

# **Лабораторная работа №2**

**дисциплина “Архитектура компьютера”**

Комаров Владимир Артемович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
<b>5</b>	<b>Вывод</b>	<b>17</b>

# Список иллюстраций

4.1	Базовая настройка Git	10
4.2	Создание SSH-ключа	10
4.3	Новый SSH-ключ	10
4.4	Создание директории	11
4.5	Готовый репозиторий	11
4.6	Копирование репозитория	12
4.7	Создание и удаление файлов и директорий	12
4.8	Команда <code>make</code> и создание коммита	13
4.9	Загрузка коммита на сервер	13
4.10	Репозиторий гитхаб	14
4.11	Создание файла отчёта	14
4.12	Проверка расположения файлов	15
4.13	Копирование файла	15
4.14	Коммичу изменения	15
4.15	Загрузка данных на сервер	15
4.16	Лаба 1	16

## **Список таблиц**

# 1 Цель работы

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

## 2 Задание

1. Настройка GitHub.
2. Базовая 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.
7. Выполнение заданий для самостоятельной работы.

### 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 локальное дерево и/или ветке. После завершения внесения какого-то изменения в файлы и/или каталоги проекта необходимо разместить их в центральном репозитории.



## 4 Выполнение лабораторной работы

1. Настройка Github На гитхабе у меня уже была учётная запись и поэтому создавать новую мне не пришлось.
2. Базовая настройка 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. Настраиваю utf-8 в выводе сообщений git для корректного отображения символов. 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. 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. При значении `warn` Git только выведет предупреждение, но будет принимать необратимые конвертации.

```
(vakomarov@vakomarov)~$ git config --global user.name "kerfarion"
(vakomarov@vakomarov)~$ git config --global user.email "vladimirkomarov4323@gmail.com"
(vakomarov@vakomarov)~$ git config --global core.quotepath false
(vakomarov@vakomarov)~$ git config --global init.defaultBranch master
(vakomarov@vakomarov)~$ git config --global core.autocrlf input
(vakomarov@vakomarov)~$ git config --global core.safecrlf warn
```

Рис. 4.1: Базовая настройка Git

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. 4.2). Ключ автоматически сохранится в каталоге `~/.ssh/`.

```
(vakomarov@vakomarov)~$ ssh-keygen -C "Комаров Владимир <vladimirkomarov4323@gmail.com>"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/vakomarov/.ssh/id_ed25519):
Created directory /home/vakomarov/.ssh/.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/vakomarov/.ssh/id_ed25519
Your public key has been saved in /home/vakomarov/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:kJAinslhgyXpORBI1ijPchhK3xK9tnpimL4KRsktuM Комаров Владимир <vladimirkomarov4323@gmail.com>
The key's randomart image is:
+--[ED25519 256]--+
|BO*+o.           |
|BOB+..           |
|+==o.  o         |
|..+o.  o         |
|lo*..  S         |
|*..+ o.  o       |
|o+ o o          |
|+o o           |
|+Eo ..         |
+--[SHA256]-----+
(vakomarov@vakomarov)~$ cat ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIDjTNxdxAV6qx2PrfP4t5Y67ozu3357offpcAMbv2xYH Комаров Владимир <vladimirkomarov4323@gmail.com>
```

Рис. 4.2: Создание SSH-ключа

Копирую ключ из открытой директории в которой он был, перехожу на сайт Github и добавляю там новый SSH-ключ

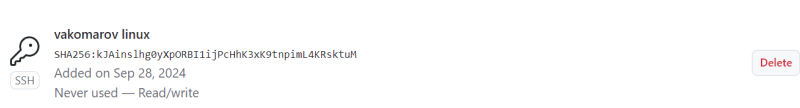
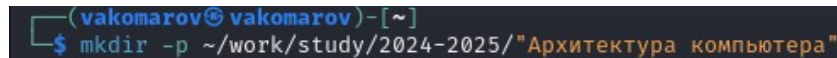


Рис. 4.3: Новый SSH-ключ

4. Создание рабочего пространства и репозитория курса на основе шаблона  
на Закрываю браузер, открываю терминал. Создаю директорию, рабочее пространство, с помощью утилиты `mkdir`, благодаря ключу `-p` создаю все директории после домашней `~/work/study/2022-2023/“Архитектура компьютера”` рекурсивно.



```
(vakomarov@vakomarov) ~  
$ mkdir -p ~/work/study/2024-2025/\"Архитектура компьютера\"
```

Рис. 4.4: Создание директории

5. Создание репозитория курса на основе шаблона В браузере перехожу на страницу репозитория с шаблоном курса по адресу <https://github.com/yamadharma/course-directory-student-template>. Далее выбираю «Use this template», чтобы использовать этот шаблон для своего репозитория. В открывшемся окне задаю имя репозитория (Repository name): `study_2022– 2023_arh- pc` и создаю репозиторий, нажимаю на кнопку «Create repository from template».

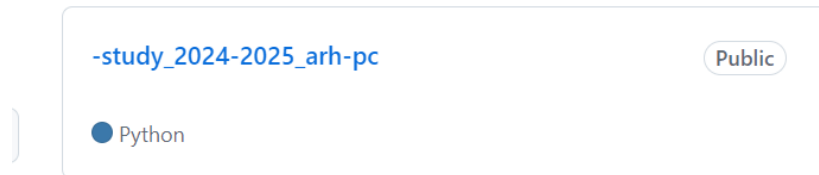


Рис. 4.5: Готовый репозиторий

Через терминал перехожу в созданный каталог курса и клонирую созданный репозиторий с помощью команды `git clone –recursive git@github.com:/study_2024–2025_arh-pc.git arch-pc` (рис. 4.6)

```

(vakomarov@vakomarov:~)$ cd ~/work/study/2024-2025/Архитектура компьютера
(vakomarov@vakomarov:~)$ cd ~/work/study/2024-2025/Архитектура компьютера
$ git clone --recursive git@github.com:kerfaron/study_2024-2025_arh-pc.git arch-pc
Cloning into 'arch-pc'...
The authenticity of host 'github.com (140.82.121.4)' can't be established.
ED25519 key fingerprint is SHA256:DIY3wvW6TUj2hg213rZLDAA9v5wHdkr4VUCoQU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
remote: Enumerating objects: 34, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (33/33), done.
remote: Total 34 (delta 1), reused 18 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (24/24), 19.58 KiB | 3.26 MiB/s, done.
Resolving deltas: 100% (1/1), done.
Submodule 'template/presentation' (https://github.com/yamadaharua/academic-presentation-markdown-template.git) registered for path 'template/presentation'
Cloning into 'template/presentation':
Cloning into '/home/vakomarov/work/study/2024-2025/Архитектура компьютера/arch-pc/template/presentation'...
remote: Enumerating objects: 111, done.
remote: Counting objects: 100% (111/111), done.
remote: Compressing objects: 100% (77/77), done.
remote: Total 111 (delta 42), reused 108 (delta 31), pack-reused 0 (from 0)
Receiving objects: 100% (111/111), 102.17 KiB | 145.00 KiB/s, done.
Resolving deltas: 100% (42/42), done.
Cloning into '/home/vakomarov/work/study/2024-2025/Архитектура компьютера/arch-pc/template/report'...
remote: Enumerating objects: 142, done.
remote: Counting objects: 100% (142/142), done.
remote: Compressing objects: 100% (97/97), done.
remote: Total 142 (delta 60), reused 121 (delta 39), pack-reused 0 (from 0)
Receiving objects: 100% (142/142), 341.09 KiB | 260.00 KiB/s, done.
Resolving deltas: 100% (60/60), done.
Submodule path 'template/presentation': checked out 'c9b2722bb4b2d413d5866c7a2b28dfcda6d'
Submodule path 'template/report': checked out 'c2e27ef7e2b3455782d9e518546e18547f8'

```

Рис. 4.6: Копирование репозитория

6. Настройка каталога курса Перехожу в каталог `arch-rc` с помощью утилиты `cd` Удаляю лишние файлы с помощью утилиты `rm` Создаю необходимые каталоги Отправляю созданные каталоги с локального репозитория на сервер: добавляю все созданные каталоги с помощью `git add`, комментирую и сохраняю изменения на сервере как добавление курса с помощью `git commit` (рис. 4.7, 4.8).

```
(vakomarov@vakomarov)-[~/work/study/2024-2025/Архитектура компьютера]
$ cd ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc

(vakomarov@vakomarov)-[~/.../study/2024-2025/Архитектура компьютера/arch-pc]
$ rm package.json

(vakomarov@vakomarov)-[~/.../study/2024-2025/Архитектура компьютера/arch-pc]
$ echo arch-pc > COURSE
```

Рис. 4.7: Создание и удаление файлов и директорий

```

(vakomarov@vakomarov)-[~/../study/2024-2025/Архитектура компьютера/arch-pc]
$ make prepare

(vakomarov@vakomarov)-[~/../study/2024-2025/Архитектура компьютера/arch-pc]
$ ls
CHANGELOG.md  COURSE  LICENSE  Makefile  README.en.md  README.git-flow.md  README.md  config  labs  prepare  presentation  template

(vakomarov@vakomarov)-[~/../study/2024-2025/Архитектура компьютера/arch-pc]
$ git add .

(vakomarov@vakomarov)-[~/../study/2024-2025/Архитектура компьютера/arch-pc]
$ git commit -am 'feat(main): make course structure'
[master d08596f] feat(main): make course structure
221 files changed, 53600 insertions(+)
create mode 100644 labs/README.md
create mode 100644 labs/README.ru.md
create mode 100644 labs/lab01/presentation/.projectile
create mode 100644 labs/lab01/presentation/textlabroot
create mode 100644 labs/lab01/presentation/Makefile
create mode 100644 labs/lab01/presentation/image/kulyabov.jpg
create mode 100644 labs/lab01/presentation/presentation.md
create mode 100644 labs/lab01/report/bib/cite.bib
create mode 100644 labs/lab01/report/image/placeimg_600_600_tech.jpg
create mode 100644 labs/lab01/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_fignos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_secnos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/_init_.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/core.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/main.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 labs/lab01/report/report.md
create mode 100644 labs/lab02/presentation/.projectile
create mode 100644 labs/lab02/presentation/textlabroot
create mode 100644 labs/lab02/presentation/Makefile
create mode 100644 labs/lab02/presentation/image/kulyabov.jpg
create mode 100644 labs/lab02/presentation/presentation.md
create mode 100644 labs/lab02/report/bib/cite.bib
create mode 100644 labs/lab02/report/image/placeimg_600_600_tech.jpg
create mode 100644 labs/lab02/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_fignos.py
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_secnos.py
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/_init_.py
create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/core.py

```

Рис. 4.8: Команда make и создание коммита

С помощью команды push выгружаем всё на сервер

```

(vakomarov@vakomarov)-[~/../study/2024-2025/Архитектура компьютера/arch-pc]
$ git push
Enumerating objects: 39, done.
Counting objects: 100% (39/39), done.
Delta compression using up to 4 threads
Compressing objects: 100% (31/31), done.
Writing objects: 100% (37/37), 341.47 KiB | 533.00 KiB/s, done.
Total 37 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), completed with 1 local object.
To github.com:kerfarion/-study_2024-2025_arh-pc.git
a1d551b..d08596f master -> master

```

Рис. 4.9: Загрузка коммита на сервер

Проверяю правильность выполнения работы сначала на самом сайте GitHub.

Name	Last commit message	Last commit date
..		
lab01	feat(main): make course structure	2 minutes ago
lab02	feat(main): make course structure	2 minutes ago
lab03	feat(main): make course structure	2 minutes ago
lab04	feat(main): make course structure	2 minutes ago
lab05	feat(main): make course structure	2 minutes ago
lab06	feat(main): make course structure	2 minutes ago
lab07	feat(main): make course structure	2 minutes ago
lab08	feat(main): make course structure	2 minutes ago
lab09	feat(main): make course structure	2 minutes ago
lab10	feat(main): make course structure	2 minutes ago
lab11	feat(main): make course structure	2 minutes ago
README.md	feat(main): make course structure	2 minutes ago

Рис. 4.10: Репозиторий гитхаб

7. Выполнение заданий для самостоятельной работы Перехожу в директорию `labs/lab03/report` с помощью утилиты `cd`. Создаю в каталоге файл для отчета по третьей лабораторной работе с помощью утилиты `touch`, после чего перехожу в директорию `labs/lab01/report`(рис. 4.11).

```
(vakomarov@vakomarov) ~/.../study/2024-2025/Архитектура компьютера/arch-pc
$ cd labs/lab02/report

(vakomarov@vakomarov) ~/.../arch-pc/labs/lab02/report
$ touch Л02_Комаров_отчет

(vakomarov@vakomarov) ~/.../arch-pc/labs/lab02/report
$ cd ..

(vakomarov@vakomarov) ~/.../Архитектура компьютера/arch-pc/labs/lab02
$ cd ..

(vakomarov@vakomarov) ~/.../2024-2025/Архитектура компьютера/arch-pc/labs
$ cd lab01/report
```

Рис. 4.11: Создание файла отчёта

Проверяю местонахождение файла с отчетом по первой лабораторной работой. Он должен быть в подкаталоге домашней директории «Загрузки», для проверки использую команду `ls`

```
(vakomarov@vakomarov) - [~/../arch-pc/labs/lab01/report]
$ ls ~/Downloads
Л01_Комаров_отчет.pdf
```

Рис. 4.12: Проверка расположения файлов

Копирую первую лабораторную с помощью утилиты `cp` и проверяю правильность выполнения команды `cp` с помощью `ls`

```
(vakomarov@vakomarov) - [~/../arch-pc/labs/lab01/report]
$ cp ~/Downloads/Л01_Комаров_отчет.pdf /home/vakomarov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab01/report
(vakomarov@vakomarov) - [~/../arch-pc/labs/lab01/report]
$ ls
Makefile bib image pandoc report.md Л01_Комаров_отчет.pdf
```

Рис. 4.13: Копирование файла

После чего добавляю файл в `add` Сохраняю изменения на сервере командой `git commit -m "..."`, поясняя, что добавил файлы.

```
(vakomarov@vakomarov) - [~/../arch-pc/labs/lab01/report]
$ git add Л01_Комаров_отчет.pdf
(vakomarov@vakomarov) - [~/../arch-pc/labs/lab01/report]
$ git commit -m "Add lab01/report"
[master 7c8b9f2] Add lab01/report
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 "labs/lab01/report/320\23381_\320\232\320\276\320\274\320\260\321\200\320\276\320\262_\320\276\321\202\321\207\320\265\321\202_.pdf"
(vakomarov@vakomarov) - [~/../arch-pc/labs/lab01/report]
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

Untracked files:
  (use "git add <file> ..." to include in what will be committed)
  ../../lab01/report/320\23382_\320\232\320\276\320\274\320\260\321\200\320\276\320\262_\320\276\321\202\321\207\320\265\321\202_.pdf
nothing added to commit but untracked files present (use "git add" to track)
```

Рис. 4.14: Коммичу изменения

Аналогично я поступаю с отчётом по этой лабораторной работе, после чего отправляю коммит на сервер.

```
(vakomarov@vakomarov) - [~/../study/2024-2025/Архитектура компьютера/arch-pc]
$ git push
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.21 MiB | 1.68 MiB/s, done.
Total 6 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To github.com:kerfarion/-study_2024-2025_arh-pc.git
d08596f..7c8b9f2 master -> master
```

Рис. 4.15: Загрузка данных на сервер

Вижу, что отчеты по лабораторным работам находятся в соответствующих каталогах репозитория: отчет по первой - в `lab01/report`

study\_2024-2025\_arh-pc / labs / lab01 / report /

kerfarion Add lab01/report 7c8b9f2 - 1 hour ago

Name	Last commit message	
..		
bib	feat(main): make course structure	
image	feat(main): make course structure	
pandoc	feat(main): make course structure	
Makefile	feat(main): make course structure	
report.md	feat(main): make course structure	
/01_Комаров_отчет.pdf	Add lab01/report	

Рис. 4.16: Лаба 1



## **5 Вывод**

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