

Отчёт по лабораторной работе №2

Дисциплина: архитектура компьютера

Хакдан Сабуров

1 Цель работы

Целью данной работы является изучить идеологию и применение средств контроля версий, а также приобрести практические навыки по работе с системой git.

2 Задание

- Базовая настройка Git.
- Создание SSH-ключа.
- Создание рабочего пространства и репозитория курса на основе шаблона.
- Создание репозитория курса на основе шаблона.
- Настройка каталога курса.
- Выполнение заданий для самостоятельной работы.

3 Теоретическое введение

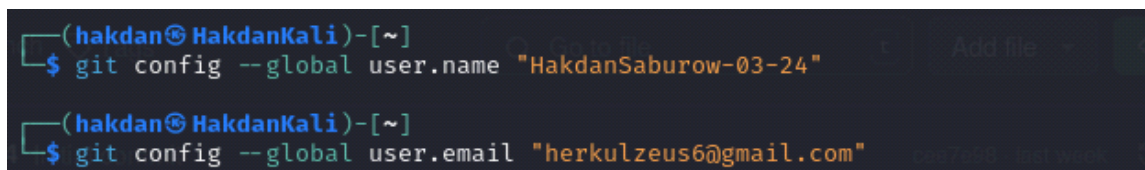
Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется. 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 изменения,

сделанные разными участниками, 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. 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. 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. 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). 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.

4 Выполнение лабораторной работы

4.1 Базовая настройка Git

Открываю виртуальную машину, затем открываю терминал и делаю предварительную конфигурацию git. Ввожу команду `git config --global user.name ""`, указывая свое имя и команду `git config --global user.email "work@mail"`, указывая в ней электронную почту владельца, то есть мою (рис. 1).

A screenshot of a terminal window with a dark background. The prompt is `(hakdan@HakdanKali)~`. The first command entered is `$ git config --global user.name "HakdanSaburow-03-24"`. The second command entered is `$ git config --global user.email "herkulzeus6@gmail.com"`.

```
(hakdan@HakdanKali)~  
$ git config --global user.name "HakdanSaburow-03-24"  
  
(hakdan@HakdanKali)~  
$ git config --global user.email "herkulzeus6@gmail.com"
```

Рис. 1: Предварительная конфигурация git

Настраиваю utf-8 в выводе сообщений git для корректного отображения символов (рис. 2).

```
(hakdan@HakdanKali)-[~]  
$ git config --global core.quotePath false
```

Рис. 2: Настройка кодировки

Задаю имя «master» для начальной ветки (рис. 3).

```
(hakdan@HakdanKali)-[~]  
$ git config --global init.defaultBranch master
```

Рис. 3: Создание имени для начальной ветки

Задаю параметр autocrlf со значением input, так как я работаю в системе Linux, чтобы конвертировать CRLF в LF только при коммитах (рис. 4). CR и LF – это символы, которые можно использовать для обозначения разрыва строки в текстовых файлах.

```
(hakdan@HakdanKali)-[~]  
$ git config --global core.autocrlf input
```

Рис. 4: Параметр autocrlf

Задаю параметр safecrlf со значением warn, так Git будет проверять преобразование на обратимость (рис. 5). При значении warn Git только выведет предупреждение, но будет принимать необратимые конвертации.

```
(hakdan@HakdanKali)-[~]  
$ git config --global core.safecrlf warn
```

Рис. 5: Параметр safecrlf

4.2 Создание SSH-ключа

Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый). Для этого ввожу команду `ssh-keygen -C "Имя Фамилия, work@email"`, указывая имя владельца и электронную почту владельца (рис. 6). Ключ автоматически сохранится в каталоге `~/.ssh/`.

Копирую открытый ключ из директории, в которой он был сохранен, с помощью утилиты xclip (рис. 8).

```
(hakdan@HakdanKali)-[~]  
$ cat ~/.ssh/id_ed25519.pub | xclip -sel clip
```

Рис. 8: Копирование содержимого файла

Открываю браузер, захожу на сайт GitHub. Открываю свой профиль и выбираю страницу «SSH and GPG keys». Нажимаю кнопку «New SSH key» (рис. 9).

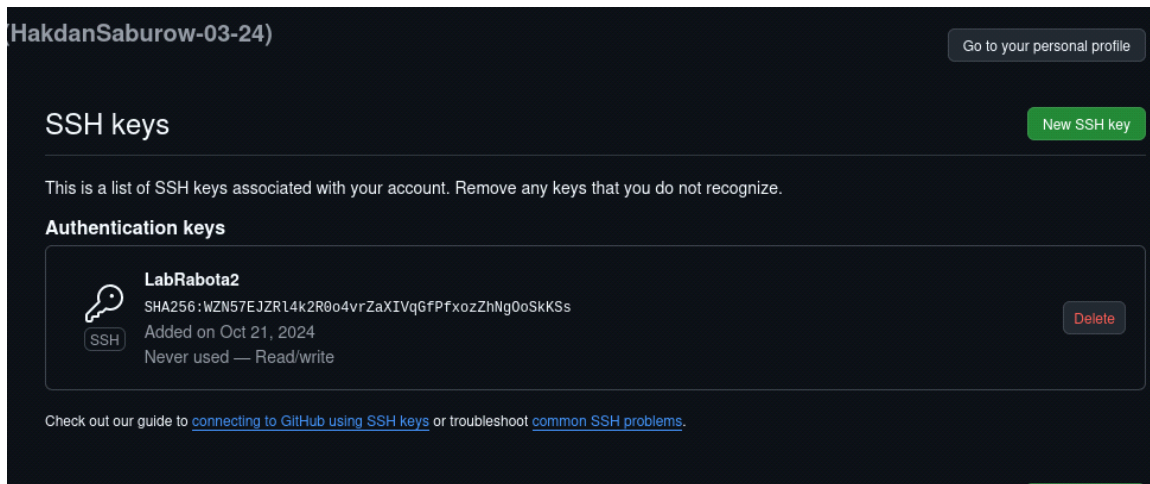


Рис. 9: Окно SSH and GPG keys

Вставляю скопированный ключ в поле «Key». В поле Title указываю имя для ключа. Нажимаю «Add SSH-key», чтобы завершить добавление ключа (рис. 10).

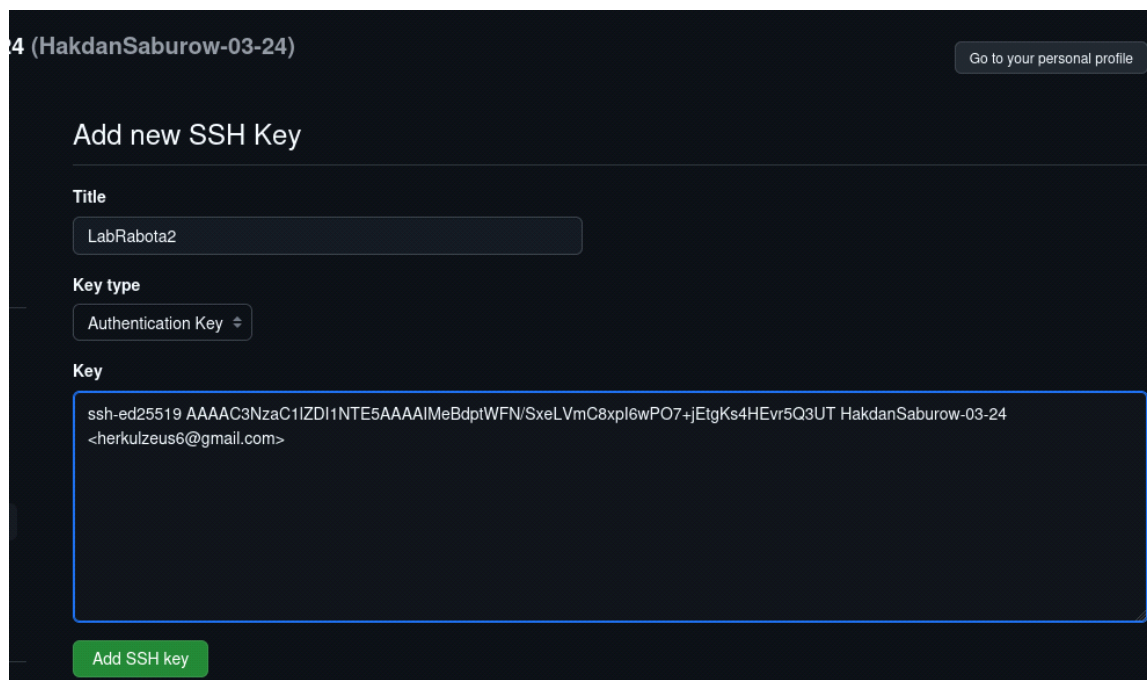


Рис. 10: Добавление ключа

4.3 Создание рабочего пространства и репозитория курса на основе шаблона

Закрываю браузер, открываю терминал. Создаю директорию, рабочее пространство, с помощью утилиты `mkdir`, благодаря ключу `-p` создаю все директории после домашней `~/work/study/2022-2023/` "Архитектура компьютера" рекурсивно. Далее проверяю с помощью `ls`, действительно ли были созданы необходимые мне каталоги (рис. 11).

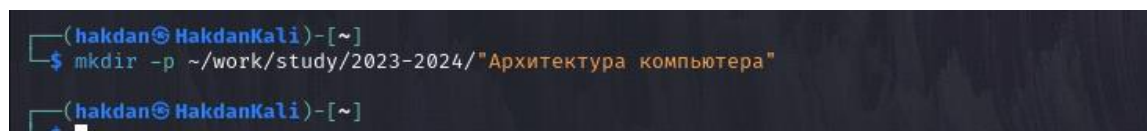


Рис. 11: Создание рабочего пространства

4.4 Создание репозитория курса на основе шаблона

В браузере перехожу на страницу репозитория с шаблоном курса по адресу <https://github.com/yamadharm/course-directory-student-template>. Далее выбираю «Use this template», чтобы использовать этот шаблон для своего репозитория (рис. 12).

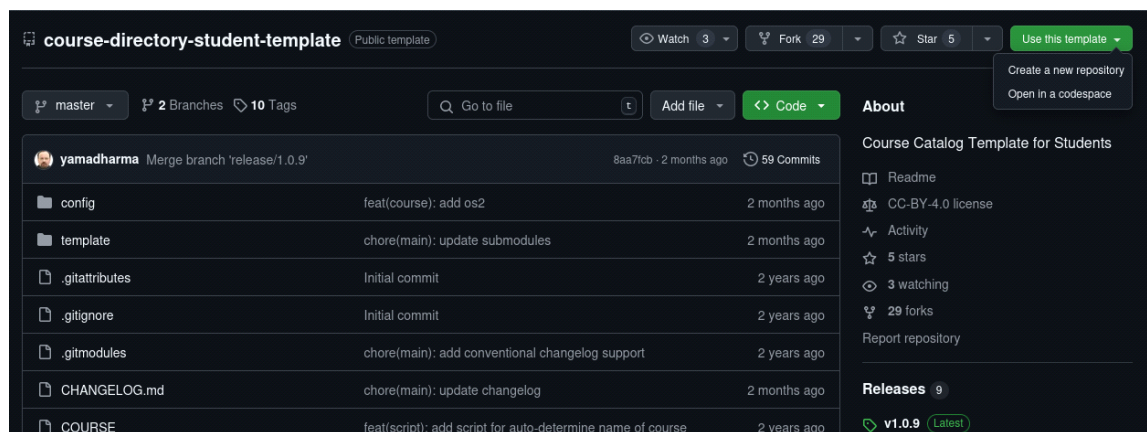


Рис. 12: Страница шаблона для репозитория


В открывшемся окне задаю имя репозитория (Repository name): study_2022–2023_arh-рс и создаю репозиторий, нажимаю на кнопку «Create repository from template» (рис. 13).

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

 yamadharm/course-directory-student-template ▾

Start your repository with a template repository's contents.

☐ **Include all branches**
Copy all branches from yamadharm/course-directory-student-template and not just the default branch.


Repository **Repository name ***


 HakdanSaburow-03-24 ▾ / study_2023-2024_arhpc


✔ study_2023-2024_arhpc is available.

Great repository names are short and memorable. Need inspiration? How about **super-engine** ?

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. 13: Окно создания репозитория

Через терминал перехожу в созданный каталог курса с помощью утилиты `cd` (рис. 14).

```
(hakdan@HakdanKali) - [~/work/study/2023-2024/Архитектура компьютера]
$ cd ~/work/study/2023-2024/"Архитектура компьютера"
```

Рис. 14: Перемещение между директориями

Клонирую созданный репозиторий с помощью команды `git clone -recursive git@github.com:/study_2022-2023_arh-pc.git arch-pc` (рис. 15).


```
(hakdansaburow@Hakdan)-[~/work/study/2023-2024/Архитектура компьютера]
$ git clone --recursive git@github.com:HakdanSaburow-03-24/study_2023-2024_arhpc
Cloning into 'study_2023-2024_arhpc'...
The authenticity of host 'github.com (140.82.121.4)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdKr4UvCOqU.
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: 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)
```

Рис. 15: Клонирование репозитория

4.5 Настройка каталога курса

Перехожу в каталог arch-pc с помощью утилиты cd (рис. 16).

```
(hakdan@HakdanKali)-[~/work/study/2023-2024/Архитектура компьютера]
$ cd ~/work/study/2023-2024/"Архитектура компьютера"/study_2023-2024_arhpc/
```

Рис. 16: Перемещение между директориями

Удаляю лишние файлы с помощью утилиты rm (рис. 17).

```
(hakdan@HakdanKali)-[~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc]
$ rm package.json
```

Рис. 17: Удаление файлов

Создаю необходимые каталоги (рис. 18).

```
(hakdan@HakdanKali)-[~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc]
$ echo arch-pc > COURSE

(hakdan@HakdanKali)-[~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc]
```

Рис. 18: Создание каталогов

Отправляю все на сервер с помощью push (рис. 19).

```
(hakdan@HakdanKali)-[~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc]
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 7 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 297 bytes | 297.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:HakdanSaburow-03-24/study_2023-2024_arhpc.git
f7776fc..5b1f86b master -> master
```

Рис. 19: Выгрузка изменений на сервер

