Отчёт по лабораторной работе №2

Архитектура вычислительных систем

Сабралиева Марворид Нуралиевна

Содержание

# 1 Цель работы

Изучить идеологию и применение средств контроля версий. Приобрести практические навыки по работе с системой git.

# 2 Задание

1. Создайте отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства (labs>lab03>report).
2. Скопируйте отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданного рабочего пространства.
3. Загрузите файлы на github. # Теоретическое введение
4. Cистемы контроля версий. Общие понятия Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево про- екта 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 зависbмости 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.
5. Основные команды git Наиболее часто используемые команды git представлены в таблице 2.1. Таблица 2.1. Основные команды git Команда Описание git init создание основного дерева репозитория git pull получение обновлений (изменений) текущего дерева из центрального репозитория git push отправка всех произведённых изменений локального дерева в центральный репозиторий git status просмотр списка изменённых файлов в текущей директории git diff просмотр текущих изменения git add . добавить все изменённые и/или созданные файлы и/или каталоги git add файлы и/или каталоги git rm имена\_файлов удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории) git commit -am ‘Описание коммита’ сохранить все добавленные изменения и все изменённые файлы git checkout -b имя\_ветки создание новой ветки, базирующейся на текущей git checkout имя\_ветки переключение на некоторую ветку (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой) git push origin имя\_ветки отправка изменений конкретной ветки в wентральный репозиторий git merge –no-ff имя\_ветки слияние ветки с текущим деревом git branch -d имя\_ветки удаление локальной уже слитой с основным деревом ветки git branch -D имя\_ветки принудительное удаление локальной ветки git push origin :имя\_ветки удаление ветки с центрального репозитория

# 3 Выполнение лабораторной работы

1. Создадим учётную запись на сайте https://github.com/ и заполним основные данные как видно на рисунке 1

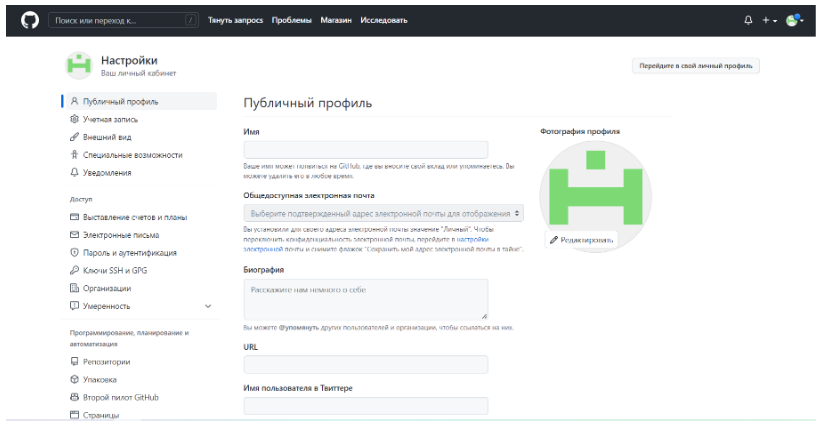


Рис. 1: создание учётной записи

1. Сначала сделаем предварительную конфигурацию «git». Для этого нужно открыть терминал и ввести команды «git config –global», указав имя и email владельца репозитория:рис 2

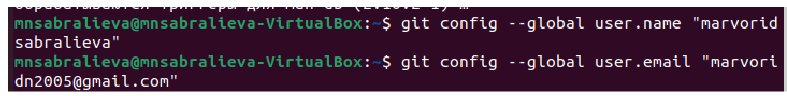


Рис. 2: предваритетельная конфигурация

Настроим utf-8 в выводе сообщений “git”:3

Рис. 3: настройка utf-8

Рис. 3: настройка utf-8

Зададим имя начальной ветки (будем называть её master):4

Рис. 4: название

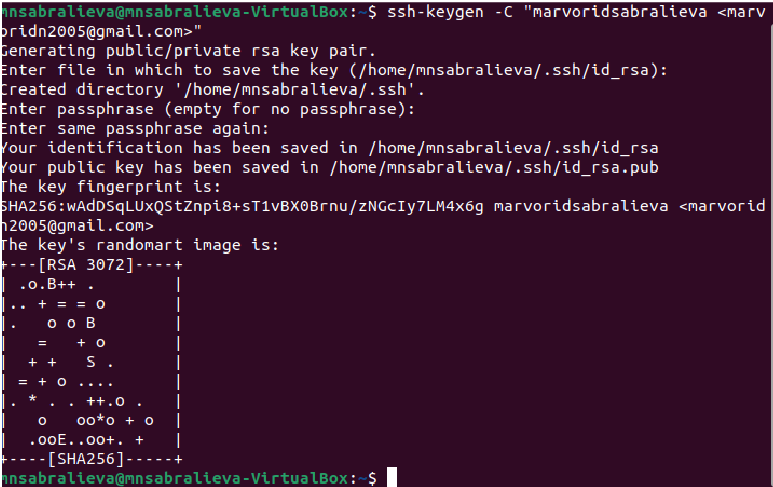
Рис. 4: название

Введём параметр “autocrlf” и параметр “safecrlf”:5

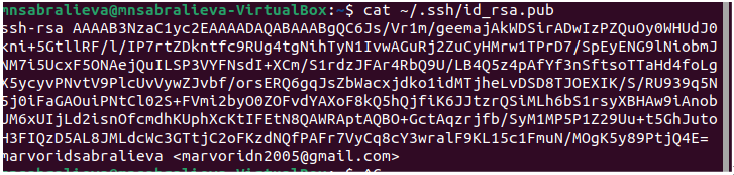
Рис. 5: введение параметров

Рис. 5: введение параметров

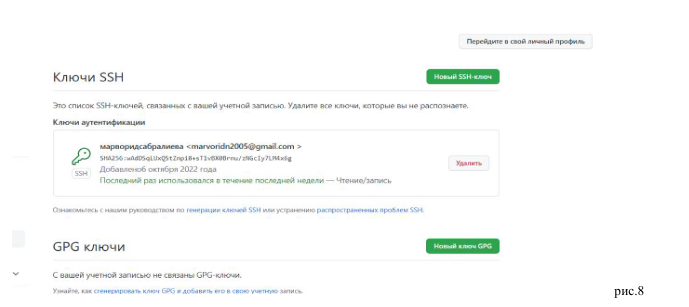
1. Для последующей идентификации пользователя на сервере репозиториев сгенерируем пару ключей (приватный и открытый):**¿fig:006?**



Далее необходимо загрузить сгенерённый открытый ключ на сайт http://github.org Для этого копируем из локальной консоли ключ в буфер обмена:**¿fig:007?**



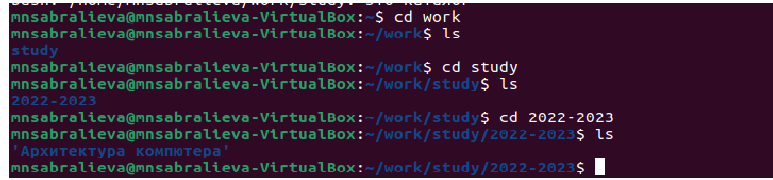
Вставляем ключ в появившееся на сайте поле и указываем для ключа имя (Title).**¿fig:008?**



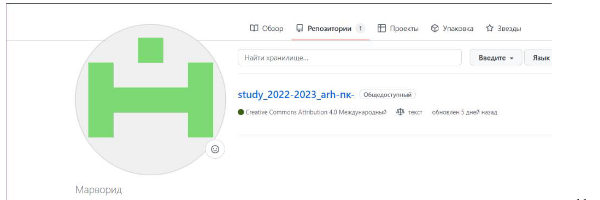
1. Откройте терминал и создайте каталог для предмета «Архитектура компьютера»:**¿fig:009?**



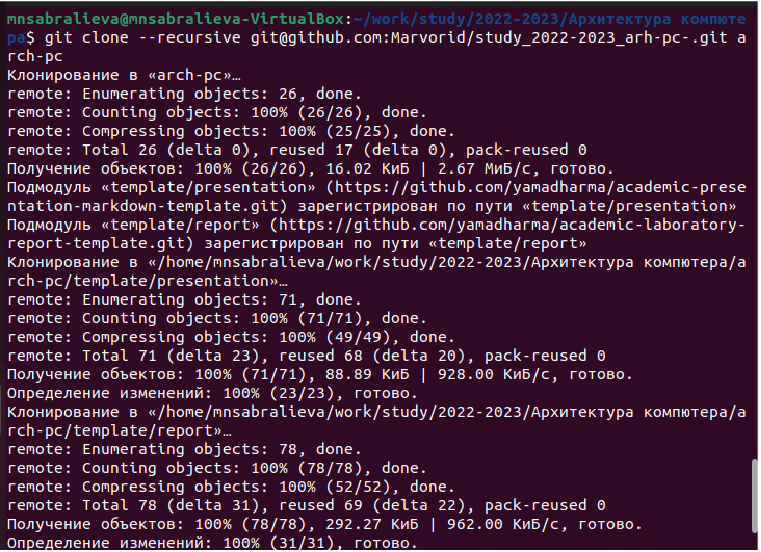
Проверим созданные каталоги:**¿fig:010?**



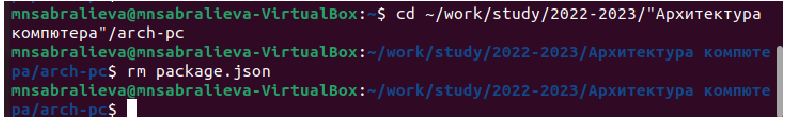
1. Создадим репозиторий на основе шаблона через web-интерфейс github. Для этого перейдём на станицу репозитория с шаблоном курса https://github.com/yam adharma/course-directory-student-template. Далее выберем «Use this template». В открывшемся окне зададим имя репозитория (Repository name) study\_2022–2023\_arh-pc и создадим репозиторий:**¿fig:011?**



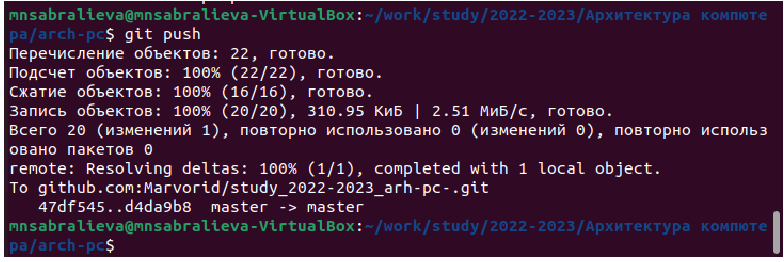
Откроем терминал и перейдём в каталог курса: « work/study/2022- 2023/“Архитектура компьютера”» и клонируем созданный репозиторий с помощью команды “git clone”:**¿fig:012?**

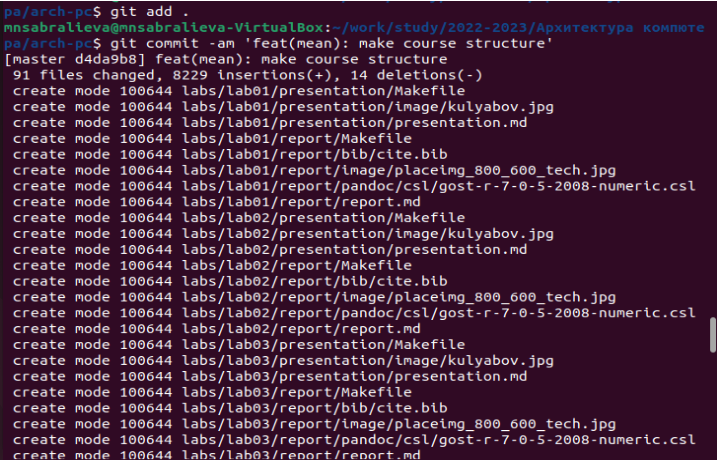


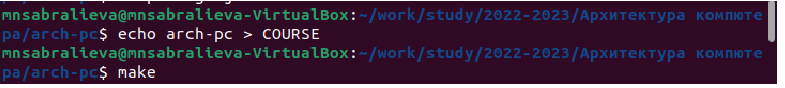
1. Перейдём в каталог курса и удалим лишние файлы с помощью команды “ rm” **¿fig:013?**

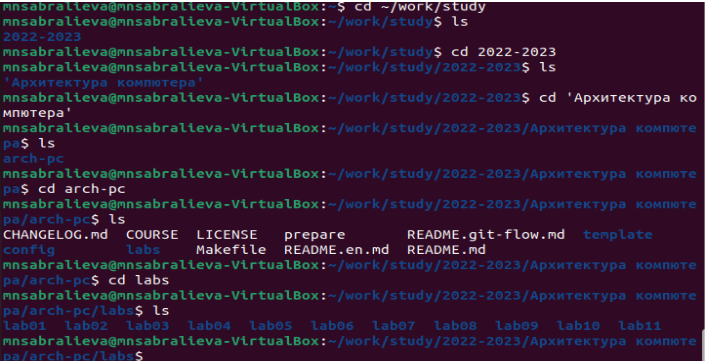


Создадим необходимые каталоги(рис.14) и отправим файлы на сервер (рис.15 и рис.16)**¿fig:14?** **¿fig:015?** **¿fig:016?**

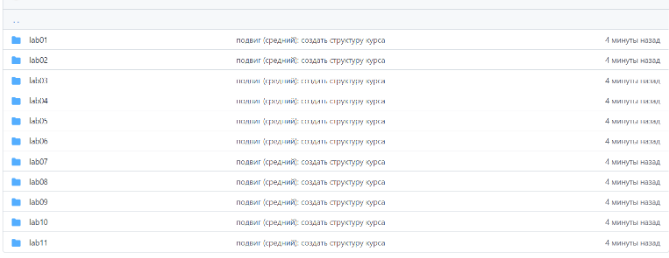


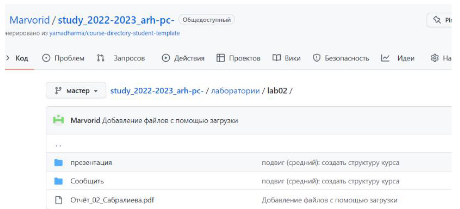


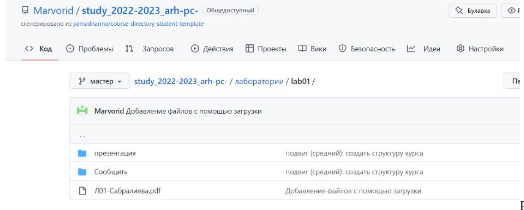
 Проверим правильность создания иерархии рабочего пространства в локальном репозитории (рис.17) и на странице github ( рис.18):**¿fig:017?**



**¿fig:018?**

 5. Скопируем отчеты по выполнению предыдущей лабораторной работы(рис.19) и текущей(рис.20) в соответствующие каталоги созданного рабочего пространства и загрузим их на сайт github:**¿fig:019?** **¿fig:020?**





# 4 Выводы

в ходе выполнения данной лабораторной работы я изучила идеологию и применение средств контроля версий, и также приобрела практические навыки по работе с системой git.