

# **Шаблон отчёта по лабораторной работе**

**Простейший вариант**

Дмитрий Сергеевич Кулябов

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Теоретическое введение</b>	<b>6</b>
2.1	Системы контроля версий. Общие понятия . . . . .	6
2.2	Система контроля версий Git . . . . .	7
2.3	Основные команды git . . . . .	8
2.4	Стандартные процедуры работы при наличии центрального . . . . .	9
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>11</b>
3.1	Настройка github . . . . .	11
3.2	Базовая настройка git . . . . .	11
3.3	Создание SSH-ключа . . . . .	12
3.4	Создание рабочего пространства и репозитория курса на основе шаблона . . . . .	13
3.5	Создание репозитория курса на основе шаблона . . . . .	13
3.6	Настройка каталога курса . . . . .	14
3.7	Задание для самостоятельной работы . . . . .	16
<b>4</b>	<b>Выводы</b>	<b>18</b>

# Список иллюстраций

3.1	Созданный аккаунт в Github . . . . .	11
3.2	Команды с именем пользователя и e-mail . . . . .	11
3.3	Команда для настройки utf-8 . . . . .	12
3.4	Задание начальной ветки и установка параметров . . . . .	12
3.5	Генерация ключа . . . . .	12
3.6	Копирование ключа . . . . .	13
3.7	Подтверждение успешного добавления ключа . . . . .	13
3.8	Создание директории . . . . .	13
3.9	Проверка создания репозитория . . . . .	14
3.10	Переход в каталог . . . . .	14
3.11	Клонирование репозитория . . . . .	14
3.12	Переход в каталог . . . . .	15
3.13	Создание каталогов . . . . .	15
3.14	Проверка правильность создания . . . . .	16
3.15	Создание отчета по 2 лабораторной . . . . .	17
3.16	Копирование отчета по 2 лабораторной . . . . .	17
3.17	Загрузка файлов на github . . . . .	17

## **Список таблиц**

# 1 Цель работы

Цель данной лабораторной работы – изучить идеологии и применение средств контроля версий, и получить практические навыки по работе с системой контроля версий git.

## 2 Теоретическое введение

### 2.1 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 уужную 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.

## **2.2 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.

## 2.3 Основные команды git

Наиболее часто используемые команды git представлены в таблице.

Команда	Описание
<code>git init</code>	создание основного дерева репозитория
<code>git pull</code>	получение обновлений (изменений) текущего дерева из центрального репозитория
<code>git push</code>	отправка всех произведённых изменений локального дерева в центральный репозиторий
<code>git diff</code>	просмотр текущих изменений
<code>git add .</code>	добавить все изменённые и/или созданные файлы и/или каталоги
<code>git add</code>	добавить конкретные изменённые и или созданные файлы и или каталоги
<code>git rm</code>	удалить файл и/или каталог из индекса репозитория, при этом файл и/или каталог остаётся в локальной директории
<code>git commit -am</code>	сохранить все добавленные изменения и все изменённые файлы
<code>git checkout -b</code>	создание новой ветки, базирующейся на текущей
<code>git checkout</code>	переключение на некоторую ветку, при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана удалённой
<code>git push origin</code>	отправка изменений конкретной ветки в центральный репозиторий



Команда	Описание
<code>git merge --no-ff</code>	слияние ветки с текущим деревом
<code>git branch -d</code>	удаление локальной уже слитой с основным деревом ветки
<code>git branch -D</code>	принудительное удаление локальной ветки
<code>git push origin</code>	удаление ветки с центрального репозитория

## 2.4 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
```

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
```

## 3 Выполнение лабораторной работы

### 3.1 Настройка github

Создали учётную запись на сайте <https://github.com/> и заполнили основные данные.



Рисунок 3.1: Созданный аккаунт в Github

### 3.2 Базовая настройка git

Сначала сделаем предварительную конфигурацию git. Открыли терминал и ввели следующие команды, указав имя IvanovaAngelina и e-mail 1032252598@pfur.ru.

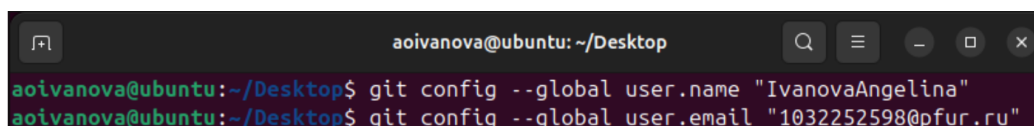


Рисунок 3.2: Команды с именем пользователя и e-mail

Настроили utf-8 в выводе сообщений git.

```
aoivanova@ubuntu:~$ git config --global core.quotePath false
```

Рисунок 3.3: Команда для настройки utf-8

Задали имя начальной ветки (назвали её master), а также установили параметры autocrlf и safecrlf.

```
aoivanova@ubuntu:~$ git config --global init.defaultBranch master
aoivanova@ubuntu:~$ git config --global core.autocrlf input
aoivanova@ubuntu:~$ git config --global core.safecrlf warn
```

Рисунок 3.4: Задание начальной ветки и установка параметров

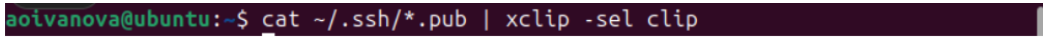
### 3.3 Создание SSH-ключа

Для последующей идентификации пользователя на сервере репозитория сгенерировали приватный и открытый ключи, которые сохранились в каталоге ~/.ssh/.

```
aoivanova@ubuntu:~$ ssh-keygen -C "IvanovaAngelina 1032252598@pfur.ru"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/aoivanova/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/aoivanova/.ssh/id_ed25519
Your public key has been saved in /home/aoivanova/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:VzXcEgJS7CK6CPay+0k0mCytpKxQQSkIvy0gh93cXao IvanovaAngelina 1032252598@pfur.ru
The key's randomart image is:
+--[ED25519 256]--+
|+ .. .oo...+o |
|oo. . . oo..|
|..o . . . |
|oo = ...o..|
|+ * o...S..|
|oO .. . .|
|*+=. .E|
|B+.+.|
|*=*|
+----[SHA256]-----+
```

Рисунок 3.5: Генерация ключа

Загрузили сгенерированный открытый ключ. Зашли на сайт <http://github.org/> под своей учётной записью и перешли в меню Setting . После этого выбрали в боковом меню SSH and GPG keys и нажали кнопку New SSH key. Скопировали из локальной консоли ключ в буфер обмена



```
aoivanova@ubuntu:~$ cat ~/.ssh/*.pub | xclip -sel clip
```

Рисунок 3.6: Копирование ключа

Вставили ключ в появившееся на сайте поле и указали для ключа имя IvanovaAngelina. После чего нажали на кнопку Add SSH Key. Таким образом, успешно добавили ключ.

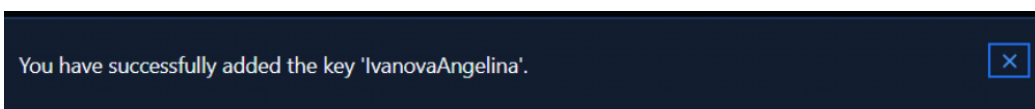
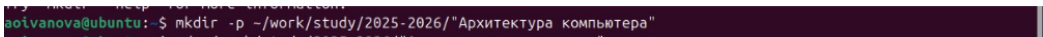


Рисунок 3.7: Подтверждение успешного добавления ключа

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

Открыли терминал и создали каталог для предмета «Архитектура компьютера»



```
aoivanova@ubuntu:~$ mkdir -p ~/work/study/2025-2026/"Архитектура компьютера"
```

Рисунок 3.8: Создание директории

### 3.5 Создание репозитория курса на основе шаблона

Перешли на страницу репозитория с шаблоном курса <https://github.com/yamadharma/course-directory-student-template>. Далее выбрали Use this template. В открывшемся окне задали имя репозитория study\_2025–2026\_arh-ps и создали репозиторий, нажав кнопку „Create repository from template“

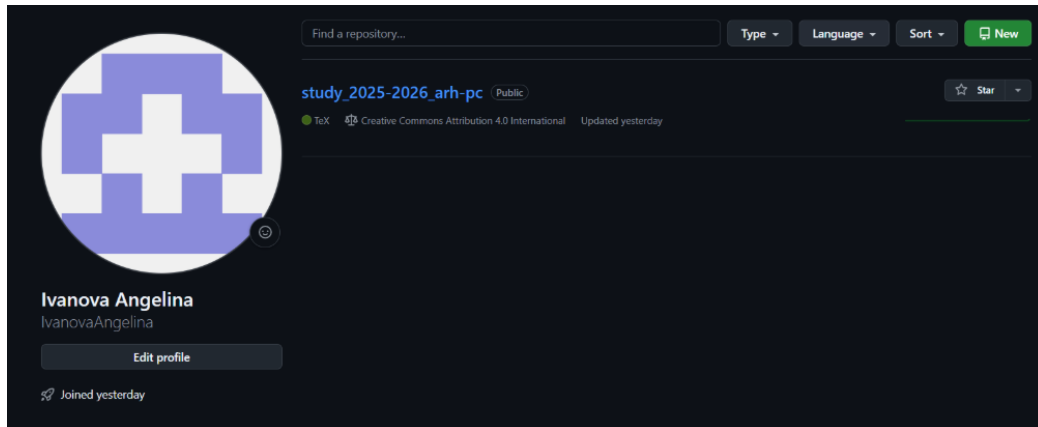


Рисунок 3.9: Проверка создания репозитория

Открыли терминал и перешли в каталог курса

```
aoivanova@ubuntu:~$ cd ~/work/study/2025-2026/"Архитектура компьютера"
```

Рисунок 3.10: Переход в каталог

Клонировали созданный репозиторий

```
aoivanova@ubuntu:~/work/study/2025-2026/Архитектура компьютера$ git clone --recursive git@github.com:IvanovaAngelina/study_2025-2026_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:+D1Y3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvC0qU.
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: 38, done.
remote: Counting objects: 100% (38/38), done.
remote: Compressing objects: 100% (36/36), done.
Receiving objects: 100% (38/38), 23.46 KiB | 5.86 MiB/s, done.
Resolving deltas: 100% (1/1), done.
remote: Total 38 (delta 1), reused 27 (delta 1), pack-reused 0 (from 0)
Submodule 'template/presentation' (https://github.com/yamadharma/academic-presentation-markdown-template.git) registered for path 'template/presentation'
Submodule 'template/report' (https://github.com/yamadharma/academic-laboratory-report-template.git) registered for path 'template/report'
Cloning into '/home/aoivanova/work/study/2025-2026/Архитектура компьютера/arch-pc/template/presentation'...
remote: Enumerating objects: 161, done.
remote: Counting objects: 100% (161/161), done.
remote: Compressing objects: 100% (111/111), done.
remote: Total 161 (delta 60), reused 142 (delta 41), pack-reused 0 (from 0)
Receiving objects: 100% (161/161), 2.65 MiB | 4.05 MiB/s, done.
Resolving deltas: 100% (60/60), done.
Cloning into '/home/aoivanova/work/study/2025-2026/Архитектура компьютера/arch-pc/template/report'...
remote: Enumerating objects: 221, done.
```

Рисунок 3.11: Клонирование репозитория

## 3.6 Настройка каталога курса

Перешли в каталог курса.

```
aoivanova@ubuntu: ~/work/study/2025-2026/Архитектура компьютера$ cd arch-pc
```

Рисунок 3.12: Переход в каталог

Создали необходимые каталоги.

```
aoivanova@ubuntu: ~/work/study/2025-2026/Архитектура компьютера/arch-pc$ echo arch-pc > COURSE
aoivanova@ubuntu: ~/work/study/2025-2026/Архитектура компьютера/arch-pc$ make prepare
```

Рисунок 3.13: Создание каталогов

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

```
aoivanova@ubuntu: ~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git add .
aoivanova@ubuntu: ~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git commit -m 'feat(main): make course structure'
[master a01413a] feat(main): make course structure
212 files changed, 8074 insertions(+), 207 deletions(-)
delete mode 100644 CHANGELOG.md
create mode 100644 labs/README.md
create mode 100644 labs/README.ru.md
create mode 100644 labs/lab01/presentation/.gitignore
create mode 100644 labs/lab01/presentation/.marksmen.toml
create mode 100644 labs/lab01/presentation/.projectile
create mode 100644 labs/lab01/presentation/Makefile
create mode 100644 labs/lab01/presentation/_quarto.yml
create mode 100644 labs/lab01/presentation/_resources/image/logo_rudn.png
create mode 100644 labs/lab01/presentation/arch-pc--lab01--presentation.qmd
create mode 100644 labs/lab01/presentation/image/kulyabov.jpg
create mode 100644 labs/lab01/report/.gitignore
create mode 100644 labs/lab01/report/.marksmen.toml
create mode 100644 labs/lab01/report/.projectile
create mode 100644 labs/lab01/report/Makefile
create mode 100644 labs/lab01/report/_quarto.yml
create mode 100644 labs/lab01/report/_resources/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab01/report/arch-pc--lab01--report.qmd
create mode 100644 labs/lab01/report/bib/cite.bib
create mode 100644 labs/lab01/report/image/solvay.jpg
create mode 100644 labs/lab02/presentation/.gitignore
create mode 100644 labs/lab02/presentation/.marksmen.toml
create mode 100644 labs/lab02/presentation/.projectile
create mode 100644 labs/lab02/presentation/Makefile
create mode 100644 labs/lab02/presentation/_quarto.yml
create mode 100644 labs/lab02/presentation/_resources/image/logo_rudn.png
create mode 100644 labs/lab02/presentation/arch-pc--lab02--presentation.qmd
create mode 100644 labs/lab02/presentation/image/kulyabov.jpg

aoivanova@ubuntu: ~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git push
Command 'push' not found, did you mean:
  command 'pwsh' from snap powershell (7.5.3)
  command 'posh' from deb posh (0.14.1)
  command 'rush' from deb rush (2.3.1)
  command 'ppsh' from deb pps (1.10-3build2)
  command 'pdsh' from deb pdsh (2.34-2)
See 'snap info <snapname>' for additional versions.
aoivanova@ubuntu: ~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git push
Enumerating objects: 67, done.
Counting objects: 100% (67/67), done.
Compressing objects: 100% (52/52), done.
Writing objects: 100% (64/64), 700.10 KiB | 3.35 MiB/s, done.
Total 64 (delta 23), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (23/23), completed with 1 local object.
To github.com:IvanovaAngelina/study_2025-2026_arh-pc.git
   bc1064b..a01413a master -> master
aoivanova@ubuntu: ~/work/study/2025-2026/Архитектура компьютера/arch-pc$
```

Рисунок 3.14: Проверка правильность создания

## 3.7 Задание для самостоятельной работы

Создали отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства (labs/lab02/report).



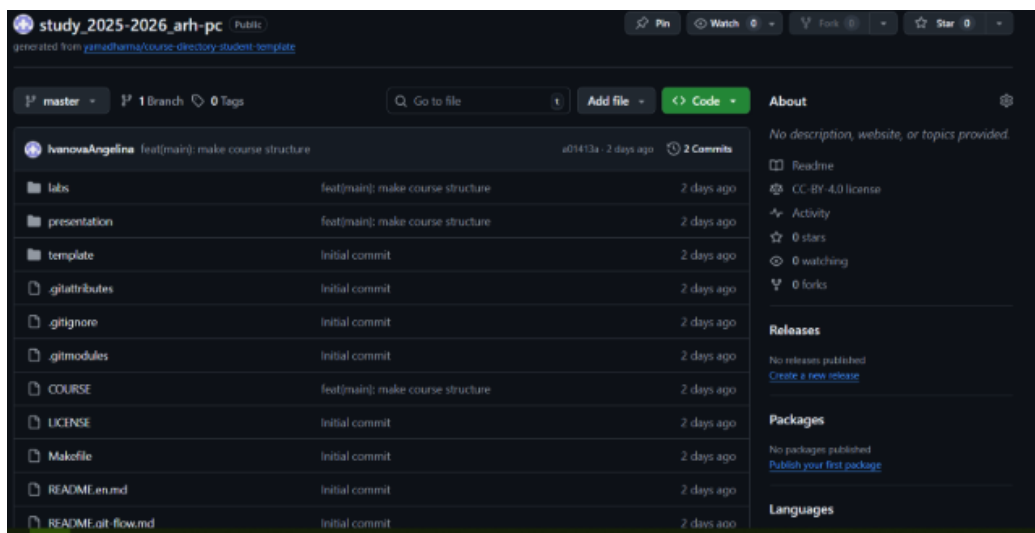


Рисунок 3.15: Создание отчета по 2 лабораторной

Скопировали отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданного рабочего пространства с помощью команды `mc`

```
aoivanova@ubuntu:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ cd labs/lab02/report
aoivanova@ubuntu:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab02/report$ touch 'lab 2.pdf'
```

Рисунок 3.16: Копирование отчета по 2 лабораторной

Загрузили файлы на github.

```
aoivanova@ubuntu:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab01/report$ ls
arch-pc--lab01--report.qmd bib image lab1.pdf Makefile _quarto.yml _resources
```

Рисунок 3.17: Загрузка файлов на github

## 4 Выводы

Изучили идеологии и применение средств контроля версий, и получили практические навыки по работе с системой контроля версий git. Создали аккаунт github, связали со своим устройством, создали рабочее пространство и репозиторий и загрузили отчеты по первым лабораторным работам. ::: {#refs} :::