Отчёт по лабораторной работе №3

Архитектура компьютера

Рогожина Надежда Александровна

Содержание

# 1 Цель работы

Целью работы является изучить идеологию и применения средств контроля версий, а также приобрести практические навыки по работе с системой git.

# 2 Задание

1. Создайте отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства (labs>lab03>report)
2. Скопируйте отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданного рабочего пространства.
3. Загрузите файлы на github.

# 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 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. 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. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. 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.

## 3.2 Система контроля версий Git

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.

## 3.3 Основные команды git

В табл. **¿tbl:std-dir?** приведено краткое описание основный команд git.

Основные команды git { #tbl:std-dir }

| Команда | Краткое описание |
| --- | --- |
| git unit | Создание основного дерева репозитория |
| git push | Отправка всех произведённых изменений локального дерева в центральный репозиторий |
| git status | Просмотрр списка изменённых файлов в текущей директории |
| git add | Добавить все изменённые и/или созданные файлы и/или каталоги |
| git rm имена\_файлов | Удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории) |
| git checkout -b имя\_ветки | Создание новой ветки, базирующейся на текущей |
| git merge --no-ff имя\_ветки | Слияние ветки с текущим деревом |
| git push origin:имя\_ветки | Удаление ветки с центрального репозитория |

## 3.4 Стардартные 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 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

Первым шагом создадим аккаунт на github.com: (рис. 1)

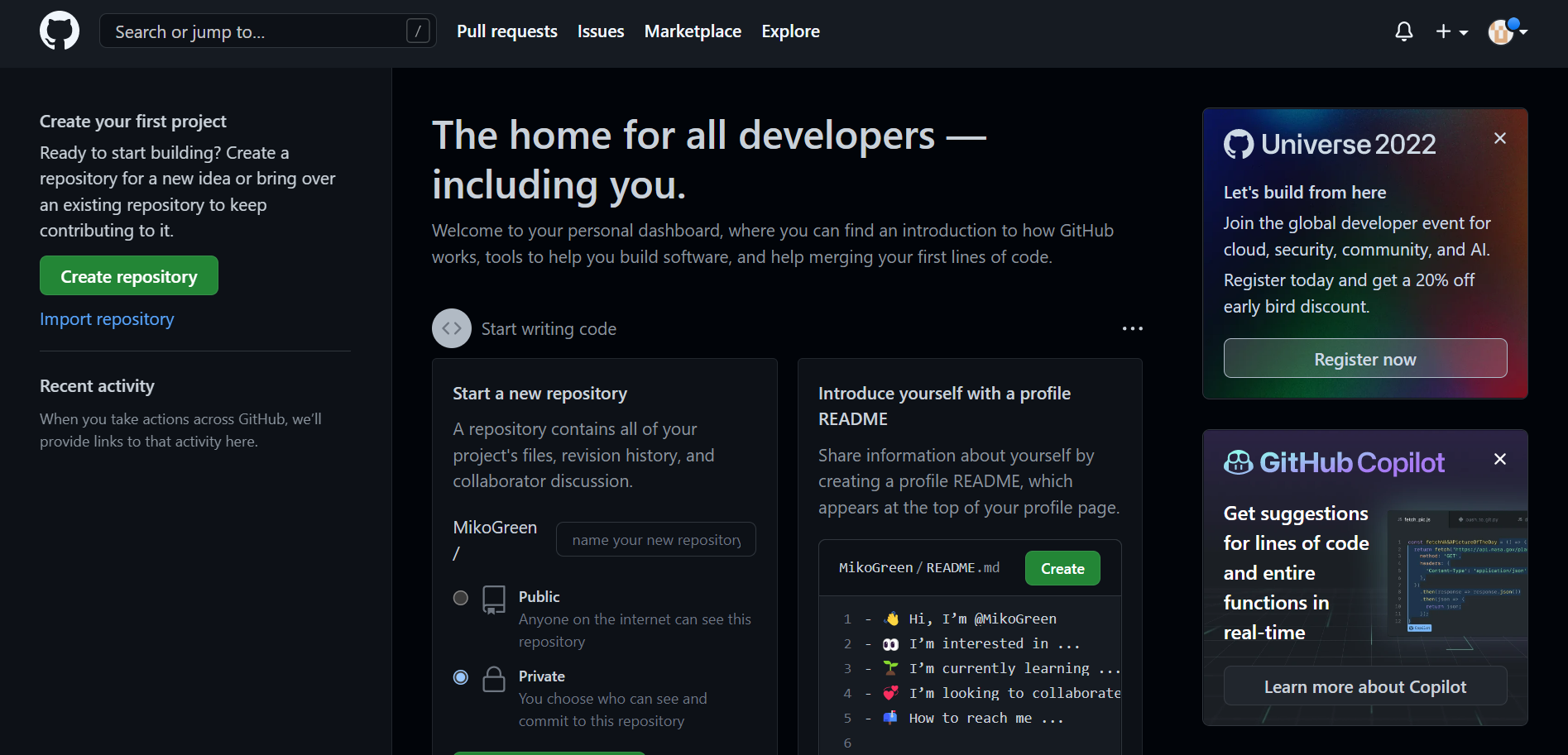
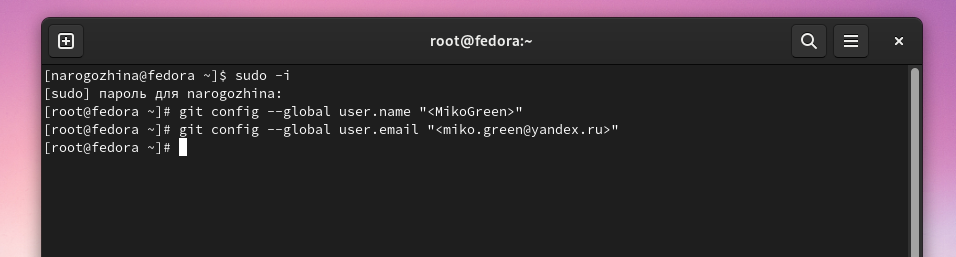
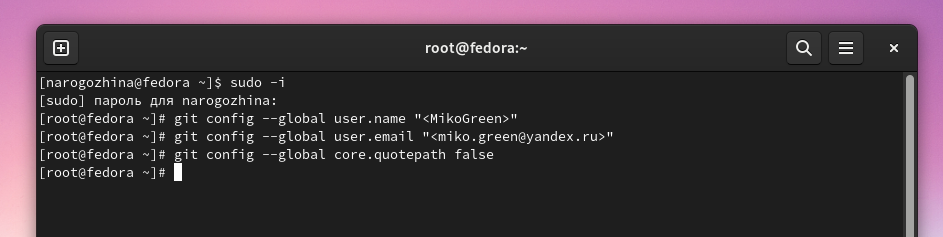
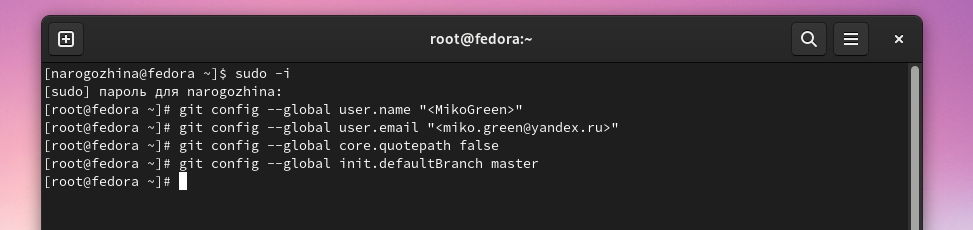


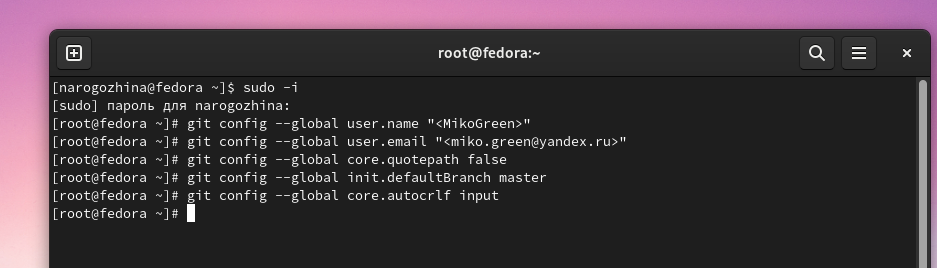
Рис. 1: Создание аккаунта на github.com

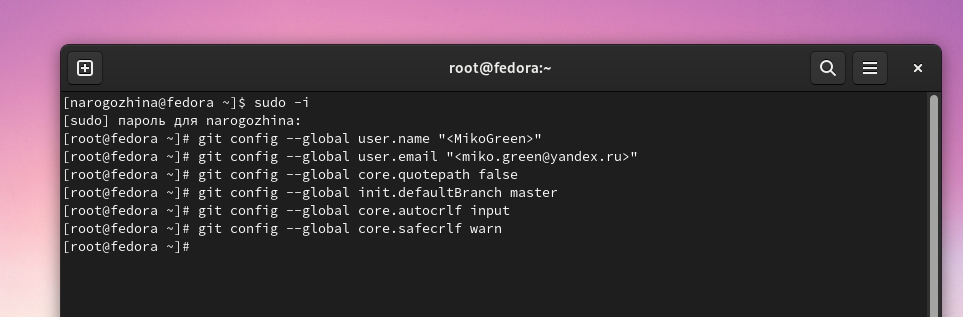
Прежде всего проведем базовую настройку git:

 { #fig:002 width=70% }

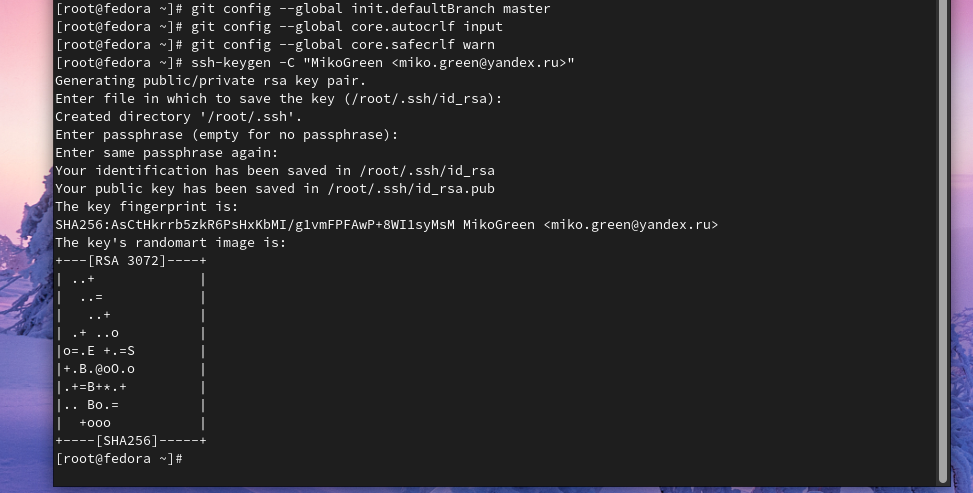
 { #fig:003 width=70% }

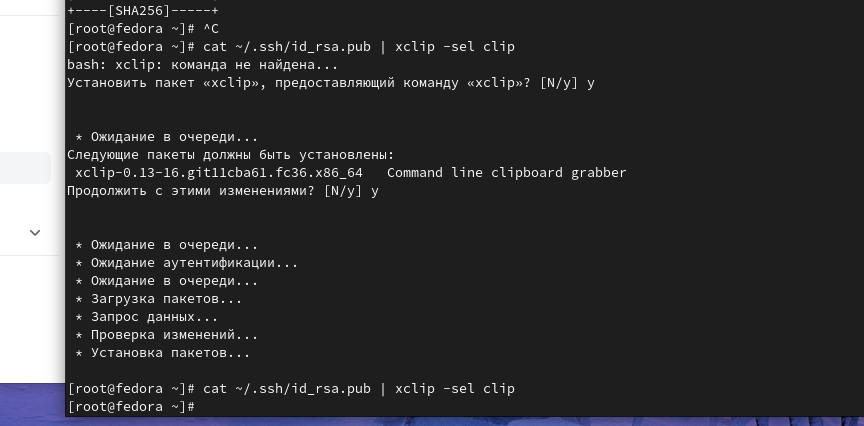
 { #fig:004 width=70% }

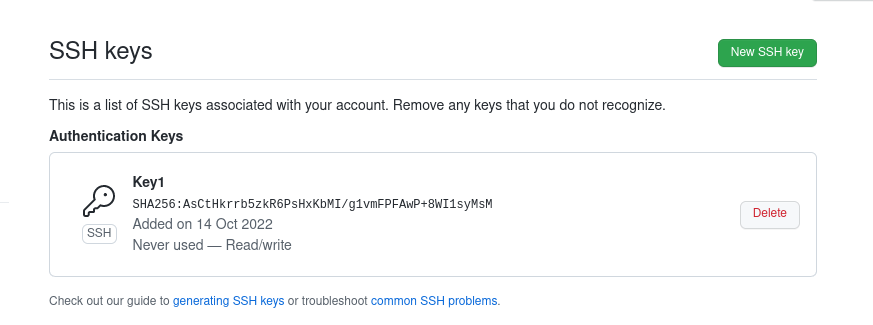
 { #fig:005 width=70% }

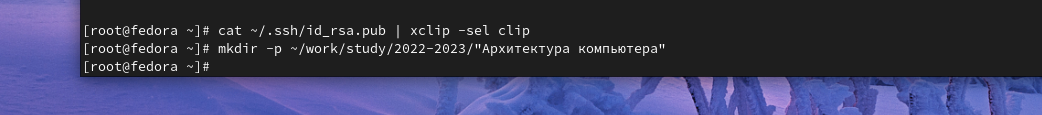
 { #fig:006 width=70% }

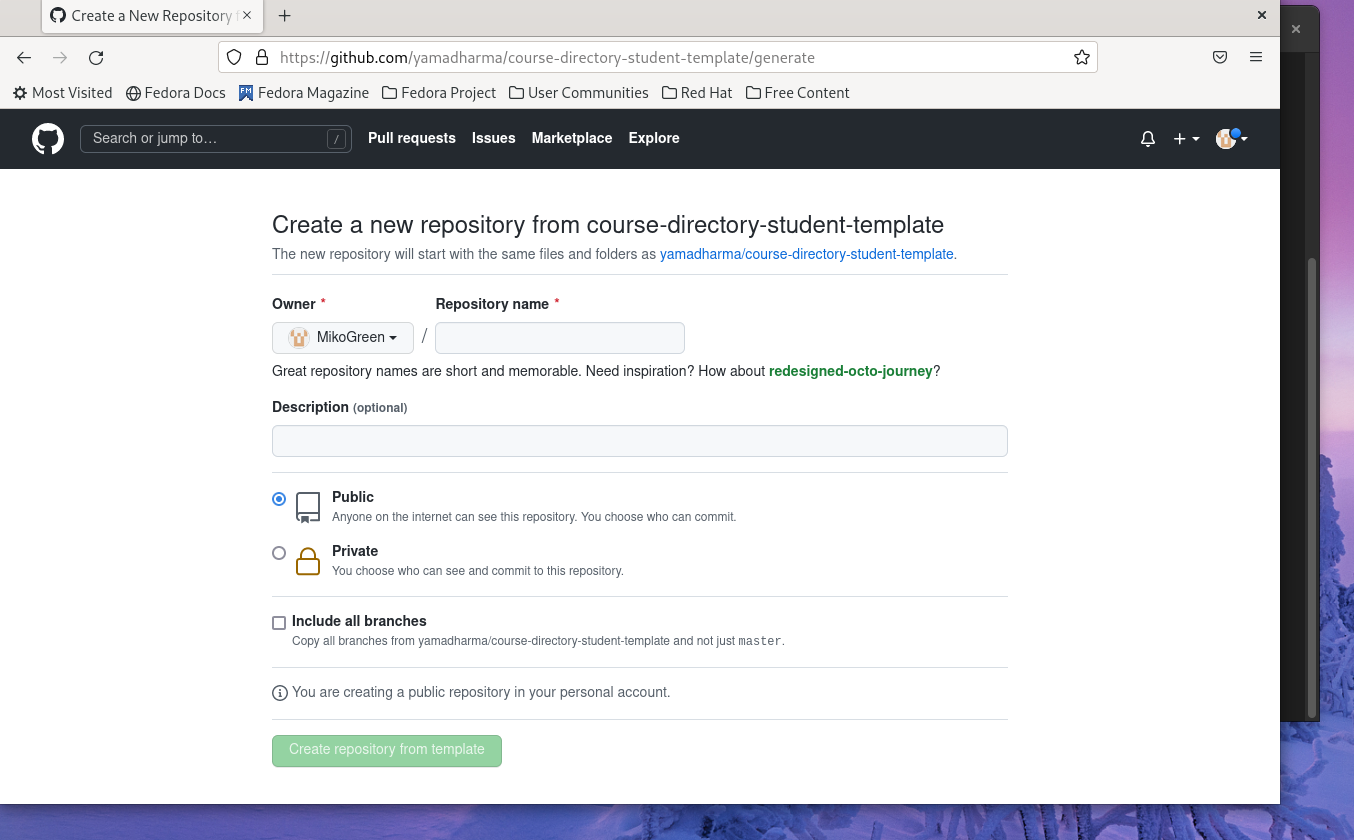
Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый):

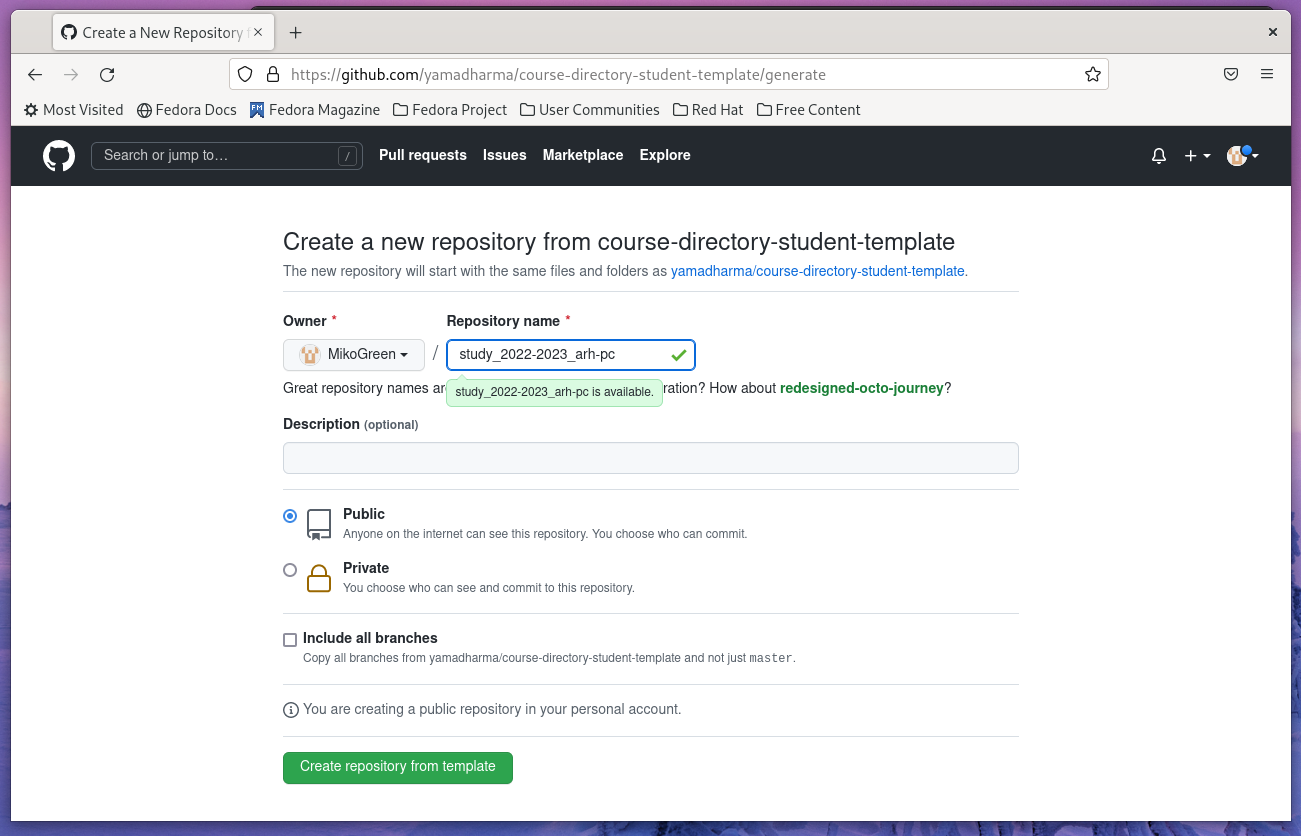
 { #fig:007 width=70% }

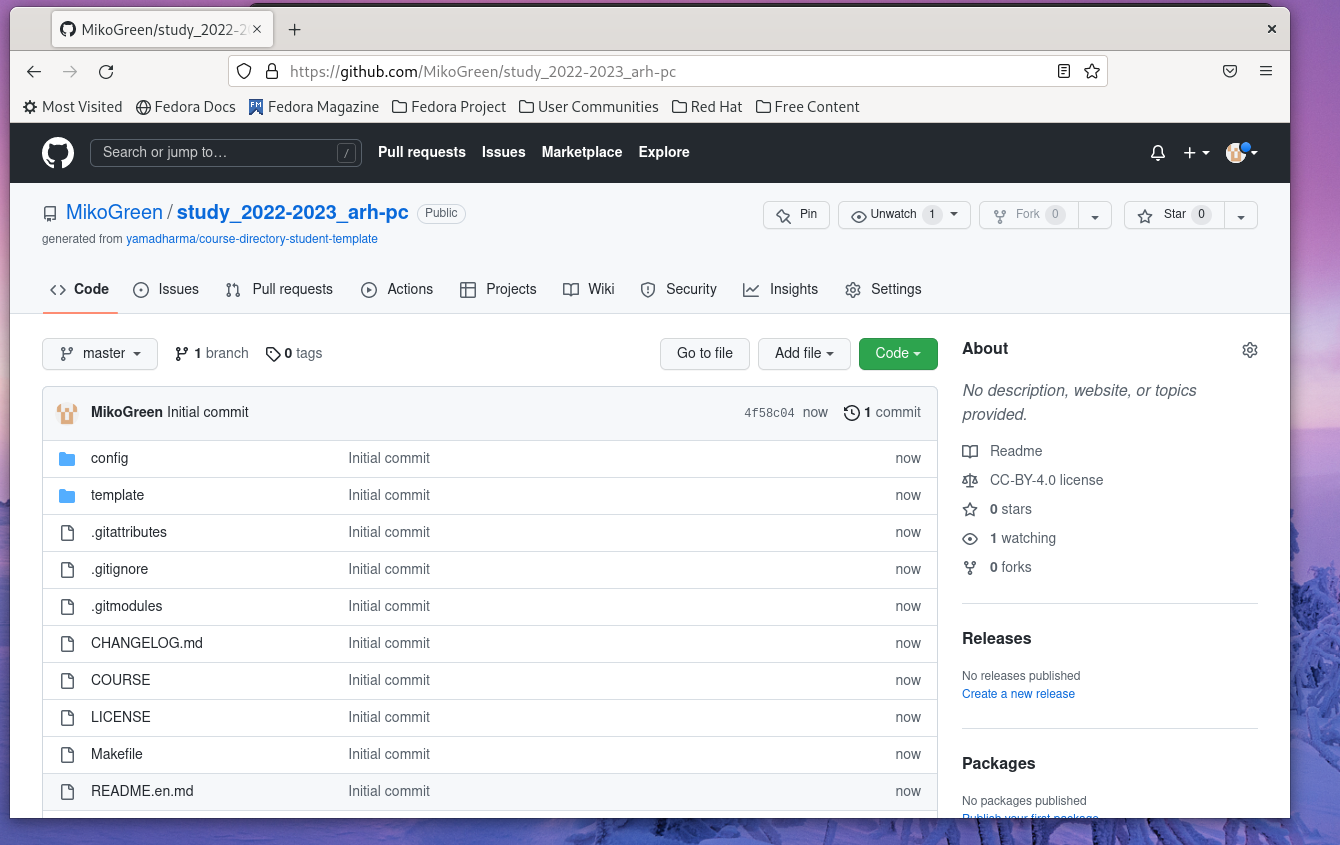
 { #fig:008 width=70% }

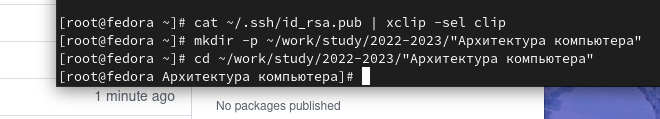
 { #fig:009 width=70% }

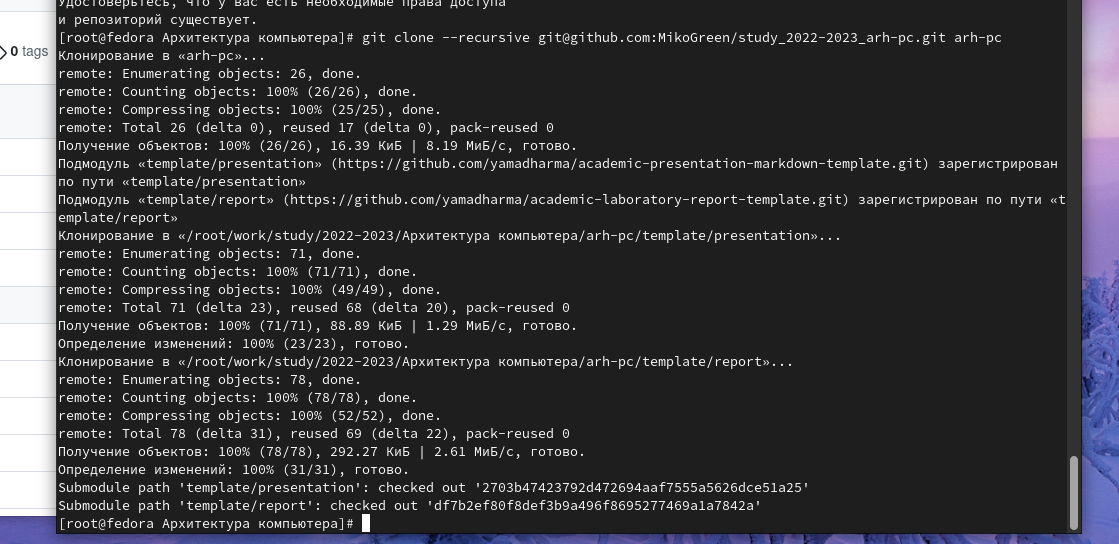
 { #fig:010 width=70% }

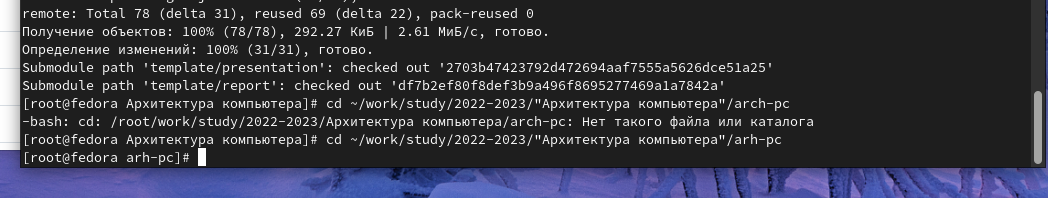
 { #fig:011 width=70% }

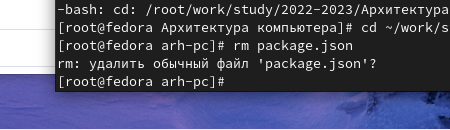
 { #fig:012 width=70% }

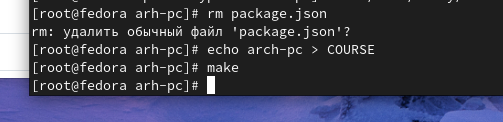
 { #fig:013 width=70% }

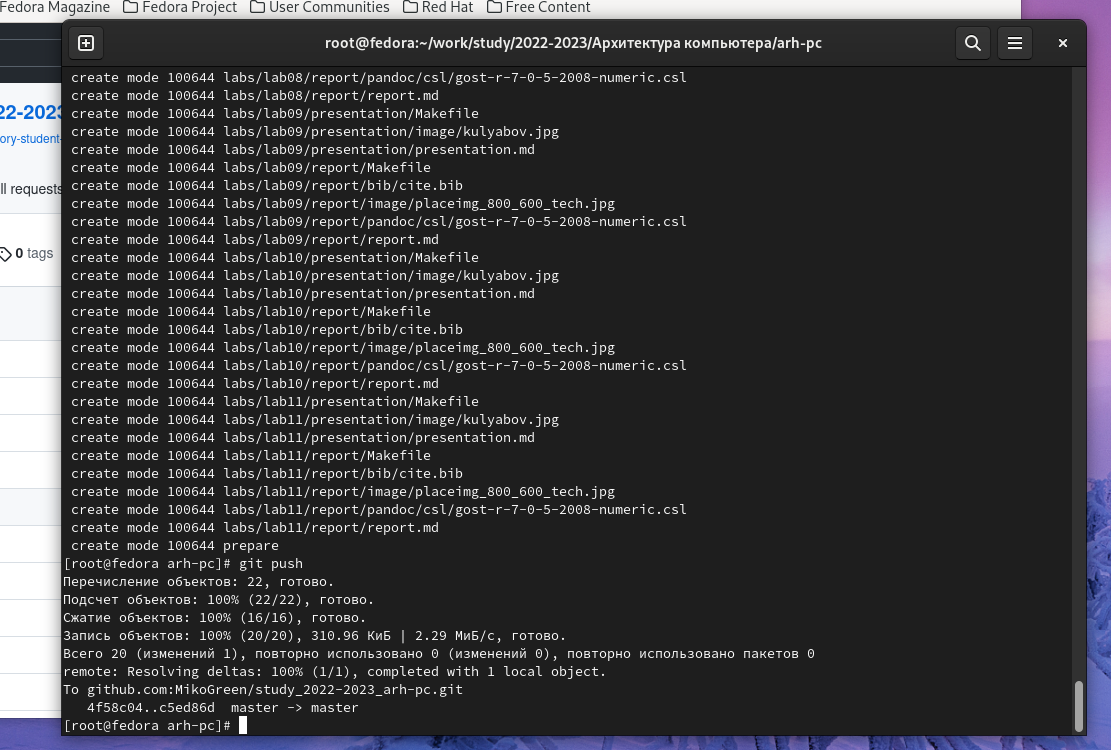
 { #fig:014 width=70% }

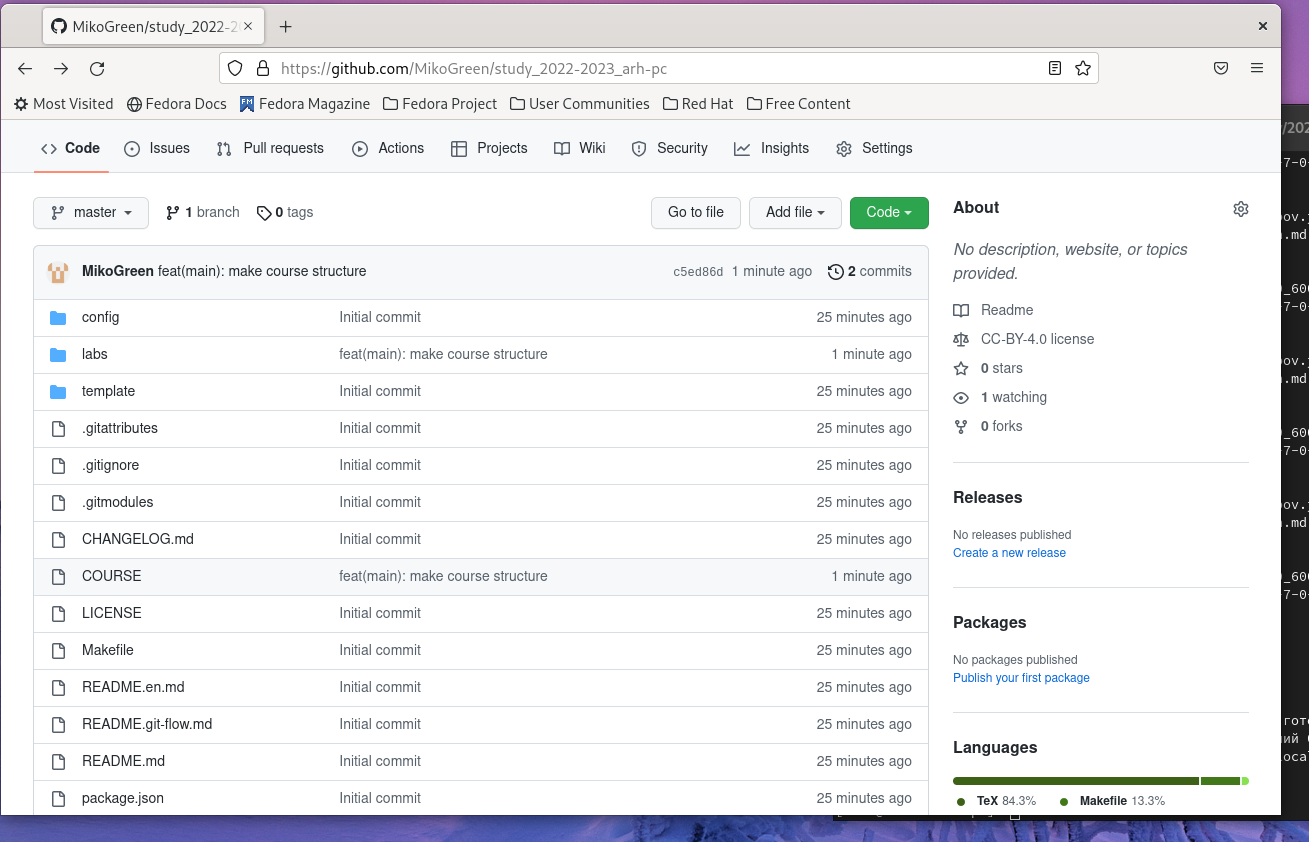
 { #fig:015 width=70% }

 { #fig:016 width=70% }

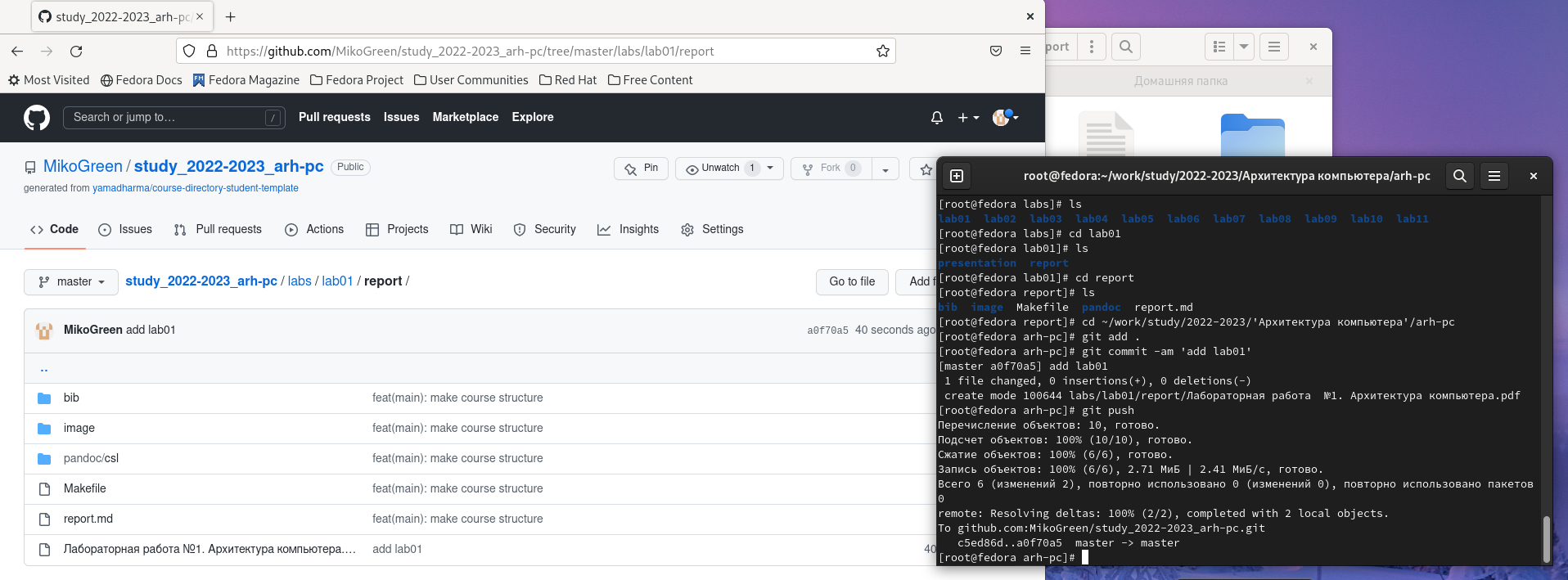
 { #fig:017 width=70% }

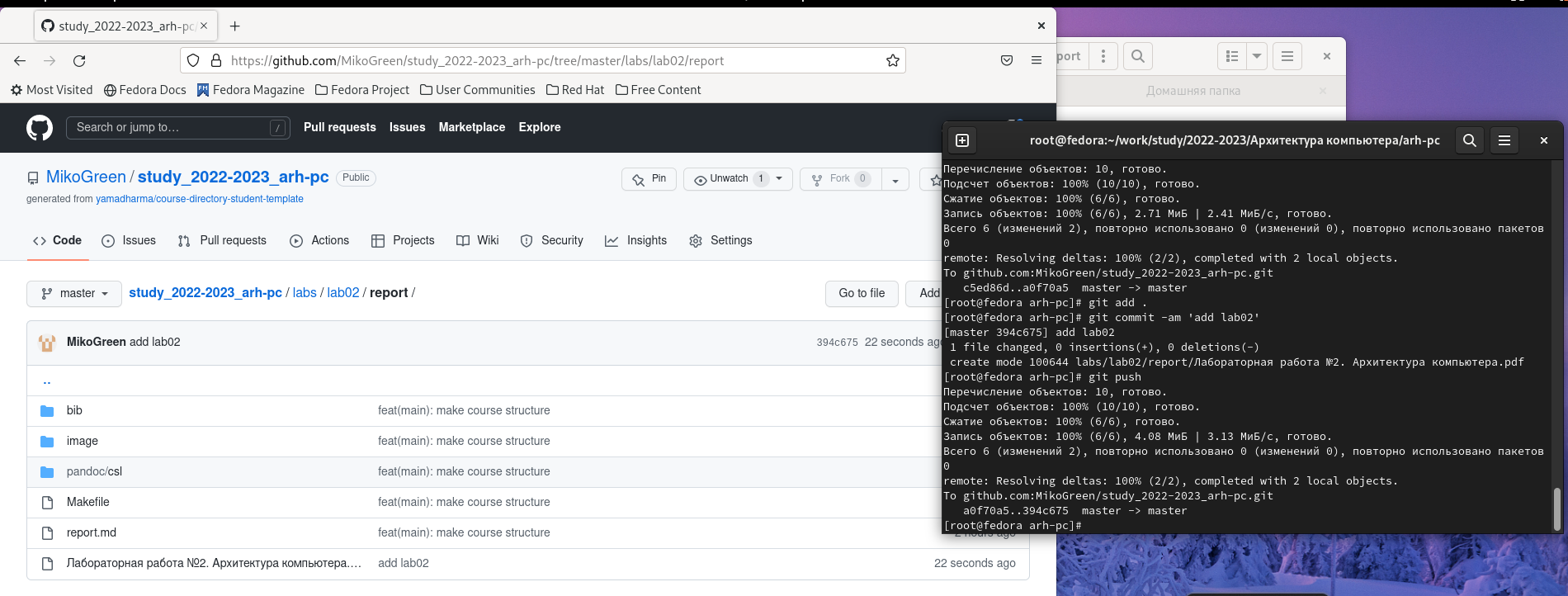
 { #fig:018 width=70% }

 { #fig:019 width=70% }

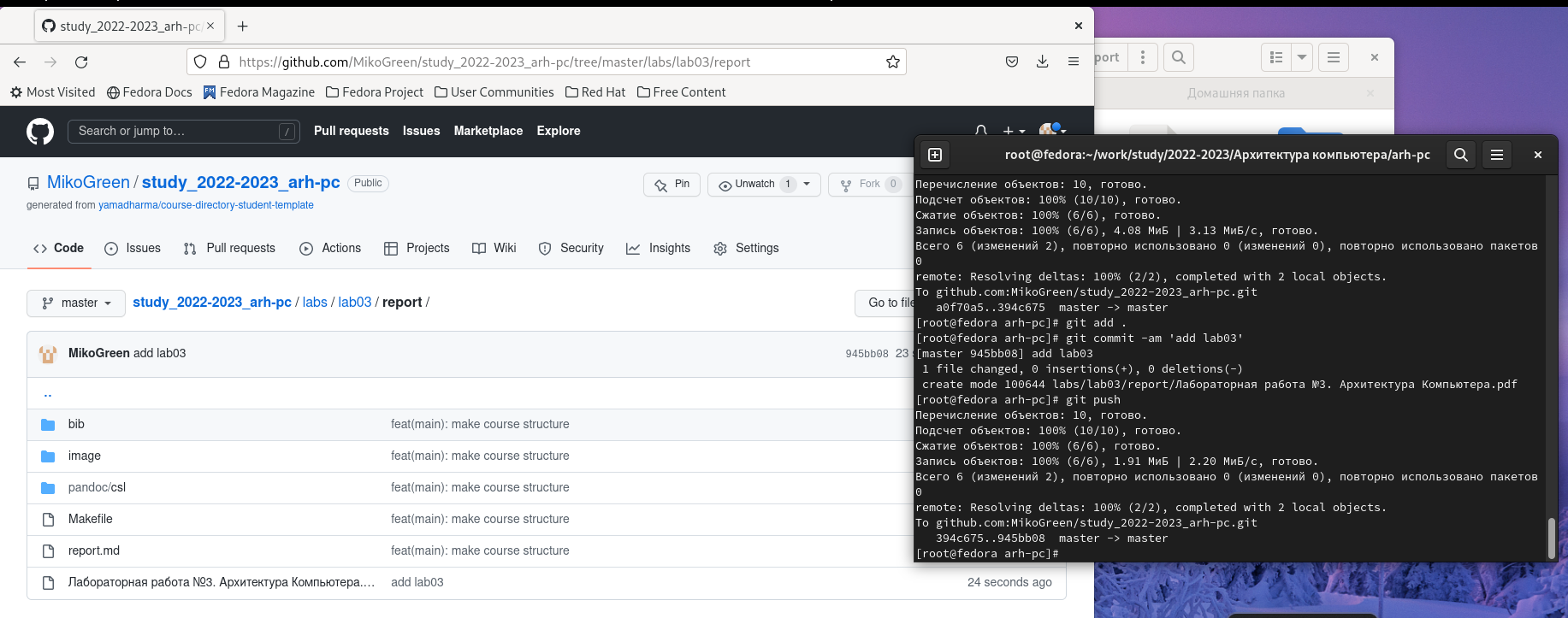
 { #fig:020 width=70% }

Подгружаем предыдущие лабораторные работы

 { #fig:021 width=70% }

 { #fig:022 width=70% }

А также подгружаем данную работу:

 { #fig:023 width=70% }

# 5 Выводы

В процессе выполнения работы мне удалось изучить идеологию и применение средств контроля версий, а также приобрести навыки работы с системой git.