

Отчёт по лабораторной работе № 3

Дисциплина: Архитектура Компьютера

Гибшер Кирилл Владимирович, НКАбд-01-22

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Выводы	19
	Список литературы	20

Список иллюстраций

4.1	Рис. 6	13
4.2	Рис. 7	13
4.3	Рис. 8	14
4.4	Рис. 9	14
4.5	Рис. 10	15
4.6	Рис. 11	15
4.7	Рис. 12	16
4.8	Рис. 13	16
4.9	Рис. 14	17
4.10	Рис. 15	17
4.11	Рис. 16	17
4.12	Рис. 17	18
4.13	Рис. 18	18

Список таблиц

1 Цель работы

Целью данной лабораторной работы является изучить идеологию и применение средств контроля версий, а также приобретение практических навыков по работе с системой git.

2 Задание

1. Настроить учетную свою запись на сервере GitHub.
2. Провести базовую конфигурацию git в терминале Linux.
3. Создать SSH ключ для последующей идентификации пользователя на сервере репозитория.
4. Просмотреть рабочее пространство и репозитория курса и на основе шаблона создать его, а также настроить его каталоги.
5. Сделать отчёт по третьей лабораторной в формате markdown.
6. Загрузил файлы на github.

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. 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. 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.

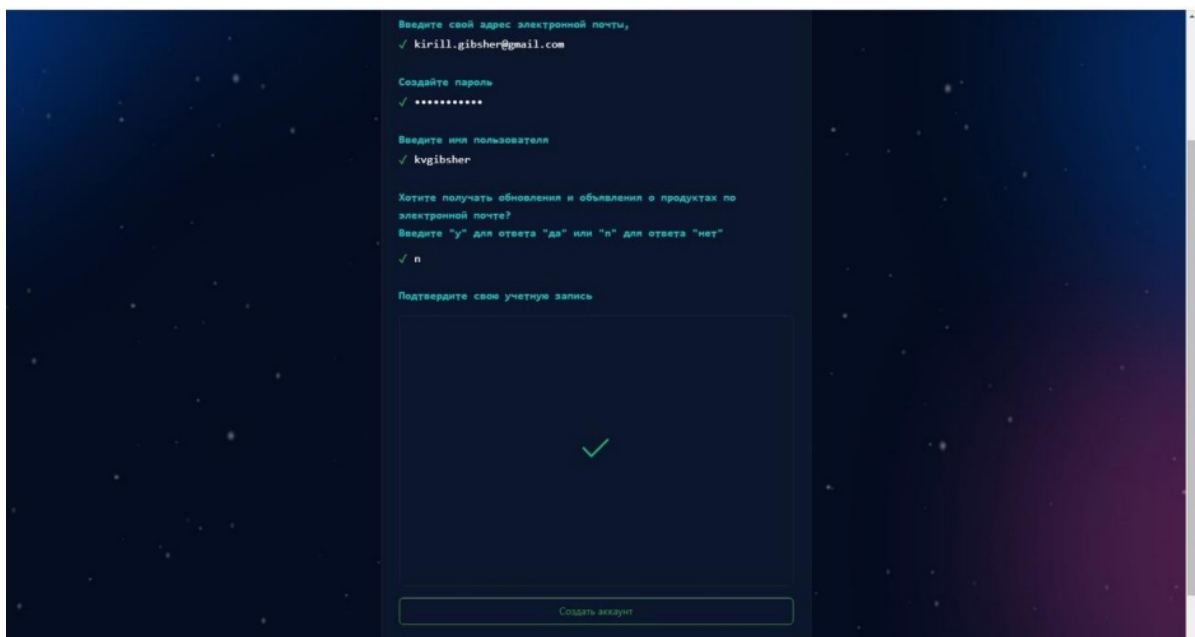
Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` с различными опциями. Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией. Наиболее часто используемые команды `git` представлены на рисунке ниже. (рис. ??)

[Рис.1] (image/1.jpg) {#fig:001 width=70% }

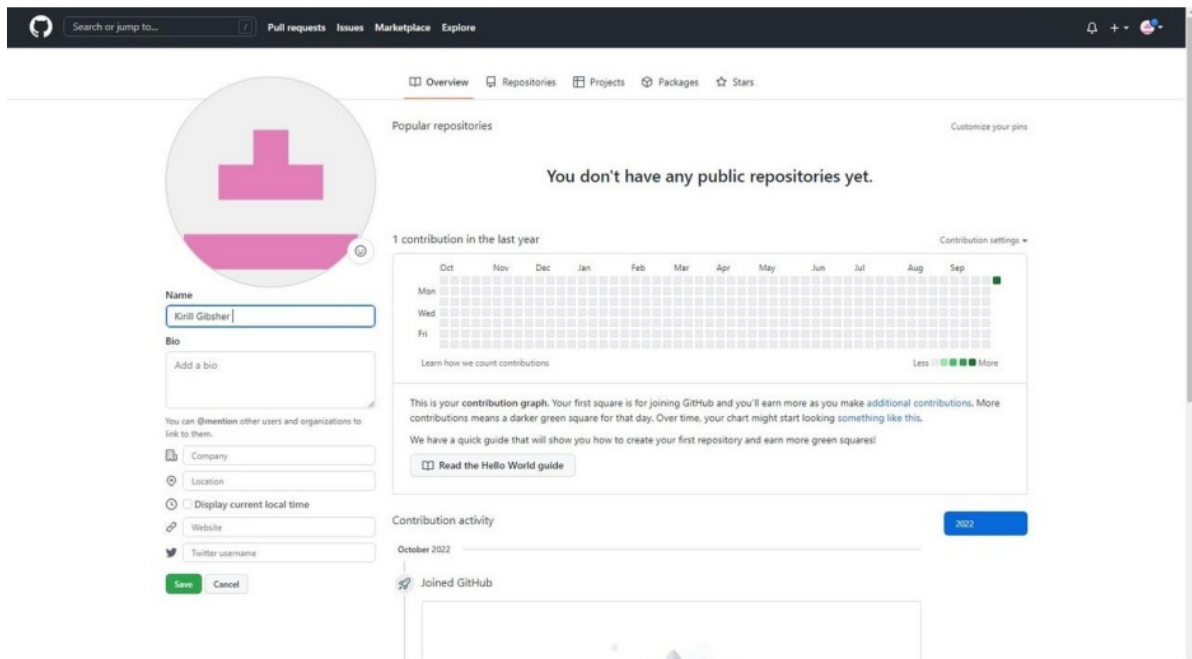
Работа пользователя со своей веткой начинается с проверки и получения изменений из центрального репозитория (при этом в локальное дерево до начала этой процедуры не должно было вноситься изменений): `git checkout master` `git pull` `git checkout -b имя_ветки` Затем можно вносить изменения в локальном 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` / `git push`

4 Выполнение лабораторной работы

1. Для начала лабораторной работы необходимо создать учетную запись на GitHub и заполнить основные данные (рис. ??, ??).

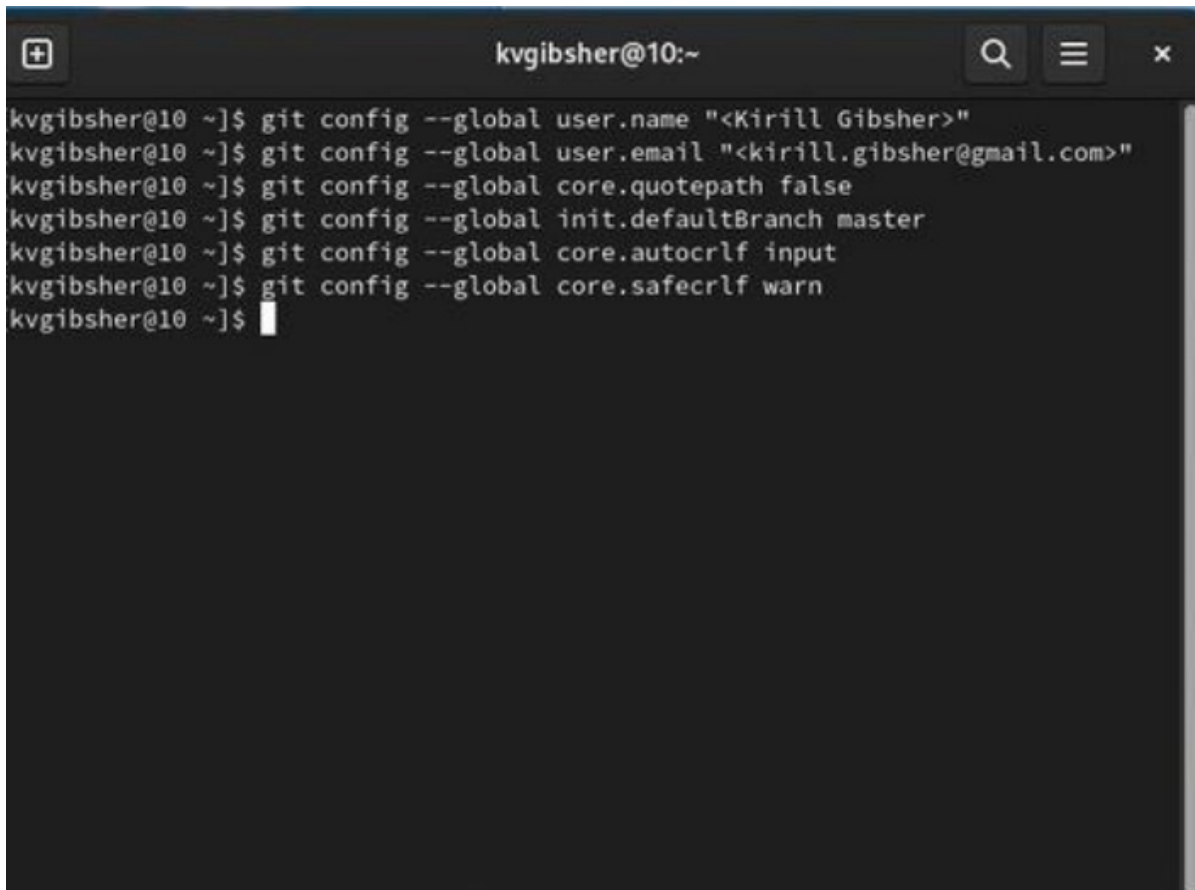


{ #fig:002 width=70% }



{ #fig:003 width=70% }

2. Далее сделаем предварительную конфигурацию git. Открываем терминал и вводим следующие команды , указав свой email и имя владельца репозитория. Также настроим utf-8 в выводе сообщений git.Также зададим имя начальной ветки(назвав ее master) и настроим параметры autocrlf и safecrlf. (рис. ??).

A terminal window titled 'kvgibsher@10:~' with search, menu, and close icons in the title bar. The terminal displays a series of git configuration commands being entered at the prompt 'kvgibsher@10 ~]\$:'.

```
kvgibsher@10 ~]$ git config --global user.name "<Kirill Gibsher>"
kvgibsher@10 ~]$ git config --global user.email "<kirill.gibsher@gmail.com>"
kvgibsher@10 ~]$ git config --global core.quotepath false
kvgibsher@10 ~]$ git config --global init.defaultBranch master
kvgibsher@10 ~]$ git config --global core.autocrlf input
kvgibsher@10 ~]$ git config --global core.safecrlf warn
kvgibsher@10 ~]$
```

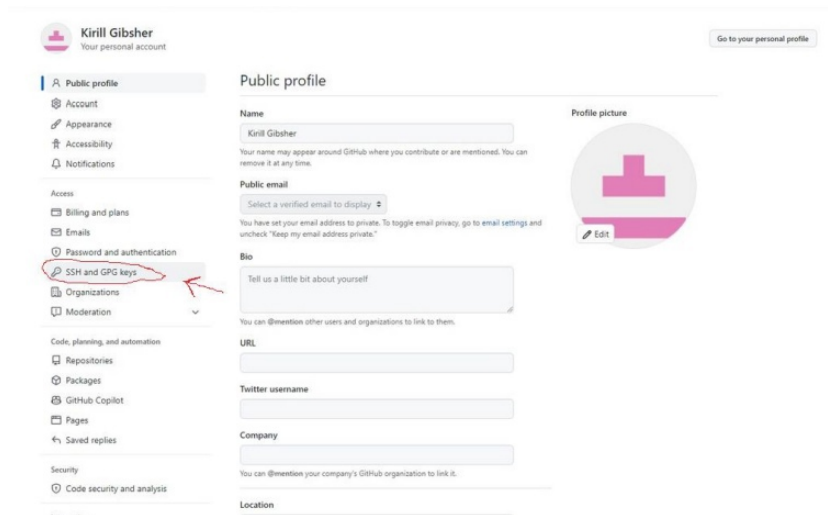
{ #fig:004 width=70% }

3. Далее для последующей идентификации на сервере репозитория нам нужно сгенерировать пару ключей (приватный и открытый). Ключи сохраняются в каталоге ~/.ssh/. (рис. ??).

```
kvgibsher@10:~$ ssh-keygen -C "Kirill Gibsher <kirill.gibsher@gmail.com>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kvgibsher/.ssh/id_rsa):
Created directory '/home/kvgibsher/.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/kvgibsher/.ssh/id_rsa
Your public key has been saved in /home/kvgibsher/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:X24F5NT9MbG9k8h63myeoBImvKawHUqE2wR3ibLj+tI Kirill Gibsher <kirill.gibsher@gmail.com>
The key's randomart image is:
+---[RSA 3072]-----+
|  . .  o. o. |
| o o o  + .o+ |
| * .    o .:= |
| + o     ... + |
| . * . S .o.+ |
| o o + + o.. . |
| o . + = o.oo |
| o E + + . .+ +. |
| o...o  .. .o= |
+---[SHA256]-----+
[kvgibsher@10 ~]$
```

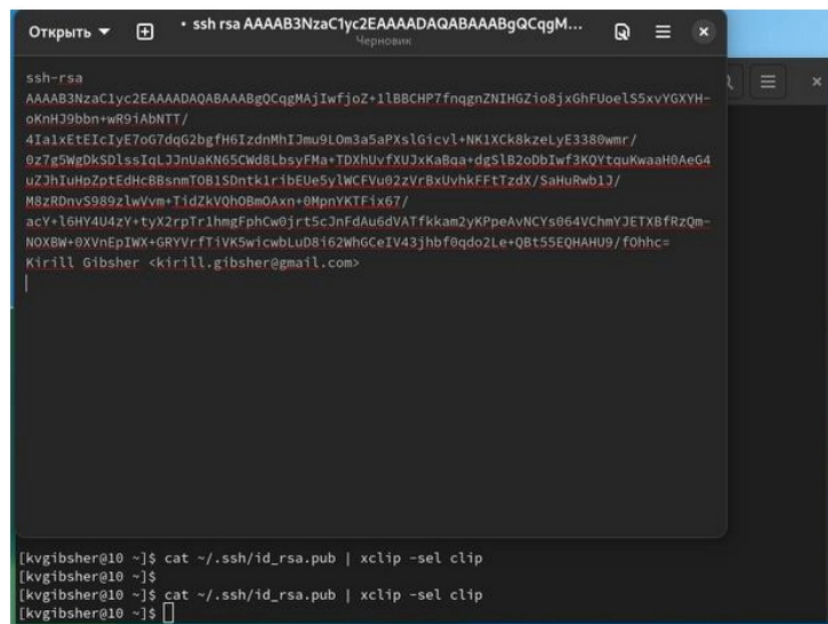
{ #fig:005 width=70% }

4. Далее необходимо загрузить сгенерированный открытый ключ. Для этого мы переходим под свою учетную запись на сайте [github.ru](https://github.com) , перейти в меню , затем в настройки , после этого выбрать в боковом меню SSH and GPG Keys и нажать New SSH Key (рис. 4.1).



4.1: Рис. 6

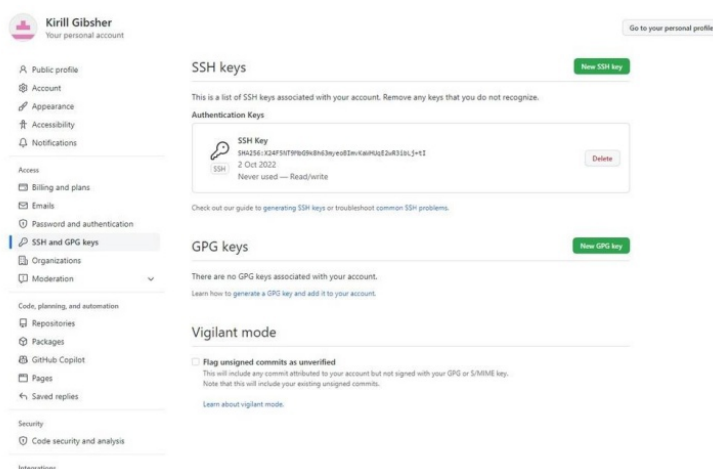
5. Далее нам необходимо скопировать из локальной консоли сгенерированный ключ в буфер обмена для последующей вставки в нужную графу на сайте. Делаем это с помощью соответствующей команды `cat` и `xclip -sel`. (рис. 4.2).



4.2: Рис. 7

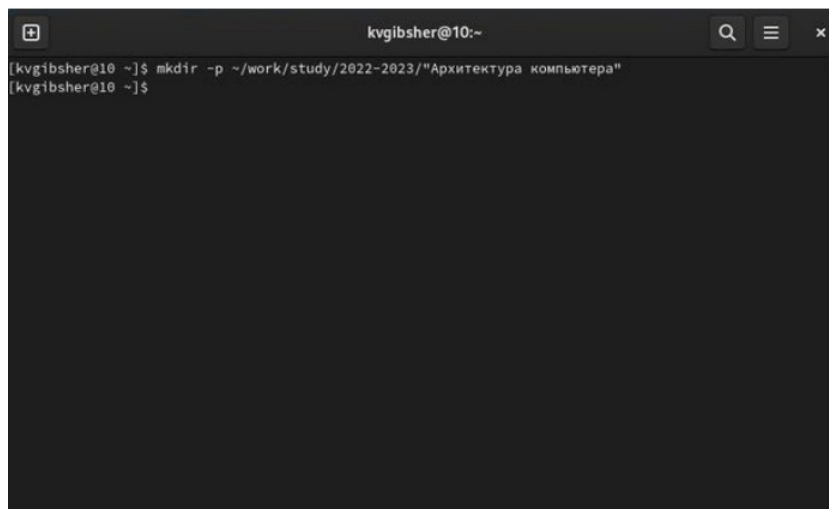
6. Затем вставляем ключ в появившееся на сайте поле и указываем для ключа

имя ,в данном случае я назвал его «SSH Key» (рис. 4.3).



4.3: Рис. 8

7. Далее нам необходимо создать рабочее пространство. Для этого создаем каталог для предмета «Архитектура компьютера» (рис. 4.4).



4.4: Рис. 9

8. Затем переходим на страницу репозитория с шаблоном курса , по указанной ссылке и выбираем «Use this template» (рис. 4.5).


```
kvigibsher@10:~/work/study/2022-2023/Архитектура компьютера
kvigibsher@10:~/work/study/2022-2023/Архитектура компьютера
[kvigibsher@10 Архитектура компьютера]$ git clone --recursive git@github.com:kvigibsher/study_2022-2023_arh-pc.git arch-pc
bash: kvigibsher: Нет такого файла или каталога
[kvigibsher@10 Архитектура компьютера]$ git clone --recursive git@github.com:kvigibsher/study_2022-2023_arh-pc.git arch-pc
Клонирование в «arch-pc...»
The authenticity of host 'github.com (149.82.121.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3vW9t3HbPZisFzLDA8zPMSVdK4UvCQdU.
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: 26, done.
remote: Counting objects: 100% (26/26), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 26 (delta 0), reused 17 (delta 0), pack-reused 0
Получение объектов: 100% (26/26), 16.83 КБ | 1.78 МБ/с, 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/kvigibsher/work/study/2022-2023/Архитектура компьютера/arch-pc/template/presentation...»
remote: Enumerating objects: 71, done.
remote: Counting objects: 100% (71/71), done.
remote: Compressing objects: 100% (49/49), done.
remote: Total 71 (delta 23), reused 68 (delta 20), pack-reused 0
Получение объектов: 100% (71/71), 88.89 КБ | 842.00 КБ/с, готово.
Определение изменений: 100% (23/23), готово.
Клонирование в «/home/kvigibsher/work/study/2022-2023/Архитектура компьютера/arch-pc/template/report...»
remote: Enumerating objects: 78, done.
remote: Counting objects: 100% (78/78), done.
remote: Compressing objects: 100% (52/52), done.
remote: Total 78 (delta 31), reused 66 (delta 22), pack-reused 0
Получение объектов: 100% (78/78), 292.27 КБ | 1.87 МБ/с, готово.
Определение изменений: 100% (31/31), готово.
Submodule path 'template/presentation': checked out '2703b474237926d72694aaf7555a5626dce51a25'
Submodule path 'template/report': checked out 'df7b2ef80f8def3b9a496f8695277469a1a7842a'
[kvigibsher@10 Архитектура компьютера]$
```

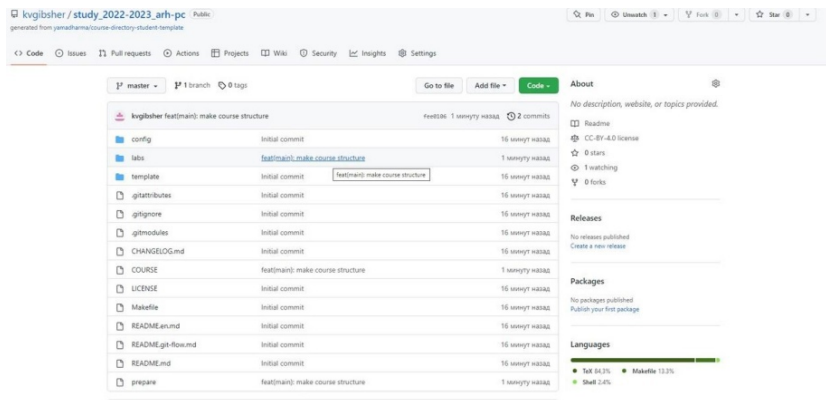
4.7: Рис. 12

11. Создаем необходимые каталоги и отправляем файлы на сервер. (рис. 4.8).

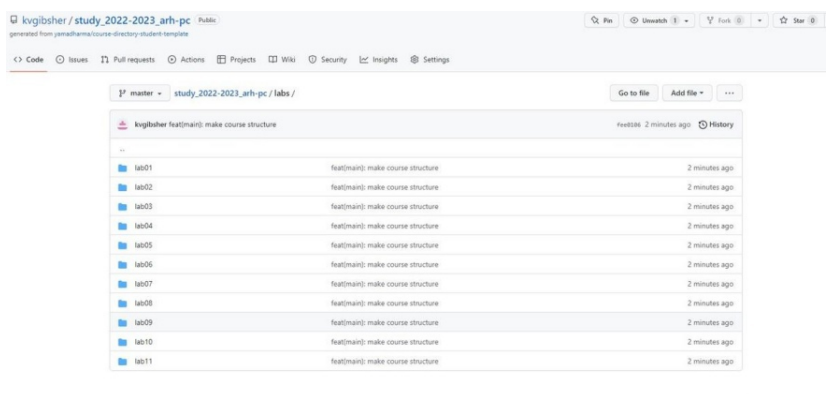
```
[kvigibsher@10 arch-pc]$ echo arch-pc > COURSE
[kvigibsher@10 arch-pc]$ make
[kvigibsher@10 arch-pc]$ git add .
[kvigibsher@10 arch-pc]$ git commit -m 'feat(main): make course structure'
[master (root)] feat(main): make course structure
91 files changed, 8229 insertions(+), 14 deletions(-)
create 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/placement_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/placement_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
create mode 100644 labs/Lab03/report/Makefile
create mode 100644 labs/Lab03/report/bib/cite.bib
create mode 100644 labs/Lab03/report/image/placement_800_600_tech.jpg
create mode 100644 labs/Lab03/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/Lab03/report/report.md
create mode 100644 labs/Lab04/presentation/Makefile
create mode 100644 labs/Lab04/presentation/image/kulyabov.jpg
create mode 100644 labs/Lab04/presentation/presentation.md
create mode 100644 labs/Lab04/report/Makefile
```

4.8: Рис. 13

12. Проверяем правильность создания иерархии рабочего пространства в на странице github. (рис. 4.9,4.10).

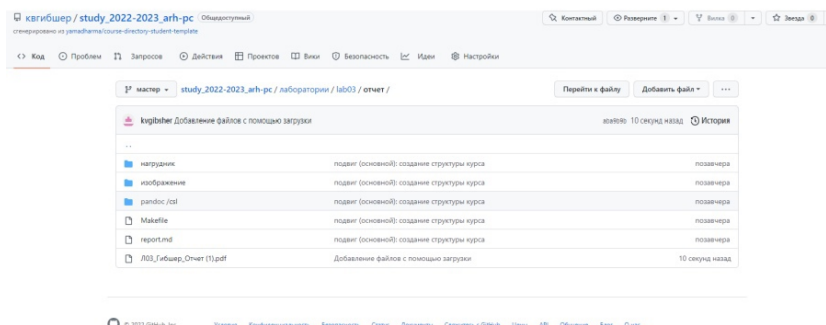


4.9: Рис. 14



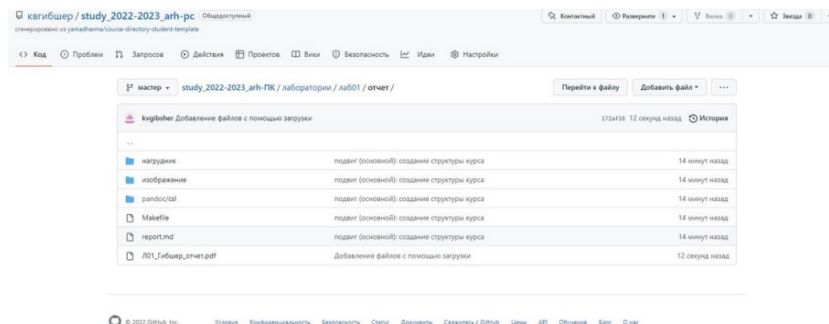
4.10: Рис. 15

13. Создадим отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства. (рис. 4.11).

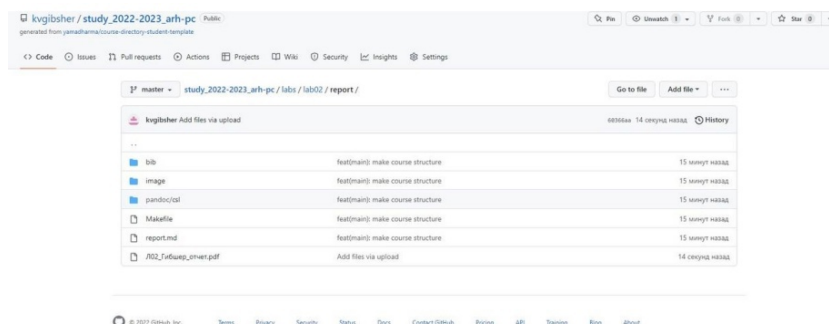


4.11: Рис. 16

14. Скопируем отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданного рабочего пространства. Таким образом все необходимые файлы были загружены на GitHub (рис. 4.12 4.13).



4.12: Рис. 17



4.13: Рис. 18

5 Выводы

Как итог выполнения данной лабораторной работы я создал собственную учетную запись на пространстве GitHub, который пригодится для выполнения следующих работ и прогресса по курсу «Архитектура ЭВМ». Также изучил идеологию и применение средств контроля версий. Приобрел практические навыки по работе с системой git в терминале, самостоятельно создал SSH ключ и понял структуру пространства, в котором необходимо будет работать по мере прохождения курса.

Список литературы

1. Текстовый файл «Лабораторная работа №3. Система контроля версий Git