

# 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: Грицко Сергей

Группа: НКАбд-02-25

МОСКВА

2025 г.

## Оглавление

Цель работы .....	3
Теоретическое введение.....	4
2.1 Системы контроля версий. Общие понятия .....	4
2.2 Система контроля версий Git.....	4
2.2 Основные команды git.....	5
Выполнения лабораторной работы .....	7
3.1 Настройка github.....	7
3.2 Базовая настройка git.....	8
3.3 Создание SSH-ключа .....	8
3.4 Создание рабочего пространства и репозитория курса.....	10
3.5 Настройка каталога курса .....	11
Выполнение самостоятельной работы.....	12
Выводы .....	13

## **Цель работы**

Целью работы является изучение идеологии и применения средств контроля версий, приобретение практических навыков по работе с системой контроля версий git.

## Теоретическое введение

### *2.1 Системы контроля версий. Общие понятия*

Что такое «**система контроля версий**» и почему это важно? **Система контроля версий** — это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии. Для контроля версий файлов в этой книге в качестве примера будет использоваться исходный код программного обеспечения, хотя на самом деле вы можете использовать контроль версий практически для любых типов файлов.

Если вы графический или web-дизайнер и хотите сохранить каждую версию изображения или макета (скорее всего, захотите), система контроля версий (далее VCS) — как раз то, что нужно. Она позволяет вернуть файлы к состоянию, в котором они были до изменений, вернуть проект к исходному состоянию, увидеть изменения, увидеть, кто последний менял что-то и вызвал проблему, кто поставил задачу и когда и многое другое. Использование VCS также значит в целом, что, если вы сломали что-то или потеряли файлы, вы спокойно можете всё исправить. В дополнение ко всему вы получите всё это без каких-либо дополнительных усилий. [1]

### *2.2 Система контроля версий Git*

**Git** — это распределённая система управления версиями, которая позволяет командам разработчиков программного обеспечения иметь несколько локальных копий кодовой базы проекта, независимых друг от друга. Эти копии, или ветви, можно быстро создавать, объединять и удалять, что позволяет командам экспериментировать с минимальными вычислительными затратами, прежде чем объединить их в основную ветку (иногда называемую главной веткой). Git известен своей скоростью, совместимостью с рабочими процессами и открытым исходным кодом. [2]

## Распределённая система контроля версий

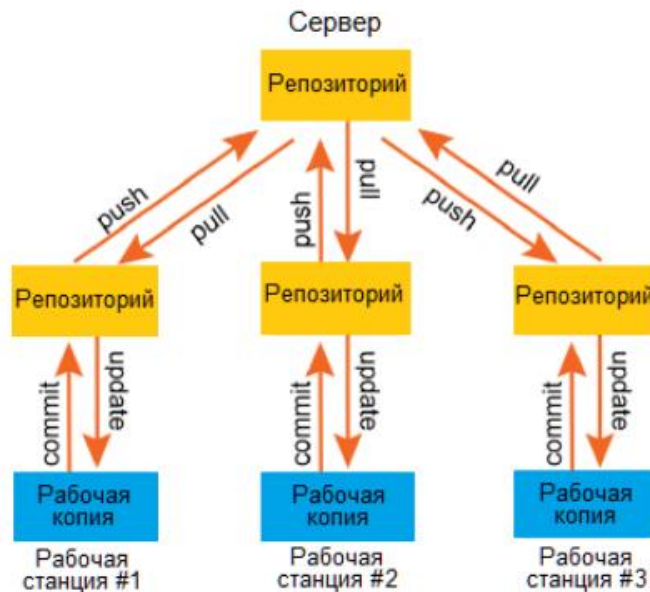


Рисунок 1. Система контроля версий

### 2.2 Основные команды *git*

В данной таблице будут предоставлены все основные команды **Git**. [3]

Основные команды	Описание команд
<b>git init</b>	Инициализация репозитория
<b>git config</b>	Настройки параметров конфигурации Git.
<b>git status</b>	Состояние рабочего каталога и индекса.
<b>git add</b>	Добавление изменений в индекс.
<b>git reset</b>	Отмена изменений в репозитории.
<b>git commit</b>	Сохранение изменений в локальном репозитории Git.
<b>git log</b>	Просмотр истории коммитов
<b>git push</b>	Отправка коммитов в удаленный репозиторий
<b>git branch</b>	Управление ветками
<b>git switch</b>	Переключение между ветками

<b>git clone</b>	Создание копии удаленного репозитория
<b>git stash</b>	Временное хранилище
<b>git config alias</b>	Создание псевдонимов
<b>git checkout</b>	Работа с ветками, восстановление файлов и переключение на конкретные коммиты
<b>git merge</b>	Слияние изменений веток
<b>git fetch</b>	Загрузка обновлений из удаленного репозитория
<b>git pull</b>	Извлечение и слияние изменения
<b>git rebase</b>	Ашалеть, что это за ЗВЕРЬ?
<b>git diff</b>	Просмотр различий между файлами
<b>git difftool</b>	Просмотр различий и редактирование файлов
<b>git remote</b>	Работа с удалёнными репозиториями
<b>git tag</b>	Теги
<b>git restore</b>	Восстановления файлов из индекса или коммитов
<b>git cherry-pick</b>	Применение коммитов из одной ветки на другую
<b>git revert</b>	Откат изменений

Таблица 1. Основные команды

# Выполнения лабораторной работы

## 3.1 Настройка github

Первым шагом было создание учётной записи на сайте GitHub, который будет использоваться как удаленный сервер для хранения репозиториев.

### Инструкция по выполнению:

1. Перейдите на сайт <https://github.com/>.
2. Нажмите на кнопку "Sign up".
3. Следуйте инструкциям на экране: введите вашу электронную почту, создайте пароль и выберите имя пользователя.
4. Подтвердите свою учетную запись через письмо, которое придет на указанную почту.

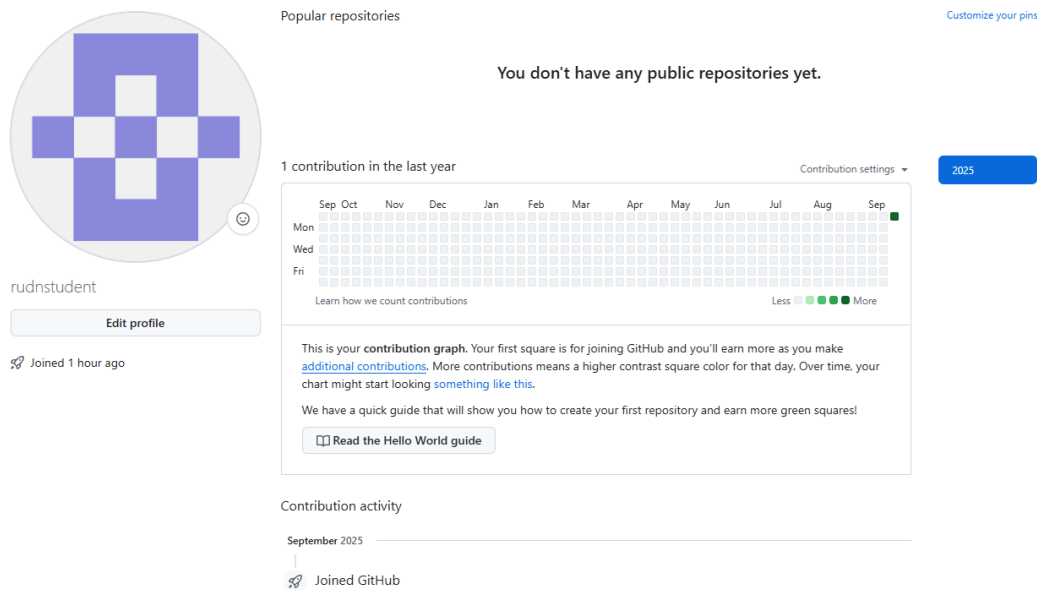
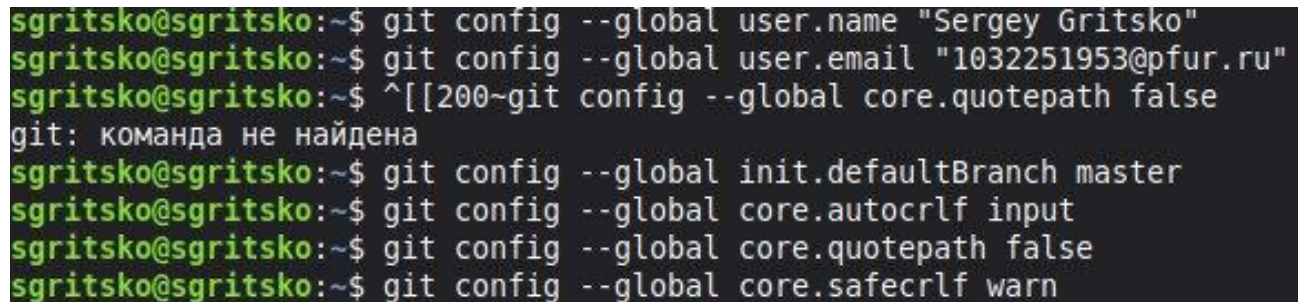


Рисунок 2. Мой профиль на GitHub

Я успешно зарегистрировался на GitHub. Процесс оказался простым и интуитивно понятным.

### 3.2 Базовая настройка git

После установки **Git** на локальной машине необходимо было выполнить базовую конфигурацию: указать имя пользователя и адрес электронной почты, которые будут отображаться в истории коммитов, а также настроить другие параметры для корректной работы.

A screenshot of a terminal window showing a series of Git configuration commands being executed. The prompt is 'sgritsko@sgritsko:~\$'. The commands are: 'git config --global user.name "Sergey Gritsko"', 'git config --global user.email "1032251953@pfur.ru"', '^[[200~git config --global core.quotepath false' (which results in an error 'git: команда не найдена'), 'git config --global init.defaultBranch master', 'git config --global core.autocrlf input', 'git config --global core.quotepath false', and 'git config --global core.safecrlf warn'.

```
sgritsko@sgritsko:~$ git config --global user.name "Sergey Gritsko"
sgritsko@sgritsko:~$ git config --global user.email "1032251953@pfur.ru"
sgritsko@sgritsko:~$ ^[[200~git config --global core.quotepath false
git: команда не найдена
sgritsko@sgritsko:~$ git config --global init.defaultBranch master
sgritsko@sgritsko:~$ git config --global core.autocrlf input
sgritsko@sgritsko:~$ git config --global core.quotepath false
sgritsko@sgritsko:~$ git config --global core.safecrlf warn
```

Рисунок 2. Настройка **Git**

Я выполнил базовую настройку **Git**. Эти команды нужны, чтобы каждый мой коммит (сохранение изменений) был подписан моим именем и почтой. Это очень важно для отслеживания истории изменений, особенно при работе в команде.

### 3.3 Создание SSH-ключа

Для безопасного подключения к GitHub без необходимости каждый раз вводить пароль, я сгенерировал пару SSH-ключей (приватный и публичный) и добавил публичный ключ в свой профиль на GitHub.



```

sgritsko@sgritsko:~$ ssh-keygen -C "Sergey Gritsko 1032251953@pfur.ru"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/sgritsko/.ssh/id_ed25519):
Created directory '/home/sgritsko/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match. Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/sgritsko/.ssh/id_ed25519
Your public key has been saved in /home/sgritsko/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:HuMuSSKa5sbB8GKvYhHHxcNwKg9UH2Avu3P0TIQYwjQ Sergey Gritsko 1032251953@pfur.ru
The key's randomart image is:
+--[ED25519 256]--+
| .oE=0            |
| . + 0=0          |
| +.+.+. .        |
| ..+00 .          |
| 0.00 . .S        |
| .=0 + =0 0       |
| 0+++ + +0        |
| += .0 0.         |
| *0. . .          |
+-----[SHA256]-----+

```

Рисунок 3. Генерация ключи

Все, мы создали ключ, теперь его можно скопировать данной командой:

```

sgritsko@sgritsko:~$ cat ~/.ssh/id_ed25519.pub | xclip -sel clip

```

Рисунок 4. Копирование ключа

Теперь давайте добавим ключ на **GitHub**:

1. Зайдем в свой профиль на **GitHub**.
2. Перейдем в **Settings -> SSH and GPG keys**.
3. Нажмем **New SSH key**.
4. В поле **Title** введем название ключа (например, "**My Work Laptop**"), а в поле **Key** вставим скопированный ключ.
5. Нажмите **Add SSH key**.

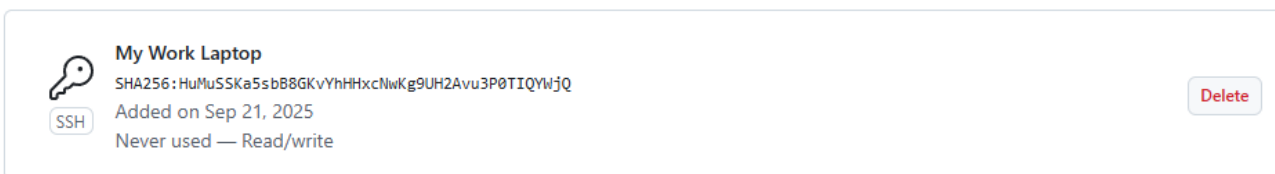


Рисунок 5. SSH key в GitHub

Я сгенерировал SSH-ключ. Это позволяет мне безопасно подключаться к моему репозиторию на **GitHub**. Публичный ключ я добавил в настройки аккаунта, а приватный остался на моем компьютере, что обеспечивает безопасность соединения.

### 3.4 Создание рабочего пространства и репозитория курса

На этом этапе я создал репозиторий на **GitHub** на основе предоставленного шаблона, затем создал локальную структуру каталогов для учебных проектов и клонировал удаленный репозиторий на свой компьютер.

1. Я перешел на страницу репозитория с шаблоном курса:  
<https://github.com/yamadharma/course-directory-student-template>.
2. Нажал на кнопку **Use this template**.
3. В открывшемся окне задал имя репозитория (**Repository name**) **study\_2025–2026\_arch-pc** и создал репозиторий, нажав кнопку **Create repository from template**.

Теперь создадим каталог и перейдем в него:

```
sgritsko@sgritsko:~$ mkdir -p ~/work/study/2025-2026/"Архитектура компьютера"
sgritsko@sgritsko:~$ cd ~/work/study/2025-2026/"Архитектура компьютера"
```

Рисунок 6. Новый каталог

Клонируем созданный удаленный репозиторий на свой компьютер, скопировав SSH-ссылку со страницы репозитория.

```
sgritsko@sgritsko:~/work/study/2025-2026/Архитектура компьютера$ git clone --recursive git@github.com:rudnstudent/study_2025-2026_arch-pc.git arch-pc
Клонирование в «arch-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 | 3.35 МБ/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/sgritsko/work/study/2025-2026/Архитектура компьютера/arch-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 | 8.07 МБ/с, готово.
Определение изменений: 100% (60/60), готово.
Клонирование в «/home/sgritsko/work/study/2025-2026/Архитектура компьютера/arch-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 | 3.58 МБ/с, готово.
Определение изменений: 100% (98/98), готово.
Submodule path 'template/presentation': checked out '6efd5c4ee78e4456caff3dc7062cfcad26058ca6'
Submodule path 'template/report': checked out '89a9622199b4df88227b9b3fa3d4714c85f68dd2'
sgritsko@sgritsko:~/work/study/2025-2026/Архитектура компьютера$
```

Рисунок 7. Клонирование

Сначала я создал репозиторий на **GitHub**, используя веб-интерфейс и готовый шаблон. Затем я создал локальную структуру папок на своем компьютере и клонировал туда удаленный репозиторий. Теперь у меня есть локальная копия проекта, которая связана с сервером на **GitHub**.

### 3.5 Настройка каталога курса

Завершающим этапом основной части работы была настройка структуры каталога курса с помощью `make` и отправка начальных файлов на сервер **GitHub**.

```
sgritsko@sgritsko:~$ cd ~/work/study/2025-2026/"Архитектура компьютера"/arch-pc
sgritsko@sgritsko:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ echo arch-pc > COURSE
sgritsko@sgritsko:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ make prepare
```

Рисунок 8. Подготовка файлов

Сейчас я перешел в каталог с проектом и выполнил подготовку структур.

```
sgritsko@sgritsko:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git add .
sgritsko@sgritsko:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git commit -am 'feat(main): make course structure'
[master a61bb76] feat(main): make course structure
212 files changed, 8074 insertions(+), 207 deletions(-)
```

Рисунок 9. Выполнение команд `add` и `commit`

```
sgritsko@sgritsko:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git push
Перечисление объектов: 67, готово.
Подсчет объектов: 100% (67/67), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (52/52), готово.
Запись объектов: 100% (64/64), 700.30 КиБ | 5.35 МБ/с, готово.
Всего 64 (изменений 22), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (22/22), completed with 1 local object.
To github.com:rudnstudent/study_2025-2026_arch-pc.git
  8a144b0..a61bb76 master -> master
```

Рисунок 10. Сжатие файлов

Я завершил настройку структуры курса, а затем использовал основной рабочий цикл Git: добавил изменения в индекс (**git add**), зафиксировал их с осмысленным комментарием (**git commit**) и отправил на удаленный сервер (**git push**). Теперь все мои локальные изменения синхронизированы с **GitHub**.

## Выполнение самостоятельной работы

Задание №1. Создайте отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства (labs/lab02/report).

```
sgritsko@sgritsko:~$ cd /home/sgritsko/work/study/2025-2026/Архитектура\ компьютера/arch-pc/labs/lab01/report
sgritsko@sgritsko:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab01/report$ git status
```

Рисунок 11. Переход в каталог

Я перешел в каталог для сдачи лабораторной работы.

Задание №2. Скопируйте отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданного рабочего пространства.

```
sgritsko@sgritsko:~$ cp ~/Загрузки/lab01_report.pdf labs/lab01/report/report.md
sgritsko@sgritsko:~$
```

Рисунок 12. Перемещение файла

В данном задании, я перенес лабораторную работу №1 в каталог, который меня просят.

Задание №3. Загрузите файлы на github.

```
sgritsko@sgritsko:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab01/report$ git add .
sgritsko@sgritsko:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab01/report$ git commit -m "feat(labs): add report for lab02 and copy lab01"
[master 70feb39] feat(labs): add report for lab02 and copy lab01
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 labs/lab01/report/lab01_report.pdf
sgritsko@sgritsko:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab01/report$ git push
Перечисление объектов: 10, готово.
Подсчет объектов: 100% (10/10), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 787.50 Киб | 5.58 Миб/с, готово.
Всего 6 (изменений 3), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To github.com:rudnstudent/study_2025-2026_arch-pc.git
a61bb76..70feb39 master -> master
sgritsko@sgritsko:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab01/report$
```

Рисунок 13. Выгрузка на github

Я выполнил самостоятельное задание. Создал файл для отчета по этой лабораторной работе и скопировал отчет по предыдущей. Все новые файлы я так же добавил, закоммитил и отправил на **GitHub**. Это закрепило мои практические навыки работы с основными командами **Git**.

## Выводы

В ходе выполнения данной лабораторной работы я изучил теоретические основы и получил практические навыки работы с системой контроля версий **Git**. Я научился выполнять базовую настройку **Git**, создавать и настраивать репозитории на **GitHub**, использовать **SSH-ключи** для безопасного соединения, а также освоил основной рабочий цикл: добавление изменений (add), их фиксацию (commit) и отправку на удаленный сервер (push). Цель работы была полностью достигнута.

1. Введение - О системе контроля версий/ <https://git-scm.com/book/ru/v2/%D0%92%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5-%D0%9E-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B5-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8F-%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D0%B9>
2. What is Git version control? / <https://about.gitlab.com/topics/version-control/what-is-git-version-control/>
3. Основные команды GIT / <https://habr.com/ru/articles/918386/>