Лабораторная работа №3

Отчет

Бондарь Алексей Олегович

Содержание

# Цель работы

Научиться оформлять отчёты с помощью легковесного языка разметки Markdown

# Задание

–Сделайте отчёт по предыдущей лабораторной работе в формате Markdown.–В качестве отчёта просьба предоставить отчёты в 3 форматах:pdf,docxиmd(вархиве,поскольку он должен содержать скриншоты,Makefile ит.д.)

# Выполнение лабораторной работы

Создаем учетную запись на https://github.com.(рис. 1)

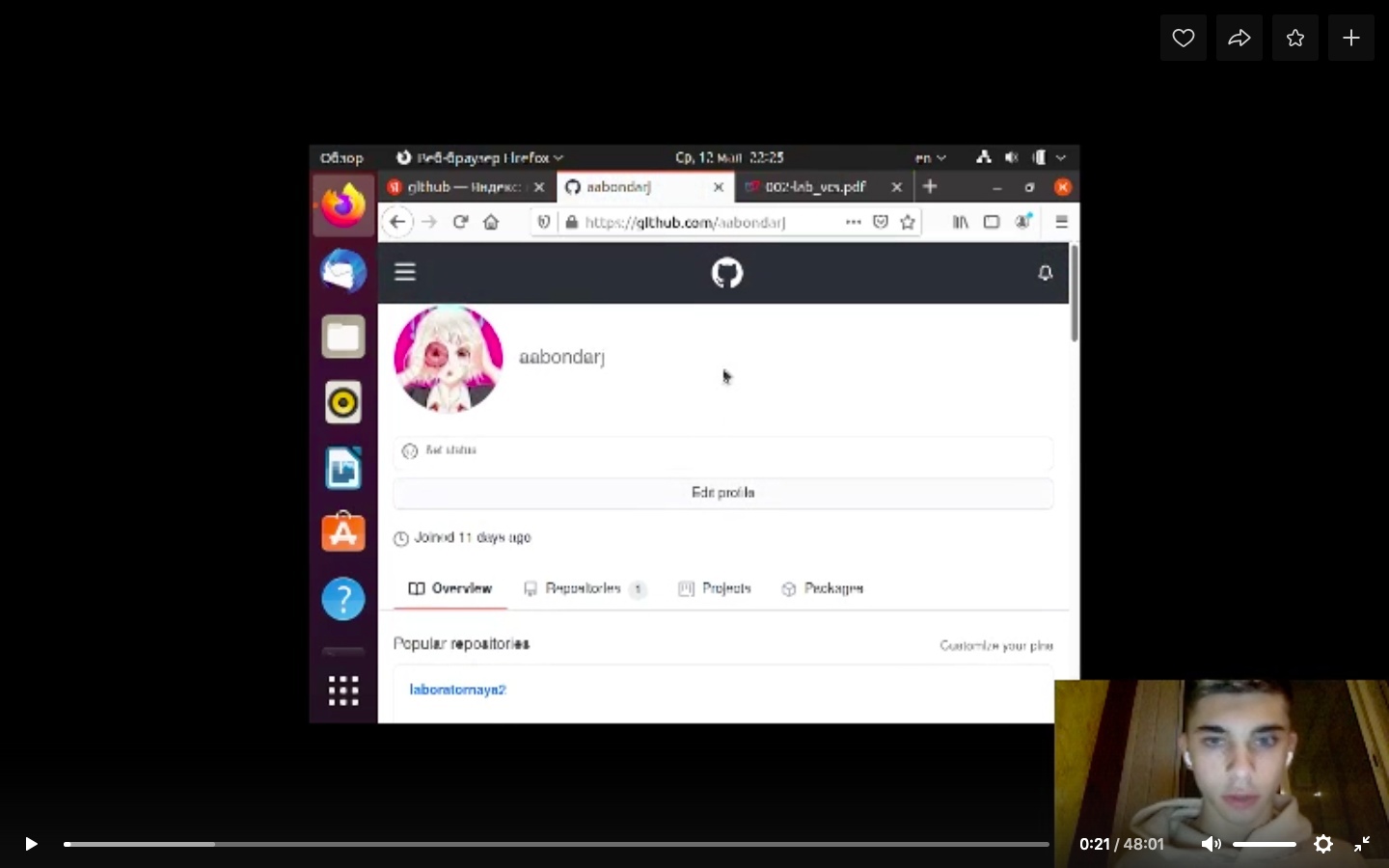


Figure 1: Учетная запись

Настраиваем систему контроля версий git. Синхранизируем учётную запись github с компьютером.(рис. 2)

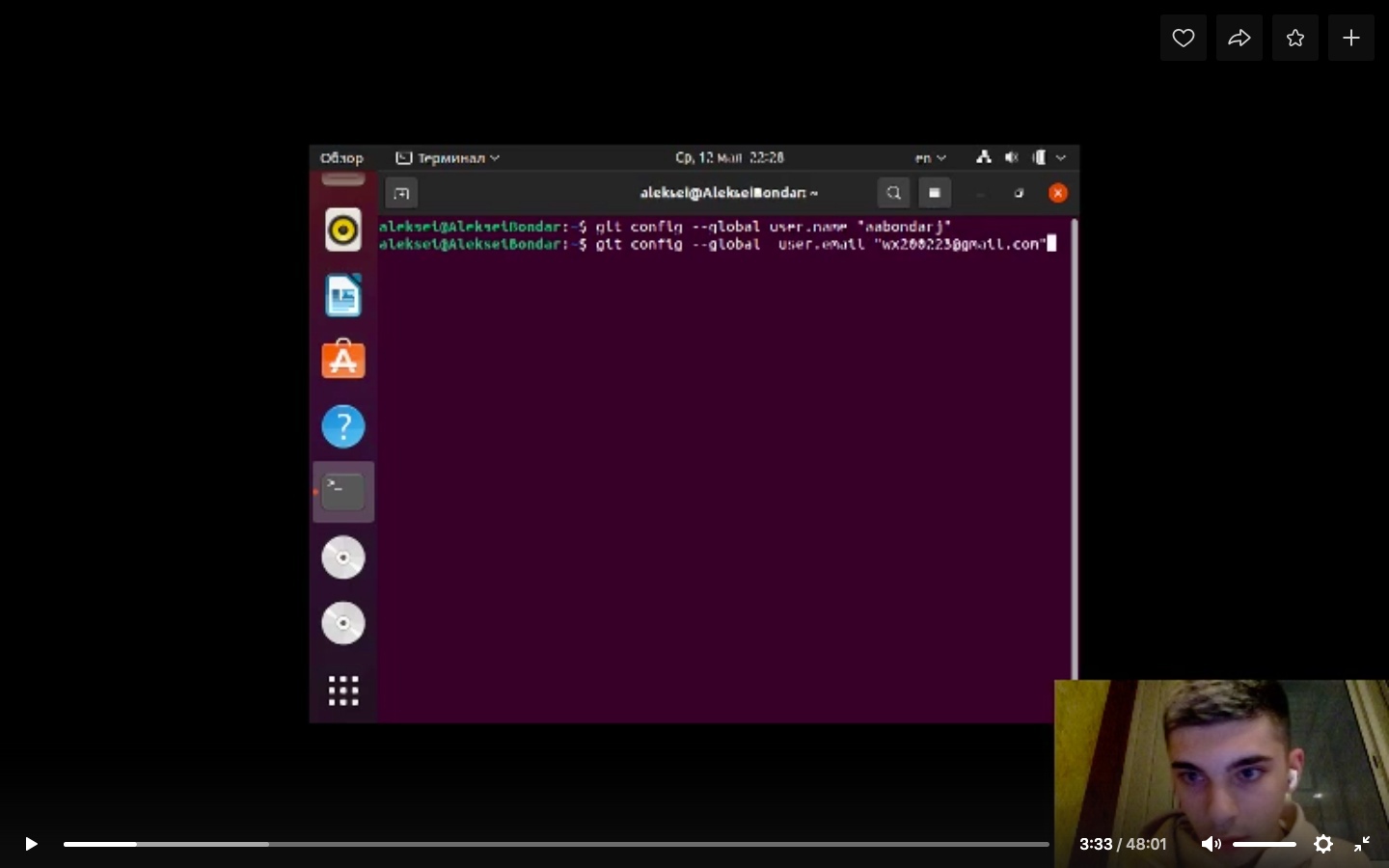


Figure 2: Система контроля git

После этого создаём новый ключ на github и привязываем его к копьютеру через консоль.(рис. 3)

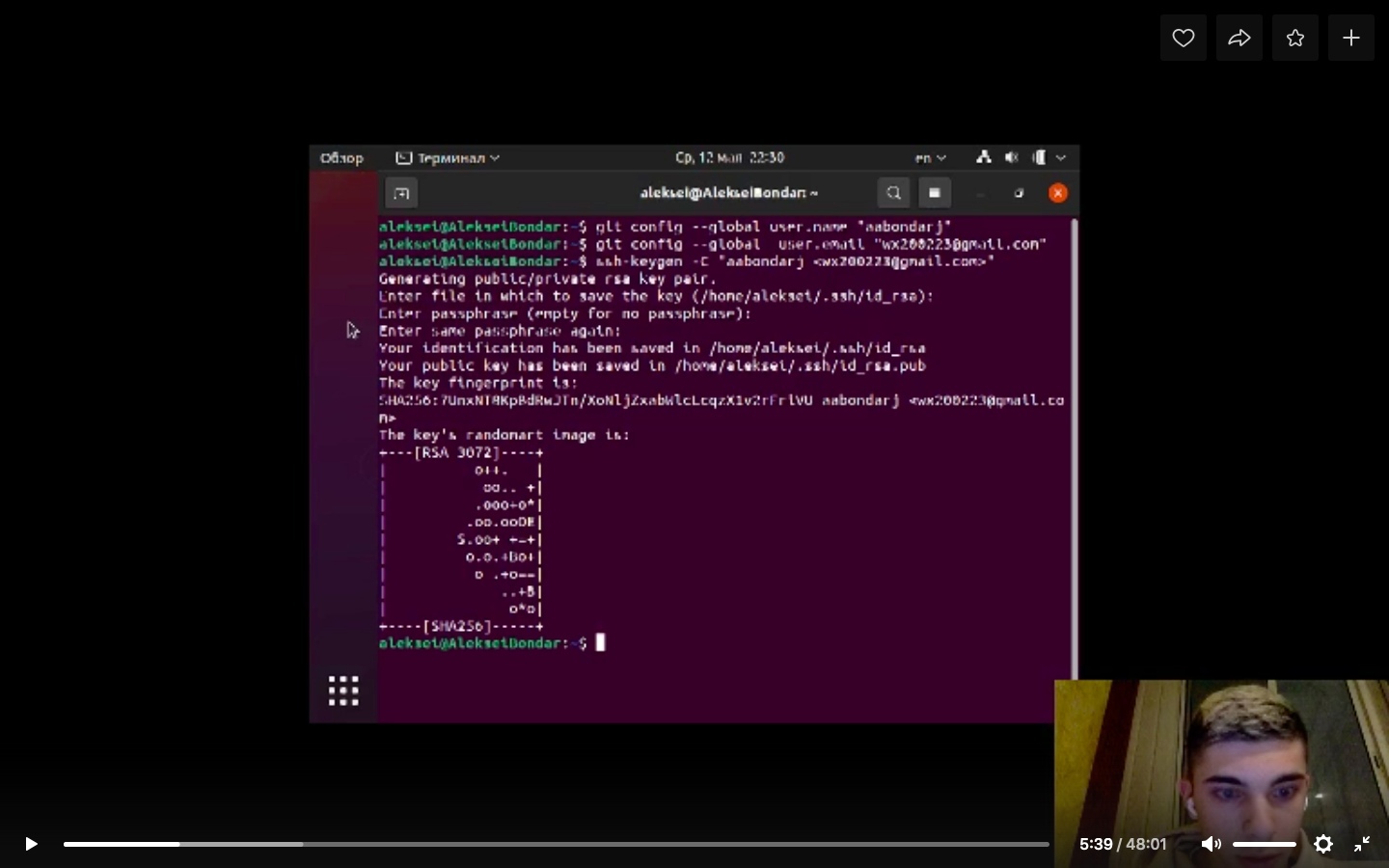


Figure 3: Новый ключ

Следующим шагом будет создание и подключение репозитория к github. В gethup заходим в «repository» и создаём новый репозиторий (имя «lab2», а заголовок для файла README). Копируем в консоль ссылку на репозиторий (для дальнейшей работы с файлами):.(рис. 4)

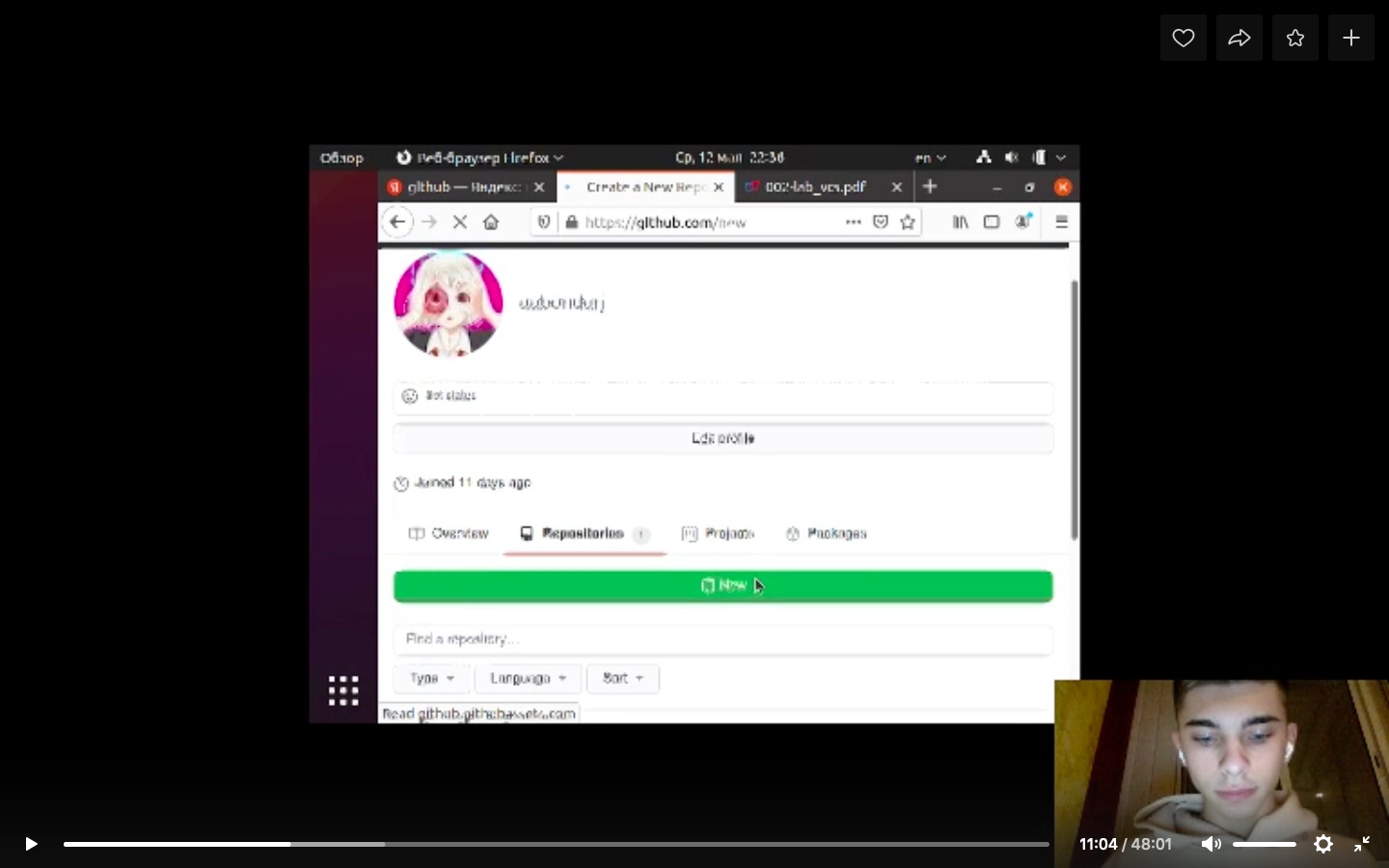


Figure 4: Создание репозитория

(рис. 5)

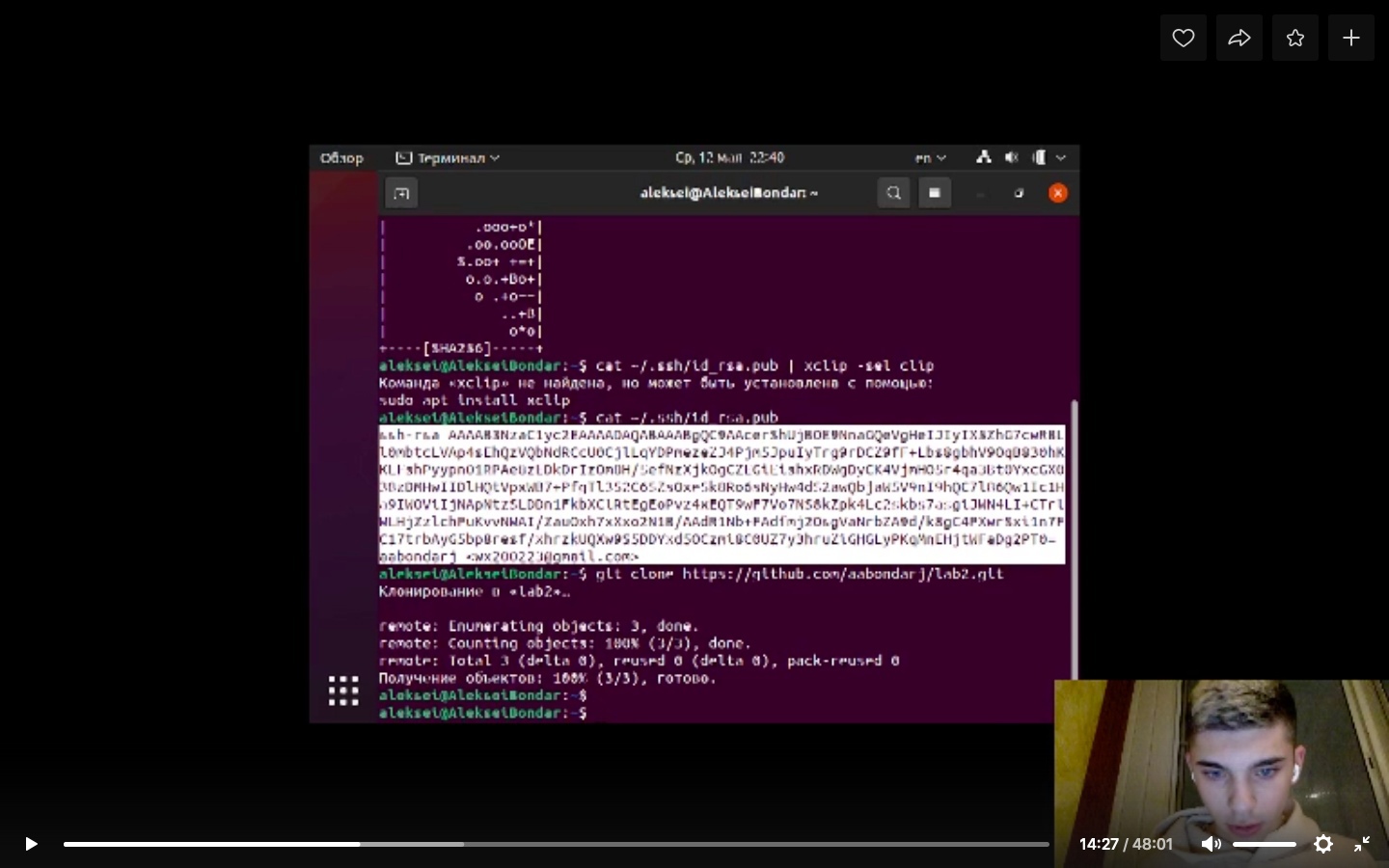


Figure 5: Копируем ссылку на репозиторий

В лабораторной работе описан логаритм создания структуры католога через консоль. Но легче будет создать репозиторий в githup и после этого работать с каталогом и папками через консоль (перед этим необходимо скопировать ссылку на репозиторий в консоль, в формате https или ssh). Перед тем, как создавать файлы, заходим в наш репозиорий.(рис. 6)

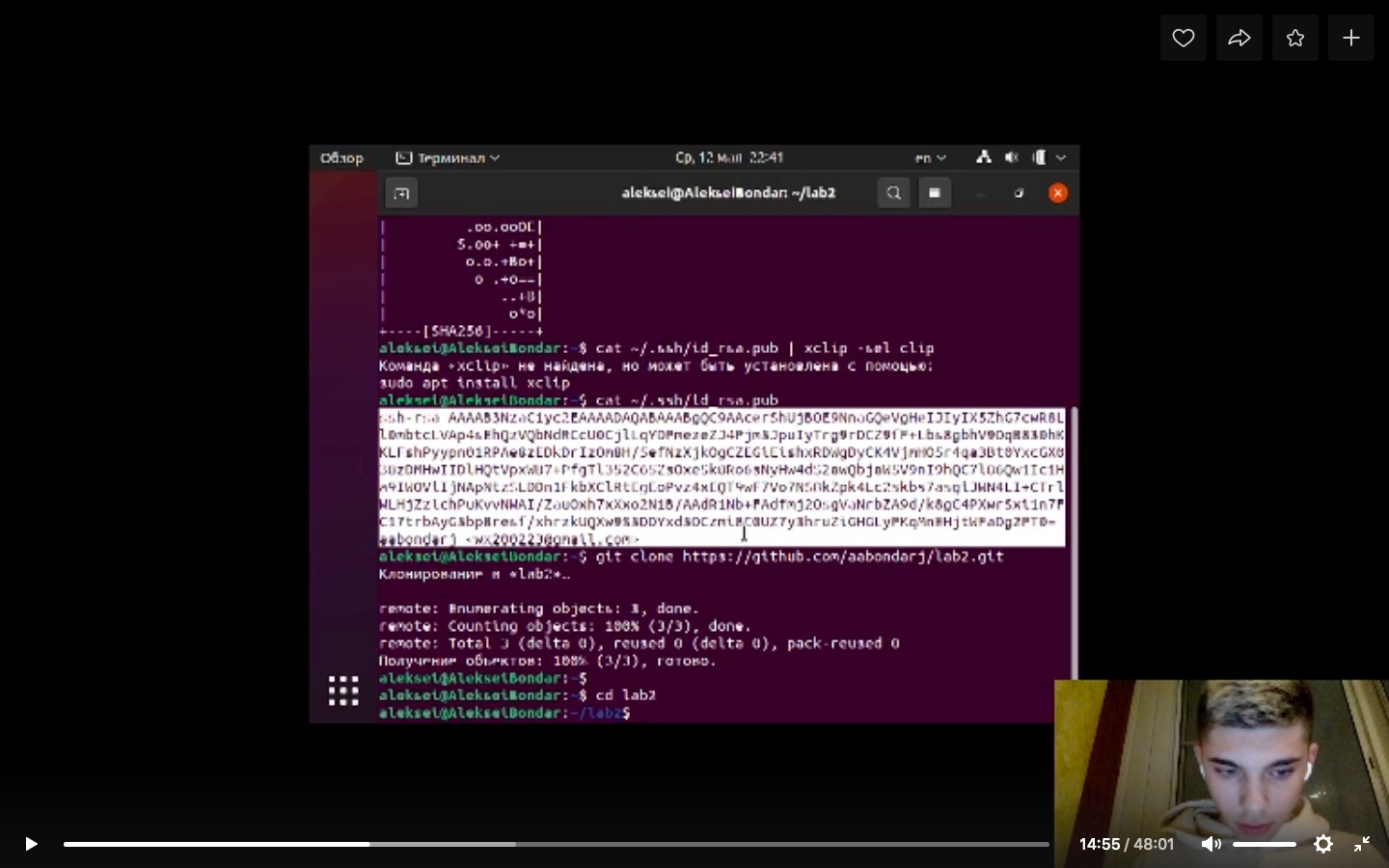


Figure 6: Репозиторий

После этого можем уже создавать наши файлы.(рис. 7)

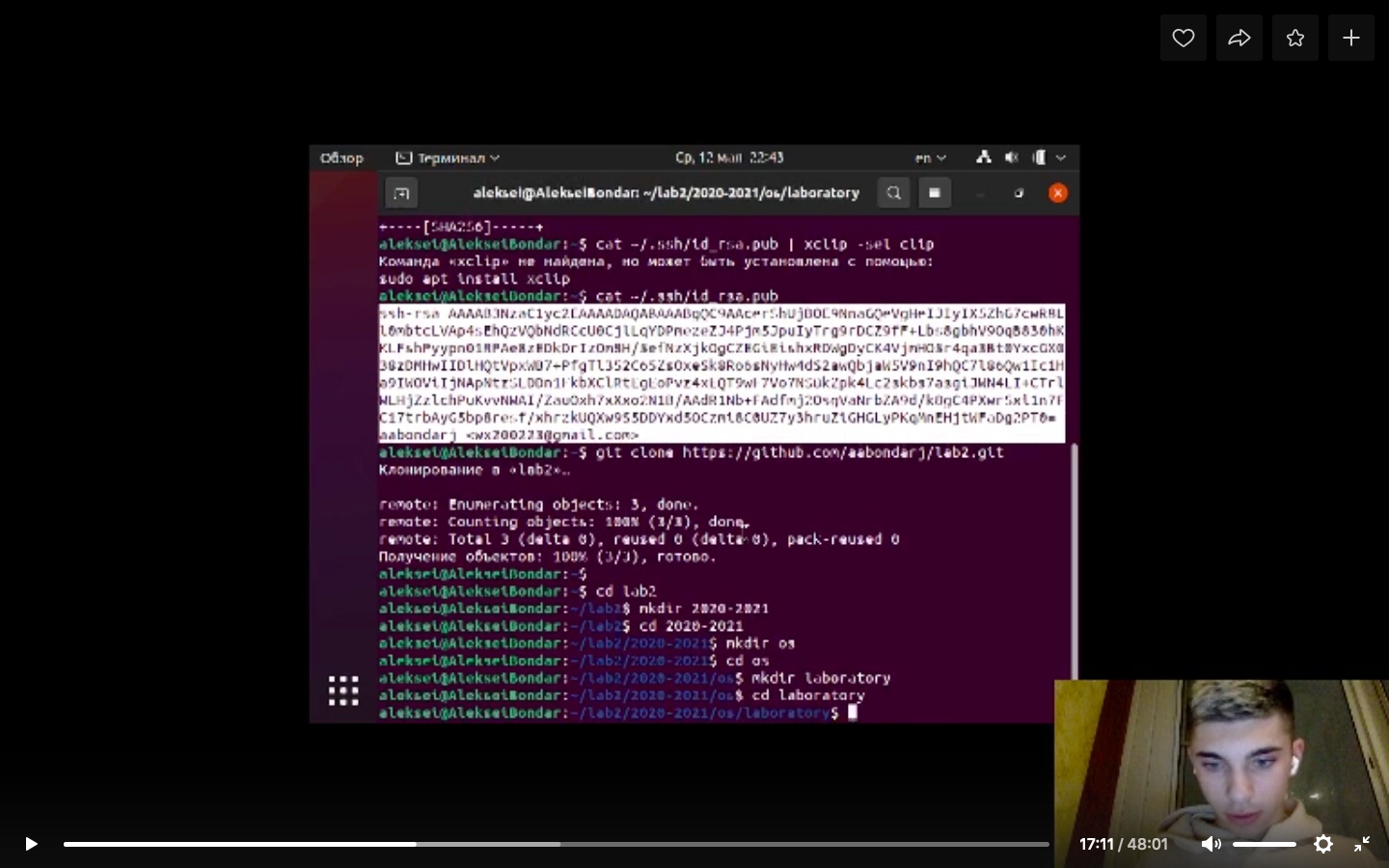


Figure 7: Создание файлов

Добавляем первый коммит и выкладываем на gethup. Для того, чтобы правильно разместить первый коммит, необходимо добавить команду git add ., после этого с помощью команды git commit -m “first commit” выкладываем коммит.(рис. 8)

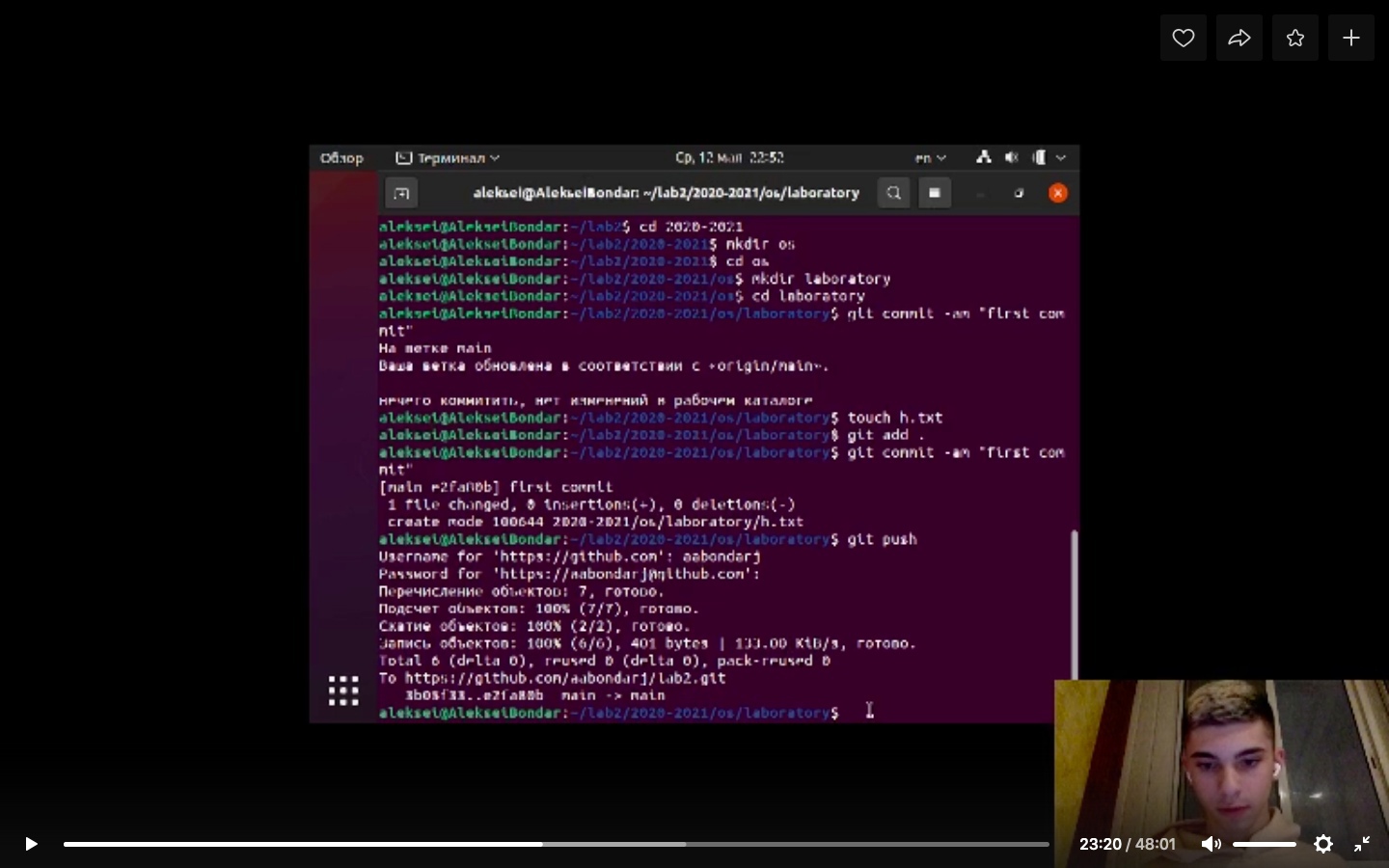


Figure 8: Первый коммит

Сохраняем первый коммит, используя команду git push.(рис. 9)

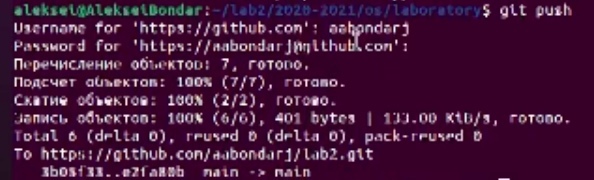


Figure 9: Сохранение

Первичная конфигурация:

1. Добавляем файл лицензии.(рис. 10)

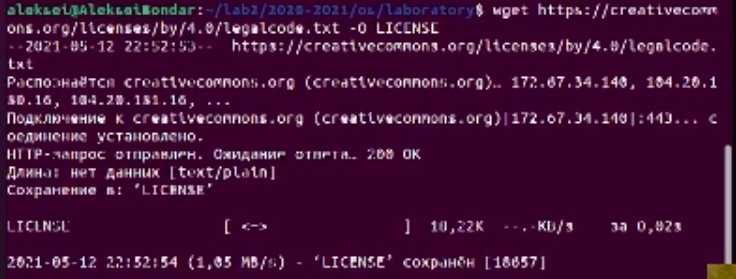


Figure 10: Файл лицензии

1. Добавим шаблон игнорируемых файлов. Просмотрим список имеющихся шаблонов (на скриншоте список шаблонов представлен не в целом виде).(рис. 11)

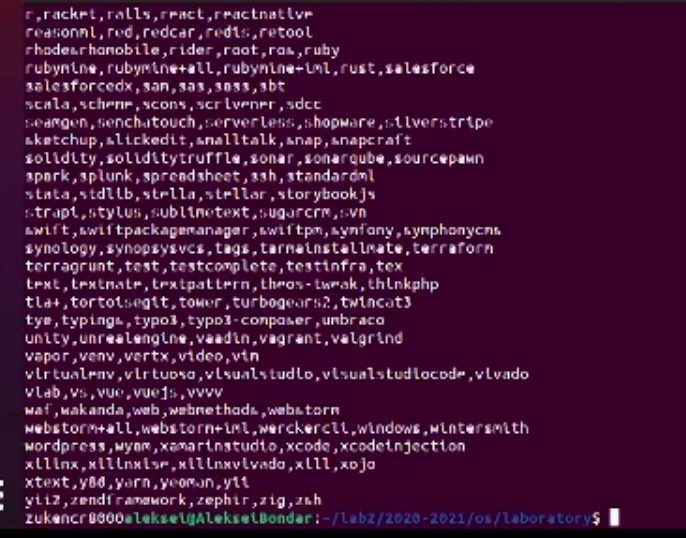


Figure 11: Шаблон игнорируемых файлов

1. Скачиваем шаблон, например, для C. Также добавляем новые файлы и выполняем коммит.(рис. 12)

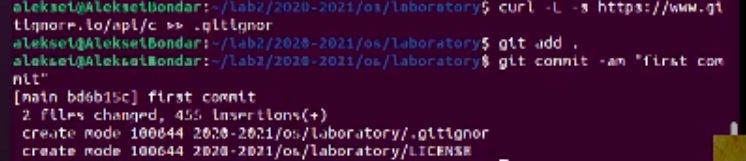


Figure 12: Скачивание шаблона

1. Отправим на github (для этого сохраним все созданные шаблоны и файлы, используя команду git push).(рис. 13)

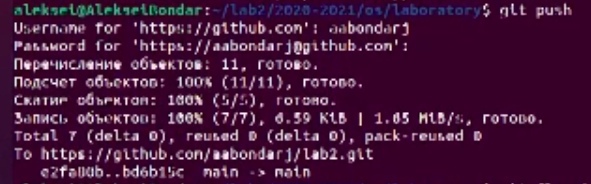


Figure 13: Отправление на github

Работаем с конфигурацией git-flow.

1. Инициализируем git-flow, используя команду git flow init -f (префикс для ярлыков установлен в v).(рис. 14)

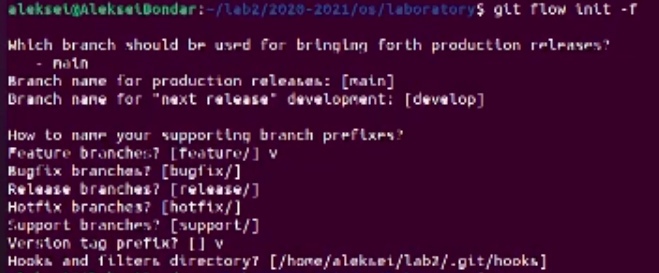


Figure 14: Инициализация git-flow

1. Проверяем, что мы находимся на ветке develop (используем команду git branch).(рис. 15)

Figure 15: git brunch

Figure 15: git brunch

(рис. 16)

Figure 16: Ветка develop

Figure 16: Ветка develop

1. Создаём релиз с версией 1.0.0.(рис. 17)

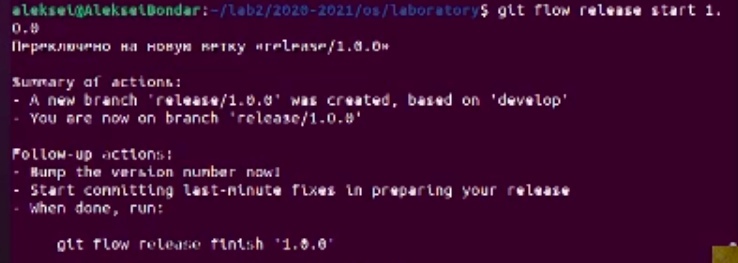


Figure 17: Создание релиза

1. Запишем версию и добавим в индекс.(рис. 18)

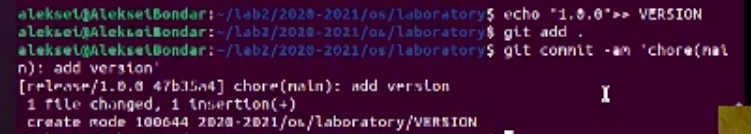


Figure 18: Версия

1. Заливаем релизную ветку в основную ветку (используем команду git flow release finish1.0.0).(рис. 19)

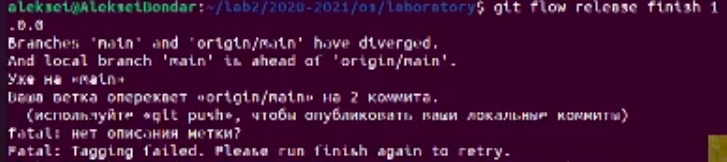


Figure 19: Релизная ветка

1. Отправляем данные на github.(рис. 20)

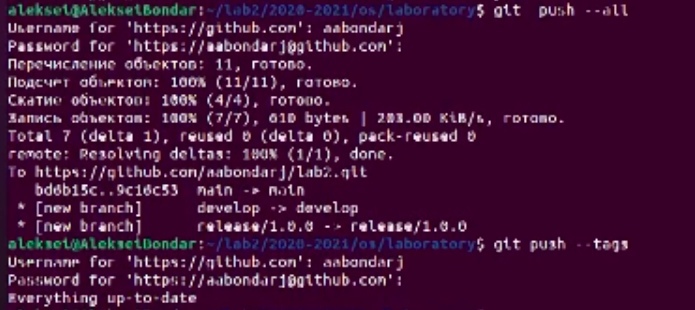


Figure 20: Отправление данных

Создаем релиз на github. Для этого заходим в «Releases», нажимаем «Создать новый релиз». Заходим в теги и заполняеем все поля (создаём теги для версии 1.0.0). После создания тега, автоматически сформируется релиз.(рис. 21)

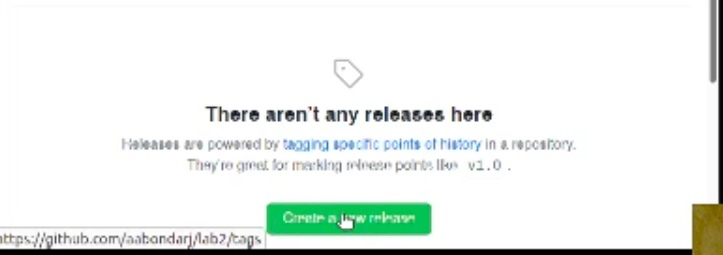


Figure 21: Releases

(рис. 22)

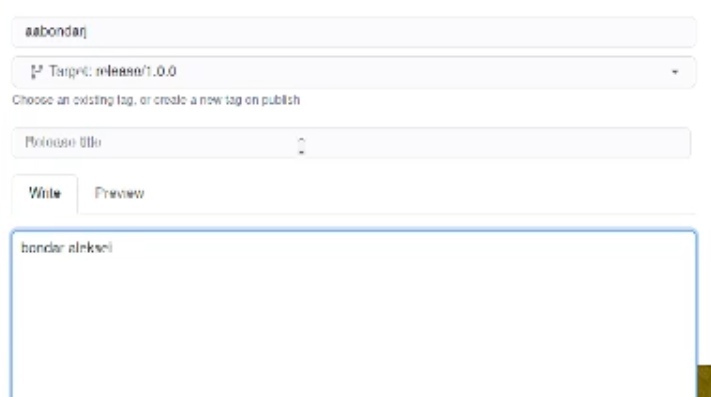


Figure 22: Заполнение полей

(рис. 23)



Figure 23: Сформированный релиз

# Выводы

Я изучил идеологию и научился применять средства контроля версий.