Отчёт по лабораторной работе №3.Система контроля версий Git

Королёв Иван Андреевич

Содержание

# 1 Цель работы

Целью работы является изучить идеологию и применение средств контроля версий. Приобрести практические навыки по работе с системой git. # Задание Необходимо зарегестрироваться на GitHub. Сделать базовые настройки,создать SSH-ключ, создать рабочее пространоство и рупозиторий 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 заполняюю основные данные(**¿fig:1?**)

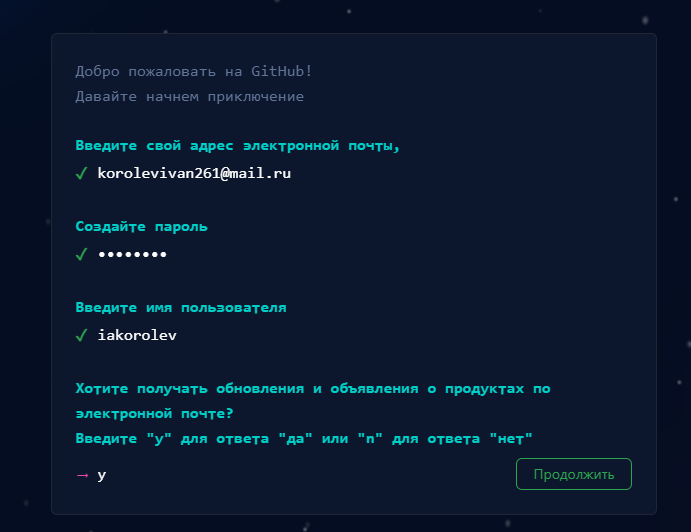


Рис. 1: Создание учетной записи на GitHub

## 2.2 Базовая настройка git:

Сначала сделаю предварительную конфигурацию git. Открою терминал и введу следующие команды, указав имя и email(**¿fig:2?**)

Рис. 2: Указываем имя и email

Рис. 2: Указываем имя и email

Настрою utf-8 в выводе сообщений git(**¿fig:3?**)

Рис. 3: Настройка utf-8

Рис. 3: Настройка utf-8

Задам имя начальной ветки(**¿fig:4?**)

Рис. 4: Имя начальной ветки

Рис. 4: Имя начальной ветки

Параметр autocrlf(**¿fig:5?**)

Рис. 5: Параметр autocrlf

Рис. 5: Параметр autocrlf

Параметр safecrlf(-fig. **¿fig:6?**])

Рис. 6: Параметр safecrlf

Рис. 6: Параметр safecrlf

## 2.3 Создание SSH-ключа:

Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый)(**¿fig:7?**)

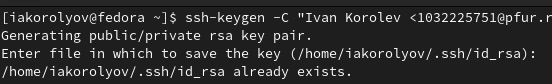


Рис. 7: Создание ssh-ключа

Ключи сохраняться в каталоге ~/.ssh/. Далее необходимо загрузить сгенерённый 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 **¿fig:8?**)



Рис. 8: Нажимаю New SSH Key, загружаю сгенерированный ключ

Скопирую из локальной консоли ключ в буфер обмен с помощью команды(рис.9 **¿fig:9?**)

Рис. 9: Команда cat

Рис. 9: Команда cat

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

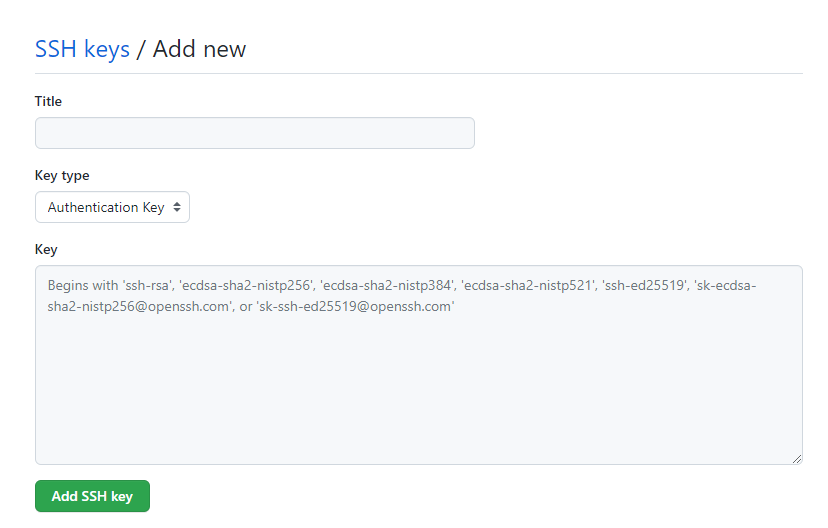


Рис. 10: Создание ssh-ключа

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

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

Рис. 11: Создание каталога

Рис. 11: Создание каталога

## 2.5 Создание репозитория курса на основе шаблона:

Репозиторий на основе шаблона можно создать через web-интерфейс github. Перейду на станицу репозитория с шаблоном курса https://github.com/yam adharma/course-directory-student-template. Далее выберу Use this template(рис.12 **¿fig:12?**)

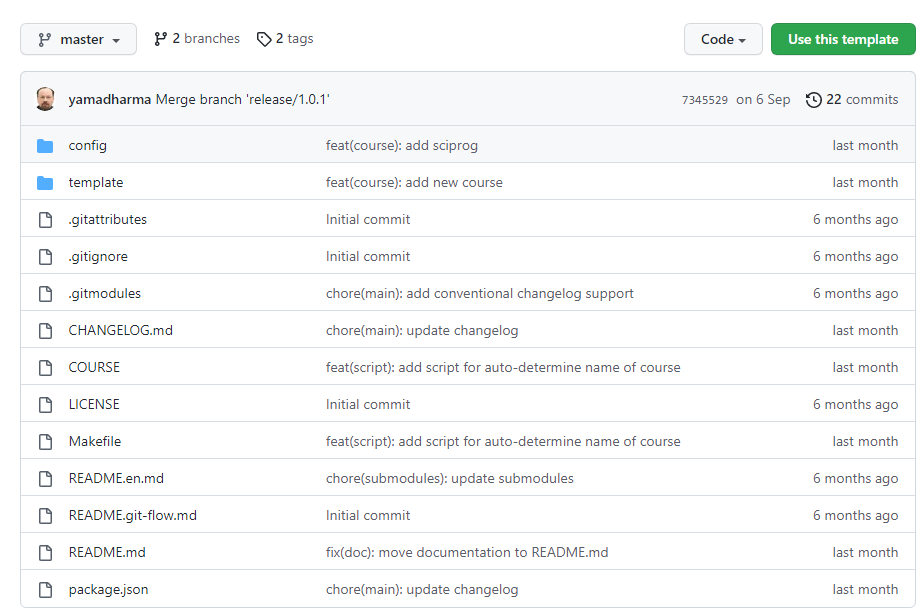


Рис. 12: Создаю репозиторий

В открывшемся окне задаю имя репозитория (Repository name) study\_2022–2023\_arh-pc и создаю репозиторий (кнопка Create repository from template). Открою терминал и перейду в каталог(рис.13 **¿fig:13?**)

Рис. 13: Команда cd

Рис. 13: Команда cd

Клонирую созданный репозиторий(рис.14 **¿fig:14?**)

Рис. 14: Клонирую созданный репозиторий

Рис. 14: Клонирую созданный репозиторий

## 2.6 Настройка каталога курса:

Перейду в каталог курса(рис.15 **¿fig:15?**)

Рис. 15: Команда cd

Рис. 15: Команда cd

Удалю лишние файлы,cоздам необходимые каталоги(рис.16 **¿fig:16?**)

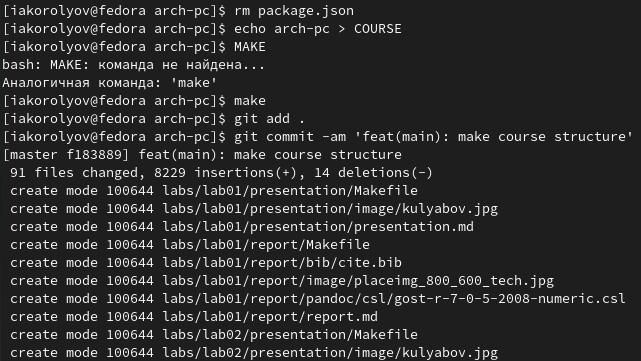


Рис. 16: Удаление лишнего и создание необходимых файлов

Отправлю файлы на сервер(рис.17 **¿fig:17?**)

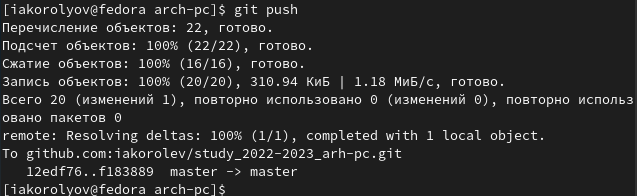


Рис. 17: Команда git push

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

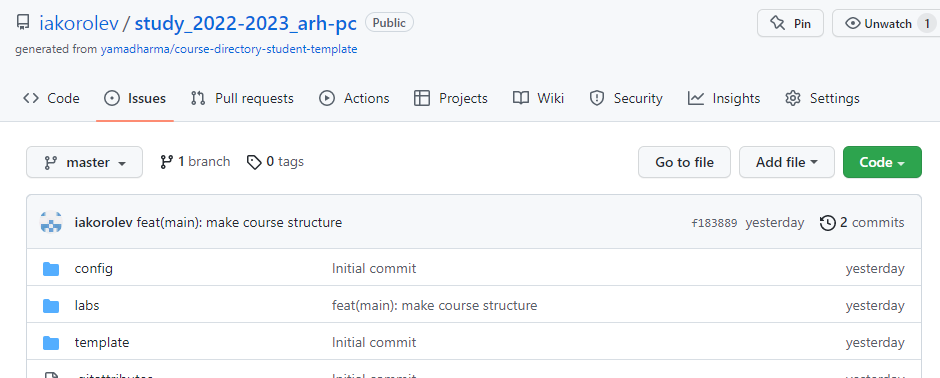


Рис. 18: Создал рабочее пространство

# 3 Выполнение самостоятельной работы:

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

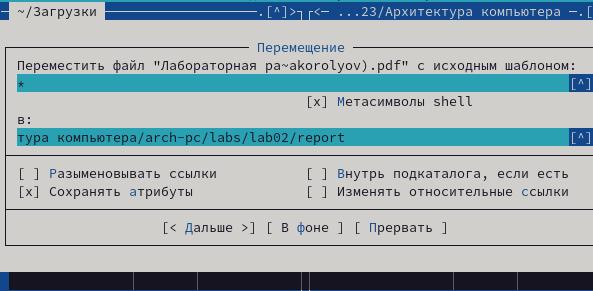


Рис. 19: Перемещаю загруженную лабораторную работу

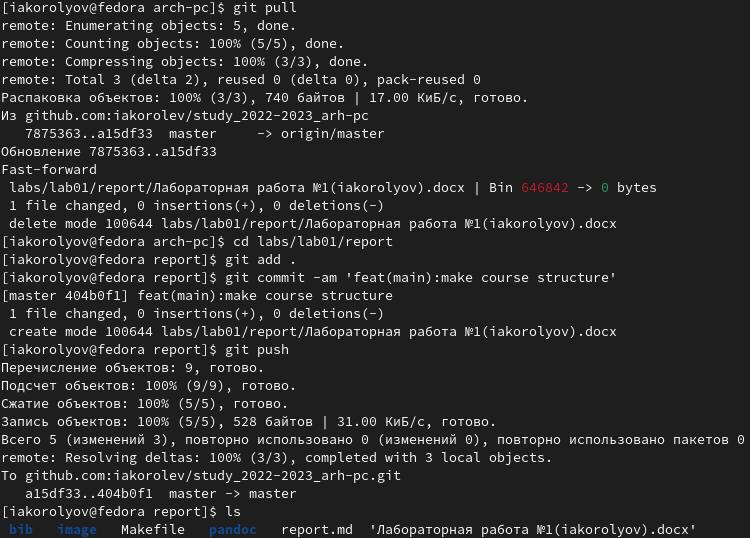


Рис. 20: С помощью команд git pull, add, commit, push, загружаю все лабораторные в репозиторий

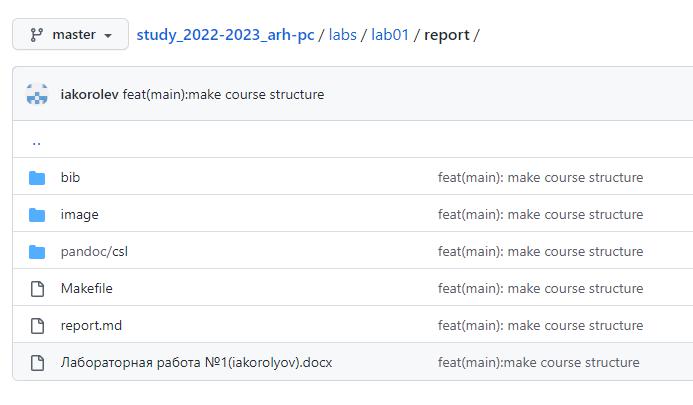


Рис. 21: Лабораторная работа

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

# 4 Выводы

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