ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

Дисциплина: Архитектура компьютера

Обрезкова Анастасия Владимировна

Содержание

# 1 Цель работы

Изучить идеологию и применение средств контроля версий. Приобрести практические навыки по работе с системой git.

# 2 Задание

Изучить идеологию и применение системы git.

# 3 Теоретическое введение

## 3.1 Системы контроля версий. Общие понятия

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево про- екта 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 — со- хранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Системы контроля версий поддерживают возможность отслеживания и разре- шения 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. В зависи- мости 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 функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изме- нений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, ко- гда 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 цен- тральный репозиторий не является обязательным.

Среди классических VCS наиболее известны CVS, Subversion, а среди распре- делённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

## 3.2 Система контроля версий Git

Система контроля версий Git представляет собой набор программ команд- ной 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 копировани- ем или архивацией.

## 3.3 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 необходимо 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 прави- лам 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

## 4.1 Настройка github

1. Создала учетную запись на сайте Github, заполнила основные данные. (рис. 1)

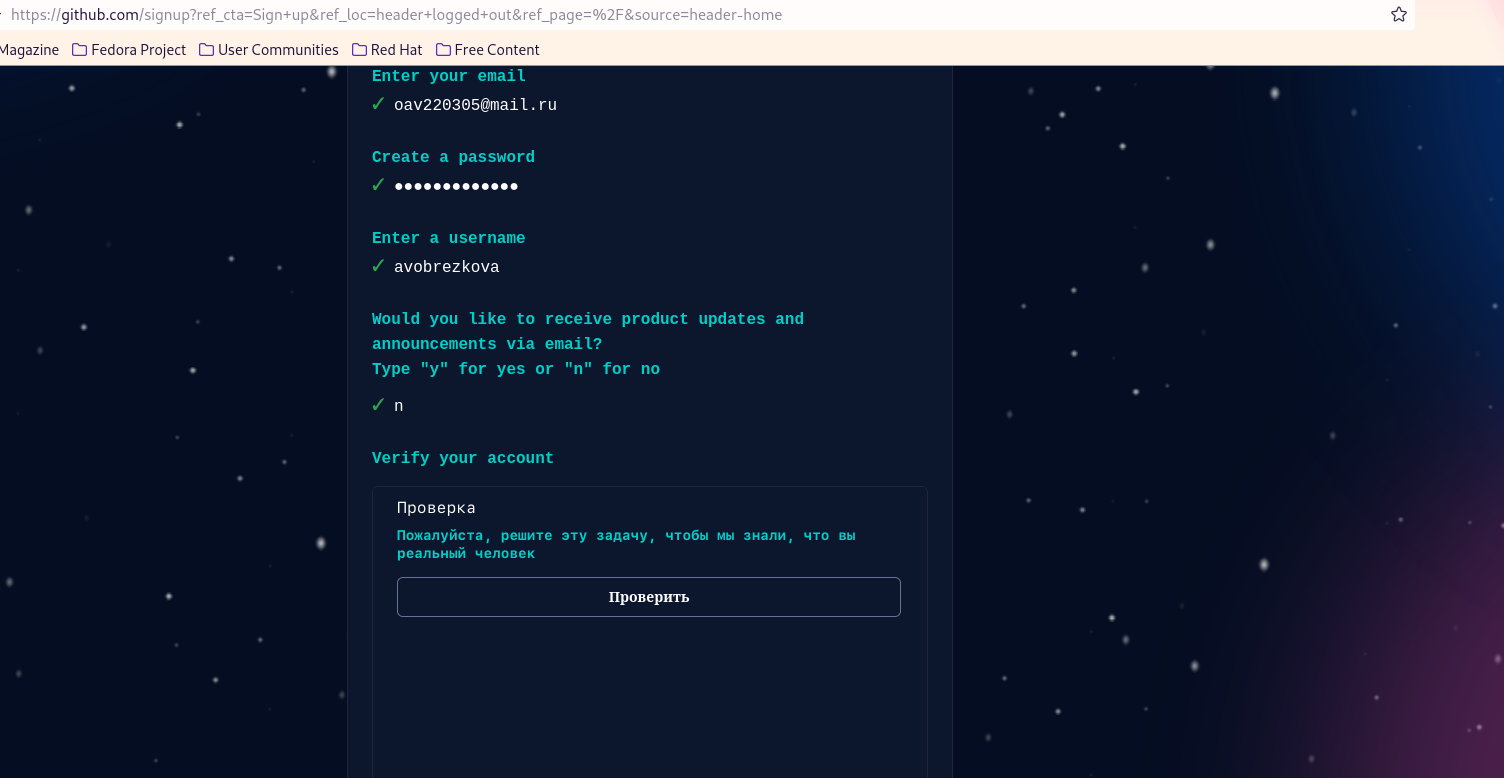


Рис. 1: Создание учетной записи

## 4.2 Базовая настройка git.

1.Создала предварительную конфигурацию git. (рис. 2)

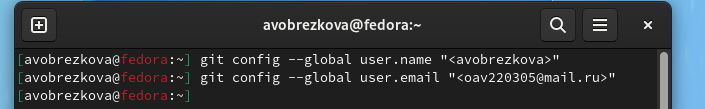


Рис. 2: Предварительная конфигурация git

1. Настроила utf-8 в выводе сообщений git. (рис. 3)

Рис. 3: Настройка utf-8

Рис. 3: Настройка utf-8

1. Задала имя начальной ветки. (рис. 4)

Рис. 4: Начальная строка

Рис. 4: Начальная строка

1. Параметр autocrlf. (рис. 5)

Рис. 5: Параметр autocrlf

Рис. 5: Параметр autocrlf

1. Параметр safecrlf. (рис. 6)

Рис. 6: Параметр safecrlf

Рис. 6: Параметр safecrlf

## 4.3 Создание SSH ключа.

1. Генерирование ключей. (рис. 7, рис. 8)

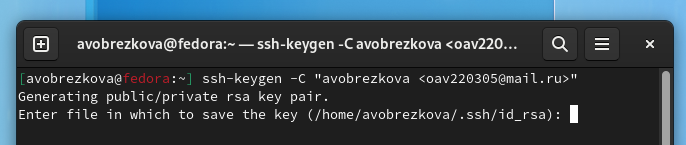


Рис. 7: Генерирование ключей

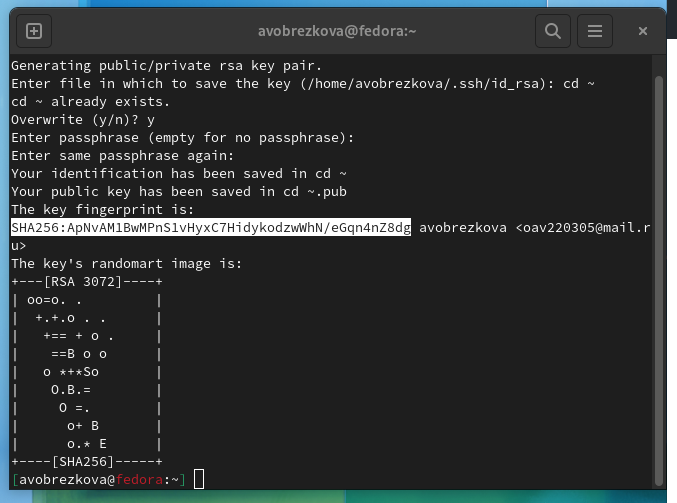


Рис. 8: Генерирование ключей

1. С помощью команды cat скопировала из локальной консоли ключ в буфер обмена и вставила его в появившееся на сайте поле. Указала для ключа имя и создала его. (рис. 9)

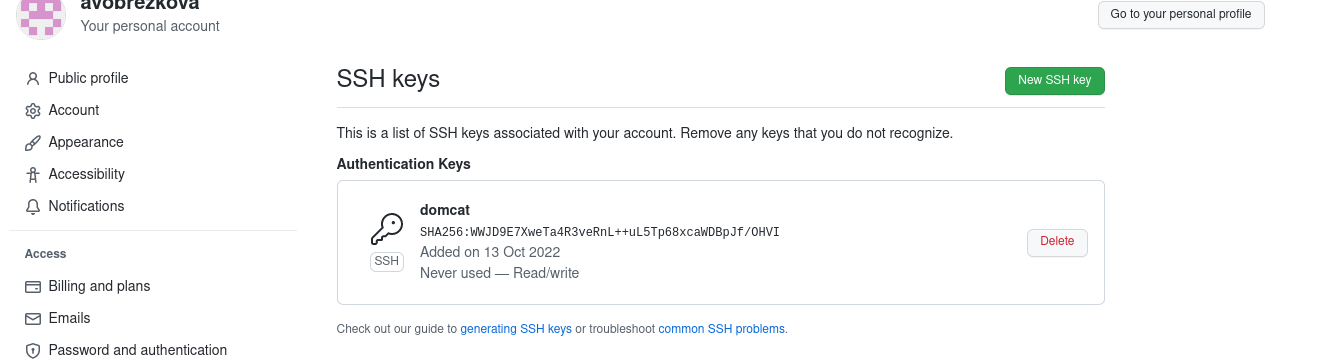


Рис. 9: Создание ключа

## 4.4 Создание рабочего пространства и репозитория курса на основе шаблона.

1. Создала католог для предмета “Архитектура компьютера”. (рис. 10)

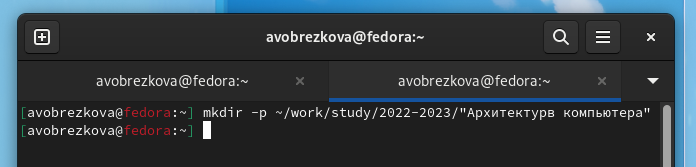


Рис. 10: Создание каталога для предмета

## 4.5 Создание репозитория курса на основе шаблона.

1. Перешла на страницу репозитория с шаблоном курса, задала имя репозитория и создала его. (рис. 11)

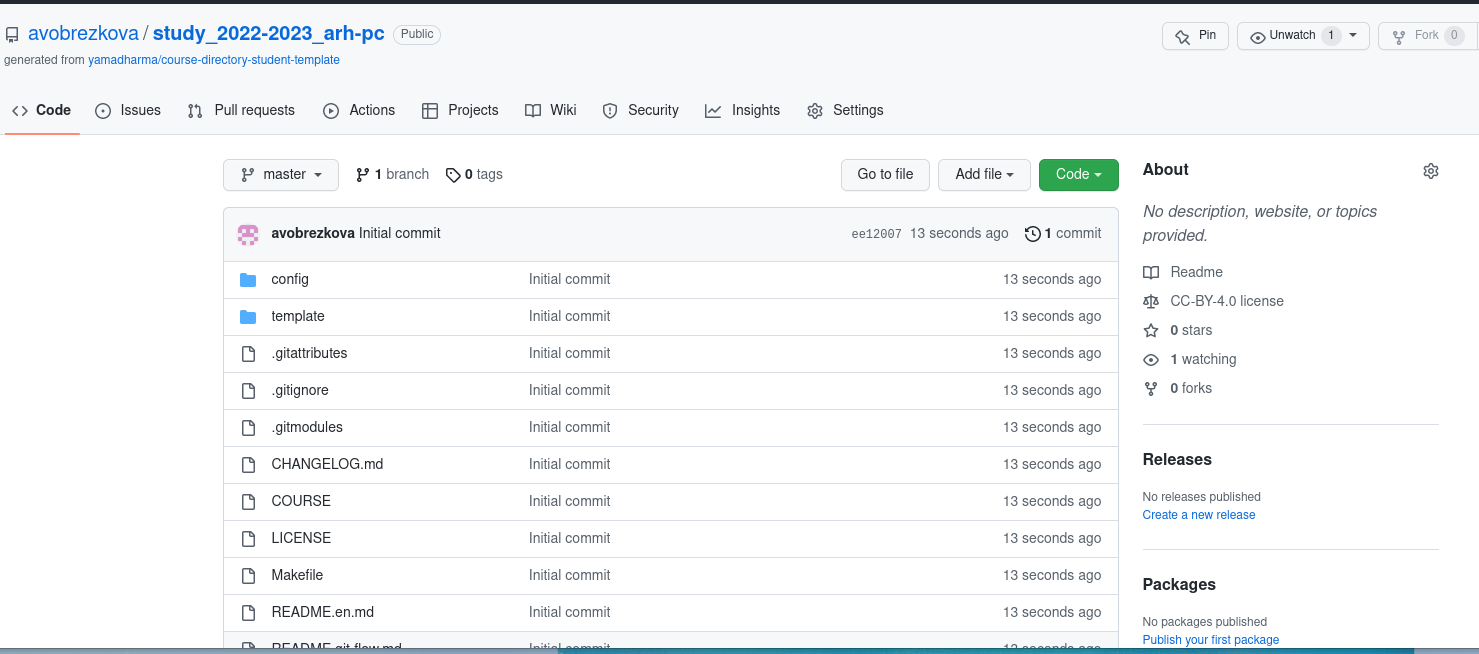


Рис. 11: Страница репозитория

1. Перешла в каталог курса. (рис. 12)

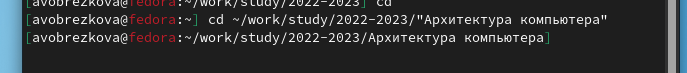


Рис. 12: Переход в каталог курса

1. Клонировала созданный репозиторий. (рис. 13, рис. 14)

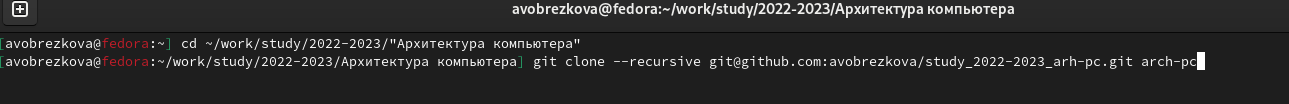


Рис. 13: Клонирование репозитория

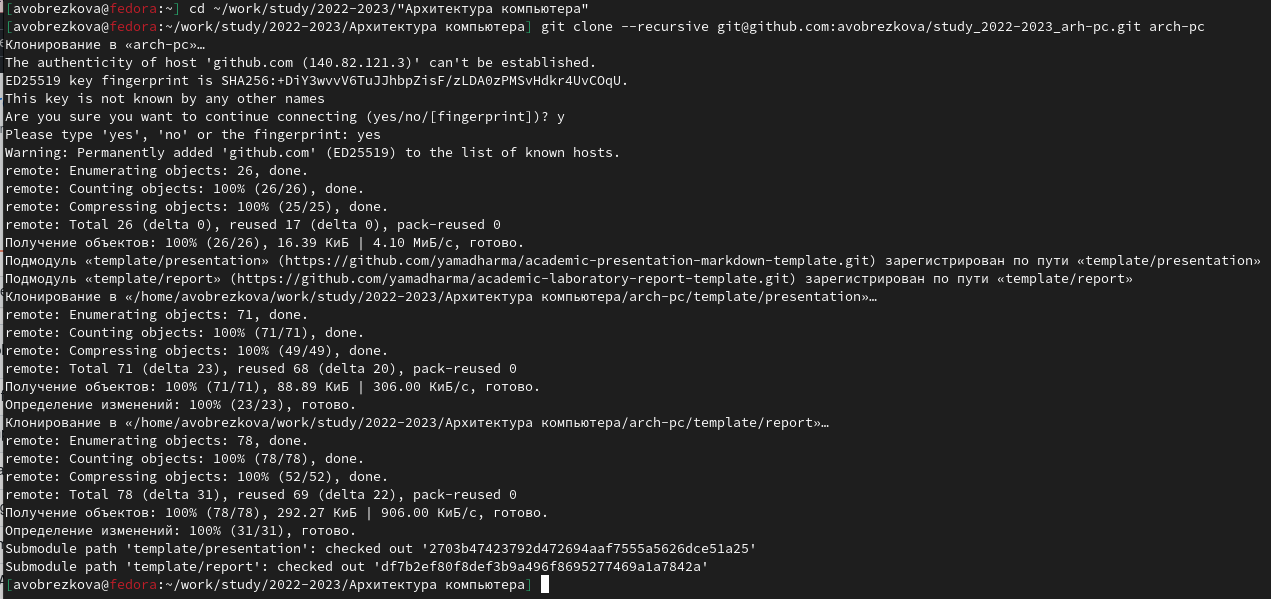


Рис. 14: Результат клонирования

## 4.6 Настройка каталога курса.

1. Перешла в каталог курса и удалила лишние файлы. (рис. 15)

Рис. 15: Удаление лишних файлов

Рис. 15: Удаление лишних файлов

1. Создала нобходимые каталоги. (рис. 16)

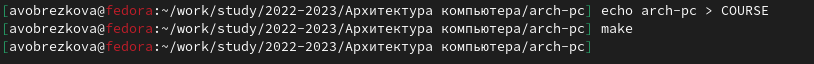


Рис. 16: Создание каталога

1. Отправила файлы на сервис. (рис. 17), рис. 18, рис. 19)

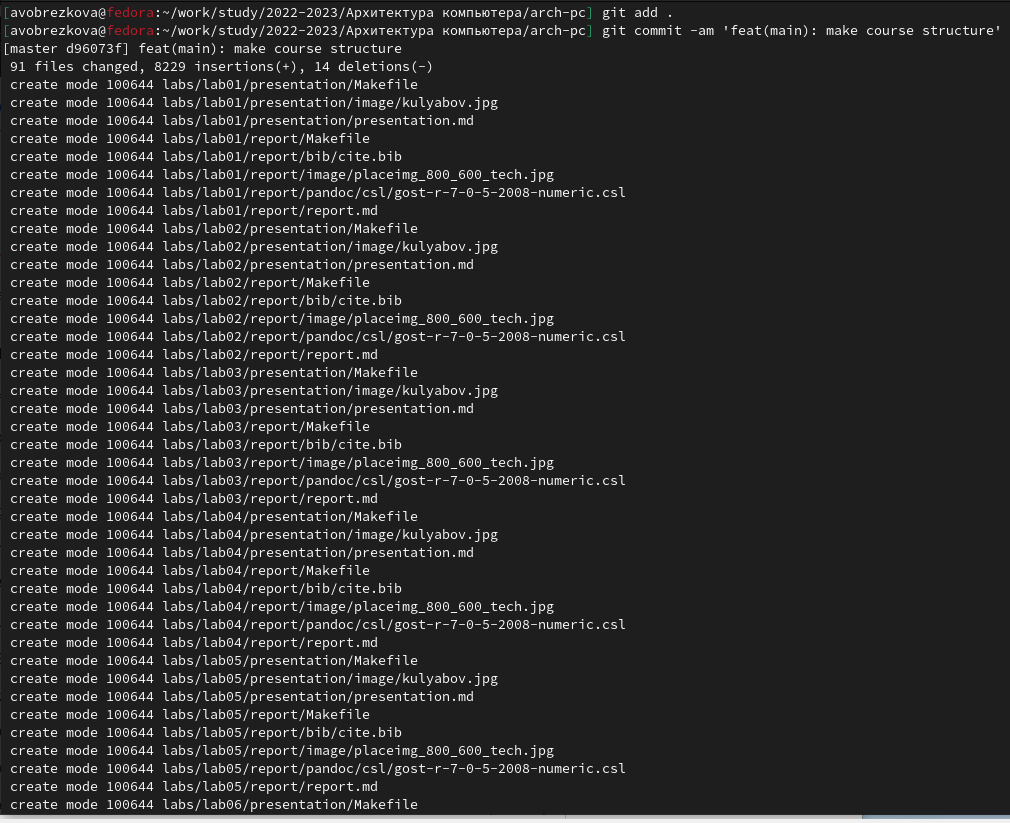


Рис. 17: Отправка файлов

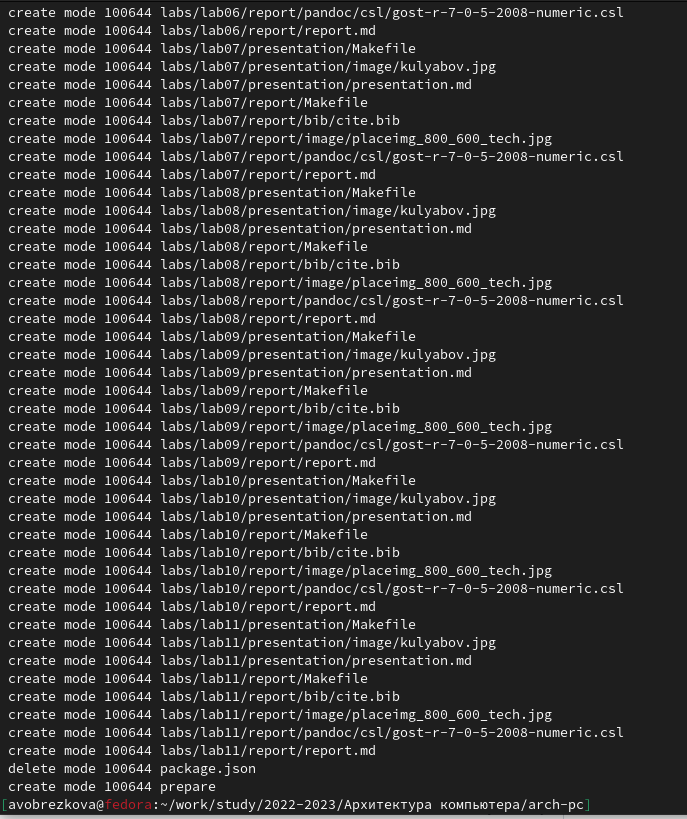


Рис. 18: Отправка файлов

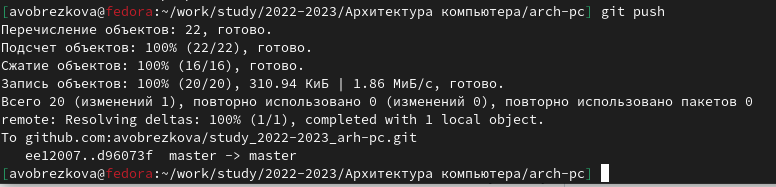


Рис. 19: Отправка файлов

# 5 Задания для самостоятельной работы.

1. Создала отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства. (рис. 20, рис. 21)

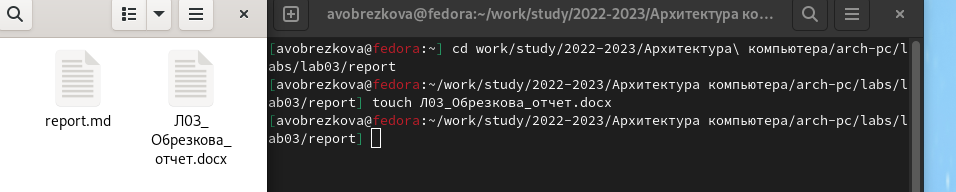


Рис. 20: Создание файло отчета

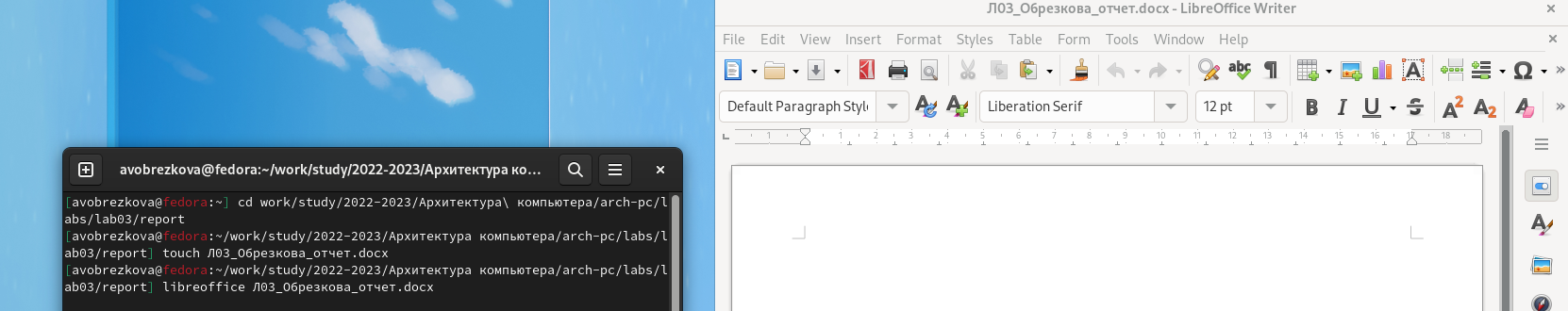


Рис. 21: Начало создания отчета

1. Скопировала отчеты по выполнению предыдущих лабораторных работ в соответвующие каталоги. Отчеты выполнялись на виртуальной машине, поэтому я просто перенесла их из одного каталога в другой. (рис. 22, рис. 23)

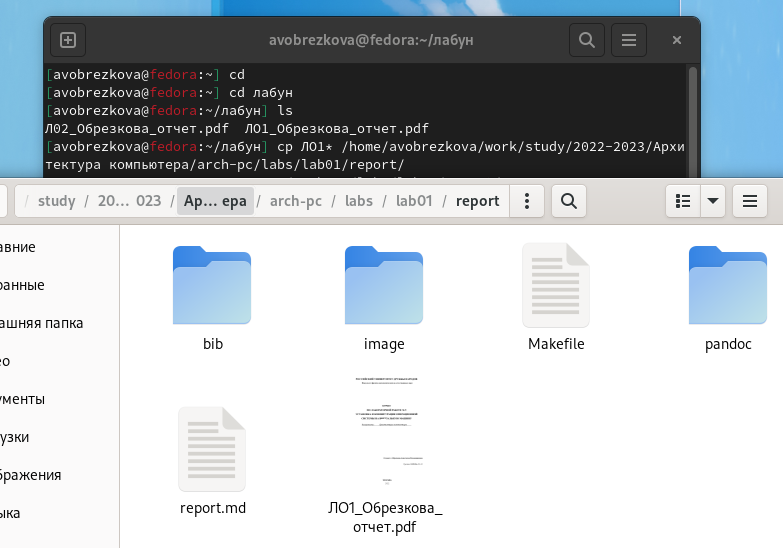


Рис. 22: Перенос лабораторной работы №1

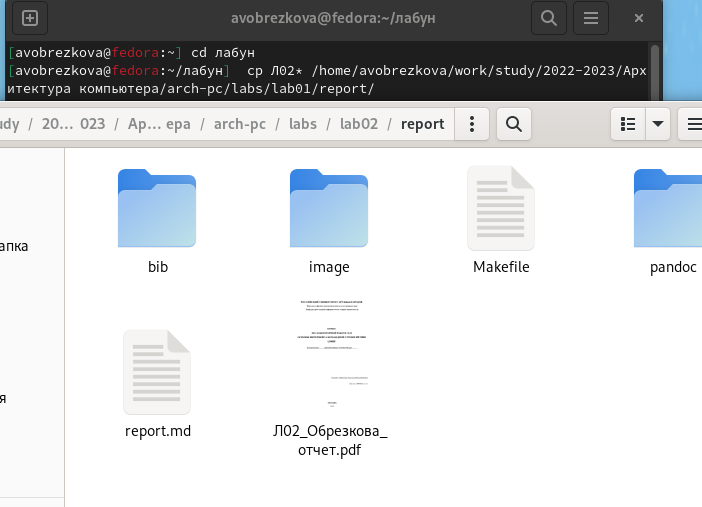


Рис. 23: Перенос лабораторной работы №2

1. Загрузка файлов на github. (рис. 24, рис. 25)

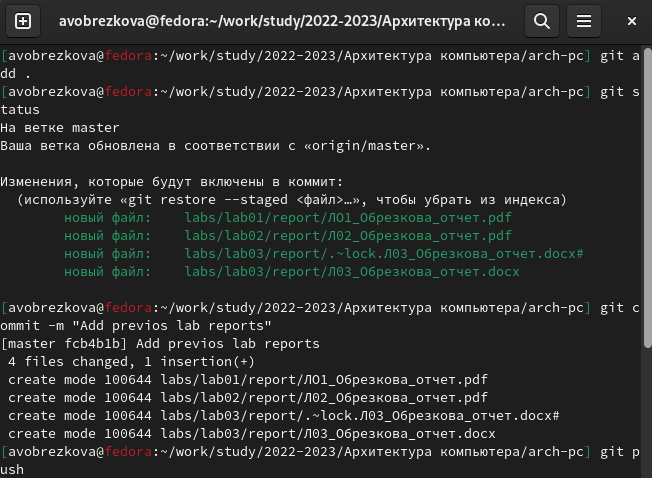


Рис. 24: Добавленные изменения

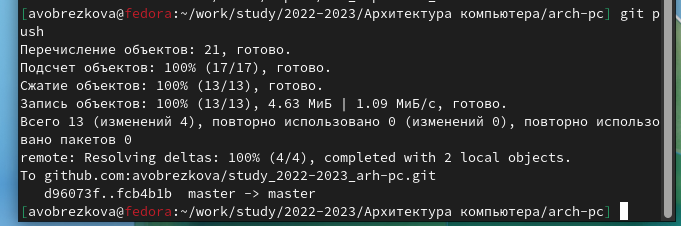


Рис. 25: Запустим изменения

Данные изменения можно проверить по ссылке:

<https://github.com/avobrezkova/study_2022-2023_arh-pc/tree/master/labs>

# 6 Выводы

Я ознакомилась с системой версий GIT. Создала репозиторий из шаблона сервиса GitHub и внесла в него необходимые изменения. Я освоила базовые команды утилита git.

# Список литературы

1. <https://esystem.rudn.ru/user/policy.php>
2. <https://github.com/yamadharma/academic-laboratory-report-template/tree/master/report>