста в установке и может быстро развиваться. Но у этой системы есть важное ограничение. Если сервер выходит из строя, система перестает работать должным образом, и пользователи не могут получить доступ к данным. (CVS, Subversion)

У *децентрализованных* систем нет единого центрального владельца. Вместо этого они используют нескольких центральных владельцев, каждый из которых обычно хранит копию ресурсов, к которым пользователи могут получить

доступ. Децентрализованная система может быть так же уязвима к сбоям, как и централизованная. Однако по своей конструкции система более устойчива к неисправностям. (Git, Mercurial)

4. При единоличной работе с хранилищем все изменения, созданные пользователем, не влияют на общий репозиторий.
5. Опишите порядок работы с общим хранилищем VCS: Из общего хранилища можно получать изменения проекта.
6. Каковы основные задачи, решаемые инструментальным средством git? git позволяет нескольким людям работать над одним проектом.
7. add - добавить файлы в коммит, push - отправить коммит на удалённый репозиторий; pull - импортировать проект с удалённого репозитория.
8. Приведите примеры использования при работе с локальным и удалённым репозиториями.

Ответ:

9. Что такое и зачем могут быть нужны ветви (branches)? Создав новую ветвь, можно, не вредя проекту, работать над конкретной частью проекта.
10. Как и зачем можно игнорировать некоторые файлы при commit? some files may well be user specific. gitignore нужен для скрытия файлов и папок от системы контроля версий Git. Обычно скрывают конфигурационные файлы (особенно с паролями), временные файли и папки.

5 Выводы

В результате выполнения лабораторной работы была изучена идеология средств контроля версий, а также были приобретены навыки по работе с git.