Лаборатоная работа №2

Дисциплина Архитектура компьютера

Алехин Давид Андреевич

Содержание

# 1 Цель работы

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

# 2 Задание

1.Базовая настройка git. 2.Сознание рабочего пространства и репозитория курса на основе шаблона. 3.Настройка каталога курса. 4.Добавление файлов с 1 и 2 лабораторной в папки report.

# 3 Теоретическое введение

2.2.1. Системы контроля версий. Общие понятия 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 про- екта. 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, пред- полагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник про- екта (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 хранили- ща 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 кон- фликтов, 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 заблоки- ровать 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 функ- циональные возможности. Например, они могут поддерживать работу с несколькими вер- сиями 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. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным. Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

2.2.2. Система контроля версий Git репозитория Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями. Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией.

2.2.4. Стандартные процедуры работы при наличии центрального репозитория Работа пользователя со своей веткой начинается с проверки и получения изменений из центрального репозитория (при этом в локальное дерево до начала этой процедуры не должно было вноситься изменений): git checkout master git pull git checkout -b имя\_ветки Затем можно вносить изменения в локальном дереве и/или ветке. После завершения внесения какого-то изменения в файлы и/или каталоги проекта необ- ходимо разместить их в центральном репозитории. Для этого необходимо проверить, какие файлы изменились к текущему моменту: git status и при необходимости удаляем лишние файлы, которые не хотим отправлять в центральный репозиторий. Затем полезно просмотреть текст изменений на предмет соответствия правилам ведения чистых коммитов: git diff Если какие-либо файлы не должны попасть в коммит, то помечаем только те файлы, изменения которых нужно сохранить. Для этого используем команды добавления и/или удаления с нужными опциями: git add имена\_файлов git rm имена\_файлов Если нужно сохранить все изменения в текущем каталоге, то используем: git add . Затем сохраняем изменения, поясняя, что было сделано: git commit -am “Some commit message” и отправляем в центральный репозиторий: git push origin имя\_ветки или git push.

# 4 Выполнение лабораторной работы

1.Выполняю пункт 2.4.2. (Базовая настройка git): Создаю учётную запись и создаю предварительную конфигурацию git, после настраиваю utf-8 в выводе сообщений git, задаю имя начальной ветки master и ввожу параметр autocrlf и safecrlf (рис. 1).

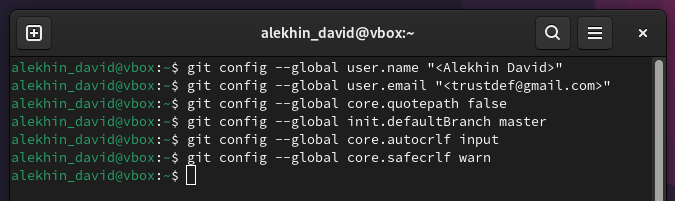


Рис. 1: Пункт 2.4.2

2.Выполняю пункт 2.4.3. (Создание SSH ключа) Генерирую пару ключей далее генерирую открытый ключ (рис. [2]3[4]5).

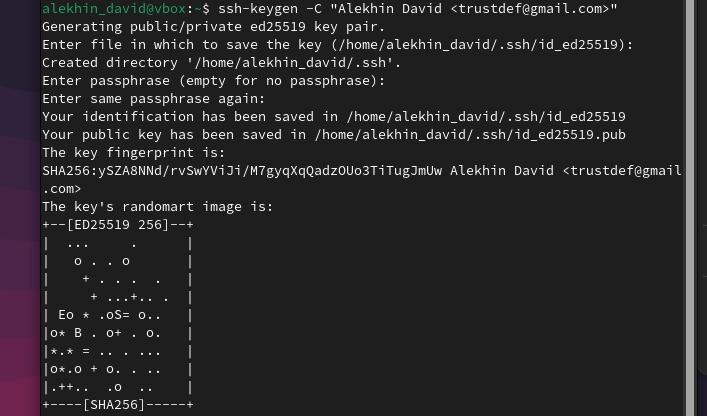


Рис. 2: Пункт 2.4.3

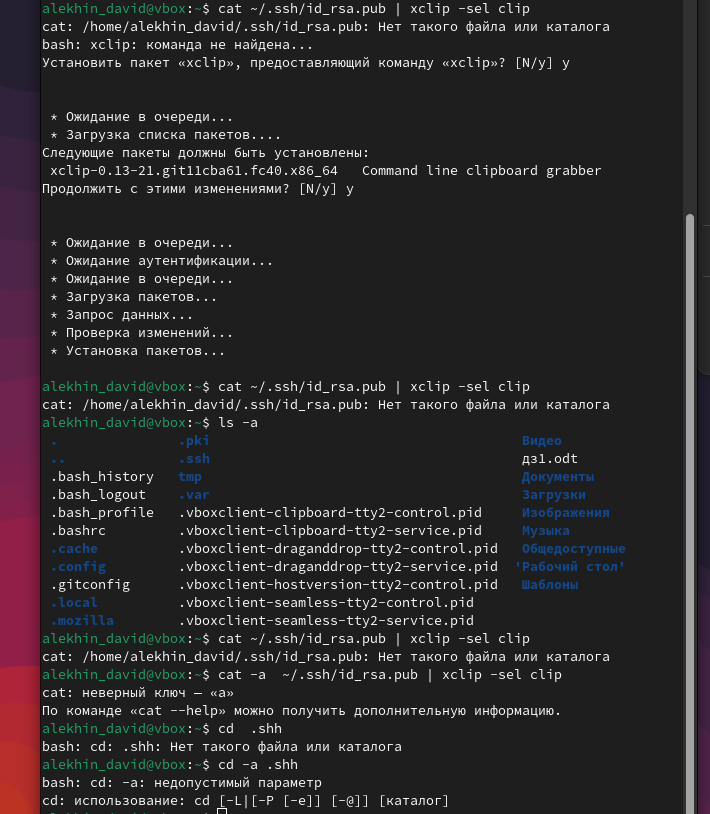


Рис. 3: Пункт 2.4.3

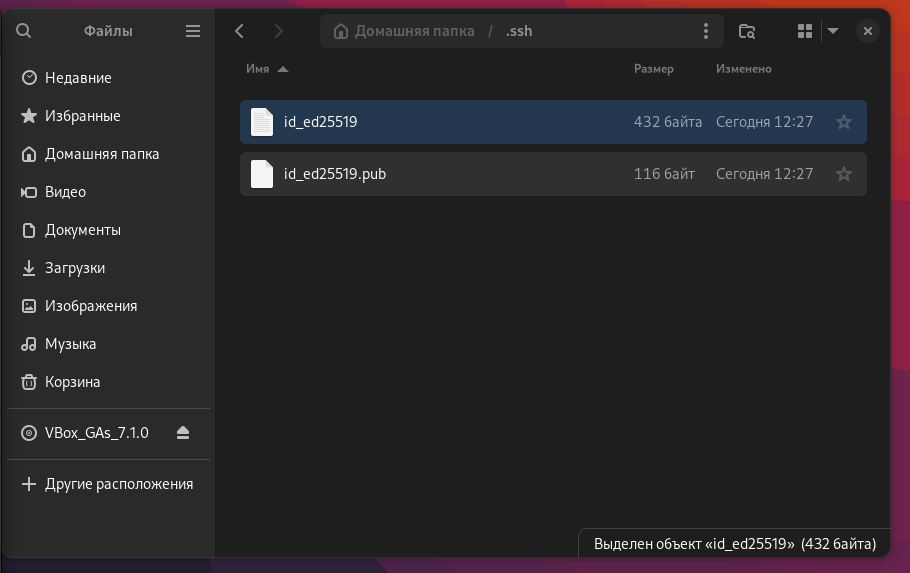


Рис. 4: Пункт 2.4.3

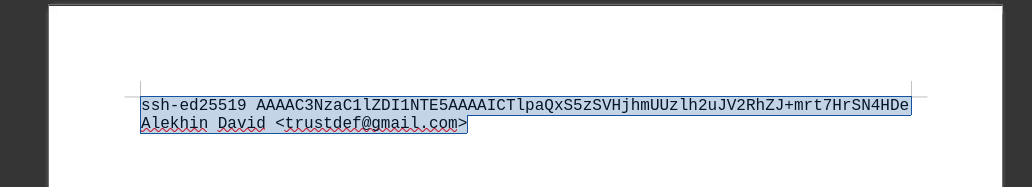


Рис. 5: Пункт 2.4.3

3.Изучаю структуру рабочего пространства и выполняю пункт 2.4.4. (Сознание рабочего пространства и репозитория курса на основе шаблона) Создаю каталог для предмета Архитектура компьютера (рис. 6).

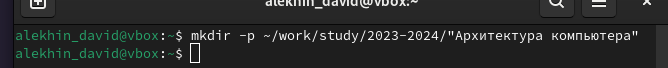


Рис. 6: Пункт 2.4.4

4.Выполняю пункт 2.4.5. (Сознание репозитория курса на основе шаблона), создаю репозиторий фото 7-8.(рис. [7]8).

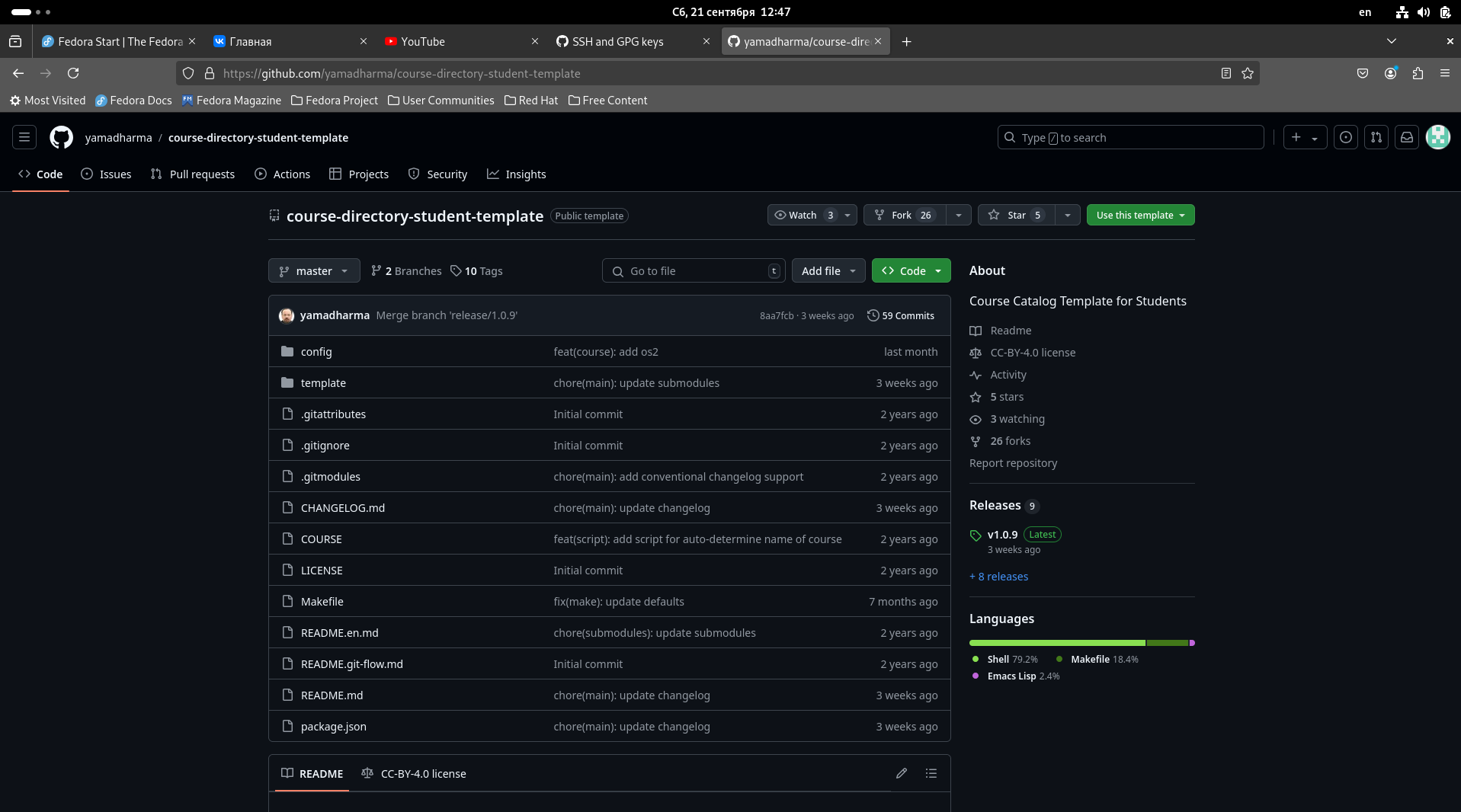


Рис. 7: Пункт 2.4.5

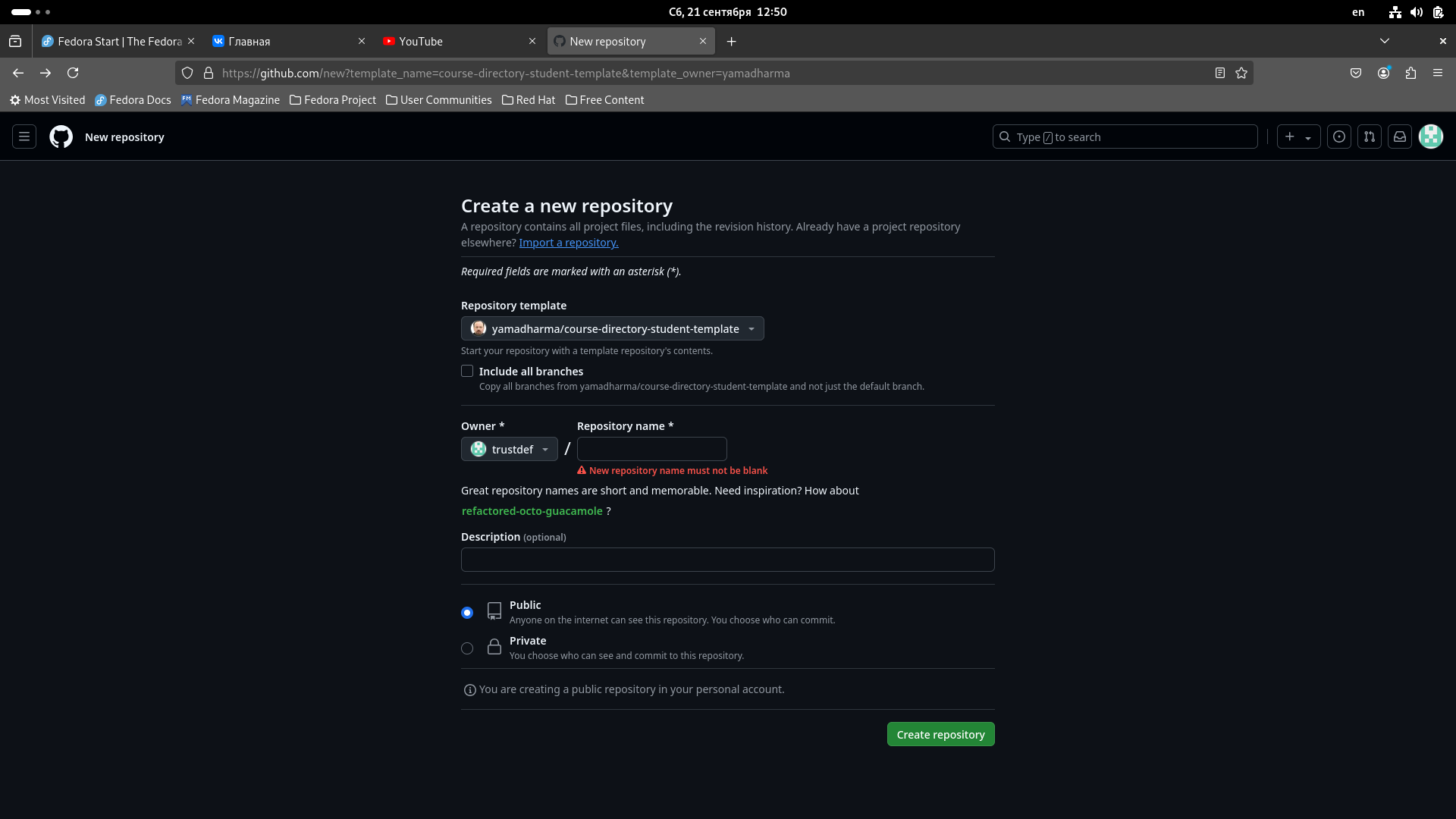


Рис. 8: Пункт 2.4.5

5.Далее клонирую созданный репозиторий в ранее созданный каталог (почему-то не получается перейти в каталог одной командой, перехожу с помощью последовательного перемещения по каталогу с помощью команды cd ), (рис. [9]10).

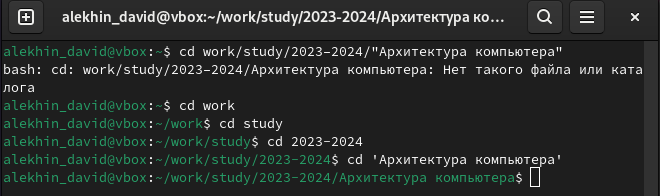


Рис. 9: Пункт 2.4.5 продолжение

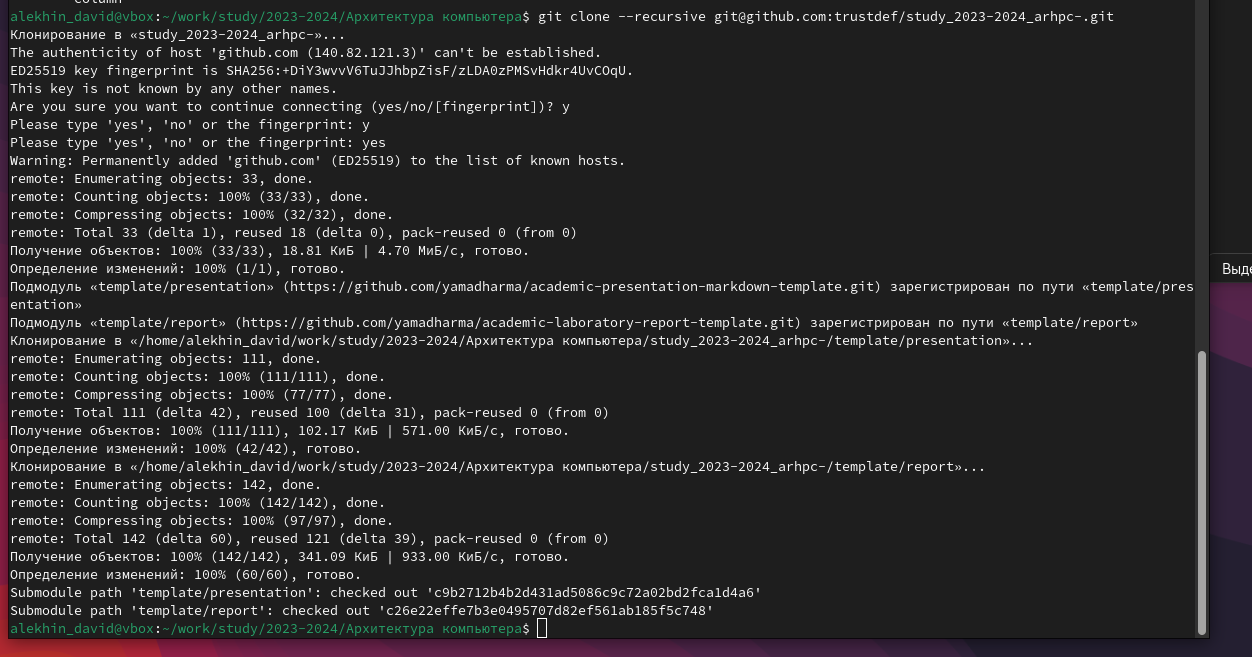


Рис. 10: Пункт 2.4.5 продолжение

6.Выполняю пункт 2.4.6. (Настройка каталога курса) Удаляю лишние файлы, добавляю нужные и отправляю их на сервер (рис. 11).Проверяю правильность создания (рис. [12]13).

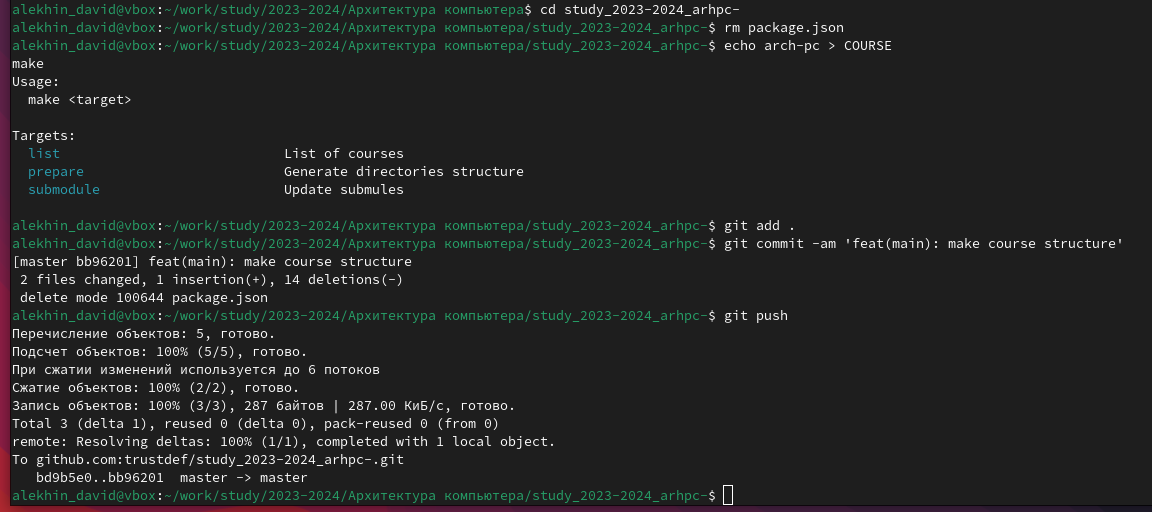


Рис. 11: Пункт 2.4.6

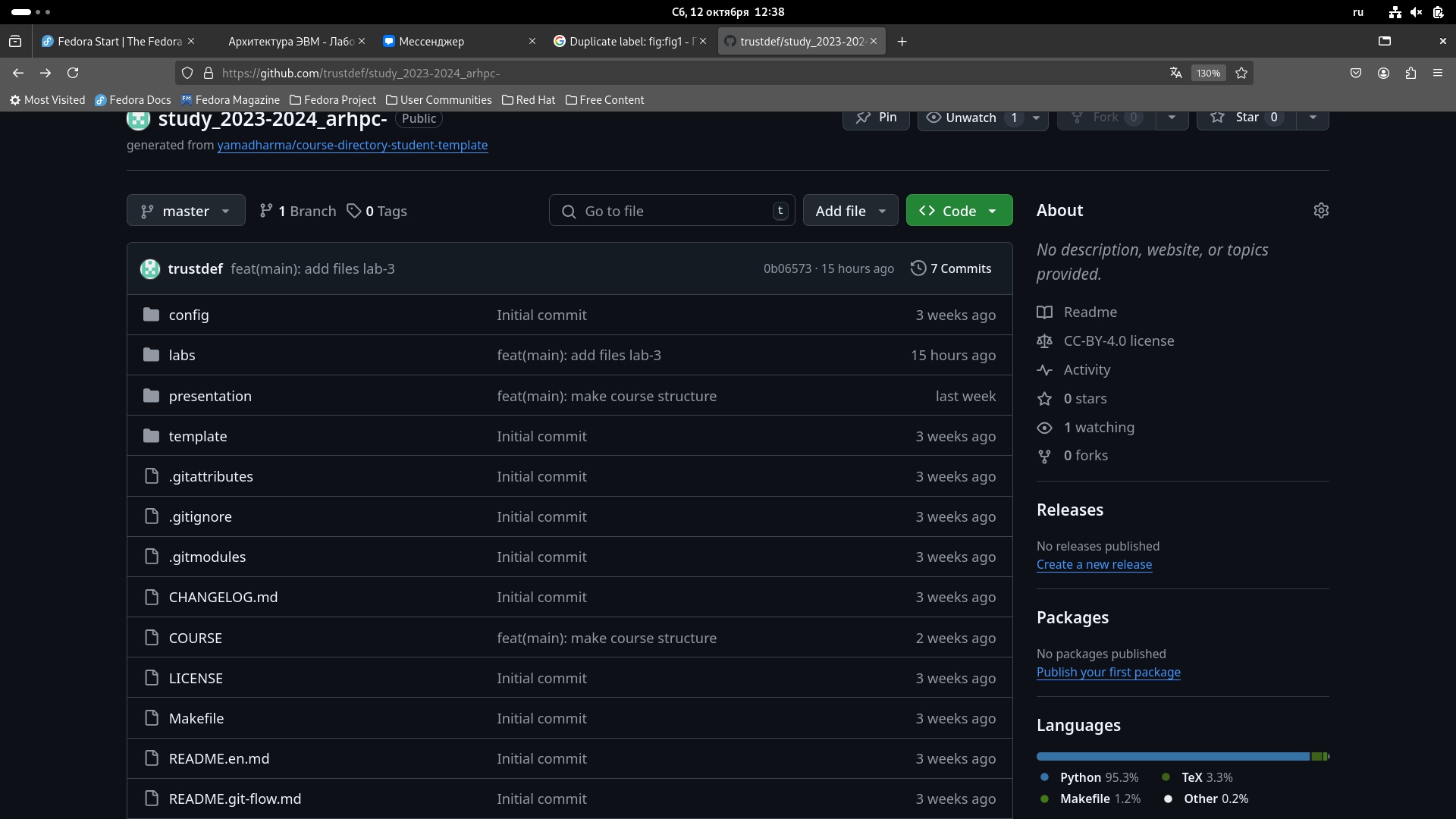


Рис. 12: Пункт 2.4.6

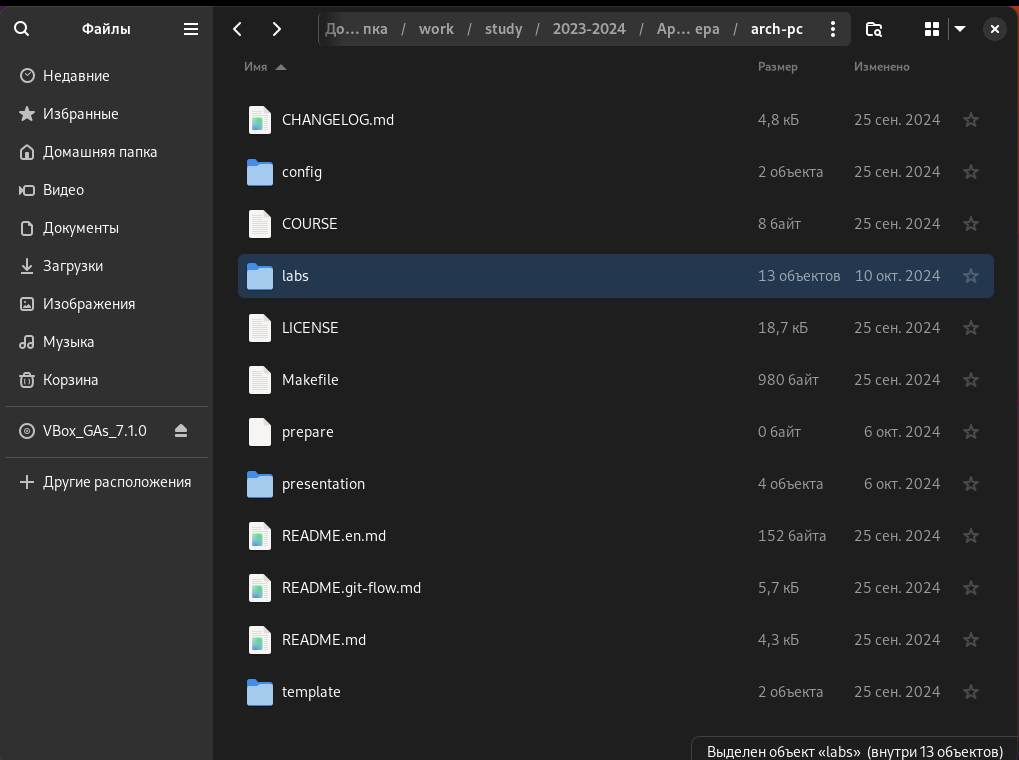


Рис. 13: Пункт 2.4.6

7.Выполняю пункт 2.5: Добавляю файлы с 1 и 2 лабораторной в папки labs/lab01, labs/lab02 в рабочее пространство git и отправляю их на сервер (рис. 14).

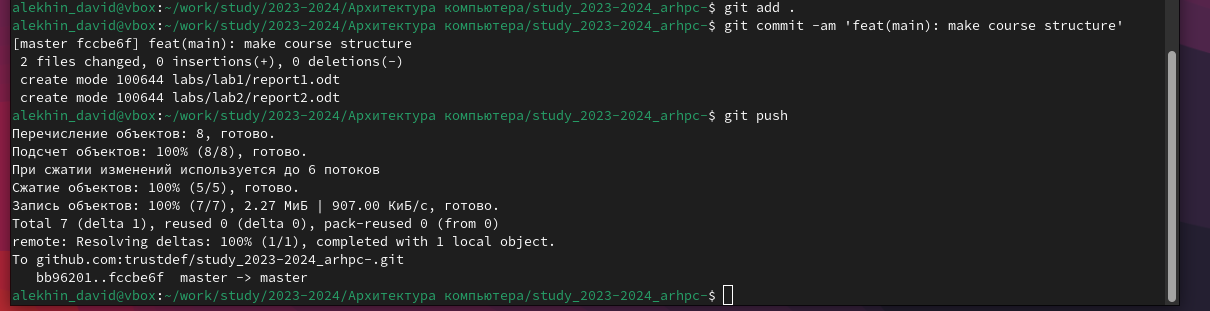


Рис. 14: Пункт 2.5

# 5 Выводы

Проведя лабораторную работу я научился работать с системой git, создал в этой системе рабочее пространство и настроил его. А также изучил идеологию и применение средств контроля версий.

# Список литературы

::: Файл с лабораторной работой 2 и прилагающиеся к нему материалы: 1. GDB: The GNU Project Debugger. — URL: https://www.gnu.org/software/gdb/. 2. GNU Bash Manual. — 2016. — URL: https://www.gnu.org/software/bash/manual/. 3. Midnight Commander Development Center. — 2021. — URL: https://midnight-commander. org/. 4. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/. 5. Newham C. Learning the bash Shell: Unix Shell Programming. — O’Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: http://www.amazon.com/Learning- bash-Shell-Programming-Nutshell/dp/0596009658. 6. Robbins A. Bash Pocket Reference. — O’Reilly Media, 2016. — 156 с. — ISBN 978-1491941591. :::