

Лабораторная работа 2

Операционные системы

Савурская Полина

Список иллюстраций

| | | |
|-----|-------------------------------|---|
| 3.1 | новый репозиторий | 5 |
| 3.2 | команда git clone | 5 |
| 3.3 | необходимые команды | 6 |
| 3.4 | команда git push | 6 |

1 Цель работы

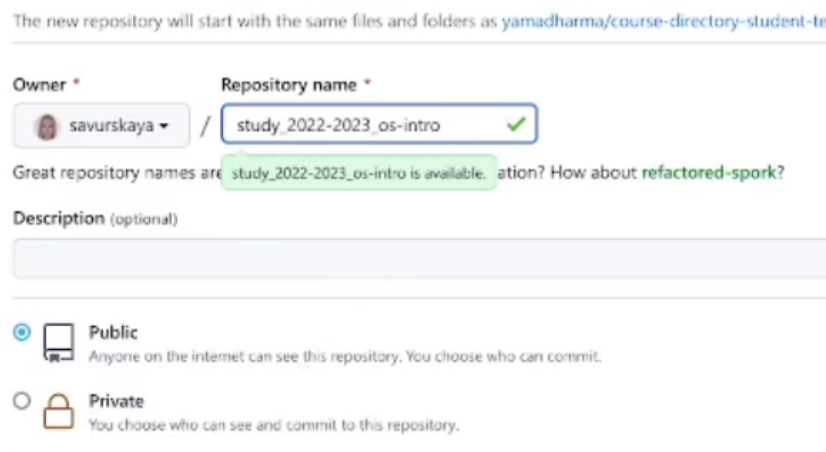
Изучить идеологию и применение средств контроля версий и освоить умения по работе с git.

2 Задание

Ответить на контрольные вопросы: 1) Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? 2) Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. 3) Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. 4) Каковы основные задачи, решаемые инструментальным средством git? 5) Что такое и зачем могут быть нужны ветви (branches)?

3 Выполнение лабораторной работы

- 1) Так как мой SSH ключ, который я создала в сентябре, все еще рабочий, в этой л/р я его не создавала.
- 2) Создаю на Гитхаб новый репозиторий для операционных систем.




The new repository will start with the same files and folders as [yamadhama/course-directory-student-te](#)

Owner * savurskaya / Repository name * study_2022-2023_os-intro ✓

Great repository names are study_2022-2023_os-intro is available. How about [refactored-spork?](#)

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.


☐  **Private**
You choose who can see and commit to this repository.

Рис. 3.1: новый репозиторий

- 3) Создаю шаблон рабочего пространства с помощью команды git clone.



```
os-intro $ ls
os-intro $ git clone --recursive git@github.com:savurskaya/study_2022-2023_os
```

Рис. 3.2: команда git clone

- 4) Удаляю лишние файлы, создаю необходимые каталоги.

```
темы/os-intro $ rm package.json
темы/os-intro $ make COURSE=os-intro
темы/os-intro $ git add .
темы/os-intro $ git commit -am 'feat(main): make course structure'
```

Рис. 3.3: необходимые команды

- 5) Отправляю файлы на сервер.

```
work/study/2022-2023/Операционные системы/os-intro $ git push
38, готово.
(38/38), готово.
используется до 6 потоков
(30/30), готово.
(37/37), 342.39 КиБ | 2.45 МиБ/с, готово.
, повторно использовано 0 (изменений 0), повторно использовано пакетов 0
as: 100% (4/4), completed with 1 local object.
ya/study_2022-2023_os-intro.git
aster -> master
work/study/2022-2023/Операционные системы/os-intro $
```

Рис. 3.4: команда git push

#Ответы на контрольные вопросы

- 1) Система контроля версий (VCS) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов.
- 2) Хранилище-система, которая обеспечивает хранение всех существовавших вариантов файлов
Commit фиксация изменений История-список предыдущих ревизий
Рабочая копия-копия другой ветки
Команде commit можно передать сообщение, описывающее изменения в ревизии. Она также записывает идентификатор пользователя, текущее время и временную зону, плюс список изменённых файлов и их содержимого. Сообщение, описывающее изменения, определяется через опцию -m, или – message. Можно

также вводить сообщения, состоящие из нескольких строк; в большинстве оболочек вы можете сделать это оставив открытую кавычку в конце строки. `commit -m` “добавлен первый файл.

- 3) Системы контроля версий. Централизованная система контроля версий Subversion и децентрализованная система контроля версий Mercurial. Существуют СКВ централизованные, в которых имеется один репозиторий, в который собираются изменения со всех рабочих копий разработчиков, и децентрализованные, когда репозиториев много, и они могут обмениваться изменениями между собой. Централизованные СКВ - репозиторий один. У каждого разработчика своя рабочая копия. Время от времени разработчик может затягивать к себе в рабочую копию новые изменения из репозитория, или проталкивать свои изменения из своей рабочей копии в репозиторий. Прочие особенности централизованных СКВ зависят от реализации.
- 4) Основные задачи: создание ветки, размещение веток, просмотр изменений, фиксация изменений, сообщение из текстового редактора, выборочная фиксация, удаление зафиксированных изменений, игнорирование файлов, просмотр истории, статистика ветки, контроль файлов и каталогов, ветвление, объединение веток, публикация ветки.
- 5) Часто вместо того что бы начинать свой собственный проект, вы хотите предложить изменения для уже готового проекта. Что бы сделать это вам нужно получить копию готовой ветки. Так как эта копия может быть потенциальной новой веткой эта команда называется `branch`: Управление версиями `git branch cd git.dev` Эта команда копирует полную историю ветки и после этого вы можете делать все операции с ней локально: просматривать журнал, создавать и объединять другие ветки.

4 Выводы

Я освоила применение средства контроля версий, а также получила умения работы с git.