Лабараторная работа 2

Первоначальна настройка git

Хрусталев Влад Николаевич

Содержание

# 1 Цель работы

Изучить идеологию и применение средств контроля версий.

Освоить умения по работе с git.

# 2 Выполнение лабораторной работы

Установим git и gh.(рис. fig. 1)

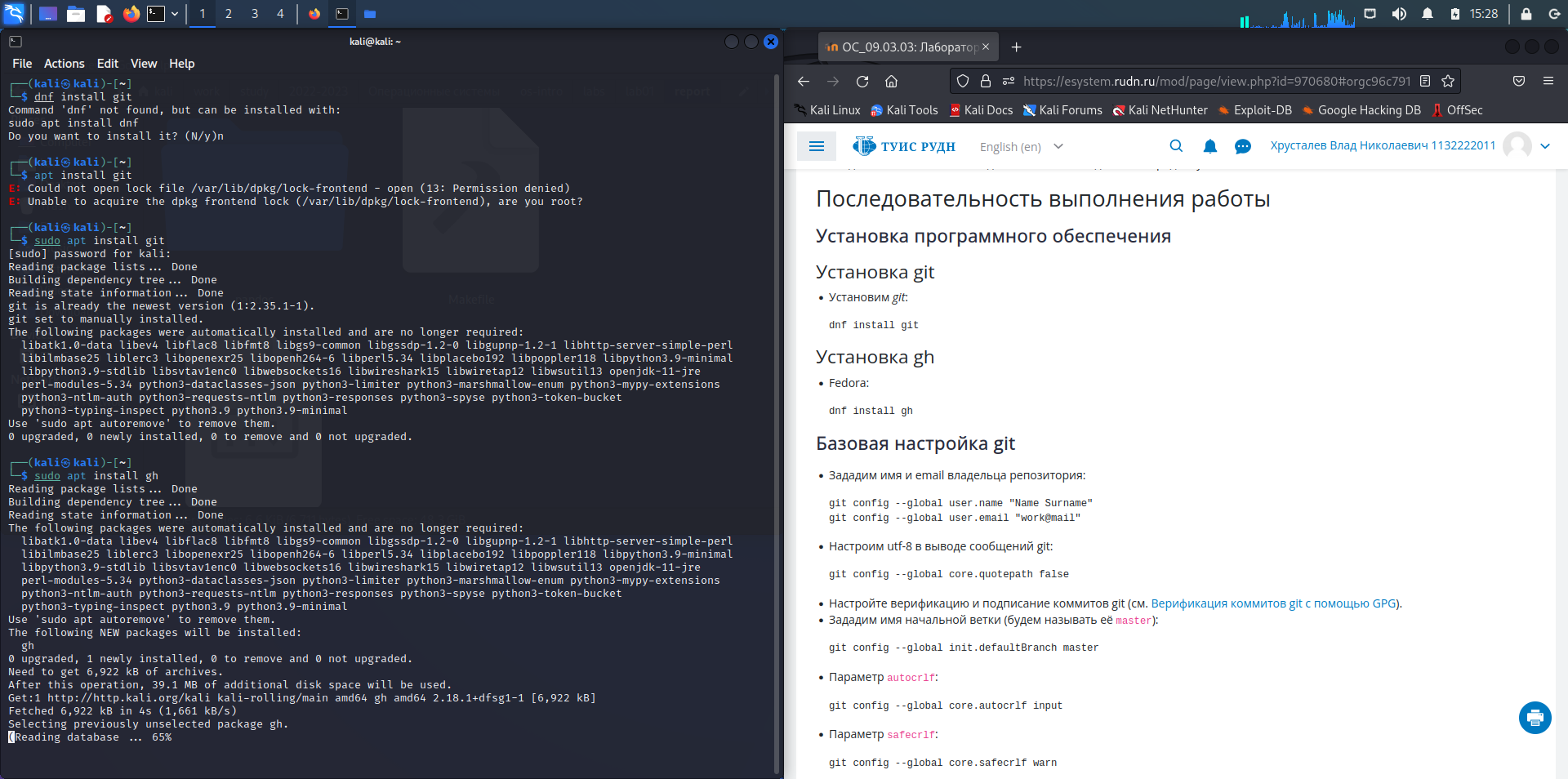


Рис. 1: Установка пакетов

Первичная настройка git по инструкции(рис. fig. 2)

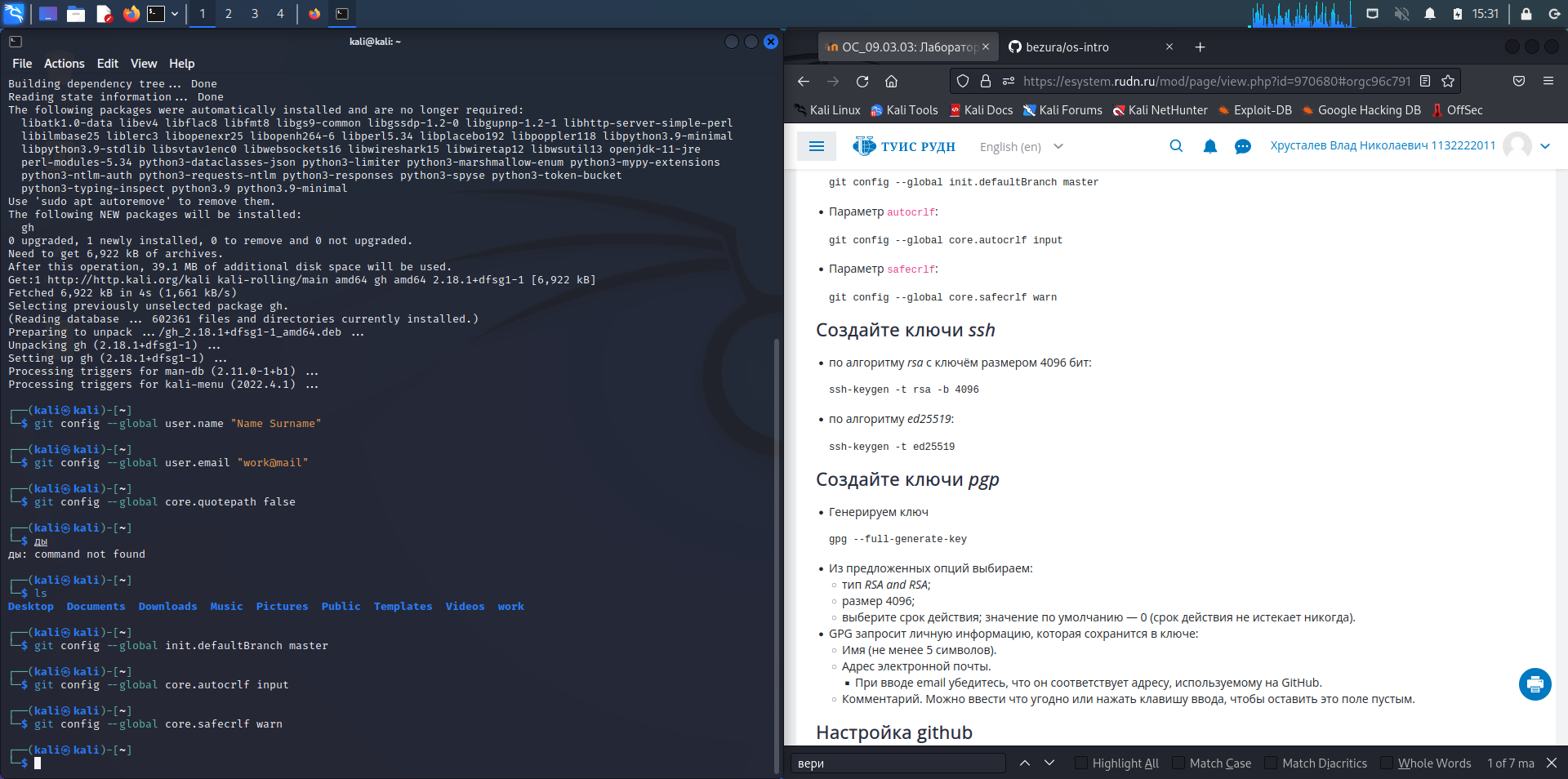


Рис. 2: Первичная настройка git

Создание двух ключей с алгоритмами rsa 4096 и ed25519(рис. fig. 3)

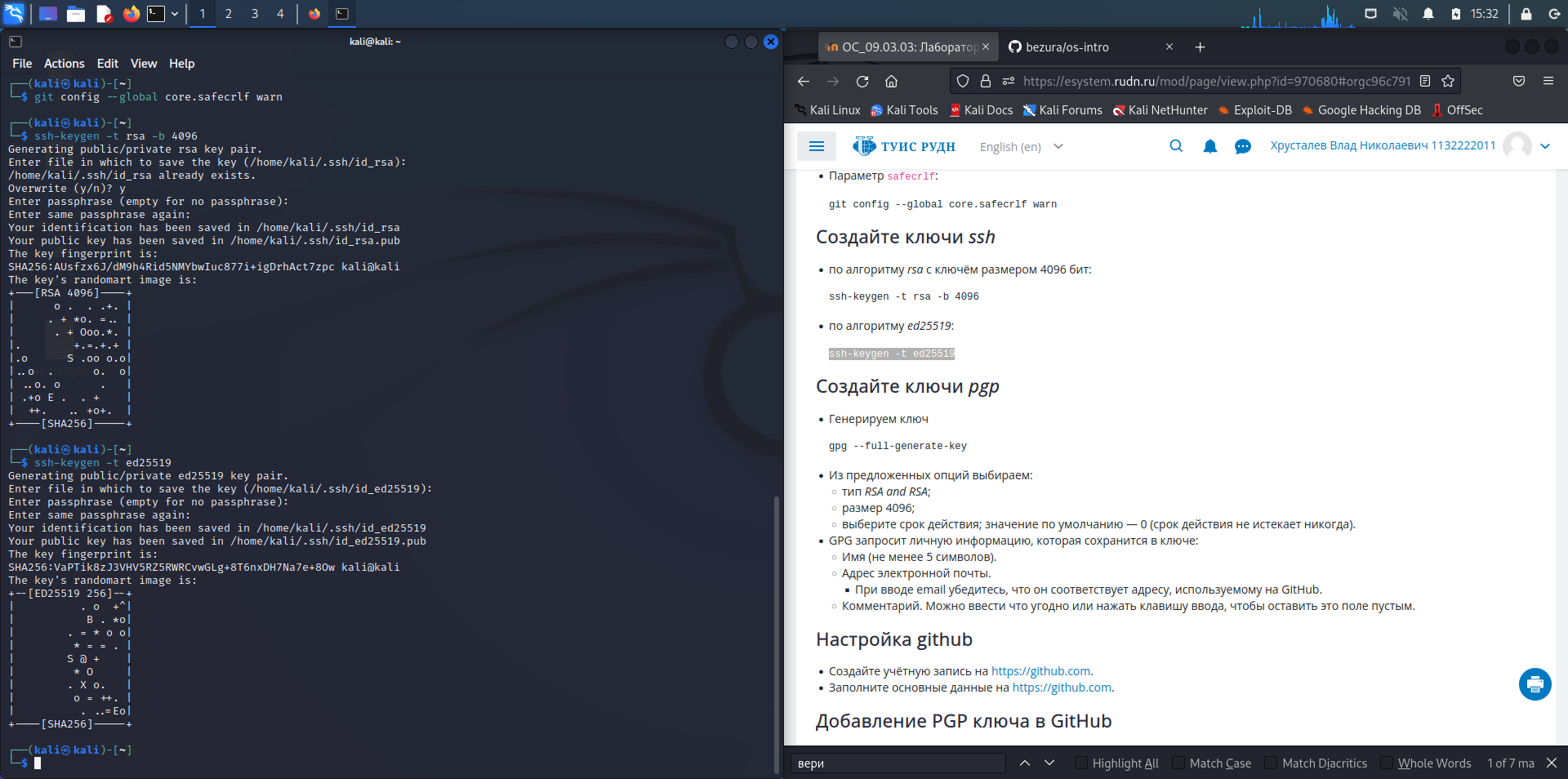


Рис. 3: Cоздание SSH ключей

Создание PGP ключа(рис. fig. 4)

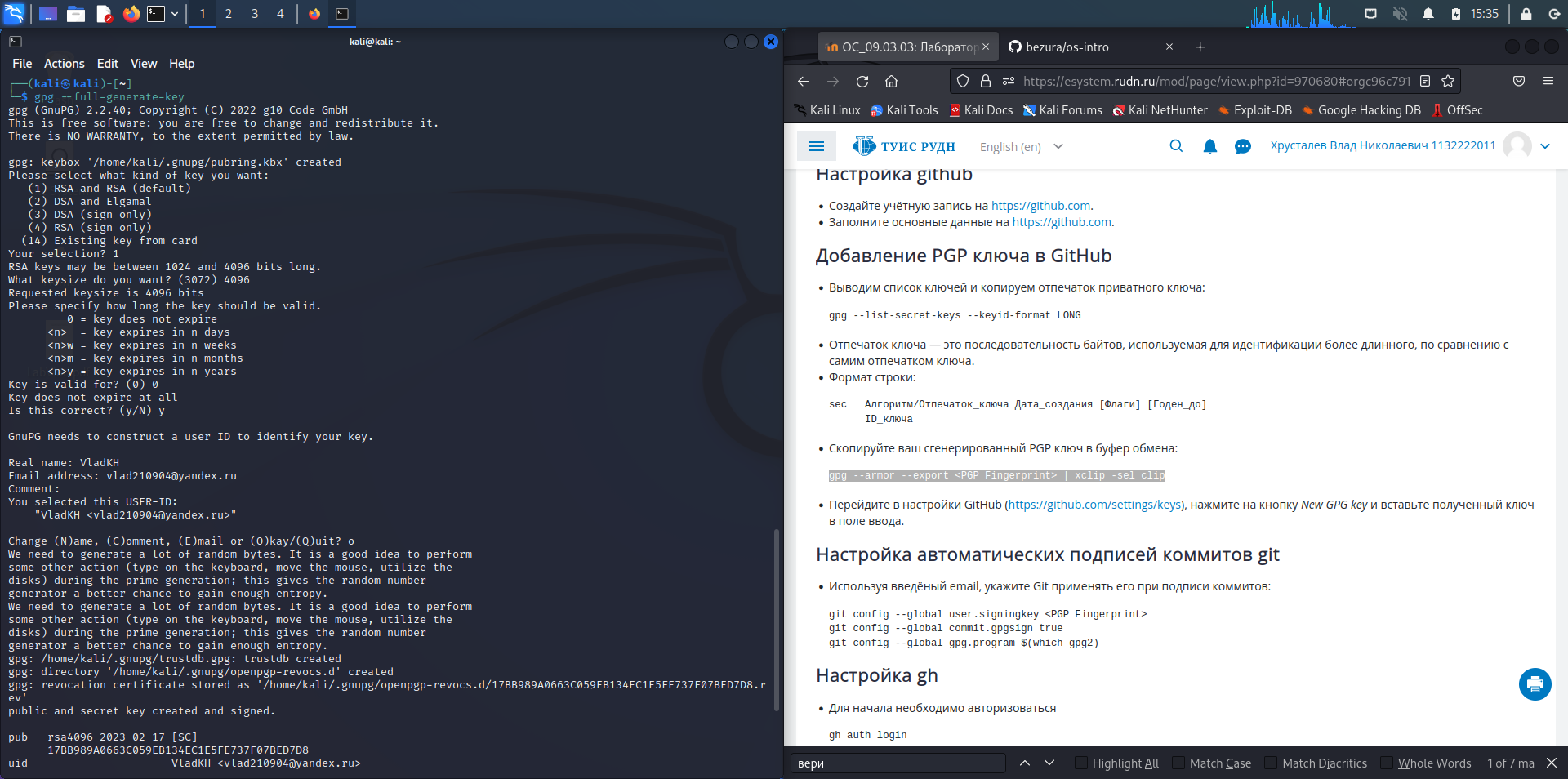


Рис. 4: Cоздание PGP ключей

Теперь скопируем готовый ключ и добавим на сайте github в GPG keys.(рис. fig. 5)

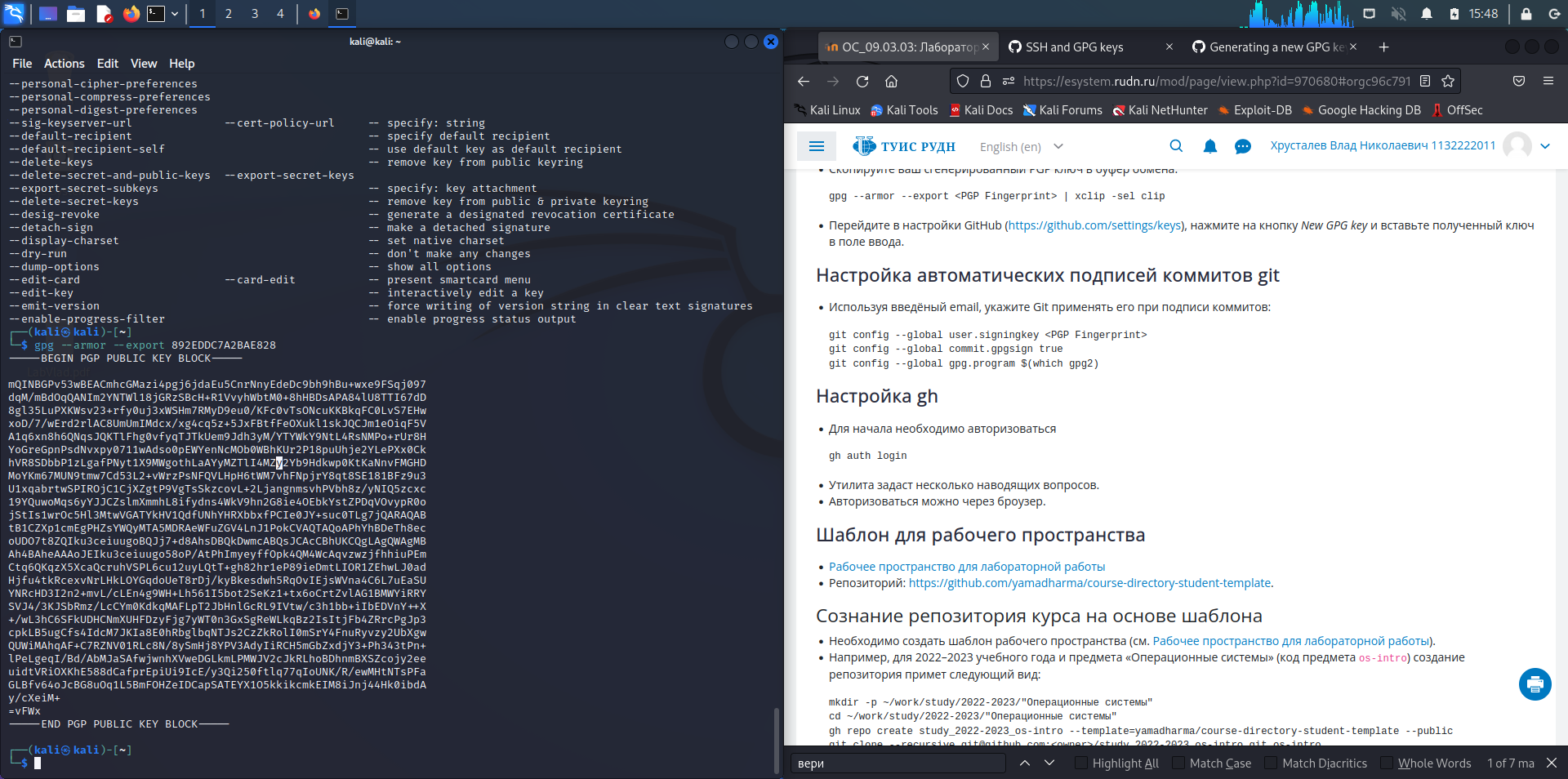


Рис. 5: Использование PGP ключей

Завершим настройку git.(рис. fig. 6)

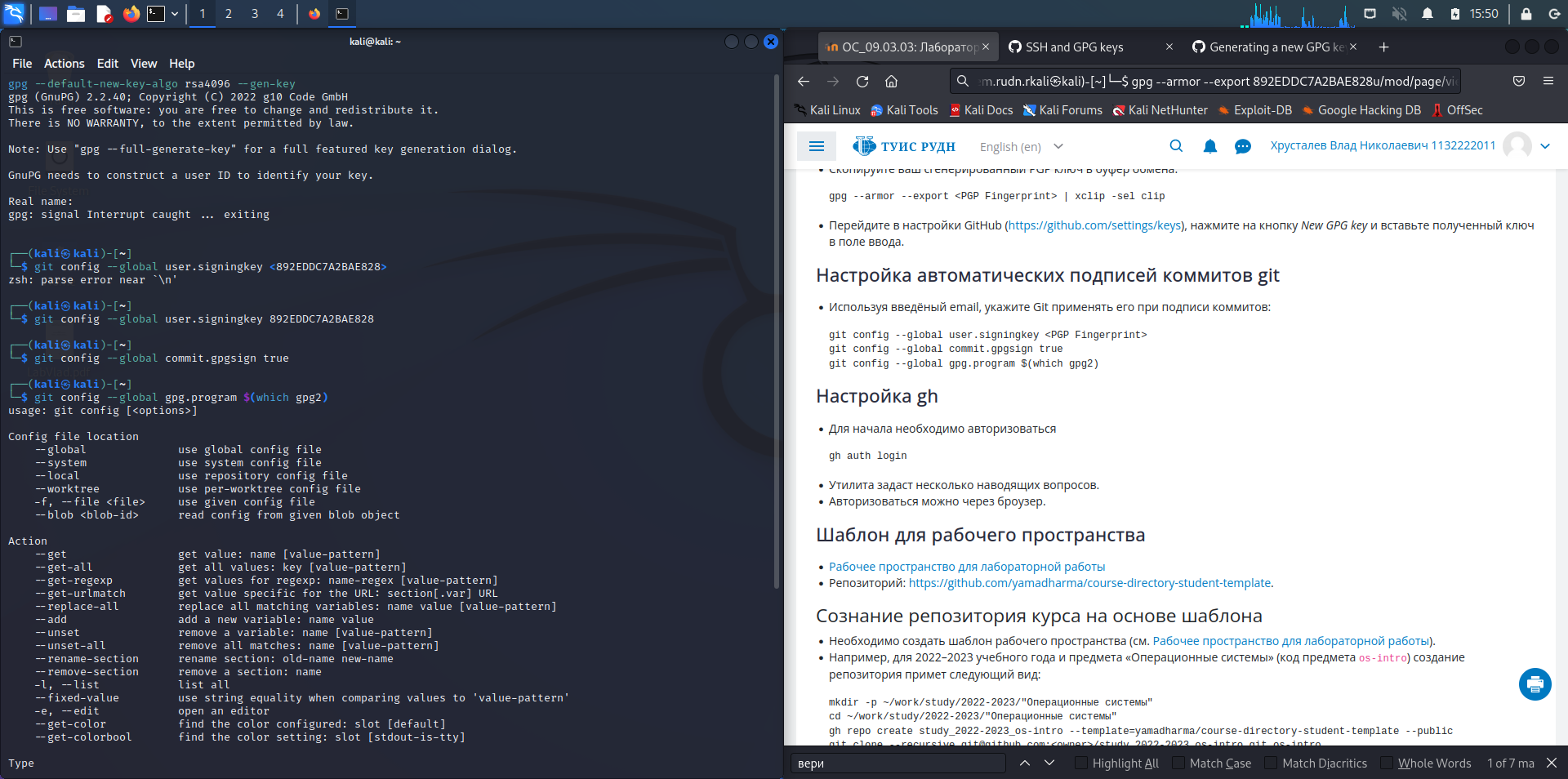


Рис. 6: Окончательная настройка git

Теперь открою репозиторий-шаблон и скопирую к себе.(рис. fig. 7).

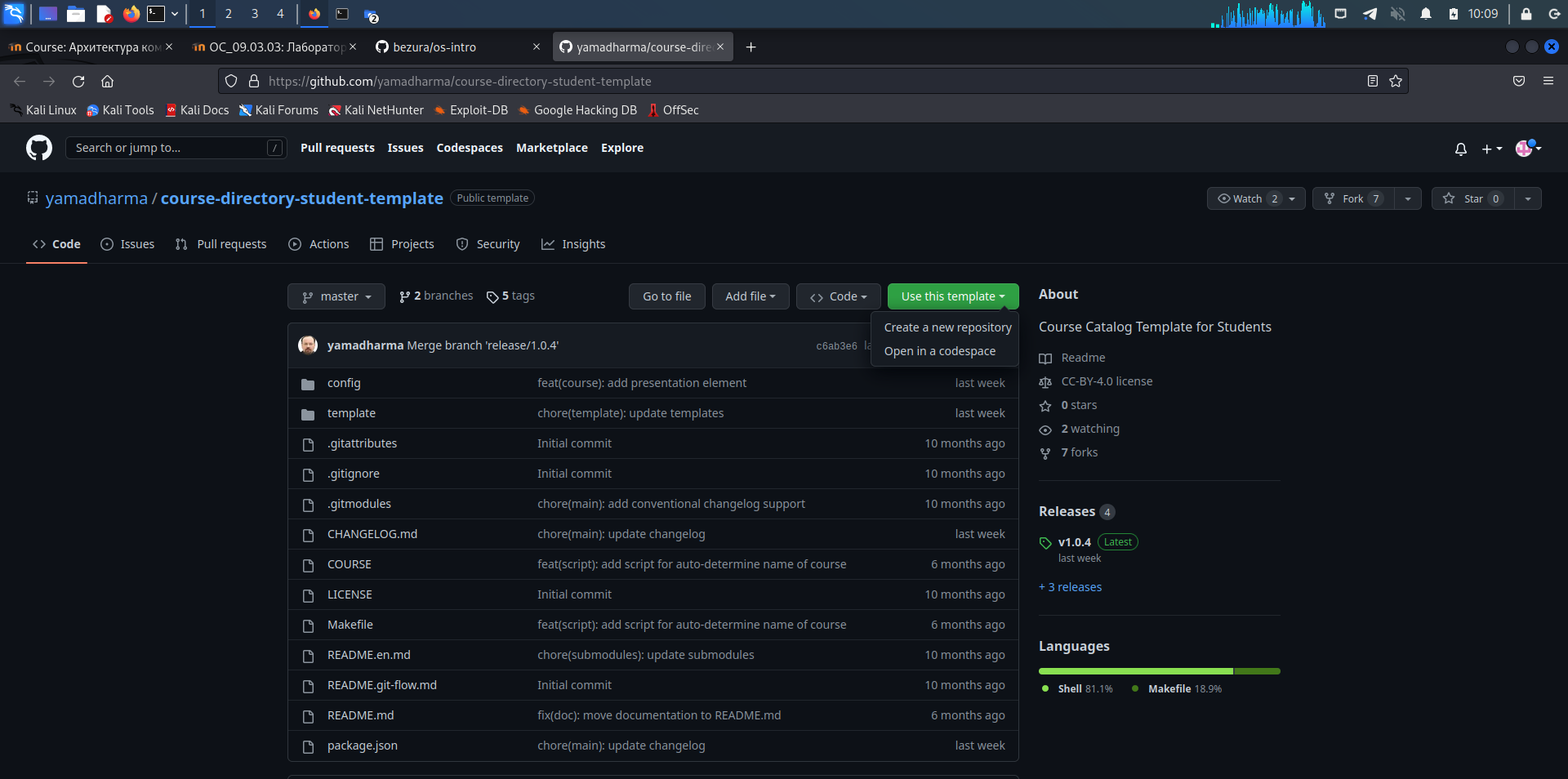


Рис. 7: копирование репозитория

Создадим нужную иерархию и скопируем репозиторий(рис. fig. 8).

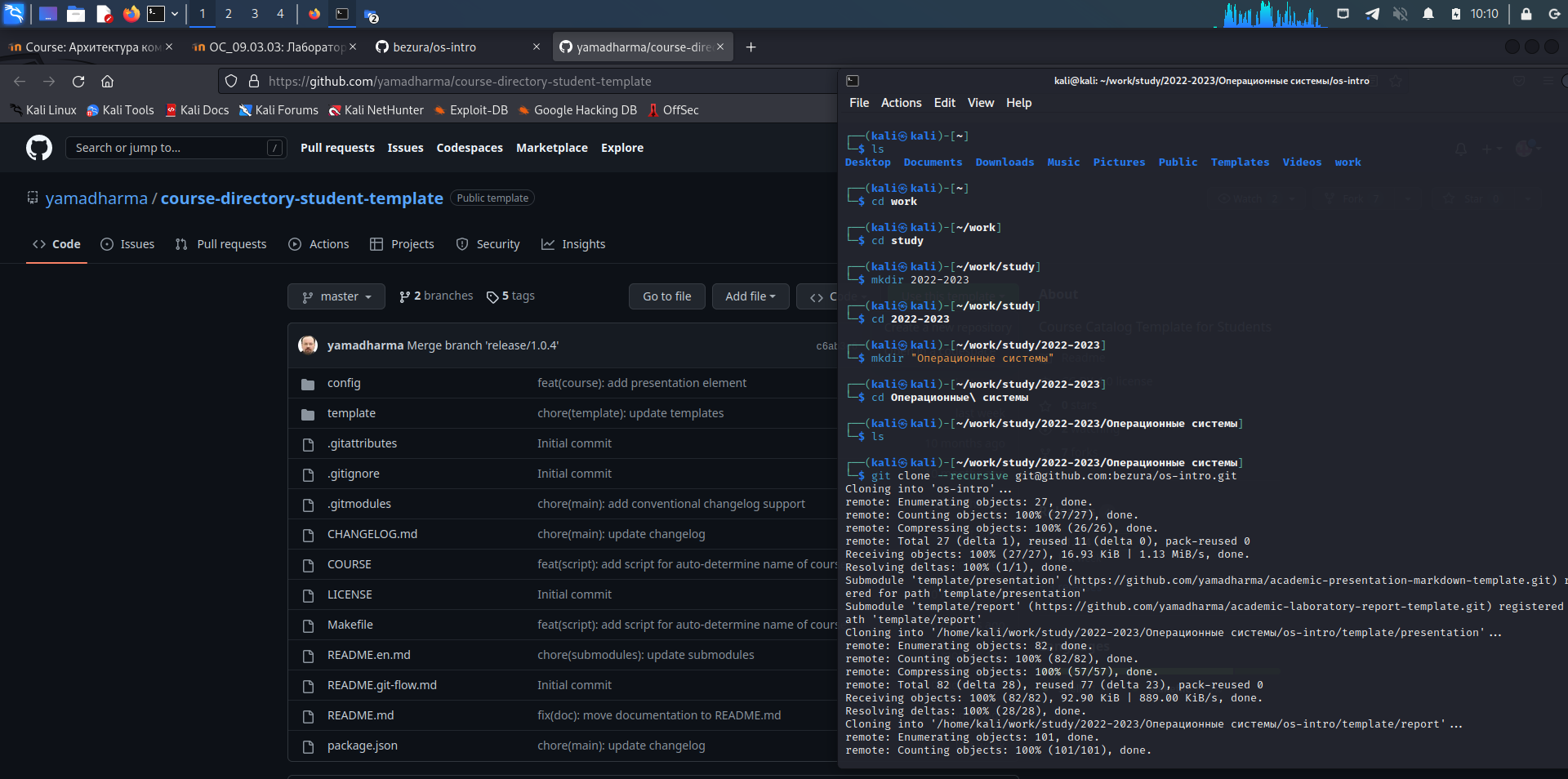


Рис. 8: Организация рабочего пространства

Удалим ненужные файлы и запустим файл make для настройки репозитория(рис. fig. 9).

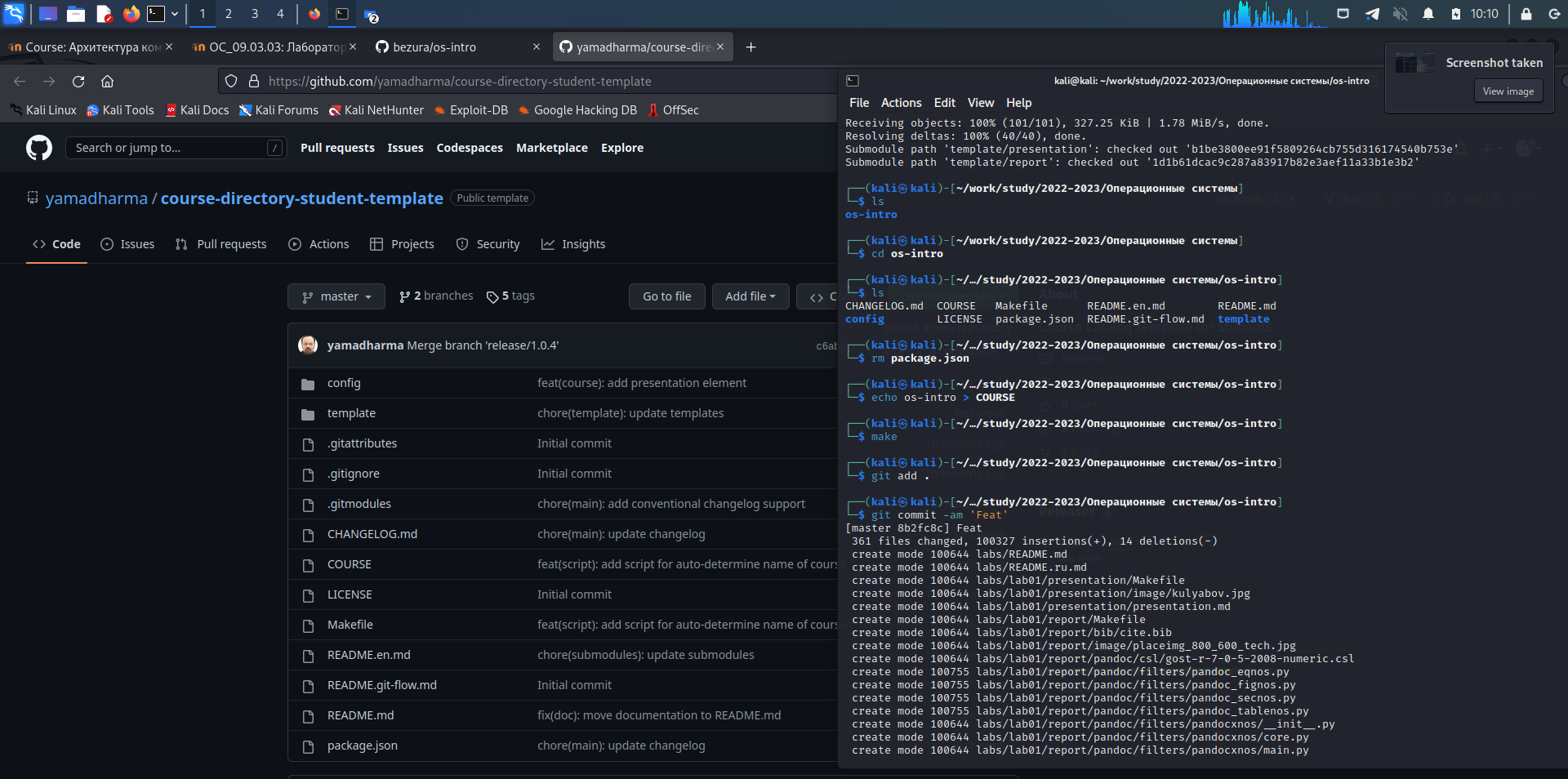


Рис. 9: Настройка рабочего пространства(1)

Далее загрузим обратно в гитхаб(рис. fig. 10)

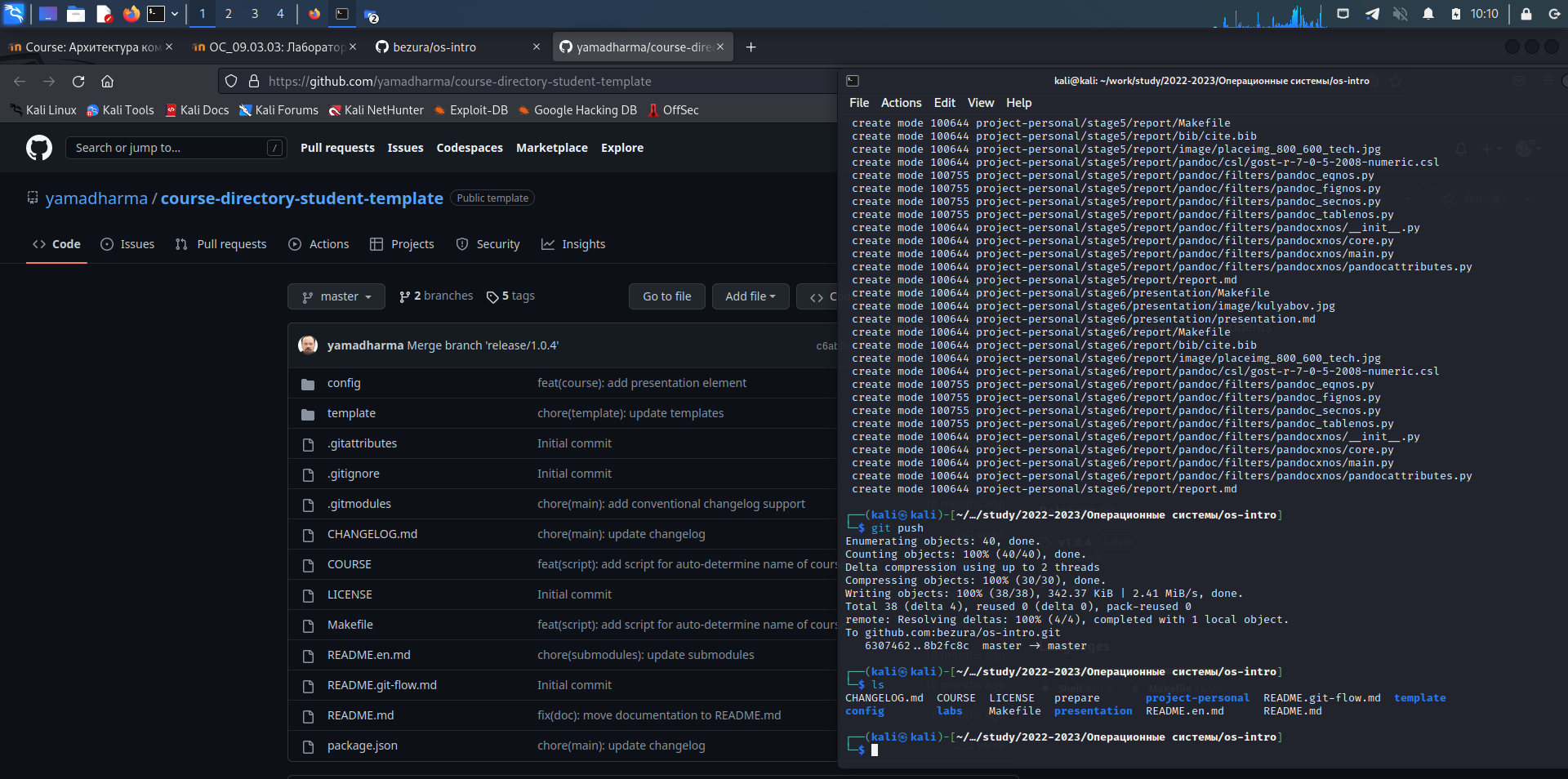


Рис. 10: Настройка рабочего пространства(2)

#Контрольные вопросы

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

Место где храняться какие-либо данные и к которому можно получить доступ, менять(сохраняя историю изменений). Используется для написания больших программ командой разрабочиков и в многих других задачах

2)Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

Хранилище - то место что является эталоном

Commit - коментарий к изменным файлам

История - кто когда и что редактировал

Рабочая копия - копия на ПК, в которой производятся изменения, а далее обратно выгружается на сервер

3)Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

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.

4)Опишите действия с VCS при единоличной работе с хранилищем.

Один человек загрузил и выгрузил обратно. Хранилище у него на пк(вероятнее всего)

5)Опишите порядок работы с общим хранилищем VCS.

Хоть сколько человек загрузили и обновии данные. Сохраняется кто что изменял.

6)Каковы основные задачи, решаемые инструментальным средством git?

История изменнений, лёгкость правок и отката.

7)Назовите и дайте краткую характеристику командам git.

Перечислим наиболее часто используемые команды 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.  
  
Локальный репозиторий - фрилансер один пишет код.  
  
Удалённый репозиторий - например на гитхабе репозиторий большого проекта команды разработчиков.  
  
9)Что такое и зачем могут быть нужны ветви (branches)?  
История изменений, правок - даёёт удобство. Или же можно посмотреть вариацию изменений одного куска кода( текста) и т.д.  
  
10)Как и зачем можно игнорировать некоторые файлы при commit?  
  
Не изменные файлы остаются такими же

# 3 Выводы

На данной лабороторной, я закрепил знания полученные в прошлом семестре по работе с git.