

Лабораторная работа 2

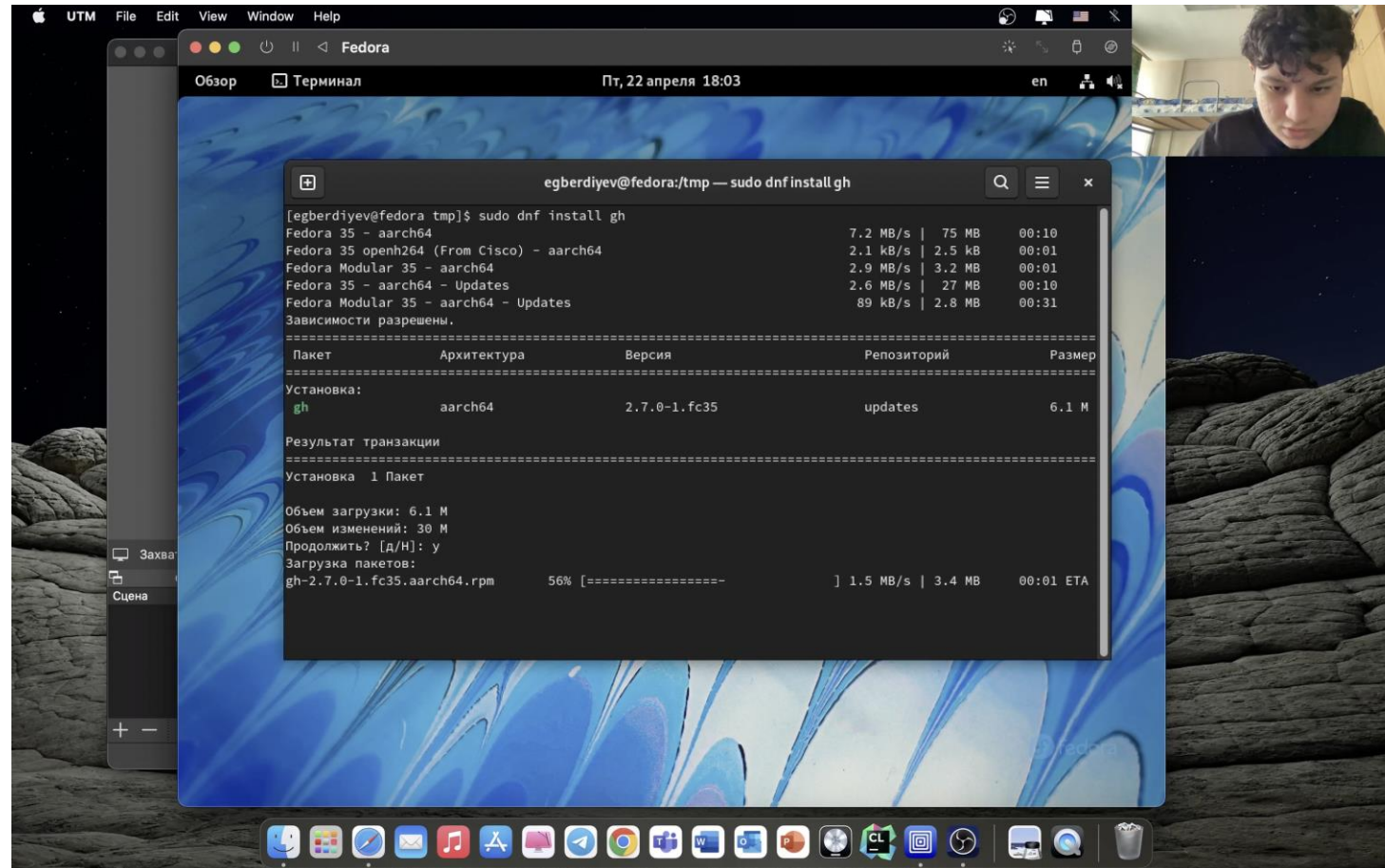
Бердыев Эзиз

НФИбд-01-21

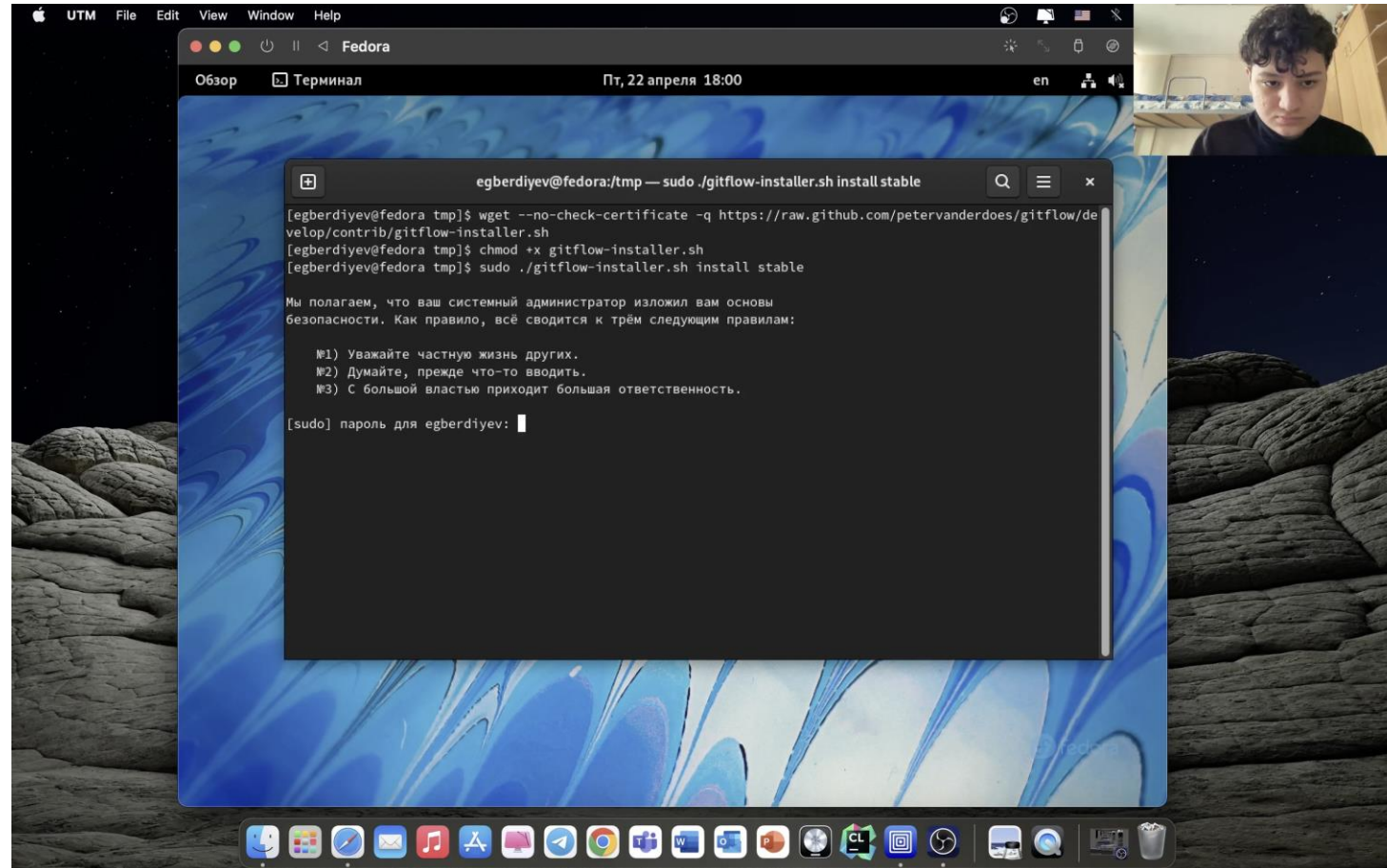
Цель работы

- – Изучить идеологию и применение средств контроля версий.
- – Освоить умения по работе с git.

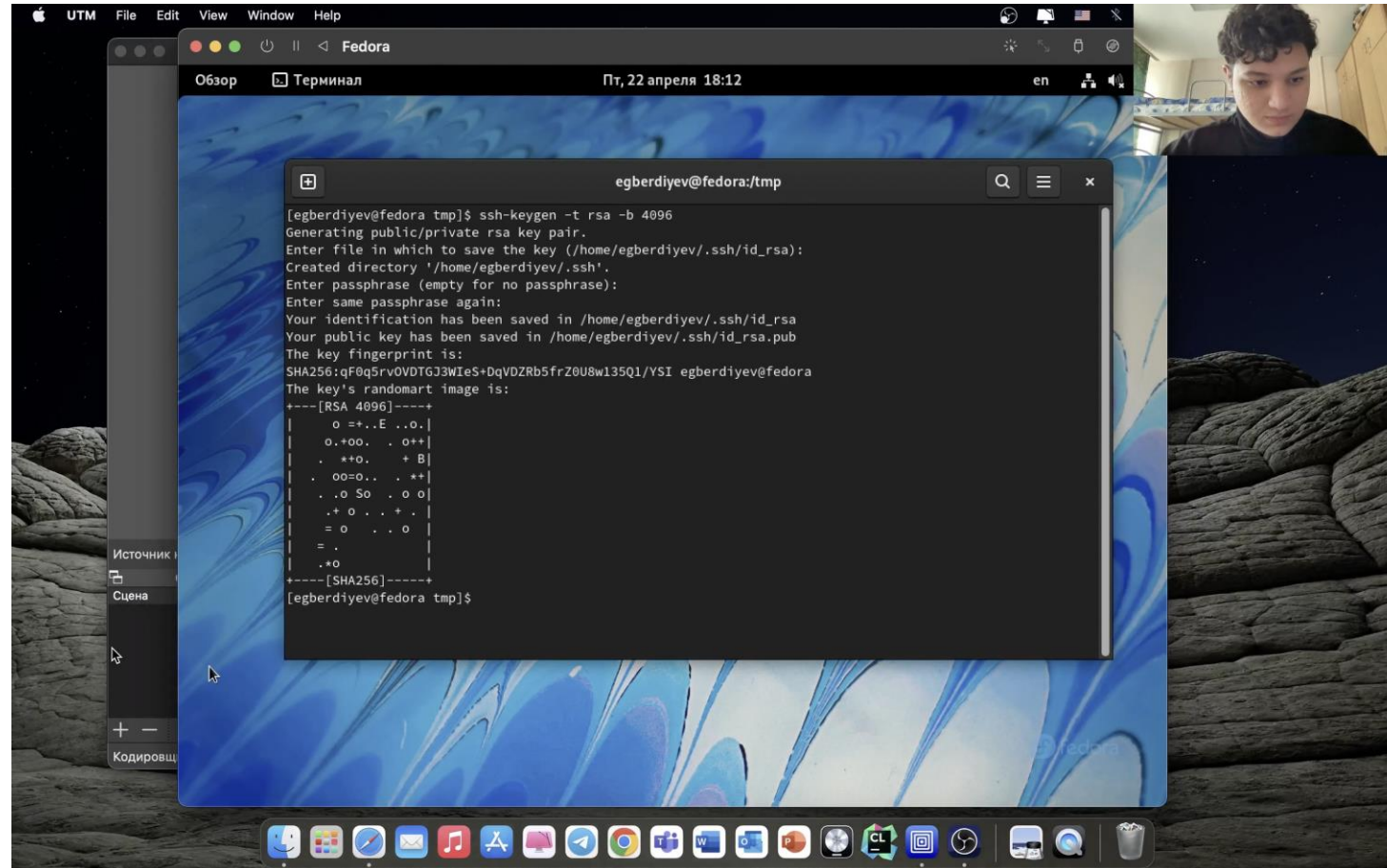
Установка git-flow в Fedora Linux



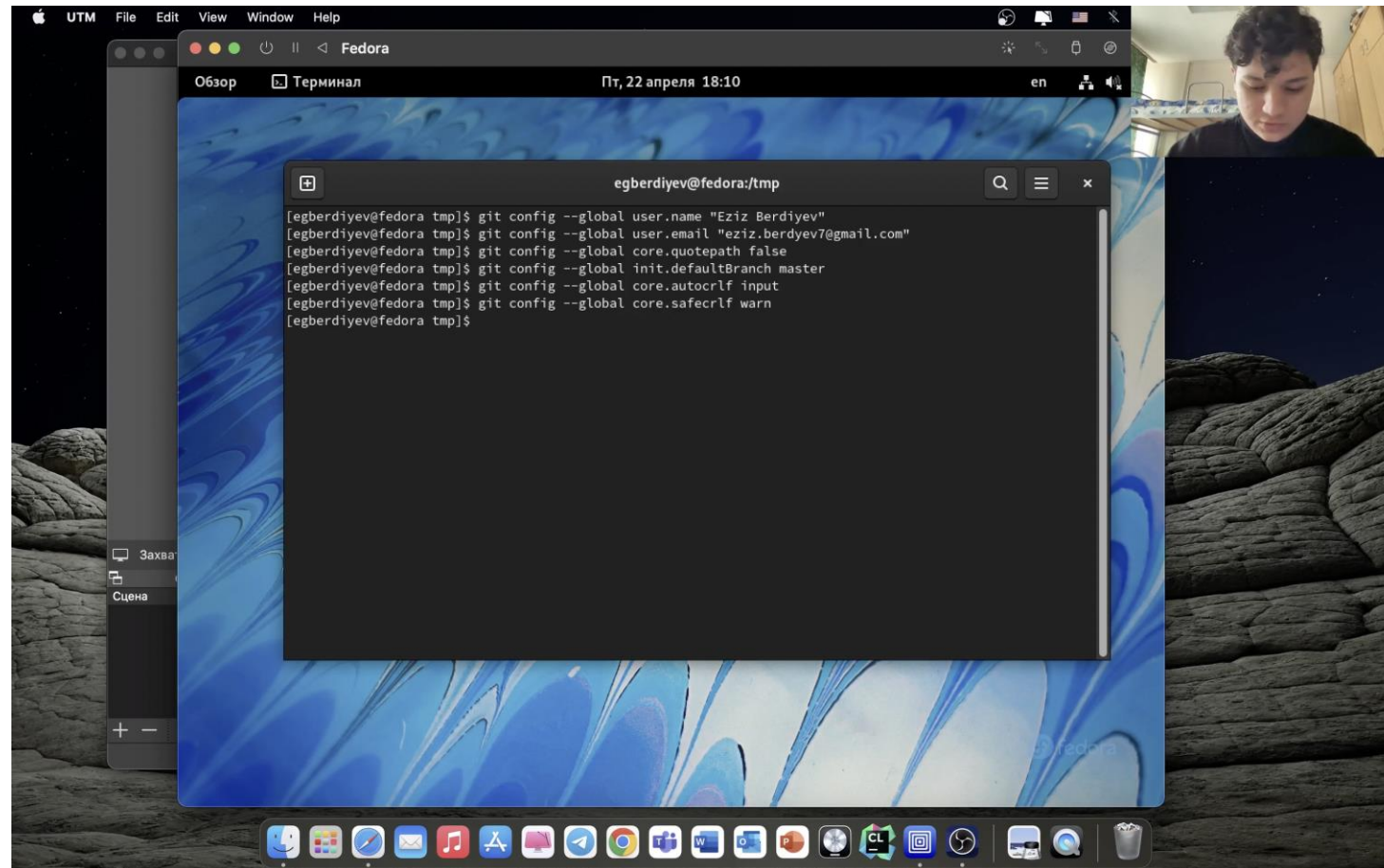
Установка gh в Fedora Linux



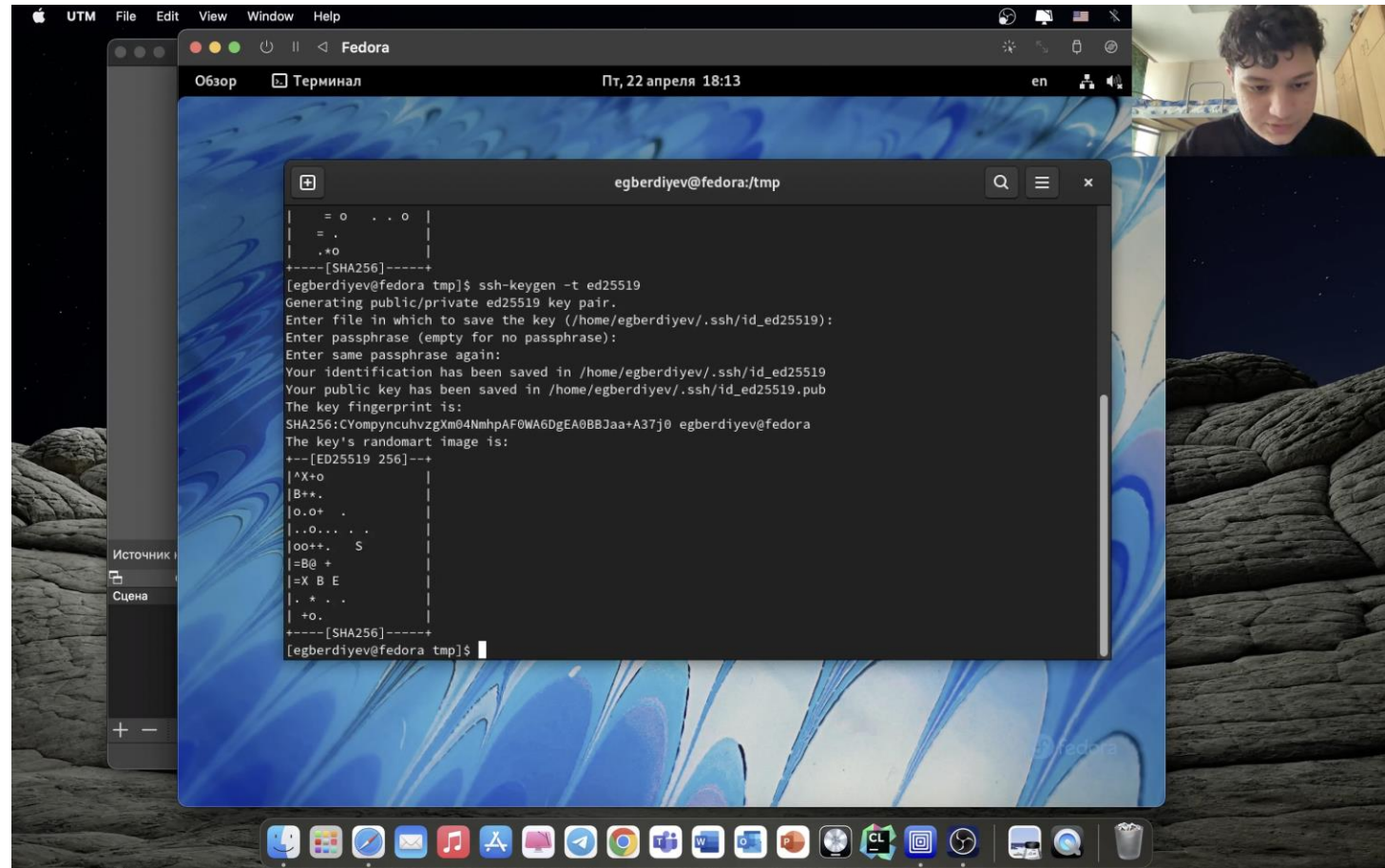
Базовая настройка в git



Создание ssh ключей

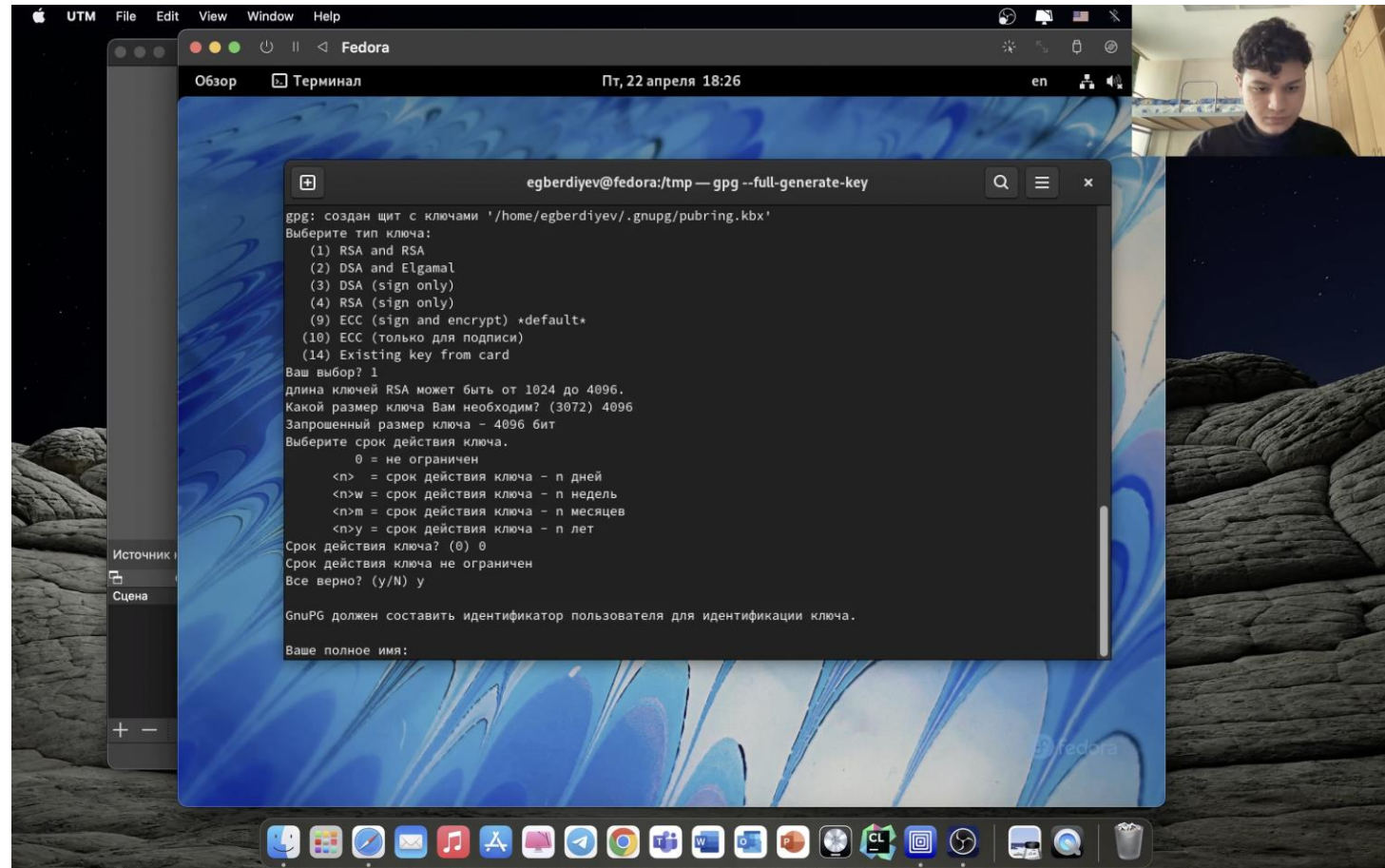


По алгоритму rsa и ed25519

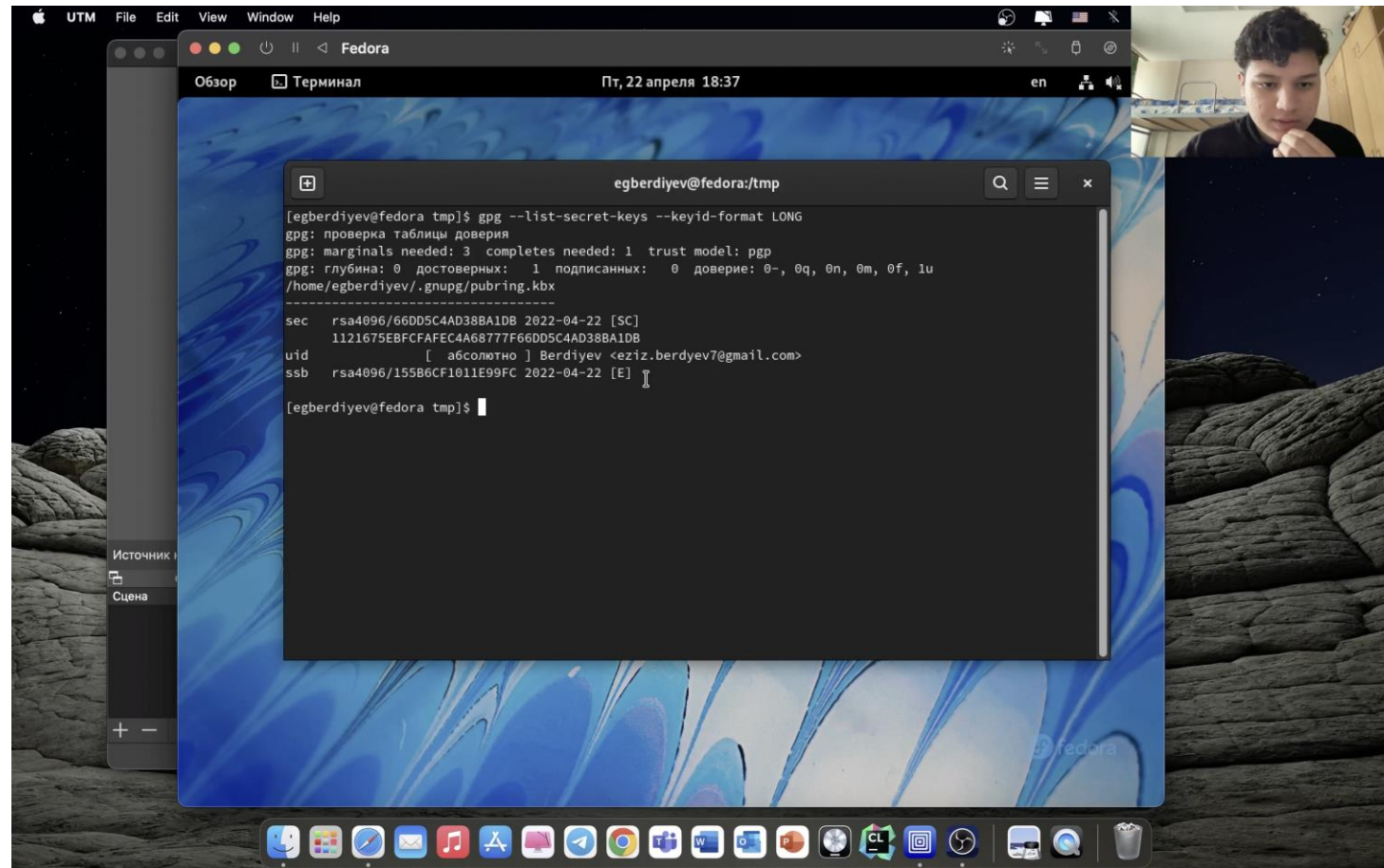


```
egberdiyev@fedora:~/tmp$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/egberdiyev/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/egberdiyev/.ssh/id_ed25519
Your public key has been saved in /home/egberdiyev/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:CYompyncuhvzgXm04NmhpAF0WA6DgEA0BBJaa+A37j0 egberdiyev@fedora
The key's randomart image is:
+--[ED25519 256]--+
|  ^X+o              |
| 8+*               |
|o.o+              |
|..O...           |
|oo++   S          |
|=B@ +             |
|=X B E            |
|. + .             |
|+o.               |
+-----[SHA256]-----+
[egberdiyev@fedora tmp]$
```

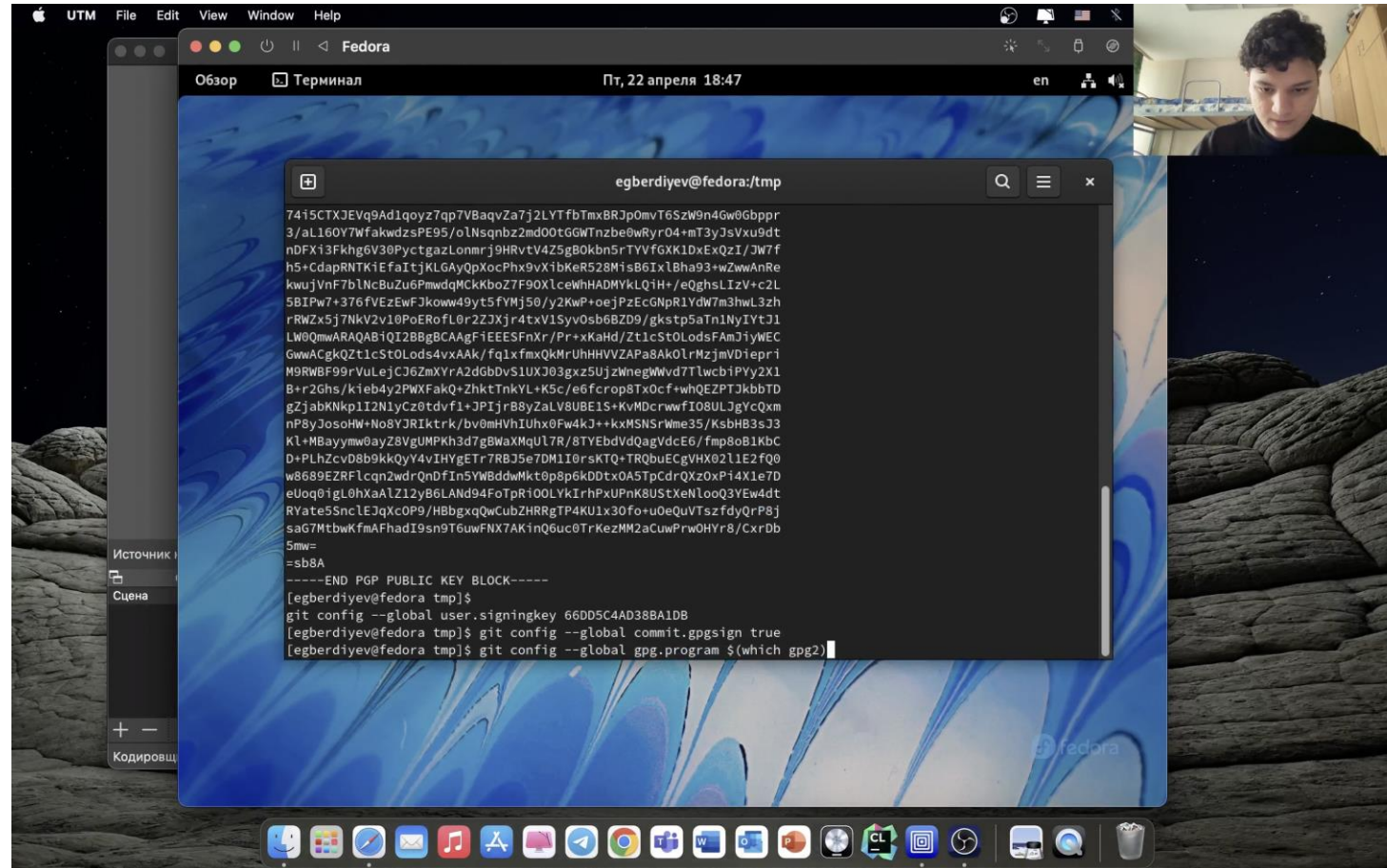
Создание pgp ключа



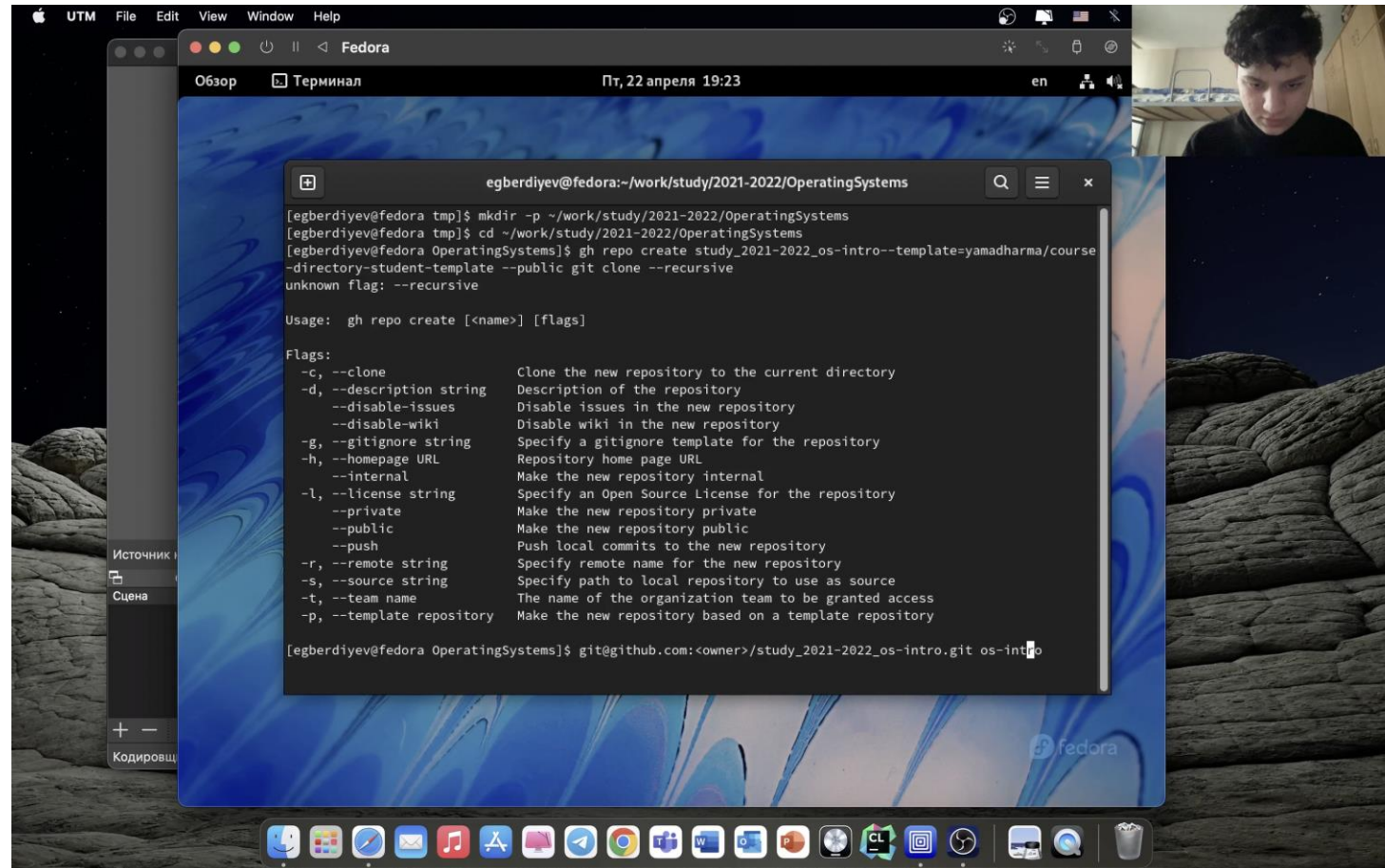
Добавление pgp ключа в GitHub



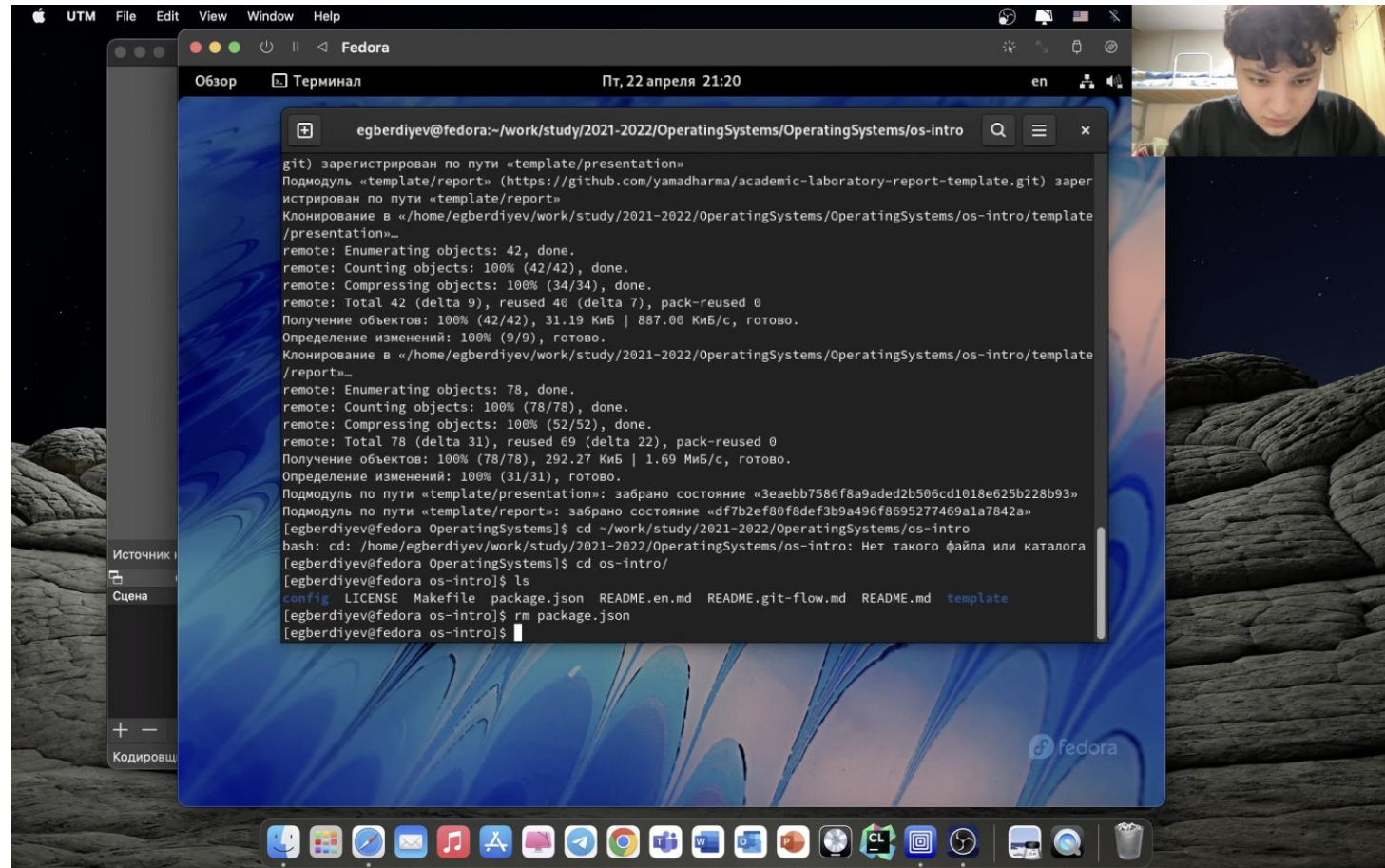
Настройка автоматических подписей коммитов git



Создание репозиториев на основе шаблона

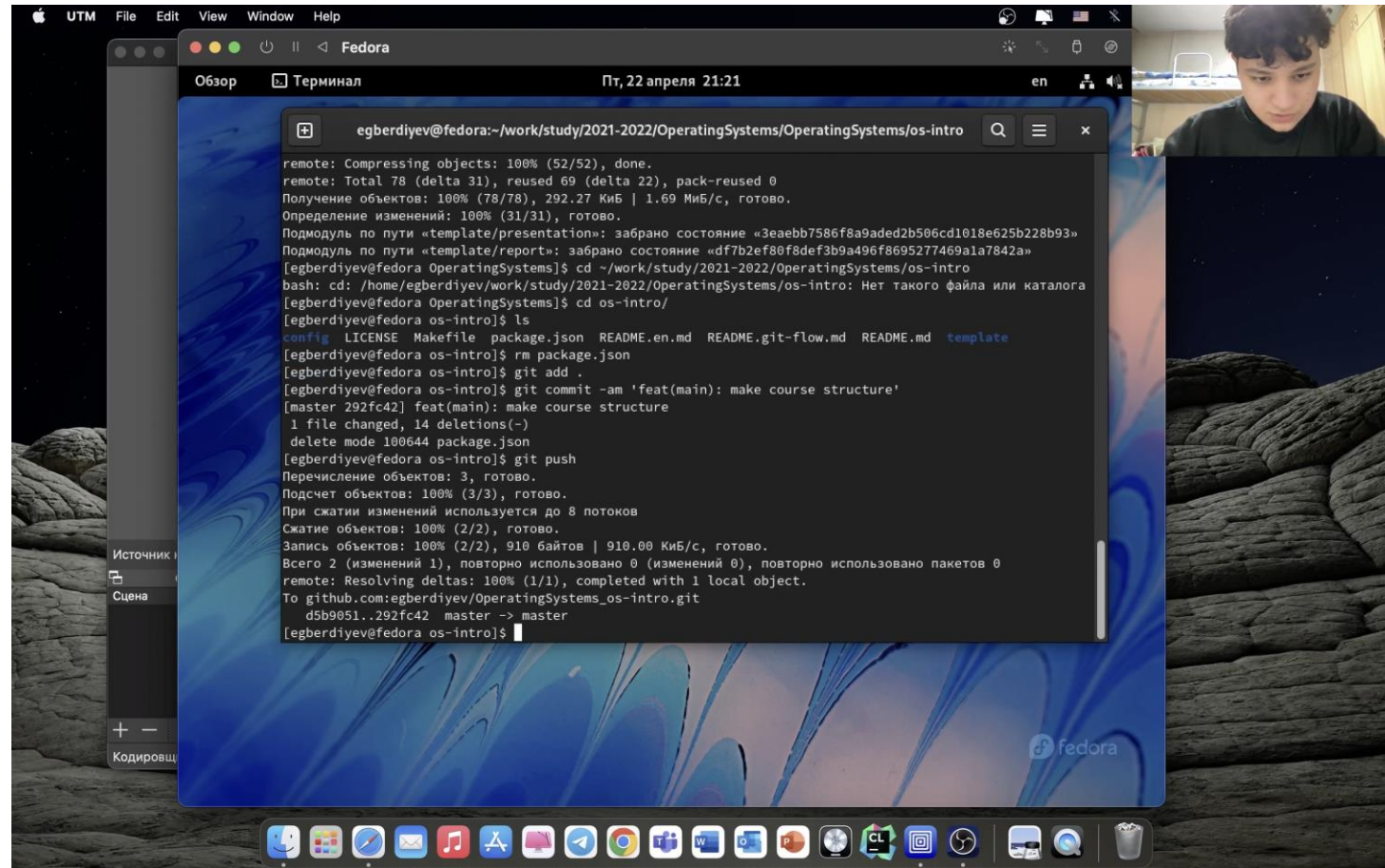


Настройка каталога курса и удаление лишних файлов



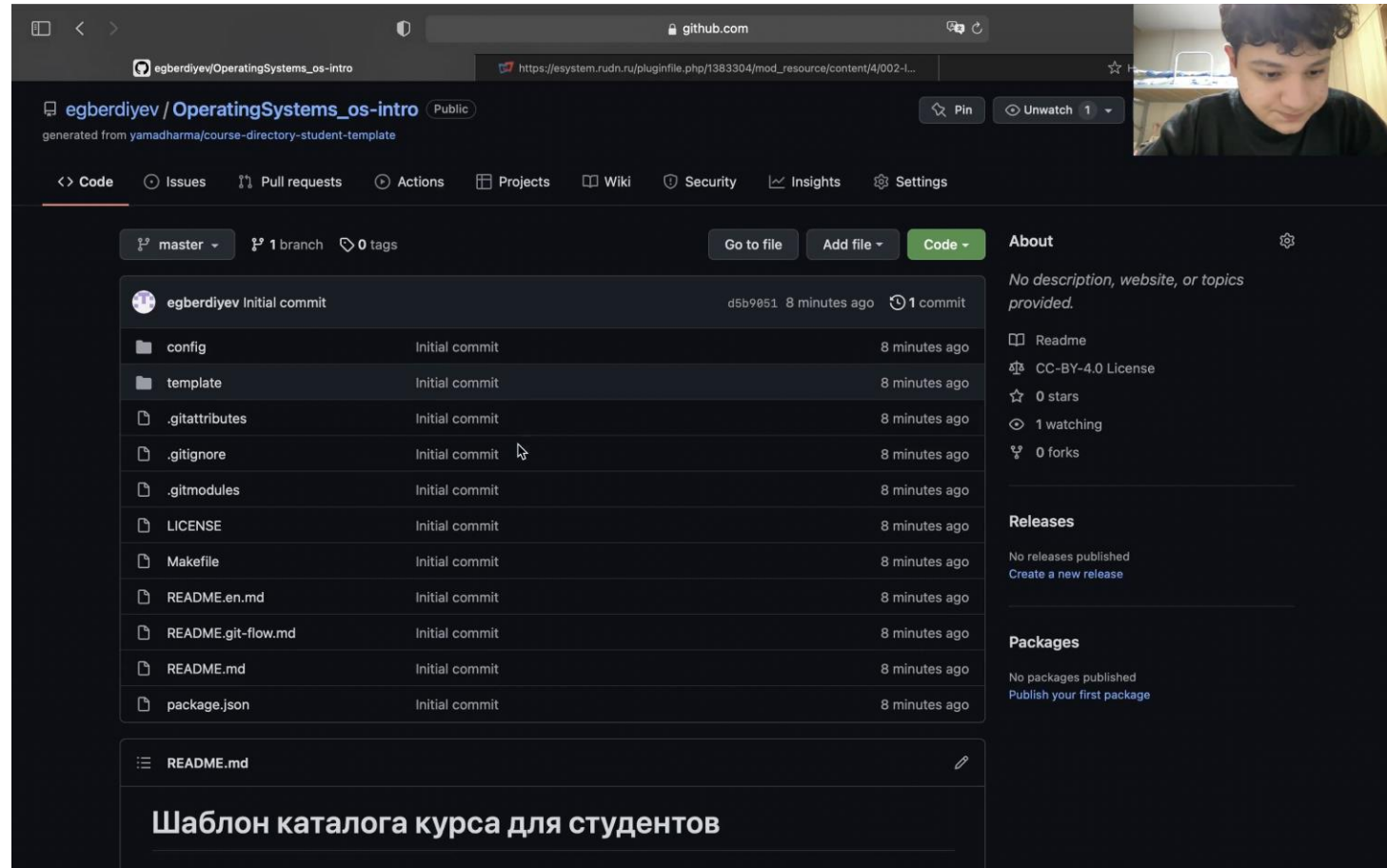
```
egberdiyev@fedora:~/work/study/2021-2022/OperatingSystems/OperatingSystems/os-intro
git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) запер
истрирован по пути «template/report»
Клонирование в «/home/egberdiyev/work/study/2021-2022/OperatingSystems/OperatingSystems/os-intro/template
/presentation»...
remote: Enumerating objects: 42, done.
remote: Counting objects: 100% (42/42), done.
remote: Compressing objects: 100% (34/34), done.
remote: Total 42 (delta 9), reused 40 (delta 7), pack-reused 0
Получение объектов: 100% (42/42), 31.19 КиБ | 887.00 КиБ/с, готово.
Определение изменений: 100% (9/9), готово.
Клонирование в «/home/egberdiyev/work/study/2021-2022/OperatingSystems/OperatingSystems/os-intro/template
/report»...
remote: Enumerating objects: 78, done.
remote: Counting objects: 100% (78/78), done.
remote: Compressing objects: 100% (52/52), done.
remote: Total 78 (delta 31), reused 69 (delta 22), pack-reused 0
Получение объектов: 100% (78/78), 292.27 КиБ | 1.69 МиБ/с, готово.
Определение изменений: 100% (31/31), готово.
Подмодуль по пути «template/presentation»: забрано состояние «3eae7586f8a9aded2b506cd1018e625b228b93»
Подмодуль по пути «template/report»: забрано состояние «df7b2ef80f8def3b9a496f8695277469ala7842a»
[egberdiyev@fedora OperatingSystems]$ cd ~/work/study/2021-2022/OperatingSystems/os-intro
bash: cd: /home/egberdiyev/work/study/2021-2022/OperatingSystems/os-intro: Нет такого файла или каталога
[egberdiyev@fedora OperatingSystems]$ cd os-intro/
[egberdiyev@fedora os-intro]$ ls
config LICENSE Makefile package.json README.en.md README.git-flow.md README.md template
[egberdiyev@fedora os-intro]$ rm package.json
[egberdiyev@fedora os-intro]$
```

Отправка файлов на сервер



```
egberdiyev@fedora:~/work/study/2021-2022/OperatingSystems/OperatingSystems/os-intro
remote: Compressing objects: 100% (52/52), done.
remote: Total 78 (delta 31), reused 69 (delta 22), pack-reused 0
Получение объектов: 100% (78/78), 292.27 КиБ | 1.69 МиБ/с, готово.
Определение изменений: 100% (31/31), готово.
Подмодуль по пути «template/presentation»: забрано состояние «3eae7b7586f8a9aded2b506cd1018e625b228b93»
Подмодуль по пути «template/report»: забрано состояние «df7b2ef80f8def3b9a496f8695277469a1a7842a»
[egberdiyev@fedora OperatingSystems]$ cd ~/work/study/2021-2022/OperatingSystems/os-intro
bash: cd: /home/egberdiyev/work/study/2021-2022/OperatingSystems/os-intro: Нет такого файла или каталога
[egberdiyev@fedora OperatingSystems]$ cd os-intro/
[egberdiyev@fedora os-intro]$ ls
config LICENSE Makefile package.json README.en.md README.git-flow.md README.md template
[egberdiyev@fedora os-intro]$ rm package.json
[egberdiyev@fedora os-intro]$ git add .
[egberdiyev@fedora os-intro]$ git commit -am 'feat(main): make course structure'
[master 292fc42] feat(main): make course structure
1 file changed, 14 deletions(-)
delete mode 100644 package.json
[egberdiyev@fedora os-intro]$ git push
Перечисление объектов: 3, готово.
Подсчет объектов: 100% (3/3), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (2/2), 910 байтов | 910.00 КиБ/с, готово.
Всего 2 (изменений 1), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:egberdiyev/OperatingSystems_os-intro.git
d5b9051..292fc42 master -> master
[egberdiyev@fedora os-intro]$
```


Результат выполненной работы



The screenshot shows a GitHub repository page for 'egberdiyev/OperatingSystems_os-intro'. The repository is public and was generated from 'yamadharm/course-directory-student-template'. The page displays a list of files and folders, all of which were committed initially 8 minutes ago. The files include 'config', 'template', '.gitattributes', '.gitignore', '.gitmodules', 'LICENSE', 'Makefile', 'README.en.md', 'README.git-flow.md', 'README.md', and 'package.json'. The 'README.md' file is selected, showing its content: 'Шаблон каталога курса для студентов'. The right sidebar shows the repository's metadata, including no description, website, or topics provided, a CC-BY-4.0 license, 0 stars, 1 watching, and 0 forks. There are no releases or packages published yet.

egberdiyev / OperatingSystems_os-intro Public

generated from yamadharm/course-directory-student-template

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags

Go to file Add file Code

egberdiyev Initial commit d5b9051 8 minutes ago 1 commit

config	Initial commit	8 minutes ago
template	Initial commit	8 minutes ago
.gitattributes	Initial commit	8 minutes ago
.gitignore	Initial commit	8 minutes ago
.gitmodules	Initial commit	8 minutes ago
LICENSE	Initial commit	8 minutes ago
Makefile	Initial commit	8 minutes ago
README.en.md	Initial commit	8 minutes ago
README.git-flow.md	Initial commit	8 minutes ago
README.md	Initial commit	8 minutes ago
package.json	Initial commit	8 minutes ago

README.md

Шаблон каталога курса для студентов

About

No description, website, or topics provided.

- Readme
- CC-BY-4.0 License
- 0 stars
- 1 watching
- 0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

1 вопрос

- 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 вопрос

- 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 вопрос

- 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 осуществляется 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 вопрос

- 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 вопрос

- 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 вопрос

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

7 вопрос

- Наиболее часто используемые команды git:
- Создание основного дерева репозитория:
- `git init`
- получение обновлений (изменений) текущего дерева из центрального репозитория:
- `git pull`
- отправкавсехпроизведённыхизмененийлокальногодеревавцентральныйрепозиторий:
- `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 вопрос

- 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 репозиторий:

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 вопрос

- 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 вопрос

- 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 -а 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.