

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: Леонов Никита Артемович

Группа: НКАбд-04-25

МОСКВА

2025 г.

Оглавление

Цель работы	3
1. Теоретическое введение	4
1.1. Системы контроля версий. Общие понятия	4
1.2. Система контроля версий Git.....	5
1.3. Стандартные процедуры работы при наличии центрального репозитория	6
2. Выполнение лабораторной работы.....	8
2.1. Настройка github.....	8
2.2. Базовая настройка git.....	8
2.3 Создание SSH-ключа	9
2.4 Создание рабочего пространства и репозитория курса на основе шаблона.....	10
2.5. Создание репозитория курса на основе шаблона	10
2.6 Настройка каталога курса	11
Вывод.....	13
Список литературы:	14

Цель работы

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

1. Теоретическое введение

1.1. Системы контроля версий. Общие понятия

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. 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.

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.

1.2. Система контроля версий Git

Система контроля версий Git представляет собой набор программ командной строки.

Основные команды git :

Таблица 1 Описание основных команд git

Команда	Описание
git init	создание основного дерева репозитория
git pull	получение обновлений (изменений) текущего дерева из центрального репозитория
git push	отправка всех произведённых изменений локального дерева в центральный репозиторий
git status	просмотр списка изменённых файлов в текущей директории
git diff	просмотр текущих изменений
git add .	добавить все изменённые и/или созданные файлы и/или каталоги

Команда	Описание
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

1.3. 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

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

2. Выполнение лабораторной работы

2.1. Настройка github

Создадим учётную запись на сайте <https://github.com/> и заполним основные данные:

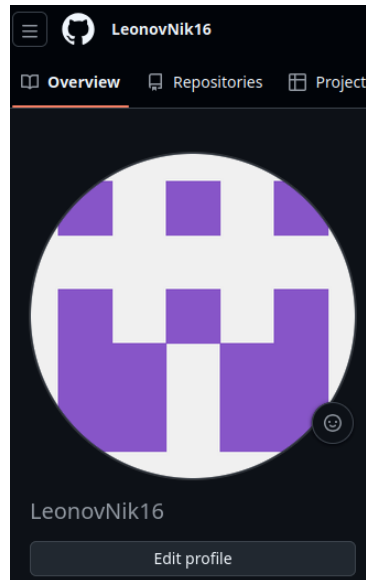


Рисунок 1 Созданный аккаунт в github

2.2. Базовая настройка git

Сначала сделаем предварительную конфигурацию **git**.

Откроем терминал и введём следующие команды, указав имя и e-mail владельца репозитория:

```
nleonov@nleonov:~$ git config --global user.name "<LeonovNik16>"
nleonov@nleonov:~$ git config --global user.email "<leonov16nik@gmail.com>"
```

Рисунок 2 Вводим данные

Настроим utf-8 в выводе сообщений **git**:

```
nleonov@nleonov:~$ git config --global core.quotePath false
```

Рисунок 3 Настройка utf-8

Зададим имя начальной ветки (будем называть её **master**):

```
nleonov@nleonov:~$ git config --global init.defaultBranch master
```

Рисунок 4 Задаём название начальной ветки

Параметр **autocrlf**:

```
nleonov@nleonov:~$ git config --global core.autocrlf input
```

Рисунок 5 Используем autocrlf

Параметр **safecrlf**:

```
nleonov@nleonov:~$ git config --global core.safecrlf warn
```

Рисунок 6 Используем safecrlf

2.3 Создание SSH-ключа

Создадим ключ с помощью команды :

ssh-keygen -C “Имя Фамилия <work@mail>”

```
nleonov@nleonov:~$ ssh-keygen -C "Leonov Nikita <leonov16nik@gmail.com>"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/nleonov/.ssh/id_ed25519):
/home/nleonov/.ssh/id_ed25519 already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/nleonov/.ssh/id_ed25519
Your public key has been saved in /home/nleonov/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:mMHcDACLnfUerGjbztoYeKNRUvcKDoEzQP+KbGbwlk Leonov Nikita <leonov16nik@gmail.com>
The key's randomart image is:
+--[ED25519 256]--+
|o..+oo..          |
|oo.o = +          |
|* o.. B o         |
| = o.+ =          |
|+ = ..= S         |
|oO.*..           |
|E*@.o            |
|+* B             |
|. o.+            |
+-----[SHA256]-----+
```

Рисунок 7 Создание ключа

Копируем из локальной консоли ключ в буфер обмена,
Вставляем его в появившееся на сайте поле и указываем для
ключа имя (Title).

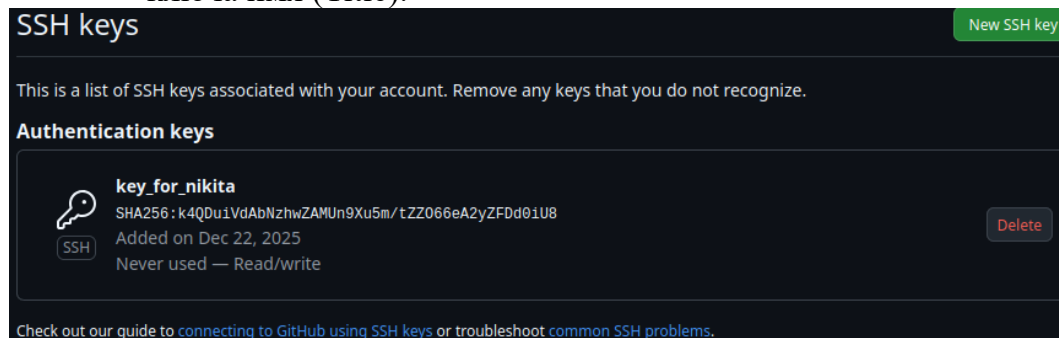


Рисунок 8 Ключ

2.4 Создание рабочего пространства и репозитория курса на основе шаблона

Откроем терминал и создадим каталог для предмета
«Архитектура компьютера»:

```
mkdir -p ~/work/study/2025-2026/"Архитектура компьютера"
```

```
nleonov@nleonov:~$ mkdir -p ~/work/study/2025-2026/"Arhitektura computerov"
```

Рисунок 9 Создание рабочего пространства

2.5. Создание репозитория курса на основе шаблона

Репозиторий можно создать через web-интерфейс
github.

Перейдём на страницу репозитория с шаблоном курса:
<https://github.com/yamadharm/course-directory-student-template>.

Далее выберем : **Use this template** .

В открывшемся окне зададим имя репозитория
(Repository name) **study_2025–2026_arh-pc** и создадим
репозиторий:

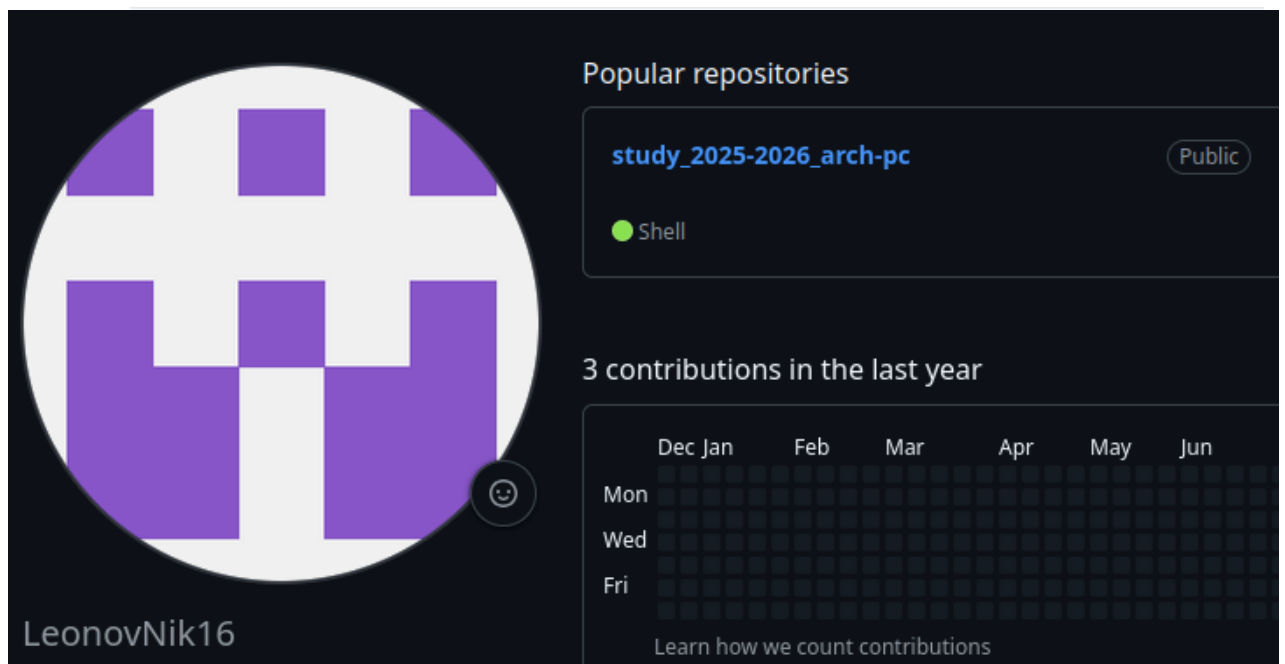


Рисунок 10 Создание репозитория

Теперь откроем терминал и перейдем в каталог курса с помощью команды **cd** и скопируем созданный репозиторий с

ПОМОЩЬЮ КОМАНДЫ

```
nleonov@nleonov:~/study_2025-2026_arch-pc$ git clone --recursive https://github.com/LeonovNik16/study_2025-2026_arch-pc.git
Cloning into 'study_2025-2026_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 26 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (38/38), 23.57 KiB | 431.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.
Submodule 'template/presentation' (https://github.com/yamadharm/academic-presentation-markdown-template.git) registered for path 'template/presentation'
Submodule 'template/report' (https://github.com/yamadharm/academic-laboratory-report-template.git) registered for path 'template/report'
Cloning into '/home/nleonov/study_2025-2026_arch-pc/study_2025-2026_arch-pc/template/presentation'...
remote: Enumerating objects: 219, done.
remote: Counting objects: 100% (219/219), done.
remote: Compressing objects: 100% (151/151), done.
remote: Total 219 (delta 86), reused 189 (delta 56), pack-reused 0 (from 0)
Receiving objects: 100% (219/219), 2.66 MiB | 5.58 MiB/s, done.
Resolving deltas: 100% (86/86), done.
Cloning into '/home/nleonov/study_2025-2026_arch-pc/study_2025-2026_arch-pc/template/report'...
remote: Enumerating objects: 251, done.
remote: Counting objects: 100% (251/251), done.
remote: Compressing objects: 100% (172/172), done.
remote: Total 251 (delta 111), reused 204 (delta 64), pack-reused 0 (from 0)
Receiving objects: 100% (251/251), 775.12 KiB | 2.72 MiB/s, done.
Resolving deltas: 100% (111/111), done.
Submodule path 'template/presentation': checked out '1c93acf9e731bf186384c85de4aff70037314240'
Submodule path 'template/report': checked out '8ee157c58b3362947b1c71492a65d4dc6882d5ad'
nleonov@nleonov:~/study_2025-2026_arch-pc$
```

Рисунок 11 Копирование репозитория

2.6 Настройка каталога курса

Перейдем в каталог курса с помощью команды **cd**, создадим необходимые каталоги(

echo arch-pc > COURSE

make prepare

)

и отправим файлы на сервер (

git add .

git commit -am 'feat(main): make course structure'

git push

)

```
create mode 100644 study_2025-2026_arch-pc/template/presentation/presentation/.marksman.toml
create mode 100644 study_2025-2026_arch-pc/template/presentation/presentation/.projectile
create mode 100644 study_2025-2026_arch-pc/template/presentation/presentation/Makefile
create mode 100644 study_2025-2026_arch-pc/template/presentation/presentation/_assets/auto/beamer.el
create mode 100644 study_2025-2026_arch-pc/template/presentation/presentation/_assets/beamer.tex
create mode 100644 study_2025-2026_arch-pc/template/presentation/presentation/_quarto.yml
create mode 100644 study_2025-2026_arch-pc/template/presentation/presentation/_resources/image/logo_rudn.png
create mode 100644 study_2025-2026_arch-pc/template/presentation/presentation/image/kulyabov.jpg
create mode 100644 study_2025-2026_arch-pc/template/presentation/presentation/presentation.qmd
create mode 100644 study_2025-2026_arch-pc/template/report/.gitattributes
create mode 100644 study_2025-2026_arch-pc/template/report/.gitignore
create mode 100644 study_2025-2026_arch-pc/template/report/CHANGELOG.md
create mode 100644 study_2025-2026_arch-pc/template/report/LICENSE
create mode 100644 study_2025-2026_arch-pc/template/report/README.git-flow.md
create mode 100644 study_2025-2026_arch-pc/template/report/README.md
create mode 100644 study_2025-2026_arch-pc/template/report/README.ru.md
create mode 100644 study_2025-2026_arch-pc/template/report/package.json
create mode 100644 study_2025-2026_arch-pc/template/report/report/.gitignore
create mode 100644 study_2025-2026_arch-pc/template/report/report/.marksman.toml
create mode 100644 study_2025-2026_arch-pc/template/report/report/.projectile
create mode 100644 study_2025-2026_arch-pc/template/report/report/Makefile
create mode 100644 study_2025-2026_arch-pc/template/report/report/_assets/preamble.tex
create mode 100644 study_2025-2026_arch-pc/template/report/report/_quarto.yml
create mode 100644 study_2025-2026_arch-pc/template/report/report/_resources/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 study_2025-2026_arch-pc/template/report/report/bib/cite.bib
create mode 100644 study_2025-2026_arch-pc/template/report/report/image/solvay.jpg
create mode 100644 study_2025-2026_arch-pc/template/report/report/report.qmd
create mode 100755 study_2025-2026_arch-pc/template/report/scripts/image-report
create mode 100755 study_2025-2026_arch-pc/template/report/scripts/mpv-shot
create mode 100644 work/study/2025-2026/arch-pc/COURSE
create mode 160000 work/study/2025-2026/arch-pc/study_2025-2026_arch-pc
leonyov@leonyov: ~/study_2025-2026_arch-pc$
```

Рисунок 12 Настройка каталога курса

Вывод

В ходе выполнения лабораторной работы успешно достигнул поставленной цели по изучению идеологии и практическому освоению системы контроля версий Git. Были приобретены следующие ключевые навыки:

- Освоена работа с базовыми командами Git: **git add, git commit, git status;**
- Приобретен опыт работы с удаленными репозиториями: **git clone, git push, git pull;**
- Приобретен навык работы с хостингом **Git-репозитория** (GitHub).

Список литературы:

- 1) Основные команды git : <https://git-scm.com/book/ru/v2/Приложение-C%3A-Команды-Git-Основные-команды>
- 2) Настройка Git и создание SSH ключа : <https://htmlacademy.ru/blog/git/git-console>
- 3) Создание репозитория : <https://docs.github.com/ru/repositories/creating-and-managing-repositories/creating-a-new-repository>