Отчёт по лабораторной работе №2

Дисциплина: архитектура компьютера.

Наговицын Арсений Владимирович

Содержание

# 1 Цель работы

Целью данной работы является изучить идеологию и применение средств контроля версий, а также приобрести практические навыки 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. Выполнение заданий 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 изменения, сделанные разными участниками, 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. 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. 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. 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). 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. # Выполнение лабораторной работы

## 3.1 Настройка GitHub:

Так как мой аккаунт на GitHub уже создан перехожу к следующему заданию (рис. 3.1).

fig:
Аккаунт GitHub.

Рис. 3.1: Аккаунт GitHub.

## 3.2 Базовая настройка Git:

Открываю виртуальную машину, затем открываю терминал и делаю предварительную конфигурацию git. Ввожу команду git config –global user.name “”, указывая свое имя и команду git config –global user.email “work@mail”, указывая в ней электронную почту владельца, то есть мою(рис. 3.2).

fig:
Предварительная конфигурация git.

Рис. 3.2: Предварительная конфигурация git.

Настраиваю utf-8 в выводе сообщений git для корректного отображения символов(рис. 3.3).

fig:
Настройка кодировки.

Рис. 3.3: Настройка кодировки.

Задаю имя «master» для начальной ветки(рис. 3.4).

fig:
Настройка кодировки.

Рис. 3.4: Настройка кодировки.

Задаю параметр autocrlf со значением input, так как я работаю в системе Linux, чтобы конвертировать CRLF в LF только при коммитах (рис. 5). CR и LF – это символы, которые можно использовать для обозначения разрыва строки в текстовых файлах.(рис. 3.5)

fig:
Параметр autocrlf.

Рис. 3.5: Параметр autocrlf.

Задаю параметр safecrlf со значением warn, так Git будет проверять преобразование на обратимость (рис. 6). При значении warn Git только выведет предупреждение, но будет принимать необратимые конвертации (рис. 3.6)

fig:
Параметр safecrlf.

Рис. 3.6: Параметр safecrlf.

## 3.3 Создание SSH-ключа:

Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый). Для этого ввожу команду ssh-keygen -C “Имя Фамилия, work@email”, указывая имя владельца и электронную почту владельца. Ключ автоматически сохранится в каталоге ~/.ssh/ (рис. 3.7).

fig:
Генерация SSH-ключа.

Рис. 3.7: Генерация SSH-ключа.

Копирую ключ из открытой директории, в которой он был сохранен(рис. 3.8).

fig:
Копирование содержимого файла.

Рис. 3.8: Копирование содержимого файла.

Открываю браузер, захожу на сайт GitHub. Открываю свой профиль и выбираю страницу «SSH and GPG keys». Нажимаю кнопку «New SSH key» (рис. 3.9).

fig:
Окно SSH and GPG keys.

Рис. 3.9: Окно SSH and GPG keys.

Вставляю скопированный ключ в поле «Key». В поле Title указываю имя для ключа. Нажимаю «Add SSH-key», чтобы завершить добавление ключа. После чего получаю такое сообщение (рис. 3.10).

fig:
Добавление ключа.

Рис. 3.10: Добавление ключа.

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

Открываю терминал. Создаю директорию, рабочее пространство, с помощью утилиты mkdir, блягодаря ключу -p создаю все директории после домашней ~/work/study/2023-2024/“Архитектура компьютера”. Далее проверяю с помощью ls, были ли созданы необходимые мне каталоги (рис. 3.11).

fig:
Создание рабочего пространства.

Рис. 3.11: Создание рабочего пространства.

## 3.5 Создание репозитория курса на основе шаблона:

В браузере перехожу на страницу репозитория с шаблоном курса. Далее выбираю «Use this template», чтобы использовать этот шаблон для своего репозитория (рис. 3.12).

fig:
Страница шаблона для репозитория.

Рис. 3.12: Страница шаблона для репозитория.

В открывшемся окне задаю имя репозитория (Repository name): study\_2023–2024\_arh-pc и создаю репозиторий, нажав на кнопку «Create repository from template» (рис. 3.13).

fig:
Окно создания репозитория.

Рис. 3.13: Окно создания репозитория.

Репозиторий создан(рис. 3.14).

fig:
Созданный репозиторий.

Рис. 3.14: Созданный репозиторий.

Через терминал перехожу в созданный каталог курса (рис. 3.15).

fig:
Премещение между директориями.

Рис. 3.15: Премещение между директориями.

Копирую ссылку для клонирования на странице созданного репозитория, снача ла перейдя в окно «code», далее выбрав в окне вкладку «SSH» (рис. 3.16).

fig:
Окно с ссылкой для копирования репозитория.

Рис. 3.16: Окно с ссылкой для копирования репозитория.

Клонирую созданный репозиторий с помощью команды git clone –recursive git@github.com:AVNagov/study\_2022–2023\_arh-pc.git arch-pc (рис. 3,17).

fig:
Клонирование репозитория.

Рис. 3.17: Клонирование репозитория.

## 3.6 Настройка каталога курса:

Перехожу в каталог arch-pc (рис. 3.18).

fig:
Перемещение между директориями.

Рис. 3.18: Перемещение между директориями.

Удаляю лишние файлы с помощью утилиты rm(рис. 3.19).

fig:
Удаление файлов.

Рис. 3.19: Удаление файлов.

Создаю необходимые каталоги (рис. 3.20).

fig:
Создание каталогов.

Рис. 3.20: Создание каталогов.

Отправляю созданные каталоги с локального репозитория на сервер: добавляю все созданные каталоги с помощью git add, комментирую и сохраняю изменения на сервере как добавление курса с помощью git commit(рис. 3.21).

fig:
Добавление и сохранение изменений на сервере.

Рис. 3.21: Добавление и сохранение изменений на сервере.

Отправляю все на сервер с помощью push (рис. 3.22).

fig:
Выгрузка изменений на сервер.

Рис. 3.22: Выгрузка изменений на сервер.

Проверяю правильность выполнения работы сначала на самом сайте GitHub (рис. 3.23).

fig:
Страница репозитория.

Рис. 3.23: Страница репозитория.

## 3.7 Выполнение заданий для самостоятельной работы:

1. Перехожу в директорию labs/lab02/report с помощью утилиты cd. Создаю в каталоге файл для отчета по второй лабораторной работе (рис. 3.24).

fig:
Создание файла.

Рис. 3.24: Создание файла.

Оформить отчет я смогу в текстовом процессоре LibreOffice Writer, найдя его в открыть с помощью (рис. 3.25).

fig:
Меню приложений.

Рис. 3.25: Меню приложений.

После открытия текстового процессора открываю в нем созданный файл и могу начать в нем работу над отчетом (рис. 3.26).

fig:
Работа с отчетом в текстовом редакторе.

Рис. 3.26: Работа с отчетом в текстовом редакторе.

1. Перехожу из подкаталога lab03/report в подкаталог lab01/report (рис. 3.27).

fig:
Перемещение между директориями.

Рис. 3.27: Перемещение между директориями.

Проверяю местонахождение отчета по первой лабораторной работе (рис. 3.28).

fig:
Проверка местонахождения файла.

Рис. 3.28: Проверка местонахождения файла.

Копирую отчет по лабораторной работе с помощью утилиты cp(рис. 3.29).

fig:
Копирование файла.

Рис. 3.29: Копирование файла.

1. Добавляю с помощью команды git add созданный файл: Л01\_Наговицын\_Отчет (рис. 3.30).

fig:
Добавление файла на сервер.

Рис. 3.30: Добавление файла на сервер.

Сохраняю все добавленные изменения(рис. 3.31).

fig:
Сохранение изменений.

Рис. 3.31: Сохранение изменений.

Отправляю в центральный репозиторий изменения командой git push -f origin master (рис. 3.32).

fig:
Отправка в центральный репозиторий сохраненных изменений.

Рис. 3.32: Отправка в центральный репозиторий сохраненных изменений.

Проверяю на сайте GitHub правильность выполнения заданий. Вижу, что пояснение к совершенным действиям отображается (рис. 3.33).

fig:
Страница каталога в репозитории.

Рис. 3.33: Страница каталога в репозитории.

fig:
Каталог lab01/report.

Рис. 3.34: Каталог lab01/report.

fig:
Каталог lab02/report.

Рис. 3.35: Каталог lab02/report.

# 4 Выводы

При выполнении данной лабораторной работы я изучил идеологию и применение средств контроля версий, а также приобрел практические навыки по работе с системой git.