

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

## ОТЧЕТ

### ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

*дисциплина:* Архитектура компьютера

Студент: Юсупова Амина Руслановна

Группа: НКАбд-06-25

МОСКВА

2025\_г.

## Содержание

1 Цель работы.....	3
2 Задание.....	4
3 Теоретическое введение.....	5
4 Выполнение лабораторной работы.....	8
4.1 Настройка GitHub.....	8
4.2 Стандартная настройка Git.....	8
4.3 Создание SSH-ключа.....	9
4.4 Создание рабочего пространства и репозитория курса на основе шаблона.....	10
4.5 Создание репозитория курса на основе шаблона.....	11
4.6 Настройка каталога курса.....	12
5 Выполнение заданий для самостоятельной работы.....	15
6 Ответы на контрольные вопросы для самопроверки.....	17
7 Выводы.....	24
8 Список литературы.....	25

## **1 Цель работы**

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

## **2 Задание**

1. Настройка GitHub.
2. Стандартная настройка Git.
3. Создание SSH-ключа.
4. Создание рабочего пространства и репозитория курса на основе шаблона.
5. Создание репозитория курса на основе шаблона.
6. Настройка каталога курса.

### 3 Теоретическое введение

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. 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 — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить изменения, сделанные разными участниками, вручную выбрать нужную версию, отменить изменения вовсе или заблокировать 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 функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. 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. 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. 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). 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.

Некоторые команды Git (таб. 3.1)

Команда	Описание
<code>git init</code>	создание основного дерева репозитория
<code>git pull</code>	получение обновлений (изменений) текущего дерева из центрального 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 diff</code>	просмотр текущих изменений
<code>git add .</code>	добавить все изменённые и/или

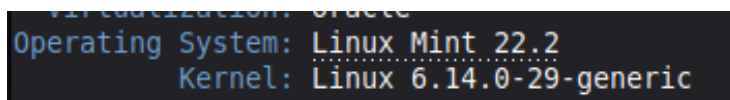
	созданные файлы и/или каталоги
git add имена_файлов	добавить конкретные изменённые и/или созданные файлы и/или каталоги
git rm имена_файлов	удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории)
git commit -am 'Описание коммита'	сохранить все добавленные изменения и все изменённые файлы
git checkout -b имя_ветки	создание новой ветки, базирующейся на текущей
git checkout имя_ветки	переключение на некоторую ветку (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой)
git push origin имя_ветки	отправка изменений конкретной ветки в центральный репозиторий
git merge --no-ff имя_ветки	слияние ветки с текущим деревом
git branch -d имя_ветки	удаление локальной уже слитой с основным деревом ветки
git branch -D имя_ветки	принудительное удаление локальной ветки
git push origin :имя_ветки	удаление ветки с центрального репозитория

Таблица 3.1 Описание некоторых команд git

## 4 Выполнение лабораторной работы

### 4.1 Техническое обеспечение

Работа была выполнена на домашнем компьютере под управлением операционной системы Linux Mint 22.2 (рис. 4.1.1).



```
Virtualization: Oracle  
Operating System: Linux Mint 22.2  
Kernel: Linux 6.14.0-29-generic
```

рис. 4.1.1 Операционная система компьютера

Создаю учетную запись на сайте GitHub (рис. 4.1.2). Далее я заполнила основные данные учетной записи.

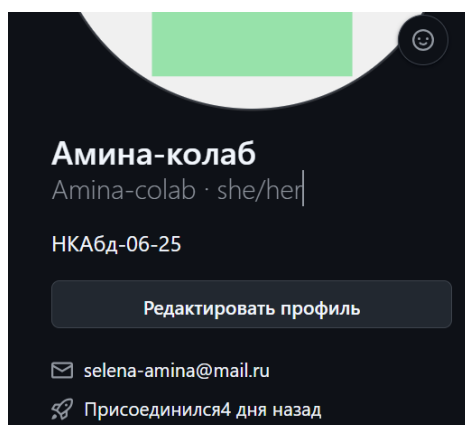
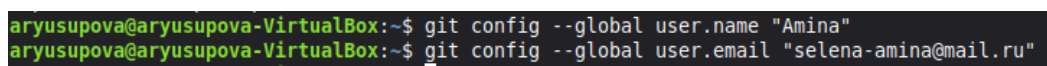


Рис. 4.1.2 Основные данные учетной записи Git hub

## 4.2 Базовые настройки Git

Открываю виртуальную машину, затем открываю терминал и делаю предварительную конфигурацию git. Ввожу команду `git config --global user.name ""`, указывая свое имя и команду `git config --global user.email "work@mail"`, указывая в ней электронную почту владельца, то есть мою (рис. 4.2.1).



```
aryusupova@aryusupova-VirtualBox:~$ git config --global user.name "Amina"  
aryusupova@aryusupova-VirtualBox:~$ git config --global user.email "selena-amina@mail.ru"
```

Рис. 4.2.1 Предварительная конфигурация git

Далее настраиваю параметры `utf-8`, имя начальной ветки, `autocrlf` и `safecrlf` (рис. 4.2.2). Задаю имя «master» для начальной ветки. Задаю параметр `autocrlf` со значением `input`, так как я работаю в системе Linux, чтобы конвертировать CRLF в LF только при коммитах (рис. 4.2.2). CR и LF – это символы, которые можно использовать для обозначения разрыва строки в текстовых файлах. Задаю параметр



safecrlf со значением warn, так Git будет проверять преобразование на обратимость (рис. 4.2.2). При значении warn Git только выведет предупреждение, но будет принимать необратимые конвертации.

```
aryusupova@aryusupova-VirtualBox:~$ git config --global core.quotePath false
aryusupova@aryusupova-VirtualBox:~$ git config --global init.defaultBranch master
aryusupova@aryusupova-VirtualBox:~$ git config --global core.autocrlf input
aryusupova@aryusupova-VirtualBox:~$ git config --global core.safecrlf warn
aryusupova@aryusupova-VirtualBox:~$
```

Рис. 4.2.2 Настройка параметров Git

### 4.3 Создание SSH-ключа

Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый). Для этого ввожу команду `ssh-keygen -C "Имя Фамилия, work@email"`, указывая имя владельца и электронную почту владельца (рис. 4.3.1). Ключ автоматически сохранится в каталоге `~/.ssh/`.

```
aryusupova@aryusupova-VirtualBox:~$ ssh-keygen -C "Amina Yusupova selenamail.ru"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/aryusupova/.ssh/id_ed25519):
Created directory '/home/aryusupova/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/aryusupova/.ssh/id_ed25519
Your public key has been saved in /home/aryusupova/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:sdmYQjk4xDSJJeYE+HD4wQ2a1kgidu5HNDVfopYEsP5k Amina Yusupova selenamail.ru
The key's randomart image is:
+--[ED25519 256]--+
|  =*0*o..o.      |
| 0*X++=++ .      |
| oB.+=.B.=       |
| . + .E + B       |
|   .o S .        |
|   . .           |
+-----[SHA256]-----+
aryusupova@aryusupova-VirtualBox:~$
```

Рис. 4.3.1 Генерация SSH-ключа

Xclip – утилита, позволяющая скопировать любой текст через терминал. В дистрибутиве Linux Mint утилита уже была. Копирую открытый ключ из директории, в которой он был сохранен, с помощью утилиты `xclip` (рис. 4.3.2).

```
aryusupova@aryusupova-VirtualBox:~$ cat ~/.ssh/id_rsa.pub | xclip -sel clip
```

Рис. 4.3.2 Копирование содержимого файла

Открываю браузер, захожу на сайт GitHub. Открываю свой профиль, настройки профиля и выбираю страницу «SSH and GPG keys». Нажимаю кнопку «New SSH key». Вставляю скопированный ключ в поле «Key». В поле Title указываю имя для ключа. Нажимаю «Add SSH-key», чтобы завершить добавление ключа (рис. 4.3.2).

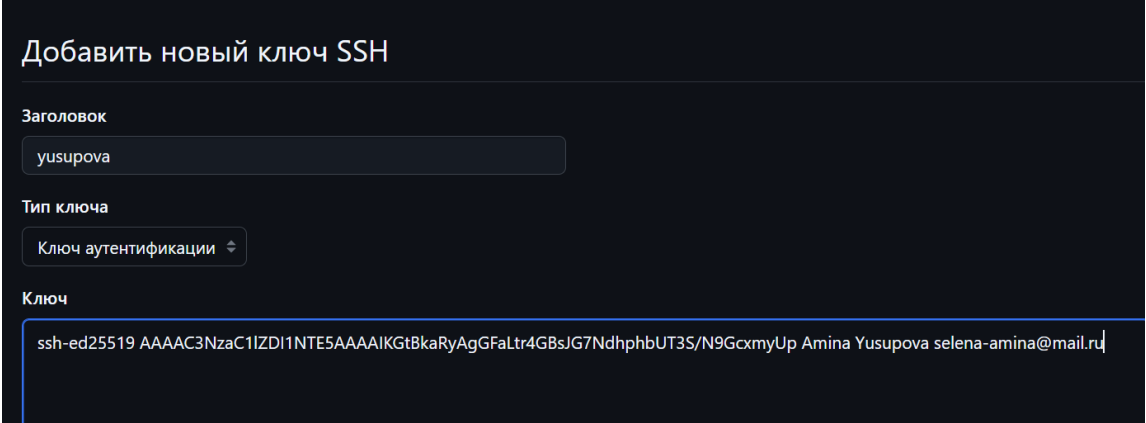


Рис. 4.3.2 Добавление ключа

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

Открываю терминал. Создаю директорию, рабочее пространство, с помощью утилиты `mkdir`, благодаря ключу `-p` создаю все директории после домашней `~/work/study/2025-2026/“Архитектура компьютера”` рекурсивно. Далее проверяю с помощью `ls`, действительно ли были созданы необходимые мне каталоги (рис. 4.4.1).

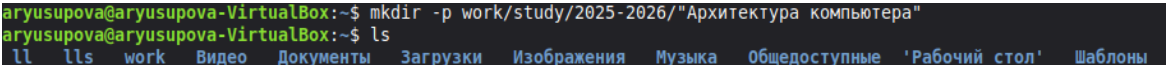


Рис. 4.4.1 Создание рабочего пространства

#### 4.5 Создание репозитория курса на основе шаблона

Создаю репозиторий на основе имеющего шаблона (рис. 4.5.1) через функционал клонирования интерфейса GitHub. (рис 4.5.2).

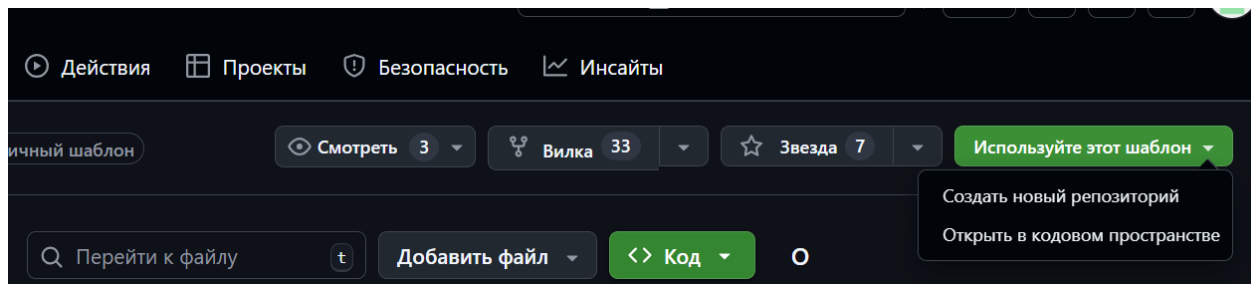


Рис. 4.5.1 Создание репозитория по шаблону

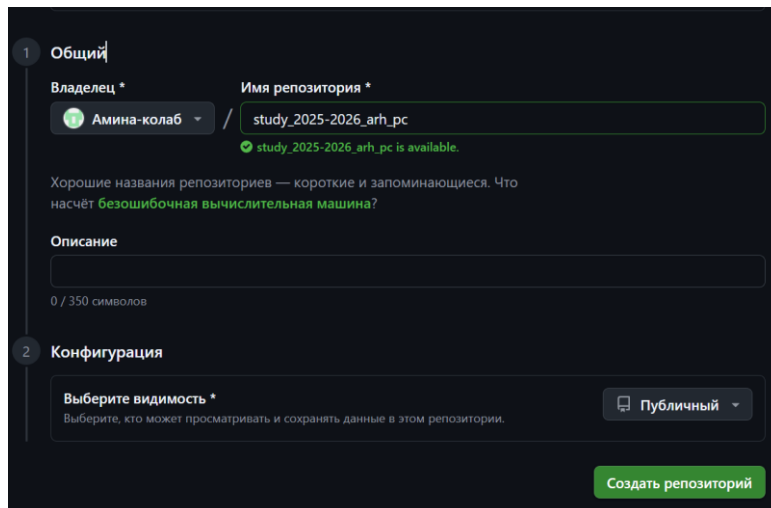


Рис. 4.5.2 Процесс клонирования репозитория через интерфейс GitHub

Сгенерированный репозиторий на основе шаблона клонирую на свой рабочий компьютер, для этого беру ссылку для клонирования через интерфейс GitHub (рис. 4.5.3) и затем ввожу в терминале `git clone`. (рис 4.5.4). Перед тем как вставлять ссылку в терминал необходимо перейти в нужный нам каталог с помощью команды `cd` (рис.4.5.5).

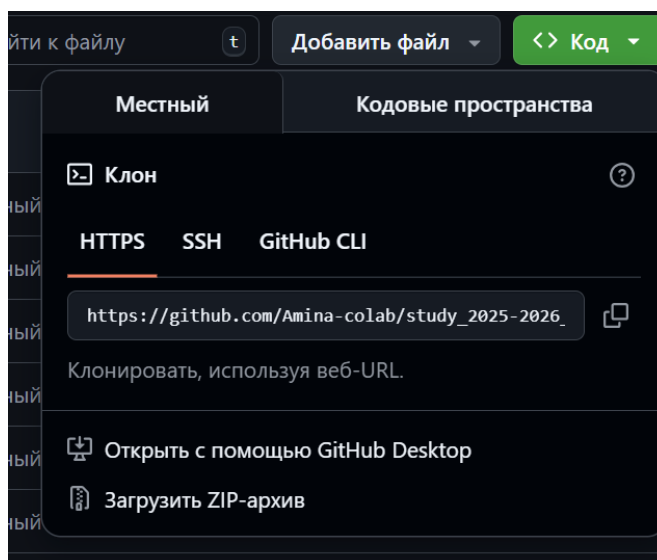


Рис. 4.5.3 Копирование ссылки для последующей вставки в терминал

```
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера$ git clone --recursive git@github.com:Amina-colab/study_2025-2026_arh_pc.git
Клонирование в «study_2025-2026_arh_pc»...
remote: Enumerating objects: 38, done.
remote: Counting objects: 100% (38/38), done.
remote: Compressing objects: 100% (36/36), done.
remote: Total 38 (delta 1), reused 27 (delta 1), pack-reused 0 (from 0)
Получение объектов: 100% (38/38), 23.45 КиБ | 1.07 МиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/aryusupova/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh_pc/template/presentation»...
remote: Enumerating objects: 161, done.
remote: Counting objects: 100% (161/161), done.
remote: Compressing objects: 100% (111/111), done.
remote: Total 161 (delta 60), reused 142 (delta 41), pack-reused 0 (from 0)
Получение объектов: 100% (161/161), 2.65 МиБ | 1.18 МиБ/с, готово.
Определение изменений: 100% (60/60), готово.
Клонирование в «/home/aryusupova/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh_pc/template/report»...
remote: Enumerating objects: 221, done.
remote: Counting objects: 100% (221/221), done.
remote: Compressing objects: 100% (152/152), done.
remote: Total 221 (delta 98), reused 180 (delta 57), pack-reused 0 (from 0)
Получение объектов: 100% (221/221), 765.46 КиБ | 934.00 КиБ/с, готово.
Определение изменений: 100% (98/98), готово.
Submodule path 'template/presentation': checked out '6efd5c4ee78e4456caff3dc7062cfcad26058ca6'
Submodule path 'template/report': checked out '89a9622199b4df88227b9b3fa3d4714c85f68dd2'
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера$
```

Рис. 4.5.4 Копирование репозитория на рабочий компьютер

```
aryusupova@aryusupova-VirtualBox:~$ cd ~/work/study/2025-2026/"Архитектура компьютера"
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера$
```

Рис. 4.5.5 Переход в каталог

## 4.6 Настройка каталога курса

Перехожу в каталог курса и удаляю лишние файлы (рис. 4.6.1). Также создам необходимые каталоги (рис. 4.6.2, рис 4.6.3) и отправлю файлы на сервер (рис. 4.6.4, рис.4.6.5). Затем проверю правильность создания иерархии рабочего стола в локальном репозитории (рис. 4.6.6, рис. 4.6.7) и на странице github (рис. 4.6.8).

```
aryusupova@aryusupova-VirtualBox:~$ cd ~/work/study/2025-2026/"Архитектура компьютера"/arch-pc
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc$
```

Рис. 4.6.1 Переход в каталог курса и удаление ненужных файлов

```
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ rm package.json
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ echo arch-pc > COURSE
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc$
```

Рис. 4.6.2 Создание необходимых каталогов

```
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare       Generate directories structure
  submodule     Update submules
```

Рис. 4.6.3 Создание необходимых каталогов

```

aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git add .
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git commit -am 'feat(main): make course structure'
[master 515b11d] feat(main): make course structure
211 files changed, 8069 insertions(+), 216 deletions(-)
delete mode 100644 CHANGELOG.md
create mode 100644 labs/README.md
create mode 100644 labs/README.ru.md
create mode 100644 labs/lab01/presentation/.gitignore
create mode 100644 labs/lab01/presentation/.marksmen.toml
create mode 100644 labs/lab01/presentation/.projectile
create mode 100644 labs/lab01/presentation/Makefile
create mode 100644 labs/lab01/presentation/quarto.yml
create mode 100644 labs/lab01/presentation/resources/image/logo_rudn.png
create mode 100644 labs/lab01/presentation/arch-pc--lab01--presentation.qmd
create mode 100644 labs/lab01/presentation/image/kulyabov.jpg
create mode 100644 labs/lab01/report/.gitignore
create mode 100644 labs/lab01/report/.marksmen.toml
create mode 100644 labs/lab01/report/.projectile
create mode 100644 labs/lab01/report/Makefile
create mode 100644 labs/lab01/report/quarto.yml
create mode 100644 labs/lab01/report/resources/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab01/report/arch-pc--lab01--report.qmd
create mode 100644 labs/lab01/report/bib/cite.bib
create mode 100644 labs/lab01/report/image/solvay.jpg
create mode 100644 labs/lab02/presentation/.gitignore
create mode 100644 labs/lab02/presentation/.marksmen.toml
create mode 100644 labs/lab02/presentation/.projectile
create mode 100644 labs/lab02/presentation/Makefile
create mode 100644 labs/lab02/presentation/quarto.yml
create mode 100644 labs/lab02/presentation/resources/image/logo_rudn.png
create mode 100644 labs/lab02/presentation/arch-pc--lab02--presentation.qmd
create mode 100644 labs/lab02/presentation/image/kulyabov.jpg
create mode 100644 labs/lab02/report/.gitignore
create mode 100644 labs/lab02/report/.marksmen.toml
create mode 100644 labs/lab02/report/.projectile
create mode 100644 labs/lab02/report/Makefile
create mode 100644 labs/lab02/report/quarto.yml
create mode 100644 labs/lab02/report/resources/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab02/report/arch-pc--lab02--report.qmd

```

Рис. 4.6.4 Отправка файлов на сервер

```

aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git push

kk
^[D^[D^[D^[D^[D
Перечисление объектов: 65, готово.
Подсчет объектов: 100% (65/65), готово.
Сжатие объектов: 100% (51/51), готово.
Запись объектов: 100% (63/63), 699.76 КиБ | 60.00 КиБ/с, готово.
Всего 63 (изменений 23), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (23/23), completed with 1 local object.
To github.com:Amina-colab/study_2025-2026_arh_pc.git
  00376c6..515b11d  master -> master

```

Рис. 4.6.5 Отправка файлов на сервер

```

aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ ls
COURSE labs LICENSE Makefile presentation README.en.md README.git-flow.md README.md template

```

Рис. 4.6.6 Проверка локального репозитория

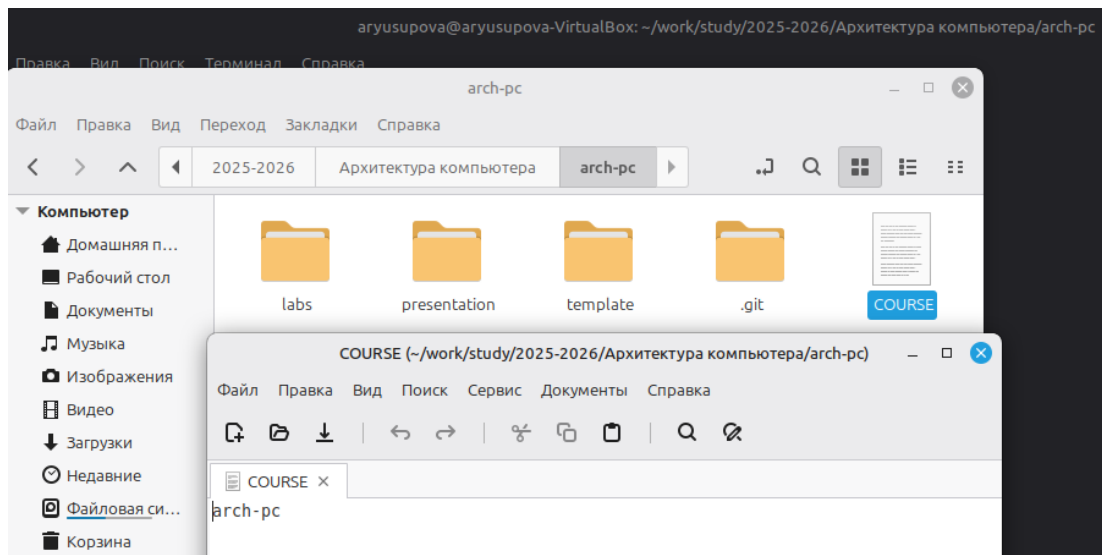


Рис. 4.6.7 Результат выполнения команды echo в локальном репозитории

Амина-колаб Первоначальный коммит 00376c6 · 3 hours ago 1 коммит		
шаблон	Первоначальный коммит	3 hours ago
.gitattributes	Первоначальный коммит	3 hours ago
.gitignore	Первоначальный коммит	3 hours ago
.gitmodules	Первоначальный коммит	3 hours ago
CHANGELOG.md	Первоначальный коммит	3 hours ago
КУРС	Первоначальный коммит	3 hours ago
ЛИЦЕНЗИЯ	Первоначальный коммит	3 hours ago
Makefile	Первоначальный коммит	3 hours ago
README.ru.md	Первоначальный коммит	3 hours ago
README.git-flow.md	Первоначальный коммит	3 hours ago
README.md	Первоначальный коммит	3 hours ago
paket.json	Первоначальный коммит	3 hours ago

Рис. 4.6.8 Проверка репозитория на github.com

## 5 Задания для самостоятельной работы

Создаю в локальном репозитории файл отчета 2-ой лабораторной работы в соответствующей папке, также копирую отчет первой лабораторной работы в папку, предназначенную для него. (Рис. 5.1)

```
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ touch labs/lab02/report/л02_Юсупова_отчёт.pdf
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ cp /home/Amina-colab/Загрузки/л01_Юсупова_отчёт.pdf labs/lab01/report
cp: не удалось выполнить stat для '/home/Amina-colab/Загрузки/л01_Юсупова_отчёт.pdf': Нет такого файла или каталога
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ cp /home/aryusupova/Загрузки/л01_Юсупова_отчёт.pdf labs/lab01/report
```

Рис. 5.1 Создание и копирование файлов отчетов в локальном репозитории

Далее загружаю файлы на github (рис. 5.2). Наконец проверяю наличие файлов на github (рис. 5.3, рис. 5.4).

```
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git add .
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git commit -am 'uploaded previous reports'
[master 22f8ce5] uploaded previous reports
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 labs/lab01/report/л01_Юсупова_отчёт.pdf
create mode 100644 labs/lab02/report/л02_Юсупова_отчёт.pdf
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git push
Перечисление объектов: 14, готово.
Подсчет объектов: 100% (14/14), готово.
Сжатие объектов: 100% (8/8), готово.
Запись объектов: 100% (8/8), 1.01 Миб | 3.50 Миб/с, готово.
Всего 8 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 3 local objects.
To github.com:Amina-colab/study_2025-2026_arh_pc.git
515b11d..22f8ce5 master -> master
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc$
```

Рис. 5.2 Загрузка файлов на github

исследование_2025-2026_арх_пк / лаборатории / lab01 / отчет			Добавить файл	...
Амина-колаб загрузил предыдущие отчеты			22f8ce5 · 1 минуту назад	История
Имя	Последнее сообщение о коммите	Дата последн...		
..				
_resources/ csl	feat(main): создать структуру курса	11 часов назад		
нагрудник	feat(main): создать структуру курса	11 часов назад		
изображение	feat(main): создать структуру курса	11 часов назад		
.gitignore	feat(main): создать структуру курса	11 часов назад		
.marksmat.toml	feat(main): создать структуру курса	11 часов назад		
.онارد	feat(main): создать структуру курса	11 часов назад		
Makefile	feat(main): создать структуру курса	11 часов назад		
_quarto.yml	feat(main): создать структуру курса	11 часов назад		
arch-pc--lab01--report.qmd	feat(main): создать структуру курса	11 часов назад		
л01_Юсупова_отчёт.pdf	загрузил предыдущие отчеты	1 минуту назад		

Рис. 5.3 Проверка файлов на github.

исследование\_2025-2026\_арх\_пк / лаборатории / lab02 / отчет /

Добавить файл

Амина-колаб загрузил предыдущие отчеты

22f8ce5 · 5 минут назад

История

Имя	Последнее сообщение о коммите	Дата последнег...
..		
.._resources/ csl	feat(main): создать структуру курса	11 часов назад
нагрудник	feat(main): создать структуру курса	11 часов назад
изображение	feat(main): создать структуру курса	11 часов назад
.gitignore	feat(main): создать структуру курса	11 часов назад
.marksmark.toml	feat(main): создать структуру курса	11 часов назад
.снаряд	feat(main): создать структуру курса	11 часов назад
Makefile	feat(main): создать структуру курса	11 часов назад
_quarto.yml	feat(main): создать структуру курса	11 часов назад
arch-pc-lab02---report.qmd	feat(main): создать структуру курса	11 часов назад
/l02_Юсупова_отчёт.pdf	загрузил предыдущие отчеты	5 минут назад

Рис. 5.4 Проверка файлов на github



## 6 Ответы на вопросы для самопроверки

1) Системы контроля версий (VCS – Version Control Systems) – это программное обеспечение, которое отслеживает изменения в файлах (чаще всего исходном коде программ, но также и в других документах) с течением времени. Проще говоря, VCS позволяет сохранять "снимки" вашего проекта на разных этапах его разработки, создавая историю всех внесенных правок.

2) Хранилище (репозиторий) в системе контроля версий (VCS) — это система, которая обеспечивает хранение всех существовавших версий файлов. Часто говорят об удалённом репозитории (копия кода на каком-то сервере) и локальном репозитории (копия на компьютере разработчика).

Commit — это запись изменений. С помощью коммитов изменения, внесённые в рабочую копию, заносятся в хранилище.

История — это список предыдущих изменений. Благодаря истории можно отследить изменения, вносимые в репозиторий.

Рабочая копия — это копия файла, с которой непосредственно ведётся работа (находится вне репозитория). Перед началом работы рабочую копию можно получить из одной из версий, хранящихся в репозитории

3) В централизованной системе существует один главный, центральный сервер, на котором хранится вся история проекта (репозиторий). Разработчики работают со своими локальными копиями файлов, но для того, чтобы зафиксировать изменения (commit) или получить последние обновления, они должны постоянно взаимодействовать с этим центральным сервером.

Примеры: Subversion (SVN): Один из наиболее известных представителей централизованных VCS; CVS (Concurrent Versions System): Более старая система, предшественник SVN.

В децентрализованной системе каждый разработчик имеет полную копию всего репозитория (включая всю историю изменений) на своем локальном компьютере. Нет единого "главного" сервера в том смысле, как это было в CVS. Взаимодействие происходит между локальными репозиториями, и команды часто

используют удаленный репозиторий (например, на GitHub) для синхронизации и обмена изменениями.

Примеры DVCS: Git: Самая популярная и широко используемая DVCS в мире; Mercurial: Еще одна мощная DVCS, похожая на Git, но с другими командами и философией; Bazaar (bzt): Менее распространенная, но также децентрализованная VCS.

Сравнительная таблица централизованной системы и децентрализованной системы

параметры	Централизованные VCS (CVCS)	Децентрализованные VCS (DVCS)
Хранение истории	Один центральный сервер	Каждый разработчик имеет полную копию
Основная точка работы	Центральный сервер	Локальный репозиторий
Зависимость от сети	Высокая	Низкая (в большинстве операций)
Отказоустойчивость	Низкая (единая точка отказа)	Высокая
Производительность	Ниже (операции с сервером)	Выше (локальные операции)

4) При единоличной работе с VCS, вы выполняете следующие действия:

Инициализация репозитория:

- \* Если проект новый: `git init` (создает локальное хранилище).
- \* Если проект уже существует: `git init` в корне проекта.

Создание/изменение файлов:

- \* Я работаю с файлами в вашей локальной папке проекта.

Отслеживание изменений:

- \* `git add <файл>`: Добавить конкретный файл для следующего коммита.

- \* `git add .`: Добавить все измененные и новые файлы в текущей директории. Сохранение изменений (Commit):

- \* `git commit -m "Описание изменений"`: Сохранить добавленные файлы как новую версию в локальном хранилище.

Просмотр истории:

- \* `git log`: Посмотреть список всех коммитов.

Возврат к предыдущим версиям:

- \* `git checkout <хэш_коммита>`: Переключиться на состояние проекта из указанного коммита.

- \* `git checkout -- <файл>`: Откатить изменения в конкретном файле к состоянию последнего коммита.

Создание и управление ветками (опционально):

- \* `git branch <название_ветки>`: Создать новую ветку.

- \* `git checkout <название_ветки>`: Переключиться на другую ветку.

- \* `git merge <название_ветки>`: Объединить изменения из другой ветки в текущую.

Синхронизация с удаленным репозиторием (если используется):

- \* `git remote add origin <URL_репозитория>`: Подключить локальное хранилище к удаленному.

- \* `git push origin <название_ветки>`: Отправить локальные коммиты в удаленное хранилище.

- \* `git pull origin <название_ветки>`: Загрузить изменения из удаленного хранилища.

5) Опишите порядок работы с общим хранилищем VCS.

При работе с общим хранилищем VCS (распределенным или централизованным) соблюдается следующий порядок:

Pull/Update: Перед началом работы или перед фиксацией своих изменений, загрузите последние версии файлов из общего хранилища.

- \* Git: `git pull origin <ветка>`

- \* SVN: `svn update`

Редактирование файлов: Вносим необходимые изменения в локальную рабочую копию.

Staging/Add: Подготовьте измененные файлы к фиксации.

- \* Git: `git add <файл>` или `git add .`

Commit: Сохраните изменения.

- \* Git: `git commit -m "Описание изменений"` (сохраняет локально)

- \* SVN: `svn commit -m "Описание изменений"` (сохраняет в общем хранилище)

Push (для DVCS): Отправьте локальные коммиты в общее хранилище.

- \* Git: `git push origin <ветка>`

Разрешение конфликтов (при возникновении): Если VCS сообщает о конфликте, вручную отредактируйте файлы, уберите маркеры конфликтов (`<<<<<<<`, `=====`, `>>>>>>>`), сохраните, добавьте разрешенные файлы (`git add <файл>`) и сделайте новый commit.

Работа с ветками (опционально):

- \* Создание: `git checkout -b <новая_ветка>`

- \* Переключение: `git checkout <ветка>`

- \* Слияние: `git merge <ветка>` (после переключения на целевую ветку).

Кратко: Pull -> Редактирование -> Add -> Commit -> Push (если DVCS) -> Разрешение конфликтов (при необходимости).

б) Git решает следующие основные задачи:

1. Отслеживание истории изменений: Сохраняет все версии файлов и их изменения.
2. Совместная работа: Позволяет нескольким разработчикам работать над проектом одновременно.
3. Управление ветками: Изолирует разработку новых функций или исправление ошибок.
4. Резервное копирование: Хранит полный образ проекта.
5. Разрешение конфликтов: Помогает управлять ситуациями, когда изменения пересекаются.
6. Быстрый доступ к данным: Операции выполняются локально, что обеспечивает высокую скорость.

7) git init: Инициализирует новый Git-репозиторий в текущей директории.

git clone <URL>: Создает локальную копию удаленного репозитория.

git add <файл>: Добавляет изменения в файле в индекс для следующего коммита.

git commit -m "<сообщение>": Сохраняет изменения из индекса в локальный репозиторий с описанием.

git status: Показывает текущее состояние рабочей директории (измененные, новые, добавленные файлы).

git log: Отображает историю коммитов.

git diff: Показывает различия между рабочей директорией и индексом, или между индексами и коммитами.

`git checkout <ветка>`: Переключает рабочую директорию на указанную ветку или коммит.

`git branch`: Управляет ветками (создает, удаляет, перечисляет).

`git merge <ветка>`: Объединяет изменения из указанной ветки в текущую.

`git pull`: Загружает изменения из удаленного репозитория и объединяет их с текущей веткой.

`git push`: Отправляет локальные коммиты в удаленный репозиторий.

`git remote`: Управляет удаленными репозиториями.

`git reset`: Отменяет коммиты или возвращает файлы в определенное состояние.

`git revert`: Создает новый коммит, отменяющий изменения предыдущего коммита.

`git stash`: Временно сохраняет незафиксированные изменения, чтобы очистить рабочую директорию.

8) Приведите примеры использования при работе с локальным и удалённым репозиториями.

Работа с локальным репозиторием:

1. Инициализация: `git init` (создать новый репозиторий).
2. Создание/изменение файлов: Работаете с файлами.
3. Статус: `git status` (увидеть, что изменилось).
4. Добавление: `git add <файл>` (подготовить к коммиту).
5. Коммит: `git commit -m "Добавлена новая функция"` (сохранить локально).
6. Просмотр истории: `git log` (увидеть коммиты).
7. Создание ветки: `git checkout -b новая_фича` (для новой фичи).

8. Слияние: `git merge main` (объединить ветку в main).

Работа с удаленным репозиторием (например, на GitHub):

1. Клонирование: `git clone <URL_репозитория>` (скопировать с сервера).

2. Pull: `git pull origin main` (получить последние изменения с сервера).

3. Push: `git push origin main` (отправить локальные коммиты на сервер).

4. Работа с ветками на удалёнке:

- \* `git push origin <новая_ветка>` (отправить локальную ветку на сервер).

- \* `git checkout -b <ветка_с_сервера> origin/<ветка_с_сервера>` (создать локальную ветку на основе удалённой).

## **7 Выводы**

При выполнении лабораторной работы я изучила идеологию и применение средств контроля версий. Также я приобрела практические навыки по работе с системой git.



## **8 Список литературы**

- 1 <https://esystem.rudn.ru/mod/resource/view.php?id=1030495>
- 2 <https://docs.github.com/ru/get-started/learning-to-code/getting-started-with-git>
- 3 [https://github.com/Amina-colab/study\\_2025-2026\\_arh\\_pc/tree/master](https://github.com/Amina-colab/study_2025-2026_arh_pc/tree/master)
- 4 <https://github.com/tgabriel22/Lab2>