Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Інфокомунікаціїі*
(повна назва)

Кафедра *Інформаційно-мережної інженерії*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти *другий (магістерський)*

Дослідження інструментів автоматизації розгортування приватної гіперконвергентної хмари на базі OpenStack
(тема)

Виконав:
студент 2 курсу, групи *ІМІм-20-2*
Коротіч О.В.

Спеціальності *172 Телекомунікації та радіотехніка*
(код і повна назва спеціальності)
Тип програми *Освітньо-наукова*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Інформаційно-мережна інженерія*
(повна назва освітньої програми)
Керівник *доц. Костромицький А.І.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ *Безрук В.М.*
(підпис) (прізвище, ініціали)

2022 р.

Не містить відомостей, заборонених до відкритого публікування

Студент _____ *Коротіч О.В.*
(підпис) (прізвище та ініціали)

Керівник _____ *Костромицький А.І.*
(підпис) (прізвище та ініціали)

Харківський національний університет радіоелектроніки

Факультет *Інфокомунікацій*
(повна назва)

Кафедра *Інформаційно-мережної інженерії*
(повна назва)

Рівень вищої освіти *другий (магістерський)*

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 ІМІ_____
(підпис)
“____” _____2022 року

ЗАВДАННЯ
НА КВАЛІФЛКАЦІЙНУ РОБОТУ

Студентові *Коротіч Олексію Віталійовичу*
(прізвище, ім’я, по батькові)

1. Тема роботи *Дослідження інструментів автоматизації розгортування приватної гіперконвергентної хмари на базі OpenStack*
затверджені наказом університету від 14 березня 2022 року № 592 *Ст*
2. Термін подання студентом роботи до екзаменаційної комісії *13 травня 2022 р.*
3. Вихідні дані до роботи
Дослідити тенденції розвитку хмарних обчислень, розбрати основні компоненти платформи Openstack та на практиці дослідити інструменти автоматизації її розгортування.

4. Перелік питань, що потрібно опрацювати в роботі
Вступ
1. Види хмар
2. OpenStack та його компоненти
3. Інструменти автоматизації розгортування Openstack
4. Порівняння інструментів автоматизації
Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) *Слайди у форматі Power Point (назва, мета і задачі роботи, види хмар, публічні, приватні, гібридні, гіперконвергентність, OpenStack, компоненти OpenStack, інструменти розгортування OpenStack, OpenStack-Ansible, OpenStack-Chef, висновки)*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів атестаційної роботи	Строк виконання етапів роботи	Примітка
1	*Ознайомлення із завданням. Уточнення ТЗ*	*14.03.22*	*виконано*
2	*Підбір літератури за темою роботи*	*15.03-18.03.22*	*виконано*
3	*Виконання розділу 1*	*19.03-29.03.22*	*виконано*
4	*Виконання розділу 2*	*30.03-09.04.22*	*виконано*
5	*Виконання розділу 3*	*10.04-20.04.22*	*виконано*
6	*Виконання розділу 4*	*21.04-01.05.22*	*виконано*
7	*Виконання розділу 5*	*02.05-08.05.22*	*виконано*
8	*Оформлення пояснювальної записки*	*09.05-11.05.22*	*виконано*
9	*Оформлення презентаційного матеріалу,*	*12.05-13.05.22*	*виконано*
	підготовка до захисту у ЕК		

Дата видачі завдання *14.03.2022 р.*

Студент _____ *Коротіч О.В.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Костромицький А.І.*
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка: 70 с., 4 рис., 2 табл., 16 джерел, 2 додатки.

Об'єкт дослідження – інструменти автоматизації розгортування платформи OpenStack.

Мета роботи – дослідження та порівняння сучасних інструментів автоматизації розгортування платформи OpenStack.

Результати – в роботі було розглянуто ряд питань що стосуються хмарних обчислень, побудови таких платформ та автоматизації їх процесів життєдіяльності, розглянуто види хмар, їх розподіл на ринку, особливості, переваги та недоліки. Було розглянуто тему гіперконвергентності у контексті побудови хмарних рішень. Досліджено проекти автоматизації розгортування та налаштування платформи Openstack з використанням так і таких загально визнаних інструментів автоматизації як Chef, Ansible, Puppet, Saltstack так і спеціалізованих розробок як OpenStack Autopilot та Fuel Openstack. Було приділено увагу детальному розгляду таким проектам як OpenStack-Ansible та OpenStack-Chef. Досліджено їх структуру, функціонал, можливості використання та модифікації. Проведено практичне дослідження із розгортування платформи Openstack за допомогою цих проектів та проведено порівняння за результатами.

OPENSTACK, ANSIBLE, CHEF, PUPPET, SALTSTACK, OPENSTACK AUTOPILOT, FUEL OPENSTACK

THE ABSTRACT

Explanatory note: 70 p., 4 fig., 2 tabl., 16 sources, 2 app.

The object of the study-deployment automation tools for the OpenStack platform.

The purpose of the work is to study and compare modern tools for automation of OpenStack platform deployment.

Results-the work considered a number of issues related to cloud computing, the construction of such platforms and automation of their life processes, considered the types of clouds, their distribution in the market, features, advantages and disadvantages. The topic of hyperconvergence in the context of building cloud solutions was considered. The projects of automation of Openstack platform using both generally accepted automation tools such as Chef, Ansible, Puppet, Saltstack and specialized developments such as OpenStack Autopilot and Fuel Openstack were examined. Attention was paid to such projects as OpenStack-Ansible and OpenStack-Chef. Their structure, functionality, use and modification possibilities were investigated. A practical study on Openstack platform deployment with the help of these projects has been performed and the results have been compared.

OPENSTACK, ANSIBLE, CHEF, PUPPET, SALTSTACK, OPENSTACK AUTOPILOT, FUEL OPENSTACK

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API – Application Programming Interface (інтерфейс програмування додатків);

EBS – Elastic Block Storage;

NFS – Netwoek File System

IT – Information Technologies (інформаційні технології);

SDK – Software Development Kit (комплект програмного забезпечення);

VPN – Virtual Private Netrowk;

ОС – операційна система;

ПЗ – програмне забезпечення;

СУБД – Система Управління Базами Даних.

ВСТУП

У сучасному світі робота в галузі виробництва програмного забезпечення зазвичай зводиться до безперервного розгортання та експлуатації середовища, розподіленого по всьому світу. Сьогодення неможливо уявити без хмарних технологій. Вони знаходять застосування у безлічі неймовірно важливий галузей, таких як: медицина, державне управління, наука та бізнес. Хмарні технології пропонують принципово нові та ефективні можливості для усіх галузей.

Хмарні провайдери пропонують постійний доступ обчислювальних потужностей та надійних сервісів, які мають системи моніторингу, можливість швидкого масштабування, та швидку міграцію даних між сервісами в хмарі та поза хмарою. [1]

З такими могутніми прихильниками в галузі, як Redhat і Rackspace, хмарна технологія OpenStack безумовно є перспективною та буде мати попит в майбутньому. Однак, враховуючи безліч варіантів розгортання OpenStack є велика кількість шляхів для вибору технологій. Вибір технологій залежать від цілей використання та середовища.

OpenStack - це платформа з відкритим кодом, яка дозволяє створювати хмарну інфраструктуру як сервіс (IaaS), що працює на будь-якому обладнанні. Технологія, що лежить в основі OpenStack, складається з серії взаємопов'язаних проектів, що постачають різні компоненти для хмарного інфраструктурного рішення. [1]

OpenStack це повна різноманітна екосистема з міцним фундаментом, тож його не слід вимірювати як звичайне програмне забезпечення. Коли існує загальна базова інфраструктура, цілій галузі набагато легше підключитися до неї і підтримувати її окремо. Це відноситься до всіх рівнів, починаючи з сховища та мережі і закінчуючи більш високорівневими сервісами, такими як сервіси великих даних і аналітика.

Важливою частиною масштабованості хмари є кількість зусиль, необхідних для його запуску. Щоб мінімізувати експлуатаційні витрати на запуск хмари, було створено багато різноманітних інструментів для автоматизації розгортання та конфігурації інфраструктури за допомогою систем управління конфігурацією. Тому ціллю цієї кваліфікаційної роботи є аналіз та порівняння систем управління конфігураціями для розгортання інфраструктури в OpenStack.

Використання автоматизації та хмарних технологій є актуальною темою для галузей розробки та бізнесу тому підвищення надійності для розгортання власних хмарних рішень є важливим етапом у життєвому циклі хмар у тому числі на базі платформи OpenStack.

Метою кваліфікаційної роботи є дослідження тенденцій розвитку хмарних обчислень, розбір основних компонентів платформи Openstack та практичне дослідження інструментів автоматизації її розгортування.

Результатом цього дослідження буде розбір особливостей розвитку, аналіз видів хмар, розгляд структури платформи Openstack, аналіз існуючих інструментів для автоматизації розгортування такої платформи, функціональних можливостей, та широти функціоналу двох таких проектів та їх порівняльний аналіз.

1 ВИДИ ХМАР

Хмара – це зручний та простий спосіб створення інфраструктури компанії без залучення величезних власних потужностей. За останні роки досліджується стійка тенденція у організацій до трансформації методів роботи з обчислювальними ресурсами. Основними рушійними силами таких змін є такі фактори як:

- Масштабованість - дозволяє витримувати навантаження, що збільшуються, без збоїв і втрати даних;
- Оптимізація видатків - оплата оренди сервісу проводиться у міру використання і залежить від розмірів обсягу даних;
- Простота керування - користувач може самостійно керувати необхідними ресурсами, без сторонньої допомоги;
- Еластичність - хмарний сервіс допомагає нарощувати інфраструктуру без використання нового обладнання та додаткового програмного забезпечення.

Хмарні сховища забезпечують неперервний доступ до всієї інфраструктури, оскільки зазвичай використовується низка дублюючих серверів, що забезпечують високу доступність. У разі недоступності одного з них користувача перемикають на інший, внаслідок чого не втрачається необхідний доступ. Виділяють кілька категорій користувачів, котрим використання хмарних сховищ має величезну вигоду, це:

- Відділи розробників - фахівці зможуть швидко розгорнути необхідне середовище розробки;
- Керівництво компаній будь-якого типу- при використанні хмари користувач отримує гнучку IT-інфраструктуру, яку можна адаптувати під будь-які зміни діяльності фірми;
- Підприємства з тестування програмного забезпечення - наприклад, хмарний сервіс дозволить легко клонувати великі бази даних;
- Фінансові підрозділи організацій -при використанні хмари можна гнучко контролювати витрати на IT;

- Підрозділи, які потребують створення віртуальних робочих місць .

Розрізняють декілька видів хмар за особливостями їх розташування на способу обслуговування: публічна, приватна та гібридна (змішана).

Під хмарними обчисленнями розуміється надання користувачеві комп'ютерних ресурсів і потужностей таким чином, що користувач не знає, які комп'ютери обробляють його запити, та під управлінням якої операційної системи це відбувається. [2]

Незалежно від підходів, організації приступають до розгортання приватних хмар з двох основних причин: заради зниження витрат і створення більш гнучкого середовища IT. Першими на цей шлях ступають компанії, які потребують високого ступеня гнучкості інфраструктури і швидкого виділення інформаційних ресурсів. Хмари забезпечують динамічне виділення ресурсів за запитом для різних навантажень, використовують гетерогенну віртуалізовану інфраструктуру і мають високу масштабованість. Автоматизація процесів знижує кількість помилок, спрощує налаштування конфігурації безпеки, мереж і програмного забезпечення. [3]

Згідно CloudTech, очікується, що витрати на публічні хмари зростуть з $229 млрд в 2019 році, до $500 млрд до 2023 року, при цьому очікуваний сукупний річний темп зростання складе 22,3%. У 2021 році основними тенденціями розвитку хмарного ринку є: хмарні безсерверні обчислення; гібридні хмарні рішення; технології контейнеризації та Kubernetes. [4]

Протягом 2020 та 2021 років попит на хмарні обчислення збільшився, оскільки робота стала віртуальною, а підприємства адаптувалися до глобальної пандемії, зосередившись на наданні цифрових послуг; у 2022 році, безсумнівно, відбудеться продовження швидкого впровадження та зростання. [4]

1.1 Публічні хмари

Публічна хмара — це коли віртуальна IT-інфраструктура хмари належить провайдеру і надається компанії-клієнту в оренду. Служби та послуги надають-

ся величезній кількості клієнтів одним власником. Тобто така хмара має одного власника, який надає доступ до віртуального сервісу всім, хто оплатить послугу. З точки зору економіки публічна хмара це сервіс, а саме — віртуальна інфраструктура як сервіс (IaaS). Провайдер виділяє пул віртуальних ресурсів в тому обсязі, в якому вони потрібні компанії-клієнту, яка запускає на них свої додатки. [5]

У таких системах реалізований принцип самообслуговування: користувач може створити необхідну кількість віртуальних серверів або мереж, самостійно управляти наявними даними та розміром сховища, що використовується.

Основні відмінності між публічною та приватною хмарою полягають у зонах відповідальності (управління) інфраструктурою. Розглянемо основні плюси та мінуси кожної із систем віртуального зберігання даних.

Серед переваг таких хмарних сервісів відзначають таке:

- Еластична структура;
- Відносно низька ціна;
- Оптимізація витрат фірми на IT-інфраструктуру;
- Просте та зрозуміле використання.

Серед мінусів зазначають:

- Ризики що пов’язані зі зберіганням та роботою з важливою приватною інформацією;
- Пряму залежність від якості інтернет-з'єднання;
- Ризик втрати даних;
- Використання провайдера неякісного обладнання.

Незважаючи на ризики та недоліки, публічна хмара може стати справжнім порятунком для невеликої компанії із застарілим обладнанням. Заміна техніки стане великою статтею витрат, тоді як проблему недостатніх потужностей можна вирішити за рахунок публічного віртуального сховища.

1.2 Приватні хмари

Приватна хмара - це хмарна інфраструктура, яка належить одній компанії. Вона розгорнута на базі власної фізичної інфраструктури компанії або на орендованому обладнанні. Компанія, яка використовує приватну хмару, не ділить ні з ким фізичні і віртуальні ресурси, вони належать їй. [5]

Власник хмари застосовує її для вирішення власних завдань. Найчастіше така хмара створюється окремою компанією під внутрішньо-корпоративні потреби, при цьому обладнання розташовується на власній території або в ЦОД.

На відміну від публічного, у приватному cloud-сервісі всі ресурси розміщуються на конкретному фізичному сервері.

Приватна хмара більше підходить для великих організацій, які мають складну інфраструктуру і мають необхідні потужності для розгортання віртуального сховища. Головними перевагами такого хмарного сервісу є:

- Ізольованість IT-інфраструктури;
- Надійне зберігання даних;
- Можливість довгострокового вкладення обладнання для підтримки роботи системи;
- Доступна конфігурованість.

Звичайно, такий сервіс має свої недоліки. Серед них:

- Обмежений обсяг потужностей;
- Необхідність тривалої підготовки до розгортання;
- Висока ціна, що включає покупку чи оренду обладнання;
- Необхідність наявності IT-фахівців для адміністрування системи.

При створенні приватної хмари багато компаній стикаються з проблемою безпеки даних. У цьому випадку часто потрібна допомога професіоналів у цій сфері або використання додаткових програмних засобів. [6]

1.3 Гібридні хмари

Гібридні хмари - коли частина інфраструктури розміщена в публічній хмарі провайдера, а частина в приватній хмарі компанії. У більш широкому сенсі: коли власна частина інфраструктури компанії-це навіть не приватна хмара, а звичайна (традиційна, «залізна») інфраструктура, що може не задіювати віртуалізацію. Якщо вона працює в парі з публічною хмарою, то таку IT-інфраструктуру теж можна назвати гібридною хмарою. Наприклад, у гібридній інфраструктурі традиційні локальні системи зберігання даних можуть передавати дані в публічну хмару для обробки. [6]

Поєднання публічної та приватної хмари, в якій реалізується перехресне використання файлів та програм. У такій хмарі застосовується концепція cloudbursting – можливість запобігти перевитраті потужностей приватної хмари завдяки використанню при піковому навантаженні потужностей публічного сервера. Це дозволяє уникнути проблем з використанням віртуального сховища для розробки та тестування нових систем, а також інтеграції B2B-рішень. Важливою ознакою гібридної інфраструктури є застосування елементів, розташованих на різних хмарах, в одній інформаційній системі. [6]

Гібридна хмара часто стає оптимальним рішенням за наявності у компанії потрібного обладнання, але не достатньої кількості. При використанні подібної системи частина ресурсів розміщується у публічній хмарі, інша – у приватній.

Подібне рішення має свої плюси:

- Створення відповідного середовища для тестування та розробки.
- Зниження ймовірності збоїв системи.
- Значне спрощення локальної інфраструктури.
- Велике робоче навантаження із забезпеченням високого рівня захисту.
- Масштабованість інфраструктури.
- Постійний доступ до створених даних.

Незважаючи на перспективність подібних сховищ, існує й низка проблем, пов'язаних з їх використанням та побудовою. Серед основних недоліків можна відзначити таке:

- Високий ризик втрати даних при передачі інформації з приватної до публічної хмари і навпаки.
- Складнощі з виконанням резервного копіювання БД.
- Неможливість відстежити фактичне знаходження даних, розташованих поза приватним сегментом хмари.

1.4 Гіперконвергентність

Гіперконвергентність - це IT-інфраструктура, яка об'єднує сховище, обчислювальні та мережеві ресурси в єдину систему, яка допомагає спростити середовище ЦОД і підвищити масштабованість. Гіперконвергентна платформа включає в себе гіпервізор для віртуалізованих обчислювальних ресурсів, програмно-визначеного сховища і віртуалізації мережі. [7]

Гіперконвергентність - це програмно-орієнтована архітектура, в якій обчислювальні ресурси, сховище і засоби віртуалізації інтегровані в єдину систему. Вона надається у вигляді пристрою на базі серверів x86 або як ПЗ, яке можна встановити на наявному обладнанні. [7]

Гіперконвергентність забезпечує реалізацію всіх основних можливостей ЦОД на тісно інтегрованому програмному рівні, а не на спеціалізованому обладнанні. Гіперконвергентні платформи складаються з трьох програмних компонентів: віртуалізації обчислювальних ресурсів, віртуалізації сховища і засобів управління. ПЗ для віртуалізації абстрагує і об'єднує в пули базові ресурси, а потім динамічно виділяє їх додаткам, які виконуються в ВМ або контейнерах. [8]

Гіперконвергентна інфраструктура об'єднує віртуальні обчислення, програмно-визначені системи зберігання даних і віртуальні мережі на одному сервері. Ці вузли можна швидко масштабувати відповідно до ваших потреб.

Зростаючий попит на хмарні сервіси, а також необхідність заміни застарілого апаратного забезпечення і зниження експлуатаційних витрат змушують компанії впроваджувати гіперконвергентну інфраструктуру в центри обробки даних. За допомогою гіперконвергентної інфраструктури можна виконувати робочі навантаження для обчислень і систем зберігання даних на одному сервері. Сервери групуються у віртуальний кластер ресурсів для обчислень і зберігання даних по стандартній мережі Ethernet. [7]

Гіперконвергенція - це простий спосіб створити приватне або гібридне хмарне середовище. Вона являє собою сучасну, програмно-визначену інфраструктуру з можливістю швидкого розгортання і простого управління. Крім того, вона добре поєднується з інструментами гібридного хмарного середовища. Завдяки цьому забезпечується самообслуговування і портативність щодо стратегічного розміщення робочих навантажень. Простіше кажучи, гіперконвергентна інфраструктура розширює можливості програмно-визначених центрів обробки даних. Вона покращує масштабованість і надає широкі можливості, дозволяючи створити комплексну інфраструктуру, орієнтовану на програмне забезпечення. [7]

Більшість компаній починають з використання трьох-чотирьох вузлів у гіперконвергентному кластері. У міру виникнення потреб в більш високій продуктивності і широких можливостях вони збільшують їх кількість. Ця модульна архітектура з горизонтальним масштабуванням дозволяє з упевненістю створити інфраструктуру центрів обробки даних.

Гіперконвергентна інфраструктура-це перевірена архітектура, що підтримує цілий ряд робочих навантажень. У ній можна налаштувати сервери з урахуванням конкретних робочих навантажень.

Гіперконвергентна інфраструктура дозволяє почати з маленької інфраструктури і поступово розширювати її. Вона забезпечує більш прості умови для управління капітальними витратами і контролю за придбанням надлишкових коштів. Ви можете вкладати кошти з плином часу і використовувати переваги

кожного покоління технологій. Крім того, гіперконвергентною інфраструктурою може управляти IT-фахівець широкого профілю, знайомий з інструментами віртуалізації, що дозволяє скоротити експлуатаційні витрати. [7]

Завдяки гіперконвергентній інфраструктурі дані розподіляються по кластеру, що забезпечує відсутність точок відмови. Розгортайте кілька кластерів для підвищення довговічності і безперервного ведення бізнесу. Якщо дані в одному місці недоступні, їх можна витягти з іншого кластера.

Гіперконвергентна інфраструктура дозволяє почати з маленької інфраструктури і поступово розширювати її. Вона забезпечує більш прості умови для управління капітальними витратами і контролю за придбанням надлишкових коштів. [7]

2 OPENSTACK ТА ЙОГО КОМПОНЕНТИ

2.1 OpenStack

Архітектура OpenStack складається з безлічі проектів з відкритим вихідним кодом. Ці проекти використовуються для налаштування undercloud і overcloud OpenStack, які використовуються системними адміністраторами і користувачами Хмари відповідно. У першій категорії містяться основні компоненти, необхідні системним адміністраторам для налаштування і управління середовищами OpenStack кінцевих користувачів, відомими як overclouds.

Існує 6 основних служб, які обробляють обчислення, мережі, сховище, ідентифікацію та образи систем, в той час як більше 30 додаткових служб дозволяють розширювати функціонал хмари у напрямку оркестрації, автоматизації процесів, моніторингу, алертингу, резервування ресурсів та білінгу. Ці 6 основних сервісів являють собою інфраструктуру, яка дозволяє іншим проектам використовувати та доповнювати її.

Розгортання OpenStack містить ряд компонентів, що надають API для доступу до ресурсів інфраструктури. Далі будуть розглянуті деякі служби, які можуть бути розгорнуті для надання таких ресурсів кінцевим користувачам хмари. Як видно з рисунку 2.1 сервіси розподіляються на такі категорії:

- Обчислювання;
    - Nova - сервіс обчислень;
    - Zun - сервіс контейнеризації;
- Життєвий цикл обладнання;
    - Ironic - сервіс керування фізичними серверами;
    - Cyborg - сервіс керування спеціалізованими пристроями;
- Зберігання даних;
    - Swift - об’єктно орієнтоване сховище;
    - Cinder - блокове сховище;
    - Manila - сервіс розподілених файлових систем;

- Мережа;
    - Neutron - сервіс керування мережами;
    - Octavia - розподілення мережевого навантаження;
    - Designate - DNS сервіс;
- Сервіси загального користування;
    - Keystone - сервіс ідентифікації;
    - Placement - сервіс розподілення ресурсів;
    - Glance - сервіс керування образами систем;
    - Barbican - сервіс керування ключами;
- Оркестрування;
    - Heat - сервіс забезпечення оркестрування;
    - Senlin - сервіс кластеризації;
    - Mistral - сервіс побудови робочих процесів;
    - Zaqar - сервіс забезпечення черг;
    - Blazar - сервіс для резервування ресурсів;
    - Aodh - сервіс для нотифікування про тривоги;
- Підготовка робочого навантаження;
    - Magnum - сервіс оркестрування контейнерів;
    - Sahara - надання інфраструктури обробки великих даних;
    - Trove - база даних як сервіс;
- Життєвий цикл застосунків;
    - Masakari - сервіс забезпечення високої доступності;
    - Murano - каталог застосунків;
    - Solum - сервіс автоматизації життєвого циклу розробки програмного забезпечення;
    - Freezer - сервіс забезпечення резервного копіювання, відновлення та аварійного відновлення;
- Веб-інтерфейси;
    - Horizon - панель керування;
    - Skyline -панель керування наступного покоління (у розробці) [1]

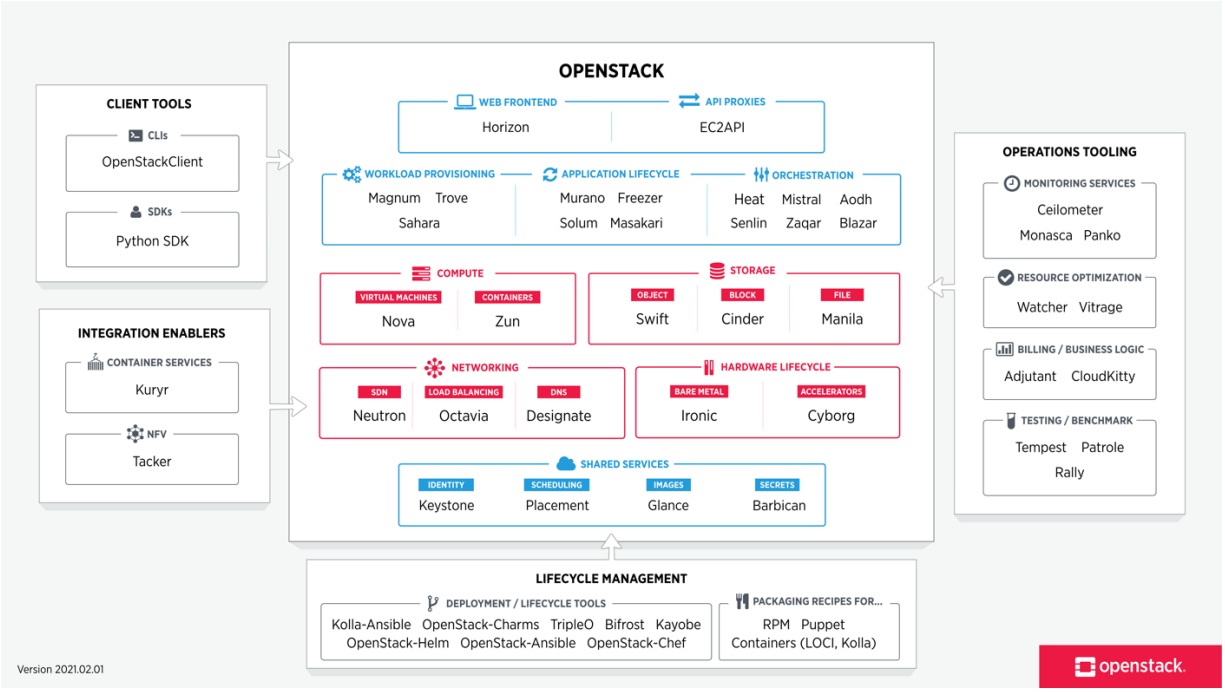

Рисунок 2.1 - The OpenStack Landscape

2.2 Keystone

Хмарні середовища на рівні інфраструктури як сервісу (IaaS) надають користувачам доступ до ключових ресурсів, таким як екземпляри віртуальних машин, великі обсяги зберігання блоків та об’єктів, а також пропускна здатність мережі. Найважливішою особливістю будь хмарного середовища є те, як вона забезпечує безпечний та контрольований доступ до цих цінних ресурсів. У середовищах OpenStack служба Keystone відповідає за забезпечення безпечного контрольованого доступу до всіх ресурсів хмари. Keystone зарекомендував себе як життєво важливий компонент безпечного хмари. Щоб зрозуміти, як Keystone забезпечує безпечний та контрольований доступ до локальних ресурсів OpenStack, необхідно визначити фундаментальні можливості Keystone для забезпечення ідентифікації, аутентифікації і управління доступом, загальна діаграма яких відображена на рисунку 2.2. [9]

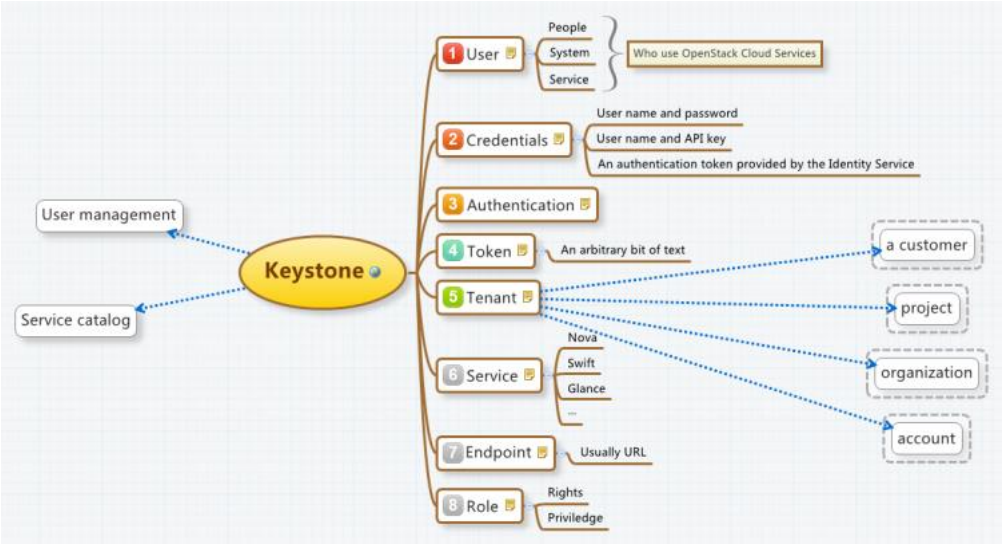

Рисунок 2.2 – Функціонал Keystone

Ідентифікація - ототожнення, прирівнювання, уподібнення, розпізнавання іншої системи або об’єкта за наперед заданими критеріями. У контексті Keystone ідентифікація того, хто намагається отримати доступ до локальних ресурсів. У простих розгортаннях особистість користувача може зберігатися у власній базі даних Keystone. У виробничому або корпоративному середовищі зазвичай використовується зовнішній постачальник посвідчень особи. Прикладом цього може служити IBM Tivoli Federated Identity Manager, Red Hat Keycloak Single Sign-On, тощо. Keystone має можливість отримувати ідентифікаційну інформацію користувача від зовнішніх постачальників ідентифікаційних даних. [9]

Автентифікація - це процес підтвердження особи користувача. У багатьох випадках автентифікація спочатку здійснюється шляхом входу користувача в систему за допомогою ідентифікатора користувача і пароля. У мінімальному середовищі OpenStack Keystone здатний самостійно виконати всі кроки автентифікації, проте це не рекомендується для виробничих або корпоративних середовищ. Для реальних середовищ, де паролі повинні бути належним чином захищені та керовані, Keystone є підключеним, що дозволяє легко інтегрувати його з існуючою перевіреною службою аутентифікації виробництва, такою як LDAP або Active Directory. [9]

Хоча особистість користувача зазвичай спочатку автентифікується за допомогою пароля, дуже часто в рамках цієї первісної аутентифікації створюється токен для подальших автентифікацій. Це зменшує видимість і розкриття пароля, який повинен бути максимально прихований і захищений. Токени також мають обмежений термін служби і закінчуються, тому їх корисність обмежена в разі крадіжки. OpenStack в значній мірі покладається на токени для аутентифікації та інших цілей і Keystone є єдиним сервісом OpenStack, який може їх видавати. В даний час Keystone використовує форму токена, звану токеном на пред'явника. Це означає, що хто б не отримав право власності на токен, будь то належним або неналежним чином (тобто вкрадений), він може використовувати його для аутентифікації і доступу до ресурсів. Тому при використанні Keystone дуже важливо захистити токени і їх вміст. Управління доступом, також зване авторизацією, являє собою процес визначення того, до яких ресурсів дозволений доступ користувачеві. Хмарні середовища, такі як OpenStack, надають користувачам доступ до великої кількості ресурсів. Наприклад, необхідний механізм визначення того, яким користувачам дозволено створювати нові екземпляри певної віртуальної машини, яким користувачам дозволено приєднувати або видаляти томи блочного сховища, яким користувачам дозволено створювати віртуальні мережі і т.д. в OpenStack Keystone прив'язує користувачів до проектів або доменів, асоціюючи роль користувача з проектом або доменом. Інші підпроекти OpenStack, такі як Nova, Cinder і Neutron, вивчають асоціації проекту і ролі користувача і оцінюють цю інформацію за допомогою механізму політик. Механізм політики вивчає цю інформацію (зокрема, значення ролі) і визначає, які дії дозволено виконувати користувачеві. [9]

Основні переваги Keystone

Хоча Keystone в основному орієнтований на забезпечення управління ідентифікацією, аутентифікацією та доступом, він надає велику кількість переваг для середовища OpenStack. До основних переваг відносяться наступні: - Єдиний інтерфейс аутентифікації та управління доступом для інших сервісів OpenStack. Keystone вирішує складні завдання інтеграції із зовнішніми системами аутентифікації, а також забезпечує єдине управління доступом для всіх інших сервісів OpenStack, таких як Nova, Glance, Cinder, Neutron і т.д., і таким чином Keystone ізолює всі інші сервіси від необхідності спілкуватися з різними провайдерами ідентифікації та авторизації. Основні можливості які надає Keystone:

- Реєстр контейнерів (“Projects”), які інші сервіси OpenStack можуть використовувати для поділу ресурсів (наприклад, серверів, образів і т.д.);
- Реєстр доменів, які використовуються для визначення окремих просторів імен для користувачів, груп і проектів, щоб забезпечити поділ між клієнтами;
- Реєстр ролей, які будуть використовуватися для авторизації між Keystone і файлами політик кожного з сервісів OpenStack;
- Сховище призначень, що дозволяє призначати користувачам і групам ролі в проектах і доменах;
- Каталог, що зберігає сервіси OpenStack, кінцеві точки і регіони, що дозволяє клієнтам знаходити потрібний їм сервіс або кінцеву точку. [10]

2.3 Nova

OpenStack Compute Service (Nova) - це контролер екземплярів хмарних обчислень, який є основною частиною системи IaaS. Nova - це Проект OpenStack, що реалізує спосіб надання обчислювальних екземплярів (вони ж віртуальні сервери), який використовується для розміщення та управління хмарними обчислювальними системами. Він забезпечує повний контроль над обчислювальними ресурсами і дозволяє працювати в обчислювальному середо-

вищі OpenStack. Це скорочує час, необхідний для отримання та завантаження нових екземплярів сервера, до декількох хвилин, дозволяючи швидко масштабувати потужності, як у бік збільшення, так і зменшення, у міру зміни ваших вимог. Nova працює як набір демонів поверх існуючих серверів Linux (обчислювальний вузол) для надання цієї послуги, у які входять такі сервіси як nova-api, nova-conductor, nova-scheduler та nova-compute. Їх зв'язки відображено на рисунку 2.3 На обчислювальному вузлі працює гіпервізорна частина Compute, яка управляє екземплярами. За замовчуванням Compute використовує гіпервізор VM (KVM) на базі ядра. На обчислювальному вузлі також працює агент служби Networking, який підключає екземпляри до віртуальних мереж і надає послуги міжмережевого екранування для екземплярів за допомогою груп безпеки. Кінцеві користувачі nova, можуть використовувати nova для створення та управління серверами як за допомогою інструментів та і через API безпосередньо. [11]

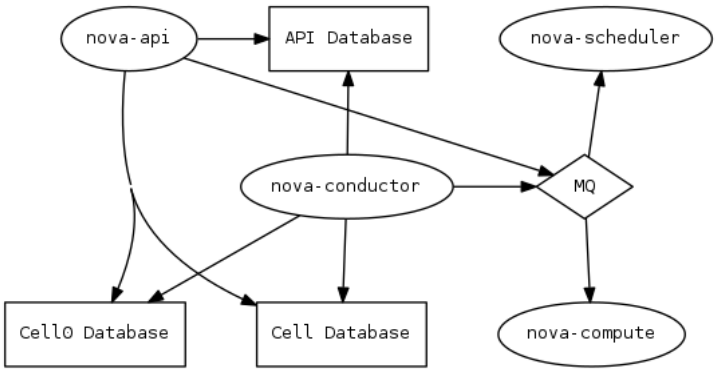

Рисунок 2.3 – Сервіси Nova

2.4 Neutron

Neutron-це Проект OpenStack для надання “Мережевих підключень як Послуги (NaaS)”. Сервіс OpenStack Networking (Neutron) надає API, який дозволяє користувачам будувати багаті мережеві топології, встановлювати і визначати мережеві підключення, налаштовувати розширені мережеві політики і адресацію в хмарі. OpenStack Networking управляє створенням і управлінням вірту-

альною мережевою інфраструктурою, включаючи мережі, Комутатори, підмережі і маршрутизатори для пристроїв, керованих сервісом OpenStack Compute (nova). Як зображено на рисунку 2.4, сервіс складається із серверу та різних агентів, взаємодія між ними відбувається через систему обміну повідомленнями. Також можуть використовуватися розширені сервіси, такі як брандмауери або віртуальні приватні мережі (VPN), але в даний час вони недоступні в цій хмарі. OpenStack Networking інтегрується з різними компонентами OpenStack: - Сервіс OpenStack Identity (Keystone) використовується для аутентифікації та авторизації запитів API. - Сервіс OpenStack Compute (nova) використовується для підключення кожної віртуальної мережевої карти на ВМ до певної мережі. - OpenStack Dashboard (Horizon) використовується адміністраторами і користувачами проекту для створення і управління мережевими сервісами через графічний веб-інтерфейс. [12]

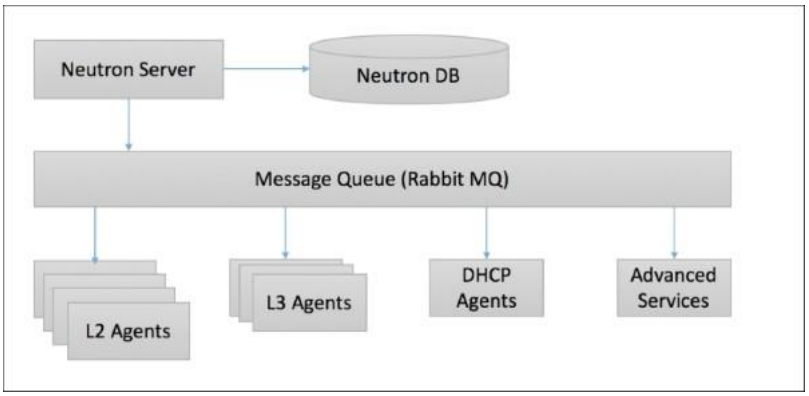

Рисунок 2.4 – Сервіси Neutron

2.5 Cinder

Cinder - це служба блочного сховища для OpenStack, що надає “блочне сховище як послугу”. Він призначений для надання кінцевим користувачам ресурсів зберігання, які можуть бути використані OpenStack Compute Project (Nova).

Короткий опис Cinder полягає в тому, що він віртуалізує управління блоковими пристроями зберігання і надає кінцевим користувачам API самообслу-

говування для запиту і споживання цих ресурсів, не вимагаючи ніяких знань про те, де насправді розгорнуто їх сховище або на якому типі пристрою. [13]

Служба OpenStack Block Storage (Cinder) додає постійне сховище до віртуальної машини. Block Storage надає інфраструктуру для управління томами і взаємодіє з OpenStack Compute для надання томів для екземплярів. Служба також дозволяє управляти моментальними знімками томів і типами томів. [13]

Служба OpenStack Block Storage надає постійні ресурси блочного сховища, які можуть споживати екземпляри OpenStack Compute. Сюди входить Вторинне підключене сховище, аналогічне пропозиції Amazon Elastic Block Storage (EBS). Крім того, ви можете записувати образи на пристрій Block Storage для використання Compute в якості завантажуваного постійного екземпляра. Однак служба Block Storage не надає рішення для загального зберігання, як NFS. За допомогою служби Block Storage можна прикріпити пристрій тільки до одного примірника. Доступні засоби управління квотами для обмеження кількості створюваних томів, кількості моментальних знімків і загальної кількості ГБ, дозволених для орендаря (розподіляються між моментальними знімками і томами). [13]

Доступ користувачів до певних томів обмежений по орендарям, але ім’я користувача і пароль призначаються кожному користувачеві. Пари ключів, що надають доступ до того, дозволені для кожного користувача, але квоти для контролю споживання ресурсів за наявними апаратними ресурсами призначаються на кожного орендаря.

Основними ресурсами, пропонованими службою Block Storage, є томи і моментальні знімки, які створюються з томів і резервних копій томів:

- Том - виділений ресурс блочного сховища, який може бути підключений до екземплярів в якості вторинного сховища або може використовуватися в якості кореневого сховища для завантаження екземплярів;
- Моментальний знімок - копія тома, доступна тільки для читання. Знімок може бути створений з диску, який в даний час використовується. Знімок мож-

на використовувати для створення нового тому за допомогою команди create from snapshot;

- Резервна копія - архівна копія томи, що зберігається в даний час в OpenStack Object Storage (Swift). [13]

2.6 Glance

OpenStack Glance-це служба образів, яка забезпечує швидкий і зручний спосіб копіювання і запуску екземплярів. За допомогою Glance користувачі можуть завантажувати, виявляти, реєструвати і витягувати образи віртуальних машин з швидкістю і легкістю. Glance забезпечує безпечне завантаження і безпечне вивантаження завдяки перевірці зображень з підписом. Це означає, що дані можуть бути перевірені через Glance перед збереженням у вашій хмарі. Тому, якщо перевірка не пройшла, завантаження буде невдалою, а зображення буде видалено. Те ж саме стосується всіх завантажень образів. Якщо дані не можуть виконати відповідну перевірку при завантаженні зображення, то воно не буде зберігатися у вашому хмарі. Що стосується сумісності, Glance не обмежується конкретними серверами, оскільки він може завантажувати віртуальні машини поряд з Cinder і Ironic. Завдяки RESTful API Glance можливий запит метаданих образу віртуальної машини, а також отримання фактичного образу. Нарешті, однією з переваг передових технологій OpenStack є проста інтеграція Glance з блоковим сховищем Cinder в рамках інфраструктури. Це дозволяє забезпечити зберігання і просту у використанні віртуалізацію управління блоковим сховищем. [14]

2.7 Horizon

Horizon-це реалізація панелі OpenStack, що надає користувальницький веб-інтерфейс для таких сервісів OpenStack, як Nova і Keystone та інших. Почавши як єдиний додаток для управління обчислювальним проектом OpenStack, Horizon виріс до підтримки безлічі проектів і API OpenStack. Управління середовищем OpenStack через інтерфейс командного рядка дозволяє повністю кон-

тролювати хмарне середовище, але наявність веб-інтерфейсу, який оператори та адміністратори можуть використовувати для управління середовищем та екземплярами, спрощує цей процес. Ресурси в хмарі організовані як проекти і можуть також називатися підписками або обліковими записами. Користувачі в Horizon можуть бути членами одного або декількох проектів, а в рамках проекту користувач може створювати і управляти екземплярами (або віртуальними машинами). Нижче наведено короткий огляд налаштувань і вкладок на панелі інструментів. Після входу в OpenStack і панель Horizon ви побачите основну вкладку проекти в лівому кутку. Елементи вкладки Projects дозволяють переглядати і управляти ресурсами в обраному проекті, такими як екземпляри і образи. Також є можливість перемикатися між проектами за допомогою випадаючого меню в лівому верхньому кутку. Horizon-це веб-служба, яка запускається з установки Apache з використанням інтерфейсу шлюзу веб-служб Python (WSGI) і Django. [9]

3 ІНСТРУМЕНТИ АВТОМАТИЗАЦІЇ РОЗГОРТУВАННЯ OPENSTACK

Технології DevOps змінили спосіб ведення бізнесу організаціями та підприємствами, заснованими на IT, завдяки чому все стало масштабованим, гнучким, а продуктивність систем кратно збільшилася. Використання підходу IaC, може здійснювати ефективне управління та надання даних на віртуальних серверах, незважаючи на використання фізичних центрів обробки даних та інфраструктури на базі IT. Оскільки IaC стає новим стандартам передової практики IT, інфраструктури переходять від традиційних режимів роботи до більш незмінних. Це пов'язано з тим, що IT-відділи прагнуть досягти безперервної доставки, з версіонуванням і автоматизованим тестуванням, вбудованим в процес DevOps. Мета цього полягає в тому, щоб IT-відділ послідовно розгортав пакет і його залежності, кожен раз створюючи ідентичне оточення.

Завдяки автоматизованим процесам IaC допомагає підприємствам керувати потребами IT-інфраструктури кількома способами. Нижче перераховані деякі переваги впровадження IaC:

- Послідовність: IaC може поліпшити узгодженість і зменшити кількість помилок, які часто виникають при ручному налаштуванні. Вона також усуває будь-яке зміщення конфігурації, яке може статися під час ручного процесу. Завдяки кодифікації та документуванню специфікацій конфігурації, IaC допомагає уникнути недокументованих, ситуативних змін конфігурації;
- Зниження витрат: IaC дозволяє програмно управляти віртуальними машинами, усуваючи необхідність в ручному налаштуванні і оновленні обладнання. Один оператор може розгорнути і керувати однією машиною або 1000 машин, використовуючи один і той же набір коду. Це означає, що потрібно менше співробітників і відпадає необхідність в покупці нового обладнання, що значно знижує витрати;

- Ефективність: кодифікація інфраструктури дає вам шаблон для ініціалізації, що спрощує конфігурацію системи, обслуговування та управління. Вона створює еластичну інфраструктуру, яка є повторюваною і масштабованою. Це означає, що DevOps може прискорити кожен етап розробки програмного забезпечення, що призведе до щоденного випуску більшої кількості додатків;

- Швидкість: IaC перетворює трудомістку роботу розробників з ініціалізації в просте виконання сценарію, щоб інфраструктура була готова. В результаті розгортання додатків більше не потрібно чекати інфраструктуру, і нове програмне забезпечення можна випускати набагато швидше;

- Зниження ризиків: IaC також підтримує контроль версій, тому ваші конфігураційні файли можуть потрапити під контроль вихідного коду, як і будь-який інший файл вихідного коду програмного забезпечення. Таким чином, знижується ризик. [15]

IAC зазвичай має дві моделі, push і pull. Основна відмінність між цими двома методами полягає в тому, як конфігуруються сервери. У методі pull, конфігурується сервер отримує свою конфігурацію від керуючого сервера, в той час як в методі push, керуючий сервер передає конфігурацію цільовій системі.

3.1 Chef

Chef - це платформа автоматизації, яка дозволяє легко розгортати сервери і додатки у будь-якому фізичному або віртуальному хмарному місцезнаходження, незалежно від розміру інфраструктури. Як випливає з вищесказаного, Chef - це продукт, орієнтований на свою власну базу розробників.

Основною категорією розмежування ресурсів є організація. Кожна організація складається з однієї (або декількох) робочих станцій, одного сервера і кожного ресурсу, який буде налаштовуватися і обслуговуватися шеф-клієнтом. Кулінарні книги (і рецепти) використовуються для того, щоб повідомити шеф-клієнту, як повинен бути налаштований кожен вузол у організації. Крім того,

оскільки Chef зосереджений навколо Git, він забезпечує відмінні можливості керування версіями.

Існує кілька варіантів конфігурації, включаючи блочне сховище, гіпервізори, бази даних, чергу повідомлень, мережа, сховище об'єктів, вихідні складання і так далі. Поточні підтримувані сценарії розгортання включають All-in-One Compute, Single Controller + N Compute і Vagrant.

Chef має агентну архітектуру, яка вимагає наявності клієнта на кожній віртуальній машині або сервері, керованого центральним головним агентом. Шеф-кухар також користується великою підтримкою у вигляді великої кількості кулінарних книг і документації.

Chef використовує Ruby в якості мови для розширень, тому Ruby є необхідною умовою для використання Chef. Одним з головних недоліків цієї системи є те, що вона не підтримує функцію push. Цей інструмент автоматизації та керування IT добре підходить для інфраструктурних проектів, орієнтованих на розвиток.

3.2 Ansible

Ansible - ще один популярний інструмент автоматизації інфраструктури. Він може налаштовувати системи, розгортати програмне забезпечення та організовувати більш складні IT-завдання, такі як безперервне розгортання або оновлення з нульовим часом простою.

Основними цілями Ansible є простота і зручність використання. Він також приділяє велику увагу безпеці і надійності, показуючи мінімум рухомих частин, використання OpenSSH для транспортування (з прискореним режимом сокета і режимами витягування в якості альтернативи), а також мову розширень, яка розроблена для прощення роботи людям – навіть тим, хто не знайомий з програмою.

OpenStack-Ansible - це офіційний проект OpenStack, метою якого є розгортання виробничих середовищ з вихідного коду таким чином, щоб зробити їх масштабованими, а також простими в експлуатації, оновлення і розвиток.

Ще однією важливою перевагою використання Ansible є його безагентна архітектура. Завдяки цій властивості немає необхідності заздалегідь налаштовувати примірники віртуальної машини або робочої станції, Ansible може працювати безпосередньо з ними з командного рядка. Крім того, користувальницький модуль може бути будь-якою мовою, якщо виводиться у форматі JSON.

Однак робота з Ansible є і серйозний недолік - обмежена підтримка користувачів. Крім того, він ефективно не підтримується на всіх ОС.

Ansible, з його простим розгортанням і нескладними шляхами відтворення, привертає системних адміністраторів для керування системної та хмарної інфраструктурою.

3.3 Puppet

Puppet Enterprise допоможе почати роботу з OpenStack з першого дня і масштабувати хмару у міру зростання потреб організації. Він найбільш повний з точки зору доступних дій, модулів і користувальницьких інтерфейсів. Puppet більше схожий на Chef, ніж на Ansible у багатьох аспектах. Puppet також дотримується агентської архітектури і більше орієнтований на проекти, що орієнтовані на розробку.

Всі модулі і конфігурації створюються з використанням специфічного для Puppet мови, заснованого на Ruby, або самого Ruby, і тому в додаток до навичок системного адміністрування потрібні знання програмування.

За допомогою проекту OpenStack Puppet Modules, підтримуваного співтовариством OpenStack, є можливість використовувати Puppet Enterprise для розгортання і налаштування самих компонентів OpenStack, таких як Nova, Keystone, Glance та Swift.

Puppet Enterprise дозволяє в режимі реального часу управляти керованими вузлами за допомогою готових модулів і наборів інструкцій, присутніх на головних серверах. Інструменти звітності добре розроблені і містять детальну інформацію про те, як ведуть себе агенти і які зміни були внесені.

3.4 Saltstack

Saltstack, новий підхід до управління інфраструктурою, досить проста для запуску за лічені хвилини, досить масштабована, щоб управляти десятками тисяч серверів, і досить швидка, щоб спілкуватися з цими серверами за лічені секунди. Він може бути встановлений через Git або через систему управління пакетами на майстрів і клієнтів. Клієнти роблять запит на головний сервер, який, будучи прийнятий на головному сервері, дозволяє управляти цим міньйоном.

Система Salt проста і зручний у налаштуванні. Кожен з двох компонентів системи Salt має відповідний конфігураційний файл. Salt-master налаштовується через файл конфігурації майстра, а Salt-міньйон - через файл конфігурації міньйонам.

Як і у випадку з Ansible, ви можете видавати міньйонам команди безпосередньо з CLI, наприклад для запуску служб або установки пакетів, або використовувати конфігураційні файли YAML, звані "станами", для обробки більш складних завдань. Існують також "стовпи", які являють собою централізовано розташовані набори даних, до яких держави можуть отримати доступ під час роботи.

Ви можете запросити інформацію про конфігурації, таку як версія ядра або відомості про мережевому інтерфейсі, у міньйонів безпосередньо з CLI. Міньйони можуть бути виокремлені з допомогою елементів інвентарю, званими "зернами", що дозволяє легко видавати команди певного типу сервера, не покладаючись на налаштовані групи. Наприклад, в одному напрямку CLI ви можете націлити на кожного міньйона, що працює під управлінням певної версії ядра.

Користувацькі модулі можуть бути написані на Python або PyDSL. Salt пропонує управління Windows та Unix, але більше підходить для систем Unix та Linux.

Найбільшою перевагою Salt є його масштабованість і відмовостійкість. Ви можете мати кілька рівнів майстрів, що призводить до багаторівневого розташування, яке одночасно розподіляє навантаження і збільшує надмірність. Вищі майстри можуть контролювати нижчих майстрів та їх міньйонів. Ще однією перевагою є система пірингу, яка дозволяє міньйонам задавати питання майстрам, які потім можуть отримувати відповіді від інших серверів, щоб доповнити картину поточної конфігурації. Це може бути зручно, якщо для завершення налаштування міньйона необхідно знайти дані в базі даних реального часу.

3.5 OpenStack Autopilot

OpenStack Autopilot - це один з найпростіших способів створення приватного хмари OpenStack і управління ним. Цей установник OpenStack пропонує на вибір декілька технологій зберігання даних і SDN, високопродуктивну хмарну архітектуру, високі можливості масштабування і підтримку основних серверів постачальників. Autopilot працює дуже простим способом, який включає в себе використання MAAS для реєстрації машин, вибору конфігурації OpenStack і додавання обладнання.

Metal as a Service – MAAS – дозволяє розглядати фізичні сервери як віртуальні машини в хмарі. Замість того щоб керувати кожним сервером окремо, MAAS перетворює фізичний сервер в еластичний хмарний ресурс.

3.6 Fuel OpenStack

Fuel-це інструмент розгортання і управління з відкритим вихідним кодом для OpenStack. Розроблений співтовариством OpenStack, він забезпечує інтуї-

тивно зрозумілий графічний інтерфейс для розгортання і управління OpenStack, пов'язаними з ним громадськими проектами і плагінами.

Fuel поставляється з декількома готовими конфігураціями розгортання, які можна використовувати для негайного створення власної хмарної інфраструктури OpenStack.

Конфігурація для одного вузла встановить весь кластер OpenStack на одній фізичній або віртуальній машини.

Багатовузлова конфігурація кластера дозволяє використовувати додаткові сервіси OpenStack, такі як Cinder, Quantum та Swift, не вимагаючи збільшення кількості апаратних засобів, необхідних для забезпечення високої доступності.

Багатовузловий кластер (з резервуванням) створить кластер OpenStack, що забезпечує високу доступність, з трьома вузлами контролера і можливістю індивідуального завдання служб, таких як Cinder, Quantum та Swift. [16]

З одного боку, Puppet і Chef сподобаються розробникам і орієнтованим на розробку, Salt і Ansible набагато більше пристосовані до потреб системних адміністраторів. Простий інтерфейс і зручність використання Ansible ідеально вписуються в мислення системного адміністратора, а в середах з великою кількістю Linux і Unix-систем Ansible можна швидко і легко запустити без попередніх налаштувань.

Salt – найбільш функціональний і надійний з чотирьох, і, як і Ansible, він буде корисним системним адміністраторам. Проте недоліком Salt веб – інтерфейс та дещо ускладнена для використання багатокомпонентна модель.

Puppet-найбільш зріла і, ймовірно, найбільш доступна з точки зору зручності використання, хоча попередньо рекомендується добре розбиратися в Ruby. Puppet не така гнучка, як Ansible або Salt, і її конфігурація іноді може стати перевантаженою. Puppet-найбезпечніший вибір для гетерогенних середовищ, але ви можете виявити, що Ansible або Salt краще підходять для більш великої або однорідної інфраструктури.

Chef має стабільну і добре продумане компонування, і хоча вона не зовсім відповідає рівню Puppet з точки зору реалізованих функцій, це дуже потужне рішення. Chef може представляти собою один із найскладніших інструментів для адміністраторів, які не мають значного досвіду програмування, але він може бути найбільш придатним для адміністраторів, орієнтованих на розробку.

Fuel і Autopilot - відносно нові інструменти, які, хоча і побудовані виключно з урахуванням хмарної структури OpenStack, не так досконалі, як інші інструменти, згадані вище.

З урахуванням наведених вище висновків у подальшій роботі будуть детально розглядатися та порівнюватися Ansible та Chef.

4 ПОРІВНЯННЯ ІНСТРУМЕНТІВ АВТОМАТИЗАЦІЇ

4.1 Використання OpenStack-Ansible

OpenStack-Ansible (OSA) - це офіційний модульний проект для деплою компонентів Openstack, що складається з низки репозиторіїв у яких знаходяться плейбуки, ролі, шаблони налаштувань та скрипти для спрощення розгортування. Розробка ведеться шляхом верифікації змін через Gerrit Code Review та внесення нового функціоналу до master гілок з наступним перенесенням покращень у останню стабільну гілку що зараз знаходиться у розробці. У випадку внесення виправлень зміни також будуть перенесені у стабільні гілки попередніх релізів, які ще підтримуються. Для кожного випуску OSA ролі Ansible, що складають цей випуск, встановлюються на певний хеш Git SHA - 1 (SHA). Вони оновлюються після кожного релізу OSA. OSA часто робить проактивні бекпорти виправлень. Щоб знизити ризик того, що ці бекпорти внесуть дестабілізацію, OSA реалізує період “витримки” для будь-яких патчів, впроваджених в стабільні гілки ролей, але також передбачає можливість обійти це у виняткових випадках. Патч, злитий в роль, негайно тестується іншими тестами ролі, що гарантує, що будь-яка велика руйнівна зміна буде відловлено. Як тільки запитується мінорний патч або реліз, інтегрована збірка отримує патч ‘SHA bump’ для оновлення інтегрованої збірки до використання останніх доступних ролей, включаючи новий патч. Цей новий набір доступний для тестування всім, хто хоче використовувати голову стабільної гілки, і перевіряється в періодичних тестах до наступного релізу. В цілому, це означає, що час циклу патча від злиття до випуску становить від двох тижнів до місяця. Якщо виникає необхідність в терміновому внесенні патча ролі в наступний реліз, то будь-хто може запропонувати зміну у файлі ansible-role-requirements.yml в репозиторії openstack/openstack-ansible з відповідним обґрунтуванням.

Таблиця 4.1 - Версії Openstack що зараз підтримуються:

Версія	**Статус Ansible**	**Статус Openstack**
Zed		У розробці
Yoga	У розробці	Підтримка
Xena	Підтримка	Підтримка
Wallaby	Підтримка	Підтримка
Victoria	Підтримка	Розширена підтримка
Ussuri	Розширена підтримка	Розширена підтримка
Train	Розширена підтримка	Розширена підтримка
Stein	Розширена підтримка	Розширена підтримка
Rocky	Розширена підтримка	Розширена підтримка
Queens	Розширена підтримка	Розширена підтримка
Pike	Розширена підтримка	Розширена підтримка
Ocata	Розширена підтримка	Не підтримується
Newton	Не підтримується	Не підтримується
Mitaka	Не підтримується	Не підтримується

Як видно із таблиці 4.1, статус розробки Ansible плейбуків відстає на один реліз від статусу розробки Openstack платформи. Тут і далі буде розглядатися останній стабільний реліз Ansible плейбуків - Xena (гілка - stable/xena). Основні релізи випускаються кожні шість місяців відповідно до графіка випуску релізів OpenStack. Мінорні та патчі-релізи запитуються для стабільних гілок у другу і останню п’ятницю кожного місяця. Релізи зазвичай завершуються протягом декількох днів після запиту.

OpenStack-Ansible розгортає контейнери Linux (LXC) і використовує міст Linux між контейнером і хост-інтерфейсами, щоб гарантувати, що весь трафік з контейнерів проходить через кілька хост-інтерфейсів. У цьому додатку описується, як з’єднуються інтерфейси і як проходить трафік. Використання контейнерів дозволяє виконувати додаткові Абстракції цілих стеків додатків на одних

і тих же фізичних хост-комп’ютерах. Контейнерні додатки іноді групуються в одному контейнері, де це має сенс, або розподіляються по декількох контейнерах в залежності від додатків і/або архітектурних потреб. Архітектура контейнера за замовчуванням побудована таким чином, щоб забезпечити масштабованість і високу доступність розгортань.

Проста природа машинних контейнерів дозволяє розробнику розглядати контейнери як фізичні машини. Ті самі концепції застосовуються до машин контейнерів та фізичних машин: це дозволить використовувати існуючі набори операційних інструментів для вирішення проблем у розгортанні та можливість повернути додаток або службу в межах інвентаризації до відомого робочого стану без необхідності повторного запуску. LXC контейнери були обрані через їх практичність щодо забезпечення більш рівномірного розгортання OpenStack. Навіть якщо абстракції, надані контейнерами, покращують загальну безпеку розгортання, ці потенційні переваги не є метою контейнеризації сервісів. Не всі сервіси контейнеризовані: деякі не мають сенсу запускати в контейнері. Логіка повинна застосовуватися щодо того, як сервіси контейнеризуються. Якщо їх вимоги не можуть бути виконані через системні обмеження (ядро, зрілість програми і т.д.), то служба не налаштована для роботи в контейнері. Приклад неконтейнерних сервісів: - Nova Compute (для прямого доступу до пристроїв віртуалізації) - Swift storage (для прямого доступу до диска)

Основним репозиторієм що містить плейбуки, конфігураційні файли та скрипти є openstack-ansible. Посилання на ролі що використовуються у цих плейбуках задані у файлі ansible-role-requirements.yml. Ці посилання ведуть на репозиторії у форматі openstack-ansible-{назва компоненту}.

Типова роль Ansible слідує певній структурі каталогів, яка зазвичай складається з наступних каталогів:

- defaults - містить змінні за замовчуванням для ролі, які повинні бути легко перезаписані;

- vars - містить стандартні змінні для ролі, які не повинні бути перезаписані;

- tasks - містить набір завдань, які повинна виконувати роль;

- handlers - містить набір обробників, які будуть використовуватися в ролі;

- templates - містить шаблони Jinja2, які будуть використовуватися в ролі;

- files - містить статичні файли, необхідні з рольових завдань;

- tests - може містити додатковий файл інвентарю, а також playbook test.yml, який можна використовувати для тестування ролі;

- meta - містить метадані ролі, такі як інформація про автора, Ліцензія, залежність тощо.

Для цілей дослідження будемо проводити установку в режимі All-in-one (AIO). Абсолютний мінімум ресурсів сервера:

- - 8 vCPU;

- - 50 GB вільного дискового простору на кореневому розділі;

- - 8 GB RAM.

Рекомендовані ресурси сервера:

- - Процесор / материнська плата, що підтримує апаратну віртуалізацію;

- - 8 vCPU;

- - 80 GB вільного дискового простору на кореневому розділі або 60 ГБ+ на порожньому вторинному диску;

- - 16GB RAM.

Виходячи із цих рекомендацій тестування буде проводитись на віртуальній машині з рекомендованими виділеними ресурсами та операційною системою Ubuntu Server 20.04 LTS Focal Fossa. Запуск збірки AIO згідно офіційної документації складається з чотирьох етапів: - Підготовка хоста - Завантаження Ansible та необхідних ролей - Підготовка конфігурації AIO - Запуск плейбуків.

Процес підготовки до установки AIO на новому сервері включає оновлення всіх системних пакетів, та перезавантаження в нове ядро.

Процес завантаження Ansible та необхідних ролей виконується у два етапи - клонування репозиторію openstack-ansible та запуск скрипта bootstrap-ansible.sh, що встановить усі необхідні системні пакети та Ansible з його залежностями.

Підготовка конфігурації AIO виконується у декілька етапів. Перший з них це об’ява змінних оточення що відповідають за те, які функції установки будуть задіяні, такі як BOOTSTRAP_OPTS та SCENARIO. Наступний необов'язковий етап це копіювання файлів зі змінними що описують параметри конфігурації для сервісів та їх корегування у випадку якщо значення за замовчуванням не підходять. Наступний етап це запуск скрипту bootstrap-aio.sh який викликає плейбук bootstrap-aio що виконає підготовку хоста для подальшої установки, як то розбиття диску, створення мережевих мостів та налаштування мережевих адрес, тощо.

Останньою частиною є послідовний запуск плейбуків setup-hosts.yml, setup-infrastructure.yml та setup-openstack.yml які виконують такі налаштування як генерацію та встановлення сертифікатів, конфігурацію LXC хоста та контейнерів, установку і налаштування програм залежностей, таких як haproxy, memcached, galera кластер, rabbitmq, etcd, ceph та встановлення і налаштування компонентів OpenStack безпосередньо. Після виконання цих етапів можна отримати налаштовану OpenStack платформу що складається з одного хоста.

4.2 Використання OpenStack-Chef

OpenStack-Chef (OSC) - це офіційний проект для деплою компонентів Openstack, що складається з низки репозиторіїв у яких знаходяться кукбуки, рецепти, атрибути, ресурси та шаблони потрібні для розгортування Openstack за допомогу Chef Infra. Цей проект сфокусований на підтримці основних компонентів платформи. Розробка ведеться шляхом верифікації змін через Gerrit Code Review та внесення нового функціоналу до master гілок з наступним перенесенням покращень у останню стабільну гілку що зараз знаходиться у розробці. У

випадку внесення виправлень зміни також будуть перенесені у стабільні гілки попередніх релізів, які ще підтримуються. На поточний момент підтримуються такі компоненти як Ironic, Cinder, Nova, Horizon, Designate, Glance, Neutron, Heat, Ceilometer, Gnocchi та кукбуки що задовольняють внутрішні потреби, такі як база даних та RabbitMQ.

Таблиця 4.1 - Версії Openstack що зараз підтримуються:

Версія	**Статус Chef**	**Статус Openstack**
Zed		У розробці
Yoga		Підтримка
Xena		Підтримка
Wallaby		Підтримка
Victoria		Розширена підтримка
Ussuri	У розробці	Розширена підтримка
Train	Підтримка	Розширена підтримка
Stein	Підтримка	Розширена підтримка
Rocky	Підтримка	Розширена підтримка
Queens	Підтримка	Розширена підтримка
Pike	Розширена підтримка	Розширена підтримка
Ocata	Розширена підтримка	Не підтримується
Newton	Не підтримується	Не підтримується
Mitaka	Не підтримується	Не підтримується

Як видно із таблиці 4.2, розробка Chef кукбуків знаходиться у стані підтримки версій Openstack платформи що мають розширену підтримку. Тут і далі буде розглядатися останній стабільний реліз Chef кукбуків - Train (гілка - stable/train). Заявлена підтримка операційних систем Ubuntu 18.04 LTS (Bionic Beaver), CentOS 7 та Stream 8.

Для цілей дослідження будемо проводити установку в режимі All-in-one (AIO). Абсолютний мінімум ресурсів сервера:

- - 8 vCPU;

- - 50 GB вільного дискового простору на кореневому розділі;

- - 8 GB RAM.

Рекомендовані ресурси сервера:

- - Процесор / материнська плата, що підтримує апаратну віртуалізацію;

- - 8 vCPU;

- - 80 GB вільного дискового простору на кореневому розділі або 60 ГБ+ на порожньому вторинному диску;

- - 16GB RAM.

Виходячи із цих рекомендацій тестування буде проводитись на віртуальній машині з рекомендованими виділеними ресурсами та операційною системою Ubuntu Server 18.04 LTS Bionic Beaver.

Пререквізитами для запуску установки за допомогою Chef є встановлені пакети git та chef-client. Для встановлення git можна скористатися пакетним менеджером apt та встановити його і залежності звідти. Для встановлення chef-client потрібно завантажити пакет із сайту розробників на сервер та встановити його за допомогою утиліти dpkg.

Закінчивши підготовку потрібно клонувати репозиторій openstack-chef та переключитися на потрібну гілку (stable/train). Наступним кроком є копіювання ключа для розшифровки секретів у теку /etc/chef. Після цього можна запускати chef-client у локальному режимі використовуючи файл оточення та роль allinone. Локальний режим - це спосіб запуску Chef-клієнта з chef-repo на локальній машині, так якщо б він працював на сервері Chef. Локальний режим покладається на chef-zero, який діє як дуже легкий екземпляр сервера Chef. chef-zero читає і записує в chef_repo_path, що дозволяє використовувати всі команди, які зазвичай працюють з сервером Chef, для локального chef-repo.

Роль allinone включає у себе такі ролі як common для підготовки сервера, ops_database для встановлення та налаштування бази даних, ops_messaging для встановлення та налаштування RabbitMQ, та ролі для встановлення усіх інших сервісів що підтримуються, таких як Identity, Image, Network і та далі. Після ви-

конання цих етапів можна отримати налаштовану OpenStack платформу що складається з одного хоста.

4.3 Порівняння

У ході дослідження було проведено ряд експериментів із встановлення та налаштування платформи OpenStack з використанням двох засобів автоматизації розгортування - OpenStack-Ansible та OpenStack-Chef. Вони обидва показали чудовий результат та високу відтворюваність. Під час експериментів досліджуватися такі показники як відносний час потрібний для дослідження структури проекту, відносний час на підготовку до розгортування, час виконання розгортування. Також було оцінено стабільність проектів, можливий вплив людського фактору та потенційна вірогідність помилок.

Проект Ansible досить активно та його релізний цикл відстає на один реліз від циклу OpenStack. Проект Chef зараз знаходиться у фазі підтримки версій OpenStack, що знаходяться у довгостроковій підтримці. Останні два роки суттєвого розширення функціоналу не відбувається, скоріш за все це пов’язано з відтоком компаній доглядачів, що відповідали за цей проект. Проте таке відставання не є перешкодою до самостійної модифікації коду для введення підтримки актуальних версій платформи. Підходи використані у його розробці сприяють легкому внесенню змін такого порядку.

Не зважаючи на те що кодова база Ansible проекту більша ніж у Chef, загальний час який було витрачено на дослідження цих проектів був приблизно однаковим. Це загалом пов’язано із більш високою складністю проекту та меншою кількістю документації.

Час потрібний на підготовку до розгортування за допомогою Ansible був вищим через те, що кількість підготовчих етапів вища.

Час потрібний на розгортування за допомогою Ansible навіть у випадку якщо скоротити список компонентів для установки до такого ж як і для проекту Chef був вищим через наявність більшої кількості підготовчих етапів, що

пов’язано з використанням технології LXC. Використання цієї технології несе за собою як значні переваги так і суттєві недоліки, відношення яких треба оцінювати для конкретного стенду розгортування, його цілей та навичок адміністраторів.

Стабільність розгортування у обох проектів на високому рівні завдяки зрілості обох проектів - OpenStack-Ansible було розпочато у 2014 році, OpenStack-Chef у 2011.

Потенційна кількість помилок під час використання цих проектів напряму залежить як від навичок розробників та адміністраторів так і від розміру кодової бази. Ansible проект розрахований більше на адміністраторів та для використання його “як є” у той час як проект Chef більше розрахований на розробників та для використання у якості основи для наслідування з подальшими модифікаціями під задачі проекту - до цього спонукають підходи до розробки що закладені у основу Chef та яких притримувалися розробники проекту.

Обидва проекти мають досить високий потенціал до використання, хоча у проекті Ansible він незначним чином вищий за рахунок більш активного його розвитку та використання адміністраторами “як є” з модифікацією параметрів у рамках власних розгортуваннь. Проте проект Chef хоч а не знаходиться у фазі активного розвитку, проте має гарні перспективи для використання у якості платформи для побудови власних проектів на базі для використання Chef у більш гнучких цілях як то гібридні розгортування чи проекти що динамічно змінюють свою конфігурацію.

ВИСНОВКИ

У 2022 році засоби автоматизації розгортування програмного забезпечення набули широкого визнання у світі та дуже активно використовуються у бізнесі та IT-компаніях. Не стали винятком засоби автоматизації розгортування хмарних платформ, таких як OpenStack. На сьогоднішній день існують десятки способів пришвидшення та підвищення надійності його розгортування, конфігурації, підтримки та оновлення. Деякі з них були розглянуті у цій роботі.

У роботі було розглянуто ряд питань що стосуються хмарних обчислень, побудови таких платформ та автоматизації їх процесів життєдіяльності.

Перший розділ кваліфікаційної роботи присвячено аналізу видів хмар, їх розподілу на ринку, особливостям, перевагам та недолікам. Також було розглянуто тему гіперконвергентності у контексті побудови хмарних рішень.

У другому розділі розглядається широке різноманіття доступних для використання компонентів та досліджуються основні компоненти з яких складається хмарна платформа OpenStack.

Третій розділ присвячено дослідженню проектів автоматизації розгортування та налаштування платформи Openstack з використанням так і таких загально визнаних інструментів автоматизації як Chef, Ansible, Puppet, Saltstack так і спеціалізованих розробок як OpenStack Autopilot та Fuel Openstack

В четвертому розділі приділено увагу детальному розгляду таким проектам як OpenStack-Ansible та OpenStack-Chef. Досліджено їх структуру, функціонал, можливості використання та модифікації. Проведено практичне дослідження із розгортування платформи Openstack за допомогою цих проектів та проведено порівняння за результатами.

В роботі визначено переваги і недоліки цих проектів та виявлено що проект OpenStack-Ansible зараз більш активно розвивається та є більш підходящим для використання адміністраторами «як є», натомість проект OpenStack-Chef

знаходиться у фазі підтримки та є більш підходящим для розробників у якості платформи для побудови власних проектів на базі для використання Chef у більш гнучких цілях як то гібридні розгортування чи проекти що динамічно змінюють свою конфігурацію.

Результати роботи було апробовано на дев’ятій міжнародній науково-технічній конференції «Проблеми інформатизації» та опубліковано тези доповіді [4] за тематикою кваліфікаційної роботи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Cloud Computing Platform Software - OpenStack. openstack.org. [Онлайновий] https://www.openstack.org/software/project-navigator/openstack-components#openstack-services.

2. Кононюк, А. Е. Фундаментальная теория облачных технологий. Київ : Освіта України, 2018. с. 620.

3. Тенденции развития облачных технологий. Новые информационные технологии и программы. [Онлайновий] https://pro-spo.ru/cloud-technology/3211-tendenczii-razvitiya-oblachnyx-texnologij.

4. Тенденції розвитку гіперконвергентних хмарних обчисленнь для корпоративного сектору. Коротіч, О. В. та Костромицький, А. І. Черкаси, 18 – 19 листопада 2021 р., Тези доповідей дев’ятої міжнародної науково-технічної конференції «Проблеми інформатизації».

5. Кушнир, Екатерина. Что такое частные, публичные и гибридные облака: в чем разница и куда перенести сервисы компании. [Онлайновий] 12 08 2020 р. https://mcs.mail.ru/blog/chto-takoe-chastnye-publichnye-i-gibridnye-oblaka.

6. Частное или публичное облако: что выбрать. [Онлайновий] 6 05 2020 р. https://www.xelent.ru/blog/chastnoe-ili-publichnoe-oblako-chto-vybrat/.

7. Что такое гиперконвергентная инфраструктура. Intel. [Онлайновий] https://www.intel.ru/content/www/ru/ru/cloud-computing/hyperconverged-infrastructure.html.

8. Гиперконвергентность . VMware. [Онлайновий] https://www.vmware.com/ru/topics/glossary/content/hyperconvergence.html.

9. Packt. OpenStack Cloud Computing Cookbook - Fourth Edition. місце видання невідоме : Packt, 2018. 9781788398763.

10. O'Reilly. Identity, Authentication, and Access Management in OpenStack. місце видання невідоме : O'Reilly Media, Inc., 2015. 9781491941201.

11. University of Cambridge. Computing service: OpenStack Nova. Research Computing Services HPC Documentation. [Онлайновий] https://docs.hpc.cam.ac.uk/cloud/userguide/03-nova.html.

12. —. Networking service: OpenStack Neutron RCC Documentation. Research Computing Services HPC Documentation. [Онлайновий] https://docs.hpc.cam.ac.uk/cloud/userguide/02-neutron.html.

13. —. Storage services: OpenStack Cinder and Swift. Research Computing Services HPC Documentation. [Онлайновий] https://docs.hpc.cam.ac.uk/cloud/userguide/06-cinder_swift.html.

14. Vexxhost. Looking At OpenStack Glance. [Онлайновий] https://vexxhost.com/blog/openstack-glance/.

15. Hewlett Packard Enterprise. Infrastructure as Code. Hewlett Packard Enterprise (HPE) | Deutschland. [Онлайновий] https://www.hpe.com/de/de/what-is/infrastructure-as-code.html.

16. More, Siddheshwar. 6 OpenStack Deployment tools that are awesome for your project. Opcito. [Онлайновий] https://www.opcito.com/blogs/6-openstack-deployment-tools-that-are-awesome-for-your-project-and-why.