Лабораторная работа №2

Отчёт

Ермишина Мария Кирилловна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Целью данной лабораторной работы являются: изучение идеологии и применение средств контроля версий; освоение умения по работе с git. [1]

2 Задание

- Создать базовую конфигурацию для работы с git.
- Создать ключ SSH.
- Создать ключ PGP.
- Настроить подписи git.
- Зарегистрироваться на Github.
- Создать локальный каталог для выполнения заданий по предмету.

3 Теоретическое введение

Основные команды git	Описание каталога
`git init`	Создание основного дерева репозитория
`git pull`	Получение обновлений (изменений) текущего дерева из центрального репозитория
`git push`	Отправка всех произведённых изменений локального дерева в центральный репозиторий
`git status`	Просмотр списка изменённых файлов в текущей директории
`git diff`	Просмотр текущих изменений
`git add .`	добавить все изменённые и/или созданные файлы и/или каталоги
`git add имена_файлов`	добавить конкретные изменённые и/или созданные файлы и/или каталоги
`git rm имена_файлов`	удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории)

Основные команды git	Описание каталога
`git commit -am 'Описание коммита'`	сохранить все добавленные изменения и все изменённые файлы
`git commit`	сохранить добавленные изменения с внесением комментария через встроенный редактор
`git checkout -b имя_ветки`	создание новой ветки, базирующейся на текущей
`git checkout имя_ветки`	переключение на некоторую веткупереключение на некоторую ветку (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой)
`git push origin имя_ветки`	отправка изменений конкретной ветки в центральный репозиторий
`git merge --no-ff имя_ветки`	слияние ветки с текущим деревом

Основные команды git	Описание каталога
`git branch -d имя_ветки`	удаление локальной уже слитой с основным деревом ветки
`git branch -D имя_ветки`	принудительное удаление локальной ветки
`git push origin :имя_ветки`	удаление ветки с центрального репозитория
`git push origin :имя_ветки`	

4 Выполнение лабораторной работы

1. Установа git и gh. Их мы устанавливаем с помощью команд:

- dnf install git (рис. 4.1)
- dnf install gh (рис. 4.2)

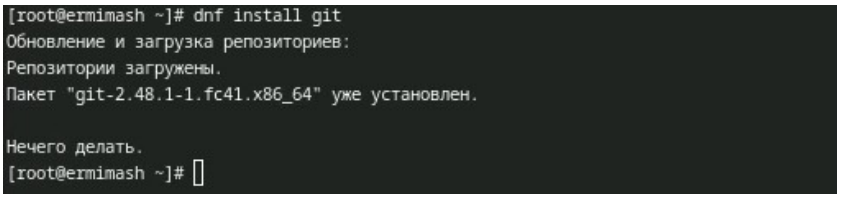

Рис. 4.1: Установка git

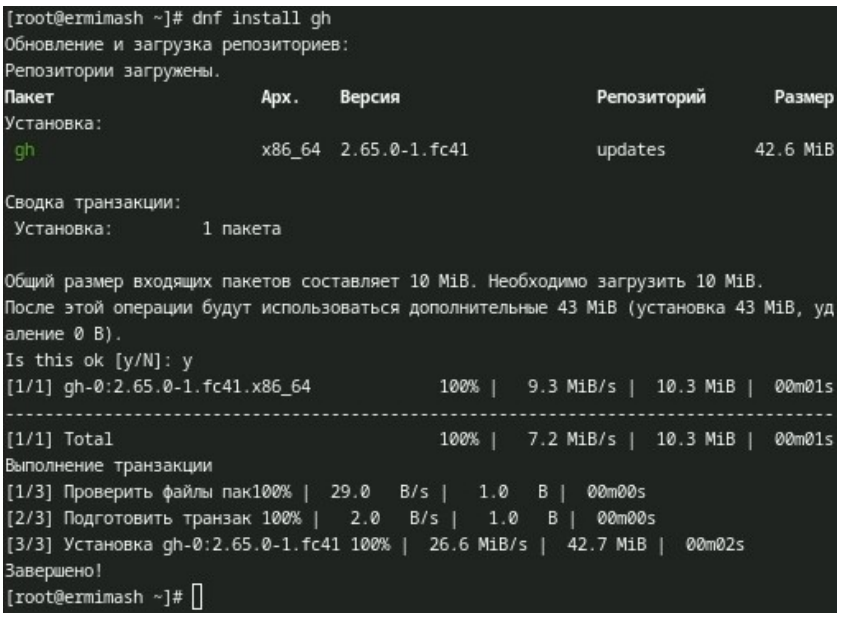

Рис. 4.2: Установка gh

2. Базовая настройка git. (рис. 4.3)

3. Зададим имя и email владельца репозитория с помощью команд: git config –global user.name “Name Surname” git config –global user.email “work@mail”

4. Настроим utf-8 в выводе сообщений git: git config –global core.quotepath false

5. Зададим имя начальной ветки (будем называть её master): git config –global init.defaultBranch master

6. Параметры autocrlf и safecrlf: git config –global core.safecrlf warn

```
[root@ermimash ~]# git config --global user.name "ErmiMash"
[root@ermimash ~]# git config --global user.email "1132230166@pfur.ru"
[root@ermimash ~]# git config --global core.quotepath false
[root@ermimash ~]# git config --global init.defaultBranch master
[root@ermimash ~]# git config --global core.autocrlf input
[root@ermimash ~]# git config --global core.safecrlf warn
```

Рис. 4.3: Базовая настройка git

3. Создаём ключи SSH. (рис. 4.4) Их мы создаём с помощью команд:

- ssh-keygen -t rsa -b 4096 (создание по алгоритму *rsa* с размером 4096 бит)
- ssh-keygen -t ed25519 (создание по алгоритму *ed25519*)

Рис. 4.4: Создание SSH

4. Создаём ключ PGP. (рис. 4.5) Используем команду для генерации ключа:

- gpg –full-generate-key

Данный ключ мы создаём со следующими опциями: 1. тип RSA and RSA 2. размер 4096 3. 0 (срок действия не истекает никогда)

Также указфываем свои данные (Имя, Почту (которая соответствует почте на GitHub), комментарий), которые будут храниться в ключе.

Рис. 4.5: Создание PGP

5. Настройка GitHub Данный пункт я пропустила, так как уже имею аккаунт на GitHub, который я создавала в 1 семестре учёбы.

6. Добавление PGP ключа в GitHub.

7. Выводим список ключей и копируем отпечаток приватного ключа с помощью следующей команды: gpg –list-secret-keys –keyid-format LONG (рис. 4.6)

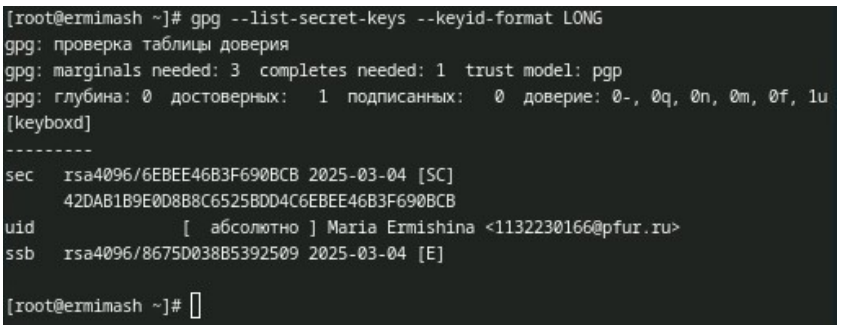

Рис. 4.6: Вывод ключей

2. Копируем сгенерированный ключ PGP в буфер обмена (с помощью команды получаем полный код, а далее вручную его копируем): gpg –armor –export xclip -sel clip (рис. 4.7)

Рис. 4.7: Копирование PGP

Вставляем наш ключ на сайте GitHub и даём ему название. (рис. 4.8) Проверяем наличие ключа, обновив страницу. (рис. 4.9)

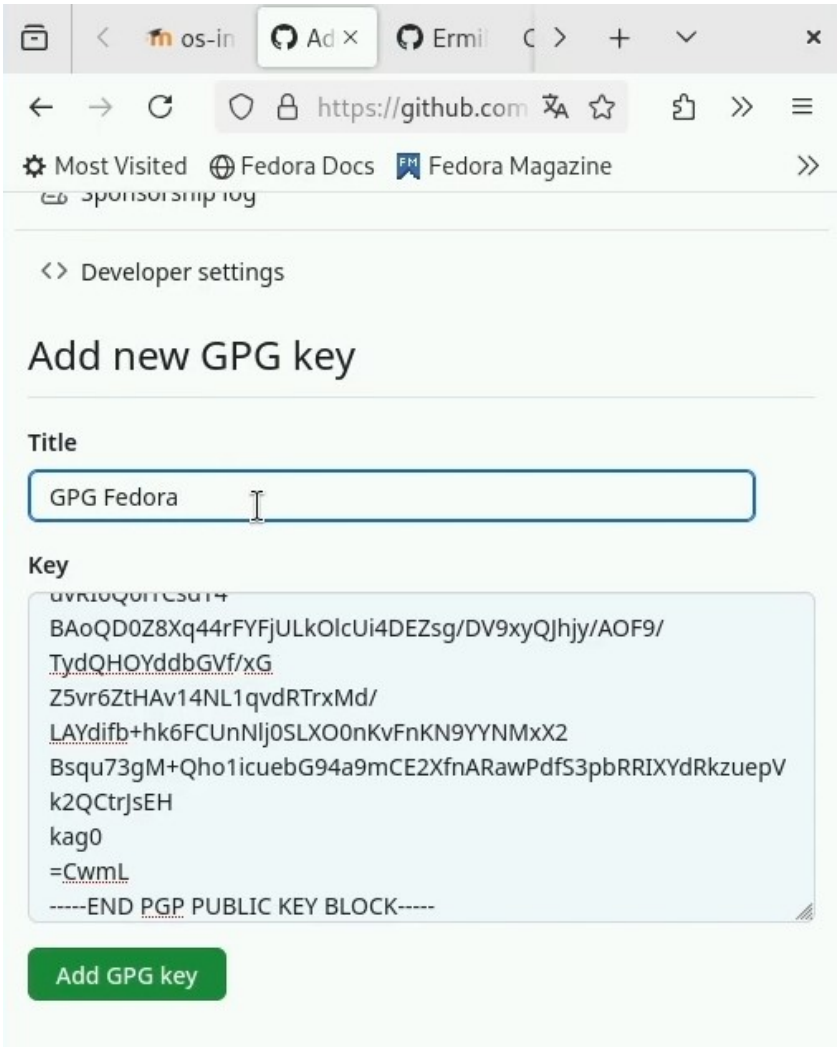

Рис. 4.8: Ключ на Git

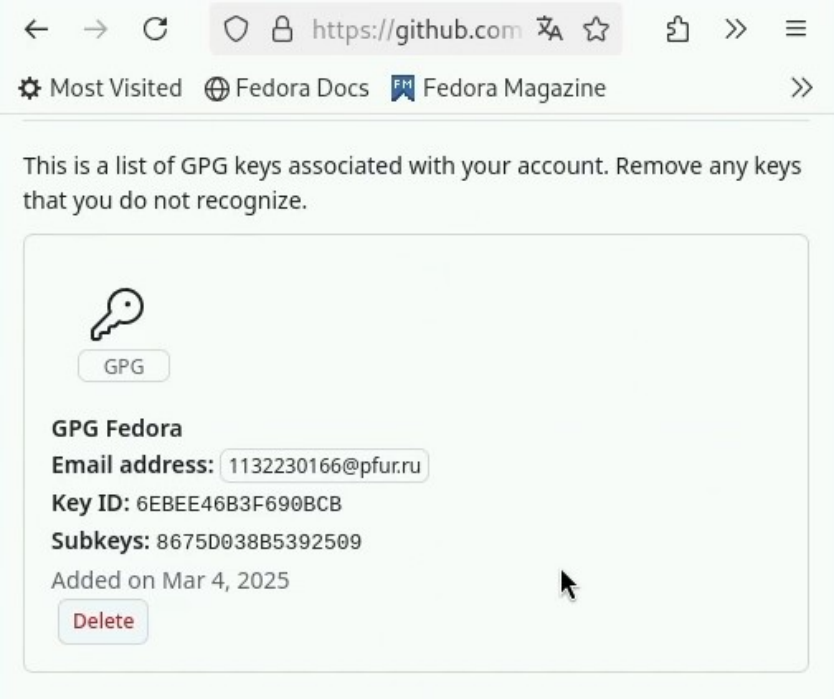

Рис. 4.9: Наличие ключа

7. Настройка автоматических подписей коммитов git. (рис. 4.10) Используя введёный email, укажите Git применять его при подписи коммитов с помощью следующих команд:

- git config –global user.signingkey
- git config –global commit.gpgsign true
- git config –global gpg.program $(which gpg2)

```
[root@ermimash ~]# git config --global user.signingkey 1132230166@pfur.ru
[root@ermimash ~]# git config --global commit.gpgsign true
[root@ermimash ~]# git config --global gpg.program $(which gpg2)
[root@ermimash ~]#
```

Рис. 4.10: Настройка авто-подписей

8. Настройка gh. (рис. 4.11) Мы авторизуемся с помощью команды:

- gh auth login

После ввода команды мы отвечаем на несколдько вопросов и авторизуемс через браузер. При успешной авторизации получаем данную информацию в терминале (рис. 4.11) и на сайте (рис. 4.12).

Рис. 4.11: Терминал

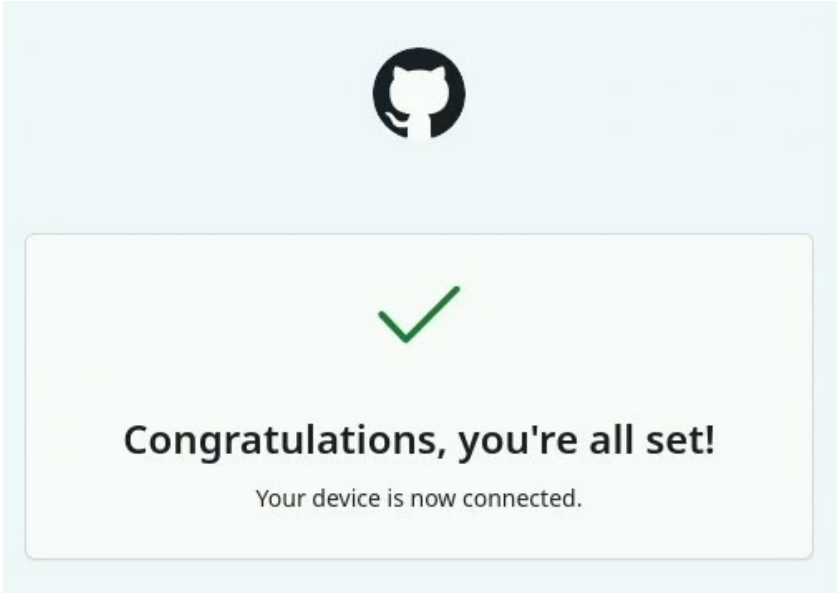

Рис. 4.12: Подтверждение на сайте

9. Создание репозитория курса на основе шаблона. Для начала мы создаём каталог для создания шаблона с помощью следующей команды:

- mkdir -p ~/work/study/2024-2025/“Операционные системы” (рис. 4.13)

Далее мы переходим в созданный каталог и создаём шаблон рабочего пространства, указывая актуальный учебный год и название предмета, с помощью следующих команд: - cd ~/work/study/2024-2025/“Операционные системы” (рис. 4.13) - gh repo create study_ 2024-2025_os-intro –template=yamadharma/course-directory-student-template –public (рис. 4.13) - git clone –recursive git@github.com:/study_2022-2023_os-intro.git os-intro (рис. 4.14)

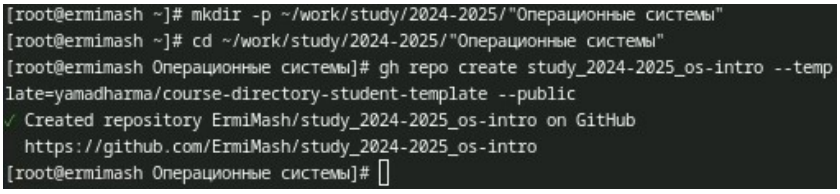

Рис. 4.13: Создание каталогов

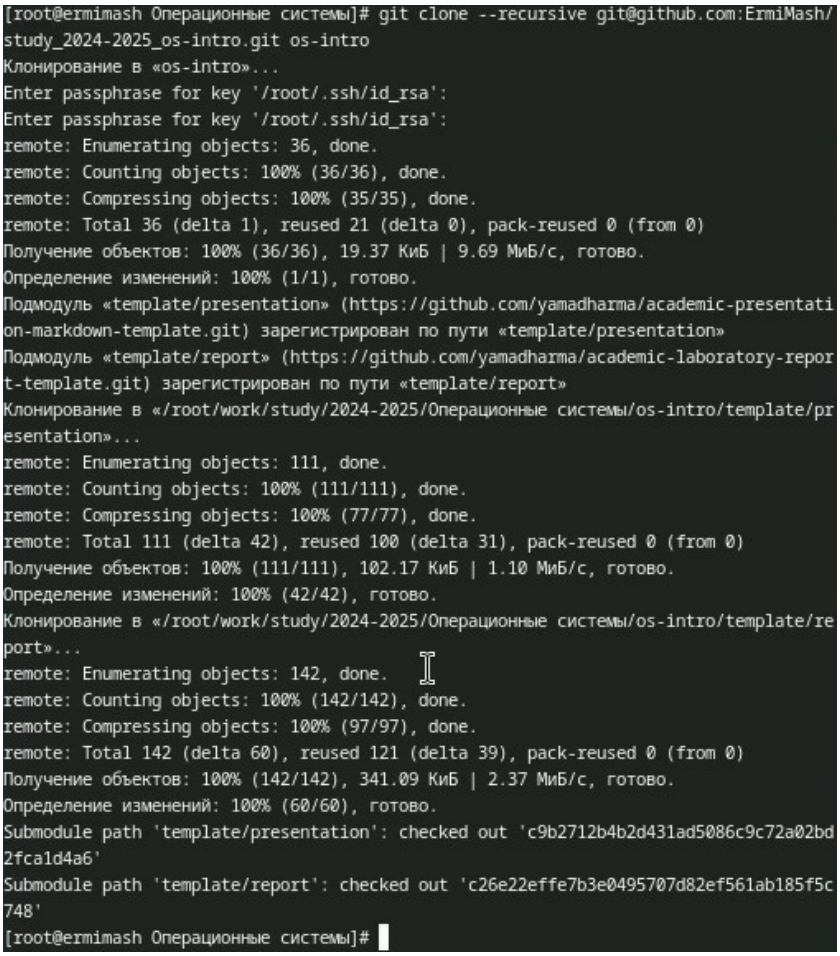

Рис. 4.14: git clone

Проверяем наличие созданного шаблона на сайте. (рис. 4.15)

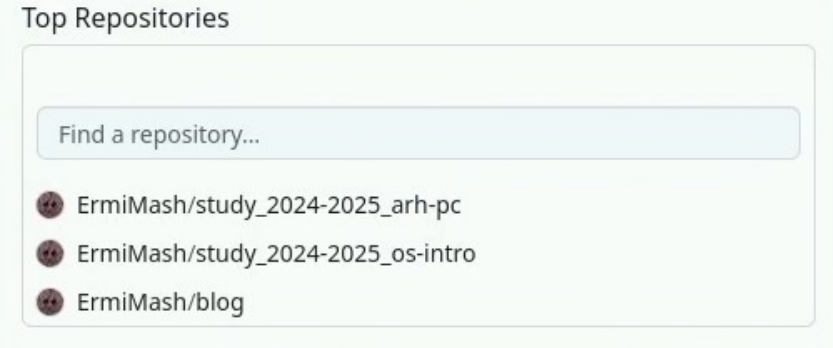

Рис. 4.15: Наличие шаблона на сайте

10. Настройка каталога курса. Мы переходим в каталог курса и удаляем лишние файлы с помощью следующих команд: (рис. 4.16)

- cd ~/work/study/2022-2023/“Операционные системы”/os-intro
- rm package.json

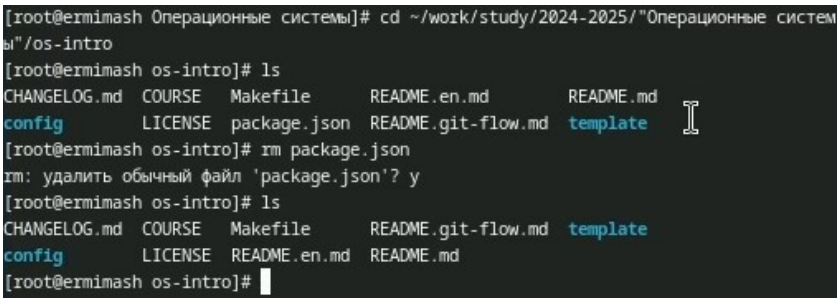

Рис. 4.16: Удаление файлов

Далее создаём необходимфе каталоги: (рис. 4.17) - echo os-intro > COURSE - make prepare

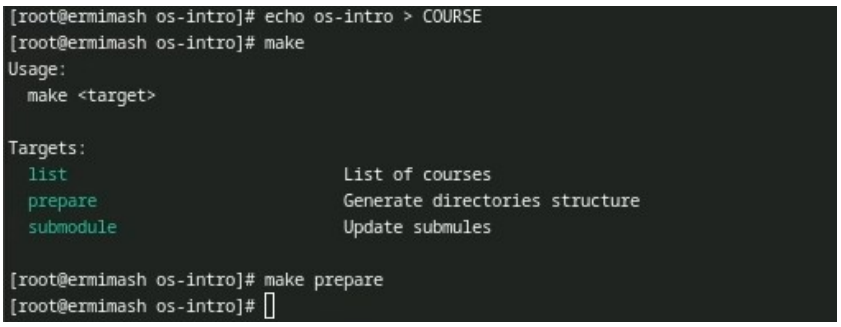

Рис. 4.17: Создание каталогов

После создания каталогов отправляем файлы на сервер: (рис. 4.18) - git add . - git commit -am ‘feat(main): make course structure’ - git push

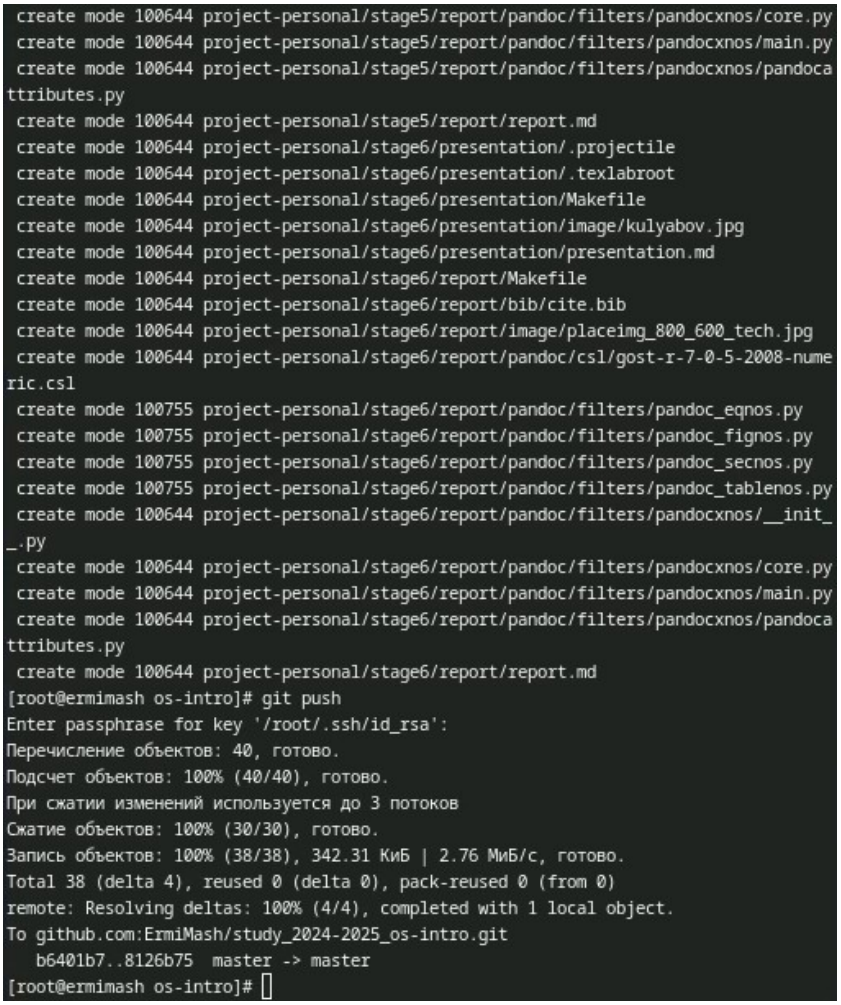

Рис. 4.18: Отправка файлов на сервер

5 Выводы

Изучила идеологии и применение средств контроля версий; освоила умения по работе с git.

6 Контрольные вопросы

1. Это програмное обеспечение для облегчения работы с изменяющейся информаицей. Позволяет хранить несколько версий одного документы, возвращаться к более ранним версиям, опередилять кто и когда сделал изменение, и многое другое.
2. Хранилище - место хранения всех версий и служеб. информации. Commit - версия; процесс создания новой версии. История - место, где хранятся все коммиты, по которым можно смотреть данные по коммитам. Рабочая копия - тек. состояние файла проекта, осн. на версии из хранилища.
3. Централизованные VCS: одно основное хранилище всего проекта, каждый копирует себе файлы и изменяет, а потом добавляет изменения; Децентрализованные: у каждого пользователя свой вариант (возможно не один) репозитория.
4. Инициализация репозитория, добавление файлов, фиксация изменений, проверка статуса, просмотр истории коммитов, создание и слияние веток, удаление веток, откат изменений, резервное копирование.
5. Клонирование репозитория, создание новой ветки, создание новой ветки, внесение изменений, добавление изменений в индекс, создание коммита, обновление локальной ветки, слияние изменений, отправка изменений в удалённый репозиторий, открытие Pull Request (PR), код-ревью, слияние PR, регулярное обновление.
6. Две основные задачи: 1 - хранить информацию о всех изменениях в коде, начиная с самой первой строчки; 2 - обеспечивать удобство командной

работы с кодом.

7. git init (инициализировать тек. каталог как git-репозиторий), git -version (проверка версии Git), git remote (посмотреть список удалённых репозиториев), git remote -v (более подробный вывод), git add . (позвол. охватить все файлы в тек. каталоге (вместе с теми, которые начинаются на точку)), git -htlp (узнать больше о доступных параметрах и командах), git push origin master (передать лок. коммиты в ветку удал. репозитория)
8. Локальный: создание новой ветки, работа без интернета, сохранение изменений; удалённый: отправки кодов, получение изменений, создание новых веток.
9. Ветки нужны, чтобы несколько программистов могли работать над одним и тем же проектом (или даже файлом) одновременно, не мешая друг другу. Также используются для тестов: чтобы не мешать основному поекту создаётся отдельная ветка для экспериментов.
10. Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты. Игнорируемые файлы отслеживаются в специальном файле .gitignore, который регистрируется в корневом каталоге репозитория.

Список литературы

1. Д. К. Лабораторная работа № 3. Markdown. RUDN.