

Отчет по лабораторной работе №2

Дисциплина: Архитектура компьютера

Максимова Дарья Валерьевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Выводы	13

Список иллюстраций

4.1	Создание учетной записи на github	9
4.2	Предварительная конфигурация	9
4.3	Настройка utf-8	9
4.4	Генерация пары ключ	10
4.5	Добавление ключа	10
4.6	Создание каталога	10
4.7	Создание репозитория	11
4.8	Клонирование	11
4.9	Удаление лишнего; создание нужного	12
4.10	Создание нужных каталогов	12

Список таблиц

1 Цель работы

Целью данной работы является изучение идеологии и применение средств контроля версий, а также приобрести практические навыки работы с системой git.

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. Выполнение заданий для самостоятельной работы.

3 Теоретическое введение

Системы управления версией (Система управления версией, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удаленном репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведенные 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 управления версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы с опреем определенных 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 измененных файлов, а производить так называемое дельта-сжатие — сохранять только изменения между последовательными версиями, что позволяет уменьшить объем хранимых данных. Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить изменения, сделанные разными участниками, вручную выбрать нужную версию, отменить изменения или заблокировать файлы для изменения. В зависимости от настро-

ек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла с средствами файловой системы ОС, обеспечивая таким образом привилегированный доступ только одному пользователю, работающему с файлом. Системы управления версиями 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. Обычно доступна информация о том, кто из участников, когда и какие изменения вносили. Обычно такая информация хранится в журнале изменений, доступ к которому можно ограничить. В отличие от классических, в распределённых системах управления версиями центральный репозиторий не является обязательным. Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы похожи, они отличаются в основном синтаксисом используемых в работе команд. Система управления версиями Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала, введя `git`-коллама с различными опциями. Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией. 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 быть внесено изменений). Затем можно внести изменения в локальное дерево и/или ветку. После внесения каких-либо изменений в файлы и/или каталоги проекта необходимо разместить их в центральном репозитории.

4 Выполнение лабораторной работы

Создаю учетную запись на сайте <https://github.com/> и заполняю основные данные. (рис. [-fig:001])

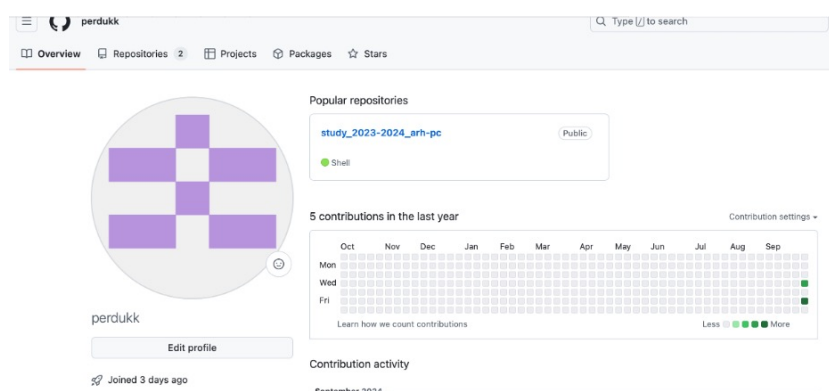


Рис. 4.1: Создание учетной записи на github

Делаю предварительную конфигурацию git, указав имя и email владельца репозитория (рис. [-fig:002])

```
((base) dashamaximova@MacBook-Dasha ~ % git config --global user.email "<dashka.maxsimova101105@gmail.com>"  
((base) dashamaximova@MacBook-Dasha ~ % git config --global user.name "<Дарья Максимова>"
```

Рис. 4.2: Предварительная конфигурация

Настроим utf-8 в выводе сообщений git, зададим имя начальной ветке(будем называть её master), укажем значение параметров autocrlf и safecrlf (рис. [-fig:003])

```
((base) dashamaximova@MacBook-Dasha ~ % git config --global core.quotepath false  
((base) dashamaximova@MacBook-Dasha ~ % git config --global init.defaultBranch master  
((base) dashamaximova@MacBook-Dasha ~ % git config --global core.autocrlf input  
((base) dashamaximova@MacBook-Dasha ~ % git config --global core.safecrlf warn
```

Рис. 4.3: Настройка utf-8

Для последующей идентификации пользователя на сервере репозитория сгенерируем пару ключей (приватный и открытый) (рис. [-fig:004])

```
(base) dashamaximova@MacBook-Dasha ~ % ssh-keygen -C "Дарья Максимова <dashka.maksimova101105@gmail.com>"  
Generating public/private rsa key pair.
```

Рис. 4.4: Генерация пары ключ

(у меня уже был сгенерирован код) Имеющийся ключ я загрузила на github, загрузив его в буфер обмена. Вставляю скопированный ключ в поле «Ключ». В поле Название указываю имя для ключа. Нажимаю «Добавить SSH-ключ», чтобы завершить добавление ключа. (рис. [-fig:005])

```
(base) dashamaximova@MacBook-Dasha ~ % cat ~/.ssh/id_rsa.pub  
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGD1T1D0P4aBc2pF8MUP7hucLd7AuYF8L4u0mly0RyUeDhARPOjwvSEYH3u05vca3n8c2DP7h8u8c3h8d8c28MFu0uY72u8wz3h8Gyy1318p4b3a438dva5jDc348718y1VhAc5y0P7p8Aey1V8yUu710Pp3  
uDXwK71L8RPP71p0e023K1z8h7w12758W9757F8q2Z0h8m8Z7uAaK1A8wWuYpF8MGTzL1uY07ShLqAK2c4SRE8b0hFhF5/8A71a4Eag818qzSP728u18m0V10j0c11JEKyy8b2z8F8Bp+8c8p78a0g80P8b2h8y8u8d8G8C8y8g8h8126f8m058b214  
88m8p8j08C1M8P28p8c218b8M8d8c11v8b2+ac8B8G8a8D8Y8S88u72u5j8Y8P+u378uFu378uF358e8c8u8T78au8138a81Q8v8N8Y8+78u7188u8D8P8L8u88+D8p8a8Maximova<dashka.maksimova101105@gmail.com>
```

Рис. 4.5: Добавление ключа

Создадим каталог для предмета «Архитектура компьютера» для последующего создания рабочего пространства. (рис. [-fig:006])

```
(base) dashamaximova@MacBook-Dasha ~ % mkdir -p ~/work/study/2023-2024/Архитектура_компьютера"
```

Рис. 4.6: Создание каталога


Через web-интерфейс github создадим репозиторий на основе шаблона, указав имя study_2024-2025_arh-pc (рис. [-fig:007])

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (*).

Repository template

 yamadharma/course-directory-student-template ▾

Start your repository with a template repository's contents.

☐ Include all branches

Copy all branches from yamadharma/course-directory-student-template and not just the default branch.

Repository

owner and name
perdukk ▾

Repository name *

study_2023-2024_arh-pc

✓ Your new repository will be created as study_2023-2024_arh-pc.

The repository name can only contain ASCII letters, digits, and the characters ., -, and _.

Great repository names are short and memorable. Need inspiration? How about [didactic-pancake](#) ?

Description (optional)

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

 You are creating a public repository in your personal account.

Create repository

Рис. 4.7: Создание репозитория

Перейдем в каталог курса и скопируем в него созданный репозиторий с помощью ссылки для клонирования: (рис. [-fig:008])

```
(base) dashamaximova@MacBook-Dasha Архитектура_компьютера % git clone --recursive git@github.com:perdukk/:
Cloning into 'study_2023-2024_arh-pc'...
The authenticity of host 'github.com (140.82.121.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdKr4UvC0qU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? dsf
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
remote: Enumerating objects: 33, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (32/32), done.
remote: Total 33 (delta 1), reused 18 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (33/33), 18.82 KiB | 448.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.
Submodule 'template/presentation' (https://github.com/yamadharma/academic-presentation-markdown-template.git) regist
Submodule 'template/report' (https://github.com/yamadharma/academic-laboratory-report-template.git) regist
Cloning into '/Users/dashamaximova/work/study/2023-2024/Архитектура_компьютера/study_2023-2024_arh-pc/tem
remote: Enumerating objects: 111, done.
remote: Counting objects: 100% (111/111), done.
remote: Compressing objects: 100% (77/77), done.
remote: Total 111 (delta 42), reused 100 (delta 31), pack-reused 0 (from 0)
Receiving objects: 100% (111/111), 102.17 KiB | 1.22 MiB/s, done.
Resolving deltas: 100% (42/42), done.
Cloning into '/Users/dashamaximova/work/study/2023-2024/Архитектура_компьютера/study_2023-2024_arh-pc/tem
remote: Enumerating objects: 142, done.
remote: Counting objects: 100% (142/142), done.
remote: Compressing objects: 100% (97/97), done.
remote: Total 142 (delta 60), reused 121 (delta 39), pack-reused 0 (from 0)
Receiving objects: 100% (142/142), 341.09 KiB | 1.46 MiB/s, done.
Resolving deltas: 100% (60/60), done.
Submodule path 'template/presentation': checked out 'c9b2712b4b2d431ad5086c9c72a02bd2fca1d4a6'
Submodule path 'template/report': checked out 'c26e22effe7b3e0495707d82ef561ab185f5c748'
(base) dashamaximova@MacBook-Dasha Архитектура_компьютера % ls
```

Рис. 4.8: Клонирование

Перейдём в каталог курса, удалим лишние файлы, создадим нужные каталоги и загрузим файлы на сервер: (рис. [-fig:009])

```
((base) dashamaximova@MacBook-Dasha Архитектура_компьютера % cd arch-pc
((base) dashamaximova@MacBook-Dasha arch-pc % rm package.json
```

Рис. 4.9: Удаление лишнего; создание нужного

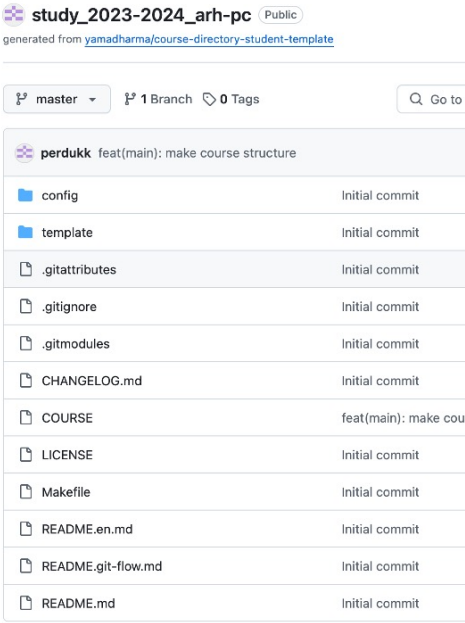
Создание нужных каталогов: (рис. [-fig:010])

```
((base) dashamaximova@MacBook-Dasha arch-pc % echo arch-pc > COURSE
((base) dashamaximova@MacBook-Dasha arch-pc % make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare       Generate directories structure
  submodule     Update submules

((base) dashamaximova@MacBook-Dasha arch-pc % git add .
```

Рис. 4.10: Создание нужных каталогов



Проверим правильность введенных команд: (рис. [-fig:011])

5 Выводы

В ходе выполнения этой я исследовала концепции и познакомилась с использованием систем контроля версий, а также приобрела практические навыки работы с git.