РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

дисциплина:

Операционные системы

Студент: Бердыев Эзиз

Студенческий билет № 1032214711

Группа: НФИбд – 01-21

МОСКВА

Цель работы

– Изучить идеологию и применение средств контроля версий.

– Освоитьумения по работе с git.

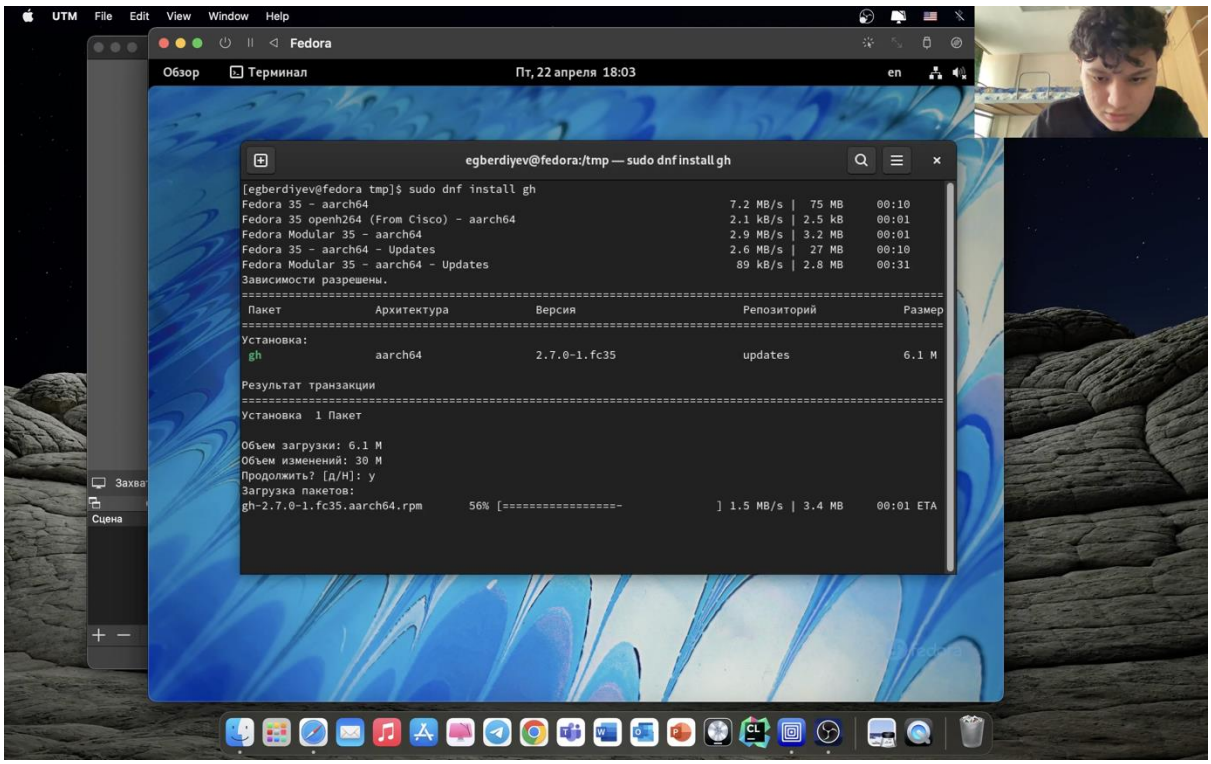

Рис.1. Установка git-flow в Fedora Linux

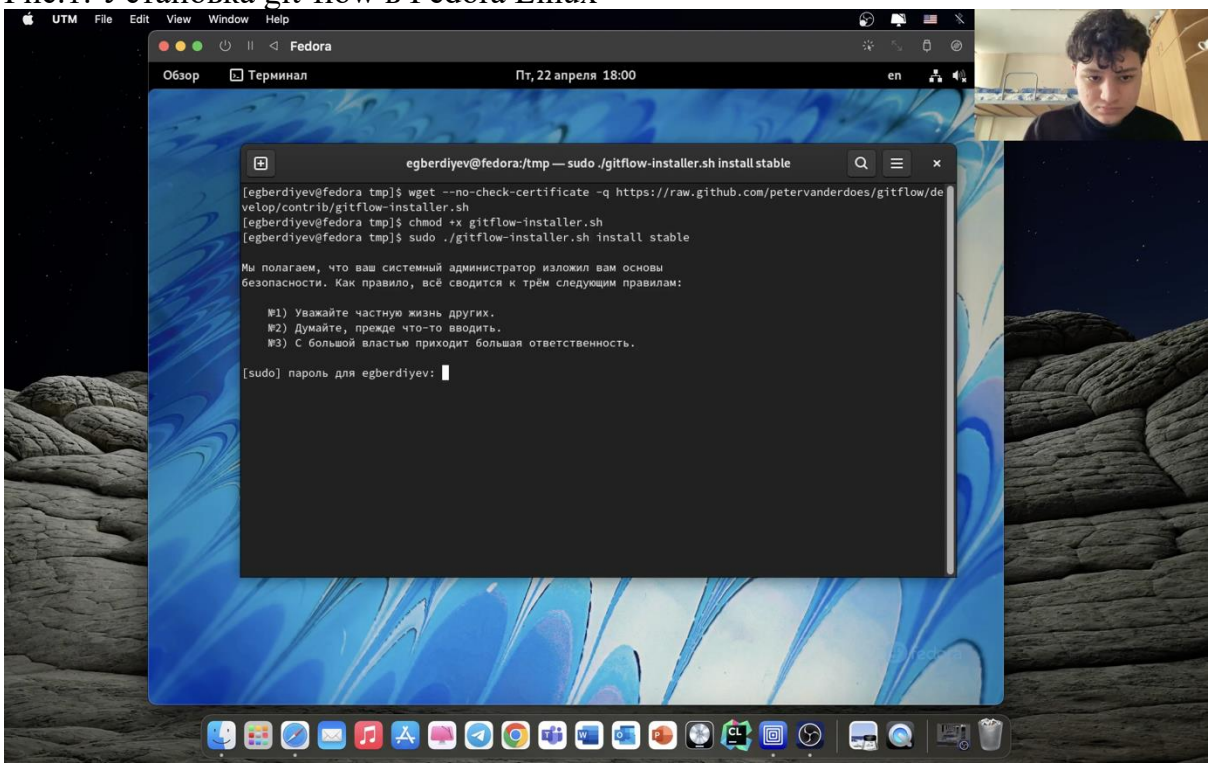

Рис.2. Установка gh в Fedora Linux

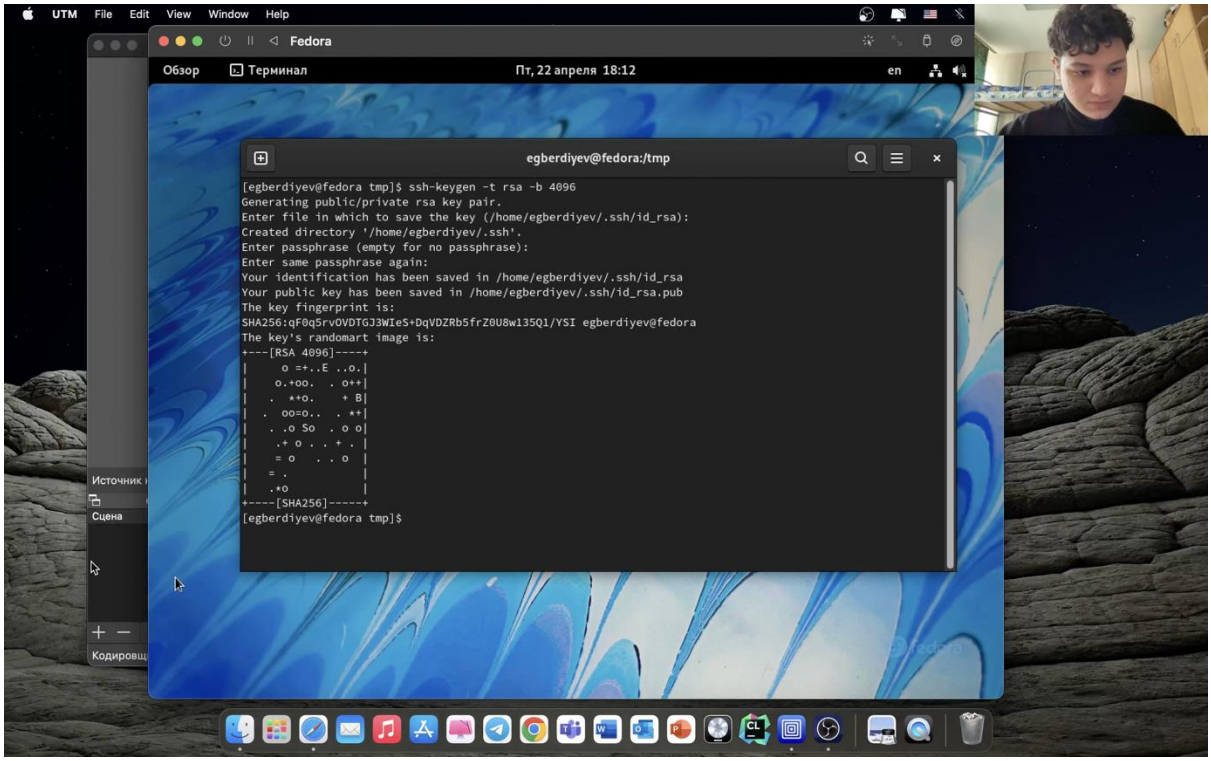

Рис.3. Базовая настройка в git

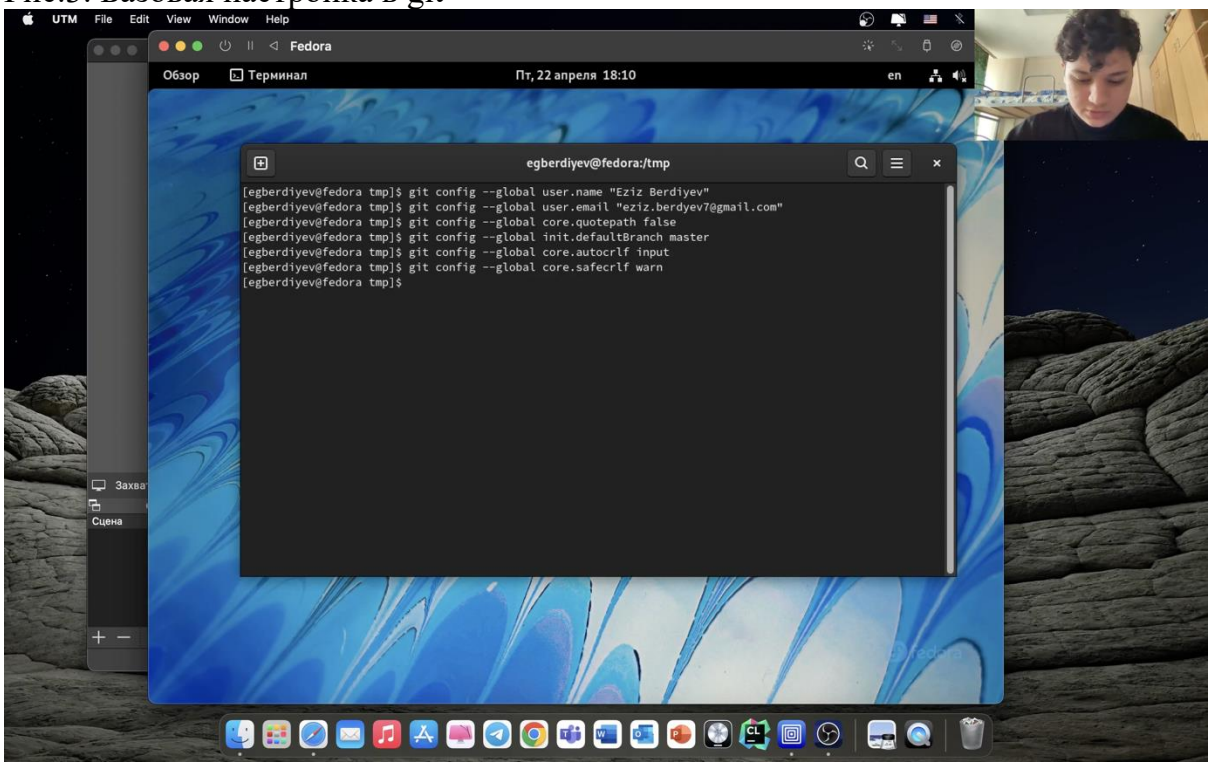

Рис.4. Создание ssh ключей

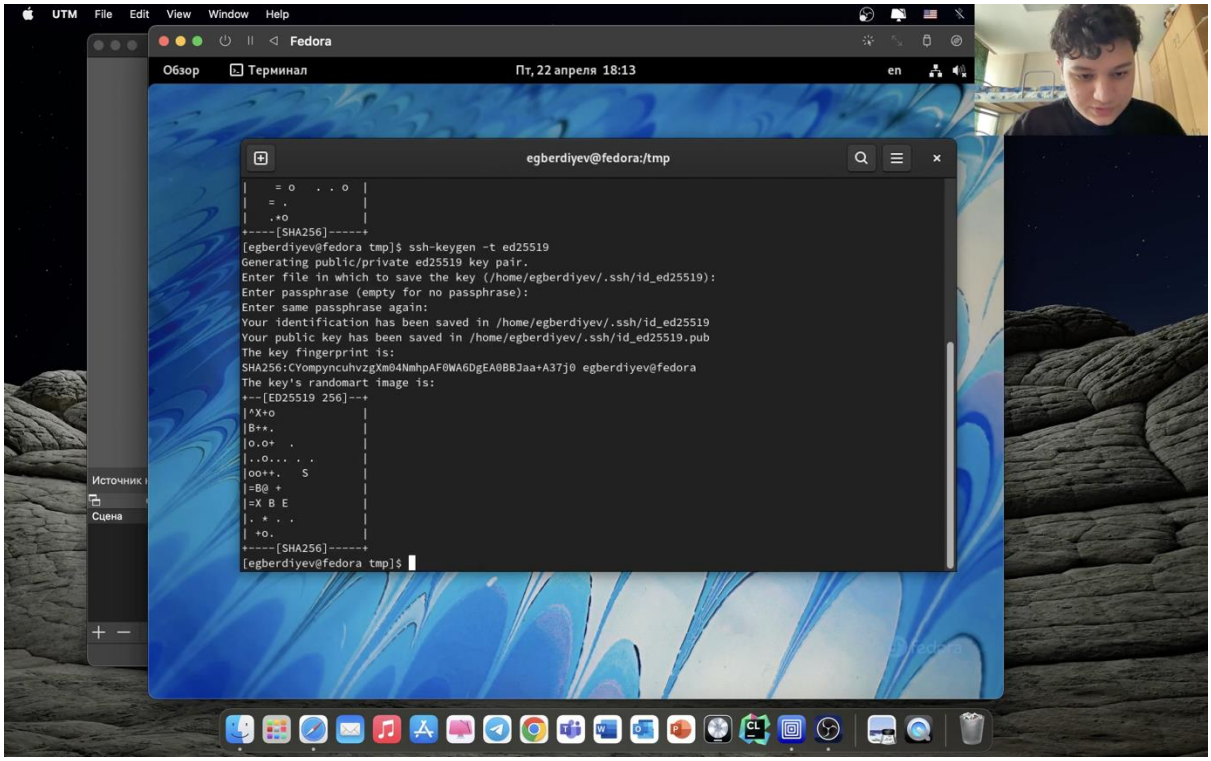

Рис.5. По алгоритму rsa и ed25519

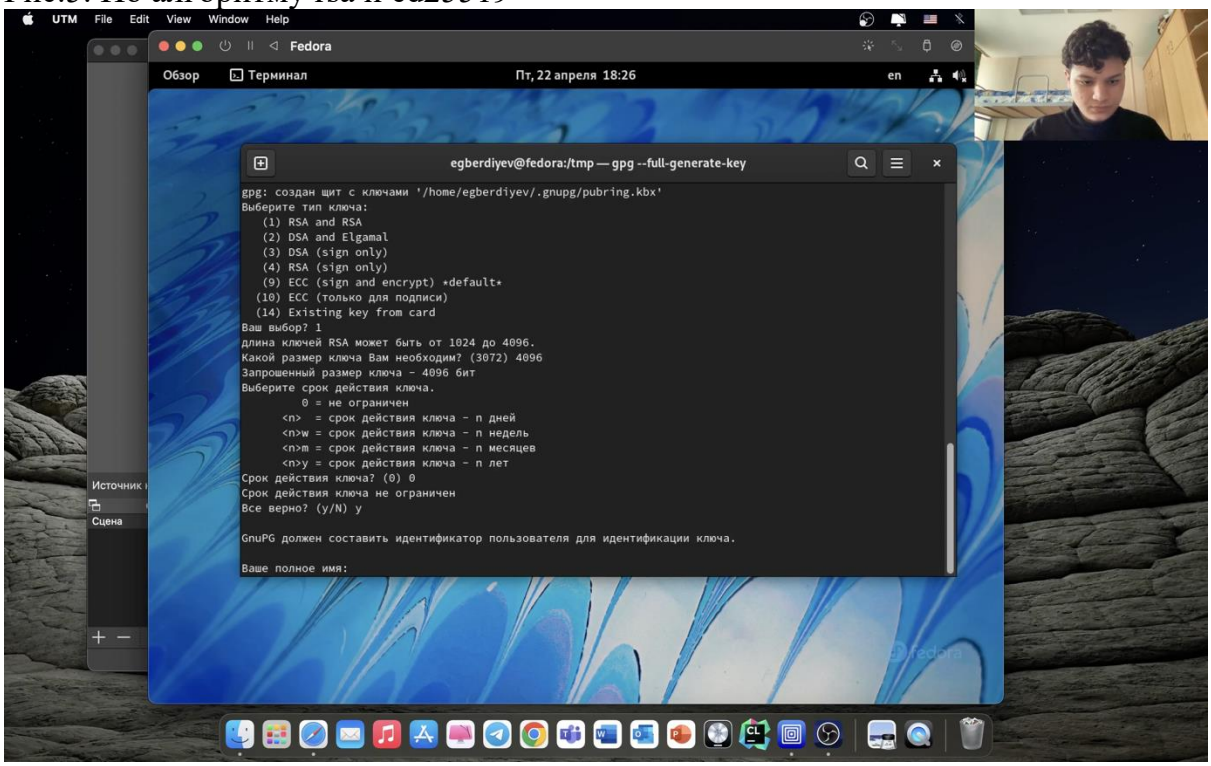

Рис.6. Создание pgp ключа

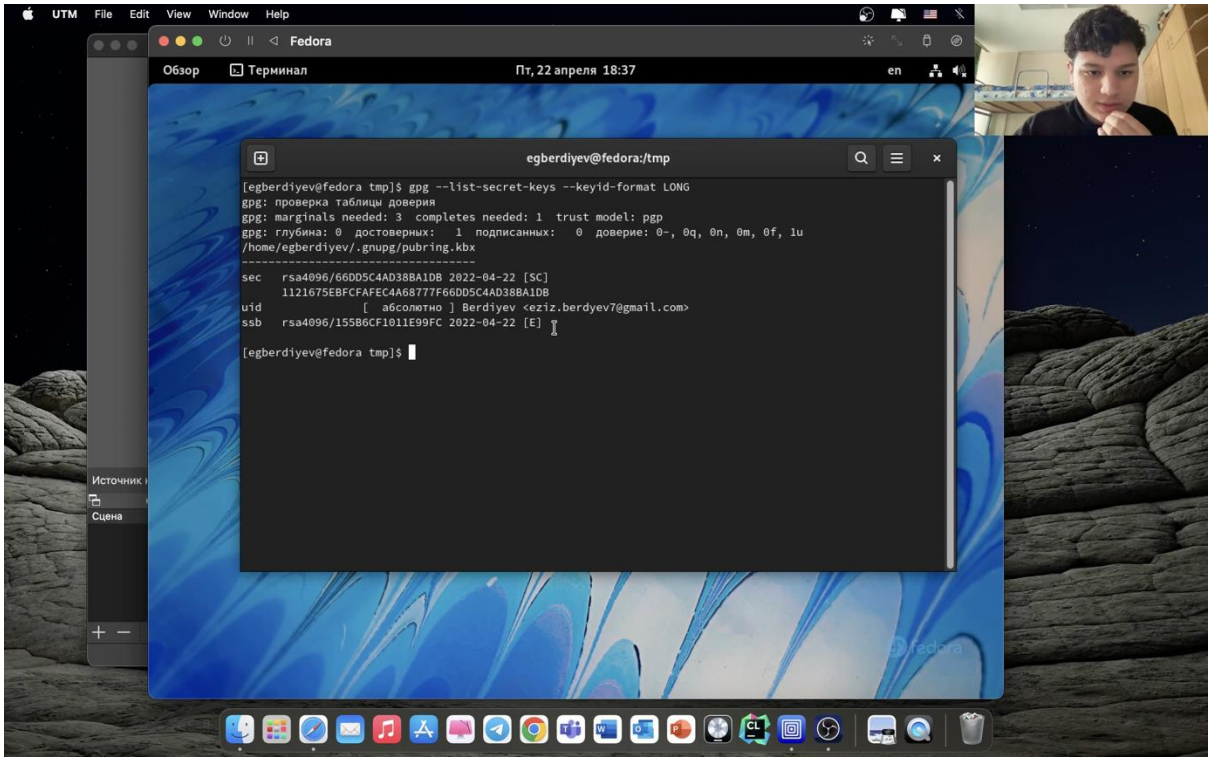

Рис.7. Добавление pgp ключа в GitHub

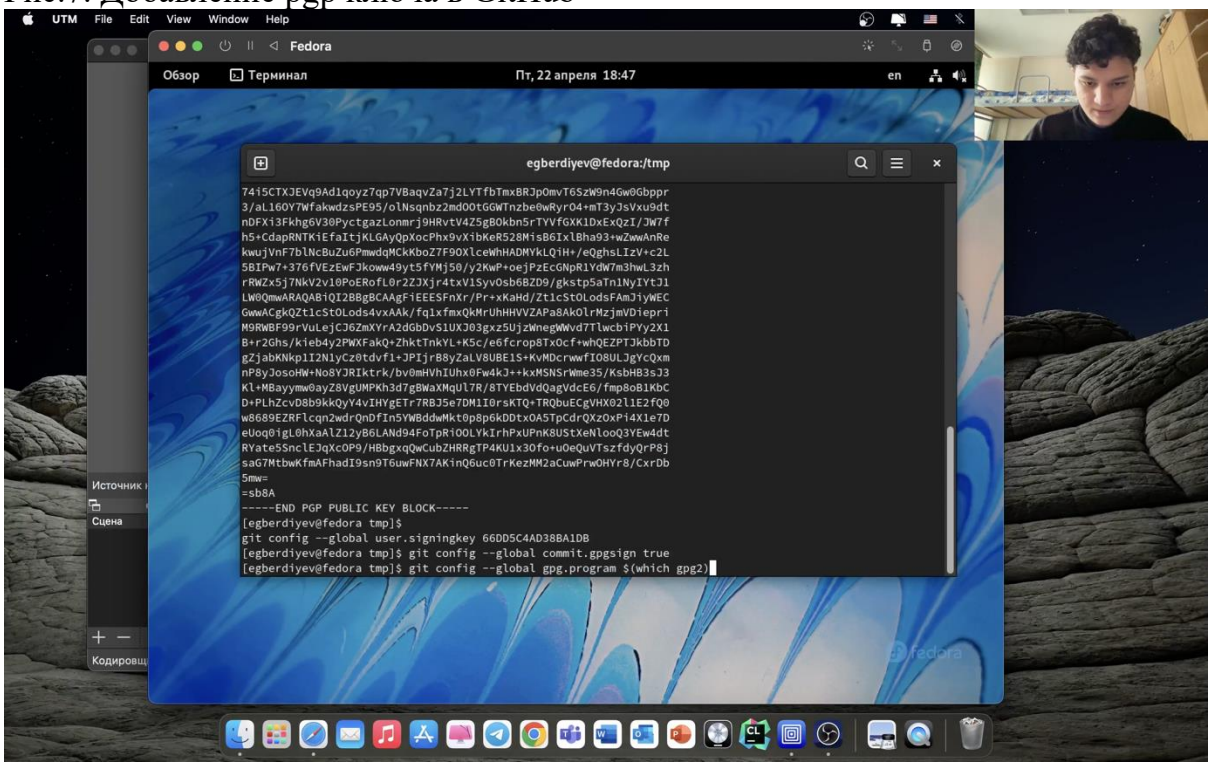

Рис.8. Настройка автоматических подписей коммитов git

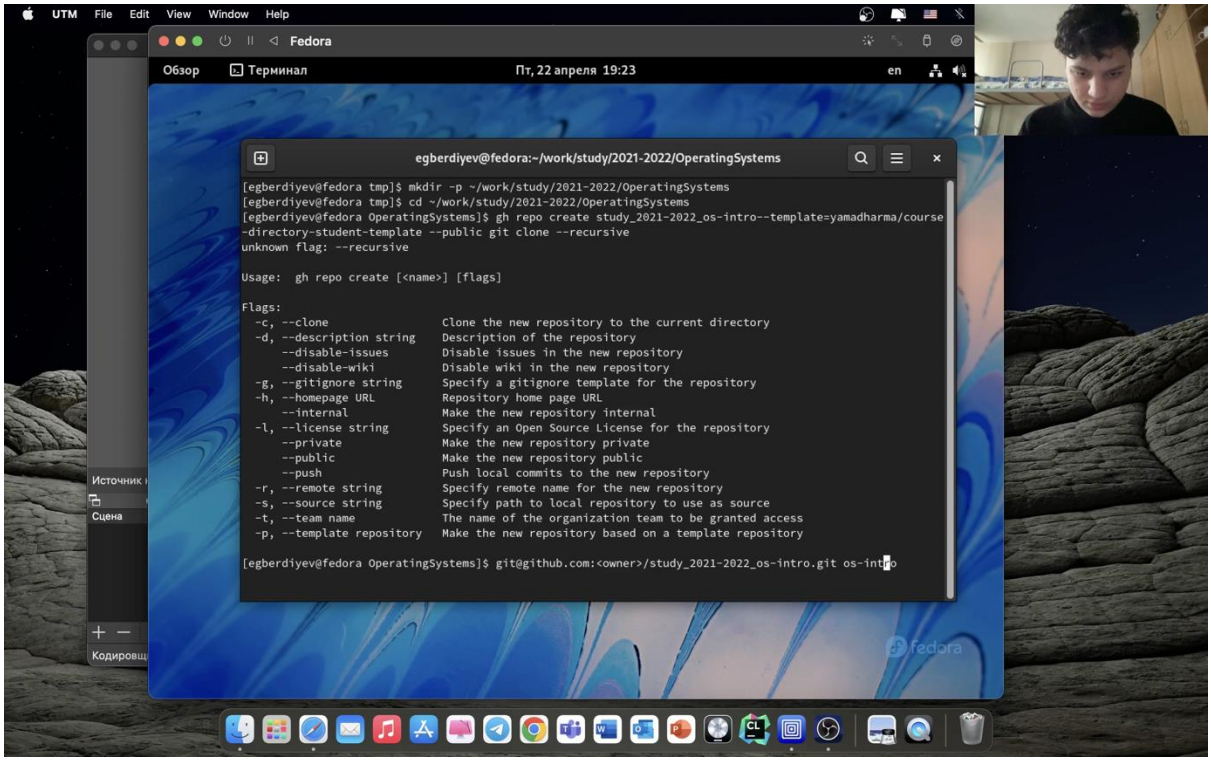

Рис.9. Создание репозиториев на основе шаблона

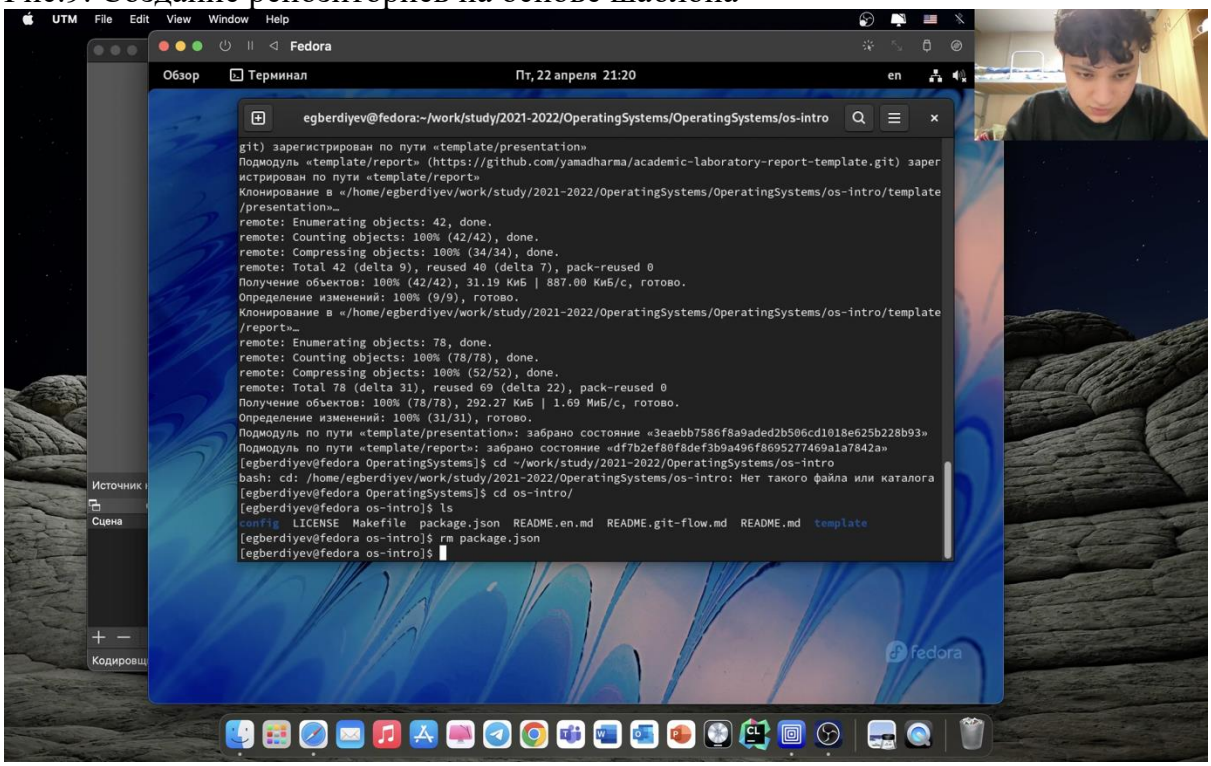

Рис.10 Настройка каталога курса и удаление лишних файлов

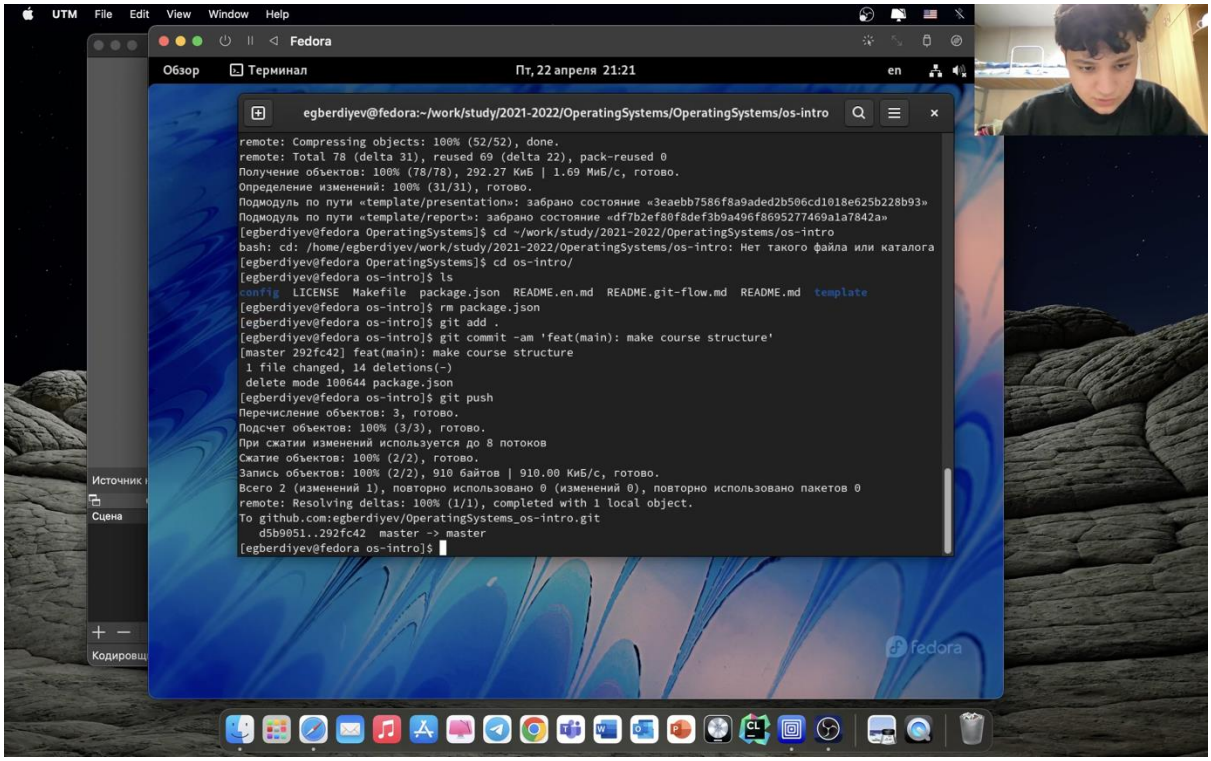

Рис.11. Отправка файлов на сервер

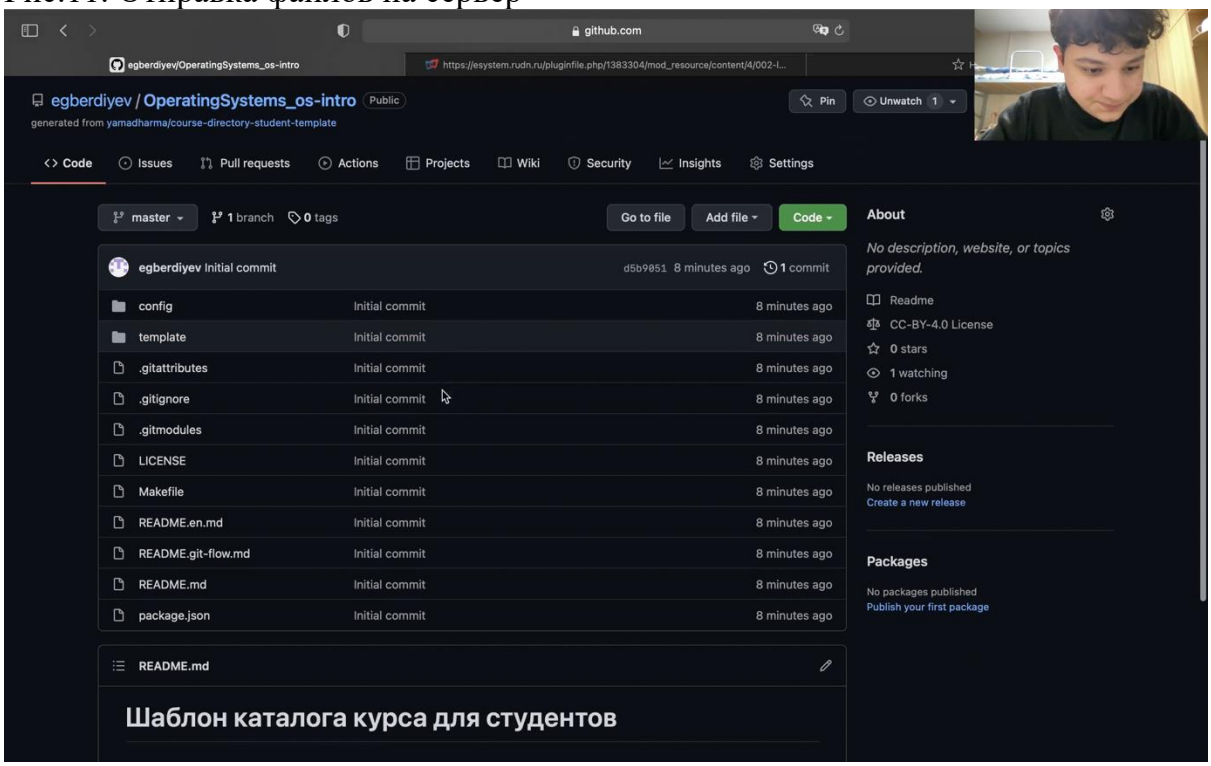

Рис.12. Результат выполненной работы

Контрольные вопросы

1. Система контроля версий *(от англ. Version Control System, VCS)* — это место хранения кода.
Она заточена именно на разработку продуктов. То есть на хранение кода, синхронизацию работы нескольких человек, создание релизов.

2.
1) Репозиторий - хранилище версий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией.

2) Commit представляет собой по сути snapshot файлов, которые были поставлены на commit. В комментариях к commit нужно указывать нужную информацию о причинах commit, например номера тикетов или ссылок на документы, rfc, переписки и прочее. Если коммит длинный – то комментарий к commit не нужно указывать непосредственно в cli, а просто ввести commit. Тогда откроется текстовый редактор, в нем первой строкой краткое описание, а далее подробный текст изменения.

3) При добавлении файла для отслеживания в SCCS создаётся файл специального типа, который называется s-файл или файл истории. Он именуется как исходный файл, только с префиксом s., и хранится в подкаталоге SCCS. Таким образом, для файла test.txt будет создан файл истории s.test.txt в директории ./SCCS/. В момент создания файл истории содержит начальное содержимое исходного файла, а также некоторые метаданные, помогающие отслеживать версии. Здесь хранятся контрольные суммы для гарантии, что содержимое не было изменено. Содержимое файла истории не сжимается и не кодируется.

4) Для работы с содержимым репозитория каждый разработчик имеет собственную рабочую копию. Рабочая копия- «снимок» содержимого репозитория, плюс некоторая служебная информация.

3.

1) Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере. Доступ к нему осуществлялся через специальное клиентское приложение. В качестве примеров таких программных продуктов можно привести CVS, Subversion.

2) Распределенные системы контроля версий (Distributed Version Control System, DVCS) позволяют хранить репозиторий (его копию) у каждого разработчика, работающего с данной системой. При этом можно выделить центральный репозиторий (условно), в который будут отправляться изменения из локальных и, с ним же эти локальные репозитории будут синхронизироваться. При работе с такой системой, пользователи периодически синхронизируют свои локальные репозитории с центральным и работают непосредственно со своей локальной копией. После внесения достаточного количества изменений в локальную копию они (изменения) отправляются на сервер. При этом сервер, чаще всего, выбирается условно, т.к. в большинстве DVCS нет такого понятия как “выделенный сервер с центральным репозиторием”. Две наиболее известные DVCS – это Git и Mercurial.

4.

Хранилище является разновидностью файл-сервера, однако не совсем обычного.

• Хранилище Subversion запоминает каждое внесенное изменение:

-любое изменение любого файла,

-изменения в самом дереве каталогов, такие как добавление, удаление и реорганизация файлов и каталогов.

• При чтении данных из хранилища клиент обычно видит только последнюю версию дерева файлов.

• Клиент также имеет возможность просмотреть предыдущие состояния файловой системы.

• Вопросы типа

«Что содержал этот каталог в прошлую среду?», «Кто был последним, изменявшим этот файл, и какие вносились изменения?»

являются основополагающими для любой системы управления версиями — системы, разработанной для записи и отслеживания изменений информации во времени.

5. Первым действием, которое должен выполнить разработчик, является извлечение рабочей копии проекта или той его части, с которой предстоит работать. Это действие выполняется с помощью стандартной команды извлечения версии (checkout или clone) либо специальной команды, фактически выполняющей то же самое действие. Разработчик задаёт версию, которая должна быть скопирована, по умолчанию обычно копируется последняя (или выбранная администратором в качестве основной) версия.

По команде извлечения устанавливается соединение с сервером, и проект в виде дерева каталогов и файлов копируется на компьютер разработчика. Обычной практикой является дублирование рабочей копии: помимо основного каталога с проектом на локальный диск дополнительно записывается ещё одна его копия. Работая с проектом, разработчик изменяет только файлы основной рабочей копии. Вторая локальная копия хранится в качестве эталона, позволяя в любой момент без обращения к серверу определить, какие изменения внесены в конкретный файл или проект в целом и от какой версии была сделана рабочая копия; как правило, любая попытка ручного изменения этой копии приводит к ошибкам в работе программного обеспечения СКВ.

6. Git позволяет фиксировать и совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется.

7. Наиболее часто используемые команды git:

Создание основного дерева репозитория:

git init

получение обновлений (изменений) текущего дерева из центрального репозитория:

git pull

отправкавсехпроизведённыхизмененийлокальногодеревавцентральныйрепози- торий:

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. Пример можно взять из лабораторной работы

Создадим локальный репозиторий.

Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория:

и настроив utf-8 в выводе сообщений git: git config --global quotepath false

git add ...

git rm ...

git config --global user.name "Имя Фамилия"

git config --global user.email "work@mail"

Для инициализации локального репозитория, расположенного, например, в каталоге ~/tutorial, необходимо ввести в командной строке:

cd

mkdir tutorial

cd tutorial

git init

После это в каталоге tutorial появится каталог .git, в котором будет храниться история изменений.

Создадим тестовый текстовый файл hello.txt и добавим его в локальный репозито- рий:

Воспользуемся командой status для просмотра изменений в рабочем каталоге, сделанных с момента последней ревизии:

git status

Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов. Для этого сначала нужно получить список имеющихся шаблонов:

curl -L -s https://www.gitignore.io/api/list

Затем скачать шаблон, например, для C и C++

curl -L -s https://www.gitignore.io/api/c >> .gitignore

curl -L -s https://www.gitignore.io/api/c++ >> .gitignore

9. Ветка в Git — это простой перемещаемый указатель на один из таких коммитов. По умолчанию, имя основной ветки в Git — master. Как только вы начнёте создавать коммиты, ветка master будет всегда указывать на последний коммит. Каждый раз при создании коммита указатель ветки master будет передвигаться на следующий коммит автоматически.

10. Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты.

Проект часто создает файлы которые вы не хотите отслеживать с помощью git. Это обычно включает файлы генерируемые процессом сборки или временные файлы созданные вашим редактором. Конечно, понятие неотслеживаемые файлы git-ом означает что они не будут обрабатываться при выполнении git add. Но это быстро начинает раздражать, когда вокруг лежат неотслеживаемые файлы; например они делают git add . и git commit -a практически бесполезными, и они могут содержаться в выводе команды "git status".

Вы можете указать git игнорировать определенные файлы создав файл .gitignore на самом верхнем уровне рабочей директории.

Вывод: изучил идеологию и применение средств контроля версий. Освоил умения по работе git.