МІНІСТЕРСТВО 1ОСВІТИ І НАУКИ УКРАЇНИ

Український державний університет науки і технологій

Кафедра «Електронні обчислювальні машини»

«ДО ЗАХИСТУ»

Завідувач кафедри

_____ Жуковицький І.В.

(підпис) (ПІБ)

«21» 12 2021 р.

ДИПЛОМНА РОБОТА

на здобуття освітнього ступеня «магістр»

Галузь знань 12 Інформаційні технології

(шифр) (назва)

Спеціальність 123 Комп'ютерна інженерія

(код) (повна назва)

Тема Розробка методики автоматизованого розгортання серверів з організацією відмовостійкості

Theme Developmen of a methodology for automated server deployment with faulth tolerance organization

Керівник дипломного проекту проф. (посада) _____ (підпис) Жуковицький І.В. (ПІБ)

Нормоконтролер доцент (посада) _____ (підпис) Шаповалов В.О. (ПІБ)

Студентка групи _____ (група) _____ (підпис) Роща Ю.М. (ПІБ)

Student Roshcha Yuliia

(family name)

Дніпро

2021

Довідка

про відсутність плагіату у випускній кваліфікаційній роботі

Міністерство освіти і науки України
Український державний університет науки і технологій

Кафедра «Електронні обчислювальні машини»

ДОВІДКА

За результатами перевірки випускної кваліфікаційної роботи здобувача вищої освіти __Рощі Юлії Миколаївни_____

на тему: __Розробка методики автоматизованого розгортання серверів з організацією відмовостійкості_____

в роботі не виявлено порушень академічної доброчесності.

Керівник ВКР _____ Жуковицький І.В.

Український державний університет науки і технологій

Факультет Комп'ютерних технологій і систем
Кафедра Електронні Обчислювальні Машини
Спеціальність Комп`ютерна інженерія

«ЗАТВЕРДЖУЮ»
Завідувач кафедри
_____ Жуковицький І. В.
(підпис)
«___»_____20_ р.

ЗАВДАННЯ

до дипломної роботи на здобуття освітнього ступеня _____магістра_____
(освітнього ступеня)

студента групи КС2021 _____Рощі Юлії Миколаївни_____
(номер групи) (ПІБ)

1 Тема дипломної роботи _Дослідження методики автоматизованого розгортання серверів з організацією відмовостійкості_____
_____
_____

затверджена наказом по університету від «09» березня 2021 р. № 131-СТ.

2 Термін подання студентом закінченої роботи____16.12.2021_____

3 Вихідні дані до дипломної роботи_перелік серверного обладнання, CLI-скрипт, KS-файл_____
_____
_____

4 Зміст пояснювальної записки (перелік питань до розробки) _огляд процесів мережевої взаємодії при розгортанні фізичних серверів, моделей та методик автоматизації розгортання серверів з організацією відмовостійкості, дослідження методики автоматизованого розгортання серверів_____
_____

5 Перелік креслень (демонстраційного матеріалу) _____
_____
_____
_____
_____
_____
_____
_____
_____

6 Розділи та консультанти

Розділ	Консультант	Підпис, дата	
		завдання видав	завдання прийняв
Аналіз теми та постановка задачі	Заєць О.П.	_____ _____	_____ _____
Дослідження алгоритму автоматичного розгортання серверів	Заєць О.П.	_____ _____	_____ _____
Дослідження автоматичної моделі розгортання серверів, її тестування і аналіз	Заєць О.П.	_____ _____	_____ _____

КАЛЕНДАРНИЙ ПЛАН

Назва розділу	Термін виконання	Обсяг розділу, %
Аналіз теми та постановка задачі	26.09.2021	30
Дослідження алгоритму автоматичного розгортання серверів	15.10.2021	65
Дослідження автоматичної моделі розгортання серверів, її тестування і аналіз	12.12.2021	100

Дата видачі завдання: «___»_____ 20___ р.

Керівник дипломної роботи _____ ___Заєць О.П.___
(підпис) (ПІБ)

Завдання прийняв до виконання _____ ___Роща Ю.М.___
(підпис) (ПІБ)

РЕФЕРАТ

Пояснювальна записка: 100 стор., 32 рис., 2 таблиці, 1 додаток, 36 джерел.

Об'єкт дослідження: процеси мережевої взаємодії при розгортанні фізичних виділених серверів.

Предмет дослідження: моделі та методики автоматизації розгортання серверів з організацією відмовостійкості.

Мета магістерської роботи: дослідження ефективності розгортання фізичних серверів за допомогою автоматизації рутинних процесів мережевої взаємодії.

Методи дослідження. Для вирішення завдань, поставлених у цій роботі використовуються методи аналізу даних, порівняння існуючих моделей розгортання серверів.

Наукова новизна полягає у тому, що було досліджено процес розгортання фізичних виділених серверів за рахунок обґрунтування доцільності автоматичної моделі розгортання серверів.

Практична цінність полягає у можливості використання методики автоматичного розгортання серверів, що дозволить більш ефективно розподіляти людські, часові та матеріальні ресурси.

У розділі «Економіка» проведені розрахунки трудомісткості розробки програмного забезпечення, витрат на створення ПЗ і тривалості його розробки, а також проведені маркетингові дослідження ринку збуту дослідженного програмного продукту.

Список ключових слів: інформаційні технології, розгортання серверів, операційна система, Unix, Linux, дисковий масив RAID, Ruby, PXE, LLDP.

ABSTRACT

Explanatory note: 100 pages, 32 fig., 2 tables, 1 application, 36 sources.

Object of research: processes of network interaction while dedicated servers' deployment.

Subject of research: models and methods of automation of dedicated servers' deployment with the organization of redundancy.

Purpose of the Master's thesis: investigation of the efficiency of deploying physical servers by automating the routine processes of network interaction.

Research methods. The methods of data analysis, comparison of existing operating systems, object-oriented programming were used in this thesis.

Originality of research is in the fact that the process of deployment of physically dedicated servers was investigated by substantiating the feasibility of an automatic model of server deployment.

Practical value of the results consists of the possibility of using the method of automatic deployment of servers that will allow allocating human, time and material resources more efficiently.

In the Economics section, we calculated the complexity of software development, the cost of software creation and the duration of its development, as well as marketing researches of the market of the created software product is carried out.

Keywords list: information technology, deployment of servers, operating system, Unix, Linux, RAID array, Ruby, PXE, LLDP.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ОС – Операційна система;

СУБД – Система управління базами даних;

USB – Universal Serial Bus;

GPT – Guid Partition Table;

MBR – Master Boot Record;

SNMP – Simple Network Management Protocol;

VPN – Virtual Private Network;

VLAN – Virtual Local Area Network;

BGP – Border Gateway Protocol;

TelNet – Teletype Network;

SSH – Secure Shell;

SMTP – Simple Mail Transfer Protocol;

POP – Post Office Protocol;

SMB – Server Message Block;

IMAP – Internet Message Access Protocol;

HTTP – Hyper Text Transfer Protocol;

HTTPS – Hyper Text Transfer Protocol Secure;

FTP – File Transfer Protocol;

TCP – Transmission Control Protocol;

UDP – User Datagram Protocol;

DHCP – Dynamic Host Configuration Protocol;

ICMP – Internet Control Message Protocol;

BIOS – Basic Input Output System;

UEFI – Unified Extensible Firmware Interface;

SATA – Serial ATA;

ATA – Advanced Technology Attachment;

SSD – Solid State Drive;

IBM – International Business Machines;

PCI – Peripheral Component Interconnect;

GNU – GNU is Not Unix;

RIR – Regional Internet Registry;

LIR – Local Internet Registry;

NAT – Network Address Translation;

PXE – Preboot eXecution Environment;

TFTP – Trivial File Transfer Protocol;

BOOTP – Bootstrap Protocol;

LLDP – Link Layer Discovery Protocol;

IPMI – Intelligent Platform Management Interface;

VNC – Virtual Network Computing;

NIC – Network Interface Controller/Card;

PDU – Power Distribution Unit;

HDD – Hard Drive Disk;

SAS – Serial Attached SCSI;

SCSI – Small Computer System Interface;

NVMe – Non-Volatile Memory express;

DRAC – Dell Remote Access Controller;

iDRAC – integrated Dell Remote Access Controller;

LCC – Lifecycle Controller;

VNC – Virtual Network Computing;

SSL – *Secure Sockets Layer;*

RAID – Redundant Array of Independent Disks;

OSI – Open Systems Interconnection;

MAC – Media Access Control;

DMGMT – Dell Management;

DNS – Domain Name Server/System;

NTP – Network Time Protocol;

LACP – Link Aggregation Control Protocol.

ЗМІСТ

ВСТУП

Актуальність теми кваліфікаційної роботи магістра полягає в тому, що сьогодні більшість підприємств, рід діяльності яких пов’язаний напряму із функціонуванням серверів, надають перевагу ручному налаштуванню, перевірці серверів та встановленні операційної системи витрачаючи багато ресурсів. Таким чином, автоматизація рутинних процесів може мати значний вплив на роботу таких підприємств.

Сьогодення надає можливість знайти будь яку інформацію завдяки доступу до мережі Інтернет, проте усі дані повинні мати фізичне розташування. Ці дані адмініструються, додаються, прибираються, корегуються на серверах.

Розмір підприємства визначає кількість серверів, потрібних для зберігання даних. Невеликі підприємства використовують декілька, компанії, що займаються розробкою, можуть використовувати десяток серверів, об’єднаних у кластер, у якості потужного комп’ютера для проведення великої кількості обчислень. Інтернет гіганти – великі соціальні мережі, як Facebook або Instagram, використовують десятки тисяч фізичних серверів для зберігання своїх даних.

Для коректної роботи кожен з цих серверів має бути правильно налаштований: він має бути під’єднаний до локальної чи глобальної мережі, на ньому має бути встановлена операційна система та налаштовані IP-адреси: приватні, публічні або обидві. Має бути перевірена апаратна та програмна складова перед початком взаємодії із сервером.

В кваліфікаційній роботі буде досліджена модель автоматичного розгортання фізичних виділених серверів із організацією відмовостійкості та запропонований новий підхід до ключових налаштувань сервера та установкою операційної системи.

Метою магістерської роботи є дослідження ефективності розгортання фізичних серверів за допомогою автоматизації рутинних процесів мережевої взаємодії.

Об'єктом дослідження є процеси мережевої взаємодії при розгортанні фізичних виділених серверів.

Предметом дослідження є моделі та методики автоматизації розгортання серверів з організацією відмовостійкості.

Методи дослідження. Для вирішення завдань, поставлених у цій роботі використовуються методи аналізу даних, порівняння існуючих моделей розгортання серверів, об'єктно-орієнтоване програмування.

Наукова новизна визначається тим, що:

- досліджено автоматичну систему розгортання фізичних виділених серверів із організацією відмовостійкості;
- удосконалено процес автоматичної установки обраної операційної системи;
- обґрунтовано доцільність автоматичної моделі розгортання серверів, що дозволяє заощаджувати певні ресурси.

Практична цінність полягає у можливості використання методики автоматичного розгортання серверів, що дозволить більш ефективно розподіляти людські, часові та матеріальні ресурси.

Особистим внеском автора є проведений аналіз серверного обладнання, операційних систем та протоколів мережевої взаємодії, дослідження алгоритму автоматичного розгортання серверів, дослідження моделі з автоматичним розгортанням сервера та установкою найоптимальнішої серверної операційної системи.

Структура та обсяг дипломної роботи. Обсяг кваліфікаційної роботи складає 75 сторінки. Основна частина робота складається із чотирьох розділів. Перший розділ містить системний аналіз предметної області. У другому розділі описано алгоритм розв’язування досліджуваної проблеми, методику її вирішення. Третій розділ містить опис дослідженої програми, інструкцію щодо її використання та результати проведених досліджень та експериментів. Четвертий розділ присвячено оцінці економічної ефективності та містить інформацію про маркетингові дослідження ринку збуту результатів роботи.

РОЗДІЛ 1
АНАЛІЗ ТЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Основні поняття

1.1.1. Фізичний виділений сервер

Сервер – це спеціалізований програмно-апаратний комплекс, призначений для обслуговування мережі організації. Сервери вирішують безліч різних завдань: починаючи з простого файлового сховища із захистом від втрати інформації або управлінням доступу до мережі Інтернет, до завдань по обробці даних. В ідеалі фізичний сервер повинен бути налаштований таким чином, щоб виконувати свої функції без втручання людини, але для початкового налаштування і позаштатних ситуацій він має пристрій введення-виведення для впровадження команд [1].

Найголовніша характеристика сервера – це його продуктивність, яка залежить від декількох параметрів:

1) тип і продуктивність процесорів;

2) обсяг і тип оперативної пам'яті;

3) продуктивність дискової підсистеми [20].

Друга важлива характеристика сервера – його керованість. Мається на увазі, що повинні бути забезпечені такі функції, як віртуальні монітори і інтерфейси діагностика. Тобто бажано, щоб сервером можна було керувати на відстані: включати і перезавантажувати, діагностувати і виправляти недоліки навіть у вимкненому стані (за умови, що він підключений до електричної мережі).

Перші дві характеристики – продуктивність і керованість в значній мірі впливають на надійність сервера [35], що має на увазі не тільки фізичну його надійність і якісну зборку, але і програмну, яка складається в стабільній роботі всіх програм.

Крім перерахованого, слід звернути увагу на масштабованість сервера, що дозволяє значно збільшити його потужність в плані виконання операційною системою обчислювальних операцій [29]. Масштабованість означає, що система має здатність збільшувати потужність в разі збільшення робочого навантаження без зниження таких показників, як надійність і відмовостійкість.

На рис. 1.1 зображений сервер Dell PowerEdge R540, який є прикладом сучасного серверного обладнання.

Рис. 1.1. Сервер Dell PowerEdge R540

Основні типи серверів

Файл-сервер – це централізоване сховище інформації, доступ до дисків якого мають підключені до локальної мережі персональні комп'ютери. Основне завдання файлового сервера зводиться до надійного збереження даних і організації постійного доступу до неї, а в разі пошкодження файлів – повного їх відновлення [1].

Сервер бази даних – засіб не стільки зберігання і доступу, скільки обробки масивів інформації [6]. Через клієнтські запити отримується необхідна інформація, дані обробляються, структуруються, змінюються в залежності від налаштувань сервера. Керують роботою таких серверів СУБД (Системи Управління Базами Даних), найвідоміші з них – MS SQL Server, Oracle, MySQL.

Залежно від кількості користувачів і розміру бази даних, а також перспективи їх збільшення в майбутньому, визначають такі важливі характеристики сервера бази даних, як потужність і масштабованість [2].

Принт-сервер дозволяє використовувати один принтер для обслуговування декількох комп'ютерів. Функції принт-сервера – приймати запити на друк, вибудовувати їх в чергу і, згідно з нею, відправляти на принтер. Таким чином, заощаджуються кошти на комплектацію кожного комп'ютера власним принтером, їх пам'ять звільняється для інших завдань, раціонально використовується офісний простір [4, 8].

Сервер робочої групи – багатофункціональне апаратне рішення для групи комп'ютерів (зазвичай не більше 20). Об'єднує в собі можливості файлового сервера, сервера додатків, бази даних, принт-сервера, поштового та інших, в залежності від потреб. При спільному використанні сервер робочої групи зобов'язаний розмежовувати доступ до даних і права користувачів. Зазвичай має один процесор, найчастіше використовується в невеликих фірмах, де немає потреби у виділенні серверів для окремих завдань [4].

Контролер домену – головний комп'ютер в локальній мережі, що має ієрархічну структуру – домени. Через контролер домену здійснюється централізоване управління ресурсами домену – обліковими записами комп'ютерів і користувачів. За допомогою служби директорій Active Directory він зберігає дані про користувачів і здійснює їх аутентифікацію для доступу до ресурсів локальної мережі. Працює під управлінням серверних ОС від MS Windows, починаючи з Windows 2000 Server. Контролер домену – важливий елемент мережевої інфраструктури великих компаній. Крім того, він може виконувати роль файлового сервера і сервера друку [7].

Поштовий сервер, або сервер електронної пошти, сервер повідомлень – назва говорить сама за себе. Основне завдання такого сервера полягає в розпізнаванні адрес вхідних повідомлень електронної кореспонденції та розподілі її по локальній мережі, відправку вихідної пошти, забезпечення внутрішнього інформаційного обміну. Поштовий сервер забезпечує надійну

фільтрацію спаму і шкідливих програм, що розповсюджуються з повідомленнями, і захищає внутрішню інформацію від несанкціонованого доступу [8].

Сервери FTP – невід'ємна частина технічного забезпечення всесвітньої павутини. Їх завдання переміщати файли за запитом простих файлових менеджерів за допомогою стандартного протоколу FTP. Сучасні сервери FTP вміють розділяти файли за типами і місцем розміщення, обмежувати доступ до них або надавати можливості спільного використання в мережі Інтернет [5].

Проксі-сервер – посередник між користувачами локальної мережі та Інтернетом. Забезпечує безпечний вихід в Інтернет, захищаючи від небажаного доступу зовні і при необхідності обмежуючи вихід на певні ресурси користувачам локальної мережі. Крім того, виконує ряд інших функцій: облік і економія трафіку шляхом стиснення даних, кешування, анонімізація доступу [5].

Web-сервер (сервер web-додатків) – спеціально виділений комп'ютер, який відповідає за доступ до сайту кампанії користувачів Інтернету, коректне і швидке відображення статичних або динамічних сторінок. Веб-сервер зобов'язаний забезпечити безперебійну роботу Інтернет-ресурсу з урахуванням відвідуваності, протистояти мережевим атакам, не допускати можливості злому. Чим більшу роль відіграє Інтернет-сайт в бізнес-процесі (наприклад, забезпечує зв'язок з клієнтами, є каналом збуту продукції), тим важливіше для неї цей сервер. В останні роки веб-сервером називають частіше не саму машину, а програму, що виконує перераховані вище функції [1].

1.1.2. Комутатор як засіб для з’єднання вузлів мережі

Мережевий комутатор (або свіч від англ. Switch – перемикач) – пристрій, призначений для з'єднання декількох вузлів комп'ютерної мережі в межах одного або декількох сегментів мережі. Комутатор працює на канальному (другому) рівні моделі OSI. Комутатори були розроблені з використанням мостових

технологій і часто розглядаються як багатопортові мости [11]. Для з'єднання декількох мереж на основі мережевого рівня служать маршрутизатори (3 рівень OSI).

На відміну від концентратора (1 рівень OSI), який поширює трафік від одного підключеного пристрою до всіх інших, комутатор передає дані лише безпосередньо отримувачу (виняток становить широкомовний трафік всіх вузлів мережі і трафік для пристроїв, для яких невідомий вихідний порт комутатора). Це підвищує продуктивність і безпеку мережі, позбавляючи інші сегменти мережі від необхідності (і можливості) обробляти дані, які їм не призначалися [11].

Принцип роботи мережевого комутатора

В мережевому комутаторі закладена спеціальна схема комутації у вигляді таблиці, яка зберігається в асоціативної пам'яті самого пристрою. В даній таблиці розташовуються MAC-адреси вузлів. Під час запуску комутатора таблиця залишається порожньою. На наступному етапі дані, які надійшли на один з портів автоматично відправляються на всі порти. У цей момент даний пристрій знаходиться в процесі аналізування кадрів, визначивши MAC-адресу відправника залишає його в таблиці. Далі, якщо MAC-адресу клієнта не ініціалізована будь-яким портом, то фрейми (кадри) відправляються на решту портів, крім порту відправника.

На рис. 1.2 зображений мережевий комутатор Brocade VDX 6740T, який є прикладом сучасного серверного обладнання.

Рис. 1.2. Мережевий комутатор Brocade VDX 6740T

Режими комутації мережевого комутатора

Однією з характеристик є режим комутації. Поширені три режими, кожен з яких комбінує в собі режим очікування і рівень надійності:

1. Режим тимчасового зберігання: мережевий комутатор зчитує дані у фреймі, здійснює перевірку на наявність помилок, потім визначає порт і відправляє в нього фрейм.
2. Прямий потоковий: комутатор читає у фреймі тільки адреси, потім виконується процес комутації. Головна перевага даного режиму – висока швидкість передачі даних.
3. Безфрагментний: модифікований варіант прямого режиму. Дані передаються після фільтрації фрагментів на визначенні колізії (конфліктів). Перші 64 байти першого кадру проходять перевірку на наявність колізій (конфліктів), якщо фрейм виявляється пошкоджений або визначається колізія, то передача даних неможлива.

Види мережевих комутаторів

Мережеві комутатори прийнято ділити на два види:

1. Некеровані комутатори – це комутатори, які не мають конфігураційного інтерфейсу або будь-яких інших налаштувань. Це такі пристрої, які працюють за принципом "Plug and Play", наприклад при установці Windows Server 2003 некерований комутатор можна встановити і відразу користуватися.
2. Керовані комутатори є складними пристроями і дозволяють налаштовувати комутацію на мережевому рівні моделі OSI. Мають кілька варіантів зміни режиму роботи: інтерфейс командного рядка, TelNet, Secure Shell, що працюють через протокол управління мережею (SNMP). Приклади конфігурації: настройка пропускної здатності, створення або зміна віртуальної приватної мережі (VPN).

У свою чергу керовані комутатори діляться на два підвиди:

1. Прості – мережеві комутатори з обмеженим набором конфігураційних налаштувань. В даному варіанті надана можливість управління

пристроєм через веб-інтерфейс, а так само такі базові настройки як: настройка VLAN, управління пропускною спроможністю.

2. Складні (корпоративні) комутатори – мають повний набір функціонального управління, в тому числі: CLI, SNMP, веб-інтерфейс. У деяких випадках можна використовувати додаткові конфігураційні функції, наприклад: резервне копіювання і відновлення конфігурацій. Корпоративні комутатори зазвичай використовуються в у великих продуктивних системах і знаходяться в спеціальних стійках. Складні комутатори часто об'єднують в одне мережеве пристрій, іменоване – стек. Робиться це для збільшення кількості портів.

1.1.3. Основні протоколи мережевої взаємодії

Мережевий протокол в комп'ютерних мережах – це набір правил та угод, що використовуються при передачі даних. Протоколи визначають принципи взаємодії комп'ютерів в мережі. Протокол також задає загальні правила взаємодії різноманітних програм, мережевих вузлів чи систем і створює таким чином єдиний простір передачі. Для того, щоб прийняти і обробити відповідним чином повідомлення, вузлам мережі необхідно знати, як сформовані повідомлення і що вони означають [1].

Протокол описує формат повідомлення, якому зобов'язані слідувати, а також спосіб обміну повідомленнями між комп'ютерами в контексті визначеної дії, як, наприклад, пересилка повідомлення по мережі.

Найпоширеніші протоколи мережевої взаємодії

TCP – протокол управління передачею, що орієнтований на роботу із підключеннями і передає дані у вигляді потоку байтів. Ці дані пересилаються у вигляді пакетів (TCP-сегментів), які складаються з заголовків TCP та даних. Цей протокол є надійним, адже він використовує контрольні суми для перевірки

цілісності даних та відправку підтверджень, які можуть гарантувати, що дані було передано без помилок.

UDP – протокол дейтаграм користувача, який не залежить від підключень. Він також передає дані пакетами, що називаються UDP-дейтаграмами, але він не є надійним протоколом, оскільки відправник не отримує інформацію, ка б могла стверджувати, що отримувач дійсно отримав дейтаграму.

BGP – протокол граничного шлюзу, основний протокол динамічної маршрутизації в мережі Інтернет. BGP відрізняється від інших протоколів динамічною маршрутизацією, він призначений для обміну інформації про маршрути не між окремим маршрутизаторами, а між цілими автономними системами і тому, крім інформації про маршрути в мережі, переносить також інформацію про маршрути на автономні системи. BGP не використовує технічні метрики, а здійснює вибір найкращого маршруту виходячи з правил, прийнятих в мережі.

TelNet – мережевий протокол для реалізації текстового інтерфейсу по мережі (у сучасній формі — за допомогою транспорту TCP). У протоколі не передбачено використання ні шифрування, ні перевірки достовірності даних. Тому він вразливий для будь-якого виду атак, до яких вразливий його транспорт, тобто протокол TCP.

DHCP – протокол, який дозволяє вузлам мережі автоматично отримувати IP-адресу та інші параметри, необхідні для роботи у мережі.

SSH – мережевий протокол, що дозволяє проводити віддалене управління комп'ютером і передачу файлів. Клієнти і сервери, що підтримують цей протокол, доступні для різних платформ. Крім того, протокол дозволяє не тільки використовувати безпечний віддалений shell на машині, але і тунелювати графічний інтерфейс X Tunneling (тільки для UNIX-подібних ОС або застосунків, що використовують графічний інтерфейс X Window System). Так само SSH здатний передавати через безпечний канал будь-який інший мережевий протокол, забезпечуючи (при належній конфігурації) можливість безпечної

пересилки не тільки X-інтерфейсу, але і звуку. Підтримка SSH реалізована у всіх UNIX системах [17].

SMTP – це протокол, який використовується для пересилання електронної пошти до поштового сервера або з клієнта-комп'ютера, або між поштовими серверами.

ICMP – протокол, який в основному використовується для передачі повідомлень о помилках та інших ситуацій, які можуть виникати при передачі даних: послуга недоступна, вузол мережі не відповідає.

POP – це протокол, що використовується клієнтом для доступу до повідомлень електронної пошти на сервері. POP3 (остання версія протоколу) дозволяє клієнтові мати вибірковий доступ до повідомлень на сервері. За своєю функціональністю POP є набагато біднішим за IMAP протокол, не надаючи клієнту інтерфейсу з маніпулювання папками на сервері, вибірковим отриманням частин повідомлення чи можливості завантаження заголовків листів.

IMAP – мережевий протокол прикладного рівня для доступу до електронної пошти. Аналогічно POP3, служить для роботи з вхідними листами, однак забезпечує додаткові функції, зокрема, можливість пошуку за ключовим словом без збереження пошти в локальній пам'яті. Цей протокол надає користувачеві великі можливості для роботи з поштовими скриньками, розташованими на центральному сервері. Поштовий клієнт, що використовує цей протокол, отримує доступ до сховища кореспонденції на сервер так, начебто ця кореспонденція розташована на комп'ютері одержувача.

HTTP – протокол передачі даних, що використовується в комп'ютерних мережах. Основним призначенням протоколу HTTP є передача веб-сторінок, хоча за допомогою нього успішно передаються і інші файли, які пов'язані з веб-сторінками (зображення і додатки), так і не пов'язані з ними (у цьому HTTP конкурує з складнішим FTP). HTTPS – це не окремий протокол, а звичайний HTTP, що має додатковий шар шифрування та аутентифікації.

FTP – це протокол, що дає можливість абоненту обмінюватися двійковими і текстовими файлами з будь-яким комп'ютером мережі, що підтримує протокол FTP. Установивши зв'язок з віддаленим комп'ютером, користувач може скопіювати файл з віддаленого комп'ютера на свій, або скопіювати файл з свого комп'ютера на віддалений.

Samba – вільна реалізація мережевого протоколу SMB/CIFS. Samba випускається під ліцензією GNU. Назва Samba походить від SMB – назви протоколу, який використовується Microsoft Windows для мережевої файлової системи. Головною перевагою Samba є те, що з її допомогою можливо використовувати у мережі одночасно комп'ютери з операційними системами Windows та UNIX, організовувати обмін файлами між ними без окремого Windows-сервера.

1.2. Огляд ключових налаштувань сервера

1.2.1. Базова система введення/виведення

BIOS – базова система вводу/виведення у IBM PC-сумісних комп'ютерах. Це тип програмного забезпечення, що зберігається у постійній пам'яті і виконує початкову ініціалізацію машини після її ввімкнення, а також надає спеціальні точки входу для сервісних процедур, що можуть використовуватися операційною системою. Фізично код BIOS записаний у мікросхемах постійної або флеш-пам'яті, розташованих на системній платі комп'ютера. Більшість реалізацій BIOS проектуються і кодуються для роботи з однією специфічною версією чипсета або материнської плати. Використання флеш-пам'яті дозволяє оновлення версії BIOS без використання програматора, але водночас створює потенційну проблему повної втрати працездатності машини у випадку, якщо процес запису не завершився успішно, або якщо такі дії зі знищення інформації у флеш-пам'яті було проведено зумисно (наприклад, вірусом) [2].

Стандарт UEFI є наступником BIOS, спрямованим на усунення її технічних і архітектурних недоліків. Для звичайного користувача головною відмінністю стане більш сучасний та адаптований під сучасні графічні можливості обладнання зовнішній вигляд. Також в UEFI є підтримка миші, таким чином навігація пунктами меню стала набагато швидша [35].

Принципіальними відмінностями UEFI від BIOS є те, що UEFI завантажується швидше, є більш безпечною прошивкою, а також підтримує розбивку на розділи GTP замість MBR. Таким чином ігнорується обмеження BIOS по максимальному об'єму дисків, з яких загружається система. Для BIOS це обмеження складає всього лише 2,2 ТБ [34].

На рис. 1.3 зображено головне меню BIOS (System Setup) сервера Dell PowerEdge R440.

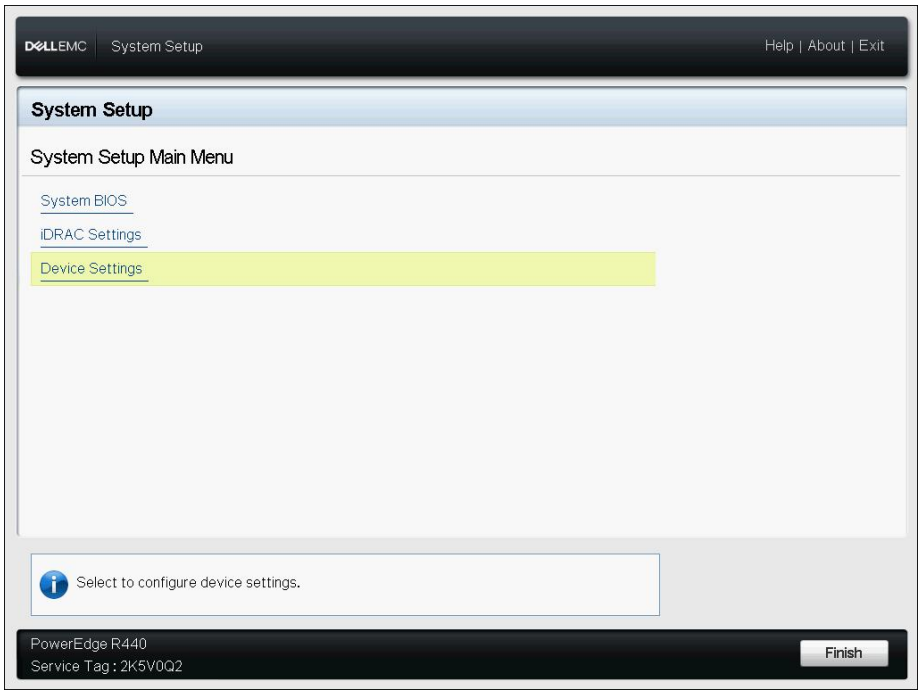


Рис. 1.3. Головне меню BIOS сервера Dell PowerEdge R440

Головне меню складається з трьох пунктів:

1) System BIOS;

2) iDRAC Settings;

3) Device Settings.

Пункт меню “System BIOS” складається з таких підпунктів, як:

1) System Information – пункт відображає базову інформацію про систему: модель фізичного сервера, версію прошивок BIOS, серійний номер сервера, виробника та іншу інформацію;
2) Memory Settings – показує загальний обсяг встановлених модулів оперативної пам’яті, тип модулів, частоту модулів, об’єм відеопам’яті, наявність автоматичного тестування модулів оперативної пам’яті при увімкненні сервера;
3) Processor Settings – системна інформація і налаштування, пов’язані з процесором, встановленим у сервер;
4) SATA Settings – інформація щодо SATA, SSD, SAS носіїв, встановлених у сервер і підключених до сервера напряму, без використання апаратного RAID-контролера;
5) NVMe Settings – інформація щодо режиму NVMe носіїв;
6) Boot Settings – режим запуску сервера (BIOS / UEFI), а також порядок запуску сервера (з якого з носіїв буде запускатись сервер);
7) Integrated Devices – інформація щодо наявності додаткових девайсів, наприклад, USB-накопичувачів, оптичних дисководів, мережевих карт, вбудованих у сервер та інше;
8) Serial Communication – налаштування послідовного зв’язку сервера;
9) System Profile Settings – вибір робочого профілю сервера (спрямований на максимальну продуктивність, максимальну енергоефективність або налаштування особистого профілю користувача);
10) System Security – налаштування системної безпеки, системного паролю та дуже важлива функція для серверів, що знаходяться фізично у віддалених датацентрах – AC Power Recovery, що дозволяє автоматично умикати сервер за умови подання на нього електричного струму;
11) Redundant OS Control – налаштування системної відмовостійкості;

12) Miscellaneous Settings – налаштування системної дати, часу, часового поясу та інші налаштування.

На рис. 1.4 зображено розгорнутий пункт меню “System BIOS” разом із вищеперерахованими підпунктами.

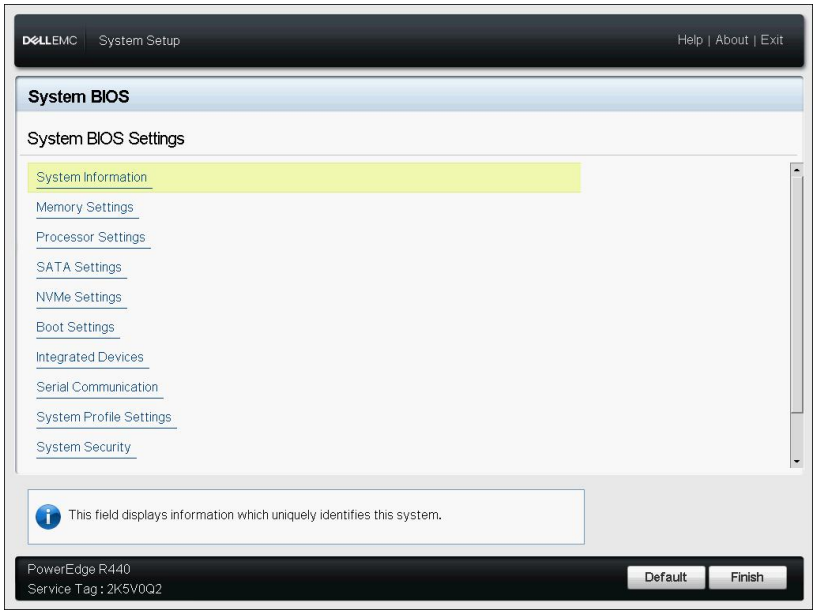


Рис. 1.4. Розгорнутий пункт меню “System BIOS” сервера Dell PowerEdge R440

Пункт меню “System BIOS” складається з таких підпунктів, як:

1) System Summary – розгорнута інформація про сервер: версія деяких прошивок, MAC-адреса інтегрованої плати DRAC, налаштування IPv4 та IPv6, кількість корзин для встановлення носіїв інформації та інша системна інформація;
2) System Event Log – кількість записів у системному логу, можливість передивитися системний лог;
3) Network – деякі мережеві налаштування;
4) Alerts – налаштування сповіщень;
5) Front Panel Security – налаштування безпеки передньої панелі, можливість позбавити будь-якого ефекту фізичну кнопку умикання сервера;

6) Media and USB Port Settings – налаштування віртуальних медіа-носіїв, наприклад, DVD-дисків або USB-накопичувачів;
7) Lifecycle Controller – налаштування наявності LCC;
8) Power configuration – налаштування блоків живлення сервера, вибір основного та резервного блоків (у випадку з сервером, що має більше одного блоку живлення);
9) Thermal – температурні налаштування, вибір профілю користування та швидкість обертів системних кулерів;
10) System Location – вільні поля для заповнення інформації щодо знаходження сервера у віддаленому датацентрі (назва датацентру, номер стійки, позиція у стійці);
11) User Configuration – налаштування ім’я користувача та пароля root, інші налаштування користувачів;
12) Smart Card – налаштування Smart Card: можливість використання клавіатури із пристроєм зчитування пластикових карт, наприклад, у випадку взаємодії покупця із продавцем невеликого магазину напряму через сервер;
13) Communications Permissions – налаштування дозволів послідовного зв’язку сервера;
14) Remote Enablement – налаштування можливостей керування віддаленим вузлом мережі, операційна система якого не підтримує звичайного управління, наприклад VMWare ESXi;
15) Reset iDRAC configuration to defaults – приведення серверу до заводських налаштувань, окрім налаштувань мережі та даних, що задані користувачем;
16) Reset iDRAC configuration to defaults all – повне приведення серверу до заводських налаштувань, включаючи налаштування мережі і даних, що задані користувачем;
17) System Lockdown Mode – наявність режиму Lockdown, який передбачає захист сервера від несанкціонованої зміни апаратної конфігурації.

На рис. 1.5 зображено розгорнутий пункт меню “iDRAC Settings” разом із вищеперерахованими підпунктами.

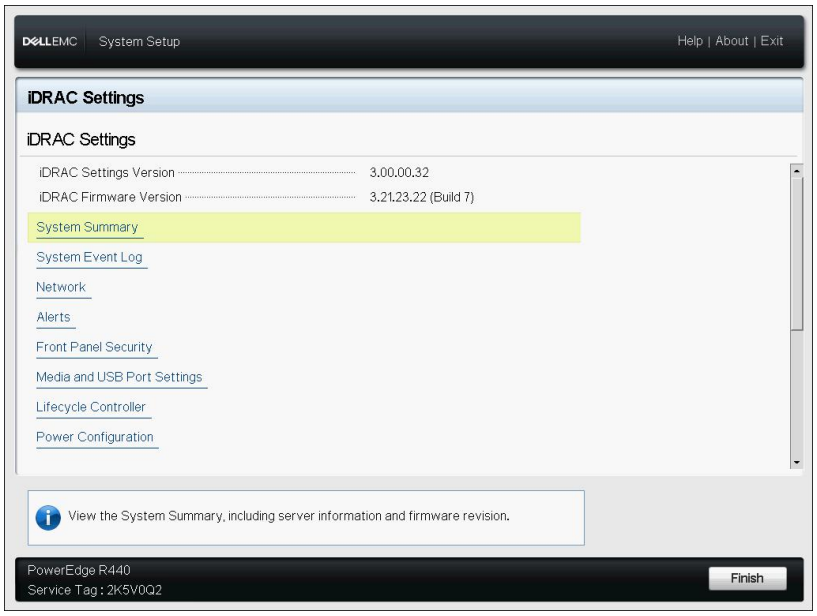


Рис. 1.5. Розгорнутий пункт меню “iDRAC Settings” сервера Dell PowerEdge R440

Пункт меню “Device Settings” є переліком девайсів, встановлених у сервер. Такими девайсами можуть бути вбудовані або зовнішні мережеві карти, пристрої для зчитування інформації з компакт-дисків чи USB-накопичувачів, а також апаратний RAID-контролер. Для апаратного RAID-контролера наявне своє меню, у якому можна побачити перелік носіїв інформації, які під’єднані до цього контролера, рівень RAID, а також доступні базові операції по роботі із масивами даних: створення нового масиву з наявних дисків, знищення існуючого масиву, зміна рівню RAID, імпорт чужої конфігурації, якщо така конфігурація наявна. Для вбудованих та зовнішніх мережевих карт можна побачити їхні MAC-адреси, пропускну спроможність портів, актуальні версії прошивок, а також виконати базові налаштування, серед яких варто підкреслити параметр “Legacy Boot Protocol”, завдяки якому пізніше буде здійснюватися автоматичне розгортання серверу із встановленням операційної системи [34].

На рис. 1.6 зображено розгорнутий пункт меню “Device Settings”, на якому можна побачити вбудовану (Broadcom Gigabit Ethernet), зовнішні (Intel X550-t) мережеві карти, а також апаратний RAID-контролер Perc H330.

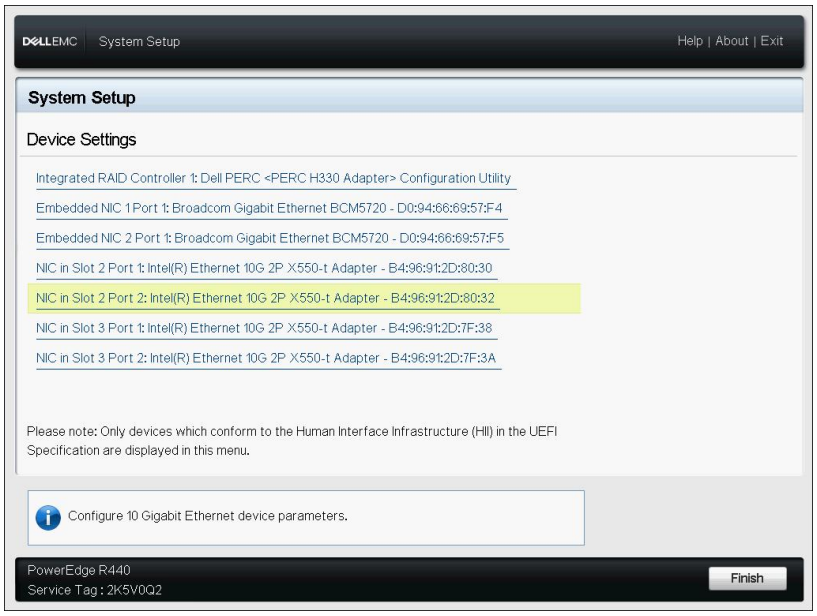


Рис. 1.6. Розгорнутий пункт меню “Device Settings”

1.2.2. Засіб віддаленого керування iDRAC

iDRAC – засіб віддаленого керування серверами від виробника Dell. iDRAC представляє собою інтерфейсну плату. Інтерфейс Dell iDRAC обладнаний своїм процесором, модулем оперативної пам’яті, системною батарейкою і мережевим інтерфейсом зі своєю MAC-адресою, щоб отримати доступ до системної шини.

Ця інтерфейсна плата може буди реалізована у вигляді окремої плати (DRAC) або бути інтегрованою у материнську плату сервера (iDRAC). На моделях серверів Dell, новіших за 11 покоління (R220 та більш нові сервери), як правило, є вбудована плата iDRAC. На старих поколіннях серверів вбудована плата iDRAC відсутня. Для користувачів серверів без вбудованої плати iDRAC є можливість доустановити її, але за окрему плату, що складає близько 150 доларів США.

На рисунку 1.7 зображена плата DRAC, яка не є інтегрованою у материнську плату сервера і постачається окремо за додаткову оплату.

Рис.1.7. Плата DRAC

Основними функціями WEB-інтерфейсу iDRAC є:

1) можливість підключатися до віртуальної консолі сервера;
2) управління електроживленням;
3) отримання системної інформації про сервер без входу у BIOS та без перезавантаження сервера (кількість і об’єм модулів оперативної пам’яті, носіїв інформації, мережевих карт, їх серійні номери);
4) моніторинг фізичного стану сервера;
5) монтування віртуальних образів, наприклад, для встановлення операційної системи;
6) віддалене включення і перезавантаження сервера;
7) оновлення прошивок.

Доступ до WEB-інтерфейсу iDRAC можна отримати за допомогою будь-якого браузеру. Для цього необхідно ввести IP-адресу iDRAC в адресну строку браузера, а потім ввести актуальний логін та пароль доступу. Також є можливість конфігурації існуючих користувачів системи, додавання нових, регуляція їх повноважень.

iDRAC використовує протокол HTTPS та самозавірений сертифікат, тому більшість сучасних браузерів будуть сигналізувати про небезпеку сайту, але це попередження можна ігнорувати.

На рис. 1.8 зображений головний екран WEB-інтерфейсу iDRAC.

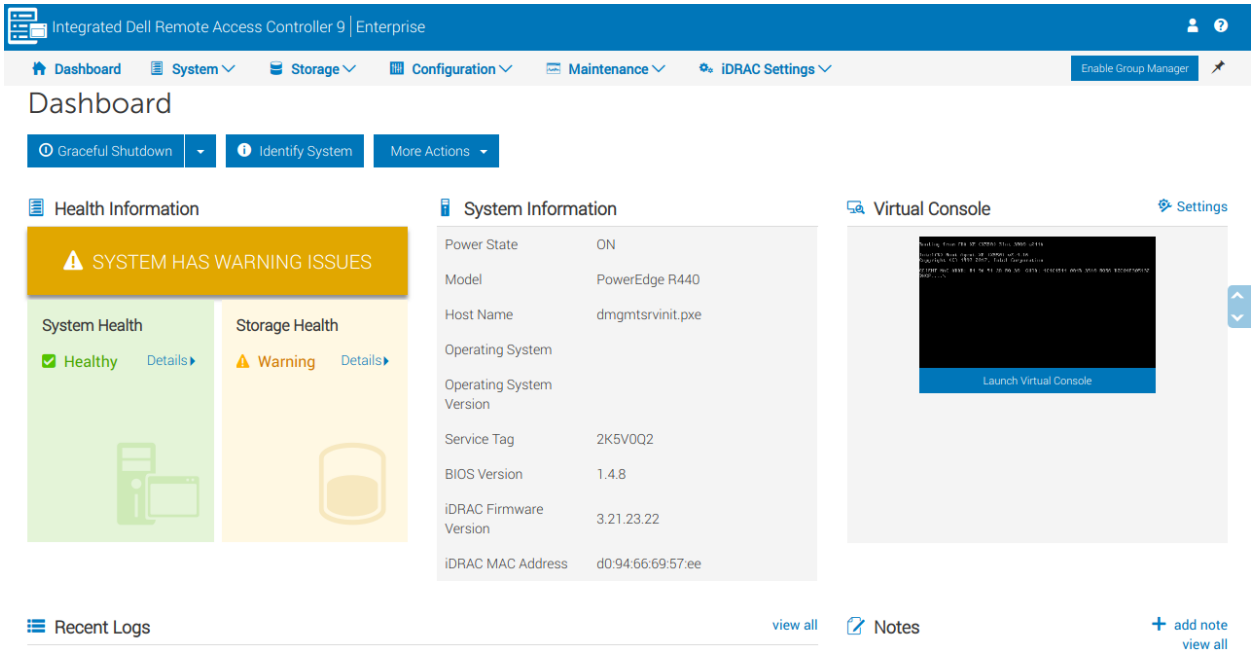


Рис. 1.8. Головний екран WEB-інтерфейсу iDRAC

На головному екрані можна побачити короткий лист системної інформації у блоці “System Information”, загальні відомості про “здоров’я” сервера у блоці “Health Information”, отримати доступ до віртуальної консолі сервера у блоці “Virtual Console”, а також побачити останній системний лог у блоці “Recent Logs”.

У пункті “Summary” у вкладці “Overview” знаходяться наступні пункти:

1) Summary – загальна інформація про сервер, його модель, серійний номер та графіки використання електричної енергії;
2) Batteries – ім’я та статус батарейки у модулі iDRAC;
3) Cooling – інформація щодо роботи системних кулерів сервера, налаштування роботи, температурні показники;
4) CPUs – стисла та розгорнута інформація о процесорах сервера;

5) Front Panel – симуляція LED-вогників передньої частини сервера;
6) Intrusion – пункт показує, чи закриті шасі сервера у момент часу;
7) Memory – інформація щодо кількості та об’єму модулів оперативної пам’яті, встановленої у сервер, тип модулів та їхня частота;
8) Network Devices – інформація щодо мережевих карт (інтегрованих чи додаткових зовнішніх): MAC-адреси, пропускна спроможність портів, модель;
9) Power Supplies – інформація щодо блоків живлення сервера: їх фізичний стан, вольтаж та частота;
10) Voltages – інформація щодо напруги на різних вузлах сервера.

На рис. 1.9 зображений пункт меню “System” WEB-інтерфейсу iDRAC із відкритою вкладкою “Overview”.

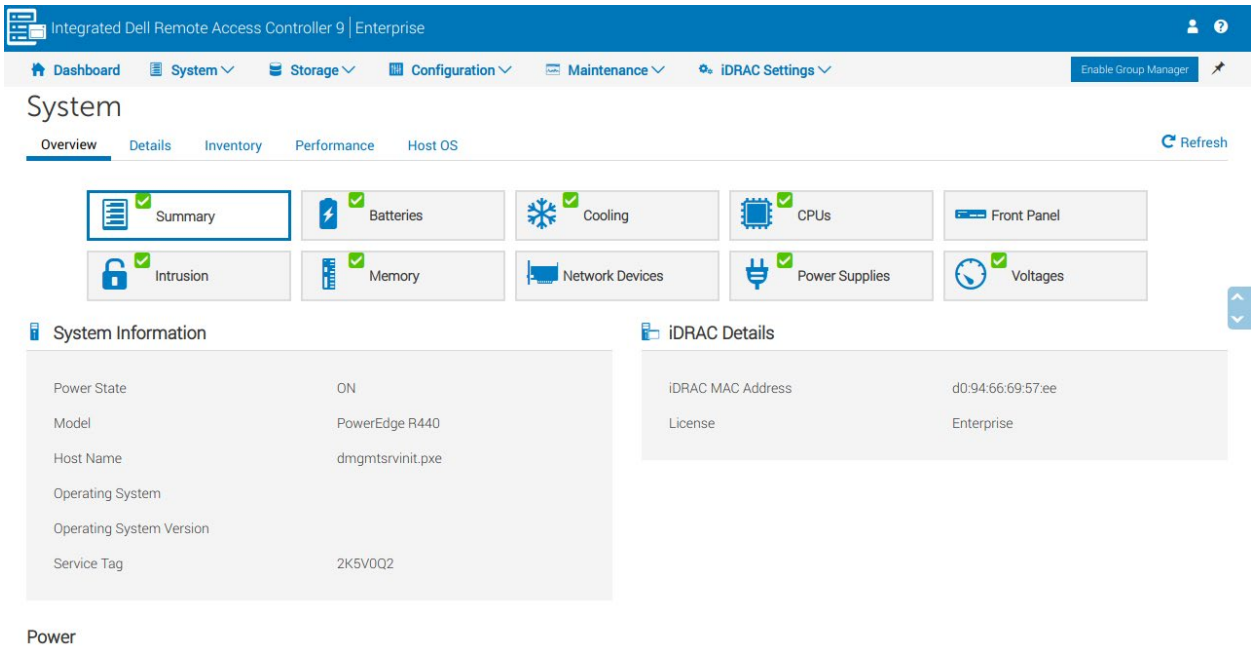


Рис. 1.9. Пункт меню “System” WEB-інтерфейсу iDRAC із відкритою вкладкою “Overview”

Наступний пункт меню – “Storage” дозволяє керувати та переглядати інформацію щодо носіїв інформації, встановлених у сервер. Для цього доступні 5 підпунктів:

1) Summary – наглядна інформація щодо кількості фізичних та віртуальних дисків, встановлених у сервер, а також лог за останній час;
2) Controllers – інформація щодо апаратного RAID-контролера, встановленого в сервер;
3) Physical Disks – інформація щодо кількості, типу, об’єму, виробника та серійних номерів усіх носіїв інформації, встановлених у сервер;
4) Virtual Disks – інформація про всі віртуальні диски, що існують у системи, а також з яких фізичних дисків вони складаються;
5) Enclosures – загальна інформація про наявні слоти.

На рис. 1.10 зображений пункт меню “Storage” WEB-інтерфейсу iDRAC.

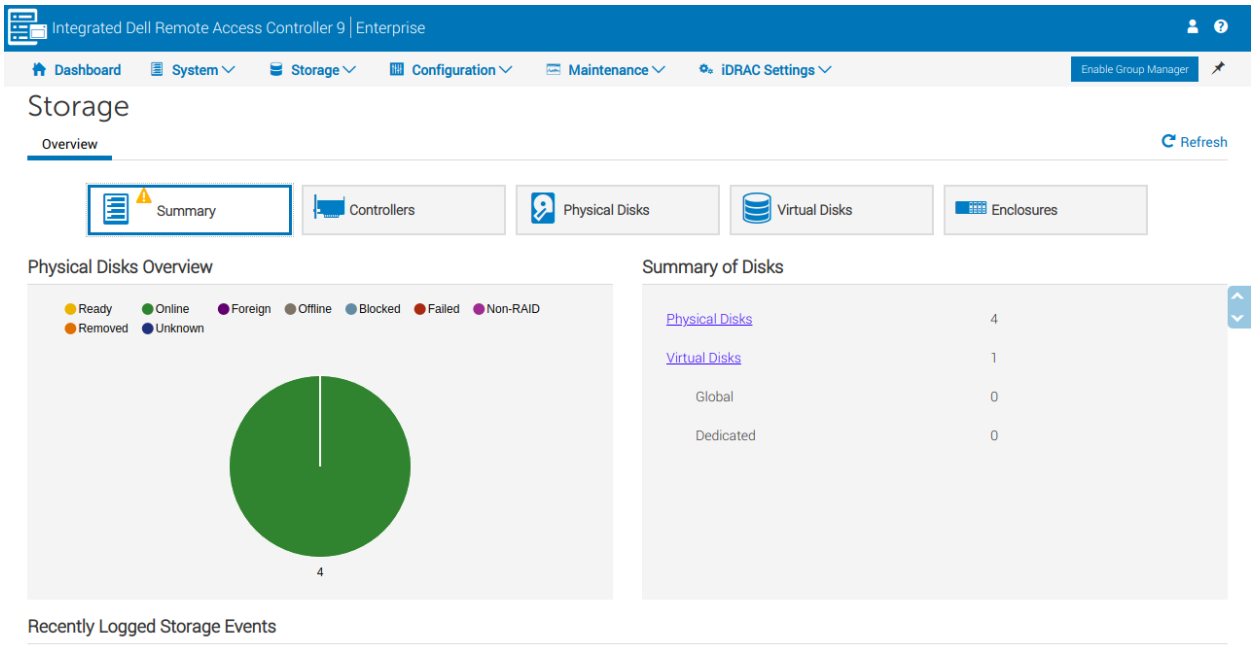


Рис. 1.10. Пункт меню “Storage” WEB-інтерфейсу iDRAC

Наступний пункт “Configuration”, на відміну від попередніх, призначений для проведення налаштувань і складається з наступних пунктів:

1) Power Management – налаштування лімітів та політик використання струму, вибір основного блоку живлення, вимкнення сервера та інші налаштування;

2) Virtual Console – налаштування віртуальної консолі, підключення до неї, налаштування VNC-консолі;
3) Virtual Media – налаштування віртуальних пристроїв для встановлення операційної системи тощо;
4) Licenses – управління ліцензіями iDRAC;
5) System Settings – конфігурація кулерів та іншого технічного обладнання, налаштування індикації та сповіщень;
6) Storage Configuration – створення нових, управління та видалення існуючих віртуальних дисків системи, зміна стану дисків із “Non-RAID” до “Ready” без вимикання сервера;
7) BIOS Settings – налаштування, ідентичні до налаштувань у BIOS;
8) Server Configuration Profile – імпорт або експорт існуючої конфігурації із налаштуваннями.

На рис. 1.11 зображений пункт меню “Configuration” WEB-інтерфейсу iDRAC.

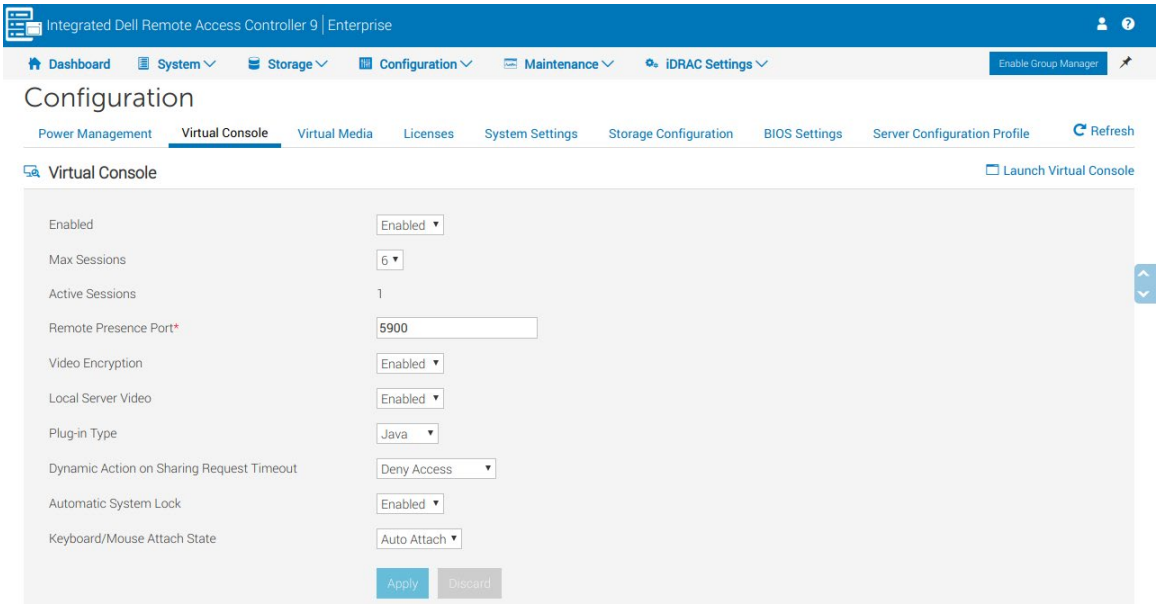


Рис. 1.11. Пункт меню “Configuration” WEB-інтерфейсу iDRAC.

Пункт “Maintenance” призначений для моніторингу та деяких робіт, які потрібно проводити на сервері час від часу, наприклад оновлення прошивок, тестування апаратної частини. Цей пункт складається з наступних вкладок:

1) Lifecycle Log – лог із повідомленнями, ідентифікаційними номерами ситуацій та стислим описом;
2) Job Queue – робоча черга, яка складається із завершених, запланованих та перебуваючих у процесі виконання завдань, наприклад, створення нового RAID-масиву;
3) System Update – оновлення прошивок iDRAC вручну;
4) System Event Log – стислий лог повідомлень, пов’язаних із важливими системними повідомленнями, наприклад, про диск, що вийшов з ладу;
5) Troubleshooting – запис останніх подій на відео та перегляд таких відео;
6) Diagnostics – консольна командна строка для виконання діагностичних команд, а також кнопки перезавантаження актуальної сесії і скидання усіх налаштувань до заводських;
7) SupportAssist – пункт для реєстрації і прямого зв’язку з інженерами Dell.

На рис. 1.12 зображений пункт меню “Maintenance” WEB-інтерфейсу iDRAC.

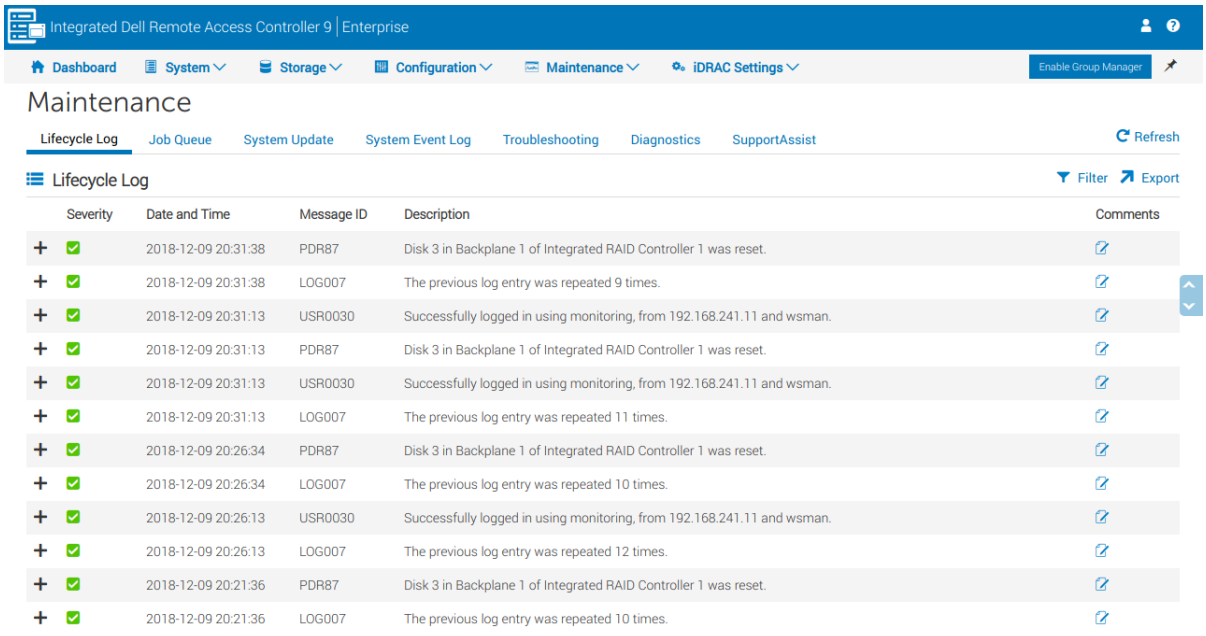


Рис. 1.12. Пункт меню “Maintenance” WEB-інтерфейсу iDRAC.

Останній пункт “iDRAC Settings” призначений для налаштувань і моніторингу мережевих параметрів, конфігурації на коригування існуючих користувачів, обрання їх прав тощо. Цей пункт складається з наступних вкладок:

1) Overview – огляд встановлених налаштувань IPv4, IPv6 мереж, інформація щодо останнього оновлення прошивок;
2) Connectivity – вкладка для ручного налаштування мережевих параметрів, наприклад статичну IP-адресу, статичну адресу шлюзу, SSL-сертифікатів тощо;
3) Services – налаштування можливості взаємодії з сервером шляхом SSH, TelNet, SNMP, VNC-консолі тощо;
4) Users – конфігурація, створення та видалення користувачів, обмеження їх прав та зміна паролю;
5) Settings – налаштування часової зони та профілю сервера.

На рис. 1.13 зображений пункт меню “iDRAC Settings” WEB-інтерфейсу iDRAC.

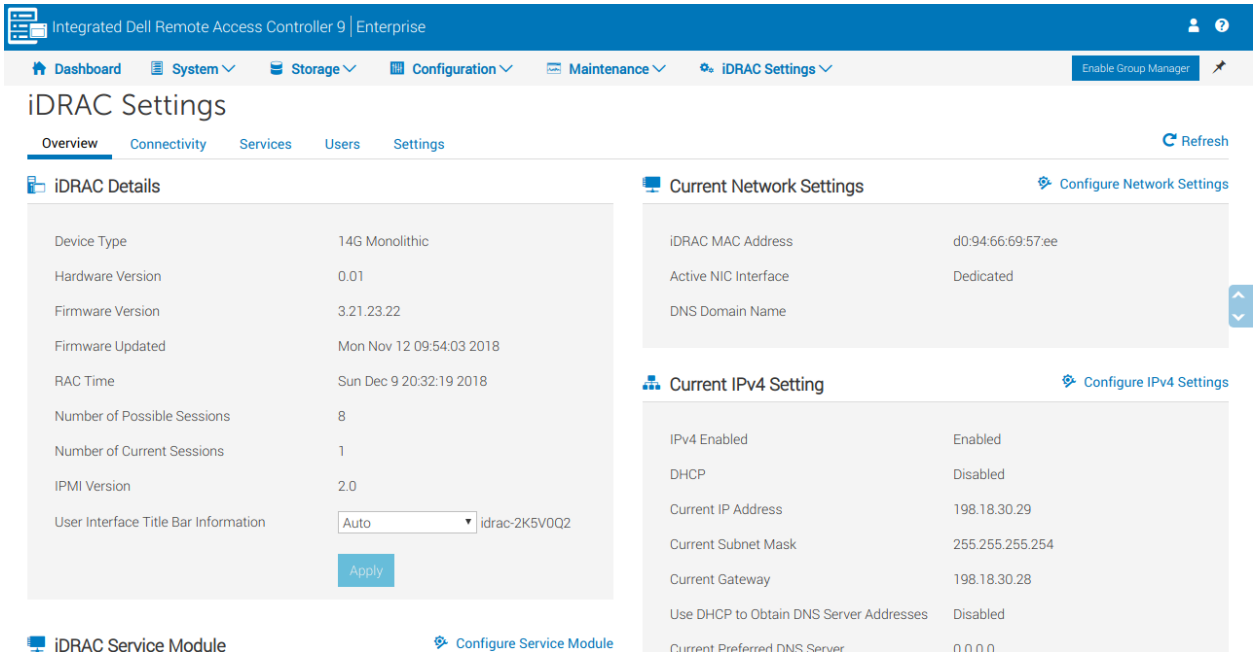


Рис. 1.13. Пункт меню “iDRAC Settings” WEB-інтерфейсу iDRAC

1.2.3. Технологія віртуалізації даних RAID

RAID – технологія віртуалізації даних, заснована на об’єднанні декількох фізичних дисків в один віртуальний [20]. Це може сприяти суттєвому збільшенню відмовостійкості зберігання даних або підвищувати продуктивність функціонування системи за рахунок прискорення операцій читання та запису.

Технологія RAID вперше була представлена в 1987 році, її створили Девід Петтерсон, Гарет Гібсон та Ренді Катцом [25].

Розрізняють апаратний та програмний RAID-масиви. Апаратний характеризується тим, що всі диски, які входять до масиву, мають бути підключені до RAID-контролера [19]. У цьому випадку такі параметри, як максимальна кількість дисків у масиві, можливі рівні масиву ті деякі інші параметри будуть залежати від встановленого апаратного RAID-контролера [22].

На рис. 1.14 зображено апаратний RAID-контролер Perc H730, який підтримує підключення до 255 дисків та підтримує найбільш поширені рівні масивів.

Рис. 1.14. Апаратний RAID-контролер Perc H730

Якщо сервер не має встановленого апаратного RAID-контролера, то деякі сучасні операційні системи надають користувачеві можливість створення програмного масиву. Наприклад, в операційних системах сімейства Linux підтримка програмного RAID існує безпосередньо на рівні ядра системи [28]. Функції управління RAID-пристроями виконує утиліта mdadm. Перевагою програмного RAID-масиву є те, що він нічого не коштує у той час, як ціни на апаратні RAID-контролери починаються із 120 доларів США. Проте програмний RAID використовує ресурси центрального процесора, що має вплив на загальну продуктивність сервера.

Сьогодні існує досить багато базових рівнів RAID: RAID 0, RAID 1, RAID 2, RAID 3, RAID 4, RAID 5 та RAID 6, кожен з яких має певні недоліки та переваги. Також існують комбіновані рівні, утворені комбінацією базових: RAID 10, RAID 50, RAID 60 тощо [23].

Як показує практика, сьогодні найпоширенішими рівнями масивів з базових рівнів є наступні:

1. RAID 0 – дисковий масив з двох чи більше дисків, в яких відсутнє резервування. Інформація розбивається на блоки А фіксованої довжини та записується на N дисків, що входять до масиву, по черзі. Таким чином досягається швидкість, більша в N разів, аніж швидкість будь-якого диска масиву. З іншого боку це також призводить до збільшення ймовірності втрати усіх даних, що записані на масив, у разі відмови будь-якого з дисків. На рис. 1.15 схематично зображено побудову масиву RAID 0.

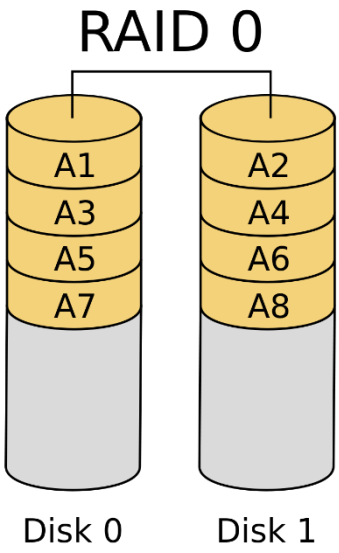


Рис. 1.15. Схема побудови масиву RAID 0

2. RAID 1 – дисковий масив з парною кількістю дисків, які є повними копіями один одного. При цьому користувач не має збільшення швидкості читання та запису, проте отримує майже безвідмовну систему, адже вона може функціонувати аж поки в системі є хоча б один справний диск у масиві. Цей рівень вимагає наявності параметру гарячого резерву – запасного диску, який заміняє той диск, що вийшов з ладу, без припинення функціонування системи. На рис. 1.16 схематично зображено побудову масиву RAID 1.

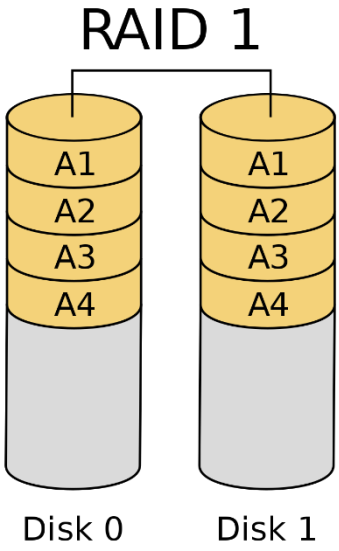


Рис. 1.16. Схема побудови масиву RAID 1

3. RAID 5 – дисковий масив з мінімальною кількістю дисків, що дорівнює трьом. При побудові цього рівня RAID дані та контрольні суми циклічно записуються на усі диски масиву. Перевагою цього рівня масиву є висока економічність, яка зростає пропорційно збільшенню кількості дисків у масиві. Недоліком є менша продуктивність, адже потребує більшої кількості операцій читання та запису з дисками, а також ризик пошкодження деякого з дисків у процесі відновлення масиву. Варто зазначити, що рівень RAID 5 майже не застосовується у такому вигляді. Замість цього він активно використовується для будування комбінованих рівнів, наприклад RAID 50.

На рисунку 1.17 схематично зображено побудову масиву RAID 5.

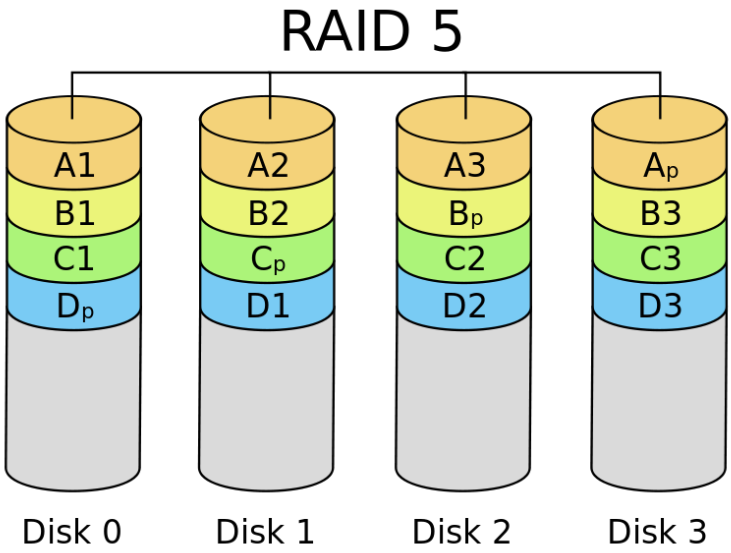


Рис. 1.17. Схема побудови масиву RAID 5

4. RAID 6 – дисковий масив з мінімум п’ятьма дисками, з яких три використовуються для запису даних, а два – для запису контрольних сум. Перевагою цього рівня масиву є висока відмовостійкість (масив продовжує функціонувати після одночасного виходу з ладу двох з п’яти дисків). Недоліком, як і у випадку з масивом рівня RAID 5, є менша продуктивність за рахунок необхідності взаємодії з усіма дисками масиву і перезапису більшої кількості дискових блоків.

На рис. 1.18 схематично зображено побудову масиву RAID 6.

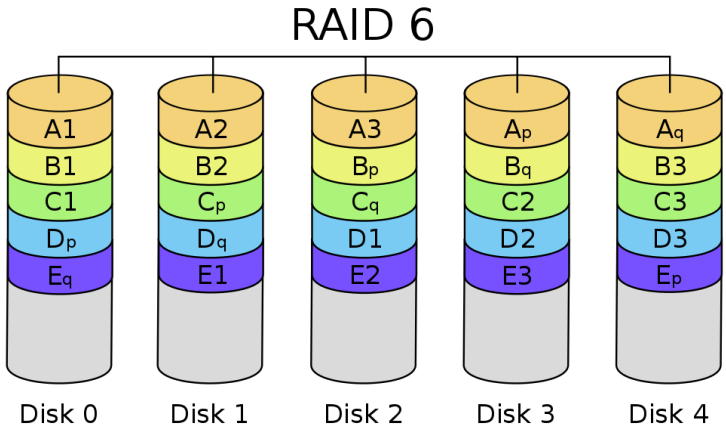


Рис. 1.18. Схема побудови масиву RAID 6

При порівнянні рівнів RAID необхідно брати до уваги такі параметри як надійність, швидкість читання та запису, а також мінімальну кількість дисків, з яких може бути утворено масив [26]. Також слід звертати увагу на показник максимального числа дисків масиву, що можуть вийти з ладу одночасно [24].

Порівняльну характеристику основних рівнів масивів RAID, згаданих у цій роботі, можна побачити у табл. 1.1. Приймемо кількість дисків у масиві як N та об’єм найменшого диска як S.

Таблиця 1.1

Порівняльна таблиця деяких рівнів масивів RAID

Рівень RAID	Min дисків	Ефективна ємність	Max дисків, що можуть вийти з ладу одночасно	Надійність	Швидкість читання	Швидкість запису
0	Від 2	S * N	0	Дуже низька	Висока	Висока
1	Від 2	S	N -1	Висока	Висока	Середня
5	Від 3	S * (N-1)	1	Середня	Висока	Середня
6	Від 4	S * (N - 2)	2	Висока	Висока	Низька чи середня
10	Парна, від 4	S * N/2	Від 1 до N/2	Середня	Висока	Висока

Якщо в сервері встановлений апаратний RAID-контролер, то при запуску сервера за допомогою комбінації клавіш “Ctrl + R” можна увійти в меню контролера.

Меню RAID-контролера дозволяє виконувати такі прості операції, як створення, змінення чи видалення віртуальних дисків, відновлення масиву після виходу з ладу диска, перегляд фізичних дисків, їх стан.

На рис. 1.19 зображено зовнішній вигляд меню RAID-контролера Perc H330.

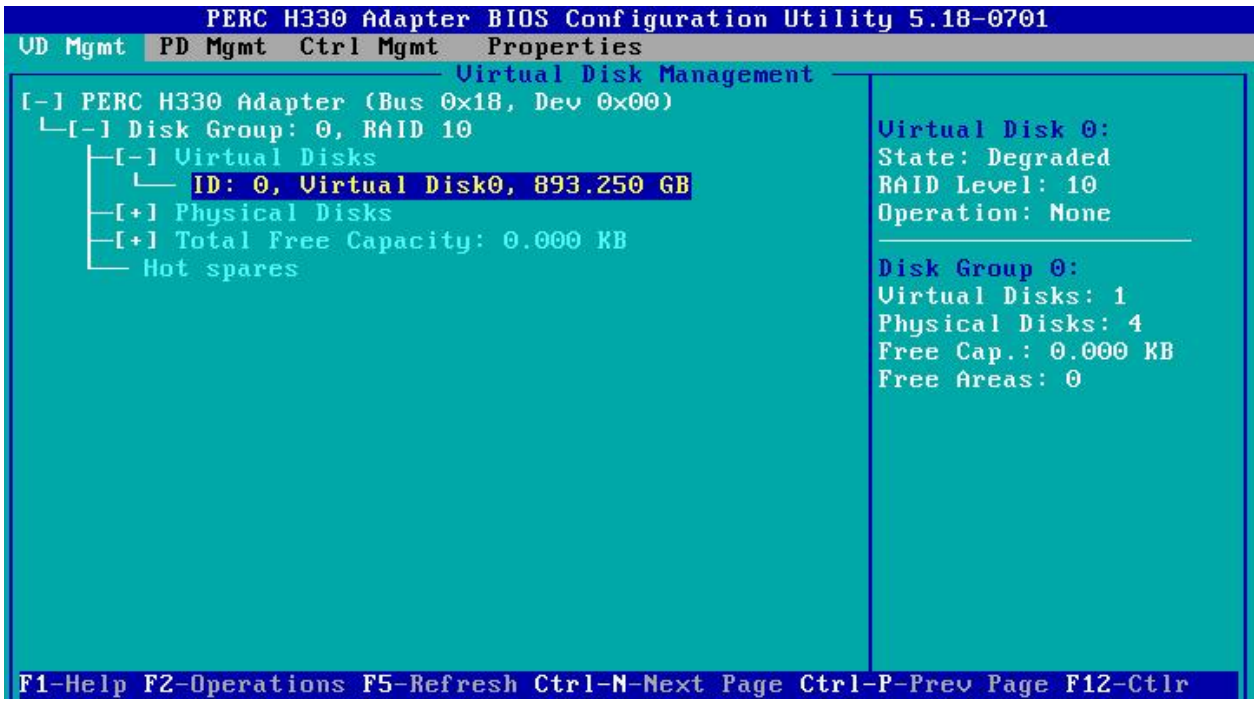


Рис. 1.19. Зовнішній вигляд меню RAID-контролера Perc H330

1.2.4. Поняття приватної та публічної мережі, IP-адреса

IP-адреса – унікальний ідентифікатор (адреса) пристрою, підключеного до локальної мережі або Інтернету.

IP-адреса являє собою 32-бітове (за версією IPv4) або 128-бітове (за версією IPv6) двійкове число. Зручною формою запису IP-адреси (IPv4) є запис у вигляді чотирьох десяткових чисел (від 0 до 255), розділених крапками,

наприклад, 192.168.0.1. (Або 128.10.2.30 – традиційна десяткова форма представлення адреси, а 10000000 00001010 00000010 00011110 – двійкова форма представлення цього ж адреси).

IP-адреси являють собою основний тип адрес, на підставі яких мережевий рівень протоколу IP передає пакети між мережами. IP-адреса призначається адміністратором під час конфігурування комп'ютерів і маршрутизаторів.

IP-адреса складається з двох частин: номера мережі і номера вузла. У разі ізольованої мережі її адресу може бути обраний адміністратором із спеціально зарезервованих для таких мереж блоків адрес (192.168.0.0/16, 172.16.0.0/12 або 10.0.0.0/8). Якщо ж мережа повинна працювати як складова частина Інтернету, то адреса мережі видається провайдером або регіональним інтернет реєстратором RIR. Всього існує п'ять RIR: ARIN, обслуговуючий Північну Америку; APNIC, обслуговуючий країни Південно-Східної Азії; AfriNIC, обслуговуючий країни Африки; LACNIC, обслуговуючий країни Південної Америки і басейну Карибського моря; і RIPE NCC, обслуговуючий Європу, Центральну Азію, Близький Схід. Регіональні реєстратори отримують номери автономних систем і великі блоки адрес у ICANN, а потім видають номери автономних систем і блоки адрес меншого розміру локальним інтернет реєстраторам LIR, зазвичай є великими провайдерами.

Номер вузла в протоколі IP призначається незалежно від локальної адреси вузла. Маршрутизатор по визначенню входить відразу в кілька мереж. Тому кожен порт маршрутизатора має власну IP-адресу. Кінцевий вузол також може входити в кілька IP-мереж. У цьому випадку комп'ютер повинен мати кілька IP-адрес, по числу мережевих зв'язків. Таким чином, IP-адреса характеризує не окремий комп'ютер або маршрутизатор, а одне мережеве з'єднання [1].

IP-адреса називають динамічним, якщо він призначається автоматично при підключенні пристрою до мережі і використовується протягом обмеженого проміжку часу, як правило, до завершення сеансу підключення [2].

Розрізняють публічні (або глобальні, або зовнішні) “білі” та приватні (або локальні, або внутрішні) “сірі” IP-адреси. У мережі Інтернет використовуються

саме публічні адреси, які використовуються для виходу у Інтернет. Такі адреси маршрутизуються в Інтернеті і можуть отримувати трафік із Інтернету. Приватні IP-адреси не маршрутизуються в Інтернеті, відправити напряму трафік із Інтернету на такий адрес неможливо. Існує можливість використання NAT для трансляції мережевих адрес та заміни приватної адреси на публічну. Приватні адреси вільно функціонують у межах локальних мереж.

Як показує практика, комунікації у межах одного підприємства чи підрозділів одного підприємства набагато зручніше і безпечніше створювати, спираючись на приватну мережу, ніж на публічну. У такому випадку приватні адреси також мають бути унікальними у межах своєї локальної мережі, аналогічно до того, як публічні адреси мають бути унікальними у межах мережі Інтернет.

1.3. Операційна система

1.3.1. Поняття операційної системи

Операційна система – це комплекс керуючих і обробних програм, які, з одного боку, виступають як інтерфейс між пристроями обчислювальної системи і прикладними програмами, а з іншого боку призначені для управління пристроями, обчислювальними процесами, ефективного розподілу обчислювальних ресурсів між обчислювальними процесами і організації надійних обчислень [18, 28]. Це визначення можна застосувати до більшості сучасних операційних систем загального призначення.

Основними функціями операційної системи є:

1) дозволяє виконувати запити команд;
2) завантажує програми в оперативну пам'ять і виконує їх;
3) забезпечує призначений для користувача інтерфейс;
4) надає доступ до різних пристроїв, включаючи периферійні;
5) керує оперативною пам'яттю, а також доступом до різного виду носіїв;
6) розмежовує доступ процесів до ресурсів;

7) організовує процес взаємодії між робочими процесами;
8) забезпечує розрахований на багато користувачів режим між різними користувачами;
9) забезпечує виконання багатозадачності.

Серверні операційні системи в цілому мають таке ж призначення, що і їх десктопні версії. Серверні ОС керують додатками, що обслуговують усіх користувачів приватної або публічної мережі. Це можуть бути системи управління базами даних, засоби керування мережами, аналізу явищ у мережі, засоби обміну повідомленнями, WEB-сервери тощо. Зважаючи на високе та постійне навантаження таких операційних систем, вони мають високі вимоги до показників продуктивності та надійності. Дуже важливими аспектами є можливість дублювання та резервування, зміни програмної та апаратної частин сервера без припинення його функціонування та перезавантаження операційної системи [27].

1.3.2. Найпоширеніші сімейства серверних операційних систем

Станом на грудень 2021 року, рейтинг та поширеність серверних операційних систем за даними W3Techs виглядає наступним чином:

1) Сімейство операційних систем сімейства Linux – приблизно 77.6%;
2) Сімейство операційних систем сімейства Windows – приблизно 21.4%;
3) UNIX та інші UNIX-подібні операційні системи (не враховуючи Linux) – менше 1%.

Розглянемо найпоширеніші сімейства серверних операційних систем.

Linux – сімейство UNIX-подібних операційних систем на базі ядра Linux, що включають той чи інший набір утиліт і програм проекту GNU, і, можливо, інші компоненти. Як і ядро Linux, системи на його основі, як правило, створюються і поширюються відповідно до моделі розробки вільного та відкритого програмного забезпечення. Linux-системи поширюються в

основному безкоштовно у вигляді різних дистрибутивів – у формі, готової для установки і зручною для супроводу і оновлень, а також мають свій набір системних і прикладних компонентів [18, 28].

Можна виділити кілька основних областей, де нерідко можна зустріти Linux:

1) Сервери, що вимагають високих показників безперервної роботи;
2) Комп'ютери нестандартної архітектури (наприклад, суперкомп'ютери) через можливість швидкої адаптації ядра операційної системи і великої кількості ПО під нестандартну архітектуру;
3) Системи військового призначення з міркувань безпеки;
4) Комп'ютери, вбудовані в різні пристрої (банкомати, термінали оплати, мобільні телефони, маршрутизатори, пральні машини і навіть безпілотні військові апарати) через широких можливостей по конфігурації Linux під задачу, виконувану пристроєм, а також відсутність плати за кожен пристрій;
5) Масові спеціалізовані робочі місця (наприклад, тонкі клієнти, нетбуки) – також через відсутність плати за кожне робоче місце і через їх обмеженою обчислювальної потужності, якої може не вистачати для пропрієтарних ОС;
6) Старі комп'ютери з обмеженими ресурсами швидкодії і оперативної пам'яті, для них використовуються швидкі робочі оточення або віконні менеджери, які не вимогливі до ресурсів (наприклад, LXDE, Openbox, Xfce, Fluxbox).

Дистрибутиви Linux вже давно використовуються в якості серверних операційних систем і зайняли значну частку цього ринку; за даними компанії Netcraft на лютий 2021 року, сім з десяти найбільш надійних інтернет компаній, що надають хостинг, використовують Linux на своїх веб-серверах [27].

Нижче наведено рейтинг найпоширеніших дистрибутивів Linux станом на грудень 2021 року за даними W3Techs:

1) Ubuntu Server – 34.8%;

2) Debian – 15.5%;

3) CentOS – 9.9%;

4) Red Hat Enterprise Linux – 0.9%;

5) Gentoo – 0.6%;

6) Fedora – 0.2%;

7) Інші дистрибутиви Linux – меньш ніж 1%.

Linux є ключовим компонентом комплексу серверного програмного забезпечення LAMP (Linux, Apache, MariaDB / MySQL, Perl / PHP / Python), який придбав популярність серед розробників і став однією з найбільш поширених платформ для хостингу веб-сайтів

Windows – сімейство комерційних операційних систем корпорації Microsoft, орієнтованих на застосування графічного інтерфейсу при управлінні. Спочатку Windows була всього лише графічною надбудовою-програмою для операційної системи 80-х і 90-х років MS-DOS. Станом на грудень 2021 року, під управлінням операційних систем сімейства Windows, за даними ресурсу Net Applications, працює близько 83% персональних комп'ютерів [35].

Windows Server – лінійка серверних операційних систем. Станом на грудень 2021 року складається з наступних систем:

1) Windows Server 2003
2) Windows Server 2003 R2
3) Windows Server 2008
4) Windows Server 2008 R2
5) Windows HPC Server 2008
6) Windows Server 2012
7) Windows Server 2012 R2
8) Windows Server 2016
9) Windows Server 2019
10) Windows Server 2022

macOS Server – UNIX-подібна пропрієтарна операційна система виробництва Apple [31]. Починаючи з 2011 року, серверна редакція була

вбудована в звичайну версію тоді ще OS X [30]. Включає в себе програми адміністрування та управління робочими групами, що дають спрощений доступ до таких мережевих сервісів як поштовий сервер, SMB-сервер, DNS та іншим [33]. Останньою версією є macOS Server 5.10, яка була випущена 1 квітня 2020 року.

У macOS використовується ядро XNU, засноване на мікроядрі Mach і містить програмний код, розроблений компанією Apple, а також код з ОС NeXTSTEP і FreeBSD [32].

У macOS (як і в будь-якій UNIX-системі) використовується багатозадачність і захист пам'яті, що дозволяють запускати кілька ізольованих один від одного процесів, кожен з яких не може перервати або модифікувати всі інші [31]. На архітектуру macOS вплинула OpenStep, яка була задумана як переноситься операційна система (наприклад, NeXTSTEP була перенесена з оригінальною платформи 68k комп'ютера NeXT до придбання NeXTSTEP компанією Apple). Аналогічним чином OpenStep була перенесена на PowerPC в рамках проекту Rhapsody [36].

1.3.3. Порівняння сімейств серверних операційних систем

Для того, щоб обрати операційну систему для автоматичної установки, було проведено аналіз існуючих сімейств операційних систем. Було розглянуто сімейства серверних операційних систем Linux, Windows та macOS.

Для аналізу операційних систем сімейства Linux використовувалися Debian 9, Ubuntu Server 18.04 та CentOS 7.

Для аналізу операційних систем сімейства Windows використовувалися windows Server 2012 R2 та Windows Server 2016.

Операційні системи сімейства macOS розглядалися лише теоретично, адже неможливо отримати безкоштовну пробну версію операційних систем цього

сімейства. Виходячи з цього, щоб сформувати уявлення та практичний досвід взаємодії з цими системами, було оглянуто відповідну літературу [31, 32, 33, 34].

В ході дослідження було виявлено деякі переваги на недоліки перерахованих серверних операційних систем.

Порівняльна характеристика сімейств найпоширеніших операційних систем представлена у табл. 1.2.

Таблиця 1.2

Порівняльна характеристика сімейств операційних систем

Критерії оцінювання	Linux	Windows	macOS
Доступність та популярність	Абсолютно безкоштовна, є частиною проекту GNU. Для установки дистрибутивів достатньо завантажити його.	Платна операційна система, має велику популярність, доступність цієї ОС висока.	Платна операційна система, включена у вартість ноутбуків та ПК від Apple.
Інтерфейс користувача	Інтерфейс представлений у вигляді командної строки, графічний інтерфейс відсутній чи майже зведений до нуля.	Інтерфейс серверних версій операційної системи схожий із версією для ПК. Наявний графічний інтерфейс.	У серверній версії також наявний графічний інтерфейс. Візуально схожий із Windows.
Установка та налаштування	Установка займає від 20 хвилин до 2 годин, під час процесу можна налаштувати систему під певного користувача.	Установка займає приблизно 1 годину. Налаштовується автоматично ручному режимі	Встановлюється та оновлюється автоматично. На пристроях від компанії Apple встановлена.
Сумісність з обладнанням	Сумісна з середньою кількістю пристроїв, але список постійно поповнюється.	Сумісна з більшістю пристроїв. Є всі драйвера для всіх пристроїв.	Сумісна лише із сімейством Intel та своєю спеціалізованою периферією.
Набір інстальованих програм	Великий набір вбудованих програм для роботи з різними типами файлів.	Мінімальний набір вбудованих програм. Автоматично встановлюється найнеобхідніше.	Мінімальний набір вбудованих програм. Автоматично встановлюється найнеобхідніше.
Захист від шкідливого ПО та вірусів	Засоби захисту передбачені в ядрі ОС.	Частково захищена від вірусів, потрібна додаткова покупка і установка антивірусів.	Засоби захисту передбачені в ядрі операційної системи.

1.4. Висновок до першого розділу та постановка задач дослідження

У першому розділі було розглянуто базові поняття сервера та серверного обладнання, протоколи мережевої взаємодії. Також було розглянуто ключові налаштування і параметри серверів, варіанти впровадження системних налаштувань апаратної і програмної частин. Було проведено аналіз існуючих операційних систем, наведено порівняльну характеристику з перевагами та недоліками сучасних серверних ОС.

Виходячи із визначених переваг і недоліків, було вирішено дослідити систему автоматичного розгортання виділеного фізичного сервера із установкою операційної системи Ubuntu Server 18.04, яка є найпопулярнішим з дистрибутивів Linux.

Головною характеристикою роботи алгоритму автоматичного розгортання сервера є зведення помилок до мінімальної кількості, швидкість розгортання сервера, варіативність системних налаштувань, перевірка коректності фізичних підключень сервера, організація відмовостійкої моделі за рахунок агрегації фізичних каналів зв’язку у логічний з метою збільшення пропускної спроможності та збільшення надійності.

Доречно розділити виконання завдання на декілька основних етапів: виділення публічної та приватної адреси з наявних пулів адрес, перевірку коректності підключень, очистку дисків у разі необхідності, встановлення операційної системи та перевірку доступності сервера по публічній та приватній мережам. Кожен з цих етапів також складається з одного чи деяких кроків.

Виходячи з вищевказаного, можемо сформулювати мету та завдання роботи.

Метою роботи є підвищення ефективності розгортання фізичних серверів за допомогою автоматизації рутинних процесів мережевої взаємодії.

Для досягнення цієї мети поставлено такі завдання:

1) Проаналізувати можливість автоматичної установки ОС Ubuntu Server 18.04;
2) Дослідити алгоритм автоматичної установки ОС;
3) Дослідити програмне забезпечення, робота якого базується на алгоритмі;
4) Протестувати алгоритм на реальному фізичному сервері;
5) Впевнитися, що операційна система встановлена і налаштована коректно і готова до функціонування.

РОЗДІЛ 2
АЛГОРИТМ АВТОМАТИЧНОГО РОЗГОРТАННЯ ВИДІЛЕНОГО СЕРВЕРА

2.1. Загальне рішення задачі

Задачу автоматичного розгортання виділеного сервера пропонується виконати за допомогою виконання скриптів, які дозволяють уникнути рутинної роботи із установкою операційної системи.

Для запуску скрипта буде використано фізичний виділений сервер, який не має встановленої ОС, командної строки і фактично є недоступним для використання. Сервер є недоступним, адже його загрузчик намагатиметься запустити операційну систему, яка встановлена на його жорсткий диск, але такі спроби будуть неминуче завершуватися невдачею. Таким чином, сервер буде знаходитися у вічних спробах запуску ОС.

Маємо сервер моделі Dell PowerEdge R440. На рис. 2.1 зображений цей сервер.

Рис. 2.1. Сервер моделі Dell PowerEdge R440

Стислі характеристики цієї моделі сервера наведено у табл. 2.1.

Таблиця 2.1

Характеристики сервера Dell PowerEdge R440, використаного для даної роботи

Показник	Характеристика
Процесори	2 x Intel Xeon Silver 4114
Оперативна пам’ять	32 GB (2 x 16 GB DDR4 RDIMM)
Накопичувачі	4 x 480 GB SSD SATA (2 x Kingston, 2 x EDGE)
Модель RAID-контролера	Perc H330

Фізично сервер може знаходитися де завгодно. У нашому випадку сервер знаходиться у дата-центрі у місті Форт-Уорт, що знаходиться у 20 км від Далласу, штат Техас, США. Звісно, що ми не маємо фізичного доступу до сервера, тобто ми не можемо під’єднатися до сервера напряму, використовуючи свій ноутбук, не можемо використовувати компакт-диск чи USB-накопичувач, щоб встановити ОС на цей сервер. Проте ми знаємо, що сервер має вбудовану мережеву карту, що має два порти зі швидкістю каналів до 1 Гбіт/с. Окрім вбудованої карти, сервер має ще дві зовнішніх мережевих карти моделі Intel X550-t Adapter, кожна з цих карт має по два порти, швидкість яких складає до 10 Гбіт/с [19].

Окрім використання локальних носіїв даних (жорсткого диска, USB-накопичувача тощо) існує можливість загрузки комп’ютера за допомоги спеціальної середи, що називається PXE. PXE-код, який зазвичай зберігається в ПЗУ мережевої карти, отримує за протоколом BOOTP IP-адресу сервера, на якому знаходиться виконуваний файл, отримує цей файл по мережі за допомоги протоколу TFTP. Після цього управління передається цьому файлу.

2.2. Схема та опис алгоритму

Алгоритм автоматичного розгортання сервера може бути відображений за допомогою схеми, зображеній на рис. 2.2.

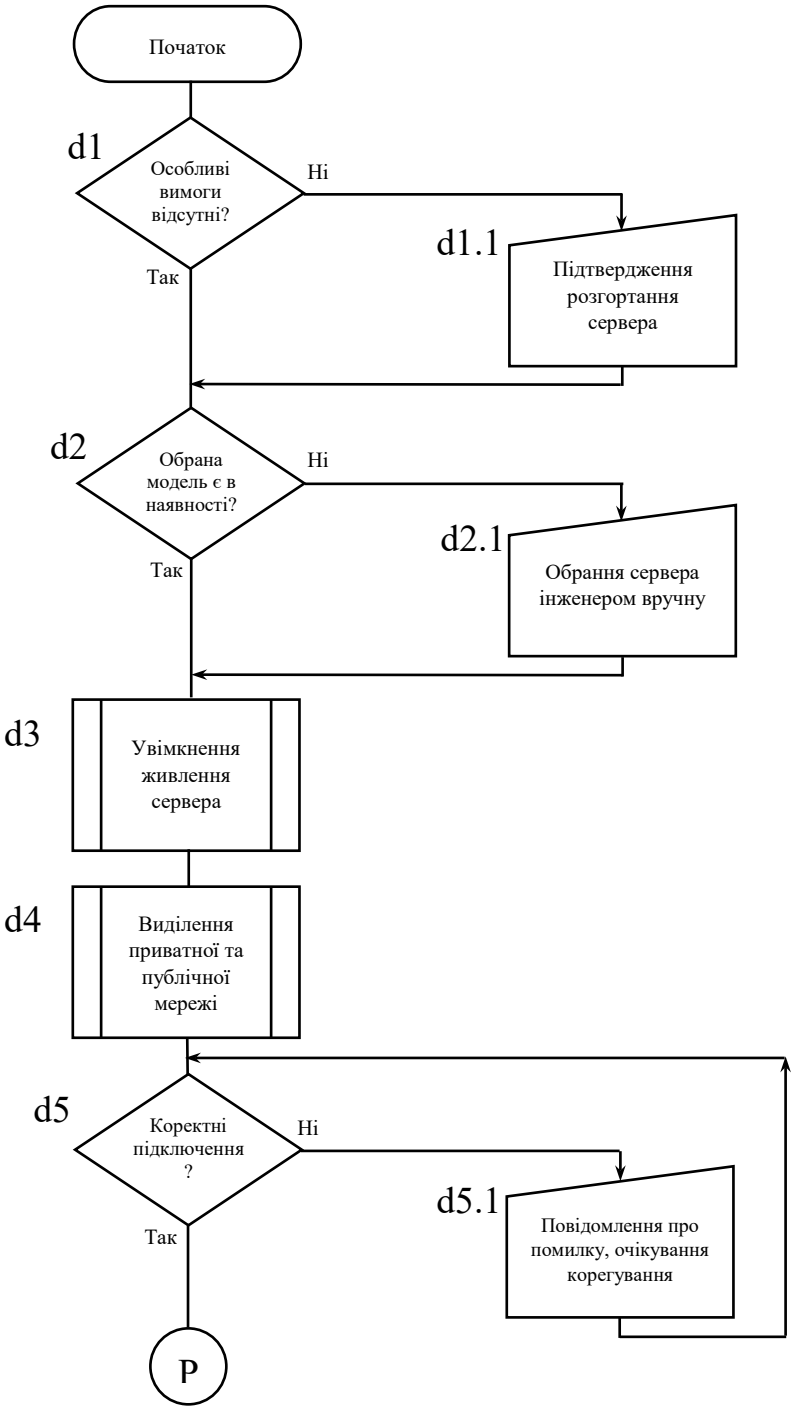
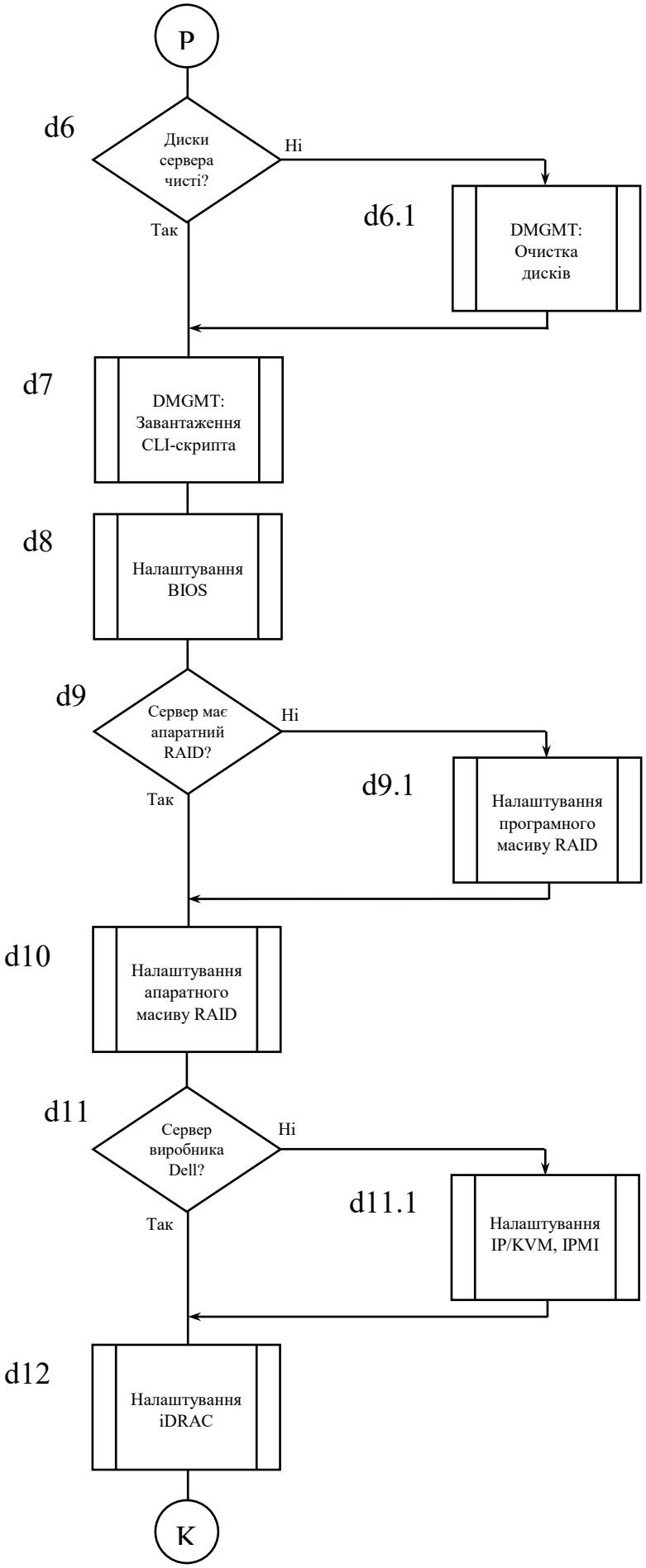
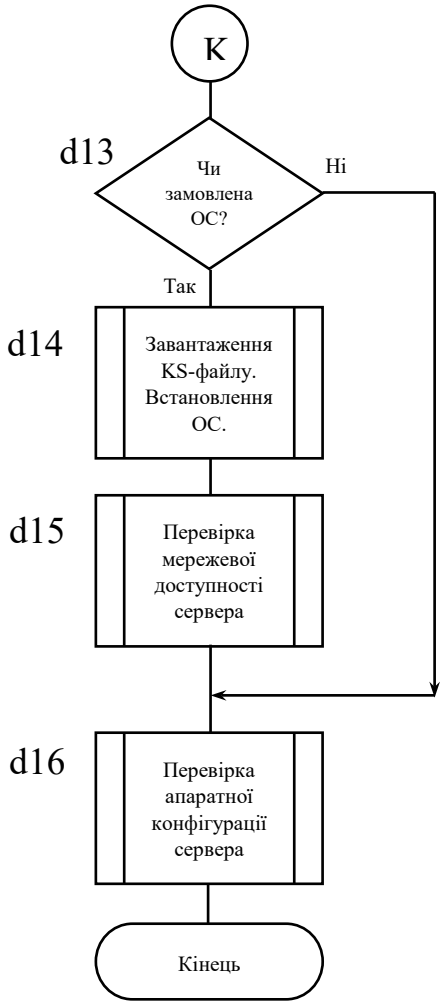


Рис. 2.2. Процес автоматичного розгортання виділеного сервера


P
d6
Диски сервера чисті?
Ні
Так
d6.1
DMGMT: Очистка дисків
d7
DMGMT: Завантаження CLI-скрипта
d8
Налаштування BIOS
d9
Сервер має апаратний RAID?
Ні
Так
d9.1
Налаштування програмного масиву RAID
d10
Налаштування апаратного масиву RAID
d11
Сервер виробника Dell?
Ні
Так
d11.1
Налаштування IP/KVM, IPMI
d12
Налаштування iDRAC
K



Розглянемо більш детально деякі кроки цього алгоритму.

Блок d1 перевіряє, чи є особливі вимоги до сервера, що видається. Це можуть бути диски якоїсь певної моделі, наявність чи відсутність апаратного RAID-контролера. Кожний сервер перевіряється на наявність особливих вимог до нього, тому на цьому кроці необхідно особисто перевірити вимоги до сервера. Технічно автоматизувати цей крок дуже легко, проте практика показує, що це досить ризиковано, адже вимоги до сервера можуть бути дуже різноманітними і нема впевненості, що автоматична система зможе правильно реагувати на всі види вимог. Таким чином, алгоритм передбачає очікування, доки повноважна особа підтвердить, що вимоги до сервера відсутні або виконані (блок d1.1).

Блок d2 перевіряє резерв усіх серверів, що маються у підприємстві і успішно проходить у разі, якщо сервер замовленої конфігурації дійсно є в

наявності і його конфігурація співпадає із замовленою. Якщо такого сервера немає в наявності, то інженер, відповідальний за функціонування системи, шукає альтернативні варіанти, а в разі їх відсутності констатує, що зараз сервер замовленої конфігурації не може бути підготовлений (блок d2.1).

Блок d3 виконує увімкнення сервера шляхом подання на нього електричного струму. Для заощадження грошових коштів усі сервери з резерву перебувають у вимкненому стані.

У блоці d4 відбувається комплекс дій: шукаються усі групи комутаторів, до яких має бути підключений сервер, виділяється та налаштовується тимчасова мережа для наступної перевірки коректності підключень. Виділення публічної та приватної мереж для сервера.

Блок d5 є одним з найголовніших кроків на шляху до автоматичного розгортання сервера. На цьому етапі сервер завантажується через PXE до спеціального LLDP-образу. Цей образ має лише одну функцію: за протоколом канального рівня визначити, за допомогою яких портів яких мережевих карт сервер підключений до комутаторів в локальній мережі. Цей протокол не залежить від виробників мережевого обладнання. Дуже важливо те, що цей етап проводить перевірку не тільки на те, чи підключені мережеві карти до комутаторів, а й те, якими саме портами. Ця перевірка відбувається із винайденням MAC-адрес усіх портів мережевих карт. Якщо схема підключень відрізняється від стандартної, це може призвести до тяжких наслідків, тому у блоці d5.1 автоматична система повідомляє інженера, що підключення переплутані, відсутні або некоректні.

Для виконання блоку d6 сервер перезавантажується і знову грузиться по PXE, але вже у спеціальний DMGMT образ на базі ОС CentOS 7. Також формується CLI-скрипт для запуску очистки дисків, якщо вона потрібна. Очистка дисків необхідна, якщо в сервері встановлені не нові диски, які можуть мати дані останнього користувача сервером. Це дуже небезпечно, тому якщо в сервері стоять старі диски, то всі вони налаштовуються в RAID 0 та проходять очистку паралельно за допомогою утиліти shred. Очистка проходить у блоці d6.1.

У блоці d7 сервер знову перезавантажується і грузиться в PXE-меню, знову завантажує DMGMT-образ і формує CLI-скрипт для налаштувань сервера. До цих налаштувань входять налаштування BIOS (блок d8) формування апаратного RAID-масиву, якщо в сервері встановлено апаратний контролер (блоки d9 та d9.1), або налаштування програмного RAID-масиву, якщо в сервері не встановлено апаратного контролера (блок d10). Також на блоці d12 виконуються налаштування iDRAC (тільки для серверів Dell) або IPMI (для інших виробників) чи VNC (блок d11.1).

У блоці d13 перевіряється, чи було замовлено автоматичну установку операційної системи. Якщо операційна система замовлена, то у блоці d14 формується спеціальний kickstart-файл, котрий перелаштовує DMGMT-образ саме на образ тієї системи, яку треба встановити. Цей файл містить усі необхідні параметри для налаштування системи, які обираються при заказі (ім’я виділеного сервера, параметри розмітки дисків, SSH-ключ тощо).

Після установки операційної системи сервер знову перезавантажується і на цей раз грузиться в операційну систему. Доходячи до екрану вводу логіна та пароля, проводиться перевірка сервера за протоколом ICMP в межах приватної та публічної мереж (блок d15).

В останньому блоці d16 перевіряється апаратна частина сервера, її збіг із інформацією, вказаній про сервер у базі даних.

2.3. Попередні налаштування

Для того, щоб автоматична система мала можливість виконати перевірку підключень, а потім загрузити сервер у DMGMT-образ, на сервері необхідно зробити деякі налаштування. Ігнорування цих налаштувань можуть призвести до помилок на різних стадіях розгортання сервера автоматикою.

На мережевих картах (принаймні на першому порті мережевої карти в PCI-слоті №1) має бути увімкнена функція завантаження по PXE. Для цього треба

виконати наступне налаштування у BIOS: Device Settings → NIC1 → NIC Configuration → Legacy Boot Protocol = PXE. На рис. 2.3 наведено скріншот налаштування у BIOS завантаження по PXE для першого порту зовнішньої мережевої карти в PCI-слоті №1.

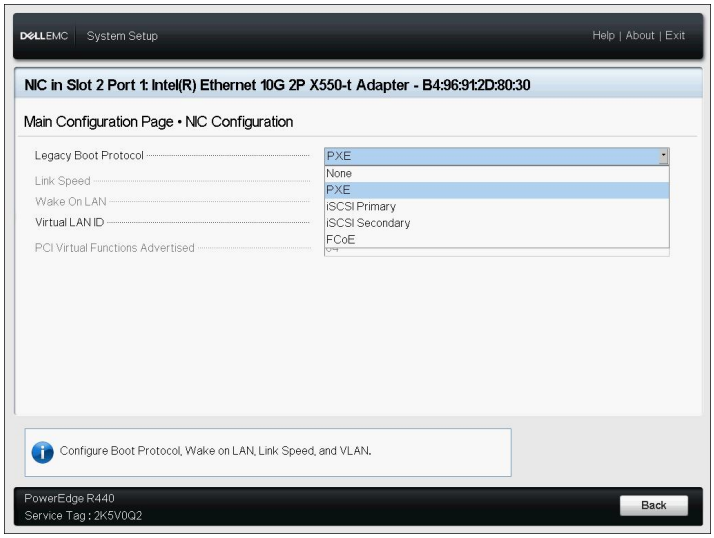


Рис. 2.3. Налаштування загрузки по PXE

Якщо, як у нашому випадку, сервер фізично знаходиться у віддаленому дата-центрі, то також бажано налаштувати параметр AC Power Recovery. Цей параметр визначає, чи буде умикатися сервер при поданні на нього живлення електричним струмом. Включення цього параметра є дуже зручним, адже віддати команду на увімкнення порту PDU набагато швидше, аніж зв'язуватися із працівниками датацентру, просити їх знайти потрібний сервер серед сотень стійок та натиснути кнопку увімкнення.

Для цього треба виконати наступне налаштування у BIOS: System BIOS → System Security → AC Power Recovery = ON.

На рис. 2.4 наведено скріншот налаштування цього параметра у BIOS.

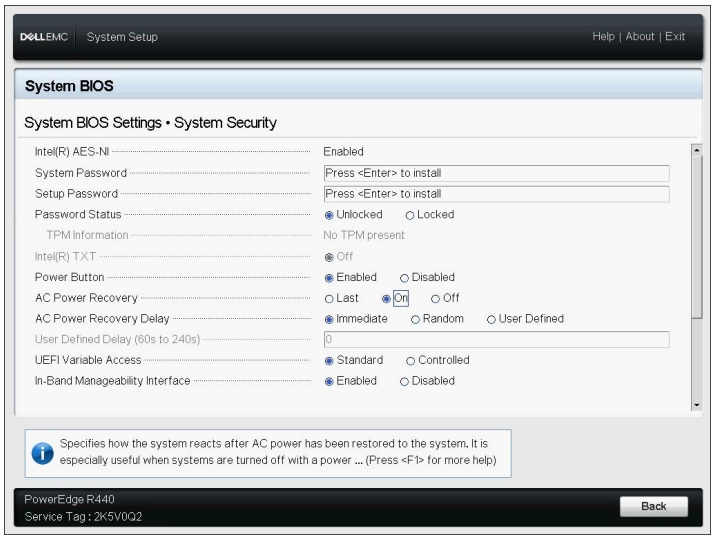


Рис. 2.4. Налаштування у BIOS параметра AC Power Recovery

Також для завантаження сервера в PXE треба переключити режим роботи UEFI BIOS в режим сумісності Legacy BIOS. Для цього треба виконати наступне налаштування у BIOS: System BIOS → Boot Settings → Boot Mode = BIOS. На рис. 2.5 наведено скріншот переключення режиму роботи UEFI BIOS.

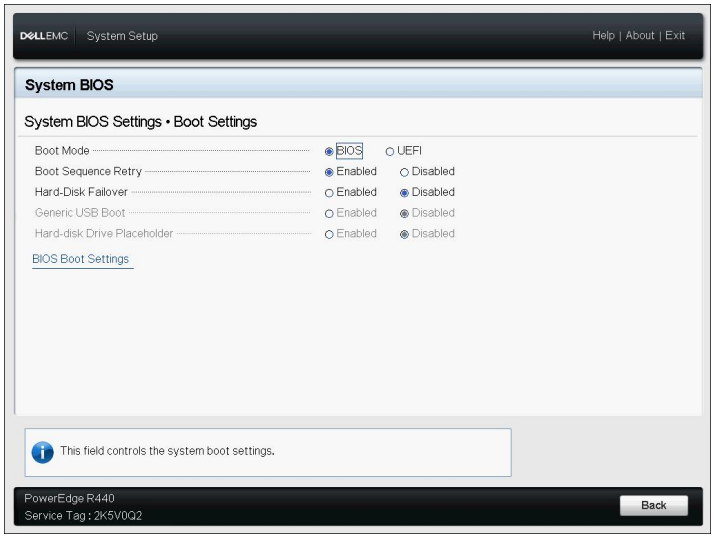


Рис. 2.5 Переключення режиму роботи UEFI BIOS.

Для заощадження часу на проходження алгоритму доречно також встановити системний порядок загрузки таким чином, щоб в першим у черзі загрузки стояв визначений порт, який обрано для загрузки сервера у PXE-меню.

Наступним має стояти жорсткий диск, на який буде встановлено операційну систему. Таким чином, коли процес автоматичної установки операційної системи на сервер буде завершено, одразу після спроби загрузитися по PXE, загрузчик сервера намагатиметься загрузитися із жорсткого диска. На рис. 2.6 наведено скріншот найкращого попереднього налаштування черги загрузки.

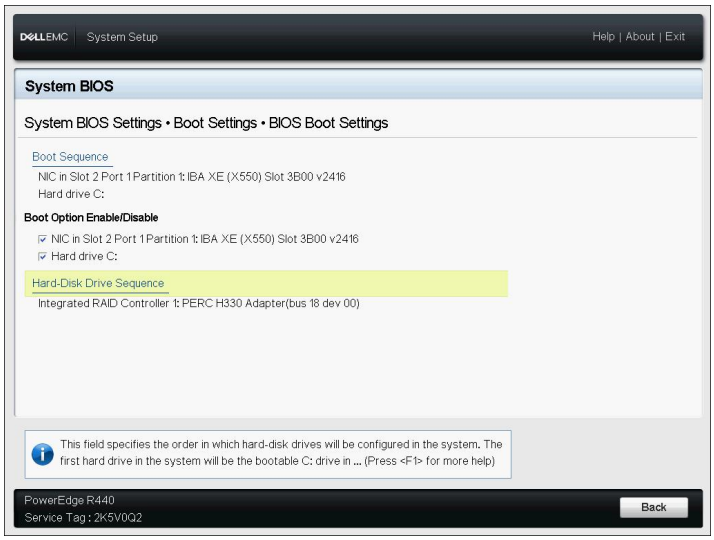


Рис. 2.6. Скріншот попереднього налаштування порядку загрузки сервера

З огляду на заощадження системних ресурсів, можливість загрузки по PXE була деактивована на залишку портів вбудованих та зовнішніх мережевих карт. Це означає, що такі порти більше не будуть показуватися у порядку загрузки.

2.4. Автоматична установка операційної системи

Для автоматичної установки обраної операційної системи, а саме Ubuntu Server 18.04, будемо використовувати DMGMT-образ, який розміщується на іншому робочому сервері. Під час загрузки сервера по PXE, ми потраплятимемо у спеціальне PXE-меню, де можна побачити усі варіанти операційних систем для установки [18].

DMGMT – це спеціальний образ на базі CentOS 7, котрий вміщує деякі системні утиліти з пакета Dell OpenManage Deployment Toolkit. Ці утиліти використовуються для налаштування компонентів серверів.

Під час завантаження DMGMT-образа також завантажується підготовлений раніше CLI-скрипт, ім’я котрого для надійності складається з MAC-адреси першого порту мережевої карти в PCI слоті №1.

CLI-скрипт є ключовим елементом у роботі DMGMT-образа при виконанні автоматичних задач розгортання фізичного виділеного сервера. Цей скрипт зберігається у файлі srvinit.cli. Після завантаження DMGMT-образа він завантажується в каталог /tmp/ і запускається автоматично. Автоматичний запуск скрипта прописано у /etc/rc.3/99SRVINIT.

Скрипт використовує деякі утіліти з пакета Dell OpenManage:

1) syscfg – утиліта для налаштування BIOS сервера;
2) racadm – утиліта для налаштування iDRAC сервера;
3) raidcfg – утиліта для налаштування RAID-контролерів моделей Perc H330, H730, H740P (як виняток, для контролерів H200 використовується утиліта sis2ircu);
4) shred – утиліта для знищення даних на носіях інформації;
5) dsu – утиліта для оновлення прошивок апаратної частини серверів Dell з офіційного репозиторію.

Для автоматичної установки операційних систем сімейства Linux використовується так званий KS або kickstart-файл. Також цей файл іноді називають файлом відповідей.

Роль KS-файлів під час автоматичної установки операційної системи зводиться до налаштування наступних параметрів операційної системи:

1. Якщо сервер не має апаратного RAID-контролера, виконується налаштування програмного RAID-масиву (mdadm у Linux).
2. Виконується розмітка файлової системи (кількість, розмір та тип розділів файлової системи) виходячи з вказаних параметрів під час замовлення сервера.

3. Перейменування інтерфейсів у файлі /etc/udev/rules.d70-persistent-net-rules за їх MAC-адресами із стандартних eth0, en0, p1p1 до уніфікованих імен int1, int2, ext1,ext2. Перейменування інтерфейсів виконується виходячи з MAC-адрес, вказаних у БД серверів за схемою: NIC1 – int1; NIC2 – ext1; NIC3 – int2; NIC4 – ext2.
4. Налаштовується агрегація каналів з боку сервера: основний та резервний інтерфейси приватної мережі об’єднуються у віртуальний інтерфейс aggi (int1 + int2), у той же час основний та резервний інтерфейси публічної мережі об’єднуться у віртуальний інтерфейс agge (ext1 + ext2).
5. Налаштовуються виділені публічні та приватні мережі. За потреби також налаштовуються аліасні (додаткові) адреси.
6. Налаштовуються статичні маршрути для доступу до внутрішніх сервісів: у мережу 10.0.0.0/8 для роботи приватної мережі між фізичними виділеними серверами; у мережу 192.168.0.0/16 для доступу до внутрішніх серверів компанії, наприклад, для користування NTP або DNS, дзеркал дистрибутивів Linux тощо.
7. Виконуються додаткові налаштування: DNS у файлі /etc/resolv.conf; синхронізація часу по NTP на внутрішні сервери; встановлюється root-пароль, який генерується автоматично із випадкових символів і має довжину 10 символів; якщо потрібно, то додається SSH-ключ для доступу до сервера по SSH; налаштування імені хоста.

2.5. Висновок до другого розділу

У другому розділі кваліфікаційної роботи освітнього рівня магістра було описано алгоритм автоматичного розгортання фізичного виділеного сервера.

Було виконане загальне рішення задачі, створено схему та опис алгоритму, розглянуто важливі попередні налаштування, до яких має бути приведений сервер перед процедурою автоматичного розгортання.

Також були описані можливості DMGMT-образів та CLI-скриптів, їх роль у автоматичному розгортанні серверів. Також було розглянуто застосування KS-файлів, було оглянуто галузі їх застосування та налаштування, для яких вони використовуються для автоматичної установки операційної системи.

РОЗДІЛ 3
ДОСЛДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Загальні відомості

Досліджуємий скрипт, що використовується алгоритмом автоматичного розгортання фізичного виділеного сервера, – CLI-скрипт був створения за допомогою мови програмуваня Ruby. [12].

Ruby – це динамічна та рефлективна об’єктно-орієнтована мова програмування. Великою перевагою Ruby є кросплатформеність, а також те, що ця мова програмування є вільною для використання [13]

Мова програмування Ruby була створена японським розробником вільного ПЗ Юкихіро Мацумото, перша версія мови була випущена у 1995 році [15].

Прийнято виділяти наступні можливості мови програмування Ruby:

1) Лаконічний та простий синтаксис;
2) Обробка виключень у стилі Java та Python [14];
3) Можливість перевизначення операторів, що є методами;
4) Є повність об’єктно-орієнтованою мовою програмування;
5) Включає в себе автоматичний збірник сміття;
6) Автоматична конвертація типів цілих перемінних;
7) Реалізація шаблонів проектування;
8) Можливість перенесення на безліч платформ.

3.2. Опис системи

3.2.1. Функціональні вимоги до досліджуємої системи

Система, що досліджується, має чітко поставлені функціональні вимоги [16]. Основними вимогами є наступні:

1) Перевіряти наявність додаткових вимог до програмної чи апаратної конфігурації сервера;
2) Перевіряти наявність серверів потрібної конфігурації серед доступних;
3) Перевіряти коректність мережевих підключень сервера та сигналізувати про помилку у разі її знаходження;
4) Налаштовувати публічну та приватну мережі, аналогічно виділяти публічну та приватну адресу для сервера;
5) Чистити диски у разі потреби;
6) Завантажувати CLI-скрипт і виконувати ключові налаштування: BIOS, iDRAC та RAID;
7) Встановлювати операційну систему;
8) Перевіряти апаратну частину на її збіг із даними у БД.

3.2.2. Вхідні та вихідні дані системи

Не дивлячись на те, що система автоматичного розгортання сервера є доволі складною і її коректна робота залежить від багатьох факторів, вона має небагато вхідних даних. Майже усі вхідні дані обираються зі списку кінцевим користувачем перед замовленням сервера. Цими даними є:

1) Модель сервера, конфігурація дисків (HDD, SSD, SAS, їх форм-фактор, кількість та об’єм);
2) Кількість та об’єм модулів оперативної пам’яті;
3) Наявність апаратного RAID-контролера;
4) Операційна система;
5) Кількість, ім’я та об’єм розділів;
6) Ім’я системи (хостнейм);

Аналогічно, вихідними даними є лише пароль root операційної системи, публічна та приватна адреса, виділена для сервера. Мається на увазі, що якщо замовник отримав root-пароль від сервера, то цей сервер задовольняє усім вимогам, які були описані замовником під час замовлення.

3.2.3. Графічний інтерфейс користувача

Система автоматичного розгортання фізичних виділених серверів не має особливого графічного інтерфейсу. Як показує практика, дії по виділенню публічної та приватної адреси, а також виконання ключових налаштувань на сервері під час відпрацювання CLI-скрипта (налаштування iDRAC, BIOS та RAID) не потребують графічного інтерфейса, адже виконуються дуже швидко та автоматично.

Існує можливість перевірки процесу роботи автоматичної системи за допомогою підключення до віртуальної консолі сервера. У цьому випадку можливо слідкувати за проходженням розгортання сервера в режимі реального часу.

3.3. Результати проведених досліджень

Після дослідження алгоритму системи, та CLI-скрипта (див. Додаток А), для впевненості необхідно провести дослідження роботи системи автоматичного розгортання серверів.

3.3.1. Перевірка коректності мережевих підключень за допомогою LLDP-образу

LLDP – протокол канального рівня, який дозволяє мережевому обладнанню сповіщати обладнання, що працює у локальній мережі, про свою наявність та передавати йому свої характеристики, а також отримувати від цього обладнання аналогічні відомості.

Інформація, що збирається за допомогою протоколу LLDP, окрім іншої інформації зазвичай включає:

1) Ім’я обладнання та його опис;

2) Ім’я порта та його опис;
3) Ім’я VLAN;
4) IP-адресу обладнання, якому сервер доступний для управління по протоколу SNMP;
5) Функції обладнання (комутація, маршрутизація тощо);
6) Параметри об’єднання каналів;
7) MAC-адресу.

LLDP-образ є підвидом DMGMT-образу. Головною відміною використовуваного LLDP-образу від DMGMT-образу, є те, що LLDP-образ має лише одну функцію: за однойменним протоколом канального рівня визначити, за допомогою яких портів яких мережевих карт сервер підключений до комутаторів в локальній мережі. Цей протокол не залежить від виробників мережевого обладнання. Дуже важливо те, що цей етап проводить перевірку не тільки на те, чи підключені мережеві карти до комутаторів, а й те, якими саме портами. Якщо схема підключень відрізняється від стандартної, це може призвести до тяжких наслідків, тому автоматична система повідомляє інженера в разі необхідності, що підключення переплутані, відсутні або некоректні.

Ця перевірка дозволяє бути впевненим у тому, яким чином і до яких саме портів мережевого обладнання (маються на увазі мережеві комутатори) підключено сервер. Одночасно з цим з’являється додаткова відмовостійкість, адже невірно зазначені порти на мережевих комутаторах можуть перетворити звичайні регламентні роботи, такі як заміна патчкорду або чистка оптоволоконних кабелів, на серйозну аварію лише за рахунок неправильних відомостей про підключення портів.

На рис. 3.1 приведено скріншот віртуальної консолі сервера у момент його загрузки у LLDP-образ.

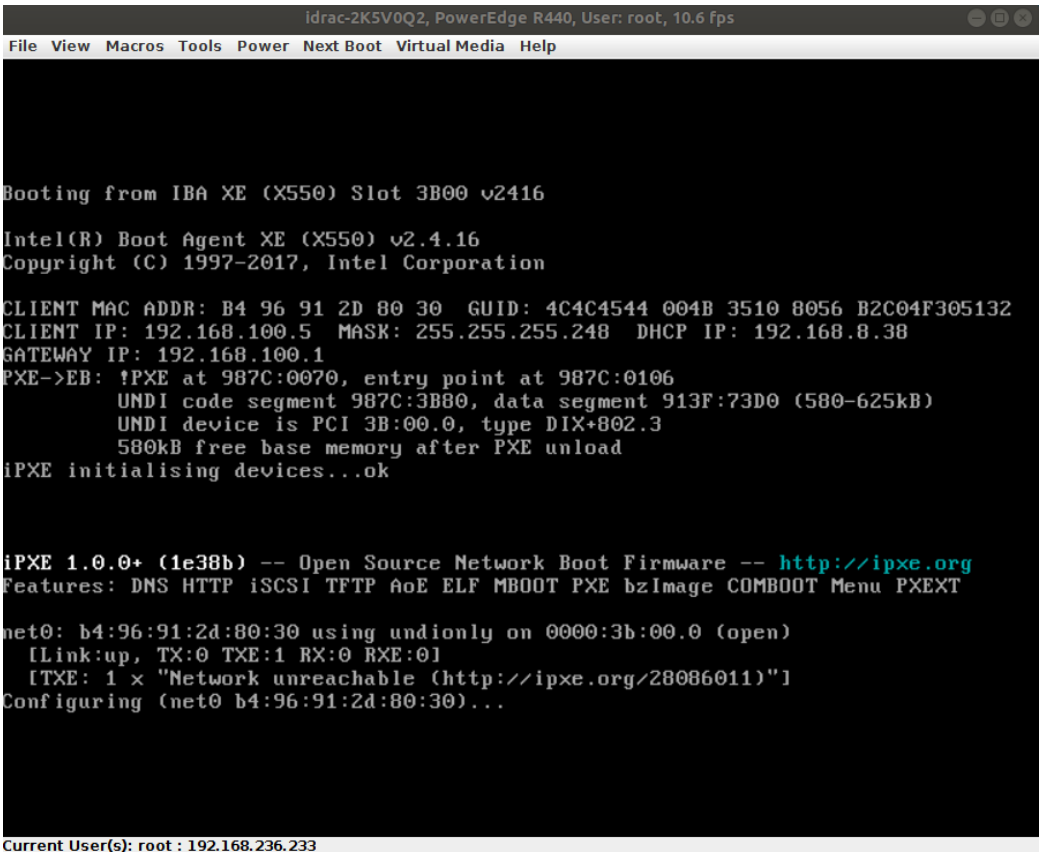


Рис. 3.1. Загрузка сервера у LLDP-образ

3.3.2. CLI-скрипт та виконання ключових налаштувань

Під час виконання CLI-скрипта, сервер іншого, виконується налаштування ключових параметрів сервера. Повний лістинг CLI-скрипта наведено у додатку А.

Ключовими налаштуваннями сервера є iDRAC, BIOS та RAID. Розглянемо налаштування BIOS більш детально:

--acpower=on – активація автоматичного вмикання сервера при подачі живлення електричним струмом на PDU;

--memtest=disablef – вимкнення перевірки модулів оперативної пам’яті на наявність помилок при кожному запуску системи з метою пришвидшити загрузку сервера;

--rptkeyerr=disable – вимкнення повідомлень про помилки клавіатури;

--numlock=off – деактивація клавіши Num Lock при увімкненні сервера;

--memoperatingmode=optimizermode – увімкнення оптимізованого режима роботи модулів оперативної пам’яті, що дозволяє нормально працювати з сервером при будь-якому обсязі та конфігурації модулів;

--virtualization=enabled – активація підтримки апаратної віртуалізації;

--bootseqretry=enable – увімкнення повторних спроб завантаження сервера, використовуючи вказані у меню Boot Device (див. пункт 1.2.1).

Розглянемо детально основні налаштування iDRAC:

set idrac.users.2.password – задається root-пароль для iDRAC;

set idrac.snmp.agentenable Disabled – вимкнення SNMP;

set idrac.vncserver.enable Enabled – активація VNC-сервера, інтегрованого в iDRAC;

set idrac.vncserver.password – задається пароль VNC-сервера;

set idrac.virtualmedia.attached Detached – деактивація Virtual Media;

system.power.RedundancyPolicy 1 – увімкнення відмовостійкої політики щодо блоків живлення;

set System.Power.HotSpare.Enable 0 – вимкнення резервних блоків живлення;

setniccfg -s – налаштовується мережевий інтерфейс iDRAC.

Розглянемо детально основні налаштування RAID:

run [RAID_UTIL, "-ctrl", "-c=#{ctrl_id}", "-ac=fgnclr"], :ignore_exceptions => true – очистка старої конфігурації RAID (еквівалентно пункту “Cleaf Foreign Config” у меню RAID-контролера);

run [RAID_UTIL, "-c=#{ctrl_id}", "-vd", "-ac=dvd"], :ignore_exceptions => true – видалення існуючих віртуальних дисків;

run [RAID_UTIL, "-ctrl", "-ac=cvd", "-c=#{ctrl_id}", "-ad=#{disks}", "-sz=#{cutsize}", "-r=#{d['raid']}"] – створення нового віртуального диску з замовленою конфігурацією;

Слід зазначити ще деякі умовні позначення команд:

RAID_UTIL – путь до утиліти raidcfg

-c=#ctrl_id – ID RAID-контролера controller id/PCI slot id;

-ac=fgnclr – команда очистки Foreign Config;

-ac=dvd – команда видалення віртуальних дисків (Delete Virtual Disk);

-ac=cvd – команда створення віртуальних дисків (Create Virtual Disk);

-ad=#{disks} – список дисків, з яких буде зібрано масив;

-sz=#{cutsize} – обсяг віртуального диска;

-r=#{d['raid']} – рівень масива.

3.3.3. Автоматична установка операційної системи

За автоматичну установку операційної системи, окрім CLI-скрипта, відповідає інший скрипт: KS або kickstart-файл (див. розділ 2.4.).

Основними функціями KS-файла є налаштування програмного RAID у разі відсутності апаратного, перейменування інтерфейсів та об’єднання фізичних інтерфейсі у логічні.

Розглянемо більш детально процес перейменування та об’єднання інтерфейсів:

d-i preseed/late_command string sh -c '(\

echo "bonding" >> /target/etc/modules; \

rm -rf /target/etc/udev/rules.d/70-persistent-net.rules; \ – відкриття файла

echo -e " \

SUBSYSTEM==\"net\",ACTION==\"add\",DRIVERS==\"?*\",ATTR{address}==\"20:47:47:8c:c8:5e\",ATTR{type}==\"1\",KERNEL==\"eth*\",NAME=\"int1\"\n\ – перейменування інтерфейса

SUBSYSTEM==\"net\",ACTION==\"add\",DRIVERS==\"?*\",ATTR{address}==\"a0:36:9f:83:93:ec\",ATTR{type}==\"1\",KERNEL==\"eth*\",NAME=\"int2\"\n\ – перейменування інтерфейса

SUBSYSTEM==\"net\",ACTION==\"add\",DRIVERS==\"?*\",ATTR{address}==\"20:47:47:8c:c8:60\",ATTR{type}==\"1\",KERNEL==\"eth*\",NAME=\"ext1\"\n\ – перейменування інтерфейса

```
SUBSYSTEM==\"net\",ACTION==\"add\",DRIVERS==\"?*\",ATTR{addres
s}==\"a0:36:9f:83:93:ed\",ATTR{type}==\"1\",KERNEL==\"eth*\",NAME=\"ext2\"
\n\ – перейменування інтерфейса
" > /target/etc/udev/rules.d/70-persistent-net.rules; \
echo -e "\
auto aggi\n\ – організація агрегації приватних інтерфейсів
iface aggi inet static \n\
   hwaddress ether 20:47:47:8C:C8:5E\n\ – MAC-адреса основного
інтерфейса приватної мережі
   address 10.107.4.172\n\ – IP-адреса
   netmask 255.255.255.248\n\ – маска

   mtu 1500\n\ – розмір MTU (Maximal Transition Unit)
   slaves int1 int2\n\ – вказання імен фізичних інтерфейсів
   bond_mode 4\n\
   bond_miimon 100\n\
   bond_downdelay 200\n\
   bond_updelay 200\n\
   bond_lacp_rate slow\n\
   bond_xmit_hash_policy layer3+4\n\
   post-up /sbin/ethtool -K aggi tx off tso off\n\
\n\
auto agge\n\ організація агрегації публічних інтерфейсів
iface agge inet static \n\
   hwaddress ether 20:47:47:8C:C8:60\n\
   address 23.105.234.36\n\
   netmask 255.255.255.248\n\
   gateway 23.105.234.35\n\
   slaves ext1 ext2\n\
   bond_mode 4\n\
```

```
    bond_miimon 100\n\
    bond_downdelay 200\n\
    bond_updelay 200\n\
    bond_lacp_rate slow\n\
    bond_xmit_hash_policy layer3+4\n\
    post-up /sbin/ethtool -K agge tx off tso off\n\
\n\
```

Після загрузки сервера у DMGMT-образ для установки ОС і формування KS-файла починається процес автоматичної установки ОС. Таким чином користувачу не потрібно налаштовувати систему вручну, за рахунок чого суттєво заощаджуються ресурси підприємства.

На рис. 3.2 наведено скріншот віртуальної консолі сервера у момент початку автоматичної установки ОС Ubuntu Server 18.04.

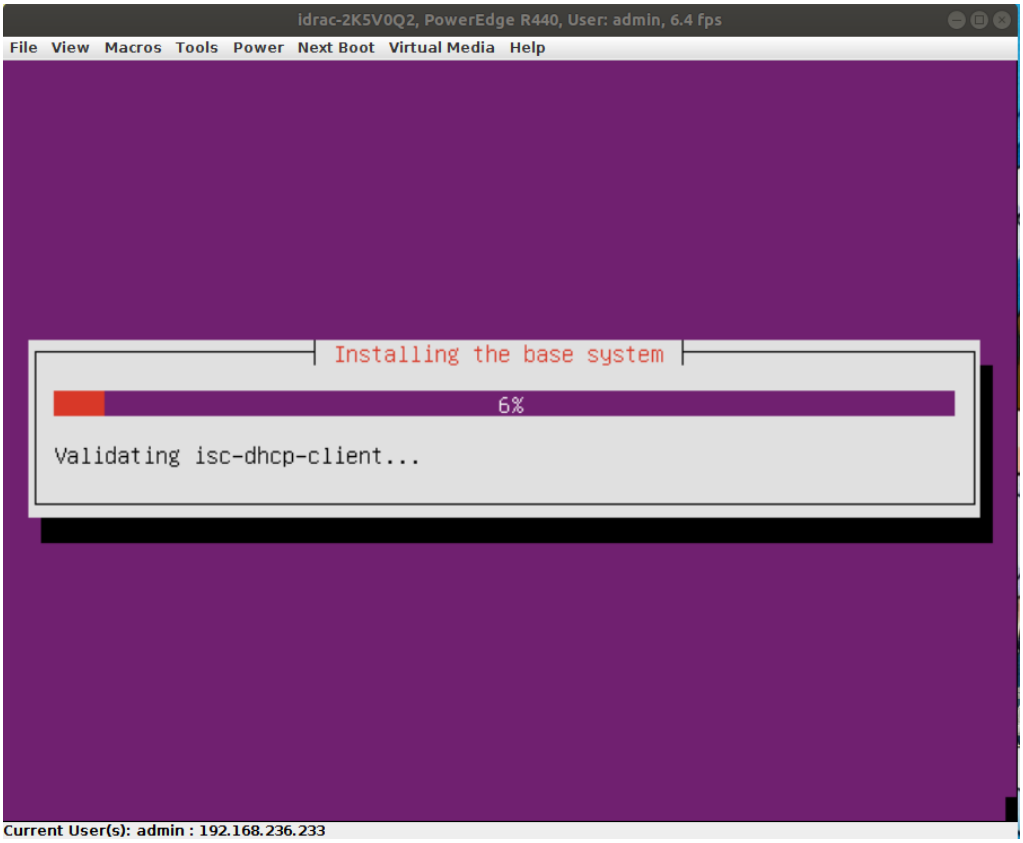


Рис. 3.2. Початок автоматичної установки ОС Ubuntu Server 18.04

3.3.4. Організація відмовостійкості

Відмовостійкість системи організована за рахунок двох компонентів:

1) Парні підключення сервера до мережевих комутаторів;
2) Інфраструктура мережі датацентра, побудована у стилі “Leaf-Spine”.

Парні підключення означають, що сервер, на якому планується організація відмовостійкості, підключений фізично двома різними мережевими картами до чотирьох мережевих комутаторів наступним чином:

1) Перший порт першої зовнішньої мережевої карти підключений до основного комутатора приватної мережі;
2) Другий порт першої зовнішньої мережевої карти підключений до основного комутатора публічної мережі;
3) Перший порт другої зовнішньої мережевої карти підключений до резервного комутатора приватної мережі;
4) Другий порт другої зовнішньої мережевої карти підключений до резервного комутатора публічної мережі.

Таким чином, навіть за поломки порту або усієї зовнішньої мережевої карти, сервер зберігатиме працеспроможність, але зі зменшенням швидкості пропускної спроможності, зменшеною вдвічі.

Окрім підключень сервера, схема мережі, до якої входить цей сервер, теж має бути побудована із задовільненням вимог відмовостійкості.

Архітектура “Leaf-Spine” є швидкою, масштабованою та ефективною схемою для зберігання серверів у межах датацентра. Вона має лише два рівні. За цих умов усі кінцеві пристрої рівновіддалені від комутаторів ядра (XCore). Таким чином, вони мають однакові та передбачувані затримки при передачі пакетів інформації. Ці рівні називаються “Spine” та “Leaf” [9].

Рівень “Leaf” складається з комутаторів доступу, до яких підключаються інші хости датацентра, у нашому випадку лише сервери. Рівень “Spine” також складається з комутаторів, до яких підключаються комутатори рівня “Leaf”. У

свою чергу, комутатори рівня “Spine” підключаються до маршрутизаторів, які і є ядром цієї системи.

На рис. 3.3 приведено схему класичної архітектури “Leaf-Spine”.

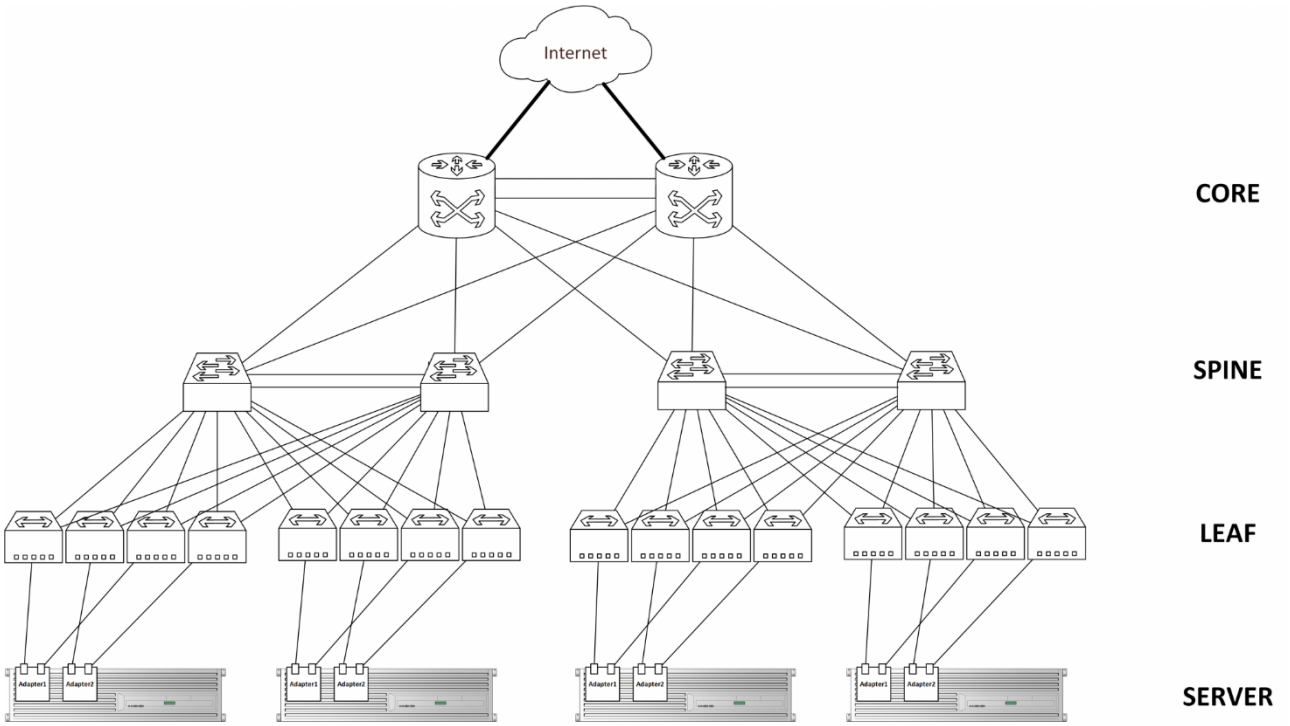


Рис. 3.3. Архітектура “Leaf-Spine”

Головною перевагою такої архітектури є легкість додавання нового обладнання та розширення інфраструктури. Головним недоліком є наявність великої кількості патчкордів, але за розумного підходу до задачі мінімізації простору, що займається патчкордами, цей недолік також стає перевагою, адже вдало організована та згрупована мережа патчкордів дозволяє не лише правильно використовувати обладнання, а й заощаджувати чимало часу на пошук потрібного патчкорду у разі потреби.

3.4. Перевірка коректної роботи сервера

Після завершення автоматичного розгортання виділеного сервера і установки операційної системи на сервер, щойно встановлена ОС виконує

перезавантаження сервера з метою застосування останніх системних налаштувань.

Оскільки у порядку загрузки сервера досі на першому місці стоїть перший порт першої зовнішньої мережевої карти, сервер знову намагатиметься загрузитись у PXE-меню. Але цього разу на цей порт не надходитиме сигналу з боку Leaf-комутатора, тому сервер намагатиметься загрузитися по PXE лише протягом перших 30 секунд. Далі він використовуватиме наступний пункт у порядку загрузки сервера – загрузку з диска C. На рис. 3.4 наведено скріншот віртуальної консолі сервера у той момент, коли сервер не зміг загрузитись за допомогою PXE, і тому переходить до загрузки з жорсткого диска.

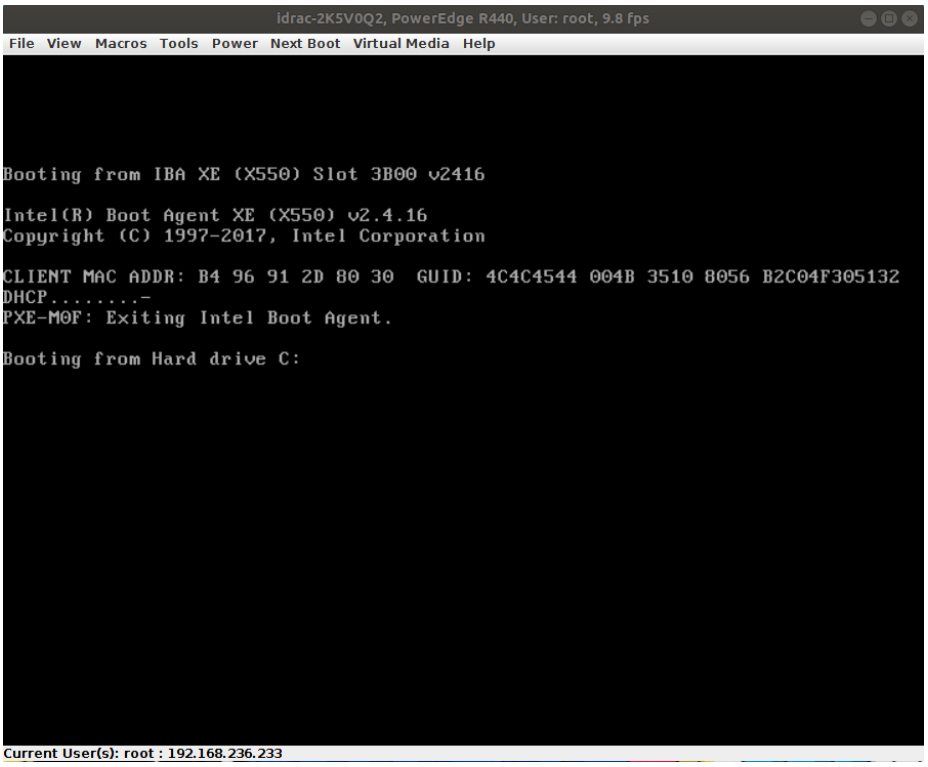


Рис. 3.4. Скріншот запуску сервера із жорсткого диска після установки ОС

Коли сервер загружає операційну систему, встановлену на нього, потрібно ввести отриманий пароль root. На рис. 3.5 наведено скріншот віртуальної консолі сервера на екрані аутентифікації.

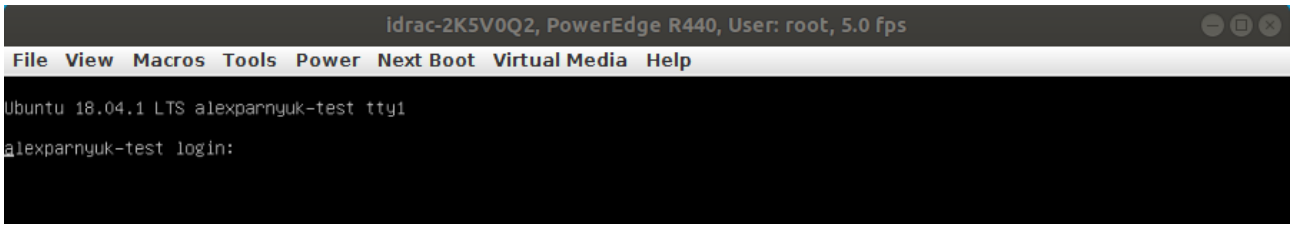


Рис. 3.5. Аутентифікація у встановленій ОС Ubuntu Server 18.04

Після вводу правильного пароля, операційна система вітає користувача і фактично готова до роботи. На рис. 3.6 наведено скріншот віртуальної консолі сервера після вводу правильного root-пароля.

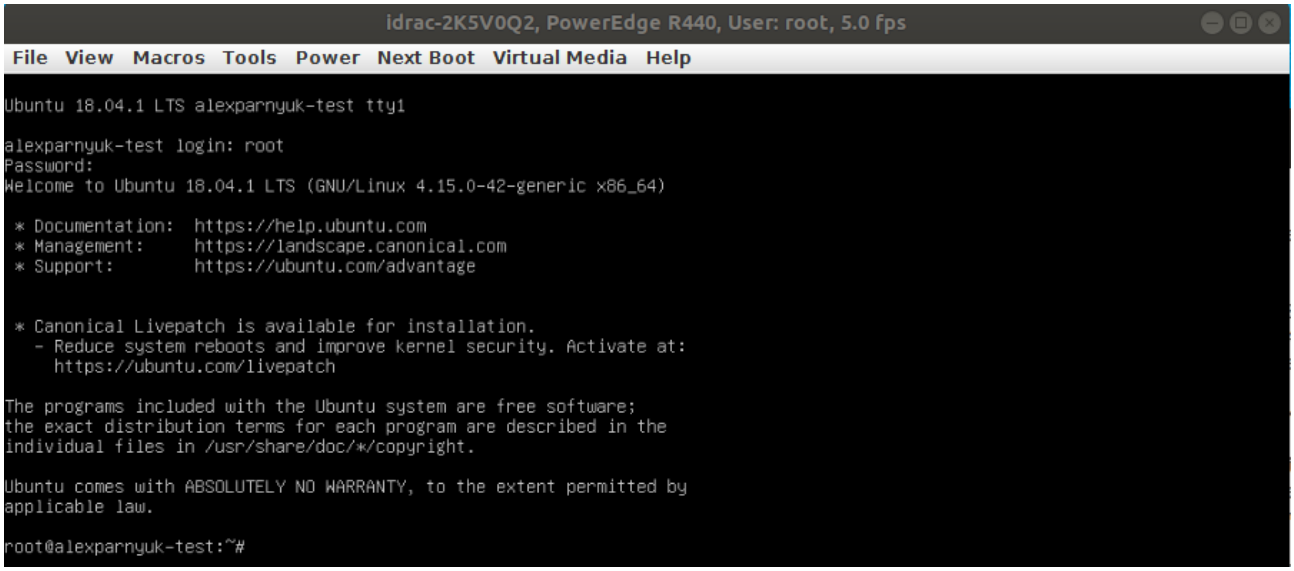


Рис. 3.6. Операційна система Ubuntu Server 18.04, що готова до роботи

Для того, щоб перевірити коректність роботи мережевих сервісів, можна скористатися командою ping. На рис. 3.7 наведено результат виконання команди ping (кінцева адреса 8.8.8.8 – Google).

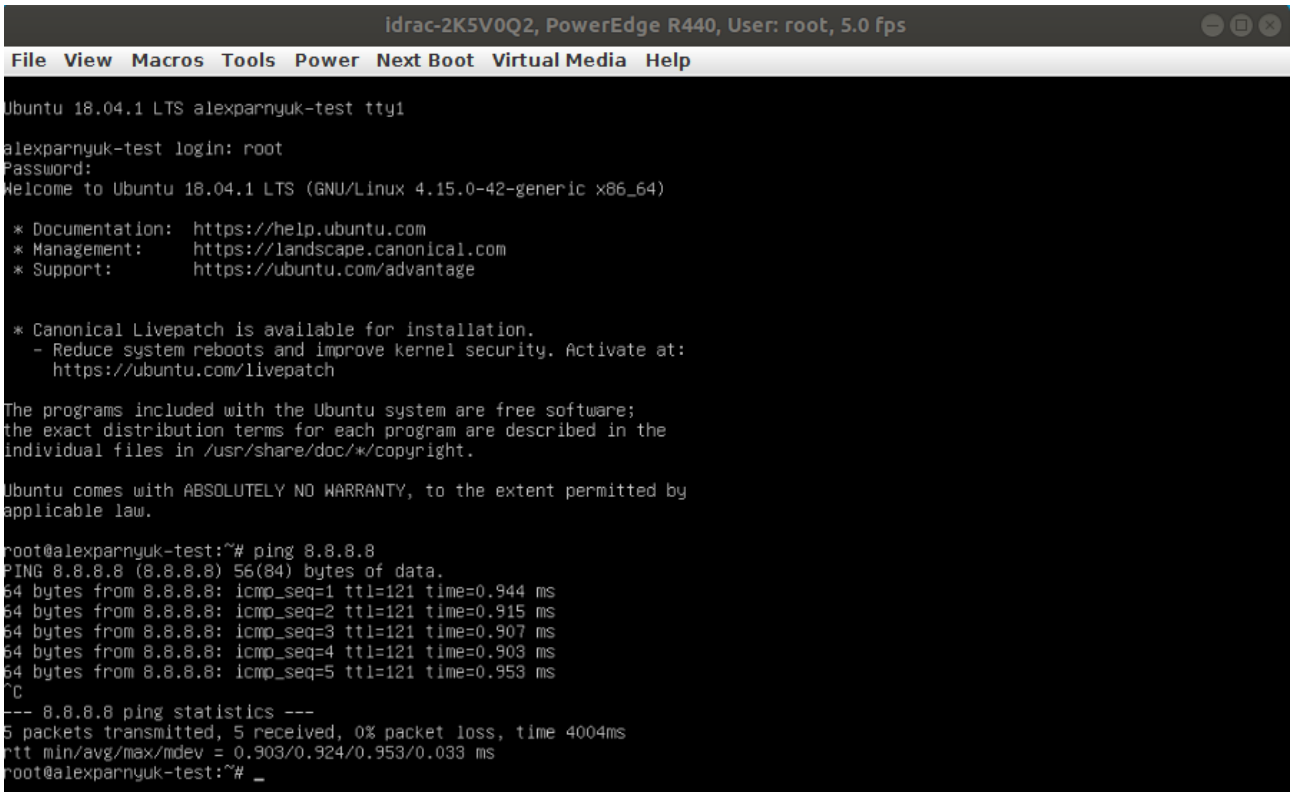


Рис. 3.7. Результат виконання команди ping до IP-адреси Google

Таким чином ми пересвідчилися у тому, що операційна система не лише встановлена успішно, а й має коректні мережеві налаштування. Фактично такий сервер повністю готовий до роботи, за винятком встановлення деяких не обов’язкових системних сервісів.

3.5. Висновок до третього розділу

У третьому розділі кваліфікаційної роботи було досліджено CLI-скрипт, за допомогою якого сервер приводиться до коректних ключових налаштувань. Також було досліджено KS-файл для автоматичної установки операційної системи Ubuntu Server 18.04.

В ході дослідження автоматичної системи було проведено та досліджено весь шлях сервера від базових налаштувань, до яких має бути приведений сервер перед автоматичним розгортанням, до повної готовності сервера до роботи. Було

досліджено практично, що система спроможна виконувати розгортання сервера коректно і без помилок.

РОЗДІЛ 4

ЕКОНОМІКА

4.1. Визначення трудомісткості розробки програмного забезпечення

Задані параметри:

1. Передбачуване число операторів – 700;
2. Коефіцієнт складності програми – 1,4;
3. Коефіцієнт кореляції програми в ході її розробки – 0,3;
4. Коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,1;
5. Коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 0,5;
6. Середня годинна заробітна плата програміста, грн/година – 90;
7. Трудомісткість налагодження програми на еом, год. – 45;
8. Вартість машино-години еом, грн/год – 7.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,}$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p),$$

де q - передбачуване число операторів;

C - коефіцієнт складності програми;

p - коефіцієнт кореляції програми в ході її розробки.

$$Q = 700 \cdot 1,4 \cdot (1 + 0,3) = 1274;$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин,}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

$$t_u = \frac{1274 \cdot 1,1}{75 \cdot 0,5} \approx 37,4 \text{ людино-годин}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20 \ldots 25) \cdot k}, \text{ людино-годин.}$$

$$t_a = \frac{1274}{20 \cdot 0,5} = 127, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot k}$$, людино-годин.

$$t_n = \frac{1274}{24\cdot 0,5} \approx 106,1$$, людино-годин.

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}$$, людино-годин.

$$t_{отл} = \frac{1274}{5\cdot 0,5} = 509,6$$, людино-годин.

- за умови комплексного налагодження завдання:

$$t^k_{отл} = 1,5\cdot t_{отл}$$, людино-годин.

$$t^k_{отл} = 1,5\cdot 509,6 = 764,4$$, людино-годин.

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}$$, людино-годин,

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{(15..20)\cdot k}$$, людино-годин.

$$t_{др} = \frac{1274}{20\cdot 0,5} = 127,4$$, людино-годин.

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0,75 \cdot t_{др}$$, людино-годин.

$$t_{до} = 0,75 \cdot 127,4 \approx 96$$, людино-годин.

$$t_{д} = 127,4 + 96 = 223,4$$, людино-годин.

Розрахуємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 37,4 + 127 + 106,1 + 509,6 + 223,4 = 1053,5$$ людино-годин.

4.2. Витрати на створення програмного забезпечення

Витрати на створення ПЗ *Кпо* включають витрати на заробітну плату виконавця програми *Зз/п* і витрат машинного часу, необхідного на налагодження програми на ЕОМ

$$К_{ПО} = З_{ЗП} + З_{МВ}$$, грн.

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}$$, грн,

$$З_{ЗП} = 1053,5 \cdot 90 = 94815$$, грн,

де: *t* - загальна трудомісткість, людино-годин;

$С_{пр}$ - середня годинна заробітна плата програміста, грн/година

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{мв} = t_{отл} \cdot С_{мч}$$, грн,

$$З_{мв} = 509,6 \cdot 7 = 3567,2 \text{, грн,}$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год.

$C_{мч}$ - вартість машино-години ЕОМ, грн/год.

Тоді:

$$К_{ПО} = 94815 + 3567,2 = 98382,2 \text{, грн.}$$

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{, міс,}$$

де B_k - число виконавців;

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

$$T = \frac{1053,5}{1 \cdot 176} \approx 5,98 \text{, міс,}$$

4.3. Маркетингові дослідження ринку збуту дослідженого програмного продукту

Основним ринком збуту програмного забезпечення є компанії, які виконують роль хостинг-провайдерів. Вони мають сотні, а подекуди і десятки тисяч фізичних серверів, які можуть бути розташовані по всій земній кулі. Такі компанії, як правило, займаються наданням серверів у оренду кінцевим клієнтам.

На сьогоднішній день методи розгортання виділених серверів здебільшого автоматизовані. У цій роботі підіймається питання покращення та прискорення автоматизованих процесів, вдосконалення існуючої теоретичної моделі.

Варто зазначити, що аналогічні моделі у вільному доступі захищені так званим “ноу-хау” і неможливо відтворити їх роботу без відтворення аналогічної інфраструктури.

Виходячи з цього, основним клієнтом може бути компанія, яка має змогу привести свою інфраструктуру до необхідного вигляду з метою впровадження моделі, описаної у роботі.

4.4. Оцінка економічної ефективності впровадження програмного забезпечення

Оскільки даний продукт не має економічних показників, ми не маємо можливості розрахувати його економічну ефективність впровадження, але ми отримаємо ефект від реалізації результатів роботи, який полягає у наступному:

1) зменшення кількості працівників за рахунок автоматизації рутинних процесів та значному зростанню показників автономності системи від впливу людського фактору;
2) скорочення часу на підготовку, перевірку сервера, установку на нього операційної системи;
3) зменшення помилок та неточностей при установці операційної системи за рахунок передачі у kickstart-файл параметрів змовника напряму;
4) збільшення відмовостійкості та, як наслідок, зменшення аварійних ситуацій, що можуть викликати погіршення репутації.

ВИСНОВКИ

Виконана науково-кваліфікаційна робота, що містить науково обґрунтовані, методичні, організаційні та технічні рішення для підвищення ефективності налаштування та розгортання фізичних виділених серверів.

У рамках кваліфікаційної роботи магістра було виконано огляд апаратних складових серверного обладнання: фізичного виділеного сервера та мережевого комутатора, розглянуто основні протоколи мережевої взаємодії, було оглянуто та проаналізовано сучасні сімейства серверних операційних систем. Було обрано операційну систему, яка, на думку автора, підходить найбільше, а саме Ubuntu Server 18.04.

Було встановлено, що сучасні технології автоматизації рутинних процесів розгортання серверів не дозволяють проводити попередні ключові налаштування сервера та його подальше розгортання без суттєвих витрат часу, багато з якого відводиться на застосування необхідних налаштувань вручну. Виходячи з цього, було досліджено алгоритм автоматичного розгортання виділеного фізичного сервера.

Цей алгоритм включає в себе основні налаштування сервера, перевірку коректності мережевих підключень та автоматичну установку операційної системи. Використовуючи цей алгоритм, було досліджено спеціальний CLI-скрипт та kickstart-файл, які задовольняють потреби сервера та виконують автоматично необхідні налаштування.

Після дослідження CLI-скрипта, його було застосовано під час тестування програмного забезпечення. Було досліджено, що система повністю спроможна виконувати поставлені перед нею задачі.

Швидкість проходження сервера через усі етапи налаштування і встановлення операційної системи склало 27 хвилин.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Оліфер В. Г. Комп’ютерні мережі. Принципи, технології, протоколи / Оліфер В.Г., Оліфер Н.А.; Підручник для ВНЗ. 4-е вид. — СПб Пітер, 2010. — 944 с.: іл.
2. Таненбаум Э. Комп’ютерні мережі / Э. Таненбаум, Д. Уэзерол; 5-е вид. — СПб: Пітер, 2016. — 960 с.: іл.
3. Yochai B. The Wealth of Networks: How Social Production Transforms Markets and Freedom / Yochai B; Edition 2, Publisher “Yale University Press”, 2006. — 516 p.: il.
4. Mazin G. Artificial Intelligence for Autonomous Networks / Mazin G.; Edition 1, Publisher "Chapman and Hall/CRC", 2018. — 392 p.: il.
5. Raymond R. Business Data Networks and Security 10th Edition / Raymond R.Panko, Julia L. Panko; Edition 1, Publisher “Prentice Hall”, 2014. — 528 p.: il.
6. Phil P. Concept of database managment eighth edition / Plil Pratt, Mary Lost; Edition 2, Publisher "Cengage Learning", 2014. — 432 p.: il.
7. Куроуз Д. Комп’ютерні мережі. Спадний підхід / Куроуз Д., Росс К.; 6-е вид. — Ексмо, 2016. — 912 с.: іл.
8. Сергєєв О. М. Основи локальних комп’ютерних мереж. Навчальний посібник / Сергєєв О. М.; 1-е вид. — Лань, 2016. — 184 с.: іл.
9. Куроуз Д. Комп’ютерні мережі. Настільна книга системного адміністратора / Куроуз Д., Росс Т.; 1-е вид. — Ексмо, 2016. — 932 с.: іл.
10. Робачевський А. Інтернет зсередини: Екосистема глобальної мережі / А. Робачевський; 1-е вид. — Альпіна Паблішер, 2015. — 223 с.: іл.
11. Одом У. Офіційне керівництво Cisco для підготовки до сертифікованих екзаменів CCNA ICND2 200-101. Маршрутизація та комутація / Одом У.; 1-вид. — Діатектика Вільямс, 2016. — 736 с.: іл.
12. Фоулер Ч. Rails. Збірник рецептів / Фоулер Ч.; 1-е вид. — СПб Пітер, 2008. — 256 с.: іл.

13. Метц С. Ruby. Об’єктно-орієнтоване програмування / Метц С.; 1-е вид. — СПб Пітер, 2008. — 258 с.: іл.
14. Едельсон Дж. Ruby на платформі Java / Едельсон Дж., Лю Г.; 2-е вид. — ДМК Пресс, 2011. — 240 с.: іл.
15. Фленаган Д. Мова програмування Ruby / Фленаган Д., Мацумото Ю.; 1-е вид. — СПб Пітер, 2011. — 496 с.: іл.
16. Фернандес О. Шлях Rails. Докладне керівництво по створенню додатків у середі Ruby on Rails/ Фернандес О.; 4-е вид. — Символ-Плюс, 2009 — 768 с.: іл.
17. Robert G. Byrnes. SSH, The Secure Shell: The Definitive Guide, 2nd Edition/ Robert G. Byrnes, Richard E. Silverman, Daniel J. Barrett; Publisher "O'Reilly Media", 2005 — 670 p.: il.
18. Родерик В. Сміт Мережеві засоби Linux / Родерик В. Сміт; 1-е вид. — "Вильямс", 2003 — 672 с.: іл.
19. Бараш Л. Мережеве обладнання масової пам’яті [Електронний ресурс]. – 1999. – Доступ за посиланням: http://itsfor.narod.ru/net_hard/2/115_1.htm
20. Егоров А. RAID 0, RAID 1, RAID 5, RAID 10 або що таке рівні RAID? / Компания «ТИМ» [Електронний ресурс]. – 2005. – Доступ за посиланням: http://timcompany.ru/article4.html
21. Журавльов В. Технології SAN // Комп’ютерний портал PCGuru.ru [Електронний ресурс]. – Доступ за посиланням: http://pcguru.ru/arhiv/santechnology.shtml
22. Зиновьєв А. Тлумачний англо-російський словник по RAID технології // Новий Клондайк [Електронний ресурс]. – Доступ за посиланням: http://tech.stolica.ru/bus/raid/faqraid.htm
23. Курмаз М. Переваги RAID // Белорусский «железный» сайт [Електронний ресурс]. – 2001. – Доступ за посиланням: http://www.hw.by
24. Поляков А. Тестування продуктивності дискового масива SATA RAID // Тестовая лаборатория Ferra [Електронний ресурс]. – 2004. –Доступ за посиланням: http://www.ferra.ru/online/storage/25425/

25. Радаев А. Знайомство з RAID-масивами // Тестовая лаборатория Ferra [Електронний ресурс]. – 2005. – Доступ за посиланням: http://www.ferra.ru/online/storage/26107/
26. Руководство по Fibre Channel // Засоби і системи комп'ютерної автоматизації [Електронний ресурс]. – 2008. – Режим доступа: http://www.asutp.ru/?p=600021
27. Голобродський К. В. Знайомтесь: Ubuntu / Голобродський К. В; 1-е вид. — Феникс, 2010. — 160 с.: іл.
28. Brian W How Linux Works: What Every Superuser Should Know / Brian Ward; Edition 1, Publisher "No Starch Press", 2016. — 368 p.: il.
29. Колісніченко Д. Н. Командна строка Linux та автоматизація рутинних задач / Колісніченко Д. Н.; 3-є вид. — БХВ-Петербург, 2014. — 352 с.: іл.
30. Девид П. Mac OS X Leopard. Основне керівництво / Девид П.; 1-е вид. — Символ-Плюс, 2008. — 880 с.: іл.
31. Knaster S. Learn Objective-C on the Mac For OS X and iOS / Knaster Scott, Dalrymple Mark, Malik Waqar; Edition 3, Publisher "Apress", 2012. — 384 p.: il.
32. Kevin M. White OS X Support Essentials 10.10 Apple Pro Training Series - Supporting and Troubleshooting OS X Yosemite, 1/e / Kevin M. White & Gordon Davisson; Edition 1, Publisher "Pearson Education", 2015. — 936 p.: il.
33. Guy Hart-Davis AppleScript: A Beginner's Guide 1st Edition / Guy Hart-Davis; Edition 1, Publisher "McGraw-Hill Education", 2009. — 448 p.: il.
34. Родичев Ю. А. Нормативна база та стандарти в області інформаційної безпеки. Навчальний посібник / Родичев Ю.А.; 2-е вид. — СПб: Пітер, 2017. —256с.
35. Баранова Є.К. Інформаційна безпека та захист. Навчальний посібник / Баронова Є.К., Бабаш А.В.; 1-е вид. —РІОР, Інфра-М, 2017. — 324 с.

36. Бирюков А.А. Інформаційна безпека. Захист та напад / Бирюкав А.А.; 1-е вид. — ДМК-Прес, 2017. — 434 с.

ДОДАТОК А

ЛІСТИНГ CLI-СКРИПТА

```
#!/usr/bin/env ruby

#
# DMGMTSrvInit 1.0 configuration script
#

require 'net/http'
require 'open3'
require 'logger'
require 'fileutils'

STDOUT.sync = true
STDERR.sync = true

BIOS_UTIL = '/opt/dell/toolkit/bin/syscfg'
DRAC_UTIL = '/opt/dell/srvadmin/bin/idracadm7'
RAID_UTIL = '/opt/dell/toolkit/bin/raidcfg'

DEFAULT_PXESERVER_ROUTE = '192.168.0.0/16'

HWADDR = "0c:c4:7a:85:19:a4"
PXESERVER_URL = "http://pxe.servers.com"

SETTINGS = {"location_id"=>1, "server_model_id"=>145, "ram_size"=>128,
"os"=>{"name"=>"dmgmtsrvinit", "version"=>"1.0", "arch"=>"x86_64"}, "hdds"=>{"0"=>{"interface"=>7,
"physical_size"=>2, "hdd"=>{"id"=>11, "name"=>"1 TB SATA 2.5\"", "size"=>1000}},
"1"=>{"interface"=>7, "physical_size"=>2, "hdd"=>{"id"=>11, "name"=>"1 TB SATA 2.5\"",
"size"=>1000}}, "2"=>{"interface"=>7, "physical_size"=>2, "hdd"=>nil}, "3"=>{"interface"=>7,
"physical_size"=>2, "hdd"=>nil}, "4"=>{"interface"=>7, "physical_size"=>2, "hdd"=>nil},
"5"=>{"interface"=>7, "physical_size"=>2, "hdd"=>nil}, "6"=>{"interface"=>7, "physical_size"=>2,
"hdd"=>nil}, "7"=>{"interface"=>7, "physical_size"=>2, "hdd"=>nil}, "8"=>{"interface"=>7,
"physical_size"=>2, "hdd"=>nil}, "9"=>{"interface"=>7, "physical_size"=>2, "hdd"=>nil},
"10"=>{"interface"=>7, "physical_size"=>2, "hdd"=>nil}, "11"=>{"interface"=>7, "physical_size"=>2,
"hdd"=>nil}, "12"=>{"interface"=>7, "physical_size"=>2, "hdd"=>nil}, "13"=>{"interface"=>7,
"physical_size"=>2, "hdd"=>nil}, "14"=>{"interface"=>7, "physical_size"=>2, "hdd"=>nil},
"15"=>{"interface"=>7, "physical_size"=>2, "hdd"=>nil}, "16"=>{"interface"=>7, "physical_size"=>2,
"hdd"=>nil}, "17"=>{"interface"=>7, "physical_size"=>2, "hdd"=>nil}, "18"=>{"interface"=>7,
"physical_size"=>2, "hdd"=>nil}, "19"=>{"interface"=>7, "physical_size"=>2, "hdd"=>nil},
"20"=>{"interface"=>7, "physical_size"=>2, "hdd"=>nil}, "21"=>{"interface"=>7, "physical_size"=>2,
"hdd"=>nil}, "22"=>{"interface"=>7, "physical_size"=>2, "hdd"=>nil}, "23"=>{"interface"=>7,
"physical_size"=>2, "hdd"=>nil}}, "disks"=>[{"disks"=>["0", "1"], "partitions"=>[{"target"=>"/boot",
"fs"=>"ext3", "size"=>500, "fill"=>false}, {"target"=>"swap", "fs"=>"swap", "size"=>2048,
"fill"=>false}, {"target"=>"/", "fs"=>"ext4", "size"=>997452, "fill"=>true}], "raid"=>1}],
"uplinks"=>{"public"=>{"id"=>1, "uplink_port_speed"=>1000, "redundancy"=>false},
"private"=>{"id"=>2, "uplink_port_speed"=>1000, "redundancy"=>false}}, "public_bandwidth"=>1,
"ips"=>0, "host_names"=>["hack-ai-13", "hack-ai-14", "hack-ai-15", "hack-ai-16", "hack-ai-17"],
"server_model_name"=>"Supermicro 4028GR-TR / 128 GB RAM / 2 x Nvidia GTX 1080 GPU", "rack_id"=>nil,
"rack_name"=>"shared", "hostname"=>"hack-ai-15", "provisioning_eta"=>1440, "ticket_id"=>"2592719",
"kraken_conf"=>{"stage_2071300_1"=>{"kraken_task_id"=>7672},
"stage_2071303_1"=>{"kraken_task_id"=>7673}, "stage_2071312_1"=>{"kraken_task_id"=>7674}},
"network_conf"=>{"oob_network_stage"=>{"switch_group"=>"smgmt-nl01.servers.com",
"oob_port"=>"ethernet1/1/22"}, "routing_alias_ips_stage"=>{"switch_group"=>"se-nl01-l18"}},
"networks"=>[{"type"=>"drac", "ip"=>"198.18.35.37", "netmask"=>"255.255.255.254",
"routes"=>[{"network"=>"default", "via"=>"198.18.35.36"}]}], "manage_bios"=>true,
"manage_drac"=>false, "manage_raid"=>true, "password"=>"VWH4SPxoYcRV", "server_id"=>7503,
"final_status"=>392, "mustdie"=>false, "r220"=>false}
STATUSES = {"INIT"=>310, "CLI_LOADED"=>311, "RAID"=>320, "BIOS"=>330, "RACRESETCFG"=>331,
"DRAC"=>340, "CLEAR_HDD"=>350, "CLEAR_HDD_START"=>351, "CLEAR_HDD_FINISHED"=>352,
"FIRMWARE_UPDATE"=>360, "RESCUE"=>380, "DONE"=>390, "DONE_POWER_OFF"=>391, "CONTINUE"=>392,
"FAILED"=>399}

def configure_bios
  set_status :bios

  run [BIOS_UTIL, '--acpower=on'], :ignore_exceptions => true
  sleep 5

  run [BIOS_UTIL, '--memtest=disable'], :ignore_exceptions => true
  sleep 5
```

```
  run [BIOS_UTIL, '--rptkeyerr=disable'], :ignore_exceptions => true
  sleep 5

  run [BIOS_UTIL, '--f1f2promptonerror=disable'], :ignore_exceptions => true
  sleep 5

  run [BIOS_UTIL, '--numlock=off'], :ignore_exceptions => true
  sleep 5

  run [BIOS_UTIL, '--memoperatingmode=optimizermode'], :ignore_exceptions => true
  sleep 5

  run [BIOS_UTIL, '--virtualization=enabled'], :ignore_exceptions => true
  sleep 5

  run [BIOS_UTIL, '--bootseqretry=enable'], :ignore_exceptions => true
  sleep 5


end

def configure_drac
  set_status :drac

  out = run [DRAC_UTIL, 'set', 'idrac.snmp.agentenable', 'Disabled'], :out => true
  raise StandardError, 'snmp.agentenable' unless out.match(/successfully/)
  sleep 5

  out = run [DRAC_UTIL, 'set', 'idrac.vncserver.enable', 'Enabled'], :out => true
  raise StandardError, 'vnc.enable' unless out.match(/successfully/)
  sleep 5

  out = run [DRAC_UTIL, 'set', 'idrac.vncserver.password', SETTINGS['drac_password'][0, 8]], :out =>
true
  raise StandardError, 'vnc.password' unless out.match(/successfully/)
  sleep 5

  out = run [DRAC_UTIL, 'set', 'idrac.virtualmedia.attached', 'Detached'], :out => true
  raise StandardError, 'virtualmedia' unless out.match(/successfully/)
  sleep 5

  true
end

def configure_drac_credentials
  if SETTINGS['drac_password']
    out = run [DRAC_UTIL, 'set', 'idrac.users.2.password', SETTINGS['drac_password']], :out => true
    raise StandardError, 'users.2.password' unless out.match(/successfully/)
    sleep 5
  end

  if SETTINGS['networks']
    nw = SETTINGS['networks'].select { |n| n['type'].upcase == 'DRAC' }.first
    raise StandardError, 'no networks' if nw.empty?

    out = run [DRAC_UTIL, 'setniccfg', '-s', nw['ip'], nw['netmask'], nw['routes'][0]['via']], :out
=> true
    raise StandardError, 'setniccfg' unless out.match(/successfully/)
  end
end

def configure_drac_power_redundancy
  out = run [DRAC_UTIL, 'set', 'System.Power.RedundancyPolicy', '1'], :out => true
  raise StandardError, 'System.Power.RedundancyPolicy' unless out.match(/successfully/)
  sleep 5

  out = run [DRAC_UTIL, 'set', 'System.Power.HotSpare.Enable', '0'], :out => true
  raise StandardError, 'System.Power.HotSpare.Enable' unless out.match(/successfully/)
  sleep 5
end

def configure_raid
  set_status :raid

  ctrl_id = get_raid_ctrl_id
  x, z = get_raid_disk_locators

  cutsize = 51200
```

```
  mustdie = false

  # clear foreing config
  run [RAID_UTIL, "-ctrl",  "-c=#{ctrl_id}", "-ac=fgnclr"], :ignore_exceptions => true

  # delete all virtual disks
  run [RAID_UTIL, "-c=#{ctrl_id}", "-vd", "-ac=dvd"], :ignore_exceptions => true

  # create virtual disks
  vi = 0
  SETTINGS['disks'].each_with_index do |d, index|
    # make RAID of an every single disk
    d['raid'] ||= 0

    disks = d['disks'].map { |y| sprintf("%d:%s:%d", x, y.match(/(\d+)$/)[1], z) }.join(',')

    begin
    out = run [RAID_UTIL, "-ctrl | grep Array_Disks | awk '{ print \$2 }'"], :out => true
      ctr_disks = out.split(/\n/).join(',').split(/,/)

      ctr_disks.each do |disk|
        run [RAID_UTIL, "-ad", "-ac=ctr", "-ad=#{disk}", "-c=#{ctrl_id}"], :ignore_exceptions =>
true
      end
    end

    if d['raid'] == 50 || d['raid'] == 60
      count = d['disks'].length / 2
      sp = "-sp=#{count}"

      if mustdie && index == 0
        run [RAID_UTIL, "-ctrl", "-ac=cvd", "-c=#{ctrl_id}", "-ad=#{disks}", "-sz=#{cutsize}", "-
r=#{d['raid']}", sp]
        sleep 5
        run [RAID_UTIL, "-c=#{ctrl_id}", "-vd", "-ac=fi", "-vd=#{vi}"], :ignore_exceptions => true
        vi += 1
      end

      run [RAID_UTIL, "-ctrl", "-ac=cvd", "-c=#{ctrl_id}", "-ad=#{disks}", "-r=#{d['raid']}", sp]
    else

      if mustdie && index == 0
        run [RAID_UTIL, "-ctrl", "-ac=cvd", "-c=#{ctrl_id}", "-ad=#{disks}", "-sz=#{cutsize}", "-
r=#{d['raid']}"]
        sleep 5
        run [RAID_UTIL, "-c=#{ctrl_id}", "-vd", "-ac=fi", "-vd=#{vi}"], :ignore_exceptions => true
        vi += 1
      end

      run [RAID_UTIL, "-ctrl", "-ac=cvd", "-c=#{ctrl_id}", "-ad=#{disks}", "-r=#{d['raid']}"]
    end

    sleep 5
    run [RAID_UTIL, "-c=#{ctrl_id}", "-vd", "-ac=fi", "-vd=#{vi}"], :ignore_exceptions => true
    vi += 1
  end

  true
end

def server_cleanup
  set_status :racresetcfg



  ctrl_id = get_raid_ctrl_id

  if !ctrl_id.nil? && !ctrl_id.strip.empty?
    set_status :raid

    # clear foreing config
    run [RAID_UTIL, "-ctrl",  "-c=#{ctrl_id}", "-ac=fgnclr"], :ignore_exceptions => true

    # delete all virtual disks
    run [RAID_UTIL, "-c=#{ctrl_id}", "-vd", "-ac=dvd"], :ignore_exceptions => true

    # make an array of disks
    out = run [RAID_UTIL, "-ctrl | grep Array_Disks | awk '{ print \$2 }'"], :out => true
    disks = out.split(/\n/).join(',').split(/,/)
```

```
    disks.each do |disk|
      # change HBA to RAID mode
      run [RAID_UTIL, "-ad", "-ac=ctr", "-ad=#{disk}", "-c=#{ctrl_id}"], :ignore_exceptions => true

      # create a RAID0 virtual disk for each disk
      run [RAID_UTIL, "-ctrl", "-ac=cvd", "-c=#{ctrl_id}", "-r=0", "-ad=#{disk}"]
    end

    set_status :clear_hdd_start
    clear_disks!
    set_status :clear_hdd_finished

    # delete the virtual disk
    set_status :raid
    run [RAID_UTIL, "-c=#{ctrl_id}", "-vd", "-ac=dvd"]
  else
    set_status :clear_hdd_start
    clear_disks!
    set_status :clear_hdd_finished
  end

  true
end

def rescue_boot
  set_status :rescue
  external_network = SETTINGS['networks'].find { |n| n['type'].include?('external') }
  gateway = external_network['routes'].first['via']
  password = SETTINGS['password'].crypt('$6$' + external_network['ip'])
  pxeserver_route = SETTINGS['pxeserver_route'] || DEFAULT_PXESERVER_ROUTE
  external_network_has_bonding = external_network['hwaddrs'].length > 1

  sleep 30
  pgw = run ["ip r|grep ^default|awk {'print $3'}"], :out => true
  pgw.delete!("\n")
  pname = run ["ip r|grep ^default|awk {'print $NF'}"], :out => true
  pname.delete!("\n")
  run ['ip r'], :out => true
  run ['killall -9 dhclient'], :out => true
  sleep 10
  run ["/sbin/dhclient -H dmgmtsrvinit -1 -q -lf /var/lib/dhclient/dhclient-#{pname}.leases -pf
/var/run/dhclient-#{pname}.pid -R 'subnet-mask, broadcast-address, time-offset, domain-name-servers,
netbios-scope, interface-mtu' #{pname}"], :out => true
  run ['ip r add', pxeserver_route, 'via', pgw], :out => true
  run ['ip r del default'], :out => true

  if external_network_has_bonding
    pi1 = run ['grep -i', external_network['hwaddrs'][0], "/sys/class/net/*/address|sed -e
's|/sys/class/net/||g' -e 's|/address.*||g'"], :out => true
    pi2 = run ['grep -i', external_network['hwaddrs'][1], "/sys/class/net/*/address|sed -e
's|/sys/class/net/||g' -e 's|/address.*||g'"], :out => true
    pi1.delete!("\n")
    pi2.delete!("\n")
    run ['modprobe bonding'], :out => true
    run ['ip link set bond0 down'], :out => true
    run ['echo 4 > /sys/class/net/bond0/bonding/mode'], :out => true
    run ['echo 1 > /sys/class/net/bond0/bonding/xmit_hash_policy'], :out => true
    run ['ip link set bond0 up'], :out => true
    run ['ifenslave bond0', pi1, pi2], :out => true
    run ['ip link set', pi1, 'up'], :out => true
    run ['ip link set', pi2, 'up'], :out => true
    run ["ip a add #{external_network['ip']}/#{external_network['netmask']} dev bond0"], :out =>
true
  else
    pi = run ['grep -i', external_network['hwaddrs'].first, "/sys/class/net/*/address|sed -e
's|/sys/class/net/||g' -e 's|/address.*||g'"], :out => true
    pi.delete!("\n")
    run ["ip a add #{external_network['ip']}/#{external_network['netmask']} dev", pi], :out => true
  end

  run ['ip r add default via', gateway], :out => true
  run ['ip r'], :out => true

  run ['arping -c 1', gateway], :out => true
  run ['ping -c 1', gateway], :out => true

  switch_1_ip = gateway.sub(/\d+\Z/) {|x| x.to_i - 1}
  run ['arping -c 1', switch_1_ip], :out => true
```

```
  run ['ping -c 1', switch_1_ip], :out => true

  if external_network_has_bonding
    switch_2_ip = gateway.sub(/\d+\Z/) {|x| x.to_i - 2}
    run ['arping -c 1', switch_2_ip], :out => true
    run ['ping -c 1', switch_2_ip], :out => true
  end

  run ['usermod', " -p '#{password}'", 'root'], :out => true
end

def clear_disks!
  log_files = []

  list_block_devices.each do |d|
    log_to_file = "/tmp/shred/#{d}.txt"
    log_files << log_to_file

    fork { shred_disk("/dev/#{d}", log_to_file) }
  end

  Process.wait

  log_files.each { |log_file| log(File.read(log_file)) }
end # clear_disks!

def shred_disk(disk, log_to_file)
  run ['/usr/bin/shred', '-fvzn 0', disk], :log_to_file => log_to_file
end

def list_block_devices
  out = run ["/bin/lsblk | grep disk | awk '{ print \$1 }' | sort"], :out => true
  out.split("\n")
end

def get_raid_ctrl_id
  out = run [RAID_UTIL, '-ctrl'], :out => true
  if m = out.match(/Controller_ID\/Slot_ID: (\d*)/)
    m[1]
  end
end

def get_raid_disk_locators
  out = run [RAID_UTIL, '-ctrl'], :out => true
  out.match(/Array_Disks: (?:(\d+):\d+:(\d+))*,?/)[1, 2]
end

# @@@@

def set_status(status, msg = '')
  log("Status #{status}: #{msg}")
  unless status = STATUSES[status.to_s.upcase]
    status = STATUSES['FAILED']
    log("Failed: unknown status")
  end
  attempts = 5
  begin
    log("attempts: #{attempts}")

Net::HTTP.get(URI.parse(URI.escape("#{PXESERVER_URL}/status/#{HWADDR}/notify?status=#{status}&messag
e=#{msg}")))
  rescue Exception => e
    attempts -= 1
    if attempts > 0
      sleep 3
      retry
    else
      raise e
    end
  end
end

def run(args, params = {})
  cmd = args.join(' ')
  log(cmd)

  if params[:out]
    out = `#{cmd}`
    log(out)
```

```
    return out
  elsif params[:log_to_file]
    run_and_log_to_file(cmd, params[:log_to_file])
  else
    success = system(cmd)
    raise StandardError, "#{cmd} failed" if !success && !params[:ignore_exceptions]
  end

  true
rescue StandardError => e
  upload_log
  log(e.message)
  set_status :failed, e.message
  exit false
end # run

def log(message)
  $stderr.puts(message) # Why $stderr ?
end

def run_and_log_to_file(cmd, log_to_file)
  dirname = File.dirname(log_to_file)
  FileUtils.mkdir_p(dirname) unless File.directory?(dirname)

  logger = Logger.new(log_to_file).tap do |l|
    l.formatter = proc do |severity, datetime, _progname, msg|
      "[#{datetime.strftime('%Y-%m-%d %H:%M:%S')} ##{Process.pid}] #{severity} -- : #{msg}"
    end
  end

  Open3.popen3(cmd) do |_stdin, stdout, stderr, _wait_thread|
    stdout.sync = true
    stderr.sync = true

    thread_out = Thread.new do
      while str = stdout.gets
        logger.info(str)
      end
    end

    thread_err = Thread.new do
      while str = stderr.gets
        logger.fatal(str)
      end
    end

    thread_err.join
    thread_out.join
  end
end # run_and_log_to_file

def upload_log
  dt = Time.now.strftime("%Y%m%d/%H%M")
  hwaddr = HWADDR.tr(":", "-")

  attempts = 5
  begin
    log("attempts: #{attempts}")
    run ["curl -T /tmp/srvinit.log #{PXESERVER_URL}/upload/#{dt}-#{hwaddr}.log"]
  rescue Exception => e
    attempts -= 1
    if attempts > 0
      sleep 3
      retry
    end
  end
end

def current_status
  status_json = Net::HTTP.get(URI.parse(URI.escape("#{PXESERVER_URL}/status/#{HWADDR}")))
  status_id = status_json[/:(\d+)\}/, 1].to_i
  current_status = STATUSES.to_a.rassoc(status_id)[0].downcase.to_sym
end

def after_firmware_update?(initial_status)
  initial_status == :firmware_update
end

def update_firmware
```

```
  output = run ['dsu', '-n'], :out => true

  if output.include?("Found firmware which needs to be updated")
    set_status :firmware_update
    true
  else
    false
  end
end

# @@@@

begin
  initial_status = current_status()

  set_status :cli_loaded unless after_firmware_update?(initial_status)

  # ERB MANAGE_BIOS
  configure_bios();
  # ~
  # ERB MANAGE_DRAC
  # ERB MANAGE_DRAC_PWRRED
  # ERB MANAGE_DRAC_CREDENTIALS
  # ERB MANAGE_RAID
  configure_raid();
  # ~
  # ERB MANAGE_CLEAR
  # ERB MANAGE_RESCUE

  #
    # finalize
    set_status :continue
    Net::HTTP.get(URI.parse("#{PXESERVER_URL}/status/#{HWADDR}/finalize"))
    upload_log
      run ["reboot"]
  #

rescue StandardError => e
  upload_log
  log(e.message)
  set_status :failed, "#{e.message}@#{e.backtrace[0]}"
  exit false
end
```

ЛІСТИНГ KS-ФАЙЛА

```
d-i console-keymaps-at/keymap           select us

d-i debian-installer/country            string NL
d-i debian-installer/language           string en
d-i debian-installer/locale             string en_US.UTF-8

d-i mirror/country                      string manual
d-i mirror/http/directory               string /ubuntu
d-i mirror/http/hostname                string mirror.servers.com
d-i mirror/http/proxy                   string

d-i apt-setup/services-select           multiselect security
d-i apt-setup/security_host             string mirror.servers.com
d-i apt-setup/security_path             string /ubuntu

d-i clock-setup/ntp                     boolean false
d-i clock-setup/ntp-server              string  time.servers.com
d-i clock-setup/utc                     boolean true

d-i time/zone string                    Europe/Moscow

d-i passwd/make-user                    boolean false
d-i passwd/root-login                   boolean true
d-i passwd/root-password-crypted        password
$6$$3NOl9aqkbtsiRGbEd8gPkU5D1jNrpeZcG8lfil/7tkgTO9v05.CnTaeoqEQdJ9FJ6yIlw2VIXoFm2MkuqYN9f0

d-i partman-auto/choose_recipe          select  boot-root

d-i partman-auto/disk                   string /dev/sda /dev/sdb
d-i partman-auto/method                 string raid

d-i partman-auto/purge_lvm_from_device  boolean true
d-i partman-auto-lvm/new_vg_name        string  U1521308965I0
d-i partman-auto-lvm/guided_size        string  max

d-i partman-lvm/device_remove_lvm       boolean true
d-i partman-lvm/device_remove_lvm_span  boolean true
d-i partman-lvm/confirm                 boolean true
d-i partman-lvm/confirm_nooverwrite     boolean true

d-i partman-md/device_remove_md         boolean true
d-i mdadm/boot_degraded                 boolean false
d-i partman-md/confirm                  boolean true
d-i partman-md/confirm_nooverwrite      boolean true

d-i partman/mount_style                 select  uuid
d-i partman/confirm_write_new_label     boolean true
d-i partman-partitioning/confirm_write_new_label    boolean true
d-i partman/choose_partition            select  finish
d-i partman/confirm                     boolean true
d-i partman/confirm_nooverwrite         boolean true

d-i partman-basicmethods/method_only    boolean false


# http://anonscm.debian.org/gitweb/?p=d-i/debian-installer.git;a=blob;f=doc/devel/partman-auto-
recipe.txt
d-i partman-auto/expert_recipe string \
  boot-root :: \
    1 1 1 free $primary{ } $bios_boot{ } method{ biosgrub } . \
    500 500 500 raid \
      $lvmignore{ } \
      $primary{ } \
      $bootable{ } \
      method{ raid } \
    . \
    1 1 -1 raid \
      $lvmignore{ } \
      $primary{ } \
      method{ raid } \
    . \
    2048 2048 2048 linux-swap \
```

```
      $defaultignore{ } \
      $lvmok{ } \
      lv_name{ swap } \
      method{ swap } \
      format{ } \
    . \
    10240 10240 10240 ext4 \
      $defaultignore{ } \
      $lvmok{ } \
      lv_name{ rootfs } \
      method{ format } \
      format{ } \
      use_filesystem{ } \
      filesystem{ ext4 } \
      mountpoint{ / } \
    . \
    1 1 -1 ext4 \
      $defaultignore{ } \
      $lvmok{ } \
      lv_name{ home } \
      method{ format } \
      format{ } \
      use_filesystem{ } \
      filesystem{ ext4 } \
      mountpoint{ /home } \
    .
d-i partman-auto-raid/recipe string \
    1 2 0 ext4 /boot \
          /dev/sda2#/dev/sdb2 \
    . \
    1 2 0 lvm - \
          /dev/sda3#/dev/sdb3 \
    .

d-i partman/early_command string sh -c ' \
    wget -q -O /dev/null
"http://pxe.servers.com/status/20:47:47:8c:c8:5e/notify?status=520&task_id=1886" ; \
    set -- $(vgs --rows --noheadings | head -n 1); \
    for vg in "$@"; do \
        vgremove -f "$vg"; \
    done; \
    set -- $(pvs --rows --noheadings | head -n 1); \
    for pv in "$@"; do \
      pvremove -f "$pv"; \
    done; \
    set -- $(cat /proc/mdstat | grep md | grep "sda\|sdb" | cut -d \  -f 1); \
    for md in "$@"; do \
        echo clear > /sys/block/"$md"/md/array_state; \
        /sbin/mdadm --stop /dev/"$md"; \
        /sbin/mdadm --remove /dev/"$md"; \
    done; \
  exit 0 \
'

d-i grub-installer/only_debian                       boolean true
d-i grub-installer/grub2_instead_of_grub_legacy     boolean true

d-i grub-installer/bootdev string /dev/sda /dev/sdb

tasksel tasksel/first multiselect none
popularity-contest popularity-contest/participate boolean false



d-i pkgsel/include string ssh joe ethtool dstat rcconf less exim4-base mdadm ifenslave
d-i pkgsel/upgrade select full-upgrade
d-i pkgsel/update-policy select none

d-i preseed/late_command string sh -c '(\
echo "bonding" >> /target/etc/modules; \
rm -rf /target/etc/udev/rules.d/70-persistent-net.rules; \
echo -e " \
SUBSYSTEM==\"net\",ACTION==\"add\",DRIVERS==\"?*\",ATTR{address}==\"20:47:47:8c:c8:5e\",ATTR{type}==
\"1\",KERNEL==\"eth*\",NAME=\"int1\"\n\
SUBSYSTEM==\"net\",ACTION==\"add\",DRIVERS==\"?*\",ATTR{address}==\"a0:36:9f:83:93:ec\",ATTR{type}==
\"1\",KERNEL==\"eth*\",NAME=\"int2\"\n\
SUBSYSTEM==\"net\",ACTION==\"add\",DRIVERS==\"?*\",ATTR{address}==\"20:47:47:8c:c8:60\",ATTR{type}==
\"1\",KERNEL==\"eth*\",NAME=\"ext1\"\n\
```

```
SUBSYSTEM==\"net\",ACTION==\"add\",DRIVERS==\"?*\",ATTR{address}==\"a0:36:9f:83:93:ed\",ATTR{type}==
\"1\",KERNEL==\"eth*\",NAME=\"ext2\"\n\
" > /target/etc/udev/rules.d/70-persistent-net.rules; \
echo -e "\
auto aggi\n\
iface aggi inet static \n\
    hwaddress ether 20:47:47:8C:C8:5E\n\
    address 10.107.4.172\n\
    netmask 255.255.255.248\n\

    mtu 1500\n\
    slaves int1 int2\n\
    bond_mode 4\n\
    bond_miimon 100\n\
    bond_downdelay 200\n\
    bond_updelay 200\n\
    bond_lacp_rate slow\n\
    bond_xmit_hash_policy layer3+4\n\
    post-up /sbin/ethtool -K aggi tx off tso off\n\
\n\
auto agge\n\
iface agge inet static \n\
    hwaddress ether 20:47:47:8C:C8:60\n\
    address 23.105.234.36\n\
    netmask 255.255.255.248\n\
    gateway 23.105.234.35\n\
    slaves ext1 ext2\n\
    bond_mode 4\n\
    bond_miimon 100\n\
    bond_downdelay 200\n\
    bond_updelay 200\n\
    bond_lacp_rate slow\n\
    bond_xmit_hash_policy layer3+4\n\
    post-up /sbin/ethtool -K agge tx off tso off\n\
\n\


up route add -net 10.0.0.0/8 gw 10.107.4.171 dev aggi\n\
up route add -net 192.168.0.0/16 gw 10.107.4.171 dev aggi\n\
up route add -net 188.42.208.0/21 gw 10.107.4.171 dev aggi\n\
\n\
auto int1\n\
iface int1 inet manual\n\
bond-master aggi\n\
\n\
auto int2\n\
iface int2 inet manual\n\
bond-master aggi\n\
\n\
auto ext1\n\
iface ext1 inet manual\n\
bond-master agge\n\
\n\
auto ext2\n\
iface ext2 inet manual\n\
bond-master agge\n\
\n\
auto lo\n\
iface lo inet loopback\n\
" > /target/etc/network/interfaces; \
cp /target/etc/network/interfaces /etc/network/interfaces; \
echo -e "\
options rotate timeout:1 \n\
nameserver 192.168.8.88\n\
nameserver 192.168.8.8\n\
" > /target/etc/resolv.conf; \
cat /target/etc/resolv.conf >> /target/etc/resolvconf/resolv.conf.d/head;\
sed -i -e "s/^ *//" /target/etc/resolvconf/resolv.conf.d/head;\
echo "#" > /target/etc/udev/rules.d/75-persistent-net-generator.rules; \
echo "*/30 * * * * root ntpdate -b time.servers.com > /dev/null 2>&1" >> /target/etc/crontab; \
sed -ie s/"^PermitRootLogin prohibit-password"/"PermitRootLogin yes"/g /target/etc/ssh/sshd_config;
\
sed -ie "s/GRUB_CMDLINE_LINUX_DEFAULT=.*/GRUB_CMDLINE_LINUX_DEFAULT=\"net.ifnames=0\"/g"
/target/etc/default/grub; \
in-target update-grub; \
rm -rf /target/var/log/installer/*; \
/usr/bin/wget -q -O /dev/null
"http://pxe.servers.com/status/20:47:47:8c:c8:5e/finalize?status=590&task_id=1886"; \
exit 0 \
```

```
)'

d-i netcfg/target_network_config select ifupdown

d-i finish-install/keep-consoles      boolean false
d-i finish-install/reboot_in_progress note

d-i cdrom-detect/eject                boolean false

d-i debian-installer/exit/halt        boolean false
d-i debian-installer/exit/poweroff    boolean false
```