Отчёт по лабораторной работе №2

Дисциплина: Архитектура компьютеров

Карпова Анастасия Александровна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Целью данной работы является изучение идеологии и применение средств контроля версий. Приобретение практических навыков по работе с системой git.

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. Задание 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 (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. 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. 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.

4 Выполнение лабораторной работы

Настройка GitHub

Создаю учётную запись на GitHub (рис. 4.1).

Создание учётной записи на GitHub.

Рис. 4.1: Создание учётной записи на GitHub.

Базовая настройка git Открываю терминал для дальнейшей работы. Выполняю предварительную конфигурацию git: открыв терминал ввожу команды git config –global user.name “” и git config –global user.email “work@mail”, указывая имя и email владельца репозитория

Выполнение предварительной конфигурации git

Рис. 4.2: Выполнение предварительной конфигурации git

Настраиваю utf-8 в выводе сообщений git. (рис. 4.3)

Настройка кодировки

Рис. 4.3: Настройка кодировки

Задаю имя начальной ветки(будем называть ее master). (рис. 4.4)

Создание имени начальной ветки

Рис. 4.4: Создание имени начальной ветки

Задаю параметр autocrlf и параметр salecrlf со значениями input и warn соответственно. (рис. 4.5)

Параметры autocrlf и safecrlf

Рис. 4.5: Параметры autocrlf и safecrlf

Создание SSH ключа

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

Генерация SSH-ключа

Рис. 4.6: Генерация SSH-ключа

Далее необходимо загрузить сгенерённый открытый ключ. Для этого зайти на сайт http: //github.org/ под своей учётной записью и перейти в меню Setting. После этого выбрать в боковом меню SSH and GPG keys и нажать кнопку New SSH key. Скопировав из локальной консоли ключ в буфер обмена при помощи команды cat ~/.ssh/id_rsa.pub | xclip -sel clip вставляю ключ в появившееся на сайте поле и ауказываем для ключа имя. Обратим внимание на команду, где xclip – утилита для копирования любого текста через терминал. Но изначально она не установлена в дистрибутиве, поэтому для дальнейшей работы устанавливаем при помощи команды apt-get install с ключом -y, при этом в начале команды использую sudo (для установки от имени админа. (рис. 4.7)

Установка утилиты xclip.

Рис. 4.7: Установка утилиты xclip.

После установки утилиты xclip мы можем возпользоваься командой cat ~/.ssh/id_rsa.pub | xclip -sel clip (рис. 4.8)

Копирование содержимого файла.

Рис. 4.8: Копирование содержимого файла.

Теперь захожу в свой профиль и выбираю категорию SSH and GPG keys. (рис. 4.9)

Выбор SSH and GPG keys.

Рис. 4.9: Выбор SSH and GPG keys.

Далее нажимаю на кнопку new SSH key и создаю новый ключ. (рис. 4.10)

Создание нового ключа.

Рис. 4.10: Создание нового ключа.

Вставляю скопированный ключ в поле “key”. В поле Title указываю имя ключа. Нажимаю Add SSH-key, чтобы добавить ключ. (рис. 4.11)

Создание SSH-ключа.

Рис. 4.11: Создание SSH-ключа.

Сознание рабочего пространства и репозитория курса на основе шаблона

Создаю директорию и рабочее пространство с помощью утилиты mkdir, благодаря ключу -p создаю все директории после домашней ~/work/study/2023-2024/”Архитектура компьютера” рекурсивно. Для проверки использую ls. (рис. 4.12)

Создание рабочего пространства.

Рис. 4.12: Создание рабочего пространства.

Создание репозитория курса на основе шаблона

Перехожу на страницу репозитория с шаблоном курса введя адрес https://github.com/yamadharma/cour se-directory-student-template в браузерной строке. Далее выбираю Use this template. (рис. 4.13)

Выбор шаблона.

Рис. 4.13: Выбор шаблона.

В открывшемся окне задаю имя репозитория (Repository name) study_2023–2024_arhpc и создайте репозиторий (кнопка Create repository from template). (рис. 4.14)

Создание репозитория.

Рис. 4.14: Создание репозитория.

Открываю терминал и перехожу в каталог курса. (рис. 4.15)

Перемещение в каталог курса.

Рис. 4.15: Перемещение в каталог курса.

Клонирую созданный репозиторий (рис. 4.17), предварительно копируя ссылку для клонирования, которую можно найти на странице созданного репозитория. (рис. **??**)

Копирование ссылки для клонирования.

Рис. 4.16: Копирование ссылки для клонирования.

Клонирование репозитория..

Рис. 4.17: Клонирование репозитория..

Настройка каталога курса

Перехожу в каталог курса при помощи cd и удаляю лишние файлы при помощи команды rm package.json. (рис. 4.18)

Удаление лишних файлов.

Рис. 4.18: Удаление лишних файлов.

Создаю необходимые каталоги(рис. 4.19). И отправляю файлы на сервер(рис. 4.20).

Создание необходимых каталогов.

Рис. 4.19: Создание необходимых каталогов.

Отправка файлов на сервер.

Рис. 4.20: Отправка файлов на сервер.

Так как я забыла сделать скриншот результата команды git commit в терминале, я прикрепляю итог использования данной команды. (рис. 4.21)

Проверка.

Рис. 4.21: Проверка.

Выполнение заданий для самостоятельной работы.

1. Перехожу в директорию labs/lab02/report при помощи cd и создаю в каталоге файл для отчета по второй лабораторной работе. (рис. [**fig022?**])

Создание файла.

Рис. 4.22: Создание файла.

2. Перехожу из подкаталога lab02/report в подкаталог lab01/report. (рис. 4.23)

Перемещение между каталогами.

Рис. 4.23: Перемещение между каталогами.

Проверяю местонахождение файлов с отчетами по первой и второй лабораторным работам, используя команду ls. (рис. 4.24)

Проверка.

Рис. 4.24: Проверка.

Копирую первую лабораторную работу при помощи cp и проверяю правильность выполнения, используя ls. (рис. 4.25)

Копирование файла

Рис. 4.25: Копирование файла

Перехожу из подкаталог lab02/report в подкаталог lab01/report. И копирую вторую работу аналогично первой. (рис. 4.26)

Копирование файла.

Рис. 4.26: Копирование файла.

Добавляю файл Л01_Карпова отчет.pdf и файл Л02_Карпова отчет, используя git add. (рис. 4.27) Сохраняю изменения на сервере при помощи команды git commit -m. (рис. 4.28)

Добавление файлов на сервер.

Рис. 4.27: Добавление файлов на сервер.

Сохранение изменений.

Рис. 4.28: Сохранение изменений.

Отправляю в центральный репозиторий сохраненные изменения. (рис. 4.29)

Отправка сохраненных изменений.

Рис. 4.29: Отправка сохраненных изменений.

Проверяю выполненные команды. Для этого захожу на github. (рис. 4) (рис. 4) (рис. 4)

Проверка. Проверка. Проверка.

5 Выводы

В ходе лабораторной работы я изучила идеологию и приминение средств контроля версий и приобрела практические навыки по работе с системой git.

Список литературы

1. Архитектура ЭВМ
2. Инструкция по использованию git ::: {#refs} :::