РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

Дисциплина: *Операционные системы*

Студент: Алламе Ормиз

Группа: НФИбд-01-21

МОСКВА

2022 г.

Цель работы

– Изучить идеологию и применение средств контроля версий.

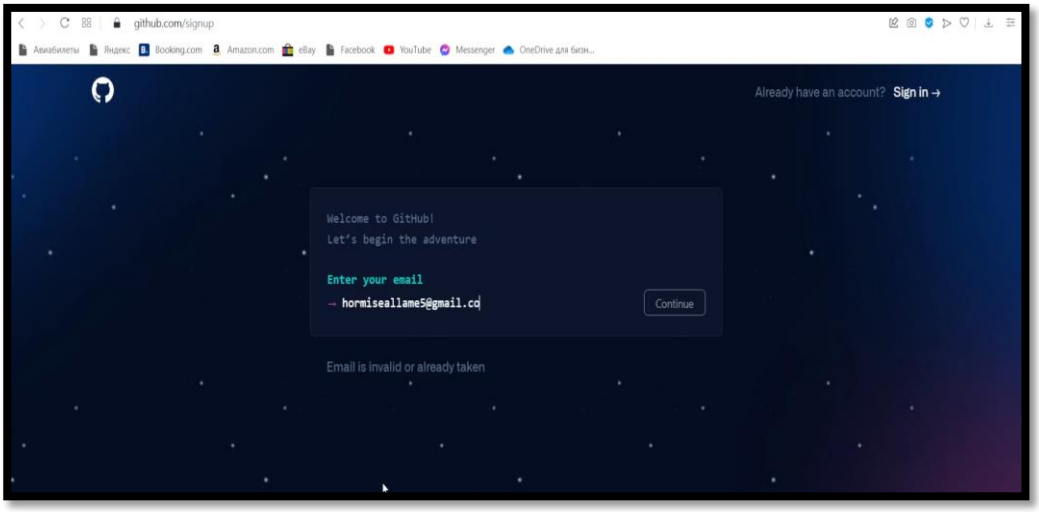
– Освоить умения по работе с git.

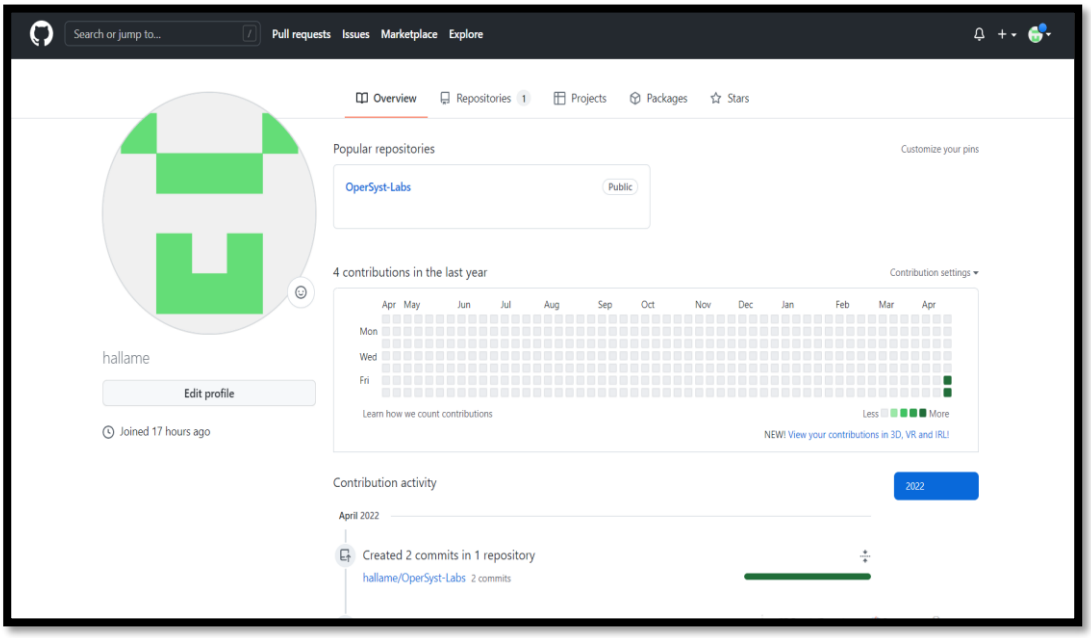
Ход работы

1. Настройка github

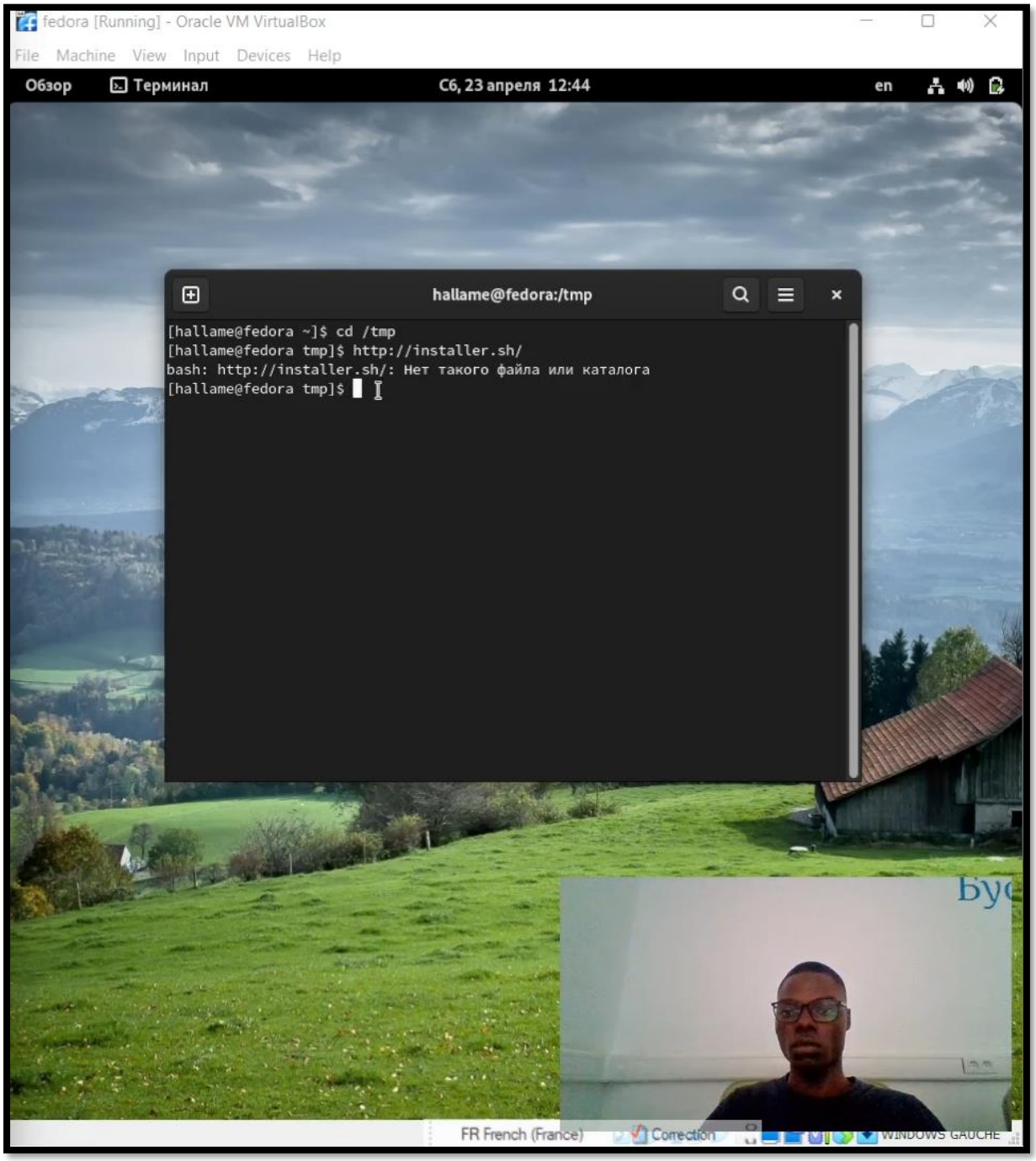
Создайте учётную запись на https://github.com.

Заполните основные данные на https://github.com.

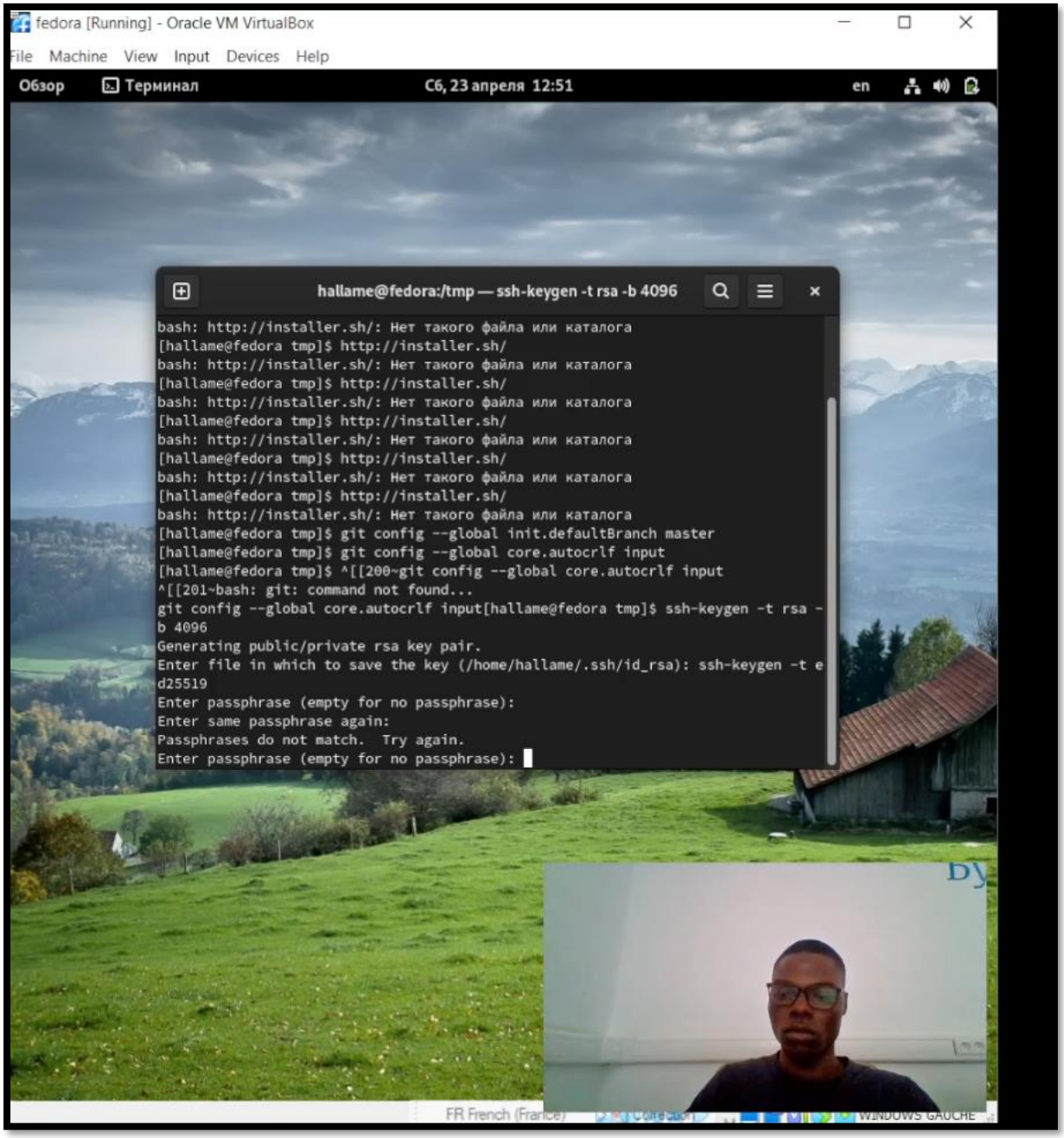




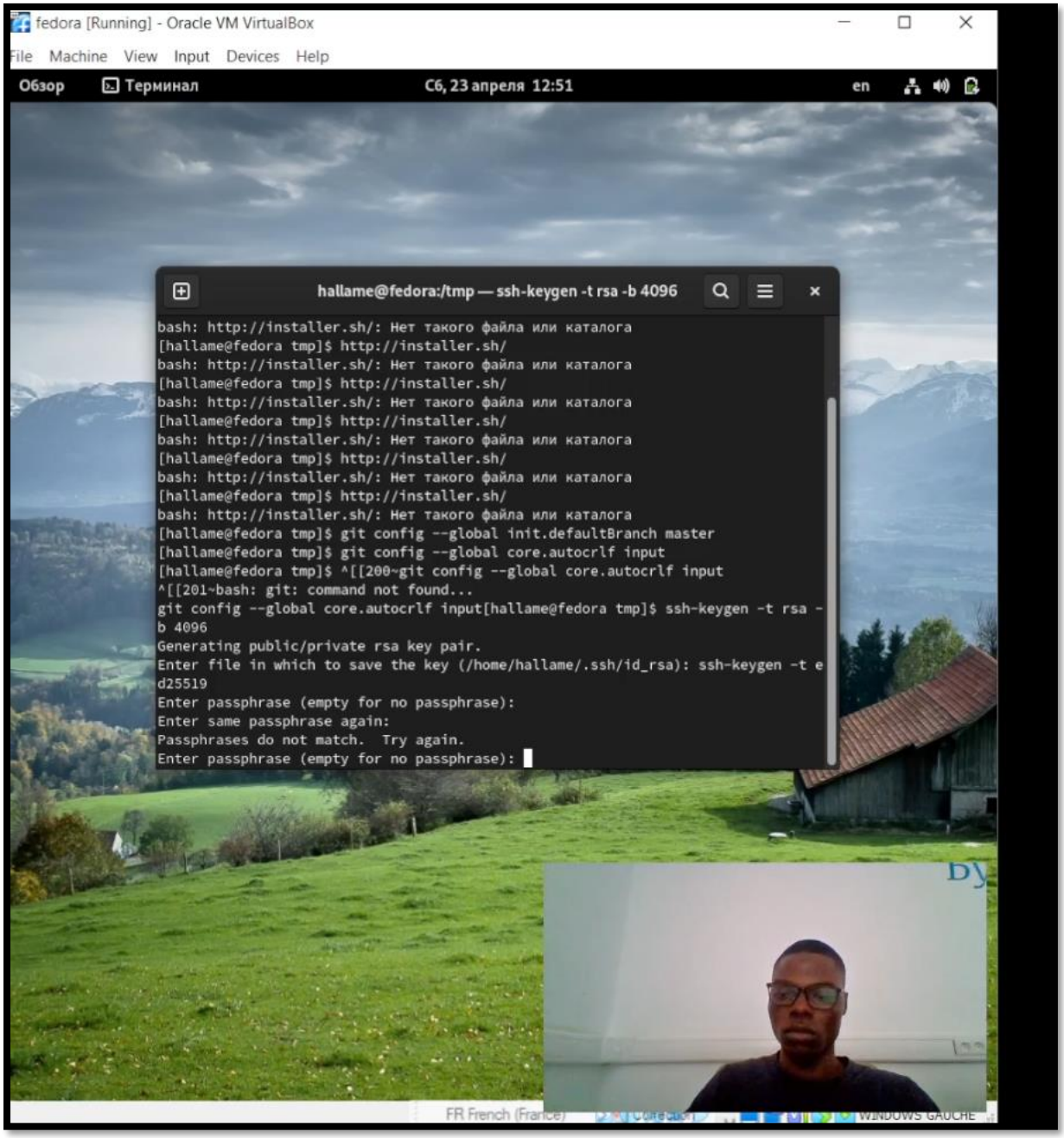
2. **Установка программного обеспечения**
3. **Базовая настройка git**



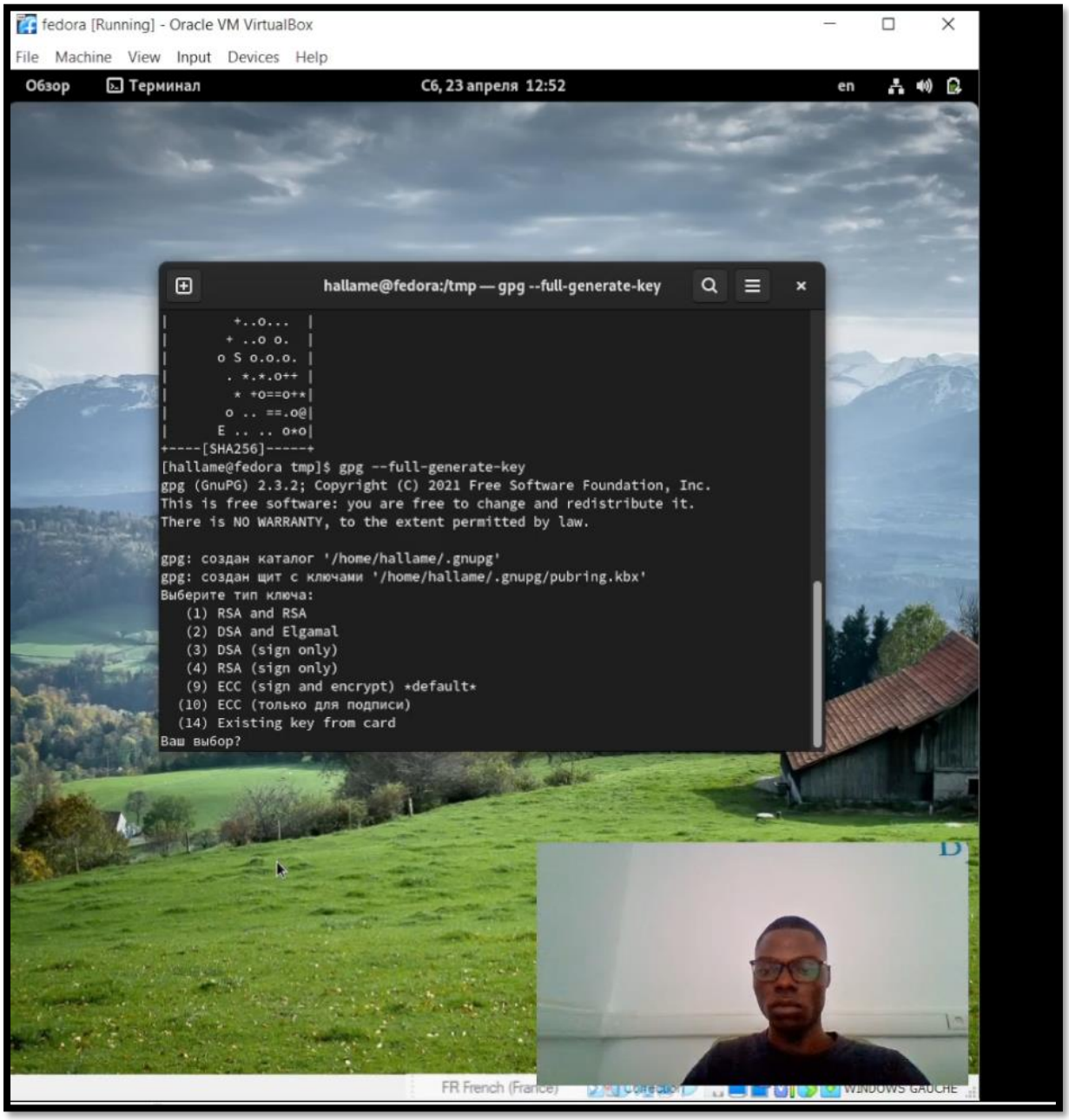
4. Создайте ключи *ssh*



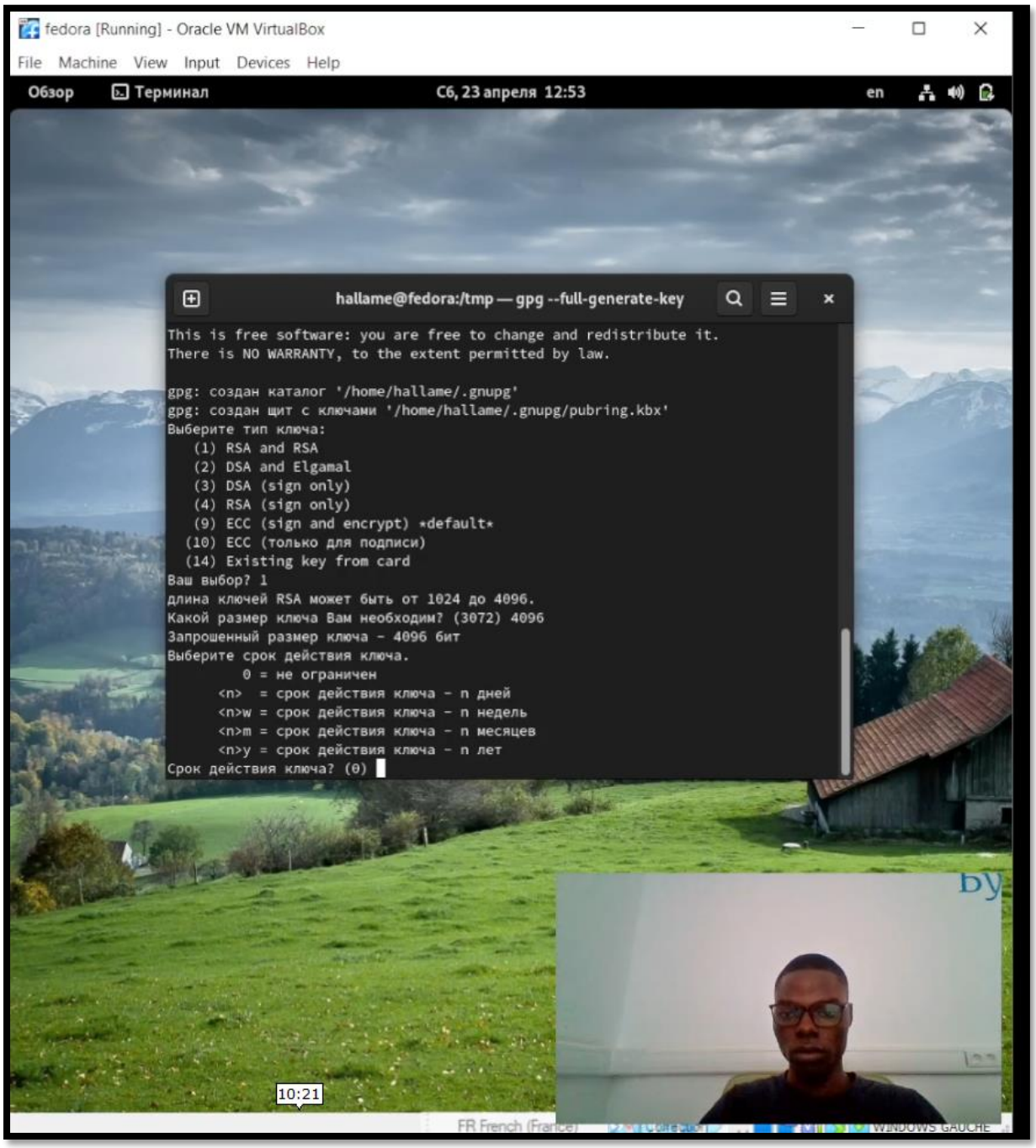
5. Создайте ключи *pgp*



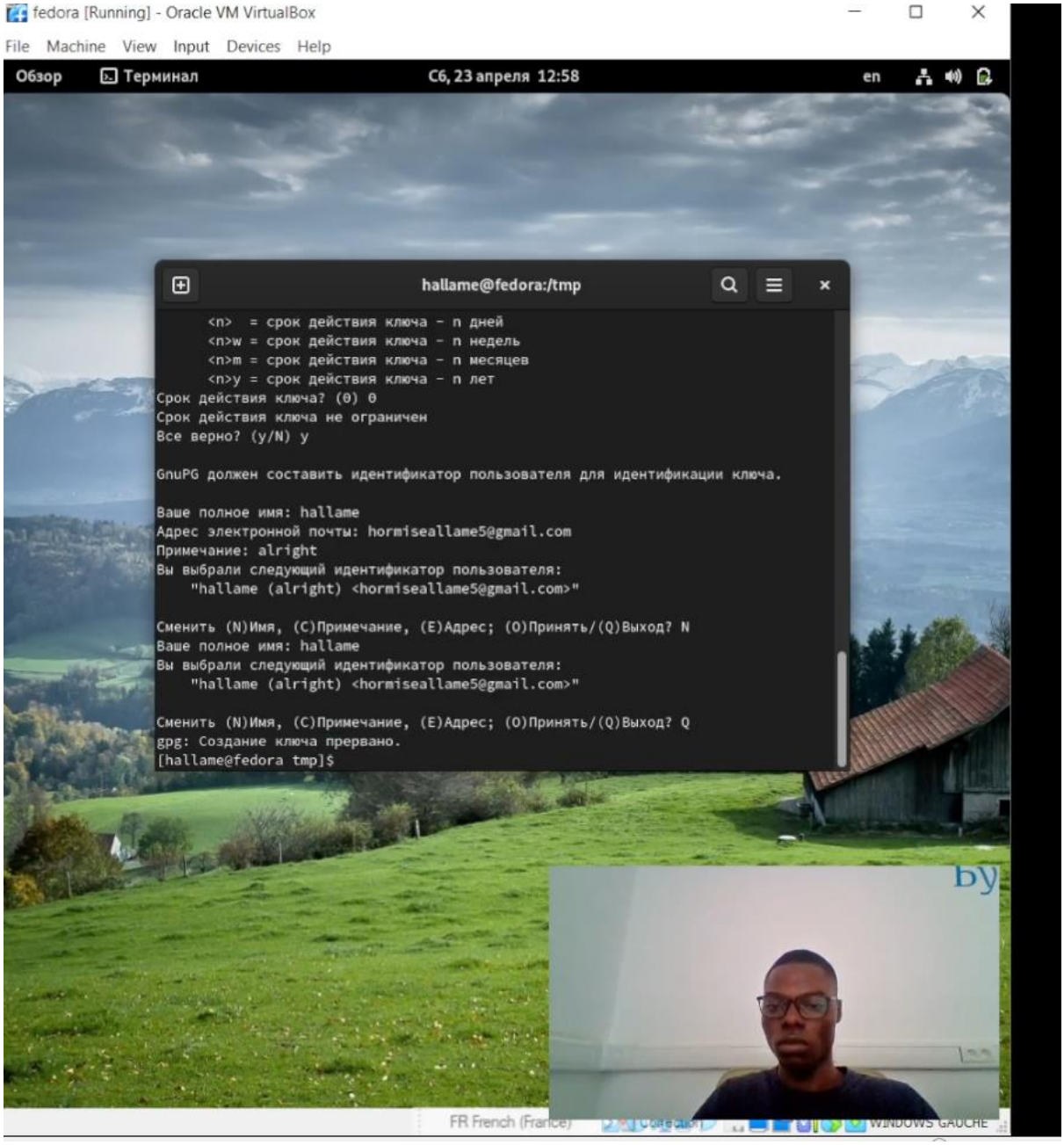
6. **Добавление PGP ключа в GitHub**
7. **Настройка автоматических подписей коммитов git**



8. Настройка gh



9. Шаблон для рабочего пространства



Выводы

Системы контроля версиями записывают и сохраняют несколько изменений в файлах. Благодаря этому можно вернуться к определенной точке истории изменения файла или проекта. Некоторые системы, такие как Subversion, отслеживают историю отдельных файлов. Другие, такие как Git и Mercurial, отслеживают историю целых репозиториев.

Ответы на контрольные вопросы

1. Система контроля версий, также называемая системой управления версиями, относится к практике отслеживания изменений в коде программного обеспечения и управления ими. Это программные инструменты, которые позволяют командам разработчиков управлять изменениями исходного кода с течением времени. Благодаря ускорению среды разработки системы контроля версий помогают командам разработчиков работать быстрее и умнее. Они особенно полезны для команд DevOps, поскольку позволяют им сократить время разработки и обеспечить успешное развертывание.

2.

Хранилище (repository), или репозитарий, — место **хранения** всех версий и служебной информации

Commit представляет собой по сути snapshot файлов, которые были поставлены на commit. В комментариях к commit нужно указывать нужную информацию о причинах commit

История остается линейной, ветвление исчезает. Она искажается: изменяются коммиты **и их** порядок..

Рабочая копия (working copy) – полученное из репозитория дерево каталогов проекта некой версии (текущей или прошлой)

Вся история всегда в едином общем хранилище. Нужно подключение к сети. История не содержит еще коммитов. ... добавлен новый коммит (а заодно, меняет состояние рабочей копии.

3.

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*.

Децентрализованные 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”.

4. В начале, разработчик работает с веткой master. При реализации отдельных частей проекта можно создавать филиалы для них. Когда изменения будут завершены, разработчик фиксирует и просто отправляет изменения на сервер.

5. Каждый разработчик работает в своем филиале над отдельной частью проекта. Если в одной ветке работает несколько разработчиков, они обычно получают изменения, внесенные другими, и уже работают с ними.

6. Git (читается как «гит») — это **система контроля версий**, которая помогает отслеживать историю изменений в файлах. Git используют программисты для совместной работы над проектами. Git помогает команде разработчиков управлять изменениями, внесенными в исходный код с течением времени. Git позволяет отслеживать дополнения и изменения в исходном коде на графике.

Таким образом, если будет допущена ошибка, разработчики могут вернуться и сравнить более ранние версии кода, что позволит им исправить ошибку, минимизируя сбои для всех членов команды.

7. Просмотреть список текущих веток можно с помощью команды branch. Звездочка (*) появится рядом с текущей активной веткой.

git branch

Попробуйте создать новую ветку. Вы останетесь в текущей активной ветке, пока не переключитесь в новую.

git branch new-branch

Перейдите в любую существующую ветку и проверьте ее в текущем рабочем каталоге.

git checkout another-branch

Вы можете объединить создание и проверку новой ветки, используя флаг -b.

git checkout -b new-branch

Переименуйте ветку.

git branch -m current-branch-name new-branch-name

Объедините историю указанной ветки с той, в которой вы сейчас работаете.

git merge branch-name

Прервите слияние, если возникнут конфликты.

git merge –abort

Вы также можете выбрать конкретный коммит для объединения с помощью команды cherry-pick. В команде нужно указать строку, которая ссылается на конкретный коммит.

git cherry-pick f7649d0

Если после слияния ветка вам больше не нужна, вы можете удалить ее.

git branch -d branch-name

Если вы не слили ветку с master, но уверены, что хотите удалить ее, вы можете принудительно удалить ветку.

git branch -D branch-name

Совместная разработка и обновление

Чтобы загрузить изменения из другого каталога, например, из удаленного upstream, используйте команду:

git fetch upstream

Объедините извлеченные коммиты:

git merge upstream/master

Чтобы передать локальные коммиты в ветку удаленного репозитория, введите:

git push origin master

Извлечь и слить все коммиты из удаленной ветки можно с помощью команды:

git pull

Просмотр информации

Чтобы вывести историю коммитов для текущей активной ветки, введите:

git log

Также можно просмотреть коммиты, которые изменили определенный файл. Укажите имя файла (даже если он был переименован).

git log --follow my_script.py

Чтобы просмотреть коммиты конкретной ветки можно использовать следующую команду. Она покажет коммиты в ветке a-branch, которых нет в b-branch.

git log a-branch..b-branch

Чтобы просмотреть логи ссылок (reflog) и увидеть, когда ссылки обновлялись в репозитории последний раз, введите:

git reflog

Вы можете запросить любой объект в Git через строку его коммита или хэш в более удобном для восприятия формате.

git show de754f5

Чтобы скрыть текущую работу, введите:

git stash

Чтобы просмотреть скрытый код:

git stash list

Скрытые изменения будут называться stash@{0}, stash@{1} и так далее.

Вы также можете запросить информацию о конкретном скрытом коде:

git stash show stash@{0}

Чтобы вывести скрытые файлы, сохраняя при этом хранилище, используйте команду apply.

git stash apply stash@{0}

Если вы хотите вывести скрытые файлы и вам больше не нужно хранилище для них, используйте pop.

git stash pop stash@{0}

Если вам больше не нужны файлы, сохраненные в определенном хранилище, вы можете удалить хранилище.

git stash drop stash@{0}

Если у вас есть несколько хранилищ скрытых файлов и одно из них больше не нужно, вы можете удалить его с помощью команды clear.

git stash clear

Если вы хотите сохранить файлы в локальном каталоге Git, но не хотите фиксировать их в проекте, вы можете добавить эти файлы в свой файл .gitignore, чтобы они не вызывали конфликтов.

Чтобы начать перебазирование, нужно указать количество совершенных вами коммитов, которые вы хотите перебазировать (например, 5).

git rebase -i HEAD~5

Также перебазирование можно выполнить на основе конкретной строки коммита или хэша:

git rebase -i 074a4e5

После того, как вы изменили коммиты, вы можете переместить ветку в начало последней версии исходного кода проекта.

git rebase upstream/master

8. Часто временные файлы содержат специфические суффиксы, по которым их легко обнаружить и в последствии удалить

Удаленный репозиторий- Это репозиторий, который хранится в облаке, на сторонних сервисах, специально созданных под работу с проектами git.

- выполняет роль резервной копии
- возможность работать в команде
- некоторые дополнительные возможности, которые предоставляет хостинг. Например, визуализация истории или возможность работать над проектом прямо в веб-интерфейсе

Цель использования удаленного хранилища: резервное копирование, реализация централизованного управления кодом и совместное использование кода.

9. Ветка (англ. branch) — это последовательность коммитов, в которой ведётся параллельная разработка какого-либо функционала Основная ветка– master Ветки в GIT. Показать все ветки, существующие в репозитарии git branch. Создать ветку git branch имя.

Ветки нужны, чтобы несколько программистов могли вести ра боту над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций: чтобы не повредить основному проекту, создается новая ветка специально для экспериментов.

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.

Игнорируемый файл — файл, явным образом помеченный для Git как файл, который необходимо игнорировать. 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.