

## Лабораторная работа №3

Выполнил Илья Егорович Тайц

Группа НКАбд-02 22

### Содержание

1 Цель работы

2 Задание

3 Теоретическое введение

4 Выполнение лабораторной работы

5 Выводы

6Список литературы

7Список иллюстраций #было решено использовать иллюстрации в части 4 для большего удобства восприятия и наглядности пояснений.

## 1 Цель работы

Целью работы является изучить идеологию и применение средств контроля версий. Приобрести практические навыки по работе с системой git.

## 2 Задание

Задания для лабораторной работы

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 2
6. Настройка каталога курса

Задания для самостоятельной работы

1. Создайте отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства (labs>lab03>report).
2. Скопируйте отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданного рабочего пространства.
3. Загрузите файлы на github.

## 3 Теоретическое введение

### 1. Системы контроля версий. Общие понятия

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. 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. Кроме того, 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.

## 2. Система контроля версий Git

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.

### 3. Основные команды git

Наиболее часто используемые команды git представлены в таблице 3.1

Таблица 3.1. Основные команды 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 изменения
git add .	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 branch (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

#### 4. 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`

и 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

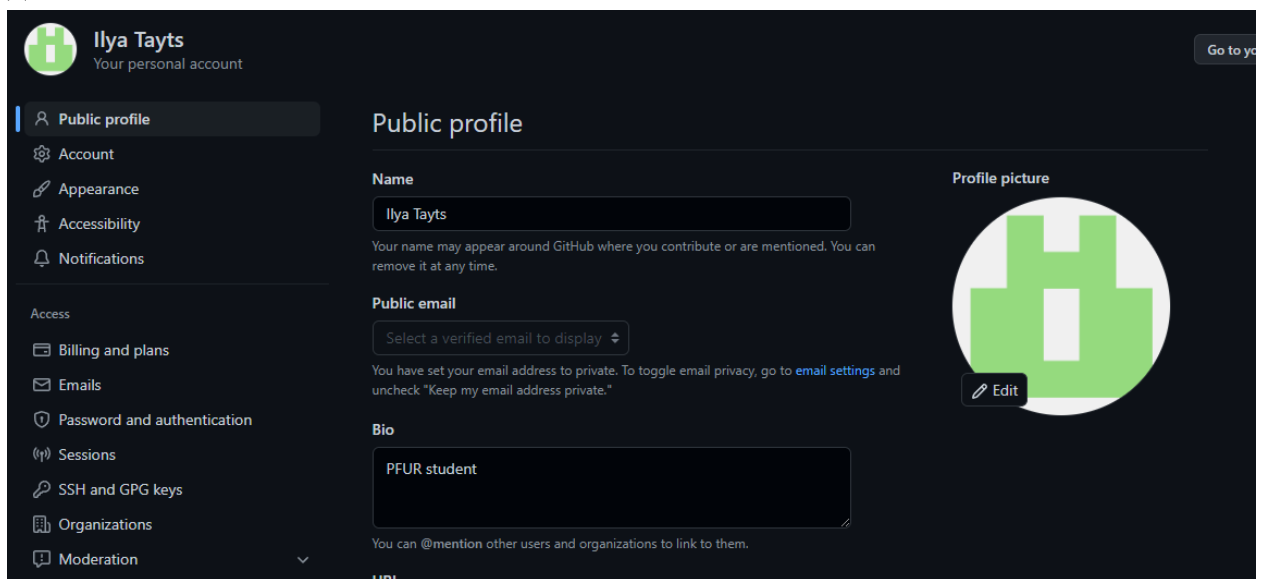
```
git push
```

## 4 Выполнение лабораторной работы

### Задания для лабораторной работы

#### 1. Настройка github

Создайте учётную запись на сайте <https://github.com/> и заполните основные данные.



Не уверен какие данные являются основными но учётная запись сделана.

#### 2. Базовая настройка git

задания и код:

Откройте терминал и введите следующие команды, указав имя и email владельца репозитория:

```
git config --global user.name "<Name Surname>"
```

```
git config --global user.email "<work@mail>"
```

Настроим utf-8 в выводе сообщений git:

```
git config --global core.quotePath false
```

Зададим имя начальной ветки (будем называть её master):

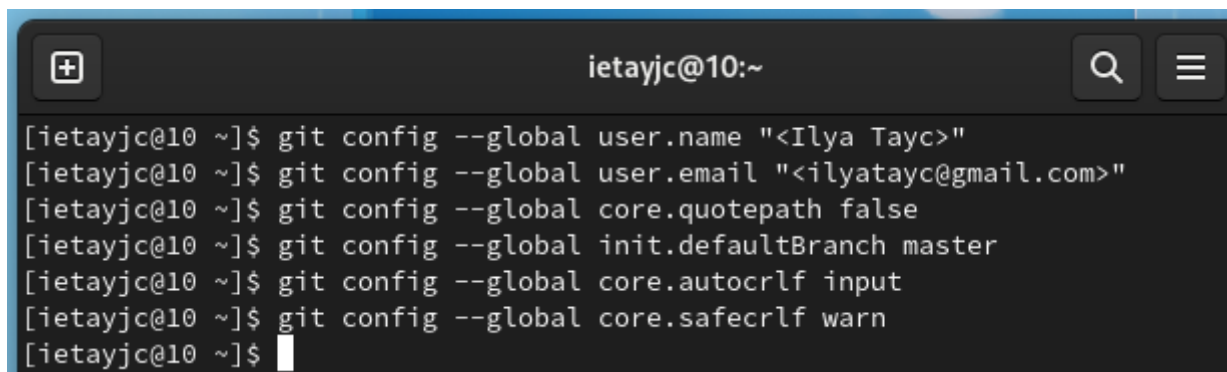
```
git config --global init.defaultBranch master
```

Параметр autocrlf:

```
git config --global core.autocrlf input
```

Параметр safecrlf:

```
git config --global core.safecrlf warn
```

A screenshot of a terminal window with a dark background. The window title is 'ietayjc@10:~'. The terminal shows a series of git config commands being executed. The prompt is '[ietayjc@10 ~]\$' and the commands are: 'git config --global user.name "<Ilya Tayc>"', 'git config --global user.email "<ilyatayc@gmail.com>"', 'git config --global core.quotePath false', 'git config --global init.defaultBranch master', 'git config --global core.autocrlf input', and 'git config --global core.safecrlf warn'. The last command is followed by a new prompt '[ietayjc@10 ~]\$' and a cursor.

Настройки заданы, можно приступать.

### 3. Создание SSH ключа

Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый):

```
ssh-keygen -C "Имя Фамилия <work@mail>"
```

Ключи сохраняться в каталоге ~/.ssh/.

```

[ietayjc@10 ~]$ ssh-keygen -C "Ilya Tayc <ilyatayc@gmail.com>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ietayjc/.ssh/id_rsa):
Created directory '/home/ietayjc/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ietayjc/.ssh/id_rsa
Your public key has been saved in /home/ietayjc/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:gkR08+fIffTYapl4v02Hcgx1RDWtkFozu9zKdAa4rh4 Ilya Tayc <ilyatayc@gmail.com>
>
The key's randomart image is:
+---[RSA 3072]-----+
|  .o o      . o*|
|  . . o    *  .o|
|  . . . .+. =...|
|  . . . =o.o=..|
|  . . S oo++o |
|  . . .o+*+. |
|  E..oB++ o|
|  ..oo+ o.|
|  .o.   o..|
+-----[SHA256]-----+
[ietayjc@10 ~]$ █

```

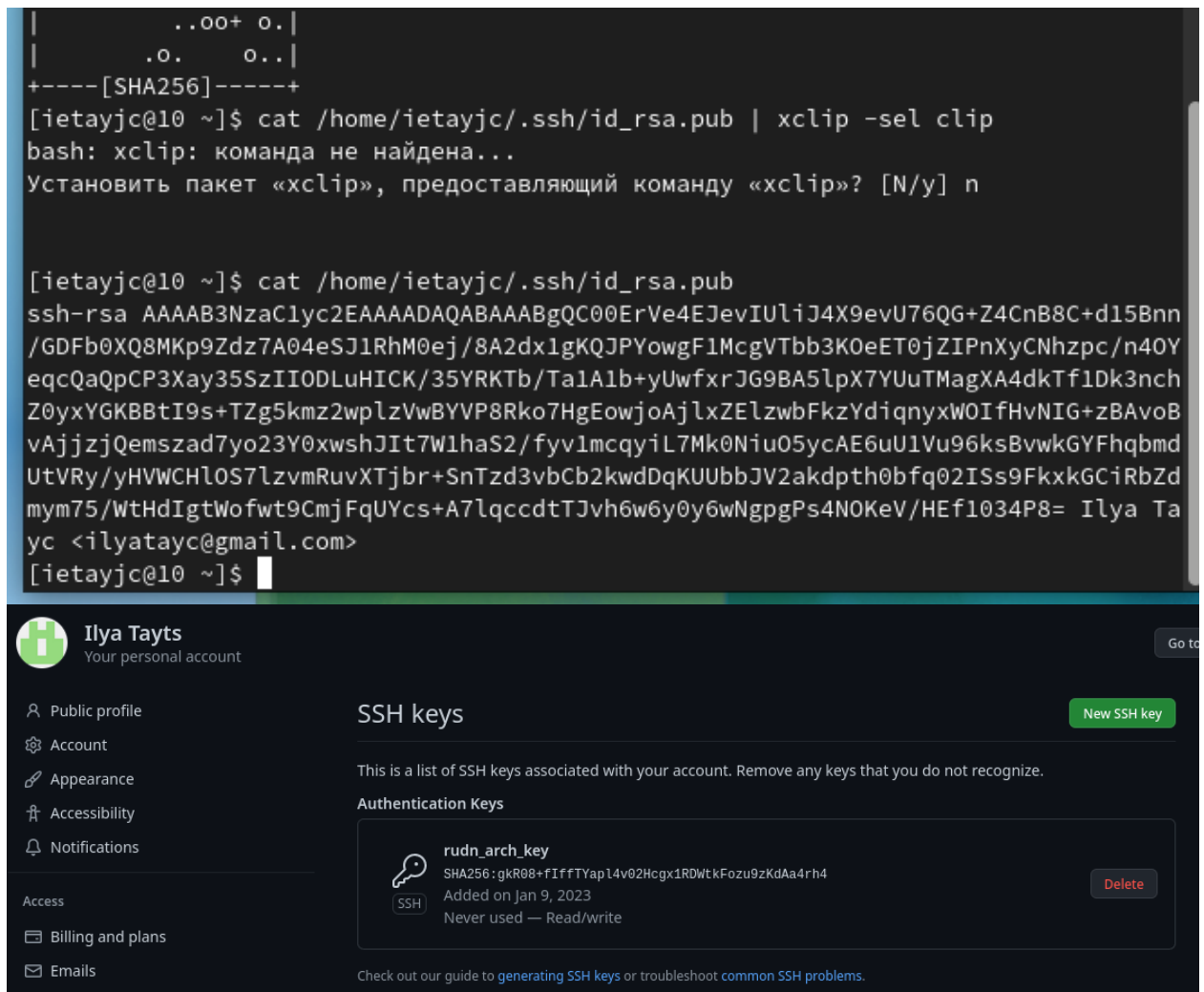
Зайти на сайт <http://github.org/> под своей учётной записью и перейти в меню Setting . После этого выбрать в боковом меню SSH and GPG keys и нажать кнопку New SSH key .

Скопировав из локальной консоли ключ в буфер обмена

```
cat ~/.ssh/id_rsa.pub | xclip -sel clip
```

вставляем ключ в появившееся на сайте поле и указываем для ключа имя (Title).





Пакет `xclip` не был установлен и не имея большой в нём нужды я решил получить ключ через выведение в командную строку и копирование от руки.

Ключ успешно указан.

#### 4. Сознание рабочего пространства и репозитория курса на основе шаблона

Рабочее пространство по предмету располагается в следующей иерархии:

`~/work/study/`

└── `<учебный год>/`

└── `<название предмета>/`

└── `<код предмета>/`

Например, для 2022–2023 учебного года и предмета «Архитектура компьютера» (код предмета `arch-rc`) структура каталогов примет следующий вид:

~/work/study/

└─ 2022–2023/

└─ Архитектура компьютера/

└─ arch-pc/

└─ labs/

└─ lab01/

└─ lab02/

└─ lab03/

...

- Каталог для лабораторных работ имеет вид labs.
- Каталоги для лабораторных работ имеют вид lab<номер>, например: lab01, lab02 и т.д.

Название проекта на хостинге git имеет вид:

study\_<учебный год>\_<код предмета>

Например, для 2022–2023 учебного года и предмета «Архитектура компьютера» (код предмета arch-pc) название проекта примет следующий вид:

study\_2022–2023\_arch-pc

задания и код:

Откройте терминал и создайте каталог для предмета «Архитектура компьютера»:

```
mkdir -p ~/work/study/2022-2023/"Архитектура компьютера"
```

```

[ietayjc@10 ~]$ mkdir -p ~/work/study/2022-2023/"Архитектура компьютера"
[ietayjc@10 ~]$ ls ~
work      Документы  Изображения  Общедоступные  Шаблоны
Видео     Загрузки  Музыка       'Рабочий стол'
[ietayjc@10 ~]$ ls -R ~/work
/home/ietayjc/work:
study

/home/ietayjc/work/study:
2022-2023

/home/ietayjc/work/study/2022-2023:
'Архитектура компьютера'

'/home/ietayjc/work/study/2022-2023/Архитектура компьютера':
[ietayjc@10 ~]$

```

Каталоги рабочего пространства созданы, соответствуют указанному шаблону.

## 5. Создание репозитория курса на основе шаблона 2

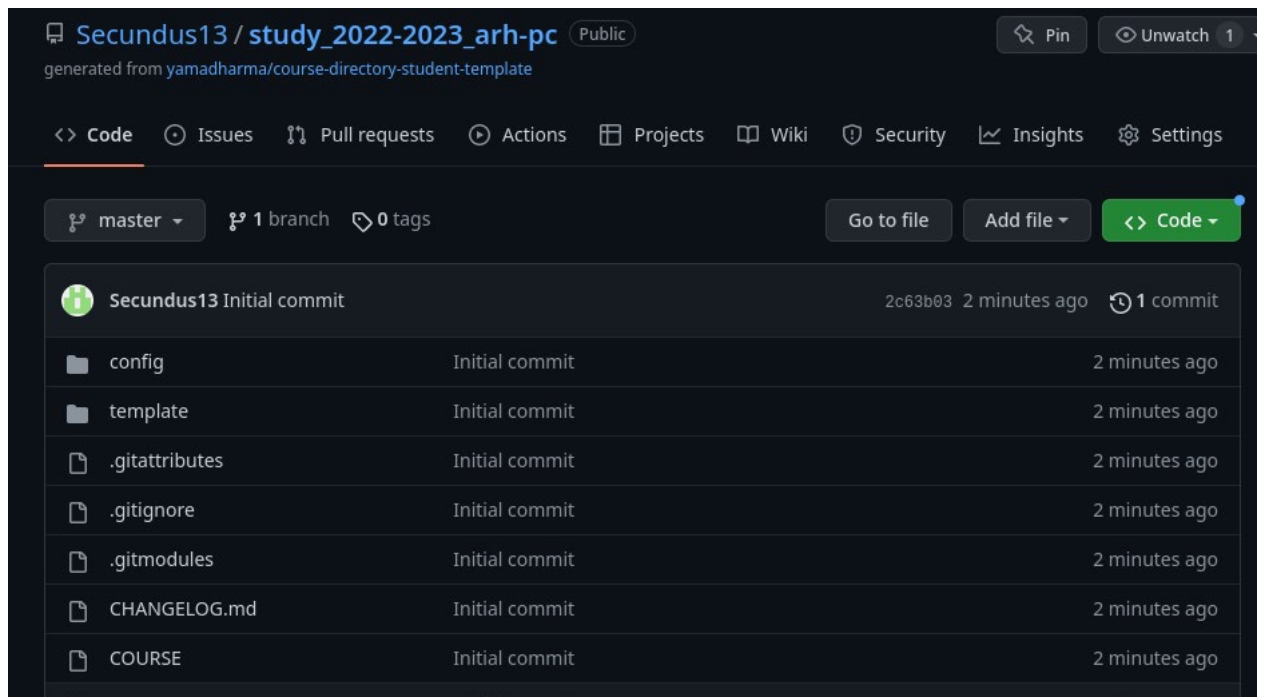
Перейдите на страницу репозитория с шаблоном курса  
<https://github.com/yamadharm/course-directory-student-template>.

Далее выберите Use this template.

В открывшемся окне задайте имя репозитория (Repository name)

study\_2022-2023\_arh-pc

и создайте репозиторий (кнопка Create repository from template).



Репозиторий создан. Поскольку других указаний не было, настройки создания я не менял, он публичный и копировалась только основная ветвь.

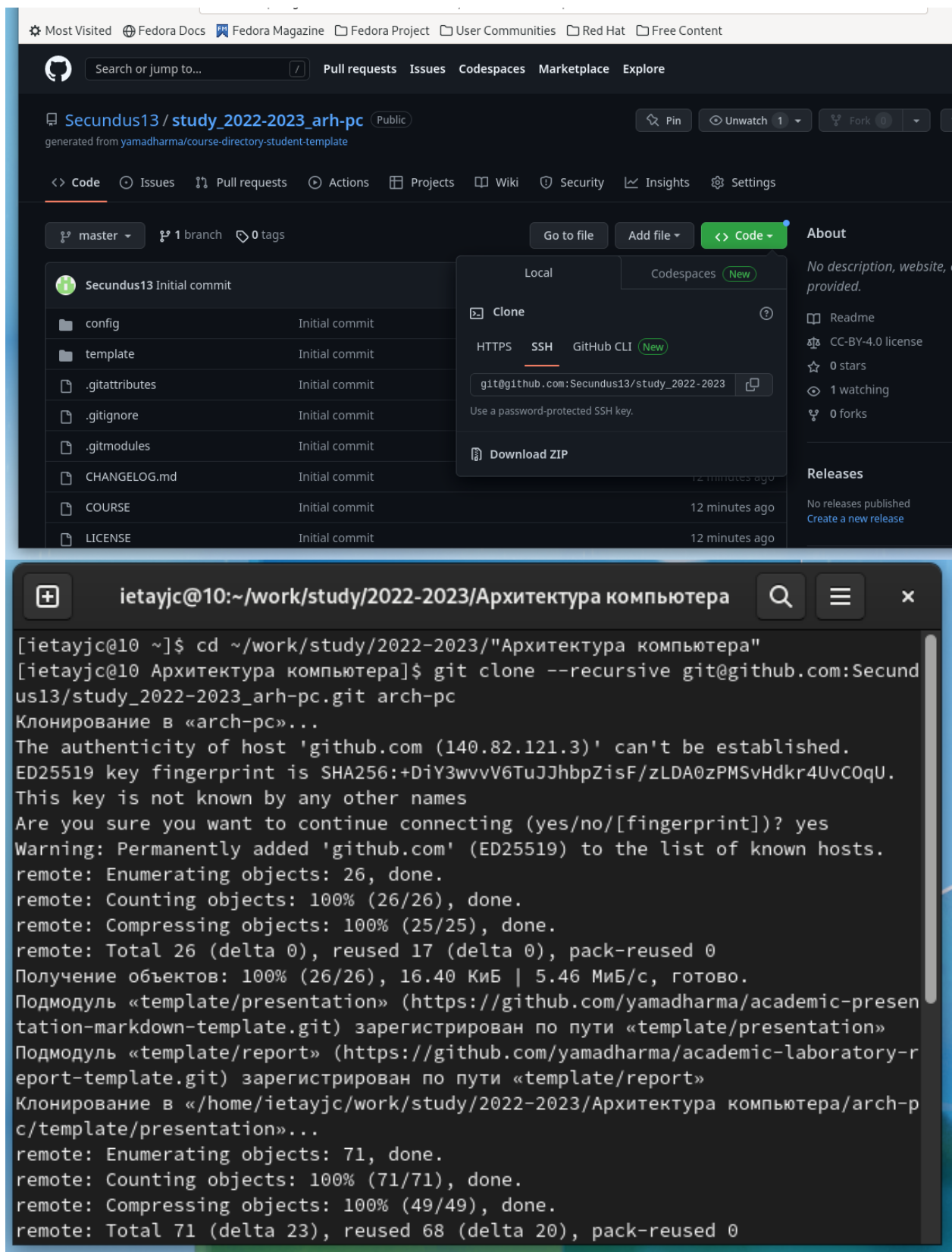
Откройте терминал и перейдите в каталог курса:

```
cd ~/work/study/2022-2023/"Архитектура компьютера"
```

клонировать созданный репозиторий:

```
git clone --recursive git@github.com:<user_name>/study_2022-2023_arh-pc.git  
arch-pc
```

Ссылку для клонирования можно скопировать на странице созданного репозитория Code -> SSH:



Клонирование завершено, ключ, кажется, исправен.

## 6. Настройка каталога курса

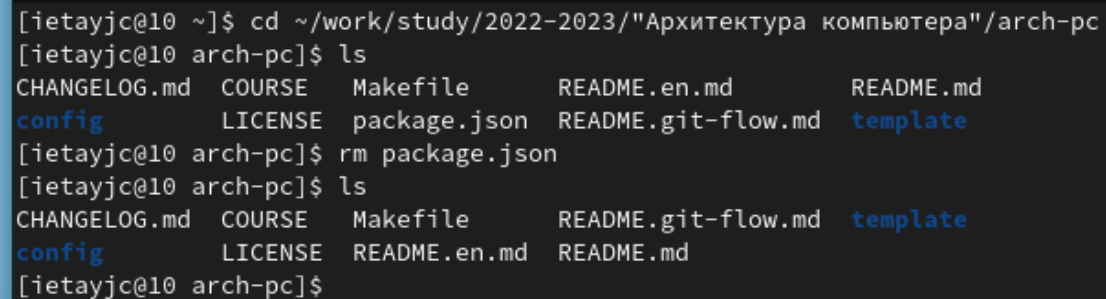
Задания и код:

Перейдите в каталог курса:

```
cd ~/work/study/2022-2023/"Архитектура компьютера"/arch-pc
```

Удалите лишние файлы:

```
rm package.json
```

A terminal window showing the execution of commands to navigate to a directory and remove a file. The prompt is [ietayjc@10 ~]. The user enters 'cd ~/work/study/2022-2023/"Архитектура компьютера"/arch-pc'. The prompt changes to [ietayjc@10 arch-pc]. The user enters 'ls', showing a directory listing: CHANGELOG.md, COURSE, Makefile, README.en.md, README.md, config, LICENSE, package.json, README.git-flow.md, and template. The user then enters 'rm package.json'. The prompt changes to [ietayjc@10 arch-pc]. The user enters 'ls' again, showing the updated directory listing: CHANGELOG.md, COURSE, Makefile, README.git-flow.md, template, config, LICENSE, README.en.md, and README.md. The prompt is now [ietayjc@10 arch-pc].

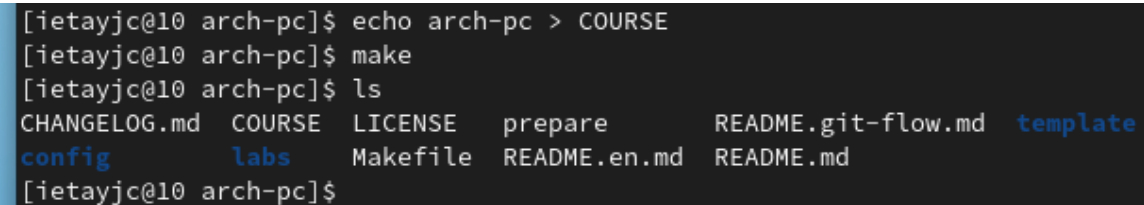
```
[ietayjc@10 ~]$ cd ~/work/study/2022-2023/"Архитектура компьютера"/arch-pc
[ietayjc@10 arch-pc]$ ls
CHANGELOG.md  COURSE  Makefile  README.en.md  README.md
config        LICENSE package.json README.git-flow.md template
[ietayjc@10 arch-pc]$ rm package.json
[ietayjc@10 arch-pc]$ ls
CHANGELOG.md  COURSE  Makefile  README.git-flow.md  template
config        LICENSE README.en.md README.md
[ietayjc@10 arch-pc]$
```

Поскольку файл package.json был указан как лишний – он был удалён согласно указаниям.

Создайте необходимые каталоги:

```
echo arch-pc > COURSE
```

```
make
```

A terminal window showing the execution of commands to create a directory and run make. The prompt is [ietayjc@10 arch-pc]. The user enters 'echo arch-pc > COURSE'. The prompt changes to [ietayjc@10 arch-pc]. The user enters 'make'. The prompt changes to [ietayjc@10 arch-pc]. The user enters 'ls', showing a directory listing: CHANGELOG.md, COURSE, LICENSE, prepare, README.git-flow.md, template, config, labs, Makefile, README.en.md, and README.md. The prompt is now [ietayjc@10 arch-pc].

```
[ietayjc@10 arch-pc]$ echo arch-pc > COURSE
[ietayjc@10 arch-pc]$ make
[ietayjc@10 arch-pc]$ ls
CHANGELOG.md  COURSE  LICENSE  prepare  README.git-flow.md  template
config        labs    Makefile README.en.md README.md
[ietayjc@10 arch-pc]$
```

Появился каталог labs.

Отправьте файлы на сервер:

```
git add .
```

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







```
git push
```

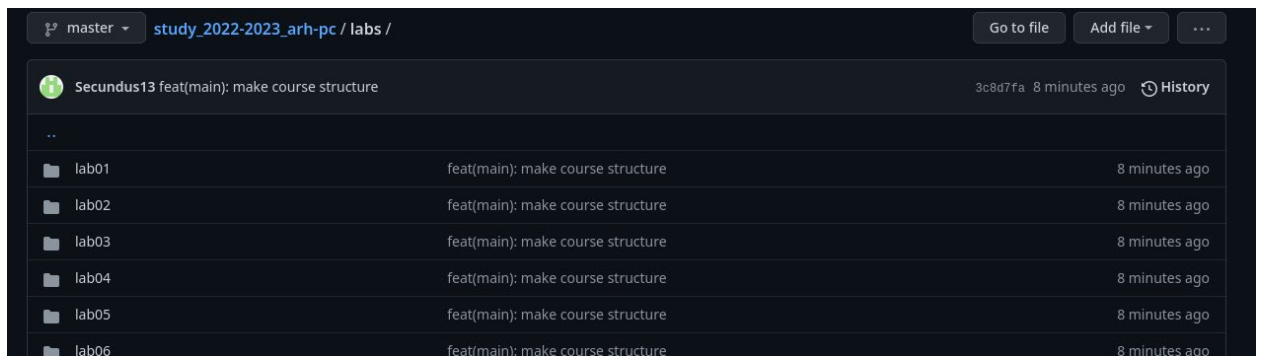
```
[ietayjc@10 arch-pc]$ git add .
[ietayjc@10 arch-pc]$ git commit -am 'feat(main): make course structure'
[master 3c8d7fa] feat(main): make course structure
 91 files changed, 8229 insertions(+), 14 deletions(-)
 create mode 100644 labs/lab01/presentation/Makefile
 create mode 100644 labs/lab01/presentation/image/kulyabov.jpg
 create mode 100644 labs/lab01/presentation/presentation.md
 create mode 100644 labs/lab01/report/Makefile
 create mode 100644 labs/lab01/report/bib/cite.bib
 create mode 100644 labs/lab01/report/image/placeimg_800_600_tech.jpg
 create mode 100644 labs/lab01/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
 create mode 100644 labs/lab01/report/report.md
 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/Makefile
 create mode 100644 labs/lab02/report/bib/cite.bib
 create mode 100644 labs/lab02/report/image/placeimg_800_600_tech.jpg
 create mode 100644 labs/lab02/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
 create mode 100644 labs/lab02/report/report.md
 create mode 100644 labs/lab03/presentation/Makefile
 create mode 100644 labs/lab03/presentation/image/kulyabov.jpg
 create mode 100644 labs/lab03/presentation/presentation.md
```

```
[ietayjc@10 arch-pc]$ git push
Перечисление объектов: 22, готово.
Подсчет объектов: 100% (22/22), готово.
Сжатие объектов: 100% (16/16), готово.
Запись объектов: 100% (20/20), 310.94 КиБ | 2.17 МиБ/с, готово.
Всего 20 (изменений 1), повторно использовано 0 (изменений 0), повторно использо
вано пакетов 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:Secundus13/study_2022-2023_arh-pc.git
   2c63b03..3c8d7fa  master -> master
[ietayjc@10 arch-pc]$
```

Внесение произведённых изменений на сервер. На первой иллюстрации видно часть отчёта о произведённых изменениях, на второй – об успешной отправке и сохранении.

Проверьте правильность создания иерархии рабочего пространства в локальном репозитории и на странице github.

	Secundus13 feat(main): make course structure	3c8d7fa 7 minutes ago	 2 commits
	config	Initial commit	37 minutes ago
	labs	feat(main): make course structure	7 minutes ago
	template	Initial commit	37 minutes ago
	gitattributes	Initial commit	37 minutes ago



Иерархия и строение соответствуют указанному шаблону.

### Задания для самостоятельной работы

1. Создайте отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства (labs>lab03>report).

Создание иллюстрации к этому заданию было затруднительным в виду некоторых логических противоречий. Если вы получили этот файл через github то думаю проблем с подтверждением возникнуть не должно.

2. Скопируйте отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданного рабочего пространства.

Отчёты делались в основной рабочей системе (Windows), в результате чего потребовалось ознакомиться с загрузкой файлов в виртуальную машину. Копирование, как и прочие действия проводились в визуальной среде ввиду большего удобства пользования для данных целей.

```
[ietayjc@10 ~]$ ls /home/ietayjc/work/study/2022-2023/"Архитектура компьютера"/arch-pc/labs/lab01 /home/ietayjc/work/study/2022-2023/"Архитектура компьютера"/arch-pc/labs/lab02
'/home/ietayjc/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab01':
presentation report ЛабАрх1a.pdf

'/home/ietayjc/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab02':
presentation report ЛабАрх2a.pdf
[ietayjc@10 ~]$
```

3. Загрузите файлы на github.

Поскольку внести изменения в уже загруженный файл можно только с загрузкой нового, в загруженном сейчас файле не будет данного отчёта. Тем не менее с учётом обыденности таких процедур подтверждение не должно стать проблемой.

код:



git add .

git commit -am 'feat(main): loaded 3 labs'

git push

```
[ietayjc@10 arch-pc]$ git add .
[ietayjc@10 arch-pc]$ git commit -am 'feat(main): loaded 3 labs'
[master 66c1160] feat(main): loaded 3 labs
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100755 labs/lab01/ЛабАрх1a.pdf
 create mode 100755 labs/lab02/ЛабАрх2a.pdf
[ietayjc@10 arch-pc]$ git push
Перечисление объектов: 11, готово.
Подсчет объектов: 100% (10/10), готово.
Сжатие объектов: 100% (7/7), готово.
Запись объектов: 100% (7/7), 2.47 МиБ | 1.19 МиБ/с, готово.
Всего 7 (изменений 3), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:Secundus13/study_2022-2023_arh-pc.git
   3c8d7fa..66c1160  master -> master
[ietayjc@10 arch-pc]$
```

## 5 Выводы

Система контроля версий git это удобный и полезный инструмент для ведения совместной работы с проектами, имеющая эффективную систему сжатия и удобный веб интерфейс. Система является общепринятой в сообществе программистов. Полученные в ходе выполнения навыки необходимы для дальнейшей работы.

## 6 Список литературы