Лабораторная работа №2

Первоначальная настройка git

Майзингер Эллина Сергеевна НПИбд-02-22

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Изучить идеологию и применение средств контроля версий и освоить умения по работе с git

2 Задание

1) Изучить основные функции при работе с git
2) Склонировать репозиторий
3) Создать ключи
4) Настроить рабочее пространство

3 Теоретическое введение

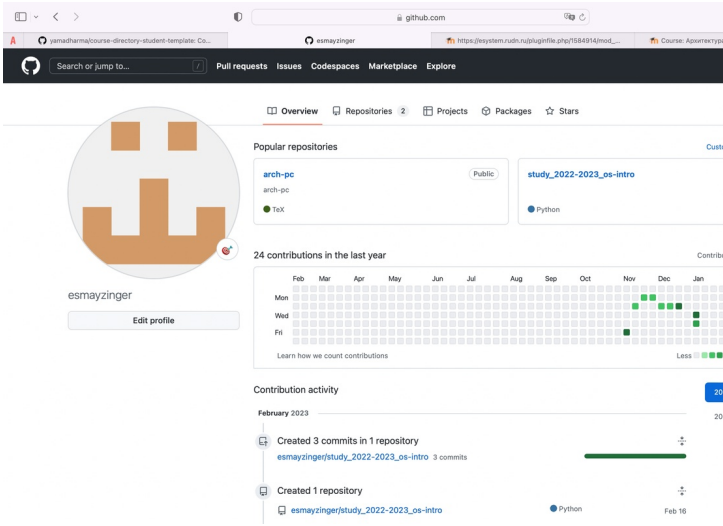
Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется.

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.

4 Выполнение лабораторной работы

4.1 Настройка github

На курсе “Архитектура компьютера” мы уже создавали учетную запись на гит-



хаб, поэтому будем использовать ее. (рис. [**??**])

Базовая настройка git

Зададим имя и email владельца репозитория:

git config –global user.name “Name Surname”

git config –global user.email “work@mail”

Настроим utf-8 в выводе сообщений git:

git config –global core.quotepath false

Зададим имя начальной ветки (будем называть её master):

git config –global init.defaultBranch master

Параметр autocrlf:

git config –global core.autocrlf input

Параметр safecrlf:

git config –global core.safecrlf warn (рис. [**??**])

```
[esmayjzinger@fedora ~]$ git config --global user.name "esmayzi
[esmayjzinger@fedora ~]$ git config --global user.email "113222
[esmayjzinger@fedora ~]$ git config --global core.quotepath fal
[esmayjzinger@fedora ~]$ git config --global init.defaultBranch
[esmayjzinger@fedora ~]$ git config --global core.autocrlf inpu
[esmayjzinger@fedora ~]$ git config --global core.safecrlf warn
```

4.2 Создаём ключ SSH

(рис. [4.1]) В терминале вводим данную команду:

ssh-keygen -t rsa -b 4096

Далее во всех пунктах пользуемся клавишей Enter и получаем наш ключ.

```
[esmayjzinger@fedora ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/esmayjzinger/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/esmayjzinger/.ssh/id_rsa
Your public key has been saved in /home/esmayjzinger/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:TTOSrToDizlt5jjwaIvkBZV4/13mUlOwUAmUW86IoP4 esmayjzinger@fedora
The key's randomart image is:
+---[RSA 4096]----+
|         .++o.   |
|  . ..    +.oo   |
| . +. . + @. .   |
|  o..  . B =.    |
| .. ..  S .=     |
|. .= o... = .    |
| =+.* +. o .     |
```

Рис. 4.1: Создание ключа

Используя команду cat ~/ .ssh/id_rsa/pub копируем наш ключ и далее добавляем его на гитхаб (рис. [4.2])

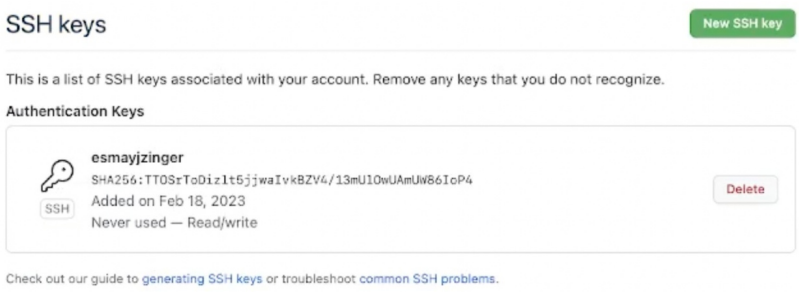


Рис. 4.2: Ключ на гитхабе

4.3 Создание репозитория

Переходим в терминал и авторизуемся с помощью команды gh auth login (рис. [4.3]).

Создаём репозиторий курса на основе шаблона, с помощью команд, указанных в методичке к лабораторной работе:

1. mkdir -p ~/work/study/2021-2022/“Операционные системы”
2. cd ~/work/study/2021-2022/“Операционные системы”
3. gh repo create study_2021-2022_os-intro –template=yamadharma/course-directory-student-template –public
4. git clone –recursive git@github.com:/study_2021-2022_os-intro.git os-intro (рис. [4.4])

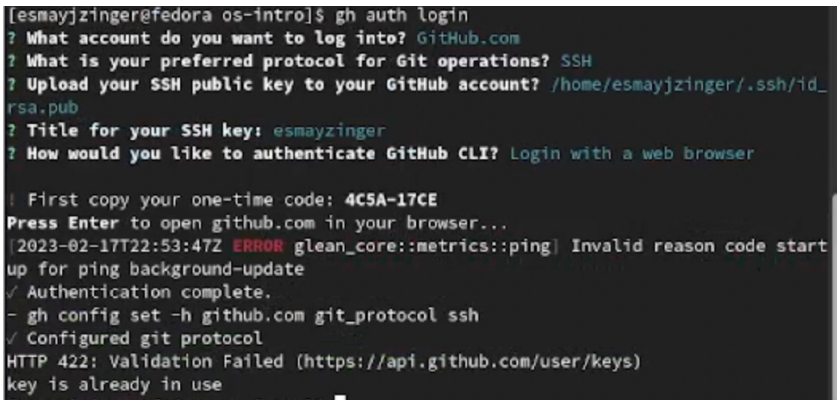


Рис. 4.3: Авторизация

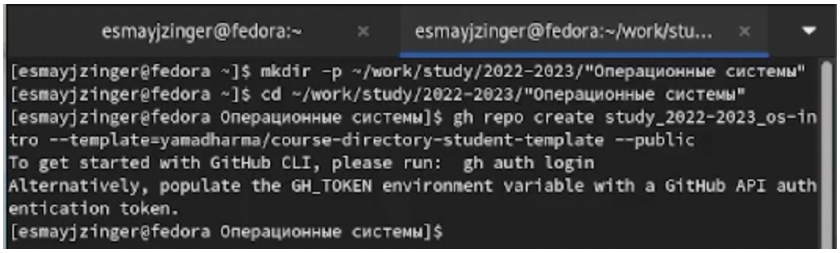


Рис. 4.4: Клонирование репозитория

4.4 Настройка каталога курса

Для этого переходим в него командой:

cd ~/work/study/2021-2022/“Операционные системы”/os-intro

В каталоге “os-intro” нам потребуется удалить файл “package.json”. Выполняем данную задачу командой:

rm package.json (рис. [4.5])

Создаём необходимые каталоги и отправляем наши файлы на сервер (рис. [4.6]).

make COURSE=os-intro

1. git add .
2. git commit -am ‘feat(main): make course structure’

3. git push

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

Рис. 4.5: Настройка необходимых каталогов

```
[esmayjzinger@fedora os-intro]$ git push
Перечисление объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 342.39 КиБ | 2.61 МиБ/с, готово.
Всего 38 (изменений 4), повторно использовано 0 (изменений 0), повторно использо
вано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:esmayzinger/study_2022-2023_os-intro.git
   3ee61d4..8343543  master -> master
[esmayjzinger@fedora os-intro]$
```

Рис. 4.6: Отправка файлов на сервер

5 Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Это программное обеспечение для облегчения работы с изменяющейся информацией. VCS 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.
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Хранилище (repository), или репозитарий, — место хранения всех версий и служебной информации. Commit («[трудовой] вклад», не переводится) — синоним версии; процесс создания новой версии. История – место, где сохраняются все коммиты, по которым можно посмотреть данные о коммитах. Рабочая копия – текущее состояние файлов проекта, основанное на версии, загруженной из хранилища.
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные VCS:одно основное хранилище всего проекта и каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно. Децентрализованные VCS: у каждого пользователя свой вариант (возможно не один) репозитория.
4. Опишите действия с VCS при единоличной работе с хранилищем.
5. Опишите порядок работы с общим хранилищем VCS.
6. Каковы основные задачи, решаемые инструментальным средством git?

Git — это система управления версиями. У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.

7. Назовите и дайте краткую характеристику командам git. git –version (Проверка версии Git) git init (Инициализировать ваш текущий рабочий каталог как Git-репозиторий) git clone https://www.github.com/username/repo-name (Скопировать существующий удаленный Git-репозиторий) git remote (Просмотреть список текущих удалённых репозиториев Git) git remote -v (Для более подробного вывода) git add my_script.py (Можете указать в команде конкретный файл). git add . (Позволяет охватить все файлы в текущем каталоге, включая файлы, чье имя начинается с точки) git commit -am “Commit message” (Вы можете сжать все индексированные файлы и отправить коммит). git branch (Просмотреть список текущих веток можно с помощью команды branch) git –help (Чтобы узнать больше обо всех доступных параметрах и командах) git push origin master (Передать локальные коммиты в ветку удаленного репозитория).
8. Приведите примеры использования при работе с локальным и удалённым репозиториями.
9. Что такое и зачем могут быть нужны ветви (branches)? Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций: чтобы не повредить основному проекту, создается новая ветка специально для экспериментов.
10. Как и зачем можно игнорировать некоторые файлы при commit? Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты.

6 Выводы

В ходе выполнения лабораторной работы изучили идеологию и применение средств контроля версий, а также освоили умения по работе с git.

Список литературы

esystem rudn