Отчёта по лабораторной работе №2

Еремина Оксана Андреевна

Содержание

# Цель работы

Целью данной лабораторной работы является освоение практических навыков умений по работе с git

# Задание

1. Создать базовую конфигурацию для работы с git.
2. Создать ключ SSH.
3. Создать ключ PGP.
4. Настроить подписи git.
5. Зарегистрироваться на Github.
6. Создать локальный каталог для выполнения заданий по предмету.

# 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.

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.

# Выполнение лабораторной работы

1. Установка программного обеспечения

В терминале ввожу команду, которая устанавливает git и gh(рис.1)

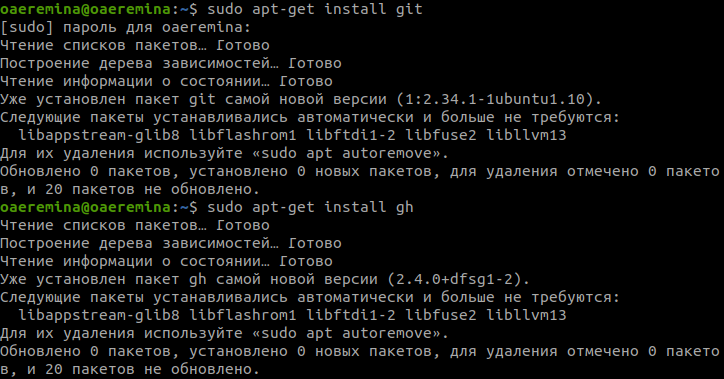


Figure 1: Установка git и gh

1. Базовая настройка git

В терминале задаю имя и email моего репозитория. Далее настраиваю его вводя команды. (рис.2)

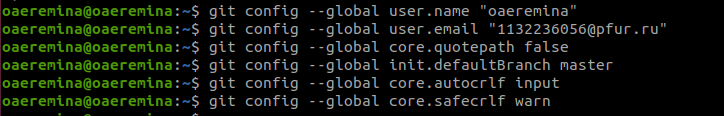


Figure 2: Настройка git

1. Создание ключа ssh

Генерирую ключ и выбираю нужные опции (рис. 3)

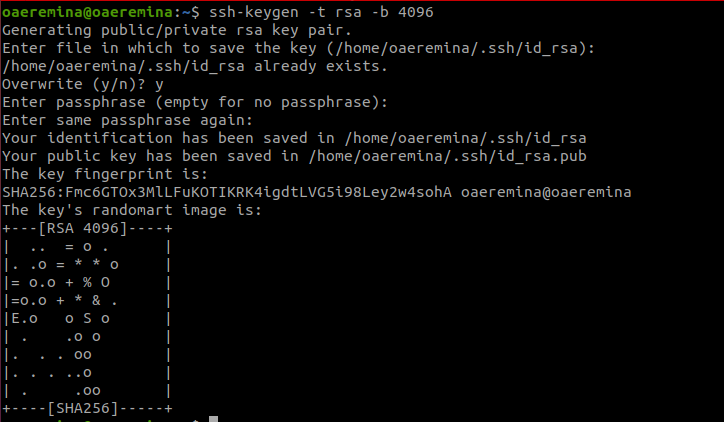


Figure 3: Создание ключа ssh

1. Создание ключа pgp

Вывожу список ключей и копирую сгенерированый ключ в буфер обмена (рис.4)

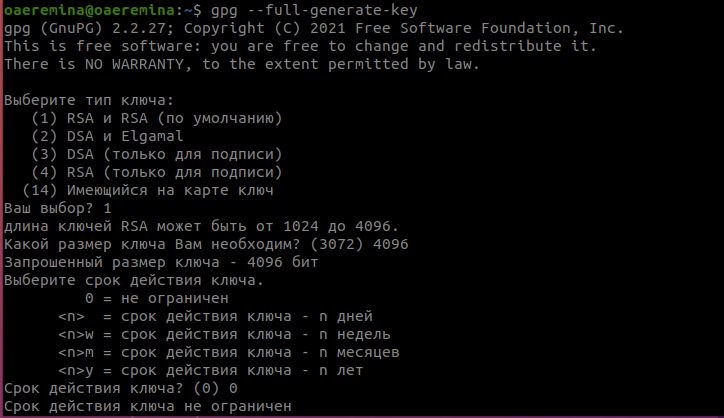


Figure 4: Создание ключа pgp

Перехожу в свой репозиторий, переходу в New GPG key и вставляю код из буфера обмена (рис.5)



Figure 5: Добавление ключа pgp

1. Настройка автоматических подписей коммитов git

Ввожу команды, чтобы git применял мой email при подписи коммитов (рис.6)

Figure 6: Настройка подписей коммитов

Figure 6: Настройка подписей коммитов

1. Настройка gh

Авторизуюсь, вводя команду, далее отвечаю на вопросы (рис.7) После ответов на вопросы меня переносит в браузер и я ввожу код, который написан в терминале.

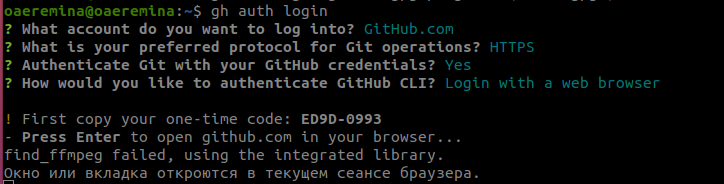


Figure 7: Настройка gh

1. Сознание репозитория курса на основе шаблона

Создаю шаблон рабочего пространства, воодя нужные команды (рис.8)

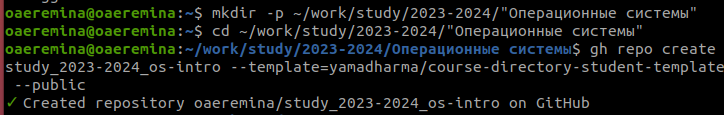


Figure 8: Создание шаблона курса

Перехожу в каталог курса, удаляю лишние файлы, затем создаю необходимые каталоги (рис.9) И отправляю эти файлы на сервер

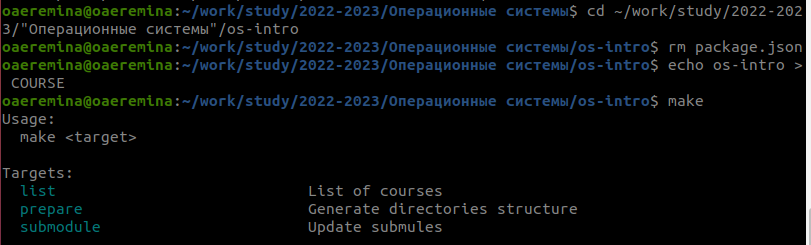


Figure 9: Настройка каталога курса

# Выводы

При выполнение данной лабораторной работы я приобрела практические навыки работы с git

# Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

Системы контроля версий (VCS) - программное обеспечение для облегчения работы с изменяющейся информацией. Они позволяют хранить несколько версий изменяющейся информации, одного и того же документа, может предоставить доступ к более ранним версиям документа. Используется для работы нескольких человек над проектом, позволяет посмотреть, кто и когда внес какое-либо изменение и т. д. VCS ррименяются для: Хранения понлой истории изменений, сохранения причин всех изменений, поиска причин изменений и совершивших изменение, совместной работы над проектами.

2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.  
  
Хранилище – репозиторий, хранилище версий, в нем хранятся все документы, включая историю их изменения и прочей служебной информацией. commit – отслеживание изменений, сохраняет разницу в изменениях. История – хранит все изменения в проекте и позволяет при необходимости вернуться/обратиться к нужным данным. Рабочая копия – копия проекта, основанная на версии из хранилища, чаще всего последней версии.  
  
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.  
  
Централизованные VCS (например: CVS, TFS, AccuRev) – одно основное хранилище всего проекта. Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет, затем добавляет изменения обратно в хранилище. Децентрализованные VCS (например: Git, Bazaar) – у каждого пользователя свой вариант репозитория (возможно несколько вариантов), есть возможность добавлять и забирать изменения из любого репозитория. В отличие от классических, в распределенных (децентралиованных) системах контроля версий центральный репозиторий не является обязательным.  
  
4.Опишите действия с VCS при единоличной работе с хранилищем.  
   
Сначала создается и подключается удаленный репозиторий, затем по мере изменения проекта эти изменения отправляются на сервер.  
  
5. Опишите порядок работы с общим хранилищем VCS.  
  
Участник проекта перед началом работы получает нужную ему версию проекта в хранилище, с помощью определенных команд, после внесения изменений пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются. К ним можно вернуться в любой момент.  
  
6. Каковы основные задачи, решаемые инструментальным средством git?  
  
Хранение информации о всех изменениях в вашем коде, обеспечение удобства командной работы над кодом.  
  
7. Назовите и дайте краткую характеристику командам git.  
  
Создание основного дерева репозитория: git init

Получение обновлений (изменений) текущего дерева из центрального репозитория: git pull Отправка всех произведённых изменений локального дерева в центральный репозиторий: git push Просмотр списка изменённых файлов в текущей директории: git status Просмотр текущих изменений: git diff Сохранение текущих изменений: добавить все изменённые и/или созданные файлы и/или каталоги: git add . добавить конкретные изменённые и/или созданные файлы и/или каталоги: git add имена\_файлов удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): git rm имена\_файлов Сохранение добавленных изменений: сохранить все добавленные изменения и все изменённые файлы: git commit -am ‘Описание коммита’ сохранить добавленные изменения с внесением комментария через встроенный редактор: git commit создание новой ветки, базирующейся на текущей: git checkout -b имя\_ветки переключение на некоторую ветку: git checkout имя\_ветки (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой) отправка изменений конкретной ветки в центральный репозиторий: git push origin имя\_ветки слияние ветки с текущим деревом: git merge –no-ff имя\_ветки Удаление ветки: удаление локальной уже слитой с основным деревом ветки: git branch -d имя\_ветки принудительное удаление локальной ветки: git branch -D имя\_ветки удаление ветки с центрального репозитория: git push origin :имя\_ветки

8. Приведите примеры использования при работе с локальным и удалённым репозиториями.  
  
git push -all отправляем из локального репозитория все сохраненные изменения в центральный репозиторий, предварительно создав локальный репозиторий и сделав предварительную конфигурацию.  
  
9. Что такое и зачем могут быть нужны ветви (branches)?  
  
Ветвление - один из параллельных участков в одном хранилище, исходящих из одной версии, обычно есть главная ветка. Между ветками, т. е. их концами возможно их слияние. Используются для разработки новых функций.  
  
10. Как и зачем можно игнорировать некоторые файлы при commit?  
  
Во время работы над проектом могут создаваться файлы, которые не следуют добавлять в репозиторий. Например, временные файлы. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов.

# Список литературы

1. Лабораторная работа № 2 URL: https://esystem.rudn.ru/mod/page/view.php?id=1098790