Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра *Інформаційно-мережної інженерії*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Комплексна програмна платформа для розгортання та надання хмарних сервісів

(тема)

Виконала:
здобувачка *4* року навчання,
групи *ТРІМІз-22-1*

Мілена БОЧАРОВА
(власне ім'я, прізвище)

Спеціальність *172 Телекомунікації та радіотехніка*
(код і повна назва спеціальності)

Тип програми *освітньо-професійна*

Освітня програма *«Інформаційно-мережна інженерія»*
(повна назва освітньої програми)

Керівник доцент *Юрій КОЛТУН*
(посада, власне ім'я, прізвище)

Допускається до захисту

В.о. зав. кафедри _____ *Дарія ЧЕБОТАРЬОВА*
(підпис) (власне ім'я, прізвище)

2026 р.

Не містить відомостей заборонених до відкритого публікування.

Здобувач / *Мілена БОЧАРОВА* /

Керівник / *Юрій КОЛТУН* /

Харківський національний університет радіоелектроніки

Навчально-науковий центр заочної форми навчання

Кафедра *Інформаційно-мережної інженерії*
(повна назва)

Рівень вищої освіти *перший (бакалаврський)*

Спеціальність *172 Телекомунікації та радіотехніка*
(код і повна назва)

Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)

Освітня програма *«Інформаційно-мережна інженерія»*
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«*18*» *травня* 2026 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачці *Бочаровій Мілені Сергіївні*
(прізвище, ім'я, по батькові)

1. Тема роботи *Комплексна програмна платформа для розгортання та надання хмарних сервісів*

затверджена наказом університету від «*18*» *травня* 2026 р. № 73 *Стз*

2. Термін подання студентом роботи до екзаменаційної комісії *14 липня 2026 р.*

3. Вихідні дані до роботи *Тип розгортання хмарної інфраструктури – гібридна хмара. Моделі обслуговування – Iaas, PaaS. Технологічний підхід до розробки архітектури – мікросервісний підхід. Базова програмна платформа розробки – відкритий комплекс ПЗ OpenStack. Важливо забезпечити підтримку широкого спектру платформ віртуалізації.*

Розглянути загальні особливості ПЗ OpenStack. Обґрунтувати підходи та визначити сервісні компоненти щодо створення комплексної програмної платформи та на основі цього розробити її варіант архітектурної реалізації. Проаналізувати сервісні модулі розробленої архітектури в аспекті програмної реалізації у форматі JSON.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ

1. Фундаментальна концепція, загальні принципи, типи і моделі організації хмарних обчислень.

2. Аналіз та порівняння комплексних рішень програмних продуктів для розгортання і надання хмарних сервісів.

3. Розробка архітектури комплексної програмної платформи для створення хмарної інфраструктури на основі OpenStack.

Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) *Слайди у форматі Power Point (назва, мета і актуальність роботи, подання та основні характеристики хмарних обчислень, функціональна архітектура, моделі обслуговування та типи розгортання хмарних обчислень, існуючі рішення програмних продуктів для розгортання і надання хмарних сервісів, результати порівняння програмних рішень для розгортання і надання хмарних сервісів, спрощена базова функціональна архітектура OpenStack, підходи щодо розробки архітектурного рішення комплексної програмної платформи розгортання хмарної інфраструктури, архітектура комплексної програмної платформи для розгортання та надання хмарних послуг, особливості та сутності основних сервісних модулів, висновки)*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Ознайомлення із завданням. Уточнення ТЗ.*	18.05 – 20.05.26	*виконано*
2	*Підбір літератури за темою роботи.*	21.05 – 25.05.26	*виконано*
3	*Виконання розділу 1*	26.05 – 06.06.26	*виконано*
4	*Виконання розділу 2*	07.06 – 17.06.26	*виконано*
5	*Виконання розділу 3*	18.06 – 02.07.26	*виконано*
7	*Оформлення пояснювальної записки*	03.07 –08.07.26	*виконано*
8	*Оформлення презентаційного матеріалу,*		
	підготовка до захисту у ЕК, захист.	09.07 – 18.07.26	*виконано*

Дата видачі завдання *18 травня 2026 р.*

Здобувачка _____
(підпис)

Керівник роботи _____ *(доц. Юрій КОЛТУН)*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 78 с., 20 рис., 1 табл., 27 джерел, 2 додатки.

ХМАРНІ ОБЧИСЛЕННЯ, ХМАРНА ІНФРАСТРУКТУРА, AOMEI TECHNOLOGY MULTCLOUD, VMWARE VSPHERE, NUTANIX ENTERPRISE CLOUD PLATFORM, OPENNEBULA, OPENSTACK, АРХІТЕКТУРА, КОМПЛЕКСНА ПРОГРАМНА ПЛАТФОРМА, API, OPENSTACK API МОДУЛЬНА АРХІТЕКТУРА, МІКРОСЕРВІСНИЙ ПІДХІД, СЕРВІС, СЕРВІСНИЙ МОДУЛЬ, СУТНІСТЬ, ВІРТУАЛІЗАЦІЯ, ПЛАТФОРМА ВІРТУАЛІЗАЦІЇ

Об'єкти дослідження – хмарні обчислення, платформа OpenStack, сервісні модулі комплексної програмної платформи.

Мета роботи – обґрунтування підходів щодо створення розподіленої комплексної програмної платформи для розгортання хмарної інфраструктури, визначення складу її сервісних модулів та розробка типового архітектурного рішення на основі платформи OpenStack.

Розглянуто фундаментальну концепцію, загальні принципи, типи і моделі організації хмарних обчислень. Проведений аналіз та порівняння комплексних рішень існуючих програмних продуктів для розгортання і надання хмарних сервісів. Зокрема були проаналізовані платформи AOMEI Technology MultCloud, VMware vSphere, Nutanix Enterprise Cloud Platform, OpenNebula, OpenStack. Розроблена архітектура комплексної програмної платформи для розгортання та надання хмарних послуг на базовій основі OpenStack. Для цього обґрунтоване застосування мікросервісного підходу та визначено склад основних сервісних модулів.

THE ABSTRACT

Explanatory note 78 pages, 20 fig., 1 tab., 27 sources, 2 app.

CLOUD COMPUTING, CLOUD INFRASTRUCTURE, AOMEI TECHNOLOGY MULTCLOUD, VMWARE VSPHERE, NUTANIX ENTERPRISE CLOUD PLATFORM, OPENNEBULA, OPENSTACK, ARCHITECTURE, INTEGRATED SOFTWARE PLATFORM, API, OPENSTACK API MODULAR ARCHITECTURE, MICROSERVICES APPROACH, SERVICE, SERVICE MODULE, ENTITY, VIRTUALIZATION, VIRTUALIZATION PLATFORM

Objects of research – cloud computing, the OpenStack platform, service modules of a comprehensive software platform.

The purpose of work – justification of approaches to creating a distributed, comprehensive software platform for deploying cloud infrastructure; defining the composition of its service modules; and developing a model architectural solution based on the OpenStack platform.

The fundamental concept, general principles, types, and models of cloud computing organization are examined. An analysis and comparison of comprehensive solutions offered by existing software products for deploying and delivering cloud services is conducted. In particular, the following platforms were analyzed: AOMEI Technology MultCloud, VMware vSphere, Nutanix Enterprise Cloud Platform, OpenNebula, and OpenStack. An architecture for a comprehensive software platform for deploying and providing cloud services based on OpenStack has been developed. To this end, the use of a microservices approach has been justified, and the composition of the main service modules has been defined.

ЗМІСТ

С.

Перелік скорочень 9

Вступ 11

1 Фундаментальна концепція, загальні принципи, типи і моделі організації хмарних обчислень 14

1.1 Основні етапи і фактори що впливали на розвиток хмарних обчислень 14

1.2 Ключові характеристики та базова функціональна архітектура хмарних обчислень 16

1.3 Моделі обслуговування сервісів у хмарі та її типи 20

1.3.1 Моделі обслуговування хмарних сервісів 20

1.3.2 Типи хмар за своїм призначенням 25

2 Аналіз та порівняння комплексних рішень програмних продуктів для розгортання і надання хмарних сервісів 30

2.1 Платформа AOMEI Technology MultCloud 30

2.2 Платформа VMware vSphere 31

2.3 Платформа Nutanix Enterprise Cloud Platform 32

2.4 Платформа OpenNebula 33

2.5 Платформа OpenStack 35

2.6 Порівняння комплексних програмних платформ в аспекті розгортання гібридної хмари 36

3 Розробка архітектури комплексної програмної платформи для створення хмарної інфраструктури на основі OpenStack 38

3.1 Спрощена функціональна архітектура OpenStack 38

3.2 Обґрунтування та визначення архітектурного рішення комплексної програмної платформи для розгортання хмарних обчислень 42

3.3 Особливості та сутності основних сервісних модулів розробленої архітектури комплексної програмної платформи 47

3.3.1 Сервіс авторизації – Identity 47

3.3.2 Сервіс керування блоковими сховищами – Block storage 50

3.3.3 Сервіс образів серверів – Images 53

3.3.4 Сервіс керування мережною інфраструктурою – Networking 54

3.3.5 Сервіс керування VM – Compute 57

3.3.6 Сервіс керування контейнерними кластерами Kubernetes – Magnum 60
Висновки 63
Перелік джерел посилання 66
Додаток А Архітектура комплексної програмної платформи для розгортання та надання хмарних послуг 69
Додаток Б Слайди презентації 70

ПЕРЕЛІК СКОРОЧЕНЬ

API (Application Programming Interface) – прикладний програмний інтерфейс;

CI/CD (Continuous Integration / Continuous Delivery) – безперервна інтеграція / безперервне розгортання;

DB (Data Base) – база даних;

gRPC (Google Remote Procedure Call, gRPC) – протокол віддаленого виклику процедур;

HTTP (Hypertext Transfer Protocol) – протокол передачі (переносу) гіпертексту;

IaaS (Infrastructure as a Service) – інфраструктура, як сервіс;

IT (Information Technology) – інформаційні технології;

KVM (Kernel-based Virtual Machine) – віртуальна машина на основі ядра;

MapReduce – модель розподілених обчислень, що використовується для паралельних обчислень над дуже великими, кілька петабайт, наборами даних у комп'ютерних кластерах;

NGN (Next Generation Network) – мережа наступного покоління;

NIST (National Institute of Standards and Technology) – Національний інститут стандартів і технологій (США);

PaaS (Platform as a Service) – платформа як сервіс;

REST (Representational State Transfer) – це архітектурний стиль для створення web-сервісів, набір правил і обмежень, який визначає, як компоненти розподілених систем (наприклад, клієнт і сервер), що повинні взаємодіяти один з одним через Інтернет, використовуючи стандартні HTTP-методи для управління ресурсами (даними).

S3 (Simple Storage Service) – проста служба зберігання;

SaaS (Software as a Service) – програмне забезпечення як сервіс;

UUID (Universally Unique Identifier) – універсальний унікальний ідентифікатор);

VDI (Virtual Desktop Infrastructure) – віртуальний робочий стіл;

VLAN (Virtual Local Area Network) – віртуальна локальна мережа;

VM (Virtual Machine) – віртуальна машина.

АПЗ – апаратно-програмні засоби;

БД – база даних;

МСА – мікросервісна архітектура;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

ЦОД – центр обробки даних.

ВСТУП

Традиційно розгортання, як обчислювальних, так і програмних ресурсів здійснюється безпосередньо в локальній інфраструктурі інформаційно-технологічної (ІТ) компанії. Такий підхід потребує значних початкових фінансових інвестицій в апаратно-програмні засоби (АПЗ), мережну інфраструктуру та персонал, що зумовлює здійснювати достатньо високі капітальні витрати. Водночас він надає можливість реалізувати повний контроль над інфраструктурою, що розгорнута, та її АПЗ. Проте вже на цей час багато ІТ-компаній стикається з проблемою не завжди ефективного використання придбаних обчислювальних ресурсів, програмного забезпечення (ПЗ) та іншого обладнання. Наприклад, така ситуація може виникнути у разі припинення розробки програмних проєктів, які не виправдали очікувань, хоча необхідна для їх створення, тестування та подальшого впровадження інфраструктура вже була придбана і розгорнута. Крім того, значна кількість підрозділів, що була організована для здійснення розробки різних програмних продуктів, нерідко потребує використання різних платформ віртуалізації та обчислювальних середовищ. Це ускладнює розподіл ресурсів між підрозділами та призводить до технологічної фрагментації внутрішньої корпоративної структури компанії знову таки, як на апаратному, так і програмно-інформаційному рівні. Водночас більшість програмних проєктів, що знаходяться у стадії здійснення розробки, відладки та тестування, використовує типовий набір програмних компонентів, зокрема системи баз даних (БД), системи обміну та обробки даних, веб-сервери, кеші, проксі, та інше. Їх розгортання та подальшій супровід потребують залучення відповідних фахівців у кожному окремому підрозділі компанії. Одним із можливих шляхів розв’язання зазначених вище проблем є централізоване надання обчислювальних ресурсів і необхідного ПЗ [1].

Одним із перших підходів щодо централізації ресурсів в рамках зазначеної задачі стало їх розміщення в так званих хостингових компаніях або по іншому у хостинг-провайдерів (хостинг). Подальший розвиток цього підходу отримав продовження завдяки появі концепції надання інфокомунікаційних послуг на основі мереж наступного покоління (Next Generation Networks, NGN), яка надає можливості кінцевим користувачам здійснювати мультисервісний доступ до різноманітних послуг, а також до інших ресурсів, незалежно від технології

доступу або від типу мережі, до якої підключений цей кінцевий користувач. Використання хостингу, а у подальшому концепції NGN, дозволяє суттєво знизити витрати на створення та підтримку ІТ-інфраструктури за рахунок здійснення оренди потрібних обчислювальних ресурсів та ПЗ, або, навіть, всієї мережної інфраструктури з персоналом, який нею опікується. Разом із цим користувач отримує менший рівень контролю над орендованими ресурсами, а модель оплати, як правило, базується на фіксованому обсязі ресурсів, що були виділені, незалежно від інтенсивності їх використання. Тому, хоча такий підхід є економічно вигіднішим у порівнянні із обслуговуванням своєї локальної інфраструктури, але він не усуває всіх існуючих недоліків. Насамперед це стосується відсутності моделі оплати за фактичне використання того або іншого ресурсу, а також недостатнього рівня незалежності ПЗ від конкретної фізичної апаратної платформи [1].

Тому наступним етапом розвитку ІТ стало впровадження концепції хмарних обчислень (Cloud Computing), яка дозволила усунути більшість зазначених обмежень і зробити роботу з ресурсами більш гнучкою. У цьому випадку АПЗ (як правило у віртуалізованому вигляді) розміщуються у інфраструктурі хмарного провайдера, а користувач отримує доступ до них за гнучкою моделлю сплати лише за фактично використані ресурсів. Такий підхід забезпечує суттєве зниження початкових витрат, оскільки дозволяє перейти від капітальних фінансових вливань до здійснення поточних операційних витрат. За такого підходу контроль над орендованими у «хмарі» ресурсами у значній мірі покладається на постачальника хмарних послуг, а користувач до цього процесу практично не має відношення [1].

Таким чином, хмарні обчислення являють собою модель надання, розміщення та використання АПЗ, сервісів і застосунків, за якої доступ до них здійснюється через мережу Інтернет у вигляді хмарних веб-сервісів. До таких веб-сервісів можна отримати доступ для їх використання із різних апаратних і програмних платформ, а їх оплата робиться у відповідності із фактичним обсягом або часом використання [2].

Хмарні обчислення функціонально поділяються на два рівні: «front-end» і «back-end». Перший, front-end рівень – являє собою шар користувача, з яким працює користувач під час роботи з хмарними сервісами. Користувач отримує доступ до інформаційних ресурсів через зовнішній сервіс з оплатою відповідно до обсягу ресурсів, що ним споживаються. Наприклад, у поштовому Gmail-сервісі

front-end являє собою інтерфейс роботи з електронною поштою в браузері. Другий, back-end рівень – приховується від впливу користувача внутрішнім шаром хмарної інфраструктури. До його складу входить обладнання, програмні сервіси та центри обробки даних (ЦОД), які підтримують функціонування хмарного сервісу [3].

У цій кваліфікаційній роботі береться до уваги, що використання сучасних технологій і засобів розроблення back-end, дає можливість створити комплексну програмну платформу для розгортання хмарної інфраструктури та централізованого надання на її основі різних ресурсів і послуг. Звідси метою роботи є пошук оптимального підходу до створення розподіленої комплексної програмної платформи, яка забезпечуватиме надання обчислювальних ресурсів на основі моделей обслуговування хмарних обчислень «інфраструктура як сервіс» (Infrastructure as a Service, IaaS) та «платформа як сервіс» (Platform as a Service, PaaS), з можливістю підтримки здійснення простого і гнучко масштабування окремих компонентів системи відповідно до поточного навантаження. Водночас аналіз сучасних існуючих програмних рішень показує, що більшість із них має низку суттєвих обмежень. До таких недоліків належать обмежена підтримка різних платформ віртуалізації, недостатні можливості інтеграції з різними хмарними провайдерами, а також відсутність механізмів, які б дозволяли користувачу самостійно обирати платформу для надання необхідних хмарних сервісів. Саме тому аналіз підходів і рішень в аспекті реалізації розподілених комплексних платформ розгортання хмарної інфраструктури та надання на її основі хмарних сервісів є актуальним для виконання практичним завданням.

1 ФУНДАМЕНТАЛЬНА КОНЦЕПЦІЯ, ЗАГАЛЬНІ ПРИНЦИПИ, ТИПИ І МОДЕЛІ ОРГАНІЗАЦІЇ ХМАРНИХ ОБЧИСЛЕНЬ

1.1 Основні етапи і фактори що впливали на розвиток хмарних обчислень

Хмарні обчислення – це концепція організації обчислювальних ресурсів, яка ґрунтується на використанні великих ЦОД, доступних одночасно значній кількості користувачів і організацій. Консолідація ресурсів у таких дата-центрах забезпечує не тільки ефективне територіальне масштабування фізичних ресурсів в аспекті його розміщення, але й високу продуктивність, можливість гнучкого функціонального масштабування та значний рівень резервування обладнання і сервісів. З практичної точки зору хмарні обчислення являють собою модель надання обчислювальних ресурсів за запитом, після підтвердження якого користувач через мережу Інтернет отримує віддалений доступ до мережних структур, ЦОД, конкретних фізичних або віртуальних серверів, ПЗ, БД або веб-сервісів (служб) зберігання даних (сховищам), та інших інформаційних ресурсів [4].

Формування концепції хмарних обчислень стало результатом тривалого еволюційного розвитку інформаційних технологій. Однією з перших передумов її появи стало стрімке зростання пропускної здатності мережі Інтернет у 90-х роках XX століття та можливості доступу до неї широкого кола користувачв. Саме це створило технічні передумови для перенесення обчислювальних ресурсів у віддалені ЦОД, хоча на той час більшість компаній ще не мала достатнього рівня розвитку АПЗ для практичного впровадження такого підходу. Але вже у 1993 році термін «хмара» вперше був використаний у комерційній сфері для позначення великих інфокомунікаційних мереж, у яких реалізовувалася високошвидкісна одночасна передача різних видів трафіку (зокрема даних, голосу та відео). Для графічного позначення таких мереж на структурно-архітектурних схемах почали використовувати умовний символ у вигляді хмари. Завдяки цьому, поняття «хмара» міцно закріпилося в офіційній мережній термінології, де воно символізує абстракцію складних мережних інфраструктур, що охоплюють програмні, апаратні, обчислювальні ресурси та сервіси зберігання даних (рис. 1.1) [4, 5].

Подальший розвиток концепції хмарних обчислень пов'язаний із появою комерційних платформ, які реалізували нові моделі надання інформаційних ресурсів і сервісів через мережу Інтернет. Саме ці рішення заклали основу

сучасних моделей хмарного обслуговування, що на цей час широко використовуються.

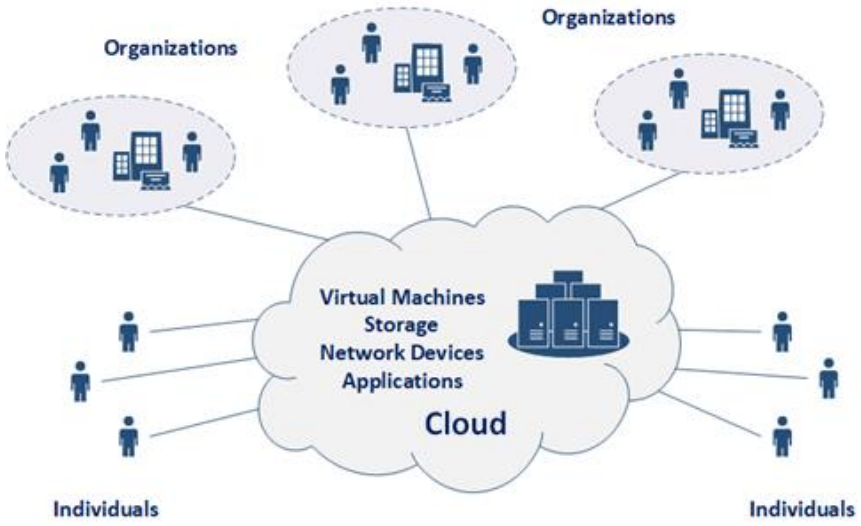


Рисунок 1.1 – Абстрактне подання хмарних обчислень

Одним із перших важливих етапів стало створення у 1999 році компанією Salesforce.com сервісу, який надав користувачам можливість працювати з прикладним ПЗ без його встановлення на локальному ПК. Доступ до додатків здійснювався через веб-інтерфейс, що фактично започаткувало реалізацію моделі «програмне забезпечення як сервіс» (Software as a Service, SaaS). У подальшому саме ця модель стала однією з базових моделей надання хмарних послуг [5].

Наступним важливим кроком став запуск компанією Amazon у 2002 році хмарного веб-сервісу Simple Storage Service (S3), що забезпечував централізоване зберігання інформації на віддалених серверах із можливістю здійснення доступу до неї через мережу Інтернет практично з будь-якого пристрою через веб-браузер. Окрім зберігання даних, сервіс надавав користувачам можливість використовувати віддалені обчислювальні ресурси, що стало одним із перших прикладів практичної реалізації хмарних технологій [5].

Подальший розвиток цієї концепції хмарних обчислень відбувся у 2006 році, коли компанія Amazon представила хмарний веб-сервіс EC2 (Elastic Compute Cloud – еластична обчислювальна хмара). Його основною особливістю стала можливість запуску додатків користувача на віддалених обчислювальних ресурсах без необхідності придбання власного обладнання. Веб-сервіси S3 і EC2 стали одними із перших широкодоступних рішень, що продемонстрували

практичні переваги використання хмарних обчислень і сформували повноцінну хмарну платформу [5].

З невеликим запізненням, у 2008 року компанією Microsoft було оголошено про створення платформи Windows Azure, яка вже на початку 2009 року стала доступною в режимі бета-тестування під робочою назвою Project Red Dog. Надалі Microsoft активно розвивала платформу шляхом введення різноманітних веб-сервісів, як інфраструктурного напряму, так і прикладних. В результаті у 2012 році була представлена повноцінна версія хмарної платформи [6].

Платформа Windows Azure була створена як операційна система (ОС) для центрів обробки даних, що забезпечує комплекс засобів для створення, розгортання та експлуатації різноманітних хмарних сервісів. Використання цієї платформи дало можливість значно скоротити час від початку здійснення розробки проекту до його введення в експлуатацію, оскільки усувало необхідність придбання обладнання і операційних систем [7].

Ще одним важливим етапом розвитку хмарних технологій стало створення компанією Google у 2009 році платформи Google Apps. Вона об'єднала комплекс готових інструментів, до складу яких входять обчислювальні ресурси, системи зберігання даних, засоби адміністрування, сервіси аналітики, технології машинного навчання, інформаційної безпеки, обробки даних та Інтернету речей. Крім того, платформа підтримує використання контейнерних технологій для побудови хмарної інфраструктури та забезпечує засоби централізованого управління нею. Одним із найвідоміших прикладів використання хмарних обчислень від компанії Google є сервіс електронної пошти Gmail, у якому всі програмні функції, обчислення та зберігання даних виконуються у віддаленій хмарній інфраструктурі, а користувач взаємодіє із сервісом через веб-браузер або мобільний додаток.

1.2 Ключові характеристики та базова функціональна архітектура хмарних обчислень

На цей час не існує ні стандартного, ні стандартизованого, ні універсального визначення хмарних обчислень, тому що вони знаходяться у постійному розвитку відповідно до технологічних змін, що відбуваються в інфокомунікаційних системах, IT сфері, і взагалі у сучасному інформаційному суспільстві для реалізації можливості

забезпечення доступу до всіх видів інформаційних ресурсів. Попри існування великої кількості визначень і підходів до трактування поняття «хмарні обчислення», найбільшого поширення в науковій літературі та міжнародній практиці набуло визначення, запропоноване Національним інститутом стандартів і технологій США (National Institute of Standards and Technology, NIST). Відповідно до визначення NIST, хмарні обчислення – це модель організації доступу до обчислювальних ресурсів, яка забезпечує зручний мережний доступ на вимогу до спільного пулу конфігурованих ресурсів, зокрема мереж, серверів, систем зберігання даних, прикладного ПЗ та різноманітних сервісів. Такі ресурси можуть бути оперативно виділені або звільнені з мінімальними зусиллями з боку користувача та без необхідності безпосереднього втручання постачальника послуг. Крім того, згідно з рекомендаціями NIST, повноцінна хмарна інфраструктура повинна відповідати низці базових характеристик, які визначають принципи надання, використання та здійснення управління хмарними сервісами (рис. 1.2) [3, 4, 8]:

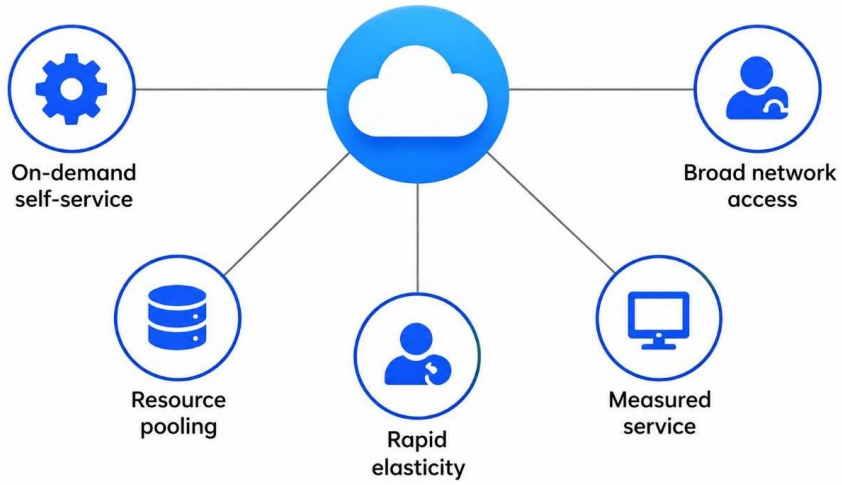


Рисунок 1.2 – Ключові характеристики хмарних обчислень

- самообслуговування на вимогу (On-demand Self-Service) – користувач може самостійно та в будь-який момент мати доступ до необхідних обчислювальних ресурсів (віртуальних машин, сховищ даних, мережних ресурсів та інших веб-сервісів), використовуючи для цього веб-браузер або інший прикладний інтерфейс доступу. При цьому відсутня потреба в поданні запитів адміністрації хмарного провайдера, а також немає потреби у придбанні фізичного обладнання;

- широкий мережний доступ (Broad Network Access) – хмарні ресурси розміщуються у розподілених ЦОД і доступні через мережу Інтернет із будь-якого

місця, будь-яких мереж та за допомогою різних пристроїв користувача (зокрема ПК, ноутбуків, планшетів і смартфонів). Доступ до веб-сервісів та послуг здійснюється через стандартні інтерфейси, такі як веб-браузер або спеціалізованих додатків;

- об'єднання ресурсів (Resource Pooling) – фізичні та віртуальні ресурси об'єднуються в єдиний пул і спільно використовуються багатьма користувачами, що мають до них доступ. Виділення та перерозподіл ресурсів між користувачами здійснюються автоматично відповідно до їх поточних потреб без впливу на роботу інших користувачів;

- швидка еластичність (Rapid Elasticity) – обсяг обчислювальних ресурсів може автоматично і оперативно збільшуватися (тобто додатково виділятися користувачу) або зменшуватися (тобто звільнятися) за необхідності або залежно від зміни навантаження. З боку користувача такі ресурси практично виглядають необмеженими та доступними в необхідному обсязі у будь-який момент часу;

- вимірюваний сервіс (Measured Service) – використання обчислювальних ресурсів постійно контролюється та автоматично здійснюється їх облік з метою оптимізації. Це забезпечує прозорість використання ресурсів як для постачальника, так і для користувача, а оплата здійснюється відповідно до фактичного обсягу спожитих ресурсів.

Окрім вищезазначених ключових характеристик, що визначені рекомендаціями NIST, також важливими практичними характеристиками технологій хмарних обчислень є масштабованість і віртуалізація, які забезпечують ефективне використання обчислювальних ресурсів та гнучкість функціонування хмарної інфраструктури [9].

Масштабованість – це здатність хмарної системи автоматично адаптувати обсяг доступних ресурсів відповідно до зміни навантаження. Така адаптація може бути пов'язана зі збільшенням або зменшенням кількості користувачів, потребами у віртуальному дисковому просторі чи обчислювальній потужності. Завдяки цьому ресурси можуть оперативно виділятися або звільнятися відповідно до поточних потреб користувачів і додатків [9].

Основою реалізації більшості хмарних платформ є віртуалізація, яка забезпечує абстрагування фізичних ресурсів та їх ефективне використання. Ця характеристика має дві основні властивості [9]:

- абстракція – дає змогу об'єднати фізичні обчислювальні та мережні ресурси, а також системи зберігання даних у єдиний пул ресурсів і сформувати уніфікований рівень їх подання. У результаті користувачі та програмні сервіси працюють не з

фізичним обладнанням, а з його віртуальними поданнями, такими як віртуальні сервери, серверні кластери, файлові системи або системи управління БД;

- інкапсуляція – забезпечує ізоляцію програмних компонентів і середовищ виконання, що підвищує рівень безпеки, спрощує адміністрування та забезпечує незалежне функціонування окремих додатків.

Треба зазначити, що віртуалізація є основою функціонування хмарної інфраструктури. Використання технологій віртуалізації дає змогу розміщувати кілька віртуальних машин (Virtual Machine, VM) на одному фізичному сервері, що суттєво підвищує ефективність використання апаратних ресурсів. Завдяки цьому з'являється можливість більш раціонально розподіляти обчислювальне навантаження між фізичними серверами та оперативно реагувати на його зміни. За наявності вільних ресурсів віртуалізація дозволяє без зупинки функціонуючих сервісів збільшувати кількість процесорних ядер або обсяг оперативної пам'яті, що вже були виділені працюючим VM. Крім того, за потреби можна швидко створювати копії існуючих VM. Важливою перевагою сучасних платформ віртуалізації є підтримка технології «живої» міграції VM, яка дає змогу переносити працюючі VM між фізичними серверами практично без переривання їх роботи. Це забезпечує ефективне балансування навантаження, підвищує відмовостійкість системи та дозволяє виконувати технічне обслуговування обладнання без припинення надання сервісів. Крім експлуатації віртуальних систем, сервісів і додатків у хмарі, віртуалізація широко використовується у процесі здійснення розробки, тестування та відладки ПЗ. Вона створює середовища, які максимально точно відтворюють конфігурацію майбутньої системи, у якій буде у тестовому режимі функціонувати програмний продукт після його розробки. Таким чином, використання технологій віртуалізації є важливою характеристикою, на яку треба спиратися при розгортанні сучасних хмарних платформ, тому що вона забезпечує їх гнучкість, швидке розгортання та масштабування сервісів, а також ефективний розподіл обчислювальних ресурсів і, як наслідок, зменшення експлуатаційних витрат [10].

Саме на особливостях і властивостях технологій віртуалізації ґрунтується архітектура хмарних обчислень, яка подається у вигляді двох компонент: front-end і back-end. Ці компоненти взаємодіють між собою через мережу Інтернет та забезпечують надання обчислювальних ресурсів і сервісів за запитом користувача (рис. 1.3) [9, 11].

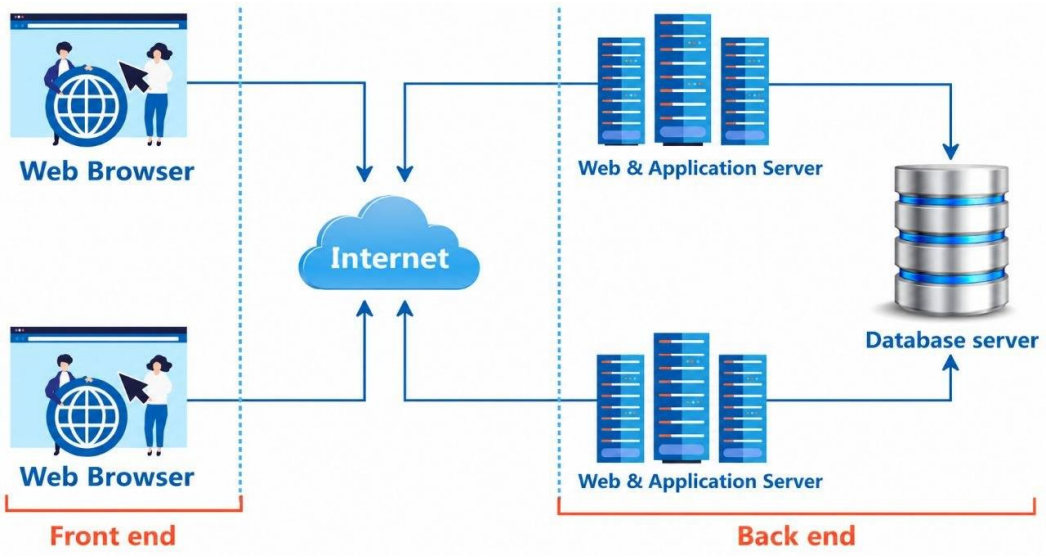


Рисунок 1.3 – Базова функціональна архітектура хмарних обчислень

Усе, з чим взаємодіє кінцевий користувач, є частиною інфраструктури front-end. До його складу входить інтерфейс користувацький – це результат інтеграції різних підкомпонентів, таких як браузери, планшети, мобільні пристрої тощо. За допомогою front-end кінцевий користувач може підключитися до інфраструктури хмарних обчислень. Іншими словами, «фронтенд» – це той компонент, з яким взаємодіє користувач [11].

Back-end – це все, чого користувач зазвичай не бачить, і все, що plsqcy.' обробку даних. Хмарний провайдер використовує back-end для управління всіма ресурсами, що є необхідними для надання хмарних сервісів та послуг, такими як сховища даних, механізми безпеки, VM, моделі розгортання послуг, сервери, механізми управління трафіком тощо. Іншими словами, «back-end» – це той компонент, з яким взаємодіє хмарний провайдера [11].

1.3 Моделі обслуговування сервісів у хмарі та її типи

1.3.1 Моделі обслуговування хмарних сервісів

Згідно рекомендаціям інституту NIST, визначення та ключові характеристики якого стосовно концепції організації хмарних обчислень були розглянуті вище, існує багато моделей, що регламентують здійснення обслуговування хмарних сервісів. Але серед них виділяють три основні, які

відображають ріні найпоширеніші способи і підходи щодо надання обчислювальних ресурсів, сервісів і додатків користувачам через Інтернет, а також здійснення їх розробки та підтримки функціональності. Зокрема це моделі, що вже були згадувані раніше: IaaS, PaaS, SaaS. Їх особливість полягає в тому, що у абстрактному наближенні вони формують площинну модель хмарних обчислень (рис. 1.4) [3, 4].

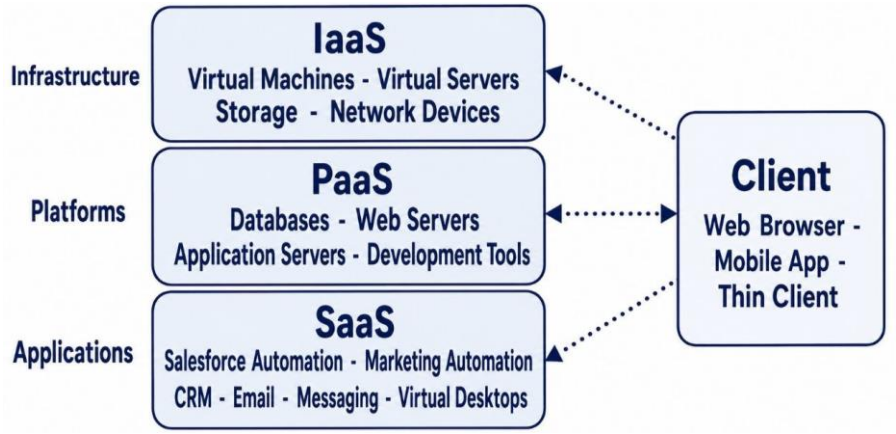


Рисунок 1.4 – Основні моделі обслуговування хмарних севісів

1) Модель IaaS забезпечує користувачам доступ до обчислювальних ресурсів, систем зберігання даних і мережної інфраструктури у вигляді сервісу. Доступ і отримання хмарних ресурсів здійснюється за потребою на вимогу, а їх оплата, як правило, залежить від фактичного обсягу споживання. Такий підхід дозволяє компаніям відмовитися від придбання власного серверного обладнання та використовувати необхідні ресурси лише тоді, коли вони їм потрібні [4, 12].

У цій моделі хмарний провайдер сервісів IaaS забезпечує функціонування і контроль за фізичною інфраструктурою хмари, до складу якої входять ЦОД, серверне обладнання, системи зберігання даних, інше мережне обладнання та платформа віртуалізації (гіпервізор). Натомість користувач отримує можливість самостійно здійснювати розгортання, налаштування і гнучке управління ОС, VM сховищами даних, ПЗ та власними додатками [4, 12].

Основними характеристиками моделі IaaS є [12]:

- надання обчислювальних ресурсів у вигляді хмарного сервісу;
- висока масштабованість обчислювальної інфраструктури;
- динамічне надання та вивільнення ресурсів відповідно із потребами користувача;

- можливість здійснення управління ресурсами через графічний веб-інтерфейс або прикладний програмний інтерфейс (Application Programming Interface, API);

- підтримка інструментів автоматизації процесів адміністрування та розгортання інфраструктури.

До основних переваг моделі IaaS належать [12]:

- можливість швидкого автоматизованого розгортання серверів, мереж і систем зберігання даних;

- зменшення витрат на придбання та модернізацію фізичного обладнання компанії завдяки оплаті лише за використані ресурси;

- повний контроль користувачем над ОС, ПЗ і конфігурацією своєї інфраструктури у хмарі;

- оперативне надання додаткових ресурсів за вимогою хмарним провайдером;

- можливість здійснення швидкого масштабування наданих ресурсів відповідно із змінам навантаження та потреби.

Разом із перевагами модель IaaS має і певні обмеження. Зокрема, користувач самостійно відповідає за розгортання, налаштування та безпечне функціонування ОС і прикладного ПЗ. Крім того, на нього покладається відповідальність за цілісність, резервне копіювання та відновлення власних даних. Хмарний провайдер, що постачає послуги IaaS, як правило, забезпечує лише доступ до обчислювальної інфраструктури та засобів управління нею, тоді як конфігурування ОС, програмних платформ і додатків виконується безпосередньо користувачем [12].

2) Модель PaaS призначена для розробки, тестування, розгортання та супроводу ПЗ. Вона надає користувачам готову програмну платформу, яка містить необхідні інструменти, середовище розробки та засоби управління додатками. При цьому масштабування ресурсів здійснюється автоматично відповідно до поточного навантаження.

У цій моделі хмарний провайдер бере на себе здійснення управління середовищем розробки, серверами, системами зберігання даних, мережним обладнанням, ОС та програмною платформою. Користувач, у свою чергу, зосереджується виключно на розробці, налаштуванні та супроводі своїх прикладних додатків без необхідності здійснення адміністрування базової інфраструктури. Завдяки цьому модель PaaS забезпечує повний набір засобів, які

є необхідними для підтримки всіх етапів життєвого циклу веб-додатків і послуг, від їх розробки та тестування до розгортання та подальшої експлуатації. Це дозволяє значно скоротити час на створення додатків та на їх розгортання [4, 12].

Основними характеристиками моделі PaaS є [12]:

- використання технологій віртуалізації, що забезпечують автоматичне масштабування (надання або звільнення) обчислювальних ресурсів відповідно із наявними потребами користувачів;

- підтримка різних мов програмування, програмних платформ і фреймворків;

- інтеграція із БД, веб-сервісами та іншими компонентами хмарної інфраструктури.

До основних переваг моделі PaaS належать [12]:

- простий та ефективний процес розробки і розгортання веб-додатків;

- немає необхідності в адмініструванні програмної платформи та базової інфраструктури;

- підтримка автоматизації процесів розробки, тестування та розгортання веб-додатків;

- можливість інтеграції з гібридними хмарними технологіями;

- зосередження розробників на створенні додатків та їх функціоналу без необхідності здійснювати управління ОС та хмарною інфраструктурою;

- підтримка групової розробки програмного проекту кількома розробниками.

До недоліків PaaS можна віднести наступні обмеження і зауваження. Зокрема, користувач отримує контроль лише над власними додатками та їх програмним кодом, тоді як інфраструктура і програмна платформа залишаються під управлінням хмарного провайдера. Крім того, зберігання даних у середовищі провайдера може створювати додаткові вимоги до забезпечення інформаційної безпеки. Також важливим фактором є правильний хмарного провайдера, оскільки функціональні можливості різних платформ PaaS можуть дуже відрізнятися. Окрім цього, використання специфічних (пропрієтарних) технологій і сервісів окремого провайдера може призвести до залежності від його інфраструктури, що може ускладнити подальшу міграцію додатків до інших хмарних платформ [12].

3) Модель SaaS є моделлю хмарних обчислень, за якої готові програмні продукти надаються користувачам через мережу Інтернет. Доступ до таких додатків здійснюється переважно за допомогою веб-браузера або із

спеціалізованого клієнтського додатку без необхідності їх персональної інсталяції на комп'ютері користувача. У SaaS користувачу не потрібно знати, де фізично розміщене ПЗ, за допомогою яких технологій воно розроблене, або яка ОС використовується на сервері. Вирішення всіх цих питань, що пов'язані з функціонуванням ПЗ, веб додатків і сервісів, їх оновленням, технічною підтримкою та адмініструванням, – бере на себе хмарний провайдер. Завдяки цьому користувачі завжди працюють з актуальною версією ПЗ. Водночас вони не мають можливості здійснювати управління обчислювальною інфраструктурою хмари, системами зберігання даних чи іншими ресурсами, на яких функціонує сервіс [4, 12].

Основними характеристиками моделі SaaS є [12]:

- централізоване розміщення та адміністрування ПЗ;
- виконання програм на віддалених серверах хмарного провайдера;
- доступ до додатків через мережу Інтернет;
- відсутність необхідності встановлення, оновлення та супроводу ПЗ з боку користувача;
- оплата сервісу відповідно із моделлю підписки до сервісів або за фактичним їх використанням.

До основних переваг моделі SaaS належать [12]:

- швидкі впровадження та початок використання ПЗ без здійснення складних процесів налаштування;
- зменшення витрат у порівнянні із вкладеннями у персональне ПЗ;
- автоматичне оновлення та технічна підтримка ПЗ хмарним провайдером;
- відсутність потреби у використанні персональних апаратних ресурсів для інсталяції та використання ПЗ;
- широкий вибір готових хмарних сервісів і додатків для виконання різноманітних прикладних задач;
- можливість доступу до ПЗ з будь-якого термінального обладнання користувача через веб-браузер або клієнтський додаток.

Обмеження модель SaaS, що пов’язані з її недоліками полягають в наступному. Так, можливості інтеграції з іншими інформаційними системами визначаються хмарним провайдером, що обмежує можливість їх самостійного налаштування. Крім того, деякі SaaS-рішення можуть бути несумісними з програмними продуктами або обладнанням, які вже використовуються в компанії. Також до обмежень можна віднести залежність від рівня інформаційної безпеки,

який забезпечується хмарним провайдером, оскільки компрометація його інфраструктури може призвести до втрати конфіденційності або цілісності даних користувачів.

На рисунку 1.5 більш докладно показані основні відмінності розглянутих вище моделей обслуговування по зонам забезпечення управління ресурсами хмарним провайдером та користувачами [7].

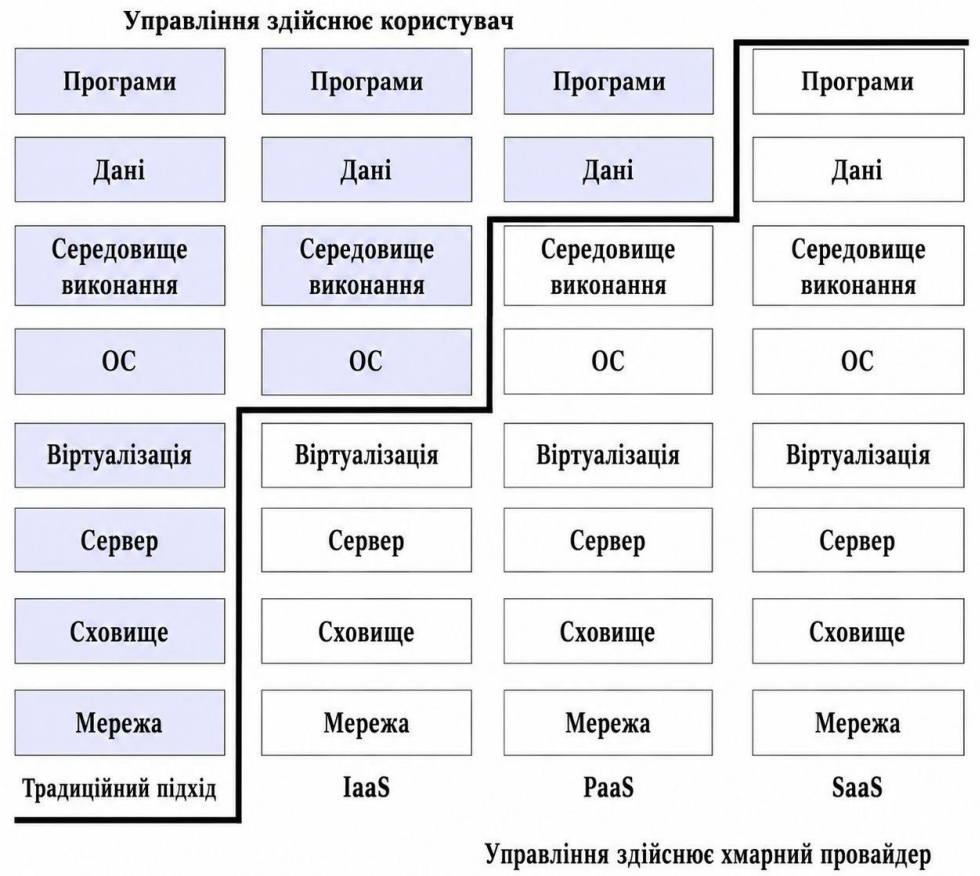


Рисунок 1.5 – Розподіл зон управління ресурсами між хмарним провайдером та користувачем у відповідності до моделей IaaS, PaaS та SaaS

1.3.2 Типи хмар за своїм призначенням

За типом призначення хмарні технології розгортаються з урахуванням визначення переваг та недоліків, що залежать від ряду факторів: моделей надання послуг, рівня доступності, типі даних, що обробляються, тощо. Серед них виділяють три основні типи таких хмар (рис. 1.4) [13].



Рисунок 1.6 – Основні типи хмар

1) Публічна хмара – це IT-інфраструктура, якою користуються численні компанії та окремі користувачі. При цьому користувачі не мають можливості здійснювати управління «хмарою» і не приймають участь у її обслуговуванні. Вся відповідальність щодо питань підтримки працездатності сервісів покладається на власника «хмари». Підписатися на публічну хмару може будь-яка компанія, організація чи окремий індивідуальний користувач. «Хмари» такого типу забезпечують простий і доступний за вартістю спосіб розгортання веб-сайтів або бізнес-систем із високою масштабованістю, які є недоступні в «хмарах» інших типів [13, 14].

Проте використання публічної хмари не гарантує належного рівня захисту даних клієнта, що може бути критичним, наприклад, у разі роботи з персональними даними користувачів. Також можлива ситуація, коли велика компанія має значну кількість власних ресурсів і використовує традиційну IT-інфраструктуру, унаслідок чого виникає низька проблем, зокрема таких як [14]:

- збільшується кількість обладнання та ускладняється процес його адміністрування;

- інфраструктура використовується нерівномірно;

- закупівля, встановлення і налаштування нового обладнання потребує досить довгого часу, через що IT-відділ не може вчасно реагувати на потреби бізнесу;

- у випадку виникнення непередбачуваної відмови обладнання створюються суттєві проблеми для функціонування всієї інфраструктури.

Для вирішення такого роду проблем можна розглянути можливість впровадження у своїй діяльності приватної хмари.

2) Приватна хмара це захищена IT-інфраструктура, яка контролюється та експлуатується однією компанією або організацією без спільного використання фізичних чи віртуальних ресурсів з іншими користувачами. Така інфраструктура може функціонувати як на власному фізичному обладнанні компаніях, так і на орендованих обчислювальних ресурсах. Розміщення хмарної інфраструктури здійснюється або у власному ЦОД, або в дата-центрі постачальника послуг. Доступ до приватної хмари для вирішення бізнес-задач надається клієнтам через мережу Інтернет [15].

У результаті переходу від традиційної IT-інфраструктури до приватної хмари, компанія може отримати покращення наступних позицій [15]:

- зменшення витрат на фізичне обладнання, оскільки завдяки технологіям віртуалізації підвищується ефективність використання існуючих обчислювальних потужностей;

- спрощення розгортання інфраструктури компанії як у разі виникнення аварійних ситуацій, так і під час відкриття філій в інших регіонах;

- спрощення масштабування у випадку зміни робочих навантажень;

- швидке створення віртуальних серверів за вимогою;

- оскільки приватна хмара будується на базі ресурсів, що перебувають у власності компанії, то вона має повний контроль над архітектурою і усіма процесами, що забезпечуються у хмарній інфраструктурі.

Але треба зважувати на те, що обслуговування приватної хмари потребує значних витрат на всіх етапах її експлуатації, проте для великих компаній, підприємств або організацій це може бути оптимальним рішенням, оскільки спрощує процес роботи з інфраструктурою та гарантує захист даних. Також приватна хмара може не впоратися із навантаженням у періоди значних сплесків активності користувачів. Цей фактор залежить від можливостей апаратних ресурсів компанії. У результаті з’явилися гібридні хмари, які поєднують можливості приватних і публічних хмар.

3) Гібридна хмара призначена для розв’язання проблеми нестачі внутрішніх обчислювальних ресурсів компанії у разі використання приватної хмари шляхом перерозподілу частини задач на сервери сервіс-провайдера, тобто до публічної

хмари. Концепція гібридної хмари дає змогу об’єднати в єдиний хмарний простір внутрішню корпоративну хмару та зовнішню хмару провайдера. Завдяки такому підходу компанія може зберігати критично важливі та конфіденційні дані користувачів у власній приватній хмарі, тоді як у загальнодоступній частині можуть розміщуватися тестові середовища, середовища для розробки проектів, а також ті додатки, що використання яких не є регулярним [3, 13].

До недоліків моделі на основі гібридної хмари можна віднести її відносну новизну та складність реалізації. На сьогодні ще не сформовано стандартних хмарних структур, які б забезпечували тісну та безшовну взаємодію приватних і публічних хмар. У багатьох випадках упровадження гібридної моделі надмірно ускладнює інфраструктуру компанії, тому без нагальної потреби і розуміння можливостей хмарних технологій поєднувати приватну та публічну хмару не рекомендується. Але слід зазначити, якщо є розуміння принципів розгортання хмарних моделей і є свідомий вибір моделей обслуговування, що в них реалізуються, то компанія може суттєво підвищити ефективність своєї роботи, зменшити витрати на підтримку і закупівлю IT-обладнання, а також навіть скоротити штат своїх працівників.

Треба зазначити, що на практиці впровадження технологій хмарних обчислень далеко не завжди є простою задачею. Наприклад, у процесі побудови гібридної хмари з моделями послуг IaaS та PaaS компанії спочатку необхідно створити (розробити) власну приватну хмару. Для цього потрібно об’єднати всі наявні апаратні ресурси та організувати управління їх розподілом за допомогою спеціалізованого хмарного ПЗ. При цьому великі компанії можуть зіткнутися з проблемою застосування різних технологій віртуалізації різними підрозділами без централізованого узгодження здійснення їх впливу на загальну ефективність роботи корпоративної інфраструктури в цілому. Така ситуація може виникати з кількох причин:

- компанія свідомо закуповувала різні платформи віртуалізації, щоб уникнути залежності від одного постачальника;

- у компанії функціонує велика кількість відділів, які працюють незалежно один від одного, тому їх працівники обирали платформи, з якими мали найбільший досвід робити і компетенції;

- деякі платформи ще знаходилися на стадії тестування та не були повністю впроваджені в широку експлуатацію, але їх використання з експериментальною метою, можливо, надавало значні фінансові переваги компанії.

Але треба звернути увагу, що через природні обмеження фізичного обладнання на всі сервери компанії неможливо встановити одну і ту саму систему віртуалізації, тому цей процес має бути продуманим і ним потрібно постійно здійснювати управління. Крім того, перехід до хмарного управління ресурсами має відбуватися поступово, а перенесення вже VM до хмарного середовища має бути максимально прозорим і з мінімальною кількістю дій з боку адміністраторів [16].

Одним із можливих рішень цієї проблеми є використання такого хмарного ПЗ, яке здатне одночасно працювати з різними платформами віртуалізації, а також легко масштабувати кількість підтримуваних платформ, хмарних провайдерів і доступних керованих сервісів або послуг (managed-сервісів) [17]. При цьому користувачеві має надаватися єдина точка доступу та можливість самостійно обирати, з якої платформи віртуалізації , та що в середині компанії чи зовнішньої хмари, будуть надаватимуться хмарні сервіси. Managed-сервіси – це модель, згідно з якою компанія передає управління та підтримку своєї IT-інфраструктури (або її частини) на аутсорсинг сторонньому хмарному провайдеру [17]. Як можна бачити, що у визначенні цієї моделі як раз і закладений основний підхід розгортання корпоративної гібридної хмари, який аналізувався вище.

2 АНАЛІЗ ТА ПОРІВНЯННЯ КОМПЛЕКСНИХ РІШЕНЬ ПРОГРАМНИХ ПРОДУКТІВ ДЛЯ РОЗГОРТАННЯ І НАДАННЯ ХМАРНИХ СЕРВІСІВ

2.1 Платформа AOMEI Technology MultCloud

AOMEI Technology MultCloud – це комплексна програмна платформа для здійснення управляння кількома хмарними обліковими записами з єдиного інтерфейсу та з одним входом у систему. Її використання усуває потребу окремо здійснювати управління численними обліковими записами, постійно перемикатися між ними або відкривати кілька веб-сторінок для роботи. Перша версія цієї програмної платформи була представлена у 2013 році і на цей час її підтримує понад 30 провідних хмарних сервісів [18].

MultCloud дає змогу об’єднати всі хмарні сервіси в єдину платформу та здійснювати пряму передачу даних між ними. Основний функціонал цієї платформи наступний [18]:

- забезпечення прямої передачі даних між сховищами;
- синхронізація даних між хмарними сховищами;
- централізований доступ до сервісів з одного місця;
- безпосереднє завантаження файлів з комп’ютера користувача або з використанням URL-адрес;
- підтримка протоколів для роботи з файлами FTP/SFTP/WEB.Dav;
- здійснення резервного копіювання БД MySQL до хмарних сховищ;
- попередній перегляд і пошук файлів у всіх підключених хмарних сховищах.

Водночас це ПЗ також має низку недоліків у контексті побудови комплексної програмної платформи гібридної хмари [18]:

- можливість працювати лише зі сховищами даних без підтримки створення інших типів хмарних об’єктів і елементів;
- відсутність підтримки платформ віртуалізації, оскільки система взаємодіє лише з хмарними провайдерами;
- відсутність інструментарію для створення керованих сервісів (managed-сервісів) з метою їх подальшого надання за моделлю PaaS;
- підтримується лише web-взаємодія;
- відсутні інтерфейси API.

2.2 Платформа VMware vSphere

VMware vSphere – це програмна платформа віртуалізації корпоративного рівня, яка призначена для віртуалізації серверів і управління хмарною інфраструктурою. Основною метою продукту є створення інфраструктури з високим рівнем відмовостійкості, спрощеним адмініструванням і відповідністю сучасним технологічним стандартам. Платформа дає змогу запускати велику кількість VM на одному фізичному сервері, об’єднувати сотні серверів у єдину віртуальну мережну інфраструктуру та реалізувати механізми здійснення автоматизованого розподілу обчислювальних ресурсів. До цього комплексного ПЗ входить багато програмних продуктів, серед яких найбільш відомими є [19]:

- VMware ESXi Server – гіпервізор (монітор VM), що встановлюється безпосередньо на фізичний сервер, завдяки чому забезпечує прямий доступ до ресурсів цього серверу і потрібний контроль над ними;

- VMware vCenter Server – система централізованого управління та моніторингу VM, що створюється і функціонує на гіпервізорах. Дає змогу адмініструвати всі сервери VMware ESXi з єдиної консолі. У разі організації управління великою інфраструктурою дозволяє застосувати механізми автоматизації рутинних операцій та задач, що часто повторюються.

З розвитком застосувань хмарних обчислень і сервісів, що надаються на їх платформах, компанія VMware розробила низку нових продуктів, які можуть стати доступними за окремими ліцензіями та дають змогу суттєво розширити функціональні можливості платформи віртуалізації [19]:

- VMware Tanzu – комплексна платформа, що містить мінімально необхідний набір компонентів для здійснення запуску кластерів і контейнерів Kubernetes (K8s), а також надає засоби їх моніторингу та управління усім їх життєвим циклом;

- vSphere Integrated Container Engine – являє собою середовище розгортання контейнерів для vSphere та дає змогу розробникам, що працюють із платформою контейнеризації Docker, створювати контейнери та розгортати їх поряд, і використовуючи для цього традиційні робочі навантаження на основі VM у кластерах vSphere, а також дозволяє здійснювати управління цими робочими навантаженнями через інтерфейс користувача vSphere;

- VMware Horizon – платформа для запуску і управління віртуальних робочих столів (Virtual Desktop Infrastructure, VDI), додатків і сервісів на віддаленому сервері під управлінням засобів віртуалізації VMware.

Попри широкі функціональні можливості, рішення на основі VMware Vsphere має певні недоліки у аспекті побудови комплексної програмної платформи гібридної хмари [19]:

- відсутня можливість одночасного використання сторонніх платформ віртуалізації для реалізації функціоналу моделей обслуговування IaaS та PaaS;

- перелік доступних керованих сервісів (managed-сервісів) і послуг визначається безпосередньо компанією VMware;

- використання низки продуктів, що розширює можливості віртуалізації платформи VMwar, потребує придбання додаткових ліцензій.

2.3 Платформа Nutanix Enterprise Cloud Platform

Nutanix Enterprise Cloud Platform – це програмно-визначена гіперконвергентна інфраструктура від компанії Nutanix, яка дозволяє об’єднати обчислювальні ресурси, ПЗ, системи зберігання даних та мережі в єдину платформу. Вона дозволяє створювати приватні та гібридні хмари для реалізації управління будь-якими додатками. Іншими словами – ця платформа дає змогу працювати з будь-якою кількістю додатків, будувати розподілені сховища даних і обчислювальні кластери, а також здійснювати управління масивами даних. Базовими складовими платформи є [20]:

- Nutanix Acropolis OS – операційна система (ОС) Nutanix для здійснення управління хмарною інфраструктурою;

- Nutanix Prism – сервіс для організації звітності, адміністрування та здійснення управління віртуалізованими середовищами Nutanix. Дає змогу забезпечити процеси управління кластерами, VM, сховищами, сервісами та послугами протягом їх повного життєвого циклу. Його складовими є Prism Element – дозволяє реалізувати управління одним кластером, і Prism Central – дозволяє реалізувати управління кількома кластерами.

Платформа Nutanix працює на гіпервізорі власної розробки, що називається AHV (Acropolis Hypervisor). Він створений на базі open-source рішення KVM (Kernel-based Virtual Machine – віртуальна машина на основі ядра) і, відповідно,

являє собою ПЗ для віртуалізації на платформі Linux, що функціонує на різних ОС. Це ПЗ управляє та розподіляє ресурси між всіма підключеними фізичними серверами, на яких можуть стояти різні ОС, що дозволяє запускати їх одночасно. Також у Nutanix реалізована підтримка інтеграції зі сторонніми гіпервізорами, такими як VMware ESXi та Microsoft Hyper-V [20].

В якості керованих (managed-) сервісів, що надаються поверх платформ віртуалізації Nutanix, компанія пропонує наступні рішення [20]:

- Nutanix Era (Database Service) – сервіс для створення БД з можливістю подальшого управління їх усім життєвим циклом. Дає змогу одночасно працювати з декількома БД, такими як, наприклад, Oracle, Microsoft SQL Server, MySQL, MariaDB, PostgreSQL та SAP HANA;

- Nutanix Frame – сервіс для організації та здійснення управління VDI;

- Nutanix Karbon (Kubernetes Engine) – сервіс для реалізації управління повним життєвим циклом кластерів K8s. Він спрощує процедури їх розгортання, подальшого обслуговування, а також за потреби масштабування та оновлення.

Недоліками Nutanix Enterprise Cloud Platform в аспекті побудови комплексної програмної платформи гібридної хмари [20]:

- немає можливості залучити сторонні хмарні платформи для виділення ресурсів;

- платформою підтримується дуже обмежена кількість сторонніх платформ віртуалізації;

- застосовується обмежена кількість керованих (managed-) сервісів.

2.4 Платформа OpenNebula

OpenNebula – це комплексне ПЗ з відкритим вихідним кодом для створення та управління хмарами і віртуалізованими ЦОД. Вона призначена для побудови приватних, гібридних та периферійних (edge) хмар та надає уніфіковане управління інфраструктурою та додатками незалежно від постачальників. Це рішення поєднує технології віртуалізації та контейнеризації з можливістю одночасного автоматизованого і еластичного надання багатьох веб-додатків і веб-сервісів у приватних і гібридних середовищах. Її архітектура складається з таких основних сервісних компонентів [21]:

- OpenNebula Daemon – це базовий сервіс здійснення управління хмарною інфраструктурою. Він забезпечує управління вузлами кластерів, віртуальними мережами та сховищами даних, групами користувачів та окремими користувачами і їх VM;

- БД OpenNebula – сервіс, що зберігає поточний стан хмарної інфраструктури;

- Scheduler – сервіс, що відповідає за планування запуску VM на доступних вузлах гіпервізора;

- Edge Cluster Provision – забезпечує створення повнофункціональних кластерів OpenNebula у публічних хмарах або на периферійних платформах провайдерів;

- Monitoring – сервіс, який функціонує як частина базового сервісу OpenNebula Daemon. Реалізує моніторинг і збір інформації про фізичні сервери та VM, що на них розміщені.

Особливостями платформи OpenNebula є широкий функціонал та простота експлуатації. Вона підтримує управління як контейнерами, так і повноцінними VM, надає змогу будувати гібридні хмари шляхом розширення приватної хмарної інфраструктури за допомогою сторонніх хмарних провайдерів з відкритою архітектурою, зокрема AWS (Amazon Web Services). Також передбачена можливість інтеграції кількох платформ віртуалізації (зокрема, VMware ESXi та KVM) для повністю віртуалізованих хмар, а також контейнерних систем LXC (LinuX Containers), що дозволяла запускати декілька ізольованих Linux-систем (контейнерів) на одному сервері, і мікросервісних Firecrackerx-систем для можливості розгортання безсерверних рішень [21, 22].

У разі необхідності використання платформи контейнеризації Kubernetes, то платформа OpenNebula надає це зробити за допомогою віртуального пристрою, що є сертифікованим міжнародним некомерційним фондом для розвитку технологій хмарних обчислень та контейнеризації CNCF (Cloud Native Computing Foundation) [23].

В якості комплексної програмної платформи OpenNebula пропонує модель хмарного забезпечення, що ґрунтується на концепції віртуальних ЦОД. Ця модель дає змогу інтегрованій комплексній платформі динамічно надавати ресурси інфраструктури у великих середовищах з декількома ЦОД та декількома хмарами різним користувачам, групам користувачів чи, навіть, бізнес-компаніям. Наприклад, локальні приватні хмари можуть реалізувати обслуговування кількох

проєктів, відділів або організацій у межах однієї компанії, тоді як хмарні провайдери забезпечують організацію ізольованих хмарних середовищ з можливістю окремого адміністрування користувачів і всього спектру обчислювальних ресурсів у приватній хмарі [24].

Але, попри значні функціональні можливості, підхід, що реалізований в OpenNebula, має також недоліки в аспекті побудови комплексної програмної платформи гібридної хмари:

- відсутні можливості щодо забезпечення розширення кількості доступних керованих (managed-) сервісів;
- реалізується підтримка обмеженої кількості публічних хмар;
- реалізується підтримка обмеженої кількості платформ віртуалізації сторонніх провайдерів.

2.5 Платформа OpenStack

Платформа OpenStack – це відкритий комплексний програмний продукт (або комплекс ПЗ), який надає функціонал хмарної платформи та дає змогу розгортати інфраструктуру за моделлю IaaS у публічних і приватних хмарних структурах. Архітектура OpenStack реалізована за модульним принципом, що забезпечує еластичне надання набору базових сервісів, обчислювальних та мережних ресурсів, ресурсів зберігання даних в масштабовану інфраструктуру, якою можна здійснювати управління через веб-інтерфейс, API або командний рядок. Модульний підхід у реалізації архітектури OpenStack забезпечує підключення не тільки основних базових компонентів, тут також користувач може підключати додаткові сервіси для виконання інших задач, наприклад, такі сервіси як: резервного копіювання, роботи з контейнерами, балансування навантаження на серверах, тощо. Також для цієї платформи створено велику кількість плагінів і інтеграційних сервісів із стороннім ПЗ, зокрема такі відомі як: Terraform OpenStack Provider, Ansible та Packer. Таким чином, для користувачів платформа OpenStack пропонує широкий перелік рішень, який охоплює практично всі аспекти роботи в хмарному середовищі [25, 26].

Комплексне ПЗ OpenStack складається з широкого переліку підсистем, частина з яких можуть використовуватися як окремі сервіси, так і комбіновано у складі інших систем. Серед них в якості основних можна виділити наступні [25, 26]:

- OpenStack Compute (Nova) – забезпечує управління окремими VM шляхом взаємодії з гіпервізором;

- OpenStack Networking (Neutron) – реалізує мережну взаємодію та зв’язність;

- OpenStack Keystone (Identity) – виконує функції централізованої системи автентифікації платформи OpenStack;

- OpenStack Image Service (Glance) – зберігає образи VM, які використовуються користувачами для їх запуску;

- OpenStack Block Storage (Cinder) – забезпечує управління сховищем даних, що використовується для VM;

- OpenStack Dashboard (Horizon) – веб-інтерфейс, що забезпечує управління інфраструктурою комплексної програмної платформи OpenStack.

Треба зазначити, що програмна платформа OpenStack не є гіпервізором, проте підтримує інтеграцію зі сторонніми гіпервізорами, такими як KVM, Xen, QEMU, Hyper-V і VMware vSphere (але пряма підтримка гіпервізора VMware ESXi є відсутньою). Крім того, цю платформу можна розширювати додатковими модулями для роботи з кластерами Kubernetes, БД та іншими managed-сервісами.

Як і в попередніх випадках, при зазначеній широкій функціональності, цій платформі також притаманні і недоліки, в аспекті розгортання комплексної програмної платформи гібридної хмари [25]:

- відсутня можливість використання наявних найбільш популярних хмарних платформ (AWS, Windows Agur, та інших) для побудови власної гібридної хмари;

- проєкт, що є створеним у інфраструктурі OpenStack, характеризується високим рівнем складності впровадження та адміністрування.

2.6 Порівняння комплексних програмних платформ в аспекті розгортання гібридної хмари

Таким чином, із вище проведеного розгляду наявних на цей час рішень комплексного ПЗ для створення і розгортання хмарної інфраструктури можна бачити, що у них всіх є недоліки і навіть досить критичні, що не дозволяють ці рішення використовувати в якості платформи для розгортання повноцінної хмари, яка б забезпечила роботу з різними типами віртуалізації одночасно, а також надала б можливості простого розширення кількості доступних керованих сервісів і послуг (так званих managed-сервісів). Також можна бачити, що

розглянуті платформи не можуть підтримувати достатню кількість платформ віртуалізації і хмарних технологій. Крім того, вони мають невеликий перелік керованих (managed-) сервисов, що ними підтримується.

У процесі здійснення порівняння наведених вище комплексних програмних платформ, а також джерел, де вони описуються [3, 18 - 26], були отримані такі результати, які зведені до таблиці 2.1.

Таблиця 2.1 – Результати порівняння проаналізованих платформ комплексного ПЗ для організації хмарних інфраструктур

Функціонал	MultCloud	VMware	Nutanix	Open Nebula	Open Stack	Мета, якої треба досягти
Здатність простого расширения кількості managed-сервисов, що підтримуються платформою	–	–	–	±	±	+
Потенційно необмежена кількість хмарних платформ, що підтримують надання обчислювальних ресурсів	–	–	–	±	±	+
Підтримка зовнішніх хмарних провайдерів	+	–	–	+	–	+
Підтримка віртуалізаторів сторонніх розробників	–	–	–	+	+	+
Наявність зовнішнього інтерфейсу для інтеграції	–	+	+	+	+	+

Із отриманих результатів, що наведені в табл. 2.1, можна бачити, що програмна платформа OpenStack найбільше підходить для розгортання хмарної інфраструктури з усіх тих, що були вище розглянуті. Але треба звернути увагу, що навіть OpenStack повністю не задовольняє всім необхідним вимогам. Тобто, для вирішення поставлених задач потрібне створення власної комплексної програмної платформи розгортання хмарної інфраструктури та надання хмарних сервісів, для якої в якості базової основи будемо спиратися на ПЗ OpenStack.

3 РОЗРОБКА АРХІТЕКТУРИ КОМПЛЕКСНОЇ ПРОГРАМНОЇ ПЛАТФОРМИ ДЛЯ СТВОРЕННЯ ХМАРНОЇ ІНФРАСТРУКТУРИ НА ОСНОВІ OPENSTACK

3.1 Спрощена функціональна архітектура OpenStack

Як зазначалося, платформа OpenStack організована за модульною архітектурою, до складу якої входить набір взаємопов'язаних сервісних модулів (сервісів). Частина цих сервісів є базовою, тоді як інші можуть встановлюватися лише за необхідності залежно від функціональних вимог хмарної інфраструктури. Такий підхід забезпечує високу масштабованість, гнучкість та еластичність платформи, дозволяючи адаптувати її конфігурацію до конкретних умов експлуатації. Основні сервісні модулі OpenStack були розглянуті у підрозділі 2.5 цієї пояснювальної записки, де також стисло наведене їх призначення. Взаємодія між окремими сервісами OpenStack реалізується за допомогою системи обміну повідомленнями, побудованої на основі відкритого розширеного протоколу черг повідомлень (Advanced Message Queuing Protocol, AMQP). Цей протокол призначений для надійного асинхронного обміну повідомленнями між програмними компонентами розподілених інформаційних систем. Завдяки використанню AMQP окремі модулі OpenStack можуть взаємодіяти між собою незалежно один від одного, що підвищує продуктивність, відмовостійкість та спрощує масштабування всієї платформи.

Кожний сервісний модуль OpenStack також має власний набір API, які забезпечують стандартизовану взаємодію між окремими компонентами платформи, а також надають можливість інтеграції OpenStack зі сторонніми програмними продуктами, засобами автоматизації, системами оркестрації та інструментами управління хмарною інфраструктурою [27].

В якості серверів для організації зв'язку та обміну даними між різними сервісами архітектури найчастіше використовуються програмні інструменті RabbitMQ, Apache Qpid або ZeroMQ. Вони забезпечують приймання, маршрутизацію, буферизацію та доставку службових повідомлень між сервісними модулями платформи, координуючи виконання їх спільних функцій [27].

Спрощена базова функціональна архітектура OpenStack наведена на рис. 3.1. Наведена архітектура відображає взаємодію основних сервісних модулів платформи у процесі створення, розгортання та управління хмарною

інфраструктурою. Взаємодія всіх сервісів між собою у процесі надання хмарних послуг здійснюється через стандартизовані API та в цілому забезпечують функціонування хмарного середовища [27].

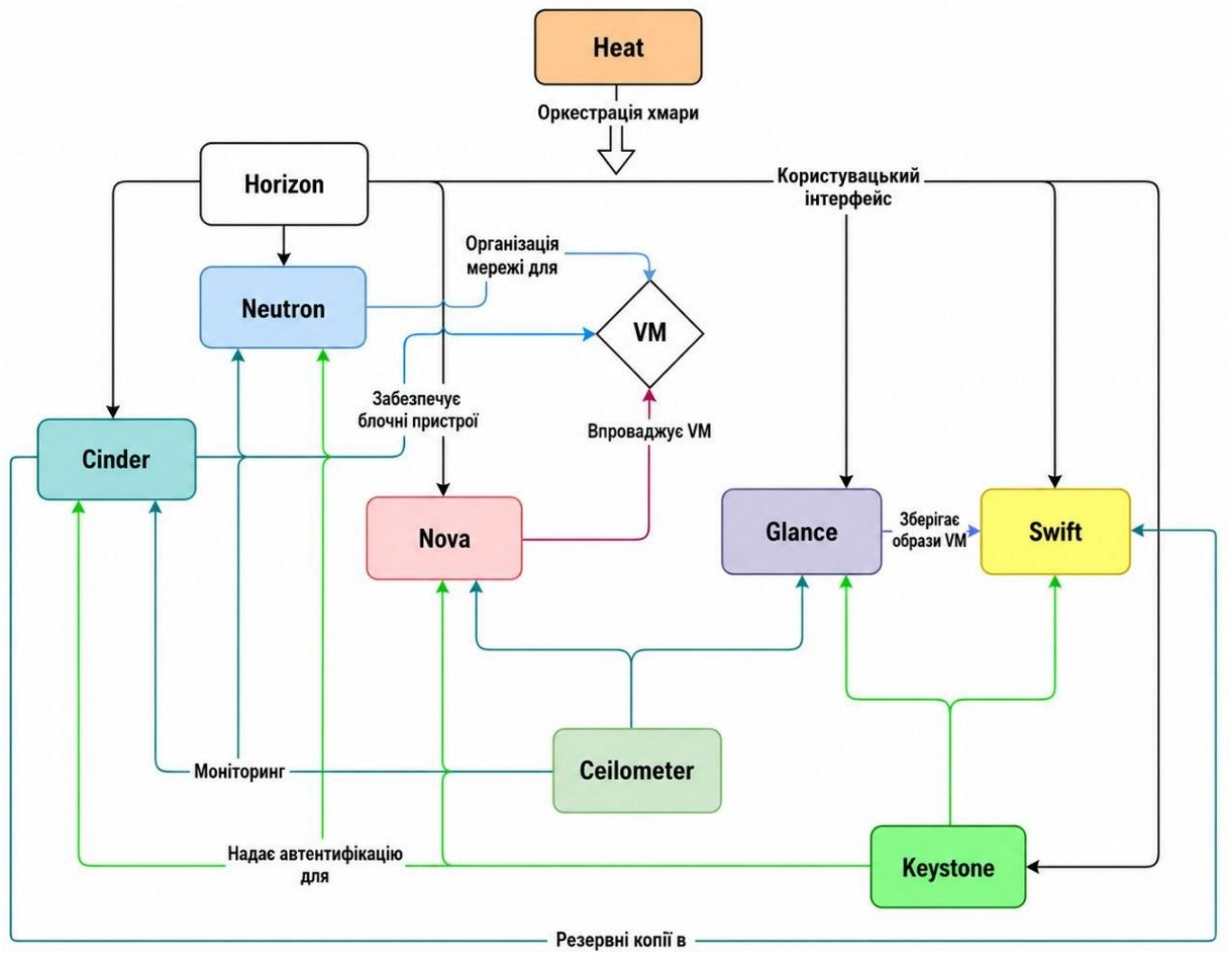


Рисунок 3.1 – Спрощена базова функціональна архітектура OpenStack

Взаємодія користувача із хмарною платформою здійснюється за допомогою сервісного модуля Horizon, який реалізує веб-інтерфейс управління OpenStack. Через Horizon користувач отримує доступ до функцій створення та адміністрування віртуальної інфраструктури, а всі сформовані запити передаються відповідним сервісним модулям. Зокрема, Horizon взаємодіє із сервісами Nova, Neutron, Glance, Swift, Cinder та Heat, забезпечуючи централізоване управління всією хмарною платформою.

Автоматизація розгортання хмарної інфраструктури виконується сервісним модулем Heat, який реалізує функції оркестрації (управління) хмари. На схемі

показано, що Heat працює через інтерфейс користувача для сервісу Horizon і забезпечує автоматичне створення взаємопов'язаних ресурсів хмарної інфраструктури відповідно до заданих шаблонів, координуючи запуск VM, віртуальних мереж, сховищ та інших компонентів.

Основним модулем управління обчислювальними ресурсами є Nova. Саме цей сервіс відповідає за створення, запуск, зупинку, міграцію та видалення VM. Як показано на схемі, Nova взаємодіє з модулем Glance, звідки отримує образи ОС для розгортання VM. Після цього Nova передає команди на створення екземпляра VM та забезпечує її запуск. Крім того, Nova взаємодіє із сервісом Neutron, який надає мережні ресурси для VM, що створюються. Також із сервісом Cinder, який забезпечує підключення зовнішніх блокових сховищ даних.

Організація мережної взаємодії покладається на сервісний модуль Neutron. Він забезпечує створення віртуальних мереж, маршрутизаторів, підмереж, політик безпеки та інших мережних ресурсів. На схемі показано, що саме Neutron організовує мережне підключення для VM та взаємодіє із сервісом Nova у процесі її створення. Одночасно Neutron забезпечує підключення блокових сховищ, які надаються сервісом Cinder, до початку створених екземплярів VM.

Для роботи з образами ОС застосовується сервісний модуль Glance. Він забезпечує централізоване зберігання, каталогізацію та надання образів VM іншим компонентам OpenStack. На схемі показано, що Glance передає необхідні образи сервісу Nova для подальшого створення VM. Крім цього, Glance взаємодіє із сервісом Swift, у якому можуть зберігатися образи VM, що мають великий розмір.

Сервіс Swift реалізує об'єктне сховище даних OpenStack. Його основне призначення полягає у довготривалому зберіганні образів, резервних копій, архівів і великих обсягів неструктурованих даних. Як показано на схемі, Swift взаємодіє з модулем Glance у процесі зберігання образів VM, а також із сервісом Keystone, який контролює доступ користувачів до об'єктного сховища.

Для роботи з блоковими сховищами даних використовується сервіс Cinder. Блокове сховище – це спосіб зберігання інформації, за якого дані записуються і зчитуються окремими блоками фіксованого розміру, а ОС сприймає таке сховище як звичайний жорсткий диск. Сервіс Cinder забезпечує створення, підключення та управління постійними блоковими томами, які використовуються VM незалежно від їхнього життєвого циклу. На схемі показано, що Cinder взаємодіє із сервісом

Nova, забезпечуючи VM додатковими сховищами, а також підтримує взаємодію з Horizon для адміністрування ресурсів.

Єдиним центром автентифікації та авторизації в наведеній архітектурі OpenStack є сервісний модуль Keystone. Він перевіряє облікові дані користувачів, видає маркери доступу (токени) та забезпечує контроль прав доступу до всіх сервісів платформи. Keystone взаємодіє практично з усіма основними сервісами OpenStack, забезпечуючи безпечний доступ до ресурсів хмарної інфраструктури.

Моніторинг стану компонентів платформи реалізується сервісним модулем Ceilometer. Він збирає інформацію про використання процесорних ресурсів, оперативної пам'яті, дискового простору, мережного трафіку та інших параметрів функціонування сервісів OpenStack. На схемі видно, що Ceilometer взаємодіє з модулями Nova, Glance, Cinder та Keystone, отримуючи від них телеметричну інформацію, яка використовується для моніторингу, аналізу навантаження та обліку використання ресурсів.

Хоча програмна платформа OpenStack і не відповідає всім вимогам різних компаній, проте вона має дуже розвинений API-інтерфейс. Треба зазначити, що API тут має кожний сервісний модуль. Вони забезпечують стандартизовану взаємодію між окремими компонентами платформи, а також надають можливість інтеграції OpenStack зі сторонніми програмними продуктами, засобами автоматизації, системами оркестрації та інструментами управління хмарною інфраструктурою. Крім того, для платформи OpenStack розроблена значна кількість програмних засобів, які спрощують автоматизацію розгортання, конфігурування та подальше адміністрування хмарної інфраструктури. До найбільш поширених із них належать [27]:

- OpenStack Terraform Provider – інструментарій, який дозволяє автоматизувати управління інфраструктурою OpenStack шляхом опису її бажаного стану у вигляді програмного коду (Infrastructure as Code, IaC);

- OpenStack-Ansible – надає ролі і сценарії автоматизації, призначений для розгортання, налаштування та супроводу компонентів хмарної платформи OpenStack;

- OpenStack-Packer – інструментарій для автоматизованого створення образів VM та їх подальшого завантаження до платформи OpenStack за допомогою інтерфейсу OpenStack API.

Треба зазначити, що використання пропрієтарного OpenStack API під час подальшої розробки комплексної програмної платформи потребувало б створення окремих модулів інтеграції для більшості сучасних засобів автоматизації та управління хмарною інфраструктурою. Тому для спрощення та забезпечення сумісності з існуючими програмними рішеннями доцільним є використання OpenStack-сумісного API, що дозволяє інтегрувати розроблений програмний комплекс із наявними інструментами автоматизації та адміністрування без необхідності їх додаткової модифікації.

3.2 Обґрунтування та визначення архітектурного рішення комплексної програмної платформи для розгортання хмарних обчислень

Перш за все, при проведенні подальших розробок будемо дотримуватися мікросервісного підходу. Це пов’язано з тим, що побудований за модульним принципом, який є одним із основним підходом до розробки сучасного ПЗ великої складності. Як видно із наведеної вище архітектури OpenStack, та як було обґрунтовано – модульна архітектура дозволяє розділити програмний комплекс на окремі функціональні компоненти (сервіси або сервісні модулі), кожен із яких може розроблятися, тестуватися та супроводжуватися незалежно від інших. Такий підхід спрощує розвиток програмного продукту, покращує його супроводжуваність і дозволяє залучати до розробки окремих модулів різні команди фахівців. Різновидом модульної архітектури є мікросервісна архітектура (МСА), у якій мінімальною функціональною одиницею виступає окремий сервіс, що має власний API. Кожний такий мікросервіс може незалежно розгортатися, оновлюватися та масштабуватися без необхідності внесення змін до інших компонентів програмного комплексу [27].

Однією з головних особливостей МСА є слабкий рівень зв'язності між окремими сервісами. Їх взаємодія здійснюється виключно через стандартизовані API, що забезпечує незалежність окремих компонентів системи. Для зменшення взаємозалежності кожний мікросервіс, як правило, використовує власну БД або власне сховище даних, що дозволяє ізолювати дані та спростити супровід окремих сервісів.

Використання МСА забезпечує низку важливих переваг. Зокрема, вона дозволяє реалізувати процеси CI/CD, спрощує супровід окремих сервісів,

забезпечує їх незалежне розгортання та масштабування, створює умови для автономної роботи команд розробників, полегшує впровадження нових технологій і спрощує локалізацію та усунення програмних помилок.

Разом із цим мікросервісний підхід має також і певні недоліки. Зокрема До них належать ускладнення тестування окремих сервісів через їх взаємодію з іншими компонентами системи, складність реалізації функціоналу, який одночасно охоплює кілька сервісів, а також необхідність визначення оптимальної кількості мікросервісів під час проєктування комплексного ПЗ [27].

Таким чином для проведення розробок будемо застосовувати мікросервісний підхід. При цьому розподіл функціональних обов'язків між окремими сервісами здійснюється за принципом, аналогічним модульній архітектурі платформи OpenStack, що забезпечує незалежність окремих компонентів, спрощує їх масштабування та подальший розвиток програмної платформи.

Також для подальшої розробки комплексної програмної платформи хмарних обчислень необхідно визначити склад основних сервісних модулів, які реалізовуватимуть її функціональні можливості. Основна логіка дій програмного комплексу зосереджується в сервісах, що забезпечують управління базовими сутностями моделі IaaS. До таких сутностей належать VM, образи, блокові томи, мережі, проєкти та інші хмарні ресурси. Зокрема до складу програмної платформи доцільно включити такі сервіси:

- Identity service –авторизації користувачів;
- Images service – образів серверів;
- Block storage service – керування блоковими сховищами;
- Networking service – керування мережною інфраструктурою;
- Compute service – керування VM;
- Magnum service – керування кластерами Kubernetes;
- Trove service –керування кластерами БД (DB кластерами).

Оскільки система, що розроблюється, ґрунтується на мікросервісній архітектурі, то це передбачає мінімальну зв'язність між сервісами, кожен із них керує лише власними сутностями і не має інформації про використання цих сутностей іншими сервісами. Такий підхід мінімізує зв'язність між сервісами, однак створює проблему контролю взаємозалежностей між різними ресурсами. Наприклад, при створенні віртуальної машини Compute Service буде

використовувати образ, який належить Images Service. Якщо інформація про цей зв'язок не зберігатиметься централізовано, виникає ризик видалення образу, що вже використовується функціонуючими VM.

Для усунення подібних конфліктів доцільно реалізувати окремий сервіс зв’язків абстрактних сутностей (Relations Service), основним призначенням якого буде створення, зберігання та надання інформації про зв'язки і залежності між будь-якими сутностями комплексної програмної платформи. Цей сервіс має підтримувати довільні типи ресурсів, що забезпечить його універсальність і можливість використання всіма сервісними модулями платформи.

Однією з основних вимог до архітектури програмної платформи є підтримка як можна більшої кількості платформ віртуалізації (в ідеалі – необмеженої кількості), що мають API або бібліотеки для взаємодії. Незалежно від конкретної реалізації, усі платформи містять схожий набір характеристик, зокрема таких як назва, зони доступності, мережні параметри, типи ресурсів тощо. Для централізованого зберігання цієї інформації доцільно в архітектурі реалізувати сервіс Platform-Datastore, який буде здійснювати взаємодію з платформами віртуалізації, що нею підтримуються, зокрема, зчитуватиме і кешуватиме їх загальну інформацію. Це необхідно для того, щоб інші сервісні модулі були в змозі отримувати інформацію про платформи віртуалізації, що підтримуються, з одного джерела. Такий підхід дозволить уникнути дублювання програмного коду та забезпечить використання єдиного джерела інформації про всі підключені платформи.

Ще одним необхідним компонентом є сервіс автоматичної реєстрації доменних імен Go-DNS-Tsig. Його призначення полягає у створенні та оновленні DNS-записів для нових об'єктів хмарної інфраструктури незалежно від того, який сервісний модуль забезпечував їх створення.

Оскільки різні платформи віртуалізації використовують свої власні API, свої формати запитів і механізми виконання операцій, то для взаємодії з ними доцільно застосувати підходи, що ґрунтуються на здійсненні обміну подіями. У цьому випадку сервісні модулі формують події створення або зміни ресурсів та передають їх через чергу повідомлень у відповідні тематичні канали (topics). Мікросервіси конкретних платформ віртуалізації обробляють ці події, виконують необхідні операції та повертають інформацію про їх виконання. Надалі виконання таких операцій може контролюватися шляхом відстеження стану відповідної операції або при отриманні повідомлення про її завершення. Такий підхід

дозволяє незалежно змінювати кількість обробників повідомлень з боку мікросервісів, що підтримують платформи віртуалізації, в залежності від навантаження без зміни логіки дій у системі в цілому. Це можливо тому, що сховище завдань є загальним для всіх діючих обробників повідомлень.

Винятком є операції, які пов'язані з передачею великих обсягів інформації, наприклад, завантаження образів ОС, розмір яких може досягати десятків гігабайтів. Для таких задач використання черг повідомлень є неефективним, тому сервіс створення образів у платформах віртуалізації працюватиме за протоколом віддаленого виклику процедур gRPC (Google Remote Procedure Call, gRPC), вд компанії Google, який є сучасним високопродуктивним фреймворком з відкритим вихідним кодом, та який в нашій архітектурі буде забезпечувати високопродуктивний прямий обмін даними між компонентами системи.

З урахуванням вже прийнятих допущень до створення архітектури комплексної програмної платформи для кожної платформи віртуалізації, що буде нею підтримуватися, необхідно реалізувати окремі мікросервіси, які забезпечуватимуть процеси створення певних ресурсів (сутностей). На першому етапі розробки в нашій архітектурі передбачається підтримка платформ віртуалізації VMware, Nutanix та деякої абстактної хмарної платформи (назвемо її YCloud). Для кожної з цихплатформ передбачемо реалізацію таких сервісних модулів:

- Images [Platform] Provider – для створення образів серверів у зазначених платформах віртуалізації (VMware, Nutanix, YCloud);
- Compute [Platform] Provider – для створення віртуальних серверів (на основі створення VM) у зазначених платформах віртуалізації;
- Volumes [Platform] Provider – для створення блокових сховищ;
- Networks [Platform] Provider – для створення віртуальної мережі.

Треба зазначити, що для кожної платформи віртуалізації необхідно створити та впровадити весь набір сервісів. Сутність реалізації для цього окремих мікросервісів полягає в тому, що сервісні модулі (Compute, Images, Networking) не працюють напряму з API, наприклад, VMware чи Nutanix, звертаються до відповідного сервісного модулю Platform Provider, а той уже здійснює взаємодію з API конкретної платформи віртуалізації.

Взаємодія між основними сервісними модулями комплексної програмної платформи, як вище зазначалося, має реалізовуватися за допомогою протоколу gRPC, що обумовлено його високою продуктивністю завдяки використанню

протоколу HTTP/2, автоматичною генерацією клієнтських бібліотек на основі Protocol Buffers (Protobuf), стандартизованим форматом повідомлень та підтримкою паралельної обробки великої кількості запитів.

Зокрема перевагою цього є використання протоколу HTTP/2, який є основною версією мережного протоколу для ПД у мережі Інтернет, що прийшла на заміну застарілому HTTP/1.1. Його базова мета – прискорення завантаження веб-сайтів і безпечніша ПД. Відповідно, Protocol Buffers – це програмні інтерфейси що розроблені компанією Google. Вони перетворюють схеми .proto (текстові файли з розширенням .proto, які містять схеми даних) на готовий код для взаємодії мікросервісів або API-клієнтів, забезпечуючи при цьому високу швидкість і малий розмір.

Оскільки платформа OpenStack використовує для взаємодії із зовнішніми клієнтами архітектуру REST API, тому для забезпечення сумісності комплексної програмної платформи з OpenStack передбачається реалізація сервісу OpenStack API Gateway. Його функцією буде перетворення REST-запитів, сумісних із OpenStack API, у внутрішні виклики сервісних модулів за протоколом gRPC. В нашій архітектурі цей сервіс буде поданий як openstack-api-gateway.

Крім того, для взаємодії із веб-сайтиами в архітектурі, що розроблюється передбачається окремий сервіс iaas-api-gateway, який реалізуватиме REST-інтерфейс без необхідності повного відтворення формату відповідей OpenStack. Такий підхід спрощує взаємодію із власним інтерфейсом користувача та іншими зовнішніми інформаційними системами.

Централізоване зберігання облікових записів користувачів в сучасних корпоративних інформаційних системах, до якої відноситься і наша програмна платформа, здійснюється за допомогою спрощеного протоколу доступу до служб каталогів (Lightweight Directory Access Protocol, LDAP). Автентифікацію цих користувачів із області корпоративного каталогу LDAP в архітектурі, що розроблюється, передбачається реалізувати за допомогою системи Keycloak, інтеграція з якою виконуватиметься через Identity Service, через здійснення їх авторизації.

Для створення та управління managed-сервісами, такими як кластери Kubernetes (k8s) і БД, планується використання відкритої платформи AWX, яка реалізує веб-інтерфейс користувача, REST API та механізм вирішення задач щодо автоматизації на основі Ansible.

В якості БД для зберігання інформації комплексної програмної платформи передбачається використання Percona Server, що є розширеною реалізацією реляційної системи управління базами даних на платформі MySQL. Вибір саме цієї БД обумовлений її поширеним використанням на підприємстваї і організаціях.

Таким чином, орієнтуючись на вищепроведене обґрунтування побудови архітектури комплексної програмної платформи для розгортання та надання хмарних послуг, та на описаний взаємозв'язок усіх передбачених для неї сервісних модулів, можно навести її варіант реалізації, що представлений у додатку А.

3.3 Особливості та сутності основних сервісних модулів розробленої архітектури комплексної програмної платформи.

3.3.1 Сервіс авторизації – Identity

Identity service – це розробленій архітектурі комплексної програмної платформи той сервіс, який відповідає за централізовану систему авторизації користувачів. Його основним призначенням є перевірка автентичності користувачів під час доступу до функцій платформи через API, а також управління їх правами доступу до ресурсів хмарної інфраструктури. Identity service також забезпечує механізм виявлення інших хмарних сервісів платформи. Для цього він підтримує каталог сервісів і відповідних endpoints (кінцевих точок доступу), що дозволяє клієнтським додаткам автоматично отримувати інформацію про доступні сервіси та взаємодіяти з ними, маючи початковий доступ лише до Identity service. Сервіс підтримує як локальні облікові записи користувачів, так і механізм централізованої автентифікації, який дозволяє використовувати єдиний обліковий запис для доступу до кількох взаємопов'язаних сервісів, що реалізується на основі Keycloak. Keycloak перевіряє користувача хмарної інфраструктури та підтверджує його особу. Після цього користувач отримує доступ до всіх дозволених сервісів хмарної платформи. Такий механізм централізованої автентифікації має назву федеративної системи керування ідентифікацією.

У процесі роботи Identity service взаємодіє із Relations service, який використовується для перевірки наявності пов'язаних сутностей у разі необхідності видалення проєкту. Іншими словами, це унеможливлює видалення проєкту, якщо в ньому ще існують ресурси чи об’єкти, що створені іншими сервісами. Для функціонування Identity service необхідна власна БД, а у разі використання федеративної автентифікації, також потрібне підключення до сервера Keycloak.

До основних сутностей Identity service відносяться наступні:

- projects (endpoints: /v3/projects);
- users (endpoints: /v3/users);
- roles (endpoints: /v3/roles);
- services (endpoints: /v3/services);
- endpoints (endpoints:/v3/endpoints);
- tokens (endpoints: /v3/tokens).

Projects – це сутність, яка представляє проєкт (тенант), тобто логічну область, у межах якої групуються користувачі, ролі, квоти та хмарні ресурси (VM, мережі, томи, тощо). Відповідно /v3/projects – це endpoints (кінцеві точки REST API), через які здійснюється робота із сутністю Projects. Наприклад, GET /v3/projects – отримати список проєктів або POST/v3/projects – створити новий проєкт.

Users – це сутність, яка представляє облікові записи користувачів хмарної платформи, тобто користувач від імені цієї сутності виконує дії. Вона містить відомості про користувача, зокрема його ідентифікатор, ім’я, стан облікового запису та приналежність до домену. Його endpoints (/v3/users) дозволяє отримати список користувачів, створити нового користувача, або змінити його дані чи видалити. Зокрема, користувачі можуть створюватися як безпосередньо засобами Identity service, так і автоматично під час автентифікації через систему керування ідентифікацією Keycloak. Якщо у разі входу користувача до хмарної платформи відповідний обліковий запис відсутній у локальній БД, то Identity service автоматично створює його, використовуючи універсальний унікальний ідентифікатор UUID, що отримується із токена Keycloak. Такий підхід забезпечує використання єдиного ідентифікатора користувача в усіх компонентах програмного комплексу, що спрощує його ідентифікацію та супровід. Зберігання локальних облікових записів необхідне для призначення користувачам ролей і

прав доступу в межах проєктів, а також для подальшого керування цими повноваженнями. Модель користувача у форматі JSON наведена на рис 3.2.


```
{
    "user": {
        "default_project_id": "000000",
        "domain_id": "000000",
        "enabled": true,
        "id": "000000",
        "name": "demo",
        "password_expires_at": "2026-07-15T15:32:17.000000"
    }
}
```


Рисунок 3.2 – Реалізація моделі Users як сутності сервісу Identity

Roles – це сутність, яка визначає ролі користувачів, тобто набір прав і дозволів на виконання певних дій у хмарній платформі. Роль сама по собі не надає доступу, а призначається користувачеві або групі в межах конкретного проєкту чи домену. Через його endpoints (/v3/roles) здійснюється управління ролями: їх створення, перегляд, зміна та видалення.

Tokens – це сутність, яка використовується для автентифікації користувачів і підтвердження їх права доступу до сервісів хмарної платформи. Токен є тимчасовим цифровим маркером, який видається після успішної автентифікації та використовується замість повторного введення облікових даних під час кожного звернення до API. Через його endpoints (/v3/tokens) здійснюється створення, перевірка та відкликання токенів.

Service – це сутність, яка містить інформацію про зареєстровані сервіси програмної платформи, зокрема їх назву, внутрішнє службове ім'я, версію та унікальний ідентифікатор. Ці відомості використовуються механізмом Service Discovery для автоматичного виявлення доступних сервісів і організації взаємодії між ними. Реєстрація та конфігурування сервісів здійснюються за допомогою конфігураційного файла. Модель сервісу у форматі JSON наведена на рис 3.3.

Endpoints – це сутність, яка містить інформацію про кінцеві точки доступу (URL-адресу) зареєстрованих сервісів. Кожний endpoint є пов'язаним із відповідним сервісом через його унікальний ID та визначає мережуу адресу, за якою цей сервіс є доступним для взаємодії. Ці дані використовуються механізмом Service Discovery для автоматичного пошуку сервісів і маршрутизації запитів між

компонентами розробленої програмної платформи. Налаштування endpoints здійснюється за допомогою конфігураційного файлу. Модель endpoint у форматі json в рамках її запуску на локальному ПК наведена на рис. 3.4.

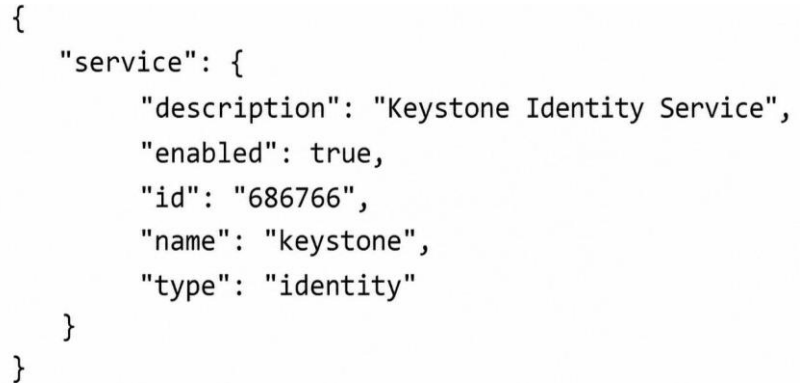


Рисунок 3.3 – Реалізація моделі Service як сутності сервісу Identity

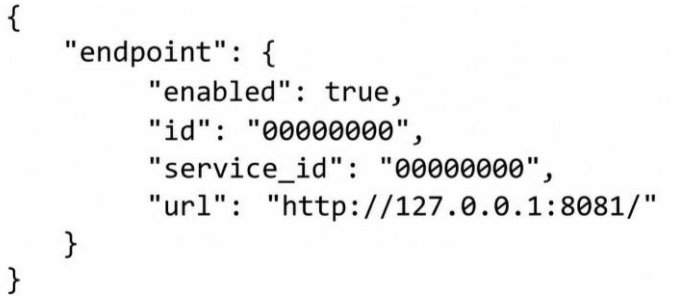


Рисунок 3.4 – Реалізація моделі Endpoint як сутності сервісу Identity

3.3.2 Сервіс керування блоковими сховищами – Block storage

Block storage service – це сервіс, що призначений для керування постійними ресурсами блокового сховища даних, які використовуються зокрема VM. Він забезпечує створення, зміну, видалення та підключення блокових томів до обчислювальних ресурсів у межах проєкту. За замовчуванням один блоковий том може бути підключений лише до однієї VM одночасно, якщо інше не передбачено його конфігурацією.

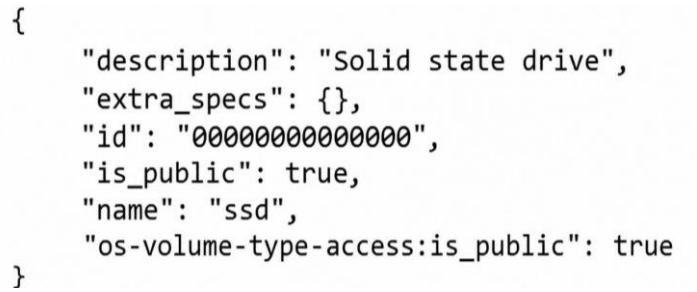
У процесі роботи Block storage service взаємодіє з Identity service, отримуючи інформацію про існуючі проєкти та права доступу користувачів, а також із Relations service, який забезпечує збереження залежностей між

блоковими томами, VM та проєктами, у межах яких вони створені. Для функціонування сервісу необхідні власна БД і підключення до брокера повідомлень Apache Kafka, через який передаються події про створення, зміну, підключення або видалення ресурсів. Безпосереднє створення блокових томів та їх підключення до VM у конкретній платформі віртуалізації здійснюється відповідним провайдером платформи віртуалізації (VMware, Nutanix, тощо – дивись розроблену архітектуру у додатку А). До основних сутностей Block storage service відносяться наступні:

- volume types (endpoints: /v3/types);
- volumes (endpoints: /v3/volumes);
- attachments (endpoints: /v3/attachments);
- os-quota-sets (endpoints: /v3/os-quota-sets).

Сутність volume types визначає тип блокового сховища, яке використовується у процесі створення тому. Вона містить інформацію про тип фізичного або логічного носія даних, наприклад твердотільний накопичувач (SSD), традиційний жорсткий диск (HDD) або дисковий масив. Використання різних типів сховищ дозволяє користувачеві обирати необхідні характеристики продуктивності та ємності ресурсів залежно від вимог конкретного додатку.

Модель сутності volume types у форматі JSON наведена на рис 3.5.



```
{
    "description": "Solid state drive",
    "extra_specs": {},
    "id": "00000000000000",
    "is_public": true,
    "name": "ssd",
    "os-volume-type-access:is_public": true
}
```


Рисунок 3.5 – Реалізація моделі volume types як сутності сервісу Block storage

Сутність volumes описує окремий блоковий том, який є аналогом зовнішнього жорсткого диска та може бути підключений до VM. Для кожного тому підтримується набір станів, що відображає його поточний життєвий цикл:

- creating – створення тому;
- available – том створено та готовий до підключення;
- reserved – том зарезервований;

- attaching – виконується підключення до VM;
- detaching — виконується відключення;
- in-use - том використовується VM;
- deleting – виконується видалення тому;
- error – під час створення або використання тому виникла помилка.

Модель сутності volumes у форматі JSON наведена на рис 3.6.


```
{
        "volume": {
        "attachments": [],
        "availability_zone": "nova",
        "bootable": "false",
        "created_at": "2026-07-15T06:50:07.770785",
        "description": null,
        "encrypted": false,
        "id": "f7223234-1afc-4d19-bfa3-d19deb6235ef",
        "metadata": {},
        "multiattach": false,
        "name": "demo",
        "os-vol-tenant-attr:tenant_id": "89afd400-b646-4bbc-b12b-c0a4d63e5bd3",
        "size": 10,
        "status": "creating",
        "updated_at": null,
        "user_id": "c853ca26-e8ea-4797-8a52-ee124a013d0e",
        "volume_type": "__DEFAULT__",
        "volume_type_id": "5fed9d7c-401d-46e2-8e80-f30c70cb7e1d",
        }
}
```


Рисунок 3.6 – Реалізація моделі volumes як сутності сервісу Block storage

Сутність attachments використовується для керування процесом підключення та відключення блокових томів до VM, а також містить інформацію про поточний зв'язок між відповідним томом і обчислювальним ресурсом. Підтримує наступні стани підключення:

- attached –підключення тому зроблено;
- attaching – том знаходиться у процесі підключення;
- detached – проведено відключення тому;
- reserved – резервування тому;
- error_attaching – при підключенні тому виникла помилка;
- error_detaching – при відключенні тому виникла помилка.

Сутність os-quota-sets призначена для керування квотами проєкту. Вона дозволяє встановлювати та отримувати обмеження на максимальну кількість

блокових томів, які можуть бути створені в межах проєкту, а також визначати граничний сумарний обсяг дискового простору, що є доступним користувачам.

3.3.3 Сервіс образів серверів – Images

Image service – це сервіс, призначений для керування образами VM, які використовуються під час розгортання віртуальних машин користувача і managed-services. Цей сервіс підтримує завантаження образів у форматах VMDK (Virtual Machine Disk), QCOW2 (QEMU Copy-On-Write version 2) та інших, що підтримуються платформами віртуалізації.

У процесі роботи Image Service взаємодіє із relations service, який забезпечує контроль залежностей між образами та створеними на їх основі VM. Це унеможливлює видалення образу, якщо він використовується іншими ресурсами системи. Водночас інтеграція з Identity service не потрібна, оскільки образи є глобальними ресурсами платформи і не належать окремим проєктам. Метадані образів зберігаються в БД, тоді як завантаження файлів образів до конкретної платформи віртуалізації здійснюється відповідним провайдером цієї платформи.

Для управління доступністю образів використовуються додаткові властивості. Зокрема, параметр os_hidden дозволяє приховати застарілі або службові образи зі списку доступних, а властивість visibility визначає область їх видимості та доступу для користувачів.

Основною сутністю цього сервісу є image, яка містить інформацію про образ, його унікальний ідентифікатор, поточний стан та інші службові характеристики. Унікальний ID образу використовується всіма сервісами програмного комплексу для його однозначної ідентифікації. Крім основних параметрів, до образу можуть додаватися теги та довільні метадані, які спрощують його класифікацію, пошук і подальше використання. Протягом свого життєвого циклу image може перебувати в одному з кількох статусних станів:

- статус queued означає, що для образу вже створено запис і зарезервовано ідентифікатор, однак файл образу ще не завантажено;
- статус uploading вказує, що процес передавання файлу до сховища триває;
- після успішного завершення завантаження image переходить у стан active, що означає його готовність до використання під час створення VM;

- статус deactivated свідчить про тимчасове вимкнення image без його видалення, тому він буде недоступний для використання;

- якщо під час завантаження виникла помилка, image отримує статус killed;

- після видалення image – отримується статус deleted, що означає збереження службової інформації про нього за відсутності самого файлу образу.

Модель сервісу image у форматі JSON наведена на рис. 3.7:


```
{
    "status": "active",
    "name": "cirros-0.3.2-x86_64-disk",
    "tags": [],
    "container_format": "bare",
    "created_at": "2026-07-15T17:15:10Z",
    "disk_format": "qcow2",
    "updated_at": "2026-07-15T17:15:11Z",
    "visibility": "public",
    "min_disk": 0,
    "protected": false,
    "id": "1bea47ed-f6a9-463b-b423-14b9cca9ad27",
    "file": "/v2/images/1bea47ed-f6a9-463b-b423-14b9cca9ad27/file",
    "checksum": "64d7c1cd2b6f60c92c14662941cb7913",
    "os_hash_algo": "sha512",
    "os_hash_value": "ьщьершгпмфдгу",
    "os_hidden": false,
    "owner": "0880000000000000",
    "size": 13167616,
    "min_ram": 1024
}
```


Рисунок 3.7 – Реалізація моделі сервісу image

3.3.4 Сервіс керування мережною інфраструктурою – Networking

Networking service – це сервіс, що призначений для створення та керування віртуальною мережною інфраструктурою хмарної платформи. Він надає API для побудови мережних топологій, налаштування мережних з'єднань, адресного простору та мережних політик, а також забезпечує взаємодію VM у хмарі.

Основним призначенням Networking service є створення та адміністрування віртуальних мереж, підмереж, маршрутизаторів і комутаторів, які використовуються VM, управління якими здійснює обчислювальна служба, що реалізована сервісом Nova платформи OpenStack. Для забезпечення коректної роботи сервіс інтегрується з Identity service, який виконує автентифікацію та авторизацію користувачів, а також забезпечує створення мереж у межах

відповідних проєктів. Крім того, Networking service взаємодіє з Relations Service, який використовується для контролю залежностей між мережами та іншими сутностями програмного комплексу. Безпосереднє створення мережуих ресурсів у платформі віртуалізації здійснюється відповідним провайдером. Основними сутностями Networking service є:

- networks (endpoints: /v2.0/networks);
- subnets (endpoints: /v2.0/subnets).

Сутність networks описує віртуальну мережу другого рівня моделі OSI (L2) та використовується для її створення та ідентифікації в платформі віртуалізації. До створеної мережі можуть підключатися VM, що забезпечує їх мережну взаємодію. Під час створення мережі визначається віртуальна локальна мережа (Virtual Local Area Network, VLAN), пакети якої використовуються для ПД. Мережа може бути спільною для всіх проєктів або доступною лише в межах одного проєкту, однак незалежно від режиму доступу вона завжди створюється в конкретному проєкті. Для спрощення класифікації мереж підтримується використання тегів, а кожна зміна конфігурації супроводжується збільшенням значення унікального Revision Number, що дозволяє запобігти конфліктам під час одночасного оновлення даних кількома користувачами або сервісами.

Протягом життєвого циклу мережа може перебувати в одному з кількох станів:

- active означає, що мережа створена та готова до використання;
- build на процес її створення;
- down вказує на тимчасову недоступність мережі;
- error свідчить про виникнення помилки під час створення або налаштування мережного ресурсу.

Модель сутності networks у форматі JSON наведена на рис 3.8.

Сутність subnets описує віртуальну підмережу третього рівня моделі OSI (L3), яка створюється в межах раніше сформованої мережі L2. Вона визначає мережну адресу, маску підмережі та діапазон IP-адрес, що можуть автоматично призначатися DHCP-сервером або задаватися вручну у процесі створення VM. Як і для мереж, до підмереж можуть додаватися службові теги, а контроль одночасних змін забезпечується за допомогою унікального Revision Number. На відміну від сутності Networks, сутність Subnets не має власних статусних станів.

Модель сутності subnets у форматі JSON наведена на рис 3.9.

```
{
    "network": {
        "admin_state_up": true,
        "availability_zone_hints": [],
        "availability_zones": [
            "nova"
        ],
        "created_at": "2026-07-15T20:19:41",
        "dns_domain": "my-domain.org.",
        "id": "0000000000000000",
        "12_adjacency": false,
        "name": "private-network",
        "project_id": "0000000000",
        "revision_number": 1,
        "shared": true,
        "status": "ACTIVE",
        "subnets": [
            "00000000000000"
        ],
        "tags": ["tag1,"tag2"],
        "tenant_id": "000000000000",
        "updated_at": "2026-07-15T20:19:41",
        "vlan_transparent": false,
        "description": "",
        "is_default": true
    }
}
```


Рисунок 3.8 – Реалізація моделі networks як сутності сервісу Networking


```
{
    "subnet": {
       "name": "my_subnet",
       "enable_dhcp": true,
       "network_id": "00000000000000000000000",
       "segment_id": null,
       "project_id": "000000000000000000",
       "tenant_id": "00000000000000000",
       "created_at": "2026-07-15T20:19:41",
       "allocation_pools": [
            {
              "start": "192.0.0.2",
              "end": "192.255.255.254"
            }
       ],
       "ip_version": 4,
       "gateway_ip": "192.0.0.1",
       "cidr": "192.0.0.0/8",
       "updated_at": "2026-07-15T20:19:41",
       "id": "00000000000000000000",
       "description": "",
       "revision_number": 2,
       "tags": ["tag1","tag2"]
}     }
```


Рисунок 3.9 – Реалізація моделі subnets як сутності сервісу Networking

3.3.5 Сервіс керування VM – Compute

Compute Service – це сервіс, призначений для надання обчислювальних ресурсів хмарної платформи у вигляді VM. Він забезпечує створення, запуск, зупинку, зміну конфігурації, перезавантаження та видалення VM, а також керує їх життєвим циклом.

Для виконання своїх функцій сервіс Compute взаємодіє з кількома сервісами розробленої програмної платформи. Зокрема інтеграція з сервісом Identity забезпечує автентифікацію та авторизацію користувачів, а також створення VM у межах відповідних проєктів. Сервіс Image використовується для отримання та перевірки образів ОС, на основі яких розгортаються VM. Взаємодія з сервісом Networking необхідна для отримання параметрів мереж і перевірки можливості їх використання у процесі створення сервера. Крім того, Relations Service забезпечує створення та видалення залежностей між VM та іншими сутностями програмного комплексу.

Для функціонування Compute service використовуються власна БД і брокер повідомлень Apache Kafka, через який передаються події про всі операції з VM. Безпосереднє створення обчислювальних ресурсів у платформі віртуалізації виконується відповідним провайдером. За необхідності сервіс може взаємодіяти з іншими компонентами архітектури, зокрема з Block storage service, для підключення до VM постійних блокових томів.

Основними сутностями сервісу Compute є:

- flavors (endpoints: /v2/flavors);
- servers (endpoints: /v2/servers);
- keypairs (endpoints: /v2/keypairs);
- os-quota-sets (endpoints: /v2/os-quota-sets).

Сутність flavors визначає шаблони конфігурації VM. Вона містить параметри, що характеризують обчислювальні ресурси сервера, зокрема кількість віртуальних процесорів, обсяг оперативної пам'яті та дискового простору, який виділяється під час створення VM. При цьому обсяг зовнішніх блокових томів, що можуть бути підключені додатково, до цієї конфігурації не входить.

Модель сутності flavors у форматі JSON наведена на рис 3.10.

```
{
    "flavor": {
        "OS-FLV-DISABLED:disabled": false,
        "disk": 20,
        "os-flavor-access:is_public": true,
        "id": "7",
        "name": "demo",
        "ram": 2048,
        "swap": 0,
        "vcpus": 1,
        "rxtx_factor": 1.0,
        "description": "test description"
    }
}
```


Рисунок 3.10 – Реалізація моделі flavors як сутності сервісу Compute

Сутність servers є основною сутністю сервісу Compute і описує VM як обчислювальний ресурс. У процесі її створення обов'язково визначається відповідний шаблон Flavor, який задає характеристики сервера. Зміна конфігурації VM також виконується шляхом вибору іншого шаблону Flavor. У процесі здійснення запуску сервера може передаватися сценарій користувача щодо початкових налаштувань, що здійснюються автоматично, який сформований у форматі Cloud-Init та закодований у Base64. Використання програмного механізму Cloud-Init дозволяє автоматизувати первинне налаштування ОС після запуску VM, зокрема створення користувачів, встановлення ПЗ, конфігурування мережних параметрів та виконання інших адміністративних операцій. Base64 – це стандартний спосіб кодування двійкових даних у текстовий формат, що використовується для безпечної передачі файлів або службових даних через API та мережні протоколи. Протягом свого життєвого циклу VM може перебувати в одному з кількох станів, зокрема:

- статус ACTIVE означає, що сервер успішно створений і готовий до використання;
- статус BUILD вказує на процес його створення;
- статус REBOOT вказує на виконання перезавантаження;
- статус RESIZE вказує на зміну конфігурації обчислювальних ресурсів;
- статус SHUTOFF вказує на вимкнений стан сервера;
- статус DELETED вказує на успішне видалення VM;
- статус ERROR вказує на виникнення внутрішньої помилки;

- статус UNKNOWN свідчить про неможливість визначити поточний стан сервера.

Модель сутності servers у форматі JSON наведена на рис 3.11.


```
{
      "server": {
        "OS-DCF:diskConfig": "AUTO",
        "OS-EXT-AZ:availability_zone": "nova",
        "OS-EXT-SRV-ATTR:host": "compute",
        "OS-EXT-SRV-ATTR:hostname": "new-server-test",
        "OS-EXT-SRV-ATTR:user_data":
"IyEvYm1uL2Jhc2gKL2Jpbi9zdG9pY2hvIC3JIGFtIGluIHJvdSEiCg==",
        "OS-EXT-STS:power_state": 1,
        "OS-EXT-STS:vm_state": "active",
        "OS-SRV-USG:launched_at": "2026-07-15T15:19:10.855016",
        "OS-SRV-USG:terminated_at": null,
        "accessIPv4": "1.2.3.4",
        "accessIPv6": "80fe::",
        "created": "2026-07-15T15:19:09Z",
        "description": null,
        "flavor": {},
        "id": "0000000000000000000",
        "key_name": null,
        "locked": true,
        "locked_reason": "I don't want to work",
        "metadata": {
               "My Server Name": "Apache1"
        },
        "name": "demo-server",
        "os-extended-volumes:volumes_attached": [],
        "server_groups": [],
        "status": "ACTIVE",
        "tags": [],
        "tenant_id": "000000000000",
        "updated": "2026-07-15T15:19:11Z",
        "user_id": "admin"
      }
}
```


Рисунок 3.11 – Реалізація моделі servers як сутності сервісу Compute

Сутність keypairs призначена для керування парами криптографічних ключів, що використовуються для безпечного доступу до VM за протоколом SSH (Secure Shell). Вона забезпечує імпорт і зберігання відкритих ключів користувачів, які можуть автоматично додаватися до ОС під час створення сервера. Це дозволяє забезпечити безпечний доступ до VM відразу після її запуску без необхідності попереднього внесення ключів до образу або їх ручного налаштування.

Модель пари криптографічних ключів у відповідності із сутністю keypairs у форматі JSON наведена на рис 3.12.

```
{
      "keypair": {
      "fingerprint": "44:fe:29:6e:23:14:b9:53:5b:65:82:58:1c:fe:5a:c3",
      "name": "keypair-demo",
      "type": "ssh",
      "public_key": "ssh-rsa ___",
      "user_id": "fake",
      "deleted": false,
      "created_at": "2026-07-15T12:06:13.681238",
      "updated_at": null,
      "deleted_at": null,
      "id": 1
}
```


Рисунок 3.12 – Реалізація моделі keypairs як сутності сервісу Compute

3.3.6 Сервіс керування контейнерними кластерами Kubernetes – Magnum

Magnum service призначений для керування життєвим циклом контейнерних кластерів Kubernetes. Він надає стандартизований API для створення, масштабування, оновлення та адміністрування цих кластерів. Для виконання своїх функцій сервіс Magnum взаємодіє з іншими сервісами розробленої програмної платформи. Сервіс Image використовується для отримання образів оОС, а Compute – для створення VM, на яких розгортається кластер Kubernetes. Автоматичне розгортання кластера виконується за допомогою попередньо підготовленої ролі Ansible, яка запускається сервісом AWX через REST API. Такий підхід забезпечує автоматизацію процесу створення та подальшого керування контейнерною інфраструктурою.

Особливістю Magnum service є підтримка повного життєвого циклу контейнерних кластерів. Сервіс дозволяє змінювати кількість робочих вузлів кластера, виконувати оновлення версії Kubernetes, а також централізовано керувати його конфігурацією без необхідності ручного адміністрування окремих VM.

Основні сутності сервісу Magnum:

- clustertemplate (endpoints: /v1/clustertemplates);
- clusters (endpoints: /v1/clusters).

Сутність clustertemplate визначає шаблон контейнерного кластера, який містить набір параметрів, що є необхідним для його розгортання. До таких параметрів належать характеристики інфраструктури кластера, конфігурація його

вузлів, а також функціональні можливості та налаштування оркестратора. Зазвичай користувач спочатку створює шаблон, після чого на його основі може розгортати один або декілька кластерів. Видалення шаблону неможливе, якщо хоча б один створений за цим шаблоном кластер продовжує існувати.

Модель сутності clustertemplate у форматі JSON наведена на рис 3.13.


```
{
   "http_proxy":"http://10.164.177.169:8080",
   "floating_ip_enabled":true,
   "fixed_subnet":null,
   "master_flavor_id":null,
   "uuid":"00000000000000000000000000",
   "no_proxy":"10.0.0.0/8,172.0.0.0/8,192.0.0.0/8,localhost",
   "https_proxy":"http://10.164.177.169:8080",
   "tls_disabled":false,
   "keypair_id":"0000000000",
   "public":false,
   "docker_volume_size":3,
   "server_type":"vm",
   "external_network_id":"public",
   "cluster_distro":"fedora-atomic",
   "image_id":"fedora-atomic-latest",
   "volume_driver":"cinder",
   "registry_enabled":false,
   "docker_storage_driver":"devicemapper",
   "apiserver_port":8443,
   "name":"k8s-bm2",
   "fixed_network":null,
   "coe":"kubernetes",
   "flavor_id":"m1.small",
}  "dns_nameserver":"8.8.8.8"
```


Рисунок 3.13 – Реалізація моделі clustertemplate як сутності сервісу Magnum

Сутність clusters описує розгорнутий екземпляр кластера Kubernetes, що створений на основі відповідного шаблону. У процесі його розгортання сервіс Magnum використовує параметри, що визначені у сутністю clustertemplate, після чого автоматично створює необхідну інфраструктуру та виконує налаштування оркестратора. Після успішного завершення цього процесу кластер переходить у робочий стан і стає готовим до розгортання контейнерів.

Модель сутності clusters у форматі JSON наведена на рис 3.14.

```
{
   "status":"CREATE_COMPLETE",
   "uuid":"00000000000000000000000000000000",
   "created_at":"2026-07-15T06:51:31+00:00",
   "api_address":"https://172.24.4.6:6443",
   "updated_at":"2026-07-15T06:53:24+00:00",
   "master_count":1,
   "coe_version": "v1.24.0",
   "keypair":"my_keypair",
   "cluster_template_id":"00000000000000000000000000000000",
   "master_addresses":[
       "172.24.4.6"
   ],
   "node_count":1,
   "node_addresses":[
       "172.24.4.13"
   ],
   "status_reason":"completed",
   "create_timeout":60,
   "name":"k8s-demo"
}
```


Рисунок 3.14 – Реалізація моделі clusters як сутності сервісу Magnum

ВИСНОВКИ

У цій кваліфікаційній роботі проведено аналіз сучасних підходів, що застосовуються при створенні комплексних програмних платформ для розгортання хмарних технологій та надання різноманітних сервісів і додатків. Зокрема розглянуті особливості функціонування існуючих базових програмних платформ OpenStack, VMware vSphere, Nutanix Enterprise Cloud Platform, OpenNebula та AOMEI MultCloud з позиції можливості їх застосування для створення комплексної програмної платформи хмарних обчислень. Запропоновано використовувати платформу OpenStack і дотримуватися мікросервісного підходу. Визначено склад основних сервісних модулів, що складають ядро комплексної програмної платформи, та проаналізовано їх функціональне призначення, взаємодію та принципи інтеграційних процесів, що ними реалізуються. На основі цього було розроблено архітектуру комплексної програмної платформи для розгортання абстрактної хмарної інфраструктури, що орієнтована на підтримку різних платформ віртуалізації, автоматизацію керування ресурсами та забезпечення її масштабованості, розширюваності і сумісності з OpenStack API.

У першому розділі кваліфікаційної роботи було розглянуто фундаментальну концепцію, загальні принципи, типи і моделі організації хмарних обчислень. Зокрема послідовно описані етапи розвитку хмарних обчислень. Показано, що хмарні обчислення у значній мірі розвивалися завдяки удосконаленням апаратного забезпечення. Насамперед це стосується появи багатоядерних процесорів, збільшення обсягів оперативної пам'яті та суттєвого зростання ємності накопичувачів даних. Такі технологічні зміни не лише підвищили продуктивність хмарних платформ, а й зробили їх доступними не тільки для великих підприємств, але й для малого бізнесу та індивідуальних користувачів. Показано, що базова функціональна архітектура хмарних обчислень складається з взаємопов'язаних компонентів: front-end і back-end. Ці компоненти здійснюють взаємодію через мережу Інтернет та реалізують надання різноманітних ресурсів, веб-сервісів та веб-додатків за запитом користувача [4, 5, 9, 11]

Крім того, у першому розділі були розглянуті особливості основних моделей обслуговування IaaS, PaaS і SaaS, а також визначено переваги та сфери застосування публічної, приватної та гібридної моделей розгортання хмар.

Показано, що ключовими характеристиками хмарних обчислень є самообслуговування, широкий мережний доступ, об'єднання ресурсів, швидка еластичність і вимірюваність сервісів, а також масштабованість та віртуалізація, які забезпечують гнучкий розподіл ресурсів, автоматизацію процесів керування та ефективне функціонування всієї хмарної інфраструктури та її сервісів [4, 8, 11]. Отримані результати стали теоретичною основою для подальшого аналізу існуючих платформ і розробки архітектури комплексної програмної платформи хмарних обчислень.

У другому розділі кваліфікаційної роботи був проведений аналіз та порівняння комплексних рішень існуючих програмних продуктів для розгортання і надання хмарних сервісів. Зокрема були проаналізовані платформи AOMEI Technology MultCloud, VMware vSphere, Nutanix Enterprise Cloud Platform, OpenNebula, OpenStack. Аналіз показав, що кожна з них орієнтована на вирішення окремих задач побудови та керування хмарною інфраструктурою, однак має певні функціональні обмеження щодо підтримки різних платформ віртуалізації, розширення сервісів або інтеграції з іншими рішеннями. За результатами порівняльного аналізу було зроблено висновок, що OpenStack найбільш повно відповідає вимогам до створення комплексної програмної платформи розгортання хмарної інфраструктури. Це обумовлено його відкритою модульною архітектурою, підтримкою стандартизованого API, можливістю інтеграції з різними платформами віртуалізації та широкими можливостями масштабування і розширення функціональності. Саме ці особливості стали підставою для вибору OpenStack як базової основи для подальшої розробки архітектури комплексної програмної платформи, що здатна працювати з різними платформами віртуалізації та забезпечувати централізоване керування ресурсами хмарної інфраструктури.

У третьому розділі була проведене обґрунтування та визначення архітектурного рішення комплексної програмної платформи для розгортання хмарних обчислень. В результаті чого була розроблена архітектура комплексної програмної платформи для розгортання та надання хмарних послуг на базовій основі OpenStack. Зокрема с початку був обґрунтований мікросервісний підхід. При цьому зазначено, що розподіл функціональних обов'язків між окремими сервісами буде здійснюватися за принципом, аналогічним модульній архітектурі платформи OpenStack, що забезпечує незалежність окремих компонентів, спрощує їх масштабування та подальший розвиток програмної платформи. Також

був визначений і обґрунтований набір сервісних модулів, що забезпечують її функціональність і керованість. Зокрема показано, що реалізація сервісів Identity**,** Compute, Networking, Block Storage, Image, Magnum та інших забезпечує централізоване керування обчислювальними, мережними, та контейнерними ресурсами, а також сховищами даних і механізмами автентифікації та авторизації користувачів. Взаємодія сервісів, що здійснюється через стандартизовані API, брокер повідомлень та сервіси керування залежностями дозволяє забезпечити цілісність даних, автоматизацію процесів і узгоджену роботу всіх компонентів системи. Використання мікросервісної архітектури створює передумови для незалежного масштабування, модернізації та розширення функціональних можливостей програмної платформи, а також забезпечує її сумісність із різними платформами віртуалізації та сучасними технологіями автоматизації розгортання хмарної інфраструктури.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Лобанов А.А. Облачные вычисления как развитие информационного сервиса [Електронний ресурс] / А.А Лобанов // Перспективы науки и образования. – №2. – 2014. – С. 40 - 45. – Режим доступу до ресурсу: http://cyberleninka.ru/article/n/oblachnye-vychisleniya-kak-razvitie-informatsionnogo-servisa.

2. Владимир Сафонов Платформа облачных вычислений Microsoft Windows Azure [Електронний ресурс] / Учебный Internet-курс. –2012. – Режим доступу до ресурсу: http://www.intuit.ru/studies/courses/2314/614/info.

3. Батура Т.В. Облачные технологии: основные модели, приложения, концепции и тенденции развития/ Т.В. Батура, Ф.А. Мурзин, Д.Ф. Семич // Программные продукты и системы. – 2014. – №3. – С. 64 – 72.

4. Gets A. Основы облачных вычислений – характеристики, типы, модели [Електронний ресурс] / Adam Gets // компания LinkedIn. – 2022. – Режим доступу до ресурсу: https://ru.linkedin.com/pulse/cloud-computing-essentials-characteristics-types-adam?tl=ru.

5. Клементьев И.П. Введение в облачные вычисления: учеб. курс [Електронний ресурс] / И.П. Клементьев, В.А. Устинов // НОУ «ИНТУИТ». – 2011. – Режим доступу: https://www.intuit.ru/studies/courses/673/529/info.

6. Windows Azure сегодня – что это, зачем и для кого [Електронний ресурс] // PC Week/RE. – 2013. – №10 (830). Режим доступу до ресурсу: https://www.itweek.ru/its/article/detail.php?ID=149720.

7. Федоров А. Windows Azure™: облачная платформа Microsoft®/ Алексей Федоров, Дмитрий Мартынов // Компания Microsoft – 2010. – 100 с.

8. What are Characteristics of Cloud Computing? [Електронний ресурс] // YouStable. – 2026. – Режим доступу до ресурсу: https://www.youstable.com/ru/Блог/Каковы-характеристики-облачных-вычислений/.

9. Пойченко Д. П. Аналіз підходів до оцінки ефективності хмарних технологій [Електронний ресурс] / Д. П. Пойченко // магістерська кваліфікаційна робота. – Харків: ХНУРЕ, 2022. – 65 с. – Режим доступу до ресурсу: https://openarchive.nure.ua/entities/publication/cab6bf00-5236-485e-b748-e18b707feb75

10. Детков Е.А. Виртуализация и ее особенности / Е.А. Детков, А.А. Коревых // Международный журнал прикладных наук и технологий Integral. – №3. – 2019. – С. 181 - 193.

11. Singh A. What is Cloud Computing Architecture? Advantages and Disadvantages [Електронний ресурс] / Anshuman Singh // Shiksha Online. – Режим доступу до ресурсу: https://www.shiksha.com/online-courses/articles/cloud-computing-architecture-advantages-and-disadvantages/.

12. Райт І. Моделі хмарних сервісів: пояснення IaaS, PaaS та SaaS [Електронний ресурс] / Ітан Райт // Guru99. – Доступ здійснено 13.07.2026. – Режим доступу до ресурсу: https://www.guru99.com/uk/cloud-service-models.html.

13. Balykina A. Хмарні обчислення. Теорія та практика [Електронний ресурс] / Alexandra Balykina // Блог компанії SIM-Networks. – 2023. – Режим доступу до ресурсу: https://www.sim-networks.com/ukr/blog/cloud-technologies.

14. Chernikov I. Публічна хмара: 6 переваг та 3 недоліки [Електронний ресурс] / Illia Chernikov // Блог компанії SIM-Networks. – 2022. – Режим доступу до ресурсу: https://www.sim-networks.com/ukr/blog/benefits-of-public-cloud.

15. Balykina A. Що таке приватна хмара? Визначення та кейси використання [Електронний ресурс] / Alexandra Balykina // Блог компанії SIM-Networks. – 2022. – Режим доступу до ресурсу: https://www.sim-networks.com/ukr/blog/what-is-a-private-cloud.

16 Медведев А. Облачные технологии: тенденции развития, примеры исполнения // Современные технологии автоматизации. – 2013. – № 2. – С. 6–9.

17. Кто такой поставщик управляемых услуг (MSP)? [Електронний ресурс] // Trend Micro. – 2025. – Режим доступу до ресурсу: https://www.trendmicro.com/ru_ru/what-is/xdr/managed-service-provider.html.

18. AOMEI Multcloud cloud storage review [Електронний ресурс] // TechRadar Pro: the business technology experts. – Доступ здійснено 23.05.2026. – Режим доступу до ресурсу: https://www.techradar.com/reviews/aomei-multcloud.

19. Бархатов О. Що таке віртуалізація VMware і як вона працює [Електронний ресурс] / Олександр Бархатов // ООО «ТАЙМВЭБ.КЛАУД». – 2025. – Режим доступу до ресурсу: https://timeweb.cloud/blog/chto-takoe-virtualizaciya-vmware-i-kak-ona-rabotaet#vmware-vsphere.

20. Решения на платформе Nutanix [Електронний ресурс] / компания INT: Innovative Network Technologies. –https://innettech.ru/nutanix-enterprise-cloud-platform.

21. OpenNebula: Enterprise Cloud and Virtualization Platform [Електронний ресурс]. – Доступ здійснено 25.05.2026. – Режим доступу до ресурсу: https://opennebula.io/?utm_source=chatgpt.com.

22. Amri Aymen Недостающее введение в контейнеризацию [Електронний ресурс] / Aymen Eon Amri // Habr. – 2021. – Режим доступу до ресурсу: https://habr.com/ru/articles/541288/.

23. Shurup Новая статистика CNCF о контейнерах, cloud native и Kubernetes [Електронний ресурс] / Блог компании Флант // Habr. – 2018. – Режим доступу до ресурсу: https://habr.com/ru/companies/flant/articles/422949/.

24. OpenNebula 6.6.1 Documentation [Електронний ресурс] // OpenNebula – open source cloud management platform. – Режим доступу до ресурсу: https://docs.opennebula.io/6.6/index.html.

25. Chernikov I. Хмара OpenStack – переваги та недоліки [Електронний ресурс] / Illia Chernikov // Блог компанії SIM-Networks. – 2022. – Режим доступу до ресурсу: https://www.sim-networks.com/ukr/blog/openstack-cloud.

26. OpenStack [Електронний ресурс] // компания ITGLOBAL.COM BEL (ООО «ИТГЛОБАЛКОМ БЕЛ»). – Доступ здійснено 25.05.2026. – Режим доступу до ресурсу: https://itglobal.com/ru-by/company/glossary/openstack/.

27. OpenStack Architecture Design Guide [Электронный ресурс] // OpenStack. – Доступ здійснено 14.07.2026. – Режим доступу до ресурсу: https://docs.openstack.org/arch-design/.