Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут телекомунікаційних систем
(повна назва інституту/факультету)

Кафедра телекомунікацій
(повна назва кафедри)

«На правах рукопису»
УДК 621.39

До захисту допущено
В.о. завідувача кафедри

_____ Явіся В.С.___
(підпис) (ініціали, прізвище)
“___”_____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 172 Телекомунікації та радіотехніка,
(код і назва)

спеціалізація Апаратно-програмні засоби електронних комунікацій

на тему: «Аналіз ефективності використання технологій віртуальних машин і віртуальних контейнерів в системах хмарних сервісів»

Виконав: студент 6 курсу, групи ТЗ-61м
(шифр групи)

Омельченко Руслан Юрійович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник професор каф. телекомунікацій, д.т.н., проф. Романов О.І.
(посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Рецензент доцент каф. телеком систем Бердников О.М.
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали)

_____
(підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент ___
(підпис)

Київ – 2018 рік

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут телекомунікаційних систем
(повна назва)

Кафедра телекомунікацій
(повна назва)

Рівень вищої освіти – другий (магістерський)

Спеціальність 172 Телекомунікації та радіотехніка
(код і назва)

Спеціалізація Апаратно-програмні засоби електронних комунікацій

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри

_____ Явіся В.С.
(підпис) (ініціали, прізвище)

«___»_____2017 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Омельченку Руслану Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Аналіз ефективності використання технологій віртуальних машин і віртуальних контейнерів в системах хмарних сервісів

науковий керівник дисертації професор, д.т.н. Романов О.І. ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « 06 » « 04 » 2018р. № 1105-с

2. Строк подання студентом дисертації « 18 » травня 2018р.

3. Об'єкт дослідження системи віртуалізації фізичних серверів

4. Предмет дослідження технології апаратної та контейнерної віртуалізації

5. Перелік завдань, які потрібно розробити
   - аналіз технології віртуалізації
   - аналіз типів віртуалізації обладнання та розгляд архітектури кожного типу
   - аналіз особливостей віртуалізації мереж
   - розгляд переваг та недоліків систем віртуалізації

- аналіз архітектурних особливостей віртуальних машин, огляд рішень від основних компаній-розробників
- аналіз архітектурних особливостей віртуальних контейнерів, огляд рішень від основних компаній-розробників
- розгляд основних відмінностей між віртуальними машинами та віртуальними контейнерами
- створення макетів віртуального контейнера на базі програмного продукту Docker та віртуальної машини на базі Hyper-V, аналіз особливостей роботи в умовах хмарного середовища.

6. Орієнтовний перелік ілюстративного матеріалу

_____
_____
_____
_____
_____
_____
_____
_____

7. Орієнтовний перелік публікацій

Гордашник Є. Аналіз ефективності технології контейнерної віртуалізації / Є. Гордашник, Р. Омельченко. // Збірник тез одинадцятої міжнародної науково-технічної конференції "ПРОБЛЕМИ ТЕЛЕКОМУНІКАЦІЙ". – 2017. – С. 139–141.

Верес Л. Аналіз архітектурних особливостей віртуалізації та переваги віртуальних контейнерів / Л. Верес, Р. Омельченко. // Збірник тез дванадцятої міжнародної науково-технічної конференції "ПРОБЛЕМИ ТЕЛЕКОМУНІКАЦІЙ". – 2018. – С. 116–118.

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання 10.09.2016

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Розробка, оформлення, узгодження та затвердження технічного завдання на дипломну роботу	10.09.2016	Виконано
2	Опрацювання літературних джерел з теми досліджень	01.03.2017	Виконано
3	Аналіз вимог завдання, вибір методів і засобів розв’язання поставленої задачі	01.04.2017	Виконано
4	Детальний аналіз технології віртуалізації	01.06.2017	Виконано
5	Детальний аналіз архітектурних особливостей апаратної віртуалізації	01.08.2017	Виконано
6	Детальний аналіз архітектурних особливостей контейнерної віртуалізації	01.10.2017	Виконано
7	Порівняльна характеристика систем балансування навантаження	01.12.2017	Виконано
8	Створення макету хмарного сервісу в системах Docker та Hyper-V	01.02.2018	Виконано
9	Оформлення магістерської дисертації	01.05.2018	Виконано

Студент _____ Омельченко Р.Ю.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Романов О.І.
(підпис) (ініціали, прізвище)

Пояснювальна записка
до магістерської дисертації

на тему: Аналіз ефективності використання технологій віртуальних машин і віртуальних контейнерів в системах хмарних сервісів

Київ – 2018 року

РЕФЕРАТ

на магістерську дисертацію

виконану на тему: «Аналіз ефективності використання технологій віртуальних машин і віртуальних контейнерів в системах хмарних сервісів»

Пояснювальна записка виконана на 108 сторінках та включає 58 ілюстрацій, та 16 джерел за переліком посилань.

У зв'язку з новітніми тенденціями в галузі інформаційних систем, все більша кількість сервісів в Інтернеті стають хмарними. Це означає, що послуги можуть надаватись користувачам віддалено, без необхідності встановлення серверного обладнання в себе. Технологія віртуалізації дозволяє значно покращити процес побудови хмарних сервісів, оскільки зменшує кількість фізичного обладнання, вартість на купівлю і обслуговування якого може бути досить високою. Оскільки технологія віртуалізації вже набула значного поширення і є кілька основних її типів, постає питання вибору найбільш підходящого для ефективної роботи хмарного сервісу. Два основні типи – апаратна та контейнерна віртуалізація, мають як спільні так і відмінні риси, що дозволяє використовувати певні їхні характеристики для забезпечення надійності мережі. Тому вибір підходящого методу віртуалізації при розробці хмарного сервісу має важливе значення.

Метою магістерської роботи є проведення аналізу характеристик віртуальних машин та віртуальних контейнерів, порівняння обох технологій між собою, та налаштування моделей кожного з досліджуваних типів віртуальних середовищ в системах Hyper-V та Docker. Для досягнення цієї мети, визначено наступні завдання:

– Проаналізувати технологію віртуалізації;

- Провести дослідження архітектури віртуальних машин;
- Дослідити архітектуру віртуальних контейнерів;
- Побудувати макети віртуальної машини та контейнера і проаналізувати ефективність використання кожного з цих рішень при побудові хмарного сервісу.

Об’єктом дослідження виступають системи віртуалізації фізичних серверів. Предметом дослідження – технології апаратної віртуалізації та технології контейнерної віртуалізації.

Методи досліджень, які використовуються в магістерській дисертації:

- Аналіз технологій контейнерної та апаратної віртуалізації;
- Порівняння особливостей віртуальних машин і контейнерів;
- Дослідження переваг та недоліків кожного типу віртуалізації
- Експериментальна побудова макету хмарного файлообмінника на базі віртуальної машини Hyper-V;
- Експериментальна побудова макету хмарного файлообмінника на базі віртуального контейнера Docker;
- Порівняння особливостей роботи макетів на базі кожного типу віртуалізації.

Наукова новизна роботи полягає в проведенні аналізу роботи технологій контейнерної та апаратної віртуалізації в якості складових при розгортанні хмарного сервісу, а також порівняння обох технологій з метою вибору більш ефективної.

Результати досліджень, які приводяться в магістерській дисертації, можуть бути використані компаніями при побудові власного корпоративного хмарного середовища, в навчальних дисциплінах при розгляді технологій віртуалізації та хмарних обчислень, а також Інститутом телекомунікаційних систем при побудові власного хмарного сервісу.

Результати досліджень оприлюднені у двох публікаціях на Міжнародній науково-технічній конференції «Проблеми телекомунікацій» у 2017 та 2018 роках:

Омельченко Р. Аналіз ефективності технології контейнерної віртуалізації / Є. Гордашник, Р. Омельченко. // Збірник тез одинадцятої міжнародної науково-технічної конференції "ПРОБЛЕМИ ТЕЛЕКОМУНІКАЦІЙ". – 2017. – С. 139–141.

Омельченко Р. Аналіз архітектурних особливостей віртуалізації та переваги віртуальних контейнерів / Л. Верес, Р. Омельченко. // Збірник тез дванадцятої міжнародної науково-технічної конференції "ПРОБЛЕМИ ТЕЛЕКОМУНІКАЦІЙ". – 2018. – С. 116–118.

Ключові слова: віртуалізація, віртуальна машина, віртуальний контейнер, Hyper-V, Docker.

ABSTRACT

on master's thesis

on topic: Analysis of efficiency of virtual machines and virtual containers technologies in cloud service systems.

The explanatory note is made on 108 pages and includes 58 illustrations, and 16 sources by the list of references.

Due to the latest trends in information systems, an increasing number of services on the Internet are becoming cloudy. This means that services can be provided to users remotely, without the need to install server hardware in themselves. Virtualization technology can significantly improve the process of building cloud services, because it reduces the amount of physical equipment, the cost of purchasing and maintenance of which can be quite high. As the virtualization technology has already become widespread and there are several basic types, it raises the question of choosing the most suitable cloud service for effective operation. The two main types are hardware and container virtualization, which have both common and distinctive features, which allows you to use certain characteristics to ensure network reliability. Therefore, the choice of a suitable virtualization method when developing a cloud service is important.

The aim of the master's thesis is to analyze the characteristics of virtual machines and virtual containers, compare both technologies with each other, and configure the models of each of the types of virtual environments studied in Hyper-V and Docker systems. To achieve this goal, the following tasks are defined:

- To analyze virtualization technology;
- Conduct research on the architecture of virtual machines;
- Explore the architecture of virtual containers;

– Build virtual machine and container layouts and analyze the effectiveness of each of these solutions when building a cloud service.

The object of research is virtual server virtualization systems. The subject of the study is technology of hardware virtualization and technology of container virtualization.

Methods of research used in the master's thesis:

– Analysis of container and hardware virtualization technologies;
– Comparison of features of virtual machines and containers;
– Study the advantages and disadvantages of each type of virtualization
– Experimental construction of the layout of the cloud file-changer based on the Hyper-V virtual machine;
– Experimental layout design of a cloud file-changer based on the virtual container Docker;
– Comparison of the features of the layout work based on each type of virtualization.

The scientific novelty of the work is to conduct analysis of the technology of container and hardware virtualization as components in the deployment of cloud service, as well as comparison of both technologies in order to choose more efficient.

The results of the research, which are given in the master's thesis, can be used by companies in the construction of their own corporate cloud environment, in academic disciplines when considering the technologies of virtualization and cloud computing, as well as the Institute of Telecommunication Systems in the construction of its own cloud service.

The results of the research were published in two publications at the International Scientific and Technical Conference " Modern Challenges in Telecommunications " in 2017 and 2018:

Омельченко Р. Аналіз ефективності технології контейнерної віртуалізації / Є. Гордашник, Р. Омельченко. // Збірник тез одинадцятої міжнародної науково-технічної конференції "ПРОБЛЕМИ ТЕЛЕКОМУНІКАЦІЙ". – 2017. – Р. 139–141.

Омельченко Р. Аналіз архітектурних особливостей віртуалізації та переваги віртуальних контейнерів / Л. Верес, Р. Омельченко. // Збірник тез дванадцятої міжнародної науково-технічної конференції "ПРОБЛЕМИ ТЕЛЕКОМУНІКАЦІЙ". – 2018. – Р. 116–118.

Keywords: virtualization, virtual machine, virtual container, Hyper-V, Docker.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

Англійські скорочення

ADC	(Application delivery controller) – контролер доставки додатків
AMD	(Advanced Micro Devices) – передові мікропристрої
API	(Application Programming Interface) – прикладний програмний інтерфейс
BGP	(Border Gateway Protocol) – протокол граничного шлюза
CPU	(Central Processing Unit) – центральний процесорний пристрій
CSV	(Cluster Shared Volumes) – спільний обсяг кластера
DevOps	(development and operations) – розробка та операції
DHCP	(Dynamic Host Configuration Protocol) - протокол динамічної конфігурації вузла
HDD	(Hard Disk Drive) – жорсткий диск
IBM	(International Business Machines) – міжнародні бізнес-машини
IEEE	(Institute of Electrical and Electronics Engineers) - міжнародна організація інженерів у галузі електротехніки, радіоелектроніки та радіоелектронної промисловості
IIS	(Internet Information Services) – інформаційні сервіси Інтернету
INI	(Initialization file) – файл ініціалізації
IP	(Internet Protocol) – інтернет протокол
IT	(Information Technology) – інформаційні технології
KVM	(Kernel-based Virtual Machine) – віртуальна машина на базі Kernel
LAN	(Local Area Network) – локальна комп’ютерна мережа
LXC	(LinuX Containers) – контейнери Linux
MAC	(Media Access Control) - управління доступом до носія
MS	(Microsoft) - Майкрософт
MySQL	(My Structured Query Language) – моя структурована мова запитів
NAT	(Network Address Translation) – перетворення мережевих адрес
NEC	(Nippon Electric Corporation) - японська корпорація, виробник електронної, комп'ютерної техніки, телекомунікаційного

	устаткування, одна з найбільших світових телекомунікаційних компаній
NFV	(Network Functions Virtualization) – віртуалізація мережевих функцій
NIC	(network interface controller) – мережевий адаптер
ONF	(Open Networking Foundation) – відкритий мережевий фонд
OpenVZ	(Open Virtuozzo) - технологія віртуалізації на рівні ОС, яка базується на ядрі Linux.
OSPF	(Open Shortest Path First) - протокол динамічної маршрутизації, заснований на технології відстеження стану каналу
PC	(Personal computer) – персональний комп’ютер
PHP	(Hypertext Preprocessor) – препроцесор гіпертексту
RDP	(Remote Desktop Protocol) – протокол віддаленого робочого стола
REST	(Representational State Transfer) - передача стану представлення
RHEL	(Red Hat Enterprise Linux) - дистрибутив Linux виробництва компанії Red Hat
SCSI	(Small Computer Systems Interface) - інтерфейс, розроблений для об'єднання на одній шині різних за своїм призначенням пристроїв
SDN	(software-defined networking) - мережа передачі даних, в якій рівень управління мережею відділений від пристроїв передачі даних і реалізується програмно
SELinux	(Security-Enhanced Linux) - Linux з покращеним рівнем безпеки
SP3	(Service Pack 3) – сервіс пак 3
TCP	(Transmission Control Protocol) – протокол управління передачею
UDP	(User Datagram Protocol) – протокол датаграм користувача
USB	(Universal Serial Bus) - універсальна послідовна шина, призначена для з'єднання периферійних пристроїв обчислювальної техніки
VDI	(Virtual Desktop Infrastructure) – інфраструктура віртуального робочого місця

VLAN (Virtual Local Area Network) – віртуальна локальна комп’ютерна мережа

VMI (Virtual Machine Interface) – інтерфейс віртуальної машини

vNIC (virtual network interface controller) – віртуальний мережевий адаптер

WAN (Wide Area Network) – глобальна комп’ютерна мережа

WDDM (Windows Display Driver Model) - архітектура графічних драйверів для відеокарти під керуванням Microsoft Windows

Українські скорочення

ВМ віртуальна машина

ЕОМ електронна обчислювальна машина

ОЗП оперативний запам’ятовуючий пристрій

ОС операційна система

ПЗ програмне забезпечення

ПК персональний комп’ютер

ЦОД центр обробки даних

ЦП центральний процесор

ВСТУП

На даному етапі розвитку обчислювальних мереж все більш широко використовується перехід від горизонтального до вертикального масштабування. Одним із аспектів цього принципу є зменшення кількості фізичних серверів, які замінюються їх віртуальними аналогами. У зв'язку з цим, технологія віртуалізації фізичного обладнання вийшла на перші ролі в питаннях побудови інформаційно-комунікаційних мереж. Суть технології віртуалізації полягає в можливості розміщення декількох ізольованих середовищ на одному фізичному сервері, що дозволяє зменшити кількість фізичного обладнання і заощадити матеріальні ресурси.

На сьогоднішній день основними типами віртуалізації є апаратна та контейнерна. У зв'язку із все більшим поширенням хмарних сервісів, є доцільним використовувати технології віртуалізації в корпоративних та публічних хмарах. Тому постає питання вибору того типу віртуалізації, який буде більш доречним при побудові певного хмарного сервісу. При цьому слід врахувати такі параметри, як швидкодія віртуального середовища, навантаження на систему тощо, простота розгортання тощо.

Метою магістерської роботи є проведення аналізу характеристик технологій апаратної та контейнерної віртуалізації, задля вибору підходящого рішення для впровадження в хмарному середовищі.

Для досягнення мети дослідження необхідно вирішити такі основні задачі:

1. Провести аналіз особливостей технології віртуалізації фізичного обладнання.
2. Дослідити архітектуру віртуальних машин і виконати огляд рішень основних розробників у сфері апаратної віртуалізації.
3. Дослідити архітектуру віртуальних контейнерів та виконати огляд рішень основних розробників у сфері контейнерної віртуалізації.
4. Розглянути основні відмінності між технологіями апаратної та контейнерної віртуалізації.

5. Використовуючи макети віртуальної машини та віртуального контейнера, проаналізувати придатність їх використання в системі хмарних сервісів.

Об’єктом дослідження виступають системи віртуалізації фізичної інфраструктури, в тому числі серверів, а предметом дослідження – технології апаратної віртуалізації та технології контейнерної віртуалізації.

РОЗДІЛ 1
ВІРТУАЛІЗАЦІЯ ЯК СПОСІБ ОРГАНІЗАЦІЇ ІНФРАСТРУКТУРИ ІНФОРМАЦІЙНО-ТЕЛЕКОМУНІКАЦІЙНИХ МЕРЕЖ

1.1 Поняття віртуалізації. Існуючі типи організації віртуальних середовищ

Згідно зі статистикою [1], середній рівень завантаження процесорних потужностей у серверів під управлінням Windows не перевищує 10%, у Unix-систем цей показник краще, але, тим не менше, в середньому не перевищує 20%. Низька ефективність використання серверів пояснюється широко застосовуваним з початку 90-х років підходом "один додаток - один сервер", тобто кожного разу для розгортання нової програми компанія купує новий сервер. Очевидно, що на практиці це означає швидке збільшення серверного парку і, як наслідок - зростання витрат на його адміністрування, енергоспоживання та охолодження, а також потреби в додаткових приміщеннях для установки все нових серверів і придбання ліцензій на серверну ОС.

Віртуалізація ресурсів фізичного сервера дозволяє гнучко розподіляти їх між додатками, кожен з яких при цьому "бачить" тільки призначені йому ресурси і "вважає", що йому виділено окремий сервер, тобто в даному випадку реалізується підхід "один сервер - кілька додатків", але без зниження продуктивності, доступності та безпеки серверних додатків. Крім того, рішення віртуалізації дають можливість запускати в розділах різні ОС за допомогою емуляції їх системних викликів до апаратних ресурсів сервера.

В основі віртуалізації лежить можливість одного комп'ютера виконувати роботу декількох комп'ютерів завдяки розподілу його ресурсів за кількома середовищами. За допомогою віртуальних серверів і віртуальних настільних комп'ютерів можна розмістити кілька ОС і кілька додатків в єдиному місці розташування. Таким чином, фізичні і географічні обмеження перестають мати будь-яке значення. Крім енергозбереження та скорочення витрат завдяки більш

ефективному використанню апаратних ресурсів, віртуальна інфраструктура забезпечує високий рівень доступності ресурсів, більш ефективну систему управління, підвищену безпеку і вдосконалену систему відновлення в критичних ситуаціях.

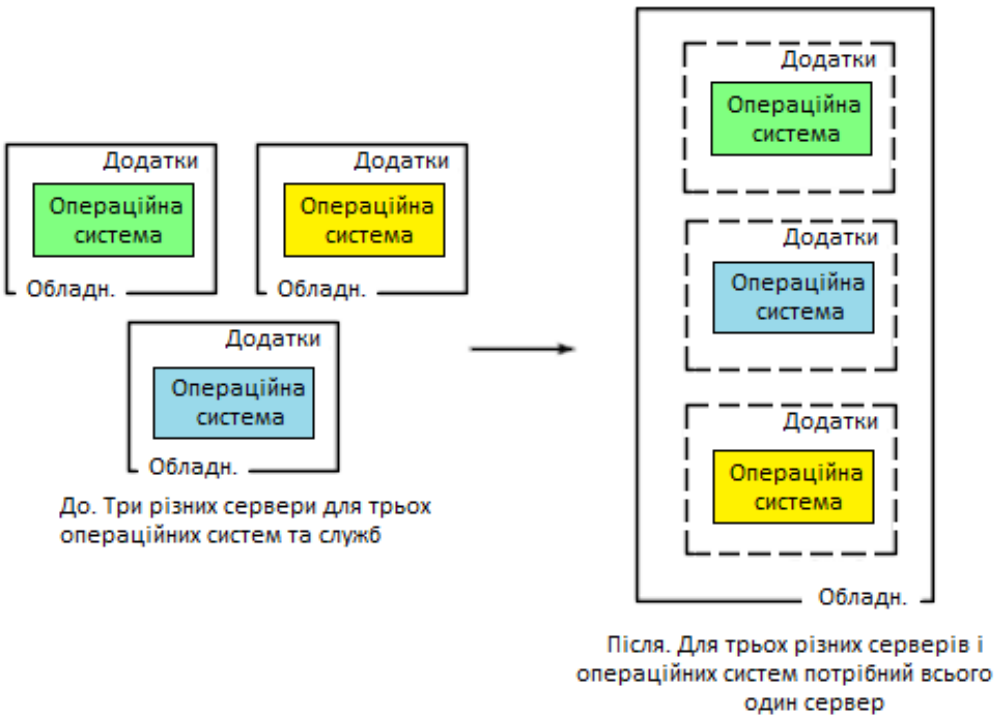


Рис. 1.1 Віртуалізація – перехід від фізичного відокремлення серверів до логічного

У широкому сенсі, поняття віртуалізації являє собою приховування справжньої реалізації будь-якого процесу або об'єкта від істинного його представлення для того, хто ним користується. Продуктом віртуалізації є щось зручне для використання і яке насправді, має більш складну або зовсім іншу структуру, відмінну від тієї, яка сприймається при роботі з об'єктом. Іншими словами, відбувається відділення представлення від реалізації чого-небудь. Віртуалізація покликана абстрагувати програмне забезпечення від апаратної частини.

У комп'ютерних технологіях під терміном "віртуалізація" зазвичай розуміється абстракція обчислювальних ресурсів і надання користувачеві системи, яка "інкапсулює" (приховує в собі) власну реалізацію. Простіше

кажучи, користувач працює із зручним для себе поданням об'єкта, і для нього не має значення, як об'єкт влаштований в дійсності.

Зараз можливість запуску декількох віртуальних машин на одній фізичній викликає великий інтерес серед комп'ютерних фахівців, не тільки тому, що це підвищує гнучкість IT-інфраструктури, а й тому, що віртуалізація, насправді, дозволяє економити гроші.

Історія розвитку технологій віртуалізації налічує понад сорок років. Компанія IBM була першою, хто задумався про створення віртуальних середовищ для різних призначених для користувача завдань, тоді ще в мейнфреймах. У 60-х роках минулого століття віртуалізація представляла чисто науковий інтерес і була оригінальним рішенням для ізоляції комп'ютерних систем в рамках одного фізичного комп'ютера. Після появи персональних комп'ютерів інтерес до віртуалізації дещо послабився через бурхливого розвитку операційних систем, які пред'являли адекватні вимоги до апаратного забезпечення того часу. Однак бурхливе зростання апаратних потужностей комп'ютерів в кінці дев'яностих років минулого століття змусив IT-спільнота знову згадати про технології віртуалізації програмних платформ.

У 1999 р компанія VMware представила технологію віртуалізації систем на базі x86 в якості ефективного засобу, здатного перетворити системи на базі x86 в єдину апаратну інфраструктуру загального користування та призначення, що забезпечує повну ізоляцію, мобільність і широкий вибір ОС для прикладних середовищ. Компанія VMware була однією з перших, хто зробив серйозну ставку виключно на віртуалізацію. Як показав час, це виявилося абсолютно виправданим. Сьогодні VMware пропонує комплексну віртуалізаційну платформу четвертого покоління VMware vSphere 4, яка включає засоби як для окремого ПК, так і для центру обробки даних. Ключовим компонентом цього програмного комплексу є гіпервізор VMware ESX Server. Пізніше в "битву" за місце в цьому модному напрямку розвитку інформаційних технологій включилися такі компанії як Parallels (раніше SWsoft), Oracle (Sun Microsystems), Citrix Systems (XenSourse).

Корпорація Microsoft вийшла на ринок засобів віртуалізації в 2003 році з придбанням компанії Connectix, випустивши свій перший продукт Virtual PC для настільних ПК. З тих пір вона послідовно нарощувала спектр пропозицій в цій галузі і на сьогодні майже завершила формування віртуалізаційної платформи, до складу якої входять такі рішення як Windows 2008 Server R2 з компонентом Hyper-V, Microsoft Application Virtualization (App-v), Microsoft Virtual Desktop Infrastructure (VDI), Remote Desktop Services, System Center Virtual Machine Manager.

Підвищений інтерес до технологій віртуалізації в даний час не випадковий. Обчислювальна потужність нинішніх процесорів швидко зростає, і питання навіть не в тому, на що цю міць витрачати, а в тому, що сучасна "мода" на двоядерні і багатоядерні системи, що проникла вже і в персональні комп'ютери (ноутбуки та десктопи), як не можна краще дозволяє реалізувати багатющий потенціал ідей віртуалізації операційних систем і додатків, виводячи зручність користування комп'ютером на новий якісний рівень. Технології віртуалізації стають одним з ключових компонентів (в тому числі, і маркетингових) в найновіших і майбутніх процесорах Intel і AMD, в операційних системах від Microsoft і ряду інших компаній.

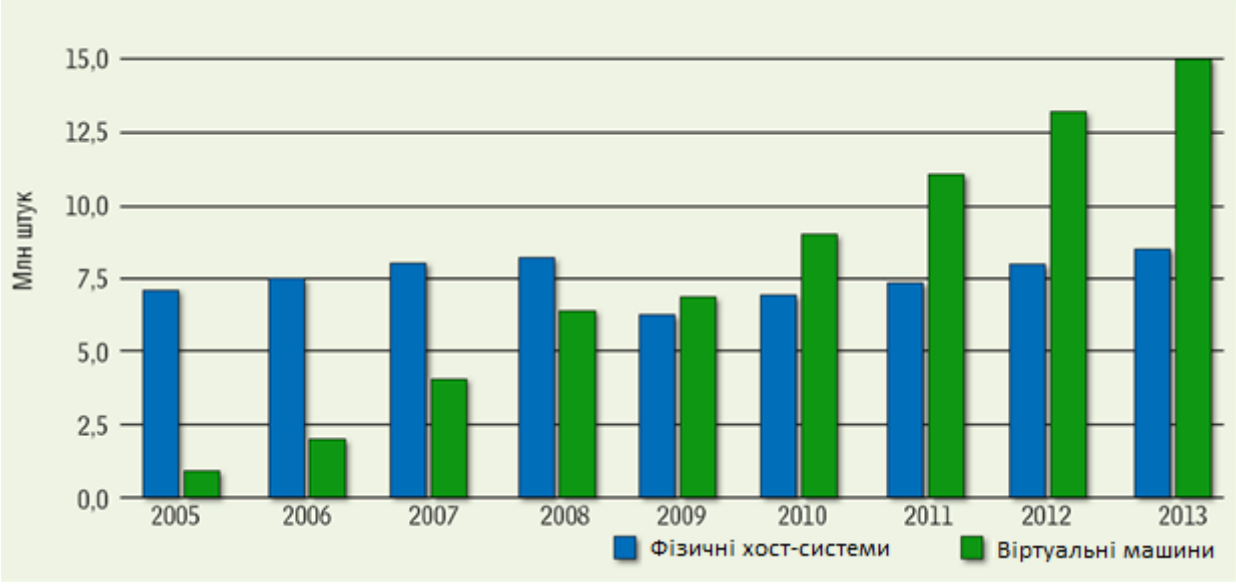


Рис. 1.2 Співвідношення кількості віртуальних і фізичних серверів

Віртуальною машиною будемо називати програмне або апаратне середовище, яке приховує справжню реалізацію будь-якого процесу або об'єкта від його видимого уявлення.

Віртуальна машина - це повністю ізольований програмний контейнер, який працює з власною ОС і додатками, подібно фізичному комп'ютеру. Віртуальна машина діє так само, як фізичний комп'ютер, і містить власні віртуальні (тобто програмні) ОЗП, жорсткий диск і мережевий адаптер.

ОС не може розрізнити віртуальну і фізичну машини. Те ж саме можна сказати про додатки та інших комп'ютерах в мережі. Навіть сама віртуальна машина вважає себе "справжнім" комп'ютером. Але незважаючи на це віртуальні машини складаються виключно з програмних компонентів і не включають обладнання. Це дає їм ряд унікальних переваг над фізичним обладнанням.

Розглянемо основні особливості віртуальних машин більш детально:

1. **Сумісність**. Віртуальні машини, як правило, сумісні з усіма стандартними комп'ютерами. Як і фізичний комп'ютер, віртуальна машина працює під управлінням власної гостьової операційної системи і виконує власні додатки. Вона також містить всі компоненти, стандартні для фізичного комп'ютера (материнську плату, відеокарту, мережевий контролер і т. д.). Тому віртуальні машини повністю сумісні з усіма стандартними операційними системами, програмами та драйверами пристроїв. Віртуальну машину можна використовувати для виконання будь-якого програмного забезпечення, придатного для відповідного фізичного комп'ютера.
2. **Ізольованість**. Віртуальні машини повністю ізольовані один від одного, як якщо б вони були фізичними комп'ютерами. Віртуальні машини можуть використовувати загальні фізичні ресурси одного комп'ютера і при цьому залишатися повністю ізольованими один від одного, як якщо б вони були окремими фізичними машинами. Наприклад, якщо на одному фізичному сервері запущено чотири віртуальних машини, і одна з них дає

збій, це не впливає на доступність інших трьох машин. Ізольованість - важлива причина набагато більш високої доступності та безпеки додатків, які виконуються в віртуальному середовищі, в порівнянні з додатками, виконуваними в стандартній, невіртуалізірованій системі.

3. **Інкапсуляція**. Віртуальні машини повністю інкапсулюють обчислювальну середу. Віртуальна машина являє собою програмний контейнер, що зв'язує, або "інкапсулює" повний комплект віртуальних апаратних ресурсів, а також ОС і все її застосування в програмному пакеті. Завдяки інкапсуляції віртуальні машини стають неймовірно мобільними і зручними в управлінні. Наприклад, віртуальну машину можна перемістити або скопіювати з одного місцеположення до іншого так само, як будь-який інший програмний файл. Крім того, віртуальну машину можна зберегти на будь-якому стандартному носії даних: від компактної карти Flash-пам'яті USB до корпоративних мереж зберігання даних.
4. **Незалежність від обладнання**. Віртуальні машини повністю незалежні від базового фізичного обладнання, на якому вони працюють. Наприклад, для віртуальної машини з віртуальними компонентами (ЦП, мережевою картою, контролером SCSI) можна задати налаштування, що абсолютно не збігаються з фізичними характеристиками базового апаратного забезпечення. Віртуальні машини можуть навіть виконувати різні операційні системи (Windows, Linux і ін.) на одному і тому ж фізичному сервері. У поєднанні з властивостями інкапсуляції і сумісності, апаратна незалежність забезпечує можливість вільно переміщувати віртуальні машини з одного комп'ютера на базі x86 на інший, не змінюючи драйвери пристроїв, ОС або додатки. Незалежність від обладнання також дає можливість запускати в поєднанні абсолютно різні ОС і додатки на одному фізичному комп'ютері.

Розглянемо основні різновиди віртуалізації, такі як:

- віртуалізація серверів (повна віртуалізація і паравіртуалізація);
- віртуалізація на рівні операційних систем;

– віртуалізація додатків;
– віртуалізація представлень.

1.1.1 Різновиди апаратної віртуалізації

Віртуалізація серверів

Сьогодні, говорячи про технології віртуалізації, як правило, мають на увазі віртуалізацію серверів, так як остання стає найбільш популярним рішенням на ринку IT. Віртуалізація серверів має на увазі запуск на одному фізичному сервері декількох віртуальних серверів. Віртуальні машини або сервери є програми, запущені на хостовой операційній системі, які емулюють фізичні пристрої сервера. На кожній віртуальній машині може бути встановлена операційна система, на яку можуть бути встановлені додатки і служби. Типові представники - це продукти VmWare (ESX, Server, Workstation) і Microsoft (Hyper-V, Virtual Server, Virtual PC).

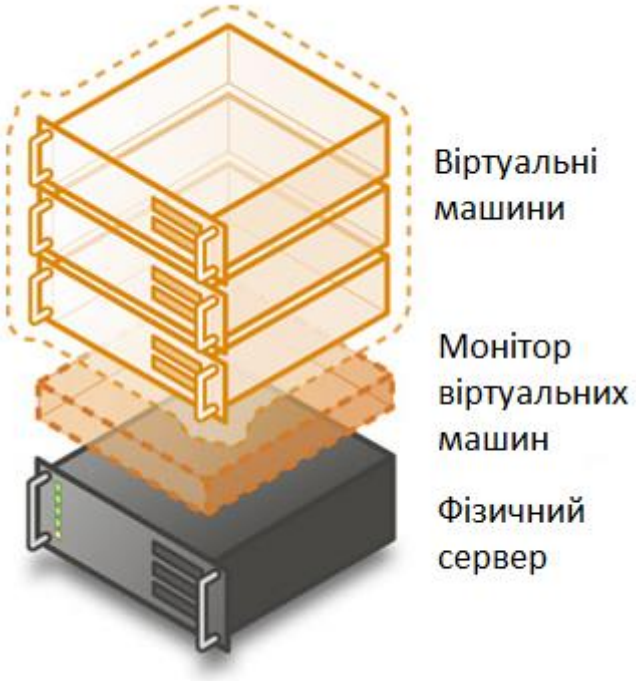


Рис. 1.3 Представлення віртуалізації серверів

Центри обробки даних використовують великий простір і величезну кількість енергії, особливо якщо додати до цього системи охолодження і

інфраструктуру, що їх супроводжують. Засобами технологій віртуалізації виконується консолідація серверів, розташованих на великій кількості фізичних серверів у вигляді віртуальних машин на одному високопродуктивному сервері. Число фізичних машин, необхідних для роботи в якості серверів зменшується, що знижує кількість енергії, необхідної для роботи машин і простір, необхідний для їх розміщення. Скорочення кількості серверів і простору зменшує кількість енергії, необхідної для їх охолодження. При меншій витраті енергії виробляється менша кількість вуглекислого газу. Даний показник, наприклад, в Європі, має досить важливу роль.

Важливим фактором є фінансова сторона. Віртуалізація є важливим моментом економії. Віртуалізація не тільки зменшує потребу в придбанні додаткових фізичних серверів, але і мінімізує вимоги до їх розміщення. Використання віртуального сервера надає переваги по швидкості впровадження, використання і управління, що дозволяє зменшити час очікування розгортання будь-якого проекту.

Не так давно з'явилися моделі останнього покоління процесорів в архітектурі x86 корпорацій AMD і Intel, де виробники вперше додали технології апаратної підтримки віртуалізації. До цього віртуалізація підтримувалася програмно, що природно приводила до великих накладних витрат продуктивності.

Віртуалізація для серверної інфраструктури стала застосовуватися трохи пізніше, і пов'язано це було, перш за все, з вирішенням завдань консолідації обчислювальних ресурсів. Але тут відразу сформувалося два незалежних напрямки:

– підтримка неоднорідних операційних середовищ (в тому числі, для роботи успадкованих додатків). Цей випадок найбільш часто зустрічається в рамках корпоративних інформаційних систем. Технічно проблема вирішується шляхом одночасної роботи на одному комп'ютері декількох віртуальних машин, кожна з яких включає екземпляр операційної системи. Але реалізація цього

режиму виконувалася за допомогою двох принципово різних підходів: повної віртуалізації і паравіртуалізації;

– підтримка однорідних обчислювальних середовищ на увазі ізоляцію служб в рамках одного примірника ядра операційної системи (віртуалізація на рівні ОС), що найбільш характерно для хостингу додатків провайдерами послуг. Звичайно, тут можна використовувати і варіант віртуальних машин, але набагато ефективніше створити ізольовані контейнерів на базі одного ядра однієї ОС.

Наступний життєвий етап технологій x86-віртуалізації стартував в 2004-2006 рр. і був пов'язаний з початком їх масового застосування в корпоративних системах. Відповідно, якщо раніше розробники в основному займалися створенням технологій виконання віртуальних середовищ, то тепер на перший план стали виходити завдання управління цими рішеннями і їх інтеграції в загальну корпоративну IT-інфраструктуру. Одночасно позначилося помітне підвищення попиту на віртуалізацію з боку персональних користувачів (але якщо в 90-х це були розробники і тестери, то зараз мова вже йде про кінцевих користувачів як професійних, так і домашніх).

Багато складнощів і проблем розробки технологій віртуалізації пов'язані з подоланням успадкованих особливостей програмно-апаратної архітектури x86. Для цього існує кілька базових методів:

Повна віртуалізація (Full, Native Virtualization). Використовуються не модифіковані екземпляри гостьових операційних систем, а для підтримки роботи цих ОС слугує загальний шар емуляції їх виконання поверх хостової ОС, в ролі якої виступає звичайна операційна система. Така технологія застосовується, зокрема, в VMware Workstation, VMware Server (колишній GSX Server), Parallels Desktop, Parallels Server, MS Virtual PC, MS Virtual Server, Virtual Iron. До переваг даного підходу можна зарахувати відносну простоту реалізації, універсальність і надійність рішення; всі функції управління бере на себе хост-ОС. Недоліки - високі додаткові накладні витрати на апаратні

ресурси, відсутність врахування особливостей гостьових ОС, менша, ніж потрібно, гнучкість у використанні апаратних засобів.

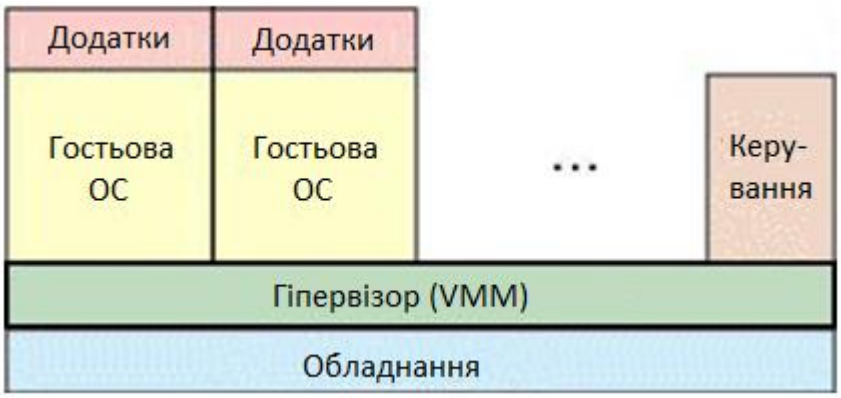


Рис. 1.4 Повна віртуалізація

Паравіртуалізація (paravirtualization). Модифікація ядра гостьової ОС виконується таким чином, що в неї включається новий набір API, через який вона може безпосередньо працювати з апаратурою, не конфліктуючи з іншими віртуальними машинами. При цьому немає необхідності задіювати повноцінну ОС в якості хостового ПЗ, функції якого в даному випадку виконує спеціальна система, що отримала назву гіпервізора (hypervisor). Саме цей варіант є сьогодні найбільш актуальним напрямком розвитку серверних технологій віртуалізації і застосовується в VMware ESX Server, Xen (і рішеннях інших постачальників на базі цієї технології), Microsoft Hyper-V.

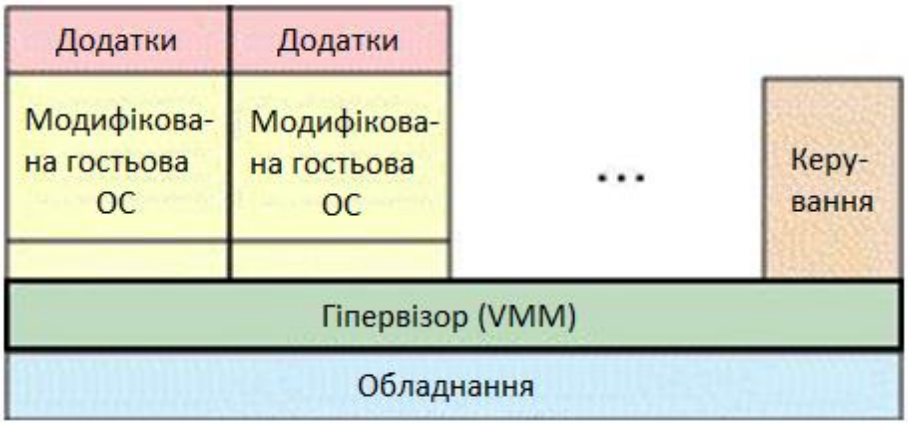


Рис. 1.5 Паравіртуалізація

Переваги даної технології полягають у відсутності потреби в хостовій ОС - ВМ встановлюються фактично на "голе залізо", а апаратні ресурси використовуються ефективно. Недоліки - в складності реалізації підходу і необхідності створення спеціалізованої ОС-гіпервізора.

Віртуалізація на рівні ядра ОС

Цей варіант передбачає використання одного ядра хостової ОС для створення незалежних паралельно працюючих операційних середовищ. Для гостьового ПЗ створюється тільки власне мережеве та апаратне оточення. Такий варіант використовується в Virtuozzo (для Linux і Windows), OpenVZ (безкоштовний варіант Virtuozzo) і Solaris Containers. Переваги - висока ефективність використання апаратних ресурсів, низькі накладні технічні витрати, відмінна керованість, мінімізація витрат на придбання ліцензій. Недоліки - реалізація тільки однорідних обчислювальних середовищ.

Рис. 1.6 Віртуалізація на рівні ОС

Віртуалізація додатків

Віртуалізація додатків передбачає застосування моделі сильної ізоляції прикладних програм з керованим взаємодією з ОС, при якій віртуалізується кожен екземпляр додатків, все його основні компоненти: файли (включаючи системні), реєстр, шрифти, INI-файли, COM-об'єкти, служби. Додаток

виповнюється без процедури інсталяції в традиційному її розумінні і може запускатися прямо з зовнішніх носіїв (наприклад, з флеш-карт або з мережевих папок). З точки зору IT-відділу, такий підхід має очевидні переваги: прискорення розгортання настільних систем і можливість управління ними, зведення до мінімуму не тільки конфліктів між додатками, а й потреби в тестуванні додатків на сумісність. Дана технологія дозволяє використовувати на одному комп'ютері, а точніше в одній і тій же операційній системі кілька несумісних між собою додатків одночасно. Віртуалізація додатків дозволяє користувачам запускати один і той же заздалегідь сконфігурований додаток або групу додатків з сервера. При цьому додатки будуть працювати незалежно один від одного, не вносячи жодних змін в операційну систему. Фактично саме такий варіант віртуалізації використовується в Sun Java Virtual Machine, Microsoft Application Virtualization (раніше називалося Softgrid), Thinstall (на початку 2008 р увійшла до складу VMware), Symantec / Altiris.

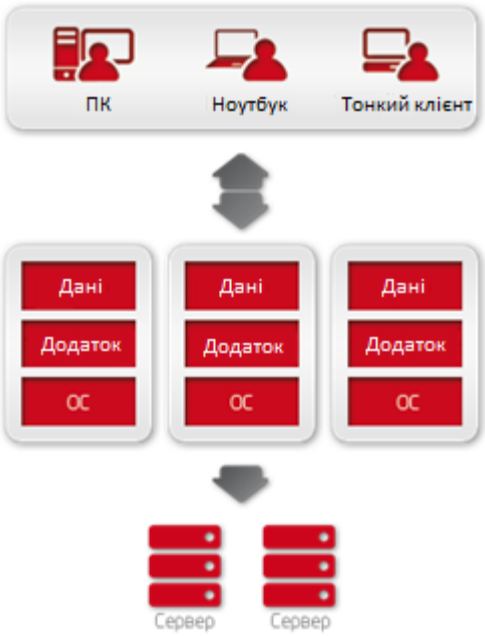


Рис. 1.7 Взаємодія «клієнт-сервер» при віртуалізації додатків

Віртуалізація представлень (робочих місць)

Віртуалізація представлень передбачає емуляцію інтерфейсу користувача [1]. Тобто користувач бачить додаток і працює з ним на своєму терміналі, хоча

насправді додаток виконується на віддаленому сервері, а користувачеві передається лише картинка віддаленого додатку. Залежно від режиму роботи, користувач може бачити віддалений робочий стіл і запущений на ньому додаток, або тільки саме вікно програми.

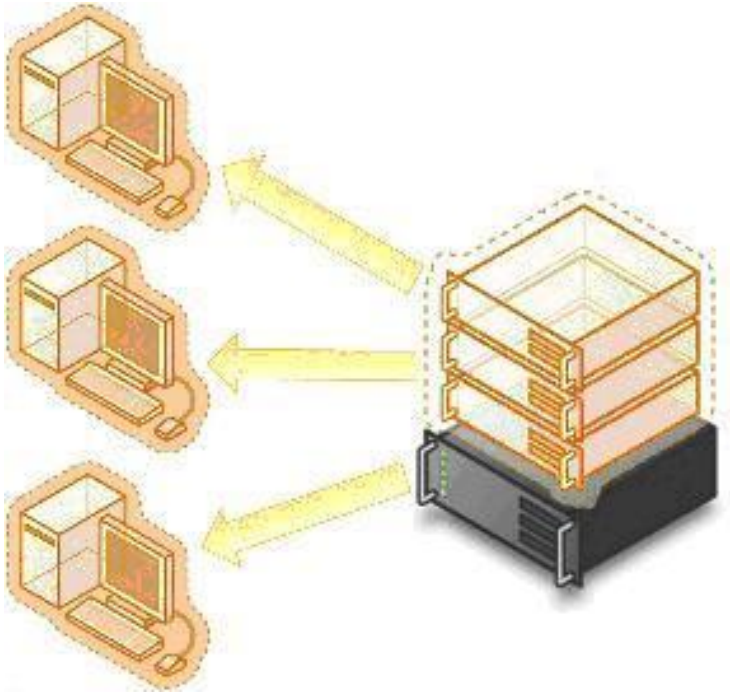


Рис. 1.8 Віртуалізація робочих місць

Потреби бізнесу змінюють наші уявлення про організацію робочого процесу. Персональний комп'ютер, що став за останні десятиліття невід'ємним атрибутом офісу і засобом виконання більшості офісних завдань, перестає встигати за зростаючими потребами бізнесу. Реальним інструментом користувача виявляється програмне забезпечення, яке лише прив'язане до ПК, роблячи його проміжною ланкою корпоративної інформаційної системи. В результаті активного розвитку набувають "хмарні" обчислення, коли користувачі мають доступ до власних даних, але не управляють і не замислюються про інфраструктуру, операційну систему і власне програмне забезпечення, з яким вони працюють.

Разом з тим, з ростом масштабів організацій, використання в IT-інфраструктурі призначених для користувача ПК викликає ряд складнощів:

– великі операційні витрати на підтримку комп'ютерного парку;

- складність, пов'язана з управлінням настільними ПК;
- забезпечення користувачам безпечного та надійного доступу до ПЗ і додатків, необхідним для роботи;
- технічний супровід користувачів;
- установка і оновлення ліцензій на ПЗ і технічне обслуговування;
- резервне копіювання і т.д.

Піти від цих складнощів і скоротити витрати, пов'язані з їх вирішенням, можливо завдяки застосуванню технології віртуалізації робочих місць співробітників на базі інфраструктури віртуальних ПК - Virtual Desktop Infrastructure (VDI). VDI дозволяє відокремити призначене для користувача ПЗ від апаратної частини - персонального комп'ютера, - а також доступ до клієнтських додатків через термінальні пристрої.

VDI - комбінація з'єднань з віддаленим робочим столом і віртуалізації. На обслуговуючих серверах працює безліч віртуальних машин, з такими клієнтськими операційними системами, як Windows 7, Windows Vista і Windows XP або Linux. Користувачі дистанційно підключаються до віртуальної машини свого настільного середовища. На локальних комп'ютерах користувачів в якості віддаленого настільного клієнта можуть застосовуватися термінальні клієнти, старе обладнання з Microsoft Windows Fundamentals або дистрибутив Linux.

VDI повністю ізолює віртуальне середовище користувачів від інших віртуальних середовищ, так як кожен користувач підключається до окремої віртуальної машині. Іноді використовується статична інфраструктура VDI, в якій користувач завжди підключається до тієї ж віртуальної машині, в інших випадках динамічна VDI, в якій користувачі динамічно підключаються до різних віртуальних машин, і віртуальні машини створюються в міру необхідності. При використанні будь-якої моделі важливо зберігати дані користувачів поза віртуальних машин і швидко надавати додатки.

Поряд з централізованим управлінням і простим наданням комп'ютерів, VDI забезпечує доступ до настільної середовищі з будь-якого місця, якщо користувачі можуть дистанційно підключитися до сервера.

Як приклад віртуалізації подань можна розглядати і технологію тонких терміналів, які фактично віртуалізують робочі місця користувачів настільних систем: користувач не прив'язаний до якогось конкретного ПК, а може отримати доступ до своїх файлів і додатків, які розташовуються на сервері, з будь-якого віддаленого терміналу після виконання процедури авторизації. Всі команди користувача і зображення сеансу на моніторі емулюються за допомогою ПЗ керування тонкими клієнтами.

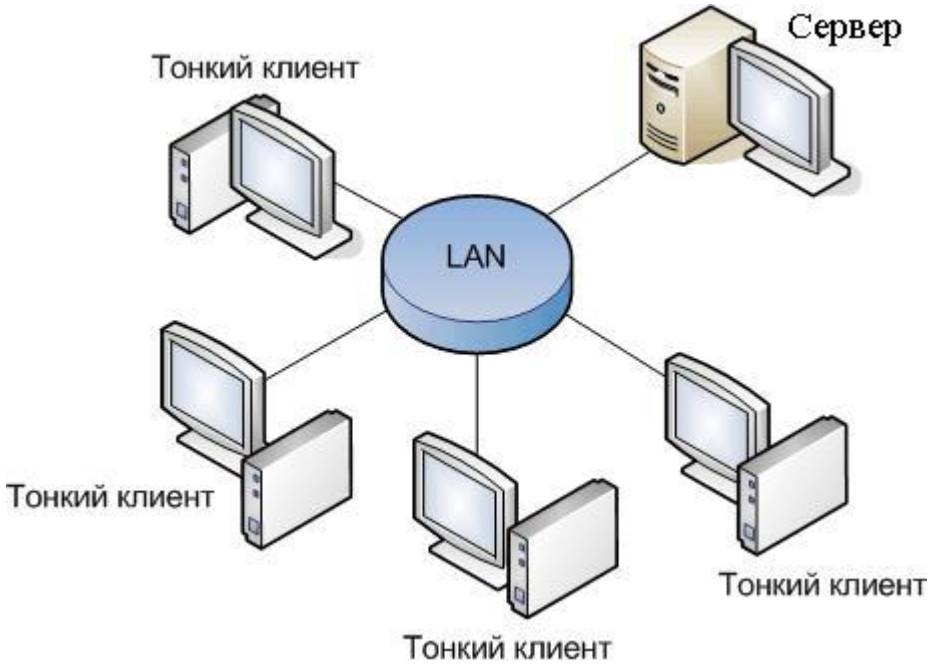


Рис. 1.9 Схема мережі з тонкими клієнтами

Застосування цієї технології дозволяє централізувати обслуговування клієнтських робочих місць і різко скоротити витрати на їх підтримку - наприклад, для переходу на наступну версію клієнтської програми нове ПЗ потрібно інсталювати тільки один раз на сервері [1].

1.2 Віртуалізація мереж

У традиційному середовищі (див. Рис. 1.10) комплекс необхідних додатків розгортається на групі фізичних серверів. Щоб організувати обмін інформацією між серверами, на кожен з них встановлюється один або кілька мережних адаптерів (NIC), підключених до зовнішньої мережевої інфраструктури. Мережеві адаптери і стек мережевого програмного забезпечення через мережеву інфраструктуру забезпечують комунікації між кінцевими точками. Як показано на малюнку 1.10, ця функціональність реалізується комутатором (switch), який відповідає за ефективний обмін пакетами між взаємодіючими кінцевими точками.

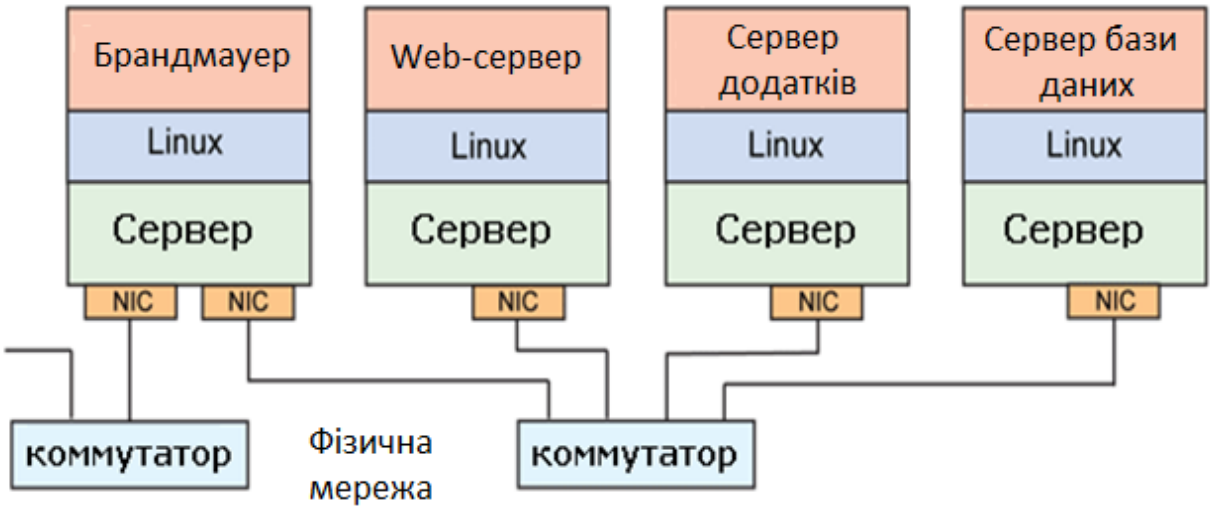


Рис. 1.10 Традиційна мережева інфраструктура

Ключова інновація, що лежить в основі об'єднання серверів, - це абстрактне апаратне забезпечення, що дозволяє декільком операційним системам і додаткам спільно використовувати фізичне апаратне забезпечення (див. Рис. 1.11). Ця абстракція називається гіпервізором або монітором віртуальних машин (virtual machine monitor). Кожна віртуальна машина (операційна система з набором додатків) бачить базове апаратне забезпечення як систему, яка використовується в монопольному режимі, хоча її окремі компоненти можуть бути розділені між декількома віртуальними машинами (ВМ) або взагалі не існувати. Як приклад можна розглянути віртуальний

мережевий адаптер (vNIC). Гіпервізор може створити один або кілька віртуальних мережевих адаптерів для кожної ВМ. Ці адаптери будуть видні в ВМ як фізичні, але в дійсності вони тільки надають інтерфейс до реально існуючого адаптера змінного струму. Гіпервізор також дозволяє динамічно створити віртуальну мережу з віртуальними комутаторами, щоб забезпечити настроюються комунікації між кінцевими точками віртуальних машин. Також гіпервізор допускає взаємодію з інфраструктурою фізичної мережі шляхом підключення фізичних мережевих адаптерів сервера до своєї логічної інфраструктурі, в результаті чого стають можливими комунікації між віртуальними машинами в рамках гіпервізора і комунікації в рамках зовнішньої мережі.

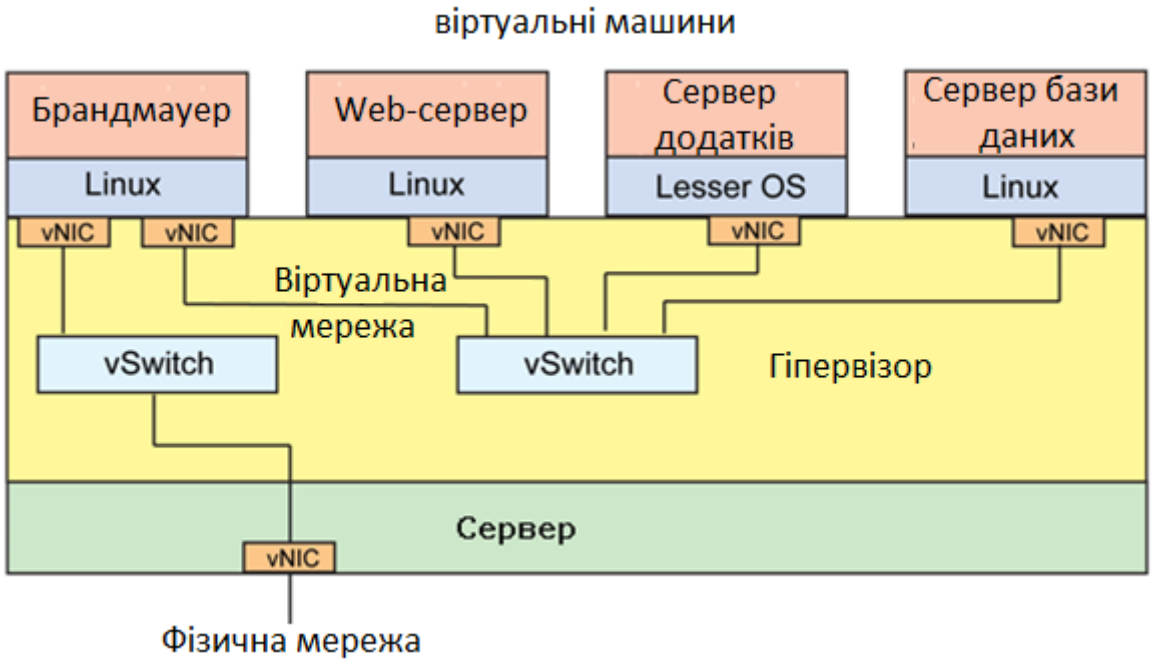


Рис. 1.11 Віртуалізована мережева інфраструктура

Віртуальна комутація

Віртуальний комутатор - це ключовий компонент, необхідний для віртуалізації мережевої інфраструктури. Віртуальний комутатор з'єднує віртуальні мережеві адаптери (vNIC) з фізичними мережевими адаптерами, встановленими на сервері, і, що більш важливо, пов'язує одні віртуальні мережеві адаптери з іншими для локального взаємодії в рамках сервера. Варто

зазначити, що для віртуального комутатора межа можливостей визначається не швидкістю мережі, а пропускною спроможністю пам'яті, яка дозволяє налагодити ефективні комунікації між локальними віртуальними машинами і мінімізувати навантаження на мережеву інфраструктуру. Ця економія пов'язана з тим, що ресурси фізичної мережі витрачаються тільки на комунікації між серверами, а трафік між віртуальними машинами ізолюється всередині серверів.

Оскільки в ядрі Linux вже присутній комутатор другого рівня, виникає питання, навіщо потрібен ще й віртуальний комутатор. Відповідь на це питання враховує кілька факторів, але найбільш важливий з них - це нова класифікація комутаторів. Новий клас комутаторів називається розподіленим віртуальним комутатором (distributed virtual switch) і застосовується для з'єднання серверів таким способом, щоб розміщена нижче архітектура стала "прозорою" для використовуючих її додатків. Віртуальний комутатор на одному сервері може «прозоро» для користувачів підключитися до віртуального комутатора на іншому сервері (див. Рис. 1.12). Це значно спрощує міграцію віртуальних машин між серверами і їх віртуальними інтерфейсами, так як вони можуть підключитися до розподіленого віртуального комутатора іншого сервера і "прозоро" приєднатися до його віртуальної мережі.

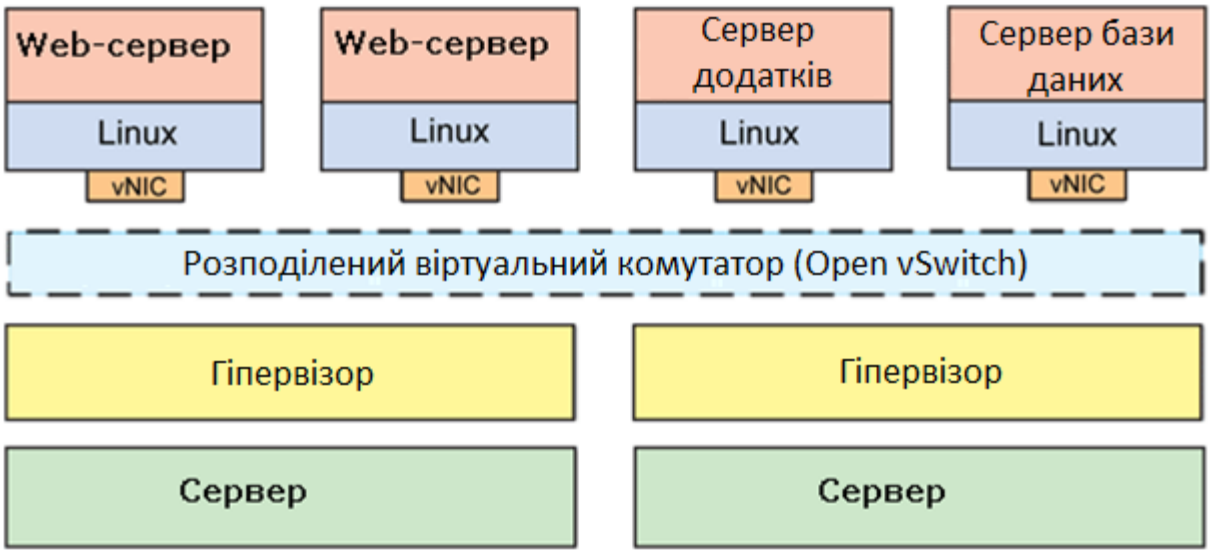


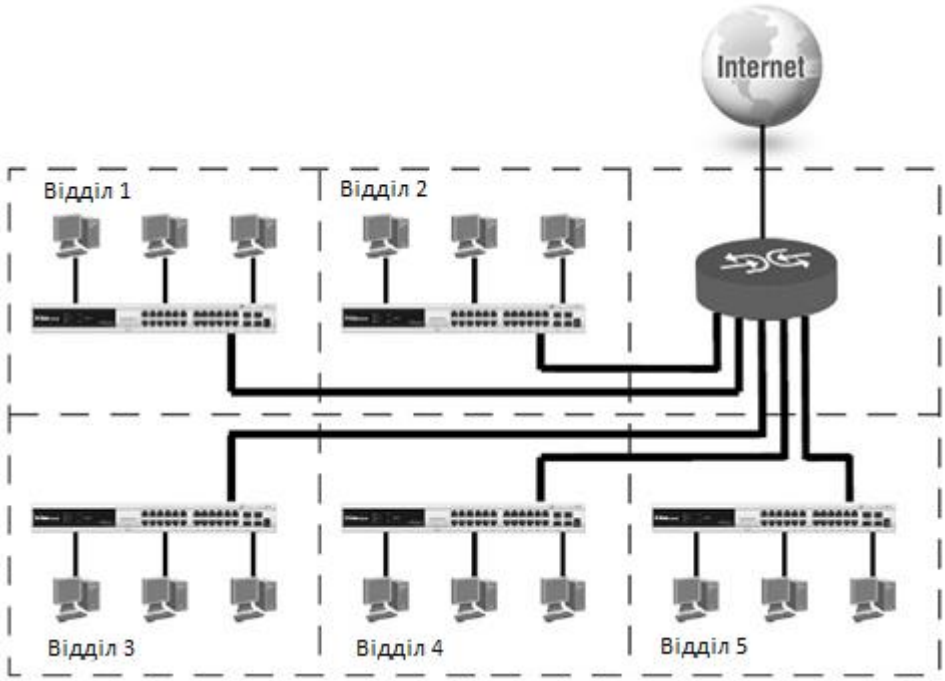
Рис. 1.12 Розподілений віртуальний комутатор

Один з найбільш важливих проектів в цій області - проект Open vSwitch, який буде розглядатися далі.

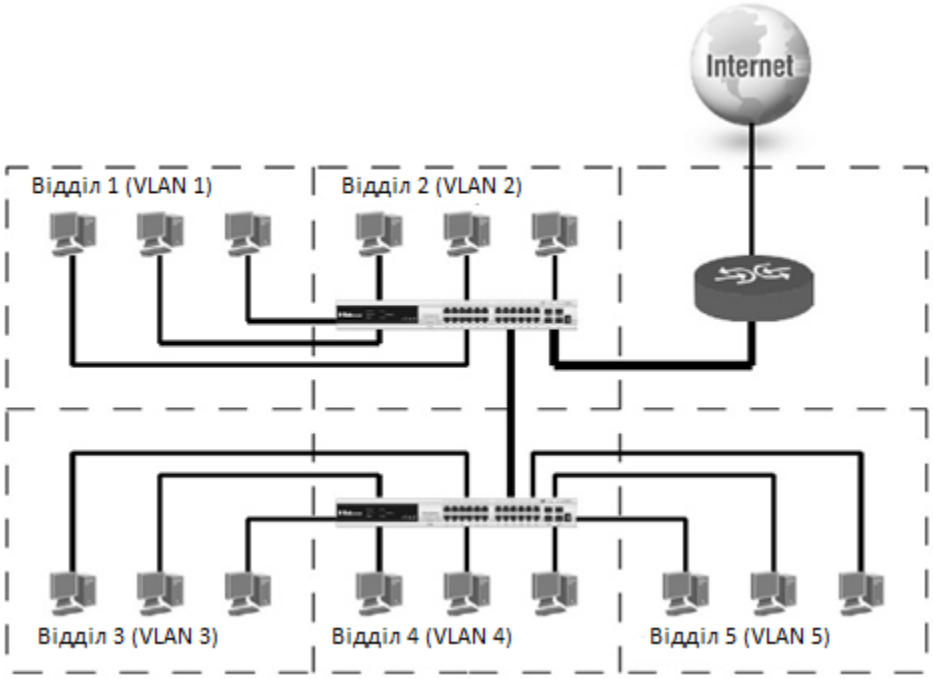
Через ізоляцію локального трафіку всередині сервера виникає проблема, пов'язана з тим, що цей трафік виявляється недоступний ззовні (наприклад, мережевим аналізаторам). У різних реалізаціях ця проблема вирішується за допомогою різних схем, наприклад, OpenFlow, NetFlow і sFlow, покликаних забезпечити віддалений доступ для контролю і моніторингу трафіку.

Віртуальні локальні мережі

Віртуальні локальні мережі (virtual LAN - vLAN) - це один із способів віртуалізації мереж. Ця технологія дозволяє створити віртуальну мережу в рамках розподіленої мережі, так що різні комп'ютери в незалежних мережах будуть відображатися як елементи однієї віртуальної мережі. Це досягається завдяки постачанню фреймів з інформацією віртуальної мережі спеціальними позначками, що дозволяє впізнати приналежність цих фреймів до певної фізичної мережі (відповідно до стандарту IEEE 802.1Q). Віртуалізація мережі виконується самими хостами за допомогою VLAN-комутатора. Але хоча віртуальні мережі забезпечують ілюзію незалежності мереж один від одного, вони все одно використовують загальну фізичну мережу, тому інтенсивний обмін даними може чинити негативний вплив на загальну пропускну здатність мережі.



а)



б)

Рис. 1.13 Різниця в організації локальної мережі: а) з використанням лише фізичної комутації; б) з використанням VLAN

На малюнку показана різниця в організації локальної мережі з використанням VLAN і без. Як приклад наведена топологія локальної мережі підприємства з декількома відділами [2].

3 види мережевої взаємодії між віртуальними машинами

Продукти VMware Workstation і VMware Server надають користувачам можливість призначити віртуальній машині один з трьох базових типів мережевої взаємодії для кожного з віртуальних мережевих адаптерів:

- Bridged (Мережевий міст)
- Host-only (Тільки хост)
- NAT (Трансляція мережевих адрес)

Кожен з цих видів мережевої взаємодії може застосовуватися для різних варіантів використання віртуальних машин і необхідно ретельно підбирати тип мережевої взаємодії віртуальної машини для більш ефективного її використання спільно з іншими компонентами мережевої інфраструктури.

Bridged Networking

Цей тип мережевої взаємодії дозволяє прив'язати мережевий адаптер віртуальної машини до фізичної мережевому інтерфейсу комп'ютера, що дає можливість розділяти ресурси мережевої карти між хостовой і віртуальною системою. Віртуальна машина з таким типом мережевої взаємодії буде вести себе по відношенню до зовнішньої мережі хостовой системи як незалежний комп'ютер. Ви можете призначити такій машині власний IP-адреса в домашній мережі або мережі організації, або вона отримає його від зовнішнього DHCP-сервера. Для створюваної віртуальної машини цей тип мережевої взаємодії призначається за замовчуванням, оскільки це найбільш простий спосіб організації мережевої взаємодії між віртуальною машиною, хостом і зовнішньою мережею. Структура Bridged Networking приведена нижче.

Віртуальний мережевий адаптер гостьової системи підключається до віртуального комутатора VMnet0, до якого також підключений віртуальний

міст, який взаємодіє безпосередньо з фізичним мережним адаптером.

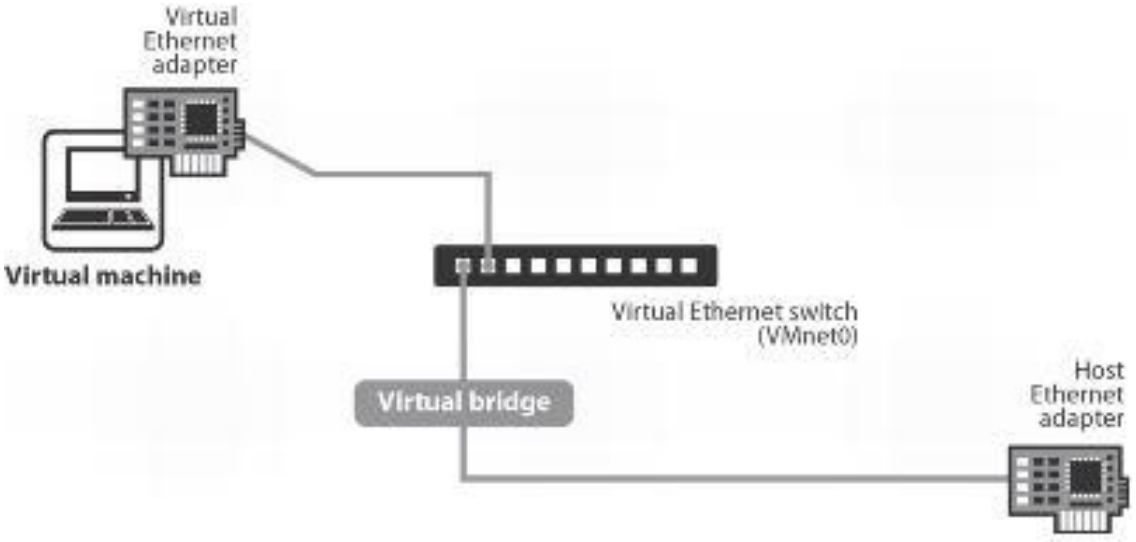


Рис. 1.14 Взаємодія з допомогою мережевого моста

Host-Only Networking

Такий тип мережевої взаємодії оптимальний для цілей тестування програмного забезпечення, коли вам потрібно організувати віртуальну мережу в межах хоста, а віртуальним машинам не потрібно вихід в зовнішню мережу. У віртуальній підмережі діє DHCP-сервер, підключений до віртуального комутатора VMnet1 і призначає віртуальним машинам IP-адреси з заданого діапазону (за замовчуванням 192.168.179.128 - 192.168.179.254). Структура Host-Only Networking приведена нижче:

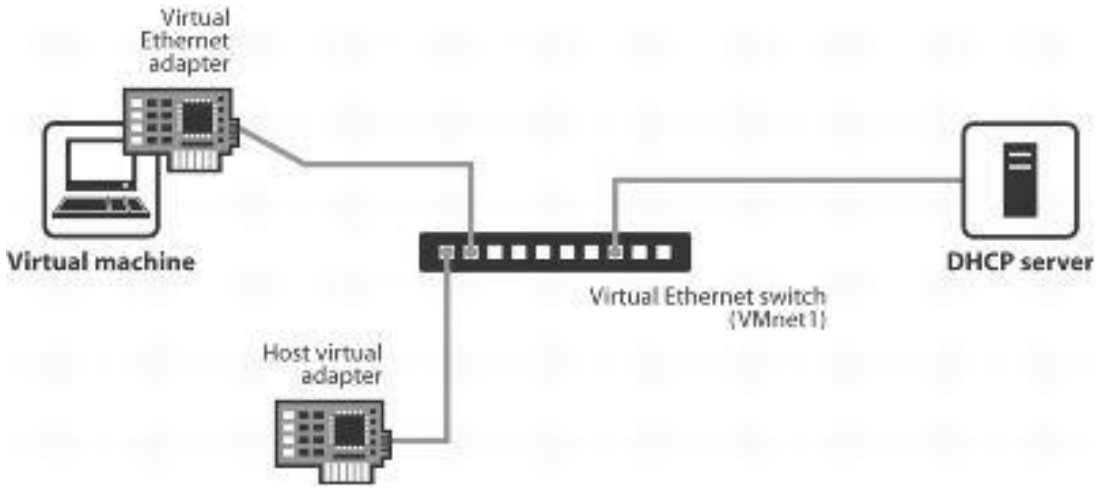


Рис. 1.15 Взаємодія типу Host-Only

Віртуальні мережеві адаптери гостьових систем підключаються до комутатора VMnet1 і взаємодіють в підмережі 192.168.179.0/24. У хостовой системі створюється також віртуальний мережевий інтерфейс, підключений до VMnet1, який дозволяє взаємодіяти з віртуальними машинами.

NAT Networking

Цей тип мережевої взаємодії дуже схожий на Host-Only, за одним винятком: до віртуального комутатора VMnet8 підключається пристрій трансляції IP-адрес (NAT). До цього комутатора також підключається DHCP-сервер, що роздає віртуальним машинам адреси з заданого діапазону (за замовчуванням 192.168.89.128 - 192.168.89.254) і, безпосередньо, самі віртуальні машини. NAT-пристрій дозволяє здійснювати трансляцію IP-адрес, що дозволяє віртуальним машинам ініціювати з'єднання в зовнішню мережу, котрі дають при цьому механізму доступу до віртуальних машин ззовні. Структура NAT Networking приведена нижче:

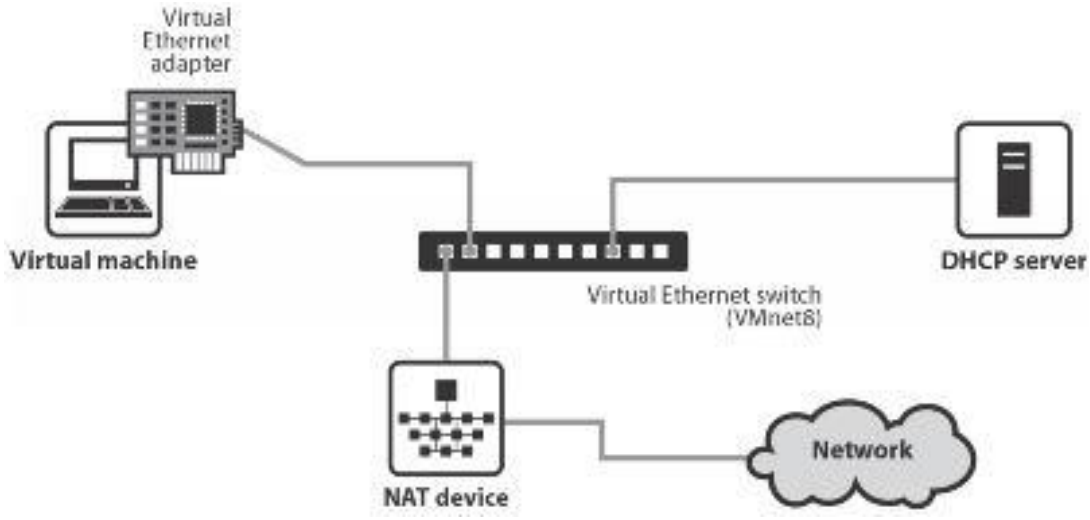


Рис. 1.16 Мережева взаємодія NAT

У хостовой операційній системі, також, як і для Host-Only Networking, створюється віртуальний мережевий інтерфейс для комутатора VMnet 8, що дозволяє хосту спілкуватися з віртуальними машинами.

Така модель мережевої взаємодії оптимальна з точки зору безпеки (оскільки неможливо ініціювати ззовні з'єднання з віртуальною машиною),

проте істотно знижує швидкодію мережі (іноді, до 20-30 відсотків). NAT-з'єднання може використовуватися, наприклад, для безпечної роботи в Інтернет з віртуальної машини [3].

1.2.1 Використання віртуалізації в інфраструктурі SDN та NFV

Головна ідея SDN полягає в відділенні функцій передачі трафіку від функцій управління (включаючи контроль як самого трафіку, так і здійснення його передачі пристроям). У традиційних комутаторах і маршрутизаторах ці процеси невіддільні одна від одної і реалізовані в одній «коробці»: спеціальні мікросхеми забезпечують пересилання пакетів з одного порту на інший, а вищерозміщене ПЗ визначає правила такої пересилання, виконує необхідний аналіз пакетів, виробляє зміна міститься в них службової інформації і т. д. (див. Малюнок 1). Для визначення маршруту передачі або недопущення зациклення трафіку пристрої, звичайно, «спілкуються між собою», для чого розроблено безліч протоколів, таких як OSPF, BGP і Spanning Tree, але при цьому кожне функціонує досить автономно.

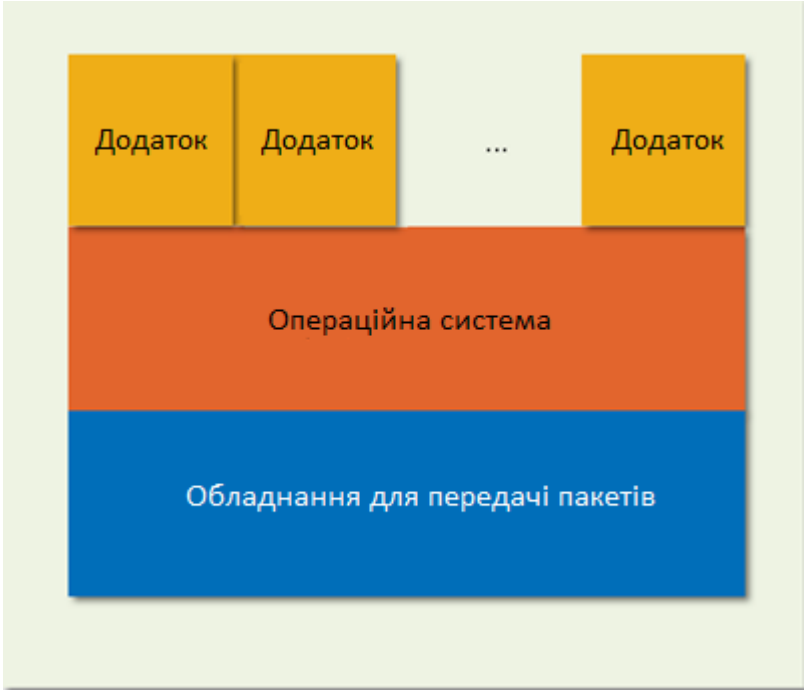


Рисунок 1.17 Архітектура типового комутатора або маршрутизатора.

Згідно з концепцією SDN, вся логіка управління виноситься в так звані контролери, які здатні відслідковувати роботу всієї мережі (див. Рис. 1.18). Не можна сказати, що це революційно нова ідея: зв'язківці пам'ятають, що подібну логіку свого часу передбачалося реалізувати при модернізації телефонних мереж, результатом чого стала поява «інтелектуальних мереж», а потім і комутаторів класу Softswitch.

SDN передбачає перегляд мережевої архітектури з поділом управління, комутації і передачі даних - перехід від автономних розподілених систем, де кожне мережеве пристрій має свій «інтелект» і конфігурацію і здійснює передачу даних відповідно до закладеної в ньому логікою і протоколами, до мереж з поділом передачі даних (Data Plane) і управління (Control Plane). У таких мережах всі завдання, пов'язані з конфігурацією обладнання, маршрутизацією трафіку і мережевими сервісами, винесені на рівень управління - рівень контролера SDN та мережевої операційної системи [4].

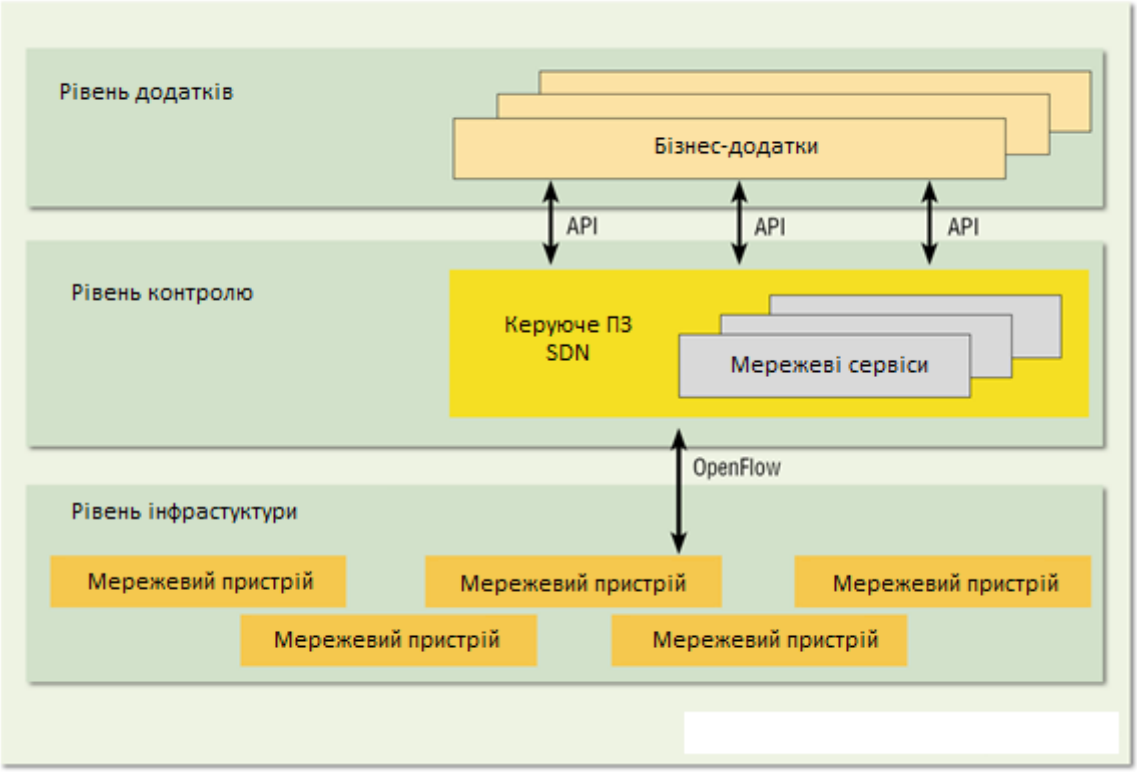


Рисунок 1.18 Архітектура SDN

Основним елементом концепції SDN є протокол OpenFlow, який забезпечує взаємодію контролера з мережевими пристроями (див. Рис. 1.18).

Як і випливає з назви, протокол OpenFlow при ідентифікації трафіку оперує поняттям «потоку». Ключовим елементом комутатора, що підтримує цей протокол, є таблиця потоків (Flow Table). Група стовпців в лівій частині таблиці формує поля відповідності, де вказані характеристики потоків: це можуть бути різні параметри, включаючи MAC і IP-адреси відправника і одержувача, ідентифікатор VLAN, номера протокольних портів TCP і UDP, а також інша інформація. Ці дані за допомогою протоколу OpenFlow записує в таблицю комутатора контролер, він же визначає пріоритет різних потоків: чим вище пріоритет, тим вище відповідний запис в таблиці потоків. Вхідні пакети перевіряються на відповідність зазначеним у таблиці параметрами. Якщо відповідність виявлено, до пакетів застосовується дія, яка вказана в наступному стовпці таблиці. Типовим дією є пересилання пакета на один або кілька вихідних портів. Крім того, комутатор може змінити вміст службових полів пакету, скинути його, направити для аналізу контролеру і т. д. У разі якщо збіг не знайдено, пакет скидається або надсилається контролера, який визначить, як слід обробляти даний потік, і додасть відповідний запис в таблицю. Статистика по проходить трафіку - число пакетів, байтів тощо. - поміщається у відповідні поля. Використовуючи протокол OpenFlow, контролер додає, модифікує і видаляє записи в таблиці потоків. Крім того, він може запитувати у комутатора його характеристики і зібрану статистику, конфігурувати комутатор і його окремі порти.

На «північній» стороні контролер надає програмні інтерфейси (API), наявність яких дозволяє власнику мережі або стороннім розробникам створювати додатки для управління мережею. Такі додатки можуть виконувати найрізноманітніші функції в інтересах бізнес-завдань (наприклад, контролювати доступ, управляти пропускною спроможністю і т. п.), Причому їх розробникам не треба знати деталі функціонування конкретних мережевих пристроїв. Завдяки контролеру, вся мережа, що складається з безлічі різнотипних пристроїв різних виробників, постає для додатка як один логічний комутатор.

Контролер забезпечує автоматичне адміністрування мережевих пристроїв, при цьому оператор зв'язку має можливість створювати власне ПЗ управління мережею, а відкриті програмні інтерфейси (API) дозволяють впроваджувати мережеві додатки сторонніх вендорів. Крім того, контролер забезпечує зв'язок між рівнем додатків і мережею. Наприклад, він може передати з додатком інформацію про трафік, після чого програму за допомогою контролера змінює конфігурацію мережі. Таким чином, мережева інфраструктура стає динамічною.

При впровадженні SDN необхідно правильно спланувати мережу, з урахуванням її архітектури і функціональності мережевих елементів. Реалізація SDN кожним виробником має свої особливості. Існують складності інтеграції з наявними системами керування, розширення / зміни мережевої інфраструктури і реалізації нових сценаріїв [5].

Поділ «площин» передачі та управління можна реалізувати взагалі не зачіпаючи наявну фізичну мережу - задіюючи віртуальні комутатори зразок Cisco Nexus 1000v, VMware DVS, IBM 5000v або навіть Open vSwitch з відкритим вихідним кодом. Програмування таких комутаторів за допомогою контролера дозволяє створити віртуальну мережу SDN поверх наявної фізичної інфраструктури. Деякі експерти розглядають цей підхід як альтернативу развиваемому ONF, але насправді, оскільки описувана схема не виключає можливості використання стандартного протоколу OpenFlow, протиставляти її рішенням ONF не варто.

Якщо в такій мережі звичайні комутатори також будуть підтримувати OpenFlow, то до віртуальної мережі можна буде підключити і фізичні сервери. Управління такими комутаторами теж можна буде передати контролеру, якщо це не увійде в суперечність з принципом поділу фізичному або віртуальних мереж. Цей приклад показує, що в моделі SDN конкретна реалізація комутатора - будь то фізичний пристрій або програма на гіпервізора - не має принципового значення, головне, щоб він міг отримувати і виконувати інструкції контролера.

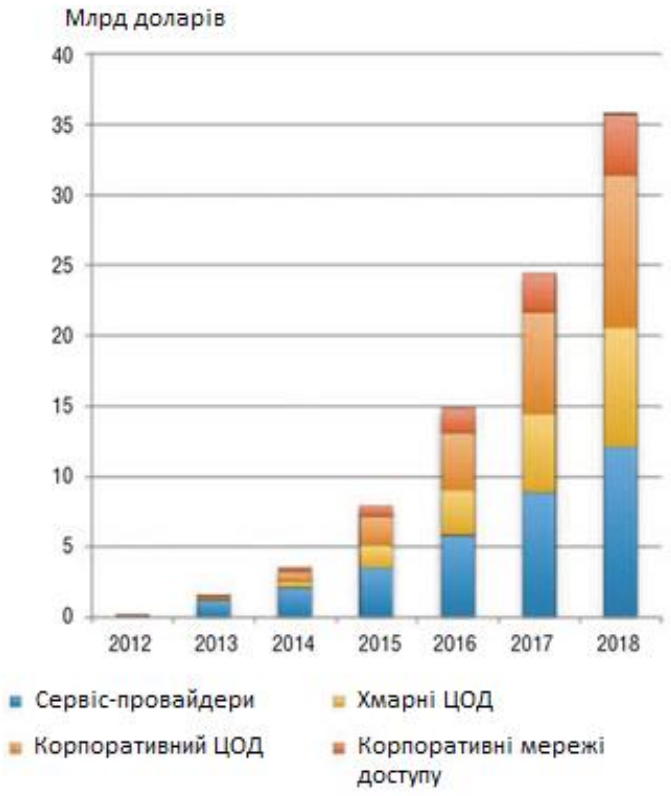


Рисунок 1.19 Ріст витрат на SDN в світі

За прогнозом аналітиків, до 2018 року обсяг світового ринку SDN виросте до 35 млрд доларів (див. Рис. 1.19). А 40% всіх витрат на мережі передачі даних будуть так чи інакше пов'язані з SDN. В першу чергу SDN будуть затребувані сервіс-провайдерами, хмарними комерційними центрами обробки даних, великими корпоративними ЦОД.

З SDN тісно пов'язане інший напрямок - віртуалізація мережевих функцій. NFV означає перегляд архітектури компонентів мережі та їх віртуалізацію і передбачає заміну традиційного спеціалізованого мережевого обладнання на віртуальні пристрої - віртуальні машини (ВМ) з відповідною функціональністю, що працюють в середовищі віртуалізації на стандартних серверах (NEC застосовує для цього гіпервізор «операторського класу »на базі KVM).

Замість спеціалізованого мережевого обладнання NFV передбачає використання стандартних серверів з гіпервізором віртуалізації і переклад всіх мережевих компонентів на рівень ВМ. Як наслідок, важливим завданням стає створення гіпервізора віртуалізації операторського класу, що дозволяє

подолати проблеми продуктивності мережевого стека і забезпечити моніторинг ВМ з контролем продуктивності і функціональності, підкреслює Олексій Стребулаев. З огляду на обмежену продуктивність ВМ, потрібен максимально ефективний механізм балансування навантаження, і таким є мережа SDN на базі OpenFlow, що дозволяє використовувати фізичні порти, а не прошарок у вигляді Open vSwitch, продуктивність якої обмежена мережевим стеком.

Основні типи мереж, де може бути затребувана (і вже задіюється) технологія SDN - це кампусні мережі, мережі ЦОД, хмарні платформи, а області застосування - мережі для хмар, оркестрації і автоматизація. В даний час основними об'єктами впровадження SDN є великі хмарні мережі і платформи. Поки лише деякі замовники вирішуються побудувати ЦОД на базі SDN, хоча такі приклади є. Технологія NFV використовується в маршрутизаторах, міжмережевих екранах і шлюзах, пристроях CDN, акселераторах WAN, контролерах доставки додатків (ADC) в мережах операторів і сервіс-провайдерів. SDN та NFV не пов'язані між собою, але добре доповнюють один одного, роблять висновок експерти [6].

1.3 Переваги та недоліки технології віртуалізації

Як показано в [1], основними перевагами технологій традиційної віртуалізації є:

1. *Ефективне використання обчислювальних ресурсів*. Замість 3х, а то 10 серверів, завантажених на 5-20% можна використовувати один, який використовується на 50-70%. Крім іншого, це ще й економія електроенергії, а також значне скорочення фінансових вкладень: купується один високотехнологічний сервер, що виконує функції 5-10 серверів. За допомогою віртуалізації можна досягти значно більш ефективного використання ресурсів, оскільки вона забезпечує об'єднання стандартних ресурсів інфраструктури в єдиний пул і долає обмеження застарілої моделі "один додаток на сервер".

2. *Скорочення витрат на інфраструктуру.* Віртуалізація дозволяє скоротити кількість серверів і пов'язаного з ними IT-обладнання в інформаційному центрі. В результаті цього потреби в обслуговуванні, електроживленні і охолодженні матеріальних ресурсів скорочуються, і на IT витрачається значно менше коштів.
3. *Зниження витрат на програмне забезпечення*. Деякі виробники програмного забезпечення ввели окремі схеми ліцензування спеціально для віртуальних середовищ. Так, наприклад, купуючи одну ліцензію на Microsoft Windows Server 2008 Enterprise, ви отримуєте право одночасно її використовувати на 1 фізичному сервері і 4 віртуальних (в межах одного сервера), а Windows Server 2008 Datacenter ліцензується тільки на кількість процесорів і може використовуватися одночасно на необмеженій кількості віртуальних серверів.
4. *Підвищення гнучкості і швидкості реагування системи*. Віртуалізація пропонує новий метод управління IT-інфраструктурою та допомагає IT-адміністраторам витрачати менше часу на виконання повторюваних завдань - наприклад, на ініціацію, настройку, відстежування і технічне обслуговування. Багато системні адміністратори відчували неприємності, коли "валиться" сервер. І не можна, витягнувши жорсткий диск, переставивши його в інший сервер, запустити все як раніше ... А установка? пошук драйверів, настройка, запуск ... і на все потрібні час і ресурси. При використанні віртуального сервера - можливий моментальний запуск на будь-якому "залозі", а якщо немає подібного сервера, то можна завантажити готову віртуальну машину з встановленим і налаштованим сервером, з бібліотек, підтримуваних компаніями розробниками гіпервізора (програм для віртуалізації).
5. *Несумісні додатки можуть працювати на одному комп'ютері*. При використанні віртуалізації на одному сервері можлива установка linux і windows серверів, шлюзів, баз даних та інших абсолютно несумісних в рамках однієї не віртуалізованої системи додатків.

6. *Підвищення доступності додатків і забезпечення безперервності роботи підприємства*. Завдяки надійній системі резервного копіювання та міграції віртуальних середовищ цілком без перерв в обслуговуванні ви зможете скоротити періоди планового простою і забезпечити швидке відновлення системи в критичних ситуаціях. "Падіння" одного віртуального сервера не веде до втрати інших віртуальних серверів. Крім того, в разі відмови одного фізичного сервера можливо зробити автоматичну заміну на резервний сервер. Причому це відбувається непомітно для користувачів без перезавантаження. Тим самим забезпечується безперервність бізнесу.
7. *Можливості легкої архівації*. Оскільки жорсткий диск віртуальної машини зазвичай представляється у вигляді файлу певного формату, розташований на будь-якому фізичному носії, віртуалізація дає можливість простого копіювання цього файлу на резервний носій як засіб архівування та резервного копіювання всієї віртуальної машини цілком. Можливість підняти з архіву сервер повністю ще одна чудова особливість. А можна підняти сервер з архіву, не знищуючи поточний сервер і подивитися стан справ за минулий період.
8. *Підвищення керованості інфраструктури*. Використання централізованого управління віртуальною інфраструктурою дозволяє скоротити час на адміністрування серверів, забезпечує балансування навантаження і "живу" міграцію віртуальних машин [1].

Однак, застосування засобів віртуалізації, як і будь-якого складного продукту, пов'язане з низкою проблем:

1. В процесі реалізації деяких проектів доводиться стикатися з ситуаціями, коли засоби віртуалізації не дозволяють прикладному ПЗ або інформаційним системам клієнта працювати з периферійними пристроями. Інакше кажучи, проміжний рівень представлення даних вносить певні труднощі в роботу тих чи інших сервісів. Це означає, що використання віртуалізації вимагає підготовки і ретельного тестування

пілотного проекту. Тобто необхідно зібрати пул необхідного периферійного обладнання, пакет прикладного ПЗ, реалізувати заплановану архітектуру і домогтися повноцінної роботи і функціональності тестованого рішення.

2. Другий момент - витрата обчислювальної потужності сервера на віртуалізацію, який також треба враховувати. Щоб віртуалізація була ефективною, потрібно дуже уважно стежити за завантаженням віртуальних серверів і своєчасним виділенням достатніх для оптимальної роботи ресурсів. Справа в тому, що при пікових навантаженнях обробка конкурентного доступу до ресурсів викликає зростання накладних витрат на віртуалізацію. Тому все віртуальні сервери необхідно ретельно налаштовувати, для чого зазвичай використовуються засоби моніторингу навантаження, якими оснащуються всі віртуальні сервери. Крім того, існує маса програмних засобів, що дозволяють перерозподілити навантаження між серверами, встановленими на різних комп'ютерах.
3. Є і третій момент. Основна мета віртуалізації - оптимізувати використання апаратного забезпечення. Для цього адміністратор системи повинен стежити за завантаженістю віртуальних серверів і при необхідності перерозподіляти обчислювальні ресурси між ними або самі сервери переміщати між апаратними платформами. Саме тому застосування засобів віртуалізації, як правило, передбачає використання як мінімум двох комплектів апаратних засобів. Подібне рішення корисно ще й тим, що позбавляє нас від збоїв апаратури. Таким чином, під час налаштування віртуальних серверів необхідно застосовувати спеціальні засоби визначення завантаженості, що допомагають встановлювати оптимальний баланс навантаження і, крім того, рекомендується використовувати такі архітектурні рішення засобів віртуалізації, які встановлюються на порожню апаратну платформу [7].

РОЗДІЛ 2
АПАРАТНА ВІРТУАЛІЗАЦІЯ ТА ВАРІАНТИ ЇЇ РЕАЛІЗАЦІЇ

2.1 Архітектурні особливості апаратної віртуалізації. Гіпервізори та їх типи.

Віртуалізація зазвичай застосовується до фізичних апаратних ресурсів шляхом об'єднання кількох фізичних ресурсів в загальні пули, з яких користувачі отримують віртуальні ресурси. За допомогою віртуалізації з одного фізичного ресурсу можна зробити кілька віртуальних.

Більш того, віртуальні ресурси можуть мати функції або особливості, які відсутні у вихідних фізичних ресурсів.

При віртуалізації в рамках однієї фізичної системи створюється кілька віртуальних систем. Віртуальні системи - це незалежно функціонуючі середовища, які використовують віртуальні ресурси. Віртуальні системи часто називають логічними розділами або віртуальними машинами. Віртуалізація системи найчастіше здійснюється за допомогою технології гіпервізора.

Гіпервізор - це програма або вбудоване програмне забезпечення, що дозволяє віртуалізувати системні ресурси і забезпечує одноразову паралельне функціонування декількох операційних систем на одному комп'ютері [8].

Основне призначення гіпервізора - забезпечення ізольованих середовищ виконання для кожної віртуальної машини і управління доступом віртуальної машини і гостьовий операційної системи до фізичних апаратних ресурсів комп'ютера.

2.1.1 Основні типи гіпервізорів

Виділяють кілька типів гіпервізорів.

Гіпервізор першого типу

Гіпервізор першого типу виконується як контрольна програма безпосередньо на стороні апаратної частини комп'ютера. Операційні системи віртуальних машин виконуються рівнем вище.

Рис. 2.1 Схематичне представлення гіпервізора 1-го типу

Оскільки даний гіпервізор працює незалежно від операційної системи, він забезпечує більшу продуктивність, надійність і безпеку.

Гіпервізор першого типу використовуються в наступних рішеннях:

- Microsoft Hyper - V.
- VMware ESX Server.
- Citrix XenServer.

Гіпервізор другого типу

Гіпервізор другого типу виконується в рамках хостової операційної системи. Гостьові операційні системи віртуальних машин розташовуються рівнем вище.

Даний тип гіпервізора забезпечує гіршу продуктивність, в порівнянні з першим типом.

Рис. 2.2 Схематичне представлення гіпервізора 2-го типу

Гіпервізор другого типу використовуються в наступних рішеннях:

– Microsoft Virtual Server.
– VMware Server.
– Microsoft Virtual PC.

Монолітний гіпервізор

Наступним типом гіпервізора є монолітний. До складу монолітного гіпервізора включені драйвери апаратних пристроїв.

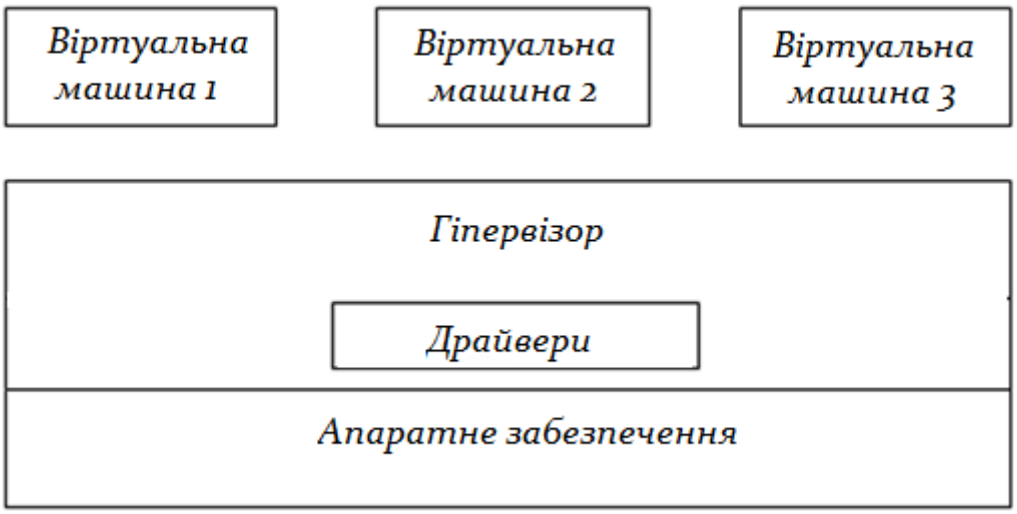


Рис. 2.3 Схематичне представлення монолітного гіпервізора

У монолітної моделі гіпервізор використовує для доступу до обладнання власні драйвери. Гостьові ОС працюють на віртуальних машинах поверх гіпервізора. Коли гостьовий системі потрібен доступ до обладнання, вона повинна пройти через гіпервізор і його модель драйверів. Зазвичай одна з гостьових ОС грає роль адміністратора або консолі, в якій ви запускаєте компоненти для надання ресурсів, управління і моніторингу всіх гостьових ОС, що працюють на комп'ютері.

Модель монолітного гіпервізора забезпечує прекрасну продуктивність, але «кульгає» з точки зору захищеності і стійкості. Це пов'язано з тим, що вона за своєю суттю має більш широким фронтом нападу і піддає систему більшого потенційному ризику, оскільки дозволяє роботу драйверів (а іноді навіть програм сторонніх виробників) в дуже чутливій області. Скажімо, шкідлива програма здатна встановити в гіпервізора під виглядом драйвера пристрою реєстратор натискань клавіш. Якщо таке трапитися, під контролем реєстратора виявляться всі гостьові ОС системи. Гірше того, цей «жучок» буде абсолютно неможливо виявити засобами гостьових ОС: гіпервізор для них невидимий!

Використання монолітного гіпервізора має як ряд переваг, так і деякі недоліки. Як перевага можна відзначити порівняно високу продуктивність, оскільки гостьові операційні системи взаємодіють безпосередньо з апаратним забезпеченням хостового комп'ютера.

Ще одна проблема - стійкість: якщо в оновлену версію драйвера закралась помилка, в результаті поновлення збої почнуться у всій системі, у всіх її віртуальних машинах. Іншими словами, в цій моделі критичне значення набуває стійкість драйверів, а вони часто створюються сторонніми виробниками, що може стати причиною проблем При цьому, обладнання серверів постійно еволюціонує, і тому драйвери оновлюються досить часто, що збільшує ризик різних неприємностей. Можна вважати монолітну модель моделлю «товстого гіпервізора» - через кількість драйверів, підтримка яких йому потрібна.

При цьому, з огляду на різноманіття апаратних пристроїв таких як материнські плати, контролери HDD, мережеві адаптери і т.д., розробники рішень віртуалізації змушені тісно співпрацювати з виробниками апаратного забезпечення. Таким чином, підтримується тільки обладнання, драйвери якого містяться в гіпервізора. Також варто відзначити нижчу безпеку, оскільки все відбувається в найбільш привілейованої частини системи.

Монолітний гіпервізор використовується в рішенні VMware ESX.

Мікроядерний гіпервізор

Альтернативу монолітному підходу становить мікроядерна модель. У ній можна говорити про «тонкому гіпервізора» - в цьому випадку в ньому зовсім немає драйверів. Замість цього драйвери працюють в кожному індивідуальному розділі, щоб будь-яка гостьова ОС мала можливість отримати через гіпервізор доступ до обладнання. При такій розстановці сил кожна віртуальна машина займає абсолютно окремий розділ, що позитивно позначається на захищеності і надійності.

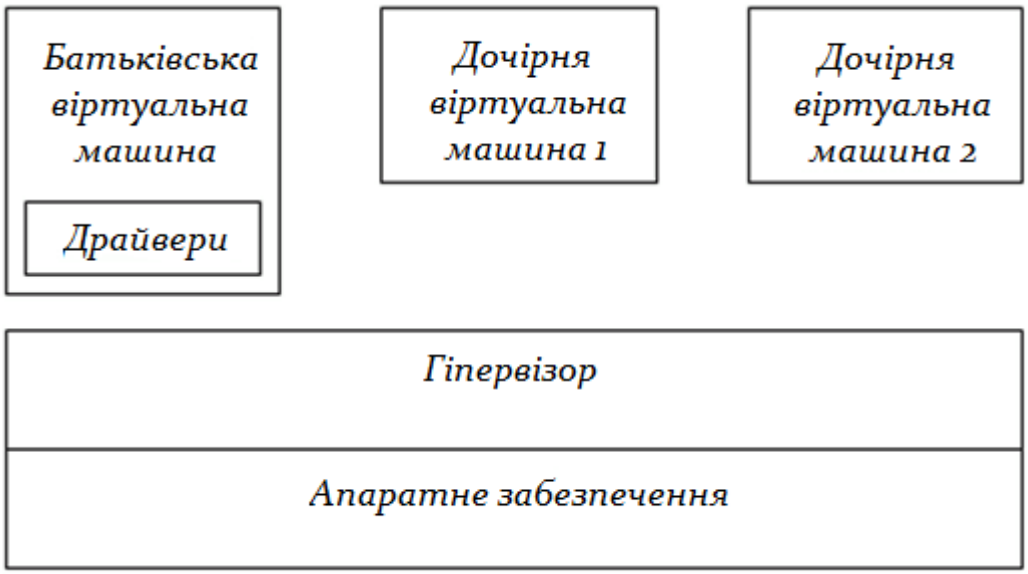


Рис. 2.4 Схематичне представлення мікроядерного гіпервізора

У мікроядерній моделі один розділ є батьківським (parent), решта - дочірніми (child). Розділ - найменша ізольована одиниця, підтримувана гіпервізором. Він складається з простору фізичних адрес і одного або декількох

віртуальних процесорів. Кожному розділу призначаються конкретні апаратні ресурси - частку процесорного часу, пам'ять, пристрої та ін. Батьківський розділ створює дочірні розділи і керує ними, а також містить стек віртуалізації (virtualizationstack), який використовується для управління дочірніми розділами. Перейти до основного розділу, є також кореневим (root), оскільки він створюється першим і володіє всіма ресурсами, що не належать Гіпервізор. Володіння всіма апаратними ресурсами означає серед іншого, що саме кореневої (тобто, батьківський) розділ керує живленням, підключенням самоналагоджувальних пристроїв, відає питаннями апаратних збоїв і навіть управляє завантаженням гіпервізора.

У батьківському розділі міститься стек віртуалізації - набір програмних компонентів, розташованих поверх гіпервізора і спільно з ним підтримують віртуальні машин. Стек віртуалізації обмінюється даними з гіпервізором і виконує всі функції по віртуалізації, які не підтримуються безпосередньо гіпервізором. Велика частина цих функцій пов'язана зі створенням дочірніх розділів і управлінням ними і необхідними їм ресурсами (ЦП, пам'ять, пристрої).

Даний тип гіпервізора має низку переваг у порівнянні з монолітними:

- сумісність з будь-яким обладнанням, драйвери якого розташовуються в рамках батьківської ОС.
- більш високий рівень безпеки.
- більш висока продуктивність гіпервізора, оскільки він не повинен взаємодіяти з драйверами пристроїв.

З іншого боку, мікроядерна модель може дещо програвати монолітній моделі в продуктивності. Однак в наші дні головним пріоритетом стала безпека, тому для більшості компаній цілком прийнятна втрата пари відсотків в продуктивності заради скорочення фронту нападу і підвищення стійкості.

Прикладом використання даного типу гіпервізора є Microsoft Hyper – V [1,9].

2.2 Мобільність віртуальних машин

Безсумнівно, властивість віртуальних машин, яка полягає в тому, що окрема віртуальна машина може бути за короткий проміжок часу згорнута, перенесена на іншу інфраструктуру і запущена зі збереженням своїх характеристик і топології, є важливою перевагою і дуже зручним рішенням. На прикладі рішень для віртуальних машин Hyper-V від Microsoft буде показано різні способи виконання цього процесу, залежно від версії гіпервізора.

Можливість переміщення віртуальних машин з одного вузла на інший підтримувалася з моменту появи самої першої версії Hyper-V. У ранніх версіях Hyper-V (при використанні Windows Server 2008) віртуальні машини можна було переміщати тільки у вимкненому режимі (Рис. 2.5): Необхідно було виключити віртуальну машину, перенести її, а потім знову включити, для цього використовувалися функції експорту та імпорту. В результаті досягалася певна мобільність, але простої віртуальних машин при цьому були неминучі.

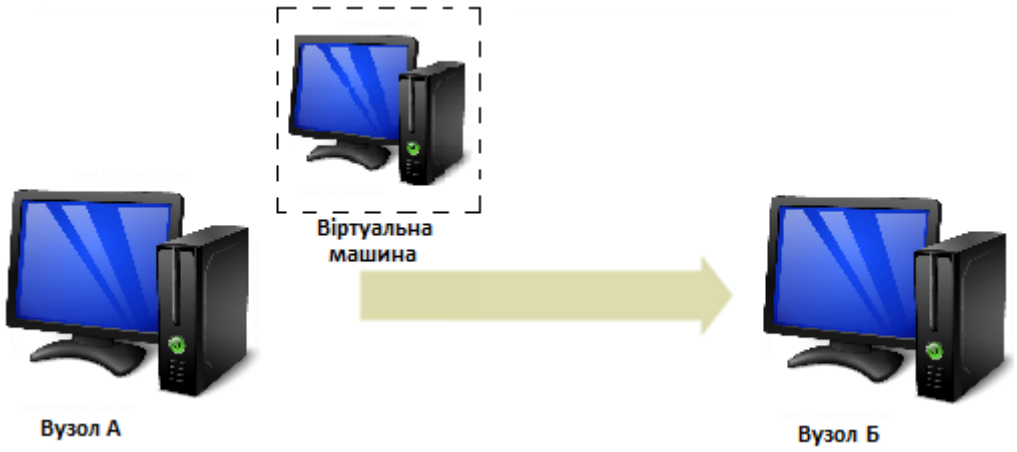


Рис. 2.5 Переміщення віртуальної машини в автономному режимі

З версії Windows Server 2008 R2 з'явилася можливість динамічного переміщення - можливість переміщати працюючу віртуальну машину. Втім, динамічне переміщення було можливо тільки між вузлами кластера Hyper-V, в яких віртуальні машини знаходилися в загальному томі кластера (CSV), як показано на Рис. 2.6:

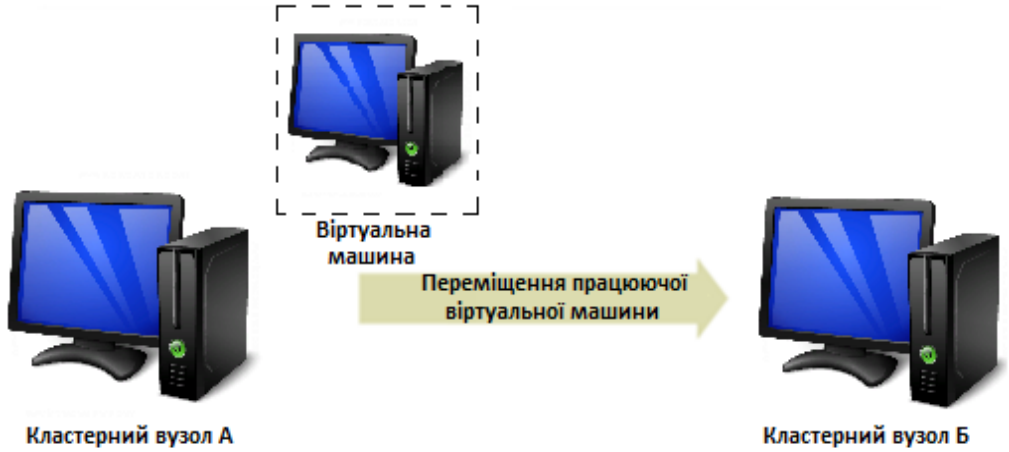


Рис. 2.6 Динамічне переміщення віртуальної машини

У Windows Server 2012 був досягнутий абсолютно новий ступінь свободи: можливість переміщати працюючі віртуальні машини між будь-якими вузлами Hyper-V однакової версії (Рис. 2.7) Незалежно від того, чи входять вони до складу відмовостійкого кластеру.

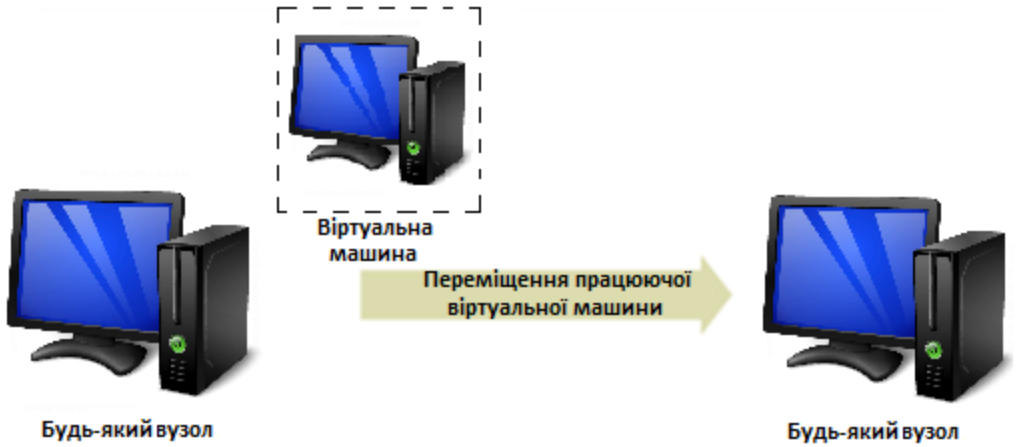


Рис. 2.7 Переміщення між будь-якими вузлами з однаковою операційною системою

У Windows Server 2012 R2 можливості були знову розширені - вперше стало можливо переміщати працюючі віртуальні машини між різними версіями вузлів. Тепер можна було переносити працюючі віртуальні машини з будь-якого вузла Windows Server 2012 на будь-який вузол Windows Server 2012 R2 незалежно від приналежності до відмовостійкого кластеру (Рис. 2.8).

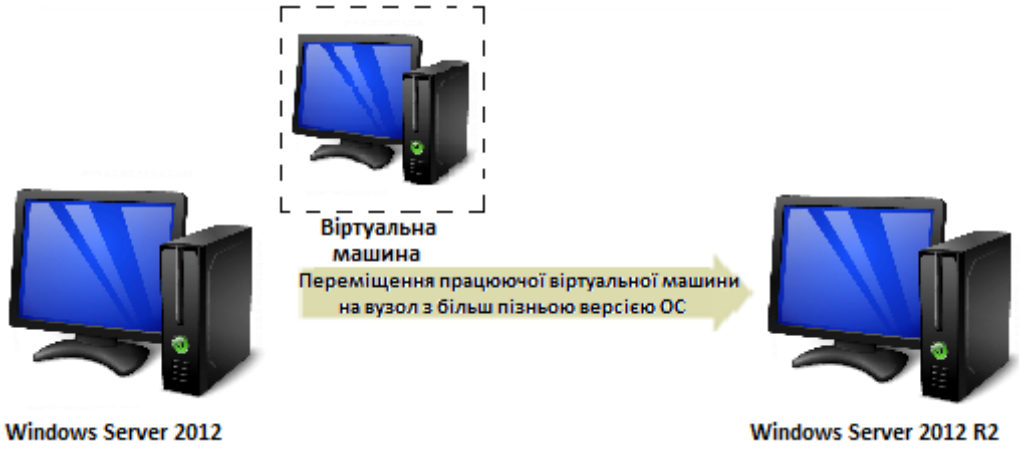


Рис. 2.8 Динамічне переміщення з вузла Windows Server 2012 на вузол Windows Server 2012 R2

У Windows Server 2016 подолано ще одне обмеження: тепер можна переміщати віртуальні машини на вузли не тільки з більш пізніми, але і з більш ранніми версіями, завдяки чому адміністратори отримали справжню свободу дій при управлінні віртуальними машинами. Раніше динамічне переміщення було можливо або в разі, якщо на обох вузлах була однакова версія, або при переміщенні на вузол з наступною, більш пізньою версією Windows Server.

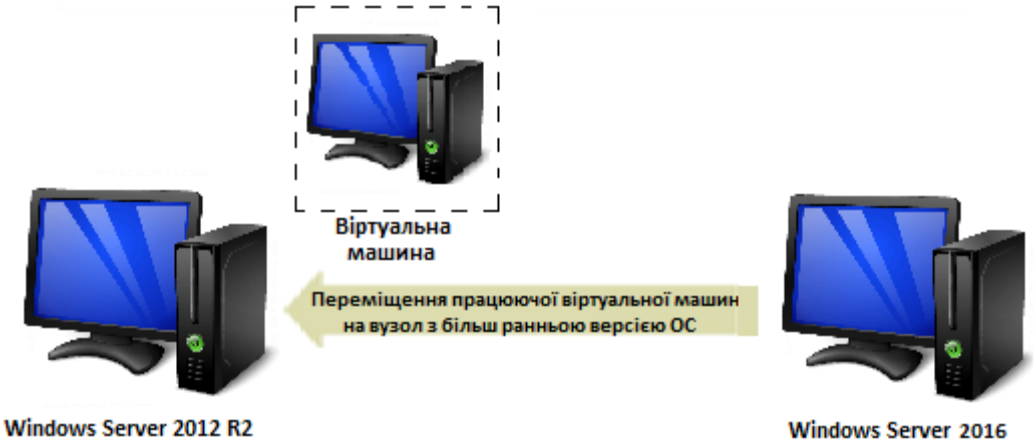


Рис. 2.9 Переміщення віртуальної машини з вузла Windows Server 2016 на вузол з більш ранньою версією Windows Server

Windows Server 2016 - єдина версія, в якій доступна можливість динамічно переміщати працюючу віртуальну машину на вузол з більш ранньою версією Windows Server (Рис. 2.9).

Для динамічного переміщення працюючих віртуальних машин з вузла Windows Server 2016 на вузли з більш ранніми версіями Windows Server слід дотримуватись таких умов:

– обидва вузла повинні бути членами одного і того ж каталогу Active Directory;
– на обох вузлах повинна бути включена функція динамічного переміщення [10].

2.3 Існуючі реалізації віртуальних машин

VMware

Компанія VMware - один з перших гравців на ринку платформ віртуалізації. У 1998 році VMware запатентувала свої програмні техніки віртуалізації і з тих пір випустила чимало ефективних і професійних продуктів для віртуалізації різного рівня: від VMware Workstation, призначеного для настільних ПК, до VMware ESX Server, що дозволяє консолідувати фізичні сервери підприємства у віртуальній інфраструктурі.

На відміну від ЕОМ (мейнфрейм), пристрої на базі x86 не підтримують віртуалізацію в повній мірі.

Тому компанії VMware довелося подолати чимало проблем в процесі створення віртуальних машин для комп'ютерів на базі x86. Основні функції більшості ЦП (в ЕОМ і ПК) полягають у виконанні послідовності збережених інструкцій (тобто програм). У процесорах на базі x86 містяться 17 особливих інструкцій, що створюють проблеми при віртуалізації, через які операційна система відображає попередження, перериває роботу програми або просто видаетобщій збій. Отже, ці 17 інструкцій виявилися значною перешкодою на початковому етапі впровадження віртуалізації для комп'ютерів на базі x86.

Для подолання цієї перешкоди компанія VMware розробила адаптивну технологію віртуалізації, яка "перехоплює" дані інструкції на етапі створення і перетворює їх в безпечні інструкції, придатні для віртуалізації, не зачіпаючи при цьому процеси виконання всіх інших інструкцій. В результаті ми

отримуємо високопродуктивну віртуальну машину, відповідну апаратного забезпечення вузла і підтримуючу повну програмну сумісність. Компанія VMware першої розробила і впровадила дану інноваційну технологію, тому на сьогоднішній день вона є незаперечним лідером технологій віртуалізації.

VMware Workstation - платформа, орієнтована на desktop-користувачів і призначена для використання розробниками ПЗ, а також професіоналами в сфері IT. Нова версія популярного продукту VMware Workstation 7 стала доступна в 2009 р, в якості хостових операційних систем підтримуються Windows і Linux. VMware Workstation 7 може використовуватися спільно з середовищем розробки, що робить її особливо популярною в середовищі розробників, викладачів і фахівців технічної підтримки. Вихід VMware Workstation 7 означає офіційну підтримку Windows 7 як в якості гостьової, так і хостовой операційної системи. Продукт включає підтримку Aero Peek і Flip 3D, що робить можливим спостерігати за роботою віртуальної машини, підбиваючи курсор до панелі завдань VMware або до відповідної вкладці на робочому столі хоста. Нова версія може працювати на будь-якій версії Windows 7, також як і будь-які версії Windows можуть бути запущені в віртуальних машинах. Крім того, віртуальні машини в VMware Workstation 7, повністю підтримують Windows Display Driver Model (WDDM), що дозволяє використовувати інтерфейс Windows Aero в гостьових машинах.

VMware Player - безкоштовний "програвач" віртуальних машин на основі віртуальної машини VMware Workstation, призначений для запуску вже готових образів віртуальних машин, створених в інших продуктах VMware, а також в Microsoft VirtualPC і Symantec LiveState Recovery. Починаючи з версії 3.0 VMware Player дозволяє також створювати образи віртуальних машин. Обмеження функціональності тепер стосується в основному функцій, призначених для IT-фахівців і розробників ПЗ.

VMware Server - безкоштовний продукт VMware Server, який є досить потужною платформою віртуалізації, яка може бути запущена на серверах під управлінням хостових операційних систем Windows і Linux. Основне

призначення VMware Server - підтримка малих і середніх віртуальних інфраструктур невеликих підприємств. У зв'язку з невеликою складністю його освоєння і установки, VMware Server може бути розгорнутий в найкоротші терміни, як на серверах організацій, так і на комп'ютерах домашніх користувачів.

VMware vSphere – комплекс продуктів, що представляє надійну платформу для віртуалізації ЦОД. Компанія позиціонує даний комплекс також як потужну платформу віртуалізації для створення і розгортання приватного "хмари". VMware vSphere поставляється в декількох випусках з можливостями, призначеними спеціально для малих підприємств і середніх компаній і корпорацій.

VMware vSphere включає ряд компонентів, що перетворюють стандартне обладнання в загальну стійку середу, нагадує мейнфрейм і включає вбудовані елементи управління рівнями обслуговування для всіх додатків:

- Служби інфраструктури - це компоненти, що забезпечують всебічну віртуалізацію ресурсів серверів, сховищ і мереж, їх об'єднання та точне виділення додатків на вимогу і відповідно до пріоритетів бізнесу.
- Служби додатків - це компоненти, що надають вбудовані елементи управління рівнями обслуговування для всіх додатків на платформі платформи vSphere незалежно від їх типу або ОС.
- VMware vCenter Server надає центральну консоль для управління віртуалізацією, що забезпечує адміністрування служб інфраструктури і додатків. Ця консоль підтримує всебічну візуалізацію всіх аспектів віртуальної інфраструктури, автоматизацію повсякденної експлуатації і масштабованість для управління великими середовищами ЦОД.

VMware ESX Server – це гіпервізор, який розбиває фізичні сервери на безліч віртуальних машин. VMware ESX є основою пакету VMware vSphere і входить в усі випуски VMware vSphere.

Citrix (Xen)

Розробка некомерційного гіпервізора Xen починалася як дослідницький проект комп'ютерної лабораторії Кембриджського університету. Засновником проекту і його лідером був Іан Пратт (Ian Pratt) співробітник університету, який створив згодом компанію XenSource, що займається розробкою комерційних платформ віртуалізації на основі гіпервізора Xen, а також підтримкою Open Source співтовариства некомерційного продукту Xen. Спочатку Xen був найрозвинутішу платформу, що підтримує технологію паравіртуалізації. Ця технологія дозволяє Гіпервізор в хостовой системі управляти гостьовий ОС за допомогою гіпервізорів VMI (Virtual Machine Interface), що вимагає модифікації ядра гостьової системи. На даний момент безкоштовна версія Xen включена в дистрибутиви декількох ОС, таких як Red Hat, Novell SUSE, Debian, Fedora Core, Sun Solaris. В середині серпня 2007 року компанія XenSource була поглинена компанією Citrix Systems. Сума проведеної операції близько 500 мільйонів доларів (акціями та грошовими коштами) говорить про серйозні наміри Citrix щодо віртуалізації. Експерти вважають, що не виключена і покупка Citrix компанією Microsoft, враховуючи давнє її співпрацю з XenSource.

Citrix XenApp - призначений для віртуалізації і публікації додатків з метою оптимізації інфраструктури доставки сервісів у великих компаніях. XenApp має величезну кількість користувачів по всьому світу і в багатьох компаніях є ключовим компонентом IT-інфраструктури.

Citrix XenServer - платформа для консолідації серверів підприємств середнього масштабу, що включає основні можливості для підтримки віртуальної інфраструктури. Виробник позиціонує даний продукт як рішення Enterprise-рівня для віртуалізації серверів, що підтримує роботу в "хмарному" оточенні.

Citrix XenDesktop - рішення по віртуалізації десктопів підприємства, що дозволяє централізовано зберігати і доставляти робочі оточення в віртуальних машинах користувачам. Продукт поддерживаетнесколько сценаріїв доставки

додатків на настільні ПК, тонкі клієнти і мобільні ПК ісовместім з серверними віртуалізаційних рішеннями конкурентів.

Microsoft

Для Microsoft все почалося, коли в 2003 році вона придбала компанію Connectix, одну з небагатьох компаній виробляє програмне забезпечення для віртуалізації під Windows. Разом з Connectix, компанії Microsoft дістався продукт Virtual PC, конкурував тоді з розробками компанії VMware щодо настільних систем віртуалізації. За великим рахунком, Virtual PC надавав тоді така кількість функцій, що і VMware Workstation, і при належній увазі міг би бути в даний час повноцінним конкурентом цієї платформи. Однак з того часу, компанія Microsoft випускала по мінорному релізу в рік, не приділяючи особливої уваги продукту Virtual PC, в той час як VMware стрімко розвивала свою систему віртуалізації, перетворивши її по-справжньому в професійний інструмент. Усвідомивши своє технологічне відставання в сфері віртуалізації серверних платформ, компанія Microsoft випустила продукт Virtual Server 2005, націлений на створення і консолідацію віртуальних серверів організацій. Однак було вже пізно. Компанія VMware вже захопила лідерство в цьому сегменті ринку, пропонуючи в той момент дві серверні платформи віртуалізації VMware GSX Server і VMware ESX Server, кожна з яких за багатьма параметрами перевершувала платформу Microsoft. Остаточний удар був нанесений в 2006 році, коли VMware фактично оголосила продукт VMware GSX Server безкоштовним, взявшись за розробку продукту VMware Server на його основі і сконцентрувавши всі зусилля на продажах потужної корпоративної платформи VMware ESX Server в складі віртуальної інфраструктури Virtual Infrastructure 3. У компанії Microsoft був тільки єдиний вихід у цій ситуації: в квітні 2006 року вона також оголосила про безкоштовність продукту Microsoft Virtual Server 2005. також існували раніше два видання Standard Edition і Enterprise Edition були об'єднані в одне - Microsoft Virtual Server Enterprise Edition. З тих пір Microsoft істотно змінила стратегію щодо віртуалізації, і влітку 2008 року був випущений фінальний

реліз платформи віртуалізації Microsoft Hyper-V, інтегрованої в ОС Windows Server 2008. Тепер роль сервера віртуалізації доступна всім користувачам нової серверної операційної системи Microsoft.

Віртуалізація серверів

Hyper-V. Продукт Microsoft позиціонується як основний конкурент VMware ESX Server в області корпоративних платформ віртуалізації. Microsoft Hyper-V являє собою рішення для віртуалізації серверів на базі процесорів з архітектурою x64 в корпоративних середовищах. На відміну від продуктів Microsoft Virtual Server або Virtual PC, Hyper-V забезпечує віртуалізацію на апаратному рівні, з використанням технологій віртуалізації, вбудованих в процесори. Hyper-V забезпечує високу продуктивність, практично рівну продуктивності однієї операційної системи, що працює на виділеному сервері. Hyper-V поширюється двома способами: як частина Windows Server 2008 або в складі незалежного безкоштовного продукту Microsoft Hyper-V Server.

Hyper-V є вбудованим компонентом 64-розрядних версій Windows Server 2008 Standard, Windows Server 2008 Enterprise і Windows Server 2008 Datacenter. Ця технологія недоступна в 32-розрядних версіях Windows Server 2008, в Windows Server 2008 Standard без Hyper-V, Windows Server 2008 Enterprise без Hyper-V, Windows Server 2008 Datacenter без Hyper-V, в Windows Web Server 2008 і Windows Server2008 для систем на базі Itanium.

Розглянемо коротко особливості архітектури Hyper-v. Hyper-v є гіпервізор, тобто прошарок між обладнанням і віртуальними машинами рівнем нижче операційної системи. Ця архітектура була спочатку розроблена IBM в 1960-і роки для мейнфреймів і недавно стала доступною на платформах x86 / x64, як частина низки рішень, включаючи Windows Server 2008 Hyper-V і Vmware ESX [1].

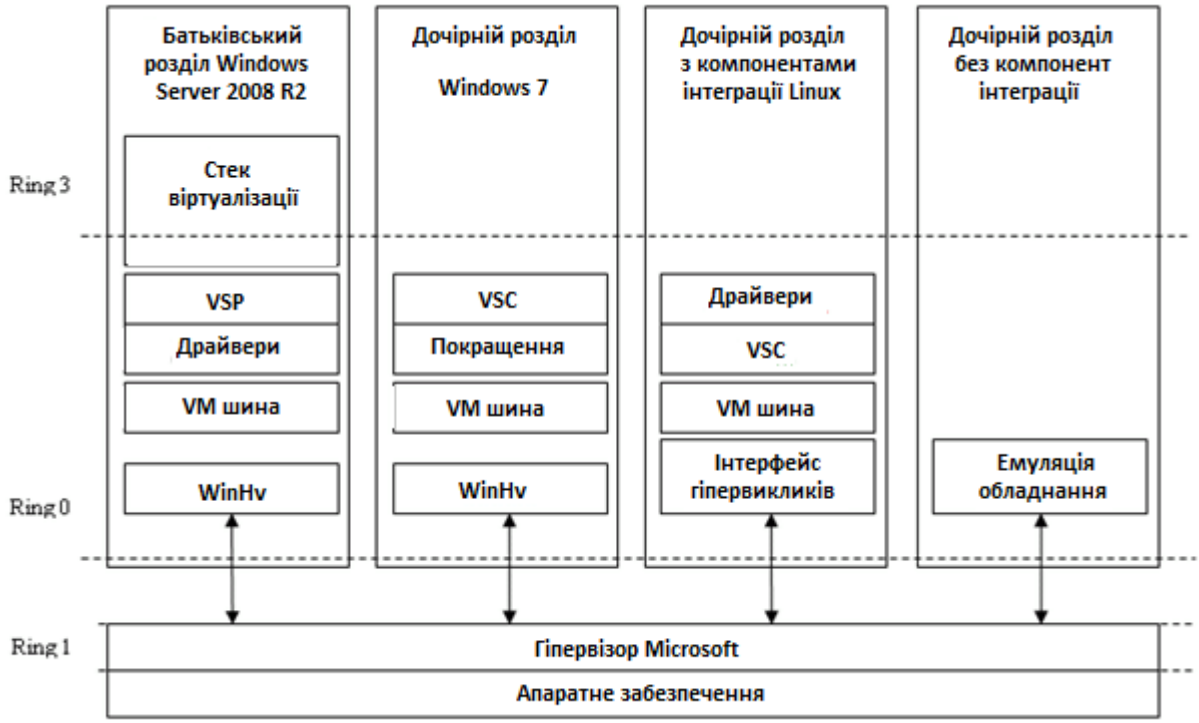


Рис. 2.10 Архітектура Hyper – V: Ring 0 – рівень ядра. Ring 1 – рівень гіпервізора. Ring 3 – рівень користувача.

Гіпервізор Hyper - V є гіпервізором першого типу. Рівнем вище за гіпервізор розташовуються батьківський і дочірній розділи - області ізоляції, в рамках яких працюють інформаційні системи. Хостової операційна система запускається в батьківському розділі, як і стек віртуалізації. Гостьові операційні системи розташовуються в дочірніх розділах. Кожен розділ пов'язаний з гіпервізором за допомогою інтерфейсу гіпервикликів.

Доступ к аппаратному обеспечению есть только у родительского раздела, дочерние могут взаимодействовать с "железом" только через родителя.

Батьківський розділ (parent partition)

Батьківський розділ технології віртуалізації Hyper - V містить ряд компонент, яких немає в дочірніх розділах.

Батьківський розділ створюється системою в першу чергу, як тільки гіпервізор починає роботу. Батьківський розділ створюється тільки для операційної системи Windows Server 2008 R2 з Hyper-V роллю.

Особливості батьківського розділу:

Батьківський розділ використовується для створення та управління дочірніми розділами системи і включає WMI провайдера, який надає інтерфейс для віддаленого адміністрування.

1. Батьківський розділ управляє і розподіляє апаратні ресурси, за винятком процесу фізичного розподілу пам'яті, який здійснюється гіпервізором.
2. Апаратні ресурси батьківського розділу є загальними і виділяються для використання дочірніми розділами.
3. Батьківський розділ управляє харчуванням, "plug and play" - операціями і веде журнали апаратних збоїв.

Стек віртуалізації

Ряд компонент, що розташовується в батьківському розділі, називається стеком віртуалізації. Стек віртуалізації має прямий доступ до апаратного забезпечення хостового комп'ютера.

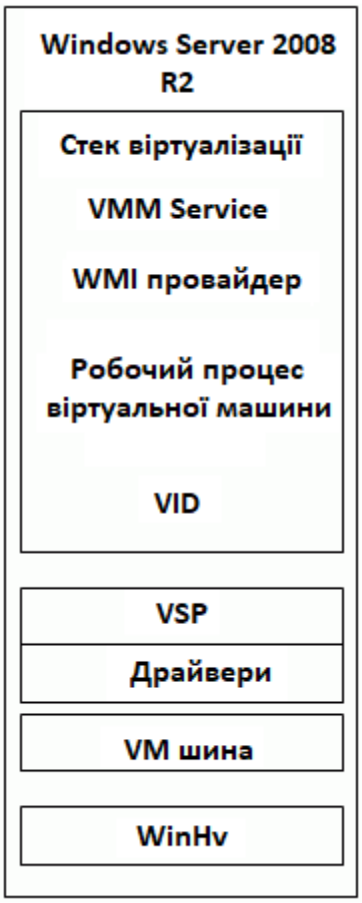


Рис. 2.11 Структура батьківського розділу

Стек віртуалізації складається з наступних компонент:

- Virtual Machine Management Service.
- Virtual Machine Worker Process (робочий процес віртуальної машини).
- Virtual Devices.
- Virtualization Infrastructure Driver.
- Windows Hypervisor Interface Library.

Дочірній розділ (child partition)

Як уже зазначалося, в рамках дочірніх розділів функціонують гостьові операційні системи. Гіпервізор першого типу підтримує три основні типи дочірніх розділів:

1. З операційною системою сімейства Windows і встановленими компонентами інтеграції;
2. З операційною системою, відмінною від сімейства Windows і з встановленими компонентами інтеграції;
3. З операційною системою, що не підтримує компоненти інтеграції.

У кожному із зазначених випадків, набір компонент дочірнього розділу буде різним.

Дочірній розділ з Windows і встановленими компонентами інтеграції містить наступні компоненти:

- Клієнти служб віртуалізації (VSC) - пристрої дозволяють дочірнім розділах отримати доступ до апаратних ресурсів.
- Покращення - модифікації в коді операційної системи.

Дочірній розділ з встановленими компонентами інтеграції і відмінною від Windows операційної системи використовують сторонні клієнти служб віртуалізації для отримання доступу до апаратних ресурсів.

Дочірній розділ без компонент інтеграції повинні емулювати пристрої замість VSC, що негативно позначається на продуктивності [11].

Віртуалізація настільних комп'ютерів

В рамках даного підходу до віртуалізації Microsoft пропонує три технології:

- **Microsoft Virtual PC** - компонент операційної системи Windows 7, що дозволяє користувачам запускати кілька операційних систем на одному комп'ютері. Включає в себе режим Windows XP mode, який представляє собою вже налаштовану віртуальну машину з операційною системою Windows XP + SP3.
- **Microsoft Enterprise Desktop Virtualization** - рішення для корпоративної віртуалізації робочих станцій, дозволяє адміністраторам створювати і управляти корпоративними образами віртуальних машин на всіх робочих станціях під керуванням Windows.
- **Microsoft Application Virtualization** - рішення для перетворення додатків в централізовано керовані віртуальні служби.

Віртуалізація віддалених робочих столів

Особливістю віртуалізації віддалених робочих столів є те, що сама віртуальне середовище виконується на сервері. Рішення Microsoft в даній області:

- **Служба віддалених робочих столів** - являє собою колишню службу терміналів. Включає в себе можливість надання користувачам віртуальних машин по протоколу RDP (Remote Desktop Protocol).
- **Microsoft Application Virtualization** для служб віддалених робочих столів - рішення, що дозволяє перетворювати додатки в централізовано керовані віртуальні служби і надавати їх користувачам за допомогою протоколу RDP.
- **Інфраструктура віддалених робочих столів** - архітектурна модель, що включає в себе Hyper - V, Microsoft Desktop Optimization Pack і Microsoft System Center. Користувачам надається доступ до особистого віртуального віддаленого столу, за допомогою протоколу RDP [12].

РОЗДІЛ 3
КОНТЕЙНЕРНА ВІРТУАЛІЗАЦІЯ ТА ВАРІАНТИ ЇЇ РЕАЛІЗАЦІЇ

3.1 Архітектурні особливості контейнерної віртуалізації

Технология контейнеризации далеко не новая и не новаторская технология. Она используется уже довольно длительное время. Тем не менее, данная технология стала набирать значительную популярность с приходом облачных технологий. Недостатки традиционных виртуальных машин стали катализатором роста популярности контейнеров. Поставщики инструментов для работы с контейнерами, например, Docker, в значительной степени упростили технологию контейнеризации, что способствовало внедрению данной технологии в широкие массы. Популярность DevOps и микросервисной архитектуры также ускорила перерождение технологии контейнеризации [7].

Технологія контейнерної віртуалізації є так званою технологією «віртуалізації рівня операційної системи». Вона не використовує гіпервізор, але тим не менш надає можливість одночасного запуску декількох екземплярів ОС на одному обладнанні.

Ці ОС використовують ядро хостовой ОС, а їх одночасна робота забезпечується різними механізмами ізоляції і поділу ресурсів, за які відповідає ядро хостовой системи і так званий віртуалізаційних шар (container engine). В даному підході, ядро операційної системи надає ізольоване віртуальний простір. Кожне з віртуальних просторів називається контейнером.

Архітектура системи, що використовує контейнерну віртуалізацію представлена на Рис. 3.1 [13]:

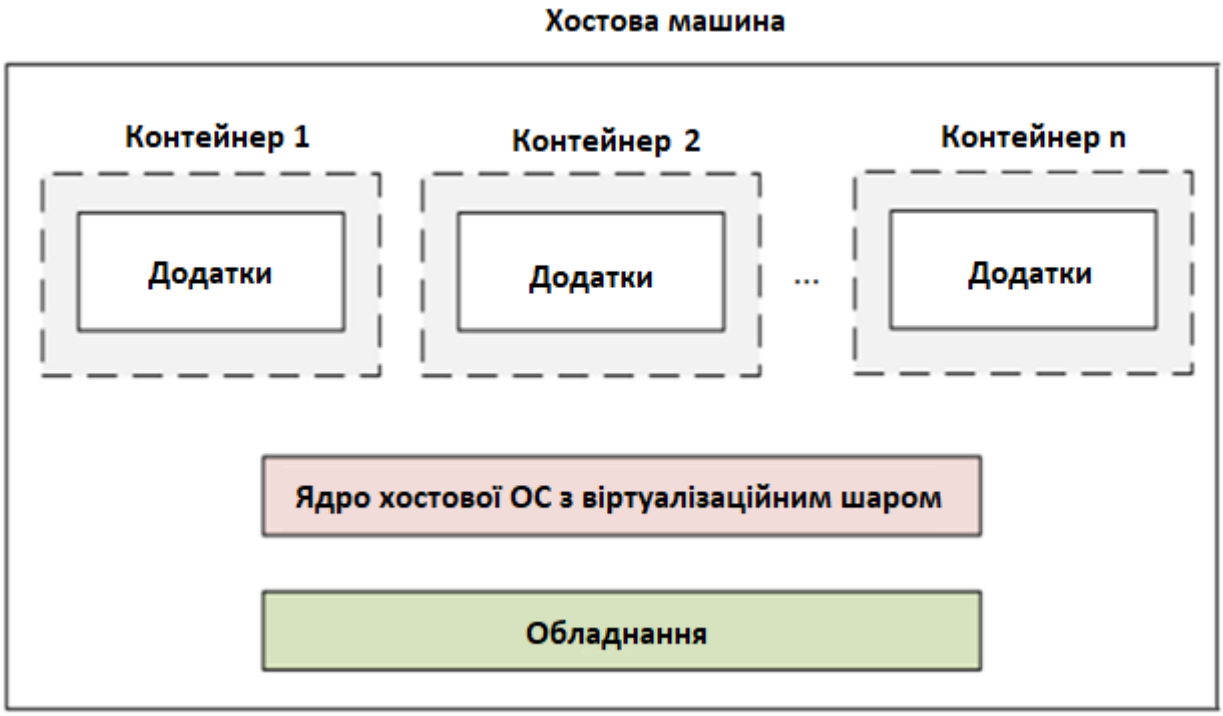


Рис. 3.1 Схематичне представлення контейнерної віртуалізації

Віртуальні контейнери є ізольованим середовищем, в якій можна запускати додатки, не побоюючись того, що при цьому зміняться інші додатки або параметри. У контейнерів є загальні компоненти (ядро, системні драйвери тощо.), завдяки чому прискорюється їх запуск і забезпечується більш висока щільність, ніж при використанні віртуальних машин.

Як видно з малюнка, в операційній системі фізичного сервера може бути розміщено безліч контейнерів. При цьому вони повністю ізольовані один від одного, але разом використовують основні компоненти операційної системи, наприклад, ядро. Інтерес представляє поведінку додатків в контейнерах. У додатку може бути ряд залежних компонентів, необхідних для його роботи. Такі залежні компоненти можуть існувати тільки всередині того ж контейнера, в якому знаходиться додаток. Це означає, що в разі будь-яких неполадок програми та файлів, від яких воно залежить, в контейнері 1, робота додатки в контейнері 2 (і його файлів) не буде порушена. Якщо в звичайному середовищі додаток 1, наприклад, видалить реєстр, це порушить роботу і додатки 1 і додатки 2. У разі якщо використовуються контейнери, додатки 1 і 2 один від одного ізольовані, тому зміна реєстру додатком 1 ніяк не вплине на додаток 2.

Всі виконавчі файли і залежні компоненти розміщені всередині контейнера, тому додаток, що працює в контейнері, повністю переноситься. Це означає, що контейнер можна розгорнути на будь-якому вузлі, на якому запущений диспетчер контейнерів, після чого контейнер можна запустити і використовувати без будь-яких додаткових змін і налаштувань. Наприклад, розробник може почати розгортати додаток і розгорнути його в контейнері Hyper-V в Windows 10. Коли додаток буде готове до розгортання в робочому середовищі, його можна запустити в Windows Server 2016, в тому числі в Nano Server, в загальнодоступному, приватному або гібридній хмарі.

Контейнери складаються з декількох рівнів. Перший рівень є базовим. Це образ операційної системи, на основі якого будуються всі інші рівні. Образ зберігається в репозиторії образів, тому в потрібних випадках можна скористатися посиланням на нього. Наступним (і іноді останнім) рівнем є рівень платформи додатків, який може бути загальним для всіх додатків. Наприклад, якщо базовим рівнем є ядро Windows Server, то рівнем платформи додатків можуть бути .NET Framework і Internet Information Services (IIS). Другий рівень теж може зберігатися у вигляді образу, який при виклику також описує залежність від базового рівня - ядра Windows Server. І нарешті, рівень додатків: на цьому рівні зберігається сам додаток, що посилається на рівень платформи додатків і, в свою чергу, на базовий рівень.

На базовий рівень і на рівень платформи додатків можуть в будь-який час посилатися всі інші створювані контейнери додатків. Кожен рівень доступний тільки для читання, виключаючи верхній рівень, що розгортається образу. Наприклад, якщо розгортається контейнер, що залежить тільки від способу ядра Windows Server, цей рівень ядра Windows Server буде верхнім рівнем контейнера; для збереження всіх операцій запису і змін, зроблених під час виконання, створюється пісочниця. Після цього зроблені зміни можна зберігати в якості іншого способу для використання в подальшому. Цей же принцип застосовується і при розгортанні образу з рівнем платформи додатків:

цей рівень отримає власну пісочницю, а при розгортанні програми можна зберегти цю пісочницю у вигляді образу для подальшого використання.

Загальний принцип такий: при розгортанні контейнера на вузлі вузол сам визначає наявність базового рівня. Якщо цього рівня немає, вузол отримує його зі сховищ образів. Надалі цей процес повторюється для рівня платформи додатків, а потім створюється контейнер додатки, який потрібно розгорнути. Якщо після цього буде потрібно створити інший контейнер з такими ж залежностями, то буде досить виконати команду на створення нового контейнера додатків. Цей контейнер буде підготовлений до роботи практично миттєво, оскільки все залежить вже є в наявності. Якщо є контейнер додатки, що залежить від іншого рівня платформи додатків і від вихідного базового рівня ядра Windows Server, то можна просто отримати інший контейнер платформи додатків зі сховища образів і запустити новий контейнер додатки [10].

Контейнери, це простий механізм для збірки і постачання слабосвязанних компонентів програмного забезпечення. У загальному випадку, контейнери упаковують всі виконувані файли і бібліотеки, які необхідні для запуску програми. Контейнери повністю ізолюють наступні елементи:

- файлову системи;
- IP адреса;
- мережеві інтерфейси;
- внутрішні процеси;
- простору імен;
- бібліотеки операційної системи;
- бінарні файли програми;
- залежності;
- файли конфігурації програми [7].

3.2 Існуючі рішення в області контейнерної віртуалізації

OpenVZ

OpenVZ є реалізацією віртуалізації рівня операційної системи з використанням контейнерів від компанії Parallels. Для роботи системи необхідна спеціальна версія ядра Linux.

Управління контейнерами здійснюється з хостової ОС. Так само її іноді називають CT0, VE0 (нульовий контейнер). Контейнери в технології OpenVZ, так і називаються «контейнерами» (CT, Container), а також гостьовими оточеннями (VE, Virtual Enviroment). Кожен контейнер має доступ до свого пулу ресурсів, який може бути обмежений як знизу, так і зверху.

Створення контейнерів здійснюється за допомогою так званих темплейтов - шаблонів контейнерів. Технологія дозволяє створювати і налаштовувати групи ресурсів, а також окремі від них дискові квоти, квоти на споживання CPU, і пріоритети і / о. Майже всі ресурси можуть бути гарантованими, і негарантованими (зайві наявні ресурси можуть перерозподілятися між контейнерами). Можливі також і жорсткі ліміти на більшу частину ресурсів.

Наявність окремих ресурсів для кожного контейнера надає стійкість сервісів, а існування негарантованих ресурсів може дати продуктивність, так як дає можливість використовувати всю потужність фізичного сервера під сервіси, якщо він простоює. Номери процесів всередині контейнерів віртуалізувати: pid процесу init всередині контейнера завжди дорівнює одному, хоча на хостовой системі у нього, як і у інших процесів контейнера, буде інший номер.

OpenVZ надає майже повноцінний мережевий стек для кожного контейнера. Є можливість використовувати два види мережевих інтерфейсів: venet, більш продуктивний, і veth, що надає підтримку Ethernet mac-адрес для контейнерів.

Linux upstream containers (LXC)

Linux upstream containers є дуже схожою на OpenVZ технологією, але при цьому одна важлива відмінність - все необхідне для підтримки LXC вже впроваджено в стандартне ядро Linux.

По суті, LXC є об'єднанням використання технологій ядра Linux, що забезпечують ізоляцію різного роду ресурсів і набір утиліт для управління контейнерами.

Управління ресурсами по суті аналогічно OpenVZ, але здійснюється вбудованим в ядро механізмом cgroups. Ізоляція простору процесів, мережевого стека і інших механізмів роботи ОС здійснюється завдяки технології namespaces.

В даний час, розробка OpenVZ і LXC тісно пов'язані, так як технології по суті представляють собою дві реалізації одного і того ж підходу [11].

Контейнери Windows Server і контейнери Hyper-V

У Windows Server 2016 є два типи контейнерів:

- контейнери Windows Server;
- контейнери Hyper-V.

Контейнери Windows Server можна вважати рівноцінними контейнерам Linux. Контейнери Windows Server ізолюють додатки на одному і тому ж вузлі. Кожен контейнер отримує власне представлення системи вузла, в тому числі ядра, процесів, файлових систем, реєстру і інших компонентів. Контейнери Windows Server працюють між рівнем користувальницького режиму і рівнем режиму ядра.

Контейнери Hyper-V створені на основі технології, що спирається на віртуалізацію з апаратною підтримкою. За рахунок апаратної віртуалізації додатки в контейнерах Hyper-V отримують середовище з високим ступенем ізоляції, в якій запущені контейнери жодним чином не можуть вплинути на операційну систему фізичного сервера. На Рис. 3.2 показаний принцип функціонування обох технологій контейнерів, доступних в Windows Server 2016.

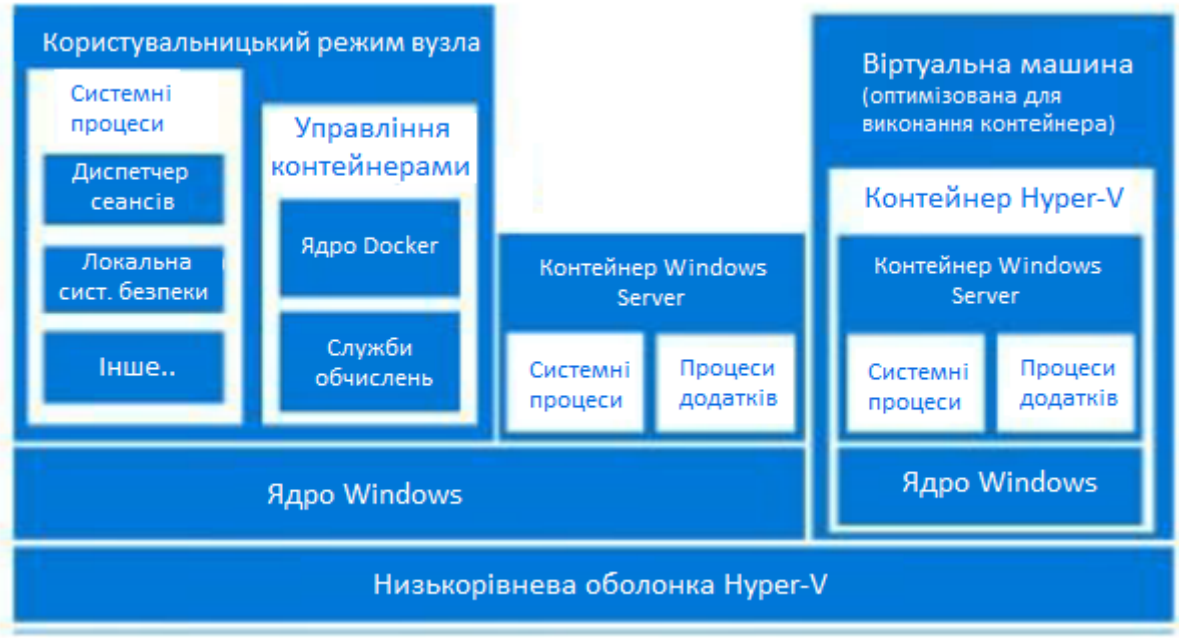


Рис. 3.2 Контейнери Windows Server 2016 і контейнери Hyper-V на одному і тому ж фізичному сервері

Як показано на малюнку, на одному фізичному сервері може одночасно використовуватися цілий набір з контейнерів Hyper-V, віртуальних машин і контейнерів Windows Server. Оскільки доступні два типи контейнерів, виникає питання, в яких випадках для розробки і розгортання додатків слід використовувати контейнери Windows Server, а в яких - контейнери Hyper-V. Вибір залежить від вимог додатка (або замовника, що використовує контейнери) до масштабування та ізоляції з апаратною підтримкою.

Наприклад, якщо потрібно розширене масштабування, його простіше досягти при використанні контейнерів Windows Server. Якщо ж потрібні розширені можливості ізоляції, доступні при віртуалізації з апаратною підтримкою, краще використовувати контейнери Hyper-V.

Незалежно від обраного типу контейнера додаток, розгорнуте в контейнері, буде сумісно з обома типами. Розробник може створити додаток, розмістити його в контейнері Windows Server, а потім перемістити в контейнер Hyper-V без яких би то не було змін. За рахунок цього забезпечується висока гнучкість, особливо якщо вимоги щодо масштабування або ізоляції змінюються в порівнянні з використалися при плануванні.

Мережеве підключення

Як показано на Рис. 3.3, кожен контейнер підключається за допомогою віртуального мережевого адаптера (контейнер Windows Server) або мережевого адаптера віртуальної машини (контейнер Hyper-V) до віртуального комутатора, налаштованому на вузлі. Кожен віртуальний мережевий адаптер ізольований від наступного і розглядається як окрема секція. Ці віртуальні мережеві адаптери підключаються до віртуального комутатора через порти (аналогічно Hyper-V). Віртуальний мережевий адаптер фізичного вузла ізольований від контейнерів. Сумісність із мережею до контейнерів Hyper-V є прозорими для віртуальної машини і проходять через мережевий адаптер віртуальної машини.

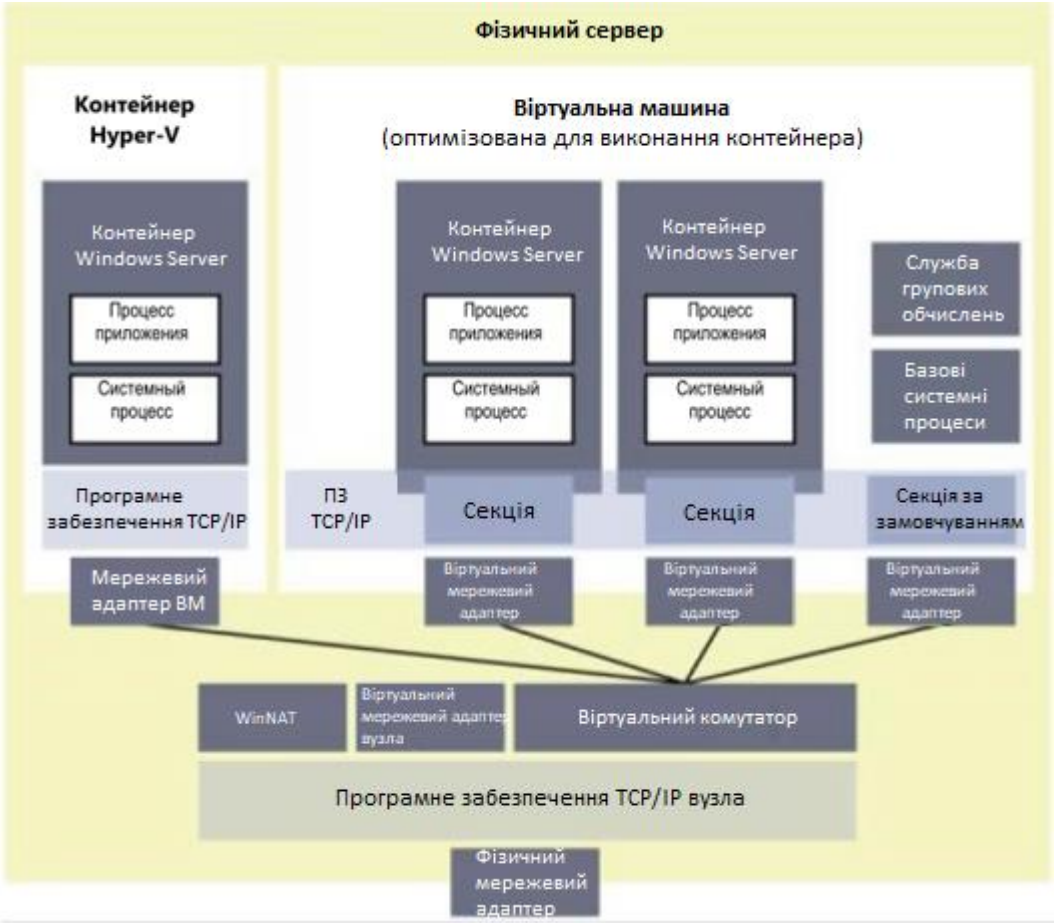


Рис. 3.3 Мережеве підключення контейнерів Windows Server і Hyper-V

Підключення до зовнішніх ресурсів можна реалізувати різними способами. Кожен з них залежить від конкретного сценарію, прийнятого для

використання контейнерів. Наприклад, якщо потрібно створити середовище контейнера для розробників, найкраще використовувати перетворення мережевих адрес (NAT) для мережі контейнерів. При цьому надається приватна IP-простір (IP-адреси призначаються DHCP-сервером), ізольоване від зовнішнього середовища. З'єднання між контейнерами обмежені, але підтримується переадресація портів в контейнерну середу, з якої ви працюєте. Весь трафік, що надходить на загальнодоступний IP-адреса NAT (IP-адреса зовнішнього мережного адаптера сервера) буде порівнюватися з таблицею, керованої через WinNAT, і передаватися в контейнер [10].

Docker

Docker також являє собою систему віртуалізації рівня операційної системи, що використовує контейнери. Спочатку, Docker базувався на технології LXC і представляв собою гнучкий і зручний інтерфейс управління контейнерами в поєднанні з системою контролю версій контейнерів, що робило його більш зручним при використанні контейнерної віртуалізації для розгортання та тестування додатків. При цьому, в Docker дещо відрізняється архітектура побудови віртуальної системи: тоді як в LXC прийнято поміщати в контейнер все дерево процесів, пов'язане з гостьової ОС, в один контейнер, в Docker ж додатка (процеси) по можливості поміщаються в окремі контейнери - шари (layers), між якими налаштовується обмежена взаємодія. З одного боку, це надає можливість більш гнучкого налаштування контейнерів, а з іншого ж ускладнює архітектуру системи. Схема організації контейнерів відповідно до ідеологією Docker зображена на Рис. 3.4 [13].

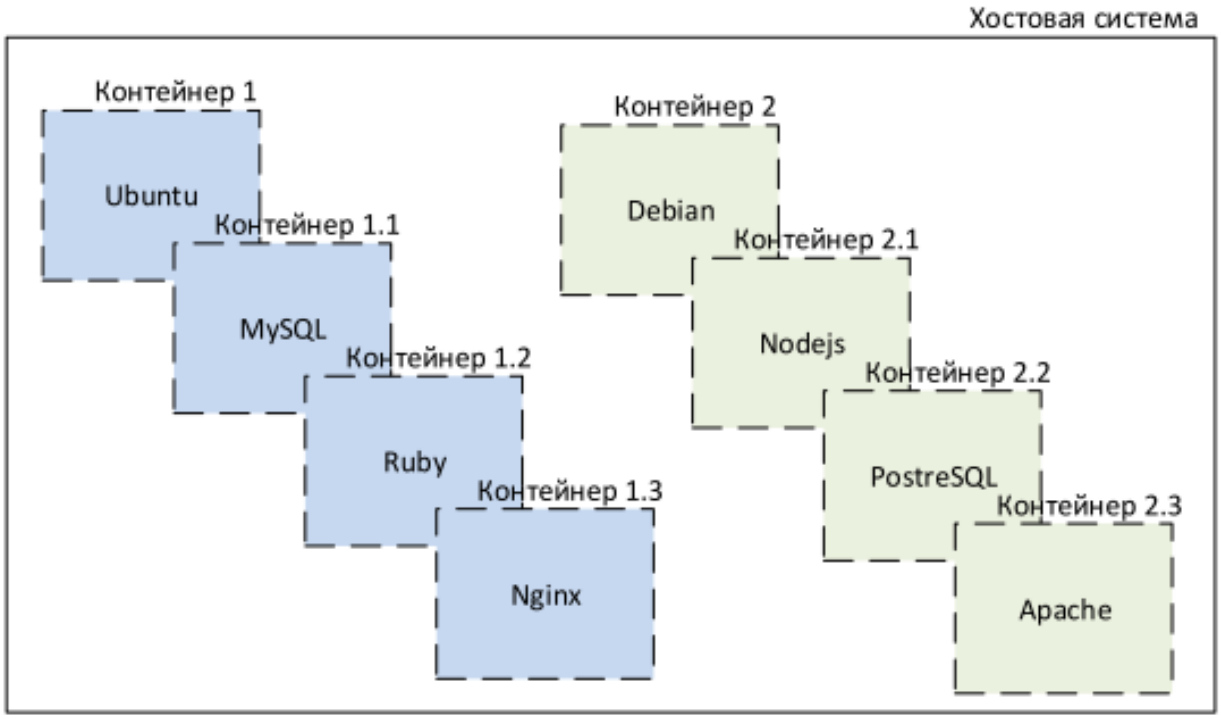


Рис. 3.4 Схематичне представлення шарів Docker

Платформа Docker вирішує три основні проблеми розгортання сервісів:

– доставка коду на сервер;
– запуск коду;
– однаковість оточення.

Docker дозволяє ізолювати сервіси від інфраструктури, таким чином досягається можливість доставляти їх набагато швидше. Docker дозволяє управляти інфраструктурою за допомогою тих же принципів, які використовуються при управлінні додатками. При використанні інструментів Docker для доставки, тестування і розгортання, значно скорочується затримка між фіксацією нового коду в системі контролю версій і запуском його на сервері промислової експлуатації.

Docker надає можливість упаковувати і запускати сервіси в ізольованому оточенні, яке і називається контейнером. Дана ізольованість і безпеку дозволяє запускати кілька контейнерів на одному хості одночасно. Контейнери легкі, оскільки не вимагають роботи гіпервізора. Вони запускаються безпосередньо на ядрі хостової машини. Таким чином, досягається можливість запуску більшої кількості контейнерів, ніж віртуальних машин, на одному і тому ж

фізичному обладнанні. Також, Docker контейнери можна запускати в самих віртуальних машинах.

Docker надає наступні інструменти і платформу для управління життєвим циклом контейнерів. За допомогою Docker відбувається упаковка додатки і їх компонентів в контейнер. Після розробки, додаток може бути розгорнуто на сервері промислової експлуатації вручну або за допомогою оркестратораа контейнерів. Дана техніка розгортання одноманітна незалежно від того де розгортається додаток на промислову експлуатацію, будь то в локальному дата центрі, хмарному провайдера або в гібридному оточенні.

Архітектура Docker

Docker використовує клієнт-серверну архітектуру [7].

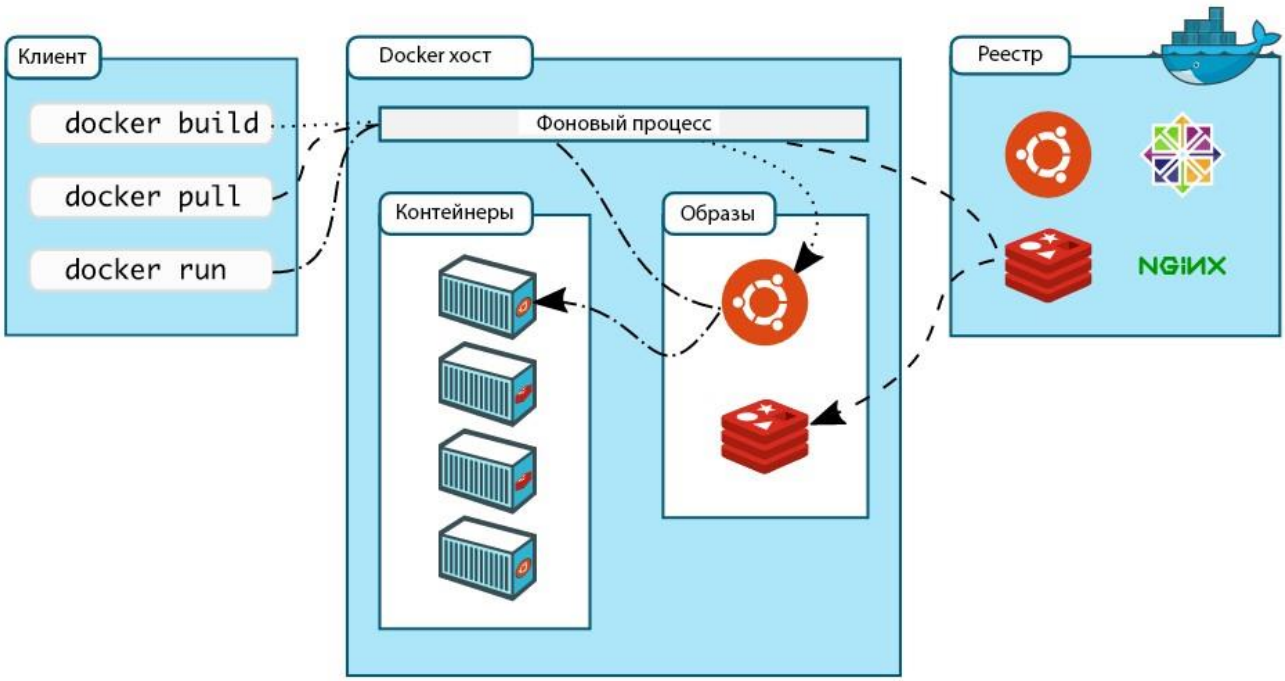


Рис. 3.5 Схематичне представлення архітектури Docker

Docker клієнт взаємодіє з фоновим процесом - сервером Docker, який в свою чергу запускає контейнери. Клієнт і фоновий процес можуть виконуватися в одній операційній системі, також є можливість підключити клієнта до віддаленого фонового процесу. Докер клієнт і фоновий процес взаємодіють через REST API поверх сокетів або через мережевий інтерфейс.

Фоновий процес Docker (dockerd) приймає запити і управляє об'єктами Docker. Фоновий процес може також взаємодіяти з іншими фоновими процесам.

Фоновий процес управляє наступними об'єктами:

– образи;
– контейнери;
– мережеву взаємодію;
– томи.

Docker клієнт основний спосіб взаємодії з сервером. При виконанні команди клієнт відправляє цю команду фонового процесу, який в свою чергу її виконує. Docker клієнт може взаємодіяти з безліччю різних серверів.

Реєстр контейнерів

Реєстр контейнерів зберігає образи. Даний реєстр може бути, як публічним, так і приватним. Docker Hub є офіційним відкритим реєстром, в якому зберігаються офіційні образи від різних провайдерів технологій, наприклад, ubuntu або nginx. Додавати образи в публічний реєстр може будь-хто. За замовчуванням Docker налаштований на пошук образів в цьому публічному реєстрі. Також є можливість створювати приватні реєстри образів.

При використанні команд docker pull або docker run, необхідні образи будуть завантажені із сконфігурованого реєстру. При використанні команди docker push, вказаний образ буде поміщений в реєстр.

Об’єкти Docker

При використанні Docker створюються і використовуються образи, контейнери, мережеве взаємодія, тому, плагіни і інші компоненти.

Образ - це незмінний шаблон, що містить інструкції для створення контейнера. У більшості випадків образ базується на іншому образі, доповнюючи його новою функціональністю. Наприклад, можна побудувати образ, який базується на образі ОС Linux Ubuntu, розширивши його встановивши веб-сервер Apache з розробленим додатком, а також сконфігурованої сервер для правильної роботи програми.

Є можливість створювати нові образи або ж перевикористати вже створені і розміщені в реєстрі. Щоб створити новий образ, потрібно створити конфігураційний файл, який називається DockerFile. Даний файл містить інструкції для створення і запуску образу. Кожна інструкція в конфігураційному файлі створює шар в образі. Коли конфігураційний файл змінюється і запущений процес повторної збірки способу, тільки ті верстви, які змінилися будуть пересобран. Саме ця технологія робить контейнери легковагими і швидкими в порівнянні з іншими технологіями віртуалізації.

Контейнер - це запущений екземпляр образу. Контейнери можна створювати, запускати, зупиняти і видаляти, використовуючи Docker API або інтерфейс командного рядка. Контейнер може бути підключений до однієї або кілька мереж. До нього може бути змонтована частина файлової системи хост машини. Також є можливість створити новий образ з поточного стану контейнера.

За замовчуванням, контейнери відносно добре ізольовані від інших контейнерів і хостової машини. Є можливість управляти ізоляцією мережевої взаємодії контейнерів і підключеними томами між іншими контейнерами і хостовой системою.

Контейнер створюється з образу, а також містить всі конфігураційні параметри, які надаються при його створенні та запуску. При зупинці контейнера, всі зміни в його стані, які не зафіксовано в постійному сховище пропадають.

3.3 Відмінності між віртуальними машинами та віртуальними контейнерами

Віртуальні машини такі як Hyper-V, VMWare і Zen були популярним вибором для віртуалізації дата-центрів кілька років тому. Промислові компанії значно заощадили свої кошти завдяки впровадженню віртуалізації в порівнянні з використанням серверів без віртуалізації. Віртуалізація також допомогла багатьом підприємствам оптимізувати використання своїх

існуючих інфраструктур. Оскільки віртуальні машини дозволяють автоматизувати процес побудови інфраструктури, багато компаній зіткнулися з необхідністю зменшення зусиль з управління цією автоматизації. Віртуальні машини також допомогли компаніям отримати ізольоване середовище для запуску додатків.

На перший погляд віртуалізація і контейнеризація мають однакові характеристики. Однак, контейнери та віртуальні машини - це не одне і теж. Таким чином твердження, що віртуальні машини і контейнери працюють однаково, некоректно. Віртуальні машини і контейнери абсолютно різні за своєю природою технології, також вони вирішують різні проблеми віртуалізації. Ці відмінності видно на Рис. 3.6.

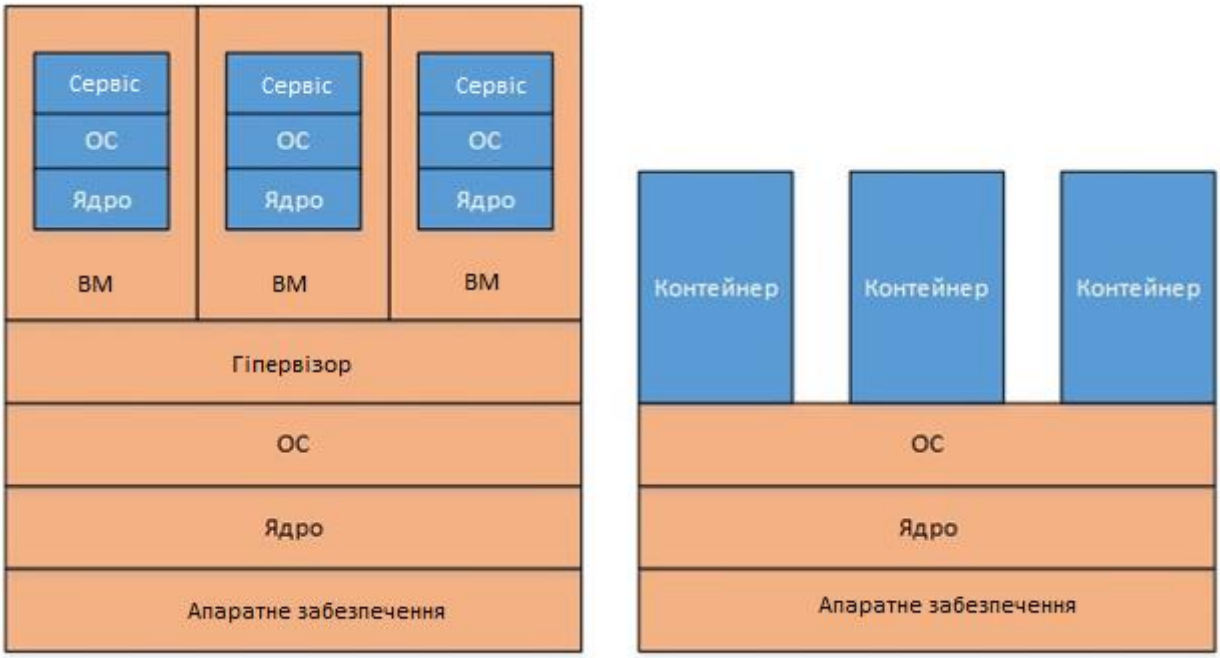


Рис. 3.6 Відмінність між віртуальною машиною і віртуальним контейнером

Віртуальні машини працюють на значно нижчому рівні ніж контейнери. Віртуальні машини надають віртуалізацію програмного забезпечення, наприклад, CPU, систем введення-виведення, пам'яті і так далі. Віртуальна машина ізольований компонент з вбудованою операційною системою, яка називається гостьовою операційною системою. Віртуальна машина містить в собі операційну систему цілком і виконує її без будь-якої залежності оточення

операційної системи, на якій запущений образ віртуальної машини. Оскільки віртуальна машина містить оточення операційної системи цілком, вона досить тяжеловесна. З одного боку, даний аспект можна вважати перевагою, а з іншого недоліком. Перевага полягає в повній ізоляції процесів, що виконуються на віртуальній машині. Недолік полягає в обмеженій кількості віртуальних машин, які можуть виконуватися на сервері, оскільки віртуальні машини резервують ресурси необхідні для їх роботи [14].

Розмір віртуальної машини безпосередньо впливає на час запуску і зупинки. Запуск віртуальної машини, в свою чергу, починає завантаження операційної системи, тому загальний час запуску віртуальної машини як правило високий. Віртуальні машини більш привабливі для співробітників компаній, які управляють інфраструктурою, оскільки не вимагають високого рівня кваліфікації.

У світі контейнерів, контейнери не емулюють апаратне забезпечення або операційну систему цілком. На відміну від віртуальних машин, контейнери використовують певні частини ядра як гостьовий операційної системи, так і хостовой операційної системи. Контейнери не використовують концепцію гостьовий операційної системи. Технологія контейнеризації надає ізольоване виконання оточення безпосередньо на хостовой операційній системі. У свою чергу, даний аспект також є одночасно перевагою і недоліком. Перевага полягає в легковажності і швидкості. Оскільки контейнери спільно використовують хостовую операційну системи, використання ресурсів контейнерами досить мало. В результаті на одній машині може бути запущено безліч невеликих контейнерів, чого не можна домогтися при використанні віртуальних машин. Оскільки контейнери запущені на одній хостовій операційній системі існують і обмеження. Наприклад, неможливо встановити настройки брандмауера (iptables) всередині контейнера. Процеси всередині контейнера повністю ізольовані і незалежні від процесів в інших контейнерах, що виконуються на одній хостовій системі.

На відміну від віртуальних машин, образи контейнерів публічно доступні на різних порталах. Це в значній мірі спрощує роботу розробників, оскільки вони не витрачають свій час на побудову образів з нуля. Можна скористатися базовими образами з сертифікованих джерел і додати додаткові шари програмних компонентів на базовий образ.

Легка природа контейнерів також відкриває безліч можливостей для їх автоматичного створення, публікації, завантаження і копіювання. Можливість завантажити, зібрати, доставити і запустити контейнер всього лише з використанням декількох команд або з використанням REST API більш підходящим вибором при розробці систем. Збірка нового контейнера займає кілька секунд. Таким чином збірка контейнера може стати етапом процесу безперервного постачання.

Таким чином у контейнерів є величезною перевага перед віртуальними машинами, але у віртуальних машин є свої сильні сторони. Більшість компаній використовують як контейнери, так і віртуальні машини, наприклад, для запуску контейнерів у віртуальній машині.

3.4 Переваги та недоліки контейнерної віртуалізації

Перший аспект в розрізі переваг контейнерів полягає в їх автономності. Контейнери упаковують бінарні файли і їх залежності спільно, щоб забезпечити відсутність відмінностей між різними оточеннями, такими як оточення розробки, тестування або промислової експлуатації.

Контейнери, в основному, мають менший розмір і менше споживання пам'яті. Найменший контейнер, Alpine, має розмір менше 5 мегабайт. Розмір контейнера набагато менше образу віртуальної машини. Розмір образу віртуальної машини зазвичай обчислюється в гігабайтах. Невелике споживання пам'яті контейнерами не тільки прискорюють запуск нових контейнерів, але також полегшують складання і поставку.

Оскільки розмір образу контейнера досить маленький і при запуску контейнера не починається операційна система, вони досить швидко

запускаються і зупиняються. Це властивість контейнерів полегшує масштабування, а також збільшують їх популярність при побудові хмарних додатків.

Контейнери підтримують переносимість між серверами і хмарними платформами. Як тільки контейнер створений і в нього включені всі його залежності, його можна перенести на інші сервера або на інші хмарні платформи без попередньої настройки хостової операційної системи.

Безліч ліцензій платного програмного забезпечень оперують кількістю фізичних ядер процесора, на яких вони запущені. Оскільки контейнери спільно використовують операційну систему і не використовують віртуалізацію фізичних ресурсів, вони не потрапляють під описані вище правила ліцензування.

Мінімальне використання пам'яті контейнерами дозволяють автоматизувати процес складання і публікувати контейнери в репозиторії. Це значно полегшує використання гнучких методологій розробки за допомогою етапів автоматизації етапів безперервного постачання. Контейнери також підтримують концепцію, в якій одного разу зібраний контейнер може виконуватися в різних середовищах. Оскільки контейнери абстраговані від інфраструктури, адміністратори можуть з легкістю управляти їх розгортанням.

Контейнери підтримують версіонування. Це полегшує збірку артефактів, оскільки вони за своєю природою повинні версіонуватися.

Образи контейнерів можна перевикористати. Якщо образ будується для вирішення конкретної проблеми, його можна буде завжди використовувати в подібних ситуаціях.

Існує так звана концепція незмінних контейнерів. У цій концепції контейнери створюються і видаляються після рішення свого завдання. Вони ніколи не оновлюються або виправляються. Незмінні контейнери використовуються в різних середовищах уникаючи ускладнення при виправленні компонентів розгортання. Виправлення тягнуть за собою

відсутність відстеження та відсутності можливості створити оточення консистентно [7].

Незважаючи на свою привабливість, технологія контейнерної віртуалізації має кілька недоліків.

Безпека - найсерйозніша проблема, яка, однак, часто не береться до уваги. Деніел Уолш, інженер з безпеки з Red Hat, опублікував статтю «Порожні контейнери». Там розглядається Docker, який використовує libcontainers в якості основи. Libcontainers взаємодіє з п'ятьма просторами - Process, Network, Mount, Hostname, Shared Memory - при роботі з Linux. При цьому є безліч важливих підсистем ядра Linux поза контейнера.

До них відносяться всі пристрої, SELinux, Cgroups і вся файлова система всередині /sys. Це означає, що при наявності у користувача або додатки прав суперкористувача в контейнері операційна система може бути зламана. І це погано.

Є багато способів убезпечити Docker і інші технології контейнеризації. Наприклад, можна змонтувати /sys тільки для читання, змусити процеси контейнерів працювати з певними розділами файлової системи і налаштувати мережу так, щоб можна було підключатися лише до певних внутрішнім сегментам. Але нічого з цього за замовчуванням не зроблено, і вам доведеться додатково потурбуватися цими питаннями.

В якості базового правила можна порекомендувати ставитися до контейнерів так само, як ви ставитеся до будь-якого серверного додатку. Ось що радить Уолш:

- Видаліть привілеї якомога швидше.
- Запускайте служби без прав root всюди, де це можливо.
- Ставтеся до прав root в контейнері так, як якщо б вони діяли і поза ним.

Інша проблема полягає в тому, що контейнеризовано додатки випускає хто завгодно. І серед розробників завжди знайдуться «погані хлопці». Якщо, наприклад, ви або ваші співробітники досить ліниві, щоб запустити на сервері

перший-ліпший контейнер, то цілком можете отримати трояна. Важливо донести до співробітників думка, що не можна просто завантажити будь-які додатки з Інтернету так само, як вони це роблять на своїх смартфонах.

Взагалі, на смартфони теж не варто ставити що завгодно, але це вже зовсім інша історія.

Інші проблеми контейнерів

Роб Хиршфельд, генеральний директор RackN, також вказує на серйозність проблеми розростання числа контейнерів. Варто пам'ятати про те, що «Поділ системи на безліч більш дрібних окремих частин - хороша ідея. Але це означає і те, що управляти доведеться ще більшим числом елементів. Завжди існує якась переломна точка між декомпозицією і неконтрольованим розростанням ».

Варто пам'ятати: вся суть контейнера полягає в запуску одного ізольованого додатку. Чим більше завдань ви призначаєте на контейнер, тим вище ймовірність того, що їх варто було вирішувати за допомогою віртуальних машин.

Вірно, деякі технології контейнеризації, наприклад, Linux Containers (LXC) можуть використовуватися замість VM. Наприклад, LXC можна використовувати для запуску певних програм Red Hat Enterprise Linux (RHEL) 6 на екземплярі RHEL 7. Але якщо говорити в загальному, то для запуску однієї програми підходять контейнери, а для кількох краще використовувати віртуальні машини [15].

РОЗДІЛ 4
РЕАЛІЗАЦІЯ ХМАРНИХ СЕРВІСІВ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ АПАРАТНОЇ ТА КОНТЕЙНЕРНОЇ ВІРТУАЛІЗАЦІЇ

4.1 Задачі і аспекти впровадження технологій віртуалізації при побудові хмарних сервісів

У зв’язку із значним поширенням технології хмарних обчислень зросла роль віртуалізації при побудові хмарних додатків, адже досить зручно реалізовувати певні компоненти обчислювальної мережі віртуально.

Рішення віртуалізації придатні для застосування у всіх моделях хмарних сервісів. Далі більш детально зупинимось на кожній з моделей.

В даний час прийнято виділяти три основні моделі обслуговування хмарних технологій, які іноді називають шарами хмари. Можна сказати, що ці три шари - послуги інфраструктури, послуги платформи і послуги додатків - відображають будову не тільки хмарних технологій, а й інформаційних технологій в цілому. Зупинимося докладніше на кожному з них [16].

До послуг інфраструктури (Infrastructure as a Service - IaaS) можна віднести набір фізичних ресурсів, таких як сервери, мережеве обладнання та накопичувачі, пропоновані замовникам в якості наданих послуг. Послуги інфраструктури вирішують задачу належного оснащення ЦОД, надаючи обчислювальні потужності в міру необхідності. Зазвичай ці послуги підтримують інфраструктуру і набагато більше число споживачів в порівнянні з послугами додатків. Окремим прикладом послуг інфраструктури є апаратне забезпечення як послуга (Hardware as a Service - HaaS). Як послуги користувач отримує обладнання, на основі якого розгортає свою власну інфраструктуру з використанням найбільш підходящого ПЗ.

Споживач при цьому не керує базовою інфраструктурою хмари, але має контроль над операційними системами, системами зберігання, розгорнутими додатками і, можливо, обмежений контроль вибору мережевих компонентів (наприклад, хост з мережевими екранами). В такому випадку захист платформ

і додатків забезпечує сам споживач, а провайдер хмари повинен організувати захист інфраструктури. Для надання ресурсів на вимогу часто використовується віртуалізація.

Послуги платформи (Platform as a Service - PaaS) - це модель обслуговування, в якій споживачеві надаються додатки (створені або придбані) як набір послуг. У нього входять, зокрема, проміжне ПЗ як послуга, обмін повідомленнями як послуга, інтеграція як послуга, інформація як послуга, зв'язок як послуга і т.д. Наприклад, робоче місце як послуга (Workplace as a Service - WaaS) дозволяє компанії використовувати хмарні обчислення для організації робочих місць своїх співробітників, налаштувавши і встановивши все необхідне для роботи персоналу ПЗ. Дані як послуга (Data as a Service - DaaS) надають користувачеві дисковий простір, який він може використовувати для зберігання великих обсягів інформації. Безпека як послуга (Security as a Service - SaaS) дає можливість користувачам швидко розгортати продукти, що дозволяють забезпечити безпечне використання веб-технологій, безпеку електронного листування, а також безпеку локальної системи. Цей сервіс дозволяє користувачам економити на розгортанні та підтримці своєї власної системи безпеки.

Прикладами послуг платформи служать IBM SmartCloud Application Services, Amazon Web Services, Windows Azure, Boomi, Cast Iron, Google App Engine і інші.

Послуги додатків (Software as a Service - SaaS) припускають доступ до додатків як до сервісу, тобто додатки провайдера запускаються в хмарі і надаються користувачам на вимогу, як послуги. Іншими словами, користувач може отримувати доступ до програмного забезпечення, розгорнутого на віддалених серверах, за допомогою Інтернету, причому всі питання оновлення та ліцензій на ці програми регулюються постачальником даної послуги. Оплата в даному випадку здійснюється за фактичне використання ПЗ. Іноді ці послуги постачальники роблять безкоштовними, так як у них є можливість отримувати дохід, наприклад, від реклами.

Програма є доступною за допомогою різних клієнтських пристроїв або через інтерфейси тонких клієнтів, такі, наприклад, як веб-браузер, або веб-пошта, або інтерфейси програм. Споживач при цьому не керує базовою інфраструктурою хмари, в тому числі мережами, серверами, операційними системами. В кінці кінців, ви несете відповідальність тільки за збереження параметрів доступу (логінів, паролів і т.д.) і виконання рекомендацій провайдера з безпечних налаштувань додатків.

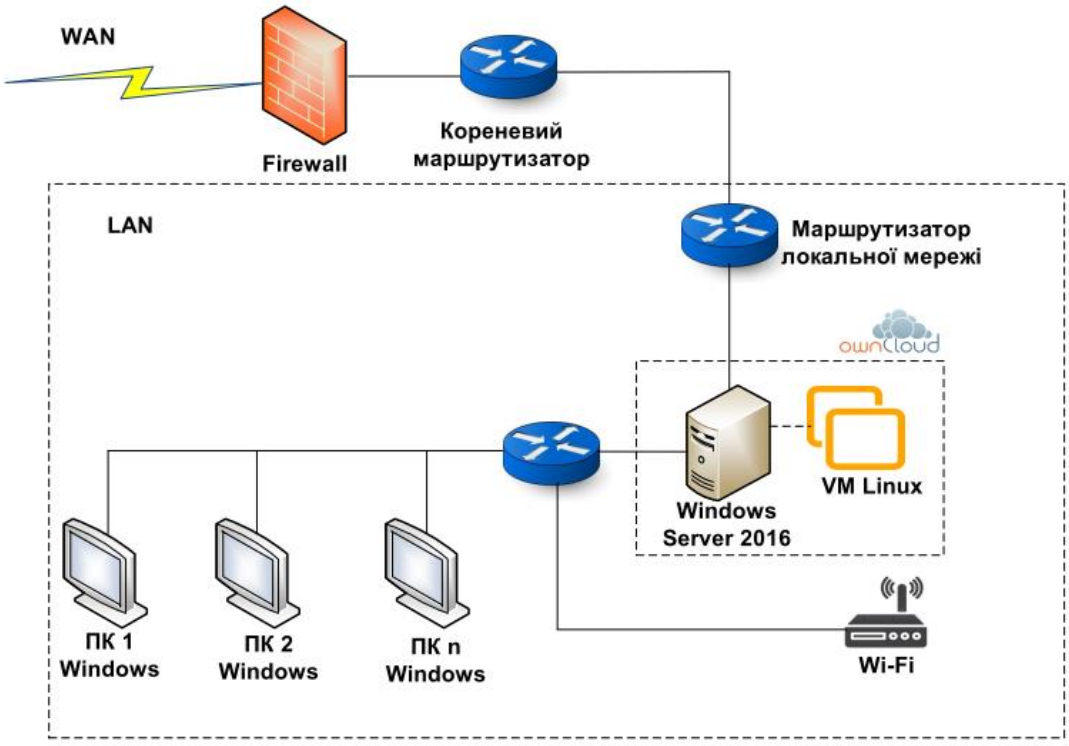


Рис. 4.1 Корпоративна інформаційна мережа із хмарних сервісом, побудованим на віртуальній машині

На Рис. 4.1 зображено схему реальної мережі одного з підприємств, на якому реалізовано хмарний сервіс. Цим сервісом є рішення для зберігання та обміну даними між користувачами корпоративної мережі ownCloud. На фізичному сервері з ОС Windows Server за допомогою вбудованого гіпервізора створено віртуальну машину із ОС Linux. Вибір Linux обгрунтований тим, що ownCloud реалізований як сервіс для роботи на Unix-системах. Кожен користувач в локальній мережі VLAN має свій обліковий запис в хмарі ownCloud. На кожному ПК є свій каталог синхронізації, в який користувач

добавляє ті файли, які необхідно поширити серед колег. Відповідно, після синхронізації ті користувачі, які мають право на перегляд чи редагування/видалення файлів в цьому каталозі, можуть користуватись ними, отримавши свою локальну копію.

Керування роботою хмарного додатку, створення облікових записів, паролів, надання прав доступу виконується адміністратором.

Отже, як бачимо, ідея використання методів віртуалізації при побудові хмарного сервісу (в даному випадку відтуальних машин) є цілком придатною для реалізації на практиці.

4.2 Побудова макетів хмарного сервісу із використанням методів апаратної та контейнерної віртуалізації

Для демонстрації можливості використання віртуальних машин та віртуальних контейнерів при побудові хмарних сервісів було вирішено реалізувати практично макет певного хмарного сервісу по черзі в середовищі апаратної та контейнерної віртуалізації.

Макет хмарного сервісу в обох віртуалізованих середовищах буде побудовано на ПК Lenovo з такими характеристиками:

- Процесор Intel Core-i3 з тактовою частотою 2.2 GHz
- Обсяг оперативної пам’яті 8 Гб
- Відеоадаптер Nvidia GeForce 610M
- Хостова ОС Windows 10 Pro для робочих станцій

В якості досліджуваного хмарного сервісу було вибрано ownCloud - систему для організації зберігання, синхронізації й обміну даними, розміщеними на зовнішніх серверах. Від схожих сервісів, таких як Google Docs, Dropbox, box.net і Ubuntu One система ownCloud відрізняється наданням користувачеві повного контролю над своїми даними — інформація не прив'язується до зовнішніх закритих хмарних систем зберігання, а розміщується на підконтрольних користувачеві системах. Дане рішення дуже корисне при розробці власних хмарних файлообмінників приватними

підприємствами. Зручність полягає в тому, що система зберігання і обміну даними є повністю закритою, що значно ускладнює несанкціонований доступ до файлів.

Для доступу до даних, збережених в ownCloud, можна використовувати веб-інтерфейс або протокол WebDAV. Додатково до зберігання даних можна відзначити функції підтримки засобів для забезпечення спільного доступу і можливість синхронізації між різними машинами таких даних, як адресна книга, календар-планувальник і закладки, з можливістю їхнього перегляду і редагування з будь-якого пристрою в будь-якій точці мережі.

В якості ПЗ для створення віртуальної машини було обрано програмний продукт Hyper-V, про який вже згадувалося в розділі 2.3.

Макет на основі віртуального контейнера буде реалізований із використанням продукту Docker, про який також вже згадувалося раніше.

Обидва програмних продукти є типовими представниками своїх типів віртуалізації і добре підходять для реалізації поставленої задачі.

4.2.1 Створення та налаштування сервісу ownCloud на базі віртуальної машини Hyper-V

Реалізацію хмарного сервісу ownCloud виконаємо у віртуальній машині Hyper-V – стандартного гіпервізора від Microsoft, присутнього в ОС Windows 10 Pro та Windows 10 Enterprise. Для створення віртуальної машини слід увімкнути службу Hyper-V.

Після цього, знайшовши в меню «Пуск» диспетчер Hyper-V, виконаємо налаштування віртуального комутатора, який буде необхідний для підключення віртуальної машини до інтернету, що потрібно для завантаження ownCloud з репозиторію Ubuntu. Вибір ОС Ubuntu 16.04 мотивований тим, що сервіс ownCloud розрахований на роботу в Unix-подібних ОС, типовим представником яких є ОС Ubuntu 16.04.

Не будемо детально описувати процес створення віртуальної машини в Hyper-V. Відмітимо лише характеристики, які матиме створена віртуальна машина:

- 4 Гб RAM
- 1 віртуальний процесор
- 10 Гб об’єм віртуального жорсткого диска
- ОС Ubuntu 16.04 x64

Під час налаштування слід вказати, що ОС буде інстальована при створенні віртуальної машини і вибрати образ ОС на жорсткому диску ПК.

Після того, як віртуальна машина створена, в меню вибираємо Пуск і очікуємо запуску ОС Ubuntu.

Після того, як ОС почала роботу, знаходимо Термінал і далі встановлення ownCloud будемо виконувати в інтерфейсі командного рядка. Увівши команду apt-get update, виконаємо оновлення ОС, після чого безпосередньо перейдемо до встановлення хмарного сервісу (Рис. 4.2). Слід зазначити, що ця і наступні команди повинні виконуватися під правами суперкористувача root.

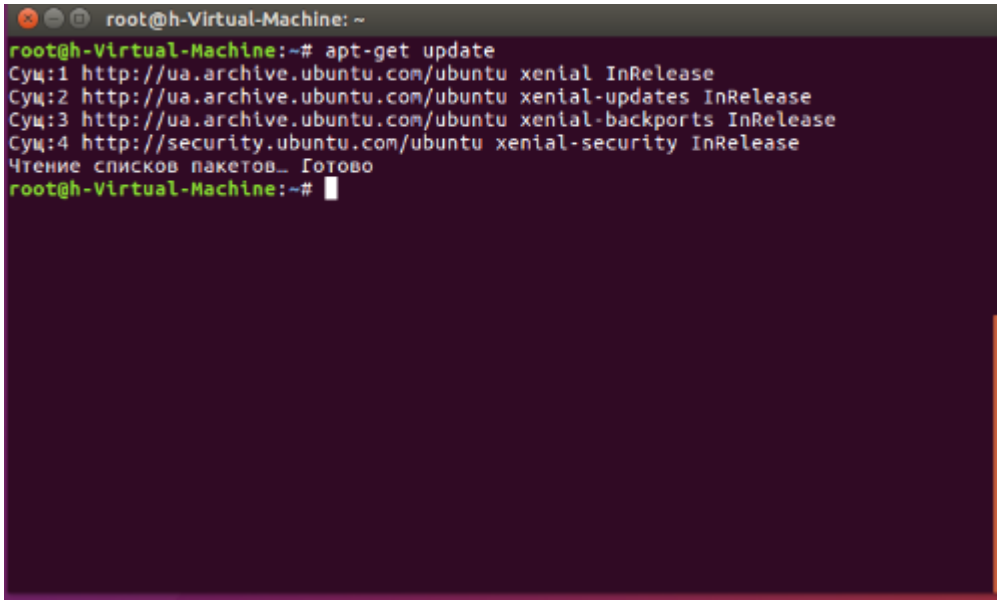
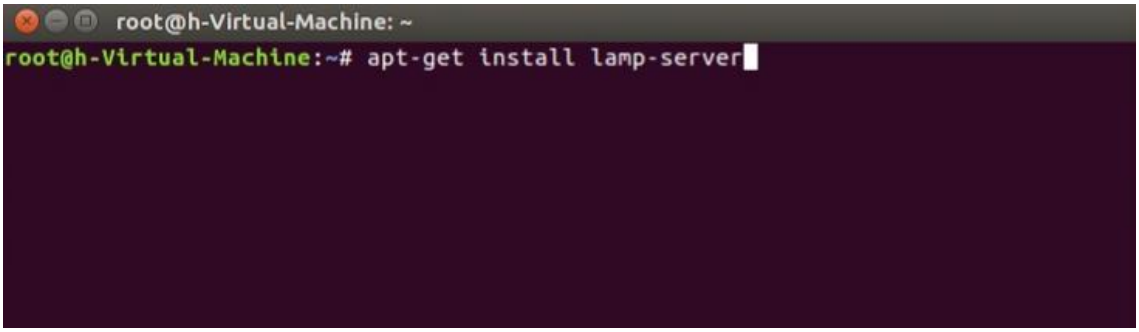


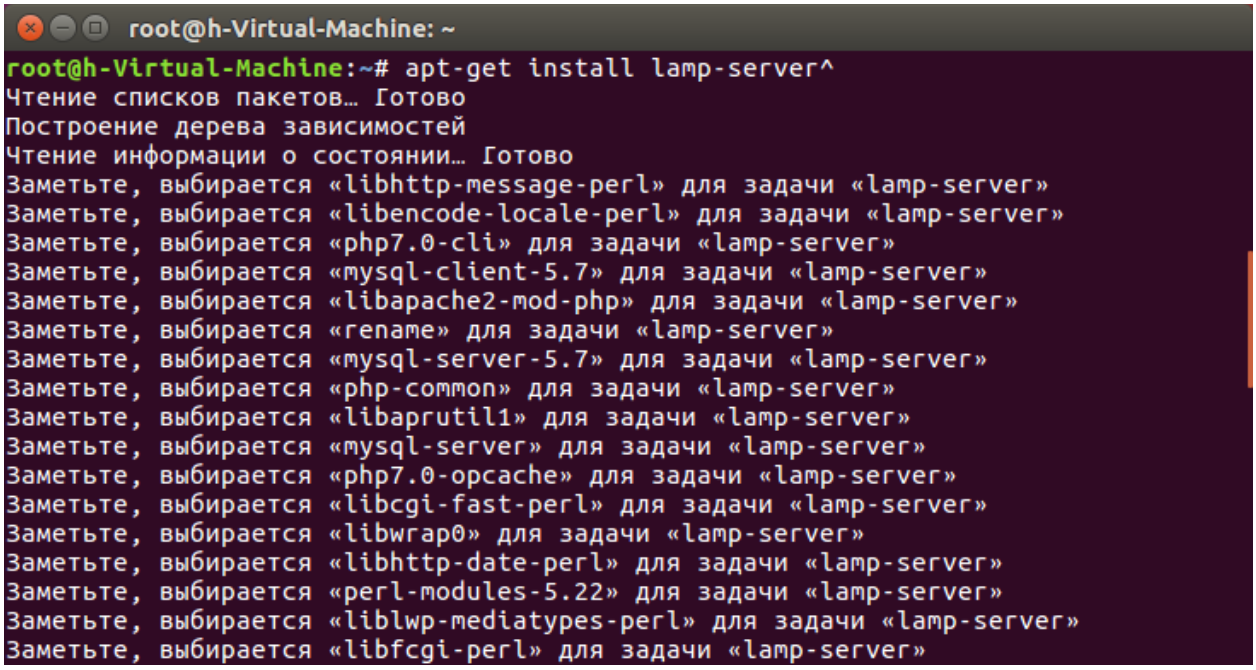
Рис. 4.2 Оновлення ОС Ubuntu

OwnCloud написаний на php, тому перед тим, як встановлювати саму програму нам, у першу чергу, необхідно встановити LAMP сервер. Цей набір

складається з веб-сервер Apache, сервер баз даних MySQL і мова програмування PHP.



а)



б)

Рис. 4.3 Завантаження сервера LAMP (а), б))

Після успішного завершення налаштування lamp сервера у нашій системі, завантажимо останню версію OwnCloud з офіційного сайту.

Ми не будемо використовувати репозиторії Ubuntu, оскільки нам потрібна установка OwnCloud 9 ubuntu 16.04, а в офіційних репозиторіях може бути вже застаріла версія. Для завантаження потрібні такі команди (Рис. 4.4):

Рис. 4.4 Завантаження ownCloud

Тепер винесемо все викачані файли в кореневу папку веб-сервера і налаштуємо потрібні дозволи на наші файли і каталоги:

cd /var/www/html

$ sudo tar xjf /opt/owncloud-9.0.1.tar.bz2

$ sudo chown -R www-data:www-data owncloud

$ sudo chmod -R 755 owncloud

Після того, як код буде розпакований потрібно створити базу даних MySQL і обліковий запис користувача для налаштування OwnCloud (Рис. 4.5). Для цього виконаємо такі команди:

$ mysql -u root -p

Enter password:

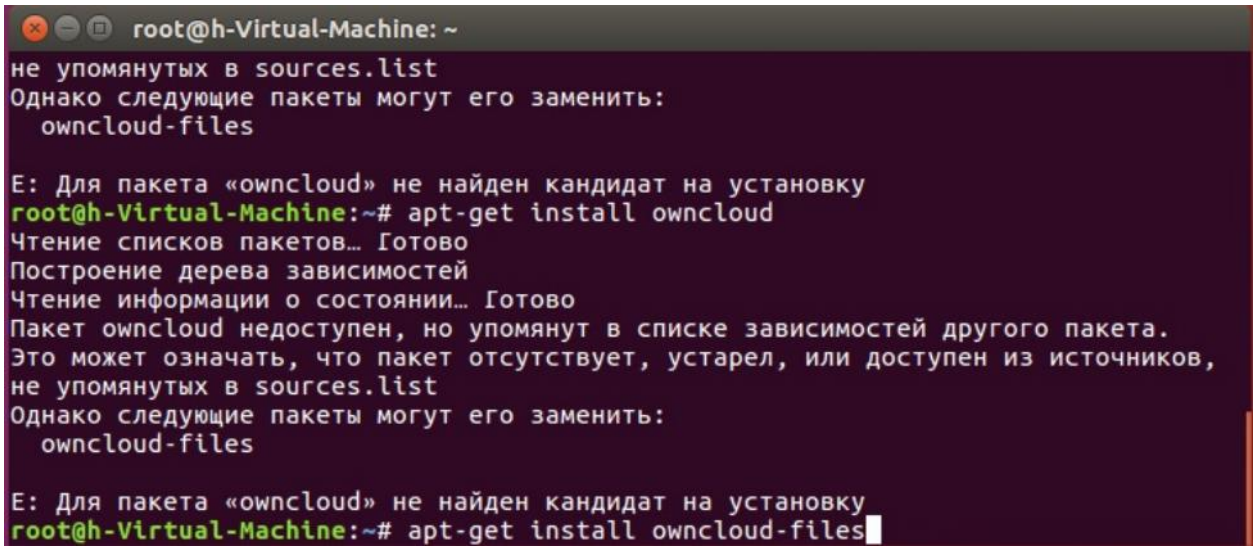


Рис. 4.5 Інсталювання OwnCloud

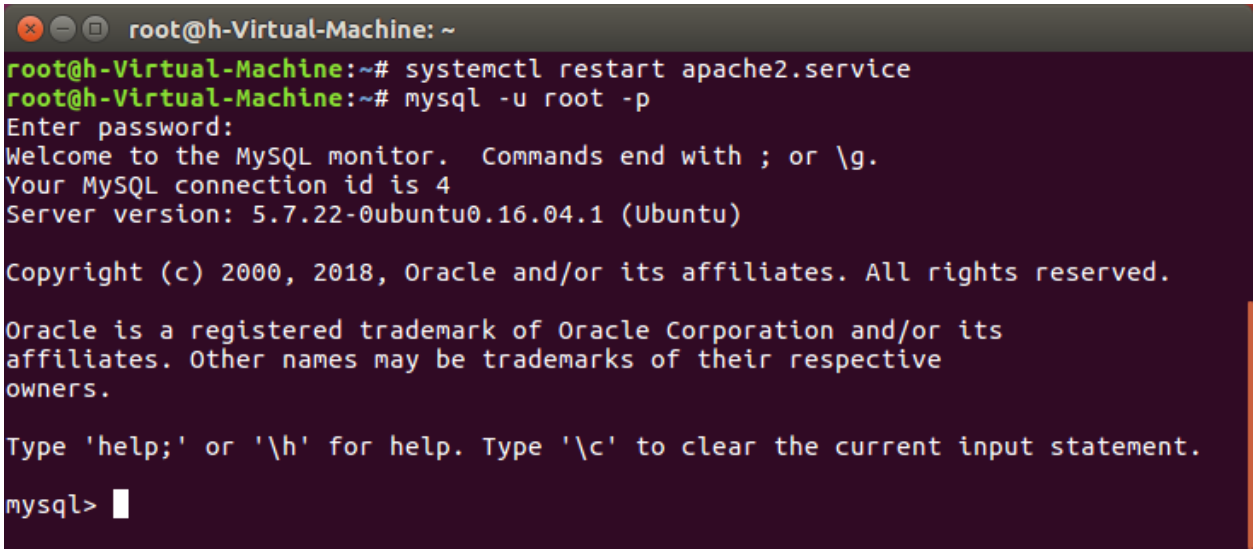


Рис. 4.6 Створення бази даних MySQL

Для своєї роботи OwnCloud вимагає ще кілька пакетів, це бібліотека gd і curl, їх теж потрібно встановити:

$ sudo apt install php-gd php-curl

Тепер ми можемо отримати доступ до OwnCloud з веб-браузера. Просто використовуємо адресу localhost: http://localhost/owncloud /

Введемо дані адміністратора, щоб створити обліковий запис адміністратора і вкажемо розташування папки з файлами (Рис. 4.7):

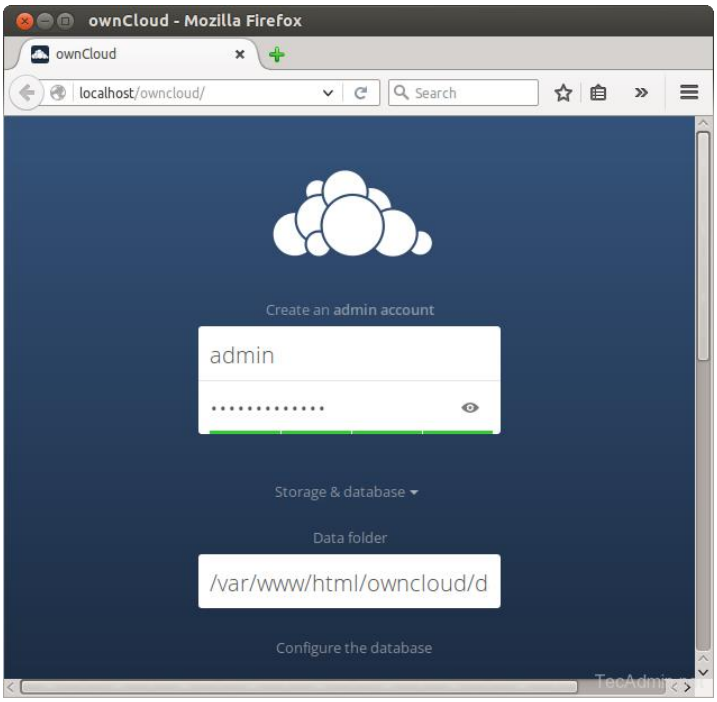


Рис. 4.7 Вікно авторизації ownCloud

Тепер погортаємо сторінки вниз і введемо дані для підключення до бази даних, яку ми раніше налаштували:

Після завершення налаштування відкриється панель адміністратора, де можна створювати користувачів, групи і призначати їм права доступу (Рис. 4.8):

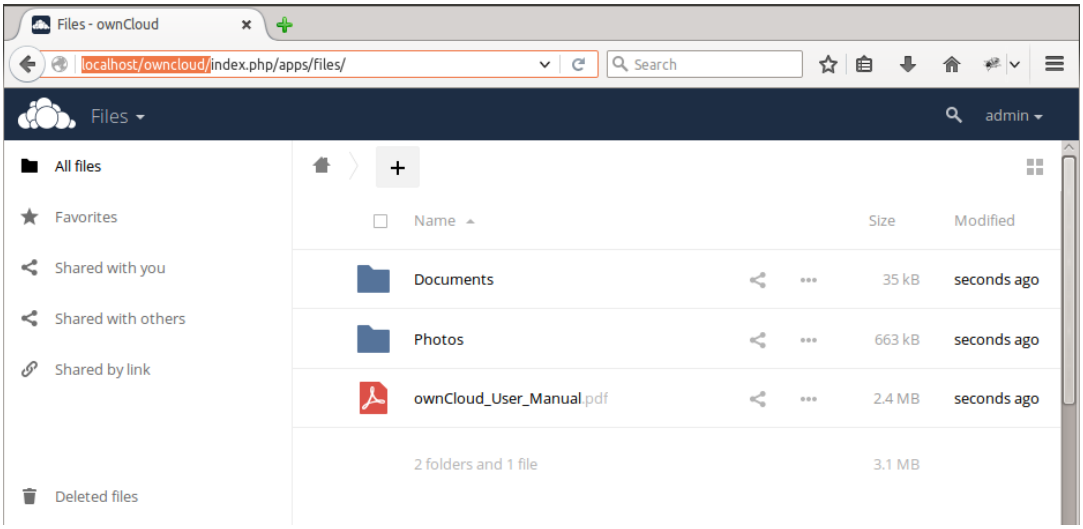


Рис. 4.8 Каталог адміністратора

Тепер можна перевірити роботу сервісу ownCloud. Для цього в корінь каталогу адміністратора завантажимо кілька файлів. Протягом кількох секунд

файли синхронізувались і стали доступними для перегляду та редагування як адміністратору, так і іншим користувачам, яким надано доступ до каталогу (Рис. 4.9).

Отже, хмарний сервіс ownCloud, побудований у віртуальному середовищі Hyper-V виявився працездатним. Створення макету пройшло успішно.

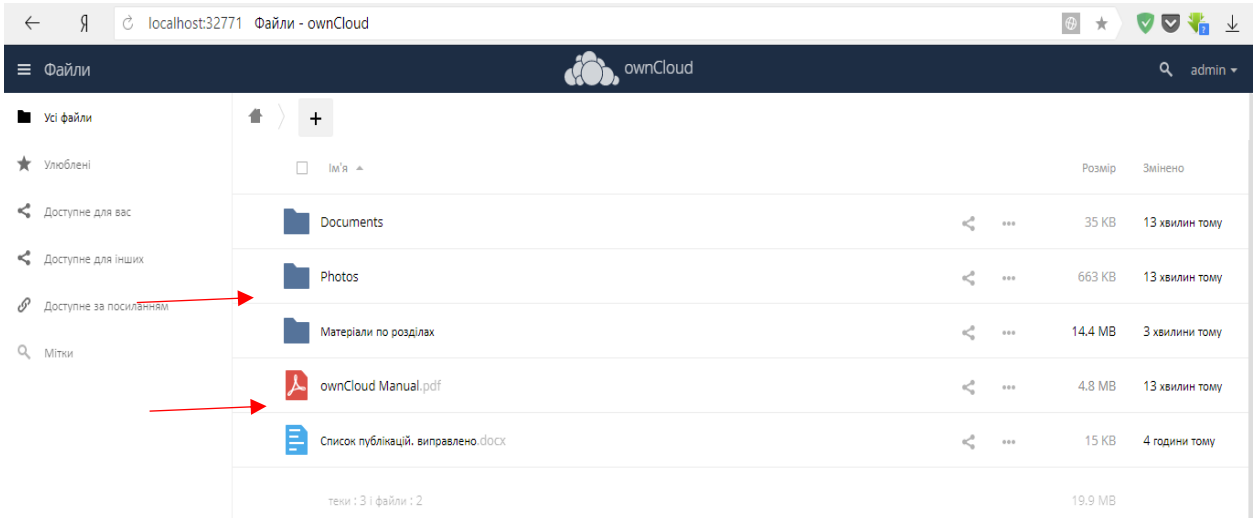


Рис. 4.8 Перевірка працездатності файлообмінника

4.2.2 Створення та налаштування сервісу ownCloud на базі віртуального контейнера Docker

Хмарний сервіс для зберігання та обміну данними ownCloud може бути реалізований не тільки на базі віртуальної машини, а й на базі віртуального контейнера. Раніше в розділі 3.2 було розглянуто архітектуру Docker та дано загальну характеристику цій системі віртуалізації на рівні ОС. Тому наразі зосередимось лише на створенні і налаштуванні сервісу ownCloud у віртуальному контейнері Docker.

OwnCloud може бути встановлений в контейнері Docker, використовуючи офіційний образ ownCloud для Docker.

Для інсталювання контейнера Docker, в першу чергу необхідно завантажити файл інсталяції Docker з сайта https://download.docker.com. Після інсталяції слід відкрити з правами адміністратора ярлик і запустити програму. На Рис. 4.9 зображене стартове вікно Docker.

Далі необхідно авторизуватися, для чого створюється обліковий запис користувача на сайті https://www.docker.com/ (Рис. 4.10).

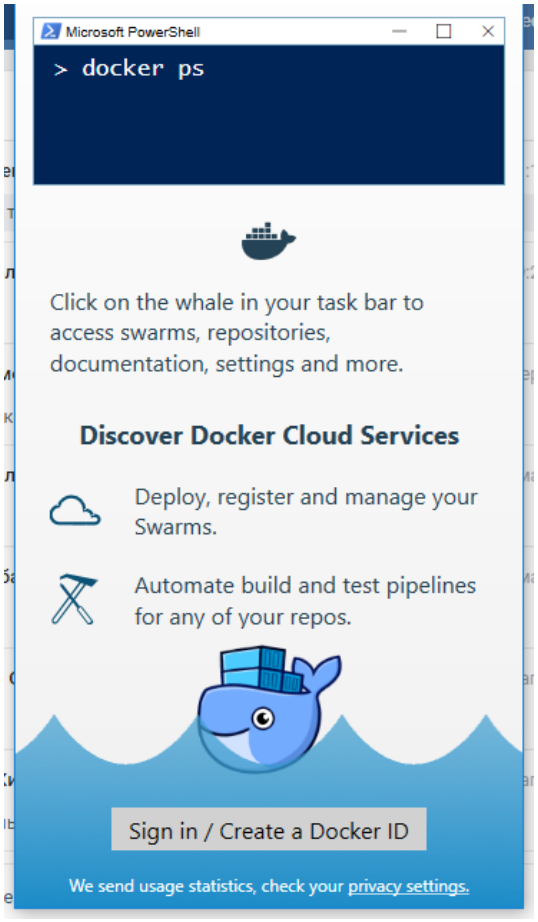


Рис. 4.9 Стартове вікно Docker

Рис. 4.10 Авторизація в Docker

Далі запускається програма Docker. Контекстним меню запускається меню, в якому треба вибрати “Switch to Linux containers”, оскільки Docker

розрахований на роботу в середовищі Linux (Рис. 4.11). В цьому випадку емулюється ядро ОС Linux.

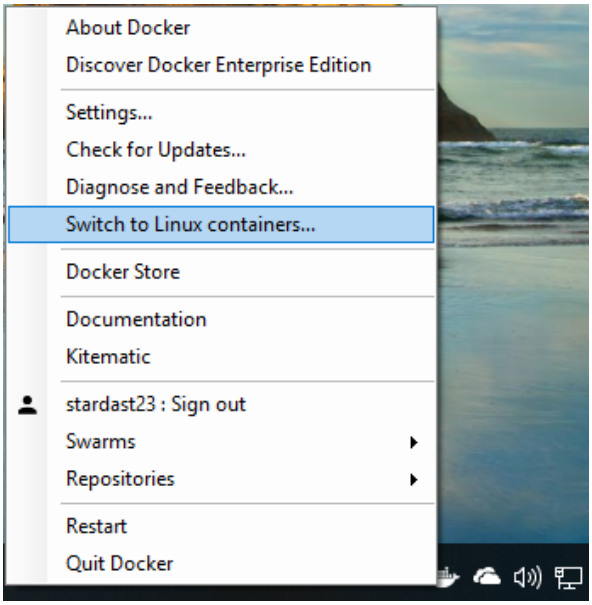


Рис. 4.11 Переключення на середовище Linux

Налаштування і робота в Docker можливі як використовуючи командний рядок або PowerShell, так і з використанням графічного інтерфейсу Kinematic. Ми будемо використовувати графічний інтерфейс. Для його встановлення в контекстному меню слід вибрати Kinematic, після завантаження встановити програму і запустити ярлик (Рис. 4.12).

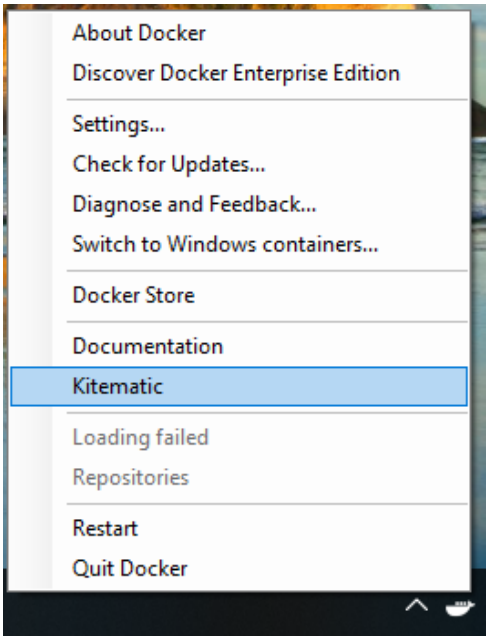


Рис. 4.12 Завантаження графічного інтерфейсу Kinematic

Зайшовши у головне меню (Рис. 4.13), потрапляємо в офіційний репозиторій Docker. В лівому верхньому кутку бачимо список встановлених контейнерів. Через пошук знаходимо контейнер ownCloud і встановлюємо (Рис. 4.14).

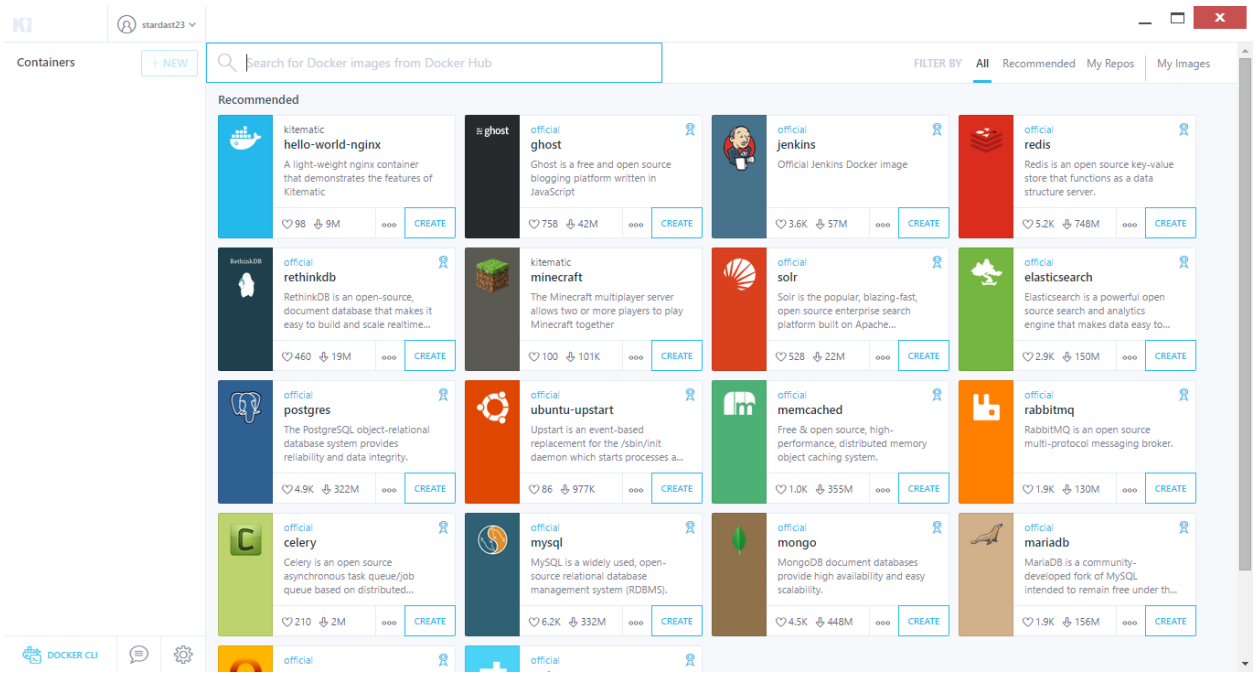


Рис. 4.13 Стартове вікно графічного інтерфейсу Kinematic

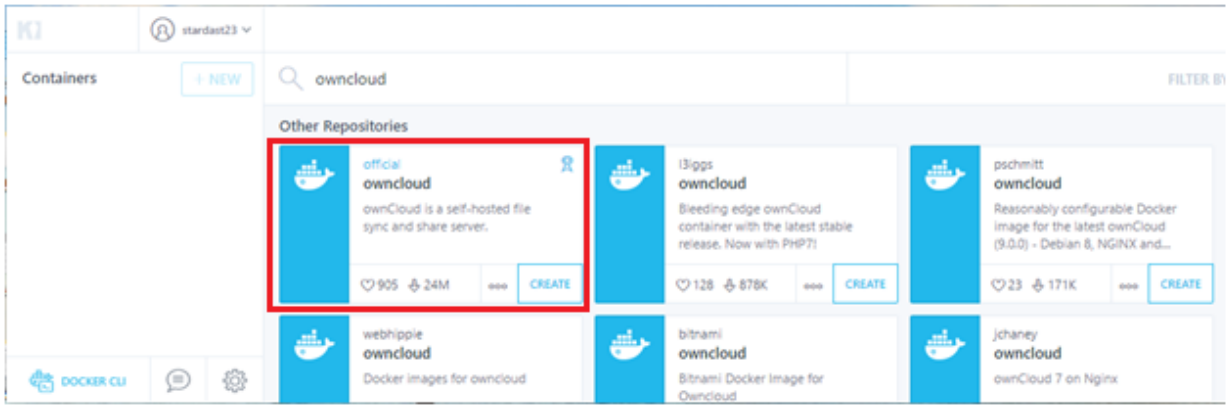


Рис. 4.14 Контейнер ownCloud

Образ ownCloud вже повністю придатний до роботи і не потребує встановлення ніяких інших компонентів. Після запуску контейнера протягом кількох секунд сервіс запускається і у вікні справа ми можемо бачити зображення стартового вікна хмарного сервісу

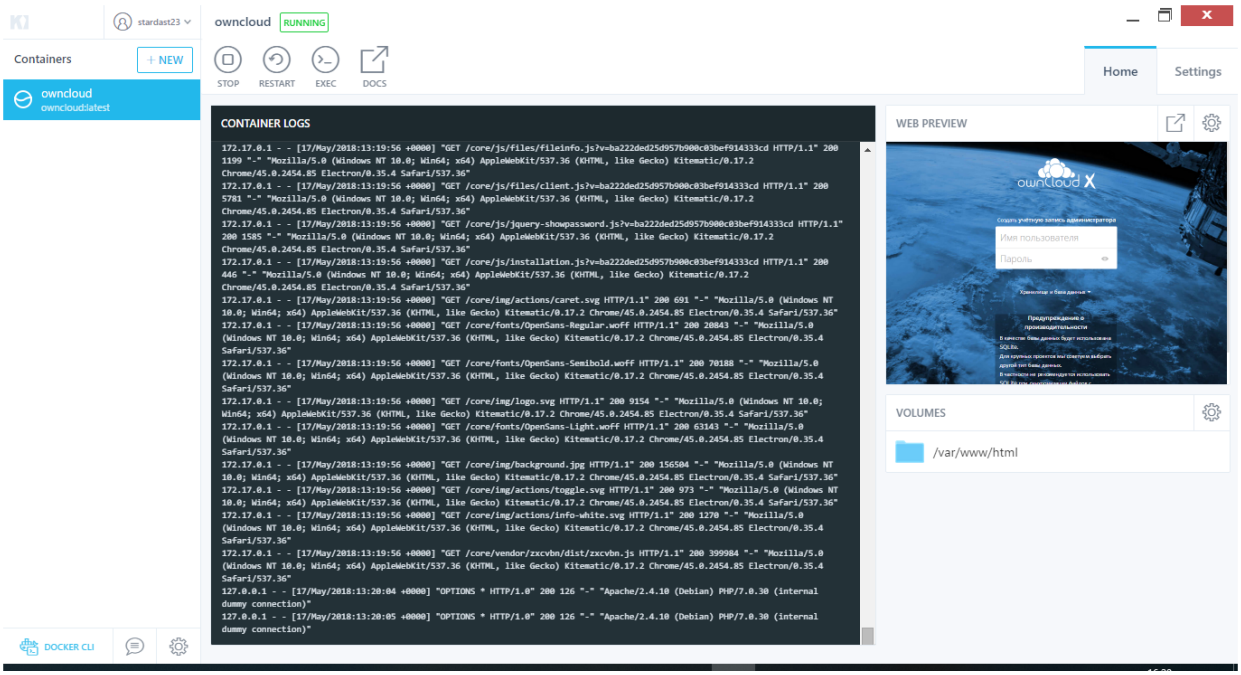


Рис. 4.15 Запус контейнера ownCloud

В веб-браузері відкривається стартове вікно сервісу ownCloud, де можна відразу створити обліковий запис адміністратора (Рис. 4.16).

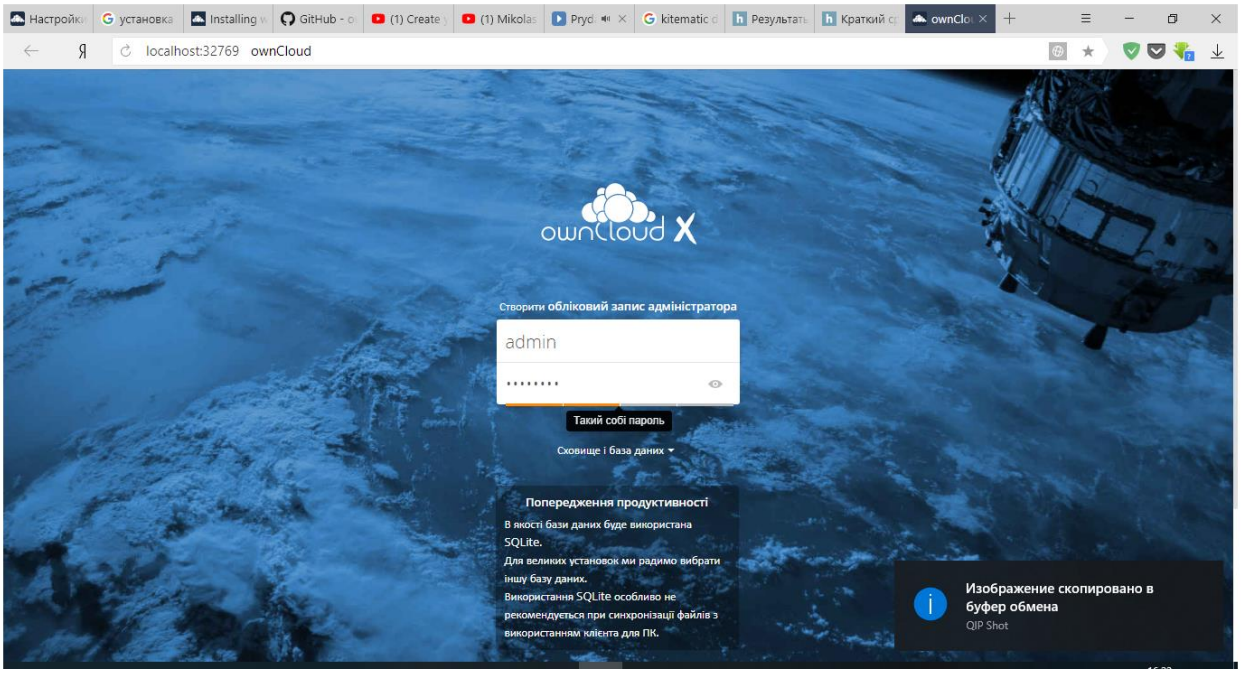


Рис. 4.16 Вікно авторизації ownCloud

Після авторизації потрапляємо у кореневий каталог, де бачимо список файлів, які наявні у даного користувача (Рис. 4.17).

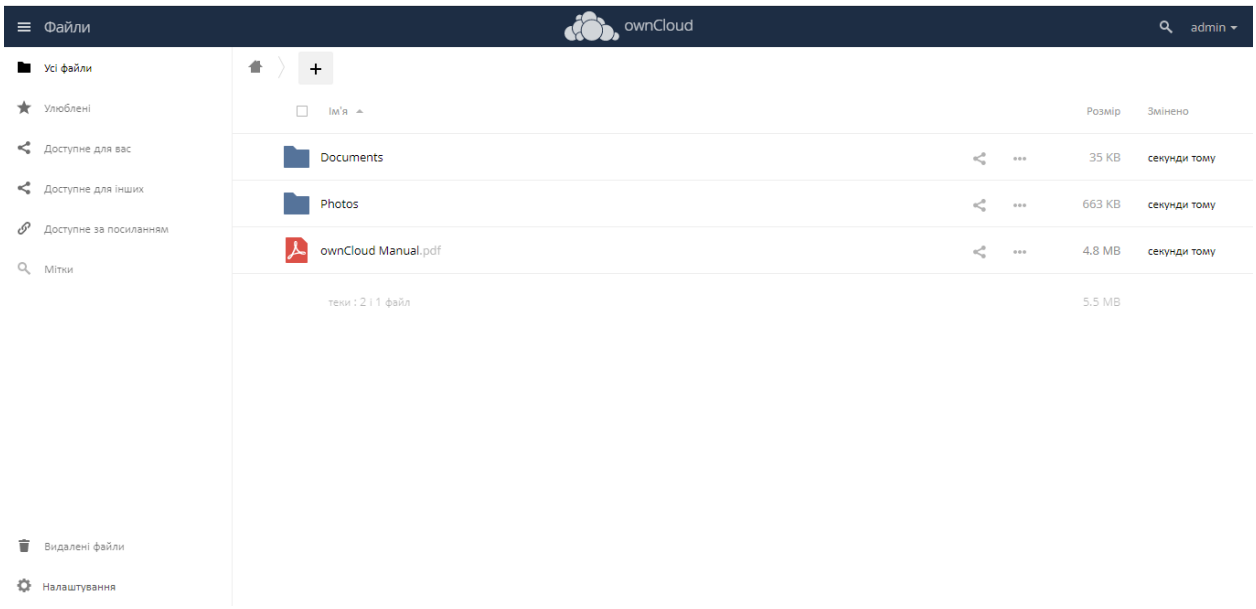


Рис. 4.18 Каталог користувача із списком файлів

Вибравши у правому верхньому списку «Користувачі», потряпляємо у меню, де можна створювати, редагувати, видаляти облікові записи та надавати права доступу до файлів інших користувачів (Рис. 4.19)

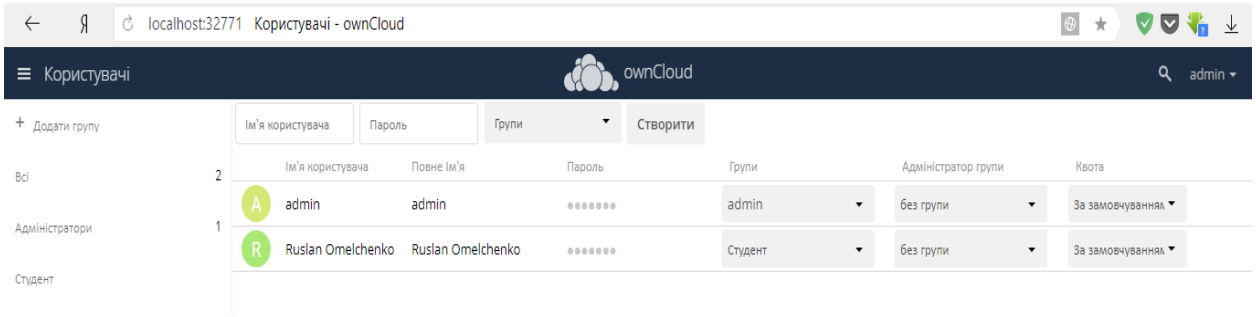


Рис. 4.19 Створення нового користувача із присвоєнням прав на доступ і адміністрування

На Рис. 4.20 зображено принцип роботи сервісу ownCloud. Нові файли завантажуються у хмару, вони відразу синхронізуються і стають доступними в каталогах користувачів, яким надано доступ на перегляд даних файлів.

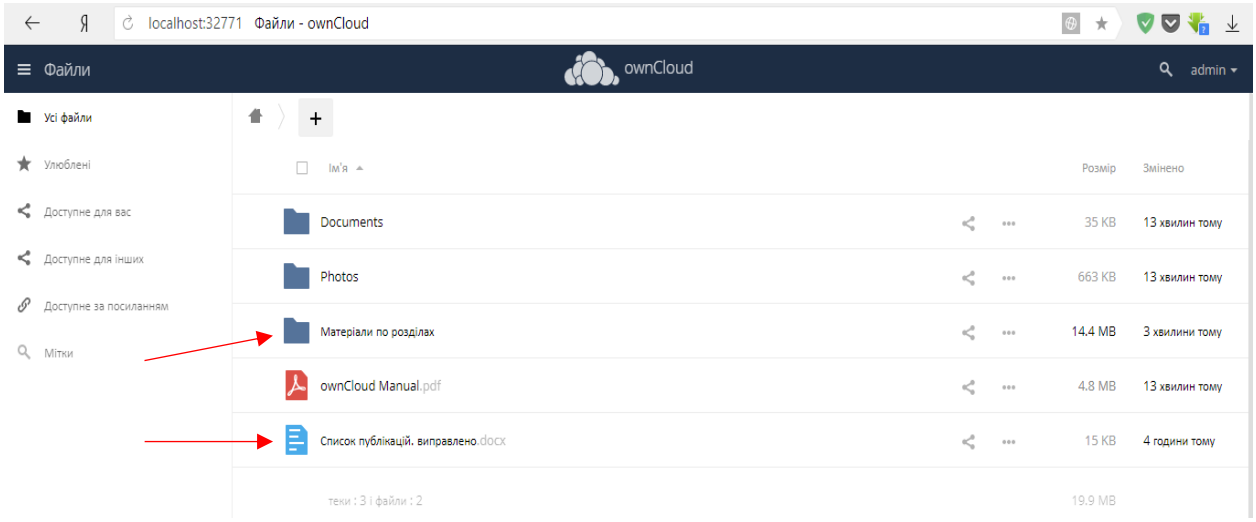


Рис. 4.20 Завантаження файлів у каталог користувача

Переконавшись у працездатності сервісу, зупиняємо його роботу, вибравши у контекстному меню програми Quit Docker. Віртуальний контейнер згортається і припиняє роботу лише за декілька секунд.

Таким чином, макет хмарного сервісу ownCloud, реалізованого на базі контейнера в середовищі Docker, є працюючим. Файли завантажуються, синхронізуються і копіюються до каталогів інших користувачів з правами на перегляд. Сховищем даних виступає ПК, на якому реалізовано систему Docker.

ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ

1. Розглянуто поняття віртуалізації, основні властивості технології. Розглянуто класифікацію типів віртуалізації та показано структурні особливості кожного типу.
2. Досліджено основні аспекти віртуалізації мереж та комутації у віртуалізованих середовищах. Проаналізовано основні переваги використання віртуалізації, а також її недоліки. Показано можливість широкого використання віртуалізованих вузлів в інфраструктурі SDN та NFV.
3. Проаналізовано архітектурні особливості апаратної віртуалізації. Розглянуто типи гіпервізорів, їх будову (на прикладі гіпервізора віртуальної машини Hyper – V) та проведено огляд основних розробників рішень у сфері апаратної віртуалізації.
4. Досліджено архітектуру віртуальних контейнерів і на прикладі віртульних контейнерів Docker показано їхні властивості. Проведено огляяд основних рішень у сфері контейнерної віртуалізації. Також наведено переваги використання віртуальних контейнерів та їх недоліки і показано основні відмінності між апаратною та контейнерною віртуалізацією.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Илья Клементьев. Технологии виртуализации [Електронний ресурс] / Илья Клементьев, Владимир Устинов // Национальный Открытый Университет "ИНТУИТ". – 2015. – Режим доступу до ресурсу: www.intuit.ru.

2. М. Тим Джонс. Сетевые адаптеры, коммутаторы, сети и устройства / М. Тим Джонс // Виртуализация сетей в Linux / М. Тим Джонс., 2011.

3. Самойленко О. Сетевое взаимодействие в VMware Workstation и VMware Server [Електронний ресурс] / Олександр Самойленко. – 2007. – Режим доступу до ресурсу: http://www.vmgu.ru/articles/vmware-workstation-server-networking.

4. Александр Барсков. SDN: от концепции к решениям / Александр Барсков. // Журнал сетевых решений/LAN. – 2015. – №9.

5. Сергей Орлов. Многоконтроллерная инфраструктура SDN / Сергей Орлов. // Журнал сетевых решений/LAN. – 2014. – №12.

6. Сергей Орлов. SDN и другие / Сергей Орлов. // Журнал сетевых решений/LAN. – 2014. – №6.

7. Павлов В. В. Использование технологии контейнеризации для оптимизации микросервисов / В. В. Павлов, Ю. А. Шичкина. – Санкт-Петербург, 2017.

8. Бхану П. Толети. Гипервизоры, виртуализация и облако: О гипервизорах, виртуализации систем и о том, как это работает в облачной среде / Бхану П. Толети. – 2012.

9. Виртуализация в задачах оптимизации нагрузок на компьютерные системы [Електронний ресурс]. – 2011.

10. Джон Мак-Кейб (John McCabe) и команда Windows Server. Введение в Windows Server 2016 / Джон Мак-Кейб (John McCabe) и команда Windows Server. – Редмонд, штат Вашингтон: Microsoft Press, 2016. – 18-20 с.

11. Алексей Савельев. Hyper - V [Електронний ресурс] / Алексей Савельев // Национальный Открытый Университет "ИНТУИТ". – 2015. – Режим доступу до ресурсу: www.intuit.ru.

12. Алексей Савельев. Решения виртуализации [Електронний ресурс] / Алексей Савельев // Национальный Открытый Университет "ИНТУИТ". – 2015. – Режим доступу до ресурсу: www.intuit.ru.

13. Яцкин А. Д. Динамические honeypot-системы на основе контейнерной виртуализации / А. Д. Яцкин. – 2016.

14. Омельченко Р. Аналіз архітектурних особливостей віртуалізації та переваги віртуальних контейнерів / Л. Верес, Р. Омельченко. // Збірник тез дванадцятої міжнародної науково-технічної конференції "ПРОБЛЕМИ ТЕЛЕКОМУНІКАЦІЙ". – 2018. – С. 116–118

15. V.Sinitskiy. Контейнеры или виртуальные машины — что лучше выбрать для своей компании? / V.Sinitskiy. – 2015.

16. Т.В. Батура, к.ф.-м.н. Облачные технологии: основные понятия, задачи и тенденции развития / Т.В. Батура, к.ф.-м.н., Ф.А. Мурзин, к.ф.-м.н., Д.Ф. Семич, к.ф.-м.н.. – 2014. – №1. – С. 1–2.