**Лабораторная работа №2**

**Отчёт**

Лемуш Гонсалвеш Дуарти Афонсу Де

**Содержание**

1. **Цель работы**
2. **Задание**
3. **Выполнение лабораторной работы**
4. **Выводы**
5. **Ответы на контрольные вопросы**

**Список литературы**

1. **Цельработы**

Поглубже изучить концепцию и практику использования систем контроля версий. Приобрести навыки работы с Git.

1. **Задание**

1. Установить Git на компьютер, если он не установлен.

2. Создать учетную запись на GitHub.

3. Сгенерировать SSH-ключ для безопасной работы с репозиториями на GitHub.

4. Добавить SSH-ключ в настройки GitHub.

5. Создать PGP-ключ для подписи коммитов и тегов в Git.

6. Настроить глобальные параметры пользователя и электронной почты в Git.

7. Создать локальный каталог для клонирования репозиториев и выполнения заданий.

1. **Выполнение лабораторной работы**

Для начала установим git. В моём случае он уже установлен (рис. 3.1)

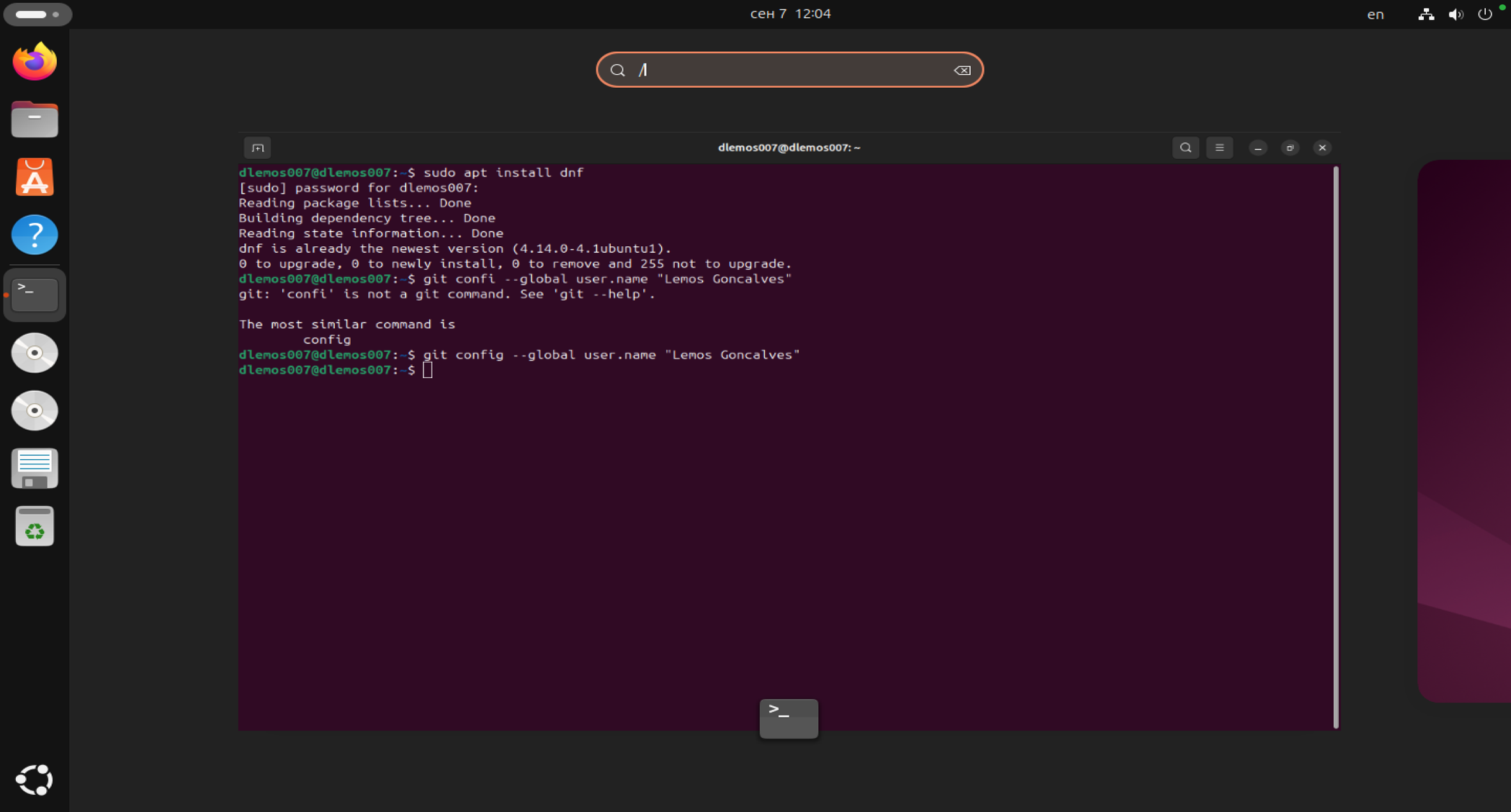
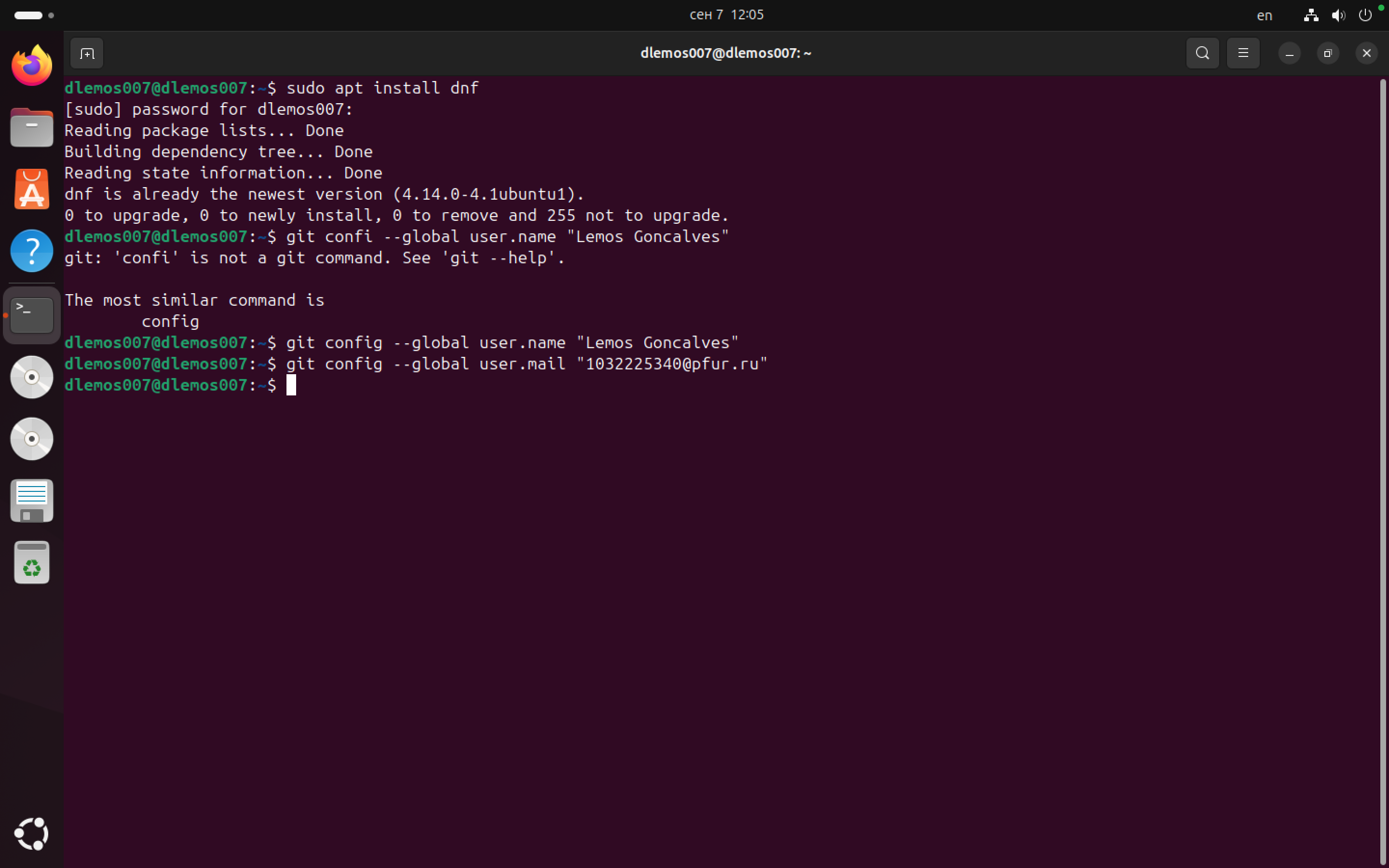


Рис. 3.1: Установка git

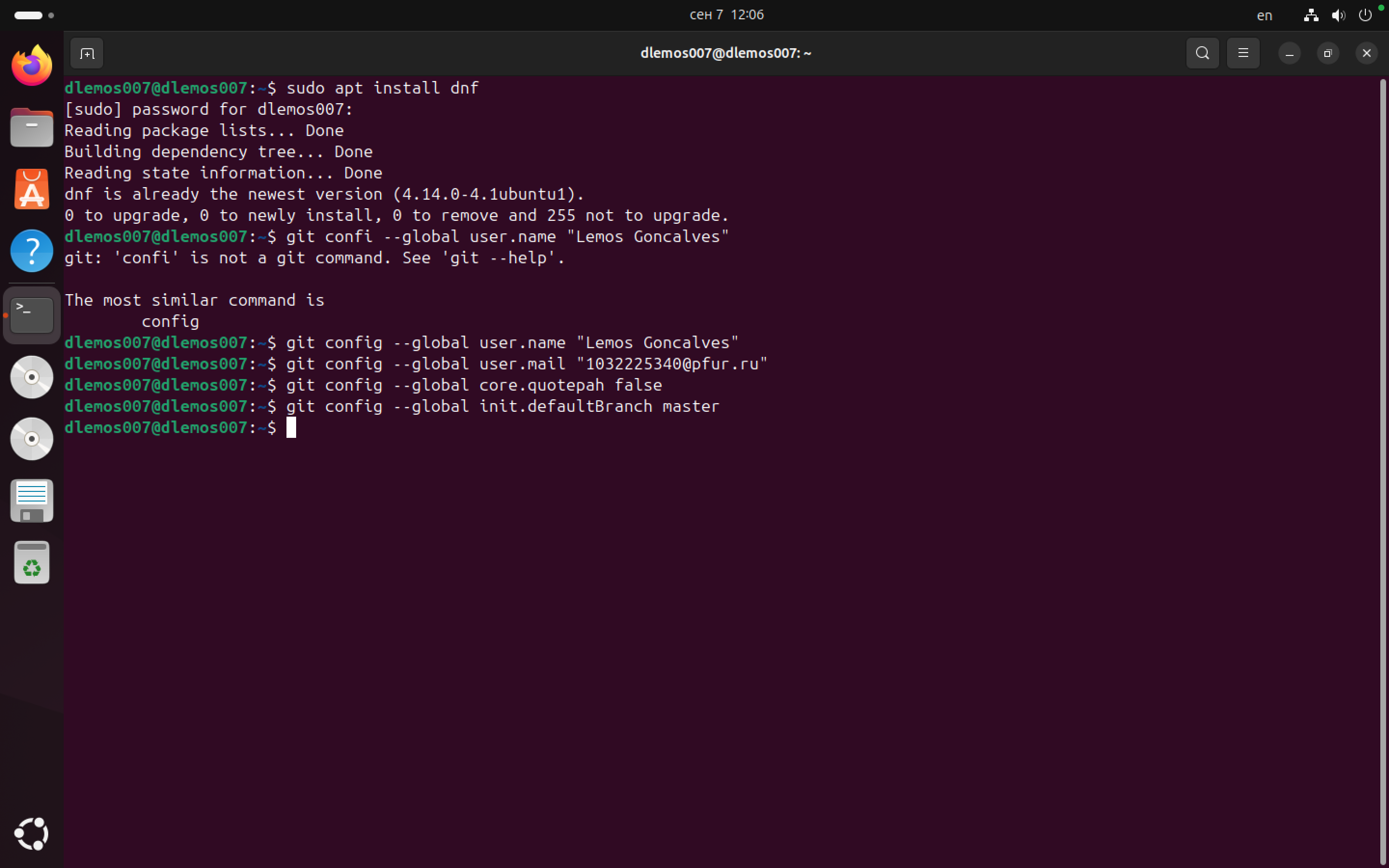
Теперь установим gh (рис. 3.2)



## Рис. 3.2: Установка gh

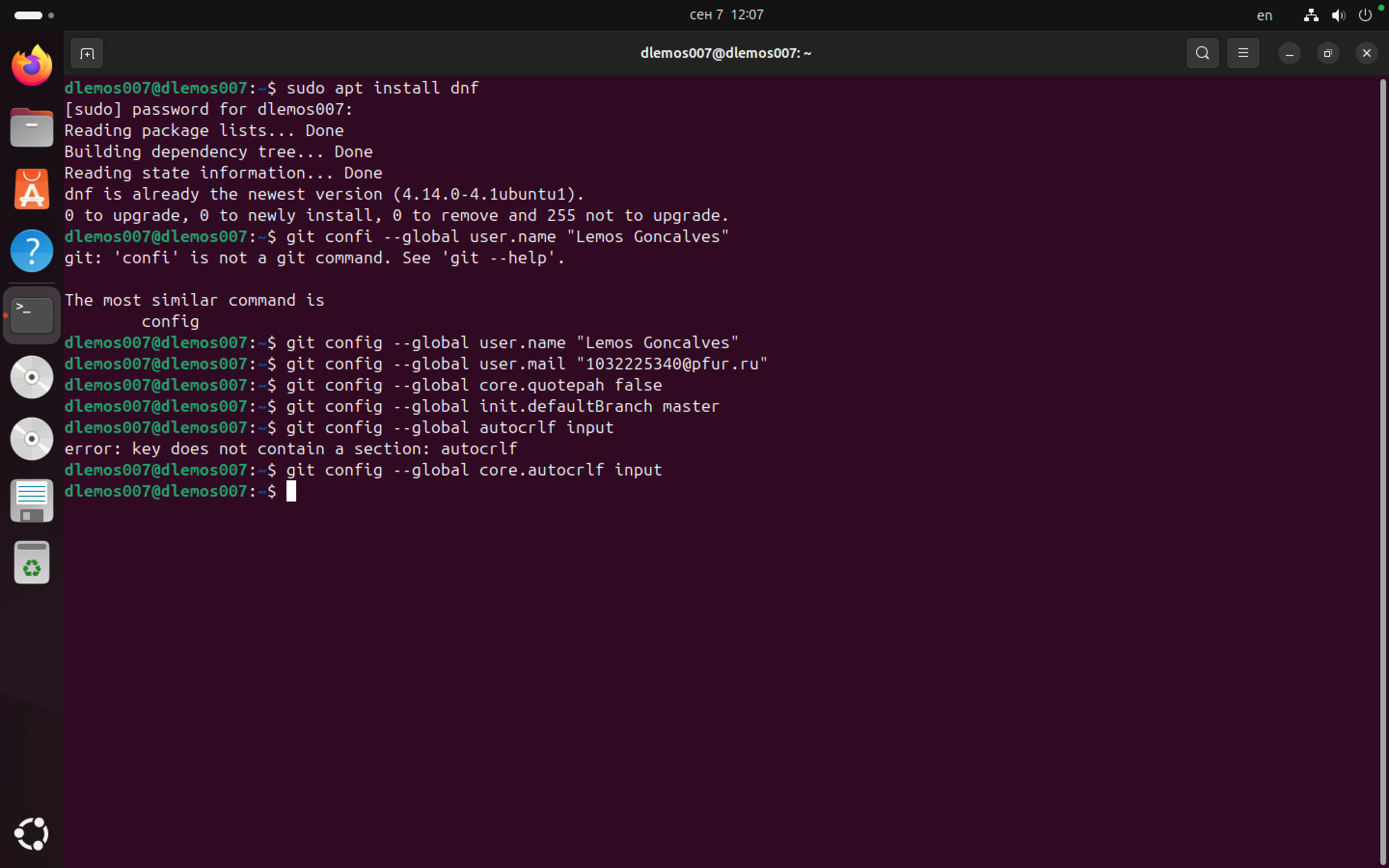
Далее, зададим имя для владельца репозитория. В данном случае это моё имя

(рис. 3.3)



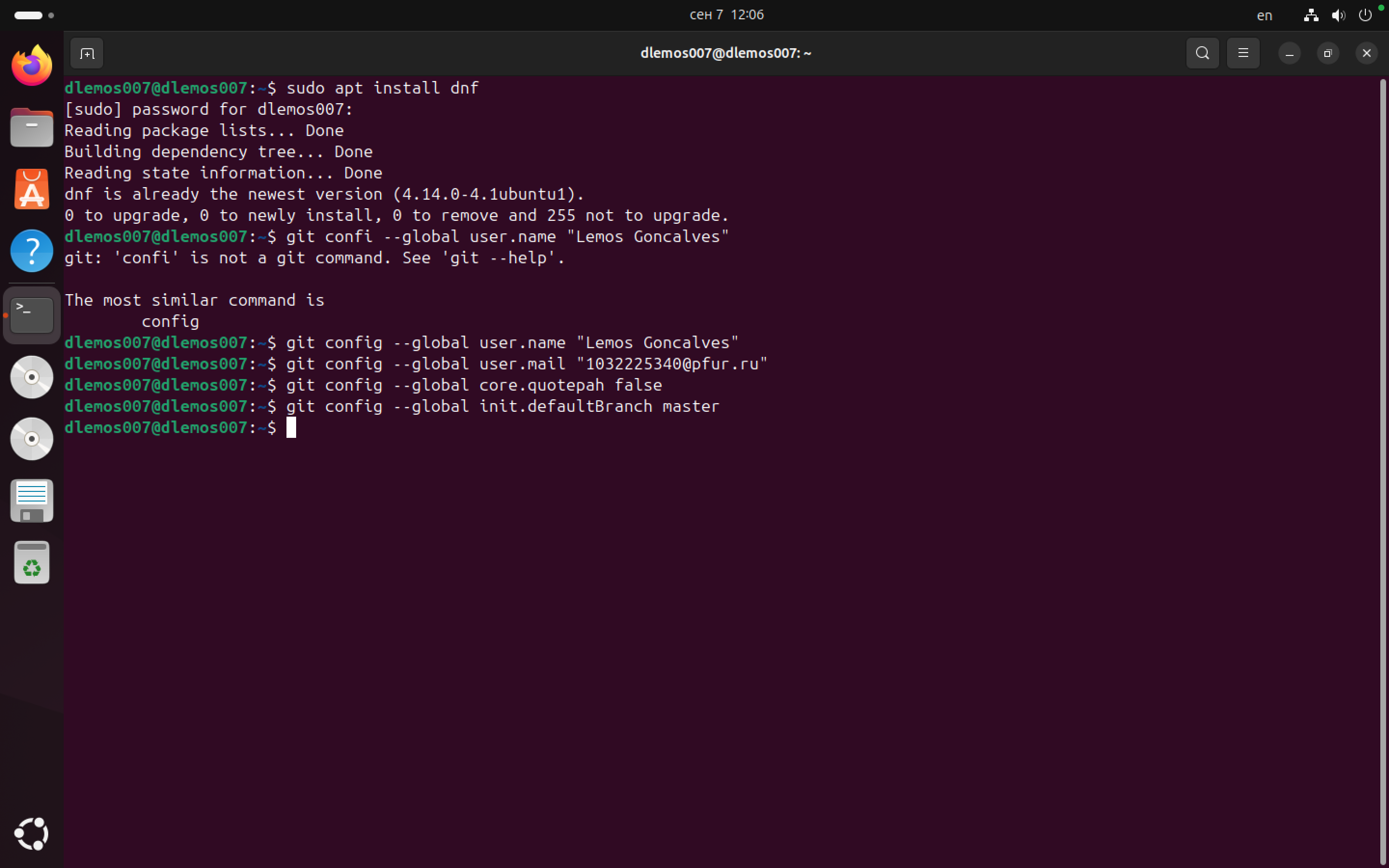
## Рис. 3.3: Указание имени

Теперь зададим почту. Я задал почту, на которую у меня зарегистрирован аккаунт на github (рис. 3.4)



## Рис. 3.4: Указание почты

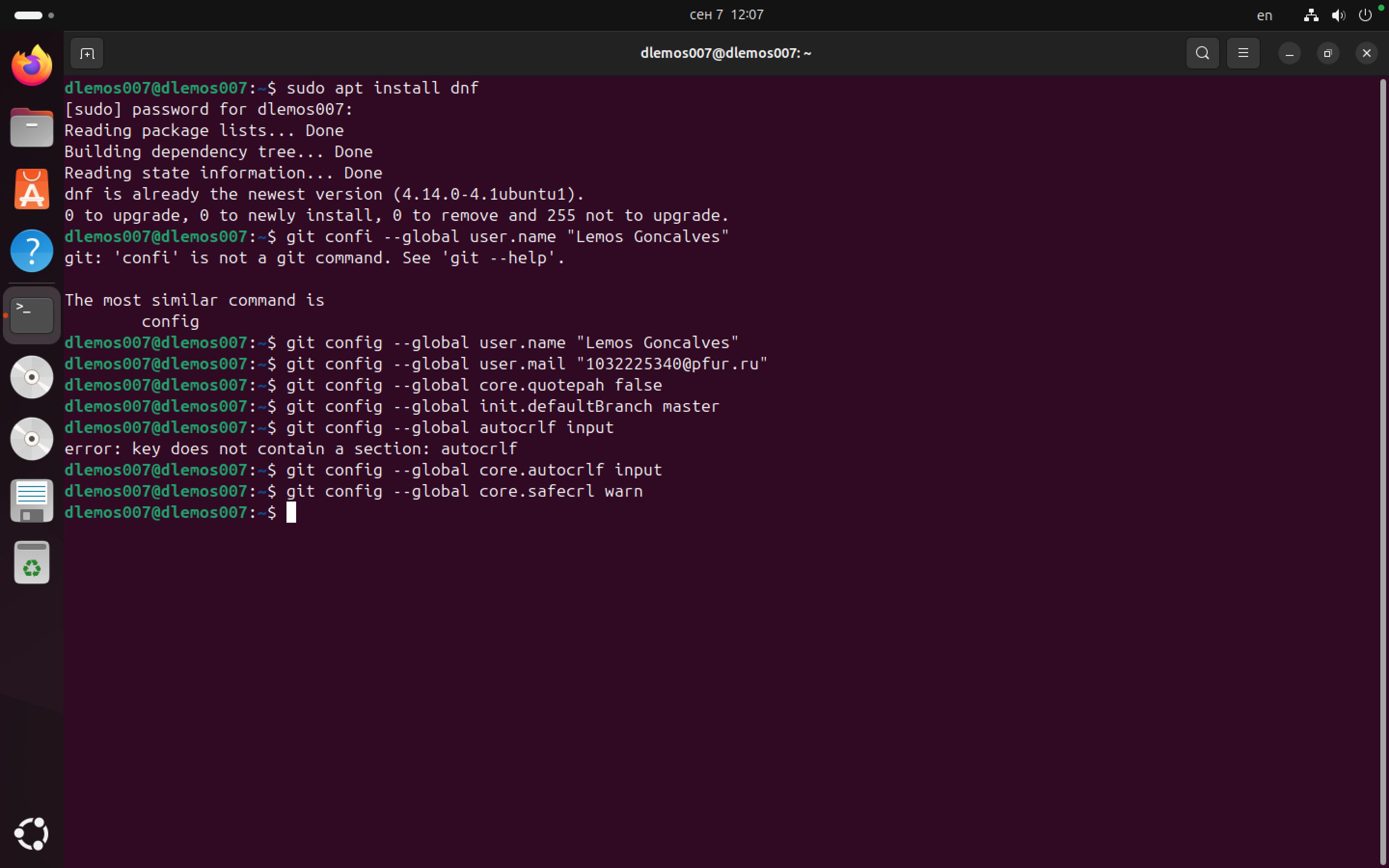
Настроим кодировку utf8 в выводе сообщений git (рис. 3.5)



## Рис. 3.5: Настройка кодировки utf8

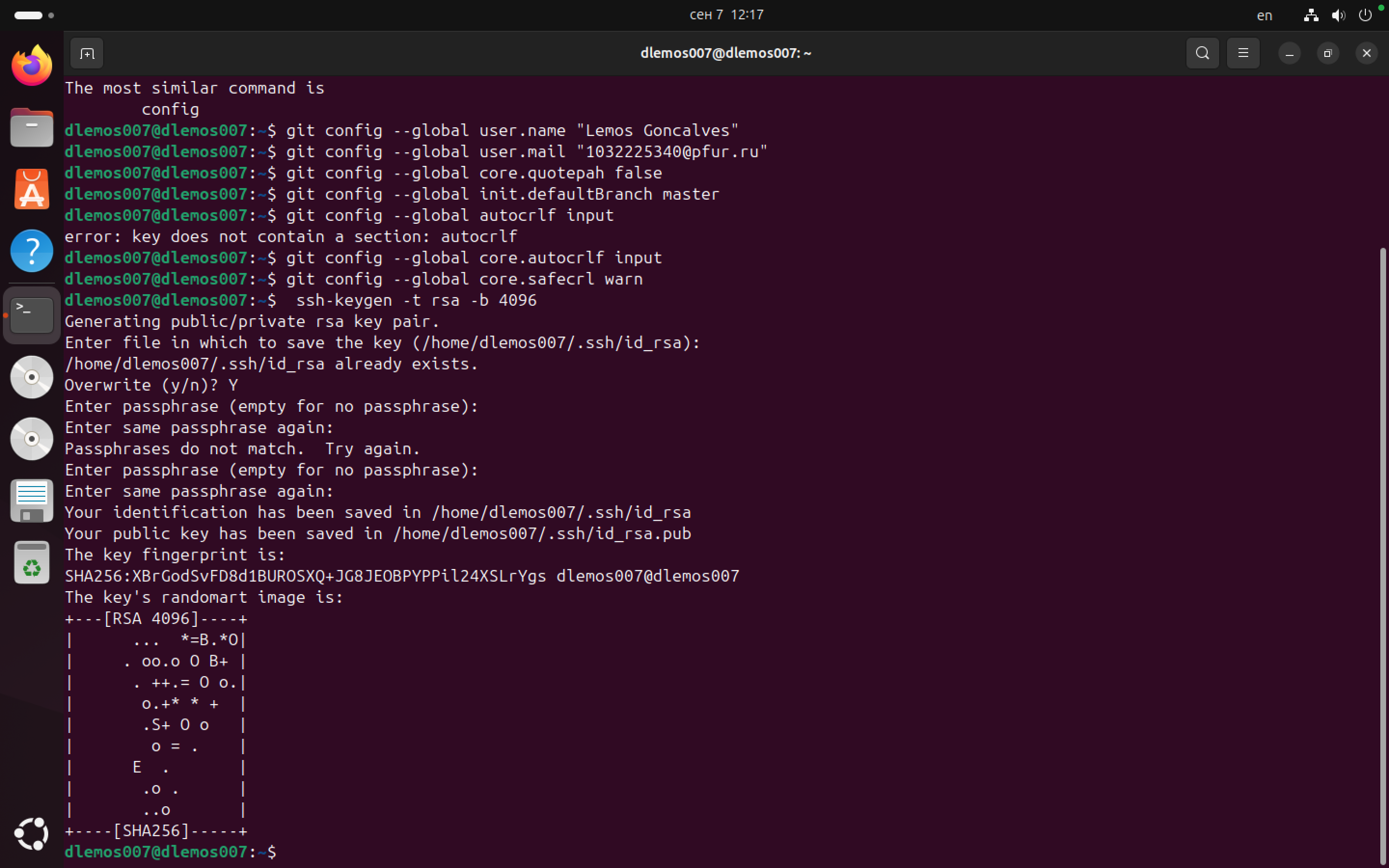
Зададим имя начальной ветки, настроим параметры autocrlf и safecrlf (рис.

3.6)

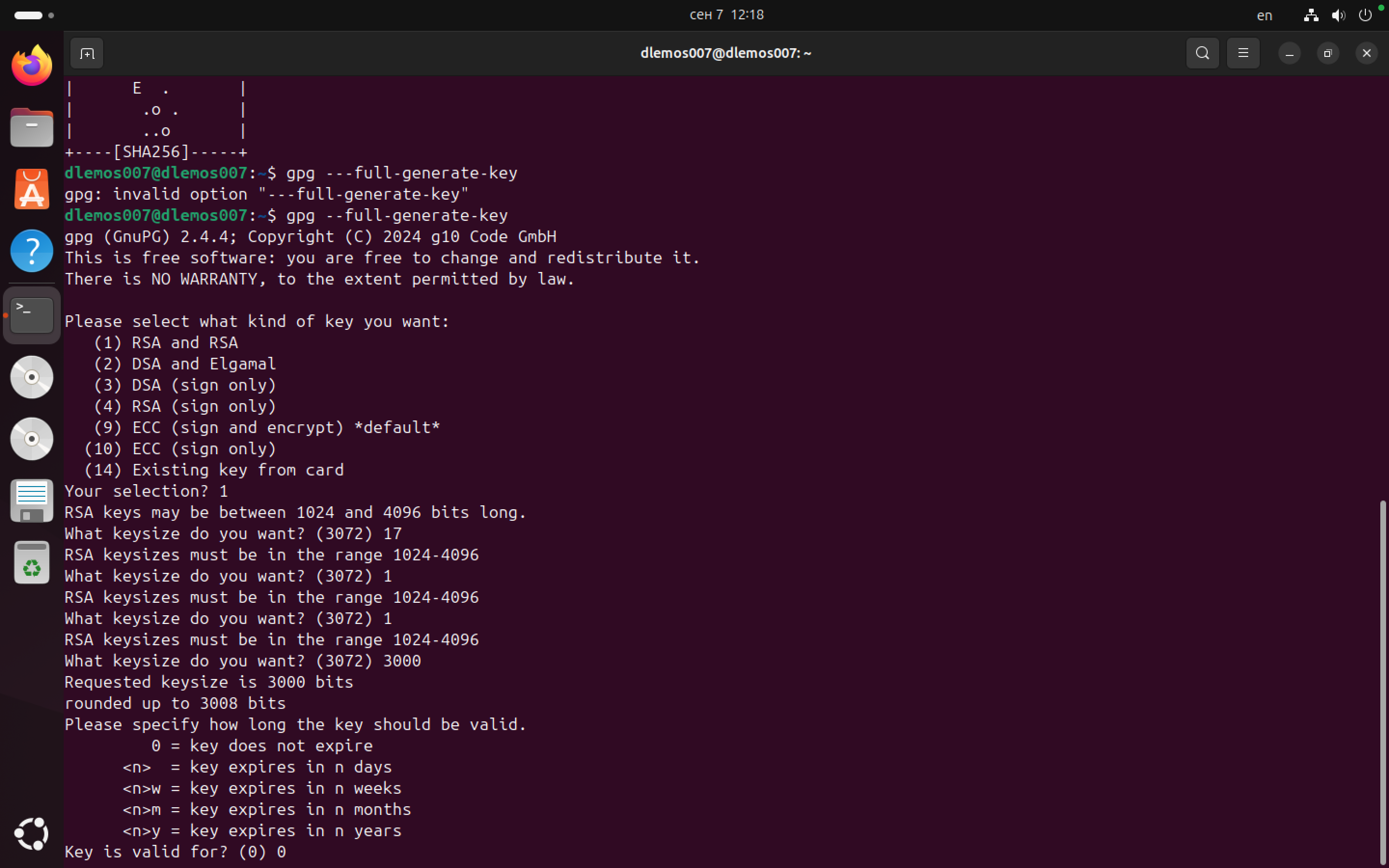


## Рис. 3.6: Настройка git

Создадим ключ RSA размером 4096 бит (рис. 3.7)

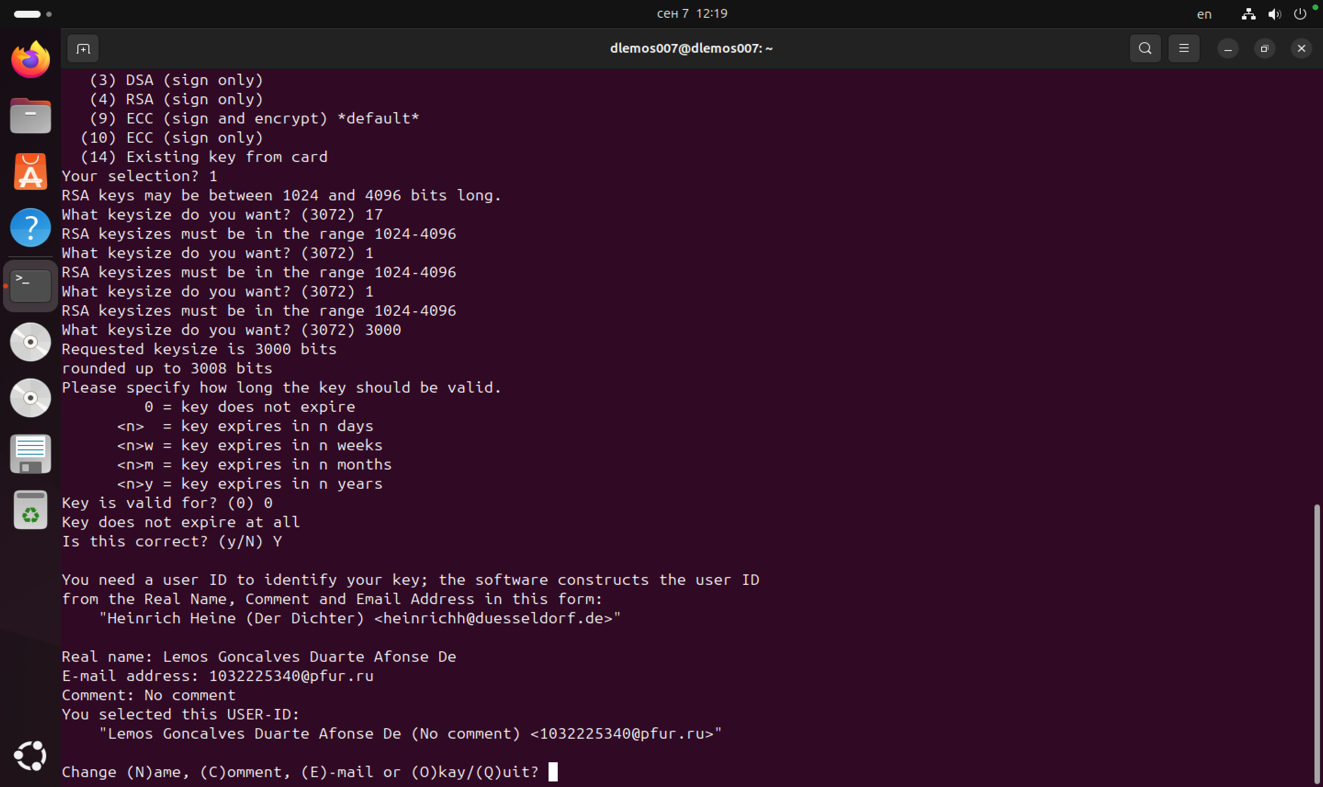


## Рис. 3.7: Создание ключа RSA



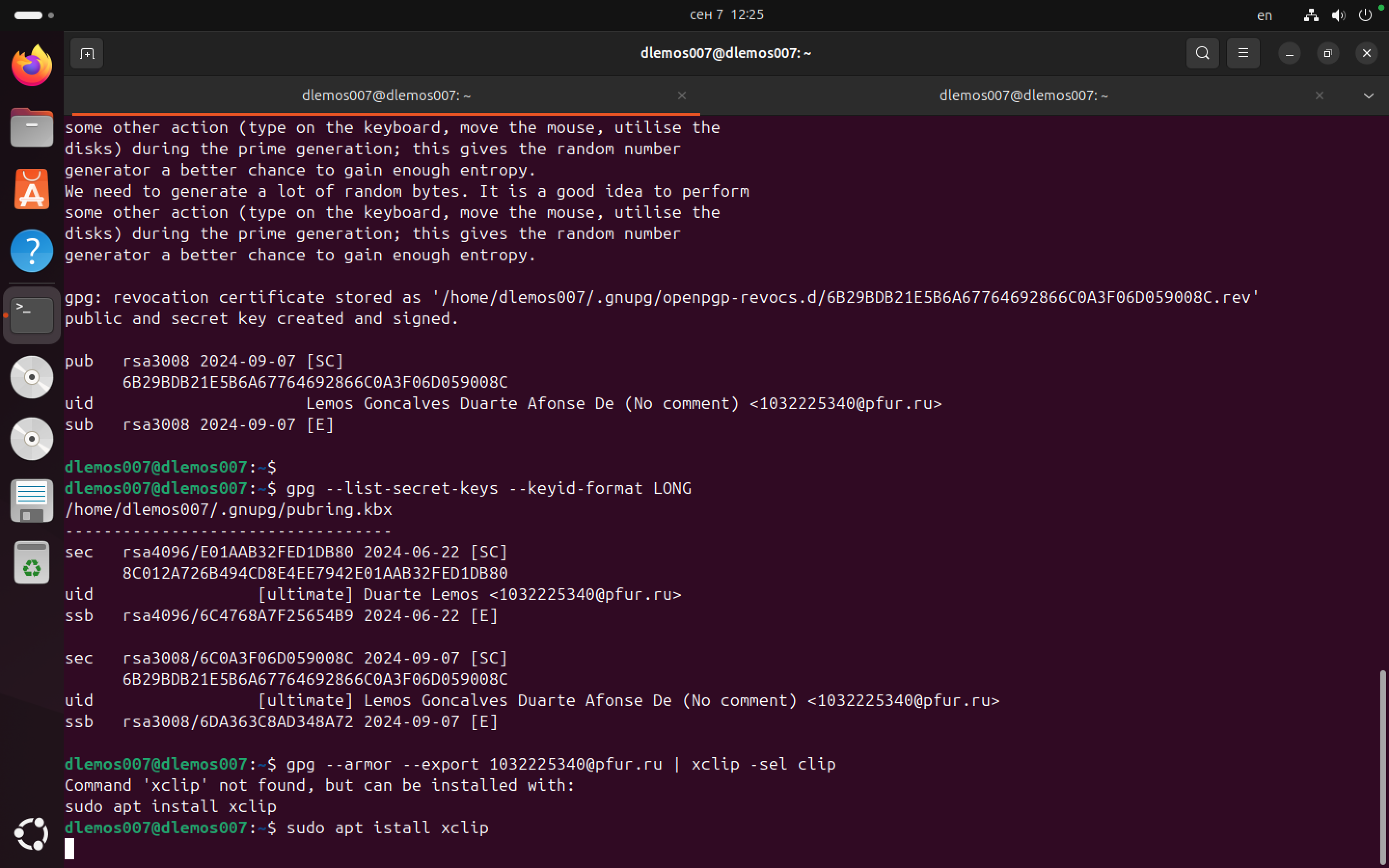
## Рис. 3.9: оздание ключа pgp (1)

После нас попросят ввести свои данные. Мы вводим имя и адрес электронной почты. После этого соглашаемся с генерацией ключа (рис. 3.10)



## Рис. 3.10: оздание ключа pgp (2)

Далее, выводим список pgp ключей (рис. 3.11)



## Рис. 3.11: Список pgp ключей

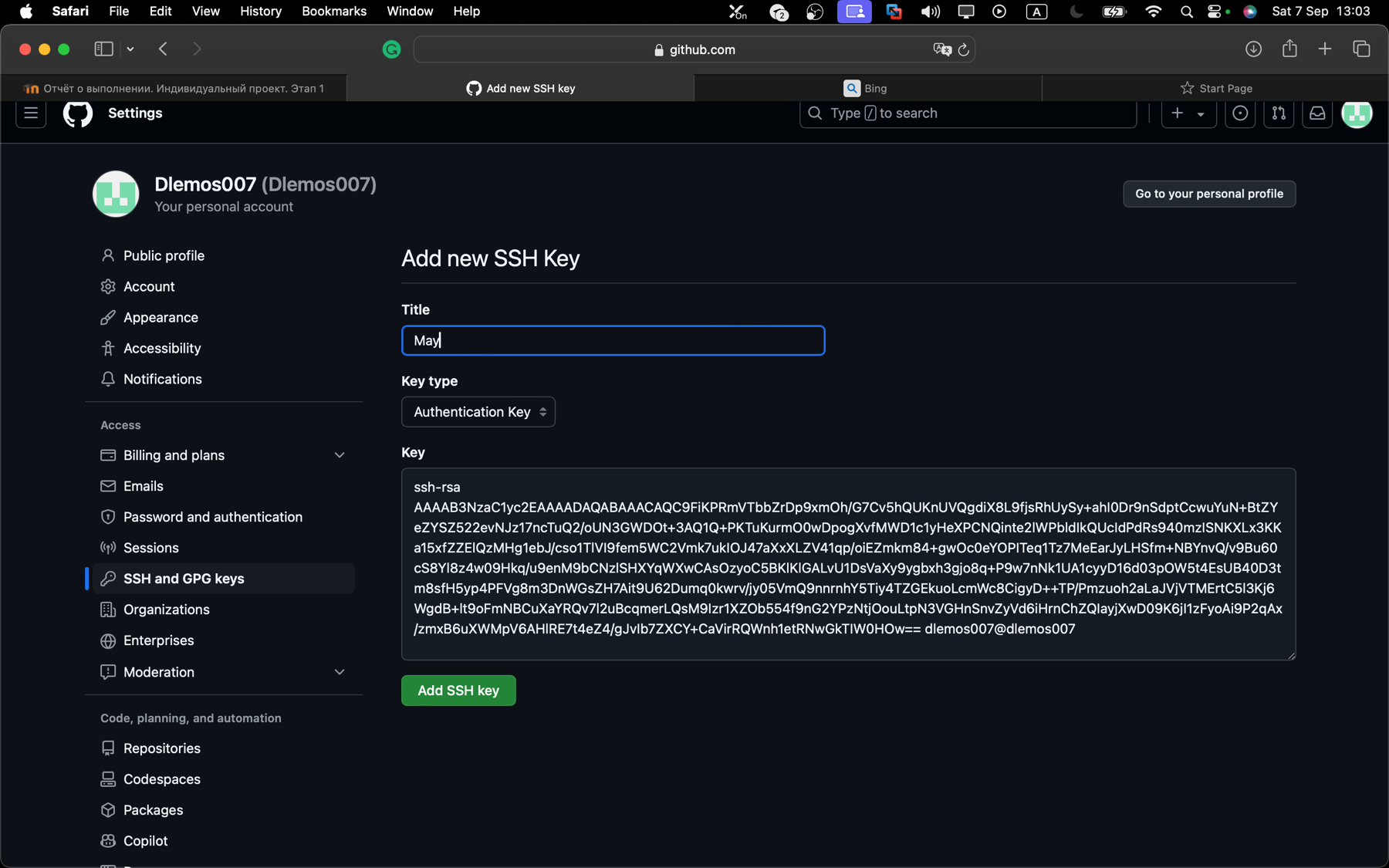
Копируем наш ключ в буфер обмена (рис. 3.12)



## Рис. 3.12: Копирование ключа

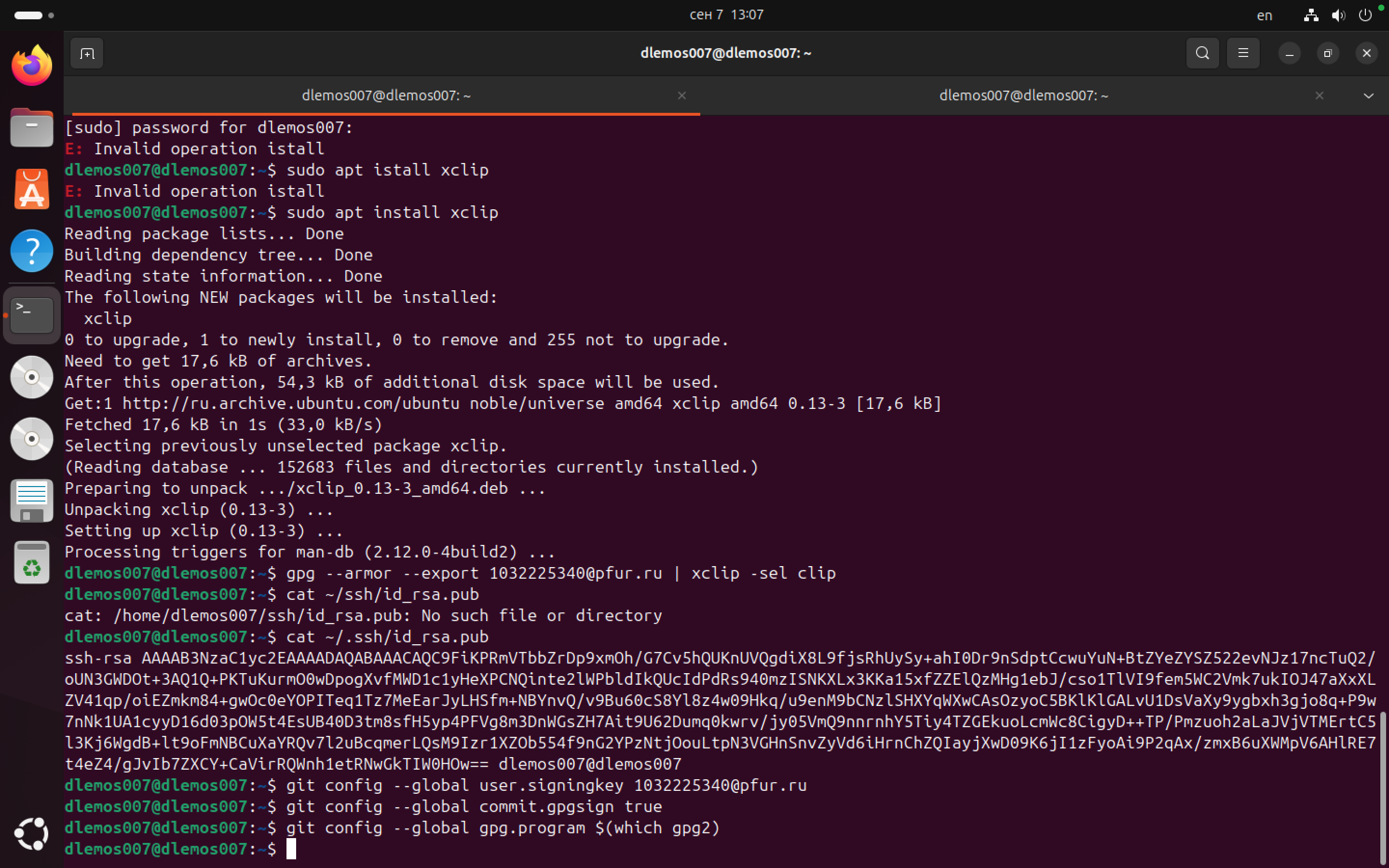
Вставляем этот ключ на гитхаб, и задаём ему имя. Я выбрал имя Sway (рис.

3.13)



## Рис. 3.13: Вставка ключа в GitHub

Теперь производим настройку автоматических подписей (рис. 3.14)



## Рис. 3.14: Настройка автоматических подписей коммитов git

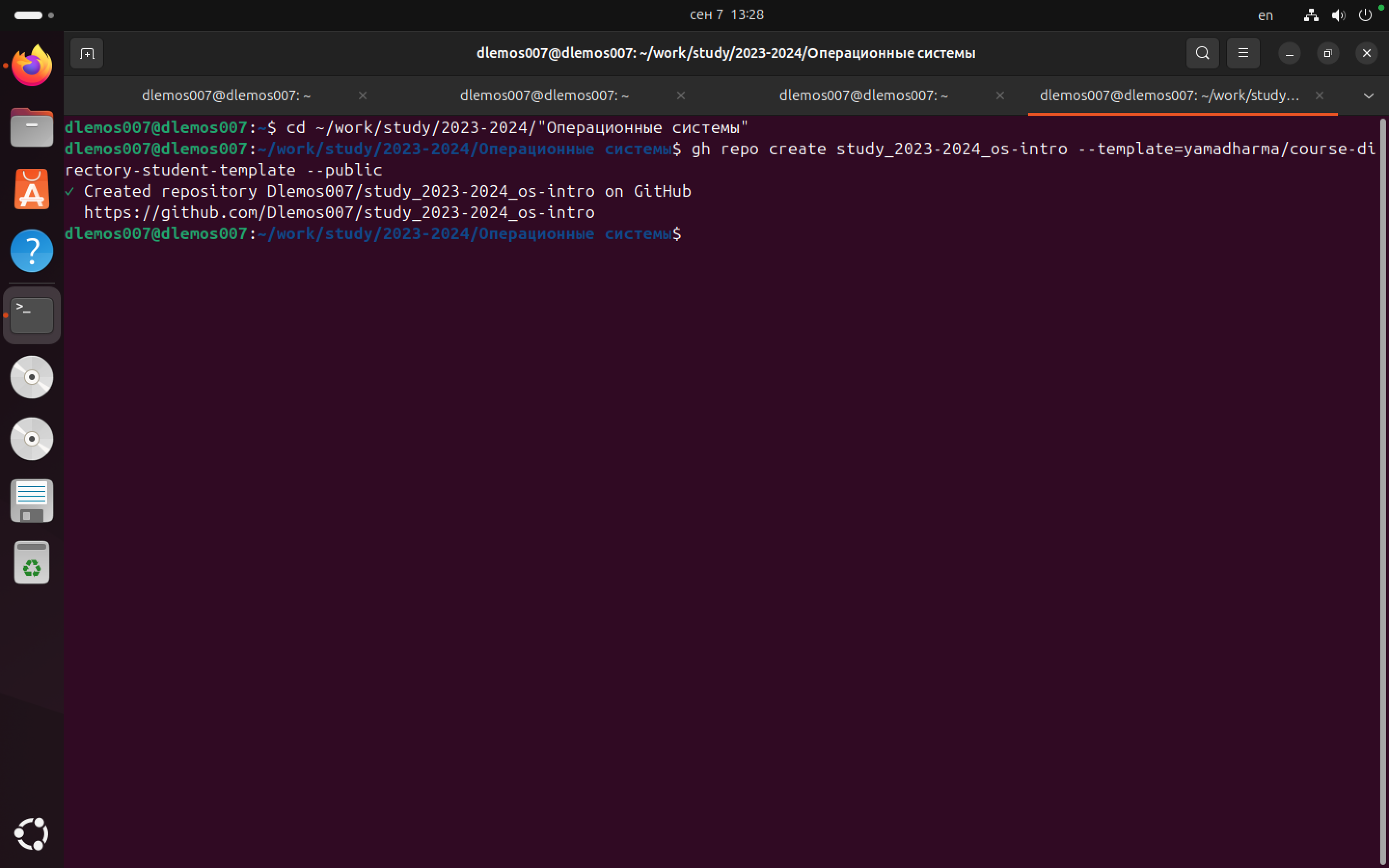
После, нам нужно авторизоваться в github с помощью gh. Мы выбираем сайт для авторизации (GitHub.com), после выбираем предпочитаемый протокол (SSH), публичный SSH ключ (id\_rsa.pub), и имя для ключа (Sway). В качестве способа авторизации выбираем авторизацию через браузер (рис. 3.15)

A screenshot of a computer

Description automatically generated

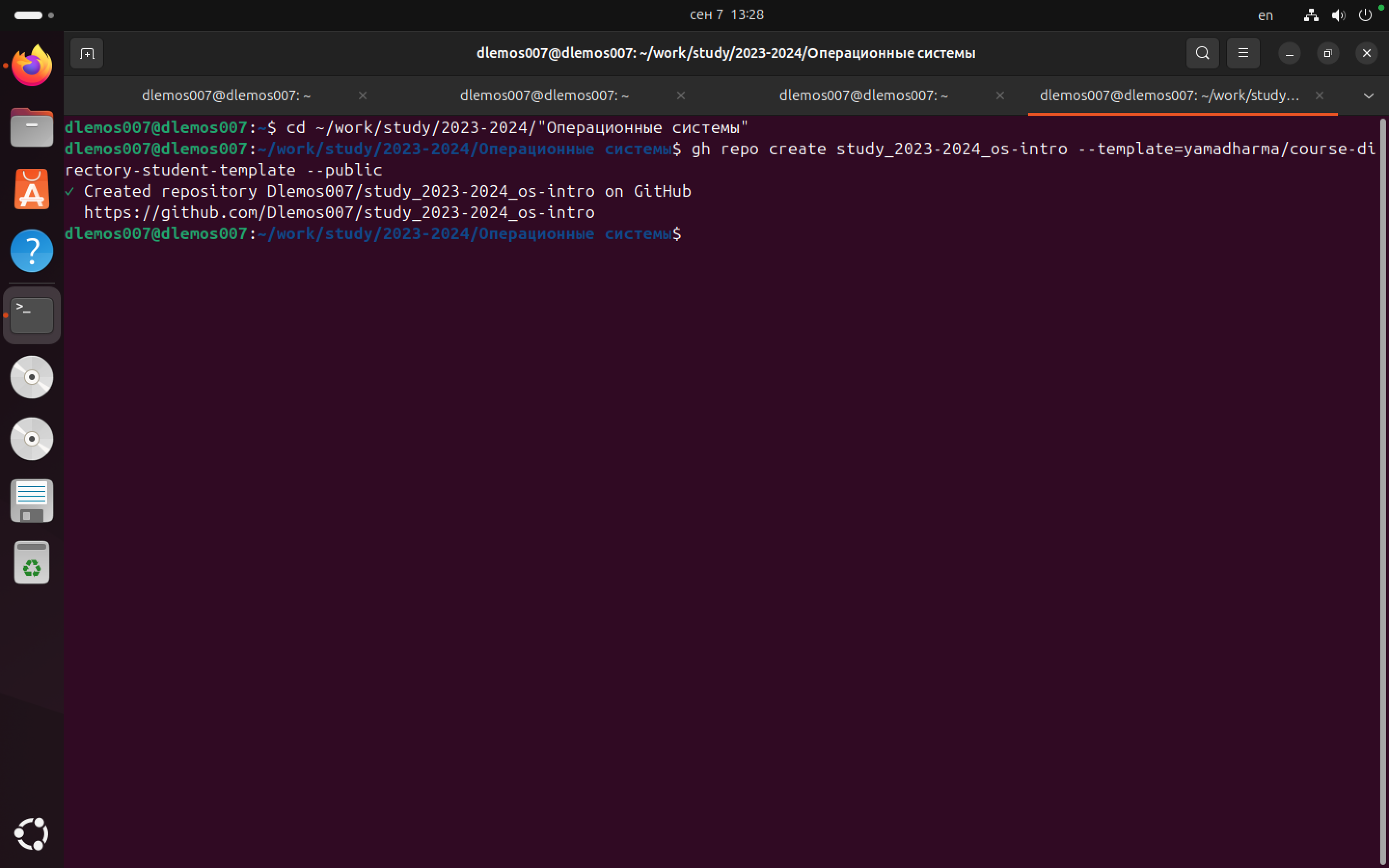
## Рис. 3.15: Авторизация в gh

Теперь создаём рабочую директорию курса и переходим в неё (рис. 3.16)

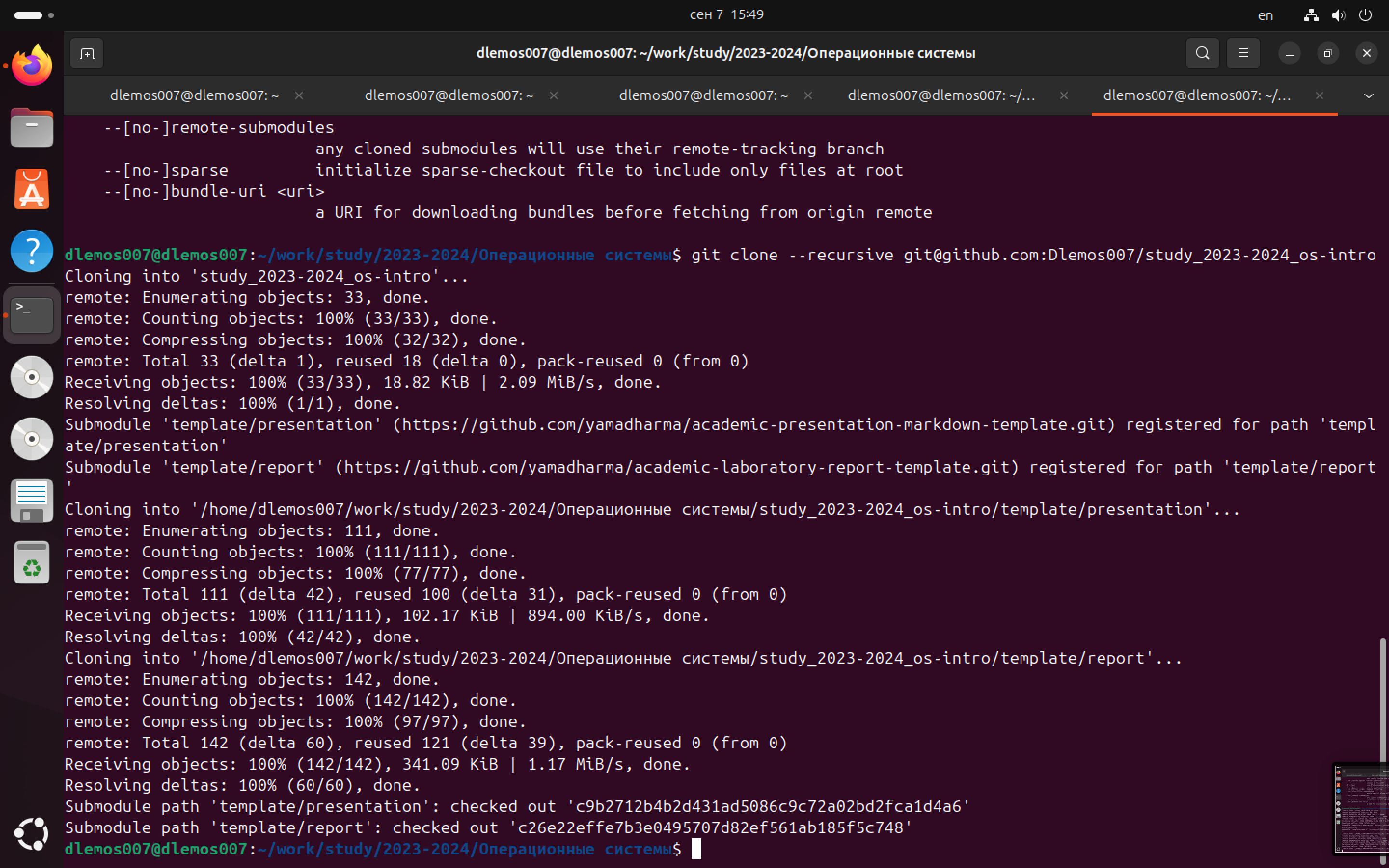


## Рис. 3.16: Создание рабочей директории и переход в неё

Далее, создаём репозиторий для лабораторных работ из шаблона (рис. 3.17)



## Рис. 3.17: Сознание репозитория курса



## Рис. 3.18: Клонирование репозитория

1. **Выводы**

Произведена установка Git, выполнена его базовая настройка, сгенерированы ключи для авторизации и подписи, а также создан репозиторий курса на основе предложенного шаблона.

1. **Ответы на контрольные вопросы**
   1. Системы контроля версий – это системы, в которых мы можем хранить свои проекты и выкладывать их обновления, контролируя релизы и каждые внесённые изменения. Эти системы нужны для работы над проектами, чтобы иметь возможность контролировать версии проектов и в случае командной работы контролировать изменения, внесённые всеми участниками. Также, VCS позволяют откатываться на более ранние версии
   2. Хранилище – репозиторий, в нём хранятся все файлы проекта и все его версии

commit – внесённые изменения в репозитории история – это история изменений файлов проекта

рабочая копия – копия, сделанная из версии репозитория, с которой непосредственно работает сам разработчик

* 1. Централизованные системы контроля версий имеют один центральный репозиторий, с которым работают все разработчики. Примером является CVS, который является уже устаревшей системой.

В децентрализованных системах же используется множество репозиториеводногопроектаукаждогоизразработчиков,приэтомрепозиторииможно объединять брать из каждого только то, что нужно. Примером является знакомый нам Git

* 1. Создаётся репозиторий, и разрабатывается проект. При внесении изменений файлы отправляются на сервер
  2. Разработчик клонирует репозиторий к себе на компьютер, и после внесения изменений выгружает их на сервер в качестве отдельной версии. После этого разработчики с более высокими правами могут, например, объединить его версию с текущей
  3. Хранение файлов проекта, а также обеспечение командной работы, и контроль за версиями проекта
  4. git clone – клонирует проект с сервера на компьютер git add – добавляет папку для выгрузки на сервер git commit – фиксирует изменения репозитория git push – выгружает изменения на сервер git pull – получить изменения с сервера git rm – удалить файл

git status – получить статус репозитория

* 1. С локальным: git commit -am “added files” – создаёт коммит С удалённым: git push – загрузить данные на удалённый сервер
  2. Ветки – это несколько независимых копий проекта, в каждой из которых ведётся разработка какой-то конкретной функции, при этом ветки существуютпараллельно.Онинужны,когданужнопараллельновестиразработку нескольких функций, а в конце их можно объединить в одну
  3. Игнорироватьфайлыможно,вносяихвфайл.gitignore.Игнорироватьфайлы нужно, когда их не нужно добавлять в репозиторий. Например, это могут быть файлы виртуального окружения (venv)

**Список литературы**

1. Kulyabov. [Лабораторная работа № 2. Первоначальна настройка git](https://esystem.rudn.ru/mod/page/view.php?id=1098933). RUDN.