РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

**Отчет по лабораторной работе №2**

*дисциплина: Архитектура ЭВМ*

Студент: Агзамов Артур Дамирович(1032253528)

Группа: НКАбд-01-25

**Содержание**

**1 Цель работы стр.4**

**2 Теоретическое введение стр.5**

**3 Выполнение лабораторной работы стр.9**

**4 Выполнение самостоятельной работы стр.16**

**5 Выводы стр.17**

**Список иллюстраций**

1. *3.2 рисунок 1 стр.9*
2. *3.2 рисунок 2 стр.9*
3. *3.2 рисунок 3 стр.9*
4. *3.2 рисунок 4 стр.9*
5. *3.3 рисунок 5 стр.10*
6. *3.3 рисунок 6 стр.10*
7. *3.4 рисунок 7 стр.11*
8. *3.5 рисунок 8 стр.11*
9. *3.5 рисунок 9 стр.11*
10. *3.5 рисунок 10 стр.12*
11. *3.5 рисунок 11 стр.12*
12. *3.6 рисунок 12 стр.15*
13. *3.2 рисунок 1*

**1.Цель работы**

Изучение идеологии и применения средств контроля версий, приобретение практических навыков по работе с системой контроля версий git.

**2. Теоретическое введение**

**2.1.** **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.

**2.2.** **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.

**2.3. Основные команды git**

|  |  |
| --- | --- |
| Команда | Описание |
| git init  git pull  git push  git status  git diff  git add .  git add  имена\_файлов  git rm  имена\_файлов  git commit -am  ‘Описание коммита’  git checkout -b  имя\_ветки  git checkout  имя\_ветки  git push origin  имя\_ветки  git merge  --no-ff имя\_ветки  git branch -d  имя\_ветки  git branch -D  имя\_ветки  git push origin  :имя\_ветки | создание основного дерева репозитория  получение обновлений текущего дерева из центрального репозитория  отправка всех произведенных изменений локального дерева в центральный репозиторий  просмотр списка измененных файлов в текущей директории  просмотр текущих изменений  добавить все измененные и/или созданные файлы и/или каталоги  добавить конкретные измененные и/или созданные файлы и/или каталоги  удалить файл и/или каталог из индекса репозитория(при этом файл и/или каталог остается в локальной директории)  сохранить все добавленные изменения и все измененные файлы  создание новой ветки, базирующейся на текущей  переключение на некоторую ветку(при переключении на ветку, которой еще нет в локальном репозитории, она будет создана и связана с удаленной)  отправка изменений конкретной ветки в центральный репозиторий  слияние ветки с текущим деревом  удаление локальной уже слитой с основным деревом ветки  принудительное удаление локальной ветки  удаление ветки с центрального репозитория |

**2.4. Стандартные процедуры работы при наличии центрального репозитория**

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): git checkout master git pull git checkout -b имя\_ветки 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. Для этого необходимо проверить, какие файлы изменились к текущему моменту: git status

При необходимости удаляем лишние файлы, которые не хотим отправлять в центральный репозиторий. Затем полезно просмотреть текст изменений на предмет соответствия правилам ведения чистых коммитов: git diff

Если какие-либо файлы не должны попасть в коммит, то помечаем только те файлы, изменения которых нужно сохранить. Для этого используем команды добавления и/или удаления с нужными опциями:

git add имена\_файлов

git rm имена\_файлов

Если нужно сохранить все изменения в текущем каталоге, то используем: git add . Затем сохраняем изменения, поясняя, что было сделано: git commit -am "Some commit message" и отправляем в центральный репозиторий: git push origin имя\_ветки или git push

**3. Выполнение лабораторной работы**

**3.1. Настройка github**

Создаем учётную запись на сайте https://github.com/ и заполняем основные данные.

**3.2. Базовая настройка git**

Указываем имя и e-mail владельца репозитория.

****

Рисунок

Настраиваем utf-8 в выводе сообщений git.

Рисунок

Задаем имя начальной ветки(master).



Рисунок

Задаем параметры autocrlf и safecrlf



Рисунок

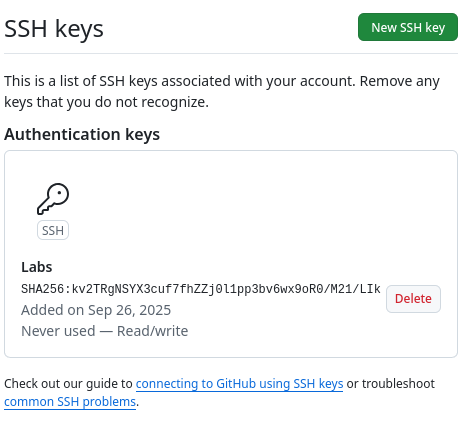
**3.3. Создание SSH-ключа**

Для последующей идентификации на сервере репозиториев генерируем приватный и открытый ключ. Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок

Открытый ключ загружаем на github.



Рисунок

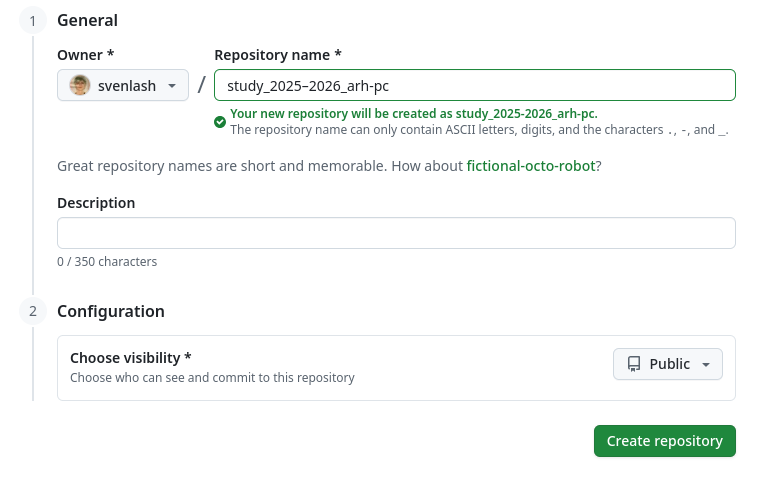
**3.4. Создание рабочего пространства и репозитория курса на основе шаблона**

Открываем терминал и создаем каталог для предмета “Архитектура компьютера”****

Рисунок

**3.5. Создание репозитория курса на основе шаблона**

Переходим на станицу репозитория с шаблоном курса https://github.com/yamadharma/cour se-directory-student-template.

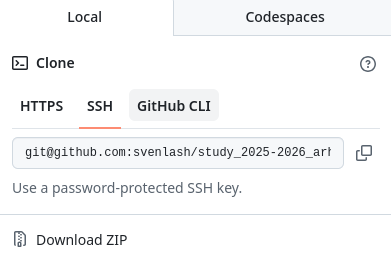
Далее выбираем «Use this template». В открывшемся окне задаем имя репозитория (Repository name) study\_2025–2026\_arh-pc и создайте репозиторий (кнопка Create repository from template). Открываем терминал и переходим в каталог курса: 

Рисунок



Рисунок

Клонируем созданный репозиторий по ссылке для клонирования.



Рисунок

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок

**3.6. Настройка каталога курса**

Переходим в каталог курса, создаем необходимые каталоги и отправляем файлы на сервер.

Изображение выглядит как текст, снимок экрана, меню

Содержимое, созданное искусственным интеллектом, может быть неверным.Изображение выглядит как текст, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

**Изображение выглядит как текст, снимок экрана, меню

Содержимое, созданное искусственным интеллектом, может быть неверным.**Изображение выглядит как текст, снимок экрана, меню

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, меню

Содержимое, созданное искусственным интеллектом, может быть неверным.

**Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.**

Рисунок

Проверяем правильность создания иерархии рабочего пространства в локальном репозитории и на странице github.

**4. Самостоятельная работа**

Создаем отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства (labs/lab02/report).

Копируем отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданного рабочего пространства.

Загружаем файлы на github.

**5.Выводы**

Изучили идеологию git. Научились управлять репозиториями через сайт github и терминал.

Все поставленные задачи были выполнены.