**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2**

*дисциплина: Архитектура компьютера и операционные системы*

Студент: Симонова В.И. Группа: НКАбд-05-2023

**МОСКВА** 2023 г.

**Содержание** 1 Цель работы

2 Задание

3 Теоретическое введение

4 Выполнение лабораторной работы

Порядок выполнения лабораторной работы:

1. Настройка Github
   1. Базовая настройка git
   2. Создание SSH ключа
   3. Сознание рабочего пространства и репозитория курса на основе шаблона
   4. Сознание репозитория курса на основе шаблона
   5. Настройка каталога курса

Задание для самостоятельной работы

**Цель работы**

Целью работы является изучить идеологию и применение средств контроля версий. Приобрести практические навыки по работе с системой git.

**Задание**

1.Настройка GitHub.

2.Базовая настройка Git.

3.Создание SSH-ключа.

4.Создание рабочего пространства и репозитория на основе шаблона курса. 5.Создание репозитория курса на основе шаблона.

6.Настройка каталога курса.

7.Выполнение заданий для самостоятельной работы.

**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 (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. Демидова А. В. 14 Архитектура ЭВМ 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.

**4 Выао 4.1.Настройка Github**

Захожу на сайт https://github.com/ и создаю учётную запись (Рисунок 1), заполняя основные данные.

*Рисунок 1. Создание профиля на Github*

1. Базовая настройка Github

Запускаю виртуальную машину и в терминале ввожу команды,указав имя влядельца репозитория и email. (Рисунок 2)

git config –global user.name “”

git config –global user.email “<vikas2008.va@gmail.com”

*Рисунок 2..Предварительная конфигурация Git*

Настраиваю utf-8 в выводе сообщений git, используя команду git config.( Рисунок 3)

*Рисунок 3.Стандарт кодировки* Задаю имя «master» для начальной ветки.( Рисунок 4.)

*Рисунок 4.Имя начальной ветки* Задаю параметр autocrlf со значением input.( Рисунок 5.)

*Рисунок 5. Параметр autocrlf*

Задаю параметр safecrlf со значением warn, для проверки преобразований на обратимость (Рисунок 6)

*Рисунок 6.Параметр safecrlf*

1. Создание SSH ключа

Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый). Для этого необходимо использовать команду ssh-keygen -C “Имя Фамилия”. Ключи сохраняться в каталоге ~/.ssh/.( Рисунок 7.)

*Рисунок 7. Генерация SHH ключа*

Чтобы скопировать текст через терминал мне необходимо воспользоваться

утилитой xclip. Для начала, в Linux Fedora ёё нужно установить. Устанавливаю утилиту xclip c помощью sudo dnf.

*Рисунок 8.Установка утилиты xclip(часть1)* Получаю сообщение о том, что функция установлена.

*Рисунок 9. Установка утилиты xclip(часть2)* Копирую открытый ключ из директории ~/.ssh.( Рисунок 10.)

*Рисунок 10.Копирование колюча*

Загружаю сгенерированный ключ http: //github.org/, войдя под своей учётной записью. Перехожу в «setting», выбираю в боковом «SHH and GPG keys» , затем

* New SSH key». Вставляю ключ в появившееся на сайте поле и указываю для ключа имя (Title).( Рисунок 11.)

*Рисунок 11.Добавление ключа*

1. Создание рабочего пространства и репозитория курса на основе шаблона.

Открываю терминал, с помощью команды mkdir создаю последовательность директорий ~/work/study/2023-2024/“Архитектура компьютера”. Командой ls проверяю создание каталога.( Рисунок 12.)

*Рисунок 12.Создание рабочего пространства*

1. Сознание репозитория курса на основе шаблона.

По ссылке https://github.com/yamadharma/cour se-directory-student-template перехожу на страницу с шаблоном, выбираю «Use template».

*Рисунок 13.Шаблон для репозитория*

В открывшемся окне задаю имя репозитория (Repository name) study\_2023– 2024\_arhpc и создаю репозиторий (кнопка Create repository from template).( Рисунок 14.)

*Рисунок 14.Создание репозитория* Проверяю создание репозитория (Рисунок 15.)

*Рисунок 15..Созданный репозиторий*

Используя утилиту cd перехожу в созданный каталог курса.( Рисунок 16.)

*Рисунок 16.переход к каталогу курса*

Клонирую созданный репозиторий, используя команду «git clone –recursive git@github.com:/study\_2023–2024\_arh-pc.git».( Рисунок 17)

*Рисунок 17.Клонирование репозитория* Копирую ссылку для клонирования. (Рисунок 18.)

*Рисунок 18.Копирую ссылку*

1. Настройка каталога курса.

Перехожу в каталог курса , используя команду cd.( Рисунок 19.)

*Рисунок 19.премещение по директориям* Удаляю файл «package.json», используя команду «rm».( Рисунок 20)

*Рисунок 20.Удаление лишних файлов* Создание необходимых каталогов.( Рисунок 21.)

*Рисунок 21.Создание каталогов*

*Отправляю файлы на сервер, используя команды «git add», «git commit -am ‘feat(main): make course structure’», « git push».( Рисунок 22.)*

*Рисунок 22.*

Команда «git push».( Рисунок 23.)

*Рисунок 23.Отправление данных на сервер* Открываю GitHub и проверяю , что все команды сработали правильно.

*Рисунок 24.Проверка выполнения команд* Задание для самостоятельной работы.

Перемещаюсь в директорию …/labs/lab02/report.( Рисунок 25.)

*Рисунок 25. Перемещение по директориям*

Создаю файл для отчёта по лабораторной работе 2, используя утилиту

* touch».( Рисунок 26)

*Рисунок 26.Создание файла*

Проверяю наличие недавно созданного файла, в нужной мне директории.( Рисунок 27.)

*Рисунок 27. Создание файла* Перемещаюсь в директорию …/labs/lab01/report.( Рисунок 28)

*Рисунок 28. Пермещение по директории.*

Проверяю наличие файла, содержащего первую лабораторную работу и добавляю его на сервер, используя «git add».( Рисунок 29.)

*Рисунок 29. Проверка и добавление файла*

Перемещаюсь по директории и добавляю созданный файл с помощью «git add».( Рисунок 30.)

*Рисунок 30. Добавление файла*

Комментирую и сохраняю изменения на сервере как добавление файлов, используя «git commit».( Рисунок 31)

*Рисунок 31. Добавление файла* Отправляю файлы в центральный репозиторий.( Рисунок 32.)

*Рисунок 32. Отправление файлов.*

*Захожу на GitHub и проверяю, что оба файла находятся в нужных мне папках.( Рисунок 33) (Рисунок 34).*

*Рисунок 33. Наличие первой лабораторной работы*

*Рисунок 34. Наличие второй лабораторной работы*

Вывод

В ходе выполнения лабораторной работы я приобрела практические навыки работы с системой Git, создала репозиторий в GitHub с помощью консоли в ОС Linux. Научилась управлять перемещением файлов в репозиторий.

Список литературы:

1. «The Linux Command Line» Уильям Шоттс
2. UNIX and Linux System Administration Handbook

Авторы: Эви Немет, Гарт Снайдер, Трент Хейн, Бен Уэйли, Дэн Макни. 18