

Отчёт по лабораторной работе №3

Архитектура компьютера

Рогожина Надежда Александровна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
3.1	Системы контроля версий. Общие понятия	7
3.2	Система контроля версий Git	8
3.3	Основные команды git	8
3.4	Стандартные процедуры работы при наличии центрального репозитория	9
4	Выполнение лабораторной работы	11
5	Выводы	19

Список иллюстраций

4.1	Создание аккаунта на github.com	11
4.2	Сделаем предварительную конфигурацию	11
4.3	Настроим utf-8 в выводе сообщений git	11
4.4	Зададим имя начальной ветки(будем называть её master):	12
4.5	Параметр autocrlf	12
4.6	Параметр safecrlf	12
4.7	Генерация ключа SSH	12
4.8	Скопировав ключ из локальной консоли в буфер обмена:	13
4.9	Вставляем в специальное поле на сайте github.com и указываем имя ключа	13
4.10	Создадим каталог для предмета “Архитектура компьютера” . . .	13
4.11	Переходим по ссылке на страницу репозитория с шаблоном курса	14
4.12	Зададим имя репозитория и создадим его	14
4.13	Созданный репозиторий	15
4.14	Перейдем из терминала в каталог курса	15
4.15	Клонируем созданный репозиторий	15
4.16	Переходим в каталог курса	15
4.17	Удаляем лишние файлы	16
4.18	Создаем необходимые каталоги	16
4.19	Отправляем файлы на сервер	16
4.20	Проверяем правильность создания иерархии рабочего пространства	17
4.21	Загрузка отчета по 1 лабораторной работе	17
4.22	Загрузка отчета по 2 лабораторной работе	17
4.23	Загрузка отчета по 3 лабораторной работе	18

Список таблиц

3.1	Основные команды git { #tbl:std-dir }	8
-----	---	---

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 таблице ниже приведено краткое описание основных команд git.

Таблица 3.1: Основные команды git { #tbl:std-dir }

Команда	Краткое описание
<code>git init</code>	Создание основного дерева репозитория
<code>git push</code>	Отправка всех произведённых изменений локального дерева в центральный репозиторий
<code>git status</code>	Просмотр списка изменённых файлов в текущей директории
<code>git add</code>	Добавить все изменённые и/или созданные файлы и/или каталоги
<code>git rm</code>	Удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории)

Команда	Краткое описание
<code>git checkout -b имя_ветки</code>	Создание новой ветки, базирующейся на текущей
<code>git merge --no-ff имя_ветки</code>	Слияние ветки с текущим деревом
<code>git push origin:имя_ветки</code>	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:

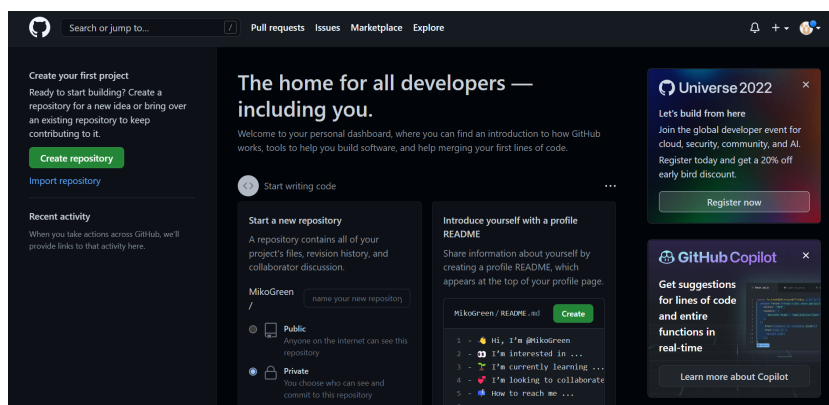


Рис. 4.1: Создание аккаунта на github.com

Прежде всего проведем базовую настройку git:

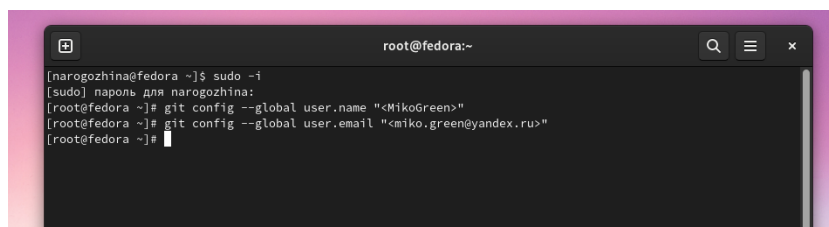


Рис. 4.2: Сделаем предварительную конфигурацию

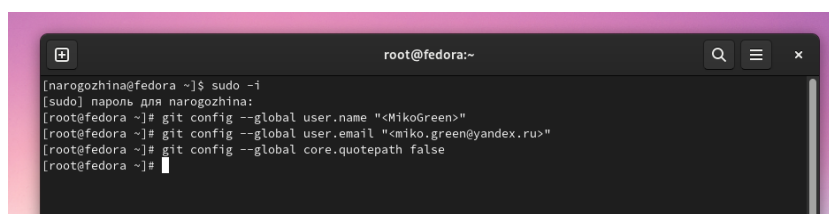
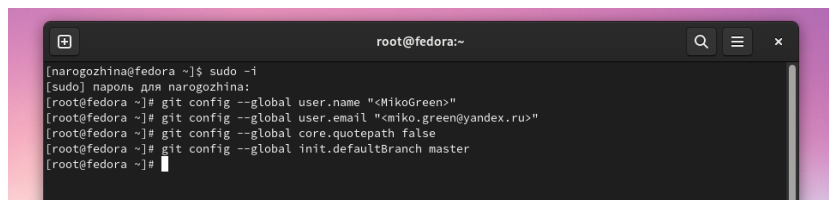
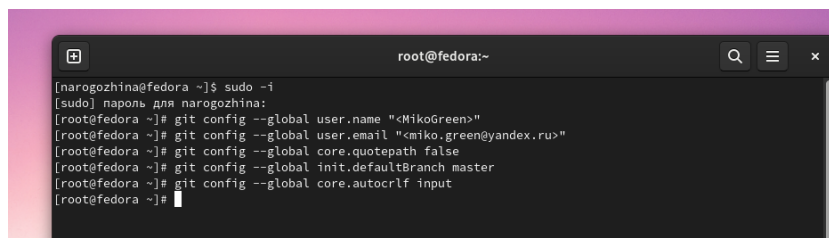


Рис. 4.3: Настроим utf-8 в выводе сообщений git



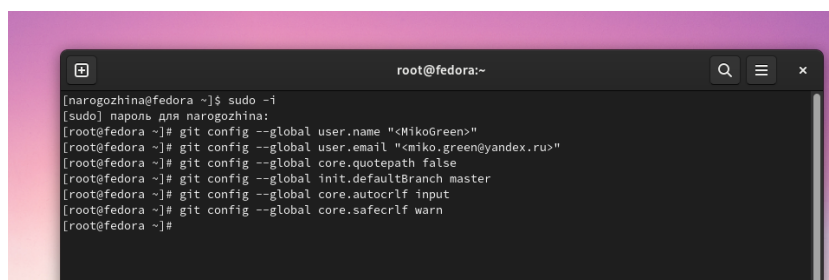
```
root@fedora:~  
[narogozhina@fedora ~]$ sudo -i  
[sudo] пароль для narogozhina:  
[root@fedora ~]# git config --global user.name "MikoGreen"  
[root@fedora ~]# git config --global user.email "miko.green@yandex.ru"  
[root@fedora ~]# git config --global core.quotepath false  
[root@fedora ~]# git config --global init.defaultBranch master  
[root@fedora ~]#
```

Рис. 4.4: Зададим имя начальной ветки(будем называть её master):



```
root@fedora:~  
[narogozhina@fedora ~]$ sudo -i  
[sudo] пароль для narogozhina:  
[root@fedora ~]# git config --global user.name "MikoGreen"  
[root@fedora ~]# git config --global user.email "miko.green@yandex.ru"  
[root@fedora ~]# git config --global core.quotepath false  
[root@fedora ~]# git config --global init.defaultBranch master  
[root@fedora ~]# git config --global core.autocrlf input  
[root@fedora ~]#
```

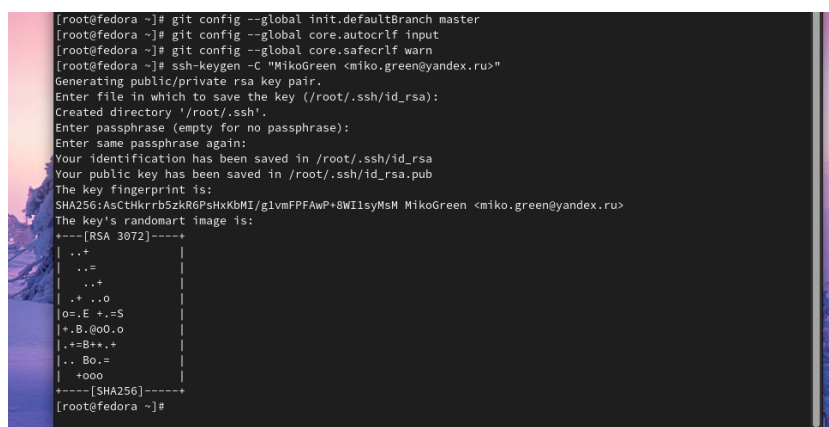
Рис. 4.5: Параметр autocrlf



```
root@fedora:~  
[narogozhina@fedora ~]$ sudo -i  
[sudo] пароль для narogozhina:  
[root@fedora ~]# git config --global user.name "MikoGreen"  
[root@fedora ~]# git config --global user.email "miko.green@yandex.ru"  
[root@fedora ~]# git config --global core.quotepath false  
[root@fedora ~]# git config --global init.defaultBranch master  
[root@fedora ~]# git config --global core.autocrlf input  
[root@fedora ~]# git config --global core.safecrlf warn  
[root@fedora ~]#
```

Рис. 4.6: Параметр safecrlf

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



```
[root@fedora ~]# git config --global init.defaultBranch master  
[root@fedora ~]# git config --global core.autocrlf input  
[root@fedora ~]# git config --global core.safecrlf warn  
[root@fedora ~]# ssh-keygen -C "MikoGreen <miko.green@yandex.ru>"  
Generating public/private rsa key pair.  
Enter file in which to save the key (/root/.ssh/id_rsa):  
Created directory '/root/.ssh'.  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /root/.ssh/id_rsa  
Your public key has been saved in /root/.ssh/id_rsa.pub  
The key fingerprint is:  
SHA256:AsCthkrrb5zkr6PsHxKbMI/g1vmFPFAwP+8WilsyMsM MikoGreen <miko.green@yandex.ru>  
The key's randomart image is:  
+----[RSA 3072]-----+  
| ..+ |  
| ..= |  
| ..+ |  
| .+ ..0 |  
|o=.E +.=S |  
|+.B.@0.0 |  
|..+B+++. |  
|.. Bo.= |  
| +ooo |  
+----[SHA256]-----+  
[root@fedora ~]#
```

Рис. 4.7: Генерация ключа SSH

```
+----[SHA256]-----+
[root@fedora ~]# ^C
[root@fedora ~]# cat ~/.ssh/id_rsa.pub | xclip -sel clip
bash: xclip: команда не найдена...
Установить пакет «xclip», предоставляющий команду «xclip»? [N/y] y

* Ожидание в очереди...
Следующие пакеты должны быть установлены:
xclip-0.13-16.git11cba61.fc36.x86_64 Command line clipboard grabber
Продолжить с этими изменениями? [N/y] y

* Ожидание в очереди...
* Ожидание аутентификации...
* Ожидание в очереди...
* Загрузка пакетов...
* Запрос данных...
* Проверка изменений...
* Установка пакетов...

[root@fedora ~]# cat ~/.ssh/id_rsa.pub | xclip -sel clip
[root@fedora ~]#
```

Рис. 4.8: Скопировав ключ из локальной консоли в буфер обмена:

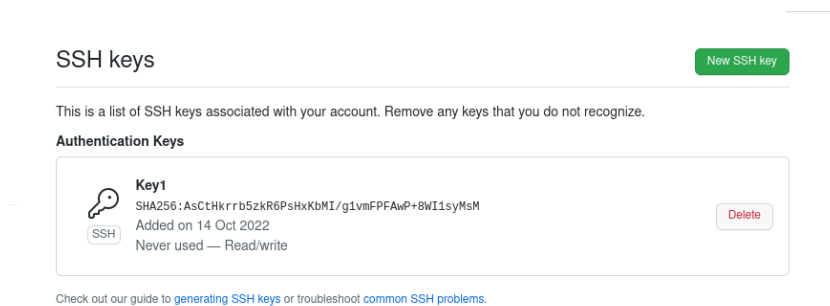


Рис. 4.9: Вставляем в специальное поле на сайте github.com и указываем имя ключа

```
[root@fedora ~]# cat ~/.ssh/id_rsa.pub | xclip -sel clip
[root@fedora ~]# mkdir -p ~/work/study/2022-2023/"Архитектура компьютера"
[root@fedora ~]#
```

Рис. 4.10: Создадим каталог для предмета “Архитектура компьютера”

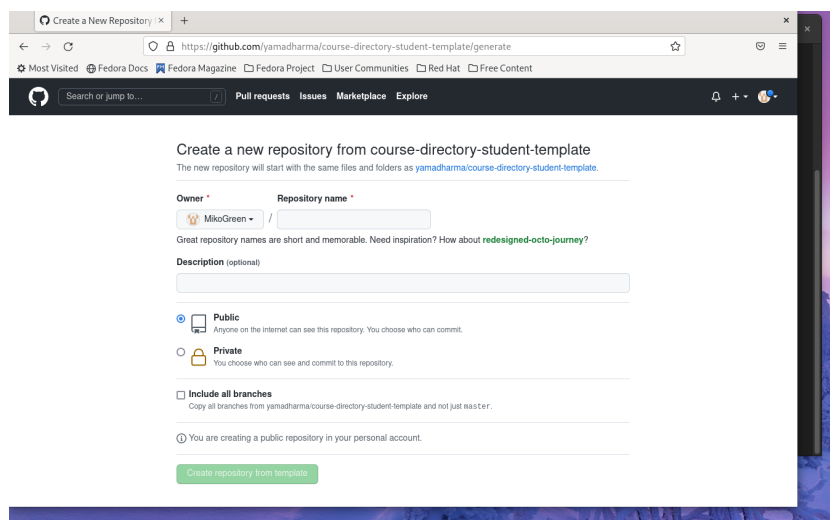


Рис. 4.11: Переходим по ссылке на страницу репозитория с шаблоном курса

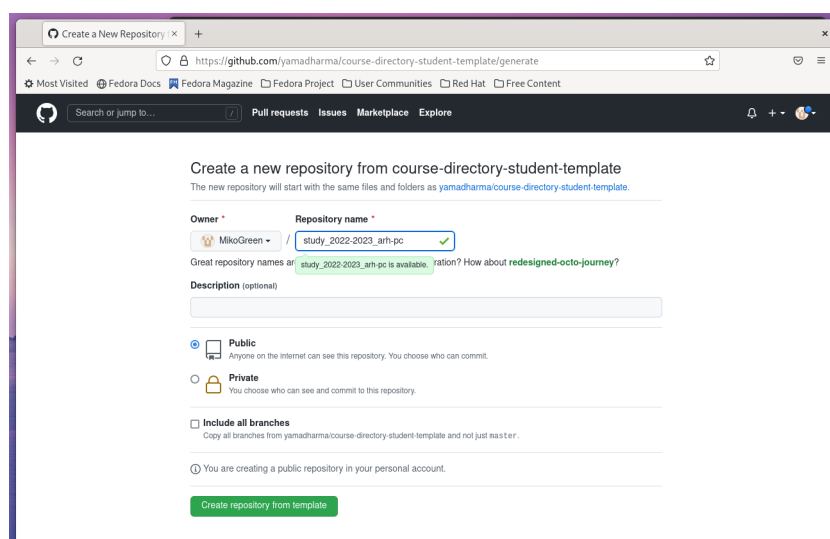


Рис. 4.12: Зададим имя репозитория и создадим его

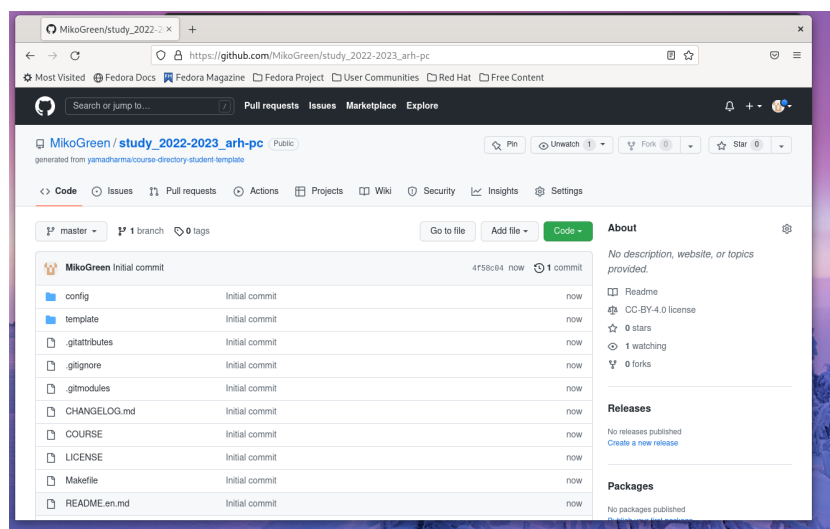


Рис. 4.13: Созданный репозиторий

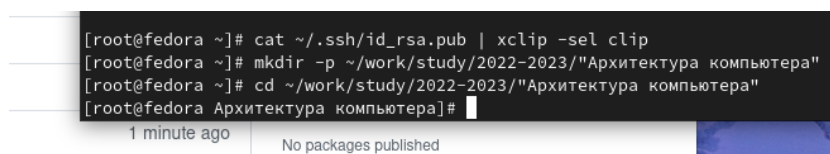


Рис. 4.14: Перейдем из терминала в каталог курса

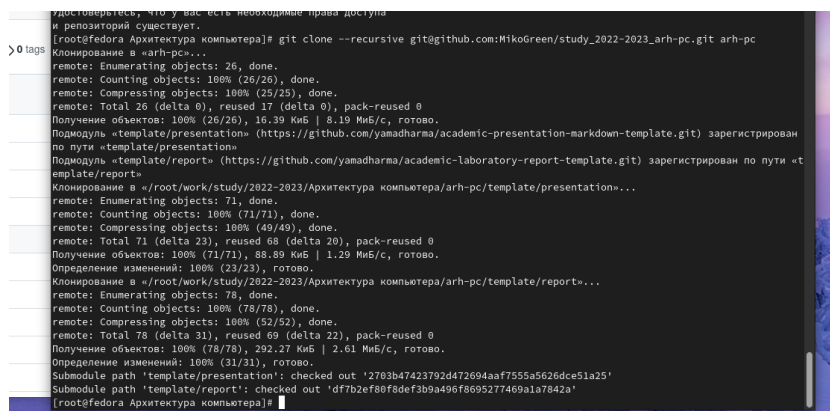


Рис. 4.15: Клонировем созданный репозиторий

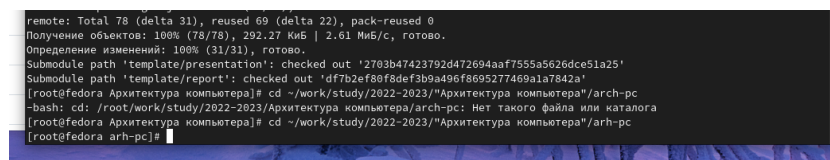


Рис. 4.16: Переходим в каталог курса

```
-bash: cd: /root/work/study/2022-2023/Архитектура
[root@fedora Архитектура компьютера]# cd ~/work/s
[root@fedora arh-pc]# rm package.json
rm: удалить обычный файл 'package.json'?
[root@fedora arh-pc]#
```

Рис. 4.17: Удаляем лишние файлы

```
[root@fedora arh-pc]# rm package.json
rm: удалить обычный файл 'package.json'?
[root@fedora arh-pc]# echo arch-pc > COURSE
[root@fedora arh-pc]# make
[root@fedora arh-pc]#
```

Рис. 4.18: Создаем необходимые каталоги

```
Fedora Magazine  Fedora Project  User Communities  Red Hat  Free Content

root@fedora:~/work/study/2022-2023/Архитектура компьютера/arh-pc

create mode 100644 labs/lab08/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab08/report/report.md
create mode 100644 labs/lab09/presentation/Makefile
create mode 100644 labs/lab09/presentation/image/kulyabov.jpg
create mode 100644 labs/lab09/presentation/presentation.md
create mode 100644 labs/lab09/report/Makefile
create mode 100644 labs/lab09/report/bib/cite.bib
create mode 100644 labs/lab09/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab09/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab09/report/report.md
create mode 100644 labs/lab10/presentation/Makefile
create mode 100644 labs/lab10/presentation/image/kulyabov.jpg
create mode 100644 labs/lab10/presentation/presentation.md
create mode 100644 labs/lab10/report/Makefile
create mode 100644 labs/lab10/report/bib/cite.bib
create mode 100644 labs/lab10/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab10/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab11/presentation/Makefile
create mode 100644 labs/lab11/presentation/image/kulyabov.jpg
create mode 100644 labs/lab11/presentation/presentation.md
create mode 100644 labs/lab11/report/Makefile
create mode 100644 labs/lab11/report/bib/cite.bib
create mode 100644 labs/lab11/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab11/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab11/report/report.md
create mode 100644 prepare
[root@fedora arh-pc]# git push
Перечисление объектов: 22, готово.
Подсчет объектов: 100% (22/22), готово.
Сжатие объектов: 100% (16/16), готово.
Запись объектов: 100% (20/20), 310.96 Кб | 2.29 Мб/с, готово.
Всего 20 (изменений 1), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:MikoGreen/study_2022-2023_arh-pc.git
  4f58c04..c5ed80d master -> master
[root@fedora arh-pc]#
```

Рис. 4.19: Отправляем файлы на сервер

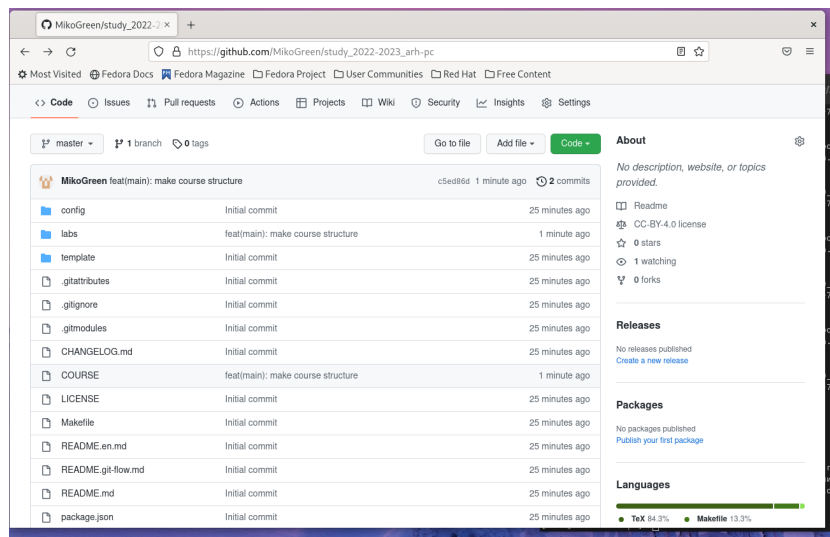


Рис. 4.20: Проверяем правильность создания иерархии рабочего пространства

Подгружаем предыдущие лабораторные работы

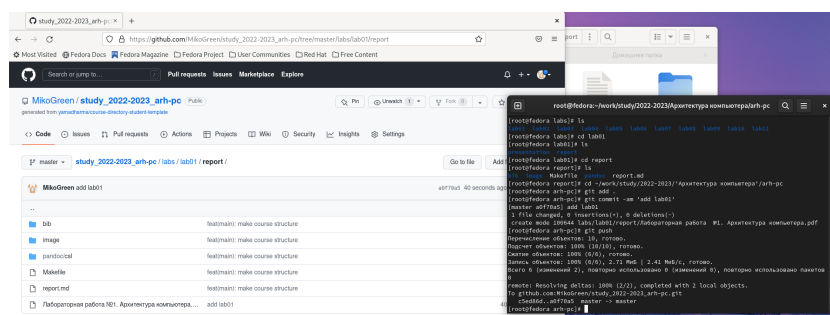


Рис. 4.21: Загрузка отчета по 1 лабораторной работе

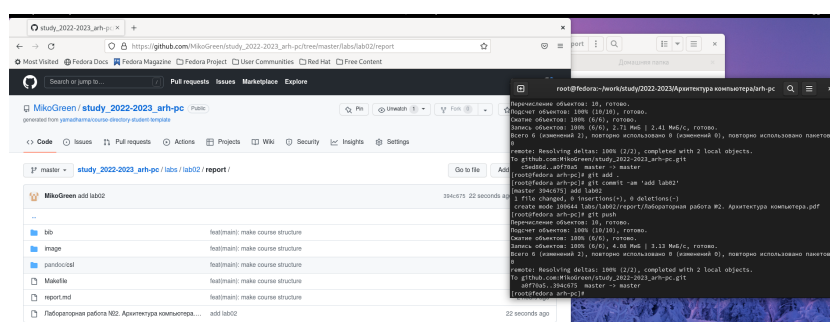


Рис. 4.22: Загрузка отчета по 2 лабораторной работе

А также подгружаем данную работу:

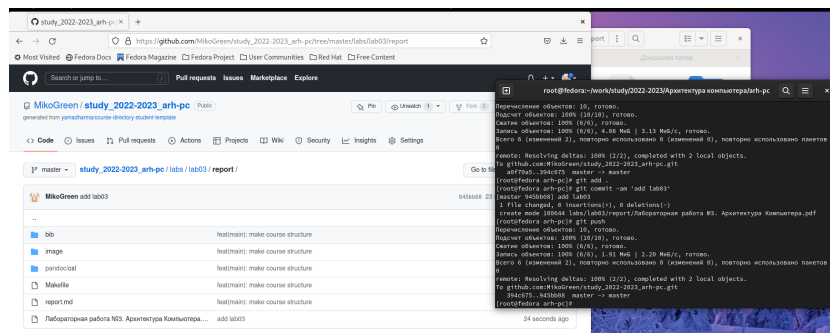


Рис. 4.23: Загрузка отчета по 3 лабораторной работе

5 Выводы

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