Отчёт по лабораторной работе №2

Дисциплина: Опреционные системы

Долгаев Евгений НММбд-01-24

Содержание

# 1 Цель работы

Изучить идеологию и применение средств контроля версий. Освоить умения по работе с git.

# 2 Задание

1. Выполнение лабораторной работы
   1. Создать базовую конфигурацию для работы с git.
      1. Создать ключ SSH.
      2. Создать ключ PGP.
      3. Настроить подписи git.
      4. Создать локальный каталог для выполнения заданий по предмету.
2. Ответы на контрольные вопросы

# 3 Выполнение лабораторной работы

Для начала установим git и gh (рис. 1).

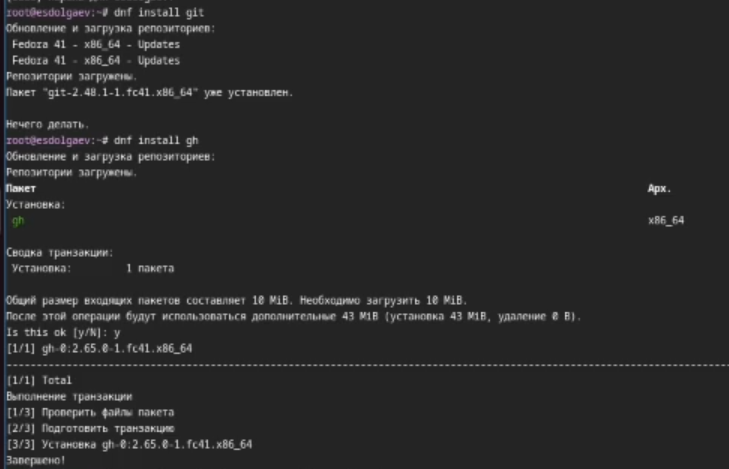


Рис. 1: Установка git и gh

Проведём первоначальную настройку git (рис. 2).

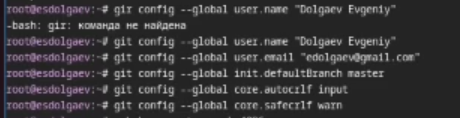


Рис. 2: Настройка git

Создадим SSH и GPG ключи (рис. 3).

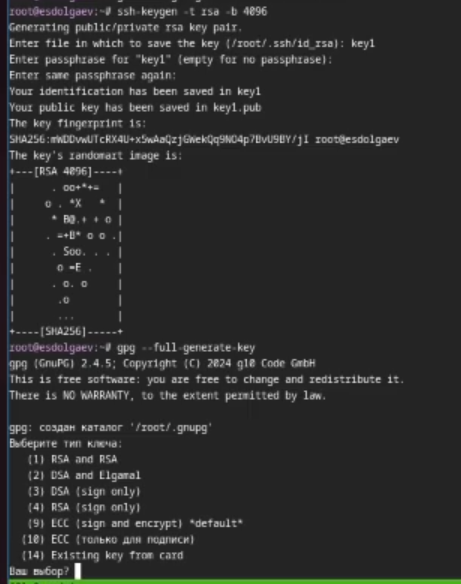


Рис. 3: Создание ключей

Загрузим ключи на github (рис. 4, 5).

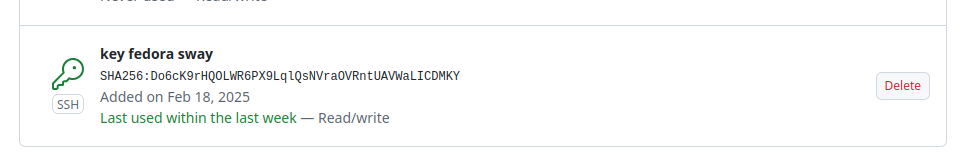


Рис. 4: Загруженный ключ

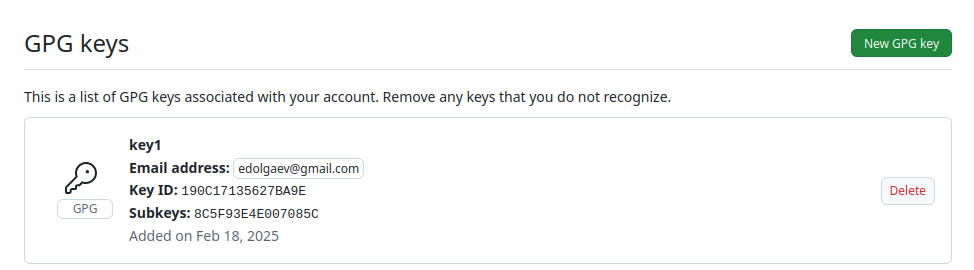


Рис. 5: Загруженный ключ

Настроим автоматических подписей коммитов git (рис. 6).

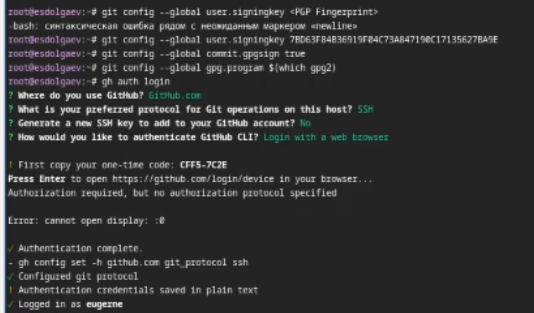


Рис. 6: Настройка автоматических подписей

Создадим рабочий каталог и скопируем в него наш созданный репозиторий (рис. 7, 8).

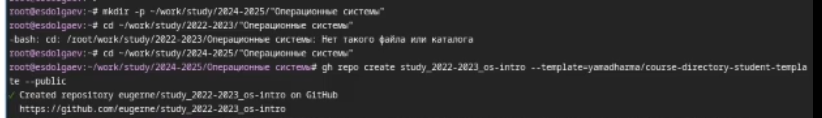


Рис. 7: Создание каталога

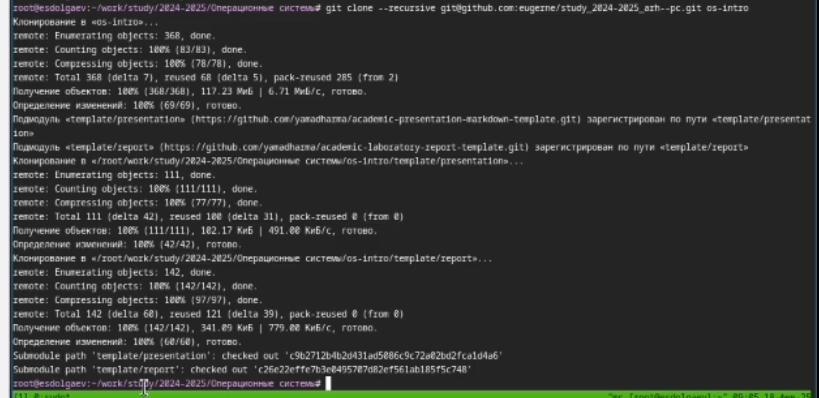


Рис. 8: Клонирование репозитория

Загрузим файлы на github (рис. 9).

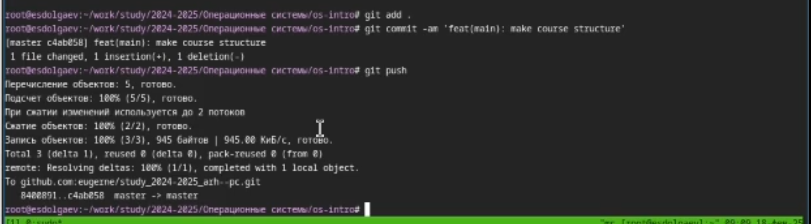


Рис. 9: Загрузка файлов на github

# 4 Ответы на контрольные вопросы

1. 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.
2. Хранилище - это единый репозиторий для хранения файлов. С помощью commit можно сохранить в репозитории добавленные изменения. Все изменения сохраняются в истории изменений. Рабочей копией называется файл, находящийся в хранилище.
3. В централизованной системе все пользователи подключены к центральному владельцу сети или «серверу». Центральный владелец хранит данные, к которым могут получить доступ другие пользователи, а также информацию о пользователях. У децентрализованных систем нет единого центрального владельца. Вместо этого они используют нескольких центральных владельцев, каждый из которых обычно хранит копию ресурсов, к которым пользователи могут получить доступ. Централизованные: Subversion, децентрализованные: Git
4. Действия с VCS 1.Стандартные процедуры работы при наличии центрального репозитория
   1. Работа пользователя со своей веткой начинается с проверки и получения изменений из центрального репозитория
   2. Затем можно вносить изменения в локальном дереве и/или ветке.
   3. Работа с локальным репозиторием
   4. Создание локального репозитория
   5. Первичная конфиграция репозитория
   6. Работа с сервером репозиториев
   7. Для идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый)
   8. Далее ключ нужно связать с репозиторием с) Теперь на локальном компьютере можно выполнять стандартные процедуры для работы с git при наличии центрального репозитория
5. Стандартные процедуры работы при наличии центрального репозитория

* 1.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

2. Затем можно вносить изменения в локальном дереве и/или ветке.  
3. 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

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

git diff

6. 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 …  
git rm …

7. Если нужно сохранить все изменения в текущем каталоге, то используем:

git add .

8. Затем сохраняем изменения, поясняя, что было сделано:

git commit -am "Some commit message"

9. Отправляем изменения в центральный репозиторий:

git push

1. Основные задачи, решаемые инструментальным средством git
   1. Как не потерять файлы с исходным кодом?
   2. Как защититься от случайных исправлений и удалений?
   3. Как отменить изменения, если они оказались некорректными?
   4. Как одновременно поддерживать рабочую версию и разработку новой?
2. Основные команды git
   1. Создание основного дерева репозитория:

git init

2. Получение обновлений (изменений) текущего дерева из центрального репозитория:

git pull

3. Отправка всех произведённых изменений локального дерева в центральный репозиторий:

git push

4. Просмотр списка изменённых файлов в текущей директории:

git status

5. Просмотр текущих изменений:

git diff

6. Сохранение текущих изменений:  
  
a) добавить все изменённые и/или созданные файлы и/или каталоги:

git add .

b) добавить конкретные изменённые и/или созданные файлы и/или каталоги:

git add имена\_файлов

c) удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории):

git rm имена\_файлов

7. Сохранение добавленных изменений:  
  
a) сохранить все добавленные изменения и все изменённые файлы:

git commit -am ‘Описание коммита’

b) сохранить добавленные изменения с внесением комментария через встроенный редактор:

git commit

c) создание новой ветки, базирующейся на текущей:

git checkout -b имя\_ветки

d) переключение на некоторую ветку:

git checkout имя\_ветки

e) отправка изменений конкретной ветки в центральный репозиторий:

git push origin имя\_ветки

f) слияние ветки с текущим деревом:

git merge –no-ff имя\_ветки

8. Удаление ветки:  
  
a) удаление локальной уже слитой с основным деревом ветки:

git branch -d имя\_ветки

b) принудительное удаление локальной ветки:

git branch -D имя\_ветки

c) удаление ветки с центрального репозитория:

git push origin :имя\_ветки

1. Команда status для просмотра изменений в рабочем каталоге, сделанных с момента последней ревизии:

git status

Команда, которая связывается с указанным удалённым проектом и забирает все те данные проекта, которых у вас ещё нет.

git fetch []

1. Команда git branch — главный инструмент для работы с ветвлением. С ее помощью можно добавлять новые ветки, перечислять и переименовывать существующие и удалять их.
2. 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 …  
git rm …

# 5 Выводы

В ходе выполнения лабораторной работы я освоил навыки по работе с git.

# Список литературы