

Лабораторная работа №2

Дисциплина: Операционные системы

Алиева Милена Арифовна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Ответы на контрольные вопросы	13
5	Выводы	16

Список иллюстраций

3.1	Указание имени и владельца репозитория	7
3.2	Установка имени начальной ветки, параметров autocrlf и safecrlf .	8
3.3	Генерация приватного и открытого ключей SSH	8
3.4	Созданный ключ	8
3.5	Генерация GPG ключа	9
3.6	Сгенерённый GPG ключ	9
3.7	Авторизация	10
3.8	Создание репозитория на основе шаблона	10
3.9	Клонирование репозитория	11
3.10	Папка “Операционные системы”	11
3.11	Переход в каталог курса, удаление лишних файлов, создание необходимых каталогов и отправка файлов на сервер	12

Список таблиц

1 Цель работы

Целью работы является изучить идеологию и применение средств контроля версий git. Приобрести практические навыки по работе с системой git.

2 Задание

- 1) Базовая настройка git
- 2) Создание SSH ключа
- 3) Создание GPG ключа
- 4) Создание рабочего пространства и репозитория курса на основе шаблона
- 5) Настройка каталога курса

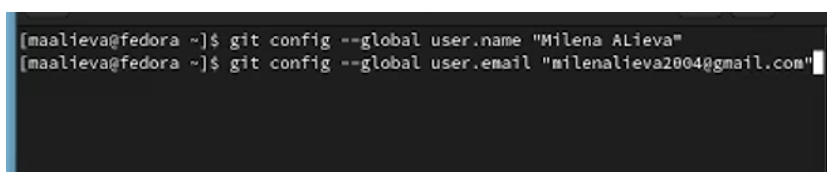
3 Теоретическое введение

Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` с различными опциями

Также перечислим наиболее часто используемые команды `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` # Выполнение лабораторной работы

1. Сначала сделаем предварительную конфигурацию `git`. Откроем терминал и введём следующие команды, указав имя и email владельца репозитория и настроим `utf-8` в выводе сообщений `git` (рис. [3.1])



```
[maalieva@fedora ~]$ git config --global user.name "Milena ALieva"
[maalieva@fedora ~]$ git config --global user.email "milenalieva2004@gmail.com"
```

Рис. 3.1: Указание имени и владельца репозитория

2. Зададим имя начальной ветки (будем называть её `master`), также зададим параметры `autocrlf` и `safecrlf` (рис. [3.2])

```
[maalieva@fedora ~]$ git config --global core.autocrlf input
[maalieva@fedora ~]$ git config --global core.safecrlf warn
```

Рис. 3.2: Установка имени начальной ветки, параметров autocrlf и safecrlf

3. Для последующей идентификации пользователя на сервере репозитория сгенерируем пару ключей (приватный и открытый) (рис. [3.3])

```
[maalieva@fedora ~]$ ssh-keygen -C "Milena Alieva <milenalieva2004@gmail.com>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/maalieva/.ssh/id_rsa):
/home/maalieva/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/maalieva/.ssh/id_rsa
Your public key has been saved in /home/maalieva/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:pP8u0QdU87R9pQ2BxKULysHPdfz+H/n6B0jIVeBqCU Milena Alieva <milenalieva2004@gmail.com>
The key's randomart image is:
+---[RSA 3072]---+
  |                 |
  |  =+*=..+       |
  |   E + +=..+    |
  |  . = . . + +   |
  | =.O.O .. .    |
  | . S...+       |
  | O.=O ++O      |
  | ...+.O.O      |
  | .. ...O       |
  +---+-----+

```

Рис. 3.3: Генерация приватного и открытого ключей SSH

4. Далее загрузим сгенерённый открытый ключ. Для этого зайдём на сайт под своей учетной записью и перейдем в Setting, затем в боковом меню выберем SSH and GPG keys и нажмём на New SSH key. Скопировав из локальной консоли ключ в буфер обмена вставляем ключ в появившееся на сайте поле и указываем для ключа имя (Title)(рис. [3.4])

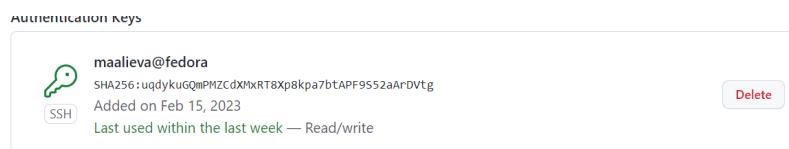


Рис. 3.4: Созданный ключ

5. Для последующей идентификации пользователя на сервере репозитория сгенерируем Gpg ключ (рис. [3.5])

```
[maalieva@fedora ~]$ 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
/home/maalieva/.gnupg/pubring.kbx
-----
sec   rsa4096/01A92FBC7D376CAF 2023-02-15 [SC]
      2B7E0E0B63C7DFA1A2BE91B01A92FBC7D376CAF
uid   [ абсолютно ] Milena <milenalieva2004@gmail.com>
ssb   rsa4096/0458DDF7D9C0B0FD 2023-02-15 [E]
```

Рис. 3.5: Генерация GPG ключа

6. Сгенерённый GPG ключ (рис. [3.6])

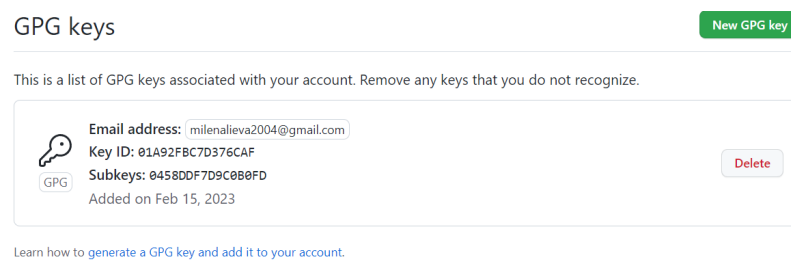


Рис. 3.6: Сгенерённый GPG ключ

7. Авторизация на гитхаб (рис. [3.7])

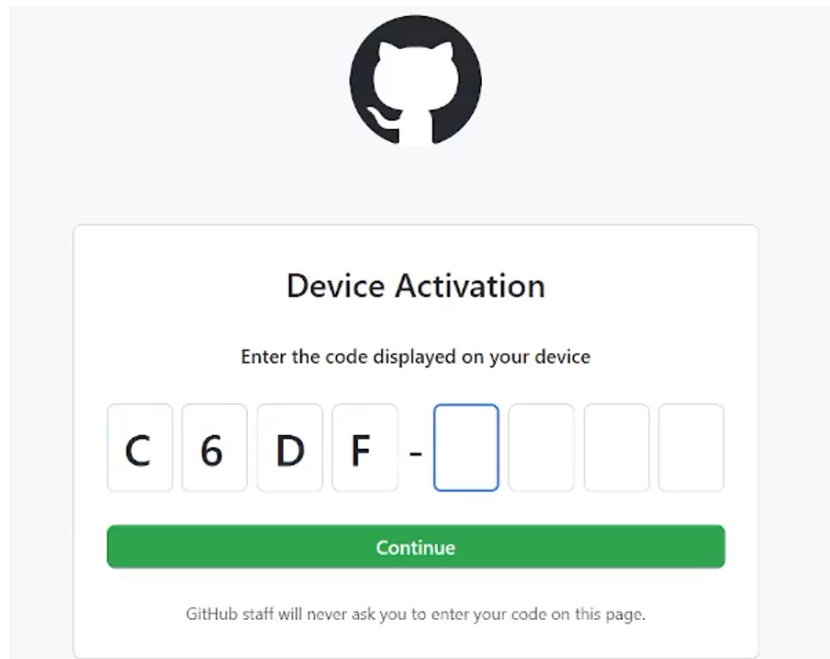


Рис. 3.7: Авторизация

8. Создадим репозиторий на основе шаблона можно через web-интерфейс github. Перейдём на страницу репозитория с шаблоном курса, выберем Use this template. В открывшемся окне зададим имя репозитория (Repository name) study_2022–2023_os-intro и создадим репозиторий (Create repository from template) (рис. [3.8])

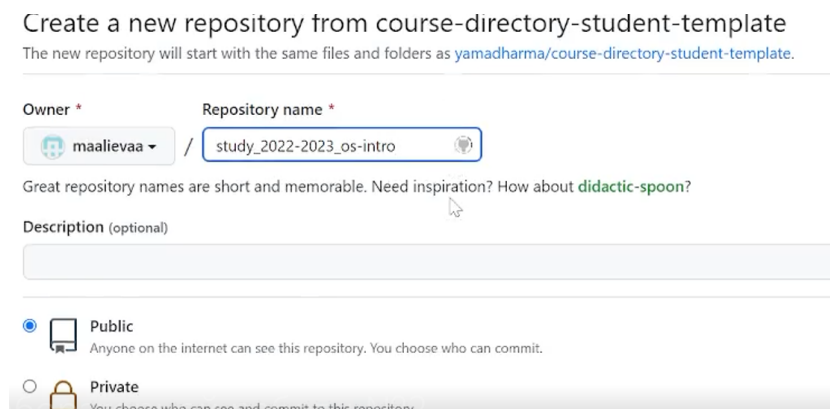


Рис. 3.8: Создание репозитория на основе шаблона

9. Откроем терминал и перейдем в каталог курса. Клонировем созданный репозиторий(рис. [3.9])

```
tation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-r
eport-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/maalieva/work/study/2022-2023/Операционные системы/os-intr
o/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), 92.98 КиБ | 119.80 КиБ/с, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в «/home/maalieva/work/study/2022-2023/Операционные системы/os-intr
o/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 | 535.80 КиБ/с, готово.
Определение изменений: 100% (40/40), готово.
Submodule path 'template/presentation': checked out 'b1be380ee91f5809264cb755d3
16174540b753e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a83917b82e3aef11a3
3b1e3b2'
[maalieva@fedora Операционные системы]$
```

Рис. 3.9: Клонирование репозитория

10. Проверим правильность выполнения, найдем папку “Операционные системы” и пройдем по ней (рис. [3.10])

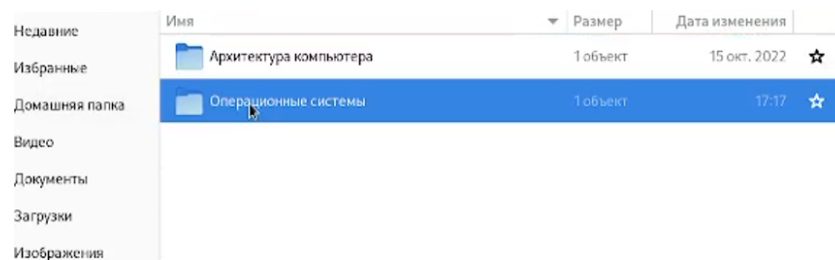


Рис. 3.10: Папка “Операционные системы”

11. Перейдем в каталог курса и удалим лишние файлы. Создадим необходимые каталоги и отправим файлы на сервер (рис. [3.11])

```
[maalieva@fedora os-intro]$ rm package.json
[maalieva@fedora os-intro]$ echo os-intro > COURSE
[maalieva@fedora os-intro]$ make
[maalieva@fedora os-intro]$ git add .
[maalieva@fedora os-intro]$
```

Рис. 3.11: Переход в каталог курса, удаление лишних файлов, создание необходимых каталогов и отправка файлов на сервер

4 Ответы на контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначены? Система контроля версий — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое. Системы контроля версий (Version Control System, VCS) успешно 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. Репозиторий - хранилище версий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией. Commit — отслеживание изменений, сохраняет разницу в изменениях. Рабочая копия - копия проекта, связанная с репозиторием (текущее состояние файлов проекта, основанное на версии из хранилища)
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. В централизованном - одно основное хранилище всего проекта; каждый пользователь копирует

необходимые ему файлы из репозитория, изменяет и, затем, добавляет свои изменения обратно В децентрализованном - у каждого пользователя свой вариант (возможно не один) репозитория; присутствует возможность добавлять и забирать изменения из любого репозитория.

4. Опишите действия с VCS при единоличной работе с хранилищем. Необходимо создать и подключить удаленный репозиторий. Затем по мере изменения проекта отправлять эти изменения на сервер.
5. 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.
6. Каковы основные задачи, решаемые инструментальным средством git? Обеспечить удобство командной работы, хранить информацию о всех изменениях.
7. 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` 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`
8. Приведите примеры использования при работе с локальным и удалённым репозиториями. Отправка всех произведённых изменений локального дерева в центральный репозиторий: `git push`

9. Что такое ветви и зачем могут быть нужны ветви (branches)? Ветвь - один из параллельных участков истории в одном хранилище, исходящих из одной версии. Между ветками возможно слияние, что используется для создания новых функций.
10. Как и зачем можно игнорировать некоторые файлы при commit? Есть такие файлы, зачастую временные, которые не нужно добавлять в репозиторий (например, объектные файлы). Так, можно прописать шаблоны игнорируемых при добавлении в репозиторий файлов.

5 Выводы

В ходе выполнения лабораторной работы я изучила идеологию и применение средств контроля версий git, а также приобрела практические навыки по работе с системой git.