

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра
на тему: «Проект підсистеми адміністрування ОС Linux
на основі мови програмування Python»

Виконала студентка 4-го курсу групи КІ-19-1
спеціальності 123 «Комп'ютерна інженерія»



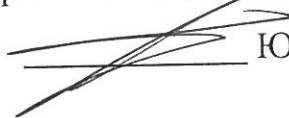
Мар'яна Матвієнко

Керівник доцент кафедри КІ та КБ, к.пед.н.



Олена Головня

Рецензент доцент кафедри КІ та КБ, к.т.н., доцент



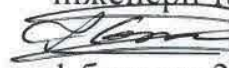
Юрій Бродський

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 123 «Комп'ютерна інженерія»
Освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної
інженерії та кібербезпеки

 Андрій Єфіменко
1 березня 2023 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу бакалавра

Студент: Матвієнко Мар'яна Валеріївна

Тема роботи: «Проект підсистеми адміністрування ОС Linux на основі мови програмування Python»

затверджена Наказом університету від 1 березня 2023 р. № 98/с

Термін здачі студентом закінченої роботи 26 червня 2023 р.

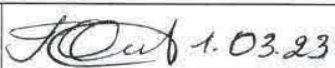
Вихідні дані роботи (зазначається предмет і об'єкт дослідження):
засоби мови програмування Python та їх використання для автоматизації адміністрування ОС Linux; процес адміністрування операційної системи Linux.

Обладнання та програмне забезпечення:

1. GNS3.
2. Oracle VirtualBox.
3. Linux Server Ubuntu 22.04.2 LTS.
4. Fedora Workstation 36

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
I	Головня О. С.	 1.03.23	 1.03.23

2	Головня О. С.	 1.03.23	 1.03.23.
3	Головня О. С.	 1.03.23	 1.03.23

Студент  Мар'яна Матвієнко

Керівник  Олена Головня

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	1.03.2023 – 7.03.2023	
2	Аналіз літературних джерел	8.03.2023 – 20.03.2023	
3	Аналіз об'єкта дослідження	21.03.2023 – 25.03.2023	
4	Аналіз аналогічних розробок	25.03.2023 – 1.04.2023	
6	Розробка скрипта для PXE-сервера	2.04.2023 – 22.04.2023	
7	Прописання управління PXE-сервером	23.04.2023 – 25.04.2023	
8	Розробка скрипта для Apache-сервера	26.04.2023 – 10.05.2023	
9	Прописання управління	11.05.2023 – 26.05.2023	
11	Розробка управління користувачами	26.05.2023 – 5.06.2023	
12	Тестування працездатності підсистеми	06.06.2023 – 10.06.2023	
13	Написання пояснювальної записки	11.06.2023 – 25.06.2023	
14	Захист	27.06.2023	

Студент  Мар'яна Матвієнко

Керівник  Олена Головня

АНОТАЦІЯ

Текстова частина кваліфікаційної роботи складає 89 сторінок, містить 60 рисунків, 1 таблицю. Список використаних джерел налічує 15 джерел і займає 2 сторінки.

Метою кваліфікаційної роботи є розробка та реалізація підсистеми автоматизації адміністрування ОС Linux з використанням мови програмування Python.

У кваліфікаційній роботі визначено та реалізовано автоматизацію згідно з наступними завданнями.

- автоматизоване розгортання PXE-сервера та додавання різних конфігурацій ОС на віддалених серверах;
- автоматизація розгортання Apache-сервера;
- автоматична зміна конфігураційних файлів при додаванні нових віртуальних доменів;
- автоматизоване управління Apache-сервером та віртуальними доменами;
- автоматизоване підключення по SSH до віддалених серверів та здійснення управління користувачами.

Розглянуто особливості аналогів та мов програмування, обрано найбільш перспективну в умовах проекту.

Показано розробку підсистеми автоматизації адміністрування ОС Linux з використанням мови програмування Python.

Наведено результати тестування реалізованої підсистеми.

Ключові слова: АДМІНІСТРУВАННЯ LINUX, АВТОМАТИЗАЦІЯ, PXE-СЕРВЕР, APACHE, PYTHON.

					КІК.КР.Б – 123 – 12 – ПЗ			
Зми.	Арк.	№ докум.	Підпис	Дата	Проект підсистеми адміністрування ОС Linux на основі мови програму- вання Python	Літ.	Арк.	Аркушів
Розроб.		Матвієнко М.В.		09.03				90
Керівник		Головня О.С.		20.03.23				
Реценз.		Бродський Ю.Б.		20.03.23				
Н. контр.								
Затверд.		Єфіменко А.А.		20.03.23		ФІКТ, Гр. КІ-19-1		

ANNOTATION

The text part of the qualification work is 89 pages, contains 60 figures, 1 table. The list of references consists of 15 items presented on 2 pages.

The purpose of the qualification work is to develop and implement a subsystem for automating the administration of Linux using the Python programming language.

The main tasks for automation are defined and implemented.

- automated deployment of the RCE server and adding different OS configurations on remote servers;
- automated deployment of the Apache server;
- automatic change of configuration files when adding new virtual domains;
- automated management of the Apache server and virtual domains;
- automated SSH connection to remote servers and user management.

The features of analogues and programming languages are considered, and the most promising one is selected for the project.

The development of a Linux administration automation subsystem using the Python programming language is shown.

The results of testing the implemented subsystem are presented.

Keywords: LINUX ADMINISTRATION, AUTOMATION, PXE SERVER, APACHE, PYTHON.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ВСТУП	9
РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ АВТОМАТИЗАЦІЇ АДМІНІСТРУВАННЯ ОС LINUX.....	11
1.1 Підсистеми адміністрування ОС Linux.....	11
1.1.1 Systemd	12
1.1.2 Puppet.....	16
1.2 Аналіз можливих рішень та обґрунтування вибору Python.....	22
1.2.1 Perl	23
1.2.2 Ruby	24
1.2.3 PowerShell Core.....	25
1.2.4 Python.....	27
1.3 Інструменти для автоматизації адміністрування Linux на Python	28
Висновки до першого розділу.....	31
РОЗДІЛ 2 ПРОЕКТУВАННЯ ПІДСИСТЕМИ АВТОМАТИЗАЦІЇ АДМІНІСТРУВАННЯ ОС LINUX.....	32
2.1 Визначення задач для автоматизації	32
2.2 Розробка плану автоматизації.....	36
2.3 Вимоги до підсистеми автоматизації	38
2.4 Проектування мережі.....	40
Висновки до другого розділу	41
РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПІДСИСТЕМИ АДМІНІСТРУВАННЯ ОС LINUX...	42
3.1 Автоматизація створення PXE-сервера та завантажувальних кодів	42
3.1.1 Встановлення та налаштування сервера DHCP і TFTP.....	45
3.1.2 Встановлення та налаштування сервера NFS	46
3.1.3 Завантаження та створення конфігураційних файлів ОС	48
3.2 Автоматизація створення Apache-сервера.....	52
3.3 Автоматизація управління користувачами.....	56

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

3.4 Створення текстового інтерфейсу	60
3.5 Тестування	60
Висновки до третього розділу.....	68
ВИСНОВКИ.....	69
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	71
ДОДАТКИ.....	73

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ОС – операційна система

ПЗ – програмне забезпечення

PXE – Preboot Execution Environment

DHCP – Dynamic Host Configuration Protocol

TFTP – Trivial File Transfer Protocol

NFS – Network File System

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. ОС Linux широко використовується як серверна платформа у багатьох сферах, включаючи хмарні обчислення, мережеві пристрої, веб-хостинг, бази даних, наукові дослідження та багато іншого.

Автоматизація адміністрування ОС Linux дозволяє забезпечити ефективне керування великими фермами серверів, підтримку стабільності та безпеки системи, швидку відмовостійкість, масштабованість та гнучкість у вирішенні завдань адміністрування.

Python, як мова програмування, має безліч переваг для автоматизації адміністративних задач. Ця мова проста у вивченні, має чистий синтаксис, багатий набір бібліотек і фреймворків, що дозволяють розробляти потужні та розширювані інструменти для керування ОС Linux. Python також підтримує велику спільноту розробників, що сприяє обміну знаннями, підтримці та постійному розвитку екосистеми.

Застосування автоматизації адміністрування ОС Linux з використанням Python дозволяє підвищити ефективність роботи адміністраторів, зменшити кількість помилок, забезпечити швидкий відгук на зміни та спростити процеси управління. Крім того, це сприяє автоматизації рутинних задач, що дозволяє адміністраторам більше часу приділяти стратегічним ініціативам та вдосконаленню інфраструктури.

Метою кваліфікаційної роботи є розробка та реалізація підсистеми автоматизації адміністрування ОС Linux з використанням мови програмування Python.

Визначена мета обумовлює наступні завдання.

- Розробка модулів для розгортання серверів.
- Розробка модуля для управління користувачами.
- Налаштування мережі.
- Тестування реалізованої підсистеми.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

Об’єктом дослідження є процес адміністрування операційної системи Linux.

Предметом дослідження є засоби мови програмування Python та їх використання для автоматизації адміністрування ОС Linux.

У процесі виконання кваліфікаційної роботи було розроблено проект підсистеми автоматизації адміністрування операційної системи Linux з використанням мови програмування Python. Реалізовано автоматичне розгортання DHCP, TFTP, NFS, PXE та Apache серверів, створено автоматизоване віддалене управління користувачами та налаштуванням конфігурацій ОС.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ АВТОМАТИЗАЦІЇ АДМІНІСТРУВАННЯ ОС LINUX

1.1 Підсистеми адміністрування ОС Linux

ОС Linux є однією з найпопулярніших операційних систем у світі, особливо в ІТ-середовищі. Для ефективного адміністрування ОС Linux інженери зазвичай використовують підсистеми адміністрування, які забезпечують автоматизацію багатьох процесів та спрощують роботу системного адміністратора.

Основні задачі, які можуть вирішуватися за допомогою підсистем адміністрування ОС Linux, є наступними.

1. Управління пакетами – встановлення, оновлення та видалення програм, забезпечення безпеки та стабільності системи.
2. Моніторинг – збір та аналіз системної інформації, визначення проблем та потенційних загроз для системи.
3. Налаштування – налаштування системи та її компонентів, забезпечення належного функціонування системи.
4. Автоматизація – створення скриптів та програм для автоматизації рутинних процесів, забезпечення швидкості та ефективності роботи.
5. Забезпечення безпеки – захист системи від вірусів, зломів та інших загроз, забезпечення безпеки даних.

Для розв'язання цих задач можна використовувати різні підсистеми адміністрування ОС Linux, перелічені нижче.

- Systemd – стандартна система ініціалізації в Linux, що забезпечує автоматичне запускання сервісів та демонів під час завантаження системи, моніторинг процесів та контроль їх стану;
- Puppet – система конфігураційного управління, що дозволяє автоматизувати процеси налаштування та управління системою;

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

- Ansible – інструмент для автоматизації конфігурації та управління системами, забезпечує простоту використання та можливість працювати з багатьма ОС;
- Chef – система управління конфігураціями, що дозволяє налаштовувати та управляти великою кількістю серверів з однієї точки керування;
- SaltStack – інструмент для автоматизації управління та моніторингу великих мереж інфраструктури, що дозволяє керувати більшою кількістю серверів.

Кожна з цих підсистем має свої переваги та недоліки.

1.1.1 Systemd

Systemd – це стандартна система ініціалізації в більшості сучасних дистрибутивів ОС Linux. Це програмне забезпечення відповідає за запуск та управління процесами під час завантаження системи. Systemd був розроблений з метою заміни класичної системи ініціалізації SysVinit, яка була використовувана раніше в більшості дистрибутивів ОС Linux.

У Systemd існує централізований менеджер процесів, який відповідає за запуск, зупинку та моніторинг процесів в системі. Крім того, Systemd надає широкі можливості для автоматизації та налаштування процесів запуску та управління різноманітними сервісами та демонами в системі. Systemd також може служити системою журналювання подій, що дозволяє вести детальний моніторинг та аналізувати стан системи.

На рисунку 1.1 зображені компоненти, що входять до Systemd. Це спрощене представлення, яке показує лише основну архітектуру.

Systemd має кілька переваг, серед яких:

- швидкість завантаження – systemd запускає процеси паралельно, що дозволяє зменшити час завантаження ОС;

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

- висока надійність – systemd має вбудовані засоби контролю над відповідністю процесів встановленим параметрам, відновлення процесів у разі помилок та попередження витоку пам'яті;
- простота налаштування – systemd забезпечує зручний інтерфейс для конфігурації та керування системою;
- широкий функціонал – systemd має багато вбудованих інструментів для автоматизації різних задач, таких як моніторинг процесів, логування, мережеві налаштування тощо.

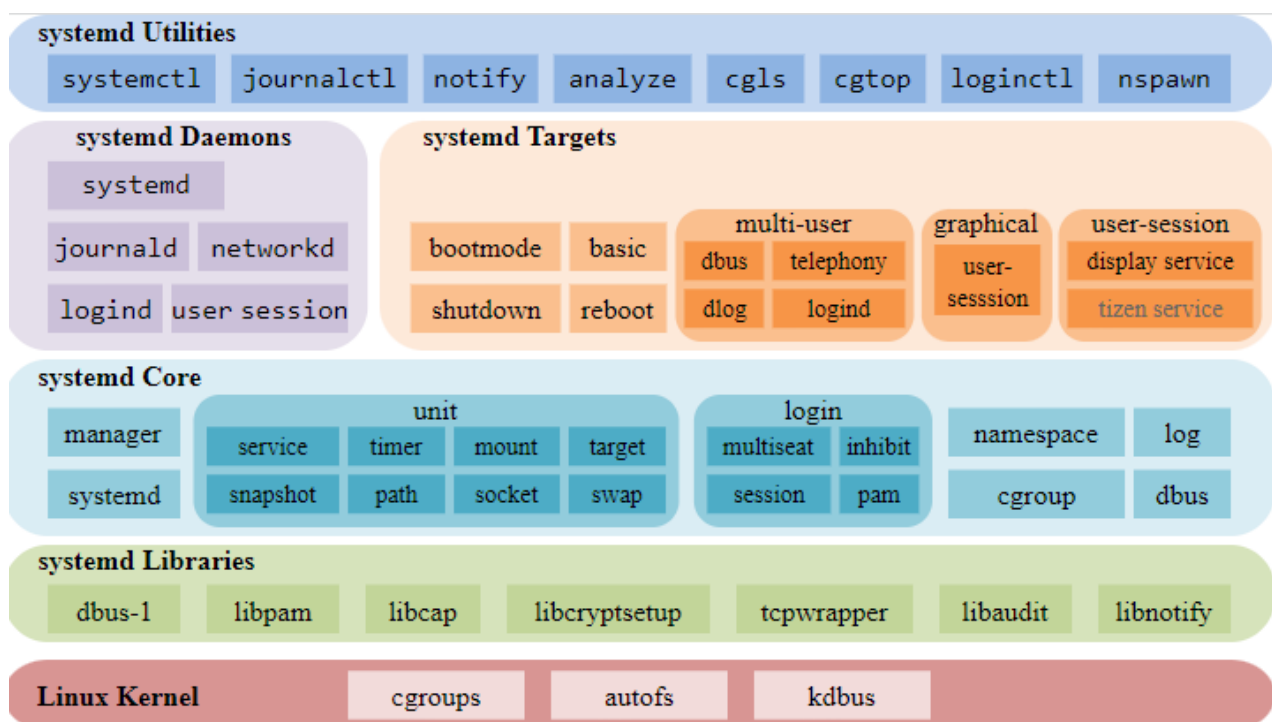


Рисунок 1.1 – Компоненти Systemd

Але, як і будь-яка технологія, systemd має й деякі недоліки:

- Складність – systemd має складну структуру та вимагає від користувачів певного рівня знань та досвіду для ефективного використання;
- залежність від systemd – деякі користувачі вважають, що залежність від systemd обмежує можливості вибору інших інструментів та використання альтернативних ОС;
- вимоги до обладнання – для ефективної роботи systemd потребує обладнання з достатньою кількістю ядер процесора та достатньої кількості оперативної пам'яті;

- некоректна робота з деякими програмами – у деяких випадках, systemd може працювати нестабільно з деякими програмами, які не дотримуються стандартів або мають специфічні вимоги до середовища виконання.

Systemd є стандартною системою ініціалізації в більшості сучасних дистрибутивів Linux, тому зазвичай вона вже встановлена в системі. Щоб перевірити її наявність, необхідно виконати наступну команду:

systemctl

Ця команда повинна показати перелік доступних служб і станів їх виконання (рис. 1.2).

```
student@student-VirtualBox:~$ systemctl
UNIT
proc-sys-fs-binfmt_misc.automount
dev-fuse.device
sys-devices-pci0000:00-0000:00:01.1-ata2-host1-target1:0:0-1:0:0:0-block-sr0.device
sys-devices-pci0000:00-0000:00:03.0-net-enp0s3.device
sys-devices-pci0000:00-0000:00:05.0-sound-card0.device
sys-devices-pci0000:00-0000:00:0d.0-ata3-host2-target2:0:0-2:0:0:0-block-sda-sda1.d
sys-devices-pci0000:00-0000:00:0d.0-ata3-host2-target2:0:0-2:0:0:0-block-sda-sda2.d
sys-devices-pci0000:00-0000:00:0d.0-ata3-host2-target2:0:0-2:0:0:0-block-sda-sda5.d
sys-devices-pci0000:00-0000:00:0d.0-ata3-host2-target2:0:0-2:0:0:0-block-sda.device
sys-devices-platform-serial8250-tty-ttyS0.device
sys-devices-platform-serial8250-tty-ttyS1.device
sys-devices-platform-serial8250-tty-ttyS10.device
sys-devices-platform-serial8250-tty-ttyS11.device
```

Рисунок 1.2 – Перелік доступних служб і станів їх виконання

Як бачимо, Systemd наявна в системі. В разі ж відсутності її необхідно встановити за допомогою менеджера пакетів дистрибутиву Linux та перезавантажити систему. Для цього необхідно виконати наступні команди:

sudo apt-get install systemd

sudo reboot

Далі виконаємо кілька команд для моніторингу (рис. 1.3 – 1.5).

```

student@student-VirtualBox:~$ systemctl status ntp.service
● ntp.service - Network Time Service
   Loaded: loaded (/lib/systemd/system/ntp.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2023-03-27 03:37:00 EEST; 44s ago
     Docs: man:ntpd(8)
    Main PID: 9414 (ntpd)
      Tasks: 2 (limit: 1111)
     Memory: 1.3M
    CGroup: /system.slice/ntp.service
            └─9414 /usr/sbin/ntpd -p /var/run/ntpd.pid -g -u 126:133

бep 27 03:37:04 student-VirtualBox ntpd[9414]: Soliciting pool server 91.236.251.14
бep 27 03:37:04 student-VirtualBox ntpd[9414]: Soliciting pool server 213.231.5.55
бep 27 03:37:04 student-VirtualBox ntpd[9414]: Soliciting pool server 162.159.200.123
бep 27 03:37:05 student-VirtualBox ntpd[9414]: Soliciting pool server 94.158.46.150
бep 27 03:37:05 student-VirtualBox ntpd[9414]: Soliciting pool server 82.193.104.168
бep 27 03:37:05 student-VirtualBox ntpd[9414]: Soliciting pool server 91.236.251.13
бep 27 03:37:05 student-VirtualBox ntpd[9414]: Soliciting pool server 91.189.94.4
бep 27 03:37:06 student-VirtualBox ntpd[9414]: Soliciting pool server 185.125.190.57
бep 27 03:37:06 student-VirtualBox ntpd[9414]: Soliciting pool server 79.142.192.4

```

Рисунок 1.3 – Перевірка статусу служби

```

student@student-VirtualBox:~$ systemd-analyze blame
35.902s plymouth-quit-wait.service
25.050s vboxadd.service
 5.957s snapd.service
 3.627s dev-sda5.device
 3.017s networkd-dispatcher.service
 2.930s man-db.service
 2.695s udisks2.service
 2.515s accounts-daemon.service
 1.966s NetworkManager-wait-online.service
 1.695s logrotate.service
 1.629s NetworkManager.service
 1.565s avahi-daemon.service
 1.474s polkit.service
 1.196s systemd-resolved.service

```

Рисунок 1.4 – Перегляд часу, який потрібен для запуску програми чи служби

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

```

student@student-VirtualBox:~$ systemctl stop ntp.service
student@student-VirtualBox:~$ systemctl status ntp.service
● ntp.service - Network Time Service
   Loaded: loaded (/lib/systemd/system/ntp.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Mon 2023-03-27 03:39:28 EEST; 2s ago
     Docs: man:ntpd(8)
    Main PID: 9414 (code=exited, status=0/SUCCESS)

бер 27 03:37:05 student-VirtualBox ntpd[9414]: Soliciting pool server 94.158.46.150
бер 27 03:37:05 student-VirtualBox ntpd[9414]: Soliciting pool server 82.193.104.168
бер 27 03:37:05 student-VirtualBox ntpd[9414]: Soliciting pool server 91.236.251.13
бер 27 03:37:05 student-VirtualBox ntpd[9414]: Soliciting pool server 91.189.94.4
бер 27 03:37:06 student-VirtualBox ntpd[9414]: Soliciting pool server 185.125.190.57
бер 27 03:37:06 student-VirtualBox ntpd[9414]: Soliciting pool server 79.142.192.4
бер 27 03:37:06 student-VirtualBox ntpd[9414]: Soliciting pool server 91.236.251.12
бер 27 03:39:28 student-VirtualBox systemd[1]: Stopping Network Time Service...
бер 27 03:39:28 student-VirtualBox systemd[1]: ntp.service: Succeeded.
бер 27 03:39:28 student-VirtualBox systemd[1]: Stopped Network Time Service.
student@student-VirtualBox:~$ systemctl restart ntp.service
student@student-VirtualBox:~$ systemctl status ntp.service
● ntp.service - Network Time Service
   Loaded: loaded (/lib/systemd/system/ntp.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2023-03-27 03:39:54 EEST; 1s ago
     Docs: man:ntpd(8)
  Process: 10772 ExecStart=/usr/lib/ntp/ntp-systemd-wrapper (code=exited, status=0/SUCCESS)

```

Рисунок 1.5 – Зупинка роботи служби та перезапуск

Щоб перевірити, чи присутній сервіс в автозавантаженні, можна використати команду `systemctl is-enabled [назва_сервісу]` (рис. 1.6).

```

student@student-VirtualBox:~$ systemctl is-enabled ntp.service
enabled

```

Рисунок 1.6 – Перевірка наявності сервісу в автозагрузці

1.1.2 Puppet

Puppet – це система управління конфігурацією, що дозволяє автоматизувати процес налаштування та управління системою. За допомогою Puppet можна налаштувати різноманітні параметри на комп'ютерах, серверах та інших пристроях, що працюють під управлінням ОС Linux та інших операційних систем. Puppet дозволяє встановлювати програми, налаштовувати мережеві інтерфейси, файрволи, сертифікати, робити резервні копії та багато іншого.

Переваги використання Puppet для автоматизації адміністрування:

- ефективність – Puppet дозволяє ефективно керувати налаштуваннями на багатьох серверах;
- простота – Puppet забезпечує зручний і простий інтерфейс для зміни налаштувань сервера;

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

- гнучкість – Puppet може працювати з різними ОС і різними конфігураціями;
- масштабованість – Puppet дозволяє легко налаштувати налаштування для великої кількості серверів.

Недоліки використання Puppet для автоматизації адміністрування:

- Складність – Puppet може бути складним для налаштування і вимагає певного рівня знань;
- залежність – Puppet залежить від роботи самої системи ініціалізації;
- навчання – використання Puppet може вимагати додаткового часу для навчання та оволодіння;
- витрати – Puppet може вимагати додаткових витрат на обладнання та програмне забезпечення для налагодження та роботи.

Основна ідея роботи з Puppet полягає в тому, щоб зберігати конфігурацію серверів в файлових конфігураціях, які можуть бути контрольовані, редаговані та збережені за допомогою системи контролю версій (наприклад, Git).

Розглянемо приклад роботи з Puppet на прикладі встановлення та налаштування веб-сервера Apache на сервері з операційною системою Ubuntu 20.04.

Для початку встановлюємо дві віртуальні машини Ubuntu 20.04 у VirtualBox та налаштовуємо їх (рис. 1.7 – 1.8).

```
student@puppet-Agent:~$ sudo ip addr add 192.191.0.2/255.255.255.0 broadcast 192.191.0.255 dev enp0s3
student@puppet-Agent:~$ sudo ip route add default via 192.191.0.1
student@puppet-Agent:~$ ip addr show enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:2c:91:7d brd ff:ff:ff:ff:ff:ff
    inet 192.191.0.2/24 brd 192.191.0.255 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe2c:917d/64 scope link
        valid_lft forever preferred_lft forever
```

Рисунок 1.7 – Налаштування IP-адресації на першому сервері

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

```

student@student-VirtualBox:~$ sudo ip addr add 192.191.0.1/255.255.255.
0 broadcast 192.191.0.255 dev enp0s3
student@student-VirtualBox:~$ sudo ip route add default via 192.191.0.2
student@student-VirtualBox:~$ ip addr show enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel st
ate UP group default qlen 1000
    link/ether 08:00:27:2c:91:7d brd ff:ff:ff:ff:ff:ff
    inet 192.191.0.1/24 brd 192.191.0.255 scope global enp0s3
        valid_lft forever preferred_lft forever
student@student-VirtualBox:~$ ping 192.191.0.2
ЛУНА-ІМПУЛЬС 192.191.0.2 (192.191.0.2) 56(84) байтів даних.
64 байтів з 192.191.0.2: icmp_seq=1 ttl=64 час=1.04 мс
64 байтів з 192.191.0.2: icmp_seq=2 ttl=64 час=0.694 мс
64 байтів з 192.191.0.2: icmp_seq=3 ttl=64 час=0.702 мс
^C
--- Статистика луна-імпульсів 192.191.0.2 ---

```

Рисунок 1.8 – Налаштування IP-адресації на другому сервері та перевірка

Встановлюємо Puppet Master та Puppet Agent на сервері Ubuntu 20.04 (рис.1.9).

```
sudo apt-get update
```

```
sudo apt-get install puppetmaster puppet
```

```

Отр:70 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 c-n-f Metadata
a [516 B]
Отримано 16,6 MB за 4сВ (3 705 kB/s)

Зчитування переліків пакунків... Виконано
student@student-VirtualBox:~$
student@student-VirtualBox:~$ sudo apt-get install puppetmaster puppet
Зчитування переліків пакунків... Виконано
Побудова дерева залежностей
Зчитування інформації про стан... Виконано
Буде встановлено такі додаткові пакунки:
  augeas-lenses debconf-utils facter fonts-lato hiera javascript-common libaugeas0
  libboost-log1.71.0 libboost-program-options1.71.0 libboost-regex1.71.0
  libcpp-hocon0.1.7 libcurl4 libfacter3.11.0 libjs-jquery libleatherman1.4.2

```

Рисунок 1.9 – Встановлення Puppet Master та Puppet Agent

Далі встановлюємо пакет Puppet-Agent на другому сервері. Для цього до-
даємо репозиторій Puppet до списку репозиторіїв системи (рис. 1.10).

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

```

student@student-VirtualBox:~$ wget https://apt.puppetlabs.com/puppet7-r
elease-focal.deb
--2023-03-27 05:04:32-- https://apt.puppetlabs.com/puppet7-release-foc
al.deb
Визначення імені apt.puppetlabs.com (apt.puppetlabs.com)... 52.84.195.121
, 52.84.195.67, 52.84.195.84, ...
Встановлення з'єднання з apt.puppetlabs.com (apt.puppetlabs.com)|52.84.
195.121|:443... з'єднано.
HTTP-запит надіслано, очікуємо на відповідь... 200 OK
Довжина: 11720 (11K) [application/x-debian-package]
Зберігаємо до «puppet7-release-focal.deb»

puppet7-release-f 100%[=====>] 11,45K --.-KB/s за 0s

2023-03-27 05:04:32 (173 MB/s) - «puppet7-release-focal.deb» збережено
[11720/11720]

student@student-VirtualBox:~$ sudo dpkg -i puppet7-release-focal.deb
Вибір раніше не обраного пакунку puppet7-release.
dpkg: розглядується вилучення puppet6-release, замість якого буде викор
истано puppet7-release ...
dpkg: так, пакунок puppet6-release буде вилучено, щоб можна було встано

```

Рисунок 1.10 – Додавання репозиторію Puppet до списку репозиторіїв системи

Оновлюємо список пакетів за допомогою команди *sudo apt update* та вста-
новлюємо пакет *puppet-agent* (рис. 1.11).

```

student@student-VirtualBox:~$ sudo apt install puppet-agent
Зчитування переліків пакунків... Виконано
Побудова дерева залежностей
Зчитування інформації про стан... Виконано
Наступні пакунки були встановлені автоматично і більше не потрібні:
augeas-lenses fonts-lato javascript-common libaugeas0 libjs-jquery
libruby2.7 linux-headers-5.4.0-47 linux-headers-5.4.0-47-generic
linux-image-5.4.0-47-generic linux-modules-5.4.0-47-generic
linux-modules-extra-5.4.0-47-generic rake ruby ruby-augeas
ruby-deep-merge ruby-minitest ruby-net-telnet ruby-power-assert

```

Рисунок 1.11 – Встановлення пакету

Прописуємо конфігурацію головного сервера (рис. 1.12).

Для зручності варто перейменувати другий сервер на *puppet-Agent* (рис.
1.13). Після цього запускаємо *puppet* на головному сервері та додаємо сервіс до
списку сервісів для автозавантажень (рис. 1.14).

Відповідно необхідно запустити *puppet* на другому сервері (рис. 1.15).

```

[main]
certname = puppetAgent
server = student-VirtualBox

```

Рисунок 1.12 – Прописування головного сервера

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

```
student@student-VirtualBox:~$ sudo hostnamectl set-hostname puppet-Agent
student@student-VirtualBox:~$ sudo reboot
```

Рисунок 1.13 – Перейменовування сервера

```
student@student-VirtualBox:~$ sudo systemctl start puppet
student@student-VirtualBox:~$ sudo systemctl enable puppet
```

Рисунок 1.14 – Запуск puppet

```
student@puppet-Agent:~$ sudo systemctl start puppet
student@puppet-Agent:~$ sudo systemctl status puppet
● puppet.service - Puppet agent
   Loaded: loaded (/lib/systemd/system/puppet.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2023-03-27 05:56:41 EEST; 1s ago
     Docs: man:puppet-agent(8)
   Main PID: 2878 (puppet)
    Tasks: 1 (limit: 1111)
   Memory: 61.0M
   CGroup: /system.slice/puppet.service
           └─2878 /opt/puppetlabs/puppet/bin/ruby /opt/puppetlabs/puppet/bin/puppetd

6ep 27 05:56:41 puppet-Agent systemd[1]: Started Puppet agent.
lines 1-11/11 (END)
```

Рисунок 1.15 – Запуск puppet на puppet-Agent

Конфігуруємо Puppet Master, створюючи файл `/etc/puppet/manifests/site.pp` з таким змістом:

```
GNU nano 4.8 /etc/puppet/manifests/site.pp
node 'server.example.com' {
    include apache
}

class apache {
    package {'apache2':
        ensure => 'installed',
    }
    service {'apache2':
        ensure => 'running',
        enable => true,
    }
}
```

Рисунок 1.16 – Конфігурування Puppet Master

Тут використовується Puppet для налаштування веб-сервера Apache на сервері з іменем `server.example.com`. Створюється новий клас `apache`, який містить два налаштування: встановлення пакету `Apache2` та налаштування його сервісу

таким чином, щоб він завжди був запущеним та автоматично запускався під час завантаження системи.

Створюємо асоціації на кожному із серверів (рис. 1.17).

GNU nano 4.8 /etc	
127.0.0.1	localhost
127.0.1.1	student-VirtualBox
192.191.0.2	puppet-Agent

Рисунок 1.17 – Створення хосту на головному сервері

На другому сервері прописується хост для puppet-master (192.191.0.1).

Копіюємо файл site.pp з Puppet Master на Puppet Agent за допомогою наступної команди:

```
sudo scp /etc/puppet/manifests/site.pp student@puppet-Agent:/etc/puppet/manifests/site.pp
```

Після виконання команди вивід може бути різним, залежно від того, чи успішно відбулася передача файлу на віддалений сервер. Так як все пройшло успішно, вивід порожній.

Після виконання команди `sudo puppet agent --test` Puppet агент почне зв'язок з Puppet сервером і спробує завантажити конфігурацію. Вивід команди буде залежати від того, яку конфігурацію було застосовано і які модулі використовуються. Очікуваний результат – успішне завершення з виведенням повідомлень про те, як Puppet агент оновлює конфігурацію на сервері.

В даному випадку Puppet зчитає конфігурацію, яка включає налаштування для Apache, та перевірить, чи всі необхідні пакети встановлені та сервіс запущений. Вивід був наступний:

Info: Retrieving pluginfacts

Info: Retrieving plugin

Info: Caching catalog for student-VirtualBox

Info: Applying configuration version 'timestamp'

Notice: /Stage[main]/Apache/Package[apache2]/ensure: created

Notice: /Stage[main]/Apache/Service[apache2]/ensure: ensure changed 'stopped' to 'running'

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

Notice: Applied catalog in 0.10 seconds

При цьому Puppet спочатку виконує попередні завдання, такі як зчитування даних про плагіни та кешування каталогу для швидкого доступу до нього. Далі виводиться повідомлення про те, що застосовується конфігураційна версія, яка містить налаштування для Apache. На останок, Puppet повідомляє, які зміни були застосовані до системи, які сервіси були запущені або зупинені.

1.2 Аналіз можливих рішень та обґрунтування вибору Python

Вибір відповідної мови програмування має велике значення для успішної реалізації автоматизаційних задач. Різні мови програмування мають різні функціональні можливості, які можуть бути важливими для автоматизаційних задач. Наприклад, деякі мови можуть мати вбудовану підтримку для роботи з файлами, мережевими операціями або базами даних, що спрощує розробку скриптів автоматизації.

Доступність розширень, бібліотек та інструментів, які підтримують автоматизацію, є не менш важливим аспектом. Розробникам буде легше реалізувати автоматизаційні завдання, якщо мова програмування має широкий вибір корисних інструментів, які спрощують роботу. Важливо також, щоб мова мала активну спільноту розробників та підтримки. Підтримка спільноти може забезпечити доступ до документації, форумів, пакетів розширень та порад, що спростить розробку та підтримку автоматизаційних скриптів.

У деяких випадках, особливо при великому обсязі даних або складних завданнях, важливим аспектом буде виступати швидкодія мови програмування, яке буде впливати на продуктивність автоматизаційних рішень.

Нижче наведений перелік популярних мов програмування та скриптових мов, які можуть бути використані для автоматизації адміністрування Linux:

- Python;
- Bash;
- Perl;

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

- Ruby;
- PowerShell.

1.2.1 Perl

Perl (Practical Extraction and Reporting Language) – це мова програмування загального призначення, створена Ларрі Уоллом у 1987 році. Perl була спеціально розроблена для обробки текстових даних, забезпечуючи потужні можливості для маніпуляції рядками, регулярних виразів та обробки файлів.

Perl став популярним в середовищі Linux-адміністрування з кінця 1990-х років. З точки зору системного адміністрування Perl має кілька переваг:

- У мові Perl видно вплив різних оболонок Unix та мови C, які є інструментами, з якими працює багато системних адміністраторів;
- Perl доступний практично на всіх сучасних операційних системах і намагається забезпечити однорідний інтерфейс на кожній з них. Це важливо для багатоплатформенного адміністрування систем;
- в ньому є відмінні інструменти для маніпулювання текстом, доступу до баз даних та мережевого програмування, які є тригерами професії системного адміністратора;
- мова легко розширюється за допомогою ретельно побудованого механізму модулів;
- велика спільнота користувачів витратила безліч годин на створення модулів для практично будь-якого завдання [1].

Однак з метою відображення повної ситуації, необхідно зазначити, що Perl не є універсальним рішенням. Головні його недоліки:

- Perl включає нестійкий механізм об'єктно-орієнтованого програмування. У порівнянні з мовами, такими як Python або Ruby, реалізація об'єктно-орієнтованого підходу в Perl вважається менш надійною та стійкою;
- Perl не завжди є простим або внутрішньо узгодженим. Мова включає в себе багато незвичайних синтаксичних конструкцій, що може

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				23
Змн.	Арк.	№ докум.	Підпис	Дата		

ускладнювати завдання розробників щодо підтримки та розуміння Perl-коду. В порівнянні з іншими мовами програмування, Perl виявляє меншу рівномірність та прогнозованість.

Perl є потужною мовою, що може мати певні негативні наслідки. Perl може ставити під загрозу безпеку та стабільність системи, особливо якщо використовується неправильно або без належного контролю [1].

1.2.2 Ruby

Ruby – це об'єктно-орієнтована мова програмування загального призначення, яка відзначається простим синтаксисом. Вона була розроблена з акцентом на зручність і приємність програмування, що робить її популярним вибором серед розробників.

Ruby підтримує повну об'єктно-орієнтовану парадигму, де все є об'єктом, включаючи числа і навіть функції. Вона має потужну систему вбудованих класів і модулів, яка сприяє швидкому створенню і розширенню програмного коду.

Ruby також може бути використана для автоматизації адміністрування Linux. Завдяки своїй гнучкості та багатому набору бібліотек, Ruby надає можливості для швидкої і ефективної реалізації автоматизованих задач на Linux-системах.

Особливості Ruby, які можуть бути корисними для автоматизації адміністрування Linux, включають наступне:

- Ruby має потужні функції для роботи з файлами і директоріями, що дозволяють швидко і зручно виконувати операції з файлами, такі як створення, читання, запис і видалення файлів або каталогів;
- Ruby має багаті можливості для маніпулювання текстовими даними, що дозволяє легко здійснювати обробку логів, конфігураційних файлів та інших текстових даних, які зустрічаються при адмініструванні Linux;
- Ruby надає зручні механізми для виконання зовнішніх команд системи, таких як запуск командних скриптів або виконання системних

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				24
Змн.	Арк.	№ докум.	Підпис	Дата		

команд, що дозволяє автоматизувати виконання різноманітних операцій на рівні командного рядка Linux;

- Ruby має багатий набір бібліотек для роботи з мережевими протоколами, такими як HTTP, SSH, FTP тощо. Це дозволяє здійснювати автоматичний доступ до веб-серверів, виконувати операції по мережі та взаємодіяти з іншими системами;
- У Ruby запроваджено принципи об'єктно-орієнтованого програмування настільки повноцінно, що дозволяє використовувати надзвичайно потужні техніки метапрограмування [2].

Незважаючи на багато переваг, які має Ruby для автоматизації адміністрування Linux, варто враховувати і деякі його недоліки:

- Порівняно з іншими мовами програмування, такими як C або Go, Ruby може бути менш ефективним з точки зору швидкодії. Це особливо важливо при автоматизації завдань, які вимагають високої продуктивності або обробки великого обсягу даних;
- Ruby може вимагати більше пам'яті та обчислювальних ресурсів у порівнянні з іншими мовами програмування. Це може бути проблемою, особливо якщо виконувати автоматизовані задачі на системах з обмеженими ресурсами.

Ruby може мати меншу кількість бібліотек і фреймворків, специфічних для автоматизації адміністрування Linux. Це може обмежити доступність готових рішень і змусити розробника створювати власні інструменти.

1.2.3 PowerShell Core

PowerShell – це скриптова мова та оболонка командного рядка, розроблена компанією Microsoft для автоматизації адміністративних завдань та керування операційною системою Windows. Однак PowerShell можна інсталиувати і в різних дистрибутивах Linux. Більшість платформ і дистрибутивів Linux щорічно випускають основні випуски та забезпечують менеджер пакетів, який використовується для встановлення PowerShell [3].

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

Альтернативна реалізація PowerShell, яка є крос-платформним інструментом – PowerShell Core.

PowerShell Core дозволяє розробникам і адміністраторам Linux використовувати PowerShell для автоматизації адміністративних завдань на Linux-системах. Вона забезпечує доступ до широкого спектру функцій, таких як керування файловою системою, мережеві операції, взаємодія з системними ресурсами та багато іншого.

PowerShell Core має ряд переваг для автоматизації адміністрування Linux, включаючи:

- Можливість використовувати знайомий синтаксис PowerShell, що спрощує розробку і забезпечує сумісність з наявними скриптами та інструментами для Windows;
- підтримка багатфункціональних модулів та бібліотек, що дозволяє розширити можливості автоматизації та взаємодії з Linux-системами;
- інтеграція з іншими інструментами та середовищами розробки, що полегшує роботу з PowerShell Core на різних платформах;
- можливість використання механізму Remoting для віддаленого керування і виконання команд на віддалених Linux-системах.

Враховуючи ці функції, PowerShell Core є привабливим вибором для автоматизації адміністрування Linux, зокрема для тих, хто вже має досвід роботи з PowerShell на платформі Windows.

Незважаючи на переваги, PowerShell Core для автоматизації адміністрування Linux також має деякі недоліки, які варто врахувати:

1. Обмежена підтримка. Хоча PowerShell Core може бути використаний для автоматизації адміністрування Linux, його екосистема, модулі та ресурси не так широко розповсюджені, як для інших мов, таких як Python або Bash. Це може обмежити доступність готових рішень та зробити потрібні розробки більш затратними у часі.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				26
Змн.	Арк.	№ докум.	Підпис	Дата		

2. Навчання та досвід. Якщо команда адміністраторів Linux не має досвіду з PowerShell, це може вимагати часу та зусиль для оволодіння мовою та її особливостями. Це може вплинути на продуктивність та швидкість розробки автоматизованих задач.
3. Швидкодія. PowerShell Core може бути менш ефективним з точки зору швидкодії порівняно з іншими мовами, такими як Python або Go. Це особливо важливо при автоматизації завдань, які вимагають високої продуктивності або обробки великого обсягу даних.
4. Залежність від платформи. Використання PowerShell Core для автоматизації адміністрування Linux вимагає наявності та підтримки PowerShell на конкретній Linux-платформі. Це може обмежити переносимість скриптів та інструментів між різними дистрибутивами Linux.

1.2.4 Python

Python є потужною мовою програмування, яка широко використовується для автоматизації адміністрування Linux-систем. Вона надає розробникам гнучкість і зручність для виконання різноманітних завдань адміністрування, налаштування і моніторингу Linux-систем.

Ось деякі ключові переваги Python для автоматизації адміністрування Linux:

- Простота використання та вивчення;
- платформонезалежність;
- розширюваність;
- ліквідація проблем, пов'язаних із супроводом кода;
- підтримка ООП;
- багато функціональних можливостей стандартної бібліотеки;
- простий доступ до великої кількості пакетів інших розробників;
- наявність інтерактивного інтерпретатора.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				27
Змн.	Арк.	№ докум.	Підпис	Дата		

Python добре підходить для автоматизації різних аспектів адміністрування Linux, таких як конфігураційне управління, автоматизоване розгортання, моніторинг систем, автоматизоване тестування та резервне копіювання. Завдяки своїй простоті, зручності та потужним можливостям, Python є популярним вибором серед системних адміністраторів та розробників для автоматизації адміністративних завдань у середовищі Linux.

1.3 Інструменти для автоматизації адміністрування Linux на Python

Для адміністрування Linux за допомогою Python зазвичай використовують різноманітні модулі, які забезпечують доступ до системних ресурсів, можливість взаємодії з іншими програмами, мережеві операції та багато іншого.

Модулі на Python для адміністрування Linux – це збірники попередньо написаних функцій і класів, які надають можливості для автоматизації адміністративних завдань у Linux-середовищі. Ці модулі містять різні функції і методи, які допомагають працювати з файловою системою, мережевими налаштуваннями, процесами та іншими системними ресурсами.

Такі модулі можуть включати у себе вбудовані бібліотеки Python, такі як `os`, `sys`, `shutil`, `subprocess`, або сторонні модулі, такі як `paramiko` для виконання команд на віддалених машинах по SSH, `psutil` для моніторингу процесів та системних ресурсів або `netifaces` для роботи з мережевими інтерфейсами.

Завдяки наявності готових модулів Python для роботи з різними системними ресурсами, розробники можуть значно ефективніше та швидше створювати автоматизовані скрипти для адміністрування Linux-серверів та виконувати різноманітні завдання, такі як моніторинг системи, збір та аналіз лог-файлів, копіювання та переміщення файлів, налаштування мережових інтерфейсів тощо.

Модулі `OS (os)` та `Subprocess (subprocess)` – це два з основних модулів Python, які використовують для адміністрування Linux.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

Модуль `os` дозволяє взаємодіяти з операційною системою, виконувати команди `shell`, доступатися до змінних середовища та ін. Цей модуль включає багато функцій для роботи з файловою системою, дозволяє зчитувати та записувати файли, отримувати інформацію про процеси, мережеві інтерфейси та інше. Наприклад, з його допомогою можна виконувати наступні дії:

- отримати список файлів та директорій в поточній директорії;
- створити нову директорію;
- отримати інформацію про систему (таку як ім'я операційної системи, кількість процесорів тощо).

Основні команди модуля `os` включають:

- `os.getcwd()`: повертає поточну робочу директорію.
- `os.chdir(path)`: змінює поточну робочу директорію на вказану.
- `os.listdir(path)`: повертає список файлів та директорій в заданій директорії.
- `os.mkdir(path)`: створює нову директорію з вказаним ім'ям.
- `os.rmdir(path)`: видаляє директорію з вказаним ім'ям (директорія повинна бути порожньою).
- `os.remove(path)`: видаляє файл з вказаним ім'ям.
- `os.rename(src, dst)`: перейменовує файл або директорію з `src` на `dst`.
- `os.system(command)`: виконує команду в командному рядку.
- `os.getenv(varname)`: повертає значення змінної середовища з заданим ім'ям.
- `os.getpid()`: повертає ідентифікатор поточного процесу.
- `os.kill(pid, signal)`: надсилає сигнал до процесу з вказаним ідентифікатором.

Це лише деякі з основних команд, які можна виконувати з модулем `os`. Модуль також містить багато інших корисних функцій для взаємодії з операційною системою.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

Модуль Subprocess дозволяє запускати зовнішні процеси з Python-скрипту та зчитувати та записувати дані в їхній вхід та вихід. Цей модуль надає більш гнучкий та безпечний спосіб виконання зовнішніх команд. Він дозволяє створювати підпроцеси, перенаправляти вхід та вихід, запускати команди в фоновому режимі та інше.

Наприклад, з його допомогою можна виконувати наступні дії:

- запустити команду в shell та зчитати її вихід;
- запустити багато команд та підключитися до їх вхідного та вихідного потоків;
- запустити команду в фоновому режимі та отримати її PID (ідентифікатор процесу).

Модуль socket є одним з основних модулів Python для роботи з мережевими операціями. Він надає інтерфейс для створення сокетів, що дозволяє здійснювати з'єднання з віддаленими серверами, відправляти та отримувати дані через мережу, використовуючи різні протоколи (наприклад, TCP або UDP).

Цей модуль є досить потужним і дозволяє розробникам створювати складні мережеві додатки, такі як веб-сервери, клієнти для мережевих протоколів, інструменти для тестування мережевих додатків тощо. За допомогою модуля socket можна отримати доступ до різноманітних функцій мережевої роботи і керувати ними в програмах на Python.

Деякі основні команди, що можна виконувати за допомогою цього модуля:

- socket(): створює новий сокет;
- bind(): прив'язує сокет до конкретного IP-адреси та порту;
- listen(): дозволяє серверу приймати вхідні з'єднання;
- accept(): приймає вхідне з'єднання та створює новий сокет для обміну даними;
- connect(): встановлює з'єднання з віддаленим сервером;
- send(): відправляє дані через сокет;
- recv(): отримує дані через сокет;

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

– `close()`: закриває сокет.

Ці команди дозволяють створювати мережеві додатки, які можуть взаємодіяти з іншими пристроями через мережу. Для більш детальної інформації можна переглянути документацію модуля `socket` в Python.

Висновки до першого розділу

В розділі було проведено аналіз підсистем адміністрування ОС Linux, таких як `Systemd` і `Puppet`, з метою визначення підходів до автоматизації адміністрування. Визначено задачі, які підлягають автоматизації.

У розділі також проведено аналіз можливих рішень для автоматизації, зокрема мов програмування `Perl`, `Ruby`, `PowerShell Core` і `Python`. Враховуючи широкий спектр функціональності, зручний синтаксис, багату екосистему і популярність серед спільноти розробників, було прийняте рішення використовувати мову програмування `Python` як основну для реалізації проекту.

`Python` має декілька переваг, які роблять його ідеальним вибором для автоматизації адміністрування Linux. Переваги включають простоту використання та вивчення, платформонезалежність, розширюваність, ліквідацію проблем, пов'язаних із супроводом коду, підтримку ООП, багато функціональних можливостей стандартної бібліотеки, простий доступ до великої кількості пакетів інших виробників та наявність інтерактивного інтерпретатора.

Ці переваги роблять `Python` потужним і ефективним інструментом для автоматизації різноманітних задач адміністрування Linux, спрощуючи розробку, збір та обробку даних, автоматичне виконання операцій та взаємодію з різними системними ресурсами.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				31
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2 ПРОЕКТУВАННЯ ПІДСИСТЕМИ АВТОМАТИЗАЦІЇ АДМІНІСТРУВАННЯ ОС LINUX

2.1 Визначення задач для автоматизації

Автоматизація адміністрування операційних систем Linux передбачає розробку та впровадження різних рішень, що спрощують та автоматизують рутинні задачі, пов'язані з управлінням та підтримкою Linux-систем.

При визначенні задач для автоматизації адміністрування ОС Linux слід враховувати наступні аспекти:

1. Управління користувачами та доступом. Розробка автоматизованих рішень для створення, видалення та керування обліковими записами користувачів, налаштування прав доступу та контролю привілеїв. Це може включати автоматичне створення нових користувачів, налаштування групової політики, управління пароллями та ідентифікаційними даними.
2. Керування конфігураціями. Розробка засобів для автоматичного налаштування та управління конфігураціями Linux-систем. Це може включати створення шаблонів конфігураційних файлів, їх автоматичне розгортання та оновлення, а також контроль версій конфігурацій.
3. Управління мережею. Розробка інструментів для автоматичного налаштування та управління мережевими інтерфейсами, налаштування IP-адрес, маршрутизації, фаєрволу та інших мережевих параметрів.
4. Моніторинг та аудит. Розробка рішень для автоматичного моніторингу стану системи, ресурсів, сервісів та подій. Це може включати збір та аналіз журналів, сповіщення про аварійні ситуації, моніторинг ресурсів та продуктивності.
5. Резервне копіювання та відновлення. Розробка автоматизованих засобів для створення та відновлення резервних копій системи, файлів, баз

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

даних та інших важливих ресурсів. Це може включати розробку резервного планування, автоматичне створення резервних копій, контроль цілісності даних та відновлення в разі втрати.

6. Автоматичне встановлення та розгортання. Розробка інструментів для автоматичного встановлення та розгортання Linux-систем, додатків та сервісів. Це може включати розробку скриптів для автоматичного встановлення пакетів, налаштування параметрів системи та сервісів.
7. Забезпечення безпеки. Розробка засобів для автоматичного сканування, виявлення та виправлення вразливостей, контроль доступу та аудиту безпеки системи. Це може включати розробку скриптів для автоматичного встановлення оновлень, моніторингу атак та налаштування безпекових політик.

Вибір конкретних задач для автоматизації адміністрування Linux залежить від потреб і вимог організації або проекту.

Розглянемо кілька конкретних задач, які варто автоматизувати в контексті адміністрування Linux і пояснимо, чому вони мають велике значення та є інноваційними.

1. Автоматичне розгортання і налаштування нових серверів. Однією з найперших задач адміністратора є розгортання нових серверів та їх налаштування згідно з вимогами. Автоматизація цього процесу дозволить швидко створювати та налаштовувати нові сервери, зменшуючи час і зусилля, необхідні для виконання цих завдань вручну.
2. Моніторинг і керування системними ресурсами. Автоматизація моніторингу системних ресурсів, таких як процесор, пам'ять, диски та мережа, дозволить адміністраторам отримувати швидку інформацію про поточний стан системи та реагувати на проблеми автоматично. Наприклад, можна налаштувати автоматичне сповіщення про перевищення обсягів використання ресурсів, автоматичне масштабування ресурсів або навіть автоматичне виправлення проблем.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

3. Резервне копіювання і відновлення даних. Забезпечення резервного копіювання і відновлення важливих даних є критичним завданням для забезпечення безпеки та надійності системи. Автоматизація цього процесу дозволить налаштувати регулярне автоматичне створення резервних копій, їх зберігання та можливість швидкого відновлення даних в разі потреби.
4. Автоматизоване встановлення та оновлення програмного забезпечення. Ручне встановлення та оновлення програмного забезпечення на кожному сервері може бути часомістким та призводити до помилок. Автоматизація цього процесу дозволить налаштувати централізоване управління пакетами, дистрибутивами та оновленнями, що спростить процес установки та забезпечить консистентність між серверами.
5. Забезпечення безпеки. Автоматизація задач забезпечення безпеки, таких як сканування вразливостей, встановлення оновлень, налаштування брандмауерів та контроль доступу, є критичним для захисту системи. Це дозволить виявляти та виправляти вразливості автоматично, реагувати на загрози швидко та забезпечувати високий рівень безпеки системи.

Кожна з цих задач має важливе значення для адміністрування Linux, оскільки спрощує та прискорює процеси, зменшує витрати часу та зусиль, покращує безпеку та забезпечує надійність системи. Крім того, автоматизація цих задач може бути інноваційною, оскільки дозволяє використовувати нові технології, інструменти та підходи для розв'язання проблем, які раніше вирішувалися лише вручну або з використанням обмежених і неефективних рішень.

Автоматичне розгортання і налаштування нових серверів є важливим кроком у забезпеченні ефективного адміністрування Linux-серверів. Ця задача вимагає багато часу і зусиль, коли вона виконується вручну. Однак, за допомогою автоматизації на базі Python, можна значно спростити і прискорити цей процес.

Автоматичне розгортання серверів включає створення нових віртуальних машин або фізичних серверів, установку операційної системи та необхідного

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				34
Змн.	Арк.	№ докум.	Підпис	Дата		

програмного забезпечення, налаштування мережевих параметрів, налаштування безпеки та доступу, а також встановлення додаткових компонентів та сервісів.

При автоматизації цього процесу можна використовувати різні інструменти та технології, такі як:

- Інструменти для управління інфраструктурою, такі як Ansible, Puppet або Chef, які дозволяють створювати описові файли або скрипти для розгортання та налаштування серверів;
- шаблони віртуальних машин або контейнерів, які містять заздалегідь визначені конфігураційні параметри та програмне забезпечення, що можуть бути використані для швидкого розгортання нових серверів;
- скрипти на Python для автоматизації конкретних кроків процесу, наприклад, створення віртуальних машин, налаштування мережевих інтерфейсів або установка необхідного програмного забезпечення;
- інструменти контейнеризації, такі як Docker або Kubernetes, які дозволяють створювати і управляти контейнерами з попередньо налаштованими середовищами, що прискорює процес розгортання.

Новизна та інноваційність полягає в наступному:

- Використання інфраструктури як коду (Infrastructure as Code), де конфігурація серверів описується у вигляді коду, що дозволяє швидко впроваджувати зміни та контролювати конфігурацію за допомогою систем контролю версій;
- використання контейнеризації для розгортання додатків та сервісів у віртуальних середовищах, що дозволяє швидко масштабувати та управляти інфраструктурою;
- використання хмарних сервісів, таких як AWS, Google Cloud або Azure, для автоматичного розгортання та керування серверами;
- використання машинного навчання або штучного інтелекту для аналізу логів та виявлення аномалій у системі, що дозволяє автоматично реагувати на проблеми та забезпечувати високу доступність.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

Ці інноваційні підходи можуть надати значні переваги в термінах ефективності, безпеки та надійності адміністрування Linux-серверів.

2.2 Розробка плану автоматизації

Розподіл завдань на модулі та функціональні блоки є важливою частиною розробки плану автоматизації. Це допомагає створити структуровану організацію проекту. Кожен модуль або функціональний блок відповідає за конкретну функцію або задачу, що спрощує розробку, тестування та зрозумілість коду.

Крім цього, розподіл сприяє модульній розробці, де окремі модулі можуть бути розроблені та тестовані незалежно один від одного. Це дає можливість швидко знаходити та виправляти помилки, а також змінювати або доповнювати функціональність окремих модулів без впливу на інші частини системи.

Розподіл завдань на модулі та функціональні блоки сприяє повторному використанню коду. Коли кожен елемент виконує конкретну функцію, його можна легко використовувати в інших проектах або модифікувати для вирішення схожих задач.

Для досягнення мети автоматизації адміністрування Linux, визначено наступні модулі та функціональні блоки:

- Модуль «Розгортання і налаштування нових серверів».

Цей модуль відповідає за створення та налаштування нових серверів. Його призначення – забезпечити базову інфраструктуру для подальшої автоматизації. Основні функції модуля включають встановлення операційної системи на віддаленій машині, налаштування мережі та передача файлів. Його вихідними даними будуть готові сервери з налаштованою базовою інфраструктурою. Вхідними даними слугуватимуть посилання на завантаження ОС для додавання їх в конфігураційні файли, а також назви доменів.

- Модуль «Адміністрування».

Модуль «Адміністрування» відповідає за керування користувачами і забезпечує належне функціонування аспектів системи, пов'язаних з управлінням користувачами. Основна функція цього модуля полягає в налаштуванні та управлінні обліковими записами користувачів на системі.

Для ефективної автоматизації адміністрування Linux і планування послідовності виконання модулів або функціональних блоків, слід враховувати їх залежності та послідовність операцій.

З урахуванням залежності модуля «Адміністрування» від модуля «Розгортання і налаштування нових серверів», можна спланувати наступну послідовність виконання автоматизації:

1. Виконати модуль «Розгортання і налаштування нових серверів»:

- Розгортання сервера PXE;
 - 1. Розгортання сервера DHCP;
 - 2. розгортання сервера TFTP;
 - 3. розгортання сервера NFS;
 - 4. конфігурування сервера PXE.
- встановити операційну систему на нових серверах;
- передати файли для конфігурації ОС;
- забезпечити базову безпеку серверів.

2. Виконати модуль «Адміністрування»:

- Додати нових користувачів;
- виконати управління користувачами.

3. Виконати модуль «Розгортання і налаштування нових серверів»:

- Створити сервер Apache;
- додати віртуальні хости.

Ця послідовність дозволить спочатку розгорнути і налаштувати нові сервери, включаючи їх конфігураційні файли, а потім налаштувати управління доступом та налаштування додаткових серверів.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

Після успішного виконання основних модулів можна додати включення додаткових модулів або функціональних блоків для інших аспектів автоматизації, таких як моніторинг системи або керування доступом.

2.3 Вимоги до підсистеми автоматизації

Для досягнення успішної автоматизації адміністрування Linux, важливо визначити наступні вимоги до підсистеми:

- Надійність – підсистема повинна бути стабільною та надійною, здатною виконувати свої функції без збоїв або помилок. Вона повинна виявляти і обробляти помилки та незвичайні ситуації, забезпечуючи безперебійну роботу;
- швидкодія – підсистема повинна працювати ефективно та швидко, забезпечуючи швидкість виконання запитів та операцій. Вона повинна бути оптимізована для швидкого доступу до ресурсів, зменшуючи затримки та час відгуку.
- гнучкість – підсистема повинна бути гнучкою та адаптованою до різних потреб та вимог. Вона повинна підтримувати різноманітні налаштування, параметри та можливості, щоб задовольнити потреби користувачів;
- безпека – підсистема повинна мати високий рівень безпеки, забезпечуючи захист від несанкціонованого доступу та збереження конфіденційності, цілісності та доступності даних. Вона повинна використовувати механізми автентифікації, авторизації та шифрування для захисту важливої інформації;
- масштабованість – підсистема повинна бути масштабованою, здатною працювати зі зростанням обсягу даних та навантаження. Вона повинна мати можливість розширення та горизонтального масштабування, щоб задовольнити потреби інфраструктури, що розгортається;

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				38
Змн.	Арк.	№ докум.	Підпис	Дата		

- підсистема повинна бути простою у використанні та зрозумілою для адміністраторів. Вона повинна мати зручний інтерфейс, налаштування та управління, які легко розуміються та використовуються.

Крім того, підсистема повинна бути сумісною з різними версіями та дистрибутивами Linux. Це є важливою вимогою, оскільки існує багато різних варіацій та конфігурацій Linux-систем, що використовуються в різних організаціях і проектах.

Підсистема повинна дотримуватись стандартів, які встановлені для Linux-середовища. Це можуть бути стандарти, такі як POSIX (Portable Operating System Interface), що визначає інтерфейси для програм та операційних систем, або стандарти, які стосуються мережевої взаємодії, такі як TCP/IP або IPv6.

Також важлива підтримка протоколів, що використовуються в Linux-середовищі. Наприклад, для взаємодії з мережею, підсистема повинна підтримувати протоколи, такі як TCP (Transmission Control Protocol) або UDP (User Datagram Protocol).

Необхідно передбачити роботу з різними форматами файлів, що використовуються в Linux-середовищі. Це можуть бути текстові файли з конфігураціями, бінарні файли, бази даних тощо. Підсистема повинна бути здатною розпізнавати та обробляти ці формати файлів.

Важливо, щоб підсистема здатна була працювати з різними дистрибутивами Linux, такими як Ubuntu, CentOS, Debian, Fedora тощо. Кожен дистрибутив може мати свої особливості та конфігураційні файли, і підсистема повинна бути здатною взаємодіяти з цими різними дистрибутивами.

Забезпечення сумісності підсистеми з різними версіями та дистрибутивами Linux допомагає забезпечити, що підсистема може працювати на будь-якому Linux-середовищі і взаємодіяти з іншими компонентами і системами, що також використовують Linux.

Ці вимоги до підсистеми допоможуть забезпечити ефективну та надійну автоматизацію адміністрування Linux, зменшуючи витрати часу та зусиль.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

2.4 Проектування мережі

Для реалізації та тестування роботи підсистеми було побудовано мережі, топологію якої зображено на рис. 2.1.

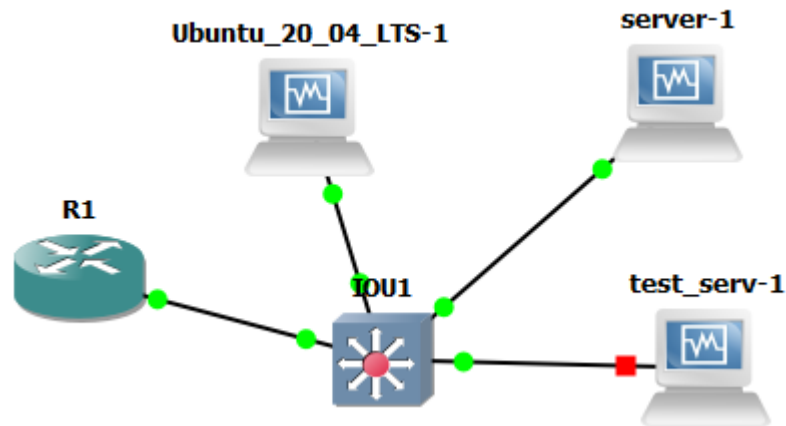


Рисунок 2.1 – Схема мережі

Схему адресації мереж наведено у табл. 2.1. Під час розрахунку враховувалося, що кожному інтерфейсу маршрутизатора мережі виділяється по одній IP-адресі.

Таблиця 2.1 – Схема IP-адресації пристроїв

Мережа/пристрій	Інтерфейс/Мережний адаптер/Шлюз	MAC-адреса	IP - адреса	Маска	Префікс
Мережа А	—		195.191.14.0	255.255.255.0	/24
Комутатор SW-191-14-1	Інтерфейс VLAN 1		195.191.14.2	255.255.255.0	/24
	Шлюз за замовчуванням	—	195.191.14.1	—	—
Маршрутизатор R-191-14-1	Інтерфейс Fa0/0	CA:01:35:40:00:00	195.191.14.1	255.255.255.0	/24
Сервер Ubuntu_22.04.2_LTS	Мережний адаптер	08:00:27:12:44:5E	195.191.14.4	255.255.255.0	/24
	Шлюз за замовчуванням	—	195.191.14.1	—	—
Робоча станція Ubuntu_20.04_LTS	Мережний адаптер	08:00:27:2C:91:7D	195.191.14.5	255.255.255.0	/24
	Шлюз за замовчуванням	—	195.191.14.1	—	—
Робоча станція	Мережний адаптер	08:00:27:25:3E:07	DHCP		
	Шлюз за замовчуванням	—			

Висновки до другого розділу

На даному етапі розробки курсового проекту було проведено проектування підсистеми автоматизації адміністрування ОС Linux.

Розроблений план автоматизації, який описує послідовність дій та процедури, необхідні для виконання автоматизованих завдань. Цей план визначає, які кроки потрібно виконати для успішного впровадження автоматизованої системи адміністрування ОС Linux. В результаті було визначено необхідність розробки двох модулів для розгортання серверів та управління користувачами.

Встановлені вимоги до підсистеми автоматизації. Вимоги включають функціональні та нефункціональні вимоги, які мають бути виконані системою. Функціональні вимоги визначають, які функції та можливості має мати підсистема автоматизації, наприклад, установка та налаштування серверів DHCP, TFTP і NFS, PXE та Apache. Нефункціональні вимоги визначають вимоги до продуктивності, безпеки, надійності та інших аспектів системи.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПІДСИСТЕМИ АДМІНІСТРУВАННЯ ОС LINUX

3.1 Автоматизація створення PXE-сервера та завантажувальних кодів

Для автоматизованих процесів було створено машину із IP-адресою 195.191.14.4 на базі сервера Linux Ubuntu 22.04.2 LTS із встановленим інтерфейсом за допомогою Gnome Jammy Jellyfish.

Для автоматизації процесу розгортання PXE-сервера було створено код на Python з окремими функціями та головною функцією, яка викликає ці функції у правильній послідовності. Кожна функція відповідає одному з кроків розгортання.

Спочатку виконується функція `create_directory_structure()` (рис. 3.1). Цей фрагмент коду створює наступну структуру каталогів для сервера PXE:

- `/pxeboot`
- `config/`
- `firmware/`
- `os-images/`

В каталозі `/pxeboot/config/` знаходяться всі файли конфігурації, необхідні для налаштування завантаження iPXE.

Каталог `/pxeboot/firmware/` містить файли мікропрограм, необхідних для завантаження iPXE.

У каталозі `/pxeboot/os-images/` створюється окремий підкаталог для кожного дистрибутиву Linux, який буде завантажуватись через PXE. У цих підкаталогах зберігаються образи ISO дистрибутивів Linux. Наприклад, якщо необхідно завантажити Ubuntu Desktop 22.04 LTS через PXE, буде створено каталог `ubuntu-22.04-desktop-amd64/` у каталозі `/pxeboot/os-images/` і розміщено у ньому вміст ISO-образу Ubuntu Desktop 22.04 LTS.

Функція має перевіряти, чи існує кожен з каталогів у списку `directories`. Якщо якийсь каталог не існує, то він створюється за допомогою функції `os.makedirs()`.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

```
def create_directory_structure():
    directories = [
        '/pxeboot/config',
        '/pxeboot/firmware',
        '/pxeboot/os-images'
    ]
    for directory in directories:
        if not os.path.exists(directory):
            os.makedirs(directory)
```

Рисунок 3.1 – Функція create_directory_structure()

Функція compile_ipxe (рис. 3.2) створена для компілювання інсталяційних файлів та налаштування IPXE (Preboot Execution Environment).

```
def compile_ipxe():
    subprocess.call(['sudo', 'apt', 'update'])
    subprocess.call(['sudo', 'apt', 'install', 'build-essential', 'liblzma-dev',
        'isolinux', 'git', '-y'])

    os.chdir('/home/mariana/Downloads')
    subprocess.call(['git', 'clone', 'https://github.com/ipxe/ipxe.git'])
    os.chdir('/home/mariana/Downloads/ipxe/src')

    bootconfig_content = '''#!ipxe

dhcp

chain tftp://195.191.14.4/config/boot.ipxe
...

with open('bootconfig.ipxe', 'w') as f:
    f.write(bootconfig_content)

subprocess.call(['make', 'bin/ipxe.pxe', 'bin/undionly.kpxe', 'bin/undionly.kkpxe',
    'bin/undionly.kkkpxe', 'bin-x86_64-efi/ipxe.efi', 'MBED=bootconfig.ipxe'])
```

Рисунок 3.2 – Функція compile_ipxe

В функції відбувається наступне:

- Виконується команда `subprocess.call(['sudo', 'apt', 'update'])`, яка оновлює локальний пакетний менеджер АРТ, щоб отримати найновіші версії пакетів;

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

- виконується команда `subprocess.call(['sudo', 'apt', 'install', 'build-essential', 'liblzma-dev', 'isolinux', 'git', '-y'])`, яка встановлює необхідні пакети для компіляції IPXE, зокрема `build-essential`, `liblzma-dev`, `isolinux` і `git`;
- змінюється поточний робочий каталог на `/home/mariana/Downloads`;
- виконується команда `subprocess.call(['git', 'clone', 'https://github.com/ipxe/ipxe.git'])`, яка клонує репозиторій IPXE;
- змінюється поточний робочий каталог на `/home/mariana/Downloads/ipxe/src`;
- встановлюється вміст змінної `bootconfig_content`, яка містить конфігурацію завантаження IPXE. Цей конфігураційний файл (`bootconfig.ipxe`) містить команду `dhcp`, яка запускає автоматичне отримання налаштувань мережі через DHCP, а також команду `chain tftp://195.191.14.4/config/boot.ipxe`, яка переходить до іншого IPXE скрипту за допомогою TFTP (Trivial File Transfer Protocol);
- відкривається файл `bootconfig.ipxe` для запису, і йому присвоюється вміст `bootconfig_content`;
- виконується команда `subprocess.call(['make', 'bin/ipxe.pxe', 'bin/undionly.kpxe', 'bin/undionly.kkpxe', 'bin/undionly.kkkpxe', 'bin-x86_64-efi/ipxe.efi', 'EMBED=bootconfig.ipxe'])`, яка запускає компіляцію IPXE і генерує різні інсталяційні файли з вбудованим `bootconfig.ipxe`. У результаті компіляції створюються файли `bin/ipxe.pxe`, `bin/undionly.kpxe`, `bin/undionly.kkpxe`, `bin/undionly.kkkpxe` і `bin-x86_64-efi/ipxe.efi`.

Отже, функція `compile_ipxe` оновлює пакети, встановлює необхідні компіляційні залежності, клонує репозиторій IPXE, створює конфігураційний файл IPXE, а потім компілює IPXE і генерує відповідні інсталяційні файли.

Функція `copy_firmwares` (рис. 3.3) призначена для копіювання скомпільованих файлів IPXE до каталогу `/pxeboot/firmware/`. В функції змінюється поточний робочий каталог на `/home/mariana/Downloads/ipxe/src`, щоб переконатися, що

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

наступні команди виконуються у відповідному контексті. Далі копіюються файли `ipxe.pxe`, `undionly.kpxe`, `undionly.kkpxe`, `undionly.kkkpxe` та `ipxe.efi` з поточного робочого каталогу до каталогу `/pxeboot/firmware/`.

```
def copy_firmwares():
    os.chdir('/home/mariana/Downloads/ipxe/src')
    subprocess.call(['sudo', 'cp', '-v', 'bin/ipxe.pxe', 'bin/undionly.kpxe',
'bin/undionly.kkpxe', 'bin/undionly.kkkpxe', 'bin-x86_64-efi/ipxe.efi',
'/pxeboot/firmware/'])
```

Рисунок 3.3 – Функція `copy_firmwares`

Копіювання цих файлів до `/pxeboot/firmware/` важливе для налаштування IPXE, оскільки вони використовуються під час завантаження системи через мережу.

3.1.1 Встановлення та налаштування сервера DHCP і TFTP

Для належної роботи завантаження PXE необхідний робочий сервер DHCP і TFTP, який може бути запущений на комп'ютері користувача. Серед різноманітних серверних програм для цих цілей широко використовується `dnsmasq`. `Dnsmasq`, хоча в основному відомий як сервер DNS і DHCP, також може бути налаштований як сервер TFTP.

Варто зазначити, що в Ubuntu 22.04.2 LTS `dnsmasq` не встановлено за замовчуванням, але це програмне забезпечення доступне у офіційному сховищі пакетів Ubuntu 22.04.2. Встановити `dnsmasq` дуже просто, використовуючи зручний менеджер пакетів APT, який надається в Ubuntu.

Для встановлення та налаштування конфігурації створена функція `install_configure_dnsmasq()`. В ній встановлюється пакет `dnsmasq` на системі з правами суперкористувача та створюється новий файл конфігурації `dnsmasq.conf` (рис. 3.4). Старий файл перейменовується.

У файлі конфігурації `dnsmasq.conf` задаються різні параметри, які визначають налаштування DHCP і TFTP сервера, такі як інтерфейс мережі, діапазон IP-адрес для DHCP, опції DHCP, налаштування TFTP та інші.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

Після прописання конфігурації, файл зберігається, а служба dnsmasq перезапускається для застосування змін.

```
dnsmasq_content = '''interface=enp0s3
bind-interfaces
domain=server.local
dhcp-range=ens38,195.191.14.50,195.191.14.200,255.255.255.0,8h
dhcp-option=option:router,195.191.14.1
enable-tftp
tftp-root=/pxeboot
dhcp-match=set:bios-x86,option:client-arch,0
dhcp-boot=tag:bios-x86,firmware/ipxe.pxe
dhcp-match=set:efi-x86_64,option:client-arch,7
dhcp-match=set:efi-x86_64,option:client-arch,9
dhcp-boot=tag:efi-x86_64,firmware/ipxe.pxe
...
'''
```

Рисунок 3.4 – Конфігурація файлу dnsmasq.conf

Важливо зауважити, що код використовує модуль subprocess для виконання команд в терміналі з допомогою Python. Це дозволяє виконувати команди з правами суперкористувача (sudo) для встановлення пакету та управління системними службами.

3.1.2 Встановлення та налаштування сервера NFS

У Ubuntu Desktop 22.04 LTS та деяких інших дистрибутивах Linux, таких як Fedora, CentOS/RHEL, для завантаження PXE використовується компонент під назвою casper. Особливість casper полягає в тому, що він підтримує лише завантаження через протокол NFS (Network File System). Це означає, що для успішного PXE-завантаження Ubuntu Desktop 22.04 LTS та інших сумісних дистрибутивів Linux потрібний сервер NFS, який надає доступ до відповідних файлів через мережу.

Сервер NFS виконує важливу роль у PXE-завантаженні, дозволяючи обмінюватися файлами між PXE-клієнтом і сервером. Він забезпечує доступ до образів операційних систем та інших необхідних файлів, що потрібні для живого інсталяційного середовища.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

Функція `install_configure_nfs_server` (рис. 3.5) виконує налаштування та встановлення сервера NFS (Network File System).

```
def install_configure_nfs_server():
    subprocess.call(['sudo', 'apt', 'install', 'nfs-kernel-server'])

    nfs_config =
"/pxeboot\t*(ro,sync,no_wdelay,insecure_locks,no_root_squash,insecure,no_subtree_check)
"

    with open('/etc/exports', 'w') as f:
        f.write(nfs_config)

    subprocess.call(['sudo', 'exportfs', '-av'])
```

Рисунок 3.5 – Функція `install_configure_nfs_server`

Основні дії, які вона виконує, перераховані нижче:

1. Встановлення пакету `nfs-kernel-server`, який є необхідним для налаштування сервера NFS на системі.
2. Відкриття файлу `/etc/exports` за допомогою текстового редактора `nano`. У цьому файлі будуть вказані директорії та параметри, які будуть доступні через NFS.
3. Встановлення налаштувань для NFS шляху `/pxeboot` з наступними параметрами: `"ro"` (дозвіл тільки для читання), `"sync"` (синхронне записування), `"no_wdelay"` (відключення затримки запису), `"insecure_locks"` (використання небезпечних блокувань), `"no_root_squash"` (не змінювати ідентифікатор користувача `"root"`), `"insecure"` (дозвіл на монтування з непривілейованого порту), `"no_subtree_check"` (відключення перевірки піддерев).
4. Додавання налаштувань до файлу `/etc/exports` за допомогою методу `write` з модифікатором `a`, що додає новий рядок.
5. Виклик команди `exportfs -av` для оновлення і виконання налаштувань NFS.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

3.1.3 Завантаження та створення конфігураційних файлів ОС

Функції `configure_ipxe_pxe_boot_ubuntu()` та `configure_ipxe_pxe_boot()` (рис. 3.6) мають на меті налаштувати PXE-завантаження для різних дистрибутивів Linux, зокрема Ubuntu.

```
def configure_ipxe_pxe_boot_ubuntu():
    iso_url = input("Введіть URL для завантаження ISO-образу: ")
    iso_name_match = re.search(r'([^\s/]+)\.iso$', iso_url)
    if iso_name_match:
        iso_name = iso_name_match.group(1)
    else:
        print("Невдалось отримати назву ISO-образу з URL")
        return

    iso_path = os.path.expanduser(f"/home/mariana/Downloads/{iso_name}.iso")
    pxeboot_path = f"/pxeboot/os-images/{iso_name}"

    subprocess.call(["wget", iso_url, "-O", iso_path])
    subprocess.call(["sudo", "mount", "-o", "loop", iso_path, "/mnt"])
    subprocess.call(["sudo", "mkdir", "-pv", pxeboot_path])
    subprocess.call(["sudo", "rsync", "-avz", "/mnt/", pxeboot_path])
    subprocess.call(["sudo", "umount", "/mnt"])
    os.remove(iso_path)
    configure_boot_ipxe(iso_name)
```

Рисунок 3.6 – Функція `configure_ipxe_pxe_boot_ubuntu`

Обидві функції починаються з отримання URL-адреси ISO-образу, який необхідно завантажити. Після цього проводиться пошук назви ISO-образу з отриманого URL-адреси.

Далі виконується завантаження ISO-образу за допомогою команди `wget` і збереження його у вказаному шляху. Після завантаження образу, виконуються такі дії:

1. Монтується ISO-образ на тимчасову директорію `/mnt` за допомогою команди `sudo mount`.
2. Створюється директорія `pxeboot_path`, куди будуть скопійовані файли з ISO-образу за допомогою команди `sudo mkdir`.

3. Виконується копіювання файлів з монтованого образу до `pxeboot_path` за допомогою команди `sudo rsync`.
4. Знімається монтування образу командою `sudo umount`.
5. Видаляється завантажений ISO-образ командою `os.remove`.
6. Далі викликається функція `configure_boot_ipxe()`, яка налаштовує завантаження через iPXE для відповідного ISO-образу.

Різниця між цими двома функціями полягає у використанні конкретного дистрибутиву Linux. Функція `configure_ipxe_pxe_boot_ubuntu()` спеціалізується на налаштуванні PXE-завантаження для Ubuntu, тоді як функція `configure_ipxe_pxe_boot()` загальна і може використовуватись для налаштування PXE-завантаження для будь-якого іншого дистрибутиву Linux.

Ці функції допомагають автоматизувати процес налаштування PXE-завантаження, що спрощує і прискорює процес розгортання операційних систем на віддалених комп'ютерах через мережу.

Функція `configure_boot_ipxe_ubuntu()` конкретно налаштовує `boot.ipxe` для дистрибутиву Ubuntu. Вона встановлює стандартний вміст для PXE-завантаження Ubuntu Desktop 22.04 LTS. Вміст містить налаштування сервера, пункт меню для Ubuntu 22.04 LTS та відповідні параметри завантаження через протокол NFS (рис. 3.7).

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

```

boot_ipxe_content = '''#!ipxe

set server_ip 195.191.14.4
set root_path /pxeboot

menu Select an OS to boot
item ubuntu-22.04.2-desktop-amd64          Install Ubuntu Desktop 22.04.2 LTS
choose --default exit --timeout 10000 option && goto ${option}

:ubuntu-22.04.2-desktop-amd64
set os_root os-images/ubuntu-22.04.2-desktop-amd64
kernel tftp://${server_ip}/${os_root}/casper/vmlinuz
initrd tftp://${server_ip}/${os_root}/casper/initrd
imgargs vmlinuz initrd=initrd boot=casper maybe-ubiquity netboot=nfs ip=dhcp
nfsroot=${server_ip}:${root_path}/${os_root} quiet splash ---
boot
...

```

Рисунок 3.7 – Вміст конфігураційного файлу

Функція `configure_boot_ipxe(iso_name)` призначена для налаштування файлу `boot.ipxe`, який використовується для вибору операційної системи при PXE-завантаженні.

Основні кроки функції включають:

1. Зчитування вмісту файлу `boot.ipxe` за допомогою команди `with open()`.
2. Запитування користувачем назви для нового пункту меню.
3. Додавання нового пункту меню до вмісту файлу `boot.ipxe` за допомогою регулярного виразу (рис. 3.8).
4. Зчитування вмісту файлу, що містить користувацький код.
5. Додавання завантажувального коду в кінець файлу `boot.ipxe`.
6. Запис оновленого вмісту у файл `boot.ipxe`.

```

pattern = r"menu Select an OS to boot\s+item\s+(\w+)\s+Install\s+(\w+)"
replacement = rf"menu Select an OS to boot\nitem \1 Install \2\nitem {iso_name}
Install {name}"
updated_content = re.sub(pattern, replacement, boot_ipxe_content)

```

Рисунок 3.8 – Додавання нового пункту меню до вмісту файлу `boot.ipxe`

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

Основна різниця між цими двома функціями полягає у вмісті файлу `boot.ipxe`. Функція `configure_boot_ipxe()` працює з файлом, який має динамічно додавати нові пункти меню в залежності від переданого параметру `iso_name`, тоді як `configure_boot_ipxe_ubuntu()` встановлює фіксований вміст для PXE-завантаження Ubuntu 22.04 LTS.

Після завантаження всіх необхідних файлів необхідно змінити конфігурацію мережі на сервері. Для цього створена функція `configure_netplan`.

Основні дії, які вона виконує, перераховані нижче:

1. Визначення шляху до файлу конфігурації Netplan `/etc/netplan/00-installer-config.yaml`.
2. Виклик команди `sudo nano` для відкриття файлу конфігурації Netplan за допомогою текстового редактора `nano`. В цьому файлі можна внести зміни до налаштувань мережі.
3. Визначення вмісту файлу конфігурації \, який містить налаштування мережі, такі як IP-адреса, маршрутизація та DNS-сервери (рис. 3.9).
4. Відкриття файлу конфігурації Netplan за допомогою методу `open` з режимом запису ('w').
5. Запис вмісту конфігурації Netplan у відкритий файл за допомогою методу `write`.
6. Застосування змін Netplan за допомогою команди `sudo netplan apply`, що виконує нову конфігурацію мережі.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

```

netplan_content = '''
network:
  version: 2
  ethernets:
    enp0s3:
      dhcp4: no
      addresses:
        - 195.191.14.4/24
      routes:
        - to: default
          via: 195.191.14.1
      nameservers:
        addresses:
          - 1.1.1.1
          - 8.8.8.8
...

```

Рисунок 3.9 – Конфігурація файлу /etc/netplan/00-installer-config.yaml

3.2 Автоматизація створення Apache-сервера

Веб-сервер Apache, створений Робертом МакКулом, є відкритим сервером з відкритим вихідним кодом, який виконує функцію обробки HTTP-запитів від користувачів і надання необхідної інформації у вигляді веб-сторінок і файлів. Цей сервер широко використовується програмістами для тестування коду та розробки веб-додатків.

Основна функція Apache полягає в обробці запитів від клієнтів і наданні їм веб-сторінок, які містять необхідну інформацію. Він може розпізнавати і обробляти різні типи запитів, такі як GET, POST, PUT і DELETE, і відповідно передавати результат користувачеві.

Окрім цього, Apache пропонує функцію безпечного обміну файлами, що дозволяє користувачам зберігати свої важливі файли в кореневому каталозі сервера і ділитися ними з іншими користувачами. Це забезпечує зручний і безпечний спосіб обміну даними та спільної роботи над файлами в мережевому середовищі.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

Apache здатний обслуговувати велику кількість запитів одночасно, що робить його популярним веб-сервером для великих проектів і високонавантажених веб-додатків. Його широкі можливості налаштування і розширення дозволяють програмістам адаптувати його під свої потреби і створювати потужні веб-додатки з високою надійністю та швидкодією.

Першою запускається функція `install_apache()` (рис. 3.10), яка перевіряє наявність Apache за допомогою `check_apache_installed()`. Якщо Apache не встановлено, вона виконує команду `sudo apt-get install apache2` для установки Apache. Після успішної установки вона викликає функцію `configure_apache()` для конфігурування Apache.

```
def install_apache():
    if not check_apache_installed():
        command = "sudo apt-get install apache2"
        subprocess.call(command, shell=True)

    if check_apache_installed():
        print("Apache успішно встановлено.")
        configure_apache()
    else:
        print("Виникла помилка під час установки Apache.")
```

Рисунок 3.10 – Функція `install_apache`

Функція `configure_apache()` конфігурує Apache, дозволяючи доступ до нього через брандмауер і оновлюючи конфігураційний файл Apache (`/etc/apache2/apache2.conf`). Вона здійснює заміну вмісту конфігураційного файлу за допомогою регулярного виразу і записує оновлений вміст назад у файл.

На веб-сервері Apache можна використовувати віртуальні хости для ізоляції налаштувань та розміщення декількох доменів на одному сервері. У випадку з Apache на Ubuntu 22.04.2 за замовчуванням доступний один включений віртуальний хост, який обслуговує каталог `/var/www/html`. Цього достатньо для роботи одного сайту, але якщо необхідно розмістити декілька сайтів, потрібно створити віртуальні хости. Для цього була створена функція `create_virtual_host`.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

```

command = "sudo ufw allow 'Apache'"
subprocess.call(command, shell=True)
config_file = "/etc/apache2/apache2.conf"

with open(config_file, "r") as file:
    content = file.read()

pattern =
r"<Directory\s+/var/www/>\n\s+AllowOverride\s+None\n\s+Require\s+all\s+granted"
    replacement = "<Directory /var/www/>\n    AllowOverride All\n    Require all
granted"
    updated_content = re.sub(pattern, replacement, content, flags=re.MULTILINE)

with open(config_file, "w") as file:
    file.write(updated_content)

```

Рисунок 3.11 – Вміст функції `configure_apache`

Спочатку створюється структура каталогів у каталозі `/var/www` для сайту `example.com`, `/var/www/html` залишається як каталог за замовчуванням, який буде обслуговуватись, якщо запит клієнта не відповідає іншим сайтам (рис. 3.12). Також вказуються права на каталог.

```

command = f"sudo mkdir -p /var/www/{domain_name}/html"
subprocess.call(command, shell=True)
command = f"sudo chown -R $USER:$USER /var/www/{domain_name}/html"
subprocess.call(command, shell=True)
command = f"sudo chmod -R 755 /var/www/{domain_name}"
subprocess.call(command, shell=True)

```

Рисунок 3.12 – Створення структури каталогів та надання прав

Для створюваного домену створюється окрема стартова сторінка. Для цього була прописана окрема функція `create_sample_page`, яка приймає на вхід доменне ім'я.

Згенеровано новий файл віртуального хосту з необхідними налаштуваннями для обслуговування контенту Apache (рис. 3.13). У вказаному файлі

/etc/apache2/sites-available/{domain_name}.conf були внесені зміни, щоб відповідати потребам сайту.

```
virtual_host_config = f"""
    <VirtualHost *:80>

        ServerAdmin admin@{domain_name}

        ServerName {domain_name}

        ServerAlias {domain_name}

        DocumentRoot /var/www/{domain_name}/html

        ErrorLog ${APACHE_LOG_DIR}/error.log

        CustomLog ${APACHE_LOG_DIR}/access.log combined

    </VirtualHost>

    """
```

Рисунок 3.13 – Файл віртуального хосту з необхідними налаштуваннями

Цей новий файл віртуального хосту містить конфігурацію, що визначає поведінку сервера Apache. Налаштування включають поштову адресу адміністратора сайту, основний домен (ServerName) та додаткові імена (ServerAlias), які пов'язані з цим віртуальним хостом. Крім того, було вказано шлях до кореневого каталогу сайту (DocumentRoot) і розміщення файлів журналів помилок та доступу.

Далі потрібно активувати цей віртуальний хост за допомогою команди `sudo a2ensite your_domain.conf`, щоб Apache почав обслуговувати контент згідно з цим файлом конфігурації.

Також можна відключити віртуальний хост за замовчуванням, який був визначений в файлі `000-default.conf`, за допомогою команди `sudo a2dissite 000-default.conf`. Це дозволить новому віртуальному хосту вступити в дію (рис. 3.14).

Насамкінець, необхідно перезапустити службу Apache, щоб зміни набрали чинності.

Функції `enable_virtual_host(virtual_host)` і `disable_virtual_host(virtual_host)` включають або вимикають віртуальний хост, а потім перезапускають Apache з використанням `restart_apache()`.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

```

command = f"sudo a2ensite {domain_name}.conf"
subprocess.call(command, shell=True)
command = "sudo a2dissite 000-default.conf"
subprocess.call(command, shell=True)
command = "sudo service apache2 restart"
subprocess.call(command, shell=True)

```

Рисунок 3.14 – Активація віртуального хосту та перезапуск служби

Функція `edit_servername_config(domain_name)` (рис. 3.15) редагує файл конфігурації серверного імені Apache (`/etc/apache2/conf-available/servername.conf`), додаючи рядок з заданим доменним іменем. Після редагування вона активує конфігурацію за допомогою `enable_servername_config()`.

```

def edit_servername_config(domain_name):
    config_file = "/etc/apache2/conf-available/servername.conf"
    line_to_insert = "ServerName {domain_name}\n".format(domain_name=domain_name)

    try:
        with open(config_file, "w") as file:
            file.writelines(line_to_insert)

        print("Вміст файлу {config_file} успішно онов-
лено.".format(config_file=config_file))
        enable_servername_config()
    except IOError:
        print("Помилка: Не вдалося оновити вміст файлу.")

```

Рисунок 3.15 – Функція `edit_servername_config`

Функції `stop_apache()`, `start_apache()` і `restart_apache()` відповідають за зупинку, запуск та перезапуск Apache відповідно. Вони виконують відповідні команди системного управління службами.

3.3 Автоматизація управління користувачами

В модулі автоматизації управління були реалізовані функції для керування користувачами на віддаленому сервері.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				56
Змн.	Арк.	№ докум.	Підпис	Дата		

Для забезпечення можливості встановлення SSH-з'єднання з віддаленим сервером та виконання команд на цьому сервері була використана бібліотека paramiko. Вона надає зручний інтерфейс для роботи з протоколом SSH.

Створена функція create_user() створює нового користувача на віддаленому сервері. Вона використовує бібліотеку Paramiko для підключення до сервера за допомогою SSH (рис. 3.16) і виконує команди на сервері для створення нового користувача (рис. 3.17). Функція запитує ім'я хоста, нове ім'я користувача, новий пароль користувача та чи цей користувач є адміністратором. Якщо користувач вказав, що цей користувач є адміністратором, то до команди створення додається параметр -G sudo.

```
client = paramiko.SSHClient()
client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

try:
    client.connect(hostname, username=username, password=password)
    print("Connected to the server.")
```

Рисунок 3.16 – Підключення за допомогою SSH

```
command = f'sudo -S useradd {new_username}'
if is_admin:
    command += ' -G sudo'
stdin, stdout, stderr = client.exec_command(command)
stdin.write(f'{password}\n')
stdin.flush()
```

Рисунок 3.17 – Створення користувача

Якщо користувач успішно створений, викликається функція create_pass() для встановлення пароля для нового користувача (рис. 3.18). Функція отримує новий пароль, ім'я користувача та з'єднання SSH (client), яке вже було встановлено.

```

command = f'sudo -S passwd {new_username}'
stdin, stdout, stderr = client.exec_command(command)
stdin.write(f'{password}\n')
stdin.write(f'{new_password}\n')
stdin.write(f'{new_password}\n')
stdin.flush()

```

Рисунок 3.18 – Додавання паролю до нового користувача

Функція `all_user` виводить список всіх користувачів на віддаленому сервері (рис. 3.19). Вона використовує бібліотеку `Paramiko` для підключення до сервера за допомогою SSH та виконує команду на сервері для перегляду списку користувачів. Функція запитує ім'я хоста сервера.

```

hostname = input("Введіть ім'я хоста: ")
client = paramiko.SSHClient()
client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
try:
    client.connect(hostname, username=username, password=password)
    print("Connected to the server.")
    stdin, stdout, stderr = client.exec_command('cat /etc/passwd | cut -d: -f1')
    users = stdout.read().decode().split('\n')

    if users[-1] == '':
        users.pop()

    print("Users on the server:")
    for user in users:
        print(user)

```

Рисунок 3.19 – Тіло функції

Функція `delete_user()` виконує видалення користувача на віддаленому сервері з використанням SSH-з'єднання (рис. 3.20).

Основні кроки функції `delete_user()` включають:

1. Запит ім'я хоста та ім'я користувача, якого потрібно видалити.

2. Встановлення SSH-клієнта за допомогою бібліотеки paramiko. Це включає налаштування політики автоматичного додавання ключа хоста.
3. Встановлення SSH-з'єднання з вказаним хостом, використовуючи надані ім'я користувача та пароль.
4. Створення команди для видалення користувача на віддаленому сервері. Команда використовує параметр userdel з вказаним ім'ям користувача для видалення.
5. Виконання команди видалення користувача за допомогою методу exec_command() SSH-клієнта. Пароль користувача також передається для підтвердження привілейних дій.
6. Отримання виводу та повідомлень про помилки в результаті виконання команди. Якщо вивід команди має код виходу 0, це означає успішне видалення користувача. У протилежному випадку, виводиться повідомлення про помилку.
7. Незалежно від результату виконання, SSH-з'єднання закривається в розділі finally.

```
hostname = input("Введіть ім'я хоста: ")
user_to_delete = input("Введіть ім'я користувача для видалення: ")
client = paramiko.SSHClient()
client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
try:
    client.connect(hostname, username=username, password=password)
    print("Connected to the server.")

    command = f'sudo -S userdel {user_to_delete}'
    stdin, stdout, stderr = client.exec_command(command)
    stdin.write(f'{password}\n')
    stdin.flush()
    error_message = stderr.read().decode('utf-8')
```

Рисунок 3.20 – Тіло функції видалення користувача

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				59
Змн.	Арк.	№ докум.	Підпис	Дата		

3.4 Створення текстового інтерфейсу

Для взаємодії з різними функціями та модулями, пов'язаними зі сервером, користувачами та сервером Apache, був створений окремий текстовий інтерфейс.

Основний виконавчий код розміщений в функціях `main_menu()` та `apache_menu()`. Головне меню `main_menu()` надає користувачу список опцій для вибору, які відповідають різним функціональностям. Залежно від вибору користувача виконується відповідна функція.

Меню Apache (`apache_menu()`) надає користувачу доступ до функцій пов'язаних з установкою та управлінням сервером Apache, включаючи встановлення Apache, додавання віртуальних хостів, запуск/зупинку/перезапуск Apache та ввімкнення/вимкнення віртуальних хостів.

Щоб викликати функції `main` і `configure_ipxe_pxe_boot()` з файлу `/server.module/pxe.py` у файлі `menu.py`, потрібно імпортувати цей файл та відповідні функції. Аналогічно для інших функцій.

Блок-схема управління функціями через меню наведена на в додатку Е.

Лістинг коду наведено в додатку D. Код надає зручний спосіб використання різних функцій та модулів.

3.5 Тестування

Для початку перевіримо роботу PXE-сервера. Всі необхідні структури каталогів та скомпільовані мікропрограми iPXE зображено на рисунку 3.20.

```
mariana@server:~$ tree /pxeboot
/pxeboot
├── config
│   └── boot.ipxe
├── firmware
│   ├── ipxe.efi
│   ├── ipxe.pxe
│   ├── undionly.kkpxe
│   ├── undionly.kkpxe
│   └── undionly.kpxe
└── os-images
```

Рисунок 3.20 – Структура каталогів

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

Створений каталог із завантаженим вмістом для розвертання сервера зображено на рисунках 3.21 – 3.22.

```
mariana@server:~$ cd ~/Downloads
mariana@server:~/Downloads$ ls -lh
total 4.0K
drwxr-xr-x 6 root root 4.0K Jun 19 18:41 ipxe
```

Рисунок 3.21 – Створений каталог

```
mariana@server:~/Downloads/ipxe/src$ ls -lh
total 268K
drwxr-xr-x 9 root root 4.0K Jun 19 18:41 arch
drwxr-xr-x 3 root root 36K Jun 19 20:35 bin
drwxr-xr-x 3 root root 36K Jun 19 20:35 bin-x86_64-efi
-rw-r--r-- 1 root root 57 Jun 19 20:41 bootconfig.ipxe
drwxr-xr-x 8 root root 4.0K Jun 19 18:41 config
drwxr-xr-x 2 root root 4.0K Jun 19 18:41 core
drwxr-xr-x 3 root root 4.0K Jun 19 18:41 crypto
drwxr-xr-x 2 root root 4.0K Jun 19 18:41 doc
-rw-r--r-- 1 root root 62K Jun 19 18:41 doxygen.cfg
drwxr-xr-x 10 root root 4.0K Jun 19 18:41 drivers
drwxr-xr-x 6 root root 4.0K Jun 19 18:41 hci
drwxr-xr-x 2 root root 4.0K Jun 19 18:41 image
drwxr-xr-x 9 root root 4.0K Jun 19 18:41 include
drwxr-xr-x 8 root root 4.0K Jun 19 18:41 interface
drwxr-xr-x 2 root root 4.0K Jun 19 18:41 libgcc
-rw-r--r-- 1 root root 7.7K Jun 19 18:41 Makefile
-rw-r--r-- 1 root root 1.5K Jun 19 18:41 Makefile.efi
-rw-r--r-- 1 root root 44K Jun 19 18:41 Makefile.housekeeping
-rw-r--r-- 1 root root 1.3K Jun 19 18:41 Makefile.linux
drwxr-xr-x 7 root root 4.0K Jun 19 18:41 net
drwxr-xr-x 3 root root 4.0K Jun 19 19:52 pxeboot
drwxr-xr-x 2 root root 4.0K Jun 19 18:41 scripts
drwxr-xr-x 2 root root 4.0K Jun 19 18:41 tests
drwxr-xr-x 2 root root 4.0K Jun 19 18:41 usr
drwxr-xr-x 3 root root 4.0K Jun 19 18:45 util
```

Рисунок 3.22 – Вміст каталогу

Перевіримо, чи працює служба dnsmasq, за допомогою якої налаштовані сервери DHCP та TFTP (рис. 3.23).

Перевіримо роботу функції додавання та завантаження нової ОС до конфігурації PXE-сервера. Для цього додамо образ Fedora Workstation 36. Завантажені файли мають зберігатися в каталозі /pxeboot/os-images/Fedora-Workstation-Live-x86_64-1.5 (рис. 3.24).

```

mariana@server:~$ sudo systemctl status dnsmasq
[sudo] password for mariana:
● dnsmasq.service - dnsmasq - A lightweight DHCP and caching DNS server
   Loaded: loaded (/lib/systemd/system/dnsmasq.service; enabled; vendor pres>
   Active: active (running) since Tue 2023-06-20 14:15:32 UTC; 34min ago
     Process: 800 ExecStartPre=/etc/init.d/dnsmasq checkconfig (code=exited, st>
     Process: 826 ExecStart=/etc/init.d/dnsmasq systemd-exec (code=exited, stat>
     Process: 841 ExecStartPost=/etc/init.d/dnsmasq systemd-start-resolvconf (c>
   Main PID: 840 (dnsmasq)
     Tasks: 1 (limit: 2213)
    Memory: 2.8M
       CPU: 95ms
    CGroup: /system.slice/dnsmasq.service
            └─840 /usr/sbin/dnsmasq -x /run/dnsmasq/dnsmasq.pid -u dnsmasq -7>

```

Рисунок 3.23 – Перевірка стану служби

```

os-images
├── Fedora-Workstation-Live-x86_64-36-1.5
│   ├── EFI
│   │   └── BOOT
│   │       ├── BOOT.conf
│   │       ├── BOOTIA32.EFI
│   │       ├── BOOTX64.EFI
│   │       ├── fonts
│   │       │   └── unicode.pf2
│   │       ├── grub.cfg
│   │       ├── grubia32.efi
│   │       ├── grubx64.efi
│   │       ├── mmia32.efi
│   │       └── mmx64.efi
│   ├── Fedora-Legal-README.txt
│   ├── images
│   │   ├── efiboot.img
│   │   ├── macboot.img
│   │   └── pxeboot
│   │       ├── initrd.img
│   │       └── vmlinuz
│   └── isolinux

```

Рисунок 3.24 – Вміст каталогу (скриншот обрізано)

Перевіримо сконфігурований файл із завантажувальними кодами. В ньому має бути додано новий пункт меню та конфігурація для завантаження нової ОС (рис. 3.25).

Додамо нову віртуальну машину та спробуємо завантажити її як PXE-клієнт. При запуску має автоматично визначатись динамічно IP-адреса та відбуватись підключення до TFTP-сервера (рис. 3.26).

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

```

GNU nano 6.2 /pxeboot/config/boot.ipxe
#!ipxe

set server_ip 195.191.14.4
set root_path /pxeboot

menu Select an OS to boot
item ubuntu-22.04.2-desktop-amd64 Install Ubuntu Desktop 22.04.2>
item Fedora-Workstation-Live-x86_64-36-1.5 Install Fedora Workstation 36
choose --default exit --timeout 10000 option && goto ${option}

:ubuntu-22.04.2-desktop-amd64
set os_root os-images/ubuntu-22.04.2-desktop-amd64
kernel tftp://${server_ip}/${os_root}/casper/vmlinuz
initrd tftp://${server_ip}/${os_root}/casper/initrd
imgargs vmlinuz initrd=initrd boot=casper maybe-ubiquity netboot=nfs ip=dhcp n>
boot

:Fedora-Workstation-Live-x86_64-36-1.5
set os_root os-images/Fedora-Workstation-Live-x86_64-36-1.5
kernel tftp://${server_ip}/${os_root}/images/pxeboot/vmlinuz
initrd tftp://${server_ip}/${os_root}/images/pxeboot/initrd.img
imgargs vmlinuz initrd=initrd.img ip=dhcp rd.live.image root=live:nfs://${serv>
boot

```

Рисунок 3.25 – Додана конфігурація

```

test_serv [Запущено] - Oracle VM VirtualBox
Файл  Машина  Перегляд  Введення  Пристрої  Довідка

iPXE 1.21.1 -- Open Source Network Boot Firmware -- http://ipxe.org
Features: DNS TFTP PXE PXEXT

net0: 08:00:27:04:fd:ea using 82540em on 0000:00:03.0 (open)
  [Link:down, TX:0 TXE:0 RX:0 RXE:0]
  [Link status: Down (http://ipxe.org/38086101)]
Waiting for link-up on net0..... ok
Configuring (net0 08:00:27:04:fd:ea)..... ok
net0: 195.191.14.166/255.255.255.0 gw 195.191.14.1
Next server: 195.191.14.4
Filename: firmware/ipxe.pxe
tftp://195.191.14.4/firmware/ipxe.pxe... ok
ipxe.pxe : 340722 bytes [PXE-NBP]
PXE->EB: ?PXE at 9C48:0580, entry point at 9C48:04C8
        UNDI code segment 9C48:0801, data segment 9CCC:2B30 (625-638kB)
        UNDI device is PCI 00:03.0, type DIX+802.3
        638kB free base memory after PXE unload
iPXE initialising devices...ok

iPXE 1.21.1+ (g4fa40) -- Open Source Network Boot Firmware -- https://ipxe.org
Features: DNS HTTP iSCSI TFTP SRP AoE ELF MBOOT PXE bzImage Menu PXEXT
Waiting for link-up on net0.....

```

Рисунок 3.26 – Підключення до DHCP- та TFTP-сервера

Після успішної ініціалізації мікропрограми iPXE має відкритись меню із доступними до встановлення ОС (рис. 3.27). Обираємо другий пункт.

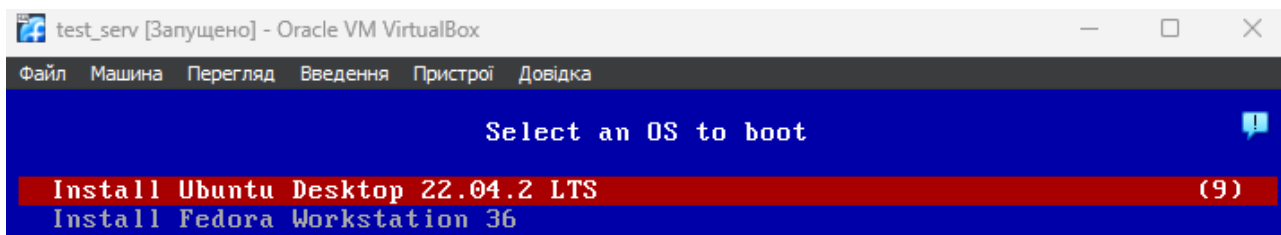


Рисунок 3.27 – Меню завантаження

Запускається завантаження iPXE файлів vmlinuz та initrd із сервера завантаження PXE (рис. 3.28).

```
tftp://195.191.14.4/os-images/Fedora-Workstation-Live-x86_64-36-1.5/images/pxeboot/vmlinuz... ok
tftp://195.191.14.4/os-images/Fedora-Workstation-Live-x86_64-36-1.5/images/pxeboot/initrd.img... 56%
```

Рисунок 3.28 – Завантаження файлів для Fedora Workstation 36

Після завантаження запускається конфігурація (рис. 3.29) та відбувається встановлення ОС (рис. 3.30 – 3.31).

```
test_serv [Запущено] - Oracle VM VirtualBox
Файл  Машина  Перегляд  Введення  Пристрої  Довідка
ON +UTMP +SYSVINIT default-hierarchy=unified)
[ 23.552311] systemd[1]: Detected virtualization oracle.
[ 23.554697] systemd[1]: Detected architecture x86-64.

Welcome to Fedora Linux 36 (Workstation Edition)!

[ 23.675067] systemd[1]: Initializing machine ID from random
[ 23.788412] systemd[1]: 1234-5678-9012-3456 (1687287252-823178) + more
```

Рисунок 3.29 – Запуск установки



Рисунок 3.30 – Вікно встановлення ОС Fedora

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

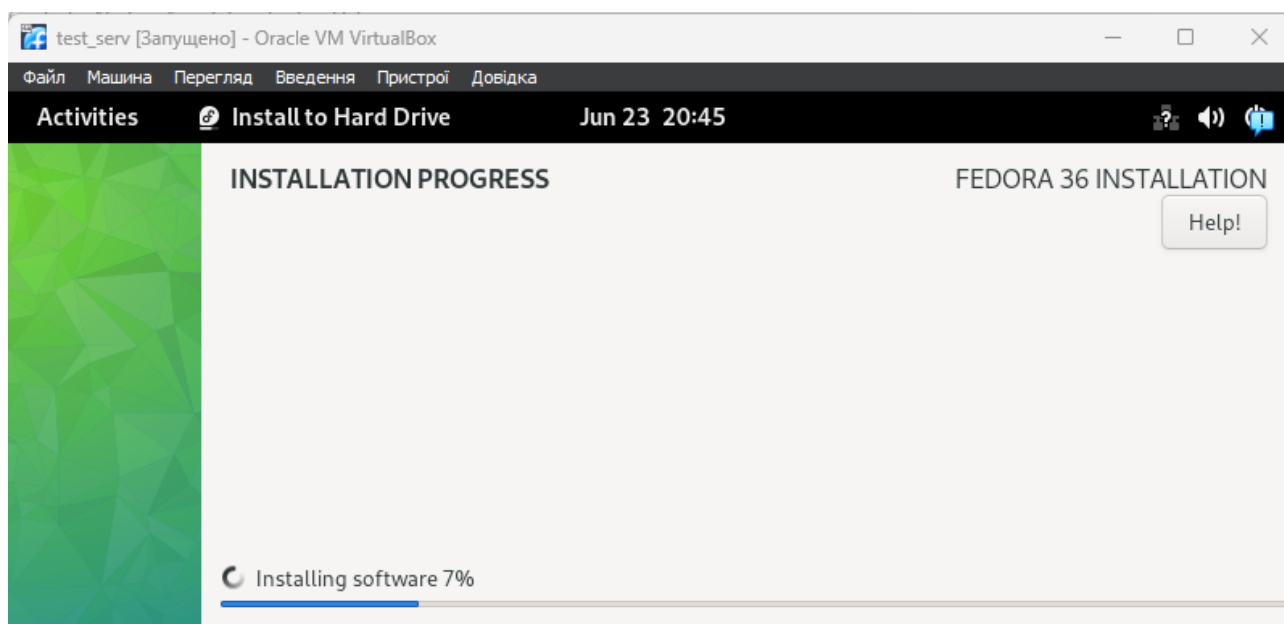


Рисунок 3.31 – Процес встановлення ОС Fedora

Перевіримо мережеві параметри нової машини:

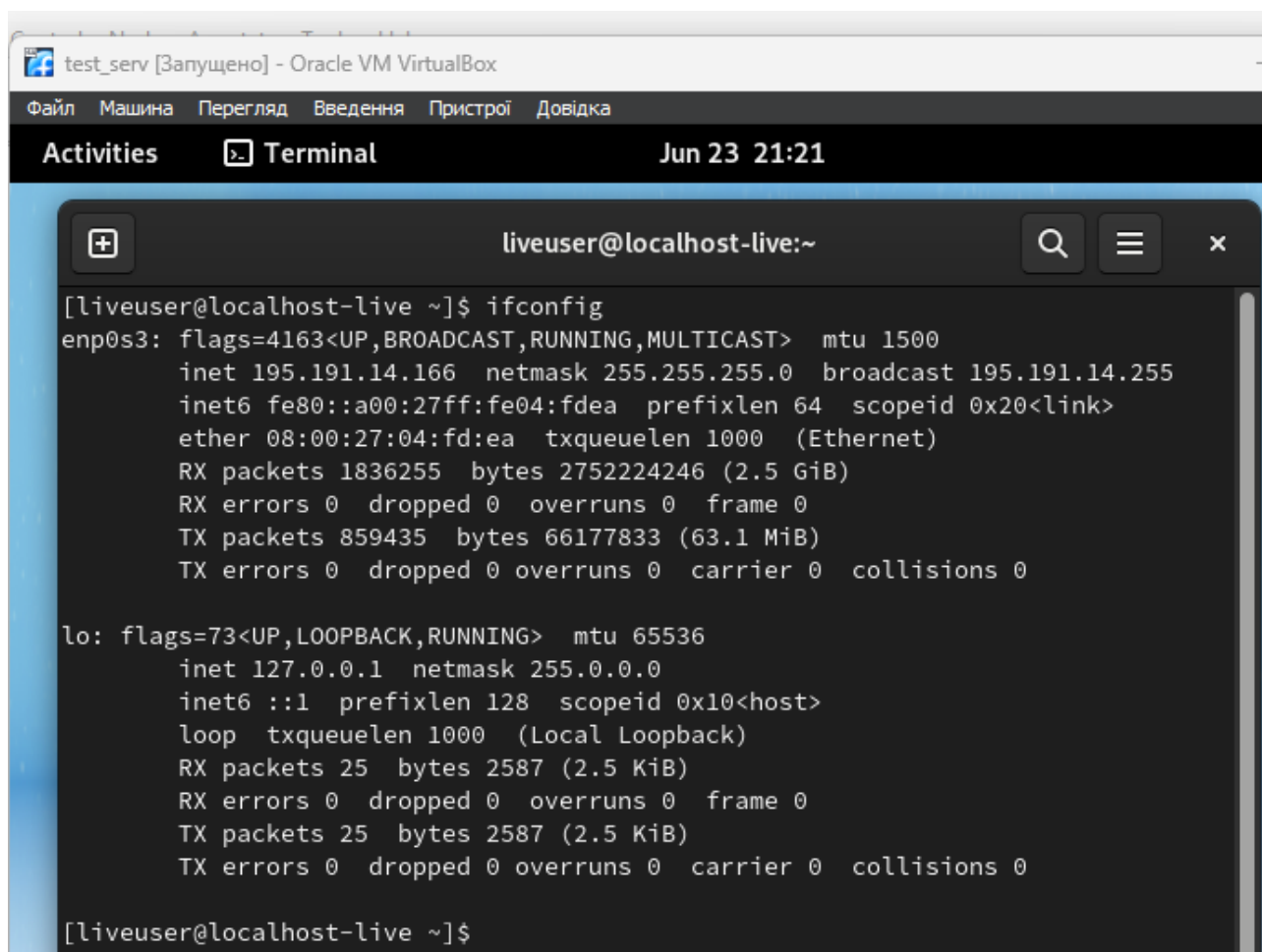


Рисунок 3.32 – Мережеві параметри

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

Далі перевіримо роботу встановлення сервера Apache. При першому запуску відбувається перевірка. Якщо служби не встановлено, виводиться відповідне повідомлення та відбувається встановлення (рис. 3.33).

```
mariana@server:~$ sudo python3 apache.py
Apache не встановлено на системі.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

Рисунок 3.33 – Встановлення необхідних компонентів

В разі якщо служби встановлено, буде виведено відповідне повідомлення:

```
Меню Apache:
1. Встановлення Apache
2. Додавання віртуального хосту Apache
3. Запустити Apache
4. Зупинити Apache
5. Перезапустити Apache
6. Ввімкнути віртуальний хост
7. Вимкнути віртуальний хост
Виберіть опцію: 1
Server version: Apache/2.4.52 (Ubuntu)
Server built: 2023-03-01T22:43:55

Apache вже встановлено.
```

Рисунок 3.34 – Apache встановлено

Перевіримо спрацювання функцій над управлінням сервером (рис. 3.34 – 3.36):

```
Меню Apache:
1. Встановлення Apache
2. Додавання віртуального хосту Apache
3. Запустити Apache
4. Зупинити Apache
5. Перезапустити Apache
6. Ввімкнути віртуальний хост
7. Вимкнути віртуальний хост
Виберіть опцію: 2
Введіть ім'я домену: info.net
Sample page created at /var/www/info.net/html/index.html
Enabling site info.net.
To activate the new configuration, you need to run:
    systemctl reload apache2
Site 000-default already disabled
Створено віртуальний хост для домену info.net.
Вміст файлу /etc/apache2/conf-available/servername.conf успішно оновлено.
Conf servername already enabled
Syntax OK
```

Рисунок 3.34 – Встановлення нового домена

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				66
Змн.	Арк.	№ докум.	Підпис	Дата		



Рисунок 3.35 – Відкриття сторінки в браузері



Рисунок 3.36 – Перевірка запису в конфігураційному файлі

Перевіримо як відбувається підключення по SSH та управління користувачами. Підключимось до робочої станції та створимо нового користувача (рис. 3.37).

```
Введіть ім'я хоста: 195.191.14.5
Введіть нове ім'я користувача: newUser
Введіть новий пароль користувача: 123456
Чи є цей користувач адміністратором? (y/n): y
Connected to the server.
Користувач 'newUser' був успішно створений.
Пароль для користувача 'newUser' був встановлений.
```

Рисунок 3.37 – Створено нового користувача

Перевіримо створення та відразу виведемо всіх наявних користувачів в системі (рис. 3.38).

```
nm-openvpn
nmap
nmap
whoopsie
colord
geoclue
pulse
gnome-initial-setup
gdm
student
systemd-coredump
vboxadd
sshd
adminTest
adminNew
adminTest10
newUser
```

Рисунок 3.38 – Створений користувач є в системі

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

Спробуємо видалити тільки що створеного користувача (рис. 3.39) та перевіriamo пропроблену операцію (рис. 3.40).

```
Введіть ім'я хоста: 195.191.14.5
Введіть ім'я користувача для видалення: newUser
Connected to the server.
Користувач 'newUser' був успішно видалений.
```

Рисунок 3.39 – Видалення користувача

```
gnome-initial-setup
gdm
student
systemd-coredump
vboxadd
sshd
adminTest
adminNew
adminTest10
```

Рисунок 3.40 – Користувача видалено

Висновки до третього розділу

В даному розділі була реалізована підсистема адміністрування ОС Linux.

Проведено налаштування мережі. Це включало налаштування IP-адреси, маршрутизації та серверів DNS для забезпечення належної мережевої комунікації.

Здійснена автоматизація створення PXE-сервера та завантажувальних кодів. Були написані скрипти на мові Python, що дозволяють автоматично встановлювати та налаштовувати сервер DHCP, TFTP та NFS. Також було здійснено автоматичне завантаження образів операційних систем і налаштування файлу boot.ipxe для вибору операційної системи під час завантаження.

Налаштовано автоматизацію Apache-сервера та автоматичного додавання нового віртуального домену. Реалізовано систему керування Apache-сервером та доменами.

Налаштовано автоматизацію керування користувачами на віддаленій ОС.

Виконано тестування реалізованої підсистеми адміністрування.

		Матвієнко М.В.			KIK.KP.B – 123 – 12 – ПЗ	Арк.
		Головня О.С.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

В кваліфікаційній роботі було успішно досягнуто завдань, поставлених на початку дослідження і розробки. Загальні висновки за кожним розділом роботи підтверджують успішність реалізації автоматизованих процесів управління та адміністрування ОС Linux.

У першому розділі було проведено аналіз підсистем адміністрування ОС Linux Puppet та Systemd та визначено задачі, які підлягають автоматизації. Також було обрано мову програмування Python як основну для реалізації проекту. Виявлено переваги Python для автоматизації завдань адміністрування, такі як простота використання, розширюваність, підтримка ООП та багатофункціональність стандартної бібліотеки.

Другий розділ був присвячений проектуванню підсистеми автоматизації адміністрування ОС Linux. Було розроблено план автоматизації, який описує послідовність дій та процедури, необхідні для виконання автоматизованих наступних завдань:

- Автоматизоване розгортання PXE-сервера та додавання різних конфігурацій ОС на віддалених серверах;
- автоматизація розгортання Apache-сервера;
- автоматична зміна конфігураційних файлів при додаванні нових віртуальних доменів;
- автоматизоване управління Apache-сервером та віртуальними доменами;
- автоматизоване підключення по SSH до віддалених серверів та здійснення управління користувачами.

Встановлені вимоги до підсистеми автоматизації, включаючи функціональні та нефункціональні вимоги. Спроековано майбутню мережу. Зазначений розділ підтверджує успішне проектування та визначення необхідних етапів для реалізації автоматизованої системи адміністрування ОС Linux.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

Третій розділ був присвячений реалізації підсистеми адміністрування ОС Linux. Було налаштовано мережу, здійснено автоматизацію створення PXE-сервера та завантажувальних кодів для різних ОС, налаштовано автоматизацію Apache-сервера та керування користувачами на віддаленій ОС. Проведене тестування підтверджує правильну роботу реалізованої підсистеми адміністрування.

Загалом, реалізація кваліфікаційної роботи демонструє успішність досягнення завдань, визначених на початку дослідження. Автоматизація адміністрування ОС Linux була успішно реалізована з використанням мови програмування Python та розроблених скриптів. Реалізована підсистема дозволяє здійснювати автоматизовані процеси управління серверами, налаштування мережі, керування користувачами та інші операції адміністрування, що спрощує роботу та підвищує ефективність управління ОС Linux.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Blank-Edelman D. N. Automating System Administration with Perl / David Blank-Edelman., 2009. – 670 с. – (Second edition).
2. Ben-Hamou A. Practical Ruby for System Administration / Andre Ben-Hamou., 2007. – 261 с. – (1st ed.).
3. Install PowerShell on Linux [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/powershell/scripting/install/installing-powershell-on-linux?view=powershell-7.3>.
4. Gift N. Python for Unix and Linux System Administration / N. Gift, J. M. Jones., 2008. – 458 с. – (1st ed.).
5. Python for DevOps: Learn Ruthlessly Effective Automation / N. Gift, B. Kennedy, D. Alfredo, G. Grig., 2020. – 506 с. – (1st).
6. Кендалл С. Об'єктно-орієнтоване програмування [Електронний ресурс] / Саймон Кендалл. – 2011. – Режим доступу до ресурсу: <http://aems.edu.sd/wp-content/uploads/2019/03/object-oriented-programming-using-c-sharp.pdf>.
7. Сучасні інформаційні технології [Електронний ресурс]. — Режим доступу: <http://it-tehnolog.com/statti/suchasni-informatsiyni-tehnologiyi>
8. How to Install Fedora 36 Workstation Step by Step [Електронний ресурс] – Режим доступу до ресурсу: <https://www.linuxtechi.com/how-to-install-fedora-workstation/>.
9. Apache The Definitive Guide [Електронний ресурс] // O'Reilly Media, Incorporated. – 2003. – Режим доступу до ресурсу: <https://github.com/anzhihe/Free-Web-Books/blob/master/book/O'Reilly%20-%20Apache%20The%20Definitive%20Guide%203rd%20Edition.pdf>.

10. Booting Theseus on Real Hardware via PXE [Електронний ресурс] – Режим доступу до ресурсу: <https://www.theseus-os.com/Theseus/book/running/pxe.html>.
11. Acronis Storage 2.4 Installation Using PXE [Електронний ресурс] // Acronis. – 2019. – Режим доступу до ресурсу: https://dl.acronis.com/u/storage2/pdf/AcronisStorage_2_installation_pxe_guide_en-US.pdf.
12. Preboot Execution Environment (PXE) Specification [Електронний ресурс] // Intel Corporation. – 1999. – Режим доступу до ресурсу: <http://www.pix.net/software/pxeboot/archive/pxespec.pdf>.
13. Linux User Administration: A Complete Guide to Managing Multi-User Systems [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: <https://www.linuxfordevices.com/tutorials/linux-user-administration>.
14. Learning Puppet 4 [Електронний ресурс] // O'Reilly. – 2016. – Режим доступу до ресурсу: <https://oiipdf.com/download/learning-puppet-4>.
15. Linux Service Management Made Easy with systemd: Advanced techniques to effectively manage, control, and monitor Linux systems and services [Електронний ресурс] // Packt Publishing. – 2022. – Режим доступу до ресурсу: https://www.packtpub.com/product/linux-service-management-made-easy-with-systemd/9781801811644?utm_source=github&utm_medium=repository&utm_campaign=9781801811644.

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				72
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import os
import subprocess
import configparser

def create_directory_structure():
    directories = [
        '/pxeboot/config',
        '/pxeboot/firmware',
        '/pxeboot/os-images'
    ]
    for directory in directories:
        if not os.path.exists(directory):
            os.makedirs(directory)

def compile_ipxe():
    subprocess.call(['sudo', 'apt', 'update'])
    subprocess.call(['sudo', 'apt', 'install', 'build-essential', 'liblzma-dev',
'isolinux', 'git', '-y'])

    os.chdir('/home/mariana/Downloads')
    subprocess.call(['git', 'clone', 'https://github.com/ipxe/ipxe.git'])
    os.chdir('/home/mariana/Downloads/ipxe/src')

    bootconfig_content = '''#!ipxe

dhcp

chain tftp://195.191.14.4/config/boot.ipxe
...

with open('bootconfig.ipxe', 'w') as f:
    f.write(bootconfig_content)

    subprocess.call(['make', 'bin/ipxe.pxe', 'bin/undionly.kpxe', 'bin/undionly.kkpxe',
'bin/undionly.kkkpxe', 'bin-x86_64-efi/ipxe.efi', 'MBED=bootconfig.ipxe'])

def copy_firmwares():
    os.chdir('/home/mariana/Downloads/ipxe/src')
    subprocess.call(['sudo', 'cp', '-v', 'bin/ipxe.pxe', 'bin/undionly.kpxe', 'bin/un-
dionly.kkpxe', 'bin/undionly.kkkpxe', 'bin-x86_64-efi/ipxe.efi', '/pxeboot/firmware/'])

```

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

def install_configure_dnsmasq():
    os.chdir('/home/mariana/')
    subprocess.call(['sudo', 'apt', 'install', 'dnsmasq', '-y'])
    subprocess.call(['sudo', 'mv', '-v', '/etc/dnsmasq.conf', '/etc/dnsmasq.conf.back-
up'])

    dnsmasq_content = '''interface=enp0s3
bind-interfaces
domain=server.local
dhcp-range=ens38,195.191.14.50,195.191.14.200,255.255.255.0,8h
dhcp-option=option:router,195.191.14.1
enable-tftp
tftp-root=/pxeboot
dhcp-match=set:bios-x86,option:client-arch,0
dhcp-boot=tag:bios-x86,firmware/ipxe.pxe
dhcp-match=set:efi-x86_64,option:client-arch,7
dhcp-match=set:efi-x86_64,option:client-arch,9
dhcp-boot=tag:efi-x86_64,firmware/ipxe.pxe
'''

    with open('/etc/dnsmasq.conf', 'w') as configfile:
        configfile.write(dnsmasq_content)

    subprocess.call(['sudo', 'systemctl', 'restart', 'dnsmasq'])

def install_configure_nfs_server():
    subprocess.call(['sudo', 'apt', 'install', 'nfs-kernel-server'])

    nfs_config = "/pxeboot\t*(ro,sync,no_wdelay,insecure_locks,no_root_squash,inse-
cure,no_subtree_check)"
    with open('/etc/exports', 'w') as f:
        f.write(nfs_config)

    subprocess.call(['sudo', 'exportfs', '-av'])

def open_file_dialog():
    root = tk.Tk()
    root.withdraw()
    file_path = filedialog.askopenfilename()
    return file_path

```

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

def configure_ipxe_pxe_boot_ubuntu():
    iso_url = input("Введіть URL для завантаження ISO-образу: ")
    iso_name_match = re.search(r'([^\s/]+)\.iso$', iso_url)
    if iso_name_match:
        iso_name = iso_name_match.group(1)
    else:
        print("Невдалось отримати назву ISO-образу з URL")
        return

    iso_path = os.path.expanduser(f"/home/mariana/Downloads/{iso_name}.iso")
    pxeboot_path = f"/pxeboot/os-images/{iso_name}"

    subprocess.call(["wget", iso_url, "-O", iso_path])
    subprocess.call(["sudo", "mount", "-o", "loop", iso_path, "/mnt"])
    subprocess.call(["sudo", "mkdir", "-pv", pxeboot_path])
    subprocess.call(["sudo", "rsync", "-avz", "/mnt/", pxeboot_path])
    subprocess.call(["sudo", "umount", "/mnt"])
    os.remove(iso_path)
    configure_boot_ipxe(iso_name)

def configure_ipxe_pxe_boot():
    iso_url = input("Введіть URL для завантаження ISO-образу: ")
    iso_name_match = re.search(r'([^\s/]+)\.iso$', iso_url)
    if iso_name_match:
        iso_name = iso_name_match.group(1)
    else:
        print("Невдалось отримати назву ISO-образу з URL")
        return

    iso_path = os.path.expanduser(f"/home/mariana/Downloads/{iso_name}.iso")
    pxeboot_path = f"/pxeboot/os-images/{iso_name}"

    subprocess.call(["wget", iso_url, "-O", iso_path])
    subprocess.call(["sudo", "mount", "-o", "loop", iso_path, "/mnt"])
    subprocess.call(["sudo", "mkdir", "-pv", pxeboot_path])
    subprocess.call(["sudo", "rsync", "-avz", "/mnt/", pxeboot_path])
    subprocess.call(["sudo", "umount", "/mnt"])
    os.remove(iso_path)
    configure_boot_ipxe(iso_name)

```

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

def configure_boot_ipxe(iso_name):
    boot_ipxe_path = '/pxeboot/config/boot.ipxe'

    # Зчитуємо вміст файлу boot.ipxe
    with open(boot_ipxe_path, 'r') as f:
        boot_ipxe_content = f.read()

    # Запитуємо назву для пункту меню від користувача
    name = input("Введіть назву для пункту меню: ")

    # Додаємо новий пункт меню за допомогою регулярних виразів
    pattern = r"menu Select an OS to boot\s+item\s+(\w+)\s+Install\s+(\w+)"
    replacement = rf"menu Select an OS to boot\nitem \1 Install \2\nitem {iso_name} In-
stall {name}"
    updated_content = re.sub(pattern, replacement, boot_ipxe_content)

    # Зчитуємо вміст файлу з кодом, що надає користувач
    code_file_path = open_file_dialog()
    with open(code_file_path, 'r') as code_file:
        code_content = code_file.read()

    # Дописуємо завантажуваний код в кінець файлу boot.ipxe
    updated_content += '\n' + code_content

    # Записуємо оновлений вміст у файл boot.ipxe
    with open(boot_ipxe_path, 'w') as f:
        f.write(updated_content)

def configure_boot_ipxe_ubuntu():
    boot_ipxe_path = '/pxeboot/config/boot.ipxe'
    boot_ipxe_content = ''#!ipxe

set server_ip 195.191.14.4
set root_path /pxeboot

menu Select an OS to boot
item ubuntu-22.04.2-desktop-amd64 Install Ubuntu Desktop 22.04.2 LTS
choose --default exit --timeout 10000 option && goto ${option}

:ubuntu-22.04.2-desktop-amd64
set os_root os-images/ubuntu-22.04.2-desktop-amd64

```

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

kernel tftp://${server_ip}/${os_root}/casper/vmlinuz
initrd tftp://${server_ip}/${os_root}/casper/initrd
imgargs vmlinuz initrd=initrd boot=casper maybe-ubiquity netboot=nfs ip=dhcp nfs-
root=${server_ip}:${root_path}/${os_root} quiet splash ---
boot
...

    with open(boot_ipxe_path, 'w') as f:
        f.write(boot_ipxe_content)

def configure_netplan():
    netplan_path = '/etc/netplan/00-installer-config.yaml'
    netplan_content = '''
network:
  version: 2
  ethernets:
    enp0s3:
      dhcp4: no
      addresses:
        - 195.191.14.4/24
      routes:
        - to: default
          via: 195.191.14.1
      nameservers:
        addresses:
          - 1.1.1.1
          - 8.8.8.8
    ...

    with open(netplan_path, 'w') as f:
        f.write(netplan_content)

    subprocess.call(['sudo', 'netplan', 'apply'])

def main():
    create_directory_structure()
    compile_ipxe()
    copy_firmwares()
    install_configure_dnsmasq()
    install_configure_nfs_server()
    configure_ipxe_pxe_boot_ubuntu()
    configure_boot_ipxe_ubuntu()

```

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

```
configure_netplan()

if __name__ == '__main__':
    main()
```

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import subprocess
import re

def check_apache_installed():
    # Перевірити, чи Apache вже встановлено
    check_command = "apache2 -v"
    try:
        output = subprocess.check_output(check_command, shell=True, stderr=subprocess.STDOUT)
        print(output.decode("utf-8"))
        print("Apache вже встановлено.")
        return True
    except subprocess.CalledProcessError as e:
        print("Apache не встановлено на системі.")
        return False

def install_apache():
    if not check_apache_installed():
        # Виконати команду для установки Apache
        command = "sudo apt-get install apache2"
        subprocess.call(command, shell=True)

        # Перевірити, чи Apache успішно встановлено
        if check_apache_installed():
            print("Apache успішно встановлено.")
            configure_apache()
        else:
            print("Виникла помилка під час установки Apache.")

def create_sample_page(domain_name):
    content = """
<!DOCTYPE html>
<html>
<head>
    <title>Welcome to {domain_name}</title>
</head>
<body>
    <h1>Welcome to {domain_name}</h1>
    <p>This is a sample page for your website.</p>
</body>

```

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

</html>

"".format(domain_name=domain_name)

file_path = "/var/www/{domain_name}/html/index.html".format(domain_name=do-
main_name)

try:
    with open(file_path, "w") as file:
        file.write(content)
        print("Sample page created at {file_path}".format(file_path=file_path))
except IOError:
    print("Error: Failed to create the sample page.")

def create_virtual_host(domain_name):
    # Створити директорію для веб-сайту
    command = f"sudo mkdir -p /var/www/{domain_name}/html"
    subprocess.call(command, shell=True)

    # Налаштувати власника та права доступу для директорії
    command = f"sudo chown -R $USER:$USER /var/www/{domain_name}/html"
    subprocess.call(command, shell=True)

    command = f"sudo chmod -R 755 /var/www/{domain_name}"
    subprocess.call(command, shell=True)

    create_sample_page(domain_name)

    # Створити файл конфігурації віртуального хосту
    virtual_host_config = f"""
<VirtualHost *:80>
    ServerAdmin admin@{domain_name}
    ServerName {domain_name}
    ServerAlias {domain_name}
    DocumentRoot /var/www/{domain_name}/html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
"""

    # Записати конфігурацію в файл
    with open(f"{domain_name}.conf", "w") as file:

```

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        file.write(virtual_host_config)

# Перемістити файл конфігурації в папку Apache
command = f"sudo mv {domain_name}.conf /etc/apache2/sites-available/"
subprocess.call(command, shell=True)

# Активувати файл конфігурації
command = f"sudo a2ensite {domain_name}.conf"
subprocess.call(command, shell=True)

# Вимкнути віртуальний файл конфігурації за замовчуванням
command = "sudo a2disssite 000-default.conf"
subprocess.call(command, shell=True)

# Перезапустити Apache
command = "sudo service apache2 restart"
subprocess.call(command, shell=True)

print(f"Створено віртуальний хост для домену {domain_name}.")
edit_servername_config(domain_name)

def configure_apache():
    # Дозволити доступ до Apache через брандмауер
    command = "sudo ufw allow 'Apache'"
    subprocess.call(command, shell=True)
    config_file = "/etc/apache2/apache2.conf"

    with open(config_file, "r") as file:
        content = file.read()

    pattern = r"<Directory\s+/var/www/>\n\s+AllowOverride\s+None\n\s+Re-
quire\s+all\s+granted"
    replacement = "<Directory /var/www/>\n    AllowOverride All\n    Require all
granted"
    updated_content = re.sub(pattern, replacement, content, flags=re.MULTILINE)

    with open(config_file, "w") as file:
        file.write(updated_content)

    print("Конфігураційний файл Apache успішно оновлено.")

```

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

```

def edit_servername_config(domain_name):
    config_file = "/etc/apache2/conf-available/servername.conf"
    line_to_insert = "ServerName {domain_name}\n".format(domain_name=domain_name)

    try:
        with open(config_file, "w") as file:
            file.writelines(line_to_insert)

            print("Вміст файлу {config_file} успішно онов-
лено.".format(config_file=config_file))
            enable_servername_config()
    except IOError:
        print("Помилка: Не вдалося оновити вміст файлу.")

def enable_servername_config():
    command = "sudo a2enconf servername"
    subprocess.call(command, shell=True)
    command = "sudo systemctl reload apache2"
    subprocess.call(command, shell=True)
    test_apache_config()

def test_apache_config():
    command = "sudo apache2ctl configtest"
    subprocess.call(command, shell=True)

def stop_apache():
    subprocess.run(['sudo', 'systemctl', 'stop', 'apache2'])

def start_apache():
    subprocess.run(['sudo', 'systemctl', 'start', 'apache2'])

def restart_apache():
    subprocess.run(['sudo', 'systemctl', 'restart', 'apache2'])

def enable_virtual_host(virtual_host):
    subprocess.run(['sudo', 'a2ensite', virtual_host])
    restart_apache()

def disable_virtual_host(virtual_host):
    subprocess.run(['sudo', 'a2dissite', virtual_host])
    restart_apache()

```

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				83
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import paramiko
import argparse

username = "student"
password = "111111"

def create_user():
    hostname = input("Введіть ім'я хоста: ")
    new_username = input("Введіть нове ім'я користувача: ")
    new_password = input("Введіть новий пароль користувача: ")
    is_admin_input = input("Чи є цей користувач адміністратором? (y/n): ")
    is_admin = is_admin_input.lower() == 'y'

    client = paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    try:
        client.connect(hostname, username=username, password=password)
        print("Connected to the server.")

        command = f'sudo -S useradd {new_username}'
        if is_admin:
            command += ' -G sudo'
        stdin, stdout, stderr = client.exec_command(command)
        stdin.write(f'{password}\n')
        stdin.flush()

        error_message = stderr.read().decode('utf-8')
        if stdout.channel.recv_exit_status() == 0:
            print(f"Користувач '{new_username}' був успішно створений.")
            create_pass(new_password, new_username, client)
        else:
            print("Під час створення користувача сталася помилка.")
            print(error_message)

    except paramiko.AuthenticationException:
        print("Authentication failed.")
    except paramiko.SSHException as e:
        print("Error SSH:", str(e))
    finally:

```

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				84
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        client.close()

def create_pass(new_password, new_username, client):
    command = f'sudo -S passwd {new_username}'
    stdin, stdout, stderr = client.exec_command(command)
    stdin.write(f'{password}\n')
    stdin.write(f'{new_password}\n')
    stdin.write(f'{new_password}\n')
    stdin.flush()

    error_message = stderr.read().decode('utf-8')
    if stdout.channel.recv_exit_status() == 0:
        print(f"Пароль для користувача '{new_username}' був встановлений.")
    else:
        print("Під час створення сталася помилка.")
        print(error_message)

def all_user():
    hostname = input("Введіть ім'я хоста: ")

    client = paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    try:
        client.connect(hostname, username=username, password=password)
        print("Connected to the server.")

        # Виконати команду для перегляду списку користувачів
        stdin, stdout, stderr = client.exec_command('cat /etc/passwd | cut -d: -f1')

        # Отримати вивід команди
        users = stdout.read().decode().split('\n')

        # Видалити останній порожній рядок
        if users[-1] == '':
            users.pop()

        # Вивести список користувачів
        print("Users on the server:")
        for user in users:
            print(user)

```

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

except paramiko.AuthenticationException:
    print("Authentication failed.")
except paramiko.SSHException as e:
    print("Error SSH:", str(e))
finally:
    client.close()

def delete_user():
    hostname = input("Введіть ім'я хоста: ")
    user_to_delete = input("Введіть ім'я користувача для видалення: ")

    client = paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    try:
        client.connect(hostname, username=username, password=password)
        print("Connected to the server.")

        # Створення команди для видалення користувача
        command = f'sudo -S userdel {user_to_delete}'
        stdin, stdout, stderr = client.exec_command(command)
        stdin.write(f'{password}\n')
        stdin.flush()
        error_message = stderr.read().decode('utf-8')

        # Перевірка результату виконання команди
        if stdout.channel.recv_exit_status() == 0:
            print(f"Користувач '{user_to_delete}' був успішно видалений.")
        else:
            print("Під час видалення користувача сталася помилка.")
            print(error_message)

    except paramiko.AuthenticationException:
        print("Помилка автентифікації.")
    except paramiko.SSHException as e:
        print("SSH помилка:", str(e))
    finally:
        client.close()

if __name__ == "__main__":

```

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				86
Змн.	Арк.	№ докум.	Підпис	Дата		

```

parser = argparse.ArgumentParser()
parser.add_argument("function", choices=["create_user", "all_user", "delete_user"],
                    help="Назва функції, яку потрібно виконати")
args = parser.parse_args()

if args.function == "create_user":
    create_user()
elif args.function == "all_user":
    all_user()
elif args.function == "delete_user":
    delete_user()

```

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				87
Змн.	Арк.	№ докум.	Підпис	Дата		

```
import tkinter as tk
from tkinter import filedialog
import os
import subprocess
import re

from server.module.pxe import main, configure_ipxe_pxe_boot
from admin.module.user import create_user, all_user, delete_user
from server.module.apache import install_apache, create_virtual_host, stop_apache,
start_apache, restart_apache, enable_virtual_host, disable_virtual_host

def main_menu():
    print("Меню:")
    print("1. Компіляція та розгортання iPXE")
    print("2. Додавання ОС до конфігурації PXE")
    print("3. Додавання користувача")
    print("4. Виведення всіх наявних користувачів в системі")
    print("5. Видалення користувача")
    print("6. Apache")
    print("7. Вихід")

    choice = input("Виберіть опцію: ")

    if choice == "1":
        main()
    elif choice == "2":
        configure_ipxe_pxe_boot()
    elif choice == "3":
        create_user()
    elif choice == "4":
        all_user()
    elif choice == "5":
        delete_user()
    elif choice == "6":
        apache_menu()
    elif choice == "7":
        return
    else:
        print("Невірний вибір. Будь ласка, спробуйте знову.")
```

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				88
Змн.	Арк.	№ докум.	Підпис	Дата		

```

main_menu()

def apache_menu():
    print("Меню Apache:")
    print("1. Встановлення Apache")
    print("2. Додавання віртуального хосту Apache")
    print("3. Запустити Apache")
    print("4. Зупинити Apache")
    print("5. Перезапустити Apache")
    print("6. Ввімкнути віртуальний хост")
    print("7. Вимкнути віртуальний хост")

    choice = input("Виберіть опцію: ")

    if choice == "1":
        install_apache()
    elif choice == "2":
        domain_name = input("Введіть ім'я домену: ")
        create_virtual_host(domain_name)
    elif choice == "3":
        start_apache()
    elif choice == "4":
        stop_apache()
    elif choice == "5":
        restart_apache()
    elif choice == "6":
        domain_name = input("Введіть ім'я хосту: ")
        enable_virtual_host(domain_name)
    elif choice == "7":
        domain_name = input("Введіть ім'я хосту: ")
        disable_virtual_host(domain_name)
    else:
        print("Невірний вибір. Будь ласка, спробуйте знову.")

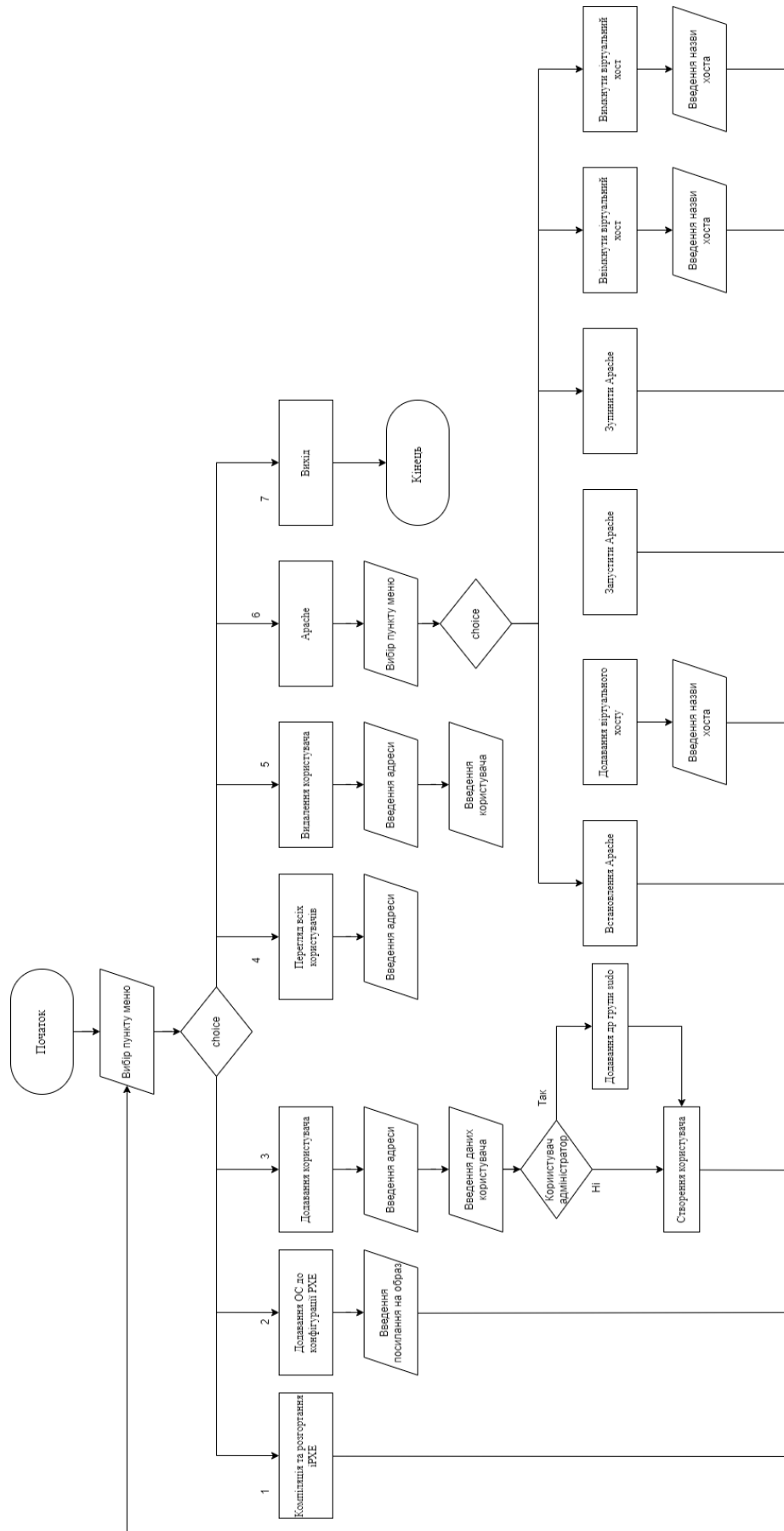
    main_menu()

if __name__ == '__main__':
    main_menu()

```

		Матвієнко М.В.			КІК.КР.Б – 123 – 12 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		89

Додаток Е – Блок-схема управління підсистемою



		Матвієнко М.В.		
		Головня О.С.		
Змн.	Арк.	№ докум.	Підпис	Дата

КІК.КР.Б – 123 – 12 – ПЗ

Арк.

90