

# **Отчёт по лабораторной работе №3. Система контроля версий Git**

Королёв Иван Андреевич

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
2.1	Настройка GitHub: . . . . .	7
2.2	Базовая настройка git: . . . . .	7
2.3	Создание SSH-ключа: . . . . .	8
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
<b>3</b>	<b>Выполнение самостоятельной работы:</b>	<b>13</b>
<b>4</b>	<b>Выводы</b>	<b>15</b>

## Список иллюстраций

2.1	Создание учетной записи на GitHub . . . . .	7
2.2	Указываем имя и email . . . . .	8
2.3	Настройка utf-8 . . . . .	8
2.4	Имя начальной ветки . . . . .	8
2.5	Параметр autocrlf . . . . .	8
2.6	Параметр safecrlf . . . . .	8
2.7	Создание ssh-ключа . . . . .	8
2.8	Нажимаю New SSH Key, загружаю сгенерированный ключ . . . . .	9
2.9	Команда cat . . . . .	9
2.10	Создание ssh-ключа . . . . .	9
2.11	Создание каталога . . . . .	10
2.12	Создаю репозиторий . . . . .	10
2.13	Команда cd . . . . .	11
2.14	Клонирую созданный репозиторий . . . . .	11
2.15	Команда cd . . . . .	11
2.16	Удаление лишнего и создание необходимых файлов . . . . .	11
2.17	Команда git push . . . . .	12
2.18	Создал рабочее пространство . . . . .	12
3.1	Перемещаю загруженную лабораторную работу . . . . .	13
3.2	С помощью команд git pull, add, commit, push, загружаю все лабораторные в репозиторий . . . . .	14
3.3	Лабораторная работа . . . . .	14

## Список таблиц

# 1 Цель работы

Целью работы является изучить идеологию и применение средств контроля версий. Приобрести практические навыки по работе с системой git. # Задание Необходимо зарегистрироваться на GitHub. Сделать базовые настройки, создать SSH-ключ, создать рабочее пространство и репозиторий курса и настроить каталог курса # 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.

## 2 Выполнение лабораторной работы

### 2.1 Настройка GitHub:

Создаю учетную запись на GitHub и заполняю основные данные(??)

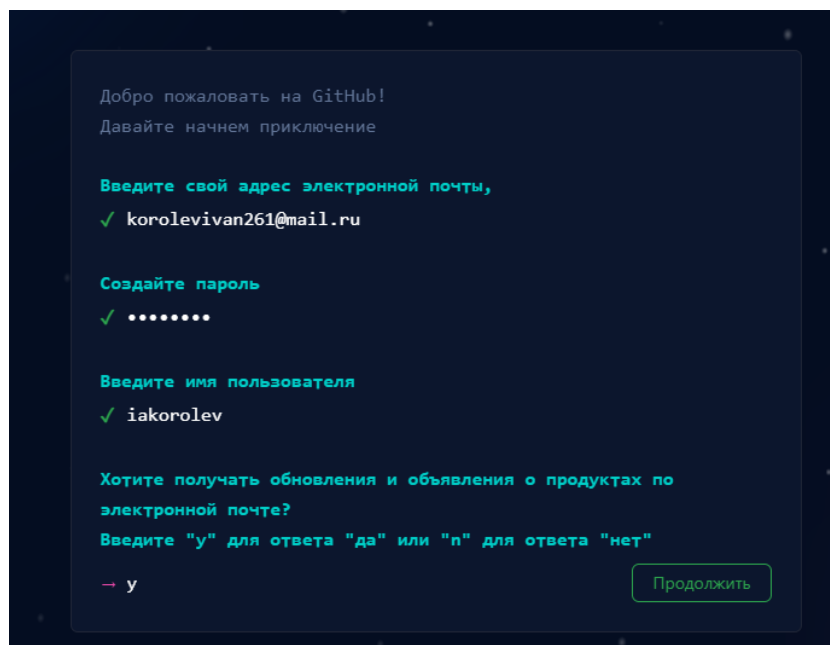


Рис. 2.1: Создание учетной записи на GitHub

### 2.2 Базовая настройка git:

Сначала сделаю предварительную конфигурацию git. Открою терминал и введу следующие команды, указав имя и email(??)

```
[iakorolyov@fedora ~]$ git config --global user.name "<iakorolyov>"  
[iakorolyov@fedora ~]$ git config --global user.email "<1032225751@pfur.ru>"
```

Рис. 2.2: Указываем имя и email

Настрою utf-8 в выводе сообщений git(??)

```
[iakorolyov@fedora ~]$ git config --global core.quotepath false
```

Рис. 2.3: Настройка utf-8

Задам имя начальной ветки(??)

```
[iakorolyov@fedora ~]$ git config --global init.defaultBranch master
```

Рис. 2.4: Имя начальной ветки

Параметр autocrlf(??)

```
[iakorolyov@fedora ~]$ git config --global core.autocrlf input
```

Рис. 2.5: Параметр autocrlf

Параметр safecrlf(-fig. ??])

```
[iakorolyov@fedora ~]$ git config --global core.safecrlf warn
```

Рис. 2.6: Параметр safecrlf

## 2.3 Создание SSH-ключа:

Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый)(??)

```
[iakorolyov@fedora ~]$ ssh-keygen -C "Ivan Korolev <1032225751@pfur.ru>"  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/iakorolyov/.ssh/id_rsa):  
/home/iakorolyov/.ssh/id_rsa already exists.
```

Рис. 2.7: Создание ssh-ключа



Ключи хранятся в каталоге ~/.ssh/. Далее необходимо загрузить сгенерированный открытый ключ. Для этого захожу на сайт github. Нажимаю на иконку своей учетной записи и перехожу в Settings. Далее нажимаю SSH and GPG key, и нажимаю New Key()(рис.8 ??)

SSH keys

New SSH key

Рис. 2.8: Нажимаю New SSH Key, загружаю сгенерированный ключ

Скопирую из локальной консоли ключ в буфер обмена с помощью команды(рис.9 ??)

```
[iakorolyov@fedora ~]$ cat ~/.ssh/id_rsa.pub | xclip -sel clip  
[iakorolyov@fedora ~]$
```

Рис. 2.9: Команда cat

Вставляем ключ в появившееся на сайте поле и указываем для ключа имя(gite)(рис.10 ??)

SSH keys / Add new

Title

Key type

Authentication Key ↕

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

Add SSH key

Рис. 2.10: Создание ssh-ключа

## 2.4 Создание рабочего пространства и репозитория курса на основе шаблона:

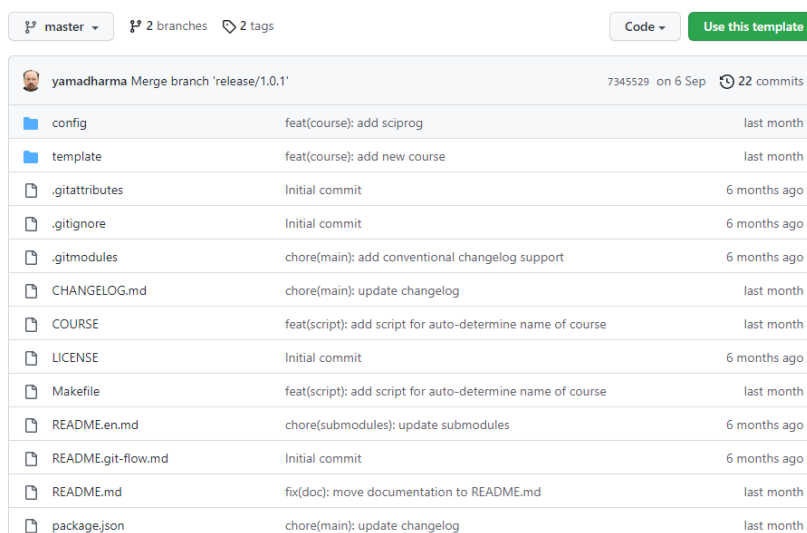
Открою терминал и создам каталог для предмета «Архитектура компьютера»(рис.11 ??)

```
[iakorolyov@fedora ~]$ mkdir -p ~/work/study/2022-2023/"Архитектура компьютера"
[iakorolyov@fedora ~]$
```

Рис. 2.11: Создание каталога

## 2.5 Создание репозитория курса на основе шаблона:

Репозиторий на основе шаблона можно создать через web-интерфейс github. Перейду на страницу репозитория с шаблоном курса <https://github.com/yamadharma/course-directory-student-template>. Далее выберу Use this template(рис.12 ??)



master 2 branches 2 tags		Code Use this template
yamadharma Merge branch 'release/1.0.1' 7345529 on 6 Sep 22 commits		
config	feat(course): add sciprog	last month
template	feat(course): add new course	last month
.gitattributes	Initial commit	6 months ago
.gitignore	Initial commit	6 months ago
.gitmodules	chore(main): add conventional changelog support	6 months ago
CHANGELOG.md	chore(main): update changelog	last month
COURSE	feat(script): add script for auto-determine name of course	last month
LICENSE	Initial commit	6 months ago
Makefile	feat(script): add script for auto-determine name of course	last month
README.en.md	chore(submodules): update submodules	6 months ago
README.git-flow.md	Initial commit	6 months ago
README.md	fix(doc): move documentation to README.md	last month
package.json	chore(main): update changelog	last month

Рис. 2.12: Создаю репозиторий

В открывшемся окне задаю имя репозитория (Repository name) study\_2022–2023\_arh-  
ps и создаю репозиторий (кнопка Create repository from template). Открою

терминал и перейду в каталог(рис.13 ??)

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

Рис. 2.13: Команда cd

Клонирую созданный репозиторий(рис.14 ??)

```
git clone --recursive git@github.com:iakorolev/study_2022-2023_arc-pc.git arch-pc
```

Рис. 2.14: Клонирую созданный репозиторий

## 2.6 Настройка каталога курса:

Перейду в каталог курса(рис.15 ??)

```
[iakorolyov@fedora Архитектура компьютера]$ cd ~/work/study/2022-2023/"Архитектура компьютера"/arch-pc  
[iakorolyov@fedora arch-pc]$
```

Рис. 2.15: Команда cd

Удалю лишние файлы,создам необходимые каталоги(рис.16 ??)

```
[iakorolyov@fedora arch-pc]$ rm package.json  
[iakorolyov@fedora arch-pc]$ echo arch-pc > COURSE  
[iakorolyov@fedora arch-pc]$ MAKE  
bash: MAKE: команда не найдена...  
Аналогичная команда: 'make'  
[iakorolyov@fedora arch-pc]$ make  
[iakorolyov@fedora arch-pc]$ git add .  
[iakorolyov@fedora arch-pc]$ git commit -am 'feat(main): make course structure'  
[master f183889] 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
```

Рис. 2.16: Удаление лишнего и создание необходимых файлов

Отправлю файлы на сервер(рис.17 ??)

```
[iakorolyov@fedora 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 | 1.18 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:iakorolev/study_2022-2023_arh-pc.git
   12edf76..f183889  master -> master
[iakorolyov@fedora arch-pc]$
```

Рис. 2.17: Команда git push

Проверю правильность создания иерархии рабочего пространства в локальном репозитории и на странице github(рис.18 ??)

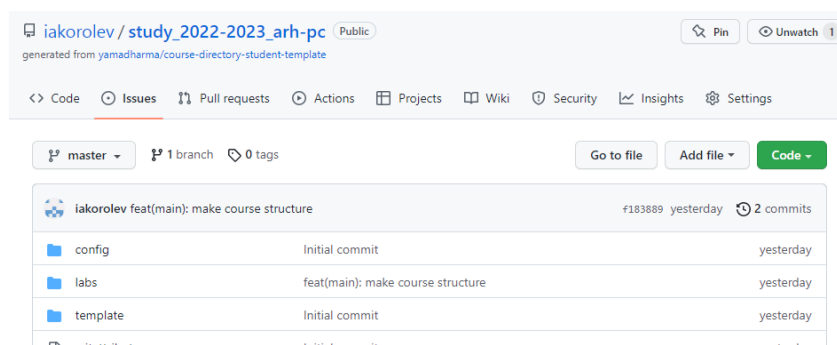


Рис. 2.18: Создал рабочее пространство

### 3 Выполнение самостоятельной работы:

Загружу все отчеты по выполненным работам в каталоги рабочего пространства(labs>lab01>report), и так для каждой лабораторной работы.(рис.19 ??),(рис.20 ??),(рис.21 ??)

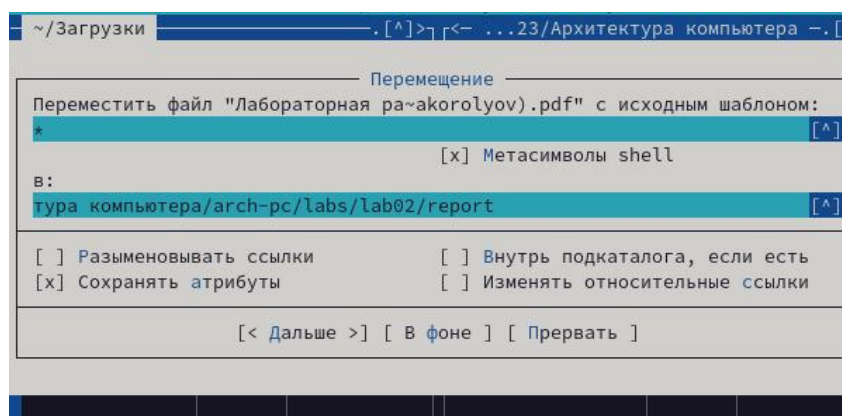


Рис. 3.1: Перемещаю загруженную лабораторную работу

```
[iakorolyov@fedora arch-pc]$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
Распаковка объектов: 100% (3/3), 740 байтов | 17.00 КиБ/с, готово.
Из github.com:iakorolev/study_2022-2023_arh-pc
7875363..a15df33 master -> origin/master
Обновление 7875363..a15df33
Fast-forward
 labs/lab01/report/Лабораторная работа №1(iakorolyov).docx | Bin 646042 -> 0 bytes
1 file changed, 0 insertions(+), 0 deletions(-)
delete mode 100644 labs/lab01/report/Лабораторная работа №1(iakorolyov).docx
[iakorolyov@fedora arch-pc]$ cd labs/lab01/report
[iakorolyov@fedora report]$ git add .
[iakorolyov@fedora report]$ git commit -am 'feat(main):make course structure'
[master 404b0f1] feat(main):make course structure
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 labs/lab01/report/Лабораторная работа №1(iakorolyov).docx
[iakorolyov@fedora report]$ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
Сжатие объектов: 100% (5/5), готово.
Запись объектов: 100% (5/5), 528 байтов | 31.00 КиБ/с, готово.
Всего 5 (изменений 3), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To github.com:iakorolev/study_2022-2023_arh-pc.git
a15df33..404b0f1 master -> master
[iakorolyov@fedora report]$ ls
bib image Makefile pandoc report.md 'Лабораторная работа №1(iakorolyov).docx'
```

Рис. 3.2: С помощью команд git pull, add, commit, push, загружаю все лабораторные в репозиторий








master ▾ study_2022-2023_arh-pc / labs / lab01 / report /	
 iakorolev feat(main):make course structure	
..	
 bib	feat(main): make course structure
 image	feat(main): make course structure
 pandoc/csl	feat(main): make course structure
 Makefile	feat(main): make course structure
 report.md	feat(main): make course structure
 Лабораторная работа №1(iakorolyov).docx	feat(main):make course structure

Рис. 3.3: Лабораторная работа

Такие действия, я проделываю для каждой лабораторной работы, загружаю все на github. Моя ссылка на репозиторий: [https://github.com/iakorolev/study\\_2022-2023\\_arh-pc](https://github.com/iakorolev/study_2022-2023_arh-pc)

## 4 Выводы

Я изучил идеологию и применение средств контроля версий. Приобрел практические навыки по работе с системой git.