Лабораторная работа 5

Настройка рабочей среды

Головина Мария Игоревна

Содержание

# 1 Цель работы

Приобрести навыки по настройке рабочей среды.

# 2 Задание

1. Установить необходимые пакеты для работы с менеджером паролей.
2. Настроить ключ gpg.
3. Инициализировать хранилище.
4. Произвести синхронизацию с git.
5. Настроить интерфейс с броузером.
6. Добавить новый пароль.
7. Заменить существующий пароль.
8. Установить дополнительное программное обеспечение.
9. Установить шрифты.
10. Установить бинарный файл.
11. Создать собственный репозиторий с помощью утилит.
12. Подключить репозиторий к своей системе.
13. Использовать chezmoi на нескольких машинах.
14. Произвести настройку новой машины с помощью одной команды.
15. Настроить ежедневные операции с chezmoi.

# 3 Теоретические введения

Менеджер паролей pass

Менеджер паролей pass — программа, сделанная в рамках идеологии Unix. Также носит название стандартного менеджера паролей для Unix (The standard Unix password manager).

Основные свойства

Данные хранятся в файловой системе в виде каталогов и файлов. Файлы шифруются с помощью GPG-ключа.

Структура базы паролей

Структура базы может быть произвольной, если Вы собираетесь использовать её напрямую, без промежуточного программного обеспечения. Тогда семантику структуры базы данных Вы держите в своей голове. Если же необходимо использовать дополнительное программное обеспечение, необходимо семантику заложить в структуру базы паролей.

Семантическая структура базы паролей

Рассмотрим пользователя user в домене example.com, порт 22.

Отсутствие имени пользователя или порта в имени файла означает, что любое имя пользователя и порт будут совпадать:

example.com.pgp

Соответствующее имя пользователя может быть именем файла внутри каталога, имя которого совпадает с хостом. Это полезно, если в базе есть пароли для нескольких пользователей на одном хосте:

example.com/user.pgp

Имя пользователя также может быть записано в виде префикса, отделенного от хоста знаком @:

user@example.com.pgp

Соответствующий порт может быть указан после хоста, отделённый двоеточием (:):

example.com:22.pgp example.com:22/user.pgp user@example.com:22.pgp

Эти все записи могут быть расположены в произвольных каталогах, задающих Вашу собственную иерархию.

Реализации

Утилиты командной строки

На данный момент существует 2 основных реализации: pass — классическая реализация в виде shell-скриптов (https://www.passwordstore.org/); gopass — реализация на go с дополнительными интегрированными функциями (https://www.gopass.pw/). Дальше в тексте будет использоваться программа pass, но всё то же самое можно сделать с помощью программы gopass.

Графические интерфейсы

qtpass qtpass — может работать как графический интерфейс к pass, так и как самостоятельная программа. В настройках можно переключаться между использованием pass и gnupg.

gopass-ui gopass-ui — интерфейс к gopass.

webpass Репозиторий: https://github.com/emersion/webpass Веб-интерфейс к pass. Написано на golang.

Приложения для Android

Password Store URL: https://play.google.com/store/apps/details?id=dev.msfjarvis.aps Репозиторий с кодом: https://github.com/android-password-store/Android-Password-Store Документация: https://android-password-store.github.io/docs/ Для синхронизации с git необходимо импортировать ssh-ключи. Поддерживает разблокировку по биометрическим данным. Для работы требует наличия OpenKeychain: Easy PGP.

OpenKeychain: Easy PGP URL: https://play.google.com/store/apps/details?id=org.sufficientlysecure.keychain Операции с ключами pgp. Необходимо будет импортировать pgp-ключи. Не поддерживает разблокировку по биометрическим данным. Необходимо набирать пароль ключа.

Пакеты для Emacs

pass Основной режим для управления хранилищем и редактирования записей. Emacs. Пакет pass Репозиторий: https://github.com/NicolasPetton/pass Позволяет редактировать базу данных паролей.

Запуск:

M-x pass

helm-pass Интерфейс helm для pass. Репозиторий: https://github.com/emacs-helm/helm-pass

Запуск:

M-x helm-pass

Выдаёт в минибуфере список записей из базы паролей. При нажатии Enter копирует пароль в буфер.

ivy-pass Интерфейс ivy для pass. Репозиторий: https://github.com/ecraven/ivy-pass

Управление файлами конфигурации

Использование chezmoi для управления файлами конфигурации домашнего каталога пользователя.

Общая информация

Сайт: https://www.chezmoi.io/ Репозиторий: https://github.com/twpayne/chezmoi

Конфигурация chezmoi

Рабочие файлы

Состояние файлов конфигурации сохраняется в каталоге

~/.local/share/chezmoi

Он является клоном вашего репозитория dotfiles. Файл конфигурации ~/.config/chezmoi/chezmoi.toml (можно использовать также JSON или YAML) специфичен для локальной машины. Файлы, содержимое которых одинаково на всех ваших машинах, дословно копируются из исходного каталога. Файлы, которые варьируются от машины к машине, выполняются как шаблоны, обычно с использованием данных из файла конфигурации локальной машины для настройки конечного содержимого, специфичного для локальной машины.

При запуске

chezmoi apply

вычисляется желаемое содержимое и разрешения для каждого файла, а затем вносит необходимые изменения, чтобы ваши файлы соответствовали этому состоянию.

По умолчанию chezmoi изменяет файлы только в рабочей копии.

Автоматически создавать файл конфигурации на новой машине

При выполнении chezmoi init также может автоматически создать файл конфигурации, если он еще не существует. Если ваш репозиторий содержит файл с именем .chezmoi.$FORMAT.tmpl, где $FORMAT есть один из поддерживаемых форматов файла конфигурации (json, toml, или yaml), то chezmoi init выполнит этот шаблон для создания исходного файла конфигурации.

Например, пусть ~/.local/share/chezmoi/.chezmoi.toml.tmpl выглядит так:

{{- $email := promptStringOnce . “email” “Email address” -}}

[data] email = {{ $email | quote }}

При выполнении chezmoi init будет создан конфигурационный файл ~/.config/chezmoi/chezmoi.toml. promptStringOnce — это специальная функция, которая запрашивает у пользователя значение, если оно еще не установлено в разделе data конфигурационного файла.

Чтобы протестировать этот шаблон, используйте chezmoi execute-template с флагами –init и –promptString, например:

chezmoi execute-template –init –promptString email=me@home.org < ~/.local/share/chezmoi/.chezmoi.toml.tmpl

Пересоздание файл конфигурации

Если вы измените шаблон файла конфигурации, chezmoi предупредит вас, если ваш текущий файл конфигурации не был сгенерирован из этого шаблона.

Вы можете повторно сгенерировать файл конфигурации, запустив:

chezmoi init

Шаблоны

Общая информация

Шаблоны используются для изменения содержимого файла в зависимости от среды. Используется синтаксис шаблонов Go. Файл интерпретируется как шаблон, если выполняется одно из следующих условий: имя файла имеет суффикс .tmpl; файл находится в каталоге .chezmoitemplates.

Данные шаблона

Полный список переменных шаблона:

chezmoi data

Источники переменных: файлы .chezmoi, например, .chezmoi.os; файлы конфигурации .chezmoidata.$FORMAT. Форматы (json, jsonc, toml, yaml) читаются в алфавитном порядке; раздел data конфигурационного файла.

Способы создания файла шаблона

При первом добавлении файла передайте аргумент –template:

chezmoi add –template ~/.zshrc

Если файл уже контролируется chezmoi, но не является шаблоном, можно сделать его шаблоном:

chezmoi chattr +template ~/.zshrc

Можно создать шаблон вручную в исходном каталоге, присвоив ему расширение .tmpl:

chezmoi cd $EDITOR dot\_zshrc.tmpl

Шаблоны в каталоге .chezmoitemplates должны создаваться вручную:

chezmoi cd mkdir -p .chezmoitemplates cd .chezmoitemplates $EDITOR mytemplate

Редактирование файла шаблона

Используйте chezmoi edit:

chezmoi edit ~/.zshrc

Чтобы сделанные вами изменения сразу же применялись после выхода из редактора, используйте опцию –apply:

chezmoi edit –apply ~/.zshrc

Тестирование шаблонов

Тестирование с помощью команды chezmoi execute-template.

Тестирование небольших фрагментов шаблонов:

chezmoi execute-template ‘{{ .chezmoi.hostname }}’

Тестирование целых файлов:

chezmoi cd chezmoi execute-template < dot\_zshrc.tmpl

Синтаксис шаблона

Действия шаблона записываются внутри двойных фигурных скобок, {{ }}. Действия могут быть переменными, конвейерами или операторами управления. Текст вне действий копируется буквально.

Переменные записываются буквально:

{{ .chezmoi.hostname }}

Условные выражения могут быть записаны с использованием if, else if, else, end:

{{ if eq .chezmoi.os “darwin” }} darwin

{{ else if eq .chezmoi.os “linux” }} linux

{{ else }} other operating system

{{ end }}

Удаление пробелов

Для удаления проблем в шаблоне разместите знак минус и пробела рядом со скобками:

HOSTNAME={{- .chezmoi.hostname }}

В результате получим:

HOSTNAME=myhostname

Отладка шаблона

Используется подкоманда execute-template:

chezmoi execute-template ‘{{ .chezmoi.os }}/{{ .chezmoi.arch }}’

Интерпретируются любые данные, поступающие со стандартного ввода или в конце команды.

Можно передать содержимое файла этой команде:

сat foo.txt | chezmoi execute-template

Логические операции Возможно выполнение логических операций.

Если имя хоста машины равно work-laptop, текст между if и end будет включён в результат:

# common config export EDITOR=vi

# machine-specific configuration {{- if eq .chezmoi.hostname “work-laptop” }} # this will only be included in ~/.bashrc on work-laptop {{- end }}

Логические функции eq: возвращает true, если первый аргумент равен любому из остальных аргументов, может принимать несколько аргументов; not: возвращает логическое отрицание своего единственного аргумента; and: возвращает логическое И своих аргументов, может принимать несколько аргументов; or: возвращает логическое ИЛИ своих аргументов, может принимать несколько аргументов.

Целочисленные функции len: возвращает целочисленную длину своего аргумента; eq: возвращает логическую истину arg1 == arg2; ne: возвращает логическое значение arg1 != arg2; lt: возвращает логическую истину arg1 < arg2; le: возвращает логическую истину arg1 <= arg2; gt: возвращает логическую истину arg1 > arg2; ge: возвращает логическую истину arg1 >= arg2.

Переменные шаблона

Чтобы просмотреть переменные, доступные в вашей системе, выполните:

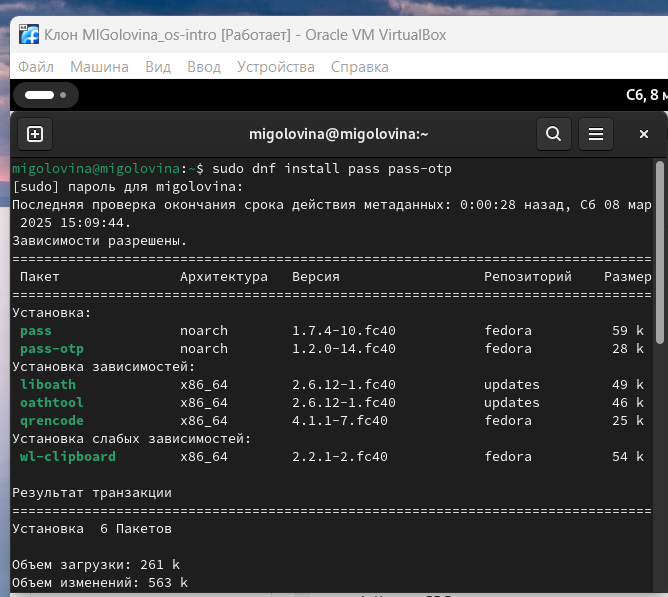
chezmoi data

Чтобы получить доступ к переменной chezmoi.kernel.osrelease в шаблоне, используйте:

{{ .chezmoi.kernel.osrelease }}

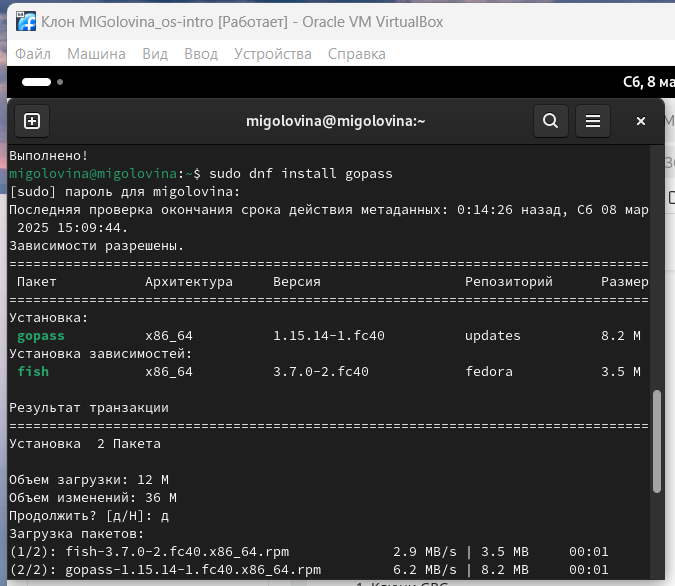
# 4 Выполнение лабораторной работы

1. Установка pass (рис. 4.1).



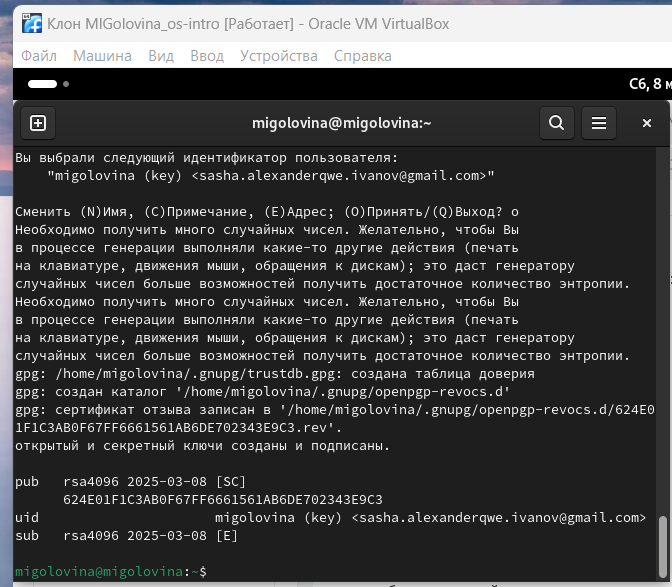
Установка pass

1. Установка gopass (рис. 4.2).

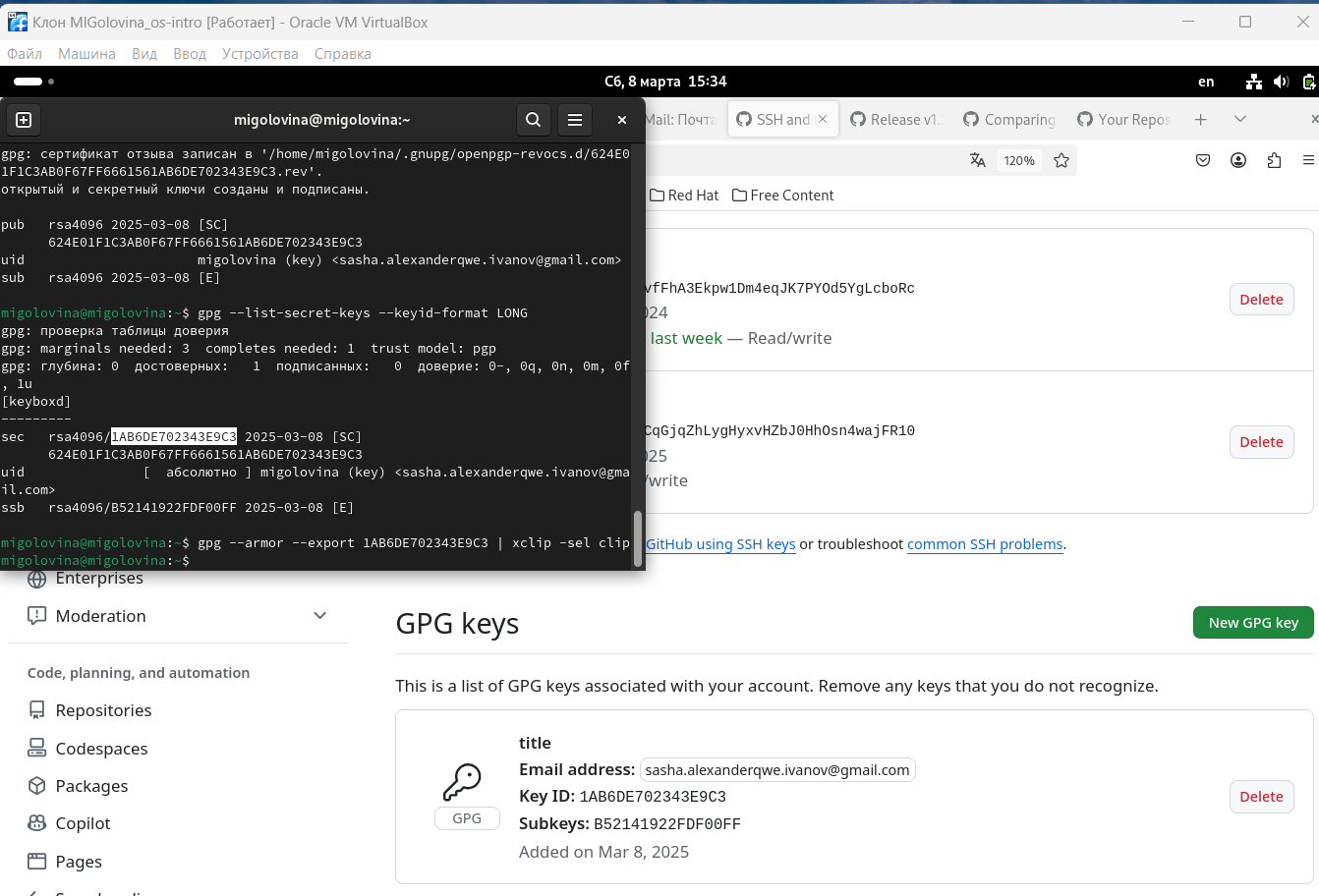


Установка gopass

1. Создание нового ключа gpg (рис. 4.3-4.4).

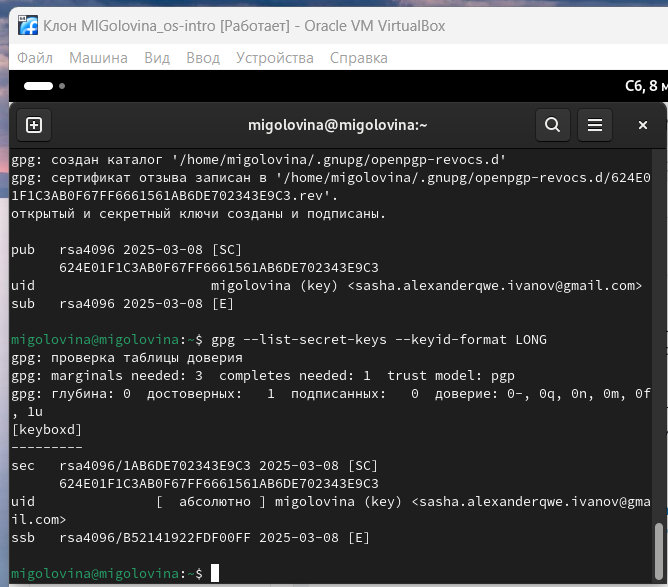


Создание нового ключа gpg



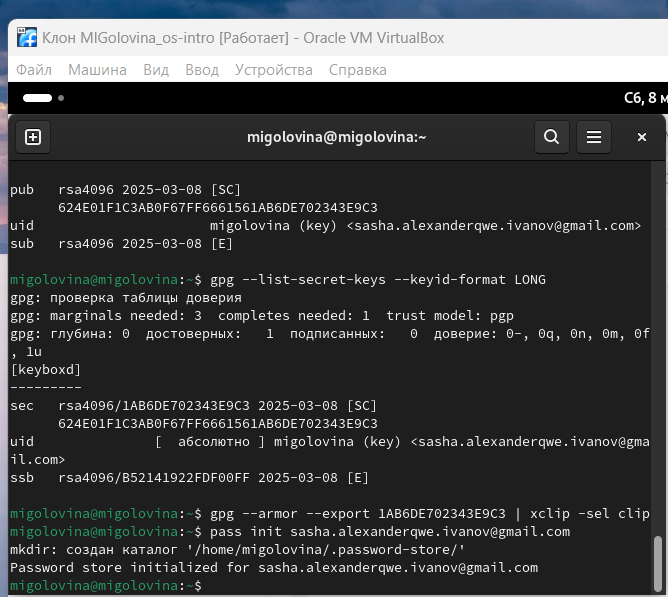
Создание нового ключа gpg

1. Просмотр списка ключей (рис. 4.5).



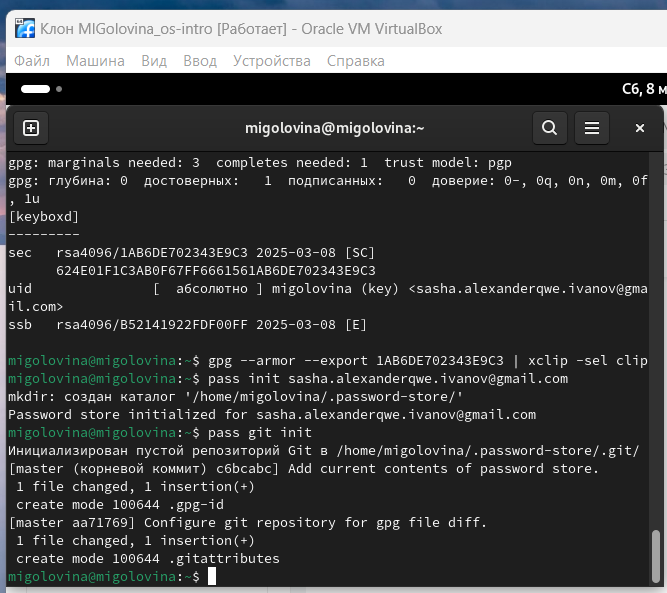
Просмотр списка ключей

1. Инициализация хранилища (рис. 4.6).



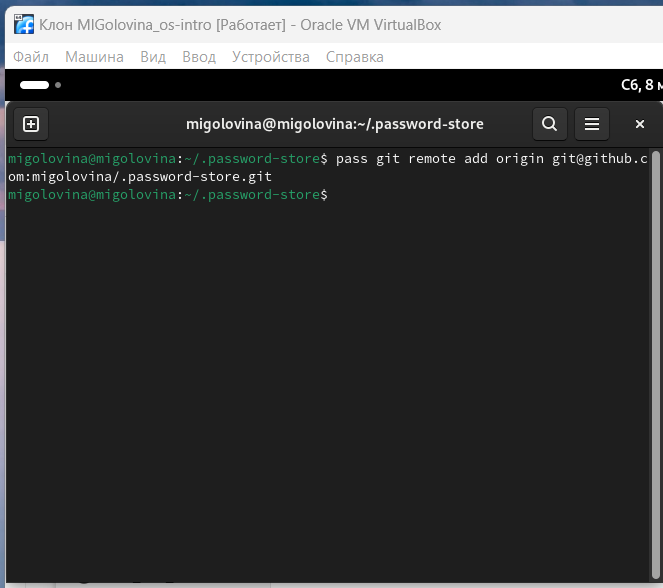
Инициализация хранилища

1. Создание структуры git (рис. 4.7).



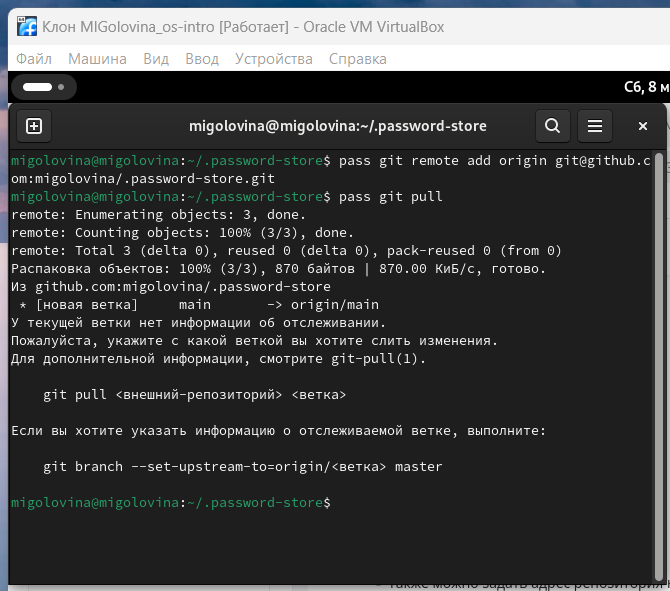
Создание структуры git

1. Задаю адрес репозитория на хостинге (рис. 4.8).

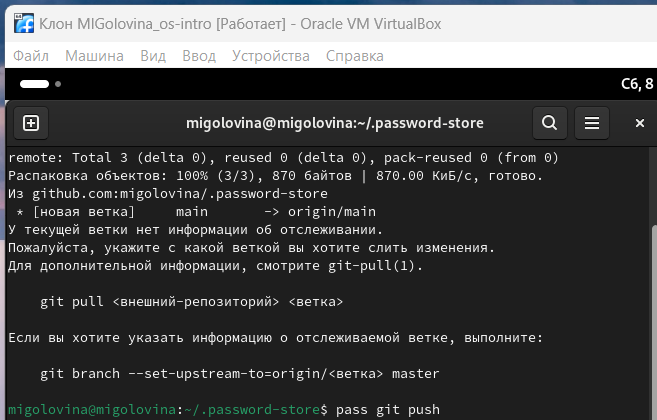


Задаю адрес репозитория на хостинге

1. Для синхранизации выполняются следующие команды (рис. 4.9-4.10).

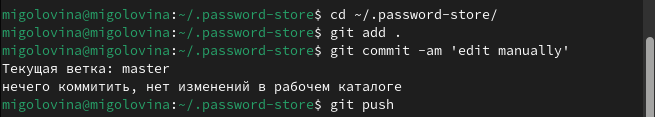


Команды



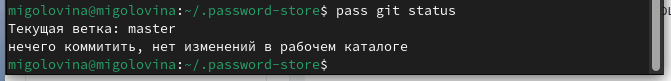
Команды

1. Коммит и выкладывание изменений (рис. 4.11).



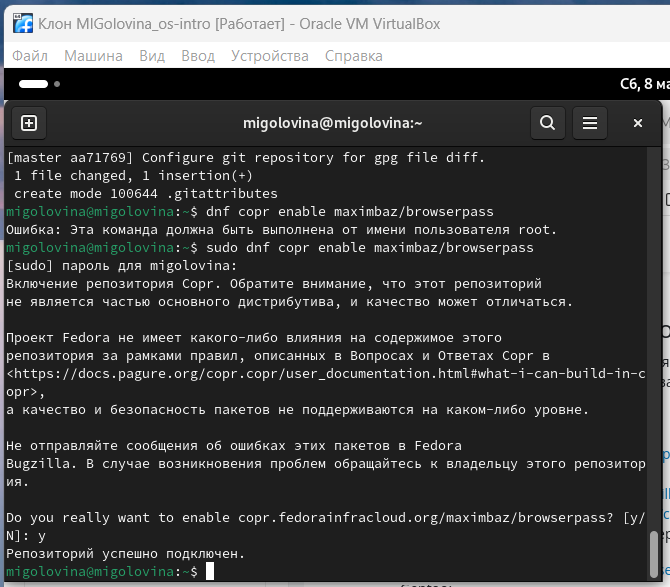
Коммит и выкладывание изменений

1. Проверка статус синхронизации модно командой (рис. 4.12).

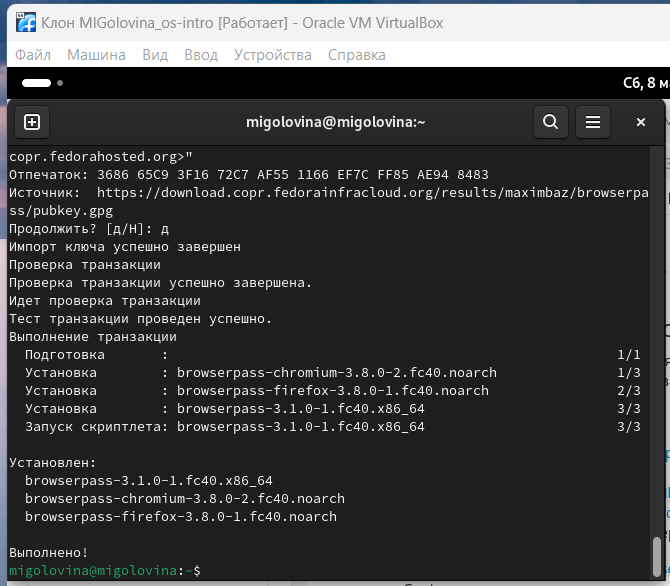


Проверка

1. Устанавка программ (рис. 4.13-4.14).

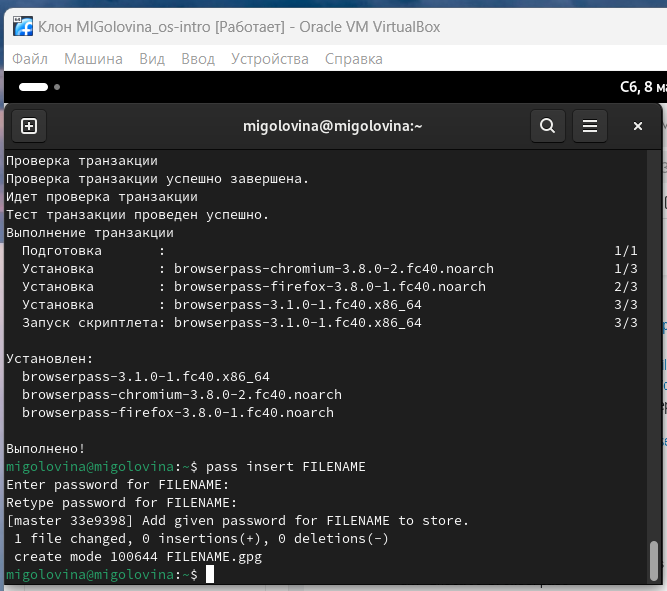


dnf copr enable maximbaz/browserpass



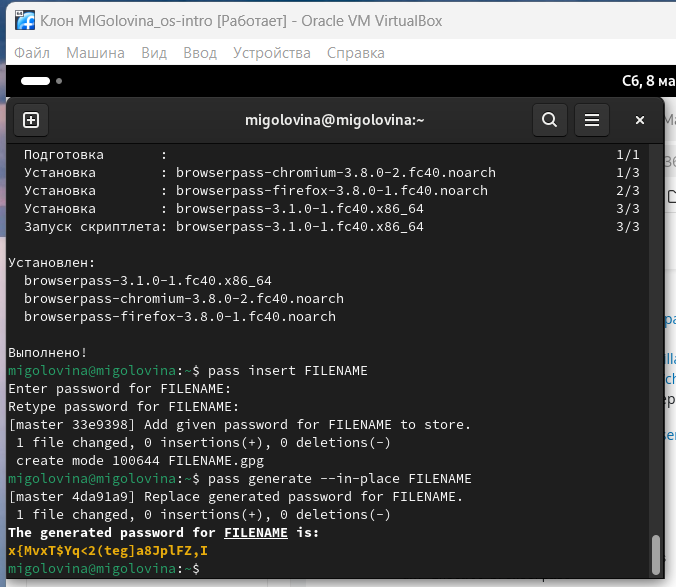
dnf install browserpass

1. Добавление нового пароля (рис. 4.15).



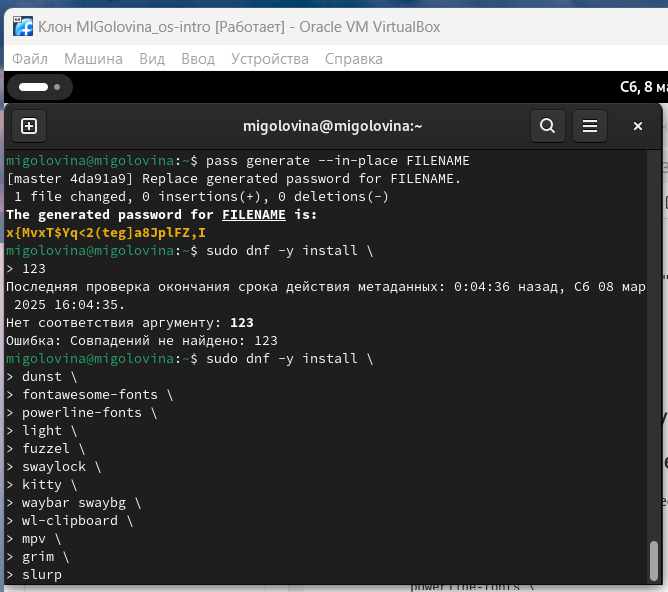
Добавление

1. Замена существующего пароля (рис. 4.16).



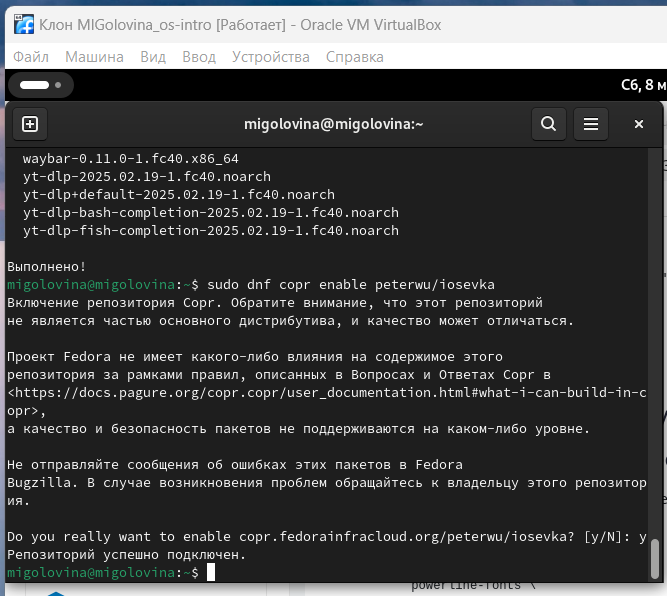
Замена

1. Установка дополнительного программного обеспечения (рис. 4.17).

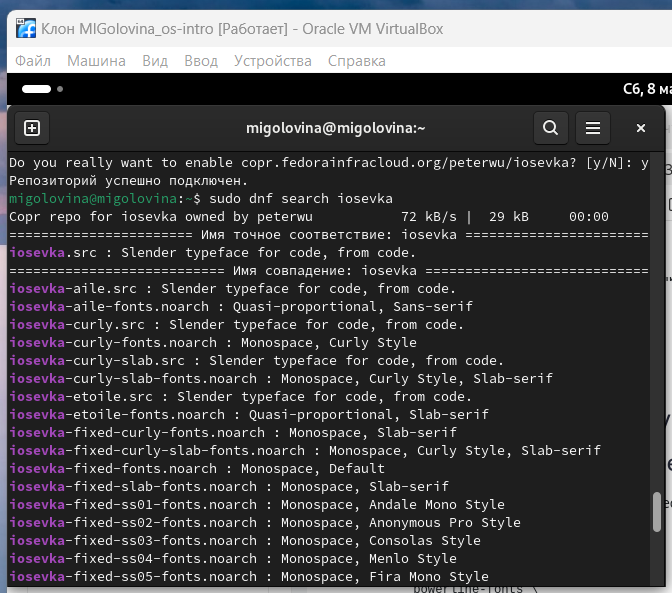


Установка

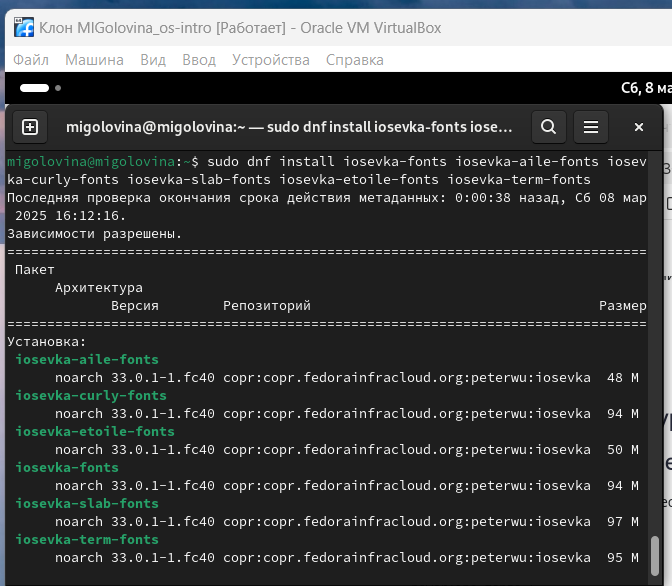
1. Установка шрифтов (рис. 4.18-4.20).



sudo dnf copr enable peterwu/iosevka

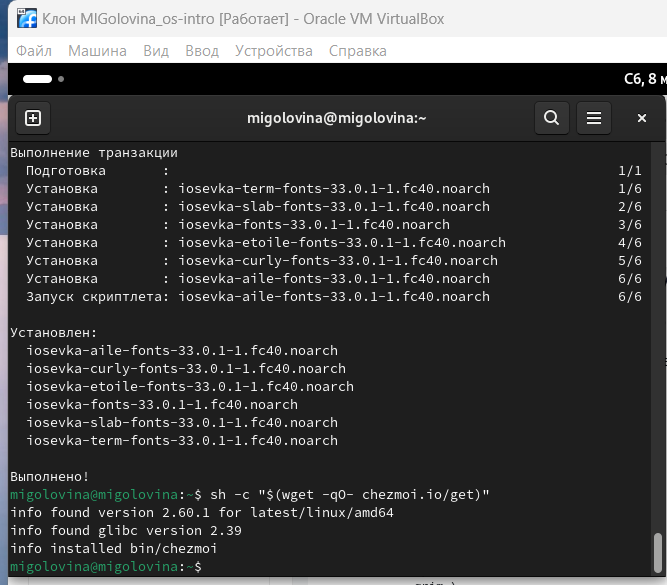


sudo dnf search iosevka



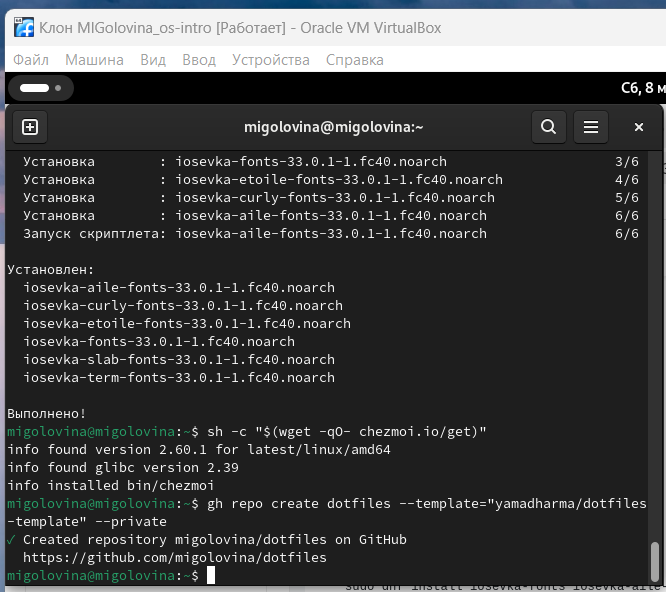
sudo dnf install iosevka-fonts iosevka-aile-fonts iosevka-curly-fonts iosevka-slab-fonts iosevka-etoile-fonts iosevka-term-fonts

1. Установка бинарного файла (рис. 4.21).



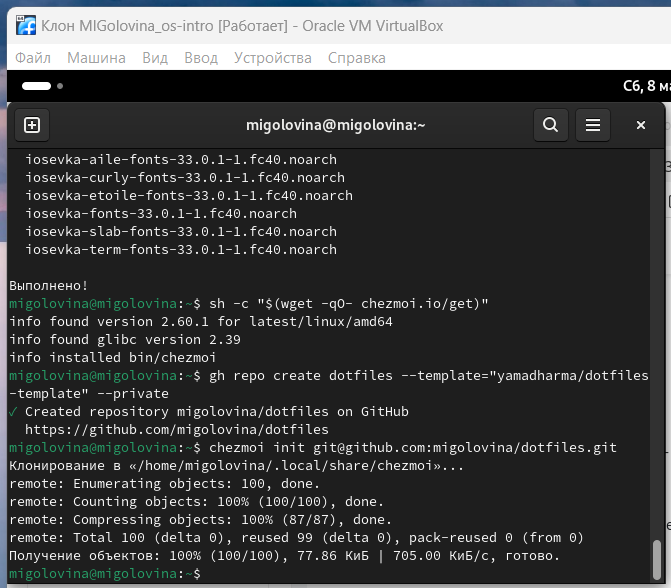
Установка

1. Созданией своего репозитория для конфигурационных файлов на основе шаблона (рис. 4.22).

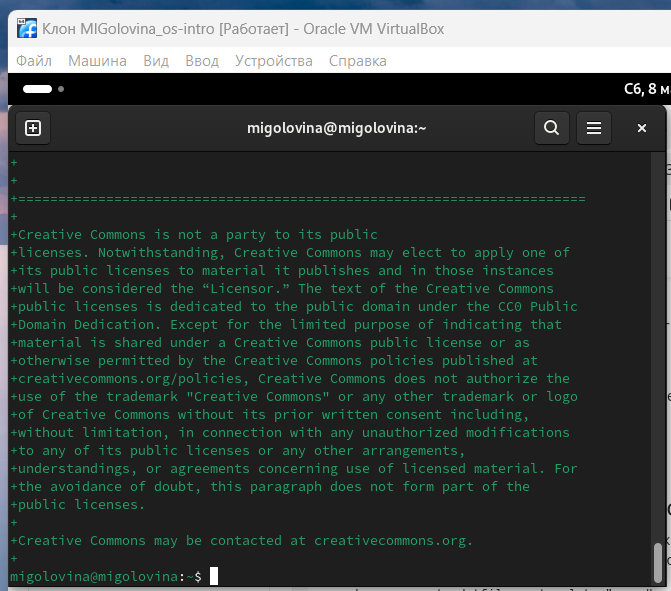


Установка

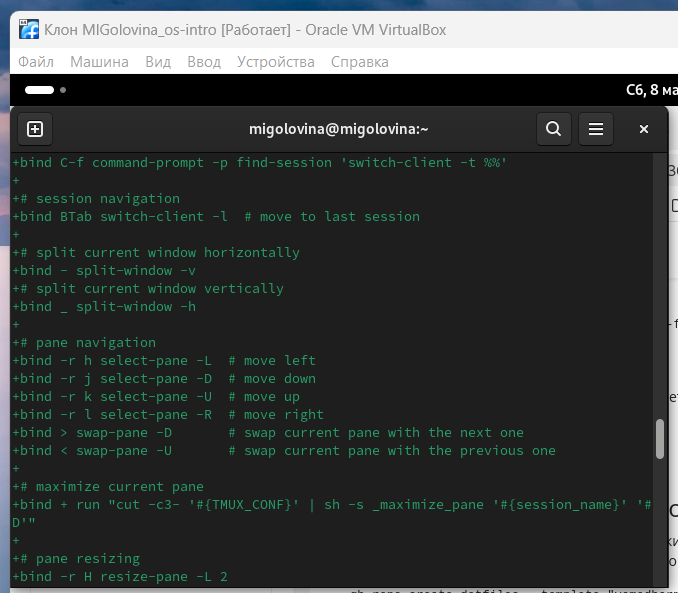
1. Подключение репозитория к своей системе (рис. 4.23-4.25).



Инициализация chezmoi с вашим репозиторием dotfiles

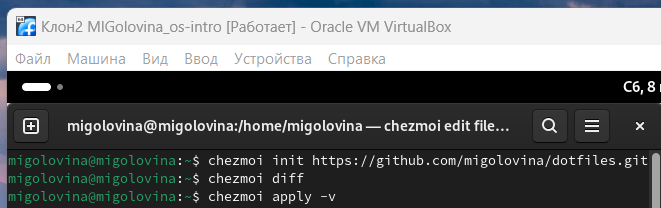


Проверка, какие изменения внесёт chezmoi в домашний каталог

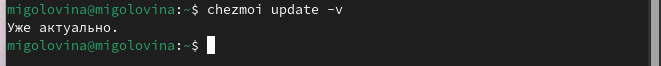


Меня устроили внесённые изменения, поэтому ввела следущую команду

1. Переход во вторую машину. Использование chezmoi на нескольких машинах (рис. 4.26-4.27).



Инициализация chezmoi, проверка, chezmoi apply -v



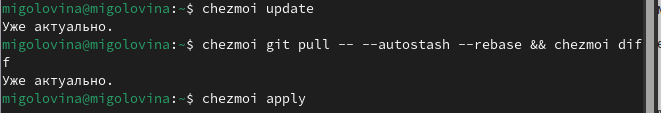
получение и применения последних изменений из вашего репозитория

1. Настройка новой машины с помощью одной команды (рис. 4.28).

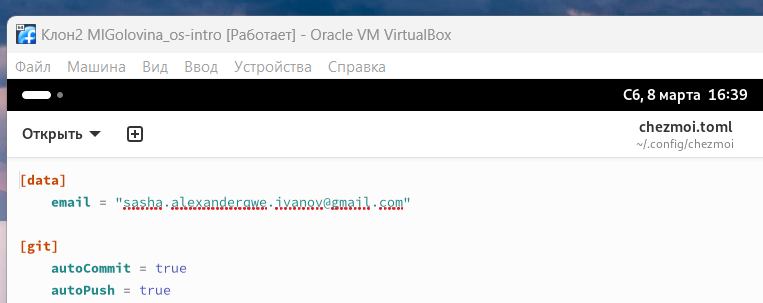
Установка dotfiles

Установка dotfiles

1. Ежедневные операции c chezmoi (рис. 4.29-4.30).



chezmoi update, chezmoi git pull – –autostash –rebase && chezmoi diff, chezmoi apply



Автоматическая фиксация и отправка изменения в репозиторий

# 5 Выводы

Я приобрела навыки по настройке рабочей среды.

# Список литературы

1. Dash, P. Getting Started with Oracle VM VirtualBox / P. Dash. – Packt Publishing Ltd, 2013. – 86 сс.
2. Colvin, H. VirtualBox: An Ultimate Guide Book on Virtualization with VirtualBox. VirtualBox / H. Colvin. – CreateSpace Independent Publishing Platform, 2015. – 70 сс.
3. Vugt, S. van. Red Hat RHCSA/RHCE 7 cert guide : Red Hat Enterprise Linux 7 (EX200 and EX300) : Certification Guide. Red Hat RHCSA/RHCE 7 cert guide / S. van Vugt. – Pearson IT Certification, 2016. – 1008 сс.
4. Робачевский, А. Операционная система UNIX / А. Робачевский, С. Немнюгин, О. Стесик. – 2-е изд. – Санкт-Петербург : БХВ-Петербург, 2010. – 656 сс.
5. Немет, Э. Unix и Linux: руководство системного администратора. Unix и Linux / Э. Немет, Г. Снайдер, Т.Р. Хейн, Б. Уэйли. – 4-е изд. – Вильямс, 2014. – 1312 сс.
6. Колисниченко, Д.Н. Самоучитель системного администратора Linux : Системный администратор / Д.Н. Колисниченко. – Санкт-Петербург : БХВ-Петербург, 2011. – 544 сс.
7. Robbins, A. Bash Pocket Reference / A. Robbins. – O’Reilly Media, 2016. – 156 сс.