

# **Отчёт по лабораторной работе №2**

**Дисциплина: Архитектура компьютеров**

Филиппева Ксения Дмитриевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Создание SSH ключа</b>	<b>10</b>
<b>6</b>	<b>Создание рабочего пространства и репозитория курса на основе шаблона</b>	<b>12</b>
<b>7</b>	<b>Настройка каталога курса</b>	<b>14</b>
<b>8</b>	<b>Выполнение заданий для самостоятельной работы.</b>	<b>16</b>
<b>9</b>	<b>Выводы</b>	<b>18</b>

# Список иллюстраций

4.1	Имя и почта пользователя . . . . .	8
4.2	Вывод utf-8 и начальная ветка “master” . . . . .	8
4.3	Параметры “autocrlf” и “safecrlf” . . . . .	9
5.1	Команда «ssh-keygen» и сам ключ . . . . .	10
5.2	Команда “cat” . . . . .	11
5.3	Ключ на Github . . . . .	11
6.1	Команда “mkdir” . . . . .	12
6.2	Команда “cd” с переходом к каталогу . . . . .	12
6.3	Использование команды «git clone» . . . . .	13
7.1	Удаление файла «package.json» . . . . .	14
7.2	Команды «echo» и «make» . . . . .	14
7.3	Команда “git commit” . . . . .	15
7.4	Команда “git push” . . . . .	15
8.1	lab02 на Github . . . . .	16
8.2	lab01 на Github . . . . .	17

# 1 Цель работы

Целью данной работы является изучить идеологию и применение средств контроля версий, а также освоить умения по работе с github.

## 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. Задание 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 (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. Системы

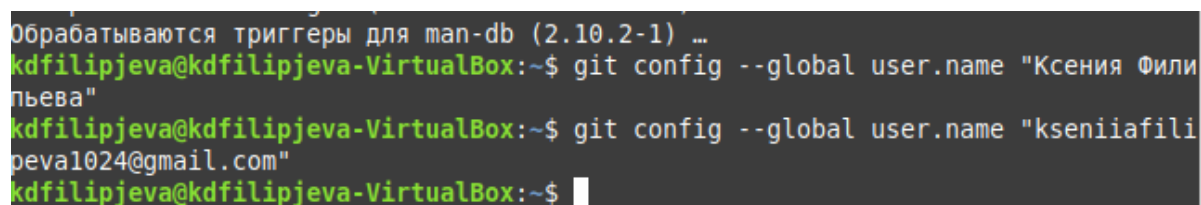
контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. В отличие от классических, 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.

## 4 Выполнение лабораторной работы

### Настройка GitHub

Поскольку учетная запись на Github уже имеется, регистрировать ее нет необходимости.

Базовая настройка git Откроем терминал и введем следующие команды с указанием имени и фамилии, а также нашей электронной почты. (рис. 4.1)



```
Обрабатываются триггеры для man-db (2.10.2-1) ...
kdfilipjeva@kdfilipjeva-VirtualBox:~$ git config --global user.name "Ксения Филиппева"
kdfilipjeva@kdfilipjeva-VirtualBox:~$ git config --global user.email "kseniiafilipjeva1024@gmail.com"
kdfilipjeva@kdfilipjeva-VirtualBox:~$
```

Рис. 4.1: Имя и почта пользователя

Настроим вывод utf-8 в выводе сообщений git. Так же зададим имя начальной ветки – «master». (рис. 4.2)



```
kdfilipjeva@kdfilipjeva-VirtualBox:~$ git config --global core.quotePath false
kdfilipjeva@kdfilipjeva-VirtualBox:~$ git config --global init.defaultBranch master
kdfilipjeva@kdfilipjeva-VirtualBox:~$
```

Рис. 4.2: Вывод utf-8 и начальная ветка “master”

Так же подключим параметры «autocrlf» и «safecrlf». (рис. 4.3)



```
kdfilipjeva@kdfilipjeva-VirtualBox:~$ git config --global core.autocrlf input
kdfilipjeva@kdfilipjeva-VirtualBox:~$ git config --global core.safecrlf warn
kdfilipjeva@kdfilipjeva-VirtualBox:~$
```

Рис. 4.3: Параметры “autocrlf” и “safecrlf”

## 5 Создание SSH ключа

Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый). Для этого используем команду «ssh-keygen». (рис. 5.1). После генерации ключ сохраняется в каталоге ~/.ssh/.

```
The key's randomart image is:
+---[RSA 3072]---+
|                 +. =+. |
|                 + +. += |
|                 . = BE= |
|                 + 0+++ |
|      S .  ++*0. |
|                 +  ==+=0 |
|                 . .+ 0+0B |
|                 .  0+0 |
|                 . |
+---[SHA256]-----+
kdfilipjeva@kdfilipjeva-VirtualBox:~$
```

Рис. 5.1: Команда «ssh-keygen» и сам ключ

Скопируем получившийся ключ с помощью команды «cat» и загрузим его в

наш аккаунт Github, указав имя для этого ключа.(рис. 5.2)(рис. 5.3)

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


Рис. 5.2: Команда “cat”

## SSH keys

[New SSH key](#)

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

### Authentication Keys

**ssh key**  
SHA256: 21TSex7CR2Y2sF9wbJqeaLb02TU/tLRVtVIE0F2+Upc  
Added on Oct 12, 2023  
Never used — Read/write

Delete

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

Рис. 5.3: Ключ на Github

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

При выполнении лабораторных работ следует соблюдать определенную

иерархию, которая у нас и соблюдается. Создадим каталог для предмета «Архитектура компьютера» с помощью терминала.(рис. 6.1)

```
kdfilipjeva@kdfilipjeva-VirtualBox:~$ mkdir -p ~/work/study/2023-2024/"Архитектура компьютера"
kdfilipjeva@kdfilipjeva-VirtualBox:~$
```

Рис. 6.1: Команда “mkdir”

### Создание репозитория курса на основе шаблона

Перейдя на страницу с шаблоном курса (<https://github.com/yamadharm/course-directory-student-template>)

создадим репозиторий и присвоим ему имя. (рис. 6.2)

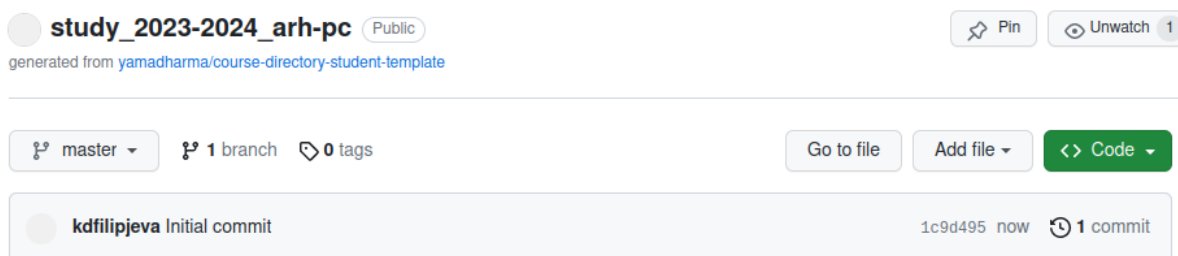
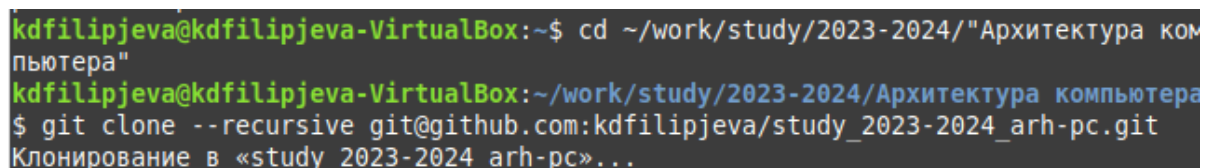


Рис. 6.2: Команда “cd” с переходом к каталогу

Используя терминал перейдем к каталогу курса. Клонировать данный репозиторий используя команду `git clone --recursive`

предварительно скопировав ссылку для клонирования в нашем личном кабинете за счет SSH-ключа.(рис. 6.3)



```
kdfilipjeva@kdfilipjeva-VirtualBox:~$ cd ~/work/study/2023-2024/"Архитектура компьютера"
kdfilipjeva@kdfilipjeva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера$ git clone --recursive git@github.com:kdfilipjeva/study_2023-2024_arh-pc.git
Клонирование в «study 2023-2024 arh-pc»...
```

Рис. 6.3: Использование команды «git clone»

## 7 Настройка каталога курса

Перейдем в каталог курса используя команду «cd» и удалим файл

«package.json» используя команду «rm». (рис. 7.1)

```
kdfilipjeva@kdfilipjeva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера
$ cd ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc
kdfilipjeva@kdfilipjeva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера
/arch-pc$ rm package.json
kdfilipjeva@kdfilipjeva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера
/arch-pc$
```

Рис. 7.1: Удаление файла «package.json»

Создаем необходимые нам каталоги в репозитории используя команду «echo»

и «make». (рис. 7.2).

```
kdfilipjeva@kdfilipjeva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера
/arch-pc$ echo arch-pc > COURSE
kdfilipjeva@kdfilipjeva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера
/arch-pc$ make
kdfilipjeva@kdfilipjeva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера
/arch-pc$
```

Рис. 7.2: Команды «echo» и «make»

После всех сделанных действий отправляем файлы на сервер используя

череду команд «git add», «git commit» и «git push». (рис. 7.3, рис. 7.4).

```
kdfilipjeva@kdfilipjeva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера
/arch-pc$ git commit -am 'feat(main): make course structure'
[master 88c3d3d] feat(main): make course structure
199 files changed, 54725 insertions(+), 14 deletions(-)
```

Рис. 7.3: Команда “git commit”

```
kdfilipjeva@kdfilipjeva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера
/arch-pc$ git push
Перечисление объектов: 37, готово.
Подсчет объектов: 100% (37/37), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (29/29), готово.
Запись объектов: 100% (35/35), 342.17 КиБ | 1.92 МиБ/с, готово.
Всего 35 (изменений 4), повторно использовано 0 (изменений 0), повторно использо
вано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:kdfilipjeva/study_2023-2024_arh-pc.git
  1c9d495..88c3d3d master -> master
kdfilipjeva@kdfilipjeva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера
/arch-pc$
```

Рис. 7.4: Команда “git push”

## 8 Выполнение заданий для самостоятельной работы.

Отчет по проведенной лабораторной работе создан и загружен в

(labs>lab02>report). (рис. 8.1). Так же предыдущая лабораторная работа была загружена в (labs>lab01>report). (рис. 8.2).

[study\\_2023-2024\\_arh-pc](#) / [labs](#) / [lab02](#) / **report** / 










 <b>kdfilipjeva</b> feat(main): make course structure	
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
 ЛР02_Филиппева_отчет.pdf	feat(main): make course structure

Рис. 8.1: lab02 на Github



 **kdfilipjeva** feat(main): make course structure








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
 ЛР01_Филипьева_отчет.pdf	feat(main): make course structure

Рис. 8.2: lab01 на Github

## 9 Выводы

В ходе лабораторной работы я изучила идеологию и применение средств контроля версий и приобрела практические навыки по работе с системой git.