

# **Лабораторная работа №2**

## **Первоначальная настройка git**

**Дисциплина Операционные системы**

**Фадин В.В. НПМбв-02-20**

# **Содержание**

|                                              |           |
|----------------------------------------------|-----------|
| <b>1 Цель работы</b>                         | <b>5</b>  |
| <b>2 Выполнение лабораторной работы</b>      | <b>6</b>  |
| 2.1 Настройка GIT . . . . .                  | 6         |
| 2.2 Создание рабочего пространства . . . . . | 10        |
| <b>3 Ответы на контрольные вопросы</b>       | <b>12</b> |
| <b>4 Выводы</b>                              | <b>15</b> |

# Список иллюстраций

|                                                                    |    |
|--------------------------------------------------------------------|----|
| 2.1 Конфигурация GIT 1 . . . . .                                   | 6  |
| 2.2 Конфигурация GIT 2 . . . . .                                   | 6  |
| 2.3 Создание ключа ssh 1 . . . . .                                 | 7  |
| 2.4 Создание ключа ssh 2 . . . . .                                 | 7  |
| 2.5 Создание ключа ssh 3 . . . . .                                 | 7  |
| 2.6 Создание PGP ключа . . . . .                                   | 8  |
| 2.7 Добавление ключей в Github . . . . .                           | 8  |
| 2.8 Добавление ключей в Github . . . . .                           | 9  |
| 2.9 Авторизация в Github . . . . .                                 | 9  |
| 2.10 Создание репозитория 1 . . . . .                              | 10 |
| 2.11 Удаление файла package.json и создание файла COURSE . . . . . | 10 |
| 2.12 Git push . . . . .                                            | 11 |

# **Список таблиц**

# **1 Цель работы**

- Изучить применение средств контроля версий
- Освоить умения по работе с git'ом

## 2 Выполнение лабораторной работы

### 2.1 Настройка GIT

Сконфигурируем git и создадим SSH ключ

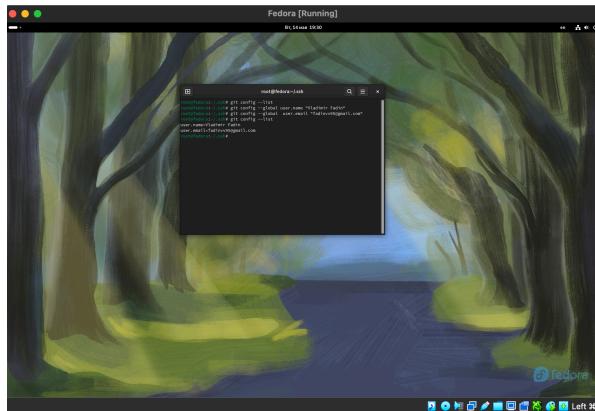


Рис. 2.1: Конфигурация GIT 1

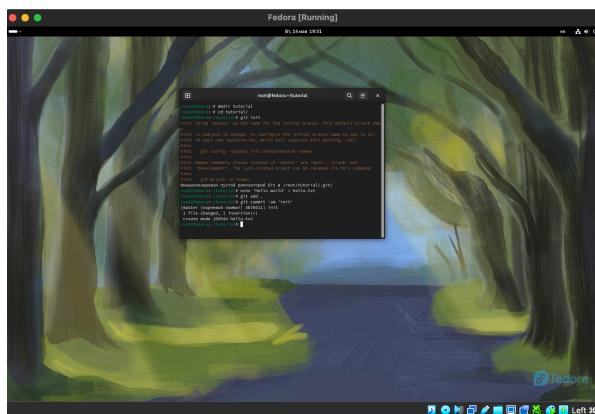


Рис. 2.2: Конфигурация GIT 2

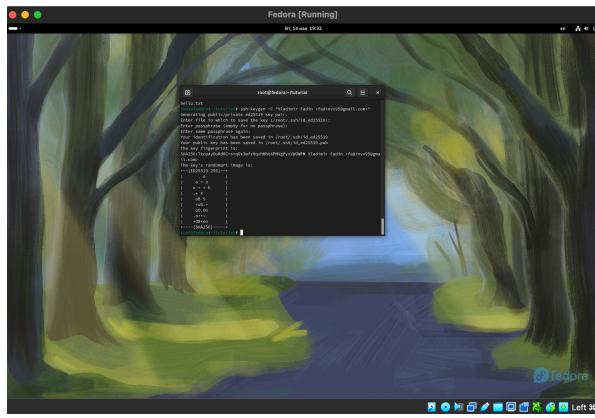


Рис. 2.3: Создание ключа ssh 1

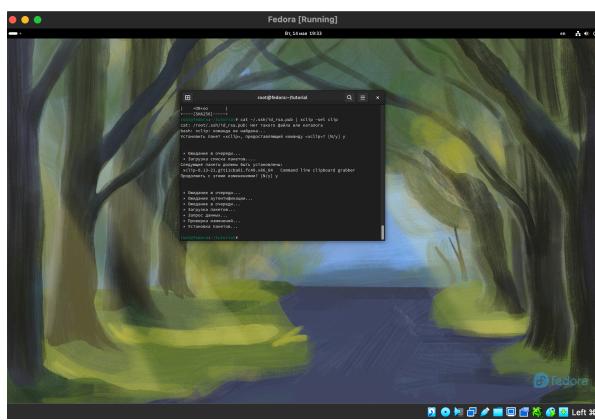


Рис. 2.4: Создание ключа ssh 2

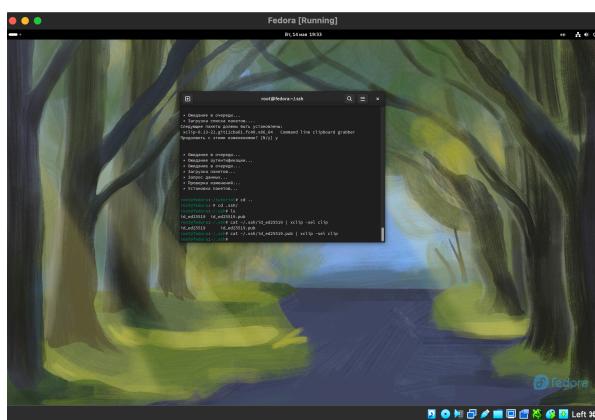


Рис. 2.5: Создание ключа ssh 3

Создадим PGP ключ

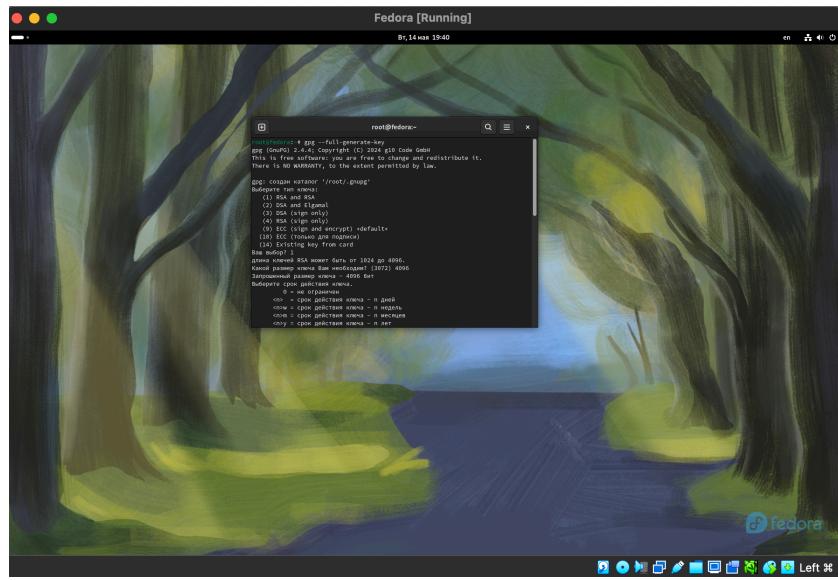


Рис. 2.6: Создание PGP ключа

С помощью команды `gpg --armor --export <PGP Fingerprint> | xclip -sel clip` скопируем PGP ключ. И вставим в наш Github.

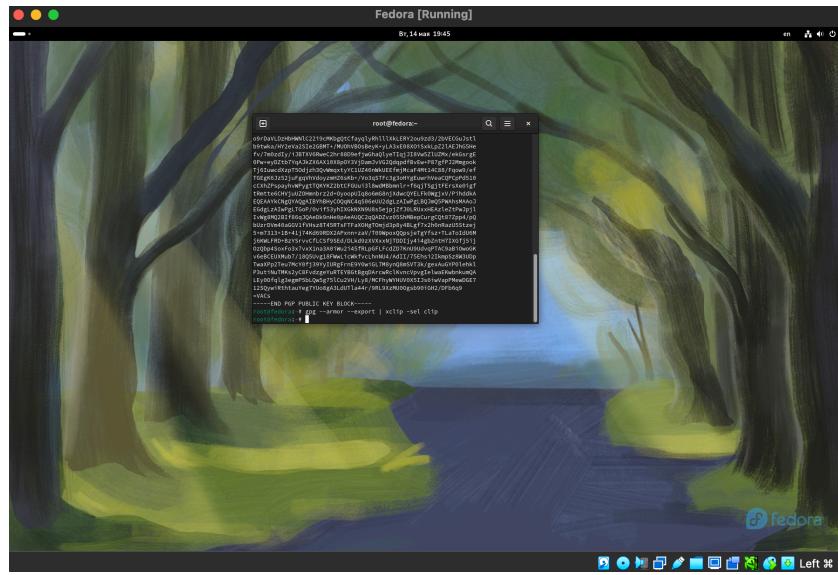


Рис. 2.7: Добавление ключей в Github

Рис. 2.8: Добавление ключей в Github

## Авторизуемся в Github

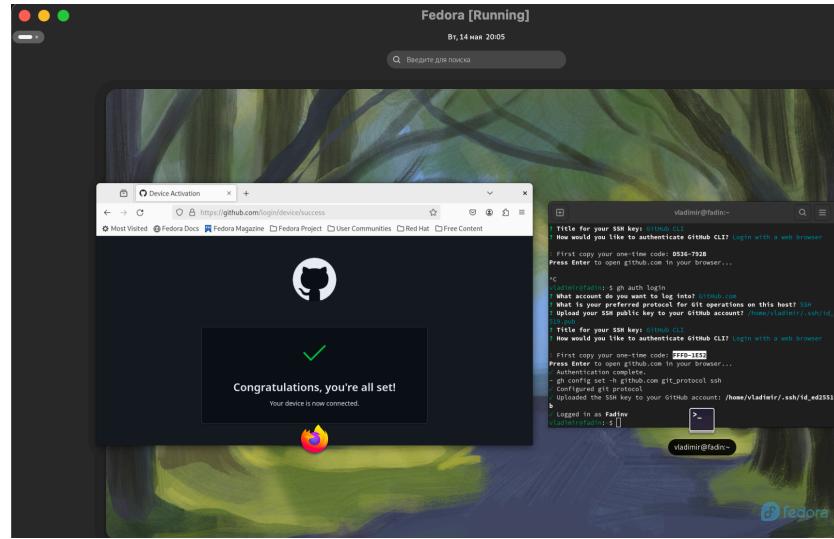
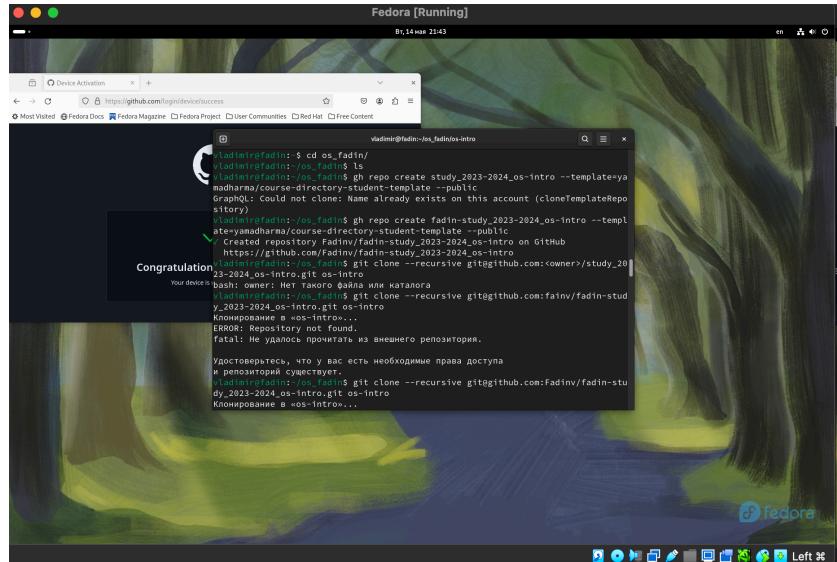


Рис. 2.9: Авторизация в Github

## 2.2 Создание рабочего пространства

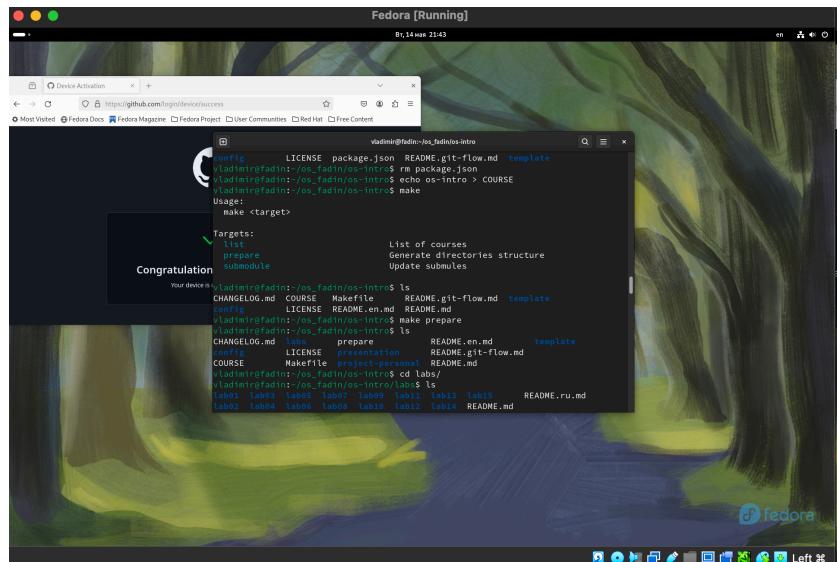
Переходим в репозиторий с шаблоном и создаем из него шаблон.



The screenshot shows a terminal window titled "Fedora [Running]" on a Fedora desktop environment. The user is in a directory named "os\_fadin/os-intro". They run the command "gh repo create study\_2023-2024\_os-intro --template=ya-madhama/course-directory-student-template --public". The output shows that the repository was created successfully on GitHub at [https://github.com/Fadinv/study\\_2023-2024\\_os-intro](https://github.com/Fadinv/study_2023-2024_os-intro). The terminal then attempts to clone the repository again using "git clone --recursive git@github.com:Fadinv/study\_2023-2024\_os-intro", but it fails with an "ERROR: Repository not found" message. The user also tries cloning from the GitHub URL, which also fails with a "fatal: Not found" error. A "Congratulation" message is visible in the background of the terminal window.

Рис. 2.10: Создание репозитория 1

Перейдем в репозиторий и удалим package.json. Также создадим файл COURSE с текстом “os-intro”



The screenshot shows a terminal window titled "Fedora [Running]" on a Fedora desktop environment. The user is in a directory named "os\_fadin/os-intro". They run "rm package.json" to delete the package.json file. Then they run "echo os-intro > COURSE" to create a new file named COURSE with the text "os-intro". The terminal shows the directory structure and the newly created COURSE file.

Рис. 2.11: Удаление файла package.json и создание файла COURSE

Теперь запушим изменения. До этого исполнив git add и git commit с назва-

## нием коммита feat(main): make course structure

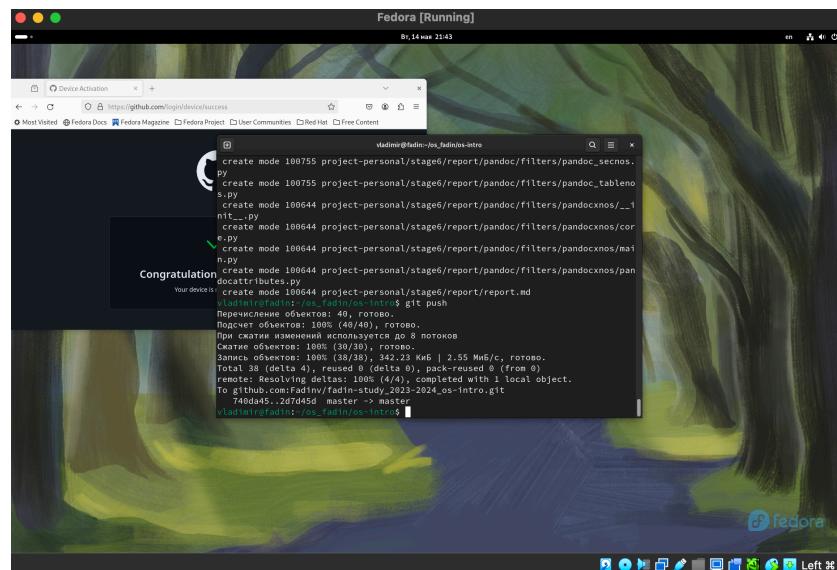


Рис. 2.12: Git push

### **3 Ответы на контрольные вопросы**

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

Системы контроля версий (VCS) предназначены для отслеживания изменений в программном коде и обеспечения коллективной разработки.

#### **3.0.0.2 Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.**

Хранилище: Место, где хранятся все изменения и версии программного кода.  
Commit: Отдельное изменение или набор изменений в коде, зафиксированное в системе контроля версий. История: Последовательность коммитов, отображающая эволюцию кода. Рабочая копия: Локальная копия проекта, с которой работает разработчик.

#### **3.0.0.3 Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.**

Децентрализованные VCS копируют всю историю изменений на каждый клиентский компьютер, в то время как централизованные VCS хранят все изменения на центральном сервере и клиенты получают только последние версии файлов. Примеры децентрализованных VCS: Git, Mercurial. Примеры централизованных VCS: Subversion, CVS.

### **3.0.0.4 Опишите действия с VCS при единоличной работе с хранилищем.**

При индивидуальной разработке пользователь клонирует проект на свой компьютер, вносит изменения и создает новые версии, коммитя их в системе контроля версий.

### **3.0.0.5 Опишите порядок работы с общим хранилищем VCS.**

Пользователь получает версию проекта из центрального хранилища, вносит изменения, коммитит их и отправляет обратно в хранилище.

### **3.0.0.6 Каковы основные задачи, решаемые инструментальным средством git?**

Git используется для разработки проектов в команде, контроля изменений в файлах и возможности сохранения нескольких состояний проекта.

### **3.0.0.7 Назовите и дайте краткую характеристику командам git.**

`git add` - добавляет изменения для коммита.

`git commit` - сохраняет изменения в репозитории с названием.

`git push` - отправляет изменения на удаленный репозиторий.

`git config` - позволяет изменить настройки Git.

### **3.0.0.8 Приведите примеры использования при работе с локальным и удалённым репозиториями.**

В локальном репозитории разработчик может вносить изменения в код и коммитить их без доступа к сети. В удаленном репозитории команда разработчиков может совместно работать над проектом, обмениваясь изменениями через централизованный сервер.

### **3.0.0.9 Что такое и зачем могут быть нужны ветви (branches)?**

Ветви используются для параллельной разработки функций или исправлений, чтобы избежать конфликтов между изменениями и обеспечить безопасное тестирование нового кода.

### **3.0.0.10 Как и зачем можно игнорировать некоторые файлы при commit?**

Файлы могут быть проигнорированы с помощью файла .gitignore, чтобы избежать загрязнения репозитория лишними или конфиденциальными файлами.

## **4 Выводы**

Мы изучили идеологию применения средств контроля версий и освоили базовые команды git'a.