АЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра Автоматики і управління у технічних системах

«На правах рукопису»
УДК _____

«Допущений до захисту»
Завідувач кафедри
_____ Ролік О.І.
(підпис) (ініціали, прізвище)
«____» _____ 2018 р.

Магістерська дисертація

зі спеціальності (спеціалізації) 126 Інформаційні системи та технології
(код і назва спеціальності)

на тему: Лабораторний стенд для проектування та управління хмарними IT-інфраструктурами

Виконав студент II курсу, групи ІА-72мп
(шифр групи)

Данильченко Михайло Андрійович _____
(прізвище, ім'я, по-батькові) (підпис)

Керівник Зав. каф. АУТС д.т.н. проф Ролік О.І. _____
(посада, наукова ступінь, звання, прізвище, ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (посада, наукова ступінь, звання, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, наукова ступінь, вчене звання, прізвище, ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 р.

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

ФІОТ

Кафедра АУТС

Ступінь вищої освіти «магістр»

Зі спеціальності 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Ролік А.І._____
(підпис) (ініціали, прізвище)
«__» _____.

ЗАВДАННЯ
на магістерську дисертацію студенту
Данильченку Михайлу Андрійовичу .
(прізвище, ім'я, по батькові)

1. Тема роботи: Лабораторний стенд для проектування та управління хмарними IT-інфраструктурами _____,

керівник роботи д.т.н. професор кафедри АУТС Ролік Олександр Іванович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «__» _____ р. №

2. Строк подання студентом проекту_____

3. Вихідні дані до магістерської дисертації_____

_____

_____

4. Перелік завдань, які потрібно розробити: огляд, аналіз основних хмарних IT-інфраструктур, дослідження можливостей OpenStack, розробка структурної та функціональної схеми хмарної IT-інфраструктури, процес розгортання та управління лабораторного стенду_____

5. Перелік графічного матеріалу функціональна та структурна схема лабораторного стенду, UML діаграмма, діграма вза'ємозвязків між сервісами, алгоритм розгортання віртуальної машини, алгоритм розгортання скриптів, мережеві графи, KVM діаграмма ,

6. Консультанти розділів проекту

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

7. Дата видачі завдання 28.10.2018

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Строк виконання етапів проекту	Примітка
	Огляд, аналіз та опис предметної галузі і готових рішень	28.10.2018	
	Дослідження можливостей вибраної платформи	06.11.2018	
	Розробка структурної та функціональної схеми	14.11.2018	
	Процес розгортання та управління лабораторного стенду	22.11.2018	
	Проведення аналізу стартап-проекту	25.11.2018	
	Оформлення ПЗ	03.12.2018	

Студент _____ Данильченко М.А.
(підпис) (ініціали, прізвище)

Керівник проекту _____ Ролік О.І.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Магістерська дисертація освітньо-кваліфікаційного рівня “магістр” на тему: «Лабораторний стенд для проектування та управління хмарною IT-інфраструктурою». Робота включає 101 сторінок.

Магістерська дисертація містить результати розроблення лабораторного стенду хмарної IT-інфрастрктури, яка може бути використана як основа для реалізації аналогічних рішень. Була розвернена базова архітектура OpenStack. Продемонстровано взаємодія сервісів всередині платформи.

Ключові слова: Хмарна IT-інфраструктура

ABSTRACT

Master's dissertation of the educational qualification level "Master" on the topic: "Laboratory stand for designing and managing cloud-based IT infrastructure". The work includes 101 pages.

The master's dissertation contains the results of the development of a laboratory stand of the cloud IT infrastructure, which can be used as a basis for the implementation of similar solutions. The basic OpenStack architecture was deployed. The interaction of services inside the platform is demonstrated.

Key words: Cloud IT infrastructure

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IT – інформаційні технології

ПЗ – програмне забеспечення

QEMU – Quick Emulator

KVM – Kernel-based Virtual Machine

NFV – Network Functions Virtualization

VM – Virtual Machine

API – Application Programming Interface

RDO – Raster Document Object

RHEL – Red Hat Enterprise Linux

SDN – Software-defined networking

URL – Uniform Resource Locator

SQL – Structured Query Language

NTP – Network Time Protoco

DHCP – Dynamic Host Configuration Protocol

BIOS – Basic Input/Output System

HDL – Hardware description language

CLI – Call Level Interface

SHH – Shoshoni language

DNS – Domain Name System

IP – Internet Protocol PNI – Private Network Interconnect

LVM – Logical Volume Manager

UUID – Universally unique identifier

UML – Unified Modeling Language

ВСТУП

Сьогодні вже неможливо уявити собі сучасний офіс, успішно розвивається бізнесу без застосування інформаційних технологій (ІТ). Успішний розвиток сучасного бізнесу неможливо без ефективного (стабільного і безпечного) функціонування ІТ – інфраструктури, її відповідності поставленим завданням і якісних ІТ–послуг.

ІТ–інфраструктура є не просто фундаментом для існування будь–якої сучасної компанії. ІТ в даний час стають стратегічним активом, який є рушійною силою бізнесу. Побудова надійної ІТ–інфраструктури, що задовольняє бізнес процесів компанії – складне завдання, практично не має вирішення власними силами ІТ–відділу компанії.

Як відомо, вміло і правильно вибудувана ІТ–інфраструктура підприємства може в рази підвищити прибутковість бізнесу, скоротити витрати і непродуктивні витрати, збільшити віддачу від роботи і значно полегшити діяльність компанії.

Сучасні ІТ–інфрастурктури дуже часто використовують "хмарних" середовища: як в глобальному інтернеті, так і на підприємствах.

Суть концепції "хмарних" технологій полягає в наданні кінцевим користувачам віддаленого динамічного доступу до послуг, обчислювальних ресурсів і додатків (включаючи операційні та інформаційні системи, серверне ПЗ і інше) через інтернет або по засобом корпоративної мережі. Розвиток сфери хостингу і необхідність масового використання громадськими ресурсами було обумовлено виниклою потребою в програмному забезпеченні і цифрових послугах, якими можна було б управляти зсередини, але які були б при цьому більш економічними і ефективними за рахунок економії на масштабі.

Технології "хмарних" обчислень мають величезний потенціал, тому що всі сучасні комп'ютерні продукти постійно збільшують свої вимоги до технічного оснащення комп'ютера користувача, що неминуче веде до значних витрат на апгрейд. Так що дана технологія дозволяє вирішити проблему надмірної вимогливості додатків до ресурсів кінцевого користувача.

При використанні "хмарних" технологій можливо не тільки скорочення витрат на фізичне устаткування, але і глобальне об'єднання даних з їх захистом, можливість роботи віддалено з інформаційною системою підприємства і персоналізація "хмарного" ядра під потреби підприємства.

Інфраструктура як послуга (англ. Infrastructure as a service, IaaS) — це модель обслуговування, в межах якої споживачу надається можливість керувати засобами обробки та збереження, комунікаційними мережами, та іншими фундаментальними обчислювальними ресурсами, на базі яких споживач може розгортати та виконувати довільне програмне забезпечення, до складу якого можуть входити операційні системи та прикладні програми. Споживач не керує фізичною та віртуальною інфраструктурою, що лежить в основі хмари, проте він контролює операційні системи, системи збереження, встановлені програми та, можливо, має обмежений контроль над деякими мережевими компонентами (наприклад, мережевими екранами вузлів).

IaaS складається з трьох основних компонентів:

- апаратні засоби (сервери, системи зберігання даних, клієнтські системи, мережеве обладнання);
- операційні системи та системне ПЗ (засоби віртуалізації, автоматизації, основні засоби управління ресурсами);
- в'язуюче ПЗ (наприклад, для управління системами).

Віртуалізація всієї інфраструктури або якийсь її частини стала буденною завданням для більшості компаній рівня enterprise. Причому, якщо на слуху насамперед ті хмарні рішення, які використовуються для побудови публічних хмар, від Microsoft або, наприклад, Google, корпоративні замовники віддають перевагу приватні хмарні інфраструктури або інфраструктури гібридного типу.

На зорі загальної хмарної трансформації в рамках однієї великої компанії могло реалізовуватися відразу кілька хмарних ініціатив, причому просуваються незалежними один від одного командами. Крім того, підприємства активно тестували хмарні продукти з тим, щоб визначити, яким чином вони справляються з навантаженнями в польових умовах і які вигоди може принести їх використання. В

даний час підхід корпоративних замовників до вибору хмарних рішень кардинально змінився: компанії знаходяться в пошуку гнучко настроюються IaaS–платформ, які могли б стати міцним віртуальним «фундаментом» для всієї IT–інфраструктури і при цьому були б здатні нести високі обчислювальні навантаження і з легкістю підлаштовуватися під конкретні завдання і вимоги.

Причому нерідко їх вибір схиляється на користь open source платформ, так як, по–перше, і це підтримує головну ідею «хмар» про економію операційних витрат, відкриті платформи не вимагають додаткових інвестицій в ліцензійну складову. А, по–друге, open source IaaS платформи пропонують порівнянний інструментарій та можливості в порівнянні з пропрієтарними аналогами, так як їх ком'юніті в останні роки активно розвиваються і спільноти розробників реагують на потреби конвертових проектів часом навіть більш оперативно, ніж вендори ПО віртуалізації.

Насправді, найбільш життєздатні платформи в сегменті open source IaaS можна перелічити на пальцях однієї руки, але деякі з них досить впливові, щоб впливати на всю хмарну індустрію, як OpenStack, а деякі залишаються лише нішевими рішеннями, що не скасовує їх цінності для окремо взятих проектів. В якості найбільш вагомою альтернативи OpenStack серед платформ з відкритим кодом ті ж аналітики Forrester називають CloudStack, Euacalyptus і Open Nebula. У цьому ж ряду можна згадати екзотику типу рішень Abiquo і ProxMox.

1 ОГЛЯД ТА АНАЛІЗ ОСНОВНИХ ХМАРНИХ IT-ІНФРАСТРУКТУР

1.1 Огляд архітектури vCloud Director

vCloud Director платформа для побудови хмар від VMware. Якщо вся ваша інфраструктура, побудована на продукту VMware, то впровадження vCloud Director буде ідеальним рішення. vCloud Director дозволяє будувати дійсно гібридні хмари. За допомогою VMware vCloud Connector ви можете мігрувати свої віртуальні машини між своїм public хмарою і private хмарою. Основні компонети:

- Virtual Datacenters;
- vShield Security Technologies;
- Infrastructure Service Catalog;
- Multi–tenant Organizations;
- Self–service Portal;
- VMware vCloud API, Open Virtualization Format, and Callouts.

Платформа є платною. Для того щоб дізнатися приблизну ціну пакета, який вам підійде, потрібно зв'язуватися з регіональними менеджерами.

Через те, що VMware є лідером ринку у нього прекрасне співтовариство. Відмінна підтримка у вигляді knowledge base. Не варто забувати і про платну підтримку і підтримку після покупки продуктів у компанії VMware.

У платних продуктів завжди хороша документація. І як показала практика, якщо слідувати їй, то установка пройде без проблем.

1.2 Огляд архітектури OpenStack

OpenStack — це комплекс проектів вільного програмного забезпечення для створення обчислювальних хмар і хмарних сховищ, як публічних, так і приватних (працюють тільки для забезпечення внутрішніх потреб компанії). Серед

підтримуваних систем віртуалізації: KVM, QEMU, Xen, Hyper–V, Citrix XenServer, контейнери LXC і VMWare/vSphere ESX/ESXi. OpenStack чудово масштабується і здатний обслуговувати інфраструктуру з сотень тисяч віртуальних серверів.

На тлі перерахованих програмних продуктів, навіть настільки життєздатних як CloudStack або Eucalyptus, успіх OpenStack виглядає феноменальним. Головний козир платформи, мабуть, полягає в тому, що вона є вагомою і вже реально діючу пенсійну систему сотнях проектів в усьому світі альтернативою VMware.

OpenStack – найбільш швидко розвивається відкрита платформа для створення хмарних рішень, яка підтримується спільнотою розробників, що включає 500 організацій з 165 країн, вони написали вже більше 4 млн рядків коду.

За підсумками декількох років існування платформи, кількість публічних хмар на OpenStack в світі обчислюється десятками, а приватних – сотнями. Найбільші компанії на зразок американської Walmart використовують хмарні рішення на OpenStack для високонавантажених обчислювальних інфраструктур.

Для компаній великою перевагою OpenStack є те, що в розвитку платформи беруть участь не тільки окремі ентузіасти, а й найбільші сервіс–провайдери. Так, наприклад, «Сервіоніка» надає власну «збірку» OpenStack, а також ряд оригінальних рішень на її базі, платформу для побудови приватних і гібридних хмар eCloud і сервіс для самостійного створення та управління IT–інфраструктурою MakeCloud.

Наявність такого активного співтовариства і російськомовної експертизи – величезний плюс для вибору на користь OpenStack, але не єдиний: у платформи є також і ряд технологічних переваг. За даними останнього дослідження Forrester Research, платформа OpenStack повністю готова для використання бізнесом [13]. Як явних ознак того, що продукт дозрів для таких впроваджень, експерти називають доведеним до ладу код (особливо в 11 релізі під назвою Klio), наявність задовільних посилань від тих, хто вже впровадив платформу.

Третій безсумнівний плюс – бурхливий розвиток функціоналу платформи, в тому числі за рахунок її адаптації в рамках нових користувальницьких кейсів і нових виникаючих програмно–апаратних комбінацій. При цьому помилки усуваються

дуже швидко, так як вони мають критичне значення для тих проектів, де платформа адаптується: відгук користувачів дуже високою.

І якщо коло перших користувачів OpenStack обмежувався технологічними і близько технологічними компаніями на кшталт CERN, Digital Film Tree або NSA, то тепер вибір на користь платформи роблять представники списку Fortune 100 начебто WalMart. Тим не менше, більшість компаній асоціюють успіх використання OpenStack з співпрацею з надійним сервісним партнером. Винятки становлять хіба що Comcast і eBay, які займалися налаштуванням платформи власними силами.

Інші перераховані IaaS платформи на відкритому коді, залишаючись нішевими рішеннями, присутні в проектах, але їх не можна можна назвати масово поширеними: такі впровадження найчастіше ведуться невеликими командами для потреб невеликих організацій – наприклад, проектно–дослідницьких інститутів. У таких випадках в хитросплетіннях документації та особливості коду платформи IT-фахівцям доводиться розбиратися самостійно, а вузьке коло впроваджень не дає можливості тиражувати досвід таких проектів.

1.3 Огляд платформи CloudStack

Cloudstack – це консоль управління обчислювальними ресурсами вашого датацентру. На цій платформі побудували свої хмари такі великі компанії як Zynga, Nokia Research Center, Cloudcentral і інші. Розвиток проекту здійснюється компанією Citrix. У платформи є своє API, який дозволяє налаштовувати та інтегрувати платформу з наявною інфраструктурою, а за допомогою перехідника CloudBridge Amazon EC2 можна конвертувати Amazon API в Cloudstack API. Повний список команд підтримуваних переходником можна знайти тут.

Компонети платформи:

- Hypervisors Agnostic (KVM, XEN, ESXi, OVM, BareMetal);
- Roles;
- Virtual network;
- Resource pool;

- Snapshots and volumes;
- Virtual routers, firewall and load balancer;
- Live Migration with host maintenance.

При роботі з гіпервізором ESXi платформа використовує vCenter API. Таким чином, впровадження платформи в наявну інфраструктуру побудованої на VMware пройде легко.

На даний момент Cloudstack безкоштовний і поширюється під ліцензією GNU Public License Version 3.

Під час установки платформи може виникнути багато проблем. Спробувати вирішити їх можна звернувшись із запитанням до товариства. На форумі обговорюють багато питань і допомагають вирішити їх. Є канал в IRC, в якому також можна задати питання.

Напевно, досить важливим елементом у всіх продуктах є документація. Вона визначає, наскільки легким буде вхід користувача в цю технологію. Чим простіше і зрозуміліше документація, тим більше шансів, що платформу зможе налаштувати зацікавився чоловік. Якщо у вас є певний background, то ви без проблем проведете просту інсталяцію платформи, для ознайомлення цього достатньо. Складніша інсталяція та планування архітектури буде займати багато часу, в документації не все особливість розкриті. Документація зроблена в стилі step by step і не описує, як платформа працює. Частина знань прийде до вас тільки після реального використання платформи.

1.4 Порівняння платформ за їх можливостями

Для порівняння я вибрав дві онлайн платформи OpenStack і CloudStack, ось основні їх переваги і недоліки.

1.4.1 CloudStack

Переваги: це справді не поліпшується. Останній випуск CloudStack насправді дуже приємний. Розгортання дійсно гладке, що складається з лише одного віртуальної машини, що працює на сервері CloudStack Management, а іншу – як справжню хмарну інфраструктуру. Насправді, ви можете розгорнути все це на одному фізичному хості.

Складнощі: Перший стабільний (технічний) випуск CloudStack вийшов навколо 2013 року з 4.0.2; деякі з них все ще сумніваються в швидкості прийняття CloudStack. Навіть з деякими великими досягненнями, деякі скаржаться на те, що процес архітектури та встановлення – хоч і спрощений – вимагає трохи знань і часу для розгортання.

Нове у платформі: 4.1 (з випущеним 4.4.2) – це поліпшена безпека, агностицизм гіпервізора та вдосконалений керування мережевим рівнем. Крім того, великі оновлення обертаються навколо:

- покращений менеджмент зберігання даних;
- віртуальні приватні області Cloud тепер можуть охоплювати гостьові мережі через доступні зони;
- підтримка планувальника розподілених ресурсів VMware;
- покращена підтримка гіпер–V зон, VPC і міграції зберігання.

Хто використовує це: DataPipe в даний час використовує свою глобальну Cloud інфраструктуру CloudStack. За даними DataPipe, їх причини для переходу на платформу включають:

- Призупинені віртуальні машини підтримують стан машини без обчислення витрат;
- Масштаб зберігання незалежно від обчислення;
- Єдина зона безпеки в усіх регіонах;
- Доступ до економічної зони Гонконгу та Шанхаю (материковий Китай);
- Додаткова економія коштів в результаті високоефективних віртуальних машин, які вимагають менше обчислювальних ресурсів.

За межами Datapipe – найбільшого поточного користувача CloudStack – є й інші менші, але важливі послідовники. Це включає в себе Shopzilla, послуги SunGard Availability, CloudOps, Citrix, WebMD Health та ряд інших.

Загальний консенсус полягає в тому, що CloudStack, хоча і сильно набирає популярність, полягає в тому, що він все ще знаходиться в тіні openStack. Тим не менш, існує ще більше організацій, які рухаються до сьогоднішньої моделі Apache Inkubation Program CloudStack, особливо тому, що багато ранніх припущень про виразки були вирішені.

1.4.2 Огляд переваг і недоліків платформи OpenStack

Переваги: це, безумовно, більш зрілий продукт. Крім того, існує понад 150 компаній (AMD, Brocade, Dell, HP, IBM, VMware та Yahoo), які всі сприяють розвитку. Вона розглядається як лідер у сфері керування хмарними платформами, а темпи зростання продовжуються.

Складнощі: навіть при такому значному впровадженні та розробці навколо платформи OpenStack все ще складно розгортати, і в багатьох випадках це потрібно управляти з різних консольних систем CLI. Фрагментована архітектура складається з декількох різних модульних компонентів, включаючи Compute, Open Storage, Block Storage, Networking, Dashboard, службу ідентифікації, Image Server, Telemetry, Cloud Messaging для декількох орендарів, Elastic Map Reduce та інші. Хороша новина полягає в тому, що там існує безліч конфігурацій та сценаріїв інсталяції для використання як шаблон.

Нове у платформі: Так, існують ще деякі технічні проблеми та проблеми розгортання. Це зупинило імпульс прийняття? Не зовсім Останній випуск Juno надає 342 нових функцій. Випуск Juno додає корпоративні функції, такі як політика зберігання, новий сервіс обробки даних для Hadoop і Spark, і закладає основу для OpenStack як платформи для віртуалізації мережевих функцій (NFV), що є основним перетворенням, що сприяє підвищенню оперативності та ефективності в telco. і центри обробки даних постачальника послуг.

Хто це використовує: Спільно розпочатий NASA та Rackspace Hosting, OpenStack мав серйозних підтримувачів з самого початку. Тепер OpenStack використовується такими організаціями, як AT & T, CERN, Yahoo!, HP Public Cloud, Red Hat OpenShift та деякі інші.

2 ДОСЛІДЖЕННЯ МОЖЛИВОСТЕЙ OPENSTACK

2.1 Архітектура OpenStack

Проект OpenStack, який також називають хмарної операційної системою, складається з ряду окремих проектів, які розробляють окремі підсистеми. Конкретна установка OpenStack може включати в себе лише частину з них. Деякі підсистеми можуть використовуватися взагалі автономно або як частина інших OpenSource проектів. Їх набір збільшується від версії до версії проекту OpenStack як за рахунок появи нових, так і за рахунок поділу функціоналу сущестчих. Наприклад, сервіс nova-volume виділився в окремий проект Cinder. Кожен з проектів має свій документований набір REST API, утиліт командного рядка і «рідні» інтерфейси Python, що надають набір функцій, аналогічних утиліт командної рядки. Одним з базових сервісів є OpenStack Compute (Nova). Цей сервіс встановлюється на всіх обчислювальних вузлах кластера. Він надає рівень абстракції віртуального обладнання (Процесори, пам'ять, блокові пристрої, мережеві адаптери). Nova забезпечує управління екземплярами віртуальних машин, звертаючись до гіпервізор і віддаючи такі команди, як, наприклад, їх запуск і зупинку.

2.1.1 Введення в OpenStack

Важливо відзначити, що технології OpenStack незалежні від гипервизора. За адресою HypervisorSupportMatrix розташовується матриця сумісності гіпервізора. Підтримка реалізується через відповідні драйвери в проекті Nova [4]. Розробка і тестування OpenStack ведуться в першу чергу для KVM. Більшість впроваджень також зав'язано на гипервизор KVM. Наступний сервіс під назвою OpenStack Networking (Neutron) відповідає за мережеву зв'язаність. Користувачі можуть самостійно створювати віртуальні мережі, маршрутизатори, призначати IP-адреси. Один з механізмів, що забезпечуються Neutron, називається «Плаваючі адреси». Завдяки цьому механізму віртуальні машини можуть отримувати зовнішні фіксовані IP-адреси. через механізм модулів можна реалізувати такий функціонал, як

балансувальник мережевого навантаження як сервіс, брандмауер як сервіс і VPN як сервіс. Служба ідентифікації OpenStack Keystone є централізований каталог користувачів і сервісів, до яких вони мають доступ. Keystone виступає у вигляді єдиної системи аутентифікації хмарної операційної системи. Keystone перевіряє дійсність облікових записів користувачів, перевіряє зіставлення користувачів проектам OpenStack і ролям і в разі успіху видає токен на доступ до інших сервісів. Також Keystone веде каталог служб. OpenStack Image Service (Glance) веде каталог образів віртуальних машин, які користувачі можуть використовувати як шаблонидля запуску примірників віртуальних машин в хмарі. Також даний сервіс надає функціонал резервного копіювання і створення моментальних знімків. Glance підтримує безліч різних форматів, включаючи vhd, vmdk, vdi, iso, qcow2, ami і ін. Сервіс OpenStackBlockStorage (Cinder) управляє блоковим сховищем, яке можуть використовувати запущені екземпляри віртуальних машин. Це постійне сховище інформації для віртуальних машин. Можна використовувати моментальні знімки для збереження і відновлення інформації або для цілей клонування. Найчастіше всього з cinder використовують сховище на основі Linux-серверів, проте є і модулі для апаратних сховищ. Сервіс OpenStack Object Storage (Swift), крім Nova, є одним з двох перших проектів, що з'явилися в OpenStack. Від самого початку він називався Rackspace Cloud Files. сервіс являє собою об'єктне сховище, що дозволяє користувачам зберігати

Гипервизор KVM і емулятор QEMU. Swift має розподілену архітектуру, забезпечуючи горизонтальне масштабування, а також надмірність і реплікацію для цілей відмовостійкості. Swift орієнтований переважно на статичні дані, такі як образи віртуальних машин, резервні копії та архіви. Сервіс OpenStack Telemetry (Celiometer) являє собою централізоване джерело інформації по метриках хмари і даними моніторингу. Цей компонент забезпечує можливість білінгу для OpenStack. OpenStack Orchestration (Heat) – сервіс, завдання якого – забезпечення життєвого циклу додатки в хмарної інфраструктурі. За допомогою шаблону формату AWS CloudFormation сервіс управляє іншими сервісами OpenStack, дозволяючи створити більшість типів ресурсів (віртуальні машини, тому, плаваючі IP, користувачі, групи

безпеки і т. д.). Heat за допомогою даних Ceilometer також може здійснювати автоматичне масштабування додатки. Шаблони описують відносини між ресурсами, і це дозволяє сервісу Heat здійснювати виклики API OpenStack в правильному порядку, наприклад спочатку створити сервер, а потім підключити до нього тому. І нарешті, найближчий до користувача хмари сервіс OpenStack.

Dashboard (Horizon), що дозволяє управляти ресурсами хмари через веб-консоль.

І нарешті, найближчий до користувача хмари сервіс OpenStack Dashboard (Horizon), що дозволяє управляти ресурсами хмари через веб-консоль.

Також існує проект, чий код використовується більшістю інших компонентів OpenStack, – це проект, який об'єднує загальні бібліотеки компонентів хмари OpenStack Oslo.

2.2 Дистрибутиви OpenStack

Як вже зазначалося вище, OpenStack – це проект зі створення хмарної інфраструктури, але не продукт. Однак безліч компаній, що беруть участь у створенні OpenStack, створюють на основі його коду свої продукти або дистрибутиви, часто використовуючи свої пропрієтарні компоненти. Тут ситуація приблизно аналогічна створенню дистрибутивів GNU / Linux. Тексти програм у вигляді пакетів tar.gz доступні за адресою [14]. На даний момент це понад 350 окремих проектів. Автор спробував дати дуже короткий огляд дистрибутивів OpenStack. Огляд не претендує на всеосяжний охоплення. Також інформація, наведена нижче, актуальна на момент написання книги. Досить повний список основних дистрибутивів наведено в розділі Marketplace офіційного сайту OpenStack [1]. Книга описує використання дистрибутива RDO, з нього і почнемо наш короткий огляд [2]. RDO – спонсорований Red Hat проект по створенню відкритого дистрибутива OpenStack. На противагу комерційному дистрибутива Red Hat Enterprise OpenStack Platform (RHOSP), для RDO не можна купити підтримку. Взаємозв'язок між RHOSP і RDO приблизно така ж, як між Red Hat Enterprise Linux

(RHEL) і Fedora. RDO покликаний створити спільноту навколо розробок Red Hat. Як установника спочатку пропонувалося використовувати скрипт packstack для тестів і Foreman для промислового розгортання. В останніх версіях замість Foreman пропонується RDO Manager, заснований на проектах OpenStack Ironic і OpenStack TripleO. RDO можна розгорнути на RHEL і його похідних (CentOS, Oracle Linux та інших). Другий за популярністю комерційний вендор GNU / Linux також має свій власний дистрибутив OpenStack під назвою SUSE OpenStack Cloud. Як дистрибутива операційної системи використовується на вибір SUSE Linux Enterprise Server 11 або SUSE Linux Enterprise Server 12. В якості системи зберігання даних підтримується SUSE Enterprise Storage – власний варіант збірки програмно-обумовленого сховища даних Ceph. В якості інструменту установки використовуються проект Crowbar (http: // crowbar.github.io) і Chef – один з лідируючих інструментів управління конфігураціями в світі OpenSource. Наступний дистрибутив – Mirantis OpenStack (MOS). Як і RDO, в ньому відсутні пропрієтарні компоненти, а відмітною особливістю Mirantis позиціонує систему установки Fuel, яка значно спрощує масштабні розгортання. Також потрібно відзначити підтримку OpenStack Community Application Catalog, заснованого на каталозі додатків Murano. Як дистрибутива GNU / Linux MOS вимагає на вибір Ubuntu або CentOS. З метою спрощення розгортання демостендов або вивчення OpenStack є скрипти для швидкого розгортання на VirtualBox. Сама компанія Mirantis має в Росії і Україні офіси розробки, а також регулярно проводить заходи, присвячені OpenStack. Автор вважає, що, як і RDO, дистрибутив від Mirantis може стати гарною відправною точкою для знайомства з OpenStack. VMware Integrated OpenStack (VIO) було трохи відособлене від інших розглянутих в цьому розділі, оскільки для своєї роботи вимагає розгорнутої проприетарной інфраструктури VMware, включаючи гипервизор ESXi, платформу віртуалізації мережі NSX і систему управління vCenter. Все це робить VIO з урахуванням відповідних ліцензій щодо дорогим рішенням. всі компоненти OpenStack розгортаються всередині віртуальних машин на ESXi в завчастному створеному кластері Vmware. Оскільки в основі лежить інфраструктура VMware vSphere, то відмітною перевагою дистрибутива буде «рідний» функціонал

VMware як корпоративної системи віртуалізації: HA, DRS, vMotion і т. д. Це кілька переорієнтує OpenStack від хмарної до традиційної навантаженні. Також VIO знижує поріг входження для існуючих інфраструктур і навчених адміністраторів VMware. З іншого боку, крім необхідності купувати ліцензії на пропрієтарне програмне забезпечення, VIO прив'язує користувача цього дистрибутива до вендору рішення. Oracle OpenStack for Oracle Linux виділяється помітно недорогий, в порівнянні з конкурентами, технічною підтримкою в разі комерційного використання. Він безкоштовний при наявності преміальної підтримки. З відмітних особливостей можна відзначити підтримку Oracle ZFS. Також в якості віртуальних машин підтримується Solaris x86. Як і в разі інших виробників апаратного забезпечення, наприклад IBM і HP, Oracle підтримує комерційне використання своєї збірки тільки на своєму «залізі». Для інсталяції використовуються проект OpenStack Kolla і контейнери Docker.

Ericsson Cloud Execution Environment – дистрибутив, створений з урахуванням вимог додатків віртуалізації втратити зв'язок із мережею (NFV) і специфіки операторів зв'язку. В першу чергу це означає підвищену продуктивність мережевий підсистеми і орієнтацію на додатки з вимогою операцій реального часу. У порівнянні з збірками традиційних IT-компаній, орієнтований на операторів дистрибутив Ericsson Cloud Execution Environment відрізняється гарантованим компанією Ericsson SLA. Як приклад функціоналу можна привести підтримку VLAN Trunking, прискорений за допомогою бібліотеки Intel DPDK віртуальний комутатор (Ericsson Virtual Switch), моніторинг, високу доступність віртуальних машин і багато іншого. Також з відмінностей можна назвати свій власний веб-інтерфейс на базі Horizon. Дистрибутив створений на базі Mirantis OpenStack і в якості опції надалі пропонуватиме підтримку дистрибутива Red Hat. Hewlett Packard Enterprise (HPE) Helion OpenStack – збірка від компанії HPE, яка останнім часом активно вкладає ресурси в розвиток проекту. На момент написання книги компанія була найбільшим роботодавцем для фахівців в OpenStack з сотнями вакансій по всьому світу. Крім того, компанія є вендором номер один за вкладом в розробку OpenStack. Дистрибутив HPE Helion OpenStack побудований на основі Ubuntu Linux. В якості

системи розгортання використовується проект TripleO – OpenStack on OpenStack. TripleO дозволяє інсталювати хмара OpenStack за допомогою самого OpenStack. Для цього створюється спеціальний керуючий хмара у віртуальній машині, яке в подальшому використовується тільки для адміністрування основного хмари. проект TripleO входить в число базових проектів OpenStack. З цієї точки зору знайомство з розгортанням HPE Helion OpenStack було б також цікавим.

Говорячи про дистрибутивах OpenStack, також необхідно згадати проект OPNFV [3]. OPNFV – це проект з побудови відкритої стандартної платформи для віртуалізації мережевих функцій (NFV). OPNFV об'єднує ряд проектів, включаючи OpenStack, OpenDaylight, Ceph Storage, KVM, Open vSwitch і GNU / Linux. У проекті беруть участь найбільші телекомунікаційні компанії і вендори (AT & T, Cisco, EMC, Ericsson, HP, Huawei, IBM, Intel, NEC, Nokia, Vodafone, ZTE і багато інших). На даний момент проект надає кілька збірок у вигляді ISO з різними установниками.

2.3 OpenStack сервіси

В даній архітектурі будуть використані такі сервіси як: Nova, Neutron, Swift, Cinder, Keystone, Glance (рисунок 2.1).

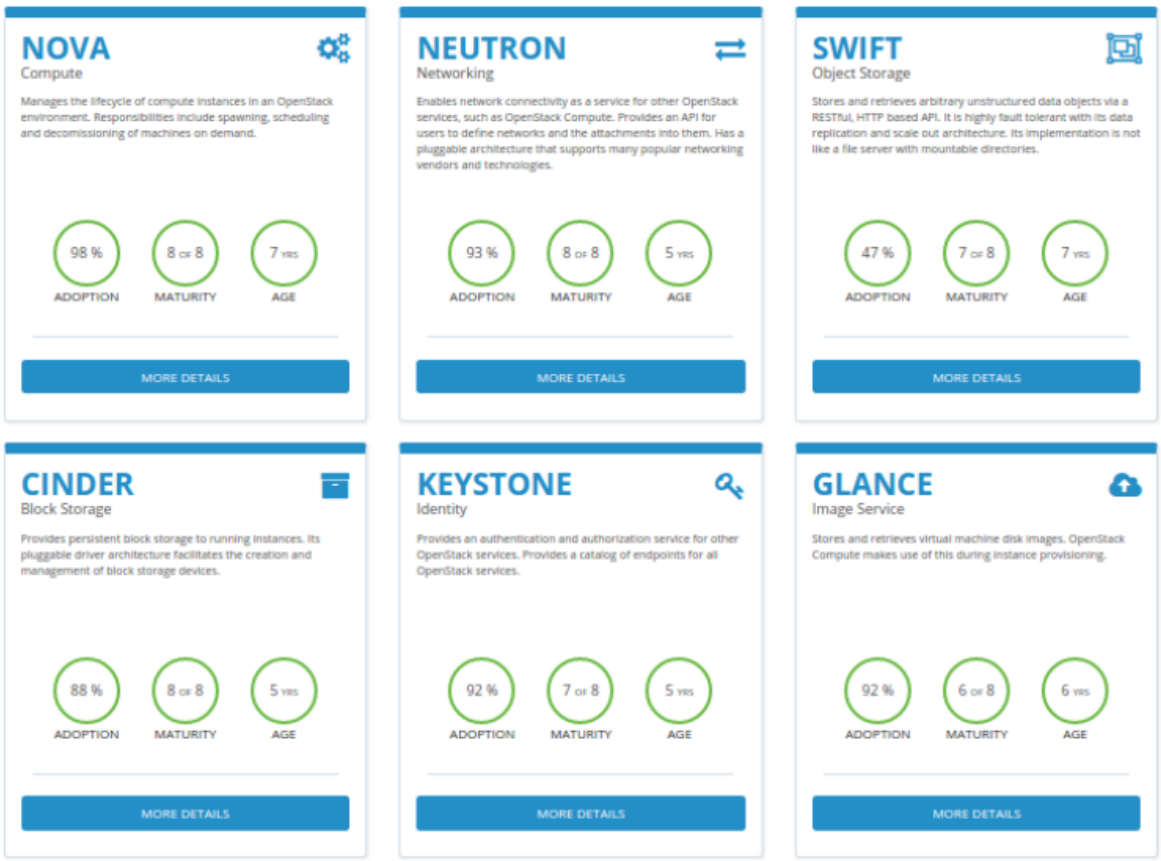


Рисунок 2.1 – Openstack сервіси

2.3.1 NOVA "Обчислювальний" сервіс

Nova – найважливіший основний проект у OpenStack. Він обробляє обчислення навколишнє середовище, життєвий цикл прикладу VM. Сервіс Nova – це не гіпервізор сам по собі, але інтерфейси до ряду різних гіпервізорів, таких як Xen, віртуальна машина Kernel (KVM) / швидкий емулятор (QEMU), VMware, vSphere. Nova встановлює агент на гіпервізор, тому він підтримується в середовищі OpenStack. Служба Nova відповідає за нерест, планування та виведення з експлуатації ВМ за вимогою. Вона включає сервісні процеси Nova, які працюють у хмарі контролера, а також агентів Nova, які працюють на обчислювальних вузлах.

2.3.2Neutron сервіс "Мережі"

OpenStack Neutron дозволяє використовувати програмне забезпечення, що визначає мережу (SDN) (рисунок 2.2).

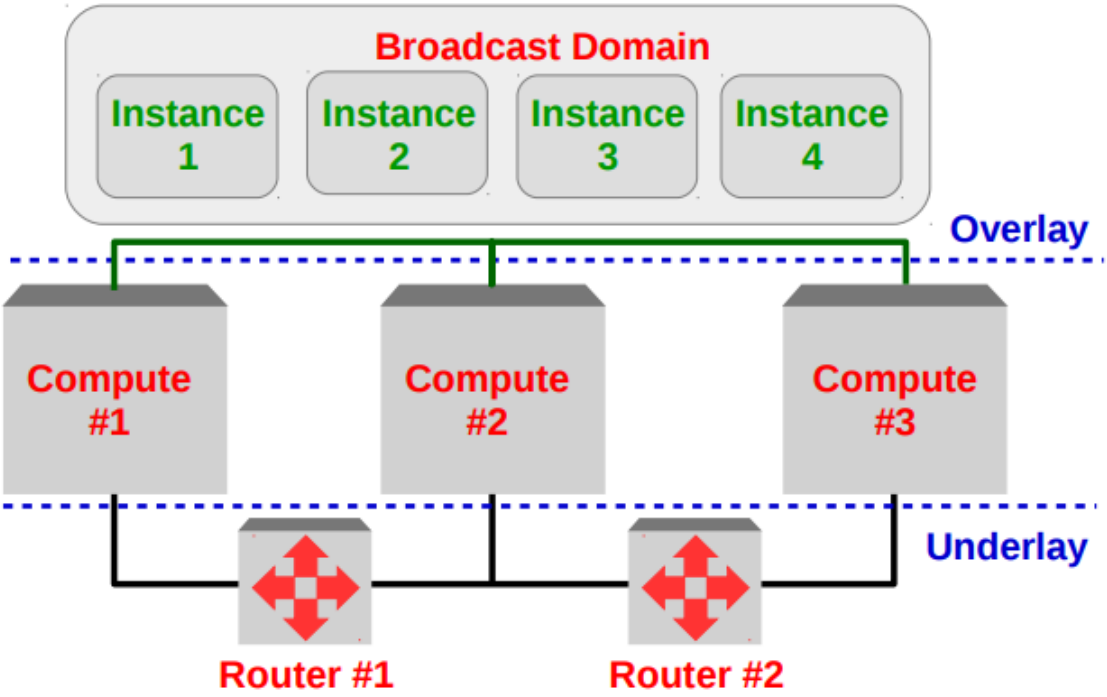


Рисунок 2.2 – Neutron

SDN дозволяє користувачам щоб визначити власну мережу між розгорнутими екземплярами. У середовищі є а Кількість різних обчислювальних вузлів, пов'язаних з використанням фізичної мережі підкладки що включає функцію маршрутизації. Користувач OpenStack не знає про деталі підкладка Користувач може бачити абстрактну мережу на більш високому рівні, тобто називається Накладання мережі. SDN дозволяє Користувачеві створювати логічні мережі, які не потрібні розгляд основної фізичної мережі. Насправді користувач, швидше за все, буде не знати топологію підкладкової мережі. Сервіс Neutron керує цим шляхом взаємодії з фізичною мережею архітектура за допомогою навігаційної архітектури, яка підтримує багатьох мережних постачальників і технології. Крім того, служба Neutron також надає API для користувачів для визначення мереж та вкладень у них.

2.3.3 Swift сервіс "Об'єкт зберігання"

Сервіс Swift 'Object Storage' зображений на рисунку 2.3, забезпечує масштабованість на рівні пам'яті.

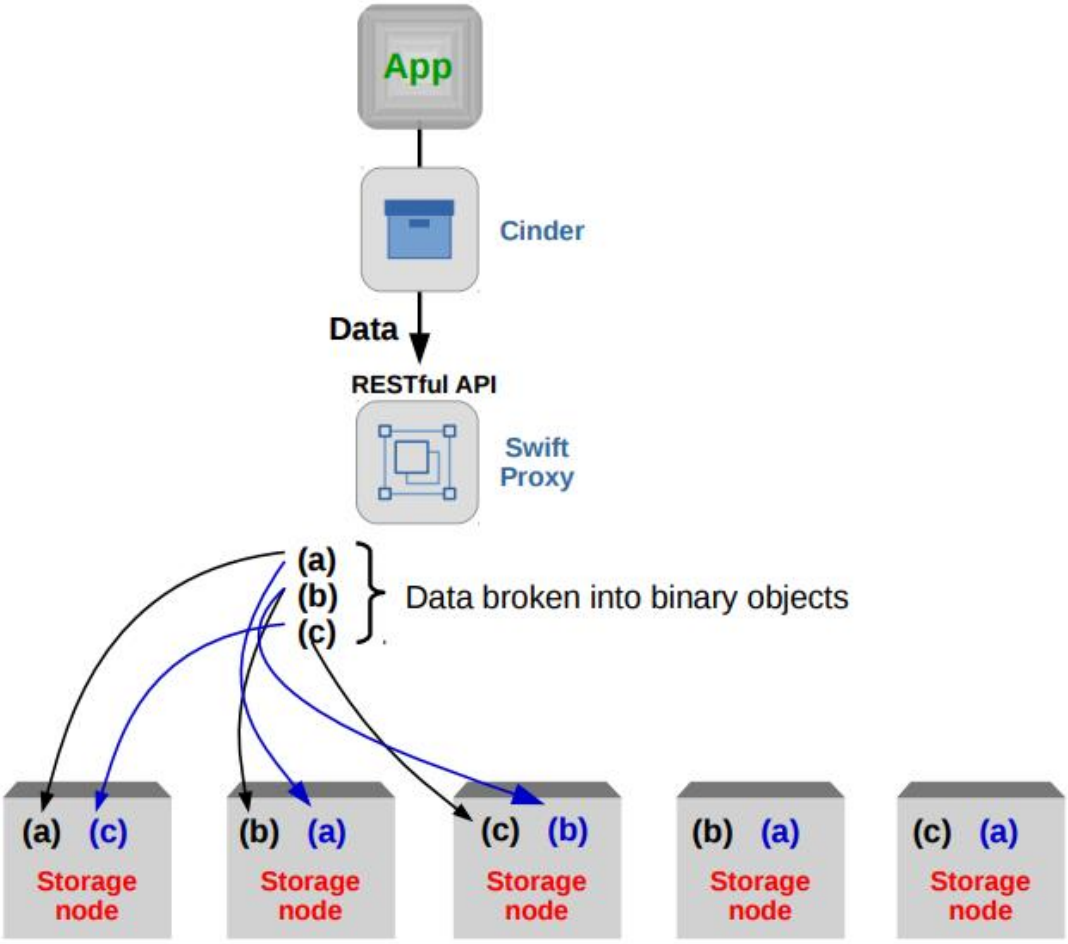


Рисунок 2.3 – Swift 'Object Storage'

Він працює з бінарними об'єктами для зберігання даних у розподіленому, відтвореному вигляді. Жорсткі диски є фізичними пристроями, вони обмежені, і вони не дуже масштабовані. Служба Swift забезпечує масштабованість, надаючи об'єктну модель збереження. Зазвичай програма, щоб писати дані, записує до файлу. У середовищі OpenStack додаток записує до файлу, але не на жорсткий диск, програма через службу Cinder 'Block Storage' інтерфейсу з Swift 'Object Storage' через RESTful API, яка, у свою чергу, може зв'язатися з багатьма багатьма вузли зберігання. Swift використовує проксі-сервіс, який, отримуючи дані від Cinder,

створює шматки даних, які називаються бінарними об'єктами. Як показано в ілюстрації 3, отримані дані розбиті на три бінарні об'єкти (a), (b) та (c). У Swift бінарні об'єкти (a) можуть бути збережені в першому вузлі зберігання, а бінарний об'єкт (b) у другому вузлі зберігання з бінарним об'єктом (c), який зберігається у третьому вузлі зберігання. Для створення відмовостійкості Swift включає в себе алгоритм реплікації, який зберігає бінарні об'єкти на декількох вузлах зберігання. За замовчуванням він робить це тричі, але це можливо зробити, якщо потрібно. Ефективність також досягається, оскільки, коли додаток потребує отримання даних, він звертається до проксі-сервера Swift через службу Cinder, яка використовує розширений алгоритм, щоб визначити, де саме розташовані бінарні об'єкти. Потім він посилає дзвінки на всі вузли зберігання, які беруть участь, вони здатні працювати паралельно. Дані надходитимуть до проксі-сервера Swift, а потім – швидко та ефективно до програми через Cinder.

Якщо вузли зберігання є, наприклад, один терабайт (ТБ) кожен, і зберігання буде низьким, можна просто додати інші Swift-вузли зберігання, а бінарні об'єкти балансуватимуться, як вказано в конфігурації зберігання Swift. Проксі Swift повідомляється з використанням Restful API. REST – це стандартний спосіб спілкування в середовищі OpenStack. Програма не записує файл до файлової системи, він використовує виклик RESTful API, який розуміється Swift proxy. Цей API дозволяє створювати, читати, оновлювати та видаляти (CRUD) функції. RESTful API є рідною мовою OpenStack, що робить Swift власним вибором для зберігання об'єктів у OpenStack. Альтернативою використанню служби Swift 'Object Storage' є Ceph. Ceph – це подібний магазин розподілених об'єктів і файлова система, призначена для забезпечення відмінної продуктивності, надійності та масштабованості.

2.3.4 Cinder сервіс "Блокування зберігання"

Зберігання блоків забезпечує постійне зберігання екземплярів. За замовчуванням зберігання в екземплярах VM є ефемерним, нестійким. Іншими словами, вміст VM втрачається, коли цей VM вимкнений. Потік дозволяє адміністраторам приєднувати додаткові стійкі блокові пристрої до таких випадків, коли дані можна зберігати. Інтерфейс Cinder задає ряд дискретних функцій, включаючи базові функції, такі як створення гучності, видалення гучності та приєднання. Є також більш розширені функції, які дозволяють збільшити обсяги, робити знімки, клонувати обсяги та створювати обсяги з зображення VM. Служба "Сіндер" також може використовувати різні задні пристрої. Це може бути локальне сховище, як локальний диспетчер логічних обсягів Linux (LVM), або це може включати Swift і Ceph об'єкт зберігання, а також для збільшення масштабованості

2.3.5 "Identity" сервіс ключового тракту

Сервіс Keystone Identity є основним елементом у OpenStack і використовується для автентифікації та авторизації. Тут також перелічено всі поточні послуги та кінцеві точки. Для доступу до Nova, наприклад, вона повинна бути визначена як служба в Keystone. Конечна точка забезпечує уніфікований локатор ресурсів (URL), який забезпечує доступ до певної служби. У Keystone створюються користувачі та ролі, і вони призначаються для проектів. Проект зазвичай є клієнтом OpenStack. Якщо OpenStack використовується як загальнодоступне хмара, різні компанії, які купують простір хмар, відрізняються один від одного від Проекту, який містить ресурси, що використовуються цим клієнтом. В межах середовища проекту для облікових записів користувачів призначаються певні ролі, і в залежності від ролі, призначеної для певної облікового запису користувача, користувачі матимуть більше або менше параметрів у OpenStack. Keystone використовує базу даних для зберігання інформації. База даних MariaDB є стандартною, однак інші бази даних можуть бути використані як Oracle DB або просто каталог легкого каталогу доступу до каталогів (LDAP).

2.3.6 Glance сервіс "Магазин зображень"

Служба Glance Image store використовується для зберігання образів віртуального диска. Віртуальні машини, які є фактичними прикладами, не встановлюються кожного разу, замість цього їх виводять з зображення. Glance забезпечує зберігання зображення. Якщо адміністратор хоче завантажити екземпляр, то екземпляр завантажиться з магазину зображення Glance. Для масштабованості, хоча це не є суто необхідним, у магазині зображень Glance зазвичай використовується Swift або Ceph Object Storage як back-end. У невеликих розгортаннях можна використовувати локальне запам'ятовування як задній пристрій для Glance, але потім все пов'язано з фізичним сервером, який містить фізичні зображення, які не дуже масштабовані.

2.3.6. Інші сервіси

Існує ще низка інших сервісів, серед них:

- Horizon – Надає інформаційну панель, яка є веб-інтерфейсом для керування службою OpenStack;
- Heat – надає послуги для оркестровки складних додатків у хмарі, використовуючи декларативний формат шаблону за допомогою OpenStack-рідного API REST;
- Ceilometer – це частина проекту "Телеметрія" та надає послуги з збору даних для білінгу та інших цілей;
- Trove – Створення та керування базами даних;
- Sahara – надає простий спосіб надання кластеру додатків, що потребує даних;
- Magnum – забезпечує такі двигуни, як Docker Swarm, Kubernetes та Apache Mesos для контейнерних (КТ). Magnum використовує Heat для оркестровки зображення ОС, що містить Docker і Kubernetes, і запускає це зображення в будь-яких VM або в голі метали в конфігурації кластеру;

- Ironica – програма створення готових металів і була розроблена для розгортання фізичних машин (а не віртуальних машин).

3 РОЗРОБКА СТРУКТУРНОЇ ТА ФУНЦІОНАЛЬНОЇ СХЕМИ ЛАБОРАТОРНОГО СТЕНДУ ХМАРНОЇ ІТ-ІНРАСТРУКТУРИ

3.1 Розробка структурної схема лабораторного стенду хмарної ІТ-інфраструктури

Структурна схема лабораторного стенду хмарної ІТ-інфраструктури зображена на рисунку 3.1 та в Додатку А.

Рисунок 3.1 – Структурна схема лабораторного стенду хмарної ІТ-інфраструктури

Структурна схема лабораторного стенду хмарної ІТ-інфраструктури слкадєьбся з підсистеми доступу до інтернет мережі, яка з'єднує вузли до інтернету, приватна мережа яка забезпечує керування внутрішніми зв'язками між об'єктами і, нарешті, зовнішня мережа, щоб забезпечити інтерфейс для віртуальних машин. На

хостовій системі необхідно ввести правила NAT Masquerade, щоб дозволити мережі провайдера отримувати доступ до Інтернету.

3.1.1Підсистема управління

Підсистема управління запускає такі служби OpenStack:

а) Keystone, служба ідентифікації;

б) Glance, сервіс зображень.

в) управління частинами:

1) Nova, Обчислювальна служба;
2) Neuron, мережевий сервіс;
3) Horizon, служба Dashboard.

г) служби підтримки OpenStack, такі як:

4) база даних Structured Query Language (SQL);
5) черга повідомлень кролика (RabbitMQ);
6) протокол мережевого часу (NTP).

Підсистема управління також запускає додаткові послуги, такі як:

- Cinder, сервіс блокових сховищ;
- Swift служба зберігання об'єктів;
- Heat, сервіс оркестрації;
- телеметричні послуги.

3.1.2 Обчислювальний вузол

Обчислювальний вузол запускає частину обчислення гіпервізора, яка експлуатує екземпляри, це типово для гіпервізору KVM / QEMU. Багато обчислювальних вузлів можуть бути використані для роботи системи. Включає в себе N кількість серверів на основі підключення DHCP.

3.1.3 Емулятори мереж.

Обчислювальний вузол також запускає агент служби мережевих служб Neutron, який з'єднує екземпляри з віртуальними мережами та надає послуги брандмауера для екземплярів через групи безпеки. Скрипти автоматично створюють дві мережі – мережу керування мережею і постачальника або зовнішню мережу. Вони називаються по-різному залежно від використовуваного гіпервізора.

3.2 Розробка функціональної схеми лабораторного стенду хмарної IT-інфраструктури

Функціональна схема лабораторного стенду хмарної IT-інфраструктури зображена на рисунку 3.2 та в Додатку Б.

На функціональній схемі зображені пристрої, що виконують певні операції та зв’язки між ними, по яким передаються дані у аналоговому та цифровому вигляді. Дана схема є більш конкретною, по відношенню до структурної схеми. В цьому розділі описані кожні з функціональних вузлів для побудови лабораторного стенду хмарної IT-інфраструктури.

Розроблювана функціональна схема складається з п’яти основних блоків: підсистеми управління, обчислювального вузла, двох віртуальних мереж, та підсистеми доступу до інтернет мережі. Підсистема доступу до інтернет мережі з'єднує вузли до інтернету, і служить головним вузлом для передачі інформації між обчислювальним вузлом і підсистемою управління.

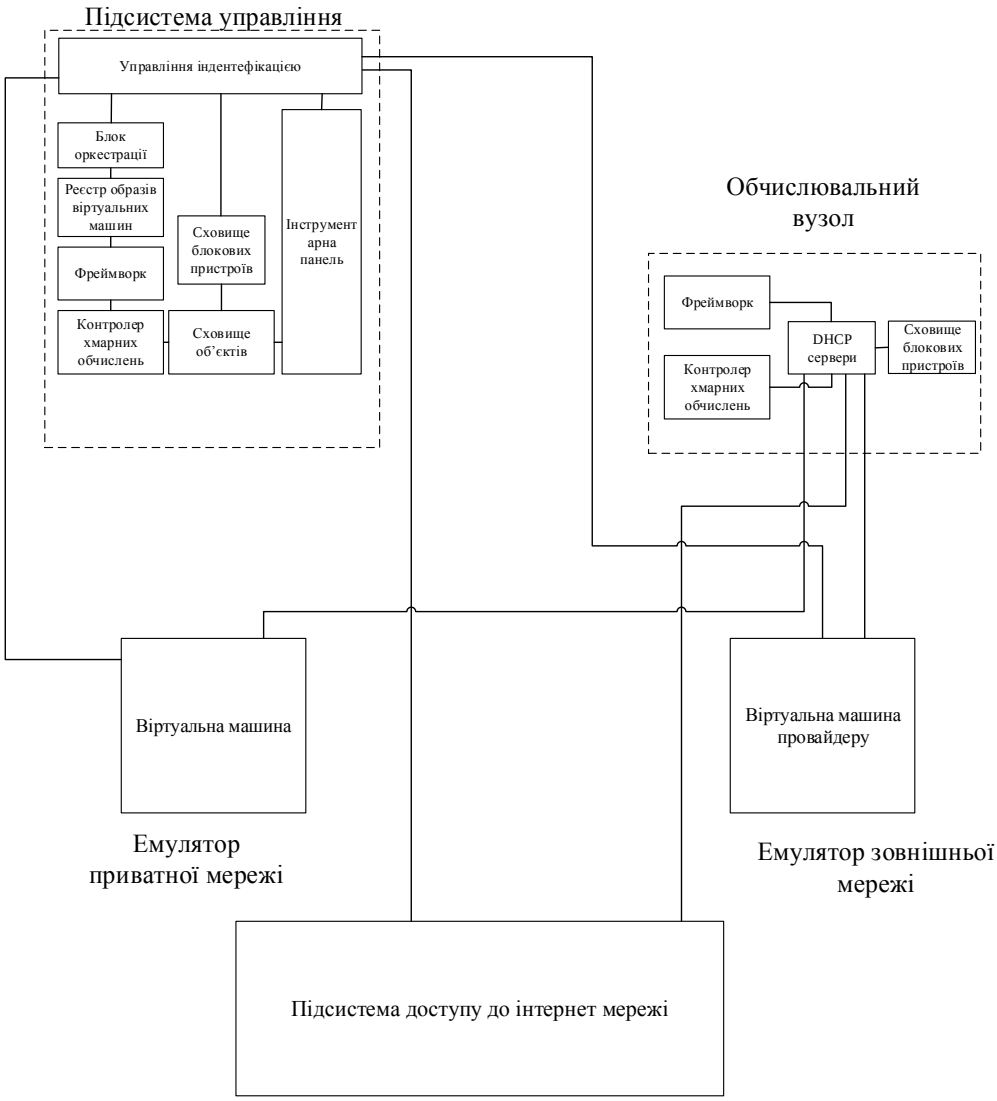


Рисунок 3.2 – Функціональна схема лабораторного стенду хмарної IT-інфраструктури

У обчислювальному узлі присутні такі елементи як, DHCP сервери, сховище блокових пристроїв, контролеру хмарних обчислень та фреймворку.

Віртуальні мережі служать доступом до роботи з обчислювальним вузлом,а також з підсистему управління.

Більш подрібне про підсистему управління у додатку В та на рисунку 3.3

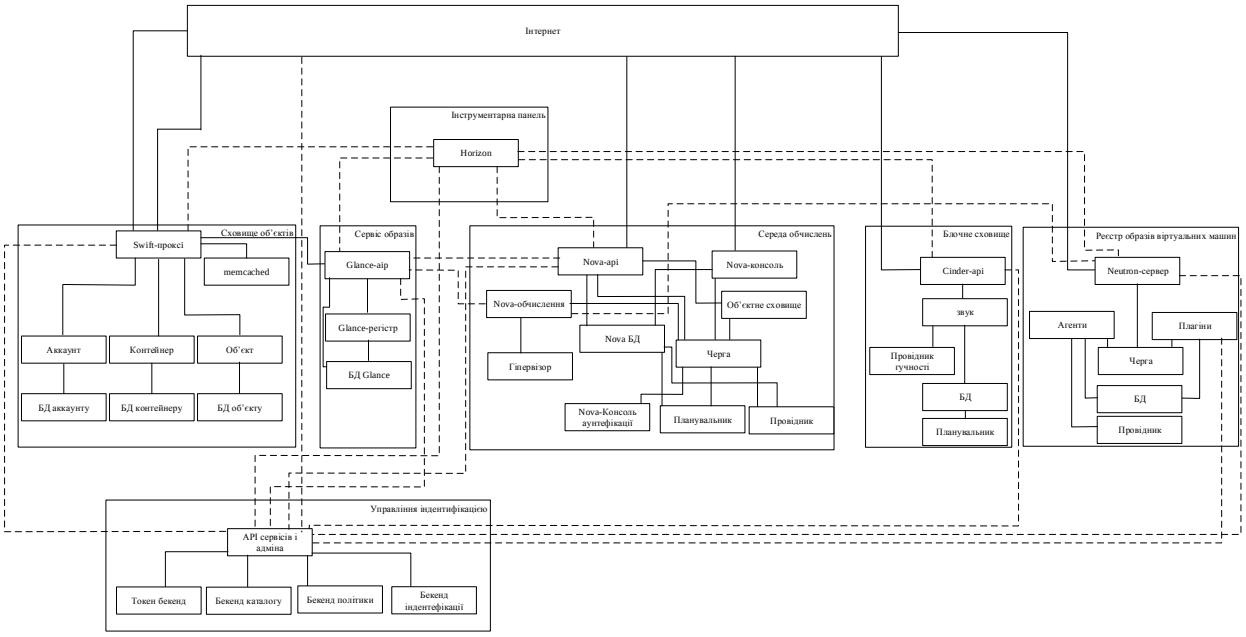


Рисунок 3.3 – Функціональна схема лабораторного стенду хмарної ІТ-інфраструктури

Розглянемо взаємодію між декількома компонентами всередині OpenStack:

- інструментарна панель (Horizon) забезпечує інтерфейс для всіх служб OpenStack.
- compute ("Nova") отримує зображення віртуального диска ("Glance"), прикріплює пов'язані метадані, а також перетворює запити кінцевого користувача API в виконувані примірники.
- мережа ("Neutron") забезпечує віртуальні мережі для обчислення, що дозволяє користувачам створювати власні мережі, а потім зв'язати їх з екземплярами.
- блочне сховище ("Cinder") забезпечує постійні обсяги зберігання для обчислюваних підстанцій.
- зображення ("Glance") дозволяє зберігати фактичні файли віртуальних дисків.
- управління індентифікацією ("Keystone") забезпечує автентифікацію та авторизацію для всіх служб OpenStack.

Потік запиту для створення екземпляра виглядає таким чином:

а) Інформаційна панель або CLI отримує обліковий запис користувача і робить виклик REST для Keystone для автентифікації;

б) Keystone перевіряє автентичність облікових даних і генерує і відправляє назад аутентостанцію, яка буде використовуватися для надсилання запиту до інших компонентів за допомогою REST-дзвінка;

в) інформаційна панель або CLI перетворюють новий запит на екземпляр, вказаний у формі 'launch instance' або 'nova-boot', у запит REST API і відправляє його в nova-api;

г) Nova-api отримує запит і відправляє запит на авторизацію автентифікації та дозвіл на доступ до ключового тракту;

д) Keystone перевіряє токен і надсилає оновлені заголовки auth з ролями та правами;

е) Nova-api взаємодіє з новою базою даних;

ж) Створює початковий запис db для нового екземпляра;

з) Nova-api надсилає запит rpc.call на nova-scheduler, за винятком, щоб отримати оновлений запис екземпляра з вказаним ідентифікатором хоста;

и) Nova-планувальник вибирає запит з черг;

к) Nova-планувальник взаємодіє з новою базою даних, щоб знайти відповідний хост за допомогою фільтрації та зважування;

л) повертає оновлений запис екземпляра з відповідним ідентифікатором хоста після фільтрування та зважування;

м) Nova-планувальник надсилає запит rpc.cast на nova-compute для запуску екземпляра на відповідному хості;

н) Nova-обчислення вибирає запит з черги;

о) Nova-обчислення надсилає запит rpc.call на nova-conductor, щоб отримати інформацію про екземпляр, таку як ідентифікатор хоста та смак (Ram, CPU, Disk);

п) Nova-провідник вибирає запит з черги;

р) Nova-диригент взаємодіє з новою базою даних;

с) Поверніть інформацію про екземпляр;

т) Nova-обчислення вибирає інформацію про екземпляр з черги;

у) Nova-обчислюється виклик REST, передаючи аут-токену на огляд-api, щоб отримати Image URI за ідентифікатором зображення з погляду та завантажити зображення із сховища зображень;

ф) Glance-api перевіряє авт-токен з ключем;

х) нові обчислення отримують метадані зображення;

ц) Nova-обчислює робить REST-виклик, передаючи аут-токен в Network API для виділення та налаштування мережі таким чином, щоб екземпляр отримував IP-адресу;

ч) Neuron сервер перевіряє авт-токен з ключем;

ш) Nova обчислення отримують інформацію про мережу;

щ) Nova-обчислення робить виклик REST, передаючи аут-токен в Volume API, щоб прикріпити томи до екземпляра;

ь) Cinder-api перевіряє авт-токен із трапецеїалізмом;

ю) Nova-обчислення отримує інформацію про блокування пам'яті;

я) Nova-compute генерує дані для драйвера гіпервізора і виконує запит на гіпервізора (через libvirt або api).

4 ПРОЦЕСС РОЗГОРТАННЯ ТА УПРАВЛІННЯ ЛАБОРАТОРНОГО СТЕНДУ

4.1 Скрипти для встановлення OpenStack

OpenStack Training Labs – це набір скриптів, які встановлюють робочий кластер OpenStack на комп'ютері. Це автоматизований, швидкий, надійний та відтворюваний спосіб відстеження інструкцій з встановлення OpenStack для створення кластера з використанням KVM / QEMU або віртуальних машин VirtualBox. Навчальні лабораторії OpenStack повинні працювати на найпоширеніших апаратних засобах (настільні / ноутбуки) з коробки, що мають мінімум 8 Гб оперативної пам'яті (ОЗП). Ці нотатки, однак, працюють над припущенням про 16 ГБ оперативної пам'яті з 1 ТБ накопичувачем.

На сучасному етапі розвитку інформаційних технологій важливе місце займає синтез цифрових пристроїв на основі функціональних моделей. Його суть полягає в автоматизованому формуванні логічної структури пристрою на основі необхідної поведінки. Логічна структура описується інженером за допомогою спеціальної мови програмування – HDL (Hardware Description Language – мова опису обладнання та апаратного забезпечення). Найбільшого поширення серед мов класу HDL набули такі мови як ABEL, Verilog та VHDL. На рисунку 4.1 зображений алгоритм побудови лабораторного стенду.

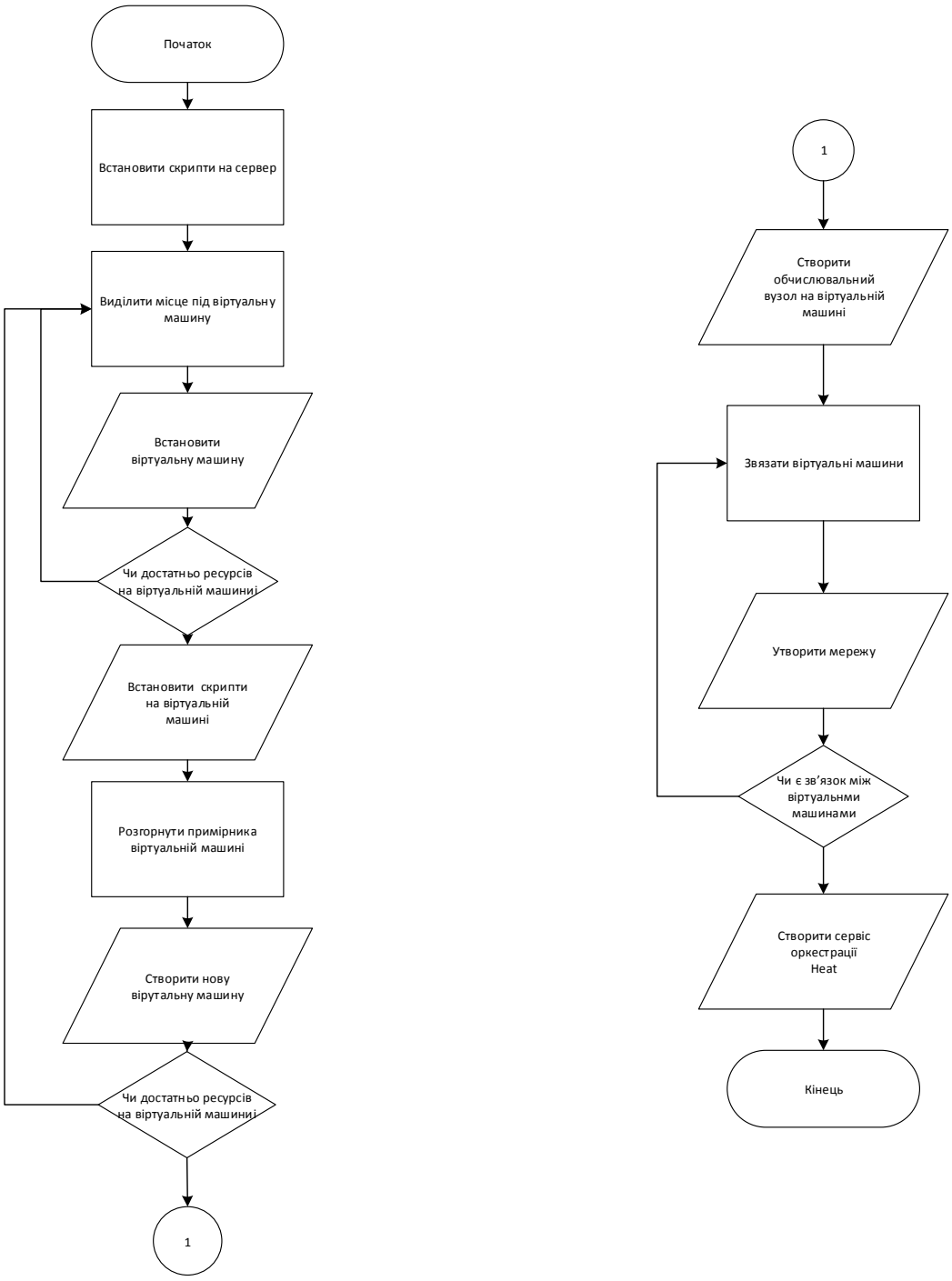


Рисунок 4.1 – Алгоритм побудови лабораторного стенду

4.1.1Перевірка перед встановленням

Увімкніть підтримку віртуалізації в BIOS.

Щоб підтримувати гостьові комп'ютери з віртуальною машиною (HVM), потрібно ввімкнути розширення віртуалізації у базовій системі вводу / виводу

(BIOS). У BIOS параметр "Віртуалізація" з'являється в розділі "Додаткові функції чіпсета" як один з таких:

- включити технологію віртуалізації – архітектури x86 (VT-x);
- включити Intel VT;
- технологія Vanderpool;

Також включити:

- технологія віртуалізації Intel для напрямних входів / виходів (VT-d).

Підтвердьте, що віртуалізація апаратного забезпечення тепер підтримується центральним процесором (CPU) за допомогою пошуку віртуальної машини eXtensions (VMX), щоб перевірити, чи має комп'ютер процесор Intel або Secure Virtual Machine (SVM) для підтримки AMD, якщо обладнання має AMD процесор Переконайтеся, що процесор підтримує віртуалізацію апаратного забезпечення. 0 означає, що процесор не підтримує апаратну віртуалізацію, тоді як> 0 означає, що це робиться, але його потрібно ввімкнути в BIOS. (тобто. vmx або svm з'явилося x разів у виведенні команди.

```
ada:~$ egrep -c '(vmx|svm)' /proc/cpuinfo
4
```

Перевірте, чи працює 64-розрядний ядро. 0 означає, що процесор не є 64-розрядним. Довгий режим (LM) дорівнює 64-бітному ЦП.

```
ada:~$ egrep -c ' lm ' /proc/cpuinfo
8

ada:~$ uname -m
x86_64
```

4.2 Створення, розгорнення та запуск віртуальної машини в OpenStack

Розглянемо алгоритм створення віртуальної машини на рисунку 4.1

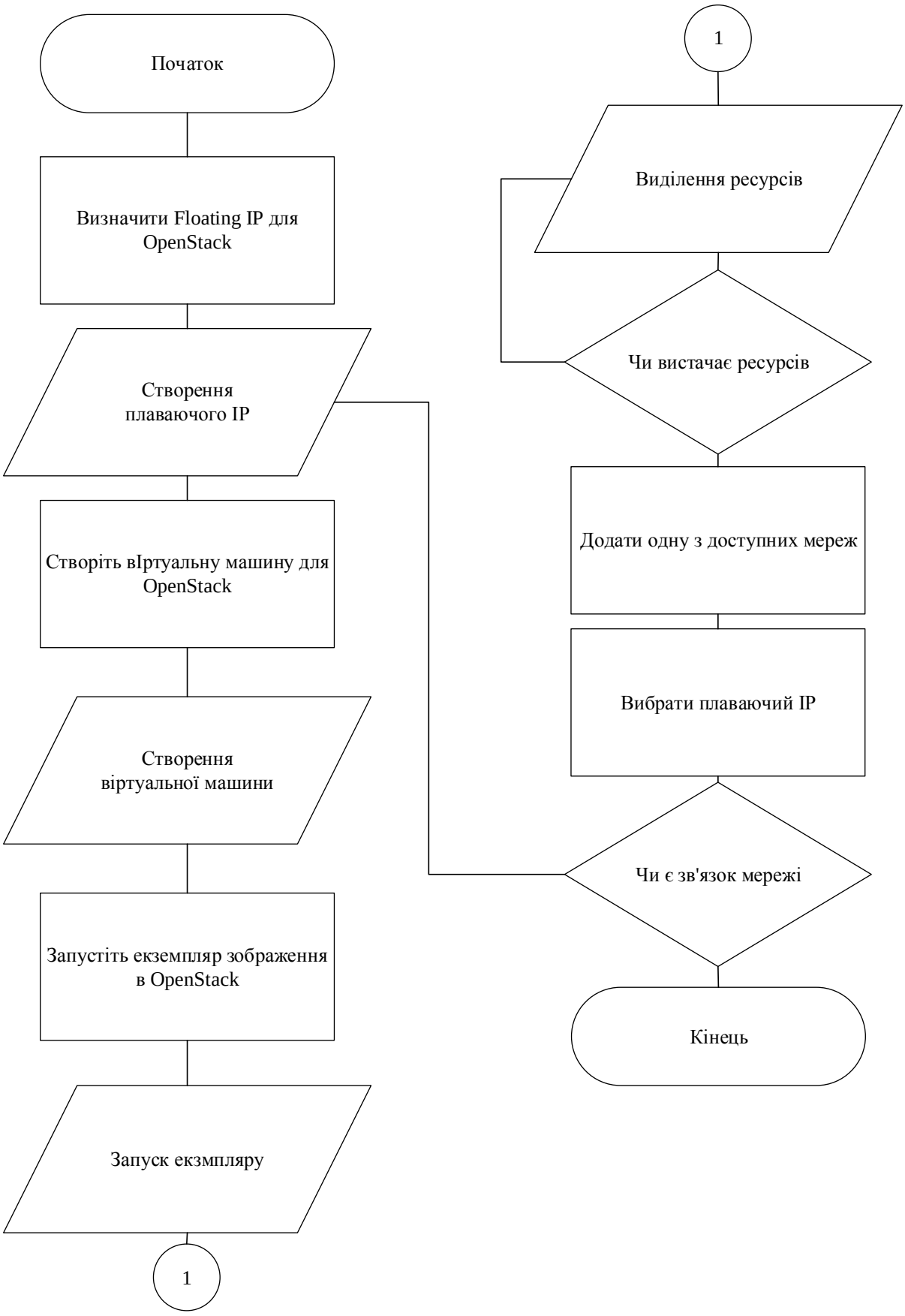


Рисунок 4.2 – Алгоритм створення віртуальної машини

4.2.1 Виділити плаваючу IP на OpenStack

Перед тим, як розгорнути зображення OpenStack, спершу потрібно переконатися, що всі елементи є на місці, і ми почнемо з розподілу плаваючого IP.

Плаваючий IP дозволяє зовнішньому доступу з зовнішніх мереж або Інтернету на віртуальну машину Openstack. Щоб створити плаваючі IP-адреси для вашого проекту, увійдіть у свій обліковий запис за допомогою облікових даних користувача та перейдіть на вкладку Проект -> Обчислювач -> Доступ і безпека -> Плаваючі IP та натисніть Allocate IP на проект.

Виберіть зовнішній басейн і натисніть кнопку IP Allocate, а IP-адреса повинна з'явитися на інформаційній панелі. Дуже добре виділити Floating IP для кожного виконуваного вами прикладу (Рис. 4.3).

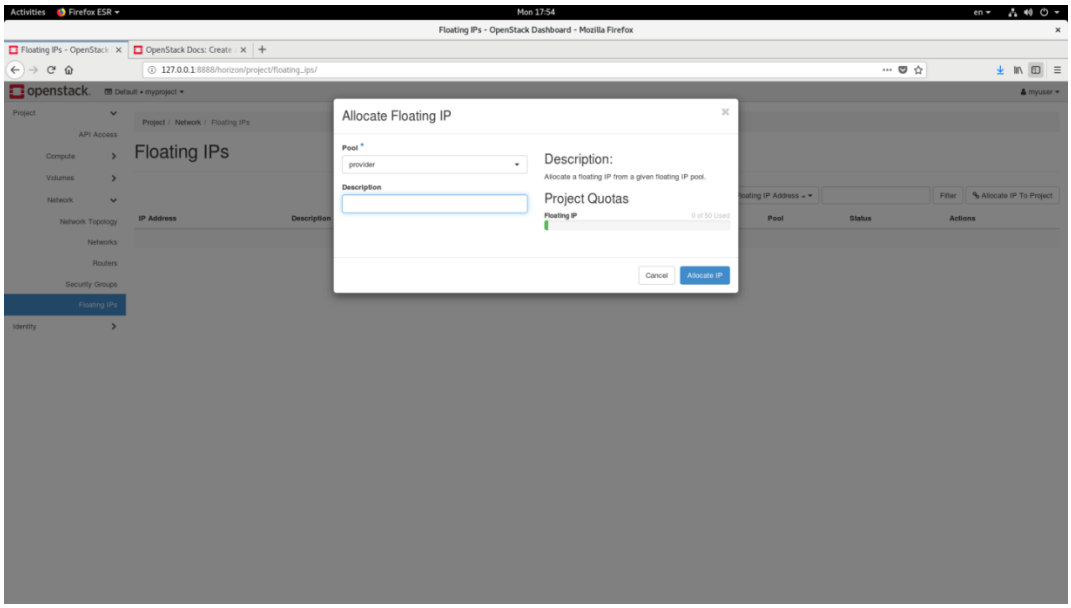


Рисунок 4.3 – Створення плаваючого IP

4.2.2 Створіть образ OpenStack

Образ OpenStack – це лише віртуальні машини, вже створені третіми сторонами. Ви можете створити власні настроювані образи на вашому комп'ютері,

встановивши ОС Linux у віртуальну машину, використовуючи інструмент віртуалізації, такий як KVM, VirtualBox, VMware або Hyper-V.

Після того як ви встановите ОС, просто перетворіть файл на сировину та завантажте його у свою інфраструктуру хмарної інфраструктури OpenStack.

Файл зображення може бути використаний безпосередньо з HTTP-посилання або завантажений локально на вашому комп'ютері та завантажений в хмару OpenStack.

Щоб створити образи, перейдіть на веб-панель OpenStack і перейдіть до Проектування -> Обчислення -> образи та натисніть кнопку Створити образ. У рядку образів скористайтеся наведеними нижче налаштуваннями та натисніть «Створити образи» після завершення (Рис 4.4)

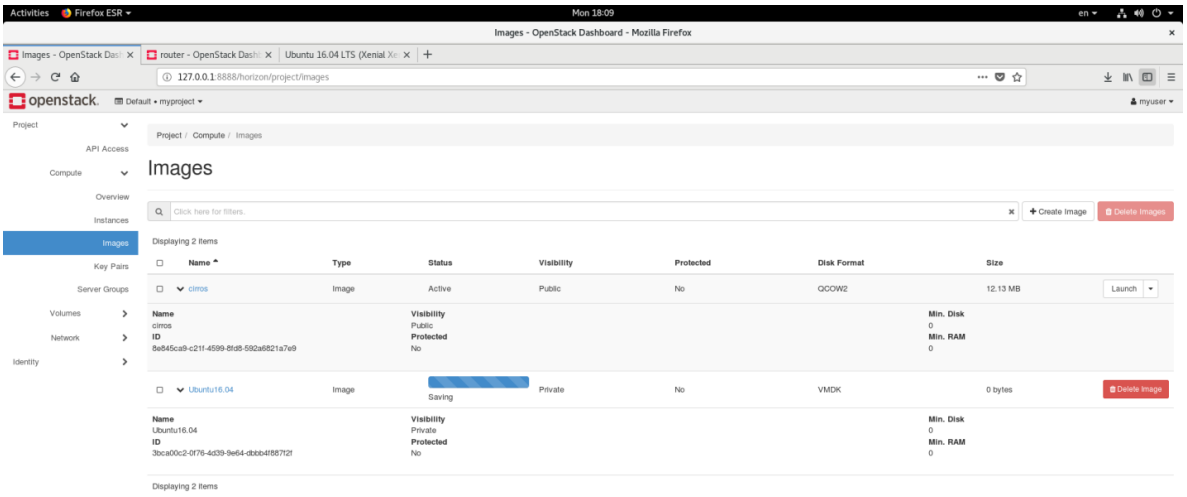


Рисунок 4.4 – Створення образу

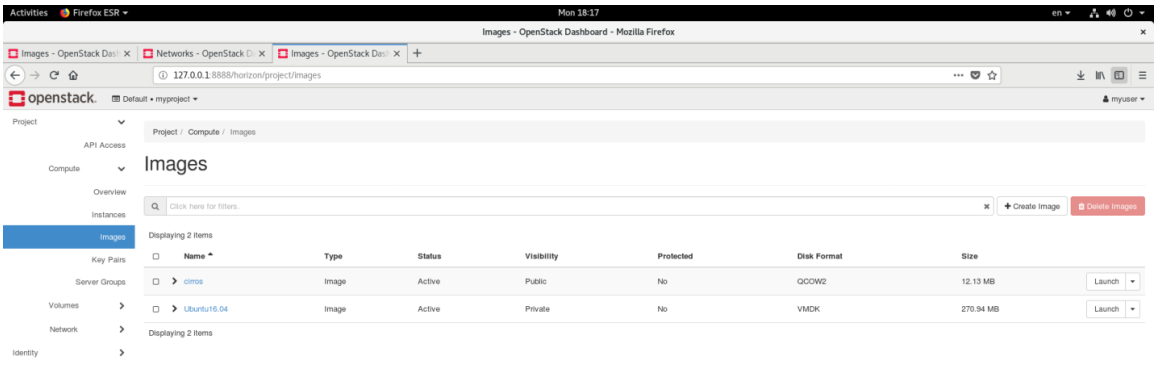


Рисунок 4.5 – Створеня образу

4.2.3. Запустіть екземпляр образу в OpenStack

Після того, як ви створили образ, який вам підходить. Тепер ви можете запустити віртуальну машину на основі образу, створеного раніше у вашому обласному середовищі.

Перейдіть до пункту «Проект -> Екземпляри» та натисніть кнопку «Запускати екземпляр», і з'явиться нове вікно.

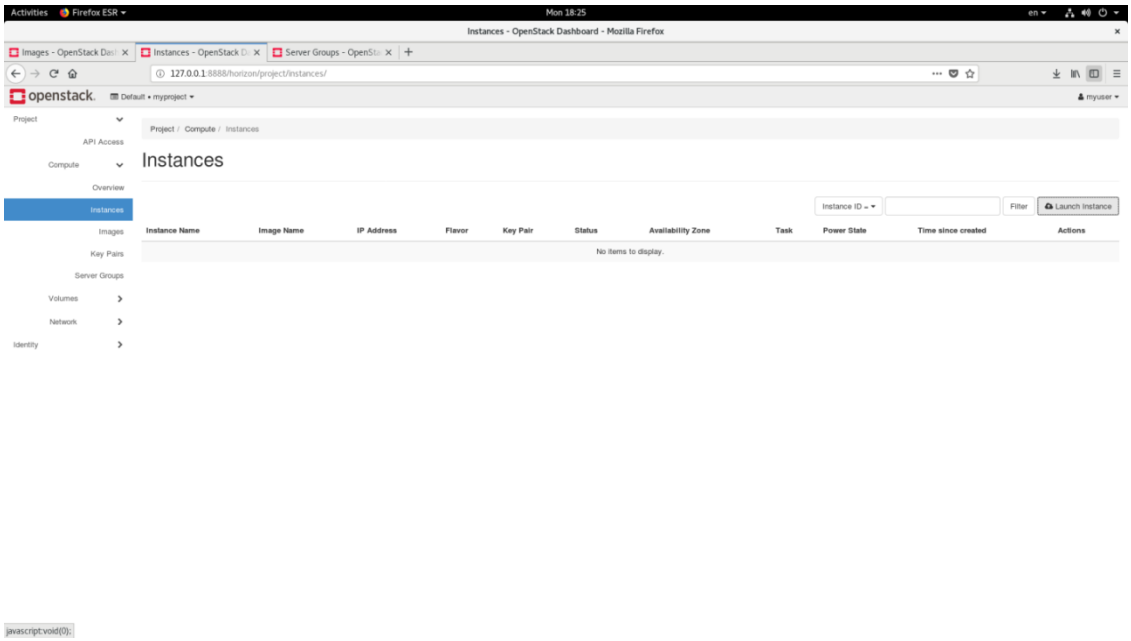


Рисунок 4.6 – Створення інстанцій

На першому екрані додайте назву свого екземпляра, залиште зону доступності для nova, використовуйте один екземпляр і натисніть кнопку Далі, щоб продовжити.

Виберіть назву описового екземпляра для вашого екземпляра, оскільки це ім'я буде використано для формування імені комп'ютера віртуальної машини.

Потім виберіть Image як джерело завантаження, додайте тестове зображення Cirros, створене раніше, натиснувши кнопку + і натисніть Next, щоб продовжити.

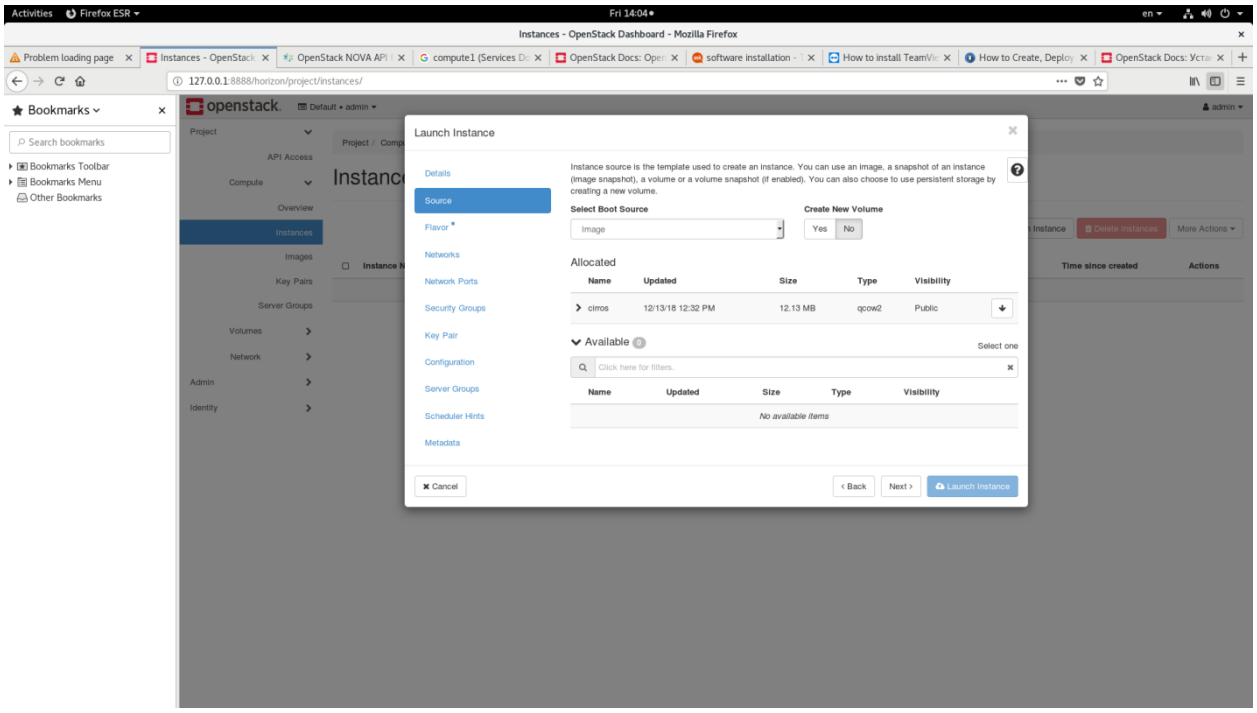


Рисунок 4.7 – Виділення ресурсів

Виділіть ресурси віртуальної машини, додавши смак, який найкраще підходить для ваших потреб, і натисніть кнопку Далі, щоб перейти.

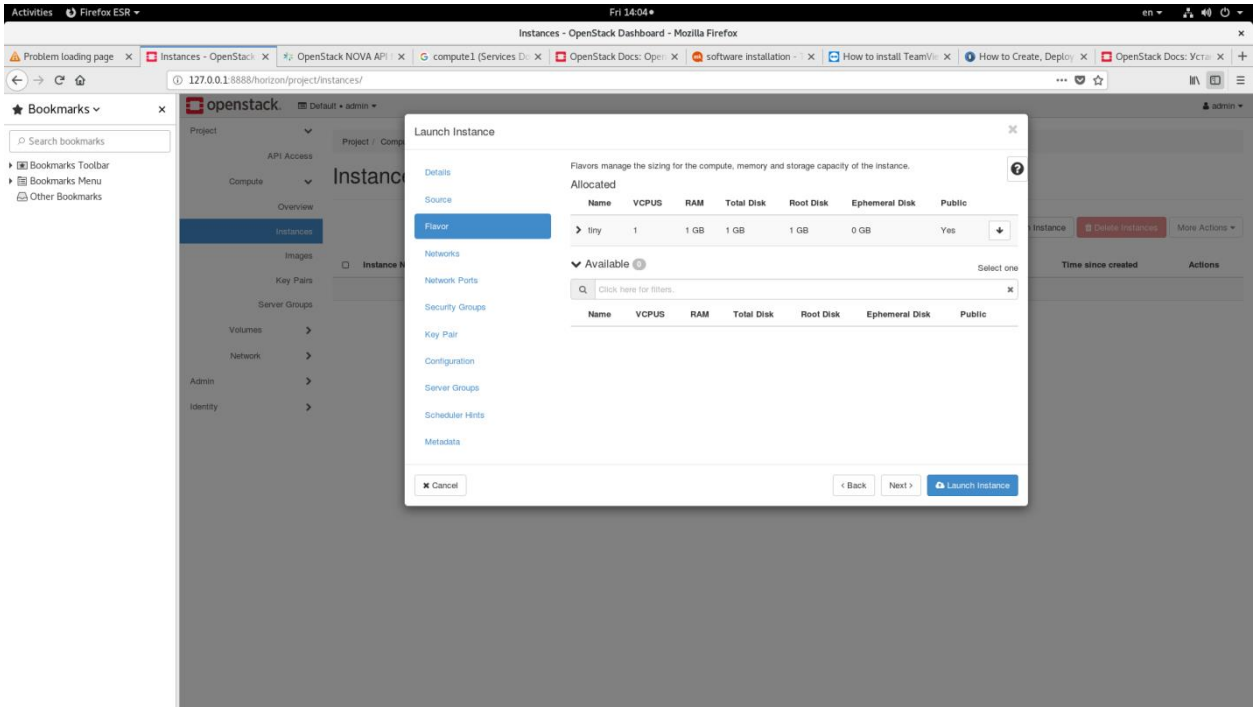


Рисунок 4.8 – Виділення ресурсів

Нарешті, додайте одну з доступних мереж OpenStack до свого екземпляра за допомогою кнопки + та натисніть «Запускати екземпляр», щоб запустити віртуальну машину (рис 4.9).

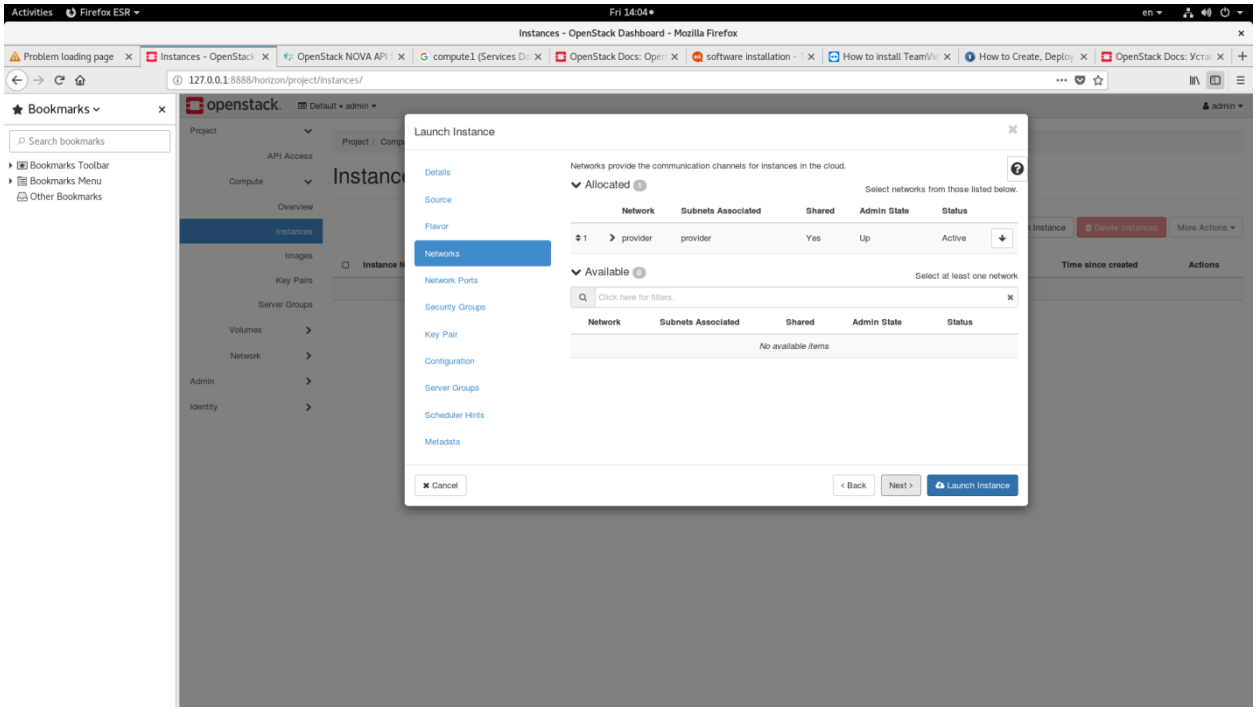


Рисунок 4.9 – Запуск еземпляру

Після запуску екземпляра, натисніть кнопку зі стрілкою праворуч від кнопки меню Створити знімок та виберіть «Асоційований плаваючий IP».

Виберіть один з плаваючого IP, створений раніше, та натисніть кнопку Associate, щоб екземпляр можна було отримати з вашої внутрішньої локальної мережі.

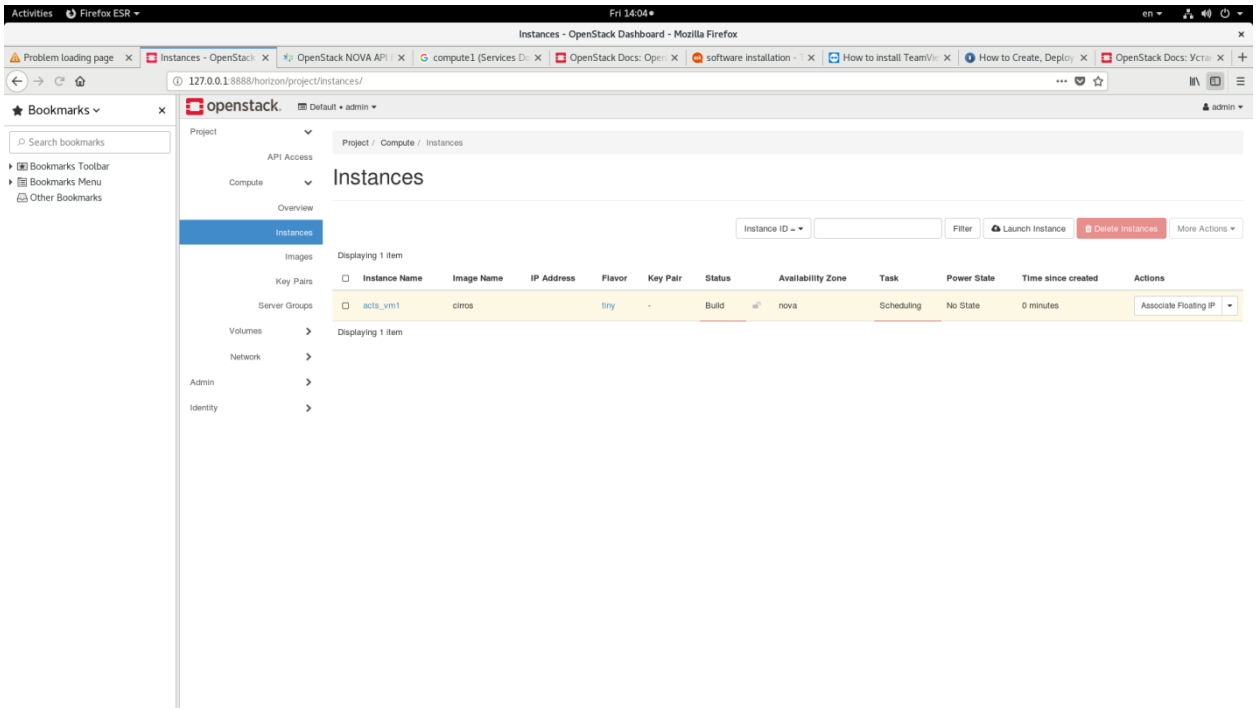


Рисунок 4.10 – Вибір IP

Щоб протестувати підключення до мережі для активної віртуальної машини, виконайте команду ping на екземплярі плаваючої IP-адреси (рис 4.10) з віддаленого комп'ютера у вашій локальній мережі.

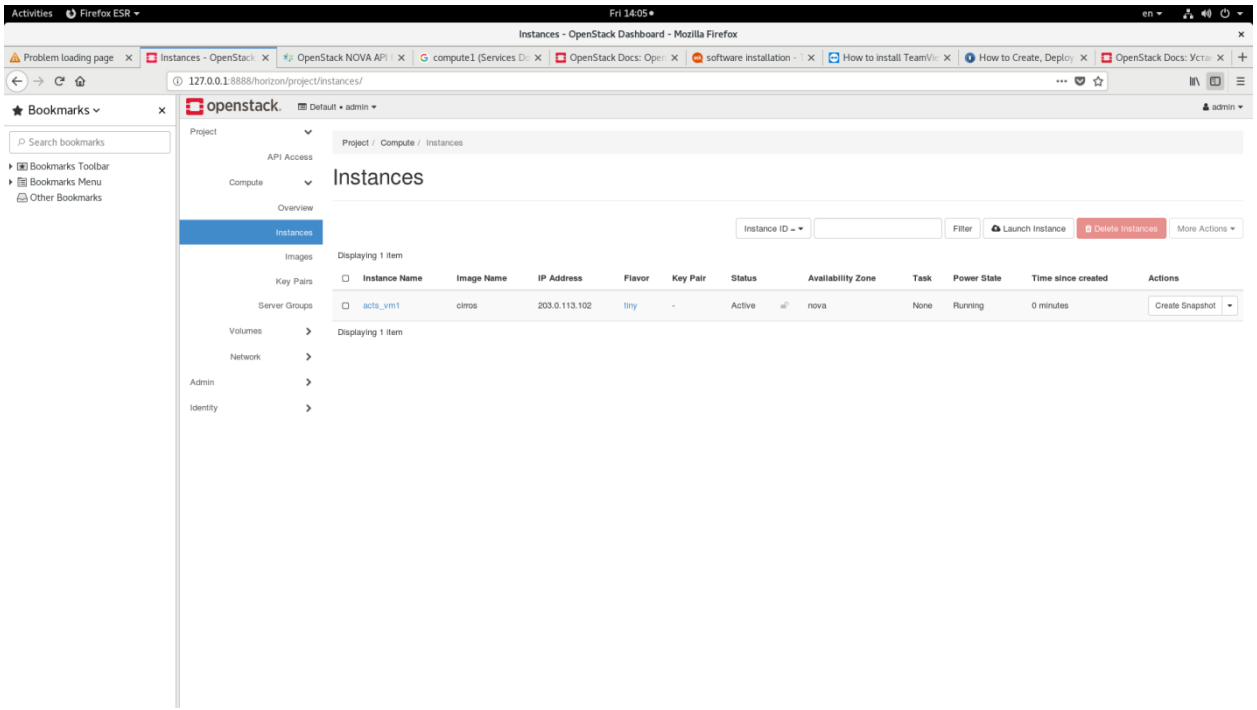


Рисунок 4.11 – Пінгування вибраного IP

Якщо у вашому прикладі немає проблем (рис 4.10), і команда ping досягає успіху, ви можете дистанційно входити через SSH на ваш екземпляр.

Використовуйте утиліту "Перегляд журналу екземплярів", щоб отримати облікові дані за умовчанням Cirros, як це показано на знімках нижче.

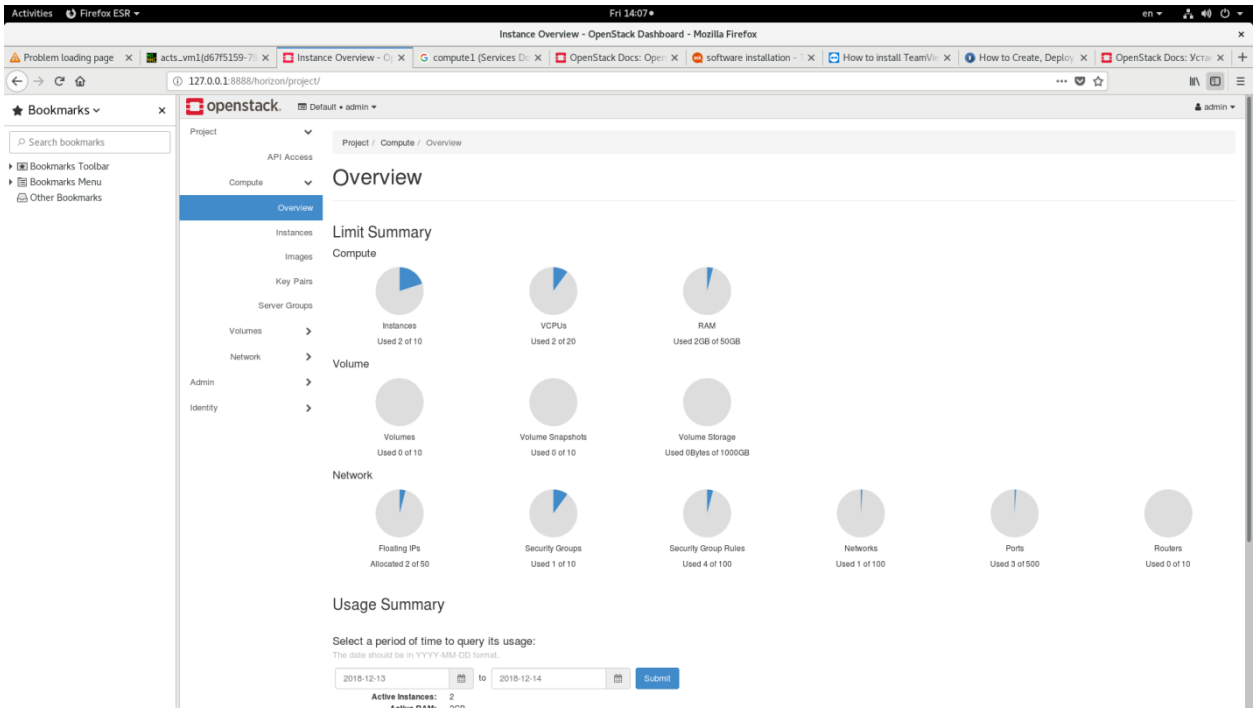


Рисунок 4.12 – Огляд ресурсів

За замовчуванням сервери імен DNS не будуть виділятися з внутрішньої мережі DHCP-сервера для вашої віртуальної машини. Ця проблема призводить до проблем з підключенням домену з екземпляра екземпляра.

Щоб вирішити цю проблему, спочатку зупиніть екземпляр і перейдіть до Проектування -> Мережа -> Мережі та відредагуйте відповідну підмережу, натиснувши кнопку Подробиці підмережі.

Додайте потрібні сервери імен DNS, збережіть конфігурацію, запустіть та підключіться до консолі екземпляра, щоб перевірити, чи застосовано нову конфігурацію, натискаючи ім'я домену. Використовуйте наступні скріншоти як посібник.

Якщо у вашій інфраструктурі є обмежені фізичні ресурси, а деякі ваші екземпляри відмовляються розпочинати, відредагуйте наступний рядок з файлу налаштувань nova та перезапустіть машину, щоб застосувати зміни.

4.3 Налаштування скриптів OpenStack на KVM / QEMU

Гіпервізор віртуальної машини на базі ядра включений в основні версії GNU / Linux, оскільки він став гіпервізором вибору в спільноті GNU / Linux, тому він доступний в різних сховищах розповсюдження(Рис. 4.13).

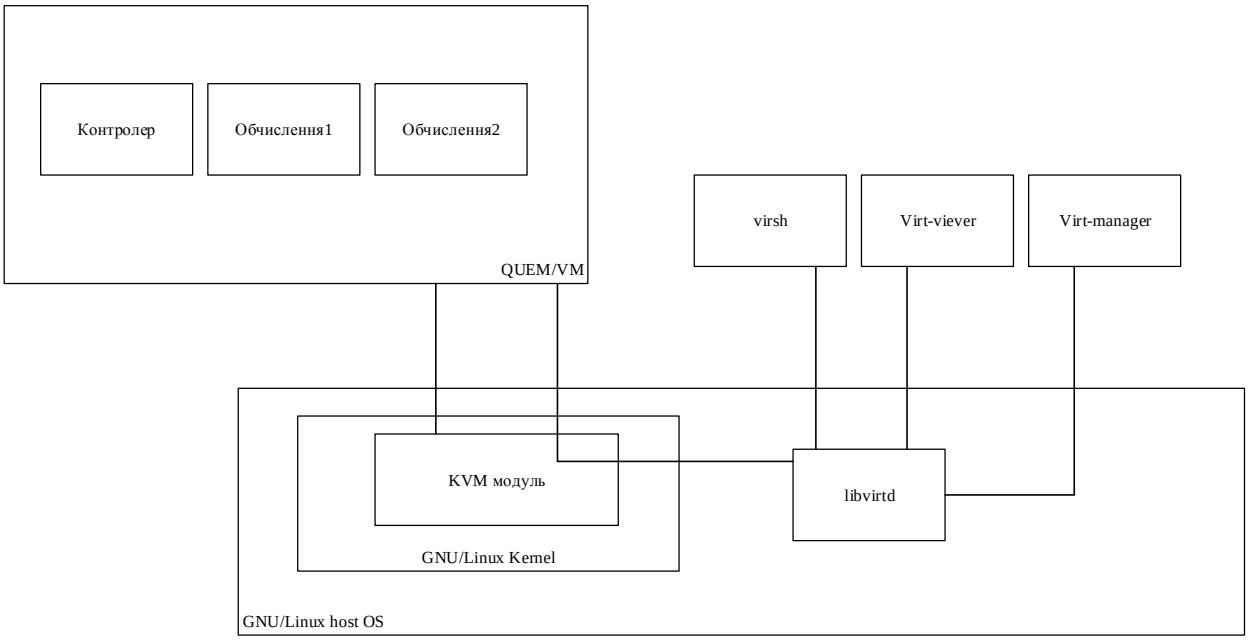


Рисунок 4.13 – Блок-схема віртуалізації KVM

Насправді KVM – це модуль ядра GNU / Linux, який дозволяє програмам в межах користувача доступ до функцій віртуалізації процесора Intel або AMD. В результаті KVM VMs фактично запускаються як користувацькі космічні процеси. KVM використовує QEMU – загальний та вихідний комп'ютерний емулятор та віртуалізатор для емуляції апаратних засобів вводу / виводу (I / O). Він може імітувати різні процесори на гостьовому процесорі та об'єднати з модулем ядра KVM, який може наближатися до власних швидкостей. Підтримуються всі

комбінації 32-бітових та 64-розрядних серверів та гостьових систем, крім 64-розрядних гостей на 32-розрядних хостах.

KVM управляється через libvirt API та інструменти, такі як virsh, virtinstall, virt-clone, virtviewer і virt-manager.

KVM – це гіпервізор типу 1, який працює безпосередньо на апаратній системі x86. Інтерфейс GNU / Linux робить вигляд, що на ньому працює гіпервізор, але інфекції кожна віртуальна машина працює на "голій метал" з операційною системою GNU / Linux, що надає панель запуску для гіпервізора, а потім здійснює взаємодію з кооперацією з гіпервізором.

На апаратній системі x86 KVM покладається на інструкції з віртуалізації обладнання, які вбудовані в процесори, і тому ці додаткові функції чіпсета повинні бути включені. Використовуючи ці вказівки, гіпервізор і кожна гостьова VM запускаються безпосередньо голий метал, і більшість перекладів ресурсу виконуються обладнанням.

Libvirt надає інструменти командного рядка під командою root, тоді як virt-manager забезпечує графічний інструмент. Ця лабораторія в основному працює в режимі без головного режиму, тобто він не вимагає графічного інструменту.

4.3.1Установка скриптів

Для роботи KVM потрібен ряд елементів. Встановити пакети KVM:

- libvirt – набір інструментів C для взаємодії з можливостями віртуалізації GNU / Linux. Бібліотека надає C API для різних механізмів віртуалізації і в даний час підтримує QEMU, KVM, XEN, OpenVZ, LXC і VirtualBox;
- qemu-kvm – дозволяє працювати з декількома віртуальними комп'ютерами, кожен з яких використовує незмінені версії GNU / Linux або Windows на апаратурі X86. У кожного VM є приватне віртуалізоване обладнання: мережева карта, диск, графічний адаптер і т. Д;
- libguestfs-tools – бібліотека, яка дозволяє отримувати доступ до образів гостьових дисків та їх модифікацію;

- virt-manager – графічний інтерфейс користувача.

```
ada:~$ sudo apt-get install qemu-kvm libvirt-bin libguestfs-tools
virt-manager
[sudo] password for alovelace: babbage
```

4.3.2 Утиліти GNU / Linux Bridge

Без моста VM-сервери KVM матимуть лише мережевий доступ до інших віртуальних машин на одному сервері та самому хості через спільну приватну мережу. Щоб дозволити доступ до віртуальних машин до локальної мережі, створіть мережний міст на хості.

```
ada:~$ sudo apt-get install bridge-utils
ada:~$ sudo usermod -aG libvirt `id -un`
```

4.3.2 Вірт-менеджер

Це може здатися дивним, але важливо запустити virt-manager. Це запускає QEMU для створення пулу за замовчуванням для зберігання. Оскільки сервер безголовий, це потрібно виконувати, використовуючи переадресацію Secure Sheell (SSH) X11.

SSH для хоста за допомогою таких перемикачів:

- M поміщає клієнта SSH в основний режим для спільного доступу до з'єднання.
- Y Дозволяє надійний переадресація X11. Надруковані пересилання X11 не піддаються до елементів керування розширенням X11 SECURITY.

Встановіть уніфікований URI для virsh за замовчуванням, що виключає необхідність використання довготривалої команди virsh для гіпервізора KVM / QEMU. Увімкнути, запустивши файл.

```
ada:~$ cat << EOM >> ~/.bashrc

# Variable to set virsh default URI to QEMU
VIRSH_DEFAULT_CONNECT_URI='qemu:///system'
EOM

 ada:~$ . .bashrc

 ada:~$ echo $VIRSH_DEFAULT_CONNECT_URI
 qemu:///system
```

Тепер підключіться до оболонки virsh на гіпервізор KVM / QMEU

```
ada:~$ virsh

virsh # uri
qemu:///system
```

Також можна запускати virsh, що віддає перевагу безпосередньо з оболонки.

```
ada:~$ virsh uri
qemu:///system
```

Перевірте, чи існує пул резервування за умовчанням.

```
virsh # pool-list
 Name                 State      Autostart
-------------------------------------------
 default              active     yes
```

4.3.3 Введення конструкції

Кластер побудований у три етапи:

а) завантажте зображення ОС;

б) створіть базовий диск, від 40 до 50 хвилин;

в) створіть контролери та compute1 VM вузлів на основі базового диска, приблизно від 50 до 65 хвилин.

По суті, скрипти завантажують образ Ubuntu, запускають його на KVM / QEMU VM з деякими конфігураційними файлами. Сценарій відновлює адресу

Інтернет-протоколу (IP) бази бази і підключається до нього через SSH на стандартному порту 22. Він оновлює VM і встановлює відповідне програмне забезпечення кластерного програмного забезпечення OpenStack. Після базового диска команда створює два вузли віртуальних машин (контролер і обчислення) з нього.

Якщо у вас є попередній збірка і виявлено, що необхідно відновити базовий диск, просто видаліть файл диска в каталозі $ OS_ST / img. Це змусить завантажити нового базового диска, інакше, якщо скрипт знайде існуючий базовий диск, він просто обійдеться цим етапом і збиратиметься для побудови контролера та обчислення вузлів та початкової конфігурації, як узагальнено в процесі збірки.

Вузол контролера:

- відредагуйте файл / etc / hosts;
- увімкнути osbash SSH ключі;
- встановити mysql;
- встановіть rabbitmq;
- встановити memcached;
- налаштувати keystone;
- налаштування Glance;
- налаштування контролера Nova;
- налаштування Neutron контролера;
- налаштуйте контролер самообслуговування;
- налаштування Horizon;
- налаштування Cinder контролеру;
- налаштування оркестрації Heat (якщо ввімкнено у файлі scripts.ubuntu_cluster).

Обчислювальний вузол:

- налаштування Nova compute;
- налаштування Neutron обчислення;
- налаштування системи самообслуговування;
- налаштування Cinder volumes.

Підсистема управління:

- налаштування загальної мережі;
- налаштувати приватну мережу.

4.3.4 Служба зображень – glance

Служба Image OpenStack дозволяє користувачам знаходити, зареєструвати та отримувати зображення VM. Glance пропонує RESTful API, який дозволяє запитувати метадані VM та вилучати зображення. Він підтримує зберігання дискових або серверних зображень на різних типах репозиторіїв, включаючи OpenStack Object Storage.

Служба Image OpenStack містить наступні компоненти:

- glance-api: приймає виклики API Image для пошуку, вилучення та зберігання зображень;
- glance-реєстр: зберігає, обробляє та отримує метадані про зображення. Метадані включають такі елементи, як розмір і тип;
- база даних – зберігає метадані зображення, і ви можете вибрати свою базу даних залежно від ваших уподобань. Більшість розгортань використовують MySQL або SQLite;
- сховище зберігання – підтримуються різні типи репозиторіїв, включаючи звичайні файлові системи, об'єкт зберігання, надійні автономні пристрої розподіленого об'єкта (RADOS), HTTP і Amazon S3. Зауважте, що деякі сховища підтримуватимуть лише читання.

4.3.5 Сервіс обчислень – Nova

OpenStack Обчислює хости і керує системами хмарних обчислень. Вона взаємодіє з ідентифікацією OpenStack для аутентифікації, службою Image OpenStack для зображень на диску та серверах, а також інформаційної панелі OpenStack для користувача та адміністративного інтерфейсу. Доступ до зображень обмежується

проектами, а також користувачами; квоти обмежені для кожного проекту (наприклад, кількість екземплярів). OpenStack Compute може масштабувати горизонтально на стандартному апаратурі та завантажувати зображення для запуску екземплярів.

OpenStack Compute складається з наступних областей та їх компонентів:

а) api:

1) nova-api сервіс – приймає та відповідає на виклики API, що обчислюються кінцевими користувачами. Служба підтримує API OpenStack Compute, API Amazon Elastic Compute 2 (EC2) та спеціальний API адміністратора для привілейованих користувачів для виконання адміністративних дій. Він забезпечує виконання певних правил і ініціює більшість операцій з оркестуванням, наприклад, запуск екземпляра

2) nova-api метадата сервіс – приймає запити метаданих від екземплярів. Служба nova-api-metadata зазвичай використовується, коли ви запускаєте в режимі multi-host за допомогою нових установок.

б) Ядро обчислень

1) nova-обчислювальний сервіс – сервіс, який створює та завершує візуальні екземпляри за допомогою API-інтерфейсів гіпервізора.

2) nova-планувальник – Вводить запит на віртуальний екземпляр із черги і визначає, на якому комп'ютері обчислюється сервер.

3) nova-модуль провідника – Посередляє взаємодія між новообчислювальною службою та базою даних. Це усуває прямі доступ до хмарної бази, створеної службою nova compute. Nova-провідний модуль масштабує горизонтально Проте, не розгортайте його в вузлах, де запускається служба nova обчислювальник.

в) Мережа для VMs

1) nova-мережевий працівник – подібно новій службі обчислень, приймає сеансові завдання з черги та маніпулює мережею. Виконання завдань, таких як встановлення перехресних інтерфейсів або зміна правил IPtables.

г) Інші компоненти

1) Черга – Центральний вузол для передачі повідомлень між демонами. Зазвичай реалізується з RabbitMQ, але може бути реалізований з чергою повідомлень AMQP, такими як Apache Qpid або Zero MQ.

2) База даних SQL – зберігає більшість стадій часу і станів виконання для хмарної інфраструктури

Теоретично OpenStack Compute може підтримувати будь-яку базу даних, яку підтримує SQL-Alchemy. Спільними базами даних є SQLite3 для роботи з тестування та розробки, MySQL та PostgreSQL. SQL-Alchemy – це інструментарій Python SQL і об'єктивний реляційний маппер (ORM), який надає розробникам додатків повну потужність та гнучкість SQL. Він надає повний комплект добре відомих моделей стійкості на рівні підприємства, розроблений для ефективного та високопродуктивного доступу до бази даних, адаптований до простої мови та домену Pythonic.

4.3.6 Мережевий сервіс Neutron

OpenStack Networking дозволяє створювати та прикріплювати інтерфейсні пристрої, що управляються іншими службами OpenStack для мереж, віртуальної мережевої інфраструктури (VNI) та способів доступу до інфраструктури фізичної мережі (PNI). Плагіни можуть бути реалізовані для розміщення різного мережевого обладнання та програмного забезпечення, що забезпечує гнучкість архітектури OpenStack та її розгортання. Для цього Neutron забезпечує вилучення об'єктів, щоб імітувати його фізичний аналог:

- мережі;
- підмережі;
- маршрутизатори;
- групи безпеки.

Neutron включає в себе наступні компоненти:

а) Neutron сервер – приймає та передає запити API на відповідний мережевий плагін для дій;

б) мережеві плагіни та агенти:

1) підключіть та від'єднуйте роз'єми, створюйте мережі або підмережі та надайте IP адресацію. Ці плагіни та агенти відрізняються залежно від постачальника та технологій, що використовуються в конкретній хмарі;

2) мережа поставляється з плагінами та агентами для віртуальних і фізичних комутаторів Cisco, продуктів NEC OpenFlow, Open vSwitch (OvS), мережевого взаємодії Linux та продукту VMware NSX;

3) загальними агентами є L3 (рівень 3), протокол конфігурації динамічного хоста (DHCP) та агент плагіна.

в) Черга повідомлень – використовується більшістю установок OpenStack Networking для маршрутизації інформації між нейтрон-сервером та різними агентами. Він також виступає як база даних для зберігання мережевого стану для окремих плагінів.

Cinder, служба зберігання блочних пристроїв OpenStack додає постійне зберігання VM. Блок-накопичувач забезпечує інфраструктуру для керування обсягами та взаємодіє з OpenStack Compute, щоб забезпечити обсяги для екземплярів. Послуга також дозволяє керувати об'ємними знімками та типи об'ємів.

Служба зберігання блочних пристроїв складається з наступних компонентів:

а) cinder-api – Приймає запити API і направляє їх до обсягу звуку для дії.

б) cinder-volume

1) Взаємодіятиме безпосередньо з службою блокування зберігання та виконує такі процеси, як планувальник cinder

2) Відповідає на читання та запис запитів, надісланих службі Block Storage для підтримки стану

3) Взаємодіє з різними постачальниками послуг зберігання за допомогою архітектури драйверів.

4) Взаємодіє з цими процесами через чергу повідомлень

в) планувальник cinder – Вибір оптимального вузла постачальника зберігання, на якому можна створити гучність.

г) cinder-backup

1) Забезпечує резервне копіювання обсягів будь-якого типу на постачальника резервного копіювання

2) Взаємодія з різними постачальниками послуг зберігання за допомогою архітектури драйверів.

д) Черга повідомлень – маршрутне повідомлення між процесами Блокування зберігання.

4.3.7 Сервіс оркестрації Heat

Служба Orchestration забезпечує оркестровку, основану на шаблонах, для опису обласного додатка, запустивши виклики API OpenStack для створення запущених додатків для хмар. Програмне забезпечення інтегрує інші основні компоненти OpenStack в єдину систему шаблонів. Шаблони дозволяють створювати більшість типів ресурсів OpenStack, таких як екземпляри, плаваючі IP-адреси, томи, групи безпеки та користувачі.

Він також забезпечує розширені функції, такі як, наприклад, висока доступність (HA), автоматичне масштабування примірників та вкладені стеки. Це дозволяє основним проектам OpenStack отримувати більшу базу користувачів.

Сервіс включений для інтеграції з сервісом оркестування безпосередньо або через спеціальні плагіни.

Служба оркестрації складається з наступних компонентів:

а) Heat – командний рядок клієнта

1) heat – CLI, який зв'язується з heat-api для запуску API CloudFormation AWS

2) heat – CLI, який зв'язується з heat-api для запуску API CloudFormation AWS

б) heat – api компонент – OpenStack-рідний API REST, який обробляє запити API, відправляючи їх в тепловий двигун за допомогою функції віддаленого виклику процедури (RPC).

в) heat-api-cfn компонент

1) API AWS Query, сумісний з AWS CloudFormation
2) Процесує API запити, відправивши їх на тепловий двигун RPC

г) Heat-двигун – організовує запуск шаблонів та передає події назад споживачу API.

4.3.8 Розгортання примірника VM

Для розгортання інстанції обчислювальний вузол має запущений гіпервізор KVM / QEMU.

Цей гіпервізор розкручує віртуальну машину. Існують і інші вимоги. Служба Glance забезпечує безпеку від сервісу Nova, мережа забезпечується сервісом Neutron та сховищем з сервісу Cinder та, нарешті, сам примірник служби Nova. Для роботи служби Neutron потрібна приватна мережа, яка буде зарезервована для цього конкретного проекту, щоб запустити екземпляр.

Розгортання екземпляра зображено на рисунку 4.14

Щоб розгорнути екземпляр, слід виконати наступні кроки:

а) Налаштування мережі;
б) Призначити плаваючі IP-адреси;
в) Визначте групу захисту в хмарі;
г) Створіть пару ключів SSH;
д) Створіть уявлення про зображення;
е) Виберіть аромат;
ж) Екземпляр можна завантажити.

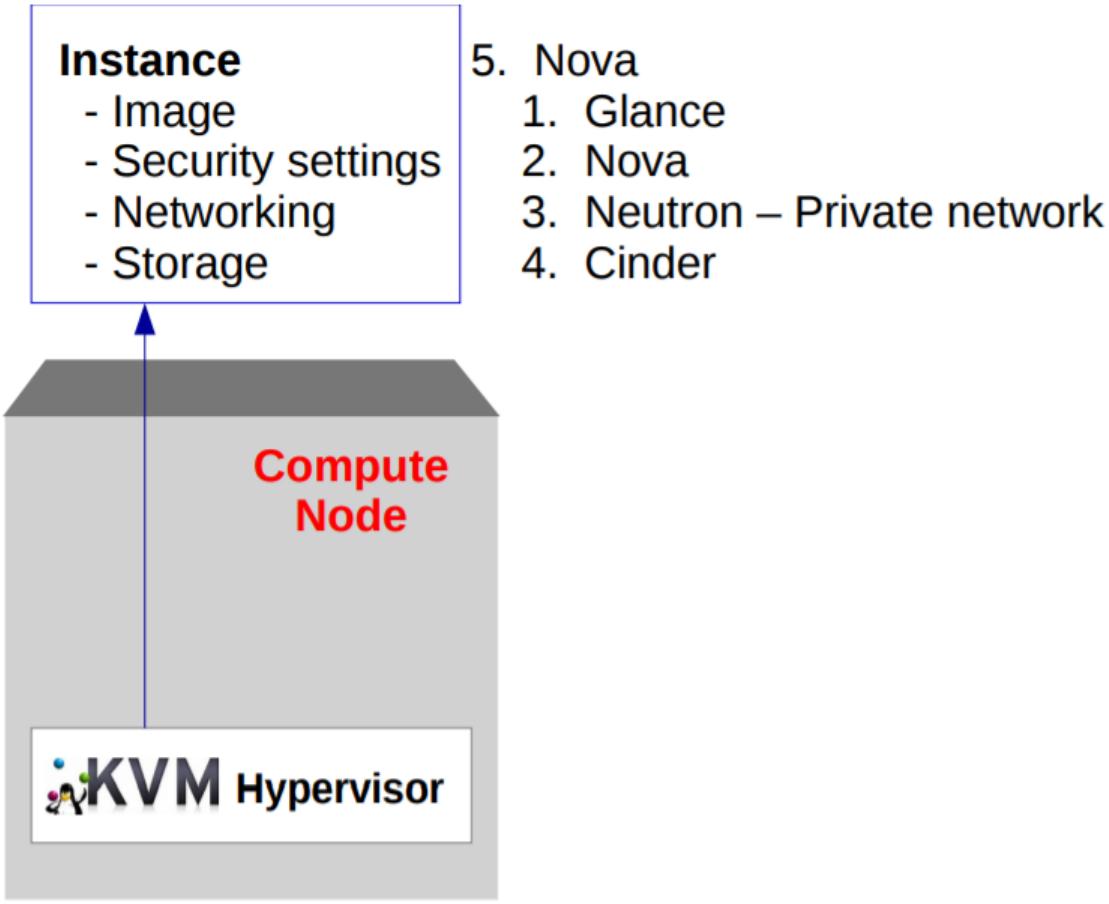


Рисунок 4.14 – Розгортання екземпляра

Мережа є важливою частиною хмарного середовища. Тому середовище SDN створюється перед початком створення екземплярів. Це середовище SDN дозволяє екземплярам підключатися до приватної внутрішньої мережі, а також призначати плаваючу IP-адресу, щоб їх можна було добрати ззовні.

Рисунок 4.15 показує два випадки: Instance 1 та Instance 2 з приватними IP-адресами 192.168.10.21 та 192.168.10.22 з повагою. Ця мережа веде себе як мережа поза мережею NAT. IP-адреси 192.168.10.21 та 192.168.10.22 недоступні безпосередньо з зовнішньої мережі. Щоб дозволити доступ до віртуальних екземплярів, функція маршрутизації SDN має плаваючу IP-адресу, у цьому прикладі з мережі 203.0.113.0/24, яка може отримати доступ до зовнішньої мережі.

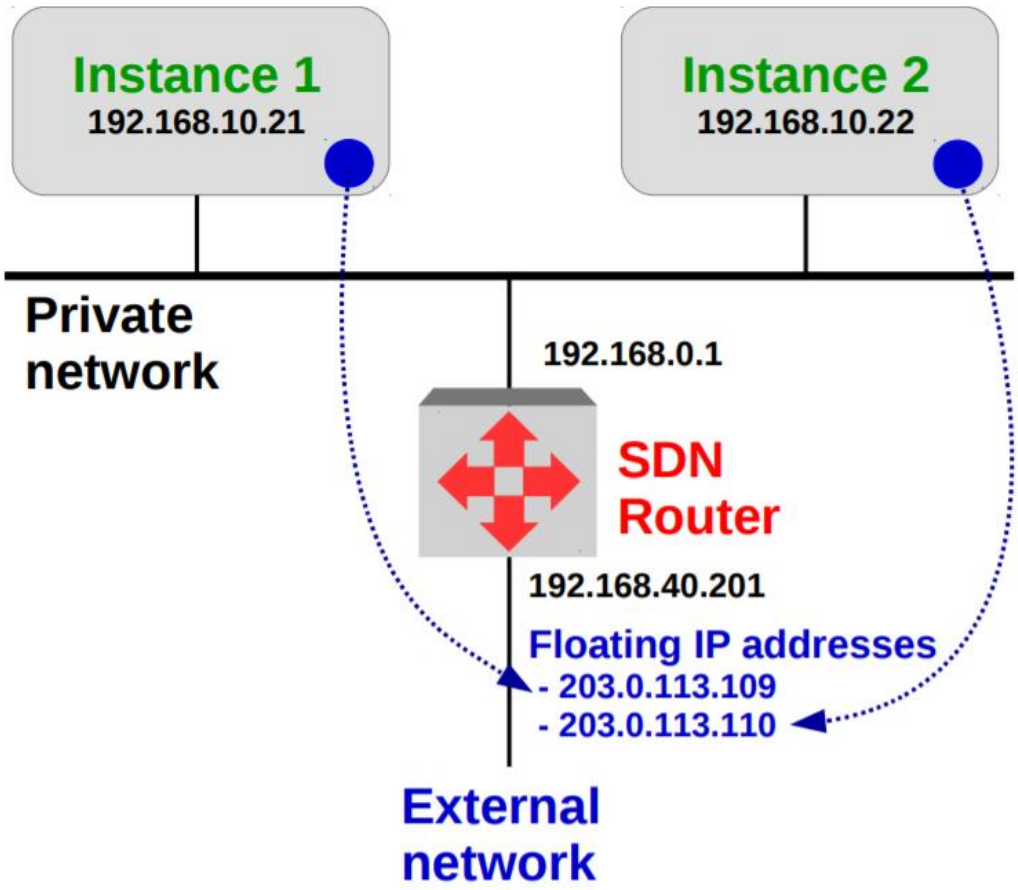


Рисунок 4.15 – Плаваючі IP-адреси

Плаваюча IP-адреса – це IP-адреса, зарезервована для екземпляра, і виставлена на зовнішній стороні маршрутизатора SDN. Зовнішній трафік доступ до екземпляру через його плаваючу IP-адресу.

Включити демо-openrc змінні, щоб застосувати демонстраційні вказівки користувачів. Це дає доступ до команд перегляду користувача. Діяльність користувачів в рамках Проектів та адміністратора не має видимості віртуальних екземплярів, створених користувачами, це тому, що OpenStack є інструментом оркестування, і постачальник інфраструктури не має потреби знати, що їхні клієнти організовують.

Хмарне зображення, як правило, підтримує автентифікацію за допомогою звичайного ключа замість звичайної перевірки паролем. Тому необхідно додати відкритий ключ до служби Compute, яку можна вибрати при запуску екземпляра.

Генеруйте пару ключових слів, файл .pem буде доступний для користувачів, які потребують доступу.

```
osbash@controller:~$ openstack keypair create mykey > mykey.pem

osbash@controller:~$ openstack keypair show mykey
+-------------+-------------------------------------------------+
| Field       | Value                                           |
+-------------+-------------------------------------------------+
| created_at  | 2017-05-04T15:02:50.000000                      |
| deleted     | False                                           |
| deleted_at  | None                                            |
| fingerprint | 5d:0c:ad:d4:0f:f8:a5:30:18:71:48:ab:db:39:3e:e6 |
| id          | 5                                               |
| name        | mykey                                           |
| updated_at  | None                                            |
| user_id     | 3fc36085c8fd4472bf507d03fa50dcd2                |
+-------------+-------------------------------------------------+
```

Створіть 1 Гб логічний том, який можна прикріпити до примірника VM пізніше. Cinder використовує LVM у GNU / Linux. LVM управляє дисковими накопичувачами та аналогічними накопичувачами. Об'єм – це дисковий диск або розділ диска. Вона була написана в 1998 році Хайнцем Маулшагеном, який заснований на дизайні LVM у HP-UX. LVM можна розглядати як тонкий рівень програмного забезпечення на вершині жорстких дисків та розділів, що створює абстракцію безперервності та простоти використання для керування заміною жорсткого диска, перерозподілом та резервуванням. Cinder створює гучність на обчислювальному вузлі.

```
osbash@controller:~$ openstack volume create --size 1 1GB-vol
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| attachments         | []                                   |
| availability_zone   | nova                                 |
| bootable            | false                                |
| consistencygroup_id | None                                 |
| created_at          | 2017-05-04T15:09:25.737145           |
| description         | None                                 |
| encrypted           | False                                |
| id                  | 10c8ca56-e831-4ada-9a6c-a2ee7b3a03ed |
| multiattach         | False                                |
| name                | 1GB-vol                              |
| properties          |                                      |
| replication_status  | None                                 |
| size                | 1                                    |
| snapshot_id         | None                                 |
| source_volid        | None                                 |
| status              | creating                             |
| type                | None                                 |
| updated_at          | None                                 |
| user_id             | 3fc36085c8fd4472bf507d03fa50dcd2     |
+---------------------+--------------------------------------+

osbash@controller:~$ openstack volume list
+--------------------------------------+--------------+-----------+------+-------------+
| ID                                   | Display Name | Status    | Size | Attached to |
+--------------------------------------+--------------+-----------+------+-------------+
| 10c8ca56-e831-4ada-9a6c-a2ee7b3a03ed | 1GB-vol      | available |    1 |             |
+--------------------------------------+--------------+-----------+------+-------------+
```

4.4 Створення мережі

Тепер, коли освоєно створення екземплярів, розглянемо створення мереж. Мережеві графи на рисунку 4.16 та додатку Е, демонструє просту мережу з 4 хостами, дві в мережі постачальника за замовчуванням і дві в новій приватній мережі, яка підключена до мережі постачальника через маршрутизатор.

Ось пояснення процесу створення додаткової приватної мережі, хостів, маршрутизатора та підключення мереж.

Наступні кроки:

а) Увімкніть змінні admin-openrc

б) Створіть flavour

в) Увімкніть змінні demo-openrc

г) Додайте порт 22 (SSH) та ICMP до групи безпеки за умовчанням

д) Створіть приватну мережу

е) Витягнете Провайдер та приватні UUID мережі

ж) Створення хостів у мережі постачальника послуг

з) Створення хостів у приватній мережі

и) Створіть маршрутизатор

к) Додайте підмереж до маршрутизатора

л) Додайте маршрут за замовчуванням до маршрутизатора через 203.0.113.1

м) Додайте маршрут на хост до приватної мережі.

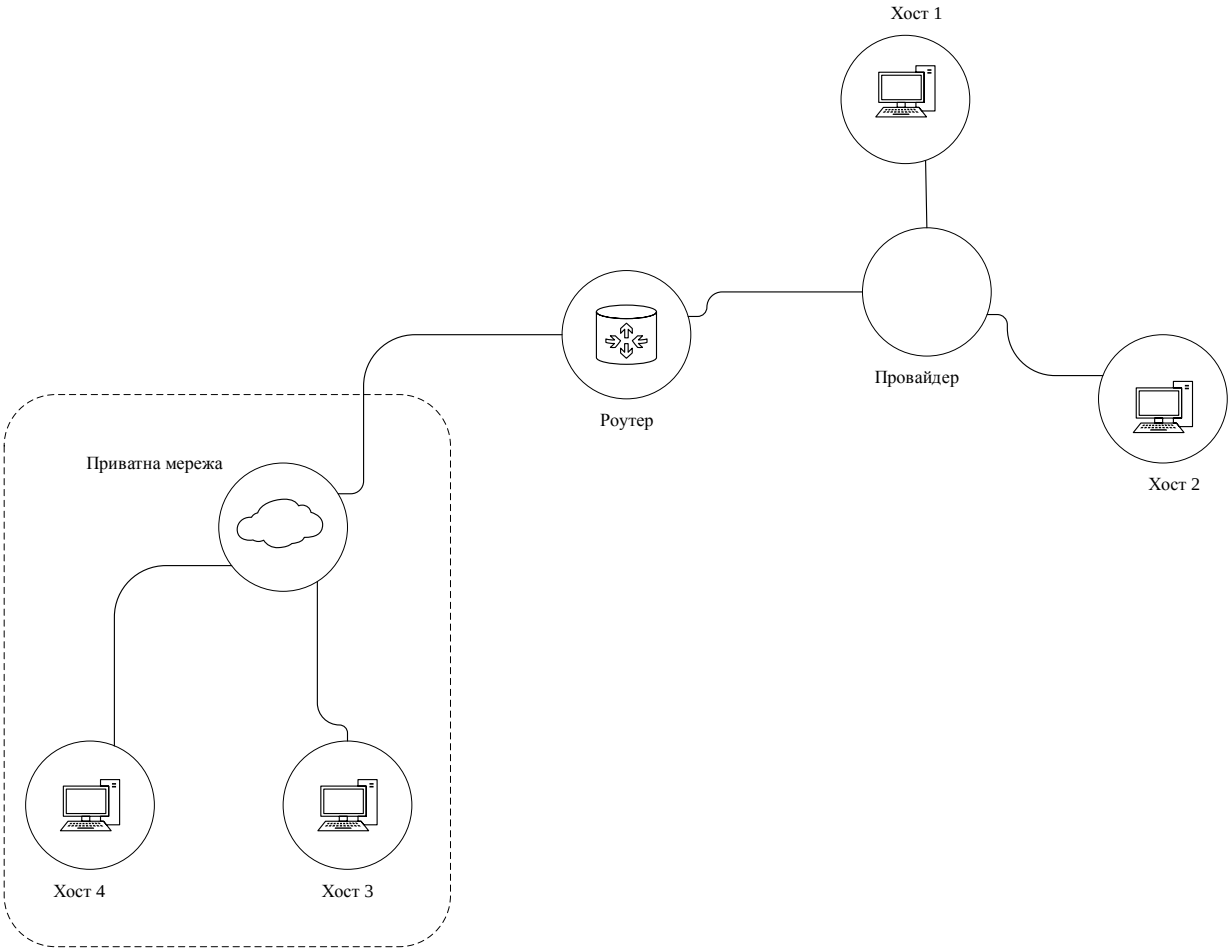


Рисунок 4.16 – Мережеві графи

Початкова конфігурація

Перш ніж потрапити до елементу мережі, дозвольте змінювати адміністратор-openrc і створити аромат

Тепер увімкніть змінні demo-openrc і додайте до порту 22 (SSH) і ICMP групу безпеки за умовчанням

Створити приватну мережу

Створіть приватну мережу та призначте йому інформацію про мережу.

```
osbash@controller:~$ openstack network create PRIV-NET

osbash@controller:~$ openstack subnet create --network PRIV-NET \
  --subnet-range 192.168.95.0/24 --gateway 192.168.95.1 --dhcp \
  --allocation-pool start=192.168.95.10,end=192.168.95.20 \
  --dns-nameserver 8.8.8.8 PRIV-SUBNET
```

Вилучіть постачальника та приватні UUID мережі

```
osbash@controller:~$ openstack network list | grep provider | awk
'{print $2}'
1ad8799b-8d9a-4ddd-801f-942da3549ee4

osbash@controller:~$ openstack network list | grep PRIV-NET | awk
'{print $2}')
34477f0c-cedc-4a0c-a7ab-66439a5c709b
```

Створіть маршрутизатор.

```
osbash@controller:~$ openstack router create router1
```

Встановіть зовнішній шлюз маршрутизатора до мережі постачальника.

```
osbash@controller:~$ openstack router set --external-gateway=provider
router1
```

Додайте підмереж приватної мережі до маршрутизатора.

```
osbash@controller:~$ openstack router add subnet router1 PRIV-SUBNET
```

Додайте маршрут за замовчуванням до маршрутизатора через 203.0.113.1

```
osbash@controller:~$ openstack router set router1 --route
destination=0.0.0.0/0,gateway=203.0.113.1
```

Додайте маршрут до гіпервізора до приватної мережі

Додайте статичний маршрут з гіпервізора до приватної мережі. Спочатку визначте IP-адресу, призначену інтерфейсу провайдера маршрутизатора OpenStack.

```
osbash@controller:~$ openstack router show router1 | grep
external_gateway_info | awk -F '"' '{print $16}'
203.0.113.104
```

Потім додайте статичний маршрут до приватної мережі.

```
ada:~$ sudo ip route add 192.168.95.0/24 metric 1 nexthop via
203.0.113.104
```

Перевірте конфігурацію. З гіпервізора хост пінг чотирьох хостів.

```
ada:~$ ping -c1 203.0.113.102
PING 203.0.113.102 (203.0.113.102) 56(84) bytes of data.
64 bytes from 203.0.113.102: icmp_seq=1 ttl=64 time=0.960 ms

--- 203.0.113.102 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.960/0.960/0.960/0.000 ms

ada:~$ ping -c1 203.0.113.103
PING 203.0.113.103 (203.0.113.103) 56(84) bytes of data.
64 bytes from 203.0.113.103: icmp_seq=1 ttl=64 time=2.10 ms

--- 203.0.113.103 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2.101/2.101/2.101/0.000 ms

ada:~$ ping -c1 192.168.95.12
PING 192.168.95.12 (192.168.95.12) 56(84) bytes of data.
64 bytes from 192.168.95.12: icmp_seq=1 ttl=63 time=0.866 ms

--- 192.168.95.12 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.866/0.866/0.866/0.000 ms

ada:~$ ping -c1 195.168.95.17
PING 195.168.95.17 (195.168.95.17) 56(84) bytes of data.
64 bytes from 195.168.95.17: icmp_seq=1 ttl=235 time=238 ms

--- 195.168.95.17 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 238.718/238.718/238.718/0.000 ms
```

4.5 UML Діаграма адміністрування OpenStack через сервіс Horison

В даній дисертації спроектована та побудована UML діаграмма адміністрування OpenStack лабораторного стенду хмарної IT-інфраструктури. UML діаграмма адміністрування OpenStack зображена на рисунку 4.17. та на додатку Ж.

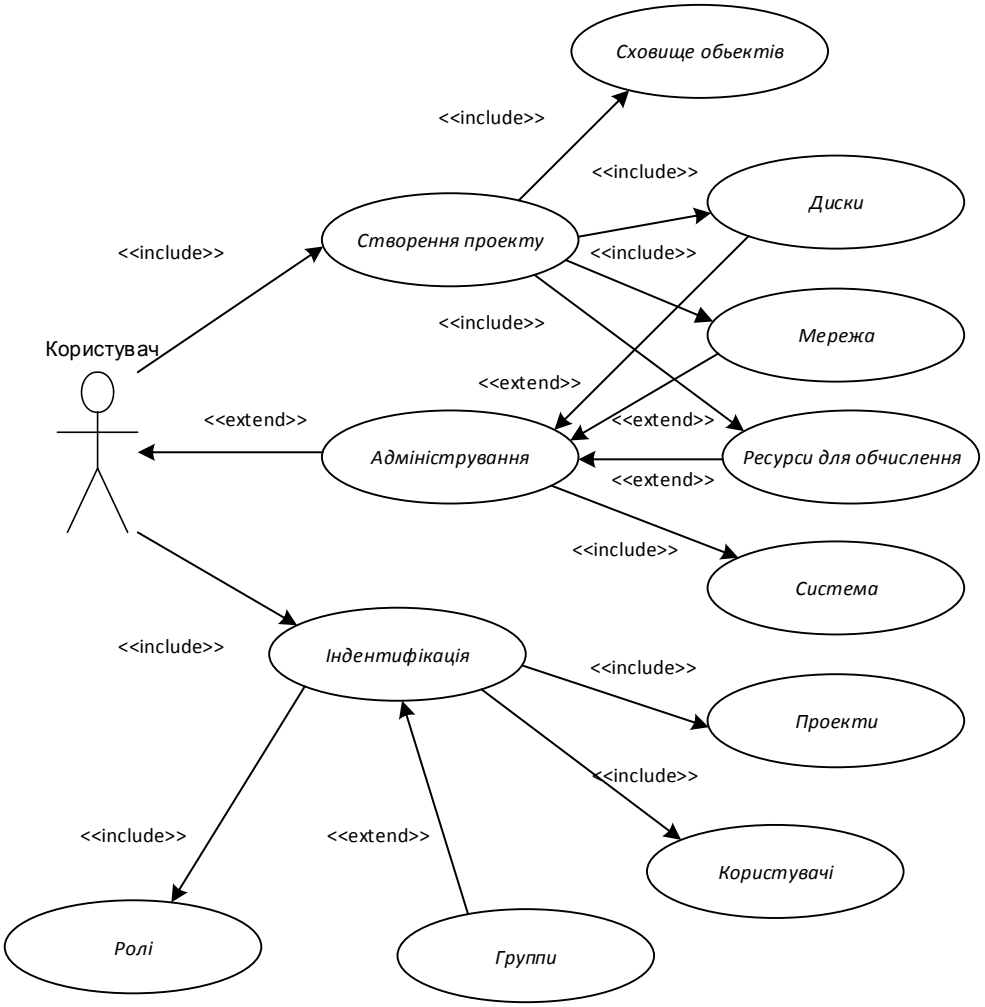


Рисунок 4.17 – UML діаграмма адміністрування OpenStack

Діаграма прецедентів в UML – це діаграма, яка відображає відношення між акторами, моделі прецедентів, яка є основною частиною, а також самих прецедентів.

Основна задача приведеної діаграми – опис функціональності, який дозволяє керувати IT-інфраструктурою через сервіс Horizon.

Не існує стандартного способу опису вмісту прецеденту; в різних випадках застосовуються різні формати. На рисунку 4.17 показаний загальний стиль використання. Починається робота з вибору одного з сценаріїв в якості головного успішного сценарію. Спочатку описується тіло прецеденту, в якому головний успішний сценарій представлений послідовністю кроків. Потім берете інший сценарій і вставляєте його у вигляді розширення, описуючи його в термінах змін головного успішного сценарію. Розширення можуть бути успішними – користувач достиг свої мети.

Кожен крок у прецеденті – це елемент взаємодії актора з системою. Кожен крок повинен бути простим твердженням і повинен чітко вказувати, хто виконує цей крок. Крок повинен показувати намір актора, а не механіку його дій. Отже, в прецеденті інтерфейс актора не описується. Дійсно, складання прецеденту зазвичай передує розробці інтерфейсу користувача.

Користувач

Користувач – являє собою якусь роль, яку користувач грає по відношенню до системи. Акторами можуть бути користувач, торговий представник користувача, менеджер з продажу та товарознавець.

В данній діаграмі зображений один користувач який і керує інфраструктуру через інструментарну панель.

Прецедент

Прецедент – це можливість модулюємо системи, за допомогою якої користувач може отримати потрібний йому, вимірюваний і точний результат. Прецедент відповідає окремому з варіантів його використання і описує типовий спосіб взаємодії користувача з системою.

Прецидент створення проекту, надає можливість користувачу утворити проект, виділити під нього ресурси, створити мережу, сховище обьектів та диски до яких буде звертатися користувач

Прецедент адмініструання довзоляє вносити зміни в прецеденти мереж і дисків, а також дозволяе змінювати кількість ресурсів для обчислення.

Прецендент ідентифікації регулює доступ до проектів створеним іншим користувачем. Індентефікація дозволяж не тільки створити нового користувача, а й виділити йому роль та присвоїти группу

5 РОЗРОБКА СТАРТАП-ПРОЕКТУ

5.1 Вступ

Напрацювання, викладені у даній магістерській дисертації, можуть бути практично застосовані і мають комерційну цінність для систем електронного документообігу з підтримкою багатомовних документів. Дана система може бути корисною в першу чергу в наукових, освітніх, юридичних та комерційних організаціях, що орієнтовані на міжнародну співпрацю. Такі системи на даний момент відсутні на ринку.

5.2 Опис ідеї стартап-проекту

Ідея стартап-проекту описана у таблиці 5.1.

Таблиця 5.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Лабораторний стенд для управління та проектування хмарної IT інфраструктури	1. Створення хмарної IT інфрструктури	Створенная моделі для надання широкодоступного та зручного доступа до мережі загальному пулу налаштованих ресурсів за вимогою

Продовження таблиці 5.1

Зміст ідеї	Напрямки застосування	Вигоди для користувача
	2. Підготовка спеціалістів з використання хмарних IT інфраструктур	1. Вивчення використання хмарних IT- інфраструктур. 2. Впевненість у професійності підготовлених спеціалістів.
	3. Інтегрування IT-інфраструктуру в підприємство.	Використання програми для керування та автоматизації пулів комп'ютерних ресурсів і може працювати з поширеними технологіями віртуалізації, звичайними комп'ютерами та конфігураціями для високопродуктивних обчислень (HPC).

Аналіз сильних, слабких та нейтральних сторін проекту наведено у таблиці 5.2. Резюмуючи коротко, перевагами системи є інтегрованість підсистем документообігу та перекладу, кросплатформенність (не вимагає серверів з пропрієтарним програмним забезпеченням) та робота з багатомовними шаблонами документів.

Таблиця 5.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

Продовження таблиці 5.2

		(потенційні) товари/концепції конкурентів						
		Мій проект	Microsoft	Google	Amazon			
1	Вартість експлуатації	500 грн за місяць	Від 5,00 доларів США/місяць за користувача	50,00 грн/місяць	Від 50 доларів США	-	-	+
2	Кросплатформенність	+	–	–	–	-	-	+
3	Інтегрованість перекладу та документообігу	+	–	–	–	-	-	+
4	Робота в глобальній мережі	+	+	+	+	-	+	-
5	Робота в локальній мережі	+	–	+	+	-	+	-
6	Стійкість до відмов	Висока	Висока	Висока	Висока	-	+	-
7	Наявність системи	+	+	+	+	-	+	-

Продовження таблиці 5.2

		(потенційні) товари/концепції конкурентів						
	резервування							
8	Ступінь розширюваності	Висока	Середня	Низька	Низька	-	-	+
9	Робота з великою кількістю сервісів	+	+	+	+	-	+	-
10	Складна структура інфраструктура	+	-	+	+	-	-	+
11	Зручність у користуванні	Середня	Висока	Висока	Висока	+	-	-
12	Зручність у управління IT-інфраструктурою	Висока	Низька	Низька	Низька	-	-	+
13	Швидкість обчислень	Висока	Середня	Висока	Висока	-	+	-
14	Налаштування прав доступу	+	+	+	+	-	+	-

5.3 Технологічний аудит проекту

Аналіз складових технологічного процесу наведено у таблиці 3.3.

Таблиця 5.3 – Технологічна здійсненність ідеї проекту

№ п\п	Ідея проекту	Технології реалізації	Наявність технологій	Доступність технологій
1		OpenStack	Наявні	Доступні
2		CloudStack	Наявні	Доступні
Обрана технологія реалізації ідеї проекту: OpenStack				
4		Virtual Box	Наявні	Доступні
5		KVM	Наявні	Доступні
Обрана система віртулізації: KVM				
7		Nova	Наявні	Доступні
8		Neutron	Наявні	Доступні
9		Swift	Наявні	Доступні
10		Glance	Наявні	Доступні
12		Cinder	Наявні	Доступні
13		Keystone	Наявні	Доступні
14		Horizon	Наявні	Доступні
15		Sahara	Наявні	Доступні

В цілому, були обрані крос-платформенні технології. Система не вимагає специфічного типу реляційної бази і може використовувати будь-яку СУБД, для якої існує Ado.Net адаптер.

5.4 Аналіз ринкових можливостей запуску проекту

В цілому, ринок готовий до сприйняття продукту, і продукт має свою нішу на ринку. Це підтверджується попереднім оглядом ринку, наведеному у таблиці 5.4.

Умовною одиницею продажу є ліцензія на 1 місяць на 1 користувача.

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, одиниць	4
2	Загальний обсяг продаж, грн/ум.од	500 грн/ум.од
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Недискримінацій ні якісні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні на початковому етапі
6	Середня норма рентабельності в галузі (або по ринку), %	80%

Потенційні характеристики клієнтів стартап-проекту описані у таблиці 5.5. Це можуть бути як приватні, так і державні установи.

Таблиця 5.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару

Продовження таблиці 5.5

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Створення хмарної IT інфрструктури	Компанії та установи з активною міжнародною діяльністю: підприємства, заклади освіти та науки	Створенная моделі, орієнтовану на потреби цільвих груп клієнтів, для надання широкодоступного та зручного доступа до мережі загальному пулу налаштованих ресурсів за вимогою	Визначення умов яким має відповідати хмарна IT-інфраструктура
2	Підготовка спеціалістів з використання хмарних IT інфраструктур		Знаходження персоналу для вивчення хмарних IT-інфрсатрутур	1. Вивчення використання хмарних IT-інфраструктур. 2. Впевненість у професійності підготовлених спеціалістів.

Продовження таблиці 5.5

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
3	. Інтегрування IT-інфраструктуру в підприємство		1. Допомога у створенні документів 2. Генерація документів з автоматичним перекладом та транслітерацією змінних реквізитів	Використання програми для керування та автоматизації пулів комп'ютерних ресурсів і може працювати з поширеними технологіями віртуалізації, звичайними комп'ютерами та конфігураціями для високопродуктивних обчислень (HPC).

Проте, український ринок в цілому і ринок інформаційних послуг в цілому є нестабільним і необхідним є врахування низки загрозливих факторів. Фактори ризику і загрози на ринку описані у таблиці 5.6.

Таблиця 5.6 – Фактори загроз

Продовження таблиці 5.6

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Непередбачувані зміни законодавства	Зміна вимог до організації з боку органів державної влади (наприклад, рівень оподаткування)	Організація юридичної особи в країні зі сприятливим інвестиційним кліматом
2	Відсутнітьс інтернет мережі	Без відсутності мережі, підприємство не зможе звертатися до хмарних схованок.	Підписання надійного контракту з інтеренет провайдером, для стабільної роботи мережі.
3	Недостатня швидкість інтернет з’єднання	Через недостатньо швидке інтернет з’єднання, пересилання і отримання данних може відбуватися з затримкою.	Підписання надійного контракту з інтеренет провайдером, для надання високої швидкості інтернет з’єднання.
5	Недовіра користувачів	Користувачі не довіряють новому	Впровадження на безоплатній

Продовження таблиці 5.6

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
		гравцю на ринку	основі в одній з організацій

Проте, наразі ринок сформував також низку можливостей, реалізація яких дозволить отримати прибуток і завоювати ринок.

Таблиця 5.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Отримання необхідних інвестицій	Сформований початковий капітал, необхідний для реалізації мінімально життєздатного продукту	Розробка мінімально життєздатного продукту
2	Розвиток ринку хмарних IT-інфраструктур	Установи що активно впроваджують інформаційну інфраструктуру	Надання послуг державним організаціям, участь у тендерах
3	Зростання промисловість	Незважаючи на складнощі, ВВП країни зростає, на ринку з’являються нові підприємства	Робота з новими підприємствами на ринку, реорганізація документообігу у існуючих

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
5	Успішна маркетингова та інформаційна політика	Інформаційна кампанія з прикладами успішного впровадження системи	Збільшення кількості споживачів при фактично сталій вартості розробки та підтримки системи

Далі було розглянуто конкуренцію на ринку. Результати ступеневого аналізу подані у таблиці 5.8.

Таблиця 5.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Олігополія	Незначна кількість конкурентів Велика ринкова сила Схожість використовуваних технологій	Інформування ринку щодо появи нової веб-служби
Галузевий	Загроза появи нових конкурентів Ринкова влада споживачів Висока потреба у	Інформування ринку щодо якості використовуваної новаторської технології Пропозиція гнучких цін

Таблиця 5.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
	товарі	
Внутрішньогалузева	Діяльність в одній галузі економіки Надання сервісів одного типу	Зменшення вартості сервісу Примноження каналів розподілу
Товарно-видова	Надання різних сервісів одного виду	Маркетингова політика
Цінова	Використання цін для покращення економічних умов збуту	Зменшення вартості сервісу Використання нових каналів розподілу
Марочна	Пропозиція схожого сервісу Спільна цільова аудиторія	Інформування ринку щодо якості використовуваної новаторської технології Примноження каналів розподілу

Більш детальний аналіз конкуренції на ринку було виконано за моделлю 5 сил М. Портера, результати наведені у таблиці 5.9.

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

Продовження таблиці 5.9

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складовий аналіз	СЕД: «Діло-Підприємство», «Optima Workflow», «IT Enterprise» Перекладачі: Promt, Google Translator, Microsoft Translator	Капіталовкладення, існуючі напрацювання в кожній з поєднуваних компонент	Відсутні	- Змінні витрати: Виробничі непрямі дегресивні - Системи інформації: реклама та директ-маркетинг, - Рівень чутливості до цін: орієнтовані на цінність продукту - Продуктова диференціація: якість, спосіб отримання сервісу, швидкість обслуговування Методи контролю якості:	Копіювання функціонал, Монополізація дистриб'юторів, Демпінг

Продовження таблиці 5.9

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
				ручне та автоматичне тестування, профілювання, прототипування, рев'ю коду, аудит	
Висновки	CR4 = 85% Індекс Херфіндаля-Хіршмана (HHI) = 3450 Значення показників вказує на високу концентрацію (монополізацію) даного ринку	Можливості входу на ринок забезпечить мінімізація цін, швидкість та простота надавання послуги споживачам, швидке реагування на вимоги користувачів. В результаті аналізу проектів на народно-громадських інтернет-платформах	Відсутні	Клієнти диктують умови гнучкості цінової політики, високої і довгострокової якості послуг та підтримку всіх видів документів, що вони використовують	Пропонування вигідних умов безпосереднім користувачам, виділення ліній підтримки

Продовження таблиці 5.9

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
		потенційних конкурентів знайдено не було			

Незважаючи на високу конкуренцію на ринку і наявних на ньому гравців, пропонована система має низку переваг, що описані у таблиці 5.10.

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Унікальність сервісу	Створення стенду для проектування та управління хмарної IT-інфраструктури
2	Модель бізнес для бізнесу	Бізнес модель ґрунтується на співпраці із іншими підприємствами та інститутами
3	Цінова політика	Отримання прибутку здійснюється за рахунок продажу ліцензій, що є звичайною політикою у галузі
4	Додаткові послуги	Можливе розширення функціональності під вимоги конкретного користувача, в тому числі безкоштовно – якщо таке розширення збільшує цінність системи для потенційних клієнтів

За визначеними факторами конкурентоспроможності, наведеними у таблиці 5.10, було виконано порівняльний аналіз сильних та слабких сторін стартап-проекту зі зв’язкою Microsoft SharePoint + Google Translate. Результати аналізу наведені у таблиці 5.11.

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні зі зв’язкою Cloudstack						
			–3	–2	–1	0	+1	+2	+3
1	Унікальність сервісу	17						+	
2	Модель бізнес для бізнесу	14					+		
3	Цінова політика	12							+
4	Додаткові послуги	18							+

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу на основі раніше отриманих даних.

SWOT-аналіз полягає у побудові матриці, що включає:

- сильні сторони проекту (Strength);
- слабкі сторони проекту (Weak);
- загрози (Troubles);
- можливості (Opportunities).

На основі SWOT-аналізу були розроблені альтернативи ринкової поведінки для виведення стартап-проекту на ринок та розрахований орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок.

Результати проведеного SWOT-аналізу наведені у таблиці 5.12.

Таблиця 5.12 – SWOT- аналіз стартап-проекту

Сильні сторони: Гнучка цінова політика Додаткові сервіси	**Слабкі сторони:** Нестача стартових капіталовкладень Сильна монополізація ринку Інертний ринок
Можливості: Інвестиції Реалізація лабораторного стенду. Висока зацікавленість інститутів які вивчають мережі	**Загрози:** Застарілі системи на підприємствах де не можна буде ввести данни стенд Вибір споживачів на користь перевірених рішень

Визначені альтернативи були проаналізовані з точки зору термінів та ймовірності отримання ресурсів. Результати аналізу наведені у таблиці 5.13.

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Маркетингова кампанія для приваблювання покупців	Ймовірне	3 місяці
2	Безкоштовне впровадження лабораторного стенду в одного із покупців	Дуже ймовірне	1 місяць
3	Пошук підприємств та інститутів які хотіли би впровадити ІТ-інфраструктуру	Ймовірне	4 місяців
Обрана альтернатива: Безкоштовне впровадження лабораторного стенду в одного із покупців			

5.5 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів. Виконаний опис наведено у таблиці 5.14.

Таблиця 5.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Заклади освіти та науки	Висока	90%		Низькі бар’єри входу
2	Юридичні компанії	Середня	60%		Низькі бар’єри входу
3	Промислові підприємства	Середня	50%		Високі бар’єри входу
Які цільові групи обрано: Заклади освіти та науки					

За результатами аналізу потенційних груп споживачів (сегментів) була обрана цільова група – заклади освіти та науки (тобто установи державної форми власності), для впровадження у яких буде пропонуватись система, та визначили стратегію охоплення ринку, яка наведена у таблиці 5.15.

Таблиця 5.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Встановлення лабораторного стенду у закладах науки і освіти, а також у підприємствах які потребують хмарні IT-інфраструктури	Концентрований маркетинг	Низькі витрати Ефективна співпраця посередників	Стратегія диференціації

Обраною базовою стратегією розвитку є стратегія диференціації, яка передбачає надання товару важливих з точки зору споживача відмітних властивостей, які роблять товар відмінним від товарів конкурентів. Така відмінність базується на відсутності аналогічних рішень з точки зору інтеграції документообігу з засобами перекладу.

Виходячи з активного впровадження IT-сервісів у державі, було визначено базові стратегії конкурентної поведінки, наведені у таблиці 5.16.

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Так	Залучати нових	Модуль документообігу	Стратегія заняття конкурентної ніші.

Важливим при виході на ринок є впізнаваність бренду, чому сприяє формування комплексу асоціацій, за якими користувачі можуть однозначно ідентифікувати продукт.

Для формування ринкової позиції (комплексу асоціацій), за яким споживачі будуть ідентифікувати проект, було визначено стратегію позиціонування, наведену у таблиці 5.17.

Таблиця 5.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Простота у користуванні Інтегрованість документообігу з перекладом Допомога при створенні документів іншою мовою Пошук готових вріантів у існуючих документах (паралельних корпусах) Автоматичний переклад та транслітерація змінних реквізитів документу Верифікований та затверджений професійним перекладачем переклад	Стратегія диференціації	Формування регулярного попиту Збільшення разового використання послуги Виявлення нових груп споживачів Нові напрями застосування існуючої послуги	Багатомовний документ Документ в один клік Автоматичний переклад

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
	шаблону, що не потребує додаткових ревізій при використанні			

5.6 Розроблення маркетингової програми стартап-проекту

Першим кроком є формування маркетингової концепції товару, який отримає споживач. У таблиці 5.18 підсумовані результати попереднього аналізу конкурентоспроможності товару.

Таблиця 5.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Створення багатомовних	Гарантоване виявлення	Простота у користуванні Інтегрованість документообігу

Продовження таблиці 3.18

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
	документів	власника роботи Збереження первісного вигляду зображення Низькі ціни	з перекладом Допомога при створенні документів іншою мовою
2	Використання шаблонів працівниками, що слабо цільовою мовою	Оперативність пошуку плагіату Підтвердження порушення авторських прав	Пошук готових вріантів у існуючих документах (паралельних корпусах) Автоматичний переклад та транслітерація змінних реквізитів документу
3	Побудова еталонного джерела документів (golden source)	Простота зв’язку з автором роботи Можливість ознайомлення з іншими роботами автора	Верифікований та затверджений професійним перекладачем переклад шаблону, що не потребує додаткових ревізій при використанні

Далі за уточненими характеристиками продукту була розроблена трирівнева маркетингова модель товару, яка наведена у таблиці 5.19.

Таблиця 5.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
І. Товар за	Система багатомовного документообігу з підтримкою

Продовження таблиці 5.19

Рівні товару	Сутність та складові		
задумом	шаблонів		
	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Доступність 2. Зручність 3. Швидка обробка данних		
	Якість: коректність розгортання програми		
	До продажу: Навчання користувачів		
	Після продажу: реалізація додаткових сервісів		
За рахунок чого потенційний товар буде захищено від копіювання: підписанням умови про нерозголошення			

Далі були оцінені принципи ціноутворення на ринку, за результатами чого були визначені межі встановлення ціни, наведені у таблиці 5.20.

Таблиця 5.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	—	500 – 1500 грн	50000 грн	500 – 1500 грн

Розрахована оптимальна система збуту наведена у таблиці 5.21.

Таблиця 5.21 – Оптимальна система збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
	Закупівля здійснюється через відкриті тендери	Створення тендерної документації Інформування користувачів	Канал одного рівня	Селективна з використанням комбінованого каналу збуту

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфікою поведінки клієнтів. Результати розробки концепції маркетингових комунікацій наведені у таблиці 3.22.

Таблиця 5.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комуніка цій, якими користу ються цільові клієнти	Ключові позиції, обрані для позиціон ування	Завдання рекламного повідомлення	Концепц ія рекламн ого звернен ня
1	Ведення діяльності в державних	Прямі офіційні	Послідов ність в реалізації обраної	Формування у цільової аудиторії обізнаності про появу нового	Раціонал істична стратегі я

Продовження таблиці 5.22

№ п/п	Специфіка поведінки цільових клієнтів	Канали комуніка цій, якими корист ую ться цільові клієнти	Ключові позиції, обрані для позиціон ування	Завдання рекламного повідомлення	Концепц ія рекламн ого звернен ня
	структурах, а також на приватних підприємст вах		позиції Доступні сть та об’єктивн ість інформац ії про фірму і товар Унікальні сть послуги Відповідн ість вимогам тендеру	сервісу Інформування користувачів про властивості та переваги сервісу Інформування користувачів про нові способи використання відомого сервісу Пояснення цільовій аудиторії принципу дії послуги Виправити у користувачів неправильні представлення про продукт	реклами

Висновки

В результаті проведеної розробки стартап-проекту було отримано список очікувань користувачів стенду для проектування та управління хмарними IT інфраструктурами, були проведені технологічний аудит та аналіз ринкових можливостей запуску проекту, розроблені ринкова стратегія та маркетингова програма.

В цілому, можливість ринкової комерціалізації проекту є, наявний попит на такого роду системи, динаміка ринку позитивна – ринок зростає. Рентабельність роботи на ринку висока, проте ускладнена високою конкуренцією за кваліфіковані трудові ресурси.

Є перспективи впровадження в освітніх, наукових, промислових та юридичних організаціях різних форм власності. Незважаючи на високу конкуренцію і монополізацію ринку, проект є конкурентоспроможним. Проте через високі стартові капіталовкладення та високу трудомісткість більш перспективною є реалізація ідеї великими IT-компаніями України.

ВИСНОВКИ

В магістерській дисертації було створенну хмарну IT-інфраструктурі на основі OpenStack. В архітектурі OpenStack було реалізовано такі сервіси як: Nova, Neutron, Horizon, Heat, Cinder, Swift, Glance, Keystore.

Були розробленні структурна та функціональна схема отриманної IT-інфраструктури.

Був продемонстрований процес розгортання та управління хмарної IT-інфраструктури

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) [Електронний ресурс] – Режим доступу до ресурсу: https://www.openstack.org/marketplace/distros/.
2) [Електронний ресурс] – Режим доступу до ресурсу: https:// www.rdoproject.org/.
3) [Електронний ресурс] – Режим доступу до ресурсу: https://www.opnfv.org.
4) Итак, вы решили развернуть OpenStack [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/post/335530/.
5) System architecture [Електронний ресурс] – Режим доступу до ресурсу: https://docs.openstack.org/nova/pike/admin/arch.html.
6) Request Flow for Provisioning Instance in Openstack [Електронний ресурс] – Режим доступу до ресурсу: https://ilearnstack.com/2013/04/26/request-flow-for-provisioning-instance-in-openstack/.
7) OpenStack Neutron – The Dirty Network Detail [Електронний ресурс] – Режим доступу до ресурсу: https://packetpushers.net/openstack-neutron/.
8) How to Create, Deploy and Launch Virtual Machines in OpenStack [Електронний ресурс] – Режим доступу до ресурсу: https://www.tecmint.com/create-deploy-and-launch-virtual-machines-in-openstack/.
9) OpenStack [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/OpenStack.
10) Пример архитектур [Електронний ресурс] – Режим доступу до ресурсу: http://onreader.mdl.ru/openstack-ops/content/example_architecture.html.
11) Включение Ceph в состав OpenStack [Електронний ресурс] – Режим доступу до ресурсу: http://onreader.mdl.ru/LearningCeph/content/Ch09.html.

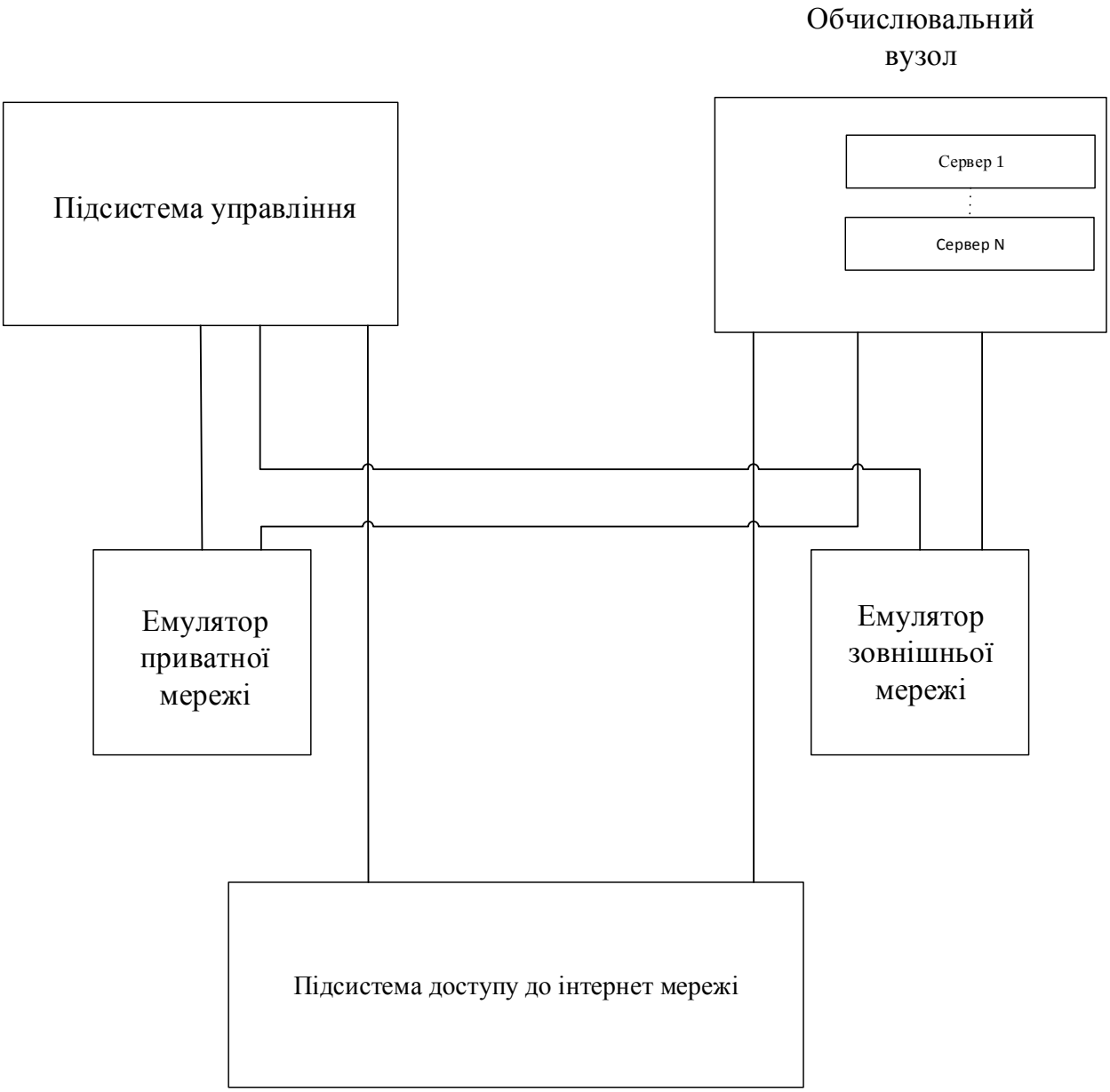
12) Подготовка сети для OpenStack [Електронний ресурс] – Режим доступу до ресурсу: http://onreader.mdl.ru/LearningOpenStackNetworking/content/Ch01.html.

13) Forrester Research 2017 [Електронний ресурс] – Режим доступу до ресурсу: https://www.forrester.com/Forrester+Research+Reports+2017+FourthQuarter+And+FullYear+Financial+Results/-/E-PRE10205.

14) [Електронний ресурс] – Режим доступу до ресурсу: http://tarballs.openstack.org.

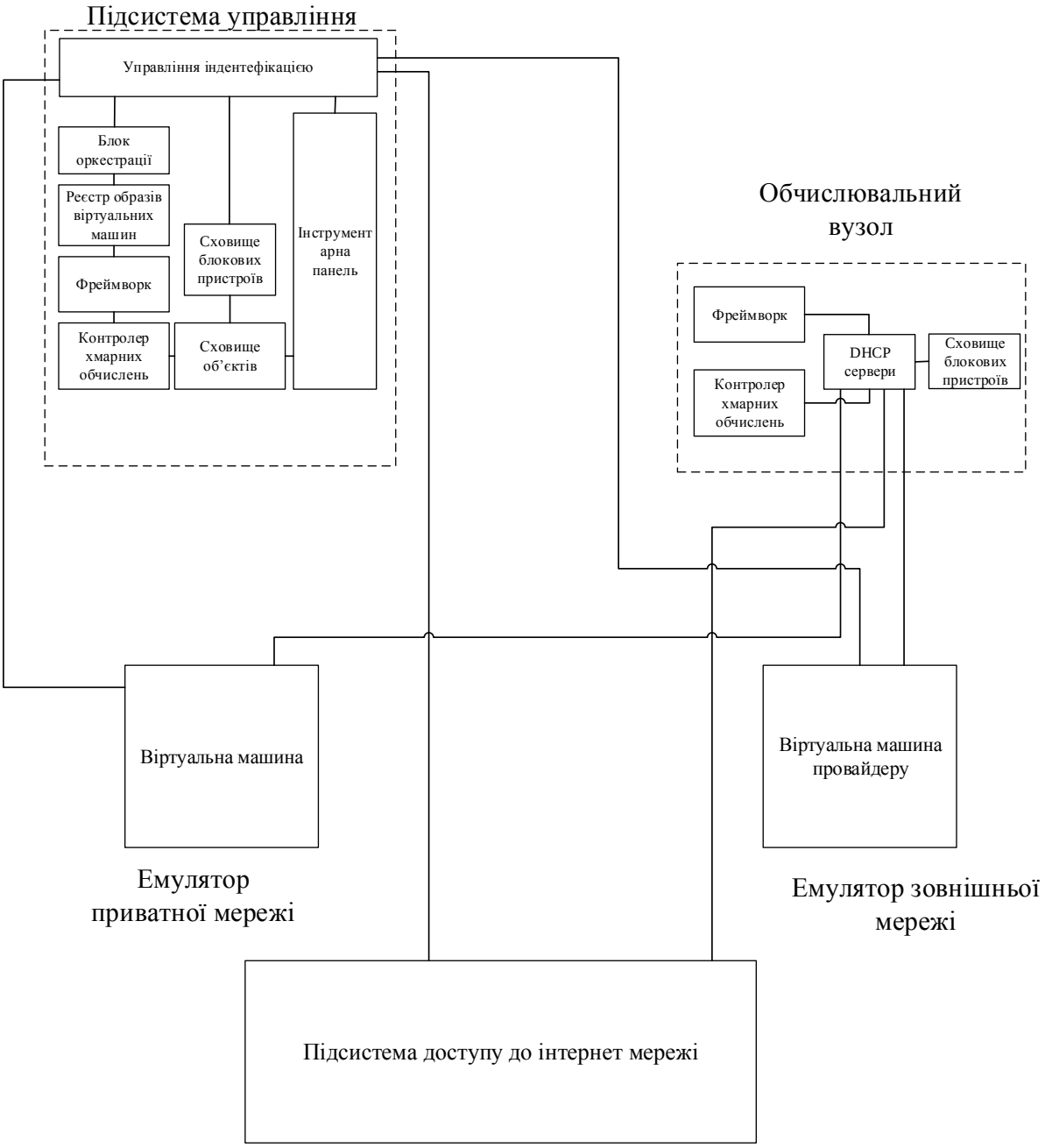
ДОДАТОК А

Структурная схема хмарної IT-інфраструктури



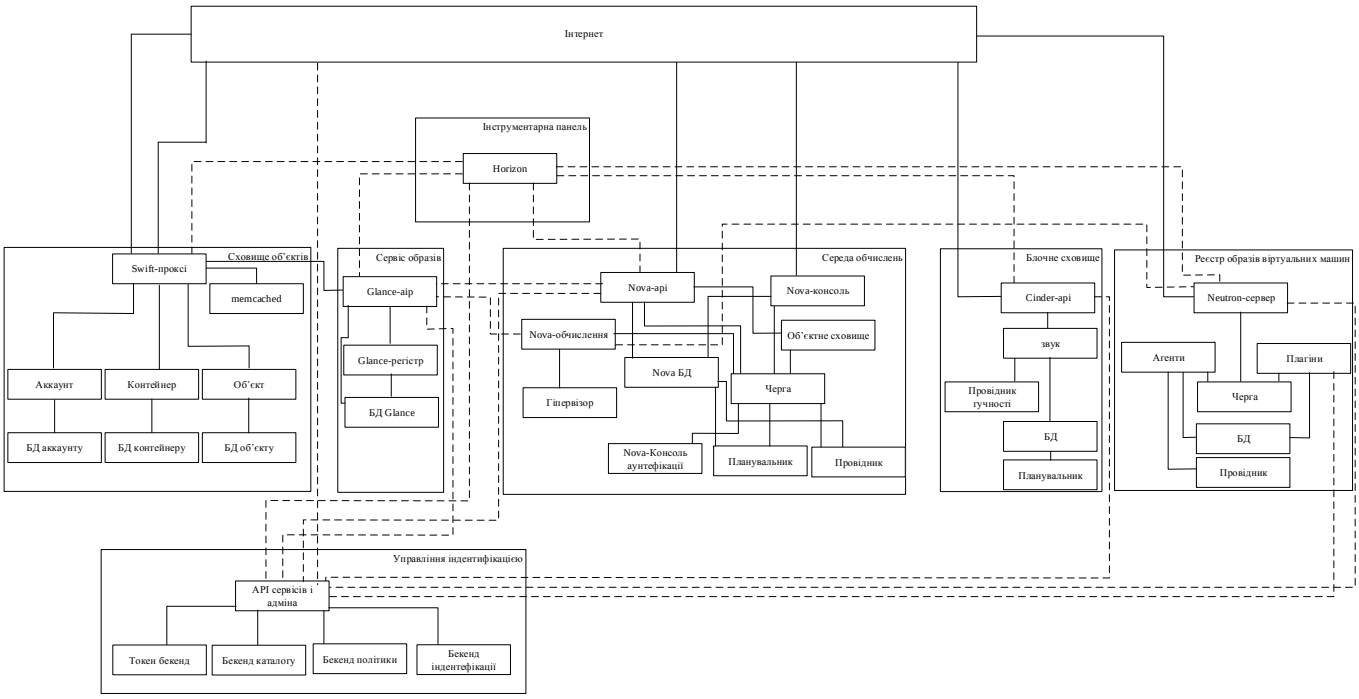
ДОДАТОК Б

Функіональна схема хмарної ІТ-інфраструктури



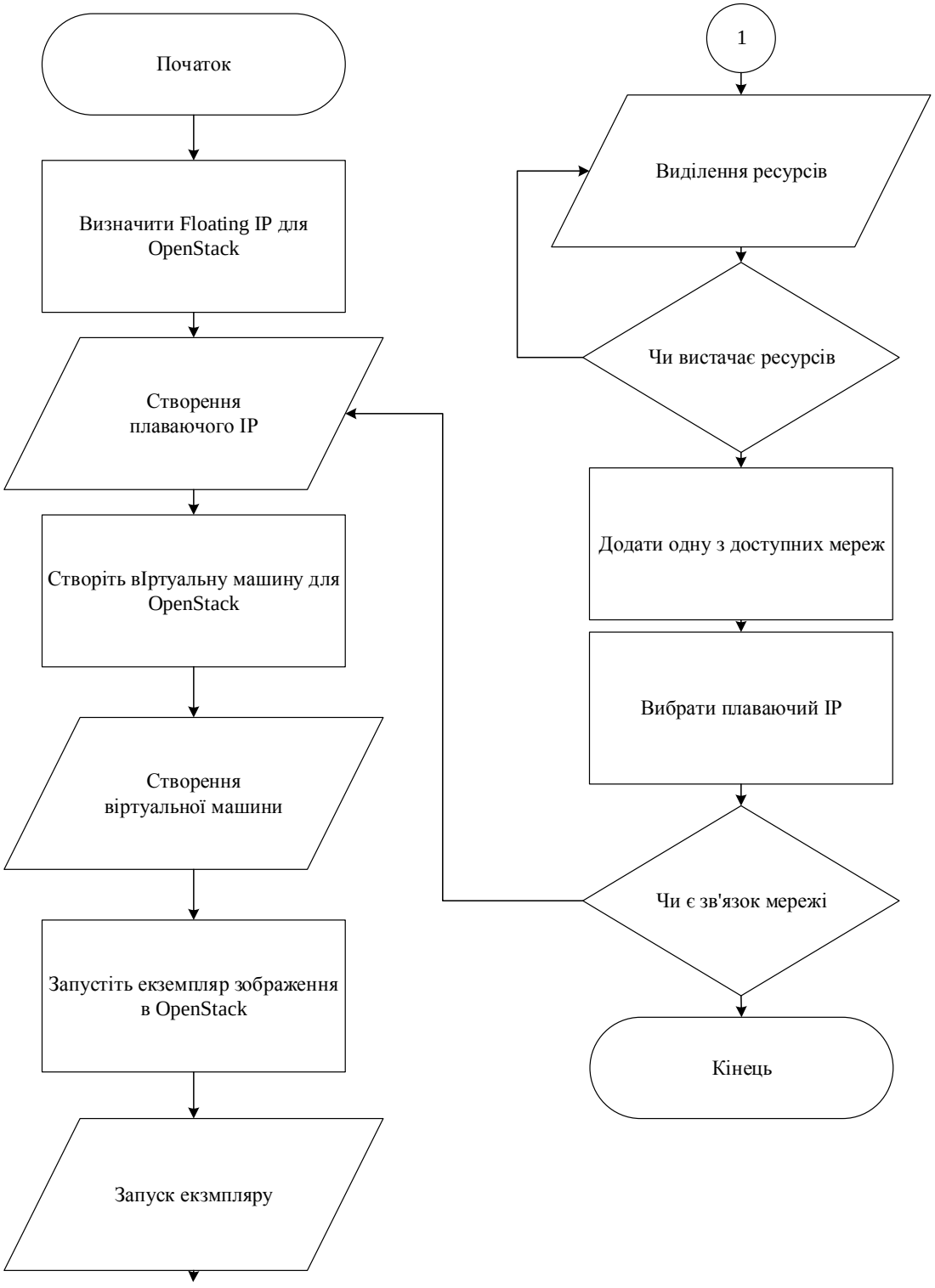
ДОДАТОК В

Схема взаємозв’язку між сервісами OpenStack



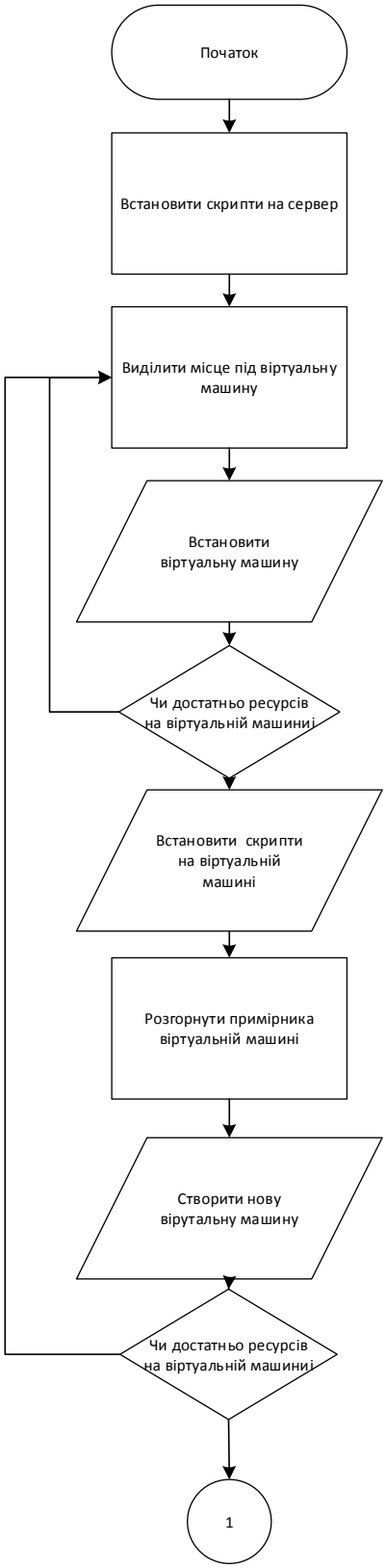
ДОДАТОК Г

Алгоритм створення і запуску віртуальної машини в OpenStack



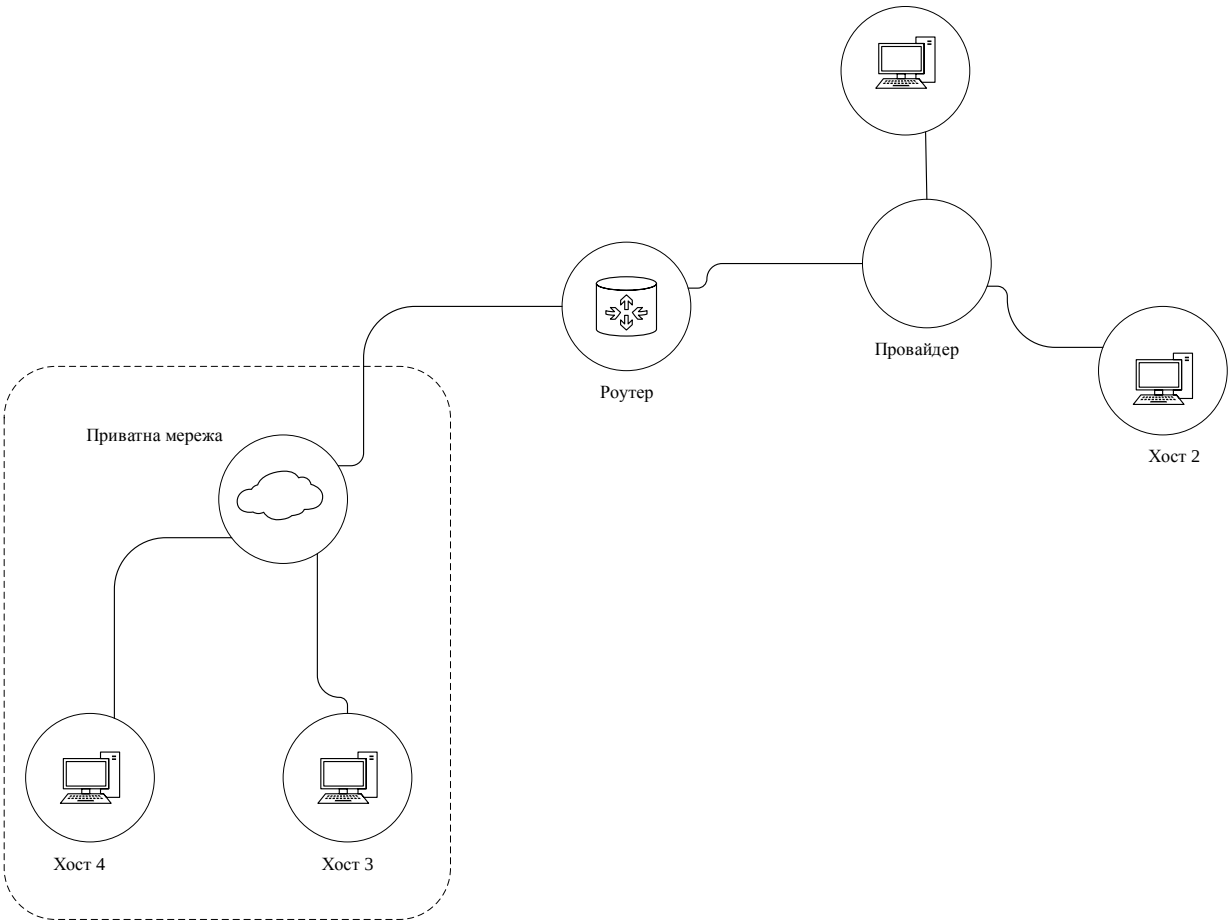
ДОДАТОК Д

Алгоритм побудови лабораторного стенду



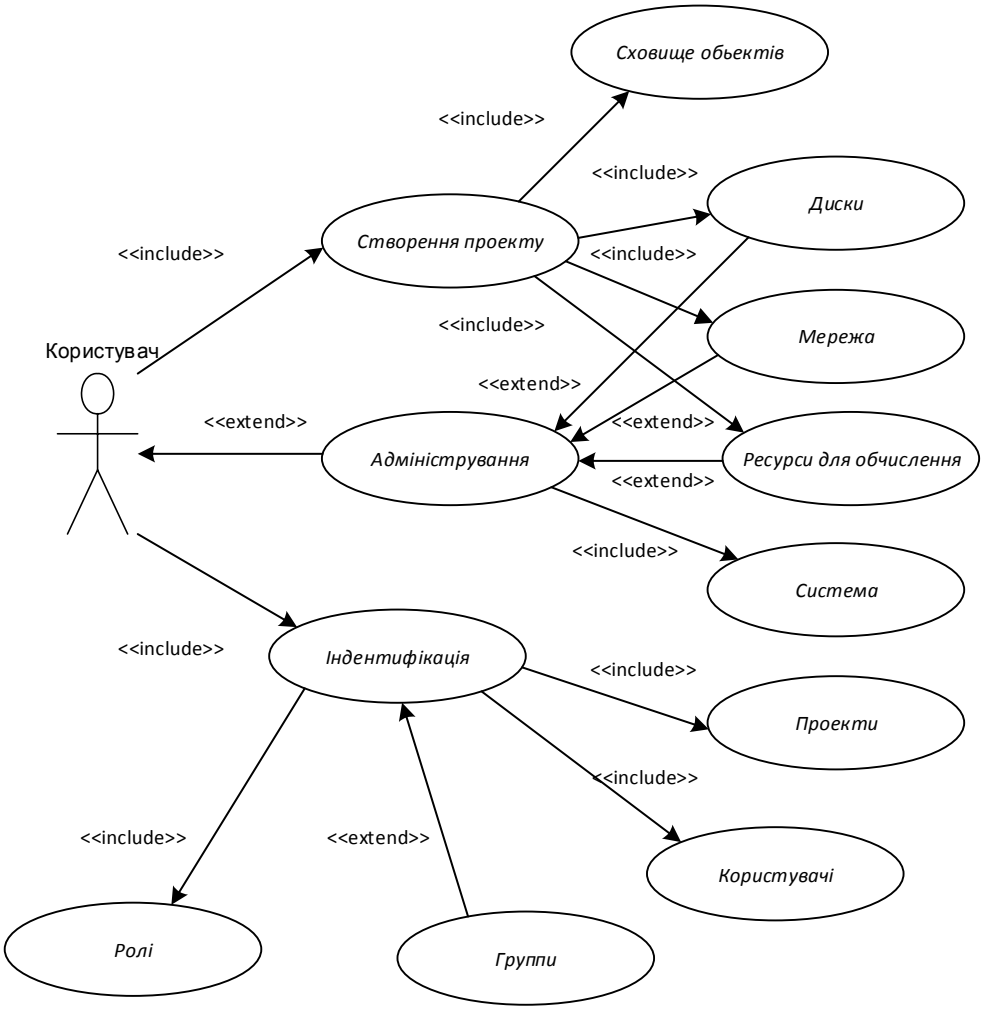
ДОДАТОК Е

Мережеві графи лабораторнго стєнду



ДОДАТОК Ж

UML Діаграма адміністрування OpenStack



ДОДАТОК К

Блок-схема віртуалізації KVM