Отчет по лабораторной работе №2

Дисциплина: Архитектура компьютера

Краснопер Данила Олегович

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Целью данной лабораторной работы является получение практических навыков работы с системой Git при помощи командной строки. В ходе неё мы я изучить идеологию и применение средств контроля версий.

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 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 позво-

ляет 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

Создаю учетную запись на сайте Github. Заполняю основные данные для учетной записи. (Рис. 4.1)

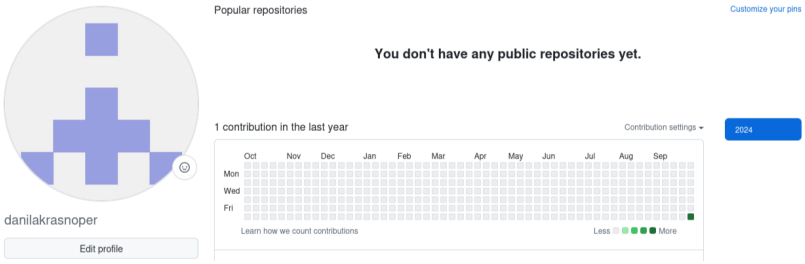

Рис. 4.1: Создание учетной записи

Открываем терминал, делаю конфигурацию git. Вводим команду git config –global user.name и указываем имя, следом вводим git config –global user.email, вписывая электронную почту аккаунта github. Настраиваем utf- в выводе сообщений git. Задаем имя для начальной ветки. Она будет называться «master». Задаем параметры autocrlf и safecrlf для корректного выполнения команд. (Рис.4.2)

```
user@debian:~$ git config --global user.name "<danilakrasnoper>"
user@debian:~$ git config --global user.email "<danila11082006@gmail.com>"
user@debian:~$ git config --global init.defaultBranch master
user@debian:~$ git config --global core.autocrlf input
user@debian:~$ git config --global core.safecrlf warn
user@debian:~$
```

Рис. 4.2: Базовая настройка git.

Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый). Для этого

вводим в командную строку команду ssh-keygen –C “”, вводя имя пользователя и электронную почту. (Рис. 4.3)

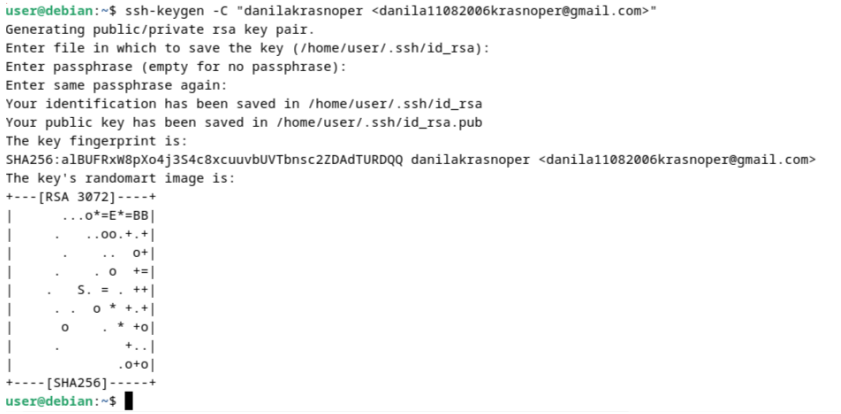

Рис. 4.3: Генерация ключа SSH.

Загружаем сгенерированный открытый ключ. Заходим на сайт под своей учетной записью, переходим в меню, находим меню ключей и создаем новый ключ. Копируем из локальной консоли ключ в буфер обмена и используем команду cat. (Рис. 4.4)

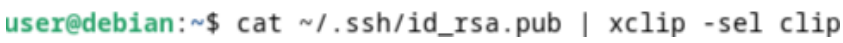

Рис. 4.4: Команда cat.

Вставляем ключ в поле сайта и указываем имя для ключа. (Рис. 4.5)

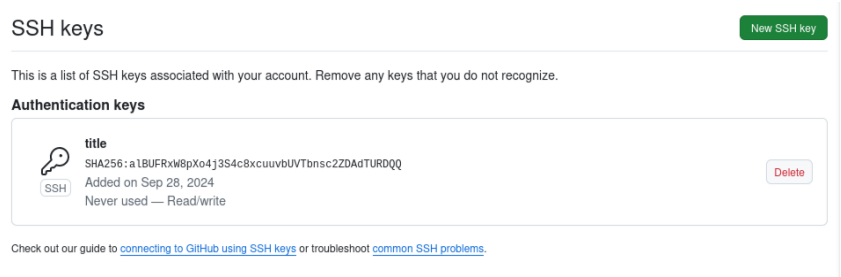

Рис. 4.5: Добавление ключа.

Создание рабочего пространства и репозитория курса на основе шаблона. Открываем терминал и создаем каталог для предмета “Архитектура компьютера”. (Рис. 4.6)

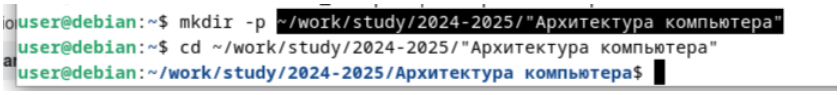

Рис. 4.6: Создание каталога.

Создание репозитория курса на основе шаблона. Переходим на станицу github в репозиторий с шаблоном курса https://github.com/yamadharma/course-directorystudent-template. Далее выбираем “Use this template”. В открывшемся окне задаем имя репозитория. Создаем репозиторий. Проверяем создался ли репозиторий. (Рис. 4.7)

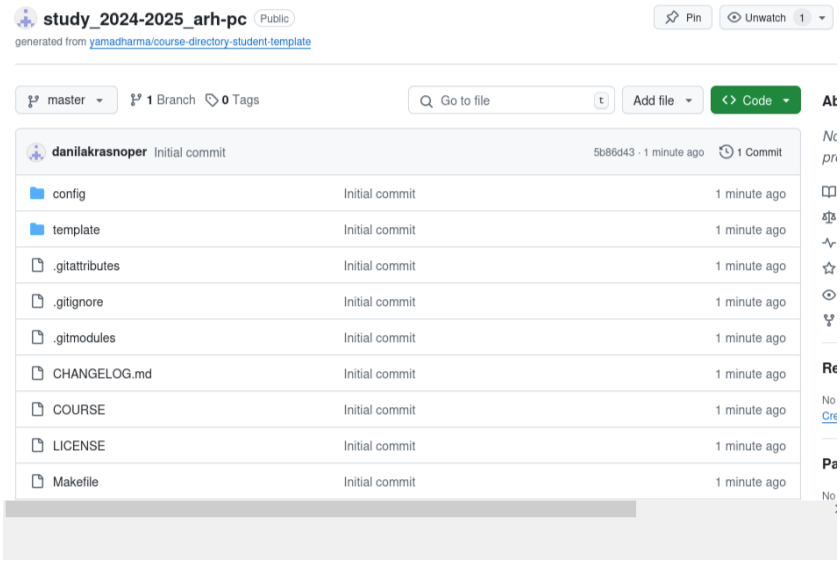

Рис. 4.7: Создание репозитория

Открываем терминал и переходим в каталог курса при помощи cd и клонируем созданный репозиторий и помощи команды git clone –recursive ’’. (Рис. 4.8)

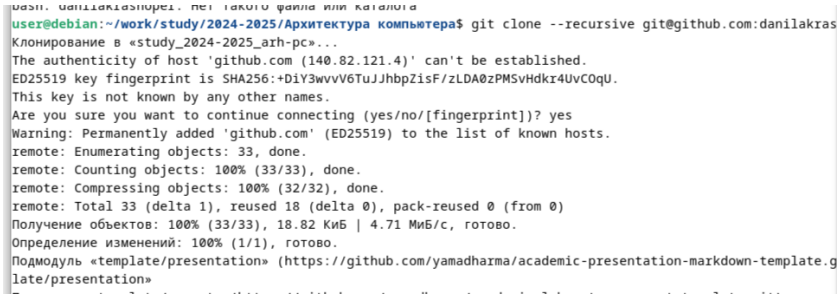

Рис. 4.8: Клонирование репозитория.

Копируем ссылку для клонирования на странице созданного репозитория. Переходим в окно Code, следом в SSH. (Рис. 4.9)

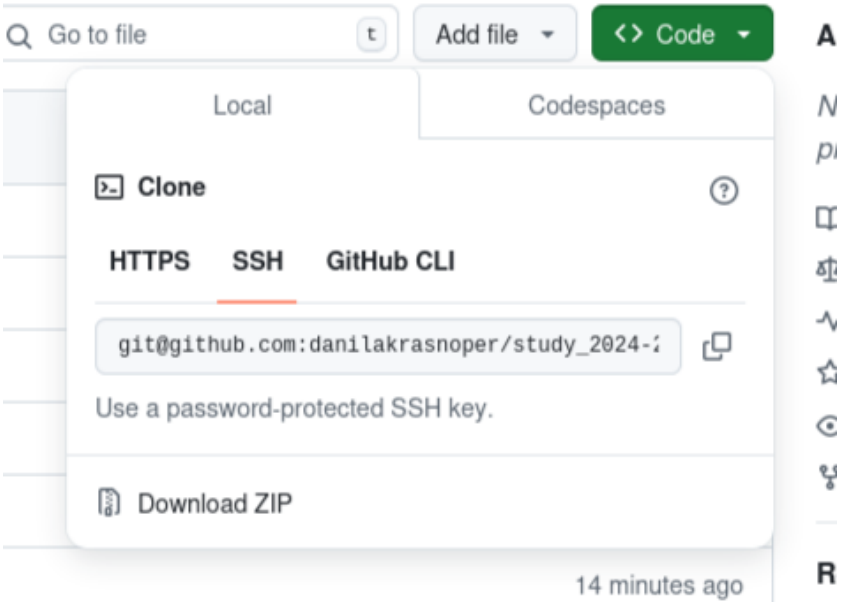

Рис. 4.9: Ссылка для копирования.

Настройка каталога курса. Переходим в каталог курса при помощи cd. Удаляем лишние файлы. (Рис. 4.10)

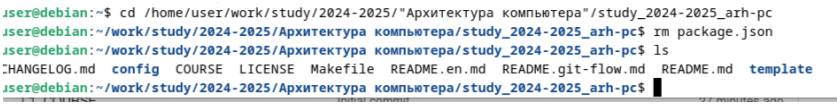

Рис. 4.10: Удаление Файлов

Создаем необходимые каталоги. Отправляем файлы на сервер используя команды git add ., git commit –am и git push. (Рис. 4.11)

Рис. 4.11: Отправка файлов.

Проверяем правильность создания иерархии рабочего пространства в локальном репозитории и на странице github. (Рис. 4.12)

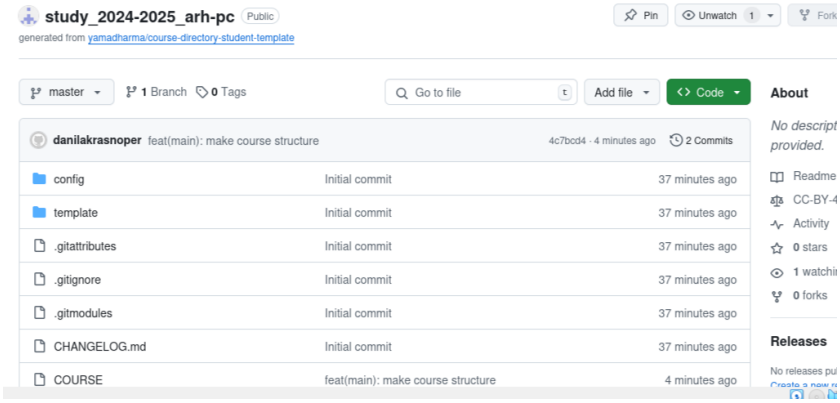

Рис. 4.12: Страница рабочего пространства.

5 Выполнение заданий для самостоятельной работы

Переходим в labs/lab02/report с помощью cd. Создаем файл для отчета по лабораторной работе, копируем первую лабораторную с помощью и проверяем правильность выполнения команды. Переходим в подкаталог lab02/report. Копируем вторую лабораторную в каталог. (Рис 5.1)

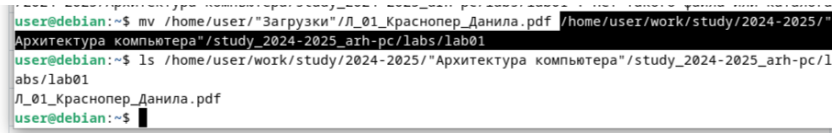

Рис. 5.1: Копирование.

При помощи команды git add ’’ добавляем новые файлы. Сохраняем изменение при помощи команды git commit – am. Переносим в репозиторий сохраненные изменения командой git push. (Рис. 5.2)

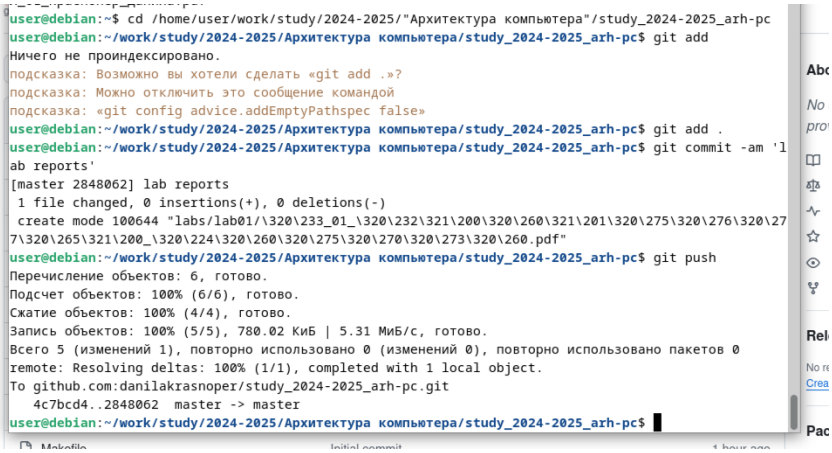

Рис. 5.2: Сохранение и перенос изменений.

Проверяем на сайте правильность выполнения заданий. (Рис. 5.3)

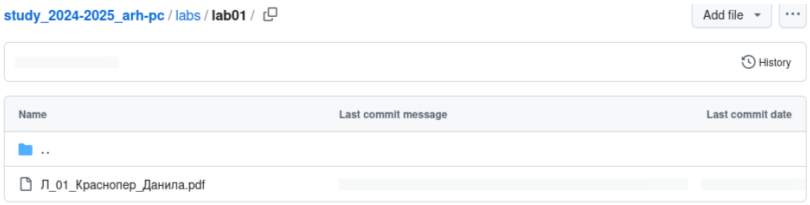

Рис. 5.3: Проверка.

6 Вывод

При выполнении данной лабораторной работы я изучил идеологию и применение средств контроля версий, а также приобрел практические навыки по работе с системой git.