Отчёт по лабораторной работе №5

Настройка рабочей среды

Седохин Даниил Алексеевич

Содержание

# 1 Цель работы

Получение навыков настройки рабочей среды.

# 2 Задание

Настроить рабочую среду.

# 3 Теоретическое введение

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:  
  
 cat 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  
   Установим pass и gopass (рис. 1 2).

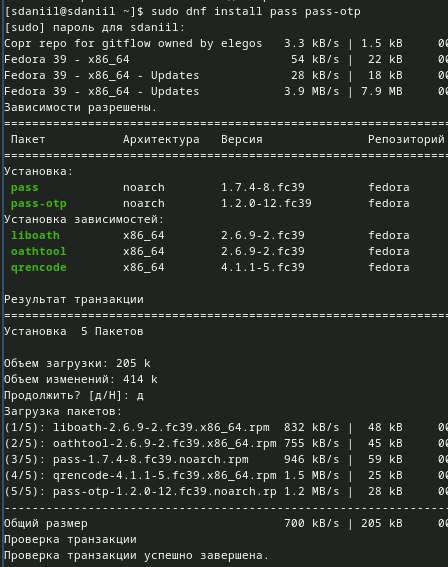


Рис. 1: Установка pass

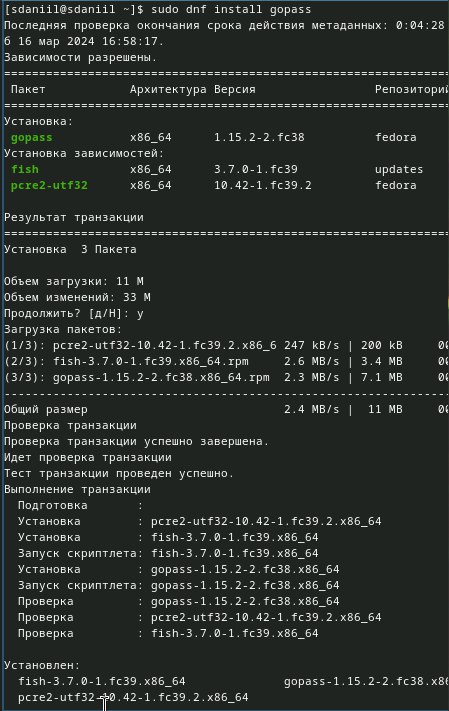


Рис. 2: Установка gopass

1. Настройка

Ключи GPG

Просмотр списка ключей:   
  
gpg --list-secret-keys (рис. [-@fig:003]).

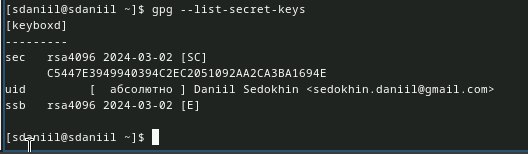


Рис. 3: Просмотр списка ключей

1. Инициализация хранилища

* Инициализируем хранилище:
* pass init (рис. 4).

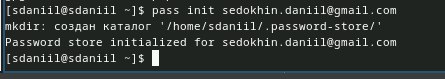


Рис. 4: Инициализация хранилища

1. Синхронизация с git

* Создадим структуру git:

pass git init (рис. 5)

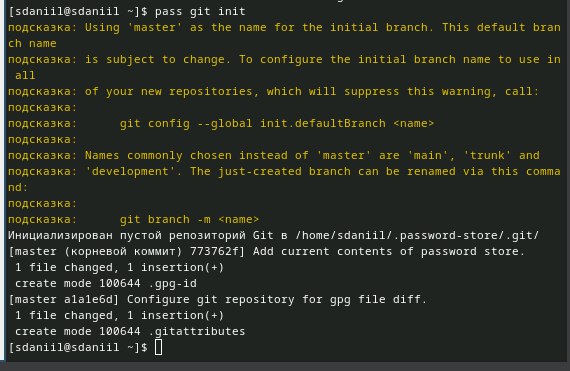


Рис. 5: Создание стркутуры git

1. Также можно задать адрес репозитория на хостинге (репозиторий необходимо предварительно создать):

pass git remote add origin git@github.com:/.git (рис. 6).

Задаем адрес репозитория

Рис. 6: Задаем адрес репозитория

1. Для синхронизации выполняется следующая команда:

pass git pull  
pass git push (рис. 7).

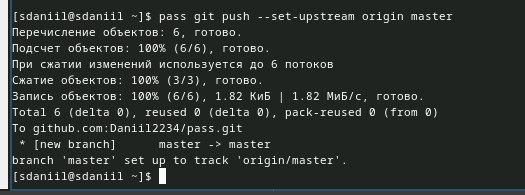


Рис. 7: Синхронизация

1. Прямые изменения

* Следует заметить, что отслеживаются только изменения, сделанные через сам gopass (или pass).
* Если изменения сделаны непосредственно на файловой системе, необходимо вручную закоммитить и выложить изменения:
* cd ~/.password-store/  
  git add .  
  git commit -am ‘edit manually’  
  git push (рис. 8).

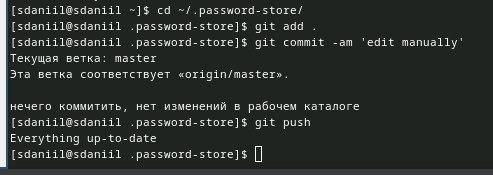


Рис. 8: Коммит

1. Проверить статус синхронизации модно командой

pass git status (рис. 9).

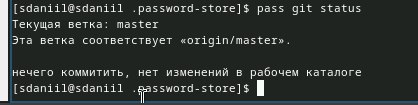


Рис. 9: Проверка статуса синхронизации

1. Настройка интерфейса с броузером

* Для взаимодействия с броузером используется интерфейс native messaging.  
  Поэтому кроме плагина к броузеру устанавливается программа, обеспечивающая интерфейс native messaging.
* Плагин browserpass  
  Репозиторий: https://github.com/browserpass/browserpass-extension Плагин для брoузера Плагин для Firefox: https://addons.mozilla.org/en-US/firefox/addon/browserpass-ce/. Плагин для Chrome/Chromium: https://chrome.google.com/webstore/detail/browserpass-ce/naepdomgkenhinolocfifgehidddafch.
* Интерфейс для взаимодействия с броузером (native messaging)  
   Репозиторий: https://github.com/browserpass/browserpass-native  
    
   Gentoo:  
    
   emerge www-plugins/browserpass  
    
   Fedora  
    
   dnf copr enable maximbaz/browserpass  
   dnf install browserpass

(рис. 10 11).

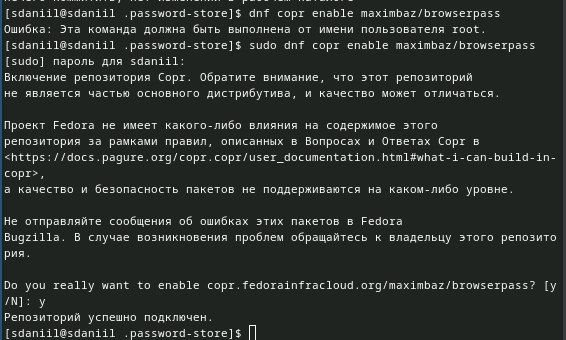


Рис. 10: Настройка интерфейса

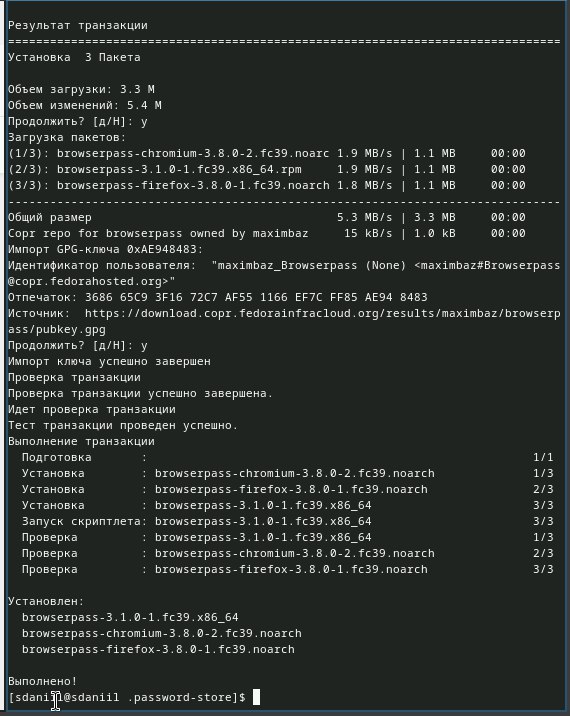


Рис. 11: Плагин browserpass

1. Сохранение пароля

* Добавить новый пароль
* Выполним:  
    
  pass insert [OPTIONAL DIR]/[FILENAME]
* (рис. 12).

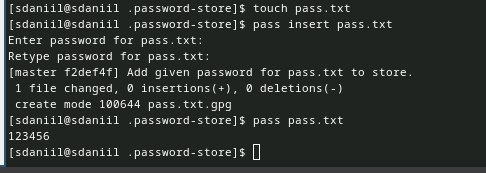


Рис. 12: Добавление нового пароля

1. OPTIONAL DIR: необязательное имя каталога, определяющее файловую структуру для вашего хранилища паролей; FILENAME: имя файла, который будет использоваться для хранения пароля.

Отобразите пароль для указанного имени файла:

pass [OPTIONAL DIR]/[FILENAME]

Замените существующий пароль:

pass generate –in-place FILENAME (рис. 13).

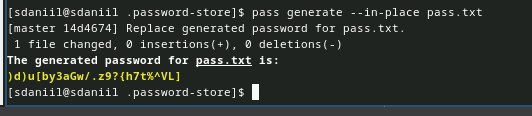


Рис. 13: Замена существующего пароля

1. Установите дополнительное программное обеспечение:

sudo dnf -y install   dunst   fontawesome-fonts   powerline-fonts   light   fuzzel   swaylock   kitty   waybar swaybg   wl-clipboard   mpv   grim   slurp (рис. 14).

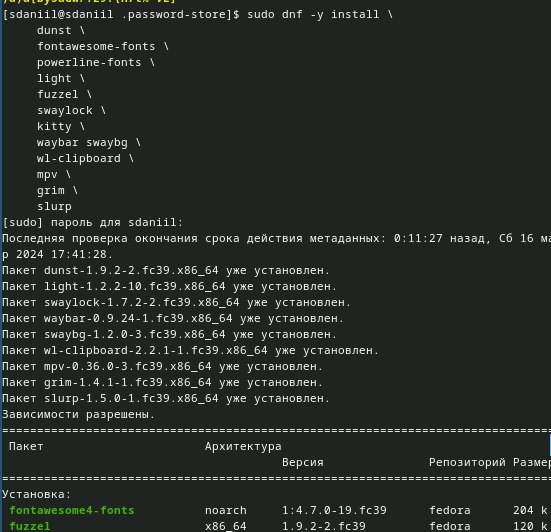


Рис. 14: Установка дополнительного ПО

1. Установим шрифты:

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 (рис. 15 16).

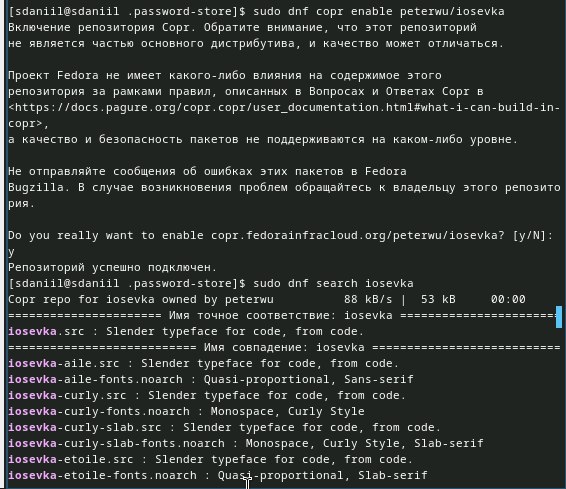


Рис. 15: Установка шрифтов

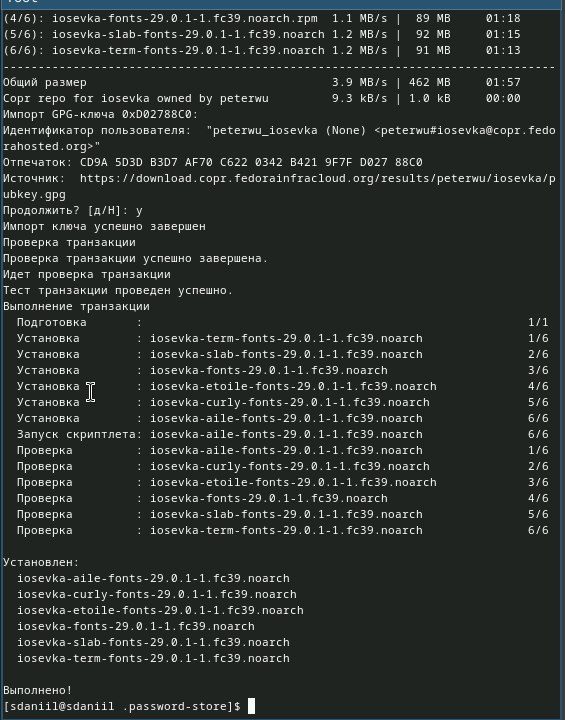


Рис. 16: Установка шрифтов

1. Установка бинарного файла. Скрипт определяет архитектуру процессора и операционную систему и скачивает необходимый файл:

* с помощью wget:  
    
  sh -c "$(wget -qO- chezmoi.io/get)"

(рис. 17).

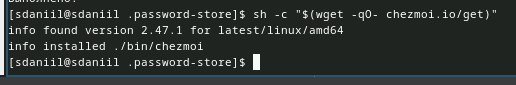


Рис. 17: Установка бинарного файла

1. Создание собственного репозитория с помощью утилит

* Будем использовать утилиты командной строки для работы с github.
* Создадим свой репозиторий для конфигурационных файлов на основе шаблона:
* gh repo create dotfiles –template=“yamadharma/dotfiles-template” –private 18).

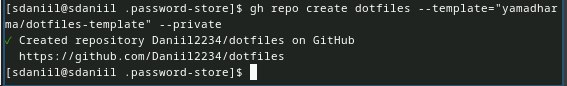


Рис. 18: Создание репозитория для конфигурационных файлов

1. Подключение репозитория к своей системе

* Инициализируйте chezmoi с вашим репозиторием dotfiles:
* chezmoi init git@github.com:/dotfiles.git (рис. 19).

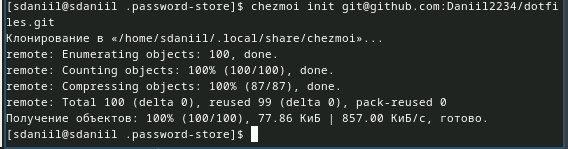


Рис. 19: Создание релиза на github

1. Проверим, какие изменения внесёт chezmoi в домашний каталог, запустив: (рис. 20).

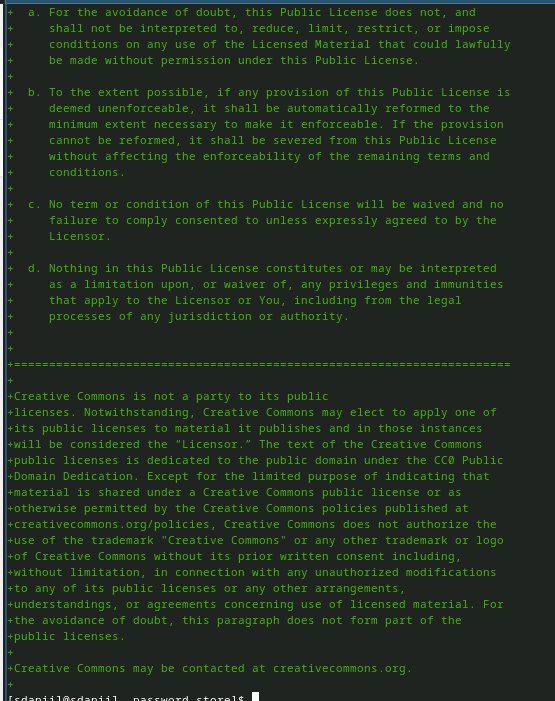


Рис. 20: Проверка изменений

1. Запуск:

chezmoi apply -v (рис. 21).

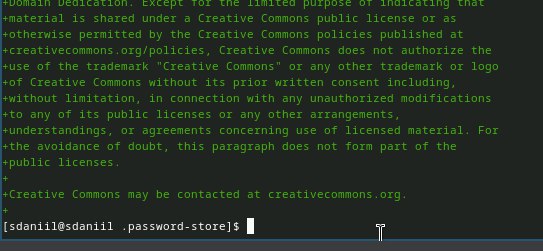


Рис. 21: Запуск chezmoi

1. Использование chezmoi на нескольких машинах

На второй машине инициализируйте chezmoi с вашим репозиторием dotfiles:   
  
chezmoi init https://github.com/<username>/dotfiles.git   
  
Или через ssh:   
  
chezmoi init git@github.com:<username>/dotfiles.git

(рис. 22).

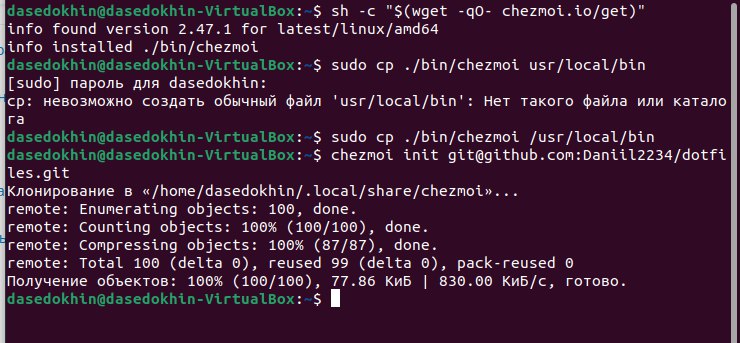


Рис. 22: Инициализация chezmoi на второй машине

1. Проверим, какие изменения внесёт chezmoi в домашний каталог, запустив:

chezmoi diff (рис. **¿fig:0023?**).

![Запуск chezmoi diff](image/23.jpg){#fig:0023 width=100%}

1. Запустим

chezmoi apply -v (рис. 23).

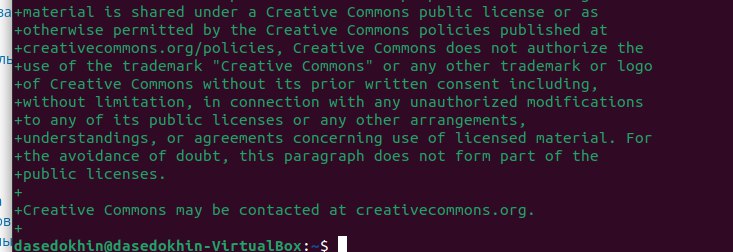


Рис. 23: Запуск chezmoi

1. При существующем каталоге chezmoi можно получить и применить последние изменения из вашего репозитория:

chezmoi update -v (рис. 24).

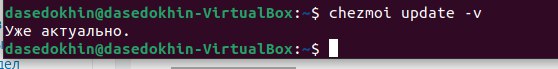


Рис. 24: Применение изменений

1. Настройка новой машины с помощью одной команды

* Можно установить свои dotfiles на новый компьютер с помощью одной команды:
* chezmoi init –apply https://github.com//dotfiles.git
* Через ssh:

chezmoi init –apply git@github.com:/dotfiles.git (рис. 25).

Установка dotfiles

Рис. 25: Установка dotfiles

25)Ежедневные операции c chezmoi

Извлеките последние изменения из репозитория и примените их   
  
 Можно извлечь изменения из репозитория и применить их одной командой:  
  
 chezmoi update (рис. [-@fig:0027]).

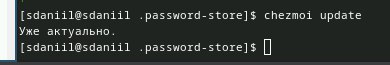


Рис. 26: Извлечение изменений

Это запускается git pull –autostash –rebase в вашем исходном каталоге, а затем chezmoi apply.

1. Извлеките последние изменения из своего репозитория и посмотрите, что изменится, фактически не применяя изменения

* Выполним:
* chezmoi git pull – –autostash –rebase && chezmoi diff
* Это запускается git pull –autostash –rebase в вашем исходном каталоге, а chezmoi diff затем показывает разницу между целевым состоянием, вычисленным из вашего исходного каталога, и фактическим состоянием.
* Если вы довольны изменениями, вы можете применить их:
* chezmoi apply (рис. 27).

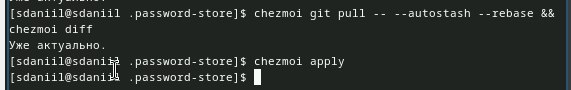


Рис. 27: Извлечение изменений

1. Автоматически фиксируйте и отправляйте изменения в репозиторий

* Можно автоматически фиксировать и отправлять изменения в исходный каталог в репозиторий.  
  Эта функция отключена по умолчанию.
* Чтобы включить её, добавьте в файл конфигурации ~/.config/chezmoi/chezmoi.toml следующее:
* [git] autoCommit = true autoPush = true
* Всякий раз, когда в исходный каталог вносятся изменения, chezmoi фиксирует изменения с помощью автоматически сгенерированного сообщения фиксации и отправляет их в ваш репозиторий.  
  Будьте осторожны при использовании autoPush. Если ваш репозиторий dotfiles является общедоступным, и вы случайно добавили секрет в виде обычного текста, этот секрет будет отправлен в ваш общедоступный репозиторий. (рис. 28).



Рис. 28: Автоматическая фиксация и отправление изменений в репозиторий

# 5 Выводы

Я Получил навыки настройки рабочей среды.