

Лабораторная работа 2

Управление версиями

Бешкуров Тимофей Борисович

Содержание

Цель работы	3
Задание	4
Выполнение лабораторной работы	5
Выводы	14
Контрольные вопросы	15

Цель работы

Изучить идеологию и применение средств контроля версий. Освоить работу с git.

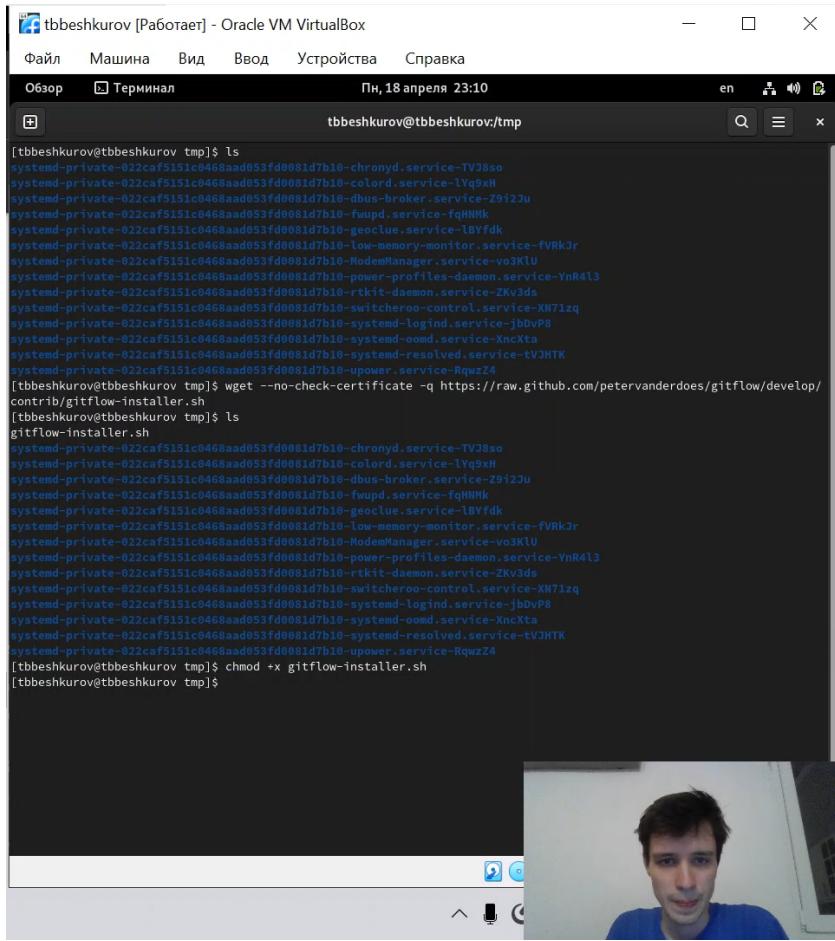
Задание

- Сделайте отчёт по предыдущей лабораторной работе в формате Markdown.
- В качестве отчёта просьба предоставить отчёты в 3 форматах: pdf, docx и md (в архиве)

Выполнение лабораторной работы

Необходимо завести учетную запись на сайте github.com. У меня уже существует аккаунт на гитхабе, поэтому этот пункт я пропустил.

1 Скачивание и установка git-flow (рис. 1 и рис. 2)



```
[tbbeshkurov@tbbeshkurov tmp]$ ls
systemd-private-022caf5151c0468aad053fd0081d7b10-chrony.service-TVJ8so
systemd-private-022caf5151c0468aad053fd0081d7b10-colord.service-1Yq9XH
systemd-private-022caf5151c0468aad053fd0081d7b10-dbus-broker.service-Z9i2Ju
systemd-private-022caf5151c0468aad053fd0081d7b10-fwupd.service-fqNMMk
systemd-private-022caf5151c0468aad053fd0081d7b10-geoclue.service-lBYfdk
systemd-private-022caf5151c0468aad053fd0081d7b10-low-memory-monitor.service-FVRkJr
systemd-private-022caf5151c0468aad053fd0081d7b10-ModemManager.service-v03KLU
systemd-private-022caf5151c0468aad053fd0081d7b10-power-profiles-daemon.service-YnR4l3
systemd-private-022caf5151c0468aad053fd0081d7b10-rtkit-daemon.service-ZKv3ds
systemd-private-022caf5151c0468aad053fd0081d7b10-switcheroo-control.service-XN71zq
systemd-private-022caf5151c0468aad053fd0081d7b10-systemd-logind.service-jbDvP8
systemd-private-022caf5151c0468aad053fd0081d7b10-systemd-oomd.service-KncXta
systemd-private-022caf5151c0468aad053fd0081d7b10-systemd-resolved.service-tVJHTK
systemd-private-022caf5151c0468aad053fd0081d7b10-upower.service-Rqwz4
[tbleshkurov@tbleshkurov tmp]$ wget --no-check-certificate -q https://raw.github.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh
[tbleshkurov@tbleshkurov tmp]$ ls
gitflow-installer.sh
[tbleshkurov@tbleshkurov tmp]$ chmod +x gitflow-installer.sh
[tbleshkurov@tbleshkurov tmp]$
```

Рис. 1: Скачивание git-flow

```
[tbbeshkurov@tbbeshkurov tmp]$ sudo ./gitflow-installer.sh install stable
[sudo] пароль для tbbeshkurov:
## git-flow no-make installer ##
Installing git-flow to /usr/local/bin
Cloning repo from GitHub to gitflow
Клонирование в «gitflow»...
remote: Enumerating objects: 4270, done.
remote: Total 4270 (delta 0), reused 0 (delta 0), pack-reused 4270
Получены объектов: 100% (4270/4270), 1.74 МиБ | 1.25 МиБ/с, готово.
Определение изменений: 100% (2533/2533), готово.
Уже обновлено.
Ветка «master» отслеживает внешнюю ветку «master» из «origin».
Переключено на новую ветку «master»
```

Рис. 2: Установка

2 Установка пакета gh (рис. 3)

```
tbbeshkurov [Работает] - Oracle VM VirtualBox
Файл Машина Вид Ввод Устройства Справка
Обзор Терминал Пн, 18 апреля 23:17
en ⚡ 🌐 🔍
[tbbeshkurov@tbbeshkurov:tmp]$ sudo dnf install gh
Fedora 35 - x86_64
Fedora 35 openSUSE (From Cisco) - x86_64
Fedora Modular 35 - x86_64
Fedora 35 - x86_64 - Updates
Fedora Modular 35 - x86_64 - Updates
Зависимости разрешены.

=====
Пакет Архитектура Версия Репозиторий Размер
=====
Установка:
gh x86_64 2.7.0-1.fc35 updates 6.8 M

Результат транзакции
=====
Установка 1 Пакет

Объем загрузки: 6.8 М
Объем изменений: 32 М
Продолжить? [д/н]: у
Загрузка пакетов:
gh-2.7.0-1.fc35.x86_64.grm 145 kB/s | 6.8 MB 00:47
Общий размер 143 kB/s | 6.8 MB 00:48

Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции проведен успешно.
Выполнение транзакции
Подготовка : 1/1
Установка : gh-2.7.0-1.fc35.x86_64 1/1
Запуск скриплета: gh-2.7.0-1.fc35.x86_64 1/1
Проверка : gh-2.7.0-1.fc35.x86_64 1/1

Установлен:
gh-2.7.0-1.fc35.x86_64

Выполнено!
[tbbeshkurov@tbbeshkurov:tmp]$ git config --global user.name "Name Surname"
```

Рис. 3: Установка gh

3 Базовая настройка git: имя, email, кодировка, имя начальной ветки, autocrlf и safecrlf (рис. 4)

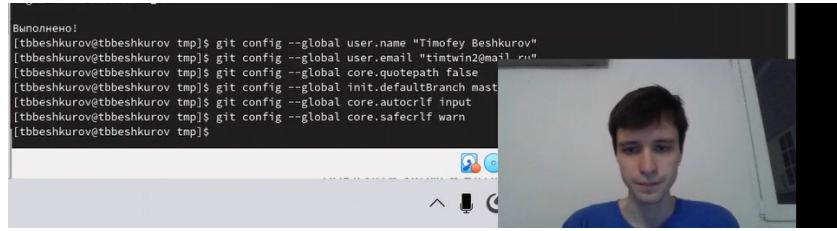


Рис. 4: Базовая настройка git

4 Создание ssh ключей с двумя алгоритмами шифрования: RSA и ed25519 (рис. 5)

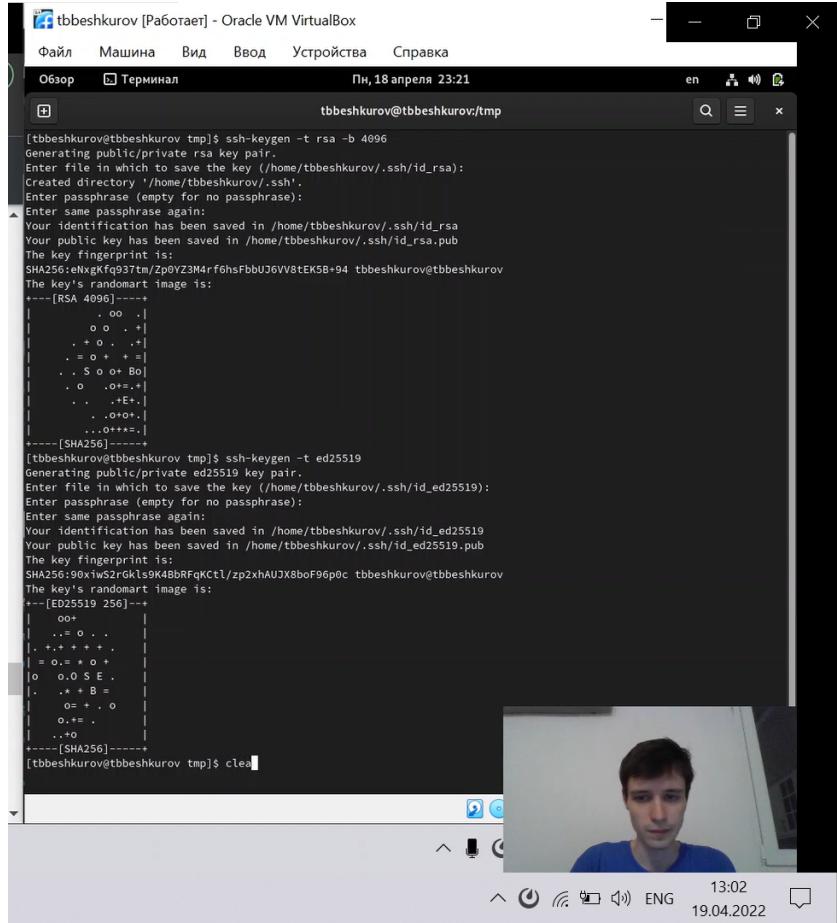


Рис. 5: Создание ssh ключей

5 Создание pgp ключа (рис. 6)

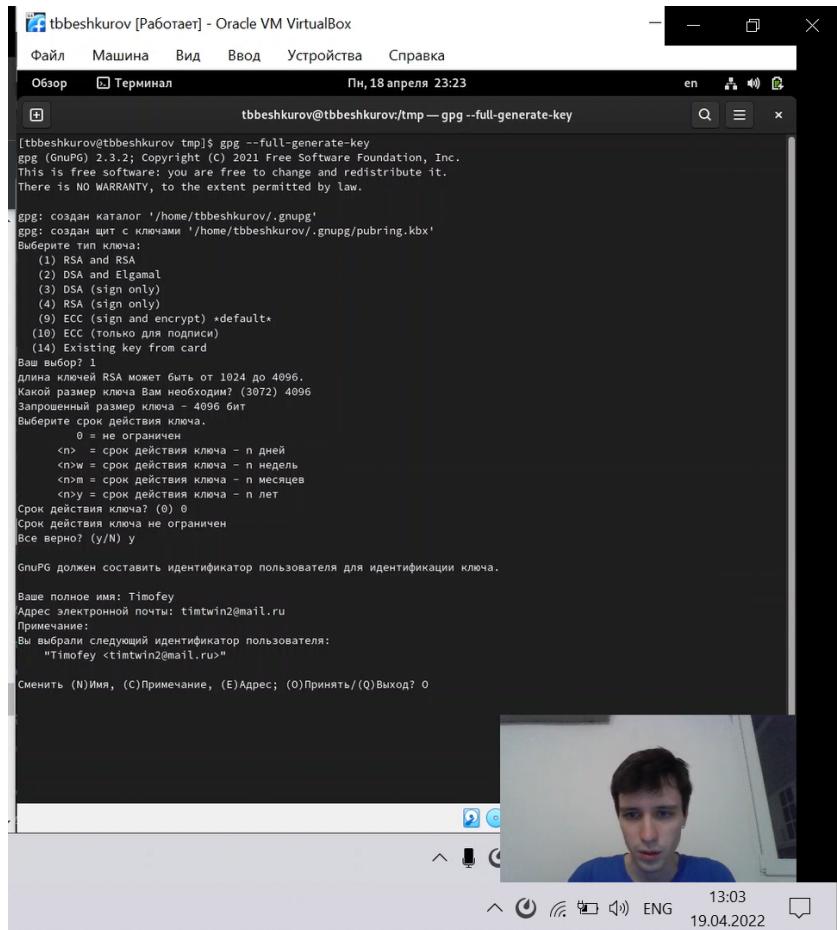


Рис. 6: Создание pgp ключа

6 Добавление PGP ключа на GitHub (рис. 7 и рис. 8)

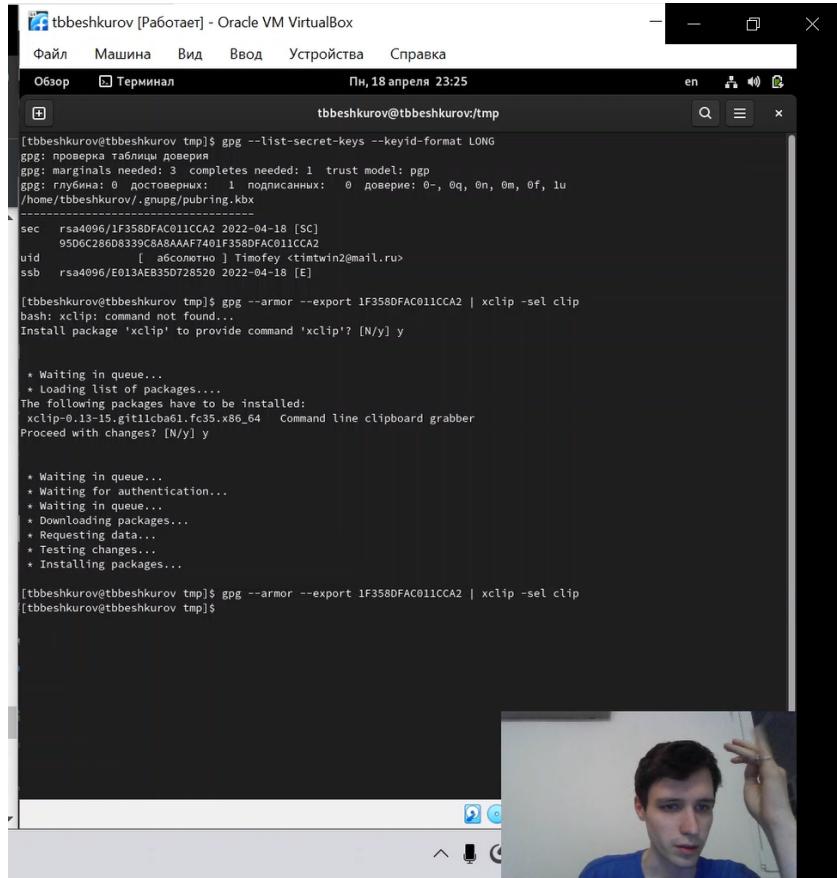


Рис. 7: Просмотр отпечатка и копирование в буфер pgp ключа

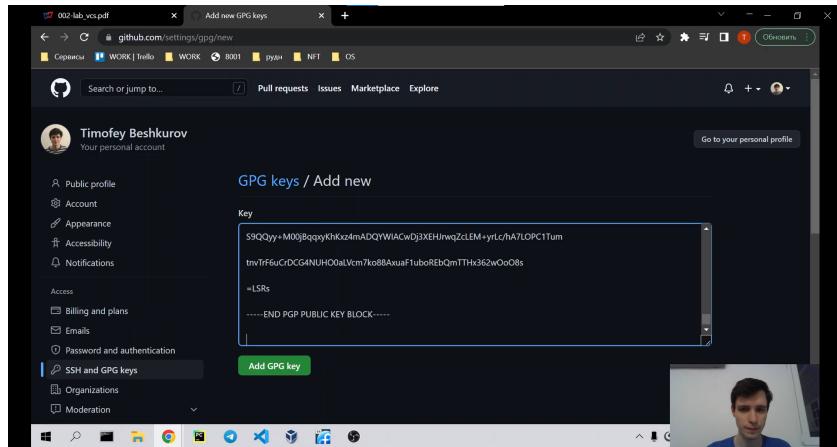


Рис. 8: Добавление PGP ключа на GitHub

7 Создание и клонирование репозитория (рис. 9)

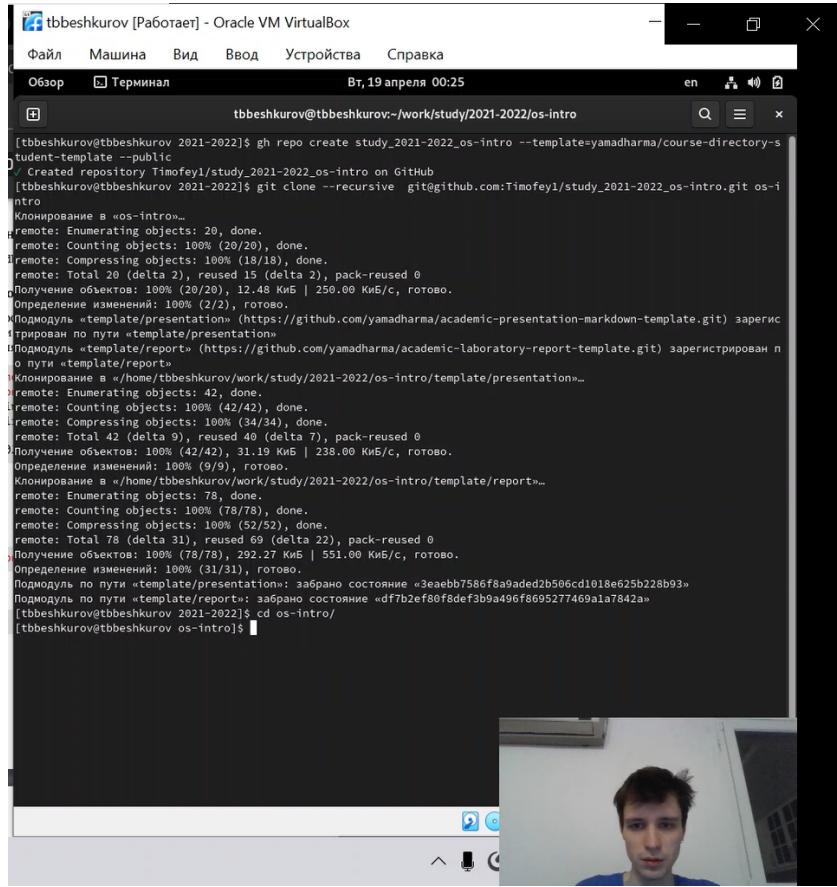


Рис. 9: Создание и клонирование репозитория

8 Настройка каталога курса: Удаление json файла и создание структуры (рис. 10), создание коммита (рис. 11), пуш на сервер (рис. 12)

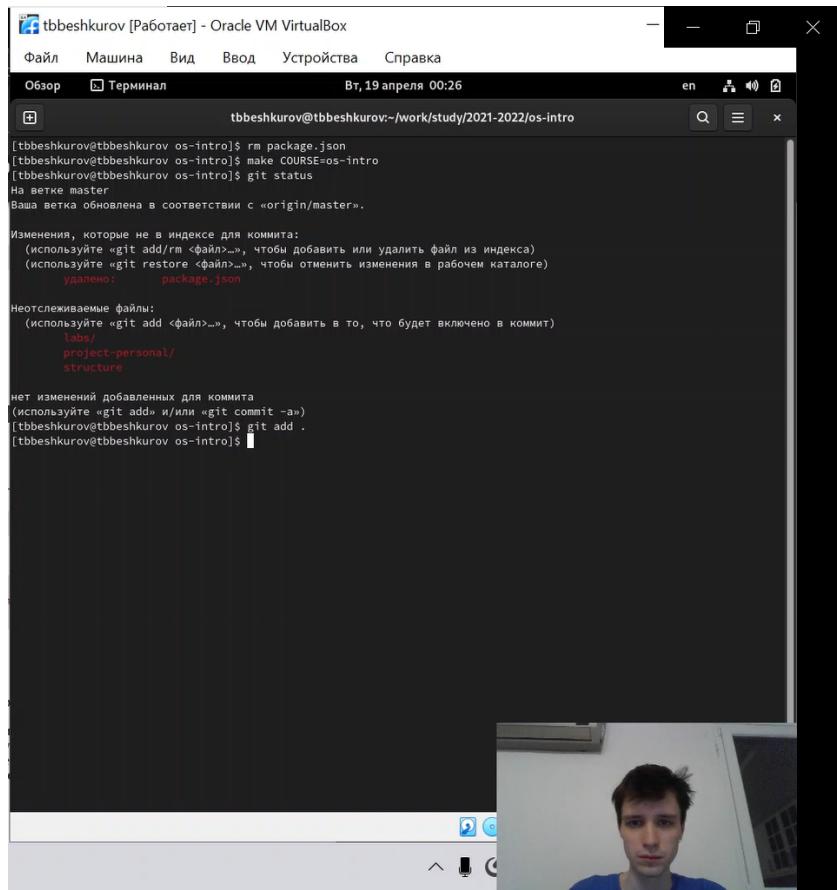


Рис. 10: Удаление json файла и создание структуры

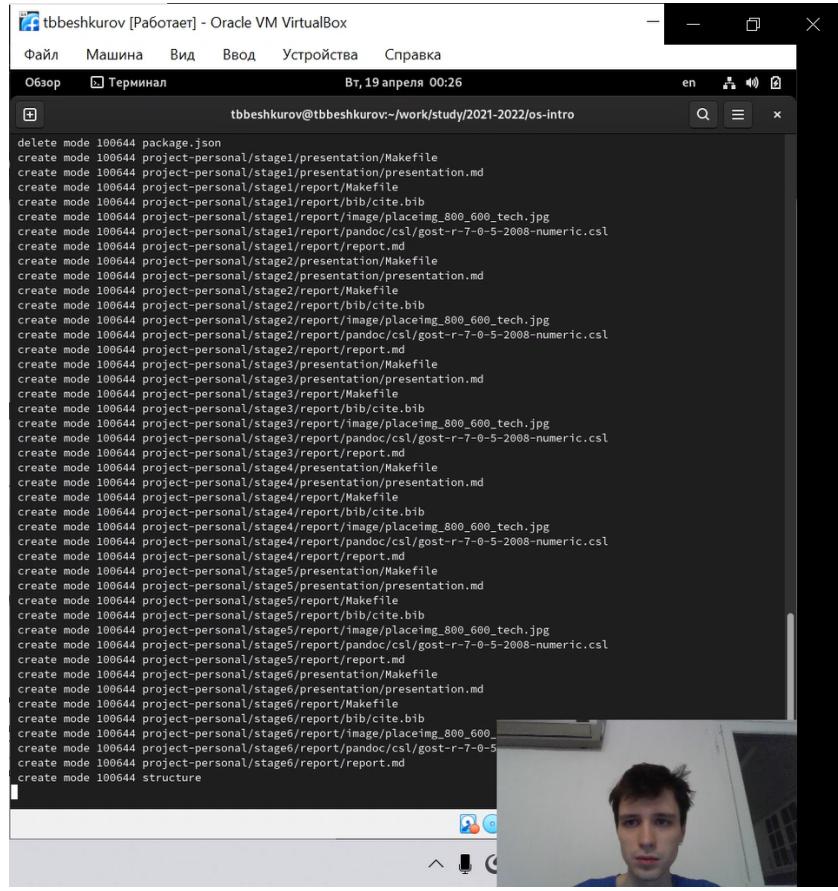


Рис. 11: Коммит

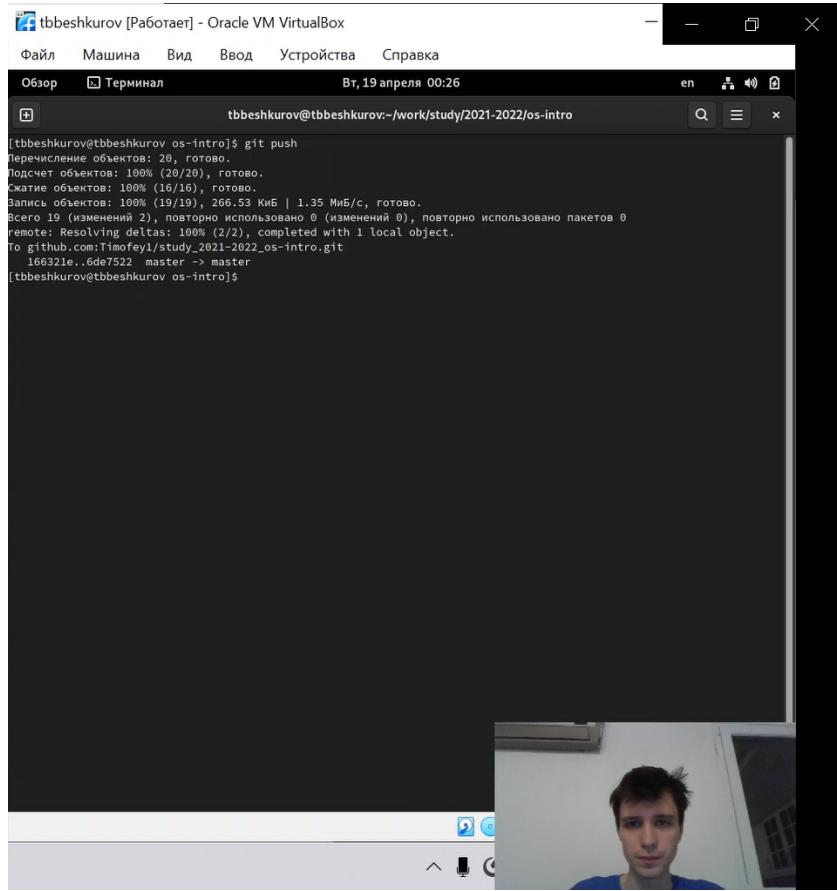


Рис. 12: Пуш на сервер

9 Проверим что репозиторий появился на GitHub (рис. 13)

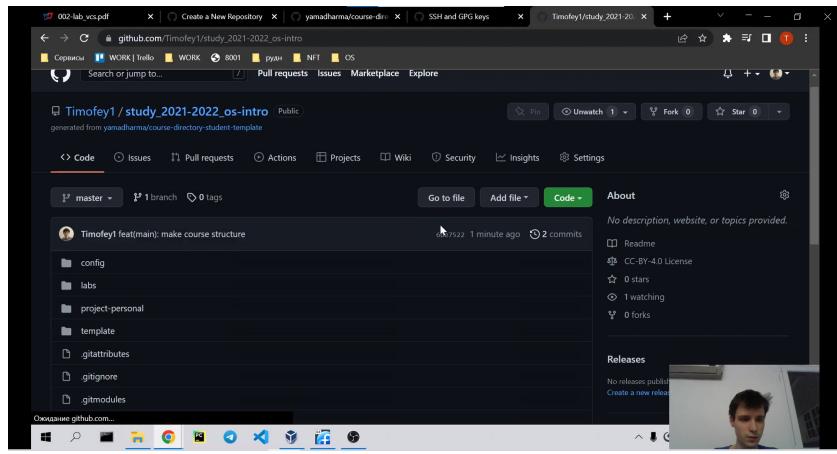


Рис. 13: Репозиторий для работ по ОС

Выводы

Получили знания о системах контроля версий, изучили идеологию и применение данной технологии. Получили практические навыки по настройке системы, а также о создании репозитория и обработки файлов.

Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Система контроля версий (VCS) — это место хранения кода. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое. Такие системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы.
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Репозиторий - хранилище версий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией Commit («[трудовой] вклад», не переводится) — процесс создания новой версии Рабочая копия (working copy) — текущее состояние файлов проекта, основанное на версии, загруженной из хранилища (обычно на последней). Версия (revision), или ревизия, — состояние всех файлов на определенный момент времени, сохраненное в репозитарии, с дополнительной информацией
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. (Пример — Wikipedia.) В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. (Пример — Bitcoin)

4. Опишите действия с VCS при единоличной работе с хранилищем. Изначально разработчик работает с веткой master. При реализации отдельных частей проекта может создать ветки для них. При завершении изменений разработчик коммитит и пушит изменения на сервер. Если разработка на сторонней ветке завершена, то её можно смерджить (merge), например с основной веткой master.
5. Опишите порядок работы с общим хранилищем VCS. Каждый разработчик работает в своей ветке над отдельной частью проекта. Если на ветке работает несколько разработчиков они как правило достать изменения, сделанные другим и работать уже с ними (git pull). После завершения или заканчивая какой-то логический кусок они делают коммит и пушат на сервер. Если работа закончена необходимо смерджить ветки, например с главной веткой(master)
6. Каковы основные задачи, решаемые инструментальным средством git? У Git есть две основные задачи: хранить информацию обо всех изменениях в коде, начиная с самой первой строчки, и обеспечить удобства командной работы над кодом.
7. Назовите и дайте краткую характеристику командам 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 изменения: 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. Что такое и зачем могут быть нужны ветви (branches)? ‘Git branch’ – это команда для управления ветками в репозитории Git. Ветка – это просто «скользящий» указатель на один из коммитов. Когда мы создаём новые коммиты, указатель ветки автоматически сдвигается вперёд, к вновь созданному коммиту. Ветки используются для разработки одной части функционала изолированно от других. Каждая ветка представляет собой отдельную копию кода проекта. Ветки позволяют одновременно работать над разными версиями проекта. Ветвление («ветка», branch) — один из параллельных участков истории в одном хранилище, исходящих из одной версии (точки ветвления). Ветки нужны для того, чтобы программисты могли вести совместную работу над проектом и не мешать друг другу при этом.
9. Как и зачем можно игнорировать некоторые файлы при commit? Игнорировать файлы коммита можно с помощью файла `.gitignore`. Туда обычно помещаются файлы, которые занимают много места или не особо нужны для проекта. Например, картинки, папки с кешем, веса для нейросетей и другие файлы.